19-08-2015

Specifica Tecnica

Informazioni sul documento

Nome Documento	Specifica Tecnica
Versione	2.4.0
Stato	*Formale*
Uso	*Esterno*
Data Creazione	18-05-2015
Data Ultima Modifica	19-08-2015
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
2.5.0	Gabelli Pietro	19-08-2015	Rimozione componenti di ApacheServer
2.4.0	Tollot Pietro	02-07-2015	Modifica schema backEndProgettazione
2.3.0	Tollot Pietro	27-06-2015	Aggiornamento schemi di Authenitcation, Loader, Register, accessControll, fileServerRelation, mongoRelation, nodeAPI, serverRelation; modifica capitolo Model::MongoRelations
2.2.0	Fossa Manuel	22-06-2015	Aggiornamento schema View; aggiornamento capitolo View::Pages
2.1.0	Gabelli Pietro	17-06-2015	Aggiornamento contenuti: architettura da MVP a MVC e Controller
2.0.0	Venturelli Giovanni	16-06-2015	Approvazione Documento
1.3.0	Gabelli Pietro	16-06-2015	Eseguite correzioni automatiche
1.2.0	Busetto Matteo	10-06-2015	Aggionamento capitolo Stime di fattibilità e di bisogno risorse
1.1.0	Fossa Manuel	09-06-2015	Aggiornamento capitolo Premi::View
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: server Node e View
0.3.0	Petrucci Mauro	14-05-2015	Aggiunta dei contenuti: Slideshowelements e InsertEditRemove
0.2.0	Fossa Manuel	12-05-2015	Aggiunta dei contenuti: Stumenti e Design Pattern

Versione	Autore	Data	Descrizione
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

RP -> RQ

Versione 3.0.0	**Nominativo**
Redazione	Busetto Matteo, Tollot Pietro, Petrucci Mauro
Verifica	Venturelli Giovanni
Approvazione	Busetto Matteo

Tab 3: Storico ruoli RP -> RQ

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.3.0.0.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.3.0.0.pdf;
- Analisi dei Requisiti: AnalisiDeiRequisiti_v.3.0.0.pdf;
- Piano di qualifica: PianoDiQualifica_v.3.0.0.pdf;
- Piano di progetto: PianoDiProgetto_v.3.0.0.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

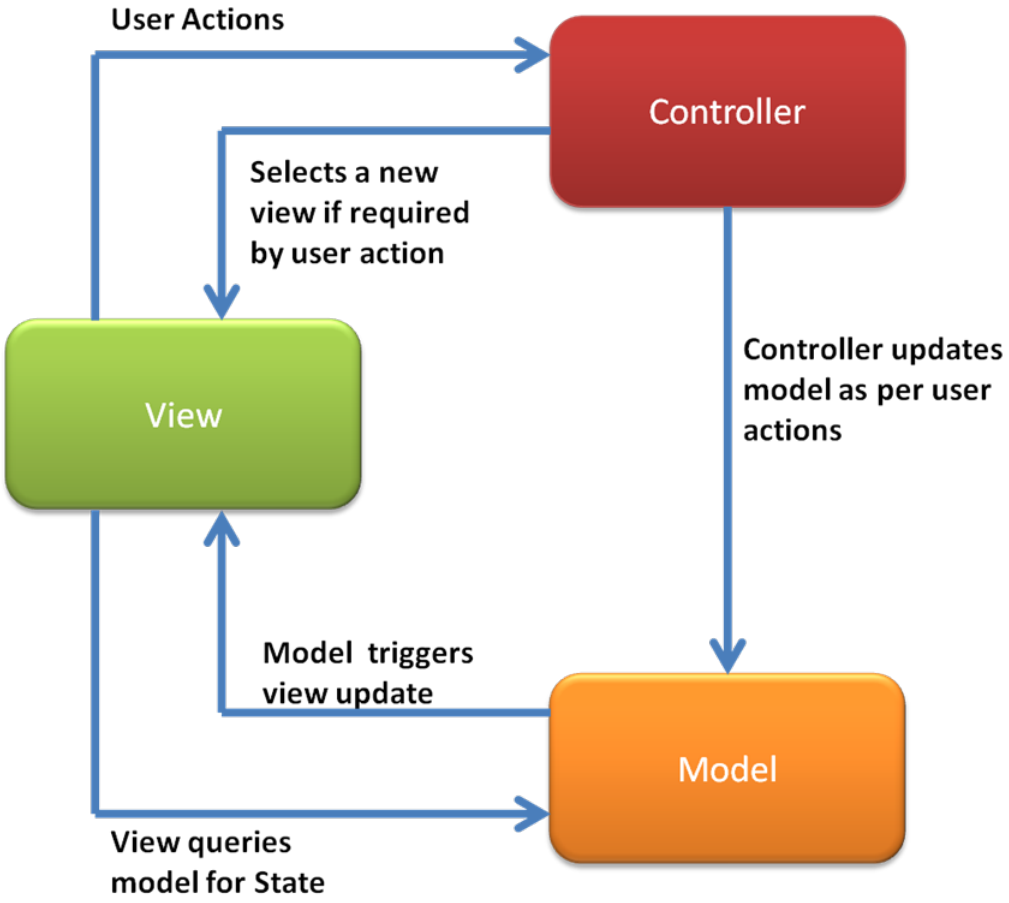

Fig 1: Model View Controller

- **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

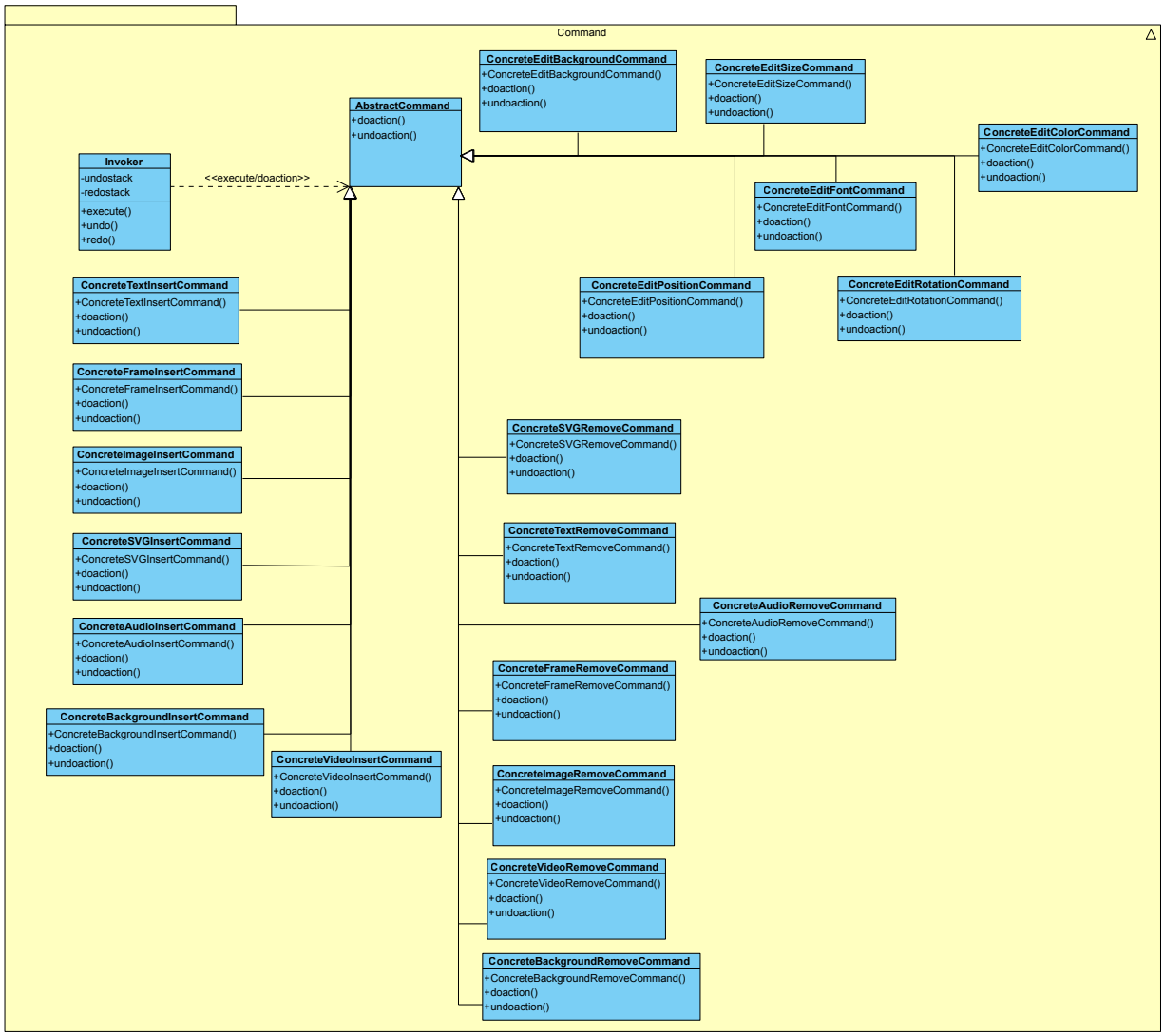

Fig 2: Diagramma delle classi del package Command

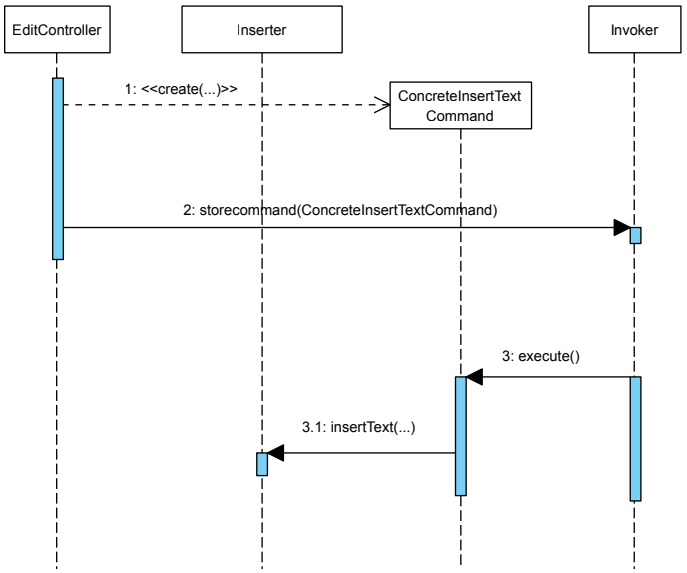

Fig 3: diagramma di sequenza del Pattern Command

- **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.

