



TheFellowshipOfTheCode

Definizione di Prodotto

Informazioni sul documento

Nome Documento	DefinizioneDiProdotto_v2_0_0.pdf
Versione	2
Data di Creazione	23 Febbraio 2016
Data ultima modifica	10 Aprile 2016
Stato	Approvato
Redazione	Franco Berton Alberto Ferrara Matteo Gnoato Matteo Granzotto Simone Magagna Marco Prelaz Mattia Varotto Matteo Granzotto
Verifica	Marco Prelaz Mattia Varotto Alberto Ferrara
Approvazione	Esterno
Uso	TheFellowshipOfTheCode
Distribuzione	Prof. Tullio Vardanega, Prof. Riccardo Cardin, Zucchetti S.P.A.
Destinato a	
Email di riferimento	thefellowshipofthecode@gmail.com

Sommario

Questo documento descrive la progettazione di dettaglio delle componenti definita dal gruppo TheFellowshipOfTheCode relativa al progetto QuizziPedia.



Registro delle modifiche

Versione	Descrizione	Autore e Ruolo	Data
2.0.0	Approvazione Documento	Alberto Ferrara Responsabile	2016-04-10
1.5.0	Verifica Documento	Marco Prelaz Verificatore	2016-04-09
1.4.19	Stesura §E algoritmo di selezione della domanda	Mattia Varotto Progettista	2016-04-08
1.4.18	Stesura §D.4 statistiche questionario	Marco Prelaz Progettista	2016-04-08
1.4.17	Stesura §D.3 statistiche argomento	Matteo Gnoato Progettista	2016-04-08
1.4.16	Stesura §D.2 statistiche domanda	Mattia Varotto Progettista	2016-04-08
1.4.15	Stesura §D.1.2 statistiche utente - utente non autenticato	Matteo Gnoato Progettista	2016-04-08
1.4.14	Stesura §D.1.1 statistiche utente - utente autenticato	Marco Prelaz Progettista	2016-04-08
1.4.13	Stesura §D.1 statistiche utente	Franco Berton Progettista	2016-04-08
1.4.12	Stesura §D Calcolo delle statistiche	Mattia Varotto Progettista	2016-04-08
1.4.11	Stesura §B.2.6 JSON domanda ad area cliccabile	Matteo Gnoato Progettista	2016-04-08
1.4.10	Stesura §B.2.5 JSON domanda a collegamento di elementi	Franco Berton Progettista	2016-04-08
1.4.9	Stesura §B.2.4 JSON domanda a ordinamento immagini	Mattia Varotto Progettista	2016-04-08
1.4.8	Stesura §B.2.3 JSON domanda a ordinamento di stringhe	Marco Prelaz Progettista	2016-04-08
1.4.7	Stesura §B.2.2 JSON domanda a risposta multipla	Mattia Varotto Progettista	2016-04-08
1.4.6	Stesura §B.2.1 JSON domanda a vero/falso	Marco Prelaz Progettista	2016-04-08
1.4.5	Stesura §B.2 generazione del JSON	Franco Berton Progettista	2016-04-08
1.4.4	Stesura §B.1.7 sintassi domanda a riempimento spazi vuoti	Franco Berton Progettista	2016-04-08
1.4.3	Stesura §B.1.6 sintassi domanda ad area cliccabile	Matteo Gnoato Progettista	2016-04-08



Versione	Descrizione	Autore e Ruolo	Data
1.4.2	Stesura §B.1.5 sintassi domanda a collegamento di elementi	Matteo Gnoato Progettista	2016-04-08
1.4.1	Stesura §B.1.4 sintassi domanda a ordinamento immagini	Matteo Gnoato Progettista	2016-04-08
1.4.0	Verifica Documento	Matteo Granzotto Verificatore	2016-04-08
1.3.18	Stesura §B.1.3 sintassi domanda a ordinamento di stringhe	Mattia Varotto Progettista	2016-04-08
1.3.17	Stesura §B.1.2 sintassi domanda a risposta multipla	Matteo Gnoato Progettista	2016-04-08
1.3.16	Stesura §B.1.1 sintassi domanda vero/falso	Franco Berton Progettista	2016-04-08
1.3.15	Stesura §B.1 definizione della grammatica	Matteo Gnoato Progettista	2016-04-07
1.3.14	Stesura Appendice B: QML - Quiz Markup Language	Mattia Varotto Progettista	2016-04-07
1.3.13	Stesura Diagramma di sequenza: Back-End, PUT /:lang/user/training/userlevelupdate	Marco Prelaz Progettista	2016-04-07
1.3.12	Stesura Diagramma di sequenza: Back-End, PUT /:lang/user/:userId/training/questionStatistics	Franco Berton Progettista	2016-04-07
1.3.11	Stesura Diagramma di sequenza: Back-End, PUT /:lang/user/:userId/training/userStatistics	Franco Berton Progettista	2016-04-07
1.3.10	Stesura Diagramma di sequenza: Back-End, GET /:lang/user/:userId/training/question	Mattia Varotto Progettista	2016-04-07
1.3.9	Stesura Diagramma di sequenza: Back-End, GET /:lang/user/:userId/search/quizzes/:quizId	Matteo Gnoato Progettista	2016-04-07
1.3.8	Stesura Diagramma di sequenza: Back-End, GET /:lang/user/:userId/search/users/:users	Mattia Varotto Progettista	2016-04-07
1.3.7	Stesura Diagramma di sequenza: Back-End, GET /:lang/user/:userId/search/:keyword/quizzes	Marco Prelaz Progettista	2016-04-07
1.3.6	Stesura Diagramma di sequenza: Back-End, GET /:lang/user/:userId/search/:keyword/users	Marco Prelaz Progettista	2016-04-07
1.3.5	Stesura Diagramma di sequenza: Back-End, POST /:lang/user/:userId/question	Marco Prelaz Progettista	2016-04-07
1.3.4	Stesura Diagramma di sequenza: Back-End, GET /:lang/user/:userId/question/:questionId	Mattia Varotto Progettista	2016-04-07



Versione	Descrizione	Autore e Ruolo	Data
1.3.3	Stesura Diagramma di sequenza: GET /:lang/user/:userId/question	Back-End, Progettista	Matteo Gnoato 2016-04-07
1.3.2	Stesura Diagramma di se- quenza: Back-End, POST /:lang/user/:userId/quiz/:quizId/summary	di se- Back-End, POST	Matteo Gnoato 2016-04-07 Progettista
1.3.1	Stesura Diagramma di sequenza: GET /:lang/user/:userId/quiz/:quizId/test	Back-End, Progettista	Franco Berton 2016-04-07
1.2.19	Verifica Documento	Mattia Varotto Verificatore	2016-04-06
1.2.18	Stesura Diagramma di sequenza: PUT /:lang/user/:userId/quiz/:quizId	Back-End, Progettista	Matteo Gnoato 2016-04-05
1.2.17	Stesura Diagramma di se- quenza: Back-End, POST /:lang/user/:userId/quiz/:quizId/activeUser	di se- Back-End, POST	Franco Berton 2016-04-05 Progettista
1.2.16	Stesura Diagramma di se- quenza: Back-End, POST /:lang/user/:userId/quiz/:quizId/addUser	Back-End, Progettista	Mattia Varotto 2016-04-05
1.2.15	Stesura Diagramma di se- quenza: Back-End, PUT /:lang/user/:userId/quiz/:quizId/removeUser	di se- PUT	Franco Berton 2016-04-05 Progettista
1.2.14	Stesura Diagramma di sequenza: POST /:lang/user/:userId/quiz	Back-End, Progettista	Matteo Gnoato 2016-04-05
1.2.13	Stesura Diagramma di sequenza: GET /:lang/user/:userId/quiz/:quizId	Back-End, Progettista	Mattia Varotto 2016-04-05
1.2.12	Stesura Diagramma di sequenza: GET /:lang/user/:userId/quiz	Back-End, Progettista	Matteo Gnoato 2016-04-05
1.2.11	Stesura Diagramma di se- quenza: Back-End, GET /:lang/user/:userId/statistics/summary/:summaryId	di se- GET	Mattia Varotto 2016-04-05 Progettista
1.2.10	Stesura Diagramma di sequenza: GET /:lang/user/:userId/statistic/summary/	Back-End, Progettista	Franco Berton 2016-04-01
1.2.9	Stesura Diagramma di sequenza: GET /:lang/user/:userId/statistic	Back-End, Progettista	Marco Prelaz 2016-04-01
1.2.8	Stesura Diagramma di sequenza: PUT /:lang/user/:userId/privacy	Back-End, Progettista	Marco Prelaz 2016-04-01
1.2.7	Stesura Diagramma di sequenza: PUT /:lang/user/:userId	Back-End, Progettista	Franco Berton 2016-04-01
1.2.6	Stesura Diagramma di sequenza: GET /:lang/user/:userId	Back-End, Progettista	Mattia Varotto 2016-04-01
1.2.5	Stesura Diagramma di sequenza: DELETE /:lang/user/:userId	Back-End, Progettista	Mattia Varotto 2016-03-31



Versione	Descrizione	Autore e Ruolo	Data
1.2.4	Stesura Diagramma di sequenza: Back-End, POST /:lang/recovery	Matteo Gnoato Progettista	2016-03-31
1.2.3	Stesura Diagramma di sequenza: Back-End, GET /:lang/loggedin	Franco Berton Progettista	2016-03-31
1.2.2	Stesura Diagramma di sequenza: Back-End, POST /:lang/signout	Matteo Gnoato Progettista	2016-03-31
1.2.1	Stesura Diagramma di sequenza: Back-End, POST /:lang/signin	Marco Prelaz Progettista	2016-03-31
1.2.0	Verifica Documento	Marco Prelaz Verificatore	2016-03-30
1.1.18	Stesura Diagramma di sequenza: Back-End, POST /:lang/singup	Matteo Gnoato Progettista	2016-03-29
1.1.17	Stesura Diagramma di sequenza: Back-End, GET /:lang	Franco Berton Progettista	2016-03-29
1.1.16	Stesura Diagramma di sequenza: Back-End, gestione generale delle richieste	Mattia Varotto Progettista	2016-03-29
1.1.15	Stesura Diagramma di sequenza: Operazioni Front-End Modalità allenamento, Risposta a una domanda	Alberto Ferrara Progettista	2016-03-29
1.1.14	Stesura Diagramma di sequenza: Operazioni Front-End Modalità allenamento, Composizione di una domanda	Matteo Granzotto Progettista	2016-03-29
1.1.13	Stesura Diagramma di sequenza: Operazioni Front-End Modalità allenamento, download di una domanda	Matteo Granzotto Progettista	2016-03-29
1.1.12	Stesura Diagramma di sequenza: Operazioni Front-End, Modalità allenamento, selezione parametri di set-up	Alberto Ferrara Progettista	2016-03-29
1.1.11	Stesura Diagramma di sequenza: Operazioni Front-End, Modalità allenamento	Simone Magagna Progettista	2016-03-29
1.1.10	Stesura Diagramma di sequenza: Operazioni Front-End, Creazione Domande	Alberto Ferrara Progettista	2016-03-29
1.1.9	Stesura Diagramma di sequenza: Operazioni Front-End, Ricerca, Redirect alla pagina di un utente	Alberto Ferrara Progettista	2016-03-29
1.1.8	Stesura Diagramma di sequenza: Operazioni Front-End, Ricerca, Iscrizione ad un questionario	Matteo Granzotto Progettista	2016-03-29
1.1.7	Stesura Diagramma di sequenza: Operazioni Front-End, Ricerca, ricerca	Simone Magagna Progettista	2016-03-29
1.1.6	Stesura Diagramma di sequenza: Operazioni Front-End, Ricerca	Simone Magagna Progettista	2016-03-29



Versione	Descrizione	Autore e Ruolo	Data
1.1.5	Stesura Diagramma di sequenza: Operazioni Front-End	Matteo Granzotto Progettista	2016-03-29
1.1.4	Stesura QuizziPedia::Front-End::ModelViews ::UserDetailsModelView	Matteo Granzotto Progettista	2016-03-28
1.1.3	Stesura QuizziPedia::Front-End::ModelViews ::TrueFalseQuestionsModelView	Alberto Ferrara Progettista	2016-03-28
1.1.2	Stesura QuizziPedia::Front-End::ModelViews ::TrainingModelView	Alberto Ferrara Progettista	2016-03-28
1.1.1	Stesura QuizziPedia::Front-End::ModelViews ::TopicKeywordsModelView	Simone Magagna Progettista	2016-03-28
1.1.0	Verifica Documento	Matteo Granzotto Verificatore	2016-03-27
1.0.18	Stesura QuizziPedia::Front-End::ModelViews ::StringSortingQuestionsModelView	Alberto Ferrara Progettista	2016-03-26
1.0.17	Stesura QuizziPedia::Front-End::ModelViews ::StatisticsModelView	Simone Magagna Progettista	2016-03-26
1.0.16	Stesura QuizziPedia::Front-End::ModelViews ::SignUpModelView	Matteo Granzotto Progettista	2016-03-26
1.0.15	Stesura QuizziPedia::Front-End::ModelViews ::ResultsOfQuizzesModelView	Alberto Ferrara Progettista	2016-03-26
1.0.14	Stesura QuizziPedia::Front-End::ModelViews ::ResultQuestionnaireModelView	Simone Magagna Progettista	2016-03-25
1.0.13	Stesura QuizziPedia::Front-End::ModelViews ::ResultModelView	Simone Magagna Progettista	2016-03-25
1.0.12	Stesura QuizziPedia::Front-End::ModelViews ::RegistrationQuestionnaireModelView	Simone Magagna Progettista	2016-03-25
1.0.11	Stesura QuizziPedia::Front-End::ModelViews ::QuizEventModelView	Alberto Ferrara Progettista	2016-03-25
1.0.10	Stesura QuizziPedia::Front-End::ModelViews ::QuestionsModelView	Alberto Ferrara Progettista	2016-03-24
1.0.9	Stesura QuizziPedia::Front-End::ModelViews ::QuestionsManagementModelView	Alberto Ferrara Progettista	2016-03-24
1.0.8	Stesura QuizziPedia::Front-End::ModelViews ::QuestionnaireDetailsModelView	Simone Magagna Progettista	2016-03-24
1.0.7	Stesura QuizziPedia::Front-End::ModelViews ::ProfileManagementModelView	Matteo Granzotto Progettista	2016-03-24
1.0.6	Stesura QuizziPedia::Front-End::ModelViews ::PasswordForgotModelView	Matteo Granzotto Progettista	2016-03-23
1.0.5	Stesura QuizziPedia::Front-End::ModelViews ::MultipleQuestionModelView	Alberto Ferrara Progettista	2016-03-23



Versione	Descrizione	Autore e Ruolo	Data
1.0.4	Stesura QuizziPedia::Front-End::ModelViews ::MenuBarModelView	Simone Magagna Progettista	2016-03-23
1.0.3	Stesura QuizziPedia::Front-End::ModelViews ::LoginModelView	Alberto Ferrara Progettista	2016-03-23
1.0.2	Stesura QuizziPedia::Front-End::ModelViews ::ImagesSortingQuestionsModelView	Matteo Granzotto Progettista	2016-03-23
1.0.1	Stesura QuizziPedia::Front-End::ModelViews ::HomeModelView	Simone Magagna Progettista	2016-03-21
1.0.0	Approvazione Documento	Simone Magagna Responsabile	2016-03-20
0.7.0	Verifica Documento	Alberto Ferrara Verificatore	2016-03-19
0.7.17	Stesura QuizziPedia::Front-End::ModelViews ::FillingQuestionsModelView	Matteo Granzotto Progettista	2016-03-19
0.7.16	Stesura QuizziPedia::Front-End::ModelViews ::FillingQuestionnaireModelView	Simone Magagna Progettista	2016-03-19
0.7.15	Stesura QuizziPedia::Front-End::ModelViews ::EditorQMLModelView	Alberto Ferrara Progettista	2016-03-19
0.7.14	Stesura QuizziPedia::Front-End::ModelViews ::CreateQuestionnaireModelView	Simone Magagna Progettista	2016-03-19
0.7.13	Stesura QuizziPedia::Front-End::ModelViews ::ConnectionQuestionsModelView	Matteo Granzotto Progettista	2016-03-19
0.7.12	Stesura QuizziPedia::Front-End::ModelViews ::ClickableAreaQuestionsModelView	Simone Magagna Progettista	2016-03-19
0.7.11	Stesura QuizziPedia::Front-End::ModelViews, informazioni generali	Alberto Ferrara Progettista	2016-03-18
0.7.10	Stesura QuizziPedia::Front-End::Views ::RegistrationManagementView	Alberto Ferrara Progettista	2016-03-18
0.7.9	Stesura QuizziPedia::Front-End::Views ::ResultsQuestionnaireView	Simone Magagna Progettista	2016-03-18
0.7.8	Stesura QuizziPedia::Front-End::Views ::CreateQuestionnaireView	Simone Magagna Progettista	2016-03-18
0.7.7	Stesura QuizziPedia::Front-End::Views ::QuestionnaireManagementView	Alberto Ferrara Progettista	2016-03-17
0.7.6	Stesura QuizziPedia::Front-End::Views ::FillingQuestionView	Alberto Ferrara Progettista	2016-03-17
0.7.5	Stesura QuizziPedia::Front-End::Views ::TrainingView	Matteo Granzotto Progettista	2016-03-17
0.7.4	Stesura QuizziPedia::Front-End::Views ::EditorQMLView	Matteo Granzotto Progettista	2016-03-16



Versione	Descrizione	Autore e Ruolo	Data
0.7.3	Stesura QuizziPedia::Front-End::Views ::ClickableAreaQuestionView	Matteo Granzotto Progettista	2016-03-16
0.7.2	Stesura QuizziPedia::Front-End::Views ::FillingQuestionsView	Alberto Ferrara Progettista	2016-03-16
0.7.1	Stesura QuizziPedia::Front-End::Views ::StringsSortingQuestionsView	Simone Magagna Progettista	2016-03-16
0.7.0	Verifica Documento	Matteo Gnoato Verificatore	2016-03-15
0.6.19	Stesura QuizziPedia::Front-End::Views ::ImagesSortingQuestionsView	Simone Magagna Progettista	2016-03-15
0.6.18	Stesura QuizziPedia::Front-End::Views ::ConnectionQuestionsView	Matteo Granzotto Progettista	2016-03-15
0.6.17	Stesura QuizziPedia::Front-End::Views ::MultipleQuestionView	Matteo Granzotto Progettista	2016-03-14
0.6.16	Stesura QuizziPedia::Front-End::Views ::True-FalseQuestionView	Alberto Ferrara Progettista	2016-03-14
0.6.15	Stesura QuizziPedia::Front-End::Views ::QuestionsManagementView	Simone Magagna Progettista	2016-03-14
0.6.14	Stesura QuizziPedia::Front-End::Views ::ProfileManagementView	Simone Magagna Progettista	2016-03-14
0.6.13	Stesura QuizziPedia::Front-End::Views ::OtherUserView	Alberto Ferrara Progettista	2016-03-14
0.6.12	Stesura QuizziPedia::Front-End::Views ::UserView	Matteo Granzotto Progettista	2016-03-13
0.6.11	Stesura QuizziPedia::Front-End::Views ::ResultView	Matteo Granzotto Progettista	2016-03-13
0.6.10	Stesura QuizziPedia::Front-End::Views ::HomeView	Matteo Granzotto Progettista	2016-03-13
0.6.9	Stesura QuizziPedia::Front-End::Views ::PasswordForgotView	Matteo Granzotto Progettista	2016-03-13
0.6.8	Stesura QuizziPedia::Front-End::Views ::SignUpView	Simone Magagna Progettista	2016-03-13
0.6.7	Stesura QuizziPedia::Front-End::Views ::LoginView	Simone Magagna Progettista	2016-03-13
0.6.6	Stesura QuizziPedia::Front-End::Views, informazioni generali	Simone Magagna Progettista	2016-03-12
0.6.5	Stesura QuizziPedia::Front-End::Services	Alberto Ferrara Progettista	2016-03-12



Versione	Descrizione	Autore e Ruolo	Data
0.6.4	Stesura QuizziPedia::Front-End::Services ::UserDetailsService	Alberto Ferrara Progettista	2016-03-12
0.6.3	Stesura QuizziPedia::Front-End::Services ::StatisticsService	Alberto Ferrara Progettista	2016-03-12
0.6.2	Stesura QuizziPedia::Front-End::Services ::SearchService	Simone Magagna Progettista	2016-03-12
0.6.1	Stesura QuizziPedia::Front-End::Services ::QuizService	Matteo Granzotto Progettista	2016-03-12
0.6.0	Verifica Documento	Franco Berton Verificatore	2016-03-11
0.5.18	Stesura QuizziPedia::Front-End::Services ::QuestionService	Simone Magagna Progettista	2016-03-11
0.5.17	Stesura QuizziPedia::Back-End::App::Router ::LangRouter	Matteo Gnoato Progettista	2016-03-11
0.5.16	Stesura QuizziPedia::Front-End::Services ::LangService	Matteo Granzotto Progettista	2016-03-11
0.5.15	Stesura QuizziPedia::Front-End::Services ::AuthService	Alberto Ferrara Progettista	2016-03-11
0.5.14	Stesura QuizziPedia::Front-End::Services, informazioni generali	Simone Magagna Progettista	2016-03-11
0.5.13	Stesura QuizziPedia::Front-End::Directives ::TrueFalseAnswerDirective	Simone Magagna Progettista	2016-03-11
0.5.12	Stesura QuizziPedia::Front-End::Directives ::TrainingSetUpDirective	Matteo Granzotto Progettista	2016-03-11
0.5.11	Stesura QuizziPedia::Front-End::Directives ::SortTextAnswerDirective	Alberto Ferrara Progettista	2016-03-10
0.5.10	Stesura QuizziPedia::Front-End::Directives ::SortImagesAnswerDirective	Alberto Ferrara Progettista	2016-03-10
0.5.9	Stesura QuizziPedia::Front-End::Directives ::MultipleChoiceAnswerDirective	Simone Magagna Progettista	2016-03-10
0.5.8	Stesura QuizziPedia::Front-End::Directives ::LinkingAnswerDirective	Alberto Ferrara Progettista	2016-03-10
0.5.7	Stesura QuizziPedia::Front-End::Directives ::HeaderTextQuestionDirective	Matteo Granzotto Progettista	2016-03-09
0.5.6	Stesura QuizziPedia::Front-End::Directives ::EmptySpaceAnswerDirective	Matteo Granzotto Progettista	2016-03-09
0.5.5	Stesura QuizziPedia::Front-End::Directives ::ClickableAnswerDirective	Simone Magagna Progettista	2016-03-09
0.5.4	Stesura QuizziPedia::Front-End::Directives ::UserResultsDirective	Alberto Ferrara Progettista	2016-03-09



Versione	Descrizione	Autore e Ruolo	Data
0.5.3	Stesura QuizziPedia::Front-End::Directives ::UserDetailsDirective	Alberto Ferrara Progettista	2016-03-09
0.5.2	Stesura QuizziPedia::Front-End::Directives ::TopicKeywordsDirective	Simone Magagna Progettista	2016-03-09
0.5.1	Stesura QuizziPedia::Front-End::Directives ::SubscribeResultDirective	Matteo Granzotto Progettista	2016-03-09
0.5.0	Verifica Documento	Alberto Ferrara Verificatore	2016-03-08
0.4.18	Stesura QuizziPedia::Front-End::Directives ::StatisticDirective	Matteo Granzotto Progettista	2016-03-08
0.4.17	Stesura QuizziPedia::Front-End::Directives ::SearchDirective	Simone Magagna Progettista	2016-03-08
0.4.16	Stesura QuizziPedia::Front-End::Directives ::QuestionnaireResultDirective	Alberto Ferrara Progettista	2016-03-08
0.4.15	Stesura QuizziPedia::Back-End::App::Router ::QuizRouter	Marco Prelaz Progettista	2016-03-08
0.4.14	Stesura QuizziPedia::Front-End::Directives ::QuestionnaireDoneDetailsDirective	Matteo Granzotto Progettista	2016-03-08
0.4.13	Stesura QuizziPedia::Front-End::Directives ::QuestionnaireDetailsDirective	Simone Magagna Progettista	2016-03-08
0.4.12	Stesura QuizziPedia::Back-End::App::Router ::QuestionRouter	Mattia Varotto Progettista	2016-03-07
0.4.11	Stesura QuizziPedia::Front-End::Directives ::NewQuestionButtonDirective	Matteo Granzotto Progettista	2016-03-07
0.4.10	Stesura QuizziPedia::Front-End::Directives ::OneQuestionDirective	Alberto Ferrara Progettista	2016-03-07
0.4.9	Stesura QuizziPedia::Back-End::App::Router ::UserRouter	Franco Berton Progettista	2016-03-07
0.4.8	Stesura QuizziPedia::Front-End::Directives ::MenuBarDirective	Matteo Granzotto Progettista	2016-03-07
0.4.7	Stesura QuizziPedia::Front-End::Directives ::ImageInTheQuestionDirective	Simone Magagna Progettista	2016-03-07
0.4.6	Stesura QuizziPedia::Back-End::App::Router	Marco Prelaz Progettista	2016-03-07
0.4.5	Stesura QuizziPedia::Back-End::App::Controllers::NotFoundHandler	Marco Prelaz Progettista	2016-03-07
0.4.4	Stesura QuizziPedia::Front-End::Directives ::FooterDirective	Alberto Ferrara Progettista	2016-03-07



Versione	Descrizione	Autore e Ruolo	Data
0.4.3	Stesura QuizziPedia::Front-End::Directives ::ExamModalityDirective	Alberto Ferrara Progettista	2016-03-07
0.4.2	Stesura QuizziPedia::Front-End::Directives ::EliminationAndModifyDirective	Simone Magagna Progettista	2016-03-07
0.4.1	Stesura QuizziPedia::Back-End::App::Controllers::LangController	Matteo Gnoato Progettista	2016-03-07
0.4.0	Verifica Documento	Matteo Gnoato Verificatore	2016-03-06
0.3.17	Stesura QuizziPedia::Back-End::App::Controllers::SummaryController	Mattia Varotto Progettista	2016-03-06
0.3.16	Stesura QuizziPedia::Front-End::Controllers ::UserDetailsController	Matteo Granzotto Progettista	2016-03-06
0.3.15	Stesura QuizziPedia::Front-End::Controllers ::TrueFalseQuestionController	Simone Magagna Progettista	2016-03-06
0.3.14	Stesura QuizziPedia::Back-End::App::Controllers ::TopicController	Mattia Varotto Progettista	2016-03-06
0.3.13	Stesura QuizziPedia::Front-End::Controllers ::TrainingController	Matteo Granzotto Progettista	2016-03-05
0.3.12	Stesura QuizziPedia::Front-End::Controllers ::TopicKeywordsController	Matteo Granzotto Progettista	2016-03-05
0.3.11	Stesura QuizziPedia::Front-End::Controllers ::StringsSortingQuestionsController	Alberto Ferrara Progettista	2016-03-05
0.3.10	Stesura QuizziPedia::Back-End::App::Controllers ::QuizController	Marco Prelaz Progettista	2016-03-05
0.3.9	Stesura QuizziPedia::Front-End::Controllers ::StatisticsController	Simone Magagna Progettista	2016-03-05
0.3.8	Stesura QuizziPedia::Front-End::Controllers ::SignUpController	Matteo Granzotto Progettista	2016-03-05
0.3.7	Stesura QuizziPedia::Front-End::Controllers ::SearchController	Matteo Granzotto Progettista	2016-03-05
0.3.6	Stesura QuizziPedia::Front-End::Controllers ::ResultQuestionnaireController	Alberto Ferrara Progettista	2016-03-05
0.3.5	Stesura QuizziPedia::Front-End::Controllers ::ResultsController	Alberto Ferrara Progettista	2016-03-05
0.3.4	Stesura QuizziPedia::Front-End::Controllers ::RegistrationManagementController	Simone Magagna Progettista	2016-03-05
0.3.3	Stesura QuizziPedia::Back-End::App::Controllers ::QuestionController	Matteo Gnoato Progettista	2016-03-04



Versione	Descrizione	Autore e Ruolo	Data
0.3.2	Stesura QuizziPedia::Back-End::App::controllers::UserController	Franco Berton Progettista	2016-03-04
0.3.1	Stesura QuizziPedia::Front-End::Controllers ::QuizEventController	Simone Magagna Progettista	2016-03-04
0.3.0	Stesura QuizziPedia::Front-End::Controllers ::QuestionManagementController	Simone Magagna Progettista	2016-03-04
0.3.0	Verifica Documento	Franco Berton Verificatore	2016-03-04
0.2.18	Stesura QuizziPedia::Front-End::Controllers ::QuestionsController	Alberto Ferrara Progettista	2016-03-03
0.2.17	Stesura QuizziPedia::Front-End::Controllers ::QuestionnaireQuestionManagementController	Alberto Ferrara Progettista	2016-03-03
0.2.16	Stesura QuizziPedia::Back-End::App::Models ::LangModel	Matteo Gnoato Progettista	2016-03-03
0.2.15	Stesura QuizziPedia::Front-End::Controllers ::QuestionnaireManagementController	Simone Magagna Progettista	2016-03-02
0.2.14	Stesura QuizziPedia::Front-End::Controllers ::QuestionnaireDetailsController	Matteo Granzotto Prog	2016-03-02
0.2.13	Stesura QuizziPedia::Back-End::App::Models ::SummaryModel	Mattia Varotto Progettista	2016-03-02
0.2.12	Stesura QuizziPedia::Front-End::Controllers ::ProfileManagementController	Alberto Ferrara Progettista	2016-03-02
0.2.11	Stesura QuizziPedia::Front-End::Controllers ::PasswordForgotController	Matteo Granzotto Progettista	2016-03-02
0.2.10	Stesura QuizziPedia::Back-End::App::Models ::TopicModel	Mattia Varotto Progettista	2016-03-02
0.2.9	Stesura QuizziPedia::Front-End::Controllers ::NewQuestionButtonController	Simone Magagna Progettista	2016-03-02
0.2.8	Stesura QuizziPedia::Front-End::Controllers ::MultipleQuestionController	Simone Magagna Progettista	2016-03-01
0.2.7	Stesura QuizziPedia::Front-End ::MenuBarController	Matteo Granzotto Progettista	2016-03-01
0.2.6	Stesura QuizziPedia::Front-End::Controllers ::LoginController	Simone Magagna Progettista	2016-03-01
0.2.5	Stesura QuizziPedia::Front-End::Controllers ::InputToListController	Matteo Granzotto Progettista	2016-03-01
0.2.4	Stesura QuizziPedia::Front-End::Controllers ::ImagesSortingQuestionsController	Simone Magagna Progettista	2016-03-01



Versione	Descrizione	Autore e Ruolo	Data
0.2.3	Stesura QuizziPedia::Back-End::App::Models ::QuizModel	Marco Prelaz Progettista	2016-03-01
0.2.2	Stesura QuizziPedia::Front-End::Controllers ::HomeController	Alberto Ferrara Progettista	2016-03-01
0.2.1	Stesura QuizziPedia::Front-End::Controllers ::FillingQuestionsController	Alberto Ferrara Progettista	2016-03-01
0.2.0	Verifica Documento	Alberto Ferrara Verificatore	2016-03-01
0.1.18	Stesura QuizziPedia::Back-End::App::Models ::Session	Franco Berton Progettista	2016-03-01
0.1.17	Stesura QuizziPedia::Front-End::Controllers ::FillingQuestionnaireController	Simone Magagna Progettista	2016-02-29
0.1.16	Stesura QuizziPedia::Front-End::Controllers ::EditorQMLController	Matteo Granzotto Progettista	2016-02-29
0.1.15	Stesura QuizziPedia::Back-End::App::Models ::UserProModel	Franco Berton Progettista	2016-02-29
0.1.14	Stesura QuizziPedia::Back-End::App::Models ::UserModel	Franco Berton Progettista	2016-02-29
0.1.13	Stesura QuizziPedia::Front-End::Controllers ::CreateQuestionnaireController	Matteo Granzotto Progettista	2016-02-29
0.1.12	Stesura QuizziPedia::Front-End::Controllers ::ConnectionQuestionController	Alberto Ferrara Progettista	2016-02-29
0.1.11	Stesura QuizziPedia::Front-End::Controllers ::ClickableAreaQuestionsController	Simone Magagna Progettista	2016-02-29
0.1.10	Aggiunte risorse REST: <i>question/updateStatistics, question/editQuestion</i>	Matteo Gnoato Progettista	2016-02-28
0.1.9	Aggiunte risorse REST: <i>question/mine, question/send, question/createQuestion</i>	Matteo Gnoato Progettista	2016-02-28
0.1.8	Stesura QuizziPedia::Front-End::Models ::ErrorInfoModel	Matteo Granzotto Progettista	2016-02-28
0.1.7	Aggiunte risorse REST: <i>user/userId, quiz-/search, /stats</i>	Matteo Gnoato Progettista	2016-02-28
0.1.6	Stesura QuizziPedia::Front-End::Models ::LangModel	Alberto Ferrara Progettista	2016-02-28
0.1.5	Aggiunte risorse REST: <i>lang, error, user-/search, user/me</i>	Matteo Gnoato Progettista	2016-02-28
0.1.4	Aggiunte risorse REST: <i>signin, signout, signup, recovery</i>	Matteo Gnoato Progettista	2016-02-28



Versione	Descrizione	Autore e Ruolo	Data
0.1.3	Stesura QuizziPedia::Back-End::App::Controllers::QuestionController	Matteo Gnoato Progettista	2016-02-27
0.1.2	Stesura QuizziPedia::Front-End::Models ::MenuBarModel	Simone Magagna Progettista	2016-02-27
0.1.1	Stesura QuizziPedia::Front-End::Models ::QuestionItemModel	Matteo Granzotto Progettista	2016-02-27
0.1.0	Verifica Documento	Matteo Gnoato Verificatore	2016-02-27
0.0.14	Stesura QuizziPedia::Front-End::Models ::QuestionnaireModel	Matteo Granzotto Progettista	2016-02-26
0.0.13	Stesura QuizziPedia::Front-End::Models ::TrainingModeModel	Simone Magagna Progettista	2016-02-26
0.0.12	Stesura QuizziPedia::Front-End::Models ::User-DetailsModel	Alberto Ferrara Progettista	2016-02-26
0.0.11	Stesura QuizziPedia::Front-End::Index	Simone Magagna Progettista	2016-02-25
0.0.10	Stesura QuizziPedia::Front-End::AppRun	Alberto Ferrara Progettista	2016-02-25
0.0.9	Stesura QuizziPedia::Front-End::AppRouter	Matteo Granzotto Progettista	2016-02-25
0.0.8	Stesura QuizziPedia::Back-End::App::Models ::QuestionModel	Matteo Gnoato Progettista	2016-02-25
0.0.7	Stesura QuizziPedia::Back-End::App::Controllers, QuizziPedia::Back-End::App::Models	Franco Berton Progettista	2016-02-25
0.0.6	Stesura QuizziPedia::Back-End, QuizziPedia::Back-End::Config	Mattia Varotto Progettista	2016-02-25
0.0.5	Stesura Appendice Design Pattern: Facade, Chain-of-responsability	Matteo Gnoato Progettista	2016-02-25
0.0.4	Inizio stesura Appendice interfaccia REST	Matteo Gnoato Progettista	2016-02-24
0.0.3	Inseriti i primi riferimenti informativi	Marco Prelaz Progettista	2016-02-24
0.0.2	Stesura scopo del documento, scopo del prodotto , riferimenti normativi	Matteo Gnoato Progettista	2016-02-23
0.0.1	Creato template documento	Matteo Gnoato Progettista	2016-02-23



Indice

1	Introduzione	8
1.1	Scopo del documento	8
1.2	Scopo del prodotto	8
1.3	Glossario	8
1.4	Riferimenti	8
1.4.1	Normativi	8
1.4.2	Informativi	8
2	Specifiche del Front-End	11
2.1	QuizziPedia::Front-End	11
2.1.1	Informazioni generali	11
2.1.2	Classi	11
2.1.2.1	QuizziPedia::Front-End::Index	11
2.1.2.2	QuizziPedia::Front-End::AppRun	13
2.1.2.3	QuizziPedia::Front-End::AppRouter	15
2.2	QuizziPedia::Front-End::Models	16
2.2.1	Informazioni generali	16
2.2.2	Classi	16
2.2.2.1	QuizziPedia::Front-End::Models::UserDetailsModel	16
2.2.2.2	QuizziPedia::Front-End::Models::TrainingModeModel	19
2.2.2.3	QuizziPedia::Front-End::Models::QuestionnaireModel	21
2.2.2.4	QuizziPedia::Front-End::Models::QuestionItemModel	22
2.2.2.5	QuizziPedia::Front-End::Models::MenuBarModel	26
2.2.2.6	QuizziPedia::Front-End::Models::LangModel	27
2.2.2.7	QuizziPedia::Front-End::Models::ErrorInfoModel	28
2.3	QuizziPedia::Front-End::Controllers	30
2.3.1	Informazioni generali	30
2.3.2	Classi	30
2.3.2.1	QuizziPedia::Front-End::Controllers ::ClickableAreaQuestionsController	30
2.3.2.2	QuizziPedia::Front-End::Controllers ::ConnectionQuestionsController	32
2.3.2.3	QuizziPedia::Front-End::Controllers ::CreateQuestionnaireController	34
2.3.2.4	QuizziPedia::Front-End::Controllers::EditorQMLController	37
2.3.2.5	QuizziPedia::Front-End::Controllers ::FillingQuestionnaireController	38
2.3.2.6	QuizziPedia::Front-End::Controllers::FillingQuestionsController	40
2.3.2.7	QuizziPedia::Front-End::Controllers::HomeController	42
2.3.2.8	QuizziPedia::Front-End::Controllers ::ImagesSortingQuestionsController	43
2.3.2.9	QuizziPedia::Front-End::Controllers::InputToListController	45
2.3.2.10	QuizziPedia::Front-End::Controllers::LoginController	47
2.3.2.11	QuizziPedia::Front-End::Controllers::MenuBarController	48
2.3.2.12	QuizziPedia::Front-End::Controllers::MultipleQuestionsController	50
2.3.2.13	QuizziPedia::Front-End::Controllers ::NewQuestionsButtonController	52
2.3.2.14	QuizziPedia::Front-End::Controllers::PasswordForgotController	53



2.3.2.15	QuizziPedia::Front-End::Controllers ::ProfileManagementController	55
2.3.2.16	QuizziPedia::Front-End::Controllers ::QuestionnaireDetailsController	57
2.3.2.17	QuizziPedia::Front-End::Controllers ::QuestionnaireManagementController	58
2.3.2.18	QuizziPedia::Front-End::Controllers::QuestionsController	60
2.3.2.19	QuizziPedia::Front-End::Controllers ::QuestionsManagementController	63
2.3.2.20	QuizziPedia::Front-End::Controllers::QuizEventController	64
2.3.2.21	QuizziPedia::Front-End::Controllers ::RegistrationManagementController	66
2.3.2.22	QuizziPedia::Front-End::Controllers ::ResultsQuestionnaireController	67
2.3.2.23	QuizziPedia::Front-End::Controllers::SearchController	69
2.3.2.24	QuizziPedia::Front-End::Controllers::SignUpController	71
2.3.2.25	QuizziPedia::Front-End::Controllers::StatisticsController	73
2.3.2.26	QuizziPedia::Front-End::Controllers ::StringsSortingQuestionsController	74
2.3.2.27	QuizziPedia::Front-End::Controllers::TopicKeywordsController	76
2.3.2.28	QuizziPedia::Front-End::Controllers::TrainingController	77
2.3.2.29	QuizziPedia::Front-End::Controllers::TrueFalseQuestionsController	79
2.3.2.30	QuizziPedia::Front-End::Controllers::UserDetailsController	81
2.4	QuizziPedia::Front-End::Directives	83
2.4.1	Informazioni generali	83
2.4.2	Classi	83
2.4.2.1	QuizziPedia::Front-End::Directives ::EliminationAndModifyDirective	83
2.4.2.2	QuizziPedia::Front-End::Directives::ExamModalityDirective	84
2.4.2.3	QuizziPedia::Front-End::Directives::FooterDirective	85
2.4.2.4	QuizziPedia::Front-End::Directives::ImageInTheQuestionDirective	86
2.4.2.5	QuizziPedia::Front-End::Directives::MenuBarDirective	86
2.4.2.6	QuizziPedia::Front-End::Directives::OneQuestionDirective	88
2.4.2.7	QuizziPedia::Front-End::Directives::NewQuestionButtonDirective	89
2.4.2.8	QuizziPedia::Front-End::Directives::QuestionTextDirective	89
2.4.2.9	QuizziPedia::Front-End::Directives::QuestionnaireDetailsDirective	90
2.4.2.10	QuizziPedia::Front-End::Directives ::QuestionnaireDoneDetailsDirective	92
2.4.2.11	QuizziPedia::Front-End::Directives::QuestionnaireResultsDirective	93
2.4.2.12	QuizziPedia::Front-End::Directives::SearchDirective	94
2.4.2.13	QuizziPedia::Front-End::Directives::StatisticsDirective	95
2.4.2.14	QuizziPedia::Front-End::Directives::SubscribeResultDirective	96
2.4.2.15	QuizziPedia::Front-End::Directives::TopicKeywordsDirective	97
2.4.2.16	QuizziPedia::Front-End::Directives::UserDetailsDirective	98
2.4.2.17	QuizziPedia::Front-End::Directives::UserResultsDirective	99
2.4.2.18	QuizziPedia::Front-End::Directives::ClickableAnswerDirective	100
2.4.2.19	QuizziPedia::Front-End::Directives::EmptySpaceAnswerDirective	101
2.4.2.20	QuizziPedia::Front-End::Directives::HeaderTextQuestionDirective	103
2.4.2.21	QuizziPedia::Front-End::Directives::InfoQuestionnaireDirective	104
2.4.2.22	QuizziPedia::Front-End::Directives::LinkingAnswerDirective	105



2.4.2.23	QuizziPedia::Front-End::Directives ::MultipleChoiceAnswerDirective	107
2.4.2.24	QuizziPedia::Front-End::Directives::SortImagesAnswerDirective .	108
2.4.2.25	QuizziPedia::Front-End::Directives::SortTextAnswerDirective .	110
2.4.2.26	QuizziPedia::Front-End::Directives::TrainingSetUpDirective . .	111
2.4.2.27	QuizziPedia::Front-End::Directives::TrueFalseAnswerDirective .	112
2.4.2.28	QuizziPedia::Front-End::Directives::LoginBarDirective	113
2.4.2.29	QuizziPedia::Front-End::Directives::SignUpBarDirective	114
2.4.2.30	QuizziPedia::Front-End::Directives::UserBarDirective	115
2.4.2.31	QuizziPedia::Front-End::Directives ::ProfileManagementBarDirective	115
2.4.2.32	QuizziPedia::Front-End::Directives ::QuestionsManagementBarDirective	116
2.4.2.33	QuizziPedia::Front-End::Directives::LogoutBarDirective	117
2.4.2.34	QuizziPedia::Front-End::Directives ::QuestionnaireManagementBarDirective	117
2.5	QuizziPedia::Front-End::Services	119
2.5.1	Informazioni generali	119
2.5.2	Classi	119
2.5.2.1	QuizziPedia::Front-End::Services::AuthService	119
2.5.2.2	QuizziPedia::Front-End::Services::LangService	121
2.5.2.3	QuizziPedia::Front-End::Services::QuestionsService	122
2.5.2.4	QuizziPedia::Front-End::Services::QuizService	124
2.5.2.5	QuizziPedia::Front-End::Services::SearchService	128
2.5.2.6	QuizziPedia::Front-End::Services::StatisticsService	129
2.5.2.7	QuizziPedia::Front-End::Services::UserDetailsService	130
2.6	QuizziPedia::Front-End::Views	132
2.6.1	Informazioni generali	132
2.6.2	Classi	132
2.6.2.1	QuizziPedia::Front-End::Views::LoginView	132
2.6.2.2	QuizziPedia::Front-End::Views::SignUpView	134
2.6.2.3	QuizziPedia::Front-End::Views::PasswordForgotView	135
2.6.2.4	QuizziPedia::Front-End::Views::HomeView	136
2.6.2.5	QuizziPedia::Front-End::Views::ResultsView	137
2.6.2.6	QuizziPedia::Front-End::Views::UserView	138
2.6.2.7	QuizziPedia::Front-End::Views::OtherUserView	139
2.6.2.8	QuizziPedia::Front-End::Views::ProfileManagementView	139
2.6.2.9	QuizziPedia::Front-End::Views::QuestionsManagementView . . .	141
2.6.2.10	QuizziPedia::Front-End::Views::TrueFalseQuestionsView	142
2.6.2.11	QuizziPedia::Front-End::Views::MultipleQuestionsView	142
2.6.2.12	QuizziPedia::Front-End::Views::ConnectionQuestionsView . . .	143
2.6.2.13	QuizziPedia::Front-End::Views::ImagesSortingQuestionsView . .	145
2.6.2.14	QuizziPedia::Front-End::Views::StringsSortingQuestionsView . .	146
2.6.2.15	QuizziPedia::Front-End::Views::FillingQuestionsView	147
2.6.2.16	QuizziPedia::Front-End::Views::ClickableAreaQuestionsView . .	148
2.6.2.17	QuizziPedia::Front-End::Views::EditorQMLView	149
2.6.2.18	QuizziPedia::Front-End::Views::TrainingView	150
2.6.2.19	QuizziPedia::Front-End::Views::FillingQuestionnaireView	152
2.6.2.20	QuizziPedia::Front-End::Views::QuestionnaireManagementView .	155
2.6.2.21	QuizziPedia::Front-End::Views::CreateQuestionnaireView	156
2.6.2.22	QuizziPedia::Front-End::Views::ResultsQuestionnaireView	157



2.6.2.23	QuizziPedia::Front-End::Views::RegistrationManagementView	157
2.7	QuizziPedia::Front-End::ModelViews	159
2.7.1	Informazioni generali	159
2.7.2	Classi	159
2.7.2.1	QuizziPedia::Front-End::ModelViews ::ClickableAreaQuestionsModelView	159
2.7.2.2	QuizziPedia::Front-End::ModelViews ::ConnectionQuestionsModelView	160
2.7.2.3	QuizziPedia::Front-End::ModelViews ::CreateQuestionnaireModelView	161
2.7.2.4	QuizziPedia::Front-End::ModelViews::EditorQMLModelView	163
2.7.2.5	QuizziPedia::Front-End::ModelViews ::FillingQuestionnaireModelView	164
2.7.2.6	QuizziPedia::Front-End::ModelViews::FillingQuestionsModelView	165
2.7.2.7	QuizziPedia::Front-End::ModelViews::HomeModelView	166
2.7.2.8	QuizziPedia::Front-End::ModelViews ::ImagesSortingQuestionsModelView	167
2.7.2.9	QuizziPedia::Front-End::ModelViews::LoginModelView	168
2.7.2.10	QuizziPedia::Front-End::ModelViews::MenuBarModelView	168
2.7.2.11	QuizziPedia::Front-End::ModelViews ::MultipleQuestionsModelView	170
2.7.2.12	QuizziPedia::Front-End::ModelViews::PasswordForgotModelView	171
2.7.2.13	QuizziPedia::Front-End::ModelViews ::ProfileManagementModelView	172
2.7.2.14	QuizziPedia::Front-End::ModelViews ::QuestionnaireDetailsModelView	173
2.7.2.15	QuizziPedia::Front-End::ModelViews ::QuestionnaireManagementModelView	174
2.7.2.16	QuizziPedia::Front-End::ModelViews ::QuestionsManagementModelView	175
2.7.2.17	QuizziPedia::Front-End::ModelViews::QuestionsModelView	175
2.7.2.18	QuizziPedia::Front-End::ModelViews::QuizEventModelView	178
2.7.2.19	QuizziPedia::Front-End::ModelViews ::RegistrationManagementModelView	180
2.7.2.20	QuizziPedia::Front-End::ModelViews::ResultsModelView	180
2.7.2.21	QuizziPedia::Front-End::ModelViews ::ResultsQuestionnaireModelView	182
2.7.2.22	QuizziPedia::Front-End::ModelViews::SignUpModelView	182
2.7.2.23	QuizziPedia::Front-End::ModelViews::StatisticsModelView	183
2.7.2.24	QuizziPedia::Front-End::ModelViews ::StringsSortingQuestionsModelView	184
2.7.2.25	QuizziPedia::Front-End::ModelViews::TopicKeywordsModelView	185
2.7.2.26	QuizziPedia::Front-End::ModelViews::TrainingModelView	185
2.7.2.27	QuizziPedia::Front-End::ModelViews ::TrueFalseQuestionsModelView	187
2.7.2.28	QuizziPedia::Front-End::ModelViews::UserDetailsModelView	188
3	Specifiche del Back-End	189
3.1	QuizziPedia::Back-End	189
3.1.1	Informazioni generali	189
3.1.2	Classi	189
3.1.2.1	QuizziPedia::Back-End::Server	189



3.2	QuizziPedia::Back-End::Config	190
3.2.1	Informazioni generali	190
3.2.2	Classi	190
3.2.2.1	QuizziPedia::Back-End::Config::Config	190
3.3	QuizziPedia::Back-End::App	191
3.3.1	Informazioni generali	191
3.4	QuizziPedia::Back-End::App::Controllers	191
3.4.1	Informazioni generali	191
3.4.2	Classi	192
3.4.2.1	QuizziPedia::Back-End::App::Controllers::ErrorsHandler	192
3.4.2.2	QuizziPedia::Back-End::App::Controllers::TopicController	193
3.4.2.3	QuizziPedia::Back-End::App::Controllers::SummaryController	195
3.4.2.4	QuizziPedia::Back-End::App::Controllers::QuizController	196
3.4.2.5	QuizziPedia::Back-End::App::Controllers::QuestionController	199
3.4.2.6	QuizziPedia::Back-End::App::Controllers::UserController	201
3.4.2.7	QuizziPedia::Back-End::App::Controllers::LangController	202
3.4.2.8	QuizziPedia::Back-End::App::Controllers::NotFoundHandler	203
3.5	QuizziPedia::Back-End::App::Controllers::Errors	204
3.5.1	Informazioni generali	204
3.5.2	Classi	204
3.5.2.1	QuizziPedia::Back-End::App::Controllers::Errors ::QuizziPediaError	204
3.6	QuizziPedia::Back-End::App::Controllers::Users	205
3.6.1	Informazioni generali	205
3.6.2	Classi	206
3.6.2.1	QuizziPedia::Back-End::App::Controllers::Users ::AuthenticationController	206
3.6.2.2	QuizziPedia::Back-End::App::Controllers::Users ::SessionController	207
3.6.2.3	QuizziPedia::Back-End::App::Controllers::Users ::UserManagementController	208
3.7	QuizziPedia::Back-End::App::Models	211
3.7.1	Informazioni generali	211
3.7.2	Classi	212
3.7.2.1	QuizziPedia::Back-End::App::Models::Session	212
3.7.2.2	QuizziPedia::Back-End::App::Models::UserModel	213
3.7.2.3	QuizziPedia::Back-End::App::Models::UserProModel	217
3.7.2.4	QuizziPedia::Back-End::App::Models::QuestionModel	218
3.7.2.5	QuizziPedia::Back-End::App::Models::QuizModel	221
3.7.2.6	QuizziPedia::Back-End::App::Models::TopicModel	224
3.7.2.7	QuizziPedia::Back-End::App::Models::SummaryModel	226
3.7.2.8	QuizziPedia::Back-End::App::Models::LangModel	227
3.7.2.9	QuizziPedia::Back-End::App::Models::ModelError	228
3.8	QuizziPedia::Back-End::App::Routers	229
3.8.1	Informazioni generali	229
3.8.2	Classi	229
3.8.2.1	QuizziPedia::Back-End::App::Routers::UserRouter	229
3.8.2.2	QuizziPedia::Back-End::App::Routers::QuestionRouter	230
3.8.2.3	QuizziPedia::Back-End::App::Routers::QuizRouter	231
3.8.2.4	QuizziPedia::Back-End::App::Routers::LangRouter	232



4 Diagrammi di sequenza	234
4.1 Front-End	234
4.1.1 Principali operazioni Front-End	234
4.1.1.1 Autenticazione	234
4.1.1.2 Registrazione	234
4.1.1.3 Recupero password	235
4.1.1.4 Ricerca	236
4.1.1.4.1 Ricerca	236
4.1.1.4.2 Iscrizione ad un questionario	236
4.1.1.4.3 Redirect alla pagina di un utente	237
4.1.1.5 Visualizzazione pagina profilo	237
4.1.1.5.1 Pagina profilo personale	237
4.1.1.5.2 Pagina profilo altri utenti	238
4.1.1.6 Gestione profilo utente	238
4.1.1.7 Gestione Domande	238
4.1.1.8 Creazione Domande	239
4.1.1.9 Modalità allenamento	240
4.1.1.9.1 Selezione parametri di set-up	240
4.1.1.9.2 Download di una domanda	240
4.1.1.9.3 Composizione di una domanda	241
4.1.1.9.4 Risposta ad una domanda	241
4.1.1.10 Compilazione questionario	242
4.1.1.10.1 Inizio questionario	242
4.1.1.10.2 Composizione di una domanda	242
4.1.1.10.3 Risposta ad una domanda	242
4.1.1.11 Gestione Questionari	242
4.1.1.12 Creazione Questionario	243
4.1.1.13 Gestione iscrizioni ad un questionario	243
4.1.1.14 Gestione questionari	244
4.1.1.14.1 Gestione iscrizioni del questionario	244
4.1.1.15 Recupero utenti del questionario	244
4.2 Back-End	245
4.2.1 Gestione generale delle richieste	245
4.2.2 Richieste REST	246
4.2.2.1 GET /:lang	246
4.2.2.2 POST /:lang/signup	247
4.2.2.3 POST /:lang/signin	248
4.2.2.4 POST /:lang/signout	249
4.2.2.5 GET /:lang/loggedin	250
4.2.2.6 POST /:lang/recovery	251
4.2.2.7 DELETE /:lang/user/:userId	252
4.2.2.8 GET /:lang/user/:userId	253
4.2.2.9 PUT /:lang/user/:userId	254
4.2.2.10 PUT /:lang/user/:userId/privacy	255
4.2.2.11 GET /:lang/user/:userId/statistics	256
4.2.2.12 GET /:lang/user/:userId/statistics/summary/	257
4.2.2.13 GET /:lang/user/:userId/statistics/summary/:summaryId	258
4.2.2.14 GET /:lang/user/:userId/quiz	259
4.2.2.15 GET /:lang/user/:userId/quiz/:quizId	260
4.2.2.16 POST /:lang/user/:useId/quiz	261
4.2.2.17 PUT /:lang/user/:userId/quiz/:quizId/removeUser	262



4.2.2.18 POST /:lang/user/:userId/quiz/:quizId/addUser	263
4.2.2.19 POST /:lang/user/:userId/quiz/:quizId/activeUser	264
4.2.2.20 PUT /:lang/user/:userId/quiz/:quizId	265
4.2.2.21 GET /:lang/user/:userId/quiz/:quizId/test	266
4.2.2.22 POST /:lang/user/:userId/quiz/:quizId/summary	267
4.2.2.23 GET /:lang/user/:userId/question	268
4.2.2.24 GET /:lang/user/:userId/question/:questionId	269
4.2.2.25 POST /:lang/user/:userId/question	270
4.2.2.26 GET /:lang/user/:userId/search/:keyword/users	271
4.2.2.27 GET /:lang/user/:userId/search/:keyword/quizzes	272
4.2.2.28 GET /:lang/user/:userId/search/users/:users	273
4.2.2.29 GET /:lang/user/:userId/search/quizzes/:quizId	274
4.2.2.30 GET /:lang/user/training/question	275
4.2.2.31 PUT /:lang/user/:userId/training/userstatistics	276
4.2.2.32 PUT /:lang/user/:userId/training/questionstatistics	277
4.2.2.33 PUT /:lang/user/training/userlevelupdate	278
5 Tracciamento Classi-Requisiti	279
6 Tracciamento Requisiti-Classi	319
7 Tracciamento Componenti-Requisiti	392
8 Tracciamento Requisiti-Componenti	437
A Design patterns	483
A.1 Strutturali	483
A.1.1 Facade	483
A.2 Creazionali	484
A.2.1 Dependecy Injection	484
A.3 Comportamentali	486
A.3.1 Chain-of-responsability	486
A.3.2 Observer	487
B QML - Quiz Markup Language	489
B.1 Definizione della grammatica	489
B.1.1 Sintassi domanda Vero/Falso	489
B.1.2 Sintassi domanda a Risposta Multipla	489
B.1.3 Sintassi domanda a Ordinamento di Stringhe	490
B.1.4 Sintassi domanda a Ordinamento Immagini	490
B.1.5 Sintassi domanda a Collegamento di Elementi	491
B.1.6 Sintassi domanda ad Area Cliccabile	491
B.1.7 Sintassi domanda a Riempimento spazi vuoti	492
B.2 Generazione del JSON	492
B.2.1 JSON domanda Vero/Falso	492
B.2.2 JSON domanda a Risposta Multipla	493
B.2.3 JSON domanda a Ordinamento di Stringhe	494
B.2.4 JSON domanda a Ordinamento Immagini	494
B.2.5 JSON domanda a Collegamento di Elementi	495
B.2.6 JSON domanda ad Area Cliccabile	495
B.2.7 JSON domanda a Riempimento spazi vuoti	496



C Interfaccia REST	497
C.1 Errori	497
C.1.1 Errori generici	497
C.1.1.1 Errori lato server	497
C.1.1.2 Errori nelle richieste da parte del client	497
C.1.1.3 Errori specifici di QuizziPedia	498
C.2 Risorse REST	500
D Calcolo delle Statistiche	509
D.1 Statistiche Utente	510
D.1.1 Utente autenticato	510
D.1.2 Utente non autenticato	510
D.2 Statistiche Domanda	510
D.3 Statistiche Argomento	511
D.4 Statistiche Questionario	511
E Algoritmo di selezione della domanda	511



Elenco delle figure

1	QuizziPedia::Front-End	11
2	QuizziPedia::Front-End::Views::Index	11
3	QuizziPedia::Front-End::AppRun	13
4	QuizziPedia::Front-End::AppRouter	15
5	QuizziPedia::Front-End::Models	16
6	QuizziPedia::Front-End::Models::UserDetailsModel	17
7	QuizziPedia::Front-End::Models::TrainingModeModel	19
8	QuizziPedia::Front-End::Models::QuestionnaireModel	21
9	QuizziPedia::Front-End::Models::QuestionItemModel	23
10	QuizziPedia::Front-End::Models::MenuBarModel	26
11	QuizziPedia::Front-End::Models::LangModel	27
12	QuizziPedia::Front-End::Models::ErrorInfoModel	28
13	QuizziPedia::Front-End::Controllers	30
14	QuizziPedia::Front-End::Controllers::ClickableAreaQuestionsController	30
15	QuizziPedia::Front-End::Controllers::ConnectionQuestionsController	33
16	QuizziPedia::Front-End::Controllers::CreateQuestionnaireController	35
17	QuizziPedia::Front-End::Controllers::EditorQMLController	37
18	QuizziPedia::Front-End::Controllers::FillingQuestionnaireController	39
19	QuizziPedia::Front-End::Controllers::FillingQuestionsController	41
20	QuizziPedia::Front-End::Controllers::HomeController	43
21	QuizziPedia::Front-End::Controllers::ImagesSortingQuestionsController	44
22	QuizziPedia::Front-End::Controllers::InputToListController	46
23	QuizziPedia::Front-End::Controllers::LoginController	47
24	QuizziPedia::Front-End::Controllers::MenuBarController	48
25	QuizziPedia::Front-End::Controllers::MultipleChoiceQuestion	51
26	QuizziPedia::Front-End::Controllers::NewQuestionsButtonController	53
27	QuizziPedia::Front-End::Controllers::PasswordForgotController	54
28	QuizziPedia::Front-End::Controllers::ProfileManagementController	55
29	QuizziPedia::Front-End::Controllers::QuestionnaireDetailsController	57
30	QuizziPedia::Front-End::Controllers::QuestionnaireManagementController	59
31	QuizziPedia::Front-End::Controllers::QuestionsController	60
32	QuizziPedia::Front-End::Controllers::QuestionsManagementController	63
33	QuizziPedia::Front-End::Controllers::QuizEventController	64
34	QuizziPedia::Front-End::Controllers::RegistrationManagementController	66
35	QuizziPedia::Front-End::Controllers::ResultsQuestionnaireController	68
36	QuizziPedia::Front-End::Controllers::SearchController	69
37	QuizziPedia::Front-End::Controllers::SignUpController	71
38	QuizziPedia::Front-End::Controllers::StatisticsController	73
39	QuizziPedia::Front-End::Controllers::StringsSortingQuestionsController	74
40	QuizziPedia::Front-End::Controllers::TopicKeywordsController	76
41	QuizziPedia::Front-End::Controllers::TrainingController	77
42	QuizziPedia::Front-End::Controllers::TrueFalseQuestionsController	79
43	QuizziPedia::Front-End::Controllers::UserDetailController	81
44	QuizziPedia::Front-End::Directives	83
45	QuizziPedia::Front-End::Directives::EliminationAndModifyDirective	83
46	QuizziPedia::Front-End::Directives::ExamModalityDirective	84
47	QuizziPedia::Front-End::Directives::FooterDirective	85
48	QuizziPedia::Front-End::Directives::ImageInTheQuestionDirective	86
49	QuizziPedia::Front-End::Directives::MenuBarDirective	87



50	QuizziPedia::Front-End::Directives::OneQuestionDirective	88
51	QuizziPedia::Front-End::Directives::NewQuestionButtonDirective	89
52	QuizziPedia::Front-End::Directives::QuestionTextDirective	90
53	QuizziPedia::Front-End::Directives::QuestionnaireDetailsDirective	90
54	QuizziPedia::Front-End::Directives::QuestionnaireDetailsDoneDirective	92
55	QuizziPedia::Front-End::Directives::QuestionnaireResultsDirective	93
56	QuizziPedia::Front-End::Directives::SearchDirective	94
57	QuizziPedia::Front-End::Directives::StatisticsDirective	95
58	QuizziPedia::Front-End::Directives::SubscribeResultDirective	96
59	QuizziPedia::Front-End::Directives::TopicKeywordsDirective	97
60	QuizziPedia::Front-End::Directives::UserDetailsDirective	98
61	QuizziPedia::Front-End::Directives::UserResultsDirective	99
62	QuizziPedia::Front-End::Directives::ClickableAnswerDirective	100
63	QuizziPedia::Front-End::Directives::EmptySpaceAnswerDirective	102
64	QuizziPedia::Front-End::Directives::HeaderTextQuestionDirective	103
65	QuizziPedia::Front-End::Directives::InfoQuestionnaireDirective	104
66	QuizziPedia::Front-End::Directives::LinkingAnswerDirective	106
67	QuizziPedia::Front-End::Directives::MultipleChoiceAnswerDirective	107
68	QuizziPedia::Front-End::Directives::SortImagesAnswerDirective	109
69	QuizziPedia::Front-End::Directives::SortTextAnswerDirective	110
70	QuizziPedia::Front-End::Directives::TrainingSetUpDirective	111
71	QuizziPedia::Front-End::Directives::TrueFalseAnswerDirective	112
72	QuizziPedia::Front-End::Directives::LoginBarDirective	114
73	QuizziPedia::Front-End::Directives::SignUpBarDirective	114
74	QuizziPedia::Front-End::Directives::UserBarDirective	115
75	QuizziPedia::Front-End::Directives::ProfileManagementBarDirective	115
76	QuizziPedia::Front-End::Directives::QuestionsManagementBarDirective	116
77	QuizziPedia::Front-End::Views::LogoutBarDirective	117
78	QuizziPedia::Front-End::Directives::QuestionnaireManagementBarDirective	117
79	QuizziPedia::Front-End::Services	119
80	QuizziPedia::Front-End::Services::AuthService	119
81	QuizziPedia::Front-End::Services::LangService	121
82	QuizziPedia::Front-End::Services::QuestionService	122
83	QuizziPedia::Front-End::Services::QuizService	125
84	QuizziPedia::Front-End::Services::SearchService	128
85	QuizziPedia::Front-End::Services::StatisticsService	129
86	QuizziPedia::Front-End::Services::UserDetailsService	130
87	QuizziPedia::Front-End::Views	132
88	QuizziPedia::Front-End::Views::LoginView	133
89	QuizziPedia::Front-End::Views::SignUpView	134
90	QuizziPedia::Front-End::Views::PasswordForgotView	135
91	QuizziPedia::Front-End::Views::HomeView	136
92	QuizziPedia::Front-End::Views::ResultsView	137
93	QuizziPedia::Front-End::Views::UserView	138
94	QuizziPedia::Front-End::Views::OtherUserView	139
95	QuizziPedia::Front-End::Views::ProfileManagementView	140
96	QuizziPedia::Front-End::Views::QuestionsManagementView	141
97	QuizziPedia::Front-End::Views::TrueFalseQuestionsView	142
98	QuizziPedia::Front-End::Views::MultipleQuestionsView	143
99	QuizziPedia::Front-End::Views::ConnectionQuestionsView	144
100	QuizziPedia::Front-End::Views::ImagesSortingQuestionsView	145



101	QuizziPedia::Front-End::Views::StringsSortingQuestionsView	146
102	QuizziPedia::Front-End::Views::FillingQuestionsView	147
103	QuizziPedia::Front-End::Views::ClickableAreaQuestionsView	148
104	QuizziPedia::Front-End::Views::EditorQMLView	150
105	QuizziPedia::Front-End::Views::TrainingView	151
106	QuizziPedia::Front-End::Views::FillingQuestionnaireView	153
107	QuizziPedia::Front-End::Views::QuestionnaireManagementView	155
108	QuizziPedia::Front-End::Views::CreateQuestionnaireView	156
109	QuizziPedia::Front-End::Views::ResultsQuestionnaireView	157
110	QuizziPedia::Front-End::Views::RegistrationManagementView	158
111	QuizziPedia::Front-End::ModelViews	159
112	QuizziPedia::Front-End::ModelViews::ClickableAreaQuestionsModelView	159
113	QuizziPedia::Front-End::ModelViews::ConnectionQuestionsModelView	161
114	QuizziPedia::Front-End::ModelViews::CreateQuestionnaireModelView	162
115	QuizziPedia::Front-End::ModelViews::EditorQMLModelView	163
116	QuizziPedia::Front-End::ModelViews::FillingQuestionnaireModelView	164
117	QuizziPedia::Front-End::ModelViews::FillingQuestionsModelView	165
118	QuizziPedia::Front-End::ModelViews::HomeModelView	166
119	QuizziPedia::Front-End::ModelViews::ImagesSortingQuestionsModelView	167
120	QuizziPedia::Front-End::ModelViews::LoginModelView	168
121	QuizziPedia::Front-End::ModelViews::MenuBarModelView	169
122	QuizziPedia::Front-End::ModelViews::MultipleQuestionsModelView	170
123	QuizziPedia::Front-End::ModelViews::PasswordForgotModelView	171
124	QuizziPedia::Front-End::ModelViews::ProfileManagementModelView	172
125	QuizziPedia::Front-End::ModelViews::QuestionnaireDetailsModelView	173
126	QuizziPedia::Front-End::ModelViews::QuestionnaireManagementModelView	174
127	QuizziPedia::Front-End::ModelViews::QuestionsManagementModelView	175
128	QuizziPedia::Front-End::ModelViews::QuestionsModelView	176
129	QuizziPedia::Front-End::ModelViews::QuizEventModelView	178
130	QuizziPedia::Front-End::ModelViews::RegistrationManagementModelView	180
131	QuizziPedia::Front-End::ModelViews::ResultsModelView	180
132	QuizziPedia::Front-End::ModelViews::ResultsQuestionnaireModelView	182
133	QuizziPedia::Front-End::ModelViews::SignUpModelView	183
134	QuizziPedia::Front-End::ModelViews::StatisticsModelView	183
135	QuizziPedia::Front-End::ModelViews::StringsSortingQuestionsModelView	184
136	QuizziPedia::Front-End::ModelViews::TopicKeywordsModelView	185
137	QuizziPedia::Front-End::ModelViews::TrainingModelView	186
138	QuizziPedia::Front-End::ModelViews::TrueFalseQuestionsModelView	187
139	QuizziPedia::Front-End::ModelViews::UserDetailsModelView	188
140	QuizziPedia::Back-End	189
141	QuizziPedia::Back-End::Server	189
142	QuizziPedia::Back-End::Config	190
143	QuizziPedia::Back-End::Config::Config	190
144	QuizziPedia::Back-End::App	191
145	QuizziPedia::Back-End::App::Controllers	192
146	QuizziPedia::Back-End::App::Controllers::ErrorsHandler	192
147	QuizziPedia::Back-End::App::Controllers::TopicController	194
148	QuizziPedia::Back-End::App::Controllers::SummaryController	195
149	QuizziPedia::Back-End::App::Models::Controllers::TopicController	196
150	QuizziPedia::Back-End::App::Models::Controllers::QuestionController	200
151	QuizziPedia::Back-End::App::Controllers::UserController	201



152	QuizziPedia::Back-End::App::Controllers::LangController	202
153	QuizziPedia::Back-End::App::Controllers::NotFoundHandler	203
154	QuizziPedia::Back-End::App::Controllers::Errors	204
155	QuizziPedia::Back-End::App::Controllers::QuizziPediaError	204
156	QuizziPedia::Back-End::App::Controllers::Users	205
157	QuizziPedia::Back-End::App::Controllers::Users::AuthenticationController	206
158	QuizziPedia::Back-End::App::Controllers::Users::SessionController	207
159	QuizziPedia::Back-End::App::Controllers::Users::UserManagementController	208
160	QuizziPedia::Back-End::App::Models	212
161	QuizziPedia::Back-End::App::Models::Session	212
162	QuizziPedia::Back-End::App::Models::UserModel	213
163	QuizziPedia::Back-End::App::Models::UserProModel	217
164	QuizziPedia::Back-End::App::Models::QuestionModel	218
165	QuizziPedia::Back-End::App::Models::QuizModel	222
166	QuizziPedia::Back-End::App::Models::TopicModel	224
167	QuizziPedia::Back-End::App::Models::summaryModel	226
168	QuizziPedia::Back-End::App::Models::LangModel	227
169	QuizziPedia::Back-End::App::Models::ModelError	228
170	QuizziPedia::Back-End::App::Routers	229
171	QuizziPedia::Back-End::App::Routers::UserRouter	230
172	QuizziPedia::Back-End::App::Routers::QuestionRouter	231
173	QuizziPedia::Back-End::App::Routers::QuizRouter	232
174	QuizziPedia::Back-End::App::Routers::LangRouter	232
175	Autenticazione	234
176	Registrazione	235
177	Recupero password	235
178	Ricerca	236
179	Iscrizione ad un questionario	236
180	Redirect alla pagina di un utente	237
181	Pagina profilo personale	237
182	Gestione profilo utente	238
183	Gestione delle domande create da un utente	239
184	Creazione domanda vero/falso	239
185	Selezione parametri di set-up	240
186	Download di una domanda	240
187	Composizione di una domanda	241
188	Risposta ad una domanda	241
189	Inizio questionario	242
190	Recupero dei questionari di un utente	243
191	Creazione questionario	243
192	Recupero risultati del questionario	244
193	Recupero risultati del questionario	244
194	Recupero singoli utenti del questionario	245
195	Gestione di una richiesta generica	246
196	Procedura di traduzione	247
197	Fallimento della procedura di traduzione	247
198	Procedura di registrazione	248
199	Fallimento della procedura di registrazione	248
200	Procedura di autenticazione	249
201	Fallimento della procedura di autenticazione	249
202	Procedura di logout	250



203	Procedura di controllo di sessione	250
204	Fallimento della procedura di controllo di sessione	251
205	Procedura di recupero password	251
206	Fallimento della procedura di recupero della password	252
207	Procedura di eliminazione account	252
208	Fallimento della procedura di eliminazione account	253
209	Procedura di visualizzazione dati utente	253
210	Fallimento della procedura di visualizzazione dati utente	254
211	Procedura di modifica dati utente	254
212	Fallimento della procedura di modifica dati utente	255
213	Procedura di modifica password	255
214	Fallimento della procedura di modifica password	256
215	Procedura di visualizzazione delle statistiche	256
216	Fallimento della procedura di visualizzazione delle statistiche	257
217	Procedura di visualizzazione della cronologia dei questionari svolti	257
218	Fallimento della procedura di visualizzazione della cronologia dei questionari svolti	258
219	Procedura di visualizzazione del riepilogo di un particolare questionario svolto	258
220	Fallimento della procedura di visualizzazione del riepilogo di un particolare questionario svolto	259
221	Procedura di visualizzazione questionari creati	259
222	Fallimento della procedura di visualizzazione questionari creati	260
223	Procedura di visualizzazione questionari creati	260
224	Fallimento della procedura di visualizzazione questionari creati	261
225	Procedura di creazione di un questionario	261
226	Fallimento della procedura di creazione di un questionario	262
227	Procedura di rimozione di un utente iscritto ad un questionario creato	262
228	Fallimento della procedura di visualizzazione questionari creati	263
229	Procedura di iscrizione ad un questionario	263
230	Fallimento della procedura di iscrizione ad un questionario	264
231	Procedura di aggiunta alla lista di utenti che hanno compilato un questionario	264
232	Fallimento della procedura di aggiunta alla lista di utenti che hanno compilato un questionario	265
233	Procedura di modifica di un questionario	265
234	Fallimento della procedura di modifica di un questionario	266
235	Procedura di visualizzazione del questionario da compilare	266
236	Fallimento della procedura di visualizzazione del questionario da compilare	267
237	Procedura di creazione di un riepilogo	267
238	Fallimento della procedura di creazione di un riepilogo	268
239	Procedura di visualizzazione delle domande create dall'utente	268
240	Fallimento della procedura di visualizzazione delle domande create dall'utente	269
241	Procedura di visualizzazione delle domande create dall'utente	269
242	Fallimento della procedura di visualizzazione delle domande create dall'utente	270
243	Procedura della creazione di una domanda	270
244	Fallimento della procedura di creazione di una domanda	271
245	Procedura di visualizzazione dei risultati della ricerca utente	271
246	Fallimento della procedura di visualizzazione dei risultati della ricerca utente	272
247	Procedura di visualizzazione dei risultati della ricerca questionario	272
248	Fallimento della procedura di visualizzazione dei risultati della ricerca questionario	273
249	Procedura di visualizzazione di un utente	273
250	Fallimento della procedura della visualizzazione di un utente	274
251	Procedura della visualizzazione di un questionario	274



252	Fallimento della procedura di visualizzazione di un questionario	275
253	Procedura di restituzione della domanda successiva nella modalità allenamento	275
254	Fallimento della procedura di restituzione della domanda successiva nella modalità allenamento	276
255	Procedura di aggiornamento delle statistiche dell'utente dopo una risposta ad una domanda	276
256	Fallimento della procedura di aggiornamento delle statistiche dell'utente dopo una risposta ad una domanda	277
257	Procedura di aggiornamento delle statistiche di una domanda	277
258	Fallimento della procedura di aggiornamento delle statistiche di una domanda	278
259	Procedura di aggiornamento del livello di un utente non autenticato	278
260	Fallimento della procedura di aggiornamento del livello di un utente non autenticato	279
261	Parte di QuizziPedia::Back-End senza utilizzo di Facade	484
262	Parte di QuizziPedia::Back-End con l'utilizzo di Facade	484
263	Struttura logica del pattern Dependency Injection	485
264	Struttura logica del pattern Chain-of-responsability	487
265	Utilizzo di Observer per l'aggiornamento delle statistiche	488
266	Funzionamento Algoritmo di selezione della domanda	512



Elenco delle tabelle

1	Tracciamento Classi-Requisiti	318
2	Tracciamento Requisiti-Classi	391
3	Tracciamento Componenti-Requisiti	436
4	Tracciamento Requisiti-Componenti	482



1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire in dettaglio la struttura e il funzionamento delle componenti del progetto Quizzipedia. Questo documento servirà come guida per i *Programmatori* del gruppo TheFellowshipOfTheCode fornendo direttive e vincoli per la realizzazione del *progettoG*.

1.2 Scopo del prodotto

Lo scopo del prodotto è di permettere la creazione e gestione di questionari in grado di identificare le lacune dei candidati prima, durante e al termine di un corso di formazione.

Il sistema dovrà offrire le seguenti funzionalità:

- Archiviare questionari in un server suddivisi per argomento;
- Somministrare all'utente, tramite un'interfaccia, questionari specifici per argomento scelto;
- Verificare e valutare i questionari scelti dagli utenti in base alle risposte date.

La parte destinata ai creatori di questionari dovrà essere fruibile attraverso un *browserg* desktop, abilitato all'utilizzo delle tecnologie *HTML5G*, *CSS3G* e *JavaScriptG*. La parte destinata agli esaminandi sarà utilizzabile su qualunque dispositivo: dal personal computer ai tablet e smartphone.

1.3 Glossario

Al fine di evitare ogni ambiguità i termini tecnici del dominio del progetto, gli acronimi e le parole che necessitano di ulteriori spiegazioni saranno nei vari documenti marcate con il pedice G e quindi presenti nel documento *Glossario*.

1.4 Riferimenti

1.4.1 Normativi

- *NormeDiProgetto_v_3_0_0*;
- *AnalisiDeiRequisiti_v_2_0_0*;

1.4.2 Informativi

- **Ingegneria del software - Ian Sommerville - 8a edizione (2007)**:
Parte terza: Progettazione, capitolo 11: Progettazione architettonale, Capitolo 14: Progettazione orientata agli oggetti;
- **Design Patterns** - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - 1a edizione italiana (2006);
- **Slide dell'insegnamento - Design patterns**:
 - Strutturali: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E07.pdf>
 - Creazionali: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E08.pdf>
 - Comportamentali: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E09.pdf>
 - Architetturali:



* http://www.math.unipd.it/~rcardin/sweb/Design%20Pattern%20Architetturali%20-%20Model%20View%20Controller_4x4.pdf;
* http://www.math.unipd.it/~rcardin/sweb/Design%20Pattern%20Architetturali%20-%20Dependency%20Injection_4x4.pdf.

- Martin Fowler - **UML_G** Distilled - 2nd edition;
- Slide dell'insegnamento - Diagrammi delle classi:
<http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E03.pdf>
- Slide dell'insegnamento - Diagrammi dei packages:
<http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E04.pdf>
- Slide dell'insegnamento - Diagrammi di sequenza:
<http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E05.pdf>
- Documentazione del *Framework_G MEAN_G.js*:
<http://learn.mean.io/>
- Documentazione della *piattaforma Node.js*:
<https://nodejs.org/api/>
- Guida all'utilizzo dei middleware Express:
<http://expressjs.com/it/guide/using-middleware.html>
- Guida all'utilizzo dei middleware Passport:
<http://passportjs.org/docs>
- Manuale del database *MongoDB_G*:
<https://docs.mongodb.org/manual/>
- Documentazione dell'interfaccia REST:
 - Descrizione di REST: https://it.wikipedia.org/wiki/Representational_State_Transfer
 - Descrizione risorse REST: <http://stashboard.readthedocs.org/en/latest/restapi.html>
- Documentazione del *framework_G AngularJS_G*:
 - Documentazione generica: <https://docs.angularjs.org/guide>
 - Documentazione servizio \$http: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)
 - Documentazione servizio \$location: [https://docs.angularjs.org/api/ng/service/\\$location](https://docs.angularjs.org/api/ng/service/$location)
 - Documentazione servizio \$windows: [https://docs.angularjs.org/api/ng/service/\\$window](https://docs.angularjs.org/api/ng/service/$window)
 - Documentazione servizio \$routeParams: [https://docs.angularjs.org/api/ngRoute/service/\\$routeParams](https://docs.angularjs.org/api/ngRoute/service/$routeParams)
 - Documentazione servizio \$q: [https://docs.angularjs.org/api/ng/service/\\$q](https://docs.angularjs.org/api/ng/service/$q)
- Documentazione del *framework_G Material for Angular*:
<https://material.angularjs.org/latest/>



- Documentazione del $framework_G$ Chart.js
<http://www.chartjs.org/docs/>
- Documentazione del $wrapper_G$ Angles.js
<https://github.com/gonewandering/angles>
- Documentazione del $framework_G$ TextAngular.js
<https://github.com/fraywing/textAngular/wiki/textAngular-Docs-v1.1.x>
- Guida all'utilizzo della direttiva $ng-file-upload$
<https://github.com/danialfarid/ng-file-upload>
- Documentazione di $jison_G$ per la definizione della grammatica di QML
<http://zaa.ch/jison/docs/>



2 Specifica del Front-End

2.1 QuizziPedia::Front-End

2.1.1 Informazioni generali

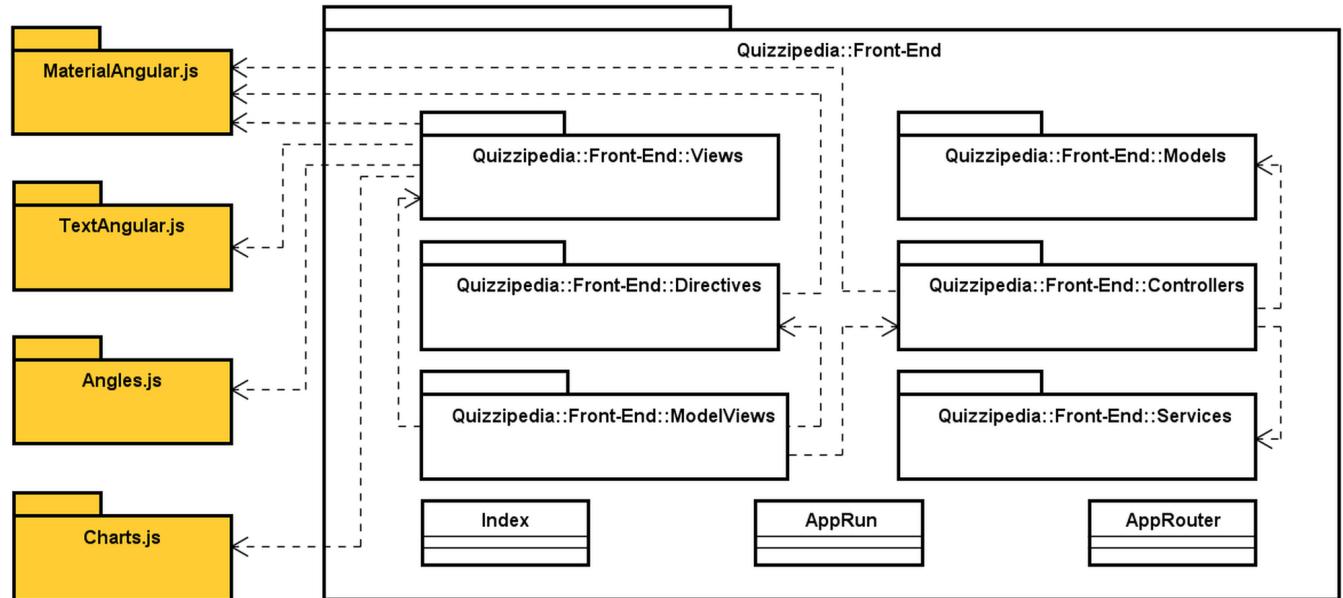


Figura 1: QuizziPedia::Front-End

- **Descrizione:** package_G contenente le componenti front-end dell'applicazione;
- **Package contenuti:**
 - **Controllers:** package_G contenente i *controllers_G* front-end dell'applicazione;
 - **Directives:** package_G contenente le *directives_G* front-end dell'applicazione;
 - **Models:** package_G contenente le classi che definiscono la business logic dell'applicazione;
 - **Services:** package_G contenente i *services_G* front-end dell'applicazione;
 - **Views:** package_G contenente le *views_G* front-end dell'applicazione;
 - **Templates:** package_G contenente i *templates_G* front-end dell'applicazione.

2.1.2 Classi

2.1.2.1 QuizziPedia::Front-End::Index



Figura 2: QuizziPedia::Front-End::Views::Index

- **Descrizione:** *view_G* generale dell'applicazione;
- **Utilizzo:** contiene gli elementi che saranno presenti in ogni pagina dell'applicazione;



- Relazioni con altre classi:

- **IN AppRun**: classe che verifica se l’utente sia autenticato e che abbia le giuste autorizzazioni per la pagina in cui si trova;
- **IN MenuBarDirective**: rappresenta il menù, presente in ogni pagina dell’applicazione, generato in base agli oggetti passati nello \$scope isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un’icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata;
- **IN FooterDirective**: *directive_G* che mostra il footer dell’applicazione che sarà presente in ogni pagina;
- **IN ClickableAreaQuestionsView**: *view_G* contenente i campi e le direttive per creare una domanda ad area cliccabile;
- **IN ConnectionQuestionsView**: *view_G* contenente i campi e le direttive per creare una domanda a collegamento;
- **IN CreateQuestionnaireView**: *view_G* per la creazione del questionario. In questo componente viene permesso anche all’utente di:
 - * Effettuare delle ricerche sul database di domande;
 - * Selezionare le domande da inserire nel questionario;
 - * Mostrare le domande già inserite e permettere all’utente di eliminarle da tale lista.
- **IN EditorQMLView**: *view_G* contenente l’editor QML per la creazione di domande personalizzate;
- **IN FillingQuestionnaireView**: *view_G* principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
- **IN FilligQuestionsView**: *view_G* contenente i campi e le direttive per creare una domanda a riempimento testo;
- **IN HomeView**: *view_G* contenente la direttiva per barra di ricerca degli utenti e questionari e il bottone che porterà l’utente nella modalità allenamento;
- **IN ImagesSortingQuestionsView**: *view_G* contenente i campi e le direttive per creare una domanda a ordinamento immagini;
- **IN LoginView**: *view_G* contenente le form necessarie per effettuare il login. Contiene inoltre un link alla pagina di registrazione e uno alla pagina per il recupero della password;
- **IN MultipleQuestionsView**: *view_G* contenente le direttive per creare una domanda a risposta multipla;
- **IN OtherUserView**: *view_G* contenente le direttive dei dati personali e delle statistiche di un utente ricercato;
- **IN PasswordForgotView**: *view_G* contenente le form necessarie per il recupero della password dimenticata;
- **IN ProfileManagementView**: *view_G* contenente i dati personali che un utente può modificare dopo essersi registrato al sistema;
- **IN QuestionnaireManagementView**: *view_G* principale per la gestione dei questionari;
- **IN QuestionsManagementView**: *view_G* contenente l’elenco delle domande create;
- **IN RegistrationManagementView**: *view_G* che permette di visualizzare gli utenti iscritti ad un questionario;
- **IN ResultsQuestionnaireView**: *view_G* contenente i risultati conseguiti dagli utenti che hanno compilato il proprio questionario;



- **IN ResultsView:** $view_G$ contenente i risultati della ricerca effettuata. Vengono visualizzati sia gli utenti che i questionari trovati;
- **IN SignUp:** $view_G$ contenente le form dedicate alla registrazione utente. Contiene inoltre un link alla pagina di login;
- **IN StringSortingQuestionsView:** $view_G$ contenente i campi e le direttive per creare una domanda a ordinamento stringhe;
- **IN TrainingView:** $view_G$ principale della modalità allenamento; conterrà i vari templates di ogni domanda dell’allenamento;
- **IN TrueFalseQuestionsView:** $view_G$ contenente le direttive per creare una domanda vero/falso;
- **IN UserView:** $view_G$ contenente le direttive dei dati personali dell’utente, delle sue statistiche relative ai questionari e agli allenamenti effettuati e dei questionari a cui è iscritto.

2.1.2.2 QuizziPedia::Front-End::AppRun

AppRun
<pre> - \$scope : \$scope - \$rootScope : \$rootScope - \$location : \$location - \$mdDialog : \$mdDialog - AuthService : AuthService + userOnScope : UserDetailsModelView - user : UserDetailsModel + lang : LangModel + AppRun(\$scope : \$scope, \$rootScope : \$rootScope, \$location : \$location, \$mdDialog : \$mdDialog, AuthService : AuthService, UserDetailsModel : UserDetailsModel) - getUserDetails(username : String) : UserDetailsModel - getLang(lang : String) : LangModel </pre>

Figura 3: QuizziPedia::Front-End::AppRun

- **Descrizione:** classe che verifica se l’utente sia autenticato e che abbia le giuste autorizzazioni per la pagina in cui si trova;
- **Utilizzo:** viene utilizzata per verificare che l’utente sia autenticato e che abbia la giusta autorizzazione per la pagina in cui si trova;
- **Relazioni con altre classi:**
 - **IN LangService:** questa classe permette di gestire la lingua nella quale si è scelto di utilizzare l’applicazione;
 - **IN LangModel:** rappresenta le informazioni per la giusta traduzione dell’applicazione;
 - **IN UserDetailsModel:** rappresenta un utente. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un utente sia nella visualizzazione che nella gestione di un profilo;
 - **IN AuthService:** questa classe permette di gestire la registrazione e l’autenticazione di un utente.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all’oggetto \$scope creato da $Angular_G$;
 - **- \$rootScope: \$rootScope**
Campo dati contenente il riferimento all’oggetto globale \$rootScope creato da $Angular_G$;



- - \$location: \$location
Campo dati contenente un riferimento al servizio creato da *AngularG* che permette di accedere alla barra degli indirizzi del *browserG*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- - \$mdDialog: \$mdDialog
Campo dati contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
- - AuthService: AuthService
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione;
- + userOnScope: UserDetailsModelView
Oggetto di tipo *UserDetailsModelView*. Rappresenta l'oggetto dell'utente autenticato all'interno dello *\$rootScope*;
- - user: UserDetailsModel
Oggetto di tipo *UserDetailsModel*. Rappresenta l'oggetto dell'utente autenticato;
- + lang: LangModel
Oggetto di tipo *LangModel*. Rappresenta l'oggetto contenente la giusta traduzione del template delle pagine.

- **Metodi:**

- + AppRun(\$scope: \$scope, \$rootScope: \$rootScope, \$location: \$location, \$mdDialog: \$mdDialog, AuthService: AuthService, UserDetailsModel: UserDetailsModel)
Metodo costruttore della classe. Recupera dopo il login tutte le informazioni dell'utente. Recupera anche da

Parametri:

- * \$scope: \$scope
Parametro contenente un riferimento all'oggetto *\$scope* creato da *AngularG*. Viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il.viewmodel e il model dell'applicazione;
 - * \$rootScope: \$rootScope
Parametro contenente il riferimento all'oggetto globale *\$rootScope* creato da *AngularG*;
 - * \$location: \$location
Parametro contenente un riferimento al servizio creato da *AngularG* che permette di accedere alla barra degli indirizzi del *browserG*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
 - * \$mdDialog: \$mdDialog
Parametro contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
 - * AuthService: AuthService
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione;
 - * UserDetailsModel: UserDetailsModel
Parametro contenente un riferimento alla classe per poter istanziare un oggetto di tipo *UserDetailsModel*.
- - getUserDetails(username: String): UserDetailsModel
Metodo che permette di ottenere i dati con una chiamata a *UserDetailsService*;

Parametri:



- * `username: String`: parametro che identifica l'utente del quale saranno scaricati i dati.
- `- getLang(lang: String): LangModel`
Metodo che permette di ottenere i dati con una chiamata a `LangService`;
- Parametri:**
- * `lang: String`: parametro che identifica la lingua del sistema.

2.1.2.3 QuizziPedia::Front-End::AppRouter

AppRouter
- <code>\$routeProvider: \$routeProvider</code>
- <code>\$locationProvider: \$locationProvider</code>
+ <code>appRouter(\$routeProvider: \$routeProvider, \$locationProvider: \$locationProvider)</code>

Figura 4: QuizziPedia::Front-End::AppRouter

- **Descrizione:** classe che gestisce i routes dell'applicazione, utilizza il servizio `$routeProvider` per associare ad ogni route un *controller_G* e una *view_G*;
- **Utilizzo:** viene utilizzata per associare un URL alle varie view dell'applicazione;
- **Attributi:**
 - `$routeProvider: $routeProvider`
Campo dati contenente un riferimento al servizio di *Angular_G* che si occupa di definire le route dell'applicazione;
 - `$locationProvider: $locationProvider`
Campo dati contenente un riferimento al servizio di *Angular_G* che si occupa di configurare come i paths vengono memorizzati.
- **Metodi:**
 - + `AppRouter($routeProvider: $routeProvider, $locationProvider: $locationProvider):`
Metodo che gestisce i routes dell'applicazione. Utilizza il servizio `$routeProvider` per associare ad ogni route un *controller_G* e una *view_G*; e `$locationProvider` per configurare come i paths dell'applicazione vengono salvati. Questa funzione viene utilizzata come parametro nel metodo `textttconfig` di *Angular_G*. Il metodo `config` permette di impostare l'esecuzione di una funzione al caricamento del *modulo_G* principale di *Angular_G*;
 - Parametri:**
 - * `$routeProvider: $routeProvider`: campo dati contenente un riferimento al servizio di *Angular_G* che si occupa di definire le route dell'applicazione;
 - * `$locationProvider: $locationProvider`: campo dati contenente un riferimento al servizio di *Angular_G* che si occupa di configurare come i paths vengono memorizzati.



2.2 QuizziPedia::Front-End::Models

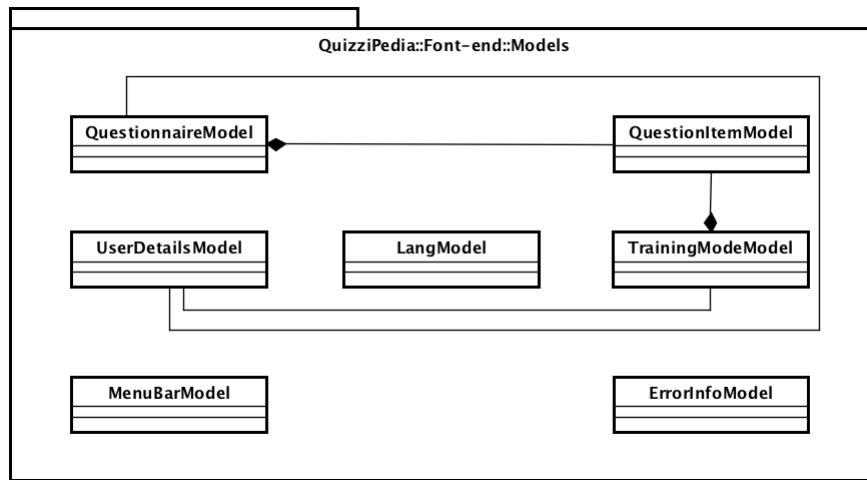


Figura 5: QuizziPedia::Front-End::Models

2.2.1 Informazioni generali

- **Descrizione:** *package_G* contenente le classi che definiscono la business logic dell'applicazione;
- **Padre:** Front-End;
- **Iterazioni con altri componenti:**
 - **Controllers:** *package_G* contenente i controllers front-end dell'applicazione;
 - **Directives:** *package_G* contenente le directives front-end dell'applicazione;
 - **Models:** *package_G* contenente le classi che definiscono la business logic dell'applicazione;
 - **Templates:** *package_G* contenente i templates necessari per la creazione dinamica delle viste per le domande;
 - **Services:** *package_G* che contiene le classi individuate che permettono la comunicazione del lato front-end con il lato back-end attraverso l'architettura REST_G.

