

18-05-2015



Specifica Tecnica

Informazioni sul documento

| | |
|-----------------------------|------------------------------|
| Nome Documento | Specifica Tecnica |
| Versione | 1.0.0 |
| Stato | <i>Formale</i> |
| Uso | <i>Esterno</i> |
| Data Creazione | 18-05-2015 |
| Data Ultima Modifica | 18-05-2015 |
| Redazione | Fossa Manuel, Petrucci Mauro |
| Approvazione | Tollot Pietro |
| Verifica | Gabelli Pietro |
| Lista distribuzione | <i>LateButSafe</i> |
| | Prof. Tullio Vardanega |
| | Prof. Riccardo Cardin |
| | Proponente Zucchetti S.p.a. |



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

| | | |
|----------|---|-----------|
| 1 | Introduzione | 7 |
| 1.1 | Scopo del documento | 7 |
| 1.2 | Scopo del Prodotto | 7 |
| 1.3 | Glossario | 7 |
| 1.4 | Riferimenti | 7 |
| 1.4.1 | Normativi | 7 |
| 1.4.2 | Informativi | 7 |
| 2 | Strumenti | 9 |
| 2.1 | HTML | 9 |
| 2.2 | JavaScript | 9 |
| 2.3 | jQuery | 10 |
| 2.4 | Angular.js | 10 |
| 2.5 | Node.js | 10 |
| 2.6 | MongoDB | 10 |
| 2.7 | Impress.js | 10 |
| 3 | Design Pattern | 11 |
| 3.1 | MVC | 11 |
| 3.2 | Singleton | 11 |
| 3.3 | Utility | 11 |
| 3.4 | Builder | 11 |
| 3.5 | Command | 11 |
| 3.6 | Iterator | 12 |
| 3.7 | Template Method | 12 |
| 3.8 | Strategy | 12 |
| 4 | Descrizione architetturale | 13 |
| 4.1 | Metodo e formalismi | 13 |
| 4.2 | Architettura generale | 13 |
| 4.2.1 | Model | 13 |
| 4.2.2 | View | 13 |
| 4.2.3 | ViewModel | 13 |
| 5 | Descrizione dei singoli componenti | 15 |
| 5.1 | Model | 15 |
| 5.1.1 | Premi::Model::Inserimento | 15 |
| 5.1.1.1 | Premi::Model::Inserimento::Inserter | 15 |
| 5.1.1.2 | Premi::Controller::Presentazione::Inserimento::ConcreteTextInserter | 16 |
| 5.1.1.3 | Premi::Model::Inserimento::ConcreteFrameInserter | 16 |
| 5.1.1.4 | Premi::Model::Inserimento::ConcreteSvgInserter | 16 |
| 5.1.1.5 | Premi::Model::Inserimento::ConcreteImageInserter | 17 |
| 5.1.1.6 | Premi::Model::Inserimento:: ConcreteVideoInserter | 17 |
| 5.1.1.7 | Premi::Model::Inserimento:: ConcreteAudioInserter | 17 |



Università degli studi di Padova - 2014/2015

| | | | |
|-----|---------|---|----|
| | 5.1.7.5 | Premi::Model::Caricamento::ConcreteAudioUploader | 36 |
| 5.2 | | Controller | 38 |
| | 5.2.1 | Premi::Controller::Presentazione | 38 |
| | 5.2.1.1 | Premi::Controller::Presentazione::EditController | 38 |
| | 5.2.1.2 | Premi::Controller::Presentazione::HomeController | 39 |
| | 5.2.1.3 | Premi::Controller::Presentazione::ExecutionController | 39 |
| 5.3 | | View | 40 |
| | 5.3.1 | Premi.View.Pages.IndexPage | 40 |
| | 5.3.2 | Premi.View.Pages.Home | 41 |
| | 5.3.3 | Premi.View.Pages.Manifest | 41 |
| | 5.3.4 | Premi.View.Pages.Profile | 42 |
| | 5.3.5 | Premi.View.Pages.Execution | 42 |
| | 5.3.6 | Premi.View.Pages.DesktopEdit | 42 |
| | 5.3.7 | Premi.View.Pages.MobileEdit | 43 |

Sommario

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



1.1 Scopo del documento

1.2 Scopo del Prodotto

1.3 Glossario

1.4 Riferimenti

1.4.1 Normativi

- ### 1.4.2 Informativi

- Università degli studi di Padova - 2014/2015

- 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/>.



2.1 HTML

- **Vantaggi:**

- **Svantaggi:**

- ## 2.2 JavaScript

Università degli studi di Padova - 2014/2015

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 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. Il Model notifica ogni cambiamento al controller che la comunica alla View.

3.2 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.

3.3 Utility

- ???

3.4 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.

3.5 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.



- **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.

- **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.

- **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.



4.1 Metodo e formalismi

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

Per i diagrammi di Package, classi e attività verrà usata la notazione UML 2.(DA AGGIUNGERE INDICE).

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.

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.

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.

