18-05-2015

Specifica Tecnica

Informazioni sul documento

Nome Documento	Specifica Tecnica
Versione	1.0.0
Stato	*Formale*
Uso	*Esterno*
Data Creazione	18-05-2015
Data Ultima Modifica	18-05-2015
Redazione	Fossa Manuel, Petrucci Mauro
Approvazione	Tollot Pietro
Verifica	Gabelli Pietro
Lista distribuzione	*LateButSafe*

Prof. Tullio Vardanega
Prof. Riccardo Cardin
Proponente Zucchetti S.p.a.

Registro delle modifiche

Tab 1: Versionamento del documento

Versione	Autore	Data	Descrizione
1.0.0	Tollot Pietro	13-04-2015	Approvazione del documento
0.7.0	Petrucci Mauro	08-04-2015	Apportate le modifiche segnalate dal verificatore Fossa Manuel
0.3.0	Petrucci Mauro	25-03-2015	Aggiunta dei contenuti
0.2.0	Fossa Manuel	24-03-2015	Aggiunta dei contenuti
0.1.0	Busetto Matteo	20-03-2015	Stesura dello scheletro del documento

Storico

pre-RR

Versione 1.0	**Nominativo**
Redazione	Fossa Manuel, Petrucci Mauro
Verifica	Gabelli Pietro
Approvazione	Tollot Pietro

Tab 2: Storico ruoli pre-RR

Indice

Sommario

Il presente documento contiene la specifica tecnica delle componenti che costituiscono il prodotto software Premi.

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del progetto Premi. Verrà presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software e saranno descritti i Design Pattern utilizzati.

1.2 Scopo del Prodotto

Lo scopo del progetto$_g$ è la realizzazione un software$_g$ per la creazione ed esecuzione di presentazioni multimediali favorendo l'uso di tecniche di storytelling e visualizzazione non lineare dei contenuti.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento Glossario_v.1.0.0.pdf. Ogni occorrenza di vocaboli presenti nel Glossario è marcata da una "g" minuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto C4: Premi: Software$_g$ di presentazione "better than Prezi" http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf;
- Norme di Progetto$_g$: NormeDiProgetto_v.1.0.0.pdf;
- Analisi dei Requisiti: AnalisiDeiRequisiti_v.1.0.0.pdf;
- Piano di qualifica: PianoDiQualifica_v.1.0.0.pdf;
- Piano di progetto: PianoDiProgetto_v.1.0.0.pdf.

1.4.2 Informativi

- **Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, 1995**;
- Descrizione dei Design Pattern http://sourcemaking.com/design_patterns;
- Ingegneria del software$_g$ - Ian Sommerville - 9a Edizione (2010):
- Slide del docente per l'anno accademico 2014/2015 reperibili al sito http://www.math.unipd.it/~tullio/IS-1/2014/;
- MongoDB: http://docs.mongodb.org/manual/;

- Node.js: https://nodejs.org/documentation/;
- Angular.js: https://docs.angularjs.org/tutorial;
- jQuery: http://api.jquery.com/ ;
- Impress.js: https://github.com/bartaz/impress.js/.

2 Strumenti

2.1 HTML

Si è deciso di utilizzare HTML5 e CSS3 per la presentazione grafica dell'applicazione web. Si è scelto di utilizzare HTML5 al posto di xHTML 1.1 perchè ormai è diventato uno standard de facto e permette una maggiore integrazione con i linguaggi di scripting e contenuti multimediali di cui questa applicazione fa forte uso.

- **Vantaggi**:
  - **Multi piattaforma**: Poiché l'applicazione deve essere disponibile sia su dispositivi desktop che mobile HTML5 permette la creazione di strutture responsive in grado di adattarsi alle dimensioni dello schermo;
  - **Integrazione con linguaggi di scripting**: Con HTML5 c'è una maggiore integrazione con i linguaggi di scripting come javacript questo permetterà di rendere l'applicazione dinamica;
  - **Nessuna installazione**: Il fatto che l'applicazione sia sviluppata con tecnologie web quali HTML permetterà all'utente finale di poter utilizzare il prodotto senza doverlo scaricare e installare.
- **Svantaggi**:
  - **Browser**: E' possibile che i browser meno recenti abbiano difficoltà ad interpretare correttamente le informazioni contenute nelle pagine, rendendo difficile , se non impossibile, l'utilizzo dell'applicazione con questo linguaggio.

