

19-08-2015



Specifica Tecnica

Informazioni sul documento

Nome Documento	Specifica Tecnica
Versione	2.4.0
Stato	<i>Formale</i>
Uso	<i>Esterno</i>
Data Creazione	18-05-2015
Data Ultima Modifica	19-08-2015
Redazione	Fossa Manuel, Petrucci Mauro
Approvazione	Tollot Pietro
Verifica	Gabelli Pietro
Lista distribuzione	<i>LateButSafe</i> 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 Authentication, 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	Aggiornamento 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: Strumenti e Design Pattern


$$\mathbf{RR} \rightarrow \mathbf{RP}$$

Tab 2: Storico ruoli RR \rightarrow RP

Tab 3: Storico ruoli RP \rightarrow RQ

Indice

1	Introduzione	9
1.1	Scopo del documento	9
1.2	Scopo del Prodotto	9
1.3	Glossario	9
1.4	Riferimenti	9
1.4.1	Normativi	9
1.4.2	Informativi	9
2	Strumenti	11
2.1	HTML	11
2.2	JavaScript	11
2.3	jQuery	11
2.4	MEAN	12
2.4.1	MongoDB	12
2.4.2	Express.js	12
2.4.3	AngularJS	12
2.4.4	Node.js	12
2.5	Impress.js	12
3	Design Pattern e Pattern Architeturali	13
3.1	MVC	13
3.2	Command	14
3.2.1	Premi::Model::SlideShow::SlideShowActions::Command	15
4	Descrizione architeturale	17
4.1	Metodo e formalismi	17
4.2	Architettura generale	17
4.2.1	Model	18
4.2.2	View	18
4.2.3	Controller	18
4.3	Servizi Api nodeAPI	18
5	Descrizione dei singoli componenti	23
5.1	Model	23
5.1.1	Model::SlideShow	23
5.1.2	Model::SlideShow::SlideShowActions	23
5.1.3	Model::SlideShow::SlideShowActions::InsertEditRemove	23
5.1.3.1	Editor	24
5.1.3.2	Inserter	26
5.1.3.3	Remover	27
5.1.4	Model::SlideShow::SlideShowActions::Command	27
5.1.4.1	Invoker	28
5.1.4.2	AbstractCommand	29
5.1.4.3	ConcreteTextInsertCommand	30



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



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



1	Model View Controller	13
2	Diagramma delle classi del package Command	14
3	diagramma di sequenza del Pattern Command	15
4	Architettura generale del sistema	17
5	Servizio Api nodeApi	19
6	InsertEditRemove	24
7	Command Package	28
8	SlideShowElements	46
9	diagramma package Model::serverRelations	51
10	accessControl	52
11	serverRelation::Loader	53
12	View	55
13	Attività Principali	64
14	Gestione Presentazioni	65
15	Caricare File	65
16	Modificare Presentazione da Desktop	66
17	Modificare Presentazione da Mobile	66
18	Gestire Sfondo	67
19	Inserire Elemento	68
20	Modificare Elemento	68
21	Modificare Frame	69
22	Modificare SVG	69
23	Modificare Testo	70

1	Versionamento del documento	1
2	Storico ruoli RR -> RP	3
3	Storico ruoli RP -> RQ	3
4	Tracciamento Componenti-Requisiti	70
5	Tracciamento Requisiti-Componenti	74

Sommario

Il presente documento contiene la specifica tecnica delle componenti che costituiscono il prodotto software Premi.



1.1 Scopo del documento

1.2 Scopo del Prodotto

1.3 Glossario

1.4 Riferimenti

1.4.1 Normativi

- ### 1.4.2 Informativi

- Università degli studi di Padova - 2014/2015

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



2.1 HTML

- **Vantaggi:**

- **Svantaggi:**

- ## 2.2 JavaScript

2.3 jQuery

Il nucleo di jQuery è una libreria di manipolazione DOM (Document Object Model). DOM è una struttura ad albero che rappresenta tutti gli elementi di una pagina web e jQuery rende la ricerca, selezione e manipolazione di questi elementi DOM semplice e conveniente. I vantaggi nell'uso di jQuery sono l'incoraggiamento alla separazione di Javascript ed HTML, la brevità e la chiarezza, l'eliminazione di incompatibilità cross-browser, l'estendibilità.

2.4 MEAN

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

2.4.1 MongoDB

MongoDB è un database NoSQL open source orientato ai documenti, facilmente scalabile e ad alte prestazioni. Si allontana dalla struttura tradizionale basata su tabelle dei database relazionali, in favore di documenti in stile JSON con schema dinamico; 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, mantenuto da Google e da una comunità di sviluppatori e corporations. Mira a semplificare lo sviluppo ed il test di applicazioni single-page fornendo un framework per l'architettura model-view-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 Architeturali