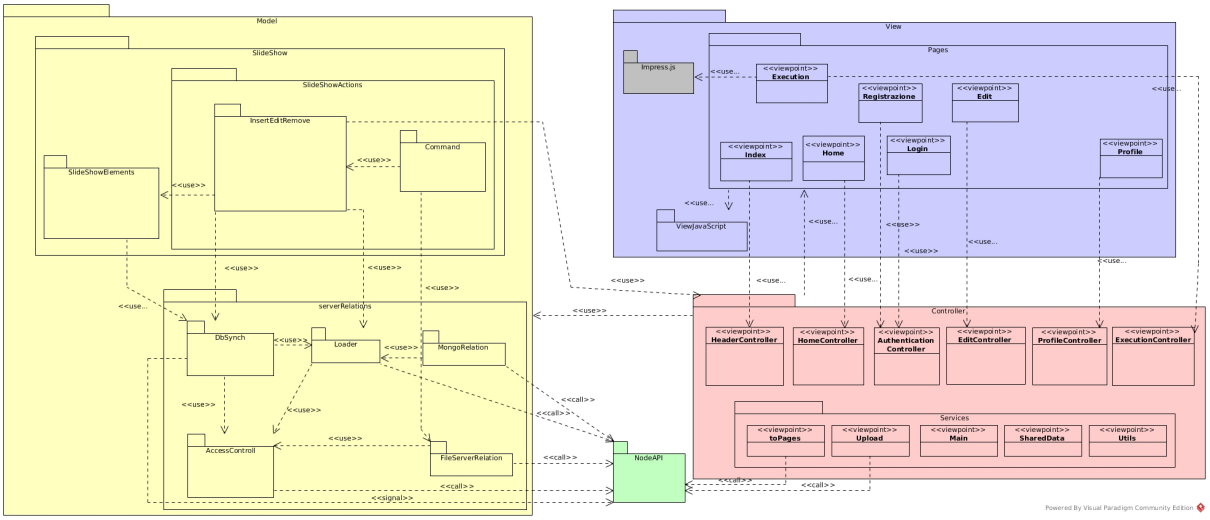

Fig 4: Architettura generale del sistema

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.

4.3 Servizi Api nodeAPI

Il seguente diagramma delle classi è stato esteso con le primitive:

- **«Resource»** : rappresenta una risorsa associata ad un certo url a cui sono disponibili dei servizi
- **«Node»** : rappresenta una parte di url a cui non sono disponibili servizi ma è utile per suddividere quest'ultimi
- **«Server»** : rappresenta la radice dei servizi offerti dal server
- **«Path»** : indica una aggiunta in coda all' url attuale per raggiungere una nuova risorsa o nodo
- **«Middleware»** : indica un middleware, un insieme di funzionalità chiamate ogni qualvolta si accede a risorse attraversando questo elemento

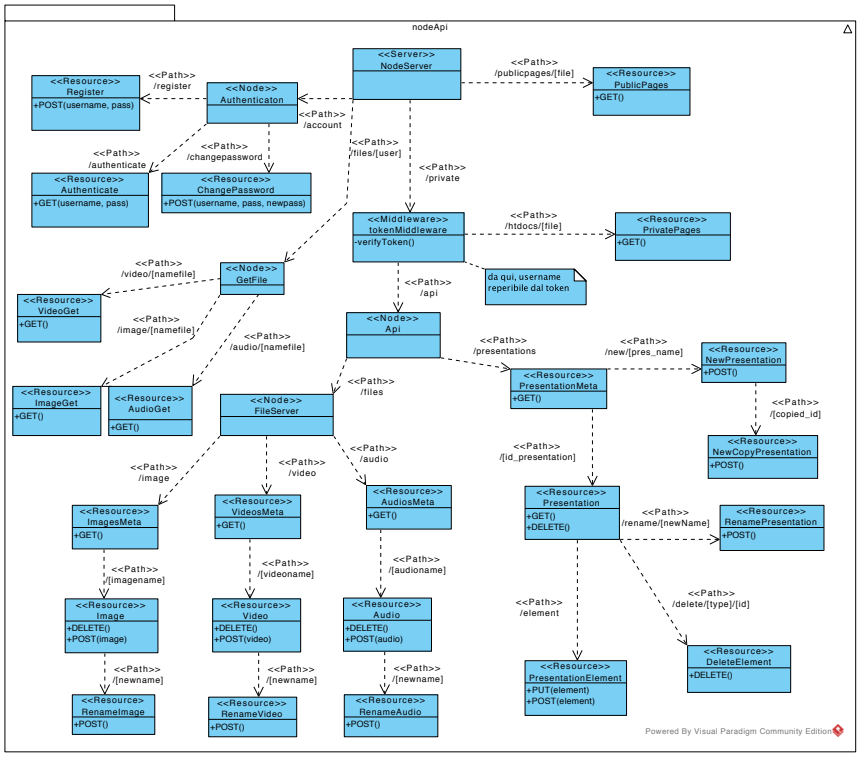

Fig 5: Servizio Api nodeApI

- **NodeServer:** radice dei servizi offerti dal server:
  1. server per pagine html e file statici associati
  2. servizi di autenticazione stateless
  3. servizi di upload e reperimento file statici multimediali per utente
  4. servizi di interazione con MongoDB per salvataggio persistente delle presentazioni
- **Register:**
  - **POST** /account/register
    - **descrizione:** inserisce nuovo utente in MongoDB, crea una nuova collezione 'presentations'+username, crea le cartelle per i file utente
- **Authenticate:**
  - **GET** /account/authenticate
    - **descrizione:** verifica se username e password sono corretti e ritorna un token per l'accesso ai servizi protetti
- **ChangePassword:**
  - **POST** /account/changepassword
    - **descrizione:** verifica la correttezza di username e password e modifica questa'ultima con la nuova

- **PublicPages:**
  - **GET** /publicpages/[file]
    - **descrizione:** se presente [file] nella cartella /public_html del server ritorna il file stesso
- **tokenMiddleware:** verifica che il token passato nel campo Authorization dell' Header sia valido, dal token ricava lo username dell'utente
- **PrivatePages:**
  - **GET** /private/htdocs/[file]
    - **descrizione:** se presente [file] nella cartella /private_html del server ritorna il file stesso
- **PresentationMeta:**
  - **GET** /private/api/presentations
    - **descrizione:** cerca in mongoDB nella collezione associata alle presentazioni dell'utente, ritorna un array i cui elementi sono i campi meta delle presentazioni dell'utente
- **NewPresentation:**
  - **POST** /private/api/presentations/new/[presentationName]
    - **descrizione:** se non esiste già crea una nuova presentazione con il nome [presentatioNname]
- **NewCopyPresentation:**
  - **POST** /private/api/presentations/new/[newPresentationName]/[oldPresentationName]
    - **descrizione:** crea una nuova presentazione con nome [newPresentationName] dalla presentazione con titolo [oldPresentationName]
- **Presentation:**
  - **GET** /private/api/presentations/[presentationName]
    - **descrizione:** recupera se presente la presentazione dell'utente associata al titolo passato nell'url
  - **DELETE** /private/api/presentations/[presentationName]
    - **descrizione:** elimina se presente la presentazione dell'utente associata al titolo passato nell'url
- **RenamePresentation:**
  - **POST** /private/api/presentations/[presentationName]/rename/[newname]
    - **descrizione:** rinomina se presente la presentazione dell'utente associata al titolo passato nell'url con il nome [newname]

- **PresentationElement:**
  - **POST** /private/api/presentations/[presentationName]/element
    - **descrizione:** inserisce nella presentazione dell'utente individuata da [presentationName] l'oggetto element passato nel body della richiesta
  - **PUT** /private/api/presentations/[presentationName]/element
    - **descrizione:** sostituisce nella presentazione dell'utente l'elemento passato nel body della richiesta
- **DeleteElement:**
  - **DELETE** /private/api/presentations/[presentationName]/delete/[type/[id_element]]
    - **descrizione:** elimina dalla presentazione con il titolo [presentationName] l'elemento con identificativo [idElement]
- **GetImage:**
  - **GET** /files/[user]/image/[imagename]
    - **descrizione:** ritorna il file [imagename] nella cartella /users/[username]/image
- **GetAudio:**
  - **GET** /files/[user]/audio/[audioname]
    - **descrizione:** ritorna il file [audioname] nella cartella /users/[username]/audios
- **GetVideo:**
  - **GET** /files/[user]/video/[videoname]
    - **descrizione:** ritorna il file [videoname] nella cartella /users/[username]/videos
- **ImagesMeta:**
  - **GET** /private/api/files/image
    - **descrizione:** ritorna un array con i nomi dei file immagine dell'utente
- **Image:**
  - **POST** /private/api/files/image/[imagename]
    - **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/images
  - **DELETE** /private/api/files/image/[imagename]
    - **descrizione:** elimina il file immagine [imagename] dalla cartella /users/[username]/images
- **RenameImage:**
  - **POST** /private/api/files/image/[imagename]/[newname]
    - **descrizione:** rinomina il file immagine [imagename] con [newname] nella cartella /users/[username]/images

