09-06-2015

Definizione del Prodotto

Informazioni sul documento

Nome Documento	Definizione del Prodotto
Versione	2.0.0
Stato	*Formale*
Uso	*Esterno*
Data Creazione	09-06-2015
Data Ultima Modifica	09-06-2015
Redazione	Fossa Manuel, Venturelli Giovanni, Tollot Pietro, Busetto Matteo
Approvazione	Busetto Matteo
Verifica	Petrucci Mauro
Lista distribuzione	*LateButSafe* Prof. Tullio Vardanega Prof. Riccardo Cardin Proponente Zucchetti S.p.a.

Sommario

Il presente documento riporta la Definizione del Prodotto effettuata per il capitolato Premi.

Registro delle modifiche

Tab 1: Versionamento del documento

Versione	Autore	Data	Descrizione
1.11.0	Busetto Matteo	04-08-2015	Aggiornato Controller::goRegistrazione, goHome, AccessController
1.10.0	Fossa Manuel	03-08-2015	Corretto output di NodeServer::PresentationMeta: GET /private/api/presentations
1.9.0	Fossa Manuel	02-08-2015	Corretto serverRelation::MongoRelation, schema di NodeServer, definizione di NodeServer
1.8.0	Busetto Matteo, Petrucci Mauro	31-07-2015	Aggiornata specifica classi del front-end, Premi::App
1.7.0	Petrucci Mauro	28-07-2015	Inserimento services di Angular in Premi::App
1.6.0	Petrucci Mauro	27-07-2015	Aggiunta di contenuti. Inserimento di Premi::Services, modifica di Premi::Controller
1.5.0	Venturelli Giovanni	19-07-2015	Aggiornamento capitolo Premi::Model, metodi di inserimento oggetti della presentazione.
1.4.0	Tollot Pietro	14-07-2015	Aggiornamento metodi di serverRelation::Registration, MongoRelation, Loader, FileServerRelation, Authentication
1.3.0	Tollot Pietro	02-07-2015	
1.2.0	Tollot Pietro	30-06-2015	Aggiornamento schema del backEndProgettazione
1.1.0	Busetto Matteo	28-06-2015	Aggiornamento Controller::InsertFrame
1.0.0	Tollot Pietro	27-06-2015	Aggiornamento Model:: ServerRelations, Model::MongoRelations

Versione	Autore	Data	Descrizione
0.7.0	Busetto Matteo	24-06-2015	Aggiunta di contenuti. Inserimento del capitolo Package::Premi::Controller
0.5.0	Venturelli Giovanni	20-06-2015	Aggiunta di contenuti. Inserimento del capitolo Package::Premi::Model
0.4.0	Fossa Manuel	15-06-2015	Aggiunta di contenuti. Inserimento del capitolo Package::Premi::View
0.3.0	Tollot Pietro	12-06-2015	Aggiunta di contenuti. Inserimento del capitolo Standard di Progetto
0.2.5	Gabelli Pietro	09-06-2015	Aggiunta di contenuti. Inserimento del capitolo Introduzione e Descrizione generale
0.1.0	Gabelli Pietro	08-06-2015	Stesura dello scheletro del documento

Storico

RP -> RQ

Versione 1..0.0	**Nominativo**
Redazione	Fossa Manuel, Venturelli Giovanni, Tollot Pietro, Busetto Matteo
Verifica	Petrucci Mauro
Approvazione	Busetto Matteo

Tab 2: Storico ruoli RP -> RQ

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il presente documento descrive la progettazione di dettaglio definita per il progetto Premi. Il documento si basa sulla SpecificaTecnica_v.1.0.0.pdf. I programmatori si serviranno di tale documento per procedere con le attività di codifica.

1.2 Scopo del Prodotto

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

1.3 Glossario

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

1.4 Riferimenti

1.4.1 Normativi

- Regole del Progetto$_g$ didattico, reperibili all'Indirizzo$_g$:
  http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/PD01.pdf
- Vincoli di organigramma, consultabili all'Indirizzo$_g$:
  http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/PD01b.html
- Norme di Progetto$_g$: NormeDiProgetto_v.2.0.0.pdf;
- Specifica Tecnica$_g$: SpecificaTecnica_v.1.0.0.pdf;
- Capitolato d'appalto C4: Premi: Software$_g$ di presentazione "better than Prezi"
  http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf.

1.4.2 Informativi

- Slide dell'insegnamento Ingegneria del Software$_g$ modulo A:
  - Il ciclo di vita$_g$ del Software$_g$;
  - Gestione di Progetto$_g$.

  http://www.math.unipd.it/~tullio/IS-1/2014/ ;
- Ingegneria del Software$_g$ - Ian Sommerville - 9a Edizione (2010).

2 Descrizione generale

Il sistema si pone come obbiettivo quello di permettere la creazione di presentazioni efficaci dal punto di vista dello storytelling anche ad utenti non esperti.
L'applicazione Premi permette all'utente di creare ed eseguire presentazioni personalizzate. Attraverso la creazione di Frame$_g$ e la loro modellazione, l'utente potrà definire un Percorso$_g$ di presentazione lineare oppure più percorsi$_g$ che prevedono la possibilità di scegliere con quale continuare il flusso di esecuzione. Questo significa che il Percorso$_g$ di Frame$_g$ che sarà visualizzato sarà scelto dal presentatore in fase di visualizzazione.
L'applicazione è strutturata in modo gerarchico, ossia ogni Frame$_g$ ha almeno un padre (tranne la radice che può avere solamente figli). Questo permette di creare presentazioni strutturate a livelli, rendendo molto semplice la possibilità, durante l'esecuzione, di saltare determinati rami della presentazione. Inoltre, l'utente potrà inserire Bookmark$_g$ i quali permettono di saltare ad un Frame$_g$ padre in modo semplice e veloce.
Il sistema permetterà di creare e modificare presentazioni se connessi alla rete mentre l'utente potrà eseguire le proprie presentazione anche offline a patto di averle precedentemente scaricate dal Server$_g$. Il sistema sarà implementato utilizzando tecnologie WEB$_g$ che lo renderanno altamente portabile.

2.1 funzioni$_g$ del prodotto

Il prodotto offre un'interfaccia WEB$_g$ che permetterà di:

- Registrarsi, accedere al proprio Account$_g$ ed effettuare il Logout$_g$;
- Gestire il proprio Account$_g$;
- Creare una nuova presentazione da dispositivo Desktop$_g$;
- Modificare una presentazione da dispositivo Desktop$_g$;
- Modificare parzialmente la presentazione da dispositivo mobile$_g$;
- Eseguire una presentazione salvata sul proprio Account$_g$;
- Eseguire una presentazione locale;
- Creare Infografiche$_g$ a partire da una presentazione;
- Modificare Infografiche$_g$ create;
- Gestire il proprio archivio di File$_g$ media;
- Scaricare una presentazione in locale.

2.2 Caratteristiche degli utenti

Il prodotto si rivolge a qualsiasi tipo di utente interessato ad una facile creazione e modellazione di presentazioni ed Infografiche$_g$. Non emergono quindi restrizioni particolari riguardo le caratteristiche dell'utenza.

2.3 Vincoli generali

Per poter utilizzare il Software$_g$ Premi è necessario disporre di un computer o di un dispositivo mobile$_g$ con installato almeno uno dei principali Browser$_g$ quali Google Chrome, Mozilla Firefox e Safari scaricabili dai rispettivi siti ufficiali. Inoltre, il dispositivo deve supportare le seguenti tecnologie:

- HTML5;
- CSS3;
- Javascript.

Il prodotto non richiede particolari Requisiti$_g$ hardware anche se gli stessi possono influenzarne la velocità di esecuzione.

3 Standard di progetto

3.1 Standard di progettazione architetturale

Gli standard di progettazione architetturale sono definiti nel documento SpecificaTecnica_v.1.0.0.pdf.

3.2 Standard di documentazione del codice

Gli standard per la scrittura di documentazione del codice sono definiti nelle NormeDiProgetto_v.2.0.0.pdf

3.3 Standard di denominazione di entità e relazioni

Tutti gli elementi (package, classi, metodi o attributi) definiti, devono avere denominazioni chiare ed autoesplicative. Nel caso il nome risulti lungo, è preferibile preferire la chiarezza alla lunghezza.
Sono ammesse abbreviazioni se:

- immediatamente comprensibili;
- non ambigue;
- sufficientemente contestualizzate.

Le regole tipografiche relative ai nomi delle entità sono definite nelle NormeDiProgetto_v.2.0.0.pdf.

3.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle NormeDiProgetto_v.2.0.0.pdf.

3.5 Strumenti di lavoro

Gli strumenti da adottare e le procedure per utilizzarli correttamente durante la realizzazione del prodotto software sono definiti nelle NormeDiProgetto_v.2.0.0.pdf.

4 NodeServer

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

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

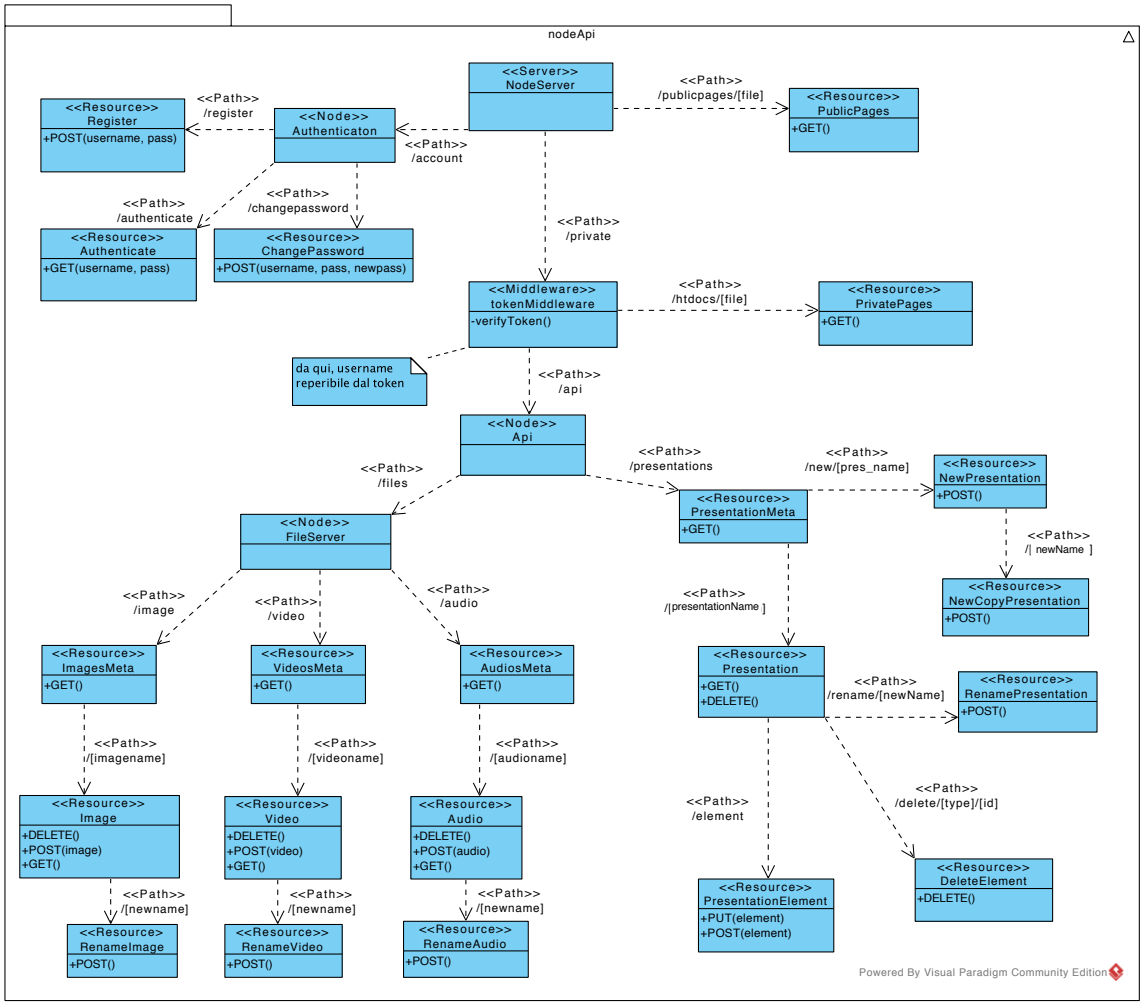

Fig 1: Servizi RESTfull offerti dal server nodeJs

4.1 Risorse e Servizi

- **NodeServer:** radice dei servizi offerti dal server:
  1. server per pagine html e file statici associati
  2. servizi di autenticazione stateless
  3. servizi di upload e reperimento file statici multimediali per utente
  4. servizi di interazione con MongoDB per salvataggio persistente delle presentazioni
- **Register:**
  - **POST** /account/register
    - **descrizione:** inserisce nuovo utente in MongoDB, crea una nuova collezione 'presentations'+username, crea le cartelle per i file utente
    - **input:** header campo Authorization: username:password
    - **output:** body in formato application/json, success : boolean
- **Authenticate:**
  - **GET** /account/authenticate
    - **descrizione:** verifica se username e password sono corretti e ritorna un token per l'accesso ai servizi protetti
    - **input:** header campo Authorization: username:password
    - **output:** body in formato application/json, success : boolean; in header campo Authorisation ritorna il token per l'accesso ai servizi protetti
- **ChangePassword:**
  - **POST** /account/changepassword
    - **descrizione:** verifica la correttezza di username e password e modifica quest'ultima con la nuova
    - **input:** in header campo Authorization: username:password:newpassword
    - **output:** body in formato application/json, success : boolean
- **PublicPages:**
  - **GET** /publicpages/[file]
    - **descrizione:** se presente [file] nella cartella /public_html del server ritorna il file stesso
    - **input** /
    - **output:** fileStatico
- **tokenMiddleware:** verifica che il token passato nel campo Authorization dell' Header sia valido, dal token ricava lo username dell'utente
- **PrivatePages:**