3.1 MVC

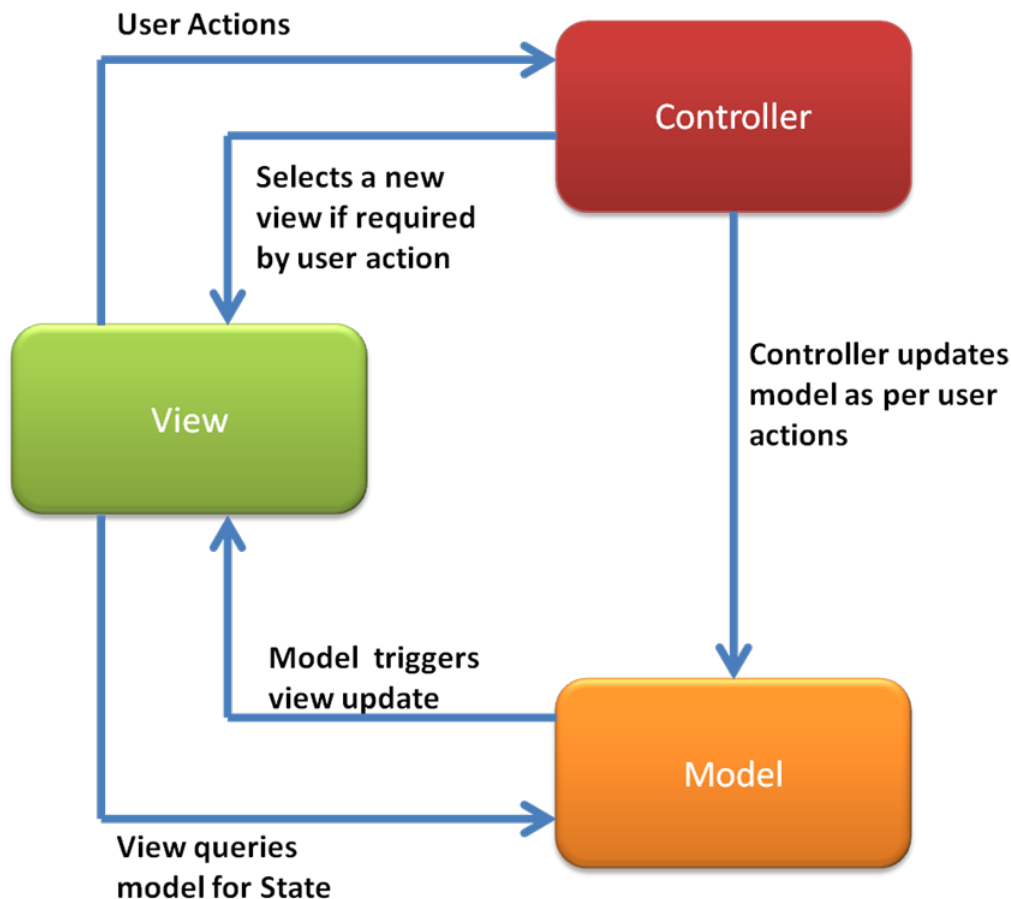


Fig 1: Model View Controller

- **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

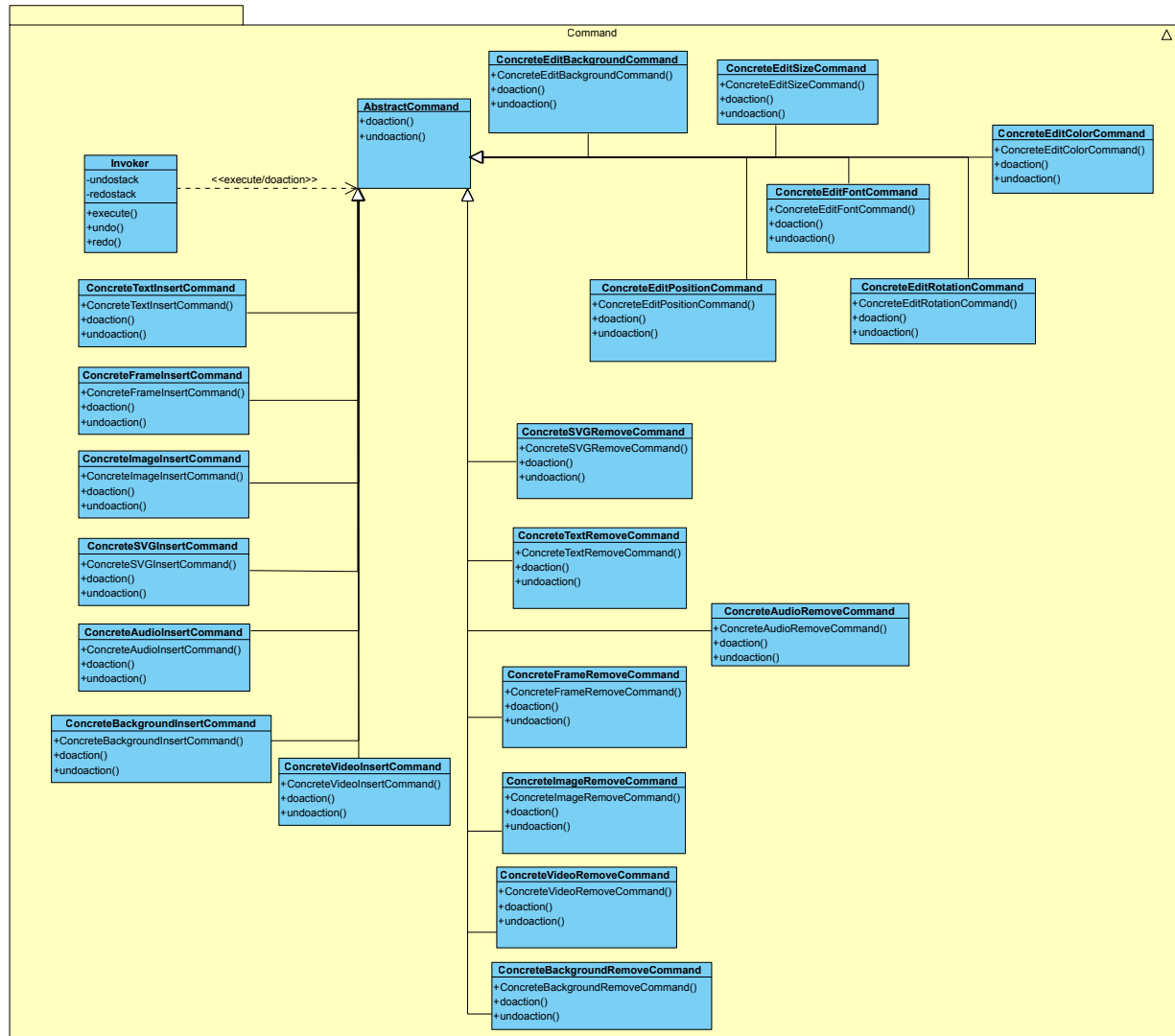


Fig 2: Diagramma delle classi del package Command



- **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

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.

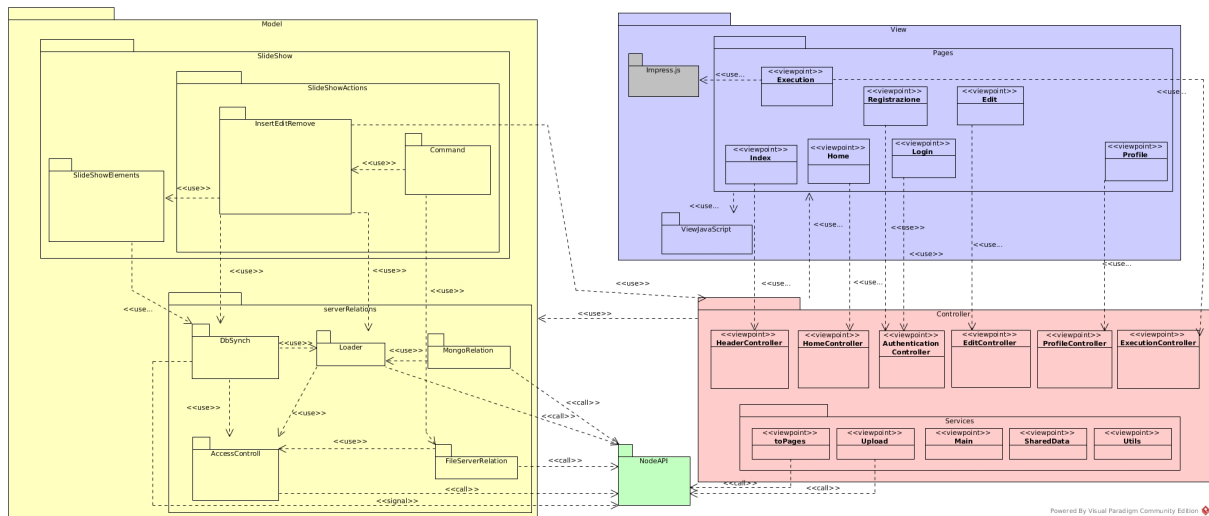


Fig 4: Architettura generale del sistema

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 architeturale del sistema rimandando 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.



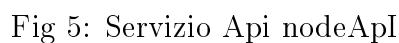
Contiene la rappresentazione dei dati, l'implementazione dei metodi da applicare ad essi e lo stato di questi ultimi; costituisce il cuore del software e risulta di fatto totalmente indipendente dagli altri due strati.

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

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

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



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



- **GET** /publicpages/[file]

- **tokenMiddleware:** verifica che il token passato nel campo Authorization dell' Header sia valido, dal token ricava lo username dell'utente

- **GET** /private/htdocs/[file]

- 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

- **POST** /private/api/presentations/new/[presentationName]

* **descrizione:** se non esiste già crea una nuova presentazione con il nome [presentationName]

- **POST** /private/api/presentations/new/[newPresentationName]/[oldPresentationName]

- * **descrizione:** crea una nuova presentazione con nome [newPresentationName] dalla presentazione con titolo [oldPresentationName]

- **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

- **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]



- **POST** /private/api/presentations/{presentationName}/element

- **PUT** /private/api/presentations/[presentationName]/element

- **DeleteElement:**

- **DELETE** /private/api/presentations/[presentationName]/delete/[type/[id element]]

- **GetImage:**

- **GET** /files/[user]/image/[imagename]

- * **descrizione:** ritorna il file [imagenname] nella cartella /users/[username]/image

- **GET** /files/[user]/audio/[audioname]

- * **descrizione:** ritorna il file [audioname] nella cartella /users/[username]/audios

- **GET** /files/[user]/video/[videoname]

- * **descrizione:** ritorna il file [videoname] nella cartella /users/[username]/videos

- **GET** /private/api/files/image

- * **descrizione:** ritorna un array con i nomi dei file immagine dell'utente

- **POST** /private/api/files/image/[imagename]

- * **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/images

- **DELETE** /private/api/files/image/[imagenamem]

- * **descrizione:** elimina il file immagine [imagename] dalla cartella /users/[username]/images

- **POST** /private/api/files/image/[imagenname]/[newname]

- * **descrizione:** rinomina il file immagine [imagenname] 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::serverRelation;

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.

