

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



Sviluppo API REST di un'applicazione web per la pianificazione delle risorse aziendali

Tesi di laurea

Relatore

Prof. Paolo Baldan

Laureando

Marco Brigo

ANNO ACCADEMICO 2022-2023

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di X ore, dal laureando Marco Brigo presso l'azienda Omicron Consulting Srl di Padova nel periodo che va dal 19 Giugno 2023 a X Agosto 2023.

Il progetto consisteva nello sviluppo di un software di richieste di pianificazioni delle risorse aziendali, verso determinati incarichi e secondo determinati parametri. L'obiettivo dello stage era inserirmi all'interno del progetto, più precisamente nel lato back-end. Il mio lavoro si concentrava sulla progettazione e l'implementazione di un'API REST e sulla creazione e la gestione di nuove tabelle nel database aziendale, utilizzate per la fruizione delle funzionalità da me implementate.

L'attività si è svolta partendo da uno studio preliminare delle tecnologie da utilizzare tramite esercitazioni e materiale fornito, passando per una progettazione dell'architettura e del database.

“Life is really simple, but we insist on making it complicated”

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Nome Cognome, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Mese AAAA

Nome Cognome

Indice

1	Introduzione	1
1.1	L'azienda ospitante	1
1.2	Il progetto	2
1.2.1	Presentazione	2
1.2.2	Motivo alla base del progetto	2
1.2.3	Il mio ruolo nel progetto	2
1.3	Convenzioni tipografiche	2
1.4	Organizzazione del testo	3
2	Analisi dei Requisiti	4
2.1	Scopo del capitolo	4
2.2	Descrizione generale	4
2.3	Attori	4
2.4	Casi d'uso	4
2.4.1	Primo caso	4
2.5	Tracciamento dei requisiti	4
3	Background tecnologico	5
3.1	Scopo del capitolo	5
3.2	Tecnologie	5
3.3	Strumenti	7
3.3.1	Strumenti di sviluppo	7
3.3.2	Strumenti di supporto	8
4	Progettazione	10
4.1	Scopo del capitolo	10
4.2	Architettura REST	10
4.2.1	Convenzioni di denominazione REST	11
4.3	Spring MVC	11
4.3.1	Model	11
4.3.2	View	12
4.3.3	Controller	12
4.4	Progettazione del database	12
4.4.1	Configurazione di Docker	12
4.4.2	Modello dati	12
4.5	Configurazione del progetto	13
4.5.1	Spring Initializr	13
4.5.2	File di configurazione	13

5	Analisi dei requisiti	17
5.1	Casi d'uso	17
5.2	Tracciamento dei requisiti	18
6	Progettazione e codifica	20
6.1	Tecnologie e strumenti	20
6.2	Ciclo di vita del software	20
6.3	Progettazione	20
6.4	Design Pattern utilizzati	20
6.5	Codifica	20
7	Verifica e validazione	21
8	Conclusioni	22
8.1	Consuntivo finale	22
8.2	Raggiungimento degli obiettivi	22
8.3	Conoscenze acquisite	22
8.4	Valutazione personale	22
A	Appendice A	23
	Bibliografia	25

Elenco delle figure

1.1	Logo Omicron	1
4.1	Interfaccia Spring Initializr	13
5.1	Use Case - UC0: Scenario principale	17

Elenco delle tabelle

5.1	Tabella del tracciamento dei requisiti funzionali	19
5.2	Tabella del tracciamento dei requisiti qualitativi	19
5.3	Tabella del tracciamento dei requisiti di vincolo	19

Listings

4.1	pom.xml con dipendenze selezionate	14
4.2	application.properties del progetto	16

Capitolo 1

Introduzione

1.1 L'azienda ospitante



Figura 1.1: Logo Omicron

L'azienda Omicron Consulting è specializzata nello sviluppo di software gestionali e di revisione di processi aziendali. Essa è presente nel mercato ICT dal 1980, spiccando su vari settori, in cui hanno effettuato importanti implementazioni in area ICT come: Manufacturing, Automotive, Aerospace, Logistics e altre aree. Con particolare riferimento al settore Manufacturing si sono specializzati nello sviluppo di progetti di trasformazione ERP (acquisendo la certificazione VAR di SAP), stringendo alleanze strategiche con realtà ICT nazionali ed internazionali.

