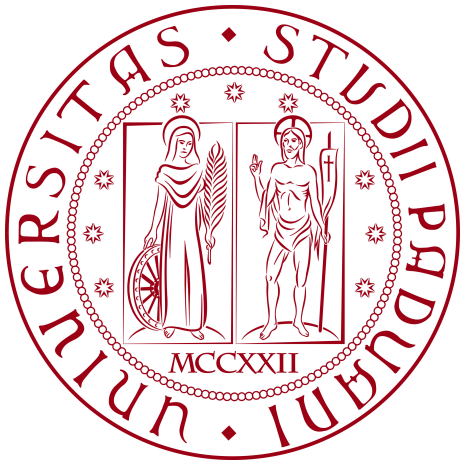
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Realizzazione di un applicativo per gestire le richieste di rimborso e le note spese dei dipendenti

Tesi di Laurea Triennale

Relatore
Prof. Claudio Enrico Palazzi

Laureando
Marco Favaretto
Matricola 2000556

Anno Accademico 2023-2024

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage presso l'azienda Consoft Informatica S.r.l. di Padova. Lo stage si è svolto in conclusione del percorso di studi del Corso di Laurea in Informatica dell'Università degli Studi di Padova ed ha avuto la durata di circa trecento ore.

Lo scopo dello stage è lo sviluppo di un'applicazione web che consenta la gestione di diverse tipologie di rimborso e note spese dei dipendenti di un'azienda. La gestione consiste nella possibilità di inserire, approvare e visionare le richieste dei lavoratori.

Gli obbiettivi includono l'apprendimento di metodologie di analisi, progettazione e sviluppo aziendale, insieme all'uso di tecnologie largamente presenti in ambiti lavorativi. Al termine verrà presentata un'analisi conclusiva confrontante le aspettative iniziali con i risultati ottenuti.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei codici sorgenti

Capitolo 1

Introduzione

Figura 1.1: Consoft Informatica

1.1 L'azienda

Consoft Informatica S.r.l. è una società di consulenza informatica che dal 2006 opera su tutto il territorio nazionale, con sedi a Roma, Milano, Napoli e Padova. L'azienda sostiene il percorso di *Digital Transformation* dei propri clienti realizzando, attraverso le competenze specialistiche su molteplici tecnologie dei nostri *Competence Center* e le partnership con aziende leader di mercato, soluzioni applicative custom ed erogando servizi di consulenza professionale di qualità. Presso le nostre software factory dislocate su Roma e Padova, i vari gruppi di sviluppo hanno la possibilità di lavorare e sperimentare, come veri e propri laboratori software, le più innovative tecnologie per la realizzazione di prodotti ad hoc per i propri clienti o da immettere sul mercato.

1.2 Introduzione al Progetto

L'azienda gestisce i rimborsi e le note spese dei dipendenti tramite l'ufficio amministrativo. Lo scopo dello stage è riuscire ad automatizzare questo processo tramite la realizzazione di una *WebApp*$_G$, che andrà ad aggiungersi a supporto dei processi aziendali amministrativi. L'applicativo in questione sarà dedicato alla gestione, da parte del reparto amministrativo, delle richieste di rimborso di varia natura e note spese inserite, tramite un'apposita sezione, dai dipendenti.

1.2.1 L'Idea

Andando incontro alle necessità dell'azienda, verrà realizzato un applicativo aziendale, fruibile tramite l'intranet aziendale interfacciandosi con altri strumenti gestionali interni. L'applicativo permetterà l'autenticazione, ai dipendenti e al reparto amministrativo, utilizzando l'account *Office365* messo a disposizione dall'azienda.

Il software dovrà:

- Consentire agli amministratori il censimento e la gestione delle varie tipologie di spesa e delle tipologie di rimborso previste dall'azienda.
- Consentire la ricerca, la visualizzazione del dettaglio e la gestione delle richieste inserite dai dipendenti.
- Consentire la compilazione delle richieste di rimborso o note spese, allegando la documentazione necessaria.

1.3 Vincoli del progetto

Vengono qui riportati alcuni dei vincoli imposti per lo sviluppo del prodotto. Questi coincidono con lo stack tecnologico tipicamente usato in azienda.

- Per lo sviluppo lato *backend*$_G$ si adotteranno i *framework*$_G$: *Maven* e *Spring Boot* e il linguaggio *Java*
- Per la persistenza dei dati si utilizzerà il *DBMS*$_G$ *PostgreSQL*
- Per lo sviluppo lato *frontend*$_G$ si userà il *framework Angular*
- Il sistema di versionamento usato sarà *GitLab*$_G$ con account aziendale

Queste e le altre tecnologie usate verranno discusse in dettaglio nell'apposito capitolo.

1.4 Aspettative personali

Lo stage si pone come conclusione del percorso di studi del Corso di Laurea in Informatica e come tale non è solo un esercizio da svolgere, ma anche un occasione di apprendimento di conoscenze sia del mondo informatico che del mondo del lavoro.
Di conseguenza, durante la ricerca di uno stage, si è posta attenzione alla ricerca di un'offerta che soddisfacesse i seguenti obbiettivi personali:

- Apprendimento di alcune pratiche di analisi, progettazione e sviluppo tipiche del mondo aziendale informatico
- Conoscenza di alcune tecnologie ampiamente usate in ambito professionale
- Ottenere una visione di insieme di un progetto informatico, motivo per cui è stato cercato un progetto in cui bisognasse realizzare sia il lato *backend*, sia il *frontend*

1.5 Organizzazione del testo

Il testo è diviso nei seguenti capitoli:

Il primo capitolo fornisce un'introduzione all'attività di stage.

Il secondo capitolo fornisce una panoramica del progetto.

Il terzo capitolo descrive i requisiti e i casi d'uso.

Il quarto capitolo descrive nel dettaglio tutte le tecnologie usate.

Il quinto capitolo presenta la base di dati e l'architettura del sistema.

Il sesto capitolo mostra le parti principali del prodotto finito.

Il settimo capitolo fornisce un resoconto delle attività di stage.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: API_G;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Descrizione del Progetto

2.1 Introduzione al progetto

2.2 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.
I rischi individuati si possono raggruppare in tre categorie:

- Rischi Tecnologici
- Rischi Organizzativi
- Rischi Comunicativi

2.2.1 Rischi Tecnologici

1. Inesperienza nell'uso delle tecnologie scelte
Descrizione: La scarsa esperienza nell'utilizzo delle tecnologie scelte può comportare rallentamenti nello sviluppo e un prodotto non conforme alla qualità richiesta.
Soluzione: Studio approfondito delle tecnologie e delle *best practice* usate, integrando alcuni esercizi semplici e personali.

2. Inconsistenza dei dati nel database
Descrizione: L'applicazione viene testata anche da altri dipendenti dell'azienda e l'aggiunta o modifica dei dati o delle tabelle nel database può comportare errori nella persistenza dei dati.

Soluzione: L'uso di tecnologie adatte per attuare *Database Migration*$_G$ permette di mitigare questo rischio.

2.2.2 Rischi Organizzativi

3. Versionamento del codice
Descrizione: È importante versionare correttamente e frequentemente il codice per permettere uno sviluppo asincrono a più persone o di diverse parti dell'applicativo, garantendo la possibilità di tornare ad una versione stabile precedente.
Soluzione:

- Versionare solo codice corretto
- Pubblicare almeno due volte al giorno sul *VCS*$_G$ adottato
- Aprire i *branch*$_G$ di lavoro necessari

2.2.3 Rischi Comunicativi

4. Mancanza di comunicazione
Descrizione: Una comunicazione inefficace con il tutor e il committente può generare inutili ritardi e una scarsa qualità del prodotto finale.
Soluzione: Chiedere gli aiuti e i *feedback* necessari per avanzare nello svolgimento del progetto.

2.3 Aspettative e richieste del capitolato

Durante i colloqui preparativi allo stage sono state discusse le aspettative e le richieste che il progetto dovrà soddisfare

- Utilizzo delle seguenti tecnologie:
  - PostgreSQL
  - Spring Boot
  - Angular
- Accesso e profilazione tramite *Office 365*$_G$
- Gestione tipologie di rimborso

- Inserimento delle richieste
- Visualizzazione delle richieste
- Comunicazione delle richieste tramite mail
- Visualizzazione di una dashboard per monitorare le richieste
- Generazione di un report, dato un insieme di richieste

Partendo da questo elenco, l'azienda ha previsto il raggiungimento dei seguenti obbiettivi, divisi nelle categorie:

- OB - Obbligatori
- DE - Desiderabili
- OP - Opzionabili

Classificazione	**Obbiettivo**
OB1	Progettazione del software applicativo e della base dati applicativa
OB2	Realizzazione della base dati
OB3	Realizzazione dell'applicativo frontend
OB4	Realizzazione dei servizi API REST backend
DE1	Autonomia nella ricerca e proposizione di soluzioni.
DE2	Invio mail ad inserimento richiesta o richiesta di revisione
OP1	Realizzazione sezione Dashboard Richieste
OP2	Realizzazione sezione report

Tabella 2.1: Tabella obbiettivi aziendali

2.4 Pianificazione

La pianificazione, in termini di quantità di ore lavorative, sarà così distribuita:

Durata in Ore	**Attività**
60	Formazione: • Spring, Spring Boot • Angular
20	Analisi
200	Progettazione e sviluppo: • Progettazione della base dati e dell'architettura applicativa • Realizzazione servizi *REST*$_G$ *backend* • Realizzazione applicativo *frontend*
20	Test e documentazione

Tabella 2.2: Tabella suddivisione ore di lavoro

Capitolo 3

Analisi dei requisiti

3.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo *Unified Modeling Language (UML)*$_G$ 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 gestionale per l'automazione di un processo, discutendo del capitolato con il committente, è risultato che le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario, nello specifico a funzioni di inserimento, visualizzazione e ricerca. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