2.2.2 Classi

2.2.2.1 QuizziPedia::Front-End::Models::UserDetailsModel

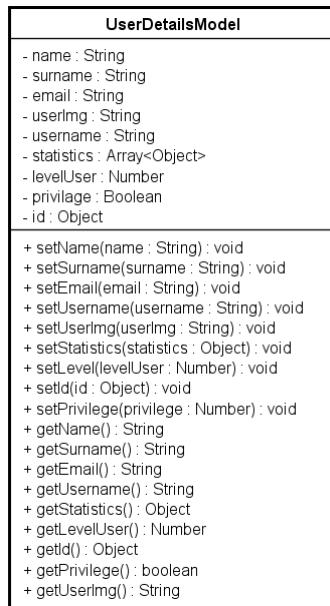


Figura 6: QuizziPedia::Front-End::Models::UserDetailsModel

- **Descrizione:** rappresenta un utente. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un utente sia nella visualizzazione che nella gestione di un profilo;
- **Utilizzo:** viene utilizzata per memorizzare i dati di un utente;
- **Relazioni con altre classi:**
 - **OUT LoginController:** questa classe permette di gestire l'autenticazione dell'utente al sistema;
 - **OUT SearchController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione;
 - **OUT UserDetailsController:** questa classe permette di gestire i dati di un utente;
 - **OUT StatisticsController:** questa classe permette di le statistiche di un utente.
- **Attributi:**
 - - name: String
Rappresenta il nome dell'utente registrato;
 - - surname: String
Rappresenta il cognome dell'utente registrato;
 - - email: String
Rappresenta l'email dell'utente registrato;
 - - userImg: String
Rappresenta il path della foto profilo dell'utente registrato;
 - - username: String
Rappresenta l'username con cui viene identificato l'utente all'interno dell'applicazione;
 - - statistics: Array<Object>
Contenente i seguenti attributi:
 - * - topicName: String
Rappresenta il nome della statistica relativa all'argomento;



```
* - topicLevel: Number
    Identifica il livello di preparazione dell'utente in un determinato argomento;
* - correctAnswers: Number
    Identifica il numero di risposte corrette date dall'utente riguardanti domande di un determinato argomento;
* - totalAnswers: Number
    Identifica il numero di risposte totali date dall'utente riguardanti domande di un determinato argomento.

- - levelUser: Number
    Identifica il livello dell'utente;
- - privilege: Boolean
    Identifica la tipologia dell'utente;
- - id: Object Identifica l'id dell'utente;
```

• **Metodi:**

```
- + setName(name: String): void
    Metodo setterG per il campo dati name.
```

Parametri:

```
* name: String
    Questo parametro contiene il nome dell'utente.
```

```
- + setSurname(surname: String): void
    Metodo setterG per il campo dati surname.
```

Parametri:

```
* surname: String
    Questo parametro contiene il cognome dell'utente.
```

```
- + setEmail(email: String): void
    Metodo setterG per il campo dati email.
```

Parametri:

```
* email: String
    Questo parametro contiene l'indirizzo email dell'utente.
```

```
- + setUsername(username: String): void
    Metodo setterG per il campo dati username.
```

Parametri:

```
* username: String
    Questo parametro contiene lo username dell'utente.
```

```
- + setUserImg(userImg: String): void
    Metodo setterG per il campo dati userImg.
```

Parametri:

```
* userImg: String
    Questo parametro contiene l'url dell'immagine profilo dell'utente.
```

```
- + setStatistics(statistics: Object): void
    Metodo setterG per il campo dati statistics.
```

Parametri:

```
* statistics: Object
    Questo parametro contiene le statistiche dell'utente.
```

```
- + setLevel(levelUser: Number): void
    Metodo setterG per il campo dati userLever.
```

Parametri:



- * levelUser: Number
Questo parametro contiene il livello dell'utente.
- + setId(id: Object): void
Metodo *setter_G* per il campo dati id.
Parametri:
 - * id: ObjectId
Questo parametro contiene l'id dell'utente.
- + setPrivilege(privilege: Number): void
Metodo *setter_G* per il campo dati privilege.
Parametri:
 - * privilege: Number
Questo parametro rappresenta la tipologia di utente.
- + getName(): String
Metodo *getter_G* che restituisce il campo dati name;
- + getSurname(): String
Metodo *getter_G* che restituisce il campo dati surname;
- + getEmail(): String
Metodo *getter_G* che restituisce il campo dati email;
- + getUsername(): String
Metodo *getter_G* che restituisce il campo dati username;
- + getStatistics(): Object
Metodo *getter_G* che restituisce il campo dati statistics;
- + getLevelUser(): Number
Metodo *getter_G* che restituisce il campo dati levelUser;
- + getId(): Object
Metodo *getter_G* che restituisce il campo dati id;
- + getPrivilege(): boolean
Metodo *getter_G* che restituisce il campo dati privilege;
- + getUserImg(): String
Metodo *getter_G* che restituisce il campo dati userImg;

2.2.2.2 QuizziPedia::Front-End::Models::TrainingModeModel

TrainingModeModel
<ul style="list-style-type: none"> - argument: String - keywords: Array<String> - numberOfQuestions: Number - questions: Array<QuestionItemModel> - rightAnswer: Number <ul style="list-style-type: none"> + setArgument(argument: String): void + setNumberOfQuestions(numberOfQuestions: Number): void + getArgument(): String + getNumberOfQuestions(): Number + addQuestion(question: int): void + removeQuestion(idQuestion: ObjectId): void + addKeyword(keyword: String): void + removeKeyword(keyword: String): void + getResult(): Object

Figura 7: QuizziPedia::Front-End::Models::TrainingModeModel

- **Descrizione:** rappresenta un allenamento. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un allenamento;



- **Utilizzo:** viene utilizzata per memorizzare i dati di un allenamento;
- **Relazioni con altre classi:**
 - **OUT TrainingController:** questa classe permette di gestire la modalità allenamento sottponendo all'utente le giuste domande adatte al suo livello;
- **Attributi:**
 - **- argument: String**
Questo attributo rappresenta l'argomento dell'allenamento;
 - **- keywords: Array<String>**
Questo attributo rappresenta l'**array** contenente le parole chiave per l'allenamento;
 - **- numberofQuestions: Number**
Questo attributo rappresenta il numero di domande che l'utente si è prefissato di rispondere per l'allenamento. Se -1 allora non esiste un numero di domande impostate, perciò l'allenamento non terminerà fino alla chiusura manuale dell'attività;
 - **- questions: Array<QuestionItemModel>**
Questo attributo contiene l'**array** di **QuestionItemModel** che rappresenta le domande dell'allenamento fino a quel momento visualizzate, compresa quella visualizzata;
 - **- rightAnswer: Number**
Questo attributo rappresenta il numero di domande corrette date.
- **Metodi:**
 - **+ setArgument(argument: String): void**
Metodo *setter_G* per il campo dati **argument**.
Parametri:
 - * **argument: String**
Questo parametro contiene l'argomento dell'allenamento.
 - **+ setNumberofQuestions(numberofQuestions: Number): void**
Metodo *setter_G* per il campo dati **numberofQuestions**.
Parametri:
 - * **author: ObjectId**
Questo parametro rappresenta l'autore che ha creato la domanda.
 - **+ getArgument(): String**
Metodo *getter_G* che restituisce l'argomento dell'allenamento;
 - **+ getNumberofQuestions(): Number**
Metodo *getter_G* che restituisce il campo dati **numberofQuestions**;
 - **+ addQuestion(question: QuestionItemModel): ObjectId**
Metodo per aggiungere una domanda all'**array** di domande.
Parametri:
 - * **question: QuestionItemModel**
Questo parametro rappresenta la domanda da inserire.
 - **+ removeQuestion(idQuestion: ObjectId): void**
Metodo per poter eliminare una domanda dall'allenamento.
Parametri:
 - * **idQuestion: ObjectId**
Questo parametro rappresenta l'id della domanda che si vuole rimuovere dall'allenamento.



- `+ addKeyword(keyword: String): void`
Metodo per aggiungere una parola chiave all'array di parole chiavi.
- Parametri:**

 - * keyword: String
Questo parametro rappresenta la parola chiave da inserire.

- `+ removeKeyword(keyword: String): void`
Metodo per poter eliminare una parola chiave dall'array di parole chiavi.
- Parametri:**

 - * keyword: String
Questo parametro rappresenta la parola chiave da eliminare.

- `+ getResult(): Object`
Metodo *getter* che restituisce risultato fino a quel momento dell'allenamento, ritornando un oggetto contenente il numero di domande giuste e totali.

2.2.2.3 QuizziPedia::Front-End::Models::QuestionnaireModel

QuestionnaireModel
<ul style="list-style-type: none"> - title : String - author : ObjectId - questions : Array<QuestionItemModel>
<ul style="list-style-type: none"> + setTitle(title : String) : void + setAuthor(author : ObjectId) : void + getTitle() : String + getAuthor() : ObjectId + getQuestion() : Array<QuestionItemModel> + addQuestion(question : QuestionItemModel) : ObjectId + removeQuestion(idQuestion : ObjectId) : void

Figura 8: QuizziPedia::Front-End::Models::QuestionnaireModel

- **Descrizione:** rappresenta un questionario. Contiene tutte le informazioni necessarie alla presentazione del contenuto del questionario;
- **Utilizzo:** viene utilizzata per memorizzare i dati di un questionario;
- **Relazioni con altre classi:**
 - **IN SearchController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione;
 - **IN QuestionnaireDetailsController:** questa classe permette di gestire tutti i questionari creati da un utente;
 - **IN FillingQuestionnaireController:** questa classe permette di gestire la creazione e la modifica di una domanda a riempimento di spazi;
 - **IN CreateQuestionnaireController:** questa classe permette di gestire la creazione di un questionario;
 - **IN RegistrationManagementController:** questa classe permette di gestire le iscrizione degli utenti ai questionari;
 - **IN ResultsController:** questa classe permette di gestire i risultati della ricerca effettuata dall'utente.
- **Attributi:**



- - title: String
Questo attributo rappresenta il titolo del questionario;
- - author: ObjectId
Questo attributo rappresenta l'autore del questionario;
- - questions: Array<QuestionItemModel>
Questo attributo contiene l'array di QuestionItemModel che rappresenta le domande di un questionario.

• **Metodi:**

- + setTitle(title: String): void
Metodo *setter_G* per il titolo del questionario.

Parametri:

- * title: String
Questo parametro contiene il titolo del questionario.

- + setAuthor(author: ObjectId): void
Metodo *setter_G* per il campo dati **author**.

Parametri:

- * author: ObjectId
Questo parametro rappresenta l'autore che ha creato la domanda.

- + getTitle(): String
Metodo *getter_G* che restituisce il titolo del questionario;

- + getAuthor(): ObjectId
Metodo *getter_G* che restituisce il campo dati **author**;

- + getQuestion(): Array<QuestionItemModel>
Metodo *getter_G* che restituisce il campo dati **questions**;

- + addQuestion(question: QuestionItemModel): ObjectId
Metodo per aggiungere una domanda all'array di domande.

Parametri:

- * question: QuestionItemModel
Questo parametro rappresenta la domanda da inserire.

- + removeQuestion(idQuestion: ObjectId): void
Metodo per poter eliminare una domanda dal questionario.

Parametri:

- * idQuestion: ObjectId
Questo parametro rappresenta l'id della domanda che si vuole rimuovere dal questionario.

2.2.2.4 QuizziPedia::Front-End::Models::QuestionItemModel

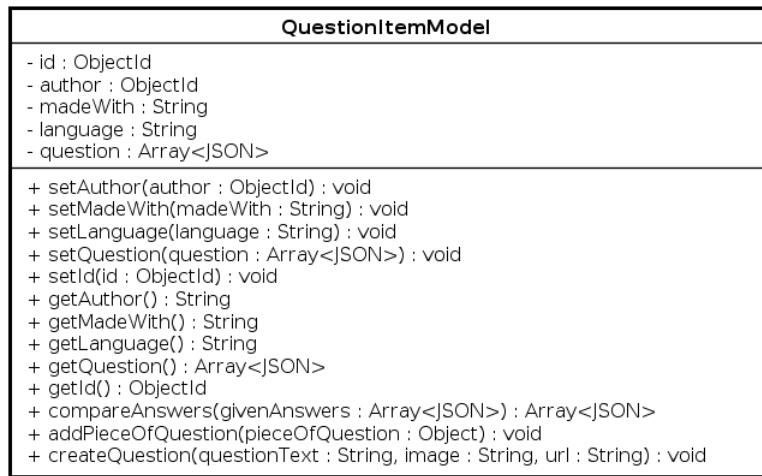


Figura 9: QuizziPedia::Front-End::Models::QuestionItemModel

- **Descrizione:** rappresenta una domanda. Contiene tutte le informazioni necessarie alla presentazione del contenuto della domanda;
- **Utilizzo:** viene utilizzata per memorizzare i dati di una domanda;
- **Relazioni con altre classi:**
 - **OUT QuestionsManagementController:** questa classe permette di gestire e di ottenere le domande create dall'utente;
 - **OUT TrueFalseQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda vero/falso;
 - **OUT MultipleQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a risposta multipla;
 - **OUT ConnectionQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a collegamento;
 - **OUT ImagesSortingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento immagini;
 - **OUT StringsSortingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento di stringhe;
 - **OUT FillingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a riempimento di spazi;
 - **OUT ClickableAreaQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda ad area cliccabile;
 - **OUT EditorQMLController:** questa classe permette di gestire la creazione e la modifica di domande create tramite editor QML;
 - **OUT QuestionsController:** questa classe permette di gestire il recupero delle domande per poterle stampare nella modalità allenamento e nei questionari;
- **Attributi:**
 - - id: ObjectId

Rappresenta l'attributo id della domanda. Viene utilizzato solamente dalle domande recuperate dal database;



- - **author:** `ObjectId`
Rappresenta il riferimento all'identificativo dell'utente che ha creato la domanda;
- - **madeWith:** `String`
Rappresenta con quale strumento è stata creata la domanda;
- - **language:** `String`
Rappresenta la lingua in cui è scritta la domanda;
- - **question:** `Array<JSON>`
Contiene un oggetto di tipo $JSON_G$. L'oggetto $JSON_G$ è rappresentato dai seguenti campi:
 - * - **type:** `String`
Rappresenta la tipologia di domanda;
 - * - **questionText:** `String`
Rappresenta il testo della domanda;
 - * - **image:** `String`
Rappresenta l' URL_G dell'immagine associata al testo della domanda;
 - * - **answers:** `Array<JSON>`
Contiene un oggetto di tipo $JSON_G$. L'oggetto $JSON_G$ è rappresentato dai seguenti campi:
 - - **text:** `String`
Rappresenta il testo della risposta;
 - - **url:** `String`
Rappresenta l'immagine della risposta;
 - **attributesForTForMultiple:** `Mixed`
Contiene i seguenti attributi:
 1. - **isItRight:** `Boolean`
Rappresenta se una risposta è giusta o sbagliata.
 - **attributesForSorting:** `Mixed`
Contiene i seguenti attributi:
 1. - **position:** `Boolean`
Rappresenta quale è la giusta posizione di un testo o immagine all'interno di un esercizio di ordinamento.
 - **attributesForLinking:** `Mixed`
Contiene i seguenti attributi:
 1. - **text1:** `String`
Rappresenta il primo elemento testuale che deve essere collegato con il secondo elemento testuale o rappresentato da un'immagine;
 2. - **text2:** `String`
Rappresenta il secondo elemento testuale che deve essere collegato con il primo elemento testuale o rappresentato da un'immagine;
 3. - **url1:** `String`
Rappresenta il primo elemento rappresentato da un'immagine che deve essere collegato con il secondo elemento testuale o rappresentato da un'immagine;
 4. - **url2:** `String`
Rappresenta il secondo elemento rappresentato da un'immagine che deve essere collegato con il primo elemento testuale o rappresentato da un'immagine.
 - **attributesForClickableArea:** `Mixed`
Contiene i seguenti attributi:



1. - **x:** Number
Rappresenta la coordinata x di una area cliccabile;
2. - **y:** Number
Rappresenta la coordinata y di una area cliccabile.
- **attributesForEmptySpaces:** Mixed
Contiene i seguenti attributi:
 1. - **wordNumber:** Number
Rappresenta la posizione dello spazio vuoto in cui deve andare inserita la parola.

• **Metodi:**

- + **setAuthor(author: ObjectId): void**
Metodo *setter_G* per il campo dati **author**.

Parametri:

- * **author:** ObjectId

Questo parametro rappresenta l'autore che ha creato la domanda.

- + **setMadeWith(madeWith: String): void**
Metodo *setter_G* per il campo dati **madeWith**.

Parametri:

- * **madeWith:** String

Questo parametro rappresenta con quale strumento è stata creata la domanda.

- + **setLanguage(language: String): void**
Metodo *setter_G* per il campo dati **language**.

Parametri:

- * **language:** String

Questo parametro rappresenta la lingua della domanda.

- + **setQuestion(question: Array<JSON>): void**
Metodo *setter_G* per il campo dati **question**

Parametri:

- * **question:** Array<JSON>

Questo parametro rappresenta l'oggetto *JSON_G* contenente le parti che compongono la totalità della domanda.

- + **setId(id: ObjectId): void**
Metodo *setter_G* per il campo dati **id**

Parametri:

- * **id:** ObjectId

Questo parametro rappresenta l'id della domanda.

- + **getAuthor(): String**
Metodo *getter_G* per il campo dati **author**;

- + **getMadeWith(): String**
Metodo *getter_G* per il campo dati **madeWith**;

- + **getLanguage(): String**
Metodo *getter_G* per il campo dati **language**;

- + **getQuestion(): Array<JSON>**
Metodo *getter_G* per il campo dati **question**;

- + **getId(): ObjectId**
Metodo *getter_G* per il campo dati **id**;



– + compareAnswers(givenAnswers: Array<JSON>): Array<JSON>

Metodo che compara le risposte date con quelle corrette. Ritorna un array contenente un valore booleano, che indica se è giusta o meno una domanda, per ogni pezzo di domanda che compone la domanda.

Parametri:

* givenAnswers: Array<JSON>

Questo parametro contiene le risposte date dall'utente.

– + addPieceOfQuestion(pieceOfQuestion: Object): void

Metodo che permette di inserire un pezzo di domanda all'attributo `question`.

Parametri:

* pieceOfQuestion: Object

Questo parametro rappresenta un pezzo di domanda.

– + createQuestion(questionText: String, image: String, url: String): void

Metodo che inizializza il campo `question`.

Parametri:

* questionText: String

Rappresenta il testo della domanda;

* image: String

Rappresenta l' URL_G dell'immagine associata al testo della domanda.

2.2.2.5 QuizziPedia::Front-End::Models::MenuBarModel

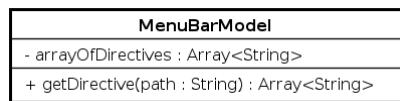


Figura 10: QuizziPedia::Front-End::Models::MenuBarModel

- **Descrizione:** questa classe racchiude i dati necessari per la creazione dinamica della barra menù posizionata in modo fisso su ogni pagina;

- **Utilizzo:** viene utilizzata per memorizzare i dati necessari per la creazione dinamica della barra menù posizionata in modo fisso su ogni pagina;

- **Relazioni con altre classi:**

- IN `MenuBarController`: questa classe permette di gestire il menù fisso per ogni pagina;

- **Attributi:**

- - `arrayOfDirectives: Array<String>`

Questo attributo contiene l'array di direttive possibili per la creazione dinamica della barra menù posizionata in modo fisso su ogni pagina;

- **Metodi:**

- + `getDirective(path: String): Array<String>`

Metodo `getter_G` che restituisce un array di `String` contenente le giuste direttive per quella pagina;



2.2.2.6 QuizziPedia::Front-End::Models::LangModel

LangModel
- lang : String - listOfKeywords : Array<String>
+ LangModel(language : String, listOfKeys : Array<String>) + getLang() : String + getListOfKeywords() : Array<String> + setNewLang(language : String, listOfKeys : Array<String>) : void

Figura 11: QuizziPedia::Front-End::Models::LangModel

- **Descrizione:** rappresenta le informazioni per la giusta traduzione dell'applicazione;
- **Utilizzo:** viene utilizzata per racchiudere tutte le informazioni riguardanti la giusta traduzione dell'applicazione;
- **Relazioni con altre classi:**
 - IN AppRun: questa classe si preoccupa di creare la prima istanza dell'applicazione.
- **Attributi:**
 - - lang: String
Questo attributo rappresenta la lingua con il quale il sistema verrà visualizzato;
 - - listOfKeywords: Array<String>
Questo attributo è un array di String che contiene il set di parole chiave che compongono la struttura dell'applicazione, necessarie per la traduzione delle pagine.
- **Metodi:**
 - + LangModel(language: String, listOfKeys: Array<String>)
Metodo costruttore della classe.
Parametri:
 - * language: String
Questo parametro rappresenta la lingua con il quale il sistema verrà visualizzato;
 - * listOfKeys: Array<String>
Questo parametro è un array di String che contiene il set di parole chiave che compongono la struttura dell'applicazione, necessarie per la traduzione delle pagine.
 - + getLang(): String
Metodo che ritorna la lingua del sistema;
 - + getListOfKeywords(): Array<String>
Metodo che ritorna la lista delle keywords nella lingua corrente del sistema;
 - + setNewLang(language: String, listOfKeys: Array<String>): void
Metodo che setta la giusta lingua e la lista delle keywords.
Parametri:
 - * language: String
Questo parametro rappresenta la lingua con il quale il sistema verrà visualizzato;
 - * listOfKeys: Array<String>
Questo parametro è un array di String che contiene il set di parole chiave che compongono la struttura dell'applicazione, necessarie per la traduzione delle pagine.



2.2.2.7 QuizziPedia::Front-End::Models::ErrorInfoModel

ErrorInfoModel
<pre>- errorCode : Number - errorMessage : String - errorTitle : String</pre>
<pre>+ ErrorModel(errorCode : Number, errorTitle : String, errorMessage : String) : void + getCode() : Number + getMessage() : String + getTitle() : String</pre>

Figura 12: QuizziPedia::Front-End::Models::ErrorInfoModel

- **Descrizione:** rappresenta le informazioni di un errore che si è verificato eseguendo una determinata operazione;
- **Utilizzo:** viene utilizzata rappresentare un oggetto di tipo errore;
- **Relazioni con altre classi:**
 - **OUT AuthService:** questa classe permette di gestire la registrazione e l'autenticazione di un utente;
 - **OUT SearchService:** questa classe permette di gestire il recupero dei dati dal backend a seguito di una ricerca effettuata da un utente;
 - **OUT LangService:** questa classe permette di gestire la lingua nella quale si è scelto di utilizzare l'applicazione.
 - **OUT QuizService:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca. Permette inoltre di iscriversi ad un questionario e di scaricare l'intera lista di domande di un questionario a partire dal suo id univoco;
 - **OUT StatisticsService:** questa classe permette di ottenere le statistiche dell'utente;
 - **OUT QuestionsService:** questa classe permette di ottenere domande esistenti e salvare nuove domande;
 - **OUT UserDetailsService:** questa classe permette di ottenere i dati personali degli utenti.
- **Attributi:**
 - **- errorCode: Number**
Rappresenta il codice dell'errore;
 - **- errorMessage: String**
Rappresenta la descrizione dell'errore;
 - **- errorTitle: String**
Rappresenta il titolo del messaggio d'errore.
- **Metodi**
 - **+ ErrorModel(errorCode: Number, errorTitle: String, errorMessage: String) : void**
Metodo costruttore della classe.
Parametri:



```
* errorCode: Number
    Parametro contenente il codice di errore;
* errorTitle: String
    Parametro contenente il titolo di errore;
* errorMessage: String
    Parametro contenente il messaggio di errore.

- + getCode(): Number
    Metodo che consente di ottenere il codice dell'errore;
- + getMessage(): String
    Metodo che consente di ottenere la descrizione dell'errore;
- + getTitle(): String
    Metodo che consente di ottenere il titolo del messaggio d'errore.
```



2.3 QuizziPedia::Front-End::Controllers

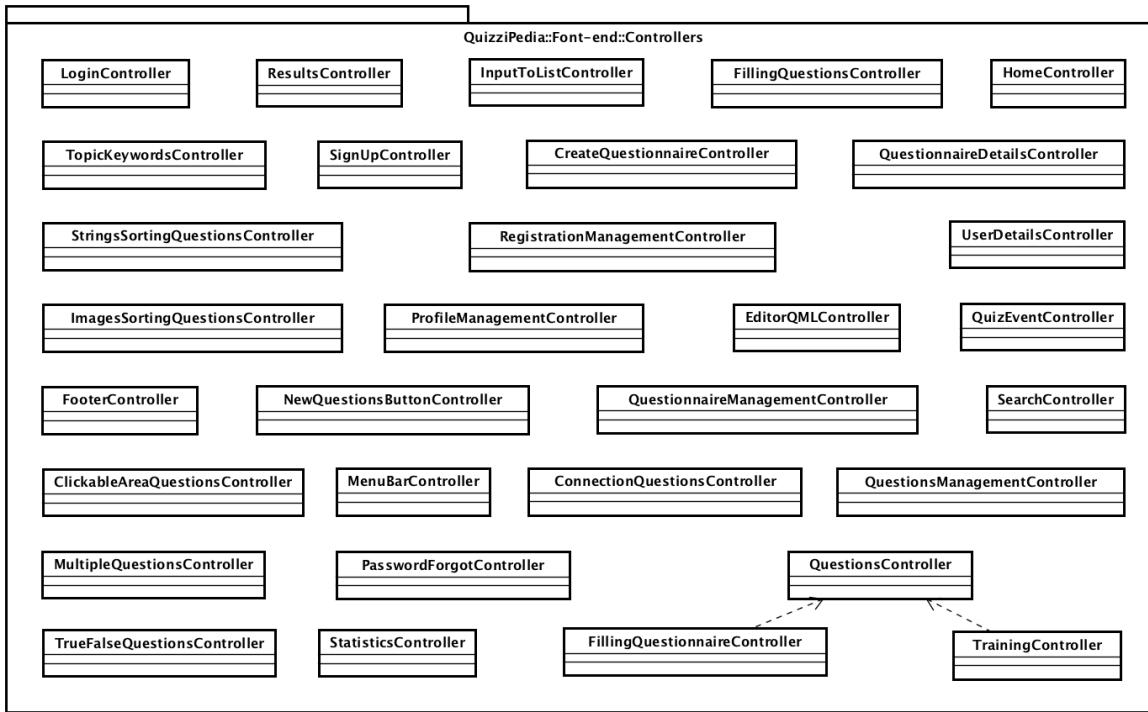


Figura 13: QuizziPedia::Front-End::Controllers

2.3.1 Informazioni generali

- Descrizione:** *package_G* che contiene i controller individuati per la parte front-end dell'applicazione;
- Padre:** Front-End;
- Interazione con altri componenti:**
 - Models:** *package_G* che contiene le classi model individuate;
 - Services:** *package_G* che contiene le classi services individuate.

2.3.2 Classi

2.3.2.1 QuizziPedia::Front-End::Controllers::ClickableAreaQuestionsController

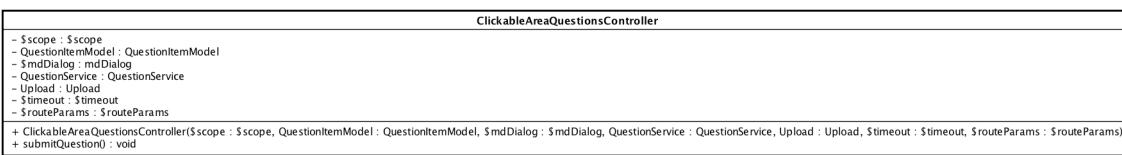


Figura 14: QuizziPedia::Front-End::Controllers::ClickableAreaQuestionsController

- Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda ad area cliccabile;



• **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda ad area cliccabile nel database e per modificarne una esistente;

• **Relazioni con altre classi:**

- **IN ClickableAreaQuestionsModelView:** classe di tipo `modelview` la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la *view_G* `ClickableAreaQuestionsView` e il *controller_G* `ClickableAreaQuestionsController`;
- **IN QuestionService:** questa classe permette di:
 - * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
- **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda.

• **Attributi:**

- - `$scope: $scope`
Campo dati contenente un riferimento all'oggetto `$scope` creato da *AngularG*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il model dell'applicazione;
- - `QuestionItemModel: QuestionItemModel`
Campo dati che si riferisce alla classe che rappresenta il modello della classe;
- - `$mdDialog: $mdDialog`
Campo dati contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
- - `QuestionService: QuestionService`
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
- - `Upload: Upload`
Campo dati contenente un riferimento alla libreria *ng-file-uploadG* necessaria per il caricamento della foto profilo dell'utente;
- - `$timeout: $timeout`
Campo dati contenente il riferimento all'oggetto globale `$timeout` creato da *AngularG*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una *promiseG*, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;
- - `$routeParams: $routeParams`
Campo dati contenente il riferimento all'oggetto globale `$routeParams` creato da *AngularG*. Tale servizio permette di recuperare il set di variabili presenti nell'url.

• **Metodi:**

- + `ClickableAreaQuestionsController($scope: $scope, QuestionItemModel: QuestionItemModel, $mdDialog: $mdDialog, QuestionService: QuestionService, Upload: Upload, $timeout: $timeout, $routeParams: $routeParams)`
Metodo costruttore della classe. Se in `$routeParams` sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il `QuestionService` il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso `$mdDialog`



indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in `$routeParams` verrà caricata la `view_G` vuota così da poter creare una nuova domanda;

Parametri:

* `$scope: $scope`

Parametro contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il `controller_G` e la `view_G`. Contiene gli oggetti che definiscono il `model_G` dell'applicazione;

* `QuestionItemModel: QuestionItemModel`

Parametro che si riferisce alla classe che rappresenta il modello della classe;

* `$mdDialog: $mdDialog`

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

* `QuestionService: QuestionService`:

Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;

* `Upload: Upload`

Parametro contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;

* `$timeout: $timeout`

Parametro contenente il riferimento all'oggetto globale `$timeout` creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una `promise_G`, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;

* `$routeParams: $routeParams`

Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url.

- + `submitQuestion(): void`

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal `modelview_G` e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando;

- + `chooseThatPoint(x: Number, y: Number): void`

Metodo che gestisce l'evento click su un punto dell'immagine. Una volta selezionato esso verrà inserito nell'array di punti;

Parametri:

* `x: Number`

Parametro contenente la coordinata x del punto;

* `y: Integer`

Parametro contenente la coordinata y del punto.

2.3.2.2 QuizziPedia::Front-End::Controllers::ConnectionQuestionsController



ConnectionQuestionsController
<pre> - \$scope : \$scope - QuestionItemModel : QuestionItemModel - \$mdDialog : mdDialog - QuestionService : QuestionService - Upload : Upload - \$timeout : \$timeout - \$routeParams : \$routeParams + ConnectionQuestionsController(\$scope : \$scope, QuestionItemModel : QuestionItemModel, \$mdDialog : \$mdDialog, QuestionService : QuestionService, Upload : Upload, \$timeout : \$timeout, \$routeParams : \$routeParams) + submitQuestion() : void </pre>

Figura 15: QuizziPedia::Front-End::Controllers::ConnectionQuestionsController

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda a collegamento;
- **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda a collegamento nel database e per modificarne una esistente;
- **Relazione con altre classi:**
 - **IN ConnectionQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G ConnectionQuestionsView* e il *controller_G ConnectionQuestionsController*;
 - **IN QuestionService:** questa classe permette di:
 - * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
 - **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
 - **- QuestionItemModel: QuestionItemModel**
Campo dati che si riferisce alla classe che rappresenta il modello della classe;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - **- QuestionService: QuestionService:**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
 - **- Upload: Upload**
Campo dati contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;
 - **- \$timeout: \$timeout**
Campo dati contenente il riferimento all'oggetto globale `$timeout` creato da *Angular.js_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una promise, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;
 - **- \$routeParams: \$routeParams**
Campo dati contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url;



- **Metodi:**

- + ConnectionQuestionsController(\$scope: \$scope, QuestionItemModel: QuestionItemModel, \$mdDialog: \$mdDialog, QuestionService: QuestionService, Upload: Upload, \$timeout: \$timeout, \$routeParams: \$routeParams)

Metodo costruttore della classe. Se in `$routeParams` sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il `QuestionService` il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso `$mdDialog` indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in `$routeParams` verrà caricata la view vuota così da poter creare una nuova domanda;

Parametri:

- * `$scope: $scope`

Parametro contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il `controllerG` e la `viewG`. Contiene gli oggetti che definiscono il model dell'applicazione;

- * `QuestionItemModel: QuestionItemModel`

Parametro che si riferisce alla classe che rappresenta il modello della classe;

- * `$mdDialog: $mdDialog`

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- * `QuestionService: QuestionService`:

Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;

- * `Upload: Upload`

Parametro contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;

- * `$timeout: $timeout`

Parametro contenente il riferimento all'oggetto globale `$timeout` creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una promise, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;

- * `$routeParams: $routeParams`

Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url.

- + `submitQuestion(): void`

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.3.2.3 QuizziPedia::Front-End::Controllers::CreateQuestionnaireController



CreateQuestionnaireController
- \$scope : \$scope
- \$rootscope : \$rootscope
- \$mdDialog : \$mdDialog
- QuizService : QuizService
- QuestionsService : QuestionsService
+ CreateQuestionnaireController(\$scope : \$scope, \$rootscope : \$rootscope, \$mdDialog : \$mdDialog, QuizService : QuizService)
+ createQuestionnaire(title : String, quiz : QuestionnaireModel) : void
+ modifyQuestionnaire(quid : QuestionnaireModel) : void
+ getQuestionnaire(quid : String) : QuestionnaireModel
+ getQuestionnairePreview(username : String) : Array<QuestionnaireModel>
+ deleteQuestionnaire(quid : String) : void
+ getQuestions() : Array<String>
+ getQuestion(questionId : String) : QuestionItemModel
+ addQuestion(questionId : String) : void

Figura 16: QuizziPedia::Front-End::Controllers::CreateQuestionnaireController

- **Descrizione:** questa classe permette di gestire la creazione di un questionario;
- **Utilizzo:** fornisce tutte le funzionalità per la creazione di un nuovo questionario e per la modifica di uno esistente;
- **Relazione con altre classi:**
 - **IN CreateQuestionnaireModelView:** oggetto di tipo `CreateQuestionnaireModelView`. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* tra la `view_G CreateQuestionnaireView` e il `controller_G CreateQuestionnaireController`;
 - **IN QuizService:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca;
 - **IN QuestionnaireModel:** `model_G` relativo ai questionari;
 - **IN QuestionsService:** questa classe permette di ottenere le domande dal back-end.
- **Attributi:**
 - **- scope: Scope**
Campo dati contenente un riferimento all'oggetto `$scope` creato da `Angular_G`, viene utilizzato come mezzo di comunicazione tra il `controller_G` e la `view_G`. Contiene gli oggetti che definiscono il `model_G` dell'applicazione;
 - **- \$rootScope: \$rootScope**
Campo dati contenente il riferimento all'oggetto globale `$rootScope` creato da `Angular_G`. Viene utilizzato per rendere accessibile a tutti i `controller_G` e a tutte le `view_G` l'oggetto `QuestionnaireModel`. In questo caso viene utilizzato per inserire in `$rootScope` l'oggetto di ritorno della chiamata a `getQuestionnaire` e la lista dei questionari ottenuta dalla chiamata `getQuestionnairePreview`;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - **- QuizService**
questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca;
- **Metodi:**
 - **+ CreateQuestionnaireController(\$scope: \$scope, \$rootscope: \$rootscope, \$mdDialog: \$mdDialog, QuizService: QuizService):**



Metodo costruttore della classe.

Parametri:

* - \$scope: \$scope

Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il viewmodel e il *model_G* dell'applicazione;

* - \$rootScope: \$rootScope

Campo dati contenente il riferimento all'oggetto globale \$rootScope creato da *Angular_G*. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto *QuestionnaireModel*. In questo caso viene utilizzato per inserire in \$rootScope l'oggetto di ritorno della chiamata a *getQuestionnaire* e la lista dei questionari ottenuta dalla chiamata *getQuestionnairePreview*;

* - \$mdDialog: \$mdDialog

Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

* - QuizService: QuizService: parametro che permette di ottenere, tramite il service, la lista di tutte le domande presenti nel quiz.

- + createQuestionnaire(title: String, quiz: QuestionnaireModel): void:
Metodo che permette di inserire un questionario nel database tramite richiesta al service.

Parametri:

* title: String: parametro che rappresenta il titolo del questionario;

* quiz: QuestionnaireModel: parametro che rappresenta l'oggetto questionario.

- + modifyQuestionnaire(quizId: QuestionnaireModel): void

Metodo che serve per modificare un questionario.

Parametri:

* quiz: QuestionnaireModel: parametro che rappresenta l'oggetto questionario.

- + getQuestionnaire(quizId: String): QuestionnaireModel

Metodo che serve per ottenere un questionario tramite l'id in modo da poterlo modificare.

Parametri:

* quizId: String: parametro che rappresenta l'id del questionario da richiedere.

- + getQuestionnairePreview(username: String): QuestionnaireModel<>

Metodo che serve per ottenere la lista di tutti i questionari di un utente.

Parametri:

* username: String: parametro che indica l'utente del quale vogliamo caricare tutti i questionari.

- + deleteQuestionnaire(quizId: String): void

Metodo che elimina un questionario.

Parametri:

* quizId: String: identificativo del questionario da eliminare.

- + getQuestions(): String<>

Metodo che permette di ottenere la lista di tutte le domande;

- + getQuestion(questionId: String): QuestionItemModel

Metodo che ritorna l'intera domanda selezionata.

Parametri:

* questionId: String: parametro che indica l'identificativo univoco di una domanda.



- + addQuestion(questionId: String) : void
Metodo che permette di inserire una domanda nel questionario.

Parametri:

- * questionId: String: parametro che indica l'identificativo univoco di una domanda.

2.3.2.4 QuizziPedia::Front-End::Controllers::EditorQMLController

EditorQMLController
<pre> - \$scope : \$scope - QuestionItemModel : QuestionItemModel - \$mdDialog : mdDialog - QuestionService : QuestionService - \$routeParams : \$routeParams - ParserQML : ParserQML + EditorQMLController(\$scope : \$scope, QuestionItemModel : QuestionItemModel, \$mdDialog : \$mdDialog, QuestionService : QuestionService, \$routeParams : \$routeParams, ParserQML : ParserQML) + submitQuestion() : void </pre>

Figura 17: QuizziPedia::Front-End::Controllers::EditorQMLController

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di domande create tramite editor QML;
- **Utilizzo:** fornisce le funzionalità per creare e modificare una domanda tramite editor QML;
- **Relazione con altre classi:**
 - **IN EditorQMLModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `EditorQMLView` e il $controller_G$ `EditorQMLController`;
 - **IN QuestionService:** questa classe permette di:
 - * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
 - **IN ParserQML:** questa classe rappresenta il parser QML. Essa fa diventare un oggetto di tipo `QuestionItemModel` in linguaggio QML e viceversa;
 - **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda.
- **Attributi:**
 - **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il model dell'applicazione;
 - **QuestionItemModel: QuestionItemModel**
Campo dati che si riferisce alla classe che rappresenta il modello della classe;
 - **\$mdDialog: mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - **QuestionService: QuestionService:**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;



- - **\$routeParams: \$routeParams**
Campo dati contenente il riferimento all'oggetto globale \$routeParams creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url;
- - **ParserQML: ParserQML**
Campo dati che si riferisce alla classe che rappresenta il parser QML. Essa fa diventare un oggetto di tipo **QuestionItemModel** in linguaggio QML e viceversa.

- **Metodi:**

- + **EditorQMLController(\$scope: \$scope, QuestionItemModel: QuestionItemModel, \$mdDialog: \$mdDialog, QuestionService: QuestionService, \$routeParams: \$routeParams, ParserQML: ParserQML)**
Metodo costruttore della classe. Se in \$routeParams sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il **QuestionService** il contenuto della domanda così da poterlo modificare in formato QML. In caso contrario verrà mostrato un errore attraverso \$mdDialog indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in \$routeParams verrà caricata la *view_G* vuota così da poter creare una nuova domanda.

Parametri:

- * **\$scope: \$scope**
Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
 - * **QuestionItemModel: QuestionItemModel**
Parametro che si riferisce alla classe che rappresenta il modello della classe;
 - * **\$mdDialog: \$mdDialog**
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - * **QuestionService: QuestionService:**
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
 - * **\$routeParams: \$routeParams**
Parametro contenente il riferimento all'oggetto globale \$routeParams creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url;
 - * **ParserQML: ParserQML**
Parametro contenente un riferimento alla classe che rappresenta il parser QML. Essa fa diventare un oggetto di tipo **QuestionItemModel** in linguaggio QML e viceversa.
- + **submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal *modelview_G*, li converte attraverso il parser QML, e li manda al server attraverso **QuestionService**. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.3.2.5 QuizziPedia::Front-End::Controllers::FillingQuestionnaireController



FillingQuestionnaireController
- \$scope : \$scope
- \$rootScope : \$rootScope
- \$mdDialog : \$mdDialog
- QuizService : QuizService
+ fillQuiz : FillingQuestionnaireModelView
+ FillingQuestionnaireController(\$scope : \$scope, \$rootScope : \$rootScope, \$mdDialog : \$mdDialog, QuizService : QuizService) : void
- getQuestionnaire(quizId : String) : QuestionnaireModel
+ getNextQuestion(questionId : String) : QuestionItemModel

Figura 18: QuizziPedia::Front-End::Controllers::FillingQuestionnaireController

- **Descrizione:** questa classe permette di gestire la compilazione del questionario;
- **Utilizzo:** fornisce le funzionalità per compilare un questionario e per gestire il cambio di domanda;
- **Relazione con altre classi:**
 - **IN FillingQuestionnaireModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ *FillingQuestionnaireView* e il $controller_G$ *FillingQuestionnaireController*;
 - **IN InfoQuestionnaireTemplate:** rappresenta il componente grafico che permette all'utente di visualizzare le informazioni principali del questionario che si sta per svolgere. Viene gestito dinamicamente all'interno della $view_G$ *TrainingView* attraverso il $controller_G$ *TrainingController*;
 - **IN QuizService:** permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca. Permette inoltre di iscriversi ad un questionario e di scaricare l'intera lista di domande di un questionario a partire dal suo id univoco;
 - **IN QuestionnaireModel:** rappresenta un questionario. Contiene tutte le informazioni necessarie alla presentazione del contenuto del questionario;
 - **IN QuestionsController:** questa classe permette di gestire il recupero delle domande per far sì che possano essere visualizzate nella modalità allenamento e nella compilazione dei questionari;
 - **IN QuestionItemModel:** rappresenta una domanda. Contiene tutte le informazioni necessarie alla presentazione del contenuto della domanda.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - **- \$rootScope: \$rootScope**
Campo dati contenente il riferimento all'oggetto globale $\$rootScope$ creato da $Angular_G$. Viene utilizzato per rendere accessibile a tutti i $controller_G$ e a tutte le $view_G$ l'oggetto *QuestionnaireModel*. In questo caso viene utilizzato per inserire in $\$rootScope$ l'oggetto di ritorno della chiamata a *getNextQuestion* e l'intero questionario ritornato dalla chiamata a *getQuestionnaire*;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;



- - QuizService: QuizService: questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca. Permette inoltre di iscriversi ad un questionario e di scaricare l'intera lista di domande di un questionario a partire dal suo id univoco;
- + fillQuiz: FillingQuestionnaireModelView
Oggetto di tipo FillingQuestionnaireModelView. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding*_G tra la view_G FillingQuestionnaireView e il controller_G FillingQuestionnaireController;

- Metodi:

- + FillingQuestionnaireController(\$scope: \$scope, \$rootScope: \$rootScope, \$mdDialog: \$mdDialog, QuizService: QuizService)
Metodo costruttore della classe.

Parametri:

* - \$scope: \$scope

Campo dati contenente un riferimento all'oggetto \$scope creato da Angular_G. Viene utilizzato come mezzo di comunicazione tra il controller_G e la view_G. Contiene gli oggetti che definiscono il.viewmodel e il model_G dell'applicazione;

* - \$location: \$location

Campo dati contenente un riferimento al servizio creato da Angular_G che permette di accedere alla barra degli indirizzi del browser_G, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;

* - \$mdDialog: \$mdDialog

Campo dati contenente un riferimento al servizio della libreria Material for Angular_G che permette di creare delle componenti a pop-up;

* - QuizService: QuizService: parametro che permette di ottenere, tramite il service, la lista di tutte le domande presenti nel quiz.

- - getQuestionnaire(quizId: String): QuestionnaireModel:

Metodo che permette di ottenere l'intero questionario.

Parametri:

* quizId: String: parametro che indica l'id del quiz che vogliamo ricevere dal back-end.

- + getNextQuestion(questionId: String): QuestionItemModel:

Metodo che ritorna la domanda successiva del quiz tramite chiamata a QuestionController.

Parametri:

* questionId: String: parametro che indica l'id della domanda che vogliamo ricevere dal back-end.

- + startQuiz(): void

Metodo che gestisce l'evento per iniziare il questionario.

2.3.2.6 QuizziPedia::Front-End::Controllers::FillingQuestionsController



FillingQuestionsController
<pre> - \$scope : \$scope - QuestionItemModel : QuestionItemModel - \$mdDialog : mdDialog - \$timeout : \$timeout - \$routeParams : \$routeParams - QuestionsService : QuestionsService + FillingQuestionsController(\$scope : \$scope, QuestionItemModel : QuestionItemModel, \$mdDialog : mdDialog, QuestionsService : QuestionService, \$timeout : \$timeout, \$routeParams : \$routeParams) + submitQuestion() : void + chooseThatWord(word : String, number : Number) : void </pre>

Figura 19: QuizziPedia::Front-End::Controllers::FillingQuestionsController

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda a riempimento di spazi;
- **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda a riempimento di spazi nel database e per modificarne una esistente;
- **Relazione con altre classi:**
 - **IN FillingQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G FillingQuestionsView* e il *controller_G FillingQuestionsController*;
 - **IN QuestionService:** questa classe permette di:
 - * Ottenere una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
 - **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
 - **- QuestionItemModel: QuestionItemModel**
Campo dati che si riferisce alla classe che rappresenta il modello della classe;
 - **- \$timeout: \$timeout**
Attributo contenente il riferimento all'oggetto globale `$timeout` creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una *promise_G*, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita; Campo dati contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - **- QuestionService: QuestionService:**
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
 - **- \$routeParams: \$routeParams**
Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url.
- **Metodi:**



- + **FillingQuestionsController**(`$scope: $scope, QuestionItemModel: QuestionItemModel, $mdDialog: $mdDialog, $timeout: $timeout, QuestionService: QuestionService, $routeParams: $routeParams`)
Metodo costruttore della classe. Se in `$routeParams` sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il `QuestionService` il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso `$mdDialog` indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in `$routeParams` verrà caricata la `view_G` vuota così da poter creare una nuova domanda;
Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url.
- Parametri:**
 - * `$scope: $scope`
Parametro contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il `controller_G` e la `view_G`. Contiene gli oggetti che definiscono il `model_G` dell'applicazione;
 - * `QuestionItemModel: QuestionItemModel`
Parametro che si riferisce alla classe che rappresenta il modello della classe;
 - * `$mdDialog: $mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - * `QuestionService: QuestionService`:
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
 - * `$timeout: $timeout`
Parametro contenente il riferimento all'oggetto globale `$timeout` creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una promise, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;
 - * `$routeParams: $routeParams`
Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url.
- + **submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando;
- + **chooseThatWord(word:String, number: Integer): void**
Metodo che gestisce l'evento click su una parola del testo. Una volta selezionata essa verrà inserita nell'array che conterrà le parole che dovranno essere nascoste quando l'esercizio sarà proposto.

Parametri:

- * `word:String`
Parametro contenente la parola scelta da nascondere;
- * `number: Integer`
Parametro che si riferisce al numero della parola scelta da nascondere.

2.3.2.7 QuizziPedia::Front-End::Controllers::HomeController



HomeController
- \$scope : \$scope - \$location : \$location
+ HomeController(\$scope : \$scope, \$location : \$location) : void + trainingMode() : void

Figura 20: QuizziPedia::Front-End::Controllers::HomeController

- **Descrizione:** questa classe permette di gestire la home page;
- **Utilizzo:** fornisce tutte le informazioni da mostrare nella homepage;
- **Relazione con altre classi:**
 - IN HomeModelView: classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular.js_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* tra la $view_G$ HomeView e il controller $_G$ HomeController;
- **Attributi:**
 - - $\$scope$: $\$scope$
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il controller $_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - - $\$location$: $\$location$
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa.
- **Metodi:**
 - + HomeController($\$scope$: $\$scope$, $\$location$: $\$location$)
Metodo costruttore della classe:
Parametri:
 - * $\$scope$: $\$scope$
Parametro contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il controller $_G$ e la $view_G$. Contiene gli oggetti che definiscono il viewmodel e il $model_G$ dell'applicazione;
 - * $\$location$: $\$location$
Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa.
 - + trainingMode(): void
Metodo che gestisce l'evento click sul pulsante di allenamento. Effettua il redirect alla pagina di allenamento.

2.3.2.8 QuizziPedia::Front-End::Controllers::ImagesSortingQuestionsController



ImagesSortingQuestionsController
<pre>\$scope : \$scope QuestionItemModel : QuestionItemModel \$mdDialog : mdDialog QuestionService : QuestionService Upload : Upload \$timeout : \$timeout \$routeParams : \$routeParams + ImagesSortingQuestionsController(\$scope : \$scope, QuestionItemModel : QuestionItemModel, \$mdDialog : SmdDialog, QuestionService : QuestionService, Upload : Upload, \$timeout : \$timeout, \$routeParams : \$routeParams) + submitQuestion() : void</pre>

Figura 21: QuizziPedia::Front-End::Controllers::ImagesSortingQuestionsController

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento immagini;
- **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda a ordinamento immagini nel database e per modificarne una esistente;
- **Relazione con altre classi:**
 - **IN ImageSortingQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* tra la view **ImagesSortingQuestionsView** e il controller **ImagesSortingQuestionsController**;
 - **IN QuestionService:** questa classe permette di:
 - * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
 - **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - **- QuestionItemModel: QuestionItemModel**
Campo dati che si riferisce alla classe che rappresenta il modello della classe;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular* $_G$ che permette di creare delle componenti a pop-up;
 - **- QuestionService: QuestionService:**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
 - **- Upload: Upload**
Campo dati contenente un riferimento alla libreria *ng-file-upload* $_G$ necessaria per il caricamento della foto profilo dell'utente;
 - **- \$timeout: \$timeout**
Campo dati contenente il riferimento all'oggetto globale $\$timeout$ creato da $Angular_G$. Il valore di ritorno di una chiamata alla funzione di $\$timeout$ è una $promise_G$, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;
 - **- \$routeParams: \$routeParams**
Campo dati contenente il riferimento all'oggetto globale $\$routeParams$ creato da $Angular_G$. Tale servizio permette di recuperare il set di variabili presenti nell'url;



- **Metodi:**

- + `ImageSortingQuestionsController($scope: $scope, QuestionItemModel: QuestionItemModel, $mdDialog: $mdDialog, QuestionService: QuestionService, Upload: Upload, $timeout: $timeout, $routeParams: $routeParams)`

Metodo costruttore della classe. Se in `$routeParams` sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il `QuestionService` il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso `$mdDialog` indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in `$routeParams` verrà caricata la view vuota così da poter creare una nuova domanda;

Parametri:

- * `$scope: $scope`

Parametro contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il `controllerG` e la `viewG`. Contiene gli oggetti che definiscono il model dell'applicazione;

- * `QuestionItemModel: QuestionItemModel`

Parametro che si riferisce alla classe che rappresenta il modello della classe;

- * `$mdDialog: $mdDialog`

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- * `QuestionService: QuestionService`:

Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;

- * `Upload: Upload`

Parametro contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;

- * `$timeout: $timeout`

Parametro contenente il riferimento all'oggetto globale `$timeout` creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una *promise_G*, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;

- * `$routeParams: $routeParams`

Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'url.

- + `submitQuestion(): void`

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.3.2.9 QuizziPedia::Front-End::Controllers::InputToListController

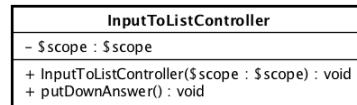


Figura 22: QuizziPedia::Front-End::Controllers::InputToListController

- **Descrizione:** questa classe permette di gestire l'inserimento di una lista di risposte durante la creazione di una domanda;
- **Utilizzo:** fornisce le funzionalità per confermare porzioni di domanda durante la creazione;
- **Relazione con altre classi:**
 - **OUT MultipleQuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular.js_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G MultipleQuestionsView* e il *controller_G MultipleQuestionsController*;
 - **OUT ConnectionQuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G ConnectionQuestionView* e il *controller_G ConnectionQuestionController*;
 - **OUT StringsSortingQuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G StringsSortingQuestionView* e il *controller_G StringsSortingQuestionController*;
 - **OUT ImagesSortingQuestionsModelView:** permette all'utente di creare una domanda a ordinamento immagini compilando i campi proposti classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular.js_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G ImagesSortingQuestionView* e il *controller_G ImagesSortingQuestionController*;
- **Attributi:**
 - - *\$scope*: *\$scope*
Campo dati contenente un riferimento all'oggetto *\$scope* creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione.
- **Metodi:**
 - + *InputToListController(\$scope: \$scope)*
Metodo costruttore della classe.
Parametri:
 - * - *\$scope*: *\$scope*
Parametro contenente un riferimento all'oggetto *\$scope* creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione.
 - - *putDownAnswer(): void*
Metodo che reagisce all'evento di aggiunta nuova risposta e la salva in modo che venga caricata e visualizzata nella pagina.



2.3.2.10 QuizziPedia::Front-End::Controllers::LoginController

LoginController
<ul style="list-style-type: none"> - \$scope : \$scope - \$location : \$location - \$mdDialog : \$mdDialog - AuthService : AuthService + user : LoginModelView
<ul style="list-style-type: none"> + LoginController(\$scope : \$scope, \$location : \$location, \$mdDialog : \$mdDialog, AuthService : AuthService) : void + login(email : String, password : String) : void + signUp() : void + recoveryPassword() : void

Figura 23: QuizziPedia::Front-End::Controllers::LoginController

- **Descrizione:** questa classe permette di gestire l'autenticazione dell'utente al sistema;
- **Utilizzo:** fornisce le funzionalità di autenticazione al sistema, compresa la gestione di situazioni di errore di autenticazione;
- **Relazione con altre classi:**
 - **IN LoginModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G LoginView* e il *controller_G LoginController*;
 - **IN AuthService:** questa classe permette di gestire la registrazione e l'autenticazione di un utente;
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto `$scope` creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel* e il *model_G* dell'applicazione;
 - **- \$location: \$location**
Campo dati contenente un riferimento al servizio creato da *Angular_G* che permette di accedere alla barra degli indirizzi del *browser_G*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - **- AuthService: AuthService**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo `logIn()` di `$textttAuthService` a cui vengono passati i parametri `username` e `password`;
 - **+ user: LoginModelView**
Oggetto di tipo *LoginModelView*. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G LoginView* e il *controller_G LoginController*;
- **Metodi:**



- + LoginController(\$scope: \$scope, \$rootScope: \$rootScope, \$location: \$location, \$mdDialog: \$mdDialog, AuthService: AuthService)

Metodo costruttore della classe.

Parametri:

 - * **\$scope: \$scope**
Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il viewmodel e il *model_G* dell'applicazione;
 - * **\$location: \$location**
Parametro contenente un riferimento al servizio creato da *Angular_G* che permette di accedere alla barra degli indirizzi del *browser_G*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
 - * **\$mdDialog: \$mdDialog**
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - * **AuthService: AuthService**
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo `logIn()` di `$textttAuthService` a cui vengono passati i parametri `email` e `password`;
- + logIn(email: String, password: String): void
Metodo che richiama il metodo `signin` del service `AuthService` passandogli `email` e `password`. Nel caso di buona riuscita dell'operazione viene effettuato il redirect alla homepage dell'applicazione. Nel caso in cui invece avvenga un errore, viene mostrato a video il messaggio di errore;
- + signUp(): void
Metodo che gestisce l'evento click sul pulsante di registrazione. Effettua il redirect alla pagina di registrazione;
- + recoveryPassword(): void
Metodo che gestisce l'evento click sul pulsante di recupero password. Effettua il redirect alla pagina per il recupero della password;

2.3.2.11 QuizziPedia::Front-End::Controllers::MenuBarController

MenuBarController
<ul style="list-style-type: none"> - \$scope : \$scope - \$rootScope : \$rootScope - \$location : \$location - \$mdDialog : \$mdDialog - AuthService : AuthService - menuBarModel : MenuBarModel <ul style="list-style-type: none"> + MenuBarController(\$scope : \$scope, \$location : \$location, \$mdDialog : \$mdDialog, AuthService : AuthService, MenuBarModel : MenuBarController) + logOut(): void + logIn(): void + signUp(): void + goToUserPage(): void + goToUserManagementPage(): void + goToQuestionsManagementPage(): void + goToQuizManagementPage(): void + \$on('\$routeChangeStart': String, callback: fuction): void

Figura 24: QuizziPedia::Front-End::Controllers::MenuBarController

- **Descrizione:** questa classe permette di gestire il menù fisso per ogni pagina;
- **Utilizzo:** fornisce le funzionalità per aggiornare, a seconda della pagina, il contenuto del menù;



- **Relazione con altre classi:**

- **IN MenuBarModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $directive_G$ **MenuBarDirective** e il $controller_G$ **MenuBarController**. Rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello $\$scope$. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata;
- **IN AuthService:** questa classe permette di gestire la registrazione e l'autenticazione di un utente;
- **IN MenuBarModel:** questa classe rappresenta la classe che contiene le informazioni per la giusta visualizzazione della barra.

- **Attributi:**

- **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
- **- \$rootScope: \$rootScope**
Campo dati contenente il riferimento all'oggetto globale $\$rootScope$ creato da $Angular_G$;
- **- \$location: \$location**
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- **- AuthService: AuthService**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione;
- **- menuBarModel: MenuBarModel:**
Campo dati contenente un riferimento all'oggetto che contiene le informazioni per la giusta visualizzazione della barra.

- **Metodi:**

- **+ MenuBarController(\$scope: \$scope, \$location: \$location, \$mdDialog: \$mdDialog, AuthService: AuthService, MenuBarModel: MenuBarModel)**

Metodo costruttore della classe.

Parametri:

- * **\$scope: \$scope**

Parametro contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel$ e il $model_G$ dell'applicazione;

- * **\$rootScope: \$rootScope**

Campo dati contenente il riferimento all'oggetto globale $\$rootScope$ creato da $Angular_G$;



- * `$location: $location`
Parametro contenente un riferimento al servizio creato da *AngularG* che permette di accedere alla barra degli indirizzi del *browserG*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
 - * `$mdDialog: $mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
 - * `AuthService: AuthService`
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo `logOut` di `$textttAuthService` a cui viene passato il parametro `username`;
 - * `MenuBarModel: MenuBarModel`
Parametro contenente un riferimento all'oggetto che contiene le informazioni per la giusta visualizzazione della barra.
 - + `logOut(): void`
Metodo che richiama il metodo `logOut` del service `AuthService` passandogli lo `username`. Prima di effettuare questa operazione viene mostrato a video un messaggio di conferma per il proseguo dell'operazione;
 - + `logIn(): void`
Metodo che gestisce l'evento click sul pulsante per effettuare il login. Effettua il redirect alla pagina per effettuare il login;
 - + `signUp(): void`
Metodo che gestisce l'evento click sul pulsante per effettuare la registrazione. Effettua il redirect alla pagina per effettuare la registrazione;
 - + `goToUserPage(): void`
Metodo che gestisce l'evento click sul pulsante di visualizzazione della pagina utente. Effettua il redirect alla pagina di visualizzazione della pagina utente;
 - + `goToUserManagementPage(): void`
Metodo che gestisce l'evento click sul pulsante di gestione del profilo utente. Effettua il redirect alla pagina di gestione del profilo utente;
 - + `goToQuestionsManagementPage(): void`
Metodo che gestisce l'evento click sul pulsante di gestione delle domande. Effettua il redirect alla pagina di gestione delle domande;
 - + `goToQuizManagementPage(): void`
Metodo che gestisce l'evento click sul pulsante di gestione dei questionari. Effettua il redirect alla pagina di gestione dei questionari;
 - + `$on('$routeChangeStart', function(next, current)): void`
Metodo che cattura i cambiamenti dell'url e che richiede al `MenuBarModel` le giuste direttive da inserire in `MenuBarDirective`.
- Parametri;**
- * `'$routeChangeStart': String`
Servizio offerto da *Angular.jsG* per catturare gli eventi sulla barra degli URL;
 - * `callback: function`
Funzione di callback per gestire i cambiamenti della barra degli indirizzi.

2.3.2.12 QuizziPedia::Front-End::Controllers::MultipleQuestionsController

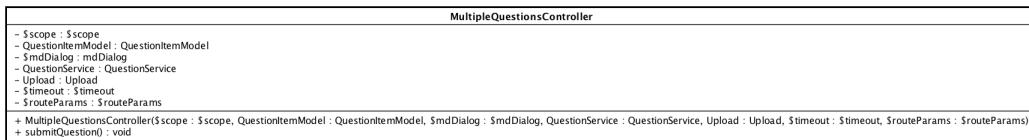


Figura 25: QuizziPedia::Front-End::Controllers::MultipleChoiceQuestion

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda a risposta multipla;
- **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda a risposta multipla nel database e per modificarne una esistente;
- **Relazione con altre classi:**
 - **IN MultipleQuestionsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `MultipleQuestionsView` e il $controller_G$ `MultipleQuestionsController`;
 - **IN QuestionService:** questa classe permette di:
 - * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
 - **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda;
- **Attributi:**
 - - **\$scope:** $\$scope$
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il model dell'applicazione;
 - - **QuestionItemModel:** `QuestionItemModel`
Campo dati che si riferisce alla classe che rappresenta il modello della classe;
 - - **\$mdDialog:** $\$mdDialog$
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a popup;
 - - **QuestionService:** `QuestionService`:
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
 - - **Upload:** `Upload`
Campo dati contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;
 - - **\$timeout:** $\$timeout$
Campo dati contenente il riferimento all'oggetto globale $\$timeout$ creato da $Angular.js_G$. Il valore di ritorno di una chiamata alla funzione di $\$timeout$ è una $promise_G$, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;
 - - **\$routeParams:** $\$routeParams$
Campo dati contenente il riferimento all'oggetto globale $\$routeParams$ creato da $Angular.js_G$. Tale servizio permette di recuperare il set di variabili presenti nell'url;



- **Metodi:**

- + **MultipleQuestionsController(\$scope: \$scope, QuestionItemModel: QuestionItemModel, \$mdDialog: \$mdDialog, QuestionService: QuestionService, Upload: Upload, \$timeout: \$timeout, \$routeParams: \$routeParams)**

Metodo costruttore della classe. Se in `$routeParams` sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il `QuestionService` il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso `$mdDialog` indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in `$routeParams` verrà caricata la `view_G` vuota così da poter creare una nuova domanda;

Parametri:

- * `$scope: $scope`

Parametro contenente un riferimento all'oggetto `$scope` creato da `Angular_G`, viene utilizzato come mezzo di comunicazione tra il `controller_G` e la `view_G`. Contiene gli oggetti che definiscono il model dell'applicazione;

- * `QuestionItemModel: QuestionItemModel`

Parametro che si riferisce alla classe che rappresenta il modello della classe;

- * `$mdDialog: $mdDialog`

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- * `QuestionService: QuestionService`:

Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;

- * `Upload: Upload`

Parametro contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;

- * `$timeout: $timeout`

Parametro contenente il riferimento all'oggetto globale `$timeout` creato da `Angular.js_G`. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una *promise_G*, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout eseguita;

- * `$routeParams: $routeParams`

Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da `Angular_G`. Tale servizio permette di recuperare il set di variabili presenti nell'url;

- + `submitQuestion(): void`

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal `modelview_G` e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando;

2.3.2.13 QuizziPedia::Front-End::Controllers::NewQuestionsButtonController



NewQuestionsButtonController
- \$scope : \$scope
- \$location : \$location
+ NewQuestionButtonsController(\$scope : \$scope, \$location : \$location)
+ newQuestion() : void

Figura 26: QuizziPedia::Front-End::Controllers::NewQuestionsButtonController

- **Descrizione:** questa classe permette di effettuare il redirect alla pagina di creazione nuova domanda;
- **Utilizzo:** effettua il redirect alla pagina di creazione di una nuova domanda quando l'utente seleziona interagisce con il bottone a cui è collegato il corrispettivo evento;
- **Relazione con altre classi:**
 - **OUT NewQuestionsButtonDirective:** rappresenta il componente grafico che permette all'utente di posizionarsi nella $view_G$ di creazione di una nuova domanda.
- **Attributi:**
 - - **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto \$scope creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - - **\$location: \$location**
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa.
- **Metodi:**
 - + **NewQuestionButtonsController(\$scope: \$scope, \$location: \$location)**
Metodo costruttore della classe.
 - Parametri:**
 - * **\$scope: \$scope**
Parametro contenente un riferimento all'oggetto \$scope creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;
 - * **\$location: \$location**
Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa.
 - + **newQuestion(): void**
Metodo che gestisce l'evento click sul pulsante per creare una nuova domanda. Effettua il redirect alla pagina di creazione di una domanda.

2.3.2.14 QuizziPedia::Front-End::Controllers::PasswordForgotController



PasswordForgotController
- \$scope : \$scope
- \$location : \$location
- \$mdDialog : \$mdDialog
- AuthService : AuthService
+ PasswordForgotController(\$scope : \$scope, \$location : \$location, \$mdDialog : \$mdDialog, AuthService : AuthService)
+ passwordForgot() : void
+ login() : void

Figura 27: QuizziPedia::Front-End::Controllers::PasswordForgotController

- **Descrizione:** questa classe permette di gestire il ripristino della password dimenticata;
- **Utilizzo:** fornisce tutte le funzionalità per ripristinare la password dopo aver verificato l'identità dell'utente;
- **Relazione con altre classi:**
 - **IN PasswordForgotModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G PasswordForgotView* e il *controller_G PasswordForgotController*;
 - **OUT AuthService:** questa classe permette di gestire la registrazione e l'autenticazione di un utente.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto *\$scope* creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
 - **- \$location: \$location**
Campo dati contenente un riferimento al servizio creato da *Angular_G* che permette di accedere alla barra degli indirizzi del *browser_G*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
 - **- \$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - **- AuthService: AuthService**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo *passwordForgot* di *AuthService* a cui viene passato il parametro *email*.
- **Metodi:**
 - **+ PasswordForgotController(\$scope: \$scope, \$location: \$location, \$mdDialog: \$mdDialog, AuthService: AuthService)**
Metodo costruttore della classe.

Parametri:

* **\$scope: \$scope**

Parametro contenente un riferimento all'oggetto *\$scope* creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;



- * `$location: $location`
Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- * `$mdDialog: $mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- * `AuthService: AuthService`
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo `passwordForgot` di `AuthService` a cui viene passato il parametro `email`.
- + `passwordForgot(): void`
Metodo che richiama il metodo `getNewPassword` del $service_G$ `AuthService` passandogli il parametro `email`. Nel caso di buona riuscita dell'operazione, viene mostrato un messaggio di successo il cui corpo contiene anche un bottone per effettuare il redirect alla pagina di login. Nel caso in cui invece avvenga un errore, viene mostrato a video il messaggio di errore;
- + `logIn(): void`
Metodo che gestisce l'evento click sul pulsante di login. Effettua il redirect alla pagina di login.

2.3.2.15 QuizziPedia::Front-End::Controllers::ProfileManagementController

ProfileManagementController
- <code>\$scope : \$scope</code> - <code>\$rootScope : \$rootScope</code> - <code>\$mdDialog : \$mdDialog</code> - <code>UserDetailsService : UserDetailsService</code> - <code>user : UserDetailsModel</code> - <code>Upload : Upload</code> - <code>\$timeout : \$timeout</code> + <code>ProfileManagementController(\$scope : \$scope, \$rootScope : \$rootScope, \$mdDialog : \$mdDialog, UserDetailsService : UserDetailsService)</code> + <code>confirm(user : Object, imageObject : Object) : void</code>

Figura 28: QuizziPedia::Front-End::Controllers::ProfileManagementController

- **Descrizione:** questa classe permette di gestire il profilo personale di un utente;
- **Utilizzo:** fornisce le funzionalità all'utente per poter gestire i propri dati;
- **Relazione con altre classi:**
 - **IN ProfileManagementModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `ProfileManagementView` e il $controller_G$ `ProfileManagementController`;
 - **IN UserDetailsService:** questa classe permette di ottenere i dati personali degli utenti;
 - **IN UserDetailsModel:** questa classe rappresenta il tipo dell'utente autenticato della pagina.
- **Attributi:**



- - **\$scope:** `$scope`
Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
- - **\$rootScope:** `$rootScope`
Campo dati contenente il riferimento all'oggetto globale \$rootScope creato da *Angular_G*. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto *UserDetailsModel*;
- - **\$mdDialog:** `$mdDialog`
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- - **UserDetailsService:** `UserDetailsService`
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate agli utenti;
- - **user:** `UserDetailsModel`:
Oggetto di tipo *UserDetailsModel*. Viene mantenuto all'interno del \$rootScope;
- - **Upload:** `Upload`
Campo dati contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;
- - **\$timeout:** `$timeout`
Campo dati contenente il riferimento all'oggetto globale \$timeout creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di \$timeout è una *promise_G*, la quale sarà risolta quando avverrà il ritardo e la funzione di \$timeout sarà eseguita;

- **Metodi:**

- + **ProfileManagementController(\$scope: \$scope, \$rootScope: \$rootScope, \$mdDialog: \$mdDialog, UserDetailsService: UserDetailsService)**
Metodo costruttore della classe.

Parametri:

- * **\$scope:** `$scope`
Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;
- * **\$rootScope:** `$rootScope`
Parametro contenente il riferimento all'oggetto globale \$rootScope creato da *Angular_G*. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto *UserDetailsModel*. In questo caso viene utilizzato per aggiornare in \$rootScope l'oggetto che rappresenta l'utente autenticato all'interno dell'applicazione;
- * **\$location:** `$location`
Parametro contenente un riferimento al servizio creato da *Angular_G* che permette di accedere alla barra degli indirizzi del *browser_G*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- * **\$mdDialog:** `$mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- * **UserDetailsService:** `UserDetailsService`
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'utente;



- * **Upload:** `Upload`
Parametro contenente un riferimento alla libreria $ng\text{-}file\text{-}upload_G$ necessaria per il caricamento della foto profilo dell'utente;
- * **\$timeout:** `$timeout`
Parametro contenente il riferimento all'oggetto globale `$timeout` creato da $Angular_G$. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una $promise_G$, la quale sarà risolta quando avverrà il ritardo e la funzione di timeout sarà eseguita.
- **+ confirm(user: Object, imageObject: Object): void**
Metodo che gestisce l'evento click sul pulsante di conferma modifica. Aggiorna, in caso di modifiche, l'oggetto locale `UserDetailsModel`. Inoltre, utilizzando il metodo dell'`UserDetailsService`, aggiorna anche nel $server_G$ i dati dell'utente.

2.3.2.16 QuizziPedia::Front-End::Controllers::QuestionnaireDetailsController

QuestionnaireDetailsController
<ul style="list-style-type: none"> - <code>\$scope : \$scope</code> - <code>\$rootScope : \$rootScope</code> - <code>\$mdDialog : \$mdDialog</code> - <code>QuizService : QuizService</code> + <code>details : QuestionnaireDetailsModelView</code> + <code>QuestionnaireDetailsController(\$scope : \$scope, \$rootScope : \$rootScope, \$mdDialog : \$mdDialog, QuizService : QuizService)</code> - <code>getQuestionnaireDetails(username : String) : Object</code> - <code>getSubscribedQuestionnaire(username : String) : Object</code>

Figura 29: QuizziPedia::Front-End::Controllers::QuestionnaireDetailsController

- **Descrizione:** questa classe permette di gestire i dettagli di un questionario;
- **Utilizzo:** fornisce le funzionalità per recuperare dal back-end i dettagli di un questionario creato da un utente al fine di poterli visualizzare nel suo profilo;
- **Relazione con altre classi:**
 - **IN QuestionnaireDetailsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `UserView` e il $controller_G$ `QuestionnaireDetailsController`;
 - **IN QuizService:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca;
 - **IN QuestionnaireModel:** rappresenta un questionario. Contiene tutte le informazioni necessarie alla presentazione del contenuto del questionario.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto `$scope` creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - **- \$rootScope: \$rootScope**
Campo dati contenente il riferimento all'oggetto globale `$rootScope` creato da $Angular_G$. Viene utilizzato per rendere accessibile a tutti i $controller_G$ e a tutte le $view_G$ l'oggetto `QuestionnaireModel`. In questo caso viene utilizzato per inserire in `$rootScope` l'oggetto di ritorno della chiamata a `getQuestionnaireDetails` del $service_G$ `QuizService`;



- - QuizService: QuizService
Questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca;
- - \$mdDialog: \$mdDialog
Parametro contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
- + details: QuestionnaireDetailsModelView
Oggetto di tipo *QuestionnaireDetailsModelView*. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la *viewG UserView* e il *controllerG QuestionnaireDetailsController*.

• **Metodi:**

- + QuestionnaireDetailsController(\$scope: \$scope, \$rootScope: \$rootScope, \$mdDialog: \$mdDialog, QuizService: QuizService)
Metodo costruttore della classe.

Parametri:

* \$scope: \$scope

Parametro contenente un riferimento all'oggetto \$scope creato da *AngularG*. Viene utilizzato come mezzo di comunicazione tra il *controllerG* e la *viewG*. Contiene gli oggetti che definiscono il *viewmodelG* e il *modelG* dell'applicazione;

* \$rootScope: \$rootScope

Parametro contenente il riferimento all'oggetto globale \$rootScope creato da *AngularG*. Viene utilizzato per rendere accessibile a tutti i *controllerG* e a tutte le *viewG* l'oggetto *QuestionnaireModel*. In questo caso viene utilizzato per inserire in \$rootScope l'oggetto di ritorno della chiamata a *getQuestionnaireDetails* del *serviceG QuizService*;

* \$mdDialog: \$mdDialog

Parametro contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;

* QuizService: QuizService

Parametro che permette di ottenere, tramite il *serviceG*, la lista di tutte le domande presenti nel quiz.

- - getQuestionnaireDetails(username: String): Object

Metodo che richiede al *serviceG* i dettagli dei questionari eseguiti dall'utente.

Parametri:

* username: String: username dell'utente del quale caricare i questionari.

- - getSubscribedQuestionnaire(username: String): Object

Metodo che ritorna i questionari a cui l'utente è iscritto.

Parametri:

* username: String: username dell'utente del quale scaricare i questionari a cui è iscritto.

2.3.2.17 QuizziPedia::Front-End::Controllers::QuestionnaireManagementController



QuestionnaireManagementController
- \$scope : \$scope
- QuizService : QuizService
- \$mdDialog : \$mdDialog
+ QuestionnaireManagementController(\$scope : \$scope, \$mdDialog : \$mdDialog, QuizService : QuizService)
+ getUserQuestionnaire(username : String) : Array<QuestionnaireModel>

Figura 30: QuizziPedia::Front-End::Controllers::QuestionnaireManagementController

- **Descrizione:** questa classe permette di gestire tutti i questionari creati da un utente;
- **Utilizzo:** fornisce le funzionalità per recuperare dal back-end tutti i questionari creati da un utente;
- **Relazione con altre classi:**
 - **IN QuestionnaireManagementModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra la $view_G$ *QuestionnaireManagementView* e il $controller_G$ *QuestionnaireManagementController*;
 - **IN QuizService:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca;
 - **IN QuestionnaireModel:** rappresenta un questionario. Contiene tutte le informazioni necessarie alla presentazione del contenuto del questionario;
- **Attributi:**
 - - **\$scope:** $\$scope$
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - - **\$mdDialog:** $\$mdDialog$
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - - **QuizService:** $QuizService$: permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca.
- **Metodi:**
 - + **QuestionnaireManagementController(\$scope: \$scope, \$mdDialog: \$mdDialog, QuizService: QuizService)**
Metodo costruttore della classe.
Parametri:
 - * **\$scope:** $\$scope$
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;
 - * **\$mdDialog:** $\$mdDialog$
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;



- * **QuizService:** `QuizService`: parametro che permette di ottenere, tramite il $service_G$, la lista di tutte le domande presenti nel quiz;
- + `getUserQuestionnaire(username: String) Array<QuestionnaireModel>`
Metodo che ritorna tutti i questionari creati da un utente in un array di `QuestionnaireModel`.

Parametri:

- * `username: String`: parametro che indica l'username dell'utente del quale vogliamo scaricare tutti i questionari.

2.3.2.18 QuizziPedia::Front-End::Controllers::QuestionsController

QuestionsController	
- <code>\$scope : \$scope</code>	
+ <code>QuestionsController(\$scope : \$scope, \$rootScope : \$rootScope, QuestionService : QuestionService, \$mdDialog : \$mdDialog, QuestionnaireModelView : QuestionnaireModelView)</code>	
+ <code>addAnswer(index : Number, typeQuestion : String, answerGiven : Array<String>) : void</code>	
+ <code>answerGiven(index : Number) : Array<String></code>	
+ <code>orderChosen(index : Number) : Array<String></code>	
+ <code>linkingMade(index : Number) : Array<String></code>	
+ <code>loadNewQuestionBy(topic : String, keywords : Array<String>, level : Number) : void</code>	
+ <code>loadNewQuestion(question : QuestionItemModel) : void</code>	
- <code>getQuestionBy(topic : String, keywords : Array<String>, level : Number) : QuestionItemModel</code>	
+ <code>checkAnswer() : Boolean</code>	

Figura 31: QuizziPedia::Front-End::Controllers::QuestionsController

- **Descrizione:** questa classe permette di gestire il recupero delle domande per far sì che possano essere visualizzate nella modalità allenamento e nella compilazione dei questionari;
- **Utilizzo:** fornisce le funzionalità per il recupero delle domande esistenti nel database al fine di poterle visualizzare nella modalità allenamento e nella compilazione dei questionari;
- **Relazione con altre classi:**
 - **IN QuestionsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le $directive_G$ che compongono dinamicamente la vista della domanda e il $controller_G$ `QuestionsController`;
 - **IN QuestionServices:** questa classe permette di ottenere domande esistenti e salvare nuove domande;
 - **IN QuestionItemModel:** rappresenta una domanda. Contiene tutte le informazioni necessarie alla presentazione del contenuto della domanda;
 - **OUT FillingQuestionnaireController:** questa classe permette di gestire la compilazione del questionario;
 - **OUT TrainingController:** questa classe permette di gestire la modalità allenamento sottoponendo all'utente le giuste domande adatte al suo livello.

- **Attributi:**

- `$scope: $scope`
Campo dati contenente un riferimento all'oggetto `$scope` creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;



- - **\$rootScope:** `$rootScope`
Campo dati contenente il riferimento all'oggetto globale `$rootScope` creato da $Angular_G$. Viene utilizzato per rendere accessibile a tutti i $controller_G$ e a tutte le $view_G$ l'oggetto `QuestionItemModel`. In questo caso viene utilizzato per inserire in `$rootScope` l'oggetto di ritorno della chiamata a `getQuestion` del $service_G$ `QuestionsService`;
- - **\$mdDialog:** `$mdDialog`
Campo dati contenente un riferimento al servizio della libreria *Material for Angular G* che permette di creare delle componenti a pop-up;
- - **QuestionService:** `QuestionsService`
Permette di ottenere domande esistenti tramite chiamata di metodo specifici;
- - **question:** `QuestionsModelView`
Classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding G* tra le $directive_G$ che compongono dinamicamente la vista della domanda e il $controller_G$ `QuestionsController`.

- **Metodi:**

- + **QuestionsController(\$scope: \$scope, \$rootScope: \$rootScope, \$mdDialog: \$mdDialog, QuestionService: QuestionService)**
Metodo costruttore della classe. Interagendo con l'oggetto `QuestionItemModel` ricevuto si interfaccia con la $view_G$ andando a visualizzare la giusta $directive_G$ della tipologia di domanda.

Parametri:

- * **\$scope:** `$scope`
Campo dati contenente un riferimento all'oggetto `$scope` creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;
- * **\$rootScope:** `$rootScope`
Parametro contenente il riferimento all'oggetto globale `$rootScope` creato da $Angular_G$. Viene utilizzato per rendere accessibile a tutti i $controller_G$ e a tutte le $view_G$ l'oggetto `QuestionItemModel`. In questo caso viene utilizzato per inserire in `$rootScope` l'oggetto di ritorno della chiamata a `getQuestion` del $service_G$ `QuestionsService`;
- * **\$mdDialog:** `$mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for Angular G* che permette di creare delle componenti a pop-up;
- * **QuestionService:** `QuestionService`
Parametro che permette di ottenere domande esistenti tramite chiamata di metodo specifici;

- - **getQuestionBy(topic: String, keywords: Array<String>, level: Number): QuestionItemModel**
Metodo che richiede al back-end una domanda.

Parametri:

- * **topic:** `String`
Parametro contenente l'argomento della domanda;
- * **keywords:** `Array<String>`
Parametro contenente un array di stringhe che rappresenta le keywords scelte per l'allenamento;



* **level:** Number

Parametro contenente il livello dell'utente.

- + **addAnswer(index: Number, typeQuestion: String, answerGiven: Array<String>): void**

Metodo che gestisce l'evento di selezione delle risposte.

Parametri:

* **index:** Number

Parametro contenente l'indice della risposta di cui si vuole tenere traccia. Rappresenta anche l'indice dell'array **answerGiven** in cui verrà inserito l'oggetto delle risposte date;

* **typeQuestion:** String

Parametro contenente una stringa la quale indica la tipologia della domanda;

* **answerGiven:** Array<String>

Parametro contenente l'array di risposte date dall'utente aggiornato all'ultima iterazione.

- + **answerGiven(index: Number): Array<String>**

Metodo di supporto che ritorna un array di stringhe contenente le risposte date. Si occupa di recuperare le risposte date nelle domande vero/falso, risposta multipla e ad area cliccabile.

Parametri:

* **index:** Number

Parametro contenente l'indice della risposta di cui si vuole raccogliere le risposte date.

- + **orderChosen(index: Number): Array<String>**

Metodo di supporto che ritorna un array di stringhe contenente le risposte date. Si occupa di recuperare le risposte date nelle domande ad ordinamento e di riempimento di spazi.

Parametri:

* **index:** Number

Parametro contenente l'indice della risposta di cui si vuole raccogliere le risposte date.

- + **linkingMade(index: Number): Array<String>**

Metodo di supporto che ritorna un array di stringhe contenente le risposte date. Si occupa di recuperare le risposte date nelle domande a collegamento.

Parametri:

* **index:** Number

Parametro contenente l'indice della risposta di cui si vuole raccogliere le risposte date.

- + **loadNewQuestionBy(topic: String, keywords: Array<String>, level: Number): void**

Metodo che gestisce l'evento per scaricare una nuova domanda in base ai parametri passati. Evoca l'evento per inserire la domanda in **TrainingModelView**.

Parametri:

* **topic:** String

Parametro contenente l'argomento della domanda;

* **keywords:** Array<String>

Parametro contenente un array di stringhe che rappresenta le keywords scelte per l'allenamento;



```
* level: Number
    Parametro contenente il livello dell'utente.

- + loadNewQuestion(question: QuestionItemModel): void
    Metodo che gestisce l'evento per visualizzare una nuova domanda.

Parametri:
    * question: QuestionItemModel
        Parametro contenente un riferimento all'oggetto di tipo QuestionItemModel.

- + checkAnswer(): Boolean
    Metodo che controlla che le risposte date siano corrette.
```

2.3.2.19 QuizziPedia::Front-End::Controllers::QuestionsManagementController

QuestionsManagementController
- \$scope : \$scope
- \$location : \$location
- QuestionService : QuestionService
+ QuestionsManagementController(\$scope : \$scope, \$location : \$location, QuestionService : QuestionService)
+ getQuestionsByUser(username : String) : Array<QuestionItemModel>
+ editQuestion(idquestion : String) : void

Figura 32: QuizziPedia::Front-End::Controllers::QuestionsManagementController

- **Descrizione:** questa classe permette di gestire le domande create dall'utente e di crearne di nuove;
- **Utilizzo:** fornisce le funzionalità per richiedere al $server_G$ le domande create dall'utente e mostrarle nella pagina dedicata. Inoltre permette di catturare gli eventi per modificare le domande esistenti e per creare una di nuova;
- **Relazione con altre classi:**
 - **IN QuestionsManagementModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `QuestionsManagementView` e il $controller_G$ `QuestionsManagementController`;
 - **IN QuestionService:** questa classe permette di ottenere domande esistenti e salvare nuove domande;
 - **IN QuestionsService:** questa classe permette di ottenere domande esistenti e salvare nuove domande.
- **Attributi:**
 - - **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - - **\$location: \$location**
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;



- - **QuestionService:** `QuestionsService`

Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande.

• **Metodi:**

- + `QuestionsManagementController($scope: $scope, $location: $location, QuestionService: QuestionService)`

Metodo costruttore della classe.

Parametri:

* `$scope: $scope`

Parametro contenente un riferimento all'oggetto `$scope` creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;

* `$location: $location`

Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;

* `QuestionService: QuestionService`

Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande.

- + `getQuestionsByUser(username: String) : Array<QuestionItemMode>`

Metodo che acquisisce le domande create dall'utente attraverso il `QuestionService`.

Parametri:

* `username: String`

Parametro contenente l'username dell'utente.

- + `editQuestion(idQuestion: String) : void`

Metodo che gestisce l'evento click sul pulsante per modificare la domanda. Effettua il redirect alla pagina di modifica della domanda.

Parametri:

* `idQuestion: String`

Parametro contenente l'id della domanda da modificare.

2.3.2.20 QuizziPedia::Front-End::Controllers::QuizEventController

QuizEventController
<ul style="list-style-type: none"> - <code>\$scope : \$scope</code> - <code>\$location : \$location</code> - <code>\$mdDialog : \$mdDialog</code> <ul style="list-style-type: none"> + <code>QuizEventController(\$scope : \$scope, \$location : \$location, \$mDialog : \$mdDialog)</code> + <code>modifyQuestionnaire(quizId : String) : void</code> + <code>deleteQuestionnaire(quizId : String) : void</code> + <code>subscribeManagement(quizId : String) : void</code> + <code>examModality(quizId : String) : void</code> + <code>resultsQuestionnaire(quizId : String) : void</code>

Figura 33: QuizziPedia::Front-End::Controllers::QuizEventController



- **Descrizione:** questa classe permette di reagire ai comandi dell'utente durante la gestione dei suoi questionari;
- **Utilizzo:** fornisce le funzionalità per reagire ai comandi dell'utente, effettua redirect alle pagine richieste, come la visualizzazione delle statistiche di un questionario e iniziare un questionario in modalità esame;
- **Relazione con altre classi:**
 - **IN EliminationAndModifyDirective:** componente grafico contenente i bottoni per eliminare o modificare un questionario;
 - **IN ExamModalityDirective:** *directive_G* contenete i componenti grafici per attivare la modalità esame su un questionario e gestire le iscrizioni;
 - **IN QuestionnaireResultsDirective:** rappresenta il componente grafico che permette all'utente autenticato pro di vedere i risultati di chi ha compilato il questionario. Tale componente è contenuto nella lista dei questionari abilitati alla compilazione. È possibile accedere alla lista dei risultati azionando l'evento ad esso collegato.
- **Attributi:**
 - - **\$scope:** `$scope`
Campo dati contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
 - - **\$location:** `$location`
Campo dati contenente un riferimento al servizio creato da *Angular_G* che permette di accedere alla barra degli indirizzi del *browser_G*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
 - - **\$mdDialog:** `$mdDialog`
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up.

• **Metodi:**

- + `QuizEventController($scope: $scope, $location: $location, $mdDialog: $mdDialog)`
Metodo costruttore della classe.

Parametri:

* **\$scope:** `$scope`

Parametro contenente un riferimento all'oggetto `$scope` creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;

* **\$location:** `$location`

Parametro contenente un riferimento al servizio creato da *Angular_G* che permette di accedere alla barra degli indirizzi del *browser_G*, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;

* **\$mdDialog:** `$mdDialog`

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up.

- + `modifyQuestionnaire(quizId: String): void`

Metodo che gestisce l'evento click sul pulsante di modifica questionario. Effettua il redirect alla pagina di gestione questionari.

Parametri:



- * quizId: String: parametro che indica l'identificativo univoco di un questionario.
- + deleteQuestionnaire(quizId: String): void
Metodo che gestisce l'evento click sul pulsante di eliminazione questionario. Effettua il redirect alla pagina di gestione questionari.
- Parametri:**
- * quizId: String: parametro che indica l'identificativo univoco di un questionario.
- + subscribeManagement(quizId: String): void
Metodo che gestisce l'evento click sul pulsante di gestione iscrizioni. Effettua il redirect alla pagina di gestione iscrizioni.
- Parametri:**
- * quizId: String: parametro che indica l'identificativo univoco di un questionario.
- + examModality(quizId: String): void
Metodo che gestisce l'evento click sul pulsante di attivazione modalità esame. Effettua il redirect alla pagina di gestione questionari.
- Parametri:**
- * quizId: String: parametro che indica l'identificativo univoco di un questionario.
- + resultsQuestionnaire(quizId: String): void
Metodo che gestisce l'evento click sul pulsante di allenamento. Effettua il redirect alla pagina di gestione questionari.
- Parametri:**
- * quizId: String: parametro che indica l'identificativo univoco di un questionario.

2.3.2.21 QuizziPedia::Front-End::Controllers::RegistrationManagementController

RegistrationManagementController
<ul style="list-style-type: none"> - \$scope : \$scope - \$mdDialog : \$mdDialog - QuizService : QuizService + fillQuiz : RegistrationManagementModelView
<ul style="list-style-type: none"> + RegistrationmanagementController(\$scope : \$scope, \$mdDialog : \$mdDialog, QuizService : QuizService) + subscribeQuestionnaire(username : String) : void

Figura 34: QuizziPedia::Front-End::Controllers::RegistrationManagementController

- **Descrizione:** questa classe permette di gestire le iscrizioni degli utenti ai questionari;
- **Utilizzo:** fornisce le funzionalità di iscrizione ad un questionario;
- **Relazione con altre classi:**
 - IN RegistrationManagementModelView: classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G* *RegistrationManagementView* e il *controller_G* *RegistrationManagementController*;



- **OUT QuizService:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca.

- **Attributi:**

- - **\$scope: \$scope**

Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;

- - **\$mdDialog: \$mdDialog**

Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- - **QuizService: QuizService**

parametro che permette di ottenere, tramite il *service_G*, la lista di tutte le domande presenti nel quiz;

- + **fillQuiz: RegistrationManagementModelView**

Oggetto di tipo *RegistrationManagementModelView*. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G* *RegistrationManagementView* e il *controller_G* *RegistrationManagementController*.

- **Metodi:**

- + **RegistrationmanagementController(\$scope: \$scope, \$mdDialog: \$mdDialog, QuizService: QuizService):**

Metodo costruttore della classe.

Parametri:

- * **\$scope: \$scope**

Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;

- * **\$mdDialog: \$mdDialog**

Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- * **QuizService: QuizService:** parametro che permette di ottenere, tramite il *service_G*, la lista di tutte le domande presenti nel quiz.

- + **subscribeQuestionnaire(username: String): void**

Metodo che permette l'iscrizione ad un questionario. Richiama la funzionalità del *QuizService*.

Parametri:

- * **username: String:** parametro che indica l'utente da iscrivere al questionario.

2.3.2.22 QuizziPedia::Front-End::Controllers::ResultsQuestionnaireController



ResultQuestionnaireController
<ul style="list-style-type: none"> - \$scope : \$scope - \$mdDialog : \$mdDialog - QuizService : QuizService + results : ResultQuestionnaireModelView
<ul style="list-style-type: none"> + ResultsQuestionnaireController(\$scope : \$scope, \$mdDialog : \$mdDialog, QuizService : QuizService) + getQuizResults(quizId : String) : Object + getUserForThisQuestionnaire(quizId : String) : Array<UserDetailsModel>

Figura 35: QuizziPedia::Front-End::Controllers::ResultsQuestionnaireController

- **Descrizione:** questa classe permette di gestire la visualizzazione dei risultati di un singolo questionario;
- **Utilizzo:** fornisce le funzionalità per recuperare i dati dal back-end e mostrarli all'utente nella $view_G$;
- **Relazione con altre classi:**
 - IN **ResultsQuestionnaireModelView**: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **ResultsQuestionnaireView** e il $controller_G$ **ResultsQuestionnaireController**;
 - IN **QuizService**: questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca.
- **Attributi:**
 - - **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - - **\$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
 - - **QuizService: QuizService**
parametro che permette di ottenere, tramite il $service_G$, la lista di tutte le domande presenti nel quiz;
 - + **results: ResultQuestionnaireModelView**
Oggetto di tipo **ResultQuestionnaireModelView**. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **ResultsView** e il $controller_G$ **ResultsController**.
- **Metodi:**
 - + **ResultsQuestionnaireController(\$scope: \$scope, \$mdDialog: \$mdDialog, QuizService: QuizService):**
Metodo costruttore della classe.
 - **Parametri:**
 - * **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;



- * \$mdDialog: \$mdDialog
Campo dati contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
- * QuizService: QuizService: parametro che permette di ottenere, tramite il *serviceG*, la lista di tutte le domande presenti nel quiz.
- + getQuizResults(quizId: String): Object
Metodo che ritorna i risultati di un questionario.

Parametri:

- * quizId: String
Id del questionario del quale recuperare i risultati.
- + getUserForThisQuestionnaire(quizId: String): Array<UserDetailsModel>
Metodo che ritorna tutti gli utenti che hanno eseguito il questionario, con il loro risultato.

Parametri:

- * quizId: String
Id del questionario del quale recuperare gli utenti.

2.3.2.23 QuizziPedia::Front-End::Controllers::SearchController

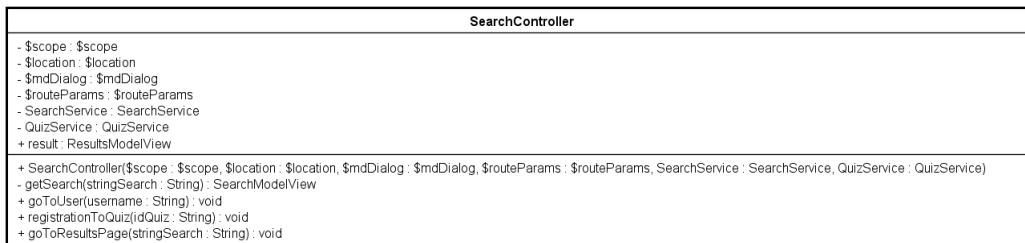


Figura 36: QuizziPedia::Front-End::Controllers::SearchController

- **Descrizione:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione;
- **Utilizzo:** fornisce all'utente le funzionalità di ricerca per utenti e questionari;
- **Relazione con altre classi:**
 - **IN ResultsModelView:** classe di tipo *modelviewG* la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la *viewG ResultsView*, la *directiveG SearchDirective* e il *controllerG ResultsController*;
 - **IN SearchService:** questa classe permette di eseguire una ricerca tra i questionari e gli utenti presenti ritornando un Object contenente i risultati di tale operazione;
 - **IN QuizService:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca. Permette inoltre di iscriversi ad un questionario.
- **Attributi:**



- - **\$scope:** `$scope`
Campo dati contenente un riferimento all'oggetto `$scope` creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
- - **\$location:** `$location`
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- item - **\$mdDialog:** `$mdDialog`
Campo dati contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
- - **\$routeParams:** `$routeParams`
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi e recuperare i parametri passati;
- - **SearchService:** `SearchService`
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alla ricerca. Viene utilizzato il metodo `search` di `SearchService` a cui viene passato come parametro la stringa di ricerca;
- - **QuizService:** `QuizService`
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate ai questionari. Viene utilizzato il metodo `subscribeQuestionnaire` di `QuizService` per iscrivere un utente ad un questionario;
- + **result:** `ResultsModelView`
Oggetto di tipo `ResultsModelView`. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `ResultView` e il $controller_G$ `SearchController`.

• **Metodi:**

- + **SearchController(\$scope: \$scope, \$location: \$location, \$mdDialog: \$mdDialog, \$routeParams: \$routeParams, SearchService: SearchService, QuizService: QuizService)**
Metodo costruttore della classe. Viene eseguita la ricerca per poter poi popolare il campo dati `result`.

Parametri:

- * **\$scope:** `$scope`
Parametro contenente un riferimento all'oggetto `$scope` creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;
- * **\$location:** `$location`
Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- * **\$mdDialog:** `$mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for AngularG* che permette di creare delle componenti a pop-up;
- * **\$routeParams:** `$routeParams`
Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi e recuperare i parametri passati;



- * **SearchService:** SearchService
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alla ricerca. Viene utilizzato il metodo **search** di **SearchService** a cui viene passato come parametro la stringa di ricerca;
- * **QuizService:** QuizService
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate ai questionari. Viene utilizzato il metodo **subscribeQuestionnaire** di **QuizService** per iscrivere un utente ad un questionario.

- - **getSearch(stringSearch: String): SearchModelView**
Metodo che esegue la ricerca utilizzando un metodo fornito dalla classe **SearchService**.

Parametri:

- * **stringSearch:** String
Parametro contenente la stringa sulla quale si deve basare la ricerca.

- + **goToUser(username: String): void**
Metodo che gestisce l'evento click sul bottone per visualizzare il profilo dell'utente selezionato. Effettua il redirect alla pagina dell'utente.

Parametri:

- * **username:** String
Parametro contenente l'username dell'utente di cui si vuole visualizzare il profilo.

- + **registrationToQuiz(idQuiz: String): void**
Metodo che gestisce l'evento click sul pulsante di registrazione al questionario.

Parametri:

- * **idQuiz:** String
Parametro contenente l'id del questionario di cui si vuole effettuare l'iscrizione.

