



Definizione di Prodotto

Gruppo SWEet BIT – Progetto SWEDesigner

Informazioni sul documento

Versione	2.0.0
Redazione	Massignan Fabio Bertolin Sebastiano Salmistraro Gianamarco
Verifica	Pilò Salvatore
Approvazione	Santimaria Davide
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Zucchetti S.p.A.

Descrizione

Questo documento descrive la struttura e le relazioni tra le parti del prodotto SWEDesigner del gruppo SWEet BIT.

Registro delle modifiche

ver: moi, seba, gian appr: fabio davide scrivere: fabio seba davide gian

Versione	Data	Persone coinvolte	Descrizione
1.0.0	2017/07/02	Santimaria Davide	Approvazione documento
0.1.4	2017/06/30	Pilò Salvatore	Verfica documento
0.0.4	2017/06/25	Salmistraro Gianmarco	Stesura Front-End
0.0.3	2017/06/10	Bertolin Sebastiano	Stesura Back-End
0.0.2	2017/06/08	Massignan Fabio	Stesura introduzione e scheletro capitoli iniziali
0.0.1	2017/06/08	Massignan Fabio	Stesura scheletro documento

Indice

1	Introduzione	9
1.1	Scopo del documento	9
1.2	Scopo del prodotto	9
1.3	Glossario	9
1.4	Riferimenti	9
1.4.1	Normativi	9
1.4.2	Informativi	10
1.5	Descrizione dell'architettura	11
2	Standard di progetto	12
2.1	Standard di progettazione architetturale	12
2.2	Standard di documentazione del codice	12
2.3	Standard di denominazione di entità e relazioni	12
2.4	Standard di programmazione	12
2.5	Strumenti di lavoro	12
3	Specifica Front-End	13
3.1	SWEDesigner::Client	13
3.1.1	Informazioni generali	13
3.1.2	Classi	13
3.1.2.1	SWEDesigner::AuthenticationGuard	13
3.1.2.2	SWEDesigner::Global	14
3.2	SWEDesigner::Client::Components	17
3.2.1	Informazioni generali	17
3.2.2	Classi	18
3.2.2.1	SWEDesigner::Client::Components::RegistrationComponent	18
3.2.2.2	SWEDesigner::Client::Components::LoginComponent	19
3.2.2.3	SWEDesigner::Client::Components::Forgot-pswComponent	19
3.3	SWEDesigner::Client::Components::Editor-container	20
3.3.1	Informazioni generali	20
3.3.2	Classi	21
3.3.2.1	SWEDesigner::Client::Components::Editor-container::Editor-containerComponent	21
3.3.2.2	SWEDesigner::Client::Components::Editor-container::Activity-frameComponent	21
3.4	SWEDesigner::Client::Components::Editor-container::Menu	21
3.4.1	Informazioni generali	21
3.4.2	Classi	22
3.4.2.1	SWEDesigner::Client::Components::Editor-container::Menu::MenuComponent	22
3.4.2.2	SWEDesigner::Client::Components::Editor-container::Menu::FileComponent	22
3.4.2.3	SWEDesigner::Client::Components::Editor-container::Menu::ModificaComponent	22

	3.4.2.4	SWEDesigner::Client::Components::Editor-container::Menu::ProfiloComponent	25
	3.4.2.5	SWEDesigner::Client::Components::Editor-container::Menu::ProgettoComponent	25
3.5		SWEDesigner::Client::Components::Editor-container::Editor	25
	3.5.1	Informazioni generali	25
	3.5.2	Classi	26
	3.5.2.1	SWEDesigner::Client::Components::Editor-container::Editor::EditorComponent	26
	3.5.2.2	SWEDesigner::Client::Components::Editor-container::Editor::Activity-menuComponent	31
	3.5.2.3	SWEDesigner::Client::Components::Editor-container::Editor::ToolbarComponent	31
	3.5.2.4	SWEDesigner::Client::Components::Editor-container::Editor::ActivityService	36
	3.5.2.5	SWEDesigner::Client::Components::Editor-container::Editor::Class-menuService	42
	3.5.2.6	SWEDesigner::Client::Components::Editor-container::Editor::All-shape	47
	3.5.2.7	SWEDesigner::Client::Components::Editor-container::Editor::Attributo	49
	3.5.2.8	SWEDesigner::Client::Components::Editor-container::Editor::Class-errors	50
	3.5.2.9	SWEDesigner::Client::Components::Editor-container::Editor::Classe-astratta	50
	3.5.2.10	SWEDesigner::Client::Components::Editor-container::Editor::Classe	51
	3.5.2.11	SWEDesigner::Client::Components::Editor-container::Editor::Elemento-metodo	55
	3.5.2.12	SWEDesigner::Client::Components::Editor-container::Editor::End	55
	3.5.2.13	SWEDesigner::Client::Components::Editor-container::Editor::If-node	56
	3.5.2.14	SWEDesigner::Client::Components::Editor-container::Editor::Interface	57
	3.5.2.15	SWEDesigner::Client::Components::Editor-container::Editor::Merge-node	58
	3.5.2.16	SWEDesigner::Client::Components::Editor-container::Editor::Metodo	59
	3.5.2.17	SWEDesigner::Client::Components::Editor-container::Editor::Operation	63
	3.5.2.18	SWEDesigner::Client::Components::Editor-container::Editor::Operazione	64
	3.5.2.19	SWEDesigner::Client::Components::Editor-container::Editor::Param	64
	3.5.2.20	SWEDesigner::Client::Components::Editor-container::Editor::Shape	65
	3.5.2.21	SWEDesigner::Client::Components::Editor-container::Editor::Start	68
	3.5.2.22	SWEDesigner::Client::Components::Editor-container::Editor::Variabile	68
	3.5.2.23	SWEDesigner::Client::Components::Editor-container::Editor::While-node	70
3.6		SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu	71
	3.6.1	Informazioni generali	71
	3.6.2	Classi	71
	3.6.2.1	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Edit-class-menuComponent	71

3.6.2.2	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Change-class-nameComponent	71
3.6.2.3	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-add-attributeComponent	72
3.6.2.4	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-add-main-methodComponent	73
3.6.2.5	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-add-methodComponent	74
3.6.2.6	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-list-attributeComponent	75
3.6.2.7	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-list-methodComponent	77
3.6.2.8	SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Display-class-nameComponent	78
3.7	SWEDesigner::Client::Services	78
3.7.1	Informazioni generali	78
3.7.2	Classi	79
3.7.2.1	SWEDesigner::Client::Services::AccountService	79
3.7.2.2	SWEDesigner::Client::Services::Main-editorService	83
3.7.2.3	SWEDesigner::Client::Services::MenuService	89
4	Specifica Back-End	93
4.1	SWEDesigner::Server	93
4.1.1	Informazioni generali	93
4.1.2	Classi	93
4.1.2.1	SWEDesigner::Server::serverLoader	93
4.2	SWEDesigner::Server::Model	94
4.2.1	Informazioni generali	94
4.2.2	Classi	94
4.2.2.1	SWEDesigner::Server::Model::mongooseConnection	94
4.2.2.2	SWEDesigner::Server::Model::mongooseRequest	95
4.3	SWEDesigner::Server::Controller::Middleware	101
4.3.1	Informazioni generali	101
4.3.2	Classi	101
4.3.2.1	SWEDesigner::Server::Controller::Middleware::midLoader	101
4.3.2.2	SWEDesigner::Server::Controller::Middleware::Parse	101
4.3.2.3	SWEDesigner::Server::Controller::Middleware::Encrypt	102
4.4	SWEDesigner::Server::Controller::Services	104
4.4.1	Informazioni generali	104
4.4.2	Classi	104
4.4.2.1	SWEDesigner::Server::Controller::Services::parseService	104
4.4.2.2	SWEDesigner::Server::Controller::Services::encryptService	105
5	Diagrammi di sequenza	107

5.1	Generazione Codice	107
5.2	Caricamento moduli del Server	107
5.3	Encrypt/Decrypt	108
6	Tracciamento	109
6.1	Tracciamento Classi-Requisiti	109
6.2	Tracciamento Requisiti-Classi	113

Elenco delle figure

1	Diagramma della classe AuthenticationGuard	13
2	Diagramma della classe Global	14
3	Diagramma della classe RegistrationComponent	18
4	Diagramma della classe LoginComponent	19
5	Diagramma della classe Forgot-pswComponent	20
6	Diagramma della classe FileComponent	22
7	Diagramma della classe ModificaComponent	24
8	Diagramma della classe ProfiloComponent	24
9	Diagramma della classe EditorComponent	26
10	Diagramma della classe Activity-menu	31
11	Diagramma della classe ToolbarComponent	35
12	Diagramma della classe ActivityService	37
13	Diagramma della classe Class-menu	43
14	Diagramma della classe All-shape	47
15	Diagramma della classe Attributo	49
16	Diagramma della classe Classe-astratta	50
17	Diagramma della classe Classe	52
18	Diagramma della classe End	55
19	Diagramma della classe If-node	56
20	Diagramma della classe Interface	57
21	Diagramma della classe Merge-node	59
22	Diagramma della classe Metodo	60
23	Diagramma della classe Operation	63
24	Diagramma della classe Param	64
25	Diagramma della classe Shape	66
26	Diagramma della classe Start	68
27	Diagramma della classe Variabile	69
28	Diagramma della classe While-node	70
29	Diagramma della classe Change-class-name	71
30	Diagramma della classe Class-add-attribute	72
31	Diagramma della classe Class-add-main-method	73
32	Diagramma della classe Class-add-method	74
33	Diagramma della classe Class-list-attribute	75
34	Diagramma della classe Class-list-attribute	76
35	Diagramma della classe Class-list-method	77
36	Diagramma della classe Display-class-name	78
37	Diagramma della classe AccountService	79
38	Diagramma della classe Main-editorService	84
39	Diagramma della classe MenuService	90
40	Diagramma della classe SWEDesigner::Server::serverLoader	93
41	Diagramma della classe SWEDesigner::Server::Model::mongooseConnection	95

42	Diagramma della classe SWEDesigner::Server::Model::mongooseRequest .	95
43	Diagramma della classe SWEDesigner::Server::Controller::Middleware::midLoader	101
44	Diagramma della classe SWEDesigner::Server::Controller::Middleware::Parse	101
45	Diagramma della classe SWEDesigner::Server::Controller::Middleware::Encrypt	102
46	Diagramma della classe SWEDesigner::Server::Controller::Services::parseService	104
47	Diagramma della classe SWEDesigner::Server::Controller::Services::encryptService	105
48	Sequence diagram generazione codice java	107
49	Sequence diagram caricamento moduli server	108
50	Sequence diagram per operazioni di encrypt e decrypt	109

Elenco delle tabelle

2	Tracciamento Classi - Requisiti	112
3	Tracciamento Requisiti - Classe	115

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire in dettaglio la struttura e il funzionamento delle componenti del prodotto SWEDesigner. Questo documento servirà come guida per i componenti del gruppo fornendo direttive e vincoli per la realizzazione del progetto.

1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di una *Web App_G* che fornisca all'*Utente_G* un *UML_G Designer_G* con il quale riuscire a disegnare correttamente *Diagrammi_G* delle *Classi_G* e descrivere il comportamento dei *Metodi_G* interni alle stesse attraverso l'utilizzo di *Diagrammi_G* delle attività. La *Web App_G* permetterà all'*Utente_G* di generare *Codice_G Java_G* dall'insieme dei *diagrammi classi_G* e dei rispettivi *metodi_G*.

1.3 Glossario

Con lo scopo di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v4.0.0*. In esso saranno definiti, in modo chiaro e conciso i termini che possono causare ambiguità o incomprensione del testo.

1.4 Riferimenti

1.4.1 Normativi

- **Capitolato d'Appalto C6: SWEDesigner**
<http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6p.pdf>;
- **Norme di Progetto:** *Norme di Progetto v4.0.0*.
- **Analisi dei Requisiti:** *Analisi dei Requisiti v4.0.0*.

1.4.2 Informativi

- Slide dell'insegnamento Ingegneria del Software modulo A:
<http://www.math.unipd.it/~tullio/IS-1/2016/>.
 - Slides del corso di Ingegneria del Software mod. A: *Diagrammi delle classi_G*: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E03.pdf>;
 - Slides del corso di Ingegneria del Software mod. A: Diagrammi dei package: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E04.pdf>;
 - Slides del corso di Ingegneria del Software mod. A: Diagrammi di sequenza: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E05.pdf>;
 - Slides del corso di Ingegneria del Software mod. A: Diagrammi di attività: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E06.pdf>;
 - Slides del corso di Ingegneria del Software mod. A: *Design pattern_G* strutturali: Decorator, Proxy, Facade, Adapter: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E07.pdf>;
 - Slides del corso di Ingegneria del Software mod. A: *Design pattern_G* creazionali: Singleton, Builder, Abstract Factory: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E08.pdf>;
 - Slides del corso di Ingegneria del Software mod. A: *Design pattern_G* comportamentali: Observer, Template Method, Command, Strategy, Iterator: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E09.pdf>;
- Design Patterns - E. Gamma, R. Helm, R. Johnson, J. Vlissides (Pearson Education, Addison-Wesley, 1995);
- *Node.js_G*: <https://nodejs.org/dist/latest-v6.x/docs/api/>;
- MongoDB: <https://docs.mongodb.org/manual/>;
- HTML5: http://www.w3schools.com/html/html5_intro.asp;
- CSS3: http://www.w3schools.com/css/css3_intro.asp;
- ExpressJS: <http://expressjs.com/en/4x/api.html>.
- Mustache: <http://mustache.github.io/>.

1.5 Descrizione dell'architettura

È doveroso soffermarsi, in questa sezione del documento, sulla descrizione generale dell'architettura utilizzata all'interno del progetto per via di alcune soluzioni "esotiche" adottate in fase di sviluppo.

L'architettura utilizzata segue, per quanto possibile, quella di Angular4 per quanto riguarda il Client anche se presenta qualche "anomalia" nel patter MVVM offerto dal Framework.

Si è scelto di inserire il Model e il Controller nel Back-End così da gestire tutte le operazioni di interaccia con il Database solo lato server alleggerendo quindi il client dal carico che le varie chiamate offrono.

Nonostante questo, è presente, seppur in maniera "nascosta", un Model e una View-Model all'interno del Client.

Ogni Component infatti fa sia da View che da View-Model mentre ogni servizio offre le funzionalità di un Model, quindi si occupa di gestire le richieste al Server.

Sul Server ogni richiesta è gestita da Express.js e dalle sue funzioni di routing contenute all'interno del file *index.js* nel quale è presente il caricamento di ogni componenete del Server e la gestione di tutte le funzioni di routing necessarie al corretto funzionamento di ogni servizio.

Al primo avvio verranno caricati tutti i servizi di Middleware e inizializzati tutti i parametri necessari al corretto funzionamento del Server.

Per ogni richiesta in arrivo dai servizi del Client, Express.js si occuperà di istanziare un servizio, tramite patter Factory, che gestisce la richiesta e restituisce una risposta, generalmente *true* o *false*, al servizio sul Client che ha effettuato la richiesta.

2 Standard di progetto

2.1 Standard di progettazione architettuale

Gli standard di progettazione sono definiti *Specifica Tecnica v 3.0.0*.

2.2 Standard di documentazione del codice

Gli standard per la scrittura della documentazione del codice sono definiti nelle *Norme di Progetto 4.0.0*.

2.3 Standard di denominazione di entità e relazioni

Tutti gli elementi definiti come package, classi, metodi o attributi, devono avere denominazioni chiare ed esplicative. Il nome deve avere una lunghezza tale da non pregiudicarne la leggibilità e chiarezza. È preferibile utilizzare dei sostantivi per le entità e dei verbi per le relazioni. Le abbreviazioni sono ammesse se:

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

Le regole tipografiche relative ai nomi delle entità sono definite nelle *Norme di Progetto v4.0.0*.

2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle *Norme di Progetto v4.0.0*.

2.5 Strumenti di lavoro

Per gli strumenti di lavoro da utilizzare durante la codifica e le procedure per il loro corretto funzionamento e coordinamento si rimanda al documento *Norme di Progetto v4.0.0*.

3 Specifica Front-End

3.1 SWEDesigner::Client

3.1.1 Informazioni generali

- **Descrizione:**
Questo package racchiude tutta la componente di Front-end scritta in TypeScript. Gli attributi e i metodi di alcune classi saranno definiti a partire dalla prossima versione.
- **Padre:** SWEDesigner
- **Package contenuti:**
 - Components
Questo package contiene tutti i components dell'applicazione
 - Services
Questo package contiene i servizi per le operazioni di iterazione tra i components e il server

3.1.2 Classi

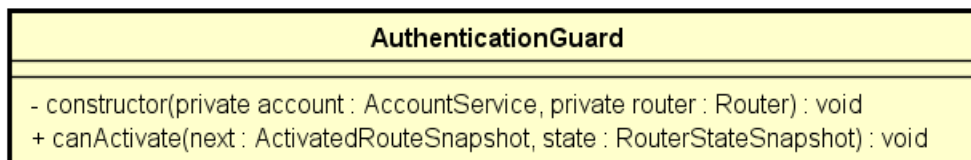


Figura 1: Diagramma della classe AuthenticationGuard

3.1.2.1 SWEDesigner::AuthenticationGuard

- **Descrizione:**
 - **Utilizzo:**
 - **Metodi:**
 - *-constructor(private account: AccountService, private router: Router)*
Costruttore della classe
- Parametri:**

- * *account: AccountService*
Crea un istanziazione di AccountService
- * *router: Router*
Crea un istanziazione di Router
- *+canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot)*

Parametri:

- * *next: ActivatedRouteSnapshot*
Un metodo dell'interfaccia
- * *state: RouterStateSnapshot*
Un booleano dell'interfaccia

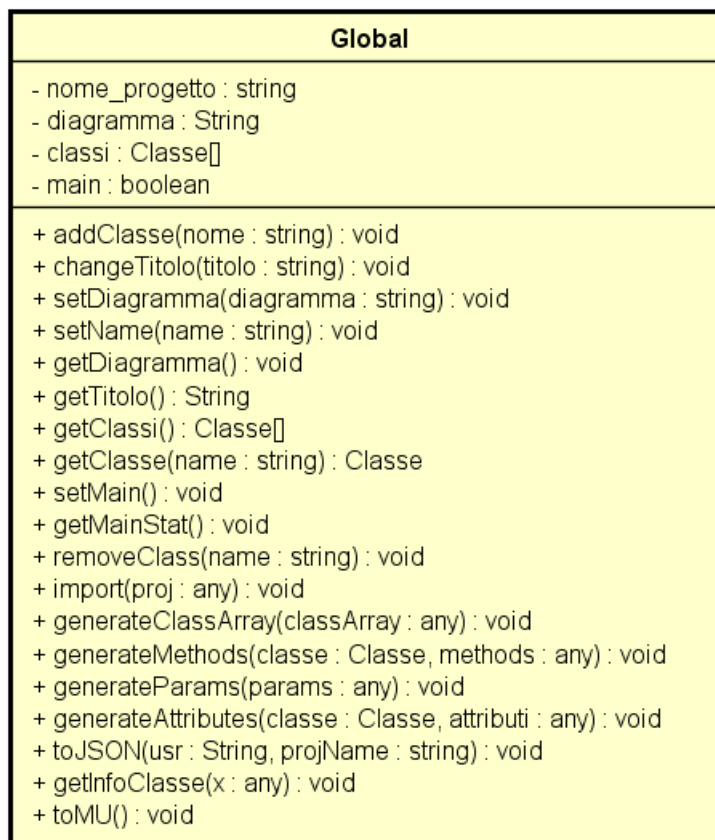


Figura 2: Diagramma della classe Global

3.1.2.2 SWEDesigner::Global

- **Descrizione:**
- **Utilizzo:**
- **Attributi:**
 - *-nome_progetto: string*
Memorizza il nome del progetto
 - *-diagramma: string*
Memorizza il diagramma
 - *-classi: Classe[]*
Memorizza le classi dell'array
 - *-main: boolean*
True se il metodo main presente
- **Metodi:**
 - *+addClasse(nome: string)*
Crea una nuova classe
Parametri:
 - * *nome: string*
Nome della classe
 - *+changeTitolo(titolo: string)*
Modifica il nome del progetto
Parametri:
 - * *titolo: string*
Nome del progetto
 - *+setDiagramma(diagramma: string)*
Resetta il diagramma
Parametri:
 - * *diagramma: string*
Diagramma da resettare
 - *+setName(name: string)*
Modifica il nome della classe selezionata
Parametri:
 - * *name: string*
Nuovo nome della classe

- *+getDiagramma()*
Ritorna la stringa diagramma
- *+getTitolo()*
Ritorna il nome del progetto
- *+getClassi()*
Ritorna l'array delle classi
- *+getClasse(name: string)*
Ritorna la classe
Parametri:
 - * *name: string*
Nome della classe da ritornare
- *+setMain()*
Setta a true l'attributo main
- *+getMainStat()*
Ritorna il valore dell'attributo main
- *+removeClass(name: string)*
Rimuove una classe dall'array
Parametri:
 - * *name: string*
Nome della classe da rimuovere
- *+import(proj: any)*
Importa il progetto
Parametri:
 - * *proj: any*
Progetto da importare
- *+generateClassArray(classArray: any)*
Genera la stringa con le informazioni della classe
Parametri:
 - * *classArray: any*
Array della classe
- *+generateMethods(classe: Classe, methods: any)*
Genera la stringa dei metodi
Parametri:
 - * *classe: Classe*
Classe di cui generare i metodi

- * *methods: any*

- Metodo

- *+generateParams(params: any)*

- Genera la stringa dei parametri

- Parametri:**

- * *params: any*

- Lista dei parametri

- *+generateAttributes(classe: Classe, attributi: any)*

- Genera la stringa degli attributi

- Parametri:**

- * *classe: Classe*

- Classe

- * *attributi: any*

- Lista di attributi

- *+toJSON(usr: String, projName: string)*

- Trasforma il progetto in un file JSON

- Parametri:**

- * *usr: String*

- Username dell'utente

- * *projName: string*

- Nome del progetto

- *+getInfoClasse(x: any)*

- Ritorna tutte le informazioni della classe

- Parametri:**

- * *x: any*

- Classe

- *+toMU()*

- Aiuta a trasformare il progetto in un JSON

3.2 SWEDesigner::Client::Components

3.2.1 Informazioni generali

- **Descrizione:**

- Questo package contiene tutti i components dell'applicazione.