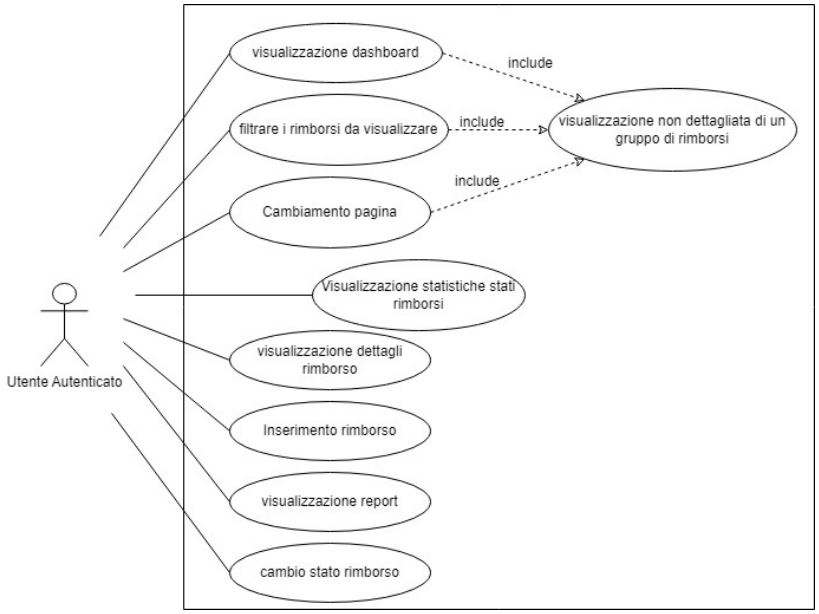


Figura 3.1: Use Cases: Principali casi d'uso

3.1.1 Accesso al sistema

Il sistema è usufruibile solo nell'*intranet* aziendale, il quale sfrutta *Office 365*$_G$ per gestire le autenticazioni. Per questo motivo non sono presenti i tipici pulsanti *login* e *sign in*, poiché l'autenticazione e la registrazione vengono gestite automaticamente dal servizio *O365*$_G$.

UC0: Scenario principale

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione via web ed è registrato nel database.
Descrizione: L'applicazione autenticherà in automatico l'utente e mostrerà la home page, con tutte le funzionalità disponibili.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC0.1: Scenario principale - utente non registrato

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione via web, ma non è registrato nel database.

Descrizione: L'applicazione registrerà in automatico l'utente nel database, ne eseguirà l'autenticazione e mostrerà la home page, con tutte le funzionalità disponibili.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

3.1.2 Visualizzazione schermata principale

La *home page* dell'applicazione racchiude diverse funzionalità e con ciò risponde a diversi casi d'uso.

UC1.0: Visualizzazione Dashboard

Attori Principali: Utente autenticato
Precondizioni: L'utente ha raggiuto l'applicazione via web ed è registrato nel database.
Descrizione: L'utente visualizza nella home page dell'applicazione una dashboard contenente diverse funzionalità e informazioni.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC1.1: Dashboard - Visualizzazione insieme di rimborsi

Attori Principali: Utente Amministratore o Operatore
Precondizioni: L'utente ha raggiuto l'applicazione ed è registrato nel database.
Descrizione: L'applicazione mostrerà in centro alla pagina un sottoinsieme degli ultimi rimborsi inseriti.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC1.2: Dashboard - Visualizzazione insieme di rimborsi utente

Attori Principali: Utente Dipendente
Precondizioni: L'utente ha raggiuto l'applicazione ed è registrato nel database.
Descrizione: L'applicazione mostrerà in centro alla pagina un sottoinsieme degli ultimi rimborsi inseriti dall'utente.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC1.3: Dashboard - Cambio Pagina

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione ed è registrato nel database.
Descrizione: L'applicazione consetirà mediante un pulsante di cambiare il sottoinsieme di rimborsi attualmente visualizzato.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC1.4: Dashboard - Visualizzazione stati rimborsi

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione ed è registrato nel database.
Descrizione: L'applicazione mostrerà il numero di rimborsi raggruppati per ciascuno degli stati del ciclo di vita di un rimborso.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC2: Visualizzazione dettagli di un rimborso

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione e sta visualizzando un insieme di rimborsi.
Descrizione: L'applicazione mostrerà un pulsante su ciascun rimborso, che permetterà di ridirezionare l'utente su una nuova pagina dove verranno mostrati tutti i dettagli del rimborso selezionato.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC3: Cambiamento stato rimborso

Attori Principali: Utente Admin o Operatore
Precondizioni: L'utente ha raggiuto l'applicazione e sta visualizzando un insieme di rimborsi.
Descrizione: L'utente preme un pulsante e seleziona il nuovo stato del rimborso

Postcondizioni: Il sistema ha aggiornato lo stato del rimborso ed è pronto per una nuova interazione.

3.1.3 Ricerca rimborsi

È richiesta la possibilità di filtrare i rimborsi presenti nel database tramite l'inserimento di una serie di valori.

UC4: Ricerca tramite filtri

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione ed è registrato nel database.
Descrizione: L'applicazione mostrerà un pulsante per permettere l'inserimento di diversi filtri per la ricerca di rimborsi all'interno del database.
Postcondizioni: Il sistema è pronto per l'inserimento dei filtri.

UC4.1: Inserimento nuovo filtro

Attori Principali: Utente Generico
Precondizioni: L'utente ha inserito un filtro di ricerca.
Descrizione: L'applicazione mostrerà un pulsante per permettere l'inserimento di una nuova serie di filtri per la ricerca di rimborsi all'interno del database.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC4.2: Rimozione filtro

Attori Principali: Utente Generico
Precondizioni: L'utente ha inserito almeno un filtro di ricerca
Descrizione: L'applicazione mostrerà un pulsante per permettere la rimozione di un filtro per la ricerca di rimborsi all'interno del database.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

3.1.4 Inserimento nuovo Rimborso

È richiesta la possibilità di inserire una nuova richiesta di rimborso.

UC5: Inserimento Rimborso

Attori Principali: Utente Dipendente
Precondizioni: L'utente ha raggiunto la home page.
Descrizione: L'applicazione permetterà all'utente di premere un pulsante che consentirà l'avvio della procedura di inserimento del rimborso
Postcondizioni: Il sistema mostrerà all'utente un menù con le tipologie di rimborso disponibili.

UC5.1: Inserimento Tipologia

Attori Principali: Utente Dipendente
Precondizioni: L'utente ha raggiunto la home page e ha premuto il pulsante di inserimento rimborso.
Descrizione: L'applicazione permetterà all'utente di selezionare la tipologia di rimborso da inserire.
Postcondizioni: Il sistema ridirezionerà l'utente verso una pagina con un form da compilare.

UC5.2: Inserimento valori Rimborso

Attori Principali: Utente Generico
Precondizioni: L'utente è stato ridirezionato verso la pagina di inserimento di un rimborso, dopo aver scelto la tipologia della richiesta.
Descrizione: L'applicazione mostrerà un form da compilare.
Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC5.3: Salvataggio Rimborso

Attori Principali: Utente Generico
Precondizioni: L'utente ha compilato il form richiesto.
Descrizione: L'utente premerà un pulsante che permetterà di salvare la richiesta.
Postcondizioni: Il sistema ha salvato la richiesta nel database ed è pronto per permettere una nuova interazione.

UC5.4: Notifica inserimento rimborso

Attori Principali: Utente Amministratore e Operatore
Precondizioni: L'utente Dipendente ha inserito un nuovo rimborso nel sistema.
Descrizione: Il sistema notificherà tramite una mail gli utenti con il ruolo Amministratore e Operatore dell'inserimento di un nuovo rimborso
Postcondizioni: Il sistema è pronto per una nuova interazione

3.1.5 Report

Una delle richieste opzionali del capitolato chiedeva la generazione di un report, riguardo uno specifico utente e i rimborsi richiesti entro un periodo.

UC6: Richiesta report

Attori Principali: Utente Generico
Precondizioni: L'utente ha raggiuto l'applicazione ed è registrato nel database.
Descrizione: L'applicazione mostrerà un pulsante per permettere la generazione di un report.
Postcondizioni: Il sistema è pronto per accettare i parametri del report.

UC6.1: Inserimento parametri report

Attori Principali: Utente Generico
Precondizioni: L'utente ha premuto il pulsante per l'ottenimento di un report.
Descrizione: Viene richiesto l'inserimento dell'utente e delle due date entro le quali si vuole generare un report.
Postcondizioni: Il sistema ha accettato la richiesta e ha ridirezionato l'utente verso la pagina di visualizzazione del report.

UC6.2: Visualizzazione Report

Attori Principali: Utente Generico
Precondizioni: L'utente ha richiesto la visualizzazione di un report e ha inserito i valori richiesti.

Descrizione: Il sistema mostrerà all'utente il report generato.
Postcondizioni: Il sistema è pronto per una nuova interazione.

3.1.6 Redirect unauthorized, interruzione procedura, ritorno in home

In questa sezione sono esaminati i casi d'uso

UC7: Accesso negato

Attori Principali: Utente Generico
Precondizioni: L'utente ha tentato di accedere ad una funzionalità non predisposta per il suo ruolo.
Descrizione: Il sistema ridirezionerà l'utente verso la *Home page*
Postcondizioni: Il sistema è pronto per una nuova interazione.

UC8: Chiusura finestra inserimento valori

Attori Principali: Utente Generico
Precondizioni: L'utente ha premuto il pulsante per iniziare una procedura di ricerca, o di inserimento rimborso, o generazione report. Tuttavia, non ha ancora completato la richiesta.
Descrizione: L'utente può chiudere la finestra cliccando gli appositi pulsanti.
Postcondizioni: Il sistema ha chiuso la finestra di inserimento valori ed è pronto per una nuova interazione.

