QUIZZIPEDIA

team404swe@gmail.com

Specifica Tecnica 1.0

Informazioni sul documento	
Nome Documento	Specifica Tecnica 1.0
Versione	1.0
Uso	Esterno
Data Creazione	22 aprile 2016
Data Ultima Modifica	12 maggio 2016
Redazione	M. Crivellaro - L. Alessio
Verifica	Alex Beccaro
Approvazione	Martin V. Mbouenda
Committente	Zucchetti SPA
Lista di distribuzione	Prof. Vardanega Tullio TEAM404

Registro delle modifiche

Versione	Autore	Data	Descrizione
0.1.0	Luca Alessio (Progettista)	30/04/2016	Prima stesura sezioni "Package della componente View" e "Diagrammi delle classi del View"
0.0.6	Davide Bortot (Progettista)	28/04/2016	Stesura delle sezioni §2.2, §2.3, §2.4 con la descrizione d'alto livello delle componenti MVP.
0.0.5	Luca Alessio (Progettista)	27/04/2016	Stesura parziale sezione "Tecnologie e strumenti utilizzati"
0.0.4	Davide Bortot (Progettista)	25/04/2016	Ampliata la sezione "Architettura generale del sistema".
0.0.3	Luca Alessio (Progettista)	24/04/2016	Stesura parziale sezione "Architettura generale del sistema"
0.0.2	Davide Bortot (Progettista)	23/04/2016	Strutturazione iniziale del documento e stesura sezione introduttiva
0.0.1	Marco Crivellaro (Progettista)	22/04/2016	Creazione documento.

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

1 Introduzione

1.1 Scopo del documento

1.2 Scopo del prodotto

Il progetto **Quizzipedia** ha come obiettivo lo sviluppo di un sistema software basato su tecnologie Web (Javascript$_G$, Node.js$_G$, HTML5$_G$, CSS3$_G$) 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 QML$_G$. 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 HTML$_G$, 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_2.0.pdf*".

1.3 Glossario

Viene allegato un glossario nel file "*glossario_2.0.pdf*" nel quale viene data una definizione a tutti i termini che in questo documento appaiono con il simbolo '$_G$' 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_2.0.pdf*"

1.4.2 Informativi

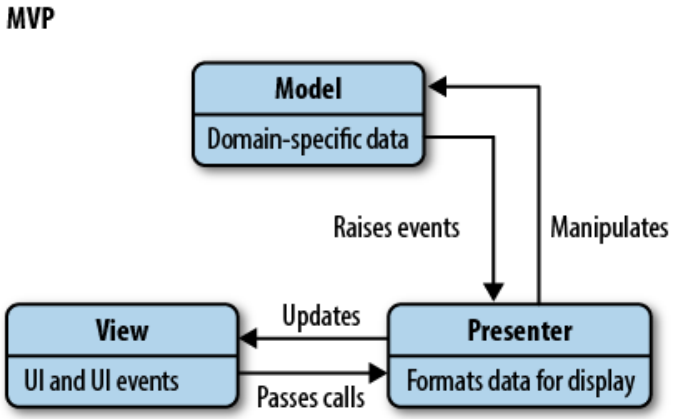
- Corso di Ingegneria del Software anno 2015/2016:
  http://www.math.unipd.it/~tullio/IS-1/2015/
- Regole del progetto didattico:
  http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/PD01.pdf http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/

2 Specifiche del prodotto

I contenuti della specifica saranno presentati seguendo l'approccio top-down: dalla descrizione macroscopica del sistema si scenderà sempre più in dettaglio passando alla descrizione delle singole componenti. Verranno inoltre descritti i design pattern utilizzati e come essi sono stati applicati. Per agevolare la comprensione si è scelto di utilizzare i diagrammi dei package, delle classi, di attività e di sequenza, descritti attraverso lo standard UML 2.0.

2.1 Architettura generale del sistema

Il sistema Quizzipedia è di tipo *client-server*; il *client* fornisce all'utente un'interfaccia web su browser per la creazione e fruizione di questionari, mentre il lato *server* si occupa di gestire e salvare i dati su DBMS PostgreSQL. La base di dati raccoglie principalmente quesiti memorizzati in QML, che su richiesta verranno elaborati da un interprete. Il suddetto interprete processa l'input QML, precedentemente validato nella forma da un parser apposito, traducendolo in linguaggio HTML5 visualizzabile da browser. «««< HEAD Nella realizzazione del sistema Quizzipedia verrà adottato il design pattern *Model View Presenter* nella sua variante *Passive View*.



