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. HTML5 è uno standard da settembre 2014 e permette una più semplice integrazione di contenuti multimediali.

- **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 JavaScript 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.

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; questo rende l'integrazione di dati più semplice e facile in alcuni tipi d'applicazioni.

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; nel nostro sistema è utilizzato in particolar modo per la gestione dei path da cui sono offerti i servizi per l'interfacciamento con il database Mongo.

2.4.3 AngularJS

AngularJS, è un framework per applicazioni web, open-source, manutenuto 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-whatever lato-client.
Il framework AngularJS come prima cosa legge la pagina HTML, che ha al suo interno degli attributi Tag personalizzati; Angular interpreta questi attributi come direttive per legare parti di input o di output della pagina ad un modello che è rappresentato da variabili Javascript standard. Il valore di queste variabili Javascript può essere impostato manualmente all'interno del codice, oppure ricavato da risorse JSON statiche o dinamiche.

2.4.4 Node.js

Node.js è un ambiente di esecuzione open source e cross-platform per applicazioni lato server; le applicazioni Node.js sono scritte in linguaggio Javascript. Node.js fornisce un'architettura scalabile orientata agli eventi grazie alla sua natura asincrona. Node.js usa il motore Javascript V8 di Google per eseguire codice, ed una larga percentuale dei moduli base è scritta in Javascript.

2.5 Impress.js

Impress.js è un framework open source che permette di visualizzare i Tag 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 MVC

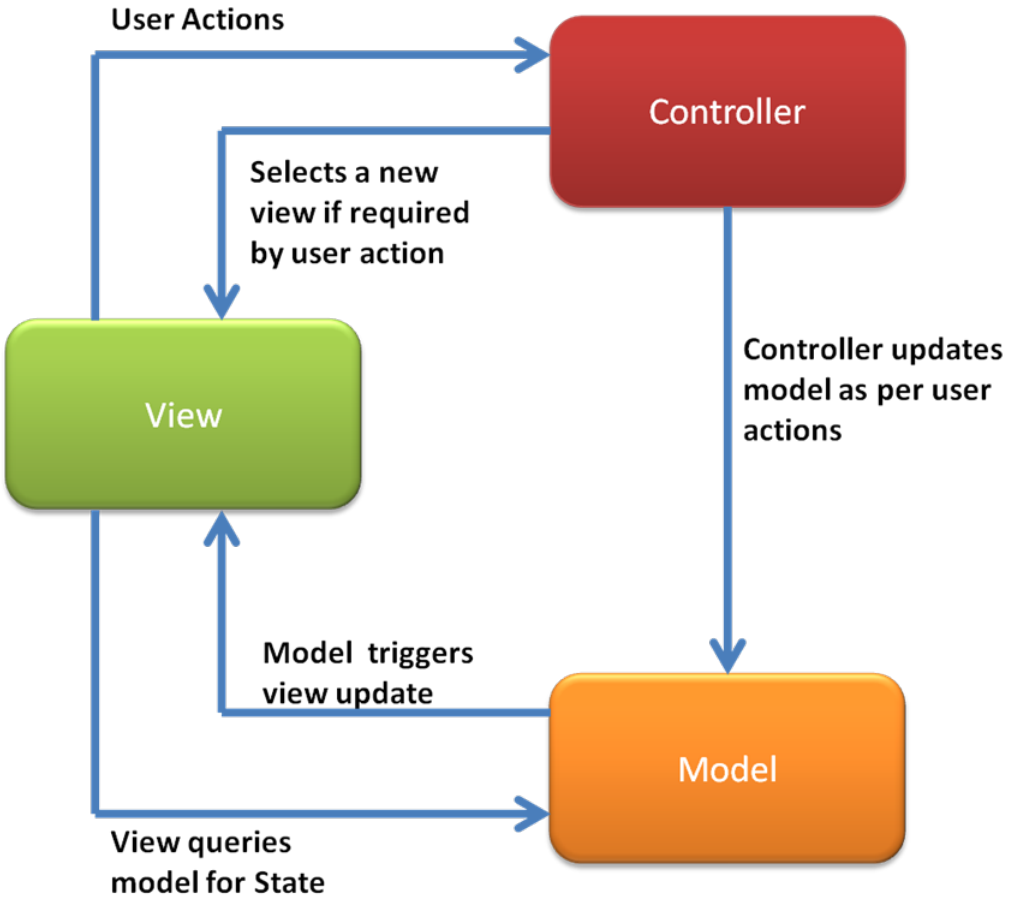

Fig 1: Model View Controller

- **Scopo dell'utilizzo**: è stato scelto il pattern MVC per separare la logica dell'applicazione dalla rappresentazione grafica;
- **Contesto d'utilizzo**: Il pattern MVC viene utilizzato per l'architettura generale dell'applicazione. Ogni modifica effettuata dall'utente sulla View viene inviata al Model tramite il Controller e viceversa.

3.2 Command

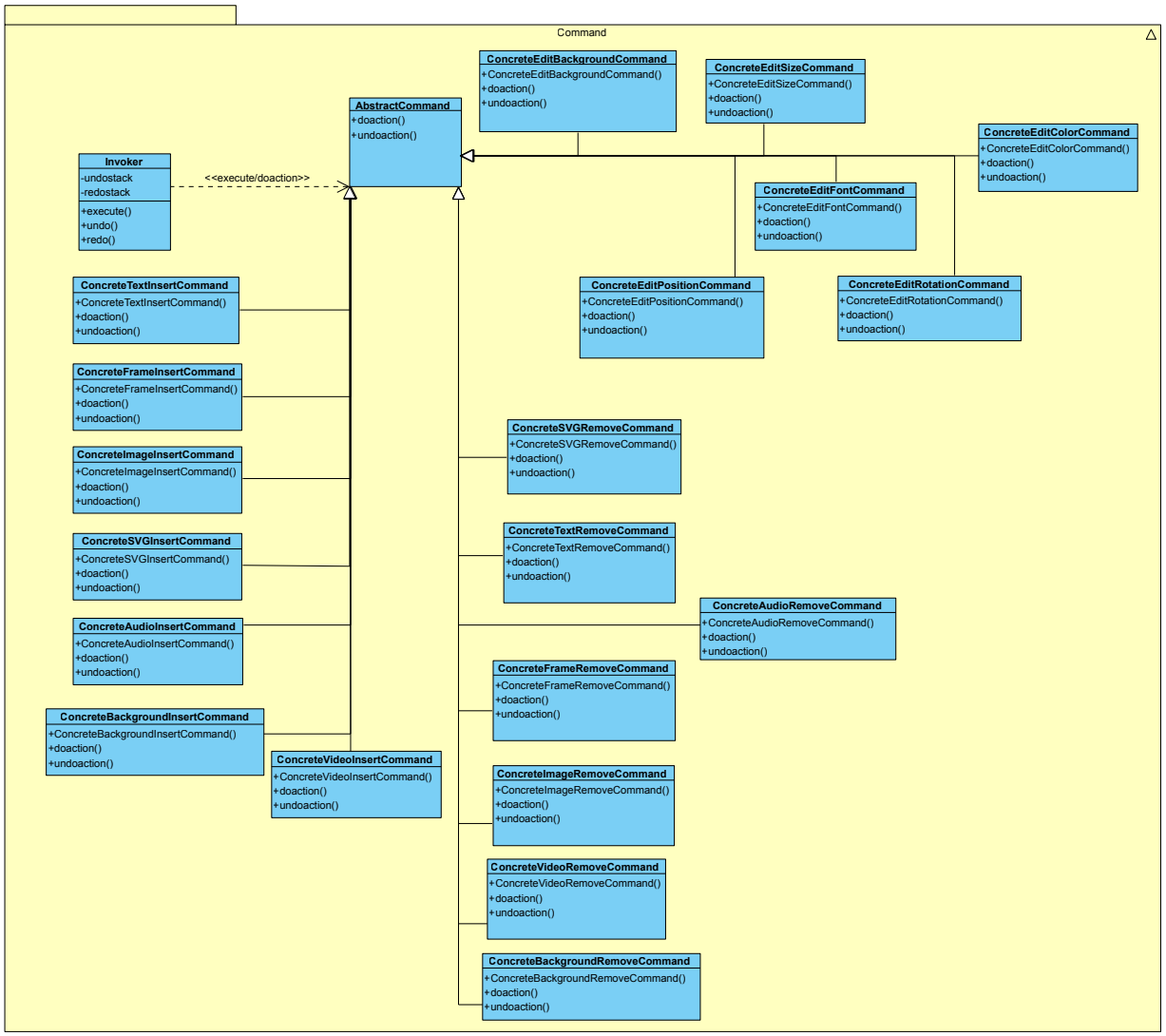

Fig 2: Diagramma delle classi del package Command

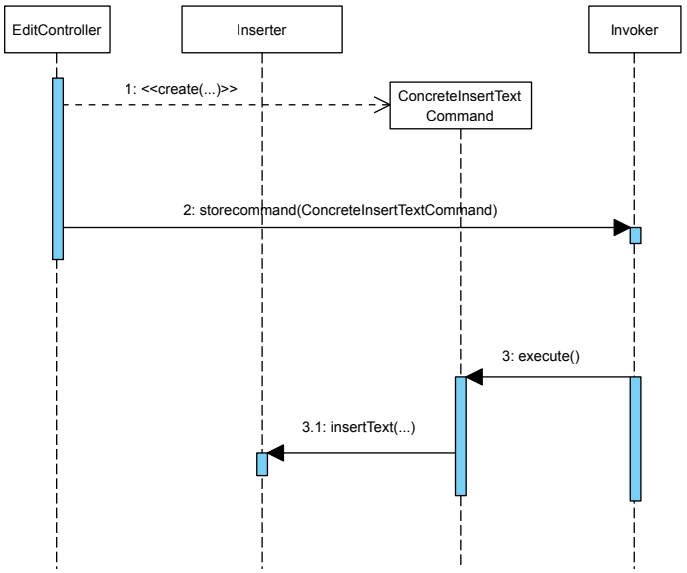

Fig 3: diagramma di sequenza del Pattern Command

- **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::Controller::Presentazione::Edit, che invoca il costruttore delle sottoclassi di Premi::Model::SlideShow::SlideShowActions::Command::AbstractCommand 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::Controller::Presentazione::Edit può invocare il metodo unexecute() di Invoker che a sua volta invoca il metodo AbstractCommand::undoCommand() nell'ultimo oggetto inserito nel membro contenitore undo. Questo metodo esegue le operazioni necessarie per annullare tutte le modifiche apportate dal comando. Quindi Invoker toglie il comando dal contenitore undo e lo inserisce nel contenitore redo. Quando Premi::Controller::Presentazione::Edit invoca il metodo

Invoker::execute(), l'oggetto Invoker esegue il comando e lo sposta nuovamente dal membro contenitore redo e lo mette nel membro undo.

3.3 Observer

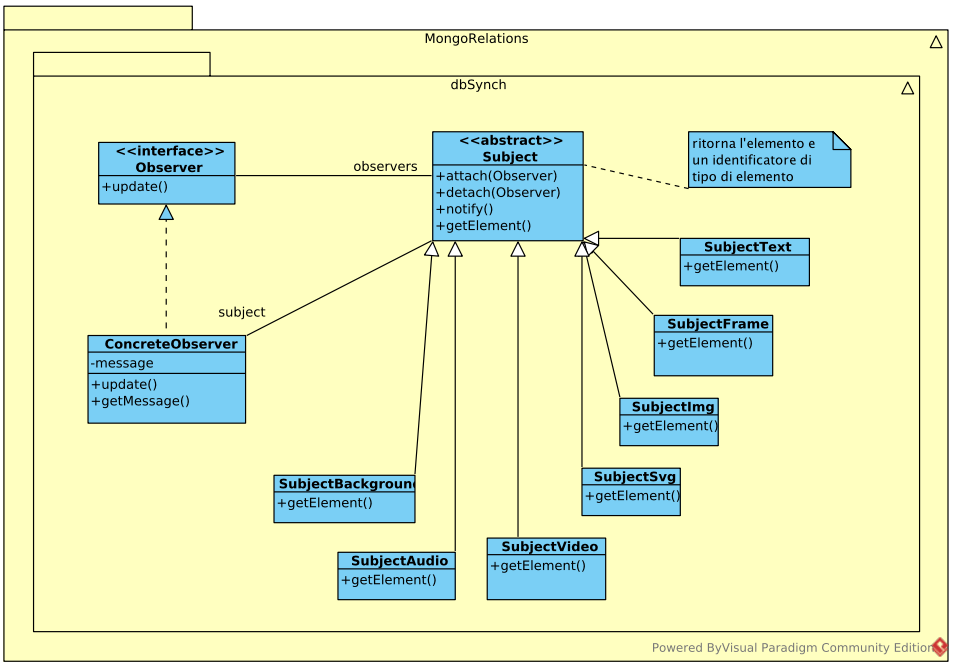

Fig 4: Observer

- **Descrizione**: L'Observer pattern è un design pattern utilizzato per tenere sotto controllo lo stato di diversi oggetti. Il pattern si basa su uno o più oggetti, chiamati listerei, che vengono registrati presso un oggetto Subject per gestire un evento che potrebbe essere generato dall'oggetto osservato.
- **Scopo dell'utilizzo**: I questo sistema il pattern viene utilizzato per aggiornare automaticamente la rappresentazione, persistente sul database MongoDB, della presentazione che l'utente sta modificando localmente.
- **Contesto d'utilizzo**: viene utilizzato in fase di modifica delle presentazioni.

