



Manuale sviluppatore v1.0.0

WarMachine – Progetto IronWorks

warmachine.swe@gmail.com

Informazioni sul documento:

Versione	1.0.0
Data di creazione	27/06/2018
Redazione	Bernucci Riccardo, Bragagnolo Leonardo, Cisternino Nicola
Verifica	Coletti Andrea, Fogarollo Stefano
Approvazione	Zanetti Ilenia
Uso	Esterno
Distribuzione	<i>Prof. Vardanega Tullio</i> <i>Prof. Cardin Riccardo</i> <i>Zucchetti s.p.a</i>



Diario delle modifiche

Versione	Data	Collaboratori	Ruolo	Descrizione
1.0.0	12/07/2018	Zanetti Ilenia	<i>Responsabile di progetto</i>	Approvazione documento.
0.4.0	10/07/2018	Fogarollo Stefano	<i>Verificatore</i>	Verificato intero documento.
0.3.0	09/07/2018	Coletti Andrea	<i>Verificatore</i>	Verificate §[7] e §[6].
0.2.2	06/07/2018	Cisternino Nicola	<i>Programmatore</i>	Scritta §[7].
0.2.1	04/07/2018	Bragagnolo Leonardo	<i>Programmatore</i>	Scritta §[6].
0.2.0	04/07/2018	Fogarollo Stefano	<i>Verificatore</i>	Verificate §[5] e §[8].
0.1.2	03/07/2018	Bragagnolo Leonardo	<i>Programmatore</i>	Scritta §[5].
0.1.1	02/07/2018	Cisternino Nicola	<i>Programmatore</i>	Corretta §[2]. Scritta §[8].
0.1.0	29/06/2018	Fogarollo Stefano	<i>Verificatore</i>	Verificate §[1], §[2], §[3] e §[4]. Da rivedere §[2].
0.0.3	29/06/2018	Bernucci Riccardo	<i>Programmatore</i>	Scritte §[3] e §[4].
0.0.2	28/06/2018	Bragagnolo Leonardo	<i>Programmatore</i>	Scritte §[1] e §[2].
0.0.1	27/06/2018	Zanetti Ilenia	<i>Programmatore</i>	Creazione documento. Scritto scheletro del documento.



Indice

1	Introduzione	5
1.1	Scopo del documento	5
1.2	Scopo del prodotto	5
1.3	Informazioni utili	5
2	Requisiti	6
2.1	Requisiti hardware	6
2.2	Dispositivi supportati	6
2.3	Browser supportati	6
3	Tecnologie utilizzate	7
3.1	HTML5	7
3.2	CSS3	7
3.3	JavaScript	7
3.4	Node.js	7
3.5	Expressjs	7
3.6	Jade	7
3.7	JointJS	8
3.8	Backbone	8
3.9	MySQL	8
4	Configurazione ambiente	9
4.1	Installazione Node.js	9
4.2	Configurazione dell'applicazione IronWorks	9
4.3	Avvio dell'applicazione	9
5	Architettura	10
5.1	Client	10
5.1.1	Package contenuti	10
5.1.2	Dettaglio classi	11
5.2	Server	12
5.2.1	Package contenuti	12
5.2.2	Dettaglio classi	13
6	Estensione delle funzionalità	16
6.1	Aggiungere nuovi oggetti al diagramma	16
6.2	Aggiungere nuovi metodi Java	16
7	Segnalazione degli errori	17
8	Come contribuire	20
8.1	Come creare una fork	20
8.2	Come creare una pull request	20



Glossario	24
A	24
<i>Applicazione web</i>	24
<i>API</i>	24
D	24
<i>DBMS</i>	24
<i>Diagramma di robustezza</i>	24
E	24
<i>Entity persistenti</i>	24
I	24
<i>Issue</i>	24
J	25
<i>Java</i>	25
L	25
<i>Linguaggio di markup</i>	25
O	25
<i>Open source</i>	25
P	25
<i>Pedice</i>	25
S	25
<i>SQL</i>	25
W	26
<i>World Wide Web</i>	26

Elenco delle figure

1	Diagramma generale dei package lato client.	10
2	Dettaglio model.	11
3	Diagramma generale dei package lato server.	12
4	Dettaglio PresentationTier.	13
5	Dettaglio ApplicationTier.	14
6	Repository Gith-Hub.	17
7	Repository sezione Issues.	17
8	Creare un issue.	18
9	Inserimento titolo issue.	18
10	Issue template.	19
11	Repository creazione Fork.	20
12	Repository creazione Pull Request.	20
13	Creare una pull request - creare una pull request.	21
14	Creare una pull request - compare across forks option.	21
15	Creare una pull request - scelta del branch su cui fare merge.	21
16	Creare una pull request - scelta del branch modificato.	22
17	Creare una pull request - titolo.	22
18	Creare una pull request - descrizione.	22
19	Creare una pull request - creazione pull request.	23



1 Introduzione

1.1 Scopo del documento

Il documento presenta le modalità di installazione e utilizzo per l'applicazione IronWorks sviluppata dal gruppo *WarMachine*. Il documento illustra i requisiti per l'utilizzo, le librerie e i framework esterni adottati per lo sviluppo dell'applicazione. Inoltre presenta anche l'architettura del prodotto così da poter aiutare gli sviluppatori a mantenere ed estendere il prodotto.

1.2 Scopo del prodotto

Lo scopo del prodotto è la creazione di un'applicazione web_G open-source_G per la costruzione di diagrammi di robustezza_G con la relativa generazione di codice Java_G e SQL_G per le entity persistenti_G. Il sistema_G dovrà fornire le seguenti funzionalità:

- Creazione di diagrammi di robustezza mediante editor_G;
- Modifica di diagrammi di robustezza mediante editor;
- Generazione codice Java;
- Generazione codice SQL per la creazione di tabelle per ospitare i dati delle entity.

1.3 Informazioni utili

Viene assunto che l'utente finale dell'applicazione sia uno sviluppatore dotato di una buona conoscenza dei diagrammi di robustezza. Per evitare ogni ambiguità sono stati inseriti i termini di ambito tecnico, o i termini che necessitano di ulteriori spiegazioni, indicati con la notazione a pedice_G, in un glossario in appendice.



2 Requisiti

2.1 Requisiti hardware

Il prodotto richiede i seguenti requisiti minimi:

- Processore dual core;
- 2GB di memoria RAM;
- 1GB di spazio libero su disco.

2.2 Dispositivi supportati

L'applicazione IronWorks è compatibile con i seguenti sistemi operativi desktop:

- Ubuntu 16.04 LTS 64 bit;
- Microsoft Windows 10;

2.3 Browser supportati

Di seguito sono elencate le versioni minime dei browser supportati dal software IronWorks:

- Google Chrome 65.0.3325;
- Mozilla Firefox 59.0.2.



