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 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/.

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.
- **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

3.2 SWEDesigner::Client::Components

3.2.1 Informazioni generali

- **Descrizione:**
  Questo package contiene tutti i components dell'applicazione.
- **Padre:** SWEDesigner::Client
- **Package contenuti:**
  - Menu
    Il package contiene tutti i components riguardanti la gestione delle funzionalità fornite dal menu.
  - Editor
    Il package contiene tutte le components riguardanti l'editor dei diagrammi.
  - ActivityFrame
    Il package contiene i components riguardanti la gestione dell'activity frame, per la visione del flusso del programma.

3.2.2 Classi

3.2.2.1 SWEDesigner::Client::Components::AppComponent

- **Descrizione:**
  Questo component descrive un contenitore per la barra di navigazione e le altre componenti dell'applicazione le quali sono istanziate dinamicamente all'interno del template http.
- **Utilizzo:**
  AppComponent è il primo component che viene istanziato tramite bootsrap.

3.2.2.2 SWEDesigner::Client::Components::NavbarComponent

- **Descrizione:**
  Questo component permette la navigazione all'interno dell'applicazione tramite links.
- **Utilizzo:**
  NavbarComponent è istanziato per bootstrap subito dopo dell'AppComponent.

3.2.2.3 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 servendosi dei metodi forniti dal servizio AuthenticationService.
- **Utilizzo:**
  Questo componente viene istanziato dinamicamente dal servizio Router del framework Angular quando viene richiesta la pagina di registrazione.

3.2.2.4 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 servendosi dei metodi forniti dal servizio AuthenticationService.
- **Utilizzo:**
  Questo componente viene istanziato dinamicamente dal servizio Router del framework Angular qunado viene richiesta la pagina di login.

3.3 SWEDesigner::Client::Components::ActivityFrame

3.3.1 Informazioni generali

- **Descrizione:**
  Questo package contiene i components riguardanti la gestione dell'activity frame, per la visione del flusso del programma.
- **Padre:** SWEDesigner::Client::Components

3.3.2 Classi

3.3.2.1 SWEDesigner::Client::Components::Editor::ActivityFrame::ActivityFrameCompone

- **Descrizione:**
  Component che descrive la struttura del frame dove l'utente può visualizzare l'activity frame che rappresenta il flusso logico del programma.
- **Utilizzo:**
  Questo component viene istanziato per bootstrap dopo l'istanziazione del component AppComponent.

3.4 SWEDesigner::Client::Components::Editor

3.4.1 Informazioni generali

- **Descrizione:**
  Il package contiene tutte le components riguardanti l'editor dei diagrammi.
- **Padre:** SWEDesigner::Client::Components

3.4.2 Classi

3.4.2.1 SWEDesigner::Client::Components::Editor::EditorComponent

- **Descrizione:**
  Questo componente contiene la rappresentazione grafica dei diagrammi disegnati dall'utente.
- **Utilizzo:**
  Questo componente viene instanziato dinamicamente dal servizio Router del framework Angular quando viene richiesta la pagina dell'editor diagrammi.
- **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
  - *-connettore: any*
    Il tipo del connettore selezionato
  - *-elementToConnect: any*
    Punta all'elemento selezionato con il click, che sarà collegato con il connettore

- **Metodi:**

  - *-constructor(private classMenuService: ClassMenuService, private editService: EditServiceService, private mainEditorService: MainEditorService): void*
    Questo metodo è il costruttore della classe
  - **Parametri:**

    * *private classMenuService: ClassMenuService*
      Service ClassMenuService
    * *private editService: EditServiceService*
      Service EditServiceService
    * *private mainEditorService: MainEditorService*
      Service MainEditorService

  - *+replaceDiagram(graph: JSON): void*
    Questo metodo viene utilizzato per rimpiazzare l'editor con una nuova finestra contenuta nel file JSON
  - **Parametri:**

    * *graph: JSON*
      Grafico da aprire in formato JSON

- *+selectElementsToConnect(cell: any): void*
  Questo metodo viene utilizzato per selezionare gli elementi da collegare con il connettore selezionato
- **Parametri:**
  - *cell: any*
    Elemento selezionato
- *+elementSelection(cellView: any): void*
  Questo metodo seleziona un elemento nell'editor
- **Parametri:**
  - *cellView: any*
    Elemento selezionato
- *+addConnettore(connettore: any): void*
  Aggiunge il connettore alla classe
- **Parametri:**
  - *connettore: any*
    Connettore da aggiungere
- *+addElement(element: any): void*
  Questo metodo aggiunge un elemento all'editor
- **Parametri:**
  - *element: any*
    Elemento da aggiungere all'editor
- *+zoomIn(): void*
  Questo metodo incrementa la scala dell'editor
- *+zoomOut(): void*
  Questo metodo decrementa la scala dell'editor
- *+cloneElement(): void*
  Questo metodo clona l'elemento selezionato