3.3.1 Premi::Model::MongoRelations::DBSynch

Il package Premi::Model::MongoRelations::DBSynch contiene le classi che costituiscono il pattern Observer, in particolare le classi ConcreteSubjects non sono le classi degli oggetti che vengono modificati(che sono invece definite in Model::SlideShow::SlideShowElements) ma fungono da tramite tra questi e gli oggetti ConcreteObserver. Questo permette di avere il pattern Observer e la gerarchia di definizione degli elementi in package differenti.

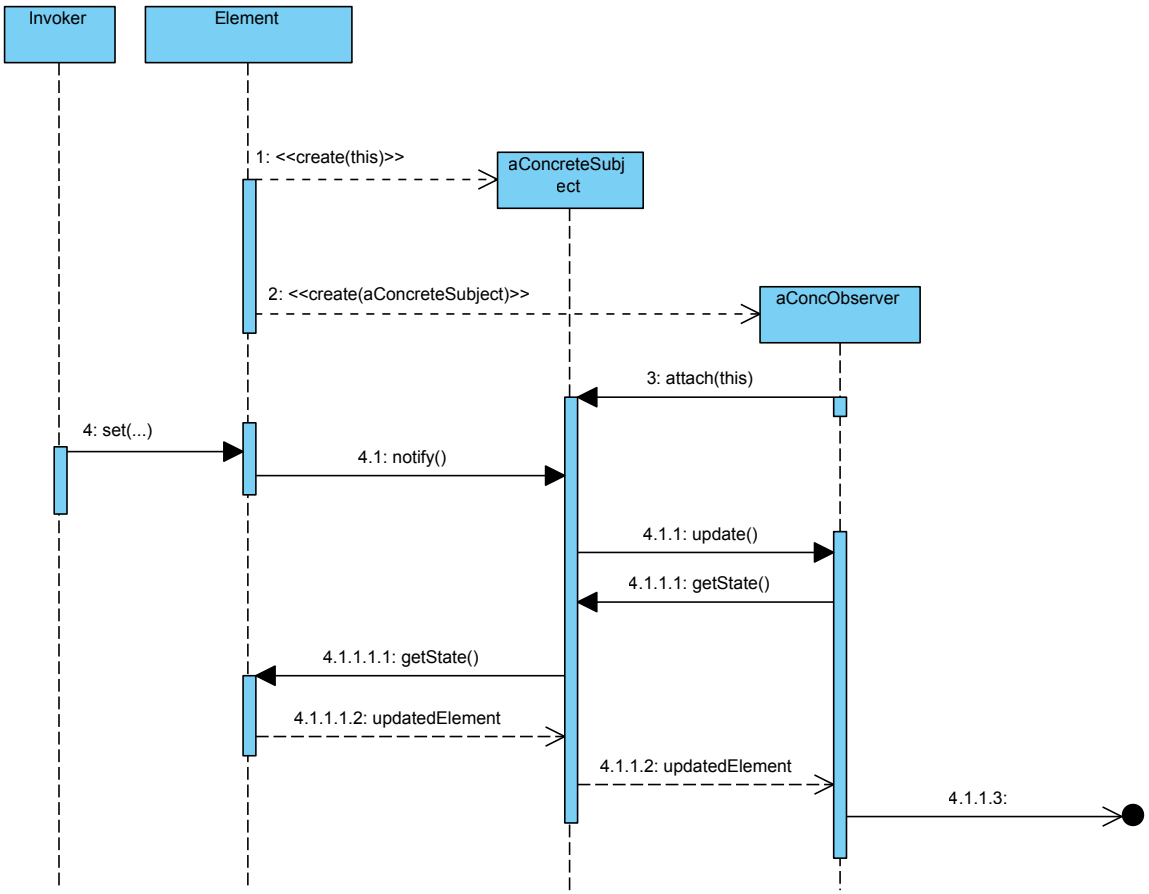

Fig 5: Diagramma di sequenza Observer

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 2.0.

4.2 Architettura generale

Il prodotto si presenta suddiviso in tre parti distinte: Model, View e Controller. Si è quindi cercato di implementare il design pattern architetturale MVC 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 Controller

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

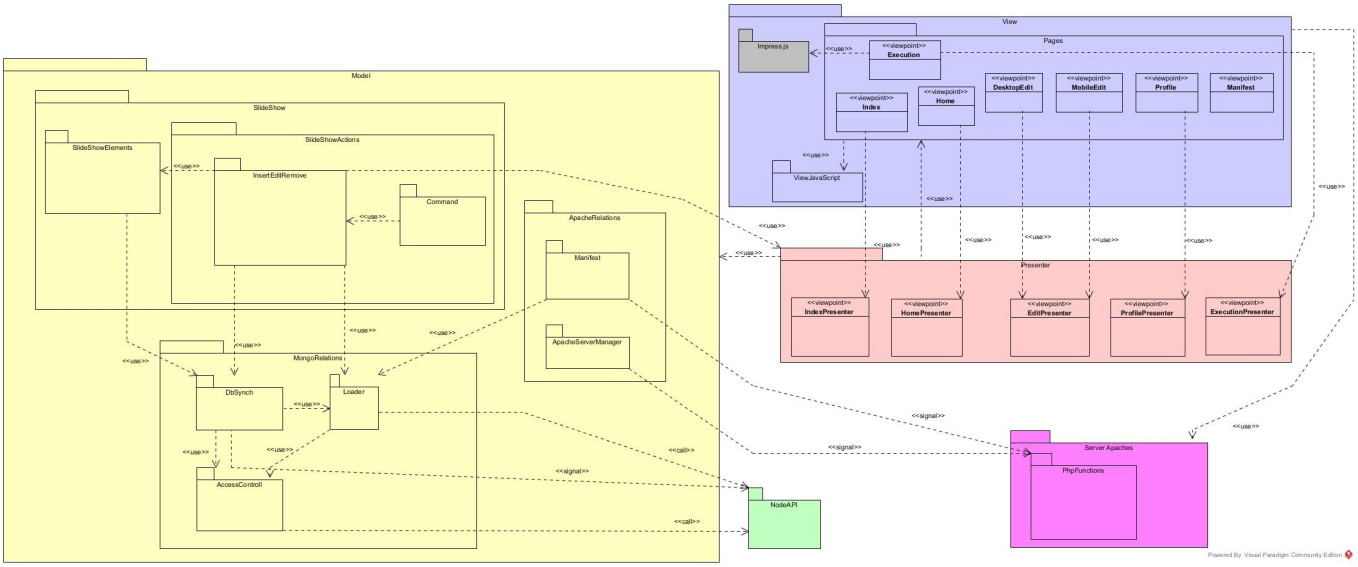

Fig 6: Architettura generale del sistema

4.3 Servizi Api nodeAPI

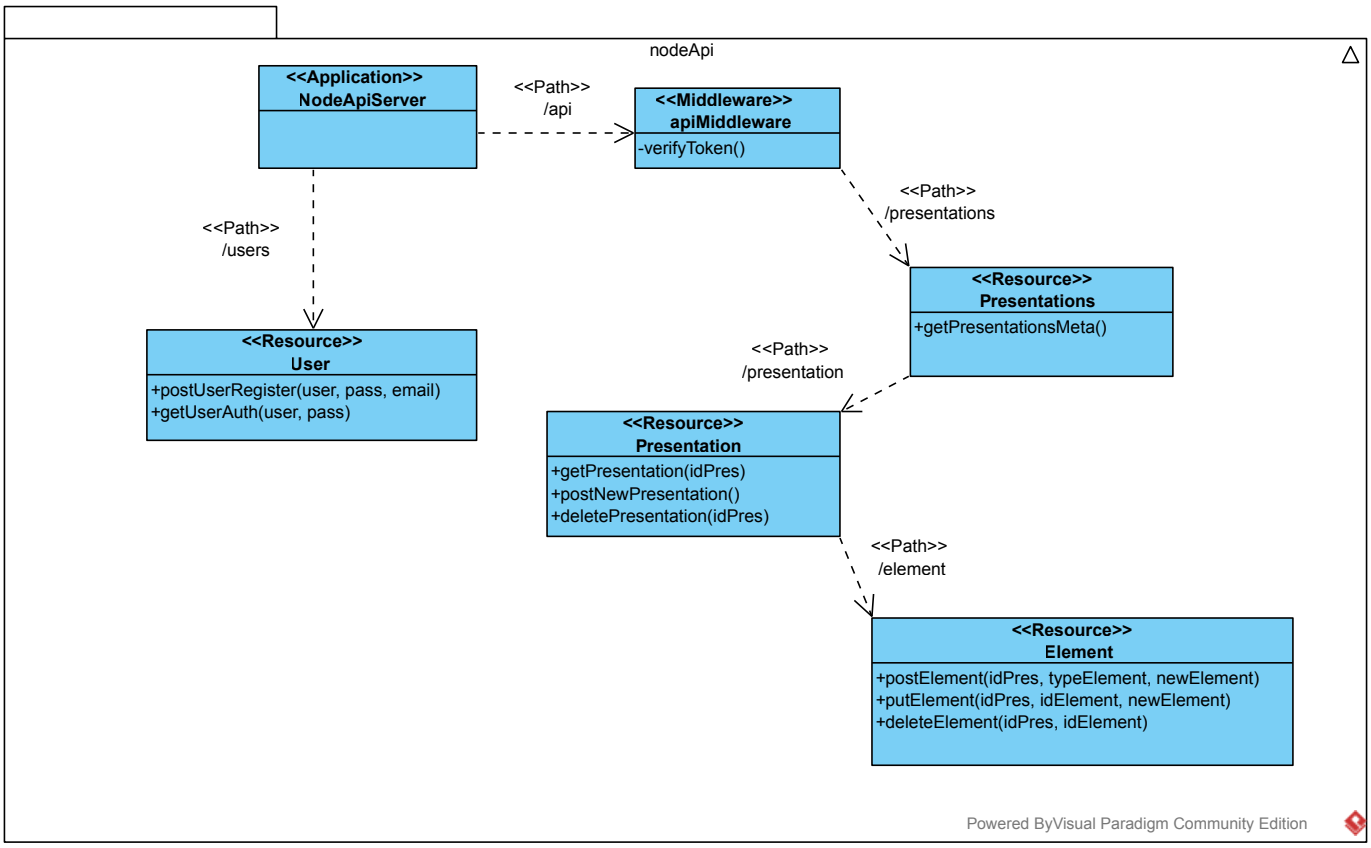

Fig 7: Servizio Api nodeApI

- **NodeApiServer:** radice dei servizi offerti da nodeApi, ovvero servizi di autenticazione e di interazione con il database MongoDB per salvare in modo persistente le presentazioni degli utenti in remoto
- **User:** servizi disponibili al path /users, offre funzionalità di registrazione e autenticazione di un utente attraverso token scambiati dal client al server ad ogni richiesta di servizio
- **apiMiddleware:** al path /api è presente un middleware per proteggere i servizi Presentations, Presentation ed Element da accessi di utenti non autenticati
- **Presentations:** al path /api/presentations è disponibile un servizio per ottenere meta-informazioni sulle presentazioni create dall'utente
- **Presentation:** al path /api/presentations/presentation sono disponibili servizi per creare una nuova presentazione e per recuperare o eliminare una presentazione dell'utente
- **Element:** al path /api/presentations/presentation/element sono disponibili servizi per inserire, sostituire o eliminare un elemento in una presentazione dell'utente

5 Descrizione dei singoli componenti

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

5.1 Model

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

- Model::SlideShow;
- Model::MongoRelations;

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, alla distruzione 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 Controller da cui riceve le chiamate relative a inserimento, eliminazione e modifica degli elementi. Inoltre comunica con il package Model::MongoRelations, inviando a questi i segnali per la modifica in tempo reale dei dati presenti nel database e inserendo gli oggetti che rappresentano gli elementi nel campo dati presentazione presente all'interno di Model::MongoRelations::Loader::LoaderClass.

5.1.2 Model::SlideShow::SlideShowActions

Tipo, obiettivo e funzione del componente: All'interno di questo Package si trovano le classi che si occupano della costruzione, dell'inserimento, della rimozione e della modifica degli elementi della presentazione.
Relazioni d'uso di altre componenti: il package è in relazione con Model::SlideShow::SlideShowActions::Command che ne invoca le funzioni passando i relativi parametri per l'inserimento, la rimozione e la modifica degli elementi. Inoltre comunica con il package Model::MongoRelations, 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>.

