Definizione di Prodotto

Gruppo SWEet BIT — Progetto SWEDesigner

Informazioni sul documento	
Versione	1.0.0
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 **8**
1.1 Scopo del documento . . . 8
1.2 Scopo del prodotto . . . 8
1.3 Glossario . . . 8
1.4 Riferimenti . . . 8
1.4.1 Normativi . . . 8
1.4.2 Informativi . . . 9
1.5 Descrizione dell'architettura . . . 10

2 Standard di progetto **11**
2.1 Standard di progettazione architetturale . . . 11
2.2 Standard di documentazione del codice . . . 11
2.3 Standard di denominazione di entità e relazioni . . . 11
2.4 Standard di programmazione . . . 11
2.5 Strumenti di lavoro . . . 11

3 Specifica Front-End **12**
3.1 SWEDesigner::Client . . . 12
3.1.1 Informazioni generali . . . 12
3.1.2 Classi . . . 12
3.1.2.1 SWEDesigner::AuthenticationGuard . . . 12
3.1.2.2 SWEDesigner::Global . . . 13
3.2 SWEDesigner::Client::Components . . . 16
3.2.1 Informazioni generali . . . 16
3.2.2 Classi . . . 16
3.2.2.1 SWEDesigner::Client::Components::RegistrationComponent 16
3.2.2.2 SWEDesigner::Client::Components::LoginComponent . . 17
3.2.2.3 SWEDesigner::Client::Components::Forgot-pswComponent 17
3.3 SWEDesigner::Client::Components::Editor-container . . . 18
3.3.1 Informazioni generali . . . 18
3.3.2 Classi . . . 18
3.3.2.1 SWEDesigner::Client::Components::Editor-container::Editor-containerComponent . . . 18
3.3.2.2 SWEDesigner::Client::Components::Editor-container::Activity-frameComponent . . . 19
3.4 SWEDesigner::Client::Components::Editor-container::Menu . . . 19
3.4.1 Informazioni generali . . . 19
3.4.2 Classi . . . 19
3.4.2.1 SWEDesigner::Client::Components::Editor-container::Menu::MenuComponent 19
3.4.2.2 SWEDesigner::Client::Components::Editor-container::Menu::FileComponent 20
3.4.2.3 SWEDesigner::Client::Components::Editor-container::Menu::ModificaComponent

3.4.2.4 SWEDesigner::Client::Components::Editor-container::Menu::ProfiloComponent 2
3.4.2.5 SWEDesigner::Client::Components::Editor-container::Menu::ProgettoComponent
3.5 SWEDesigner::Client::Components::Editor-container::Editor 22
3.5.1 Informazioni generali 22
3.5.2 Classi 22
3.5.2.1 SWEDesigner::Client::Components::Editor-container::Editor::EditorComponent
3.5.2.2 SWEDesigner::Client::Components::Editor-container::Editor::Activity-menuComponent 24
3.5.2.3 SWEDesigner::Client::Components::Editor-container::Editor::ToolbarComponent
3.5.2.4 SWEDesigner::Client::Components::Editor-container::Editor::ActivityService 28
3.5.2.5 SWEDesigner::Client::Components::Editor-container::Editor::Class-menuService 33
3.5.2.6 SWEDesigner::Client::Components::Editor-container::Editor::All-shape 37
3.5.2.7 SWEDesigner::Client::Components::Editor-container::Editor::Attributo 38
3.5.2.8 SWEDesigner::Client::Components::Editor-container::Editor::Class-errors 39
3.5.2.9 SWEDesigner::Client::Components::Editor-container::Editor::Classe-astratta 39
3.5.2.10 SWEDesigner::Client::Components::Editor-container::Editor::Classe 40
3.5.2.11 SWEDesigner::Client::Components::Editor-container::Editor::Elemento-metodo 43
3.5.2.12 SWEDesigner::Client::Components::Editor-container::Editor::End 43
3.5.2.13 SWEDesigner::Client::Components::Editor-container::Editor::If-node 44
3.5.2.14 SWEDesigner::Client::Components::Editor-container::Editor::Interface 45
3.5.2.15 SWEDesigner::Client::Components::Editor-container::Editor::Merge-node 46
3.5.2.16 SWEDesigner::Client::Components::Editor-container::Editor::Metodo 47
3.5.2.17 SWEDesigner::Client::Components::Editor-container::Editor::Operation 49
3.5.2.18 SWEDesigner::Client::Components::Editor-container::Editor::Operazione 50
3.5.2.19 SWEDesigner::Client::Components::Editor-container::Editor::Param 50
3.5.2.20 SWEDesigner::Client::Components::Editor-container::Editor::Shape 51
3.5.2.21 SWEDesigner::Client::Components::Editor-container::Editor::Start 53
3.5.2.22 SWEDesigner::Client::Components::Editor-container::Editor::Variabile 54
3.5.2.23 SWEDesigner::Client::Components::Editor-container::Editor::While-node 55
3.6 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu 56
3.6.1 Informazioni generali 56
3.6.2 Classi 56
3.6.2.1 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Edit-class-menuComponent 56