- + **goToResultsPage(stringSearch: String): void**
Metodo che gestisce l'evento click sul pulsante per effettuare una ricerca.

Parametri:

- * **stringSearch:** String
Parametro contenente la stringa sulla quale si deve basare la ricerca.

2.3.2.24 QuizziPedia::Front-End::Controllers::SignUpController

SignUpController
<ul style="list-style-type: none"> - \$scope : \$scope - \$location : \$location - \$mdDialog : \$mdDialog - AuthService : AuthService + newUser : SignUpModelView <ul style="list-style-type: none"> + SignUpController(\$scope : \$scope, \$location : \$location, \$mdDialog : \$mdDialog, AuthService : AuthService) + signUp(user : Object) : void + login() : void

Figura 37: QuizziPedia::Front-End::Controllers::SignUpController

- **Descrizione:** questa classe permette di gestire la registrazione di un utente al sistema;
- **Utilizzo:** fornisce le funzionalità di registrazione di un utente al sistema;
- **Relazione con altre classi:**



- **IN SignUpModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `SignUpView` e il $controller_G$ `SignUpController`;
- **IN AuthService:** questa classe permette di gestire la registrazione e l'autenticazione di un utente.

• **Attributi:**

- - **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione;
- - **\$location: \$location**
Campo dati contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;
- - **\$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- - **AuthService: AuthService**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo `signUp` di `AuthService` a cui viene passato come parametro un oggetto di tipo `SignUpModelView`;
- + **newUser: SignUpModelView**
Oggetto di tipo `SignUpModelView`. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `SignUpView` e il $controller_G$ `SignUpController`.

• **Metodi:**

- + **SignUpController(\$scope: \$scope, \$location: \$location, \$mdDialog: \$mdDialog, AuthService: AuthService)**
Metodo costruttore della classe.

Parametri:

* **\$scope: \$scope**

Parametro contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$. Viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $viewmodel_G$ e il $model_G$ dell'applicazione; Parametro contenente un riferimento al servizio creato da $Angular_G$ che permette di accedere alla barra degli indirizzi del $browser_G$, i cambiamenti all'URL nella barra degli indirizzi si riflettono in questo oggetto e viceversa;

* **\$mdDialog: \$mdDialog**

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

* **AuthService: AuthService**

Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate all'autenticazione. Viene utilizzato il metodo `logIn` di `$textttAuthService` a cui vengono passati i parametri `username` e `password`;



- + signUp(user: Object): void

Metodo che richiama il metodo `signup` del service `AuthService` passandogli un oggetto di tipo `SignUpModelView`. Nel caso di buona riuscita dell'operazione viene mostrato un messaggio di successo. Con l'azione di click sul bottone presentato dal messaggio di successo è possibile effettuare il redirect alla pagina di login dell'applicazione. Nel caso in cui invece avvenga un errore, viene mostrato a video il messaggio di errore.

Parametri:

* user: Object

Parametro che rappresenta un oggetto contenente tutti i parametri per la registrazione.

- + logIn(): void

Metodo che gestisce l'evento click sul pulsante di login. Effettua il redirect alla pagina di login.

2.3.2.25 QuizziPedia::Front-End::Controllers::StatisticsController

StatisticsController
- \$scope : \$scope - StatisticsService : StatisticsService + userStatistic : StatisticsModelView + StatisticsController(\$scope : \$scope, StatisticsService : StatisticsService) - getStatistics(username : String) : Object

Figura 38: QuizziPedia::Front-End::Controllers::StatisticsController

- **Descrizione:** questa classe permette di gestire le statistiche di un utente;
- **Utilizzo:** fornisce le funzionalità per ottenere le statistiche di un utente per poterle mostrare nella $view_G$;
- **Relazione con altre classi:**
 - **IN StatisticsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $directive_G$ `StatisticsDirective` e il $controller_G$ `StatisticsController`;
 - **IN StatisticsService:** questa classe permette di ottenere le statistiche dell'utente;
- **Attributi:**
 - - \$scope: \$scope
Campo dati contenente un riferimento all'oggetto $\$scope$ creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;
 - - StatisticsService: StatisticsService
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle statistiche da visualizzare;



– + **userStatistic:** StatisticsModelView

Oggetto di tipo StatisticsModelView contenente le informazioni delle statistiche. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la directive_G StatisticsDirective e il controller_G StatisticsController.

- **Metodi:**

– + **StatisticsController(\$scope: \$scope, StatisticsService: StatisticsService)**

Metodo costruttore della classe.

Parametri:

* **\$scope: \$scope**

Parametro contenente un riferimento all'oggetto \$scope creato da Angular_G. Viene utilizzato come mezzo di comunicazione tra il controller_G e la view_G. Contiene gli oggetti che definiscono il viewmodel_G e il model_G dell'applicazione;

* **StatisticsService: StatisticsService**

Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle statistiche da visualizzare.

– - **getStatistics(username: String): Object**

Metodo che permette di ottenere le statistiche di un utente grazie all'utilizzo di StatisticsService.

Parametri:

* **username: String**

Parametro contenente la stringa username utilizzata per poter recuperare le giuste statistiche attraverso lo StatisticsService.

2.3.2.26 QuizziPedia::Front-End::Controllers::StringsSortingQuestionsController

StringsSortingQuestionsController
- \$scope : \$scope - QuestionItemModel : QuestionItemModel - \$mdDialog : \$mdDialog - QuestionService : QuestionService - \$routeParams : \$routeParams + StringsSortingQuestionsController(\$scope : \$scope, QuestionItemModel : QuestionItemModel, \$mdDialog : \$mdDialog, QuestionService : QuestionService, \$routeParams : \$routeParams) + submitQuestion() : void

Figura 39: QuizziPedia::Front-End::Controllers::StringsSortingQuestionsController

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento di stringhe;
- **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda a ordinamento di stringhe nel database e per modificarne una esistente;
- **Relazione con altre classi:**

– **IN StringsSortingQuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di Angular_G. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la view_G StringsSortingQuestionsView e il controller_G StringsSortingQuestionsController;

– **IN QuestionService:** questa classe permette di:



- * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
- IN **QuestionItemModel**: questa classe rappresenta il modello di una domanda.

• **Attributi:**

- - **\$scope: \$scope**
Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
- - **QuestionItemModel: QuestionItemModel**
Campo dati che si riferisce alla classe che rappresenta il modello della domanda;
- - **\$mdDialog: \$mdDialog**
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- - **QuestionService: QuestionService**
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
- - **\$routeParams: \$routeParams**
Campo dati contenente il riferimento all'oggetto globale \$routeParams creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'URL.

• **Metodi:**

- + **StringSortingQuestionsController(\$scope: \$scope, QuestionItemModel: QuestionItemModel, \$mdDialog: \$mdDialog, QuestionService: QuestionService, \$routeParams: \$routeParams)**
Metodo costruttore della classe. Se in \$routeParams sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il **QuestionService** il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso \$mdDialog indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in \$routeParams verrà caricata la *view_G* vuota così da poter creare una nuova domanda.

Parametri:

- * **\$scope: \$scope**
Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;
- * **QuestionItemModel: QuestionItemModel**
Parametro che si riferisce alla classe che rappresenta il modello della domanda;
- * **\$mdDialog: \$mdDialog**
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- * **QuestionService: QuestionService**
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
- * **\$routeParams: \$routeParams**
Parametro contenente il riferimento all'oggetto globale \$routeParams creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'URL.



– + submitQuestion(): void

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal *modelview_G* e li manda al *server_G* attraverso **QuestionService**. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.3.2.27 QuizziPedia::Front-End::Controllers::TopicKeywordsController

TopicKeywordsController
<ul style="list-style-type: none"> - \$scope : \$scope - QuestionsService : QuestionsService + key : TopicKeywordsModelView
<ul style="list-style-type: none"> + TopicKeywordsController(\$scope : \$scope, QuestionsService : QuestionsService) + getKeywords(key : String) : Array<String>

Figura 40: QuizziPedia::Front-End::Controllers::TopicKeywordsController

- **Descrizione:** questa classe permette di gestire il recupero delle parole chiave di un questionario;

- **Utilizzo:** fornisce le funzionalità per il recupero delle parole chiave durante la creazione di un questionario;

- **Relazione con altre classi:**

- **IN TopicKeywordsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G CreateQuestionnaireView* e il *controller_G KeywordsController*;
- **IN QuestionsService:** questa classe permette di ottenere domande esistenti e salvare nuove domande.

- **Attributi:**

- - \$scope: \$scope

Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;

- - QuestionsService: QuestionsService

Permette, tra le altre cose, di ottenere le parole chiave a partire da una stringa passata;

- + key: TopicKeywordsModelView

Oggetto di tipo *CreateQuestionnaireView*. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *directive_G TopicKeywordsDirective* e il *controller_G TopicKeywordsController*.

- **Metodi:**

- + TopicKeywordsController(\$scope: \$scope, QuestionsService: QuestionsService)

Metodo costruttore della classe.

Parametri:



- * **\$scope:** \$scope
Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;
- * - **QuestionsService:** QuestionsService
Permette, tra le altre cose, di ottenere le parole chiave a partire da una stringa passata.
- + **getKeywords(key: String): Array<String>**
Metodo che ritorna le parole chiave data una una stringa di ricerca.

Parametri:

- * **key:** String: parametro che identifica la stringa con la quale cercare le keywords.

2.3.2.28 QuizziPedia::Front-End::Controllers::TrainingController

TrainingController
<ul style="list-style-type: none"> - \$scope : \$scope - \$rootScope : \$rootScope - \$mdDialog : \$mdDialog - training : TrainingModelView
<ul style="list-style-type: none"> + TrainingController(\$scope : \$scope, \$rootScope : \$rootScope, \$mdDialog : \$mdDialog, TrainingModeModel : TrainingModeModel) + loadNewQuestionBy(topic : String, keywords : Array<String>, level : Number) : void + startTraining(numberOfQuestions : Number, topic : String, keywords : Array<String>) : void + addResult(questionNumber : Number, result : Boolean) : void + addQuestion(question : QuestionItemModel) : void

Figura 41: QuizziPedia::Front-End::Controllers::TrainingController

- **Descrizione:** questa classe permette di gestire la modalità allenamento sottponendo all'utente le giuste domande adatte al suo livello;
- **Utilizzo:** fornisce le funzionalità per recuperare le domande che siano in accordo con il livello dell'utente;
- **Relazione con altre classi:**
 - **IN TrainingModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile \$scope di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G TrainingView* e il *controller_G TrainingController*;
 - **IN TrainingModeModel:** rappresenta un allenamento. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un allenamento;
 - **IN QuestionController:** questa classe permette di gestire il recupero delle domande per far si che possano essere visualizzate nella modalità allenamento e nella compilazione dei questionari.
- **Attributi:**
 - **\$scope:** \$scope
Campo dati contenente un riferimento all'oggetto \$scope creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
 - **\$rootScope:** \$rootScope
Campo dati contenente il riferimento all'oggetto globale \$rootScope creato da



Angular_G. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto **TrainingModeModel**. In questo caso viene utilizzato per inserire in \$rootScope l'oggetto di ritorno della chiamata a **getNextQuestion**;

- **\$mdDialog: \$mdDialog**

Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- **+ training: TrainingModelView**

Oggetto di tipo **TrainingModelView**. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G* **TrainingView** e il *controller_G* **TrainingController**;

- **Metodi:**

- **+ TrainingController(\$scope: \$scope, \$rootScope: \$rootScope, \$mdDialog: \$mdDialog, TrainingModeModel: TrainingModeModel)**

Metodo costruttore della classe.

Parametri:

- * **\$scope: \$scope**

Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;

- * **\$rootscope: \$rootscope**

Parametro contenente il riferimento all'oggetto globale \$rootScope creato da *Angular_G*. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto **UserDetailsModel**. In questo caso viene utilizzato per aggiornare in \$rootScope l'oggetto che rappresenta l'utente autenticato all'interno dell'applicazione;

- * **\$mdDialog: \$mdDialog**

Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- * **TrainingModeModel: TrainingModeModel**

Rappresenta un allenamento. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un allenamento.

- **+ addQuestion(question: QuestionItemModel): void**

Metodo che gestisce l'evento per inserire una domanda nella cronologia delle domande.

Parametri:

- * **question: QuestionItemModel**

Parametro contenente un riferimento all'oggetto di tipo **QuestionItemModel**.

- **+ loadNewQuestionBy(topic: String, keywords: Array<String>, level: Number): void**

Metodo che emette l'evento per scaricare una nuova domanda in base ai parametri passati. Fa una richiesta al **QuestionsController** che andrà a scaricare la domanda e ad inserirla in **TrainingModeModel** nello \$scope mediante il metodo **addQuestion**;

Parametri:

- * **topic: String**

Parametro contenente l'argomento della domanda;

- * **keywords: Array<String>**

Parametro contenente un array di stringhe che rappresenta le keywords scelte per l'allenamento;



```
* level: Number
    Parametro contenente il livello dell'utente.

- + addResult(questionNumber: Number, result: Boolean): void
    Metodo che gestisce l'evento per inserire il risultato di una domanda.

Parametri:
* questionNumber: Number
    Parametro contenente il numero della domanda risposta;
* result: Boolean
    Parametro contenente il risultato della domanda risposta.

- + startTraining(numberOfQuestions: Number, topic: String, keywords: Array<String>): void
    Metodo che gestisce l'evento per iniziare l'allenamento.

Parametri:
* numberOfQuestions: Number
    Parametro contenente il numero di domande per l'allenamento;
* topic: String
    Parametro contenente l'argomento dell'allenamento;
* keywords: Array<String>
    Parametro contenente un array di stringhe che rappresenta le keywords scelte per l'allenamento.
```

2.3.2.29 QuizziPedia::Front-End::Controllers::TrueFalseQuestionsController

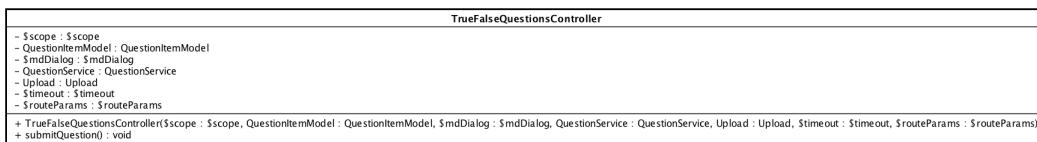


Figura 42: QuizziPedia::Front-End::Controllers::TrueFalseQuestionsController

- **Descrizione:** questa classe permette di gestire la creazione e la modifica di una domanda vero/falso;
- **Utilizzo:** fornisce le funzionalità per inserire una nuova domanda vero/falso nel database e per modificarne una esistente;
- **Relazione con altre classi:**
 - **IN QuestionService:** questa classe permette di:
 - * Ottenerne una domanda attraverso il metodo dedicato;
 - * Caricare una domanda modificata;
 - * Caricare una nuova domanda.
 - **IN QuestionItemModel:** questa classe rappresenta il modello di una domanda;
 - **OUT TrueFalseQuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G* *TrueFalseQuestionsView* e il *controller_G* *TrueFalseQuestionsController*.
- **Attributi:**



- - **\$scope:** `$scope`
Campo dati contenente un riferimento all'oggetto `$scope` creato da *Angular_G*, viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *model_G* dell'applicazione;
- - **QuestionItemModel:** `QuestionItemModel`
Campo dati che si riferisce alla classe che rappresenta il modello della domanda;
- - **\$mdDialog:** `$mdDialog`
Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- - **QuestionService:** `QuestionService`
Campo dati contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;
- - **Upload:** `Upload`
Campo dati contenente un riferimento alla libreria *ng-file-upload_G* necessaria per il caricamento della foto profilo dell'utente;
- - **\$timeout:** `$timeout`
Campo dati contenente il riferimento all'oggetto globale `$timeout` creato da *Angular_G*. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una *promise_G*, la quale sarà risolta quando avverrà il ritardo e la funzione di `$timeout` sarà eseguita;
- - **\$routeParams:** `$routeParams`
Campo dati contenente il riferimento all'oggetto globale `$routeParams` creato da *Angular_G*. Tale servizio permette di recuperare il set di variabili presenti nell'URL.

• **Metodi:**

- + **TrueFalseQuestionsController(\$scope: \$scope, QuestionItemModel: QuestionItemModel, \$mdDialog: \$mdDialog, QuestionService: QuestionService, Upload: Upload, \$timeout: \$timeout, \$routeParams: \$routeParams)**
Metodo costruttore della classe. Se in `$routeParams` sarà presente il codice univoco che rappresenta una domanda e di questa il creatore è l'utente autenticato, allora verrà scaricato attraverso il **QuestionService** il contenuto della domanda così da poterlo modificare. In caso contrario verrà mostrato un errore attraverso `$mdDialog` indicando che i privilegi per tale operazione sono negati. Nel caso in cui non ci sarà tale parametro in `$routeParams` verrà caricata la *view_G* vuota così da poter creare una nuova domanda.

Parametri:

- * **\$scope:** `$scope`
Parametro contenente un riferimento all'oggetto `$scope` creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;
- * **QuestionItemModel:** `QuestionItemModel`
Parametro che si riferisce alla classe che rappresenta il modello della domanda;
- * **\$mdDialog:** `$mdDialog`
Parametro contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;
- * **QuestionService:** `QuestionService`
Parametro contenente un riferimento al servizio che si occupa della gestione delle informazioni legate alle domande;



- * **Upload:** `Upload`
Parametro contenente un riferimento alla libreria $ng\text{-}file\text{-}upload_G$ necessaria per il caricamento della foto profilo dell'utente;
- * **\$timeout:** `$timeout`
Parametro contenente il riferimento all'oggetto globale `$timeout` creato da $Angular_G$. Il valore di ritorno di una chiamata alla funzione di `$timeout` è una $promise_G$, la quale sarà risolta quando avverrà il ritardo e la funzione di `$timeout` sarà eseguita;
- * **\$routeParams:** `$routeParams`
Parametro contenente il riferimento all'oggetto globale `$routeParams` creato da $Angular_G$. Tale servizio permette di recuperare il set di variabili presenti nell'URL.
- **+ submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal $modelview_G$ e li manda al $server_G$ attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.3.2.30 QuizziPedia::Front-End::Controllers::UserDetailsController

UserDetailsController
<ul style="list-style-type: none"> - <code>\$scope : \$scope</code> - <code>\$rootScope : \$rootScope</code> - <code>\$mdDialog : \$mdDialog</code> - <code>userDetailsService : userDetailsService</code> + <code>details : UserDetailsModelView</code> <ul style="list-style-type: none"> + <code>UserDetailsController(\$scope : \$scope, \$rootScope : \$rootScope, \$mdDialog : \$mdDialog, UserDetailsService : UserDetailsService)</code> - <code>getUserDetails(username : String) : UserDetailsModel</code>

Figura 43: QuizziPedia::Front-End::Controllers::UserDetailController

- **Descrizione:** questa classe permette di gestire i dati di un utente da mostrare nella pagina di un profilo;
- **Utilizzo:** fornisce le funzionalità per ottenere i dati di un utente per poterle mostrare nella $view_G$;
- **Relazione con altre classi:**
 - **IN UserDetailsModelView:** $directive_G$ che permette di visualizzare i dati di un utente;
 - **IN UserDetailsService:** questa classe permette di ottenere i dati dell'utente;
 - **IN UserDetailsModel:** rappresenta un utente. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un utente sia nella visualizzazione che nella gestione di un profilo.
- **Attributi:**
 - **- \$scope: \$scope**
Campo dati contenente un riferimento all'oggetto `$scope` creato da $Angular_G$, viene utilizzato come mezzo di comunicazione tra il $controller_G$ e la $view_G$. Contiene gli oggetti che definiscono il $model_G$ dell'applicazione;



- - **\$rootScope: \$rootScope**
Campo dati contenente il riferimento all'oggetto globale \$rootScope creato da *Angular_G*. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto *UserDetailsModel*. In questo caso viene utilizzato per inserire in \$rootScope l'oggetto di ritorno della chiamata a *getUserDetails* del *service_G* *UserDetailsService*;
- - **userDetailsService: userDetailsService**
Questa classe permette di ottenere i dati personali degli utenti;
- + **details: UserDetailsModelView**
Oggetto di tipo *UserDetailsModelView*. All'interno di esso sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *directive_G* *UserDetailsDirective* e il *controller_G* *UserDetailsController*.

- **Metodi:**

- + **UserDetailsController(\$scope: \$scope, \$rootScope:\$rootScope, \$mdDialog: \$mdDialog, UserDetailsService: UserDetailsService)**
Metodo costruttore della classe.

Parametri:

- * **\$scope: \$scope**

Parametro contenente un riferimento all'oggetto \$scope creato da *Angular_G*. Viene utilizzato come mezzo di comunicazione tra il *controller_G* e la *view_G*. Contiene gli oggetti che definiscono il *viewmodel_G* e il *model_G* dell'applicazione;

- * **\$rootScope: \$rootScope**

Parametro contenente il riferimento all'oggetto globale \$rootScope creato da *Angular_G*. Viene utilizzato per rendere accessibile a tutti i *controller_G* e a tutte le *view_G* l'oggetto *UserDetailsModel*. In questo caso viene utilizzato per inserire in \$rootScope l'oggetto di ritorno della chiamata a *getUserDetails* del *service_G* *UserDetailsService*;

- * **\$mdDialog: \$mdDialog**

Campo dati contenente un riferimento al servizio della libreria *Material for Angular_G* che permette di creare delle componenti a pop-up;

- * **userDetailsService: userDetailsService:** parametro che permette di ottenere, tramite il *service_G*, la lista di tutti i dati dell'utente.

- - **getUserDetails(username: String): UserDetailsModel**

Metodo che permette di ottenere i dati con una chiamata a *UserDetailsService*.

Parametri:

- * **username: String:** parametro che identifica l'utente del quale saranno scaricati i dati.



2.4 QuizziPedia::Front-End::Directives

2.4.1 Informazioni generali

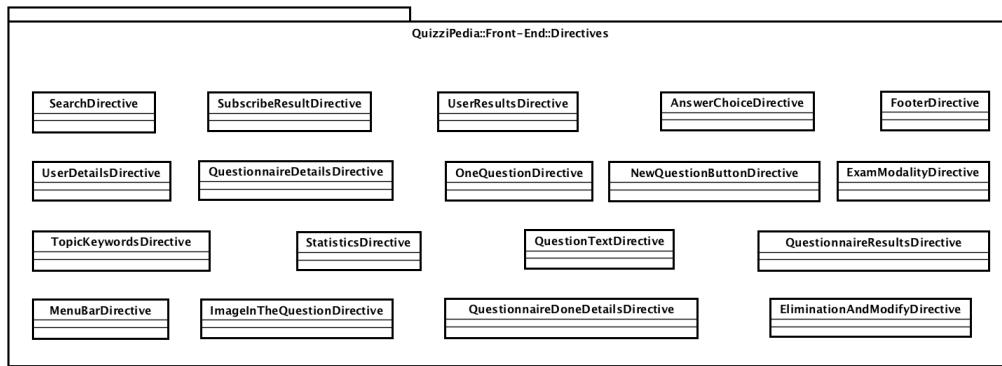


Figura 44: QuizziPedia::Front-End::Directives

- **Descrizione:** package contenente le *directives_G*;
- **Padre:** Front-End;
- **Interazione con altri componenti:**
 - **Controllers:** package contenente i *controllers_G* front-end dell'applicazione;
 - **Views:** package contenente le *views_G* front-end dell'applicazione.

2.4.2 Classi

2.4.2.1 QuizziPedia::Front-End::Directives::EliminationAndModifyDirective

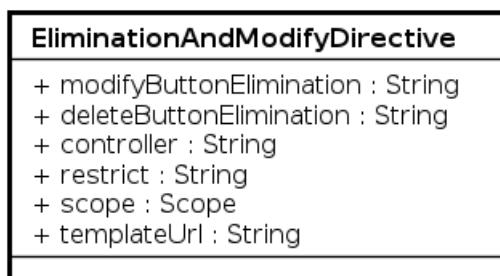


Figura 45: QuizziPedia::Front-End::Directives:EliminationAndModifyDirective

- **Descrizione:** componente grafico contenente i bottoni per eliminare o modificare un questionario;
- **Utilizzo:** permette di eliminare un questionario o di modificarne uno esistente;
- **Relazioni con altre classi:**
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione;



- **IN QuizEventModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le $directives_G$ `EliminationAndModifyDirective`, `ExamModalityDirective` e `QuestionnaireResultsDirective` e il $controller_G$ `QuizEventController`;
- **OUT QuestionnaireManagementView:** $view_G$ principale per la gestione dei questionari.

- **Attributi:**

- **+ modifyButtonElimination: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di modifica del questionario selezionato, in italiano o in inglese;
- **+ deleteButtonElimination: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di eliminazione del questionario selezionato, in italiano o in inglese;
- **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope: Scope**
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttiva.

2.4.2.2 QuizziPedia::Front-End::Directives::ExamModalityDirective

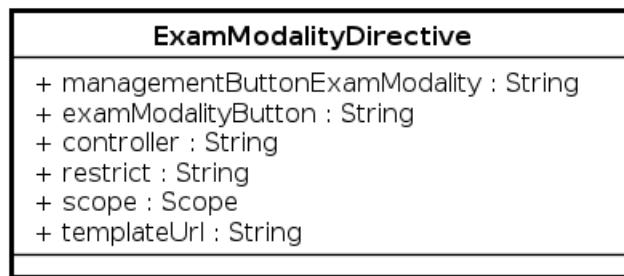


Figura 46: QuizziPedia::Front-End::Directives::ExamModalityDirective

- **Descrizione:** *directive* contenete i componenti grafici per attivare la modalità esame su un questionario e gestire le iscrizioni;
- **Utilizzo:** permette di attivare la modalità esame su un questionario e di gestirne le iscrizioni;
- **Relazioni con altre classi:**
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione;



- **IN QuizEventModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le $directives_G$ `EliminationAndModifyDirective`, `ExamModalityDirective` e `QuestionnaireResultsDirective` e il $controller_G$ `QuizEventController`;
- **OUT QuestionnaireManagementView:** $view_G$ principale per la gestione dei questionari.

- **Attributi:**

- **+ managementButtonExamModality: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di gestione delle iscrizioni al questionario selezionato, in italiano o in inglese;
- **+ examModalityButton: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di attivazione della modalità esame del questionario selezionato, in italiano o in inglese;
- **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope: Scope**
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttiva.

2.4.2.3 QuizziPedia::Front-End::Directives::FooterDirective

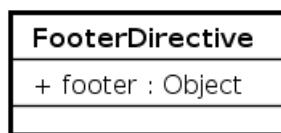


Figura 47: QuizziPedia::Front-End::Directives:FooterDirective

- **Descrizione:** $directive_G$ contenente i componenti grafici del footer dell'applicazione;
- **Utilizzo:** premette di visualizzare le informazioni contenenti nel footer in ogni pagina dell'applicazione;
- **Relazioni con altre classi:**
 - **IN Index:** contenitore generale dell'applicazione.
- **Attributi:**
 - **+ footer: Object**
Oggetto contenente le informazioni presenti nel footer:



```
* description: String
    Breve descrizione di cosa fa QuizziPedia;
* info: Array<Object>
    Array di informazioni varie. Un oggetto contenuto è così formato:
        · name: String
            Nome riassuntivo dell'informazione;
        · link: String
            Collegamento ipertestuale per l'informazione;
        · description: String
            Breve descrizione di cosa tratta questa informazione.
```

2.4.2.4 QuizziPedia::Front-End::Directives::ImageInTheQuestionDirective

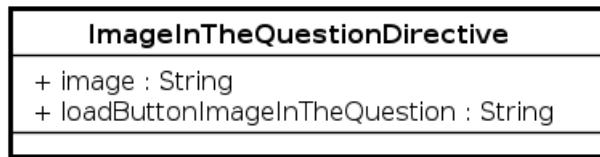


Figura 48: QuizziPedia::Front-End::Directives::ImageInTheQuestionDirective

- **Descrizione:** *directive_G* contenente i componenti grafici per l'inserimento dell'immagine nella creazione delle domande;
- **Utilizzo:** permette di inserire un'immagine in una domanda;
- **Relazioni con altre classi:**
 - **OUT TrueFalseQuestionsView:** *view_G* contenente i campi per creare una domanda vero/falso;
 - **OUT MultipleQuestionsView:** *view_G* contenente i campi per creare una domanda a risposta multipla;
 - **OUT ImagesSortingQuestionsView:** *view_G* contenente i campi per creare una domanda a ordinamento immagini;
 - **OUT ClickableAreaQuestionsView:** *view_G* contenente i campi per creare una domanda ad area cliccabile.
- **Attributi:**
 - + image: String
Attributo contenente l'URL dell'immagine caricata dall'utente;
 - + loadButtonImageInTheQuestion: String
Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di caricamento dell'immagine, in italiano o in inglese.

2.4.2.5 QuizziPedia::Front-End::Directives::MenuBarDirective

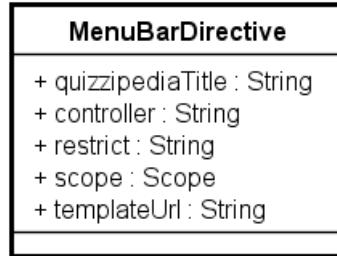


Figura 49: QuizziPedia::Front-End::Directives::MenuBarDirective

- **Descrizione:** rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello \$scope isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata;
- **Utilizzo:** viene utilizzato per realizzare il menù, presente in ogni pagina dell'applicazione, che permette all'utente di selezionare un'opzione in base al contesto in cui si trova:
 - Autenticazione;
 - Registrazione;
 - Ricerca;
 - Visualizzare il proprio profilo utente;
 - Gestire le domande create;
 - Gestire i questionari creati.
- **Relazioni con altre classi:**
 - **OUT Index:** $view_G$ generale dell'applicazione;
 - **IN MenuBarModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ Index e il $controller_G$ *MenuBarController*;
 - **IN SearchDirective:** $directive_G$ che permette di effettuare la ricerca di utenti e questionari;
 - **IN LoginBarDirective:** $directive_G$ contenente il componente che permette di effettuare il redirect alla pagina di login;
 - **IN SignUpBarDirective:** $directive_G$ contenente il componente che permette di effettuare il redirect alla pagina di registrazione;
 - **IN UserBarDirective:** $directive_G$ contenente il componente che permette di effettuare il redirect alla pagina di visualizzazione del profilo utente personale;
 - **IN ProfileManagementBarDirective:** $directive_G$ contenente il componente che permette di effettuare il redirect alla pagina di gestione del profilo;
 - **IN QuestionsManagementBarDirective:** $directive_G$ contenente il componente che permette di effettuare il redirect alla pagina di gestione delle domande;
 - **IN LogoutBarDirective:** $directive_G$ contenente il componente che permette di effettuare il logout;
 - **IN QuestionnaireManagementBarDirective:** $directive_G$ contenente il componente che permette di effettuare il redirect alla pagina di gestione dei questionari.



- **Attributi:**

- + **quizzipediaTitle:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il titolo e la descrizione dell'applicazione;
- + **controller:** String
Stringa contenente il nome del $controller_G$ della direttiva;
- + **restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- + **scope:** Scope: oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- + **templateUrl:** String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttiva.

2.4.2.6 QuizziPedia::Front-End::Directives::OneQuestionDirective

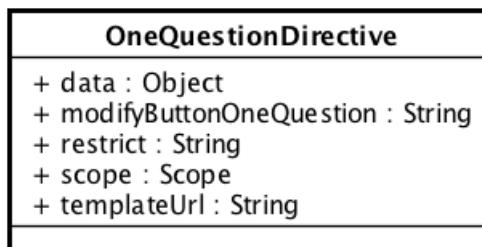


Figura 50: QuizziPedia::Front-End::Directives::OneQuestionDirective

- **Descrizione:** rappresenta il componente grafico che visualizza all'utente l'anteprima della domanda che ha creato. Eseguendo l'azione di click sul pulsante di modifica sarà possibile modificare tale domanda. All'interno di **QuestionsManagementsView** verranno stampati a video tanti componenti quanti presenti nello \$scope isolato ad esso associato;
- **Utilizzo:** viene utilizzato per permettere all'utente di visualizzare le domande che ha creato;
- **Relazioni con altre classi:**

- **IN QuestionsManagementView:** $view_G$ contenente l'elenco delle domande create;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- + **data:** Object
Oggetto contenente le informazioni da mostrare nell'anteprima della domanda;
- + **modifyButtonOneQuestion:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di modifica della domanda, in italiano o in inglese;
- + **restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;



- **+ scope:** `Scope`: oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl:** `String`
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.7 QuizziPedia::Front-End::Directives::NewQuestionButtonDirective

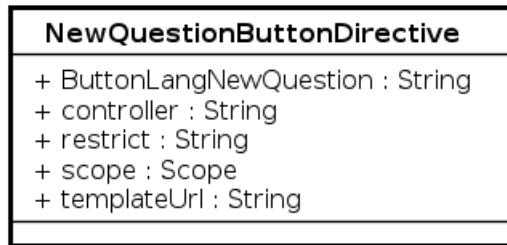


Figura 51: QuizziPedia::Front-End::Directives::NewQuestionButtonDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di posizionarsi nella $view_G$ di creazione di una nuova domanda;
- **Utilizzo:** viene utilizzato per permettere all'utente di posizionarsi nella $view_G$ di creazione di una nuova domanda;
- **Relazioni con altre classi:**
 - **IN NewQuestionsButtonsController:** questa classe permette di effettuare il redirect alla pagina di creazione nuova domanda;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione;
 - **OUT QuestionsManagementView:** $view_G$ contenente l'elenco delle domande create.
- **Attributi:**
 - **+ ButtonLangNewQuestion:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di creazione di una nuova domanda, in italiano o in inglese;
 - **+ controller:** `String`
Stringa contenente il nome del $controller_G$ della direttiva;
 - **+ restrict:** `String`
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
 - **+ scope:** `Scope`
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
 - **+ templateUrl:** `String`
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.8 QuizziPedia::Front-End::Directives::QuestionTextDirective



Figura 52: QuizziPedia::Front-End::Directives::QuestionTextDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
- **Utilizzo:** viene usato per permettere all'utente di scrivere o modificare il testo di una domanda;
- **Relazioni con altre classi:**
 - **OUT TrueFalseQuestionsView:** $view_G$ contenente i campi per creare una domanda vero/falso;
 - **OUT MultipleQuestionsView:** $view_G$ contenente i campi per creare una domanda a risposta multipla;
 - **OUT ConnectionQuestionsView:** $view_G$ contenente i campi per creare una domanda a collegamento;
 - **OUT ImagesSortingQuestionsView:** $view_G$ contenente i campi per creare una domanda a ordinamento immagini;
 - **OUT StringsSortingQuestionsView:** $view_G$ contenente i campi per creare una domanda a ordinamento stringhe;
 - **OUT FillingQuestionsView:** $view_G$ contenente i campi per creare una domanda a riempimento testo;
 - **OUT ClickableAreaQuestionsView:** $view_G$ contenente i campi per creare una domanda ad area cliccabile.
- **Attributi:**
 - + questionText : String
Attributo contenente il testo della domanda.

2.4.2.9 QuizziPedia::Front-End::Directives::QuestionnaireDetailsDirective

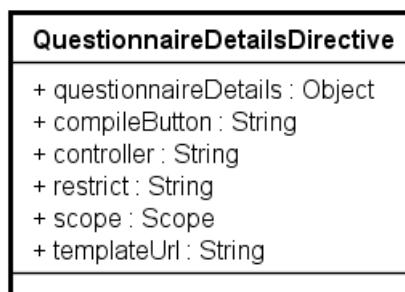


Figura 53: QuizziPedia::Front-End::Directives::QuestionnaireDetailsDirective



- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare la lista di questionari che può compilare. Ogni elemento di questa lista contiene:

- Nome del questionario;
- Autore del questionario;
- Argomento del questionario;
- Parole chiave del questionario.

Al verificarsi dell'evento click su un elemento della lista l'utente verrà indirizzato alla *view_G* per la compilazione del questionario selezionato;

- **Utilizzo:** viene utilizzato per permettere all'utente di visualizzare la lista di questionari che può compilare;

- **Relazioni con altre classi:**

- **IN UserView:** *view_G* contenente i dati personali dell'utente, le sue statistiche relative ai questionari e agli allenamenti effettuati e i questionari a cui è iscritto;
- **IN QuestionnaireDetailsController:** questa classe permette di gestire i dettagli di un questionario;
- **IN QuestionnaireDetailsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile \$scope di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G* *UserView* e il *controller_G* *QuestionnaireDetailsController*;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ questionnaireDetails: Object**
Oggetto contenente i seguenti campi dati:
 - * **name: String**
Nome del questionario;
 - * **author: String**
Autore del questionario;
 - * **topic: String**
Argomento del questionario;
 - * **keywords: Array<String>**
Parole chiave del questionario.
- **+ compileButton: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di compilazione del questionario, in italiano o in inglese;
- **+ controller: String**
Stringa contenente il nome del *controller_G* della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope: Scope**
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file *HTML_G* che contiene la direttive.



2.4.2.10 QuizziPedia::Front-End::Directives::QuestionnaireDoneDirective

QuestionnaireDetailsDoneDirective
<pre>+ questionnaireDetailsDone : Object + controller : String + restrict : String + scope : Scope + templateUrl : String</pre>

Figura 54: QuizziPedia::Front-End::Directives::QuestionnaireDetailsDoneDirective

- **Descrizione:** rappresenta il componente grafico che permette all’utente di visualizzare la lista di questionari che ha già compilato e di conseguenza vederne le valutazioni. Ogni elemento di questa lista contiene:
 - Nome del questionario;
 - Autore del questionario;
 - Argomento del questionario;
 - Parole chiave del questionario;
 - Valutazione del questionario.
- **Utilizzo:** viene utilizzato per permettere all’utente di visualizzare la lista di questionari che ha compilato;
- **Relazioni con altre classi:**
 - **OUT UserView:** $view_G$ contenente i dati personali dell’utente, le sue statistiche relative ai questionari e agli allenamenti effettuati e i questionari a cui è iscritto;
 - **IN QuestionnaireDetailsController:** questa classe permette di gestire i dettagli di un questionario;
 - **IN QuestionnaireDetailsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all’interno della variabile di ambiente $\$scope$ di $Angular_G$. All’interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* tra la $view_G$ **UserView** e il $controller_G$ **QuestionnaireDetailsController**.
- **Attributi:**
 - **+ questionnaireDetailsDone: Object**
Oggetto contenente i seguenti campi dati:
 - * **name: String**
Nome del questionario;
 - * **author: String**
Autore del questionario;
 - * **topic: String**
Argomento del questionario;
 - * **keywords: Array<String>**
Parole chiave del questionario;
 - * **judgement: Number**
Campo che indica il risultato del questionario.



- **+ controller:** String
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope:** Scope
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl:** String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.11 QuizziPedia::Front-End::Directives::QuestionnaireResultsDirective

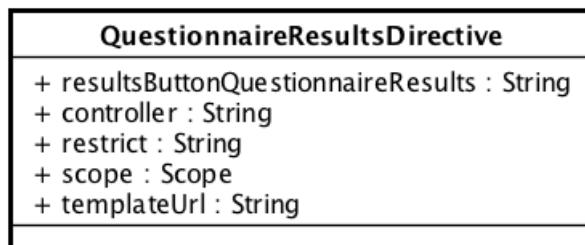


Figura 55: QuizziPedia::Front-End::Directives::QuestionnaireResultsDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente autenticato di vedere i risultati di chi ha compilato il questionario. Tale componente è contenuto nella lista dei questionari abilitati alla compilazione. È possibile accedere alla lista dei risultati azionando l'evento ad esso collegato;
- **Utilizzo:** viene utilizzato per visualizzare i questionari abilitati alla compilazione e permettere all'utente di accedere alle statistiche ad essi associati;
- **Relazioni con altre classi:**
 - **IN QuizEventManagerView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le $directives_G$ `EliminationAndModifyDirective`, `ExamModalityDirective` e `QuestionnaireResultsDirective` e il $controller_G$ `QuizEventController`;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione;
 - **OUT QuestionnaireManagementView:** $view_G$ principale per la gestione dei questionari.

- **Attributi:**

- **+ resultsButtonQuestionnaireResults:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di visualizzazione dei questionari, in italiano o in inglese;
- **+ controller:** String
Stringa contenente il nome del $controller_G$ della direttiva;



- **+ restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope:** Scope
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl:** String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.12 QuizziPedia::Front-End::Directives::SearchDirective



Figura 56: QuizziPedia::Front-End::Directives::SearchDirective

- **Descrizione:** $directive_G$ che permette di effettuare la ricerca di utenti e questionari;
- **Utilizzo:** permette all'utente di effettuare ricerche, è formata da:
 - Barra di ricerca;
 - Pulsante per effettuare la ricerca.
- **Relazioni con altre classi:**
 - **OUT HomeView:** $view_G$ contenente la barra di ricerca per gli utenti e questionari e il bottone che porterà l'utente nella modalità allenamento;
 - **OUT MenuBarDirective:** rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello \$scope isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata;
 - **IN ResultsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **ResultsView**, la $directive_G$ **SearchDirective** e il $controller_G$ **ResultsController**;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ userOrQuestionnaire:** String
Attributo che contiene l'informazione cercata;



- **+ searchButton:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di ricerca, in italiano o in inglese;
- **+ controller:** String
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope:** Scope
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl:** String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.13 QuizziPedia::Front-End::Directives::StatisticsDirective

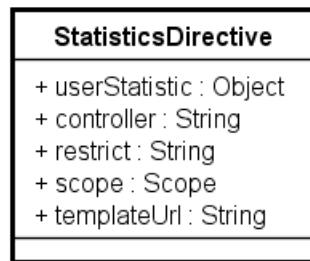


Figura 57: QuizziPedia::Front-End::Directives::StatisticsDirective

- **Descrizione:** $directive_G$ che permette di visualizzare le statistiche di un utente;
- **Utilizzo:** viene utilizzata per mostrare le statistiche nella pagina della visualizzazione del profilo e nella pagina di un utente ricercato;
- **Relazioni con altre classi:**
 - **OUT UserView:** $view_G$ contenente i dati personali dell'utente, le sue statistiche relative ai questionari e agli allenamenti effettuati e i questionari a cui è iscritto;
 - **OUT OtherUserView:** $view_G$ contenente i dati personali e le statistiche di un utente ricercato;
 - **IN StatisticsController:** questa classe permette di gestire le statistiche di un utente;
 - **IN StatisticsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $directive_G$ **StatisticsDirective** e il $controller_G$ **StatisticsController**.
- **Attributi:**
 - **+ userStatistic:** Object
Oggetto contenente le statistiche di un utente;



- **+ controller:** String
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope:** Scope
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl:** String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.14 QuizziPedia::Front-End::Directives::SubscribeResultDirective

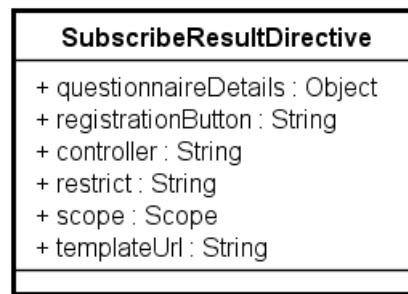


Figura 58: QuizziPedia::Front-End::Directives::SubscribeResultDirective

- **Descrizione:** directive che permette di visualizzare e iscriversi ai questionari ricercati;
- **Utilizzo:** permette di visualizzare e iscriversi ai questionari ricercati. Include un pulsante per ogni questionario che permette l'iscrizione ad esso.
- **Relazioni con altre classi:**
 - **IN ResultsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la view **ResultsView**, le directive e il controller **SearchController**;
 - **OUT ResultsView:** $view_G$ contenente i risultati della ricerca effettuata. Vengono visualizzati sia gli utenti che i questionari trovati;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ questionnaireDetails:** Object
Oggetto contenente i seguenti campi dati:
 - * **name:** String
Nome del questionario;
 - * **author:** String
Username dell'autore del questionario;
 - * **topic:** String
Argomento del questionario;



```

* keywords: Array<String>
    Array di stringhe contenente le parole chiave associate al questionario;
* idQuiz: ObjectId
    Codice identificativo del questionario di tipo ObjectId.

- + registrationButton: String
    Attributo che viene utilizzato per visualizzare la giusta traduzione della  $label_G$  per il bottone di iscrizione al questionario, in italiano o in inglese;

- + controller: String
    Stringa contenente il nome del  $controller_G$  della direttiva;

- + restrict: String: stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;

- + scope: Scope: oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;

- + templateUrl: String: stringa contenente il percorso del file  $HTML_G$  che contiene la direttive.

```

2.4.2.15 QuizziPedia::Front-End::Directives::TopicKeywordsDirective

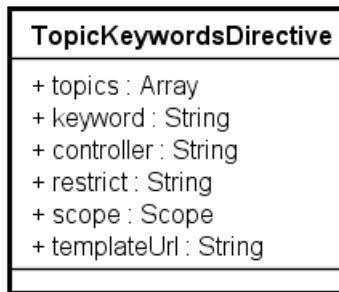


Figura 59: QuizziPedia::Front-End::Directives::TopicKeywordsDirective

- **Descrizione:** directive che permette di gestire l'inserimento dell'argomento e delle keywords al momento della creazione della domanda;
- **Utilizzo:** permette l'inserimento di keywords al momento di creazione della domanda, in particolare sarà formata da:
 - Un menù a tendina per selezionare l'argomento della domanda;
 - Un campo di testo in cui inserire le keywords.
- **Relazioni con altre classi:**
 - IN TopicKeywordsModelView: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la directive **TopicKeywordsDirective** e il controller **TopicKeywordsController**;
 - IN CreateQuestionnaireView: $view_G$ per la creazione del questionario;
 - IN TrueFalseQuestionsView: $view_G$ contenente i campi per creare una domanda vero/falso;
 - IN MultipleQuestionsView: $view_G$ contenente i campi per creare una domanda a risposta multipla;



- **IN ConnectionQuestionsView:** $view_G$ contenente i campi per creare una domanda a collegamento;
- **IN ImagesSortingQuestionsView:** $view_G$ contenente i campi per creare una domanda a ordinamento immagini;
- **IN StringsSortingQuestionsView:** $view_G$ contenente i campi per creare una domanda a ordinamento stringhe;
- **IN FillingQuestionsView:** $view_G$ contenente i campi per creare una domanda a riempimento testo;
- **IN ClickableAreaQuestionsView:** $view_G$ contenente i campi per creare una domanda ad area cliccabile;
- **IN EditorQMLView:** $view_G$ contenente l'editor QML per la creazione di domande personalizzate;

- **Attributi:**

- **+ topics:** `Array<String>`
Array contenente le stringhe dei nomi degli argomenti;
- **+ keyword:** `String`
Attributo contenente la keyword associata alla domanda/questionario;
- **+ controller:** `String`
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict:** `String`
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope:** `Scope`
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl:** `String`
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.16 QuizziPedia::Front-End::Directives::UserDetailsDirective

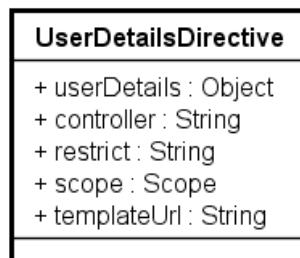


Figura 60: QuizziPedia::Front-End::Directives::UserDetailsDirective

- **Descrizione:** directive che permette di visualizzare i dati personali di un utente;
- **Utilizzo:** permette di visualizzare i dati personali di un utente, in dettaglio conterrà:
 - Username;
 - Immagine.



- **Relazioni con altre classi:**

- **OUT UserView:** $view_G$ contenente i dati personali dell’utente, le sue statistiche relative ai questionari e agli allenamenti effettuati e i questionari a cui è iscritto;
- **OUT OtherUserView:** $view_G$ contenente i dati personali e le statistiche di un utente ricercato;
- **IN UserDetailsController:** questa classe permette di gestire i dati di un utente da mostrare nella pagina di un profilo;
- **IN UserDetailsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all’interno della variabile di ambiente $\$scope$ di $Angular_G$. All’interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la directive `UserDetailsDirective` e il controller G `UserDetailsController`.

- **Attributi:**

- **+ userDetails: Object**
Oggetto contenente i seguenti campi dati:
 - * **username: String**
Campo che identifica univocamente l’utente all’interno dell’applicazione.
- **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all’interno della pagina;
- **+ scope: Scope**
Oggetto scope interno della direttiva, contiene le funzionalità per gestire i dati presenti all’interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.17 QuizziPedia::Front-End::Directives::UserResultsDirective

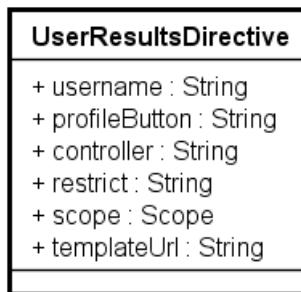


Figura 61: QuizziPedia::Front-End::Directives::UserResultsDirective

- **Descrizione:** directive che permette di visualizzare la lista degli utenti ricercati dopo aver utilizzato l’apposita funzione di ricerca;
- **Utilizzo:** permette di visualizzare la lista degli utenti, in particolare conterrà:
 - Username dell’utente;



- Pulsante per poter essere reindirizzati alla pagina di visualizzazione del profilo dell’utente selezionato.

- **Relazioni con altre classi:**

- **IN ResultsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all’interno della variabile di ambiente $\$scope$ di $Angular_G$. All’interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la view **ResultsView**, le directive e il *controller_G SearchController*;
- **OUT ResultsView:** $view_G$ contenente i risultati della ricerca effettuata. Vengono visualizzati sia gli utenti che i questionari trovati;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell’applicazione.

- **Attributi:**

- **+ username: String**
Attributo che conterrà l’username dell’utente;
- **+ profileButton: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di visualizzazione del profilo utente, in italiano o in inglese;
- **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all’interno della pagina;
- **+ scope: Scope**
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all’interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.18 QuizziPedia::Front-End::Directives::ClickableAnswerDirective

ClickableAnswerDirective
+ controller : String + restrict : String + scope : Scope + questionText : String + image : String + answers : Array<Object> + templateUrl : String

Figura 62: QuizziPedia::Front-End::Directives::ClickableAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad area cliccabile nell’immagine. Viene visualizzato dinamicamente all’interno delle views **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;



- **Utilizzo:** viene utilizzato per consentire all'utente la compilazione della domanda ad area cliccabile nell'immagine;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le directive che compongono dinamicamente la vista della domanda e il $controller_G$ **QuestionsController**;
 - **OUT TrainingView:** $view_G$ principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT FillingQuestionnaireView:** $view_G$ principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ questionText: String**
Identifica il testo della domanda;
 - **+ image: String**
Identifica l'url di una possibile immagine nella domanda;
 - **+ answers: Array<Object>**
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * **+ type: String**
Indica la tipologia della risposta;
 - * **+ text: String**
Contiene il testo dell'affermazione;
 - * **+ url: String**
Rappresenta l'immagine della risposta;
 - * **+ attributesForClickableArea: Mixed**
Contiene i seguenti attributi:
 1. **+ x: Number**
Rappresenta la coordinata x di una area cliccabile;
 2. **+ y: Number**
Rappresenta la coordinata y di una area cliccabile.
 - **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
 - **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
 - **+ scope: Scope**
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
 - **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.19 QuizziPedia::Front-End::Directives::EmptySpaceAnswerDirective



EmptySpaceAnswerDirective
+ controller : String + restrict : String + scope : Scope + questionText : String + image : String + answers : Array<Object> + templateUrl : String

Figura 63: QuizziPedia::Front-End::Directives::EmptySpaceAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all’utente di visualizzare l’esercizio a riempimento di spazi vuoti. Viene visualizzato dinamicamente all’interno delle views **TrainingView** e **FillingQuestionnaireView** mediante il *controllerG QuestionsController*;
- **Utilizzo:** viene utilizzato per consentire all’utente la compilazione dell’esercizio a riempimento di spazi vuoti;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo *modelviewG* la cui istanziazione è contenuta all’interno della variabile \$scope di *AngularG*. All’interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra le directive che compongono dinamicamente la vista della domanda e il *controllerG QuestionsController*;
 - **OUT TrainingView:** *viewG* principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell’allenamento;
 - **OUT FillingQuestionnaireView:** *viewG* principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell’applicazione.
- **Attributi:**
 - **+ questionText: String**
Identifica il testo della domanda;
 - **+ image: String**
Identifica l'url di una possibile immagine nella domanda;
 - **+ answers: Array<Object>**
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * **+ type: String**
Indica la tipologia della risposta;
 - * **+ text: String**
Contiene il testo dell’affermazione;
 - * **+ url: String**
Rappresenta l’immagine della risposta;
 - * **+ attributesForEmptySpaces: Mixed**
Contiene i seguenti attributi:



```

1. + wordNumber: Number
   Rappresenta la posizione dello spazio vuoto in cui deve andare inserita la
   parola.

- + controller: String
   Stringa contenente il nome del controllerG della direttiva;

- + restrict: String
   Stringa che permette di definire le modalità di inserimento della direttiva all'interno
   della pagina;

- + scope: Scope
   Oggetto $scope interno della direttiva, contiene le funzionalità per gestire i dati
   presenti all'interno;

- + templateUrl: String: stringa contenente il percorso del file HTMLG che contiene
   la direttiva.

```

2.4.2.20 QuizziPedia::Front-End::Directives::HeaderTextQuestionDirective

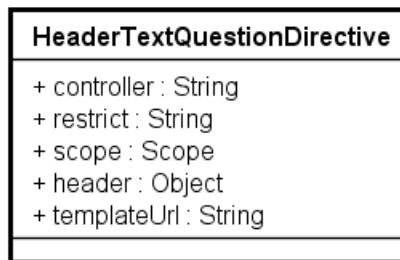


Figura 64: QuizziPedia::Front-End::Directives::HeaderTextQuestionDirective

- **Descrizione:** rappresenta il componente grafico che presenta all'utente l'argomento e le parole chiave della domanda che ha a schermo. Viene visualizzato dinamicamente all'interno delle views *TrainingView* e *FillingQuestionnaireView* mediante il *controller_G QuestionsController*;
- **Utilizzo:** viene utilizzato per consentire all'utente la visualizzazione dell'argomento della domanda e le parole chiave associate ad essa;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile di ambiente *\$scope* di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le directive che compongono dinamicamente la vista della domanda e il *controller_G QuestionsController*;
 - **OUT TrainingView:** *view_G* principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT FillingQuestionnaireView:** *view_G* principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**



- + **header:** Object
Oggetto contenente i campi dati da visualizzare nella direttiva, ovvero:
 - * **topic:** String
Argomento della domanda;
 - * **keywords:** Array<String>
Parole chiave della domanda.
- + **controller:** String
Stringa contenente il nome del $controller_G$ della direttiva;
- + **restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- + **scope:** Scope
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- + **templateUrl:** String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.21 QuizziPedia::Front-End::Directives::InfoQuestionnaireDirective

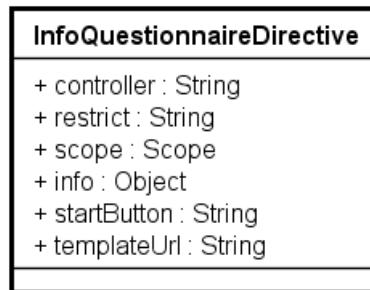


Figura 65: QuizziPedia::Front-End::Directives::InfoQuestionnaireDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare le informazioni principali del questionario che si sta per svolgere. Viene visualizzato dinamicamente all'interno della $view_G$ FillingQuestionnaireView mediante il $controller_G$ FillingQuestionsController;
- **Utilizzo:** viene utilizzato per consentire all'utente di visualizzare le informazioni principali del questionario che si sta per svolgere. Informazioni come:
 - Nome del questionario;
 - Nome dell'autore del questionario;
 - Data di creazione del questionario;
 - Argomento del questionario;
 - Bottone per iniziare il questionario.
- **Relazioni con altre classi:**
 - **OUT FillingQuestionnaireModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$.



All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G FillingQuestionnaireView* e il *controller_G FillingQuestionnaireController*;

- **IN FillingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a riempimento di spazi;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

• **Attributi:**

- **+ info: Object**

Oggetto contenente tutte le informazioni sul questionario, ovvero:

- * **name: String**
Nome del questionario;
- * **author: String**
Autore del questionario;
- * **date: Date**
Data di compilazione;
- * **topic: String**
Argomento della domanda;
- * **keywords: Array<String>**
Parole chiave della domanda.

- **+ startButton: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di inizio del questionario selezionato, in italiano o in inglese;

- **+ controller: String**

Stringa contenente il nome del *controller_G* della direttiva;

- **+ restrict: String**

Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;

- **+ scope: Scope**

Oggetto *\$scope* interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;

- **+ templateUrl: String**

Stringa contenente il percorso del file *HTML_G* che contiene la direttive.

2.4.2.22 QuizziPedia::Front-End::Directives::LinkingAnswerDirective

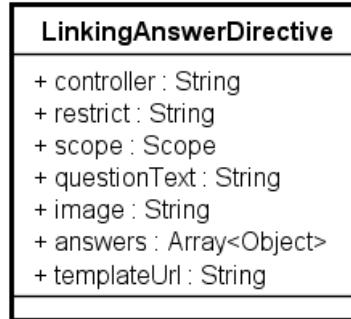


Figura 66: QuizziPedia::Front-End::Directives::LinkingAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda di collegamento. Viene visualizzato dinamicamente all'interno delle views `TrainingView` e `FillingQuestionnaireView` mediante il `controllerG QuestionsController`;
- **Utilizzo:** viene utilizzato per consentire all'utente la compilazione della domanda di collegamento;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo `modelviewG` la cui istanziazione è contenuta all'interno della variabile `$scope` di `AngularG`. All'interno di essa sono presenti le variabili e i metodi necessari per il `Two-Way Data-BindingG` tra le directive che compongono dinamicamente la vista della domanda e il `controllerG QuestionsController`;
 - **OUT TrainingView:** `viewG` principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT FillingQuestionnaireView:** `viewG` principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ questionText: String**
Identifica il testo della domanda;
 - **+ image: String**
Identifica l'url di una possibile immagine nella domanda;
 - **+ answers: Array<Object>**
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * **+ type: String**
Indica la tipologia della risposta;
 - * **+ text: String**
Contiene il testo dell'affermazione;
 - * **+ url: String**
Rappresenta l'immagine della risposta;
 - * **+ attributesForLinking: Mixed**: contiene i seguenti attributi:



```

1. + text1: String
Rappresenta il primo elemento testuale che deve essere collegato con il secondo
elemento testuale o rappresentato da un'immagine;

2. + text2: String
Rappresenta il secondo elemento testuale che deve essere collegato con il primo
elemento testuale o rappresentato da un'immagine;

3. + url1: String
Rappresenta il primo elemento rappresentato da un'immagine che deve essere
collegato con il secondo elemento testuale o rappresentato da un'immagine;

4. + url2: String
Rappresenta il secondo elemento rappresentato da un'immagine che deve
essere collegato con il primo elemento testuale o rappresentato da
un'immagine.

- + controller: String
Stringa contenente il nome del controllerG della direttiva;

- + restrict: String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno
della pagina;

- + scope: Scope
Oggetto $scope interno della direttiva, contiene le funzionalità per gestire i dati
presenti all'interno;

- + templateUrl: String
Stringa contenente il percorso del file HTMLG che contiene la direttive.

```

2.4.2.23 QuizziPedia::Front-End::Directives::MultipleChoiceAnswerDirective

MultipleChoiceAnswerDirective
+ controller : String + restrict : String + scope : Scope + questionText : String + image : String + answers : Array<Object> + templateUrl : String

Figura 67: QuizziPedia::Front-End::Directives::MultipleChoiceAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda a risposta multipla. Viene visualizzato dinamicamente all'interno delle views **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;
- **Utilizzo:** viene utilizzato per consentire all'utente la compilazione della domanda a risposta multipla;
- **Relazioni con altre classi:**



- **IN QuestionsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le directive che compongono dinamicamente la vista della domanda e il $controller_G$ **QuestionsController**;
- **OUT TrainingView:** $view_G$ principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
- **OUT FillingQuestionnaireView:** $view_G$ principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **questionText: String**
Identifica il testo della domanda;
- **image: String**
Identifica l'url di una possibile immagine nella domanda;
- **answers: Array<Object>**
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * **type: String**
Indica la tipologia della risposta;
 - * **text: String**
Contiene il testo dell'affermazione;
 - * **url: String**
Rappresenta l'immagine della risposta;
 - * **attributesForTForMultiple: Mixed**
Contiene i seguenti attributi:
 1. **isItRight: Boolean**
Contiene se la risposta è vera o falsa.
- **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope: Scope**
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.24 QuizziPedia::Front-End::Directives::SortImagesAnswerDirective

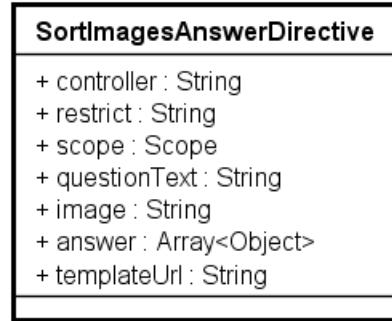


Figura 68: QuizziPedia::Front-End::Directives::SortImagesAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda ad ordinamento di immagini. Viene visualizzato dinamicamente all'interno delle views **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;
- **Utilizzo:** viene utilizzato per consentire all'utente la compilazione della domanda ad ordinamento di immagini;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo *modelview_G* la cui istanziazione è contenuta all'interno della variabile \$scope di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le directive che compongono dinamicamente la vista della domanda e il *controller_G QuestionsController*;
 - **OUT TrainingView:** *view_G* principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT FillingQuestionnaireView:** *view_G* principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ questionText: String**
Identifica il testo della domanda;
 - **+ image: String**
Identifica l'url di una possibile immagine nella domanda;
 - **+ answers: Array<Object>**
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * **+ type: String**
Indica la tipologia della risposta;
 - * **+ text: String**
Contiene il testo dell'affermazione;
 - * **+ url: String**
Rappresenta l'immagine della risposta;
 - * **+ attributesForSorting: Mixed**
Contiene i seguenti attributi:



- 1. + position: Boolean
Contiene la giusta posizione del testo o dell'immagine nell'esercizio di ordinamento.
- + controller: String
Stringa contenente il nome del $controller_G$ della direttiva;
- + restrict: String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- + scope: Scope
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- + templateUrl: String
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.25 QuizziPedia::Front-End::Directives::SortTextAnswerDirective

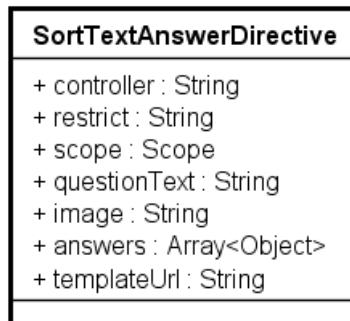


Figura 69: QuizziPedia::Front-End::Directives::SortTextAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda ad ordinamento di stringhe. Viene visualizzato dinamicamente all'interno delle views TrainingView e FillingQuestionnaireView mediante il $controller_G$ QuestionsController;
- **Utilizzo:** viene utilizzato per consentire all'utente la compilazione della domanda ad ordinamento di stringhe;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le directive che compongono dinamicamente la vista della domanda e il $controller_G$ QuestionsController;
 - **OUT TrainingView:** $view_G$ principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT FillingQuestionnaireView:** $view_G$ principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.



- **Attributi:**

- + **questionText:** String
Identifica il testo della domanda;
- + **image:** String
Identifica l'url di una possibile immagine nella domanda;
- + **answers:** Array<Object>
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * + **type:** String
Indica la tipologia della risposta;
 - * + **text:** String
Contiene il testo dell'affermazione;
 - * + **url:** String
Rappresenta l'immagine della risposta;
 - * + **attributesForSorting:** Mixed
Contiene i seguenti attributi:
 1. + **position:** Boolean
Contiene la giusta posizione del testo o dell'immagine nell'esercizio di ordinamento.
- + **controller:** String
Stringa contenente il nome del *controller_G* della direttiva;
- + **restrict:** String
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- + **scope:** Scope
Oggetto *\$scope* interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- + **templateUrl:** String
Stringa contenente il percorso del file *HTML_G* che contiene la direttive.

2.4.2.26 QuizziPedia::Front-End::Directives::TrainingSetUpDirective

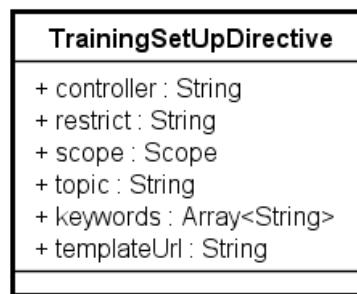


Figura 70: QuizziPedia::Front-End::Directives::TrainingSetUpDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di selezionare l'argomento e le parole chiave per iniziare un allenamento con queste caratteristiche. Viene visualizzato dinamicamente all'interno della *view_G* TrainingView mediante il *controller_G* TrainingController;



- **Utilizzo:** viene utilizzato per consentire all'utente di selezionare l'argomento e le parole chiave di un allenamento;
- **Relazioni con altre classi:**

- **IN TrainingModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **TrainingView** e il $controller_G$ **TrainingController**;
- **OUT TrainingView:** $view_G$ principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ topic: String**
Valore ottenuto come risultato dal ciclo effettuato sull'array degli argomenti;
- **+ keywords: Array<String>**
Valore ottenuto come risultato dal ciclo effettuato sull'array delle keywords;
- **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
- **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
- **+ scope: Scope**
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
- **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.27 QuizziPedia::Front-End::Directives::TrueFalseAnswerDirective

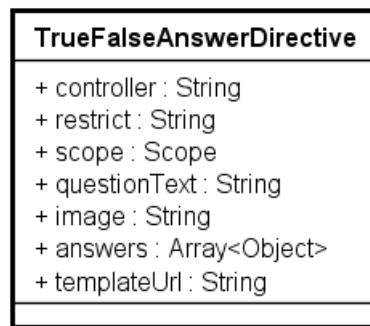


Figura 71: QuizziPedia::Front-End::Directives::TrueFalseAnswerDirective

- **Descrizione:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda vero e falso. Viene visualizzato dinamicamente all'interno delle views **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;



- **Utilizzo:** viene utilizzato per consentire all'utente la compilazione della domanda vero e falso;
- **Relazioni con altre classi:**
 - **IN QuestionsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le directive che compongono dinamicamente la vista della domanda e il $controller_G$ **QuestionsController**;
 - **OUT TrainingView:** $view_G$ principale della modalità allenamento. Conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT FillingQuestionnaireView:** $view_G$ principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ questionText: String**
Identifica il testo della domanda;
 - **+ image: String**
Identifica l'url di una possibile immagine nella domanda;
 - **+ answers: Array<Object>**
Array che contiene coppie di valori. Queste coppie sono formate da:
 - * **+ type: String**
Indica la tipologia della risposta;
 - * **+ text: String**
Contiene il testo dell'affermazione;
 - * **+ url: String**
Rappresenta l'immagine della risposta;
 - * **+ attributesForTForMultiple: Mixed**
Contiene i seguenti attributi:
 1. **+ isItRight: Boolean**
Contiene se la risposta è vera o falsa.
 - **+ controller: String**
Stringa contenente il nome del $controller_G$ della direttiva;
 - **+ restrict: String**
Stringa che permette di definire le modalità di inserimento della direttiva all'interno della pagina;
 - **+ scope : Scope**
Oggetto $\$scope$ interno della direttiva, contiene le funzionalità per gestire i dati presenti all'interno;
 - **+ templateUrl: String**
Stringa contenente il percorso del file $HTML_G$ che contiene la direttive.

2.4.2.28 QuizziPedia::Front-End::Directives::LoginBarDirective

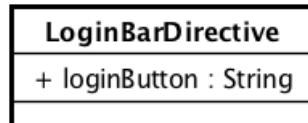


Figura 72: QuizziPedia::Front-End::Directives:LoginBarDirective

- **Descrizione:** directive contenente il componente che permette di effettuare il redirect alla pagina di login;
- **Utilizzo:** permette di effettuare il redirect alla pagina di login;
- **Relazioni con altre classi:**
 - **IN** `MenuBarModelView`: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `Index` e il `controllerGMenuBarController`;
 - **OUT** `MenuBarDirective`: rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello $\$scope$ isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.
- **Attributi:**
 - + `loginButton: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link all'autenticazione, in italiano o inglese.

2.4.2.29 QuizziPedia::Front-End::Directives::SignUpBarDirective

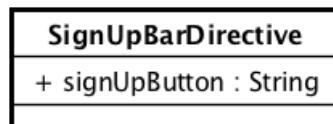


Figura 73: QuizziPedia::Front-End::Directives:SignUpBarDirective

- **Descrizione:** directive contenente il componente che permette di effettuare il redirect alla pagina di registrazione;
- **Utilizzo:** permette di effettuare il redirect alla pagina di registrazione;
- **Relazioni con altre classi:**
 - **IN** `MenuBarModelView`: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `Index` e il `controllerGMenuBarController`;



- **OUT MenuBarDirective:** rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello \$scope isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.

- **Attributi:**

- **+ signUpButton: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link alla registrazione, in italiano o inglese.

2.4.2.30 QuizziPedia::Front-End::Directives::UserBarDirective

UserBarDirective
+ profileViewButton : String

Figura 74: QuizziPedia::Front-End::Directives::UserBarDirective

- **Descrizione:** directive contenente il componente che permette di effettuare il redirect alla pagina di visualizzazione profilo;

- **Utilizzo:** permette di effettuare il redirect alla pagina di visualizzazione profilo;

- **Relazioni con altre classi:**

- **IN MenuBarModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ Index e il $controller_G$ MenuBarController;
- **OUT MenuBarDirective:** rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello \$scope isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.

- **Attributi:**

- **+ profileViewButton: String**

Attributo che viene utilizzato per visualizzare il nome dell'utente.

2.4.2.31 QuizziPedia::Front-End::Directives::ProfileManagementBarDirective

ProfileManagementBarDirective
+ profileManagementButton : String

Figura 75: QuizziPedia::Front-End::Directives::ProfileManagementBarDirective



- **Descrizione:** directive contenente il componente che permette di effettuare il redirect alla pagina di gestione del profilo;
- **Utilizzo:** permette di effettuare il redirect alla pagina di gestione del profilo;
- **Relazioni con altre classi:**

- **IN** `MenuBarModelView`: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `Index` e il `controllerGMenuBarController`;
- **OUT** `MenuBarDirective`: rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello $\$scope$ isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.

- **Attributi:**

- `+ profileManagementButton: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link alla gestione del profilo utente, in italiano o inglese.

2.4.2.32 QuizziPedia::Front-End::Directives::QuestionsManagementBarDirective

QuestionsManagementBarDirective
<code>+ questionsButton : String</code>

Figura 76: QuizziPedia::Front-End::Directives:QuestionsManagementBarDirective

- **Descrizione:** directive contenente il componente che permette di effettuare il redirect alla pagina di gestione delle domande;
- **Utilizzo:** permette di effettuare il redirect alla pagina di gestione delle domande;
- **Relazioni con altre classi:**

- **IN** `MenuBarModelView`: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `Index` e il `controllerGMenuBarController`;
- **OUT** `MenuBarDirective`: rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello $\$scope$ isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.

- **Attributi:**

- `+ questionsButton: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link alla gestione delle domande, in italiano o in inglese.



2.4.2.33 QuizziPedia::Front-End::Directives::LogoutBarDirective



Figura 77: QuizziPedia::Front-End::Views:LogoutBarDirective

- **Descrizione:** directive contenente il componente che permette di effettuare il logout dal sistema;
- **Utilizzo:** permette di effettuare il logout dal sistema;
- **Relazioni con altre classi:**
 - **IN** `MenuBarModelView`: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `Index` e il `controllerG MenuBarController`;
 - **OUT** `MenuBarDirective`: rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello $\$scope$ isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.
- **Attributi:**
 - + `logoutButton: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone per eseguire il logout dal sistema.

2.4.2.34 QuizziPedia::Front-End::Directives::QuestionnaireManagementBarDirective



Figura 78: QuizziPedia::Front-End::Directives:QuestionnaireManagementBarDirective

- **Descrizione:** directive contenente il componente che permette di effettuare il redirect alla pagina di gestione dei questionari;
- **Utilizzo:** permette di effettuare il redirect alla pagina di gestione dei questionari;
- **Relazioni con altre classi:**
 - **IN** `MenuBarModelView`: classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `Index` e il `controllerG MenuBarController`;



- **OUT MenuBarDirective:** rappresenta il menù, presente in ogni pagina dell'applicazione, generato in base agli oggetti passati nello *\$scope* isolato. Fornisce un pulsante per ogni oggetto ricevuto come parametro, ogni pulsante viene rappresentato con un'icona e con un testo. Al click di un pulsante viene invocata la funzione ad esso associata.

- **Attributi:**

- **+ questionnaireButton: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link alla gestione dei questionari, in italiano o inglese.



2.5 QuizziPedia::Front-End::Services

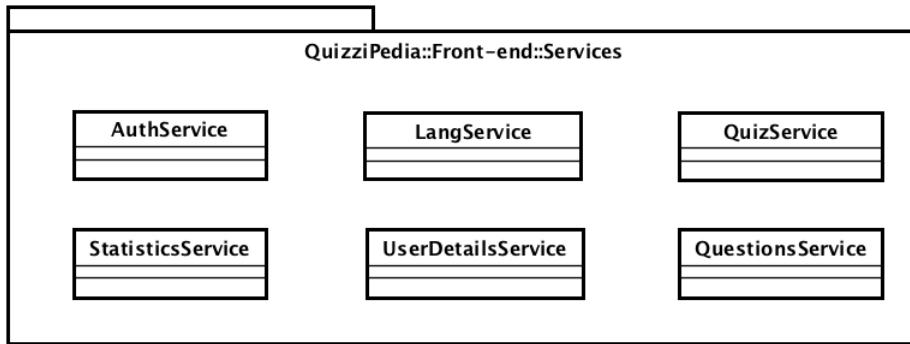


Figura 79: QuizziPedia::Front-End::Services

2.5.1 Informazioni generali

- **Descrizione:** package che contiene le classi individuate che permettono la comunicazione del lato front-end con il lato back-end;
- **Padre:** Front-End;
- **Interazione con altri componenti:**
 - **Models:** package che contiene le classi $model_G$ individuate;
 - **Controllers:** package che contiene le classi $controller_G$ individuate.

2.5.2 Classi

2.5.2.1 QuizziPedia::Front-End::Services::AuthService

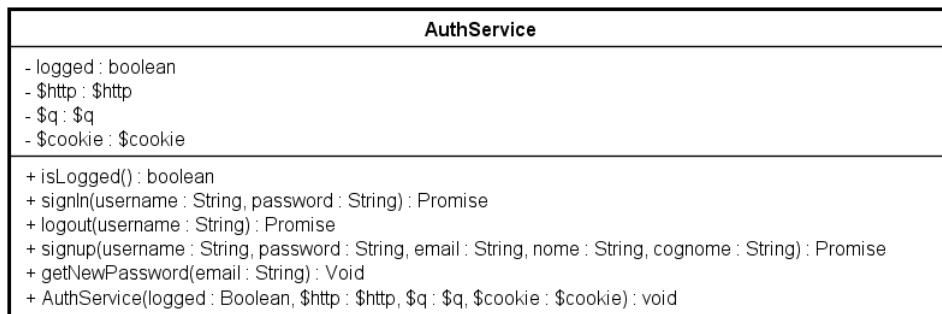


Figura 80: QuizziPedia::Front-End::Services::AuthService

- **Descrizione:** questa classe permette di gestire la registrazione e l'autenticazione di un utente;
- **Utilizzo:** fornisce le funzionalità di registrazione e autenticazione ai $controllers_G$. Controlla che i dati inseriti dall'utente siano presenti nel $Database_G$ in caso di autenticazione. Presenta anche le funzionalità per la gestione del reset della password;
- **Relazione con altre classi:**



- **OUT LoginController:** questa classe gestisce la logica alla base della pagina di autenticazione;
- **OUT PasswordForgotController:** questa classe gestisce la logica alla base del reset della password;
- **OUT SignUpController:** questa classe gestisce la logica alla base della registrazione di un nuovo utente.

- **Attributi:**

- **- logged: Boolean**
Campo dati che indica se l'utente è autenticato;
- **- \$http: \$http**
Campo dati che contiene un riferimento al servizio \$http che permette la comunicazione con il protocollo $HTTP_G$;
- **- \$q: \$q**
Campo dati che contiene un riferimento a \$q, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone;
- **- \$cookie: \$cookie**
Campo dati che consente di salvare il valore di logged ed evitare che venga richiesta l'autenticazione ad ogni ricaricamento della pagina.

- **Metodi:**

- **+ AuthService(logged: Boolean, \$http: \$http, \$q: \$q, \$cookie: \$cookie)**

Metodo costruttore della classe;

Parametri:

- * **logged: Boolean**
Parametro che indica se l'utente è loggato o no;
- * **\$http: \$http**
Parametro contenente un riferimento al servizio \$http creato da $Angular_G$ per facilitare la comunicazione mediante protocollo $HTTP_G$;
- * **\$q: \$q**
Parametro contenente un riferimento al servizio \$q creato da $Angular_G$ per facilitare la gestione di funzione asincrone mediante l'utilizzo delle $Promise_G$;
- * **\$cookie: \$cookie**
Parametro che consente di salvare il valore di logged ed evitare che venga richiesta l'autenticazione ad ogni ricaricamento della pagina.

- **+ isLoggedIn(): Boolean**

Metodo che restituisce il valore dell'attributo *logged* salvato all'interno del cookie che permette di evitare di eseguire l'autenticazione ad ogni caricamento della pagina;

- **+ signIn(email: String, password: String): Promise**

Metodo che permette di effettuare il login all'applicazione facendo una richiesta di autenticazione al back-end passando i parametri ricevuti. Il metodo ritorna una $Promise_G$. In caso la $Promise_G$ venga rifiutata, verrà restituito al **LoginController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la $Promise_G$ venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;

Parametri:

- * **username: String**

Parametro che rappresenta lo username o la email dell'utente;



```
* password: String
    Parametro che rappresenta la password dell'utente.

- + logout(username: String): Promise
    Metodo che permette di effettuare il logout dall'applicazione. Il metodo ritorna una PromiseG. In caso la PromiseG venga rifiutata, verrà restituito al LogoutController un oggetto ErrorModelInfo contenente tutti i dettagli dell'errore. In caso la PromiseG venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata.

Parametri:
* username: String
    Parametro che rappresenta l'utente che vuole eseguire il logout.

- + signup(username: String, password: String, email: String,
    nome: String, cognome: String): Promise
    Metodo che permette di effettuare la registrazione all'applicazione tramite richiesta di creazione nuovo account al back-end. Il metodo ritorna una PromiseG. In caso la PromiseG venga rifiutata, verrà restituito al SignUpController un oggetto ErrorModelInfo contenente tutti i dettagli dell'errore. In caso la PromiseG venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;

Parametri:
* username: String
    Parametro che rappresenta lo username o la email dell'utente;
* password: String
    Parametro che rappresenta la password dell'utente;
* email: String
    Parametro che rappresenta la email dell'utente;
* nome: String
    Parametro che rappresenta il nome dell'utente;
* cognome: String
    Parametro che rappresenta il cognome dell'utente.

- + getNewPassword(email: String): Void
    Metodo che permette il recupero della password. Il metodo ritorna una PromiseG. In caso la PromiseG venga rifiutata, verrà restituito al PasswordForgotController un oggetto ErrorModelInfo contenente tutti i dettagli dell'errore. In caso la PromiseG venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata.
```

2.5.2.2 QuizziPedia::Front-End::Services::LangService

LangService
- \$http : \$http
+ LangService(\$http : \$http) + getKeywords(lang : String) : Object

Figura 81: QuizziPedia::Front-End::Services::LangService

- **Descrizione:** questa classe permette di gestire la lingua nella quale si è scelto di utilizzare l'applicazione;



- **Utilizzo:** fornisce delle funzionalità per recuperare la giusta traduzione delle pagine;
- **Relazione con altre classi:**
 - **IN AppRun:** classe che permette l'avvio dell'applicazione.
- **Attributi:**
 - - **\$http: \$http**
Campo dati che contiene un riferimento al servizio \$http che permette la comunicazione con il protocollo $HTTP_G$.
- **Metodi:**
 - + **LangService(\$http: \$http)**
Metodo costruttore della classe;
Parametri:
 - * **\$http: \$http**
Campo dati contenente un riferimento al servizio \$http creato da $Angular_G$ per facilitare la comunicazione mediante protocollo $HTTP_G$.
 - + **getKeywords(lang: String): Object**
Metodo che ritorna la lista di tutte le $keywords_G$ nella lingua richiesta.
Parametri:
 - * **lang: String**
Parametro che indica la lingua delle $keywords_G$ da restituire.

2.5.2.3 QuizziPedia::Front-End::Services::QuestionsService

QuestionsService
- \$http : \$http - \$q : \$q
+ QuestionService(\$http : \$http, \$q : \$q) + sendQuestion(question : QuestionItemModel) : Promise + getUserQuestions(username : String) : Promise + getKeywords(key : String) : Array<String> + getQuestion(idQuestion : String) : Promise + getNextQuestion() : Promise

Figura 82: QuizziPedia::Front-End::Services::QuestionService

- **Descrizione:** questa classe permette di ottenere domande esistenti e salvare nuove domande;
- **Utilizzo:** utilizzata per richiedere domande presenti nel database. Offre inoltre delle funzionalità per inserire nuove domande;
- **Relazione con altre classi:**
 - **IN QuestionItemModel:** rappresenta una domanda. Contiene tutte le informazioni necessarie alla presentazione del contenuto della domanda;
 - **IN TrueFalseQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda vero/falso;
 - **IN MultiplyQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a risposta multipla;



- **IN ConnectionQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a collegamento;
- **IN ImagesSortingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento immagini;
- **IN StringsSortingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento di stringhe;
- **IN FillingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a riempimento di spazi;
- **IN ClickableAreaQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda ad area cliccabile;
- **IN EditorQMLController:** questa classe permette di gestire la creazione e la modifica di domande create tramite editor QML;
- **IN QuestionsManagementController:** questa classe permette di gestire e di ottenere le domande create dall'utente;
- **IN TopicKeywordsController:** questa classe permette di gestire il recupero delle parole chiave di un questionario;
- **IN QuestionnaireQuestionsManagementController:** questa classe permette di gestire il recupero delle domande per il questionario;
- **IN QuestionsController:** questa classe permette di gestire il recupero delle domande per poterle stampare nella modalità allenamento;
- **IN QuestionsItemModel:** rappresenta una domanda. Contiene tutte le informazioni necessarie alla presentazione del contenuto della domanda.

- **Attributi:**

- **\$http: \$http**
Campo dati che contiene un riferimento al servizio \$http che permette la comunicazione con il protocollo $HTTP_G$;
- **\$q: \$q**
Campo dati che contiene un riferimento a \$q, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone.

- **Metodi:**

- **+ QuestionService(\$http: \$http, \$q: \$q)**
Metodo costruttore della classe;
Parametri:
 - * **\$http: \$http**
Campo dati che contiene un riferimento al servizio \$http che permette la comunicazione con il protocollo $HTTP_G$;
 - * **\$q: \$q**
Campo dati che contiene un riferimento a \$q, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone.
- **+ sendQuestion(question: QuestionItemModel): Promise**
Metodo che serve per mandare una domanda al back-end in modo che venga salvata; In caso la $Promise_G$ venga rifiutata, verrà restituito al $controller_G$ un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la $Promise_G$ venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;
Parametri:



- * **question:** QuestionItemModel
Identifica tutti i dettagli di una domanda.
- + **getUserQuestions(username: String): Promise**
Metodo che ritorna tutte le domande create dall'utente. Il metodo ritorna una *Promise_G*. In caso la *Promise_G* venga rifiutata, verrà restituito al **QuestionsManagementController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la *Promise_G* venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;
Parametri:
 - * **username: String**
Parametro che indica l'utente del quale andranno caricate tutte le domande da lui create.
- + **getKeywords(key: String): Array<String>**
Metodo che permette di ottenere le keywords a partire da una stringa passata;
Parametri:
 - * **key: String**
Parametro che identifica la stringa con la quale cercare le keywords.
- + **getQuestion(idQuestion: String): Promise**
Metodo che ritorna la domanda associata all'id passato come parametro. Il metodo ritorna una *Promise_G*. In caso la *Promise_G* venga rifiutata, verrà restituito al **QuestionsManagementController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la *Promise_G* venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;
Parametri:
 - * **idQuestion: String**
Parametro che indica l'id della domanda da recuperare e restituire.
- + **getNextQuestion(): Promise**
Metodo che ritorna la domanda successiva nella modalità allenamento. Il metodo ritorna una *Promise_G*. In caso la *Promise_G* venga rifiutata, verrà restituito al **QuestionsController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la *Promise_G* venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata.

2.5.2.4 QuizziPedia::Front-End::Services::QuizService



QuizService
<ul style="list-style-type: none"> - \$http : \$http - \$q : \$q
<ul style="list-style-type: none"> + QuizService(\$http : \$http, \$q : \$q) + search(quizId : String) : Promise + subscribeQuestionnaire(username : String) : void + getQuestionnaireDetails(username : String) : Promise + getDoneQuestionnaire(username : String) : Promise + createQuestionnaire(title : String, questions : QuestionnaireItemModel) : Promise + getUserQuestionnaire(username : String) : Array<QuestionnaireModel> + getSubscribedQuestionnaire(username : String) : Array<QuestionnaireModel> + getQuiz(quizId : String, username : String) : Promise + approveSubscribeQuestionnaire(username : String) : void + getUserForThisQuestionnaire(quizId : String) : Array<UserDetailsModel> + getQuizResult(quizId : String) : Object

Figura 83: QuizziPedia::Front-End::Services::QuizService

- **Descrizione:** questa classe permette di ottenere i dati di un quiz tramite delle parole chiave inserite dall'utente nella barra di ricerca. Permette inoltre di iscriversi ad un questionario e di scaricare l'intera lista di domande di un questionario a partire dal suo id univoco;
- **Utilizzo:** fornisce le funzionalità per ottenere i dati di un quiz in seguito ad una ricerca dell'utente, passando i risultati ai *controllers_G*. Ritorna dei riferimenti alle domande del quiz;
- **Relazione con altre classi:**
 - **IN QuestionnaireModel:** rappresenta un questionario. Contiene tutte le informazioni necessarie alla presentazione del contenuto del questionario;
 - **OUT SearchController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione;
 - **OUT QuestionsController:** questa classe permette di gestire il recupero delle domande per poterle stampare nella modalità allenamento;
 - **OUT QuestionnaireDetailsController:** questa classe permette di gestire i dettagli di un questionario;
 - **OUT FillingQuestionnaireController:** questa classe permette di gestire la compilazione del questionario;
 - **OUT CreateQuestionnaireController:** questa classe permette di gestire la creazione di un questionario;
 - **OUT RegistrationManagementController:** questa classe permette di gestire le iscrizione degli utenti ai questionari;
 - **OUT ResultsController:** questa classe permette di gestire le iscrizione degli utenti ai questionari;
 - **OUT QuestionnaireManagementController:** questa classe permette di gestire tutti i questionari creati da un utente.
- **Attributi:**



- - `$http: $http`
Campo dati che contiene un riferimento al servizio `$http` che permette la comunicazione con il protocollo $HTTP_G$;
- - `$q: $q`
Campo dati che contiene un riferimento a `$q`, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone.

- **Metodi:**

- + `QuizService($http: $http, $q: $q)`
Metodo costruttore della classe;
Parametri:
 - * `$http: $http`
Campo dati che contiene un riferimento al servizio `$http` che permette la comunicazione con il protocollo $HTTP_G$;
 - * `$q: $q`
Campo dati che contiene un riferimento a `$q`, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone.
- + `search(quizId: String): Promise`
Metodo che serve per recuperare i questionari dopo averne selezionato uno dalla lista ottenuta da una ricerca. Il metodo ritorna una $Promise_G$. In caso la $Promise_G$ venga rifiutata, verrà restituito al `SearchController` un oggetto `ErrorModelInfo` contenente tutti i dettagli dell'errore. In caso la $Promise_G$ venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;
Parametri:
 - * `quizId: String`
Parametro che rappresenta il questionario da cercare.
- + `subscribeQuestionnaire(quizId: String, username: String): Void`
Metodo per iscriversi ad un questionario;
 - * `quizId: String`
Parametro che rappresenta il questionario in cui iscrivere l'utente;
 - * `username: String`
Parametro che rappresenta l'utente da registrare al questionario.
- + `getQuestionnaireDetails(username: String): Promise`
Metodo che serve per ritornare i dettagli di tutti i questionari creati da un utente; Il metodo ritorna una $Promise_G$. In caso la $Promise_G$ venga rifiutata, verrà restituito al `StatisticsController` un oggetto `ErrorModelInfo` contenente tutti i dettagli dell'errore. In caso la $Promise_G$ venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;
Parametri:
 - * `username: String`
Parametro che rappresenta l'utente del quale andranno caricati tutti i questionari.
- + `getDoneQuestionnaire(username: String): Promise`
Metodo che restituisce tutti i questionari svolti da un utente. Il metodo ritorna una $Promise_G$. In caso la $Promise_G$ venga rifiutata, verrà restituito al `UserDetailsController` un oggetto `ErrorModelInfo` contenente tutti i dettagli dell'errore;
Parametri:



- * **username:** String
Parametro che rappresenta l'utente del quale andranno caricati tutti i questionari svolti.
- + **createQuestionnaire(title: String, questions: QuestionnaireItemModel): Promise**
Metodo che permette di creare un nuovo questionario. Il metodo ritorna una Promise_G . In caso la Promise_G venga rifiutata, verrà restituito al **CreateQuestionnaireController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la Promise_G venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;
Parametri:
 - * **title:** String
Parametro che rappresenta il titolo del questionario.
 - * **questions:** QuestionnaireItemModel
Parametro contenente tutte le domande del questionario.
- + **getUserQuestionnaire(username: String): Array<QuestionnaireModel>**
Metodo che ritorna un array di QuestionnaireModel che sono tutti i questionari creati dall'utente in input;
Parametri:
 - * **username:** String
Parametro che indica l'identificativo dell'utente del quale vogliamo richiedere i questionari.
- + **getSubscribedQuestionnaire(username: String): Array<QuestionnaireModel>**
Metodo che ritorna la lista dei questionari a cui l'utente è iscritto;
Parametri:
 - * **username:** String
Parametro che indica l'utente del quale scaricare i quiz a cui risulta iscritto.
- + **approveSubscribeQuestionnaire(username: String): void**
Metodo che approva l'iscrizione di un determinato utente ad un quesitonario;
Parametri:
 - * **username:** String
Parametro che indica l'utente da abilitare ad un questionario.
- **getUserForThisQuestionnaire(quizId: String): Array**
Metodo che ritorna tutti gli utenti che hanno eseguito il questionario;
Parametri:
 - * **quizId:** String
Paramentro che indica il questionario del quale scaricare gli utenti.
- + **getQuizResults(quizId: String): Object**
Metodo che ritorna i risultati di un questionario;
Parametri:
 - * **quiId:** String
Id del questionario del quale recuperare i risultati.
- + **getQuiz(quizId: String, username: String): Promise**
Metodo che ritorna una Promise_G . In caso la Promise_G venga rifiutata, verrà restituito al **CreateQuestionnaireController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la Promise_G venga accettata, cioè se l'utente è registrato, verrà restituito al chiamante il questionario.
Parametri:



* `quiId: String`
 Id del questionario del quale recuperare i risultati.

2.5.2.5 QuizziPedia::Front-End::Services::SearchService

SearchService
<ul style="list-style-type: none"> - <code>\$http : \$http</code> - <code>\$q : \$q</code>
<ul style="list-style-type: none"> + <code>SearchService(\$http : \$http, \$q : \$q)</code> + <code>searchUsers(keyword : String) : Promise</code> + <code>searchQuestionnaire(keyword : String) : Promise</code>

Figura 84: QuizziPedia::Front-End::Services::SearchService

- **Descrizione:** questa classe permette di gestire il recupero dei dati dal back-end a seguito di una ricerca effettuata da un utente;
- **Utilizzo:** fornisce le funzionalità per recuperare dal back-end delle informazioni basandosi sulle *keyword_G* inserite dall'utente;
- **Relazione con altre classi:**
 - **OUT SearchController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione.
- **Attributi:**
 - `- $http: $http`
 Campo dati che contiene un riferimento al servizio `$http` che permette la comunicazione con il protocollo *HTTP_G*;
 - `- $q: $q`
 Campo dati che contiene un riferimento a `$q`, un servizio offerto da *Angular_G* per la gestione, tramite *Promise_G*, di chiamate asincrone.
- **Metodi:**
 - `+ SearchService($http: $http, $q: $q)`
 Metodo costruttore della classe;
Parametri:
 - * `$http: $http`
 Campo dati contenente un riferimento al servizio `$http` creato da *Angular_G* per facilitare la comunicazione mediante protocollo *HTTP_G*;
 - * `$q: $q`
 Campo dati contenente un riferimento al servizio `$q` creato da *Angular_G* per facilitare la gestione di funzione asincrone mediante l'utilizzo delle *Promise_G*.
 - `+ searchUsers(keyword: String): Promise`
 Metodo che serve per recuperare la lista di utenti dopo una ricerca. Il metodo ritorna una *Promise_G*. In caso la *Promise_G* venga rifiutata, verrà restituito al **SearchController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la *Promise_G* venga accettata, verrà restituito al chiamante del metodo il



risultato della chiamata;

Parametri:

* keyword: String

Parametro che rappresenta la parola da cercare.

- + searchQuestionnaire(keyword: String): Promise

Metodo che serve per recuperare la lista dei questionari dopo una ricerca. Il metodo ritorna una Promise_G . In caso la Promise_G venga rifiutata, verrà restituito al **SearchController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la Promise_G venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata.

Parametri:

* keyword: String

Parametro che rappresenta la parola da cercare.

2.5.2.6 QuizziPedia::Front-End::Services::StatisticsService

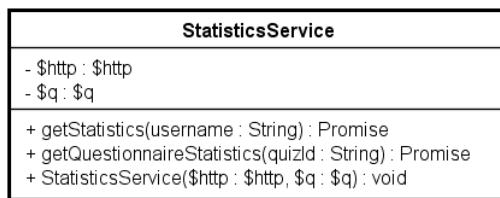


Figura 85: QuizziPedia::Front-End::Services::StatisticsService

- **Descrizione:** questa classe permette di ottenere le statistiche dell'utente;

- **Utilizzo:** fornisce al $controller_G$ le statistiche salvate;

- **Relazione con altre classi:**

– IN **StatisticsController**: questa classe permette di le statistiche di un utente.

- **Attributi:**

– - \$http: \$http

Campo dati che contiene un riferimento al servizio \$http che permette la comunicazione con il protocollo $HTTP_G$;

– - \$q: \$q

Campo dati che contiene un riferimento a \$q, un servizio offerto da $Angular_G$ per la gestione, tramite Promise_G , di chiamate asincrone.

- **Metodi:**

– + StatisticsService(\$http: \$http, \$q: \$q)

Metodo costruttore della classe;

Parametri:

* \$http: \$http

Campo dati che contiene un riferimento al servizio \$http che permette la comunicazione con il protocollo $HTTP_G$;



* \$q: \$q

Campo dati che contiene un riferimento a \$q, un servizio offerto da *AngularG* per la gestione, tramite *PromiseG*, di chiamate asincrone.

- + getStatistics(username: String): Promise

Metodo che serve per recuperare le statistiche di un utente tramite chiamata al backend. Il metodo ritorna una *PromiseG*. In caso la *PromiseG* venga rifiutata, verrà restituito al **StatisticsController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la *PromiseG* venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;

Parametri:

* username: String

Parametro che indica l'utente del quale andranno caricate tutte le statistiche.

- + getQuestionnaireStatistics(quizId: String): Promise

Metodo che restituisce le statistiche di un questionario. Il metodo ritorna una *PromiseG*. In caso la *PromiseG* venga rifiutata, verrà restituito al **StatisticsController** un oggetto **ErrorModelInfo** contenente tutti i dettagli dell'errore. In caso la *PromiseG* venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata.

Parametri:

* quizId: String

Parametro che indica il questionario del quale verranno caricate le statistiche.

