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/;
- MEAN: http://www.mean.io/; **MEAN Web Development, Amos Q. Haviv, 2014**;

- MongoDB: http://docs.mongodb.org/manual/;
- Angular.js: https://docs.angularjs.org/tutorial;
- Express.js: http://expressjs.com/;
- Node.js: https://nodejs.org/documentation/;
- 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**: È 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 richiede informazioni da recuperare in 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

jQuery è una libreria Javascript cross-platform, disegnata per semplificare lo scripting di HTML lato-client. È la libreria Javascript più popolare al momento; è un software libero ed open-source.
Il nucleo di jQuery è una libreria di manipolazione DOM (Document Object Model). DOM è una struttura ad albero che rappresenta tutti gli elementi di una pagina web e jQuery rende la ricerca, selezione e manipolazione di questi elementi DOM semplice e conveniente. I vantaggi

nell'uso di jQuery sono l'incoraggiamento alla separazione di Javascript ed HTML, la brevità e la chiarezza, l'eliminazione di incompatibilità cross-browser, l'estendibilità.

2.4 MEAN

MEAN è uno stack di software Javascript, libero ed open source per costruire siti web dinamici ed applicazioni web. È una combinazione di MongoDB, Express.js ed Angular.js, eseguita su Node.js.

2.4.1 MongoDB

MongoDB è un database NoSQL open source orientato ai documenti, facilmente scalabile e ad alte prestazioni. Si allontana dalla struttura tradizionale basata su tabelle dei database relazionali, in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON); questo rende l'integrazione di dati più semplice e facile in alcuni tipi d'applicazioni. È un software libero ed open-source.

2.4.2 Express.js

Express.js è un framework per applicazioni web Node.js, disegnato per costruire applicazioni web single-page, multi-page o ibride. È costruito sopra il modulo Connect di Node.js e fa uso della sua architettura middleware; le sue caratteristiche permettono di estendere Connect per permettere una gran varietà di casi d'uso comuni alle applicazioni web, come l'inclusione di HTML template engine modulari, l'estensione del response object per supportare vari formati di output dei dati, un sistema di routing e molto altro.

2.4.3 AngularJS

AngularJS, comunemente detto Angular, è un framework per applicazioni web, open-source, manutetenuto da Google e da una comunità di sviluppatori e corporations. Mira a semplificare lo sviluppo ed il test di applicazioni single-page fornendo un framework per l'architettura model-view-controller lato-client.
La libreria AngularJS come prima cosa legge la pagina HTML, che ha al suo interno degli attributi tag personalizzati; Angular interpreta questi attributi come direttive per legare parti di input o di output della pagina ad un modello che è rappresentato da variabili Javascript standard. Il valore di queste variabili Javascript può essere imporstato manualmente all'interno del codice, oppure ricavato da risorse JSON statiche o dinamiche.

2.4.4 Node.js

Node.js è un'ambiente di esecuzione open source e cross-platform per applicazioni lato server; le applicazioni Node.js sono scritte in linguaggio Javascript. Node.js fornisce un'architettura orientata agli eventi (event-driven) ed un'API (Application Programming Interface) con I/O non bloccante, che ottimizza il throughput e la scalabilità e permette lo sviluppo di veloci server web in Javascript.
Node.js usa il motore Javascript V8 di Google per eseguire codice, ed una larga percentuale dei moduli base è scritta in Javascript. Node.js contiene al suo interno una libreria che permette

alle applicazioni di agire come server Web senza la necessità di software come Apache HTTP Server o IIS.

2.5 Impress.js

Impress.js è 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.

Fig 1: Fig1

3 Design Pattern

(da integrare con figure di applicabilità in premi e classi che utilizzano quei design pattern)

3.1 MVC

- **Scopo dell'utilizzo**: è stato scelto il pattern MVC per separare la logica dell'applicazione dalla rappresentazione grafica;
- **Contesto d'utilizzo**: Il pattern MVC viene utilizzato per l'architettura generale dell'applicazione.// Ogni modifica effettuata dall'utente sulla View viene inviata al Controller che invoca i metodi delle classi presenti nel Model.

3.2 Singleton

- **Descrizione**: è un design pattern di tipo creazionale utilizzato per imporre ad alcune classi di avere solamente un'istanza. Questo design pattern garantisce la consistenza dello stato dell'applicazione che si andrà a sviluppare.
  Quando una classe implementa il pattern Singleton, la classe stessa si occupa di controllare il numero di istanze costruite (di solito tramite campi dati statici) e di fornire l'accesso ad esse tramite metodi appositi.

- **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.

3.2.1 Premi::Model::Command::Invoker

La prima volta in cui avviene una modifica al piano della presentazione Premi::Controller::Edit invoca il metodo Invoker::getInstance() che, tramite costruttore privato, costruisce Invoker e lo restituisce al controller che può quindi dargli in pasto i comandi per l'implementazione del Design Pattern Command.

3.2.2 Premi::Model::Presentazione::SlideShow

La prima volta in cui viene caricata una presentazione, Premi::Model::Presentazione::Loader invoca il metodo SlideShow::getInstace() che, tramite costruttore privato, costruisce Sli-

deShow.

3.3 Utility

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

3.4 Builder

- **Descrizione**:Il design pattern Builder separa la costruzione di un oggetto complesso dalla sua rappresentazione cosicché il processo di costruzione stesso possa creare diverse rappresentazioni.
  L'algoritmo per la creazione di un oggetto complesso è indipendente dalle varie parti che costituiscono l'oggetto e da come vengono assemblate. Ciò ha l'effetto immediato di rendere più semplice la classe, permettendo a una classe builder separata di focalizzarsi sulla corretta costruzione di un'istanza e lasciando che la classe originale si concentri sul funzionamento degli oggetti. Un builder permette anche di costruire un oggetto passo-passo, cosa che si può verificare quando si fa il parsing di un testo o si ottengono i parametri da un'interfaccia interattiva.

- **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 e degli oggetti in essi presenti.

3.4.1 Premi::Model::Builder

Il package Premi::Model::Builder implementa il Design Pattern Builder. Premi::Model::Presentazione::Load costruisce il Premi::Model::Builder::Director che ricopre il ruolo di director del Design Pattern. Loader passa a Director i parametri di istanziazione dell'oggetto di una sottoclasse di Premi::Model::Presentazione::SlideShowElement. Director invoca il costruttore della classe concreta di AbstractBuilder per istanziare un oggetto della sottoclasse di SlideShowElement, invoca quindi i metodi di build per settarne i campi.

3.5 Command

- **Descrizione**: viene utilizzato quando c'è la necessità di disaccoppiare l'invocazione di un comando dai suoi dettagli implementativi, separando colui che invoca il comando da colui che esegue l'operazione.