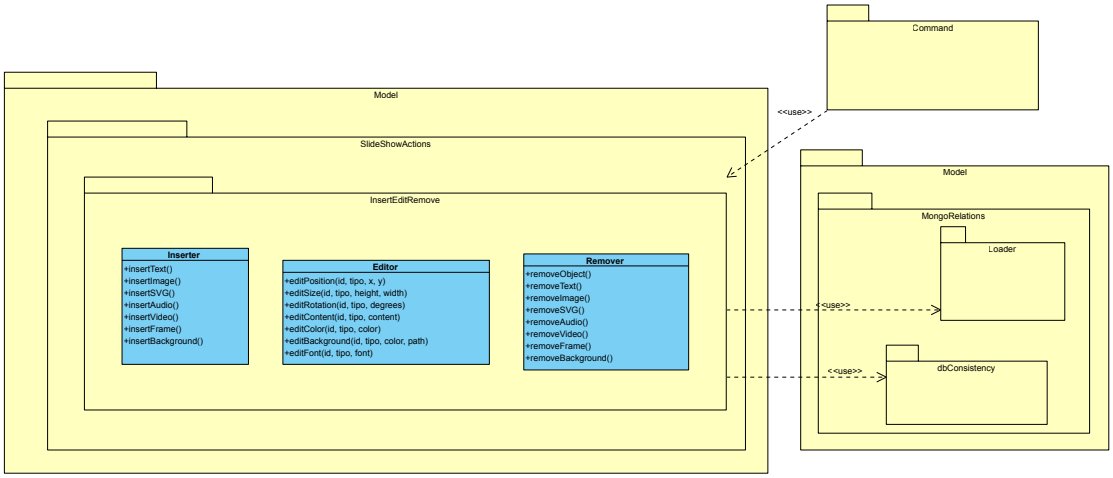

Fig 8: InsertEditRemove

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::Command che invoca i metodi delle classi del package. Inoltre Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter si occupa di costruire gli oggetti presenti nelle classi del package Model::SlideShow::SlideShowElements. InsertEditRemove è in relazione, infine, con il package Model::MongoRelations::DBSynch, infatti tramite chiamate asincrone la classe Inserter costruisce un oggetto Observer e un ConcreteSubject a esso associato.

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.
È il componente receiver del Design Pattern Command.
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;

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.
È il componente receiver del Design Pattern Command.
Relazioni d'uso di altre componenti:

- Model::SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand -> invoca il metodo insertText() messo a disposizione da Inserter;
- ConcreteFrameInsertCommand -> invoca il metodo insertFrame() messo a disposizione da Inserter;
- ConcreteImageInsertCommand -> invoca il metodo insertImage() messo a disposizione da Inserter;
- ConcreteSVGInsertCommand -> invoca il metodo insertSVG() messo a disposizione da Inserter;
- ConcreteAudioInsertCommand -> invoca il metodo insertAudio() messo a disposizione da Inserter;
- ConcreteVideoInsertCommand -> invoca il metodo insertVideo() messo a disposizione da Inserter;
- ConcreteBackgroundInsertCommand -> invoca il metodo insertBackground() messo a disposizione da Inserter;

- Model::SlideShow::SlideShowElements::Text <- Inserter costruisce gli oggetti di classe Text;
- Frame <- Inserter costruisce gli oggetti di classe Frame;
- Image <- Inserter costruisce gli oggetti di classe Image;
- SVG <- Inserter costruisce gli oggetti di classe SVG;
- Audio <- Inserter costruisce gli oggetti di classe Audio;
- Video <- Inserter costruisce gli oggetti di classe Video;
- Background <- Inserter costruisce gli oggetti di classe Background;
- Model::MongoRelations::Loader::Caricatore <- Inserter inserisce gli oggetti JSON nel campo dati contenitore presentazione.
- Model::MongoRelations::DBSynch <- Inserter costruisce un ConcreteSubject e un ConcreteObserver. L'elemento costruito da Inserter ha un riferimento al ConcreteSubject così creato.

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.
Interfacce con e relazioni d'uso e da altre componenti: È il componente receiver del Design Pattern Command.
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;
- ConcreteImageRemoveCommand -> invoca il metodo removeImage() messo a disposizione da Remover;
- ConcreteSVGRemoveCommand -> invoca il metodo removeSVG() messo a disposizione da Remover;
- ConcreteAudioRemoveCommand -> invoca il metodo removeAudio() messo a disposizione da Remover;
- ConcreteVideoRemoveCommand -> invoca il metodo removeVideo() messo a disposizione da Remover;
- 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;
- Frame <- Editor invoca i metodi di set degli oggetti di classe Frame;
- Image <- Editor invoca i metodi di set degli oggetti di classe Image;
- SVG <- Editor invoca i metodi di set degli oggetti di classe SVG;
- Audio <- Editor invoca i metodi di set degli oggetti di classe Audio;
- Video <- Editor invoca i metodi di set degli oggetti di classe Video;
- Background <- Editor invoca i metodi di set degli oggetti di classe Background;
- Text <- Editor invoca i metodi di set degli oggetti di classe Text;
- Frame <- Editor invoca i metodi di set degli oggetti di classe Frame;
- Image <- Editor invoca i metodi di set degli oggetti di classe Image;
- SVG <- Editor invoca i metodi di set degli oggetti di classe SVG;
- Audio <- Editor invoca i metodi di set degli oggetti di classe Audio;
- Video <- Editor invoca i metodi di set degli oggetti di classe Video;
- Background <- Editor invoca i metodi di set degli oggetti di classe Background;

5.1.4 Model::SlideShow::SlideShowActions::Command

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

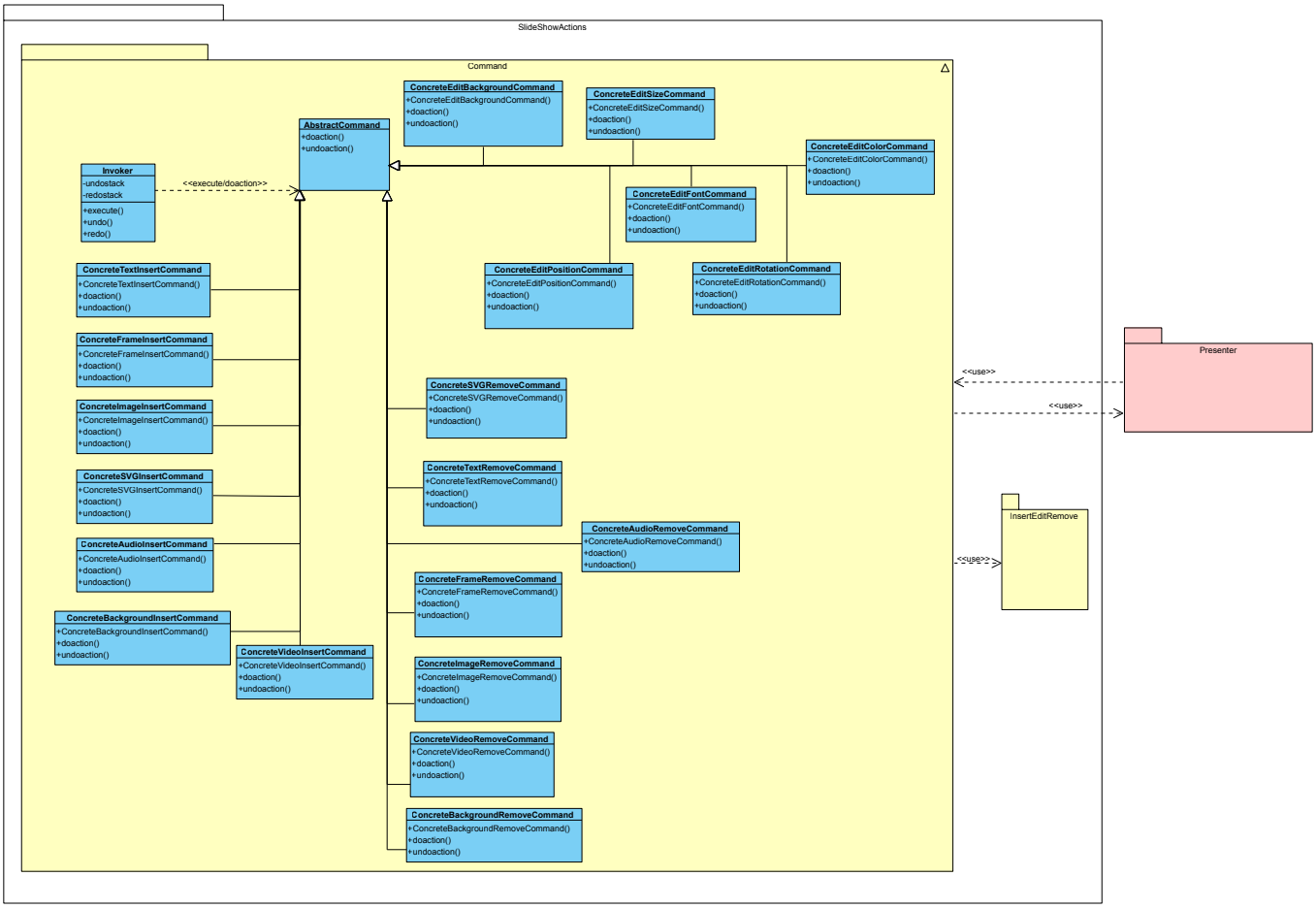

Fig 9: Command Package

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::InsertEditRemove;

Controller::EditController costruisce gli oggetti delle sottoclassi di AbstractCommand, inoltre quando viene invocato il metodo undo() di un comando concreto, questo invoca il metodo update() di EditController.

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:

- Controller::MobileEdit->crea un oggetto di una sottoclasse di Model::SlideShow::SlideShowActions::Command::AbstractCommand passandolo all'Invoker che ne invoca il metodo execute() e lo inserisce nello stack "undostack", richiama il metodo che svuota lo stack "redostack".
  Può inoltre invocare il metodo "undo()" dell'Invoker che provvede a richiamare il metodo undoaction() del comando sulla cima dello stack "undostack" e a spostarlo quindi nello stack "redostack". Alternativamente invoca il metodo "redo()" dell'Invoker che provvede a invocare il metodo doaction() del comando sulla cima dello stack "redostack" e a spostarlo quindi nello stack "undostack";

- Controller::DesktopEdit->si comporta in modo analogo a MobileEdit;

- Model::SlideShow::SlideShowActions::Command::AbstractCommand <- Invoker invoca il metodo doaction() dell'oggetto della sottoclasse di AbstractCommand. Alternativamente invoca il metodo undoaction().

Interfacce con e relazioni d'uso e da altre componenti: Viene invocato per effettuare le operazioni di modifica alla presentazione, a sua volta invoca i metodi doaction() o undoaction() di una classe derivata da Model::SlideShow::SlideShowActions::Command::AbstractCommand per eseguire materialmente il comando. Quando un comando viene eseguito, Invoker lo salva in un array $undostack[ ].

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 il metodo doaction(); inoltre provvede ad annullare l'ultima operazione invocandone il metodo undoaction().

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:

- ConcreteTextInsertCommand;
- ConcreteFrameInsertCommand;
- ConcreteImageInsertCommand;
- ConcreteSVGInsertCommand;
- ConcreteAudioInsertCommand;
- ConcreteVideoInsertCommand;
- ConcreteBackgroundInsertCommand;

- ConcreteTextRemoveCommand;
- ConcreteFrameRemoveCommand;
- ConcreteImageRemoveCommand;
- ConcreteSVGRemoveCommand;
- ConcreteAudioRemoveCommand;
- ConcreteVideoRemoveCommand;
- ConcreteBackgroundRemoveCommand;
- ConcreteEditSizeCommand;
- ConcreteEditPositionCommand;
- ConcreteEditRotationCommand;
- ConcreteEditColorCommand;
- ConcreteEditBackgroundCommand;
- ConcreteEditFontCommand;
- 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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertText(...) della classe statica per l'inserimento di un elemento;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertFrame(...) della classe statica per l'inserimento di un elemento frame nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertImage(...) della classe statica per l'inserimento di un elemento immagine nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertSVG(...) della classe statica per l'inserimento di un elemento SVG nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertAudio(...) della classe statica per l'inserimento di un elemento audio nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertVideo(...) della classe statica per l'inserimento di un elemento video nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter <- invoca il metodo insertBackground(...) della classe statica per l'inserimento di un elemento sfondo nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeText(...) della classe statica per la rimozione di un elemento testuale nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> invoca il costruttore della classe e passa l'oggetto così creato all'Invoker;
- Model::SlideShow::SlideShowActions::Command::Invoker -> invoca il metodo doaction() del comando e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeFrame(...) della classe statica per la rimozione di un elemento frame dalla presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeImage(...) della classe statica per l'eliminazione di un elemento immagine dalla presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeSVG(...) della classe statica per l'eliminazione di un elemento SVG dalla presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeAudio(...) della classe statica per l'eliminazione di un elemento immagine dalla presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeVideo(...) della classe statica per l'eliminazione di un elemento video dalla presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover <- invoca il metodo removeBackground(...) della classe statica per l'eliminazione dell'elemento sfondo dalla presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- il comando invoca il metodo editSize(...) della classe statica per la modifica dei campi dati relativi alle dimensioni dell'oggetto nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- il comando invoca il metodo editPosition(...) della classe statica per la modifica dei campi dati relativi alla posizione dell'oggetto nella presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- il comando invoca il metodo editColor(...) della classe statica per la modifica del campo dati relativo al colore dell'oggetto della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- il comando invoca il metodo editBackground(...) della classe statica per la modifica del campo dati relativo allo sfondo dell'oggetto della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- il comando invoca il metodo editRotation(...) della classe statica per la modifica del campo dati relativo all'orientamento dell'oggetto della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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:

- Controller::EditController -> 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 e lo inserisce nel campo dati undostack e ne setta il valore del campo dati booleano executed a true, o ne invoca il metodo di annullamento undoaction() e lo inserisce nel campo dati redostack;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor <- il comando invoca il metodo editColor(...) della classe statica per la modifica dei campi dati relativi al font dell'oggetto testuale della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo update() di EditController quando viene invocato il metodo doaction() e il campo dati booleano executed ha valore true, o quando viene invocato il metodo undoaction().