2.5.2.7 QuizziPedia::Front-End::Services::UserDetailsService

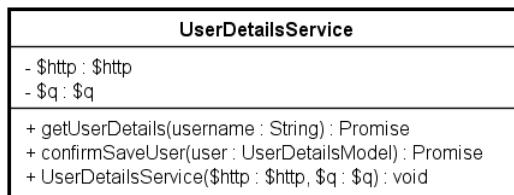


Figura 86: QuizziPedia::Front-End::Services::UserDetailsService

- **Descrizione:** questa classe permette di ottenere i dati personali degli utenti;
- **Utilizzo:** utilizzata per ottenere i dati personali di un utente. Permette inoltre di trovare i dati di utenti ricercati tramite l'apposita barra di ricerca;
- **Relazione con altre classi:**
 - **IN UserDetailsModel:** rappresenta un utente. Contiene tutte le informazioni necessarie alla presentazione del contenuto di un utente sia nella visualizzazione che nella gestione di un profilo;
 - **OUT SearchController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione;
 - **OUT UserDetailsController:** questa classe permette di gestire i dati di un utente;
 - **OUT ProfileManagementController:** questa classe permette di gestire il profilo personale di un utente.
- **Attributi:**



- - `$http: $http`
Campo dati che contiene un riferimento al servizio `$http` che permette la comunicazione con il protocollo $HTTP_G$;
- - `$q: $q`
Campo dati che contiene un riferimento a `$q`, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone.

- **Metodi:**

- + `UserDetailsService($http: $http, $q: $q)`
Metodo costruttore della classe;

Parametri:

- * `$http: $http`
Campo dati che contiene un riferimento al servizio `$http` che permette la comunicazione con il protocollo $HTTP_G$;
- * `$q: $q`
Campo dati che contiene un riferimento a `$q`, un servizio offerto da $Angular_G$ per la gestione, tramite $Promise_G$, di chiamate asincrone.

- + `getUserDetails(username: String): Promise`

Metodo che serve per ottenere i dettagli di un utente. Il metodo ritorna una $Promise_G$. In caso la $Promise_G$ venga rifiutata, verrà restituito al `SearchController` un oggetto `ErrorModelInfo` contenente tutti i dettagli dell'errore. In caso la $Promise_G$ venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata;

Parametri:

- * `username: String`
Parametro che indica l'utente del quale andranno caricati tutti i dati personali.

- + `confirmSaveUser(user: Object, imageObject: Object): Promise`

Metodo che serve per inviare al back-end una richiesta di salvataggio persistente dei dati. Viene invocato da `Confirm` in `ProfileManagementController`. Il metodo ritorna una $Promise_G$. In caso la $Promise_G$ venga rifiutata, verrà restituito al `ProfileManagementController` un oggetto `ErrorModelInfo` contenente tutti i dettagli dell'errore. In caso la $Promise_G$ venga accettata, verrà restituito al chiamante del metodo il risultato della chiamata.

Parametri:

- * `user: Object`
Parametro che indica l'oggetto contenente tutti i dati dell'utente che dovranno essere salvati dal back-end;
- * `imageObject: Object`
Oggetto contenente i seguenti attributi: + `imageUrl: String`,
+ `image: Object`.



2.6 QuizziPedia::Front-End::Views

2.6.1 Informazioni generali

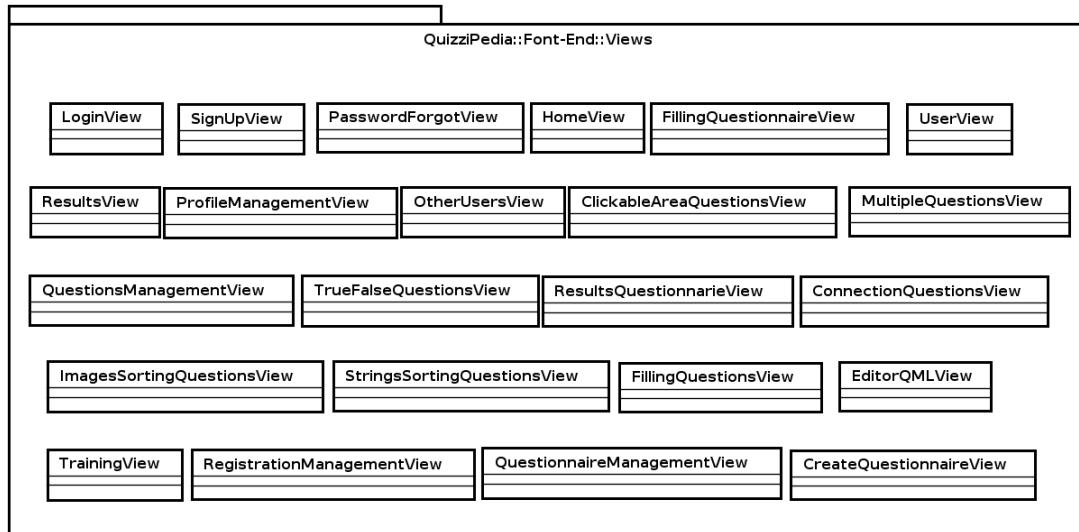


Figura 87: QuizziPedia::Front-End::Views

- **Descrizione:** package contenente le views front-end dell'applicazione;
- **Padre:** Front-End
- **Interazione con altri componenti:**
 - **Controllers:** package contenente i $controllers_G$ front-end dell'applicazione;
 - **Directives:** package contenente le directives front-end dell'applicazione;
 - **ModelViews:** package contenente le classi che saranno presenti nella variabile d'ambiente $\$scope$ di $Angular_G$ che permettono il *Two-Way Data-Binding_G* tra le views e i $controllers_G$;
 - **Views:** package contenente i templates front-end dell'applicazione.

2.6.2 Classi

2.6.2.1 QuizziPedia::Front-End::Views::LoginView

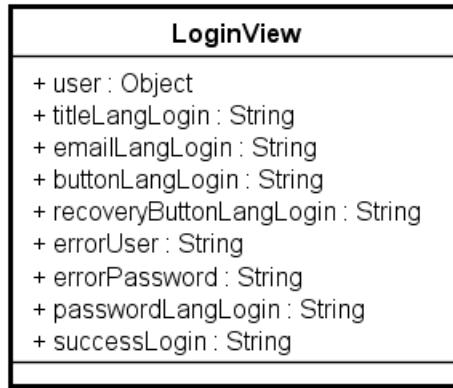


Figura 88: QuizziPedia::Front-End::Views::LoginView

- **Descrizione:** $view_G$ contenente le form necessarie per effettuare il login. Contiene inoltre un link alla pagina di registrazione e uno alla pagina per il recupero della password;
- **Utilizzo:** premette all'utente di autenticarsi inserendo username e password;
- **Relazioni con altre classi:**
 - **IN LoginModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **LoginView** e il $controller_G$ **LoginController**;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ user: Object**
Campo dati contenente due attributi: **username: String** e **password: String**;
 - **+ titleLangLogin: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - **+ emailLangLogin: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento dell'email, in italiano o in inglese;
 - **+ passwordLangLogin: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento della password, in italiano o in inglese;
 - **+ buttonLangLogin: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di autenticazione, in italiano o in inglese;
 - **+ recoveryButtonLangLogin: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link al recupero della password, in italiano o in inglese;
 - **+ successLogin: String**
Attributo che visualizza un messaggio di avvenuta registrazione;



- + errorUser: String
Attributo che visualizza un eventuale messaggio di errore nell'inserimento dell'email, in italiano o in inglese;
- + errorPassword: String
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della password, in italiano o in inglese.

2.6.2.2 QuizziPedia::Front-End::Views::SignUpView

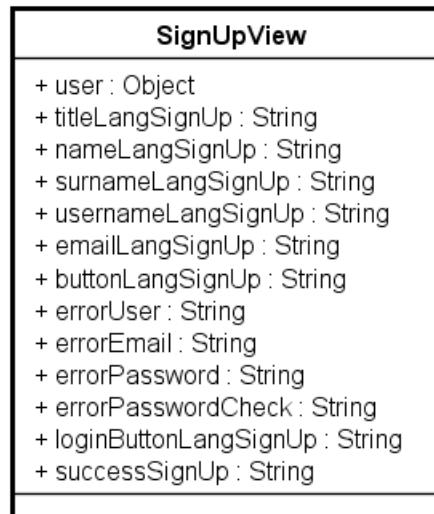


Figura 89: QuizziPedia::Front-End::Views::SignUpView

- **Descrizione:** $view_G$ contenente le form dedicate alla registrazione utente. Contiene inoltre un link alla pagina di login;
- **Utilizzo:** permette all'utente di registrarsi al sistema inserendo i campi dati necessari;
- **Relazioni con altre classi:**
 - **IN SignUpModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile \$scope di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ SignUpView e il $controller_G$ SignUpController;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - + user: Object
Campo dati contenente i seguenti attributi: name: String, surname: String, username: String, email: String, password: string , passwordCheck: String;
 - + titleLangSignUp: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;



- + **nameLangSignUp:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento del nome, in italiano o in inglese;
- + **surnameLangSignUp:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento del cognome, in italiano o in inglese;
- + **usernameLangSignUp:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento del nome utente, in italiano o in inglese;
- + **emailLangSignUp:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento della posta elettronica, in italiano o in inglese;
- + **buttonLangSignUp:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di registrazione, in italiano o in inglese;
- + **loginButtonLangSignUp:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link all'autenticazione, in italiano o in inglese;
- + **successSignUp:** String
Attributo che visualizza un messaggio di avvenuta registrazione;
- + **errorUser:** String
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della username;
- + **errorEmail:** String
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della email;
- + **errorPassword:** String
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della password;
- + **errorPasswordCheck:** String
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della password di conferma.

2.6.2.3 QuizziPedia::Front-End::Views::PasswordForgotView

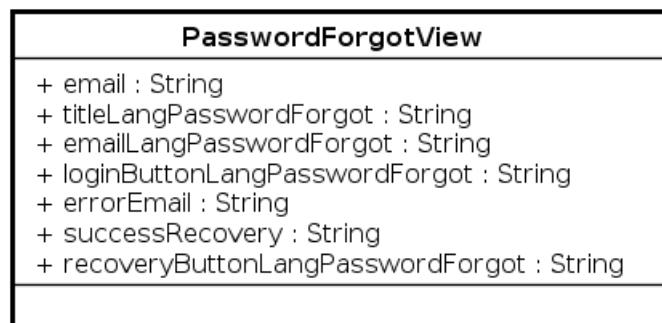


Figura 90: QuizziPedia::Front-End::Views::PasswordForgotView

- **Descrizione:** $view_G$ contenente le form necessarie per il recupero della password dimenticata;



- **Utilizzo:** permette all'utente di recuperare la password dimenticata inserendo i campi dati necessari;
- **Relazioni con altre classi:**
 - **IN PasswordForgotModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `PasswordForgotView` e il $controller_G$ `PasswordForgotController`;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - `+ email: String`
Campo dati contenente l'email per il recupero password;
 - `+ titleLangPasswordForgot: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - `+ emailLangPasswordForgot: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento della posta elettronica, in italiano o in inglese;
 - `+ recoveryButtonLangPasswordForgot: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di invio della nuova password all'indirizzo specificato, in italiano o in inglese;
 - `+ loginButtonLangPasswordForgot: String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link all'autenticazione, in italiano o in inglese;
 - `+ successRecovery: String`
Attributo che visualizza un messaggio di avvenuto invio della nuova password;
 - `+ errorEmail: String`
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della email.

2.6.2.4 QuizziPedia::Front-End::Views::HomeView

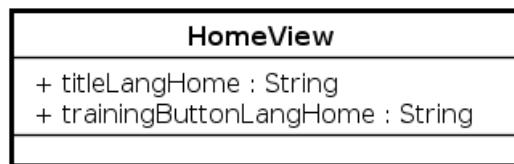


Figura 91: QuizziPedia::Front-End::Views::HomeView

- **Descrizione:** $view_G$ contenente la direttiva per barra di ricerca degli utenti e questionari e il bottone che porterà l'utente nella modalità allenamento;
- **Utilizzo:** viene utilizzata come $view_G$ iniziale dell'applicazione;
- **Relazioni con altre classi:**



- **IN HomeModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **HomeView** e il $controller_G$ **HomeController**;
- **IN SearchDirective:** directive che permette di effettuare la ricerca di utenti e questionari;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ titleLangHome: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
- **+ trainingButtonLangHome: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di link alla modalità allenamento, in italiano o in inglese.

2.6.2.5 QuizziPedia::Front-End::Views::ResultsView

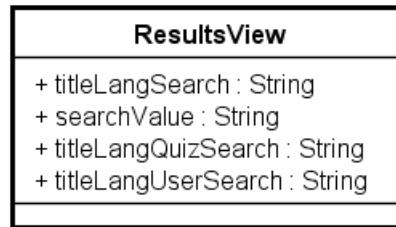


Figura 92: QuizziPedia::Front-End::Views::ResultsView

- **Descrizione:** $view_G$ contenente i risultati della ricerca effettuata. Vengono visualizzati sia gli utenti che i questionari trovati;
- **Utilizzo:** viene visualizzata dopo aver effettuato un'operazione di ricerca utilizzando **SearchDirective** e permette di selezionare un risultato presente al suo interno;
- **Relazioni con altre classi:**

- **IN ResultsModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile di ambiente $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **ResultsView** e il $controller_G$ **SearchController**;
- **IN SubscribeResultDirective:** directive che permette di visualizzare e iscriversi ai questionari ricercati;
- **IN UserResultsDirective:** directive che permette di visualizzare la lista degli utenti ricercati dopo aver utilizzato l'apposita funzione di ricerca;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**



- + titleLangSearch: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
- + searchValue: String
Attributo che rappresenta la stringa cercata;
- + titleLangQuizSearch: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della sezione questionari, in italiano o in inglese;
- + titleLangUserSearch: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della sezione utenti, in italiano o in inglese.

2.6.2.6 QuizziPedia::Front-End::Views::UserView

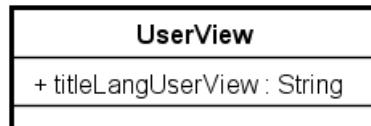


Figura 93: QuizziPedia::Front-End::Views::UserView

- **Descrizione:** $view_G$ contenente le direttive dei dati personali dell'utente, delle sue statistiche relative ai questionari e agli allenamenti effettuati e dei questionari a cui è iscritto;
 - **Utilizzo:** permette ad un utente di visualizzare i propri dati personali e le proprie statistiche e di controllare a quali questionari è iscritto.
 - **Relazioni con altre classi:**
 - **IN StatisticsDirective:** directive che permette di visualizzare le statistiche di un utente;
 - **IN UserDetailsDirective:** directive che permette di visualizzare i dati personali di un utente;
 - **IN QuestionnaireDetailsDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la lista di questionari che può compilare. Ogni item di questa lista contiene:
 - * Nome del questionario;
 - * Autore del questionario;
 - * Argomento del questionario;
 - * Parole chiave del questionario.
- Al verificarsi dell'evento click su un item della lista l'utente verrà indirizzato alla $view_G$ per la compilazione del questionario selezionato;
- **IN QuestionnaireDetailsDoneDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la lista di questionari che ha già compilato e di conseguenza vederne le valutazioni. Ogni item di questa lista contiene:
 - * Nome del questionario;
 - * Autore del questionario;



- * Argomento del questionario;
- * Parole chiave del questionario;
- * Valutazione del questionario.
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ titleLangUserView: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese.

2.6.2.7 QuizziPedia::Front-End::Views::OtherUserView



Figura 94: QuizziPedia::Front-End::Views::OtherUserView

- **Descrizione:** $view_G$ contenente le direttive dei dati personali e delle statistiche di un utente ricercato;
- **Utilizzo:** viene visualizzata dopo la ricerca e permette all'utente che ha l'ha effettuata di visualizzare i dati personali e le statistiche di un utente ricercato;
- **Relazioni con altre classi:**

- **IN StatisticsDirective:** directive che permette di visualizzare le statistiche di un utente;
- **IN UserDetailsDirective:** directive che permette di visualizzare i dati personali di un utente;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ titleLangOtherUserView: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese.

2.6.2.8 QuizziPedia::Front-End::Views::ProfileManagementView

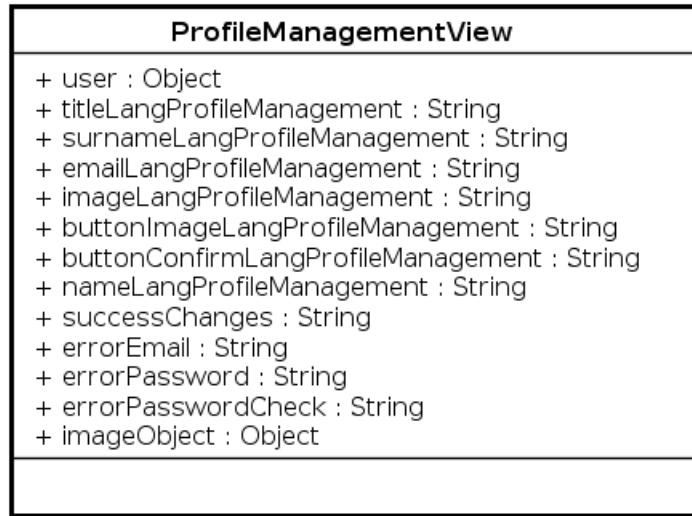


Figura 95: QuizziPedia::Front-End::Views::ProfileManagementView

- **Descrizione:** $view_G$ contenente i dati personali che un utente può modificare dopo essersi registrato al sistema;
- **Utilizzo:** permette all'utente di modificare tutti i campi elencati, tranne l'username, e di rendere persistenti tali modifiche se sono accettate dal sistema;
- **Relazioni con altre classi:**
 - **IN ProfileManagementModelView:** classe di tipo $modelview_G$ la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **ProfileManagementView** e il $controller_G$ **ProfileManagementController**;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ user: Object**
Campo dati contenente i seguenti attributi: `name: String`, `surname: String`, `email: String`, `password: String` e `passwordCheck: String`;
 - **+ imageObject: Object**
Oggetto contenente i seguenti attributi: `+ imageUrl: String`, `+ image: Object`;
 - **+ titleLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - **+ nameLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento del nome, in italiano o in inglese;
 - **+ surnameLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento del cognome, in italiano o in inglese;
 - **+ emailLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento della posta elettronica, in italiano o in inglese;



- **+ imageLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per l'inserimento dell'immagine, in italiano o in inglese;
- **+ buttonImageLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di caricamento immagine, in italiano o in inglese;
- **+ buttonConfirmLangProfileManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;
- **+ successChanges: String**
Attributo che visualizza un messaggio di conferma avvenute modifiche;
- **+ errorEmail: String**
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della email;
- **+ errorPassword: String**
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della password;
- **+ errorPasswordCheck: String**
Attributo che visualizza un eventuale messaggio di errore nell'inserimento della password di conferma.

2.6.2.9 QuizziPedia::Front-End::Views::QuestionsManagementView

QuestionsManagementView
+ titleLangQuestionsManagement : String

Figura 96: QuizziPedia::Front-End::Views::QuestionsManagementView

- **Descrizione:** $view_G$ contenente l'elenco delle domande create;
- **Utilizzo:** visualizza l'elenco delle domande create permettendo all'utente di crearne una nuova o di modificarne una presente nell'elenco;
- **Relazioni con altre classi:**
 - **IN QuestionsManagementModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **QuestionsManagementView** e il *controller_G QuestionsManagementController*;
 - **IN OneQuestionDirective:** rappresenta il componente grafico che visualizza all'utente l'anteprima della domanda che ha creato. Eseguendo l'azione di click su di essa sarà possibile modificare tale domanda. All'interno di **QuestionsManagementView** verranno stampati a video tanti componenti quanti presenti nello $\$scope$ isolato ad esso associato;
 - **IN NewQuestionButtonDirective:** rappresenta il componente grafico che permette all'utente di posizionarsi nella $view_G$ di creazione di una nuova domanda;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.



- **Attributi:**

- + titleLangQuestionsManagement: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese.

2.6.2.10 QuizziPedia::Front-End::Views::TrueFalseQuestionsView

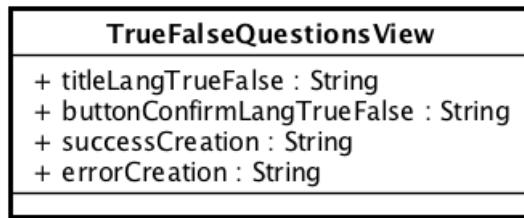


Figura 97: QuizziPedia::Front-End::Views::TrueFalseQuestionsView

- **Descrizione:** $view_G$ contenente le direttive per creare una domanda vero/falso;
- **Utilizzo:** permette all'utente di creare una domanda vero/falso compilando i campi proposti;
- **Relazioni con altre classi:**
 - **IN TrueFalseQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ **TrueFalseQuestionsView** e il $controller_G$ **TrueFalseQuestionsController**;
 - **IN TopicKeywordsDirective:** $directive_G$ che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
 - **IN ImageInTheQuestionDirective:** $directive_G$ per l'inserimento dell'immagine nella creazione delle domande;
 - **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - + titleLangTrueFalse: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - + buttonConfirmLangTrueFalse: String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;
 - + successCreation: String
Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;
 - + errorCreation: String
Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.11 QuizziPedia::Front-End::Views::MultipleQuestionsView

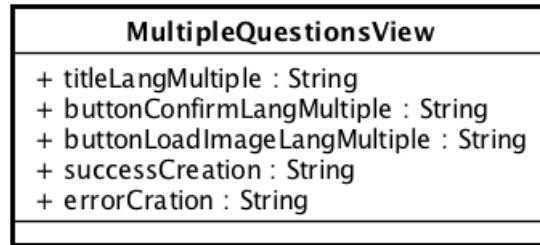


Figura 98: QuizziPedia::Front-End::Views::MultipleQuestionsView

- **Descrizione:** $view_G$ contenente le direttive per creare una domanda a risposta multipla;
- **Utilizzo:** permette all'utente di creare una domanda a risposta multipla compilando i campi proposti;
- **Relazioni con altre classi:**
 - **IN MultipleQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra la $view_G$ **MultipleQuestionsView** e il *controller* $_G$ **MultipleQuestionsController**;
 - **IN TopicKeywordsDirective:** $directive_G$ che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
 - **IN ImageInTheQuestionDirective:** $directive_G$ per l'inserimento dell'immagine nella creazione delle domande;
 - **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
 - **IN MultipleChoiceAnswerDirective:** $directive_G$ contenente il componente grafico per le risposte a scelta;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - + **titleLangMultiple:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - + **buttonConfirmLangMultiple:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;
 - + **buttonLoadImageLang:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di caricamento dell'immagine nel testo della domanda, in italiano o in inglese;
 - + **successCreation:** String
Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;
 - + **errorCreation:** String
Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.12 QuizziPedia::Front-End::Views::ConnectionQuestionsView

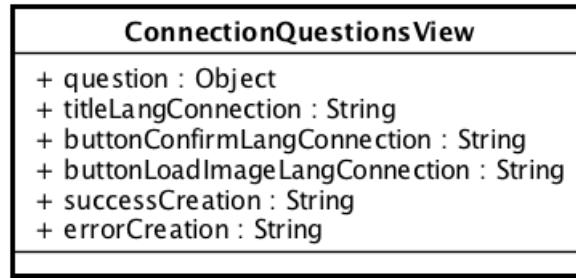


Figura 99: QuizziPedia::Front-End::Views::ConnectionQuestionsView

- **Descrizione:** view contenente i campi e le direttive per creare una domanda a collegamento;
- **Utilizzo:** permette all'utente di creare una domanda a collegamento compilando i campi proposti;
- **Relazioni con altre classi:**
 - **IN ConnectionQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di Angular_G . All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra la view_G `ConnectionQuestionsView` e il `controller` $_G$ `ConnectionQuestionsController`;
 - **IN InputToListModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di Angular_G . All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra la view_G `ConnectionQuestionsView` e il `controller` $_G$ `InputToListController`;
 - **IN TopicKeywordsDirective:** directive_G che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
 - **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ question: Object**
Oggetto contenente gli attributi per la creazione della domanda:
 - * **answer:** array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:
 - **text1:** di tipo `String`, rappresenta il primo elemento testuale che sarà collegato ad un secondo elemento (testuale o immagine);
 - **text2:** di tipo `String`, rappresenta il secondo elemento testuale che sarà collegato al primo elemento (testuale o immagine);
 - **url1:** di tipo `String`, rappresenta il primo elemento immagine che sarà collegato con il secondo elemento (testuale o immagine);
 - **url2:** di tipo `String`, rappresenta il secondo elemento immagine che sarà collegato con il primo elemento (testuale o immagine).
 - **+ titleLangConnection: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;



- + buttonConfirmLangConnection: String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;
- + buttonLoadImageLangConnection: String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di caricamento dell'immagine nel testo della domanda, in italiano o in inglese;
- + successCreation: String
Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;
- + errorCreation: String
Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.13 QuizziPedia::Front-End::Views::ImagesSortingQuestionsView

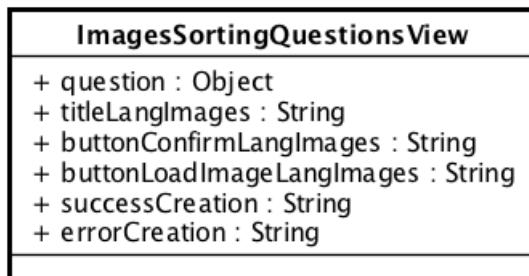


Figura 100: QuizziPedia::Front-End::Views::ImagesSortingQuestionsView

- **Descrizione:** $view_G$ contenente i campi e le direttive per creare una domanda a ordinamento immagini;
- **Utilizzo:** permette all'utente di creare una domanda a ordinamento immagini compilando i campi proposti;
- **Relazioni con altre classi:**
 - **IN ImageSortingQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ *ImagesSortingQuestionsModelView* e il *controller_G ImagesSortingQuestionsController*;
 - **IN InputToListModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ *ImageSortingQuestionsView* e il *controller_G InputToListController*;
 - **IN ImageInTheQuestionDirective:** $directive_G$ per l'inserimento dell'immagine nella creazione delle domande;
 - **IN TopicKeywordsDirective:** $directive_G$ che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
 - **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.



- **Attributi:**

- + **question: Object**

Oggetto contenente gli attributi per la creazione della domanda:

- * **answer:** array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:

- **urlSorting:** attributo di tipo **String** che contiene l' URL_G dell'immagine associata alla risposta;

- **position:** attributo di tipo **Number** che indica la giusta posizione dell'immagine.

- + **titleLangImages: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;

- + **buttonConfirmLangImages: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;

- + **buttonLoadImageLangImages: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di caricamento dell'immagine nel testo della domanda, in italiano o in inglese;

- + **successCreation: String**

Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;

- + **errorCreation: String**

Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.14 QuizziPedia::Front-End::Views::StringsSortingQuestionsView

StringsSortingQuestionsView
+ question : Object + titleLangStrings : String + buttonConfirmLangStrings : String + successCreation : String + errorCreation : String

Figura 101: QuizziPedia::Front-End::Views::StringsSortingQuestionsView

- **Descrizione:** $view_G$ contenente i campi e le direttive per creare una domanda a ordinamento stringhe;
- **Utilizzo:** permette all'utente di creare una domanda a ordinamento stringhe compilando i campi proposti;
- **Relazioni con altre classi:**
 - IN **StringsSortingQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* G tra la $view_G$ **StringsSortingQuestionsView** e il **controller_G StringsSortingQuestionsController**;



- **IN InputToListModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `StringsSortingQuestionsView` e il `controllerG InputToListController`;
- **IN TopicKeywordsDirective:** $directive_G$ che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
- **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ question: Object**
Oggetto contenente gli attributi per la creazione della domanda:
 - * **answer:** array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:
 - **textSorting:** attributo di tipo `String` che contiene il testo della risposta;
 - **position:** attributo di tipo `Number` che indica la giusta posizione del testo.
- **+ titleLangStrings: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
- **+ buttonConfirmLangStrings: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;
- **+ successCreation: String**
Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;
- **+ errorCreation: String**
Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.15 QuizziPedia::Front-End::Views::FillingQuestionsView

FillingQuestionsView
<ul style="list-style-type: none"> + question : Object + titleLangFilling : String + buttonConfirmLangFilling : String + successCreation : String + errorCreation : String

Figura 102: QuizziPedia::Front-End::Views::FillingQuestionsView

- **Descrizione:** $view_G$ contenente i campi e le direttive per creare una domanda a riempimento testo;
- **Utilizzo:** permette all'utente di creare una domanda a riempimento testo compilando i campi proposti;
- **Relazioni con altre classi:**



- **IN FillingQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* tra la $view_G$ **FillingQuestionsView** e il *controller* $_G$ **FillingQuestionsController**;
- **IN TopicKeywordsDirective:** *directive* $_G$ che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
- **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ question: Object**

Oggetto contenente gli attributi per la creazione della domanda:

- * **answer:** array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:

- **wordNumber:** attributo di tipo **Number** che indica la parola nel testo che andrà inserita in fase di compilazione.

- **+ titleLangFilling: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;

- **+ buttonConfirmLangFilling: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;

- **+ successCreation: String**

Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;

- **+ errorCreation: String**

Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.16 QuizziPedia::Front-End::Views::ClickableAreaQuestionsView

ClickableAreaQuestionsView	
<ul style="list-style-type: none"> + question : Object + titleLangClickable : String + buttonConfirmLangClickable : String + successCreation : String + errorCreation : String + buttonLoadImageLangClickable : String 	

Figura 103: QuizziPedia::Front-End::Views::ClickableAreaQuestionsView

- **Descrizione:** $view_G$ contenente i campi e le direttive per creare una domanda ad area cliccabile;
- **Utilizzo:** permette all'utente di creare una domanda ad area cliccabile compilando i campi proposti;



- **Relazioni con altre classi:**

- **IN ClickableAreaQuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G* `ClickableAreaQuestionsView` e il *controller_G* `ClickableAreaQuestionsController`;
- **IN TopicKeywordsDirective:** *directive_G* che permette di gestire l'inserimento di keywords al momento della creazione della domanda;
- **IN QuestionTextDirective:** rappresenta il componente grafico che permette all'utente di scrivere o modificare il testo di una domanda;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.

- **Attributi:**

- **+ question: Object**

Oggetto contenente gli attributi per la creazione della domanda:

- * **url:** attributo di tipo `String` che contiene l'*URL_G* associato all'immagine;
- * **answer:** array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:
 - **x:** attributo di tipo `Number` che rappresenta la posizione della risposta nell'asse delle ascisse all'interno dell'immagine;
 - **y:** attributo di tipo `Number` che rappresenta la posizione della risposta nell'asse delle ordinate all'interno dell'immagine.

- **+ titleLangClickable: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;

- **+ buttonConfirmLangClickable: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di conferma, in italiano o in inglese;

- **+ buttonLoadImageLangClickable: String**

Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di caricamento dell'immagine nel testo della domanda, in italiano o in inglese;

- **+ successCreation: String**

Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;

- **+ errorCreation: String**

Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.17 QuizziPedia::Front-End::Views::EditorQMLView

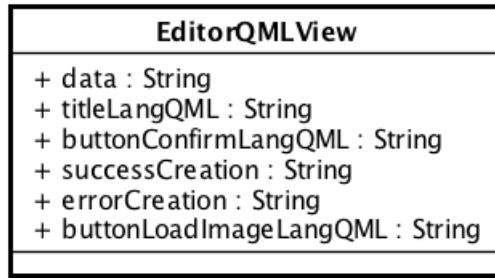


Figura 104: QuizziPedia::Front-End::Views::EditorQMLView

- **Descrizione:** $view_G$ contenente l' $editor_G$ QML_G per la creazione di domande personalizzate;
- **Utilizzo:** permette ad un utente di creare domande personalizzate attraverso la scrittura del codice QML_G direttamente nell' $editor_G$ di testo presente nella $view_G$;
- **Relazioni con altre classi:**
 - IN **EditorQMLModelView**: classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **EditorQMLView** e il $controller_G$ **EditorQMLController**;
 - IN **LangModel**: rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ data: String**
Stringa contenente il testo inserito dall'utente nell'apposito $Editor_G$;
 - **+ titleLangQML: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - **+ buttonConfirmLangQML: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;
 - **+ buttonLoadImageLangQML: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di caricamento dell'immagine nel testo della domanda, in italiano o in inglese;
 - **+ successCreation: String**
Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;
 - **+ errorCreation: String**
Attributo che visualizza un messaggio d'errore per la creazione della domanda.

2.6.2.18 QuizziPedia::Front-End::Views::TrainingView

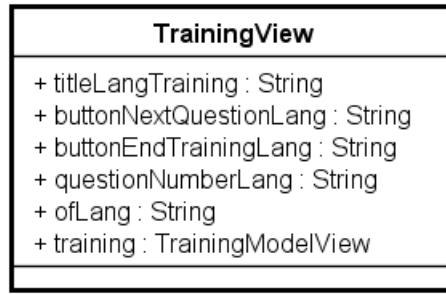


Figura 105: QuizziPedia::Front-End::Views::TrainingView

- **Descrizione:** $view_G$ principale della modalità allenamento; conterrà i vari templates di ogni domanda dell’allenamento;
- **Utilizzo:** all’interno di essa verrà caricato inizialmente il template dove si potranno scegliere l’argomento e le parole chiave dell’allenamento; verranno poi caricati i templates di ogni domanda in base alle preferenze scelte;
- **Relazioni con altre classi:**
 - **IN TrainingModelView:** classe di tipo modelview la cui istanziazione è contenuta all’interno della variabile di ambiente $\$scope$ di $AngularG$. All’interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **TrainingView** e il *controller_G TrainingController*;
 - **IN QuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all’interno della variabile di ambiente $\$scope$ di $AngularG$. All’interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le $directive_G$ che compongono dinamicamente la vista della domanda e il *controller_G QuestionsController*;
 - **IN TrainingSetUpDirective:** rappresenta il componente grafico che permette all’utente di selezionare l’argomento e le parole chiave per iniziare un allenamento con queste caratteristiche. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;
 - **IN HeaderTextQuestionDirective:** rappresenta il componente grafico che presenta all’utente il testo della domanda, l’argomento e le parole chiave. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;
 - **IN TrueFalseAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda vero e falso. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;
 - **IN MultipleChoiceAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda a risposta multipla. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;
 - **IN LinkingAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda di collegamento. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G QuestionsController*;



- **IN SortImagesAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad ordinamento di immagini. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
- **IN SortTextAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad ordinamento di stringhe. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
- **IN EmptySpaceAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare l’esercizio a riempimento di spazi vuoti. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
- **IN ClickableAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad area cliccabile nell’immagine. Viene visualizzato dinamicamente all’interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell’applicazione.

- **Attributi:**

- + **titleLangTraining:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
- + **questionNumberLang:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della frase "domanda numero";
- + **ofLang:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della parola "di";
- + **buttonNextQuestionLang:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di avanzamento a domanda successiva, in italiano o in inglese;
- + **buttonEndTrainingLang:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conclusione dell’allenamento, in italiano o in inglese;
- + **training:** TrainingModelView
Oggetto di tipo **TrainingModelView**, popolato solamente dopo aver iniziato l’allenamento, contenente al suo interno i seguenti campi:
 - * + **argument:** String
Attributo che rappresenta l’argomento scelto;
 - * + **keywords:** Array<String>
array di stringhe che contiene le parole chiave scelte;
 - * + **questionNumber:** String
Attributo che rappresenta il numero progressivo della domanda attuale;
 - * + **numberOfQuestions:** String
Attributo che rappresenta il numero di domande scelte.

2.6.2.19 QuizziPedia::Front-End::Views::FillingQuestionnaireView

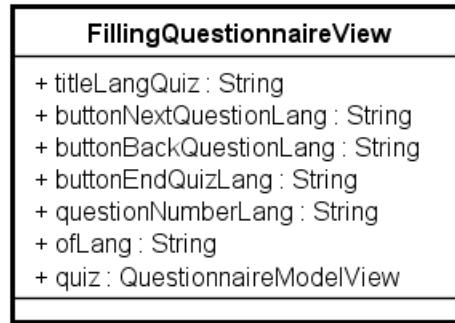


Figura 106: QuizziPedia::Front-End::Views:FillingQuestionnaireView

- **Descrizione:** $view_G$ principale per la compilazione del questionario; conterrà i vari templates di ogni domanda appartenente al questionario;
- **Utilizzo:** all'interno di essa verrà caricato inizialmente il template contenente le informazioni generali relative al questionario; verranno poi caricati i templates di ogni domanda presente nel questionario;
- **Relazioni con altre classi:**
 - **IN FillingQuestionnaireController:** questa classe permette di gestire la compilazione del questionario;
 - **IN FillingQuestionnaireModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **FillingQuestionnaireView** e il $controller_G$ **FillingQuestionnaireController**;
 - **IN QuestionsModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le $directives_G$ che compongono dinamicamente la vista della domanda e il $controller_G$ **QuestionsController**;
 - **IN InfoQuestionnaireDirective:** rappresenta il componente grafico che permette all'utente di visualizzare le informazioni principali del questionario che si sta per svolgere. Viene visualizzato dinamicamente all'interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
 - **IN HeaderTextQuestionDirective:** rappresenta il componente grafico che presenta all'utente il testo della domanda, l'argomento e le parole chiave. Viene visualizzato dinamicamente all'interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
 - **IN TrueFalseAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda vero e falso. Viene visualizzato dinamicamente all'interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;
 - **IN MultipleChoiceAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda a risposta multipla. Viene visualizzato dinamicamente all'interno delle $views_G$ **TrainingView** e **FillingQuestionnaireView** mediante il $controller_G$ **QuestionsController**;



- **IN LinkingAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda di collegamento. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **IN SortImagesAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad ordinamento di immagini. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **IN SortTextAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad ordinamento di stringhe. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **IN EmptySpaceAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare l’esercizio a riempimento di spazi vuoti. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **IN ClickableAnswerDirective:** rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad area cliccabile nell’immagine. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell’applicazione.

- **Attributi:**

- **+ titleLangQuiz:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
- **+ questionNumberLang:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della frase "domanda numero";
- **+ ofLang:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della parola "di";
- **+ buttonNextQuestionLang:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di avanzamento a domanda successiva, in italiano o in inglese;
- **+ buttonBackQuestionLang:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di ritorno alla domanda precedente, in italiano o in inglese;
- **+ buttonEndQuizLang:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione della *label_G* per il bottone di conclusione del questionario, in italiano o in inglese;
- **+ quiz:** `QuestionnaireModelView`
Oggetto di tipo `QuestionnaireModelView` contenente al suo interno i seguenti campi:
 - * **+ title:** `String`
Attributo che rappresenta il titolo del questionario;
 - * **+ argument:** `String`
Attributo che rappresenta l’argomento del questionario;



```
* + keywords: Array<String>
    array di stringhe che contiene le parole chiave del questionario;
* + questionNumber: String
    Attributo che rappresenta il numero progressivo della domanda attuale;
* + numberOfWorks: String
    Attributo che rappresenta il numero di domande.
```

2.6.2.20 QuizziPedia::Front-End::Views::QuestionnaireManagementView

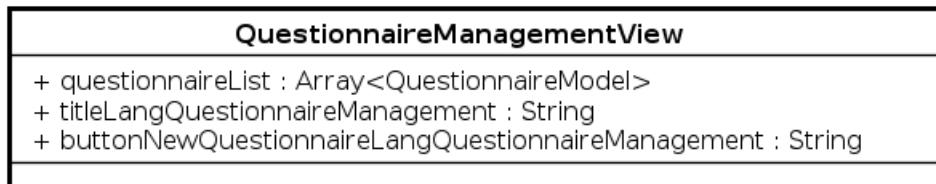


Figura 107: QuizziPedia::Front-End::Views::QuestionnaireManagementView

- **Descrizione:** *view_G* principale per la gestione dei questionari;
- **Utilizzo:** permette di visualizzare tutti i questionari creati, quelli abilitati per la compilazione e quelli non ancora abilitati; contiene inoltre il link per la creazione di un nuovo questionario;
- **Relazioni con altre classi:**
 - **IN QuestionnaireManagementModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G QuestionnaireManagementView* e il *controller_G QuestionnaireManagementController*;
 - **IN EliminationAndModifyDirective:** direttiva contenente i componenti grafici che permettono di eliminare un questionario o di modificarne uno esistente;
 - **IN ExamModalityDirective:** *directive_G* contenente i componenti grafici per attivare la modalità esame su un questionario e gestire le iscrizioni;
 - **IN QuestionnaireResultsDirective:** rappresenta il componente grafico che permette all'utente autenticato di vedere i risultati di chi ha compilato il questionario. Tale componente è contenuto nella lista dei questionari abilitati alla compilazione. È possibile accedere alla lista dei risultati azionando l'evento ad esso collegato;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ questionnaireList:** `Array<QuestionnaireModel>`
array contenente la lista dei questionari creati; ogni questionario sarà rappresentato come un oggetto;
 - **+ titleLangQuestionnaireManagement:** `String`
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;



- + buttonNewQuestionnaireLangQuestionnaireManagement: String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di creazione di un nuovo questionario, in italiano o in inglese.

2.6.2.21 QuizziPedia::Front-End::Views::CreateQuestionnaireView

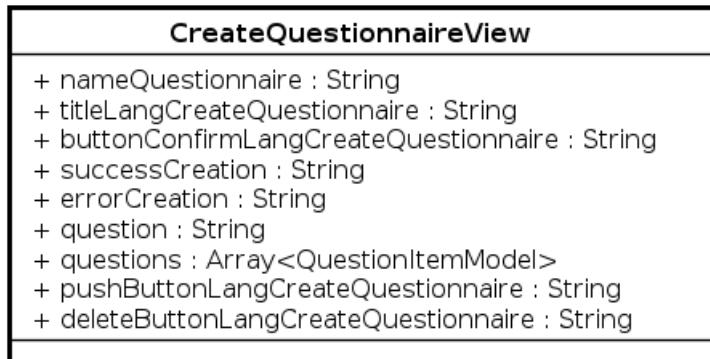


Figura 108: QuizziPedia::Front-End::Views::CreateQuestionnaireView

- **Descrizione:** $view_G$ per la creazione del questionario. In questo componente viene permesso anche all'utente di:
 - Effettuare delle ricerche sul database di domande;
 - Selezionare le domande da inserire nel questionario;
 - Mostrare le domande già inserite e permettere all'utente di eliminarle da tale lista.
- **Utilizzo:** permette all'utente di creare un questionario compilando tutti i campi proposti;
- **Relazioni con altre classi:**
 - **IN CreateQuestionnaireModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **CreateQuestionnaireView** e il **controller_G CreateQuestionnaireController**;
 - **IN TopicKeywordsDirective:** $directive_G$ che permette di gestire l'inserimento dell'argomento e delle keywords al momento della creazione della domanda;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - + nameQuestionnaire: String:
Attributo che specifica il nome del questionario creato;
 - + titleLangCreateQuestionnaire: String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;
 - + buttonConfirmLangCreateQuestionnaire: String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di conferma, in italiano o in inglese;



- + **successCreation:** String
Attributo che visualizza un messaggio di conferma avvenuta creazione della domanda;
- + **errorCreation:** String
Attributo che visualizza un messaggio d'errore per la creazione della domanda.
- + **question:** String
Attributo che conterrà la stringa per la ricerca della domanda;
- + **questions:** Array<QuestionItemModel>
array contenente le domande trovate durante la ricerca;
- + **pushButtonLangCreateQuestionnaire:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di inserimento delle domande selezionate, in italiano o in inglese;
- + **deleteButtonLangCreateQuestionnaire:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione della $label_G$ per il bottone di eliminazione delle domande dal questionario, in italiano o in inglese.

2.6.2.22 QuizziPedia::Front-End::Views::ResultsQuestionnaireView

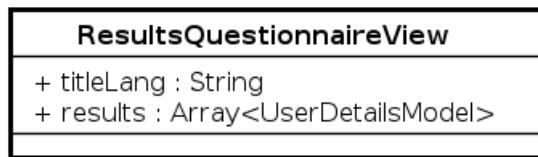


Figura 109: QuizziPedia::Front-End::Views:ResultsQuestionnaireView

- **Descrizione:** $view_G$ contenente i risultati conseguiti dagli utenti che hanno compilato il proprio questionario;
- **Utilizzo:** permette di visualizzare i risultati di ogni utente conseguiti nella compilazione del questionario;
- **Relazioni con altre classi:**
 - **IN ResultsQuestionnaireModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* tra la $view_G$ **ResultsQuestionnaireView** e il *controller_G ResultsController*;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - + **results:** Array<UserDetailsModel>
array contenente un oggetto per ogni iscritto che ha compilato il questionario. L'oggetto sarà composto dai campi: **nome** e **cognome** e **valutazione**;
 - + **titleLang:** String
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese.

2.6.2.23 QuizziPedia::Front-End::Views::RegistrationManagementView



Figura 110: QuizziPedia::Front-End::Views::RegistrationManagementView

- **Descrizione:** $view_G$ che permette di visualizzare gli utenti iscritti ad un questionario;
- **Utilizzo:** permette all'utente di visualizzare gli utenti iscritti ad un questionario e, tramite apposita sezione, approvare l'iscrizione;
- **Relazioni con altre classi:**
 - **IN RegistrationManagementModelView:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **RegistrationManagementView** e il $controller_G$ **RegistrationManagementController**;
 - **IN LangModel:** rappresenta il modello delle informazioni per la giusta traduzione dell'applicazione.
- **Attributi:**
 - **+ subscribers: Array<UserDetailsModel>**
array contenente un oggetto per ogni utente iscritto al questionario. L'oggetto sarà composto dai campi **nome** e **cognome**;
 - **+ buttonAddLang: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del bottone di approvazione iscrizione di un utente, in italiano o in inglese;
 - **+ buttonDeleteLang: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del bottone di disiscrizione di un utente, in italiano o in inglese;
 - **+ titleLangRegistrationManagement: String**
Attributo che viene utilizzato per visualizzare la giusta traduzione del titolo della pagina, in italiano o in inglese;



2.7 QuizziPedia::Front-End::ModelViews

2.7.1 Informazioni generali

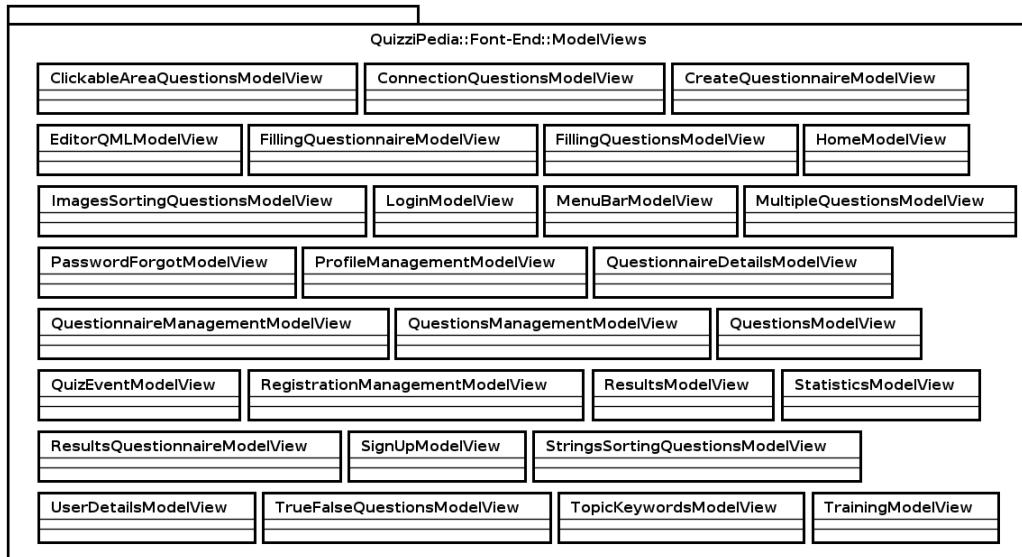


Figura 111: QuizziPedia::Front-End::ModelViews

- **Descrizione:** package contenente le classi che saranno presenti nella variabile d'ambiente $\$scope$ di $Angular_G$ che permettono il *Two-Way Data-Binding_G* tra le $views_G$ e i $controllers_G$;
- **Padre:** Front-End;
- **Interazione con altri componenti:**
 - **Controllers:** $package_G$ contenente i $controllers_G$ front-end dell'applicazione;
 - **Directives:** $package_G$ contenente le $directives_G$ front-end dell'applicazione;
 - **View:** $package_G$ contenente le $views_G$ front-end dell'applicazione;
 - **Templates:** $package_G$ contenente i templates necessari per la creazione dinamica delle viste per le domande.

2.7.2 Classi

2.7.2.1 QuizziPedia::Front-End::ModelViews::ClickableAreaQuestionsModelView

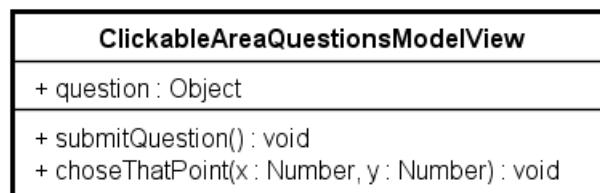


Figura 112: QuizziPedia::Front-End::ModelViews::ClickableAreaQuestionsModelView



- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la `viewG ClickableAreaQuestionsView` e il `controllerG ClickableAreaQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la `viewG ClickableAreaQuestionsView` e il `controllerG ClickableAreaQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - IN `ClickableAreaQuestionsView`: `viewG` contenente i campi e le direttive per creare una domanda ad area cliccabile;
 - IN `ClickableAreaQuestionsController`: questa classe permette di gestire la creazione e la modifica di una domanda ad area cliccabile.
- **Attributi:**
 - + `question: Object`
Oggetto contenente gli attributi per la creazione della domanda:
 - * `url: String`: attributo di tipo `String` che contiene l'`URLG` associato all'immagine;
 - * `answer: Array`: array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:
 - `x: Number`: attributo di tipo `Number` che rappresenta la posizione della risposta nell'asse delle ascisse all'interno dell'immagine;
 - `y: Number`: attributo di tipo `Number` che rappresenta la posizione della risposta nell'asse delle ordinate all'interno dell'immagine.
- **Metodi:**
 - * + `submitQuestion(): void`
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccolge i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando;
 - * + `chooseThatPoint(x: Number, y: Number): void`
Metodo che gestisce l'evento click su un punto dell'immagine. Una volta selezionato esso verrà inserito nell'array di punti.
- **Parametri:**
 - `x: Number`
Parametro contenente la coordinata x del punto;
 - `y: Number`
Parametro contenente la coordinata y del punto.

2.7.2.2 QuizziPedia::Front-End::ModelViews::ConnectionQuestionsModelView

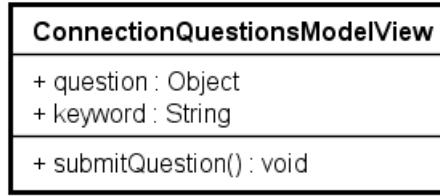


Figura 113: QuizziPedia::Front-End::ModelViews::ConnectionQuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `ConnectionQuestionsView` e il $controller_G$ `ConnectionQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la $view_G$ `ConnectionQuestionsView` e il $controller_G$ `ConnectionQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **IN ConnectionQuestionsView:** $view_G$ contenente i campi e le direttive per creare una domanda a collegamento;
 - **IN ConnectionQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a collegamento.
- **Attributi:**
 - **+ question: Object**
Oggetto contenente gli attributi per la creazione della domanda:
 - * **answer: Array**
array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:
 1. **text1:** di tipo `String`, rappresenta il primo elemento testuale che sarà collegato ad un secondo elemento (testuale o immagine);
 2. **text2:** di tipo `String`, rappresenta il secondo elemento testuale che sarà collegato al primo elemento (testuale o immagine);
 3. **url1:** di tipo `String`, rappresenta il primo elemento immagine che sarà collegato con il secondo elemento (testuale o immagine);
 4. **url2:** di tipo `String`, rappresenta il secondo elemento immagine che sarà collegato con il primo elemento (testuale o immagine).
 - **+ keyword: String**
Attributo contenente la keyword associata alla domanda/questionario;
- **Metodi:**
 - **+ submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.7.2.3 QuizziPedia::Front-End::ModelViews::CreateQuestionnaireModelView

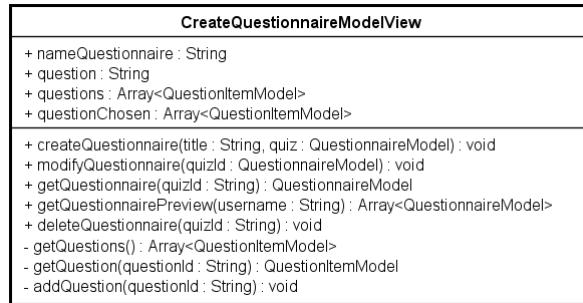


Figura 114: QuizziPedia::Front-End::ModelViews::CreateQuestionnaireModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `CreateQuestionnaireView` e il $controller_G$ `CreateQuestionnaireController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `CreateQuestionnaireView` e il $controller_G$ `CreateQuestionnaireController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT CreateQuestionnaireView:** $view_G$ per la creazione del questionario;
 - **OUT CreateQuestionnaireController:** questa classe permette di gestire la creazione di un questionario.
- **Attributi:**
 - **+ nameQuestionnaire: String;**
Attributo che specifica il nome del questionario creato;
 - **+ question: String**
Attributo che conterrà la stringa per la ricerca della domanda;
 - **+ questions: Array<QuestionItemModel>**
array contenente le domande trovate durante la ricerca;
 - **+ questionsChosen: Array<QuestionItemModel>**
array contenente le domande inserite nel questionario.

• **Metodi:**

- **+ createQuestionnaire(title: String,
quiz: QuestionnaireModel) : void**
Metodo che permette di inserire un questionario nel database tramite richiesta al service.

Parametri:

- * **title: String**
Parametro che indica il nome del questionario;
- * **quiz: QuestionnaireModel**
Parametro che racchiude tutti i dati di un questionario.

- **+ modifyQuestionnaire(quizId: QuestionnaireModel): void**
Metodo che serve per modificare un questionario.

Parametri:



```
* quiz: QuestionnaireModel
    Parametro che rappresenta l'oggetto questionario.

- + getQuestionnaire(quizId: String): QuestionnaireModel
    Metodo che serve per ottenere un questionario tramite l'id in modo da poterlo modificare.

Parametri:
    * quizId: String
        Parametro che rappresenta l'id del questionario da richiedere.

- + getQuestionnairePreview(username: String): Array<QuestionnaireModel>
    Metodo che serve per ottenere la lista di tutti i questionari di un utente.

Parametri:
    * username: String
        Parametro che indica l'utente del quale vogliamo caricare tutti i questionari.

- + deleteQuestionnaire(quizId: String): void
    Metodo che elimina un questionario.

Parametri:
    * quizId: String
        Identificativo del questionario da eliminare.

- - getQuestions(): Array<QuestionItemModel>
    Metodo che permette di ottenere la lista di tutte le domande.

- - getQuestion(questionId: String): QuestionItemModel
    Metodo che ritorna l'intera domanda selezionata.

Parametri:
    * questionId: String
        Parametro che indica l'identificativo univoco di una domanda.

- - addQuestion(questionId: String) : void
    Metodo che permette di inserire una domanda nel questionario.

Parametri:
    * questionId: String
        Parametro che indica l'identificativo univoco di una domanda.
```

2.7.2.4 QuizziPedia::Front-End::ModelViews::EditorQMLModelView

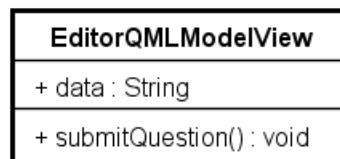


Figura 115: QuizziPedia::Front-End::ModelViews::EditorQMLModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `EditorQMLView` e il $controller_G$ `EditorQMLController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `EditorQMLView` e il $controller_G$ `EditorQMLController` rendendo disponibili variabili e metodi;



- **Relazioni con altre classi:**

- **IN EditorQMLView:** $view_G$ contenente l' $editor_G$ QML_G per la creazione di domande personalizzate;
- **IN EditorQMLController:** questa classe permette di gestire la creazione e la modifica di domande create tramite $editor_G$ QML_G ;

- **Attributi:**

- **+ data: String**
Stringa contenente il testo inserito dall'utente nell'apposito $Editor_G$.

- **Metodi:**

- **+ submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview, li converte attraverso il $parser_G$ QML_G , e li manda al server attraverso **QuestionService**. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.7.2.5 QuizziPedia::Front-End::ModelViews::FillingQuestionnaireModelView

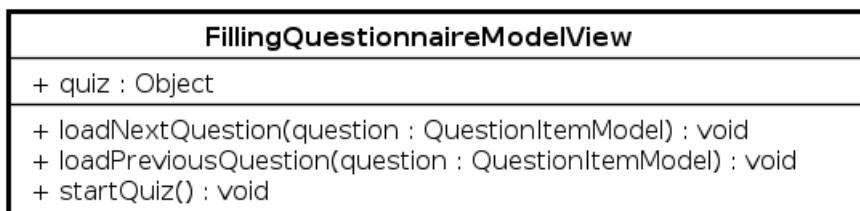


Figura 116: QuizziPedia::Front-End::ModelViews::FillingQuestionnaireModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **FillingQuestionnaireView** e il $controller_G$ **FillingQuestionnaireController**;

- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la $view_G$ **FillingQuestionnaireView** e il $controller_G$ **FillingQuestionnaireController** rendendo disponibili variabili e metodi;

- **Relazioni con altre classi:**

- **OUT FillingQuestionnaireView:** $view_G$ principale per la compilazione del questionario. Conterrà i vari templates di ogni domanda appartenente al questionario;
- **OUT FillingQuestionnaireController:** questa classe permette di gestire la compilazione del questionario.

- **Attributi:**

- **+ quiz: Object**
Oggetto contenente al suo interno i seguenti campi:
* **+ title: String**
Attributo che rappresenta il titolo del questionario;



```
* + argument: String
    Attributo che rappresenta l'argomento del questionario;
* + keywords: Array<String>
    array di stringhe che contiene le parole chiave del questionario;
* + questionNumber: String
    Attributo che rappresenta il numero progressivo della domanda attuale;
* + numberOfQuestions: String
    Attributo che rappresenta il numero di domande.
```

- **Metodi:**

- + loadNextQuestion(question: QuestionItemModel): void
Metodo che invoca l'evento per visualizzare la domanda successiva del quiz tramite `QuestionController`.

Parametri:

- * question: QuestionItemModel
Parametro contenente un riferimento all'oggetto di tipo `QuestionItemModel`.

- + loadPreviousQuestion(question: QuestionItemModel): void
Metodo che invoca l'evento per visualizzare la domanda precedente del quiz tramite `QuestionController`.

Parametri:

- * question: QuestionItemModel
Parametro contenente un riferimento all'oggetto di tipo `QuestionItemModel`.

- + startQuiz(): void
Metodo che gestisce l'evento per iniziare il questionario.

2.7.2.6 QuizziPedia::Front-End::ModelViews::FillingQuestionsModelView

FillingQuestionsModelView
+ question : Object
+ keyword : String
+ submitQuestion() : void
+ choseThatWord(word : String, number : Number) : void

Figura 117: QuizziPedia::Front-End::ModelViews::FillingQuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la `view_G FillingQuestionsView` e il `controller_G FillingQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-BindingG* tra la `view_G FillingQuestionsView` e il `controller_G FillingQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - IN `FillingQuestionsView`: `view_G` contenente i campi e le direttive per creare una domanda a riempimento testo;



- **IN FillingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a riempimento di spazi.

- **Attributi:**

- **+ question: Object**

Oggetto contenente gli attributi per la creazione della domanda:

- * **answer: Array**: array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:

1. **wordNumber: Number**: attributo di tipo **Number** che indica la parola nel testo che andrà inserita in fase di compilazione.

- **+ keyword: String**

Attributo contenente la keyword associata alla domanda/questionario.

- **Metodi:**

- **+ submitQuestion(): void**

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso **QuestionService**. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando;

- **+ choseThatWord(word: String, number: Number): void**

Metodo che gestisce l'evento click su una parola del testo. Una volta selezionata essa verrà inserita nell'array che conterrà le parole che dovranno essere nascoste quando l'esercizio sarà proposto.

Parametri:

- * **word: String**

Parametro contenente la parola scelta da nascondere;

- * **number: Number**

Parametro che si riferisce al numero della parola scelta da nascondere.

2.7.2.7 QuizziPedia::Front-End::ModelViews::HomeModelView

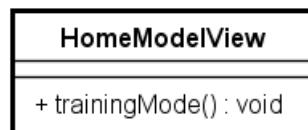


Figura 118: QuizziPedia::Front-End::ModelViews::HomeModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ **HomeView** e il $controller_G$ **HomeController**;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la $view_G$ **HomeView** e il $controller_G$ **HomeController** rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT HomeView:** $view_G$ contenente la direttiva per barra di ricerca degli utenti e questionari e il bottone che porterà l'utente nella modalità allenamento;



- **OUT HomeController:** questa classe permette di gestire la home page;
- **Metodi:**
 - **+ trainingMode(): void**
Metodo che gestisce l'evento click sul pulsante di allenamento. Effettua il redirect alla pagina di allenamento.

2.7.2.8 QuizziPedia::Front-End::ModelViews::ImagesSortingQuestionsModelView

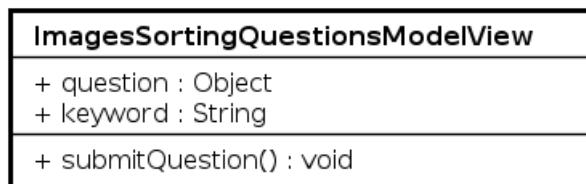


Figura 119: QuizziPedia::Front-End::ModelViews::ImagesSortingQuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `ImagesSortingQuestionsView` e il $controller_G$ `ImagesSortingQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `ImagesSortingQuestionsView` e il $controller_G$ `ImagesSortingQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **IN ImagesSortingQuestionsView:** $view_G$ contenente i campi e le direttive per creare una domanda a ordinamento immagini;
 - **IN ImagesSortingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento immagini.
- **Attributi:**
 - **+ question: Object**
Oggetto contenente gli attributi per la creazione della domanda:
 - * **answer: Array**: array contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:
 1. **urlSorting: String**: attributo di tipo `String` che contiene l' URL_G dell'immagine associata alla risposta;
 2. **position: String**: attributo di tipo `Number` che indica la giusta posizione dell'immagine.
 - **+ keyword: String**
Attributo contenente la keyword associata alla domanda/questionario.
- **Metodi:**
 - **+ submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.



2.7.2.9 QuizziPedia::Front-End::ModelViews::LoginModelView

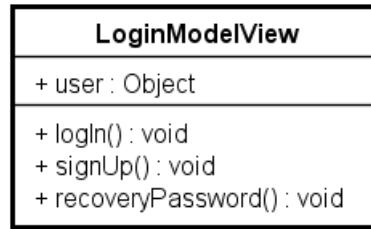


Figura 120: QuizziPedia::Front-End::ModelViews::LoginModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la *view_G LoginView* e il *controller_G LoginController*;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-BindingG* tra la *view_G LoginView* e il *controller_G LoginController* rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT LoginView:** *view_G* contenente le form necessarie per effettuare il login. Contiene inoltre un link alla pagina di registrazione e uno alla pagina per il recupero della password;
 - **OUT LoginController:** questa classe permette di gestire l'autenticazione dell'utente al sistema.
- **Attributi:**
 - + user: Object
Campo dati contenente due attributi: `username: String` e `password: String`.
- **Metodi:**
 - + logIn(email: String, password: String): void
Metodo che richiama il metodo `signin` del service `AuthService` passandogli `email` e `password`. Nel caso di buona riuscita dell'operazione viene effettuato il redirect alla homepage dell'applicazione. Nel caso in cui invece avvenga un errore, viene mostrato a video il messaggio di errore;
 - + signUp(): void
Metodo che gestisce l'evento click sul pulsante di registrazione. Effettua il redirect alla pagina di registrazione;
 - + recoveryPassword(): void
Metodo che gestisce l'evento click sul pulsante di recupero password. Effettua il redirect alla pagina per il recupero della password.

2.7.2.10 QuizziPedia::Front-End::ModelViews::MenuBarModelView

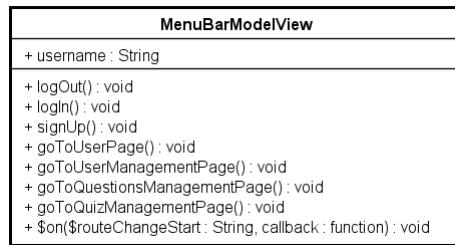


Figura 121: QuizziPedia::Front-End::ModelViews::MenuBarModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G Index* e il *controller_G MenuBarController*;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la *view_G Index* e il *controller_G MenuBarController* rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT LoginBarDirective:** *directive_G* contenente il componente che permette di effettuare il redirect alla pagina di login;
 - **OUT SignUpBarDirective:** *directive_G* contenente il componente che permette di effettuare il redirect alla pagina di registrazione;
 - **OUT UserBarDirective:** *directive_G* contenente il componente che permette di effettuare il redirect alla pagina di visualizzazione del profilo utente personale;
 - **OUT ProfileManagementBarDirective:** *directive_G* contenente il componente che permette di effettuare il redirect alla pagina di gestione del profilo;
 - **OUT QuestionsManagementBarDirective:** *directive_G* contenente il componente che permette di effettuare il redirect alla pagina di gestione delle domande;
 - **OUT LogoutBarDirective:** *directive_G* contenente il componente che permette di effettuare il logout;
 - **OUT QuestionnaireManagementBarDirective:** *directive_G* contenente il componente che permette di effettuare il redirect alla pagina di gestione dei questionari;
 - **OUT MenuBarController:** questa classe permette di gestire il menù fisso per ogni pagina.
- **Attributi:**
 - **+ username: String**
Attributo che conterrà l'username dell'utente.
- **Metodi:**
 - **+ logOut(): void**
Metodo che richiama il metodo `logOut` del service `AuthService` passandogli lo `username`. Prima di effettuare questa operazione viene mostrato a video un messaggio di conferma per il proseguo dell'operazione;
 - **+ logIn(): void**
Metodo che gestisce l'evento click sul pulsante per effettuare il login. Effettua il redirect alla pagina per effettuare il login;



- + `signUp(): void`
Metodo che gestisce l'evento click sul pulsante per effettuare la registrazione. Effettua il redirect alla pagina per effettuare la registrazione;
- + `goToUserPage(): void`
Metodo che gestisce l'evento click sul pulsante di visualizzazione della pagina utente. Effettua il redirect alla pagina di visualizzazione della pagina utente;
- + `goToUserManagementPage(): void`
Metodo che gestisce l'evento click sul pulsante di gestione del profilo utente. Effettua il redirect alla pagina di gestione del profilo utente;
- + `goToQuestionsManagementPage(): void`
Metodo che gestisce l'evento click sul pulsante di gestione delle domande. Effettua il redirect alla pagina di gestione delle domande;
- + `goToQuizManagementPage(): void`
Metodo che gestisce l'evento click sul pulsante di gestione dei questionari. Effettua il redirect alla pagina di gestione dei questionari;
- + `$on('$routeChangeStart': String, callback: function): void`
Metodo che cattura i cambiamenti dell'url e che richiede al `MenuBarModel` le giuste direttive da inserire in `MenuBarDirective`.

Parametri:

- * `'$routeChangeStart': String`
Servizio offerto da *AngularG* per catturare gli eventi sulla barra degli URL;
- * `callback: function`
Funzione di *callbackG* per gestire i cambiamenti della barra degli indirizzi.

2.7.2.11 QuizziPedia::Front-End::ModelViews::MultipleQuestionsModelView

MultipleQuestionsModelView
+ <code>image : String</code> + <code>questionText : String</code> + <code>answers : Array</code> + <code>keyword : String</code> + <code>submitQuestion() : void</code>

Figura 122: QuizziPedia::Front-End::ModelViews::MultipleQuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la *viewG* `MultipleQuestionsView` e il *controllerG* `MultipleQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-BindingG* tra la *viewG* `MultipleQuestionsView` e il *controllerG* `MultipleQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - IN `MultipleQuestionsView`: *viewG* contenente le direttive per creare una domanda a risposta multipla;



- **IN MultipleQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a risposta multipla.

- **Attributi:**

- **+ image: String**
Attributo contenente l'URL dell'immagine caricata dall'utente;
- **+ questionText: String**
Attributo contenente il testo della domanda;
- **+ answers: Array**
array che contiene coppie di valori. Queste coppie sono formate da:
 - * **type: String**
Indica la tipologia della risposta;
 - * **text: String**
Contiene il testo dell'affermazione;
 - * **url: String**
Rappresenta l'immagine della risposta;
 - * **attributesForTForMultiple: Mixed**
Contiene i seguenti attributi:
 1. **isItRight: Boolean**
Contiene se la risposta è vera o falsa.
- **+ keyword: String**
Attributo contenente la keyword associata alla domanda/questionario.

- **Metodi:**

- **+ submitQuestion(): void**
Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso **QuestionService**. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.7.2.12 QuizziPedia::Front-End::ModelViews::PasswordForgotModelView

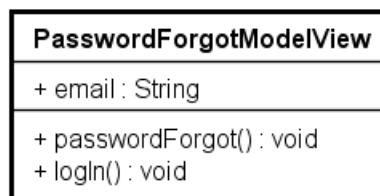


Figura 123: QuizziPedia::Front-End::ModelViews::PasswordForgotModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente **\$scope** di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G PasswordForgotView* e il *controller_G PasswordForgotController*;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la *view_G PasswordForgotView* e il *controller_G PasswordForgotController* rendendo disponibili variabili e metodi;



- **Relazioni con altre classi:**

- **OUT PasswordForgotView:** $view_G$ contenente le form necessarie per il recupero della password dimenticata;
- **OUT PasswordForgotController:** questa classe permette di gestire il ripristino della password dimenticata.

- **Attributi:**

- **+ email: String**
Campo dati contenente l'email per il recupero password.

- **Metodi:**

- **+ passwordForgot(email: String): void**
Metodo che richiama il metodo `passwordForgot` del service `AuthService` passandogli il parametro `email`. Nel caso di buona riuscita dell'operazione, viene mostrato un messaggio di successo il cui corpo contiene anche un bottone per effettuare il redirect alla pagina di login. Nel caso in cui invece avvenga un errore, viene mostrato a video il messaggio di errore;
- **+ logIn(): void**
Metodo che gestisce l'evento click sul pulsante di login. Effettua il redirect alla pagina di login.

2.7.2.13 QuizziPedia::Front-End::ModelViews::ProfileManagementModelView

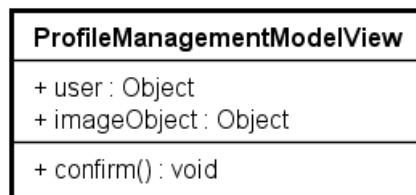


Figura 124: QuizziPedia::Front-End::ModelViews::ProfileManagementModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la $view_G$ `ProfileManagementView` e il $controller_G$ `ProfileManagementController`;

- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la $view_G$ `ProfileManagementView` e il $controller_G$ `ProfileManagementController` rendendo disponibili variabili e metodi;

- **Relazioni con altre classi:**

- **OUT ProfileManagementView:** $view_G$ contenente i dati personali che un utente può modificare dopo essersi registrato al sistema;
- **OUT ProfileManagementController:** questa classe permette di gestire il profilo personale di un utente.

- **Attributi:**



- + user: Object
Campo dati contenente i seguenti attributi: name: String, surname: String, email: String, image: String, password: String e passwordCheck: String;
- + imageObject: Object
Oggetto contenente i seguenti attributi: + imageUrl: String, + image: Object.

- **Metodi:**

- + confirm(user: Object, imageObject: Object): void
Metodo che gestisce l'evento click sul pulsante di conferma modifica. Aggiorna, in caso di modifiche accettate da sistema, l'oggetto locale UserDetailsService. Inoltre, utilizzando il metodo dell'UserDetailsService, aggiorna anche nel server i dati dell'utente.

2.7.2.14 QuizziPedia::Front-End::ModelViews::QuestionnaireDetailsModelView



Figura 125: QuizziPedia::Front-End::ModelViews::QuestionnaireDetailsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente \$scope di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la *view_G UserView* e il *controller_G QuestionnaireDetailsController*;

- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-BindingG* tra la *view_G UserView* e il *controller_G QuestionnaireDetailsController* rendendo disponibili variabili e metodi;

- **Relazioni con altre classi:**

- **OUT QuestionnaireDetailsDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la lista di questionari che può compilare;
- **OUT QuestionnaireDetailsDoneDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la lista di questionari che ha già compilato e di conseguenza vederne le valutazioni;
- **OUT QuestionnaireDetailsController:** questa classe permette di gestire i dettagli di un questionario.

- **Attributi:**

- + questionnaireDetails: Object
Oggetto contenente i seguenti campi dati:
 - * name: String
Nome del questionario;
 - * author: String
Autore del questionario;
 - * topic: String
Argomento del questionario;



- * **keywords:** Array<String>
Parole chiave del questionario.
- + **questionnaireDetailsDone:** Object
Oggetto contenente i seguenti campi dati:
 - * **name:** String
Nome del questionario;
 - * **author:** String
Autore del questionario;
 - * **topic:** String
Argomento del questionario;
 - * **keywords:** Array<String>
Parole chiave del questionario;
 - * **judgement:** Number
Campo che indica il risultato del questionario.

2.7.2.15 QuizziPedia::Front-End::ModelViews::QuestionnaireManagementModelView

QuestionnaireManagementModelView
- questionnaireList : Array<QuestionnaireModel>
+ getUserQuestionnaire(username : int) : String

Figura 126: QuizziPedia::Front-End::ModelViews::QuestionnaireManagementModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `QuestionnaireManagementView` e il $controller_G$ `QuestionnaireManagementController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `QuestionnaireManagementView` e il $controller_G$ `QuestionnaireManagementController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - IN `QuestionnaireManagementView`: $view_G$ principale per la gestione dei questionari;
 - IN `QuestionnaireManagementController`: questa classe permette di gestire tutti i questionari creati da un utente;
- **Attributi:**
 - - **questionnaireList:** Array<QuestionnaireModel>
array contenente la lista dei questionari creati; ogni questionario sarà rappresentato come un oggetto.
- **Metodi:**
 - + **getUserQuestionnaire(username: String): Array<QuestionnaireModel>**
Metodo che ritorna tutti i questionari creati da un utente in un array di `QuestionnaireModel`. **Parametri:**



* `username: String`
 Parametro che indica l'identificativo dell'utente del quale vogliamo scaricare tutti i questionari.

2.7.2.16 QuizziPedia::Front-End::ModelViews::QuestionsManagementModelView

QuestionsManagementModelView	
-	<code>data : Object</code>
+	<code>getQuestionsByUser(username : String) : Array<QuestionItemModel></code>
+	<code>editQuestion(idQuestion : String) : void</code>

Figura 127: QuizziPedia::Front-End::ModelViews::QuestionsManagementModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di Angular_G . All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra la view_G `QuestionsManagementView` e il controller_G `QuestionsManagementController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding* $_G$ tra la view_G `QuestionsManagementView` e il controller_G `QuestionsManagementController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - IN `QuestionsManagementView`: view_G contenente l'elenco delle domande create;
 - IN `QuestionsManagementController`: questa classe permette di gestire le domande create dall'utente e di crearne di nuove.
- **Attributi:**
 - - `data: Object`
 Oggetto contenente le informazioni da mostrare nell'anteprima della domanda.
- **Metodi:**
 - + `getQuestionsByUser(username: String) : Array<QuestionItemModel>`
 Metodo che acquisisce le domande create dall'utente attraverso il `QuestionService`.
Parametri:
`* username: String`
 Parametro di tipo `String` contenente l'username dell'utente.
 - + `editQuestion(idQuestion: String) : Void`
 Metodo che gestisce l'evento click sul pulsante per modificare la domanda. Effettua il redirect alla pagina di modifica della domanda.
Parametri:
`* idQuestion: username`
 Parametro di tipo `String` contenente l'id della domanda da modificare.

2.7.2.17 QuizziPedia::Front-End::ModelViews::QuestionsModelView



QuestionsModelView
+ piecesOfQuestion : Array<Object>
+ objAnswer : Array<Object>
+ addAnswer(index : Number, typeQuestion : String, answerGiven : Array<String>) : void
+ answerGiven(index : Number) : Array<String>
+ orderChosen(index : Number) : Array<String>
+ linkingMade(index : Number) : Array<String>
+ loadNewQuestionBy(topic : String, keywords : Array<String>, level : Number) : void
+ loadNewQuestion(question : QuestionItemModel) : void
+ checkAnswer() : Boolean

Figura 128: QuizziPedia::Front-End::ModelViews::QuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di Angular_G . All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra le *directives* $_G$ che compongono dinamicamente la vista della domanda e il *controller* $_G$ `QuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding* $_G$ tra le *directives* $_G$ che compongono dinamicamente la vista della domanda e il *controller* $_G$ `QuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT QuestionsController:** questa classe permette di gestire il recupero delle domande per poterle stampare nella modalità allenamento;
 - **OUT ClickableAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda ad area cliccabile nell'immagine. Viene visualizzato dinamicamente all'interno delle *views* $_G$ `TrainingView` e `FillingQuestionnaireView` mediante il *controller* $_G$ `QuestionsController`;
 - **OUT EmptySpaceAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare l'esercizio a riempimento di spazi vuoti. Viene visualizzato dinamicamente all'interno delle *views* $_G$ `TrainingView` e `FillingQuestionnaireView` mediante il *controller* $_G$ `QuestionsController`;
 - **OUT HeaderTextQuestionDirective:** rappresenta il componente grafico che presenta all'utente l'argomento, le parole chiave e il numero di domande complessive. Viene visualizzato dinamicamente all'interno delle *views* $_G$ `TrainingView` e `FillingQuestionnaireView` mediante il *controller* $_G$ `QuestionsController`;
 - **OUT LinkingAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda di collegamento. Viene visualizzato dinamicamente all'interno delle *views* $_G$ `TrainingView` e `FillingQuestionnaireView` mediante il *controller* $_G$ `QuestionsController`;
 - **OUT MultipleChoiceAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda a risposta multipla. Viene visualizzato dinamicamente all'interno delle *views* $_G$ `TrainingView` e `FillingQuestionnaireView` mediante il *controller* $_G$ `QuestionsController`;
 - **OUT SortImagesAnswerDirective:** rappresenta il componente grafico che permette all'utente di visualizzare la domanda ad ordinamento di immagini. Viene visualizzato dinamicamente all'interno delle *views* $_G$ `TrainingView` e `FillingQuestionnaireView` mediante il *controller* $_G$ `QuestionsController`;



- **OUT SortTextAnswerDirective**: rappresenta il componente grafico che permette all’utente di visualizzare la domanda ad ordinamento di stringhe. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **OUT TrueFalseAnswareDirective**: rappresenta il componente grafico che permette all’utente di visualizzare la domanda vero e falso. Viene visualizzato dinamicamente all’interno delle *views_G* **TrainingView** e **FillingQuestionnaireView** mediante il *controller_G* **QuestionsController**;
- **OUT TrainingView**: *view_G* principale della modalità allenamento, conterrà i vari templates di ogni domanda dell’allenamento.

• **Attributi:**

- **+ piecesOfQuestion: Array<Object>**
Questo attributo è un **array** di **Object** contenente la domanda da visualizzare dinamicamente attraverso le direttive all’interno le direttive di allenamento e di compilazione dei questionari;

- **+ objAnswer: Array<Object>**
Questo attributo è un **array** di **Object** contenente le risposte date fino a quel momento dall’utente in una domanda. L’**Object** è così formato:

- * **+ typeQuestion: String**
Questo attributo rappresenta il tipo della domanda;
- * **+ answerGiven: Array<String>**
Questo attributo rappresenta le riposte scelte dall’utente fino a quel momento. Può essere creato con una funzione di *callback_G*.

• **Metodi:**

- **+ addAnswer(index: Number, typeQuestion: String, answerGiven: Array<String>): void**

Metodo che gestisce l’evento di selezione delle risposte.

Parametri:

- * **index: Number**

Parametro contenente l’indice della risposta di cui si vuole tenere traccia. Rappresenta anche l’indice dell’**array** **objAnswer** in cui verrà inserito l’oggetto delle risposte date;

- * **typeQuestion: String**

Parametro contenente una stringa la quale indica la tipologia della domanda;

- * **answerGiven: Array<String>**

Parametro contenente l’**array** di risposte date dall’utente aggiornato all’ultima iterazione.

;

- **+ answerGiven(index: Number): Array<String>**

Metodo di supporto che ritorna un **array** di stringhe contenente le risposte date. Si occupa di recuperare le risposte date nelle domande vero/falso, risposta multipla e ad area cliccabile.

Parametri:

- * **index: Number**

Parametro contenente l’indice della risposta di cui si vuole raccogliere le risposte date.



– + `orderChosen(index: Number): Array<String>`

Metodo di supporto che ritorna un **array** di stringhe contenente le risposte date. Si occupa di recuperare le risposte date nelle domande ad ordinamento e di riempimento di spazi.

Parametri:

* `index: Number`

Parametro contenente l'indice della risposta di cui si vuole raccogliere le risposte date.

– + `linkingMade(index: Number): Array<String>`

Metodo di supporto che ritorna un **array** di stringhe contenente le risposte date. Si occupa di recuperare le risposte date nelle domande a collegamento.

Parametri:

* `index: Number`

Parametro contenente l'indice della risposta di cui si vuole raccogliere le risposte date.

– + `loadNewQuestionBy(topic: String, keywords: Array<String>, level: Number): void`

Metodo che gestisce l'evento per scaricare una nuova domanda in base ai parametri passati. Evoca l'evento per inserire la domanda in `TrainingModelView`.

Parametri:

* `topic: String`

Parametro contenente l'argomento della domanda;

* `keywords: Array<String>`

Parametro contenente unarray di stringhe che rappresenta le keywords scelte per l'allenamento;

* `level: Number`

Parametro contenente il livello dell'utente.

– + `loadNewQuestion(question: QuestionItemModel): void`

Metodo che gestisce l'evento per visualizzare una nuova domanda.

Parametri:

* `question: QuestionItemModel`

Parametro contenente un riferimento all'oggetto di tipo `QuestionItemModel`.

– + `checkAnswer(): boolean`

Metodo che controlla che le risposte date siano corrette.

2.7.2.18 QuizziPedia::Front-End::ModelViews::QuizEventModelView

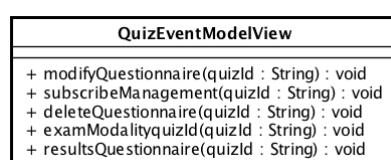


Figura 129: QuizziPedia::Front-End::ModelViews::QuizEventModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra le *directives_G*



`EliminationAndModifyDirective, ExamModalityDirective e
QuestionnaireResultsDirective e il controllerG QuizEventController;`

- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra le *directives_G* `EliminationAndModifyDirective, ExamModalityDirective e QuestionnaireResultsDirective` e il `controllerG QuizEventController` rendendo disponibili variabili e metodi;

- **Relazioni con altre classi:**

- **OUT EliminationAndModifyDirective:** componente grafico contenente i bottoni per eliminare o modificare un questionario;
- **OUT ExamModalityDirective:** *directive_G* contenente i componenti grafici per attivare la modalità esame su un questionario e gestire le iscrizioni;
- **OUT QuestionnaireResultsDirective:** rappresenta il componente grafico che permette all'utente autenticato di vedere i risultati di chi ha compilato il questionario. Tale componente è contenuto nella lista dei questionari abilitati alla compilazione. È possibile accedere alla lista dei risultati azionando l'evento ad esso collegato;
- **OUT QuizEventController:** questa classe permette di reagire ai comandi dell'utente durante la gestione dei suoi questionari.

- **Metodi:**

- `+ modifyQuestionnaire(quizId: String): void`

Metodo che gestisce l'evento click sul pulsante di modifica questionario. Effettua il redirect alla pagina di gestione questionari.

Parametri:

`* quizId: String`

Parametro che indica l'identificativo univoco di un questionario.

- `+ deleteQuestionnaire(quizId: String): void`

Metodo che gestisce l'evento click sul pulsante di eliminazione questionario. Effettua il redirect alla pagina di gestione questionari.

Parametri:

`* quizId: String`

Parametro che indica l'identificativo univoco di un questionario.

- `+ subscribeManagement(quizId: String): void`

Metodo che gestisce l'evento click sul pulsante di gestione iscrizioni. Effettua il redirect alla pagina di gestione iscrizioni. **Parametri:**

Parametri:

`* quizId: String`

Parametro che indica l'identificativo univoco di un questionario.

- `+ examModalityquizId: String(): void`

Metodo che gestisce l'evento click sul pulsante di attivazione modalità esame. Effettua il redirect alla pagina di gestione questionari. **Parametri:**

`* quizId: String`

Parametro che indica l'identificativo univoco di un questionario.

- `+ resultsQuestionnaire(quizId: String): void`

Metodo che gestisce l'evento click sul pulsante di allenamento. Effettua il redirect alla pagina di gestione questionari.

`* quizId: String`

Parametro che indica l'identificativo univoco di un questionario.



2.7.2.19 QuizziPedia::Front-End::ModelViews::RegistrationManagementModelView

RegistrationManagementModelView
+ subscribers : Array<UserDetailsModel>
+ subscribeQuestionnaire(username : String) : void

Figura 130: QuizziPedia::Front-End::ModelViews::RegistrationManagementModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `RegistrationManagementView` e il $controller_G$ `RegistrationManagementController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `RegistrationManagementView` e il $controller_G$ `RegistrationManagementController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **IN** `RegistrationManagementController`: questa classe permette di gestire le iscrizione degli utenti ai questionari;
 - **OUT** `RegistrationManagementView`: $view_G$ che permette di visualizzare gli utenti iscritti ad un questionario.
- **Attributi:**
 - + `subscribers`: `Array<UserDetailsModel>`
array contenente un oggetto per ogni utente iscritto al questionario. L'oggetto sarà composto dai campi `nome` e `cognome`;
- **Metodi:**
 - + `subscribeQuestionnaire(username : String) : void`
Metodo che permette l'iscrizione ad un questionario. Richiama la funzionalità del `QuizService`.

Parametri:
 - * `username`: `String`
Parametro che indica l'utente da iscrivere al questionario.

2.7.2.20 QuizziPedia::Front-End::ModelViews::ResultsModelView

ResultsModelView
+ result : Object
+ goToUser(username : String) : void
+ registrationToQuiz(idQuiz : String) : void
+ goToResultsPage(stringSearch : String) : void

Figura 131: QuizziPedia::Front-End::ModelViews::ResultsModelView



• **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la `viewG ResultsView` e il `controllerG ResultsController`;

• **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-BindingG* tra la `viewG ResultsView` e il `controllerG SearchController` rendendo disponibili variabili e metodi;

• **Relazioni con altre classi:**

- **OUT ResultsView:** `viewG` contenente i risultati della ricerca effettuata, sia gli utenti che i questionari trovati;
- **OUT SearchController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione.

• **Attributi:**

- `+ result: Object`

Attributo che contiene i seguenti due campi:

`* user: Array<Object>`

array contenente un oggetto così formato:

`+ userDetails: Object`

Oggetto contenente i seguenti campi dati:

· `username`.

`* quiz: Array<Object>`

array contenente un oggetto così formato: `+ questionnaireDetails: Object`

Oggetto contenente i seguenti campi dati:

· `name: String`;

· `author: String`;

· `topic: String`;

· `keywords: Array<String>`;

· `idQuiz: ObjectId`.

• **Metodi:**

- `+ goToUser(username: String): void`

Metodo che gestisce l'evento click sul bottone per visualizzare il profilo dell'utente selezionato. Effettua il redirect alla pagina dell'utente.

Parametri:

`* username: String`

Parametro contenente l'username dell'utente di cui si vuole visualizzare il profilo.

- `+ registrationToQuiz(idQuiz: String): void`

Metodo che gestisce l'evento click sul pulsante di registrazione al questionario.

Parametri:

`* idQuiz: String`

Parametro contenente l'id del questionario di cui si vuole effettuare l'iscrizione.

- `+ goToResultsPage(stringSearch: String): void`

Metodo che gestisce l'evento click sul pulsante per effettuare una ricerca.

Parametri:

`* stringSearch: String`

Parametro contenente la stringa della quale effettuare la ricerca.



2.7.2.21 QuizziPedia::Front-End::ModelViews::ResultsQuestionnaireModelView

ResultsQuestionnaireModelView
+ result : Object
+ getQuizResults(quizId : String) : Object
+ getUserForThisQuestionnaire(quizId : String) : Array<UserDetailsModel>

Figura 132: QuizziPedia::Front-End::ModelViews::ResultsQuestionnaireModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di Angular_G . All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding* $_G$ tra la $\text{view}_G \text{ResultsQuestionnaireView}$ e il $\text{controller}_G \text{ResultsQuestionnaireController}$;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding* $_G$ tra la $\text{view}_G \text{ResultsQuestionnaireView}$ e il $\text{controller}_G \text{ResultQuestionnaireController}$ rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT ResultsQuestionnaireView:** view_G contenente i risultati conseguiti dagli utenti che hanno compilato il proprio questionario;
 - **OUT ResultQuestionnaireController:** questa classe permette di gestire la ricerca di questionari e utenti all'interno dell'applicazione;
- **Attributi:**
 - + results: Object
array contenente un oggetto per ogni iscritto che ha compilato il questionario.
L'oggetto sarà composto dai campi: `nome` e `cognome` e `valutazione`.
- **Metodi:**
 - + getQuizResults(quizId: String): Object
Metodo che ritorna i risultati di un questionario.
Parametri:
* quizId: String
Id del questionario del quale recuperare i risultati.
 - + getUserForThisQuestionnaire(quizId: String): Array<UserDetailsModel>
Metodo che ritorna tutti gli utenti che hanno eseguito il questionario, con il loro risultato.
Parametri:
* quizId: String
Id del questionario del quale recuperare gli utenti.

2.7.2.22 QuizziPedia::Front-End::ModelViews::SignUpModelView

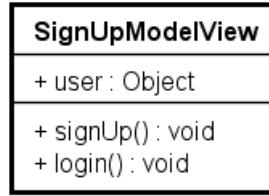


Figura 133: QuizziPedia::Front-End::ModelViews::SignUpModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `SignUpView` e il $controller_G$ `SignUpController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `SignUpView` e il $controller_G$ `SignUpController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT SignUpView:** $view_G$ contenente le form dedicate alla registrazione utente. Contiene inoltre un link alla pagina di login;
 - **OUT SignUpController:** questa classe permette di gestire la registrazione di un utente al sistema.
- **Attributi:**
 - + user: Object
Campo dati contenente i seguenti attributi: name: String, surname: String, username: String, email: String, password: String e passwordCheck: String.
- **Metodi:**
 - + signUp(user: Object): void
Metodo che richiama il metodo `signUp` del service `AuthService` passandogli un oggetto di tipo `SignUpModelView`. Nel caso di buona riuscita dell'operazione viene mostrato un messaggio di successo. Con l'azione di click sul bottone presentato dal messaggio di successo è possibile effettuare il redirect alla pagina di login dell'applicazione. Nel caso in cui invece avvenga un errore, viene mostrato a video il messaggio di errore;
 - + logIn(): void
Metodo che gestisce l'evento click sul pulsante di login. Effettua il redirect alla pagina di login.

2.7.2.23 QuizziPedia::Front-End::ModelViews::StatisticsModelView

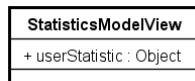


Figura 134: QuizziPedia::Front-End::ModelViews::StatisticsModelView



- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di `AngularG`. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la `viewG UserView` e il `controllerG StatisticsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la `viewG UserView` e il `controllerG StatisticsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT UserView:** `viewG` contenente le direttive dei dati personali dell'utente, delle sue statistiche relative ai questionari e agli allenamenti effettuati e dei questionari a cui è iscritto;
 - **OUT StatisticsController:** questa classe permette di gestire le statistiche di un utente.
- **Attributi:**
 - `+ userStatistic: Object`
Oggetto contenente le statistiche di un utente.

2.7.2.24 QuizziPedia::Front-End::ModelViews::StringsSortingQuestionsModelView

StringsSortingQuestionsModelView
<code>+ question : Object</code>
<code>+ keyword : String</code>
<code>+ submitQuestion() : void</code>

Figura 135: QuizziPedia::Front-End::ModelViews::StringsSortingQuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di `AngularG`. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la `viewG StringsSortingQuestionsView` e il `controllerG StringsSortingQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la `viewG StringsSortingQuestionsView` e il `controllerG StringsSortingQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **IN StringsSortingQuestionsView:** `viewG` contenente i campi e le direttive per creare una domanda a ordinamento stringhe;
 - **IN StringsSortingQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda a ordinamento di stringhe.
- **Attributi:**
 - `+ question: Object`
Oggetto contenente gli attributi per la creazione della domanda:
* `answer: array` contenente oggetti che rappresentano le risposte. Ogni oggetto risposta contiene:



- 1. `textSorting`: attributo di tipo `String` che contiene il testo della risposta;
- 2. `position`: attributo di tipo `Number` che indica la giusta posizione del testo.
- + `keyword`: `String`
Attributo contenente la keyword associata alla domanda/questionario.

- **Metodi:**

- + `submitQuestion()`: `void`

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso `QuestionService`. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.7.2.25 QuizziPedia::Front-End::ModelViews::TopicKeywordsModelView

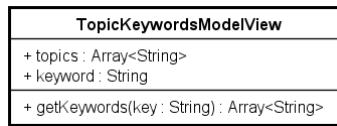


Figura 136: QuizziPedia::Front-End::ModelViews::TopicKeywordsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente `$scope` di *AngularG*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-BindingG* tra la `view_G CreateQuestionnaireView` e il `controller_G KeywordsController`;

- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-BindingG* tra la `view_G CreateQuestionnaireView` e il `controller_G KeywordsController` rendendo disponibili variabili e metodi;

- **Relazioni con altre classi:**

- IN `CreateQuestionnaireView`: `view_G` per la creazione del questionario;
- IN `TopicKeywordsController`: questa classe permette di gestire il recupero delle parole chiave di un questionario.

- **Attributi:**

- + `topics`: `Array<String>`
array contenente le stringhe dei nomi degli argomenti;
- + `keyword`: `String`
Attributo contenente la keyword associata alla domanda/questionario.

- **Metodi:**

- + `getKeywords(key : String) : Array<String>`:
Metodo che ritorna le parole che hanno a che fare con key.

Parametri:

- * `key`: `String`
Parametro che identifica la stringa con la quale cercare le keywords.

2.7.2.26 QuizziPedia::Front-End::ModelViews::TrainingModelView

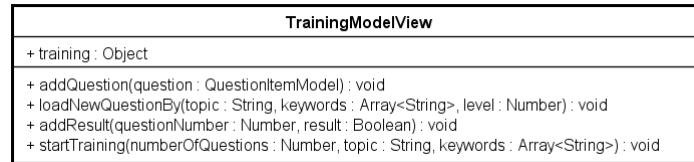


Figura 137: QuizziPedia::Front-End::ModelViews::TrainingModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ **TrainingView** e il $controller_G$ **TrainingController**;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ **TrainingView** e il $controller_G$ **TrainingController** rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **OUT TrainingView:** $view_G$ principale della modalità allenamento, conterrà i vari templates di ogni domanda dell'allenamento;
 - **OUT TrainingController:** questa classe permette di gestire la modalità allenamento sottponendo all'utente le giuste domande adatte al suo livello.
- **Attributi:**
 - + **training: Object**
Oggetto contenente al suo interno i seguenti campi:
 - * + **argument: String**
Attributo contenente l'argomento scelto dall'utente per l'allenamento;
 - * + **keywords: Array<String>**
Attributo contenente l'array di keywords scelte dall'utente per l'allenamento;
 - * + **questionNumber: String**
Attributo che rappresenta il numero progressivo della domanda attuale;
 - * + **numberOfQuestions: String**
Attributo che rappresenta il numero di domande scelte.
- **Metodi:**
 - + **addQuestion(question: QuestionItemModel): void**
Metodo che gestisce l'evento per inserire una domanda nella cronologia delle domande.
Parametri:
 - * **question: QuestionItemModel**
Parametro contenente un riferimento all'oggetto di tipo **QuestionItemModel**.
 - + **loadNewQuestionBy(topic: String, keywords: Array<String>, level: Number): void**
Metodo che emette l'evento per scaricare una nuova domanda in base ai parametri passati.
Parametri:
 - * **topic: String**
Parametro contenente l'argomento della domanda;



```
* keywords: Array<String>
Parametro contenente unarray di stringhe che rappresenta le keywords scelte per l'allenamento;
* level: Number
Parametro contenente il livello dell'utente.

- + addResult(questionNumber: Number, result: Boolean): void
Metodo che gestisce l'evento per inserire il risultato di una domanda nella cronologia delle domande.

Parametri:
* questionNumber: Number
Parametro contenente il numero della domanda risposta;
* result: Boolean
Parametro contenente il risultato della domanda risposta.

- + startTraining(numberOfQuestions: Number, topic: String,
keywords: Array[String]): void
Metodo che gestisce l'evento per iniziare l'allenamento.

Parametri:
* numberOfQuestions: Number
Parametro contenente il numero di domande per l'allenamento;
* topic: String
Parametro contenente l'argomento dell'allenamento;
* keywords: Array<String>
Parametro contenente array di parole chiave.
```

2.7.2.27 QuizziPedia::Front-End::ModelViews::TrueFalseQuestionsModelView

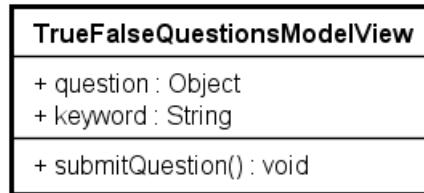


Figura 138: QuizziPedia::Front-End::ModelViews::TrueFalseQuestionsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente $\$scope$ di $Angular_G$. All'interno di essa sono presenti le variabili e i metodi necessari per il $Two-Way Data-Binding_G$ tra la $view_G$ `TrueFalseQuestionsView` e il $controller_G$ `TrueFalseQuestionsController`;
- **Utilizzo:** viene utilizzata per effettuare il $Two-Way Data-Binding_G$ tra la $view_G$ `TrueFalseQuestionsView` e il $controller_G$ `TrueFalseQuestionsController` rendendo disponibili variabili e metodi;
- **Relazioni con altre classi:**
 - **IN TrueFalseQuestionsController:** questa classe permette di gestire la creazione e la modifica di una domanda vero/falso;
 - **OUT TrueFalseQuestionsView:** $view_G$ contenente le direttive per creare una domanda vero/falso.



