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 RR -> RP

Indice

Elenco delle figure

Elenco delle tabelle

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.

3 Design Pattern e Pattern Architetturali

3.1 MVP

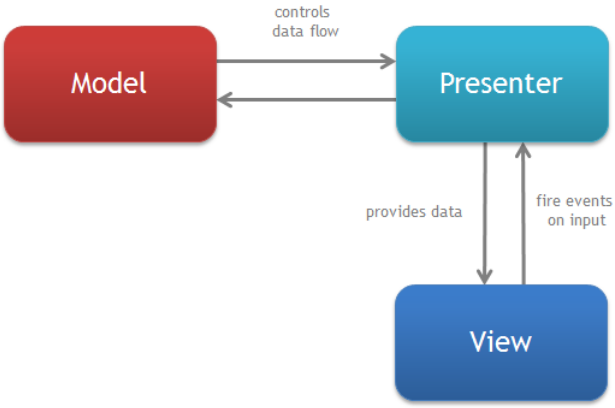

Fig 1: Model View Presenter

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

3.2 Command

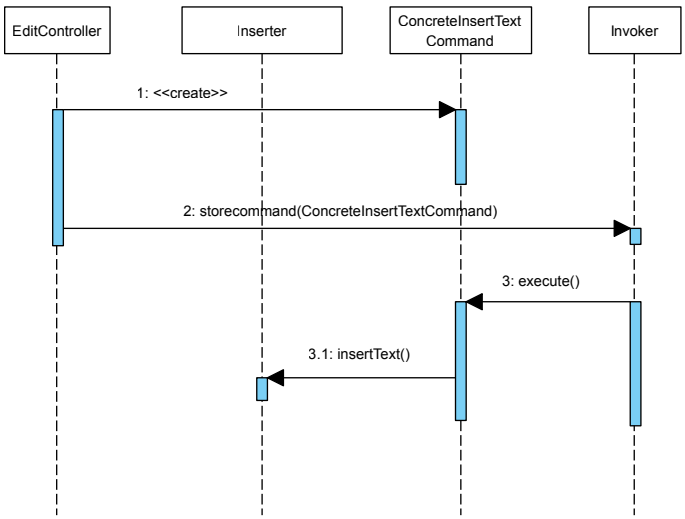

Fig 2: Command

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

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

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

- **Scopo dell'utilizzo**: si è scelto di utilizzare il pattern Command perché poter accodare o mantenere uno storico delle operazioni e gestire operazioni cancellabili;

- **Contesto d'utilizzo**: viene utilizzato in fase di modifica delle presentazioni.

3.2.1 Premi::Model::SlideShow::SlideShowActions::Command

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

3.3 Observer

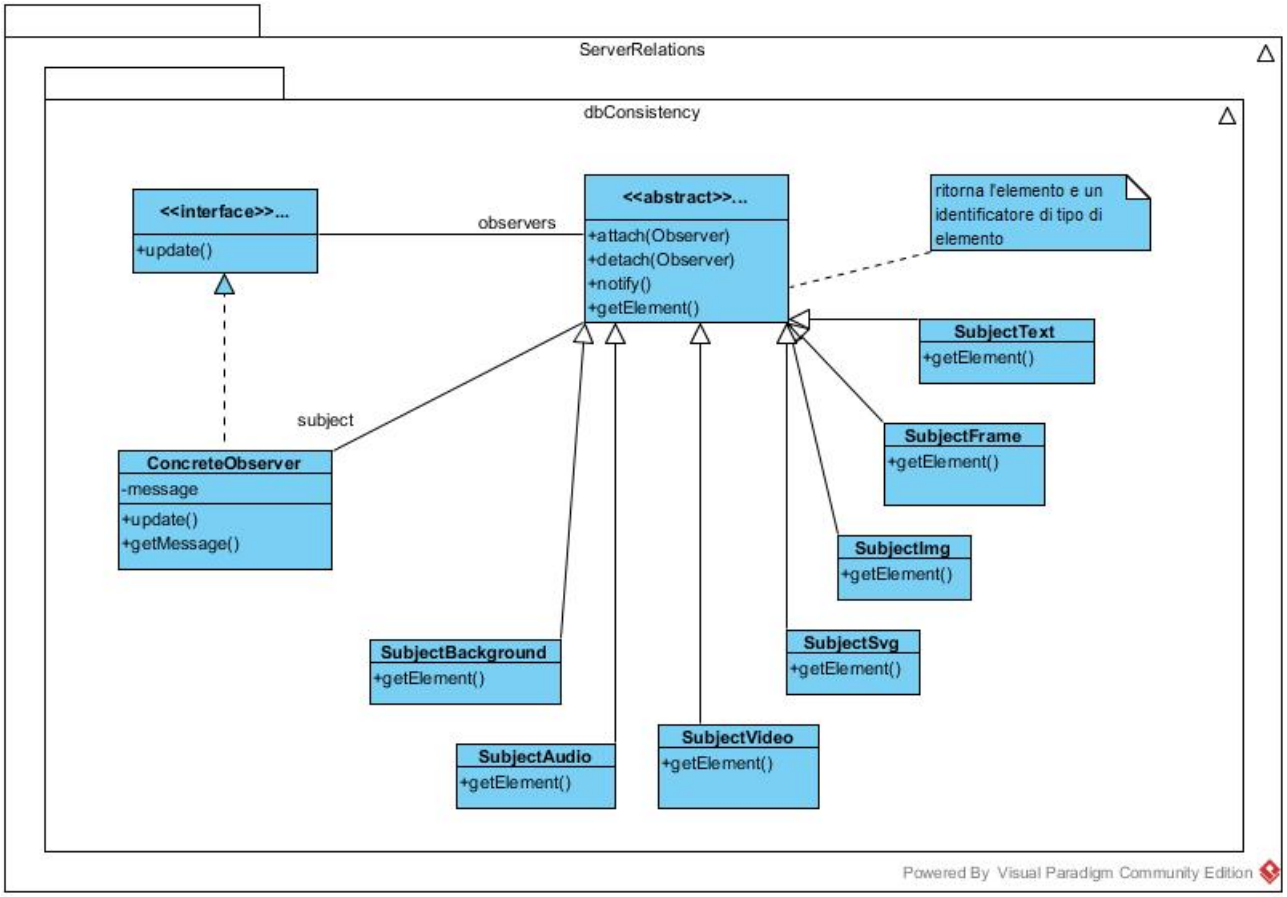

Fig 3: Observer

- **Descrizione**: L'Observer pattern è un design pattern utilizzato per tenere sotto controllo lo stato di diversi oggetti. Sostanzialmente il pattern si basa su uno o più oggetti, chiamati osservatori o listener, che vengono registrati per gestire un evento che potrebbe essere generato dall'oggetto osservato. Oltre all'observer esiste il concrete Observer che si differenzia dal primo perché implementa direttamente le azioni da compiere in risposta ad un messaggio; riepilogando il primo è una classe astratta, il secondo no.
- **Scopo dell'utilizzo**: Nel nostro caso il pattern viene utilizzato per garantire update automatici alle presentazioni dell'utente presenti consistentemente nel database MongoDB attraverso chiamate ai servizi offerti dalla web-api nodeApi.
- **Contesto d'utilizzo**: viene utilizzato in fase di modifica delle presentazioni.

3.3.1 Premi::Model::ServerRelations::dbConsinstency

Il package Premi::Model::ServerRelations::dbConsinstency contiene le classi che costituiscono il pattern observer, in particolare le classi ConcreteSubjects definiscono metodi getElement() che

ritornano l'elemento presente nel campo presentazione di Model::ServerRelations::Loader::Costruttore che riferiscono.