Offrono servizi di gestione e supporto di sistemi ERP, sviluppo di progetti di Business Intelligence e personalizzazione di sistemi software.

Per completare il pacchetto dei servizi offerti, Omicron ha un'esperienza di alto livello negli ambiti Banking, Finance and Insurance, lavorando con le principali istituzioni bancarie e assicurative italiane.

Omicron oltre alle offerte che dedica ai clienti, si occupa anche di garantire un'alta formazione delle proprie risorse, investendo su progetti di ricerca e sviluppo.

1.2 Il progetto

1.2.1 Presentazione

L'applicativo permette la creazione di richieste di pianificazione di risorse aziendali a svolgere un incarico, in un determinato lasso di tempo.

Il Project Manager_g potrà effettuare richieste di pianificazione di risorse aziendali, chiedendo disponibilità di figure professionali con determinate caratteristiche. Esso potrà inoltre visualizzare, tramite una sezione apposita, lo stato delle proprie richieste. In seguito ad una richiesta accettata, il Program Manager_g distribuirà le risorse più adeguate alla richiesta. Successivamente alla distribuzione delle risorse, il Project Manager potrà creare una pianificazione per ogni risorsa richiesta, specificando parametri quali la durata, l'attività da svolgere e altri parametri.

Il software permette la visione della lista completa dei dipendenti o consulenti, con una linea del tempo alla loro destra, contenente le tasks_g in cui sono coinvolti. È possibile interagire con quest'ultime per avere una visione rapida della task, con la possibilità di espanderla o modificarla.

1.2.2 Motivo alla base del progetto

L'approccio utilizzato per la gestione delle pianificazioni delle risorse e la loro disponibilità era gestito tramite file Excel compilati e revisionati dai Program manager. Le richieste di nuove pianificazioni, invece, sono comunicate ai Program manager nei modi più disparati (email, telefono, chat), rendendo difficile tenerne traccia.

Il progetto nasce dunque dall'esigenza di semplificare la gestione delle richieste e delle pianificazioni, garantendo una visione più rapida della disponibilità delle risorse.

1.2.3 Il mio ruolo nel progetto

Le funzionalità da me sviluppate erano principalmente due: la gestione delle richieste e delle pianificazioni. Il tutto è stato realizzato creando nuove tabelle da inserire nel database per la fruizione dei servizi dell'applicativo e lo sviluppo dell'API_g REST_g. Gli endpoint_g dell'API permettevano operazioni CRUD_g su richieste, pianificazioni e sulle altre tabelle da me create.

1.3 Convenzioni tipografiche

Durante la stesura del testo sono state adottate le seguenti convenzioni:

- in ogni elenco puntato è stato deciso di inserire un punto e virgola (;) alla fine di ogni elemento ed un punto (.) per l'ultimo elemento di ogni elenco;
- la sezione del glossario conterrà i termini ritenuti ambigui o non di uso comune, che necessitano quindi di una loro definizione;
- alla prima occorrenza di un termine inserito nel glossario verrà segnato con la seguente nomenclatura: termine_g.

1.4 Organizzazione del testo

Questa sezione è dedicata alla spiegazione della struttura del documento, per dare indicazioni su come è organizzato il testo.

Capitolo 1: introduce il progetto, il mio ruolo all'interno del progetto e il profilo aziendale. Questo capitolo è l'unica parte in cui si parlerà del progetto nella sua totalità rispetto ai capitoli successivi che saranno inerenti esclusivamente al mio lavoro svolto;

Il secondo capitolo descrive i casi d'uso individuati ed i relativi requisiti;

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

Capitolo 2

Analisi dei Requisiti

2.1 Scopo del capitolo

La seguente sezione di Analisi dei Requisiti rappresenta una dettagliata e approfondita esplorazione delle necessità e delle aspettative che guidano la creazione e lo sviluppo del progetto in questione. Questa analisi è stata condotta al fine di definire chiaramente gli obiettivi e le funzionalità del prodotto, fornendo una base solida per la progettazione e l'implementazione del software.

2.2 Descrizione generale

Ogni caso d'uso è stato schematizzato secondo i seguenti punti:

- **attore coinvolto:** in cui si specifica l'attore;
- **descrizione:** offre una spiegazione più dettagliata del caso d'uso;
- **precondizioni:** rappresenta la condizione che deve essere soddisfatta e verificata affinché il caso d'uso possa essere eseguito con successo;
- **postcondizioni:** rappresenta lo stato dell'attore in seguito all'esecuzione con successo del caso d'uso;
- **estensioni:** in cui si specificano le eventuali estensioni collegate.