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.

5.1.3 Model::SlideShow::SlideShowActions::InsertEditRemove

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



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::Insert si occupa di costruire gli oggetti presenti nelle classi del package Model::SlideShow::SlideShowElements.

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::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 `Editor`;



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



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



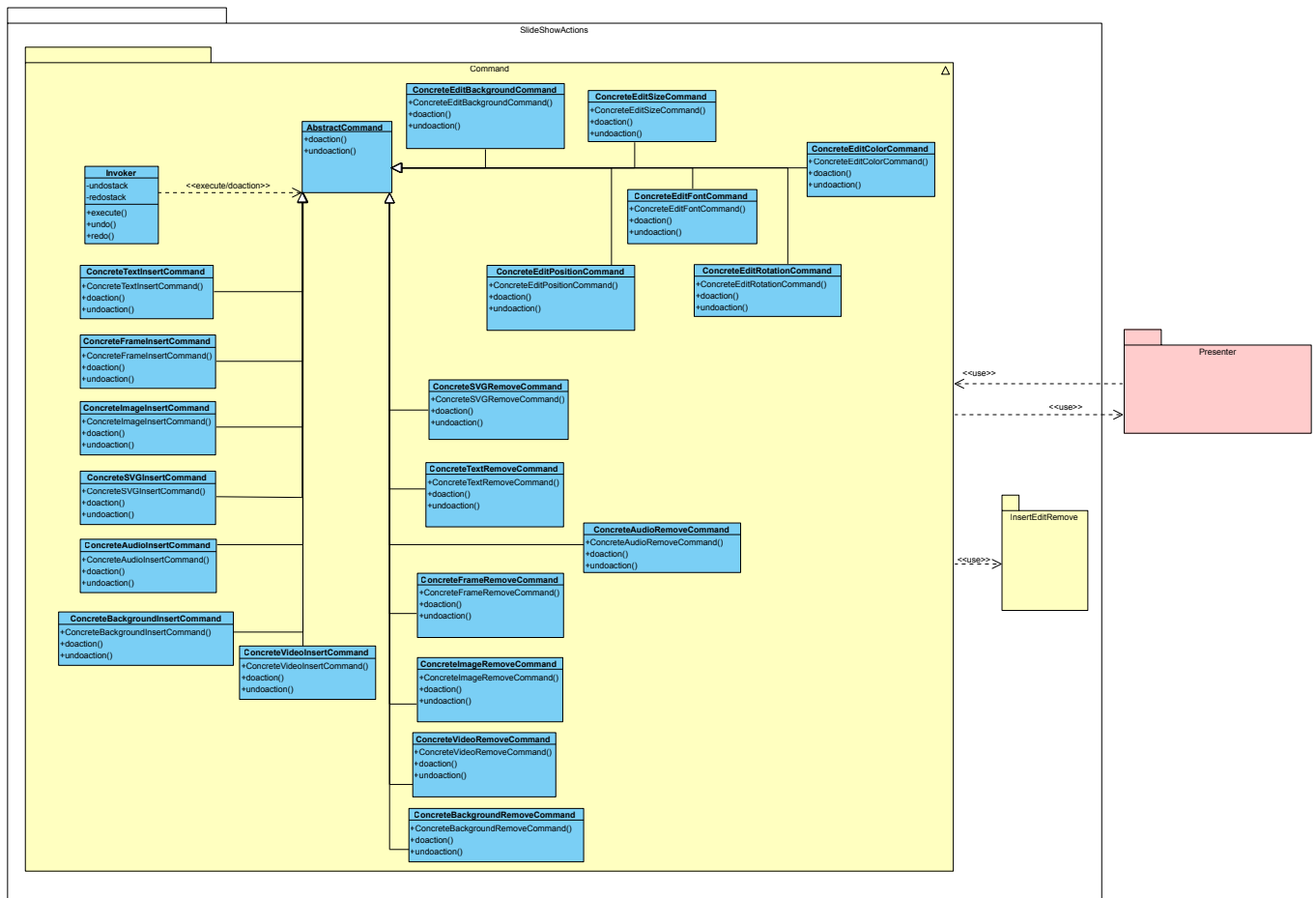


Fig 7: Command Package

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 appropriato 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::EditController->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";
- `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` [\[1\]](#).

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;



- #### 5.1.4.3 ConcreteTextInsertCommand

Relazioni d'uso di altre componenti:

- Università degli studi di Padova - 2014/2015



- `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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 appropriato 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 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 appropriato 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 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 appropriato 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 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 appropriato 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 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 appropriato 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 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 appropriato 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 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 appropriato 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 Classe ConcreteEditContentCommand

È classe concreta del Design Pattern Command, serve a assegnare del testo ad 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 -> 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 editContent(spec) della classe statica per la modifica dei campi dati relativi testo dell'oggetto della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo appropriato 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.

5.1.4.23 Classe ConcreteEditBookmarkCommand

È classe concreta del Design Pattern Command, applica un bookmark ad 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` -> 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 `updateBookmark(...)` della classe statica per la modifica del campo relativo al bookmark dell'elemento della presentazione;
- `Premi::Controller::EditController` <- l'oggetto invoca il metodo appropriato 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.

5.1.4.24 Classe ConcretePortaAvantiCommand

È classe concreta del Design Pattern Command, sposta all'indietro l'elemento su cui è stata invocata.

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::Editor <-` il comando invoca il metodo `portaAvanti()` della classe statica per la modifica del campo `z-index` degli oggetti della presentazione;
- `Premi::Controller::EditController <-` l'oggetto invoca il metodo appropriato 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.

5.1.4.25 Classe ConcretePortaDietroCommand

È classe concreta del Design Pattern Command, sposta all'indietro l'elemento su cui è stata invocata.

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::Editor` <- il comando invoca il metodo `portaDietro()` della classe statica per la modifica del campo `z-index` degli oggetti della presentazione;
- `Premi::Controller::EditController` <- l'oggetto invoca il metodo appropriato 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.

5.1.4.26 Classe ConcreteAddToMainPathCommand

È classe concreta del Design Pattern Command, inserisce in frame su cui è chiamata nella posizione specificata, all'interno del percorso principale.

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::Editor <- il comando invoca il metodo addFrameToMainPath(...) della classe statica per la modifica dell'oggetto relativo al percorso principale della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo appropriato 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.

5.1.4.27 Classe concreteRemoveFromMainPathCommand

È classe concreta del Design Pattern Command, rimuove dal percorso principale della presentazione il frame su cui è stata chiamata.

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::Editor` <- il comando invoca il metodo `removeFrameFromMainPath(...)` della classe statica per la modifica dell'oggetto relativo al percorso principale della presentazione;
- `Premi::Controller::EditController` <- l'oggetto invoca il metodo appropriato 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.

5.1.4.28 Classe concreteNewChoicePathCommand

È classe concreta del Design Pattern Command, riceve l'id del frame da cui parte il nuovo percorso scelta.