3 Tecnologie utilizzate

3.1 HTML5

E' un linguaggio di markup_G usato per strutturare e presentare contenuti sul World Wide Web_G. HTML5 è supportato su tutti i browser moderni, inoltre introduce delle API_G per applicazioni web complesse. Maggiori informazioni sono reperibili al seguente indirizzo:

<https://www.w3.org/TR/html52/>

3.2 CSS3

E' un linguaggio che permette di definire lo stile con cui vengono visualizzati gli elementi HTML di un documento. Maggiori informazioni sono reperibili al seguente indirizzo:

<https://www.w3.org/TR/css-ui-3/>

3.3 JavaScript

E' un linguaggio di programmazione interpretato e dinamico, con debole tipizzazione, multi paradigma e basato su prototipi. Maggiori informazioni sono reperibili al seguente indirizzo:

<http://devdocs.io/javascript/>

3.4 Node.js

E' un ambiente runtime JavaScript, event-driven asincrono. Maggiori informazioni sono reperibili al seguente indirizzo:

<https://nodejs.org/it/>

3.5 Expressjs

E' un framework di applicazioni Web per Node.js, rilasciato come software libero e open-source con licenza MIT. È progettato per la creazione di applicazioni Web e API. Maggiori informazioni sono reperibili al seguente indirizzo:

<https://expressjs.com/>

3.6 Jade

E' un template engine, ovvero un metodo per separare la struttura HTML dal contenuto della pagina, implementato in JavaScript. Maggiori informazioni sono reperibili al seguente indirizzo:

<http://jade-lang.com/>



3.7 JointJS

E' una libreria opensource per la creazione di diagrammi statici o strumenti di diagrammi completamente interattivi come editor di flusso di lavoro, strumenti di gestione dei processi, sistemi IVR, integratori API, applicazioni di presentazione. Maggiori informazioni sono reperibili al seguente indirizzo:

<https://www.jointjs.com/>

3.8 Backbone

E' una libreria JavaScript con un'interfaccia JSON RESTful. È progettato per lo sviluppo di applicazioni Web a pagina singola e per mantenere sincronizzate varie parti di applicazioni Web. Maggiori informazioni sono reperibili al seguente indirizzo:

<http://backbonejs.org/>

3.9 MySQL

Il codice SQL generato è compatibile con il DBMS_G relazionale MySQL. Maggiori informazioni sono reperibili al seguente indirizzo:

<https://www.mysql.com/>

4 Configurazione ambiente

Di seguito sono illustrate le modalità di installazione e configurazione del software IronWorks.

4.1 Installazione Node.js

Per utilizzare il software IronWorks è necessario installare node.js seguendo le istruzioni disponibili al seguente indirizzo:

<https://nodejs.org/it/download/package-manager/>

4.2 Configurazione dell'applicazione IronWorks

Per prima cosa è necessario clonare la seguente repository:

<https://github.com/WarMachineSwe/IronWorks>

Quindi si apra un terminale e si digiti il comando:

```
$ git clone https://github.com/WarMachineSwe/IronWorks
```

Per installare i moduli Node.js necessari al funzionamento del software bisogna:

1. Spostarsi all'interno della cartella IronWorks della repository appena scaricata;
2. Da terminale dare il seguente comando:

```
$ npm install
```

4.3 Avvio dell'applicazione

Dopo aver seguito i passi per l'installazione sopracitati, eseguire da terminale il seguente comando: `$ npm start` A questo punto il software è avviabile secondo i seguenti passi:

1. Aprire un browser a scelta;
2. Digitare sulla barra degli indirizzi "localhost:3000" o "http://127.0.0.1:3000/" senza virgolette.

5 Architettura

La presentazione dell'architettura avviene con un approccio top-down ovvero partendo dal generale e passando al particolare.

5.1 Client

L'architettura lato client segue il design pattern Model-View-Controller.

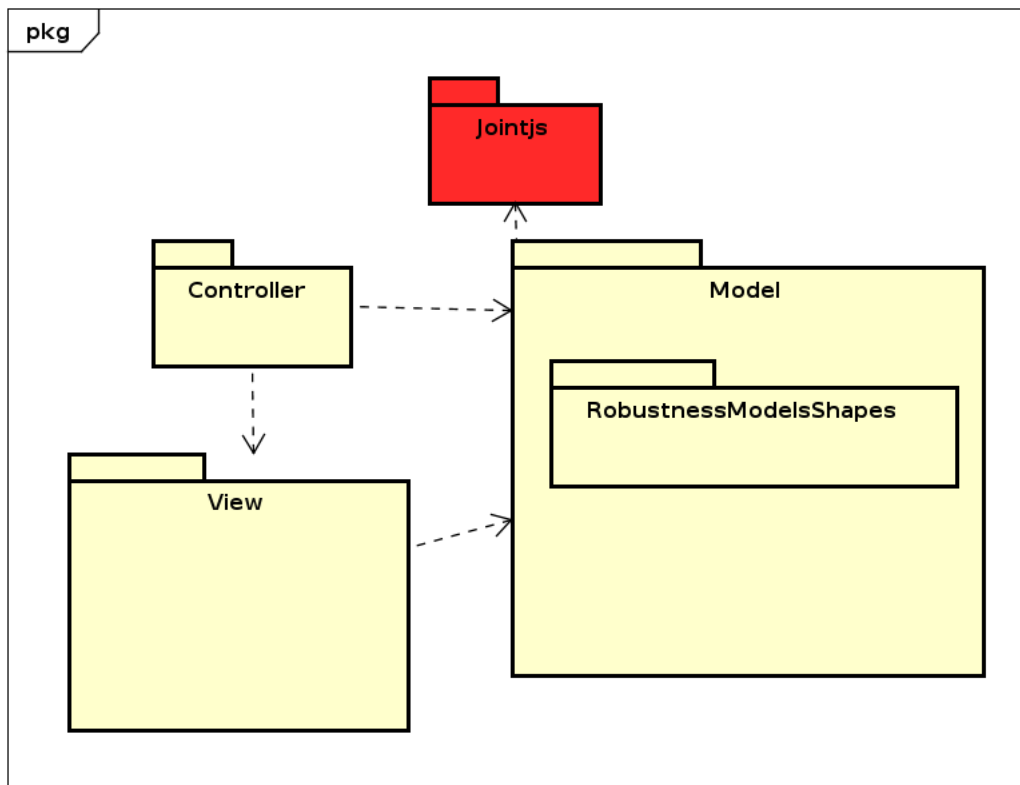


Figura 1: Diagramma generale dei package lato client.

5.1.1 Package contenuti

Il client contiene i seguenti package:

- **Model:** Contiene i modelli per generare il diagramma di robustezza.
Contiene i seguenti package:
 - **RobustnessModelsShapes:** Contiene le classi definite per creare gli elementi del diagramma.
- **View:** Contiene le componenti relative alla homepage e l'editor del diagramma di robustezza;
- **Controller:** Gestisce l'interazione e la comunicazione tra la View e il Model;

