Manuale Sviluppatore

Gruppo JurassicSWE · Progetto IronWorks

JurassicSWE@gmail.com

Informazioni sul documento

Versione	2.0.0
Redazione	Gruppo JurassicSWE
Verifica	Gianluca Travasci
Approvazione	Leo Moz
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo JurassicSWE

Sommario

Tale documento contiene linee guida e spiegazioni utili per coloro che vorranno estendere il prodotto IronWorks.

Registro delle modifiche

Versione	Data	Ruolo	Nominativo	Descrizione
2.0.0	2018-09-06	Responsabile	Lidia Alecci	Approvazione documento
1.1.0	2018-09-06	Verificatore	Gianluca Travasci	Verifica documento
1.0.7	2018-09-06	Programmatore	Leo Moz	Aggiunta §4.3
1.0.6	2018-09-04	Programmatore	Leo Moz	Correzione diagrammi di Figura 3 e 4
1.0.5	2018-09-04	Programmatore	Daniele Dal Maso	Espansione §5.3.1, §5.3.2, §5.3.3, §5.3.4 e §5.3.5
1.0.4	2018-09-04	Programmatore	Daniele Dal Maso	Correzione §5.2.2.1 e §5.2.3.1
1.0.3	2018-09-04	Programmatore	Daniele Dal Maso	Correzione diagrammi di Figura 1 e 5, aggiunti diagrammi di Figura 6 e 7
1.0.2	2018-09-04	Programmatore	Leo Moz	Correzione §5.2, §5.3 e §5.3.2.1
1.0.1	2018-09-02	Programmatore	Leo Moz	Correzione §4.2 e correzioni minori sulla base del feedback RQ.
1.0.0	2018-08-16	Responsabile	Leo Moz	Approvazione documento
0.1.0	2018-08-16	Verificatore	Gianluca Travasci	Verifica documento
0.0.9	2018-08-14	Programmatore	Marco Masiero	Stesura §6
0.0.8	2018-08-14	Programmatore	Daniele Dal Maso	Stesura §5.2.1 e sottosezioni
0.0.7	2018-08-13	Programmatore	Gianluca Travasci	Stesura §5
0.0.6	2018-08-12	Programmatore	Daniele Dal Maso	Inseriti i diagrammi delle componenti
0.0.5	2018-07-29	Programmatore	Lidia Alecci	Stesura §3 e §4.1
0.0.4	2018-07-28	Programmatore	Marco Masiero	Stesura Glossario

Versione	Data	Ruolo	Nominativo	Descrizione
0.0.3	2018-07-27	Programmatore	Lidia Alecci	Stesura §2
0.0.2	2018-07-12	Programmatore	Lidia Alecci	Stesura §1
0.0.1	2018-07-12	Programmatore	Lidia Alecci	Creazione template e stesura dell'indice

Indice

1 Introduzione **7**
1.1 Scopo del documento . . . 7
1.2 Scopo del prodotto . . . 7
1.3 Informazioni utili . . . 7
1.4 Riferimenti . . . 7
1.4.1 Riferimenti normativi . . . 7
1.4.2 Riferimenti informativi . . . 8

2 Tecnologie impiegate **8**
2.1 Client . . . 8
2.1.1 JointJS . . . 8
2.1.1.1 Vantaggi . . . 9
2.1.1.2 Svantaggi . . . 9
2.1.2 Backbone.js . . . 9
2.1.2.1 Vantaggi . . . 9
2.1.2.2 Svantaggi . . . 9
2.1.3 Bootstrap . . . 9
2.1.3.1 Vantaggi . . . 10
2.1.3.2 Svantaggi . . . 10
2.2 Server . . . 10
2.2.1 NodeJS . . . 10
2.2.1.1 Vantaggi . . . 10
2.2.1.2 Svantaggi . . . 10
2.2.2 ExpressJS . . . 10
2.2.2.1 Vantaggi . . . 10
2.2.2.2 Svantaggi . . . 11

3 Requisiti di sistema **12**
3.1 Browser supportati . . . 12

4 Ambiente di lavoro **13**
4.1 Installazione prerequisiti . . . 13
4.1.1 Node.js . . . 13
4.1.2 Npm . . . 13
4.2 Installazione e avvio dell'applicazione . . . 13
4.3 Utilizzo del codice generato . . . 13
4.3.1 Contenuto dello zip . . . 13
4.3.2 Preparativi . . . 14
4.3.3 Hibernate e dipendenze . . . 14
4.3.4 MySQL e il database . . . 15
4.3.5 I file XML . . . 15
4.3.6 Usare e compilare i file Java . . . 15

5 Architettura **17**
5.1 Architettura Generale . . . 17
5.2 Client-side . . . 17
5.2.1 Client-side::Collections . . . 19
5.2.1.1 Classi Contenute . . . 19
5.2.2 Client-side::Model . . . 20
5.2.2.1 Classi Contenute . . . 21
5.2.3 Client-side::View . . . 22
5.2.3.1 Classi Contenute . . . 23
5.3 Server-side . . . 24
5.3.1 Server-side::ApplicationTier . . . 25
5.3.1.1 Classi Contenute . . . 25
5.3.2 Server-side::ApplicationTier::Generators . . . 25
5.3.2.1 Classi Contenute . . . 25
5.3.3 Server-side::PresentationTier . . . 26
5.3.3.1 Package Contenuti . . . 27
5.3.4 Server-side::PresentationTier::Controller . . . 27
5.3.4.1 Classi Contenute . . . 27
5.3.5 Server-side::PresentationTier::Middleware . . . 27
5.3.5.1 Classi Contenute . . . 27

6 Estensione **28**
6.1 Generare in altri linguaggi . . . 28
6.2 Continuità tra diagrammi . . . 28
6.3 Estendere le possibilità di login . . . 28

A Glossario **29**

Elenco delle figure

1 Diagramma dei package lato client 17
2 Componente Collections 19
3 Componente Model 20
4 Componente View 22
5 Diagramma dei package lato server 24
6 Diagramma dei package livello Application 25
7 Diagramma dei package livello Presentation 26

1 Introduzione

1.1 Scopo del documento

Tale documento ha lo scopo di di fornire spiegazioni e linee guida utili a chi volesse estendere o mantenere il software. Inoltre, all'interno di questo documento sono spiegate le modalità di installazione e di utilizzo, i requisiti necessari per poterlo utilizzare, librerie e *framework*$_g$ esterni utilizzati per lo sviluppo dell'applicazione e struttura di *package*$_g$ e classi utilizzate.

1.2 Scopo del prodotto

Lo scopo del *prodotto*$_g$ è quello di realizzare un software, in particolare un'*applicazione web*$_g$, per disegnare diagrammi *UML*$_g$ di robustezza, e di un generatore di codice che, a partire dalle definizioni contenute in un diagramma, produca il codice di classi *Java*$_g$ per ospitare i dati delle entità persistenti, ed i metodi per leggere e scrivere questi dati in un *database relazionale*$_g$.