Vengono inserite anche delle immagini dell'UML_g per fornire una spiegazione visiva che può aiutare maggiormente la comprensione.

2.3 Attori

...

2.4 Casi d'uso

2.4.1 Primo caso

2.5 Tracciamento dei requisiti

Capitolo 3

Background tecnologico

3.1 Scopo del capitolo

Questo capitolo tratta delle tecnologie e gli strumenti di sviluppo e di supporto utilizzati per la realizzazione del prodotto.

L'apprendimento delle seguenti tecnologie e strumenti è stato affrontato nella prima parte del tirocinio. Questo periodo di formazione è durato circa due settimane in cui il tutor mi ha fornito materiale ed esercitazioni per poter comprendere al meglio il contesto tecnologico.

3.2 Tecnologie

Java

Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e fortemente tipizzato, sviluppato originariamente da Sun Microsystems.

La sua popolarità deriva dalla sua portabilità, dalla vasta comunità di sviluppatori e dalle numerose risorse disponibili per l'apprendimento e lo sviluppo.

All'interno del mio progetto è stato utilizzato per lo sviluppo del lato back-end del prodotto.

SQL

SQL, acronimo di Structured Query Language, è un linguaggio di programmazione utilizzato per gestire e manipolare dati in un database relazionale. SQL fornisce una serie di comandi standardizzati che consentono agli sviluppatori e agli amministratori di database di eseguire operazioni come l'interrogazione dei dati, l'aggiornamento dei dati, l'inserimento di nuovi dati e la creazione e gestione degli schemi dei database.

Questo noto linguaggio è stato utilizzato nel progetto per i seguenti motivi:

- creare le tabelle o trigger utili alla fruizione dei servizi del progetto;
- eseguire query per inserire, recuperare, eliminare o modificare dati in base alle richieste.

Microsoft SQL Server

SQL Server è un DBMS (Database Management System) relazionale sviluppato da Microsoft. È una piattaforma dati che si utilizza per creare e gestire database, principalmente in ambito aziendale.

Spring

Spring è un framework_g di sviluppo di applicazioni Java che offre un'ampia gamma di strumenti e librerie per semplificare la creazione di applicazioni aziendali robuste, scalabili e di alta qualità. Spring fornisce anche moduli specifici per la gestione dei dati, lo sviluppo web e la sicurezza, rendendolo uno degli strumenti più utilizzati per lo sviluppo Java.

A questo framework sono associati tanti altri progetti, che hanno nomi composti come Spring Boot, Spring Data e molti altri.

All'interno del progetto Spring è stato utilizzato per sviluppare l'API REST.

Spring Boot

Spring Boot è un modulo del framework di sviluppo Java Spring che semplifica la creazione, la configurazione e l'avvio di applicazioni Java. Fornisce un ambiente pronto all'uso per sviluppare rapidamente applicazioni Spring, eliminando gran parte della complessità associata alla configurazione. Spring Boot utilizza convenzioni intelligenti e configurazioni predefinite per accelerare lo sviluppo, consentendo agli sviluppatori di concentrarsi sulle funzionalità dell'applicazione anziché sulla configurazione di base.

Spring Data

Spring Data è un modulo del framework Spring che fornisce un'astrazione per semplificare l'accesso e la gestione dei dati nelle applicazioni Java. Esso offre un'interfaccia unificata per interagire con una varietà di fonti di dati, tra cui database relazionali, database NoSQL e altri servizi di memorizzazione dati.

Questo modulo è stato utile nel progetto per interfacciare l'applicazione con il database.

JSON

JSON, acronimo di JavaScript Object Notation, è un formato leggero di scambio di dati utilizzato comunemente per rappresentare oggetti e strutture di dati. JSON è ampiamente utilizzato per rappresentare dati in applicazioni web, servizi API, scambio di dati tra client e server, configurazioni di applicazioni e molto altro.

All'interno del progetto permette lo scambio di dati tra il lato front-end ed il lato back-end.