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 Presenter

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

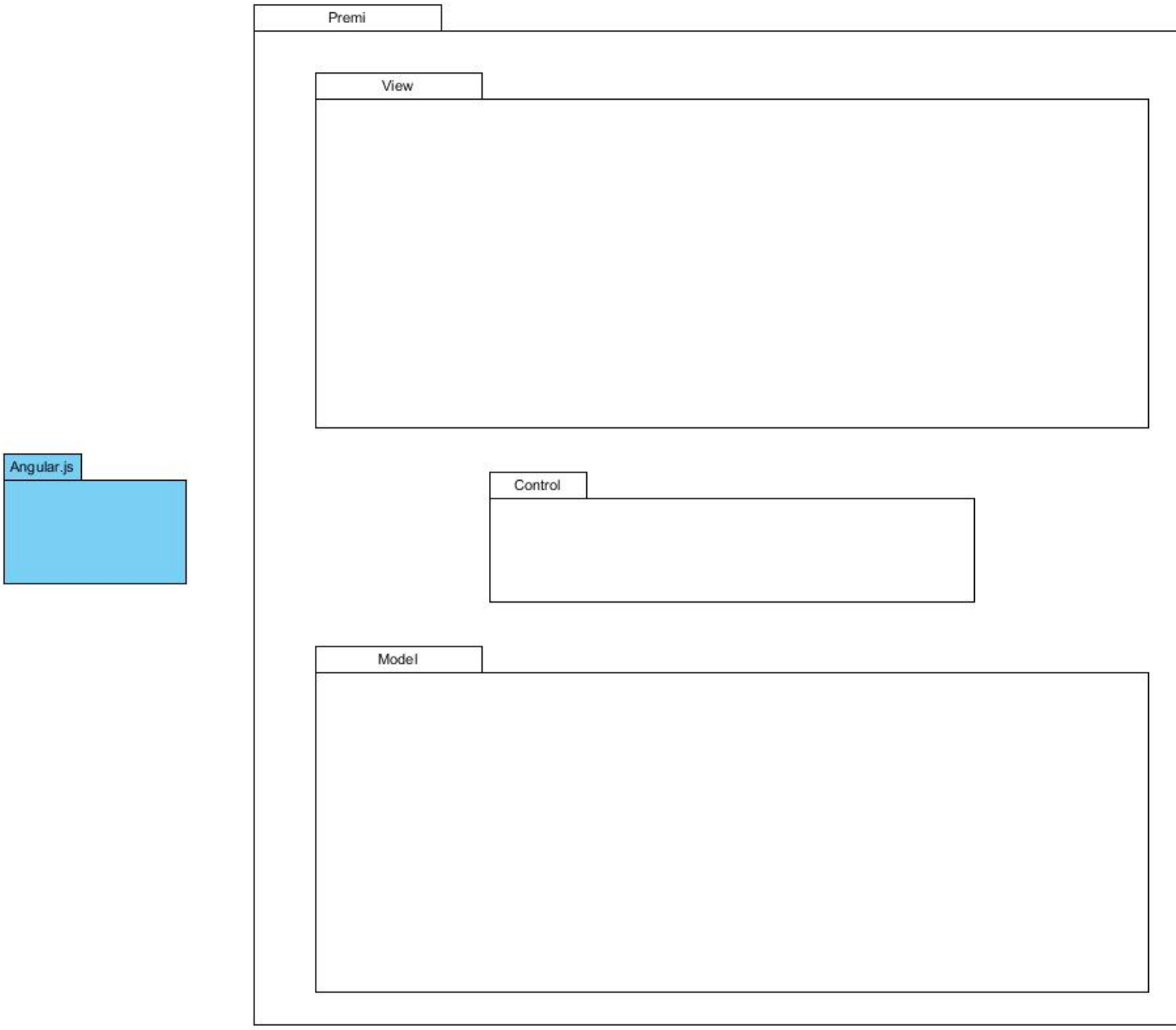

Fig 4: Architettura generale del sistema

5 Descrizione dei singoli componenti

Ogni componente appartiene al package Premi, quindi lo scope sarà Premi::<componente>.

5.1 Model

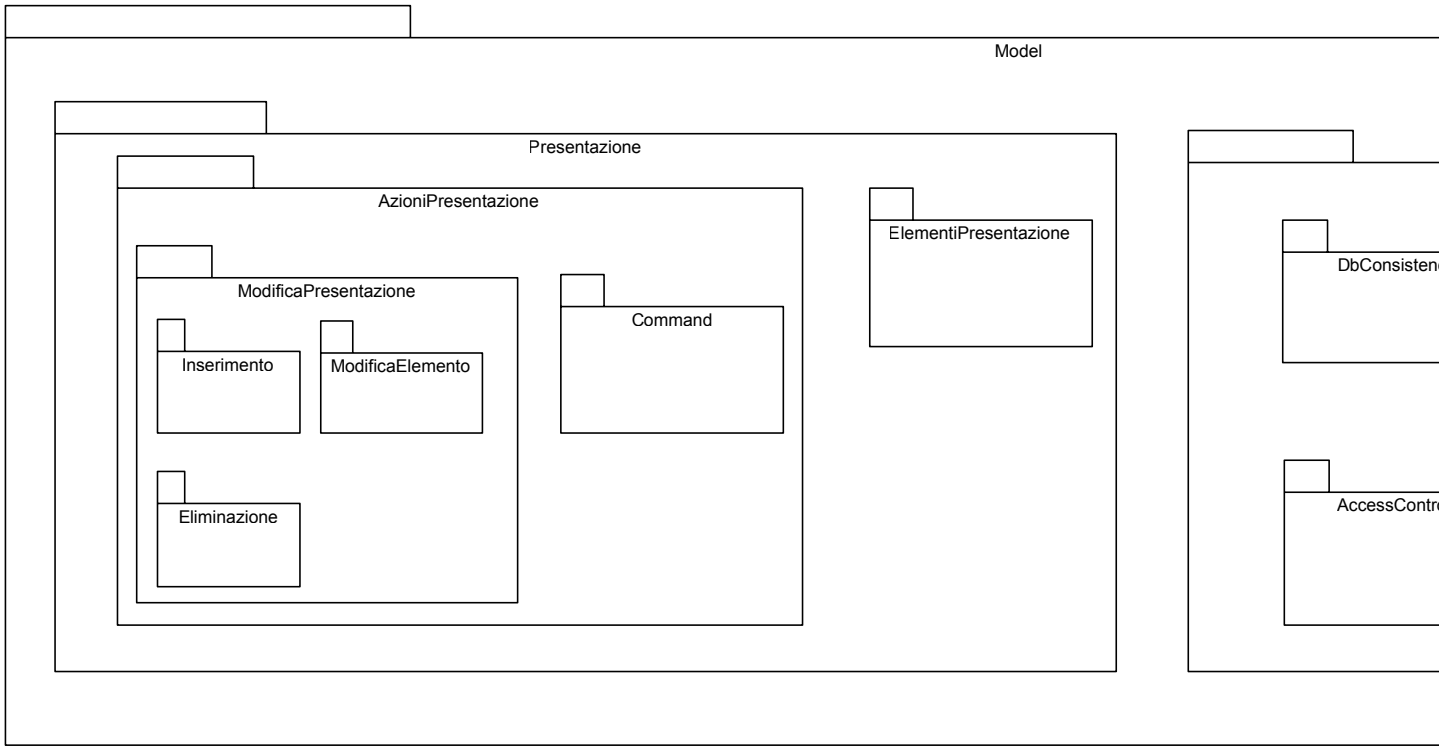

Fig 5: Model

Tipo, obiettivo e funzione del componente: è la parte Model dell'architettura MVC.
Relazioni d'uso di altre componenti: è in relazione con il package Presenter e con NodeAPI.
Package contenuti:

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

5.1.1 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 Presenter::EditPresenter da cui riceve i segnali e i parametri di inserimento e modifica degli elementi. Inoltre comunica con il package Model::ServerRelations, inviando a questi i segnali per la modifica in tempo reale dei dati presenti nel database.

5.1.2 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 Model::SlideShow::SlideShowActions: da cui riceve i segnali e i parametri di inserimento e modifica degli elementi. Inoltre comunica con il package Model::ServerRelations, inviando a questi i segnali per la modifica in tempo reale dei dati presenti nel database.

5.1.3 Model::SlideShow::SlideShowActions::InsertEditRemove

Tutti i componenti seguenti appartengono al package InsertEditRemove, quindi lo scope sarà Model::SlideShow::SlideShowActions::InsertEditRemove::<componente>.

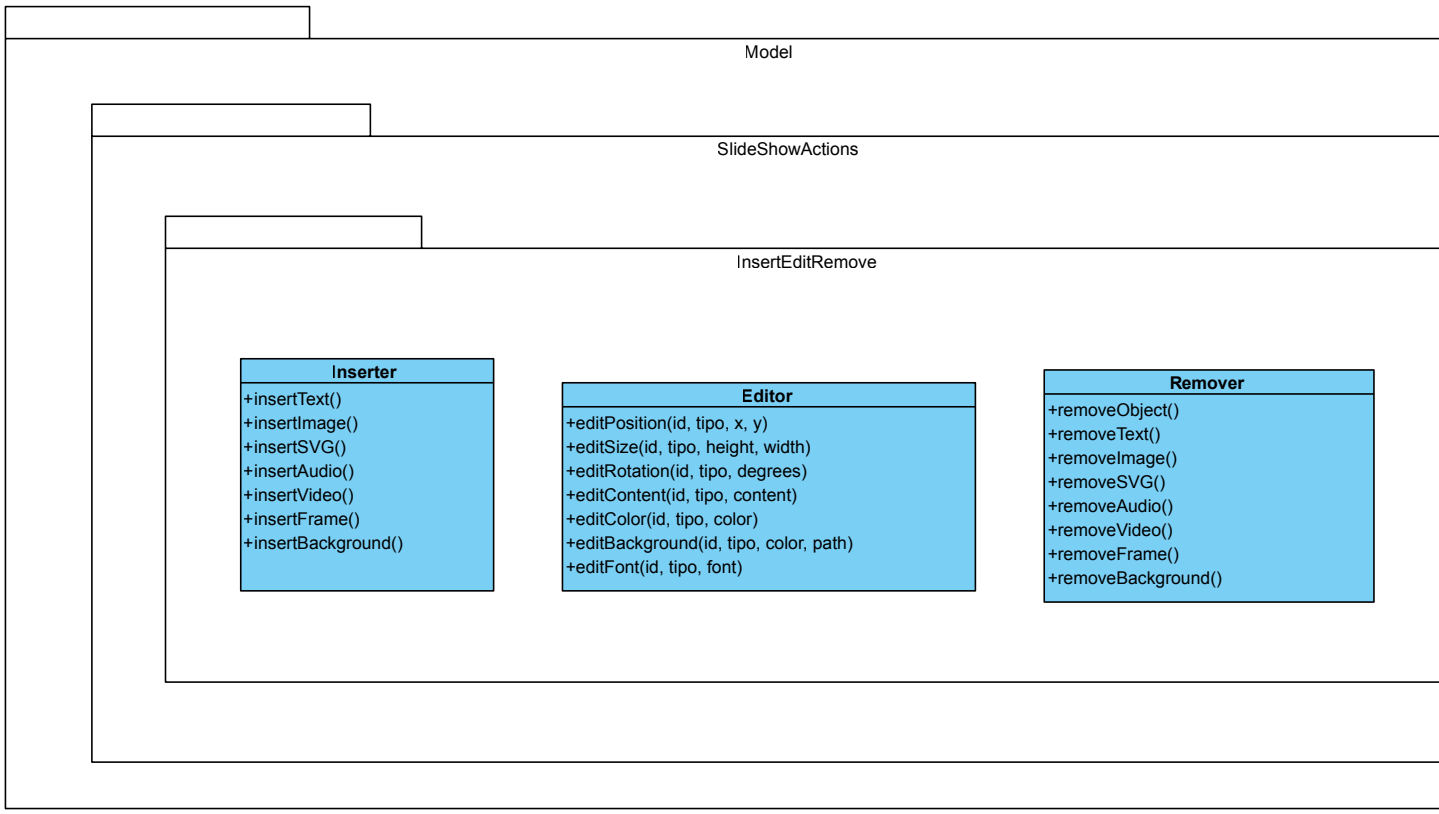

Fig 6: InsertEditRemove

Tipo, obiettivo e funzione del componente: all'interno di questo Package sono implementate le classi statiche destinate all'inserimento, alla rimozione e alla modifica degli elementi della presentazione.
Relazioni d'uso di altre componenti:il package è in relazione con Model::SlideShow::SlideShowActions:: che invoca i metodi delle classi del package.// Inoltre Model::SlideShow::SlideShowActions::InsertEditRemo si occupa di costruire gli oggetti presenti nelle classi del package Model::SlideShow::SlideShowElements.