1.3 Informazioni utili

All'interno del manuale si assume che l'utente a cui è destinato il documento possegga conoscenze, almeno basilari, dei linguaggi di programmazione usati, nello specifico: *HTML5*$_g$, *CSS3*$_g$ e *JavaScript*$_g$ (ECMAScript 6).
Per completezza all'appendice A è riportato il glossario, il quale contiene le definizioni dei termini tecnici usati all'interno del manuale. Per identificarli, i termini sono scritti in *corsivo* con una "g" pedice, solo alla loro prima occorrenza.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Norme di Progetto**: *Norme di Progetto v4.0.0*;
- **Analisi dei Requisiti**: *Analisi dei Requisiti v3.0.0*;
- **Capitolato d'appalto C5:** IronWorks: utilità per la costruzione di software robusto `http://www.math.unipd.it/~tullio/IS-1/2017/Progetto/C5.pdf` (consultato il 2018-07-27).
- **Verbale di incontro interno** con i componenti del gruppo del 2018-06-25;

1.4.2 Riferimenti informativi

- **Progettazione software - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L10.pdf (consultato il 2018-07-27);
- **Model-View Patterns - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/E11.pdf (consultato il 2018-07-27)
- **Documentazione Node.js**
  https://nodejs.org/docs/latest-v9.x/api/ (consultato il 2018-07-27);
- **Documentazione Backbone.js**
  http://backbonejs.org/ (consultato il 2018-07-27);
- **Documentazione Joint.js**
  https://resources.jointjs.com/docs/jointjs/v2.1/joint.html (consultato il 2018-07-27);
- **Documentazione Express.js**
  http://expressjs.com/en/4x/api.html (consultato il 2018-07-27);

2 Tecnologie impiegate

Vengono di seguito illustrate le tecnologie impiegate, lato *client*$_g$ e *server*$_g$, nella realizzazione dell'applicazione.

2.1 Client

2.1.1 JointJS

JointJS è una libreria JavaScript ricca di funzionalità, che permette la creazione di diagrammi interattivi. Tra le principali funzionalità che offre troviamo:

- creazione di elementi per diagrammi base (e.g. rettangoli, cerchi, linee);
- creazione di elementi custom basati su *SVG*$_g$;
- creazione collegamenti personalizzabili (frecce, etichette);
- altamente *event-driven*$_g$;
- convertire diagrammi in file *JSON*$_g$ e viceversa.

2.1.1.1 Vantaggi

- *Open-source*$_g$;
- tutorial divisi in base al livello di conoscenza dell'utente;
- presenza di esempi con il relativo codice;
- buona documentazione.

2.1.1.2 Svantaggi

- Tutorial molto specifici per un determinato problema (non permettono di imparare in modo semplice le basi);
- pochi esempi e molto specifici.

2.1.2 Backbone.js

Backbone è una libreria JavaScript basata su una architettura *MV**$_g$. Ci sono quattro componenti base:

- Backbone.Model: che rappresenta il modello;
- Backbone.Collection: una lista di modelli;
- Backbone.View: la vista;
- Backbone.Router: *routing*$_g$ che gestisce le pagine dell'applicazione.

2.1.2.1 Vantaggi

- Open-source;
- minimale con gli strumenti di base per organizzare le proprie applicazioni.

2.1.2.2 Svantaggi

- Non ha il *two-way data binding*$_g$ predefinito;
- non possiede soluzioni ricche come altri framework più completi.

2.1.3 Bootstrap

Framework open-source usato per la creazione di pagine web, contiene template HTML preconfezionati e temi applicabili in qualsiasi contesto.

2.1.3.1 Vantaggi

- Varietà di scelta;
- supporta il *responsive web design*$_g$.

2.1.3.2 Svantaggi

- È una libreria piuttosto pesante;
- è difficile modificare i template preconfezionati.

2.2 Server

2.2.1 NodeJS

Framework per far girare JavaScript lato server.

2.2.1.1 Vantaggi

- Approccio asincrono e basato su eventi, in grado di elaborare la maggior quantità di richieste in modo più efficiente;
- Architettura modulare, si appoggia a *npm*$_g$, che permette allo sviluppatore di accedere ai package messi a disposizione dalla community.

2.2.1.2 Svantaggi

- Ha un singolo processo per gestire più richieste. Se si verifica un problema, l'intera istanza di Node.js si chiude con la perdita di tutti i dati salvati negli oggetti.

2.2.2 ExpressJS

È un framework veloce, minimale e open-source per applicazioni web Node.js.

2.2.2.1 Vantaggi

- Migliora l'implementazione del sistema di routing;
- open-source;

- è molto utile quando ci si occupa di applicazioni web che gestiscono molte richieste e interazioni con l'utente.

2.2.2.2 Svantaggi

- Refactoring del codice oneroso quando il progetto raggiunge dimensioni elevate.

3 Requisiti di sistema

3.1 Browser supportati

Il funzionamento del prodotto è garantito per tutti i browser aventi come versioni minime:

- Google Chrome v50 (settembre 2016);
- Mozilla Firefox v50 (novembre 2016);
- Safari v10 per macOS (settembre 2016);
- Opera v41 (ottobre 2016);
- Microsoft Edge v40.15063 (aprile 2017).

Per rendere effettivo il funzionamento su tali browser, deve essere abilitato JavaScript.

4 Ambiente di lavoro

4.1 Installazione prerequisiti

Per poter avviare correttamente l'applicazione RobustDesigner bisogna aver installato le componenti software illustrate nelle seguenti sottosezioni.

4.1.1 Node.js

Per poter installare Node.js si rimanda al seguente link: https://nodejs.org/en/ (consultato il 2018-07-29)

4.1.2 Npm

Con l'installazione di Node.js assicurarsi che sia installato anche Npm digitando da terminale il comando "npm -v" che in caso positivo restituirà la versione presente. In caso contrario bisognerà procedere all'installazione manuale tramite il seguente link: https://www.npmjs.com/get-npm (consultato il 2018-07-29).

4.2 Installazione e avvio dell'applicazione

Per installare e avviare correttamente l'applicazione eseguire i seguenti procedimenti nell'ordine indicato:

1. Scaricare il codice dalla seguente repository: https://github.com/JurassicSWE/RobustDesigner/ (consultato il 2018-09-07).
2. Aprire il terminale all'interno della cartella appena scaricata.
3. Dal terminale appena aperto dare il comando "`npm i`" per installare i package necessari.
4. Dare il comando "`npm start`" per avviare il server.
5. Collegarsi a "`localhost:3000`" da uno qualsiasi dei browser supportati indicati in sezione 3.1.

4.3 Utilizzo del codice generato

4.3.1 Contenuto dello zip

Lo .zip scaricato dall'applicazione contiene i seguenti file:

- **nomeProgetto.json**: è lo stesso .json generato se si sceglie di esportare solo il diagramma, può essere ricaricato nell'editor;
- **nomeProgetto.java**: che contiene il codice di ogni classe associata ad una Entity e i suoi attributi;
- **sqlDatabase.sql**: contiene le istruzioni MySQL per la creazione delle tabelle associate alle Entity;
- **hibernate.cfg.xml**: il file di configurazione di Hibernate, va modificato inserendo i corretti riferimenti al database del proprio sistema (database, username, password);
- **diagram.hbm.xml**: file sempre legato ad Hibernate, contiene la mappatura tra le classi Java e le relative tabelle del database;
- **StoreData.java**: contiene il main dell'applicazione, va editato per aggiungere al suo interno le manipolazioni desiderate alle tabelle del database mediante l'uso di oggetti Java.

