

QUIZZIPEDIA



team404swe@gmail.com

Definizione di Prodotto 2.0

Informazioni sul documento	
Nome Documento	Definizione di Prodotto
Versione	2.0
Uso	Esterno
Data Creazione	20 maggio 2016
Data Ultima Modifica	25 luglio 2016
Redazione	D. Bortot, A. Beccaro, A. Multineddu
Verifica	Luca Alessio
Approvazione	Marco Crivellaro
Committente	Zucchetti SPA
Lista di distribuzione	Prof. Vardanega Tullio TEAM404

Registro delle modifiche

Versione	Autore	Data	
1.0.1	Davide Bortot (Progettista)	15/07/2016	Aggiunti metodi e attributi in §4.1.2. Aggiunto metodo in §4.1.1
1.0	Alex Beccaro (Responsabile)	24/06/2016	Approvazione del documento.
0.1.0	Marco Crivellaro (Verificatore)	22/06/2016	Verifica del documento.
0.0.11	Davide Bortot (Progettista)	20/06/2016	Definizione di pre e post condizioni per i metodi del ViewModel (§4).
0.0.10	Andrea Multineddu (Progettista)	18/06/2016	Aggiornamento immagini package "Model".
0.0.9	Davide Bortot (Progettista)	17/06/2016	Definizione delle sezioni §4.2, §4.3, §4.4, §4.5.
0.0.8	Davide Bortot (Progettista)	16/06/2016	Definizione della sezione "Controllers" §4.1.
0.0.7	Andrea Multineddu (Progettista)	15/06/2016	Definizione package "Model".
0.0.6	Luca Alessio (Analista)	14/06/2016	Prima stesura sezione "Tracciamento".
0.0.5	Andrea Multineddu (Progettista)	11/06/2016	Prima stesura sezione "Model".
0.0.4	Luca Alessio (Analista)	09/06/2016	Stesura campi "Funzione del componente" e "Relazione con altri componenti" di tutti gli elementi.
0.0.3	Luca Alessio (Analista)	08/06/2016	Adattamento delle varie sezioni alle nuove indicazioni contenute nel documento riveduto specifica_tecnica_2.0.pdf (re-impaginazione dei paragrafi e revisione della struttura del documento).
0.0.2	Davide Bortot (Progettista)	05/06/2016	Impostata la struttura della sezioni "ViewModel", "View", "Model" e della sezione "Tracciamento".
0.0.1	Davide Bortot (Progettista)	30/05/2016	Creazione e impostazione documento. Definite sezioni "Sommario" e "Introduzione".

Indice

1	Introduzione	2
1.1	Scopo del documento	2
1.2	Scopo del prodotto	2
1.3	Glossario	2
1.4	Riferimenti	2
1.4.1	Normativi	2
1.4.2	Informativi	2
2	Package View	3
2.1	View::Pages	3
2.1.1	View::Pages::Page	3
2.1.2	View::Pages::LoginPage	3
2.1.3	View::Pages::RegistrationPage	4
2.1.4	View::Pages::PasswordRecoveryPage	4
2.1.5	View::Pages::QuizCreationPage	5
2.1.6	View::Pages::QuestionUpdatePage	5
2.1.7	View::Pages::QuestionCreationPage	6
2.1.8	View::Pages::QuestionManagementPage	6
2.1.9	View::Pages::QuizResultsPage	7
2.1.10	View::Pages::QuizExecutionPage	8
2.1.11	View::Pages::QuizListPage	9
2.1.12	View::Pages::CategoryListPage	9
2.2	View::Templates	10
2.2.1	View::Templates::QuestionList	10
2.2.2	View::Templates::Question	11
2.2.3	View::Templates::QuizList	11
2.2.4	View::Templates::Quiz	11
2.2.5	View::Templates::QuestionForm	12
2.2.6	View::Templates::QuizCreationForm	12
2.2.7	View::Templates::QuestionCompilation	12
2.2.8	View::Templates::QuizResults	13
2.2.9	View::Templates::SearchForm	13
3	Package Model	14
3.1	Model::Database	14
3.1.1	Model::Database::QuizManager	14
3.1.2	Model::Database::QuestionManager	15
3.2	Model::Parser	16
3.2.1	Model::Parser::Parser	16
3.3	Model::Statistics	16
3.3.1	Model::Statistics::Statistics	16
3.4	Model::Publishers	17
3.4.1	Model::Publishers::QuizPublisher	17
3.4.2	Model::Publishers::QuestionPublisher	18

4	Package ViewModel	19
4.1	ViewModel::Controllers	19
4.1.1	ViewModel::Controllers::NewQuestionController	19
4.1.2	ViewModel::Controllers::NewQuizController	20
4.1.3	ViewModel::Controllers::QuizListController	21
4.1.4	ViewModel::Controllers::QuizDetailsController	22
4.1.5	ViewModel::Controllers::DeleteQuestionController	22
4.1.6	ViewModel::Controllers::DeleteQuizController	23
4.1.7	ViewModel::Controllers::QuizManagementController	24
4.1.8	ViewModel::Controllers::QuestionsManagementController	25
4.1.9	ViewModel::Controllers::QMLEditorController	25
4.2	ViewModel::Subscribers	26
4.2.1	ViewModel::Subscribers::QuestionsSubscriber	26
4.2.2	ViewModel::Subscribers::QuizSubscriber	27
4.2.3	ViewModel::Subscribers::UsersSubscriber	27
4.3	ViewModel::Methods	28
4.3.1	ViewModel::Methods::QuestionMethods	28
4.3.2	ViewModel::Methods::QuizMethods	29
4.3.3	ViewModel::Methods::UserMethods	29
4.4	ViewModel::Interpreter	30
4.4.1	ViewModel::Interpreter::Interpreter	30
4.4.2	ViewModel::Interpreter::InterpreterFactory	30
4.4.3	ViewModel::Interpreter::QMLInterpreterFactory	31
4.4.4	ViewModel::Interpreter::QMLInterpreter	31
4.4.5	ViewModel::Interpreter::QML2HTMLInterpreter	32
4.5	ViewModel::Router	32
4.5.1	ViewModel::Router::Router	32
5	Tracciamento	34
5.1	Mappatura classi - requisiti	34
5.2	Mappatura requisiti - classi	38

Elenco delle figure

Elenco delle tabelle

2	Mappatura delle classi sui requisiti	34
3	Mappatura dei requisiti sulle classi	38

Sommario

Questo documento, redatto dal gruppo **Team404**, contiene la descrizione di dettaglio dell'architettura software sulla quale verrà sviluppato il progetto Quizzipedia, commissionato da Zucchetti S.p.A.. Le specifiche del documento si basano sull'architettura generale descritta nel documento "*specifica_tecnica_2.0.pdf*".

1 Introduzione

1.1 Scopo del documento

Il documento ha lo scopo di definire nel dettaglio la struttura del sistema Quizzipedia, approfondendone la descrizione già definita nel documento di Specifica Tecnica. Per ogni package del sistema verrà data una descrizione estensiva delle sue classi. Per poter sviluppare al meglio il prodotto, i programmatori dovranno attenersi alle specifiche definite in questo documento.