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>.

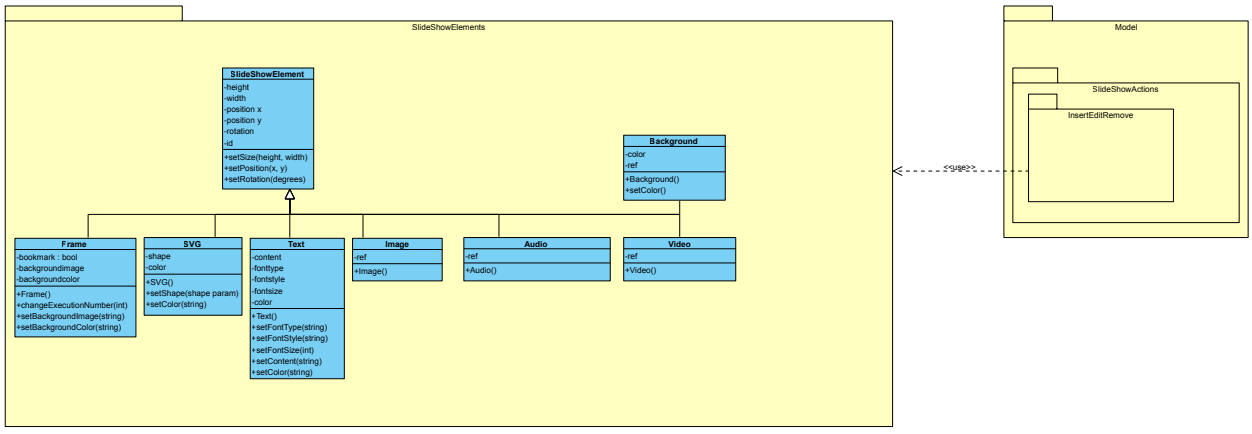

Fig 10: SlideShowElements

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.
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 MongoRelations::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;
- Model::DBSynch, i metodi di set degli oggetti delle classi del package SlideShowElements, infatti, invocano il metodo notify dell'observer contenuto nel package DBSynch a cui l'oggetto è associato.

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::InsertEditRemove::Inserter -> invoca il costruttore delle sottoclassi di SlideShowElement e li inserisce nel campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass;
- Model::SlideShow::SlideShowActions::InsertEditRemove::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;
- Model::DBSynch::ConcreteObserver <- i metodi di set degli oggetti delle sottoclassi di SlideShowElements invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto tiene un riferimento.

Interfacce con e relazioni d'uso e da altre componenti:
Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter istanzia oggetti di sottoclassi di SlideShowElement e li inserisce nel campo dati contenitore presentazione all'interno di Model::MongoRelations::Model:LoaderClass
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Text dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Frame dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Image dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto SVG dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;

- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Audio dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Video dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
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::MongoRelations::Loader::Caricatore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Video dal campo dati presentazione all'interno di Model::MongoRelations::Loader::LoaderClass, ne invoca quindi il distruttore;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto;
- Model::DBSynch::ConcreteObserver <- i metodi di set della classe invocano il metodo notify() dell'observer contenuto nel package DBSynch di cui l'oggetto della classe tiene un riferimento.

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::MongoRelations::Loader::LoaderClass.
Classi ereditate:

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

5.1.7 Model::MongoRelations

Tipo, obiettivo e funzione del componente: package, racchiude le funzionalità del sistema che interagiscono con i servizi di interazione con il database MongoDB esposti dall' interfaccia nodeApi.
Relazioni d'uso di altre componenti:

- relazioni verso **nodeApi** del quale si utilizzano i servizi dalla interfaccia;
- relazioni da **Controller** per il recupero o la creazione di una presentazione dal database MongoDB;
- relazioni da **Model::SlideShow** che utilizza la rappresentazione locale della presentazione e i servizi per l'interazione a nodeApi.

5.1.8 Model::MongoRelations::Loader

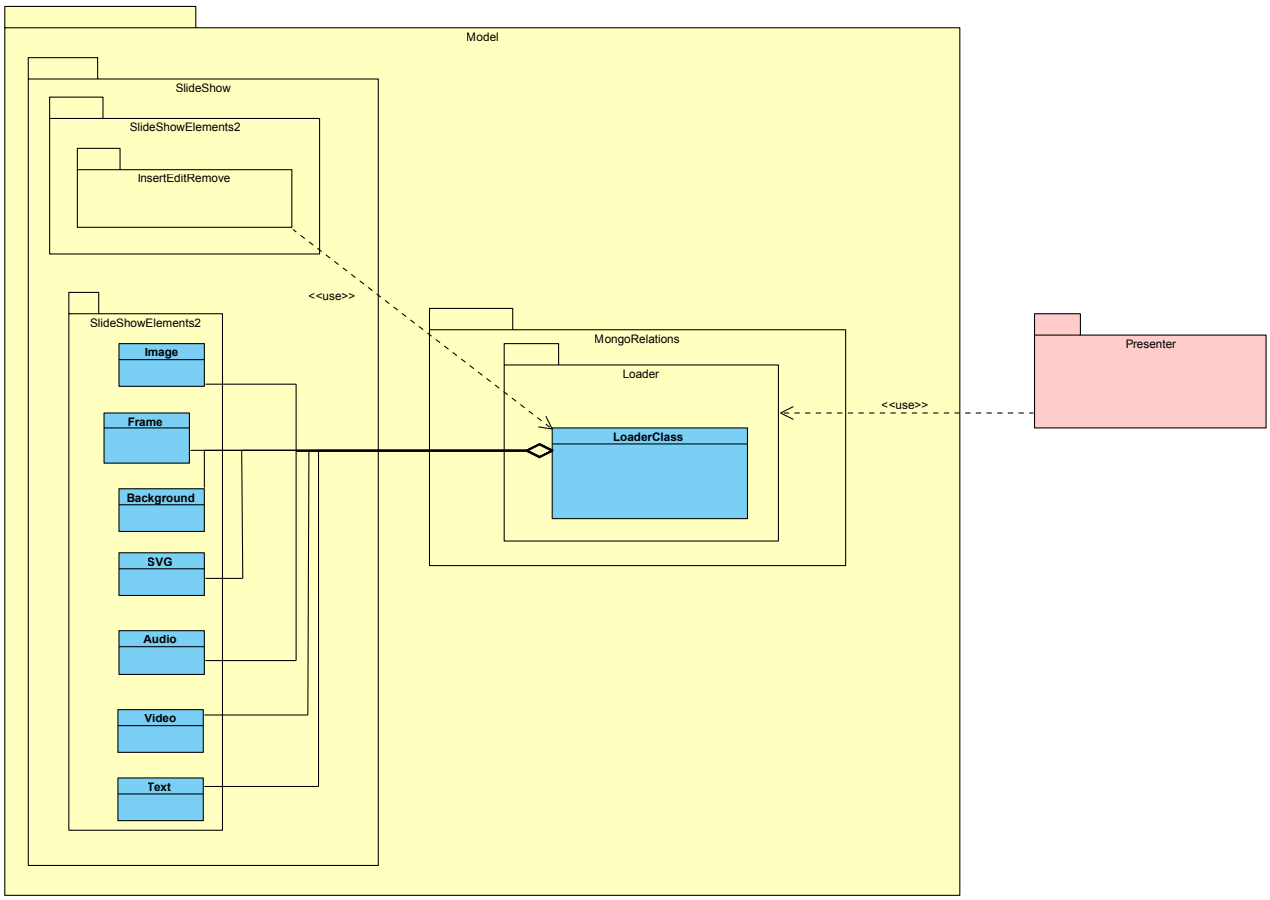

Fig 11: MongoRelationsLoader

Tipo, obiettivo e funzione del componente: package, racchiude le funzioni di recupero o creazione di una presentazione dal server attraverso i servizi nodeApi, una volta ottenuta la presentazione e' esposta per le modifiche provenienti da altri package nel Model **Relazioni d'uso di altre componenti**:

- relazione verso **nodeApi** per il recupero della presentazione dal database MongoDB
- relazione da **Model::SlideShow::InsertEditRemove** a cui viene esposta la rappresentazione della presentazione locale per essere modificata

5.1.8.1 LoaderClass

Tipo, obiettivo e funzione del componente: Classe la cui funzione è recuperare una presentazione dal database remoto ed esporla alle modifiche da parte di altri metodi definiti nel model, creare una nuova presentazione
Relazioni d'uso di altre componenti:

- relazione verso **nodeApi** per il recupero della presentazione dal database MongoDB
- relazione da **Model::SlideShow::InsertEditRemove** a cui viene esposta la rappresentazione della presentazione locale per essere modificata

5.1.9 Model::MongoRelations::AccessControl

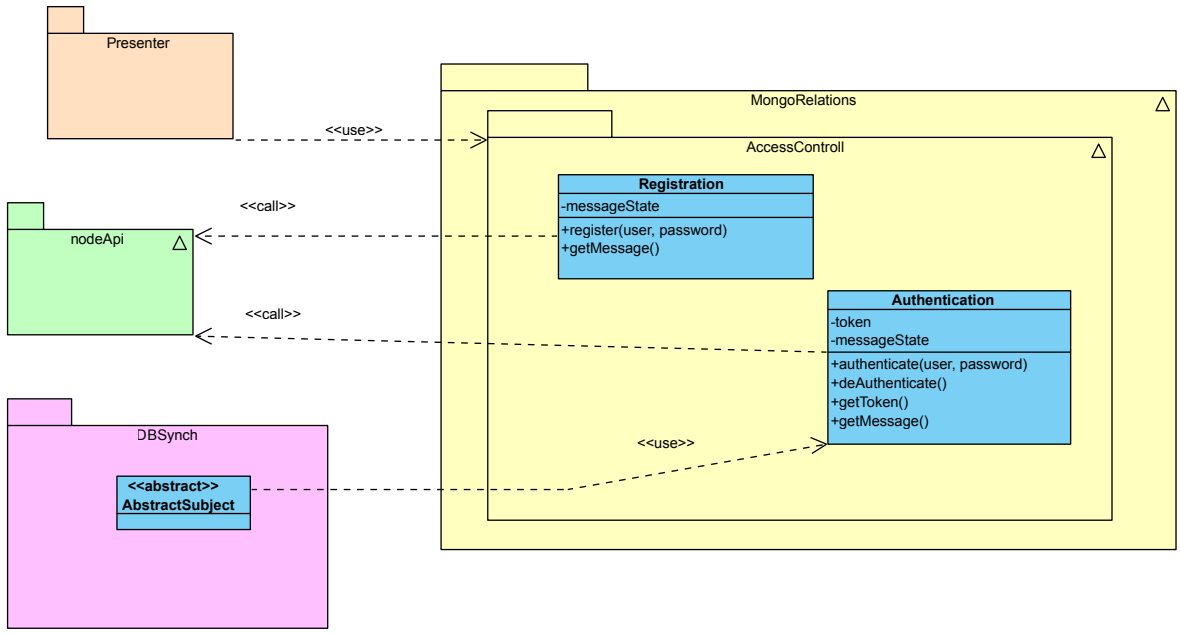

Fig 12: DbSync

Tipo, obiettivo e funzione del componente: package, racchiude le funzioni di registrazione dell'utente e autenticazione tramite token ai servizi esposti da nodeApi.
Relazioni d'uso di altre componenti:

- relazioni verso **nodeApi** a cui vengono passati i parametri per la registrazione e l'autenticazione dell'utente
- relazioni da **Controller** da cui si ricevono i parametri in input dell'utente
- relazioni da **DBSynch** a cui viene esposto il token ricevuto da nodeApi dopo la autenticazione

5.1.9.1 Autenticazione

Tipo, obiettivo e funzione del componente: Classe, fornisce le funzionalità di autenticazione e deautenticazione.
Relazioni d'uso di altre componenti:

- relazione verso **nodeApi** per il recupero del token passando i parametri di autenticazione dell'utente
- relazione da **Controller** da cui riceve in input i parametri dell'utente per la autenticazione
- relazione da **Model::MongoRelations::DBSynch** a cui espone il token per poter usare i servizi di nodeApi