- **Padre:** SWEDesigner::Client

- **Package contenuti:**

- Editor-container

Il package contiene tutti i components riguardanti l'editor e la gestione dell'utente

3.2.2 Classi

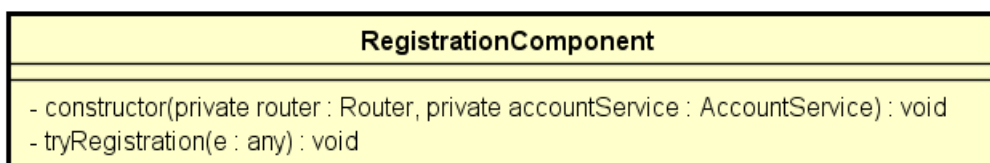


Figura 3: Diagramma della classe RegistrationComponent

3.2.2.1 SWEDesigner::Client::Components::RegistrationComponent

- **Descrizione:**

È il componente che descrive la pagina di registrazione dell'applicazione, mette a disposizione dell'utente un form dove inserire le informazioni necessarie alla creazione di un nuovo account utente. Gestisce le operazioni e la logica applicativa per la registrazione.

- **Utilizzo:**

Questo componente viene istanziato dinamicamente dal servizio Router del framework Angular quando viene richiesta la pagina di registrazione.

- **Metodi:**

- *-constructor(private router: Router, private accountService: AccountService)*
Crea un'istanza di RegistrationComponent

Parametri:

- * *-router: Router* Necessario per l'importazione del Router
- * *-accountService: AccountService* Necessario per l'importazione di AccountService

- *-tryRegistration(e: any)*
Tenta di registrare un utente

Parametri:

- * *+e: any*
Contiene i dati dell'utente da registrare

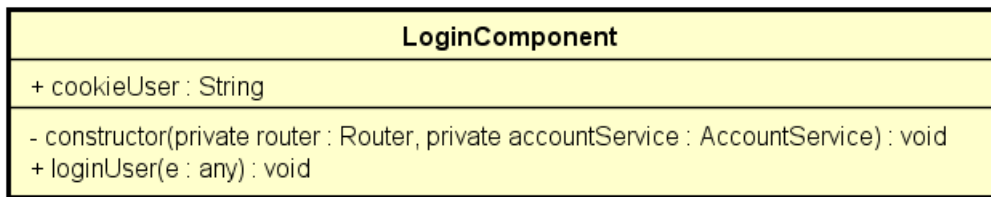


Figura 4: Diagramma della classe LoginComponent

3.2.2.2 SWEDesigner::Client::Components::LoginComponent

- **Descrizione:**

È il componente che descrive la pagina di login dell'applicazione, mette a disposizione dell'utente un form dove inserire username e password. Gestisce le operazioni e la logica applicativa per il login.

- **Utilizzo:**

Questo componente viene istanziato dinamicamente dal servizio Router del framework Angular quando viene richiesta la pagina di login.

- **Attributi:**

- *+cookieUser: String*
Riceve l'username dai cookie di sessione

- **Metodi:**

- *-constructor(private router: Router, private accountService: AccountService)*
Costruttore della classe

Parametri:

- * *-router: Router* Necessario per l'importazione del Router
- * *-accountService: AccountService* Necessario per l'importazione di AccountService
- *+loginUser(e: any)*
Effettua l'autenticazione dell'utente

Parametri:

- * *+e: any* Contiene i dati dell'utente da autenticare

3.2.2.3 SWEDesigner::Client::Components::Forgot-pswComponent

- **Descrizione:**

È il componente che descrive la pagina per il recupero della password dell'applicazione.

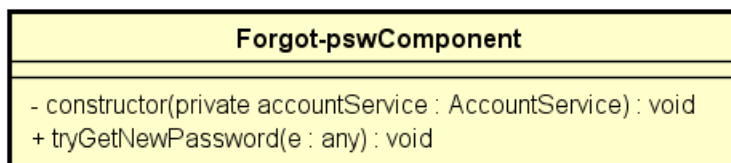


Figura 5: Diagramma della classe Forgot-pswComponent

cazione, mette a disposizione un form in cui inserire l'indirizzo email. Gestisce le operazioni e la logica applicativa relativa al recupero della password.

- **Utilizzo:**

Questo componente viene istanziato dinamicamente dal servizio Router del framework Angular quando viene richiesta la pagina di password dimenticata.

- **Metodi:**

- *-constructor(private accountService: AccountService)*

Crea un istanziazione di Forgot-pswComponent

Parametri:

- * *-accountService: AccountService* Necessario per l'importazione di AccountService

- *+tryGetNewPassword(e: any)*

Invia all'utente la password per email

Parametri:

- * *+e: any* Contiene i dati dell'utente che ha richiesto il recupero password

3.3 SWEDesigner::Client::Components::Editor-container

3.3.1 Informazioni generali

- **Descrizione:**

Questo package contiene tutti i components riguardanti l'editor e la gestione dell'utente.

- **Padre:** SWEDesigner::Client::Components

- **Package contenuti:**

- Menu

Il package contiene tutti i components riguardanti la gestione delle funzionalità offerte dal menu

- Editor

Il package contiene tutti i components riguardanti l'editor e la gestione dell'utente

3.3.2 Classi

3.3.2.1 SWEDesigner::Client::Components::Editor-container::Editor-containerComponent

- **Descrizione:**

È il componente che contiene il componente del menù e quello dell'editor.

- **Utilizzo:**

- **Attributi:**

- *-selectedGraph: any*
Punta al graph corrente.

- **Metodi:**

- *-constructor(private menuService: MenuService, private accountService: AccountService)*
Costruttore della classe

Parametri:

- * *menuService: MenuService*
Crea un istanziazione di MenuService
- * *accountService: AccountService*
Crea un istanziazione di AccountService

3.3.2.2 SWEDesigner::Client::Components::Editor-container::Activity-frameComponent

3.4 SWEDesigner::Client::Components::Editor-container::Menu

3.4.1 Informazioni generali

- **Descrizione:**

Questo package contiene tutti i components riguardanti la gestione delle funzionalità offerte dal menu.

- **Padre:** SWEDesigner::Client::Components::Editor-container

3.4.2 Classi

3.4.2.1 SWEDesigner::Client::Components::Editor-container::Menu::MenuComponent

– **Descrizione:**

È il componente che contiene il *menù*.

– **Utilizzo:**

– **Attributi:**

– **Metodi:**

* *-constructor(private accountService: AccountService)*

Costruttore della classe

Parametri:

· *accountService: AccountService*

Crea un'istanza di AccountService

FileComponent
<ul style="list-style-type: none"> - nomeProgetto : string - constructor(private menuService : MenuService, private mainEditorService : MainEditorService, private accountService : AccountService) : void - save(projName : string) : void - updateProj() : void - export() : void - generate() : void - import(event : any) : void

Figura 6: Diagramma della classe FileComponent

3.4.2.2 SWEDesigner::Client::Components::Editor-container::Menu::FileComponent

– **Descrizione:**

È il componente che descrive la voce *File* del menu dell'editor.

– **Utilizzo:**

– **Attributi:**

* *-nomeProgetto: string*

Contiene il nome del progetto corrente

– **Metodi:**

- * *-constructor(private menuService: MenuService, private mainEditorService: MainEditorService, private accountService: AccountService)*

Costruttore della classe

Parametri:

- *menuService: MenuService*
Crea un'istanziatura del MenuService
- *mainEditorService: MainEditorService*
Crea un'istanziatura del MainEditorService
- *accountService: AccountService*
Crea un'istanziatura di AccountService

- * *-save(projName: string)*

Memorizza un progetto nel database

Parametri:

- *projName: string*
Nome del progetto da memorizzare

- * *-updateProj()*

Aggiorna un progetto esistente

- * *-export()*

Esporta il progetto corrente

- * *-generate()*

Genera il codice Java

- * *-import(event: any)*

Importa un progetto esistente

Parametri:

- *event: any*
Progetto da importare

3.4.2.3 SWEDesigner::Client::Components::Editor-container::Menu::ModificaComponente

– **Descrizione:**

È il componente che descrive la voce *Modifica* del menu dell'editor.

– **Utilizzo:**

– **Metodi:**

- * *+doZoomIn()*
Esegue lo zoom in avanti

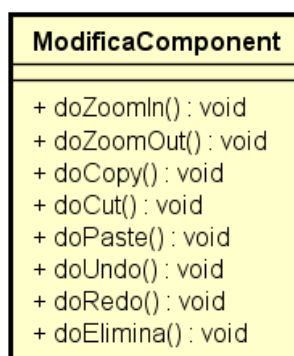


Figura 7: Diagramma della classe ModificaComponent

- * *+doZoomOut()*
Esegue lo zoom in indietro
- * *+doCopy()*
Esegue il comando copia del menu
- * *+doCut()*
Esegue il comando taglia del menu
- * *+doPaste()*
Esegue il comando incolla del menu
- * *+doUndo()*
Annulla l'ultima azione compiuta
- * *+doRedo()*
Ripristina l'ultima azione annullata
- * *+doElimina()*
Elimina l'elemento selezionato nell'editor

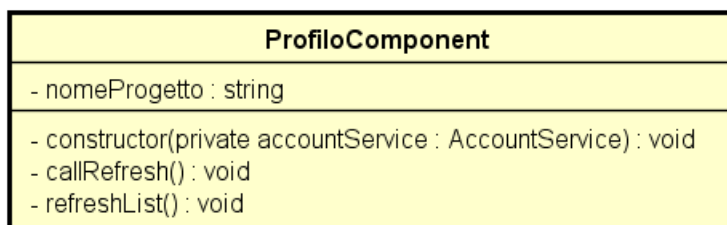


Figura 8: Diagramma della classe ProfiloComponent

3.4.2.4 SWEDesigner::Client::Components::Editor-container::Menu::ProfiloComponent

- **Descrizione:**

È il componente che descrive la voce *Profilo* del menu dell'editor.

- **Utilizzo:**

- **Attributi:**

- * *-nomeProgetto: string*

Contiene il nome del progetto corrente

- **Metodi:**

- * *-constructor(private accountService: AccountService)*

Costruttore della classe

- Parametri:**

- *accountService: AccountService*

Crea un'istanza di AccountService

- * *-callRefresh()*

Effettua la chiamata al metodo di aggiornamento della lista progetti

- * *-refreshList()*

Aggiorna la lista progetti

3.4.2.5 SWEDesigner::Client::Components::Editor-container::Menu::ProgettoComponent

3.5 SWEDesigner::Client::Components::Editor-container::Editor

3.5.1 Informazioni generali

- **Descrizione:**

Questo package contiene tutti i componenti riguardanti l'editor e la gestione dell'utente.

- **Padre:** SWEDesigner::Client::Components::Editor-container

- **Package contenuti:**

- * Edit-class-menu

3.5.2 Classi

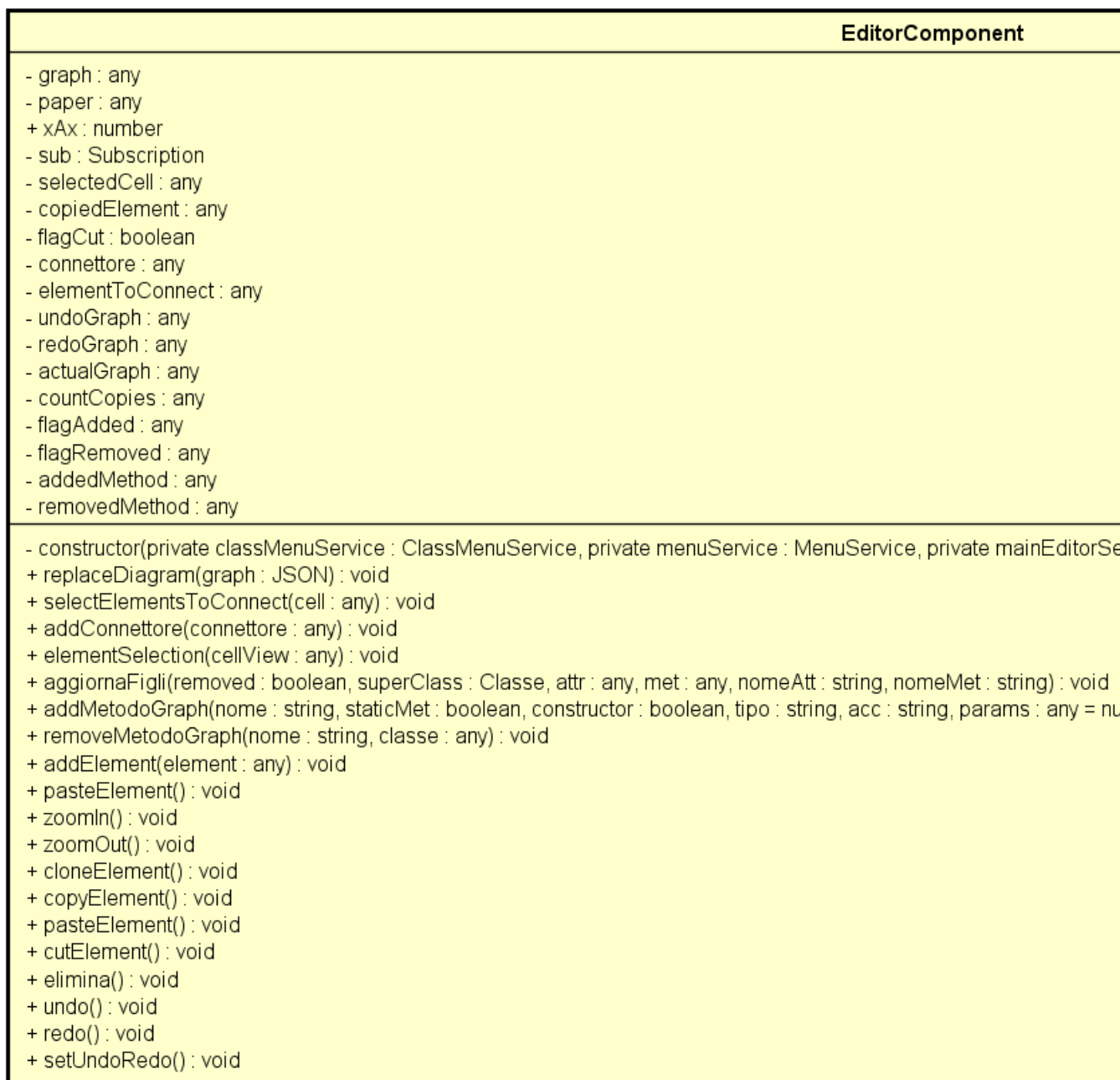


Figura 9: Diagramma della classe EditorComponent

3.5.2.1 SWEDesigner::Client::Components::Editor-container::Editor::EditorComponent

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-graph: any*
Contiene tutti gli elementi del grafico
- * *-paper: any*
Assicura che vengano renderizzati gli elementi del grafico
- * *+xAx: number*
Serve per scalare il grafico
- * *-sub: Subscription*
Permette la funzione di zoom
- * *-selectedCell: any*
Punta all'elemento selezionato con il click
- * *-copiedElement: any*
Punta all'elemento copiato o tagliato
- * *-flagCut: boolean*
Indica se l'elemento è stato tagliato, altrimenti è stato copiato
- * *-connettore: any*
Il tipo del connettore selezionato
- * *-elementToConnect: any*
Punta all'elemento selezionato con il click, che sarà collegato con il connettore
- * *-undoGraph: any*
Punta al grafico dopo aver annullato l'ultima operazione
- * *-redoGraph: any*
Punta al grafico dopo aver ripristinato l'ultima operazione annullata
- * *-actualGraph: any*
Punta al grafico attuale
- * *-countCopies: any*
Conta il numero di copie dello stesso elemento
- * *-flagAdded: any*
Indica se bisogna ascoltare l'evento aggiungere del grafo

- * *-flagRemoved: any*
Indica se bisogna ascoltare l'evento rimuovere del grafo
- * *-addedMethod: any*
Indica al metodo annulla se un metodo è stato aggiunto
- * *-removedMethod: any*
Indica al metodo annulla se un metodo è stato rimosso

– **Metodi:**

- * *-constructor(private classMenuService: ClassMenuService, private menuService: MenuService, private mainEditorService: MainEditorService, private activityService: ActivityService)*
Costruttore della classe

Parametri:

- *classMenuService: ClassMenuService*
Crea un istanziazione di ClassMenuService
- *menuService: MenuService*
Crea un istanziazione di MenuService
- *mainEditorService: MainEditorService*
Crea un istanziazione di MainEditorService
- *activityService: ActivityService*
Crea un istanziazione di ActivityService

- * *+replaceDiagram(graph: JSON)*
Rimpiazza l'editor con una nuova finestra contenuta nel file JSON

Parametri:

- *graph: JSON*
File contenente la finestra

- * *+selectElementsToConnect(cell: any)*
Seleziona gli elementi da collegare con il connettore

Parametri:

- *cell: any*
Elemento da collegare

- * *+addConnettore(connettore: any)*
Aggiunge un connettore alla classe

Parametri:

- *connettore: any*
Connettore da aggiungere

* *+elementSelection(cellView: any)*

Seleziona una shape nell'editor

Parametri:

- *cellView: any*

Shape da selezionare

* *+aggiornaFigli(removed: boolean, superClass: Classe, attr: any, met: any, nomeAtt: string, nomeMet: string)*

Aggiorna i metodi e gli attributi figli della classe padre

Parametri:

- *removed: boolean*

Indica se deve rimuovere un attributo o un metodo

- *superClass: Classe*

Classe padre

- *attr: any*

Attributo da aggiungere

- *met: any*

Metodo da aggiungere

- *nomeAtt: string*

Attributo da rimuovere

- *nomeMet: string*

Metodo da rimuovere

* *+addMetodoGraph(nome: string, staticMet: boolean, constructor: boolean, tipo: string, acc: string, params: any = null, classe: any)*

Aggiunge alla cella class un metodo

Parametri:

- *nome: string*

Nome del metodo

- *staticMet: boolean*

True se è marcato static

- *constructor: boolean*

True se è un costruttore

- *tipo: string*

Tipo di ritorno del metodo

- *acc: string*

Accessibilità del metodo

- *params: any*
Lista dei parametri del metodo

- *classe: any*
Indica la cella

- * *+removeMetodoGraph(nome: string, classe: any)*
Rimuove un metodo dalla cella classe

Parametri:

- *nome: string*
Nome del metodo

- *classe: any*
Indica la cella

- * *+addElement(element: any)*
Aggiunge un elemento all'editor

Parametri:

- *element: any*
Elemento da aggiungere

- * *+pasteElement()*
Incolla un elemento precedentemente copiato

- * *+zoomIn()*
Effettua lo zoom in avanti

- * *+zoomOut()*
Effettua lo zoom all'indietro

- * *+cloneElement()*
Clona l'elemento selezionato

- * *+copyElement()*
Copia l'elemento selezionato

- * *+pasteElement()*
Incolla l'elemento precedentemente copiato/tagliato

- * *+cutElement()*
Taglia l'elemento selezionato

- * *+elimina()*
Elimina l'elemento selezionato

- * *+undo()*
Annulla l'ultima azione

- * *+redo()*
Ripristina l'ultima azione annullata
- * *+setUndoRedo()*
Aggiorna il diagramma attuale e il undoGraph