- **AudiosMeta:**
  - **GET** /private/api/files/audio
    - **descrizione:** ritorna un array con i nomi dei file audio dell'utente
- **Audio:**
  - **POST** /private/api/files/audio/[audioname]
    - **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/audios
  - **DELETE** /private/api/files/audio/[audioname]
    - **descrizione:** elimina il file audio [audioname] dalla cartella /users/[username]/audios
- **RenameAudio:**
  - **POST** /private/api/files/audio/[audioname]/[newname]
    - **descrizione:** rinomina il file audio [audioname] con [newname] nella cartella /users/[username]/audios
- **VideosMeta:**
  - **GET** /private/api/files/video
    - **descrizione:** ritorna un array con i nomi dei file video dell'utente
- **Video:**
  - **POST** /private/api/files/video/[videoname]
    - **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/videos
  - **DELETE** /private/api/files/video/[videoname]
    - **descrizione:** elimina il file video [videoname] dalla cartella /users/[username]/videos
- **RenameVideo:**
  - **POST** /private/api/files/video/[videoname]/[newname]
    - **descrizione:** rinomina il file video [videoname] con [newname] nella cartella /users/[username]/videos

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>.

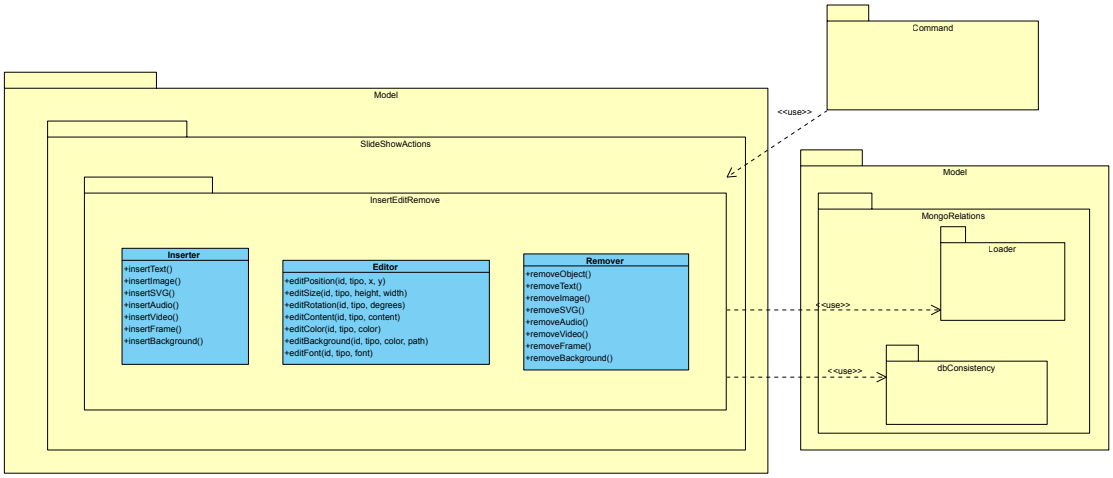

Fig 6: InsertEditRemove

Tipo, obiettivo e funzione del componente: all'interno di questo Package sono implementate le classi statiche destinate all'inserimento, alla rimozione e alla modifica degli elementi della presentazione.
Relazioni d'uso di altre componenti:il package è in relazione con Model::SlideShow::SlideShowActions::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>.

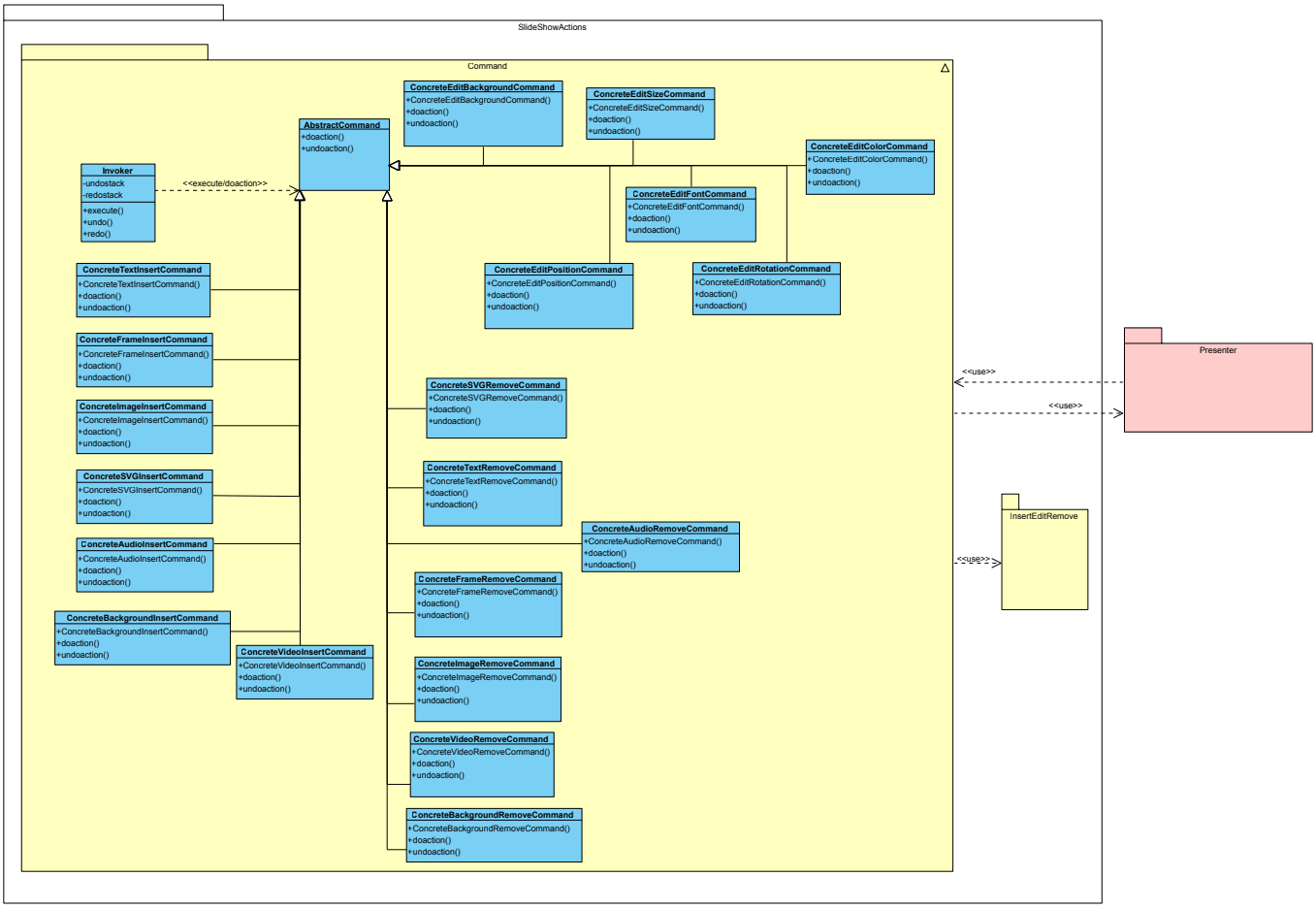

Fig 7: Command Package

Tipo, obiettivo e funzione del componente:All'interno di questo Package viene implementato il Design Pattern command, utile per la gestione di funzioni di annullamento e ripristino.
Relazioni d'uso di altre componenti:. All'interno del Model, il package è in relazione con

- Model::SlideShow::SlideShowActions::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>.

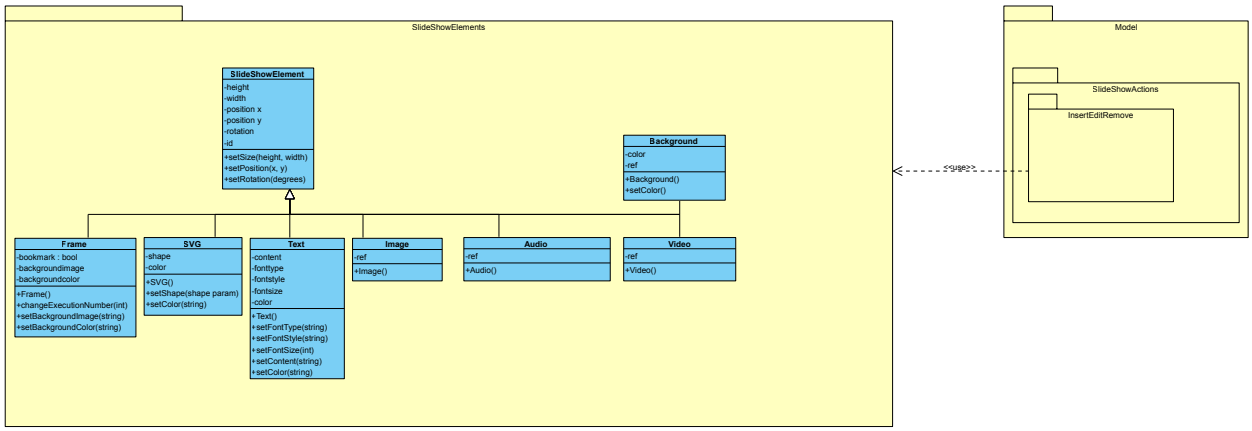

Fig 8: SlideShowElements

Tipo, obiettivo e funzione del componente:Di questo package fanno parte le classi degli elementi della presentazione e la classe che definisce la presentazione stessa.
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::serverRelations