2.2 JavaScript

JavaScript è un linguaggio di scripting lato client orientato agli oggetti, comunemente usato nei siti web, ed interpretato dai browser. Ciò permette di alleggerire il server dal peso della computazione, che viene eseguita dal client. Essendo molto popolare e ormai consolidato, JavaScript può essere eseguito dalla maggior parte dei browser , sia desktop che mobile, grazie anche alla sua leggerezza. Uno degli svantaggi di questo linguaggio è che ogni operazione che richieda informazioni che devono essere recuperate da un database deve passare attraverso un linguaggio che effettui esplicitamente la transazione, per poi restituire i risultati a JavaScript. Tale operazione richiede l'aggiornamento totale della pagina ma è stato superato questo problema adottando delle tecnologie con funzionamento asincrono.

2.3 jQuery

2.4 Angular.js

2.5 Node.js

2.6 MongoDB

2.7 Impress.js

Impress.js una libreria open-source che permette di visualizzare i div di una pagina html come passi di una presentazione. Si è deciso di affidare la visualizzazione della presentazione a questa libreria in quanto permette di conseguire quasi tutti i requisiti obbligatori relativi all'esecuzione senza dover scrivere ingenti quantità di codice aggiuntivo. Si è deciso inoltre di integrare nel framework alcune funzioni in modo da rispondere a tutti i requisiti obbligatori relativi all'esecuzione.

3 Design Pattern

(da integrare con figure di applicabilità in premi e classi che utilizzano quei design pattern) Singleton • Scopo dell'utilizzo: Viene usato il pattern Singleton per le classi che devono avere un'unica istanza durante l'esecuzione dell'applicazione; • Contesto d'utilizzo: Le classi che devono avere un'unica istanza sono: - Invoker - SlideShow

Utility

• ???

Builder

• Scopo dell'utilizzo: Viene usato il pattern Builder per separare la costruzione di un oggetto dalla sua rappresentazione e poter riusare il processo di costruzione per creare rappresentazioni differenti. • Contesto d'utilizzo: Viene utilizzato per il caricamento delle presentazioni

Command

• Scopo dell'utilizzo: Il pattern Command viene usato per separare il codice di un'azione dal codice che richiede l'esecuzione dello stesso. • Contesto d'utilizzo: Viene utilizzato per il caricamento delle presentazioni

Iterator

• Scopo dell'utilizzo: Il pattern Iterator viene usato per fornire un accesso sequenziale agli elementi che formano un oggetto composto senza esporre all'esterno la struttura dell'oggetto. • Contesto d'utilizzo: Viene utilizzato per iterare sugli elementi.

Template Method

• Scopo dell'utilizzo: Il pattern Template Method viene usato per definire la struttura di un algoritmo e lasciare alle sottoclassi la definizione di alcune parti usate. • Contesto d'utilizzo: Viene utilizzato per l'inserimento e la rimozione degli elementi

Strategy

• Scopo dell'utilizzo: Il pattern Strategy viene usato per isolare più algoritmi che svolgono la stessa funzione dal codice che esegue la funzione. • Contesto d'utilizzo: Viene utilizzato per la modifica degli elementi.

4 Descrizione architetturale

4.1 Metodo e formalismi

Si progetterà l'architettura del sistema secondo un approccio top-down, ovvero iniziando da una visione più astratta sul sistema ed aumentando di concretezza nelle iterazioni successive. Si passerà quindi alla definizione dei package e successivamente dei componenti di questi. Infine si andranno a definire le singole classi e interfacce specificando per ognuna:

- Tipo;
- Funzione;
- Classi o interfacce estese;
- Interfacce implementate;
- Relazioni con altre classi.

Verranno quindi illustrati i Design Pattern usati nella progettazione architetturale del sistema rimandano la spiegazione all'appendice (A1).
Per i diagrammi di Package, classi e attività verrà usata la notazione UML 2.(DA AGGIUNGERE INDICE).

4.2 Architettura generale

Il prodotto si presenta suddiviso in tre parti distinte: Model, View e ViewModel. Si è quindi cercato di implementare il design pattern architetturale MVVM in modo da garantire un basso livello di accoppiamento. In figura 1 viene riportato il diagramma dei package, in seguito vengono elencate le componenti dell'applicativo con le relative caratteristiche e funzionalità generali, per una trattazione più approfondita si rimanda alle sezioni specifiche dei componenti.