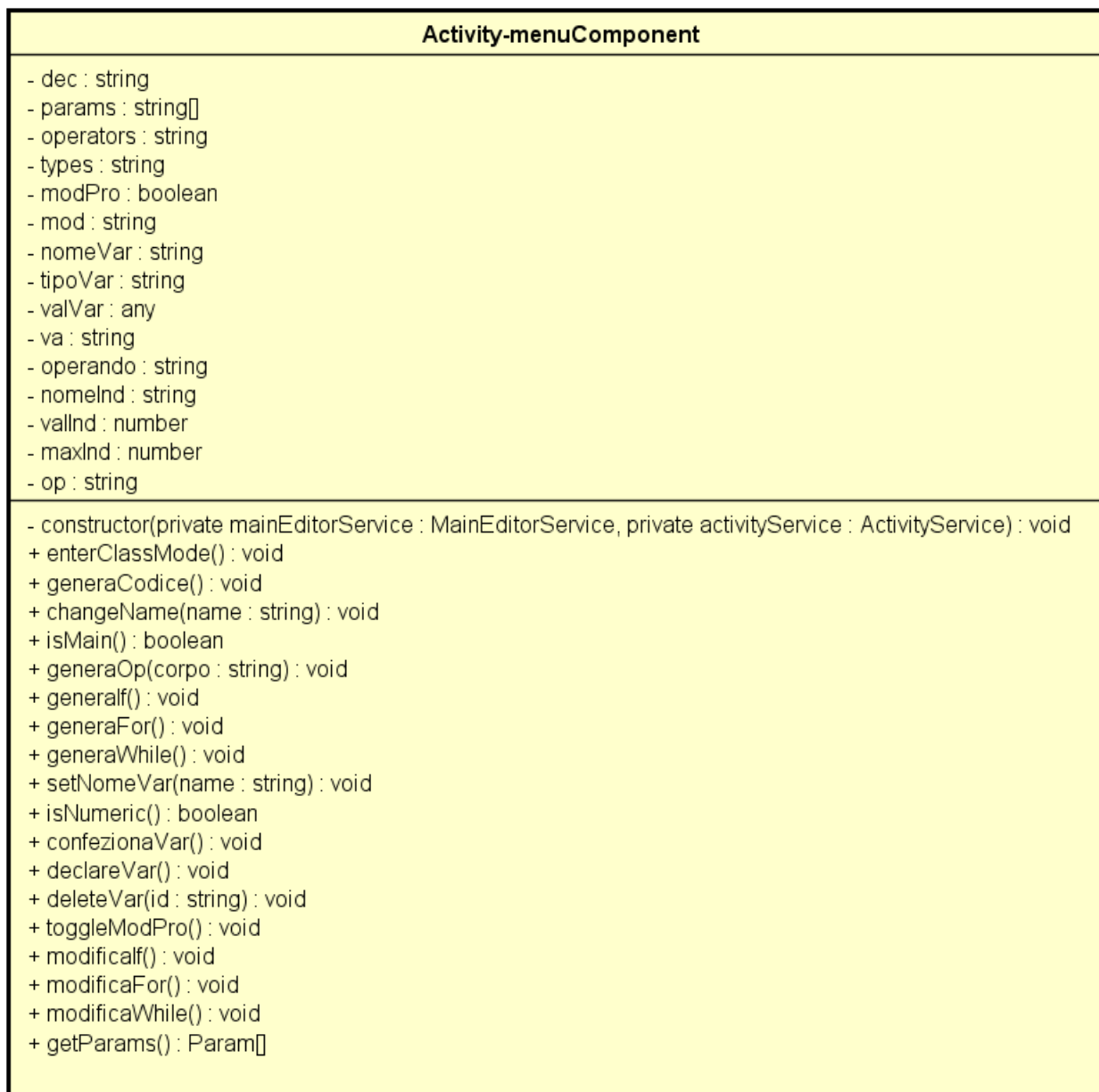


Figura 10: Diagramma della classe Activity-menu

3.5.2.2 SWEDesigner::Client::Components::Editor-container::Editor::Activity-menuCo

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-decisions: string[]*
Lista delle decisioni
- * *-dec: string*
Decisione presa
- * *-params: string[]*
Lista dei parametri
- * *-operators: string*
Lista degli operatori
- * *-types: string*
Lista dei tipi
- * *-modPro: boolean*
Modalità del body
- * *-mod: string*
Assegna un bottone in html DOM
- * *-nomeVar: string*
Nome della variabile
- * *-tipoVar: string*
Tipo della variabile
- * *-valVar: any*
Valore della variabile
- * *-va: string*
Nome della variabile dell'if statement
- * *-operando: string*
Operando dell'if
- * *-nomeInd: string*
Nome della variabile per il loop
- * *-valInd: number*
Variabile indice per il loop

* *-maxInd: number*
Indice massimo per il loop

* *-op: string*
Operazione

– **Metodi:**

* *-constructor(private mainEditorService: MainEditorService, private activityService: ActivityService)*
Costruttore della classe

Parametri:

- *mainEditorService: MainEditorService*
Crea un istanza di MainEditorService
- *activityService: ActivityService*
Crea un istanza di ActivityService

* *+enterClassMode()*
Entra nel diagramma delle classi

* *+generaCodice()*
Genera il codice

* *+changeName(name: string)*
Cambia il nome con un nuovo valore

Parametri:

- *name: string*
Nuovo nome

* *+isMain()*
Ritorna true se il metodo è main

* *+generaOp(corpo: string)*
Genera il corpo dell'operazione

Parametri:

- *corpo: string*
Corpo dell'operazione

* *+generaIf()*
Genera l'if statemente

* *+generaFor()*
Genera il ciclo for

* *+generaWhile()*
Genera il ciclo while

- * *+setNomeVar(name: string)*
Setta il nuovo nome della variabile
Parametri:
 - *name: string*
Nuovo nome della variabile
- * *+isNumeric()*
Ritorna true se la stringa è numerica
- * *+confezionaVar()*
La funzione costruisce la variabile
- * *+declareVar()*
Dichiara la variabile
- * *+deleteVar(id: string)*
Elimina una variabile
Parametri:
 - *id: string*
Id della variabile da eliminare
- * *+toggleModPro()*
Modifica il modo per editare uno statement
- * *+modificaIf()*
Edita l'if statement
- * *+modificaFor()*
Edita il ciclo for
- * *+modificaWhile()*
Edita il ciclo while
- * *+getParams()*
Ritorna la lista dei parametri

3.5.2.3 SWEDesigner::Client::Components::Editor-container::Editor::ToolbarComponent

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-classCouter: number*
Conta il numero di classi presenti nell'editor

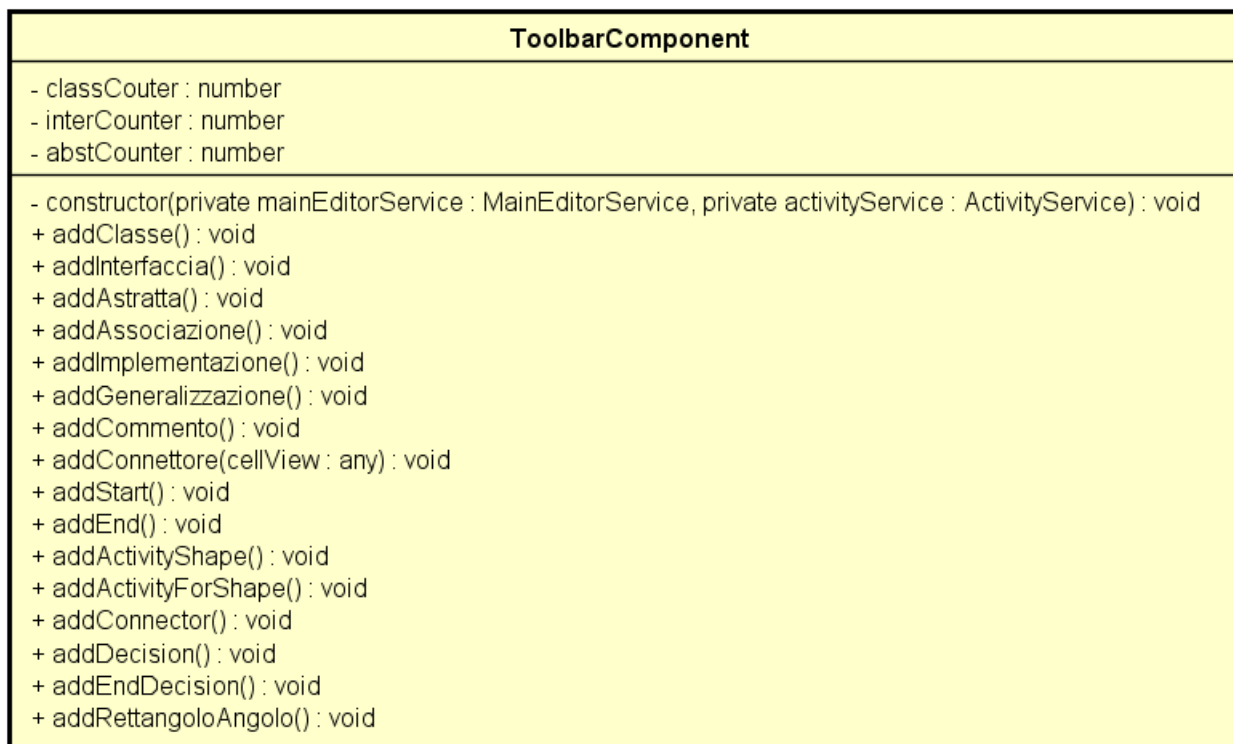


Figura 11: Diagramma della classe ToolbarComponent

* *-interCounter: number*

Conta il numero di interfacce presenti nell'editor

* *-abstCounter: number*

Conta il numero di classi astratte presenti nell'editor

– **Metodi:**

* *-constructor(private mainEditorService: MainEditorService, private activityService: ActivityService)*

Costruttore della classe

Parametri:

· *mainEditorService: MainEditorService*

Crea un'istanza di MainEditorService

· *activityService: ActivityService*

Crea un'istanza di ActivityService

* *+addClasse()*

Aggiunge una classe all'editor

- * *+addInterfaccia()*
Aggiunge un interfaccia all'editor
- * *+addAstratta()*
Aggiunge una classe astratta all'editor
- * *+addAssociazione()*
Seleziona il tipo di connettore *Associazione*
- * *+addImplementazione()*
Seleziona il tipo di connettore *Implementazione*
- * *+addGeneralizzazione()*
Seleziona il tipo di connettore *Generalizzazione*
- * *+addCommento()*
Aggiunge un commento all'editor
- * *+addConnettore(cellView: any)*
Aggiunge il connettore selezionato
 - *cellView: any*
Elemento target o source
- * *+addStart()*
Aggiunge l'elemento start all'editor
- * *+addEnd()*
Aggiunge l'elemento end all'editor
- * *+addActivityShape()*
Aggiunge un azione
- * *+addConnector()*
Seleziona il connettore freccia per l'activity diagram
- * *+addDecision()*
Aggiunge all'editor un inizio if/ciclo
- * *+addEndDecision()*
Aggiunge all'editor una fine if/ciclo
- * *+addRettangoloAngolo()*
Aggiunge un attività

3.5.2.4 SWEDesigner::Client::Components::Editor-container::Editor::ActivityService

– **Descrizione:**

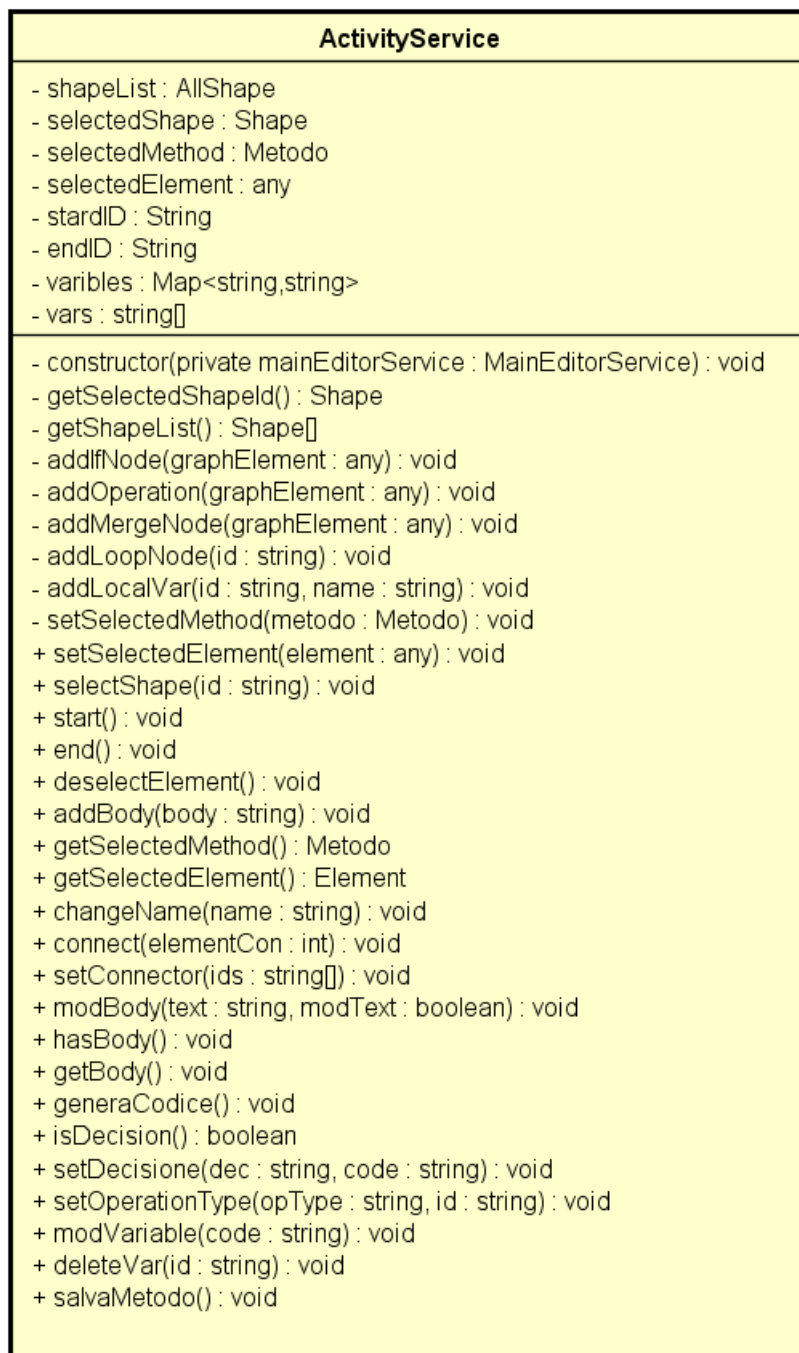


Figura 12: Diagramma della classe ActivityService

– Utilizzo:

– Attributi:

- * *-shapeList: AllShape*
Lista della shape
- * *-selectedShape: Shape*
Shape selezionata
- * *-selectedMethod: Metodo*
Metodo selezionato
- * *-selectedElement: any*
Elemento selezionato
- * *-startID: string*
Id dell'elemento start
- * *-endID: string*
Id dell'elemento end
- * *-variables: Map<string, string>*
Lista dei parametri
- * *-vars: string[]*
Lista delle variabili

– Metodi:

- * *-constructor(private mainEditorService: MainEditorService)*
Costruttore della classe

Parametri:

- *mainEditorService: MainEditorService*
Crea un'istanza di MainEditorService

- * *-getSelectedShapeId()*
Ritorna l'id della shape selezionata

- * *-getShapeList()*
Ritorna la lista delle shapes

- * *-addIfNode(graphElement: any)*
Aggiunge un if statement

Parametri:

- *graphElement: any*
Si riferisce al diagramma

* *-addOperation(graphElement: any)*

Aggiunge un operazione

Parametri:

- *graphElement: any*

Si riferisce al diagramma

* *-addMergeNode(graphElement: any)*

Aggiunge un merge

Parametri:

- *graphElement: any*

Si riferisce al diagramma

* *-addLoopNode(id: string)*

Aggiunge un ciclo

Parametri:

- *id: string*

Id della shape

* *-addLocalVar(id: string, name: string)*

Aggiunge una variabile locale

Parametri:

- *id: string*

Id della shape

- *name: string*

Nome della variabile

* *-setSelectedMethod(metodo: Metodo)*

Setta il metodo selezionato

Parametri:

- *metodo: Metodo*

Metodo da selezionare

* *+setSelectedElement(element: any)*

Setta l'elemento selezionato

Parametri:

- *element: any*

Elemento selezionato

* *+selectShape(id: string)*

Seleziona la shape

Parametri:

- *id: string*

Id della shape da selezionare

- * *+start()*

Permette di aggiungere una shape start

- * *+end()*

Permette di aggiungere una shape end

- * *+deselectElement()*

Deseleziona l'elemento

- * *+addBody(body: string)*

Aggiunge il body della shape

Parametri:

- *body: string*

Body della shape

- * *+getSelectedMethod()*

Ritorna il metodo selezionato

- * *+getSelectedElement()*

Ritorna l'elemento selezionato

- * *+getNameMethod()*

Ritorna il nome del metodo

- * *+getVarVis()*

Ritorna il valore di vars

- * *+getShapeType()*

Ritorna il tipo della shape

- * *+changeName(name:string)*

Modifica il nome della shape

Parametri:

- *name: string*

Nuovo nome della shape

- * *+connect(elementCon: any)*

Linka la shape

Parametri:

- *elementCon: any*

Shape da linkare

- * *+setConnector(ids: string[])*

Connette la shape

Parametri:

- *ids: string[]*

Id della shape da connettere

- * *+modBody(text: string, modText: boolean)*

Modifica il body della shape

Parametri:

- *text: string*

Nuovo valore

- *modText: boolean*

Nuovo valore

- * *+hasBody()*

Controlla se ha un body

- * *+getBody()*

Restituisce il body

- * *+generaCodice()*

Genera il codice

- * *+isDecision()*

True se è una decisione

- * *+isVarDeclaration()*

True se è una dichiarazione

- * *+setDecisione(dec: string, code: string)*

Setta il nuovo valore della decisione

Parametri:

- *dec: string*

Nuovo valore

- *code: string*

Codice

- * *+setOperationType(opType: string, id: string)*

Setta il tipo di operazione

Parametri:

- *opType: string*

Tipo di operazione

- *id: string*

Id della shape

- * *+modVariable(code: string)*

Setta il nuovo valore della variabile

Parametri:

- *code: string*
Nuovo valore
- * *+deleteVar(id: string)*
Elimina una variabile
- Parametri:**
 - *id: string*
Id della variabile da eliminare
- * *+salvaMetodo()*
Salva il metodo

3.5.2.5 SWEDesigner::Client::Components::Editor-container::Editor::Class-menuService

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-selectedClassSource: Subject<any>*
Risorsa observable oggetto-classe
- * *-selectedClass: any*
Stream observable oggetto-classe
- * *-classe: any*
La classe correntemente selezionata nell'editor
- * *-name: string*
Il nome della classe correntemente selezionata nell'editor
- * *-sub: Subscription*
Subscription dell'oggetto observable che è la classe selezionata nell'editor
- * *-types: string[]*
Array di tipi di dato primitivi
- * *-methodTypes: string[]*
Array di tipi di dato primitivi per i tipi di ritorno dei metodi
- * *-accessoAttr: string[]*
Array contenente le visibilità delle classi
- * *-selectedTipoAtt: string*
Usato per memorizzare il tipo selezionato per il costruttore di un nuovo attributo

Class-menu
<ul style="list-style-type: none"> - selectedClassSource : Subject<any> - selectedClass : any - classe : any - name : String - sub : Subscription - types : string[] - methodTypes : string[] - accessoAttr : string[] - selectedTipoAtt : String - selectedTipoAttEdit : String - selectedAccAtt : String - selectedAccAttEdit : String - selectedTipoMet : String - nomeMet : String - selectedAccMet : String - parametriMetodo : Param[] - costruttore : boolean - isThereAMain : boolean
<ul style="list-style-type: none"> - constructor(private mainEditorService : MainEditorService, private activityService : ActivityService) : void + classSelection(classe : any) : void + changeClassName(name : string) : void + addAttributo(nome : string, staticAtt : boolean, finalAtt : boolean, tipo : string, acc : string) : void + removeAttributo(nome : string) : void + changeAttributo(name : string, oldName : string, tipo : string, acc : string, stat : boolean, final : boolean) : void + getAttributi() : Attributo[] + addMetodo(nome : string, staticMet : boolean, constructor : boolean, tipo : string, acc : string, params : any = null) : void + removeMetodo(nome : string) : void + modifyMetodo(nome : string) : void + getMetodi() : Metodo[] + removeClass(name : string, classe : Classe) : void + isMethodAddable() : boolean

