

## Informazioni sul documento

<b>Nome Documento</b>	Specifica Tecnica
<b>Versione</b>	1.0.0
<b>Stato</b>	<i>Formale</i>
<b>Uso</b>	<i>Esterno</i>
<b>Data Creazione</b>	18-05-2015
<b>Data Ultima Modifica</b>	18-05-2015
<b>Redazione</b>	Fossa Manuel, Petrucci Mauro
<b>Approvazione</b>	Tollot Pietro
<b>Verifica</b>	Gabelli Pietro
<b>Lista distribuzione</b>	<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	Petrucci Mauro	27-05-2015	Approvazione del documento
0.7.0	Venturelli Giovanni	26-05-2015	Apportata correzioni segnalate dal verificatore Gabelli Pietro
0.5.0	Venturelli Giovanni	23-05-2015	Aggiunta dei contenuti
0.3.0	Petrucci Mauro	14-05-2015	Aggiunta dei contenuti
0.2.0	Fossa Manuel	12-05-2015	Aggiunta dei contenuti
0.1.0	Busetto Matteo	10-05-2015	Stesura dello scheletro del documento


$$\mathbf{RR} \rightarrow \mathbf{RP}$$
Tab 2: Storico ruoli pre-RR

# Indice

<b>1</b>	<b>Introduzione</b>	<b>8</b>
1.1	Scopo del documento . . . . .	8
1.2	Scopo del Prodotto . . . . .	8
1.3	Glossario . . . . .	8
1.4	Riferimenti . . . . .	8
1.4.1	Normativi . . . . .	8
1.4.2	Informativi . . . . .	8
<b>2</b>	<b>Strumenti</b>	<b>10</b>
2.1	HTML . . . . .	10
2.2	JavaScript . . . . .	10
2.3	jQuery . . . . .	10
2.4	MEAN . . . . .	11
2.4.1	MongoDB . . . . .	11
2.4.2	Express.js . . . . .	11
2.4.3	AngularJS . . . . .	11
2.4.4	Node.js . . . . .	11
2.5	Impress.js . . . . .	12
<b>3</b>	<b>Design Pattern e Pattern Architeturali</b>	<b>13</b>
3.1	MVP . . . . .	13
3.2	Command . . . . .	13
3.2.1	Premi::Model::Command . . . . .	14
3.3	Strategy . . . . .	14
3.3.1	Premi::Model::Modifica . . . . .	14
<b>4</b>	<b>Descrizione architetturale</b>	<b>15</b>
4.1	Metodo e formalismi . . . . .	15
4.2	Architettura generale . . . . .	15
4.2.1	Model . . . . .	15
4.2.2	View . . . . .	15
4.2.3	ViewModel . . . . .	15
<b>5</b>	<b>Descrizione dei singoli componenti</b>	<b>17</b>
5.1	Model . . . . .	17
5.2	Premi::Model::SlideShow . . . . .	17
5.3	Premi::Model::SlideShow::ModificaSlideShow . . . . .	17
5.4	Premi::Model::SlideShow::SlideShowActions . . . . .	17
5.5	Premi::Model::SlideShow::SlideShowActions::InsertEditRemove . . . . .	18
5.5.1	Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter . . . . .	18
5.5.2	Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover . . . . .	19
5.6	Premi::Model::SlideShow::SlideShowActions::Command . . . . .	20
5.6.1	Premi::Model::Invoker . . . . .	20
5.6.2	Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand . . . . .	21



5.6.3	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand	22
5.6.4	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand	22
5.6.5	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand	22
5.6.6	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand	23
5.6.7	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand	23
5.6.8	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand	23
5.6.9	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand	23
5.6.10	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand	23
5.6.11	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand	23
5.6.12	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand	23
5.6.13	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand	23
5.6.14	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand	23
5.6.15	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand	23
5.6.16	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemoveCommand	23
5.6.17	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand	28
5.6.18	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand	28
5.6.19	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand	28
5.6.20	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand	28
5.6.21	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand	28
5.6.22	Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditFontCommand	30
5.6.23	Premi::Model::SlideShow::SlideShowElements	30
5.6.24	Premi::Model::SlideShow::SlideShowElements::SlideShowElement	30
5.6.25	Premi::Model::SlideShow::SlideShowElements::Text	31
5.6.26	Premi::Model::SlideShow::SlideShowElements::Frame	32
5.6.27	Premi::Model::SlideShow::SlideShowElements::Image	32
5.6.28	Premi::Model::SlideShow::SlideShowElements::SVG	33
5.6.29	Premi::Model::SlideShow::SlideShowElements::Audio	33
5.6.30	Premi::Model::SlideShow::SlideShowElements::Video	34
5.6.31	Premi::Model::SlideShow::Background	34
5.7	Premi::Model::ServerRelations	35
5.8	Premi::Model::ServerRelations::Loader	35
5.8.1	Premi::Model::ServerRelations::Loader::Costruttore	35
5.9	Premi::Model::ServerRelations::AccessControl	35
5.9.1	Premi::Model::ServerRelations::AccessControl::Autenticazione	35
5.9.2	Premi::Model::ServerRelations::AccessControl::Registrazione	36
5.10	Premi::Model::ServerRelations::DbConsistency	36
5.10.1	Premi::Model::ServerRelations::DbConsistency::Observer	36
5.10.2	Premi::Model::ServerRelations::DbConsistency::ConcreteObserver	36
5.10.3	Premi::Model::ServerRelations::DbConsistency::Subject	36
5.10.4	Premi::Model::ServerRelations::DbConsistency::SubjectAudio	37
5.10.5	Premi::Model::ServerRelations::DbConsistency::SubjectAudio	37
5.10.6	Premi::Model::ServerRelations::DbConsistency::SubjectVideo	37
5.10.7	Premi::Model::ServerRelations::DbConsistency::SubjectText	37
5.10.8	Premi::Model::ServerRelations::DbConsistency::SubjectFrame	37
5.10.9	Premi::Model::ServerRelations::DbConsistency::SubjectImg	37
5.10.10	Premi::Model::ServerRelations::DbConsistency::SubjectSVG	38



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



## Elenco delle figure

1	Fig1 . . . . .	13
2	Command . . . . .	13
3	Architettura generale del sistema . . . . .	16
4	View . . . . .	45
5	Attività Principali . . . . .	50
6	Gestione Presentazioni . . . . .	51
7	Caricare File . . . . .	51
8	Modificare Presentazione da Desktop . . . . .	52
9	Modificare Presentazione da Mobile . . . . .	52
10	Gestire Sfondo . . . . .	53
11	Inserire Elemento . . . . .	54
12	Modificare Elemento . . . . .	54
13	Modificare Frame . . . . .	55
14	Modificare SVG . . . . .	55
15	Modificare Testo . . . . .	56

## Elenco delle tabelle

1	Versionamento del documento . . . . .	1
2	Storico ruoli pre-RR . . . . .	2

# 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

- 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/>.



## 2.1 HTML

- **Vantaggi:**

- **Svantaggi:**

- ## 2.2 JavaScript

## 2.3 jQuery

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, mantenuto 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 impostato 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



## 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.

## 3 Design Pattern e Pattern Architeturali

includegraphics[scale=0.6]{./images/MVP.pdf}

Fig 1: Fig1

### 3.1 MVP

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

### 3.2 Command

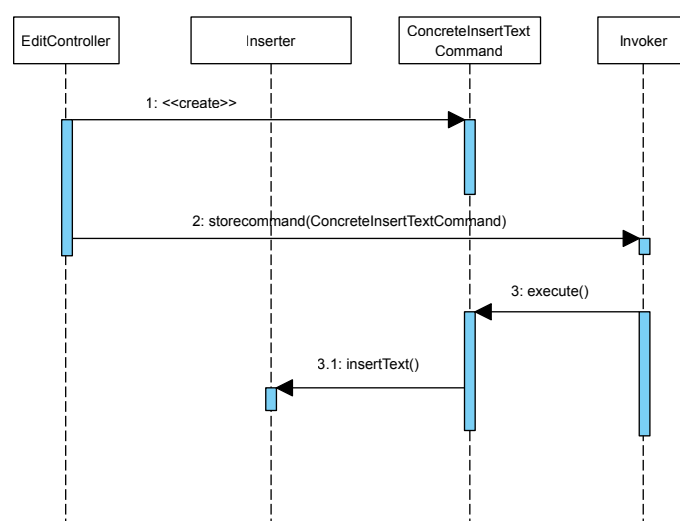


Fig 2: Command

- **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.