  Tale operazione viene realizzata attraverso questa catena: Client->Invocatore->Ricevitore

  Il Client non è tenuto a conoscere i dettagli del comando ma il suo compito è solo quello di chiamare il metodo dell' Invocatore che si occuperà di intermediare l'operazione. L'Invocatore ha l'obiettivo di incapsulare, nascondere i dettagli della chiamata come nome del metodo e parametri. Il Ricevitore utilizza i parametri ricevuti per eseguire l'operazione

- **Scopo dell'utilizzo**: si è scelto di utilizzare il pattern Command perché poter accodare o mantenere uno storico delle operazioni e gestire operazioni cancellabili;
- **Contesto d'utilizzo**: viene utilizzato in fase di modifica delle presentazioni.

3.5.1 Premi::Model::Command

Il package Premi::Model::Command implementa il pattern Command, tuttavia il client è esterno al package ed è individuabile nella classe Premi::Controller::Presentazione::Edit, che invoca il costruttore delle sottoclassi di Premi::Model::Command::AbstractCommand e dà l'oggetto creato in pasto a Premi::Model::Command::Invoker, che rappresenta, appunto, la componente invoker del pattern e che mette l'oggetto della sottoclasse di AbstractCommand in un contenitore denominato undo, invoca quindi il metodo Invoker::execute() che a sua volta esegue concretamente il comando.
Premi::Controller::Presentazione::Edit può invocare il metodo unexecute() di Invoker che a sua volta invoca il metodo AbstractCommand::undoCommand() nell'ultimo oggetto inserito nel membro contenitore undo. Questo metodo esegue le operazioni necessarie per annullare tutte le modifiche apportate dal comando. Quindi Invoker toglie il comando dal contenitore undo e lo inserisce nel contenitore redo. Quando Premi::Controller::Presentazione::Edit invoca il metodo Invoker::execute(), l'oggetto Invoker esegue il comando e lo sposta nuovamente dal membro contenitore redo e lo mette nel membro undo.

3.6 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.

3.7 Template Method

- **Descrizione**: il Design Pattern Template Method è utilizzato per descrivere un algoritmo in cui alcuni passi possono essere sovrascritti da sottoclassi al fine di differenziare il comportamento e allo stesso tempo assicurare che l'algoritmo sovrastante sia sempre seguito.

  Prima viene creata una classe che fornisce i passi base di un algoritmo. Questi passi sono implementati usando metodi astratti. Successivamente, le sottoclassi cambiano i metodi astratti per implementare l'algoritmo. Così facendo l'algoritmo generale è mantenuto valido, ma i passi concreti possono essere cambiati dalle sottoclassi.

  Template Method viene utilizzato frequentemente nei linguaggi orientati agli oggetti. Quando si ricorre al polimorfismo in generale, questo design pattern potrebbe essere definito come sua naturale conseguenza. Questo perché un metodo che invoca una funzione astratta o polimorfa è semplicemente il motivo d'essere del metodo astratto o polimorfo;

- **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.

3.7.1 Premi::Model::Inserimento

Le classi del package Premi::Model::Inserimento implementano il pattern Template. La classe astratta Premi::Model::Inserimento::Inserter definisce i metodi astratti per l'inserimento di elementi nella presentazione e descrive i passi dell'algoritmo comuni per l'inserimento di tutti i tipi di elementi. Le sottoclassi di Inserter specificano i metodi:

- createElement che invoca il costruttore della sottoclasse di Premi::Model::Presentazione::SlideSh a cui appartiene l'elemento da inserire;
- insertElement che inserisce l'oggetto nel membro contenitore all'interno dell'oggetto di classe Premi::Model::Presentazione::SlideShow.

In maniera del tutto simile al package Inserimento è organizzato il package Premi::Model::Eliminazione

3.8 Strategy

- **Descrizione**: Strategy è un Design Pattern che permette che il comportamento di un algoritmo sia deciso a runtime. Il pattern Strategy definisce una famiglia di algoritmi, incapsula ogni algoritmo e rende gli algoritmi intercambiabili all'interno di quella famiglia. Strategy permette all'algoritmo di cambiare indipendentemente dal client che lo utilizza;
- **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.

3.8.1 Premi::Model::Modifica

Il package Premi::Model::Modifica implementa il design pattern Strategy. Premi::Model::Modifica::Ed fornisce un metodo astratto editElement che viene definito in modo diverso in ognuna delle sue sottoclassi e invocato al momento della costruzione. In maniera del tutto simile a Premi::Model::Modifica è organizzato il package Premi::Model::Caricamento.

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.