  - **GET** /private/htdocs/[file]
    - **descrizione:** se presente [file] nella cartella /private_html del server ritorna il file stesso
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** fileStatico

- **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
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean, message : array

- **NewPresentation:**
  - **POST** /private/api/presentations/new/[presentationName]
    - **descrizione:** se non esiste già crea una nuova presentazione con il nome [presentatioNname]
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean

- **NewCopyPresentation:**
  - **POST** /private/api/presentations/new/[newPresentationName]/[oldPresentationName]
    - **descrizione:** crea una nuova presentazione con nome [newPresentationName] dalla presentazione con titolo [oldPresentationName]
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean

- **Presentation:**
  - **GET** /private/api/presentations/[presentationName]
    - **descrizione:** recupera se presente la presentazione dell'utente associata al titolo passato nell'url
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean, presentation : object
  - **DELETE** /private/api/presentations/[presentationName]
    - **descrizione:** elimina se presente la presentazione dell'utente associata al titolo passato nell'url
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean

- **RenamePresentation:**
  - **POST** /private/api/presentations/[presentationName]/rename/[newname]
    - **descrizione:** rinomina se presente la presentazione dell'utente associata al titolo passato nell'url con il nome [newname]
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean
- **PresentationElement:**
  - **POST** /private/api/presentations/[presentationName]/element
    - **descrizione:** inserisce nella presentazione dell'utente individuata da [presentationName] l'oggetto element passato nel body della richiesta
    - **input:** in header campo Authorization il token di autenticazione, nel body in formato application/json l'oggetto: element : object
    - **output:** body in formato application/json; success : boolean
  - **PUT** /private/api/presentations/[presentationName]/element
    - **descrizione:** sostituisce nella presentazione dell'utente l'elemento passato nel body della richiesta
    - **input:** in header campo Authorization il token di autenticazione, nel body in formato application/json l'oggetto: element : object
    - **output:** body in formato application/json; success : boolean
- **DeleteElement:**
  - **DELETE** /private/api/presentations/[presentationName]/delete/[type/[id_element]]
    - **descrizione:** elimina dalla presentazione con il titolo [presentationName] l'elemento con identificativo [idElement]
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean
- **ImagesMeta:**
  - **GET** /private/api/files/image
    - **descrizione:** ritorna un array con i nomi dei file immagine dell'utente
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean, names : array
- **Image:**
  - **GET** /private/api/files/image/[imagename]
    - **descrizione:** ritorna il file [imagename] nella cartella /users/[username]/images
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** file statico
  - **POST** /private/api/files/image/[imagename]

    - **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/images
    - **input:** in header campo Authorization il token di autenticazione; body in formato multipart/form-data; file da caricare
    - **output:** body in formato application/json; success : boolean
  - **DELETE** /private/api/files/image/[imagename]
    - **descrizione:** elimina il file immagine [imagename] dalla cartella /users/[username]/images
    - **input:** in header campo Authorization il token di autenticazione;
    - **output:** body in formato application/json; success : boolean

- **RenameImage:**
  - **POST** /private/api/files/image/[imagename]/[newname]
    - **descrizione:** rinomina il file immagine [imagename] con [newname] nella cartella /users/[username]/images
    - **input:** in header campo Authorization il token di autenticazione;
    - **output:** body in formato application/json; success : boolean

- **AudiosMeta:**
  - **GET** /private/api/files/audio
    - **descrizione:** ritorna un array con i nomi dei file audio dell'utente
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean, names : array

- **Audio:**
  - **GET** /private/api/files/audio/audioname]
    - **descrizione:** ritorna il file [audioname] nella cartella /users/[username]/audios
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** file statico
  - **POST** /private/api/files/audio/[audioname]
    - **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/audios
    - **input:** in header campo Authorization il token di autenticazione; body in formato multipart/form-data; file da caricare
    - **output:** body in formato application/json; success : boolean
  - **DELETE** /private/api/files/audio/[audioname]
    - **descrizione:** elimina il file audio [audioname] dalla cartella /users/[username]/audios
    - **input:** in header campo Authorization il token di autenticazione;
    - **output:** body in formato application/json; success : boolean

- **RenameAudio:**

  - **POST** /private/api/files/audio/[audioname]/[newname]
    - **descrizione:** rinomina il file audio [audioname] con [newname] nella cartella /users/[username]/audios
    - **input:** in header campo Authorization il token di autenticazione;
    - **output:** body in formato application/json; success : boolean

- **VideosMeta:**
  - **GET** /private/api/files/video
    - **descrizione:** ritorna un array con i nomi dei file video dell'utente
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** body in formato application/json; success : boolean, names : array

- **Video:**
  - **GET** /private/api/files/video/videoname]
    - **descrizione:** ritorna il file [videoname] nella cartella /users/[username]/videos
    - **input:** in header campo Authorization il token di autenticazione
    - **output:** file statico
  - **POST** /private/api/files/video/[videoname]
    - **descrizione:** caricare da locale un nuovo file immagine nella cartella /users/[username]/videos
    - **input:** in header campo Authorization il token di autenticazione; body in formato multipart/form-data; file da caricare
    - **output:** body in formato application/json; success : boolean
  - **DELETE** /private/api/files/video/[videoname]
    - **descrizione:** elimina il file video [videoname] dalla cartella /users/[username]/videos
    - **input:** in header campo Authorization il token di autenticazione;
    - **output:** body in formato application/json; success : boolean

- **RenameVideo:**
  - **POST** /private/api/files/video/[videoname]/[newname]
    - **descrizione:** rinomina il file video [videoname] con [newname] nella cartella /users/[username]/videos
    - **input:** in header campo Authorization il token di autenticazione;
    - **output:** body in formato application/json; success : boolean

5 Package Premi::Model

Tutti i package seguenti appartengono al package Premi, quindi per ognuno di essi lo scope sarà: Premi::[nome package].

Tipo, obiettivo e funzione del componente: classe astratta, base delle classi usate per rappresentare gli elementi della presentazione.
Relazioni d'uso di altre componenti: è in relazione con il package Controller e con NodeAPI.

5.1 Classe SlideShowElements

Funzione
Classe astratta, base delle classi usate per rappresentare gli elementi della presentazione.
Scope
Model::SlideShow::SlideShowElements.
Utilizzo
Contiene gli attributi e i metodi comuni degli oggetti che rappresentano gli elementi della presentazione.
Attributi

- **id**
  - **Accesso**: Private;
  - **Tipo**: Integer;
  - **Descrizione**: indica l'identificativo univoco dell'elemento.
- **yIndex**
  - **Accesso**: Private;
  - **Tipo**: Double;
  - **Descrizione**: rappresenta la posizione sull'asse delle y dell'elemento rispetto alla presentazione.
- **xIndex**
  - **Accesso**: Private;
  - **Tipo**: Double;
  - **Descrizione**: rappresenta la posizione sull'asse delle x dell'elemento rispetto alla presentazione.
- **rotation**
  - **Accesso**: Private;
  - **Tipo**: Double;

  - **Descrizione**: rappresenta il grado di rotazione dell'oggetto.

- **height**
  - **Accesso**: Private;
  - **Tipo**: Double;
  - **Descrizione**: rappresenta l'altezza dell'oggetto.

- **width**
  - **Accesso**: Private;
  - **Tipo**: Double;
  - **Descrizione**: rappresenta la larghezza dell'oggetto.

Metodi

- **setSize**(newHeight:double, newWidth:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta i campi height e width dell'oggetto.

- **setPosition**(xIndex:double, yIndex:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta i campi xIndex e yIndex dell'oggetto.

- **getSize**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array;
  - **Descrizione**: restituisce un array contenente i valori di height e width.

- **getWidth**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array;
  - **Descrizione**: restituisce un array contenente i valori di xIndex e yIndex.

- **getRotation**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Double;
  - **Descrizione**: restituisce il valore di rotation.

Ereditata da:

- Text (§5.1.1);
- Image (§5.1.2);
- Frame (§5.1.3);
- SVG (§5.1.4);
- Background (§5.1.7);
- Audio (§5.1.5);
- Video (§5.1.6).

5.1.1 Classe Text

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo testo.
Scope
Model::SlideShow::SlideShowElements::Text.
Utilizzo
Il costruttore viene invocato da Inserter::insertText().
Attributi

- **font**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il font dell'oggetto.
- **content**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il contenuto del testo.
- **color**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il colore dell'oggetto.

Metodi

- **Text**(id:integer, xIndex:double, yIndex:double, degrees:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation.

- **setFont**(newFont:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo content dell'oggetto.
- **setColor**(newColor:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo color dell'oggetto.
- **getFont**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di font.
- **getColor**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di color.
- **getContent**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di content.

5.1.2 Classe Image

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo immagine.
Scope
Model::SlideShow::SlideShowElements::Image.
Utilizzo
Il costruttore viene invocato da Inserter::insertImage().
Attributi

- **url**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il percorso dell'oggetto.

Metodi

- **Image**(id:integer, xIndex:double, yIndex:double, degrees:double, ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation e url.
- **getUrl**()
  - **Accesso**: Public;
  - **Tipo**: String;
  - **Descrizione**: restituisce il valore di url.

5.1.3 Classe Frame

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo frame.
Scope
Model::SlideShow::SlideShowElements::Frame.
Utilizzo
Il costruttore viene invocato da Inserter::insertFrame().
Attributi

- **prev**
  - **Accesso**: Private;
  - **Tipo**: Integer;
  - **Descrizione**: rappresenta l'id del frame precedente.
- **next**
  - **Accesso**: Private;
  - **Tipo**: Integer;
  - **Descrizione**: rappresenta l'id del frame successivo.
- **bookmark**
  - **Accesso**: Private;
  - **Tipo**: Bool;
  - **Descrizione**: è a 1 se il frame è un bookmark, 0 altrimenti.
- **choices**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: contiene i riferimenti agli id dei frame scelta selezionabili dal frame.

- **backgroundimage**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: contiene il riferimento dell'immagine di sfondo frame.
- **backgroundcolor**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il colore dello sfondo del frame.

Metodi

- **Frame**(id:integer, xIndex:double, yIndex:double, degrees:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation.
- **setPrev**(prevId: integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo prev.
- **setNext**(nextId: integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo next.
- **setBackgroundImage**(ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo BackgroundImage.
- **setBackgroundColor**(newColor:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo BackgroundColor.
- **isBookmark**()
  - **Accesso**: Public;

  - **Tipo di ritorno**: Bool;
  - **Descrizione**: ritorna il valore del campo bookmark.

- **setBookmark**(value: bool = 1)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo bookmark.
- **addChoice**(frameId:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: aggiunge una scelta all'array choices.
- **removeChoice**(frameId:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: rimuove una scelta dall'array choices.
- **getBackgroundImage**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di backgroundImage.
- **getBackgroundColor**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di backgroundColor.
- **getPrev**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: restituisce il valore di prev.
- **getNext**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: restituisce il valore di next.
- **getCoices**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array;
  - **Descrizione**: restituisce il valore di choiches.

5.1.4 Classe SVG

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo SVG.
Scope
Model::SlideShow::SlideShowElements::SVG.
Utilizzo
Il costruttore viene invocato da Inserter::insertSVG().
Attributi

- **color**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il colore dell'oggetto.
- **shape**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: rappresenta le coordinate della forma dell'oggetto.

Metodi

- **SVG**(id:integer, xIndex:double, yIndex:double, degrees:double, color:string, shape:array)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation, color, shape.
- **setColor**(newColor:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo color dell'oggetto.
- **setShape**(newShape:array)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo shape dell'oggetto.
- **getColor**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;

  - **Descrizione**: restituisce il valore di color.

- **getShape**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array;
  - **Descrizione**: restituisce il valore di shape.

5.1.5 Classe Audio

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo immagina.
Scope
Model::SlideShow::SlideShowElements::Audio.
Utilizzo
Il costruttore viene invocato da Inserter::insertAudio().
Attributi

- **url**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il percorso dell'oggetto.

