

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	17 agosto 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	
1.0.10	Marco Crivellaro (Progettista)	10/08/2016	Revisione integrale ViewModel::Interpreter e Model::Parser
1.0.9	Marco Crivellaro (Progettista)	06/08/2016	Caricamento immagini sezione ViewModel
1.0.8	Luca Alessio (Verificatore)	03/08/2016	Terminate sezioni ViewModel::Controller, ViewModel::Methods e ViewModel::Router
1.0.7	Luca Alessio (Verificatore)	01/08/2016	Stesura sezione ViewModel::Subscribers e Model::Publishers
1.0.6	Marco Crivellaro (Progettista)	28/07/2016	Inserimento immagini sezione Model
1.0.5	Luca Alessio (Verificatore)	27/07/2016	Stesura sezione ViewModel::Controller
1.0.4	Luca Alessio (Verificatore)	26/07/2016	Ristesura sezione View::Templates e rimozione sezione View::Pages
1.0.3	Luca Alessio (Verificatore)	24/07/2016	Inizio revisione sezione View (aggiunta sottosezioni NavBarController e TopBarController)
1.0.2	Luca Alessio (Verificatore)	22/07/2016	Ampliamento sezione Model (ristesura di tutti i metodi delle sezioni Database, Statistics e Publishers)
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::Quizzipedia	3
2.2	View::Templates	4
2.2.1	View::Templates::quizHome	4
2.2.2	View::Templates::quizList	4
2.2.3	View::Templates::quizResults	4
2.2.4	View::Templates::quizStatistics	4
2.2.5	View::Templates::searchForm	4
2.2.6	View::Templates::topbar	4
2.2.7	View::Templates::userProfile	5
2.2.8	View::Templates::navbar	5
2.2.9	View::Templates::question	5
2.2.10	View::Templates::questionForm	5
2.2.11	View::Templates::questionList	5
2.2.12	View::Templates::quiz	5
2.2.13	View::Templates::quizCompilation	6
2.2.14	View::Templates::quizCreationForm	6
3	Package Model	7
3.1	Model::Database	7
3.1.1	Model::Database::QuizManager	7
3.1.2	Model::Database::QuestionManager	9
3.1.3	Model::Database::Search	10
3.2	Model::Parser	11
3.2.1	Model::Parser::Parser	11
3.2.2	La sintassi QML	13
3.2.3	Esempi di domande secondo la sintassi corretta	15
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::Subscribers	19
4.1.1	ViewModel::Subscribers::QuestionsSubscriber	19
4.1.2	ViewModel::Subscribers::QuizSubscriber	21
4.2	ViewModel::Methods	22
4.2.1	ViewModel::Methods::QuestionMethods	22

4.2.2	ViewModel::Methods::QuizMethods	23
4.2.3	ViewModel::Methods::UserMethods	24
4.3	ViewModel::Interpreter	25
4.3.1	ViewModel::Interpreter::Interpreter	25
4.4	ViewModel::Router	26
4.4.1	ViewModel::Router::Router	26
4.5	ViewModel::Controller	27
4.5.1	ViewModel::Controller::QuestionList	27
4.5.2	ViewModel::Controller::Question	28
4.5.3	ViewModel::Controller::QuizList	29
4.5.4	ViewModel::Controller::Quiz	30
4.5.5	ViewModel::Controller::QuestionForm	31
4.5.6	ViewModel::Controller::QuizCreationForm	32
4.5.7	ViewModel::Controller::QuestionCompilation	33
4.5.8	ViewModel::Controller::QuizResults	34
4.5.9	ViewModel::Controller::SearchForm	35
4.5.10	ViewModel::Controller::NavBar	36
4.5.11	ViewModel::Controller::TopBar	37
4.5.12	ViewModel::Controller::UserProfile	38
4.5.13	ViewModel::Controller::QuizHome	39
4.5.14	ViewModel::Controller::QuizStatistics	40
4.5.15	ViewModel::Controller::QuizCompilation	41
5	Tracciamento	42
5.1	Mappatura classi - requisiti	42
5.2	Mappatura requisiti - classi	46

