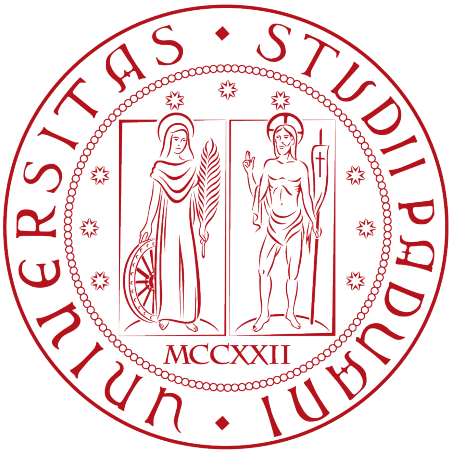
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Sviluppo componenti back end di una applicazione web per viaggi condivisi

Tesi di laurea

Relatore

Prof. Claudio Enrico Palazzi

Laureando

Luca Calabrese

Anno Accademico 2022-2023

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage dal 10/07/2023 al 01/09/2023, della durata di circa trecento ore, dal laureando Luca Calabrese presso l'azienda Sync Lab S.p.A. Gli obbiettivi da raggiungere erano molteplici.
In primo luogo era richiesto lo studio teorico dell'architettura a microservizi andando ad analizzare pro e contro e confrontandola con l'architettura monolitica. In secondo luogo era richiesto lo sviluppo di alcuni componenti del Back end di una web app tramite l'utilizzo del linguaggio JAVA e del framework Spring. Tale framework permette sviluppare servizi Back end consentendo la gestione delle dipendenze e lo sviluppo di servizi web. Terzo ed ultimo obbiettivo era la fase di integrazione con le altre parti del beck-end e con il Front end.

Ringraziamenti

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

Desidero ringraziare con affetto i miei genitori Leonardo e Manuela e tutta la mia famiglia 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, Dicembre 2023 Luca Calabrese

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Introduzione al contesto applicativo.

Esempio di utilizzo di un termine nel glossario
Application Program Interface (API).

Esempio di citazione in linea
Manifesto Agile. URL: https://agilemanifesto.org/iso/it/manifesto.html.

1.1 L'azienda

Sync Lab nasce nel 2002 a Napoli come *software house* e si è rapidamente tramutata in un *System Integrator* grazie ad una maturazione delle competenzeecnologiche, metodologiche ed applicative nel dominio del *software*. L'azienda, offre servizi proprietari sviluppati nei propri laboratori di ricerca e sviluppo che coprono gli ambiti di: mobile, videosorveglianza, sicurezza delle infrastrutture informatiche aziendali, metaverso e blockchain. Ora Sync Lab è una azienda con più di 300 dipendenti e 6 sedi in tutta Italia.

Figura 1.1: Logo Sync Lab

Il loro obiettivo è quello di aiutare i clienti nella realizzazione, passaggio in produzione e governance di soluzioni IT, offrendo le nostre competenze tecnologiche l'esperienza necessaria legata ai cambiamenti organizzativi che ne conseguono

1.2 L'idea

L'idea nasce con l'obiettivo di ridurre il traffico stradale attraverso il car sharing. Nonostante la presenza di molte app di car sharing il problema della congesione del traffico per eventi o località turistiche rimane ancora presente. L'idea è provare a proporre un prodotto il quale oltre a occuparsi dell'aspetto di car sharing si occupi anche del far condividere un'esperienza ai membri del viaggio ispirandosi all'hippie trail https://it.wikipedia.org/wiki/Hippie_trail(21/11/2023). L'hippie trail era una sorta di turismo alternativo in voga tra gli anni cinquanta e la fine anni settanta il quale si sviluppava nell'ottica di viaggiare nel modo più economico possibile e di estendere la durata del viaggio. Il metodo consisteva nel viaggiare tramite autostop ed effettuare il viaggio in più tappe fermandosi in ostelli o dove capitava lungo il tragitto. L'applicazione quindi nasce con l'intenzione di gestire sia la parte di car sharing fornendo agli utenti la possibilità di organizzare viaggi condivisi tramite car sharing, che la parte relativa alle attività da svolgere durante il viaggio nel pieno stile del viaggio triphippie che è quello di non solo condividere il viaggio ma anche di conoscere persone e provare una nuova esperienza. Il pensiero dietro l'applicazione è anche quello di permettere ai viaggiatori di selezionare il viaggio che più rispecchi le loro aspettative permettendogli di vedere oltre alle mete del viaggio anche quali attività chi ha organizzato il viaggio pensa di fare. Dopo il viaggio viene anche permesso agli utenti di lasciare recensioni sia sugli altri partecipanti che su il viaggio in generale.