1.2 Scopo del prodotto

Il progetto **Quizzipedia** ha come obiettivo lo sviluppo di un sistema software basato su tecnologie Web (Javascript₆, Node.js₆, HTML5₆, CSS3₆) che permetta la creazione, gestione e fruizione di questionari. Il sistema dovrà quindi poter archiviare i questionari suddivisi per argomento, le cui domande dovranno essere raccolte attraverso uno specifico linguaggio di markup (Quiz Markup Language) d'ora in poi denominato QML₆. In un caso d'uso a titolo esemplificativo, un "esaminatore" dovrà poter costruire il proprio questionario scegliendo tra le domande archiviate, ed il questionario così composto sarà presentato e fruibile all' "esaminando", traducendo l'oggetto QML in una pagina HTML₆, tramite un'apposita interfaccia web. Il sistema presentato dovrà inoltre poter proporre questionari preconfezionati e valutare le risposte fornite dall'utente finale.

Per un'analisi più precisa ed approfondita del progetto si rimanda al documento "*analisi_dei_requisiti_4.0.pdf*".

1.3 Glossario

Viene allegato un glossario nel file "*glossario_4.0.pdf*" nel quale viene data una definizione a tutti i termini che in questo documento appaiono con il simbolo '6' a pedice.

1.4 Riferimenti

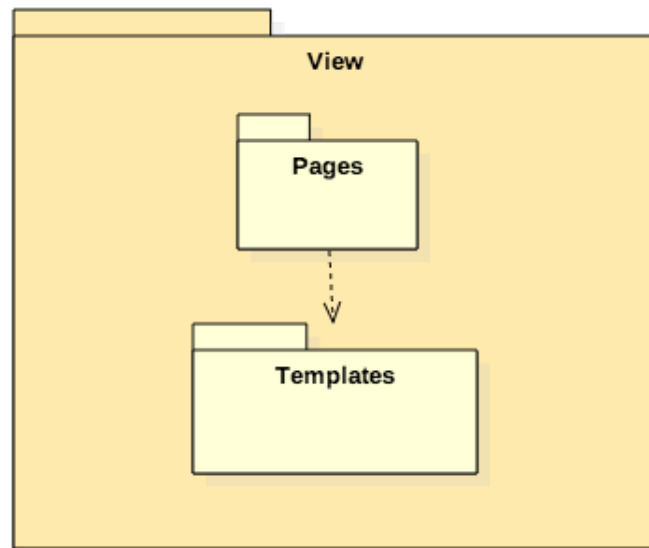
1.4.1 Normativi

- Capitolato d'appalto Quizzipedia:
<http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C5.pdf>
- Norme di Progetto: "*norme_di_progetto_4.0.pdf*"

1.4.2 Informativi

- Corso di Ingegneria del Software anno 2015/2016:
<http://www.math.unipd.it/~tullio/IS-1/2015/>
- Regole del progetto didattico:
<http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/PD01.pdf>
<http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/>
- Specifica Tecnica: "*specifica_tecnica_2.0.pdf*"
- Framework Meteor:
<https://www.meteor.com/>
- Framework AngularJs:
<https://www.angularjs.org/>

2 Package View



2.1 View::Pages

2.1.1 View::Pages::Page

- **Funzione del componente:** rappresenta una pagina web
- **Relazioni d'uso con altre componenti:** L'interfaccia Page viene concretizzata dalle sue classi derivate, una rappresentativa per ogni pagina dell'applicazione
- **Attributi:**
 - - controller: riferimento al controller della page
- **Metodi:**
 - + show(): metodo astratto che mostra il contenuto della pagina

2.1.2 View::Pages::LoginPage

- **Funzione del componente:** visualizza il form di autenticazione e permette il login dell'utente. Fornisce inoltre un link alla pagina di registrazione e uno alla pagina di recupero della password
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template LoginForm

La classe utilizza:

- View::Pages::Page
- View::Templates::LoginForm

- **Metodi:**

- + show(): visualizza il form di autenticazione al sistema
Precondizioni: viene richiesta la pagina di autenticazione. L'utente non è ancora autenticato
Postcondizioni: viene visualizzata la pagina di autenticazione

2.1.3 View::Pages::RegistrationPage

- **Funzione del componente:** visualizza il form di registrazione. Fornisce inoltre un link alla pagina di login
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template RegistrationForm

La classe utilizza:

- View::Pages::Page
- View::Templates::RegistrationForm

- **Metodi:**

- + show(): visualizza il form di registrazione al sistema
Precondizioni: viene richiesta la pagina di registrazione. L'utente non è autenticato
Postcondizioni: viene visualizzata la pagina di registrazione

2.1.4 View::Pages::PasswordRecoveryPage

- **Funzione del componente:** visualizza il form per il recupero della password
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template PasswordRecoveryForm

La classe utilizza:

- View::Pages::Page
- View::Templates::PasswordRecoveryForm

- **Metodi:**

- + `show()`: visualizza il form per il recupero della password

Precondizioni: viene richiesta la pagina della procedura per il recupero della password. L'utente non è autenticato

Postcondizioni: viene visualizzata la pagina della procedura per il recupero della password

2.1.5 View::Pages::QuizCreationPage

- **Funzione del componente:** visualizza il form di creazione di un nuovo questionario
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuizCreationForm

La classe utilizza:

- View::Pages::Page
- View::Templates::QuizCreationForm

- **Attributi:**

- - `user`: l'utente autenticato che sta creando il questionario

- **Metodi:**

- + `show()`: visualizza il form per la creazione di un questionario

Precondizioni: viene richiesta la pagina di creazione di un questionario. L'utente è autenticato

Postcondizioni: viene visualizzata la pagina di creazione di un questionario

2.1.6 View::Pages::QuestionUpdatePage

- **Funzione del componente:** visualizza il form di modifica di una domanda
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuestionForm

La classe utilizza:

- View::Pages::Page
- View::Templates::QuestionForm

- **Attributi:**

- - `user`: l'utente autenticato che sta modificando la domanda

- - question: la domanda da modificare

- **Metodi:**

- + show(): visualizza il form per la modifica di una domanda compilato con i dati attuali
Precondizioni: viene richiesta la pagina di modifica di una domanda. L'utente è autenticato
Postcondizioni: viene visualizzata la pagina di modifica di una domanda

2.1.7 View::Pages::QuestionCreationPage

- **Funzione del componente:** visualizza il form di creazione di una nuova domanda
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuestionForm

La classe utilizza:

- View::Pages::Page
- View::Templates::QuestionForm

- **Attributi:**

- - user: l'utente autenticato che sta creando la domanda

- **Metodi:**

- + show(): visualizza il form per la creazione di una domanda
Precondizioni: viene richiesta la pagina di creazione di una domanda. L'utente è autenticato
Postcondizioni: viene visualizzata la pagina di creazione di una domanda

2.1.8 View::Pages::QuestionManagementPage

- **Funzione del componente:** visualizza la lista delle domande create dall'utente
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuestionList

La classe utilizza:

- View::Pages::Page
- View::Templates::QuestionList

– View::Templates::Question

- **Attributi:**

- user: l'utente autenticato che sta gestendo le proprie domande
- questions: le domande dell'utente user

- **Metodi:**

- + show(): visualizza la lista delle domande dell'utente permettendone la modifica e l'eliminazione. Permette inoltre di creare una nuova domanda, di ordinare la lista e fare una ricerca tra le domande

Precondizioni: viene richiesta la pagina di gestione delle proprie domande. L'utente è autenticato