3.4.2.2 SWEDesigner::Client::Components::Editor::ClassMenuComponent

- **Descrizione:**
  Questo component permette la modifica dei campi dati di un oggetto selezionato nell'editorComponent.
- **Utilizzo:**
  Questo component è figlio di editorComponent viene visualizzato quando viene selezionato un elemento editabile nell'editorComponent.
- **Metodi:**

3.4.2.3 SWEDesigner::Client::Components::Editor::ToolbarComponent

- **Descrizione:**
  La classe si occupa di fornire una toolbar per l'inserimento degli elementi del diagramma delle attività o del diagramma delle classi.
- **Utilizzo:**
  Ogni volta che viene selezionato un elemento esso viene inserito sul grafico. Nel caso dei connettori occorre selezionare, successivamente al connettore, i due elementi da collegare.
- **Metodi:**

  - *+addClasse(): void*
    Il metodo aggiunge una classe di nome "Classe" nell'area di disegno;
  - *+addAstratta(): void*
    Il metodo aggiunge una classe astratta di nome "ClasseAstratta" nell'area di disegno;
  - *+addInterfaccia(): void*
    Il metodo aggiunge un interfaccia di nome "Interfaccia" nell'area di disegno;
  - *+addGeneralizzazione(): void*
    Il metodo seleziona il tipo di connettore "Generalizzazione";
  - *+addImplementazione(): void*
    Il metodo seleziona il tipo di connettore "Implementazione";
  - *+addCommento(): void*
    Il metodo aggiunge un elemento di tipo "Commento" nell'area di disegno;
  - *+addAssociazione(): void*
    Il metodo seleziona il tipo di connettore "Associazione";
  - *+addConnettore(cellView: any): void*
    Il metodo serve, in caso venga selezionato un connettore, a selezionare i

due elementi da collegare con il connettore selezionato con uno dei metodi precedenti.

– **Parametri:**

* *cellView: any*
  Elemento da selezionare per essere collegato con il connettore selezionato

3.5 SWEDesigner::Client::Components::Menu

3.5.1 Informazioni generali

- **Descrizione:**
  Il package contiene tutti i components riguardanti la gestione delle funzionalità fornite dal menu.
- **Padre:** SWEDesigner::Client::Components

3.5.2 Classi

3.5.2.1 SWEDesigner::Client::Components::Menu::MenuComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per la gestione dei progetti, dei propri dati personali, e della rappresentazione dei grafici su cui sta lavorando.
- **Utilizzo:**
  Component che viene istanziato per bootstrap dopo che è stato istanziato il component appComponent.

3.5.2.2 SWEDesigner::Client::Components::Menu::FileComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per la gestione del progetto attualmente in uso.
- **Utilizzo:**
  Component che viene istanziato per bootstrap dopo che è stato istanziato il component menuComponent.

3.5.2.3 SWEDesigner::Client::Components::Menu::LayerComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per la gestione dei layer del progetto in uso.
- **Utilizzo:**
  Component che viene istanziato per bootstrap dopo che è stato istanziato il component menuComponent.

3.5.2.4 SWEDesigner::Client::Components::Menu::ProgettoComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per la gestione dei propri progetti salvati.
- **Utilizzo:**
  progettoComponent viene istanziato per bootstrap dopo che è stato istanziato il component menuComponent.

3.5.2.5 SWEDesigner::Client::Components::Menu::ProfiloComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per la gestione dei propri dati personali.
- **Utilizzo:**
  Component che viene istanziato per bootstrap dopo che è stato istanziato il component menuComponent.