Elenco delle figure

Elenco delle tabelle

2	Mappatura delle classi sui requisiti	42
3	Mappatura dei requisiti sulle classi	46

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

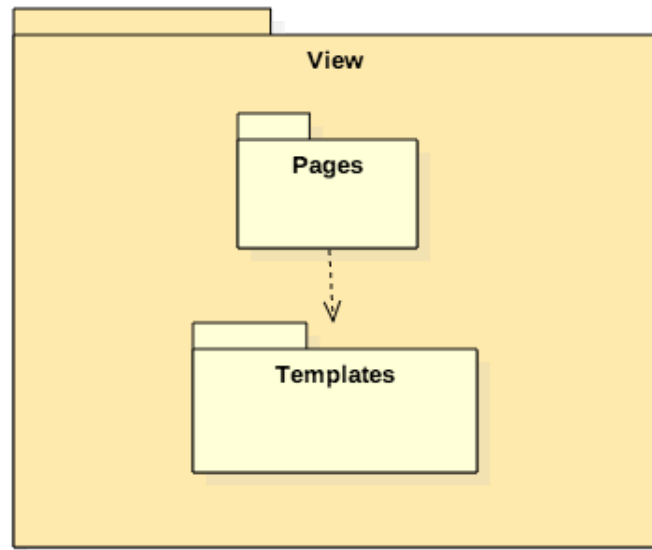
1.4.1 Normativi

- 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::Quizzipedia

- **Funzione del componente:** wrapper degli elementi comuni a tutte le pagine
- **Relazioni d'uso con altre componenti:**
 - View::Templates::quizHome
 - View::Templates::quizHome
 - View::Templates::quizList
 - View::Templates::quizResults
 - View::Templates::quizStatistics
 - View::Templates::quizzipedia
 - View::Templates::searchForm
 - View::Templates::topbar
 - View::Templates::userProfile
 - View::Templates::navbar
 - View::Templates::question
 - View::Templates::questionForm
 - View::Templates::questionList
 - View::Templates::quiz
 - View::Templates::quizCompilation
 - View::Templates::quizCreationForm
 - ViewModel::Router::Router

2.2 View::Templates

2.2.1 View::Templates::quizHome

- **Funzione del componente:** genera la pagina di benvenuto
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuizHome

2.2.2 View::Templates::quizList

- **Funzione del componente:** permette visualizzazione e interazione di una lista di questionari
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuizList

2.2.3 View::Templates::quizResults

- **Funzione del componente:** visualizza i risultati ottenuti in seguito alla compilazione di un quiz
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuizResults

2.2.4 View::Templates::quizStatistics

- **Funzione del componente:** permette la visualizzazione delle statistiche generali di una domanda REQUISITO OPZIONALE (se è questo che fa)
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuizStatistics

2.2.5 View::Templates::searchForm

- **Funzione del componente:** visualizza una form per l'inserimento dei dati desiderati e l'avvio della ricerca
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::SearchForm

2.2.6 View::Templates::topbar

- **Funzione del componente:** Fornisce il menù orizzontale con cui l'utente può interfacciarsi con il sistema
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::TopBar

2.2.7 View::Templates::userProfile

- **Funzione del componente:** permette la visualizzazione e l'interazione dell'utente con il proprio profilo e le funzionalità da esso offerto
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::UserProfile

2.2.8 View::Templates::navbar

- **Funzione del componente:** Fornisce un menù laterale che facilita l'interazione dell'utente con l'applicazione
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::NavBar

2.2.9 View::Templates::question

- **Funzione del componente:** permette la visualizzazione di una singola domanda
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::Question

2.2.10 View::Templates::questionForm

- **Funzione del componente:** visualizza il form utilizzabile sia per la creazione che per la modifica di una domanda
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuestionForm

2.2.11 View::Templates::questionList

- **Funzione del componente:** visualizza una lista di domande
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuestionList

2.2.12 View::Templates::quiz

- **Funzione del componente:** visualizza un quiz inserito in una lista
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::Quiz

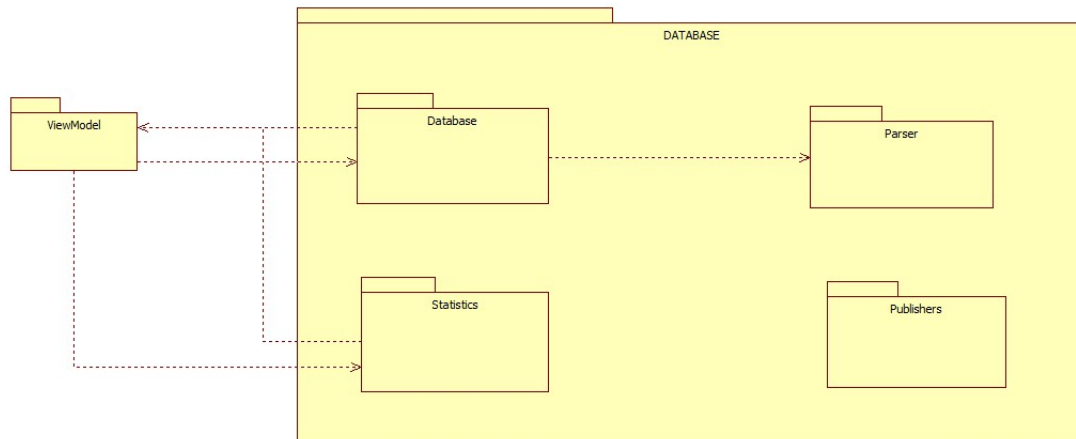
2.2.13 View::Templates::quizCompilation

- **Funzione del componente:**
- **Relazioni d'uso con altre componenti:** ViewModel::Controller::QuizCompilation