Metodi

- **Audio**(id:integer, xIndex:double, yIndex:double, degrees:double, ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation e url.

- **getUrl**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di url.

5.1.6 Classe Video

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo video.
Scope
Model::SlideShow::SlideShowElements::Video.
Utilizzo
Il costruttore viene invocato da Inserter::insertVideo().
Attributi

- **url**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il percorso dell'oggetto.

Metodi

- **Video(id:integer, xIndex:double, yIndex:double, degrees:double, ref:string)**
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation e url.
- **getUrl**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di url.

5.1.7 Classe Background

Funzione
Classe concreta, i suoi elementi rappresentano lo sfondo.
Scope
Model::SlideShow::SlideShowElements::Background.
Utilizzo
Il costruttore viene invocato da Inserter::insertBackground().
Attributi

- **url**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il riferimento dell'immagine dello sfondo.
- **color**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il colore dello sfondo.

Metodi

- **Background**(id:integer, color:string, ref:string=”undefined”)
  - **Accesso**: Public;

  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto, imposta i campi id, xIndex, yIndex, rotation, color e url.

- **setColor**(newColor:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo color dell'oggetto.

- **setUrl**(ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo url dell'oggetto.

- **getUrl**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di url.

- **getColor**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: restituisce il valore di color.

5.2 Classe InsertEditRemove

5.2.1 Classe Inserter

Funzione
Classe statica in cui vengono implementati gli algoritmi di inserimento di elementi nella presentazione.
Scope
Model::SlideShow::SlideShowActions::InsertEditRemove.
Utilizzo
Viene utilizzata dalla classe command per eseguire i comandi di inserimento.
Attributi

- **Presentazione**
  - **Accesso**: Private;
  - **Descrizione**: oggetto json che contiene gli oggetti delle classi che rappresentano gli elementi della presentazione.

- **id =0**
  - **Accesso**: Private;
  - **Descrizione**: attributo statico, indica gli id univoci degli oggetti generati.

Metodi

- **insertText**(xIndex:double, yIndex:double, degrees:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: costruisce un oggetto newText di tipo Text (id:integer, xIndex:double, yIndex:double, degrees:double) invocandone il costruttore passando come parametri id:integer, xIndex, yIndex e degrees. Inserisce l'oggetto così costruito nell'oggetto Presentazione, copia id in un intero tempid, esegue id++ e restituisce tempid.
- **insertText**(oldText:Text)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: inserisce l'oggetto passato per parametro nell'oggetto Presentazione.
- **setUrl**(ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: imposta il campo url dell'oggetto.
- **insertFrame**(xIndex:double, yIndex:double, degrees:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: costruisce un oggetto di tipo Frame(id:integer, xIndex:double, yIndex:double, degrees:double) invocandone il costruttore passando come parametri id:integer, xIndex, yIndex e degrees. Inserisce l'oggetto così costruito nell'oggetto Presentazione, copia id in un intero tempid, esegue id++ e restituisce tempid.
- **insertImage**(xIndex, yIndex, rotation, ref)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: costruisce un oggetto di tipo Image(id:integer, xIndex:double, yIndex:double, degrees:double, ref:string) invocandone il costruttore passando come parametri id:integer, xIndex, yIndex, degrees e ref. Copia id in un intero tempid, esegue id++ e restituisce tempid.
- **insertImage**(oldImage:Image)

  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: inserisce l'oggetto passato per parametro nell'oggetto Presentazione.

- **insertSVG**(xIndex:double, yIndex:double, rotation:double, shape:string, color:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: invoca il costruttore di un oggetto SVG(id:integer, xIndex:double, yIndex:double, degrees:double, shape:string, color:string). Inserisce l'oggetto così costruito nell'oggetto Presentazione, copia id in un intero tempid, esegue id++ e restituisce tempid.

- **insertSVG**(oldSVG:SVG)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: inserisce l'oggetto passato per parametro nell'oggetto Presentazione.

- **insertAudio**(xIndex:double, yIndex:double, degrees:double, ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: invoca il costruttore di un oggetto Audio(id:integer, xIndex:double, yIndex:double, degrees:double, ref:string). Inserisce l'oggetto così costruito nell'oggetto Presentazione, copia id in un intero tempid, esegue id++ e restituisce tempid.

- **insertAudio**(oldAudio:Audio)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: inserisce l'oggetto passato per parametro nell'oggetto Presentazione.

- **insertVideo**(xIndex:double, yIndex:double, degrees:double, ref:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: invoca il costruttore di un oggetto Video(id:integer, xIndex:double, yIndex:double, degrees:double, ref:string). Inserisce l'oggetto così costruito nell'oggetto Presentazione, copia id in un intero tempid, esegue id++ e restituisce tempid.

- **insertVideo**(oldVideo:Video)
  - **Accesso**: Public;

  - **Tipo di ritorno**: Void;
  - **Descrizione**: inserisce l'oggetto passato per parametro nell'oggetto Presentazione.

- **insertBackground**(ref:string, color:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Integer;
  - **Descrizione**: invoca il costruttore di un oggetto Background(id:integer, ref:string, color:string). Inserisce l'oggetto così costruito nell'oggetto Presentazione, copia id in un intero tempid, esegue id++ e restituisce tempid.

- **insertBackground**(oldBackground:Background)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: inserisce l'oggetto passato per parametro nell'oggetto Presentazione.

- **removeText**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Text;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.

- **removeFrame**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Frame;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente, accede al suo campo prev e ne identifica i predecessore, pone prev.next=next. Rimuove l'oggetto dall'oggetto Presentazione e restituisce l'oggetto copiato.

- **removeImage**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Image;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.

- **removeSVG**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: SVG;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.

- **removeAudio**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Audio;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.
- **removeVideo**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Video;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.
- **removeBackground**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Background;
  - **Descrizione**: copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.
- **editPosition**(id:integer, tipo:string, xIndex: double, yIndex: double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array[2] di double; [[[[[[[[[[[[E' CORRETTO?]]]]]]]]]]]]
  - **Descrizione**: scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l'oggetto con il campo id corrispondente, crea una coppia oldPosition di double settati con il valore di xIndex e di yIndex dell'oggetto trovato e imposta xIndex con il valore di xIndex e yIndex con il valore di yIndex. Restituisce oldPosition.
- **editRotation**(id:integer, tipo:string, degrees)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Double;
  - **Descrizione**: scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l'oggetto con il campo id corrispondente, crea un double oldRotation settato con il valore di rotation dell'oggetto trovato e imposta rotation con il valore di degrees. Restituisce oldRotation.
- **editSize**(id:integer, tipo:string, newHeight:double, newWidth:double)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array[2] di double; [[[[[[[[[[[[E' CORRETTO?]]]]]]]]]]]];

  - **Descrizione**: scorre il campo dati che contiene gli oggetti di tipo tipo in Presentazione per trovare l'oggetto con il campo id corrispondente, crea una coppia oldSize di double settati con il valore di height e di width dell'oggetto trovato e imposta height con il valore di newHeight e width con il valore di newWidth. Restituisce oldSize.

- **editBackground**(id:integer, tipo:string, newColor:string, newRef:string="undefined")
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array[2] di double; [[[[[[[[[[[[E' CORRETTO?]]]]]]]]]]]];
  - **Descrizione**: scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l'oggetto con il campo id corrispondente. Se lo trova crea una coppia oldBackground di string settati con il valore di color e di ref dell'oggetto trovato e imposta color con il valore di newColor e ref con il valore di newRef. Restituisce oldBackground.

- **editColor**(id:integer, tipo:string, newColor:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l'oggetto con il campo id corrispondente. Se lo trova copia il campo color in una string oldColor e imposta color con il valore di newColor. Restituisce oldColor.

- **editShape**(id:integer, tipo:string, newShape:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Array di integer;
  - **Descrizione**: scorre il campo dati che contiene gli oggetti di tipo SlideShowelements in Presentazione per trovare l'oggetto con il campo id corrispondente. Se lo trova copia il campo shape in un array oldShape e imposta shape con il valore di newShape. Restituisce oldShape.

5.3 Classe Command

5.3.1 Classe Invoker

Funzione

Classe, componente invoker del Design Pattern Command. Implementato tramite design pattern singleton.

Scope

Model::SlideShow::SlideShowActions::Command::Invoker.

Utilizzo

Viene utilizzata per eseguire i comandi e memorizzarli all'interno di stack dedicate all'implementazione delle funzionalità di annulla e ripristina.

Attributi

- **undoStack**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: contiene l'elenco dei comandi eseguiti e annullabili.
- **redoStack**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: contiene l'elenco dei comandi annullati e ripristinabili.

Metodi

- **execute**(AbstractCommand)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo doAction() del comando ricevuto e invoca undoStack.push(AbstractCommand) e redoStack.clear().
- **undo**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo undoAction() dell'ultimo comando in undoStack() e invoca command=undoStack.pop() e redoStack.push(command).
- **redo**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo doAction() dell'ultimo comando in redoStack() e invoca command=redoStack.pop() e undoStack.push(command).

- **getUndoStack**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Boolean;
  - **Descrizione**: ritorna true se undoStack non è vuoto, false altrimenti.
- **getRedoStack**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Boolean;
  - **Descrizione**: ritorna true se redoStack non è vuoto, false altrimenti.

5.3.2 Classe AbstractCommand

Funzione
Classe concreta, i suoi elementi rappresentano un oggetto di tipo testo.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand.
Utilizzo
È classe base per i comandi di modifica, inserimento ed eliminazione degli elementi della presentazione.
Attributi

- **id**
  - **Accesso**: Private;
  - **Tipo**: Integer;
  - **Descrizione**: indica il codice identificativo dell'oggetto su cui viene eseguito il comando.
- **executed**
  - **Accesso**: Private;
  - **Tipo**: Boolean;
  - **Descrizione**: è settata a false di default, indica se il comando è stato eseguito.

Metodi

- **AbstractCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteTextInsertCommand e setta:
    - id=spec.id;
    - xIndex=spec.xIndex;

    - yIndex=spec.yIndex;
    - rotation=spec.rotation;
    - executed=0;

- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void; [[[[[[[[[CORRETTO??]]]]]]]]]
  - **Descrizione**: metodo virtuale implementato dalle sottoclassi. Svolge le operazioni a cui è dedicato il comando.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;[[[[[[[[[[CORRETTO??]]]]]]]]]]
  - **Descrizione**: metodo virtuale implementato dalle sottoclassi. Annulla le operazioni eseguite dal comando.

Ereditata da:

- ConcreteTextInsertCommand (§5.3.2.1);
- ConcreteFrameInsertCommand (§5.3.2.2);
- ConcreteImageInsertCommand (§5.3.2.3);
- ConcreteSVGInsertCommand (§5.3.2.4);
- ConcreteAudioInsertCommand (§5.3.2.5);
- ConcreteVideoInsertCommand (§5.3.2.6);
- ConcreteBackgroundInsertCommand (§5.3.2.7);
- ConcreteTextRemoveCommand (§5.3.2.8);
- ConcreteFrameRemoveCommand (§5.3.2.9);
- ConcreteImageRemoveCommand (§5.3.2.10);
- ConcreteSVGRemoveCommand (§5.3.2.11);
- ConcreteAudioRemoveCommand (§5.3.2.12);
- ConcreteVideoRemoveCommand (§5.3.2.13);
- ConcreteBackgroundRemoveCommand (§5.3.2.14);
- ConcreteEditSizeCommand (§5.3.2.17);
- ConcreteEditPositionCommand (§5.3.2.15);

- ConcreteEditRotationCommand (§5.3.2.16);
- ConcreteEditColorCommand (§5.3.2.20);
- ConcreteEditBackgroundCommand (§5.3.2.19);
- ConcreteEditFontCommand (§5.3.2.21).

5.3.2.1 Classe ConcreteTextInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteTextInsertCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve le coordinate di inserimento di un testo nella presentazione e invoca il metodo di Inserter insertText() passandogliele.
Metodi

- **ConcreteTextInsertCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteTextInsertCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo**: Void;
  - **Descrizione**: invoca il metodo di Inserter insertText(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo**: Void;
  - **Descrizione**: invoca il metodo di Editor removeText(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

5.3.2.2 Classe ConcreteFrameInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteFrameInsertCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve le coordinate di inserimento di un testo nella presentazione e invoca il metodo di Inserter insertFrame() passandogliele.

Metodi

- **ConcreteFrameInsertCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteFrameInsertCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Inserter insertFrame(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Editor removeFrame(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

5.3.2.3 Classe ConcreteImageInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteImageInsertCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve le coordinate di inserimento di un'immagine nella presentazione e invoca il metodo di Inserter insertImage() passandogliele.

Metodi

- **ConcreteImageInsertCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteImageInsertCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Inserter insertImage(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: invoca il metodo di Editor removeImage(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

5.3.2.4 Classe ConcreteSVGInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteSVGInsertCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve le coordinate di inserimento di un SVG nella presentazione e invoca il metodo di Inserter insertSVG() passandogliele.