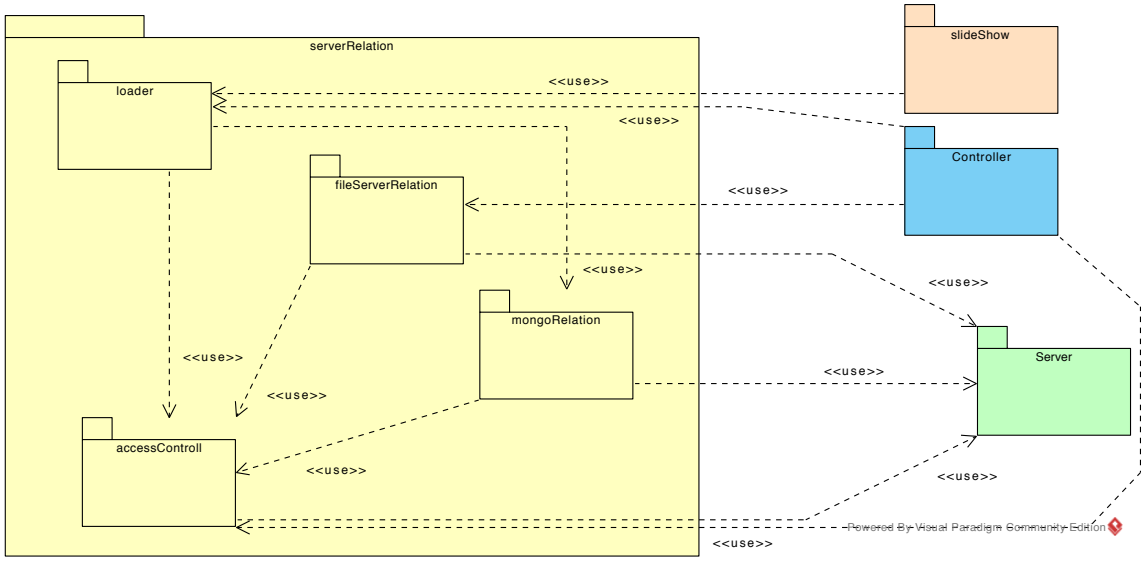

Fig 9: diagramma package Model::serverRelations

Tipo, obiettivo e funzione del componente: package, racchiude le funzionalità del sistema che interagiscono con i servizi offerti dal server nodeJs per l'interazione con la base dati MongoDB e la gestione dei file multimediali in cloud
Relazioni d'uso di altre componenti:

- relazioni verso **Server** del quale si utilizzano i servizi RESTfull;
- relazioni da **Controller** per il recupero o la creazione di una nuova presentazione dal database MongoDB al caricamento delle pagine HTML;
- relazioni da **Model::SlideShow** che utilizza la rappresentazione locale della presentazione.

5.1.8 Model::serverRelations::accessControl

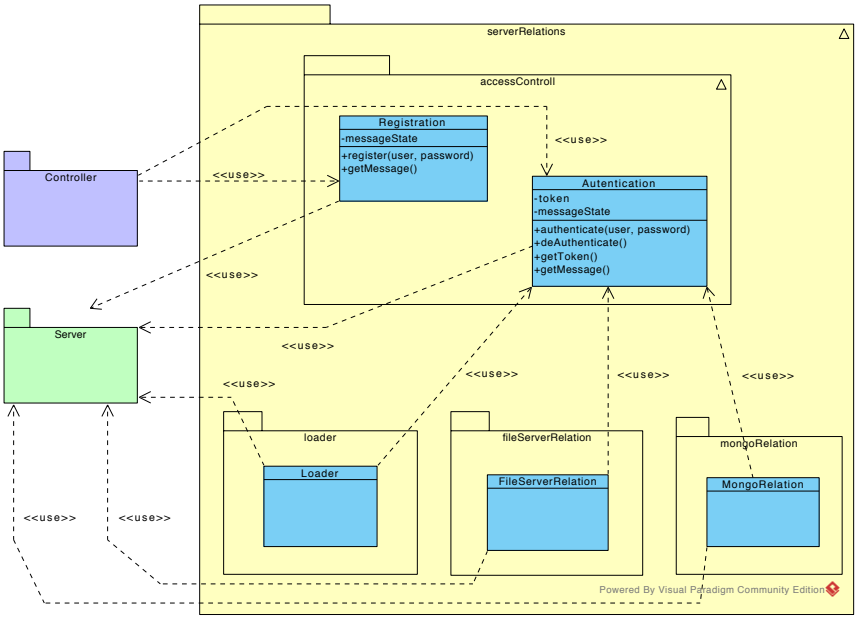

Fig 10: accessControl

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 **Server** 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 **loader, fileServerRelation, mongoRelation** a cui viene esposto il token ricevuto dal Server dopo la autenticazione

5.1.8.1 Autenticazione

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

- relazione verso **Server** 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 **loader, fileServerRelation, mongoRelation** a cui espone il token per poter usare i servizi del Server

5.1.8.2 Registrazione

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

- relazione verso **Server** 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

5.1.9 Model::serverRelations:loader

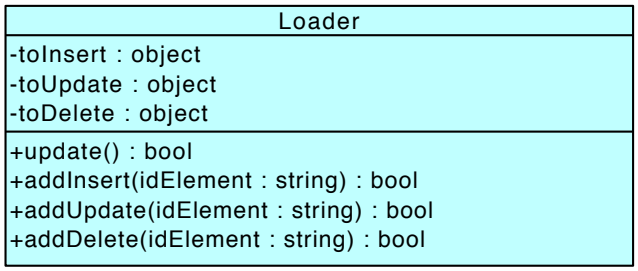

Fig 11: MongoRelations::Loader

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

- relazione verso **Server** 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

5.1.9.1 Loader

Tipo, obiettivo e funzione del componente: Classe la cui funzione è esporre una interfaccia per la sincronizzazione delle modifiche della presentazione nel model verso il server
Relazioni d'uso di altre componenti:

- relazione verso **Server** 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

5.1.9.2 FileServerRelation

Tipo, obiettivo e funzione del componente: Classe che si interfaccia con il server per l'upload, la gestione e il recupero di informazioni dei file multimediali presenti nello spazio dell'utente **Relazioni d'uso di altre componenti**:

- dipendenza verso **Server** per recupero informazioni sui file e upload e gestione di nuovi file verso nello spazio utente
- dipendenza verso **accessControll** per il recupero del token per accedere ai servizi protetti del Server
- dipendenza da **Controller** da cui vengono chiamati i metodi esposti

5.1.9.3 MongoRelation

Tipo, obiettivo e funzione del componente: Classe che si interfaccia con il server per la gestione delle presentazioni salvate in formato json su un database MongoDB **Relazioni d'uso di altre componenti**:

- dipendenza verso **Server** per l'interazione con il database MongoDB
- dipendenza verso **accessControll** per il recupero del token per accedere ai servizi protetti del Server
- dipendenza da **Loader** da cui vengono chiamati i metodi esposti

5.2 View

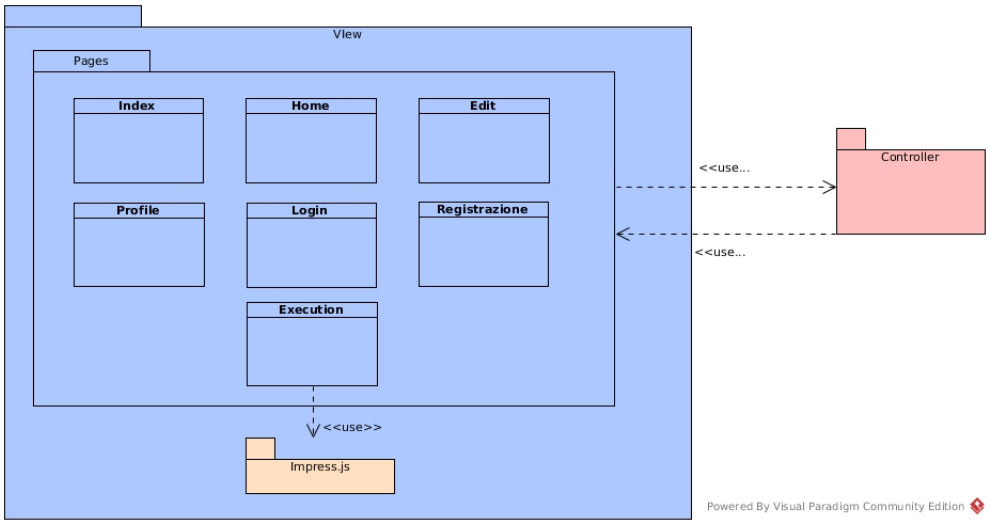

Fig 12: View

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 provenienti dalla pagina View::Pages::Edit.
Relazioni d'uso di altre componenti:

- Tutte le seguenti classi, appartenenti al package Model::SlideShow::SlideShowActions::-Command:
  - Invoker <- EditController costruisce l'oggetto Invoker, gli passa un oggetto di classe Command eseguendo e annullando tale comando;
  - ConcreteTextInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteFrameInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteImageInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteSVGInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteAudioInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteVideoInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteBackgroundInsertCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteTextRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteFrameRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteImageRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteSVGRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteAudioRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteVideoRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;

  - ConcreteBackgroundRemoveCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditSizeCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditPositionCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditRotationCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditColorCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditBackgroundCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditFontCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;
  - ConcreteEditContentCommand <- EditController costruisce un comando e lo dà in pasto a Invoker;

- Controller::Services::Upload <- EditController richiama Upload quando è necessario caricare nel server un file media;
- Controller::Services::SharedData -> EditController ricava la presentazione corrente da SharedData;
- Model::serverRelation::Loader <- EditController, ad ogni modifica della presentazione, richiama i metodi appropriati di Loader in modo tale da permettere il salvataggio della presentazione stessa nel server;
- View::Pages::Edit::[javascript_functions] <- EditController invoca le appropriate funzioni Javascript per applicare le modifiche alla presentazione sulla pagina di Edit.

