



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

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

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

1.2 Scopo del prodotto

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

1.3 Glossario

Con lo scopo di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v3.0.0*. 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 architettuale

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 v3.0.0*.

2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle *Norme di Progetto v3.0.0*.

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 v3.0.0*.

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

3.1.2.2 SWEDesigner::Global

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 inserire le informazioni necessarie alla creazione di un nuovo account utente. Gestisce le operazioni e la logica applicativa per la registrazione.

- **Utilizzo:**

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

- **Metodi:**

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

- **Parametri:**

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

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

- **Parametri:**

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

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'istanziatura di RegistrationComponent

- **Parametri:**

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

- *+loginUser(e: any)*
Effettua l'autenticazione dell'utente

- **Parametri:**

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

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

- **Descrizione:**

È il componente che descrive la pagina per il recupero della password dell'applicazione, 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'istanziatura di Forgot-pswComponent

- **Parametri:**

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

- *+tryGetNewPassword(e: any)*
Invia all'utente la password per email

- **Parametri:**

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

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

3.3.1 Informazioni generali

- **Descrizione:**
Questo package contiene tutti i components riguardanti l'editor e la gestione dell'utente.
- **Padre:** SWEDesigner::Client::Components
- **Package contenuti:**
 - Menu
Il package contiene tutti i components riguardanti la gestione delle funzionalità offerte dal menu
 - Editor
Il package contiene tutti i components riguardanti l'editor e la gestione dell'utente

3.3.2 Classi

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

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

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

3.4.1 Informazioni generali

- **Descrizione:**
Questo package contiene tutti i components riguardanti la gestione delle funzionalità offerte dal menu.
- **Padre:** SWEDesigner::Client::Components::Editor-container

3.4.2 Classi

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

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

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

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

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

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

3.5.1 Informazioni generali

- **Descrizione:**
Questo package contiene tutti i 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::EditorComponent

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

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

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

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

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

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

- 3.5.2.8 SWEDesigner::Client::Components::Editor-container::Editor::Classe-astratta
- 3.5.2.9 SWEDesigner::Client::Components::Editor-container::Editor::Classe
- 3.5.2.10 SWEDesigner::Client::Components::Editor-container::Editor::Elemento-metodo
- 3.5.2.11 SWEDesigner::Client::Components::Editor-container::Editor::End
- 3.5.2.12 SWEDesigner::Client::Components::Editor-container::Editor::If-node
- 3.5.2.13 SWEDesigner::Client::Components::Editor-container::Editor::Interface
- 3.5.2.14 SWEDesigner::Client::Components::Editor-container::Editor::Merge-node
- 3.5.2.15 SWEDesigner::Client::Components::Editor-container::Editor::Metodo
- 3.5.2.16 SWEDesigner::Client::Components::Editor-container::Editor::Operation
- 3.5.2.17 SWEDesigner::Client::Components::Editor-container::Editor::Operazione
- 3.5.2.18 SWEDesigner::Client::Components::Editor-container::Editor::Param
- 3.5.2.19 SWEDesigner::Client::Components::Editor-container::Editor::Shape
- 3.5.2.20 SWEDesigner::Client::Components::Editor-container::Editor::Start
- 3.5.2.21 SWEDesigner::Client::Components::Editor-container::Editor::Variabile
- 3.5.2.22 SWEDesigner::Client::Components::Editor-container::Editor::While-node

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::Edit-

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

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

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

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

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

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

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

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

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

3.7.2.3 SWEDesigner::Client::Services::MenuService

4 Specifica Back-End

4.1 SWEDesigner::Server

4.1.1 Informazioni generali

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

4.1.2 Classi

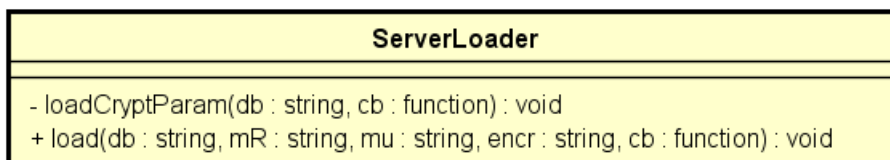


Figura 1: Diagramma della classe SWEDesigner::Server::serverLoader

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* italiano Callback che gestisce le richieste asincrone al database.

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

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

- **Parametri:**

- * *db: string*
Il path del modulo che gestisce la connessione al database.
- * *cb: function* Callback che gestisce le richieste asincrone al database.