3.3 Strumenti

3.3.1 Strumenti di sviluppo

Gli strumenti di sviluppo sono utilizzati direttamente per creare, implementare e testare le funzionalità dell'applicazione. Essi contribuiscono alla realizzazione delle funzionalità dell'applicazione stessa.

IntelliJ IDEA

IntelliJ IDEA è un potente ambiente di sviluppo integrato (IDE) sviluppato da JetBrains, progettato principalmente per la programmazione in linguaggi come Java, Kotlin, Scala e altri. È noto per la sua ricca serie di funzionalità progettate per migliorare l'efficienza degli sviluppatori e semplificare il processo di sviluppo del software. Il seguente IDE è stato utilizzato per la scrittura del codice in Java.

Maven

Maven è uno strumento di gestione delle build e delle dipendenze che fornisce un sistema di automazione per la compilazione, il testing, il packaging delle applicazioni e il download delle dipendenze.

All'interno di questo progetto Maven è stato utilizzato con Spring Boot per semplificare la gestione delle dipendenze e delle versioni.

DBeaver

DBeaver è un'applicazione di amministrazione di database universale e strumento client SQL. È utilizzato per connettersi, esplorare, gestire e interrogare diversi tipi di database.

All'interno del progetto è stato utilizzato per interagire col database.

Hibernate

Hibernate è un framework di mapping oggetto-relazionale (ORM). L'obiettivo principale di Hibernate è semplificare la gestione e l'accesso ai dati in un database relazionale utilizzando oggetti Java anziché scrivere query SQL manualmente.

Postman

Postman è un'applicazione di sviluppo di API (Application Programming Interface) che consente agli sviluppatori di creare, testare, documentare e monitorare le API. Nel corso del progetto è stato uno strumento altamente utilizzato sia come API testing tool.

3.3.2 Strumenti di supporto

Gli strumenti di supporto sono utilizzati per attività che sostengono lo sviluppo del progetto. Essi contribuiscono a migliorare la gestione, la qualità e l'efficienza del processo di sviluppo.

Microsoft Teams

Microsoft Teams è una piattaforma di comunicazione e collaborazione aziendale sviluppata da Microsoft. Offre strumenti per la chat, le videoconferenze, la condivisione di documenti e la gestione dei progetti, consentendo ai team di lavorare insieme in modo efficace sia in ufficio che a distanza.

Questa piattaforma è stata utilizzata nel corso del progetto per poter comunicare con il tutor anche quando non era in ufficio o con altri colleghi per determinate situazioni lavorative.

Notion

Notion è un'applicazione di gestione delle informazioni, utilizzata per prendere appunti, creare elenchi di attività, scrivere documenti e molto altro grazie alla sua interfaccia flessibile personalizzabile dagli utenti.

Questa applicazione è stata utilizzata nel corso della mia attività di Stage per prendere appunti o tenere traccia delle tasks che dovevo svolgere.

Microsoft Excel

Microsoft Excel è un'applicazione software di fogli di calcolo sviluppata da Microsoft. È utilizzata per creare, organizzare e analizzare dati in forma di tavole e grafici, offrendo inoltre molte funzionalità per eseguire calcoli.

Questa applicazione è stata utilizzata nel progetto allo scopo di velocizzare la creazione di dati fittizi per popolare le tabelle del database per poter testare quanto prodotto.

Visual Studio Code

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft. È progettato per fornire un ambiente di sviluppo leggero, flessibile e personalizzabile per programmatori e sviluppatori.

È stato utilizzato nel progetto scaricando varie estensioni per visualizzare l'API e l'UML del database.

Docker

Docker è una piattaforma di containerizzazione_g che consente di creare, distribuire e gestire container_g virtualizzati_g. Permette di far funzionare le applicazioni in altri ambienti con facilità.

Nel progetto è stato utilizzato per creare un server locale con un'immagine del database MSSQL.

Swagger

Swagger è un insieme di strumenti e specifiche che consentono la documentazione, la progettazione e il test di API. Permette una migliore comunicazione dei propri endpoint dell'API semplificandone la lettura e la comprensione.

Git

Git è un sistema di controllo versione distribuito utilizzato nello sviluppo software. Consente di tenere traccia delle modifiche apportate al codice sorgente e di semplificare la collaborazione tra sviluppatori in progetti di programmazione.