UC9: Ritornare alla Home Page

Attori Principali: Utente Generico
Precondizioni: L'utente ha avviato una procedura ed è stato ridirezionato in una nuova pagina.
Descrizione: Il sistema permette all'utente di tornare alla *Home page* tramite la barra di navigazione
Postcondizioni: Il sistema è pronto per una nuova interazione.

3.2 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, dove ogni requisito è identificato con il carattere **R**, è così strutturato:

F: Funzionale.

Q: Qualitativo.

V: Di vincolo.

N: Obbligatorio (necessario).

D: Desiderabile.

O: Opzionale.

Nelle tabelle 3.1 e 3.2 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

3.3 Tabelle dei requisiti

3.3.1 Requisiti Funzionali

Requisito	Descrizione	Use Case
RFN-1	Il sistema deve garantire l'autenticazione automatica	UC0
RFN-2	Il sistema deve registrare in automatico i nuovi utenti	UC0.1
Continua nella prossima pagina...		

Tabella 3.1 – Continuo della tabella

Requisito	Descrizione	Use Case
RFN-3	Il sistema deve categorizzare e riconoscere tre ruoli per l'utente: Dipendente, Amministratore e Operatore	UC0
RFN-4	Il sistema deve permettere all'utente di visualizzare la dashboard una volta autenticato	UC1
RFN-5	Il sistema deve permettere la visualizzazione di un sottoinsieme di rimborsi	UC1.1
RFN-6	Il sistema deve filtrare i rimborsi visualizzati per l'utente attivo, qualora questo abbia il ruolo Dipendente	UC1.2
RFN-7	L'applicazione deve essere fornire un sistema di paginazione per gestire la visualizzazione dei sottoinsiemi di rimborsi	UC1.3
RFN-8	Il sistema deve mostrare sulla dashboard un sommario di numeri di rimborso per ogni stato	UC1.4
RFN-9	Ciascun utente deve essere in grado di poter visualizzare i dettagli di un rimborso	UC2
RFN-10	Ciascun utente deve poter essere in grado di tornare alla home page, dalla pagina di visualizzazione del rimborso	UC9
RFN-11	Gli utenti Amministratore e Operatore devono essere in grado di aggiornare lo stato di un rimborso	UC3

Continua nella prossima pagina...

Tabella 3.1 – Continuo della tabella

Requisito	Descrizione	Use Case
RFN-12	Gli utenti Amministratore e Operatore devono essere in grado di annullare la procedura di aggiornamento lo stato di un rimborso	UC8
RFN-13	Il sistema deve essere in grado di filtrare i rimborsi visualizzati	UC4
RFN-14	Ciascun utente deve essere in grado di aggiungere filtri di ricerca	UC4.1
RFN-15	Ciascun utente deve essere in grado di rimuovere dei filtri di ricerca	UC4.2
RFN-16	Ciascun utente deve essere in grado di poter annullare la procedura di ricerca	UC8
RFN-17	L'utente Dipendente deve essere in grado di poter accedere ad una schermata per inserire un nuovo rimborso	UC5
RFN-18	L'utente Dipedente deve poter selezionare il tipo di rimborso da inserire	UC5.1
RFN-19	L'utente deve essere in grado di poter annulare la procedura di inserimento della tipologia di richiesta	UC8
RFN-20	Il sistema deve essere in grado di mostrare all'utente un form da compilare per inserire un nuovo rimborso	UC5.2
RFN-21	L'utente deve poter salvare nel sistema il rimborso creato	UC5.3

Continua nella prossima pagina...

Tabella 3.1 – Continuo della tabella

Requisito	Descrizione	Use Case
RFN-22	L'utente deve essere in grado di annullare la procedura di inserimento di un rimborso	UC9
RFN-23	L'utente Dipendente deve poter essere in grado di tornare alla home page, dalla pagina di inserimento del rimborso	UC9
RFO-24	Il sistema deve essere in grado di notificare gli utenti con il ruolo Amministratore e Operatore dell'inserimento di un nuovo rimborso	UC5.4
RFO-25	Deve essere in grado di generare un report sulla base di un insieme di dati	UC6
RFO-26	Ciascun utente deve poter richiedere la visualizzazione di un report	UC6
RFO-27	L'utente con ruolo Amministratore o Operatore devono poter generare il report sulla base dei seguenti parametri: • Utente Dipendente • Periodo di tempo: due date	UC 6.1
RFO-28	L'utente dipendente deve poter generare un report basato su un certo periodo di tempo	UC6.1
RFO-29	Ciascun utente deve poter annullare la procedura di inserimento dei parametri per la generazione di un report	UC8-9
RFO-30	Ciascun Utente deve poter visualizzare il report generato	UC6.2
Continua nella prossima pagina...		

Tabella 3.1 – Continuo della tabella

Requisito	Descrizione	Use Case
RFO-31	Ciascun utente deve poter tornare alla home page dalla pagina di visualizzazione report	UC9
RFN-32	Il sistema deve ridirezionare gli utenti in home page, qualora questi abbiano tentato di accedere a funzionalità non permesse dal loro ruolo	UC7

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali.

3.3.2 Requisiti di Vincolo

Requisito	Descrizione	Use Case
RVN-1	Il linguaggio di programmazione per il lato backend sarà Java	-
RVN-2	Verrà usato PostgreSQL per la gestione del database	-
RVO-3	Potrà essere usato Docker come container per PostgreSQL	-
RVN-4	Verrà usato il framework Maven per il lato backend	-
RVN-5	Verrà usato il framework Spring Boot per il lato backend	-
RVN-6	Il lato frontend sarà realizzato con i linguaggi Typescript, HTML e CSS	-

Continua nella prossima pagina...

Tabella 3.2 – Continuo della tabella

Requisito	Descrizione	Use Case
RVN-7	Verrà usato il framework Angular per il lato frontend	-
RVO-8	Potrà essere usata la libreria Bootstrap per il lato frontend	-
RVN-9	Sia il lato backend, che il lato frontend saranno versionati tramite GitLab	-

Tabella 3.2: Tabella del tracciamento dei requisiti di vincolo.

Capitolo 4

Tecnologie e strumenti

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

4.1 Ambienti di Sviluppo e IDE

Per l'analisi, la progettazione e lo sviluppo del codice sorgente sono state usate le seguenti tecnologie.

4.1.1 Draw.io

Figura 4.1: logo draw.io

draw.io è una tecnologia *open source*$_G$ largamente usata dagli sviluppatori che permette di creare diagrammi di diverse tipologie: flusso, classi, entità-relazione, etc.
Per questo progetto è stata utilizzata per creare i diagrammi del database e dei casi d'uso (in figura 3.1).

4.1.2 IntelliJ IDEA

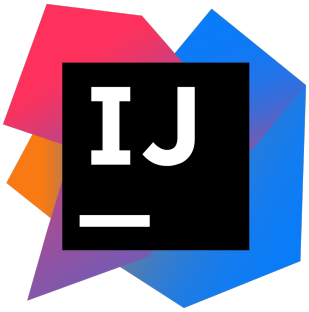


Figura 4.2: logo IntelliJ

IntelliJ IDEA è un *IDE*$_G$ per il linguaggio di programmazione Java. Sviluppato da JetBrains (prima conosciuto come IntelliJ), è disponibile sia tramite licenza a pagamento, che in edizione proprietaria commerciale. Per questo progetto è stata usata la licenza a pagamento, garantita con l'account universitario.

4.1.3 DBeaver

Figura 4.3: logo DBeaver

DBeaver è un software *open source*$_G$ che mira ad essere uno strumento di gestione di database universale, supportandone di diversi tipi e distribuzioni. Per il progetto è stata usata la versione gratuita *DBeaver Community* e il database usato è *PostgreSQL*.

4.2 Linguaggi

Per lo sviluppo del progetto sono stati usati i seguenti linguaggi.

4.2.1 Java

Figura 4.4: logo Java

Java è un linguaggio di programmazione ad alto livello orientato agli oggetti ampiamente usato nelle applicazioni sia desktop che web. Rispetta il principio *WORA (Write once, Run anywhere)* grazie alla *JVM (Java Virtual Machine)*; è quindi indipendente dall'hardware dove viene eseguito e il codice non necessita di modifiche in caso di portabilità su più dispositivi. La versione usata per questo progetto è *Java 21*.

4.2.2 SQL

Figura 4.5: logo SQL

SQL, o *Structured Query Language*, è un linguaggio standardizzato per la gestione di database basati sul modello relazionale. Permette di eseguire diverse operazioni, tra cui: definizione di dati, manipolazione di dati e interrogazione di dati.

4.2.3 Typescript

Figura 4.6: logo TypeScript

TypeScript è un linguaggio di programmazione ad alto livello che estende il più noto *JavaScript*. Questo linguaggio, come il suo linguaggio di origine, nasce per lo sviluppo del lato applicativo *frontend*.

4.2.4 HTML5

Figura 4.7: logo HTML5

HTML, o *HyperText Markup Language*, è un linguaggio di marcatura usato per la formattazione, l'impaginazione e il contenuto di documenti ipertestuali usati nel web. Si occupa della gestione del contenuto di una pagina web. La sua sintassi è stabilita dal World Wide Web Consortium (W3C).

4.2.5 CSS3

Figura 4.8: logo CSS3

CSS, o *Cascading Style Sheet*, è un linguaggio usato per la formattazione di documenti ipertestuali usati nel web. Si occupa della gestione della rappresentazione visiva di una pagina web. La sua sintassi è stabilita dal World Wide Web Consortium (W3C).