Postcondizioni: viene visualizzata la pagina di gestione delle proprie domande

- + sort(): ordina la lista delle domande secondo il criterio by

Precondizioni: l'utente ha selezionato un criterio di ordinamento

Postcondizioni: la lista di domande viene ordinata secondo il criterio inserito

Parametri:

- * by: criterio di ordinamento

- + search(): visualizza la lista delle domande che soddisfano il criterio di ricerca query

Precondizioni: l'utente ha selezionato un criterio di ricerca

Postcondizioni: la lista di domande viene filtrata secondo il criterio di ricerca

Parametri:

- * query: criterio di ricerca

2.1.9 View::Pages::QuizResultsPage

- **Funzione del componente:** visualizza i risultati del questionario appena compilato
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuizResults

La classe utilizza:

- View::Pages::Page
- View::Templates::QuizResults

- **Attributi:**

- quizResults: i risultati del quiz appena compilato dall'utente

- **Metodi:**

- + `show()`: visualizza i risultati ottenuti nel quiz appena compilato

Precondizioni: viene richiesta la pagina di visualizzazione dei risultati di un quiz. L'utente ha compilato un quiz

Postcondizioni: viene visualizzata la pagina di visualizzazione dei risultati di un quiz.

2.1.10 View::Pages::QuizExecutionPage

- **Funzione del componente:** visualizza un questionario (una domanda alla volta) e tutti i dati relativi (tempo rimasto, numero domande, ecc...)
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuestionCompilation

La classe utilizza:

- View::Pages::Page
 - View::Templates::QuestionCompilation

- **Attributi:**

- - `timer`: il tempo rimasto per la compilazione del questionario

- **Metodi:**

- + `show()`: visualizza i dati del questionario e la domanda corrente. Permette l'inserimento o la scelta della risposta

Precondizioni: viene richiesta la pagina di compilazione di un quiz. L'utente ha scelto un quiz

Postcondizioni: viene visualizzata la pagina di compilazione di un quiz.

- + `nextQuestion()`: visualizza la domanda successiva nel riquadro della domanda corrente

Precondizioni: viene richiesta la domanda successiva. L'utente ha compilato la domanda corrente o ha deciso di passare alla domanda successiva

Postcondizioni: viene visualizzata la domanda successiva.

- - `previousQuestion()`: visualizza la domanda precedente nel riquadro della domanda corrente

Precondizioni: viene richiesta la domanda precedente. L'utente ha deciso di passare alla domanda precedente

Postcondizioni: viene visualizzata la domanda precedente.

2.1.11 View::Pages::QuizListPage

- **Funzione del componente:** visualizza una lista di questionari
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente e utilizza il template QuizList

La classe utilizza:

- View::Pages::Page
- View::Templates::QuizList
- View::Templates::Quiz

- **Attributi:**

- - quizList: la lista di quiz da visualizzare

- **Metodi:**

- + show(): visualizza la lista dei questionari. Permette inoltre di ordinare la lista e fare una ricerca tra i questionari
Precondizioni: viene richiesta una pagina che mostri una lista di questionari
Postcondizioni: viene visualizzata una pagina che mostra una lista di questionari
- + sort(): ordina la lista dei questionari secondo il criterio by
Precondizioni: l'utente ha selezionato un criterio di ordinamento
Postcondizioni: la lista di questionari viene ordinata secondo il criterio inserito
Parametri:
 - * by: criterio di ordinamento
- + search(): visualizza la lista dei questionari che soddisfano il criterio di ricerca query
Precondizioni: l'utente ha selezionato un criterio di ricerca
Postcondizioni: la lista dei questionari viene filtrata secondo il criterio di ricerca
Parametri:
 - * query: criterio di ricerca

2.1.12 View::Pages::CategoryListPage

- **Funzione del componente:** visualizza la lista delle categorie
- **Relazioni d'uso con altre componenti:** concretizza l'interfaccia Page da cui è diretta discendente

La classe utilizza:

– View::Pages::Page

- **Attributi:**

- categories: la lista delle categorie da visualizzare

- **Metodi:**

- + show(): visualizza la lista delle categorie. Permette inoltre di ordinare la lista e fare una ricerca tra le categorie

- Precondizioni:** viene richiesta una pagina che mostri una lista di categorie

- Postcondizioni:** viene visualizzata una pagina che mostra una lista di categorie

- + sort(by): ordina la lista delle categorie secondo il criterio by

- Precondizioni:** l'utente ha selezionato un criterio di ordinamento

- Postcondizioni:** la lista di categorie viene ordinata secondo il criterio inserito

- Parametri:**

- * by: criterio di ordinamento

- + search(): visualizza la lista delle categorie che soddisfano il criterio di ricerca query

- Precondizioni:** l'utente ha selezionato un criterio di ricerca

- Postcondizioni:** la lista delle categorie viene filtrata secondo il criterio di ricerca

- Parametri:**

- * query: criterio di ricerca

2.2 View::Templates

2.2.1 View::Templates::QuestionList

- **Funzione del componente:** visualizza una lista di domande

- **Relazioni d'uso con altre componenti:**

- La classe utilizza:

- View::Templates::Question

- **Attributi:**

- + controller: String: variabile contenente il nome del controller del template;

- + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;

- + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template

2.2.2 View::Templates::Question

- **Funzione del componente:** visualizza una domanda inserita in una lista
- **Relazioni d'uso con altre componenti:** La classe è utilizzata da:
 - View::Templates::QuestionList
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
 - + category: String: stringa contenente la categoria della domanda
 - + type: String: stringa contenente il tipo della domanda

2.2.3 View::Templates::QuizList

- **Funzione del componente:** visualizza una lista di quiz
- **Relazioni d'uso con altre componenti:**
La classe utilizza:
 - View::Templates::Quiz
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template

2.2.4 View::Templates::Quiz

- **Funzione del componente:** visualizza un quiz inserito in una lista
- **Relazioni d'uso con altre componenti:**
La classe è utilizzata da:
 - View::Templates::QuizList
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;

- + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
- + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
- title: String: stringa contenente il titolo del quiz
- categories: String: stringa contenente le categorie del quiz

2.2.5 View::Templates::QuestionForm

- **Funzione del componente:** visualizza il form per i dati di una domanda. Utilizzabile sia per la creazione che per la modifica della domanda (se viene modificata una domanda già esistente nei campi vengono inseriti i valori attuali)
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
 - + category: String: stringa contenente la categoria della domanda
 - + text: String: stringa contenente il testo della domanda

2.2.6 View::Templates::QuizCreationForm

- **Funzione del componente:** visualizza il form di creazione di un questionario
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template

2.2.7 View::Templates::QuestionCompilation

- **Funzione del componente:** visualizza una domanda e ne permette la compilazione
- **Relazioni con altre componenti**
La classe utilizza:

- View::Templates::Question
- View::Templates::QuizResults

- **Attributi:**

- + controller: String: variabile contenente il nome del controller del template;
- + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
- + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template

2.2.8 View::Templates::QuizResults

- **Funzione del componente:** visualizza i risultati ottenuti in seguito alla compilazione di un quiz
- **Relazioni con altre componenti**
La classe utilizza:

- View::Templates::QuestionCompilation

- **Attributi:**

- + controller: String: variabile contenente il nome del controller del template;
- + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
- + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template