Si noti che per utilizzare tutte le funzionalità offerte dal codice generato dall'applicazione è necessario che sul sistema in uso sia installato un server MySQL su cui poter creare un database.

4.3.2 Preparativi

Dato lo zip scaricato dall'applicazione, eseguire questi passaggi iniziali:

1. estrarre lo zip in una cartella nomeProgetto;
2. posizionarsi in nomeProgetto;
3. creare due cartelle al suo interno, per esempio `"bin"` e `"lib"`;

4.3.3 Hibernate e dipendenze

Hibernate per funzionare necessita di due cose: i suoi file e un "connettore" specifico per il database con cui si deve interfacciare.
I file di Hibernate sono reperibili al seguente link:
`http://hibernate.org/orm/releases/5.3/` (consultato il 2018-09-06)
Precisamente sono necessari i file che si trovano all'interno della cartella `"required"`.
Il MySQL connector è disponibile a questo link:
`https://dev.mysql.com/downloads/connector/j/` (consultato il 2018-09-06)
Anche qui precisamente è necessario il .jar che si trova all'interno dello zip scaricato, avrà un nome del tipo `"mysql-connector-java-xyz.jar"`.
I file indicati vanno posizionati all'interno della cartella `"lib"`.

4.3.4 MySQL e il database

Nel proprio MySQL va creato un database, ad esempio "IRONWORKS".
Il seguente è un possibile codice utilizzabile da riga di comando mysql:
"DROP DATABASE IF EXISTS IRONWORKS;"
"CREATE DATABASE IRONWORKS DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;"
"USE IRONWORKS;"
In seguito occorre fornire a mysql il file sqlDatabase.sql per la generazione delle tabelle, sempre da riga di comando è possibile usare la seguente istruzione:
"path-to/nomeProgetto/sqlDatabase.sql;"

4.3.5 I file XML

Va editato il file "hibernate.cfg.xml" impostando i dati d'accesso corretti al MySQL negli appositi campi: connection.url, connection.username, connection.password.
Per quanto riguarda username e password sono quelle di accesso a mysql; connection.url invece è un URI formato nel seguente modo:
"jdbc:mysql://localhost:3306/ironworks"
dove "3306" è la porta su cui è in ascolto il server MySQL, e "ironworks" è il nome del database.
In ultimo copiare (o spostare) i due file .xml ("diagram.hbm.xml" e "hibernate.cfg.xml") nella cartella "bin".

Nota: Nel caso Hibernate lamentasse un problema con l'orario inviato da MySQL, appendere in fondo all'url del database la seguente stringa: "?autoReconnect=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC" in tal modo l'url completo diventa:
"jdbc:mysql://localhost:3306/ironworks?autoReconnect=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC".

4.3.6 Usare e compilare i file Java

Come detto il "main" si trova in "StoreData.java" e di default avvia solo Hibernate eseguendo la mappatura; editare quindi il main per aggiungere tutto ciò che si vuol fare usando le classi generate, al termine della procedura sarà possibile visionare i le modifiche nel database MySQL.
Dopo avere concluso le modifiche al main, aprire un terminale, posizionarsi nella cartella "nomeProgetto" e dare:
javac -cp "lib/*" -d "bin/" *.java
A compilazione in bytecode avvenuta spostarsi con il terminale in "nomeProgetto/bin" e dare:

- Windows

  - per Java 8:
    ```
    java -cp "../lib/*"; "ironworks/StoreData"
    ```
  - per Java 9 o superiore:
    ```
    java -cp "../lib/*"; --add-modules java.xml.bind "ironworks/StoreData"
    ```

- Linux/MacOSX
  - per Java 8:
    ```
    java -cp "../lib/*":  "ironworks/StoreData"
    ```
  - per Java 9 o superiore:
    ```
    java -cp "../lib/*":  --add-modules java.xml.bind "ironworks/StoreData"
    ```

Notare i `:` delle istruzioni per Linux/MacOSX anziché `;` in Windows; il racchiudere tra `"` i path dei file anche se non strettamente necessario è consigliato per evitare problemi nell'individuazione dei path corretti.

5 Architettura

5.1 Architettura Generale

L'architettura di RobustDesigner è basata sul modello client-server, costituito da due moduli distinti ma che collaborano tra loro:

- **Client-side**: editor visuale di diagrammi UML con cui l'utente si può interfacciare.
- **Server-side**: applicazione su server che fornisce al client la pagina iniziale dell'editor, e permette di produrre codice a partire dai diagrammi o da file JSON, basato su uno schema *three-tier*$_g$.