Interfacce con e relazioni d'uso e da altre componenti: EditController richiama le funzioni javascript fornite da View::Pages::Edit per la modifica della view. Successivamente istanzia un oggetto di una sottoclasse di Command e lo dà in pasto a Invoker e successivamente richiama il metodo corretto di Loader per il salvataggio nel database. Nel caso di un annullamento di una modifica o di un suo ripristino, EditController richiama il metodo undo() (o redo()) di Invoker il quale a sua volta, richiama il metodo corretto di EditController per l'aggiornamento della view.

5.3.2 Controller::ExecutionController

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di gestire i segnali provenienti della pagina View::Pages::Execution.
Relazioni d'uso di altre componenti:

- Controller::Services::SharedData -> ExecutionController ricava la presentazione corrente da SharedData;

- Controller::Services::toPages <- Quando la view invia una richiesta di reindirizzamento alla pagina View::Pages::Home o View::Pages::Edit, HeaderController invoca il metodo appropriato di toPages.

Interfacce con e relazioni d'uso e da altre componenti: la view invia a ExecutionController una richiesta di reindirizzamento ad una pagina oppure per ricavare la presentazione corrente. ExecutionController richiama il metodo appropriato di toPages, se la richiesta è un reindirizzamento, oppure di SharedData.

5.3.3 Controller::HeaderController

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

- Controller::Services::Main <- Quando la view invia una richiesta di logout, HeaderController invoca il metodo per la deautenticazione fornito da Main;
- Controller::Services::toPages <- Quando la view invia una richiesta di reindirizzamento alla pagina View::Pages::Home o View::Pages::Profile, HeaderController invoca il metodo appropriato di toPages.

Interfacce con e relazioni d'uso e da altre componenti: la view invia a HeaderController una richiesta di logout o di reindirizzamento ad una pagina. HeaderController richiama il metodo per il logout di Main, oppure, se è un reindirizzamento, richiama il metodo appropriato di toPages.

5.3.4 Controller::AuthenticationController

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di gestire i segnali e le chiamate provenienti dalle pagine View::Pages::Login e View::Pages::Registrazione.
Relazioni d'uso di altre componenti:

- Controller::Services::Main <- Quando la view invia una richiesta di autenticazione, logout o registrazione, HeaderController invoca il metodo corretto fornito da Main;

Interfacce con e relazioni d'uso e da altre componenti: la view invia a AuthenticationController una richiesta di registrazione o autenticazione. AuthenticationController richiama il metodo appropriato di Main.

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

- Controller::Services::Main <- Quando la view invia una richiesta di cambio della password, viene invocato il metodo per il cambio della password di Main.

Interfacce con e relazioni d'uso e da altre componenti: la pagina Profile invia a ProfileController la richiesta di cambio password. ProfileController richiama il metodo appropriato di Main.

5.3.6 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:

- Model::serverRelation::mongoRelation <- HomeController invoca i metodi necessari per il recupero di tutte le presentazioni dell'utente, la creazione di una nuova, la rinominazione o la cancellazione di una presentazione.

Interfacce con e relazioni d'uso e da altre componenti: la pagina Home invia a HomeController una richiesta. HomeController, in base al tipo di richiesta (creazione nuova presentazione, rinominazione, eliminazione o ottenimento della lista delle presentazioni) richiama il metodo appropriato di mongoRelation per soddisfarla.

5.3.7 Controller::Services

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di gestire le principali funzioni dell'applicazione, a partire dall'autenticazione fino ad arrivare all'upload dei file nel server.
Relazioni d'uso di altre componenti: comunica con il Model per svolgere le operazioni necessarie.

5.3.7.1 Services::toPages

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di gestire i reindirizzamenti alle pagine corrette.
Relazioni d'uso di altre componenti:

- /private <- toPages invia una richiesta http al server, il quale controlla l'esistenza del token per le pagine in cui è richiesta l'autenticazione.

Interfacce con e relazioni d'uso e da altre componenti: toPages invia una richiesta http al server per il reindirizzamento alla pagina corretta. Nel caso in cui la pagina richieda di essere autenticati, viene inviato anche il token di sessione per verificare l'effettiva autenticazione.

5.3.7.2 Services::Upload

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di permettere l'upload dei file media nel server.
Relazioni d'uso di altre componenti:

- /private/api/files/image/[filename] <- Upload invia una richiesta http al server per effettuare l'upload del file immagine filename;
- /private/api/files/video/[filename] <- Upload invia una richiesta http al server per effettuare l'upload del file video filename;
- /private/api/files/audio/[filename] <- Upload invia una richiesta http al server per effettuare l'upload del file audio filename.

Interfacce con e relazioni d'uso e da altre componenti: Upload invia una richiesta http al server per il caricamento di un file media nel server, inviando anche il token di sessione per verificare l'effettiva autenticazione.

5.3.7.3 Services::Main

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di permettere le funzioni di base dell'applicazione, tra cui l'autenticazione al server.
Relazioni d'uso di altre componenti: comunica con Model::serverRelation::accessControl per l'autenticazione, la registrazione e il cambio della password.

- Model::serverRelation::accessControl::Authentication <- Main richiama Authentication per inviare una richiesta di autenticazione o di logout al server;
- Model::serverRelation::accessControl::Registration <- Main richiama Registration per inviare una richiesta di registrazione di un nuovo utente al server;
- Model::serverRelation::accessControl::ChangePassword <- Main richiama ChangePassword per inviare una richiesta di cambio password al server.

Interfacce con e relazioni d'uso e da altre componenti: Main richiama il metodo corretto di accessControl in modo da inviare una richiesta http al server per effettuare l'autenticazione, la registrazione o il cambio della password.

5.3.7.4 Services::SharedData

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è di mantenere in memoria la presentazione corrente.
Relazioni d'uso di altre componenti: comunica con Model::serverRelation::mongoRelation per ricavare la presentazione su cui si sta lavorando dal server.

- Model::serverRelation::mongoRelation <- SharedData richiama mongoRelation per ottenere la presentazione corrente.

Interfacce con e relazioni d'uso e da altre componenti: SharedData richiama il metodo corretto di mongoRelation in modo da inviare una richiesta http al server per ottentere la presentazione voluta.

5.3.7.5 Services::Utils

Tipo, obiettivo e funzione del componente: lo scopo di questa classe è definire delle funzioni utili a tutta l'applicazione.
Relazioni d'uso di altre componenti: data la sua natura, non comunica con nessun package.

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.