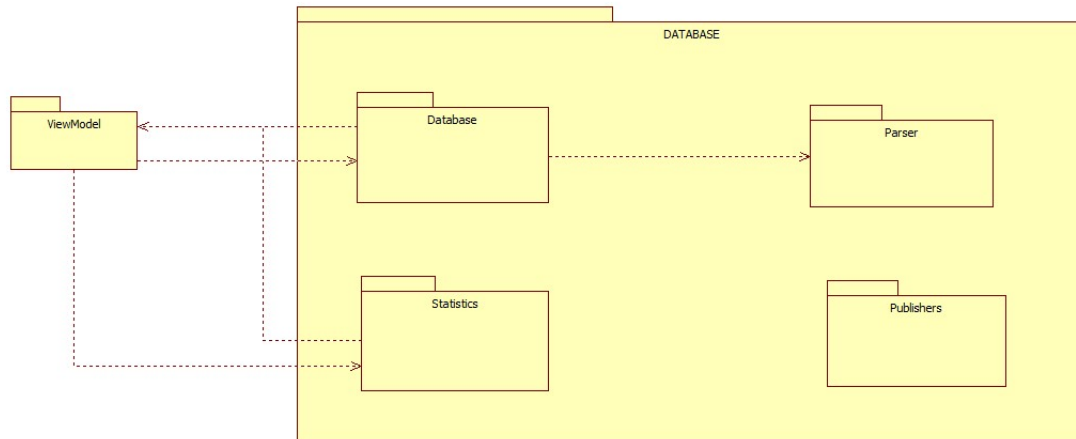
2.2.9 View::Templates::SearchForm

- **Funzione del componente:** visualizza una form per l'inserimento dei dati desiderati e l'avvio della ricerca

- **Attributi:**

- + controller: String: variabile contenente il nome del controller del template;
- + \$scope: \$scope: oggetto scope interno del template, contiene le funzionalità per gestire i dati presenti all'interno;
- + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
- + query: String: stringa contenente la query da sottomettere al sistema

3 Package Model



3.1 Model::Database

3.1.1 Model::Database::QuizManager

- **Funzione del componente:** la classe permetterà l'inserimento, la lettura e la rimozione di questionari all'interno della collezione
- **Relazioni d'uso con altre componenti:** ViewModel::Methods::QuestionMethods
- **Metodi:**

– + addQuiz() : Metodo per aggiungere un nuovo quiz al database

Parametri:

* ID_utente: int

* Titolo_quiz: string

* Tempo: int

* Domande: Array

* Categorie: Array

Precondizioni: Vengono richiesti i dati necessari per la creazione e il salvataggio di un quiz

Postcondizioni: Il quiz è stato salvato correttamente nel database

– + modifyQuiz() : Metodo per modificare un quiz nel database

Parametri:

* ID_utente: int

```
* ID_quiz:  int
```

```
* Titolo_quiz:  string
```

```
* Tempo:  int
```

```
* Domande:  Array
```

```
* Categorie:  Array
```

Precondizioni: Vengono richiesti i dati necessari per la modifica di un quiz

Postcondizioni: Il quiz e' stato modificato nel database

– + removeQuiz() : Metodo per rimuovere un quiz dal database

Parametri:

```
* ID_utente:  int
```

```
* ID_quiz:  int
```

Precondizioni: Vengono richiesti i dati necessari per la rimozione di un quiz

Postcondizioni: Il quiz e' stato rimosso dal database

3.1.2 Model::Database::QuestionManager

- **Funzione del componente:** la classe permettera' l'inserimento, la lettura e la rimozione di singoli quesiti all'interno della collezione

- **Relazioni d'uso con altre componenti:** ViewModel::Methods::QuizMethods

- **Metodi:**

– + addQuestion() : Metodo per aggiungere un nuovo quesito al database

Parametri:

```
* ID_utente:  int
```

```
* testo_QML:  string
```

```
* categoria:  string
```

Precondizioni: Vengono richiesti i dati necessari per la creazione e il salvataggio di un nuovo quesito

Postcondizioni: Il quesito e' stato salvato correttamente nel database se viene rispettata la sintassi QML altrimenti viene restituito un messaggio d'errore

– + modifyQuestion() : Metodo per modificare un quesito nel database

Parametri:

```
* ID_utente:  int
```

```
* ID_quesito:  int
```

* testo_QML: string

* categoria: string

Precondizioni: Vengono richiesti i dati necessari per la modifica di un quesito

Postcondizioni: Il quesito e' stato modificato nel database se viene rispettata la sintassi QML altrimenti viene restituito un messaggio d'errore

– + removeQuestion() : Metodo per rimuovere un quesito dal database

Parametri:

* ID_utente: int

* ID_quesito: int

Precondizioni: Vengono richiesti i dati necessari per la rimozione di un quesito

Postcondizioni: Il quesito e' stato rimosso dal database

3.2 Model::Parser

3.2.1 Model::Parser::Parser

- **Funzione del componente:** controlla che il testo fornito risulti corretto secondo la sintassi QML
- **Relazioni d'uso con altre componenti:** QuestionManager

- **Metodi:**

– + check() : Metodo per controllare che un nuovo quesito rispetti la sintassi QML prima di essere aggiunto al database

Parametri:

* testo_da_controllare: string

Precondizioni: Viene richiesta una string che contiene la domanda

Postcondizioni: Viene restituito un messaggio contenente il risultato dell'operazione di controllo sintattico

3.3 Model::Statistics

3.3.1 Model::Statistics::Statistics

- **Funzione del componente:** questa classe fornisce funzionalita' per il raccoglimento delle statistiche sulle prestazioni degli utenti del sistema
- **Relazioni d'uso con altre componenti:** Nessuna

- **Metodi:**

– + updateQuestionStatistics() : Metodo per aggiornare le statistiche di un quesito nel database

Parametri:

- * ID_quesito: int
- * Corretta: bool
 - Precondizioni:** Vengono richiesti i risultati conseguiti dall'utente per lo specifico quesito
 - Postcondizioni:** Le statistiche relative al quesito risultano aggiornate
- + updateQuizStatistics() : Metodo per aggiornare le statistiche di un quiz nel database
 - Parametri:**
 - * ID_quiz: int
 - * ID_utente: int
 - * Risposte: Array
 - Precondizioni:** Vengono richiesti i risultati conseguiti dall'utente per lo specifico quiz
 - Postcondizioni:** Viene salvato un record relativo ad una singola esecuzione del quiz
- + updateUserStatistics() : Metodo per aggiornare le statistiche di un utente nel database
 - Parametri:**
 - * ID_utente: int
 - * quesiti_corretti: int
 - * quesiti_totali_quiz: int
 - Precondizioni:** Vengono richiesti i risultati conseguiti dall'utente in un quiz
 - Postcondizioni:** Viene aggiornate le statistiche globali dell'utente