5.2 Client-side

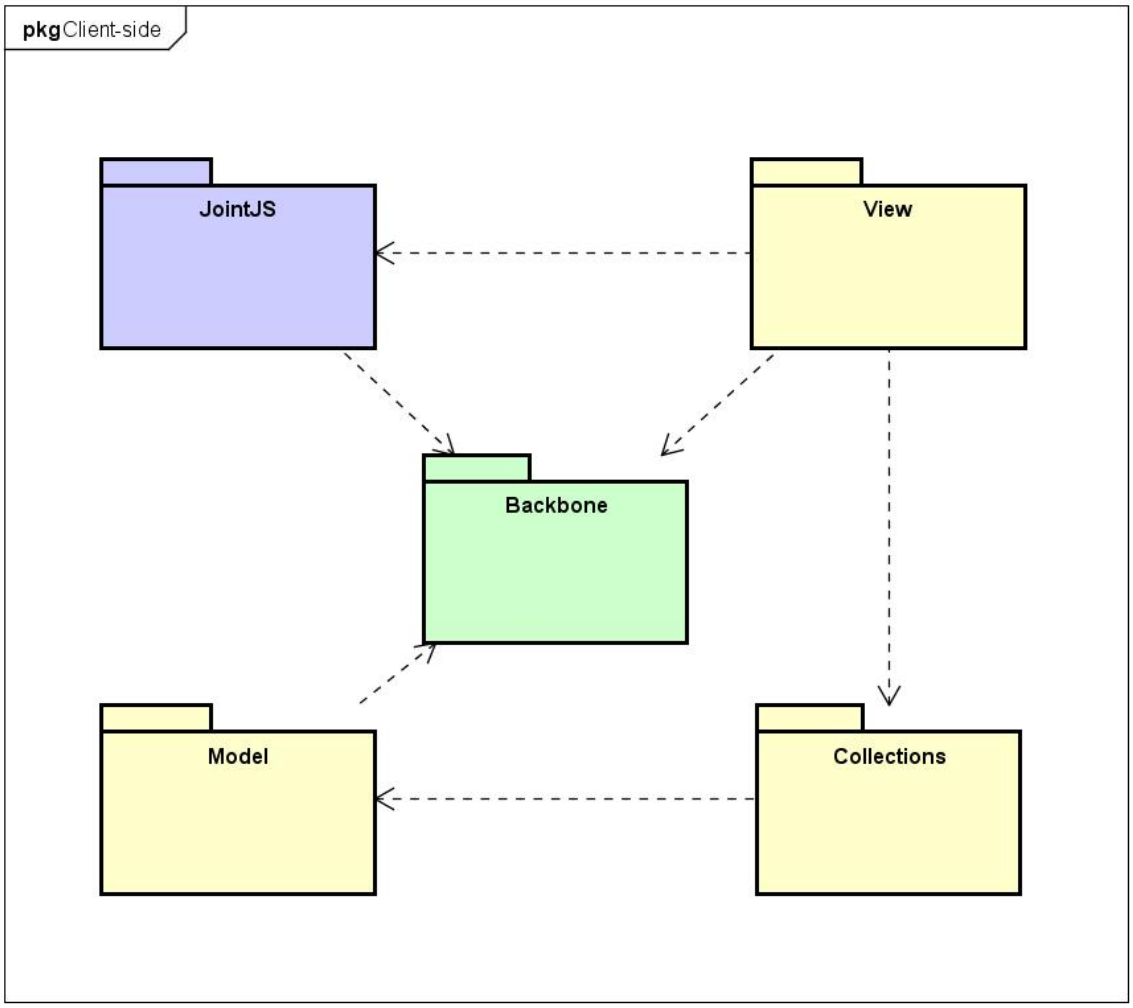

Figura 1: Diagramma dei package lato client

Dal lato client l'applicazione è pensata come a una Web Single-Page (SPA_g), consentendo all'utente di navigare fra le diverse viste rimanendo sulla stessa pagina. Ciò è implementato sfruttando la componente Router di $Backbone_g$.

L'architettura di RobustDesigner prevede l'utilizzo del pattern MV* (Model-View-Whatever), che si basa sulla suddivisione dei compiti fra tre componenti principali:

- **Model**: fornisce i metodi per accedere ai dati utili all'applicazione. Notifica alla view gli aggiornamenti del modello dati in modo che la view mostri sempre dati aggiornati;
- **View**: visualizza i dati contenuti nel model e gestisce la logica di presentazione verso l'utente: cattura l'input e delega al controller l'elaborazione. E' realizzata con HTML e CSS e manipolata con template grazie alla libreria $Undescore_g$.

A livello implementativo l'architettura viene realizzata tramite la libreria Backbone che contiene il componente Router che si occupa di gestire le diverse viste dell'applicazione tramite URI_g, delegando ad ogni View il compito di mostrare la pagina richiesta all'utente. Di conseguenza View e Model si ascoltano a vicenda modificando il proprio stato (Observer pattern).

5.2.1 Client-side::Collections

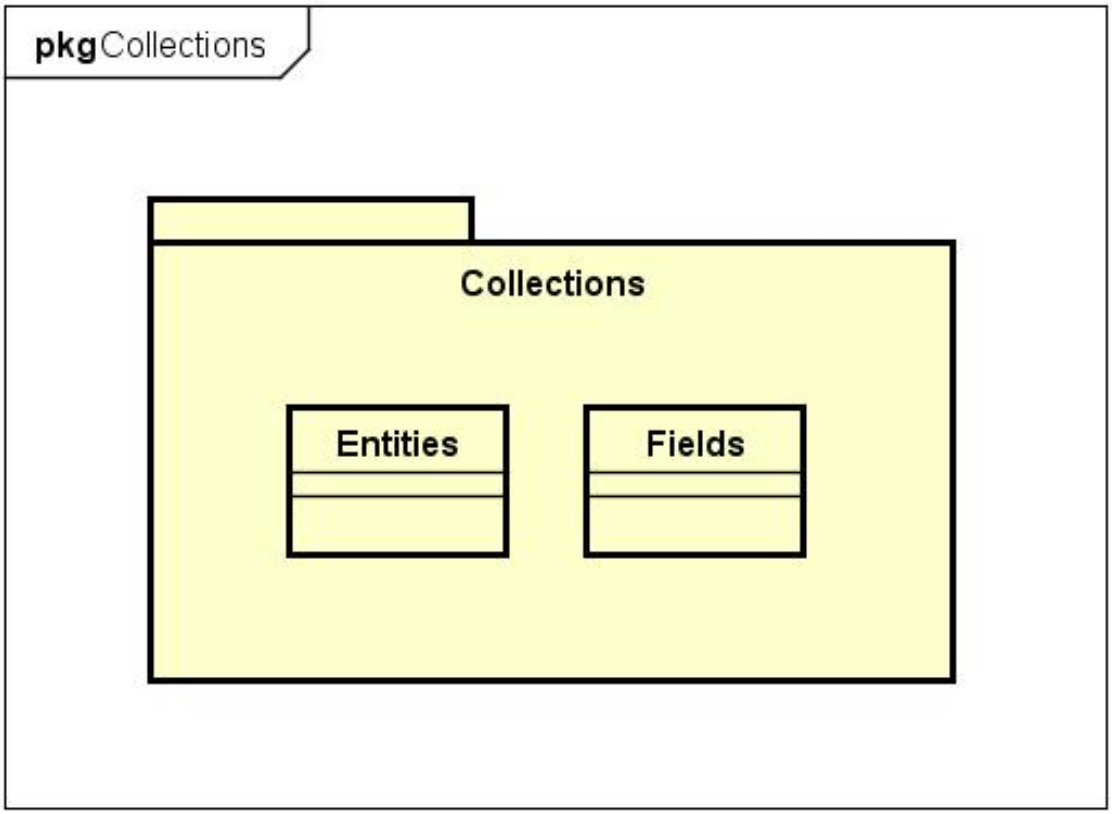

Figura 2: Componente Collections

Il package contiene le collections dei modelli utilizzati.

5.2.1.1 Classi Contenute

- **Entities**: questa classe rappresenta una collezione di modelli Entity;
- **Fields**: questa classe rappresenta una collezione di modelli Field.