GitLab

GitLab è una piattaforma di sviluppo software basata su web che offre una serie di strumenti per la gestione del ciclo di vita dello sviluppo delle applicazioni. Le principali funzionalità sono: la gestione dei repository Git, la collaborazione tra i membri del team e il monitoraggio delle issue.

All'interno del progetto è stato utilizzata come repository per tenere salvato il progresso del progetto. Nella mia attività di stage ho lavorato su un branch a me dedicati.

Git Extensions

Git Extensions è un'interfaccia grafica per il sistema di controllo versione distribuito Git. Questo software fornisce una modalità visuale per interagire con i repository Git.

Capitolo 4

Progettazione

4.1 Scopo del capitolo

In questo capitolo si tratterà dei giorni successivi all'apprendimento del background tecnologico, in cui sono state improntate le basi dell'architettura del progetto, definendo lo schema del database su cui si poggia l'API.

Nei seguenti paragrafi verrà quindi trattata l'architettura e le configurazioni utilizzate che funzionano da base per l'implementazione di quanto sviluppato.

4.2 Architettura REST

L'architettura REST, acronimo di "Representational State Transfer", è un approccio di progettazione per la creazione di servizi web che si basa sui principi dell'HTTP (Hypertext Transfer Protocol).

Le principali caratteristiche di un'architettura REST includono:

- **Sistema client-server**, dove il client è chi fa le richieste e il server fornisce le risposte;
- **Sistema layered**, perchè possono essere composte da più livelli di servizi indipendenti;
- **Stateless**, significa che il server non contiene client state, quindi ogni richiesta ha abbastanza informazione per il server per processarla;
- **Cacheable**, significa che l'architettura può memorizzare le risposte dei server e riutilizzarle;
- **Resource-based**: questo approccio è considerato tale, dato che si concentra sull'identificazione e la gestione delle risorse all'interno di un sistema. Le risorse vengono identificate da degli URI ed esse possono essere create, aggiornate, richieste o eliminate (operazioni CRUD) attraverso operazioni HTTP standard (POST, PUT, GET, DELETE);
- **Manipolazione delle risorse**, poichè le risorse sono diverse dalla loro rappresentazione logica, utilizzando formati come JSON o XML.

Questo stile architetturale è utilizzato soprattutto per la realizzazione di API poichè l'adozione dei principi standard di HTTP mantiene un'interfaccia uniforme e l'approccio resource-based ne semplifica la progettazione, dato che le risorse vengono identificate da URI_g.

4.2.1 Convenzioni di denominazione REST

Buona prassi nella creazione di un'API REST è il rispetto di convenzioni per la nomenclatura degli endpoint_g. Sebbene non esista un'unica convenzione obbligatoria, esistono delle best practices per garantire una facile lettura e comprensibilità per agevolare gli sviluppatori che adoperano l'API.

Le seguenti convenzioni sono state rispettate nella progettazione dell'API:

- pluralizzare le risorse, per distinguere se si fa riferimento ad una lista di una determinata risorsa o ad una singola risorsa;
- usare lettere minuscole;
- non usare estensioni dei file;
- in caso di nomi composti utilizzare il "-";
- non usare underscore;
- non utilizzare abbreviazioni o slang;
- non utilizzare verbi, poichè l'azione dovrebbe essere indicata dal metodo HTTP utilizzato.

4.3 Spring MVC

Nello sviluppo di una API REST tramite Spring Boot è comune l'utilizzo dell'architettura Model-View-Controller(MVC).

FOTO MVC

4.3.1 Model

Rappresenta i dati e la business logic_g dell'applicazione.

Nel contesto dello sviluppo di un'API REST, il Model contiene risorse come oggetti di dominio o dati provenienti da una fonte esterna come un database.

All'interno del progetto il Model è formato dai seguenti elementi.

Entità

In Spring Boot il Model contiene le classi Java segnate con l'annotazione *@Entity* che identifica che quella classe è mappata ad una tabella del database e contiene dati specifici della tabella.

Repository

Le repository implementate nel progetto gestiscono l'accesso ai dati e definiscono metodi per eseguire operazioni di base sui dati, come l'inserimento, la modifica, la cancellazione e la ricerca. Tutto questo estendendo interfacce JPA, come *JpaRepository*, che consentono di eseguire operazioni in una base di dati senza scrivere codice SQL o query.

DTO

Infine contiene anche i DTO (Data Transfer Object), oggetti utilizzati per modellare le rappresentazioni dei dati che vengono restituite dall'API, ma anche per le richieste effettuate.