5.1.3.1 Editor

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

- Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand -> invoca il metodo editSize() messo a disposizione da Editor;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand -> invoca il metodo editPosition() messo a disposizione da Editor;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand -> invoca il metodo editRotation() messo a disposizione da Editor;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand -> invoca il metodo editColor() messo a disposizione da Editor;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditFontCommand -> invoca il metodo editFont() messo a disposizione da Editor;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand -> invoca il metodo editBackground() messo a disposizione da Remover;
- Model::SlideShow::SlideShowElements::Text <- Editor invoca i metodi di set degli oggetti di classe Text;
- Model::SlideShow::SlideShowElements::Frame <- Editor invoca i metodi di set degli oggetti di classe Frame;
- Model::SlideShow::SlideShowElements::Image <- Editor invoca i metodi di set degli oggetti di classe Image;
- Model::SlideShow::SlideShowElements::SVG <- Editor invoca i metodi di set degli oggetti di classe SVG;
- Model::SlideShow::SlideShowElements::Audio <- Editor invoca i metodi di set degli oggetti di classe Audio;
- Model::SlideShow::SlideShowElements::Video <- Editor invoca i metodi di set degli oggetti di classe Video;
- Model::SlideShow::SlideShowElements::Background <- Editor invoca i metodi di set degli oggetti di classe Background;

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

5.1.3.2 Inserter

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

- Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand -> invoca il metodo insertText() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand -> invoca il metodo insertFrame() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand -> invoca il metodo insertImage() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand -> invoca il metodo insertSVG() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand -> invoca il metodo insertAudio() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand -> invoca il metodo insertVideo() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand -> invoca il metodo insertBackground() messo a disposizione da Inserter;
- Model::SlideShow::SlideShowElements::Text <- Inserter costruisce gli oggetti di classe Text;
- Model::SlideShow::SlideShowElements::Frame <- Inserter costruisce gli oggetti di classe Frame;
- Model::SlideShow::SlideShowElements::Image <- Inserter costruisce gli oggetti di classe Image;
- Model::SlideShow::SlideShowElements::SVG <- Inserter costruisce gli oggetti di classe SVG;
- Model::SlideShow::SlideShowElements::Audio <- Inserter costruisce gli oggetti di classe Audio;
- Model::SlideShow::SlideShowElements::Video <- Inserter costruisce gli oggetti di classe Video;
- Model::SlideShow::SlideShowElements::Background <- Inserter costruisce gli oggetti di classe Background;
- Model::ServerRelations::Loader::Caricatore <- Inserter inserisce gli oggetti json nel campo dati contenitore presentazione.

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

5.1.3.3 Remover

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

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

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

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

5.1.4 Model::SlideShow::SlideShowActions::Command

Tutti i componenti seguenti appartengono al package Command, quindi lo scope sarà Model::SlideShow::SlideShowActions::Command::<componente>.

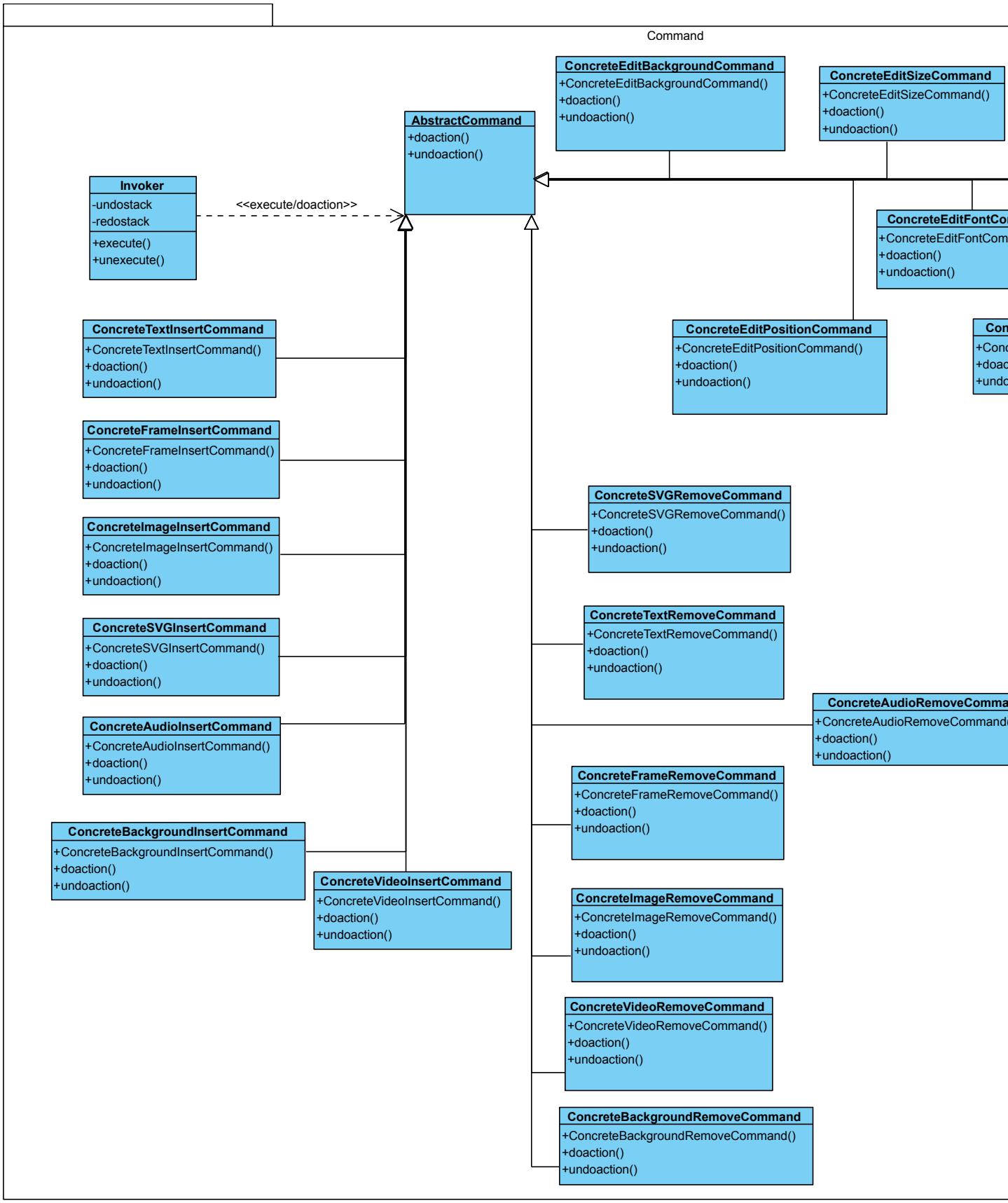

Fig 7: Command Package

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

5.1.4.1 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:

- Presenter::MobileEdit->crea un oggetto di una sottoclasse di Model::SlideShow::SlideShowActions::C 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";
- Presenter::DesktopEdit->si comporta in modo analogo a MobileEdit;
- 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 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.

5.1.4.2 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:

- 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:

- Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteFrameInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteImageInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemoveCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditFontCommand;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditContentCommand.

5.1.4.3 ConcreteTextInsertCommand

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:

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

Classi ereditate:

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

5.1.4.4 ConcreteFrameInsertCommand

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:

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

Classi ereditate:

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

5.1.4.5 ConcreteImageInsertCommand

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:

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

Classi ereditate:

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

5.1.4.6 ConcreteSVGInsertCommand

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:

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

Classi ereditate:

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

5.1.4.7 ConcreteAudioInsertCommand

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:

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

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

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

5.1.4.8 ConcreteVideoInsertCommand

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:

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

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

Classi ereditate:

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

5.1.4.9 ConcreteBackgroundInsertCommand

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:

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

Classi ereditate:

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

5.1.4.10 ConcreteTextRemoveCommand

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:

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

Classi ereditate:

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

5.1.4.11 ConcreteFrameRemoveCommand

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:

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

Classi ereditate:

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

5.1.4.12 ConcreteImageRemoveCommand

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:

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

Classi ereditate:

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

5.1.4.13 ConcreteSVGRemoveCommand

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:

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

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

Classi ereditate:

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

5.1.4.14 ConcreteAudioRemoveCommand

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:

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

Classi ereditate:

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

5.1.4.15 ConcreteVideoRemoveCommand

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:

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

Classi ereditate:

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

5.1.4.16 ConcreteBackgroundRemoveCommand

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:

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

Classi ereditate:

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

5.1.4.17 ConcreteEditSizeCommand

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

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

Classi ereditate:

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

5.1.4.18 ConcreteEditPositionCommand

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

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

- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- invoca il metodo editPosition() della classe statica per la modifica dei campi dati relativi alla posizione dell'oggetto nella presentazione.