5.2.2 Client-side::Model

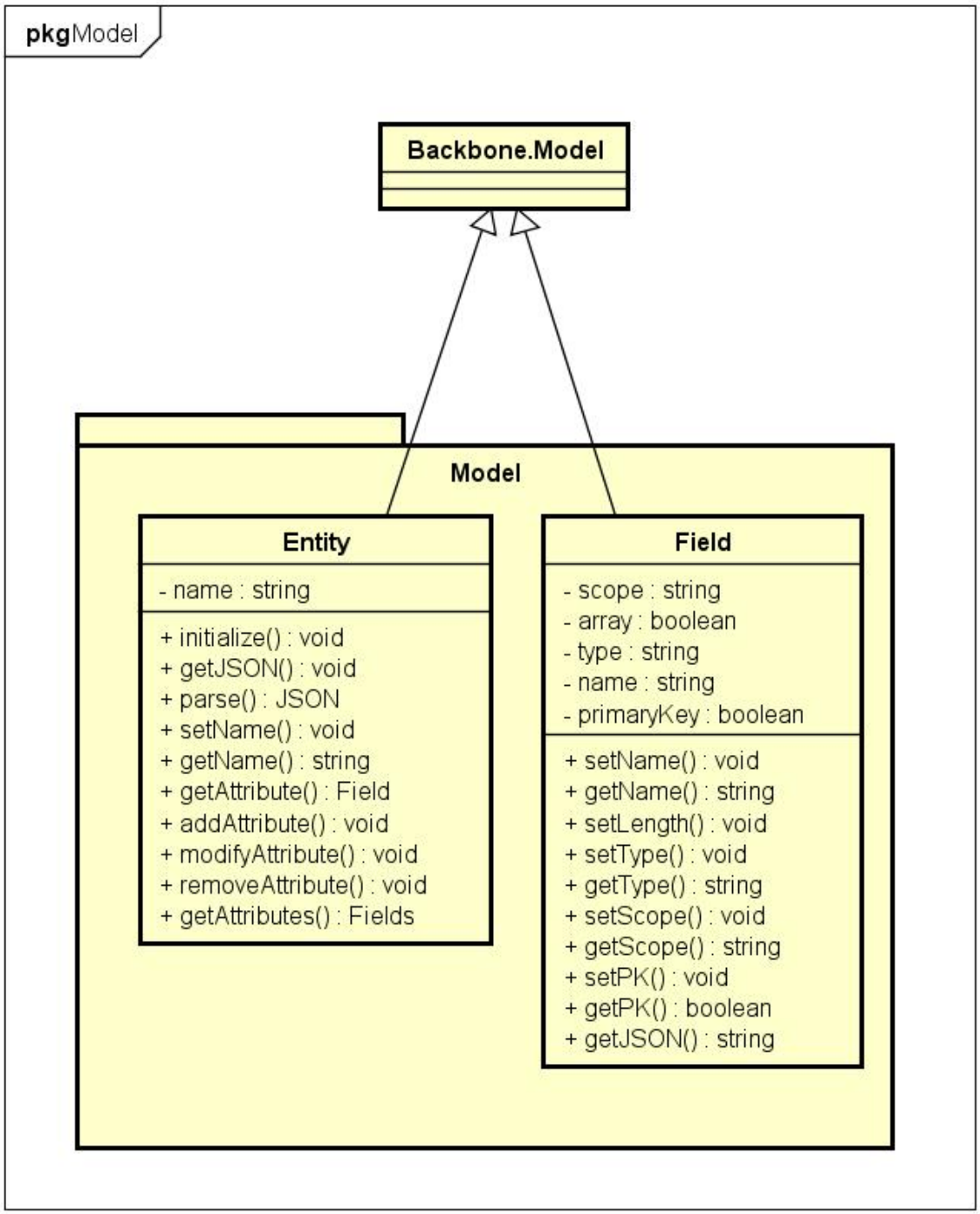

Figura 3: Componente Model

Il package contiene i modelli utilizzati dalla Client-side::View.

5.2.2.1 Classi Contenute

- **Entity**: questa classe rappresenta una Entity del diagramma, con gli attributi caratteristici;
- **Field**: questa classe rappresenta un attributo di una entity, con i parametri che lo caratterizzano.

Queste due lassi derivano dalla classe **Backbone.Model** che si occupa di fornire un contratto comune per i modelli istanziati.

5.2.3 Client-side::View

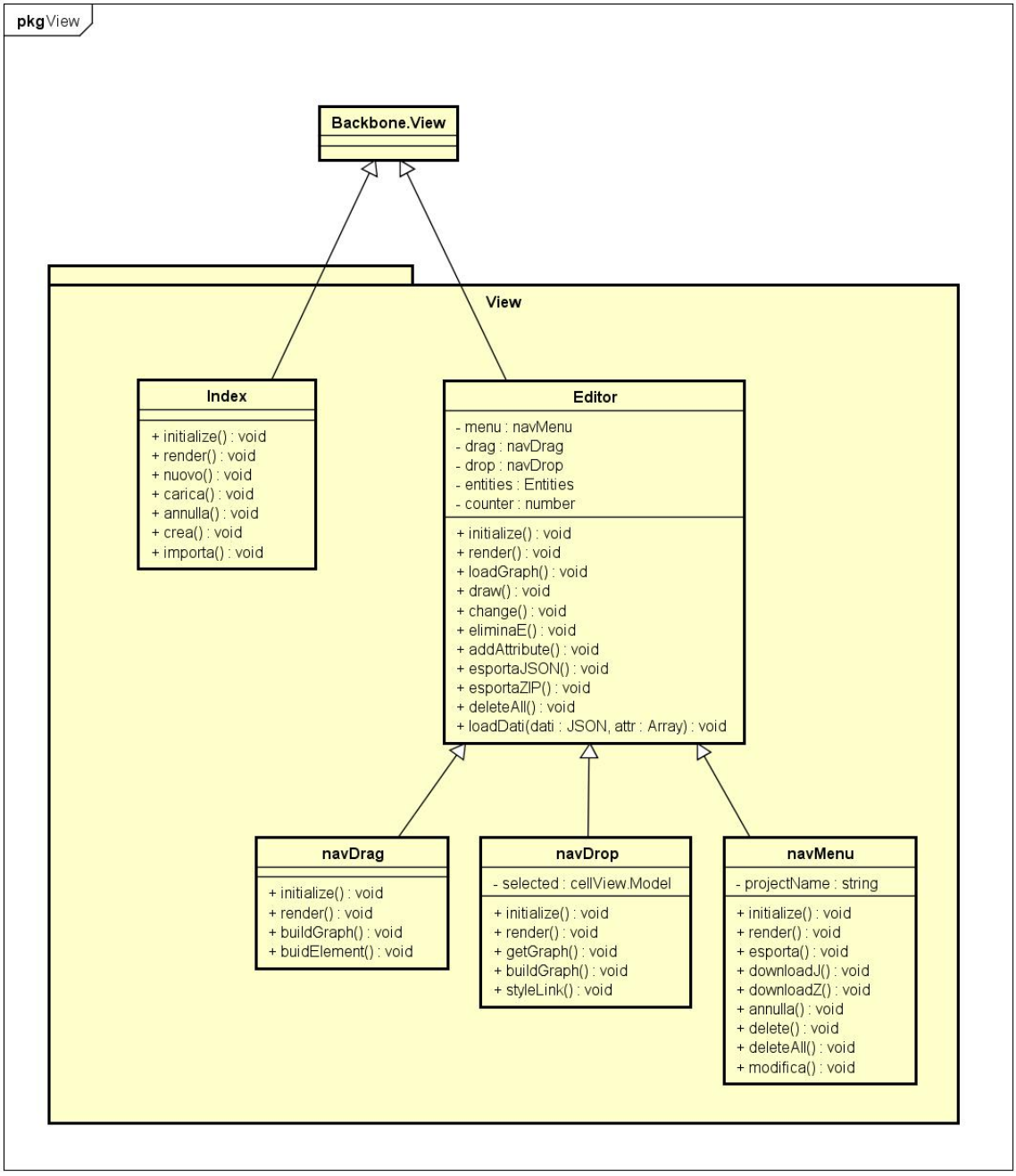

Figura 4: Componente View

Il package raccoglie le classi che si occupano di visualizzare l'editor e la homepage. Inoltre, questo package raccoglie le classi relative alla palette e gli elementi della toolbar che si occupano di gestire ulteriori dettagli relativi a specifici elementi dell'editor.