### 3.2.1 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::Presenter::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::Presenter::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::Presenter::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`.



## 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.



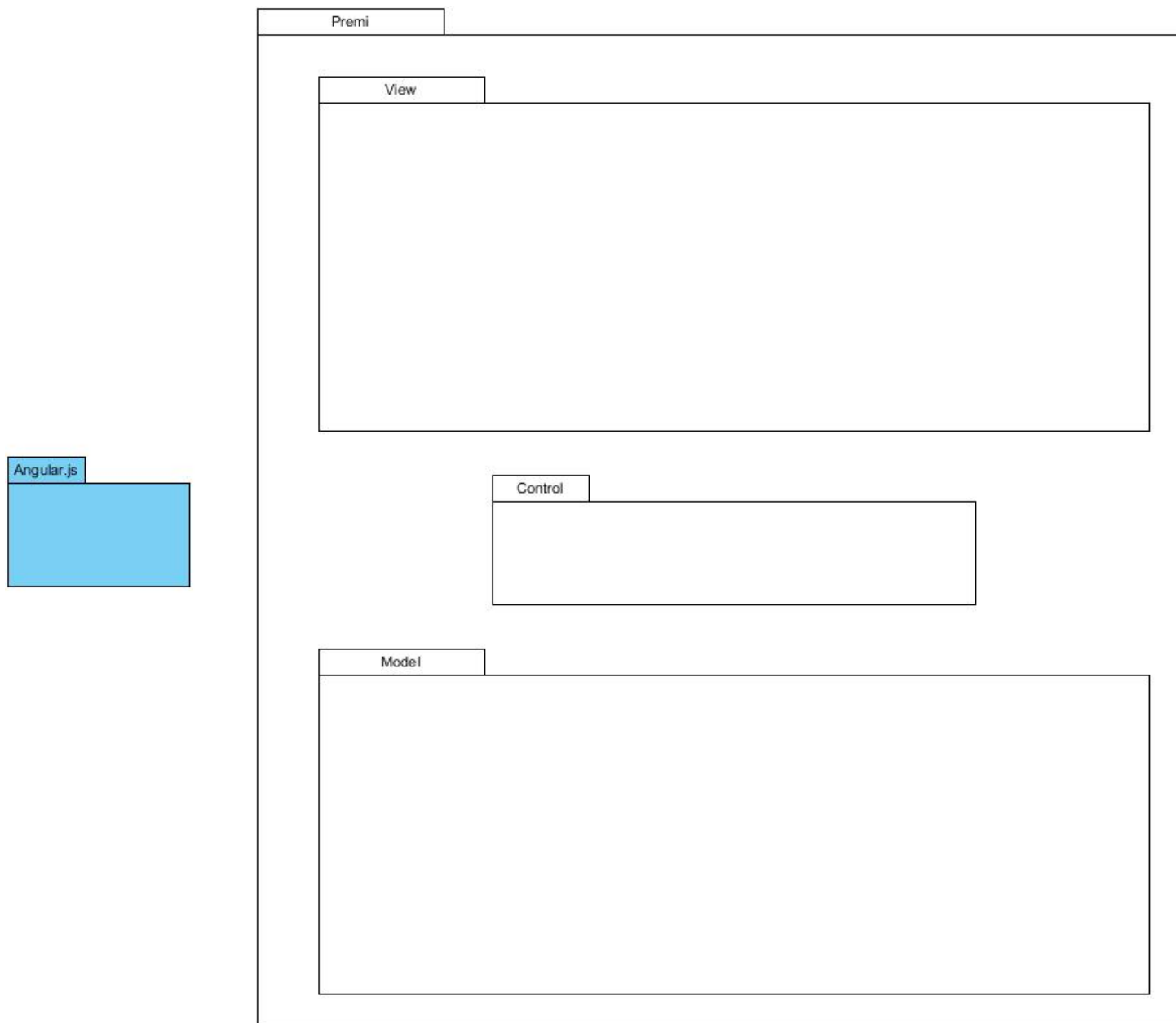


Fig 3: Architettura generale del sistema



## 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:** è in relazione con il package Presenter e con NodeAPI.

**Package contenuti:**

- Premi::Model::SlideShow;
- Premi::Model::ServerRelations;

### 5.2 Premi::Model::SlideShow

**Tipo, obiettivo e funzione del componente:** All'interno di questo Package si trovano le classi che si riferiscono alla costruzione e alla modifica degli elementi della presentazione oltre alle classi che rappresentano gli elementi stessi della presentazione.

**Relazioni d'uso di altre componenti:** il package è in relazione con Premi::Presenter::EditPresenter da cui riceve i segnali e i parametri di inserimento e modifica degli elementi. Inoltre comunica con il package Premi::Model::ServerRelations, inviando a questi i segnali per la modifica in tempo reale dei dati presenti nel database.

### 5.3 Premi::Model::SlideShow::ModificaSlideShow

**Tipo, obiettivo e funzione del componente:** All'interno di questo Package si trovano le classi che definiscono gli algoritmi di modifica, inserimento e rimozione degli elementi della presentazione.

**Relazioni d'uso di altre componenti:** il package è in relazione con Premi::Presenter::EditPresenter da cui riceve i segnali e i parametri di inserimento e modifica degli elementi. Inoltre comunica con il package Premi::Model::ServerRelations, inviando a questi i segnali per la modifica in tempo reale dei dati presenti nel database.

### 5.4 Premi::Model::SlideShow::SlideShowActions

**Tipo, obiettivo e funzione del componente:** All'interno di questo Package si trovano le classi che si riferiscono alla costruzione, all'inserimento, alla rimozione e alla modifica degli elementi della presentazione.

**Relazioni d'uso di altre componenti:** il package è in relazione con Premi::Model::SlideShow::SlideShow da cui riceve i segnali e i parametri di inserimento e modifica degli elementi. Inoltre comunica con il package Premi::Model::ServerRelations, inviando a questi i segnali per la modifica in tempo reale dei dati presenti nel database.



## 5.5 Premi::Model::SlideShow::SlideShowActions::InsertEditRemove

**Tipo, obiettivo e funzione del componente:** all'interno di questo Package sono implementate le classi statiche destinate all'inserimento, alla rimozione e alla modifica degli elementi della presentazione.

**Relazioni d'uso di altre componenti:** il package è in relazione con Premi::Model::SlideShow::SlideShowActions::InsertEditRemove che invoca i metodi delle classi del package. // Inoltre Premi::Model::SlideShow::SlideShowActions::InsertEditRemove si occupa di costruire gli oggetti presenti nelle classi del package Premi::Model::SlideShow::SlideShowElements::Text, Premi::Model::SlideShow::SlideShowElements::Frame, Premi::Model::SlideShow::SlideShowElements::Image, Premi::Model::SlideShow::SlideShowElements::SVG, Premi::Model::SlideShow::SlideShowElements::Audio.