Classi ereditate:

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

5.1.4.19 ConcreteEditColorCommand

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

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

Classi ereditate:

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

5.1.4.20 ConcreteEditBackgroundCommand

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:

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

Classi ereditate:

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

5.1.4.21 ConcreteEditRotationCommand

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:

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

Classi ereditate:

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

5.1.4.22 ConcreteEditFontCommand

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:

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

Classi ereditate:

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

5.1.5 Model::SlideShow::SlideShowElements

Tutti i componenti seguenti appartengono al package SlideShowElements, quindi lo scope sarà Model::SlideShow::SlideShowElements::<componente>.

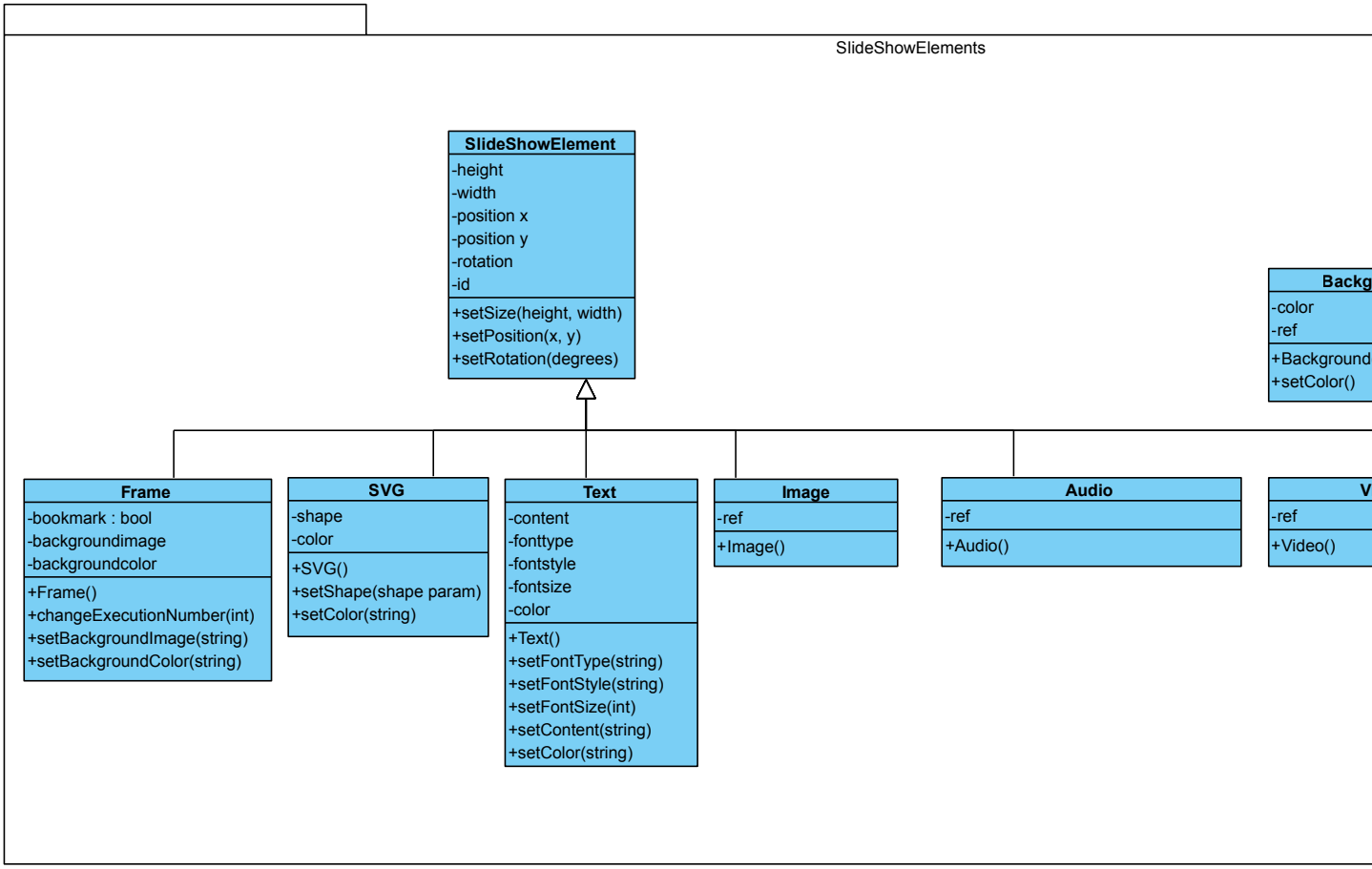

Fig 8: SlideShowElements

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:.Model::SlideShow::SlideShowElements è in comunicazione con

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

5.1.5.1 SlideShowElement

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

- Model::SlideShow::SlideShowActions::Insert::Inserter-> invoca il costruttore delle sottoclassi di SlideShowElement e li inserisce nei campi dati contenitori all'interno di Model::SlideShow::SlideShow;
- Model::SlideShow::SlideShowActions::EditElements:Editor -> gli oggetti delle sue sottoclassi richiamano le funzioni delle sottoclassi di SlideShowElement che gestiscono l'impostazione dei campi dati;
- 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: Model::SlideShow::SlideShowActions::Insert:: instanzia oggetti di sottoclassi di SlideShowElement e li inserisce nel campo dati contenitore presentazione all'interno di Model::ServerRelations::Model:Costruttore
Sottoclassi:

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

5.1.5.2 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:

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

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

- Model::SlideShow::SlideShowElement.

5.1.5.3 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:

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

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

- Model::SlideShow::SlideShowElement.

5.1.5.4 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:

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

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

- Model::SlideShow::SlideShowElement.

5.1.5.5 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:

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

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

- Model::SlideShow::SlideShowElement.

5.1.5.6 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:

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

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

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

5.1.5.7 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:

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

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

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

5.1.6 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:

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

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe Background vengono istanziati da Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter e da questi inseriti nel campo dati contenitore presentazione all'interno di Model::ServerRelations::Loader::C
Classi ereditate:

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

5.1.7 Model::ServerRelations

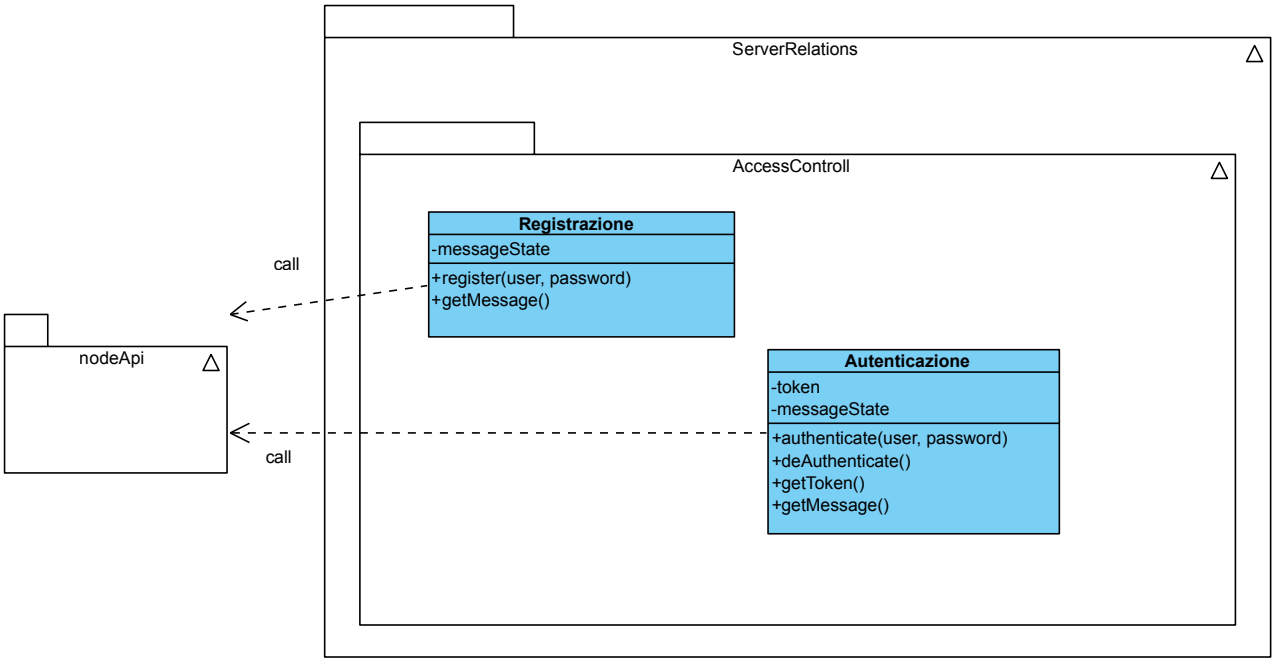

Fig 9: ServerRelations

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.

