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)

3.1 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.

3.2 Utility

- ??????????????????????????????????.

3.3 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.

3.4 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.

3.5 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.

3.6 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.

3.7 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.

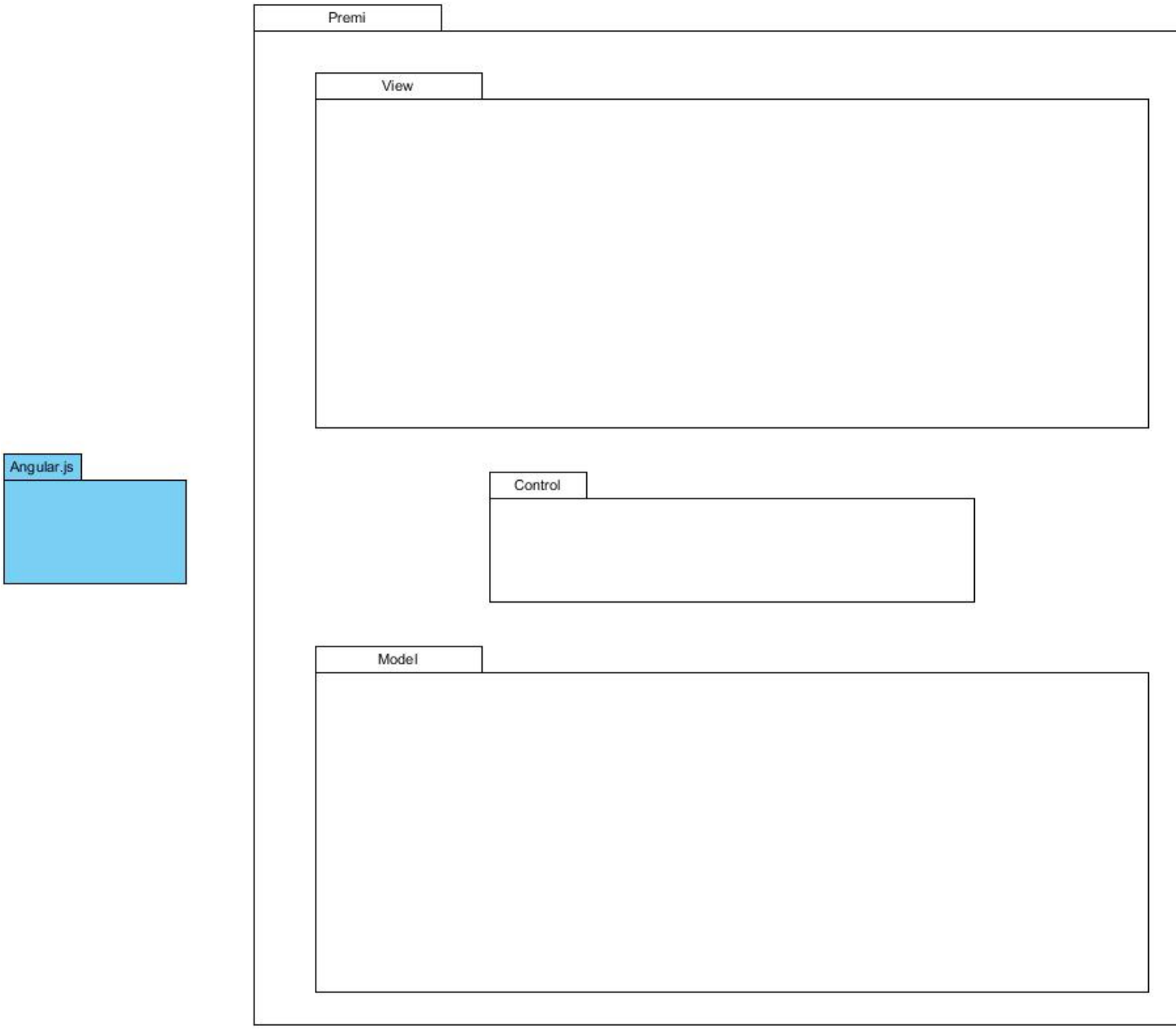

Fig 1: Architettura generale del sistema

5 Descrizione dei singoli componenti

5.1 Model

Tipo, obiettivo e funzione del componente: è la parte Model dell'architettura MVC.
Relazioni d'uso di altre componenti: ??????????????????????.
Package contenuti:

- Premi::Model::Inserimento;
- Premi::Model::Rimozione;
- Premi::Model::Modifica;
- Premi::Model::Command;
- Premi::Model::Invoker;
- Premi::Model::Builder;
- Premi::Model::Presentazione;
- Premi::Model::MongoHandler.