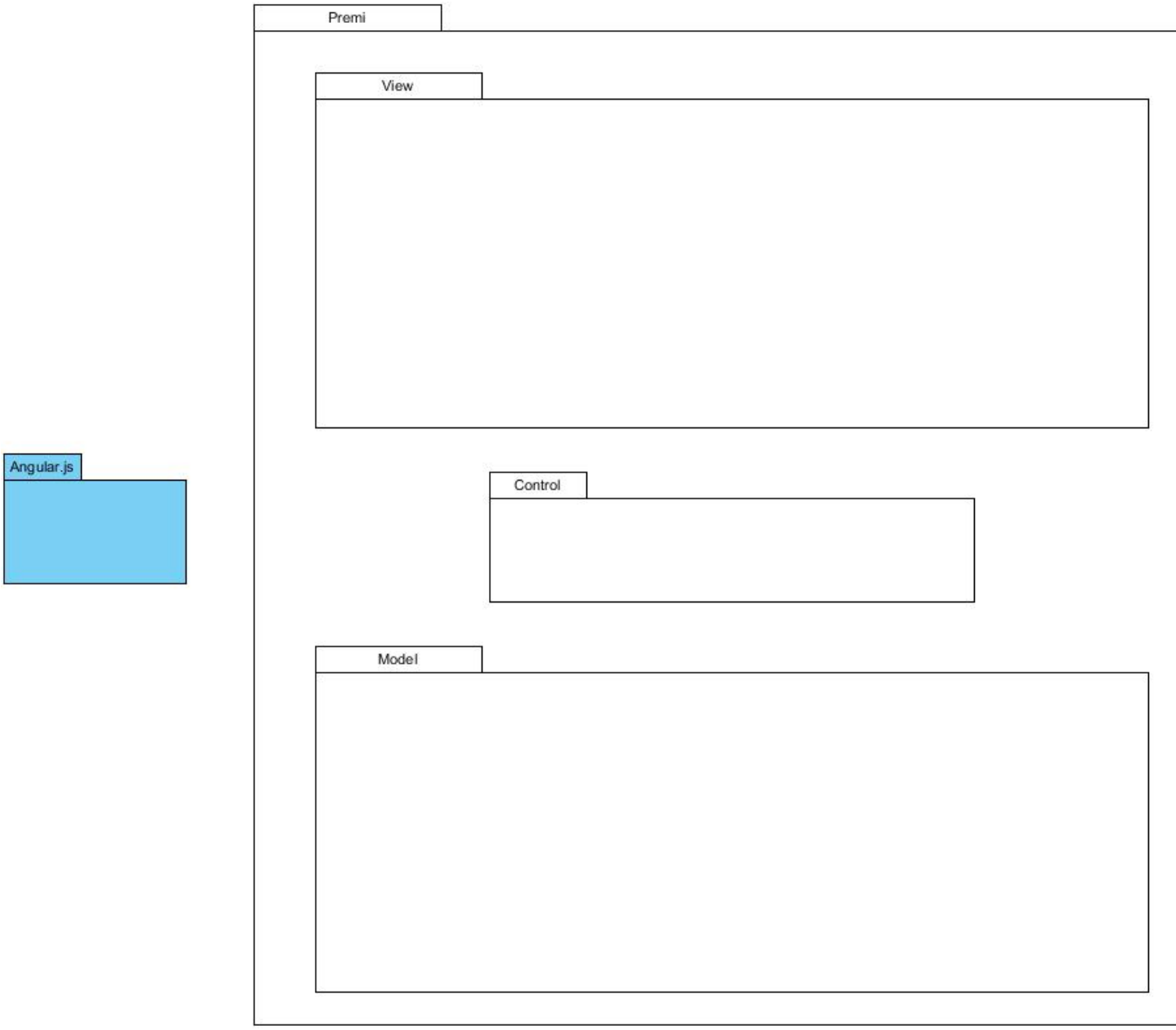

Fig 2: Architettura generale del sistema

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::Model::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.
- Premi::Model::Eliminazione::ConcreteBackgroundInserter.

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

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di creazione e 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::Presentazione::Text <- costruisce un oggetto di classe Text.
- Premi::Model::Command::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::Presentazione::Frame <- costruisce un oggetto di classe Frame.
- Premi::Model::Command::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::Presentazione::SVG <- costruisce un oggetto di classe SVG.
- Premi::Model::Command::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::Presentazione::Image <- costruisce un oggetto di classe Image.
- Premi::Model::Command::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::Presentazione::Video <- costruisce un oggetto di classe Video.
- Premi::Model::Command::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::Presentazione::Audio <- costruisce un oggetto di classe Audio.
- Premi::Model::Command::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.

5.1.1.8 Premi::Controller::Presentazione::Inserimento::ConcreteBackgroundInserter

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di creazione e inserimento di un elemento di classe Background in una Presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::Presentazione::Background <- invoca il metodo getInstance di classe Background.
- Premi::Model::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo sfondo nella presentazione.

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

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

5.1.2 Premi::Model::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;
- Premi::Model::Eliminazione::ConcreteBackgroundRemover.

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::Command::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::Command::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::Command::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::Command::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::Command::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::Command::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.

5.1.2.8 Premi::Model::Eliminazione:: ConcreteBackgroundRemover

Tipo, obiettivo e funzione del componente: Classe che implementa un algoritmo di eliminazione dello sfondo dellaa presentazione. È uno dei componenti concreti del Design Pattern Template.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteRemoveCommand -> invoca i metodi per eliminare lo sfondo dalla presentazione.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per eliminare lo sfondo dlla presentazione.
Classi ereditate:

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

5.1.3 Premi::Model::Modifica

Tipo, obiettivo e funzione del componente: All'interno di questo Package viene implementato il Design Pattern strategy per la modifica di elementi della 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 modifica dell'elemento. Inoltre comunica con il package Premi::Model::Presentazione, modificando nell'oggetto di classe SlideShow gli oggetti delle sottoclassi di SlideShowElement.

5.1.3.1 Premi::Model::Modifica::Editor

Tipo, obiettivo e funzione del componente: Interfaccia per la componente strategy del Design Pattern Strategy per la selezione dell'algoritmo di modifica della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca i costruttori delle sottoclassi di Editor;
- Premi::Model::Presentazione::SlideShow <- scorre gli elementi dei membri contenitori all'interno di SlideShow per trovare l'elemento da modificare;
- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi dell'elemento.

Interfacce con e relazioni d'uso e da altre componenti: Permette di selezionare dinamicamente ed in modo estensibile l'algoritmo di modifica della presentazione.
Sottoclassi:

- Premi::Model::Modifica::Editor::EditorPosition;
- Premi::Model::Modifica::Editor::EditorSize;
- Premi::Model::Modifica::Editor::EditorContent;
- Premi::Model::Modifica::Editor::EditorRotate;
- Premi::Model::Modifica::Editor::EditorColor;
- Premi::Model::Modifica::Editor::EditorShape.

5.1.3.2 Premi::Model::Modifica::EditorPosition

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Strategy per la modifica dei campi inerenti alla posizione di un elemento della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca il costruttore di EditorPosition;
- Premi::Model::Presentazione::SlideShow<- scorre gli elementi dei membri contenitori all'interno di SlideShow per trovare l'elemento da modificare;

- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi alla posizione dell'elemento.

Interfacce con e relazioni d'uso e da altre componenti:Premi::Model::Command::ConcreteEditComm invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorPosition invocherà quindi i metodi di modifica delle coordinate forniti all'interno della sottoclasse di Premi::Model::Presentazione::SlideShowElement di cui fa parte l'oggetto da modificare.
Classi ereditate:

- Premi::Model::Modifica::Editor.

5.1.3.3 Premi::Model::Modifica::EditorSize

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Strategy per la modifica dei campi inerenti alla dimensione di un elemento della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca il costruttore di EditorSize;
- Premi::Model::Presentazione::SlideShow<- scorre gli elementi dei membri contenitori all'interno di SlideShow per trovare l'elemento da modificare;
- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi alla dimensione dell'elemento.

Interfacce con e relazioni d'uso e da altre componenti:Premi::Model::Command::ConcreteEditComm invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorSize invocherà quindi i metodi di modifica delle dimensioni forniti all'interno della sottoclasse di Premi::Model::Presentazione::SlideShowElement di cui fa parte l'oggetto da modificare.
Classi ereditate:

- Premi::Model::Modifica::Editor.

5.1.3.4 Premi::Model::Modifica::EditorRotate

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Strategy per la modifica dei campi inerenti all'inclinazione di un elemento della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca il costruttore di EditorRotate;
- Premi::Model::Presentazione::SlideShow<- scorre gli elementi dei membri contenitori all'interno di SlideShow per trovare l'elemento da modificare;
- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi all'inclinazione dell'elemento.