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

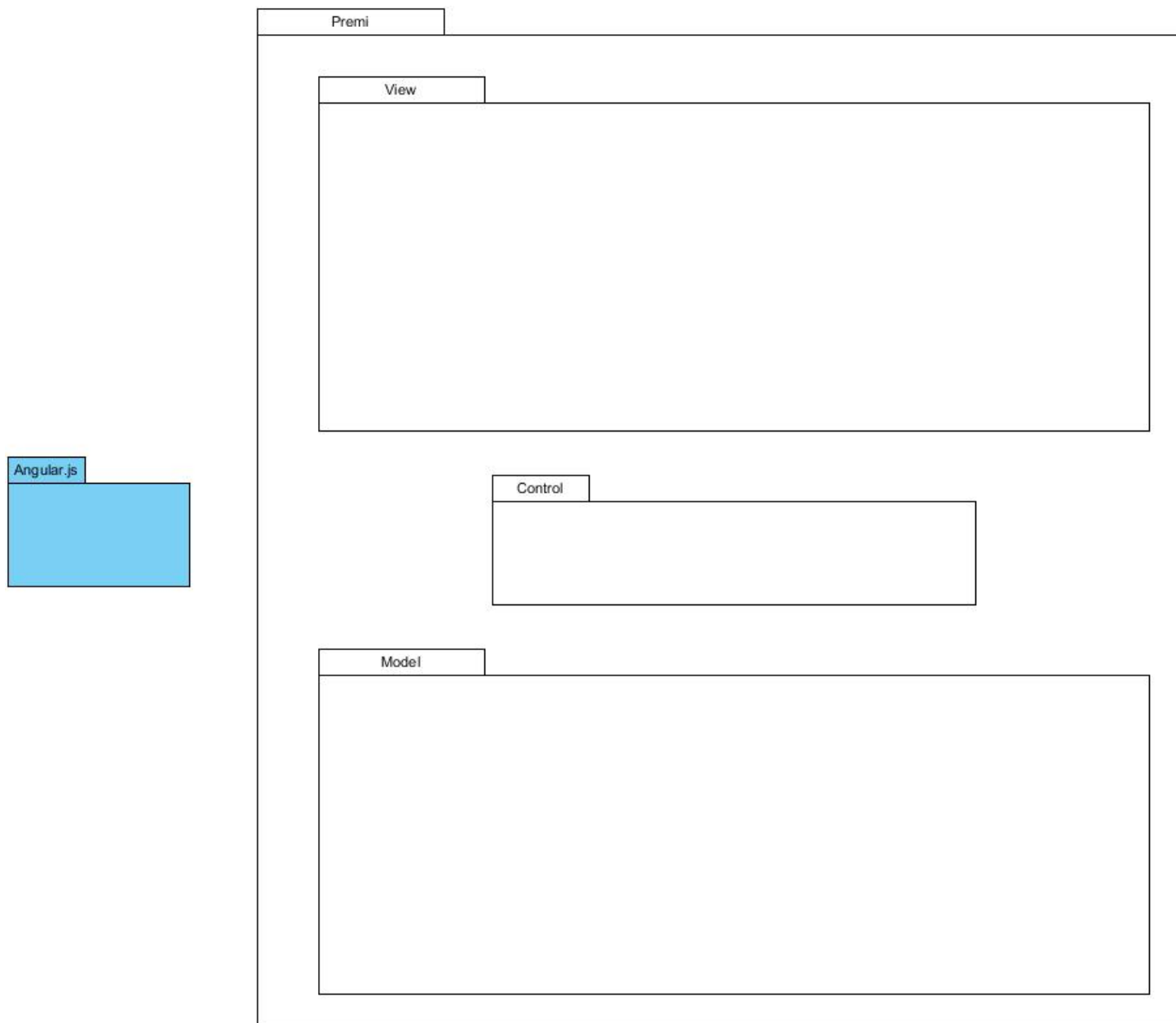


Fig 1: Architettura generale del sistema



5.1 Model

Relazioni d'uso di altre componenti: ??????????????????????.

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

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.

- `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.

Sottoclassi:



- #### 5.1.1.2 Premi::Controller::Presentazione::Inserimento::ConcreteTextInserter

Relazioni d'uso di altre componenti:

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

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

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:

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

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

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::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.

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.



Classi ereditate:

- #### 5.1.2.6 Premi::Model::Eliminazione:: ConcreteVideoRemover

Relazioni d'uso di altre componenti:

- Classi ereditate:**

- #### 5.1.2.7 Premi::Model::Eliminazione:: ConcreteAudioRemover

Relazioni d'uso di altre componenti:

- Classi ereditate:**

- ### 5.1.3 Premi::Model::Modifica

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.



Questo documento è distribuito sotto licenza [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/).



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::ConcreteEditCommand 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::ConcreteEditCommand 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.



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.

- `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.

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

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.

- `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.

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



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.

- `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.

Classi ereditate:

- #### 5.1.4 Premi::Model::Command

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::Preser

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.