### 5.5.1 Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter

**Tipo, obiettivo e funzione del componente:** Classe statica che offre dei metodi per l'inserimento di elementi all'interno di una presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand -> invoca il metodo insertText() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand -> invoca il metodo insertFrame() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand -> invoca il metodo insertImage() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand -> invoca il metodo insertSVG() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand -> invoca il metodo insertAudio() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand -> invoca il metodo insertVideo() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand -> invoca il metodo insertBackground() messo a disposizione da Inserter;
- Premi::Model::SlideShow::SlideShowElements::Text <- Inserter costruisce gli oggetti di classe Text;
- Premi::Model::SlideShow::SlideShowElements::Frame <- Inserter costruisce gli oggetti di classe Frame;
- Premi::Model::SlideShow::SlideShowElements::Image <- Inserter costruisce gli oggetti di classe Image;
- Premi::Model::SlideShow::SlideShowElements::SVG <- Inserter costruisce gli oggetti di classe SVG;
- Premi::Model::SlideShow::SlideShowElements::Audio <- Inserter costruisce gli oggetti di classe Audio;



- Interfacce con e relazioni d'uso e da altre componenti:** È il componente receiver del Design Pattern Command.

**Tipo, obiettivo e funzione del componente:** Classe statica che offre i metodi destinati all'eliminazione degli elementi all'interno di una presentazione.

- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand` -> invoca il metodo `removeText()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand` -> invoca il metodo `removeFrame()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand` -> invoca il metodo `removeImage()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand` -> invoca il metodo `removeSVG()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand` -> invoca il metodo `removeAudio()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand` -> invoca il metodo `removeVideo()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemoveCommand` -> invoca il metodo `removeBackground()` messo a disposizione da `Remover`;
- `Premi::Model::SlideShow::SlideShowElements::Text` <- Editor invoca i metodi di set degli oggetti di classe `Text`;
- `Premi::Model::SlideShow::SlideShowElements::Frame` <- Editor invoca i metodi di set degli oggetti di classe `Frame`;
- `Premi::Model::SlideShow::SlideShowElements::Image` <- Editor invoca i metodi di set degli oggetti di classe `Image`;
- `Premi::Model::SlideShow::SlideShowElements::SVG` <- Editor invoca i metodi di set degli oggetti di classe `SVG`;



- `Premi::Model::SlideShow::SlideShowElements::Audio` <- Editor invoca i metodi di set degli oggetti di classe `Audio`;
- `Premi::Model::SlideShow::SlideShowElements::Video` <- Editor invoca i metodi di set degli oggetti di classe `Video`;
- `Premi::Model::SlideShow::SlideShowElements::Background` <- Editor invoca i metodi di set degli oggetti di classe `Background`;
- `Premi::Model::SlideShow::SlideShowElements::Text` <- Editor invoca i metodi di set degli oggetti di classe `Text`;
- `Premi::Model::SlideShow::SlideShowElements::Frame` <- Editor invoca i metodi di set degli oggetti di classe `Frame`;
- `Premi::Model::SlideShow::SlideShowElements::Image` <- Editor invoca i metodi di set degli oggetti di classe `Image`;
- `Premi::Model::SlideShow::SlideShowElements::SVG` <- Editor invoca i metodi di set degli oggetti di classe `SVG`;
- `Premi::Model::SlideShow::SlideShowElements::Audio` <- Editor invoca i metodi di set degli oggetti di classe `Audio`;
- `Premi::Model::SlideShow::SlideShowElements::Video` <- Editor invoca i metodi di set degli oggetti di classe `Video`;
- `Premi::Model::SlideShow::SlideShowElements::Background` <- Editor invoca i metodi di set degli oggetti di classe `Background`;

**Interfacce con e relazioni d'uso e da altre componenti:** È il componente receiver del Design Pattern Command.

## 5.6 `Premi::Model::SlideShow::SlideShowActions::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::SlideShow::SlideShowActions::Insert`, `Premi::Model::Remove` e `Premi::Model::SlideShow::SlideShowActions::Delete`. Il package comunica, inoltre, con il presenter, infatti le sue classi sono generate da `Premi::Presenter::SlideShow::EditPresenter`.

### 5.6.1 `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::Presenter::MobileEdit->crea` un oggetto di una sottoclasse di `Premi::Model::SlideShow::SlideS` 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::Presenter::DesktopEdit->`si comporta in modo analogo a `MobileEdit`;
- `Premi::Model::SlideShow::SlideShowActions::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::SlideShow::SlideShowActions::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.

### 5.6.2 Premi::Model::SlideShow::SlideShowActions::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::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand;



- ### 5.6.3 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand

Relazioni d'uso di altre componenti:

- Classi ereditate:**

- #### 5.6.4 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertComm.

Relazioni d'uso di altre componenti:

- Università degli studi di Padova - 2014/2015



- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertFrame() della classe statica per l'inserimento di un elemento frame nella presentazione.

#### Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

#### 5.6.5 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertComm

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per inserire un nuovo elemento immagine nella presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertImage() della classe statica per l'inserimento di un elemento immagine nella presentazione.

#### Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

#### 5.6.6 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertComm

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per inserire un nuovo elemento SVG nella presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertSVG() della classe statica per l'inserimento di un elemento SVG nella presentazione.

#### Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.





### 5.6.7 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertComm

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per inserire un nuovo elemento audio nella presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertAudio() della classe statica per l'inserimento di un elemento audio nella presentazione.

**Interfacce con e relazioni d'uso e da altre componenti:** Viene utilizzata per gestire le richieste di inserimento di un nuovo elemento Audio.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.8 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertComm

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per inserire un nuovo elemento video nella presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertVideo() della classe statica per l'inserimento di un elemento video nella presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.9 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertC

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per inserire un nuovo elemento video nella presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;



- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertBackground() della classe statica per l'inserimento di un elemento sfondo nella presentazione.

#### Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

#### 5.6.10 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCom

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento dalla presentazione.

#### Relazioni d'uso di altre componenti:

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeText() della classe statica per la rimozione di un elemento testuale nella presentazione.

#### Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

#### 5.6.11 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCom

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento frame dalla presentazione.

#### Relazioni d'uso di altre componenti:

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeFrame() della classe statica per la rimozione di un elemento frame dalla presentazione.

#### Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.



### 5.6.12 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCom

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento immagine dalla presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeImage() della classe statica per l'eliminazione di un elemento immagine dalla presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.13 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCom

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento SVG dalla presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeSVG() della classe statica per l'eliminazione di un elemento SVG dalla presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.14 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCom

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento audio dalla presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;



- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover` <- invoca il metodo `removeAudio()` della classe statica per l'eliminazione di un elemento immagine dalla presentazione.

Classi ereditate:

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.15 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCom

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere un elemento video dalla presentazione.

Relazioni d'uso di altre componenti:

- `Premi::Presenter::SlideShow::EditPresenter ->` invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- `Premi::Model::SlideShow::SlideShowActions::Command::Invoker ->` Invoker invoca il metodo `doaction()` del comando o ne invoca il metodo di annullamento;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <-` invoca il metodo `removeVideo()` della classe statica per l'eliminazione di un elemento video dalla presentazione.

Classi ereditate:

- `Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand`.

### 5.6.16 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemo

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per rimuovere lo sfondo della presentazione.

Relazioni d'uso di altre componenti:

- `Premi::Presenter::SlideShow::EditPresenter ->` invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- `Premi::Model::SlideShow::SlideShowActions::Command::Invoker ->` Invoker invoca il metodo `doaction()` del comando o ne invoca il metodo di annullamento;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <-` invoca il metodo `removeBackground()` della classe statica per l'eliminazione dell'elemento sfondo dalla presentazione.

Classi ereditate:

- `Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand`.



### 5.6.17 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per modificare le dimensioni di un elemento della presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editSize() della classe statica per la modifica dei campi dati relativi alle dimensioni dell'oggetto nella presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.18 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per modificare la posizione di un elemento della presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editPosition() della classe statica per la modifica dei campi dati relativi alla posizione dell'oggetto nella presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

### 5.6.19 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per modificare il colore di un elemento della presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;



- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editColor() della classe statica per la modifica del campo dati relativo al colore dell'oggetto della presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

**5.6.20 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundC**

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per modificare lo sfondo di un elemento frame della presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editBackground() della classe statica per la modifica del campo dati relativo allo sfondo dell'oggetto della presentazione.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.