3.6.2.2 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Change-class-nameComponent 56
3.6.2.3 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-add-attributeComponent 57
3.6.2.4 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-add-main-methodComponent 58
3.6.2.5 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-add-methodComponent 58
3.6.2.6 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-list-attributeComponent 59
3.6.2.7 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Class-list-methodComponent 60
3.6.2.8 SWEDesigner::Client::Components::Editor-container::Editor::Edit-class-menu::Display-class-nameComponent 61
3.7 SWEDesigner::Client::Services 61
3.7.1 Informazioni generali 61
3.7.2 Classi 62
3.7.2.1 SWEDesigner::Client::Services::AccountService 62
3.7.2.2 SWEDesigner::Client::Services::Main-editorService 66
3.7.2.3 SWEDesigner::Client::Services::MenuService 71

4 Specifica Back-End **74**
4.1 SWEDesigner::Server 74
4.1.1 Informazioni generali 74
4.1.2 Classi 74
4.1.2.1 SWEDesigner::Server::serverLoader 74
4.2 SWEDesigner::Server::Model 75
4.2.1 Informazioni generali 75
4.2.2 Classi 75
4.2.2.1 SWEDesigner::Server::Model::mongooseConnection . . . 75
4.2.2.2 SWEDesiger::Server::Model::mongooseRequest 76
4.3 SWEDesigner::Server::Controller::Middleware 82
4.3.1 Informazioni generali 82
4.3.2 Classi 82
4.3.2.1 SWEDesigner::Server::Controller::Middleware::midLoader 82
4.3.2.2 SWEDesigner::Server::Controller::Middleware::Parse . . . 82
4.3.2.3 SWEDesigner::Server::Controller:Middleware::Encrypt . . 83
4.4 SWEDesigner::Server::Controller::Services 85
4.4.1 Informazioni generali 85
4.4.2 Classi 85
4.4.2.1 SWEDesigner::Server::Controller::Services::parseService . 85
4.4.2.2 SWEDesigner::Server::Controller::Services::encryptService 86

5 Diagrammi di sequenza **88**

5.1 Generazione Codice 88
5.2 Caricamento moduli del Server 88
5.3 Encrypt/Decrypt 89

6 Tracciamento **90**
6.1 Tracciamento Classi-Requisiti 90
6.2 Tracciamento Requisiti-Classi 94

Elenco delle figure

1 Diagramma della classe SWEDesigner::Server::serverLoader 74
2 Diagramma della classe SWEDesigner::Server::Model::mongooseConnection 76
3 Diagramma della classe SWEDesigner::Server::Model::mongooseRequest . 76
4 Diagramma della classe SWEDesigner::Server::Controller::Middleware::midLoader 82
5 Diagramma della classe SWEDesigner::Server::Controller::Middleware::Parse 82
6 Diagramma della classe SWEDesigner::Server::Controller::Middleware::Encrypt 83
7 Diagramma della classe SWEDesigner::Server::Controller::Services::parseService 85
8 Diagramma della classe SWEDesigner::Server::Controller::Services::encryptService 86
9 Sequence diagram generazione codice java 88
10 Sequence diagram caricamento moduli server 89
11 Sequence diagram per operazioni di encrypt e decrypt 90

Elenco delle tabelle

2 Tracciamento Classi - Requisiti 93
3 Tracciamento Requisiti - Classe 96

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 realizzazone 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 v*3.0.0. 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 v3.0.0.*

- **Analisi dei Requisiti:** *Analisi dei Requisiti v3.0.0.*

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 architetturale

Gli standard di progettazione sono definiti *Specifica Tecnica v* 2.0.0 .

2.2 Standard di documentazione del codice

Gli standard per la scrittura della documentazione del codice sono definiti nelle Norme di Progetto 3.0.0.

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 v*3.0.0.

2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle *Norme di Progetto v*3.0.0.

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 v*3.0.0.