5.2.3.1 Classi Contenute

- **Editor**: questa classe si occupa di visualizzare la pagina dell'editor per la costruzione di diagrammi di robustezza, con i relativi pulsanti per l'import/export in JSON. È ereditata dalla classe **Backbone.View** di Backbone;
- **Index**: questa classe gestisce la schermata iniziale dell'applicazione. È ereditata dalla classe **Backbone.View** di Backbone;
- **navDrag**: questa classe gestisce la vista degli elementi del diagramma di robustezza. Eredita dalla classe Editor.
- **navDrop**: questa classe gestisce la vista relativa al diagramma di robustezza, quindi visualizza la griglia dove l'utente può aggiungere, spostare, modificare elementi. Eredita dalla classe Editor.
- **navMenu**: questa classe gestisce la vista del menu all'utente, fornendo opzioni per l'import ed export di un diagramma, la rimozione di parte o tutti gli elementi all'interno della griglia. Eredita dalla classe Editor.

5.3 Server-side

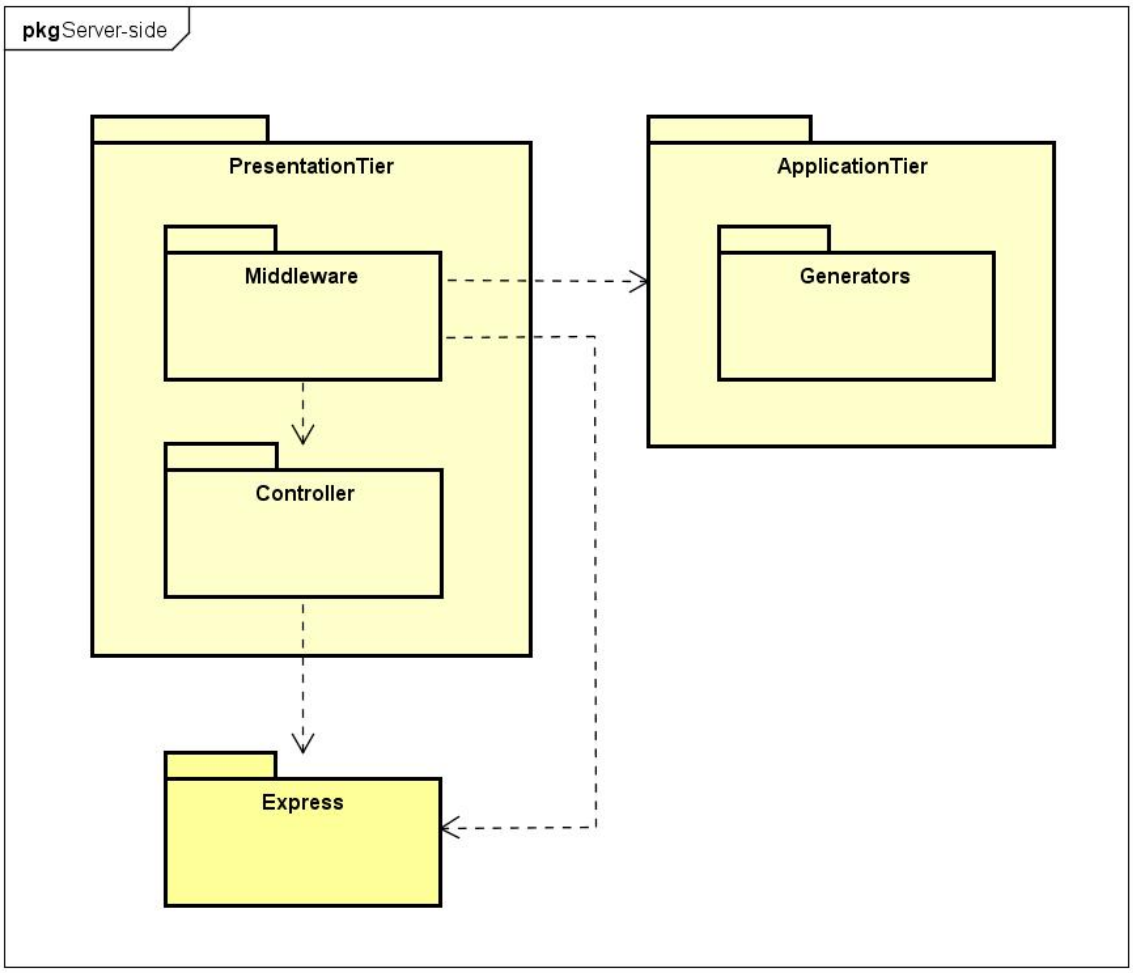

Figura 5: Diagramma dei package lato server

L'architettura dal lato server prevede uno schema *three-tier*$_g$ a livelli, così organizzato:

- **PresentationTier**: interfaccia di comunicazione verso il Client, si occupa di ricevere richieste o inviare risorse all'utente utilizzando il framework *Express*$_g$;
- **ApplicationTier**: strato della logica operativa che contiene i metodi per la generazione di codice, la gestione degli errori e la funzionalità di compressione file;
- **DataTier**: strato dedicato all'organizzazione dei dati su un database. Non essendo richiesta la persistenza dati, non è stato implementato.

L'architettura che ne consegue assomiglia quindi a un *two-tier*, con l'attenzione riservata ai soli livelli di presentazione e applicazione. Tutti questi strati utilizzano interfacce ben definite conferendo flessibilità e manutenibilità all'applicazione.

5.3.1 Server-side::ApplicationTier

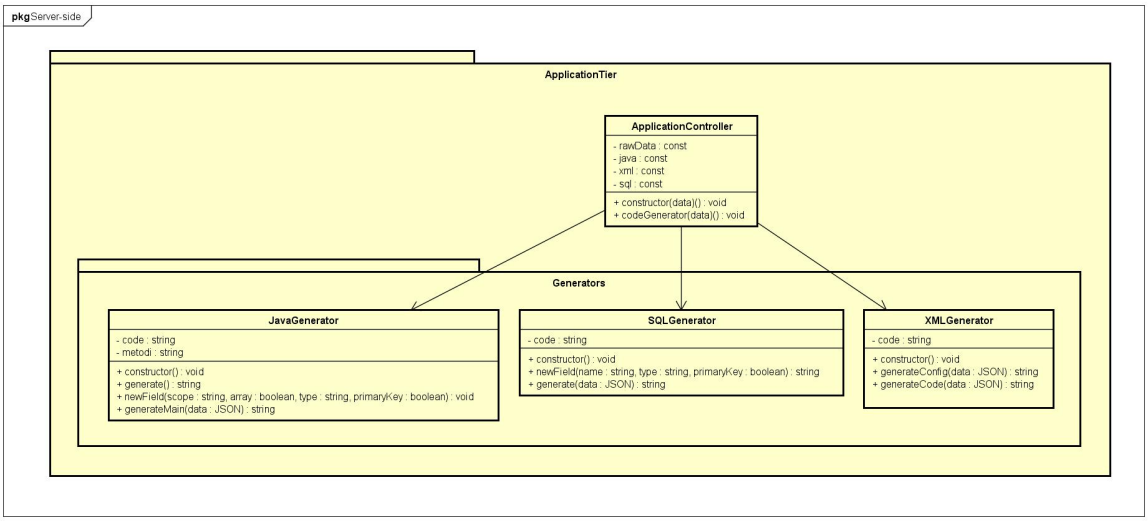

Figura 6: Diagramma dei package livello Application

Il package contiene tutte le componenti per il funzionamento dell'applicazione.