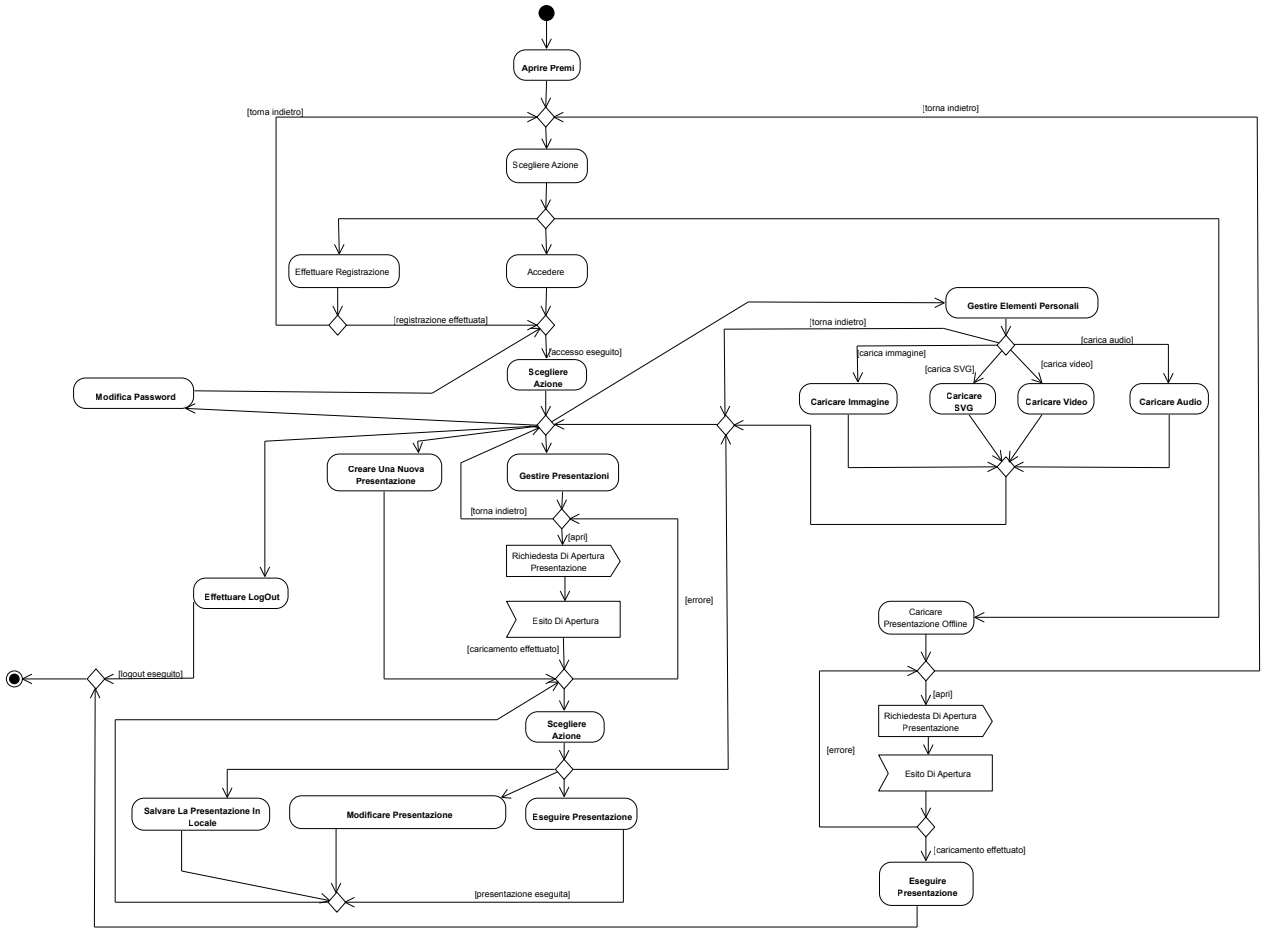

Fig 13: Attività Principali

6.1.1 Gestione presentazioni

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

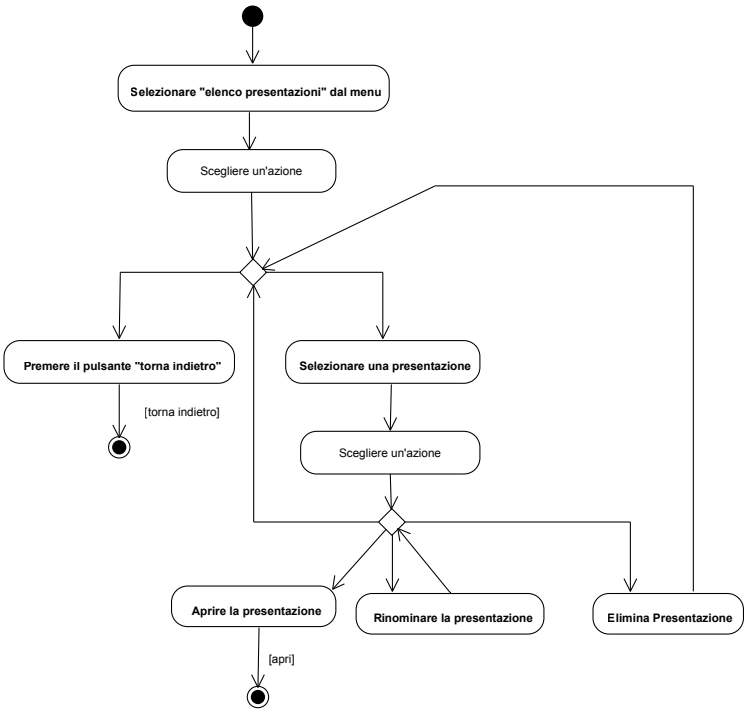

Fig 14: Gestione Presentazioni

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.

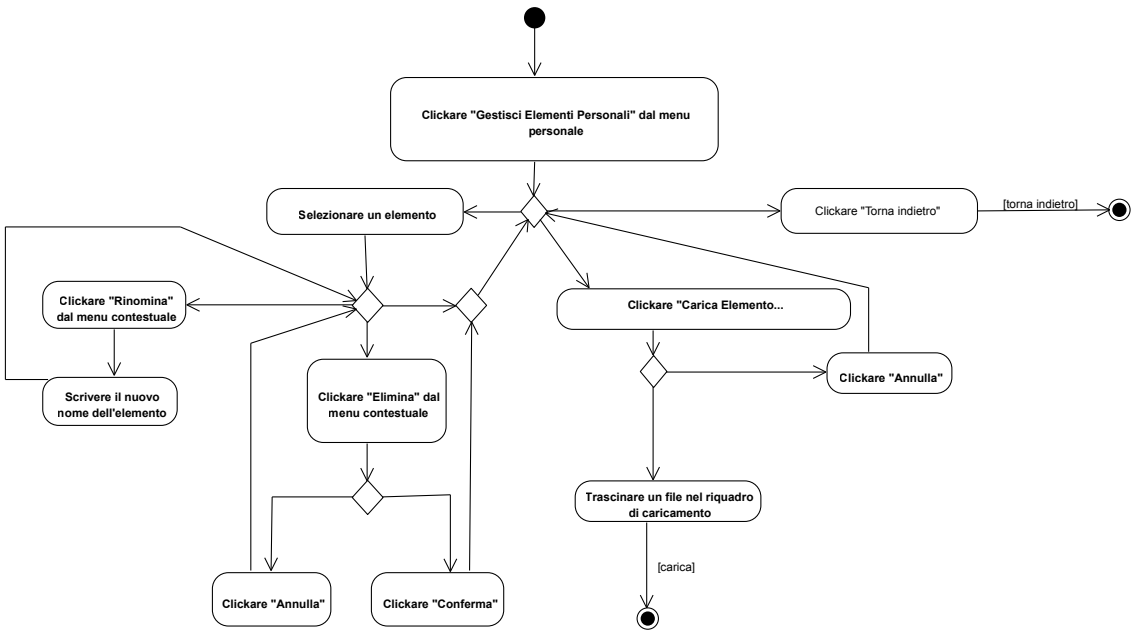

Fig 15: Caricare File

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.

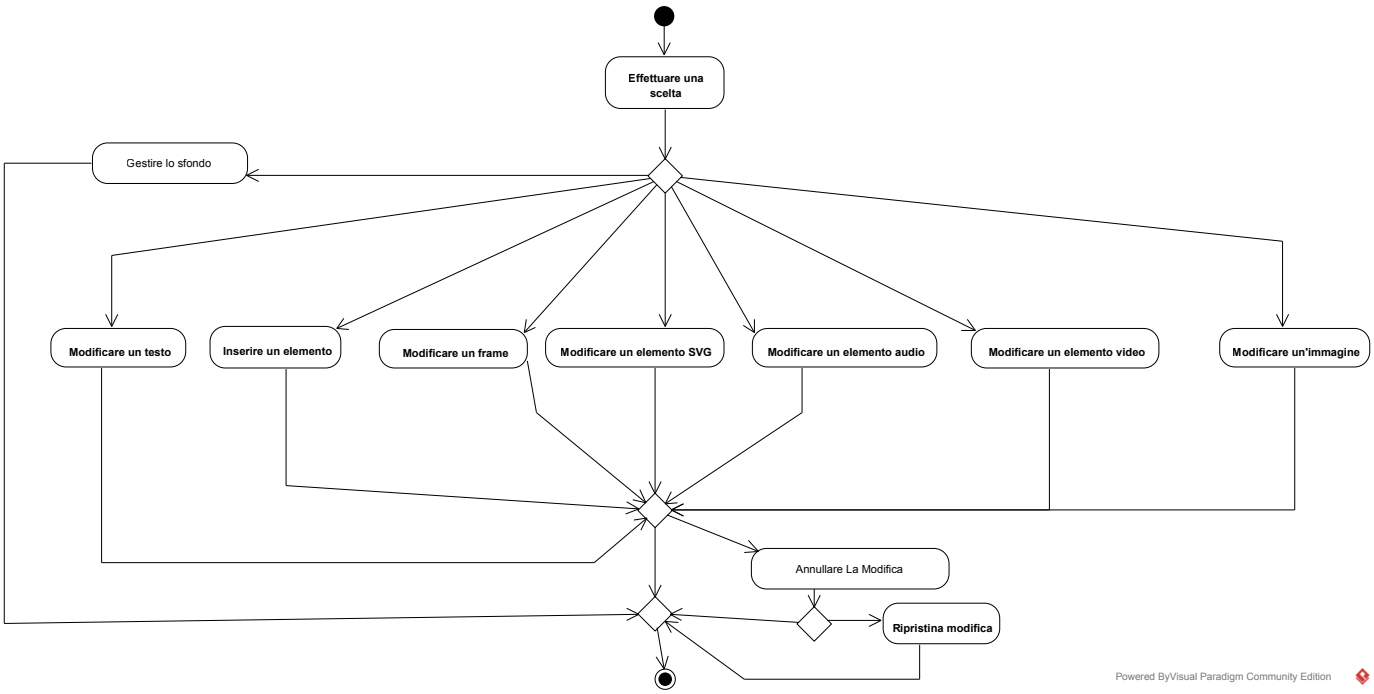

Fig 16: Modificare Presentazione da Desktop

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.