5.1.8 Model::ServerRelations::Loader

Tutti i componenti seguenti appartengono al package Loader, quindi lo scope sarà Model::ServerRelations::L

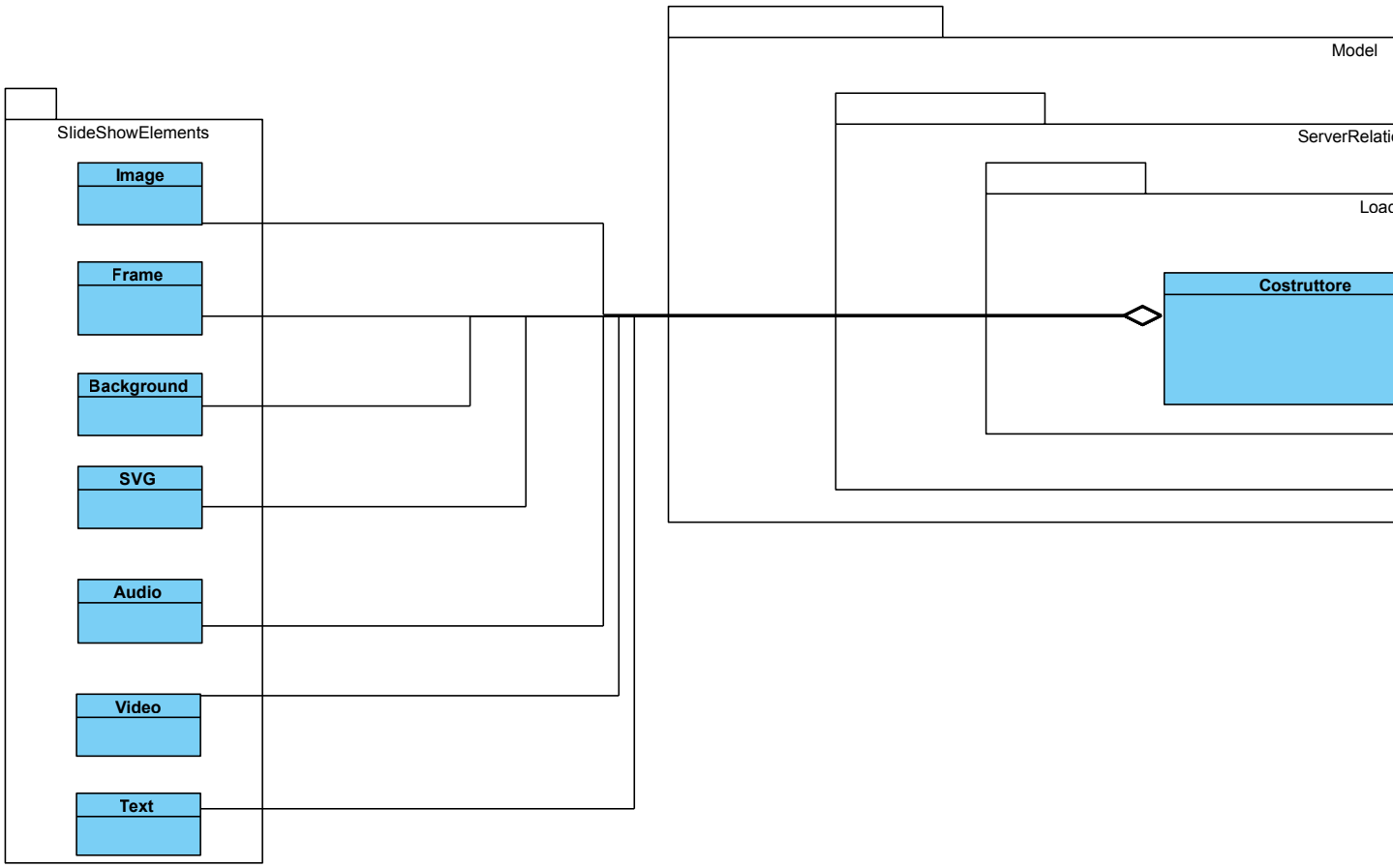

Fig 10: ServerRelationsLoader

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.

5.1.8.1 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.

5.1.9 Model::ServerRelations::AccessControl

Tutti i componenti seguenti appartengono al package AccessControl, quindi lo scope sarà Model::ServerRelations::AccessControl::<componente>. **Tipo, obiettivo e funzione del com-**

ponente: 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.

5.1.9.1 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.
- Presenter::Pagine::IndexPresenter -> invoca i metodi di Autenticazione per permettere all'utente di effettuare il login.

5.1.9.2 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.
- Presenter::Pagine::IndexPresenter -> invoca i metodi di Registrazione per permettere all'utente di registrarsi al servizio.

5.1.10 Model::ServerRelations::DbConsistency

Tutti i componenti seguenti appartengono al package DbConsistency, quindi lo scope sarà Model::ServerRelations::DbConsistency::<componente>.

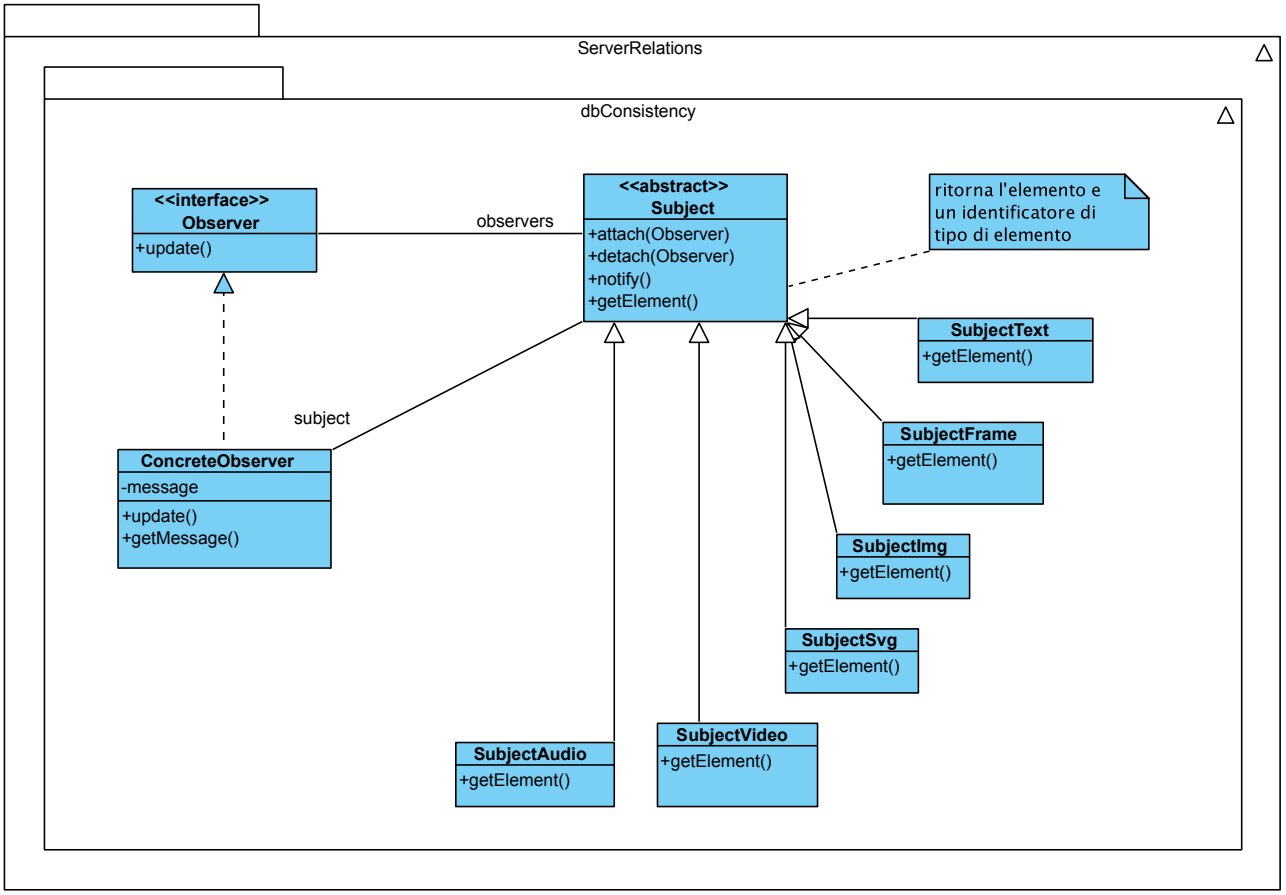

Fig 11: DbConsistency

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.

5.1.10.1 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().

5.1.10.2 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.

5.1.10.3 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.

5.1.10.4 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 Model::SlideShow::SlideShowElements::Audio di cui detiene un riferimento.

5.1.10.5 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 Model::SlideShow::SlideShowElements::Video di cui detiene un riferimento.

5.1.10.6 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 Model::SlideShow::SlideShowElements::Text di cui detiene un riferimento.

5.1.10.7 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 Model::SlideShow::SlideShowElements::Frame di cui detiene un riferimento.

5.1.10.8 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 Model::SlideShow::SlideShowElements::Image di cui detiene un riferimento.

5.1.10.9 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 Model::SlideShow::SlideShowElements::SVG di cui detiene un riferimento.

5.1.10.10 SubjectBackground

Tipo, obiettivo e funzione del componente: Classe, fornisce un'implementazione di Subject permettendo di applicare il pattern Observer.
Relazioni d'uso di altre componenti:

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

5.1.11 Model::ApacheRelations

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

- Il package comunica con la view ricevendo chiamate da View::Pages::Profile.
- I componenti del package ApacheRelations hanno relazioni di dipendenza nei confronti del package nodeApi del quale utilizzano i servizi esposti dall'interfaccia; c'è dipenda tra il package ApacheRelations ed il package ServerRelations.

5.1.12 Model::ApacheRelations::ApacheServerManager

Il componente seguente appartiene al package ApacheServerManager, quindi lo scope sarà Model::ApacheRelations::ApacheServerManager::<componente>. **Tipo, obiettivo e funzione del componente**: Compito di questo package è di gestire l'interazione con il server Apache per le operazioni di modifica dei files.
Relazioni d'uso di altre componenti:

- Il package comunica con la view ricevendo chiamate da View::Pages::Profile.