5.1.2 Dettaglio classi

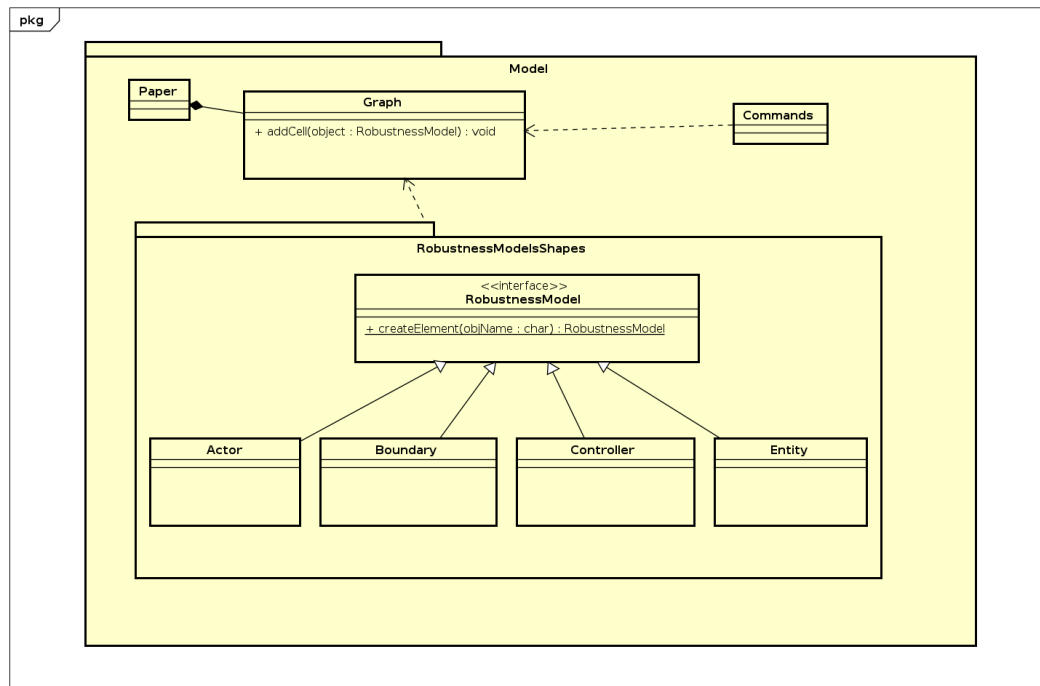


Figura 2: Dettaglio model.

Il package *Model* contiene le seguenti classi:

- **Graph:** Contiene gli elementi che vengono aggiunti nella canvas;
- **Commands:** Contiene le operazioni permesse dal progetto sugli elementi;
- **Paper:** Crea la canvas dove vengono contenuti gli elementi del diagramma.

Il package *Model::RobustnessModelsShapes* contiene le seguenti classi:

- **RobustnessModel:** Classe base astratta che crea gli elementi del diagramma;
- **Actor:** Classe derivata da RobustnessModel che rappresenta un oggetto Actor;
- **Entity:** Classe derivata da RobustnessModel che rappresenta un oggetto Entity;
- **Boundary:** Classe derivata da RobustnessModel che rappresenta un oggetto Boundary;
- **Controller:** Classe derivata da RobustnessModel che rappresenta un oggetto Controller;

Il package *View* contiene le seguenti classi:

- **ElementClick:** Gestisce la visualizzazione degli elementi della canvas;
- **EditorLoad:** Gestisce la visualizzazione dell'area della canvas;

- **OperationClick**: Gestisce gli input dell'utente ricevendo gli eventi JavaScript e mostrando i risultati sulla View.

Il package *Controller* contiene le seguenti classi:

- **ElementHandler**: Gestisce la creazione degli elementi;
- **EditorHandler**: Gestisce gli errori nella creazione della canvas, delle etichette e degli strumenti dei collegamenti;
- **OperationHandler**: Gestisce i messaggi d'errore o di successo della canvas e della descrizione delle entity.

5.2 Server

L'architettura lato server segue il pattern architetturale two-tier.

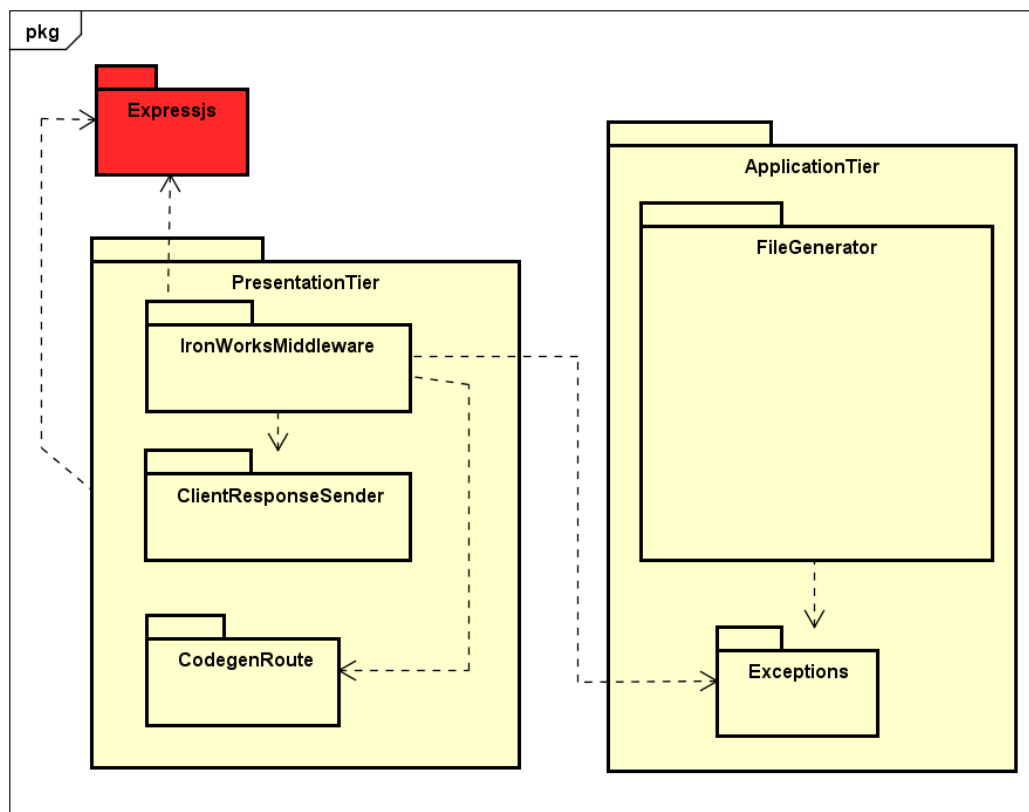


Figura 3: Diagramma generale dei package lato server.

5.2.1 Package contenuti

Il client contiene i seguenti package:

- **PresentationTier**: Contiene le classi che gestiscono le richieste del client. Contiene i seguenti package:

- **IronWorksMiddleware:** Contiene le classi che gestiscono la comunicazione tra client e server;
 - **CodegenRoute:** Contiene le classi che gestiscono le richieste di generazione del codice;
 - **ClientResponseSender:** Contiene le classi che gestiscono le richieste relative alla pagina iniziale dell'applicazione.
- **ApplicationTier:** Contiene le componenti relative alla business logic dell'applicazione.
Contiene i seguenti package:

- **FileGenerator:** Contiene le componenti relative alla creazione dei files Java e SQL.
Contiene i seguenti package:
 - * **CodeGenerator:** Contiene le classi che gestiscono la generazione del codice;
 - * **JavaGenerator:** Contiene le classi che gestiscono la generazione del codice Java;
 - * **SqlGenerator:** Contiene le classi che gestiscono la generazione del codice SQL.
- **Exceptions:** Contiene le classi che gestiscono gli errori relativi alla generazione del codice.