- **Model**: definisce l'organizzazione dei dati e ne specifica le modalità di accesso. Nel sistema Quizzipedia è situato nella parte server che opera sulla sottostante base di dati. Fa parte del Model anche la componente *Parser* che opera sui dati prima che essi vengano salvati nel DBMS.
- **View (Passive)**: rappresenta l'interfaccia grafica presentata all'utilizzatore, la quale visualizza i dati e cattura le interazioni dell'utente. L'aggettivo "Passive" indica che la View non è responsabile del proprio aggiornamento al variare del Model, compito che ricade sul Presenter.
- **Presenter**: controlla la View e ne gestisce il comportamento in reazione alle interazioni dell'utente, interagisce di conseguenza col Model per ottenere i dati necessari. Una volta ottenuti i dati si preoccupa di aggiornare la View. Nel sistema Quizzipedia implementa la parte logica, affiancata a quella grafica, del Client. Realizza un totale disaccoppiamento tra Model e View, controllandone i flussi di comunicazione.

Altre possibili architetture che sono state prese in considerazione sono definite dai design pattern *Model View Controller (MVC)* con *Front Controller*, *Model View Presenter (MVP)* nelle varianti *Presentation Model* e *Supervising Controller*, e il design pattern *Model View ViewModel (MVVM)*.

- La prima pone il Controller, componente simile al Presenter, nella parte server del sistema. Quest'opzione è stata scartata per evitare di aumentare troppo la complessità del lato server, ponendo invece il Presenter dal lato client, così da redistribuire responsabilità e carico di lavoro.
- Il *Presentation Model* invece è affine al design pattern scelto, ma impone che sia la componente View ad aggiornarsi autonomamente al variare del Model. Proprio per questo motivo si è deciso di scartarla in favore del Passive View: per disaccoppiare completamente le componenti Model e View, e per alleggerire ulteriormente quest'ultima concentrandone tutta la parte logica nel Presenter.
- Il *Supervising Controller* propone che la View si aggiorni autonomamente nel caso di piccole modifiche (tramite data-binding col Model), lasciando le manipolazioni più complicate al Presenter; è stato scartato in favore del disaccoppiamento totale tra View e Model.
- Il pattern *MVVM* prevede di creare per ogni View un ViewModel, che rappresenta tutte le informazioni e i comportamenti della corrispondente View. La View si limita infatti, a visualizzare graficamente quanto esposto dal ViewModel, a riflettere in esso i suoi cambi di stato oppure ad attivarne dei comportamenti (tramite data-binding). Tale architettura è indicata per applicazioni particolarmente dinamiche in cui View e Model devono essere costantemente aggiornati. Non è questo il caso di Quizzipedia.

2.2 Descrizione del componente Model

Il Model, situato nella parte server del sistema svolge le seguenti funzioni:

- Interagisce con un database PostgreSQL nel quale vengono salvati i dati del sistema (ad es. domande, statistiche, utenti). Fornisce quindi adeguate funzionalità di salvataggio e caricamento da database di tali dati.
- Al momento del salvataggio di una nuova domanda esegue il *parsing* del codice QML tramite un componente chiamato *Parser*. Se la domanda è sintatticamente corretta può essere salvata nel database.
- Offre un'interfaccia logica di accesso al Presenter attraverso la quale richiedere dati e operazioni su di essi. Quest'interfaccia sarà l'unico punto di accesso disponibile al Presenter. Per accentrare le funzionalità d'accesso si farà uso del design pattern *Façade*.

2.3 Descrizione del componente View

La componente View rappresenta l'interfaccia grafica che visualizza i dati del Model e inoltra i comandi dell'utente (o gli eventi da esso generati) al Presenter che si occuperà di gestire tali richieste sui dati interagendo col Model; la View si occupa quindi solamente della rappresentazione grafica dei dati e non ha alcun contatto diretto con essi. Essendo un'interfaccia web verrà realizzata tramite HTML5 e CSS3 per le parti statiche e con l'utilizzo di Javascript per le parti dinamiche. Per assicurare uno stile coerente tra le pagine web e migliorare l'adattabilità a piattaforme mobile verrà utilizzato il framework *Materialize*.