2.2.14 View::Templates::quizCreationForm

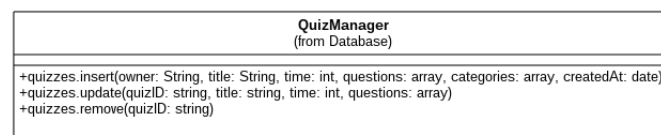
- **Funzione del componente:** visualizza il form di creazione di un questionario
- **Relazioni d'uso con altre componenti:** ViewModel::Controller:QuizCreationForm:

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:** Questo elemento è in relazione con:
 - Model::Publishers::QuizPublisher
 - Model::Statistics::Statistics
 - **Metodi:**
 - + quizzes.insert() : Metodo per aggiungere un nuovo quiz al database
- Parametri:**
- * owner: string
 - * title: string
 - * time: int

- * questions: array

- * categories: array

- * createdAt: date

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

Postcondizioni: Il quiz e' stato salvato correttamente nel database e crea una entry corrispondente nella collezione delle statistiche

– + quizzes.update() : Metodo per modificare un quiz nel database

Parametri:

- * quizID: string

- * title: string

- * time: int

- * questions: array

- * categories: array

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

Postcondizioni: Il quiz e' stato modificato nel database

– + quizzes.remove() : Metodo per rimuovere un quiz dal database

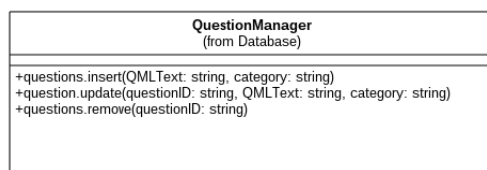
Parametri:

- * quizID: string

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

Postcondizioni: Il quiz e' stato rimosso dal database e le statistiche ad esso associate vengono anch'esse rimosse

3.1.2 Model::Database::QuestionManager



- **Funzione del componente:** la classe permetterà l'inserimento, la modifica e la rimozione di singoli quesiti all'interno della collezione
- **Relazioni d'uso con altre componenti:** Questo elemento è in relazione con:
 - Model::Publishers::QuizPublisher
 - Model::Statistics::Statistics
 - ViewModel::Interpreter::Interpreter
- **Metodi:**
 - + questions.insert() : Metodo per aggiungere un nuovo quesito al database
Parametri:
 - * QMLtext: string
 - * category: string**Precondizioni:** Vengono forniti i dati necessari per la creazione e il salvataggio di un nuovo quesito (il testo QML e la categoria vengono immessi dall'utente mentre data e utente corrente sono informazioni già presenti il sistema al momento della richiesta)
Postcondizioni: Il quesito è stato salvato correttamente nel database se viene rispettata la sintassi QML, altrimenti viene restituito un messaggio d'errore
 - + questions.update() : Metodo per modificare un quesito nel database
Parametri:
 - * questionID: string
 - * QMLtext: string
 - * category: string**Precondizioni:** Vengono forniti i dati necessari per la modifica di un intero quesito o di sue parti (es. solo alcune parti del testo o cambio di categoria)
Postcondizioni: Il quesito è stato modificato nel database se viene rispettata la sintassi QML, altrimenti viene restituito un messaggio d'errore

– + `questions.remove()` : Metodo per rimuovere un quesito dal database

Parametri:

* `questionID`: string

Precondizioni: L'utente richiede l'eliminazione di un quesito

Postcondizioni: Il quesito e' stato rimosso dal database

3.1.3 Model::Database::Search

Search (from Database)
+question.search(text: string) +quizzes.search(text: string)

- **Funzione del componente:** la classe permettera' la ricerca di sottostringhe in singoli quesiti o in diversi questionari
- **Relazioni d'uso con altre componenti:** Questo elemento è in relazione con:
 - Model::Publishers::QuizPublisher
 - Model::Publishers::QuestionPublisher
- **Metodi:**
 - + question.search() : Metodo per ricercare una stringa all'interno di tutte le domande

Parametri:

 - * string: string

Precondizioni: Viene fornito il parametro che si vuole ricercare

Postcondizioni: Vengono restituite tutte le domande che contengono il parametro
 - + quizzes.search() : Metodo per ricercare una stringa all'interno di tutti i questionari

Parametri:

 - * string: string

Precondizioni: Viene fornito il parametro che si vuole ricercare

Postcondizioni: Vengono restituite tutti i questionari che rispecchiano i criteri di ricerca

3.2 Model::Parser

3.2.1 Model::Parser::Parser

Parser (from Parser)
+checkQML(QMLtext: string) -checkMU() -checkMX() -checkAS() -checkOD() -checkVF()

