18-05-2015


# Specifica Tecnica

**Informazioni sul documento**

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

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

# Registro delle modifiche

Tab 1: Versionamento del documento

| Versione | Autore | Data | Descrizione |
|---|---|---|---|
| 1.0.0 | 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 |

# Storico

**RR -> RP**

| **Versione 1.0.0** | **Nominativo** |
|---|---|
| Redazione | Fossa Manuel, Tollot Pietro, Venturelli Giovanni |
| Verifica | Gabelli Pietro |
| Approvazione | Petrucci Mauro |

Tab 2: Storico ruoli pre-RR

# Indice

**1 Introduzione** **9**
1.1 Scopo del documento . . . 9
1.2 Scopo del Prodotto . . . 9
1.3 Glossario . . . 9
1.4 Riferimenti . . . 9
1.4.1 Normativi . . . 9
1.4.2 Informativi . . . 9

**2 Strumenti** **11**
2.1 HTML . . . 11
2.2 JavaScript . . . 11
2.3 jQuery . . . 11
2.4 MEAN . . . 12
2.4.1 MongoDB . . . 12
2.4.2 Express.js . . . 12
2.4.3 AngularJS . . . 12
2.4.4 Node.js . . . 12
2.5 Impress.js . . . 13

**3 Design Pattern** **14**
3.1 MVC . . . 14
3.2 Singleton . . . 14
3.2.1 Premi::Model::Command::Invoker . . . 14
3.2.2 Premi::Model::Presentazione::SlideShow . . . 14
3.3 Utility . . . 15
3.4 Builder . . . 15
3.4.1 Premi::Model::Builder . . . 15
3.5 Command . . . 15
3.5.1 Premi::Model::Command . . . 16
3.6 Iterator . . . 16
3.7 Template Method . . . 16
3.7.1 Premi::Model::Inserimento . . . 17
3.8 Strategy . . . 17
3.8.1 Premi::Model::Modifica . . . 17

**4 Descrizione architetturale** **18**
4.1 Metodo e formalismi . . . 18
4.2 Architettura generale . . . 18
4.2.1 Model . . . 18
4.2.2 View . . . 18
4.2.3 ViewModel . . . 18

**5 Descrizione dei singoli componenti** **20**
5.1 Model . . . 20
5.2 Premi::Model::SlideShow . . . 20
5.3 Premi::Model::SlideShow::ModificaSlideShow . . . 20
5.4 Premi::Model::SlideShow::SlideShowActions . . . 20
5.5 Premi::Model::SlideShow::SlideShowActions::Insert . . . 21
5.5.1 Premi::Model::SlideShow::SlideShowActions::Insert::Inserter . . . 21
5.5.2 Premi::Controller::SlideShow::Insert::ConcreteTextInserter . . . 21
5.5.3 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteFrameInserter 22
5.5.4 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteSvgInserter . 22
5.5.5 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteImageInserter 22
5.5.6 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteVideoInserter 23
5.5.7 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteAudioInserter 23
5.5.8 Premi::Controller::SlideShow::Insert::ConcreteBackgroundInserter . . . 24
5.6 Premi::Model::SlideShow::SlideShowActions::Remove . . . 24
5.6.1 Premi::Model::SlideShow::SlideShowActions::Remove::Remover . . . 24
5.6.2 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteTextRemover 25
5.6.3 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteFrameRemover 25
5.6.4 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteSVGtRemover 25
5.6.5 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteImageRemover 26
5.6.6 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteVideoRemover 26
5.6.7 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteAudioRemover 27
5.6.8 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteBackgroundRemover 27
5.7 Premi::Model::SlideShow::SlideShowActions::EditElements . . . 27
5.7.1 Premi::Model::SlideShow::SlideShowActions::EditElements::Editor . . . 28
5.7.2 Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor 28
5.7.3 Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor . . 29
5.7.4 Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor 29
5.7.5 Premi::Model::SlideShow::SlideShowActions::EditElements::ContentEditor 30
5.7.6 Premi::Model::SlideShow::SlideShowActions::EditElements::ColorEditor . 30
5.7.7 Premi::Model::SlideShow::SlideShowActions::EditElements::EditorFont . 31
5.8 Premi::Model::SlideShow::SlideShowActions::Command . . . 31
5.8.1 Premi::Model::Invoker . . . 31
5.8.2 Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand 32
5.8.3 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand 33
5.8.4 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand
5.8.5 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand
5.8.6 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand 34
5.8.7 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand
5.8.8 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand 3
5.8.9 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertComm
5.8.10 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand
5.8.11 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand
5.8.12 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand
5.8.13 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand
5.8.14 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand

5.8.15 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand
5.8.16 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemoveCom
5.8.17 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand 38
5.8.18 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand
5.8.19 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand 39
5.8.20 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundComma
5.8.21 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand
5.8.22 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditFontCommand 40
5.8.23 Premi::Model::SlideShow::SlideShowElements . . . 40
5.8.24 Premi::Model::SlideShow::SlideShowElements::SlideShowElement . . . 41
5.8.25 Premi::Model::SlideShow::Text . . . 42
5.8.26 Premi::Model::SlideShow::Frame . . . 42
5.8.27 Premi::Model::SlideShow::Image . . . 43
5.8.28 Premi::Model::SlideShow::SVG . . . 44
5.8.29 Premi::Model::SlideShow::Audio . . . 44
5.8.30 Premi::Model::SlideShow::Video . . . 45
5.8.31 Premi::Model::SlideShow::Background . . . 45
5.9 Premi::Model::ServerRelations . . . 46
5.10 Premi::Model::ServerRelations::Loader . . . 46
5.10.1 Premi::Model::ServerRelations::Loader::Costruttore . . . 46
5.11 Premi::Model::ServerRelations::AccessControl . . . 47
5.11.1 Premi::Model::ServerRelations::AccessControll::Autenticazione . . . 47
5.11.2 Premi::Model::ServerRelations::AccessControll::Registrazione . . . 47
5.12 Premi::Model::ServerRelations::DbConsistency . . . 47
5.12.1 Premi::Model::ServerRelations::DbConsistency::Observer . . . 47
5.12.2 Premi::Model::ServerRelations::DbConsistency::ConcreteObserver . . . 48
5.12.3 Premi::Model::ServerRelations::DbConsistency::Subject . . . 48
5.12.4 Premi::Model::ServerRelations::DbConsistency::SubjectAudio . . . 48
5.12.5 Premi::Model::ServerRelations::DbConsistency::SubjectAudio . . . 48
5.12.6 Premi::Model::ServerRelations::DbConsistency::SubjectVideo . . . 48
5.12.7 Premi::Model::ServerRelations::DbConsistency::SubjectText . . . 49
5.12.8 Premi::Model::ServerRelations::DbConsistency::SubjectFrame . . . 49
5.12.9 Premi::Model::ServerRelations::DbConsistency::SubjectImg . . . 49
5.12.10 Premi::Model::ServerRelations::DbConsistency::SubjectSVG . . . 49
5.13 Premi::Model::Manifest . . . 49
5.13.1 Premi::Model::Manifest::GestoreManifest . . . 49
5.14 Controller . . . 50
5.14.1 Premi::Controller::Presentazione . . . 50
5.14.1.1 Premi::Controller::Presentazione::EditController . . . 50
5.14.1.2 Premi::Controller::Presentazione::HomeController . . . 51
5.14.1.3 Premi::Controller::Presentazione::ExecutionController . . . 51
5.14.2 Premi::Controller::ApacheManager . . . 51
5.15 View . . . 53
5.15.1 Premi::View::Pages::IndexPage . . . 53
5.15.2 Premi::View::Pages::Home . . . 53
5.15.3 Premi::View::Pages::Manifest . . . 54

5.15.4 Premi::View::Pages::Profile . . . . . 54
5.15.5 Premi::View::Pages::Execution . . . . . 55
5.15.6 Premi::View::Pages::Edit . . . . . 56
5.15.7 Premi::View::Pages::EditDesktop . . . . . 56
5.15.8 Premi::View::Pages::EditMobile . . . . . 57

**6 Diagrammi di attività** **59**
6.1 Attività Principali . . . . . 59
6.1.1 Gestione presentazioni . . . . . 59
6.1.2 Caricare File . . . . . 60
6.1.3 Modificare Presentazione da Desktop . . . . . 61
6.1.4 Modificare Presentazione da Mobile . . . . . 61
6.1.5 Gestire Sfondo . . . . . 62
6.1.6 Inserire Elemento . . . . . 62
6.1.7 Modificare Elemento . . . . . 63
6.1.8 Modificare Frame . . . . . 63
6.1.9 Modificare SVG . . . . . 64
6.1.10 Modificare Testo . . . . . 65

**7 Stime di fattibilità e di bisogno di risorse** **66**

# Elenco delle figure

1 Fig1 . . . 14
2 Architettura generale del sistema . . . 19
3 View . . . 53
4 Attività Principali . . . 59
5 Gestione Presentazioni . . . 60
6 Caricare File . . . 60
7 Modificare Presentazione da Desktop . . . 61
8 Modificare Presentazione da Mobile . . . 61
9 Gestire Sfondo . . . 62
10 Inserire Elemento . . . 63
11 Modificare Elemento . . . 63
12 Modificare Frame . . . 64
13 Modificare SVG . . . 64
14 Modificare Testo . . . 65

# 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 Introduzione

## 1.1 Scopo del documento

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

## 1.2 Scopo del Prodotto

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

## 1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento Glossario_v.2.0.0.pdf. Ogni occorrenza di vocaboli presenti nel Glossario è marcata da una "g" minuscola in pedice.