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::Editor <- il comando invoca il metodo addChoicePath(...) della classe statica per la modifica dell'oggetto relativo ai percorsi della presentazione;
- Premi::Controller::EditController <- l'oggetto invoca il metodo appropriato 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.

5.1.4.29 Classe concreteDeleteChoicePathCommand

È classe concreta del Design Pattern Command, rimuove dal percorso scelta il frame su cui è stata chiamata.

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::Editor` <- il comando invoca il metodo `deleteChoicePath(...)` della classe statica per la modifica dell'oggetto relativo al percorso principale della presentazione;
- `Premi::Controller::EditController` <- l'oggetto invoca il metodo appropriato 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`.

5.1.4.30 Classe ConcreteRemoveFromChoicePathCommand

È classe concreta del Design Pattern Command, rimuove dal percorso scelta il frame su cui è stata chiamata.

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 `removeFrameFromChoicePath(...)` della classe statica per la modifica dell'oggetto relativo al percorso scelta della presentazione;
- `Premi::Controller::EditController` <- l'oggetto invoca il metodo appropriato 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`.

5.1.4.31 Classe ConcreteAddToChoicePathCommand

È classe concreta del Design Pattern Command, aggiunge il frame su cui è chiamata ad un percorso scelta.

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 `addFrameToChoicePath(...)` della classe statica per la modifica dell'oggetto relativo al percorso scelta della presentazione;



- Classi ereditate:**

- ### 5.1.5 Model::SlideShow::SlideShowElements

```

classDiagram
    package SlideShowElements {
        class SlideShowElement {
            -height
            -width
            -position x
            -position y
            -rotation
            -id
            +setSize(height, width)
            +setPosition(x, y)
            +setRotation(degrees)
        }
        class Background {
            -color
            -ref
            +setBackground()
            +setColor()
        }
        class Frame {
            -bookmark, -tool
            -backgroundImage
            -backgroundColor
            +Frame()
            +changeExecutionNumber(int)
            +setBackgroundImage(string)
            +setBackgroundColor(string)
        }
        class SVG {
            -shape
            -color
            +SVG()
            +setShape(shape param)
            +setColor(string)
        }
        class Text {
            -content
            -fontType
            -fontStyle
            -fontSize
            -color
            +Text()
            +setFontType(string)
            +setFontStyle(string)
            +setFontSize(int)
            +setContent(string)
            +setColor(string)
        }
        class Image {
            -ref
            +Image()
        }
        class Audio {
            -ref
            +Audio()
        }
        class Video {
            -ref
            +Video()
        }
    }

    SlideShowElement <|-- Frame
    SlideShowElement <|-- SVG
    SlideShowElement <|-- Text
    SlideShowElement <|-- Image
    SlideShowElement <|-- Audio
    SlideShowElement <|-- Video

    class Model {
        class SlideShowActions {
            class InsertEditRemove {
                ..> Background : setBackground()
            }
        }
    }

```

Fig 8: SlideShowElements

Relazioni d'uso di altre componenti: Model::SlideShow::SlideShowElements è in comunicazione con

- #### 5.1.5.1 SlideShowElement

Relazioni d'uso di altre componenti:

- Università degli studi di Padova - 2014/2015



- 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::InsertEditRemove::Remover -> gli oggetti delle sue sottoclassi rimuovono dai contenitori di SlideShow gli oggetti di classe SlideShowElement e ne richiamano i distruttori.

Interfacce con e relazioni d'uso e da altre componenti:

Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter istanzia oggetti di sottoclassi di SlideShowElement.

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 nella presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Text dalla presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

Interfacce con e relazioni d'uso e da altre componenti: gli oggetti della classe Text vengono istanziati da Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter e inseriti nella presentazione.

Classi ereditate:

- Model::SlideShow::SlideShowElement.



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



- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto SVG dalla presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

Interfacce con e relazioni d'uso e da altre componenti: Gli oggetti della classe SVG vengono istanziati da Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter e inseriti nella presentazione.

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 nella presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Audio dalla presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.

Interfacce con e relazioni d'uso e da altre componenti: gli oggetti della classe Audio vengono istanziati da Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter e inseriti nella presentazione.

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 nella presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Remover -> rimuove l'oggetto Video dalla presentazione;
- Model::SlideShow::SlideShowActions::InsertEditRemove::Editor -> invoca i metodi che modificano i campi dati dell'oggetto.



Classi ereditate:

- ### 5.1.6 Background

Relazioni d'uso di altre componenti:

- Classi ereditate:**

- ### 5.1.7 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

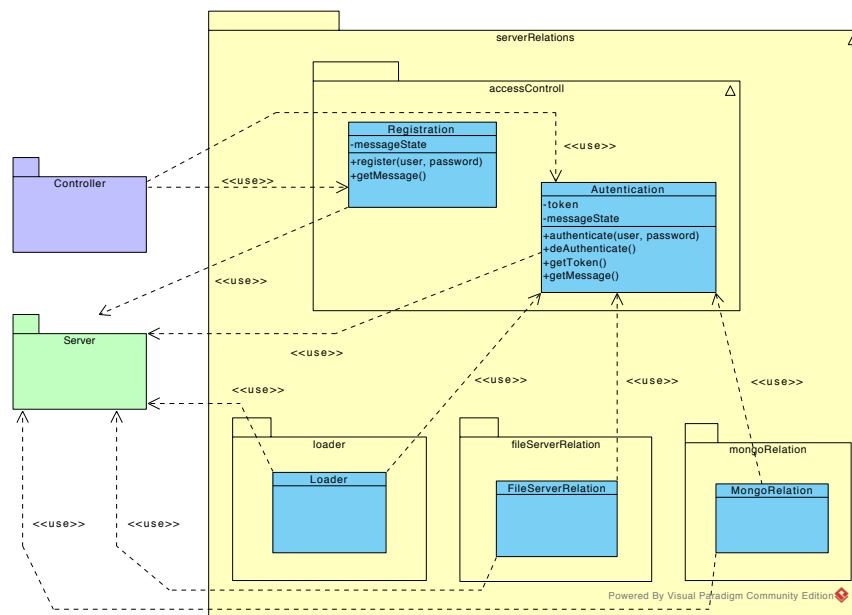


Fig 10: accessControl

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 Authentication

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, cambio password 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 Registration

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

Loader
-toInsert : object
-toUpdate : object
-toDelete : object
+update() : bool
+addInsert(idElement : string) : bool
+addUpdate(idElement : string) : bool
+addDelete(idElement : string) : bool

Fig 11: serverRelation::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 la modifica della presentazione nel database MongoDB
- relazione da **Model::SlideShow::InsertEditRemove** a cui viene esposta la rappresentazione della presentazione locale per essere modificata



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

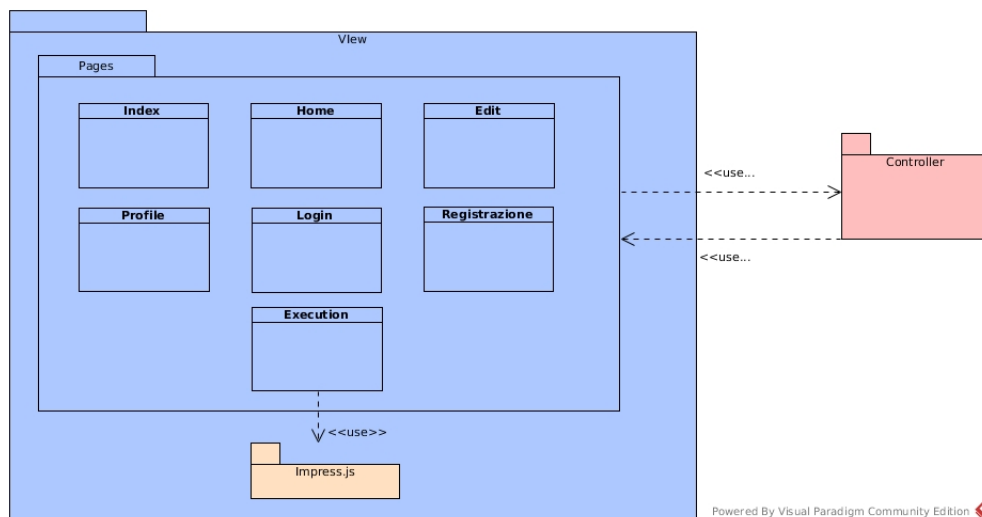


Fig 12: View

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::HeaderController per effettuare il logout o il reindirizzamento alle pagine Home e Profile.