**5.6.21 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCom**

**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per modificare l'orientamento di un elemento della presentazione.

**Relazioni d'uso di altre componenti:**

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editRotation() della classe statica per la modifica del campo dati relativo all'orientamento dell'oggetto della presentazione.

«««« e88e6bcbe29175cd959243ec0f79391e3cae0b11 **Interfacce con e relazioni d'uso e da altre componenti:** Viene utilizzata per gestire i Signal riguardanti la modifica dell'orientamento di un elemento;

===== »»»> 5d4b765697ecf4c596dcad1eb3d4f52bf617fc3d **Classi ereditate:**

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.



**Tipo, obiettivo e funzione del componente:** È classe concreta del Design Pattern Command, rappresenta un comando per modificare il carattere di un elemento testuale della presentazione.

- Premi::Presenter::SlideShow::EditPresenter -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Premi::Model::SlideShow::SlideShowActions::Command::Invoker -> Invoker invoca il metodo doaction() del comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editColor() della classe statica per la modifica dei campi dati relativi al font dell'oggetto testuale della presentazione.

- `Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand.`

**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.

- `Premi::Model::SlideShow::SlideShowActions::Insert`, i cui oggetti durante la modifica della presentazione istanziano oggetti di tipo `SlideShowElement`;
- `Premi::Model::Remove`, i cui oggetti rimuovono da `Premi::ServerRelations::Caricatore` gli oggetti di tipo `SlideShowElement` e li distruggono;
- `Premi::Model::SlideShow::SlideShowActions::EditElements`, i cui oggetti invocano metodi degli oggetti `SlideShowElement` che ne impostano i campi;

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

- `Premi::Model::SlideShow::SlideShowActions::Insert::Insert` -> invoca il costruttore delle sottoclassi di `SlideShowElement` e li inserisce nei campi dati contenitori all'interno di `Premi::Model::SlideShow::SlideShow`;
- `Premi::Model::SlideShow::SlideShowActions::EditElements::Editor` -> gli oggetti delle sue sottoclassi richiamano le funzioni delle sottoclassi di `SlideShowElement` che gestiscono l'impostazione dei campi dati;





- `Premi::Model::SlideShow::SlideShowActions::Remove::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::SlideShow::SlideShowActions` istanzia oggetti di sottoclassi di `SlideShowElement` e li inserisce nel campo dati contenitore presentazione all'interno di `Premi::Model::ServerRelations::Model::Costruttore`

**Sottoclassi:**

- `Premi::Model::SlideShow::Text`;
- `Premi::Model::SlideShow::Frame`;
- `Premi::Model::SlideShow::Image`;
- `Premi::Model::SlideShow::SVG`;
- `Premi::Model::SlideShow::Audio`;
- `Premi::Model::SlideShow::Video`;
- `Premi::Model::SlideShow::Background`.

#### 5.6.25 `Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::InsertEditRemove::Inserter` -> invoca il costruttore di `Text` e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe `Premi::Model::ServerRelations::Loader::Caricatore`;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover` -> rimuove l'oggetto `Text` dal campo dati presentazione all'interno di `Premi::Model::ServerRelations::Loader::Costruttore` e invoca quindi il distruttore;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor` -> invoca i metodi che modificano i campi dati dell'oggetto.

**Interfacce con e relazioni d'uso e da altre componenti:** Gli oggetti della classe `Text` vengono istanziati da `Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteTextInserter` e inseriti nel campo dati contenitore presentazione all'interno di `Premi::Model::ServerRelations::Loader::Costruttore`

**Classi ereditate:**

- `Premi::Model::SlideShow::SlideShowElement`.





### 5.6.26 Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::InsertEditRemove::Inserter -> invoca il costruttore di Frame e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe Premi::Model::ServerRelations::Loader::Caricatore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Frame dal campo dati presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore e invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

**Interfacce con e relazioni d'uso e da altre componenti:** Gli oggetti della classe Frame vengono istanziati da Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter inseriti nel campo dati contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowElement.

### 5.6.27 Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::InsertEditRemove::Inserter -> invoca il costruttore di Image e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe Premi::Model::ServerRelations::Loader::Caricatore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Image dal campo dati presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore e invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

**Interfacce con e relazioni d'uso e da altre componenti:** Gli oggetti della classe Image vengono istanziati da Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter inseriti nel campo dati contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore.

**Classi ereditate:**

- Premi::Model::SlideShow::SlideShowElement.



### 5.6.28 Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::InsertEditRemove::Inserter -> invoca il costruttore di SVG e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe Premi::Model::ServerRelations::Loader::Caricatore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto SVG dal campo dati presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore e invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

**Interfacce con e relazioni d'uso e da altre componenti:** Gli oggetti della classe SVG vengono istanziati da Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter e da questi inseriti nel campo dati contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore.

**Classi eritate:**

- Premi::Model::SlideShow::SlideShowElement.

### 5.6.29 Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::InsertEditRemove::Inserter -> invoca il costruttore di Audio e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe Premi::Model::ServerRelations::Loader::Caricatore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Audio dal campo dati presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore e invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

**Interfacce con e relazioni d'uso e da altre componenti:** Gli oggetti della classe Audio vengono istanziati da Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter e da questi inseriti nel campo dati contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costruttore.

**Classi eritate:**

- Premi::Model::SlideShow::SlideShowElements::SlideShowElement.



**Tipo, obiettivo e funzione del componente:** Gli oggetti della classe Video rappresentano gli elementi di tipo video della presentazione.

- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter` -> invoca il costruttore di `Video` e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe `Premi::Model::ServerRelations::Loader::Caricatore`;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover` -> rimuove l'oggetto `Video` dal campo dati presentazione all'interno di `Premi::Model::ServerRelations::Loader::Costruttore` e invoca quindi il distruttore;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor` -> invoca i metodi che modificano i campi dati dell'oggetto.

- Premi::Model::SlideShow::SlideShowElements::SlideShowElement.

**Tipo, obiettivo e funzione del componente:** Gli oggetti della classe Background rappresentano lo sfondo della presentazione.

- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter` -> invoca il costruttore di `Background` e inserisce l'oggetto nel campo dati contenitore all'interno dell'oggetto della classe `Premi::Model::ServerRelations::Loader::Caricatore`;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Remover` -> rimuove l'oggetto `Video` dal campo dati presentazione all'interno di `Premi::Model::ServerRelations::Loader::Costruttore` e invoca quindi il distruttore;
- `Premi::Model::SlideShow::SlideShowActions::InsertEditRemove::Editor` -> invoca i metodi che modificano i campi dati dell'oggetto.

- Premi::Model::SlideShow::SlideShowElements::SlideShowElement.



## 5.7 Premi::Model::ServerRelations

**Tipo, obiettivo e funzione del componente:** il package racchiude le funzionalità del sistema che interagiscono direttamente con i servizi web esposti dalla interfaccia nodeApi.

**Relazioni d'uso di altre componenti:** i componenti del package serverRelations hanno relazioni di dipendenza nei confronti del package nodeApi del quale utilizzano i servizi esposti dall'interfaccia; c'è dipendenza tra il package serverRelations ed altri package del model.

## 5.8 Premi::Model::ServerRelations::Loader

**Tipo, obiettivo e funzione del componente:** il package racchiude le funzioni di recupero di una presentazione dal server attraverso i servizi nodeApi e traduzione della presentazione in elementi html che compongono la view della presentazione recuperata.

**Relazioni d'uso di altre componenti:** relazione di dipendenza con l'interfaccia dei servizi nodeApi per il recupero della presentazione.