5.1.1 Premi::Controller::Inserimento

Tipo, obiettivo e funzione del componente: All'interno di questo Package viene implementato il Design Pattern template per l'inserimento di nuovi elementi nella presentazione.
Relazioni d'uso di altre componenti:. Il package è in relazione con Premi::Model::Command da cui riceve i segnali e i parametri di inserimento dell'elemento. Inoltre comunica con il package Premi::Model::Presentazione, istanziando gli oggetti delle sottoclassi di SlideShowElement e inserendoli in SlideShow.

5.1.1.1 Premi::Model::Inserimento::Inserter

Tipo, obiettivo e funzione del componente: Classe astratta definita per l'implementazione del Design Pattern template, per l'inserimento di elementi all'interno di una presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteConcreteInsertCommand -> utilizza i metodi messi a disposizione da Inserter e concretizzati dalle sue sottoclassi che a loro volta invocano le funzioni della classe Premi::Model::Presentazione::SlideShow per l'impostazione dei campi relativi.

Interfacce con e relazioni d'uso e da altre componenti: Definisce le operazioni primitive astratte che le classi concrete sottostanti andranno a sovraccaricare e implementa il metodo template che rappresenta lo scheletro dell'algoritmo per l'inserimento di un elemento nella presentazione. È il componente receiver del Design Pattern Command.
Sottoclassi:

- Premi::Model::Inserimento::ConcreteTextInserter;
- Premi::Model::Inserimento::ConcreteFrameInserter;
- Premi::Model::Inserimento::ConcreteSvgInserter;
- Premi::Model::Inserimento::ConcreteImageInserter;
- Premi::Model::Inserimento::ConcreteVideoInserter;
- Premi::Model::Inserimento::ConcreteAudioInserter.

5.1.1.2 Premi::Controller::Presentazione::Inserimento::ConcreteTextInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di inserimento di un elemento testuale all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo testo nella presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per inserire elementi testuali in una presentazione.
Classi ereditate:

- Premi::Model::Inserimento::Inserter.

5.1.1.3 Premi::Model::Inserimento::ConcreteFrameInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di inserimento di un elemento frame all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo Frame nella presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per inserire elementi di tipo frame in una presentazione.
Classi ereditate:

- Premi::Model::Inserimento::Inserter.

5.1.1.4 Premi::Model::Inserimento::ConcreteSvgInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di inserimento di un elemento svg all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo SVG nella presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per inserire elementi svg in una presentazione.
Classi ereditate:

- Premi::Model::Inserimento::Inserter.

5.1.1.5 Premi::Model::Inserimento::ConcreteImageInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di inserimento di un elemento immagine all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo immagine nella presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per inserire elementi di tipo immagine in una presentazione.
Classi ereditate:

- Premi::Model::Inserimento::Inserter.

5.1.1.6 Premi::Model::Inserimento:: ConcreteVideoInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di inserimento di un elemento video all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo video nella presentazione.

Interfacce con e relazioni d'uso e da altre componenti:Viene invocato per inserire elementi di tipo video in una presentazione.
Classi ereditate:

- Premi::Model::Inserimento::Inserter.

5.1.1.7 Premi::Model::Inserimento:: ConcreteAudioInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di inserimento di un elemento di tipo audio all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo audio nella presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per inserire elementi di tipo audio in una presentazione.
Classi ereditate:

- Premi::Model::Inserimento::Inserter.

5.1.2 Premi::Controller::Eliminazione

Tipo, obiettivo e funzione del componente: All'interno di questo Package viene implementato il Design Pattern template per l'eliminazione di elementi dalla presentazione.
Relazioni d'uso di altre componenti:Il package è in relazione con Premi::Model::Command da cui riceve i segnali e i parametri di eliminazione dell'elemento. Inoltre comunica con il package Premi::Model::Presentazione, rimuovendo dall'oggetto di classe SlideShow gli oggetti delle sottoclassi di SlideShowElement e distruggendoli.