4.3.2 View

4.3.3 Controller

4.4 Progettazione del database

4.4.1 Configurazione di Docker

I container Docker offrono un'isolamento completo dell'ambiente, che aiuta a evitare conflitti di dipendenze e interferenze con altre applicazioni o servizi che potrebbero essere presenti sul sistema. Per questo motivo si è deciso di utilizzare l'approccio di sandboxing che offre Docker, poichè l'utilizzo di container consente di isolare le applicazioni e i servizi in ambienti virtualizzati, condividendo il kernel del sistema operativo host ma separando le loro risorse e i loro processi.

Dato che le tabelle di mia creazione dovevano integrarsi con il database aziendale, tramite il tool Docker Compose e un file di configurazione *docker-compose.yaml*, è stato possibile avviare un nuovo container contenente un'immagine di Microsoft SQL Server, che forniva un backup del database aziendale con dati e tabelle.

4.4.2 Modello dati

IMMAGINE UML SENZA ATTRIBUTI

Nel seguente disegno si può notare il database su cui poggia l'API.

È stata inserita una nomenclatura «esterna» per indicare le tabelle provenienti dal database aziendale.

Di seguito elenco le tabelle da me create, spiegandone l'utilizzo e gli attributi:

TUTTE LE TABELLE MIE

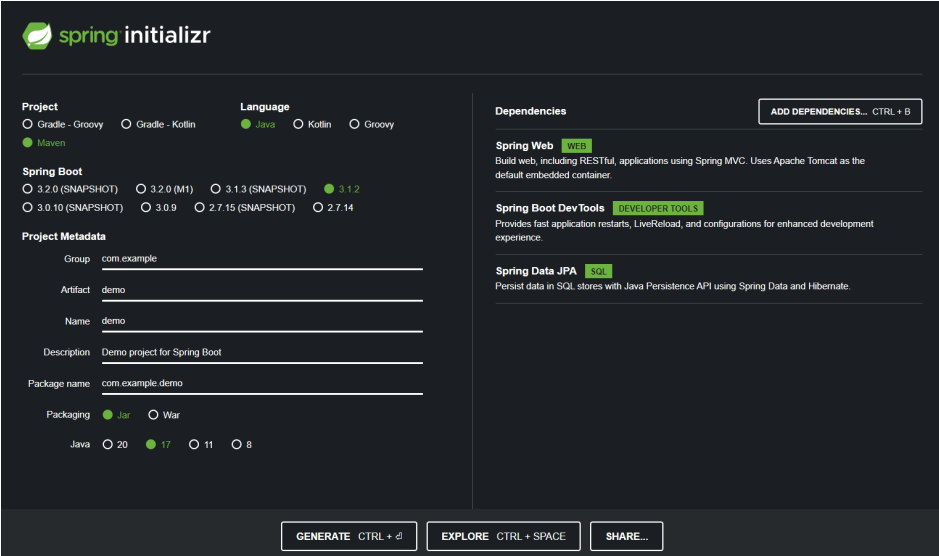


Figura 4.1: Interfaccia Spring Initializr

4.5 Configurazione del progetto

4.5.1 Spring Initializr

Spring Initializr è uno strumento online fornito dalla community di Spring Framework, che consente di creare rapidamente un progetto Spring Boot personalizzato, con le dipendenze e le configurazioni preselezionate dall'utente. Questo strumento semplifica notevolmente il processo di inizializzazione di un progetto Spring Boot, permettendo agli sviluppatori di risparmiare tempo e concentrarsi sulla scrittura del codice.

L'utente può selezionare il tipo di progetto di cui ha bisogno, come ad esempio un progetto Maven o Gradle, e specificare il linguaggio e la versione di Spring Boot desiderati. Inoltre, può anche inserire i metadati del progetto, come il nome del progetto e il nome dei packages.

Una volta selezionate le opzioni desiderate, l'utente può scegliere le dipendenze per il progetto. Le dipendenze sono librerie di terze parti che forniscono funzionalità aggiuntive al progetto.

Dopo aver selezionato le dipendenze, l'utente può scaricare il progetto Spring Boot personalizzato in formato ZIP.