4.3 Framework

I linguaggi del progetto sono stati utilizzati nel contesto dei seguenti *framework*$_G$

4.3.1 Maven

Figura 4.9: logo Maven

Apache Maven è uno strumento di gestione di progetti software. Basato sul concetto di *Project Object Model (POM)*, Maven è in grado di gestire la creazione, la gestione e il ciclo di vita di un progetto, a partire da una configurazione centrale.

4.3.2 Spring boot

Figura 4.10: logo Spring Boot

Spring Boot è un *framework*$_G$, che estende l'originale *Spring*, usato per sviluppare applicazioni Java sia web che desktop, fino anche ai microservizi. In breve, applica il principio di *convention-over-configuration* al *framework Spring* minimizzando la configurazione necessaria da fare in fase pre-sviluppo, garantendo comunque flessibilità, oltre a facilità d'uso e robustezza.

4.3.3 Angular

Figura 4.11: logo Angular

Angular è un *framework* per lo sviluppo di applicazioni web. Le applicazioni sviluppate in Angular vengono eseguite interamente dal web browser del *client* dopo essere state scaricate dal web server. Il codice generato da Angular può essere eseguito su tutti i principali web browser moderni quali ad esempio Chrome, Microsoft Edge, Opera, Firefox, Safari ed altri.

4.4 Librerie

Di seguito vengono presentate le librerie usate per lo sviluppo dell'applicazione.

4.4.1 Librerie di Java e Spring

Le seguenti librerie sono state usate per lo sviluppo del lato backend.

- *Spring Boot Starter JPA*: usata per le integrazioni con il database
- *Spring Boot Starter Web*: usata per realizzare le comunicazioni web del lato backend
- *Flyway Core e Flyway PostgreSQL*: usata per gestire le *Database Migration*$_G$
- *PostgreSQL*: usata per integrare il l'ambiente del *DBMS* nell'applicazione
- *Lombok*: usata per le annotazioni del linguaggio *Java*
- *Spring Boot Starter Test*: usata per la scrittura e l'esecuzione di test
- *Spring Boot Maven Plugin*: necessaria per l'integrazione di *Spring* con *Maven*
- *Spring Doc OpenAPI*: usata per l'interfaccia utente chiamata "*Swagger*", che genera una documentazione delle *REST API* sviluppate

4.4.2 Flyway

Figura 4.12: logo Flyway

Flyway è uno strumento per effettuare *Database Migration*$_G$. Consente di versionare e gestire le migrazioni direttamente in linguaggio SQL.

4.4.3 Bootstrap

Figura 4.13: logo Bootstrap

Bootstrap è una libreria libera usata per la creazione di applicazione e siti web. Contiene una serie di modelli *HTML* e *CSS* che possono essere integrati per migliorare l'organizzazione e la resa grafica del contenuto. Peculiarità di questa libreria sono la *responsiveness*$_G$ e la facilità di implementazione di caratteristiche di accessibilità.

4.5 Altri

Qui vengono elencati gli altri strumenti usati durante lo stage.

4.5.1 Docker

Figura 4.14: logo Docker

Docker è un software libero progettato per eseguire processi informatici in ambienti *Linux* isolabili e facilmente distribuibili.

4.5.2 PostgreSQL

Figura 4.15: logo PostgreSQL

PostgreSQL è un *DBMS*$_G$ distribuito con licenza libera. Offre tutte le funzionalità di un classico *DBMS* e aggiunge alcune peculiarità che lo rendono all'avanguardia nel settore della gestione di basi di dati.

4.5.3 GitLab

Figura 4.16: logo GitLab

GitLab è una piattaforma web *open source* che permette la gestione ed il versionamento di *repository Git*$_G$.

4.5.4 Hardware

L'azienda ha fornito un *laptop Dell* da usare per il lavoro di sviluppo, durante tutto il periodo dello stage.

Capitolo 5

Progettazione, codifica e verifica

La progettazione e la codifica sono generalmente le fasi più complesse in cui il progetto prende forma e si concretizza. Per questo motivo, come si era previsto nella tabella 2.2, è stata la fase che ha richiesto più tempo.

5.1 Progettazione

Come illustrato nel capitolo precedente, sono stati usati due principali *framework*: *Spring* e *Angular*. I *framework* sono delle tecnologie largamente usate nel mondo informatico, dato il considerevole numero di vantaggi che comportano. Ad esempio, solitamente forniscono già un'architettura logica di supporto al programmatore sulla quale un software può essere progettato e realizzato. Ciò implica che alcuni *design pattern*$_G$ e alcune scelte architetturali vengono imposte dal *framework* adottato.

5.2 Design Pattern utilizzati

5.2.1 MVC - Spring

L'intera struttura dell'applicazione si basa sul *pattern architetturale Model-View-Controller*, nel quale il lato *backend* assume i ruoli di *Model* e *Controller*, mentre il ruolo *View* viene delegato al lato *frontend*.
Questo *design pattern architetturale* è uno dei più diffusi nel mondo dello sviluppo software e uno dei suoi aspetti principali è la netta separazione tra la *Business Logic*$_G$ e la *Presentation Logic*$_G$.

5.2.1.1 Model

I *model* dell'applicazione rappresentano i dati che dovranno essere gestiti dal sistema. Perciò riflettono ogni tabella del database e ogni campo di quest'ultima. Per mantenere una maggiore separazione tra la gestione dei dati e le operazioni su questi ultimi, le classi che descrivono i tipi di dato sono prive di metodi e puramente descrittive. Infatti, la maggior parte delle operazioni viene eseguita nelle classi *service* e in classi create appositamente per specifiche operazioni.

5.2.1.2 Controller

I *Controller* dell'applicazione rappresentano l'insieme delle classi che gestiscono le chiamate *API*$_G$. Ciascun *controller* è riconosciuto dal suo personale codice di chiamata, delega la creazione della risposta alle classi *service* e restituisce la risposta sempre nello stesso formato.

5.2.1.3 View

La parte della *View* è separata dal *backend* ed è situata nel *frontend*. Il *frontend* adotta il *framework Angular* e perciò ne rispecchia le scelte architetturalli. Le funzionalità dell'applicazione vengono analizzate e scomposte in operazioni quanto più possibili atomiche e ad ognuna di queste viene associato un *component*, ossia una classe marcata da *Angular* con delle caratteristiche specifiche.

5.2.2 Data Transfer Object

Ogni modello presente nell'applicazione deve poter essere trasferito tramite chiamata HTTP. Tuttavia, le classi *model* sono oggetti talvolta complessi e con attributi che sono a loro volta oggetti.
Il *design pattern Data Transfer Object* è in grado di rispondere a questa esigenza. Ogni modello presenta la sua versione *Data Transfer Object*$_G$ e risulta necessario poter convertire *model* ad un *DTO* e viceversa. Per questo motivo vengono realizzate una serie di classi nominate *mapper*, le quali presentano solo i due metodi necessari per eseguire le due conversioni.

5.2.3 Builder

In diversi casi non tutti i membri dei *DTO* devono essere definiti in fase di assemblaggio dell'oggetto. Per rispondere a questo problema, è stato realizzato il *pattern Builder*. Questo *pattern* separa la costruzione di un oggetto dalle parti che lo compongono, permettendo di avere diverse rappresentazioni di questo.

5.3 Base di dati

In sede di analisi la struttura del database è stata attentamente studiata, per essere il più possibile adatta a descrivere le entità e le relazioni del progetto. Il database si basa su *PostgreSQL*, utilizzato attraverso il sistema *Docker*. Per eseguire le prime operazioni è stato usato il software *DBeaver*.

5.3.1 Struttura Database

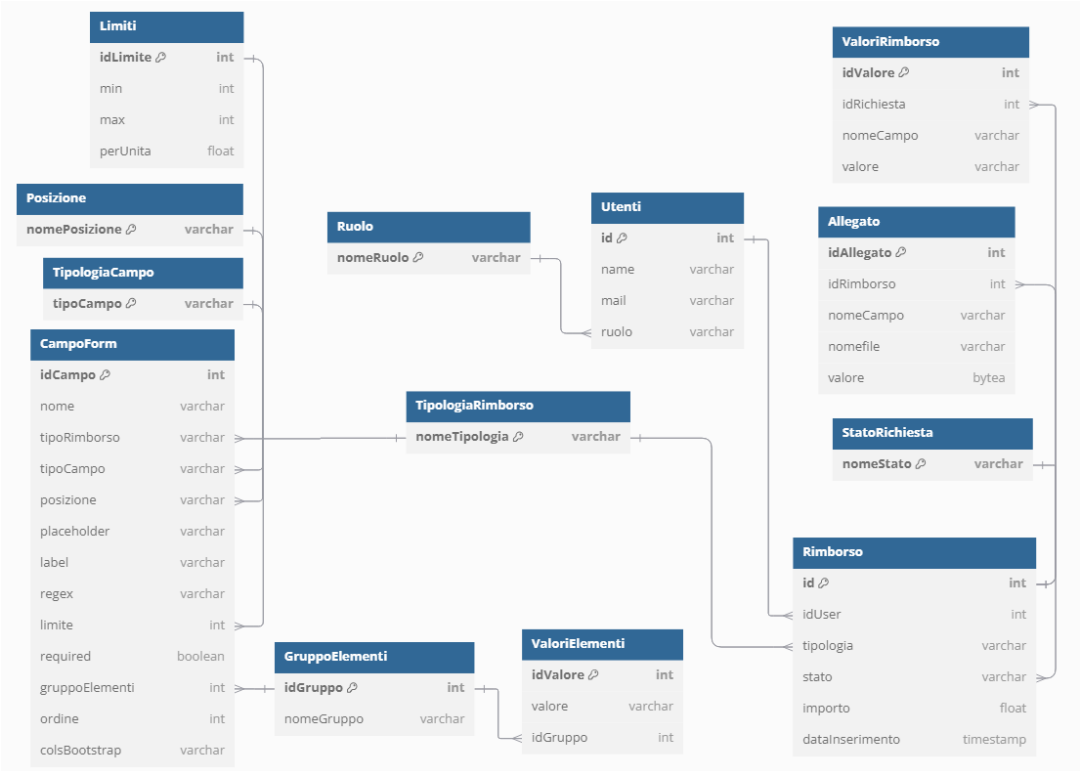


Figura 5.1: Schema Entità Relazione

La struttura del database si può dividere in due sezioni, il cui punto di incontro è la tabella centrale **tipologiaRimborso**. La parte a destra della

struttura si occupa di descrivere il rimborso, i suoi valori ed allegati e l'utente che ne ha fatto richiesta. La parte a sinistra si occupa di descrivere un campo generico appartenente ad un form. Il singolo campo eredita dalle altre tabelle vari elementi descrittivi:

- **tipologiaCampo** descrive il campo HTML
- **posizione** descrive in quale riquadro dovrà essere esposto il campo all'interno del form (tra i valori: *header, body, footer, file*)
- **gruppoElementi** e **valoriElementi** racchiudono eventuali valori per i campi composti da opzioni, quali *select, radio, checkbox, etc.*

La tabella **tipologiaRimborso** è centrale perché permette di racchiudere diversi campi in un unico form e di descrivere una tipologia di rimborso attraverso il medesimo form, legando quindi queste due tabelle. Infatti, i valori di un rimborso sono composti dal nome del campo del form e dal valore inserito dall'utente o calcolato dal sistema, a seconda dei casi.
In sede di analisi sono state identificate anche le tabelle sulle quali risulterebbe utile la generazione di un indice. Tuttavia, non sono stati implementati per mancanza di tempo e non originariamente richiesti dal committente.

5.3.2 Tabelle e relazioni

Di seguito viene mostrata una tabella contenente tutte le entità dello schema in figura 5.1. Le relazioni sono indicate nel seguente modo:

- *nomeTabella* indica da quale tabella si ottiene la *foreign key*
- Escluso dove specificato, le relazioni sono di cardinalità *(1,n)*

Nome Tabella	Descrizione	Relazioni
Ruolo	Insieme dei ruoli possibili dell'utente	-
Utente	Registra gli utenti del sistema	Ruolo
StatoRichiesta	Elenco degli stati in cui può essere un Rimborso	-
TipologiaRimborso	Indica le tipologie di Rimborso salvate nel database	-
Rimborso	Registra i dati essenziali di un rimborso	Utente, StatoRichiesta, TipologiaRimborso
ValoriRimborso	Registra i valori del rimborso richiesto dall'utente	Rimborso
Allegato	Registra gli allegati inseriti nei rimborso	Rimborso
GruppoElementi	Elenca il nome di un insieme di valori opzionali nei campi	-
ValoriElementi	Elenca i valori di un gruppo di elementi	GruppoElementi
TipologiaCampo	Indica la tipologia HTML del campo di un form	-
Posizione	Indica la posizione di un campo all'interno di un form	-
Limite	Esprime alcuni limiti di valori o vincoli per certi campi	-

Continua nella prossima pagina...

Tabella 5.1 – Continuo della tabella

Nome Colonna	Descrizione	Relazioni
CampoForm	Descrive il campo di un form	TipologiaRimborso, GruppoElementi (0,n), Posizione, TipologiaCampo, Limite (0,n)

Tabella 5.1: Tabella mostrante tutte le entità dello schema relazionale.

5.3.3 Popolamento database e Migration

Durante tutta la durata del progetto, il database ha dovuto subire modifiche e adattamenti per certe necessità emerse in fase di sviluppo. Tuttavia, la struttura si è rivelata solida, poiché gli unici cambiamenti strutturali hanno riguardato l'aggiunta di alcune colonne descrittive.
Oltre alle modifiche strutturali, durante le fasi di sviluppo e testing sono stati inseriti nel database dei valori fittizi e in fase di collaudo dei valori reali forniti dall'azienda. Normalmente ciò avrebbe comportato l'utilizzo di un *DMBS*, tuttavia, usando la libreria *Flyway* è stato possibile gestire facilmente queste modifiche con le *Database Migration*$_G$.

5.4 Codifica Back End

Come spiegato ad inizio capitolo, il *backend* si occupa dei lati *Model* e *Controller* del *pattern MVC* e gestisce la logica applicativa mediante una serie di classi specfiche.
Di seguito vengono elencati i vari *package*$_G$ creati per questo progetto.

5.4.1 Tabella Package Realizzati

Nome Package	Descrizione
constant	Contiene alcune eccezioni.
controller	Include la parte dei servizi API.
dto	*Package* dedicato agli oggetti che vengono trasferiti dal backend al frontend e viceversa. Contiene quindi i *Data Transfer Object*, insieme alle *request*, alle *response* e alle specifiche per il filtraggio dei rimborsi. Queste classi sono organizzate in *package* interni.
exception	Contiene la base per le eccezioni usate nell'applicativo.
mapper	Contiene delle classi dedicate alla conversione da *model* a *dto* e viceversa.
models	Racchiude tutte le classi descrittive delle entità del database
repo	Contiene le interfacce espandenti le classi *JpaRepository* della libreria di *Spring Boot*, utili per interrogare il database.
service	Contiene le interfacce e le rispettive implementazioni delle classi *service*, che gestiscono le richieste delle API
utils	Racchiude due classi di metodi statici, utili per tre principali funzioni: la gestione dell'inserimento di un rimborso, la gestione di filtri di ricerca e la generazione di un report.

Tabella 5.2: Tabella mostrante i *package* realizzati

5.4.2 API

Le *API* sono di tipo *REST*$_G$, sono gestite dai **controller** del lato *backend*, i quali presentano tutti la medesima struttura:

```
@RequiredArgsConstructor // necessario per la Dependency Injection
@RestController
@RequestMapping("api/tipo/")
public class TipoController {
    // Dependency Injection gestita dal framework
    private final TipoService tipoService;

    @GetMapping("/get-by-id")
    public ResponseEntity<BaseResponse<TipoDto>>
        getById(@RequestParam int id) {
            TipoDto tipoDto = tipoService.getById(id);
            BaseResponse<TipoDto> response = new BaseResponse<>(tipoDto);
            return ResponseEntity.ok().body(response);
    }
}
```

Codice 5.1: Esempio di API che verrà invocata dal frontend con il metodo HTTP GET, passando come parametro un valore numerico.

Il codice mostrato è un esempio generico di come sono state gestite le *API* nel progetto. Avendo tutte lo stesso formato, diventa facile aggiungerne di nuove e la gestione in *frontend* sarà sempre simile. Infatti, la *response* sarà un file *JSON* il cui formato è sempre:

```
    {
        "code": "200",
        "message": "string",
        "data": { }  // contiene il DTO elaborato
        "success": true
    }
```

Codice 5.2: Esempio di JSON rappresentante la struttura di una response andata a buon fine.

5.4.3 Service

Ogni *Controller* richiama il *service* per ottenere il risultato della *request*. Tutti i *service* hanno la propria interfaccia e la propria implementazione e seguono

una struttura simile.

```java
// interfaccia del service
public interface TipoService {
    TipoDto getById(int id);
}

@Service
@Transactional
@RequiredArgsConstructor
public class TipoServiceImpl implements TipoService {
    // Dependency Injection gestita da framework
    private final TipoRepository tipoRepo;
    private final TipoMapper tipoMapper;

    @Override
    public TipoDto getById(int id) {
        Tipo tipo = tipoRepo.findById(id);
        // eventuali altre operazioni necessarie
        return tipoMapper.toDto(tipo);
    }
}
```

Codice 5.3: Esempio di interfaccia di un service e della sua implementazione.

Come per i *controller*, la struttura di un *service* è sempre la stessa, quindi risulta facile la comprensione e l'aggiunta di nuovi metodi qualora sia necessario.

5.4.4 Repositories

I *service*, per interrogare il database, fanno uso delle interfacce *repository*, le quali hanno la seguente struttura:

```java
public interface TipoService extends JpaRepository<Tipo, int> {
    Tipo findById(int id);
}
```

Codice 5.4: Esempio di interfaccia repository.

L'interfaccia *JpaRepository* permette di interrogare il database tramite delle *query* generate automaticamente dal nome del metodo e di associare il risultato al *model* richiesto.

Qualora con il metodo non fosse possibile generare la *query*, si possono implementare delle *query* direttamente con un oggetto di tipo **String**, oppure, si può ricorrere alla libreria *JpaSpecificationExecutor*, la quale usando una classe chiamata *Specification* permette di aggiungere dei parametri alla *query* richiesta. Questo metodo è stato usato per poter soddisfare il requisito di filtraggio dei rimborsi.

Qui viene presentata una tabella contente le principali *API* sviluppate.

Metodo	**Indirizzo**	**Descrizione**
User Controller: api/user/		
POST	insert	Registra l'utente nel sistema
GET	all-user	Restituisce tutti gli utenti
Rimborso Controller: api/rimborsi/		
POST	update	Aggiorna lo stato del rimborso selezionato
POST	report	ricevuto utente e periodo, restituisce un report
POST	filter	Ricevuti dei parametri, ritorna una lista di rimborsi filtrati
GET	user	Ricevuto un utente, restituisci i rimborsi generati da questo
Rimborso Completo Controller: api/rimborsi-full/		
POST	insert	Ricevuti dei valori, registra il rimborso nel sistema
GET	rimborso	Ricevuto l'identificativo di un rimborso, ne ritorna i dettagli
Campo Form Controller: api/form/		
Continua nella prossima pagina...		