5.3.1.1 Classi Contenute

- **ApplicationController**: questa classe gestisce le comunicazioni tra il livelli PresentationTier e ApplicationTier.

5.3.2 Server-side::ApplicationTier::Generators

Il package si occupa di gestire la generazione di codice a partire da diagrammi.

5.3.2.1 Classi Contenute

- **JavaGenerator**: questa classe contiene i metodi per la generazione delle classi Java a partire dai dati delle entity;
- **SQLGenerator**: questa classe contiene i metodi per la generazione delle tabelle SQL a partire dai dati delle entity.
- **XMLGenerator**: questa classe contiene i metodi per la generazione di file di configurazione in XML a partire dalle entity da utilizzare con il framework *Hibernate*$_g$.

5.3.3 Server-side::PresentationTier

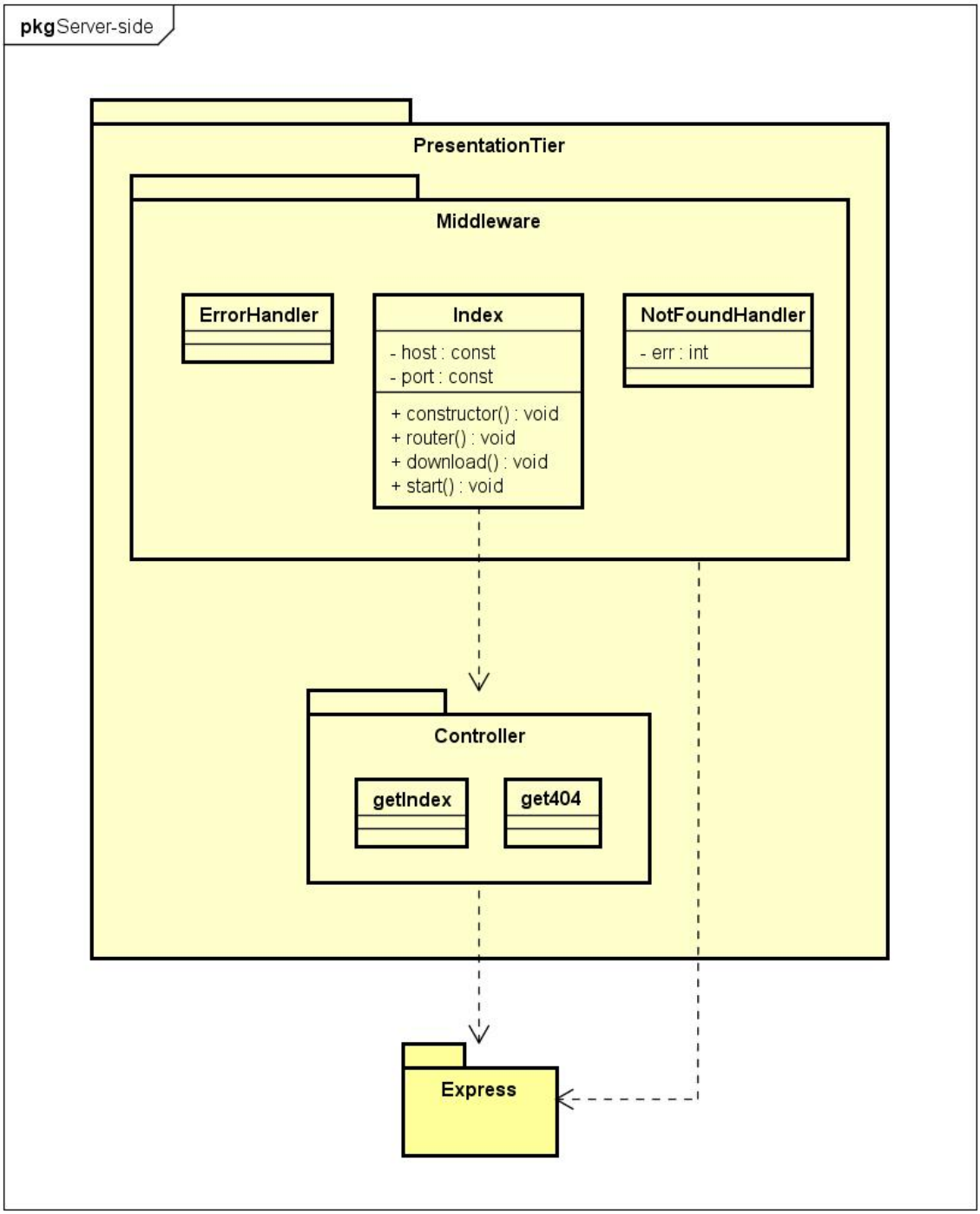

Figura 7: Diagramma dei package livello Presentation

Il package contiene tutte le classi che gestiscono le richieste da parte del client. Costituisce la parte Presentation dell'architettura Three-tier del back-end.

5.3.3.1 Package Contenuti

- **Server-side::PresentationTier::Controller**: il package contiene le classi che gestiscono la richiesta delle pagine index e di errore 404;
- **Server-side::PresentationTier::Middleware**: il package gestisce le comunicazioni tra client e server e gestisce i possibili errori quando vengono richieste risorse.

5.3.4 Server-side::PresentationTier::Controller

Il package contiene la classe che gestisce la richiesta delle pagine index e di errore 404.

5.3.4.1 Classi Contenute

- **get404**: questa classe fornisce al client la pagina di errore 404;
- **getIndex**: questa classe gestisce la richiesta della pagina index da parte del client.

5.3.5 Server-side::PresentationTier::Middleware

Il package gestisce le comunicazioni tra client e server e gestisce i possibili errori quando vengono richieste risorse.

5.3.5.1 Classi Contenute

- **ErrorHandler**: questa classe gestisce eventuali errori nei precedenti middleware inviando una risposta al client con una descrizione dell'errore in formato JSON. Fa parte dei componenti ConcreteHandler del Design Pattern Chain of Responsibility;
- **Index**: questa classe gestisce le richieste di risorse provenienti dal client;
- **NotFoundHandler**: questa classe gestisce l'errore 404 di pagina non trovata. Fa parte dei componenti ConcreteHandler del Design Pattern Chain of Responsibility.