2.4 Descrizione del componente Presenter

Il Presenter ricopre tre ruoli fondamentali: recepire ed elaborare gli input dell'utente, comunicare col Model, ed aggiornare il View con i dati ottenuti. Per poterlo fare possiede le seguenti caratteristiche:

- Conosce i riferimenti alle altre due componenti. Il Presenter è l'unica componente che conosce entrambe le altre e facendo da singolo tramite tra Model e View permette il loro totale disaccoppiamento;
- E' in grado di elaborare gli input della View e tradurli in azioni sul Model. Viceversa ad ogni modifica del Model si preoccupa di aggiornare di conseguenza la View.
- E' responsabile della traduzione delle domande dal formato QML a formato HTML visualizzabile da browser, tramite un componente *Interpreter*. Tale funzionalità viene richiesta ogni qualvolta il Presenter richiede e riceve dal Model una domanda in formato QML.
- Possiede dei gestori che possano modificare l'aspetto della View in reazione all'interazione dell'utente o ai dati ricevuti dal Model. Al suo interno il Presenter contiene delle classi che in risposta ad un evento, quale l'interazione dell'utente con la View o il ricevimento di una risposta dal Model, modificano l'aspetto della GUI presentata.
- Gestisce la somministrazione di un questionario ad un utente, domanda dopo domanda, fino alla consegna e valutazione.

3 Tecnologie e strumenti utilizzati

3.1 HTML5

Linguaggio di markup per la progettazione di pagine Web. Richiesto espressamente nel capitolato per la creazione dell'interfaccia utente (mi pare, devo controllare).

- **Utilizzo**: viene utilizzato per creare la GUI che permette all'utente di accedere al sistema mediante browser.
- **Vantaggi**: favorisce la portabilità su diversi dispositivi (desktop, smartphone, tablet...) e browser. Sono punti a favore anche l'elevata compatibilità con tecnologie quali CSS3 e Javascript.
- **Svantaggi**: essendo HTML5 un linguaggio non ancora standard il rischio nel suo utilizzo è l'instabilità dei tag utilizzati. Un tag oggi accettato potrebbe essere modificato in un futuro prossimo, rendendo la visualizzazione dell'interfaccia utente dipendente dal stabilità dei tag utilizzati.

3.2 CSS3

Principale linguaggio usato per la formattazione di pagine HTML.

- **Utilizzo**: Viene utilizzato per formattare il codice HTML, ovvero creare fogli di stile che permettono all'utente di modificare alcuni aspetti grafici della pagina Web
- **Vantaggi**: Richiede un minor sforzo di interpretazione da parte del browser ed è leggero da scaricare
- **Svantaggi**: Può presentare problemi di compatibilità con browser meno recenti

3.3 Javascript

E' un linguaggio di scripting debolmente orientato agli oggetti, utilizzato nelle applicazioni Web. Viene interpretato all'interno del browser. Permette di definire funzionalità simili a quelle offerte da C++ e Java, quali cicli e strutture di controllo. Viene solitamente affiancato a pagine statiche HTML per poter gestire i contenuti dinamicamente, offrendo funzionalità che il linguaggio di markup non può offrire.

- **Utilizzo**: nel sistema Quizzipedia Javascript rivestirà un ruolo importante, verrà infatti utilizzato nella realizzazione del parser/interprete/boh andrea dimmi te.
- **Vantaggi**: eseguito lato Client (sul browser) non sovraccarica il Server per l'esecuzione di richieste, anche se lo script è complesso.
- **Svantaggi**: per script sorgenti molto corposi, può risultare oneroso in termini di tempo lo scaricamento dei contenuti. Deve, inoltre, fare affidamento ad un linguaggio che possa fisicamente effettuare transazioni di dati quando lo script esegue operazioni su oggetti remoti (eg: database). E' altresì un linguaggio non tipizzato, quindi occorre porre attenzione ai tipi delle variabili che non sono dichiarati, ma variano dinamicamente.
- **Variabili e oggetti**: le variabili se sono dichiarate all'interno di una funzione sono visibili solo all'interno di essa; se sono invece esterne sono globali. Vengono dichiarate con la keyword *var* o semplicemente assegnando loro un valore. Ogni elemento in JavaScript è un tipo primitivo o un oggetto. Gli oggetti sono entità dotate di unicità (sono uguali solo a sé stessi) e identificabili con vettori associativi, che associano nomi di proprietà a valori.