- Premi::Controller::Inserimento::InsertController->crea un oggetto della classe Premi::Model::Command passandolo all'Invoker che lo esegue e lo inserisce nello stack “undo”, richiama il metodo che svuota lo stack “redo”;
- Premi::Controller::Eliminazione::RemoveController->crea un oggetto della classe Premi::Model::Command passandolo all'Invoker che lo esegue e lo inserisce nello stack “undo” , richiama il metodo che svuota lo stack “redo”;

- `Premi::Controller::Presentazione::EditController` -> crea un oggetto della classe `Premi::Model::Command` passandolo all'Invoker che lo esegue (`execute`) 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".

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. Si 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::Loader

Tipo, obiettivo e funzione del componente: Questa classe si occupa della costruzione dell'oggetto SlideShow quando l'utente carica una presentazione o quando l'utente crea una nuova presentazione.

Relazioni d'uso di altre componenti:

- Premi::Controller::EditController-> costruisce Loader;
- Premi::Controller::ExecutionController-> costruisce Loader;
- Premi::Model::SlideShow <- costruisce un oggetto di classe SlideShow ed eventualmente invoca le sue funzioni per inserire gli oggetti delle sottoclassi di SlideShowElement;
- Premi::Model::Builder::Director <- Loader passa a Director i parametri degli oggetti da costruire.

Interfacce con e relazioni d'uso e da altre componenti: il controller invoca il costruttore di Loader. Quando non passa alcun argomento Loader costruisce un oggetto di classe Premi::Model::SlideShow; altrimenti passa l'id dell'oggetto json che descrive la presentazione, Loader recupera i dati e costruisce un oggetto SlideShow, passa a Premi::Model::Builder::Director i parametri degli oggetti da istanziare contenuti dentro all'oggetto json che descrive la presentazione. Director restituisce gli oggetti istanziati e Loader invoca i metodi di SlideShow per inserire tali oggetti nei rispettivi contenitori.

5.1.5.2 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 costruttore 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.



Tipo, obiettivo e funzione del componente: Gli oggetti della classe `SlideShowElement` rappresentano gli elementi della presentazione.

- `Premi::Model::Inserimento::Insert` -> 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.

- Premi::Model::Presentazione:: Text;
- Premi::Model::Presentazione:: Frame;
- Premi::Model::Presentazione:: Image;
- Premi::Model::Presentazione::SVG;
- Premi::Model::Presentazione::Audio;
- Premi::Model::Presentazione::Video.

Tipo, obiettivo e funzione del componente: Gli oggetti della classe Text rappresentano gli elementi di tipo testuale della presentazione.

- `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`.

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



Questo documento è distribuito sotto licenza [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

5.1.5.7 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::SlideShow`;
- `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::ConcreteSVGInserter` 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::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::Presentazione;
- 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.



5.1.5.9 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::SlideShow;
- 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::ConcreteVideoInserter e inseriti nei membri contenitori 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 costruzione degli oggetti delle sottoclassi di Premi::Model::Presentazione::SlideShowElement e dell'oggetto Premi::Model::Presentazione::SlideShow.
- Premi::Model::Builder::AbstractBuilder <- invoca il costruttore degli oggetti delle sottoclassi di AbstractBuilder che genereranno i file delle sottoclassi di Premi::Model::Presentazione::SlideShowElement.

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



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:

- ### 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` <- istanzia 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 `Premii::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` <- istanzia 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` <- istanzia 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`.



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.

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

Classi ereditate:

- 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.

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

Classi ereditate:

- 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.



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.

- `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`:

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..

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

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.

- `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`:

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.

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

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.

Università degli studi di Padova - 2014/2015



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

Classi ereditate:

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



Questo documento è distribuito sotto licenza [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

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

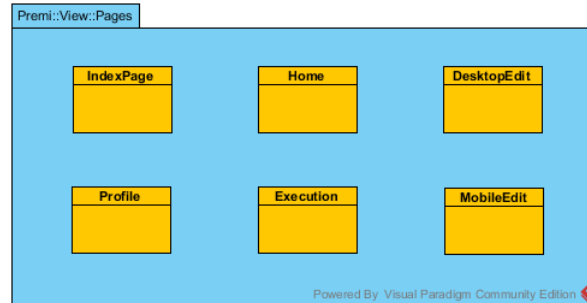


Fig 2: View

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.

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.



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.

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

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

LER LOGOUT ||||| per effettuare il logout.

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

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.

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.

- `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, 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:

MODIFICA DATI [|||||] per modificare i propri dati di profilo;

UPLOAD FILE MEDIA [||||] per il caricamento di file media nel server;

ELIMINAZIONE FILE MEDIA [||||] per la loro eliminazione dal server;

RINOMINAZIONE 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.

5.3.5 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.

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.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: manda 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:

RICARICA EDITOR [|||||] per caricare la presentazione da modificare;

INSERIMENTO [|||||] per l'inserimento di nuovi elementi;

SPOSTAMENTO [|||||] per lo spostamento di nuovi elementi;

ELIMINAZIONE [|||||] per l'eliminazione elementi;



LA PERCORSO ||||| per cambiare il percorso della presentazione.

5.3.7 Premi.View.Pages.MobileEdit

Relazioni d'uso di altre componenti: la classe MobileEdit utilizza i metodi messi a disposizione dalle seguenti classi presenti nel package Controller:

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..