1.3 Organizzazione del testo

Il secondo capitolo descrive lo stage dando una visione generale dell'azienda e di quello che ho fatto durante i due mesi di tirocinio.

Il terzo capitolo contiene la lista degli obiettivi fissati e il loro grado di importanza e completamento.

Il quarto capitolo approfondisce le metodologie e le tecnologie usate per sviluppare il progetto.

Il quinto capitolo descrive i servizi sviluppati e l'architettura utilizzata.

Il sesto capitolo approfondisce l'analisi dei requisiti riguardante la parte di progetto trattata durante il tirocinio.

Il settimo capitolo approfondisce come si è svolta la parte di progettazione e codifica.

Nel ottavo capitolo contiene le conclusioni riportate a fine tirocionio e una autovalutazione.

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: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Descrizione dello stage

2.1 Introduzione al progetto

Il progetto TripHippie consiste nel sviluppare una web app che permentta di gestire sia il car sharing sia l'experience sharing.
In particolare l'applicazione deve gestire:

- La registrazione e login degli utenti;
- L'inserimento di nuovi viaggi;
- L'invito o la richiesta di partecipare a un viaggio;
- La gestione del viaggio;
- Le recensioni del viaggio;
- Le recensioni ai partecipanti del viaggio;
- La ricerca di un viaggio.

2.2 Descrizione del team di lavoro

Il team che lavorava al progetto era diviso in tre gruppi: Back end, Front end e security. Ad ogni membro del team era assegnato lo sviluppo di una specifica parte dell'applicazione e lavorava in un proprio ambiente separato. All'inizio erano presenti solo sviluppatori Back end i quali hanno potuto così iniziare a implementare gli API endpoint necessari in seguito ha iniziato a lavorare anche il team security il quale si è occupato dello sviluppo della parte di autenticazione delle richieste e infine il team di sviluppo del Front end.
Settimanalmente venivano eseguite delle riunioni di avanzamento dei lavori per coordinare i vari membri del progetto e discutere gli sviluppi da implementare nella settimana successiva.

2.3 Descrizione metodo di lavoro

Durante il periodo di stage si è deciso di adottare un approccio agile, utilizzando una metodologia scrum insieme all'uso di un modello incrementale. Di conseguenza il lavoro è stato suddiviso in task e ogni task è stato assegnato ad un ciclo di produzione di durata settimanale detto sprint.
Per prima cosa sono state definite della fasi dal tutor aziendale, in segnuito ogni fase è stata assegnata ad un numero congruo di sprint e poi per ogni fase si sono create delle task ed inserite nel backlog. Si è adottata questa metodologia di lavoro perchè permette un controllo continuo degli stati di avanzamento del progetto.

2.3.1 Metodi comunicativi

Ogni settimana, abbiamo organizzato riunioni di avanzamento Scrum individuali, durante le quali venivano esaminati gli sviluppi compiuti, affrontato eventuali problemi riscontrati e interagito con il tutor per approfondire gli argomenti trattati. La partecipazione attiva del tutor ha fornito un prezioso supporto nel superare le sfide e nell'ottimizzare i progressi del progetto.
In aggiunta a queste riunioni individuali, è stata pianificata una sessione settimanale con l'intero team, volta a discutere complessivamente dello stato di avanzamento del progetto. Questo approccio ci ha consentito di sincronizzare gli sforzi, condividere conoscenze e favorire una visione d'insieme del lavoro svolto.
Per facilitare la comunicazione tra i membri del gruppo, l'azienda ha fornito l'accesso al proprio server Discord nel quale erano disponibi alcune risorse per la formazione condivise dai dipendenti. Qui, sono stati creati canali dedicati per gli stagisti e per il progetto, offrendo un ambiente in cui poter condividere rapidamente informazioni, risolvere dubbi e mantenere una comunicazione. Questa piattaforma è diventata un punto di incontro virtuale per il team, facilitando la collaborazione e il coordinamento nelle fasi di sviluppo del progetto.