5.1.9.2 Registrazione

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

- relazione verso **nodeApi** per il la registrazione dell'utente presso il database MongoDB
- relazione da **Controller** da cui riceve in input i parametri dell'utente per la registrazione

5.1.10 Model::MongoRelations::DBSynch

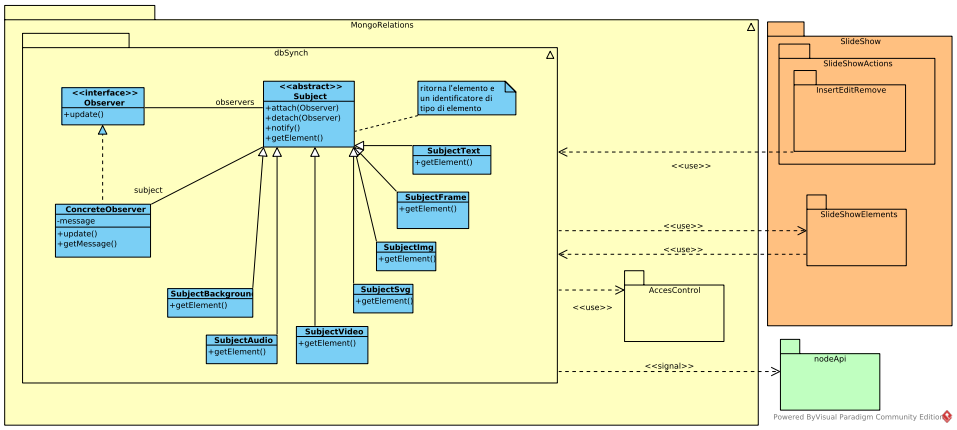

Fig 13: DbSync

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:

- relazioni da e verso **Model::SlideShow::SlideShowElements** in cui si detiene un riferimento nei confronti dei ConcreteSubjects e viceversa

- relazioni verso **Model::MongoRelations::AccessControll** da cui riceve il token per accedere ai servizi nodeApi
- relazioni verso **nodeApi** verso cui inoltra le chiamate di update degli elementi della presentazione modificata

5.1.10.1 Observer

Tipo, obiettivo e funzione del componente: Interfaccia, espone il metodo update() che una volta recuperato l'elemento modificato chiama il metodo in nodeApi per l'aggiornamento della presentazione nel database MongoDB
Relazioni d'uso di altre componenti:

- associazione con Subject che detiene un riferimento dell'Observer
- interfaccia realizzata da ConcreteObserver che definisce il metodo update()

5.1.10.2 ConcreteObserver

Tipo, obiettivo e funzione del componente: Classe, concretizza l'interfaccia Observer
Relazioni d'uso di altre componenti:

- realizza l'interfaccia Observer definendone il metodo update()
- detiene un riferimento a Subject
- relazione verso nodeApi di cui ne chiama i servizi per l'aggiornamento di una presentazione

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 aggiungere, togliere e notificare il ConcreteObserver associato.
Relazioni d'uso di altre componenti:

- associazione da ConcreteObserver che ne detiene un riferimento;
- 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 e mantenere gli oggetti le cui modifiche sono da notificare in un altro package.
Relazioni d'uso di altre componenti:

- implementa Subject definendo il metodo getElement(), associazione da e verso 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 MongoRelations.

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.

- Controller::EditController <- 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 Premi::ApacheRelations::ResourceGetter

La seguente classe appartiene al package ApacheRelations, quindi lo scope sarà Model::ApacheRelations::<componente>. **Tipo, obiettivo e funzione del componente**: Questo package ha lo scopo di rendere disponibili le presentazioni in locale tramite chiamate a funzioni o servizi del server Apache.
Relazioni d'uso di altre componenti:

- dipendenza con Premi::Model::MongoRelations::Loader::LoaderClass <- ResourceGetter invoca i metodi forniti da LoaderClass che restituiscono ;
- definisce il metodo getResources();

5.2 View

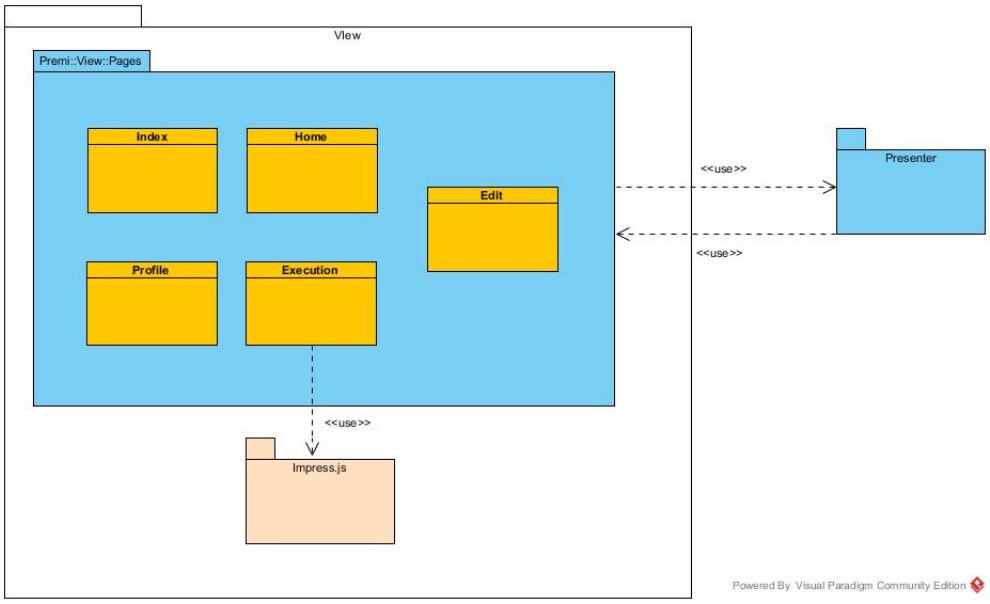

Fig 14: View

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 e la comunicazione con il Controller.

5.2.1 View::Pages

Tipo, obiettivo e funzione del componente: questo package costituisce le pagine fisiche del sistema, realizzate in HTML.
Relazioni d'uso di altre componenti: il componente comunica con il package Premi::Controller per l'utilizzo delle funzioni presenti all'interno dello stesso per l'interazione dell'utente con il sito.

5.2.2 View::Pages::Index

Tipo, obiettivo e funzione del componente: la classe Index definisce la struttura, e la conseguente visualizzazione, della pagina web che consente ad un utente di effettuare login e registrazione al sistema.
Relazioni d'uso di altre componenti: la classe Index utilizza i metodi messi a disposizione dalla classe Controller::IndexController, contenuta nel package Controller, per verificare i dati inseriti durante la fase di autenticazione, per inviare i dati relativi alla registrazione e per visualizzare eventuali errori emersi nella fase di autenticazione/registrazione.
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.

5.2.3 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 Controller::HomeController per l'eliminazione delle presentazioni dal server, per scaricare una presentazione in locale e per effettuare il logout.
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.

5.2.4 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 Controller::ProfileController, per il caricamento di file media nel server, per la loro eliminazione dal server, per la modifica della password e per rinominarli.
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.

5.2.5 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 Controller::ExecutionController per creare la pagina che verrà eseguita da Impress.js.
Attività svolte e dati trattati: La classe definisce la struttura della pagina web che consente agli utenti di eseguire la presentazione spostandosi con la tastiera avanti e indietro, passare al capitolo successivo oppure selezionare un nuovo percorso.

5.2.6 View::Pages::Edit

Tipo, obiettivo e funzione del componente: la classe Edit definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra l'editor di modifica di una presentazione.
Relazioni d'uso di altre componenti: la classe Edit utilizza i metodi messi a disposizione

dalla classe Controller::EditController 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.
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.

5.3 Controller

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.

5.3.1 Controller::EditController

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

- ConcreteBackgroundRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditSizeCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditPositionCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditRotationCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditColorCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditBackgroundCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditFontCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- ConcreteEditContentCommand <- EditController costruisce un comando e lo dà in pasto a Model:Invoker;
- Invoker <- EditController costruisce l'oggetto di classe Invoker. Invoca il metodo execute() di Invoker, passando come parametro un oggetto di classe Command oppure invoca il metodo unexecute() di Invoker;
- Model::ApacheManager::FileManager <- EditController invoca i metodi uploadFile() di FileManager quando viene inserito nella presentazione un file non ancora presente nel server;
- Model::ApacheRelations::ResourceGetter <- la classe della view invoca il metodo save() presente in ExecutionController che a sua volta invoca il metodo getResources() di ResourceGetter che aggiorna il file manifest con tutti gli elementi della presentazione e lo ricarica.
- View::Pages::DesktopEdit e View::Pages::MobileEdit -> invocano i metodi di EditController, passando i parametri degli oggetti modificati;
- View::Pages::DesktopEdit e View::Pages::MobileEdit <- quando il logout ha successo EditController comunica alla view di effettuare una redirect verso Index;
- Model::MongoRelations::Loader::Autenticazione <- Quando la view invia una richiesta di logout, EditController invoca il metodo di Autenticazione deAuthenticate(), che termina la sessione.
- Model::MongoRelations::Loader::Loaderclass <- EditController invoca i metodi forniti da LoaderClass che restituiscono l'oggetto JSON della presentazione.

- View::ViewJavascript <- EditController invoca le funzioni Javascript passando l'oggetto presentazione, destinate alla sua traduzione in formato HTML. Inoltre, EditController invoca le funzioni ViewJavascript ogniqualvolta sia richiesta una modifica della View a seguito di una modifica del model, nello specifico nelle operazioni di annulla e ripristina, gestite dal package Model::SlideShow::SlideShowActions::Command.

- Model::SlideShow::SlideShowActions::Command -> le sottoclassi concrete di AbstractCommand invocano il metodo di update di EditController, tale metodo invoca quindi le funzioni di ViewJavascript destinate alla modifica della pagina HTML.

Interfacce con e relazioni d'uso e da altre componenti: La pagina DesktopEdit o la pagina MobileEdit invia a EditController comunica l'avvenuta modifica o la rimozione di un elemento della presentazione o l'inserimento di un nuovo elemento invocando i metodi corrispondenti di EditController. EditController istanzia un oggetto di una sottoclasse di Model::SlideShow::SlideShowActions::Command::AbstractCommand e lo dà in pasto a Model::Invoker. Eventualmente EditController, dopo che la View ha invocato il metodo undo() di EditController, può semplicemente annullare il comando appena eseguito invocando il metodo unexecute di Invoker. La pagina web può, inoltre richiedere il caricamento di una presentazione o la creazione di una nuova presentazione a EditController, che, tramite invocazione dei metodi di Model::MongoRelations::Loader::LoaderClass, caricherà dal database.

5.3.2 Controller::HomeController

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 HomeController, ne invoca i metodi passando i parametri dell'utente;

- Model::MongoRelations::Loader::LoaderClass <- HomeController invoca un metodo di LoaderClass che restituisce l'elenco dei titoli delle presentazioni dell'utente;

- Model::MongoRelations::Loader::Autenticazione <- Quando la view invia una richiesta di logout, HomeController invoca il metodo deAuthenticate() fornito da Autenticazione, che termina la sessione;

- Model::ApacheRelations::ResourceGetter <- la classe della view invoca il metodo save() presente in HomeController passando per parametro un array di id di presentazioni che l'utente intende scaricare in locale, a sua volta HomeController invoca il metodo update() di ResourceGetter che controlla se esiste già un file manifest dopodiché lo aggiorna con tutti i riferimenti alle pagine da scaricare e lo ricarica.

5.3.3 Controller::ExecutionController

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 ExecutionController, ne invoca i metodi passando i parametri della presentazione da caricare;
- Model::MongoRelations::Loader::LoaderClass <- ExecutionController passa i parametri di caricamento al Loader che carica la presentazione attraverso nodeAPI e la restituisce a ExecutionController;
- View::ViewJavascript <- ExecutionController invoca le funzioni di ViewJavascript preposte alla traduzione della presentazione nel codice HTML interpretabile da Impress.

5.3.4 Controller::IndexController

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

5.3.5 Controller::ProfileController

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

- Model::ApacheRelations::FileManager <- EditController invoca i metodi di FileManager per caricare un file nel server, per modificarne il nome o per eliminarlo dal server;
- Model::MongoRelations::Loader::Autenticazione <- Quando la view invia una richiesta di logout, ProfileController invoca il metodo di Autenticazione deAuthenticate(), che termina la sessione. ProfileController invia quindi una richiesta di redirect alla pagina Index.

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. Dopodiché l'utente potrà decidere se modificare la propria password, gestire, modificare o eseguire le proprie presentazioni oppure gestire il proprio profilo.