Figura 13: Diagramma della classe Class-menu

* *-selectedTipoAttEdit: string*

Usato per memorizzare il tipo selezionato per editare l'attributo

* *-selectedAccAtt: String*

Usato per memorizzare la visibilità selezionata per costruire un nuovo attributo

* *-selectedAccAttEdit: String*

Usato per memorizzare la visibilità selezionata per editare l'attributo

- * *-selectedTipoMet: String*
Usato per memorizzare il tipo di ritorno selezionato per costruire un nuovo metodo
- * *-nomeMet: String*
Usato per memorizzare il nome del nuovo metodo
- * *-selectedAccMet: String*
Usato per selezionare la visibilità per costruire un nuovo metodo
- * *-parametriMetodo: Param[]*
Usato per memorizzare un array di parametri per costruire un nuovo metodo
- * *-costruttore: boolean*
Usato per memorizzare se il metodo è un costruttore
- * *-isThereAMain: boolean*
Usato per memorizzare se è stato aggiunto il metodo main

– **Metodi:**

- * *-constructor(private mainEditorService: MainEditorService, private activityService: ActivityService)*
Crea un'istanza di ClassMenuComponent e setta le proprietà classe e nome per subscription da classMenuService

Parametri:

- *mainEditorService: MainEditorService*
Usato per creare una nuova istanziazione di ClassMenuService
- *activityService: ActivityService*
Usato per creare una nuova istanziazione di ClassMenuService

- * *+classSelection(classe: any)*
Comandi messaggio del servizio

Parametri:

- *classe: any*
Variabile usata per settare la classe selezionata

- * *+changeClassName(name: string)*
Cambia il nome della classe selezionata

Parametri:

- *name: string*
Nuovo nome della classe

- * *+addAttributo(nome: string, staticAtt: boolean, finalAtt: boolean, tipo: string, acc: string)*

Aggiunge un nuovo attributo alla classe

Parametri:

- *nome: string*
Nome dell'attributo
- *staticAtt: boolean*
True se l'attributo è static
- *finalAtt: boolean*
True se l'attributo è final
- *tipo: string*
Tipo dell'attributo
- *acc: string*
Visibilità dell'attributo

* *+removeAttributo(nome: string)*

Rimuove un attributo dalla classe selezionata

Parametri:

- *nome: string*
Nome dell'attributo da rimuovere

* *+changeAttributo(name: string, oldName: string, tipo: string, acc: string, stat: boolean, final: boolean)*

Modifica le proprietà di un attributo della classe selezionata

Parametri:

- *name: string*
Nuovo nome dell'attributo
- *oldName: string*
Vecchio nome dell'attributo
- *tipo: string*
Tipo dell'attributo
- *acc: string*
Tipo di visibilità
- *stat: boolean*
True se è marcato static
- *final: boolean*
True se è marcato final

* *+getAttributi()*

Ritorna la lista degli attributi di una classe

- * *+addMetodo(nome: string, staticMet: boolean, constructor: boolean, tipo: string, acc: string, params: any = null)*

Aggiunge un nuovo metodo alla classe selezionata

Parametri:

- *nome: string*
Nome del metodo
- *staticMet: boolean*
True se è marcato static
- *constructor: boolean*
True se è un costruttore
- *tipo: string*
Tipo del metodo
- *acc: string*
Visibilità del metodo
- *params: any = null*
Lista di parametri del metodo

- * *+removeMetodo(nome: string)*

Rimuove un metodo dalla classe selezionata

Parametri:

- *nome: string*
Nome del metodo da rimuovere

- * *+modifyMetodo(nome: string)*

Setta la modalità activity nell'editor per editare il metodo della classe selezionata

Parametri:

- *nome: string*
Nome del metodo da editare

- * *+getMetodi()*

Ritorna la lista dei metodi di una classe

- * *+removeClass(name: string, classe: Classe)*

Rimuove la classe selezionata

Parametri:

- *name: string*
Nome della classe
- *classe: Classe*
Tipo della classe

* *+isMethodAddable()*

Ritorna true se il metodo è aggiungibile dalla logica

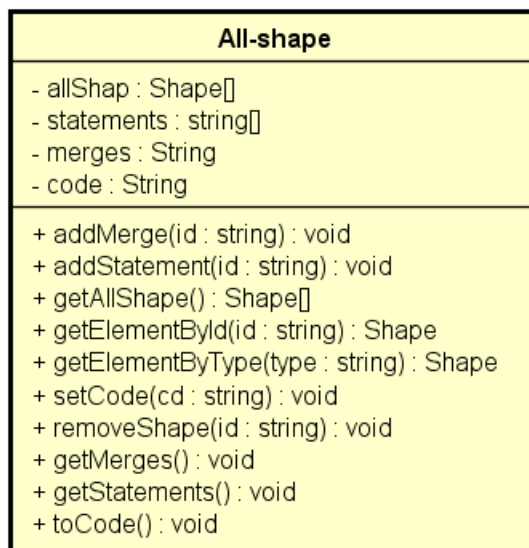


Figura 14: Diagramma della classe All-shape

3.5.2.6 SWEDesigner::Client::Components::Editor-container::Editor::All-shape

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-allShap: Shape[]*

Rappresenta l'array di shapes

* *-statements: string[]*

Rappresenta lo stato delle shapes

* *-merges: string*

Ritorna il risultato delle shapes mergiate

* *-code: string*

Contiene il codice generato dal metodo

– **Metodi:**

- * *+addMerge(id: string)*
Aggiunge il codice corrente al progetto
Parametri:
 - *id: string*
Progetto
- * *+addStatement(id: string)*
Aggiunge lo statement alla decisione
Parametri:
 - *id: string*
Statement da aggiungere
- * *+getAllShape()*
Ritorna tutte le shapes
- * *+getElementById(id: string)*
Ritorna il riferimento alla shape selezionata
Parametri:
 - *id: string*
Shape selezionata
- * *+getElementByType(type: string)*
Ritorna il riferimento alla shape selezionata
Parametri:
 - *type: string*
Shape eselzionata
- * *+setCode(cd: string)*
Setta la variabile code con il nuovo valore
Parametri:
 - *cd: string*
Nuovo valore
- * *+removeShape(id: string)*
Rimuove la shape selezionata
Parametri:
 - *id: string*
Shape da rimuovere
- * *+getMerges()*
Ritorna l'attributo merges
- * *+getStatements()*
Ritorna l'attributo statements

- * *+toCode()*
Converte le shape in codice

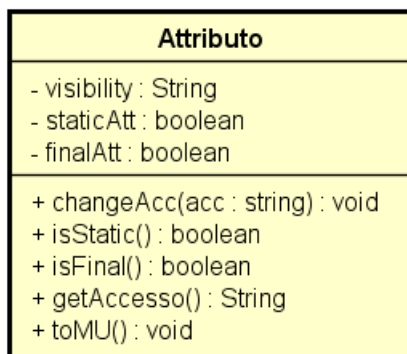


Figura 15: Diagramma della classe Attributo

3.5.2.7 SWEDesigner::Client::Components::Editor-container::Editor::Attributo

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-visibility: string*
Visibilità dell'attributo
- * *-staticAtt: boolean*
True se è marcato static
- * *-finalAtt: boolean*
True se è marcato final

– **Metodi:**

- * *+changeAcc(acc: string)*
Modifica la visibilità dell'attributo

Parametri:

- *acc: string*
Nuova visibilità dell'attributo
- * *+isStatic()*
Ritorna true se l'attributo è statico

- * *+isFinal()*
Ritorna true se l'attributo è final
- * *+getAccesso()*
Ritorna la visibilità dell'attributo
- * *+toMU()*
Converte l'attributo in una stringa o un file JSON

3.5.2.8 SWEDesigner::Client::Components::Editor-container::Editor::Class-errors

Classe-astratta
- abstractMethods : MetodiAstratti[]
+ constructor(nome : string) : void + addAbstractMethods(nome : string, tipo : string, acc : string, listaParam : string[]) : void + isAbstarct() : boolean + toJSON() : void

Figura 16: Diagramma della classe Classe-astratta

3.5.2.9 SWEDesigner::Client::Components::Editor-container::Editor::Classe-astratta

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-abstractMethods: MetodiAstratti[]*
Array contenente i metodi della classe astratta

– **Metodi:**

- * *+constructor(nome: string)*
Costruttore della classe astratta

Parametri:

- *nome: string*
Nome della classe astratta da creare

- * *+addAbstractMethods(nome: string, tipo: string, acc:string, listaParam: string[])*

Aggiunge un metodo astratto alla classe

Parametri:

- *nome: string*
Nome del metodo
- *tipo: string*
Tipo di ritorno del metodo
- *acc:string*
Visibilità del metodo
- *listaParam: string[]*
Lista dei parametri del metodo
- * *+isAbstarct()*
Ritorna true se l'oggetto è astratto
- * *+toJSON()*
Parsa la classe selezionata e la trasforma in formato JSON

3.5.2.10 SWEDesigner::Client::Components::Editor-container::Editor::Classe

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

- * *-nome: string*
Nome della classe, usato come identificatore
- * *-attributi: Attributo[]*
Lista degli attributi della classe
- * *-metodi: Metodo[]*
Lista dei metodi della classe
- * *-classePadre: string*
Nome della classe estesa da questa classe

– **Metodi:**

- * *-constructor(nome: string)*
Costruisce la classe

Parametri:

- *nome: string*
Nome della classe da costruire

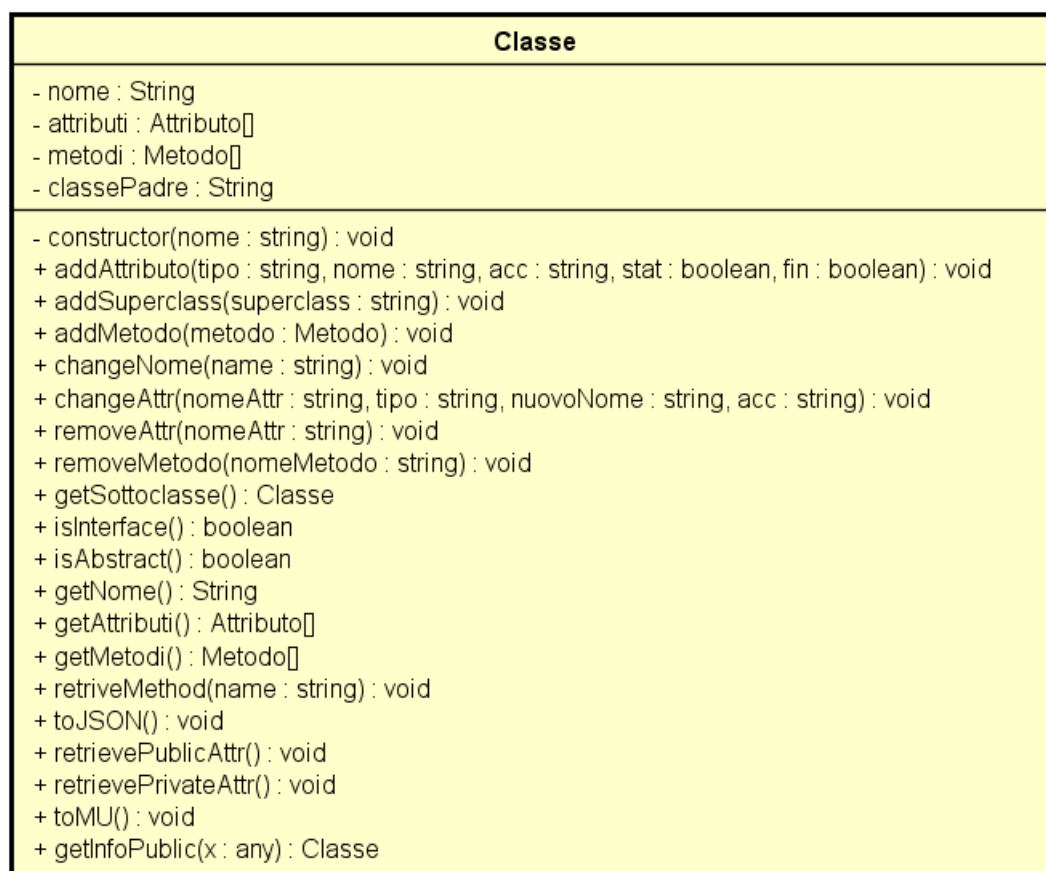


Figura 17: Diagramma della classe Classe

* *+addAttributo(tipo: string, nome: string, acc: string, stat: boolean, fin: boolean)*

Aggiunge un nuovo attributo all'array di attributi della classe selezionata

Parametri:

- *tipo: string*
Tipo dell'attributo
- *nome: string*
Nome dell'attributo
- *acc: string*
Visibilità dell'attributo
- *stat: boolean*
True se è marcato static

- *fin: boolean*
True se è marcato finale

- * *+addSuperclass(superclass: string)*
Aggiunge il nome della classe che è estesa da questa classe

Parametri:

- *superclass: string*
Nome della superclasse

- * *+addMetodo(metodo: Metodo)*
Aggiunge un metodo alla classe selezionata

Parametri:

- *metodo: Metodo*
Metodo da aggiungere

- * *+changeNome(name: string)*
Modifica il nome della classe selezionata

Parametri:

- *name: string*
Nuovo nome della classe

- * *+changeAttr(nomeAttr: string, tipo: string, nuovoNome: string, acc: string)*

Modifica un attributo della classe selezionata

Parametri:

- *nomeAttr: string*
Vecchio nome dell'attributo
- *tipo: string*
Tipo dell'attributo
- *nuovoNome: string*
Nuovo nome dell'attributo
- *acc: string*
Accessibilità dell'attributo

- * *+removeAttr(nomeAttr: string)*
Rimuove un attributo dalla lista degli attributi della classe

Parametri:

- *nomeAttr: string*
Nome dell'attributo da rimuovere

* *+removeMetodo(nomeMetodo: string)*

Rimuove un metodo dalla lista dei metodi della classe

Parametri:

· *nomeMetodo: string*

Nome del meotodo da rimuovere

* *+getSottoclasse()*

Ritorna il nome della superclasse

* *+isInterface()*

Ritorna true se l'oggetto è un interfaccia

* *+isAbstract()*

Ritorna true se l'oggetto è astratto

* *+getNome()*

Ritorna il nome della classe

* *+getAttributi()*

Ritorna la lista degli attributi della classe

* *+getMetodi()*

Ritorna la lista dei metodi della classe

* *+retriveMethod(name: string)*

Ritorna un determinato metodo dall'array dei metodi

Parametri:

· *name: string*

Nome del metodo da ritornare

* *+toJSON()*

Effettua override della funzione

* *+retrievePublicAttr()*

* *+retrievePrivateAttr()*

* *+toMU()*

* *+getInfoPublic(x: any)*

Parametri:

· *x: any*

* *+getInfoPrivate(x)*

Parametri:

· *x: any*

3.5.2.11 SWEDesigner::Client::Components::Editor-container::Editor::Elemento-metodo

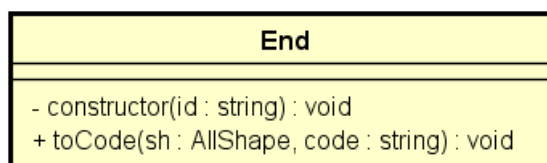


Figura 18: Diagramma della classe End

3.5.2.12 SWEDesigner::Client::Components::Editor-container::Editor::End

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(id : string)*
Costruttore della classe

Parametri:

· *id : string*
Id della shape

* *+getType()*
Ritorna il tipo della shape

* *+toCode(sh: AllShape, code: string)*
Converte la shape in codice

Parametri:

· *sh: AllShape*
Shape da convertire

· *code: string*
Stringa di codice

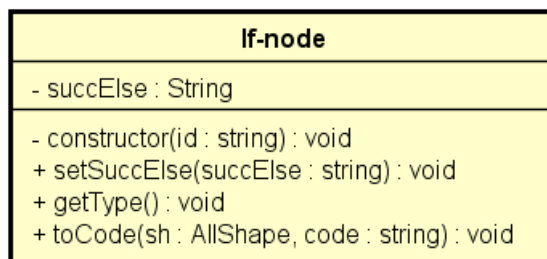


Figura 19: Diagramma della classe If-node

3.5.2.13 SWEDesigner::Client::Components::Editor-container::Editor::If-node

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-succElse: string*

Si riferisce all'else statement

– **Metodi:**

* *-constructor(id: string)*

Costruttore della classe

Parametri:

· *id: string*

Id della shape

* *+getSuccElse()*

Ritorna il valore di succElse

* *+setSuccElse(succElse: string)*

Assegna un valore a succElse

· *succElse: string*

Valore da assegnare

* *+getType()*

Ritorna il tipo della classe

* *+toCode(sh: AllShape, code: string)*

Converte la shape in codice

Parametri:

- *sh: AllShape*
Shape da convertire
- *code: string*
Stringa di codice

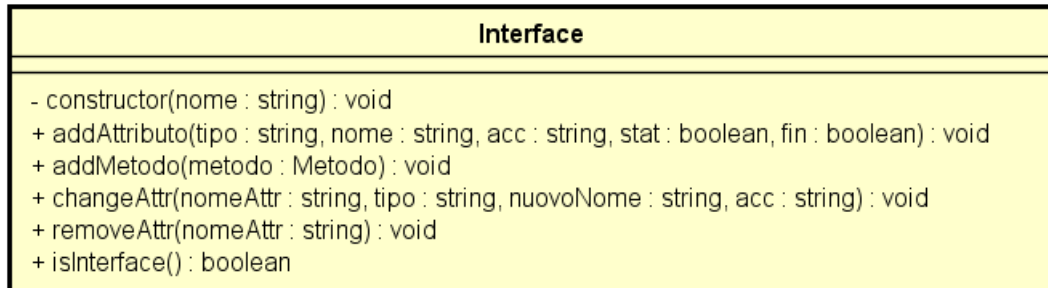


Figura 20: Diagramma della classe Interface

3.5.2.14 SWEDesigner::Client::Components::Editor-container::Editor::Interface

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

- * *-constructor(nome: string)*
Costruisce una nuova interfaccia

Parametri:

- *nome: string*
Nome dell'interfaccia

- * *+addAttributo(tipo: string, nome: string, acc: string, stat: boolean, fin: boolean)*

Aggiunge un attributo all'array di attributi dell'interfaccia

Parametri:

- *tipo: string*
Tipo dell'attributo
- *nome: string*
Nome dell'attributo
- *acc: string*
Visibilità dell'attributo

- *stat: boolean*
True se è marcato static
- *fin: boolean*
True se è marcato final
- * *+addMetodo(metodo: Metodo)*
Aggiunge un metodo all'array di metodi dell'interfaccia
Parametri:
 - *metodo: Metodo*
Metodo da aggiungere
- * *+changeAttr(nomeAttr: string, tipo: string, nuovoNome: string, acc: string)*
Modifica un attributo dell'interfaccia
Parametri:
 - *nomeAttr: string*
Vecchio nome dell'attributo
 - *tipo: string*
Tipo dell'attributo
 - *nuovoNome: string*
Nuovo nome dell'attributo
 - *acc: string*
Visibilità dell'attributo
- * *+removeAttr(nomeAttr: string)*
Rimuove un attributo dalla lista degli attributi dell'interfaccia
Parametri:
 - *nomeAttr: string*
Nome dell'attributo da rimuovere
- * *+isInterface()*
Ritorna true se è un interfaccia

3.5.2.15 SWEDesigner::Client::Components::Editor-container::Editor::Merge-node

- **Descrizione:**
- **Utilizzo:**
- **Metodi:**

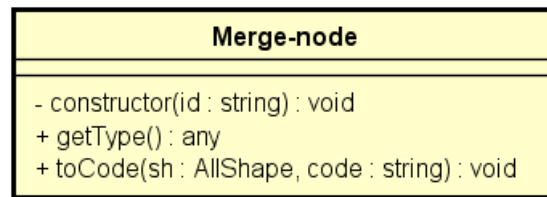


Figura 21: Diagramma della classe Merge-node

* *-constructor(id: string)*

Costruttore della classe

Parametri:

· *id: string*

Id della shape

* *+getType()*

Ritorna il tipo della classe

* *+toCode(sh: AllShape, code: string)*

Converte la shape in codice

Parametri:

· *sh: AllShape*

Shape da convertire

· *code: string*

Stringa di codice

3.5.2.16 SWEDesigner::Client::Components::Editor-container::Editor::Metodo

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-nome: string*

Nome del metodo

* *-accesso: string*

Visibilità del metodo

* *-tipoRitorno: string*

Tipo di ritorno del metodo

Metodo
<ul style="list-style-type: none"> - nome : String - accesso : String - tipoRitorno : String - listaArgomenti : Param[] - diagramma : JSON - shapeList : AllShape - mapVarVisibili : Map<string,string> - statico : boolean - main : boolean
<ul style="list-style-type: none"> - constructor(stat : boolean, costr : boolean, nome : string, acc : string, tipo : string, listaArg : Param[]) : void + changeNome(name : string) : void + changeTipoRitorno(tipo : string) : void + changeAccesso(acc : string) : void + changeListaArg(listArg : Param[]) : void + addArgomento(arg : Param) : void + addDiagram(dia : JSON) : void + getDiagram() : any + getNome() : String + getAccesso() : String + getTipoRitorno() : String + getListaArgomenti() : Attributo[] + getShapeList() : any + getMapVars() : any + isConstructor() : boolean + isStatic() : boolean + staticString() : void + setVars(vars : Map<string,string>) : void + paramToString() : void

Figura 22: Diagramma della classe Metodo

- * *-listaArgomenti: Param[]*
Lista di argomenti del metodo
- * *-diagramma: JSON*
Definisce il metodo corrente in formato JSON
- * *-shapeList: AllShape*
Lista delle shapes
- * *-statico: boolean*
True se il metodo è marcato static
- * *-main: boolean*

True se il metodo è main

– **Metodi:**

- * *-constructor(stat: boolean, costr: boolean, nome: string, acc: string, tipo: string, listaArg: Param[])*

Costruisce il metodo

Parametri:

- *stat: boolean*
True se il metodo deve essere marcato static
- *costr: boolean*
True se il metodo è un costruttore
- *nome: string*
Nome del metodo da creare
- *acc: string*
Visibilità del metodo
- *tipo: string*
Tipo di ritorno del metodo
- *listaArg: Param[]*
Lista degli argomenti del metodo

- * *+changeNome(name: string)*
Modifica il nome del metodo selezionato

Parametri:

- *name: string*
Nuovo nome del metodo

- * *+changeTipoRitorno(tipo: string)*
Modifica il tipo di ritorno del metodo

Parametri:

- *tipo: string*
Tipo di ritorno

- * *+changeAccesso(acc: string)*
Modifica la visibilità del metodo

Parametri:

- *acc: string*
Visibilità del metodo

- * *+changeListaArg(listaArg: Param[])*
Modifica la lista degli argomenti del metodo

Parametri:

- *listArg: Param[]*
Lista degli argomenti

- * *+addArgomento(arg: Param)*
Aggiunge un argomento al metodo

Parametri:

- *arg: Param*
Argomento da aggiungere

- * *+addDiagram(dia: JSON)*
Assegna il file JSON all'attributo diagramma della classe

Parametri:

- *dia: JSON*
File JSON

- * *+getDiagram()*
Ritorna l'attributo diagramma

- * *+getNome()*
Ritorna il nome del metodo

- * *+getAccesso()*
Ritorna la visibilità del metodo

- * *+getTipoRitorno()*
Ritorna il tipo di ritorno del metodo

- * *+getListArgomenti()*
Ritorna la lista degli argomenti del metodo

- * *+getShapeList()*
Ritorna la lista della shape

- * *+isConstructor()*
True se è un costruttore

- * *+isStatic()*
True se è marcato static

- * *+staticString()*
Aggiunge la stringa static al metodo

- * *+setVars(vars: Map<string, string>)*
Setta la dichiarazione delle variabili nel metodo

Parametri:

- *vars: Map<string, string>*
Variabili

* *+paramToString()*

Traduce i parametri dei metodi in string

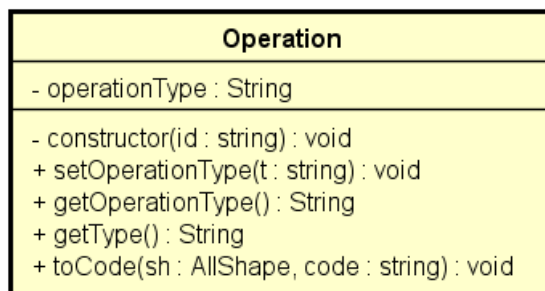


Figura 23: Diagramma della classe Operation

3.5.2.17 SWEDesigner::Client::Components::Editor-container::Editor::Operation

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-operationType : string*

Tipo di operazione

– **Metodi:**

* *-constructor(id: string)*

Costruttore della classe

Parametri:

· *id: string*

Id dell'elemento

* *+setOperationType(t: string)*

Setta l'attributo operationType

Parametri:

· *t: string*

Nuovo valore dell'attributo

* *+getOperationType()*

Ritorna il valore di operationType

- * *+getType()*
Ritorna il tipo dell'operazione
- * *+toCode(sh: AllShape, code: string)*
Converte la shape in codice

Parametri:

- *sh: AllShape*
Shape da convertire
- *code: string*
Stringa di codice

3.5.2.18 SWEDesigner::Client::Components::Editor-container::Editor::Operazione

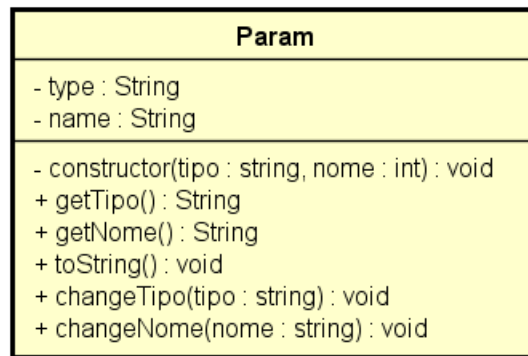


Figura 24: Diagramma della classe Param

3.5.2.19 SWEDesigner::Client::Components::Editor-container::Editor::Param

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

- * *-type: string*
Tipo del parametro
- * *-name: string*
Nome del parametro

– **Metodi:**

* *-constructor(tipo: string, nome: string)*

Costruisce un nuovo parametro

Parametri:

· *tipo: string*

Tipo del parametro

· *nome: string*

Nome del parametro

* *+getTipo()*

Ritorna il tipo del parametro

* *+getNome()*

Ritorna il nome del parametro

* *+toString()*

Tramuta il parametro in string

* *+changeTipo(tipo: string)*

Modifica il tipo del parametro

Parametri:

· *tipo: string*

Tipo del parametro

* *+changeNome(nome: string)*

Modifica il nome del parametro

Parametri:

· *nome: string*

Nome del parametro

3.5.2.20 SWEDesigner::Client::Components::Editor-container::Editor::Shape

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-id: string*

Id della shape

* *-succ: string*

Elemento linkato successivo

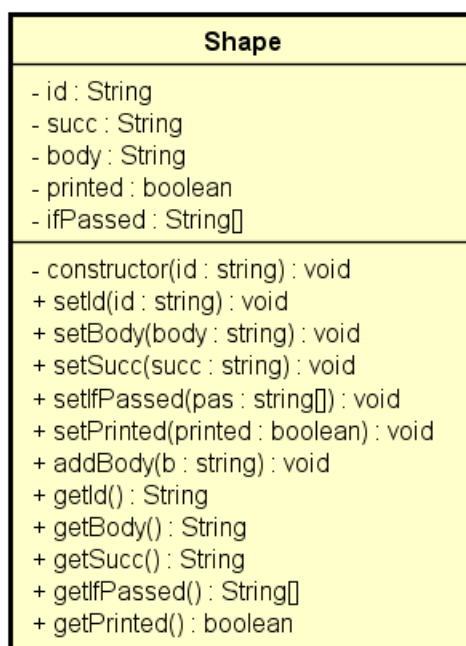


Figura 25: Diagramma della classe Shape

- * *-body: string*
Body della shape
- * *-printed: boolean*
True se la shape è stampata
- * *-ifPassed: String[]*
Rappresenta il codice generato

– Metodi:

- * *-constructor(id: string)*
Costruttore della classe

Parametri:

- *id: string*
Id della shape

- * *+setId(id: string)*
setta l'id della shape

Parametri:

- *id: string*
Id della shape

* *+setBody(body: string)*

Setta il body della shape

Parametri:

· *body: string*

Body della shape

* *+setSucc(succ: string)*

Setta l'elemento linkato successivamente alla shape

Parametri:

· *succ: string*

Elemento successivo

* *+setIfPassed(pas: string[])*

Setta l'attributo ifPassed

Parametri:

· *pas: string[]*

Nuovo valore dell'attributo

* *+setPrinted(printed: boolean)*

Setta il valore di printed

Parametri:

· *printed: boolean*

Valore dell'attributo

* *+addBody(b: string)*

Aggiunge un corpo alla shape

Parametri:

· *b: string*

Corpo da aggiungere

* *+getId()*

Ritorna l'attributo id della shape

* *+getBody()*

Ritorna l'attributo body della shape

* *+getSucc()*

Ritorna l'attributo succ della shape

* *+getIfPassed()*

Ritorna l'attributo ifPassed della shape

* *+getPrinted()*

Ritorna l'attributo printed della shape

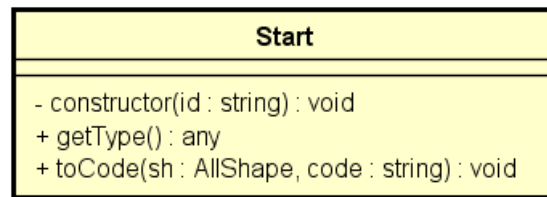


Figura 26: Diagramma della classe Start

3.5.2.21 SWEDesigner::Client::Components::Editor-container::Editor::Start

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(id : string)*

Costruttore della classe

Parametri:

· *id : string*

Id della shape

* *+getType()*

Ritorna il tipo della shape

* *+toCode(sh: AllShape, code: string)*

Traduce la shape in codice

Parametri:

· *sh: AllShape*

Shape da tradurre

· *code: string*

Stringa di codice

3.5.2.22 SWEDesigner::Client::Components::Editor-container::Editor::Variabile

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

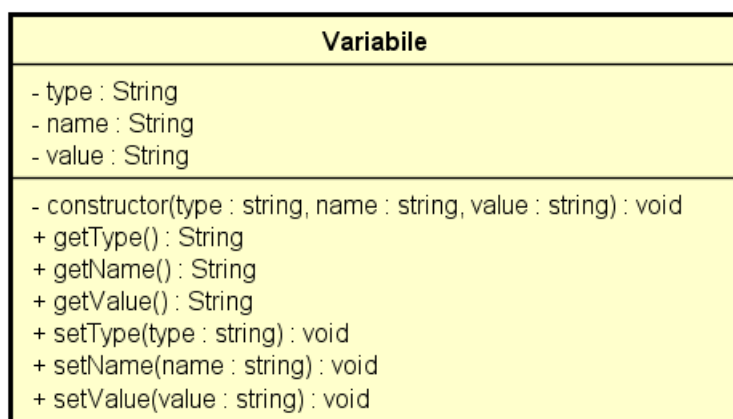


Figura 27: Diagramma della classe Variabile

- * *-type: string*
Rappresenta il tipo della variabile
- * *-name: string*
Rappresenta il nome della variabile
- * *-value: string*
Rappresenta il valore della variabile

– **Metodi:**

- * *-constructor(type: string, name: string, value: string)*
Costruttore della classe

Parametri:

- *type: string*
Inizializza il tipo
- *name: string*
Inizializza il nome
- *value: string*
Inizializza il valore
- * *+getType()*
Ritorna il tipo della variabile
- * *+getName()*
Ritorna il nome della variabile
- * *+getValue()*
Ritorna il valore della variabile

- * *+setType(type: string)*
Setta il tipo della variabile
Parametri:
 - *type: string*
Tipo della variabile
- * *+setName(name: string)*
Setta il nome della variabile
Parametri:
 - *name: string*
Nome della variabile
- * *+setValue(value: string)*
Setta il valore della variabile
Parametri:
 - *value: string*
Valore della variabile

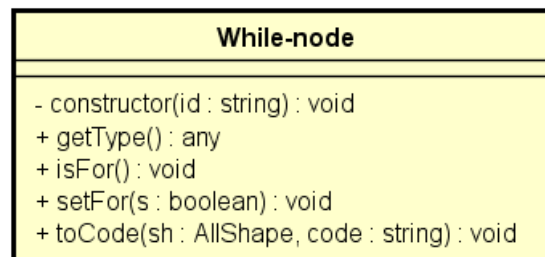


Figura 28: Diagramma della classe While-node

3.5.2.23 SWEDesigner::Client::Components::Editor-container::Editor::While-node

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

- * *-constructor(id: string)*
Costruttore della classe
Parametri:
 - *id: string*
Id della shape

- * *+getType()*
Ritorna il tipo della shape
- * *+isFor()*
Ritorna il valore del for
- * *+setFor(s: boolean)*
Setta il for
Parametri:
 - *s: boolean*
Valore del for
- * *+toCode(sh: AllShape, code: string)*
Traduce la shape in codice
Parametri:
 - *sh: AllShape*
Shape da tradurre
 - *code: string*
Stringa di codice

3.6 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu

3.6.1 Informazioni generali

- **Descrizione:**
- **Padre:** SWEDesigner::Client::Components::Editor-container::Editor

3.6.2 Classi

3.6.2.1 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

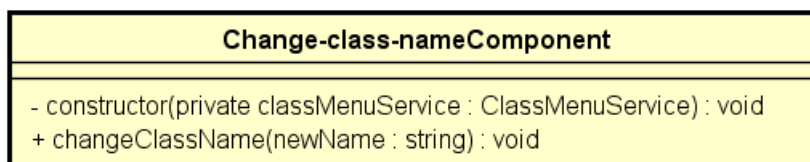


Figura 29: Diagramma della classe Change-class-name

3.6.2.2 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(private classMenuService: ClassMenuService)*

Costruttore della classe

Parametri:

· *classMenuService: ClassMenuService*

Serve per creare un istanziazione di ClassMenuService

* *+changeClassName(newName: string)*

Modifica il nome della classe

Parametri:

· *newName: string*

Nome della classe

Class-add-attributeComponent
- constructor(private classMenuService : ClassMenuService) : void + justOneCheckbox(event : any) : boolean + addAttributo(nome : string, staticAtt : boolean, finalAtt : boolean, tipo : string, acc : string) : void

Figura 30: Diagramma della classe Class-add-attribute

3.6.2.3 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(private classMenuService: ClassMenuService)*

Costruttore della classe

Parametri:

- *classMenuService: ClassMenuService*
Serve per creare un istanziazione di ClassMenuService

- * *+justOneCheckbox(event: any)*
Controlla che ci sia solo un elemento sulla checkbox attributo

Parametri:

- *event: any*
Nome dell'elemento

- * *+addAttributo(nome: string, staticAtt: boolean, finalAtt: boolean, tipo: string, acc: string)*
Aggiunge un nuovo attributo

Parametri:

- *nome: string*
Nome dell'attributo
- *staticAtt: boolean*
True se è marcato static
- *finalAtt: boolean*
True se è marcato final
- *tipo: string*
Tipo dell'attributo
- *acc: string*
Visibilità dell'attributo

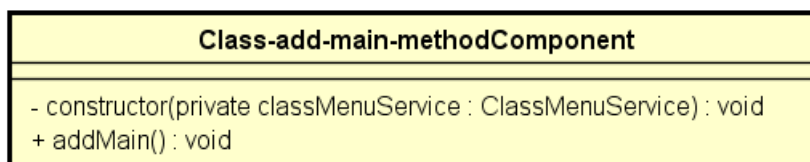


Figura 31: Diagramma della classe Class-add-main-method

3.6.2.4 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(private classMenuService: ClassMenuService)*

Costruttore della classe

Parametri:

· *classMenuService: ClassMenuService*

Crea un'istanza di ClassMenuService

* *addMain()*

Aggiunge il metodo main alla classe selezionata

Class-add-methodComponent
- constructor(private classMenuService : ClassMenuService) : void + addParam(type : string, name : string) : void + removeParam(type : string, name : string) : void + addMetodo(nome : string, staticMet : boolean, constructor : boolean, tipo : string, acc : string) : void + justOneCheckbox(event : any) : boolean

Figura 32: Diagramma della classe Class-add-method

3.6.2.5 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(private classMenuService: ClassMenuService)*

Costruttore della classe

Parametri:

· *classMenuService: ClassMenuService*

Crea un'istanza di ClassMenuService

* *+addParam(type: string, name: string)*

Aggiunge un parametro alla lista dei parametri del metodo

Parametri:

· *type: string*

Tipo del parametro

· *name: string*

Nome del parametro

* *+removeParam(type: string, name: string)*

Rimuove un parametro dalla lista dei parametri del metodo

Parametri:

- *type: string*
Tipo del parametro
- *name: string*
Nome del parametro

* *+addMetodo(nome: string, staticMet: boolean, constructor: boolean, tipo: string, acc: string)*

Aggiunge un nuovo metodo

Parametri:

- *nome: string*
Nome del metodo
- *staticMet: boolean*
True se è marcato static
- *constructor: boolean*
True se è un costruttore
- *tipo: string*
Tipo di ritorno del metodo
- *acc: string*
Visibilità del metodo

* *+justOneCheckbox(event: any)*

Controlla che ci sia solo un elemento sulla checkbox

Parametri:

- *event: any*
Nome dell'elemento

Class-list-attributeComponent
- constructor(private classMenuService : ClassMenuService) : void + changeAttributo(newName : string, oldName : string, tipo : string, acc : string, stat : boolean, final : boolean) : void + justOneCheckbox(event : any) : boolean

Figura 33: Diagramma della classe Class-list-attribute

3.6.2.6 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

Class-list-attributeComponent
- constructor(private classMenuService : ClassMenuService) : void + changeAttributo(newName : string, oldName : string, tipo : string, acc : string, stat : boolean, final : boolean) : void + justOneCheckbox(event : any) : boolean

Figura 34: Diagramma della classe Class-list-attribute

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(private classMenuService: ClassMenuService)*

Costruttore della classe

Parametri:

· *classMenuService: ClassMenuService*

Crea un istanziazione di ClassMenuService

* *+changeAttributo(newName: string, oldName: string, tipo: string, acc: string, stat: boolean, final: boolean)*

Modifica le proprietà di un attributo

Parametri:

· *newName: string*

Nuovo nome dell'attributo

· *oldName: string*

Vecchio nome dell'attributo

· *tipo: string*

Tipo dell'attributo

· *acc: string*

Visibilità dell'attributo

· *stat: boolean*

True se è marcato static

· *final: boolean*

True se è marcato final

* *+justOneCheckbox(event: any)*

Controlla che ci sia solo un elemento sulla checkbox

Parametri:

· *event: any*

Nome dell'elemento

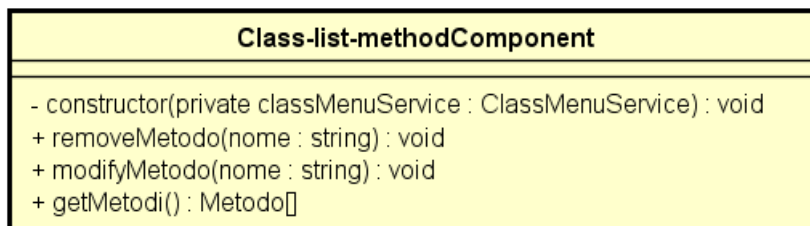


Figura 35: Diagramma della classe Class-list-method

3.6.2.7 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(private classMenuService: ClassMenuService)*

Costruttore della classe

Parametri:

· *classMenuService: ClassMenuService*

Crea un istanziazione di ClassMenuService

* *+removeMetodo(nome: string)*

Rimuove un metodo

Parametri:

· *nome: string*

Nome del metodo da rimuovere

* *+modifyMetodo(nome: string)*

Modifica il corpo del metodo

Parametri:

· *nome: string*

Nome del metodo

- * *+getMetodi()*
Ritorna la lista dei metodi

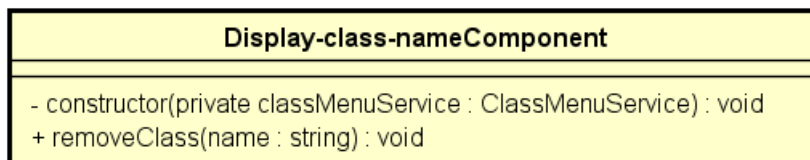


Figura 36: Diagramma della classe Display-class-name

3.6.2.8 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

- * *-constructor(private classMenuService: ClassMenuService)*
Costruttore della classe

Parametri:

- *classMenuService: ClassMenuService*
Crea un istanziazione di ClassMenuService

- * *+removeClass(name: string)*
Rimuove la classe selezionata

Parametri:

- *name: string*
Nome della classe da eliminare

3.7 SWEDesigner::Client::Services

3.7.1 Informazioni generali

– **Descrizione:**

Il package contiene i servizi per le operazioni di iterazione tra i component e il server.