- **Attributi:**

- + **question:** Object

Oggetto contenente i dati della domanda, ovvero:

- * **questionText:** String: identifica il testo della domanda;

- * **image:** String: identifica l'url di una possibile immagine nella domanda;

- * **answers:** Array: array che contiene coppie di valori. Queste coppie sono formate da:

 - **type:** String: indica la tipologia della risposta;

 - **text:** String: contiene il testo dell'affermazione;

 - **url:** String: rappresenta l'immagine della risposta;

 - **attributesForTForMultiple:** Mixed: contiene i seguenti attributi:

 - 1. **isItRight:** Boolean: contiene se la risposta è vera o falsa.

- * + **keyword:** String

Attributo contenente la keyword associata alla domanda/questionario.

- **Metodi:**

- + **submitQuestion(): void**

Metodo che gestisce l'evento click sul pulsante di conferma sulla domanda. Raccoglie i dati dal modelview e li manda al server attraverso **QuestionService**. Poi verrà effettuato il redirect alla pagina di gestione delle domande oppure al questionario che si stava creando.

2.7.2.28 QuizziPedia::Front-End::ModelViews::UserDetailsModelView

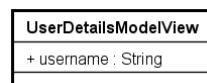


Figura 139: QuizziPedia::Front-End::ModelViews::UserDetailsModelView

- **Descrizione:** classe di tipo modelview la cui istanziazione è contenuta all'interno della variabile di ambiente **\$scope** di *Angular_G*. All'interno di essa sono presenti le variabili e i metodi necessari per il *Two-Way Data-Binding_G* tra la *view_G UserView* e il *controller_G UserDetailsController*;

- **Utilizzo:** viene utilizzata per effettuare il *Two-Way Data-Binding_G* tra la *view_G UserView* e il *controller_G UserDetailsController* rendendo disponibili variabili e metodi;

- **Relazioni con altre classi:**

- **OUT UserView:** *view_G* contenente le direttive dei dati personali dell'utente, delle sue statistiche relative ai questionari e agli allenamenti effettuati e dei questionari a cui è iscritto;

- **OUT UserDetailsController:** questa classe permette di ottenere i dati di un utente.

- **Attributi:**

- + **username:** String

Attributo che conterrà l'username dell'utente ricercato.



3 Specifica del Back-End

3.1 QuizziPedia::Back-End

3.1.1 Informazioni generali

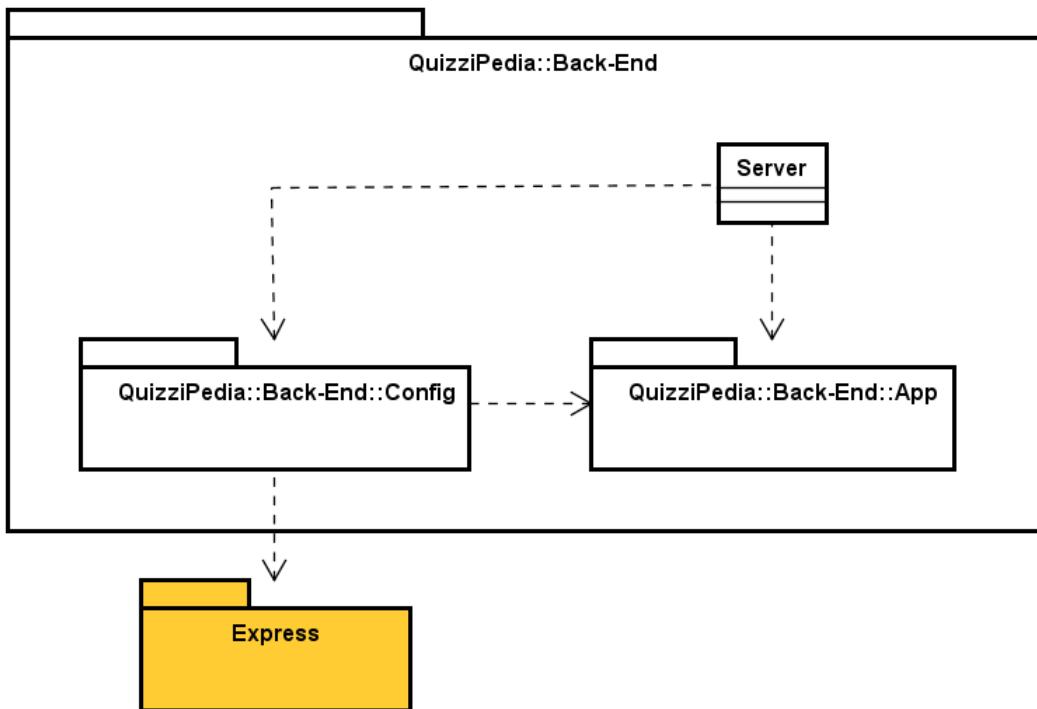


Figura 140: QuizziPedia::Back-End

- **Descrizione:** $package_G$ contenenti le componenti della parte back-end dell'applicazione;
- **Package contenuti:**
 - **App:** $package_G$ contenente le componenti del $server_G$ che implementano il $pattern\ MVC_G$;
 - **Config:** $package_G$ contenente le componenti di configurazione del $server_G$.

3.1.2 Classi

3.1.2.1 QuizziPedia::Back-End::Server

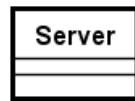


Figura 141: QuizziPedia::Back-End::Server

- **Descrizione:** classe che avvia il $server_G$. Nello specifico apre una connessione al database tramite $Mongoose_G$, invoca il $middleware_G$ $Express_G$ passando un riferimento al database $MongoDB_G$ come parametro in modo che possa configurarsi con esso, invoca il



middleware_G *Passport_G* ed infine si mette in ascolto su una determinata porta. E' il componente *client_G* del *design pattern_G* *Chain of responsibility_G*. Utilizza i moduli *Mongoose_G*, *Express_G*, *Passport_G*;

- **Utilizzo:** utilizzo per avviare l'applicazione lato *server_G*. Inizializza, internamente al back-end, la catena di gestione delle chiamate *REST_G* utilizzando le classi contenute nel *package_G* *Routers*.

3.2 QuizziPedia::Back-End::Config

3.2.1 Informazioni generali

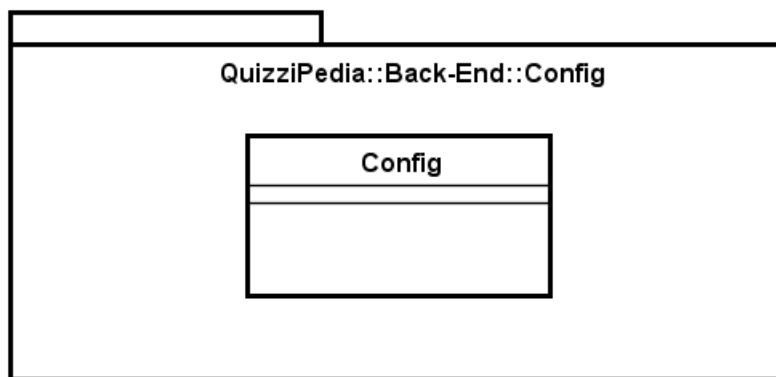


Figura 142: QuizziPedia::Back-End::Config

- **Descrizione:** *package_G* contenente le componenti di configurazione del *server_G*;
- **Padre:** Back-End;
- **Interazioni con altri componenti:**
 - App: *package_G* contenente le componenti del *server_G* che implementano il *pattern MVC_G*.

3.2.2 Classi

3.2.2.1 QuizziPedia::Back-End::Config::Config

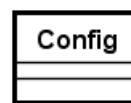


Figura 143: QuizziPedia::Back-End::Config::Config

- **Descrizione:** questa classe gestisce la configurazione del *server_G*. *Non sono stati modellati attributi e metodi di questa classe in quanto viene gestita da Express_G*;
- **Utilizzo:** viene utilizzata per descrivere i parametri dell'applicazione. La classe **Server** utilizza oggetti di questo tipo per creare ed avviare l'istanza del *server_G*;
- **Relazioni con le altre classi:**



- **IN Server:** classe che avvia il $server_G$. Nello specifico apre una connessione al database tramite $Mongoose_G$, invoca il $middleware_G$ $Express_G$ passando un riferimento al database $MongoDB_G$ come parametro in modo che possa configurarsi con esso, invoca il $middleware_G$ $Passport_G$ ed infine si mette in ascolto su una determinata porta. E’ il componente $client_G$ del $design pattern_G$ $Chain of responsibility_G$. Utilizza i moduli $Mongoose_G$, $Express_G$, $Passport_G$.

3.3 QuizziPedia::Back-End::App

3.3.1 Informazioni generali

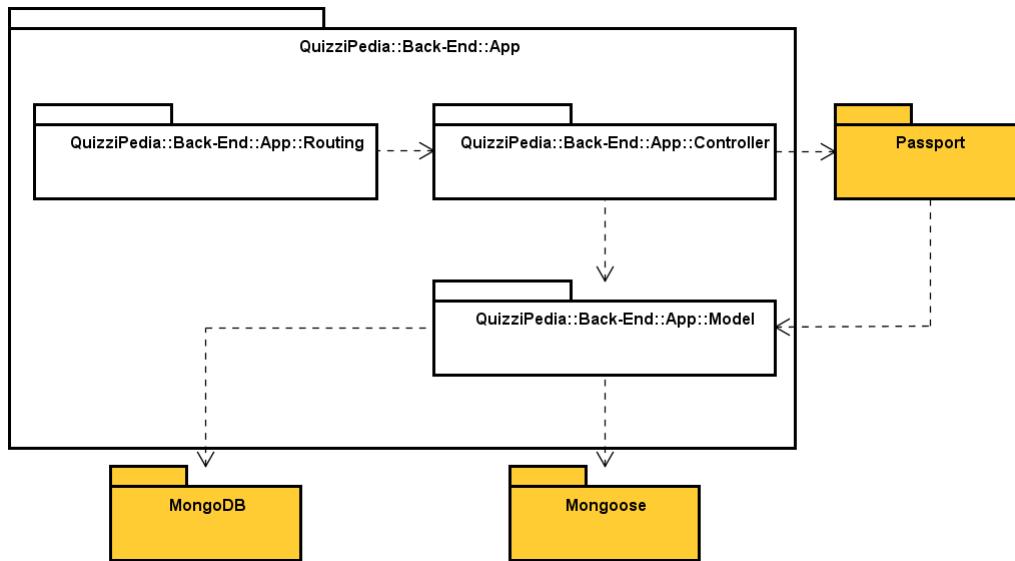


Figura 144: QuizziPedia::Back-End::App

- **Descrizione:** $package_G$ contenente le componenti del $server_G$ che implementano il $pattern_G$ MVC_G ;
- **Padre:** Back-End;
- **Interazioni con altri componenti:**
 - Config: $package_G$ contenente le componenti di configurazione del $server_G$.
- **Package contenuti:**
 - **Controllers:** $package_G$ che contiene i $controllers_G$ di $Express_G$, definisce la logica dell’applicazione;
 - **Models:** $package_G$ che contiene le classi che definiscono il model dell’applicazione. Queste classi sono definite come classi schema di $Mongoose_G$, il quale permette di utilizzare $MongoDB_G$ tramite degli oggetti;
 - **Routers:** $package_G$ contenente i $routers$ della componente back-end dell’applicazione. Contiene i file di configurazione relativi al routing delle richieste del $client_G$, ossia i $routers$ di $Express_G$.

3.4 QuizziPedia::Back-End::App::Controllers

3.4.1 Informazioni generali

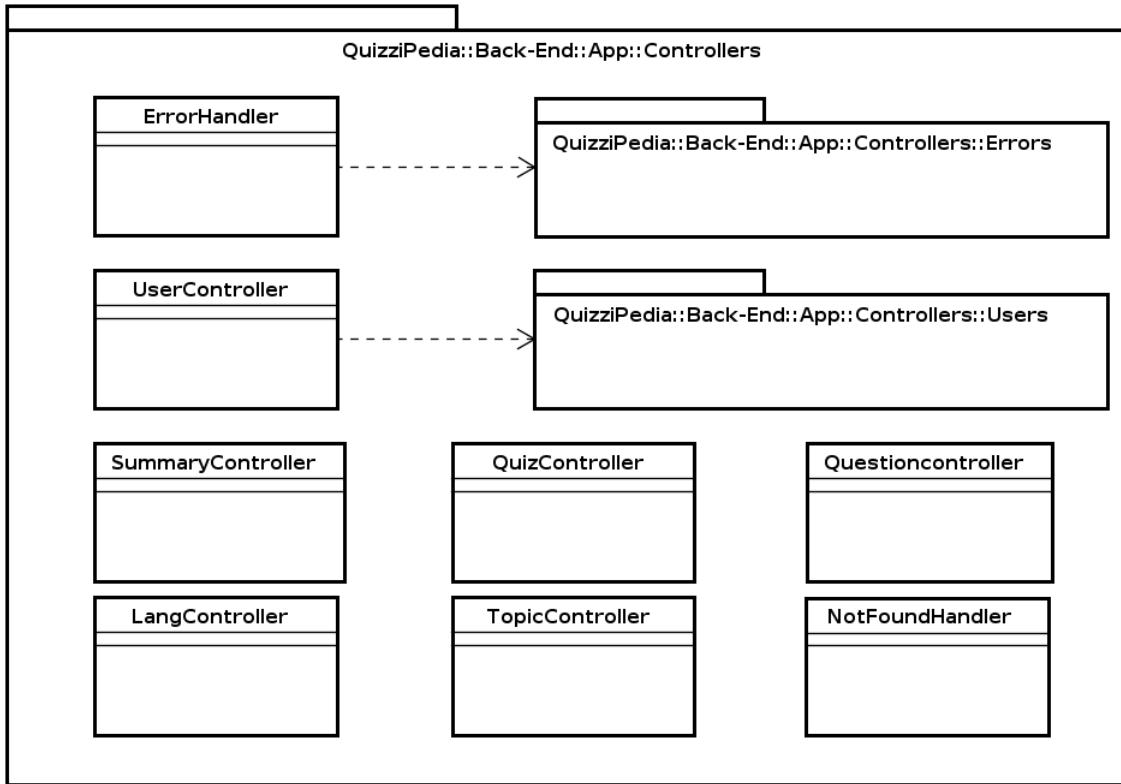


Figura 145: QuizziPedia::Back-End::App::Controllers

- **Descrizione:** $package_G$ contenente i *controllers* di $Express_G$, definisce la logica dell'applicazione;
- **Padre:** App;
- **Interazioni con altri componenti:**
 - **Routers:** $package_G$ contenente i router della componente back-end dell'applicazione. Contiene i file di configurazione relativi al routing delle richieste del $client_G$, ossia i *routers* di $Express_G$;
 - **Views:** $package_G$ contenente le *views_G* della componente back-end dell'applicazione.
- **Package contenuti:**
 - **Errors:** $package_G$ contenente i *controllers* per la gestione degli errori specifici.

3.4.2 Classi

3.4.2.1 QuizziPedia::Back-End::App::Controllers::ErrorsHandler



Figura 146: QuizziPedia::Back-End::App::Controllers::ErrorsHandler



- **Descrizione:** classe $middleware_G$ per la gestione degli errori. Ritorna al client un oggetto di tipo **Response** con stato $HTTP_G$ 500 e descrizione dell'errore in formato $JSON_G$. È un componente $ConcreteHandler_G$ del *design pattern_G Chain of responsibility_G*;
- **Utilizzo:** viene utilizzata quando si verifica un errore. Si preoccupa di delegare la costruzione del messaggio d'errore al modulo specifico qualora questo esista, altrimenti costruisce un messaggio d'errore generico. In questo modo i messaggi d'errore specifici vengono delegati ad un altro modulo, rendendo così possibile aggiungere in futuro altri moduli per gestire più flessibilmente nuove tipologie di errori;
- **Relazioni con altre classi:**
 - **IN QuizRouter:** classe che gestisce le richieste relative alla gestione di un questionario. Componente $ConcreteHandler_G$ del *design pattern_G Chain of responsibility_G*;
 - **IN QuestionRouter:** classe che gestisce le richieste relative alla gestione di una domanda. Componente $ConcreteHandler_G$ del *design pattern_G Chain of responsibility_G*;
 - **IN UserRouter:** classe che gestisce le richieste relative alla gestione di un utente. Componente $ConcreteHandler_G$ del *design pattern_G Chain of responsibility_G*; Utilizza il modulo $Passport_G$;
 - **OUT QuizziPediaError:** classe di gestione degli errori. Esegue la costruzione del messaggio d'errore specifico per i moduli di QuizziPedia::Back-End::App.

- **Metodi:**

- `+ handleError(err: QuizziPediaError, req: Request, res: Response, next: function(QuizziPediaError)):void`
Metodo che gestisce la costruzione dei messaggi d'errore ritornando un $JSON_G$ contenente il messaggio d'errore.

Parametri:

- * `err: QuizziPediaError`
Rappresenta l'errore di tipo **QuizziPediaError**;
- * `req: Request`
Rappresenta la richiesta inviata al $server_G$;
- * `res: Response`
Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;
- * `next: function(QuizziPediaError)`
Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.4.2.2 QuizziPedia::Back-End::App::Controllers::TopicController

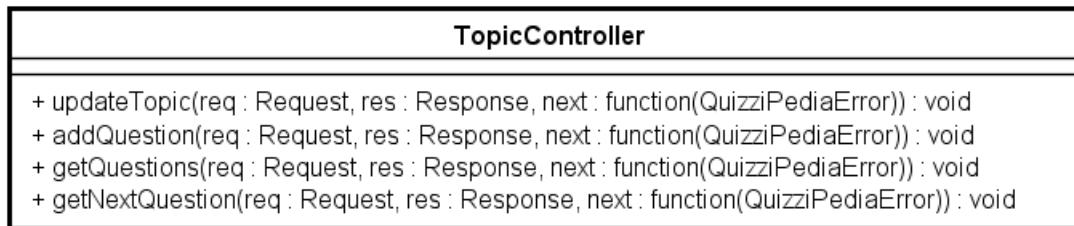


Figura 147: QuizziPedia::Back-End::App::Controllers::TopicController

- **Descrizione:** classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica degli argomenti delle domande;
- **Utilizzo:** viene utilizzata per implementare le funzionalità necessarie a gestire le richieste $REST_G$ legate agli argomenti delle domande;
- **Relazioni con altre classi:**
 - **IN QuestionRouter:** classe che gestisce le richieste relative alle operazioni riguardanti le domande. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **OUT TopicModel:** classe che modella gli argomenti delle domande.
- **Metodi:**
 - `+ updateTopic(req : Request, res : Response, next : function(QuizziPediaError)) : void`
Aggiorna il numero di risposte esatte e totali date a domande sull'argomento da parte degli utenti.
Parametri:
 - * `req: Request`
Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo dell'argomento da aggiornare e l'informazione di se è stata data o meno una risposta corretta alla domanda su quell'argomento;
 - * `res: Response`
Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;
 - * `next: function(QuizziPediaError)`
Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.
 - `+ addQuestion(req : Request, res : Response, next : function(QuizziPediaError)) : void`
Aggiunge una domanda tra le domande sull'argomento.
Parametri:
 - * `req: Request`
Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo della domanda creata da un utente e l'identificativo dell'argomento che tratta;
 - * `res: Response`
Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;



* **next:** function(QuizziPediaError)

Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **getQuestions(req : Request, res : Response,**

next : function(QuizziPediaError)): void

Restituisce le domande sull'argomento.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata al *server_G*. Contiene l'identificativo dell'argomento trattato dalle domande che si vuole ottenere;

* **res:** Response

Rappresenta la risposta che il *server_G* fornirà al termine dell'esecuzione del metodo;

* **next:** function(QuizziPediaError)

Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **getNextQuestion(req : Request,**

res : Response, next : function(QuizziPediaError)): void

Restituisce la domanda successiva di un allenamento.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata al *server_G*. Contiene l'identificativo dell'argomento trattato nell'allenamento e il livello dell'utente che lo sta svolgendo;

* **res:** Response

Rappresenta la risposta che il *server_G* fornirà al termine dell'esecuzione del metodo;

* **next:** function(QuizziPediaError)

Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.4.2.3 QuizziPedia::Back-End::App::Controllers::SummaryController

SummaryController
+ createSummary(req : Request, res : Response, next : function(QuizziPediaError)): void

Figura 148: QuizziPedia::Back-End::App::Controllers::SummaryController

- **Descrizione:** classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica dei riepiloghi dei questionari;



- **Utilizzo:** viene utilizzata per implementare le funzionalità necessarie a gestire le richieste $REST_G$ legate ai riepiloghi dei questionari;

- **Relazione con altre classi:**

- **IN UserRouter:** classe che gestisce le richieste relative alla registrazione, alla gestione della sessione e alla cronologia dei questionari svolti di un utente;
- **OUT SummaryModel:** classe che modella i riepiloghi dei questionari.

- **Metodi:**

- `+ createSummary(req: Request, res: Response,
next:
function(QuizziPediaError)): void`
Crea un riepilogo.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo del questionario per cui verrà creato il riepilogo e le risposte date alle domande;

* `res: Response`

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.4.2.4 QuizziPedia::Back-End::App::Controllers::QuizController

QuizController
<pre>+ createQuiz(req : Request, res : Response, next : function(QuizziPediaError)) : void + editQuiz(req : Request, res : Response, next : function(QuizziPediaError)) : void + addUser(req : Request, res : Response, next : function(QuizziPediaError)) : void + removeUser(req : Request, res : Response, next : function(QuizziPediaError)) : void + addActiveUser(req : Request, res : Response, next : function(QuizziPediaError)) : void + updateStatistic(req : Request, res : Response, next : function(QuizziPediaError)) : void + searchQuiz(req : Request, res : Response, next : function(QuizziPediaError)) : void + getPersonalQuizzes(req : Request, res : Response, next : function(QuizziPediaError)) : void + getQuiz(req : Request, res : Response, next : function(QuizziPediaError)) : void</pre>

Figura 149: QuizziPedia::Back-End::App::Models::Controllers::TopicController

- **Descrizione:** classe che gestisce la logica applicativa riguardante la visualizzazione e la gestione dei questionari.;

- **Utilizzo:** viene utilizzata per implementare le funzionalità necessarie a gestire le richieste $REST_G$ legate alla gestione dei questionari;

- **Relazione con altre classi:**



- **IN QuizRouter:** classe che gestisce le richieste relative alle operazioni riguardanti un questionario;
- **OUT QuizModel:** classe che modella i questionari.

• **Metodi:**

- + **createQuiz(req: Request, res: Response, next: function(QuizziPediaError)): void**
Crea un questionario.

Parametri:

* **req: Request**

Rappresenta la richiesta inviata al $server_G$. Contiene i dati che verranno utilizzati per creare il questionario e gli identificativi delle domande che verranno inserite;

* **res: Response**

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next: function(QuizziPediaError)**

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **editQuiz(req: Request, res: Response, next: function(QuizziPediaError)): void**

Modifica un questionario secondo i dati inseriti dall'utente.

Parametri:

* **req: Request**

Rappresenta la richiesta inviata al $server_G$. Contiene le modifiche apportate dall'utente e l'identificativo del questionario su cui verranno applicate;

* **res: Response**

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next: function(QuizziPediaError)**

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **addUser(req: Request, res: Response, next: function(QuizziPediaError)): void**

Iscrive un utente al questionario.

Parametri:

* **req: Request**

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo del questionario la cui lista delle iscrizioni deve essere modificata;

* **res: Response**

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next: function(QuizziPediaError)**

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.



- + removeUser(req: Request, res: Response,
next: function(QuizziPediaError)): void

Disiscrive un utente al questionario.

Parametri:

* **req: Request**

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo del questionario la cui lista delle iscrizioni deve essere modificata;

* **res: Response**

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next: function(QuizziPediaError)**

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + addActiveUser(req: Request, res: Response,
next: function(QuizziPediaError)): void

Aggiorna la lista di utenti che hanno svolto il questionario.

Parametri:

* **req: Request**

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo del questionario la cui lista delle iscrizioni deve essere modificata;

* **res: Response**

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next: function(QuizziPediaError)**

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + updateStatistic(req: Request, res: Response,
next: function(QuizziPediaError)): void

Aggiorna le statistiche del questionario.

Parametri:

* **req: Request**

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo del questionario le cui statistiche dovranno essere modificate e i valori da aggiornare;

* **res: Response**

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next: function(QuizziPediaError)**

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + searchQuiz(req: Request, res: Response,
next: function(QuizziPediaError)): void

Ricerca un questionario.

Parametri:



* **req:** Request

Rappresenta la richiesta inviata al $server_G$. Contiene il nome del questionario da ricercare;

* **res:** Response

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **getPersonalQuizzes((req: Request, res: Response,**

next: function(QuizziPediaError)): void

Mostra i questionari creati da un utente.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo dell'utente che ha creato i questionari da visualizzare;

* **res:** Response

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **getQuiz((req: Request, res: Response,**

next: function(QuizziPediaError)): void

Mostra il questionario da compilare.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo del questionario da compilare;

* **res:** Response

Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo QuizziPediaError attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.4.2.5 QuizziPedia::Back-End::App::Controllers::QuestionController

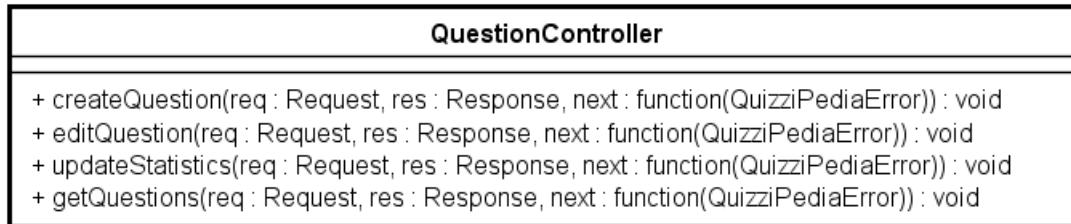


Figura 150: QuizziPedia::Back-End::App::Models::Controllers::QuestionController

- **Descrizione:** classe che gestisce la logica applicativa riguardante la visualizzazione, la creazione e la modifica delle domande presenti nell'applicazione;
- **Utilizzo:** viene utilizzata per implementare le funzionalità necessarie a gestire le richieste $REST_G$ legate alle domande;
- **Relazioni con altre classi:**
 - **OUT QuestionModel:** classe che modella le domande dell'applicazione;
 - **IN QuestionRouter:** classe che gestisce le richieste relative alle operazioni riguardanti le domande.
- **Metodi:**
 - **+ createQuestion(req: Request, res: Response, next: function(QuizziPediaError)): void**
Crea e aggiunge una nuova domanda nel sistema.
Parametri:
 - * **req: Request**
Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo dell'utente autenticato;
 - * **res: Response**
Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;
 - * **next: function(QuizziPediaError)**
Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.
 - **+ editQuestion(req: Request, res: Response, next: function(QuizziPediaError)): void**
Modifica una domanda già esistente nel sistema.
Parametri:
 - * **req: Request**
Rappresenta la richiesta inviata al $server_G$. Contiene l'identificativo dell'utente autenticato. In **req** è presente un campo che rappresenta l'identificativo della domanda nel database che il metodo deve modificare e le relative modifiche;
 - * **res: Response**
Rappresenta la risposta che il $server_G$ fornirà al termine dell'esecuzione del metodo;



* `next: function(QuizziPediaError)`

Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + `updateStatistic(req: Request, res: Response,`

`next: function(QuizziPediaError)): void`

Aggiorna le statistiche sulla difficoltà della domanda, aggiornando anche i campi relativi al numero di risposte totali date alla domanda e al numero di risposte corrette ad ogni risposta da parte di un utente.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata al *server_G*. Contiene l'identificativo della domanda, il livello dell'utente che ha risposto e se è stata data la risposta corretta;

* `res: Response`

Rappresenta la risposta che il *server_G* fornirà al termine dell'esecuzione del metodo;

* `next: function(QuizziPediaError)`

Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + `getQuestions(req: Request, res: Response,`

`next: function(QuizziPediaError)): void`

Ritorna tutte le domande create da un utente.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata al *server_G*. Contiene l'identificativo dell'utente del quale si vogliono ritornare le domande;

* `res: Response`

Rappresenta la risposta che il *server_G* fornirà al termine dell'esecuzione del metodo;

* `next: function(QuizziPediaError)`

Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.4.2.6 QuizziPedia::Back-End::App::Controllers::UserController



Figura 151: QuizziPedia::Back-End::App::Controllers::UserController



- **Descrizione:** classe che raggruppa attraverso require i vari $controllers_G$ responsabili delle operazioni legate alla gestione degli utenti. Si è scelto di predisporre questo raggruppamento per facilitare l'introduzione di nuove funzionalità legate alla gestione degli utenti;
- **Utilizzo:** viene utilizzata per raggruppare i $controllers_G$ responsabili della gestione dei dati degli utenti. In questo modo le classi che vogliono comunicare con i $controllers_G$ legati agli utenti necessitano di includere solo questa classe e non ogni singolo controller;
- **Relazioni con altre classi:**
 - **IN UserRouter:** classe che gestisce le richieste relative alla registrazione e alla gestione della sessione di un utente. Componente ConcreteHandler del design pattern *Chain of responsibility_G*. Utilizza il modulo *Passport_G*;
 - **OUT SessionController:** classe *middleware_G* che, utilizzando *Passport_G*, si occupa di controllare la consistenza dell'oggetto session durante la sessione associata all'utente autenticato. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*;
 - **OUT AuthenticationController:** classe che si occupa della registrazione e dell'autenticazione dell'utente nel sistema. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*. Risulta essere il componente che eventualmente esegue la richiesta del client attraverso *Passport_G*;
 - **OUT UserManagementController:** classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica dei dati dell'utente. Rappresenta il ConcreteHandler nel design pattern *Chain of responsibility_G*. Utilizza *Passport_G*.

3.4.2.7 QuizziPedia::Back-End::App::Controllers::LangController



Figura 152: QuizziPedia::Back-End::App::Controllers::LangController

- **Descrizione:** classe che gestisce la logica applicativa riguardante il passaggio della traduzione delle variabili;
- **Utilizzo:** viene utilizzata per gestire la richiesta di traduzione delle variabili;
- **Relazioni con altre classi:**
 - **IN LangModel:** classe che modella le informazioni riguardanti la lingua dell'applicazione.
- **Metodi:**
 - + getVarlist(req: Request, res: Response, next: function(QuizziPediaError)): void

Parametri:
 - * req: Request

Rappresenta la richiesta inviata al server_G. Contiene la stringa che indica la lingua che si vuole tradurre le variabili;



- * **res:** Response
Rappresenta la risposta che il server fornirà al termine dell'esecuzione del metodo;
- * **next:** function(QuizziPediaError)
Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo *QuizziPediaError* attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.4.2.8 QuizziPedia::Back-End::App::Controllers::NotFoundHandler

NotFoundHandler
+ handle(req : Request, res : Response, next : function(QuizziPediaError))

Figura 153: QuizziPedia::Back-End::App::Controllers::NotFoundHandler

- **Descrizione:** classe che si occupa della gestione dell'errore di pagina non trovata. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
- **Utilizzo:** viene utilizzata per generare una pagina 404 di errore nel caso in cui l'*URI_G* passato non corrisponda ad una risorsa presente nell'applicazione;
- **Relazione con altre classi:**
 - **IN UserRouter:** classe che gestisce le richieste relative alle operazioni riguardanti l'utente. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **IN QuestionRouter:** classe che gestisce le richieste relative alle operazioni riguardanti le domande. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **IN QuizRouter:** classe che gestisce le richieste relative alle operazioni riguardanti i questionari. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*.

• Metodi:

- + handle(req: Request, res: Response,
next: function(QuizziPediaError)): void
Metodo che gestisce la costruzione dei messaggi d'errore ritornando un JSON contenente il messaggio d'errore.

Parametri:

- * **req:** Request
Rappresenta la richiesta inviata al *server_G*;
- * **res:** Response
Rappresenta la risposta che il *server_G* fornirà al termine dell'esecuzione del metodo;
- * **next:** function(QuizziPediaError)
Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo *QuizziPediaError* attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.



3.5 QuizziPedia::Back-End::App::Controllers::Errors

3.5.1 Informazioni generali

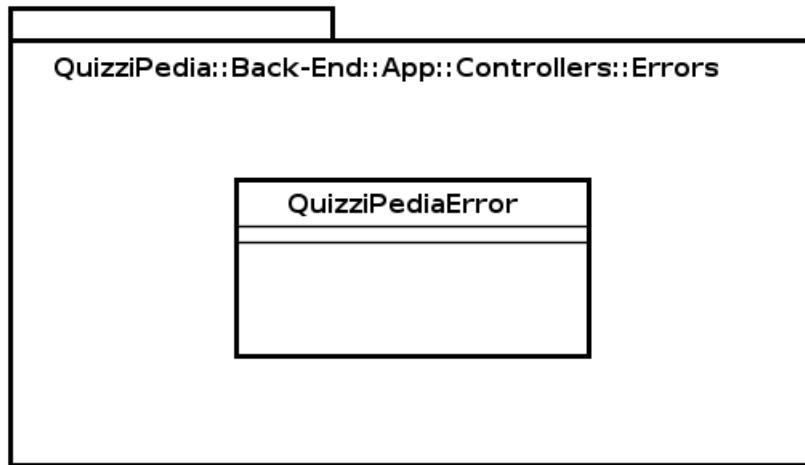


Figura 154: QuizziPedia::Back-End::App::Controllers::Errors

- **Descrizione:** package_G contenente i controllers_G per la gestione degli errori specifici;
- **Padre:** Controllers;

3.5.2 Classi

3.5.2.1 QuizziPedia::Back-End::App::Controllers::Errors::QuizziPediaError

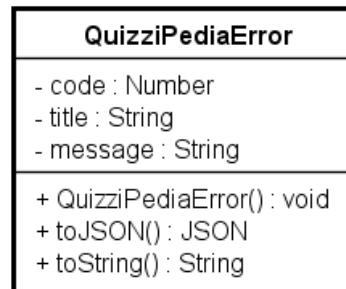


Figura 155: QuizziPedia::Back-End::App::Controllers::QuizziPediaError

- **Descrizione:** classe che contiene gli errori. Esegue la costruzione del messaggio d'errore specifico per i moduli di QuizziPedia::Back-End::App;
- **Utilizzo:** viene utilizzato da ErrorsHandler quando si verifica un errore specifico relativo alle classi di QuizziPedia::Back-End::App;
- **Relazioni con altre classi:**
 - IN ErrorsHandler
Classe middleware_G per la gestione degli errori. Ritorna al client un oggetto di tipo Response con stato $HTTP_G$ 500 e descrizione dell'errore in formato $JSON_G$. E' un componente ConcreteHandler_G del design pattern_G Chain of responsibility_G.



- **Attributi:**

- - `code : Number`
Campo dati contenente il codice dell'errore;
- - `message : String`
Campo dati contenente il messaggio che corrisponde all'errore;
- - `title : String`
Campo dati contenente il titolo dell'errore in forma di stringa.

- **Metodi:**

- + `QuizziPediaError(err: Number)`
Costruttore della classe QuizziPediaError.

Parametri:

- * `err: Number`
Rappresenta il codice identificativo dell'errore.

- + `toJSON(): JSON`
Metodo che ritorna l'errore in formato JSON;

- + `toString(): JSON`
Metodo che effettua una concatenazione dei campi dati dell'errore e li ritorna in formato String.

3.6 QuizziPedia::Back-End::App::Controllers::Users

3.6.1 Informazioni generali

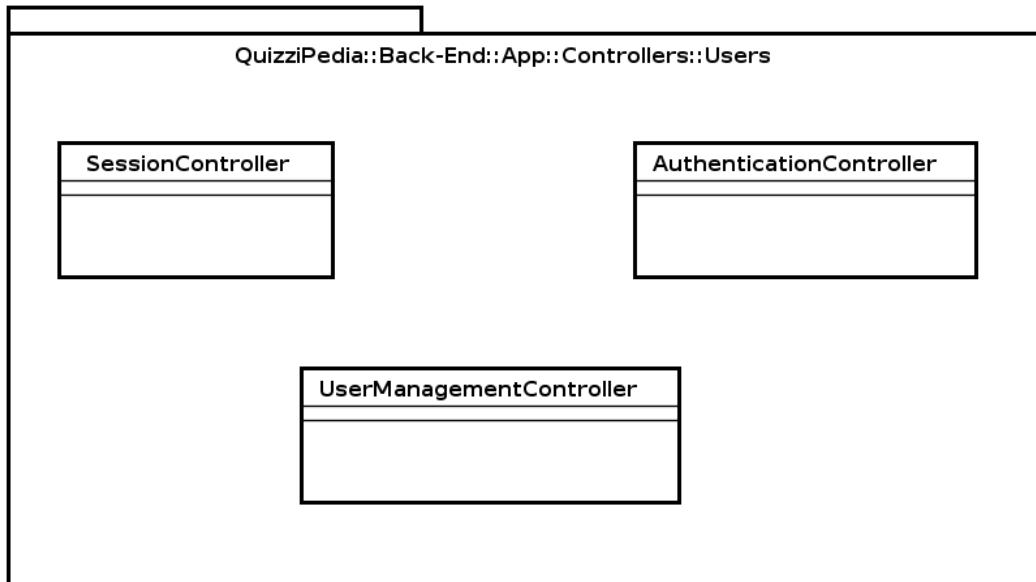


Figura 156: QuizziPedia::Back-End::App::Controllers::Users

- **Descrizione:** $package_G$ contenente i $controllers_G$ relativi alla gestione dell'autenticazione e dei dati dell'utente;
- **Padre:** Controllers;



3.6.2 Classi

3.6.2.1 QuizziPedia::Back-End::App::Controllers::Users::AuthenticationController

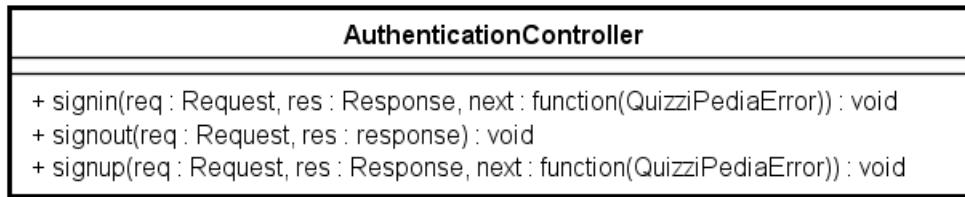


Figura 157: QuizziPedia::Back-End::App::Controllers::Users::AuthenticationController

- **Descrizione:** classe che si occupa della registrazione e dell'autenticazione dell'utente nel sistema. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*. Risulta essere il componente che eventualmente esegue la richiesta del client attraverso *Passport_G*;
- **Utilizzo:** viene utilizzata per implementare le funzionalità necessarie a gestire le richieste *REST_G* legate alla registrazione e all'autenticazione dell'utente;
- **Relazioni con altre classi:**
 - **IN UserController**
Classe che raggruppa attraverso require i vari *controllers_G* responsabili delle operazioni legate alla gestione degli utenti. Si è scelto di predisporre questo raggruppamento per facilitare l'introduzione di nuove funzionalità legate alla gestione degli utenti;
 - **OUT Session**
Classe che gestisce la sessione utente dell'applicazione. Non sono stati modellati attributi e metodi di questa classe in quanto viene inizializzata da *Express_G* ed utilizzata da *Passport_G* attraverso funzionalità interne ai due *middleware_G*;
 - **OUT UserModel**
Classe che modella la creazione e la gestione dei dati utente.
- **Metodi:**
 - `+ signin(req: Request, res: Response, next: function(QuizziPediaError)): void`
Esegue l'autenticazione attraverso *Passport_G*, aggiorna i dati della sessione e risponde al client con i dati non sensibili dell'utente che ha effettuato l'autenticazione.
Parametri:
 - * `req: Request`
Rappresenta la richiesta inviata dal client, contiene la richiesta di login dell'utente;
 - * `res: Response`
Rappresenta la risposta che il server fornirà al termine dell'esecuzione del metodo;
 - * `next: function(QuizziPediaError)`
Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi *middleware_G*. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.



- + signout(req: Request, res: Response): void

Esegue il logout dell'utente dal sistema.

Parametri:

* req: Request

Rappresenta la richiesta inviata dal client, contiene la richiesta di logout inviata dal client;

* res: Response

Rappresenta la risposta che il server fornirà al termine dell'esecuzione del metodo.

- + signup(req: Request, res: Response,

next:function(QuizzipediaError)): void Effettua la registrazione dell'utente nel sistema tramite $Passport_G$ creando ed inserendo un nuovo document all'interno della collection User.

Parametri:

* req: Request

Rappresenta la richiesta inviata dal client, contiene i dati che vengono inseriti nel nuovo document della collection User se tutti i campi dati del nuovo oggetto document creato sono corretti, altrimenti viene attivata la catena di gestione dell'errore;

* res: Response

Rappresenta la risposta che il server fornirà al termine dell'esecuzione del metodo, contiene le informazioni non sensibili che l'utente ha inserito nel database e con cui viene identificato oppure contiene un messaggio d'errore;

* next: function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.6.2.2 QuizziPedia::Back-End::App::Controllers::Users::SessionController

SessionController
+ loggedin(req: Request, res: Response, next: function(QuizziPediaerror)): void

Figura 158: QuizziPedia::Back-End::App::Controllers::Users::SessionController

- **Descrizione:** classe $middleware_G$ che, utilizzando $Passport_G$, si occupa di controllare la consistenza dell'oggetto session durante la sessione associata all'utente autenticato. È un componente ConcreteHandler del design pattern *Chain of responsibility* $_G$;
- **Utilizzo:** componente $middleware_G$ della catena, esegue controlli sullo stato della sessione relativa all'utente. Se l'utente non possiede i permessi adeguati o la sessione è scaduta restituisce un messaggio d'errore, altrimenti passa il controllo al prossimo ConcreteHandler che gestirà il normale flusso d'esecuzione del programma;
- **Relazioni con altre classi:**
 - IN UserController
Classe che raggruppa attraverso require i vari $controllers_G$ responsabili delle operazioni



legate alla gestione degli utenti. Si è scelto di predisporre questo raggruppamento per facilitare l'introduzione di nuove funzionalità legate alla gestione degli utenti;

- **OUT Session**

Classe che gestisce la sessione utente dell'applicazione. Non sono stati modellati attributi e metodi di questa classe in quanto viene inizializzata da $Express_G$ ed utilizzata da $Passport_G$ attraverso funzionalità interne ai due $middleware_G$;

- **OUT UserModel**

Classe che modella la creazione e la gestione dei dati utente.

- **Metodi:**

- + `loggedin(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per verificare che l'utente che esegue una richiesta sia effettivamente un utente autenticato.

Parametri:

- * `req: Request`

Rappresenta la richiesta inviata dal client;

- * `res: Response`

Rappresenta la risposta che fornirà il server;

- * `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo `QuizziPediaError` attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.6.2.3 QuizziPedia::Back-End::App::Controllers::Users::UserManagementController

UserManagementController
<pre>+ updateDataUser(req : Request, res : Response, next : function(QuizziPediaError)) : void + updatePasswordUser(req : Request, res : Response, next : function(QuizziPediaError)) : void + updateStatisticUser(req : Request, res : Response, next : function(QuizziPediaError)) : void + updateSummary(req : Request, res : Response, next : function(QuizziPediaError)) : void + deleteUser((req : Request, res : Response, next : function(QuizziPediaError)) : void + getInfo(req : Request, res : Response, next : function(QuizziPediaError)) : void + getSummary(req : Request, res : Response, next : function(QuizziPediaError)) : void + getSummaries(req : Request, res : Response, next : function(QuizziPediaError)) : void + getUsers(req : Request, res : Response, next : function(QuizziPediaError)) : void + getStatistics(req : Request, res : Response, next : function(QuizziPediaError)) : void</pre>

Figura 159: QuizziPedia::Back-End::App::Controllers::Users::UserManagementController

- **Descrizione:** classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica dei dati dell'utente. Rappresenta il ConcreteHandler nel design pattern *Chain of responsibility* G . Utilizza $Passport_G$;
- **Utilizzo:** viene utilizzata per gestire le richieste $REST_G$ legate agli utenti;
- **Relazioni con altre classi:**



– **IN UserController**

Classe che raggruppa attraverso require i vari $controllers_G$ responsabili delle operazioni legate alla gestione degli utenti. Si è scelto di predisporre questo raggruppamento per facilitare l'introduzione di nuove funzionalità legate alla gestione degli utenti;

– **OUT UserModel**

Classe che modella la creazione e la gestione dei dati utente.

• **Metodi:**

– + `UpdatePasswordUser(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per aggiornare la password dell'utente.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata dal client;

* `res: Response`

Rappresenta la risposta che fornirà il server;

* `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

– + `updateDataUser(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per aggiornare i dati dell'utente.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata dal client;

* `res: Response`

Rappresenta la risposta che fornirà il server;

* `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

– + `updateStatisticsUser(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per aggiornare le statistiche dell'utente.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata dal client;

* `res: Response`

Rappresenta la risposta che fornirà il server;

* `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

– + `updateSummary(req: Request, res: Response, next: function(QuizziPediaError)): void`



Metodo usato dal $middleware_G$ per aggiorna la cronologia dei questionari svolti.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata dal client;

* **res:** Response

Rappresenta la risposta che fornirà il server;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **deleteUser(req: Request, res: Response,**

next: function(QuizziPediaError)): void

Metodo usato dal $middleware_G$ per eliminare un utente dal sistema.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata dal client;

* **res:** Response

Rappresenta la risposta che fornirà il server;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **getInfo(req: Request, res: Response,**

next: function(QuizziPediaError)): void

Metodo usato dal $middleware_G$ per ottenere le informazioni dell'utente.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata dal client;

* **res:** Response

Rappresenta la risposta che fornirà il server;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + **getStatistics(req: Request, res: Response,**

next: function(QuizziPediaError)): void

Metodo usato dal $middleware_G$ per ottenere le statistiche dell'utente.

Parametri:

* **req:** Request

Rappresenta la richiesta inviata dal client;

* **res:** Response

Rappresenta la risposta che fornirà il server;

* **next:** function(QuizziPediaError)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo **QuizziPediaError** attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.



- + `getSummary(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per ottenere il riepilogo di un determinato questionario svolto.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata dal client;

* `res: Response`

Rappresenta la risposta che fornirà il server;

* `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo `QuizziPediaError` attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + `getSummaries(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per ottenere la cronologia dei questionari svolti dall'utente.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata dal client;

* `res: Response`

Rappresenta la risposta che fornirà il server;

* `next: function(QuizziPediaError)`

Rappresenta la callback che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo `QuizziPediaError` attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

- + `getUsers(req: Request, res: Response, next: function(QuizziPediaError)): void`

Metodo usato dal $middleware_G$ per ottenere i dati dei vari utenti.

Parametri:

* `req: Request`

Rappresenta la richiesta inviata dal client;

* `res: Response`

Rappresenta la risposta che fornirà il server;

* `next: function(QuizziPediaError)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione per passare il controllo ai successivi $middleware_G$. La presenza del parametro facoltativo `QuizziPediaError` attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste.

3.7 QuizziPedia::Back-End::App::Models

3.7.1 Informazioni generali

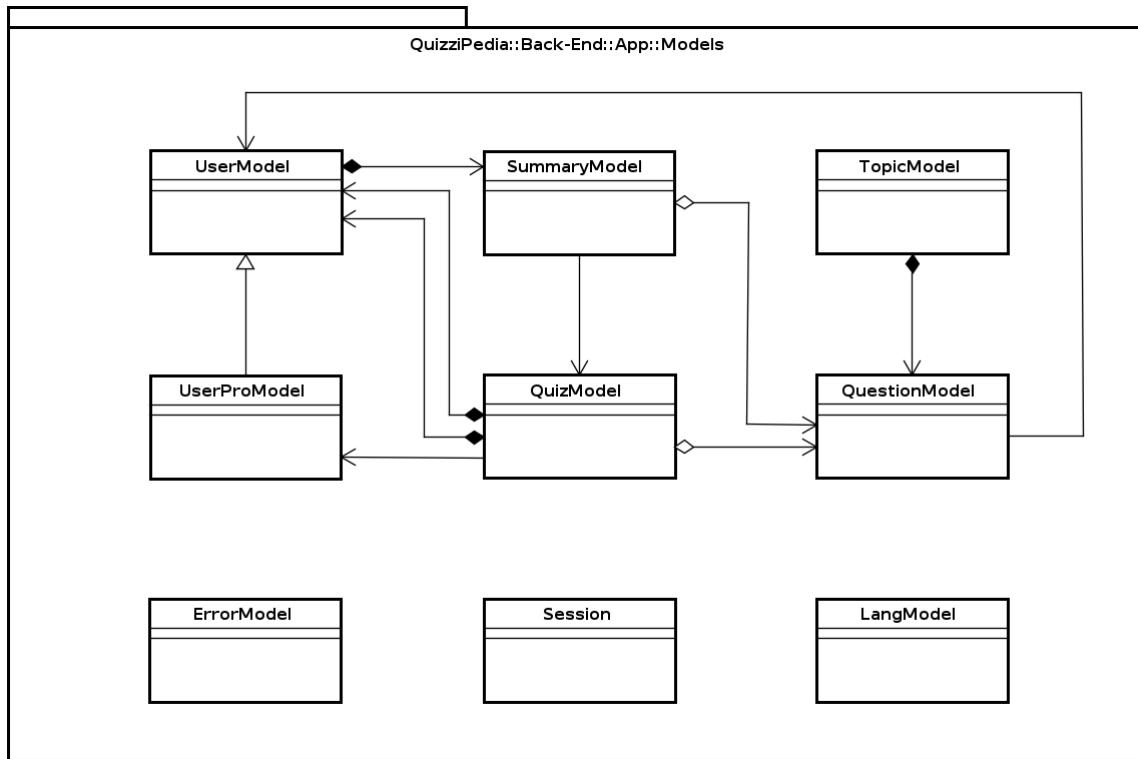


Figura 160: QuizziPedia::Back-End::App::Models

- **Descrizione:** *package_G* contenente le classi che definiscono il *model* dell'applicazione. Queste classi sono definite come classi schema di *Mongoose_G*, il quale permette di utilizzare *MongoDB_G* tramite oggetti;
- **Padre App;**
- **Interazioni con altri componenti:**
 - **Controllers:** *package_G* che contiene i *controllers_G* di *Express_G*, definisce la logica dell'applicazione.

3.7.2 Classi

3.7.2.1 QuizziPedia::Back-End::App::Models::Session



Figura 161: QuizziPedia::Back-End::App::Models::Session

- **Descrizione:** classe che gestisce la sessione utente dell'applicazione. Non sono stati modellati attributi e metodi di questa classe in quanto viene inizializzata da *Express_G* ed utilizzata da *Passport_G* attraverso funzionalità interne ai due *middleware_G*;
- **Utilizzo:** viene utilizzata da *Passport_G* per memorizzare i dati della sessione che viene creata quando un utente effettua il login;



- **Relazioni con altre classi:**

- **IN AuthenticationController**

Classe che si occupa della registrazione e dell'autenticazione dell'utente nel sistema. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*. Risulta essere il componente che eventualmente esegue la richiesta del client attraverso *Passport_G*;

- **IN SessionController**

Classe *middleware_G* che, utilizzando *Passport_G*, si occupa di controllare la consistenza dell'oggetto session durante la sessione associata all'utente autenticato. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*.

3.7.2.2 QuizziPedia::Back-End::App::Models::UserModel

UserModel
- UserSchema : Schema
- hashPassword(password : String) : String
- validPassword(password : String) : String
+ editUser(content : JSON, callback : function(JSON), errback : function(QuizziPediaError)) : void
+ editPassword(password : String, errback : function(QuizziPediaError)) : void
+ setImg(image : String, errback : function(QuizziPediaError)) : void
+ updateStatistics(statistics : JSON, callback : function(JSON)) : void
+ upLevel(callback : function(JSON)) : void
+ deleteUser(callback : function(JSON)) : void
+ updateSummary(summaryId : ObjectId) : void
+ getSummaries(callback : function(JSON), errback : function(QuizziPediaError)) : void
<u>+ getUsers(searchword : String, callback : function(JSON), errback : function(QuizziPediaError)) : void</u>

Figura 162: QuizziPedia::Back-End::App::Models::UserModel

- **Descrizione:** classe che modella la creazione e la gestione dei dati utente;
- **Utilizzo:** viene utilizzata per rappresentare i dati degli account dei vari utenti dell'applicazione. Si interfaccia alla libreria *Mongoose_G* per la creazione dello schema e dei relativi metodi statici o di istanza;
- **Relazioni con altre classi:**

- **IN AuthenticationController**

Classe che si occupa della registrazione e dell'autenticazione dell'utente nel sistema. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*. Risulta essere il componente che eventualmente esegue la richiesta del client attraverso *Passport_G*;

- **IN SessionController**

Classe *middleware_G* che, utilizzando *Passport_G*, si occupa di controllare la consistenza dell'oggetto session durante la sessione associata all'utente autenticato. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*;

- **IN UserManagementController**

Classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica dei dati dell'utente. Rappresenta il ConcreteHandler nel design pattern *Chain of responsibility_G*. Utilizza *Passport_G*;



– **OUT SummaryModel**

Questa classe rappresenta il riepilogo dei questionari svolti dagli utenti;

– **IN QuestionModel**

Questa classe rappresenta i dati delle domande create dai vari utenti;

– **IN QuizModel**

Questa classe rappresenta i dati dei questionari creati dagli utenti Pro;

– **IN UserProModel**

Questa classe rappresenta i dati riguardanti l'utente pro.

• **Attributi:**

– – **userSchema: Schema**

Questo campo dati rappresenta lo schema *MongooseG* dell'utente Quizzipedia. Lo schema prevede i seguenti attributi:

* **name: String**

Rappresenta il nome dell'utente registrato;

* **surname: String**

Rappresenta il cognome dell'utente registrato;

* **email: String**

Rappresenta l'email dell'utente registrato;

* **userImg: String**

Rappresenta il path della foto profilo dell'utente registrato;

* **username: String**

Rappresenta l'username con cui viene identificato l'utente all'interno dell'applicazione;

* **password: String**

Rappresenta la password associata all'utente, appositamente codificata mediante l'algoritmo bcrypt;

* **statistics: Array<Mixed>**

Contenente i seguenti attributi:

· **topicName: String**

Rappresenta il nome della statistica relativa all'argomento;

· **topicLevel: Number**

Identifica il livello di preparazione dell'utente in un determinato argomento;

· **correctAnswers: Number**

Identifica il numero di risposte corrette date dall'utente riguardanti domande di un determinato argomento;

· **totalAnswers: Number**

Identifica il numero di risposte totali date dall'utente riguardanti domande di un determinato argomento.

* **experienceLevel: Number**

Identifica il livello dell'utente;

* **quizSummaries: Array<ObjectId>**

Array che contiene oggetti di tipo **ObjectId**, che rappresentano i riferimenti agli identificativi nel database dei questionari svolti dall'utente.

• **Metodi:**

– - **hashPassword(password: String): String**

Effettua l'hashing della stringa password se non è già stata criptata tramite campo



salt per evitare attacchi di tipo rainbow.

Parametri:

* `password: String`

Rappresenta la password dell'utente.

- `- validPassword(password: String): String`

Effettua la validità della password inserita comparandola con la password criptata.

Parametri:

* `password: String`

Rappresenta la password inserita dall'utente.

- `+ editUser(content: JSON, callback: function(JSON), errback: function(QuizziPediaError)): void`

Questo metodo aggiorna i dati personali dell'utente. Restituisce un oggetto $JSON_G$ che descrive l'elemento dopo l'aggiornamento oppure un messaggio di errore.

Parametri:

* `content: JSON`

Rappresenta i dati dell'utente da aggiornare;

* `callback: function(JSON)`

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione;

* `errback: function(QuizziPediaError)`

Questo parametro rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.

- `+ editPassword(password: String,`

`errback: function(QuizziPediaError)): void`

Questo metodo aggiorna la password vecchia dell'utente con quella passata per parametro. Restituisce un messaggio di errore nel caso in cui si verificati dei problemi nel cambio della password.

Parametri:

* `password: String`

Rappresenta la nuova password dell'utente da sostituire a quella vecchia;

* `errback: function(QuizzipediaError)`

Questo parametro rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.

- `+ setImg(image : String, errback : function(QuizziPediaError)): void`

Questo metodo aggiorna l'immagine profilo dell'utente. Restituisce un messaggio di errore nel caso in cui si verificati dei problemi nell'aggiornamento dell'immagine.

Parametri:

* `image: String`

Rappresenta l'immagine per aggiornare la foto del profilo;

* `errback: function(QuizziPediaError)`

Questo parametro rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.

- `+ updateStatistics(statistics : JSON, callback : function(JSON)) : void`

Questo metodo aggiorna le statistiche dell'utente in un determinato argomento. Restituisce un oggetto $JSON_G$ che descrive le statistiche dell'utente dopo l'aggiornamento.

Parametri:



* **statistics:** JSON

Rappresenta il contenuto delle statistiche riguardanti l'esercitazione effettuata in un determinato argomento da utilizzare per aggiornare quelle esistenti;

* **callback:** function(JSON)

Rappresenta la callback che il metodo deve chiamare al termine dell'elaborazione.

- + **upLevel(callback : function(JSON)):** void

Questo metodo aggiorna il livello di difficoltà relativo alla capacità dell'utente a rispondere a determinate domande. Restituisce un oggetto $JSON_G$ che descrive il livello dell'utente dopo l'aggiornamento.

Parametri:

* **callback:** function(JSON)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione.

- + **deleteUser(callback : function(JSON), errback:function(QuizzipediaError))**

Questo metodo elimina dal sistema l'utente registrato. Restituisce un oggetto $JSON_G$ che descrive i dati dell'utente eliminato.

Parametri:

* **callback:** function(JSON)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione;

* **errback:** function(QuizzipediaError)

Questo parametro rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.

- + **updateSummary(summaryId: ObjectId)** Questo metodo aggiorna la cronologia dell'utente in relazione ai questionari svolti.

Parametri

* **summaryId:** ObjectId

Rappresenta il riferimento al riepilogo creato.

- + **getSummaries(callback : function(JSON), errback : function(QuizzipediaError)): void**

Questo metodo ritorna un $JSON_G$ contenente la cronologia dei questionari svolti da parte dell'utente, in caso di errori una $callback_G$ che segnalerà i relativi problemi.

Parametri:

* **callback:** function(JSON)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione;

* **errback:** function(QuizzipediaError)

Questo parametro rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.

- + **getUsers(searchword : String, callback : function(JSON), errback : function(QuizzipediaError)): void**

Questo metodo è statico e ritorna un $JSON_G$ contenente i dati degli utenti in base alla parola chiave cercata, in caso di errori una $callback_G$ che segnalerà i relativi problemi.

Parametri:

* **searchword:** String

Rappresenta la parola chiave cercata in base alla quale verrà filtrata la ricerca degli utenti;

* **callback:** function(JSON)

Rappresenta la $callback_G$ che il metodo deve chiamare al termine dell'elaborazione;

* **errback:** function(QuizzipediaError)

Questo parametro rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.



3.7.2.3 QuizziPedia::Back-End::App::Models::UserProModel

UserProModel
- UserProSchema : Schema
+ addUserPro(userId : ObjectId, callback : function(JSON), errback : function(JSON)) : void
+ deleteUserPro(userId : ObjectId, callback : function(JSON), errback : function(JSON)) : void

Figura 163: QuizziPedia::Back-End::App::Models::UserProModel

- **Descrizione:** classe che modella i dati dell’utente pro;
- **Utilizzo:** viene utilizzata per rappresentare i dati dell’utente pro. Si interfaccia alla libreria Mongoose per la creazione dello schema e dei relativi metodi statici o di istanza;
- **Relazioni con altre classi:**
 - **IN UserManagementController**
Classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica dei dati dell’utente. Rappresenta il ConcreteHandler nel design pattern Chain of responsibility. Utilizza Passport;
 - **IN QuizModel**
Questa classe rappresenta i dati dei questionari creati dagli utenti Pro;
 - **OUT UserModel**
Questa classe rappresenta i dati riguardanti i vari utenti registrati al sistema.
- **Attributi:**
 - **- UserProSchema: Schema**
Questo campo dati rappresenta lo schema Mongoose dell’utente pro QuizziPedia. Lo schema prevede i seguenti attributi:
 - * **userId: ObjectId**
Rappresenta il riferimento all’identificativo nel database contenente i dati dei vari utenti registrati.
- **Metodi:**
 - **+ addUserPro(userId : ObjectId, callback: function(JSON), errback: function(QuizzipediaError)) : void**
Questo metodo è statico ed aggiorna la tipologia di utenza. In specifico l’utente di tipo basic passerà a pro. Ritorna un JSON contenente i dati del nuovo utente pro, qualora si siano verificati problemi verrà ritornato un messaggio d’errore.
Parametri:
 - * **userId: ObjectId**
Rappresenta l’identificativo dell’utente da passare a pro;
 - * **callback: function(JSON)**
Rappresenta la callback che il metodo deve chiamare al termine dell’elaborazione;
 - * **errback: function(QuizzipediaError)**
Questo parametro rappresenta la callback che il metodo deve chiamare qualora si verificassero errori nell’esecuzione del metodo.



- + `deleteUserPro(userId : ObjectId, callback: function(JSON), errback: function(QuizzipediaError)) : void`

Questo metodo è statico ed aggiorna la tipologia di utenza. In specifico l'utente di tipo pro ritornerà a basic, eliminando il riferimento alla classe user . Ritorna un JSON contenente i dati dell'utente pro ritornato a basic, qualora si siano verificati problemi verrà ritornato un messaggio d'errore.

Parametri:

- * `userId: ObjectId`
Rappresenta l'identificativo dell'utente pro da ritornare a basic;
- * `callback: function(JSON)`
Rappresenta la callback che il metodo deve chiamare al termine dell'elaborazione;
- * `errback: function(QuizzipediaError)`
Questo parametro rappresenta la callback che il metodo deve chiamare qualora si verificassero errori nell'esecuzione del metodo.

3.7.2.4 QuizziPedia::Back-End::App::Models::QuestionModel

QuestionModel	
-	<code>questionSchema : Schema</code>
+	<code>createQuestion(question : JSON, callback : function(JSON), errback : function(QuizziPediaError)) : void</code>
+	<code>editQuestion(question : JSON, callback : function(JSON), errback : function(QuizziPediaError)) : void</code>
+	<code>updateLevel(userLevel : Number, isCorrected : boolean) : void</code>
+	<code>addKeyword(keyword : String, callback : function(JSON), errback : function(QuizziPediaError)) : void</code>
-	<code>addCorrect() : void</code>
-	<code>addTotal() : void</code>
+	<code>getQuestions(userID : ObjectId) : void</code>

Figura 164: QuizziPedia::Back-End::App::Models::QuestionModel

- **Descrizione:** classe che modella i dati relativi alle domande all'interno dell'applicazione;
- **Utilizzo:** viene utilizzata per rappresentare le domande. Si interfaccia alla libreria *MongooseG* per la creazione dello schema e dei relativi metodi statici o di istanza;
- **Relazione con altre classi:**
 - **IN UserModel**
Classe che rappresenta tutti gli utenti;
 - **OUT SummaryModel**
Classe che rappresenta i riepiloghi dei questionari svolti;
 - **OUT TopicModel**
Classe che rappresenta gli argomenti;
 - **OUT QuizModel**
Classe che modella i questionari all'interno dell'applicazione;
 - **IN QuestionController**
Classe che gestisce la logica applicativa riguardante la visualizzazione, la creazione e la modifica delle domande presenti nell'applicazione;
 - **OUT UserModel**
Classe che modella la creazione e la gestione dei dati utente;



– **IN QuizModel**

Classe che modella i questionari all'interno dell'applicazione;

– **OUT SummaryModel**

Classe che modella i riepiloghi all'interno dell'applicazione;

– **IN TopicModel**

Classe che modella gli argomenti all'interno delle domande.

• **Attributi:**

– **questionSchema: Schema**

Questo campo dati rappresenta lo schema *MongooseG* per le domande e prevede i seguenti attributi:

* **author: ObjectId**

Rappresenta il riferimento all'identificativo nel database dell'utente che ha creato la domanda;

* **makeWith: String**

Rappresenta con quale strumento è stata creata la domanda;

* **language: String**

Rappresenta la lingua in cui è scritta la domanda;

* **question: Array<Object>**

Array che contiene oggetti di tipo JSON. L'oggetto JSON è rappresentato dai seguenti campi:

· **type: String**

Rappresenta la tipologia di domanda;

· **questionText: String**

Rappresenta il testo della domanda;

· **image: String**

Rappresenta l'*URLG* dell'immagine associata al testo della domanda;

· **answers: Array<Object>**

Contiene oggetti di tipo JSON. L'oggetto JSON è rappresentato dai seguenti campi:

1. **text: String**

Rappresenta il testo della risposta;

2. **url: String**

Rappresenta l'immagine della risposta;

3. **attributesForTForMultiple: Mixed**

Contiene i seguenti attributi:

(a) **isItRight: Boolean**

Rappresenta se una risposta è giusta o sbagliata.

4. **attributesForSorting: Mixed**

Contiene i seguenti attributi:

(a) **position: Number**

Rappresenta quale è la giusta posizione di un testo o immagine all'interno di un esercizio di ordinamento.

5. **attributesForLinking: Mixed**

Contiene i seguenti attributi:

(a) **text1: String**

Rappresenta il primo elemento testuale che deve essere collegato con il secondo elemento testuale o rappresentato da un'immagine;



- (b) `text2: String`
Rappresenta il secondo elemento testuale che deve essere collegato con il primo elemento testuale o rappresentato da un'immagine;
 - (c) `url1: String`
Rappresenta il primo elemento rappresentato da un'immagine che deve essere collegato con il secondo elemento testuale o rappresentato da un'immagine;
 - (d) `url2: String`
Rappresenta il secondo elemento rappresentato da un'immagine che deve essere collegato con il primo elemento testuale o rappresentato da un'immagine.
6. `attributesForClickableArea: Mixed`
Contiene i seguenti attributi:
- (a) `x: Number`
Rappresenta la coordinata x di una area cliccabile;
 - (b) `y: Number`
Rappresenta la coordinata y di una area cliccabile.
7. `attributesForEmptySpaces: Mixed`
Contiene i seguenti attributi:
- (a) `wordNumber: Number`
Rappresenta la posizione dello spazio vuoto in cui deve andare inserita la parola.
- * `keywords: Array<String>`
Contiene oggetti di tipo `String` che rappresentano le parole chiave utili per ricercare una domanda;
- * `level: Number`
Rappresenta la difficoltà della domanda;
- * `totalAnswers: Number`
Rappresenta le risposte totali che tutti gli utenti hanno dato alla domanda;
- * `correctAnswers: Number`
Rappresenta quante risposte corrette hanno dato gli utenti che hanno risposto alla domanda.

• **Metodi:**

- + `createQuestion(question: JSON, callback: function(JSON), errback: function(QuizziPediaError)): void`
Metodo che permette la creazione di una nuova domanda.

Parametri:

- * `question: JSON`
Rappresenta le informazioni che andranno a comporre la domanda da creare;
- * `callback: function(JSON)`
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione nel caso non si verifichino errori durante l'esecuzione;
- * `errback: function(QuizziPediaError)`
Rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori durante l'esecuzione del metodo.

- + `editQuestion(question: JSON, callback: function(JSON), errback: function(QuizziPediaError)): void`
Metodo che permette di modificare una domanda già esistente.

Parametri:



* **question** : JSON
Rappresenta le nuove informazioni che andranno a modificare le informazioni precedenti di una specifica domanda;

* **callback**: function(JSON)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione del metodo in caso non si verifichino errori durante l'esecuzione;

* **errback**: function(QuizziPediaError)
Rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori durante l'esecuzione del metodo.

- + **addKeyword**(keyword: String, callback: function(JSON), errback: function(QuizziPediaError)): void
Metodo che permette di aggiungere delle parole chiave ad una specifica domanda.

Parametri:

* **keyword**: String
Rappresenta la parola chiave da inserire;

* **callback**: function(JSON)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione del metodo in caso si verifichino errori durante l'esecuzione;

* **errback**: function(QuizziPediaError)
Rappresenta la $callback_G$ che il metodo deve chiamare qualora si verificassero errori durante l'esecuzione del metodo.

- + **updateLevel**(userLevel: Number, isCorrected: boolean): void
Metodo che permette di aggiornare il livello di difficoltà della domanda, questo metodo è chiamato ogni qualvolta un utente risponde ad una domanda durante un allenamento.

Parametri:

* **userLevel**: Number
Indica, con un numero compreso tra 1 e 1000, l'abilità dell'utente che ha risposto alla domanda;

* **isCorrected**: boolean
Indica se l'utente che ha risposto alla domanda ha risposto correttamente.

- - **addCorrect()**: void
Metodo che permette di incrementare il contatore di risposte corrette di una determinata domanda;

- - **addTotal()**: void
Metodo che permette di incrementare il contatore delle risposte date di una determinata domanda;

- + **getQuestions**(userID: ObjectId): Array<QuestionModel>
Metodo che ritorna tutte le domande create da un utente.

Parametri:

* **userID**: ObjectId
Rappresenta l'identificativo dell'utente del quale si vogliono ritornare le domande.

3.7.2.5 QuizziPedia::Back-End::App::Models::QuizModel



QuizModel
- quizSchema : Schema
+ createQuiz(info : JSON, callback : function(JSON), errback : function(QuizziPediaError))
+ addQuestion(questionId : ObjectId) : void
+ removeQuestion(questionId : ObjectId) : void
+ setTitle(title : String, errback : function(QuizziPediaError)) : void
+ addUser(userId : ObjectId) : void
+ addActiveUser(userId : ObjectId) : void
+ addCorrect() : void
+ removeUser(userId : ObjectId, errback : function(QuizziPediaError)) : void
+ searchQuiz(searchword : String, callback : function(JSON), errback : function(QuizziPediaError)) : void
+ getPersonalQuizzes(userId : ObjectId, callback : function(JSON), errback : function(QuizziPediaError)) : void
+ getQuiz(userId : ObjectId, callback : function(JSON), errback : function(QuizziPediaError)) : void

Figura 165: QuizziPedia::Back-End::App::Models::QuizModel

- **Descrizione:** classe che modella i questionari all'interno dell'applicazione;
- **Utilizzo:** viene utilizzata per rappresentare i dati relativi ai questionari all'interno dell'applicazione. Si interfaccia con la libreria *MongooseG* per la creazione dello schema e dei relativi metodi statici o di istanza;
- **Relazioni con altre classi:**
 - **OUT QuestionModel**
Classe che rappresenta le operazioni relative alle domande;
 - **OUT UserModel**
Classe che rappresenta le operazioni relative agli utenti.
- **Attributi:**
 - - **quizSchema: Schema**
Questo campo dati rappresenta lo schema *MongooseG* dei quiz di QuizziPedia. Lo schema prevede i seguenti attributi:
 - * **title: String**
Rappresenta il titolo del questionario;
 - * **author: ObjectId**
Rappresenta il riferimento all'identificativo nel database dell'utente che ha creato il questionario;
 - * **questions: Array<ObjectId>**
Array che contiene oggetti di tipo **ObjectId** che rappresentano i riferimenti agli identificativi nel database delle domande appartenenti al questionario;
 - * **registeredUsers: Array**
Array che contiene oggetti di tipo **ObjectId** che rappresentano i riferimenti agli identificativi nel database degli utenti iscritti al quiz;
 - * **activeUsers: Array**
Array che contiene oggetti di tipo **ObjectId** che rappresentano i riferimenti agli identificativi nel database degli utenti che partecipano effettivamente al questionario;
 - * **correctAnswers: Number**
Rappresenta il numero totale di risposte corrette date dagli utenti alle domande del questionario.
- **Metodi:**



- + `createQuiz(info: JSON, callback: function(JSON), errback: function(QuizziPediaError)): void`
Crea un questionario con i dati che vengono passati.
Parametri:
 - * `info: JSON`
Rappresenta i dati del questionario che verrà creato;
 - * `callback: function(JSON)`
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione;
 - * `errback: function(QuizziPediaError)`
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione in caso di errore.
- + `addQuestion(questionID: ObjectId): void`
Aggiunge una domanda al questionario.
Parametri:
 - * `questionID: ObjectId`
Rappresenta l'identificativo della domanda da aggiungere al questionario.
- + `removeQuestion(questionID: ObjectId): void`
Rimuove una domanda dal questionario.
Parametri:
 - * `questionID: ObjectId`
Rappresenta l'identificativo della domanda da rimuovere dal questionario.
- + `setTitle(title: String, errback: function(QuizziPediaError)): void`
Da un titolo al questionario.
Parametri:
 - * `title: String`
Rappresenta il titolo da dare al questionario;
 - * `errback: function(QuizziPediaError)`
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione in caso di errore.
- + `addUser(userID: ObjectId): void`
Aggiunge un utente dalla lista degli iscritti al questionario.
Parametri:
 - * `userID: ObjectId`
Rappresenta l'identificativo dell'utente da aggiungere al questionario.
- + `addActiveUser(userID: ObjectId): void`
Aggiunge un utente dalla lista degli utenti che hanno svolto il questionario.
Parametri:
 - * `userID: ObjectId`
Rappresenta l'identificativo dell'utente da aggiungere al questionario.
- + `addCorrect(): void`
Incrementa il numero di risposte corrette date alle domande del questionario;
- + `removeUser(userID: ObjectId, errback: function(QuizziPediaError)): void`
Rimuovere un utente dalla lista degli iscritti al questionario.
Parametri:
 - * `userID: ObjectId`
Rappresenta l'identificativo dell'utente da rimuovere dalla lista degli iscritti al questionario;



```
* errback: function(QuizziPediaError)
Rappresenta la  $callback_G$  che verrà eseguita al termine dell'elaborazione in caso di errore.
```

- + searchQuiz(searchword: String, callback: function(JSON), errback: function(QuizziPediaError)): void
Ricerca un questionario.

Parametri:

- * **searchword:** String
Rappresenta il titolo o l'autore del questionario da ricercare;
- * **callback:** function(JSON)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione;
- * **errback:** function(QuizziPediaError)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione in caso di errore.

- + getPersonalQuizzes(userID: ObjectId, callback: function(JSON), errback: function(QuizziPediaError)): void
Ritorna i questionari creati da un utente pro.

Parametri:

- * **userID:** ObjectId
Rappresenta l'identificativo dell'utente che ha creato i questionari che si vogliono ottenere;
- * **callback:** function(JSON)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione;
- * **errback:** function(QuizziPediaError)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione in caso di errore.

- + getQuiz(callback: function(JSON), errback: function(QuizziPediaError)): void
Ritorna il questionario da compilare.

Parametri:

- * **callback:** function(JSON)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione;
- * **errback:** function(QuizziPediaError)
Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione in caso di errore.

3.7.2.6 QuizziPedia::Back-End::App::Models::TopicModel

TopicModel
- topicSchema : Schema
+ updateCorrect(): void
+ updateTotal(): void
+ addQuestion(questionID : ObjectId): void
+ getQuestions(language : String, keyword : Array<String>, callback : function(JSON), errorback : function(QuizziPediaError)) : void
+ getNextQuestion(language : String, levelUser : Number, callback : function(JSON), errback : function(QuizziPediaError)) : void

Figura 166: QuizziPedia::Back-End::App::Models::TopicModel

- **Descrizione:** classe che modella gli argomenti all'interno delle domande;



- **Utilizzo:** viene utilizzata per rappresentare i dati relativi agli argomenti delle domande. Si interfaccia con la libreria *Mongoose_G* per la creazione dello schema e dei relativi metodi statici o di istanza;

- **Relazioni con altre classi:**

- **IN TopicController**

Classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica degli argomenti delle domande;

- **OUT QuestionModel**

Questa classe rappresenta i dati delle domande create dai vari utenti.

- **Attributi:**

- **- topicSchema: Schema**

Questo campo dati rappresenta lo schema *Mongoose_G* per gli argomenti e prevede i seguenti attributi:

- * **name: String**

Rappresenta il nome dell'argomento;

- * **correctAnswers: Number**

Rappresenta il numero totale di domande alle quali gli utenti hanno risposto correttamente durante un allenamento sull'argomento;

- * **totalAnswers: Number**

Rappresenta il numero totale di domande alle quali gli utenti hanno risposto durante un allenamento sull'argomento;

- * **question: Array**

Array che contiene gli **ObjectId** delle domande sull'argomento.

- **Metodi:**

- **+ updateCorrect() : void**

Metodo che consente di tenere aggiornato il numero di risposte esatte date a domande sull'argomento;

- **+ updateTotal() : void**

Metodo che consente di tenere aggiornato il numero totale di risposte date a domande sull'argomento;

- **+ addQuestion(questionId : ObjectId) : void**

Metodo che consente di aggiunge una domanda tra le domande sull'argomento.

Parametri:

- * **questionId : ObjectId**

Rappresenta l'identificativo della domanda da aggiungere.

- **+ getQuestions(language : String, keyword : [String],**

callback : function(JSON), errback : function(QuizziPediaError))

Metodo che consente di ottenere le domande sull'argomento attraverso la funzione di *callback_G* oppure un messaggio di errore.

Parametri:

- * **language : String**

Rappresenta la lingua in cui sono scritte le domande che si vuole ottenere;

- * **keyword : Array<String>**

Rappresenta le keyword che possono essere assegnate alle domande che si vuole ottenere;



```
* callback : function(JSON)
Rappresenta la callbackG che il metodo deve chiamare al termine dell'elaborazione
nel caso in cui non si siano verificati errori;
* errback : function(QuizziPediaError)
Rappresenta la callbackG che il metodo deve chiamare qualora si verificassero
errori durante l'esecuzione del metodo.

- + getNextQuestion(language : String, levelUser : Number,
callback : function(JSON), errback : function(QuizziPediaError))::
void
Metodo che consente di ottenere la domanda successiva nella modalità allenamento,
in base al livello dell'utente che lo sta svolgendo.
```

Parametri:

- * language : String
Rappresenta la lingua in cui è scritta la domanda che si vuole ottenere;
- * levelUser : Number
Rappresenta il livello dell'utente che sta svolgendo l'allenamento;
- * callback : function(JSON)
Rappresenta la *callback_G* che il metodo deve chiamare al termine dell'elaborazione
nel caso in cui non si siano verificati errori;
- * errback : function(QuizziPediaError)
Rappresenta la *callback_G* che il metodo deve chiamare qualora si verificassero
errori durante l'esecuzione del metodo.

3.7.2.7 QuizziPedia::Back-End::App::Models::SummaryModel

SummaryModel
- summarySchema : Schema
+ createSummary(quiz : JSON) : void

Figura 167: QuizziPedia::Back-End::App::Models::summaryModel

- **Descrizione:** classe che modella i riepiloghi all'interno dell'applicazione;
- **Utilizzo:** viene utilizzata per rappresentare i dati relativi ai riepiloghi all'interno dell'applicazione. Si interfaccia con la libreria *Mongoose_G* per la creazione dello schema e dei relativi metodi statici o di istanza;
- **Relazioni con altre classi:**
 - **OUT QuizModel**
Classe che modella i questionari;
 - **OUT QuestionModel**
Classe che modella le domande;
 - **IN UserModel**
Classe che modella gli utenti;
 - **IN SummaryController**
Classe che gestisce le operazioni relative ai riepiloghi.



- **Attributi:**

- **summarySchema:** Schema

Questo campo dati rappresenta lo schema $Mongoose_G$ dei riepiloghi di QuizziPedia.

Lo schema prevede i seguenti attributi:

- * **quiz:** ObjectId

Rappresenta il riferimento all'identificativo nel database del quiz;

- * **givenAnswers:** Array<ObjectId>

Array che contiene oggetti di tipo ObjectId che rappresentano i riferimenti agli identificativi nel database delle domande a cui si è risposto e un Array contenenti oggetti di tipo JSON;

- * **date:** Date

Rappresenta la data di creazione del riepilogo;

- * **mark:** Number

Rappresenta il voto conseguito nel quiz.

- **Metodi:**

- + **createSummary(quiz: JSON): void**

Crea un riepilogo.

Parametri:

- * **quiz:** JSON

Rappresenta il quiz dalle cui risposte verrà creato il riepilogo.

3.7.2.8 QuizziPedia::Back-End::App::Models::LangModel

LangModel
- langSchema : Schema
+ getVarlist(lang : String, callback : function(JSON), errback : function(QuizziPediaError)) : void

Figura 168: QuizziPedia::Back-End::App::Models::LangModel

- **Descrizione:** classe che modella le informazioni riguardanti la lingua dell'applicazione;
- **Utilizzo:** viene utilizzata per scambiare memorizzare le traduzioni delle variabili che andranno visualizzate nella $view_G$;
- **Relazioni con altre classi:**
 - **OUT LangController**
Classe che gestisce la logica applicativa riguardante la traduzione delle variabili.
- **Attributi:**
 - **LangSchema :** Schema
Questo campo rappresenta lo schema $mongoose_G$ per le variabili della lingua e prevede i seguenti attributi:
* **lang:** String, rappresenta la lingua scelta per l'applicazione;



* **variables:** `Array<String>`
 Array associativo che contiene delle `String` necessarie per la giusta traduzione dell'applicazione.

- **Metodi:**

– + `getVarlist(lang: String, callback: function(JSON), errback: function(QuizziPediaError)): void`
 Metodo che permette di ritornare la traduzione delle variabili.

Parametri:

* `lang: String`
 Rappresenta l'informazione che indica il set di traduzione da ritornare;
 * `callback: function(JSON)`
 Rappresenta la $callback_G$ che verrà eseguita al termine dell'elaborazione nel caso non si verifichino errori durante l'esecuzione;
 * `errback: function(QuizziPediaError)`
 Rappresenta la $callback_G$ che verrà eseguito al termine dell'elaborazione nel caso si verifichino errori durante l'esecuzione del metodo.

3.7.2.9 QuizziPedia::Back-End::App::Models::ModelError

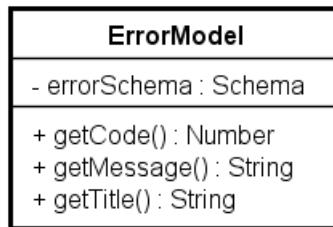


Figura 169: QuizziPedia::Back-End::App::Models::ModelError

- **Descrizione:** classe che rappresenta le informazioni di un errore che si è verificato eseguendo una determinata operazione;
- **Utilizzo:** viene utilizzata per racchiudere tutte le informazioni riguardanti l'errore;
- **Relazioni con altre classi:**

– IN **ErrorHandler**

Classe $middleware_G$ per la gestione degli errori. Ritorna al $client_G$ un oggetto di tipo `Response` con stato $HTTP_G$ 500 e descrizione dell'errore in formato $JSON_G$. È un componente $ConcreteHandler_G$ del $design pattern_G$ *Chain of responsibility*.

- **Attributi:**

– - `errorSchema: Schema`

Questo campo dati rappresenta lo schema $Mongoose_G$ per gli errori e prevede i seguenti attributi:

* `errorCode: Number`

Rappresenta il codice dell'errore;

* `errorMessage: String`

Rappresenta la descrizione dell'errore;



* `errorTitle` : `String`
Rappresenta il titolo del messaggio d'errore.

- **Metodi:**

- + `getCode()` : `Number`
Metodo che consente di ottenere il codice dell'errore;
- + `getMessage()` : `String`
Metodo che consente di ottenere la descrizione dell'errore;
- + `Title()` : `String`
Metodo che consente di ottenere il titolo del messaggio d'errore.

3.8 QuizziPedia::Back-End::App::Routers

3.8.1 Informazioni generali

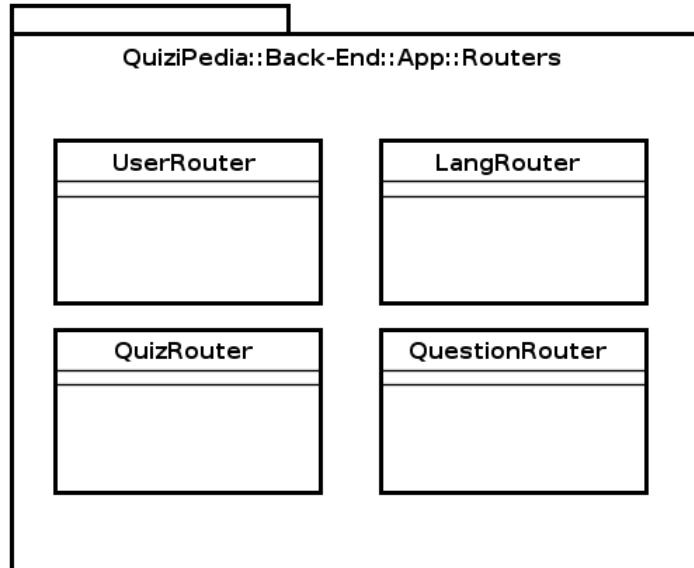


Figura 170: QuizziPedia::Back-End::App::Routers

- **Descrizione:** $package_G$ contenente i router della componente back-end dell'applicazione. Contiene i file di configurazione relativi al routing delle richieste del $client_G$, ossia i *routers* di $Express_G$;
- **Padre:** App;
- **Interazioni con altri componenti:**
 - **Controllers:** $package_G$ che contiene i *Controllers* di $Express_G$, definisce la logica dell'applicazione.

3.8.2 Classi

3.8.2.1 QuizziPedia::Back-End::App::Routers::UserRouter

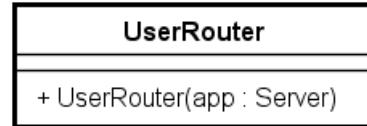


Figura 171: QuizziPedia::Back-End::App::Routers::UserRouter

- **Descrizione:** classe che gestisce le richieste relative alla registrazione, alla gestione della sessione e alla cronologia dei questionari svolti di un utente. Componente ConcreteHandler del design pattern *Chain of responsibility_G*. Utilizza il modulo *Passport_G*;
- **Utilizzo:** viene utilizzata per chiamare il *controller_G* che si occupa di gestire le *API_G* relative alla registrazione, alla gestione della sessione e alla cronologia dei questionari di un utente;
- **Relazioni con altre classi:**
 - **IN Server:** classe che avvia il server. Nello specifico apre una connessione al database tramite *Mongoose_G*, invoca il *middleware_G* *Express_G* passando un riferimento al database *MongoDB_G* come parametro in modo che possa configurarsi con esso, invoca il *middleware_G* *Passport_G* ed infine si mette in ascolto su una determinata porta. È il componente client del design pattern *Chain of responsibility_G*. Utilizza i moduli *Mongoose_G*, *Express_G*, *Passport_G*;
 - **OUT ErrorHandler:** classe *middleware_G* per la gestione degli errori. Ritorna al client un oggetto di tipo Response con stato *HTTP_G* 500 e descrizione dell'errore in formato *JSON_G*. È un componente ConcreteHandler del design pattern *Chain of responsibility_G*;
 - **OUT NotFoundHandler:** classe che si occupa della gestione dell'errore di pagina non trovata. Componente ConcreteHandler del design pattern *Chain of responsibility_G*;
 - **OUT UserController:** classe che raggruppa attraverso require i vari *controllers_G* responsabili delle operazioni legate alla gestione degli utenti. Si è scelto di predisporre questo raggruppamento per facilitare l'introduzione di nuove funzionalità legate alla gestione degli utenti;
 - **OUT SummaryController:** classe che gestisce la cronologia dei questionari svolti dall'utente.
- **Metodi:**
 - + UserRouter(app: Server)

Contiene diverse *route_G* che vengono configurate all'avvio del server. Quest'ultime ricevono le richieste del client e passano il controllo al ConcreteHandler successivo.

Parametri:

 - * app: Server

Rappresenta l'istanza del server su cui configurare i *route_G* che mappano i *controllers_G* specifici.

3.8.2.2 QuizziPedia::Back-End::App::Routers::QuestionRouter

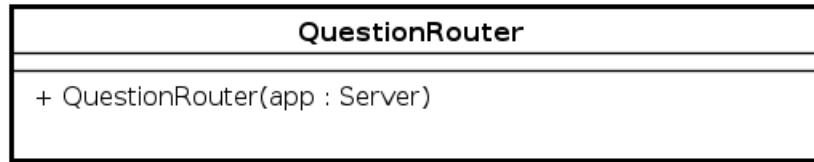


Figura 172: QuizziPedia::Back-End::App::Routers::QuestionRouter

- **Descrizione:** classe che gestisce le richieste relative alle operazioni riguardanti le domande. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
- **Utilizzo:** viene utilizzata per chiamare il *controller_G* che si occupa di gestire le *API_G* relative alle domande;
- **Relazioni con altre classi:**
 - **IN Server:** classe che avvia il *server_G*. Nello specifico apre una connessione al database tramite *Mongoose_G*, invoca il *middleware_G Express_G* passando un riferimento al database *MongoDB_G* come parametro in modo che possa configurarsi con esso, invoca il *middleware_G Passport_G* ed infine si mette in ascolto su una determinata porta. È il componente *client_G* del *design pattern_G Chain of responsibility_G*. Utilizza i moduli *Mongoose_G*, *Express_G* e *Passport_G*;
 - **OUT ErrorsHandler:** classe *middleware_G* per la gestione degli errori. Ritorna al *client_G* un oggetto di tipo **Response** con stato HTTP 500 e descrizione dell'errore in formato JSON. È un componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **OUT NotFoundHandler:** classe che si occupa della gestione dell'errore di pagina non trovata. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **OUT QuestionController:** classe che raggruppa i vari *controllers* responsabili delle operazioni riguardanti le domande attraverso **require**;
 - **OUT TopicController:** classe che gestisce la logica applicativa riguardante la visualizzazione e la modifica degli argomenti delle domande. È un componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*.
- **Metodi:**
 - + QuestionRouter(app: Server)

Contiene diverse *route_G* che vengono configurate all'avvio del *server_G*. Quest'ultime ricevono le richieste del *client_G* e passano il controllo al *ConcreteHandler_G* successivo.

Parametri:

 - * app: Server

Rappresenta l'istanza del *server_G* su cui configurare i *route_G* che mappano i *controllers_G* specifici.

3.8.2.3 QuizziPedia::Back-End::App::Routers::QuizRouter

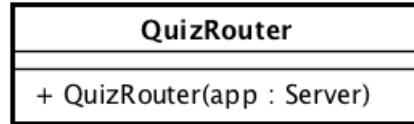


Figura 173: QuizziPedia::Back-End::App::Routers::QuizRouter

- **Descrizione:** classe che gestisce le richieste relative alle operazioni riguardanti un questionario. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
- **Utilizzo:** viene utilizzata per chiamare il *controller_G* che si occupa di gestire le *API_G* relative ad un questionario;
- **Relazioni con altre classi:**
 - **IN Server:** classe che avvia il *server_G*. Nello specifico apre una connessione al database tramite *Mongoose_G*, invoca il *middleware_G Express_G* passando un riferimento al database *MongoDB_G* come parametro in modo che possa configurarsi con esso, invoca il *middleware_G Passport_G* ed infine si mette in ascolto su una determinata porta. È il componente *client_G* del *design pattern_G Chain of responsibility_G*. Utilizza i moduli *Mongoose_G*, *Express_G*, *Passport_G*;
 - **OUT ErrorHandler:** classe *middleware_G* per la gestione degli errori. Ritorna al *client_G* un oggetto di tipo *Response* con stato *HTTP_G 500* e descrizione dell'errore in formato *JSON_G*. È un componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **OUT NotFoundHandler:** classe che si occupa della gestione dell'errore di pagina non trovata. Componente *ConcreteHandler_G* del *design pattern_G Chain of responsibility_G*;
 - **OUT QuizController:** classe che raggruppa i vari *controllers_G* responsabili delle operazioni riguardanti un questionario attraverso require.
- **Metodi:**
 - + QuizRouter(app: Server)

Contiene diverse *route_G* che vengono configurate all'avvio del *server_G*. Quest'ultime ricevono le richieste del *client_G* e passano il controllo al *ConcreteHandler_G* successivo.

Parametri:

 - * app : Server

Rappresenta l'istanza del *server_G* su cui configurare i *route_G* che mappano i *controllers_G* specifici.

3.8.2.4 QuizziPedia::Back-End::App::Routers::LangRouter



Figura 174: QuizziPedia::Back-End::App::Routers::LangRouter

- **Descrizione:** classe che gestisce le richieste relative alla lingua;



- **Utilizzo:** viene utilizzata per chiamare il $controller_G$ che si occupa di cambiare la lingua dell'applicazione;
- **Relazioni con altre classi:**

- **IN Server:** classe che avvia il $server_G$. Nello specifico apre una connessione al database tramite $Mongoose_G$, invoca il $middleware_G$ $Express_G$ passando un riferimento al database $MongoDB_G$ come parametro in modo che possa configurarsi con esso, invoca il $middleware_G$ $Passport_G$ ed infine si mette in ascolto su una determinata porta; è il componente $client_G$ del pattern $Chain\ of\ responsibility_G$. Utilizza i moduli $Mongoose_G$, $Express_G$, $Passport_G$;
- **OUT NotFoundHandler:** classe che si occupa della gestione dell'errore di una pagina non trovata. Componente $ConcreteHandler_G$ del $design\ pattern_G$ $Chain\ of\ responsibility_G$;
- **OUT LangController:** classe che gestisce la logica applicativa riguardante il passaggio della traduzione delle variabili.

- **Metodi:**

- **+ LangRouter(app: Server)**

Contiene diverse $ruote_G$ che vengono configurate all'avvio del $server_G$. Queste ultime ricevono le richieste del $client_G$ e passano il controllo al $ConcreteHandler_G$ successivo.

Parametri:

- * **app: Server**

Rappresenta l'istanza del server su cui configurare i $ruote_G$ che mappano i $controllers_G$ specifici.



4 Diagrammi di sequenza

4.1 Front-End

Nei seguenti diagrammi di sequenza vengono illustrate le interazioni tra le classi del Front-End. Nei diagrammi di sequenza viene sfruttato il servizio offerto da *AngularG*, tale $\$promise_G$. Quest'ultimo mette a disposizione delle funzioni anonime come **success** e **fail** che saranno indicate nei seguenti diagrammi anche se non compaiono tra i metodi dei *controllersG*.