5.2.2 Dettaglio classi

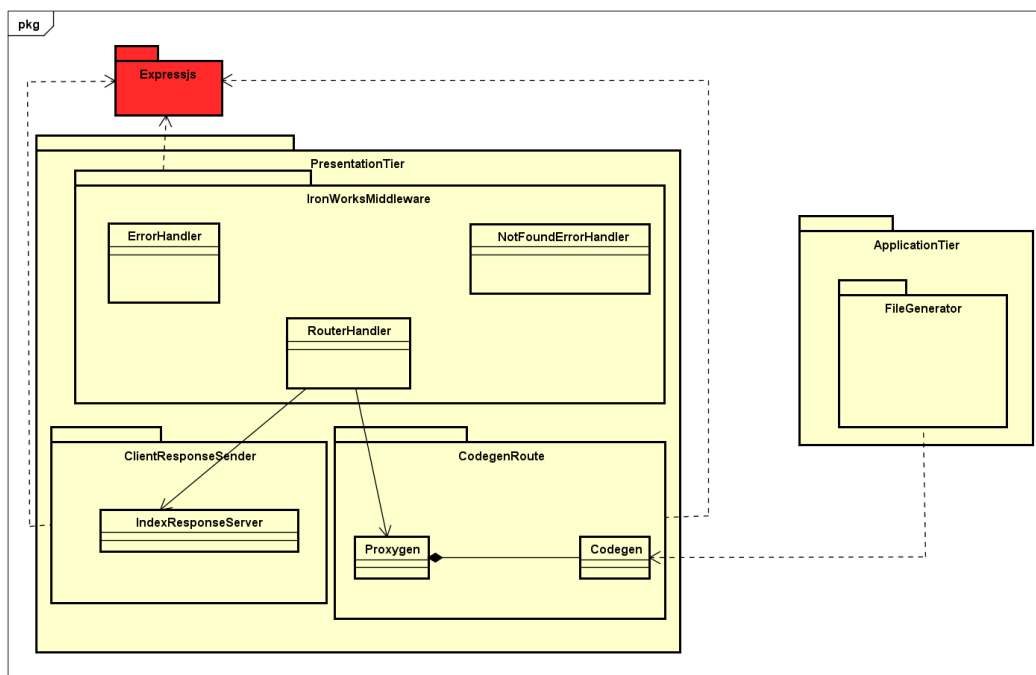


Figura 4: Dettaglio PresentationTier.

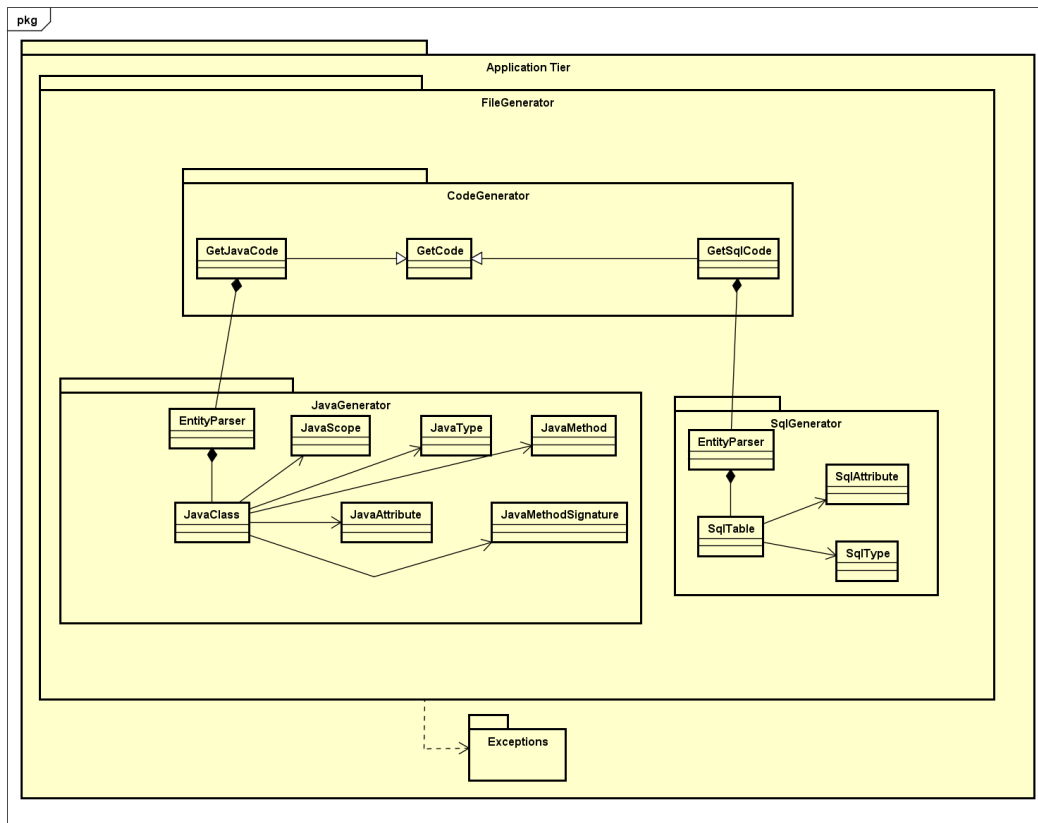


Figura 5: Dettaglio ApplicationTier.

Il package *PresentationTier::IronWorksMiddleware* contiene le seguenti classi:

- **ErrorHandler:** Gestisce gli errori del server;
- **NotFoundHandler:** Gestisce gli errori relativi a pagina non trovata;
- **RouterHandler:** Gestisce le richieste REST rivolte al server andate a buon fine;
- **IndexResponseSender:** Risponde inviando il file homepage.html ad una richiesta get diretta al server;
- **ProxyGen:** Esegue controlli su codeGen;
- **CodeGen:** Genera il codice Java e Sql, infine genera l'archivio compresso del codice;

Il package *ApplicationTier::FileGenerator::CodeGenerator* contiene le seguenti classi:

- **GetCode:** Classe base che definisce come viene elaborato e generato un file;
- **GetJavaCode:** Specializzazione di GetCode, definisce come viene elaborato e generato un file Java;
- **GetSqlCode:** Specializzazione di GetCode, definisce come viene elaborato e generato un file SQL;

Il package *ApplicationTier::FileGenerator::JavaGenerator* contiene le seguenti classi:

- **EntityParser**: Si occupa della creazione della classe, degli attributi e dei metodi Java;
- **JavaClass**: Crea la classe Java per composizione usando le classi sotto elencate;
- **JavaScope**: Crea il tipo della classe Java. Inoltre crea il tipo degli attributi della classe Java in base all'input ricevuto;
- **JavaMethodSignature**: Crea la segnatura dei costruttori e dei metodi Crud della classe Java;
- **JavaAttribute**: Crea i campi dati della classe Java;
- **JavaType**: Crea il tipo degli attributi della classe Java in base all'input ricevuto;
- **JavaMethod**: Crea il corpo dei costruttori e dei metodi Crud della classe Java;

Il package *ApplicationTier::FileGenerator::SqlGenerator* contiene le seguenti classi:

- **EntityParser**: Si occupa della creazione della tabella e degli attributi SQL;
- **SqlTable**: Crea la tabella SQL componendola con *SqlAttribute* e *SqlType*;
- **SqlAttribute**: Crea gli attributi della tabella SQL;
- **SqlType**: Crea il tipo degli attributi della tabella SQL in base all'input ricevuto.