### 5.8.1 Premi::Model::ServerRelations::Loader::Costruttore

**Tipo, obiettivo e funzione del componente:** Classe la cui funzione è recuperare una presentazione dal database remoto o creare una nuova presentazione, caricare la presentazione in formato html così da poter essere modificata o eseguita dall'utente.

**Relazioni d'uso di altre componenti:**

- nodeAPI <- dipendenza nei confronti del package nodeApi di cui chiama i servizi http in modo sincrono.

## 5.9 Premi::Model::ServerRelations::AccessControl

**Tipo, obiettivo e funzione del componente:** il package racchiude le funzioni di registrazione dell'utente e autenticazione tramite token ai servizi nodeApi.

**Relazioni d'uso di altre componenti:** dipendenza nei confronti dei servizi resi disponibili dall'interfaccia nodeApi; altri package in ServerRelations utilizzano questo package per recuperare il token per accedere ai servizi nodeApi di interazione con le presentazioni in remoto.

### 5.9.1 Premi::Model::ServerRelations::AccessControl::Autenticazione

**Tipo, obiettivo e funzione del componente:** Classe che fornisce funzionalità di autenticazione e deautenticazione ai servizi offerti da nodeApi attraverso passaggio di token.

**Relazioni d'uso di altre componenti:**

- nodeAPI <- dipendenza nei confronti di nodeApi di cui chiama in modo sincrono i servizi.
- Premi::Presenter::Pagine::IndexPresenter -> invoca i metodi di Autenticazione per permettere all'utente di effettuare il login.



- `nodeAPI <-` dipendenza nei confronti di `nodeApi` di cui chiama in modo sincrono i servizi.
- `Premii::Presenter::Pagine::IndexPresenter ->` invoca i metodi di Registrazione per permettere all'utente di registrarsi al servizio.

**Relazioni d'uso di altre componenti:** dipendenza con il package nodeApi; dipendenza nei confronti di altri package in Model per il recupero dello stato degli elementi della presentazione.

- associazione con Subject per rendere effettiva la notify(); realizzata da ConcreteObserver che definisce il metodo update().

- realizza l'interfaccia Observer definendone il metodo update(); associazione verso Subject.

- associazione da ConcreteObserver; classe astratta realizzata dalle classi: SubjectAudio, SubjectVideo, SubjectText, SubjectFrame, SubjectSvg, SubjectImg che definiscono il metodo getElement() utilizzato da ConcreteObserver per ottenere l'oggetto modificato.



#### 5.10.4 Premi::Model::ServerRelations::DbConsistency::SubjectAudio

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

**Relazioni d'uso di altre componenti:**

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Audio di cui detiene un riferimento.

#### 5.10.5 Premi::Model::ServerRelations::DbConsistency::SubjectAudio

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

**Relazioni d'uso di altre componenti:**

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Audio di cui detiene un riferimento.

#### 5.10.6 Premi::Model::ServerRelations::DbConsistency::SubjectVideo

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

**Relazioni d'uso di altre componenti:**

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Video di cui detiene un riferimento.

#### 5.10.7 Premi::Model::ServerRelations::DbConsistency::SubjectText

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

**Relazioni d'uso di altre componenti:**

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Text di cui detiene un riferimento.

#### 5.10.8 Premi::Model::ServerRelations::DbConsistency::SubjectFrame

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

**Relazioni d'uso di altre componenti:**

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Frame di cui detiene un riferimento.

#### 5.10.9 Premi::Model::ServerRelations::DbConsistency::SubjectImg

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

**Relazioni d'uso di altre componenti:**

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Image di cui detiene un riferimento.

#### 5.10.10 Premi::Model::ServerRelations::DbConsistency::SubjectSVG

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::SVG di cui detiene un riferimento.

### 5.10.11 Premi::Model::ServerRelations::DbConsistency::SubjectBackground

**Tipo, obiettivo e funzione del componente:** Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern "Observer".

Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione con la classe Premi::Model::SlideShow::SlideShowElements::Background di cui detiene un riferimento.

## 5.11 Premi::Model::Manifest

**Tipo, obiettivo e funzione del componente:** Questo package ha lo scopo di rendere disponibili le presentazioni in locale tramite chiamate ai servizi di nodeApi e ai metodi definiti in Premi::Model::ServerRelation::Loader.

Relazioni d'uso di altre componenti:

- definisce il metodo `GestoreManifest()`; relazione di dipendenza

### 5.11.1 Premi::Model::Manifest::GestoreManifest

**Tipo, obiettivo e funzione del componente:** classe, fornisce i servizi raccolti nel package;

Relazioni d'uso di altre componenti:

- definisce il metodo `insertElement()`, `addPage()`, `update()`, associazione con la classe `Premi::Model::ServerRelation::Loader`.

## 5.12 Premi::Model::GestioneFileServer

**Tipo, obiettivo e funzione del componente:** Questo package racchiude le funzioni che permettono all'utente di interfacciarsi coi propri file presenti sul server Apache. **Relazioni d'uso di altre componenti:**

- relazione di dipendenza con l'interfaccia dei servizi di Apache.

**Interfacce con e relazioni d'uso e da altre componenti:** Viene utilizzata per gestire le chiamate asincrone verso il server, per realizzare le operazioni di inserimento, cancellazione, cambio del nome di un file.



### 5.12.1 Premi::Model::GestioneFileServer::GestioneFile

**Tipo, obiettivo e funzione del componente:** classe che fornisce un'implementazione del package, per permettere il caricamento di file sul server, eliminazione di file dal server e rinominazione di file presenti sul server;

Relazioni d'uso di altre componenti:

- Premi::Presenter::EditPresenter, Premi::View::Pages::Profile, interfaccia del server Apache;



### 5.13 Presenter

**Tipo, obiettivo e funzione del componente:** fanno parte di questo livello i package che gestiscono i segnali e le chiamate effettuati dalla view.

**Relazioni d'uso di altre componenti:** comunica con il Model per rendere possibile la gestione del profilo e la gestione delle presentazioni da parte dell'utente.

### 5.13.1 Premi::Presenter::EditPresenter

**Tipo, obiettivo e funzione del componente:** Lo scopo di questa classe è di gestire i segnali e le chiamate delle pagine Premi::View::Pages::DesktopEdit e Premi::View::Pages::MobileEdit.

Relazioni d'uso di altre componenti:

- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;
- `Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand` <- `EditPresenter` costruisce un comando e lo dà in pasto a `Premi::Model::Invoker`;



- Interfacce con e relazioni d'uso e da altre componenti:** La pagina DesktopEdit o la pagina MobileEdit invia a EditPresenter comunica l'avvenuta modifica o la rimozione di un



### 5.13.2 Premi::Presenter::HomePresenter

Relazioni d'uso di altre componenti:

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

Relazioni d'uso di altre componenti:

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



#### 5.13.4 Premi::Presenter::IndexPresenter

**Tipo, obiettivo e funzione del componente:** Lo scopo di questa classe è di gestire i segnali e le chiamate della pagina Premi::View::Pages::Index.

**Relazioni d'uso di altre componenti:**

- Premi::Model::ServerRelations::Loader::Autenticazione <- Quando la view invia una richiesta di login, HomePresenter invoca il metodo authenticate() fornito da Autenticazione, se il login ha successo IndexPresenter invia alla view una richiesta di redirect alla pagina Home;
- Premi::Model::ServerRelations::Loader::Registrazione <- Quando la view invia una richiesta di registrazione, HomePresenter invoca il metodo register() fornito da Registrazione, se la registrazione ha successo viene eseguito il login e IndexPresenter invia alla view una richiesta di redirect alla pagina Home;

**Interfacce con e relazioni d'uso e da altre componenti:** La pagina Index costruisce IndexPresenter per svolgere le operazioni di login e logout.

#### 5.13.5 Premi::Presenter::ProfilePresenter

**Tipo, obiettivo e funzione del componente:** Lo scopo di questa classe è di gestire i segnali e le chiamate della pagina Premi::View::Pages::Presenter.