3.5.2.6 SWEDesigner::Client::Components::Menu::ModificaComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per la modifica del progetto in uso, come ad esempio effettuare lo zoom, oppure eliminare o copiare un elemento selezionato.
- **Utilizzo:**
  Component che viene istanziato per bootstrap dopo che è stato istanziato il component menuComponent.
- **Metodi:**
  - *+doZoomIn(): void* Esegue lo zoomIn
  - *+doZoomOut(): void* Esegue lo zoomOut

3.5.2.7 SWEDesigner::Client::Components::Menu::TemplateComponent

- **Descrizione:**
  Component che contiene l'insieme di funzionalità fornite all'utente per l'importazione e gestione dei template.
- **Utilizzo:**
  Component che viene istanziato per bootstrap dopo che è stato istanziato il component menuComponent.

3.6 SWEDesigner::Client::Services

3.6.1 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.

3.6.2 Classi

3.6.2.1 SWEDesigner::Client::Services::MenuService

- **Descrizione:**
  Classe che definisce i metodi per le operazioni fornite all'utente dal menu.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal component menuComponent.
- **Attributi:**
  - *-selectedGraphService: Subject<any>*
  - *-selectedGrapg: Observable<any>*

- **Metodi:**
  - *+zoomIn(): void*
    Aumenta la dimensione degli oggetti nell'editor
  - *+zoomOut(): void*
    Diminuisce la dimensione degli oggetti nell'editor

3.6.2.2 SWEDesigner::Client::Services::MainEditorService

- **Descrizione:**
  Classe che definisce i metodi per le operazioni all'interno dei diagrammi e la comunicazione tra componenti e server.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dai component editorComponent e classMenuComponent.
- **Attributi:**
  - *-Project: Global*
    Si utilizza per memorizzare e recuperare informazione riguardo il progetto corrente
  - *-selectedClasse: Classe*
    Memorizza la classe corrispondente di tipo "Classe" della classe selezionata nel canvas dell'editor
  - *-editorComp: EditorComponent*
    Si utilizza per accedere direttamente all'EditorComponent
  - *-graph: JSON*
    Si utilizza per per salvare il grafico dell'editor
  - *-activityMode: boolean*
    Indica se il diagramma delle attività è in uso
- **Metodi:**
  - *+setEditorComp(editCmp: EditorComponent): void*
    Questo metodo viene usato per l'istanziazione dell'EditorComponent come proprietà interna di questa classe
  - **Parametri:**

    * *editCmp: EditorComponent*
      L'istanza dell'EditorComponent

- *+getClassList(): Classe[]*
  Questo metodo viene usato per richiamare l'array di classi presente nel progetto
- *+getSelectedClasse(): void*
  Questo metodo ritorna la classe selezionata di tipo "Classe"
- *+addClass(classe: Classe, graphElement: any): void*
  Questo metodo aggiunge un oggetto di tipo classe nell'array di classi del progetto
- **Parametri:**

    * *classe: Classe*
      Questo oggetto è una rappresentazione, di tipo "Classe" o "ClasseAstratta", del parametro graphelement
    * *graphElement: any*
      Questo è un elemento della libreria grafica JointJs

- *+selectClasse(nome: string): Classe*
  Questo metodo cerca, all'interno della collezione di classi del progetto, una classe con lo stesso nome di quello fornito come parametro
- **Parametri:**

    * *nome: string*
      Nome della classe da cercare

- *+setActivityMode(): void*
  Questo metodo setta a True il valore di activityMode
- *+setClassMode(): void*
  Questo metodo setta a False il valore di activityMode
- *+getActivityModeStatus(): boolean*
  Questo metodo ritorna il valore di activityMode
- *+addAttributo(tipo: string, nome: string, acc: string): void*
  Questo metodo richiama il metodo addAttributo della "selectedClasse"
- **Parametri:**

    * *tipo: string*
      Il tipo dell'attributo da aggiungere con addAttributo

    * *nome: string*
      Il nome dell'attributo da aggiungere con addAttributo
    * *acc: string*
      La visibilità dell'attributo da aggiungere con addAttributo

- *+removeAttributo(nome: string): void*
  Questo metodo richiama il metodo removeAttr della "selectedClasse"
- **Parametri:**

    * *nome: string*
      Il nome dell'attributo da rimuovere

- *+storeGraph(graph: JSON): void*
  Questo metodo salva in "this.graph" il grafico passato come parametro
- **Parametri:**

    * *graph: JSON*
      Un grafico in formato JSON

- *+enterClassMode(): void*
  Questo metodo viene utilizzato per ripristinare il diagramma delle classi memorizzato in "this.graph"
- *+addMetodo(tipo: string, nome: string, acc: string, listArgs?: any): void*
  Questo metodo aggiunge un nuovo metodo alla "selectedClasse"
- **Parametri:**