Interfacce con e relazioni d'uso e da altre componenti:Premi::Model::Command::ConcreteEditComm
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorPosition invocherà quindi i metodi di modifica dell'inclinazione forniti all'interno della sottoclasse di Premi::Model::Presentazione::SlideShowElement di cui fa parte l'oggetto da modificare.
Classi ereditate:

- Premi::Model::Modifica::Editor.

5.1.3.5 Premi::Model::Modifica::EditorContent

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Strategy per la modifica dei campi inerenti al contenuto di un elemento di tipo testuale della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca il costruttore di EditorContent;
- Premi::Model::Presentazione::SlideShow<- scorre gli elementi del membro contenitore all'interno di SlideShow per trovare l'elemento testuale da modificare;
- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi alla contenuto dell'elemento testuale.

Interfacce con e relazioni d'uso e da altre componenti:Premi::Model::Command::ConcreteEditComm
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorContent invocherà quindi i metodi di modifica del contenuto forniti all'interno della classe Premi::Model::Presentazione::T
Classi ereditate:

- Premi::Model::Modifica::Editor.

5.1.3.6 Premi::Model::Modifica::EditorShape

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Strategy per la modifica dei campi inerenti alla forma di un elemento SVG della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca il costruttore di EditorShape;
- Premi::Model::Presentazione::SlideShow<- scorre gli elementi dei membri contenitori all'interno di SlideShow per trovare l'elemento SVG da modificare;
- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi alla forma dell'elemento SVG.

Interfacce con e relazioni d'uso e da altre componenti:Premi::Model::Command::ConcreteEditComm
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorShape invocherà quindi i metodi di modifica della forma forniti all'interno della classe Premi::Model::Presentazione::SV
Classi ereditate:

- Premi::Model::Modifica::Editor.

5.1.3.7 Premi::Model::Modifica::EditorColor

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Strategy per la modifica dei campi inerenti al colore di un elemento SVG della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Command::ConcreteEditCommand -> Invoca il costruttore di EditorColor;
- Premi::Model::Presentazione::SlideShow<- scorre gli elementi del membro contenitore all'interno di SlideShow per trovare l'elemento SVG da modificare;
- Premi::Model::Presentazione::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi alla forma dell'elemento SVG.

Interfacce con e relazioni d'uso e da altre componenti:Premi::Model::Command::ConcreteEditComm
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorShape invocherà quindi i metodi di modifica della forma forniti dalla classe Premi::Model::Presentazione::SVG.
Classi ereditate:

- Premi::Model::Modifica::Editor.

5.1.4 Premi::Model::Command

Tipo, obiettivo e funzione del componente:All'interno di questo Package viene implementato il Design Pattern command, utile per la gestione di funzioni di annullamento e ripristino.
Relazioni d'uso di altre componenti:. All'interno del Model, il package è in relazione con Premi::Model::Inserimento, Premi::Model::Eliminazione e Premi::Model::Modifica. Il package comunica, inoltre, con il controller, infatti le sue classi sono generate da Premi::Controller::Presentazione::Ed

5.1.4.1 Premi::Model::Invoker

Tipo, obiettivo e funzione del componente: È componente invoker del Design Pattern Command, il suo scopo è tenere traccia delle modifiche atomiche apportate alla presentazione (modifica di elemento, eliminazione di elemento e inserimento di elemento) per poter implementare le funzioni di annulla/ripristina.
Relazioni d'uso di altre componenti:

- Premi::Controller::MobileEdit->crea un oggetto di una sottoclasse di Premi::Model::Command::Abstr passandolo all'Invoker che lo esegue e lo inserisce nello stack "undo", richiama il metodo che svuota lo stack "redo".
  Può inoltre invocare il metodo "unexecute" dell'Invoker che provvede a richiamare il metodo undo del comando sulla cima dello stack "undo" e a spostarlo quindi nello stack "redo". Alternativamente invoca il metodo "redo" dell'Invoker che provvede a eseguire il comando sulla cima dello stack "redo" e a spostarlo quindi nello stack "undo";
- Premi::Controller::DesktopEdit->si comporta in modo analogo a MobileEdit;

- Premi::Model::Command::AbstractCommand <- Invoker invoca il metodo execute() dell'oggetto della sottoclasse di AbstractCommand. Alternativamente invoca il metodo undo().

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per effettuare le operazioni di modifica alla presentazione, a sua volta invoca una classe derivata da Premi::Model::Command per eseguire materialmente il comando. Quando un comando viene eseguito, Invoker lo salva in un array $undo[ ], insieme ai parametri necessari a riportare la presentazione allo stato precedente.

5.1.4.2 Premi::Model::Command::AbstractCommand

Tipo, obiettivo e funzione del componente: È interfaccia astratta del Design Pattern Command, è classe base per i comandi di modifica, inserimento ed eliminazione.
Relazioni d'uso di altre componenti:

- Premi::Model:: Invoker -> esegue materialmente il comando, richiamandone i metodi di esecuzione; inoltre provvede ad annullare l'ultima operazione

Interfacce con e relazioni d'uso e da altre componenti:Viene utilizzata per applicare un generico parametro di trasformazione ad un oggetto della presentazione, questo parametro verrà poi specificato dalle classi concrete.
Sottoclassi:

- Premi::Model::Command::ConcreteInsertCommand;
- Premi::Model::Command::ConcreteRemoveCommand;
- Premi::Model::Command::ConcreteEditCommand.

5.1.4.3 Premi::Model::Command::ConcreteInsertCommand

Tipo, obiettivo e funzione del componente: È classe concreta del Design Pattern Command, rappresenta un comando per inserire un nuovo elemento nell'oggetto presentazione.
Relazioni d'uso di altre componenti:

- Premi::Controller::Presentazione::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::Inserimento::Inserter <- invoca la classe concreta del template per l'inserimento di un elemento.

Interfacce con e relazioni d'uso e da altre componenti: Viene utilizzata per gestire i Signal riguardanti l'inserimento di un nuovo elemento ed invocare i corretti metodi del Model;
Classi ereditate:

- Premi::Model::Command::AbstractCommand.

5.1.4.4 Premi::Model::Command::ConcreteRemoveCommand

Tipo, obiettivo e funzione del componente: È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento dall'oggetto SlideShow.
Relazioni d'uso di altre componenti:

- Premi::Controller::Presentazione::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::Eliminazione::Remover <- invoca la classe concreta del template per l'eliminazione di un elemento.

Interfacce con e relazioni d'uso e da altre componenti: Viene utilizzata per gestire i Signal riguardanti l'eliminazione di un elemento ed invocare i corretti metodi del Model.
Classi ereditate:

- Premi::Model::Command::AbstractCommand.