3.4 Node.js

- **Utilizzo**:
- **Vantaggi**:
- **Svantaggi**:

3.5 PostgreSQL

DBMS ad oggetti open-source.

- **Utilizzo**: Base di dati con lo scopo di memorizzare domande e altri dati necessari al funzionamento del sistema.
- **Vantaggi**: Tecnologia più robusta, stabile e performante di MySQL (inizialmente preso in considerazione ma poi scartato in favore di PostgreSQL).
- **Svantaggi**: Complessità maggiore rispetto al classico MySQL.

4 Diagrammi dei packages

4.1 Package della componente Model

4.2 Package della componente View

DA INSERIRE FIGURA PACKAGE VIEW
Il package per il componente View del pattern architetturale MVP contiene i seguenti sotto packages:

- **CurrentViewManager:** questo sotto package ha lo scopo di definire lo stato attuale del sistema Quizzipedia; per fare ciò si avvale delle classi:
  - *CurrentView*
  - *Sender*
- **Pages:** contiene la classe astratta *Page*, che rappresenta una specifica situazione del sistema (ovvero una pagina web del sito), più tutte le sue derivazioni concrete:
  - *MainPage*
  - *CategoryListPage*
  - *QuizListPage*
  - *QuizExecutionPage*
  - *QuizManagementPage*
  - *ViewTutorialPage*
- **UserAuthentication:** contiene la classe *User* e la sua derivata *Admin* che raggruppano le funzionalità di autenticazione di un utente generico sulla piattaforma Quizzipedia; come *Pages* anche *UserAuthentication* è un package d'appoggio per la classe *CurrentViewManager*::*CurrentView*.

4.3 Package della componente Presenter

5 Diagrammi delle classi

5.1 Diagrammi delle classi del Model

5.2 Diagrammi delle classi del View

CONTENUTO DA VERIFICARE NON SON SICURO RISPETTI LE ULTIME DECISIONI PRESE ANCHE NUMERAZIONe DA RIVEDERE PERCHE' NON ESISTONO LE SUBSUBSUBSECTION

5.2.1 Package CurrentViewManager

IMMAGINE **CurrentView**
Funzione del componente: lo scopo di questa classe è quello di rappresentare la vista del sistema attualmente disponibile all'utente e di gestirne eventuali cambiamenti **Relazioni d'uso di altri componenti:** (da dire bene) toPresenter, fromPresenter e currentUser sono parametri di tipi appartenenti ad altri package percui c'è una relazione di dipendenza (?) verso i rispettivi packages Sender, Receiver ed UserAuthentication **Attività svolte e dati trattati:** sono presenti dei parametri per la memorizzazione dell'utente corrente, della pagina corrente e per l'invio e la ricezione di informazioni dal Presenter. La funzionalità principale offerta è enterLink (nome da rivedere) che, come suggerisce il nome, permette all'utente di spostarsi nell'applicazione cambiando pagina. Sono presenti altre funzionalità minori che gestiscono l'avvio e la chiusura della sessione e il ritorno alla homepage. **Sender**
Funzione del componente: inoltra le richieste dell'utente al presenter **Relazioni d'uso di altri componenti:** viene utilizzata all'interno di CurrentViewManager::CurrentView in risposta agli eventi che accadono sull'interfaccia web in seguito alle azioni compiute dall'utente; l'output dei metodi di questa classe viene poi processato dal presenter **Attività svolte e dati trattati:** ogni metodo di questa classe corrisponde ad una situazione che si verifica, attraverso un rispettivo metodo, in una classe derivata da Pages::Page; i dati restituiti da questi primi metodi verranno reinoltrati da questa classe al presenter

5.2.2 Package Pages

IMMAGINE
Page
Funzione del componente: rappresenta una pagina web //meglio renderla interfaccia?
Relazioni d'uso di altri componenti: classe astratta che viene concretizzata dalle sue classi derivate (MainPage, CategoryListPage, QuizListPage,QuizTutorialPage, QuizExecutionPage, QuizManagementPage)
Attività svolte e dati trattati: le funzionalità offerte dalla classe sono compiti fondamentali che una pagina web deve svolgere ovvero ricevere gli input dell'utente (quali click su link o dati inseriti in un form che poi andranno passati al presenter) e aggiornarsi quando nuovi dati sono disponibili