- **Funzione del componente:** controlla che il testo fornito risulti corretto secondo la sintassi QML
- **Relazioni d'uso con altre componenti:** Questo elemento è in relazione con:
 - Model::Database::QuestionManager
- **Metodi:** Le seguenti funzioni utilizzano le espressioni regolari per il controllo sintattico del codice QML. Le espressioni regolari sono definite a livello globale come segue:

```
var VF = /^[\s]*<question [\s]*type[\s]*=[\s]*"(VF)">[\s]*<text>[\s]*(.*) [\s]*</text>[\s]*<answer [\s]*isRight[\s]*=[\s]*"(yes|no)" [\s]*>(.*) </answer>[\s]*</question>[\s]*$/;
```

```
var MU = /^[\s]*<question [\s]*type[\s]*=[\s]*"(MU)">[\s]*<text>[\s]*(.*) [\s]*</text>((( [\s]*<answer [\s]*isRight[\s]*=[\s]*"(yes|no)" [\s]*>(.*) </answer>[\s]*) {2,}) </question>[\s]*$/;
```

```
var MX = /^[\s]*<question [\s]*type[\s]*=[\s]*"(MX)">[\s]*<text>[\s]*(.*) [\s]*</text>((( [\s]*<answer [\s]*isRight[\s]*=[\s]*"(yes|no)" [\s]*>(.*) </answer>[\s]*) {2,}) </question>[\s]*$/;
```

```
var AS = /^[\s]*<question [\s]*type[\s]*=[\s]*"(AS)">[\s]*<text>[\s]*(.*) [\s]*</text>[\s]*((<answer [\s]*set[\s]*=[\s]*"(A|B)" [\s]*pos[\s]*=[\s]*"(\d)" [\s]*>(.*) </answer>[\s]*) {2,}) </question>[\s]*$/;
```

```
var OD = /^[\s]*<question [\s]*type[\s]*=[\s]*"(OD)">[\s]*<text>[\s]*(.*) [\s]*</text>[\s]*((<answer [\s]*pos[\s]*=[\s]*"(\d)" [\s]*>(.*) </answer>[\s]*) {2,}) </question>[\s]*$/;
```

- + `checkQML()` : Metodo per controllare che un nuovo quesito rispetti la sintassi QML prima di essere aggiunto al database.

Parametri: `QMLText`: string Il metodo prende in input il parametro `QMLText` di tipo string che rappresenta la stringa composta dall'utente in seguito al tentativo di creazione della domanda.

Precondizioni: Viene richiesta una string che contiene la domanda

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

- - `checkMU()`: funzione che controlla se è presente una ed una sola risposta giusta

Precondizioni: la funzione `checkQML()` ha preso in input il testo prodotto dall'utente ed è stata creata e definita la variabile `m` di tipo array che contiene i risultati della funzione `exec()` chiamata in `checkQML()`

Postcondizioni: viene restituita una variabile di tipo bool riguardante l'esito positivo o negativo del controllo

- - `checkMX()`: funzione che controlla se sono presenti due o più risposte giuste

Precondizioni: la funzione `checkQML()` ha preso in input il testo prodotto dall'utente ed è stata creata e definita la variabile `m` di tipo array che contiene i risultati della funzione `exec()` chiamata in `checkQML()`

Postcondizioni: viene restituita una variabile di tipo bool riguardante l'esito positivo o negativo del controllo

- - `checkAS()`: funzione che controlla la coerenza degli insiemi A e B definiti nella domanda prodotta dall'utente. Viene controllato che l'utente abbia dato agli insiemi i nomi corretti e sia stato definito un codice identificativo per ogni elemento di ciascun insieme

Precondizioni: la funzione `checkQML()` ha preso in input il testo prodotto dall'utente ed è stata creata e definita la variabile `m` di tipo array che contiene i risultati della funzione `exec()` chiamata in `checkQML()`

Postcondizioni: viene restituita una variabile di tipo bool riguardante l'esito positivo o negativo del controllo

- - `checkOD()`: funzione che controlla che ogni risposta abbia un codice identificativo univoco (in relazione alle altre risposte) in modo da poter effettuare l'ordinamento delle stesse

Precondizioni: la funzione `checkQML()` ha preso in input il testo prodotto dall'utente ed è stata creata e definita la variabile `m` di tipo array che contiene i risultati della funzione `exec()` chiamata in `checkQML()`

Postcondizioni: viene restituita una variabile di tipo bool riguardante l'esito positivo o negativo del controllo

3.2.2 La sintassi QML

I seguenti metodi servono a controllare che la sintassi sia corretta secondo le regole QML. Ogni domanda deve avere un tipo appartenente al seguente elenco:

- **VF**: domanda di tipo vero/falso
- **MU**: domanda a risposta a scelta multipla con una sola risposta giusta
- **MX**: domanda a risposta a scelta multipla con più risposte giuste
- **AS**: domanda di associazioni tra elementi dell'insieme A ed elementi dell'insieme B
- **OD**: domanda di ordinamento delle risposte date

Ogni domanda deve contenere i seguenti campi:

- Tipo domanda (VF, MU, MX, AS, OD)
- Testo della domanda
- Elenco risposte (di tipologia diversa in base al tipo di domanda)
- Percorso immagine allegata (opzionale)

I suddetti campi vengono definiti nel modo seguente:

- `<question type="VF|MU|MX|AS|OD" image="path/to/image"> ... </question>`

Definisce l'inizio e la fine dell'oggetto domanda completo. L'attributo **type** definisce la tipologia di domanda che si vuole creare. Nel caso si volesse allegare un'immagine alla domanda si deve aggiungere l'attributo **image** a cui va assegnato il percorso relativo dell'immagine da allegare.

- `<text> </text>`
Per definire il campo relativo al testo della domanda
- I campi per le risposte cambiano in base alla tipologia della domanda:

VF:

`<answer isRight="yes|no"></answer>`

Per il tipo vero/falso deve esserci un solo elemento risposta. L'attributo **isRight** definisce se la risposta è vera o falsa

MU:

`<answer isRight="yes|no"> risposta </answer>`

Numero risposte > 1

Per ogni risposta va definito se è corretta o no. Il numero di risposte corrette deve essere = 1

MX:

`<answer isRight="yes|no"> risposta </answer>`

Numero risposte > 2

Per ogni risposta va definito se è corretta o no. Il numero di risposte corrette deve essere ≥ 1

AS:

`<answer set="A" pos="1"> risp </answer>`

Numero risposte > 1

Per ogni risposta va definito a che insieme appartiene tramite l'attributo **set**, e il numero identificativo di riferimento tramite l'attributo **pos**

OD:

`<answer pos="1"> risp </answer>`

Numero risposte > 1

Ad ogni risposta viene associata la sua posizione tramite la definizione dell'attributo **pos**

3.2.3 Esempi di domande secondo la sintassi corretta

- Vero/Falso:

```
<question type="VF">
  <text>
    testo della domanda
  </text>
  <answer isRight="yes"></answer>
</question>
```

- Risposta multipla:

```
<question type="MU">
  <text>
    testo della domanda
  </text>
  <answer isRight="yes"> risp1 </answer>
  <answer isRight="no"> risp2 </answer>
  <answer isRight="no"> risp3 </answer>
</question>
```

- Risposta multipla con più risposte corrette:

```
<question type="MX">
  <text>
    testo della domanda
  </text>
  <answer isRight="yes"> risp1 </answer>
  <answer isRight="yes"> risp2 </answer>
  <answer isRight="no"> risp3 </answer>
</question>
```

- Associazione:

```
<question type="AS">
  <text>
    testo della domanda
  </text>
  <answer set="A" pos="1"> risp1 </answer>
  <answer set="A" pos="2"> risp2 </answer>
  <answer set="B" pos="1"> risp3 </answer>
  <answer set="B" pos="2"> risp4 </answer>
  <answer set="B" pos="3"> risp5 </answer>
</question>
```

- Ordinamento:

```
<question type="OD">
  <text>
    testo della domanda
  </text>
  <answer pos="2"> risp2 </answer>
  <answer pos="1"> risp1 </answer>
  <answer pos="3"> risp4 </answer>
  <answer pos="5"> risp6 </answer>
  <answer pos="4"> risp5 </answer>
</question>
```

3.3 Model::Statistics

3.3.1 Model::Statistics::Statistics

Statistics (from Statistics)
+statistics.QuizExecutionStats(QuizObject: JSON, userID: string) +statistics.UserExecutionStats(QuizObject: JSON, UserID: string) +statistics.UserExecutionStats(UserID: string, QuestionAnswered: array, QuestionCorrectAnswered: array) +statistics.QuestionExecutionStats(QuestionID: string, Correct: bool)

- **Funzione del componente:** questa classe fornisce funzionalità per il raccoglimento delle statistiche sulle prestazioni degli utenti del sistema
- **Relazioni d'uso con altre componenti:** Questo elemento è in relazione con:
 - Model::Publishers::QuizPublishers
 - Model::Publishers::questionPublishers
- **Metodi:**
 - + statistics.QuizExecutionStats() : Metodo per aggiornare le statistiche di un quesito nel database
Parametri:
 - * QuizObject: JSON object
 - * UserID: string**Precondizioni:** L'utente ha consegnato il questionario
Postcondizioni: Le statistiche relative ad ogni domanda contenuta nel questionario sono state aggiornate
 - + statistics.UserExecutionStats() : Metodo per aggiornare le statistiche sulla performance di un utente registrato
Parametri:
 - * UserID: string
 - * QuestionAnswered: array/string/int
 - * QuestionCorrectAnswered: array/string/int**Precondizioni:** L'utente registrato ha consegnato un questionario
Postcondizioni: Vengono aggiornate le statistiche totali di quell'utente
 - + statistics.QuestionExecutionStats() : Metodo per aggiornare le statistiche di un quiz nel database
Parametri:

* QuestionID: string

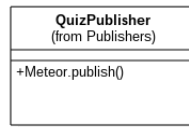
* Correct: bool

Precondizioni: L'utente ha dato una risposta alla domanda

Postcondizioni: La risposta (corretta o meno) viene integrata nelle statistiche

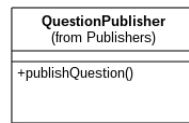
3.4 Model::Publishers

3.4.1 Model::Publishers::QuizPublisher



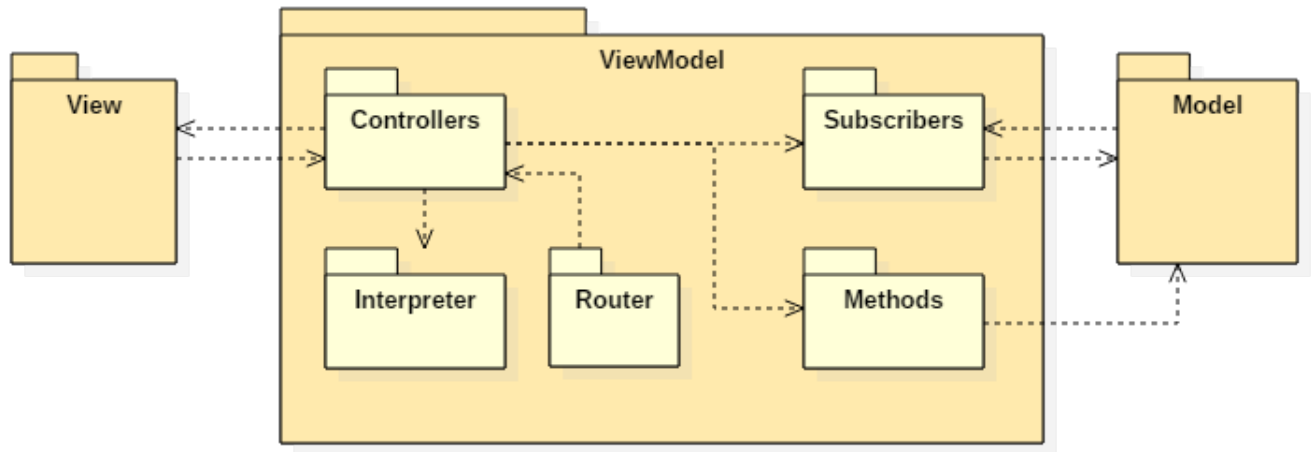
- **Funzione del componente:** questa classe fornisce funzionalita' per rendere pubblica la collezione dei quiz all'avvio del Sistema
- **Relazioni d'uso con altre componenti:** Nessuna
- **Metodi:**
 - **+ Meteor.publish()** : Metodo per rendere accessibili la collezione dei quiz all'avvio del Sistema
 - 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



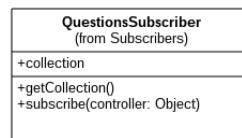
- **Funzione del componente:** questa classe fornisce funzionalita' per rendere pubblica la collezione dei quesiti all'avvio del Sistema
- **Relazioni d'uso con altre componenti:** Nessuna
- **Metodi:**
 - + **publishQuestion()** : Metodo per rendere accessibili la collezione dei quesiti all'avvio del 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::Subscribers

4.1.1 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::QuestionPublisher
- **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

4.1.2 ViewModel::Subscribers::QuizSubscriber

QuizSubscriber (from Subscribers)
+collection: string
+getCollection()
+subscribe(controller : Object)

- **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 ViewModel::Methods

4.2.1 ViewModel::Methods::QuestionMethods

QuestionMethods (from Methods)
+collection: string
+questions.insert(QMLText: string, category: string)
+questions.remove(questionId: string)

- **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
 - Model::Statistics::Statistics
- **Attributi:**
 - - collection: String: la collezione sulla quale verranno implementati i methods
- **Metodi:**
 - + questions.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:
 - * QMLtext: string: testo QML (corretto perchè viene prima parsato) corrispondente alla nuova domanda
 - * category: string: testo contenente la categorie di appartenenza della domanda che si vuole inserire
 - + questions.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

4.2.2 ViewModel::Methods::QuizMethods

QuizMethods (from Methods)
+collection: string
+quizzes.insert(title: string, questions: array, categories: array, description: string, time: int)