5.1.2.1 Premi::Model::Eliminazione::Remover

Tipo, obiettivo e funzione del componente: Classe astratta definita per l'implementazione del Design Pattern template, per l'eliminazione di elementi all'interno di una presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteConcreteRemoveCommand -> utilizza i metodi messi a disposizione da Remover e concretizzati dalle sue sottoclassi che a loro volta invocano le funzioni della classe.

Interfacce con e relazioni d'uso e da altre componenti: Definisce le operazioni primitive astratte che le classi concrete sottostanti andranno a sovraccaricare o definire e implementa il metodo template che rappresenta lo scheletro dell'algoritmo per l'eliminazione di un elemento nella presentazione.
È il componente receiver del Design Pattern Command.
Sottoclassi:

- Premi::Model::Eliminazione::ConcreteTextRemover;
- Premi::Model::Eliminazione::ConcreteFrameRemover;
- Premi::Model::Eliminazione::ConcreteSvgRemover;
- Premi::Model::Eliminazione::ConcreteImageRemover;
- Premi::Model::Eliminazione::ConcreteVideoRemover;
- Premi::Model::Eliminazione::ConcreteAudioRemover.

5.1.2.2 Premi::Model::Eliminazione:: ConcreteTextRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione di un elemento testuale all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteRemoveCommand -> invoca i metodi per eliminare un elemento di tipo testo dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare elementi testuali in una presentazione.
Classi ereditate:

- Premi::Model::Eliminazione::Remover.

5.1.2.3 Premi::Model::Eliminazione:: ConcreteFrameRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione di un elemento di tipo frame all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteRemoveCommand -> invoca i metodi per eliminare un elemento di tipo frame dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare elementi di tipo frame da una presentazione.
Classi ereditate:

- Premi::Model::Eliminazione::Remover.

5.1.2.4 Premi::Model::Eliminazione:: ConcreteSVGtRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione di un elemento di tipo SVG all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteRemoveCommand -> invoca i metodi per eliminare un elemento di tipo SVG dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare elementi SVG da una presentazione.
Classi ereditate:

- Premi::Model::Eliminazione::Remover.

5.1.2.5 Premi::Model::Eliminazione:: ConcreteImageRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione di un elemento immagine all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteRemoveCommand -> invoca i metodi per eliminare un elemento di tipo immagine dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare elementi immagine in una presentazione.
Classi ereditate:

- Premi::Model::Eliminazione::Remover.

5.1.2.6 Premi::Model::Eliminazione:: ConcreteVideoRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione di un elemento di tipo video all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteRemoveCommand -> invoca i metodi per eliminare un elemento di tipo video dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare elementi di tipo video da una presentazione.
Classi ereditate:

- Premi::Model::Eliminazione::Remover.

5.1.2.7 Premi::Model::Eliminazione:: ConcreteAudioRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione di un elemento di tipo audio all'interno di una presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::ConcreteRemoveCommand -> invoca i metodi per eliminare un elemento di tipo audio dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare elementi di tipo audio da una presentazione.
Classi ereditate:

- Premi::Model::Eliminazione::Remover.

5.2 Controller

Tipo, obiettivo e funzione del componente: fanno parte di questo livello i package che gestiscono i segnali e le chiamate effettuati dalla view verso la struttura dati.
Relazioni d'uso di altre componenti: il componente è costituito dai package Presentazione e Utente, comunica con il Model per rendere possibile la gestione del profilo e la gestione delle presentazioni da parte dell'utente.
Package contenuti:

- Premi::Controller::Presentazione
- Premi::Controller::Utente

5.2.1 Premi::Controller::Presentazione

Tipo, obiettivo e funzione del componente: fanno parte di questo package tutte le classi di controller con cui interagiscono le pagine dedicate alla gestione delle presentazioni.
Relazioni d'uso di altre componenti: il package comunica con la view ricevendo chiamate da Premi::View::Pages::MobileEdit, Premi::View::Pages::DesktopEdit, Premi::View::Pages::Execution e Premi::View::Pages::Home. Comunica, invece, con il model inviando segnali e chiamate ai package Premi::Model::Inserimento, Premi::Model::Eliminazione, Premi::Model::Modifica, Premi::Model::Command, Premi::Model::Builder e alle classi Premi::Model::Invoker, Premi::Model::MongoHan