## 1.4 Riferimenti

### 1.4.1 Normativi

- Capitolato d'appalto C4: Premi: Software$_g$ di presentazione "better than Prezi" http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf;
- Norme di Progetto$_g$: NormeDiProgetto_v.2.0.0.pdf;
- Analisi dei Requisiti: AnalisiDeiRequisiti_v.2.0.0.pdf;
- Piano di qualifica: PianoDiQualifica_v.2.0.0.pdf;
- Piano di progetto: PianoDiProgetto_v.2.0.0.pdf.

### 1.4.2 Informativi

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

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

# 2 Strumenti

## 2.1 HTML

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

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

## 2.2 JavaScript

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

## 2.3 jQuery

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

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

## 2.4 MEAN

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

### 2.4.1 MongoDB

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

### 2.4.2 Express.js

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

### 2.4.3 AngularJS

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

### 2.4.4 Node.js

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

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

## 2.5 Impress.js

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

Fig 1: Fig1

# 3 Design Pattern

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

## 3.1 MVC

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

## 3.2 Singleton

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

- **Scopo dell'utilizzo**: viene usato il pattern Singleton per le classi che devono avere un'unica istanza durante l'esecuzione dell'applicazione;
- **Contesto d'utilizzo**: le classi che devono avere un'unica istanza sono:
  - Invoker;
  - SlideShow.

### 3.2.1 Premi::Model::Command::Invoker

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

### 3.2.2 Premi::Model::Presentazione::SlideShow

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

deShow.

## 3.3 Utility

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

## 3.4 Builder

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

- **Scopo dell'utilizzo**: viene usato il pattern Builder per separare la costruzione di un oggetto dalla sua rappresentazione e poter riusare il processo di costruzione per creare rappresentazioni differenti;

- **Contesto d'utilizzo**: viene utilizzato per il caricamento delle presentazioni e degli oggetti in essi presenti.

### 3.4.1 Premi::Model::Builder

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

## 3.5 Command

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

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

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

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

### 3.5.1 Premi::Model::Command

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

## 3.6 Iterator

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

## 3.7 Template Method

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

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

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

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

### 3.7.1 Premi::Model::Inserimento

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

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

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

## 3.8 Strategy

- **Descrizione**: Strategy è un Design Pattern che permette che il comportamento di un algoritmo sia deciso a runtime. Il pattern Strategy definisce una famiglia di algoritmi, incapsula ogni algoritmo e rende gli algoritmi intercambiabili all'interno di quella famiglia. Strategy permette all'algoritmo di cambiare indipendentemente dal client che lo utilizza;
- **Scopo dell'utilizzo**: il pattern Strategy viene usato per isolare più algoritmi che svolgono la stessa funzione dal codice che esegue la funzione;
- **Contesto d'utilizzo**: viene utilizzato per la modifica degli elementi.

### 3.8.1 Premi::Model::Modifica

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

# 4 Descrizione architetturale

## 4.1 Metodo e formalismi

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

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

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

## 4.2 Architettura generale

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

### 4.2.1 Model

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

### 4.2.2 View

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

### 4.2.3 ViewModel

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

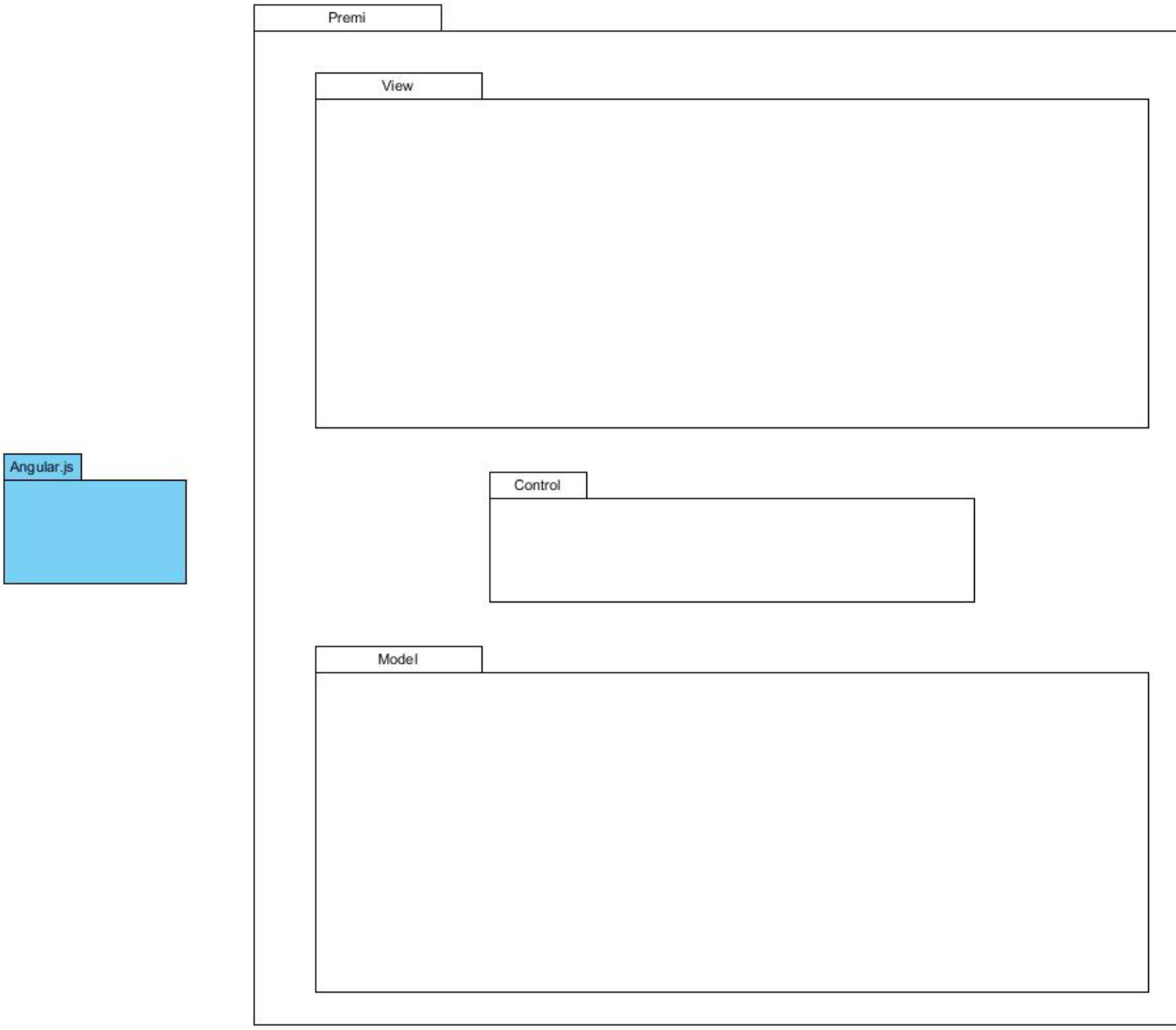


Fig 2: Architettura generale del sistema

# 5 Descrizione dei singoli componenti

## 5.1 Model

**Tipo, obiettivo e funzione del componente**: è la parte Model dell'architettura MVC.
**Relazioni d'uso di altre componenti**: è in relazione con il package Controller 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::Controller::EditController 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::Controller::EditController 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::SlideShowA 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::Insert

**Tipo, obiettivo e funzione del componente**: all'interno di questo Package viene implementato il Design Pattern template per l'inserimento di nuovi elementi nella presentazione.
**Relazioni d'uso di altre componenti**:il package è in relazione con Premi::Model::SlideShow::SlideShowA che crea gli oggetti delle classi qui presenti passando i parametri di inserimento degli elementi.//
Inoltre le classi di Premi::Model::SlideShow::SlideShowActions::Insert si occupano di costruire gli oggetti presenti nelle classi del package Premi::Model::SlideShow::SlideShowElements.