4.1.1 Principali operazioni Front-End

4.1.1.1 Autenticazione

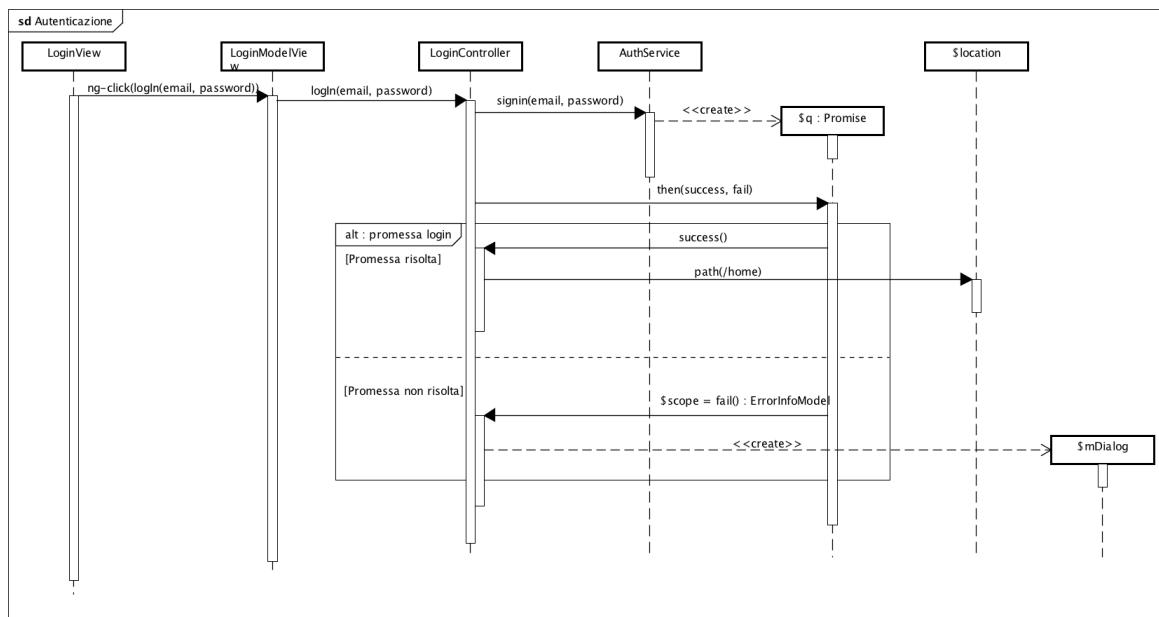


Figura 175: Autenticazione

L'utente, dopo aver inserito i campi email e password, può effettuare l'autenticazione avviando l'evento associato al bottone presente in `LoginView`. Il `LoginController` gestisce l'evento chiamando il metodo `signin` dell'`AuthService`, il quale restituisce una promessa. Se la promessa è soddisfatta l'utente viene reindirizzato alla home page, altrimenti verrà restituito un oggetto di tipo `ErrorInfoModel` e mostrato a video mediante `$mdDialog`.

4.1.1.2 Registrazione

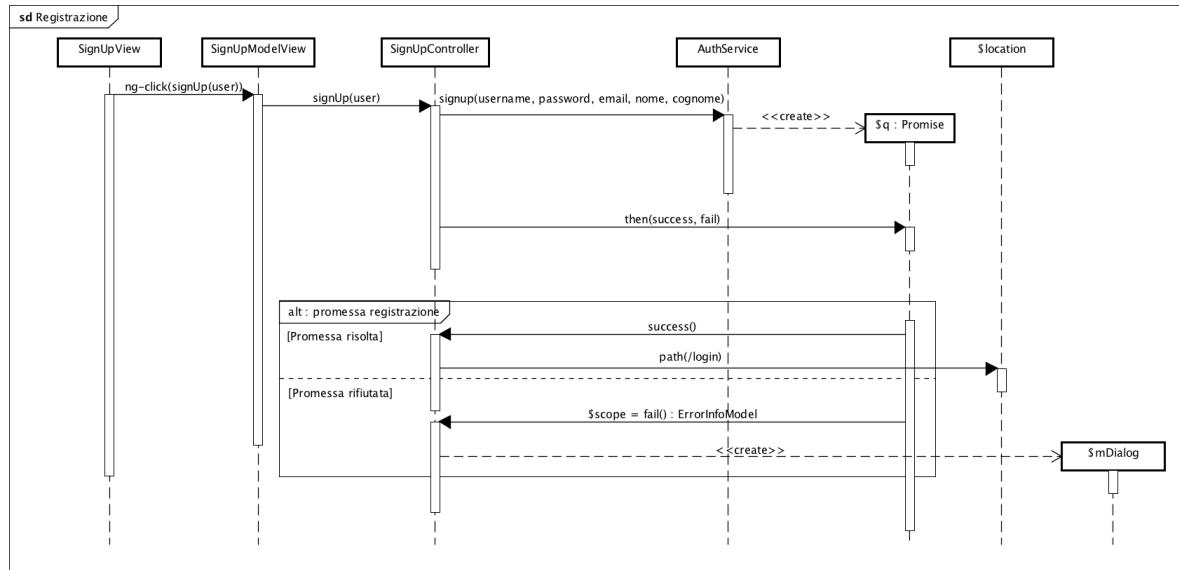


Figura 176: Registrazione

L'utente, dopo aver inserito tutti i campi richiesti, può effettuare la registrazione avviando l'evento associato al bottone presente in `SignUpView`. Il `SignUpController` gestisce l'evento chiamando il metodo `signup` dell'`AuthService`, il quale restituisce una promessa. Se la promessa viene soddisfatta l'utente viene reindirizzato alla pagina di login, altrimenti verrà restituito un oggetto di tipo `ErrorInfoModel` e mostrato a video mediante `$mdDialog`.

4.1.1.3 Recupero password

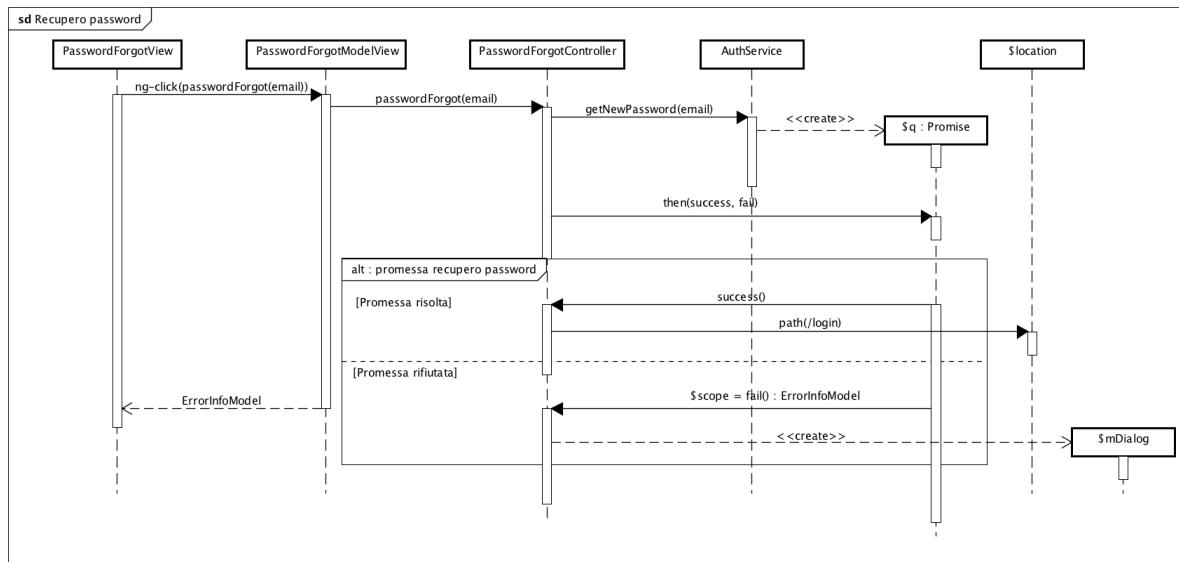


Figura 177: Recupero password

L'utente, dopo aver inserito la propria email, può recuperare una nuova password avviando l'evento associato al bottone presente in `PasswordForgotView`. Il `PasswordForgotController` gestisce l'evento chiamando il metodo `getNewPassword` dell'`AuthService`, il quale restituirà una promessa. Se la promessa viene soddisfatta allora verrà mandata una email con la nuova pas-



sword e verrà reindirizzato alla pagina di login, altrimenti verrà restituito un oggetto di tipo `ErrorInfoModel` e mostrato a video mediante `$mdDialog`.

4.1.1.4 Ricerca

4.1.1.4.1 Ricerca

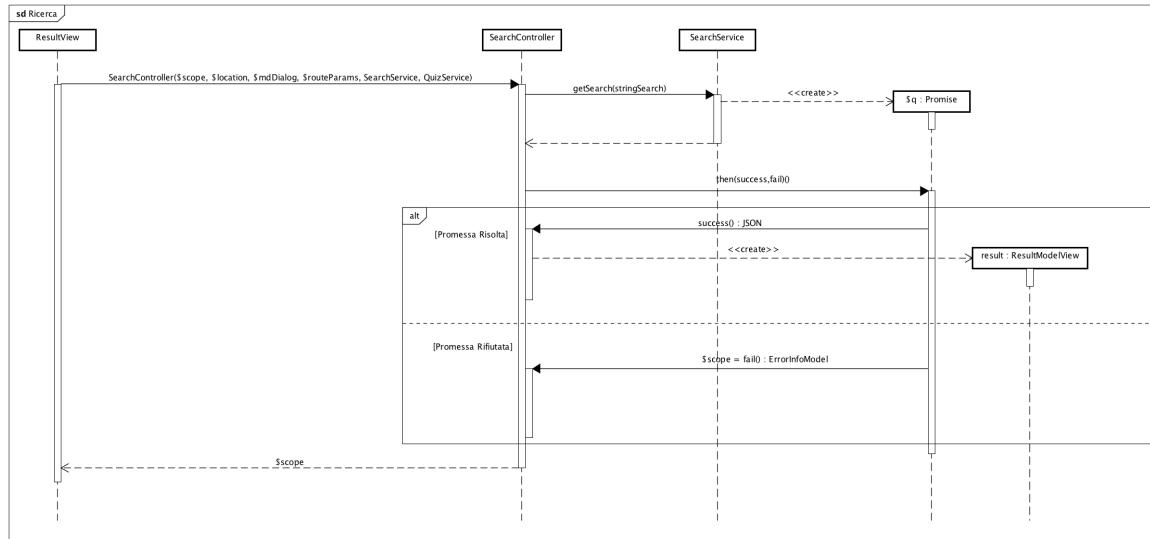


Figura 178: Ricerca

L'utente, dopo aver effettuato una ricerca, viene indirizzato alla pagina di ricerca. Una volta posizionato in questa pagina, `AngularG` provvederà a costruire la classe `SearchController`. Durante la creazione di tale classe viene eseguita la ricerca mediante `SearchService`. Questo service, restituisce una promessa, se questa verrà rispettata verrà popolato l'attributo `result` altrimenti verrà restituito un oggetto di tipo `ErrorInfoModel` e mostrato a video mediante `$mdDialog`.

4.1.1.4.2 Iscrizione ad un questionario

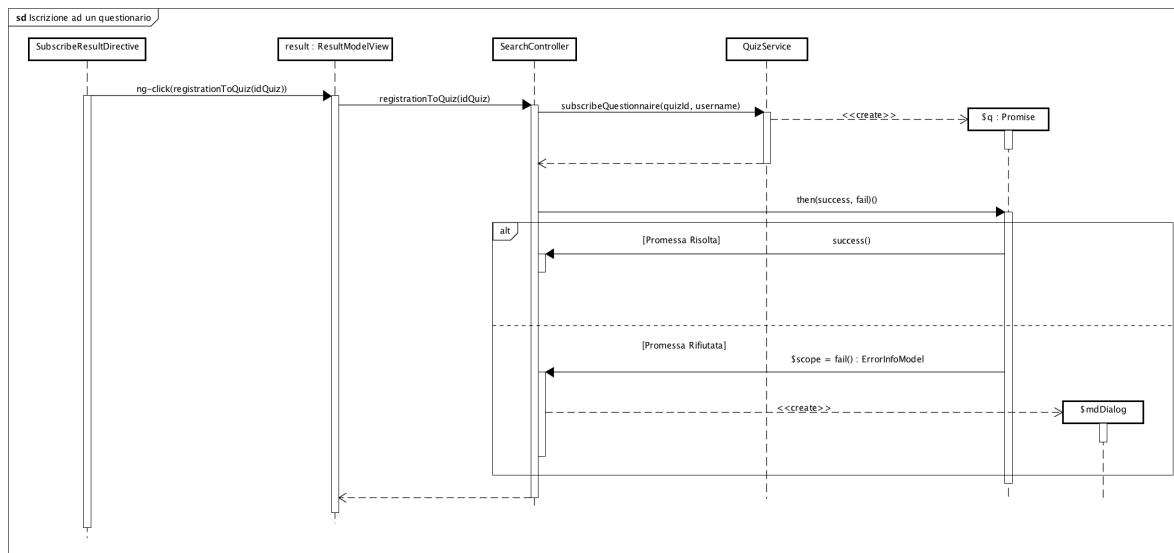


Figura 179: Iscrizione ad un questionario



L'utente, posizionato dentro la pagina di ricerca, potrà iscriversi ad un questionario azionando l'evento associato al bottone di iscrizione al questionario. Attraverso **SearchController** viene eseguita l'iscrizione mediante **QuizService**. Questo *service_G* restituisce una promessa, se accadrà un errore verrà restituito un oggetto di tipo **ErrorInfoModel** e mostrato a video mediante **\$mdDialog**.

4.1.1.4.3 Redirect alla pagina di un utente

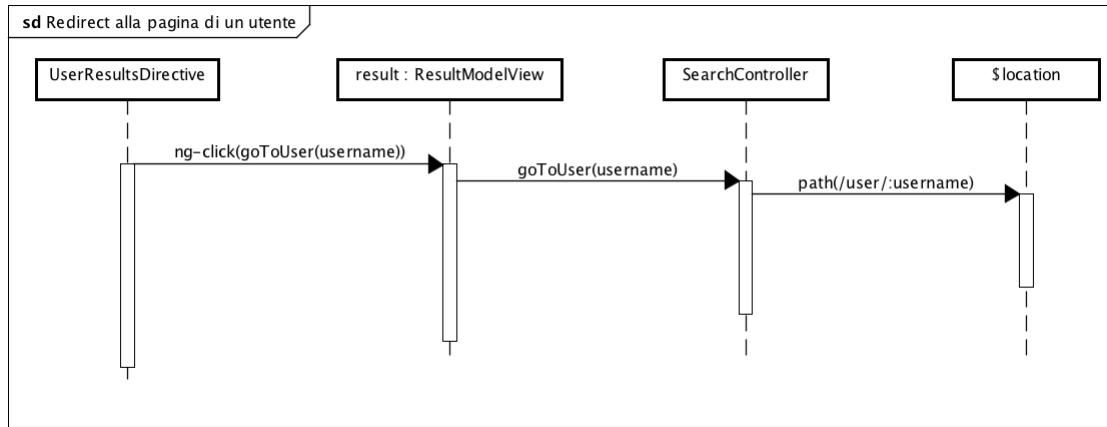


Figura 180: Redirect alla pagina di un utente

L'utente, posizionato dentro la pagina di ricerca, potrà decidere di visualizzare la pagina del profilo di altro utente azionando l'evento **ng-click** associato al bottone posizionato all'interno della *directive_G* **UserDetailDirective**.

4.1.1.5 Visualizzazione pagina profilo

4.1.1.5.1 Pagina profilo personale

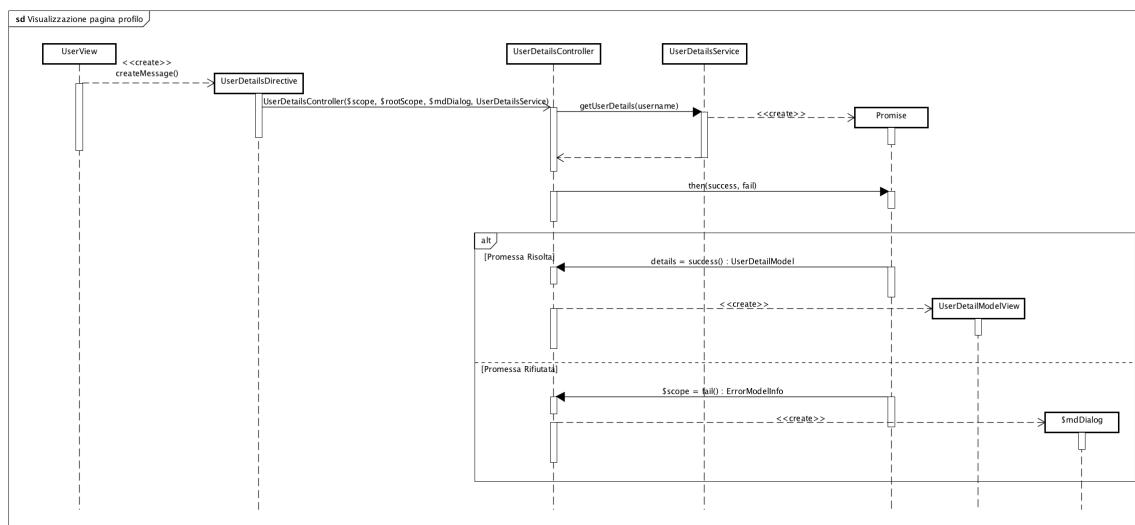


Figura 181: Pagina profilo personale

Con tre chiamate asincrone dai corrispettivi *controller_G*, le *directive_G* **StatisticsDirective**, **UserDetailsDirective**, **QuestionnaireDetailsDirective** e



`QuestionnaireDetailsDoneDirective` vengono popolate e mostrate all'utente. In questo diagramma di sequenza viene mostrato solamente il processo di creazione e visualizzazione della directive `UserDetailsDirective` poiché il procedimento per tutte le altre $directive_G$ è simile.

4.1.1.5.2 Pagina profilo altri utenti

La sequenza di operazioni necessaria per visualizzare una pagina profilo di un altro utente è del tutto simile a quella del profilo personale, ad eccezione fatta per la sezione dei questionari.

4.1.1.6 Gestione profilo utente

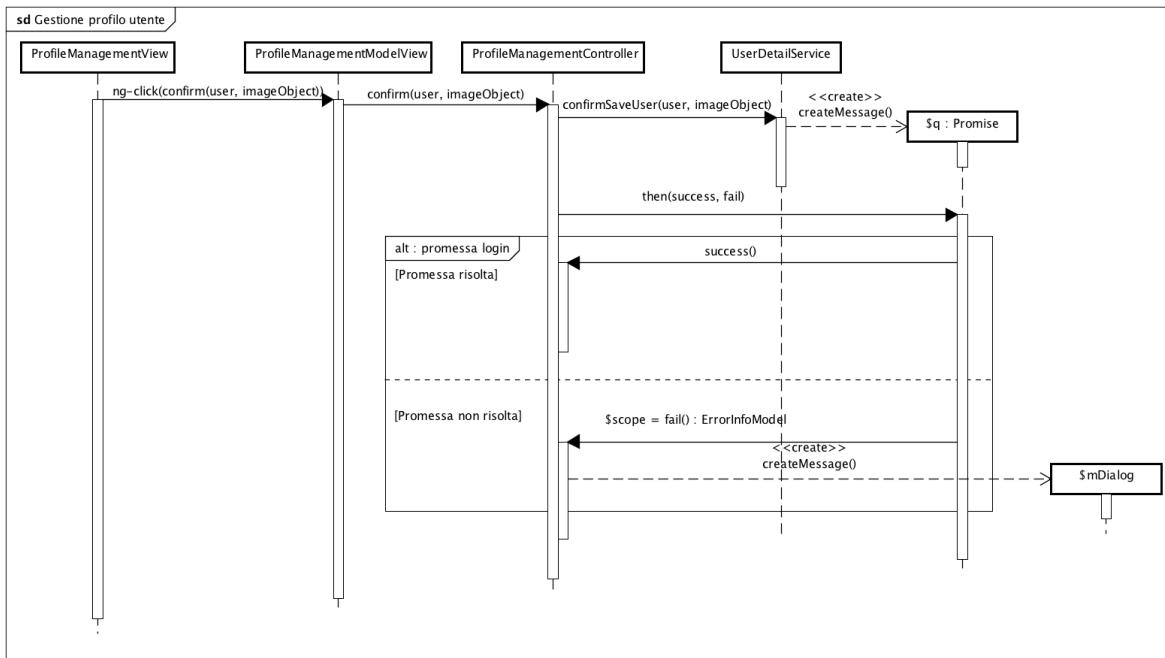


Figura 182: Gestione profilo utente

L'utente può modificare i suoi dati personali cambiando le informazioni dai campi dati presenti in `ProfileManagementView` e rendere persistenti tali modifiche avviando l'evento associato al bottone presente in tale $view_G$. Il `ProfileManagementController` gestisce l'evento chiamando il metodo `confirm` dell'`AuthService`, il quale restituirà una conferma. Se la promessa viene soddisfatta vengono rese persistenti le modifiche dei dati, altrimenti verrà restituito un oggetto di tipo `ErrorInfoModel` e mostrato a video mediante `$mdDialog`.

4.1.1.7 Gestione Domande

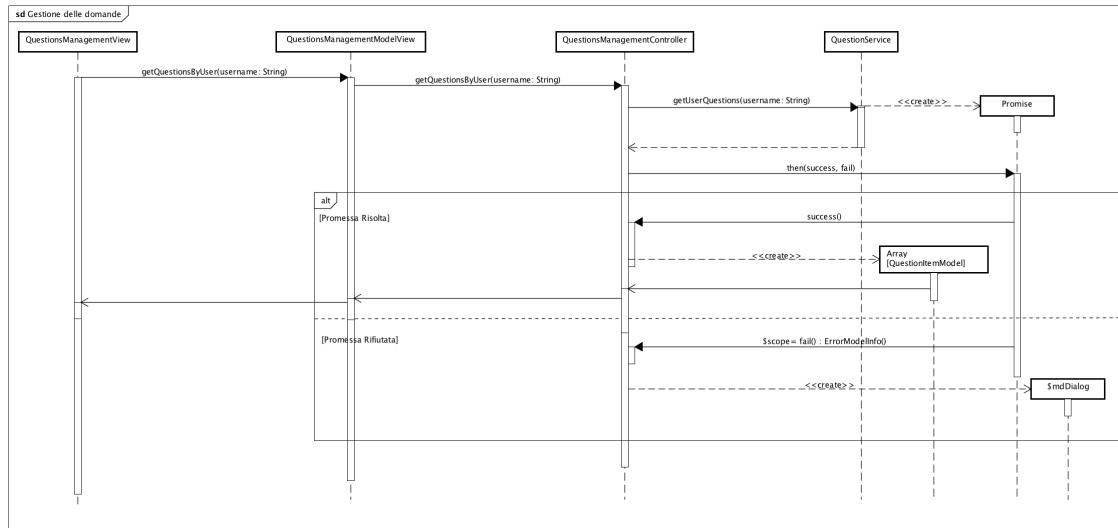


Figura 183: Gestione delle domande create da un utente

Nella view di visualizzazione domande verrà chiamato il metodo del *controller_G* che serve per ottenere tutte le domande create dall'utente autenticato. Il controller eseguirà una richiesta al *service_G*. A sua volta il *service_G* ritornerà una promessa che potrà essere risolta o rifiutata. Nel caso venga risolta verrà ritornato l'array di domande ottenuto dal back-end, nel caso invece venga rifiutata verrà restituito un oggetto contenente l'errore e visualizzato un messaggio informativo.

4.1.1.8 Creazione Domande

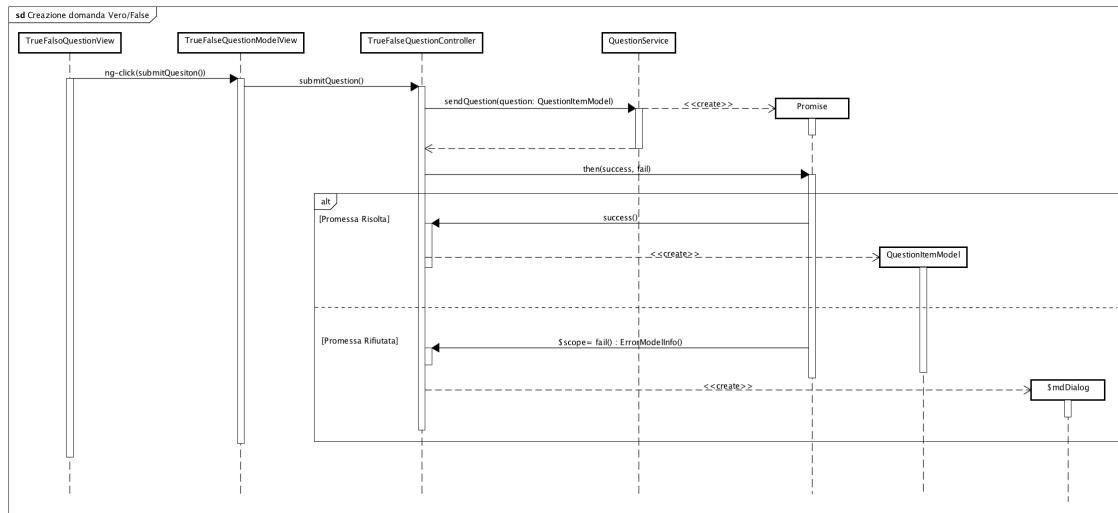


Figura 184: Creazione domanda vero/falso

Dopo che l'utente avrà selezionato il link di inserimento domanda, verrà richiamato il metodo del *controller_G* che richiederà al *service_G* l'inserimento della domanda tramite chiamata a risorse del back-end. Se la promessa creata dal *service_G* verrà accettata la domanda sarà inserita, altrimenti verrà restituito un oggetto contenente il testo dell'errore. L'operazione sarà uguale per tutte le tipologie di domanda.



4.1.1.9 Modalità allenamento

4.1.1.9.1 Selezione parametri di set-up

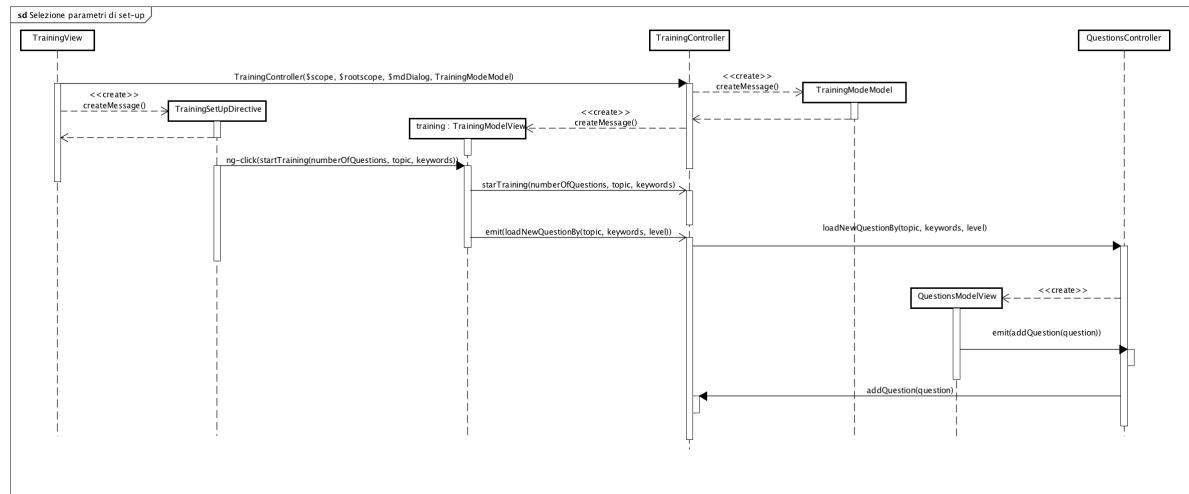


Figura 185: Selezione parametri di set-up

Dopo che l'utente avrà selezionato i parametri per eseguire l'allenamento, cioè l'argomento, le parole chiave e il numero di domande l'allenamento avrà inizio. Se il numero di domande non viene indicato l'allenamento andrà avanti ad oltranza.

4.1.1.9.2 Download di una domanda

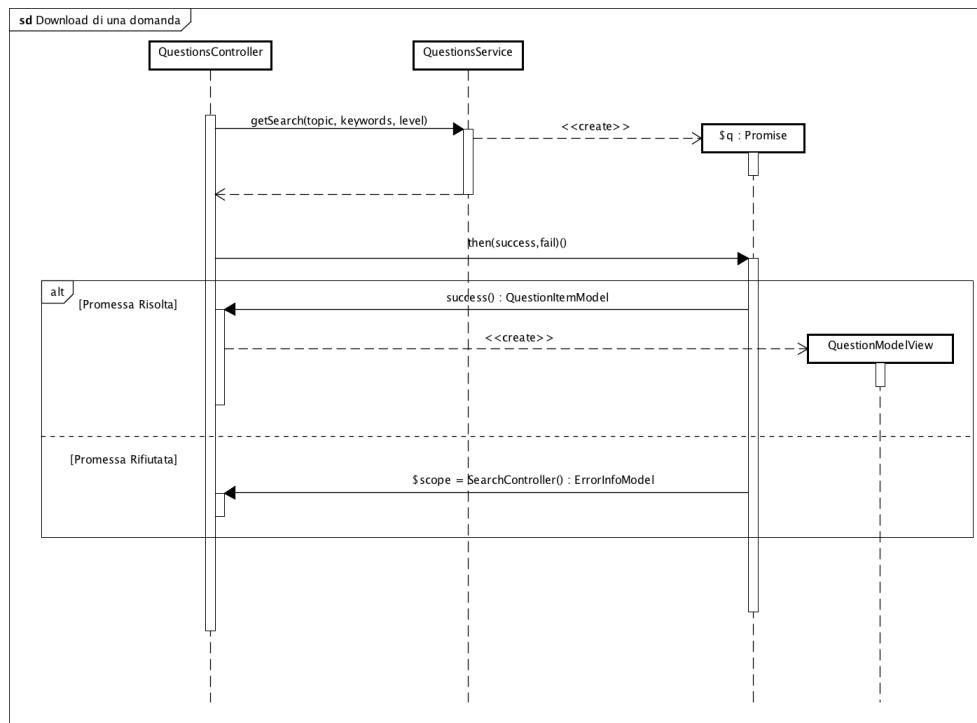


Figura 186: Download di una domanda



Il **QuestionsService** restituisce a **QuestionsController** una promessa, se questa verrà rispettata verrà ritornato un oggetto di tipo **QuestionItemModel** altrimenti verrà restituito un oggetto di tipo **ErrorInfoModel** e mostrato a video mediante **\$mdDialog**.

4.1.1.9.3 Composizione di una domanda

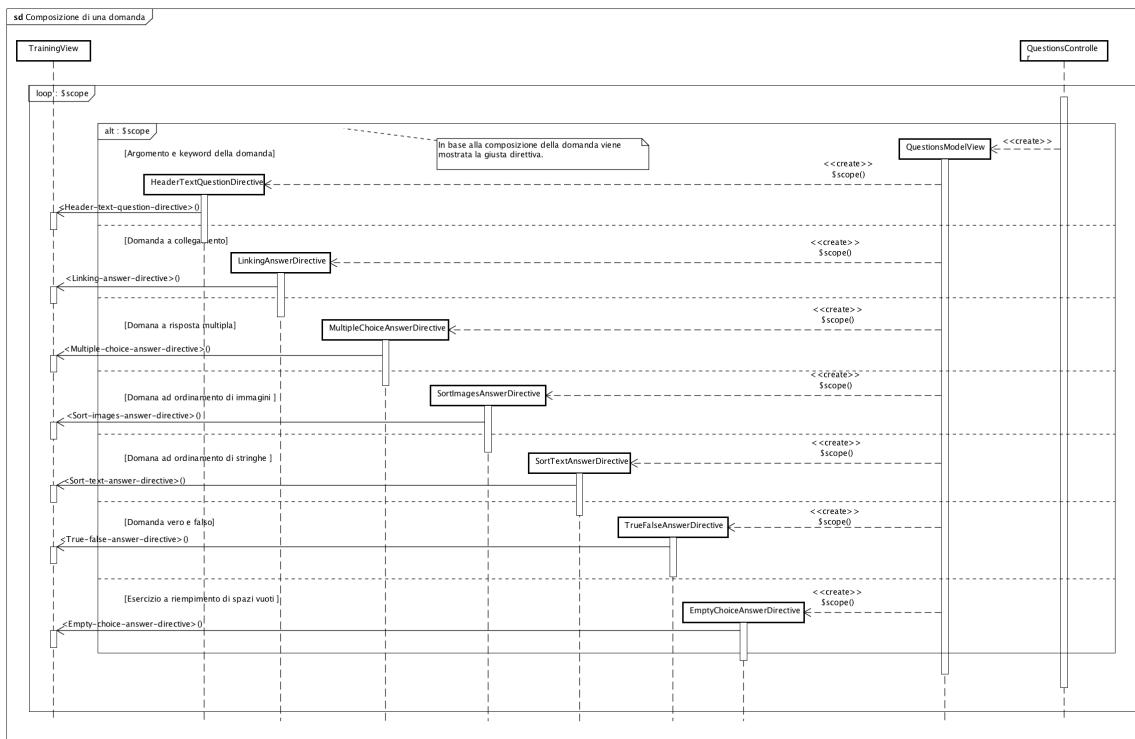


Figura 187: Composizione di una domanda

Eseguendo un loop sulla variabile di tipo **QuestionsModelView** precisamente sull'array **piecesOfQuestion**, per ogni pezzo di domanda verrà visualizzata la giusta direttiva così da comporre la domanda nella sua totalità.

4.1.1.9.4 Risposta ad una domanda

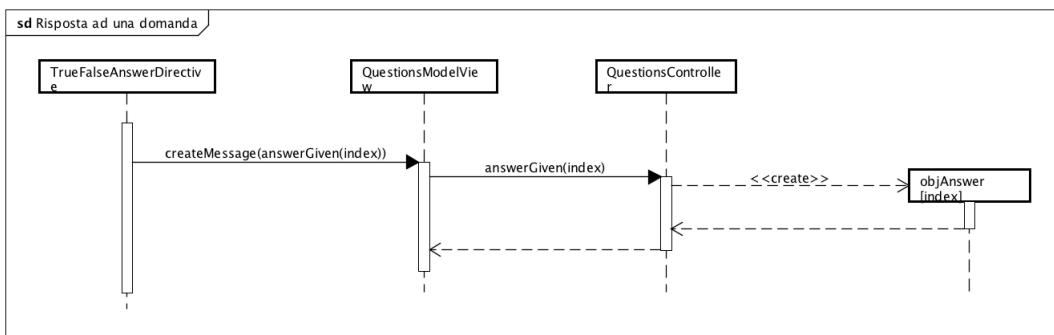


Figura 188: Risposta ad una domanda

In questo diagramma di sequenza viene mostrato come avviene la selezione di una risposta vero/falso. Il procedimento è uguale per tutte le tipologie di domande. Ad ogni evento **ng-change** o



`ng-click` viene aggiornato l'oggetto **objAnswer** nella giusta posizione. Alla conclusione della risposta l'oggetto **objAnswer** conterrà tutte le risposte date per quella domanda.

4.1.1.10 Compilazione questionario

4.1.1.10.1 Inizio questionario

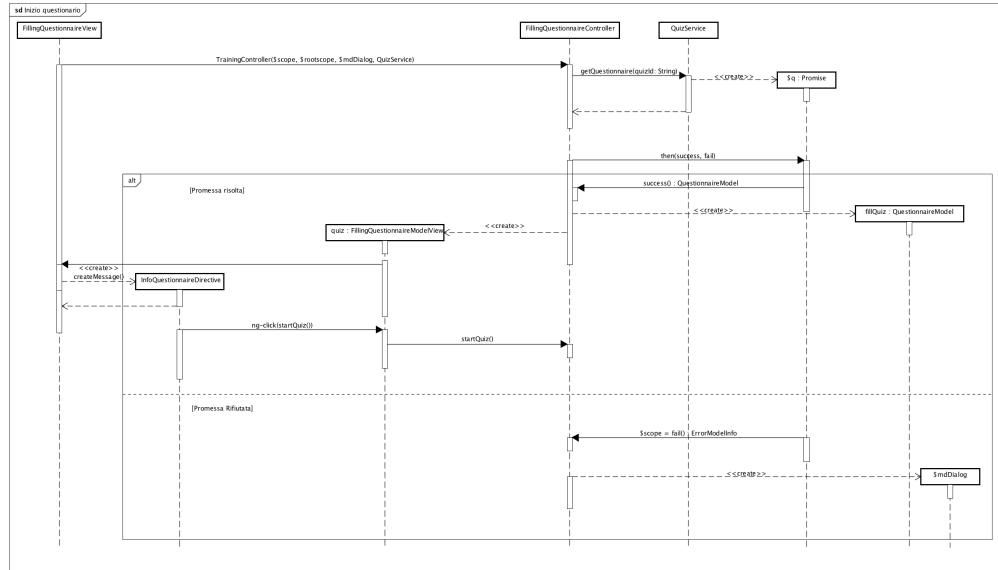


Figura 189: Inizio questionario

Al caricamento della pagina di compilazione di un questionario, il `FillingQuestionnaireController` attraverso il `QuestionnaireService` recupera dal back-end un oggetto di tipo `QuestionnaireModel`. Il `serviceG` citato esegue questa operazione mediante l'utilizzo delle `promiseG`. Se questa verrà risolta popolerà l'oggetto presente nello `$scope` di tipo `FillingQuestionnaireModelView`, così da mostrare nella `viewG` associata la pagina di inizio del questionario. Altrimenti verrà restituito un oggetto di tipo `ErrorInfoModel` che verrà visualizzato a video mediante un popup `$mdDialog`.

L'utente mediante l'evento `ng-click` potrà dare inizio al questionario.

4.1.1.10.2 Composizione di una domanda

Per comporre una domanda verrà eseguita la stessa sequenza di operazioni che sono presenti nella modalità allenamento. Eseguendo un loop sulla variabile di tipo `QuestionsModelView` precisamente sull'array `piecesOfQuestion`, per ogni pezzo di domanda verrà visualizzata la giusta direttiva così da comporre la domanda nella sua totalità.

4.1.1.10.3 Risposta ad una domanda

Per rispondere ad una domanda verrà eseguita la stessa sequenza di operazioni che sono presenti nella modalità allenamento.

4.1.1.11 Gestione Questionari

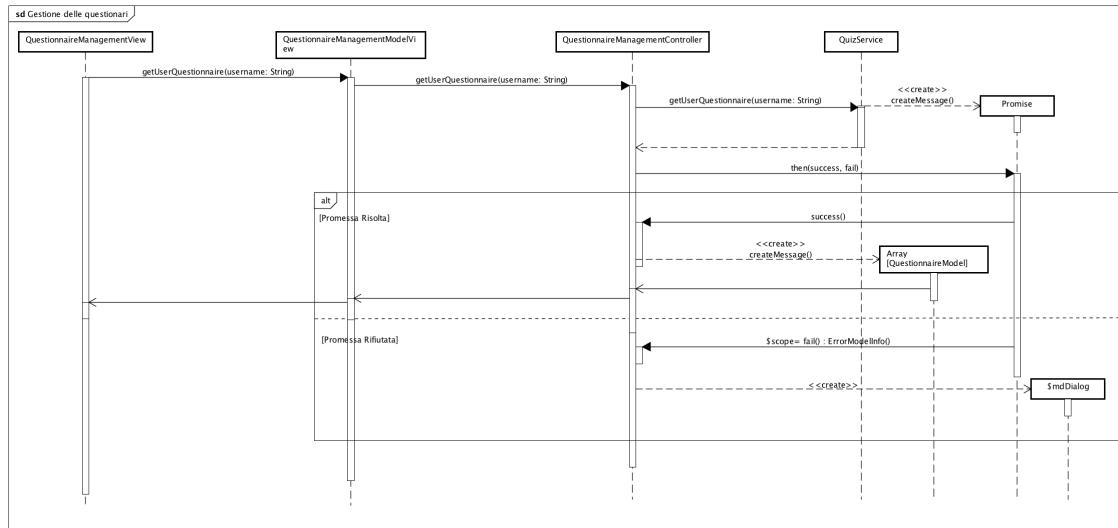


Figura 190: Recupero dei questionari di un utente

Nella view di visualizzazione domande verrà chiamato il metodo del **controller_G** che serve per ottenere tutti questionari create dall'utente autenticato. Il **controller_G** eseguirà una richiesta al **service_G**. A sua volta il **service_G** ritornerà una promessa che potrà essere risolta o rifiutata. Nel caso venga risolta verrà ritornato l'array di tutti i questionari, nel caso invece venga rifiutata verrà restituito un oggetto contenente l'errore e visualizzato un messaggio informativo.

4.1.1.12 Creazione Questionario

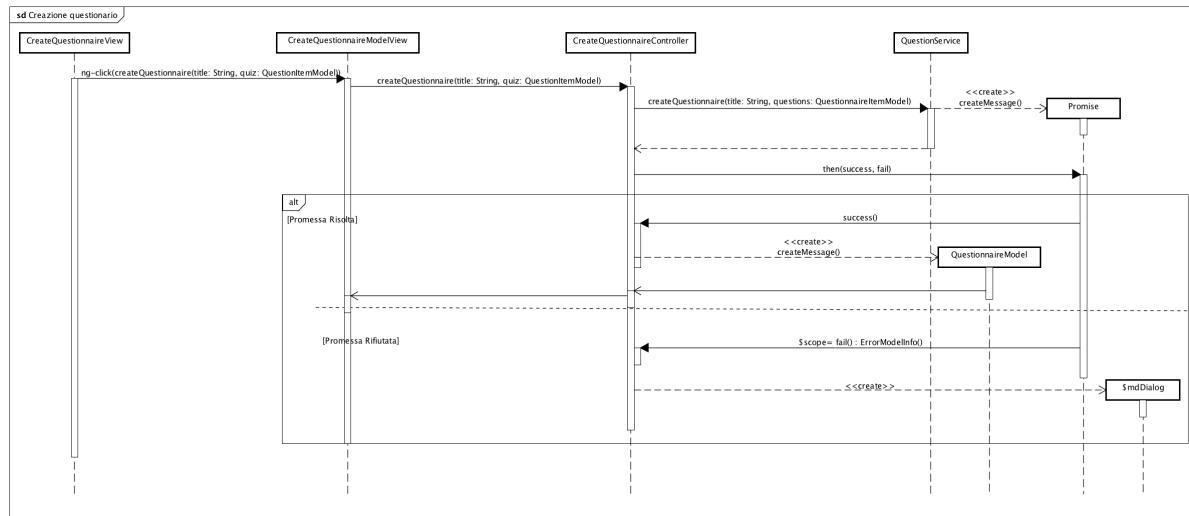


Figura 191: Creazione questionario

Dopo che l'utente avrà selezionato il link di creazione questionario, verrà richiamato il metodo del **controller_G** che richiederà al **service_G** l'inserimento del questionario tramite chiamata a risorse del back-end. Se la promessa creata dal **service_G** verrà accettata il questionario sarà inserito, altrimenti verrà restituito un oggetto contenente il testo dell'errore.

4.1.1.13 Gestione iscrizioni ad un questionario

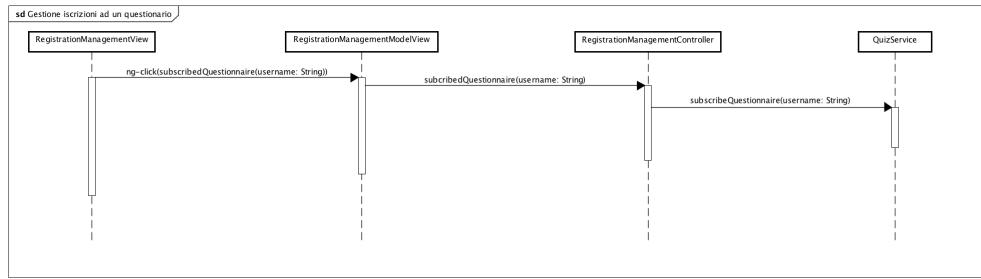


Figura 192: Recupero risultati del questionario

Al momento del dell'utente per confermare un iscrizione viene richiamato il metodo del **controller** che, a sua volta, richiama il metodo del **service** per eseguire la richiesta al backend. Se la promessa ritornata dal **service** viene accettata allora verrà effettuata l'approvazione, altrimenti verrà restituito un oggetto contenente l'errore.

4.1.1.14 Gestione questionari

4.1.1.14.1 Gestione iscrizioni del questionario

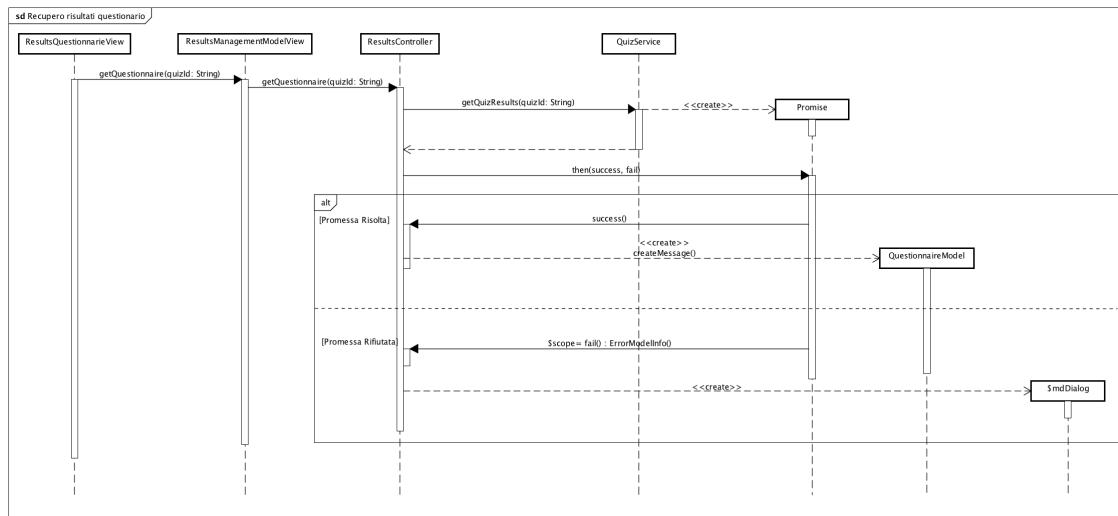


Figura 193: Recupero risultati del questionario

Al momento del caricamento della pagina viene richiamato il metodo del **controller** che, a sua volta, richiama il metodo del **service** per ottenere i risultati complessivi del questionario. Se la promessa ritornata dal **service** viene accettata allora verranno ricevuti i dati, altrimenti verrà restituito un oggetto contenente l'errore.

4.1.1.15 Recupero utenti del questionario

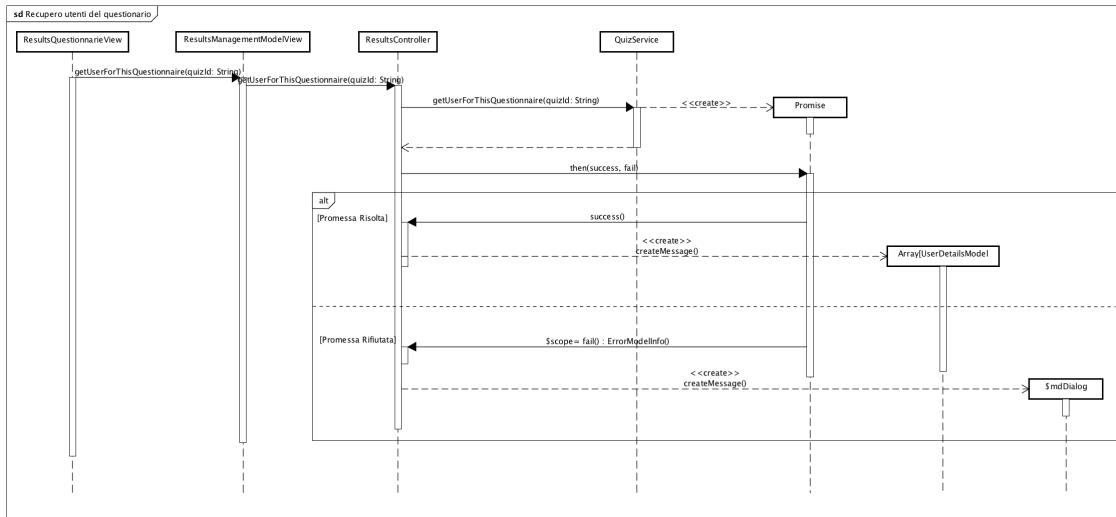


Figura 194: Recupero singoli utenti del questionario

Al momento del caricamento della pagina viene richiamato il metodo del **controller_G** che, a sua volta, richiama il metodo del **service_G** per ottenere i singoli utenti del questionario. Se la promessa ritornata dal **service_G** viene accettata allora verranno ricevuti i dati, altrimenti verrà restituito un oggetto contenente l'errore.

4.2 Back-End

Nei diagrammi di sequenza viene illustrata l'interazione tra il *server_G* ed i vari *middleware_G*, alcuni vengono forniti direttamente da *express_G* mentre altri sono stati definiti dai *progettisti*. Un *middleware_G* può terminare la propria esecuzione in tre modi diversi:

1. Terminata la propria esecuzione normalmente, eseguendo la richiesta passata dal *middleware_G* precedente, e quindi terminata la catena;
2. Esegue una *callback_G* passatagli come parametro, in questo modo passa il controllo ai successivi *middleware_G*;
3. Se si verifica un errore allora esegue la *callback_G* passando come parametro l'errore verificatosi. In questo modo la *callback_G* passerà il controllo al prossimo *middleware_G* della catena che è in grado di gestire l'errore.

4.2.1 Gestione generale delle richieste

Segue un elenco ordinato dei *middleware_G* utilizzati, l'ordine di elaborazione della richiesta è determinante poiché ciascuno costituisce un *handler_G* del *design pattern_G* *Chain of responsibility_G* ed ha la facoltà di interrompere la catena del chiamante:

- **express.compress()**: *middleware_G* per comprimere con il formato *gzip* le comunicazioni;
- **express.logger()**: *middleware_G* utilizzato per registrare un log delle richieste, utile per fare il *debugging_G* dell'applicazione;
- **express.json()**: *middleware_G* che estrae dalla richiesta i parametri che sono nel formato *JSON_G*;
- **express.urlencoded()**: *middleware_G* che estrae dalla richiesta i parametri di tipo *www-form-encoded*, questi possono essere inviati ad esempio con una richiesta POST;



- `express.methodOverride()`: *middleware_G* utilizzato per permettere anche ai vecchi *browser_G* di avere un modo per fare richieste PUT e DELETE;
- `express.cookieParser()`: *middleware_G* che analizza i *cookie_G*;
- `express.cookieSession()`: *middleware_G* che permette di memorizzare i record della sessione utente per mantenerne lo stato di login;
- `express.ruote()`: *middleware_G* che permette di configurare un singolo *ruote_G*, nell'organizzazione interna `QuizziPedia::Back-End::App::Routers` ogni modulo router contiene i routers logicamente correlati;
- `AuthenticationController`: *middleware_G* scritto dai *progettisti* per gestire l'autenticazione. Utilizza nello specifico:
 - `passport.authenticate()`.
- `NotFoundHandler`: *middleware_G* scritto dai *progettisti* per gestire le richieste che non vengono gestite da nessun *controller_G* (errore *client_G* 404);
- `ErrorHandler`: *middleware_G* scritto dai *progettisti* per gestire gli errori sollevati da altri *middleware_G* (errore *server_G* 500).

Nel seguente *diagramma di sequenza* viene rappresentata una generica richiesta del *client_G* al *server_G*. Le classi ed il loro comportamento sono state progettate in funzione del *design pattern_G* *Chain of responsibility_G* che viene utilizzato internamente da *express_G* (§A.3.1).

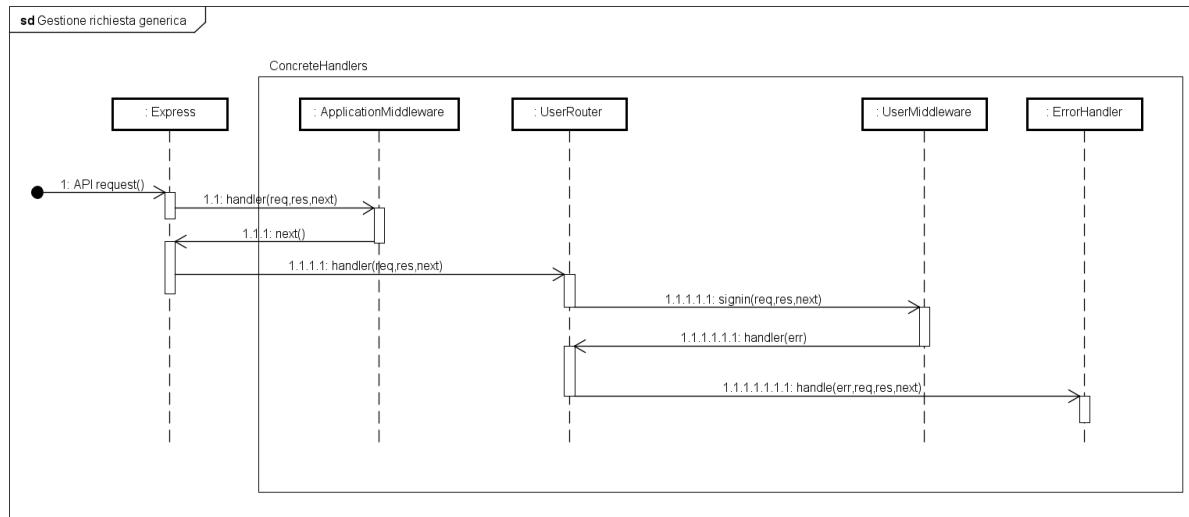


Figura 195: Gestione di una richiesta generica

4.2.2 Richieste REST

4.2.2.1 GET /:lang

- **Successo**: questo scenario rappresenta il successo di una richiesta di keywords inerenti alla lingua impostata che impone, come vincolo per poter essere effettuata, che l'utente non sia autenticato e non possieda già un account nel sistema.

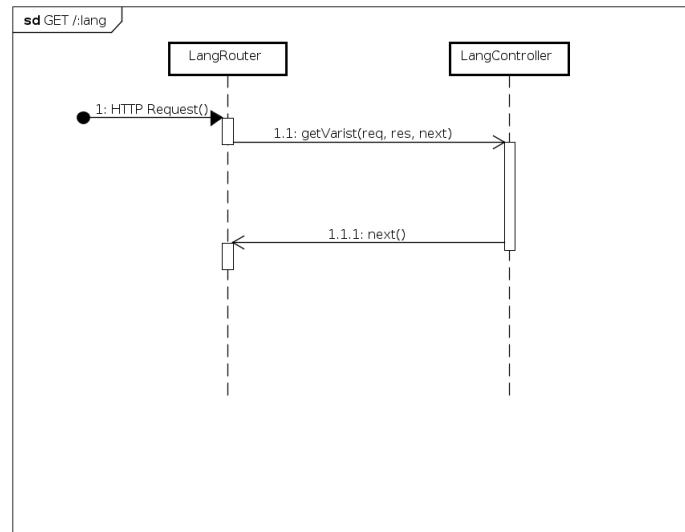


Figura 196: Procedura di traduzione

- **Fallimento:** quando viene effettuata una richiesta di traduzione in una lingua non riconosciuta dal sistema viene sollevato un errore. Tale scenario rappresenta il fallimento di una richiesta di traduzione che impone, come vincolo per poter essere effettuata, che l'utente non sia autenticato e non possieda già un account nel sistema. In questo caso il modulo **LangController** invia **next(error)** per il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

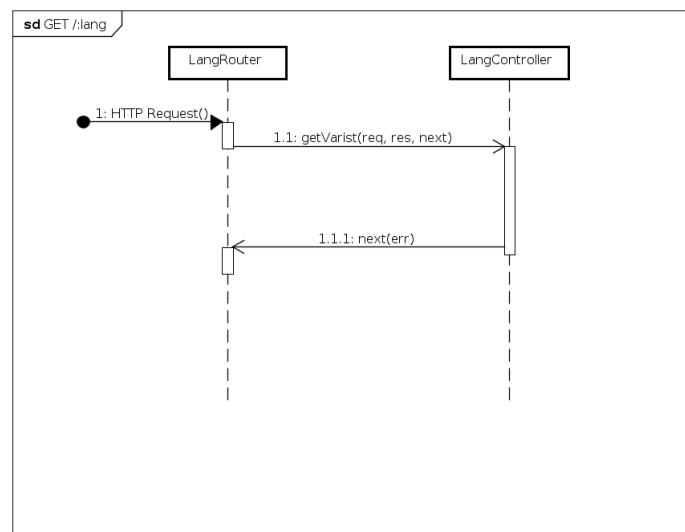


Figura 197: Fallimento della procedura di traduzione

4.2.2.2 POST /:lang/signup

- **Successo:** questo scenario rappresenta il successo di una richiesta di registrazione che impone, come vincolo per poter essere effettuata, che l'utente non sia autenticato e non possieda già un account nel sistema. La registrazione dell'utente nel sistema viene effettuata tramite *Passport_G* creando ed inserendo un nuovo *document* all'interno della collection User.

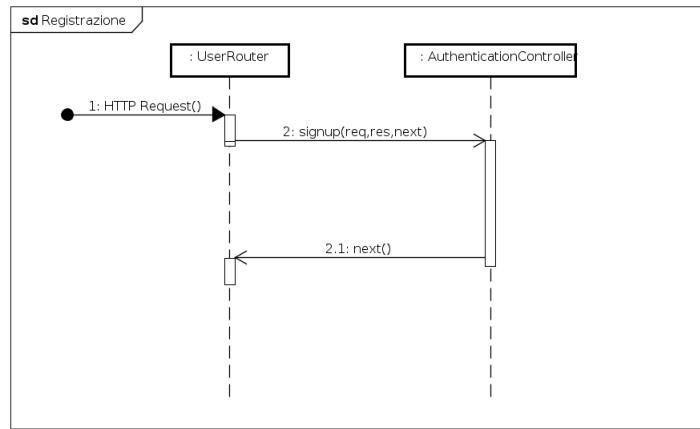


Figura 198: Procedura di registrazione

- **Fallimento:** quando un utente effettua una richiesta di registrazione e cerca di inserire dei dati già presenti del $database_G$ viene sollevato un errore. Tale scenario rappresenta il fallimento di una richiesta di registrazione che impone, come vincolo per poter essere effettuata, che l'utente non sia autenticato e non possieda già un account nel sistema. In questo caso il modulo `AuthenticationController` invia `next(error)` per il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso `ErrorHandler`).

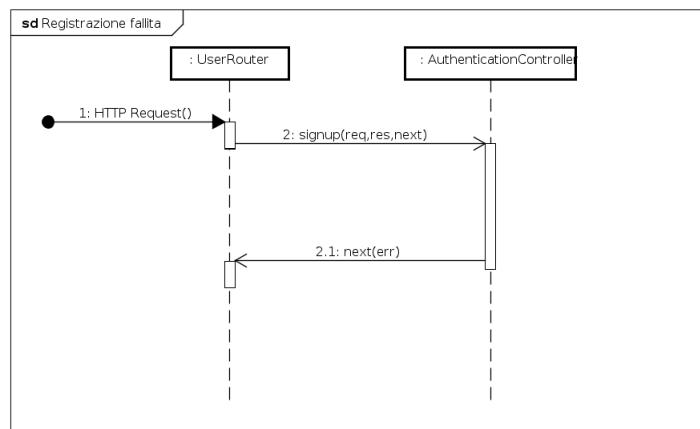


Figura 199: Fallimento della procedura di registrazione

4.2.2.3 POST /:lang/signin

- **Successo:** la maggior parte delle richieste alle risorse rese disponibili dalle API_G del $server_G$ possono essere effettuate solamente da un *utente autenticato*. Tale scenario rappresenta il successo di una richiesta di login, che impone come vincolo per poter essere effettuata, che l'utente abbia un account nel sistema, ma che non sia già autenticato. La login dell'utente viene effettuata tramite *Passport_G*.

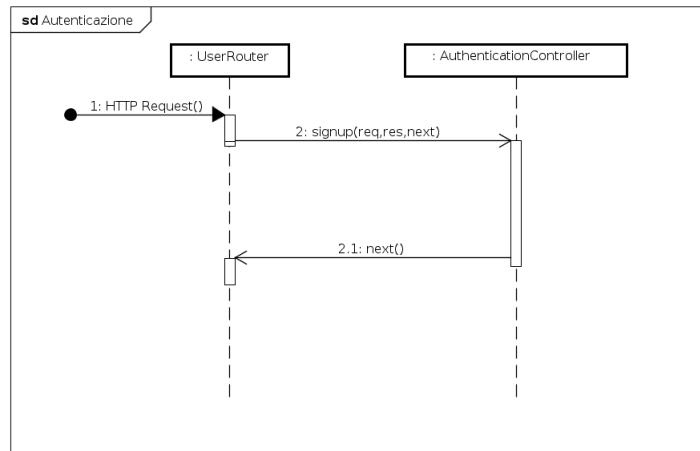


Figura 200: Procedura di autenticazione

- **Fallimento:** la maggior parte delle richieste alle risorse rese disponibili dalle API_G del $server_G$ possono essere effettuate solamente da un *utente autenticato*. Tale scenario rappresenta il fallimento della richiesta di login, che può avvenire nel caso sia stia cercando di autenticare un utente non registrato. In questo caso il modulo **AuthenticationController** invia `next(error)` al router, il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

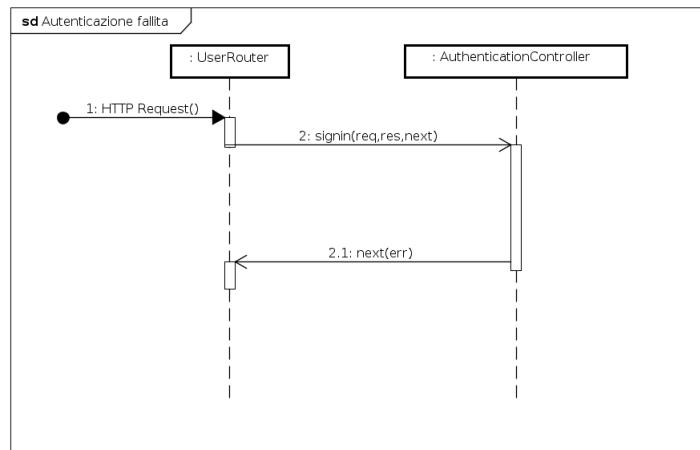


Figura 201: Fallimento della procedura di autenticazione

4.2.2.4 POST /:lang/signout

- **Successo:** questo scenario rappresenta l'azione di logout da parte di un utente precedentemente autenticato nel sistema. La logout dell'utente viene effettuata tramite $Passport_G$.

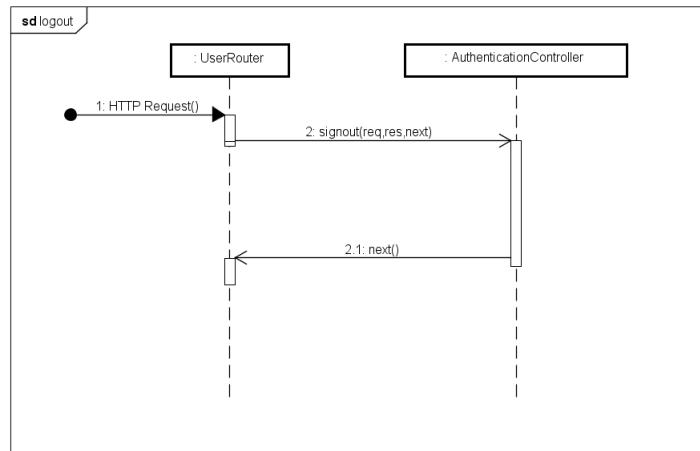


Figura 202: Procedura di logout

4.2.2.5 GET /:lang/loggedin

- Successo:** questo scenario rappresenta il successo di una richiesta di controllo di sessione che non impone alcun vincolo per poter essere effettuata. Il controllo di sessione dell'utente nel sistema viene effettuata tramite il modulo **SessionController** che invia **next()** al router, per indicare il successo dell'operazione.

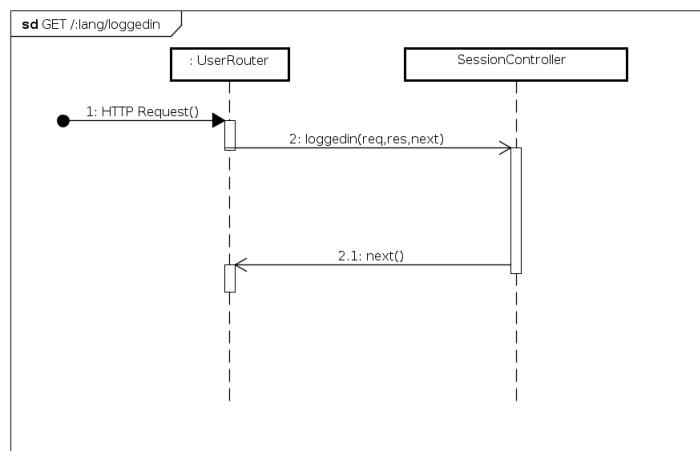


Figura 203: Procedura di controllo di sessione

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di controllo di sessione che non impone alcun vincolo per poter essere effettuata. Il controllo di sessione dell'utente nel sistema viene effettuata tramite il modulo **SessionController** che, in caso di fallimento, invia **next(err)** al router, il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

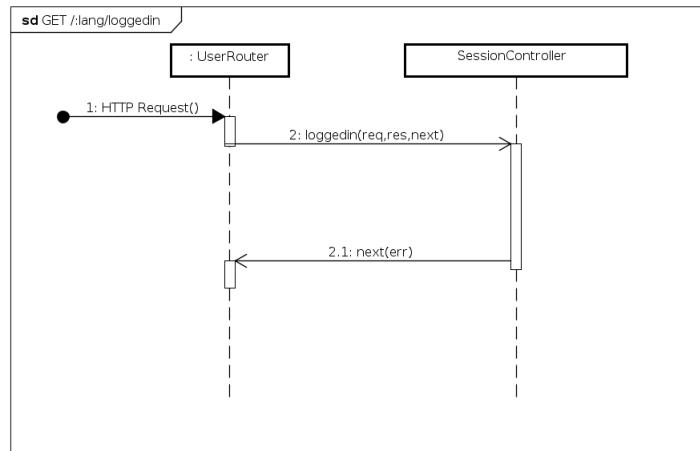


Figura 204: Fallimento della procedura di controllo di sessione

4.2.2.6 POST /:lang/recovery

- Successo:** quando un utente registrato dimentica la propria password ha la possibilità di accedere al proprio account facendosi inviare una nuova password all'indirizzo email utilizzato durante la registrazione. Questo scenario rappresenta il successo di una procedura di *recovery* per password dimenticata con vincolo che l'utente sia registrato, ma non autenticato. Il sistema dovrà occuparsi di generare un password, inviarla all'indirizzo email dell'utente e di cambiarla nel suo rispettivo account.

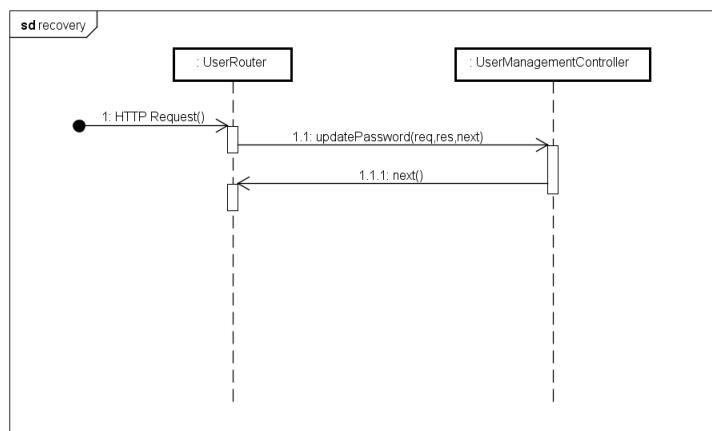


Figura 205: Procedura di recupero password

- Fallimento:** quando un utente richiede un recupero della password inserendo una email non presente nel sistema, viene sollevato un errore. Tale scenario rappresenta il fallimento di una richiesta di recupero della password che impone, come vincolo per poter essere effettuata, che l'utente non sia autenticato. In questo caso il modulo `AuthenticationController` invia `next(error)` per il fallimento di tale vincolo al router, il quale avrà compito di reinstradarlo (indirizzandolo verso `ErrorHandler`).

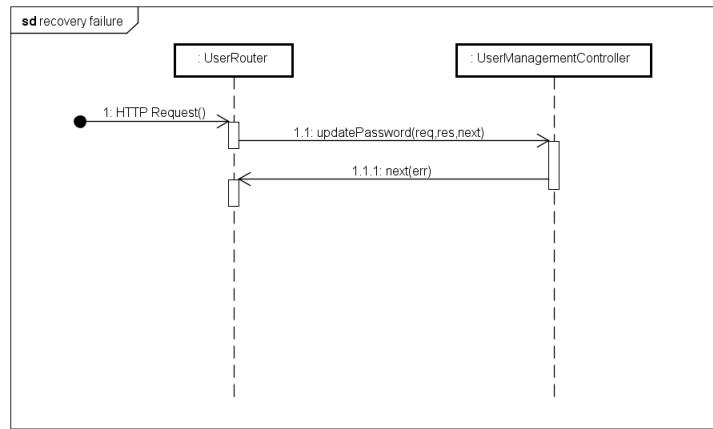


Figura 206: Fallimento della procedura di recupero della password

4.2.2.7 DELETE /:lang/user/:userId

- Successo:** questo scenario rappresenta il successo di una richiesta di eliminazione account che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita *RESTG*. In questo caso il modulo **UserManagementController** invia **next()** per indicare il successo dell'operazione e successivamente verrà effettuata l'operazione di **logout()** per completare interamente l'operazione.

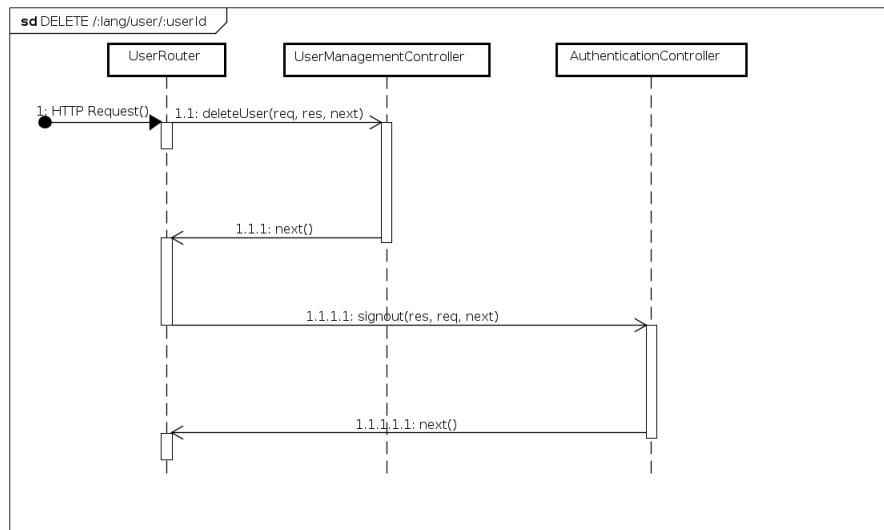


Figura 207: Procedura di eliminazione account

- Fallimento:** quando viene effettuata una richiesta di eliminazione account sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di eliminazione account che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *RESTG*. In questo caso il modulo **UserManagementController** invia **next(error)** per il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

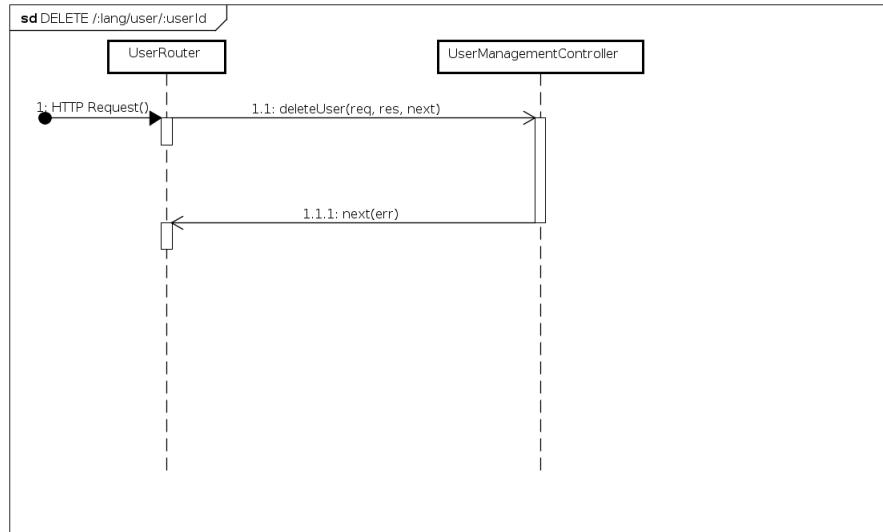


Figura 208: Fallimento della procedura di eliminazione account

4.2.2.8 GET /:lang/user/:userId

- Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione dei dati dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia `next()` al router per indicare il successo dell'operazione.

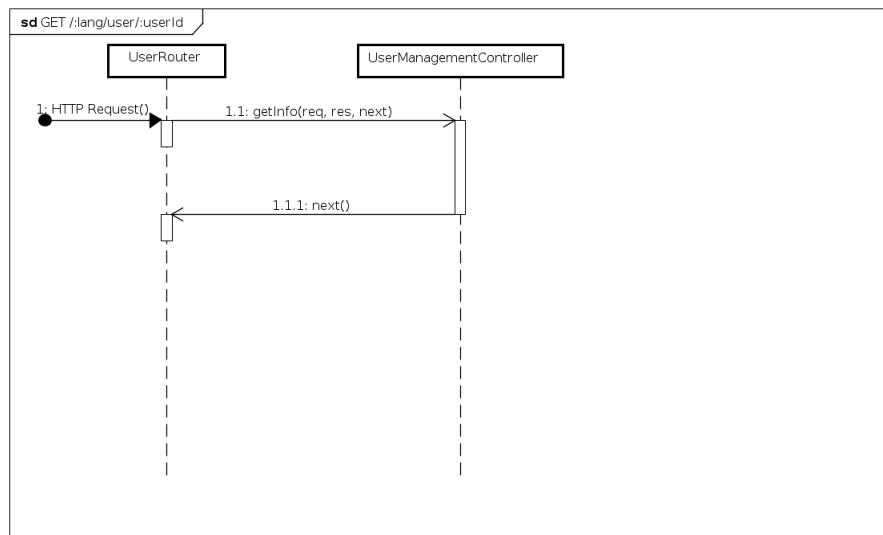


Figura 209: Procedura di visualizzazione dati utente

- Fallimento:** quando viene effettuata una richiesta di visualizzazione dei dati dell'utente sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di visualizzazione dei dati dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia `next(error)` per il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

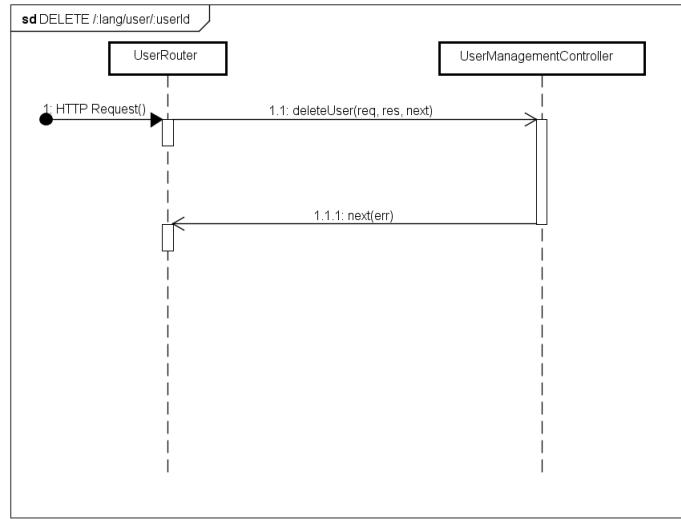


Figura 210: Fallimento della procedura di visualizzazione dati utente

4.2.2.9 PUT /:lang/user/:userId

- Successo:** questo scenario rappresenta il successo di una richiesta di modifica dei dati dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia **next()** al router per indicare il successo dell'operazione.

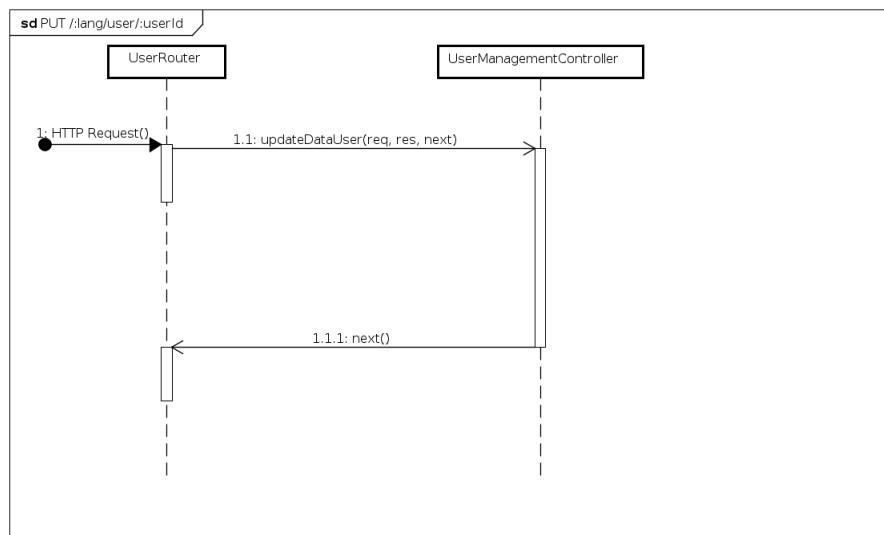


Figura 211: Procedura di modifica dati utente

- Fallimento:** quando viene effettuata una richiesta di modifica dei dati dell'utente sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di modifica dei dati dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia **next(error)** per il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

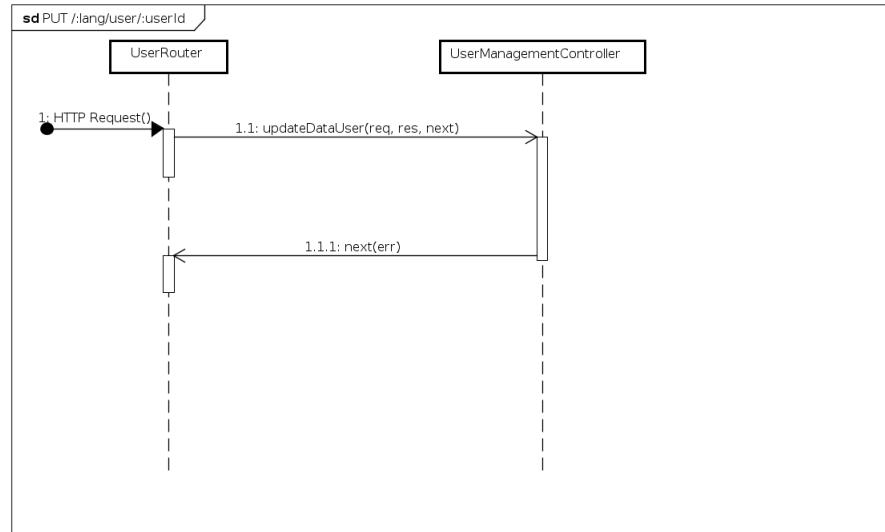


Figura 212: Fallimento della procedura di modifica dati utente

4.2.2.10 PUT /:lang/user/:userId/privacy

- Successo:** questo scenario rappresenta il successo di una richiesta di modifica della password dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia `next()` al router per indicare il successo dell'operazione.

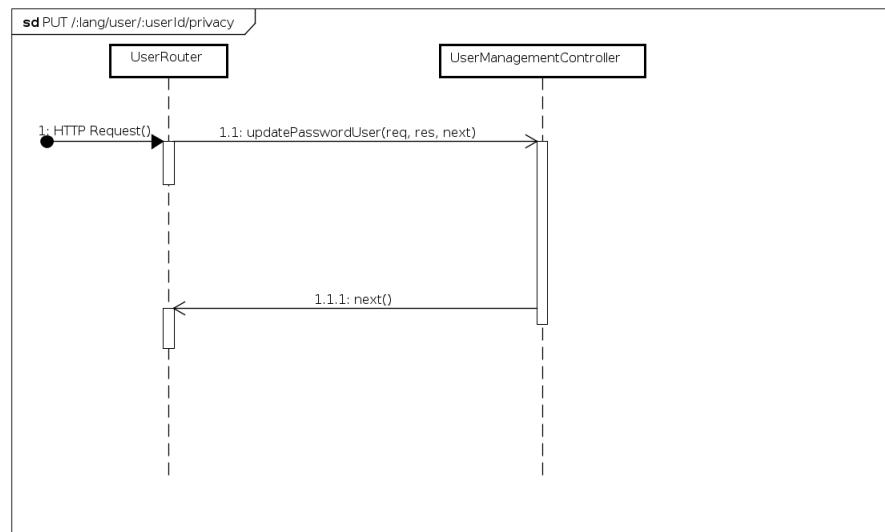


Figura 213: Procedura di modifica password

- Fallimento:** quando viene effettuata una richiesta di modifica della password dell'utente sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di modifica della password dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia `next(error)` per il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

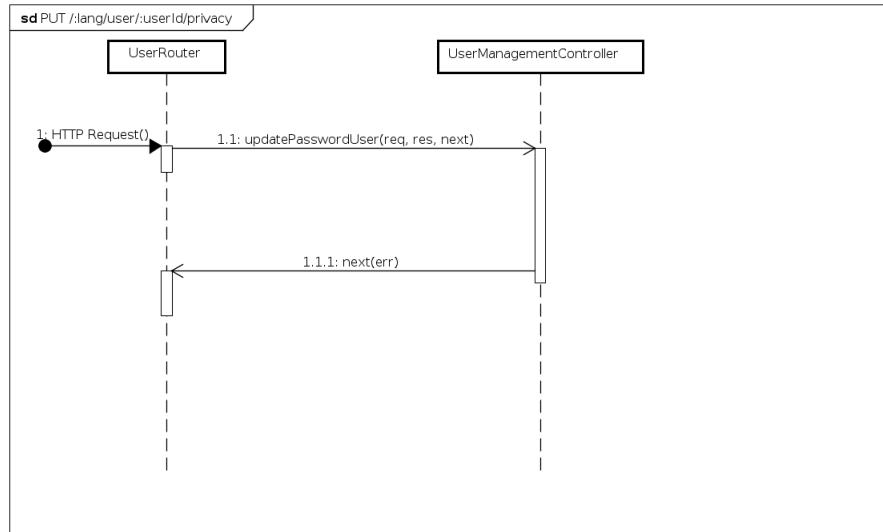


Figura 214: Fallimento della procedura di modifica password

4.2.2.11 GET /:lang/user/:userId/statistics

- Successo:** questo scenario rappresenta il successo di una richiesta di modifica delle statistiche dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `UserManagementController` invia `next()` al router per indicare il successo dell'operazione.

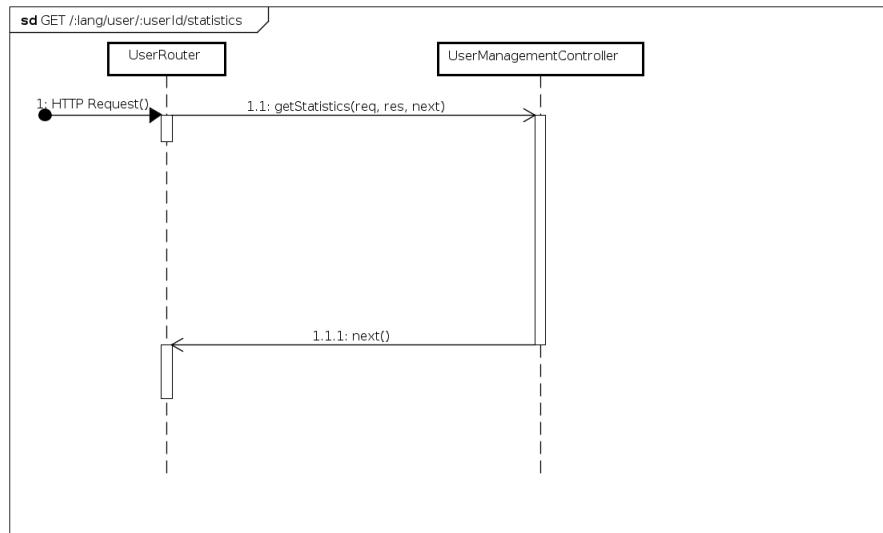


Figura 215: Procedura di visualizzazione delle statistiche

- Fallimento:** quando viene effettuata una richiesta di visualizzazione delle statistiche dell'utente sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di visualizzazione delle statistiche dell'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `UserManagementController` invia `next(error)` per il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso `ErrorHandler`).

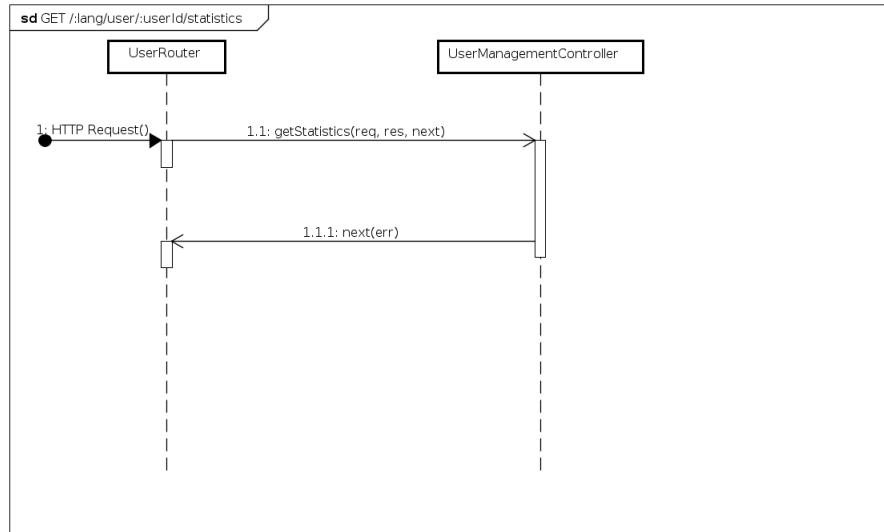


Figura 216: Fallimento della procedura di visualizzazione delle statistiche

4.2.2.12 GET /:lang/user/:userId/statistics/summary/

- **Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione della cronologia dei questionari svolti dall’utente che impone, come vincolo per poter essere effettuata, che l’utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l’apposita risorsa $REST_G$. In questo caso il modulo UserManagementController invia `next()` al router per indicare il successo dell’operazione.

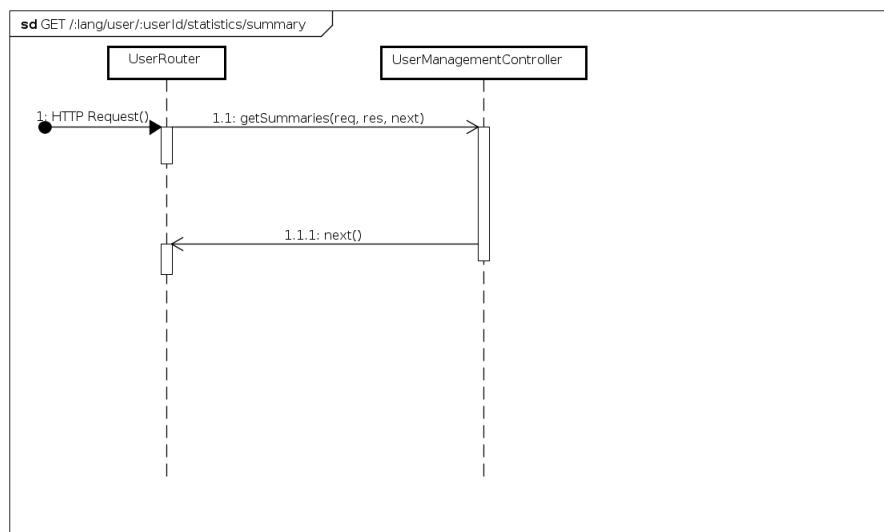


Figura 217: Procedura di visualizzazione della cronologia dei questionari svolti

- **Fallimento:** quando viene effettuata una richiesta di visualizzazione della cronologia dei questionari svolti dall’utente sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di visualizzazione della cronologia dei questionari svolti dall’utente che impone, come vincolo per poter essere effettuata, che l’utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l’apposita risorsa $REST_G$. In questo caso il modulo UserManagementController invia



`next(error)` per il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso `ErrorHandler`).

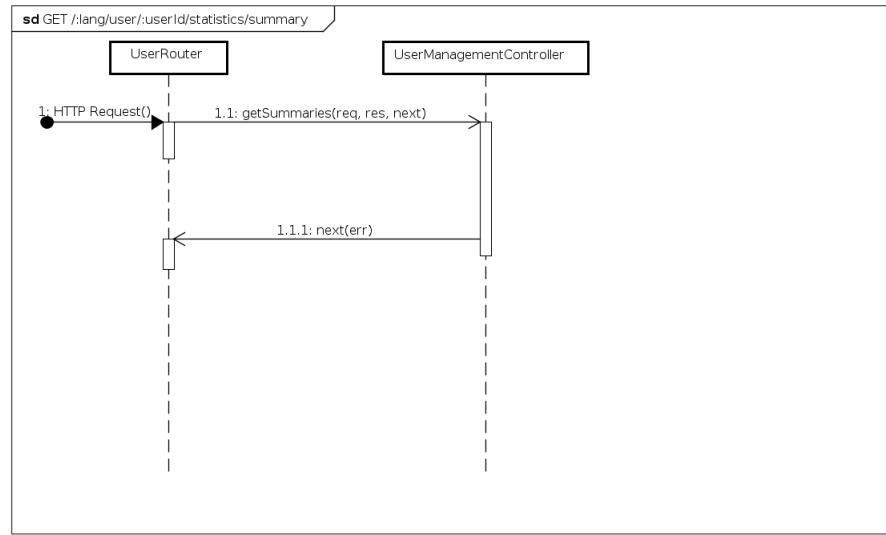


Figura 218: Fallimento della procedura di visualizzazione della cronologia dei questionari svolti

4.2.2.13 GET /:lang/user/:userId/statistics/summary/:summaryId

- **Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione del riepilogo di un particolare questionario svolto dall'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo `UserManagementController` invia `next()` al router per indicare il successo dell'operazione.

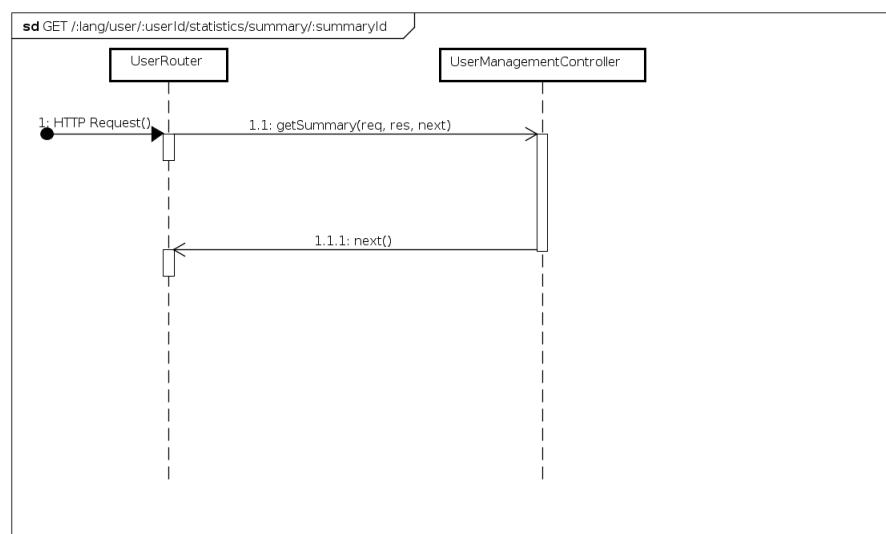


Figura 219: Procedura di visualizzazione del riepilogo di un particolare questionario svolto

- **Fallimento:** quando viene effettuata una richiesta di visualizzazione del riepilogo di un particolare questionario svolto dall'utente sollevando un errore. Tale scenario rappresenta il fallimento di una richiesta di visualizzazione del riepilogo di un particolare questionario



svolto dall'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema, quindi, prima di tale operazione, deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **UserManagementController** invia `next(error)` per il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

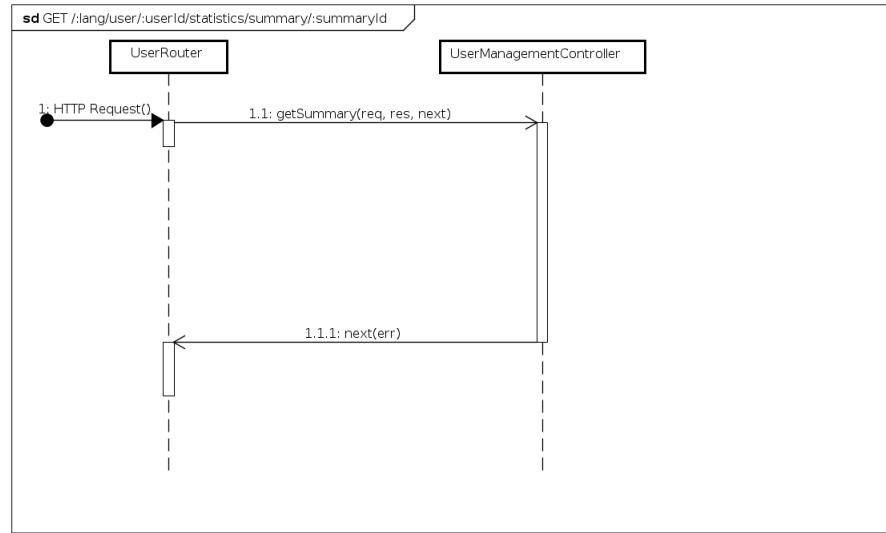


Figura 220: Fallimento della procedura di visualizzazione del riepilogo di un particolare questionario svolto

4.2.2.14 GET /:lang/user/:userId/quiz

- **Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione dei questionari creati che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia un utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next()` per indicare il successo dell'operazione.

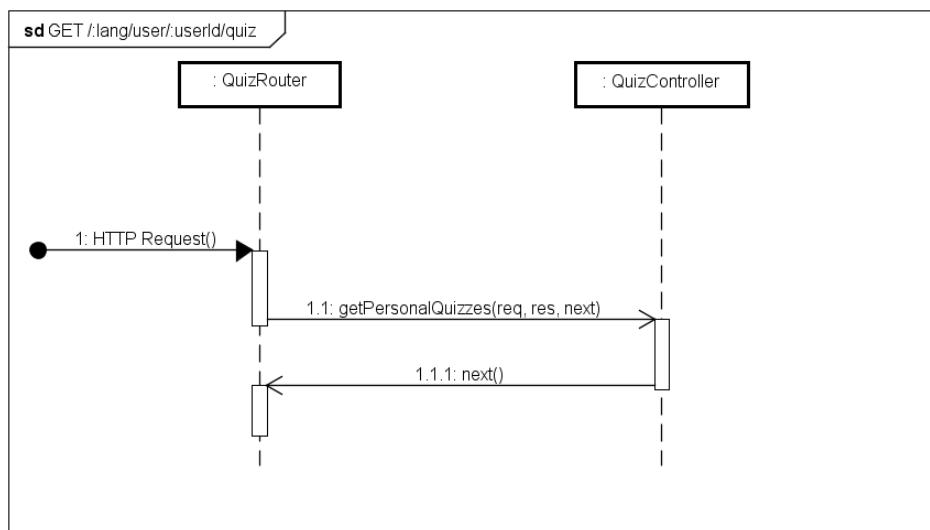


Figura 221: Procedura di visualizzazione questionari creati



- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta di visualizzazione dei questionari creati che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia un utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

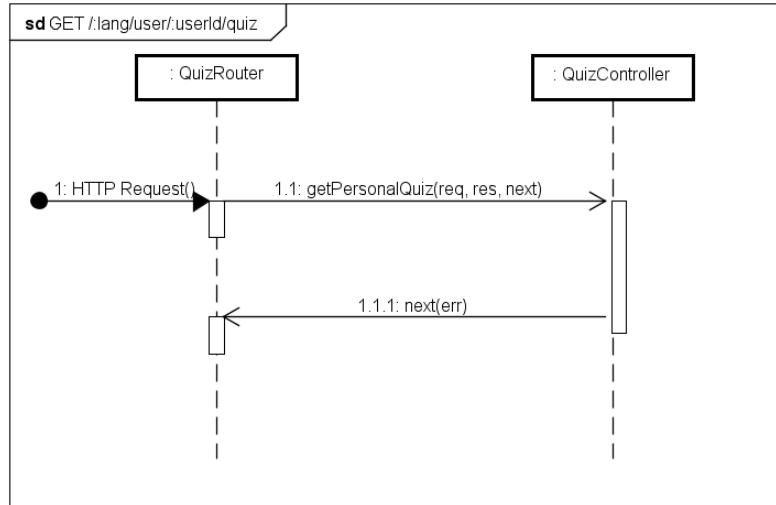


Figura 222: Fallimento della procedura di visualizzazione questionari creati

4.2.2.15 GET /:lang/user/:userId/quiz/:quizId

- **Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione del questionario creato dall'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato come utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next()` per indicare il successo dell'operazione.