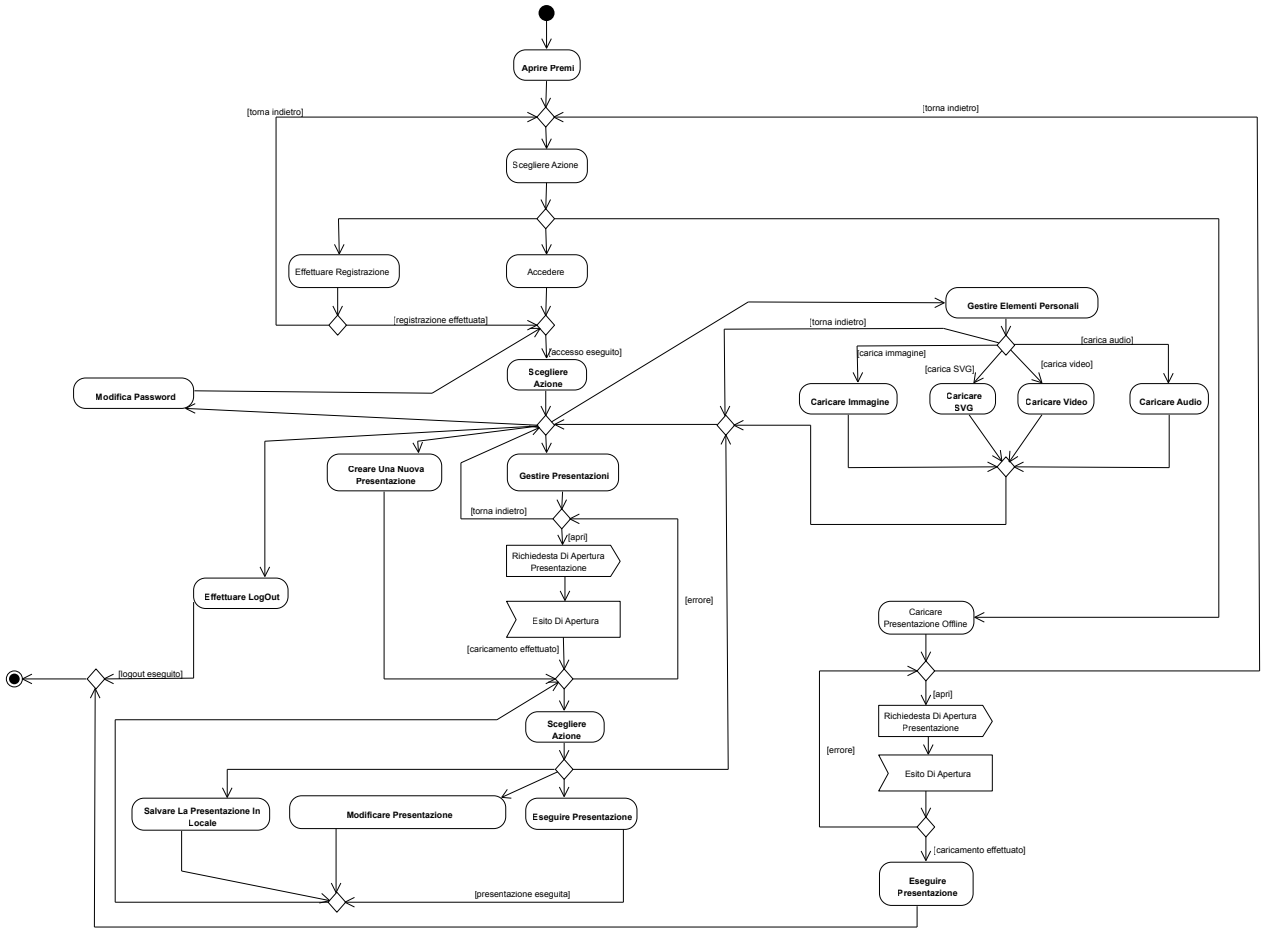

Fig 15: Attività Principali

6.1.1 Gestione presentazioni

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

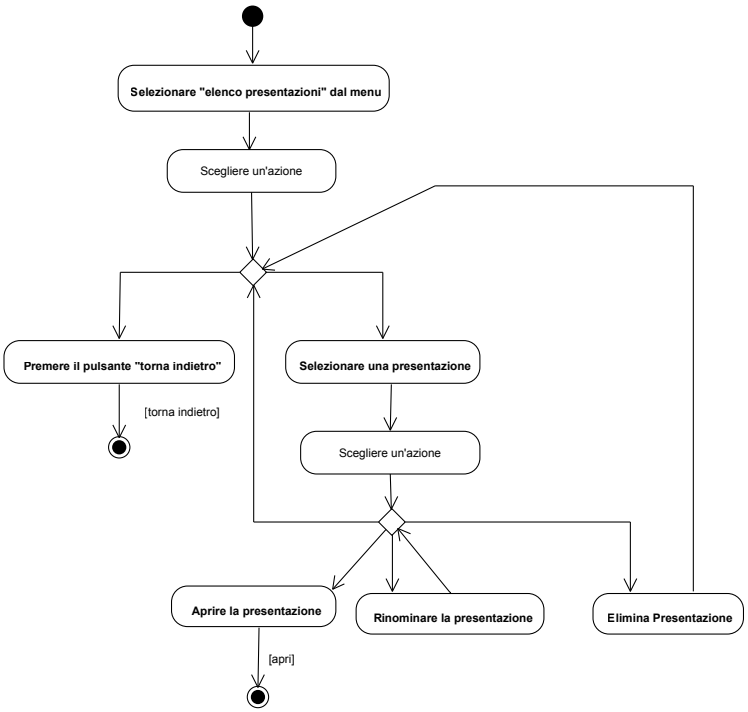

Fig 16: Gestione Presentazioni

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.

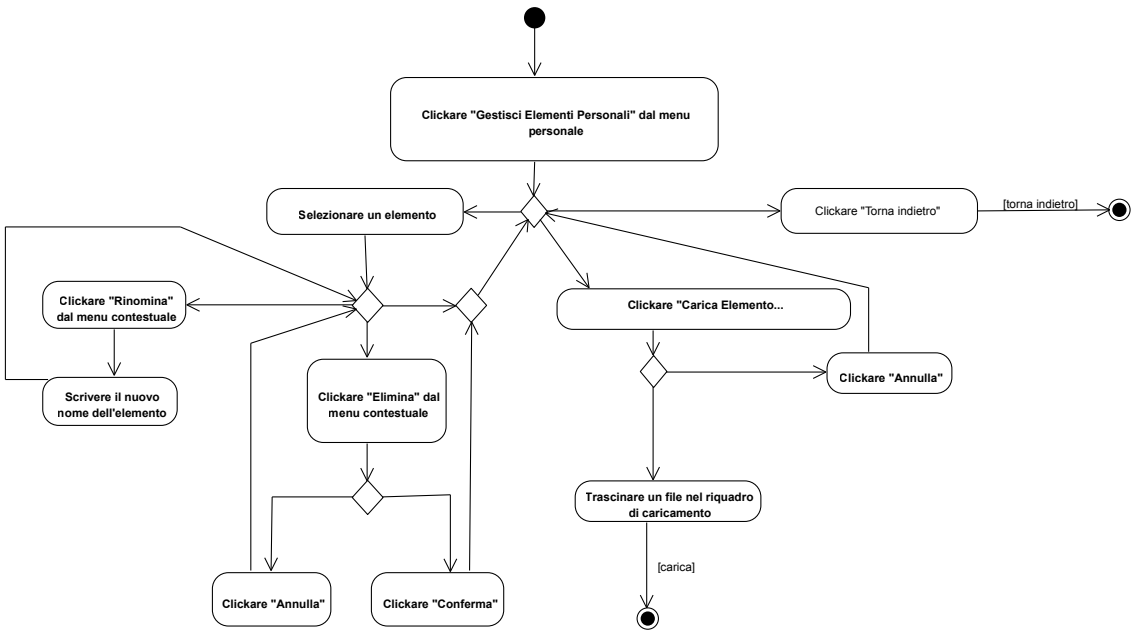

Fig 17: Caricare File

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.

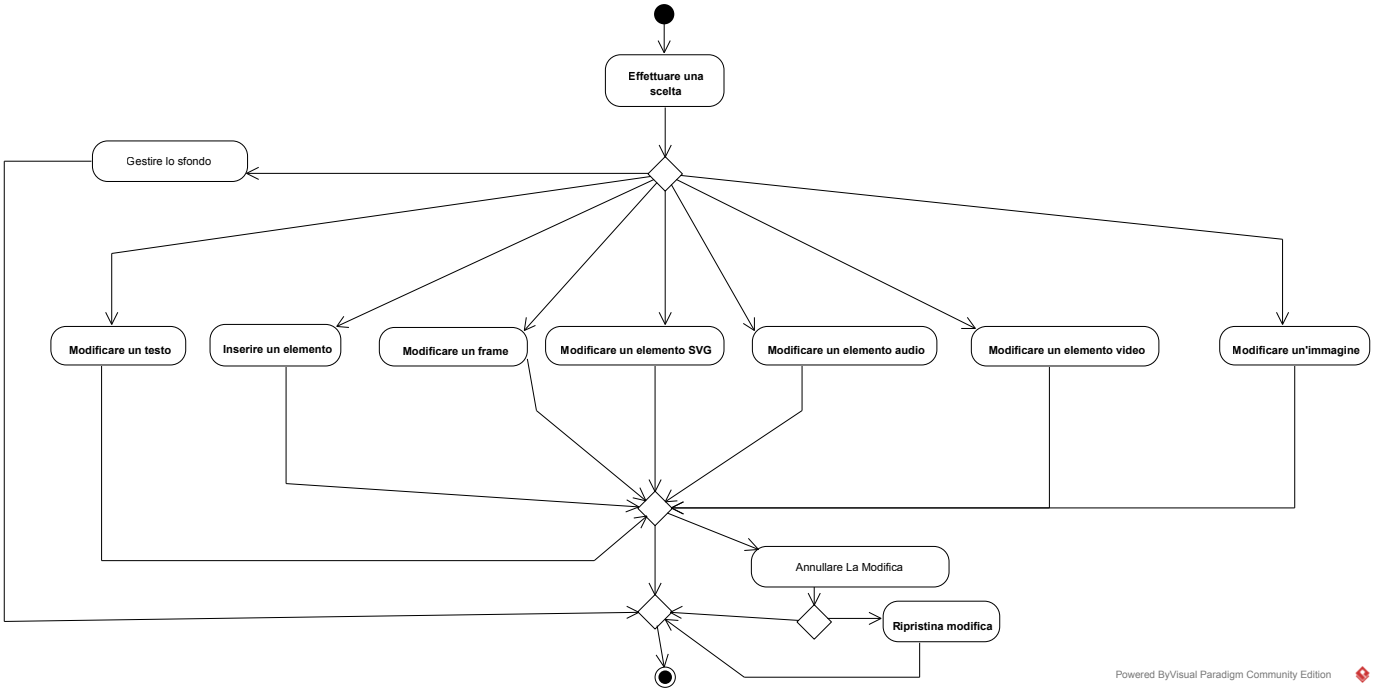

Fig 18: Modificare Presentazione da Desktop

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.

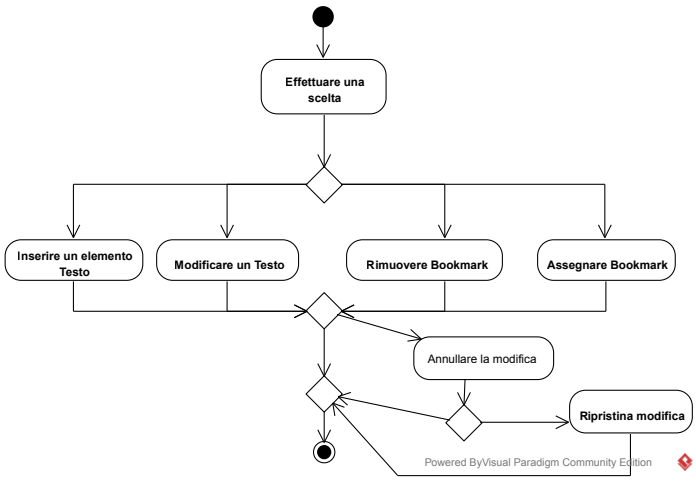

Fig 19: Modificare Presentazione da Mobile

6.1.5 Gestire Sfondo

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

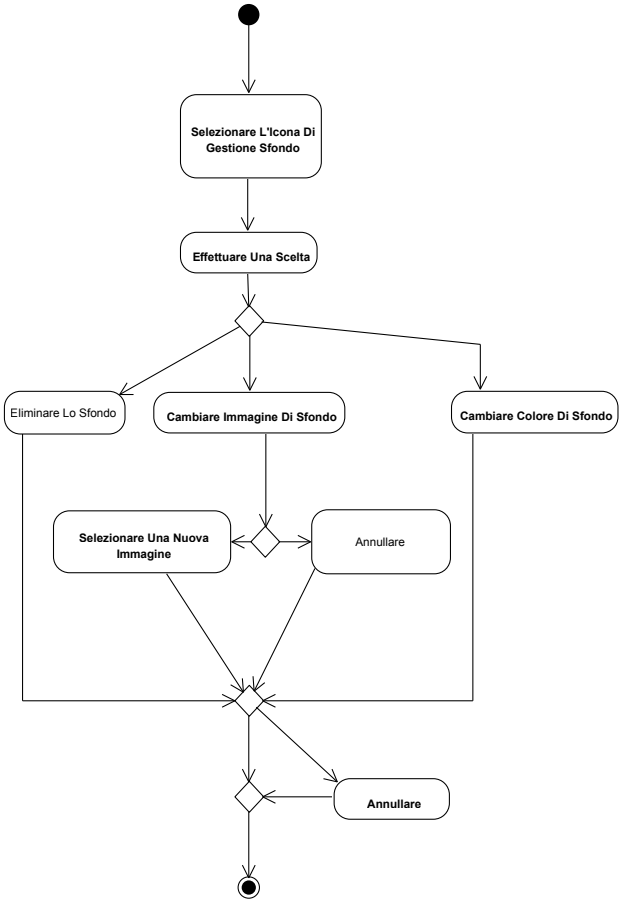

Fig 20: Gestire Sfondo

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.