5.1.12.1 FileManager

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

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

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

5.1.13 Model::ApacheRelations::Manifest

Il seguente componente appartiene al package Manifest, quindi lo scope sarà Model::ApacheRelations::Manif
Tipo, obiettivo e funzione del componente: Questo package ha lo scopo di rendere disponibili le presentazioni in locale tramite chiamate ai servizi di nodeApi e ai metodi definiti in Model::ServerRelation::Loader.
Relazioni d'uso di altre componenti:

- definisce il metodo GestoreManifest(); relazione di dipendenza

5.1.13.1 GestoreManifest

Tipo, obiettivo e funzione del componente: classe, fornisce i servizi raccolti nel package;
Relazioni d'uso di altre componenti:

- definisce il metodo insertElement(), addPage(), update()

5.2 Presenter

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

5.2.1 Presenter::EditPresenter

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

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

- Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundRemoveCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditFontCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::ConcreteEditContentCommand <- EditPresenter costruisce un comando e lo dà in pasto a Model:Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker <- EditPresenter 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;
- Model::ApacheManager::FileManager <- EditPresenter invoca i metodi uploadFile() di FileManager quando viene inserito nella presentazione un file non ancora presente nel server;
- Model::Manifest::ManifestManager <- la classe della view invoca il metodo save() presente in ExecutionPresenter che a sua volta invoca il metodo update() di ManifestManager che aggiorna il file manifest con tutti gli elementi della presentazione e lo ricarica.
- View::Pages::DesktopEdit e View::Pages::MobileEdit -> costruiscono EditPresenter, ne invocano i metodi passando i parametri degli oggetti modificati;
- View::Pages::DesktopEdit e View::Pages::MobileEdit <- quando il logout ha successo EditPresenter comunica alla view di effettuare una redirect verso Index;
- Model::ServerRelations::Loader::Autenticazione <- Quando la view invia una richiesta di logout EditPresenter invoca il metodo di Autenticazione deAuthenticate(), che termina la sessione.

Interfacce con e relazioni d'uso e da altre componenti: La pagina DesktopEdit o la pagina MobileEdit invia a EditPresenter comunica l'avvenuta modifica o la rimozione di un elemento della presentazione o l'inserimento di un nuovo elemento invocando i metodi corrispondenti di EditPresenter. EditPresenter istanzia un oggetto di una sottoclasse di Model::SlideShow::SlideShowActions::Command::AbstractCommand e lo dà in pasto a Model::Invoker.

Eventualmente EditPresenter, dopo che la View ha invocato il metodo undo() di EditPresenter, 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 EditPresenter, che, tramite invocazione dei metodi di Model::ServerRelations::Loader::Costruttore, caricherà dal database.

5.2.2 Presenter::HomePresenter

Tipo, obiettivo e funzione del componente: Lo scopo di questa classe è di gestire i segnali e le chiamate provenienti dalla pagina View::Pages::Home.
Relazioni d'uso di altre componenti:

- View::Pages::Home -> costruisce HomePresenter, ne invoca i metodi passando i parametri dell'utente;
- Model::ServerRelations::Loader::Costruttore <- HomePresenter invoca un metodo di Costruttore che restituisce l'elenco dei titoli delle presentazioni dell'utente(??????);
- Model::ServerRelations::Loader::Autenticazione <- Quando la view invia una richiesta di logout, HomePresenter invoca il metodo deAuthenticate() fornito da Autenticazione, che termina la sessione;
- Model::Manifest::ManifestManager <- la classe della view invoca il metodo save() presente in HomePresenter passando per parametro un array di id di presentazioni che l'utente intende scaricare in locale, a sua volta HomePresenter invoca il metodo update() di ManifestManager che controlla se esiste già un file manifest dopodiché lo aggiorna con tutti i riferimenti alle pagine da scaricare e lo ricarica.

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

5.2.3 Presenter::ExecutionPresenter

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

- View::Pages::Execution -> costruisce ExecutionPresenter, ne invoca i metodi passando i parametri della presentazione da caricare;
- Model::ServerRelations::Loader::Costruttore <- ExecutionPresenter passa i parametri di caricamento al Loader che carica la presentazione attraverso nodeAPI e lo traduce in html ritornando il codice a ExecutionPresenter;

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

5.2.4 Presenter::IndexPresenter

Tipo, obiettivo e funzione del componente: Lo scopo di questa classe è di gestire i segnali e le chiamate della pagina View::Pages::Index.
Relazioni d'uso di altre componenti:

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

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

5.2.5 Presenter::ProfilePresenter

Tipo, obiettivo e funzione del componente: Lo scopo di questa classe è di gestire i segnali e le chiamate della pagina View::Pages::Presenter.
Relazioni d'uso di altre componenti:

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

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

5.3 View

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

5.3.1 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.

5.3.2 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.

5.3.3 View::Pages::Manifest

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

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

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

5.3.4 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.

5.3.5 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.

5.3.6 View::Pages::Edit

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

5.3.7 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: View::Pages::Edit.

5.3.8 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: 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.

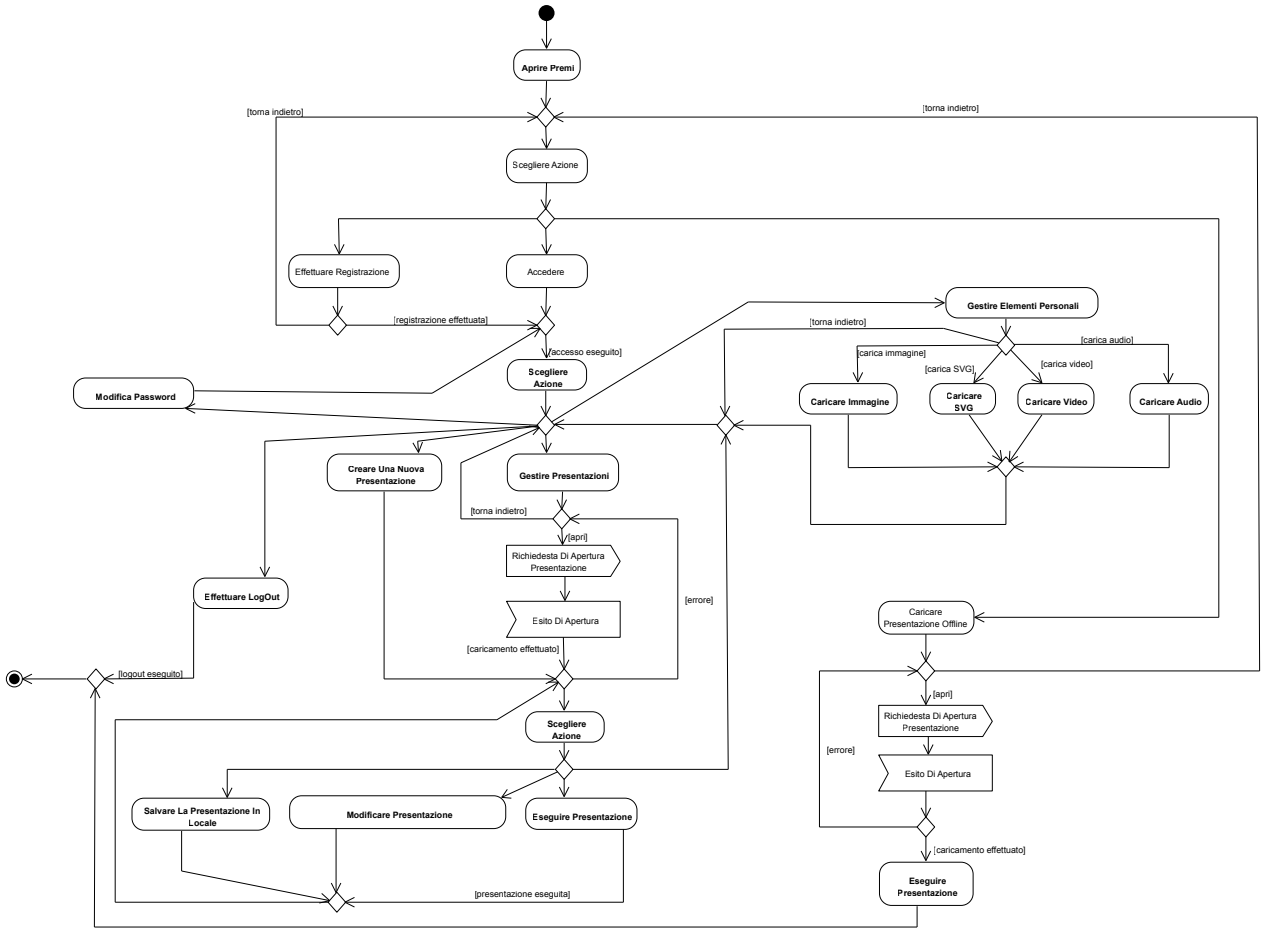

Fig 12: Attività Principali

6.1.1 Gestione presentazioni

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

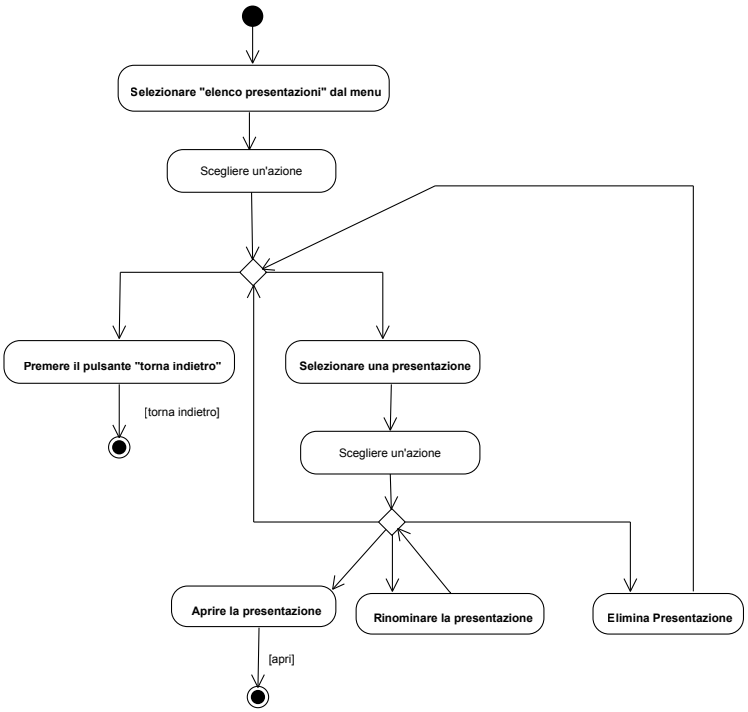

Fig 13: Gestione Presentazioni

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.