**Relazioni d'uso di altre componenti:**

- Premi::Model::ApacheManager::FileManager <- EditPresenter invoca i metodi di FileManager per caricare un file nel server, per modificarne il nome o per eliminarlo dal server;
- Premi::Model::ServerRelations::Loader::Caricatore <- EditPresente invoca i metodi di questa classe per cambiare il nome di una presentazione;
- Premi::Model::ServerRelations::Loader::Autenticazione <- Quando la view invia una richiesta di logout ProfilePresenter invoca il metodo di Autenticazione deAuthenticate(), che termina la sessione. ProfilePresenter invia quindi una richiesta di redirect alla pagina Index.

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

#### 5.13.6 Premi::Presenter::ApacheManager

**Tipo, obiettivo e funzione del componente:** Compito di questo package è di gestire l'interazione con il server Apache.

**Relazioni d'uso di altre componenti:**

- Il package comunica con la view ricevendo chiamate da Premi::View::Pages::Profile. Comunica con il model chiamando i metodi definiti nel package Premi::Model::GestioneFileServer.



**Tipo, obiettivo e funzione del componente:** Lo scopo di questa classe è di gestire le chiamate della pagina Premi::View::Pages::Profile per l'inserimento, cancellazione e rinominazione di file sul server Apache.

- `Premi::View::Pages::Profile::UploadMedia` -> costruisce `ApacheManager`, ne invoca i metodi passando i parametri dell'utente ed i parametri del file da caricare;
- `Premi::View::Pages::Profile::DeleteMedia` -> costruisce `ApacheManager`, ne invoca i metodi passando i parametri dell'utente ed i e l'id del file media da eliminare;
- `Premi::View::Pages::Profile::RenameMedia` -> costruisce `ApacheManager`, ne invoca i metodi passando i parametri dell'utente, l'id e il nuovo nome del file media da rinominare;
- `Premi::Model::Caricamento::Uploader` <- `ApacheManager` passa i parametri di caricamento ad `Uploader` che istanzia l'oggetto sul server.
- `Premi::Presenter::EditPresenter` <- `ApacheManager` passa lo username dell'utente che sta svolgendo operazioni sul file, il file ed il tipo del file al server `Apache`; questo se l'operazione è andata a buon fine, ritorna un segnale ad `ApacheManager`, che lo trasmette ad `EditController`.

Università degli studi di Padova - 2014/2015

## 5.14 View

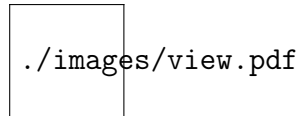


Fig 4: View

**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.

### 5.14.1 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 `Presenter::Index`, contenuta nel package `Presenter`, 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.

### 5.14.2 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



dalla classe Presenter::Home per l'eliminazione delle presentazioni dal server, per scaricare una presentazione in locale e 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 presentazione da scaricare in locale; item Home::Rename invia al controller l'id della presentazione da rinominare e il suo nuovo titolo;
- 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.

#### 5.14.3 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.

#### 5.14.4 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 dalla classe Presenter::Profile, per il caricamento di file media nel server, per la loro eliminazione





dal server, per la modifica della password e 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.

#### 5.14.5 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 della classe Presenter::Execution 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 3 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.

#### 5.14.6 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.





**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:** la classe EditDesktop utilizza i metodi messi a disposizione dalla classe Presenter::Edit per caricare la presentazione da modificare, per l'inserimento di nuovi elementi, per lo spostamento di nuovi elementi, per l'eliminazione elementi, per le modifiche effettuate agli elementi e per cambiare il percorso della presentazione.

- `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;

- `EditDesktop::InsertSVG` invia al controller al richiesta di inserimento di un nuovo elemento SVG, la sua forma, il suo colore e le coordinate di posizione e di rotazione.

**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.

#### 5.14.8 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 EditMobile utilizza i metodi messi a disposizione dalla classe Presenter::Edit per caricare la presentazione da modificare, per l'inserimento di un elemento testuale, per la modifica di un elemento testuale, per l'inserimento di un nuovo 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 Diagrammi di attività

Vengono ora illustrati i diagrammi di attività che descrivono le interazioni dell'utente con Premi. È stato disegnato un diagramma ad alto livello che descrive le attività possibili, le quali vengono poi illustrate tramite dei sotto-diagrammi specifici.

## 6.1 Attività Principali

L'utente una volta aperto il software Premi potrà loggarsi, registrarsi oppure accedere alla pagina per visualizzare le presentazioni scaricare in locale. Dopodichè l'utente potrà decidere se modificare la propria password, gestire, modificare o eseguire le proprie presentazioni oppure gestire il proprio profilo.

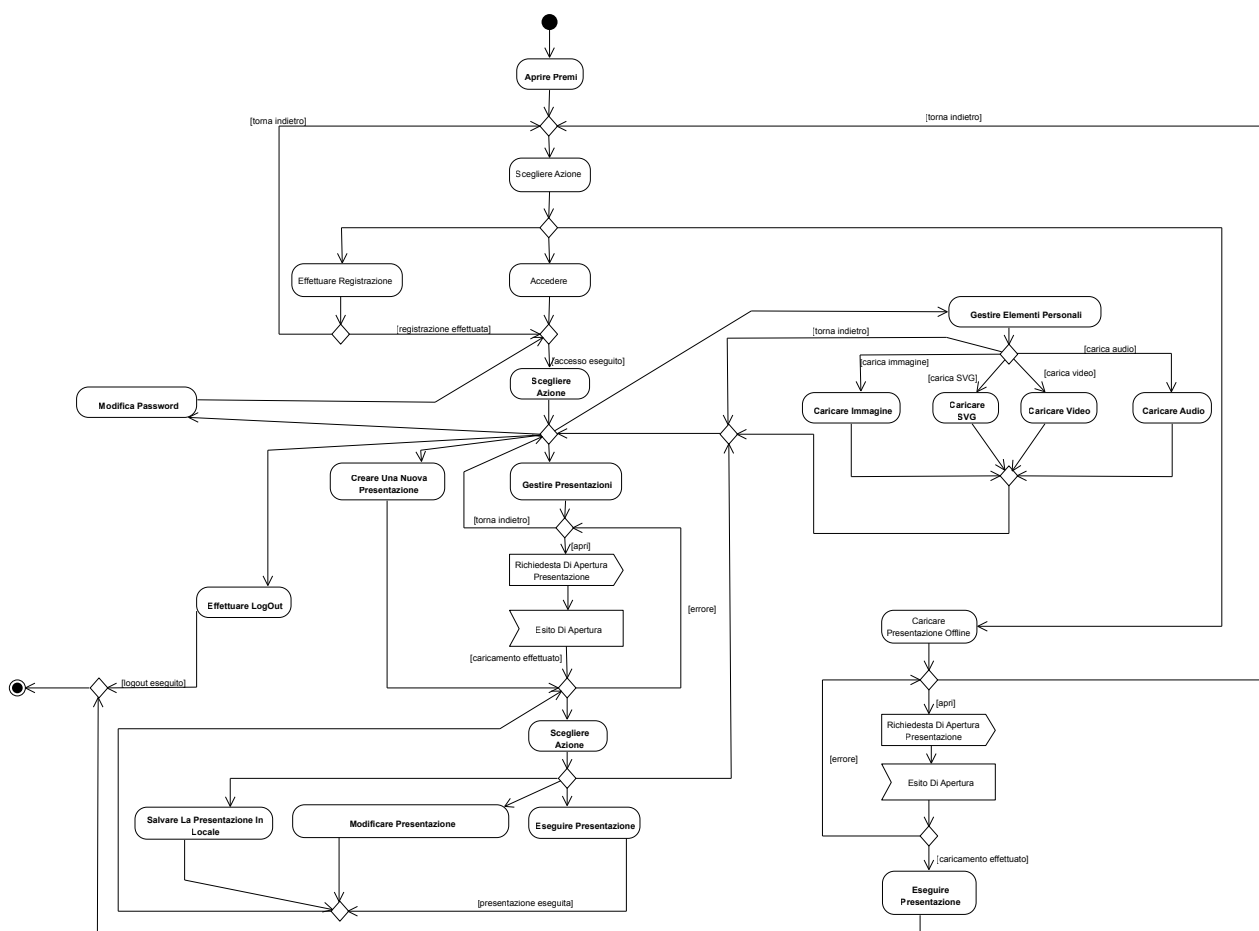


Fig 5: Attività Principali

### 6.1.1 Gestione presentazioni

L'utente una volta scelto di gestire le proprie presentazioni potrà rinominarle, aprirle o eliminarle.