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

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

3.1.2.2 SWEDesigner::Global

- **Descrizione:**
- **Utilizzo:**
- **Attributi:**
  – *-nome_progetto: string*
  – *-diagramma: string*
  – *-classi: Classe[]*
  – *-main: boolean*
- **Metodi:**
  – *+addClasse(nome: string)*

  Parametri:

  * *nome: string*

  – *+changeTitolo(titolo: string)*

  Parametri:

  * *titolo: string*

- *+setDiagramma(diagramma: string)*

  Parametri:

  * *diagramma: string*

- *+setName(name: string)*

  Parametri:

  * *name: string*

- *getDiagramma()*

- *+getTitolo()*

- *+public getClassi()*

- *+getClasse(name: string)*

  Parametri:

  * *name: string*

- *+setMain()*

- *+getMainStat()*

- *+removeClass(name: string)*

  Parametri:

  * *name: string*

- *+import(proj: any)*

  Parametri:

  * *proj: any*

- *+generateClassArray(classArray: any)*

  Parametri:

  * *classArray: any*

- *+generateMethods(classe: Classe, methods: any)*

  Parametri:

  * *classe: Classe*

  * *methods: any*

- *+generateParams(params: any)*

  Parametri:

  * *params: any*

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

  Parametri:

  * *classe: Classe*

  * *attributi: any*

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

  Parametri:

  * *usr: String*

  * *projName: string*

- *+getInfoClasse(x: any)*

  Parametri:

  * *x: any*

- *+toMU()*

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

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 iserire 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 istanziazione di RegistrationComponent
    Parametri:

    * *-router: Router* Necessario per l'importaziine 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

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)*
    Crea un istanziazione di RegistrationComponent
    Parametri:
    * *-router: Router* Necessario per l'importaziine del Router
    * *-accountService: AccountService* Necessario per l'importazione di AccountService
  - *+loginUser(e: any)*
    Effettua l'autenticazione dell'utente
    Parametri:
    * *+e: any* Contiente 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, 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* Contiente 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)*

    Parametri:

    * *menuService: MenuService*

    * *accountService: AccountService*

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

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)*

    Parametri:

· *accountService: AccountService*

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)*

Parametri:

· *menuService: MenuService*

· *mainEditorService: MainEditorService*

· *accountService: 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)*
  Importa un progetto esistente
  Parametri:
  · *event: any*
  Progetto da importare

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

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

– **Utilizzo:**

– **Metodi:**

* *doZoomIn()*
  Esegue lo zoon in avanti
* *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

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)*

    Parametri:
    - *accountService: 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::ProgettoCompone

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

3.5.1 Informazioni generali

- **Descrizione:**
  Questo package contiene tutti i components riguardanti l'editor e la gestione dell'utente.
- **Padre:** SWEDesigner::Client::Components::Editor-container
- **Package contenuti:**
  - Edit-class-menu

3.5.2 Classi

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

- **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:**

*

Parametri:

.

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

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-decisions: string[]*
* *-dec: string*
* *-params: string[]*
* *-operators: string*
* *-types: string*
* *-modPro: boolean*
* *-mod: string*
* *-nomeVar: string*
* *-tipoVar: string*

* *-valVar: any*
* *-va: string*
* *-operando: string*
* *-nomeInd: string*
* *-valInd: number*
* *-maxInd: number*
* *-op: string*

– **Metodi:**

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

  Parametri:

  · *mainEditorService: MainEditorService*

  · *activityService: ActivityService*

* *+enterClassMode()*
* *+generaCodice()*
* *+changeName(name: string)*

  Parametri:

  · *name: string*

* *+isMain()*

* *+generaOp(corpo: string)*

  Parametri:

  - *corpo: string*

* *+generaIf()*

* *+generaFor()*

* *+generaWhile()*

* *+setNomeVar(name: string)*

  Parametri:

  - *name: string*

* *+isNumeric()*

* *+confezionaVar()*

* *+declareVar()*

* *+deleteVar(id: string)*

  Parametri:

  - *id: string*

* *+toggleModPro()*

* *+modificaIf()*

* *+modificaFor()*

* *+modificaWhile()*

* *+getParams()*

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

- **Descrizione:**

- **Utilizzo:**

- **Attributi:**

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

    Parametri:

    · *mainEditorService: MainEditorService*

    · *activityService: 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()*
* *+addActivityForShape()*
* *+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()*

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

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-shapeList:AllShape*
* *-selectedShape: Shape*