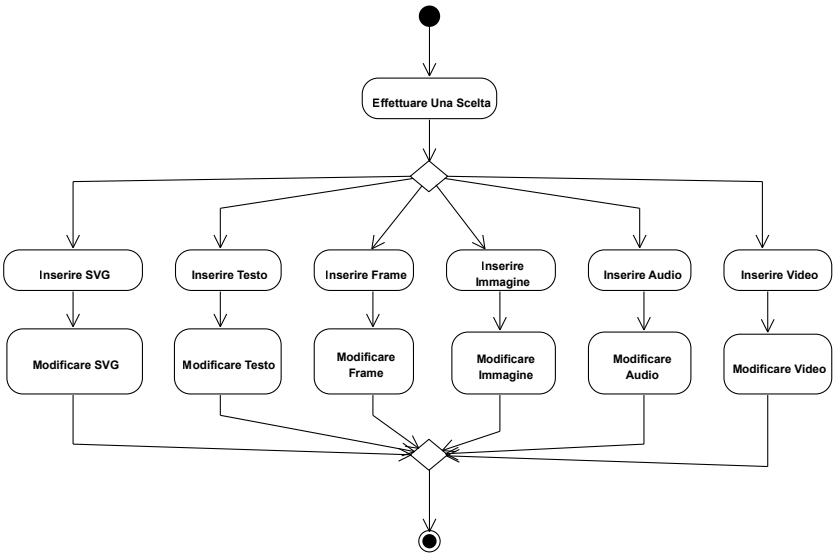

Fig 21: Inserire Elemento

6.1.7 Modificare Elemento

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

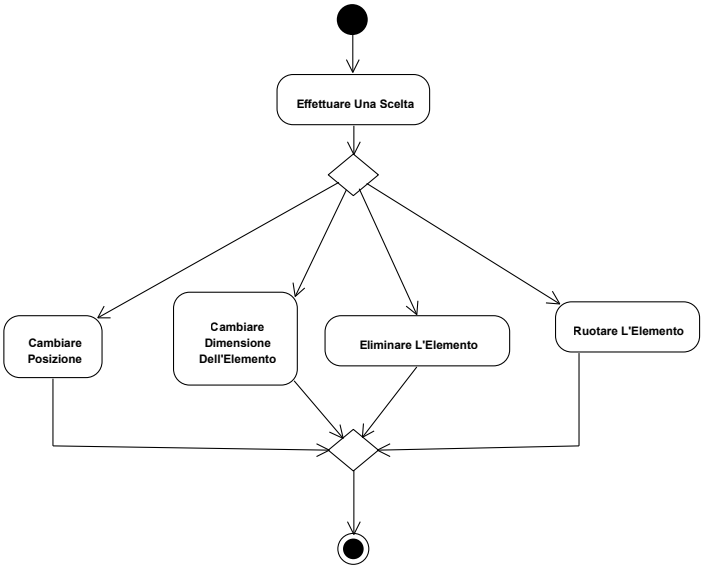

Fig 22: Modificare Elemento

6.1.8 Modificare Frame

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

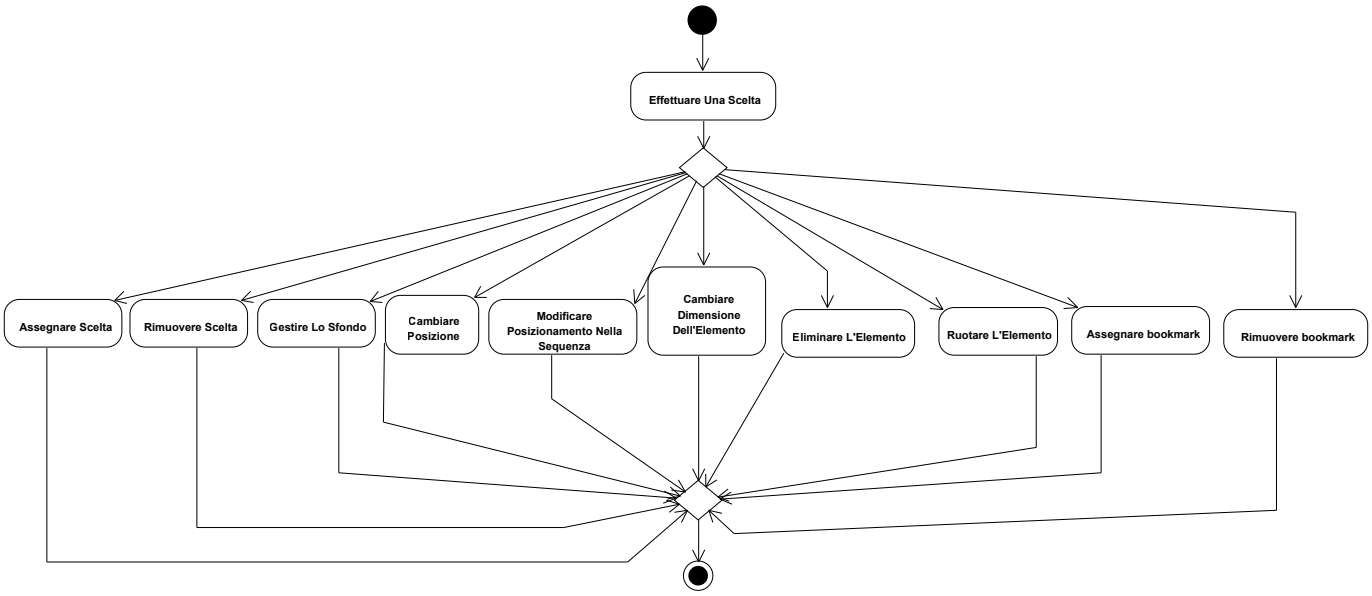

Fig 23: Modificare Frame

6.1.9 Modificare SVG

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

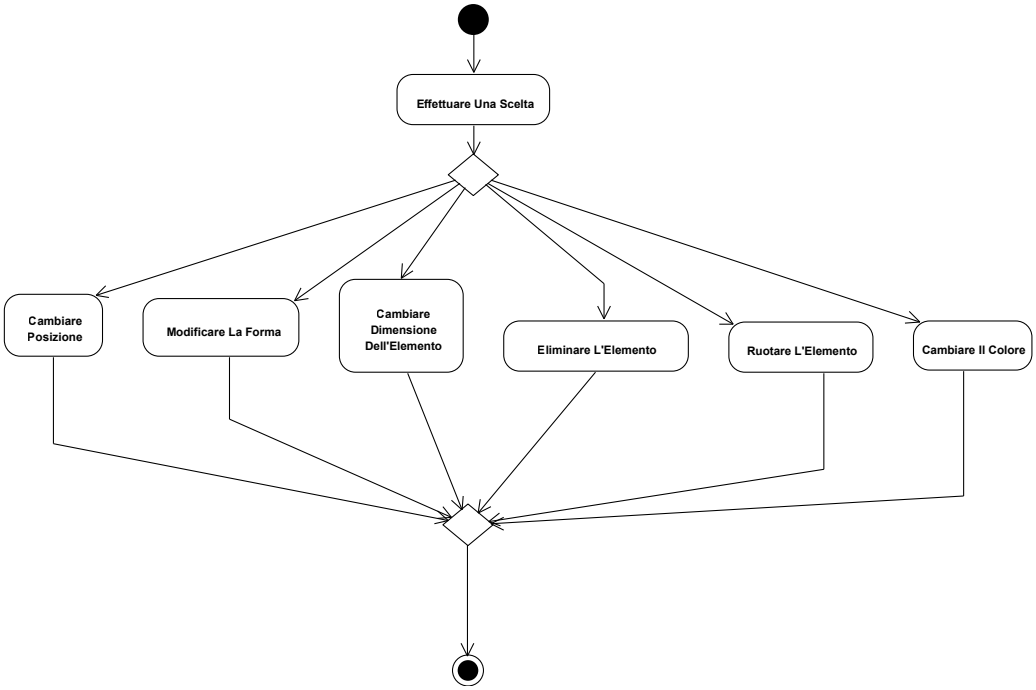

Fig 24: Modificare SVG

6.1.10 Modificare Testo

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

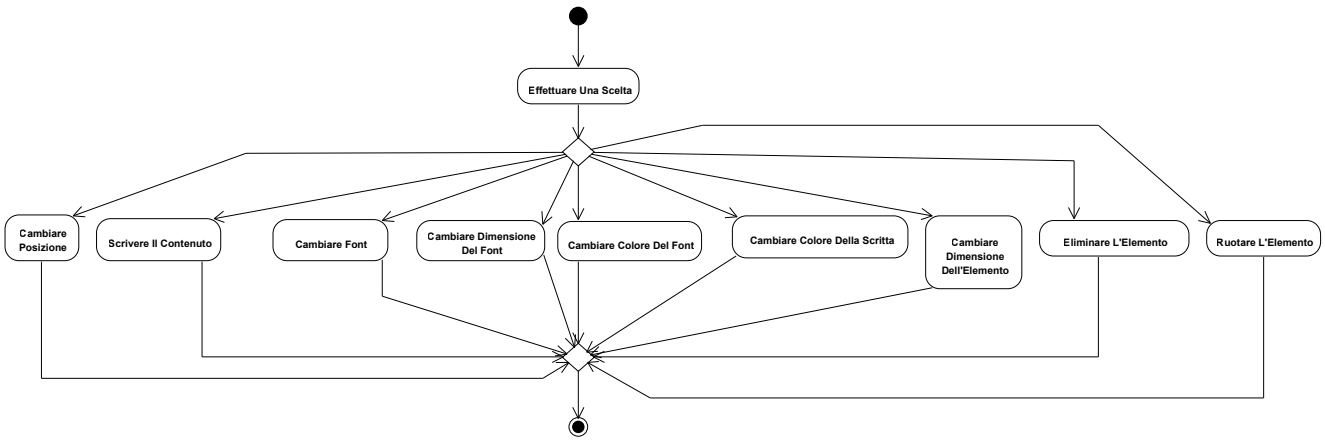

Fig 25: Modificare Testo

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. Per l'esposizione verrà utilizzato un server apache per la esposizione online dell'applicazione e l'utilizzo della tecnologia HTML5 Manifest che semplifica la gestione delle presentazioni offline.

8 Tracciamento dei Componenti coi Requisiti

8.1 Tracciamento Componenti-Requisiti

Tab 3: Tracciamento Componenti-Requisiti

Componente	**Requisiti**
Model	
- >ApacheRelations	
- - >ApacheServerManager	
- - - >FileManager	RF 13, RF 16, RF 17
- - >Manifest	
- - - >GestoreManifest	RF 49
- >ServerRelations	
- - >AccessControl	
- - - >Autenticazione	RF 3, RF 3.1, RF 3.2, RF 64
- - - >Registrazione	RF 1, RF 1.1, RF 1.2
- - >DbConsistency	
- - - >ConcreteObserver	
- - - >Observer	
- - - >Subject	
- - - - >SubjectBackground	
- - - >SubjectAudio	
- - - >SubjectFrame	
- - - >SubjectImg	
- - - >SubjectSVG	
- - - >SubjectText	
- - - >SubjectVideo	
- - >Loader	
- - - >Costruttore	RF 4, RF 7, RF 61
- >SlideShow	
- - >Background	
- - >SlideShowActions	