4.5.2 File di configurazione

Il file ZIP scaricato precedentemente contiene tutti i file necessari per iniziare a lavorare sul progetto, inclusi i file di configurazione. Di seguito riporto il file *pom.xml* che si ottiene creando un progetto con le dipendenze sopra selezionate. La configurazione di Maven avviene proprio tramite questo file.

All'interno di questo file si possono trovare i seguenti tag:

- `<dependencies>`, indica una lista di dipendenze;
- `<build>`, contiene impostazioni di costruzione e compilazione;

- `<plugins>`, contiene plugin di Maven;
- `<properties>`, contiene proprietà definite dall'utente;
- `<groupId>`, organizzazione che ha creato il progetto;
- `<artifactId>`, nome unico del progetto;
- `<version>`, versione del progetto.

Nel file si possono notare le seguenti dipendenze:

- *spring-boot-starter-web*, per utilizzare il framework Spring MVC per la creazione di applicazioni Web;
- *spring-boot-starter-jpa*, per la persistenza dei dati;
- *spring-boot-devtools*, offre tools per migliorare il processo di sviluppo come live reload e rilascio automatico;
- *spring-boot-starter-test*, per includere librerie di testing.

Rispetto al file utilizzato nel progetto mancano le seguenti dipendenze:

- *Dipendenza 1*;
- *Dipendenza 2*;
- *Dipendenza 3*.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
3   .w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.
5     apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <parent>
8     <groupId>org.springframework.boot</groupId>
9     <artifactId>spring-boot-starter-parent</artifactId>
10    <version>3.1.2</version>
11    <relativePath/> <!-- lookup parent from repository -->
12  </parent>
13  <groupId>com.example</groupId>
14  <artifactId>demo</artifactId>
15  <version>0.0.1-SNAPSHOT</version>
16  <name>demo</name>
17  <description>Demo project for Spring Boot</description>
18  <properties>
19    <java.version>17</java.version>
20  </properties>
21  <dependencies>
22    <dependency>
23      <groupId>org.springframework.boot</groupId>
24      <artifactId>spring-boot-starter-data-jpa</artifactId>
25    </dependency>
26    <dependency>
27      <groupId>org.springframework.boot</groupId>
28      <artifactId>spring-boot-starter-web</artifactId>
29    </dependency>
30    <dependency>

```

```
30         <groupId>org.springframework.boot</groupId>
31         <artifactId>spring-boot-devtools</artifactId>
32         <scope>runtime</scope>
33         <optional>true</optional>
34     </dependency>
35     <dependency>
36         <groupId>org.springframework.boot</groupId>
37         <artifactId>spring-boot-starter-test</artifactId>
38         <scope>test</scope>
39     </dependency>
40 </dependencies>
41
42 <build>
43     <plugins>
44         <plugin>
45             <groupId>org.springframework.boot</groupId>
46             <artifactId>spring-boot-maven-plugin</artifactId>
47         </plugin>
48     </plugins>
49 </build>
50
51 </project>
```

Listing 4.1: pom.xml con dipendenze selezionate

Il secondo file di configurazione è *application.properties*. Di seguito riporto il file che ho utilizzato nel progetto:

```
1 logging.level.root=INFO
2 logging.level.org.springframework=ERROR
3 logging.level.org.hibernate=TRACE
4 logging.level.it.omicronconsulting=DEBUG
5 logging.level.org.hibernate.SQL=DEBUG
6 logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
7
8
9 spring.datasource.url=jdbc:sqlserver://localhost;databaseName=
   DBDELIVERONE;Trusted_Connection=False;\MultipleActiveResultSets=true;
   encrypt=true;trustServerCertificate=true;
10 spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.
   SQLServerDriver
11 spring.datasource.username=SA
12 spring.datasource.password=Password.1
13
14
15 spring.jpa.properties.hibernate.format_sql=true
16 spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServerDialect
17 spring.jpa.hibernate.ddl-auto=validate
18
19 springdoc.swagger-ui.filter=true
20 springdoc.swagger-ui.tagsSorter=alpha
```

Listing 4.2: application.properties del progetto

Possiamo notare come il file utilizzi un formato di configurazione basato su chiavi e valori. Le chiavi rappresentano le diverse proprietà di configurazione dell'applicazione, mentre i valori rappresentano le impostazioni specifiche.

Nel file possiamo notare le seguenti chaivi:

- *logging.level*, servono per configurare il livello di dettaglio dei log per diverse classi o package all'interno dell'applicazione;
- *spring.datasource*, servono per collegarsi ad un database, in questo caso locale, inserendo username e password e driver di MSSQL;
- *spring.jpa.hibernate* e *spring.jpa.properties.hibernate*, servono per configurare le impostazioni di Hibernate, impostando la validazione schema-entità, il dialetto_g del server SQL e il suo livello di log;
- *springdoc.swagger-ui*, libreria che fornisce integrazione tra Spring Boot e Swagger UI.

Capitolo 5

Analisi dei requisiti

Breve introduzione al capitolo

5.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

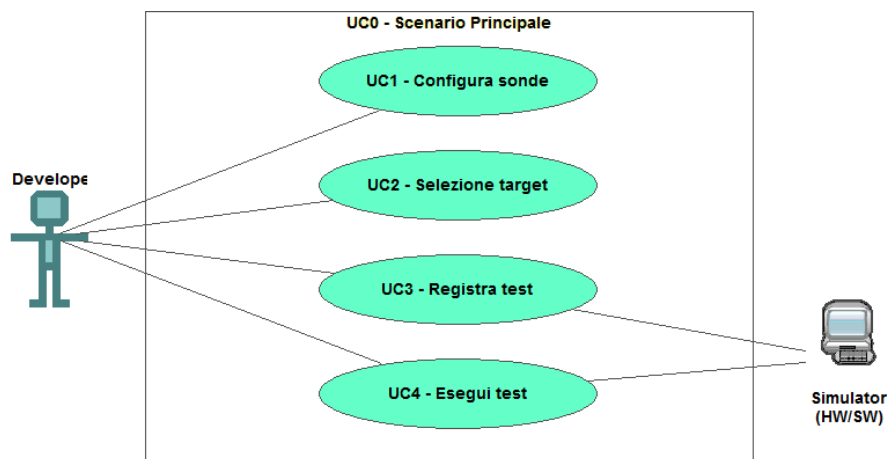


Figura 5.1: Use Case - UC0: Scenario principale

UC0: Scenario principale

Attori Principali: Sviluppatore applicativi.

Precondizioni: Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

Descrizione: La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

5.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.

Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.

Il codice dei requisiti è così strutturato $R(F/Q/V)(N/D/O)$ dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle [5.1](#), [5.2](#) e [5.3](#) sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

Tabella 5.1: Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

Tabella 5.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

Tabella 5.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-

Capitolo 6

Progettazione e codifica

Breve introduzione al capitolo

6.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

Tecnologia 1

Descrizione Tecnologia 1.

Tecnologia 2

Descrizione Tecnologia 2

6.2 Ciclo di vita del software

6.3 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

6.4 Design Pattern utilizzati

6.5 Codifica

Capitolo 7

Verifica e validazione

Capitolo 8

Conclusioni

8.1 Consuntivo finale

8.2 Raggiungimento degli obiettivi

8.3 Conoscenze acquisite

8.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia

- [https://it.wikipedia.org/wiki/Java_\(linguaggio_di_programmazione\)](https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione))
- <https://docs.spring.io/spring-framework/reference/overview.html>
- <https://spring.io/projects/spring-boot>
- <https://spring.io/projects/spring-data>
- https://it.wikipedia.org/wiki/Structured_Query_Language
- <https://www.json.org/json-it.html>
- https://it.wikipedia.org/wiki/IntelliJ_IDEA
- <https://www.baeldung.com/maven>
- <https://en.wikipedia.org/wiki/DBeaver>
- https://it.wikipedia.org/wiki/Microsoft_SQL_Server
- <https://www.baeldung.com/spring-boot-hibernate>
- <https://learning.postman.com/docs/introduction/overview/>
- https://it.wikipedia.org/wiki/Microsoft_Teams
- [https://en.wikipedia.org/wiki/Notion_\(productivity_software\)](https://en.wikipedia.org/wiki/Notion_(productivity_software))
- https://it.wikipedia.org/wiki/Microsoft_Excel
- https://it.wikipedia.org/wiki/Visual_Studio_Code
- <https://it.wikipedia.org/wiki/Docker>
- <https://swagger.io/docs/specification/about/>
- <https://git-scm.com/docs/git>
- <https://it.wikipedia.org/wiki/GitLab>
- <http://gitextensions.github.io/>