* *-selectedMethod: Metodo*
* *-selectedElement: any*
* *-stardID: string*
* *-endID: string*
* *-varibles: Map<string, string>*
* *-vars: string[]*

– **Metodi:**

* *-constructor(private mainEditorService: MainEditorService*

  Parametri:

  · *mainEditorService: MainEditorService*

* *-getSelectedShapeId()*
* *-getShapeList()*
* *-addIfNode(graphElement: any)*

  Parametri:

  · *graphElement: any*

* *-addOperation(graphElement: any)*

  Parametri:

  · *graphElement: any*

* *-addMergeNode(graphElement: any)*

  Parametri:

- *graphElement: any*

* *-addLoopNode(id: string)*

  Parametri:

  - *id: string*

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

  Parametri:

  - *id: string*

  - *name: string*

* *-setSelectedMethod(metodo: Metodo)*

  Parametri:

  - *metodo: Metodo*

* *+setSelectedElement(element: any)*

  Parametri:

  - *element: any*

* *+selectShape(id: string)*

  Parametri:

  - *id: string*

* *+start()*

* *+end()*

* *+deselectElement()*

* *+addBody(body: string)*

  Parametri:

  - *body: string*

* *+getSelectedMethod()*

* *+getSelectedElement()*

* *+getNameMethod()*

* *+getVarVis()*

* *+getShapeType()*

* *+changeName(name:string)*

  Parametri:

  - *name: string*

* *+connect(elementCon)*

  Parametri:

  - *elementCon: any*

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

  Parametri:

  - *ids: string[]*

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

  Parametri:

  - *text: string*

- *modText: boolean*

* *+hasBody()*

* *+getBody()*

* *+generaCodice()*

* *+isDecision()*

* *+isVarDeclaration()*

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

  Parametri:

  - *dec: string*

  - *code: string*

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

  Parametri:

  - *opType: string*

  - *id: string*

* *+modVariable(code: string)*

  Parametri:

  - *code: string*

* *+deleteVar(id: string)*

  Parametri:

  - *id: string*

* *+salvaMetodo()*

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

- **Descrizione:**

- **Utilizzo:**

- **Attributi:**

  * *-selectedClassSource: Subject<any>*
    Risorsa observable oggetto-classe
  * *-selectedClass*
    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
  * *-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 istanziazione 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'attribuo
  - *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

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

- **Descrizione:**

- **Utilizzo:**

- **Attributi:**
  * *-allShap: Shape[]*

  * *-statements: string[]*

  * *-merges: string*

  * *-code: string*

- **Metodi:**
  * *+addMerge(id: string)*

    Parametri:

    · *id: string*

  * *+addStatement(id: string)*

    Parametri:

    · *id: string*

  * *+getAllShape()*

  * *+getElementById(id: string)*

    Parametri:

    · *id: string*

* *+getElementByType(type: string)*

  Parametri:

  · *type: string*

* *+setCode(cd: string)*

  Parametri:

  · *cd: string*

* *+removeShape(id: string)*

  Parametri:

  · *id: string*

* *+getMerges()*

* *+getStatements()*

* *+toCode()*

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

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-visibility: string*

* *-staticAtt: boolean*

* *-finalAtt: boolean*

– **Metodi:**

* *+changeAcc(acc: string)*

  Parametri:

  · *acc: string*

* *+isStatic()*
  Ritorna true se l'attributo è statico
* *+isFinal()*
  Ritorna true se l'attributo è final
* *+getAccesso()*
  Ritorna la visibilità dell'attributo
* *+toMU()*

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

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
* *+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-metod

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

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(id : string)*

  Parametri:

  · *id : string*

* *+getType())*

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

  Parametri:

  · *sh: AllShape*

· *code: string*

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

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-succElse: string*

– **Metodi:**

* *-constructor(id: string)*

  Parametri:

  · *id: string*

* *+getSuccElse()*

* *+setSuccElse(succElse: string)*

  · *succElse: string*

* *+getType()*

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

  Parametri:

  · *sh: AllShape*

  · *code: string*

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*
      Netodo 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:**

* *-constructor(id: string)*

  Parametri:
  - *id: string*

* *+getType()*

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

  Parametri:
  - *sh: AllShape*
  - *code: string*

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
  * *-listaArgomenti: Param[]*
    Lista di argomenti del metodo
  * *-diagramma: JSON*
  * *-shapeList: AllShape*
  * *-mapVarVisibili: Map<string, string>*
  * *-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(listArg: 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
* *+getListaArgomenti()*
  Ritorna la lista degli argomenti del metodo
* *+getShapeList()*
* *+getMapVars()*
* *+isConstructor()*
* *+isStatic()*
* *+staticString()*
* *+setVars(vars: Map<string, string>)*