5.1.4.5 Premi::Model::Command::ConcreteEditCommand

Tipo, obiettivo e funzione del componente: È classe concreta del Design Pattern Command, rappresenta un comando per modificare un elemento elemento nell'oggetto SlideShow.
Relazioni d'uso di altre componenti:

- Premi::Controller::Presentazione::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::Modifica::Editor <- invoca la classe concreta del design pattern Strategy per la modifica di un elemento.

Interfacce con e relazioni d'uso e da altre componenti: Viene utilizzata per gestire i Signal riguardanti la modifica di un nuovo elemento ed invocare i corretti metodi del Model;
Classi ereditate:

- Premi::Model::Command::AbstractCommand.

5.1.5 Premi::Model::Presentazione

Tipo, obiettivo e funzione del componente:Di questo package fanno parte le classi degli elementi della presentazione e la classe che definisce la presentazione stessa. Sì tratta del package centrale del software.
Relazioni d'uso di altre componenti:.Premi::Model::Presentazione è in comunicazione con

- Premi::Model::Inserimento, i cui oggetti durante la modifica della presentazione istanziano oggetti di tipo SlideShowElement;
- Premi::Model::Eliminazione, i cui oggetti rimuovono da SlideShow gli oggetti di tipo SlideShowElement e li distruggono;

- Premi::Model::Modifica, i cui oggetti invocano metodi degli oggetti SlideShowElement che ne impostano i campi;
- Premi::Controller::Presentazione::EditController o Premi::Controller::Presentazione::ExecutionContr invocano il costruttore di Loader;
- al momento del caricamento della presentazione gli oggetti di Premi::Model::Presentazione::SlideShow sono invece costruiti dalle classi del package Premi::Model::Builder.

5.1.5.1 Premi::Model::Presentazione::SlideShow

Tipo, obiettivo e funzione del componente: Classe implementata con il Design Pattern Singleton, contiene tutte le impostazioni della presentazione caricata, gli oggetti in essa presenti e i metodi per settarli o inserirne di nuovi. Gli oggetti della presentazione si trovano all'interno di contenitori divisi per classe. Tramite un iteratore è possibile scorrere detti contenitori.
Relazioni d'uso di altre componenti:

- Premi::Model::Presentazione::Loader -> invoca il metodo SlideShow::getInstance(), che a sua volta invoca il costruttore privato di SlideShow.

Interfacce con e relazioni d'uso e da altre componenti:Viene utilizzata dalla view tramite Premi::Controller::EditController e Premi::Controller::ExecutionController per creare gli oggetti html sia in fase di esecuzione che in fase di modifica. È la classe principale del software.

5.1.5.2 Premi::Model::Presentazione::SlideShowElement

Tipo, obiettivo e funzione del componente: Gli oggetti della classe SlideShowElement rappresentano gli elementi della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::Inserter-> invoca il costruttore delle sottoclassi di SlideShowElement e li inserisce nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow;
- Premi::Model::Modifica:Editor -> gli oggetti delle sue sottoclassi richiamano le funzioni delle sottoclassi di SlideShowElement che gestiscono l'impostazione dei campi dati;
- Premi::Model::Eliminazione::Remover -> gli oggetti delle sue sottoclassi rimuovono dai contenitori di SlideShow gli oggetti di classe SlideShowElement e ne richiamano i distruttori.

Interfacce con e relazioni d'uso e da altre componenti: Premi::Model::Inserimento::Inserter instanzia oggetti di sottoclassi di SlideShowElement e li inserisce nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow
Sottoclassi:

- Premi::Model::Presentazione:: Text;
- Premi::Model::Presentazione:: Frame;

- Premi::Model::Presentazione:: Image;
- Premi::Model::Presentazione::SVG;
- Premi::Model::Presentazione::Audio;
- Premi::Model::Presentazione::Video;
- Premi::Model::Presentazione::Background.

5.1.5.3 Premi::Model::Presentazione::Text

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Text rappresentano gli elementi di tipo testuale della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteTextInserter -> invoca il costruttore di Text e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazione::
- Premi::Model::Eliminazione::ConcreteTextRemover -> rimuove l'oggetto Text dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
- Premi::Model::Modifica::ConcreteTextEditor -> invoca i metodi che impostano i campi dell'oggetto Text.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Text vengono istanziati da Premi::Model::Inserimento::ConcreteTextInserter o da Premi::Model::Builder::ConcreteT e inseriti nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.5.4 Premi::Model::Presentazione::Frame

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Frame rappresentano gli elementi di tipo frame della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteFrameInserter -> invoca il costruttore di Frame e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazio
- Premi::Model::Eliminazione::ConcreteFrameRemover -> rimuove l'oggetto Frame dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
- Premi::Model::Modifica::ConcreteFrameEditor -> invoca i metodi che impostano i campi dell'oggetto Frame.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Frame vengono istanziati da Premi::Model::Inserimento::ConcreteFrameInserter o da Premi::Model::Builder::Concrete e inseriti nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.5.5 Premi::Model::Presentazione::Image

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Image rappresentano gli elementi di tipo immagine della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteImageInserter -> invoca il costruttore di Image e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazio
- Premi::Model::Eliminazione::ConcreteImageRemover -> rimuove l'oggetto Image dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
- Premi::Model::Modifica::ConcreteFrameEditor -> invoca i metodi che impostano i campi dell'oggetto Image.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Image vengono istanziati da Premi::Model::Inserimento::ConcreteImageInserter o da Premi::Model::Builder::Concrete e inseriti nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.5.6 Premi::Model::Presentazione::SVG

Tipo, obiettivo e funzione del componente: Gli oggetti della classe SVG rappresentano gli elementi di tipo SVG della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteSVGInserter -> invoca il costruttore di SVG e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazione::
- Premi::Model::Eliminazione::ConcreteImageRemover -> rimuove l'oggetto SVG dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
- Premi::Model::Modifica::ConcreteFrameEditor -> invoca i metodi che impostano i campi dell'oggetto SVG.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe SVG vengono istanziati da Premi::Model::Inserimento::ConcreteSVGInserter o da Premi::Model::Builder::ConcreteS e inseriti nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.5.7 Premi::Model::Presentazione::Audio

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Audio rappresentano gli elementi di tipo audio della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteAudioInserter -> invoca il costruttore di Audio e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazio
- Premi::Model::Eliminazione::ConcreteAudioRemover -> rimuove l'oggetto Audio dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
- Premi::Model::Modifica::ConcreteFrameEditor -> invoca i metodi che impostano i campi dell'oggetto Audio.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Audio vengono istanziati da Premi::Model::Inserimento::ConcreteAudioInserter o da Premi::Model::Builder::Concrete e inseriti nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.5.8 Premi::Model::Presentazione::Video