Componente	Requisiti
- - - >Command	
- - - - >AbstractCommand	
- - - - >ConcreteAudioInsertCommand	RF 7.7.13
- - - - >ConcreteAudioRemoveCommand	RF 7.43
- - - - >ConcreteBackgroundInsertCommand	RF 7.13
- - - - >ConcreteBackgroundRemoveCommand	
- - - - >ConcreteEditBackgroundCommand	RF 7.7.43
- - - - >ConcreteEditColorCommand	RF 7.7.4, RF 7.7.40, RF 7.16, RF 7.40.4
- - - - >ConcreteEditFontCommand	RF 7.7.4
- - - - >ConcreteEditPositionCommand	RF 7.7.19
- - - - >ConcreteEditRotationCommand	RF 7.46, RF 7.7.46
- - - - >ConcreteEditSizeCommand	RF 7.7.10, RF 7.7.16
- - - - >ConcreteFrameInsertCommand	RF 7.1, RF 7.1.1
- - - - >ConcreteFrameRemoveCommand	RF 7.10
- - - - >ConcreteImageInsertCommand	RF 7.7.7
- - - - >ConcreteImageRemoveCommand	RF 7.43
- - - - >ConcreteSVGInsertCommand	RF 7.37
- - - - >ConcreteSVGRemoveCommand	RF 7.43
- - - - >ConcreteTextInsertCommand	RF 7.7.1
- - - - >ConcreteTextRemoveCommand	RF 7.43
- - - - >ConcreteVideoInsertCommand	RF 7.7.13
- - - - >ConcreteVideoRemoveCommand	RF 7.43
- - - - >Invoker	RF 55, RF 58
- - - >InsertEditRemove	
- - - - >Editor	RF 7.7.4, RF 7.7.10, RF 7.7.19, RF 7.7.16, RF 7.7.40, RF 7.7.43, RF 7.16, RF 7.40.4, RF 7.46, RF 7.7.46
- - - - >Inserter	RF 7.1, RF 7.1.1, RF 7.7.1, RF 7.7.7, RF 7.7.13, RF 7.13, RF 7.37

Componente	Requisiti
- - - - >Remover	RF 7.10, RF 7.43
- - >SlideShowElements	
- - - >Audio	
- - - >Frame	
- - - >Image	
- - - >SlideShowElement	
- - - >SVG	
- - - >Text	
- - - >Video	
Controller	
- >EditController	RF 7, RF 7.1, RF 7.1.1, RF 7.4, RF 7.7, RF 7.7.1, RF 7.7.4, RF 7.7.7, RF 7.7.10, RF 7.7.13, RF 7.7.19, RF 7.7.25, RF 7.7.16, RF 7.7.28, RF 7.7.31, RF 7.7.34, RF 7.7.37, RF 7.7.40, RF 7.7.43, RF 7.10, RF 7.16, RF 7.19, RF 7.19.1, RF 7.19.4, RF 7.19.10, RF 7.19.13, RF 7.22, RF 7.25, RF 7.28, RF 7.31, RF 7.34, RF 10.1, RF 10.4, RF 10.5, RF 10.8, RF 55, RF 58, RF 64, RF 7.37, RF 7.40, RF 7.40.1, RF 7.40.4, RF 7.43
- >ExecutionController	RF 61, RF 61.1, RF 61.1.1, RF 61.1.4, RF 61.1.7, RF 61.1.10, RF 61.1.13, RF 61.1.16, RF 61.1.16.1, RF 61.1.16.4, RF 61.1.16.7, RF 61.1.16.10, RF 61.4, RF 61.4.1, RF 61.4.4, RF 61.4.7, RF 61.4.10, RF 61.7, RF 61.10, RF 61.4.10.1, RF 61.4.10.4, RF 61.4.10.7, RF 61.4.10.10
- >HomeController	RF 49, RF 64, RF 34
- >IndexController	RF 1, RF 3, RF 1.1, RF 1.2, RF 3.1, RF 3.2
- >ProfileController	RF 13, RF 64, RF 16, RF 19, RF 17

Componente	Requisiti
View	
- >Pages	
- - >Edit	
- - >EditDesktop	RF 7, RF 7.1, RF 7.1.1, RF 7.4, RF 7.7, RF 7.7.1, RF 7.7.4, RF 7.7.7, RF 7.7.10, RF 7.7.13, RF 7.7.19, RF 7.7.25, RF 7.7.16, RF 7.7.28, RF 7.7.31, RF 7.7.34, RF 7.7.37, RF 7.7.40, RF 7.7.43, RF 7.10, RF 7.16, RF 7.13, RF 7.19, RF 7.19.1, RF 7.19.4, RF 7.19.10, RF 7.19.13, RF 7.22, RF 7.25, RF 7.28, RF 7.31, RF 7.34, RF 7.37, RF 7.40, RF 7.40.1, RF 7.40.4
- - >EditMobile	RF 10, RF 10.1, RF 10.4, RF 10.5, RF 10.8
- - >Execution	RF 61, RF 61.1, RF 61.1.1, RF 61.1.4, RF 61.1.7, RF 61.1.10, RF 61.1.13, RF 61.1.16, RF 61.1.16.1, RF 61.1.16.4, RF 61.1.16.7, RF 61.1.16.10, RF 61.4, RF 61.4.1, RF 61.4.4, RF 61.4.7, RF 61.4.10, RF 61.7, RF 61.10, RF 61.4.10.1, RF 61.4.10.4, RF 61.4.10.7, RF 61.4.10.10
- - >Home	RF 10, RF 49, RF 7, RF 64, RF 19, RF 34
- - >IndexPage	RF 1, RF 3, RF 1.1, RF 1.2, RF 3.1, RF 3.2
- - >Manifest	RF 52, RF 61
- - >Profile	RF 13, RF 43, RF 16, RF 17

8.2 Tracciamento Requisiti-Componenti

Tab 4: Tracciamento Requisiti-Componenti

Requisito	**Componenti**
RF 1	View::Pages::IndexPage, Model::ServerRelations::AccessControl::-Registrazione, Controller::IndexController
RF 1.1	View::Pages::IndexPage, Model::ServerRelations::AccessControl::-Registrazione, Controller::IndexController
RF 1.2	View::Pages::IndexPage, Model::ServerRelations::AccessControl::-Registrazione, Controller::IndexController
RF 3	View::Pages::IndexPage, Model::ServerRelations::AccessControl::-Autenticazione, Controller::IndexController
RF 3.1	View::Pages::IndexPage, Model::ServerRelations::AccessControl::-Autenticazione, Controller::IndexController
RF 3.2	View::Pages::IndexPage, Model::ServerRelations::AccessControl::-Autenticazione, Controller::IndexController
RF 4	Model::ServerRelations::Loader::Costruttore
RF 7	View::Pages::Home, View::Pages::EditDesktop, Model::ServerRelations::Loader::Costruttore, Controller::EditController
RF 7.1	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteFrameInsertCommand, Controller::EditController
RF 7.1.1	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteFrameInsertCommand, Controller::EditController
RF 7.4	View::Pages::EditDesktop, Controller::EditController
RF 7.7	View::Pages::EditDesktop, Controller::EditController
RF 7.7.1	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteTextInsertCommand, Controller::EditController
RF 7.7.4	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, Model::SlideShow::SlideShowActions::Command::-ConcreteEditFontCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.7	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteImageInsertCommand, Controller::EditController

Requisito	Componenti
RF 7.7.10	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditSizeCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.13	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteAudioInsertCommand, Model::SlideShow::SlideShowActions::Command::-ConcreteVideoInsertCommand, Controller::EditController
RF 7.7.16	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditSizeCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.19	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditPositionCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.25	View::Pages::EditDesktop, Controller::EditController
RF 7.7.28	View::Pages::EditDesktop, Controller::EditController
RF 7.7.31	View::Pages::EditDesktop, Controller::EditController
RF 7.7.34	View::Pages::EditDesktop, Controller::EditController
RF 7.7.37	View::Pages::EditDesktop, Controller::EditController
RF 7.7.40	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.43	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditBackgroundCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.46	Model::SlideShow::SlideShowActions::Command::-ConcreteEditRotationCommand, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.10	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Remover, Model::SlideShow::SlideShowActions::Command::-ConcreteFrameRemoveCommand, Controller::EditController
RF 7.13	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteBackgroundInsertCommand
RF 7.16	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.19	View::Pages::EditDesktop, Controller::EditController

Requisito	Componenti
RF 7.19.1	View::Pages::EditDesktop, Controller::EditController
RF 7.19.4	View::Pages::EditDesktop, Controller::EditController
RF 7.19.10	View::Pages::EditDesktop, Controller::EditController
RF 7.19.13	View::Pages::EditDesktop, Controller::EditController
RF 7.22	View::Pages::EditDesktop, Controller::EditController
RF 7.25	View::Pages::EditDesktop, Controller::EditController
RF 7.28	View::Pages::EditDesktop, Controller::EditController
RF 7.31	View::Pages::EditDesktop, Controller::EditController
RF 7.34	View::Pages::EditDesktop, Controller::EditController
RF 7.37	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, Model::SlideShow::SlideShowActions::Command::-ConcreteSVGInsertCommand, Controller::EditController
RF 7.40	View::Pages::EditDesktop, Controller::EditController
RF 7.40.1	View::Pages::EditDesktop, Controller::EditController
RF 7.40.4	View::Pages::EditDesktop, Model::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, Controller::EditController, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.43	Model::SlideShow::SlideShowActions::InsertEditRemove::-Remover, Model::SlideShow::SlideShowActions::Command::-ConcreteTextRemoveCommand, Model::SlideShow::SlideShowActions::Command::-ConcreteImageRemoveCommand, Model::SlideShow::SlideShowActions::Command::-ConcreteSVGRemoveCommand, Model::SlideShow::SlideShowActions::Command::-ConcreteAudioRemoveCommand, Model::SlideShow::SlideShowActions::Command::-ConcreteVideoRemoveCommand, Controller::EditController
RF 7.46	Model::SlideShow::SlideShowActions::Command::-ConcreteEditRotationCommand, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 10	View::Pages::Home, View::Pages::EditMobile
RF 10.1	View::Pages::EditMobile, Controller::EditController
RF 10.4	View::Pages::EditMobile, Controller::EditController
RF 10.5	View::Pages::EditMobile, Controller::EditController
RF 10.8	View::Pages::EditMobile, Controller::EditController
RF 12	
RF 13	View::Pages::Profile, Controller::ProfileController, Model::ApacheRelations::ApacheServerManager::FileManager

Requisito	Componenti
RF 16	View::Pages::Profile, Controller::ProfileController, Model::ApacheRelations::ApacheServerManager::FileManager
RF 17	View::Pages::Profile, Controller::ProfileController, Model::ApacheRelations::ApacheServerManager::FileManager
RF 19	View::Pages::Home, Controller::ProfileController
RF 25	
RF 31	
RF 34	View::Pages::Home, Controller::HomeController
RF 35	
RF 36	
RF 37	
RF 43	View::Pages::Profile
RF 46	
RF 49	View::Pages::Home, Controller::HomeController, Model::ApacheRelations::Manifest::GestoreManifest
RF 52	View::Pages::Manifest
RF 55	Controller::EditController, Model::SlideShow::SlideShowActions::Command::-Invoker
RF 58	Controller::EditController, Model::SlideShow::SlideShowActions::Command::-Invoker
RF 61	View::Pages::Manifest, View::Pages::Execution, Model::ServerRelations::Loader::Costruttore, Controller::ExecutionController
RF 61.1	View::Pages::Execution, Controller::ExecutionController
RF 61.1.1	View::Pages::Execution, Controller::ExecutionController
RF 61.1.4	View::Pages::Execution, Controller::ExecutionController
RF 61.1.7	View::Pages::Execution, Controller::ExecutionController
RF 61.1.10	View::Pages::Execution, Controller::ExecutionController
RF 61.1.13	View::Pages::Execution, Controller::ExecutionController
RF 61.1.16	View::Pages::Execution, Controller::ExecutionController
RF 61.1.16.1	View::Pages::Execution, Controller::ExecutionController
RF 61.1.16.4	View::Pages::Execution, Controller::ExecutionController

Requisito	Componenti
RF 61.1.16.7	View::Pages::Execution, Controller::ExecutionController
RF 61.1.16.10	View::Pages::Execution, Controller::ExecutionController
RF 61.4	View::Pages::Execution, Controller::ExecutionController
RF 61.4.1	View::Pages::Execution, Controller::ExecutionController
RF 61.4.4	View::Pages::Execution, Controller::ExecutionController
RF 61.4.7	View::Pages::Execution, Controller::ExecutionController
RF 61.4.10	View::Pages::Execution, Controller::ExecutionController
RF 61.4.10.1	View::Pages::Execution, Controller::ExecutionController
RF 61.4.10.4	View::Pages::Execution, Controller::ExecutionController
RF 61.4.10.7	View::Pages::Execution, Controller::ExecutionController
RF 61.4.10.10	View::Pages::Execution, Controller::ExecutionController
RF 61.7	View::Pages::Execution, Controller::ExecutionController
RF 61.10	View::Pages::Execution, Controller::ExecutionController
RF 64	View::Pages::Home, Model::ServerRelations::AccessControl::-Autenticazione, Controller::EditController, Controller::HomeController, Controller::ProfileController
RF 67	
RF 67.1	
RF 67.4	
RF 67.7	
RF 67.10	
RF 67.13	
RF 70	
RF 70.1	
RF 70.4	
RF 70.5	
RF 70.10	

Requisito	Componenti
RF 70.10.1	
RF 70.10.1.1	
RF 70.10.1.4	
RF 70.10.1.4.1	
RF 70.10.1.4.4	
RF 70.10.1.4.7	
RF 70.10.1.4.10	
RF 70.10.1.4.13	
RF 70.10.1.7	
RF 70.10.4	
RF 70.10.7	
RF 70.10.10	
RF 70.10.13	
RF 70.10.16	
RF 70.10.19	
RF 70.13	
RF 70.19	
RF 73	