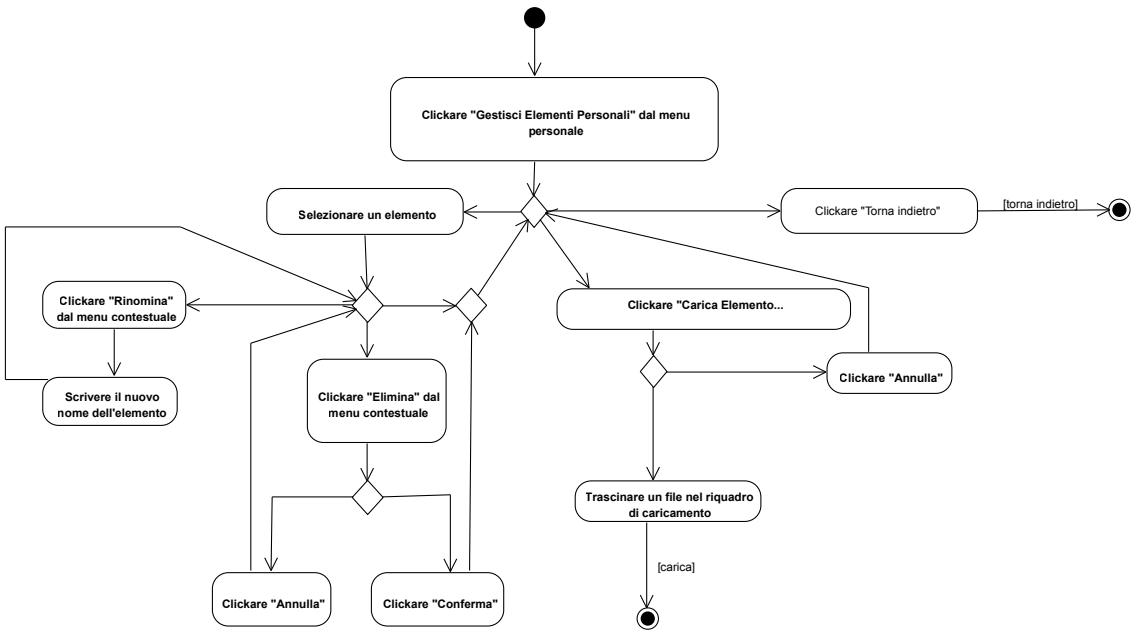

Fig 14: Caricare File

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.

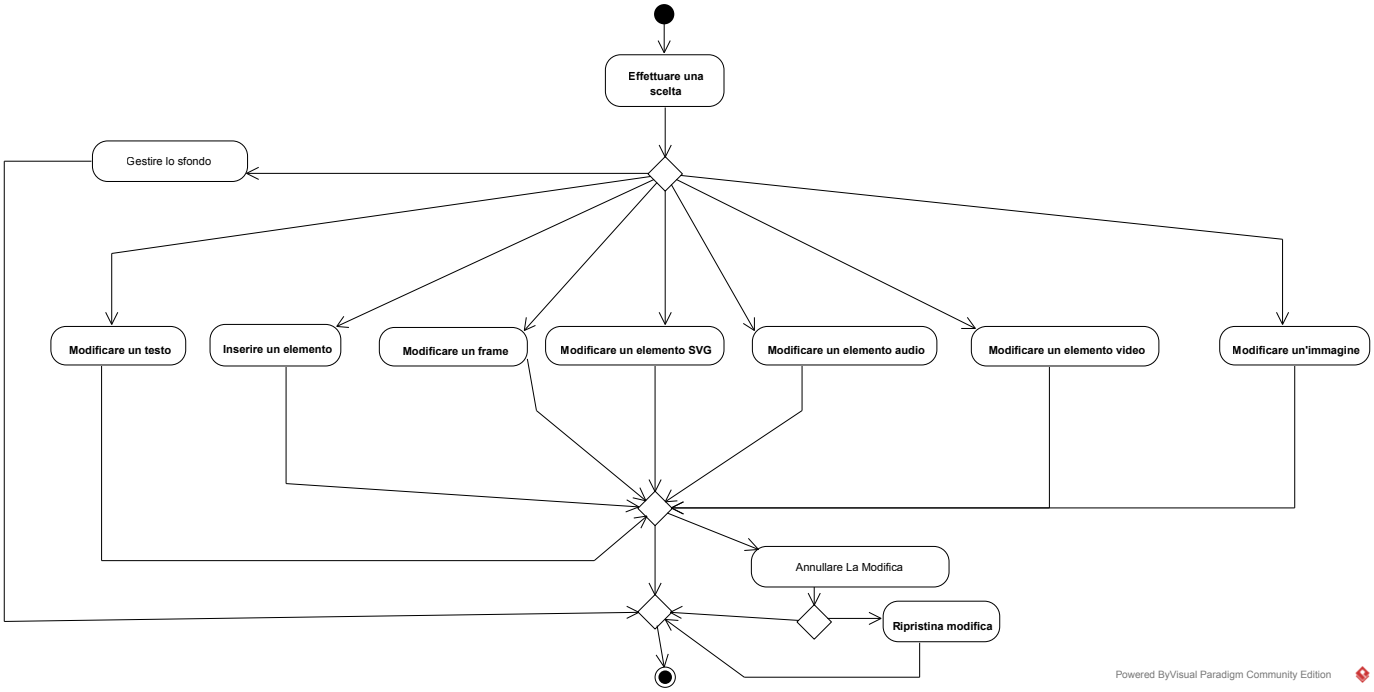

Fig 15: Modificare Presentazione da Desktop

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.

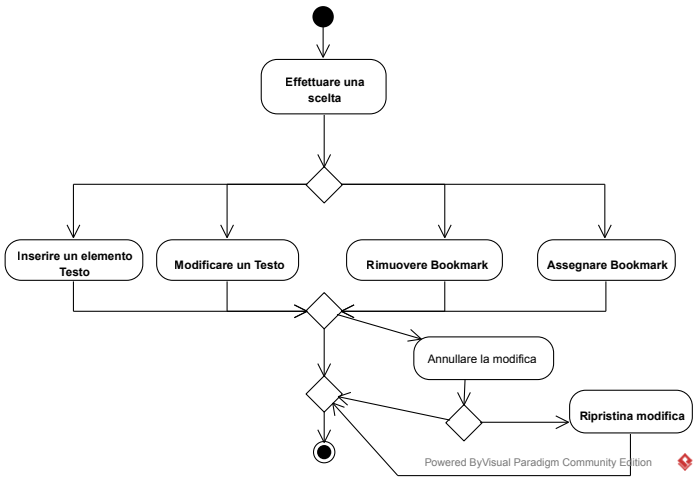

Fig 16: Modificare Presentazione da Mobile

6.1.5 Gestire Sfondo

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

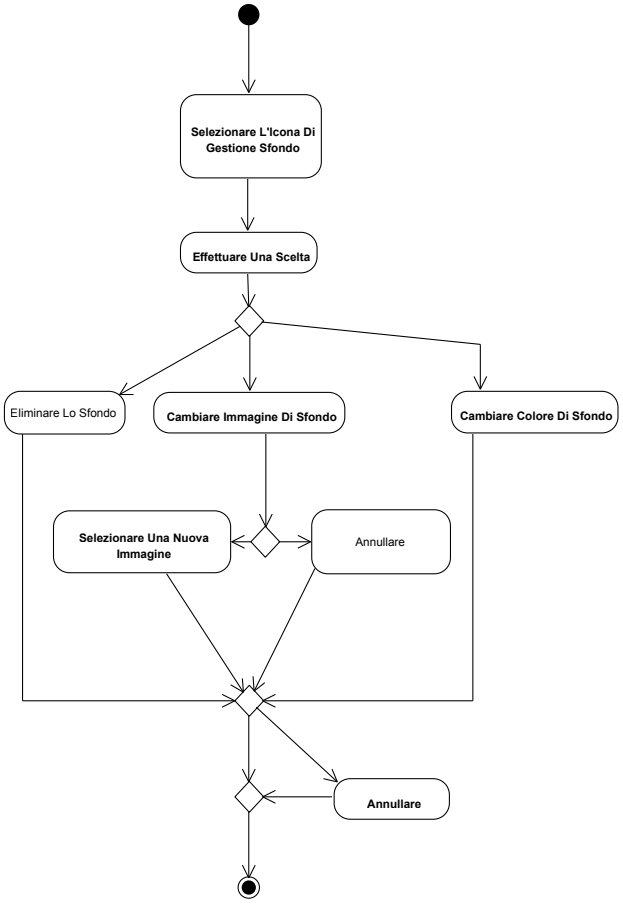

Fig 17: Gestire Sfondo

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.