Tabella 5.3 – Continuo della tabella

Metodo	Indirizzo	Descrizione
GET	get-form	Ricevuta una tipologia, restituisce un form per inserire un rimborso

Tabella 5.3: Tabella mostrante le principali *API* realizzate.

5.5 Codifica Front End

Il *frontend* è la parte dell'applicazione che gestisce la componente *View* del *design pattern MVC*. È stato realizzato usando il *framework Angular* e ne riflette la struttura a componenti. Ogni *Component* è una piccola parte dell'applicazione dotata di logica e presentazione. La logica è codificata in un file *TypeScript*, mentre la presentazione in *HTML* per la parte di contenuto, e *CSS* per la resa grafica. Tuttavia, i file *CSS* sono vuoti, poiché tutta la grafica è stata gestita mediante la libreria *Bootstrap*.
La peculiarità dei *component* è la loro riusabilità. Infatti, alcuni di questi sono presenti in ogni pagina dell'applicazione, altri sono stati riutilizzati all'interno di altri *component*.

5.5.1 Tabelle Componenti Front End

Nome Component	Descrizione
CampoForm	Mostra un singolo campo di un form.
DynamicForm	Mostra l'intero form usando il *component* CampoForm.
Filtri	Consente di inserire dei filtri per la ricerca di rimborsi.
Continua nella prossima pagina...	

Tabella 5.4 – Continuo della tabella

Nome Component	Descrizione
Footer	Gestisce la paginazione degli insiemi di rimborsi mostrati.
InsertDialog	Dialog per la richiesta di un nuovo form da compilare.
Insert	Pagina di inserimento form. Usa il componente *DynamicForm*.
ReportDialog	Dialog per la richiesta di generazione di un report.
ReportDetail	Componente che mostra il report generato.
Report	Pagina per mostrare il report. Usa il componente ReportDetail.
Header	Parte situata in alto dell'applicazione, che contiene la barra di navigazione e i *component* per la ricerca, il form e il report.
Rimborsi	Gruppo di componenti per mostrare l'insieme iniziale dei rimborsi, o un insieme filtrato di rimborsi.
RimborsoDetail	Pagina che mostra i dettagli di un rimborso. Usa il componente *DynamicForm*.
Summary	Fornisce informazioni sul numero di rimborsi, raggruppati per stato.
AppComponent	Componente principale dal quale si sviluppano gli altri.

Tabella 5.4: Tabella mostrante i *Component* realizzati

Nome gruppo classi	Descrizione
services	Insieme di classi che gestiscono le response e le request al *backend*
models	Insieme di classi che definiscono i modelli dell'applicazione. Riflettono i **DTO** del *backend*.
routing	Gestisce il *routing*$_G$ dell'applicazione.
guards	Insieme di classi che gestisce i ruoli e le funzionalità permesse.

Tabella 5.5: Tabella mostrante i gruppi di classi realizzati

5.6 Verifica e Validazione

La verifica e i vari test, ovvero il controllo che l'applicazione, ricevuta una modifica, non producesse errori, sono stati effettuati lungo tutto il periodo di sviluppo. Come spiegato nel rischio del capitolo 2, sono state adottate delle pratiche per mitigare il pericolo di generazione e propagazione di errori nel codice sorgente. Inoltre, sono stati chiesti costanti riscontri sul lavoro prodotto, il quale è stato analizzato e ridotto in tante attività atomiche e separate. Ciò ha permesso un controllo costante ed efficace sul progetto.
Gli ultimi giorni sono stati dedicati alla validazione del progetto, cioè al controllo che l'applicazione rispettasse i canoni e le aspettative del committente.
Di seguito viene presentata una tabella dei requisiti e i loro esiti:

Requisito	Descrizione	Esito
RFN-1	Il sistema deve garantire l'autenticazione automatica	Soddisfatto
Continua nella prossima pagina...		

Tabella 5.6 – Continuo della tabella

Requisito	Descrizione	Esito
RFN-2	Il sistema deve registrare in automatico i nuovi utenti	Soddisfatto
RFN-3	Il sistema deve categorizzare e riconoscere tre ruoli per l'utente: Dipendente, Amministratore e Operatore	Soddisfatto
RFN-4	Il sistema deve permettere all'utente di visualizzare la dashboard una volta autenticato	Soddisfatto
RFN-5	Il sistema deve permettere la visualizzazione di un sottoinsieme di rimborsi	Soddisfatto
RFN-6	Il sistema deve filtrare i rimborsi visualizzati per l'utente attivo, qualora questo abbia il ruolo Dipendente	Impostato
RFN-7	L'applicazione deve essere fornire un sistema di paginazione per gestire la visualizzazione dei sottoinsiemi di rimborsi	Soddisfatto
RFN-8	Il sistema deve mostrare sulla dashboard un sommario di numeri di rimborso per ogni stato	Soddisfatto
RFN-9	Ciascun utente deve essere in grado di poter visualizzare i dettagli di un rimborso	Soddisfatto
RFN-10	Ciascun utente deve poter essere in grado di tornare alla home page, dalla pagina di visualizzazione del rimborso	Soddisfatto

Continua nella prossima pagina...

Tabella 5.6 – Continuo della tabella

Requisito	Descrizione	Esito
RFN-11	Gli utenti Amministratore e Operatore devono essere in grado di aggiornare lo stato di un rimborso	Soddisfatto
RFN-12	Gli utenti Amministratore e Operatore devono essere in grado di annullare la procedura di aggiornamento lo stato di un rimborso	Soddisfatto
RFN-13	Il sistema deve essere in grado di filtrare i rimborsi visualizzati	Soddisfatto
RFN-14	Ciascun utente deve essere in grado di aggiungere filtri di ricerca	Soddisfatto
RFN-15	Ciascun utente deve essere in grado di rimuovere dei filtri di ricerca	Soddisfatto
RFN-16	Ciascun utente deve essere in grado di poter annulare la procedura di ricerca	Soddisfatto
RFN-17	L'utente Dipendente deve essere in grado di poter accedere ad una schermata per inserire un nuovo rimborso	Soddisfatto
RFN-18	L'utente Dipedente deve poter selezionare il tipo di rimborso da inserire	Soddisfatto
RFN-19	L'utente deve essere in grado di poter annullare la procedura di inserimento della tipologia di richiesta	Soddisfatto
RFN-20	Il sistema deve essere in grado di mostrare all'utente un form da compilare per inserire un nuovo rimborso	Soddisfatto

Continua nella prossima pagina...

Tabella 5.6 – Continuo della tabella

Requisito	Descrizione	Esito
RFN-21	L'utente deve poter salvare nel sistema il rimborso creato	Soddisfatto
RFN-22	L'utente deve essere in grado di annullare la procedura di inserimento di un rimborso	Soddisfatto
RFN-23	L'utente Dipendente deve poter essere in grado di tornare alla home page, dalla pagina di inserimento del rimborso	Soddisfatto
RFO-24	Il sistema deve essere in grado di notificare gli utenti con il ruolo Amministratore e Operatore dell'inserimento di un nuovo rimborso	Non Soddisfatto
RFO-25	Deve essere in grado di generare un report sulla base di un insieme di dati	Soddisfatto
RFO-26	Ciascun utente deve poter richiedere la visualizzazione di un report	Soddisfatto
RFO-27	L'utente con ruolo Amministratore o Operatore devono poter generare il report sulla base dei seguenti parametri: • Utente Dipendente • Periodo di tempo: due date	Soddisfatto
RFO-28	L'utente dipendente deve poter generare un report basato su un certo periodo di tempo	Soddisfatto
RFO-29	Ciascun utente deve poter annullare la procedura di inserimento dei parametri per la generazione di un report	Soddisfatto

Continua nella prossima pagina...

Tabella 5.6 – Continuo della tabella

Requisito	Descrizione	Esito
RFO-30	Ciascun Utente deve poter visualizzare il report generato	Soddisfatto
RFO-31	Ciascun utente deve poter tornare alla home page dalla pagina di visualizzazione report	Soddisfatto
RFN-32	Il sistema deve ridirezionare gli utenti in home page, qualora questi abbiano tentato di accedere a funzionalità non permesse dal loro ruolo	Soddisfatto
RVN-1	Il linguaggio di programmazione per il lato backend sarà Java	Soddisfatto
RVN-2	Verrà usato PostgreSQL per la gestione del database	Soddisfatto
RVO-3	Potrà essere usato Docker come container per PostgreSQL	Soddisfatto
RVN-4	Verrà usato il framework Maven per il lato backend	Soddisfatto
RVN-5	Verrà usato il framework Spring Boot per il lato backend	Soddisfatto
RVN-6	Il lato frontend sarà realizzato con i linguaggi Typescript, HTML e CSS	Soddisfatto
RVN-7	Verrà usato il framework Angular per il lato frontend	Soddisfatto
RVO-8	Potrà essere usata la libreria Bootstrap per il lato frontend	Soddisfatto

Continua nella prossima pagina...

Tabella 5.6 – Continuo della tabella

Requisito	Descrizione	Esito
RVN-9	Sia il lato backend, che il lato frontend saranno versionati tramite GitLab	Soddisfatto

Tabella 5.6: Tabella del tracciamento dei requisiti e dei loro esiti.

Ad eccezione del requisito funzionale opzionale 24, tutti i requisiti sono stati soddfisfatti. Alcuni requisiti riportano l'esito *Impostato*, questo significa che il requisito è stato soddisfatto e la funzionalità realizzata, ma è stata bloccata momentaneamente su richiesta.

5.7 Criticità, limiti e futuro dell'applicazione

Nelle fasi finali di sviluppo e collaudo dell'applicazione sono stati valutati anche alcuni aspetti migliorabili del prodotto, soprattutto dal punto di vista grafico, ed un'eventuale nuova funzionalità.
Negli ultimi giorni di stage questa nuova funzionalità è stata analizzata e sono state trovate due possibili soluzioni, le quali però non sono state realizzate per mancanza di tempo, ciò però dimostra che l'applicazione è estendibile.