4.2 SWEDesigner::Server::Model

4.2.1 Informazioni generali

- **Descrizione:**
Questo package contiene tutte le classi e le funzionalità legate al database.
- **Padre:** SWEDesigner::Server

4.2.2 Classi

4.2.2.1 SWEDesigner::Server::Model::mongooseConnection

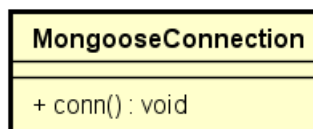


Figura 2: Diagramma della classe SWEDesigner::Server::Model::mongooseConnection

- **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.

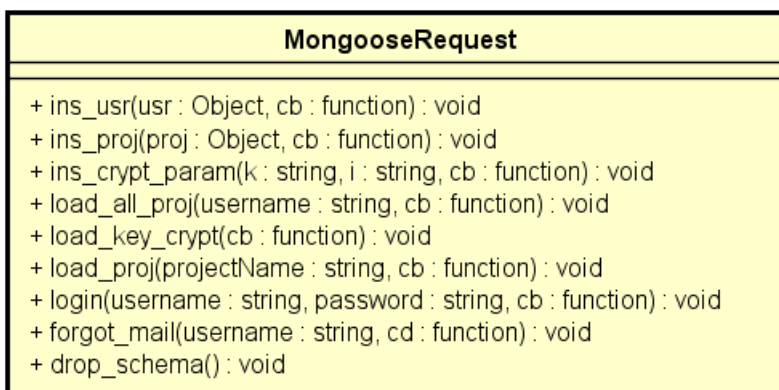


Figura 3: Diagramma della classe SWEDesigner::Server::Model::mongooseRequest

4.2.2.2 SWEDesigner::Server::Model::mongooseRequest

- **Descrizione:**

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

- **Utilizzo:**

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

- **Metodi:**

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

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

- **Parametri:**

- * *usr: Object*

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

- * *cb: function*

- Callback che gestisce le richieste asincrone al database.

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

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

- **Parametri:**

- * *proj: Object*

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

- * *cb: function*

- Callback che gestisce le richieste asincrone al database.

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

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

- **Parametri:**

- * *k: string*

- La chiave crittografica.

- * *i: string*

- Valore iv per la crittografia.

- * *cb: function*

- Callback che gestisce le richieste asincrone al database.

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

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

utente.

– **Parametri:**

* *username: string*

Nome dell'utente di cui sono richiesti i progetti.

* *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

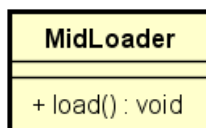


Figura 4: Diagramma della classe SWEDesigner::Server::Controller::Middleware::midLoader

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

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

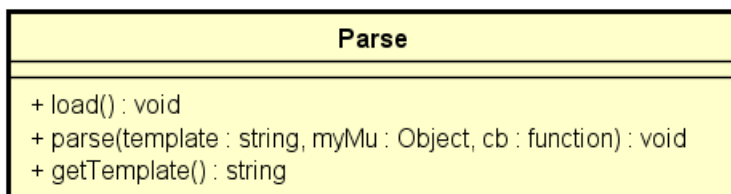


Figura 5: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Parse

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 Moustache.

- *+getTemplate() : string*

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

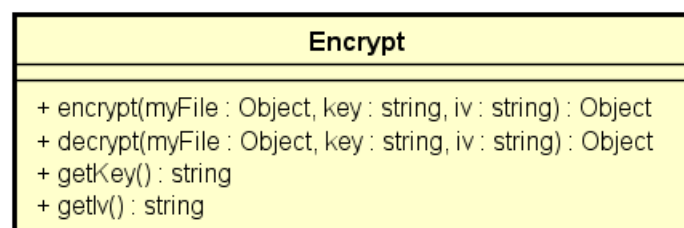


Figura 6: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Encrypt

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 decryptato.

- **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

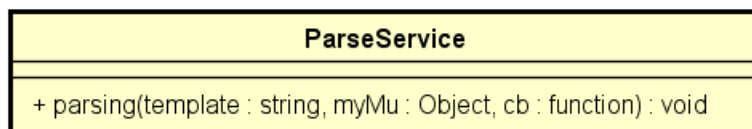


Figura 7: Diagramma della classe SWEDesigner::Server::Controller::Services::parseService

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 Moustache.

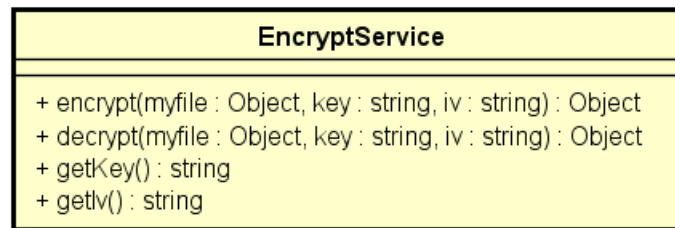


Figura 8: Diagramma della classe SWEDesigner::Server::Controller::Services::encryptService

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 gli algoritmi di Forge.