    * *tipo: string*
      Tipo di ritorno del metodo
    * *nome: string*
      Nome del metodo
    * *acc: string*
      La visibilità del metodo
    * *listArgs?: any*
      Lista dei parametri del metodo, se ce ne sono

- *+removeMetodo(nome: string): void*
  Questo metodo richiama il metodo removeMetodo della "selectedClasse"
- **Parametri:**

    * *nome: string*
      Nome del metodo da eliminare

  – *+enterActivityMode(name: string): void*
    Questo metodo cerca un metodo nella "selectedClasse" e recupera il suo diagramma per chiamare il metodo replaceDiagram dell'editorComp, il quale carica i metodi del diagramma in Canvas

  – **Parametri:**

    * *name: string*
      Nome del metodo da trovare

3.6.2.3 SWEDesigner::Client::Services::ToolbarService

- **Descrizione:**
  Classe che definisce i metodi per le operazioni di inserimento di nuovi elementi all'interno dell'editor di diagrammi.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal component editorComponent.

3.6.2.4 SWEDesigner::Client::Services::ActivityFrameService

- **Descrizione:**
  Classe che definisce i metodi per le operazioni di navigazione tra i metodi all'interno dell'activity frame.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal component activityFrameComponent.

3.6.2.5 SWEDesigner::Client::Services::ClassMenuService

- **Descrizione:**
  Classe che definisce i metodi per le operazioni di modifica di un elemento selezionato all'interno del diagramma rappresentato.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal component classMenuService.
- **Attributi:**

  - *-selectedClassSource: Subject<any>*
    Subject della classe selezionata
  - *+selectedClass: Observable<any>*
    Observable della classe selezionata

- **Metodi:**

  - *+classSelection(classe: any): void*
    Aggiorna il subject della classe

3.6.2.6 SWEDesigner::Client::Services::AccountService

- **Descrizione:**
  Classe che definisce i metodi di registrazione, login e recupero dati utente dal server.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dai component registrationComponent e loginComponent.

3.7 SWEDesigner::Client::Services::Models

3.7.1 Informazioni generali

- **Descrizione:**
  Il package contiene moduli necessari a storicizzare i dati inseriti all'interno dei diagrammi.
- **Padre:** SWEDesigner::Client::Services

3.7.2 Classi

3.7.2.1 SWEDesigner::Client::Services::Param

- **Descrizione:**
  Classe che definisce i metodi di settaggio e richiesta dei parametri nome e tipo.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal model attributo.
- **Attributi:**

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

- **Metodi:**

  - *-constructor(tipo: string, nome: string): void*
    Costruttore
  - **Parametri:**

    * *tipo: string*
      Tipo del parametro
    * *nome: string*
      Nome del parametro
  - *+getTipo(): string*
    Ritorna il tipo del parametro
  - *+getNome(): string*
    Ritorna il nome del parametro
  - *+changeTipo(tipo: string): void*
    Modifica il tipo del parametro
  - **Parametri:**

    * *tipo: string*
      Nuovo tipo
  - *+changeNome(nome: string): void*
    Modifica il nome del parametro
  - **Parametri:**

    * *nome: string*
      Nuovo nome

3.7.2.2 SWEDesigner::Client::Services::Attributo

- **Descrizione:**
  Classe derivata da Param che definisce i metodi di settaggio e richiesta dei parametri di visibilità.

- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal model classe.
- **Metodi:**

- *-constructor (tipo: string, nome: string, acc: string): void*
  Costruttore
- **Parametri:**
  - *tipo: string*
    Tipo dell'attributo
  - *nome: string*
    Nome dell'attributo
  - *acc: string*
    Accessibilità dell'attributo
- *+getAccesso(): string*
  Rstituisce l'accessibilità dell'attributo
- *+changeAccesso(acc: string): void*
  Modifica l'accessibilità dell'attributo
- **Parametri:**
  - *acc: string*
    Nuova accessibilità

3.7.2.3 SWEDesigner::Client::Services::Metodo

- **Descrizione:**
  Classe che definisce i metodi di settaggio e richiesta dei metodi definiti all'interno dei diagrammi.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal model classe.
- **Attributi:**
  - *+nome: string*
    Nome del metodo
  - *+accesso: string*
    Visibilità del metodo