Il package *ApplicationTier::Exception* contiene le seguenti classi:

- **ServerError**: Genera un messaggio d'errore personalizzato relativo alla generazione del codice.

6 Estensione delle funzionalità

In questa sezione sono elencati i possibili punti di estensione delle funzionalità dell'applicazione.

6.1 Aggiungere nuovi oggetti al diagramma

Uno dei servizi forniti dall'applicazione è la creazione di diagrammi di robustezza. Gli oggetti creabili nel diagramma sono:

- Actor;
- Boundary;
- Entity;
- Controller.

Per la definizione di nuovi oggetti si deve:

1. Estendere il modello RobustnessModel nel package RobustnessModelShapes;
2. Ridefinire il metodo createElement della classe RobustnessModel nel package RobustnessModelShapes in modo tale che tratti anche la creazione del nuovo elemento definito.

6.2 Aggiungere nuovi metodi Java

I metodi Crud Java gestiti dal sistema sono:

- Costruttore della classe (create);
- Lettura dei dati della classe (read);
- Modifica dei dati della classe (update);
- Eliminazione dei dati della classe (delete).

Per la definizione di nuovi metodi per le classi java si deve:

1. Ridefinire JavaClass nel package FileGenerator::JavaGenerator in modo tale che contenga un nuovo campo dati che si occupi della creazione dei metodi personalizzati per ogni entità;
2. Utilizzare le classi JavaMethodSignature e JavaMethod nel package FileGenerator::JavaGenerator rispettivamente per creare la firma del nuovo metodo e il corpo di tale metodo.

7 Segnalazione degli errori

Per segnalare errori o suggerire miglioramenti è possibile aprire un issue_G nella repository disponibile su GitHub secondo la seguente procedura:

1. Aprire un browser e recarsi al seguente indirizzo
<https://github.com/WarMachineSwe/IronWorks>

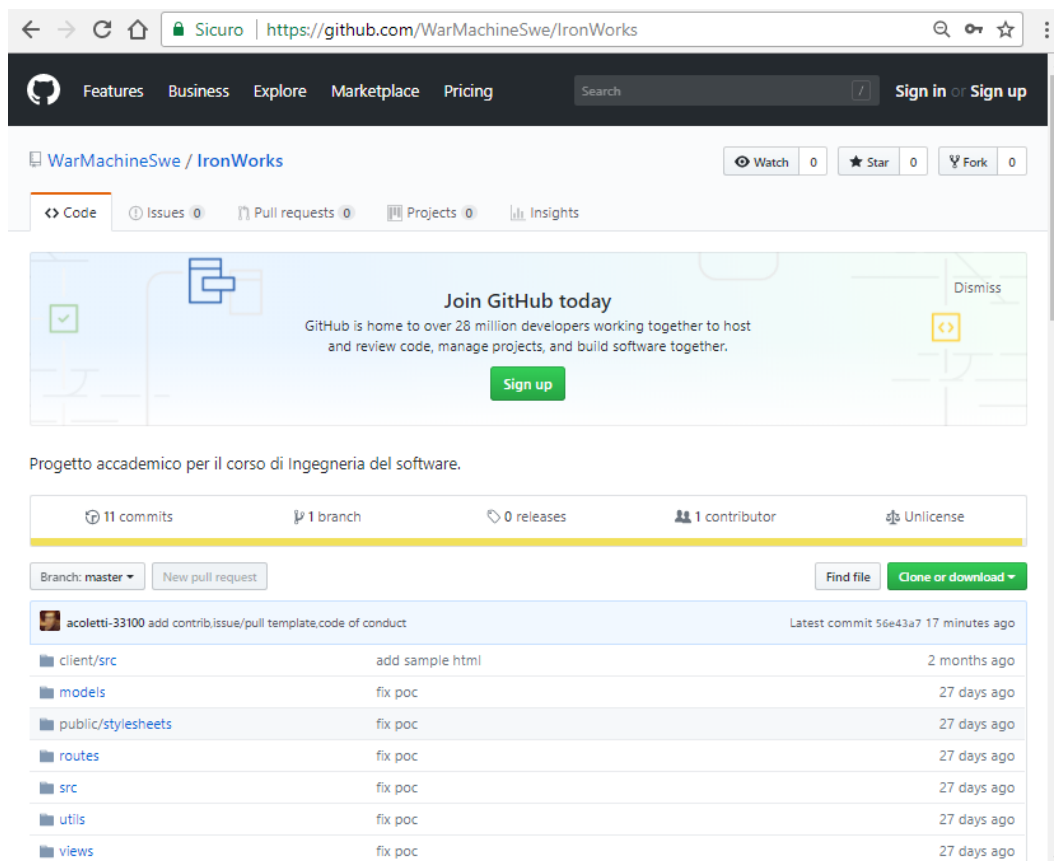


Figura 6: Repository Gith-Hub.

2. Spostarsi nella sezione *Issues*;

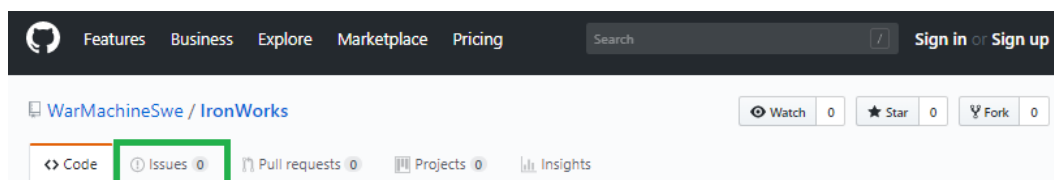


Figura 7: Repository sezione Issues.

3. Cliccare il pulsante *New issue*;

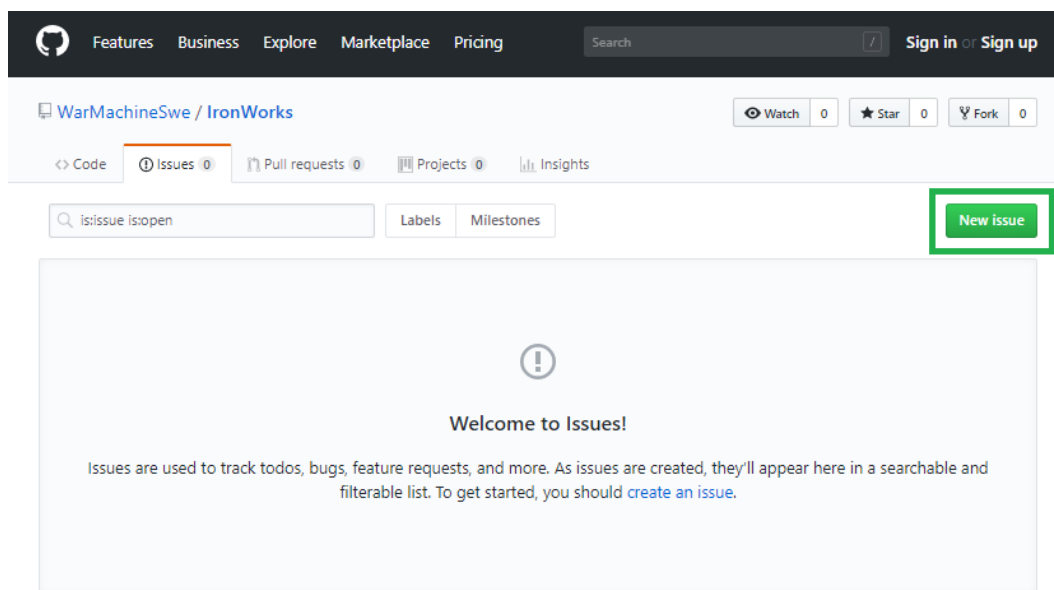


Figura 8: Creare un issue.