- **Parametri:**

- * *myFile: Object*
Oggetto JSON da crittare

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

5 Diagrammi di sequenza

5.1 Generazione Codice

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

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

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

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

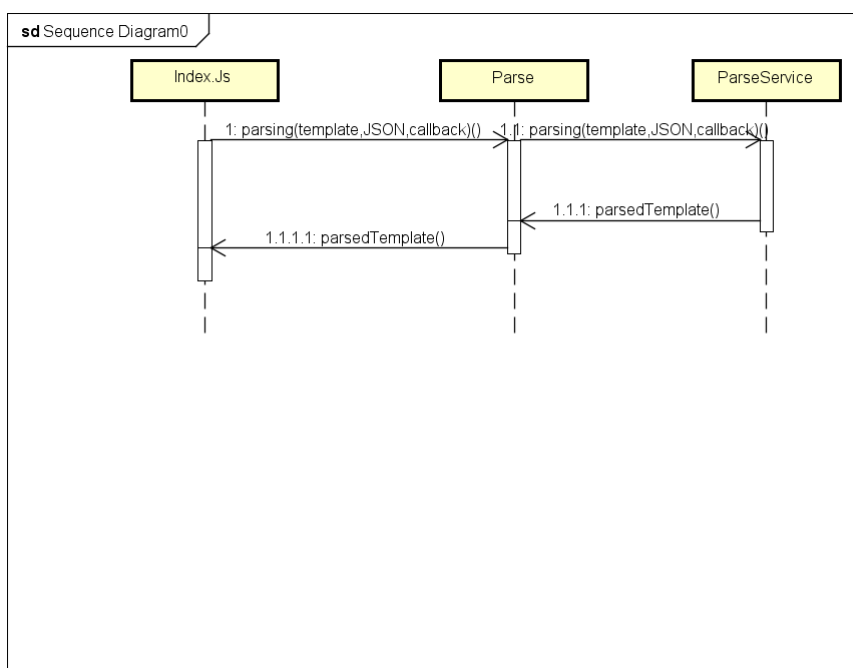


Figura 9: Sequence diagram generazione codice java

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

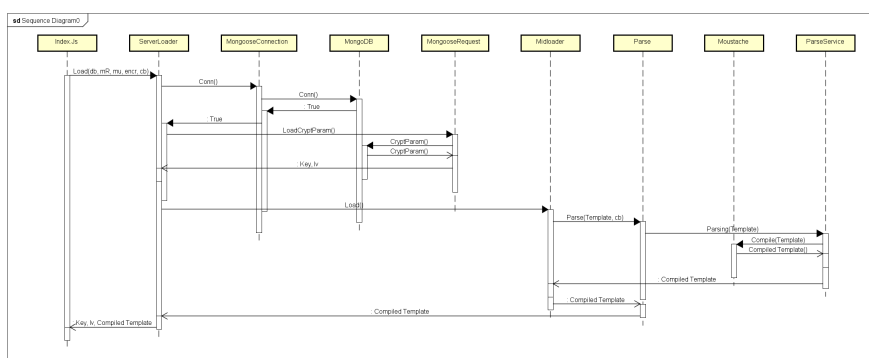


Figura 10: Sequence diagram caricamento moduli server

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.

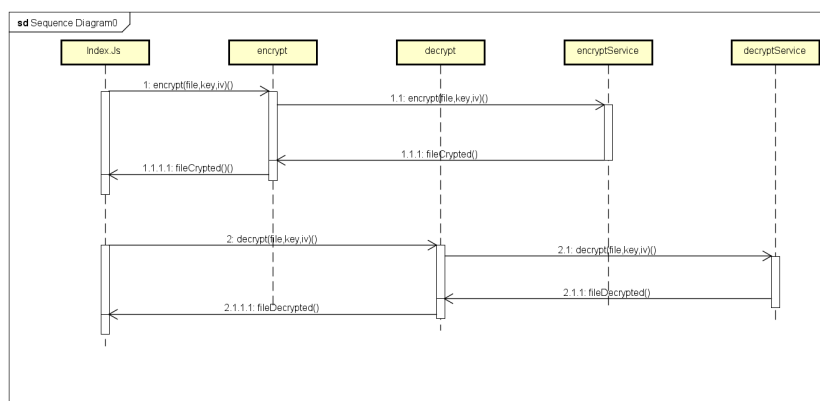


Figura 11: Sequence diagram per operazioni di encrypt e decrypt

6 Tracciamento

6.1 Tracciamento Classi-Requisiti

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

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

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

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

Tabella 2: Tracciamento Classi - Requisiti

6.2 Tracciamento Requisiti-Classi

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

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

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

Tabella 3: Tracciamento Requisiti - Classe