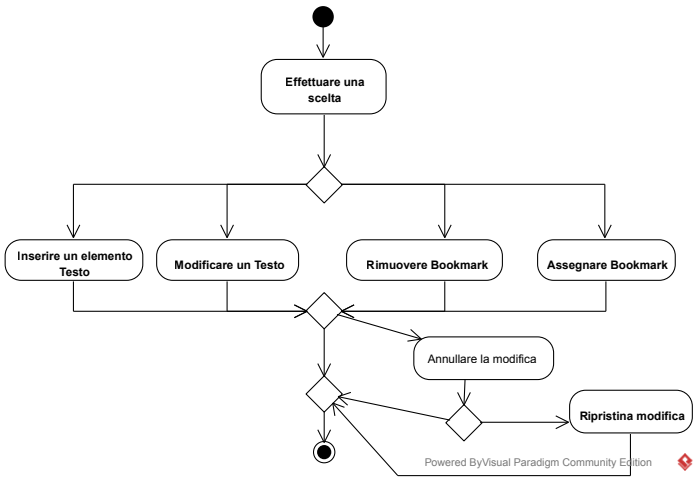

Fig 17: Modificare Presentazione da Mobile

6.1.5 Gestire Sfondo

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

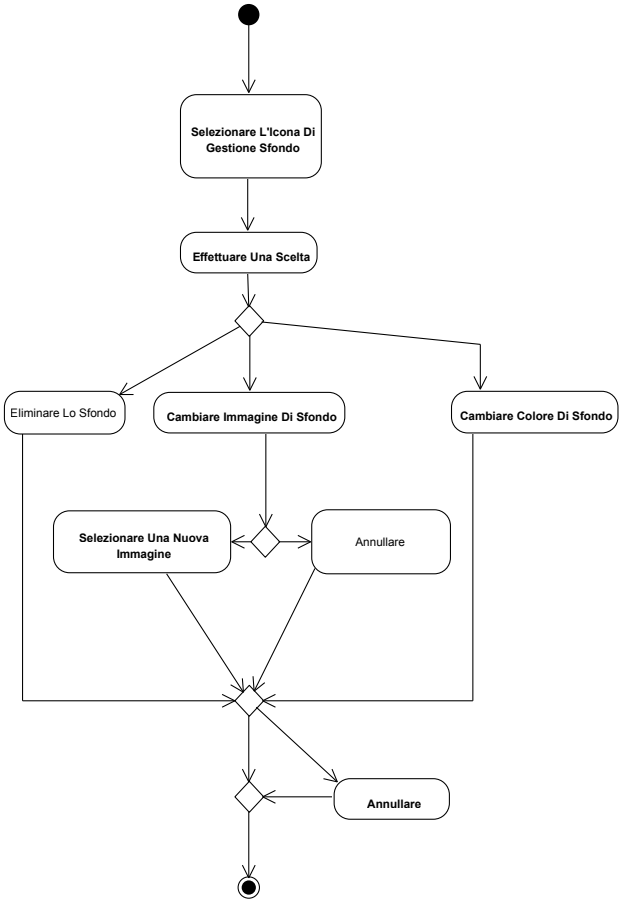

Fig 18: Gestire Sfondo

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.