  Parametri:

  · *vars: Map<string, string>*
* *+paramToString()*

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

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-operationType : string*

- **Metodi:**
  * *-constructor(id: string)*

    Parametri:

    · *id: string*

  * *+setOperationType(t: string)*

    Parametri:

    · *t: string*

  * *+getOperationType()*

  * *+getType()*

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

    Parametri:

    · *sh: AllShape*

    · *code: string*

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

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()*
* *+changeTipo(tipo: string)*
  Modifica il tipo del parametro
  Parametri:
  · *tipo: string*
  Tipo del parametro
* *+changeTipo(tipo: string)*
  Modifica il nome del parametro
  Parametri:
  · *tipo: string*
  Nome del parametro

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

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-id: string*

    * *-succ: string*
    * *-body: string*
    * *-printed: boolean*
    * *-ifPassed: String[]*

- **Metodi:**
    * *-constructor(id: string)*

      Parametri:
        - *id: string*

    * *+setId(id: string)*

      Parametri:
        - *id: string*

    * *+setBody(body: string)*

      Parametri:
        - *body: string*

    * *+setSucc(succ: string)*

      Parametri:
        - *succ: string*

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

      Parametri:
        - *pas: string[]*

* *+setPrinted(printed: boolean)*

  Parametri:

  · *printed: boolean*

* *addBody(b: string)*

  Parametri:

  · *b: string*

* *+getId()*

* *+getBody()*

* *+getSucc()*

* *+getIfPassed()*

* *+getPrinted()*

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

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(id : string)*

  Parametri:

  · *id : string*

* *+getType()*

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

  Parametri:

  - *sh: AllShape*
  - *code: string*

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

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-type: string*
* *-name: string*
* *-value: string*

– **Metodi:**

* *-constructor(type: string, name: string, value: string)*

  Parametri:

  - *type: string*
  - *name: string*
  - *value: string*

* *+getType()*
* *+getName()*

* *+getValue()*

* *+setType(type: string)*

  Parametri:

  · *type: string*

* *+setName(name: string)*

  Parametri:

  · *name: string*

* *+setValue(value: string)*

  Parametri:

  · *value: string*

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

– **Descrizione:**

– **Utilizzo:**

– **Metodi:**

* *-constructor(id: string)*

  Parametri:

  · *id: string*

* *+getType()*

* *+isFor()*

* *+setFor(s: boolean)*

  Parametri:

- *s: boolean*

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

  Parametri:

  - *sh: AllShape*

  - *code: string*

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::

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

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

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 istanziazione di ClassMenuService
  - *addMain()*
    Aggiunge il metodo main alla classe selezionata

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 istanziazione 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

3.6.2.6 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

* *+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

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

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

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
* *+makeCokie()*
  Costruisce dei cookie di sessione contenenti le informazioni dell'utente
* *+setParam(user: string, mail:string, pwd: string)*
  Setta le informazione 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:**

* *-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

* *-varCode: string[]*

– **Metodi:**

* *+setEditorComp(editCmp: EditorComponent)*
Serve per costruire un istanziazione dell'EditorComponent
Parametri:

· *editCmp: EditorComponent*
Istanzia 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*
    Conettore da aggiungere
* *+addSuperclass(subclassName: string, superclassName: string)*

  Parametri:
  - *subclassName: string*
  - *superclassName: string*
* *+getClass(name: string)*

  Parametri:

- *name: string*

* *+getProject()*

* *+retriveGraph()*

* *+importProject(importData: any)*

  Parametri:

  - *importData: any*

* *+loadProject(project: any)*

  Parametri:

  - *project: any*

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

  Parametri:

  - *name: string*

  - *classe: any*

* *+removeShapeActivity(element: any)*

  Parametri:

  - *element: any*

* *addShape(cell: any)*

  Parametri:

  - *cell: any*

* *+connetActivity(con: any)*

  Parametri:

- *con: any*

* *+setConnetionActivity(ids: string[])*

  Parametri:

  - *ids: string[]*

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

  Parametri:

  - *vars: string[]*

* *+checkrefresh()*

3.7.2.3 SWEDesigner::Client::Services::MenuService

– **Descrizione:**

– **Utilizzo:**

– **Attributi:**

* *-selectedGraphService: Subject<any>*

* *-importData: any*

– **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*
* *+updateData(proj: JSON, cb: Function)*
  Aggiorna i dati del progetto corrente nel database
  Parametri:
  - *proj: JSON*
    Progetto corrente
  - *cb: Function*
* *+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*