Capitolo 6

Prodotto Finale

In questo capitolo verrà mostrato il prodotto finale nelle sue componenti.

6.1 Struttura dell'appliczione

L'applicazione è strutturata in quattro pagine, ciascuna di esse con funzionalità diverse.

- Home Page
- Visualizzazione dettagli rimborso
- Inserimento rimborso
- Generazione e visualizzazione report

La prima è la pagina di apertura dell'applicazione, le altre sono raggiungibili tramite la pagina principale.
Ogni pagina presenta la medesima struttura:

- *header*
- Un'intestazione
- Il contenuto della pagina

6.2 Header e Intestazione

Queste due componenti sono sempre presenti in cima ad ogni pagina.
L'*header* è composto da:

- Il titolo dell'applicazione (Note spese)
- Un link alla pagina principale (Home)
- Tre pulsanti con diverse funzionalità indicate dai loro rispettivi nomi: il primo per ottenere un report di diversi rimborsi, il secondo per filtrare i rimborsi visionabili e l'ultimo per inserire un nuovo rimborso

L'*intestazione* è composta da:

- Il titolo Dashboard
- Una lista che presenta l'elenco degli stati in cui può essere un rimborso (inserito, in revisione, revisionato, in approvazione e approvato), ciascuno seguito dal numero di rimborsi in quello stato tra quelli attualmente visionati nella Home Page.

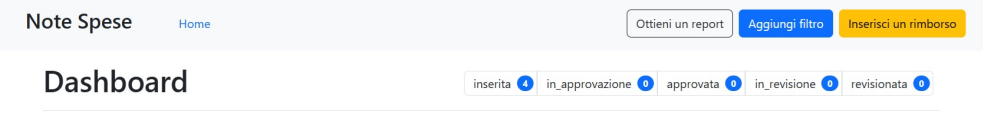


Figura 6.1: Header e Intestazione

6.2.1 Home Page

La prima pagina dell'applicazione presenta l'Header, l'Intestazione e subito sotto il suo contenuto.

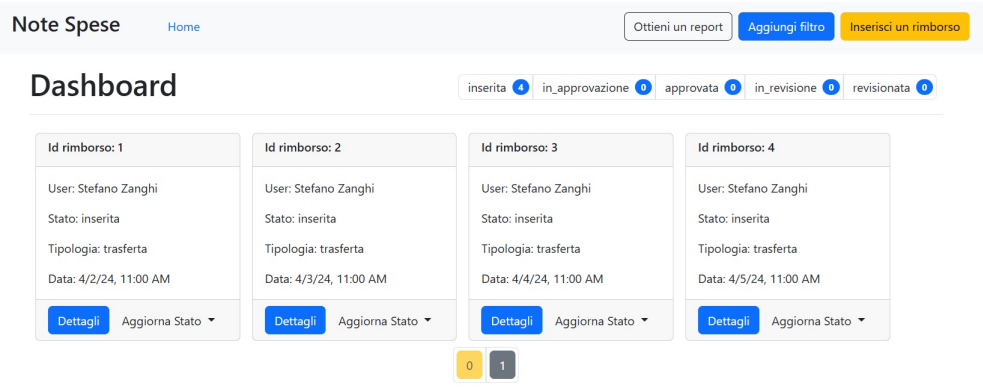


Figura 6.2: Home Page

Il contenuto della Home Page è un sottoinsieme di quattro rimborsi presenti nel database. Cliccando i pulsanti numerati in basso è possibile visionare i pros-

simi o i precedenti quattro rimborsi. Il pulsante in giallo indica il sottoinsieme corrente e perciò non è azionabile.

6.2.2 Singolo rimborso

Un singolo rimborso presenta la seguente struttura:

- Id del rimborso nel database
- Il nome dell'utente che ha richiesto il rimborso
- Lo stato del rimborso
- La tipologia del rimborso
- La data di inserimento

Infine, sono presenti due pulsanti: "*Dettagli*" e "*Aggiorna stato*". Il primo permetterà di visualizzare la pagina di dettaglio del rimborso, il secondo permetterà di cambiare lo stato del rimborso tramite un menù a tendina nel quale lo stato attuale non sarà selezionabile.

Figura 6.3: Singolo rimborso e aggiornamento stato

6.2.3 Filtrare i rimborsi

Premendo il pulsante "*Aggiungi un filtro*" presente in alto nell'Header, è possibile filtrare i rimborsi visualizzati nella home page in base a multiple condizioni.
Il pulsante apre il seguente prompt:

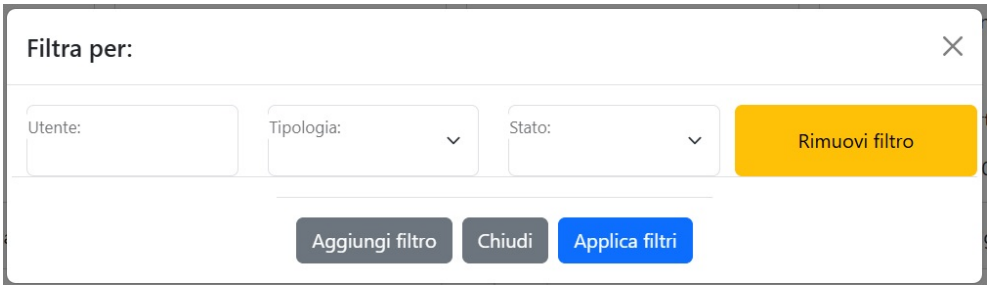


Figura 6.4: Prompt per inserire i filtri

Nessun campo deve per forza essere compilato e premendo i pulsanti "*Aggiungi filtro*" o "*Rimuovi filtro*" è possibile, appunto, aggiungere o rimuovere un filtro di ricerca.

6.2.4 Pagina di Dettaglio di un Rimborso

La pagina presenta, oltre ai consueti Header e Intestazione, i dettagli del rimborso, ossia tutti i valori specifici che sono stati inseriti durante la richiesta. Tali valori sono presentati sotto l'aspetto di form precompilato e non modificabile.

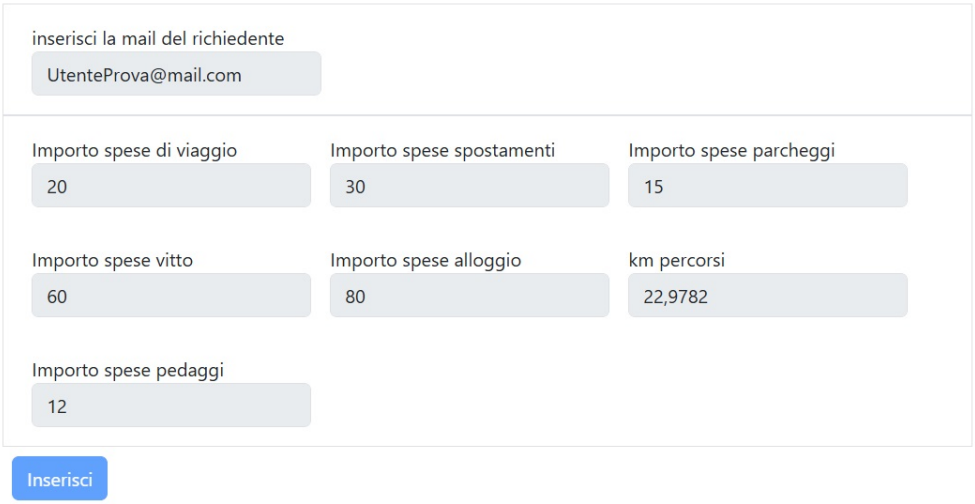


Figura 6.5: Form precompilato mostrante il dettaglio di un rimborso

6.2.5 Inserimento Rimborso

Per inserire un nuovo rimborso, l'utente dovrà cliccare sul pulsante giallo dell'header chiamato "*inserisci un rimborso*". Premendo questo pulsante, verrà

mostrato un prompt in sovraimpressione, che chiederà di scegliere una tipologia di rimborso dal menù a tendina.

Figura 6.6: Dialog inserimento

È possibile chiudere il prompt con il pulsante **X** in alto a destra, con il pulsante *chiudi* grigio in basso al centro o semplicemente cliccando all'esterno del prompt.

Figura 6.7: Menu del dialog di inserimento

Una volta selezionato il tipo di rimborso si verrà rediretti verso la pagina di inserimento rimborso. Questa conterrà il form da compilare (il seguente caso riguarda la tipologia "trasferta"):

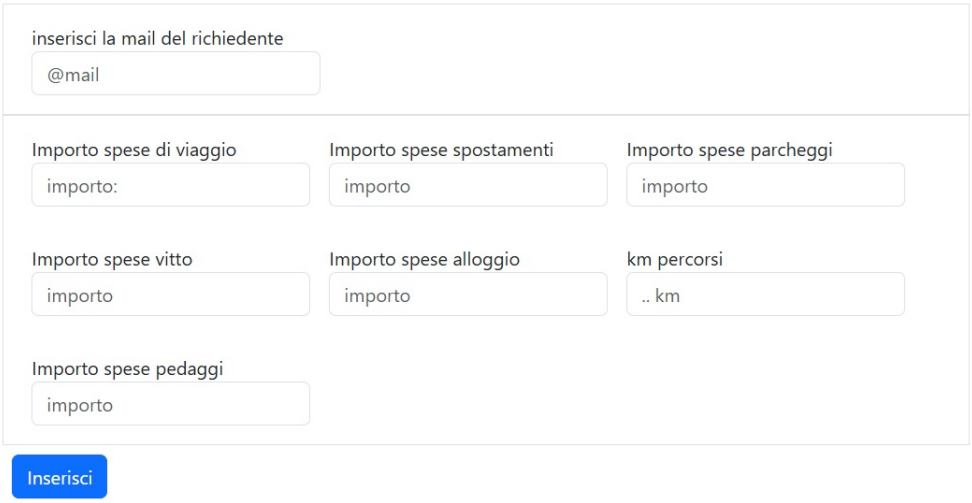


Figura 6.8: Form per inserire un rimborso

Compilando il form e premendo *inserisci*, il form verrà registrato nel database e sarà visualizzabile nella lista presente nella Home Page.