4.2.1 Model

Contiene la rappresentazione dei dati, l'implementazione dei metodi da applicare ad essi e lo stato di questi ultimi; costituisce il cuore del software e risulta di fatto totalmente indipendente dagli altri due strati.

4.2.2 View

Contiene tutti gli elementi della GUI, comprese le interfacce di comunicazione con le librerie grafiche esterne. Si limita a passare gli input inviati dall'utente allo strato che sta sotto di lei, il Controller, demandandone a quest'ultimo la gestione.

4.2.3 ViewModel

E' il punto di incontro tra la View e il Model: i dati ricevuti da quest'ultimo sono elaborati per essere presentati alla View.

5 Descrizione dei singoli componenti

5.1 View

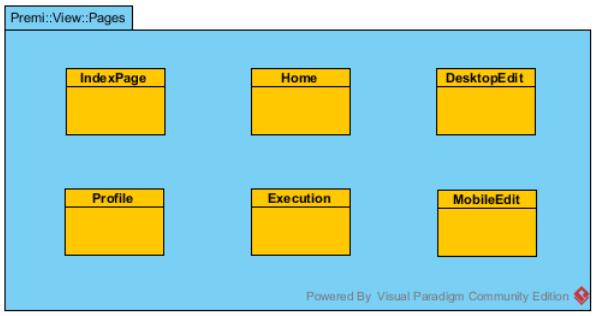

Fig 1: View

Tipo, obiettivo e funzione del componente: questo livello costituisce l'interfaccia del software utilizzabile dagli utenti mediante pagine web.
Relazioni d'uso di altre componenti: il componente è costituito dal package Pages e comunica con il Controller per rendere possibile la gestione del proprio profilo, la gestione delle presentazioni e per controllare i dati in transito per il sistema, dovuti all'interazione dell'utente con lo stesso.

5.1.1 Premi.View.Pages.IndexPage

Tipo, obiettivo e funzione del componente: la classe IndexPage definisce la struttura, e la conseguente visualizzazione, della pagina web che consente ad un utente di effettuare login e registrazione al sistema e di passare alla visualizzazione della classe Loader.
Relazioni d'uso di altre componenti: la classe IndexPage utilizza i metodi messi a disposizione dalla classe [[[[[[[[[[[[[[[[CONTROLLER LOGIN]]]]]]]]]]]]]]]], contenuta nel package Controller, per verificare i dati inseriti durante la fase di autenticazione, per inviare i dati relativi alla registrazione e per visualizzare eventuali errori emersi nella fase di autenticazione/-registrazione.
Attività svolte e dati trattati: la classe definisce la struttura della pagina web che consente agli utenti di autenticarsi e registrarsi al sistema. Essa resta in attesa che un utente inserisca i dati necessari per l'autenticazione o la registrazione al sistema oppure che l'utente decida di andare nella pagina Loader.

5.1.2 Premi.View.Pages.Home

Tipo, obiettivo e funzione del componente: la classe Home definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra ad un utente le presentazioni presenti sul server e i comandi principali di gestione del profilo e gestione presentazioni.
Relazioni d'uso di altre componenti: la classe Home, utilizza i metodi messi a disposizione delle classi [[[[[[[[CONTROLLER ELIMINAZIONE PRESENT.]]]]]]]] per l'eliminazione delle presentazioni dal server e [[[[[[[[[[CONTROLLER SCARICAMENTO MANIFEST]]]]]]]]]] per scaricare una presentazione in locale; utilizza inoltre il metodo [[[[[[[[[CONTROLLER LOGOUT]]]]]]]]] per effettuare il logout, tutti presenti nel package Controller.

Interfacce con e relazioni d'uso e da altre componenti: la classe Home manda a alla pagina Execution l'id della presentazione da eseguire, mentre manda alla pagina DesktopEditing (o mobileEditing) l'id della presentazione da modificare. **Attività svolte e dati trattati**: La classe definisce la struttura della pagina web che consente agli utenti di visualizzare le anteprime delle proprie presentazioni, crearne di nuove, modificarle, eliminarle, scaricarle in locale e andare alla pagina Profile, effettuare il logout.