* *+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*
* *+import(event: any)*
  Importa un file esterno
  Parametri:
  - *event: any*
    File da importare
* *+getImportData()*
* *+donwload()*
  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 contiente 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*$_G$. 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

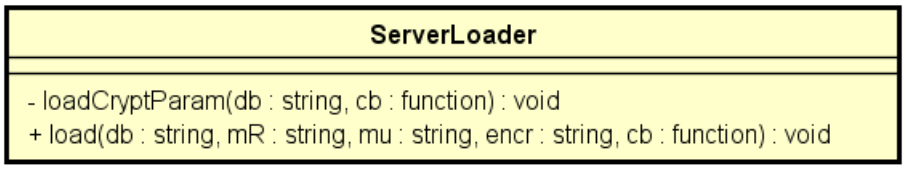

Figura 1: Diagramma della classe SWEDesigner::Server::serverLoader

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*taliano Callback che gestisce le rischieste asicnrone 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 rischieste asicnrone 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

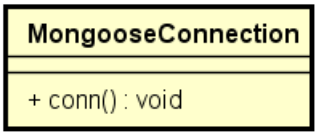

Figura 2: Diagramma della classe SWEDesigner::Server::Model::mongooseConnection

- **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.

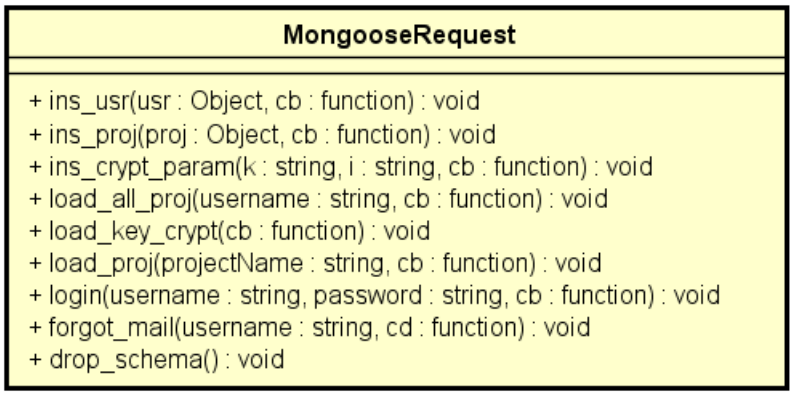

Figura 3: Diagramma della classe SWEDesigner::Server::Model::mongooseRequest

4.2.2.2 SWEDesiger::Server::Model::mongooseRequest

- **Descrizione:**
  Classe che si occupa di gestire tutte le query da e vero 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.

  · *cd: 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.
  - *cd: 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

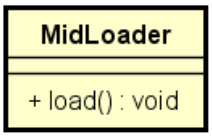

Figura 4: Diagramma della classe SWEDesigner::Server::Controller::Middleware::midLoader

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 cià che serve per il funzionamento del middleware.
- **Metodi:**

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

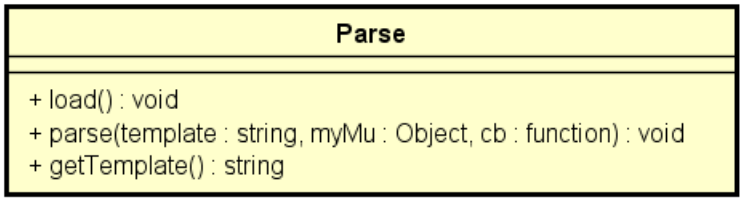

Figura 5: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Parse

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 Moustahce.
  - *+getTemplate() : string*
    La funzione ritorna il percorso in cui è contenuto il template, compilato o meno.

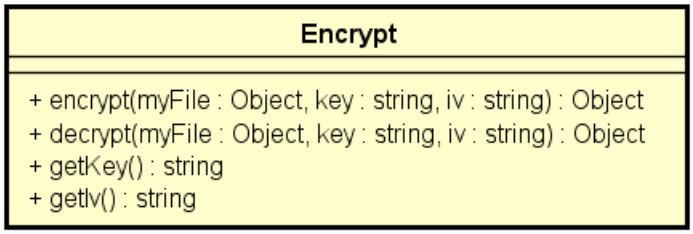

Figura 6: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Encrypt

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

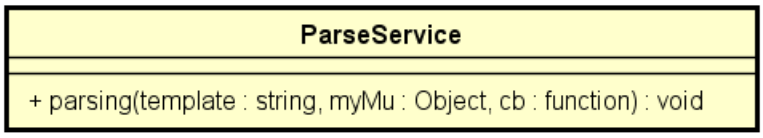

Figura 7: Diagramma della classe SWEDesigner::Server::Controller::Services::parseService

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.