  - *+tipoRitorno: string*
    Tipo di ritorno del metodo
  - *+listaArgomenti: Param[]*
    Lista argomenti del metodo
  - *+diagramma: JSON*
    Definisce il metodo corrente in JSON

- **Metodi:**

  - *-constructor(nome: string, acc: string, tipo: string, listaArg?: Param[]): void*
    Costruttore
  - **Parametri:**

    * *nome: string*
      Nome del metodo
    * *acc: string*
      Visibilità del metodo
    * *tipo: string*
      Tipo di ritorno del metodo
    * *listaArg?: Param[]*
      Lista argomenti del metodo
  - *+changeNome(name: string): void*
    Modifica il nome del metodo
  - **Parametri:**

    * *name: string*
      Nuovo nome del metodo
  - *+changeTipoRitorno(tipo: string): void*
    Modifica il tipo di ritorno del metodo
  - **Parametri:**

    * *tipo: string*
      Nuovo tipo di ritorno del metodo
  - *+changeAccesso(acc: string): void*
    Modifica la visibilità del metodo

- **Parametri:**
  - *acc: string*
    Nuova visibilità del metodo
- *+changeListaArg(listArg: Param[]): void*
  Cambia il riferimento all'array dei parametri formali
- **Parametri:**
  - *listArg: Param[]*
    Array dei parametri
- *+addArgomento(arg: Param) : void*
  Aggiunge un nuovo parametro al metodo
- **Parametri:**
  - *arg: Param*
    Parametro
- *+addDiagram(dia: JSON): void*
  Assegna il file JSON al diagramma degli attributi
- **Parametri:**
  - *dia: JSON*
    File JSON
- *+getDiagram(): void*
  Ritorna il diagramma del metodo
- *+getNome(): string*
  Ritorna il nome del metodo
- *+getAccesso(): string*
  Ritorna il tipo di visibilità del metodo
- *+getTipoRitorno(): string*
  Ritorna il tipo di ritorno del metodo
- *+getListaArgomenti(): Param[]*
  Ritorna l'array degli argomenti

3.7.2.4 SWEDesigner::Client::Services::Classe

- **Descrizione:**
  Classe che definisce i metodi di settaggio e richiesta di tutti gli elementi che sono contenuti in una classe. Contiene un array di metodi, con le relative rappresentazioni grafiche dei metodi implementati, e un array di attributi, oltre ai campi utili all'identificazione della classe.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal model global.
- **Attributi:**
  - *-nome: string*
    Il nome della classe
  - *-attributi: Attributo[]*
    Array di attributi della classe
  - *-metodi: Metodo[]*
    Array di metodi della classe
  - *-classePadre: string*
    La classe estesa da questa classe
- **Metodi:**
  - *-constructor(nome: string): void*
    Costruisce un nuovo oggetto di tipo classe e gli assegna un nome
  - **Parametri:**
    - *nome: string*
      Nome della classe
  - *+addAttributo(tipo: string, nome: string, acc?: string): void*
    Aggiunge un nuovo attribuo alla lista degli attributi dopo aver controllato che non ne esista già uno con lo stesso nome
  - **Parametri:**
    - *tipo: string*
      Tipo del nuovo attributo
    - *nome: string*
      Nome del nuovo attributo
    - *acc?: string*
      Visibilità dell'attributo

- *+addSottoclasse(superclass: string): void*
  Inserisce il nome della classe che questa classe estende
- **Parametri:**
  - *superclass: string*
    Nome della classe padre
- *+addMetodo(metodo: Metodo): void*
  Aggiunge un nuovo metodo
- **Parametri:**
  - *metodo: Metodo*
    Metodo precostruito
- *+changeNome(name: string): void*
  Modifica il nome della classe
- **Parametri:**
  - *name: string*
    Nuovo nome della classe
- *+changeAttr(nomeAttr: string, tipo?: string, nuovoNome?: string, acc?: string): void*
  Modifica un attributo, se questo è presente nell'array degli attributi
- **Parametri:**
  - *nomeAttr: string*
    Nome dell'attributo da modificare
  - *tipo?: string*
    Nuovo tipo dell'attributo
  - *nuovoNome?: string*
    Nuovo nome dell'attributo
  - *acc?: string*
    Nuova visibilità dell'attributo
- *+removeAttr(nomeAttr: string): void*
  Rimuove un attributo dalla lista degli attributi
- **Parametri:**