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Video rappresentano gli elementi di tipo video della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteVideoInserter -> invoca il costruttore di Video e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazione
- Premi::Model::Eliminazione::ConcreteVideoRemover -> rimuove l'oggetto Video dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
- Premi::Model::Modifica::ConcreteFrameEditor -> invoca i metodi che impostano i campi dell'oggetto Video.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Video vengono istanziati da Premi::Model::Inserimento::ConcreteVideoInserter o da Premi::Model::Builder::ConcreteV e inseriti nei membri contenitori all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.5.9 Premi::Model::Presentazione::Background

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Background rappresentano lo sfondo della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Inserimento::ConcreteBackgroundInserter -> invoca il costruttore di Background e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::Presentazione::SlideShow;
- Premi::Model::Eliminazione::ConcreteBackgroundRemover -> rimuove l'oggetto Video dal membro contenitore all'interno di Premi::Model::Presentazione::SlideShow, ne invoca quindi il distruttore;
-

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Background vengono istanziati da Premi::Model::Inserimento::ConcreteBacgroundInserter o da Premi::Model::Builder::ConcreteBackgroundBuilder e inseriti nel membro contenitore all'interno di Premi::Model::Presentazione::SlideShow.
Classi ereditate:

- Premi::Model::Presentazione::SlideShowElement.

5.1.6 Premi::Model::Builder

Tipo, obiettivo e funzione del componente:Il package Premi::Model::Builder implementa il Design Pattern builder. Il package ha come scopo principale la creazione degli oggetti delle sottoclassi di Premi::Model::Presentazione::SlideShowElement al momento del caricamento della presentazione nel programma.
Relazioni d'uso di altre componenti:È in relazione con il package Premi::Model::Presentazione che ne costruisce il Director, e con il package Premi::Model::Presentazione, delle cui classi, sottoclassi di SlideShowElement, costruisce gli oggetti.

5.1.6.1 Premi::Model::Builder::Director

Tipo, obiettivo e funzione del componente: Implementazione della parte Director del Design Pattern Builder. Fornisce a Premi::Model::Presentazione::Loader gli oggetti della presentazione.
Relazioni d'uso di altre componenti:

- Premi::Model::Presentazione::Loader -> costruisce Director, passandogli i parametri per la costruizione degli oggetti delle sottoclassi di Premi::Model::Presentazione::SlideShowElement.
- Premi::Model::Builder::AbstractBuilder <- invoca il costruttore degli oggetti delle sottoclassi di AbstractBuilder che genereranno i file delle sottoclassi di Premi::Model::Presentazione::SlideS

Interfacce con e relazioni d'uso e da altre componenti:Viene invocato da Premi::Model::Presentazion
È costruito con design pattern singleton. Restituisce gli oggetti di tipo Premi::Model::Presentazione::SlideSh
a Premi::Model::Presentazione::Loader.

5.1.6.2 Premi::Model::Builder::AbstractBuilder

Tipo, obiettivo e funzione del componente: Classe astratta del Design Pattern Builder.
Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti delle sottoclassi della classe AbstractBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::SlideShowElement.
Sottoclassi:

- Premi::Model::Builder::ConcreteTextBuilder;
- Premi::Model::Builder::ConcreteFrameBuilder;
- Premi::Model::Builder::ConcreteImageBuilder;
- Premi::Model::Builder::ConcreteSVGBuilder;
- Premi::Model::Builder::ConcreteAudioBuilder;
- Premi::Model::Builder::ConcreteVideoBuilder;
- Premi::Model::Builder::ConcreteBackgroundBuilder.

5.1.6.3 Premi::Model::Builder::ConcreteTextBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::Text al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia gli oggetti passando i parametri ricevuti da Premi::Model::Presentazione::Loader;
- Premi::Model::Presentazione::Text <- istanza gli oggetti della classe Text e li restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteTextBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::Text e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.6.4 Premi::Model::Builder::ConcreteFrameBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::Frame al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia gli oggetti passando i parametri ricevuti da Premi::Model::Presentazione::Loader;

- Premi::Model::Presentazione::Frame <- istanza gli oggetti della classe Frame e li restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteFrameBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::Frame e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.6.5 Premi::Model::Builder::ConcreteImageBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::Image al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia gli oggetti passando i parametri ricevuti da Premi::Model::Presentazione::Loader;
- Premi::Model::Presentazione::Image <- istanza gli oggetti della classe Image e li restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteImageBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::Image e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.6.6 Premi::Model::Builder::ConcreteSVGBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::SVG al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia gli oggetti passando i parametri ricevuti da Premi::Model::Presentazione::Loader;
- Premi::Model::Presentazione::SVG <- istanza gli oggetti della classe SVG e li restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteSVGBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::SVG e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.6.7 Premi::Model::Builder::ConcreteAudioBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::Audio al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia gli oggetti passando i parametri ricevuti da Premi::Model::Presentazione::Loader;
- Premi::Model::Presentazione::Audio <- istanza gli oggetti della classe Audio e li restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteAudioBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::Audio e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.6.8 Premi::Model::Builder::ConcreteVideoBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::Video al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia gli oggetti passando i parametri ricevuti da Premi::Model::Presentazione::Loader;
- Premi::Model::Presentazione::Video <- istanza gli oggetti della classe Video e li restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteVideoBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::Video e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.6.9 Premi::Model::Builder::ConcreteBackgroundBuilder

Tipo, obiettivo e funzione del componente: Classe concreta del Design Pattern Builder. Ha lo scopo di istanziare gli oggetti di classe Premi::Model::Presentazione::Background al momento del caricamento della presentazione e di passarli all'oggetto di classe Director.
Relazioni d'uso di altre componenti:

- Premi::Model::Builder::Director -> istanzia l'oggetto passando i parametri ricevuti da Premi::Model::Presentazione::Loader;

- Premi::Model::Presentazione::Background <- istanza l'oggetto della classe Background e lo restituisce al director.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe ConcreteBackgroundBuilder vengono istanziati da Premi::Model::Builder::Director e hanno lo scopo di istanziare gli oggetti delle sottoclassi di Premi::Model::Presentazione::Background e di passarli al Director.
Classi ereditate:

- Premi::Model::Builder::AbstractBuilder.

5.1.7 Premi::Model::Caricamento

Tipo, obiettivo e funzione del componente: All'interno di questo Package viene implementato il Design Pattern strategy per il caricamento di nuovi elementi nella presentazione.
Relazioni d'uso di altre componenti:. Il package è in relazione con Premi::Controller::MobileEdit, Premi::Controller::DesktopEdit e [[ControllerUtente]] dai quali riceve i segnali e i parametri di caricamento dell'elemento.