Attività svolte e dati trattati: la classe definisce la struttura della pagina web comune a tutte le altre pagine.



5.2.3 View::Pages::Login

Tipo, obiettivo e funzione del componente: la classe Login definisce la struttura, e la conseguente visualizzazione, della pagina web che consente ad un utente di effettuare il login al sistema.

Relazioni d'uso di altre componenti: la classe Login utilizza i metodi messi a disposizione dalla classe Controller::AuthenticationController per verificare i dati inseriti, per inviare i dati relativi alla login e per visualizzare eventuali errori.

Attività svolte e dati trattati: la classe definisce la struttura della pagina web che consente agli utenti di autenticarsi al sistema. Essa resta in attesa che un utente inserisca i dati necessari per l'autenticazione al sistema.

5.2.4 View::Pages::Registrazione

Tipo, obiettivo e funzione del componente: la classe Registrazione definisce la struttura, e la conseguente visualizzazione, della pagina web che consente ad un utente di effettuare la registrazione al sistema.

Relazioni d'uso di altre componenti: la classe Registrazione utilizza i metodi messi a disposizione dalla classe Controller::AuthenticationController per verificare i dati inseriti, per inviare i dati relativi alla registrazione e per visualizzare eventuali errori.

Attività svolte e dati trattati: la classe definisce la struttura della pagina web che consente agli utenti di registrarsi al sistema. Essa resta in attesa che un utente inserisca i dati necessari per la registrazione al sistema.

5.2.5 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 per gestirle.

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, la loro rinominazione o la creazione di una nuova.

Attività svolte e dati trattati: la classe definisce la struttura della pagina web che consente agli utenti di visualizzare una lista delle proprie presentazioni, crearne di nuove, modificarle, eliminarle, scaricarle, eseguirle o modificarle.

5.2.6 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 la modifica della password.

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 e la possibilità di modificarli.

5.2.7 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; 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.

5.2.8 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 il loro spostamento ed eliminazione, 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 e assegnare bookmark ai frame.

La classe dovrà predisporre delle apposite funzioni javascript per la gestione degli elementi nella view.

5.2.9 View::Pages::Manifest

Tipo, obiettivo e funzione del componente: la classe Manifest definisce la struttura, e la conseguente visualizzazione, della pagina web che mostra all'utente le presentazioni salvate in locale, permettendone l'esecuzione.

Attività svolte e dati trattati: la classe definisce la struttura della pagina web che consente agli utenti di visualizzare le presentazioni salvate in locale e permette la loro esecuzione.

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`;



- 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 HeaderComponent una richiesta di logout o di reindirizzamento ad una pagina. HeaderComponent 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.



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:

- `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:

- `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 ottenere 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.

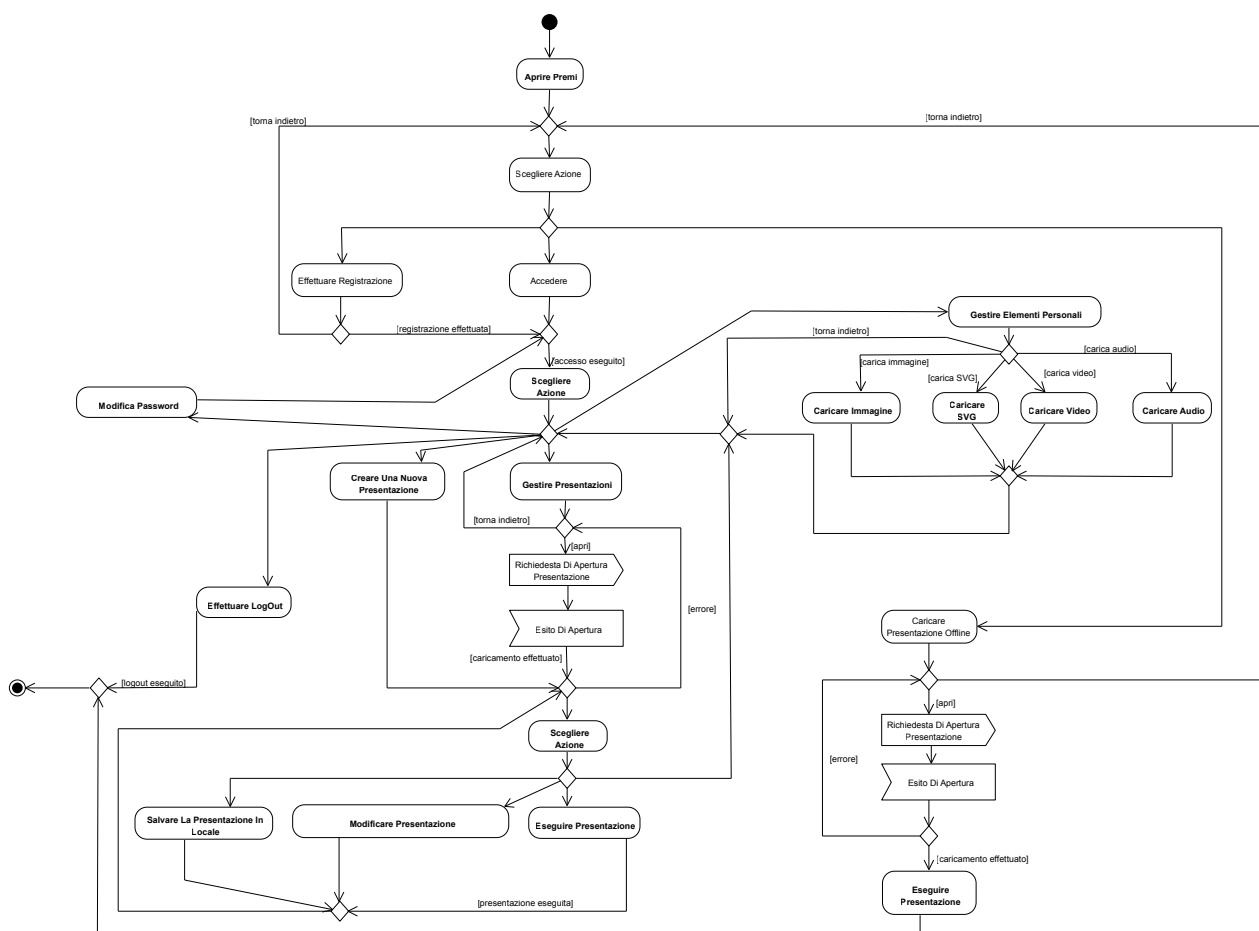


Fig 13: Attività Principali

6.1.1 Gestione presentazioni

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



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.