- **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:**
 - + quizzes.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:
 - * title: string: contiene il titolo del nuovo questionario
 - * questions: array: elenca le domande presenti nel nuovo questionario
 - * categories: array: elenca le categorie di pertinenza del nuovo questionario
 - * description: string: contiene una breve descrizione testuale del questionario
 - * time: int: stabilisce il tempo massimo per la risoluzione del quiz
 - + quizzes.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

4.2.3 ViewModel::Methods::UserMethods

UserMethods (from Methods)
+collection: string
+insert(userID: string) +remove(userID: string)

- **Funzione del componente:** permette al client di richiedere la modifica della collezione di utenti del server;
- **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

4.3 ViewModel::Interpreter

4.3.1 ViewModel::Interpreter::Interpreter

Interpreter (from Interpreter)
+QML2HTML(QMLText: string) -getRightAnsMU() -getRightAnsMX() -getRightAnsAS() -getRightAnsOD() +HTML2QML(JSElement: Object)

- **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 ViewModel::Router

4.4.1 ViewModel::Router::Router

Router (from Router)
+locationProvider +urlRouterProvider +stateProvider
+config() +state(state: string, url: string, template: string)

- **Funzione del componente:** implementa il routing dinamico dell'applicazione. Permette di dividere la parte statica dell'applicazione dalla parte che viene elaborata dinamicamente;
- **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:**
 - config(): si occupa di gestire le varie chiamate del metodo state() all'inizializzazione dell'applicazione
Precondizioni: l'applicazione Quizzipedia viene avviata
Postcondizioni: tutti i template sono disponibili al caricamento
 - 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

4.5 ViewModel::Controller

4.5.1 ViewModel::Controller::QuestionList

QuestionList (from Controller)
-controller +templateURL
+getQuestionDetails(QMLText: string) +deleteQuestion(questionID: int) +openAlertQuestion(questionID: int)

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

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

La classe si relaziona con:

- View::Templates::Question
- View::Templates::QuestionList
- Model::Publishers::QuestionPublishers
- ViewModel::Interpreter::Interpreter

- **Funzionalità:**

- + getQuestionDetails(QMLtext): restituisce i dettagli sulla domanda corrente se essa è valida;
- + deleteQuestion(questionID): richiede la rimozione dal database di una domanda;
- + openAlertQuestion(questionID): fornisce un messaggio pop-up di conferma all'utente;

- **Attributi:**

- + controller: String: variabile contenente il nome del controller del template;
- + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template

4.5.2 ViewModel::Controller::Question

Question (from Controller)
+controller: string +templateURL: string +category: string +type: string

- **Funzione del componente:** permette la visualizzazione di una singola domanda
- **Relazioni d'uso con altre componenti:** La classe è in relazione con:
 - View::Templates::Question
 - View::Templates::QuestionList
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + 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

4.5.3 ViewModel::Controller::QuizList

QuizList (from Controller)
+helpers.quizzes() +helpers.questions() +helpers.inputCategories() +getAuthor(quizOwnerId: int) +isOwner(ownerID: int) +setQuiz(quizID: int) +deleteQuiz(quizID) +openAlertQuiz(quizID: int)

- **Funzione del componente:** permette visualizzazione e interazione di una lista di questionari
- **Relazioni d'uso con altre componenti:**
La classe utilizza:
 - View::Templates::Quiz
 - Model::Publishers::QuizPublishers
 - Model::Publishers::QuestionPublishers
 - Model::Parser::Parser
- **Funzionalità:**
 - + helpers.quizzes(): restituisce i questionari della categoria ricercata;
 - + helpers.questions(): restituisce le domande contenute nel questionario selezionato;
 - + helpers.inputCategories(): recupera tutte le categorie dei quiz per popolare la select senza inserire duplicati;
 - + getAuthor(quizOwnerId): restituisce l'autore del quiz selezionato
 - + isOwner(ownerID): restituisce un esito positivo se il creatore del quiz è l'utente attualmente loggato
 - + setQuiz(qID: richiede il questionario scelto dall'utente per il caricamento dal database
 - + deleteQuiz(quiz): permette l'eliminazione del quiz scelto
 - + openAlertQuiz(quizID): fornisce un messaggio pop-up di conferma all'utente;

4.5.4 ViewModel::Controller::Quiz

Quiz (from Controller)
+controller: string +templateUrl: string +title: string +categories: string

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

4.5.5 ViewModel::Controller::QuestionForm

QuestionForm (from Controller)
+controller: string +templateUrl: string +text: string
+check(QMLText: string) +saveQuestion(QmlText: string, category: string)

- **Funzione del componente:** visualizza il form utilizzabile sia per la creazione che per la modifica di una domanda
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + 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
- **Metodi:**
 - + check(QMLtext): si occupa della verifica della correttezza del testo QML inserito dall'utente;
 - + saveQuestion(QMLtext, category): salva le modifiche effettuate su una domanda;