5.1.7.1 Premi::Model::Caricamento::Uploader

Tipo, obiettivo e funzione del componente: Classe astratta definita per l'implementazione del Design Pattern strategy, per il caricamento di elementi all'interno dello spazio personale di un utente.
Relazioni d'uso di altre componenti:

- Premi::Controller::MobileEdit e Premi::Controller::DesktopEdit -> costruiscono un oggetto di una sottoclasse di Uploader e utilizzano i metodi da questi messi a disposizione per caricare il file nel server, inoltre costruiscono un oggetto della classe Premi::Model::Command::InsertCommand e lo danno in pasto Premi::Model::Command::Invoker;

ControllerUtente] -> costruisce un oggetto di una sottoclasse di Uploader e utilizza i metodi da questi messi a disposizione per caricare il file nel server.

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 strategy che rappresenta lo scheletro dell'algoritmo per il caricamento di un elemento nella presentazione.
Sottoclassi:

- Premi::Model::Caricamento::ConcreteSvgUploader;
- Premi::Model::Caricamento::ConcreteImageUploader;
- Premi::Model::Caricamento::ConcreteVideoUploader;
- Premi::Model::Caricamento::ConcreteAudioUploader.

5.1.7.2 Premi::Model::Caricamento::ConcreteSvgUploader

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di caricamento di un elemento svg all'interno dello spazio personale di un utente. È uno dei componenti concreti del Design Pattern Strategy.
Relazioni d'uso di altre componenti:

- Premi::Controller::MobileEdit e Premi::Controller::DesktopEdit -> costruiscono un oggetto di classe ConcreteSvgUploader e utilizzano i metodi da questo messi a disposizione per caricare il file nel server, inoltre costruiscono un oggetto della classe Premi::Model::Command::InsertCommand e lo danno in pasto Premi::Model::Command::Invoker;

ControllerUtente] -> costruisce un oggetto di classe ConcreteSvgUploader e utilizza i metodi da questi messi a disposizione per caricare il file nel server.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per caricare elementi SVG nello spazio personale di un utente ed eventualmente inserirlo automaticamente nello spazio personale di un utente ed eventualmente inserirlo automaticamente in una presentazione..
Classi ereditate:

- Premi::Model::Caricamento::Uploader.

5.1.7.3 Premi::Model::Caricamento::ConcreteImageUploader

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di caricamento di un elemento immagine all'interno dello spazio personale di un utente. È uno dei componenti concreti del Design Pattern Strategy.
Relazioni d'uso di altre componenti:

- Premi::Controller::MobileEdit e Premi::Controller::DesktopEdit -> costruiscono un oggetto di classe ConcreteImageUploader e utilizzano i metodi da questo messi a disposizione per caricare il file nel server, inoltre costruiscono un oggetto della classe Premi::Model::Command::InsertCommand e lo danno in pasto Premi::Model::Command::Invoker;

ControllerUtente] -> costruisce un oggetto di classe ConcreteImageUploader e utilizza i metodi da questi messi a disposizione per caricare il file nel server.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per caricare elementi di tipo immagine nello spazio personale di un utente ed eventualmente inserirlo automaticamente in una presentazione..
Classi ereditate:

- Premi::Model::Caricamento::Uploader.

5.1.7.4 Premi::Model::Caricamento:: ConcreteVideoUploader

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di caricamento di un elemento video all'interno dello spazio personale di un utente. È uno dei componenti concreti del Design Pattern Strategy.
Relazioni d'uso di altre componenti:

- Premi::Controller::MobileEdit e Premi::Controller::DesktopEdit -> costruiscono un oggetto di classe ConcreteVideoUploader e utilizzano i metodi da questo messi a disposizione per caricare il file nel server, inoltre costruiscono un oggetto della classe Premi::Model::Command::InsertCommand e lo danno in pasto Premi::Model::Command::Invoker;

ControllerUtente] -> costruisce un oggetto di classe ConcreteVideoUploader e utilizza i metodi da questi messi a disposizione per caricare il file nel server.

Interfacce con e relazioni d'uso e da altre componenti:Viene invocato per caricare elementi di tipo video nello spazio personale di un utente ed eventualmente inserirlo automaticamente in una presentazione.
Classi ereditate:

- Premi::Model::Caricamento::Uploader.

5.1.7.5 Premi::Model::Caricamento:: ConcreteAudioUploader

Tipo, obiettivo e funzione del componente: Classe che rappresenta un algoritmo di caricamento di un elemento di tipo audio all'interno dello spazio personale di un utente. È uno dei componenti concreti del Design Pattern Strategy.
Relazioni d'uso di altre componenti:

- Premi::Controller::MobileEdit e Premi::Controller::DesktopEdit -> costruiscono un oggetto di classe ConcreteAudioUploader e utilizzano i metodi da questo messi a disposizione per caricare il file nel server, inoltre costruiscono un oggetto della classe Premi::Model::Command::InsertCommand e lo danno in pasto Premi::Model::Command::Invoker;

ControllerUtente] -> costruisce un oggetto di classe ConcreteImageUploader e utilizza i metodi da questi messi a disposizione per caricare il file nel server.

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per caricare elementi di tipo audio nello spazio personale di un utente ed eventualmente inserirlo automaticamente in una presentazione..
Classi ereditate:

- Premi::Model::Caricamento::Uploader.

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;
- Premi::Model::Presentazione::Loader <- EditController costruisce l'oggetto di classe Loader, passando come parametro i riferimenti alla presentazione da caricare.

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 o 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. La pagina web può, inoltre richiedere il caricamento di una presentazione o la creazione di una nuova presentazione a EditController, che, tramite

invocazione di Premi::Model::Presentazione::Loader, caricherà dal database.

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.

5.2.1.3 Premi::Controller::Presentazione::ExecutionController

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

- Premi.View.Pages.Execution -> costruiscono ExecutionController, ne invocano i metodi passando i parametri della presentazione da caricare;
- Premi::Model::Presentazione::Loader <- ExecutionController passa i parametri di caricamento al Loader che istanzia un oggetto di classe Premi::Model::Presentazione::SlideShow e lo restituisce a Loader.

Interfacce con e relazioni d'uso e da altre componenti: La pagina Execution costruisce ExecutionController per caricare la presentazione.

5.3 View

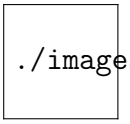

Fig 3: View

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.
Interfacce con e relazioni d'uso e da altre componenti: i metodi implementati nella classe IndexPage sono i seguenti:

- IndexPage::Login invia al controller i dati della login e, se corretti, manda alla pagina Home;
- IndexPage::Subscribe invia al controller i dati della registrazione e, se corretti, manda alla pagina Home;
- IndexPage::Manifest manda alla pagina Manifest.