3.4 Model::Publishers

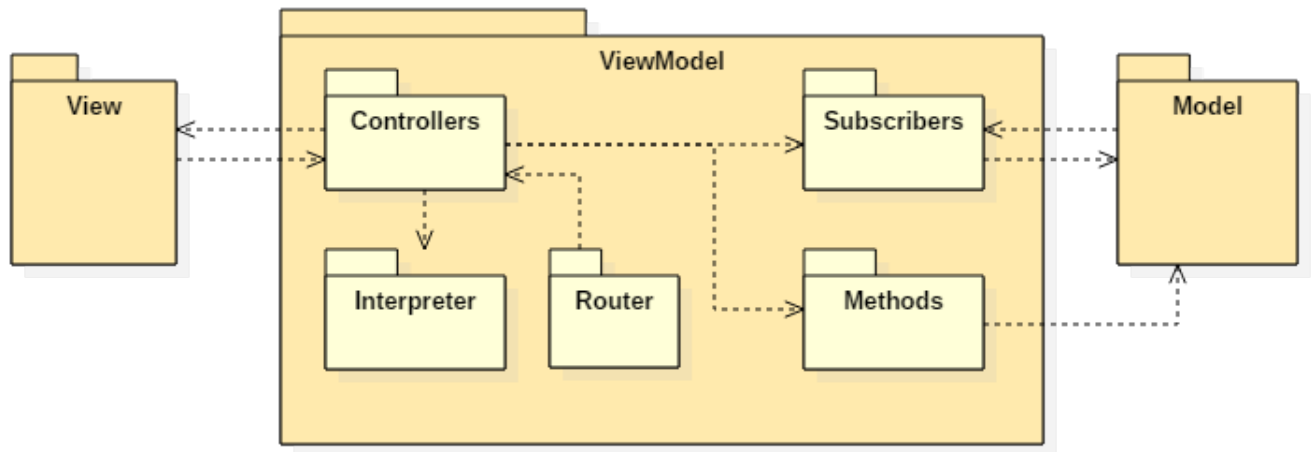
3.4.1 Model::Publishers::QuizPublisher

- **Funzione del componente:** questa classe fornisce funzionalità per rendere pubblica la collezione dei quiz all'avvio del Sistema
- **Relazioni d'uso con altre componenti:** Nessuna
- **Metodi:**
 - + publishQuiz() : Metodo per rendere accessibili la collezione dei quiz all'avvio del Sistema
 - Precondizioni:** Il Sistema non ha ancora reso pubblica la collezione dei quiz
 - Postcondizioni:** Il Sistema ha reso pubblica la collezione dei quiz

3.4.2 Model::Publishers::QuestionPublisher

- **Funzione del componente:** questa classe fornisce funzionalità per rendere pubblica la collezione dei quesiti all'avvio del Sistema
 - **Relazioni d'uso con altre componenti:** Nessuna
 - **Metodi:**
 - + `publishQuestion()` : Metodo per rendere accessibili la collezione dei quesiti all'avvio del Sistema
- Precondizioni:** Il Sistema non ha ancora reso pubblica la collezione dei quesiti
Postcondizioni: Il Sistema ha reso pubblica la collezione dei quesiti

4 Package ViewModel



4.1 ViewModel::Controllers

4.1.1 ViewModel::Controllers::NewQuestionController

- **Funzione del componente:** la classe permette di gestire la creazione di una nuova domanda
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - ViewModel::Methods::QuestionMethods
 - View::Templates::QuestionForm
- **Attributi:**
 - \$scope: Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
- **Metodi:**
 - + NewQuestionController(): metodo costruttore della classe
Parametri:
 - * \$scope: campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
 - + saveQuestion(): metodo che permette il salvataggio di una nuova domanda tramite la chiamata ad un Method.
Precondizioni: viene richiesto di salvare una nuova domanda
Postcondizioni: la nuova domanda è stata salvata nel sistema
Parametri:

- * `question`: `String`: il testo QML della domanda da salvare.
- * `category`: `String`: la categoria della domanda da salvare.
- + `check()`: metodo che invoca un `Method` per effettuare il check del testo QML inserito
Precondizioni: l'utente clicca sul bottone "Check QML"
Postcondizioni: viene effettuato il parsing del testo e l'esito viene comunicato all'utente
Parametri:
 - * `QMLtext`: `String`: il testo QML inserito del quale viene richiesto il parsing.

4.1.2 `ViewModel::Controllers::NewQuizController`

- **Funzione del componente**: la classe permette di gestire la creazione di un nuovo quiz (questionario);
- **Relazioni con altre componenti**:
La classe utilizza:
 - `ViewModel::Methods::QuizMethods`
 - `View::Templates::QuizCreationForm`
- **Attributi**:
 - `-$scope`: Campo dati contenente un riferimento all'oggetto `$scope` creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
 - + `questions`: `Array`: variabile che contiene gli id delle domande aggiunte al questionario.
 - + `categories`: `Array[category: String, counter: int]`: variabile che contiene la lista delle categorie delle domande aggiunte al questionario. Ad ogni categoria aggiunta viene associato il numero di domande del questionario appartenenti alla categoria.
- **Metodi**:
 - + `NewQuizController()`: metodo costruttore della classe
Parametri:
 - * `$scope`: campo dati contenente un riferimento all'oggetto `$scope` creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
 - + `saveQuiz()`: metodo per richiedere il salvataggio del nuovo quiz
Precondizioni: viene richiesto di salvare un nuovo quiz
Postcondizioni: il nuovo quiz è stato salvato nel sistema
Parametri:
 - * `title`: `String`: il titolo del questionario.

- * **questions:** Array: la lista di id delle domande che compongono il questionario.
 - * **categories:** Array: la lista delle categorie associate al questionario.
 - * **time:** Int: il tempo massimo in minuti per la compilazione del questionario.
- + **toggleSelection():** metodo che aggiunge la categoria della domanda scelta alle categorie del questionario. Se la categoria era già presente il contatore associato viene incrementato
- Precondizioni:** l'utente sceglie di aggiungere o di togliere una domanda al quiz in creazione, selezionando il rispettivo check-box
- Postcondizioni:** la categoria della domanda appena scelta viene aggiunta alle categorie del quiz e mostrata all'utente. Se la categoria era già presente il contatore associato viene incrementato
- Parametri:**
- * **question:** Question: la variabile che rappresenta la domanda selezionata

4.1.3 ViewModel::Controllers::QuizListController

- **Funzione del componente:** la classe permette il caricamento e la visualizzazione della lista di quiz disponibili nel sistema
- **Relazione d'uso con altre componenti:** La classe utilizza:
 - View::Templates::QuizList
- **Attributi:**
 - **\$scope:** Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
- **Metodi:**
 - + **QuizListController():** metodo costruttore della classe

Parametri:

 - * **\$scope:** campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.- + **quizzes():** metodo che ritorna la lista dei quiz del sistema.

Precondizioni: viene richiesta la collezione di quiz del sistema

Postcondizioni: la lista di quiz è disponibile
- + **orderBy():** metodo che permette l'ordinamento della lista di quiz.

Precondizioni: viene richiesto di ordinare la lista di quiz secondo un parametro

Postcondizioni: la lista è stata riordinata

Parametri:

 - * **orderBy:** String: il parametro in base al quale riordinare la lista di quiz.

4.1.4 ViewModel::Controllers::QuizDetailsController

- **Funzione del componente:** la classe permette di visualizzare le informazioni generali di un quiz, una volta selezionato dalla lista dei quiz;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - View::Templates::QuizList
- **Attributi:**
 - - \$scope: Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
- **Metodi:**
 - + QuizDetailsController(): metodo costruttore della classe
Parametri:
 - * \$scope: campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
 - + quizDetails(): metodo che permette la visualizzazione dei dettagli di un quiz.
Precondizioni: vengono richiesti i dettagli di un quiz
Postcondizioni: i dettagli del quiz vengono visualizzati
Parametri:
 - * quizID: String: identificativo univoco del quiz da visualizzare.