4. Inserire il titolo dell'issue;

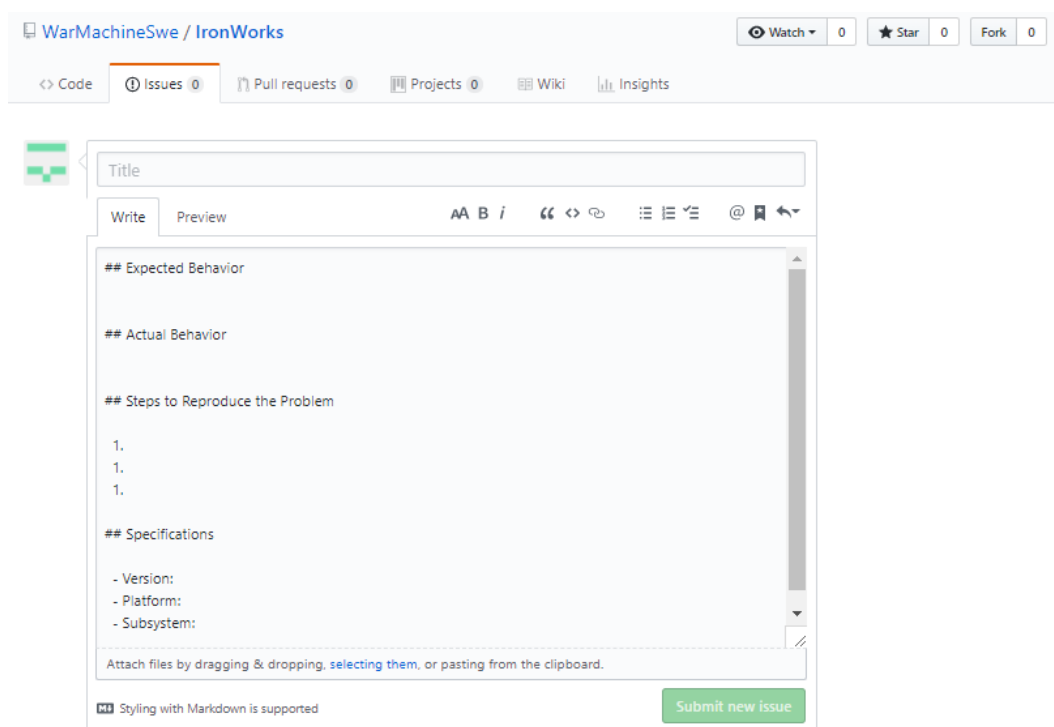


Figura 9: Inserimento titolo issue.



5. Per il testo del commento dell'issue si prega di seguire il template reperibile nella root della repository (ISSUE_TEMPLATE.md):

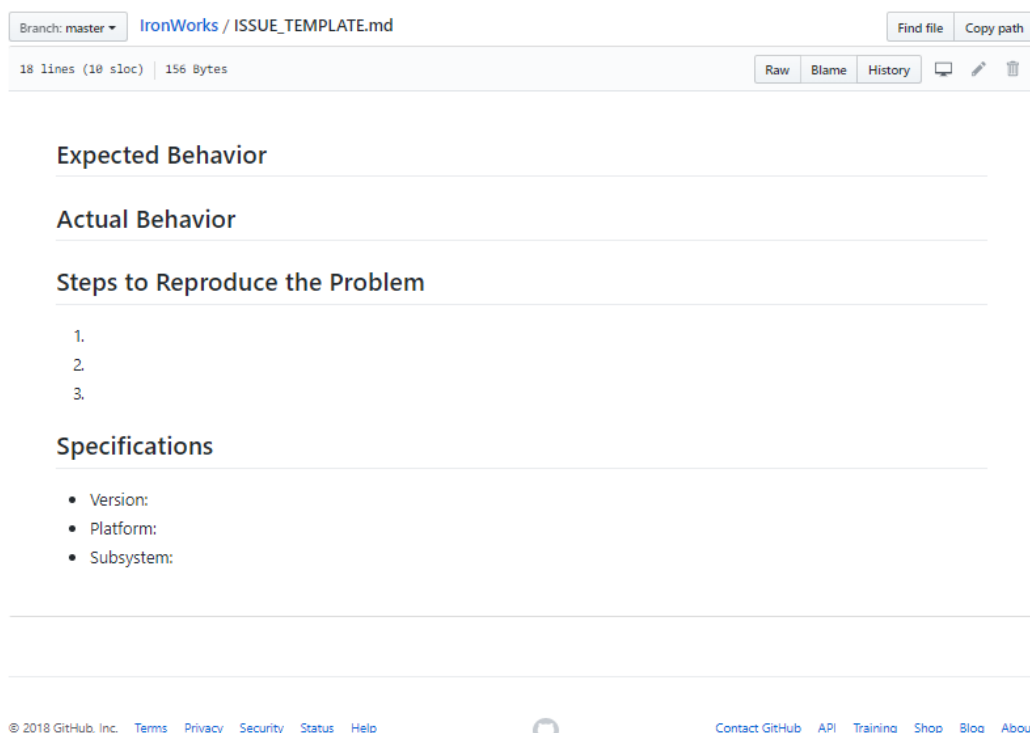


Figura 10: Issue template.

N.B. Per segnalare errori o suggerire miglioramenti è necessario essersi prima registrati a GitHub.

8 Come contribuire

IronWorks è un progetto open-source pertanto è possibile contribuire ad esso. Per lavorare in completa autonomia sul progetto è possibile creare una *fork* della repository. Successivamente per unire le proprie modifiche al progetto originale è necessario creare una *Pull Request*.

N.B. Per contribuire al progetto è necessario essersi prima registrati a GitHub.

8.1 Come creare una fork

Per creare una fork della repository seguire la seguente procedura:

1. Aprire un browser e recarsi al seguente indirizzo
<https://github.com/WarMachineSwe/IronWorks>
2. Cliccare sul pulsante *Fork*;

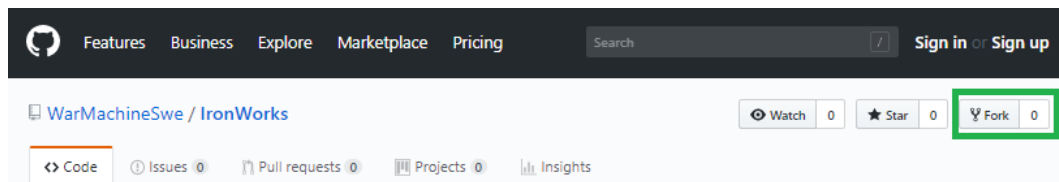


Figura 11: Repository creazione Fork.