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

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteConcreteInsertCommand -> utilizza i metodi messi a disposizione da Inserter e concretizzati dalle sue sottoclassi.
- Premi::Model::SlideShow::SlideShowElements::SlideShowElement <- Inserter costruisce gli oggetti delle sottoclassi di SlideShowElement.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteTextInserter;
- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteFrameInserter;
- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteSvgInserter;
- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteImageInserter;
- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteVideoInserter;
- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteAudioInserter.
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteBackgroundInserter.

### 5.5.2 Premi::Controller::SlideShow::Insert::ConcreteTextInserter

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

- Premi::Model::SlideShow::SlideShowElements::Text <- costruisce un oggetto di classe Text.
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo testo nella presentazione.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

### 5.5.3 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteFrameInserter

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

- Premi::Model::SlideShow::SlideShowElements::Frame <- costruisce un oggetto di classe Frame.
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo Frame nella presentazione.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

### 5.5.4 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteSvgInserter

**Tipo, obiettivo e funzione del componente**: Classe che definisce l'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::SlideShow::SlideShowElements::SVG <- costruisce un oggetto di classe SVG.
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo SVG nella presentazione.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

### 5.5.5 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteImageInserter

**Tipo, obiettivo e funzione del componente**: Classe che definisce l'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::SlideShow::SlideShowElements::Image <- costruisce un oggetto di classe Image.
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo immagine nella presentazione.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

### 5.5.6 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteVideoInserter

**Tipo, obiettivo e funzione del componente**: Classe che definisce l'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::SlideShow::Video <- costruisce un oggetto di classe Video.
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo video nella presentazione.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

### 5.5.7 Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteAudioInserter

**Tipo, obiettivo e funzione del componente**: Classe che definisce l'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::SlideShow::SlideShowElements::Audio <- costruisce un oggetto di classe Audio.
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteInsertCommand -> invoca i metodi per inserire un nuovo elemento di tipo audio nella presentazione.

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

### 5.5.8 Premi::Controller::SlideShow::Insert::ConcreteBackgroundInserter

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

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

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter.

## 5.6 Premi::Model::SlideShow::SlideShowActions::Remove

**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::SlideShow::SlideShowA che crea gli oggetti delle classi qui presenti passando i parametri di rimozione degli elementi.//

### 5.6.1 Premi::Model::SlideShow::SlideShowActions::Remove::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::SlideShow::SlideShowActions::Command::ConcreteConcreteRemoveCommand -> utilizza i metodi messi a disposizione da Remover e concretizzati dalle sue sottoclassi di cui invoca anche il costruttore.

**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::SlideShow::SlideShowActions::Remove::ConcreteTextRemover;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteFrameRemover;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteSvgRemover;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteImageRemover;

- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteVideoRemover;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteAudioRemover;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteBackgroundRemover.

### 5.6.2 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteTextRemover

**Tipo, obiettivo e funzione del componente**: Classe che implementa l'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::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand -> invoca i metodi della classe per eliminare un elemento di tipo testo dalla presentazione.
- Premi::Model::SlideShow::SlideShowElements::Text <- invoca il distruttore di Text

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

- Premi::Model::SlideShow::SlideShowActions::Remove::Remover.

### 5.6.3 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteFrameRemover

**Tipo, obiettivo e funzione del componente**: Classe che implementa l'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::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand -> invoca i metodi della classe per eliminare un elemento di tipo frame dalla presentazione;
- Premi::Model::SlideShow::SlideShowElements::Frame <- invoca il distruttore di Frame.

**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::SlideShow::SlideShowActions::Remove::Remover.

### 5.6.4 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteSVGtRemover

**Tipo, obiettivo e funzione del componente**: Classe che implementa l'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::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand -> invoca i metodi della classe per eliminare un elemento di tipo SVG dalla presentazione;

- Premi::Model::SlideShow::SlideShowElements::SVG <- invoca il distruttore di SVG.

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

- Premi::Model::SlideShow::SlideShowActions::Remove::Remover.

### 5.6.5 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteImageRemover

**Tipo, obiettivo e funzione del componente**: Classe che implementa l'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::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand -> invoca i metodi della classe per eliminare un elemento di tipo immagine dalla presentazione.

- Premi::Model::SlideShow::SlideShowElements::Image <- invoca il distruttore di Image.

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

- Premi::Model::SlideShow::SlideShowActions::Remove::Remover.

### 5.6.6 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteVideoRemover

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoCommand -> invoca i metodi di ConcreteVideoRemover per eliminare un elemento di tipo video dalla presentazione;

- Premi::Model::SlideShow::SlideShowElements::Video <- invoca il distruttore di Video.

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

- Premi::Model::SlideShow::SlideShowActions::Remove::Remover.

### 5.6.7 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteAudioRemover

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteRemoveCommand -> invoca i metodi di ConcreteAudioRemover per eliminare un elemento di tipo audio dalla presentazione.
- Premi::Model::SlideShow::SlideShowElements::Audio <- ConcreteAudioRemover invoca il distruttore di Audio.

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

- Premi::Model::SlideShow::SlideShowActions::Remove::Remover.

### 5.6.8 Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteBackgroundRemover

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteRemoveCommand -> invoca i metodi per eliminare lo sfondo dalla presentazione.
- Premi::Model::SlideShow::SlideShowElements::Background <- ConcreteBackgroundRemover invoca il distruttore di Background.

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

- Premi::Model::SlideShow::SlideShowActions::Remove::Remover.

## 5.7 Premi::Model::SlideShow::SlideShowActions::EditElements

**Tipo, obiettivo e funzione del componente**: All'interno di questo Package viene implementato il Design Pattern strategy per la modifica di elementi della presentazione.
**Relazioni d'uso di altre componenti**:il package è in relazione con Premi::Model::SlideShow::SlideShowA che crea gli oggetti delle classi qui presenti passando i parametri di modifica degli elementi.

### 5.7.1 Premi::Model::SlideShow::SlideShowActions::EditElements::Editor

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

- Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand -> Invoca i costruttori delle sottoclassi di Editor;
- Premi::Model::ServerRelation::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento da modificare;
- Premi::Model::SlideShow::SlideShowElements::SlideShowElement <- invoca i metodi della classe SlideShowElement per modificare opportunamente i campi dell'elemento.

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

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor::PositionEditor;
- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor::SizeEditor;
- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor::ContentEditor;
- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor::RotationEditor;
- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor::ColorEditor;
- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor::EditorShape.

### 5.7.2 Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditCommand -> Invoca il costruttore di PositionEditor;
- Premi::Model::ServerRelation::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento da modificare;
- Premi::Model::SlideShow::SlideShowElements::SlideShowElement <- invoca le funzioni della classe SlideShowElement per modificare opportunamente i campi relativi alla posizione dell'elemento.

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

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor.

### 5.7.3 Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor

**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::SlideShow::SlideShowActions::Command::AbstractCommand -> Invoca il costruttore di SizeEditor;
- Premi::Model::ServerRelation::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento da modificare;
- Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::C invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, SizeEditor invocherà quindi i metodi di modifica delle dimensioni forniti all'interno della sottoclasse di Premi::Model::SlideShow::SlideShowElement di cui fa parte l'oggetto da modificare.
**Classi ereditate**:

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor.

### 5.7.4 Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor

**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::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand -> Invoca il costruttore di RotationEditor;
- Premi::Model::ServerRelation::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento da modificare;
- Premi::Model::SlideShow::SlideShowElements::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::SlideShow::SlideShowActions::C invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, PositionEditor invocherà quindi i metodi di modifica dell'inclinazione forniti all'interno della sottoclasse di Premi::Model::SlideShow::SlideShowElements::SlideShowElement di cui fa parte l'oggetto da modificare.
**Classi ereditate**:

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor.

### 5.7.5 Premi::Model::SlideShow::SlideShowActions::EditElements::ContentEditor

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditShapeCommand -> Invoca il costruttore di ContentEditor;
- Premi::Model::ServerRelations::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento testuale da modificare;
- Premi::Model::SlideShow::SlideShowElements::Text <- invoca le funzioni della classe Text per modificare opportunamente i campi relativi alla contenuto dell'elemento testuale.