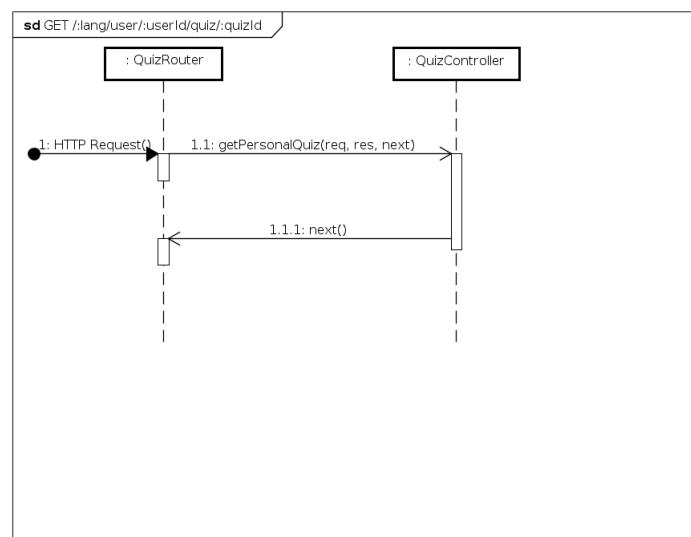


Figura 223: Procedura di visualizzazione questionari creati



- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta di visualizzazione del questionario creato che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia un utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

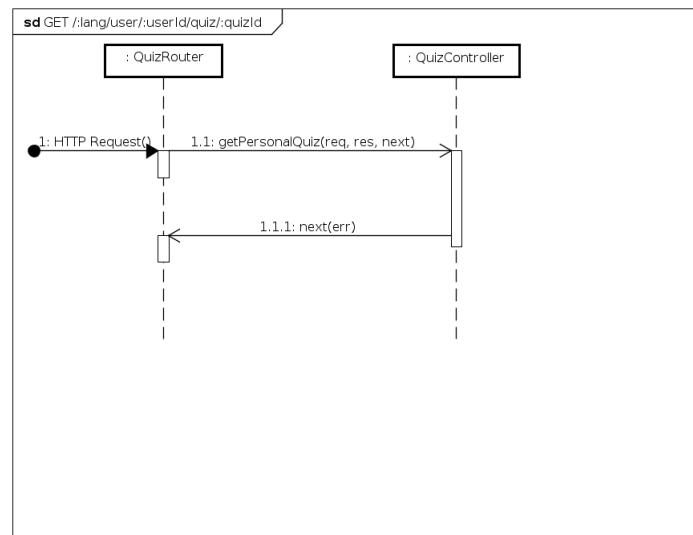


Figura 224: Fallimento della procedura di visualizzazione questionari creati

4.2.2.16 POST /:lang/:userId/:quiz

- **Successo:** questo scenario rappresenta il successo di una richiesta di creazione di un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia un utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next()` per indicare il successo dell'operazione.

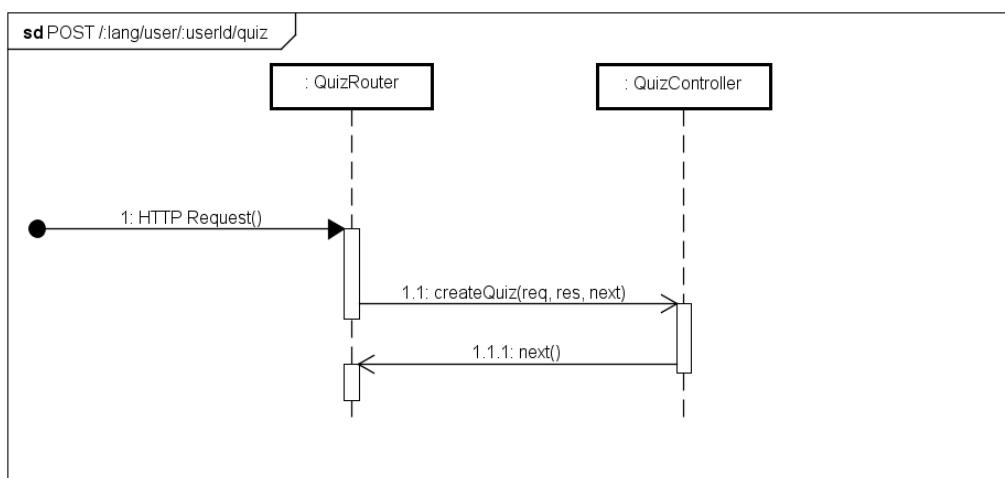


Figura 225: Procedura di creazione di un questionario

- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta di creazione di un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia



autenticato al sistema e sia un utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

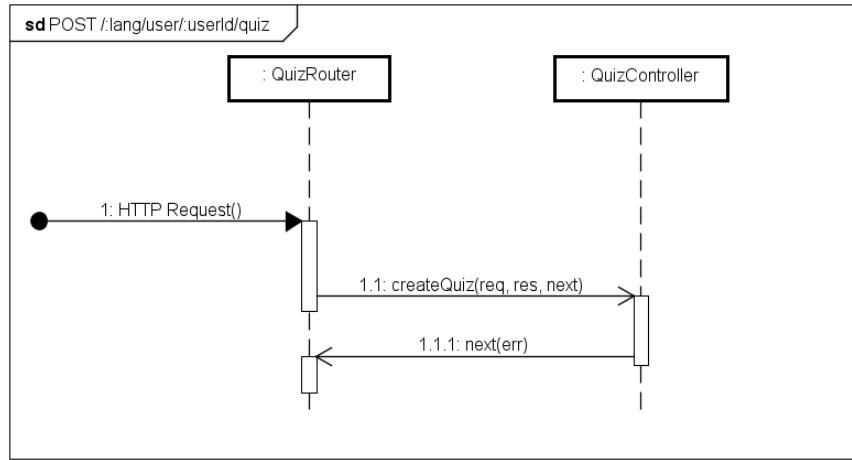


Figura 226: Fallimento della procedura di creazione di un questionario

4.2.2.17 PUT /:lang/user/:userId/quiz/:quizId/removeUser

- Successo:** questo scenario rappresenta il successo di una richiesta di eliminazione di utente iscritto ad un questionario creato dall'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato come utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next()` per indicare il successo dell'operazione.

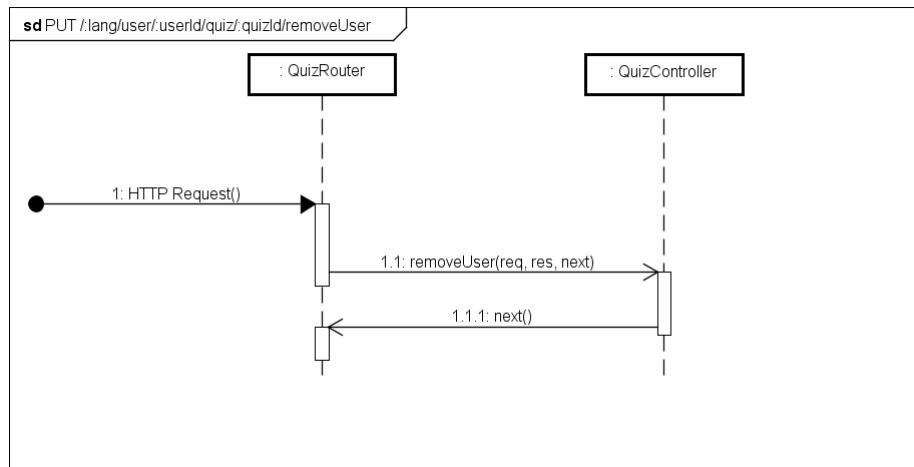


Figura 227: Procedura di rimozione di un utente iscritto ad un questionario creato

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di eliminazione di utente iscritto ad un questionario creato dall'utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia un utente pro; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita



risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

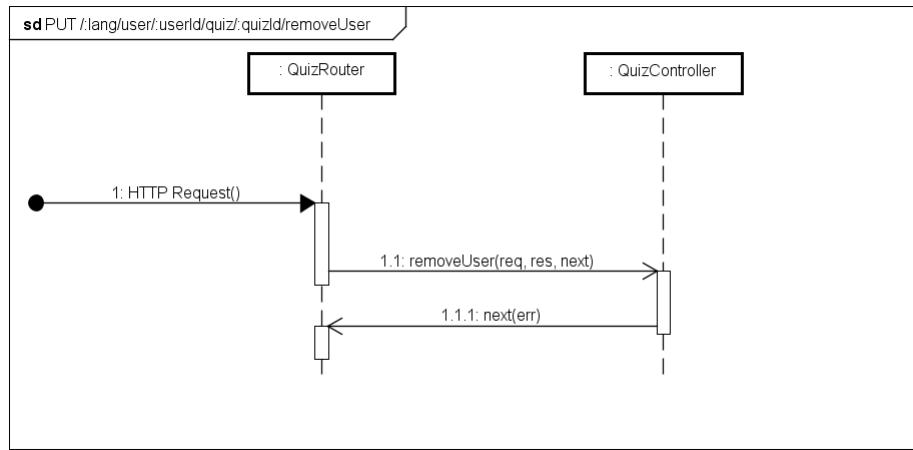


Figura 228: Fallimento della procedura di visualizzazione questionari creati

4.2.2.18 POST /:lang/user/:userId/quiz/:quizId/addUser

- Successo:** questo scenario rappresenta il successo di una richiesta di iscrizione ad un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next()` per indicare il successo dell'operazione.

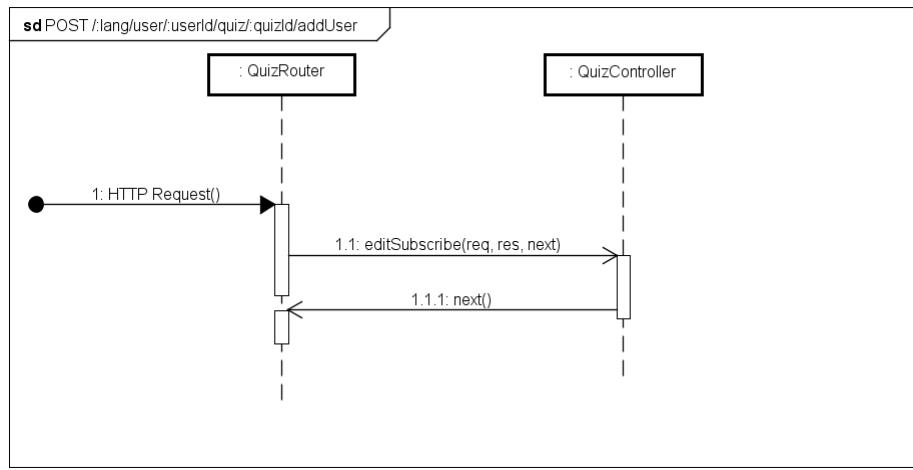


Figura 229: Procedura di iscrizione ad un questionario

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di iscrizione ad un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuizController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

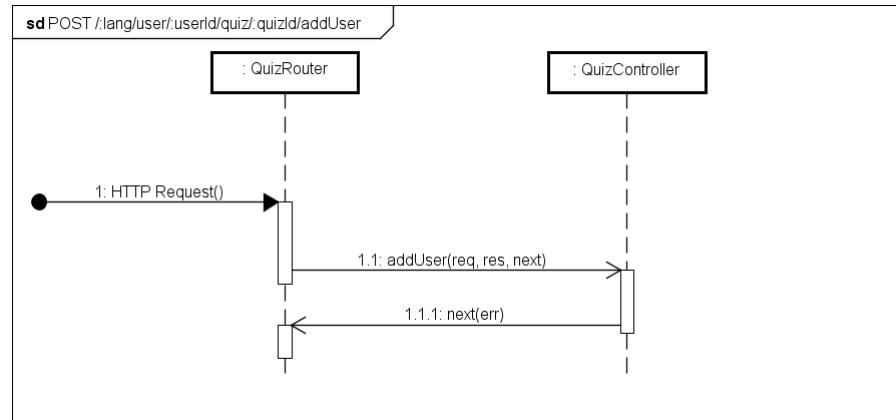


Figura 230: Fallimento della procedura di iscrizione ad un questionario

4.2.2.19 POST /:lang/user/:userId/quiz/:quizId/activeUser

- Successo:** questo scenario rappresenta il successo di una richiesta di aggiunta alla lista di utenti che hanno compilato un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `QuizController` invia `next()` per indicare il successo dell'operazione.

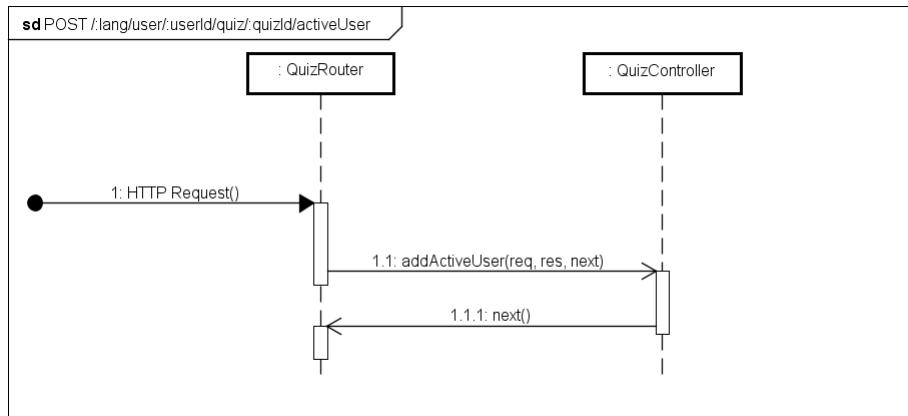


Figura 231: Procedura di aggiunta alla lista di utenti che hanno compilato un questionario

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di aggiunta alla lista di utenti che hanno compilato un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `QuizController` invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso `ErrorHandler`).

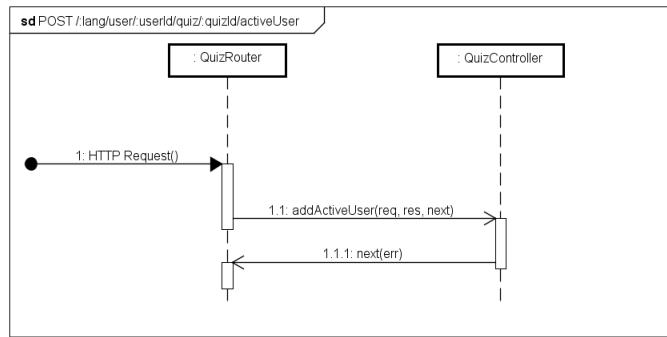


Figura 232: Fallimento della procedura di aggiunta alla lista di utenti che hanno compilato un questionario

4.2.2.20 PUT /:lang/user/:userId/quiz/:quizId

- Successo:** questo scenario rappresenta il successo di una richiesta di modifica di un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia l'utente pro che aveva creato il questionario che si vuole modificare; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo QuizController invia next() per indicare il successo dell'operazione.

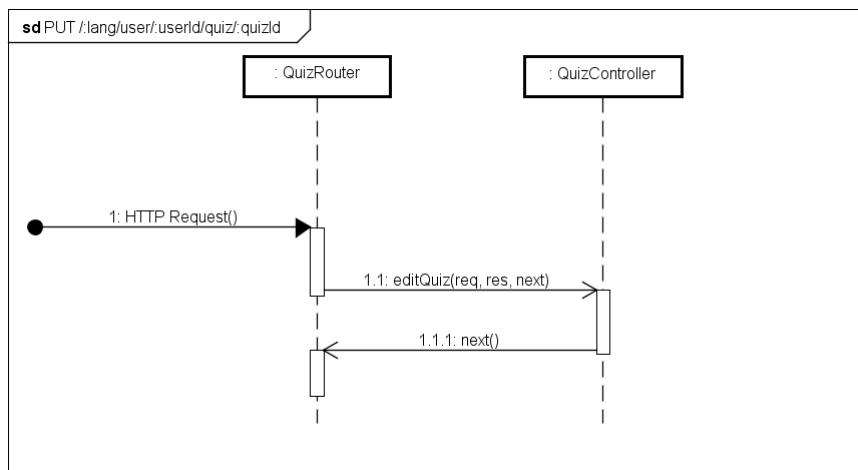


Figura 233: Procedura di modifica di un questionario

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di modifica di un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e sia l'utente pro che aveva creato il questionario che si vuole modificare; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo QuizController invia next(error) per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso ErrorHandler).

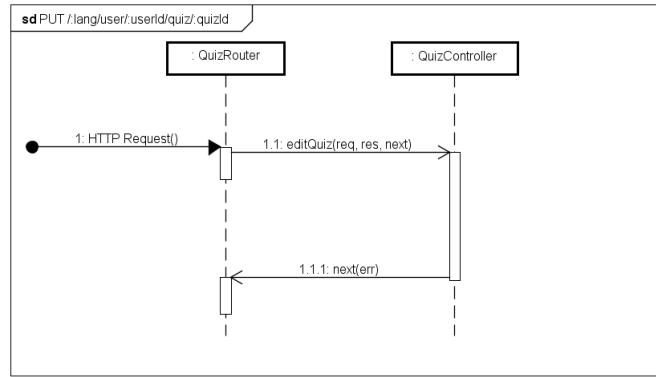


Figura 234: Fallimento della procedura di modifica di un questionario

4.2.2.21 GET /:lang/user/:userId/quiz/:quizId/test

- **Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione del questionario da compilare che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e che sia iscritto e in seguito abilitato a compilare il questionario; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo QuizController invia `next()` per indicare il successo dell'operazione.

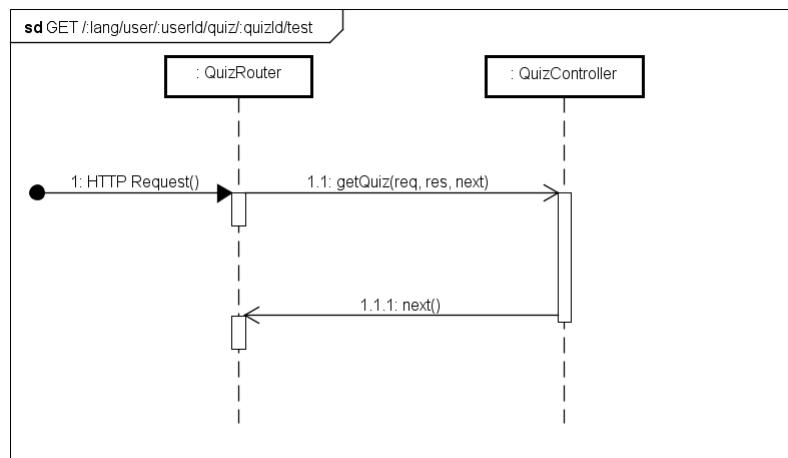


Figura 235: Procedura di visualizzazione del questionario da compilare

- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta di visualizzazione del questionario da compilare che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e che sia iscritto e in seguito abilitato a compilare il questionario; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo QuizController invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso `ErrorHandler`).

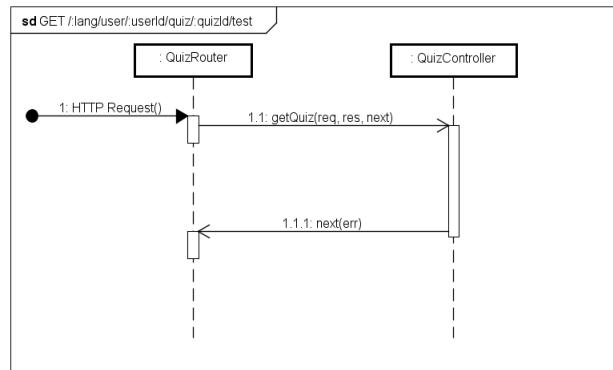


Figura 236: Fallimento della procedura di visualizzazione del questionario da compilare

4.2.2.22 POST /:lang/user/:userId/quiz/:quizId/summary

- Successo:** questo scenario rappresenta il successo di una richiesta di creazione di un riepilogo che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e che abbia finito di compilare un questionario; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **SummaryController** invia `next()` per indicare il successo dell'operazione e successivamente verrà effettuata la `updateSummary()` per aggiornare la cronologia dell'utente.

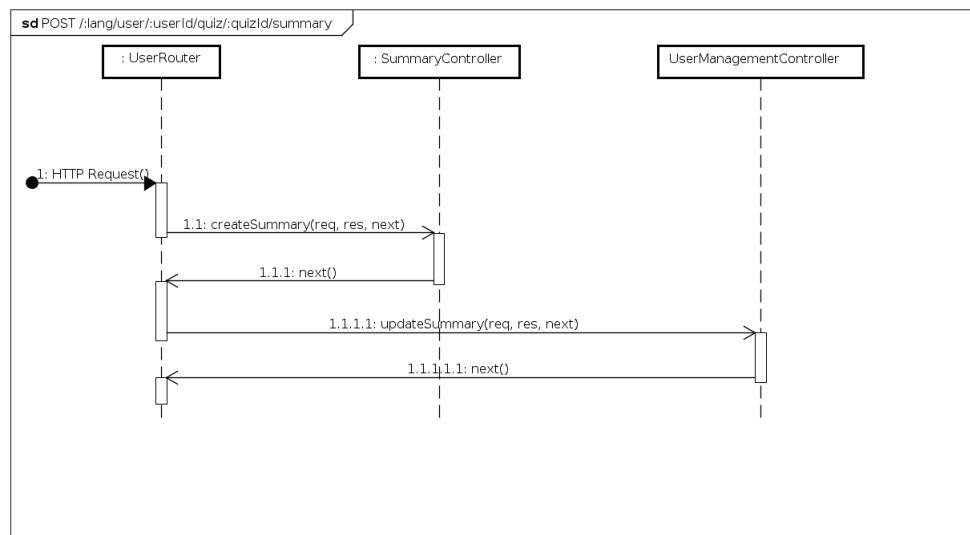


Figura 237: Procedura di creazione di un riepilogo

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di creazione di un riepilogo che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema e che abbia finito di compilare un questionario; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **SummaryController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

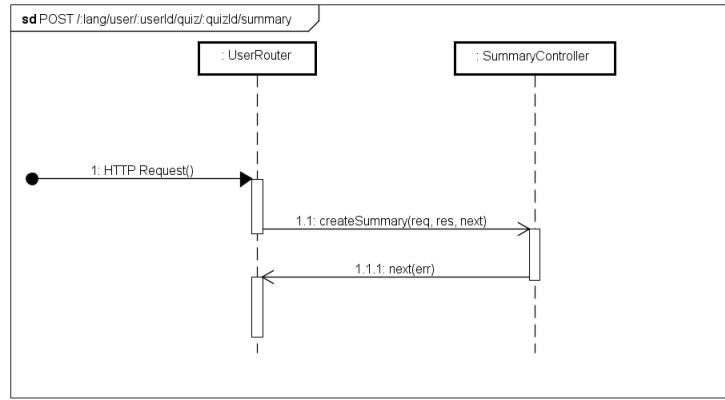


Figura 238: Fallimento della procedura di creazione di un riepilogo

4.2.2.23 GET /:lang/user/:userId/question

- **Successo:** questo scenario rappresenta il successo di una richiesta che ritorna tutte le domande create da un utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuestionController** invia `next()` per indicare il successo dell'operazione.

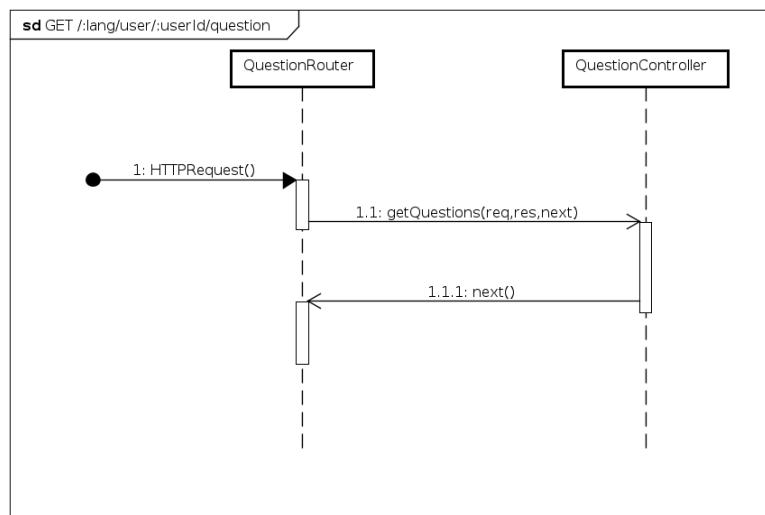


Figura 239: Procedura di visualizzazione delle domande create dall'utente

- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta che ritorna tutte le domande create da un utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo **QuestionController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

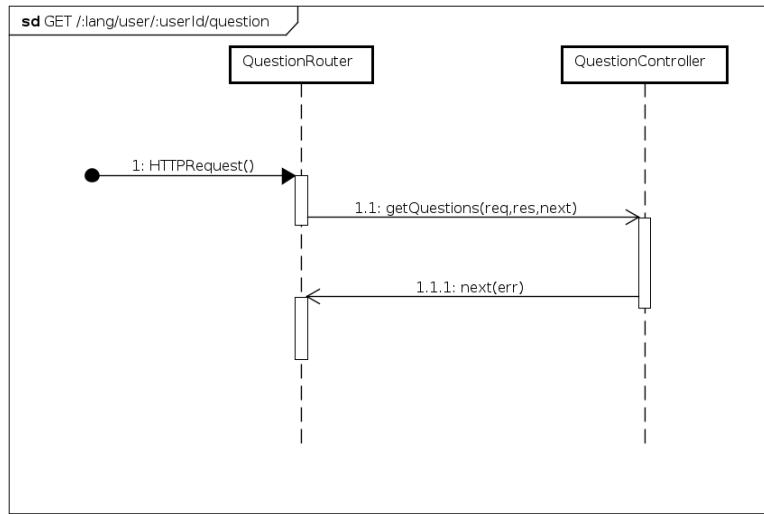


Figura 240: Fallimento della procedura di visualizzazione delle domande create dall'utente

4.2.2.24 GET /:lang/user/:userId/question/:questionId

- Successo:** questo scenario rappresenta il successo di una richiesta che ritorna una domanda creata da un utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `QuestionController` invia `next()` per indicare il successo dell'operazione.

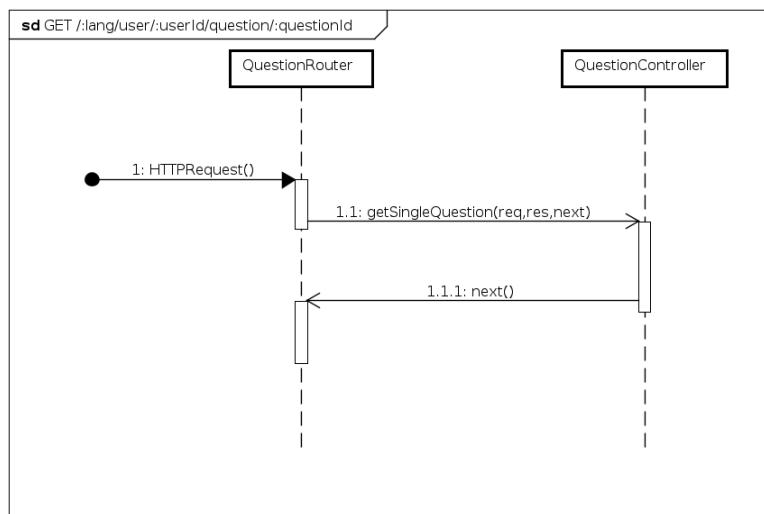


Figura 241: Procedura di visualizzazione delle domande create dall'utente

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta che ritorna una domanda creata da un utente che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `QuestionController` invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso `ErrorHandler`).

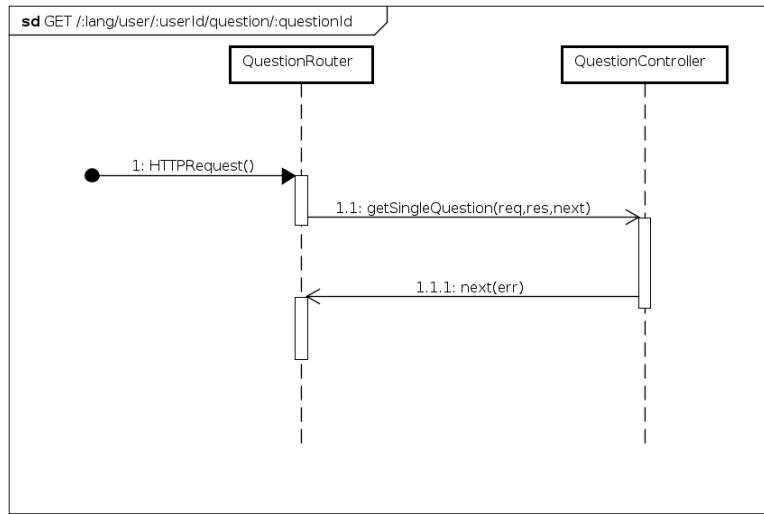


Figura 242: Fallimento della procedura di visualizzazione delle domande create dall'utente

4.2.2.25 POST /:lang/user/:userId/question

- **Successo:** questo scenario rappresenta il successo di una richiesta che inserisce una nuova domanda creata da un utente all'interno del sistema che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **QuestionController** invia `next()` per indicare il successo dell'operazione.

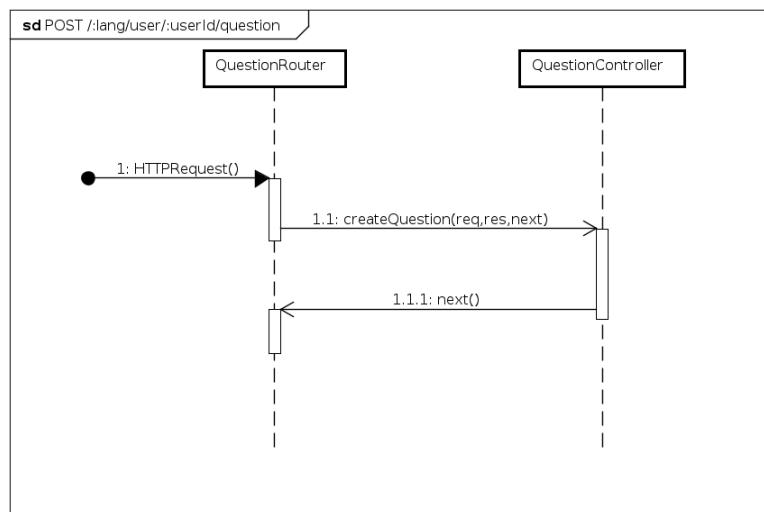


Figura 243: Procedura della creazione di una domanda

- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta che inserisce una nuova domanda creata da un utente all'interno del sistema che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **QuestionController** invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

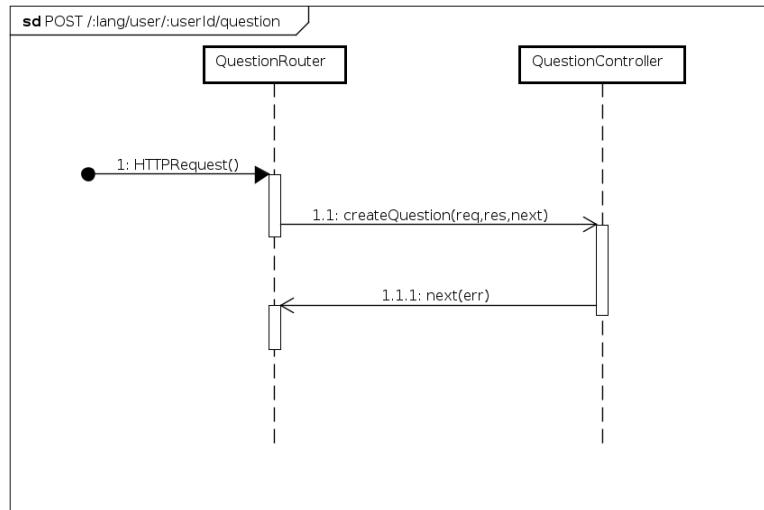


Figura 244: Fallimento della procedura di creazione di una domanda

4.2.2.26 GET /:lang/user/:userId/search/:keyword/users

- **Successo:** questo scenario rappresenta il successo di una richiesta che ritorna la lista degli utenti il cui username contiene la keyword inserita nella barra di ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **UserManagementController** invia **next()** per indicare il successo dell'operazione.

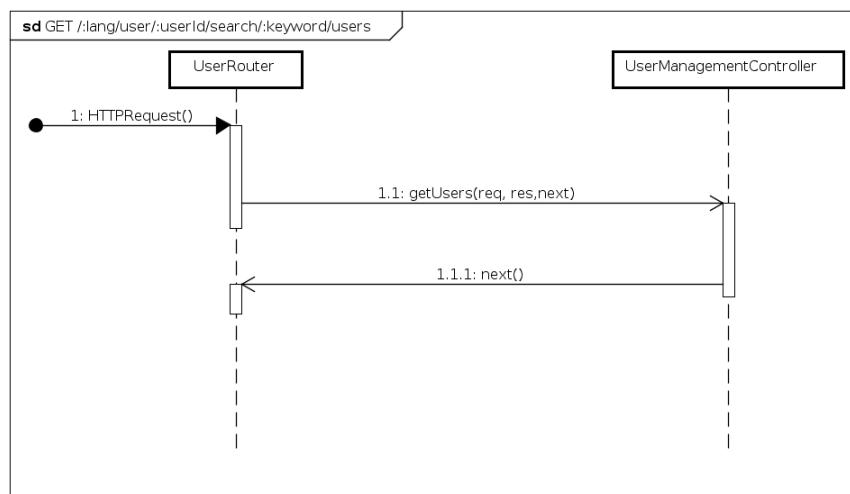


Figura 245: Procedura di visualizzazione dei risultati della ricerca utente

- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta che ritorna la lista degli utenti il cui username contiene la keyword inserita nella barra di ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **UserManagementController** invia **next(error)** per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

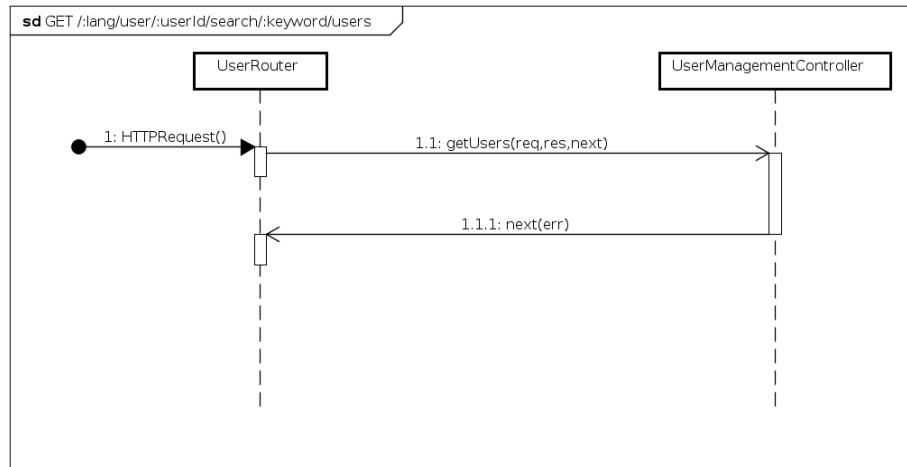


Figura 246: Fallimento della procedura di visualizzazione dei risultati della ricerca utente

4.2.2.27 GET /:lang/user/:userId/search/:keyword/quizzes

- Successo:** questo scenario rappresenta il successo di una richiesta che ritorna la lista dei questionari il cui titolo contiene la keyword inserita nella barra di ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **QuizController** invia `next()` per indicare il successo dell'operazione.

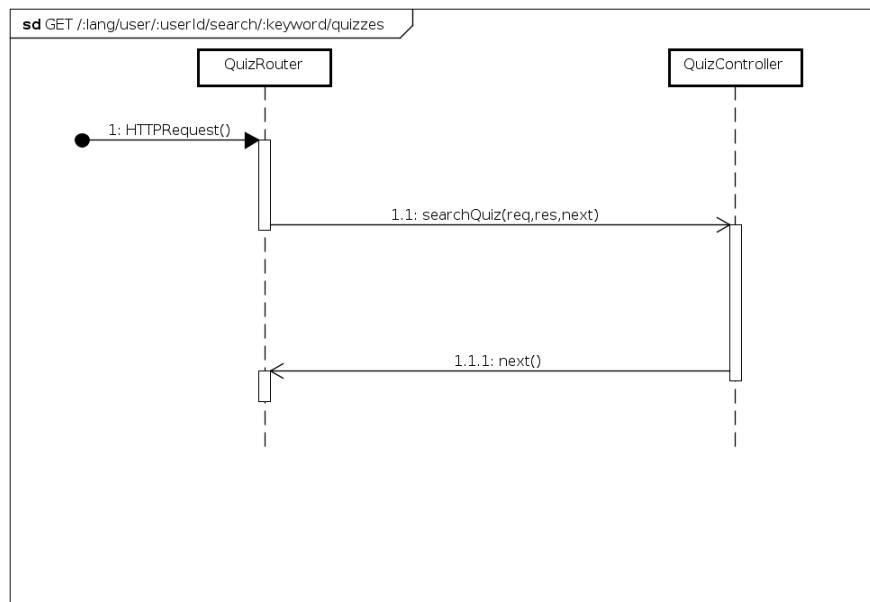


Figura 247: Procedura di visualizzazione dei risultati della ricerca questionario

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta che ritorna la lista dei questionari il cui titolo contiene la keyword inserita nella barra di ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa *REST_G*. In questo caso il modulo **QuizController** ritornerà un `next(error)` al router che avrà il compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

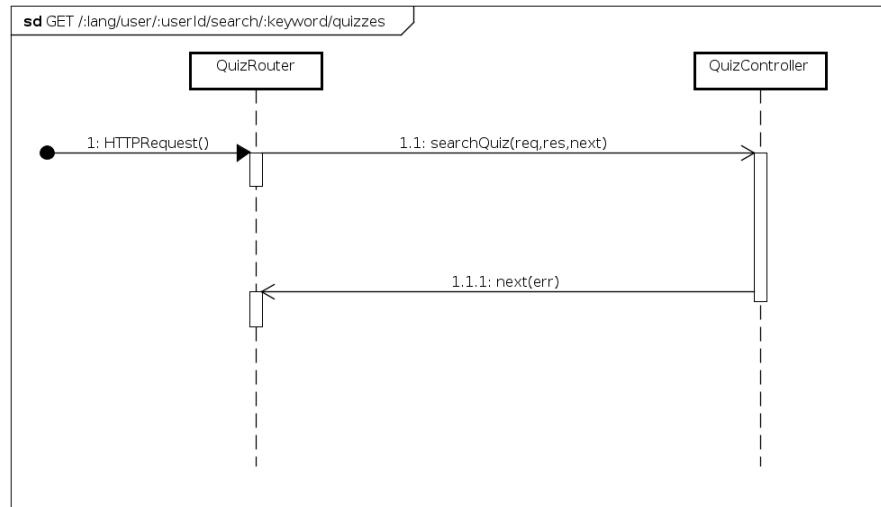


Figura 248: Fallimento della procedura di visualizzazione dei risultati della ricerca questionario

4.2.2.28 GET /:lang/user/:userId/search/users/:users

- **Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione di un utente tra quelli ritornati come risultato della ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `UserManagementController` invia `next()` per indicare il successo dell'operazione.

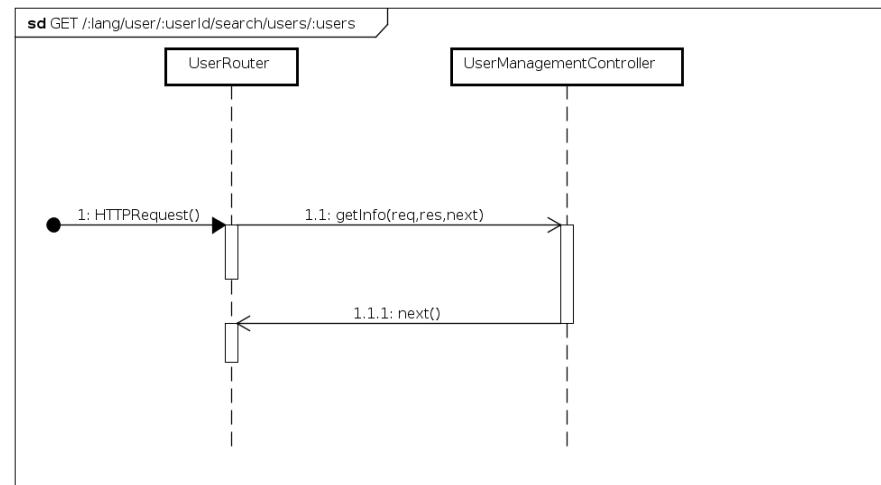


Figura 249: Procedura di visualizzazione di un utente

- **Fallimento:** questo scenario rappresenta il fallimento di una richiesta di visualizzazione di un utente tra quelli ritornati come risultato della ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `UserManagementController` invia `next(error)` per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso `ErrorHandler`).

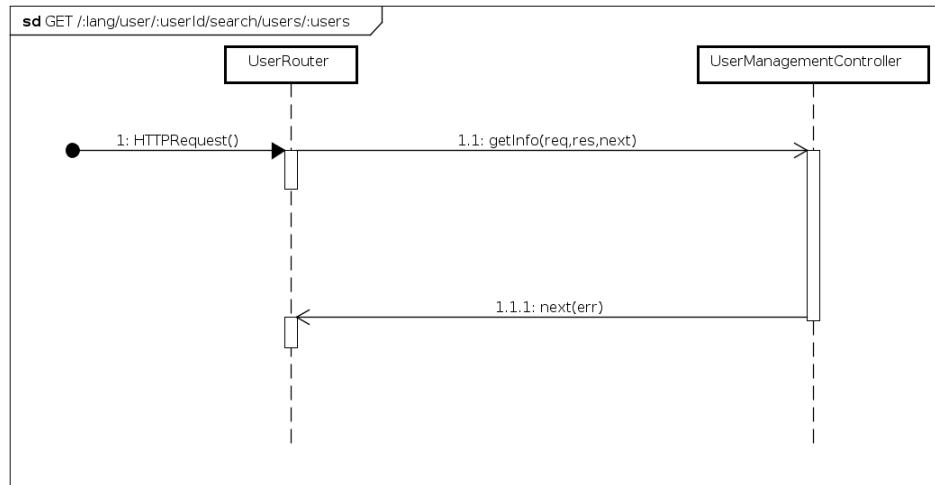


Figura 250: Fallimento della procedura della visualizzazione di un utente

4.2.2.29 GET /:lang/user/:userId/search/quizzes/:quizId

- Successo:** questo scenario rappresenta il successo di una richiesta di visualizzazione di un questionario tra quelli ritornati come risultato della ricerca che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo QuizController invia `next()` per indicare il successo dell'operazione.

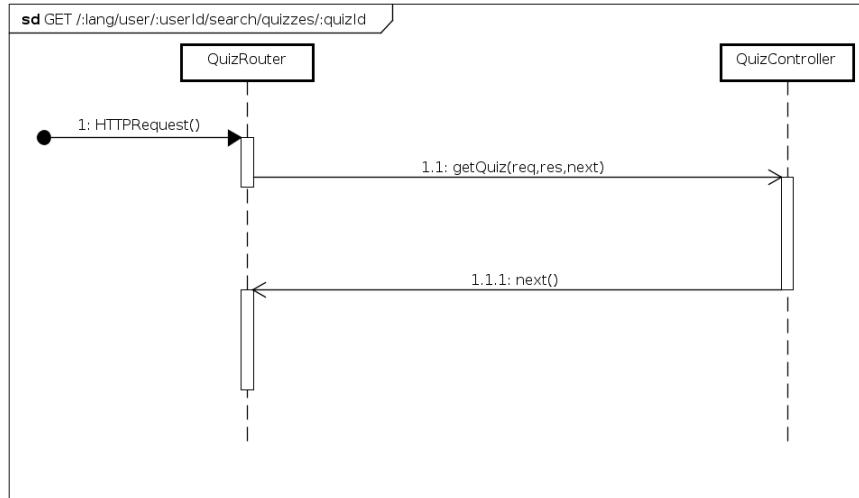


Figura 251: Procedura della visualizzazione di un questionario

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta di visualizzazione di un questionario che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo QuizController ritornerà un `next(error)` al router che avrà il compito di reinstrararlo (indirizzandolo verso ErrorHandler).

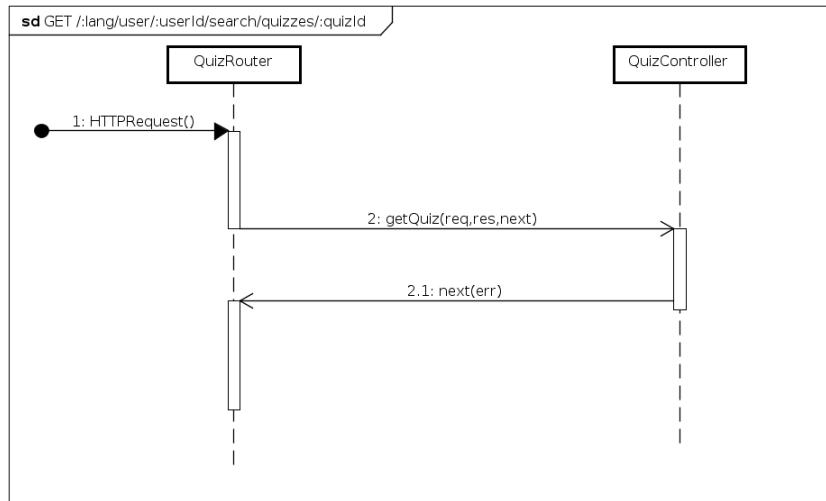


Figura 252: Fallimento della procedura di visualizzazione di un questionario

4.2.2.30 GET /:lang/user/training/question

- Successo:** questo scenario rappresenta il successo di una richiesta che ritorna la domanda successiva nella modalità allenamento. In questo caso il modulo TopicController invia `next()` per indicare il successo dell'operazione.

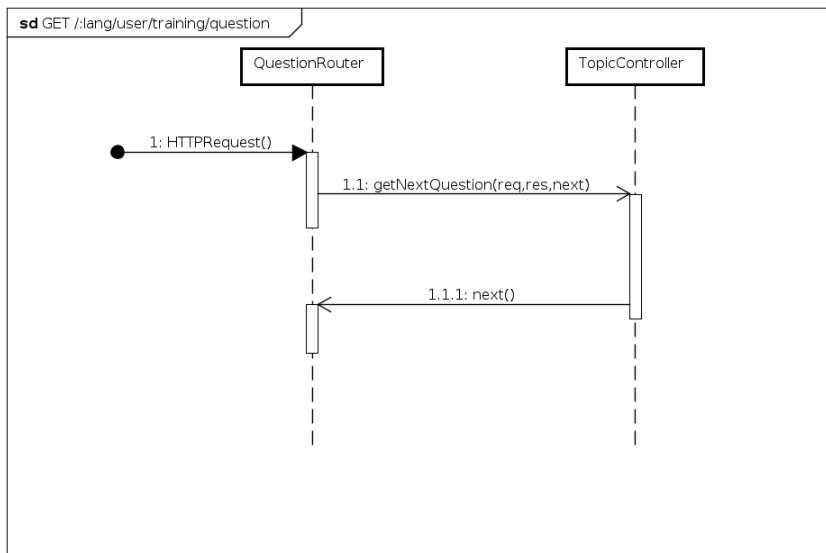


Figura 253: Procedura di restituzione della domanda successiva nella modalità allenamento

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta che ritorna la domanda successiva nella modalità allenamento; in questo caso il modulo TopicController ritornerà un `next(err)` al router che avrà il compito di reinstrararlo (indirizzandolo verso `ErrorHandler`).

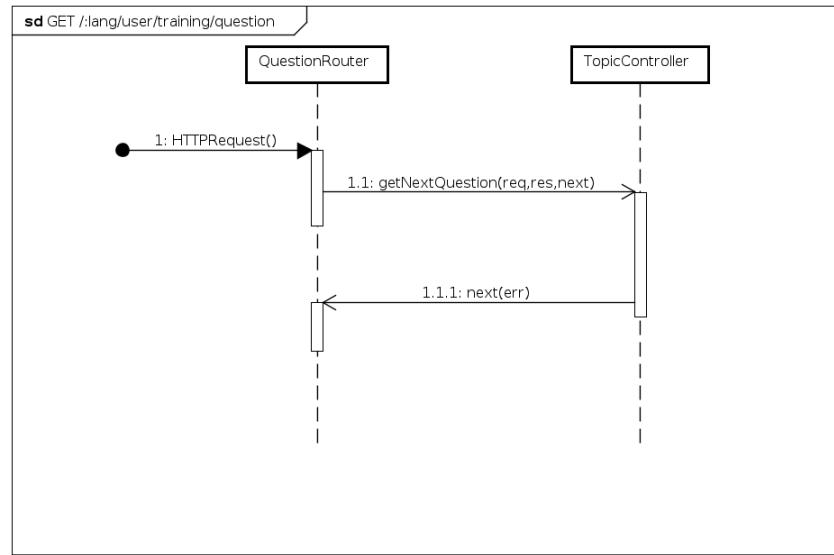


Figura 254: Fallimento della procedura di restituzione della domanda successiva nella modalità allenamento

4.2.2.31 PUT /:lang/user/:userId/training/userstatistics

- Successo:** questo scenario rappresenta il successo di una richiesta aggiorna le statistiche di un utente dopo che ha risposto ad una domanda che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `UserManagementController` invia `next()` per indicare il successo dell'operazione.

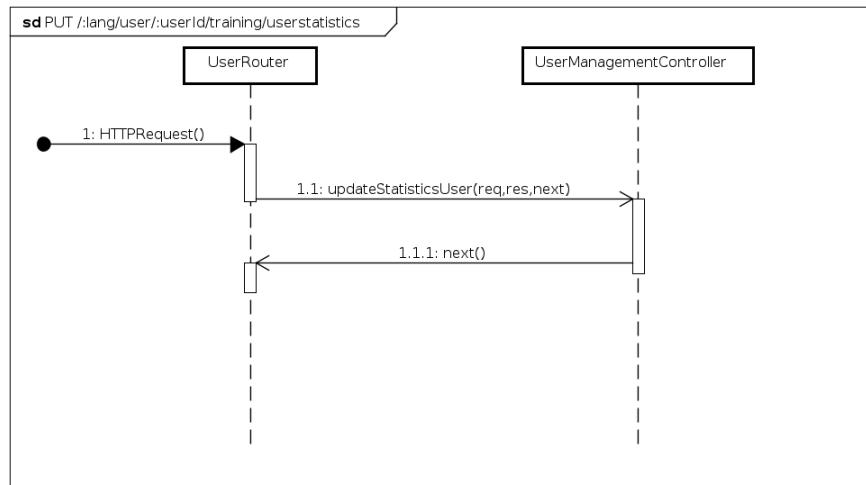


Figura 255: Procedura di aggiornamento delle statistiche dell'utente dopo una risposta ad una domanda

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta che aggiorna le statistiche di un utente dopo che ha risposto ad una domanda che impone, come vincolo per poter essere effettuata, che l'utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l'apposita risorsa $REST_G$. In questo caso il modulo `UserManagementController` invia `next(error)`



per indicare il fallimento di tale vincolo al router il quale avrà compito di reinstradarlo (indirizzandolo verso **ErrorHandler**).

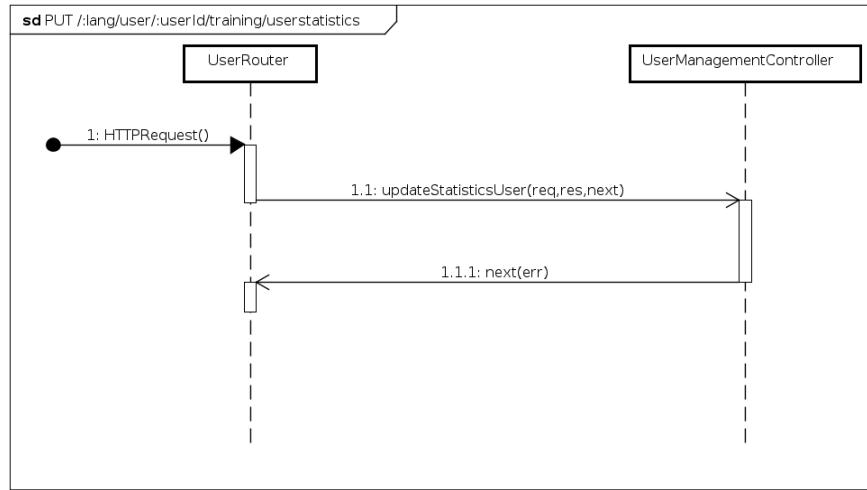


Figura 256: Fallimento della procedura di aggiornamento delle statistiche dell’utente dopo una risposta ad una domanda

4.2.2.32 PUT /:lang/user/:userId/training/questionstatistics

- Successo:** questo scenario rappresenta il successo di una richiesta aggiorna le statistiche di una domanda a cui è stata data una risposta che impone, come vincolo per poter essere effettuata, che l’utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l’apposita risorsa *REST_G*. In questo caso il modulo **QuestionController** invia `next()` per indicare il successo dell’operazione.

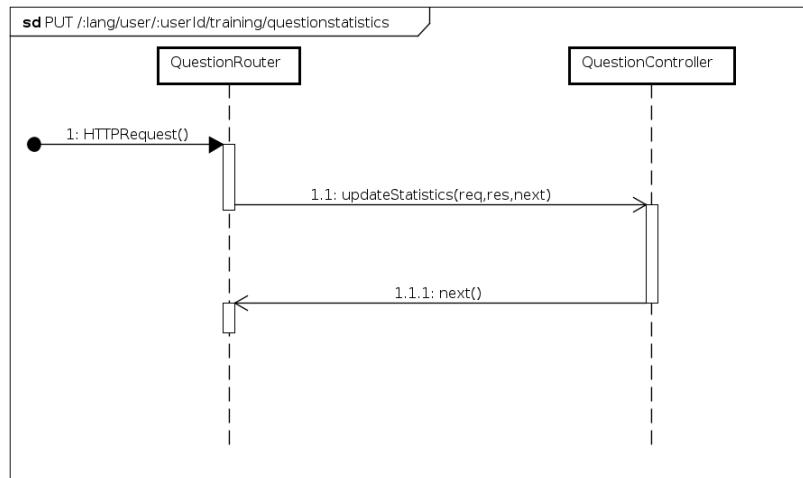


Figura 257: Procedura di aggiornamento delle statistiche di una domanda

- Fallimento:** questo scenario rappresenta il fallimento di una richiesta che aggiorna le statistiche di una domanda a cui è appena stata data una risposta che impone, come vincolo per poter essere effettuata, che l’utente sia autenticato al sistema; quindi prima di tale operazione deve venire fatta una richiesta di controllo di sessione mediante l’apposita risorsa *REST_G*. In questo caso il modulo **QuestionController** invia `next(error)` per indicare il



fallimento di tale vincolo al router il quale avrà compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

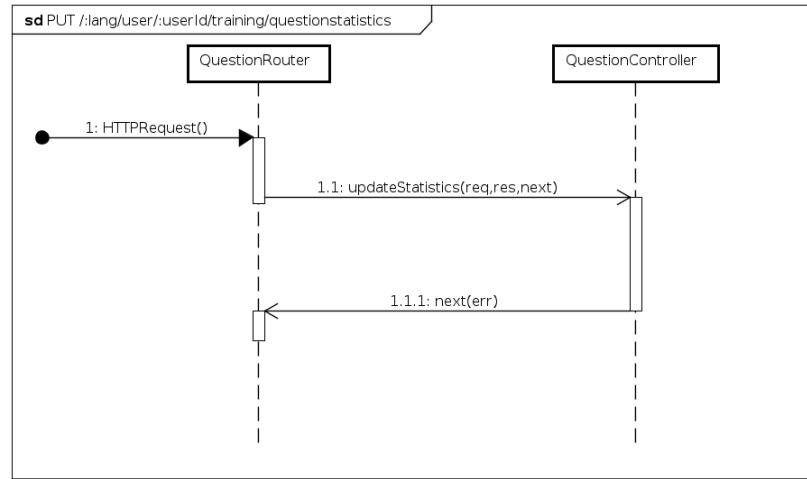


Figura 258: Fallimento della procedura di aggiornamento delle statistiche di una domanda

4.2.2.33 PUT /:lang/user/training/userlevelupdate

- Successo:** questo scenario rappresenta il successo di una richiesta che aggiorna il livello di un utente non autenticato per far sì che il sistema scelga domande adeguate alla sua preparazione. In questo caso il modulo **UserManagementController** invia **next()** per indicare il successo dell'operazione.

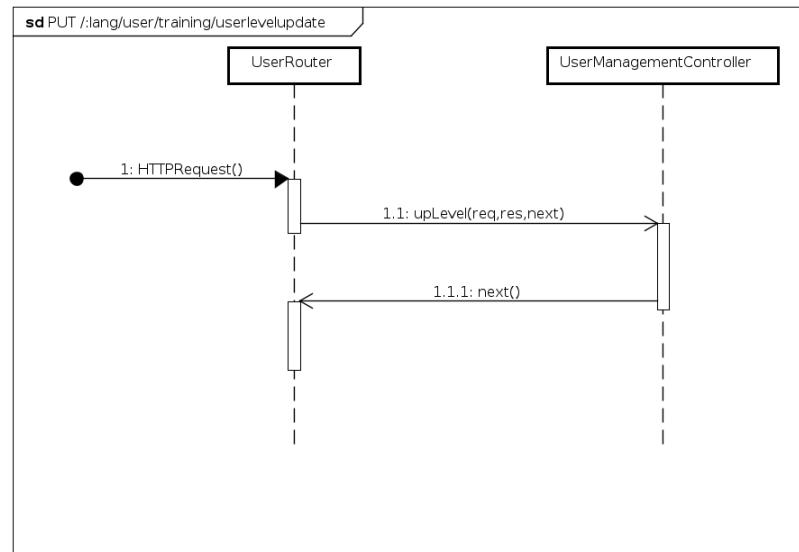


Figura 259: Procedura di aggiornamento del livello di un utente non autenticato

- Fallimento:** questo scenario rappresenta il fallimento una richiesta che aggiorna il livello di un utente non autenticato. In questo caso il modulo **UserManagementController** ritornerà un **next(err)** al router che avrà il compito di reinstrararlo (indirizzandolo verso **ErrorHandler**).

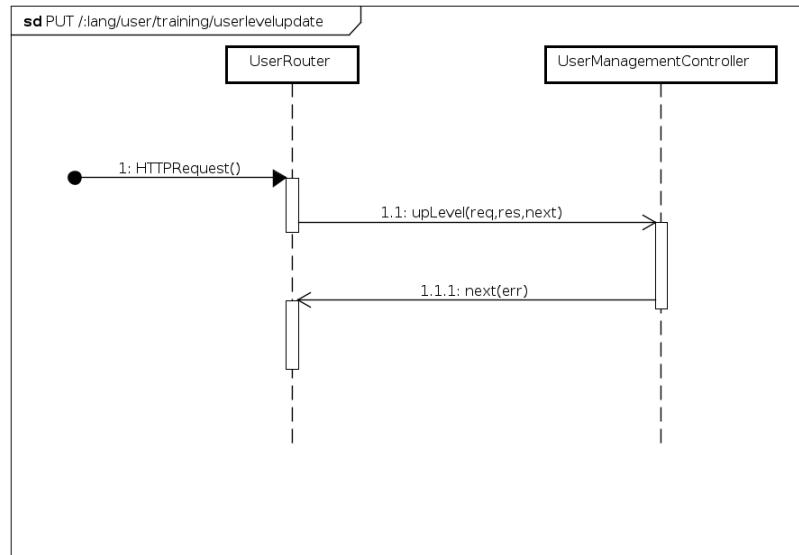


Figura 260: Fallimento della procedura di aggiornamento del livello di un utente non autenticato

5 Tracciamento Classi-Requisiti

Classe	Requisiti
Quizzipedia::Back-End::App::Controller::- ErrorsHandler	RFO1.8 RFO2.4 RFD4.7 RFD12.3.6 RFD17 RFF19.3
Quizzipedia::Back-End::App::Controller::- LangController	RFF40
Quizzipedia::Back-End::App::Controller::- NotFoundHandler	RFO43
Quizzipedia::Back-End::App::Controller::- QuestionController	RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1



Classe	Requisiti
	RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2



Classe	Requisiti
	RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO26 RFD35
Quizzipedia::Back-End::App::Controller::- QuizController	RFD5 RFO6 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.3



Classe	Requisiti
	RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.7 RFO8.7.1 RFO8.7.2 RFD8.8 RFD18 RFO21 RFO24 RFO25 RFO26 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Back-End::App::Controller::- SummaryController	RFO6.5 RFD28
Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError	RFO1.8 RFO2.4 RFD4.7 RFD12.3.6 RFD17 RFF19.3
Quizzipedia::Back-End::App::Controller::- TopicController	RFO8.7.1.1 RFD9.4.5 RFO22 RFD29
Quizzipedia::Back-End::App::Controller::- UserController	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO3 RFD8.4 RFD10.4 RFD10.5



Classe	Requisiti
	RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFF13 RFF14 RFF15 RFF16 RFF19
Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO3 RFF13 RFF14 RFF15 RFF16
Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController	RFD8.4 RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4



Classe	Requisiti
	RFD12.3.5 RFF19
Quizzipedia::Back-End::App::Controller::- Users::SessionController	RFO42
Quizzipedia::Back-End::App::Model::- LangModel	RFF40
Quizzipedia::Back-End::App::Model::- QuestionModel	RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3



Classe	Requisiti
	RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1



Classe	Requisiti
	RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO26 RFD29 RFD35
Quizzipedia::Back-End::App::Model::- QuizModel	RFD5 RFO6 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.3 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.7 RFO8.7.1 RFO8.7.2 RFD8.8 RFD18 RFO21 RFO24 RFO25 RFO26 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Back-End::App::Model::- SummaryModel	RFO6.5 RFD28
Quizzipedia::Back-End::App::Model::- TopicModel	RFO8.7.1.1 RFD9.4.5 RFO22 RFO23.8



Classe	Requisiti
	RFD29
Quizzipedia::Back-End::App::Model::- UserModel	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2 RFO2.1 RFO2.2 RFO2.3 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.9 RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFF13 RFF14 RFF15 RFF16 RFF19
Quizzipedia::Back-End::App::Model::- UserProModel	RFD4.8
Quizzipedia::Back-End::App::Routers::- LangRouter	RFF40
Quizzipedia::Back-End::App::Routers::- QuestionRouter	RFO7



Classe	Requisiti
	RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3



Classe	Requisiti
	RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8.7.1.1 RFD9.4.5 RFO22



Classe	Requisiti
	RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO26 RFD35
Quizzipedia::Back-End::App::Routers::- QuizRouter	RFD5 RFO6 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.3 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.7 RFO8.7.1 RFO8.7.2 RFD8.8 RFD18 RFO21 RFO24 RFO25 RFO26 RFD28 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Back-End::App::Routers::- UserRouter	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2 RFO2.1 RFO2.2 RFO2.3



Classe	Requisiti
	RFO3 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.9 RFO6.5 RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFF13 RFF14 RFF15 RFF16 RFF19
Quizzipedia::Back-End::Server	RFO20
Quizzipedia::Front-End::AppRouter	RFO39
Quizzipedia::Front-End::AppRun	RFO2 RFO3
Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController	RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9



Classe	Requisiti
	RFD7.5 RFD7.6 RFO26 RFD35
Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController	RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.9 RFD7.1.10 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.8 RFD7.2.9 RFO26 RFD35
Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController	RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.6.4 RFD8.6.4.1 RFO8.6.4.2 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.1.1.1



Classe	Requisiti
	RFO8.7.1.1.2 RFO8.7.2 RFO8.7.2.1 RFO24 RFO26
Quizzipedia::Front-End::Controllers::- EditorQMLController	RFD7.4 RFD7.4.1 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO26 RFD35
Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController	RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFD7.5 RFD7.6 RFO25
Quizzipedia::Front-End::Controllers::- FillingQuestionsController	RFD7.1.4.1 RFD7.1.4.2 RFD7.1.9 RFD7.1.10 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6
Quizzipedia::Front-End::Controllers::- HomeController	RFD5 RFO9 RFO12
Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController	RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.9 RFD7.1.10 RFD7.2.5.1 RFD7.2.5.2



Classe	Requisiti
	RFD7.2.5.3 RFD7.2.5.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO26 RFD35
Quizzipedia::Front-End::Controllers::- InputToListController	RFD7.1.3 RFD7.1.5 RFD7.1.6 RFD7.1.7 RFD7.2.2 RFD7.2.4 RFD7.2.5 RFD7.2.6
Quizzipedia::Front-End::Controllers::- LoginController	RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4
Quizzipedia::Front-End::Controllers::- MenuBarController	RFO3 RFO3.1 RFD10.1 RFD10.6
Quizzipedia::Front-End::Controllers::- MultipleQuestionsController	RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.9 RFD7.1.10 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO26 RFD35



Classe	Requisiti
Quizzipedia::Front-End::Controllers::- NewQuestionsButtonController	RFO7.3
Quizzipedia::Front-End::Controllers::- PasswordForgotController	RFF19 RFF19.2
Quizzipedia::Front-End::Controllers::- ProfileManagementController	RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD10.2 RFD10.3
Quizzipedia::Front-End::Controllers::- QuestionnaireDetailsController	RFD10 RFD10.5 RFD10.5.1
Quizzipedia::Front-End::Controllers::- QuestionnaireManagementController	RFO8 RFD8.1
Quizzipedia::Front-End::Controllers::- QuestionsController	RFD6.2 RFD6.3 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5 RFD11.6 RFD11.7 RFD31 RFD32



Classe	Requisiti
	RFD33 RFD34 RFF38
Quizzipedia::Front-End::Controllers::- QuestionsManagementController	RFO7 RFD7.1 RFD7.2
Quizzipedia::Front-End::Controllers::- QuizEventController	RFD8.2 RFD8.2.1 RFD8.2.2 RFD8.3 RFD8.3.1 RFD8.4 RFD8.5
Quizzipedia::Front-End::Controllers::- RegistrationManagementController	RFD8.8 RFD8.8.1 RFD8.8.2
Quizzipedia::Front-End::Controllers::- ResultsQuestionnaireController	RFO8 RFD8.4
Quizzipedia::Front-End::Controllers::- SearchController	RFO12 RFD18 RFD18.1
Quizzipedia::Front-End::Controllers::- SignUpController	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.6 RFO1.7 RFO1.8
Quizzipedia::Front-End::Controllers::- StatisticsController	RFD10 RFD10.4 RFD10.9 RFD10.10 RFD10.11 RFD12.3.3 RFD12.3.4 RFD12.3.5
Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController	RFD7.1.2 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3



Classe	Requisiti
	RFD7.1.9 RFD7.1.10 RFD7.5 RFD7.6 RFO26 RFD35
Quizzipedia::Front-End::Controllers::- TopicKeywordsController	RFD8.6.2
Quizzipedia::Front-End::Controllers::- TrainingController	RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFD9.4.6 RFD9.5 RFO11 RFO25 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController	RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.9 RFD7.1.10 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO26 RFD35
Quizzipedia::Front-End::Controllers::- UserDetailsController	RFD10 RFD10.7 RFD10.8 RFD12.3 RFD12.3.1 RFD12.3.2
Quizzipedia::Front-End::Directives::- ClickableAnswerDirective	RFO9 RFD9.4



Classe	Requisiti
	RFD9.4.1 RFO11 RFD11.7 RFD11.7.1 RFF38
Quizzipedia::Front-End::Directives::- EliminationAndModifyDirective	RFD8.2 RFD8.2.1 RFD8.2.2 RFD8.3 RFD8.3.1
Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective	RFO9 RFD9.4 RFD9.4.1 RFO11 RFO11.3 RFF38
Quizzipedia::Front-End::Directives::- ExamModalityDirective	RFD8.5
Quizzipedia::Front-End::Directives::- FooterDirective	RFD41
Quizzipedia::Front-End::Directives::- HeaderTextQuestionDirective	RFO9 RFD9.4
Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective	RFD7.1.2 RFD7.1.3 RFD7.1.6 RFD7.1.8 RFD7.2.1.1 RFD7.2.2 RFD7.2.5 RFD7.2.7
Quizzipedia::Front-End::Directives::- InfoQuestionnaireDirective	RFO8
Quizzipedia::Front-End::Directives::- LinkingAnswerDirective	RFO9 RFD9.4 RFD9.4.1 RFO11 RFD11.4 RFD11.4.1 RFF38
Quizzipedia::Front-End::Directives::- LoginBarDirective	RFO2
Quizzipedia::Front-End::Directives::- LogoutBarDirective	RFO3 RFO3.1



Classe	Requisiti
Quizzipedia::Front-End::Directives::- MenuBarDirective	RFD10.1 RFD10.2 RFD10.3 RFD10.6
Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective	RFO9 RFD9.4 RFD9.4.1 RFO11 RFO11.2 RFF38
Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective	RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.3 RFD7.1.4 RFD7.1.5 RFD7.1.6 RFD7.1.7 RFD7.1.8 RFO7.3
Quizzipedia::Front-End::Directives::- OneQuestionDirective	RFD7.2 RFD7.2.1 RFD7.2.2 RFD7.2.3 RFD7.2.4 RFD7.2.5 RFD7.2.6 RFD7.2.7 RFD7.2.8 RFD7.4 RFD7.4.1
Quizzipedia::Front-End::Directives::- ProfileManagementBarDirective	RFD10.1
Quizzipedia::Front-End::Directives::- QuestionnaireDetailsDirective	RFD10 RFD10.5 RFD10.5.1
Quizzipedia::Front-End::Directives::- QuestionnaireDoneDetailsDirective	RFD10
Quizzipedia::Front-End::Directives::- QuestionnaireManagementBarDirective	RFD10.3
Quizzipedia::Front-End::Directives::- QuestionnaireResultsDirective	RFD8.4
Quizzipedia::Front-End::Directives::- QuestionsManagementBarDirective	RFD10.2



Classe	Requisiti
Quizzipedia::Front-End::Directives::- QuestionTextDirective	RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.3 RFD7.1.4 RFD7.1.5 RFD7.1.6 RFD7.1.7 RFD7.1.8 RFD7.2 RFD7.2.1 RFD7.2.2 RFD7.2.3 RFD7.2.4 RFD7.2.5 RFD7.2.6 RFD7.2.7 RFD7.2.8
Quizzipedia::Front-End::Directives::- SearchDirective	RFD5
Quizzipedia::Front-End::Directives::- SignUpBarDirective	RFO1
Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective	RFO9 RFD9.4 RFD9.4.1 RFO11 RFD11.5 RFD11.5.1 RFF38
Quizzipedia::Front-End::Directives::- SortTextAnswerDirective	RFO9 RFD9.4 RFD9.4.1 RFO11 RFD11.6 RFF38
Quizzipedia::Front-End::Directives::- StatisticsDirective	RFD10 RFD10.4 RFD10.4.1 RFD10.9 RFD10.10 RFD10.11 RFD12.3.3 RFD12.3.4 RFD12.3.5
Quizzipedia::Front-End::Directives::- SubscribeResultDirective	RFD18



Classe	Requisiti
	RFD18.1
Quizzipedia::Front-End::Directives::- TopicKeywordsDirective	RFD7.5 RFD7.6 RFD8.6.2
Quizzipedia::Front-End::Directives::- TrainingSetUpDirective	RFO9 RFD9.1 RFD9.2 RFD9.3 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective	RFO9 RFD9.4 RFD9.4.1 RFO11 RFO11.1 RFF38
Quizzipedia::Front-End::Directives::- UserBarDirective	RFD10
Quizzipedia::Front-End::Directives::- UserDetailsDirective	RFD10 RFD10.7 RFD10.8 RFD12.3.1 RFD12.3.2
Quizzipedia::Front-End::Directives::- UserResultsDirective	RFO12 RFD12.2
Quizzipedia::Front-End::Index	RFO39
Quizzipedia::Front-End::Model::- ErrorInfoModel	RFO1.8 RFO2.4 RFD4.7 RFD7.1.10 RFD7.2.9 RFD12.3.6 RFD17 RFF19.3
Quizzipedia::Front-End::Model::- MenuBarModel	RFO3 RFO3.1 RFD10.4
Quizzipedia::Front-End::Model::- QuestionItemModel	RFD6.2 RFD6.3 RFO7



Classe	Requisiti
	RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3



Classe	Requisiti
	RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8.6 RFO9



Classe	Requisiti
Quizzipedia::Front-End::Model::- QuestionnaireModel	RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO7 RFO8 RFD8.1 RFD8.8 RFD8.8.1 RFD10 RFO26
Quizzipedia::Front-End::Model::- TrainingModeModel	RFO9 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Front-End::Model::- UserDetailsModel	RFD10 RFD12.3
Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView	RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView	RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1



Classe	Requisiti
	RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.9 RFD7.1.10 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.8 RFD7.2.9 RFO7.7 RFO23 RFO26 RFD35
Quizzipedia::Front-End::ModelViews::- CreateQuestionnaireModelView	RFO26
Quizzipedia::Front-End::ModelViews::- EditorQMLModelView	RFD7.4 RFD7.4.1 RFO26 RFD35
Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView	RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFD7.5 RFD7.6 RFO25
Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView	RFD7.1.4.1 RFD7.1.4.2 RFD7.1.9 RFD7.1.10 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.8 RFD7.2.9 RFD7.5



Classe	Requisiti
	RFD7.6 RFO23
Quizzipedia::Front-End::ModelViews::- HomeModelView	RFD5 RFO9 RFO12
Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView	RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.9 RFD7.1.10 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::ModelViews::- LoginModelView	RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4
Quizzipedia::Front-End::ModelViews::- MenuBarModelView	RFO3 RFO3.1 RFD10.6
Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView	RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.9 RFD7.1.10 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.8 RFD7.2.9



Classe	Requisiti
	RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::ModelViews::- PasswordForgotModelView	RFF19 RFF19.1
Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView	RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD10.2
Quizzipedia::Front-End::ModelViews::- QuestionnaireDetailsModelView	RFD10 RFD10.4 RFD10.5 RFD10.5.1
Quizzipedia::Front-End::ModelViews::- QuestionnaireManagementModelView	RFO8 RFD8.1
Quizzipedia::Front-End::ModelViews::- QuestionsModelView	RFD6.2 RFD6.3 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5



Classe	Requisiti
	RFD11.6 RFD11.7 RFD31 RFD32 RFD33 RFD34 RFF38
Quizzipedia::Front-End::ModelViews::- QuizEventModelView	RFD8.2 RFD8.2.1 RFD8.2.2 RFD8.3 RFD8.3.1 RFD8.4 RFD8.5
Quizzipedia::Front-End::ModelViews::- RegistrationManagementModelView	RFD8.8 RFD8.8.1 RFD8.8.2
Quizzipedia::Front-End::ModelViews::- ResultsModelView	RFO12 RFD18 RFD18.1
Quizzipedia::Front-End::ModelViews::- ResultsQuestionnaireModelView	RFO8 RFD8.4
Quizzipedia::Front-End::ModelViews::- SignUpModelView	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8
Quizzipedia::Front-End::ModelViews::- StatisticsModelView	RFD10.9 RFD10.10 RFD10.11
Quizzipedia::Front-End::ModelViews::- StringsSortingQuestionsModelView	RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.9 RFD7.1.10 RFO23
Quizzipedia::Front-End::ModelViews::- TopicKeywordsModelView	RFD8.6.2



Classe	Requisiti
Quizzipedia::Front-End::ModelViews::- TrainingModelView	RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFD9.4.6 RFD9.5
Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView	RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.9 RFD7.1.10 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23
Quizzipedia::Front-End::ModelViews::- UserDetailsModelView	RFD10.7 RFD10.8
Quizzipedia::Front-End::Services::- AuthService	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFO3.1 RFF19 RFF19.2
Quizzipedia::Front-End::Services::- LangService	RFF40
Quizzipedia::Front-End::Services::- QuestionsService	RFD6.2



Classe	Requisiti
	RFD6.3 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1



Classe	Requisiti
	RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFD8.6.2



Classe	Requisiti
	RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.2 RFO8.7.2.1 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.5 RFD9.4.6 RFD9.5 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5 RFD11.6 RFD11.7 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO23.8 RFO26 RFD31 RFD32 RFD33 RFD34 RFD35 RFF38
Quizzipedia::Front-End::Services::- QuizService	RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO8 RFD8.1 RFO8.6 RFO8.6.4 RFD8.6.4.1 RFO8.6.4.2 RFO8.7



Classe	Requisiti
	RFO8.7.1 RFO8.7.1.1 RFO8.7.2 RFO8.7.2.1 RFD8.8 RFD8.8.2 RFD10 RFD10.5 RFD10.5.1 RFO24 RFO25 RFO26 RFD28
Quizzipedia::Front-End::Services::- SearchService	RFD5 RFO12 RFD18 RFD18.1
Quizzipedia::Front-End::Services::- StatisticsService	RFD10 RFD10.4 RFD10.4.1 RFD10.9 RFD10.10 RFD10.11 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD29
Quizzipedia::Front-End::Services::- UserDetailsService	RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD10 RFD10.7 RFD10.8 RFD12.3



Classe	Requisiti
	RFD12.3.1 RFD12.3.2
Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView	RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::Views::- ConnectionQuestionsView	RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.9 RFD7.1.10 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.8 RFD7.2.9 RFO26



Classe	Requisiti
	RFD35
Quizzipedia::Front-End::Views::- CreateQuestionnaireView	RFO8.6 RFO8.6.1 RFD8.6.3 RFO8.6.4 RFD8.6.4.1 RFO8.6.4.2 RFO8.7 RFO22 RFO24 RFO26
Quizzipedia::Front-End::Views::- EditorQMLView	RFO7.3.1 RFD7.4 RFD7.4.1 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO26 RFD35
Quizzipedia::Front-End::Views::- FillingQuestionnaireView	RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO20 RFO21 RFO25
Quizzipedia::Front-End::Views::- FillingQuestionsView	RFD7.1.4.1 RFD7.1.4.2 RFD7.1.9 RFD7.1.10 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23
Quizzipedia::Front-End::Views::- HomeView	RFD5



Classe	Requisiti
	RFO9 RFO12
Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView	RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.9 RFD7.1.10 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::Views::- LoginView	RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4
Quizzipedia::Front-End::Views::- MultipleQuestionsView	RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.9 RFD7.1.10 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::Views::- OtherUserView	RFD12.3



Classe	Requisiti
Quizzipedia::Front-End::Views::- PasswordForgotView	RFF19 RFF19.1 RFF19.2
Quizzipedia::Front-End::Views::- ProfileManagementView	RFD10.2
Quizzipedia::Front-End::Views::- QuestionnaireManagementView	RFO8 RFD8.1
Quizzipedia::Front-End::Views::- QuestionsManagementView	RFO7
Quizzipedia::Front-End::Views::- RegistrationManagementView	RFD8.8 RFD8.8.1 RFD8.8.2
Quizzipedia::Front-End::Views::- ResultsQuestionnaireView	RFO8 RFD8.4
Quizzipedia::Front-End::Views::- ResultsView	RFO12 RFD12.1 RFD12.3.6 RFD17
Quizzipedia::Front-End::Views::- SignUpView	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8
Quizzipedia::Front-End::Views::- StringsSortingQuestionsView	RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.9 RFD7.1.10 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35



Classe	Requisiti
Quizzipedia::Front-End::Views::- TrainingView	RFO9 RFD9.4.5 RFD9.4.6 RFO25 RFD31 RFD32 RFD33 RFD34
Quizzipedia::Front-End::Views::- TrueFalseQuestionsView	RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.9 RFD7.1.10 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.8 RFD7.2.9 RFD7.5 RFD7.6 RFO23 RFO26 RFD35
Quizzipedia::Front-End::Views::- UserView	RFD10 RFD10.1

Tabella 1: Tracciamento Classi-Requisiti



6 Tracciamento Requisiti-Classi

Requisito	Classi
RFO1	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::Directives::- SignUpBarDirective Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView</pre>
RFO1.1	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView</pre>
RFO1.2	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService</pre>



Requisito	Classi
	Quizzipedia::Front-End::Views::- SignUpView
RFO1.3	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView
RFO1.4	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView
RFO1.5	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView
RFO1.6	Quizzipedia::Back-End::App::Controller::- UserController



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView</pre>
RFO1.7	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView</pre>
RFO1.8	<pre>Quizzipedia::Back-End::App::Controller::- ErrorsHandler Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError Quizzipedia::Front-End::Controllers::- SignUpController Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::ModelViews::- SignUpModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- SignUpView</pre>
RFO2	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::AppRun Quizzipedia::Front-End::Controllers::- LoginController Quizzipedia::Front-End::Directives::- LoginBarDirective Quizzipedia::Front-End::ModelViews::- LoginModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- LoginView</pre>
RFO2.1	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- LoginController Quizzipedia::Front-End::ModelViews::- LoginModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- LoginView</pre>
RFO2.2	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- LoginController Quizzipedia::Front-End::ModelViews::- LoginModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- LoginView</pre>
RFO2.3	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- LoginController Quizzipedia::Front-End::ModelViews::- LoginModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- LoginView</pre>
RFO2.4	<pre>Quizzipedia::Back-End::App::Controller::- ErrorsHandler Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError Quizzipedia::Front-End::Controllers::- LoginController Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::ModelViews::- LoginModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- LoginView</pre>
RFO3	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::AppRun Quizzipedia::Front-End::Controllers::- MenuBarController Quizzipedia::Front-End::Directives::- LogoutBarDirective Quizzipedia::Front-End::Model::- MenuBarModel Quizzipedia::Front-End::ModelViews::- MenuBarModelView Quizzipedia::Front-End::Services::- AuthService</pre>
RFO3.1	<pre>Quizzipedia::Front-End::Controllers::- MenuBarController Quizzipedia::Front-End::Directives::- LogoutBarDirective Quizzipedia::Front-End::Model::- MenuBarModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::ModelViews::- MenuBarModelView Quizzipedia::Front-End::Services::- AuthService</pre>
RFD4	<pre>Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService</pre>
RFD4.1	<pre>Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService</pre>
RFD4.2	<pre>Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService</pre>
RFD4.3	<pre>Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService</pre>
RFD4.4	<pre>Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.5	Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.5.1	Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.5.2	Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.5.3	Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.6	Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView



Requisito	Classi
	Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.7	Quizzipedia::Back-End::App::Controller::- ErrorsHandler Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.8	Quizzipedia::Back-End::App::Model::- UserProModel Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.8.1	Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.8.2	Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.9	Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD4.9.1	Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Services::- UserDetailsService



Requisito	Classi
RFD5	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- HomeController Quizzipedia::Front-End::Directives::- SearchDirective Quizzipedia::Front-End::ModelViews::- HomeModelView Quizzipedia::Front-End::Services::- SearchService Quizzipedia::Front-End::Views::- HomeView</pre>
RFO6	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView</pre>
RFD6.1	<pre>Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView</pre>
RFD6.2	<pre>Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Model::- QuestionnaireModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView</pre>
RFD6.3	<pre>Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView</pre>
RFD6.4	<pre>Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView</pre>
RFO6.5	<pre>Quizzipedia::Back-End::App::Controller::- SummaryController Quizzipedia::Back-End::App::Model::- SummaryModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView
RFO7	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- QuestionsManagementController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- QuestionsManagementView
RFD7.1	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- QuestionsManagementController Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService
RFD7.1.1	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService



Requisito	Classi
RFD7.1.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.1.2.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD7.1.2.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>



Requisito	Classi
RFD7.1.2.3	<pre>Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD7.1.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre> <pre>Quizzipedia::Back-End::App::Model::- QuestionModel</pre> <pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre> <pre>Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController</pre> <pre>Quizzipedia::Front-End::Model::- QuestionItemModel</pre> <pre>Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView</pre> <pre>Quizzipedia::Front-End::Services::- QuestionsService</pre> <pre>Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD7.1.3.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre> <pre>Quizzipedia::Back-End::App::Model::- QuestionModel</pre> <pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre> <pre>Quizzipedia::Front-End::Controllers::- InputToListController</pre> <pre>Quizzipedia::Front-End::Controllers::- MultipleQuestionsController</pre> <pre>Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective</pre> <pre>Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective</pre> <pre>Quizzipedia::Front-End::Directives::- QuestionTextDirective</pre> <pre>Quizzipedia::Front-End::Model::- QuestionItemModel</pre> <pre>Quizzipedia::Front-End::Services::- QuestionsService</pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView</pre>
RFD7.1.3.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView</pre>
RFD7.1.3.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView</pre>
RFD7.1.3.3.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>



Requisito	Classi
	Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.1.3.3.2	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.1.3.4	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.1.4	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService
RFD7.1.4.1	Quizzipedia::Back-End::App::Controller::- QuestionController



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- FillingQuestionsView</pre>
RFD7.1.4.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- FillingQuestionsView</pre>
RFD7.1.5	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- InputToListController Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.1.5.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.2.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.2.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.2.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.2.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.3.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.4.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController</pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.4.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.1.5.4.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.5.4.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>



Requisito	Classi
RFD7.1.6	<pre>Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.1.6.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- InputToListController Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.1.6.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>



Requisito	Classi
RFD7.1.6.3	<pre>Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>
RFD7.1.7	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>
RFD7.1.7.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>



Requisito	Classi
RFD7.1.7.2	<pre>Quizzipedia::Front-End::Views::- StringsSortingQuestionsView</pre>
RFD7.1.7.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre> <pre>Quizzipedia::Back-End::App::Model::- QuestionModel</pre> <pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre> <pre>Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController</pre> <pre>Quizzipedia::Front-End::Model::- QuestionItemModel</pre> <pre>Quizzipedia::Front-End::ModelViews::- StringsSortingQuestionsModelView</pre> <pre>Quizzipedia::Front-End::Services::- QuestionsService</pre> <pre>Quizzipedia::Front-End::Views::- StringsSortingQuestionsView</pre>
RFD7.1.8	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre> <pre>Quizzipedia::Back-End::App::Model::- QuestionModel</pre> <pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre> <pre>Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController</pre> <pre>Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective</pre> <pre>Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective</pre> <pre>Quizzipedia::Front-End::Directives::- QuestionTextDirective</pre> <pre>Quizzipedia::Front-End::Model::- QuestionItemModel</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView
RFD7.1.8.1	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView
RFD7.1.8.2	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView
RFD7.1.8.3	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService



Requisito	Classi
	Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView
RFD7.1.8.4	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView
RFD7.1.9	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView



Requisito	Classi
	<pre> Quizzipedia::Front-End::ModelViews::- StringsSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- ConnectionQuestionsView Quizzipedia::Front-End::Views::- FillingQuestionsView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView </pre>
RFD7.1.10	<pre> Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView </pre>



Requisito	Classi
	<pre> Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::ModelViews::- StringsSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- ConnectionQuestionsView Quizzipedia::Front-End::Views::- FillingQuestionsView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView </pre>
RFD7.2	<pre> Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- QuestionsManagementController Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService </pre>
RFD7.2.1	<pre> Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective </pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.2.1.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD7.2.1.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD7.2.1.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrueFalseQuestionsView
RFD7.2.2	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- InputToListController Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService
RFD7.2.2.1	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.2.2.2	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.2.2.3	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.2.2.3.1	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView
RFD7.2.2.3.2	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel



Requisito	Classi
	<pre>Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView</pre>
RFD7.2.2.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- MultipleQuestionsView</pre>
RFD7.2.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.2.3.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- FillingQuestionsView</pre>



Requisito	Classi
RFD7.2.3.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- FillingQuestionsView</pre>
RFD7.2.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- InputToListController Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.2.4.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.2.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.2.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.2.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre>



Requisito	Classi
	Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView
RFD7.2.4.2.4	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView
RFD7.2.4.3	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService
RFD7.2.4.3.1	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService



Requisito	Classi
RFD7.2.4.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.4.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.4.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.4.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.4.4.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ConnectionQuestionsView</pre>
RFD7.2.5	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- InputToListController Quizzipedia::Front-End::Directives::- ImageInTheQuestionDirective Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.2.5.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>
RFD7.2.5.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>
RFD7.2.5.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>
RFD7.2.5.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView</pre>
RFD7.2.6	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- InputToListController Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD7.2.6.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- StringsSortingQuestionsView</pre>
RFD7.2.6.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>



Requisito	Classi
RFD7.2.6.3	<pre>Quizzipedia::Front-End::Views::- StringsSortingQuestionsView</pre>
RFD7.2.7	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- StringsSortingQuestionsView</pre>
RFD7.2.7.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView</pre>
RFD7.2.7.2	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel</pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView</pre>
RFD7.2.7.3	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView</pre>
RFD7.2.7.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView</pre>
RFD7.2.8	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter</pre>



Requisito	Classi
	Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Directives::- QuestionTextDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- ConnectionQuestionsView Quizzipedia::Front-End::Views::- FillingQuestionsView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView
RFD7.2.9	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel



Requisito	Classi
	<pre> Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- ConnectionQuestionsView Quizzipedia::Front-End::Views::- FillingQuestionsView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView </pre>
RFO7.3	<pre> Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel </pre>



Requisito	Classi
	<pre>Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- NewQuestionsButtonController Quizzipedia::Front-End::Directives::- NewQuestionButtonDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFO7.3.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView</pre>
RFD7.4	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Directives::- OneQuestionDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- EditorQMLModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView</pre>
RFD7.4.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Directives::- OneQuestionDirective</pre>



Requisito	Classi
	Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- EditorQMLModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFD7.5	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Directives::- TopicKeywordsDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- FillingQuestionsView



Requisito	Classi
	<pre>Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD7.6	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController Quizzipedia::Front-End::Controllers::- FillingQuestionsController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Directives::- TopicKeywordsDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- FillingQuestionsView</pre>



Requisito	Classi
	Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView
RFO7.7	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService
RFO8	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuestionnaireManagementController Quizzipedia::Front-End::Controllers::- ResultsQuestionnaireController Quizzipedia::Front-End::Directives::- InfoQuestionnaireDirective Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- QuestionnaireManagementModelView Quizzipedia::Front-End::ModelViews::- ResultsQuestionnaireModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- QuestionnaireManagementView Quizzipedia::Front-End::Views::- ResultsQuestionnaireView
RFD8.1	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter



Requisito	Classi
	<pre>Quizzipedia::Front-End::Controllers::- QuestionnaireManagementController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- QuestionnaireManagementModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- QuestionnaireManagementView</pre>
RFD8.2	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Directives::- EliminationAndModifyDirective Quizzipedia::Front-End::ModelViews::- QuizEventModelView</pre>
RFD8.2.1	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Directives::- EliminationAndModifyDirective Quizzipedia::Front-End::ModelViews::- QuizEventModelView</pre>
RFD8.2.2	<pre>Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Directives::- EliminationAndModifyDirective Quizzipedia::Front-End::ModelViews::- QuizEventModelView</pre>
RFD8.3	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Directives::- EliminationAndModifyDirective</pre>



Requisito	Classi
	Quizzipedia::Front-End::ModelViews::- QuizEventModelView
RFD8.3.1	Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Directives::- EliminationAndModifyDirective Quizzipedia::Front-End::ModelViews::- QuizEventModelView
RFD8.4	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Controllers::- ResultsQuestionnaireController Quizzipedia::Front-End::Directives::- QuestionnaireResultsDirective Quizzipedia::Front-End::ModelViews::- QuizEventModelView Quizzipedia::Front-End::ModelViews::- ResultsQuestionnaireModelView Quizzipedia::Front-End::Views::- ResultsQuestionnaireView
RFD8.5	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuizEventController Quizzipedia::Front-End::Directives::- ExamModalityDirective Quizzipedia::Front-End::ModelViews::- QuizEventModelView
RFO8.6	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- CreateQuestionnaireView



Requisito	Classi
RFO8.6.1	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Views::- CreateQuestionnaireView</pre>
RFD8.6.2	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Controllers::- TopicKeywordsController Quizzipedia::Front-End::Directives::- TopicKeywordsDirective Quizzipedia::Front-End::ModelViews::- TopicKeywordsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD8.6.3	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Views::- CreateQuestionnaireView</pre>
RFO8.6.4	<pre>Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- CreateQuestionnaireView</pre>
RFD8.6.4.1	<pre>Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- CreateQuestionnaireView</pre>
RFO8.6.4.2	<pre>Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- CreateQuestionnaireView
RFO8.7	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- CreateQuestionnaireView
RFO8.7.1	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService
RFO8.7.1.1	Quizzipedia::Back-End::App::Controller::- TopicController Quizzipedia::Back-End::App::Model::- TopicModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService
RFO8.7.1.1.1	Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController
RFO8.7.1.1.2	Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController
RFO8.7.2	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel



Requisito	Classi
	Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService
RFO8.7.2.1	Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService
RFD8.8	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- RegistrationManagementController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- RegistrationManagementModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- RegistrationManagementView
RFD8.8.1	Quizzipedia::Front-End::Controllers::- RegistrationManagementController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- RegistrationManagementModelView Quizzipedia::Front-End::Views::- RegistrationManagementView
RFD8.8.2	Quizzipedia::Front-End::Controllers::- RegistrationManagementController Quizzipedia::Front-End::ModelViews::- RegistrationManagementModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- RegistrationManagementView
RFO9	Quizzipedia::Front-End::Controllers::- HomeController Quizzipedia::Front-End::Controllers::- QuestionsController



Requisito	Classi
	<pre>Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- ClickableAnswerDirective Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective Quizzipedia::Front-End::Directives::- HeaderTextQuestionDirective Quizzipedia::Front-End::Directives::- LinkingAnswerDirective Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective Quizzipedia::Front-End::Directives::- SortTextAnswerDirective Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective Quizzipedia::Front-End::Model::- QuestionItemModel Quizzipedia::Front-End::Model::- TrainingModeModel Quizzipedia::Front-End::ModelViews::- HomeModelView Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- HomeView Quizzipedia::Front-End::Views::- TrainingView </pre>
RFD9.1	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService </pre>
RFD9.2	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController</pre>



Requisito	Classi
	Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService
RFD9.3	Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService
RFD9.4	Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- ClickableAnswerDirective Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective Quizzipedia::Front-End::Directives::- HeaderTextQuestionDirective Quizzipedia::Front-End::Directives::- LinkingAnswerDirective Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective Quizzipedia::Front-End::Directives::- SortTextAnswerDirective Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService
RFD9.4.1	Quizzipedia::Front-End::Controllers::- QuestionsController



Requisito	Classi
	<pre>Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- ClickableAnswerDirective Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective Quizzipedia::Front-End::Directives::- LinkingAnswerDirective Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective Quizzipedia::Front-End::Directives::- SortTextAnswerDirective Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView</pre>
RFD9.4.5	<pre>Quizzipedia::Back-End::App::Controller::- TopicController Quizzipedia::Back-End::App::Model::- TopicModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrainingView</pre>
RFD9.4.6	<pre>Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::ModelViews::- TrainingModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrainingView</pre>
RFD9.5	<pre>Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::ModelViews::- TrainingModelView</pre>



Requisito	Classi
RFD10	Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Controllers::- QuestionnaireDetailsController Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Controllers::- UserDetailsController Quizzipedia::Front-End::Directives::- QuestionnaireDetailsDirective Quizzipedia::Front-End::Directives::- QuestionnaireDoneDetailsDirective Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::Directives::- UserBarDirective Quizzipedia::Front-End::Directives::- UserDetailsDirective Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::Model::- UserDetailsModel Quizzipedia::Front-End::ModelViews::- QuestionnaireDetailsModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Services::- StatisticsService Quizzipedia::Front-End::Services::- UserDetailsService Quizzipedia::Front-End::Views::- UserView
RFD10.1	Quizzipedia::Front-End::Controllers::- MenuBarController Quizzipedia::Front-End::Directives::- MenuBarDirective Quizzipedia::Front-End::Directives::- ProfileManagementBarDirective Quizzipedia::Front-End::Views::- UserView
RFD10.2	Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::Directives::- MenuBarDirective Quizzipedia::Front-End::Directives::- QuestionsManagementBarDirective Quizzipedia::Front-End::ModelViews::- ProfileManagementModelView Quizzipedia::Front-End::Views::- ProfileManagementView



Requisito	Classi
RFD10.3	<pre>Quizzipedia::Front-End::Controllers::- ProfileManagementController Quizzipedia::Front-End::Directives::- MenuBarDirective Quizzipedia::Front-End::Directives::- QuestionnaireManagementBarDirective</pre>
RFD10.4	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::Model::- MenuBarModel Quizzipedia::Front-End::ModelViews::- QuestionnaireDetailsModelView Quizzipedia::Front-End::Services::- StatisticsService</pre>
RFD10.4.1	<pre>Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::Services::- StatisticsService</pre>
RFD10.5	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- QuestionnaireDetailsController Quizzipedia::Front-End::Directives::- QuestionnaireDetailsDirective Quizzipedia::Front-End::ModelViews::- QuestionnaireDetailsModelView Quizzipedia::Front-End::Services::- QuizService</pre>
RFD10.5.1	<pre>Quizzipedia::Front-End::Controllers::- QuestionnaireDetailsController Quizzipedia::Front-End::Directives::- QuestionnaireDetailsDirective Quizzipedia::Front-End::ModelViews::- QuestionnaireDetailsModelView</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuizService
RFD10.6	Quizzipedia::Front-End::Controllers::- MenuBarController Quizzipedia::Front-End::Directives::- MenuBarDirective Quizzipedia::Front-End::ModelViews::- MenuBarModelView
RFD10.7	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- UserDetailsController Quizzipedia::Front-End::Directives::- UserDetailsDirective Quizzipedia::Front-End::ModelViews::- UserDetailsModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD10.8	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- UserDetailsController Quizzipedia::Front-End::Directives::- UserDetailsDirective Quizzipedia::Front-End::ModelViews::- UserDetailsModelView Quizzipedia::Front-End::Services::- UserDetailsService
RFD10.9	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController



Requisito	Classi
	Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::ModelViews::- StatisticsModelView Quizzipedia::Front-End::Services::- StatisticsService
RFD10.10	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::ModelViews::- StatisticsModelView Quizzipedia::Front-End::Services::- StatisticsService
RFD10.11	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::ModelViews::- StatisticsModelView Quizzipedia::Front-End::Services::- StatisticsService
RFO11	Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- ClickableAnswerDirective Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective Quizzipedia::Front-End::Directives::- LinkingAnswerDirective Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective



Requisito	Classi
	<pre>Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective Quizzipedia::Front-End::Directives::- SortTextAnswerDirective Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFO11.1	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFO11.2	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFO11.3	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD11.4	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- LinkingAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFD11.4.1	<pre>Quizzipedia::Front-End::Directives::- LinkingAnswerDirective</pre>
RFD11.5	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuestionsService
RFD11.5.1	Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective
RFD11.6	Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- SortTextAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService
RFD11.7	Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- ClickableAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService
RFD11.7.1	Quizzipedia::Front-End::Directives::- ClickableAnswerDirective
RFO12	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- HomeController Quizzipedia::Front-End::Controllers::- SearchController Quizzipedia::Front-End::Directives::- UserResultsDirective Quizzipedia::Front-End::ModelViews::- HomeModelView Quizzipedia::Front-End::ModelViews::- ResultsModelView Quizzipedia::Front-End::Services::- SearchService Quizzipedia::Front-End::Views::- HomeView Quizzipedia::Front-End::Views::- ResultsView
RFD12.1	Quizzipedia::Front-End::Views::- ResultsView
RFD12.2	Quizzipedia::Front-End::Directives::- UserResultsDirective



Requisito	Classi
RFD12.3	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- UserDetailsController Quizzipedia::Front-End::Model::- UserDetailsModel Quizzipedia::Front-End::Services::- UserDetailsService Quizzipedia::Front-End::Views::- OtherUserView</pre>
RFD12.3.1	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- UserDetailsController Quizzipedia::Front-End::Directives::- UserDetailsDirective Quizzipedia::Front-End::Services::- UserDetailsService</pre>
RFD12.3.2	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- UserDetailsController Quizzipedia::Front-End::Directives::- UserDetailsDirective Quizzipedia::Front-End::Services::- UserDetailsService</pre>
RFD12.3.3	<pre>Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel</pre>



Requisito	Classi
	Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::Services::- StatisticsService
RFD12.3.4	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::Services::- StatisticsService
RFD12.3.5	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- StatisticsController Quizzipedia::Front-End::Directives::- StatisticsDirective Quizzipedia::Front-End::Services::- StatisticsService
RFD12.3.6	Quizzipedia::Back-End::App::Controller::- ErrorsHandler Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::Views::- ResultsView
RFF13	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel



Requisito	Classi
	Quizzipedia::Back-End::App::Routers::- UserRouter
RFF14	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter
RFF15	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter
RFF16	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: AuthenticationController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter
RFD17	Quizzipedia::Back-End::App::Controller::- ErrorsHandler Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError Quizzipedia::Front-End::Model::- ErrorInfoModel Quizzipedia::Front-End::Views::- ResultsView
RFD18	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- SearchController Quizzipedia::Front-End::Directives::- SubscribeResultDirective Quizzipedia::Front-End::ModelViews::- ResultsModelView Quizzipedia::Front-End::Services::- SearchService
RFD18.1	Quizzipedia::Front-End::Controllers::- SearchController



Requisito	Classi
	Quizzipedia::Front-End::Directives::- SubscribeResultDirective Quizzipedia::Front-End::ModelViews::- ResultsModelView Quizzipedia::Front-End::Services::- SearchService
RFF19	Quizzipedia::Back-End::App::Controller::- UserController Quizzipedia::Back-End::App::Controller::- Users:: UserManagementController Quizzipedia::Back-End::App::Model::- UserModel Quizzipedia::Back-End::App::Routers::- UserRouter Quizzipedia::Front-End::Controllers::- PasswordForgotController Quizzipedia::Front-End::ModelViews::- PasswordForgotModelView Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- PasswordForgotView
RFF19.1	Quizzipedia::Front-End::ModelViews::- PasswordForgotModelView Quizzipedia::Front-End::Views::- PasswordForgotView
RFF19.2	Quizzipedia::Front-End::Controllers::- PasswordForgotController Quizzipedia::Front-End::Services::- AuthService Quizzipedia::Front-End::Views::- PasswordForgotView
RFF19.3	Quizzipedia::Back-End::App::Controller::- ErrorsHandler Quizzipedia::Back-End::App::Controller::- Errors::QuizziPediaError Quizzipedia::Front-End::Model::- ErrorInfoModel
RFO20	Quizzipedia::Back-End::Server Quizzipedia::Front-End::Views::- FillingQuestionnaireView
RFO21	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Views::- FillingQuestionnaireView



Requisito	Classi
RFO22	<pre>Quizzipedia::Back-End::App::Controller::- TopicController Quizzipedia::Back-End::App::Model::- TopicModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Views::- CreateQuestionnaireView</pre>
RFO23	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- FillingQuestionsModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::ModelViews::- StringsSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- TrueFalseQuestionsModelView Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- EditorQMLView Quizzipedia::Front-End::Views::- FillingQuestionsView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFO23.1	<pre>Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController</pre>



Requisito	Classi
	Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFO23.2	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFO23.3	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFO23.4	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFD23.5	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Services::- QuestionsService



Requisito	Classi
	Quizzipedia::Front-End::Views::- EditorQMLView
RFD23.6	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFO23.7	Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- EditorQMLView
RFO23.8	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Model::- TopicModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Services::- QuestionsService
RFO24	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- CreateQuestionnaireView
RFO25	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- FillingQuestionnaireController



Requisito	Classi
	Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::ModelViews::- FillingQuestionnaireModelView Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- FillingQuestionnaireView Quizzipedia::Front-End::Views::- TrainingView
RFO26	Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- CreateQuestionnaireController Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::Model::- QuestionnaireModel Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- CreateQuestionnaireModelView Quizzipedia::Front-End::ModelViews::- EditorQMLModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView



Requisito	Classi
	<pre>Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Services::- QuizService Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- ConnectionQuestionsView Quizzipedia::Front-End::Views::- CreateQuestionnaireView Quizzipedia::Front-End::Views::- EditorQMLView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFD28	<pre>Quizzipedia::Back-End::App::Controller::- SummaryController Quizzipedia::Back-End::App::Model::- SummaryModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Services::- QuizService</pre>
RFD29	<pre>Quizzipedia::Back-End::App::Controller::- TopicController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Model::- TopicModel Quizzipedia::Front-End::Services::- StatisticsService</pre>
RFD31	<pre>Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective</pre>



Requisito	Classi
	Quizzipedia::Front-End::Model::- TrainingModeModel Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrainingView
RFD32	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::Model::- TrainingModeModel Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrainingView
RFD33	Quizzipedia::Back-End::App::Controller::- QuizController Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::Model::- TrainingModeModel Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrainingView
RFD34	Quizzipedia::Back-End::App::Controller::- QuizController



Requisito	Classi
	<pre> Quizzipedia::Back-End::App::Model::- QuizModel Quizzipedia::Back-End::App::Routers::- QuizRouter Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Controllers::- TrainingController Quizzipedia::Front-End::Directives::- TrainingSetUpDirective Quizzipedia::Front-End::Model::- TrainingModeModel Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService Quizzipedia::Front-End::Views::- TrainingView </pre>
RFD35	<pre> Quizzipedia::Back-End::App::Controller::- QuestionController Quizzipedia::Back-End::App::Model::- QuestionModel Quizzipedia::Back-End::App::Routers::- QuestionRouter Quizzipedia::Front-End::Controllers::- ClickableAreaQuestionsController Quizzipedia::Front-End::Controllers::- ConnectionQuestionsController Quizzipedia::Front-End::Controllers::- EditorQMLController Quizzipedia::Front-End::Controllers::- ImagesSortingQuestionsController Quizzipedia::Front-End::Controllers::- MultipleQuestionsController Quizzipedia::Front-End::Controllers::- StringsSortingQuestionsController Quizzipedia::Front-End::Controllers::- TrueFalseQuestionsController Quizzipedia::Front-End::ModelViews::- ClickableAreaQuestionsModelView Quizzipedia::Front-End::ModelViews::- ConnectionQuestionsModelView Quizzipedia::Front-End::ModelViews::- EditorQMLModelView Quizzipedia::Front-End::ModelViews::- ImagesSortingQuestionsModelView Quizzipedia::Front-End::ModelViews::- MultipleQuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService </pre>



Requisito	Classi
	<pre>Quizzipedia::Front-End::Views::- ClickableAreaQuestionsView Quizzipedia::Front-End::Views::- ConnectionQuestionsView Quizzipedia::Front-End::Views::- EditorQMLView Quizzipedia::Front-End::Views::- ImagesSortingQuestionsView Quizzipedia::Front-End::Views::- MultipleQuestionsView Quizzipedia::Front-End::Views::- StringsSortingQuestionsView Quizzipedia::Front-End::Views::- TrueFalseQuestionsView</pre>
RFF38	<pre>Quizzipedia::Front-End::Controllers::- QuestionsController Quizzipedia::Front-End::Directives::- ClickableAnswerDirective Quizzipedia::Front-End::Directives::- EmptySpaceAnswerDirective Quizzipedia::Front-End::Directives::- LinkingAnswerDirective Quizzipedia::Front-End::Directives::- MultipleChoiceAnswerDirective Quizzipedia::Front-End::Directives::- SortImagesAnswerDirective Quizzipedia::Front-End::Directives::- SortTextAnswerDirective Quizzipedia::Front-End::Directives::- TrueFalseAnswerDirective Quizzipedia::Front-End::ModelViews::- QuestionsModelView Quizzipedia::Front-End::Services::- QuestionsService</pre>
RFO39	<pre>Quizzipedia::Front-End::AppRouter Quizzipedia::Front-End::Index</pre>
RFF40	<pre>Quizzipedia::Back-End::App::Controller::- LangController Quizzipedia::Back-End::App::Model::- LangModel Quizzipedia::Back-End::App::Routers::- LangRouter Quizzipedia::Front-End::Services::- LangService</pre>
RFD41	<pre>Quizzipedia::Front-End::Directives::- FooterDirective</pre>
RFO42	<pre>Quizzipedia::Back-End::App::Controller::- Users::SessionController</pre>
RFO43	<pre>Quizzipedia::Back-End::App::Controller::- NotFoundHandler</pre>



Requisito	Classi
Tabella 2: Tracciamento Requisiti-Classi	



7 Tracciamento Componenti-Requisiti

Componente	Requisiti
Quizzipedia::Back-End	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.9 RFD5 RFO6 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2



Componente	Requisiti
	RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3



Componente	Requisiti
	RFD7.2.4.2.4
	RFD7.2.4.3
	RFD7.2.4.3.1
	RFD7.2.4.4
	RFD7.2.4.4.1
	RFD7.2.4.4.2
	RFD7.2.4.4.3
	RFD7.2.4.4.4
	RFD7.2.5
	RFD7.2.5.1
	RFD7.2.5.2
	RFD7.2.5.3
	RFD7.2.5.4
	RFD7.2.6
	RFD7.2.6.1
	RFD7.2.6.2
	RFD7.2.6.3
	RFD7.2.7
	RFD7.2.7.1
	RFD7.2.7.2
	RFD7.2.7.3
	RFD7.2.7.4
	RFD7.2.8
	RFD7.2.9
	RFO7.3
	RFO7.3.1
	RFD7.4
	RFD7.4.1
	RFD7.5
	RFD7.6
	RFO7.7
	RFO8
	RFD8.1
	RFD8.2
	RFD8.2.1
	RFD8.3
	RFD8.4
	RFD8.5
	RFO8.6
	RFO8.6.1
	RFD8.6.2
	RFD8.6.3
	RFO8.7
	RFO8.7.1
	RFO8.7.1.1
	RFO8.7.2
	RFD8.8
	RFD9.4.5
	RFD10.4
	RFD10.5



Componente	Requisiti
	RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD12.3.6 RFF13 RFF14 RFF15 RFF16 RFD17 RFD18 RFF19 RFF19.3 RFO20 RFO21 RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO24 RFO25 RFO26 RFD28 RFD29 RFD31 RFD32 RFD33 RFD34 RFD35 RFF40 RFO42 RFO43
Quizzipedia::Back-End::App	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4



Componente	Requisiti
	RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.9 RFD5 RFO6 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4



Componente	Requisiti
	RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3



Componente	Requisiti
	RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.3 RFD8.4 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.2 RFD8.8 RFD9.4.5 RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3



Componente	Requisiti
	RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD12.3.6 RFF13 RFF14 RFF15 RFF16 RFD17 RFD18 RFF19 RFF19.3 RFO21 RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO24 RFO25 RFO26 RFD28 RFD29 RFD31 RFD32 RFD33 RFD34 RFD35 RFF40 RFO42 RFO43
Quizzipedia::Back-End::App::Controller	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3



Componente	Requisiti
	RFO2.4 RFO3 RFD4.7 RFD5 RFO6 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4



Componente	Requisiti
	RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4



Componente	Requisiti
	RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.3 RFD8.4 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.2 RFD8.8 RFD9.4.5 RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD12.3.6 RFF13 RFF14 RFF15 RFF16 RFD17 RFD18 RFF19 RFF19.3 RFO21 RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3



Componente	Requisiti
	RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO24 RFO25 RFO26 RFD28 RFD29 RFD31 RFD32 RFD33 RFD34 RFD35 RFF40 RFO42 RFO43
Quizzipedia::Back-End::App::Controller::- Errors	RFO1.8 RFO2.4 RFD4.7 RFD12.3.6 RFD17 RFF19.3
Quizzipedia::Back-End::App::Controller::- Users	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO3 RFD8.4 RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2



Componente	Requisiti
	RFD12.3.3 RFD12.3.4 RFD12.3.5 RFF13 RFF14 RFF15 RFF16 RFF19 RFO42
Quizzipedia::Back-End::App::Model	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2 RFO2.1 RFO2.2 RFO2.3 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.8 RFD4.9 RFD5 RFO6 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4



Componente	Requisiti
	RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2



Componente	Requisiti
	RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.3 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.2 RFD8.8 RFD9.4.5



Componente	Requisiti
	RFD10.4 RFD10.5 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFF13 RFF14 RFF15 RFF16 RFD18 RFF19 RFO21 RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO24 RFO25 RFO26 RFD28 RFD29 RFD31 RFD32 RFD33 RFD34 RFD35 RFF40
Quizzipedia::Back-End::App::Routers	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO2



Componente	Requisiti
	RFO2.1 RFO2.2 RFO2.3 RFO3 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.9 RFD5 RFO6 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6



Componente	Requisiti
	RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1



Componente	Requisiti
	RFD7.2.6.2
	RFD7.2.6.3
	RFD7.2.7
	RFD7.2.7.1
	RFD7.2.7.2
	RFD7.2.7.3
	RFD7.2.7.4
	RFD7.2.8
	RFD7.2.9
	RFO7.3
	RFO7.3.1
	RFD7.4
	RFD7.4.1
	RFD7.5
	RFD7.6
	RFO7.7
	RFO8
	RFD8.1
	RFD8.2
	RFD8.2.1
	RFD8.3
	RFD8.5
	RFO8.6
	RFO8.6.1
	RFD8.6.2
	RFD8.6.3
	RFO8.7
	RFO8.7.1
	RFO8.7.1.1
	RFO8.7.2
	RFD8.8
	RFD9.4.5
	RFD10.4
	RFD10.5
	RFD10.7
	RFD10.8
	RFD10.9
	RFD10.10
	RFD10.11
	RFO12
	RFD12.3
	RFD12.3.1
	RFD12.3.2
	RFD12.3.3
	RFD12.3.4
	RFD12.3.5
	RFF13
	RFF14
	RFF15
	RFF16



Componente	Requisiti
	RFD18 RFF19 RFO21 RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.8 RFO24 RFO25 RFO26 RFD28 RFD31 RFD32 RFD33 RFD34 RFD35 RFF40
Quizzipedia::Front-End	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFO3.1 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8



Componente	Requisiti
	RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD5 RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3



Componente	Requisiti
	RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4



Componente	Requisiti
	RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.2.2 RFD8.3 RFD8.3.1 RFD8.4 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.6.4 RFD8.6.4.1 RFO8.6.4.2 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.1.1.1 RFO8.7.1.1.2 RFO8.7.2 RFO8.7.2.1 RFD8.8 RFD8.8.1 RFD8.8.2 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFD9.4.6 RFD9.5 RFD10 RFD10.1 RFD10.2 RFD10.3 RFD10.4 RFD10.4.1



Componente	Requisiti
	RFD10.5 RFD10.5.1 RFD10.6 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.4.1 RFD11.5 RFD11.5.1 RFD11.6 RFD11.7 RFD11.7.1 RFO12 RFD12.1 RFD12.2 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD12.3.6 RFD17 RFD18 RFD18.1 RFF19 RFF19.1 RFF19.2 RFF19.3 RFO20 RFO21 RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO23.8 RFO24 RFO25



Componente	Requisiti
	RFO26 RFD28 RFD29 RFD31 RFD32 RFD33 RFD34 RFD35 RFF38 RFO39 RFF40 RFD41
Quizzipedia::Front-End::Controllers	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFO3.1 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD5 RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO7



Componente	Requisiti
	RFD7.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2



Componente	Requisiti
	RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.2.2 RFD8.3 RFD8.3.1 RFD8.4 RFD8.5 RFO8.6 RFO8.6.1 RFD8.6.2 RFD8.6.3 RFO8.6.4 RFD8.6.4.1



Componente	Requisiti
	RFO8.6.4.2 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.1.1.1 RFO8.7.1.1.2 RFO8.7.2 RFO8.7.2.1 RFD8.8 RFD8.8.1 RFD8.8.2 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFD9.4.6 RFD9.5 RFD10 RFD10.1 RFD10.2 RFD10.3 RFD10.4 RFD10.5 RFD10.5.1 RFD10.6 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5 RFD11.6 RFD11.7 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD18 RFD18.1



Componente	Requisiti
	RFF19 RFF19.2 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO24 RFO25 RFO26 RFD31 RFD32 RFD33 RFD34 RFD35 RFF38
Quizzipedia::Front-End::Directives	RFO1 RFO2 RFO3 RFO3.1 RFD5 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.3 RFD7.1.4 RFD7.1.5 RFD7.1.6 RFD7.1.7 RFD7.1.8 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.2 RFD7.2.3 RFD7.2.4 RFD7.2.5 RFD7.2.6 RFD7.2.7 RFD7.2.8 RFO7.3 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO8 RFD8.2 RFD8.2.1



Componente	Requisiti
	RFD8.2.2 RFD8.3 RFD8.3.1 RFD8.4 RFD8.5 RFD8.6.2 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD10 RFD10.1 RFD10.2 RFD10.3 RFD10.4 RFD10.4.1 RFD10.5 RFD10.5.1 RFD10.6 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.4.1 RFD11.5 RFD11.5.1 RFD11.6 RFD11.7 RFD11.7.1 RFO12 RFD12.2 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD18 RFD18.1 RFD31 RFD32 RFD33 RFD34



Componente	Requisiti
	RFF38 RFD41
Quizzipedia::Front-End::Model	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFO3.1 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD5 RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1



Componente	Requisiti
	RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3



Componente	Requisiti
	RFD7.2.3.1
	RFD7.2.3.2
	RFD7.2.4
	RFD7.2.4.1
	RFD7.2.4.2
	RFD7.2.4.2.1
	RFD7.2.4.2.2
	RFD7.2.4.2.3
	RFD7.2.4.2.4
	RFD7.2.4.3
	RFD7.2.4.3.1
	RFD7.2.4.4
	RFD7.2.4.4.1
	RFD7.2.4.4.2
	RFD7.2.4.4.3
	RFD7.2.4.4.4
	RFD7.2.5
	RFD7.2.5.1
	RFD7.2.5.2
	RFD7.2.5.3
	RFD7.2.5.4
	RFD7.2.6
	RFD7.2.6.1
	RFD7.2.6.2
	RFD7.2.6.3
	RFD7.2.7
	RFD7.2.7.1
	RFD7.2.7.2
	RFD7.2.7.3
	RFD7.2.7.4
	RFD7.2.8
	RFD7.2.9
	RFO7.3
	RFO7.3.1
	RFD7.4
	RFD7.4.1
	RFD7.5
	RFD7.6
	RFO7.7
	RFO8
	RFD8.1
	RFD8.2
	RFD8.2.1
	RFD8.2.2
	RFD8.3
	RFD8.3.1
	RFD8.4
	RFD8.5
	RFO8.6
	RFD8.6.2



Componente	Requisiti
Quizzipedia::Front-End::ModelViews	RFD8.8 RFD8.8.1 RFD8.8.2 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFD9.4.6 RFD9.5 RFD10 RFD10.2 RFD10.4 RFD10.5 RFD10.5.1 RFD10.6 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5 RFD11.6 RFD11.7 RFO12 RFD12.3 RFD12.3.6 RFD17 RFD18 RFD18.1 RFF19 RFF19.1 RFF19.3 RFO23 RFO25 RFO26 RFD31 RFD32 RFD33 RFD34 RFD35 RFF38 RFO1



Componente	Requisiti
	RFO1.1
	RFO1.2
	RFO1.3
	RFO1.4
	RFO1.5
	RFO1.6
	RFO1.7
	RFO1.8
	RFO2
	RFO2.1
	RFO2.2
	RFO2.3
	RFO2.4
	RFO3
	RFO3.1
	RFD4
	RFD4.1
	RFD4.2
	RFD4.3
	RFD4.4
	RFD4.5
	RFD4.5.1
	RFD4.5.2
	RFD4.5.3
	RFD4.6
	RFD4.7
	RFD4.8
	RFD4.8.1
	RFD4.8.2
	RFD4.9
	RFD4.9.1
	RFD5
	RFO6
	RFD6.1
	RFD6.2
	RFD6.3
	RFD6.4
	RFO6.5
	RFD7.1.2.1
	RFD7.1.2.2
	RFD7.1.2.3
	RFD7.1.3.1
	RFD7.1.3.2
	RFD7.1.3.3
	RFD7.1.3.3.1
	RFD7.1.3.3.2
	RFD7.1.3.4
	RFD7.1.4.1
	RFD7.1.4.2
	RFD7.1.5.1



Componente	Requisiti
	RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.7.1



Componente	Requisiti
	RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFD7.4 RFD7.4.1 RFD7.5 RFD7.6 RFO7.7 RFO8 RFD8.1 RFD8.2 RFD8.2.1 RFD8.2.2 RFD8.3 RFD8.3.1 RFD8.4 RFD8.5 RFD8.6.2 RFD8.8 RFD8.8.1 RFD8.8.2 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.1 RFD9.4.5 RFD9.4.6 RFD9.5 RFD10 RFD10.2 RFD10.4 RFD10.5 RFD10.5.1 RFD10.6 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5 RFD11.6



Componente	Requisiti
	RFD11.7 RFO12 RFD18 RFD18.1 RFF19 RFF19.1 RFO23 RFO25 RFO26 RFD31 RFD32 RFD33 RFD34 RFD35 RFF38
Quizzipedia::Front-End::Services	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFO3 RFO3.1 RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4 RFD4.5 RFD4.5.1 RFD4.5.2 RFD4.5.3 RFD4.6 RFD4.7 RFD4.8 RFD4.8.1 RFD4.8.2 RFD4.9 RFD4.9.1 RFD5 RFO6 RFD6.1



Componente	Requisiti
	RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO7 RFD7.1 RFD7.1.1 RFD7.1.2 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3 RFD7.1.3.1 RFD7.1.3.2 RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.2 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10



Componente	Requisiti
	RFD7.2 RFD7.2.1 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.3 RFD7.2.4.3.1 RFD7.2.4.4 RFD7.2.4.4.1 RFD7.2.4.4.2 RFD7.2.4.4.3 RFD7.2.4.4.4 RFD7.2.5 RFD7.2.5.1 RFD7.2.5.2 RFD7.2.5.3 RFD7.2.5.4 RFD7.2.6 RFD7.2.6.1 RFD7.2.6.2 RFD7.2.6.3 RFD7.2.7 RFD7.2.7.1 RFD7.2.7.2 RFD7.2.7.3 RFD7.2.7.4 RFD7.2.8 RFD7.2.9 RFO7.3 RFO7.3.1 RFD7.4 RFD7.4.1 RFD7.5



Componente	Requisiti
	RFD7.6 RFO7.7 RFO8 RFD8.1 RFO8.6 RFD8.6.2 RFO8.6.4 RFD8.6.4.1 RFO8.6.4.2 RFO8.7 RFO8.7.1 RFO8.7.1.1 RFO8.7.2 RFO8.7.2.1 RFD8.8 RFD8.8.2 RFO9 RFD9.1 RFD9.2 RFD9.3 RFD9.4 RFD9.4.5 RFD9.4.6 RFD9.5 RFD10 RFD10.4 RFD10.4.1 RFD10.5 RFD10.5.1 RFD10.7 RFD10.8 RFD10.9 RFD10.10 RFD10.11 RFO11 RFO11.1 RFO11.2 RFO11.3 RFD11.4 RFD11.5 RFD11.6 RFD11.7 RFO12 RFD12.3 RFD12.3.1 RFD12.3.2 RFD12.3.3 RFD12.3.4 RFD12.3.5 RFD18



Componente	Requisiti
	RFD18.1 RFF19 RFF19.2 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO23.8 RFO24 RFO25 RFO26 RFD28 RFD29 RFD31 RFD32 RFD33 RFD34 RFD35 RFF38 RFF40
Quizzipedia::Front-End::Views	RFO1 RFO1.1 RFO1.2 RFO1.3 RFO1.4 RFO1.5 RFO1.6 RFO1.7 RFO1.8 RFO2 RFO2.1 RFO2.2 RFO2.3 RFO2.4 RFD5 RFO6 RFD6.1 RFD6.2 RFD6.3 RFD6.4 RFO6.5 RFO7 RFD7.1.2.1 RFD7.1.2.2 RFD7.1.2.3 RFD7.1.3.1 RFD7.1.3.2



Componente	Requisiti
	RFD7.1.3.3 RFD7.1.3.3.1 RFD7.1.3.3.2 RFD7.1.3.4 RFD7.1.4.1 RFD7.1.4.2 RFD7.1.5.1 RFD7.1.5.2 RFD7.1.5.2.1 RFD7.1.5.2.2 RFD7.1.5.2.3 RFD7.1.5.2.4 RFD7.1.5.3 RFD7.1.5.3.1 RFD7.1.5.4 RFD7.1.5.4.1 RFD7.1.5.4.3 RFD7.1.5.4.4 RFD7.1.6.1 RFD7.1.6.2 RFD7.1.6.3 RFD7.1.7.1 RFD7.1.7.2 RFD7.1.7.3 RFD7.1.8 RFD7.1.8.1 RFD7.1.8.2 RFD7.1.8.3 RFD7.1.8.4 RFD7.1.9 RFD7.1.10 RFD7.2.1.1 RFD7.2.1.2 RFD7.2.1.3 RFD7.2.2.1 RFD7.2.2.2 RFD7.2.2.3 RFD7.2.2.3.1 RFD7.2.2.3.2 RFD7.2.2.4 RFD7.2.3.1 RFD7.2.3.2 RFD7.2.4.1 RFD7.2.4.2 RFD7.2.4.2.1 RFD7.2.4.2.2 RFD7.2.4.2.3 RFD7.2.4.2.4 RFD7.2.4.4 RFD7.2.4.4.1



Componente	Requisiti
	RFD7.2.4.4.2
	RFD7.2.4.4.3
	RFD7.2.4.4.4
	RFD7.2.5.1
	RFD7.2.5.2
	RFD7.2.5.3
	RFD7.2.5.4
	RFD7.2.6.1
	RFD7.2.6.2
	RFD7.2.6.3
	RFD7.2.7.1
	RFD7.2.7.2
	RFD7.2.7.3
	RFD7.2.7.4
	RFD7.2.8
	RFD7.2.9
	RFO7.3.1
	RFD7.4
	RFD7.4.1
	RFD7.5
	RFD7.6
	RFO8
	RFD8.1
	RFD8.4
	RFO8.6
	RFO8.6.1
	RFD8.6.3
	RFO8.6.4
	RFD8.6.4.1
	RFO8.6.4.2
	RFO8.7
	RFD8.8
	RFD8.8.1
	RFD8.8.2
	RFO9
	RFD9.4.5
	RFD9.4.6
	RFD10
	RFD10.1
	RFD10.2
	RFO12
	RFD12.1
	RFD12.3
	RFD12.3.6
	RFD17
	RFF19
	RFF19.1
	RFF19.2
	RFO20
	RFO21



Componente	Requisiti
	RFO22 RFO23 RFO23.1 RFO23.2 RFO23.3 RFO23.4 RFD23.5 RFD23.6 RFO23.7 RFO24 RFO25 RFO26 RFD31 RFD32 RFD33 RFD34 RFD35

Tabella 3: Tracciamento Componenti-Requisiti



8 Tracciamento Requisiti-Componenti

Requisito	Componenti
RFO1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End



Requisito	Componenti
	Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers



Requisito	Componenti
	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO1.8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::-Errors Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::-Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO2.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::-Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO2.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::-Users Quizzipedia::Back-End::App::Model



Requisito	Componenti
	Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO2.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO2.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Errors Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO3.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model



Requisito	Componenti
	Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.5.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.5.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.5.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Errors Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.8	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Model Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.8.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.8.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.9	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD4.9.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model



Requisito	Componenti
	Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD6.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD6.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD6.3	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD6.4	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO6.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.2.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.2.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.2.3	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.3.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.3.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model



Requisito	Componenti
	Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.3.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.3.3.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.3.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.4.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers



Requisito	Componenti
	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.4.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.5.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.2.1	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.2.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.2.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.2.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers



Requisito	Componenti
	Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.4.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.4.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.5.4.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model



Requisito	Componenti
	Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.5.4.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.6.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.6.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views



Requisito	Componenti
RFD7.1.6.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.1.7.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.7.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.7.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers



Requisito	Componenti
	Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.8.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.8.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.8.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.8.4	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.9	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.1.10	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services



Requisito	Componenti
RFD7.2.1.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.1.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.1.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.2.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End



Requisito	Componenti
	Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.2.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.2.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.2.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.2.3.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.2.4	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.3.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.4.1	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.2.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.2.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.2.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers



Requisito	Componenti
	Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.2.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.4.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.4.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.4.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End



Requisito	Componenti
	Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.4.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.4.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.4.4.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.5.1	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.5.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.5.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.5.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives



Requisito	Componenti
	Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.6.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.6.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.6.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFD7.2.7.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.7.2	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.7.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.7.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.2.9	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End



Requisito	Componenti
	Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO7.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services
RFO7.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.4.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.5	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD7.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO7.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD8.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers



Requisito	Componenti
	Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD8.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD8.2.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD8.2.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD8.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD8.3.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD8.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Front-End



Requisito	Componenti
	Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Views
RFD8.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFO8.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO8.6.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Views
RFD8.6.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD8.6.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End



Requisito	Componenti
	Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Views
RFO8.6.4	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD8.6.4.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO8.6.4.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO8.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO8.7.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services
RFO8.7.1.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services
RFO8.7.1.1.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers
RFO8.7.1.1.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers
RFO8.7.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers



Requisito	Componenti
RFO8.7.2.1	Quizzipedia::Front-End::Services Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services
RFD8.8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD8.8.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Views
RFD8.8.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO9	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD9.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD9.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD9.3	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews



Requisito	Componenti
RFD9.4	Quizzipedia::Front-End::Services Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD9.4.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD9.4.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD9.4.6	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD9.5	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD10.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Views
RFD10.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews



Requisito	Componenti
RFD10.3	Quizzipedia::Front-End::Views Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives
RFD10.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.4.1	Quizzipedia::Front-End Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Services
RFD10.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.5.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.6	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews
RFD10.7	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model



Requisito	Componenti
	Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.9	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.10	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD10.11	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO11	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO11.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO11.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO11.3	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD11.4	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD11.4.1	Quizzipedia::Front-End Quizzipedia::Front-End::Directives
RFD11.5	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD11.5.1	Quizzipedia::Front-End Quizzipedia::Front-End::Directives



Requisito	Componenti
RFD11.6	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD11.7	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD11.7.1	Quizzipedia::Front-End Quizzipedia::Front-End::Directives
RFO12	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD12.1	Quizzipedia::Front-End Quizzipedia::Front-End::Views
RFD12.2	Quizzipedia::Front-End Quizzipedia::Front-End::Directives
RFD12.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD12.3.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model



Requisito	Componenti
	Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Services
RFD12.3.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Services
RFD12.3.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Services
RFD12.3.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Services
RFD12.3.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Services
RFD12.3.6	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Errors Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::Views
RFF13	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers
RFF14	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers
RFF15	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers
RFF16	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers
RFD17	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller:::- Errors Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::Views
RFD18	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers



Requisito	Componenti
	Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFD18.1	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFF19	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFF19.1	Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Views
RFF19.2	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFF19.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Errors Quizzipedia::Front-End Quizzipedia::Front-End::Model
RFO20	Quizzipedia::Back-End Quizzipedia::Front-End Quizzipedia::Front-End::Views
RFO21	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Views
RFO22	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Views
RFO23	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Views
RFO23.1	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO23.2	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO23.3	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO23.4	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views



Requisito	Componenti
RFD23.5	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD23.6	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO23.7	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO23.8	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Services
RFO24	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO25	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFO26	Quizzipedia::Back-End



Requisito	Componenti
	Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD28	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Services
RFD29	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Front-End Quizzipedia::Front-End::Services
RFD31	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD32	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD33	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller



Requisito	Componenti
	Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD34	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFD35	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services Quizzipedia::Front-End::Views
RFF38	Quizzipedia::Front-End Quizzipedia::Front-End::Controllers Quizzipedia::Front-End::Directives Quizzipedia::Front-End::Model Quizzipedia::Front-End::ModelViews Quizzipedia::Front-End::Services
RFO39	Quizzipedia::Front-End
RFF40	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Model Quizzipedia::Back-End::App::Routers Quizzipedia::Front-End Quizzipedia::Front-End::Services
RFD41	Quizzipedia::Front-End Quizzipedia::Front-End::Directives
RFO42	Quizzipedia::Back-End Quizzipedia::Back-End::App



Requisito	Componenti
	Quizzipedia::Back-End::App::Controller Quizzipedia::Back-End::App::Controller::- Users
RFO43	Quizzipedia::Back-End Quizzipedia::Back-End::App Quizzipedia::Back-End::App::Controller

Tabella 4: Tracciamento Requisiti-Componenti



A Design patterns

A.1 Strutturali

A.1.1 Facade

- **Scopo:** permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi. Questo rende una libreria più facile da capire, usare e testare, inoltre permette di diminuire le dipendenze tra sottosistemi senza nascondere le funzionalità di basso livello;
- **Motivazione:** l'utilizzo del pattern *Facade_G* permette di nascondere la complessità dell'operazione. Quando un sistema complesso viene strutturato in sottosistemi, le dipendenze rischiano di aumentare in modo consistente. Applicare il pattern *Facade_G* aiuta a diminuire queste dipendenze. Il sottosistema possederà un'interfaccia semplificata che il *client_G* utilizza anziché dover gestire numerosi oggetti. Utilizzare il pattern *Facade_G* promuove un accoppiamento debole tra sottosistema ed i *client_G*, che comporta una maggiore flessibilità nello sviluppo: è possibile modificare il sottosistema senza che i *client_G* debbano adeguarsi a loro volta. Ciononostante i *client_G* possono comunque accedere alle funzionalità di basso livello ed utilizzare le classi del sottosistema;
- **Applicabilità:** il pattern *Facade_G* si usa nei seguenti casi:
 - Si vuole fornire una singola interfaccia semplice per un sottosistema complesso;
 - Si vuole promuovere il disaccoppiamento tra sottosistemi e *client_G*, semplificando le dipendenze;
 - Si vuole stratificare un sistema: è possibile definire una classe *Facade_G* come punto d'ingresso per ogni livello di sottosistema. In questo modo, se vi sono dipendenze fra sottosistemi, essi possono comunicare fra loro attraverso la propria *Facade_G*.

• Utilizzo:

- Nella parte Font-End dell'applicazione il *design pattern_G* *Facade_G* viene espresso dalla classe **AppRouter** che gestisce i routes dell'applicazione e viene utilizzata per associare un *URL_G* alle varie view dell'applicazione. Essa utilizza il servizio **\$routeProvider** per associare ad ogni route un controller e una view e **\$locationProvider** per configurare come i paths dell'applicazione vengono salvati. Questa funzione viene utilizzata come parametro nel metodo **config** di *Angular_G*. Il metoto **config** permette di impostare l'esecuzione di una funzione al caricamento del modulo principale di *Angular_G*;
- Nella parte Back-End dell'applicazione il *design pattern_G* *Facade_G* viene utilizzato per semplificare l'interfaccia con cui la risorsa *REST_G* interagisce con i metodi delle classi *controllers_G*. Senza l'utilizzo del *design pattern_G* la gestione di ogni risorsa *REST_G* verrebbe soddisfatta in modo più complesso, come si può notare nel seguente diagramma che rappresenta una sezione back-end senza l'utilizzo si *Facade_G*:

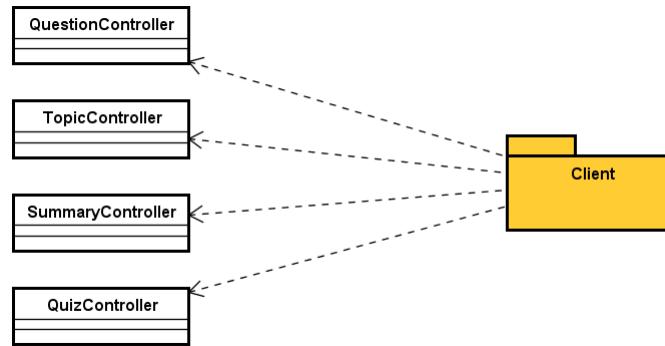


Figura 261: Parte di QuizziPedia::Back-End senza utilizzo di Facade

L'introduzione delle classi $Facade_G$ permette una miglior gestione delle risorse $REST_G$, permettendo di diminuire le dipendenze tra classi e risorse.

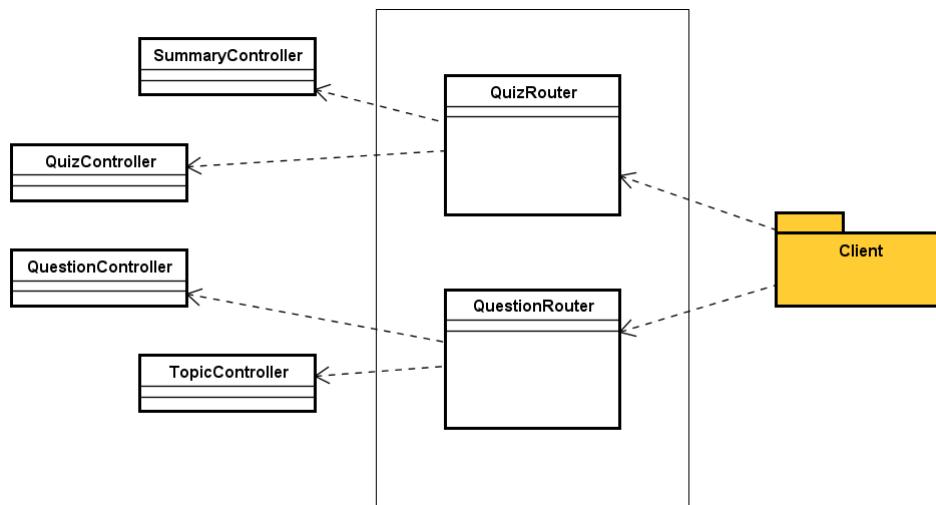


Figura 262: Parte di QuizziPedia::Back-End con l'utilizzo di Facade

A.2 Creazionali

A.2.1 Dependency Injection

- **Scopo:** semplificare lo sviluppo e rendere più testabile un software di grandi dimensioni. Il pattern $Dependency\ Injection_G$ separa il codice della componente dal codice che si occupa di risolvere le dipendenze con altre componenti;
- **Motivazione:** lasciare al componente il compito di risolvere le proprie dipendenze, creando gli oggetti necessari al suo funzionamento, aumenta l'accoppiamento tra le componenti e rende più difficoltoso progettare i test di unità; con questo pattern invece è possibile esprimere le dipendenze in modo dichiarativo e utilizzare un oggetto contenitore per risolverle dinamicamente a $runtime_G$. In questo modo è possibile scegliere anche quale componente iniettare in base allo stato del programma;
- **Applicabilità:** questo pattern viene utilizzato dalla maggior parte dei $framework_G$ moderni. In particolare, $AngularJS_G$ offre il servizio `injector` che permette di invocare delle funzioni iniettando al loro interno degli oggetti;



- **Struttura:** i componenti coinvolti nel *Dependency Injection* sono:

- Un *client_G* che viene creato e riceve le dipendenze;
- Un contenitore che si occupa di creare il *client_G* e di iniettarvi le dipendenze;
- Un servizio che deve essere iniettato al *client_G*.

Nello specifico di *Angular_G*, `$injector` funziona da contenitore che si occupa di risolvere le dipendenze. I *client_G* sono rappresentati dalle funzioni che costruiscono i componenti dell'applicazione, tipicamente *controller_G* o *service_G*. Il servizio è un oggetto *service_G* che può essere definito dall'utente oppure uno di quelli resi disponibili da *Angular_G*;

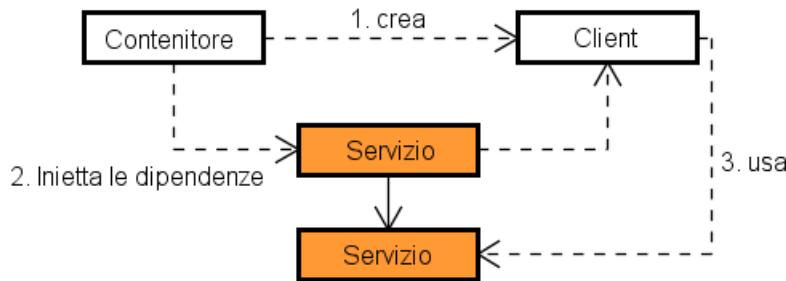


Figura 263: Struttura logica del pattern Dependency Injection

L'injection delle dipendenze può essere fatta in due modi:

- **Constructor Injection:** le dipendenze vengono dichiarate come parametri del costruttore che il container andrà a valorizzare quando crea l'oggetto. In questo modo l'oggetto costruito è subito utilizzabile e non è mai in uno stato inconsistente. Tuttavia si rende necessario avere dei costruttori con un elevato numero di parametri, che potrebbero rendere i costruttori difficili da comprendere. *Angular_G* usa questo tipo di dependency injection;
- **Setter Injection:** le dipendenze vengono dichiarate come metodi setter del componente. Questa soluzione è meno ambigua, funziona bene con le gerarchie di classi e non soffre del telescoping dei costruttori. Non è però possibile avere oggetti immutabili. È possibile che la componente rimanga in uno stato inconsistente, in quanto deve essere costruita per passi.
- **Utilizzo:** il design pattern *Dependency Injection_G* è parte integrante di *Angular_G* e viene utilizzato nel Front-End ogni volta che una funzione ha bisogno di richiamare una funzionalità esterna. Ad esempio ad ogni controller che necessita di una funzionalità appartenente ad un *service_G* viene passato un oggetto riferimento a tale *service_G* (si veda ClickableAreaQuestionsController, ConnectionQuestionsController, CreateQuestionnaireController...). In questo modo, attraverso tale oggetto, il *controller_G* può richiamare direttamente le funzioni del *service_G* desiderate. Sempre nei *controller_G*, viene dichiarata la dipendenza all'oggetto di sistema `$scope`, passandolo come parametro al costruttore del controller, così da poter accedere direttamente a tutte le informazioni contenute in esso;
- **Svantaggi:** eventuali errori legati alla risoluzione delle dipendenze o alla loro implementazione vengono rilevati solamente a *runtime_G*.



A.3 Comportamentali

A.3.1 Chain-of-responsability

- **Scopo:** il pattern *Chain-of-responsability_G* ha lo scopo di evitare l'accoppiamento tra il mittente di una richiesta e il destinatario, in modo che più di un singolo oggetto possa eseguire la richiesta. Concatenare gli oggetti destinatari e passare la richiesta di oggetto in oggetto, finché uno di questi non riesce ad esaudirla;
- **Motivazione:** l'oggetto che ha iniziato la richiesta non è a conoscenza di chi la esaudirà, per questo si parla di destinatario implicito. Ogni oggetto della catena condivide un'interfaccia comune handler per la gestione delle richieste e per accedere al successivo elemento della catena. Questo per consentire il passaggio lungo la catena e per assicurare che l'oggetto, che eventualmente esaudirà la richiesta, rimanga implicito. Il metodo che viene mantenuto in tutte le interfacce per creare la catena si chiama handle. handle, nella versione standard, esegue la chiamata al successore della catena. Alla fine della catena viene fatto l'*overriding_G* di questo metodo, implementando la richiesta iniziale oppure generando l'errore;
- **Applicabilità:** *Chain-of-responsability_G* si usa nei seguenti casi:
 - Più di un oggetto può gestire la richiesta ed il ricevente che la gestirà non è conosciuto a priori;
 - Si vuole passare una richiesta ad uno dei molti oggetti, senza esplicitare il ricevente;
 - L'insieme di oggetti che gestirà una richiesta deve essere definito dinamicamente.
- **Utilizzo:** *Express_G* usa *Chain-of-responsability_G* per la gestione dei *middleware_G* e del routing. Nell'architettura viene utilizzato all'interno del package `Back-End::App`. Ogni `ConcreteHandler` eredita da `MiddlewareHandler`, che corrisponde all'interfaccia astratta `Handler` del *design pattern_G*. Con *Express_G* i *middleware_G* corrispondono ai `ConcreteHandler`. L'implementazione di questi risulta sotto alcuni aspetti differente da una normale implementazione del pattern:
 1. I *middleware_G* di *Express_G* possono essere delle classi che implementano il metodo `handle` oppure delle funzioni secondo lo stile funzionale delle librerie e dei moduli di *Node.js_G*. Nella nostra architettura abbiamo utilizzato principalmente la seconda versione;
 2. Nel *design pattern_G* è previsto che l'oggetto `ConcreteHandler` abbia un riferimento (successor) al `ConcreteHandler` successivo. *Express_G*, anziché un riferimento, passa al metodo che esegue il *middleware_G* una `callbackG`. Il *middleware_G* che esegue la `callbackG` passa il controllo all'oggetto del *server_G* di *Express_G*, il quale passerà a sua volta il controllo al successivo *middleware_G*.

Express_G divide i *middleware_G* in due tipologie:

1. *Middleware standard* con 3 parametri formali;
2. *Middleware per la gestione degli errori* con 4 parametri formali, ovvero i 3 del middleware standard più un parametro per gli errori.

Ogni *middleware_G* può passare il controllo al *middleware_G* standard successivo, oppure ad un *middleware_G* per la gestione degli errori, passandogli l'errore relativo. Ogni *middleware_G* di *Express_G* deve essere invocato con i parametri elencati nel seguente ordine:

1. L'eventuale errore da gestire, in caso si tratti di un middleware per la gestione degli errori;



2. L'oggetto contenente la richiesta da risolvere;
3. L'oggetto che conterrà la risposta;
4. La $callback_G$ da utilizzare per passare il controllo al successivo middleware.

- **Struttura:**

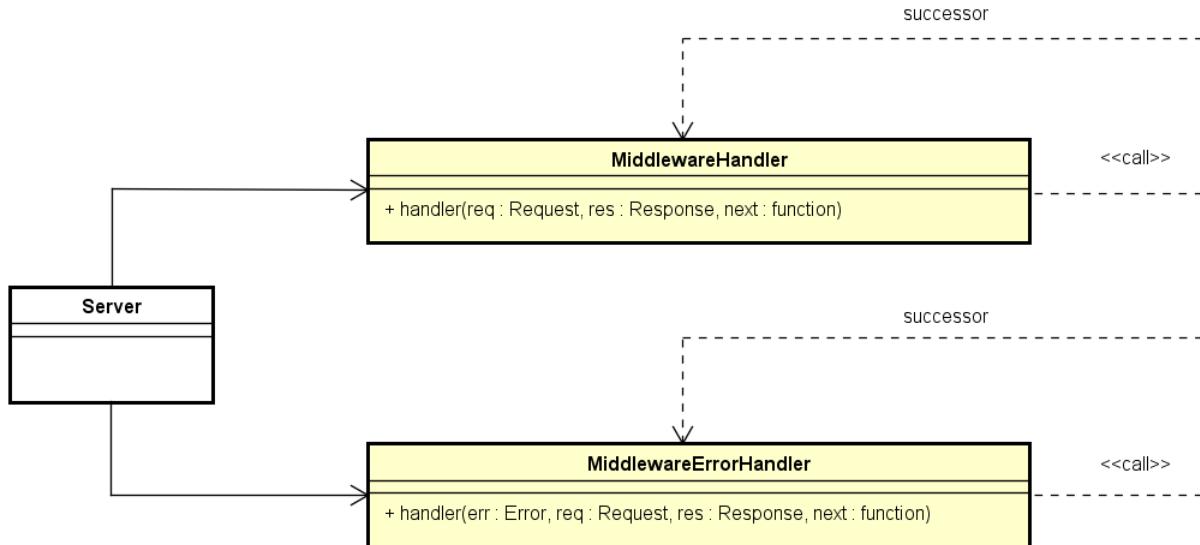


Figura 264: Struttura logica del pattern Chain-of-responsability

- **Collaborazioni:** quando un $client_G$ effettua una richiesta, questa viene propagata lungo la catena finché un ConcreteHandler non si assume la responsabilità di gestirla;
- **Svantaggi:** è necessario porre attenzione a come si configura la catena. Infatti, in caso di configurazione inappropriata, sussiste il rischio che la richiesta passi per diversi punti della catena senza essere mai realizzata, oppure che la richiesta non venga realizzata perché non esiste un ConcreteHandler che la possa risolvere.

A.3.2 Observer

- **Scopo:** definisce una dipendenza “1..n” fra oggetti, riflettendo la modifica di un oggetto sui dipendenti;
- **Motivazione:** in alcuni casi è necessario tenere sincronizzati vari oggetti e, nel caso un oggetto venga modificato, avere la possibilità di riflettere il cambiamento sugli oggetti dipendenti.

Il pattern $Observer_G$ permette di implementare ciò definendo due tipologie di oggetti: i subject, che rappresentano gli oggetti osservati, e gli observer che rappresentano gli oggetti che osservano lo stato del subject. Con questo pattern è possibile quindi realizzare un sistema publish-subscribe, nel quale i vari subject si registrano per essere notificati dalle modifiche subite dal subject;

- **Applicabilità:** il pattern $Observer_G$ viene usato nei seguenti casi:
 - Si vuole che il cambiamento dello stato di un oggetto provochi l'aggiornamento di altri oggetti;
 - Si vuole notificare degli oggetti dell'avvenimento di un determinato evento senza sapere il loro tipo.



- **Utilizzo:** Nella parte Back-End dell'applicazione si necessita di modificare le statistiche di una domanda e di un utente per ogni domanda risposta. Il *design pattern_G Observer_G* è stato utilizzato per gestire l'aggiornamento dei livelli di difficoltà e di abilità, come illustrato nel seguente diagramma:

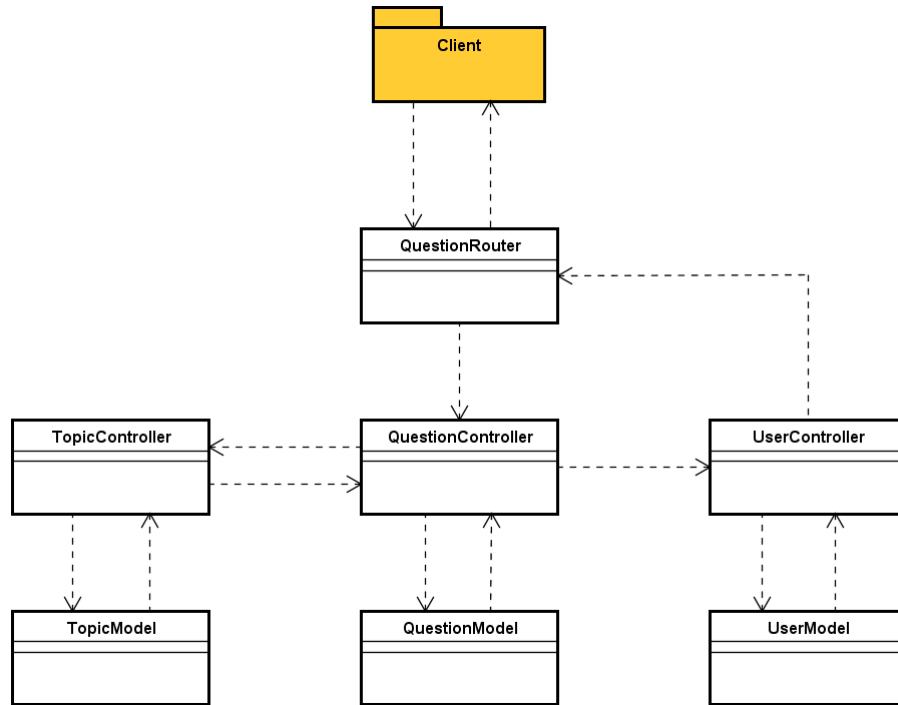


Figura 265: Utilizzo di Observer per l'aggiornamento delle statistiche



B QML - Quiz Markup Language

Uno dei punti fondamentali dell'applicazione è permettere agli utenti di poter creare domande da proporre negli allenamenti e nei questionari. Per rendere più semplice l'aggiunta di una domanda sono stati realizzati degli *wizard_G* per alcune tipologie di domande, ma questo limita l'utente alla sola creazione di tali tipologie. La soluzione adottata è la realizzazione di un nuovo linguaggio di *markup_G* che permette la definizione di domande non ordinarie, denominato QML acronimo di Quiz Markup Language. Per la realizzazione di QML abbiamo adottato il *Jison_G*, uno strumento che ci permette di definire un *lexer_G* ed un *parser_G* in grado di costruire e scorrere un albero di analisi al fine di generare un documento JSON a partire da una grammatica definita.

B.1 Definizione della grammatica

Per mantenere la semplicità di realizzazione di una domanda tramite QML è stato deciso di mantenere la sintassi *JSON*, con l'aggiunta delle sole parole chiave necessarie per mantenere coerente la grammatica con lo scopo del QML. Alla grammatica è stata data possibilità di inserire commenti che saranno rimossi prima della creazione del *JSON*.

B.1.1 Sintassi domanda Vero/Falso

```

1 {
2   "type": "veroFalso",
3   "image": "/img/veroFalso/example.png", //immagine possibile nel testo
4   "answer": [
5     [
6       "text": "In Italia la guida e' a destra",
7       "isItRight": true
8     ],
9   "keywords": [
10    [
11      "keyword_1": "guida",
12      "keyword_2": "legge",
13      "keyword_3": "italia"
14    ]
15 ]

```

B.1.2 Sintassi domanda a Risposta Multipla

```

1 {
2   "type": "rispostaMultipla",
3   "questionText": "Quali di questi numeri e' pari?",
4   "url": "/img/rispostaMultipla/D0_3.png", //immagine possibile nel testo
5   "answer": [
6     [
7       "text": "1",
8       "url": "/img/rispostaMultipla/D0_1.png", //url dell'immagine,
9       "possibile campo facoltativo
10      "isItRight": false
11    ],
12    [
13      "text": "2",
14      "url": "/img/rispostaMultipla/D0_2.png", //url dell'immagine,
15      "possibile campo facoltativo
16      "isItRight": true
17 ]

```



```
14     }, {
15     "text": "7",
16     "url": "/img/rispostaMultipla/D0_3.png", //url dell'immagine,
17     //possibile campo facoltativo
18     "isItRight": false
19   }, {
20     "text": "9",
21     "url": "/img/rispostaMultipla/D0_4.png", //url dell'immagine,
22     //possibile campo facoltativo
23     "isItRight": false
24   }],
25 "keywords": [
26   {
27     "keyword_1": "numeri",
28     "keyword_2": "matematica",
29     "keyword_3": "scuola"
30   }
31 }
```

B.1.3 Sintassi domanda a Ordinamento di Stringhe

```
1 {
2   "type": "ordinamentoStringhe",
3   "questionText": "Ordina questi numeri in modo decrescente.",
4   "answer": [
5     {
6       "text": "1",
7       "position": 4
8     }, {
9       "text": "2",
10      "position": 3
11    }, {
12      "text": "7",
13      "position": 2
14    }, {
15      "text": "9",
16      "position": 1
17    }],
18 "keywords": [
19   {
20     "keyword_1": "ordinamento",
21     "keyword_2": "numeri",
22   }
23 ]}
```

B.1.4 Sintassi domanda a Ordinamento Immagini

```
1 {
2   "type": "ordinamentoImmagini",
3   "questionText": "Ordina queste immagini in modo da ottenere la parola \"CIAO\".",
4   "url": "/img/ordinamentoImmagini/D0_1.png", //immagine possibile nel
5   //testo della domanda ad ordinamento di immagini
6   "answer": [
7     {
8       "text": "CIAO"
9     }
10   ]}
```



```
7     "url": "/img/domandeOrdinamentoImmagini/I.png", //url dell'immagine
8     "position": 2
9   },
10  },
11  "url": "/img/domandeOrdinamentoImmagini/A.png",
12  "position": 3
13  },
14  "url": "/img/domandeOrdinamentoImmagini/0.png",
15  "position": 4
16  },
17  "url": "/img/domandeOrdinamentoImmagini/C.png",
18  "position": 1
19 ],
20 "keywords":
21   [
22     {
23       "keyword_1": "parole",
24       "keyword_2": "grammatica"
25     }
26 }
```

B.1.5 Sintassi domanda a Collegamento di Elementi

```
1 {
2   "type": "collegamento",
3   "questionText": "Collega queste coppie di nemici classici.",
4   "answer":
5     [
6       {
7         "text_1_A": "cane",
8         "text_1_B": "gatto"
9       },
10      {
11        "url_2_A": "/img/collegamento/uncino.png",
12        "text_2_B": "peter pan"
13      },
14      {
15        "url_3_A": "/img/collegamento/D0_1.png",
16        "url_3_B": "/img/collegamento/D0_5.png"
17      },
18    ],
19   "keywords":
20   [
21     {
22       "keyword_1": "nemici"
23     }
24 }
```

B.1.6 Sintassi domanda ad Area Cliccabile

```
1 {
2   "type": "areaCliccabile",
3   "questionText": "Clicca quale tra le seguenti scelte e' il bicipide.",
4   "image": "/img/areaCliccabile/D0_1.png", //sfondo dell'area cliccabile
5   "resolution": { "x":400, "y":500 },
6   "answer":
7     [
8       {
9         "x": "200",
10        "y": "50",
11        "text": "testo facoltativo di arricchimento"
12      },
13      {
14        "x": "300",
15      }
16    ]
17 }
```



```

13     "y" : "120",
14     "text" : "testo facoltativo di arricchimento"
15   },
16   "x" : "200",
17   "y" : "200",
18   "text" : "testo facoltativo di arricchimento"
19 ],
20 "keywords" :
21 [
22   "keyword_1" : "corpo umano",
23   "keyword_2" : "medicina",
24   "keyword_3" : "scienze",
25   "keyword_4" : "muscoli"
26 ]
27 }
```

B.1.7 Sintassi domanda a Riempimento spazi vuoti

```

1 {
2   "type": "spaziVuoti",
3   "questionText": "Giulio Cesare era un console romano.",
4   "answer" :
5     [
6       {
7         "parolaNumero": 2 //sta ad indicare quale parola e' da oscurare.
8           In questo caso la numero 2
9       },
10      "parolaNumero": 5
11    ],
12   "keywords" :
13   [
14     "keyword_1" : "storia",
15   ]
16 }
```

B.2 Generazione del JSON

Se il parser_G non trova errori o ambiguità viene creato un $JSON_G$ contenente la domanda e altre informazioni necessarie per aggiungerla al sistema. Ogni tipologia di domanda crea un tipo di $JSON_G$ diverso, questo accade sia per la generazione tramite $wizard_G$ sia tramite QML.

B.2.1 JSON domanda Vero/Falso

```

1 {
2   "makeWith" : "QML" || "wizzard" ,
3   "lang" : [:lang], // impostata la lingua in uso
4   "question" :
5     [
6       {
7         "type": "veroFalso",
8         "image": "/img/veroFalso/example.png", //immagine possibile nel
9           testo della domanda vero e falso
10        "answer" :
11          [
12            {
13              "text": "In Italia la guida e' a destra",
14              "isItRight": true
15            }
16          ]
17      }
18    ]
19 }
```



```
13     }],
14 "keywords" :
15   [
16     {
17       "keyword_1" : "guida",
18       "keyword_2" : "legge",
19       "keyword_3" : "italia"
20     },
21   ],
22 "statistiche" :
23   [
24     {
25       "level" : 500,
26       "totalAnswers" : 0,
27       "correctAnswer" : 0
28     }
29 ]
30 }
```

B.2.2 JSON domanda a Risposta Multipla

```
1 {
2   "makeWith" : "QML" || "wizzard" ,
3   "type": "rispostaMultipla",
4   "questionText": "Quali di questi numeri e' pari?",,
5   "url": "/img/rispostaMultipla/D0_3.png", //immagine possibile nel testo
          della domanda risposta multipla
6   "answer":
7     [
8       {
9         "text": "1",
10        "url": "/img/rispostaMultipla/D0_1.png", //url dell'immagine,
11          possibile campo facoltativo
12        "isItRight": false
13      },
14       {
15         "text": "2",
16        "url": "/img/rispostaMultipla/D0_2.png", //url dell'immagine,
17          possibile campo facoltativo
18        "isItRight": true
19      },
20       {
21         "text": "7",
22        "url": "/img/rispostaMultipla/D0_3.png", //url dell'immagine,
23          possibile campo facoltativo
24        "isItRight": false
25      },
26       {
27         "text": "9",
28        "url": "/img/rispostaMultipla/D0_4.png", //url dell'immagine,
29          possibile campo facoltativo
30        "isItRight": false
31     },
32   "keywords" :
33     [
34       {
35         "keyword_1" : "numeri",
36         "keyword_2" : "matematica",
37         "keyword_3" : "scuola"
38     },
39   "statistiche" :
40   [
41     {
42       "level" : 500,
43       "totalAnswers" : 0,
44       "correctAnswers" : 0
45     }
46   ]
47 }
```



36

}

B.2.3 JSON domanda a Ordinamento di Stringhe

```
1 {
2   "makeWith" : "QML" || "wizzard" ,
3   "type" : "ordinamentoStringhe",
4   "questionText": "Ordina questi numeri in modo decrescente." ,
5   "answer" :
6     [
7       {
8         "text": "1",
9         "position": 4
10      },
11      {
12        "text": "2",
13        "position": 3
14      },
15      {
16        "text": "7",
17        "position": 2
18      },
19      {
20        "text": "9",
21        "position": 1
22      }
23   ],
24   "keywords" :
25   [
26     {
27       "keyword_1" : "ordinamento",
28       "keyword_2" : "numeri",
29     }
30   ]
```

B.2.4 JSON domanda a Ordinamento Immagini

```
1 {
2   "makeWith" : "QML" || "wizzard" ,
3   "type" : "ordinamentoImmagini",
4   "questionText": "Ordina queste immagini in modo da ottenere la parola \"CIAO\".",
5   "url": "/img/ordinamentoImmagini/D0_1.png", //immagine possibile nel testo della domanda ad ordianamento di immagini
6   "answer" :
7     [
8       {
9         "url": "/img/domandeOrdinamentoImmagini/I.png", //url dell'immagine
10        "position": 2
11      },
12      {
13        "url": "/img/domandeOrdinamentoImmagini/A.png",
14        "position": 3
15      },
16      {
17        "url": "/img/domandeOrdinamentoImmagini/O.png",
18        "position": 4
19      }
20   ]
```



```

17     "url" : "/img/domandeOrdinamentoImmagini/C.png",
18     "position" : 1
19   ],
20 "keywords" :
21   [
22     {
23       "keyword_1" : "parole",
24       "keyword_2" : "grammatica"
25     },
26 "statistiche" :
27   [
28     {
29       "total" : 500,
30       "totalAnswers" : 0,
31       "correctAnswers" : 0
32     }
33 ]

```

B.2.5 JSON domanda a Collegamento di Elementi

```

1 {
2 "makeWith" : "QML" || "wizzard" ,
3 "type": "collegamento",
4 "questionText": "Collega queste coppie di nemici classici.",
5 "answer" :
6   [
7     {
8       "text_1_A": "cane",
9       "text_1_B": "gatto"
10    }, {
11      "url_2_A": "/img/collegamento/uncino.png",
12      "text_2_B": "peter pan"
13    }, {
14      "url_3_A": "/img/collegamento/D0_1.png",
15      "url_3_B": "/img/collegamento/D0_5.png"
16    },
17 "keywords" :
18   [
19     {
20       "keyword_1" : "nemici"
21     },
22 "statistiche" :
23   [
24     {
25       "level" : 500,
26       "totalAnswers" : 0,
27       "correctAnswers" : 0
28     }
29 ]
30

```

B.2.6 JSON domanda ad Area Cliccabile

```

1 {
2 "makeWith" : "QML" || "wizzard" ,
3 "type": "areaCliccabile",
4 "questionText": "Clicca quale tra le seguenti scelte e' il bicipide.",
5 "image": "/img/areaCliccabile/D0_1.png", //sfondo dell'area cliccabile
6 "resolution": { "x":400, "y":500 },
7 "answer" :
8   [
9     {
10       "x": "200",
11       "y": "300"
12     }
13   ]
14 }

```



```
10     "y" : "50",
11     "text" : "testo facoltativo di arricchimento"
12 }, {
13     "x" : "300",
14     "y" : "120",
15     "text" : "testo facoltativo di arricchimento"
16 }, {
17     "x" : "200",
18     "y" : "200",
19     "text" : "testo facoltativo di arricchimento"
20 },
21 "keywords" :
22     [
23         {
24             "keyword_1" : "corpo umano",
25             "keyword_2" : "medicina",
26             "keyword_3" : "scienze",
27             "keyword_4" : "muscoli"
28         },
29     ],
30 "statistiche" :
31     [
32         {
33             "level" : 500,
34             "totalAnswers" : 0,
35             "correctAnswers" : 0
36         }
37 }
```

B.2.7 JSON domanda a Riempimento spazi vuoti

```
1 {
2     "makeWith" : "QML" || "wizzard" ,
3     "type" : "spaziVuoti",
4     "questionText" : "Giulio Cesare era un console romano.",
5     "answer" :
6         [
7             {
8                 "parolaNumero": 2 //sta ad indicare quale parola e' da oscurare.
9                 In questo caso la numero 2
10            },
11            {
12                "parolaNumero": 5
13            },
14        ],
15     "keywords" :
16         [
17             {
18                 "keyword_1" : "storia",
19             },
20         ],
21     "statistiche" :
22         [
23             {
24                 "level" : 500,
25                 "totalAnswers" : 0,
26                 "correctAnswers" : 0
27             }
28         ]
29 }
```



C Interfaccia REST

Per utilizzare il Web come piattaforma di elaborazione, l'interfaccia per il $Back-End_G$ del progetto Quizzipedia è realizzata in stile $RESTful_G$. L'interfaccia $REST_G$ propone una visione del Web incentrata sul concetto di risorsa, per questo motivo è stato preferito a $SOAP_G$ che espone un insieme di metodi richiamabili da remoto da parte di un $client_G$. Inoltre $SOAP_G$ sfrutta il protocollo $HTTP_G$ come semplice protocollo di trasporto; $REST_G$ lo usa come protocollo di livello applicativo, per questo ne utilizza appieno le potenzialità. I dati scambiati mediante l'interfaccia $REST_G$ sono rappresentati in formato $JSON_G$, che si integra facilmente con le tecnologie ed il linguaggio utilizzati per sviluppare Quizzipedia. Se lo scambio di dati avviene correttamente il $server_G$ può fornire in risposta un messaggio di conferma:

```
1 {  
2   "status" : "ok"  
3 }
```

C.1 Errori

In caso di errori il $server_G$ risponde con un messaggio d'errore in formato $JSON_G$, definito secondo lo schema:

```
1 {  
2   "code" : [codice dell'errore],  
3   "title" : [titolo dell'errore],  
4   "message" : [descrizione testuale dell'errore]  
5 }
```

C.1.1 Errori generici

Gli errori generici sono gestiti dalla classe **ErrorHandler** ad eccezione dell'errore 404 che viene gestito da **NotFoundHandler**. In questo modo si ottiene una gestione degli errori più flessibile e modulare, in quanto è possibile inserire in coda allo stack di chiamate di ogni ruote la classe **NotFoundHandler**, riducendo così la complessità della classe **ErrorHandler**.

C.1.1.1 Errori lato server

Nel caso si verifichi un errore lato server, questo risponde con un codice $HTTP_G$ del tipo 5xx. Di seguito vengono specificati gli errori che possono essere sollevati:

- 500 **Errore sconosciuto**: il messaggio descrive un errore generico del server che avviene quando si verifica una condizione non gestibile e quindi non identificabile con un errore specifico.

C.1.1.2 Errori nelle richieste da parte del client

Nel caso si verifichi un errore riguardo le richieste ricevute dal client, il server risponde con un codice $HTTP_G$ del tipo 4xx. Di seguito vengono specificati gli errori che possono essere sollevati:

- 400: il server non può gestire la richiesta in seguito ad una generica richiesta errata. Il contenuto del messaggio d'errore varia in base alla tipologia di richiesta ricevuta;
- 401: Accesso non autorizzato;
- 404: Pagina non trovata.



C.1.1.3 Errori specifici di QuizziPedia

Questi errori rappresentano situazioni specifiche del sistema QuizziPedia, sono quindi stati definiti dei codici personalizzati per questa tipologia di errori basandosi sull'idea dei codici d'errore $HTTP_G$ standard. I codici dei messaggi d'errore sono stati assegnati secondo diverse categorie, in modo da poter individuare facilmente quale componente dell'applicazione ha causato l'errore. Ad ogni categoria è stato assegnato successivamente un intervallo di codici possibili, evitando di utilizzare gli intervalli **4xx** e **5xx** per non creare ambiguità con i codici d'errore standard descritti nella sezione precedente. Le categorie di errori definite sono le seguenti:

- **1xx**: errori di autenticazione, registrazione, modifica su utente e del middleware `userId`;
 - **101**: credenziali non valide;
 - **102**: password assente;
 - **103**: password troppo corta;
 - **104**: password uguale all'attuale;
 - **105**: password e conferma password non coincidono;
 - **111**: campo obbligatorio vuoto;
 - **112**: username non disponibile;
 - **113**: username non valido;
 - **114**: indirizzo mail non valido;
 - **121**: formato immagine non valido;
 - **122**: immagine di dimensione troppo grande;
 - **123**: errore nel caricamento dell'immagine;
- **2xx**: errori del middleware `questionById`;
- **3xx**: errori del middleware `topicById`;
- **6xx**: errori del middleware `quizById`;
- **7xx**: errori del middleware `summaryById`;
- **8xx**: errori del controller `QuestionController`;
- **9xx**: errori del controller `TopicController`;
- **10xx**: errori del controller `QuizController`;
- **11xx**: errori del controller `SummaryController`;
- **12xx**: errori degli oggetti `QuestionModel`;
- **13xx**: errori degli oggetti `TopicModel`;
- **14xx**: errori degli oggetti `QuizModel`;
- **15xx**: errori degli oggetti `SummaryModel`;

Gli errori specifici di QuizziPediavengono gestiti dalla classe `QuizziPediaError` e sono di seguito elencati sotto forma di:

Codice Titolo: messaggio d'errore

- **100 Utente non trovato**: l'identificativo utente fornito non è un identificativo valido;



- 101 **Credenziali non valide:** è necessario fornire un username ed una password valide;
- 102 **Password assente:** è necessario inserire una password;
- 103 **Password troppo corta:** è necessario inserire una password di almeno 6 caratteri;
- 104 **Password uguale all'attuale:** è necessario inserire una password diversa della precedente;
- 105 **Le password non coincidono:** è necessario che 'password' e 'conferma password' siano identiche;
- 111 **Campo obbligatorio vuoto:** è necessario riempire tutti i campi obbligatori;
- 112 **Username non disponibile:** l'username è già stato utilizzato, prego inserire un nuovo username;
- 113 **Username non valido:** l'username non è valido;
- 114 **Indirizzo mail non valido:** è necessario fornire un indirizzo e-mail valido;
- 121 **Formato immagine non valido:** Il formato dell'immagine non è valido;
- 122 **Immagine di dimensione troppo grande:** La dimensione massima per l'immagine è di 20MB;
- 123 **Caricamento immagine fallito:** errore sconosciuto nel caricamento dell'immagine;
- 200 **Domanda non valida:** l'identificativo della domanda fornita non è un identificativo valido;
- 300 **Argomento non valido:** l'identificativo dell'argomento fornito non è un identificativo valido;
- 600 **Questionario non valido:** l'identificativo del questionario fornito non è un identificativo valido;
- 700 **Sommario non valido:** l'identificativo del questionario fornito non è un identificativo valido;
- 800 **Dati non validi:** i dati relativi al contenuto della domanda non sono validi o sono formattati in modo errato;
- 900 **Dati non validi:** i dati relativi al contenuto dell'argomento non sono validi;
- 1000 **Dati non validi:** i dati relativi al contenuto del questionario non validi;
- 1100 **Dati non validi:** i dati relativi al contenuto del sommario non validi o sono formattati in modo errato;
- 1200 **Dati non validi:** i dati per la modifica di una domanda non sono definiti;
- 1300 **Dati non validi:** i dati per la gestione dell'argomento non sono definiti;
- 1400 **Dati non validi:** i dati per la gestione del questionario non sono definiti;
- 1500 **Dati non validi:** i dati per la gestione del sommario non sono validi;



C.2 Risorse REST

In seguito vengono riportate le risorse $REST_G$ fornite, associate al tipo di metodo $HTTP_G$ che è possibile richiedere su di esse e ai permessi necessari per effettuare la richiesta. In particolare, i permessi sono:

- **Utente**: la risorsa può essere acceduta da qualsiasi tipo di utente;
- **Utente autenticato**: la risorsa può essere acceduta solo dagli utenti che hanno effettuato il login;
- **Utente autenticato pro**: la risorsa può essere acceduta solo dagli utenti pro.

Inoltre per ogni risorsa sono stati specificati i formati per lo scambio dei dati in $JSON_G$:

- **Request**: rappresenta l'oggetto $JSON_G$ che dovrà essere passato alla risorsa $REST_G$;
- **Response**: rappresenta l'oggetto $JSON_G$ che fornirà in risposta la risorsa $REST_G$;
- **/:lang**

- **Method**: GET;
- **Livello di Accesso**: Utente;
- **Descrizione**: restituisce le keywords inerenti alla lingua impostata;
- **Response**: la risposta deve avere i seguenti campi:

```
1 {
2   "keywords": [Array contenente le keywords inerenti alla lingua
               impostata]
3 }
```

- **/:lang/signup**
 - **Method**: POST;
 - **Livello di Accesso**: Utente;
 - **Descrizione**: crea un nuovo account. Un username univoco inserito dall'utente lo identifica, pertanto non ci saranno più utenti con lo stesso username. Restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
 - **Request**: lo scambio dei dati dell'utente avviene attraverso una form che deve avere i seguenti campi:

```
1 {
2   "username" : [username univoco scelto dall'utente]
3   "password" : [la password associata all'account che si vuole
                 registrare]
4   "e-mail" : [l'indirizzo e-mail con cui si effettua la
                registrazione]
5   "name" : [nome dell'utente da registrare]
6   "surname" : [cognome dell'utente da registrare]
7 }
```

- **/:lang/signin**
 - **Method**: POST;
 - **Livello di Accesso**: Utente;



- **descrizione:** effettua l'accesso all'account. Restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
- **Request:** lo scambio dei dati dell'utente avviene attraverso una form che deve avere i seguenti campi:

```
1 [
2 "username" : [username che corrisponde all'username inserita
   durante la registrazione]
3 "password" : [la password associata all'username dell'account
   dell'utente]
4 ]
```

- `/:lang/signout`
 - **Method:** POST;
 - **Livello di Accesso:** Utente autenticato/autenticato pro;
 - **Descrizione:** effettua il logout. restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
 - **Request:** la richiesta deve avere i seguenti campi:

```
1 [
2 "_userId" : [identificativo dell'utente]
3 ]
```

- `/:lang/recovery`
 - **Method:** POST;
 - **Livello di Accesso:** Utente;
 - **Descrizione:** invia una nuova password sulla mail dell'utente generata in automatico, restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
 - **Request:** la richiesta di recupero della password deve avere i seguenti campi:

```
1 [
2 "e-mail" : [la password associata all'username dell'account dell
   'utente]
3 ]
```

- `/:lang/loggedin`
 - **Method:** GET;
 - **Livello di Accesso:** Utente;
 - **Descrizione:** controlla se la sessione è attiva o meno;
 - **Response:** la risposta deve avere i seguenti dati:

```
1 [
2 "user" : [Array di JSON contenente le informazioni riguardanti l
   'utente autenticato]
3 ]
```

- `/:lang/user/:userId/search/:keyword/users`
 - **Method:** GET;



- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** effettua una ricerca di utenti da parte di un utente;
- **Request:** la richiesta deve avere i seguenti campi:

```
1 {
2 "keyword" : [stringa da ricercare]
3 }
```

- **Response:** la risposta deve avere un campo **array** contenente i risultati degli utenti trovati:

```
1 {
2 "userResult" : [
3 "_userID" : [identificativi dell'utente trovato]
4 "name" : [nome dell'utente trovato]
5 "surname" : [cognome dell'utente trovato]
6 ]
7 }
```

- `/:lang/user/:userId/search/:keyword/quizzes`

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** effettua una ricerca di questionari da parte di un utente;
- **Request:** la richiesta deve avere i seguenti campi:

```
1 {
2 "keyword" : [stringa da ricercare]
3 }
```

- **Response:** la risposta deve avere un campo **array** contenente i risultati dei questionari trovati:

```
1 {
2 "quizResult" : [
3 "_quizID" : [identificativo del questionario]
4 "title" : [titolo del questionario]
5 "author" : [username dell'autore del questionario]
6 ]
7 }
```

- `/:lang/user/:userId/search/users/:userId`

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce le informazioni dell'utente precedentemente trovato tramite una ricerca;
- **Response:**

```
1 {
2 "username" : [username utente da visualizzare]
3 "name" : [nome utente da visualizzare]
4 "surname" : [cognome utente da visualizzare]
5 "email" : [email utente da visualizzare]
6 "userImg" : [rappresenta l'immagine dell'utente da visualizzare]
```



```
7 "levelUser" : [livello utente da visualizzare]
8 "statistics": [array di JSON, contenente le statistiche dell'utente
    di ogni argomento]
9 }
```

- /:lang/:user/:userId/search/quizzes/:quizId
 - **Method:** GET;
 - **Livello di Accesso:** Utente autenticato/autenticato pro;
 - **Descrizione:** restituisce le informazioni del questionario precedentemente trovato tramite una ricerca;
 - **Response:**

```
1 {
2 "title" : [identifica il titolo del questionario]
3 "author" : [identifica l'autore del questionario]
4 "questions" : [identifica l'Array di JSON con le domande
    relative al questionario]
5 }
```
- /:lang/:user/:userId
 - **Method:** DELETE;
 - **Livello di Accesso:** Utente autenticato/autenticato pro;
 - **Descrizione:** elimina l'utente autenticato. Restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
- /:lang/:user/:userId
 - **Method:** GET;
 - **Livello di Accesso:** Utente autenticato/autenticato pro;
 - **Descrizione:** restituisce le informazioni riguardante l'utente autenticato. Restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
 - **Response:** la risposta deve contenere i seguenti campi:

```
1 {
2 "username" : [username utente da visualizzare]
3 "name" : [nome utente da visualizzare]
4 "surname" : [cognome utente da visualizzare]
5 "email" : [email utente da visualizzare]
6 "userImg": [rappresenta l'immagine dell'utente da visualizzare]
7 "experienceLevel" : [livello esperienza dell'utente ]
8 }
```
- /:lang/:user/:userId
 - **Method:** PUT;
 - **Livello di Accesso:** Utente autenticato/autenticato pro;
 - **Descrizione:** modifica le informazioni riguardante l'utente autenticato. Restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
 - **Request:** la richiesta deve contenere i seguenti campi:



```
1 {
2 "name" : [nome utente]
3 "surname" : [cognome utente]
4 "email" : [email utente]
5 "userImg" : [immagine profilo utente]
6 }
```

- /:lang/user/:userId/privacy

- **Method:** PUT;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** modifica la password di accesso al sistema riguardante l'utente autenticato. Restituisce un messaggio di conferma se viene effettuato correttamente, altrimenti un errore;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "password" : [identifica la nuova password inserita dall'utente]
3 }
```

- /:lang/user/:userId/statistics

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce le statistiche riguardanti l'utente autenticato;
- **Response:** la risposta deve contenere i seguenti campi:

```
1 {
2 "name" : [identifica il nome dell'utente]
3 "surname" : [identifica il cognome dell'utente]
4 "userImg" : [rappresenta l'immagine dell'utente]
5 "statistics" : [array di JSON, contenente le statistiche dell'utente
    di ogni argomento]
6 "summaries" : [array di JSON contenente la cronologia dei
    questionari svolti dall'utente]
7 }
```

- /:lang/user/:userId/statistics/summary

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce la cronologia dei questionari svolti;
- **Response:** la risposta deve contenere i seguenti campi:

```
1 {
2 summary : [
3 {
4   summaryId : [identifica l'id di una summary]
5 }
6 ]
7 }
```

- /:lang/user/:userId/statistics/summary/:summaryId



- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce le statistiche di un particolare questionario svolto;
- **Response:** la risposta deve contenere i seguenti campi:

```
1 [
2 "quizId" : [identifica l'id del quiz svolto]
3 "givenAnswer" : [array di JSON contenente le domande del quiz
    associate alle risposte date dall'utente]
4 "mark" : [identifica la valutazione finale del questionario
    svolto]
5 ]
```

- [/:lang/user/:userId/question](#)

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce le domande create dall'utente;
- **Response:** la risposta deve avere i seguenti campi:

```
1 [
2 "listQuestion" : [
3 "question_id" : [identifica l'identificativo della domanda]
4 "makeWith" : [identifica lo strumento utilizzato per creare la
    domanda]
5 "language" : [identifica la lingua della domanda]
6 "question" : [identifica l'array di JSON contenente gli
    attributi che formano una domanda]
7 "keywords" : [identifica le keywords inerenti alla domanda]
8 "totalAnswer" : [identifica il numero totale di risposte date
    alla domanda]
9 "correctAnswer" : [identifica il numero di risposte corrette
    date alla domanda]
10 ]
11 ]
```

- [/:lang/user/:userId/question/:questionId](#)

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce la domanda creata dall'utente;
- **Response:** la risposta deve avere i seguenti campi:

```
1 [
2 "listQuestion" : [
3 "makeWith" : [identifica tipologia domanda]
4 "language" : [identifica la lingua della domanda]
5 "question" : [identifica l'array di JSON contenente gli
    attributi che formano una domanda]
6 "totalAnswer" : [identifica il numero totale di risposte date
    alla domanda]
7 "correctAnswer" : [identifica il numero di risposte corrette
    date alla domanda]
8 ]
9 ]
```



- /:lang/:user/:userId/question

- **Method:** POST;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** aggiunge una domanda nel sistema, restituisce un messaggio di conferma o di errore;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "listQuestion" : [
3 "makeWith" : [identifica tipologia domanda]
4 "language" : [identifica la lingua della domanda]
5 "question" : [identifica l'array di JSON contenente gli
    attributi che formano una domanda]
6 ]
7 }
```

- /:lang/:user/:userId/quiz

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato pro;
- **Descrizione:** restituisce i questionari creati dall'utente pro;
- **Response:** la risposta deve avere i seguenti campi:

```
1 {
2 "listQuiz" : [
3 "title" : [identifica il titolo del questionario]
4 "questions" : [Array di JSON contenente le domande relative al
    questionario]
5 ]
6 }
```

- /:lang/:user/:userId/quiz

- **Method:** POST;
- **Livello di Accesso:** Utente autenticato pro;
- **Descrizione:** aggiunge un questionario nel sistema, restituisce un messaggio di conferma o di errore;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "title" : [titolo del questionario]
3 "questions" : [Array di JSON contenente gli identificativi delle
    domande che compongono il questionario]
4 }
```

- /:lang/:user/:userId/quiz/:quizId

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato pro;
- **Descrizione:** visualizza il questionario creato l'utente e le relative informazioni, restituisce un messaggio di conferma o di errore;



- **Response:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "questions" : [array di JSON contenente le domande che formano
     il questionario]
3 "registeredUsers" :[array di JSON contenente gli utenti
     registrati al questionario]
4 "activeUsers" :[array di JSON contenente gli utenti che hanno
     svolto il questionario]
5 "statistics" :[array di JSON contenente le statistiche del
     questionario]
6 }
```

- /:lang/user/:userId/quiz/:quizId/removeUser

- **Method:** PUT;
- **Livello di Accesso:** Utente autenticato pro;
- **Descrizione:** elimina un utente iscritto ad un questionario, restituisce un messaggio di conferma o di errore;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "_userId" : [identificativo dell'utente da iscrivere al
     questionario]
3 }
```

- /:lang/user/:userId/quiz/:quizId/addUser

- **Method:** POST;
- **Livello di Accesso:** Utente autenticato pro;
- **Descrizione:** iscrive un utente ad un questionario, restituisce un messaggio di conferma o di errore;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "_userId" : [identificativo dell'utente da iscrivere al
     questionario]
3 }
```

- /:lang/user/:userId/quiz/:quizId/activeUser

- **Method:** POST;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** aggiunge un utente iscritto al questionario nella lista degli utenti che lo hanno eseguito;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "_userId" : [identificativo dell'utente da iscrivere al
     questionario]
3 }
```

- /:lang/user/:userId/quiz/:quizId



- **Method:** PUT;
- **Livello di Accesso:** Utente autenticato pro;
- **Descrizione:** modifica in questionario creato in precedenza, restituisce un messaggio di conferma o di errore;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "title" : [titolo del questionario]
3 "questions" : [Array di JSON contenente gli identificativi delle
    domande che compongono il questionario]
4 }
```

- /:lang/user/:userId/quiz/:quizId/test

- **Method:** GET;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** restituisce il quiz selezionato dall'utente per effettuare l'esercitazione;
- **Response:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "title" : [titolo del questionario]
3 "author" : [identifica l'autore del questionario]
4 "questions" : [Array di JSON contenente le domande che
    compongono il questionario]
5 }
```

- /:lang/user/:userId/quiz/:quizId/summary

- **Method:** POST;
- **Livello di Accesso:** Utente autenticato/autenticato pro;
- **Descrizione:** crea il riepilogo del questionario svolto;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2 "givenAnswers" : [Array di JSON contenente le domande del quiz
    associate alle risposte date dall'utente]
3 }
```

- /:lang/user/training/question

- **Method:** POST;
- **Livello di Accesso:** Utente;
- **Descrizione:** restituisce una domanda in base al livello di abilità raggiunto dall'utente;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1 {
2     "topic" : [indica l'argomento in cui si vuoi esercitare
        l'utente]
3     "keywords" : [indica le keywords per effettuare la
        ricerca della domanda]
4     "level" : [indica il livello dell'utente inerente all'
        argomento scelto]
5 }
```



- **Response:** la risposta deve contenere i seguenti campi:

```
1      {
2          "_questionId" : [identificativo della domanda]
3          "makeWith" : [identifica lo strumento utilizzato per
4                          creare la domanda]
5          "language" : [identifica la lingua della domanda]
6          "question" : [identifica l'array di JSON contenente gli
7                          attributi che formano una domanda]
8      }
```

- `/:lang/:user/:userId/training/userstatistics`

- **Method:** PUT;
- **Livello di Accesso:** Utente autenticato;
- **Descrizione:** aggiorna le statistiche dell'utente dopo che ha risposto ad una domanda;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1      {
2          "statistics" : [statistiche relative alla domanda]
3          "level" : [livello dell'utente]
4      }
```

- `/:lang/:user/:userId/training/questionstatistics`

- **Method:** PUT;
- **Livello di Accesso:** Utente autenticato;
- **Descrizione:** aggiorna le statistiche della domanda;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1      {
2          "level" : [livello della domanda]
3          "totalAnswers" : [risposte totali date alla domanda]
4          "correctAnswers" : [risposte corrette totali date alla
5                               domanda]
6      }
```

- `/:lang/:user/training/userlevelupdate`

- **Method:** PUT;
- **Livello di Accesso:** Utente non autenticato;
- **Descrizione:** aggiorna il livello dell'utente non autenticato;
- **Request:** la richiesta deve contenere i seguenti campi:

```
1      {
2          "level" : [livello della domanda]
3      }
```

D Calcolo delle Statistiche

Di seguito sarà illustrato come verranno calcolate le statistiche di utente, di una domanda e di un argomento secondo le risposte (corrette o meno) date ad una singola domanda.



D.1 Statistiche Utente

D.1.1 Utente autenticato

L'utente autenticato possiede due tipologie di livello:

- **Experience level:** livello che cresce sempre man mano che un utente risponde alle domande, secondo un sistema a punti:

- +1 punto per ogni risposta errata;
- +2 punti per ogni risposta esatta;

Il livello sarà rappresentato da una barra d'esperienza che deve essere riempita totalmente per poter passare al livello successivo; per riempire la barra è necessario accumulare un ammontare di punti tanto maggiore quanto è alto il livello dell'utente. Il livello iniziale è 1.

- **Skill level:** livello che varia a seconda della risposta (corretta o errata) data dall'utente ad una domanda di un certo argomento. La variazione sarà calcolata sulla base della differenza tra lo *skill level* attuale dell'utente e il *difficulty level* della domanda a cui si sta rispondendo secondo i valori indicati in tabella.

Differenza	$0 \leq skill\ level - difficulty\ level \leq 100$		$0 \leq difficulty\ level - skill\ level \leq 100$	
	corretta	errata	corretta	errata
91 - 100	+2	-11	+11	-2
81 - 90	+3	-10	+10	-3
71 - 80	+4	-9	+9	-4
61 - 70	+5	-8	+8	-5
51 - 60	+6	-7	+7	-6
41 - 50	+7	-6	+6	-7
31 - 40	+8	-5	+5	-8
21 - 30	+9	-4	+4	-9
11 - 20	+10	-3	+3	-10
1 - 10	+11	-2	+2	-11
	corretta	errata		
0	+1	-1		

Ogni utente autenticato avrà uno *skill level* per ogni argomento presente nel sistema. Inizialmente lo *skill level* sarà uguale a 500 e potrà variare all'interno di un intervallo tra 0 e 1000.

I livelli precedentemente descritti verranno salvati e aggiornati automaticamente dal sistema.

D.1.2 Utente non autenticato

L'utente non autenticato possiederà solamente la tipologia di livello *skill level*, con la differenza che non verrà salvato dal sistema al termine di un allenamento.

D.2 Statistiche Domanda

La domanda possiederà un *difficulty level* che varia a seconda della risposta (corretta o errata) data dall'utente alla domanda. La variazione sarà calcolata sulla base della differenza tra lo *skill level* dell'utente autenticato e il *difficulty level* attuale della domanda secondo i valori indicati in tabella.



Differenza	$0 \leq skill\ level - difficulty\ level \leq 100$		$0 \leq difficulty\ level - skill\ level \leq 100$	
	corretta	errata	corretta	errata
91 - 100	-2	+11	+11	-2
81 - 90	-3	+10	+10	-3
71 - 80	-4	+9	+9	-4
61 - 70	-5	+8	+8	-5
51 - 60	-6	+7	+7	-6
41 - 50	-7	+6	+6	-7
31 - 40	-8	+5	+5	-8
21 - 30	-9	+4	+4	-9
11 - 20	-10	+3	+3	-10
1 - 10	-11	+2	+2	-11
	corretta	errata		
0	-1	+1		

Inizialmente il *difficulty level* sarà uguale a 500 e potrà variare all'interno di un intervallo tra 0 e 1000 e soltanto se a rispondere è un utente autenticato.

D.3 Statistiche Argomento

La percentuale di risposte corrette ($\%R_C$) è data dal rapporto tra risposte corrette totali (R_{CT}) e risposte totali (R_T) date alle domande relative all'argomento.

$$\%R_C = \frac{R_{CT}}{R_T} \times 100$$

D.4 Statistiche Questionario

Per un questionario vi saranno due statistiche:

- **Generale:** la percentuale di risposte corrette ($\%R_C$) è data dal rapporto tra risposte corrette totali (R_{CT}) e risposte totali (R_T) date alle domande del questionario.

$$\%R_C = \frac{R_{CT}}{R_T} \times 100$$

- **Per utente:** la percentuale di risposte corrette dell'utente ($\%R_{CU}$) è data dal rapporto tra risposte corrette totali dell'utente (R_{CTU}) e risposte totali dell'utente (R_{TU}) date alle domande del questionario.

$$\%R_{CU} = \frac{R_{CTU}}{R_{TU}} \times 100$$

E Algoritmo di selezione della domanda

Le domande che verranno presentate durante un allenamento verranno scelte secondo lo *skill level* dell'utente. L'algoritmo funzionerà come segue:

1. Verifica lo *skill level* dell'utente;
2. Definisce il range di possibili domande da presentare all'utente;
3. Genera uno o due numeri casuali compresi tra 1 e 100 che permettono di scegliere in quale intervallo di probabilità pescare la domanda, secondo i valori indicati in tabella.
Generazione del primo numero:



Numero generato	Range della domanda
1 - 50	580 - 620
51 - 75	621 - 660
76 - 85	661 - 700
86 - 95	540 - 579
96 - 100	500 - 539

Nel caso in cui il primo numero generato sia compreso tra 1 e 50, l'algoritmo ne genera un secondo:

Numero generato	Range della domanda
1 - 70	600 - 620
71 - 100	580 - 599

4. Sceglie una domanda casuale da questo insieme.

Qui di seguito proponiamo un'immagine rappresentativa di come funziona l'algoritmo per facilitare la comprensione.

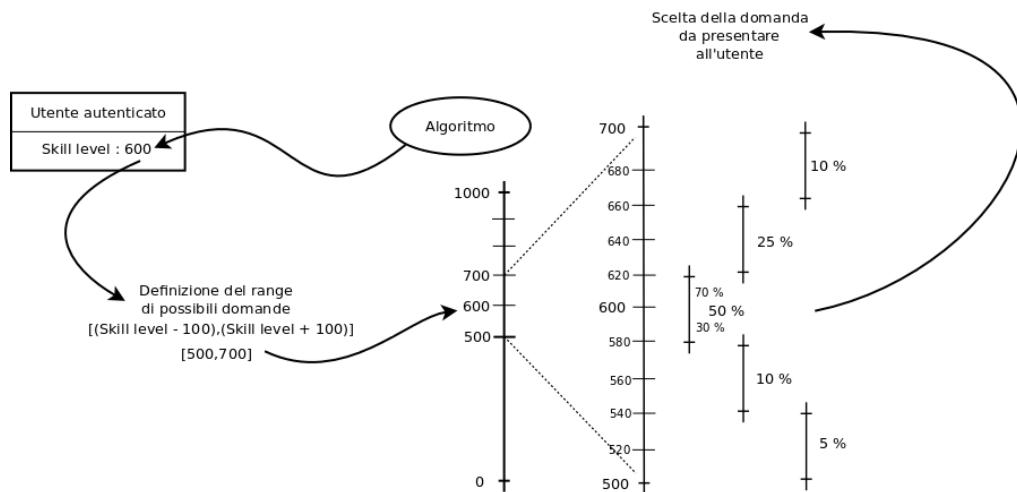


Figura 266: Funzionamento Algoritmo di selezione della domanda