4.5.6 ViewModel::Controller::QuizCreationForm

QuizCreationForm (from Controller)
+controller: string +templateUrl: string
+toggleSelection(questionID: string) +saveQuiz(title, questions, categories, description, time) +getQuestionDetails(QMLText: string)

- **Funzione del componente:** visualizza il form di creazione di un questionario
- **Relazione d'uso con altri componenti:**
 - View::Pages::QuizCreationPage
 - Model::Publishers::QuestionPublishers
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
- **Metodi;**
 - + toggleSelection(questionID): si occupa di gestire la selezione della domanda da inserire nel questionario attualmente in fase di creazione;
 - + savequiz(title,questions,categories,description,time): gestisce il salvataggio del questionario;
 - + getQuestionDetails(QMLtext): fornisce informazioni su una specifica domanda candidata all'inserimento nel questionario;

4.5.7 ViewModel::Controller::QuestionCompilation

QuestionCompilation (from Controller)
+controller +templateUrl: string

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

4.5.8 ViewModel::Controller::QuizResults

QuizResults (from Controller)
+controller: string
+templateUrl: string
+getQuiz()

- **Funzione del componente:** visualizza i risultati ottenuti in seguito alla compilazione di un quiz
- **Relazioni con altre componenti**
La classe utilizza:
 - View::Templates::QuestionCompilation
 - View::Templates::QuizResults
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
- **Metodi:**
 - + getQuiz(): si occupa di fornire il punteggio finale

4.5.9 ViewModel::Controller::SearchForm

SearchForm (from Controller)
+controller: string +templateUrl +query: string
+find() +show() +clear()

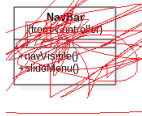
- **Funzione del componente:** visualizza una form per l'inserimento dei dati desiderati e l'avvio della ricerca
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
 - + query: String: stringa contenente la query da sottomettere al sistema
- **Funzionalità:**
 - + find(): esegue l'effettiva attività di ricerca nel sistema
 - + show(): fornisce la rappresentazione grafica dei risultati
 - + clear(): metodo di servizio per la pulizia dell'interfaccia

4.5.10 ViewModel::Controller::NavBar

NavBar (from Controller)
+navVisible() +slideMenu()

- **Funzione del componente:** Fornisce un menù laterale che facilita l'interazione dell'utente con l'applicazione
- **Relazione con altri componenti:**
 - View::Templates::navbar
- **Funzionalità:**
 - + navVisible(): imposta la larghezza del menù laterale
 - + slideMenu(): gestisce effetti grafici responsive

4.5.11 ViewModel::Controller::TopBar



- **Funzione del componente:** Fornisce il menù orizzontale con cui l'utente può interfacciarsi con il sistema
- **Relazione con altri componenti:**
 - View::Templates::topbar

4.5.12 ViewModel::Controller::UserProfile

UserProfile (from Controller)
+controller +templateUrl
+getQuestionsNum() +getQuizNum() +getUsername()

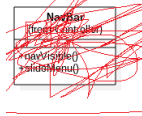
- **Funzione del componente:** permette la visualizzazione e l'interazione dell'utente con il proprio profilo e le funzionalità da esso offerto
- **Relazione con altri componenti:**
 - View::Templates::UserProfile
 - ViewModel::Controller::QuestionList
 - ViewModel::Controller::QuizList
 - Model::Publishers::QuestionPublishers
 - Model::Publishers::QuizPublishers
- **Attributi:**
 - + controller: String: variabile contenente il nome del controller del template;
 - + templateUrl: String: stringa contenente il percorso del file HTML che contiene il template
- **Metodi:**
 - + getQuestionsNum(): ritorna la quantità di domande create dallo stesso utente;
 - + getQuizNum(): ritorna la quantità di questionari creati dallo stesso utente;
 - + getUsername(): ritorna il nome dell'utente loggato;

4.5.13 ViewModel::Controller::QuizHome



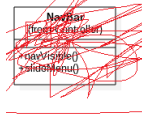
- Funzione del componente:
- Relazione con altri componenti:
 - V
- Attributi:
 - a
- Metodi:
 - V

4.5.14 ViewModel::Controller::QuizStatistics



- Funzione del componente:
- Relazione con altri componenti:
 - V
- Attributi:
 - a
- Metodi:
 - wddwdad

4.5.15 ViewModel::Controller::QuizCompilation



- **Funzione del componente:**
- **Relazione con altri componenti:**
 - afewf
- **Attributi:**
 - a
- **Metodi:**
 - faf

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::NewQuizController	F 6
ViewModel::Controllers::QuestionManagementController	F 5.3
ViewModel::Controllers::DeleteQuestionController	F 5.4
ViewModel::Controllers::DeleteQuizController	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