**Interfacce con e relazioni d'uso e da altre componenti**:Premi::Model::SlideShow::SlideShowActions::C
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, ContentEditor invocherà quindi i metodi di modifica del contenuto forniti all'interno della classe Premi::Model::SlideShow::Slide
**Classi ereditate**:

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor.

### 5.7.6 Premi::Model::SlideShow::SlideShowActions::EditElements::ColorEditor

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand -> Invoca il costruttore di ColorEditor;
- Premi::Model::ServerRelations::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento SVG da modificare;
- Premi::Model::SlideShow::SlideShowElements::SlideShowElement <- invoca le funzioni della sottoclasse di SlideShowElement per modificare opportunamente i campi relativi alla forma dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**:Premi::Model::SlideShow::SlideShowActions::C
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, ColorEditor invocherà quindi i metodi di modifica della forma forniti dalla sottoclasse classe di Premi::Model::SlideShow::SlideSh
**Classi ereditate**:

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor.

### 5.7.7 Premi::Model::SlideShow::SlideShowActions::EditElements::EditorFont

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

- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditFontCommand -> Invoca il costruttore di EditorFont;
- Premi::Model::ServerRelations::Loader::Costruttore <- scorre gli elementi del membro presentazione all'interno di Costruttore per trovare l'elemento testuale da modificare;
- Premi::Model::SlideShow::SlideShowElements::SlideShowElement <- invoca le funzioni della sottoclasse di SlideShowElement per modificare opportunamente i campi relativi alla forma dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**:Premi::Model::SlideShow::SlideShowActions::C
invoca il costruttore e la funzione di esecuzione dell'operazione di modifica, EditorFont invocherà quindi i metodi di modifica della forma forniti dalla classe Premi::Model::SlideShow::SlideShowElements::
**Classi ereditate**:

- Premi::Model::SlideShow::SlideShowActions::EditElements::Editor.

## 5.8 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::Sli
Il package comunica, inoltre, con il controller, infatti le sue classi sono generate da Premi::Controller::SlideShow::EditController.

### 5.8.1 Premi::Model::Invoker

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

- Premi::Controller::MobileEdit->crea un oggetto di una sottoclasse di Premi::Model::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::Controller::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.

### 5.8.2 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;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemoveCommand;
- Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditCommand.

### 5.8.3 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertComman

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

- Premi::Controller::SlideShow::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::SlideShow::SlideShowActions::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::Insert::TextInserter <- 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 testuale.
**Classi ereditate**:

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

### 5.8.4 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertComma

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

- Premi::Controller::SlideShow::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::SlideShow::SlideShowActions::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::Insert::FrameInserter <- 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 frame.
**Classi ereditate**:

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

### 5.8.5 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertComma

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

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti l'inserimento di un nuovo elemento immagine.
**Classi ereditate**:

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

### 5.8.6 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertComman

**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::Controller::SlideShow::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::SlideShow::SlideShowActions::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::Insert::SVGInserter <- 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 SVG.
**Classi ereditate**:

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

### 5.8.7 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertComma

**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::Controller::SlideShow::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::SlideShow::SlideShowActions::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::Insert::AudioInserter <- 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 Audio.
**Classi ereditate**:

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

### 5.8.8 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertComma

**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::Controller::SlideShow::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::SlideShow::SlideShowActions::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::Insert::VideoInserter <- 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 video.
**Classi ereditate**:

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

#### 5.8.9 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::Controller::SlideShow::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker; Premi::Model::SlideShow::SlideShowActions::Invoker -> esegue il comando o ne invoca il metodo di annullamento;
- Premi::Model::SlideShow::SlideShowActions::Insert::VideoInserter <- 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 sfondo.
**Classi ereditate**:

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

#### 5.8.10 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveComm

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::ConcreteTextRemover <- invoca la classe concreta del template per l'eliminazione di un elemento testuale.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i segnali riguardanti l'eliminazione di un elemento testuale.
**Classi ereditate**:

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

### 5.8.11 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCon

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::Remover <- invoca la classe concreta del template per l'eliminazione di un elemento frame.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti l'eliminazione di un elemento frame.
**Classi ereditate**:

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

### 5.8.12 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCon

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::ConcreteImageRemover <- invoca la classe concreta del template per l'eliminazione di un elemento immagine.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i segnali riguardanti l'eliminazione di un elemento immagine.
**Classi ereditate**:

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

### 5.8.13 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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::ConcreteSVGRemover <- invoca la classe concreta del template per l'eliminazione di un elemento SVG.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i segnali riguardanti l'eliminazione di un elemento SVG.
**Classi ereditate**:

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

#### 5.8.14 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCon

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::ConcreteAudioRemover <- invoca la classe concreta del template per l'eliminazione di un elemento audio.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i segnali riguardanti l'eliminazione di un elemento audio.
**Classi ereditate**:

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

#### 5.8.15 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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::ConcreteVideoRemover <- invoca la classe concreta del template per l'eliminazione di un elemento video.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i segnali riguardanti l'eliminazione di un elemento video.
**Classi ereditate**:

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

#### 5.8.16 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::Controller::SlideShow::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::SlideShow::SlideShowActions::Remove::ConcreteBackgroundRemover <- costruisce un oggetto della classe concreta del template per l'eliminazione di un elemento immagine.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i segnali riguardanti l'eliminazione di un elemento sfondo.
**Classi ereditate**:

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

#### 5.8.17 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeComman

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::EditElements::SizeEditor <- invoca la classe concreta del design pattern Strategy per la modifica delle dimensioni di un elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti la modifica delle dimensioni di un elemento;
**Classi ereditate**:

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

#### 5.8.18 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionComm

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::EditElements::PositionEditor <- invoca la classe concreta del design pattern Strategy per la modifica della posizione di un elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti la modifica della posizione di un elemento;
**Classi ereditate**:

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

#### 5.8.19 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditColorComma

**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::Controller::SlideShow::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::SlideShow::SlideShowActions::EditElements::ColorEditor <- invoca la classe concreta del design pattern Strategy per la modifica del colore di un elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti la modifica del colore di un elemento;
**Classi ereditate**:

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

#### 5.8.20 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::Controller::SlideShow::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::SlideShow::SlideShowActions::EditElements::BackgroundEditor <- invoca la classe concreta del design pattern Strategy per la modifica dello sfondo di un elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti la modifica dello sfondo di un elemento;
**Classi ereditate**:

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

#### 5.8.21 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::Controller::SlideShow::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::SlideShow::SlideShowActions::EditElements::RotationEditor <- invoca la classe concreta del design pattern Strategy per la modifica dell'orientamento di un elemento.

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

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

#### 5.8.22 Premi::Model::SlideShow::SlideShowActions::Command::ConcreteEditFontComman

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

- Premi::Controller::SlideShow::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::SlideShow::SlideShowActions::EditElements::FontEditor <- invoca la classe concreta del design pattern Strategy per la modifica del carattere di un elemento testuale.

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti la modifica del carattere di un testo;
**Classi ereditate**:

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

#### 5.8.23 Premi::Model::SlideShow::SlideShowElements

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

- 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;

#### 5.8.24 Premi::Model::SlideShow::SlideShowElements::SlideShowElement

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

- Premi::Model::SlideShow::SlideShowActions::Insert::Inserter-> invoca il costruttore delle sottoclassi di SlideShowElement e li inserisce nei membri 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: instanzia oggetti di sottoclassi di SlideShowElement e li inserisce nel membro 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.

### 5.8.25 Premi::Model::SlideShow::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::Insert::ConcreteTextInserter -> invoca il costruttore di Text e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteTextRemover -> rimuove l'oggetto Text dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader::Costru ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.
- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::ContentEditor -> invoca i metodi che impostano i campi che descrivono il contenuto dell'elemento testuale.
- Premi::Model::SlideShow::SlideShowActions::EditElements::FontEditor -> invoca i metodi che impostano i campi che descrivono il carattere dell'elemento testuale.

**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 membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costrut
**Classi ereditate**:

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

### 5.8.26 Premi::Model::SlideShow::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::Insert::ConcreteFrameInserter -> invoca il costruttore di Frame e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteFrameRemover -> rimuove l'oggetto Frame dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader::Co ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.

- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::BackgroundEditor -> invoca i metodi che impostano il campo che definisce lo sfondo dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Gli oggetti della classe Frame vengono istanziati da Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteFrameInserter inseriti nel membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costrut
**Classi ereditate**:

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

### 5.8.27 Premi::Model::SlideShow::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::Insert::ConcreteImageInserter -> invoca il costruttore di Image e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteImageRemover -> rimuove l'oggetto Image dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader::Co ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.
- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Gli oggetti della classe Image vengono istanziati da Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteImageInserter inseriti nel membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costrut
**Classi ereditate**:

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

### 5.8.28 Premi::Model::SlideShow::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::Insert::ConcreteSVGInserter -> invoca il costruttore di SVG e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteImageRemover -> rimuove l'oggetto SVG dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader::Cos ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.
- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::ColorEditor -> invoca i metodi che impostano i campi che individuano il colore dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Gli oggetti della classe SVG vengono istanziati da Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteSVGInserter e inseriti nel membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costrut
**Classi ereditate**:

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

### 5.8.29 Premi::Model::SlideShow::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::Insert::ConcreteAudioInserter -> invoca il costruttore di Audio e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteAudioRemover -> rimuove l'oggetto Audio dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader::Co ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.
- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.

- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Gli oggetti della classe Audio vengono istanziati da Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteAudioInserter e inseriti nel membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costrut
**Classi ereditate**:

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

### 5.8.30 Premi::Model::SlideShow::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::SlideShow::SlideShowActions::Insert::ConcreteVideoInserter -> invoca il costruttore di Video e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;
- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteVideoRemover -> rimuove l'oggetto Video dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader::Co ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.
- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.

**Interfacce con e relazioni d'uso e da altre componenti**: Gli oggetti della classe Video vengono istanziati da Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteVideoInserter e inseriti nel membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costrut
**Classi ereditate**:

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

### 5.8.31 Premi::Model::SlideShow::Background

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

- Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteBackgroundInserter -> invoca il costruttore di Background e inserisce l'oggetto nel membro contenitore all'interno dell'oggetto della classe Premi::Model::SlideShow::SlideShow;

- Premi::Model::SlideShow::SlideShowActions::Remove::ConcreteBackgroundRemover -> rimuove l'oggetto Video dal membro presentazione all'interno di Premi::Model::ServerRelations::Loader ne invoca quindi il distruttore;
- Premi::Model::SlideShow::SlideShowActions::EditElements::SizeEditor -> invoca i metodi che impostano i campi height e width dell'oggetto.
- Premi::Model::SlideShow::SlideShowActions::EditElements::PositionEditor -> invoca i metodi che impostano i campi che individuano le coordinate dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::RotationEditor -> invoca i metodi che impostano i campi che individuano l'orientamento dell'elemento.
- Premi::Model::SlideShow::SlideShowActions::EditElements::BackgroundEditor -> invoca i metodi che impostano i campi che definiscono l'immagine o il colore dell'elemento sfondo.

**Interfacce con e relazioni d'uso e da altre componenti**: Gli oggetti della classe Background vengono istanziati da Premi::Model::SlideShow::SlideShowActions::Insert::ConcreteBacgroundInser e inseriti nel membro contenitore presentazione all'interno di Premi::Model::ServerRelations::Loader::Costru
**Classi ereditate**:

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

## 5.9 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'e' dipenda tra il package serverRelations ed altri package del model.

## 5.10 Premi::Model::ServerRelations::Loader

**Tipo, obiettivo e funzione del componente**: il package racchiude le funzioni di recupero di una presentazione dal server attraversi 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.

### 5.10.1 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.

## 5.11 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.

### 5.11.1 Premi::Model::ServerRelations::AccessControll::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::Controller::Pagine::IndexController -> invoca i metodi di Autenticazione per permettere all'utente di effettuare il login.

### 5.11.2 Premi::Model::ServerRelations::AccessControll::Registrazione

**Tipo, obiettivo e funzione del componente**: Classe, fornisce funzionalità di registrazione all'utente.
**Relazioni d'uso di altre componenti**:

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

## 5.12 Premi::Model::ServerRelations::DbConsistency

**Tipo, obiettivo e funzione del componente**: il package ha lo scopo di raccogliere le funzionalità di aggiornamento delle presentazioni in remoto tramite un pattern observer e chiamate asincrone ai servizi di nodeApi
**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.

### 5.12.1 Premi::Model::ServerRelations::DbConsistency::Observer

**Tipo, obiettivo e funzione del componente**: Interfaccia, espone il metodo update(), utile per l'implementazione del design pattern "Observer".
**Relazioni d'uso di altre componenti**:

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

### 5.12.2 Premi::Model::ServerRelations::DbConsistency::ConcreteObserver

**Tipo, obiettivo e funzione del componente**: Classe, concretizza l'interfaccia Observer, utile ad implementare il pattern "Observer".
**Relazioni d'uso di altre componenti**:

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

### 5.12.3 Premi::Model::ServerRelations::DbConsistency::Subject

**Tipo, obiettivo e funzione del componente**: Classe astratta, definisce una classe astratta per i diversi tipi di subject a seconda degli elementi da osservare. Definisce i metodi attach(Observer), detach(Observer) e notify() per implementare il pattern "Observer".
**Relazioni d'uso di altre componenti**:

- 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.

### 5.12.4 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.

### 5.12.5 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.

### 5.12.6 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.

### 5.12.7 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.

### 5.12.8 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.

### 5.12.9 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.

### 5.12.10 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.

## 5.13 Premi::Model::Manifest

**Tipo, obiettivo e funzione del componente**: Questo package ha lo scopo di rendere disponibili le presentazioni in locale tramite chiamate al server Apache. **Relazioni d'uso di altre componenti**:

- definisce il metodo GestoreManifest()

### 5.13.1 Premi::Model::Manifest::GestoreManifest

**Tipo, obiettivo e funzione del componente**: classe, fornisce un'implementazione del package; **Relazioni d'uso di altre componenti**:

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

## 5.14 Premi::Model::GestioneFileServer

**Tipo, obiettivo e funzione del componente**: Questo package gestisce della gestione dei file dell'utente sul server Apache; istanzia il la classe GestioneFile per **Relazioni d'uso di altre componenti**:

- 

**Interfacce con e relazioni d'uso e da altre componenti**: Viene utilizzata per gestire i Signal riguardanti l'inserimento di un nuovo elemento testuale.

### 5.14.1 Premi::Model::GestioneFileServer::GestioneFile

**Tipo, obiettivo e funzione del componente**: classe; fornisce un'implementazione del package; **Relazioni d'uso di altre componenti**:

- Premi::Controller::EditController, Premi::View::Pages::Profile;
- definisce il metodo inserisciFile(), eliminaFile(), rinominaFile().

## 5.15 Controller

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

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

### 5.15.1 Premi::Controller::Presentazione

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

#### 5.15.1.1 Premi::Controller::Presentazione::EditController

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

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

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

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

#### 5.15.1.2 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.

#### 5.15.1.3 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.

### 5.15.2 Premi::Controller::ApacheManager

**Tipo, obiettivo e funzione del componente**: Lo scopo di questa classe è di gestire i segnali della pagina Premi::View::Pages::Profile per l'inserimento, cancellazione e rinominazione di file sul server Apache.
**Relazioni d'uso di altre componenti**:

- 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.

**Interfacce con e relazioni d'uso e da altre componenti**: La pagine Profile costruisce ApacheManager per fare modifiche ai file.

## 5.16 View

Fig 3: View

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

### 5.16.1 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.

### 5.16.2 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 su 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.

### 5.16.3 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.

### 5.16.4 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.

### 5.16.5 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.

### 5.16.6 Premi::View::Pages::Edit

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

### 5.16.7 Premi::View::Pages::EditDesktop

**Tipo, obiettivo e funzione del componente**: la classe EditDesktop definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra da dispositivo desktop ad un utente l'editor di modifica di una presentazione.
**Relazioni d'uso di altre componenti**: mandi principali di gestione del profilo e gestione presentazioni.