4.1.5 ViewModel::Controllers::DeleteQuestionController

- **Funzione del componente:** la classe fornisce le funzionalità necessarie alla cancellazione di una domanda precedentemente creata;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - ViewModel::Methods::QuestionMethods
 - View::Templates::QuestionList
- **Attributi:**
 - - \$scope: Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
- **Metodi:**
 - + DeleteQuestionController(): metodo costruttore della classe
Parametri:

- * `$scope`: campo dati contenente un riferimento all'oggetto `$scope` creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
- + `deleteQuestion()`: metodo per richiedere l'eliminazione di una domanda
Precondizioni: viene richiesta l'eliminazione di una domanda dal sistema
Postcondizioni: la domanda è stata eliminata dal sistema
Parametri:
 - * `question`: `String`: identificativo univoco della domanda da eliminare

4.1.6 ViewModel::Controllers::DeleteQuizController

- **Funzione del componente**: la classe fornisce le funzionalità necessarie alla cancellazione di un quiz precedentemente creato;
- **Relazione d'uso con altre componenti**:
La classe utilizza:
 - ViewModel::Methods::QuizMethods
 - View::Templates::QuizList
- **Attributi**:
 - - `$scope`: Campo dati contenente un riferimento all'oggetto `$scope` creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
- **Metodi**:
 - + `DeleteQuizController()`: metodo costruttore della classe
Parametri:
 - * `$scope`: campo dati contenente un riferimento all'oggetto `$scope` creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.
 - + `deleteQuiz()`: metodo per richiedere l'eliminazione di un quiz
Precondizioni: viene richiesta l'eliminazione di un quiz dal sistema
Postcondizioni: il quiz è stato eliminato dal sistema
Parametri:
 - * `quiz`: `String`: identificativo univoco del quiz da eliminare

4.1.7 ViewModel::Controllers::QuizManagementController

- **Funzione del componente**: la classe permette di gestire la somministrazione di un questionario;
- **Relazione d'uso con altre componenti**:
La classe utilizza:
 - View::Templates::QuestionCompilation

- **Attributi:**

- \$scope: Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.

- **Metodi:**

- +QuizManagementController(): metodo costruttore della classe

Parametri:

- * \$scope: campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.

- + nextQuestion(): funzione che ritorna la domanda successiva nel questionario

Precondizioni: viene richiesta la domanda successiva nel questionario

Postcondizioni: viene visualizzata la domanda successiva nel questionario

Parametri:

- * question: String: identificativo univoco della domanda

- + previousQuestion(): funzione che ritorna la domanda precedente nel questionario

Precondizioni: viene richiesta la domanda precedente nel questionario

Postcondizioni: viene visualizzata la domanda precedente nel questionario

Parametri:

- * question: String: identificativo univoco della domanda

- + endQuiz(): metodo per la consegna del quiz in compilazione

Precondizioni: viene richiesto di consegnare il quiz compilato

Postcondizioni: il quiz compilato viene consegnato e ne vengono visualizzati i risultati

4.1.8 ViewModel::Controllers::QuestionsManagementController

- **Funzione del componente:** la classe permette di gestire la somministrazione di una singola domanda all'interno di un questionario;

- **Relazione d'uso con altre componenti:**

La classe utilizza:

- View::Templates::QuestionCompilation
- ViewModel::Methods::QuestionMethods

- **Attributi:**

- \$scope: Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.

- **Metodi:**

- + QuestionsManagementController(): metodo costruttore della classe

Parametri:

- * \$scope: campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.

- + getQuestion(): metodo per caricare una domanda specifica

Precondizioni: viene richiesta una domanda del sistema

Postcondizioni: la domanda è stata caricata e viene visualizzata

Parametri:

- * question: String: identificativo univoco della domanda

4.1.9 ViewModel::Controllers::QMLEditorController

- **Funzione del componente:** fornisce le funzionalità per creare e modificare una domanda tramite editor QML;

- **Relazione d'uso con altre componenti:**

La classe utilizza:

- ViewModel::Controllers::NewQuestionController
- ViewModel::Methods::QuestionMethods

- **Attributi:**

- - \$scope: Campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.

- **Metodi:**

- + QMLEditorController(): metodo costruttore della classe

Parametri:

- * \$scope: campo dati contenente un riferimento all'oggetto \$scope creato da Angular, viene utilizzato come mezzo di comunicazione tra il controller e il template.

- check(): metodo che controlla che il testo QML inserito sia sintatticamente corretto

Precondizioni: viene richiesto di controllare la validità del testo QML inserito

Postcondizioni: il testo è stato validato e viene visualizzato l'esito della validazione

Parametri:

- * text: String: il testo QML inserito

4.2 ViewModel::Subscribers

4.2.1 ViewModel::Subscribers::QuestionsSubscriber

- **Funzione del componente:** la classe è necessaria ad effettuare il *subscribe* relativo alla collezione di domande del sistema;

- **Relazione d'uso con altre componenti:**

La classe utilizza:

- Model::Publishers::QuestionsPublisher

- **Attributi:**

- - collection: String: il nome della collezione sulla quale effettuare il subscribe.

- **Metodi:**

- + getCollection(): String: ritorna la collezione sulla quale può effettuare il subscribe.

Precondizioni: viene richiesta la collezione di pertinenza della classe

Postcondizioni: viene ritornato il nome della collezione

- + subscribe(): metodo che effettua il subscribe del parametro controller alla collezione

Precondizioni: viene richiesto di effettuare il subscribe del parametro controller alla collezione

Postcondizioni: è stato effettuato il subscribe del controller alla collezione

Parametri:

- * controller: Object: il controller che necessita della collezione

4.2.2 ViewModel::Subscribers::QuizSubscriber

- **Funzione del componente:** la classe è necessaria ad effettuare il *subscribe* relativo alla collezione di quiz del sistema;

- **Relazione d'uso con altre componenti:**

La classe utilizza:

- Model::Publishers::QuizPublisher

- **Attributi:**

- - collection: String: il nome della collezione sulla quale effettuare il subscribe.

- **Metodi:**

- + getCollection(): String: ritorna la collezione sulla quale può effettuare il subscribe.

Precondizioni: viene richiesta la collezione di pertinenza della classe

Postcondizioni: viene ritornato il nome della collezione

- + subscribe(): metodo che effettua il subscribe del parametro controller alla **Precondizioni:** viene richiesto di effettuare il subscribe del parametro controller alla collezione

Postcondizioni: è stato effettuato il subscribe del controller alla collezione

Parametri:

* controller: Object: il controller che necessita della collezione

4.2.3 ViewModel::Subscribers::UsersSubscriber

- **Funzione del componente:** la classe è necessaria ad effettuare il *subscribe* relativo alla collezione di utenti del sistema;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - Model::Publishers::UserPublisher
- **Attributi:**
 - - collection: String: il nome della collezione sulla quale effettuare il subscribe.
- **Metodi:**
 - + getCollection(): String: ritorna la collezione sulla quale può effettuare il subscribe.
Precondizioni: viene richiesta la collezione di pertinenza della classe
Postcondizioni: viene ritornato il nome della collezione
 - + subscribe(): metodo che effettua il subscribe del parametro controller alla collezione
Precondizioni: viene richiesto di effettuare il subscribe del parametro controller alla collezione
Postcondizioni: è stato effettuato il subscribe del controller alla collezione
Parametri:
 - * controller: Object: il controller che necessita della collezione