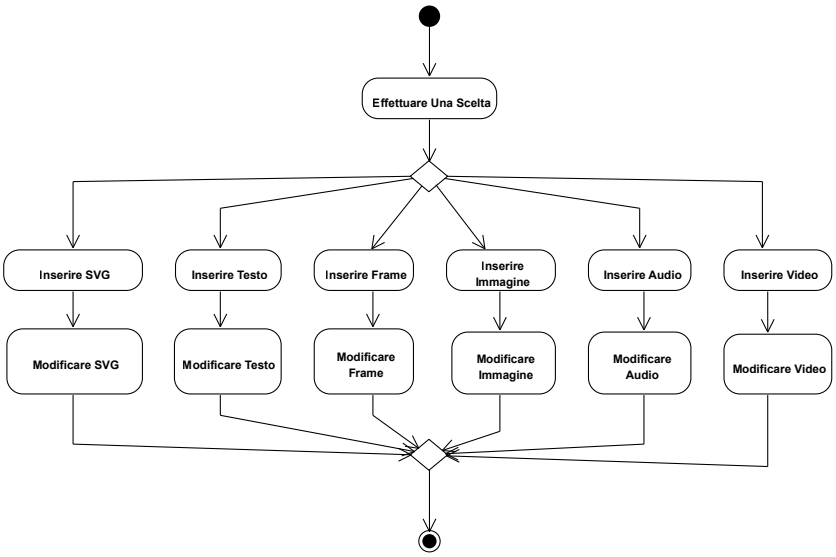

Fig 19: Inserire Elemento

6.1.7 Modificare Elemento

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

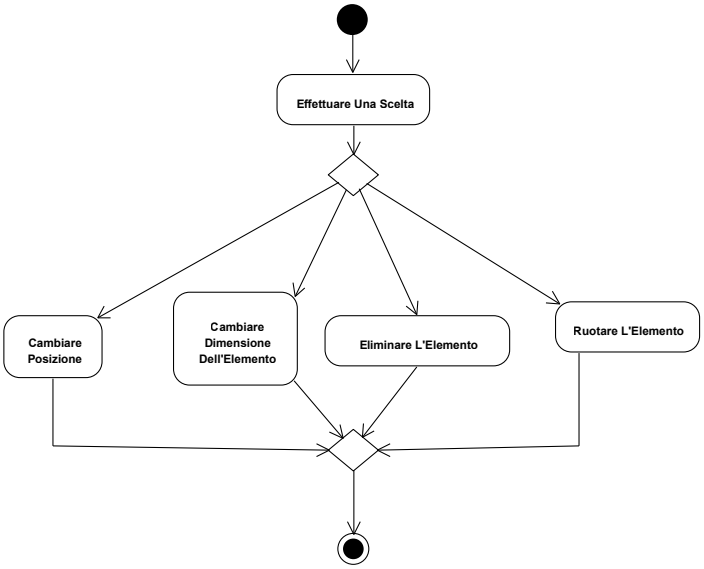

Fig 20: Modificare Elemento

6.1.8 Modificare Frame

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

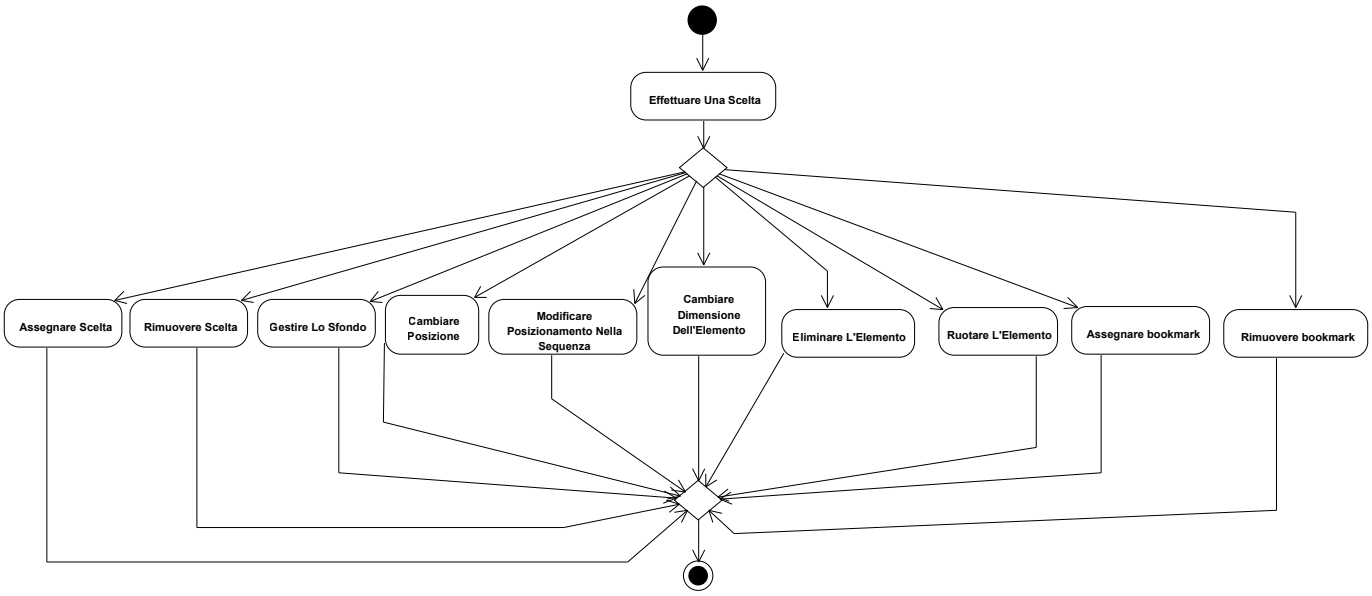

Fig 21: Modificare Frame

6.1.9 Modificare SVG

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

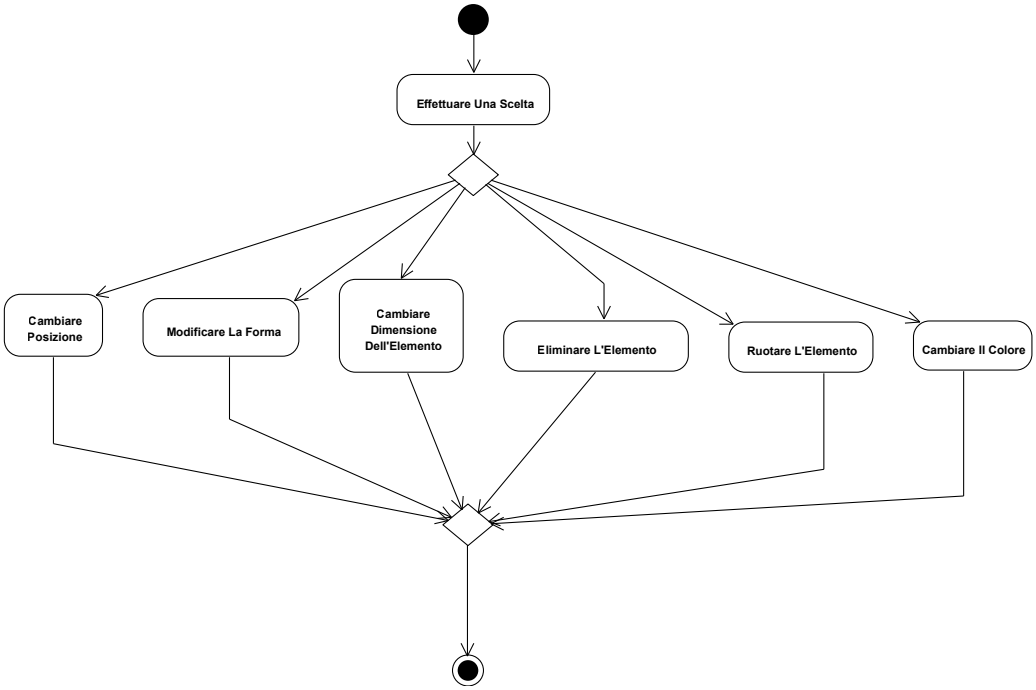

Fig 22: Modificare SVG

6.1.10 Modificare Testo

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

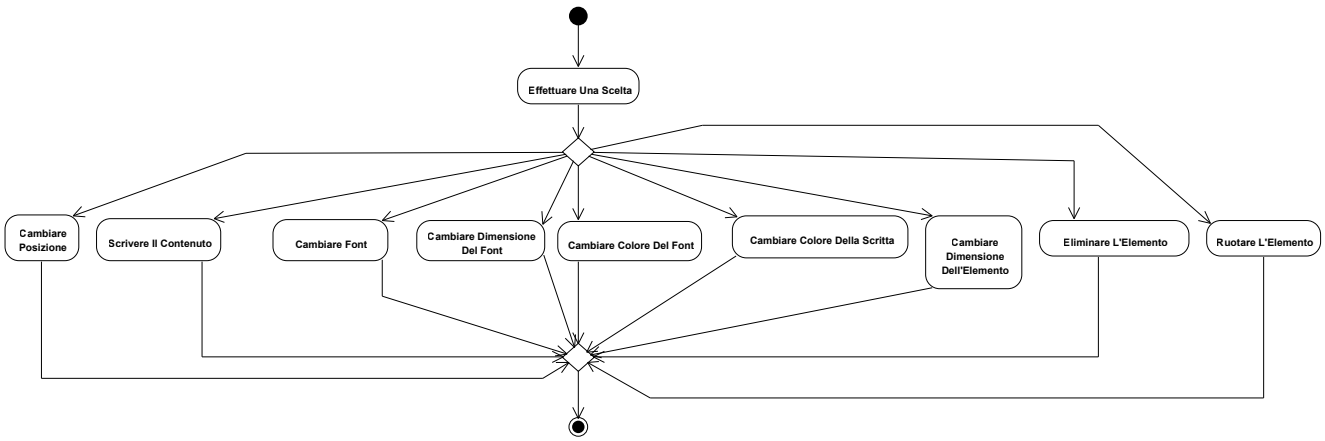

Fig 23: Modificare Testo

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 verranno utilizzati Javascript e il framework Angular.js per lo spostamento in tempo reale degli elementi delle presentazioni. Infine è stato inoltre considerato l'utilizzo della tecnologia HTML5 Manifest per la gestione delle presentazioni offline.

8 Tracciamento dei Componenti coi Requisiti

8.1 Tracciamento Componenti-Requisiti

Tab 4: Tracciamento Componenti-Requisiti

Componente	**Requisiti**
Controller	
- >AccessController	
- >EditController	
- >ExecutionController	
- >HeaderController	
- >HomeController	
- >ProfileController	
- >Services	
- - >Main	
- - >SharedData	
- - >toPages	
- - >Upload	
- - >Utils	
Model	
- >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	

Componente	Requisiti
- - - >SubjectSVG	
- - - >SubjectText	
- - - >SubjectVideo	
- - >Loader	
- - - >Costruttore	RF 4, RF 7, RF 61
- >SlideShow	
- - >Background	
- - >SlideShowActions	
- - - >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

Componente	Requisiti
- - - - >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
- - - - >Remover	RF 7.10, RF 7.43
- - >SlideShowElements	
- - - >Audio	
- - - >Frame	
- - - >Image	
- - - >SlideShowElement	
- - - >SVG	
- - - >Text	
- - - >Video	
View	
- >Pages	
- - >Edit	
- - >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

Componente	**Requisiti**
- - >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 5: Tracciamento Requisiti-Componenti

Requisito	**Componenti**
RF 1	::Pages::IndexPage, ::ServerRelations::AccessControl::Registrazione
RF 1.1	::Pages::IndexPage, ::ServerRelations::AccessControl::Registrazione
RF 1.2	::Pages::IndexPage, ::ServerRelations::AccessControl::Registrazione
RF 3	::Pages::IndexPage, ::ServerRelations::AccessControl::Autenticazione
RF 3.1	::Pages::IndexPage, ::ServerRelations::AccessControl::Autenticazione
RF 3.2	::Pages::IndexPage, ::ServerRelations::AccessControl::Autenticazione
RF 4	::ServerRelations::Loader::Costruttore
RF 7	::Pages::Home, ::ServerRelations::Loader::Costruttore
RF 7.1	::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, ::SlideShow::SlideShowActions::Command::-ConcreteFrameInsertCommand
RF 7.1.1	::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, ::SlideShow::SlideShowActions::Command::-ConcreteFrameInsertCommand
RF 7.4	
RF 7.7	
RF 7.7.1	::SlideShow::SlideShowActions::InsertEditRemove::Inserter, ::-SlideShow::SlideShowActions::Command::ConcreteTextInsertCommand
RF 7.7.4	::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, ::SlideShow::-SlideShowActions::Command::ConcreteEditFontCommand, ::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.7	::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, ::SlideShow::SlideShowActions::Command::-ConcreteImageInsertCommand
RF 7.7.10	::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand, ::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.13	::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, ::SlideShow::SlideShowActions::Command::-ConcreteAudioInsertCommand, ::SlideShow::SlideShowActions::-Command::ConcreteVideoInsertCommand
RF 7.7.16	::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand, ::SlideShow::SlideShowActions::InsertEditRemove::Editor

Requisito	Componenti
RF 7.7.19	::SlideShow::SlideShowActions::Command::-ConcreteEditPositionCommand, ::SlideShow::SlideShowActions::-InsertEditRemove::Editor
RF 7.7.25	
RF 7.7.28	
RF 7.7.31	
RF 7.7.34	
RF 7.7.37	
RF 7.7.40	::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, ::SlideShow::SlideShowActions::-InsertEditRemove::Editor
RF 7.7.43	::SlideShow::SlideShowActions::Command::-ConcreteEditBackgroundCommand, ::SlideShow::SlideShowActions::-InsertEditRemove::Editor
RF 7.7.46	::SlideShow::SlideShowActions::Command::-ConcreteEditRotationCommand, ::SlideShow::SlideShowActions::-InsertEditRemove::Editor
RF 7.10	::SlideShow::SlideShowActions::InsertEditRemove::-Remover, ::SlideShow::SlideShowActions::Command::-ConcreteFrameRemoveCommand
RF 7.13	::SlideShow::SlideShowActions::InsertEditRemove::-Inserter, ::SlideShow::SlideShowActions::Command::-ConcreteBackgroundInsertCommand
RF 7.16	::SlideShow::SlideShowActions::Command::-ConcreteEditColorCommand, ::SlideShow::SlideShowActions::-InsertEditRemove::Editor
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	

Requisito	Componenti
RF 7.34	
RF 7.37	::SlideShow::SlideShowActions::InsertEditRemove::Inserter, ::SlideShow::SlideShowActions::Command::ConcreteSVGInsertCommand
RF 7.40	
RF 7.40.1	
RF 7.40.4	::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand, ::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.43	::SlideShow::SlideShowActions::InsertEditRemove::Remover, ::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand, ::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand, ::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand, ::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand, ::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand
RF 7.46	::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand, ::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 10	::Pages::Home
RF 10.1	
RF 10.4	
RF 10.5	
RF 10.8	
RF 12	
RF 13	::Pages::Profile
RF 16	::Pages::Profile
RF 17	::Pages::Profile
RF 19	::Pages::Home
RF 25	
RF 31	
RF 34	::Pages::Home
RF 35	
RF 36	
RF 37	

Requisito	Componenti
RF 43	::Pages::Profile
RF 46	
RF 49	::Pages::Home
RF 52	::Pages::Manifest
RF 55	::SlideShow::SlideShowActions::Command::Invoker
RF 58	::SlideShow::SlideShowActions::Command::Invoker
RF 61	::Pages::Manifest, ::Pages::Execution, ::ServerRelations::Loader::-Costruttore
RF 61.1	::Pages::Execution
RF 61.1.1	::Pages::Execution
RF 61.1.4	::Pages::Execution
RF 61.1.7	::Pages::Execution
RF 61.1.10	::Pages::Execution
RF 61.1.13	::Pages::Execution
RF 61.1.16	::Pages::Execution
RF 61.1.16.1	::Pages::Execution
RF 61.1.16.4	::Pages::Execution
RF 61.1.16.7	::Pages::Execution
RF 61.1.16.10	::Pages::Execution
RF 61.4	::Pages::Execution
RF 61.4.1	::Pages::Execution
RF 61.4.4	::Pages::Execution
RF 61.4.7	::Pages::Execution
RF 61.4.10	::Pages::Execution
RF 61.4.10.1	::Pages::Execution
RF 61.4.10.4	::Pages::Execution

Requisito	Componenti
RF 61.4.10.7	::Pages::Execution
RF 61.4.10.10	::Pages::Execution
RF 61.7	::Pages::Execution
RF 61.10	::Pages::Execution
RF 64	::Pages::Home, ::ServerRelations::AccessControl::Autenticazione
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	
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	

Requisito	Componenti
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	