**Relazioni d'uso di altre componenti**: la classe 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.
**Interfacce con e relazioni d'uso e da altre componenti**: i metodi implementati nella classe EditDesktop sono i seguenti:

- EditDesktop::InsertFrame invia al controller la richiesta di inserimento di un nuovo frame, la sua forma, le coordinate di posizione;
- EditDesktop::InsertMedia invia al controller la richiesta di inserimento di un nuovo file media, le sue informazioni e le coordinate di posizione e di rotazione;
- EditDesktop::MoveElement invia al controller l'id dell'elemento spostato e le sue nuove coordinate;
- EditDesktop::InsertText invia al controller la richiesta di inserimento di un nuovo elemento di testo, il suo contenuto, la sua formattazione e le sue coordinate;
- EditDesktop::TextEdit invia al controller l'id dell'elemento di testo e il suo nuovo contenuto;
- EditDesktop::DeleteElement invia al controller l'id dell'elemento eliminato;
- EditDesktop::InsertChoice invia al controller la richiesta di inserimento di una nuova scelta e l'id del frame a cui è indirizzata la scelta;
- EditDesktop::Bookmark invia al controller l'id del frame al quale viene associato o rimosso (a seconda dello stato in quel momento) un bookmark;
- EditDesktop::ChangeSize invia al controller l'id dell'elemento al quale vengono cambiate le dimensioni e le nuove misure;
- EditDesktop::ChangeRotation invia al controller l'id dell'elemento al quale viene cambiata la rotazione la percentuale di rotazione;
- EditDesktop::ChangePath invia al controller l'id del percorso modificato e il nuovo ordine dei frame.
- EditDesktop::FrameBackground invia al controller la richiesta di inserimento di un nuovo sfondo ad un frame, l'id del frame e le informazioni dell'immagine;
- EditDesktop::Background invia al controller la richiesta di inserimento di un nuovo sfondo alla presentazione e le informazioni dell'immagine;
- 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.