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

- *+removeMetodo(nomeMetodo: string): void*
  Rimuove un metodo dall'array dei metodi
- **Parametri:**

    - *nomeMetodo: string*
      Nome del metodo da eliminare

- *+getNome() : string*
  Ritorna il nome della classe
- *+getAttributi(): Attributo[]*
  Ritorna l'array degli attributi
- *+getMetodi(): Metodo[]*
  Ritorna l'array dei metodi
- *+retriveMethod(name: string): Metodo*
  Ritorna un metodo dall'array dei metodi se è presente un metodo con quel nome
- **Parametri:**

    - *name: string*
      Nome del metodo

- *+getSottoclasse(): string*
  Ritorna il nome della superclasse
- *+toJSON(): JSON*
  Effettua override della funziona toJSON

3.7.2.5 SWEDesigner::Client::Services::ClasseAstratta

- **Descrizione:**
  Classe derivata da classe che definisce i metodi di settaggio e richiesta dei parametri di una classe astratta.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal model global.
- **Attributi:**

  - *-abstractMethods: MetodiAstratti[]*
    Contiene la lista dei metodi della classe

- **Metodi:**

  - *+addAbstractMethods(nome: string, tipo: string, acc:string, listaParam: string[]): void*
    Questo metodo aggiunge un metodo alla classe astratta
  - **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
  - *+toJSON()): void*
    Questo metodo effettua il parsing della classe selezionata e lo trasforma in JSON

3.7.2.6 SWEDesigner::Client::Services::Interface

- **Descrizione:**
  Classe derivata da classe che definisce i metodi di settaggio e richiesta dei parametri di una interface.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal model global.
- **Attributi:**

  - *+nome: string* Nome dell'Interfaccia
- **Metodi:**

  - *-constructor(nome: string): void* Costruttore
  - **Parametri:**

    * *nome: string*
      Nome dell'interfaccia

3.7.2.7 SWEDesigner::Client::Services::Global

- **Descrizione:**
  Classe che definisce i metodi di settaggio e richiesta di tutte le classi contenenti nel diagramma delle classi.
- **Utilizzo:**
  É istaziata dal framework Angular e i suoi metodi sono utilizzati dal servizio editorService.
- **Attributi:**
  - *-nomeprogetto: string*
    Nome del progetto
  - *-diagramma: string*
    JSON convertito in string del diagramma
  - *-classi: Classe[]*
    Array di classi
- **Metodi:**
  - *+addClasse(nome: string): void*
    Nome del progetto
  - *+changeTitolo(titolo: string): void*
    JSON convertito in string del diagramma
  - *+setDiagramma(diagramma: string): void*
    Setta l'attributo diagramma della classe
  - **Parametri:**
    * *diagramma: string*
      Diagramma
  - *+getDiagramma(): string*
    Ritorna il diagramma del progetto
  - *+getTitolo(): string*
    Ritorna il nome del progetto

- *+getClassi(): Classe[]*
  Ritorna la collezione di classi
- *+toJSON(): string* Ritorna un JSON del progetto in formato string

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

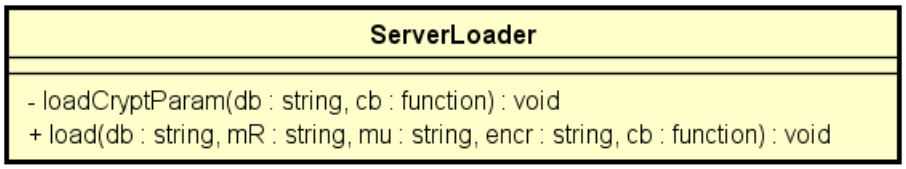

Figura 1: Diagramma della classe SWEDesigner::Server::serverLoader

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

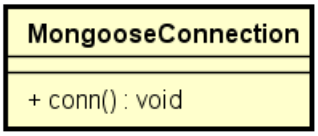

Figura 2: Diagramma della classe SWEDesigner::Server::Model::mongooseConnection

- **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.