2.4 Suddivisione del lavoro

- 120 ore: Studio individuale dei vari argomenti necessari allo sviluppo del progetto
- 120 ore: sviluppo dell'applicativo
- 80 ore: integrazione con gli altri componenti dell'applicativo e collaudo.

Capitolo 3

Obiettivi

3.1 Notazione

Si farà riferimento ai requisiti seguendo la seguente annotazione:

- O per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- D per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- F per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

3.2 Obiettivi fissati

- Obbligato ri
  - O01: Acquisizione competenze sul framework Spring;
  - O02: Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il cronoprogramma;
  - O03: Portare a termine l'implementazione dei microservizi richiesti con una percentuale di superamento pari a 80.
- Desiderabili
  - D01: Portare a termine l'implementazione dei microservizi richiesti con una percentuale di superamento pari a 100.
- Facoltativi
  - F01: Implementare anche il servizio per la gestione viaggio effettivo.

Capitolo 4

Tecnologie e metodologie

4.1 Tecnologie

4.1.1 Java

Java è un linguaggio di programmazione ad alto livello orientato agli oggetti.
La caratteristica principale di Java è la capacità pre-compilare il codice una volta sola e poterlo eseguire su diverse piattaforme grazie alla Java Virtual Machine.

4.1.2 Spring

Spring è un framework utilizzato per la programmazione di servizi web in Java. Le sue caratteristiche sono Inversion Of Control e gestione della Dependency Injection. Il framework comprende una vasta scelta di librerie. Grazie alla sua flessibilità e al suo amplio ecosistema Spring è ampliamente usato nella programmazione Java.

4.1.3 PostgreSql

PostgreSql è un open source relational database management system, permette di creare e gestire basi di dati in modo flessibile, affidabile e in conformità con gli standard. PostgreSql supporta anche tipi avanzati di dati, questo database è una scelta diffusa per basi di dati che richiedono scalabilità e integrità.

4.1.4 Maven

Apache Maven è uno strumento di automazione di build principalmente usato su progetti Java. Maven si occupa di gestire le dipendenze e di eseguuire la build del progetto in modo semi automatizzato.

4.1.5 GitHub

GitHub è una piattaforma di hosting di tipo collaborativo in cui molti programmatori condividono i propri progetti. La piattaforma permette di creare e condividere le proprie repository, di tenere traccia dellle varie versioni e di creare o gestire le issue sui progetti.

4.1.6 IntelliJ IDEA

IntelliJ è un IDE per lo sviluppo in Java, si occupa di semplificare lo sviluppo tramite: un'interfaccia grafica con uno stile che aiuta a comprendere il codice, funzioni per la generazione automatica di codice e formattazione del codice.

Capitolo 5

Descrizione dei servizi

5.1 Architettura a microservizi

L'architettura a microservizi è un approccio progettuale per lo sviluppo di applicazioni che si basa sulla suddivisione dell'applicazione in più servizi autonomi. Ogni microservizio rappresenta una funzionalità distinta e comunica con gli altri servizi utilizzando interfacce definite, solitamente i servizi sono API. Vantaggi:

- Scalabilità selettiva: grazie alla suddivisione in più parti dell'applicazione è possibile decidere di far scalare solo un servizio.
- Resistenza agli errori: l'isolamento tra microservizi consente che in caso di errore non si propaghi e inoltre consente una più facile individuazione dell'errore.
- Flessibilità tecnologica: essendo i microservizi isolati tra di loro è possibile sviluppare servizi diversi con tecnologie diverse l'unica cosa importante è mantenere delle interfacce definite.
- Migliore gestionde della complessità: la suddivisione dei microservizi produce una riduzione della complessità dell'applicazione rendendola così più facile da sviluppare e più facile da manuntenere.

Svantaggi:

- Complessità nella gestione: la gestione di un elevato numero di microservizi può diventare complessa, richiedendo strumenti e processi efficaci per il cordinamento e la gestione.
- Comunicazione e latenza: la comunicazione tra servizi può generare della latenza nelle risposte dell'applicazione e può creare sovraccarico di rete.
- Sicurezza: suddividendo l'applicazione in più servizi è necessario mettere in sicurezza ogni servizio.
- Consistenza dei dati: avendo più servizi che lavorano parrallelamente sui dati si possono creare incoerenze
- Costi iniziali e complessità nella progettazione: la progettazione di una architettura a microservizi è molto più complessa di quella di una architettura monolitica