6 Estensione

6.1 Generare in altri linguaggi

L'applicazione così pensata permette la generazione di codice Java e codice SQL_g che creano classi pensate per ospitare i dati delle entità persistenti ed i metodi per leggere e scrivere questi dati. Tuttavia si potrebbe estendere questo concetto ad altri tipi di linguaggi, con la giusta conoscenza di quest'ultimi e l'aggiunta di un nuovo package all'interno di Server-side::ApplicationTier::Generators. Una simile estensione è già stata fatta, in quanto l'applicazione web allo stato attuale genera, a parte il file Java, un file XML_g che tramite l'ORM_g Hibernate si può trasformare in un file SQL.

6.2 Continuità tra diagrammi

Un'ulteriore aggiunta può essere fatta aumentando la continuità tra i diagrammi di robustezza, delle classi e dei casi d'uso. Ciò può essere fatto aggiungendo per esempio delle schede nella web application in cui è possibile scegliere se visualizzare la stessa rappresentazione del diagramma con più viste, nello specifico: casi d'uso, classi e diagrammi di robustezza. Facendo così svolgere al diagramma di robustezza il ruolo originariamente pensato da Rosenberg e Stephen quando hanno introdotto l'analisi di robustezza come mezzo per colmare il divario tra analisi (il cosa) e design (il come).

6.3 Estendere le possibilità di login

Aggiungere una login tramite GitHub o altre piattaforme aiuterebbe da un lato ad accelerare le tempistiche di login, effettuando una semplice connessione tra un account GitHub o altri, dall'altro permetterebbe una ancor più semplice continuità del lavoro da altre postazioni, salvando il progetto sul proprio repository.

A Glossario

A

Applicazione web un'applicazione accessibile e fruibile via web per mezzo di un network che offre determinati servizi all'utente.

B

Backbone libreria javascript leggera di tipo MVC (Model View Controller) progettata per lo sviluppo di applicazioni web di tipo single-page.

C

Client indica le operazioni di elaborazione effettuate da un client in un'architettura client-server. Rappresenta dunque il front-end di un sistema informatico e di un'applicazione web con architettura multi-tier.

CSS3 acronimo di "Cascading Style Sheet", è un linguaggio utilizzato per definire il formato dei layout delle pagine web.

D

Database relazionale è un modello logico di rappresentazione o strutturazione dei dati di un database implementato su sistemi di gestione di basi di dati (DBMS), detti perciò sistemi di gestione di basi di dati relazionali (RDBMS).

E

Event-driven è un paradigma di programmazione in cui il flusso del programma è determinato dal verificarsi di eventi esterni.

Express un popolare framework per realizzare applicazioni web in Node.js.

F

Framework indica un'architettura logica di supporto utilizzata all'interno di un software per semplificare determinate operazioni ed evitare la riscrittura di codice. Un framework è composto da un insieme di classi astratte e le relazioni tra di esse. Alla base di un framework c'è sempre una libreria.

H

Hibernate è una piattaforma open source per lo sviluppo di applicazioni Java, attraverso l'appoggio al relativo framework, che fornisce un servizio di Object-Relational Mapping (ORM) ovvero gestisce la persistenza dei dati sul database attraverso la rappresentazione e il mantenimento su database relazionale di un sistema di oggetti Java. Nell'ambito dello sviluppo di applicazioni web tale strato software si frappone tra il livello logico di business o di elaborazione e quello di persistenza dei dati sul database.

HTML5 è un linguaggio di markup per la strutturazione delle pagine web.

J

Java è un linguaggio di programmazione ad alto livello orientato agli oggetti, specificatamente progettato per essere il più possibile indipendente dalla piattaforma di esecuzione.

Javascript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script.

JSON acronimo di JavaScript Object Notation, è un formato adatto all'interscambio tra applicazioni client/server.

M

MVC pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business.

MV* pattern architetturale che separa il modello e il comportamento dalla view (visualizzazione).

N

Npm principale software utilizzato per maneggiare i moduli di NodeJs.

O

Open-source indica un software non protetto da copyright e liberamente modificabile dall'utente.

ORM acronimo di "Object-Relation Mapping", è una tecnica di programmazione per convertire dati tra sistemi incompatibili usando linguaggi di programmazione orientati agli oggetti. Questo crea un database di oggetti virtuali che può essere usato dai linguaggi di programmazione.

P

Package è un meccanismo per organizzare classi Java in gruppi logici, principalmente allo scopo di definire namespace distinti per diversi contesti. Il package ha lo scopo di riunire classi logicamente correlate. Un esempio viene dalle librerie standard Java, le quali sono organizzate in un sistema di package che comprende per esempio elementi strutturali del linguaggio, servizi di rete e così via.

Prodotto bene o servizio che possa essere offerto per soddisfare un bisogno o un'esigenza.

R

Responsive web design indica una tecnica di web design per la realizzazione di siti in grado di adattarsi graficamente in modo automatico al dispositivo coi quali vengono visualizzati.

Routing Instradamento effettuato a livello di rete, le richieste vengono dirottate in base alle indicazioni fornite.

S

Server in informatica un server è un componente di elaborazione e gestione del traffico di informazioni che fornisce, a livello logico e fisico, un qualunque tipo di servizio ad altre componenti che ne fanno richiesta attraverso una rete.

SPA con Single-page application o in sigla SPA si intende un'applicazione web o un sito web che può essere usato o consultato su una singola pagina web con l'obiettivo di fornire una esperienza utente più fluida e simile alle applicazioni desktop dei sistemi operativi tradizionali.

SQL acronimo di "Structured Query Language", è un linguaggio di tipo non procedurale che consente di operare sui dati di un database tramite frasi che non fanno uso di simboli matematici ed algebrici, ma solo di parole chiave prese dal linguaggio corrente. Questo, insieme alla sintassi che riprende forme usuali, fa sì che le frasi abbiano una struttura lineare ed espressiva.

SVG linguaggio per la descrizione di immagini 2D composte da forme in grafica vettoriali, immagini bitmap e testo.

T

Three-tier indica una particolare architettura software di tipo multi-tier che prevede la suddivisione dell'applicazione in tre diversi strati: uno dedicato all'interfaccia utente, uno alla logica funzionale (business logic) e uno alla gestione dei dati persistenti.

Two-way data binding indica che ogni modifica al modello dei dati si riflette automaticamente sulla view e ogni modifica alla view viene riportata sul modello dei dati.

U

UML acronimo di "Unified Modeling Language", sviluppato da Intentional Software, fornisce aiuto e supporto nello sviluppo del software e delle applicazioni.

Underscore libreria JavaScript che aggiunge una serie di funzioni e metodi di utilità agli oggetti built-in di JavaScript.

URI sequenza di caratteri che identifica univocamente una risorsa generica. Sono esempi di URI: un indirizzo web (URL), un documento, un'immagine, un file, un servizio, un indirizzo di posta elettronica, ecc.

X

XML acronimo di "Extensible Markup Language", è un meta-linguaggio per la definizioni di linguaggi di markup, ovvero un linguaggio basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.