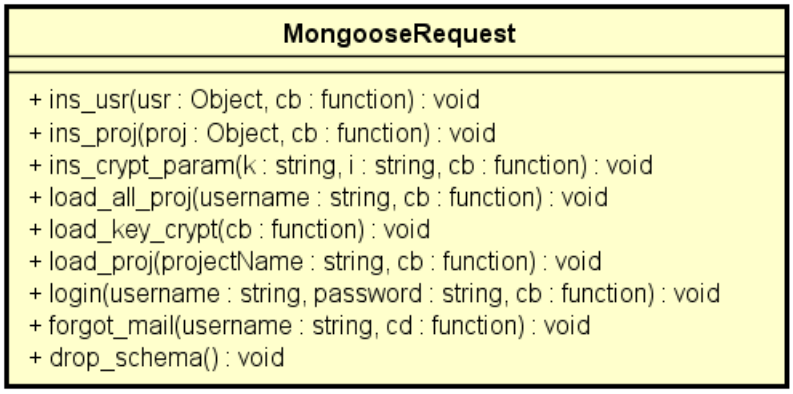

Figura 3: Diagramma della classe SWEDesigner::Server::Model::mongooseRequest

4.2.2.2 SWEDesiger::Server::Model::mongooseRequest

Tutte le query riguardanti l'aggiornamento e la cancellazione di dati dal database verranno trattate nella successiva versione di questo documento.

- **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, cd: 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.

– *+`drop_schema`() : void*
Si tratta della funzione che elimina il database.

4.3 SWEDesigner::Server::Controller::Middleware

4.3.1 Informazioni generali

In questa versione del documento sono omesse le classi errHandler e Profile pociché verranno definite in seguito.

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

4.3.2 Classi

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

- **Descrizione:**
  La classe contenente i metodi di caricamento dei servizi utilizzati dalle componenti middleware

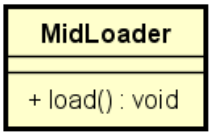

Figura 4: Diagramma della classe SWEDesigner::Server::Controller::Middleware::midLoader

- **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

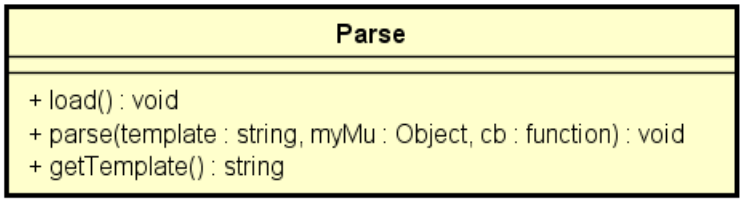

Figura 5: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Parse

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.

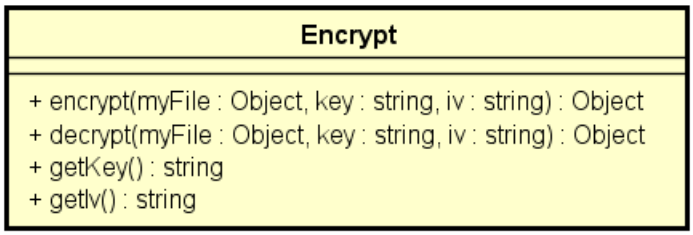

Figura 6: Diagramma della classe SWEDesigner::Server::Controller::Middleware::Encrypt

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

In tale versione del documento non sarà trattato il servizio di Profile poiché verrà trattato nelle versioni successive.

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

4.4.2 Classi

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

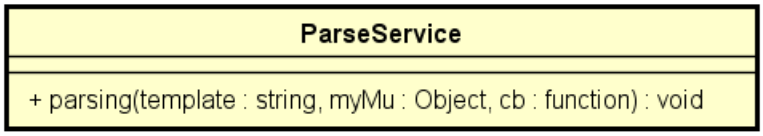

Figura 7: Diagramma della classe SWEDesigner::Server::Controller::Services::parseService

- **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.

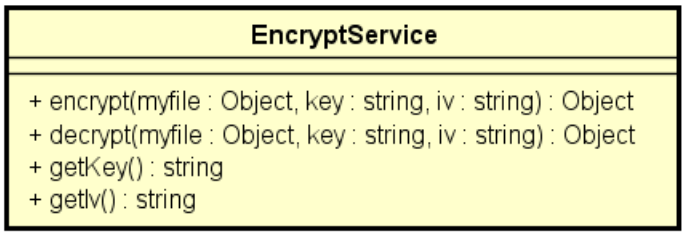

Figura 8: Diagramma della classe SWEDesigner::Server::Controller::Services::encryptService

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

6 Tracciamento

6.1 Tracciamento Classi-Requisiti

6.2 Tracciamento Requisiti-Classi

6.3 Tracciamento Componenti-Requisiti

6.4 Tracciamento Requisiti-Componenti