4.3 ViewModel::Methods

4.3.1 ViewModel::Methods::QuestionMethods

- **Funzione del componente:** permette al client di richiedere la modifica della collezione di domande del server;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - Model::Publishers::QuestionPublisher
- **Attributi:**
 - - collection: String: la collezione sulla quale verranno implementati i methods
- **Metodi:**

- + insert(): metodo che inserisce una domanda nella collezione
Precondizioni: viene richiesto di inserire una nuova domanda nella collezione
Postcondizioni: la nuova domanda è stata salvata nel sistema
Parametri:
 - * questionId: String: identificativo univoco della domanda da aggiungere
- + remove(): metodo che rimuove una domanda dalla collezione
Precondizioni: viene richiesta l'eliminazione di una domanda dal sistema
Postcondizioni: la domanda è stata eliminata dal sistema
Parametri:
 - * questionId: String: identificativo univoco della domanda da rimuovere

4.3.2 ViewModel::Methods::QuizMethods

- **Funzione del componente:** permette al client di richiedere la modifica della collezione di quiz del server;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - Model::Publishers::QuizPublisher
- **Attributi:**
 - - collection: String: la collezione sulla quale verranno implementati i methods
- **Metodi:**
 - + insert(): metodo che inserisce un quiz nella collezione
Precondizioni: viene richiesto di inserire un nuovo quiz nella collezione
Postcondizioni: il nuovo quiz è stato salvato nel sistema
Parametri:
 - * quizId: String: identificativo univoco del quiz da aggiungere
 - + remove(): metodo che rimuove un quiz dalla collezione
Precondizioni: viene richiesta l'eliminazione di un quiz dal sistema
Postcondizioni: il quiz è stato eliminato dal sistema
Parametri:
 - * quizId: String: identificativo univoco della domanda da rimuovere

4.3.3 ViewModel::Methods::UserMethods

- **Funzione del componente:** permette al client di richiedere la modifica della collezione di utenti del server;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - accounts-ui

- **Attributi:**

- - collection: String: la collezione sulla quale verranno implementati i methods

- **Metodi:**

- + insert(): metodo che inserisce un utente nella collezione
Precondizioni: viene richiesto di inserire un nuovo utente nella collezione
Postcondizioni: il nuovo utente è stato salvato nel sistema
Parametri:
 - * userId: String: identificativo univoco dell'utente da aggiungere
 - + remove(): metodo che rimuove un utente dalla collezione
Precondizioni: viene richiesta l'eliminazione di un utente dal sistema
Postcondizioni: l'utente è stato eliminato dal sistema
Parametri:
 - * userId: String: identificativo univoco dell'utente da rimuovere

4.4 ViewModel::Interpreter

4.4.1 ViewModel::Interpreter::Interpreter

- **Funzione del componente:** interfaccia di base del tipo Interpreter.
- **Relazione d'uso con altre componenti:** viene concretizzata in ViewModel::Interpreter::QMLInterpreter
- **Attributi:** nessuno
- **Metodi:**
 - + translate(): String: metodo astratto per la traduzione di un testo
Precondizioni: viene richiesta la traduzione di un testo
Postcondizioni: il testo è stato tradotto in un altro linguaggio
Parametri:
 - * data: String: l'input testuale che deve essere tradotto

4.4.2 ViewModel::Interpreter::InterpreterFactory

- **Funzione del componente:** interfaccia di base delle Factory di tipi Interpreter.
- **Relazione d'uso con altre componenti:** viene concretizzata da ViewModel::Interpreter::QMLInterpreter
- **Attributi:** nessuno
- **Metodi:**

- + createInterpreter(): Interpreter: metodo astratto per la creazione di un Interpreter.

Precondizioni: viene richiesta l'istanziatura di un Interpreter

Postcondizioni: il nuovo Interpreter viene creato e ritornato al chiamante

4.4.3 ViewModel::Interpreter::QMLInterpreterFactory

- **Funzione del componente:** crea oggetti di tipo QMLInterpreter.
- **Relazione d'uso con altre componenti:** è concretizzazione dalla classe ViewModel::Interpreter::Interpreter. Crea oggetti QMLInterpreter.
- **Attributi:**
 - - instance: QMLInterpreterFactory: campo dati statico che rappresenta l'unica istanza della classe
- **Metodi:**
 - - QMLInterpreterFactory(): costruttore privato della classe
 - + getInstance(): : QMLInterpreterFactory: metodo che ritorna l'unica istanza della classe
 - Precondizioni:** viene richiesta la factory QMLInterpreterFactory
 - Postcondizioni:** l'unica istanza della factory viene ritornata al chiamante. Se la factory non esisteva è stata creata.
 - + createInterpreter(): QMLInterpreter: metodo ereditato da InterpreterFactorye concretizzato per la creazione di un interprete di tipo QMLInterpreter.
 - Precondizioni:** viene richiesta l'istanziatura di un QMLInterpreter
 - Postcondizioni:** il nuovo QMLInterpreter viene creato e ritornato al chiamante

4.4.4 ViewModel::Interpreter::QMLInterpreter

- **Funzione del componente:** classe astratta che rappresenta gli Interpreter che traducono codice QML in un altro formato.
- **Relazione d'uso con altre componenti:** è sottotipo di ViewModel::Interpreter::Interpreter. Viene concretizzata in ViewModel::Interpreter::QML2HTMLInterpreter.
- **Attributi:** Nessuno
- **Metodi:**
 - translate(): String: metodo astratto ereditato da Interpreter
 - Precondizioni:** viene richiesta la traduzione di un testo QML
 - Postcondizioni:** il testo QML è stato tradotto in un altro linguaggio
 - Parametri:**

- * data: String: l'input QML che deve essere tradotto

4.4.5 ViewModel::Interpreter::QML2HTMLInterpreter

- **Funzione del componente:** traduce codice QML in codice HTML.
- **Relazione d'uso con altre componenti:** la classe è concretizzazione di ViewModel::Interpreter::QMLInterpreter
- **Attributi:** Nessuno
- **Metodi:**
 - translate(): String: metodo ereditato da QMLInterpreter e concretizzato per la traduzione di testo QML in testo HTML.
Precondizioni: viene richiesta la traduzione di un testo QML in HTML
Postcondizioni: il testo QML è stato tradotto in HTML
Parametri:
 - * data: String: l'input QML che deve essere tradotto

4.5 ViewModel::Router

4.5.1 ViewModel::Router::Router

- **Funzione del componente:** implementa il routing dinamico dell'applicazione. Permette di dividere la parte statica dell'applicazione dalla parte che va caricata dinamicamente;
- **Relazione d'uso con altre componenti:**
La classe utilizza:
 - ui-router
- **Attributi:**
 - \$locationProvider: \$locationProvider: provider AngularJs per rendere gli Url più leggibili
 - \$urlRouterProvider: \$urlRouterProvider: modulo del package ui-router per definire il routing di default dell'applicazione
 - \$stateProvider: \$stateProvider: modulo del package ui-router per mappare gli Url al caricamento dinamico dei template dell'applicazione. Tale associazione è definita come uno 'state'
- **Metodi:**
 - state(): metodo per collegare un Url dell'applicazione al rispettivo template da caricare
Precondizioni: viene richiesta l'associazione di un Url al caricamento di un template
Postcondizioni: un Url univoco è stato associato al caricamento dinamico di uno

specifico template

Parametri:

- * state: String: il nome dello stato rappresentato dall'associazione Url-Template
- * url: String: l'Url sul quale effettuare il matching
- * template: String: il template da caricare una volta trovato il match

5 Tracciamento

5.1 Mappatura classi – requisiti

Tabella 2: Mappatura delle classi sui requisiti

Nome classe	Codice requisito
Model::Database	F 1.2.2 F 3 F 3.1.1 F 3.2 F 6.2.2
Model::Database::QuestionManager::AddQuestion	F 5 F 5.1 F 5.2
Model::Database::QuestionManager::ModifyQuestion	F 5.3
Model::Database::QuestionManager::RemoveQuestion	F 5.4
Model::Database::QuizManager::AddQuiz	F 6 F 6.1 F 6.2
Model::Database::QuizManager::RemoveQuiz	Non tracciabile
Model::Database::UserManager::LogIn	F 2 F 2.2
Model::Database::UserManager::AddUser	F 1 F 2.3.2
Model::Database::UserManager::RemoveUser	Non tracciabile
Model::Parser::Parser	F 5.2.1 F 8
Model::Statistics::Statistics	4.5.2 4.5.2.1 4.5.2.1.1 4.5.2.1.2 4.5.2.1.3
Model::Publishers::UserPublishers	Non tracciabile
Model::Publishers::QuizPublishers	Non tracciabile
Model::Publishers::QuestionPublishers	Non tracciabile
View::Pages::RegistrationPage	F 1 F 1.1 F 1.1.1 F 1.1.2 F 1.1.2.1 F 1.2 F 1.2.1.1

Tabella 2: Mappatura delle classi sui requisiti

Nome classe	Codice requisito
	F 1.2.1.2 F 1.2.3
View::Pages::LoginPage	F 2 F 2.1 F 2.1.1 F 2.1.2 F 2.3 F 2.3.1 F 2.3.1.1 F 2.3.2
View::Pages::CategoryListPage	F 3.1 F 3.1.1 F 3.1.2 F 3.1.3
View::Pages::QuizListPage	F 3 F 3.1 F 3.1.1 F 3.1.2 F 3.1.3 F 3.2 F 3.2.1 F 3.2.2 F 3.2.3
View::Pages::QuizExecutionPage	F 4 F 4.1 F 4.2.1 F 4.2.2 F 4.2.3 F 4.2.4 F 4.2.4.1 F 4.2.5 F 4.2.5.1 F 4.2.6 F 4.2.6.1 F 4.2 F 4.3 F 4.4 F 4.5 F 4.5.1 F 4.5.2.1 F 4.5.2.1 F 4.5.2.1.1 F 4.5.2.1.2 F 4.5.2.1.3 F 4.6.2 F 4.7

Tabella 2: Mappatura delle classi sui requisiti

Nome classe	Codice requisito
View::Pages::PasswordRecoveryPage	F 2.2 F 2.2.1 F 2.2.2 F 2.2.2.1 F 2.2.2.2 F 2.2.3
View::Pages::QuizResultPage	4.5.2 4.5.2.1 4.5.2.1.1 4.5.2.1.2 4.5.2.1.3
View::Pages::QuizManagementPage	F 6
View::Pages::ViewTutorialPage	F 8.3
View::Pages::QuizCreationPage	F 6
View::Pages::QuestionCreationPage	F 5
View::Pages::QuestionUpdatePage	F 5.3
View::Templates::QuestionList	F 4.1.1
View::Templates::Question	F 4.2
View::Templates::QuizList:	F 3
View::Templates::Quiz	F 4
View::Templates::QuestionForm	F 5
View::Templates::QuizCreationForm	F 6
View::Templates::QuestionCompilation	F 4
View::Templates::QuizResults	F 4.5.2
View::Templates::RegistrationForm	F 1
View::Templates::LoginForm	F 2
View::Templates::PasswordRecoveryForm	F 2.2
View::Templates::SearchForm	F 3.3
ViewModel::Controllers::NewQuestionController	F 5
ViewModel::Controllers::NewQuiz Controller	F 6
ViewModel::Controllers::QuestionManagementController	F 5.3
ViewModel::Controllers::DeleteQuestionController	F 5.4
ViewModel::Controllers::DeleteQuiz Controller	Non tracciabile
ViewModel::Controllers::QuizManagementController	F 6
ViewModel::Controllers::QuizListController	Non tracciabile
ViewModel::Controllers::QMLEditorController	F 5.2.1

Tabella 2: Mappatura delle classi sui requisiti

Nome classe	Codice requisito
	F 8
ViewModel::Controllers::QuizDetailsController	Non tracciabile
ViewModel::Subscribers::QuestionSubscriber	Non tracciabile
ViewModel::Subscribers::QuizSubscriber	Non tracciabile
ViewModel::Subscribers::UserSubscriber	Non tracciabile
ViewModel::Methods::QuestionMethods	Non tracciabile
ViewModel::Methods::QuizMethods	Non tracciabile
ViewModel::Methods::UserMethods	Non tracciabile
ViewModel::Interpreter::QMLInterpreterFactory	F 8
ViewModel::Interpreter::QMLInterpreter	F 8.1 F 8.2
ViewModel::Interpreter::QML2HTMLInterpreter	F 8 F 8.1 F 8.2
ViewModel::Router	non tracciabile

Tabella 2: Mappatura delle classi sui requisiti

5.2 Mappatura requisiti – classi

Tabella 3: Mappatura dei requisiti sulle classi

Codice Requisito	Nome Classe
F 1	View::Pages::RegistrationPage Model::Database::UserManager::AddUser View::Templates::RegistrationForm
F 2	Model::Database::UserManager::Login View::Pages::LoginPage View::Pages::PasswordRecoveryPage View::Templates::LoginForm View::Templates::PasswordRecoveryForm
F 3	Model::Database View::Pages::CategoryListPage View::Pages::QuizListPage View::Templates::QuizList: View::Templates::SearchForm
F 4	View::Pages::QuizExecutionPage View::Pages::QuizResultPage Model::Statistics::Statistics View::Templates::QuestionList View::Templates::Question View::Templates::Quiz View::Templates::QuestionCompilation View::Templates::QuizResults
F 5	Model::Database::QuestionManager::AddQuestion Model::Database::QuestionManager::ModifyQuestion Model::Database::QuestionManager::RemoveQuestion Model::Parser::Parser View::Pages::QuestionCreationPage View::Pages::QuestionUpdatePage View::Templates::QuestionForm ViewModel::Controllers::NewQuestionController ViewModel::Controllers::QuestionManagementController ViewModel::Controllers::DeleteQuestionController ViewModel::Controllers::QMLEditorController
F 6	Model::Database::QuizManager::AddQuiz View::Pages::QuizManagementPage View::Pages::QuizCreationPage View::Templates::QuizCreationForm ViewModel::Controllers::NewQuizController ViewModel::Controllers::QuizManagementController
F 8	View::Pages::ViewTutorialPage ViewModel::Controllers::QMLEditorController ViewModel::Interpreter::QMLInterpreterFactory

Tabella 3: Mappatura dei requisiti sulle classi

Codice Requisito	Nome Classe
	ViewModel::Interpreter::QMLInterpreter
	ViewModel::Interpreter::QML2HTMLInterpreter

Tabella 3: Mappatura dei requisiti sulle classi