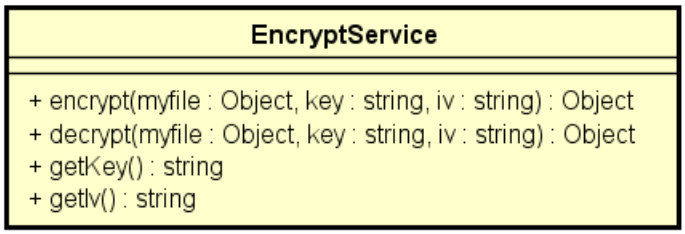

Figura 8: Diagramma della classe SWEDesigner::Server::Controller::Services::encryptService

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 gi algritmi 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 seguenza 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.

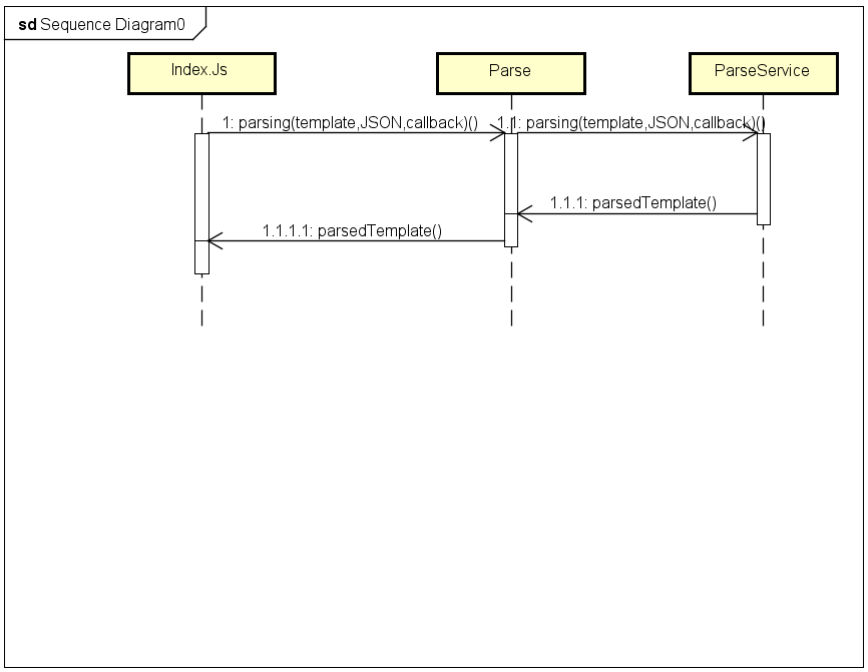

Figura 9: Sequence diagram generazione codice java

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 dispozione un template renderizzato, quindi utilizzare da Moustache, ogni volta che viene richiesta la generazione di codice.

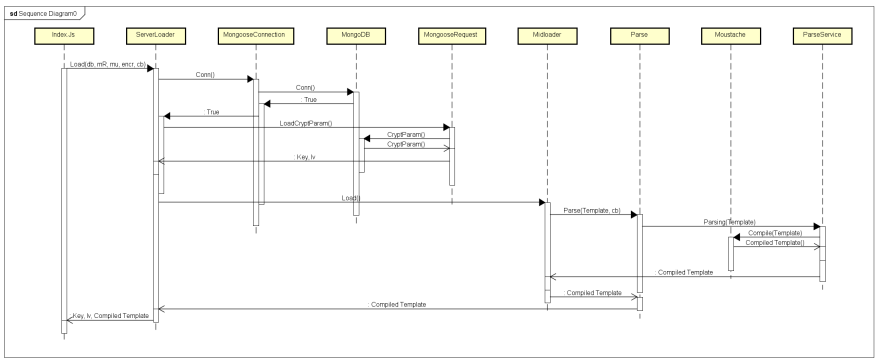

Figura 10: Sequence diagram caricamento moduli server

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 decriptati (a seconda della richiesta) al chiamante che gestirà il ritorno dell'informazione al Client.