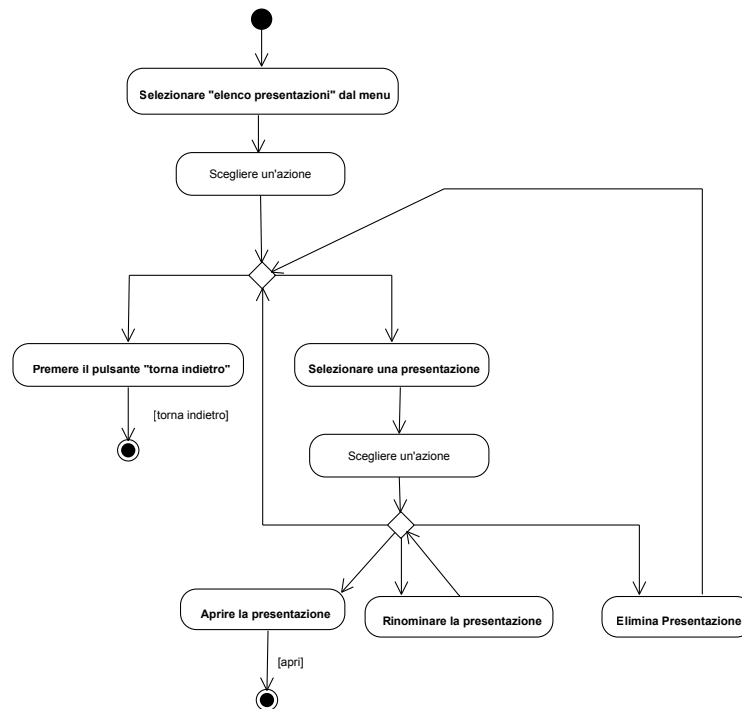


Fig 6: Gestione Presentazioni

### 6.1.2 Caricare File

L'utente una volta scelto di gestire il proprio profilo potrà caricare nuovi file all'interno del proprio spazio sul server.

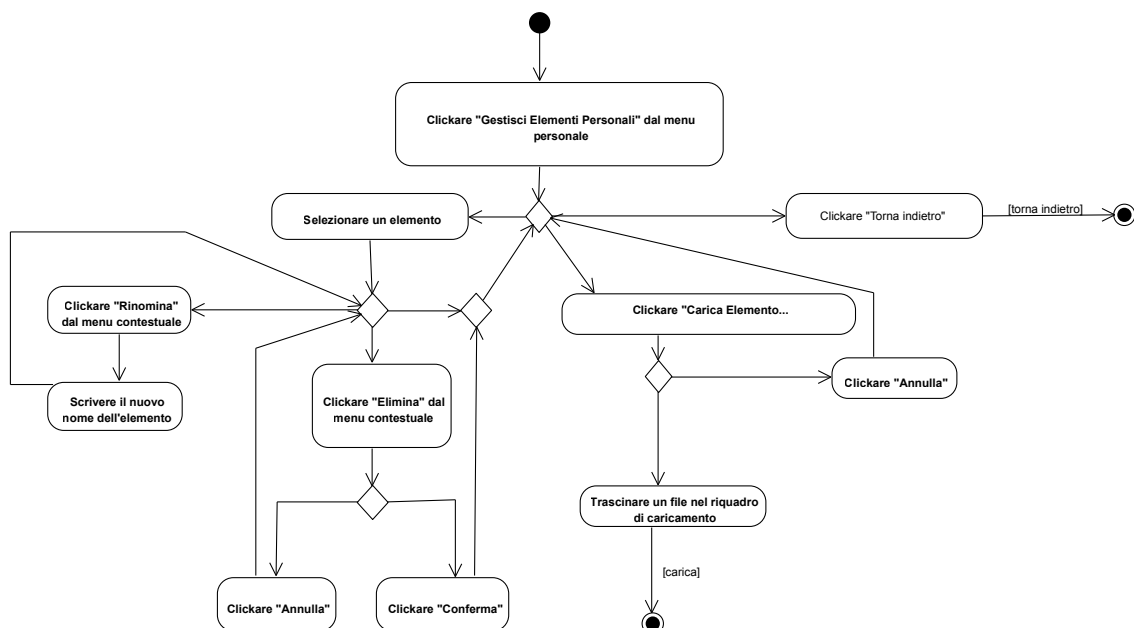


Fig 7: Caricare File

### 6.1.3 Modificare Presentazione da Desktop

L'utente una volta scelto di modificare una presentazione da mobile potrà decidere che tipo di modifiche apportare.

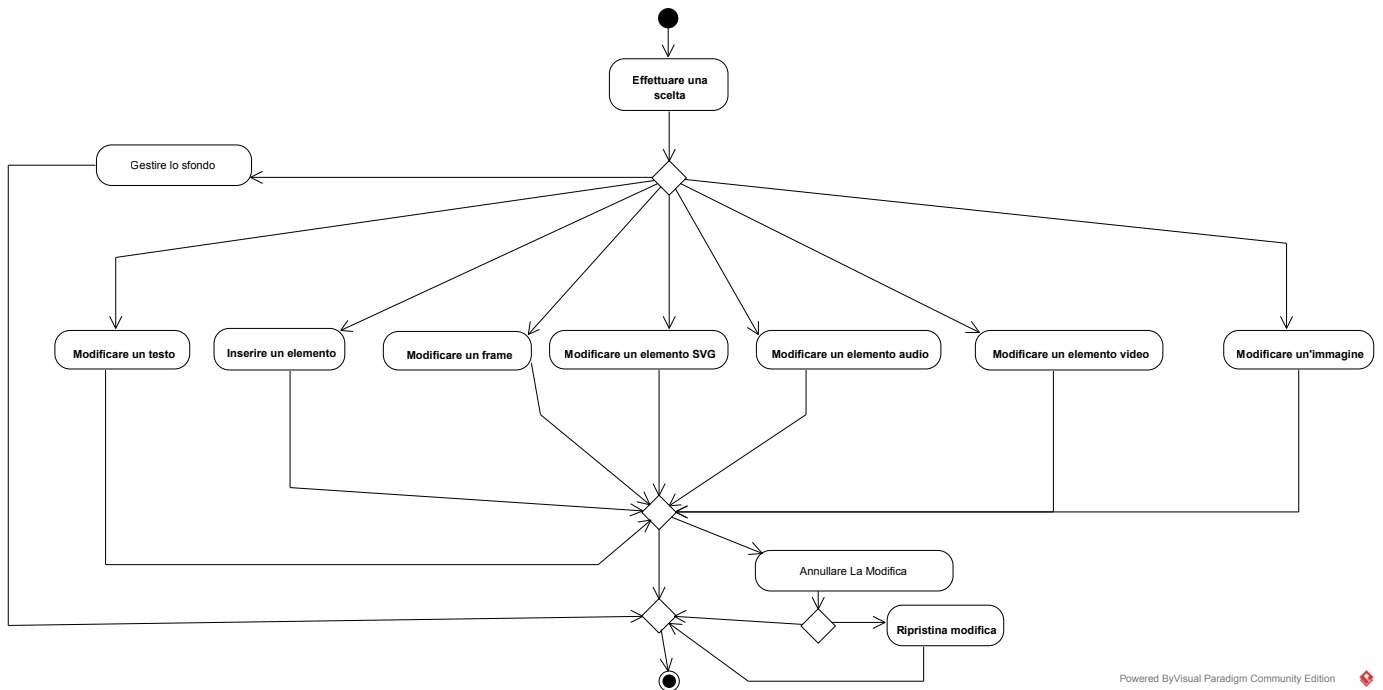


Fig 8: Modificare Presentazione da Desktop

#### 6.1.4 Modificare Presentazione da Mobile

L'utente una volta scelto di modificare una presentazione da mobile potrà decidere che tipo di modifiche apportare.