Metodi

- **ConcreteSVGInsertCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteSVGInsertCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Inserter insertSVG(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Editor removeSVG(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

5.3.2.5 Classe ConcreteAudioInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteAudioInsertCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve le coordinate di inserimento di un audio nella presentazione e invoca il metodo di Inserter insertAudio() passandogliele.
Metodi

- **ConcreteAudioInsertCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteAudioInsertCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Inserter insertAudio(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Editor removeAudio(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

5.3.2.6 Classe ConcreteVideoInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteVideoInsertCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve le coordinate di inserimento di un video nella presentazione e invoca il metodo di Inserter insertVideo() passandogliele..

Metodi

- **ConcreteVideoInsertCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteVideoInsertCommand e setta xIndex, yIndex, rotation, ref.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: ivoca il metodo di Inserter insertVideo(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: invoca il metodo di Editor removeVideo(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

5.3.2.7 Classe ConcreteBackgroundInsertCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteBackgroundInsertComn
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve i parametri di inserimento di uno sfondo nella presentazione e invoca il metodo di Inserter insertBackground() passandoglie-le.
Attributi

- **ref**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta l'url dell'oggetto inserito.
- **color**
  - **Accesso**: Private;
  - **Tipo**: String;
  - **Descrizione**: rappresenta il colore dello sfondo inserito.
- **oldBackground**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Background;
  - **Descrizione**: rappresenta il vecchio sfondo della presentazione.

Metodi

- **ConcreteBackgroundInsertCommand**(ref: string, color:string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteVideoInsertCommand e setta ref e color.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

  - **Descrizione**: invoca il metodo di Inserter insertBackground (ref, color) passando come parametri i campi dati ref e color. insertBackground restituisce un int che rappresenta l'id dell'oggetto o eventualmente un oggetto di tipo Background a cui ConcreteBackgroundInsertCommand istanzia oldBackground e al cui id inizializza il campo id. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.

- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di Remover removeBackground(id) passando come parametro il campo id. Se il campo oldBackground è inizializzato invoca il metodo InsertEditRemove::insertBackground(SlideShowElement::Background) e invoca il metodo update(id) di EditController, altrimenti ne invoca il metodo remove(id).

5.3.2.8 Classe ConcreteTextRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteTextRemoveCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento testo da rimuovere dalla presentazione e invoca il metodo di Remover removeText(id).
Attributi

- **oldText**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Text;
  - **Descrizione**: è una copia dell'elemento testo rimosso.

Metodi

- **ConcreteTextRemoveCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteTextRemoveCommand e setta l'attributo id.

- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

  - **Descrizione**: invoca il metodo InsertEditRemove::removeText(id) passando come parametro l'id dell'elemento. removeText restituisce un oggetto di tipo SlideShowElements::Text a cui viene inizializzato oldText. Se executed è settato a false lo setta come true, altrimenti invoca il metodo EditController::update(id).

- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::insertText(Text) passando come parametro text. Invoca il metodo update(id) di EditController.

5.3.2.9 Classe ConcreteFrameRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteFrameRemoveCommand
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id del frame da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeFrame(id).
Attributi

- **oldFrame**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Frame;
  - **Descrizione**: è una copia dell'oggetto rimosso.

Metodi

- **ConcreteFrameRemoveCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteFrameRemoveCommand e setta il campo dati id.

- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeFrame(id) passando come parametro il campo dati id. removeFrame restituisce una copia dell'oggetto rimosso che verrà settata come campo dati oldFrame. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.

- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::insertFrame(Frame) passando come parametro il campo oldFrame. Invoca il metodo update(id) di EditController.

5.3.2.10 Classe ConcreteImageRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteImageRemoveCommand
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'immagine da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeImage(id).
Attributi

- **oldImage**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Image;
  - **Descrizione**: è una copia dell'oggetto rimosso

Metodi

- **ConcreteImageRemoveCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteImageRemoveCommand e setta il campo id.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeImage(id) passando come parametro l'id dell'oggetto da rimuovere. removeImage restituisce un oggetto di tipo SlideShowElements::Image cui sarà settato il campo oldImage. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::insertImage(Image) passando come parametro il campo oldImage. Invoca il metodo update(id) di EditController.

5.3.2.11 Classe ConcreteSVGRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteSVGRemoveCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento SVG da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeSVG().
Attributi

- **oldSVG**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::SVG;
  - **Descrizione**: è una copia dell'oggetto che rappresenta l'elemento rimosso.

Metodi

- **ConcreteSVGRemoveCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteSVGRemoveCommand e setta il campo dati id.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeSVG(id) passando come parametro il campo dati id. removeSVG restituisce un elemento di tipo SlideShowElements::SVG che rappresenta l'elemento rimosso e a cui viene inizializzato il campo oldSVG. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo di InsertEditRemove::insertSVG(SVG) passando come parametro il campo oldSVG. Invoca il metodo update(id) di EditController.

5.3.2.12 Classe ConcreteAudioRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteAudioRemoveCommand
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento audio da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeAudio(id).
Attributi

- **oldAudio**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Audio;
  - **Descrizione**: è una copia dell'oggetto che rappresenta l'elemento rimosso.

Metodi

- **ConcreteAudioRemoveCommand**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteAudioRemoveCommand e setta il campo dati id.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeAudio(id) passando come parametro il campo dati id. removeAudio restituisce una copia dell'oggetto Audio rimosso. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::insertAudio(Audio) passando come parametro il campo oldAudio. Invoca il metodo update(id) di EditController.

5.3.2.13 Classe ConcreteVideoRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteVideoRemoveCommand
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento video da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeVideo(id).
Attributi

- **oldVideo**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Video;
  - **Descrizione**: è una copia dell'oggetto che rappresenta l'elemento rimosso.

Metodi

- **ConcreteVideoRemoveCommand**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteVideoRemoveCommand e setta il campo dati id.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeVideo(id) passando come parametro il campo dati id. removeVideo restituisce una copia dell'oggetto Video rimosso. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::insertVideo(Video) passando come parametro il campo oldVideo. Invoca il metodo update(id) di EditController.

5.3.2.14 Classe ConcreteBackgroundRemoveCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteBackgroundRemoveCon
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento sfondo da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeBackground(id).
Attributi

- **oldBackground**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements:: Background;
  - **Descrizione**: è una copia dell'oggetto che rappresenta l'elemento rimosso.

Metodi

- **ConcreteBackgroundRemoveCommand**(id:integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteBackgroundRemoveCommand e setta il campo dati id.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeBackground (id) passando come parametro il campo dati id. removeBackground restituisce una copia dell'oggetto Audio rimosso. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::insertBackground (Background) passando come parametro il campo oldBackground. Invoca il metodo update(id) di EditController.

5.3.2.15 Classe ConcreteEditPositionCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditPositionCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e le coordinate x e y in cui deve essere spostato l'elemento, invoca il metodo InsertEditRemove::editPosition(spec).
Attributi

- **oldPosition**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che contiene i parametri di posizione dell'elemento modificato, contiene al suo interno i campi:
    - **id**
      - **Accesso**: Private;
      - **Tipo**: Int;
      - **Descrizione**: indica l'id dell'elemento da modificare.
    - **tipo**
      - **Accesso**: Private;
      - **Tipo**: String;
      - **Descrizione**: indica il tipo dell'elemento da modificare.
    - **xIndex**
      - **Accesso**: Private;
      - **Tipo**: Double;
      - **Descrizione**: indica la vecchia posizione sull'asse x dell'elemento.
    - **yIndex**
      - **Accesso**: Private;
      - **Tipo**: Double;
      - **Descrizione**: indica la vecchia posizione sull'asse y dell'elemento.

Metodi

- **ConcreteEditPositionCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditPositionCommand.
- **doAction**()

  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editPosition(spec) passando come parametro l'oggetto spec. editPosition ritorna un oggetto a cui ConcreteEditPositionCommand inizializza oldPoition. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.

- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editPosition(spec) passando come parametro il campo dati oldPosition. Invoca il metodo update(id) di EditController.

5.3.2.16 Classe ConcreteEditRotationCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditRotationCommand
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e il grado a cui deve essere ruotato l'elemento, invoca il metodo InsertEditRemove::editRotation(spec).
Attributi

- **oldRotation**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che contiene i parametri di rotazione dell'elemento modificato, contiene al suo interno i campi:
    - **id**
      - **Accesso**: Private;
      - **Tipo**: Int;
      - **Descrizione**: indica l'id dell'elemento da modificare.
    - **tipo**
      - **Accesso**: Private;
      - **Tipo**: String;
      - **Descrizione**: indica il tipo dell'elemento da modificare.
    - **rotation**
      - **Accesso**: Private;
      - **Tipo**: Double;

· **Descrizione**: indica il vecchio grado di rotazione dell'elemento.

Metodi

- **ConcreteEditRotationCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditRotationCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editRotation(spec) passando come parametro l'oggetto spec. editRotation ritorna un oggetto a cui ConcreteEditRotationCommand inizializza oldRotation. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editRotation(spec) passando come parametro il campo dati oldRotation. Invoca il metodo update(id) di EditController.

5.3.2.17 Classe ConcreteEditSizeCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditSizeCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e le dimensioni a cui deve essere ridimensionato l'elemento, invoca il metodo InsertEditRemove::editSize(spec).
Attributi

- **oldSize**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che contiene i parametri di dimensione dell'elemento modificato, contiene al suo interno i campi:
    * **id**
      · **Accesso**: Private;

        - **Tipo**: Int;
        - **Descrizione**: indica l'id dell'elemento da modificare.
    - **tipo**
        - **Accesso**: Private;
        - **Tipo**: String;
        - **Descrizione**: indica il tipo dell'elemento da modificare.
    - **oldHeight**
        - **Accesso**: Private;
        - **Tipo**: Double;
        - **Descrizione**: indica la vecchia altezza dell'elemento.
    - **oldWidth**
        - **Accesso**: Private;
        - **Tipo**: Double;
        - **Descrizione**: indica la vecchia larghezza dell'elemento.

Metodi

- **ConcreteEditSizeCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditSizeCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editSize(spec) passando come parametro l'oggetto spec. editSize ritorna un oggetto a cui ConcreteEditSizeCommand inizializza oldSize. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editSize(spec) passando come parametro il campo dati oldSize. Invoca il metodo update(id) di EditController.

5.3.2.18 Classe ConcreteEditContentCommand

Funzione

Classe concreta, è interfaccia del Design Pattern Command.

Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditContentCommand.

Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e il contenuto di testo che deve essere assegnato all'elemento, invoca il metodo InsertEditRemove::editContent(spec).

Attributi

- **oldContent**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che contiene i parametri di dimensione dell'elemento modificato, contiene al suo interno i campi:
    - **id**
      - **Accesso**: Private;
      - **Tipo**: Int;
      - **Descrizione**: indica l'id dell'elemento da modificare.
    - **tipo**
      - **Accesso**: Private;
      - **Tipo**: String;
      - **Descrizione**: indica il tipo dell'elemento da modificare.
    - **oldContent**
      - **Accesso**: Private;
      - **Tipo**: Double;
      - **Descrizione**: indica il vecchio contenuto dell'elemento.

Metodi

- **ConcreteEditContentCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditContentCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editContent(spec) passando come parametro l'oggetto spec. editContent ritorna un oggetto a cui ConcreteEditContentCommand inizializza oldContent. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;

- **Tipo di ritorno**: Void;
- **Descrizione**: invoca il metodo InsertEditRemove::editContent(spec) passando come parametro il campo dati oldContent. Invoca il metodo update(id) di EditController.

5.3.2.19 Classe ConcreteEditBackgroundCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditBackgroundComma
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e lo sfondo da applicargli, invoca il metodo InsertEditRemove::editBackground(spec).
Attributi

- **oldBackground**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che contiene i vecchi parametri dello sfondo dell'elemento modificato, contiene al suo interno i campi:
    - **id**
      - **Accesso**: Private;
      - **Tipo**: Int;
      - **Descrizione**: indica l'id dell'elemento da modificare.
    - **tipo**
      - **Accesso**: Private;
      - **Tipo**: String;
      - **Descrizione**: indica il tipo dell'elemento da modificare.
    - **backgroundImage**
      - **Accesso**: Private;
      - **Tipo**: String;
      - **Descrizione**: indica l'url dell'immagine di sfondo dell'elemento.
    - **backgroundColor**
      - **Accesso**: Private;
      - **Tipo**: string;
      - **Descrizione**: indica il colore dello sfondo dell'elemento.

Metodi

- **ConcreteEditBackgroundCommand**(spec)
  - **Accesso**: Public;

  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditBackgroundCommand.

- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editBackground(spec) passando come parametro l'oggetto spec. editBackground ritorna un oggetto a cui ConcreteEditBackgroundCommand inizializza oldBackground. ConcreteEditBackgroundCommand assegna quindi i campi dati id e tipo dell'oggetto oldBackground. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.

- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editBackground(spec) passando come parametro il campo dati oldBackground. Invoca il metodo update(id) di EditController.

5.3.2.20 Classe ConcreteEditColorCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditColorCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e il colore da applicargli, invoca il metodo InsertEditRemove::editColor(spec).
Attributi

- **oldColor**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che contiene i vecchi parametri dello sfondo dell'elemento modificato, contiene al suo interno i campi:
    - **id**
      - **Accesso**: Private;
      - **Tipo**: Int;
      - **Descrizione**: indica l'id dell'elemento da modificare.
    - **tipo**
      - **Accesso**: Private;

        - **Tipo**: String;
        - **Descrizione**: indica il tipo dell'elemento da modificare.
    - **color**
        - **Accesso**: Private;
        - **Tipo**: String;
        - **Descrizione**: indica il colore dell'elemento.

Metodi

- **ConcreteEditColorCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditColorCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editColor(spec) passando come parametro l'oggetto spec. editColor ritorna un oggetto a cui ConcreteEditColorCommand inizializza oldColor. ConcreteEditColorCommand assegna quindi i campi dati id e tipo dell'oggetto oldColor. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editColor(spec) passando come parametro il campo dati oldColor. Invoca il metodo update(id) di EditController.

5.3.2.21 Classe ConcreteEditFontCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditFontCommand.
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id dell'elemento dalla presentazione e lo sfondo da applicargli, invoca il metodo InsertEditRemove::editFont(spec).
Attributi

- **oldFont**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;

  - **Descrizione**: oggetto che contiene i vecchi parametri dello sfondo dell'elemento modificato, contiene al suo interno i campi:
    * **id**
      · **Accesso**: Private;
      · **Tipo**: Int;
      · **Descrizione**: indica l'id dell'elemento da modificare.
    * **tipo**
      · **Accesso**: Private;
      · **Tipo**: String;
      · **Descrizione**: indica il tipo dell'elemento da modificare.
    * **font**
      · **Accesso**: Private;
      · **Tipo**: String;
      · **Descrizione**: indica il font dell'elemento.

Metodi

- **ConcreteEditFontCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto ConcreteEditFontCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editFont(spec) passando come parametro l'oggetto spec. editFont ritorna un oggetto a cui ConcreteEditFontCommand inizializza oldFont. ConcreteEditFontCommand assegna quindi i campi dati id e tipo dell'oggetto oldFont. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::editFont(spec) passando come parametro il campo dati oldFont. Invoca il metodo update(id) di EditController.

5.3.2.22 Classe ConcreteAddToMainPathCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::concreteAddToMainPathComma
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id del frame da inserire nella presentazione e la sua posizione all'interno del percorso principale.

Metodi

- **concreteAddToMainPathCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto concreteAddToMainPathCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::addFrameToMainPath(spec) passando come parametro l'oggetto spec. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo EditController::updateMainPath().
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeFrameToMainPath(id) passando come parametro il campo id del parametro spec. Invoca il metodo updateMainPath() di EditController.

5.3.2.23 Classe concreteRemoveFromMainPathCommand

Funzione
Classe concreta, è interfaccia del Design Pattern Command.
Scope
Model::SlideShow::SlideShowActions::Command::AbstractCommand::concreteRemoveFromMainPathC
Utilizzo
Viene costruito da Premi::Controller::EditController, riceve l'id del frame da rimuovere dal percorso principale della presentazione.
Attributi

- **oldFrame**
  - **Accesso**: Private;

- **Tipo**: Oggetto;
- **Descrizione**: oggetto che contiene i vecchi parametri che indicano la posizione del frame all'interno del percorso originale prima della rimozione:
  - **id**
    - **Accesso**: Private;
    - **Tipo**: Int;
    - **Descrizione**: indica l'id del frame da rimuovere all'interno del percorso principale.
  - **pos**
    - **Accesso**: Private;
    - **Tipo**: Integer;
    - **Descrizione**: indica la posizione del frame nel percorso principale, prima dell'avvenuta rimozione.

Metodi

- **concreteRemoveFromMainPathCommand**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto concreteRemoveFromMainPathCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::removeFrameFromMainPath(spec) passando come parametro l'oggetto spec. removeFrameFromMainPath ritorna un oggetto a cui concreteRemoveFromMainPathCommand inizializza oldFrame. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo EditController::updateMainPath().
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::addFrameToMainPath(id) passando come parametro l'oggetto oldFrame. Invoca il metodo updateMainPath() di EditController.

5.3.2.24 Classe concreteNewChoicePathCommand

Funzione

Classe concreta, è interfaccia del Design Pattern Command.

Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::concreteNewChoicePathCommar

Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id del frame da cui parte il nuovo percorso scelta.

Attributi

- **PathId**
  - **Accesso**: Private;
  - **Tipo**: Int;
  - **Descrizione**: indica l'id del percorso.

Metodi

- **concreteNewChoicePathCommand**(id: integer)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto concreteNewChoicePathCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::addChoicePath(spec) passando come parametro il valore id ricevuto come parametro. addChoicePath ritorna il valore a cui viene inizializzato pathId. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo EditController::updateChoicePath(pathId).
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::deleteChoicePath(pathId) passando come parametro il valore pathId. Invoca il metodo deleteChoicePath(pathId) di EditController.

5.3.2.25 Classe concreteDeleteChoicePathCommand

Funzione

Classe concreta, è interfaccia del Design Pattern Command.

Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::concreteDeleteChoicePathComm

Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id del frame da cui parte il nuovo percorso scelta.

Attributi

- **oldPath**
  - **Accesso**: Private;

- **Tipo**: Oggetto;
- **Descrizione**: copia del percorso eliminato.

Metodi

- **concreteDeleteChoicePathCommand**(spec: object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: costruisce l'oggetto concreteDeleteChoicePathCommand.
- **doAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::deleteChoicePath(pathId) passando come parametro il valore spec.pathId. deleteChoicePath ritorna la copia del percorso rimosso, a cui viene inizializzato oldPath. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo EditController::updateChoicePath(pathId).
- **undoAction**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: invoca il metodo InsertEditRemove::addChoicePath(spec) passando come parametro l'oggetto oldPath. Invoca il metodo addChoicePath(spec) di EditController.

5.4 serverRelation

5.4.1 Loader

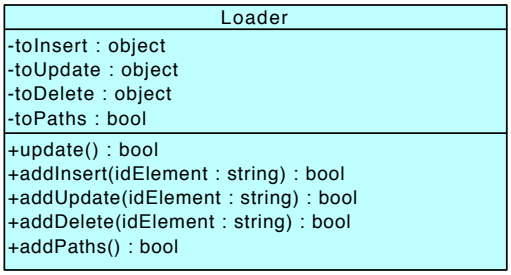

Fig 2: Diagrama classe Model::serverRelation::loader::Loader

Attributi:

- **- toInsert**
  - **Accesso:** private
  - **Tipo:** object
  - **Descrizione:** array di identificativi di elementi inseriti in locale dall'ultima sincronizzazione verso MongoDB
- **- toUpdate**
  - **Accesso:** private
  - **Tipo:** object
  - **Descrizione:** array di identificativi di elementi modificati in locale dall'ultima sincronizzazione verso MongoDB
- **- toDelete**
  - **Accesso:** private
  - **Tipo:** object
  - **Descrizione:** array di identificativi di elementi eliminati in locale dall'ultima sincronizzazione verso MongoDB
- **- toPaths**
  - **Accesso:** private
  - **Tipo:** bool
  - **Descrizione:** valore booleano che indica se è stato modificato l'oggetto paths della presentazione dall'ultima update()

Metodi:

- **update() : bool**

  - **Accessibilità:** public
  - **Descrizione:** scorre gli array toInsert, toUpdate, toDelete e chiama le funzioni in mongoRelation per la sincronizzazione della presentazione sul database MongoDB
  - **Tipo di ritorno:** bool

- **addInsert(idElement : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** inserisce la stringa parametro in toInsert
  - **Tipo di ritorno:** bool

- **addUpdate(idElement : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** inserisce la stringa parametro in toUpdate se non è presente in toInsert
  - **Tipo di ritorno:** bool

- **addDelete(idElement : string, typeObj : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** inserisce la stringa parametro in toDelete se non è presente in toInsert, altrimenti non inserisce nulla in toDelete e cancella l'identificativo da toInsert, sempre cancella da toUpdate
  - **Tipo di ritorno:** bool

- **addPaths() : bool**
  - **Accessibilità:** public
  - **Descrizione:** setta il valore del campo dati toPaths a true
  - **Tipo di ritorno:** bool

5.5 serverRelation

5.5.1 Registration

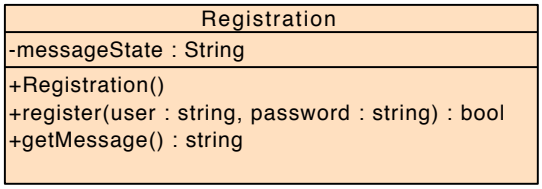

Fig 3: Diagramma classe Model::serverRelation::accessControll::Registration

Attributi:

- **-messageState**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** messaggio ritornato dall'ultima chiamata di autenticazione al server

Metodi:

- **register(user : string, password : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** chiama il servizio /account/register POST con passando i parametri attuali, scrive in messageState lo stato ritornato dalla chiamata a nodeApi, ritorna true solo se la chiamata va a buon fine
  - **Tipo di ritorno:** bool
- **getMessage() : string**
  - **Accessibilità:** public
  - **Descrizione:** ritorna il contenuto di messageState
  - **Tipo di ritorno:** string

5.5.2 Authentication

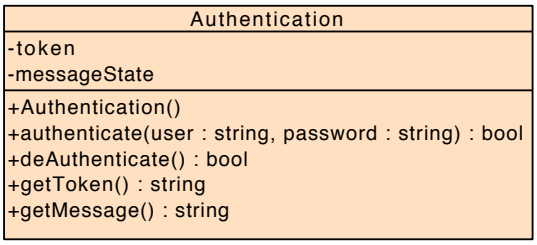

Fig 4: Diagramma classe Model::serverRelation::accessControll::Authentication

Attributi:

- **-messageState**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** messaggio ritornato dall'ultima chiamata di autenticazione al server
- **-token**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** stringa per l?autenticazione ai servizi Server nodeJs
  - **Note:** valore ?empty? se non è ancora stato richiesto il token, è stato cancellato, oppure l? autenticazione è fallita

Metodi:

- **authenticate(user : string, password : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** chiama il servizo account/authenticate, ritorna true se la chiamata è andata a buon fine, in questo caso mette nel campo token il token ritornato dal server nodeJs
  - **Tipo di ritorno:** bool
- **deAuthenticate() : bool**
  - **Accessibilità:** public
  - **Descrizione:** de-autentica l?utente dai servizi server cancellando il contenuto del campo dati token, ritorna true se l?operazione va a buon fine
  - **Tipo di ritorno:** bool
- **getToken() : string**
  - **Accessibilità:** public
  - **Descrizione:** ritorna il contenuto del campo dati token
  - **Tipo di ritorno:** string
- **getMessage() : string**
  - **Accessibilità:** public
  - **Descrizione:** ritorna il contenuto di messageState
  - **Tipo di ritorno:** string

5.6 serverRelation

5.6.1 MongoRelation

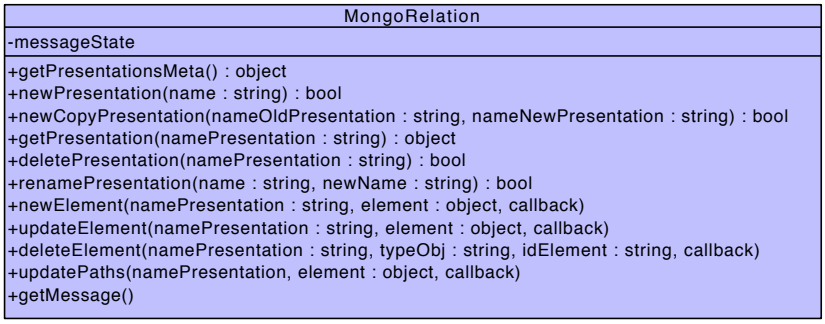

Fig 5: Diagramma classe Model::serverRelation::mongoRelation::MongoRelation

Metodi:

- **getPresentationsMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations GET ritornando l'oggetto con le informazioni sulle presentazioni create dall'utente
  - **Tipo di ritorno:** object
- **newPresentation(name : string) : void**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/new/[name] POST per creare una nuova presentazione sul database MongoDB
  - **Tipo di ritorno:** object
- **newCopyPresentation(nameOldPresentation : string, nameNewPresentation : string) : void**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/new/[nameOldPresentation]/[nameNewPresentation] POST per creare una nuova presentazione sul database MongoDB
  - **Tipo di ritorno:** object
- **getPresentation(namePresentation : string) : object**
  - **Accessibilità:** public

  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[namePresentation] GET per ricevere la presentazione [namePresentation] dell'utente
  - **Tipo di ritorno:** object