5.1.1 Come passare da monolite a microservizi

Per passare da un'architettura monolitica a una a microservizi bisogna analizzare i campi di dominio dell'applicazione, dividendo il monolite in vari domini i quali rappresentano insieme di funzioni a se stanti.
Questa divisione delle funzioni è la parte più complessa nello strutturare un'architettura a microservizi in quanto più si rendono atomici i servizi meglio è per la complessità e per favorire il lavori in parallelo tuttavia la maggiore frammentazione può portare anche un rischio di overhead delle comunicazioni tra servizi causato dalla troppa dipendenza dei servizi tra di loro. Oltre al problema dell'overhead si riscontra anche un problema riguardante la consistenza dei dati in quanto se i servizi sono troppo frammentati si avranno più servizi che operano sugli stessi dati e questo porterà a generare inconsistenza.
Il passaggio da architettura monolitica a microservizi è un processo molto complesso e dispendioso nella fase iniziale tuttavia si è visto che questa pratica porta a notevoli vantaggi nella fase di svilippo e manutenzione del codice.

5.1.2 Descrizione del servizio user

Il servizio user si occupa di offrire le seguenti funzioni offerte agli utenti:

- Registrazione
- Login
- Modifica dei campi presenti nel proprio profilo
- Cancellazione utente
- Mostrare le informazioni di un determinato utente

5.1.3 Descrizione del servizio travel

Il servizio travel si occupa di offrire le seguenti funzionalità:

- Aggiungi viaggio
- Cancella viaggio
- Fai partire un viaggio
- Invita un utente a un viaggio
- Chiedi di partecipare a un viaggio
- Rispondi a un invito
- Rispondi a una richiesta di partecipare
- Mostra le informazioni di un viaggio
- Mostra una lista di viaggi
- Mostra tutti gli inviti di uno user
- Mostra i partecipanti a un viaggio

Capitolo 6

Analisi dei requisiti

6.1 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) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

6.2 Attori

Ci sono tre tipologie di attori:

Utente non auteticato: è un utente generico che entra nella web app senza essersi autenticato;

Utente autenticato: è un utente che ha effettuato l'autenticazione all'interno della web app;

Utente proprietario del viaggio: è un utente che ha effettuato l'autenticazioen e ha creato il viaggio.