5.2.1.1 Premi::Controller::Presentazione::EditController

Tipo, obiettivo e funzione del componente: Lo scopo di questa classe è di gestire i segnali delle pagine Premi::View::Pages::DesktopEdit e Premi::View::Pages::MobileEdit verso il model.
Relazioni d'uso di altre componenti:

- Premi.View.Pages.DesktopEdit e Premi.View.Pages.MobileEdit -> costruiscono EditController, ne invocano i metodi passando i parametri degli oggetti modificati;
- Premi::Model::Command <- EditController costruisce un comando e lo dà in pasto a Premi::Model:Invoker;
- Premi::Model::Invoker <- EditController costruisce l'oggetto di classe Invoker. Invoca il metodo execute() di Invoker, passando come paramentro un oggetto di classe Command oppure invoca il metodo unexecute() di Invoker.

Interfacce con e relazioni d'uso e da altre componenti: La pagina DesktopEdit o la pagina MobileEdit invia a EditController un segnale comunicando l'avvenuta modifica o la rimozione di un elemento della presentazione, oppure l'inserimento di un nuovo elemento. EditController istanzia un oggetto di classe Premi::Model::Command e lo dà in pasto a Premi::Model::Invoker. Eventualmente EditController può semplicemente annullare il comando appena eseguito invocando il metodo unexecute di Invoker.

5.2.1.2 Premi::Controller::Presentazione::HomeController

Tipo, obiettivo e funzione del componente: Lo scopo di questa classe è di gestire i segnali della pagina Premi::View::Pages::Home verso la struttura dati.
Relazioni d'uso di altre componenti:

- Premi.View.Pages.Home -> costruisce HomeController, ne invoca i metodi passando i parametri dell'utente;
- Premi::Model::MongoHandler <- HomeController invoca un metodo di MongoHandler che restituisce l'elenco dei titoli delle presentazioni dell'utente;

Interfacce con e relazioni d'uso e da altre componenti: La pagina Home costruisce HomeController e richiede l'elenco delle presentazioni dell'utente.

5.3 View

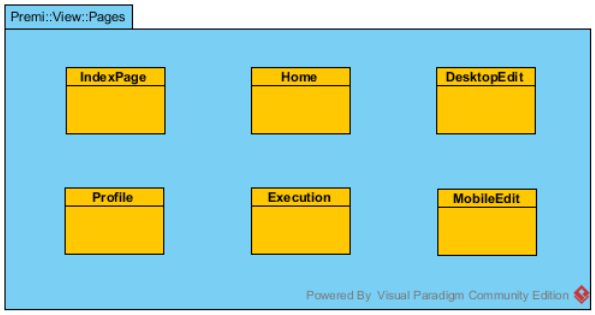

Fig 2: View

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.

5.3.1 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.

5.3.2 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 seguenti classi contenute nel package Controller:

NE PRESENT.] per l'eliminazione delle presentazioni dal server;

TO MANIFEST]]]]]]]] per scaricare una presentazione in locale;

LER LOGOUT]]]]]]] per effettuare il logout.

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.

5.3.3 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 disposizione dalle seguenti classi presenti nel package Controller:

ODIFICA DATI]]]]]]] per modificare i propri dati di profilo;

O FILE MEDIA]]]]] per il caricamento di file media nel server;

. FILE MEDIA]]]]] per la loro eliminazione dal server;

A 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.

5.3.4 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.

5.3.5 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 dalle seguenti classi presenti nel package Controller:

RICA EDITOR]]]]]]] per caricare la presentazione da modificare;

NSERIMENTO]]]]]]] per l'inserimento di nuovi elementi;

POSTAMENTO]]]]]]] per lo spostamento di nuovi elementi;

LIMINAZIONE]]]]]]] per l'eliminazione elementi;

CA ELEMENTI]]]]]]] per le modifiche effettuate agli elementi ;

CA PERCORSO]]]]]]] per cambiare il percorso della presentazione.

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.

5.3.6 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 disposizione dalle seguenti classi presenti nel package Controller:

ITOR MOBILE]]]]]]] per caricare la presentazione da modificare;

IENTO TESTO]]]]]] per l'inserimento di un elemento testuale;

DIFICA TESTO]]]] per la modifica di un elemento testuale;

O BOOKMARK]]]] per l'inserimento di un nuovo bookmark;

E 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..