### 5.16.8 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.

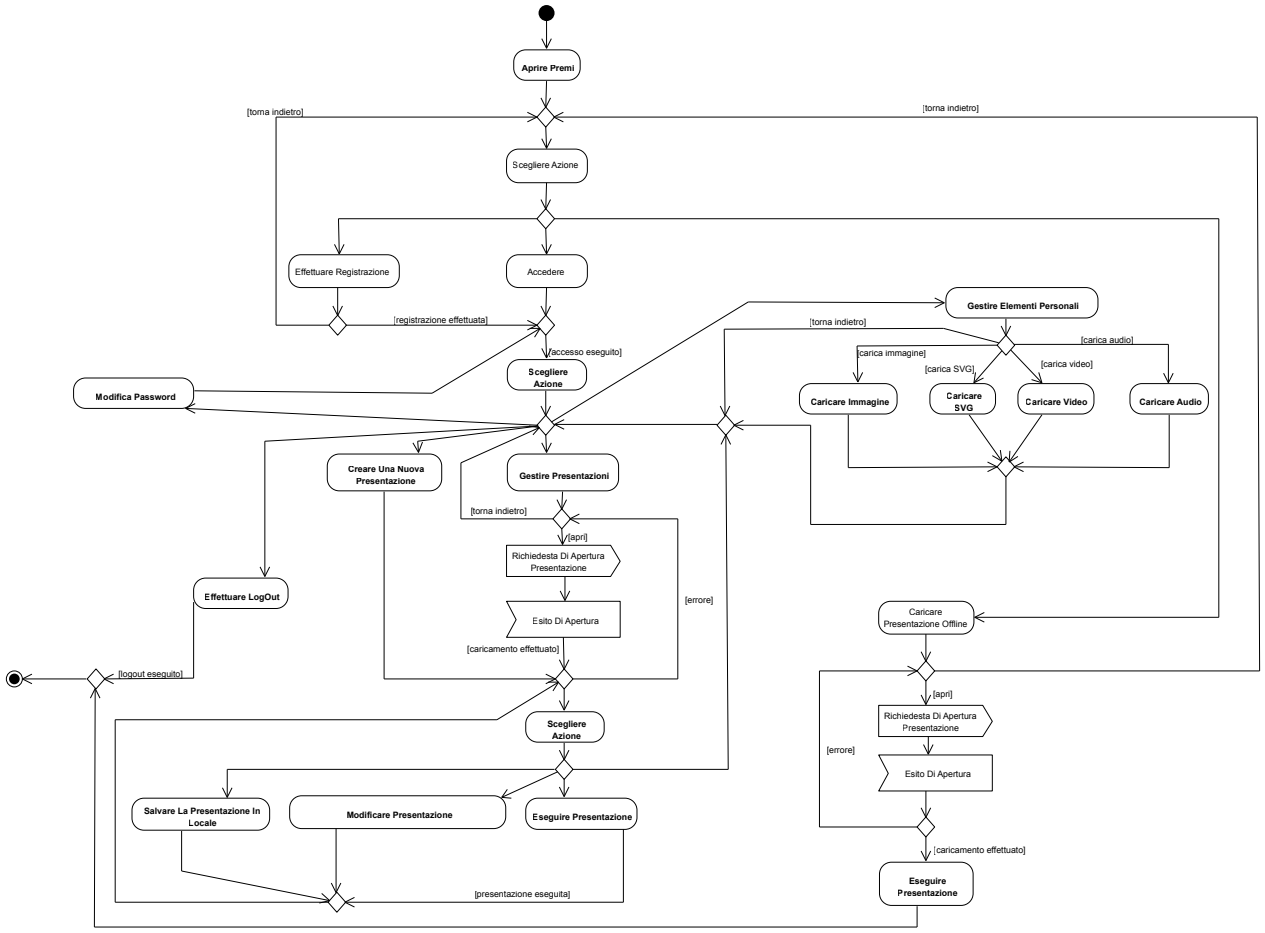


Fig 4: Attività Principali

### 6.1.1 Gestione presentazioni

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

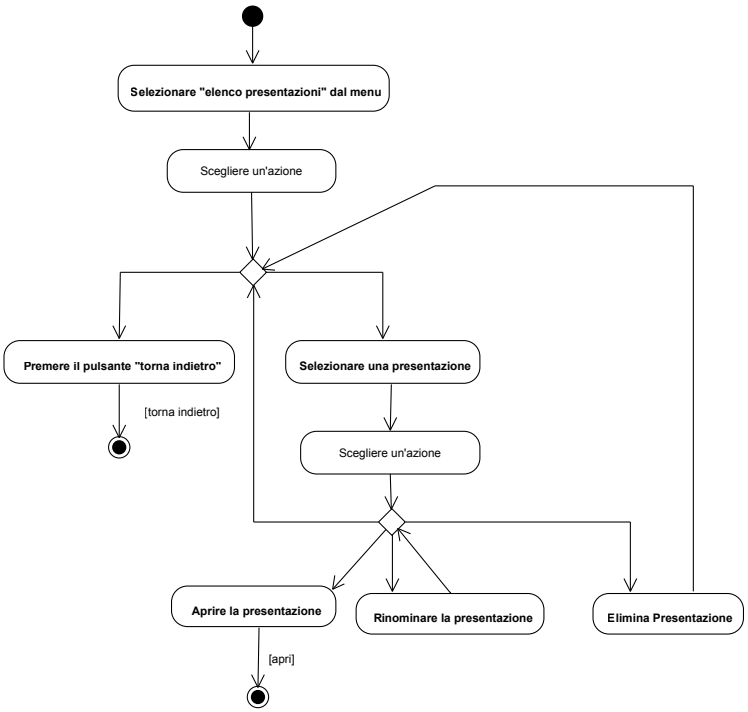


Fig 5: Gestione Presentazioni

### 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.

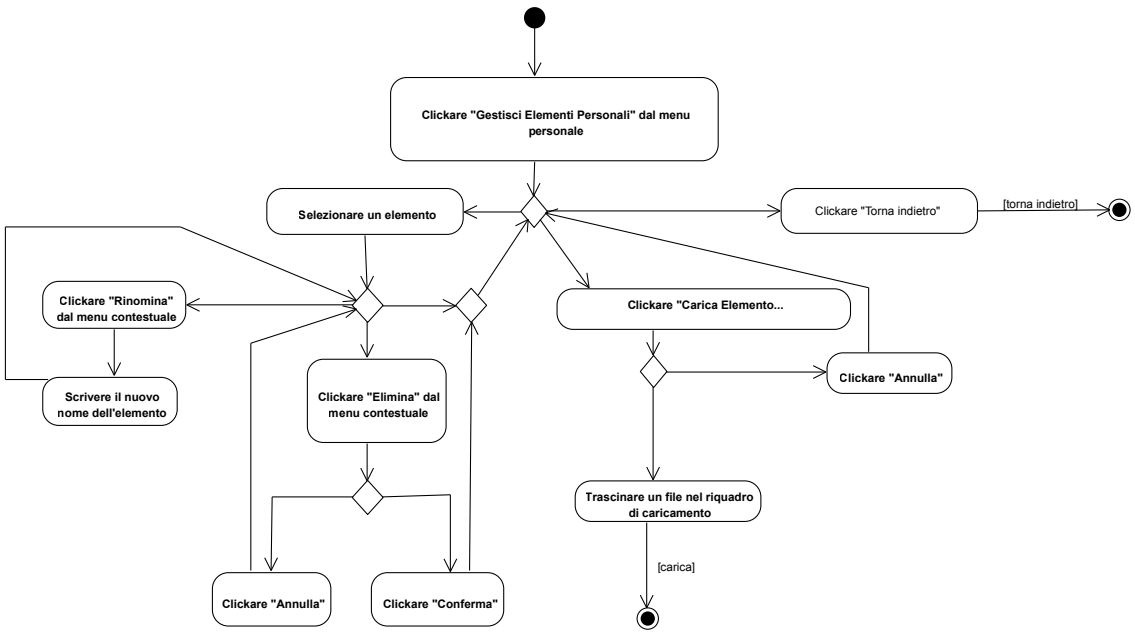


Fig 6: Caricare File

### 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.

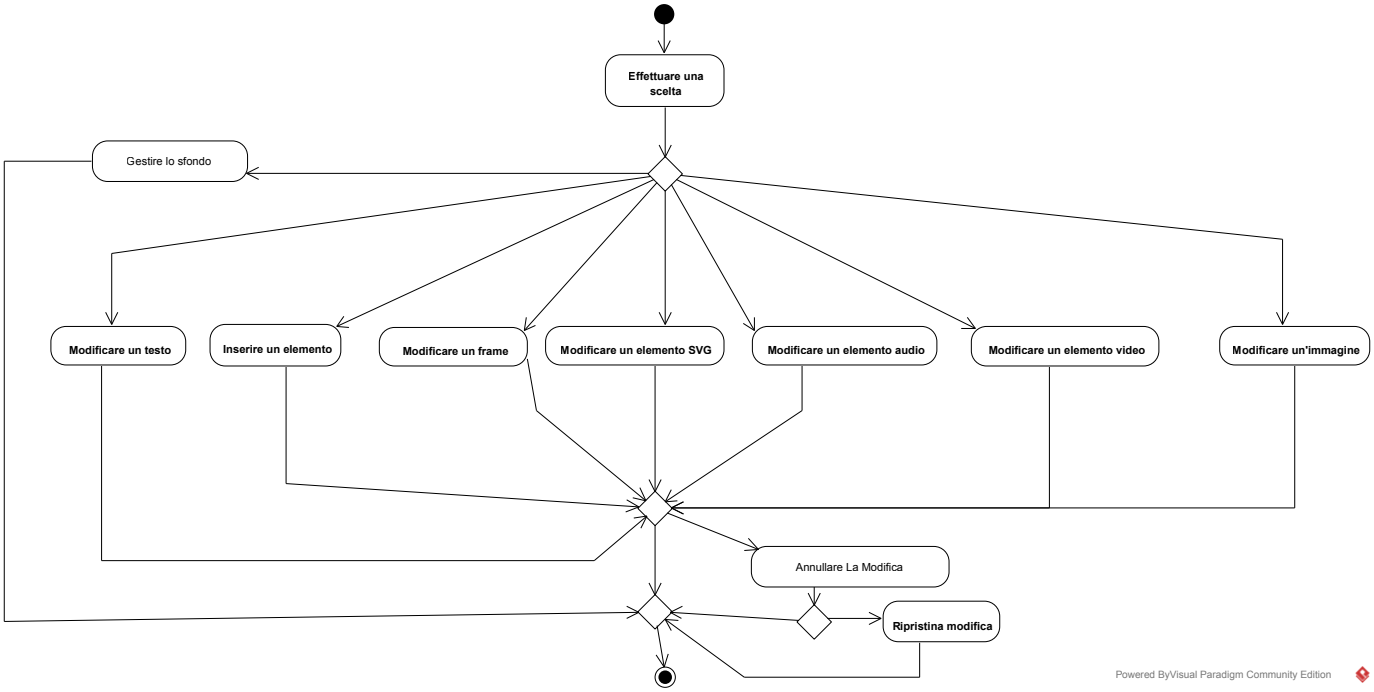


Fig 7: Modificare Presentazione da Desktop

### 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.

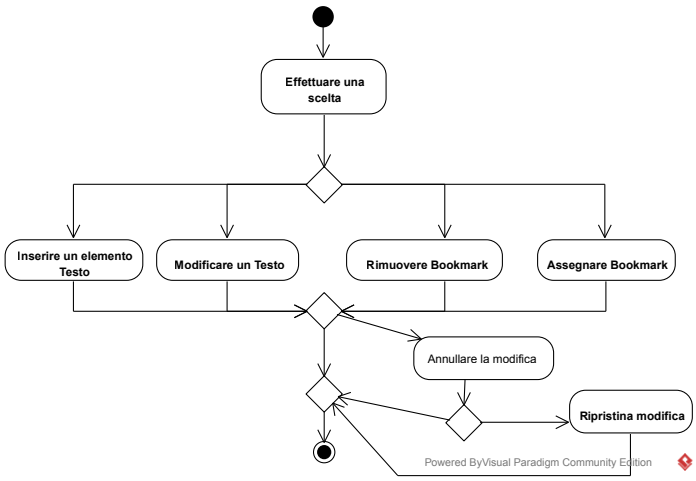


Fig 8: Modificare Presentazione da Mobile

### 6.1.5 Gestire Sfondo

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

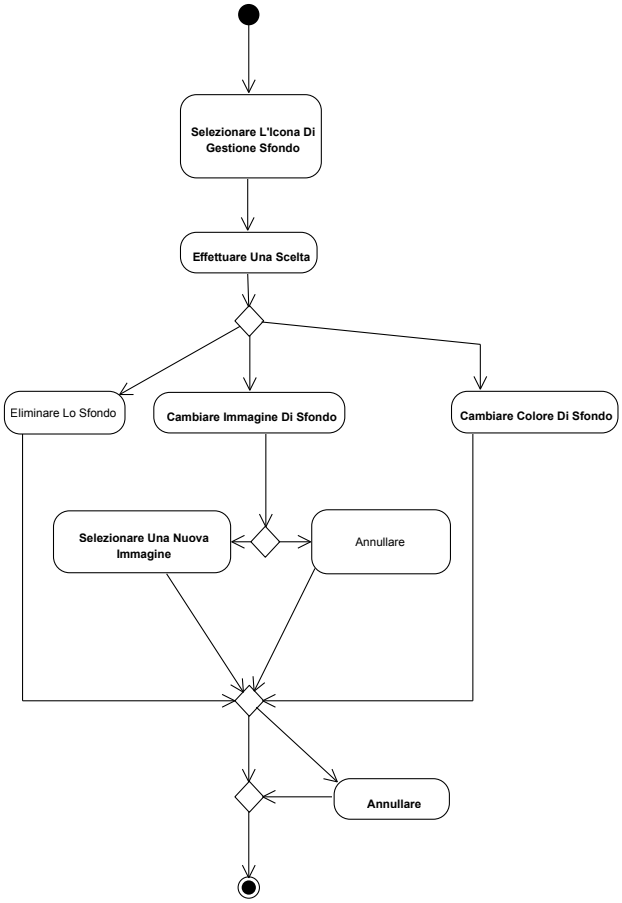


Fig 9: Gestire Sfondo