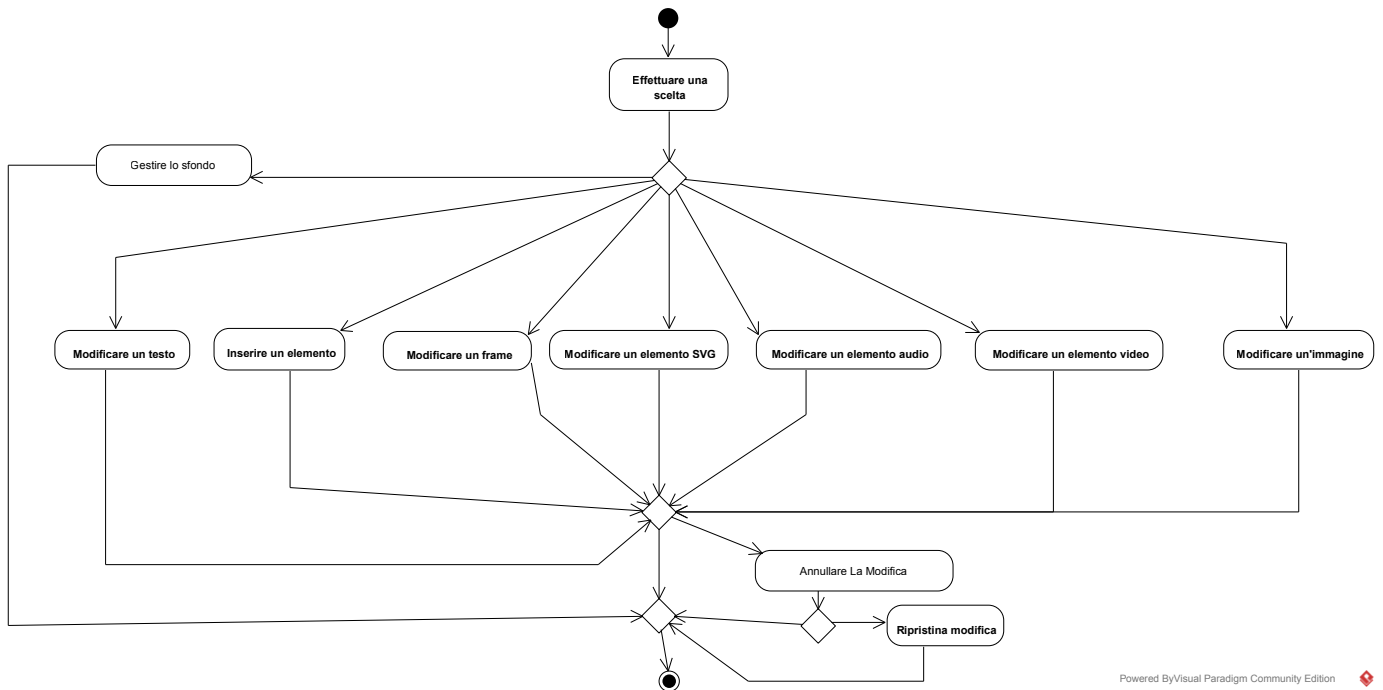


Fig 16: Modificare Presentazione da Desktop

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.