MainPage
Funzione del componente: mostra la pagina principale a cui l'utente arriva entrando nella piattaforma Quizzipedia
Relazioni d'uso di altri componenti: concretizza la classe astratta Page da cui è diretta discendente (e viene usata da CurrentView come currentPage)

Attività svolte e dati trattati: permette l'autenticazione/registrazione dell'utente nel sistema, da una panoramica del sistema generale all'utente

CategoryListPage
Funzione del componente: pagina che elenca gli argomenti tra i quali l'utente può scegliere e fornisce operazioni di ordinamento sulla vista
Relazioni d'uso di altri componenti: concretizza la classe astratta Page da cui è diretta discendente (e viene usata da CurrentView come currentPage)
Attività svolte e dati trattati: le funzionalità di questa classe permettono la visualizzazione delle varie categorie, l'ordinamento per diversi criteri (alfabetico, categorie più visitate, ultimi questionari disponibili...) e la scelta di una tra di esse

QuizListPage
Funzione del componente: pagina che elenca i questionari (su uno stesso argomento) tra i quali l'utente può scegliere e fornisce operazioni di ordinamento sulla vista
Relazioni d'uso di altri componenti: concretizza la classe astratta Page da cui è diretta discendente (e viene usata da CurrentView come currentPage). L'argomento visualizzato è quello scelto nella precedente pagina CategoryListPage
Attività svolte e dati trattati: le funzionalità di questa classe permettono la visualizzazione di informazioni sui vari questionari, l'ordinamento per diversi criteri (alfabetico, più visitato, novità...) e la scelta di uno tra essi

QuizExecutionPage
Funzione del componente: questa classe rappresenta il punto focale del sistema Quizzipedia ovvero la parte in cui l'utente svolge i questionari scelti
Relazioni d'uso di altri componenti: concretizza la classe astratta Page da cui è diretta discendente (e viene usata da CurrentView come currentPage). Il questionario visualizzato è quello scelto nella precedente pagina QuizListPage.
Attività svolte e dati trattati: l'utente può navigare tra le domande del questionario nell'ordine che preferisce, dare le proprie risposte e, al termine del questionario, visualizzarne il risultato; per ognuna di queste funzionalità è presente un metodo della classe

QuizManagementPage
Funzione del componente: gestisce creazione, modifica ed eliminazione di singole domande e interi questionari
Relazioni d'uso di altri componenti: concretizza la classe astratta Page da cui è diretta discendente (e viene usata da CurrentView come currentPage)
Attività svolte e dati trattati: le funzionalità offerte dalla classe consentono creazione, modifca ed eliminazione di singoli quesiti e di interi questionari

QuizTutorialPage
Funzione del componente: questa pagina visualizza un breve manuale che spiega l'uso e la sintassi del linguaggio QML (vedi analisi_dei_requisiti_2.0.pdf)
Relazioni d'uso di altri componenti: concretizza la classe astratta Page da cui è diretta discendente (e viene usata da CurrentView come currentPage)
Attività svolte e dati trattati: questa classe svolge solamente una semplice attività di visualizzazione di informazioni

5.3 UserAuthentication

BOZZA
DEVO FINIRE DI SPIEGARE CHE C'E' ANCHE UN UTENTE ADMIN E COSA CAMBIA
UserAuthentication:

User: regola la registrazione, l'accesso e l'uscita di un utente - login(mail,password) - register(newMail,newPassword) - logout() - changePassword() - forgotPassword() Funzione del componente: classe che rappresenta il singolo utilizzatore attuale del sistema nella propria sessione Relazioni d'uso di altri componenti: User viene utilizzato dalla classe CurrentViewManager::CurrentView Attività svolte e dati trattati: questa classe prevede le funzionalità basilari per l'autenticazione dell'utente all'interno del sistema come registrazione, login e logout ed altre operazioni secondarie come il cambio o il recupero della password smarrita

5.4 Diagrammi delle classi del Presenter

6 Diagrammi di attività

7 Diagrammi di sequenza

8 Tracciamento

9 Analisi di fattibilità