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: i metodi implementati nella classe Home sono i seguenti:

- Home::Delete invia al controller l'id della presentazione da eliminare;
- Home::Download invia al controller l'id della presetazione da scaricare in locale;
- Home::Execute manda alla pagina Execution con l'id della presentazione da eseguire
- Home::NewSlideShow manda alla pagina Edit con la richiesta di una nuova presentazione;
- Home::EditSlideShow manda alla pagina Edit con l'id della presentazione da modificare;
- Home::Logout manda al controller la richiesta di logout e manda alla pagina Index.

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::Manifest

Tipo, obiettivo e funzione del componente: la classe Manifest definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra ad un utente le presentazioni scaricate in locale e da la possibilità di eseguirle.
Interfacce con e relazioni d'uso e da altre componenti: i metodi implementati nella classe Manifest sono i seguenti:

- Manifest::ExecuteManifest esegue la presentazione selezionata utilizzando la pagina html già presente in locale e il framework impress.js;
- Manifest::DeleteManifest elimina la presentazione salvate in locale;

Attività svolte e dati trattati: la classe definisce la struttura della pagina web che consente agli utenti di visualizzare le anteprime delle proprie presentazioni, eseguirle e eliminarle dalla posizione in locale.

5.3.4 Premi::View::Pages::Profile

Tipo, obiettivo e funzione del componente: 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
Relazioni d'uso di altre componenti: la classe Profile utilizza i metodi messi a disposizione dalle seguenti classi presenti nel package Controller:

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

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

R PASSWORD]]]]] per la modifica della password;

A FILE MEDIA]]]]] per rinominarli.

Interfacce con e relazioni d'uso e da altre componenti: i metodi implementati nella classe Profile sono i seguenti:

- Profile::ChangePassword invia al controller la nuova password;
- Profile::UploadMedia invia al controller le informazioni sul nuovo file media caricato sul server;
- Profile::DeleteMedia invia al controller l'id del file media da eliminare;
- Profile::RenameMedia invia al controller l'id e il nuovo nome del file media.

Attività svolte e dati trattati: 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.

5.3.5 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.
Interfacce con e relazioni d'uso e da altre componenti: i metodi implementati nella classe Execution sono gestiti dal framework Impress.js con l'aggiunta e la modifica delle seguenti 4 funzioni all'interno del framework:

- Execution::Next va al frame successivo della presentazione;
- Execution::Prev va al frame precedente;
- Execution::Bookmark va al frame con bookmark successivo.

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.

5.3.6 Premi::View::Pages::Edit

Tipo, obiettivo e funzione del componente: la classe Edit è divisa in due sottoclassi, che sono visualizzazioni di pagine web diverse a seconda del dispositivo dalla quale viene visualizzata, Desktop o Mobile.

5.3.7 Premi::View::Pages::EditDesktop

Tipo, obiettivo e funzione del componente: la classe EditDesktop definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra da dispositivo desktop 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 ;

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

Interfacce con e relazioni d'uso e da altre componenti: i metodi implementati nella classe EditDesktop sono i seguenti:

- EditDesktop::InsertFrame invia al controller la richiesta di inserimento di un nuovo frame, la sua forma, le coordinate di posizione;
- EditDesktop::InsertMedia invia al controller la richiesta di inserimento di un nuovo file media, le sue informazioni e le coordinate di posizione e di rotazione;
- EditDesktop::MoveElement invia al controller l'id dell'elemento spostato e le sue nuove coordinate;
- EditDesktop::InsertText invia al controller la richiesta di inserimento di un nuovo elemento di testo, il suo contenuto, la sua formattazione e le sue coordinate;
- EditDesktop::TextEdit invia al controller l'id dell'elemento di testo e il suo nuovo contenuto;
- EditDesktop::DeleteElement invia al controller l'id dell'elemento eliminato;
- EditDesktop::InsertChoice invia al controller la richiesta di inserimento di una nuova scelta e l'id del frame a cui è indirizzata la scelta;

- EditDesktop::Bookmark invia al controller l'id del frame al quale viene associato o rimosso (a seconda dello stato in quel momento) un bookmark;
- EditDesktop::ChangeSize invia al controller l'id dell'elemento al quale vengono cambiate le dimensioni e le nuove misure;
- EditDesktop::ChangeRotation invia al controller l'id dell'elemento al quale viene cambiata la rotazione la percentuale di rotazione;
- EditDesktop::ChangePath invia al controller l'id del percorso modificato e il nuovo ordine dei frame.
- EditDesktop::FrameBackground invia al controller la richiesta di inserimento di un nuovo sfondo ad un frame, l'id del frame e le informazioni dell'immagine;
- EditDesktop::Background invia al controller la richiesta di inserimento di un nuovo sfondo alla presentazione e le informazioni dell'immagine.

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.
Classi ereditate: Premi::View::Pages::Edit.

5.3.8 Premi::View::Pages::EditMobile

Tipo, obiettivo e funzione del componente: la classe EditMobile. definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra da dispositivo mobile ad un utente 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.

Interfacce con e relazioni d'uso e da altre componenti: i metodi implementati nella classe EditMobile sono i seguenti:

- EditDesktop::InsertText invia al controller la richiesta di inserimento di un nuovo elemento di testo, il suo contenuto, la sua formattazione e le sue coordinate;
- EditDesktop::TextEdit invia al controller l'id dell'elemento di testo e il suo nuovo contenuto;
- EditDesktop::Bookmark invia al controller l'id del frame al quale viene associato o rimosso (a seconda dello stato in quel momento) 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..
Classi ereditate: Premi::View::Pages::Edit.

6 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente per fornire una stima sulla fattibilità e di bisogno di risorse. L'analisi dell'architettura progettata ha permesso di constatare che le tecnologie che si è scelto di adottare risultano sufficientemente adeguate per la realizzazione del prodotto e riescono a ricoprire le esigenze progettuali. Poichè tutti gli strumenti da utilizzare nello sviluppo sono gratuiti, il bisogno di risorse non si dimostra essere particolarmente problematico.
Si è deciso di utilizzare Html5 per la presentazione grafica dell'applicazione web, insieme a Javascript (e le sue librerie) e CSS3.
Per la parte di database è stato deciso di utilizzare MongoDB per la compatibilità con documenti JSON che sarà il formato delle presentazioni e i framework Express.jss e Node.js per le comunicazioni tra lato client e lato server.
Per la modifica delle presentazioni e quindi la relativa interazione dell'utente con la pagina html viene utilizzato il framework Angular.js.
Infine per la parte relativa all'esecuzione verrà utilizzato il framework Impress.js che favorisce il tipo di visualizzazione di una presentazione richiesto dai requisiti.