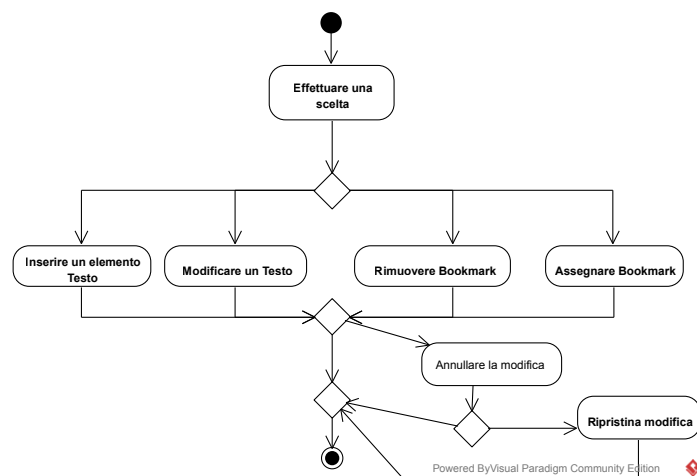


Fig 17: Modificare Presentazione da Mobile



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



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

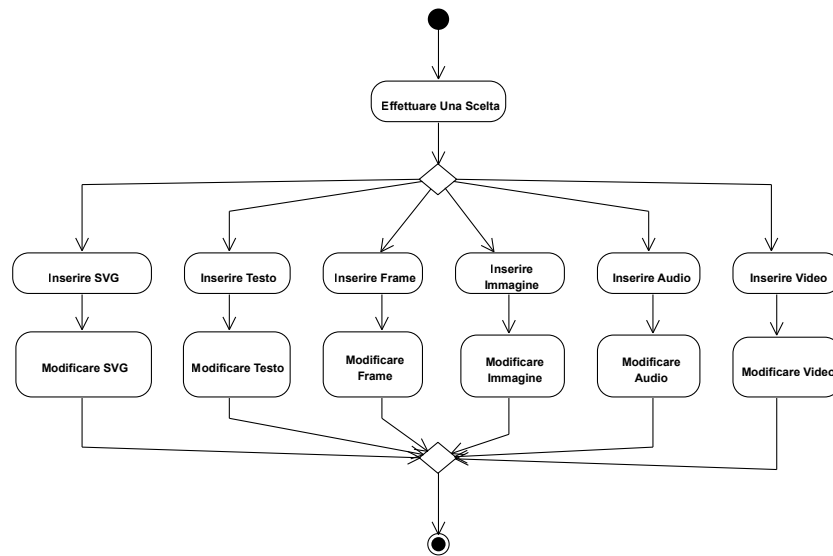


Fig 19: Inserire Elemento

6.1.7 Modificare Elemento

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

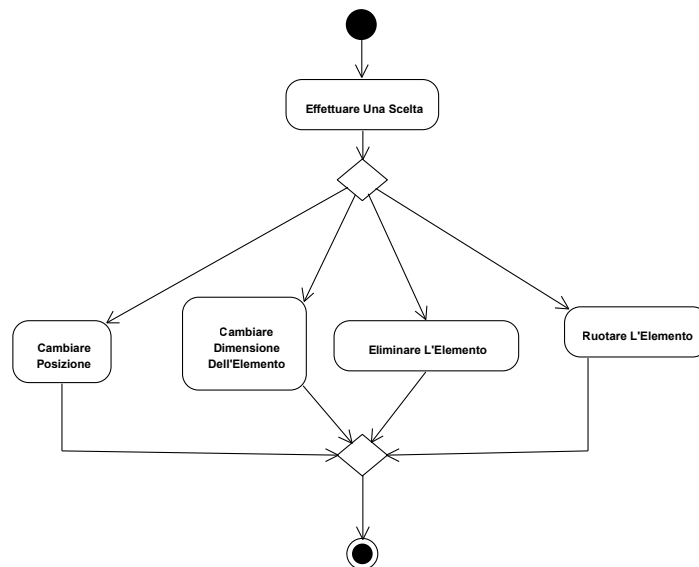


Fig 20: Modificare Elemento

6.1.8 Modificare Frame

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

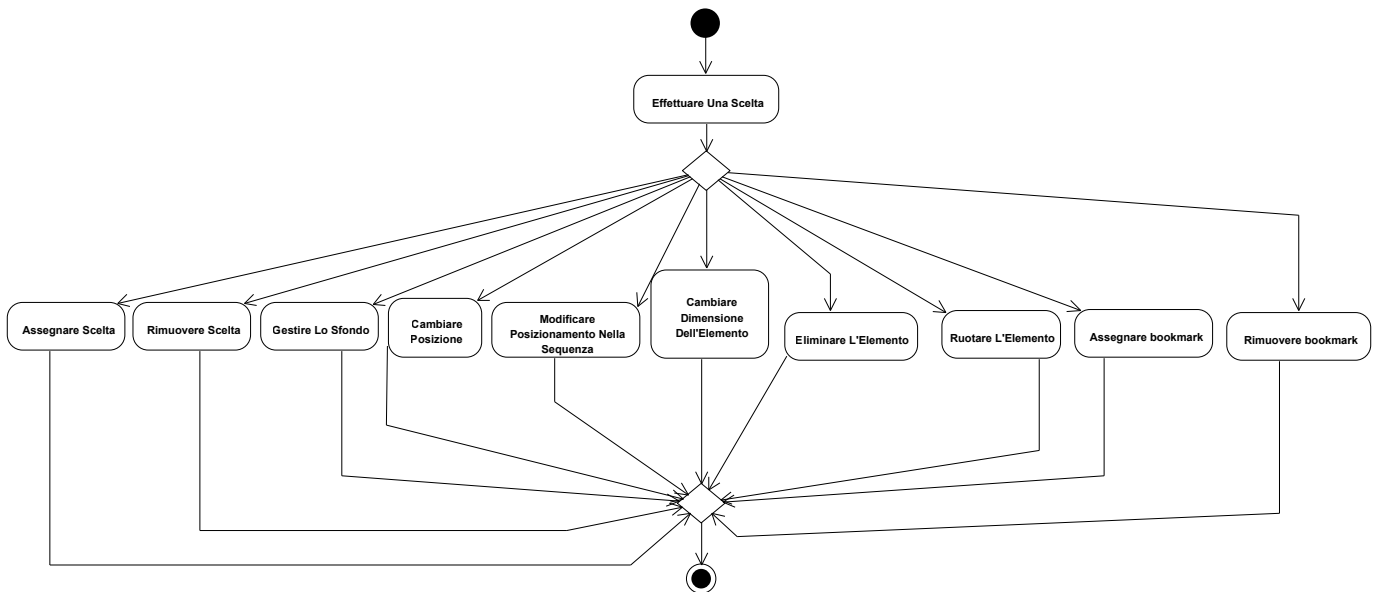


Fig 21: Modificare Frame

6.1.9 Modificare SVG

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

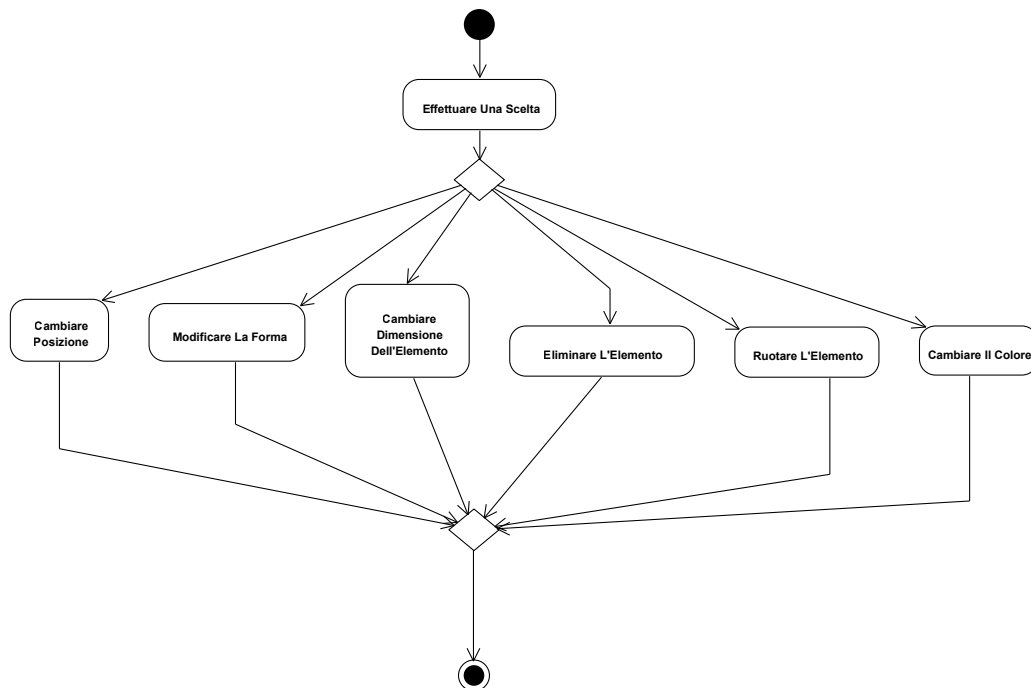


Fig 22: Modificare SVG



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





7 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente per fornire una stima sulla fattibilità e di bisogno di risorse. L'analisi dell'architettura progettata ha permesso di constatare che le tecnologie che si è scelto di adottare risultano sufficientemente adeguate per la realizzazione del prodotto e riescono a ricoprire le esigenze progettuali.

Poiché tutti gli strumenti da utilizzare nello sviluppo sono gratuiti, il bisogno di risorse non si dimostra essere particolarmente problematico.

Si è deciso di utilizzare HTML5, CSS3 e Javascript (e le sue librerie) per lo sviluppo della parte web.

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

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

Per la parte di modifica delle presentazioni 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.



8.1 Tracciamento Componenti-Requisiti

Componente	Requisiti
Controller	
- >AccessController	
- >EditController	
- >ExecutionController	
- >HeaderController	
- >HomeController	
- >ProfileController	
- >Services	
- - >Main	
- - >SharedData	
- - >toPages	
- - >Upload	
- - >Utils	
Model	
- >serverRelation	
- - >accessControl	
- - - >Authentication	RF 3, RF 43, RF 3.1, RF 3.2, RF 64
- - - >Registration	RF 1, RF 1.1, RF 1.2
- - >fileServerRelation	RF 16, RF 12, RF 17, RF 37

Componente	Requisiti
- - >Loader	RF 7, RF 7.1, RF 7.4, RF 7.7, RF 7.7.1, RF 7.7.4, RF 7.7.7, RF 7.7.10, RF 7.7.13, RF 7.7.19, RF 7.7.25, RF 7.7.16, RF 7.7.28, RF 7.7.34, RF 7.10, RF 7.16, RF 7.13, RF 7.19, RF 7.19.1, RF 7.37, RF 7.40, RF 7.40.1, RF 7.40.4, RF 7.43, RF 7.46, RF 7.7.46
- - >mongoRelation	RF 7.1, RF 7.7.7, RF 7.7.13, RF 7.16, RF 7.13, RF 7.19.1, RF 7.19.4, RF 7.19.10, RF 7.19.13, RF 19, RF 34, RF 7.37, RF 35
- >SlideShow	
- - >SlideShowActions	
- - - >Command	
- - - - >AbstractCommand	
- - - - - >ConcreteAddToChoicePathCommand	RF 7.7.25
- - - - - >ConcreteAddToMainPathCommand	RF 7.19.4
- - - - - >ConcreteAudioInsertCommand	RF 7.7.13
- - - - - >ConcreteAudioRemoveCommand	RF 7.43
- - - - - >ConcreteBackgroundInsertCommand	RF 7.13
- - - - - >concreteDeleteChoicePathCommand	RF 7.43
- - - - - >ConcreteEditBackgroundCommand	RF 7.7.43
- - - - - >ConcreteEditBookmarkCommand	RF 7.22, RF 7.25, RF 10.5, RF 10.8
- - - - - >ConcreteEditColorCommand	RF 7.7.4, RF 7.7.40, RF 7.16, RF 7.40.4
- - - - - >ConcreteEditContentCommand	RF 7.19.13
- - - - - >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