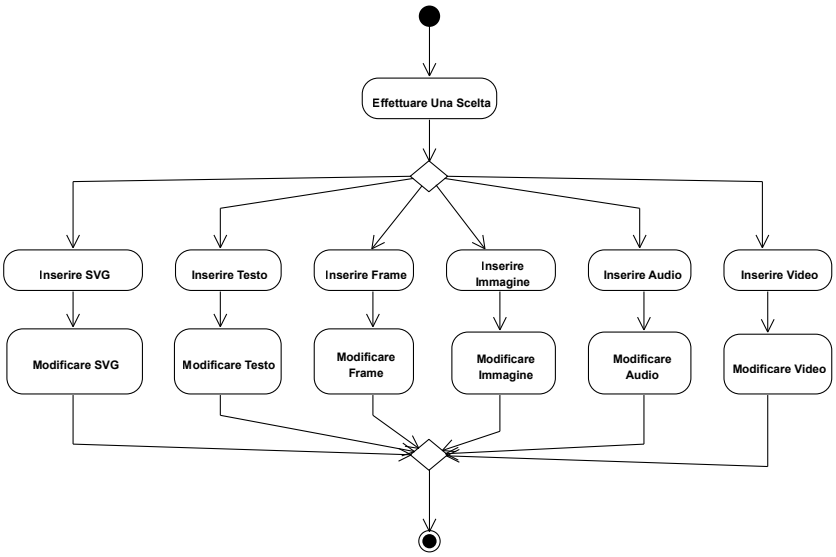

Fig 18: Inserire Elemento

6.1.7 Modificare Elemento

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

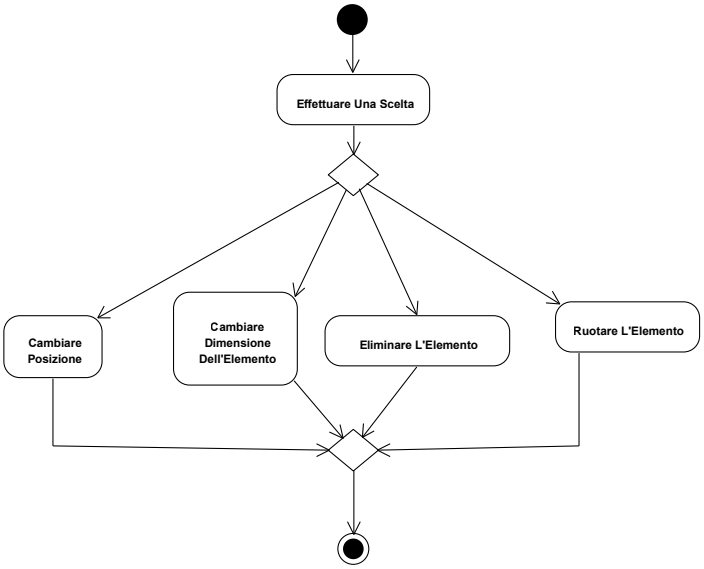

Fig 19: Modificare Elemento

6.1.8 Modificare Frame

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

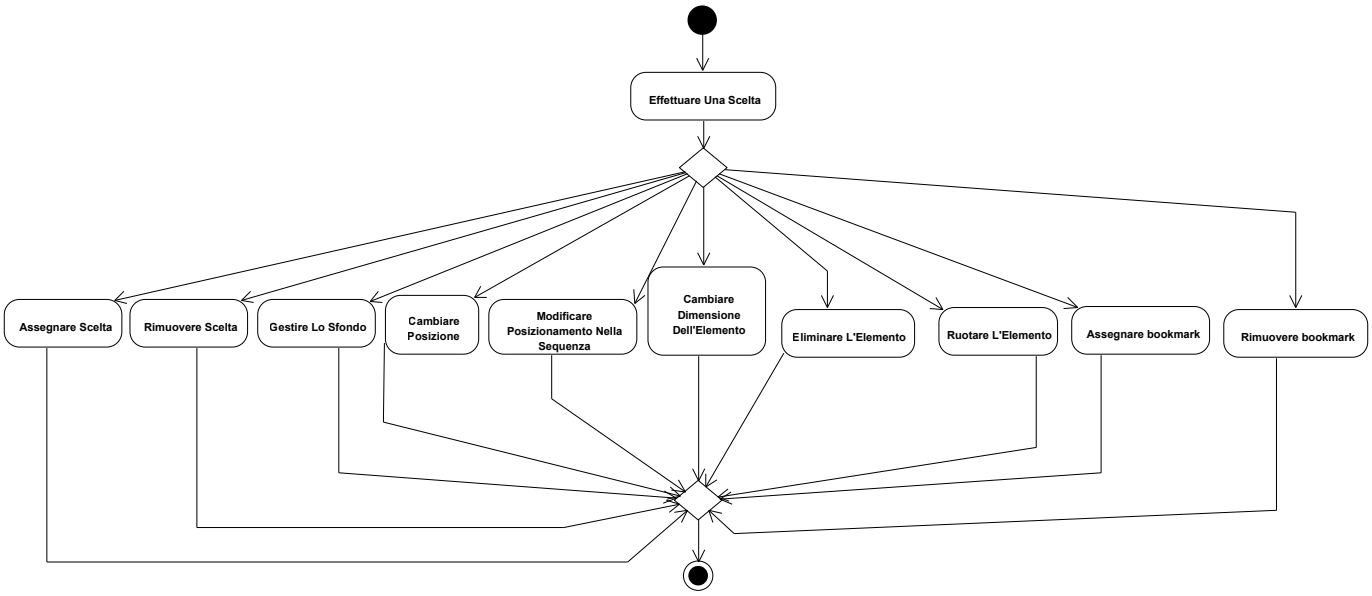

Fig 20: Modificare Frame

6.1.9 Modificare SVG

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

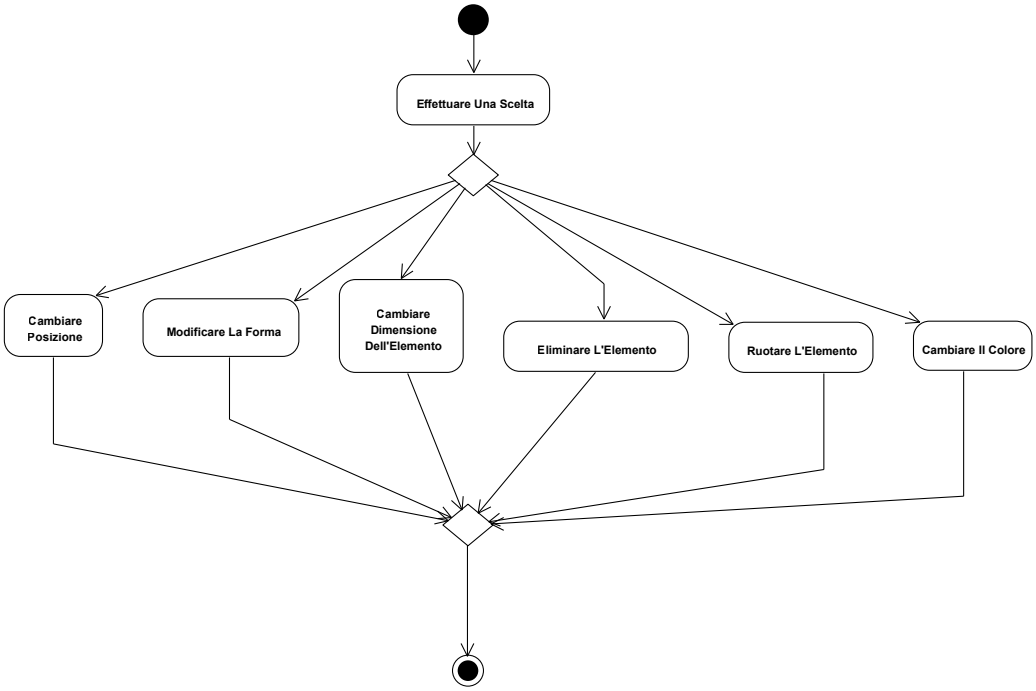

Fig 21: Modificare SVG

6.1.10 Modificare Testo

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

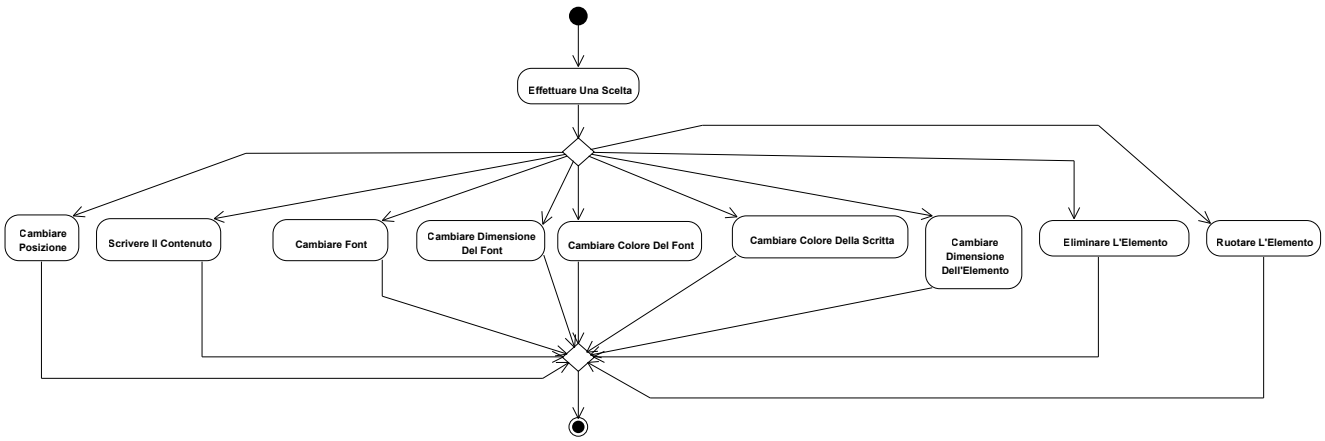

Fig 22: Modificare Testo

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.
Poiché tutti gli strumenti da utilizzare nello sviluppo sono gratuiti, il bisogno di risorse non si dimostra essere particolarmente problematico.
Si è deciso di utilizzare HTML5, CSS3 e Javascript (e le sue librerie) per lo sviluppo della parte web.
Per la parte di database si è scelto l'utilizzo di MEAN e delle librerie Express.js e Node.js per una migliore interazione con MongoDB.
Per la parte di esecuzione delle presentazioni è stato scelto Impress.js, framework che permette l'esecuzione in maniera non lineare come richiesto.
Per la parte di modifica delle presentazioni verrà utilizzato il framework Angular.js per lo spostamento in tempo reale degli elementi delle presentazioni.

8 Tracciamento dei Componenti coi Requisiti