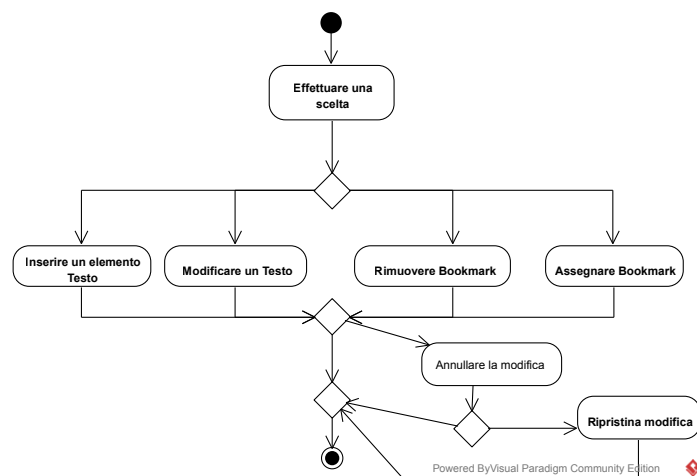


Fig 9: Modificare Presentazione da Mobile



L'utente una volta scelto di modificare una presentazione da mobile potrà decidere di apportare una modifica allo sfondo.



L'utente una volta scelto di modificare una presentazione da mobile potrà decidere di inserire un nuovo elemento sul piano della presentazione.



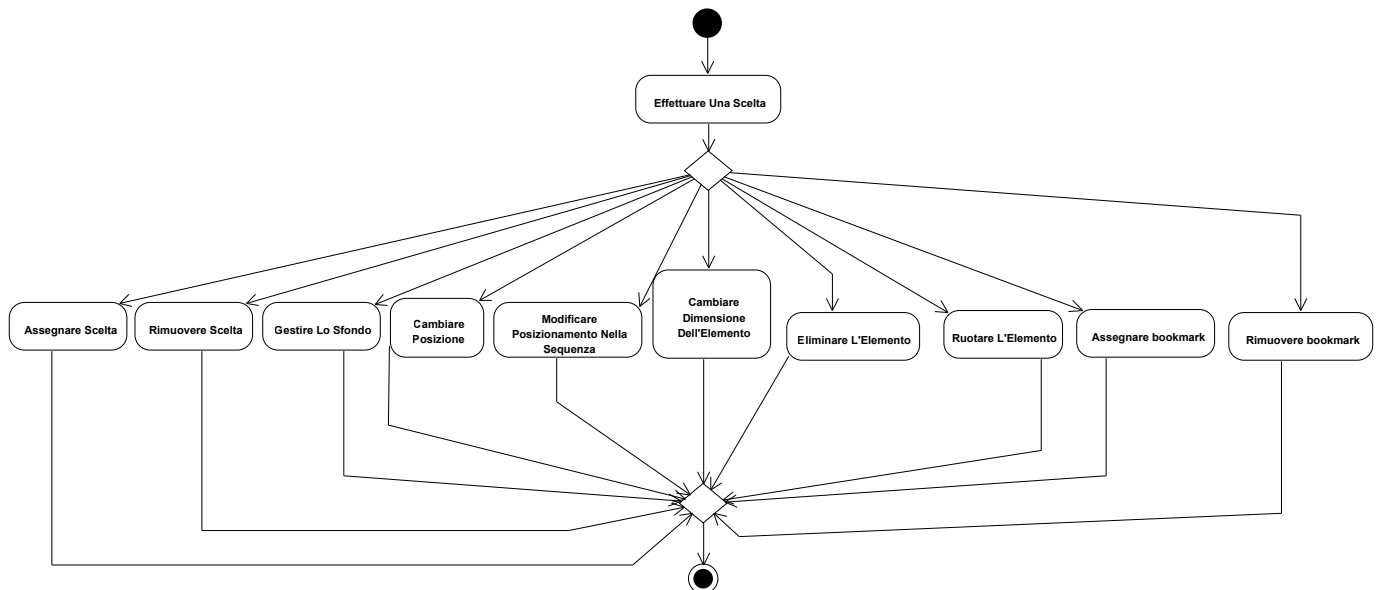


Fig 13: Modificare Frame

### 6.1.9 Modificare SVG

L'utente una volta scelto di modificare una presentazione da mobile potrà decidere di modificare un svg selezionato.

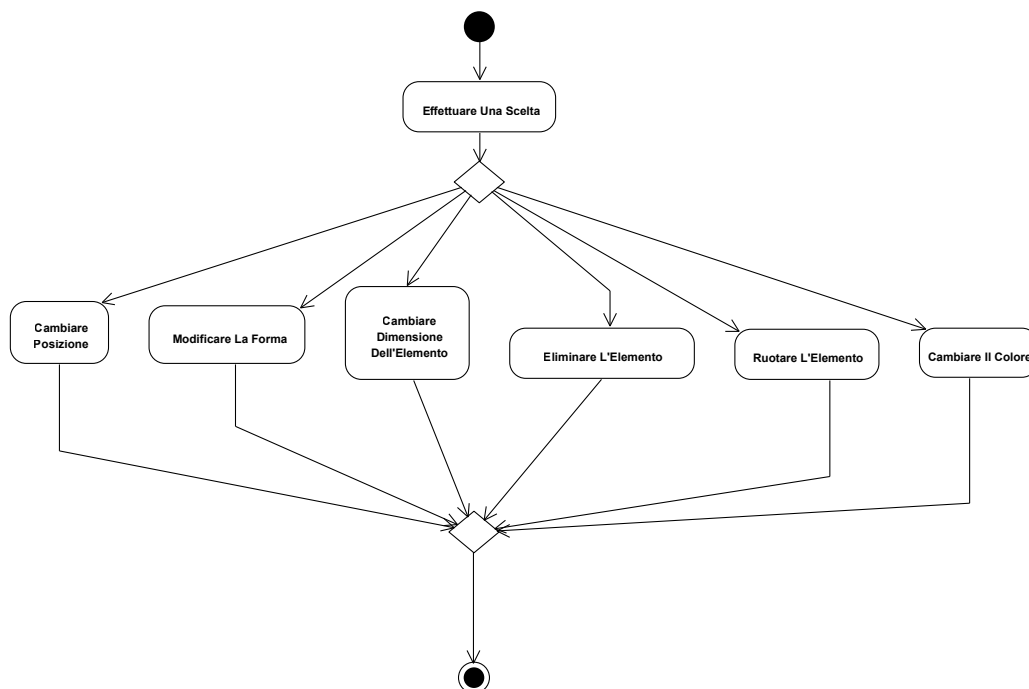


Fig 14: Modificare SVG





L'utente una volta scelto di modificare una presentazione da mobile potrà decidere di modificare un testo selezionato.





## 7 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.

Poiché 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, CSS3 e Javascript (e le sue librerie) per lo sviluppo della parte web.

Per la parte di database si è scelto l'utilizzo di MEAN e delle librerie Express.js e Node.js per una migliore interazione con MongoDB.

Per la parte di esecuzione delle presentazioni è stato scelto Impress.js, framework che permette l'esecuzione in maniera non lineare come richiesto.

Per la parte di modifica delle presentazioni verrà utilizzato il framework Angular.js per lo spostamento in tempo reale degli elementi delle presentazioni.