- **deletePresentation(namePresentation : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[namePresentation] DELETE per eliminare la presentazione [namePresentation] creata dall'utente dal database MongoDB
  - **Tipo di ritorno:** bool

- **renamePresentation(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[name]/rename/[newName] POST per rinominare la presentazione [name] dell'utente in [newName]
  - **Tipo di ritorno:** bool

- **updateElement(namePresentation: string, element : object, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[namePresentation]/element PUT per aggiornare l'elemento passato come parametro, esegue la funzione callback
  - **Tipo di ritorno:** bool

- **deleteElement(namePresentation: string, typeObj : string, idElement : string, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[namePresentation]/[typeObj]/[idElement] DELETE per eliminare l'elemento con identificativo [idElement] dalla presentazione nel database MongoDB, esegue la funzione callback
  - **Tipo di ritorno:** bool

- **newElement(namePresentation: string, element : object, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[namePresentation]/element POST per inserire un nuovo elemento nella presentazione dell'utente(element) nella base dati MongoDB, esegue la funzione callback

  - **Tipo di ritorno:** bool

- **updatePaths(namePresentation: string, element : object, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server nodeJs al servizio /private/api/presentations/[namePresentation]/paths PUT per aggiornare il campo paths della presentazione con l'elemento passato come parametro, esegue la funzione callback
  - **Tipo di ritorno:** bool

5.7 serverRelation

5.7.1 Loader

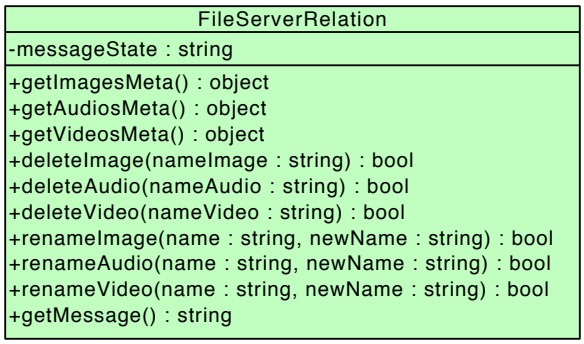

Fig 6: Diagrama classe Model::serverRelation::fileServerRelation::FileServerRelation

Attributi:

- **- messageState**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** messaggio ritornato dall'ultima chiamata al Server nodeJs

Metodi:

- **getImagesMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** ritorna un oggetto contenente informazioni sulle immagini dell'utente nel server
  - **Tipo di ritorno:** object
- **getAudiosMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** ritorna un oggetto contenente informazioni sui file audio dell'utente nel server
  - **Tipo di ritorno:** object
- **getVideosMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** ritorna un oggetto contenente informazioni sui file video dell'utente nel server
  - **Tipo di ritorno:** object

- **deleteImage(nameImage : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** cancella il file immagine nameImage dallo spazio dell'utente sul server
  - **Tipo di ritorno:** bool
- **deleteAudio(nameAudio : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** cancella il file audio nameAudio dallo spazio dell'utente sul server
  - **Tipo di ritorno:** bool
- **deleteVideo(nameVideo : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** cancella il file video nameVideo dallo spazio dell'utente sul server
  - **Tipo di ritorno:** bool
- **renameImage(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** rinomina il file immagine name dell'utente nel server con il nuovo nome newName
  - **Tipo di ritorno:** bool
- **renameAudio(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** rinomina il file audio name dell'utente nel server con il nuovo nome newName
  - **Tipo di ritorno:** bool
- **renameVideo(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** rinomina il file video name dell'utente nel server con il nuovo nome newName
  - **Tipo di ritorno:** bool

6 Specifica classi del front-end

6.1 Premi::App

Funzione

Questa classe si occuperà di fare il boostrap dell'applicazione instanziando la rootscope e iniettando tutti i moduli necessari.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Services::Utils;
- Services::Main;
- Services::toPages.

Attributi

- scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding;
- rootScope:Object
  Questo campo dati rappresenta lo scope radice dell'applicazione. Tutti gli altri scope discendono da questo;
- $routeProvider::Object
  Questo campo dati rappresenta il servizio che collega tra loro controller, view e l'URL corrente nel browser;
- $mdIconProvider
- $mdThemingProvider
- $httpProvider
- $provide
- $locationProvider

Metodi

- run;
- config;

- interceptors;
- decorator.

6.2 Package Premi::Services

Tipo, obiettivo e funzione del componente: i servizi sono degli oggetti che incapsulano del codice che si occupa di eseguire uno specifico compito, il quale sarà poi utilizzato all'interno di una o più parti dell'applicazione.
Relazioni d'uso di altre componenti: comunica con il controller per la gestione delle funzioni da eseguire e con il model per la gestione delle componenti necessarie.

6.2.1 Services::Main

Funzione
Questa classe si occuperà di eseguire le funzioni base dell'applicazione, in particolare autenticazione e registrazione al server degli utenti.
Relazioni d'uso con altri moduli
Questa classe utilizzerà le seguenti classi:

- Services::Utils;
- localStorage:Object
  Servizio angular che permette il salvataggio in locale di oggetti necessari al garantimento delle funzioni dell'applicazione.

Attributi

- **login**
  - **Accesso**: Private;
  - **Tipo**: Oggetto;
  - **Descrizione**: oggetto che mantiene la sessione corrente.

Metodi

- **value**()
  - **Accesso**: Private;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che controlla se è stato effettuato un refresh della pagina, in tal caso ripristina login.
- **register**(formData, success, error)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

  - **Descrizione**: metodo che, attraverso l'oggetto formData contenente le credenziali di accesso, effettua la registrazione al server di un nuovo utente. Se l'operazione ha successo viene invocato success altrimenti error.

- **login**(formData, success, error)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che, attraverso l'oggetto formData contenente le credenziali di accesso, effettua l'autenticazione al server di un utente. Se l'operazione ha successo viene invocato success altrimenti error.

- **logout**(success, error)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che effettua il logout dal server. Se l'operazione ha successo viene invocato success altrimenti error.

- **changepassword**(formData, success, error)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che, attraverso l'oggetto formData contenente le credenziali di accesso e la nuova password, effettua il cambio della password di un utente. Se l'operazione ha successo viene invocato success altrimenti error.

- **getToken**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: JSONWebToken;
  - **Descrizione**: metodo che ritorna il token di sessione.

- **getUser**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che ritorna l'utente attualmente autenticato con il server.

6.2.2 Services::Upload

Funzione

Questa classe si occuperà di eseguire l'upload di file media nel database.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Services::Main;

- Services::Utils;

Attributi

- **image**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: array contenente i formati immagini accettati per l'upload.
- **audio**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: array contenente i formati audio accettati per l'upload.
- **video**
  - **Accesso**: Private;
  - **Tipo**: Array;
  - **Descrizione**: array contenente i formati video accettati per l'upload.

Metodi

- **uploadmedia**(files, callback)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che invia una richiesta XMLHttpRequest a */hostname/private/api/files/-/image|audio|video/nome_file* effettuando l'upload dei file contenuti nell'array files passato come para
- **isImage**(files)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Bool;
  - **Descrizione**: metodo che ritorna true se i file contenuti nell'array files, passato come parametro, rispettano almeno uno tra i formati contenuti nell'array image, altrimenti ritorna false.
- **isAudio**(files)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Bool;
  - **Descrizione**: metodo che ritorna true se i file contenuti nell'array files, passato come parametro, rispettano almeno uno tra i formati contenuti nell'array audio, altrimenti ritorna false.

- **isVideo**(files)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Bool;
  - **Descrizione**: metodo che ritorna true se i file contenuti nell'array files, passato come parametro, rispettano almeno uno tra i formati contenuti nell'array video, altrimenti ritorna false.
- **getFileUrl**(file)
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: metodo che ritorna il percorso di salvataggio del parametro file rispetto all'utente corrente.

6.2.3 Services::Utils

Funzione

Questa classe si occuperà di eseguire piccole funzionalità utili ad ogni parte dell'applicazione.

Metodi

- **decodeToken**(token)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che decodifica token, passato come parametro, e ritorna l'oggetto utente corrispondente.
- **grade**(password)
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: metodo che determina la robustezza del parametro password. La lunghezza minima di una password è stata impostata a sei caratteri.
- **hostname**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: metodo che ritorna il dominio dell'applicazione.
- **isUndefined**(object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Bool;
  - **Descrizione**: metodo che ritorna true se il parametro object risulta indefinito, altrimento ritorna false.

- **isObject**(object)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che ritorna true se il parametro object risulta definito, altrimento ritorna false.
- **encrypt**(string)
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: metodo che ritorna il parametro string criptato. Il metodo di criptaggio scelto è lo SHA-1.

6.2.4 Services::SharedData

Funzione

Questa classe mantiene in memoria la presentazione sulla quale l'utente sta lavorando.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Services::Utils;
- Services::Main;
- localStorage:Object
  Servizio angular che permette il salvataggio in locale di oggetti necessari al garantimento delle funzioni dell'applicazione.

Attributi

- **myPresentation**
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: oggetto che rappresenta l'attuale presentazione aperta.

Metodi

- **getPresentazione**(idSlideShow)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che, nel caso in cui il parametro idSlideShow sia definito, richiama il metodo Model::ServerRelation::MongoRelation::getPresentation(idSlideShow) assegnando il risultato a myPresentation. In ogni caso myPresentation viene ritornato.

6.2.5 Services::toPages

Funzione

Questa classe si occuperà di eseguire i reindirizzamenti alle pagine corrette.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Services::Utils;
- Services::Main;
- Services::SharedData;
- $http:Object
  Servizio Angular che permette la comunicazione in remoto con un server.
- $location:Object
  Servizio Angular che gestisce gli indirizzi URL.

Metodi

- **sendRequest**(dest, success, error)
  - **Accesso**: Private;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che ritorna una richiesta http all'indirizzo definito dal parametro dest. Se l'operazione ha successo viene invocato success altrimenti error.
- **loginpage**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina di Login. Esso richiama il metodo sendRequest() il quale, se ha successo, reindirizza alla pagina richiesta.
- **registrazionepage**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina di Registrazione. Esso richiama il metodo sendRequest() il quale, se ha successo, reindirizza alla pagina richiesta.
- **homepage**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina Home. Esso richiama il metodo sendRequest() il quale, se ha successo, reindirizza alla pagina richiesta.

- **profilepage**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina Profile. Esso richiama il metodo sendRequest() il quale, se ha successo, reindirizza alla pagina richiesta.
- **editpage**(slideId)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina di Edit. Esso richiama il metodo sendRequest() il quale, se ha successo, reindirizza alla pagina richiesta e richiama il metodo SharedData.forEdit() passandogli il parametro slideId.
- **executionpage**(slideId)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina di Execution. Esso richiama il metodo sendRequest() il quale, se ha successo, reindirizza alla pagina richiesta e richiama il metodo SharedData.forEdit() passandogli il parametro slideId.

6.3 Package Premi::Controller

Tutti i package seguenti appartengono al package Premi, quindi per ognuno di essi lo scope sarà: Premi::[nome package].

Tipo, obiettivo e funzione del componente: contiene le classi che gestiscono i segnali e le chiamate effettuati dalla View.
Relazioni d'uso di altre componenti: comunica con il Model per la gestione del profilo e delle presentazioni.

6.3.1 Controller::HeaderController

Funzione
Questa classe si occuperà di controllare l'Header dell'applicazione.
Relazioni d'uso con altri moduli
Questa classe utilizzerà le seguenti classi:

- View::Pages::Index;
- Services::Utils;
- Services::Main;
- Services::toPages.

- $scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding.
- $rootScope:Object
  Questo campo dati rappresenta lo scope radice dell'applicazione. Tutti gli altri scope discendono da questo.

Metodi

- **goLogin**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::loginpage() per effettuare il reindirizzamento alla pagina di login.
- **goRegistrazione**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::registrazionepage() per effettuare il reindirizzamento alla pagina di registrazione.
- **goHome**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::homepage() per effettuare il reindirizzamento alla pagina home.
- **goProfile**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::profilepage() per effettuare il reindirizzamento alla pagina profile.
- **who**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: metodo che ritorna lo username dell'utente attualmente autenticato.
- **isToken**()
  - **Accesso**: Public;

  - **Tipo di ritorno**: Boolean;
  - **Descrizione**: metodo che verifica l'effettiva autenticazione dell'utente.

- **logout**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::Main::logout() per effettuare il logout dal server. Se l'operazione va a buon fine, viene effettuato il reindirizzamento alla pagina di login richiamando Services::toPages::loginpage().

- **error**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: String;
  - **Descrizione**: metodo che ritorna l'errore individuato; esso deve essere posto all'interno di $rootScope.error.

6.3.2 Controller::AccessController

Funzione

Questa classe si occuperà di controllare che le credenziali di accesso siano corrette nel caso dell'autenticazione oppure di registrare un nuovo utente.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Login;
- View::Pages::Registrazione;
- Model::serverRelation::Authentication;
- Model::serverRelation::Registration.
- Services::Utils;
- Services::Main;
- Services::toPages;
- $scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding.