3. Scegliere l'account su cui fare il fork;

8.2 Come creare una pull request

Per creare una pull request della repository seguire la seguente procedura:

1. Aprire un browser e recarsi al seguente indirizzo
<https://github.com/WarMachineSwe/IronWorks>
2. Cliccare sul tab *Pull requests*;

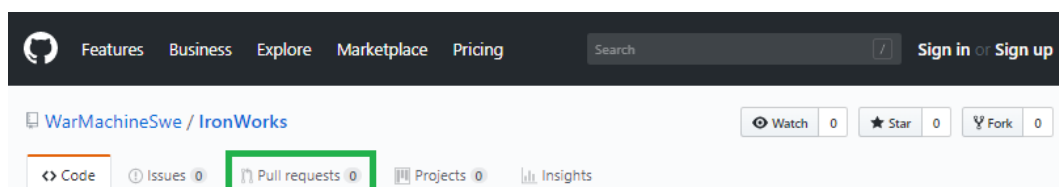


Figura 12: Repository creazione Pull Request.

3. Cliccare sul pulsante *New pull request*;

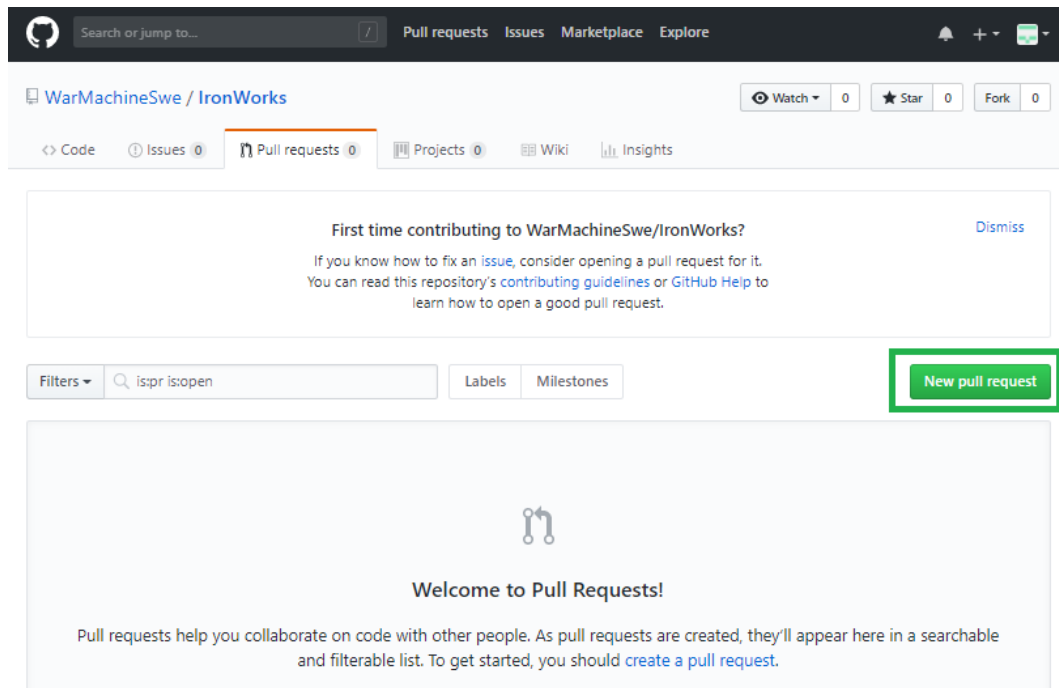


Figura 13: Creare una pull request - creare una pull request.

4. Cliccare su *Compare across forks*;

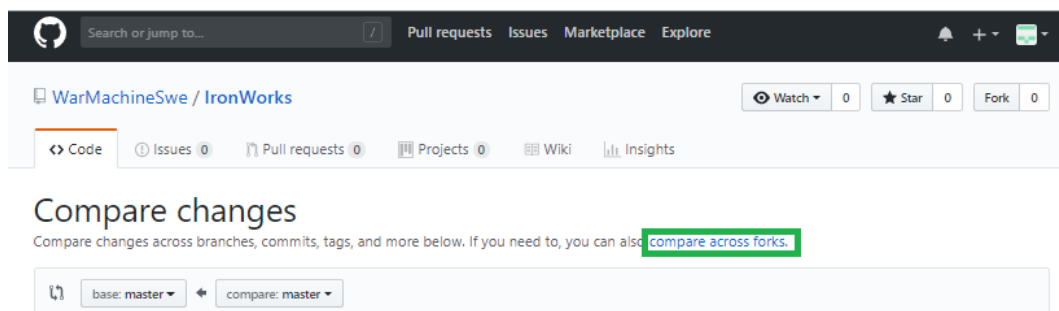


Figura 14: Creare una pull request - compare across forks option.

5. Scegliere il branch su cui si vuole fare il merge;

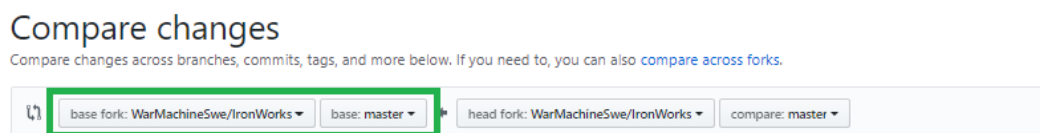


Figura 15: Creare una pull request - scelta del branch su cui fare merge.

6. Scegliere il branch in cui si sono fatti i cambiamenti;

Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also [compare across forks](#).

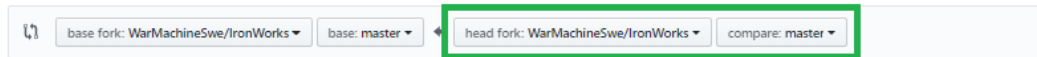


Figura 16: Creare una pull request - scelta del branch modificato.

7. Scrivere un titolo;

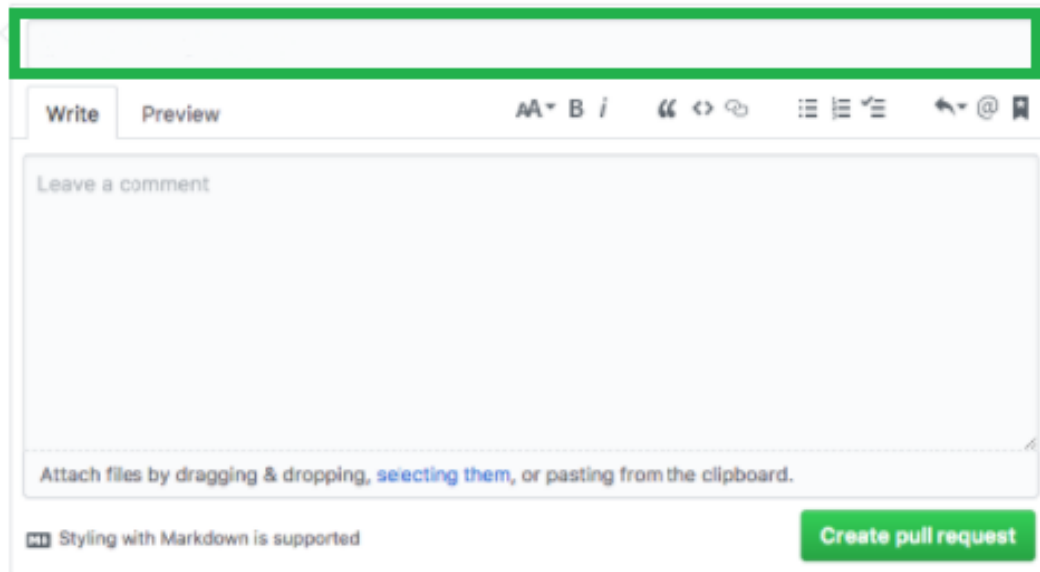


Figura 17: Creare una pull request - titolo.