5.1.3 Premi.View.Pages.Profile

Tipo, obiettivo e funzione del componente: la classe Profile definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra ad un utente i dati del proprio profilo, i propri file caricati e la possibilità di modificarli.
Relazioni d'uso di altre componenti: la classe Profile utilizza i metodi messi a disposzione dalle classi, presenti nel package Controller, [[[[[[[[[CONTROLLER MODIFICA DATI]]]]]]]]] per modificare i propri dati di profilo. Inoltre, vengono utilizzati i metodi delle classi, sempre presenti in Controller, [[[[[[CONTROLLER INSERIMENTO FILE MEDIA]]]]]] per il caricamento di file media nel server, [[[[[[CONTROLLER ELIMINAZ. FILE MEDIA]]]]]] per la loro eliminazione dal server e [[[[[[CONTROLLER RINOMINA FILE MEDIA]]]]]] per rinominarli.
Attività svolte e dati trattati: la classe Profile definisce la struttura della pagina web che consente agli utenti di modificare i propri dati di profilo e gestire i file media caricati nel server.

5.1.4 Premi.View.Pages.Execution

Tipo, obiettivo e funzione del componente: la classe Execution definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra ad un utente l'esecuzione di una presentazione.
Relazioni d'uso di altre componenti: questa classe è gestita dal framework esterno Impress.js utilizzato; utilizza i metodi messi a disposizione delle classi [[[[[[[[[CONTROLLER ESECUZIONE PRESENTAZIONE.]]]]]]]]] per creare la pagina che verrà eseguita da Impress.js.
Attività svolte e dati trattati: La classe definisce la struttura della pagina web che consente agli utenti di eseguire la presentazione spostandosi con la tastiera avanti e indietro, passare al capitolo successivo oppure selezionare un nuovo percorso.

5.1.5 Premi.View.Pages.DesktopEdit

Tipo, obiettivo e funzione del componente: la classe DesktopEdit definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra ad un utente l'editor di modifica di una presentazione.
Relazioni d'uso di altre componenti: mandi principali di gestione del profilo e gestione presentazioni.
Relazioni d'uso di altre componenti: la classe Home, utilizza i metodi messi a disposizione delle classi [[[[[[[[[CONTROLLER CARICA EDITOR]]]]]]]]] per caricare la presentazione da modificare, [[[[[[[[[CONTROLLER INSERIMENTO]]]]]]]]] per l'inserimento di nuovi elementi, [[[[[[[[[CONTROLLER SPOSTAMENTO]]]]]]]]] per lo spostamento di nuovi elementi,

[[[[[[[[CONTROLLER ELIMINAZIONE]]]]]]]] per l'eliminazione elementi e [[[[[[[[CONTROLLER MODIFICA ELEMENTI]]]]]]]] per le modifiche effettuate agli elementi e cambiare il percorso agli elementi con i metodi di [[[[[[[[CONTROLLER MODIFICA ELEMENTI]]]]]]]].

Attività svolte e dati trattati: La classe definisce la struttura della pagina web che consente agli utenti di modificare una presentazione (inserendo, spostando, modificando o eliminando elementi), cambiare il percorso, assegnare bookmark ai frame e inserire elementi scelta.

5.1.6 Premi.View.Pages.MobileEdit

Tipo, obiettivo e funzione del componente: la classe MobileEdit definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra ad un utente mobile l'editor di modifica mobile di una presentazione.
Relazioni d'uso di altre componenti: la classe MobileEdit utilizza i metodi messi a disposzione dalle classi, presenti nel package Controller, [[[[[[[[CONTROLLER CARICA EDITOR MOBILE]]]]]]]] per caricare la presentazione da modificare, [[[[[[[[CONTROLLER INSERIMENTO TESTO]]]]]]]] per l'inserimento di un elemento testuale, [[[[[CONTROLLER MODIFICA TESTO]]]]] per la modifica di un elemento testuale, [[[[[CONTROLLER INSERIMENTO BOOKMARK]]]]] per l'inserimento di un nuovo bookmark, [[[[[CONTROLLER REMOVE BOOKMARK]]]]] per rimuovere un bookmark.
Attività svolte e dati trattati: La classe definisce la struttura della pagina web che consente agli utenti di modificare una presentazione (modificando un elemento testo) e assegnare bookmark ai frame..