Attributi

- **user**()

  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: oggetto contenente username e password derivanti dal form della pagina html.

- **getData**()
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: metodo che ritorna un oggetto contenente i campi dati username e password ricavati dallo $scope. La password viene criptata grazie a Services::Utils::-encrypt(password).

Metodi

- **reset**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che cancella i valori delle variabili all'interno di $scope.
- **login**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che controlla se user è definito e, in caso affermativo, richiama Services::Main::login(data) per effettuare il login al server. Se l'operazione ha successo viene effettuato il reindirizzamento alla pagina home richiamando Services::toPages::homepage().
- **registration**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che controlla se user è definito e, in caso affermativo, richiama Services::Main::register(data) per effettuare la registrazione al server. Se l'operazione ha successo viene effettuato il reindirizzamento alla pagina home richiamando Services::toPages::homepage().

6.3.3 Controller::HomeController

Funzione

Questa classe si occuperà di gestire i segnali e le chiamate provenienti dalla pagina Home.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Home;

- Model::serverRelation::MongoRelation;
- Services::Utils;
- Services::Main;
- Services::toPages;
- $scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding.

Attributi

- **mongo**()
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: oggetto che mantiene una istanza di Model::serverRelation::MongoRelation.

Metodi

- **update**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che aggiorna i contenuti della pagina home.
- **goProfile**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::profilepage() per effettuare il reindirizzamento alla pagina profile.
- **goEdit**(slideId)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::edipage(slideId) per effettuare il reindirizzamento alla pagina di edit.
- **goExecute**(slideId)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

  - **Descrizione**: metodo che richiama Services::toPages::executionpage(slideId) per effettuare il reindirizzamento alla pagina di esecuzione.

- **getSS**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che richiama Model::serverRelation::getPresentationsMeta() per visualizzare le presentazioni dell'utente corrente.

- **deleteSlideShow**(slideId)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Model::serverRelation::deletePresentation(slideId) per eliminare la presentazione slideId dal database. Se l'operazione ha successo, viene richiamato il metodo update().

- **renameSlideShow**(nameSS, rename)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Model::serverRelation::renamePresentation(nameSS, rename) per rinominare la presentazione slideId. Se l'operazione ha successo, viene richiamato il metodo update().

- **createSlideShow**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Model::serverRelation::newPresentation() per creare una nuova presentazione. Se l'operazione ha successo, viene richiamato il metodo update().

- **salvaManifest**(slideId)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Model::??? per salvare nel manifest la presentazione slideId.

6.3.4 Controller::ProfileController

Funzione

Questa classe si occupa di gestire i segnali e le chiamate provenienti dalla pagina profilo di un utente.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Model::serverRelation::??
- Services::Utils;
- Services::Main;
- Services::toPages;
- Services::Upload;
- $scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding.

Attributi

- **user**()
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: oggetto contenente username, password e nuova password derivanti dal form della pagina html.
- **getData**()
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: metodo che ritorna un oggetto contenente i campi dati username, password e newpassword ricavati dallo $scope. Le password vengono criptate grazie a Services::Utils::encrypt(password).

Metodi

- **changepassword**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che controlla se user è definito e, in caso affermativo, richiama Model::serverRelation::??
    per cambiare la password dell'utente.

6.3.5 Controller::Execution

Funzione

Questa classe si occuperà di gestire i segnali e le chiamate provenienti dalla pagina di esecuzione.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Execution;
- Services::SharedData;
- Services::Main;
- Services::toPages;
- Services::Utils;
- $scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding;
- $route:Object
  Servizio angular che permette la gestione del routing all'interno dell'applicazione.

Metodi

- **on $locationChangeSuccess**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: evento che permette l'interazione tra Angular.js e Impress.js. Dato che il framework Impress.js cambia l'url della pagina in modo dinamico, è necessario istruire il routing di Angular su come comportarsi, in modo tale che non ci sia alcun reindirizzamento inopportuno.
- **translateImpress**(json)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama una funzione JavaScript per la traduzione dell'oggetto json, passato come parametro, in html eseguibile dal framework Impress.js.
- **goHome**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::homepage() per effettuare il reindirizzamento alla pagina home.

- **goEdit**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::editpage() per effettuare il reindirizzamento alla pagina di edit.

6.3.6 Controller::EditController

Funzione

Questa classe si occuperà di mostrare all'utente la possibilità di apportare modifiche ad una presentazione.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

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

- Services::Main;
- Services::toPages;
- Services::Utils;
- Services::SaredData;
- Services::Upload;
- Model::serverRelation::MongoRelation::Loader;
- $scope:Object
  Questo campo dati rappresenta l'oggetto che permette la comunicazione tra la view ed il controller, rendendo possibile l'accesso al model mantenendolo sincronizzato, implementando in questo modo il 2-way data binding.
- $interval:Object
  servizio Angular che permette di eseguire determinate operazioni ad ogni intervallo di tempo T.
- $q::Object
  Servizio Angular che permette di eseguire funzioni in modo asincrono;
- $mdSideNav::Object
  Servizio Angular Material per il controllo della barra laterale;
- $mdBottomSheet::Object
  Servizio Angular Material per il controllo del'oggetto mdBottomsSheet.

Attributi

- **inv**()
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: oggetto che mantiene una istanza di Model::Command::Invoker.
- **mongo**()
  - **Accesso**: Private;
  - **Tipo**: Object;
  - **Descrizione**: oggetto che mantiene una istanza di Model::serverRelation::MongoRelation.

Metodi

- **translateEdit**(json)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

  - **Descrizione**: metodo che permette la traduzione dell'oggetto json, passato come parametro, in html permettendo all'utente di modificare la presentazione.

- **goExecute**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Services::toPages::executionpage() per effettuare il reindirizzamento alla pagina di execution, passando il titolo della presentazione.

- **toggleList**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che gestisce $mdBottomSheet e $mdSidenav.

- **showPathBottomSheet**($event)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che fa apparire $mdBottomSheet per la visualizzazione dei percorsi.

- **show**(id)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che gestisce la comparsa/scomparsa dei bottoni di gestione della presentazione (inserimento, rimozione, etc.) in base all'id dell'elemento html cliccato.

- **salvaPresentazione**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che richiama Model::serverRelation::Loader::update() per salvare la presentazione nel database. Questo metodo viene utilizzato in congiunta ad $interval in modo da assicurare un salvataggio continuo della presentazione ma senza creare troppo traffico in rete.

- **inserisciFrame**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

  - **Descrizione**: metodo che inserisce un frame nel piano della presentazione, attraverso la funzione View::Pages::Edit::inserisciFrame(spec), e che richiama, utilizzando il metodo execute di inv, Model::SlideShow::SlideShowActions::Command::-ConcreteFrameInsertCommand() passandogli le specifiche del frame inserito. Infine, richiama Model::serverRelation::Loader::addInsert() passandogli l'identificativo del frame inserito. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

- **inserisciTesto**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che inserisce un elemento testo nel piano della presentazione, attraverso la funzione View::Pages::Edit::inserisciTesto(spec), e che richiama, utilizzando il metodo execute di inv, Model::SlideShow::SlideShowActions::Command::-ConcreteTextInsertCommand() passandogli le specifiche dell'elemento testo inserito. Infine, richiama Model::serverRelation::Loader::addInsert() passandogli l'identificativo del testo inserito. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

- **inserisciImmagini**(files, spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che prima richiama Services::Upload::isImage(frames) per controllare che le estensioni siano corrette, successivamente Services::Upload::uploadmedia(files, callback) per l'upload dei file immagine. Se l'operazione ha successo, viene invocato callback() il quale inserisce ogni immagine nel piano della presentazione, attraverso la funzione View::Pages::Edit::inserisciImmagine(percorso$_file$, $spec$), $erichiama, utilizzandoilmetodoexecutediin$ $SlideShow :: SlideShowActions :: Command :: ConcreteImageInsertCommand()passandoglilesp$ $serverRelation :: Loader :: addInsert()passandoglil'identificativodell'immagineinserita.Nelcaso$ $Command, percui, ilmetodosioccupersolamentediaggiornarelaview.$

- **inserisciAudio**(files, spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che prima richiama Services::Upload::isAudio(frames) per controllare che le estensioni siano corrette, successivamente Services::Upload::uploadmedia(files, callback) per l'upload dei file audio. Se l'operazione ha successo, viene invocato callback() il quale inserisce ogni audio nel piano della presentazione, attraverso la funzione View::Pages::Edit::inserisciAudio(percorso$_file$, $spec$), $erichiama, utilizzandoilmetodoexecutediinv, M$ $SlideShow :: SlideShowActions :: Command :: ConcreteAudioInsertCommand()passandoglilesp$ $serverRelation :: Loader :: addInsert()passandoglil'identificativodell'audioinserito.Nelcasoincu$ $Command, percui, ilmetodosioccupersolamentediaggiornarelaview.$

- **inserisciVideo**(files, spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;

– **Descrizione**: metodo che prima richiama Services::Upload::isVideo(frames) per controllare che le estensioni siano corrette, successivamente Services::Upload::uploadmedia(files, callback) per l'upload dei file video. Se l'operazione ha successo, viene invocato callback() il quale inserisce ogni video nel piano della presentazione, attraverso la funzione View::Pages::Edit::inserisciVideo(percorso$_{f}ile, spec), $erichiama, utilizzandoilmetodoexecutediinv, M$ $SlideShow :: SlideShowActions :: Command :: ConcreteVideoInsertCommand()passandoglilspe$ $serverRelation :: Loader :: addInsert()passandoglil'identificativodelvideoinserito.Nelcasoincuiil$ $Command, percui, ilmetodosioccupersolamentediaggiornarelaview.$

- **rimuoviElemento**(spec)
  - **Accesso**: Public;
  - **Tipo di ritorno**: Void;
  - **Descrizione**: metodo che rimuove l'elemento corrente dal piano della presentazione richiamando View::Pages::Edit::elimina(id$_{e}lemento)esuccessivamente, inbasealtipodell'elemen$

    – ConcreteTextRemoveCommand;
    – ConcreteFrameRemoveCommand;
    – ConcreteImageRemoveCommand;
    – ConcreteAudioRemoveCommand;
    – ConcreteVideoRemoveCommand.

    Infine, richiama Model::serverRelation::Loader::addDelete() passandogli l'identificativo dell'elemento eliminato. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

updateSfondo(spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che viene richiamato solo in seguito ad una operazione di undo/redo e che per questo, si occupa solamente di aggiornare il background della presentazione nella view.

cambiaColoreSfondo(color)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;

- **Descrizione**: metodo che assegna al background della presentazione il valore color, passato come parametro. Successivamente richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand() passandogli le specifiche del background. Infine, richiama Model::serverRelation::Loader::-addUpdate() passandogli l'identificativo dell'elemento background modificato.

cambiaImmagineSfondo(files)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che assegna al background della presentazione un nuovo sfondo in base al parametro files. Successivamente richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand() passandogli le specifiche del background. Infine, richiama Model::serverRelation::Loader::-addUpdate() passandogli l'identificativo dell'elemento background modificato.

rimuoviSfondo()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che rimuove colore e sfondo dal background e che successivamente, utilizzando la funzione execute di inv, richiama Model::SlideShow::SlideShowActions::-Command::ConcreteBackgroundInsertCommand() passandogli le specifiche del background. Infine, richiama Model::serverRelation::Loader::addUpdate() passandogli l'identificativo dell'elemento background modificato.

updateSfondoFrame(spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che viene richiamato solo in seguito ad una operazione di undo/redo e che per questo, si occupa solamente di aggiornare il background del frame spec.id nella view.

cambiaColoreSfondoFrame(color)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che assegna al background del frame selezionato il valore color, passato come parametro. Successivamente richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand() passandogli le specifiche del background del frame. Infine, richiama Model::serverRelation::-Loader::addUpdate() passandogli l'identificativo del frame modificato.

cambiaImmagineSfondoFrame(files)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;

- **Descrizione**: metodo che assegna al background del frame selezionato un nuovo sfondo in base al parametro files. Successivamente richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand() passandogli le specifiche del background. Infine, richiama Model::serverRelation::Loader::addUpdate() passandogli l'identificativo del frame modificato.

rimuoviSfondoFrame()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che rimuove colore e sfondo dal background del frame selezionato e che successivamente, utilizzando la funzione execute di inv, richiama Model::SlideShow::SlideShowActions::Command::ConcreteEditBackgroundCommand() passandogli le specifiche del background. Infine, richiama Model::serverRelation::Loader::addUpdate() passandogli l'identificativo del frame modificato.

mediaControl()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che attiva le funzionalità per la riproduzione di un file media.

ruotaElemento(value, spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;