– **Padre:** SWEDesigner::Client

– Package contenuti:

* Models

Il package contiene moduli necessari a storicizzare i dati inseriti all'interno dei diagrammi.

3.7.2 Classi

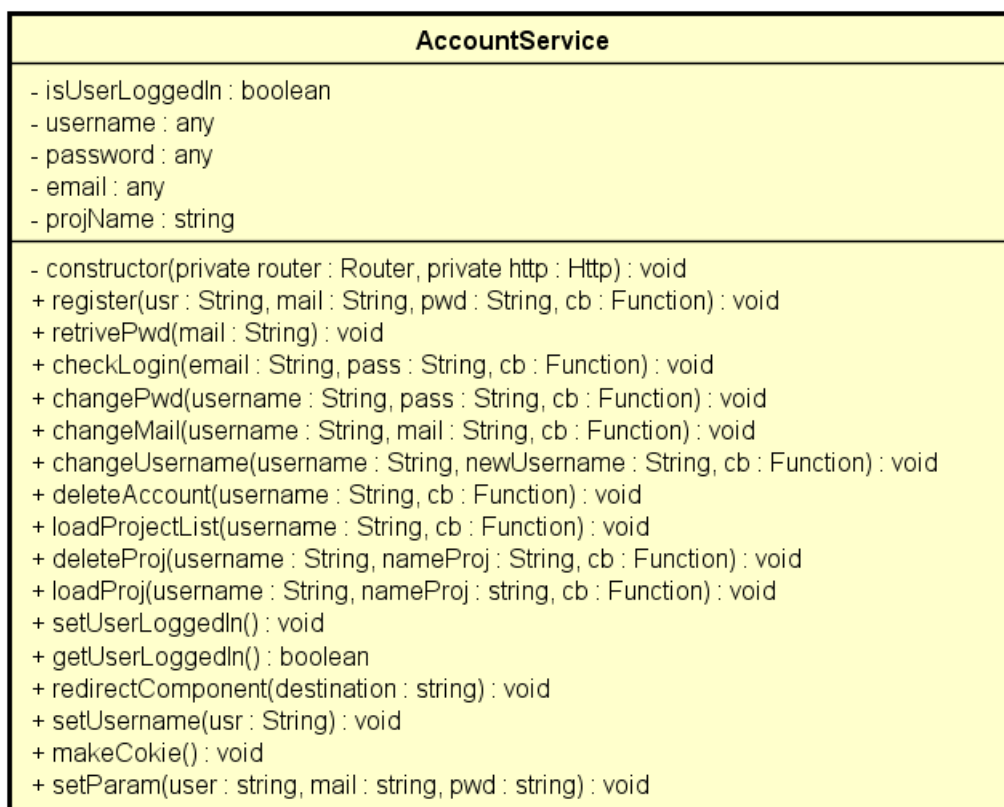


Figura 37: Diagramma della classe AccountService

3.7.2.1 SWEDesigner::Client::Services::AccountService

– Descrizione:

– Utilizzo:

– Attributi:

- * *-isUserLoggedIn: boolean*
Serve a controllare se l'utente è autenticato
- * *-username: any*
Contiene l'username dell'utente
- * *-password: any*
Contiene la password dell'utente
- * *-email: any*
Contiene l'email dell'utente
- * *-notOpenedProj: boolean*
Controlla se il progetto è aperto o meno
- * *-projName: string*
Contiene il nome del progetto attualmente aperto

– **Metodi:**

- * *-constructor(private router: Router, private http: Http)*
Costruttore della classe
- Parametri:**
 - *router: Router*
Crea una nuova istanziazione di Router
 - *http: Http*
Crea una nuova istanziazione di Http
- * *+register(usr: String, mail: String, pwd: String, cb: Function)*
Serve per registrare un nuovo utente

Parametri:

- *usr: String*
Nome utente
 - *mail: String*
Email dell'utente
 - *pwd: String*
Password dell'utente
 - *cb: Function*
- * *+retrivePwd(mail: String)*
Serve per recuperare la password di un utente

Parametri:

- *mail: String*
Email a cui mandare la password

* *+checkLogin(email: String, pass: String, cb: Function)*
Serve per effettuare l'autenticazione di un utente

Parametri:

- *email: String*
Email dell'utente
- *pass: String*
Password dell'utente
- *cb: Function*

* *+changePwd(username: String, pass: String, cb: Function)*
Serve per modificare la password di un utente

Parametri:

- *username: String*
Nome utente
- *pass: String*
Password dell'utente
- *cb: Function*

* *+changeMail(username: String, mail: String, cb:Function)*
Serve per modificare la mail dell'utente

Parametri:

- *username: String*
Nome utente
- *mail: String*
Email dell'utente
- *String, cb:Function*

* *+changeUsername(username: String, newUsername: String, cb: Function)*

Serve per modificare l'username dell'utente

Parametri:

- *username: String*
Nome utente

- *newUsername: String*
Nuovo username

- *cb: Function*

* *+deleteAccount(username: String, cb: Function)*
Serve per eliminare un account

Parametri:

- *username: String*
Nome utente
- *cb: Function*

* *+loadProjectList(username: String, cb: Function)*
Serve a caricare la lista dei progetti

Parametri:

- *username: String*
Nome utente
- *cb: Function*

* *+deleteProj(username: String, nameProj: String, cb: Function)*
Serve per eliminare un progetto

Parametri:

- *username: String*
Nome utente
- *nameProj: String*
Nome del progetto
- *cb: Function*

* *+loadProj(username: String, nameProj: string, cb: Function)*
Carica un progetto dalla lista progetti dell'utente

Parametri:

- *username: String*
Nome utente
- *nameProj: string*
Nome del progetto
- *cb: Function*

* *+setUserLoggedIn()*

Modifica lo stato di autenticazione dell'utente

* *+getUserLoggedIn()*

Ritorna true se l'utente è autenticato

* *+redirectComponent(destination:string)*

Questa funzione reindirizza questo componente al componente destinazione

Parametri:

· *destination:string*

Componente destinazione

* *+setUsername(usr: String)*

Setta l'username dell'utente

Parametri:

· *usr: String*

Nome utente

* *+makeCookie()*

Costruisce dei cookie di sessione contenenti le informazioni dell'utente

* *+setParam(user: string, mail:string, pwd: string)*

Setta le informazioni in AccountService

Parametri:

· *user: string*

Nome utente

· *mail:string*

Email dell'utente

· *pwd: string*

Password dell'utente

* *+logout()*

Esegue il logout

3.7.2.2 SWEDesigner::Client::Services::Main-editorService

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

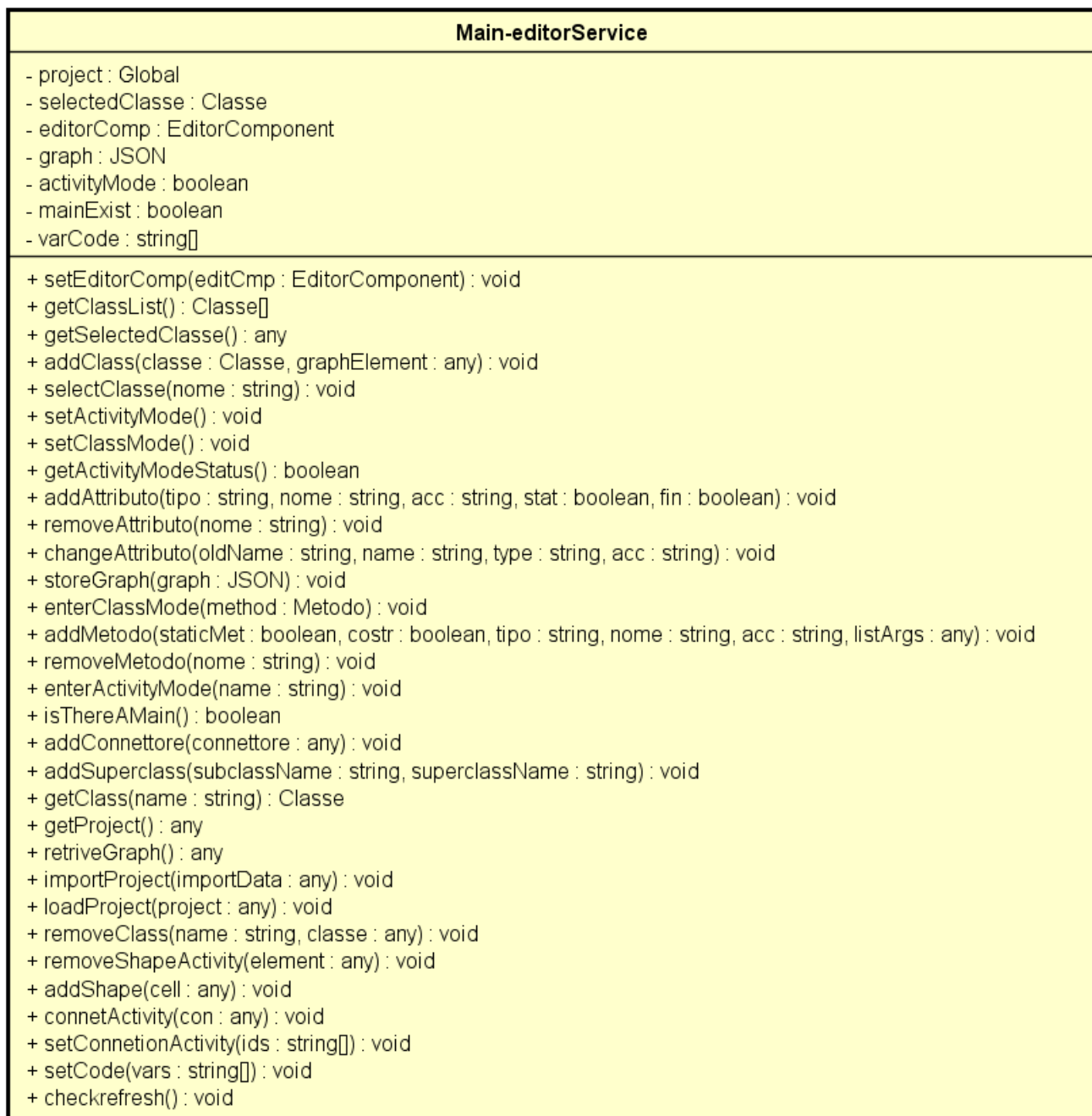


Figura 38: Diagramma della classe Main-editorService

* *-project: Global*

Serve per memorizzare informazioni riguardo il progetto corrente

- * *-selectedClasse: Classe*
Memorizza la classe corrispondente nel canvas dell'editor
- * *-editorComp: EditorComponent*
Serve per accedere direttamente all'EditorComponent
- * *-graph: JSON*
Serve per memorizzare il grafico dell'editor
- * *-activityMode: boolean*
Indica se è in uso l'activity diagram
- * *-mainExist: boolean*
Indica se esiste il metodo main

– **Metodi:**

- * *+setEditorComp(editCmp: EditorComponent)*
Serve per costruire un'istanza dell'EditorComponent

Parametri:

- *editCmp: EditorComponent*
Istanza EditorComponent
- * *+getClassList()*
Ritorna la lista delle classi presenti nel progetto
- * *+getSelectedClasse()*
Ritorna la classe selezionata
- * *+addClass(classe: Classe, graphElement: any)*
Aggiunge un oggetto di tipo classe

Parametri:

- *classe: Classe*
Classe da aggiungere
- *graphElement: any*
Elemento della libreria grafica
- * *+selectClasse(nome: string)*
Cerca una classe all'interno della lista

Parametri:

- *nome: string*
Nome della classe da cercare
- * *+setActivityMode()*
Passa alla modalità activity diagram

- * *+setClassMode()*
Passa alla modalità class diagram
- * *+getActivityModeStatus()*
Ritorna il valore del fral activityMode
- * *+addAttributo(tipo: string, nome:string, acc: string, stat: boolean, fin: boolean)*
Aggiunge un metodo alla selectedClasse

Parametri:

- *tipo: string*
Tipo dell'attributo
- *nome:string*
Nome dell'attributo
- *acc: string*
Visibilità dell'attributo
- *stat: boolean*
True se è marcato static
- *fin: boolean*
True se è marcato final

- * *+removeAttributo(nome: string)*
Rimuove un attributo dalla selectedClasse

Parametri:

- *nome: string*
Nome dell'attributo da rimuovere

- * *+changeAttributo(oldName: string, name: string, type: string, acc: string)*
Modifica un attributo della selectedClasse

Parametri:

- *oldName: string*
Vecchio nome
- *name: string*
Nuovo nome
- *type: string*
Tipo dell'attributo
- *acc: string*
Visibilità dell'attributo

* *+storeGraph(graph: JSON)*

Memorizza il grafico

Parametri:

- *graph: JSON*

Grafico da memorizzare

* *+enterClassMode(method: Metodo)*

Serve per ripristinare il diagramma delle classi dopo aver definito un metodo

Parametri:

- *method: Metodo*

Metodo definito

* *+addMetodo(staticMet: boolean, costr: boolean, tipo: string, nome:string, acc: string, listArgs: any)*

Aggiunge un nuovo metodo alla selectedClasse

Parametri:

- *staticMet: boolean*

True se è marcato static

- *costr: boolean*

True se è un costruttore

- *tipo: string*

Tipo di ritorno del metodo

- *nome:string*

Nome del metodo

- *acc: string*

Visibilità del metodo

- *listArgs: any*

Lista degli argomenti del metodo

* *+removeMetodo(nome: string)*

Rimuove un metodo dalla selectedClasse

Parametri:

- *nome: string*

Nome del metodo da eliminare

* *+enterActivityMode(name: string)*

Entra nella modalità activity per modificare il corpo un metodo

Parametri:

- *name: string*
Nome del metodo da modificare

* *+isThereAMain()*
Ritorna true se è presente il metodo main nella selectedClasse

* *+addConnettore(connettore: any)*
Aggiunge un connettore alla selectedClasse

Parametri:

- *connettore: any*
Connettore da aggiungere

* *+addSuperclass(subclassName: string, superclassName: string)*
Aggiunge una classe padre

Parametri:

- *subclassName: string*
Classe figlia
- *superclassName: string*
Classe padre

* *+getClass(name: string)*
Ritorna la classe selezionata

Parametri:

- *name: string*
Nome della classe

* *+getProject()*
Ritorna la lista dei progetti

* *+retriveGraph()*
Ritorna tutte le shape nel diagramma

* *+importProject(importData: any)*
Importa un progetto da un file JSON

Parametri:

- *importData: any*
File JSON conenente il progetto

* *+loadProject(project: any)*
Carica un progetto dal database

Parametri:

- *project: any*
Progetto da caricare

* *+removeClass(name: string, classe: any)*

Rimuove una classe

Parametri:

· *name: string*

Nome della classe

· *classe: any*

Riferimento alla shape della classe

* *+removeShapeActivity(element: any)*

Rimuove una shape dall'activity

Parametri:

· *element: any*

Shape da rimuovere

* *+addShape(cell: any)*

Aggiunge una shape al diagramma

Parametri:

· *cell: any*

Shape da aggiungere

* *+connetActivity(con: any)*

Connette le shape dell'activity

Parametri:

· *con: any*

Shape da connettere

* *+setCode(vars: string[])*

Memorizza tutti i diagrammi tradotti

Parametri:

· *vars: string[]*

Diagrammi tradotti

* *+checkrefresh()*

Controlla di poter effettuare il refresh della finestra

3.7.2.3 SWEDesigner::Client::Services::MenuService

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

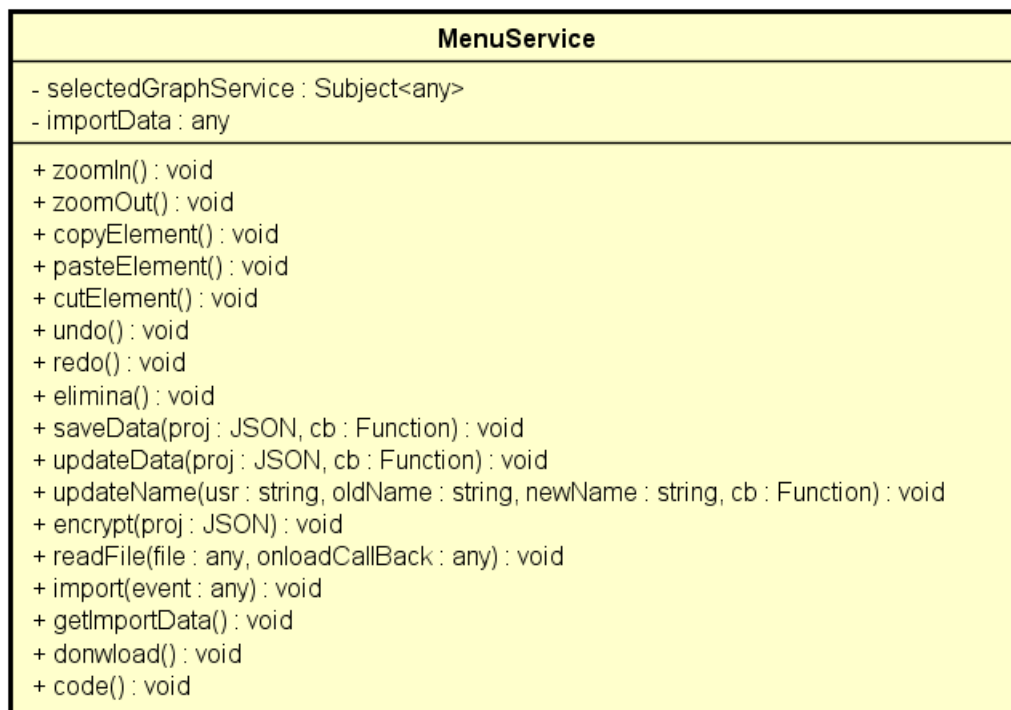


Figura 39: Diagramma della classe MenuService

* *-selectedGraphService: Subject<any>*
Memorizza l'array con tutte le shape

* *-importData: any*
Serve per importare il progetto

– **Metodi:**

* *+zoomIn()*
Esegue lo zoom in avanti

* *+zoomOut()*
Esegue lo zoom all'indietro

* *+copyElement()*
Copia l'elemento selezionato

* *+pasteElement()*
Incolla l'elemento copiato/tagliato

* *+cutElement()*
Taglia l'elemento selezionato

- * *+undo()*
Annulla l'ultima operazione
- * *+redo()*
Ripristina l'azione annullata
- * *+elimina()*
Elimina l'elemento selezionato
- * *+saveData(proj: JSON, cb: Function)*
Richiede al server dei dati del progetto corrente memorizzati nel database
Parametri:
 - *proj: JSON*
Progetto corrente
 - *cb: Function*
Funzione
- * *+updateData(proj: JSON, cb: Function)*
Aggiorna i dati del progetto corrente nel database
Parametri:
 - *proj: JSON*
Progetto corrente
 - *cb: Function*
Funzione
- * *+updateName(usr: string, oldName: string, newName: string, cb: Function)*
Aggiorna il nome del progetto corrente
Parametri:
 - *usr: string*
Nome utente
 - *oldName: string*
Vecchio nome del progetto
 - *newName: string*
Nuovo nome del progetto
 - *cb: Function*
Funzione
- * *+encrypt(proj: JSON)*
Richiede al server la funzione di criptazione
Parametri:

- *proj: JSON*
Progetto da criptare

- * *+readFile(file: any, onloadCallBack: any)*
Legge un file esterno

Parametri:

- *file: any*
File da caricare
- *onloadCallBack: any*
Funzione

- * *+import(event: any)*
Importa un file esterno

Parametri:

- *event: any*
File da importare

- * *+getImportData()*
Ritorna importData

- * *+download()*
Richiede al server la funzione di parsing e download

- * *+code()*
Richiama la funziona di download

4 Specifica Back-End

4.1 SWEDesigner::Server

4.1.1 Informazioni generali

- **Descrizione:**
Questo package contiene tutte le componenti del server scritte in JavaScript.
- **Padre:** SWEDesigner
- **Package contenuti:**
 - * Controller
Questo package contiene al suo interno tutti i controller che implementano il pattern MVVM fornito da *Angular.js*. In particolare sono contenuti i Middleware e tutti i Servizi da essi utilizzati.
 - * Model
Questo package contiene tutte le classi utili per la creazione del database, la connessione ad esso e le relative interrogazioni.