### 6.1.6 Inserire Elemento

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

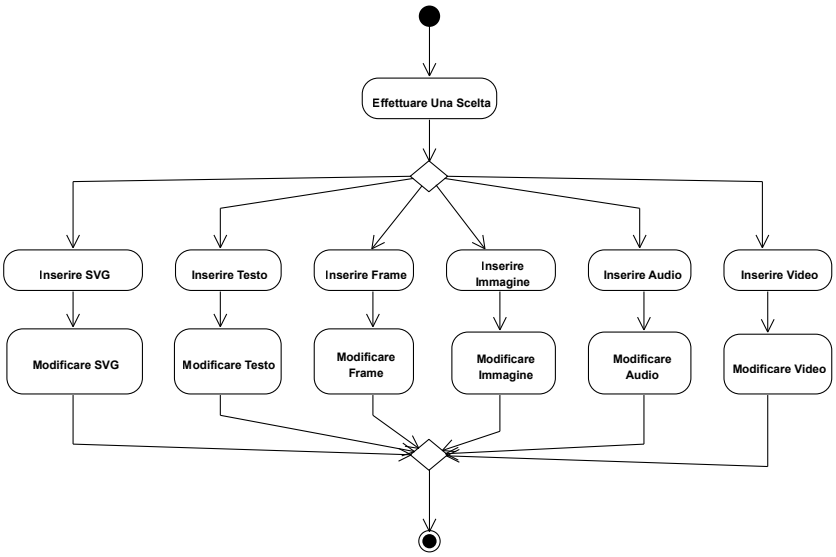


Fig 10: Inserire Elemento

### 6.1.7 Modificare Elemento

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

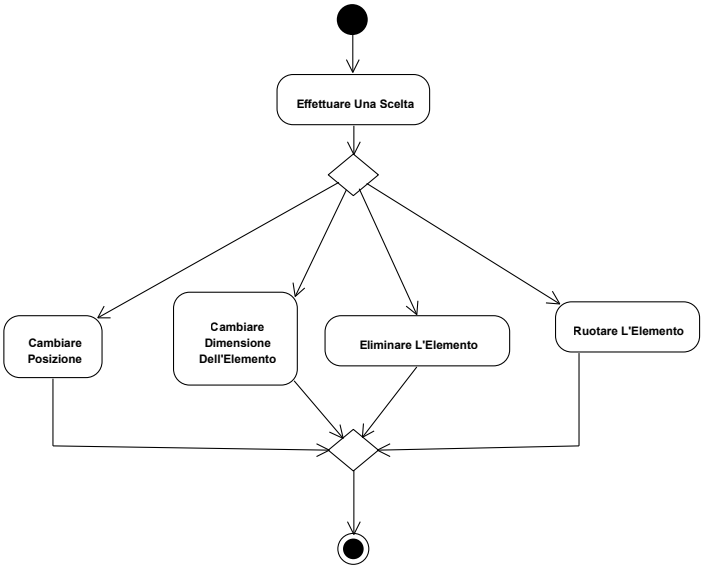


Fig 11: Modificare Elemento

### 6.1.8 Modificare Frame

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

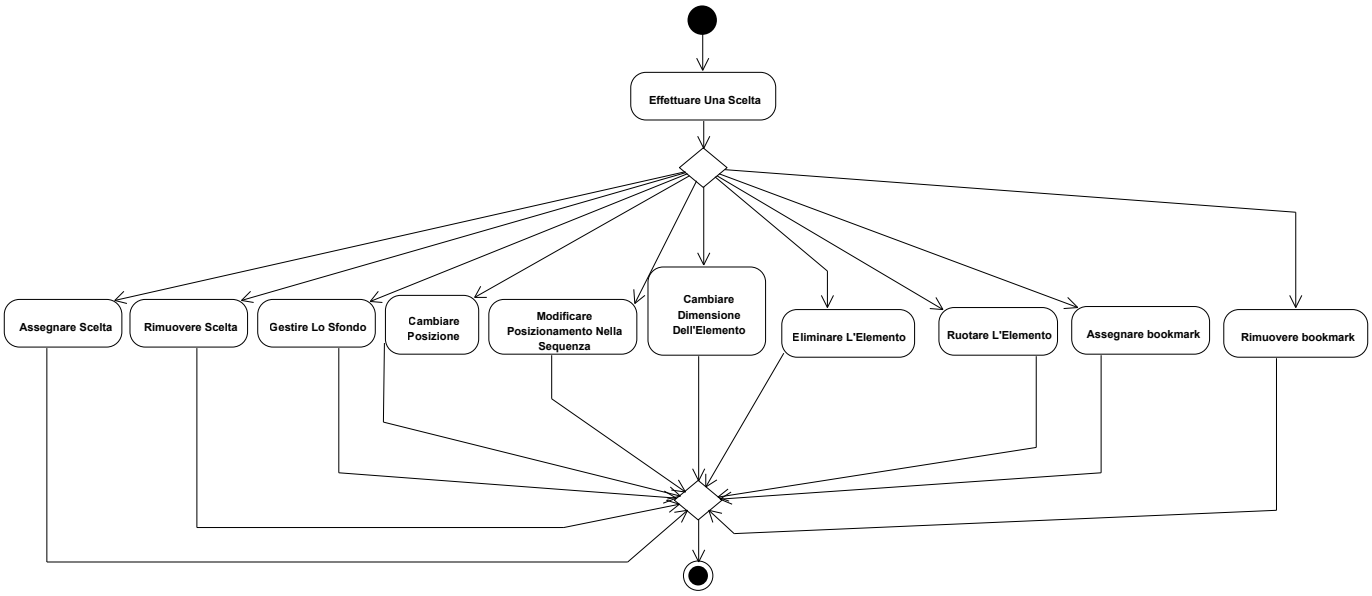


Fig 12: Modificare Frame

### 6.1.9 Modificare SVG

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

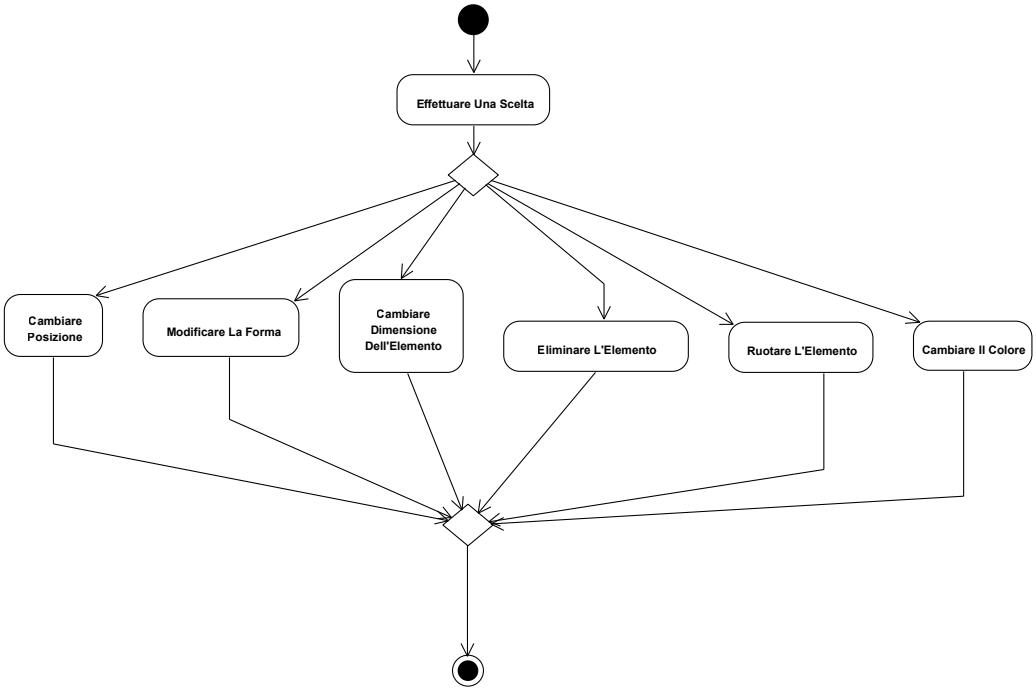


Fig 13: Modificare SVG

### 6.1.10 Modificare Testo

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

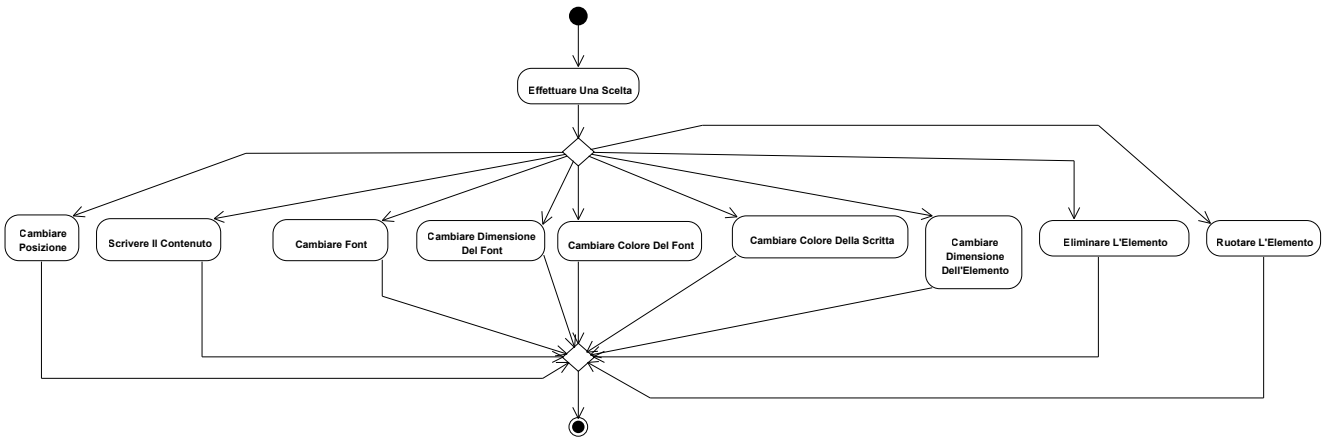


Fig 14: Modificare Testo

# 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.
Poiche 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.