8. Scrivere una descrizione;

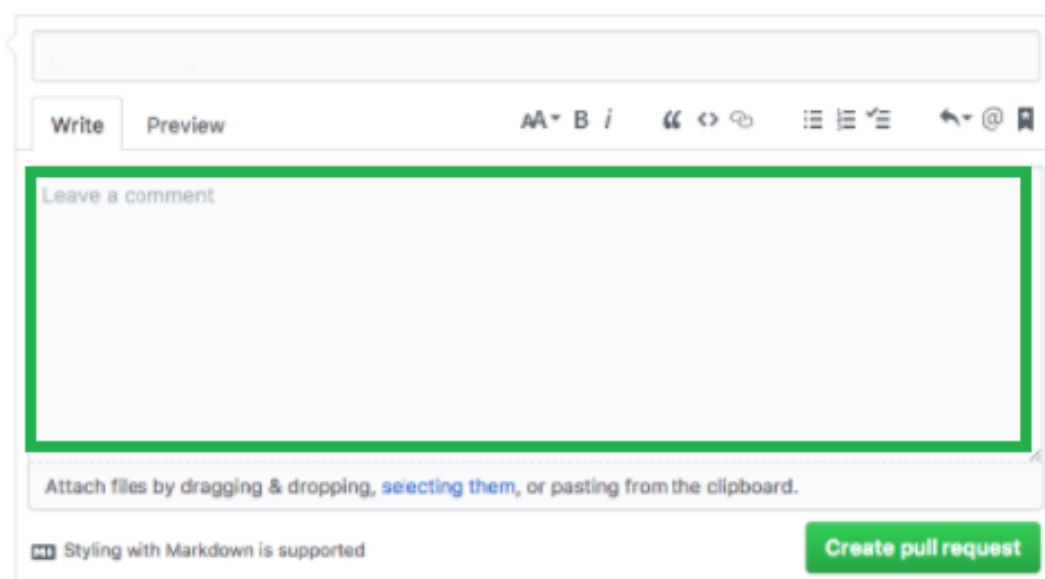


Figura 18: Creare una pull request - descrizione.

9. Cliccare sul pulsante *Create a pull request*;

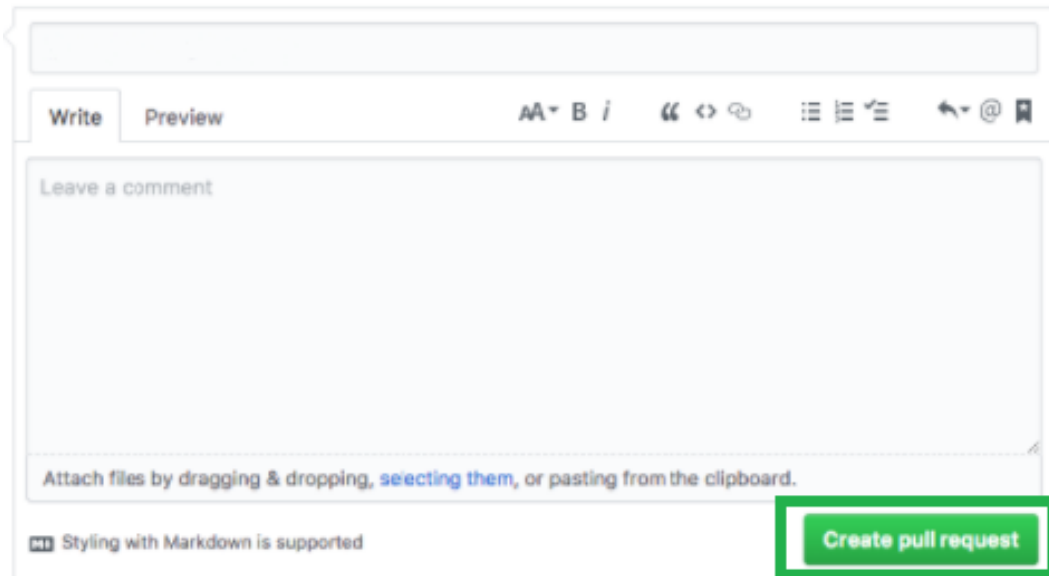


Figura 19: Creare una pull request - creazione pull request.

Glossario

A

Applicazione web

Indica un'applicazione distribuita, accessibile via web per mezzo di un network, in grado di offrire dei servizi ad un client.

API

API (acronimo di Application Programming Interface) indica un insieme di procedure disponibili al programmatore per lo svolgimento di compiti specifici all'interno di un programma. Inoltre con tale termine si intendono le librerie software disponibili in un certo linguaggio di programmazione.

D

DBMS

Database Management System (o Sistema di gestione di basi di dati) è un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database.

Diagramma di robustezza

Il diagramma di robustezza può essere considerato come un ibrido tra un diagramma di classe e un diagramma di attività. Non è uno standard, ovvero non è descritto nella specifica UML, ma utilizza tuttavia i concetti UML e viene usato per supportare la fase di analisi.

E

Entity persistenti

Sono uno degli oggetti appartenenti ai diagrammi di robustezza che appaiono nei documenti come Entity. Rappresentano classi di oggetti (fatti, cose, persone, ...) che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse.

I

Issue

Con il termine issue si intende "errore". In GitHub, gli Issues possono essere usati per tenere traccia di bug, miglioramenti o altre richieste di progetto.



J

Java

Linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, specificatamente progettato per essere il più possibile indipendente dalla piattaforma di esecuzione. Ad oggi è uno dei linguaggi più diffusi al mondo.

[https://it.wikipedia.org/wiki/Java_\(linguaggio_di_programmazione\)](https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione))

L

Linguaggio di markup

Un linguaggio di markup è un linguaggio che consente di descrivere dati tramite dei marcatori (tag).

O

Open source

Open source (sorgente aperta) è un termine che viene utilizzato per riferirsi ad un software di cui gli autori (più precisamente, i detentori dei diritti) rendono pubblico il codice sorgente, favorendone il libero studio e permettendo a programmatori indipendenti di apportarvi modifiche ed estensioni. Questa possibilità è regolata tramite l'applicazione di apposite licenze d'uso.

https://it.wikipedia.org/wiki/Open_source

P

Pedice

Valore inserito nella parte bassa destra di una parola, nella fattispecie la lettera G ad indicare che il termine o l'insieme di parole sono descritte nel glossario.

S

SQL

SQL è l'abbreviazione per Structured Query Language. SQL è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- Creare e modificare schemi di database (DDL - Data Definition Language);
- Inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);
- Interrogare i dati memorizzati (DQL - Data Query Language);
- Creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language).

https://it.wikipedia.org/wiki/Structured_Query_Language



W

World Wide Web

Sigla WWW, è un sistema che permette la condivisione di documenti ipertestuali multimediali, costituiti cioè da un insieme di contenuti testuali, visuali e audio/video, sfruttando l'infrastruttura di Internet.