4.1.2 Classi

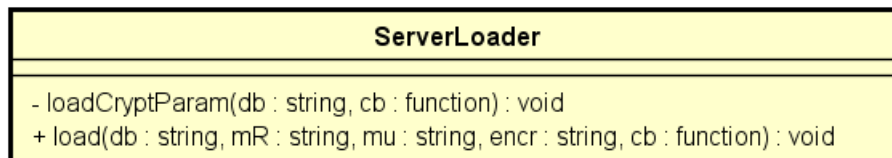


Figura 40: Diagramma della classe SWEDesigner::Server::serverLoader

4.1.2.1 SWEDesigner::Server::serverLoader

- **Descrizione:**
Classe che consente il caricamento di tutte le componenti e gli elementi utili al primo avvio dell'applicazione
- **Utilizzo:**
La classe viene utilizzata per il caricamento del server e di tutti i suoi elementi.
- **Metodi:**

* *+ load(db: string, mR: string, mu: string, encr: string, cb: function): void*

Si tratta della funzione principale che si occupa di chiamare i metodi load contenuti in tutte le altre classi.

* **Parametri:**

- *db: string*

Il path del modulo che gestisce la connessione al database.

- *mR: string*

Il path del modulo che gestisce le query.

- *mu: string*

Il path del modulo che gestisce il servizio di parsing.

- *encr: string*

Il path del modulo che gestisce il servizio di encrypt.

- *cb: function* Callback che gestisce le richieste asincrone al database.

* **- loadCryptParam(db: string, cb: function): void**

Si tratta della funzione utilizzata da load per la richiesta dei parametri crittografici al database.

* **Parametri:**

- *db: string*

Il path del modulo che gestisce la connessione al database.

- *cb: function* Callback che gestisce le richieste asincrone al database.

4.2 SWEDesigner::Server::Model

4.2.1 Informazioni generali

– **Descrizione:**

Questo package contiene tutte le classi e le funzionalità legate al database.

– **Padre:** SWEDesigner::Server

4.2.2 Classi

4.2.2.1 SWEDesigner::Server::Model::mongooseConnection

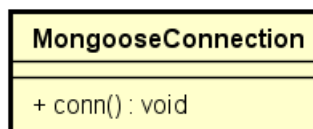


Figura 41: Diagramma della classe SWEDesigner::Server::Model::mongooseConnection

– **Descrizione:**

Classe che si occupa della connessione al database e degli errori che ne possono derivare

– **Utilizzo:**

La classe viene utilizzata per effettuare la connessione al database all'avvio dell'applicazione.

– **Metodi:**

* + conn() : void

Si tratta della funzione che effettua la connessione al database e ne gestisce gli eventuali errori derivanti.

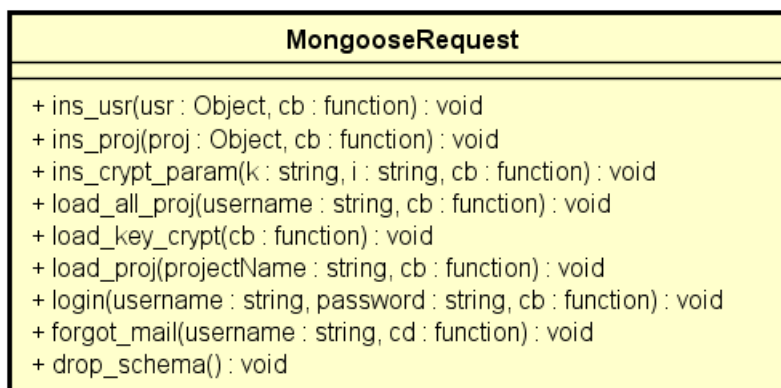


Figura 42: Diagramma della classe SWEDesigner::Server::Model::mongooseRequest

4.2.2.2 SWEDesigner::Server::Model::mongooseRequest

– **Descrizione:**

Classe che si occupa di gestire tutte le query da e verso il database.

– **Utilizzo:**

La classe viene utilizzata per tutte le richieste, inserimento e fetch, di dati dal e nel database.

– **Metodi:**

* *+ins_usr(usr: Object, cb: function) : void*

Si tratta della funzione che si occupa di inserire un utente all'interno del database.

* **Parametri:**

· *usr: Object*

L'utente, in formato JSON, da inserire all'interno dello schema.

· *cb: function*

Callback che gestisce le richieste asincrone al database.

* *+ins_proj(proj: Object, cb: function) : void*

Si tratta della funzione che si occupa di inserire un progetto all'interno del database.

* **Parametri:**

· *proj: Object*

Il progetto, in formato JSON, da inserire all'interno dello schema.

· *cb: function*

Callback che gestisce le richieste asincrone al database.

* *+ins_crypt_param(k: string, i: string, cb: function) : void*

Si tratta della funzione che si occupa di inserire una chiave crittografica all'interno del database.

* **Parametri:**

· *k: string*

La chiave crittografica.

· *i: string*

Valore iv per la crittografia.

· *cb: function*

Callback che gestisce le richieste asincrone al database.

* *+load_all_proj(username: string, cb: function) : void*

Si tratta della funzione che si occupa di richiedere tutti i progetti di un

dato utente.

*** Parametri:**

- *username: string*
Nome dell'utente di cui sono richiesti i progetti.
- *cb: function*
Callback che gestisce le richieste asincrone al database.

- * *+load_key_crypt(cb: function) : void*
Si tratta della funzione che si occupa di richiedere l'unica chiave crittografica salvata nel database.

*** Parametri:**

- *cb: function*
Callback che gestisce le richieste asincrone al database.

- * *+load_proj(projectName: string, cb: function) : void*
Si tratta della funzione che si occupa di cercare e ritornare un dato progetto.

*** Parametri:**

- *projectName: string*
Nome del progetto richiesto
- *cb: function*
Callback che gestisce le richieste asincrone al database.

- * *+login(username: string, password: string, cb: function) : void*
Si tratta della funzione che verifica che l'utente che cerca di loggare esiste all'interno del database.

*** Parametri:**

- *username: string*
L'username dell'utente che cerca di loggare.
- *password: string*
La password dell'utente che cerca di loggare.

- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+forgot_mail(username: string, cb: function)*
Si tratta della funzione che restituisce la mail dell'utente dato.

* **Parametri:**

- *username: string*
Nome dell'utente
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+update_mail(username: string, mail: string, cb: function)*
Si tratta della funzione che permette di aggiornare il campo mail di un utente.

* **Parametri:**

- *username: string*
Nome dell'utente
- *mail: string*
Nuova mail
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+update_password(username: string, password: string, cb: function)*
Si tratta della funzione che permette di aggiornare il campo password di un utente.

* **Parametri:**

- *username: string*
Nome dell'utente
- *password: string*
Nuova password
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+update_username(username: string, newUsername: string, cb: function)*

Si tratta della funzione che permette di aggiornare l'username di un utente.

* **Parametri:**

- *username: string*
Nome dell'utente
- *newUsername: string*
Nuovo username
- *cb: function*
Callback che gestisce le richieste asincrone al database.

- * ***+update_proj(projName: string, usr: string, proj: JSON, cb: function)***
Si tratta della funzione che permette di aggiornare il corpo di un progetto.

* **Parametri:**

- *projName: string*
Nome del progetto
- *usr: string*
Username dell'utente proprietario del progetto
- *proj: JSON*
Corpo del progetto
- *cb: function*
Callback che gestisce le richieste asincrone al database.

- * ***+update_nameProj(projName: string, usr: string, newName: string, cb: function)***
Si tratta della funzione che permette di aggiornare il nome di un progetto.

* **Parametri:**

- *projName: string*
Nome del progetto
- *usr: string*
Username dell'utente proprietario del progetto
- *newName: string*
Nuovo nome del progetto
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+login(mail: string, pwd: string, cb: function)*

Si tratta della funzione che permette di autenticarsi controllando che i dati richiesti dal client esistano nel database.

* **Parametri:**

- *mail: string*
E-mail dell'utente.
- *pwd: string*
Password dell'utente.
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+delete_user(username: string, cb: function)*

Si tratta della funzione che elimina un utente dal database.

* **Parametri:**

- *username: string*
Username dell'utente.
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+delete_proj(username: string, projName: string, cb: function)*

Si tratta della funzione che elimina un progetto dal database.

* **Parametri:**

- *username: string*
Username dell'utente.
- *projName: string*
Nome del progetto.
- *cb: function*
Callback che gestisce le richieste asincrone al database.

* *+drop_schema() : void*

Si tratta della funzione che elimina il database.

4.3 SWEDesigner::Server::Controller::Middleware

4.3.1 Informazioni generali

- **Descrizione:**
In questo package sono definite tutte le componenti middleware del server scritte in JavaScript.
- **Padre:** SWEDesigner::Server::Controller

4.3.2 Classi

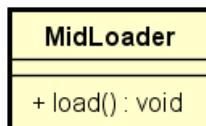


Figura 43: Diagramma della classe SWEDesigner::Server::Controller::Middleware::midLoader

4.3.2.1 SWEDesigner::Server::Controller::Middleware::midLoader

- **Descrizione:**
La classe contenente i metodi di caricamento dei servizi utilizzati dalle componenti middleware
- **Utilizzo:**
La classe viene utilizzata all'avvio dell'applicazione per caricare tutto ciò che serve per il funzionamento del middleware.
- **Metodi:**

* *+load() : void*
La funziona carica il servizio di parsing

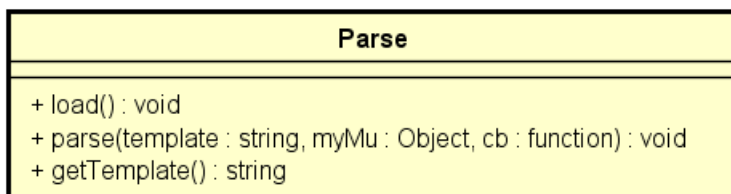


Figura 44: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Parse

4.3.2.2 SWEDesigner::Server::Controller::Middleware::Parse

– **Descrizione:**

La classe si occupa di gestire il caricamento del template e di richiamare il servizio di parsing

– **Utilizzo:**

La classe viene utilizzata sia per il caricamento del template all'avvio dell'applicazione, sia per richiamare il servizio di parsing quando il client lo richiede.

– **Metodi:**

* *+load() : void*

La funzione si occupa di ripulire la cache, compilare il template e caricarlo in cache.

* *+parse(template: Object, myMu: Object, cb: function) : void*

La funzione si occupa di richiamare la funzione di parsing del relativo servizio

* **Parametri:**

· *template: Object*

Il template precompilato da Moustache.

· *myMu: Object*

L'oggetto JSON di cui è necessario il parsing.

· *cb: function*

Callback che gestisce la chiamata asincrona al modulo di Moustache.

* *+getTemplate() : string*

La funzione ritorna il percorso in cui è contenuto il template, compilato o meno.

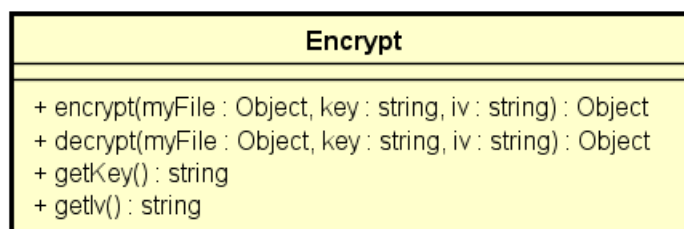


Figura 45: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Encrypt

4.3.2.3 SWEDesigner::Server::Controller::Middleware::Encrypt

– **Descrizione:**

La classe si occupa di gestire le funzionalità del servizio di encrypt.

– **Utilizzo:**

La classe viene utilizzata per chiamare le funzioni di encrypt del relativo servizio.

– **Metodi:**

* *+encrypt(myFile: Object, key: string, iv: string) : Object*

La funzione si occupa di richiamare la funzione di encrypt del relativo servizio e ritorna il file crittato correttamente.

* **Parametri:**

· *myFile: Object*

Oggetto JSON da crittare

· *key: string*

Chiave crittografica

· *iv: string*

IV necessario per la crittografia in AES

* *+decrypt(myFile: Object, key: string, iv: string) : Object*

La funzione si occupa di richiamare la funzione di decrypt del relativo servizio e ritorna il JSON decriptato.

* **Parametri:**

· *myFile: Object*

Oggetto JSON da crittare

· *key: string*

Chiave crittografica

· *iv: string*

IV necessario per la crittografia in AES

* *+getKey() : void*

La funzione si occupa di richiamare la funzione di generazione della chiave crittografica del relativo servizio.

* *+getI() : void*

La funzione si occupa di richiamare la funzione di generazione del valore iv per la crittografia del relativo servizio.

4.4 SWEDesigner::Server::Controller::Services

4.4.1 Informazioni generali

- **Descrizione:**
Questo package contiene tutti i servizi utilizzati dal middleware del server scritti in JavaScript.
- **Padre:** SWEDesigner::Server::Controller

4.4.2 Classi

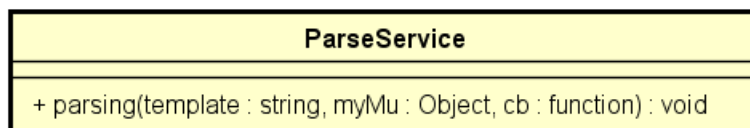


Figura 46: Diagramma della classe SWEDesigner::Server::Controller::Services::parseService

4.4.2.1 SWEDesigner::Server::Controller::Services::parseService

- **Descrizione:**
La classe si occupa di renderizzare il template pre-compilato e generare, così, un file scritto in Java.
- **Utilizzo:**
La classe viene utilizzata ogni volta che il client richiede la generazione di codice Java a partire dai diagrammi UML disegnati.
- **Metodi:**

* *+parsing(template: string, myMu: Object, cb: function) : void*

La funzione renderizza il template pre-compilato in fase di avvio dell'applicazione generando, a fronte dell'oggetto JSON inviato, un file in Java.

* **Parametri:**

· *template: string*

Il percorso del template precompilato da Moustache.

· *myMu: Object*

L'oggetto JSON di cui è necessario il parsing.

- *cb: function*
Callback che gestisce la chiamata asincrona al modulo di Moustahce.

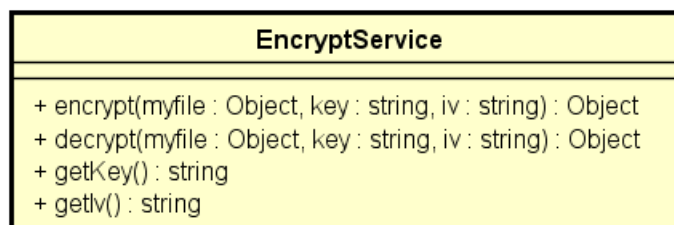


Figura 47: Diagramma della classe SWEDesigner::Server::Controller::Services::encryptService

4.4.2.2 SWEDesigner::Server::Controller::Services::encryptService

– **Descrizione:**

La classe si occupa di tutti i servizi legati alla crittografia.

– **Utilizzo:**

La classe viene utilizzata per generare le chiavi crittografiche da salvare nel database al primo avvio, qualora queste non esistessero, e di realizzare tutti i servizi legati alla crittografia, quindi encrypt e decrypt.

– **Metodi:**

- * *+encrypt(myFile: Object, key: string, iv: string) : Object*

La funzione si occupa di criptare il file in arrivo mediante codifica AES utilizzando gli algoritmi di Forge.

- * **Parametri:**

- *myFile: Object*
Oggetto JSON da crittare
- *key: string*
Chiave crittografica
- *iv: string*
IV necessario per la crittografia in AES

- * *+decrypt(myFile: Object, key: string, iv: string) : Object*

La funzione si occupa di decriptare il file in arrivo mediante gli algoritmi di Forge.

- * **Parametri:**

- *myFile: Object*
Oggetto JSON da crittare
- *key: string*
Chiave crittografica
- *iv: string*
IV necessario per la crittografia in AES
- * *+getKey() : string*
La funzione genera, tramite Forge, una chiave crittografica e la ritorna.
- * *+getIv() : string*
La funzione genera, tramite Forge, un gruppo di iv e lo ritorna.

5 Diagrammi di sequenza

5.1 Generazione Codice

Nel diagramma di sequenza di seguito è descritto il funzionamento di una richiesta di generazione di codice a partire da un progetto correttamente disegnato mediante l'applicazione.

Arrivata la richiesta a index.js, file sul Server che si occupa di gestire le richieste del Client attraverso le funzioni di routing di Express.js.

Express.js invia quindi una richiesta asincrona al Middleware che si occupa di richiedere, in maniera asincrona, ad un'istanza del servizio di parsing di effettuare l'operazione desiderata.

Ogni ritorno avviene tramite callback fino a index.js che, nuovamente tramite Express.js, restituisce un template correttamente compilato in modo tale da poter essere utilizzato per la generazione del codice Java.

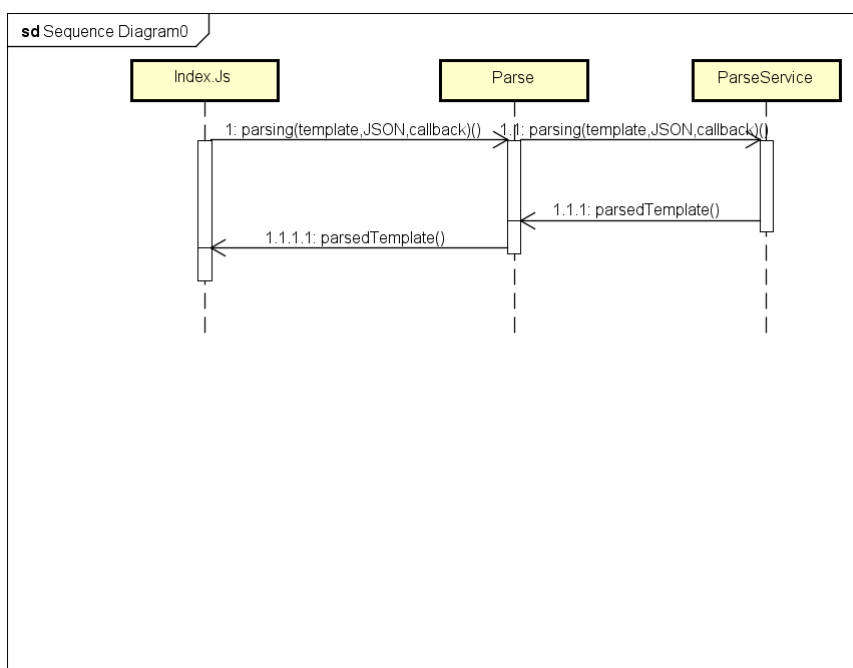


Figura 48: Sequence diagram generazione codice java

5.2 Caricamento moduli del Server

Nel diagramma di sequenza di seguito è descritto il funzionamento del caricamento di tutti i moduli del server che avviene, tipicamente, al primo avvio dell'applicativo.

Quello che accade è una singola richiesta di Load() che si occupa di chiamare il ServerLoader che, tramite Express.js, effettua tutte le chiamate ai singoli loader dei vari moduli.

Oltre all'istanza dei vari moduli presenti sul Server, vengono anche generate ed istanziate le chiavi crittografiche per la corretta gestione dei servizi di encrypt e decrypt.

Ultimo, ma non meno importante, è la renderizzazione del template di Moustache necessario al parsing e alla generazione del codice Java così da avere sempre a disposizione un template renderizzato, quindi utilizzare da Moustache, ogni volta che viene richiesta la generazione di codice.

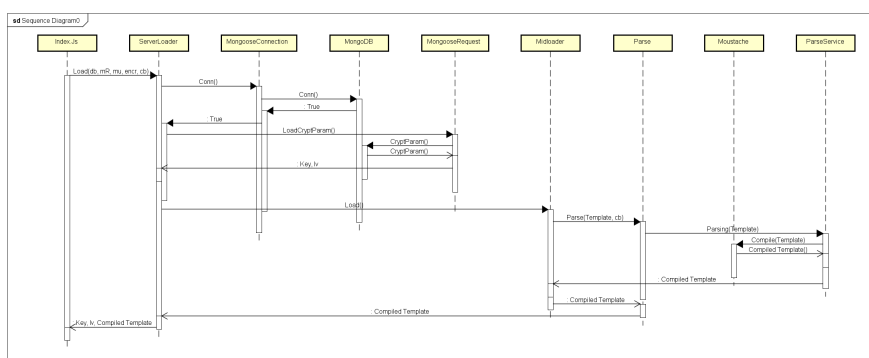


Figura 49: Sequence diagram caricamento moduli server

5.3 Encrypt/Decrypt

Il diagramma di seguito mostra il funzionamento dei servizi di Encrypt e Decrypt. Quello che accade è che Index.js, sempre tramite il routing di Express.js, effettua una richiesta di encrypting (o decrypting) al Middleware desiderato il quale, effettuerà una richiesta ad un'istanza del servizio desiderato.

Questo, tramite Forge, ritornerà semplicemente i file criptati o decrittati (a seconda della richiesta) al chiamante che gestirà il ritorno dell'informazione al Client.