Componente	Requisiti
- - - - >ConcreteFrameInsertCommand	RF 7.1, RF 7.1.1
- - - - >ConcreteFrameRemoveCommand	RF 7.10
- - - - >ConcreteImageInsertCommand	RF 7.7.7
- - - - >ConcreteImageRemoveCommand	RF 7.43
- - - - >concreteNewChoicePathCommand	RF 7.7.25
- - - - >ConcretePortaAvantiCommand	
- - - - >ConcretePortaDietroCommand	
- - - - >ConcreteRemoveFromChoicePath- Command	RF 7.7.28
- - - - >concreteRemoveFromMainPathCom- mand	RF 7.19.13
- - - - >ConcreteSVGInsertCommand	RF 7.37
- - - - >ConcreteSVGRemoveCommand	RF 7.43
- - - - >ConcreteTextInsertCommand	RF 7.7.1
- - - - >ConcreteTextRemoveCommand	RF 7.43
- - - - >ConcreteVideoInsertCommand	RF 7.7.13
- - - - >ConcreteVideoRemoveCommand	RF 7.43
- - - - >Invoker	RF 55, RF 58
- - - >InsertEditRemove	
- - - - >Editor	RF 7.7.4, RF 7.7.10, RF 7.7.19, RF 7.7.16, RF 7.7.40, RF 7.7.43, RF 7.16, RF 7.40.4, RF 7.46, RF 7.7.46
- - - - >Inserter	RF 7.1, RF 7.1.1, RF 7.7.1, RF 7.7.7, RF 7.7.13, RF 7.13, RF 7.37
- - - - >Remover	RF 7.10, RF 7.43
- - >SlideShowElements	
- - - >Audio	RF 61.1.16
- - - >Background	RF 7.7.13
- - - >Frame	RF 7.1
- - - >Image	RF 7.7.7, RF 7.7.13

Componente	Requisiti
- - - >SVG	
- - - >Text	RF 7.7.1
- - - >Video	RF 61.1.16
View	
- >Pages	
- - >Edit	RF 7, RF 7.1, RF 7.1.1, RF 7.4, RF 7.7, RF 7.7.1, RF 7.7.4, RF 7.7.7, RF 7.7.10, RF 7.7.13, RF 7.7.19, RF 7.7.25, RF 7.7.16, RF 7.7.28, RF 7.7.31, RF 7.7.34, RF 7.7.37, RF 7.7.40, RF 7.7.43, RF 7.10, RF 7.16, RF 7.13, RF 7.19, RF 7.19.1, RF 7.19.4, RF 7.19.10, RF 7.19.13, RF 7.28, RF 7.31, RF 7.34
- - >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
- - >Index	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



Tab 5: Tracciamento Requisiti-Componenti

Università degli studi di Padova - 2014/2015

Requisito	Componenti
RF 7.7.10	View::Pages::Edit, Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.13	View::Pages::Edit, Model::SlideShow::SlideShowActions::InsertEditRemove::Inserter, Model::SlideShow::SlideShowActions::Command::ConcreteVideoInsertCommand, Model::SlideShow::SlideShowElements::Image, Model::serverRelation::Loader, Model::serverRelation::mongoRelation, Model::SlideShow::SlideShowElements::Background, Model::SlideShow::SlideShowActions::Command::ConcreteAudioInsertCommand
RF 7.7.16	View::Pages::Edit, Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.19	View::Pages::Edit, Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.25	View::Pages::Edit, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::Command::concreteNewChoicePathCommand, Model::SlideShow::SlideShowActions::Command::ConcreteAddToChoicePathCommand
RF 7.7.28	View::Pages::Edit, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::Command::ConcreteRemoveFromChoicePathCommand
RF 7.7.31	View::Pages::Edit
RF 7.7.34	View::Pages::Edit, Model::serverRelation::Loader
RF 7.7.37	View::Pages::Edit
RF 7.7.40	View::Pages::Edit, Model::SlideShow::SlideShowActions::Command::ConcreteEditColorCommand, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.43	View::Pages::Edit, Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.7.46	Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 7.10	View::Pages::Edit, Model::SlideShow::SlideShowActions::InsertEditRemove::Remover, Model::SlideShow::SlideShowActions::Command::ConcreteFrameRemoveCommand, Model::serverRelation::Loader



Questo documento è distribuito sotto licenza [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).

Requisito	Componenti
RF 7.43	Model::SlideShow::SlideShowActions::InsertEditRemove::-Remover, Model::SlideShow::SlideShowActions::Command::ConcreteTextRemoveCommand, Model::SlideShow::SlideShowActions::Command::ConcreteImageRemoveCommand, Model::SlideShow::SlideShowActions::Command::ConcreteSVGRemoveCommand, Model::SlideShow::SlideShowActions::Command::ConcreteVideoRemoveCommand, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::Command::ConcreteAudioRemoveCommand, Model::SlideShow::SlideShowActions::Command::concreteDeleteChoicePathCommand
RF 7.46	Model::SlideShow::SlideShowActions::Command::ConcreteEditRotationCommand, Model::serverRelation::Loader, Model::SlideShow::SlideShowActions::InsertEditRemove::Editor
RF 10	View::Pages::Home
RF 10.1	
RF 10.4	
RF 10.5	Model::SlideShow::SlideShowActions::Command::ConcreteEditBookmarkCommand
RF 10.8	Model::SlideShow::SlideShowActions::Command::ConcreteEditBookmarkCommand
RF 12	Model::serverRelation::fileServerRelation
RF 13	View::Pages::Profile
RF 16	View::Pages::Profile, Model::serverRelation::fileServerRelation
RF 17	View::Pages::Profile, Model::serverRelation::fileServerRelation
RF 19	View::Pages::Home, Model::serverRelation::mongoRelation
RF 25	
RF 31	
RF 34	View::Pages::Home, Model::serverRelation::mongoRelation
RF 35	Model::serverRelation::mongoRelation
RF 36	
RF 37	Model::serverRelation::fileServerRelation
RF 43	View::Pages::Profile, Model::serverRelation::accessControl::Authentication
RF 46	
RF 49	View::Pages::Home



Requisito	Componenti
RF 52	View::Pages::Manifest
RF 55	Model::SlideShow::SlideShowActions::Command::Invoker
RF 58	Model::SlideShow::SlideShowActions::Command::Invoker
RF 61	View::Pages::Manifest, View::Pages::Execution
RF 61.1	View::Pages::Execution
RF 61.1.1	View::Pages::Execution
RF 61.1.4	View::Pages::Execution
RF 61.1.7	View::Pages::Execution
RF 61.1.10	View::Pages::Execution
RF 61.1.13	View::Pages::Execution
RF 61.1.16	View::Pages::Execution, Model::SlideShow::SlideShowElements::Audio, Model::SlideShow::SlideShowElements::Video
RF 61.1.16.1	View::Pages::Execution
RF 61.1.16.4	View::Pages::Execution
RF 61.1.16.7	View::Pages::Execution
RF 61.1.16.10	View::Pages::Execution
RF 61.4	View::Pages::Execution
RF 61.4.1	View::Pages::Execution
RF 61.4.4	View::Pages::Execution
RF 61.4.7	View::Pages::Execution
RF 61.4.10	View::Pages::Execution
RF 61.4.10.1	View::Pages::Execution
RF 61.4.10.4	View::Pages::Execution
RF 61.4.10.7	View::Pages::Execution
RF 61.4.10.10	View::Pages::Execution
RF 61.7	View::Pages::Execution

Requisito	Componenti
RF 61.10	View::Pages::Execution
RF 64	View::Pages::Home, Model::serverRelation::accessControl:- Authentication
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	
RF 70.10.1.7	
RF 70.10.4	
RF 70.10.7	



Università degli studi di Padova - 2014/2015