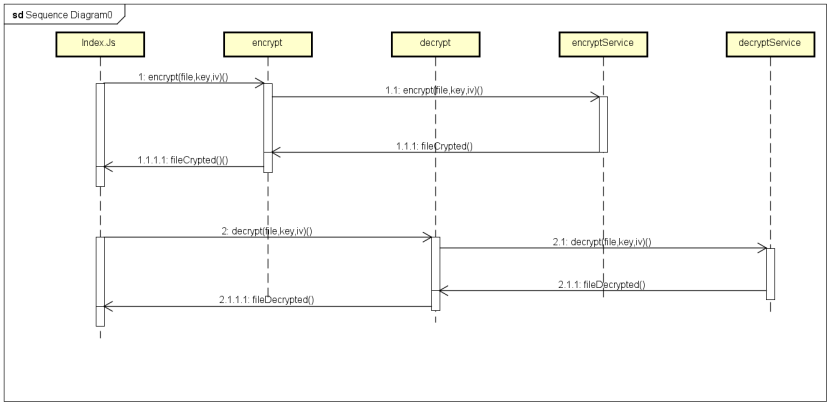

Figura 11: Sequence diagram per operazioni di encrypt e decrypt

6 Tracciamento

6.1 Tracciamento Classi-Requisiti

Componenti	Classi
SWEDesiger::Server::Model::mongooseRequest::dropSchema()	R0F6.1.1.4
SWEDesiger::Server::Model::mongooseRequest::forgotMail()	R1F13
SWEDesiger::Server::Model::mongooseRequest::inscryptparam()	R0F6.1.1.3 R0F5.1.2
SWEDesiger::Server::Model::mongooseRequest::insproj()	R0F5.1
SWEDesiger::Server::Model::mongooseRequest::insUsr()	R0F1.1 R0F1.2 R0F1.3
SWEDesiger::Server::Model::mongooseRequest::loadAllProj()	R0F5
SWEDesiger::Server::Model::mongooseRequest::loadKeyCrypy()	R0F6.1.1.3 R0F5.1.2
SWEDesiger::Server::Model::mongooseRequest::loadProj()	R0F5.1
SWEDesiger::Server::Model::mongooseRequest::login()	R0F2
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::addAttributo()	R0F6.3.1.5.3
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::addMetodo()	
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::aggiungiParam()	
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::changeAttributo()	R0F6.3.1.5.2
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::changeNome()	R0F6.3.1.5.1

Componenti	Classi
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::modifyMetodo()	R0F6.3.1.5.4
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::removeAttributo	R0F6.3.1.11
SWEDesigner::Client::Components::Editor::ClassMenuComponent ::removeMetodo()	R0F6.3.1.10
SWEDesigner::Client::Components::Editor::EditorComponent ::addConnettore()	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 ::constructior()	
SWEDesigner::Client::Components::Editor::EditorComponent ::elementSelection()	R0F6.2.1.3.1 R0F6.2.1.3.2
SWEDesigner::Client::Components::Editor::EditorComponent ::replaceDiagram()	
SWEDesigner::Client::Components::Editor::EditorComponent ::selectElementsToConnect()	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 ::addAssociazione()	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addAstratta()	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 ::addConnettore	R0F6.2.1.3
SWEDesigner::Client::Components::Editor::ToolbarComponent ::addGeneralizzazione()	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	SWEDesiger::Server::Model::mongooseRequest::insUsr()
R0F1.2	SWEDesigner::Server::Model::mongooseConnection::conn()
R0F1.2	SWEDesiger::Server::Model::mongooseRequest::insUsr()
R0F1.3	SWEDesigner::Server::Model::mongooseConnection::conn()
R0F1.3	SWEDesiger::Server::Model::mongooseRequest::insUsr()
R0F2	SWEDesiger::Server::Model::mongooseRequest::login()
R0F5	SWEDesiger::Server::Model::mongooseRequest::loadAllProj()
R0F5.1	SWEDesiger::Server::Model::mongooseRequest::insproj()
R0F5.1	SWEDesiger::Server::Model::mongooseRequest::loadProj()
R0F5.1.2	SWEDesiger::Server::Model::mongooseRequest::inscryptparam()
R0F5.1.2	SWEDesiger::Server::Model::mongooseRequest::loadKeyCrypy()
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	SWEDesiger::Server::Model::mongooseRequest::inscryptparam()
R0F6.1.1.3	SWEDesiger::Server::Model::mongooseRequest::loadKeyCrypy()
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	SWEDesiger::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::getTamplate()
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::removeAtributo()
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	SWEDesiger::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