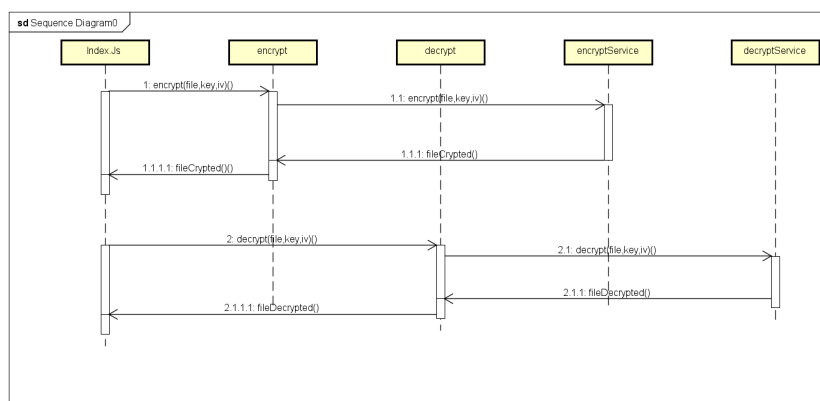


Figura 50: Sequence diagram per operazioni di encrypt e decrypt

6 Tracciamento

6.1 Tracciamento Classi-Requisiti

Componenti	Classi
SWEDesigner::Server::Model::mongooseRequest::dropSchema()	R0F6.1.1.4
SWEDesigner::Server::Model::mongooseRequest::forgotMail()	R1F13
SWEDesigner::Server::Model::mongooseRequest::inscryptparam()	R0F6.1.1.3
	R0F5.1.2
SWEDesigner::Server::Model::mongooseRequest::insproj()	R0F5.1
SWEDesigner::Server::Model::mongooseRequest::insUstr()	R0F1.1
	R0F1.2
	R0F1.3
SWEDesigner::Server::Model::mongooseRequest::loadAllProj()	R0F5
SWEDesigner::Server::Model::mongooseRequest::loadKeyCrypy()	R0F6.1.1.3
	R0F5.1.2
SWEDesigner::Server::Model::mongooseRequest::loadProj()	R0F5.1
SWEDesigner::Server::Model::mongooseRequest::login()	R0F2
SWEDesigner::Client::Components::Editor::ClassMenuComponent	R0F6.3.1.5.3
::addAttributo()	
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::addMe-	
todo()	
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::aggiungi-	
Param()	
SWEDesigner::Client::Components::Editor::ClassMenuComponent	R0F6.3.1.5.2
::changeAttributo()	
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::change-	R0F6.3.1.5.1
Nome()	

Componenti	Classi
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::modify-Metodo()	R0F6.3.1.5.4
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::removeAttributo	R0F6.3.1.11
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::remove-Metodo()	R0F6.3.1.10
SWEDesigner::Client::Components::Editor::EditorComponent ::addConnetto-re()	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::EditorComponent ::addElement()	
SWEDesigner::Client::Components::Editor::EditorComponent ::cloneElement()	R1F6.1.2.4 R1F6.1.2.5
SWEDesigner::Client::Components::Editor::EditorComponent ::constructor()	
SWEDesigner::Client::Components::Editor::EditorComponent ::elementSelec-tion()	R0F6.2.1.3.1 R0F6.2.1.3.2
SWEDesigner::Client::Components::Editor::EditorComponent ::replaceDiagram()	
SWEDesigner::Client::Components::Editor::EditorComponent ::selectElemen-tsToConnect()	R0F6.2.1.3.1 R0F6.2.1.3.2
SWEDesigner::Client::Components::Editor::EditorComponent::ZoomIn()	R0F6.3.2
SWEDesigner::Client::Components::Editor::EditorComponent::ZoomOut()	R0F6.3.3
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addAssocia-zione()	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addAstrat-ta()	R0F6.2.1.1 R0F6.3.1.5.6
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addClasse()	R0F6.2.1.1
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addCommento()	R0F6.3.1.8
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addConnet-tore	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addGenera-lizzazione()	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::ToolbarComponent::addImplementazione()	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::ToolbarComponent::addInterfaccia()	R0F6.2.1.1 R0F6.3.1.5.7
SWEDesigner::Client::Components::Menu::ModificaComponent::doZoomIN()	R0F6.3.2
SWEDesigner::Client::Components::Menu::ModificaComponent::DoZoomOut()	R0F6.3.3
SWEDesigner::Client::Services::Attributo::changeAccesso()	R0F6.3.1.5.2
SWEDesigner::Client::Services::Attributo::getAccesso()	
SWEDesigner::Client::Services::Classe::addAttributo()	R0F6.3.1.5.2
SWEDesigner::Client::Services::Classe::addMetodo()	R0F6.3.1.5.4
SWEDesigner::Client::Services::Classe::addSottoclasse()	

Componenti	Classi
SWEDesigner::Client::Services::Classe::changeAttr()	R0F6.3.1.5.2
SWEDesigner::Client::Services::Classe::changeNome()	R0F6.3.1.5.1
SWEDesigner::Client::Services::Classe::constructor()	
SWEDesigner::Client::Services::Classe::getAttributi()	R0F6.3.1.5.2
SWEDesigner::Client::Services::Classe::getMetodi()	R0F6.3.1.5.4
SWEDesigner::Client::Services::Classe::getNome()	R0F6.3.1.5.1
SWEDesigner::Client::Services::Classe::getSottoclasse()	
SWEDesigner::Client::Services::Classe::removeAttr()	R0F6.3.1.11
SWEDesigner::Client::Services::Classe::removevMetodo()	R0F6.3.1.10
SWEDesigner::Client::Services::Classe::retriveMethod()	R0F6.3.1.5.4
SWEDesigner::Client::Services::Classe::toJSON()	R0F6.1.1.1
SWEDesigner::Client::Services::ClasseAstratta::addAbstractMethods()	R0F6.3.1.5.5
SWEDesigner::Client::Services::ClasseAstratta::toJSON()	R0F6.1.1.1
SWEDesigner::Client::Services::ClassMenuService::classSelection()	
SWEDesigner::Client::Services::Global::addClasse()	
SWEDesigner::Client::Services::Global::changeTitolo()	
SWEDesigner::Client::Services::Global::getClassi()	
SWEDesigner::Client::Services::Global::getDiagramma()	
SWEDesigner::Client::Services::Global::getTitolo()	
SWEDesigner::Client::Services::Global::setDiagramma()	
SWEDesigner::Client::Services::Global::toJSON()	
SWEDesigner::Client::Services::Interface::addAbstractMethods()	R0F6.3.1.5.2
SWEDesigner::Client::Services::MainEditorService::addAttributo()	R0F6.3.1.5.3
SWEDesigner::Client::Services::MainEditorService::addClass()	R0F6.3.1.5
SWEDesigner::Client::Services::MainEditorService::addMetodo()	R0F6.3.1.5.5
SWEDesigner::Client::Services::MainEditorService::enterActivityMode()	R0F6.3
SWEDesigner::Client::Services::MainEditorService::enterClassMode()	
SWEDesigner::Client::Services::MainEditorService::getActivityModeStatus()	R0F6.3
SWEDesigner::Client::Services::MainEditorService::getClassList	
SWEDesigner::Client::Services::MainEditorService::getSelectedClasse()	R0F6.3.1.5
SWEDesigner::Client::Services::MainEditorService::removeAtributo()	R0F6.3.1.11
SWEDesigner::Client::Services::MainEditorService::removeMetodo()	R0F6.3.1.10
SWEDesigner::Client::Services::MainEditorService::selectClasse()	R0F6.3.1.5
SWEDesigner::Client::Services::MainEditorService::setActivityMode()	R0F6.3
SWEDesigner::Client::Services::MainEditorService::setClassMode()	R0F6.2
SWEDesigner::Client::Services::MainEditorService::setEditorComp()	
SWEDesigner::Client::Services::MainEditorService::storeGraph()	
SWEDesigner::Client::Services::MenuService::zoomIn()	R0F6.3.2
SWEDesigner::Client::Services::MenuService::zoomOut()	R0F6.3.3
SWEDesigner::Client::Services::Metodo::addArgomento()	R0F6.3.1.5.4.4
SWEDesigner::Client::Services::Metodo::addDiagram()	
SWEDesigner::Client::Services::Metodo::changeAccesso()	R0F6.3.1.5.4.5

Componenti	Classi
SWEDesigner::Client::Services::Metodo::changeListaArg()	R0F6.3.1.5.4.4
SWEDesigner::Client::Services::Metodo::changeNome()	R0F6.3.1.5.4.2
SWEDesigner::Client::Services::Metodo::constructor()	
SWEDesigner::Client::Services::Metodo::getAccesso()	R0F6.3.1.5.4.5
SWEDesigner::Client::Services::Metodo::getDiagram()	
SWEDesigner::Client::Services::Metodo::getListaArgomenti()	R0F6.3.1.5.4.4
SWEDesigner::Client::Services::Metodo::getNome()	R0F6.3.1.5.4.2
SWEDesigner::Client::Services::Metodo::getTipoRitorno()	R0F6.3.1.5.4.3
SWEDesigner::Client::Services::Metodo::tipoDiRitorno()	R0F6.3.1.5.4.3
SWEDesigner::Client::Services::Param::changeNome()	R0F6.3.1.5.2
SWEDesigner::Client::Services::Param::changeTipo()	
SWEDesigner::Client::Services::Param::getNome()	
SWEDesigner::Client::Services::Param::getTipo()	
SWEDesigner::Server::Controller::Middleware::midLoader::load()	
SWEDesigner::Server::Controller::Middleware::Parse::getTamplate()	R0F6.1.1.4
SWEDesigner::Server::Controller::Middleware::Parse::load()	R0F6.1.1.4
SWEDesigner::Server::Controller::Middleware::Parse::parse()	R0F6.1.1.4
SWEDesigner::Server::Controller::Services::encryptService::decrypt()	R0F5.1.2
SWEDesigner::Server::Controller::Services::encryptService::encrypt()	R0F6.1.1.3
SWEDesigner::Server::Controller::Services::encryptService::getIv()	R0F6.1.1.3
	R0F5.1.2
SWEDesigner::Server::Controller::Services::encryptService::getKey()	R0F6.1.1.3
	R0F5.1.2
SWEDesigner::Server::Controller::Services::parseService::parsing()	R0F6.1.1.4
SWEDesigner::Server::Controller::Middleware::Encrypt::decrypt()	R0F5.1.2
SWEDesigner::Server::Controller::Middleware::Encrypt::encrypt()	R0F6.1.1.3
SWEDesigner::Server::Controller::Middleware::Encrypt::getIv()	R0F6.1.1.3
	R0F5.1.2
SWEDesigner::Server::Controller::Middleware::Encrypt::getKey()	R0F6.1.1.3
	R0F5.1.2
SWEDesigner::Server::Model::mongooseConnection::conn()	R0F1.1
	R0F1.2
	R0F1.3
SWEDesigner::Server::serverLoader::load()	

Tabella 2: Tracciamento Classi - Requisiti

6.2 Tracciamento Requisiti-Classi

Requisiti	Classi
R0F1.1	SWEDesigner::Server::Model::mongooseConnection::conn()
R0F1.1	SWEDesigner::Server::Model::mongooseRequest::insUsr()
R0F1.2	SWEDesigner::Server::Model::mongooseConnection::conn()
R0F1.2	SWEDesigner::Server::Model::mongooseRequest::insUsr()
R0F1.3	SWEDesigner::Server::Model::mongooseConnection::conn()
R0F1.3	SWEDesigner::Server::Model::mongooseRequest::insUsr()
R0F2	SWEDesigner::Server::Model::mongooseRequest::login()
R0F5	SWEDesigner::Server::Model::mongooseRequest::loadAllProj()
R0F5.1	SWEDesigner::Server::Model::mongooseRequest::insproj()
R0F5.1	SWEDesigner::Server::Model::mongooseRequest::loadProj()
R0F5.1.2	SWEDesigner::Server::Model::mongooseRequest::inscryptparam()
R0F5.1.2	SWEDesigner::Server::Model::mongooseRequest::loadKeyCryp()
R0F5.1.2	SWEDesigner::Server::Controller::Middleware::Encrypt::decrypt()
R0F5.1.2	SWEDesigner::Server::Controller::Middleware::Encrypt::getKey()
R0F5.1.2	SWEDesigner::Server::Controller::Middleware::Encrypt::getIv()
R0F5.1.2	SWEDesigner::Server::Controller::Services::encryptService::decrypt()
R0F5.1.2	SWEDesigner::Server::Controller::Services::encryptService::getKey()
R0F5.1.2	SWEDesigner::Server::Controller::Services::encryptService::getIv()
R0F6.1.1.1	SWEDesigner::Client::Services::ClasseAstratta::toJSON()
R0F6.1.1.1	SWEDesigner::Client::Services::Classe::toJSON()
R0F6.1.1.3	SWEDesigner::Server::Model::mongooseRequest::inscryptparam()
R0F6.1.1.3	SWEDesigner::Server::Model::mongooseRequest::loadKeyCryp()
R0F6.1.1.3	SWEDesigner::Server::Controller::Middleware::Encrypt::encrypt()
R0F6.1.1.3	SWEDesigner::Server::Controller::Middleware::Encrypt::getKey()
R0F6.1.1.3	SWEDesigner::Server::Controller::Middleware::Encrypt::getIv()
R0F6.1.1.3	SWEDesigner::Server::Controller::Services::encryptService::encrypt()
R0F6.1.1.3	SWEDesigner::Server::Controller::Services::encryptService::getKey()
R0F6.1.1.3	SWEDesigner::Server::Controller::Services::encryptService::getIv()
R0F6.1.1.4	SWEDesigner::Server::Model::mongooseRequest::dropSchema()
R0F6.1.1.4	SWEDesigner::Server::Controller::Middleware::Parse::load()
R0F6.1.1.4	SWEDesigner::Server::Controller::Middleware::Parse::parse()
R0F6.1.1.4	SWEDesigner::Server::Controller::Middleware::Parse::getTemplate()
R0F6.1.1.4	SWEDesigner::Server::Controller::Services::parseService::parsing()
R0F6.2	SWEDesigner::Client::Services::MainEditorService::setClassMode()
R0F6.2.1.1	SWEDesigner::Client::Components::Editor::ToolbarComponent::addClasse()
R0F6.2.1.1	SWEDesigner::Client::Components::Editor::ToolbarComponent::addAstratta()
R0F6.2.1.1	SWEDesigner::Client::Components::Editor::ToolbarComponent::addInterfaccia()
R0F6.2.1.3	SWEDesigner::Client::Components::Editor::EditorComponent::addConnettore()
R0F6.2.1.3	SWEDesigner::Client::Components::Editor::ToolbarComponent::addGeneralizzazione()
R0F6.2.1.3	SWEDesigner::Client::Components::Editor::ToolbarComponent::addImplementazione()

Requisiti	Classi
R0F6.2.1.3	SWEDesigner::Client::Components::Editor::ToolbarComponent::addAssociazione()
R0F6.2.1.3	SWEDesigner::Client::Components::Editor::ToolbarComponent::addConnettore
R0F6.2.1.3.1	SWEDesigner::Client::Components::Editor::EditorComponent::selectElementsToConnect()
R0F6.2.1.3.1	SWEDesigner::Client::Components::Editor::EditorComponent::elementSelection()
R0F6.2.1.3.2	SWEDesigner::Client::Components::Editor::EditorComponent::selectElementsToConnect()
R0F6.2.1.3.2	SWEDesigner::Client::Components::Editor::EditorComponent::elementSelection()
R0F6.3	SWEDesigner::Client::Services::MainEditorService::setActivityMode()
R0F6.3	SWEDesigner::Client::Services::MainEditorService::getActivityModeStatus()
R0F6.3	SWEDesigner::Client::Services::MainEditorService::enterActivityMode()
R0F6.3.1.10	SWEDesigner::Client::Services::MainEditorService::removeMetodo()
R0F6.3.1.10	SWEDesigner::Client::Services::Classe::removevMetodo()
R0F6.3.1.10	SWEDesigner::Client::Components::Editor::ClassMenuComponent::removeMetodo()
R0F6.3.1.11	SWEDesigner::Client::Services::MainEditorService::removeAttributo()
R0F6.3.1.11	SWEDesigner::Client::Services::Classe::removeAttr()
R0F6.3.1.11	SWEDesigner::Client::Components::Editor::ClassMenuComponent::removeAttributo
R0F6.3.1.5	SWEDesigner::Client::Services::MainEditorService::getSelectedClasse()
R0F6.3.1.5	SWEDesigner::Client::Services::MainEditorService::addClass()
R0F6.3.1.5	SWEDesigner::Client::Services::MainEditorService::selectClasse()
R0F6.3.1.5.1	SWEDesigner::Client::Services::Classe::changeNome()
R0F6.3.1.5.1	SWEDesigner::Client::Services::Classe::getNome()
R0F6.3.1.5.1	SWEDesigner::Client::Components::Editor::ClassMenuComponent::changeNome()
R0F6.3.1.5.2	SWEDesigner::Client::Services::Classe::addAttributo()
R0F6.3.1.5.2	SWEDesigner::Client::Services::Classe::changeAttr()
R0F6.3.1.5.2	SWEDesigner::Client::Services::Classe::getAttributi()
R0F6.3.1.5.2	SWEDesigner::Client::Components::Editor::ClassMenuComponent::changeAttributo()
R0F6.3.1.5.2	SWEDesigner::Client::Services::Param::changeNome()
R0F6.3.1.5.2	SWEDesigner::Client::Services::Attributo::changeAccesso()
R0F6.3.1.5.2	SWEDesigner::Client::Services::Interface::addAbstractMethods()
R0F6.3.1.5.3	SWEDesigner::Client::Services::MainEditorService::addAttributo()
R0F6.3.1.5.3	SWEDesigner::Client::Components::Editor::ClassMenuComponent::addAttributo()
R0F6.3.1.5.4	SWEDesigner::Client::Services::Classe::addMetodo()
R0F6.3.1.5.4	SWEDesigner::Client::Services::Classe::getMetodi()
R0F6.3.1.5.4	SWEDesigner::Client::Services::Classe::retriveMethod()
R0F6.3.1.5.4	SWEDesigner::Client::Components::Editor::ClassMenuComponent::modifyMetodo()
R0F6.3.1.5.4.2	SWEDesigner::Client::Services::Metodo::changeNome()
R0F6.3.1.5.4.2	SWEDesigner::Client::Services::Metodo::getNome()
R0F6.3.1.5.4.3	SWEDesigner::Client::Services::Metodo::tipoDiRitorno()
R0F6.3.1.5.4.3	SWEDesigner::Client::Services::Metodo::getTipoRitorno()
R0F6.3.1.5.4.4	SWEDesigner::Client::Services::Metodo::changeListaArg()
R0F6.3.1.5.4.4	SWEDesigner::Client::Services::Metodo::addArgomento()
R0F6.3.1.5.4.4	SWEDesigner::Client::Services::Metodo::getListaArgomenti()
R0F6.3.1.5.4.5	SWEDesigner::Client::Services::Metodo::changeAccesso()

Requisiti	Classi
R0F6.3.1.5.4.5	SWEDesigner::Client::Services::Metodo::getAccesso()
R0F6.3.1.5.5	SWEDesigner::Client::Services::MainEditorService::addMetodo()
R0F6.3.1.5.5	SWEDesigner::Client::Services::ClasseAstratta::addAbstractMethods()
R0F6.3.1.5.6	SWEDesigner::Client::Components::Editor::ToolbarComponent::addAstratta()
R0F6.3.1.5.7	SWEDesigner::Client::Components::Editor::ToolbarComponent::addInterfaccia()
R0F6.3.1.8	SWEDesigner::Client::Components::Editor::ToolbarComponent::addCommento()
R0F6.3.2	SWEDesigner::Client::Components::Editor::EditorComponent::ZoomIn()
R0F6.3.2	SWEDesigner::Client::Components::Menu::ModificaComponent::doZoomIN()
R0F6.3.2	SWEDesigner::Client::Services::MenuService::zoomIn()
R0F6.3.3	SWEDesigner::Client::Components::Editor::EditorComponent::ZoomOut()
R0F6.3.3	SWEDesigner::Client::Components::Menu::ModificaComponent::DoZoomOut()
R0F6.3.3	SWEDesigner::Client::Services::MenuService::zoomOut()
R1F13	SWEDesigner::Server::Model::mongooseRequest::forgotMail()
R1F6.1.2.4	SWEDesigner::Client::Components::Editor::EditorComponent::cloneElement()
R1F6.1.2.5	SWEDesigner::Client::Components::Editor::EditorComponent::cloneElement()

Tabella 3: Tracciamento Requisiti - Classe