- **Descrizione**: metodo che ruota l'elemento selezionato in base al parametro value richiamando View::Pages::Edit::rotate(id$_elemento, value$). $Successivamenterichiama, utilizzandolafunzioneexecutedi$ $SlideShow :: SlideShowActions :: Command :: ConcreteEditRotationCommand()passandoglilespecifi$ $serverRelation :: Loader :: addUpdate()passandoglil'identificativodell'elementomodificato.Nelcasoinc$ $Command, percui, ilmetodosioccupersolamentediaggiornarelaview.$

muoviElemento(spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che, in base al nuovo posizionamento dell'elemento selezionato all'interno del piano della presentazione, richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::ConcreteEditPositionCommand() passandogli le specifiche della nuova posizione. Infine, richiama Model::serverRelation::Loader::addUpdate() passandogli l'identificativo dell'elemento modificato. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

ridimensionaElemento(spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;

- **Descrizione**: metodo che, in base alla nuova dimensione dell'elemento selezionato all'interno del piano della presentazione, richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::ConcreteEditSizeCommand(spec) passandogli le specifiche della nuova dimensione. Infine, richiama Model::serverRelation::Loader::addUpdate() passandogli l'identificativo dell'elemento modificato. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

aggiungiMainPath(spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che aggiunge il frame selezionato al percorso principale salvato in View::Pages::Edit::mainPath, e che richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::AddToMainPathCommand() passandogli le specifiche del frame da aggiungere al percorso principale. Infine, richiama Model::serverRelation::Loader::addPaths(). Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

rimuoviMainPath(spec)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che rimuove il frame selezionato dal percorso principale salvato in View::Pages::Edit::mainPath, e che richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::RemoveFromMainPathCommand() passandogli le specifiche del frame da togliere dal percorso principale. Infine, richiama Model::serverRelation::Loader::addPaths(). Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Model::Command, per cui, il metodo si occuperà solamente di aggiornare la view.

portaAvanti(id)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che aggiorna il valore zIndex dell'elemento id attraverso la funzione View::Pages::Edit::portaAvanti(id), e che richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::??? passandogli le specifiche con l'elemento da aggiornare.

portaDietro(id)

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che aggiorna il valore zIndex dell'elemento id attraverso la funzione View::Pages::Edit::mandaDietro(id), e che aggiorna il valore zIndex dell'elemento con identificativo il parametro id e che richiama, utilizzando la funzione execute di inv, Model::SlideShow::SlideShowActions::Command::??? passandogli le specifiche con l'elemento da aggiornare.

impostaPrimoSfondo()

- **Accesso**: Private;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che viene richiamato nel caso in cui la presentazione sia vuota, oppure se il valore background non è impostato. Esso richiama Model::SlideShow::SlideShowActions::Command::ConcreteBackgroundInsertCommand() passandogli le dimensioni del background. Infine, richiama Model::serverRelation::Loader::addUpdate() passandogli l'identificativo dell'elemento background modificato.

annullaModifica()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che annulla una modifica effettuata richiamando il metodo undo() di inv e che richiama uno tra i metodi addInsert, addUpdate, addDelete o addPaths di Model::serverRelation::Loader in base al tipo di azione effettuata con l'undo.

ripristinaModifica()

- **Accesso**: Public;
- **Tipo di ritorno**: Void;
- **Descrizione**: metodo che ripristina una modifica annullata richiamando il metodo redo() di inv e che richiama uno tra i metodi addInsert, addUpdate, addDelete o addPaths di Model::serverRelation::Loader in base al tipo di azione effettuata con il redo.

7 Package View

Tipo, obiettivo e funzione del componente: contiene le classi che istanzieranno gli oggetti per l'interfaccia grafica del software.
Relazioni d'uso di altre componenti: utilizza le classi contenute nel package Controller per le comunicazioni con il Model.
Attività svolte e dati trattati: rappresenta l'intera GUI del nostro sistema.

7.1 View::Pages

Tipo, obiettivo e funzione del componente: contiene le pagine in Html, rappresentano l'interfaccia grafica vera e propria.
Relazioni d'uso di altre componenti: utilizza le classi contenute nel package Controller.
Attività svolte e dati trattati: rappresenta le pagine fisiche del software.

7.2 View::Pages::Index

Funzione

Questa pagina si occuperà di mostrare all'utente header e footer validi per ogni pagina html e permettendogli di accedere alle pagine Login, Registrazione, Home e Profile e di poter effettuare il logout dal sistema .

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::HeaderController.

Input utente

- bottoneAccedi(): richiama Controller::HeaderController::goLogin() che reindirizza alla pagina View::Pages::Login;
- bottoneRegistrati(): richiama Controller::HeaderController::goRegistrazione() che reindirizza alla pagina View::Pages::Registrazione;
- bottoneHome(): richiama Controller::HeaderController::goHome() che reindirizza alla pagina View::Pages::Home;
- bottoneProfilo(): richiama Controller::HeaderController::goProfile() che reindirizza alla pagina View::Pages::Profilo;

7.3 View::Pages::Login

Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di effettuare il login.

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::AccessController.

Input utente

- bottoneLogin(): attiva il metodo Controller::AccessController::login() che controlla se i campi della form sono stati compilati correttamente. Se l'operazione ha successo, viene effettuato il reindirizzamento alla pagina View::Pages::Home;

7.4 View::Pages::Registrazione

Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di effettuare la registrazione al sistema.

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::AccessController.

Input utente

- bottoneRegistrati(): attiva il metodo Controller::AccessController::registration() che controlla se i campi della form sono stati compilati correttamente. Se l'operazione ha successo, viene effettuato il reindirizzamento alla pagina View::Pages::Home;

7.5 View::Pages::Home

Funzione

Questa pagina si occuperà di mostrare all'utente le presentazioni presenti sul proprio database dando la possibilità di eliminarle, eseguirle, scaricarle in locale, rinominarle, modificarle o crearne di nuove.

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::HomeController

Input utente

- bottoneNuovaPresentazione(): bottone che richiama il metodo di Controller::HomeController::createS
- bottoneElimina(): bottone che richiama il metodo di Controller::HomeController::deleteSlideShow() passandogli il nome della presentazione da eliminare;
- bottoneRinomina(): bottone che richiama il metodo di Controller::HomeController::renameSlideShow passandogli il nome della presentazione da rinominare;
- bottoneEsegui(): bottone che richiama il metodo di Controller::HomeController::goExecute() passandogli il nome della presentazione da eseguire;
- bottoneEdit(): bottone che richiama il metodo di Controller::HomeController::goEdit() passandogli il nome della presentazione da modificare;
- bottoneSalva(): bottone che richiama il metodo di Controller::HomeController::salvaManifest() passandogli il nome della presentazione da salvare in locale.

7.6 View::Pages::Profile

Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di cambiare i propri dati personali.

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::ProfileController

Input utente

- bottoneCambiaPassword(): bottone che richiama il metodo di Controller::ProfileController::changePa
  Il risultato dell'operazione viene in ogni caso comunicato all'utente.

7.7 View::Pages::Execution

Funzione

Questa pagina si occuperà di gestire l'esecuzione di una presentazione utilizzando il framework Impress.js associato alla pagina.

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::ExecutionController

Input utente

- +next(): metodo che viene invocato premendo il tasto freccia destra e che invoca a sua volta il metodo impress().next() implementato all'interno del framework Impress.js e che visualizzerà il frame successivo della presentazione;
- +prev(): metodo che viene invocato premendo il tasto freccia sinistra e che invoca a sua volta il metodo impress().prev() implementato all'interno del framework Impress.js e che visualizzerà il frame precedente della presentazione;
- +bookmark(): metodo che viene invocato premendo il tasto barra spaziatrice e che invoca a sua volta il metodo impress().bookmark() implementato all'interno del framework Impress.js e che visualizzerà il frame con bookmark successivo.

7.8 View::Pages::Edit

Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di apportare modifiche ad una presentazione.

Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::EditController

Attributi

- active: oggetto che rappresenta l'elemento attualmente selezionato;
- mainPath: oggetto che rappresenta il percorso principale della presentazione.

Input utente

- bottoneEseguiPresentazione(): bottone che richiama Controller::EditController::goExecute() per eseguire la presentazione;
- bottoneAnnulla: bottone che richiama Controller::EditController::annullaModifica() per annullare l'ultima modifica eseguita;
- bottoneRipristina: bottone che richiama Controller::EditController::ripristinaModifica() per ripristina l'ultima modifica annullata;
- bottoneInserisciFrame: bottone che richiama Controller::EditController::inserisciFrame() per inserire un frame nel piano della presentazione;
- bottoneInserisciTesto: bottone che richiama Controller::EditController::inserisciTesto() per inserire un elemento testo nel piano della presentazione;
- bottoneInserisciImmagine: bottone che richiama Controller::EditController::inserisciImmagine() passandogli le immagini da inserire;
- bottoneInserisciAudio: bottone che richiama Controller::EditController::inserisciAudio() passandogli gli audio da inserire;
- bottoneInserisciVideo: bottone che richiama Controller::EditController::inserisciVideo() passandogli i video da inserire;
- bottoneRuota: bottone che richiama Controller::EditController::ruotaElemento() passandogli il valore della rotazione da applicare all'elemento selezionato;
- bottoneCambiaColoreSfondo: bottone che richiama Controller::EditController::cambiaColoreSfondo(passandogli il valore del colore da applicare al background della presentazione;
- bottoneCambiaImmagineSfondo: bottone che richiama Controller::EditController::cambiaImmagineS passandogli l'immagine da applicare al background della presentazione;
- eliminaSfondoPresentazine: bottone che richiama Controller::EditController::rimuoviSfondo() per resettare lo sfondo della presentazione;
- bottoneCambiaColoreSfondoFrame: bottone che richiama Controller::EditController::cambiaColoreSf passandogli il valore del colore da applicare al frame selezionato;
- bottoneCambiaImmagineSfondoFrame: bottone che richiama Controller::EditController::cambiaImm passandogli l'immagine da applicare al background del frame selezionato;
- eliminaSfondoFrame: bottone che richiama Controller::EditController::rimuoviSfondo() per resettare lo sfondo del frame selezionato;

- bottoneAggiungiPercorsoPrincipale: bottone che richiama Controller::EditControlelr::aggiungiMainP per aggiungere il frame corrente al percorso principale di presentazione;
- inserisciFrame(spec): funzione javascript che permette l'inserimento di un nuovo frame. Se spec è definito, le proprietà contenute in esso vengono assegnate al frame appena inserito;
- inserisciTesto(spec): funzione javascript che permette l'inserimento di un nuovo elemento testo. Se spec è definito, le proprietà contenute in esso vengono assegnate al testo appena inserito;
- inserisciImmagine(x, spec): funzione javascript che permette l'inserimento di un nuovo elemento immagine con path passato tramite il parametro x. Se spec è definito, le proprietà contenute in esso vengono assegnate all'immagine appena inserita;
- inserisciAudio(x, spec): funzione javascript che permette l'inserimento di un nuovo elemento audio con path passato tramite il parametro x. Se spec è definito, le proprietà contenute in esso vengono assegnate all'audio appena inserito;
- inserisciVideo(x, spec): funzione javascript che permette l'inserimento di un nuovo elemento video con path passato tramite il parametro x. Se spec è definito, le proprietà contenute in esso vengono assegnate al video appena inserito;
- elimina(id): funzione javascript che permette l'eliminazione dell'elemento id dal piano della presentazione;
- rotate(el, value): funzione javascript che permette di ruotare l'elemento el in base al valore definito dal parametro value;
- portaAvanti(id): funzione javascript che incrementa la proprietà zIndex dell'elemento id;
- mandaDietro(id): funzione javascript che decrementa la proprietà zIndex dell'elemento id;
- DragDrop: evento javascript che permette di poter spostare un elemento all'interno del piano della presentazione;
- Resizable: evento javascript che permette di poter ridimensionare un elemento all'interno del piano della presentazione;
- bottoneTextEdit(): bottone che modifica il testo selezionato e attiva il metodo in Controller::EditController che si occupa dell'aggiornamento dell'elemento testuale modificato;
- bottoneInsertChoice(): bottone che fa inserire all'utente il testo della scelta e fa scegliere il frame al quale farà riferimento, attiva il metodo di Controller::EditController che si occuperà dell'aggiornamento della presentazione;
- bottoneBookmark(): bottone che assegna o rimuove il bookmark al frame, attiva il metodo di Controller::EditController che si occuperà dell'aggiornamento della presentazione;

- bottoneInsertSvg(): bottone che fa comparire nel piano della presentazione il nuovo elemento svg selezionato e attiva il metodo in Controller::EditController che si occuperà di aggiornare le informazioni della presentazione;

8 Tracciamento

8.1 tracciamento requisiti-classi

8.2 tracciamento metodi-test

Classe e Metodo	Test
Premi::Services::Main	TU x
Premi::Services::Upload	
Premi::Services::Utils	
Premi::Services::SharedData	
Premi::Services::ToPages	
Premi::HeaderController	TU
Premi::AccessController	TU
Premi::HomeController	TU
Premi::ProfileController	TU
Premi::ExecutionController	TU
Premi::EditController	TU

Tab 3: Metodi-Test