6.2.6 Report

Per ottenere un report è necessario premere il pulsante "*Ottieni un report*" in alto a sinistra nell'*Header*. Ciò farà apparire un prompt in cui verrà chiesto di scegliere un utente del quale si vuole generare un report.

Figura 6.9: Dialog scelta utente per cui generare un report

Selezionato l'utente dal menù a tendina e premuto il pulsante "*ottieni il report*" si verrà rediretti alla pagina di creazione e visualizzazione del report. Per prima cosa verrà richiesto di inserire un periodo per il quale generare il report, composto da una data di inizio del periodo e una di fine.

Figura 6.10: Scelta delle date nella pagina del report

Una volta premuto il pulsante a lato "*Genera Report*", questo verrà visualizzato nello spazio sottostante:

Figura 6.11: Report generato

Nell'esempio si vede un report generato per un utente fittizio di prova, considerati i rimborsi inseriti nel periodo "02/04/2024" $\sim$ "08/04/2024".

Capitolo 7

Conclusioni

In conclusione, lo stage è stata un'attività di sviluppo software in tutte le sue fasi: dall'analisi fino al collaudo e ai primi supporti per eventuali nuove funzionalità. L'esito è un prodotto che ha soddisfatto tutti i requisiti necessari, e non solo, trovati in sede di analisi e perciò è un prodotto pronto all'uso e che andrà ad aggiungersi a supporto dei processi aziendali amministrativi per automatizzarli.

7.1 Consuntivo finale

Di seguito viene presentato un rendiconto delle ore delle attività di stage. Nonostante questo rifletta abbastanza quanto presentato nella tabella 2.2, si può notare come la parte di sviluppo sia sbilanciata verso il *frontend*. Ciò è motivato dalla personale inesperienza nella progettazione e realizzazione di *frontend*, contro quella di *backend* maturata negli anni universitari.

Ore di lavoro	Descrizione
30	Formazione Spring, Spring Boot
30	Formazione Angular
20	Analisi dei requisiti
40	Progettazione e Struttura Database
	Continua nella prossima pagina...

Tabella 7.1 – Continuo della tabella

Ore di lavoro	Descrizione
70	Codifica *backend*
90	Codifica *frontend*
20	Test e Documentazione

Tabella 7.1: Consuntivo finale ore di lavoro

7.2 Raggiungimento degli obiettivi

Di seguito viene presentata la tabella degli obbiettivi aziendali raggiunti. Questa tabella riflette la tabella 2.1. Ad eccezione di un obbiettivo non necessario, tutti gli obbiettivi sono stati raggiunti. Viene qui riportata la classificazione degli obbiettivi aziendali:

- OB - Obbligatori
- DE - Desiderabili
- OP - Opzionabili

Classificazione	Descrizione	Esito
OB1	Progettazione del software applicativo e della base dati applicativa	Raggiunto
OB2	Realizzazione della base dati	Raggiunto
OB3	Realizzazione dell'applicativo frontend	Raggiunto
OB4	Realizzazione dei servizi API REST backend	Raggiunto

Continua nella prossima pagina...

Tabella 7.2 – Continuo della tabella

Classificazione	Descrizione	Esito
DE1	Autonomia nella ricerca e proposizione di soluzioni.	Raggiunto
DE2	Invio mail ad inserimento richiesta o richiesta di revisione	Non Raggiunto
OP1	Realizzazione sezione Dashboard Richieste	Raggiunto
OP2	Realizzazione sezione report	Raggiunto

Tabella 7.2: Obbiettivi Raggiunti

7.3 Conoscenze acquisite

Le aspettative personali esposte nel primo capitolo sono state tutte soddisfatte. Poter discutere di ciò che è stato prodotto con persone competenti ed esperte nel settore, mi ha fornito degli interessanti spunti che ho potuto integrare nei processi di sviluppo. Altre conoscenze importanti riguardano il campo delle tecnologie infatti molte delle tecnologie usate non mi erano conosciute ed è stato molto utile imparare ad utilizzare tecnologie largamente usate nel mondo del lavoro. Altra conoscenza importante acquisita riguarda il *modus operandi* aziendale, come ad esempio: la struttura e la rigidità dell'organizzazione dei codici sorgenti. Un'ultima importante conoscenza riguarda una generale visione d'insieme dell'applicazione, acquisita realizzando sia il *backend* e il *frontend*.

7.4 Valutazione personale

Uno degli aspetti più interessanti che sono stati notati durante il periodo di stage riguarda le differenze tra il mondo accademico e quello lavorativo. Se il primo risulta molto rigido in certe strutture, al fine di garantire il miglior risultato possibile, il secondo cerca la giusta via di mezzo tra efficacia ed efficienza. In conclusione, ritengo di aver imparato molto da questa esperienza e di aver raffinato le vecchie consocenze. Ovviamente le difficoltà ci sono state, ma queste sono diventate occasioni per imparare e accrescere le mie conoscenze.

Acronimi e Abbreviazioni

API Application Program Interface. i

BE backend. i

DBMS Database Management System. i

FE frontend. i

O365 Office365. i, 10

UML Unified Modeling Language. i, 9

VCS Version Control System. i

Glossario

API In informatica, un'API è un insieme di procedure disponibili ai programmatori, generalmente raggruppate per formare un toolkit per un'attività specifica all'interno di un programma. Il suo scopo è fornire un'astrazione, solitamente tra hardware e programmatore o tra software di basso e alto livello, semplificando il processo di programmazione. i, 4, 33

backend Parte del software applicativo che gestisce la logica del sistema e i dati generati dal frontend. i, 2

branch Nuova versione del prodotto nata da una precedente, separata da quest'ultima. Permette di sviluppare il prodotto senza modificare la versione di partenza, per poi eventualmente integrare le modifiche nella versione originale. i, 6

Business Logic Nucleo operativo di un'applicazione. Gestisce i dati e le loro operazioni.. i, 32

Data Transfer Object Oggetto descrittivo costruito ad hoc a partire da un modello, al fine di essere trasferito tramite chiamata HTTP. i, 33

Database Migration Tecnologia che permette il versionamento dello schema del database e dei suoi dati. Una migrazione viene effettuata quando bisogna aggiornare uno schema, o, al contrario, quando bisogna ritornare ad una versione precedente. i, 6, 29, 37

DBMS Sistema Software progettato per consentire la creazione, la manipolazione e l'interrogazione di una o più basi di dati in modo corretto ed efficiente. i, 2, 31

design pattern Modelli logici standard di varie tipologie che rispondono a specifiche esigenze progettuali. i, 32

framework Ambiente di lavoro definito da un insieme di classi astratte dalla relazione tra esse. Fornisce un'architettura logica di supporto al programmatore sulla quale un software può essere progettato e realizzato.. i, 2, 27, 28

frontend Il frontend è la parte di un sistema software che gestisce l'interazione con l'utente o con sistemi esterni che producono dati di ingresso. i, 2

GitLab Piattaforma web open source che permette il controllo della versione di un progetto. i, 2

IDE Applicazione che fornisce vari strumenti utili per lo sviluppo software. Tipicamente include: un editor testuale per il codice, un processo di automazione per lo sviluppo, un compilatore e un debugger. i, 24

Office 365 Servizio in abbonamento venduto da Microsoft. i, 6, 10

open source software distribuito, spesso in via gratuita, sotto una licenza che permette lo studio, l'utilizzo, la modifica e la redistribuzione. i, 23, 24

package Strumento tipico del linguaggio di programmazione Java, usato per raggruppare e organizzare classi in gruppi logici. i, 37

Presentation Logic Nucleo di presentazione di un'applicazione. Gestisce la presentazione e la resa grafica dei dati, oltre che all'interazione con l'utente finale. i, 32

repository Git Una cartella pubblicata sulle piattaforme Git (GitLab o GitHub) contenente un progetto versionato. i, 31

responsiveness Capacità di un'applicazione di rispondere e riorganizzare il contenuto in base alle dimensioni dello schermo. i, 30

REST Representational State Transfer. Stile architetturale per sistemi di trasmissione di dati su protocollo HTTP. Il dato tramesso è una rappresentazione dello stato del mittente che può essere ricostruita dal destinatario. i, 8, 39

routing Funzione che permette di gestire gli indirizzi dell'applicazione e delle sue varie pagine. i, 44

VCS I sistemi di controllo delle versioni sono software che aiutano i team di sviluppo a gestire i cambiamenti nel codice. i, 6

WebApp Software applicativo che viene eseguito su un server web ed è accessibile attraverso un browser. L'accesso risulta più flessibile perché disponibile su diverse piattaforme. i, 2

Sitografia

Angular. URL: https://angular.dev/.

Bootstrap. URL: https://getbootstrap.com/.

DBeaver. URL: https://dbeaver.io/.

Docker. URL: https://www.docker.com/.

draw.io. URL: http://drawio.com.

IntelliJ IDEA. URL: https://www.jetbrains.com/idea/.

Java. URL: https://www.java.com/it/.

Maven. URL: https://maven.apache.org/.

PostgreSQL. URL: https://www.postgresql.org/.

Spring, Spring Boot. URL: https://spring.io/.

TypeScript. URL: https://www.typescriptlang.org/.