Figura 6.1: Utenti travel

6.3 Casi d'uso - servizio user

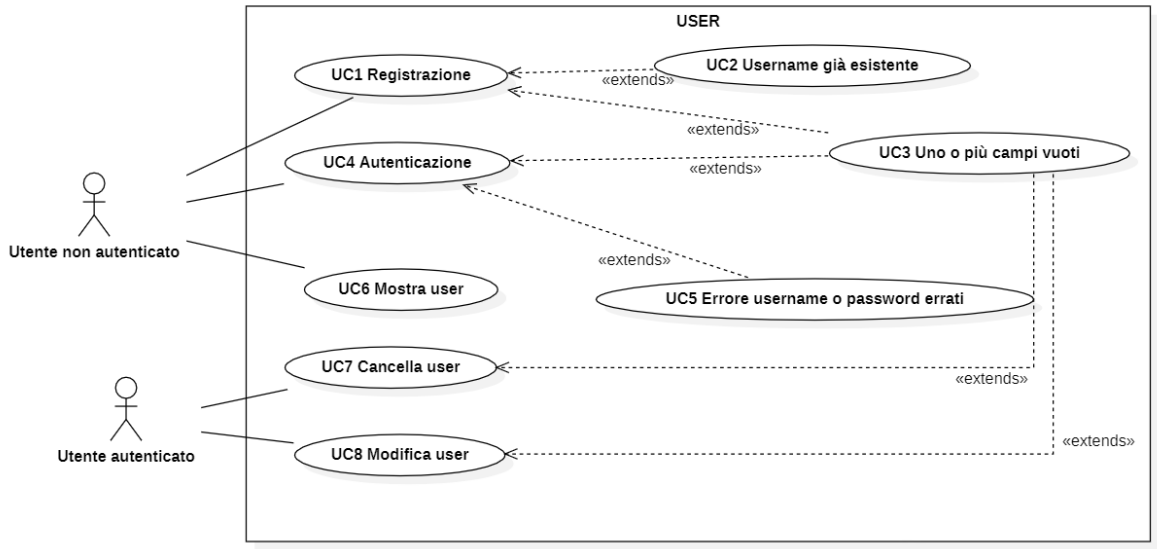


Figura 6.2: Use Case: User

UC1: Registrazione

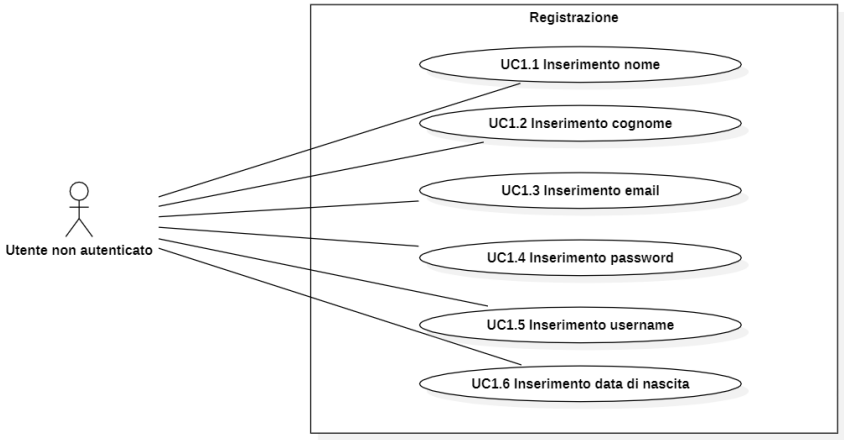


Figura 6.3: Use Case - UC1: Registrazione

Attori Principali: Utente non autenticato.
Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.
Descrizione: L'utente vuole registrarsi sull'applicazione web.
Postcondizioni: L'utente è registrato e autenticato sull'applicazione web.
Scenario Principale:

- L'utente inserisce il suo nome (UC1.1);
- L'utente inserisce il suo cognome (UC1.2);
- L'utente inserisce la sua email (UC1.3);
- L'utente inserisce una password (UC1.4);
- L'utente inserisce uno username (UC1.5);
- L'utente inserisce la sua data di nascita (UC1.6).

Estensioni:

- Username già esistente (UC2);
- Uno o più campi vuoti (UC3).

UC1.1: Inserimento nome

Attori Principali: Utente non autenticato.
Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.
Descrizione: L'utente deve inserire il suo nome per registrarsi.
Postcondizioni: L'utente ha inserito il suo nome.
Scenario Principale:

- L'utente inserisce il suo nome.

UC1.2: Inserimento cognome

Attori Principali: Utente non autenticato.
Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.
Descrizione: L'utente deve inserire il suo cognome per registrarsi.
Postcondizioni: L'utente ha inserito il suo cognome.
Scenario Principale:

- L'utente inserisce il suo cognome.

UC1.3: Inserimento email

Attori Principali: Utente non autenticato.
Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.
Descrizione: L'utente deve inserire la sua email per registrarsi.
Postcondizioni: L'utente ha inserito la sua email.
Scenario Principale:

- L'utente inserisce la sua email.

UC1.4: Inserimento password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la sua password per registrarsi.

Postcondizioni: L'utente ha inserito la sua password.

Scenario Principale:

- L'utente inserisce la sua password.

UC1.5: Inserimento username

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire il suo username per registrarsi.

Postcondizioni: L'utente ha inserito il suo username.

Scenario Principale:

- L'utente inserisce il suo username.

UC1.6: Inserimento data di nascita

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la sua data di nascita per registrarsi.

Postcondizioni: L'utente ha inserito la sua data di nascita.

Scenario Principale:

- L'utente inserisce la sua data di nascita.

UC2: Username già esistente

Attori Principali: Utente non autenticato.

Precondizioni: L'utente sta cercando di registrarsi all'applicazione web.

Descrizione: L'utente vuole registrarsi ma inserisce uno username già esistente.

Postcondizioni: L'utente visualizza un errore che lo avvisa che l'username è già esistente.

Scenario Principale:

- L'utente cerca di registrarsi con uno username già esistente;
- Il sistema avvisa l'utente che l'email è già utilizzata.

UC3: Uno o più campi vuoti

Attori Principali: Utente non autenticato.

Precondizioni: L'utente sta cercando di registrarsi all'applicazione web.

Descrizione: L'utente vuole registrarsi ma non inserisce almeno un campo.

Postcondizioni: L'utente visualizza un errore che lo avvisa non ha compilato tutti i campi.

Scenario Principale:

- L'utente cerca di registrarsi senza compilare tutti i campi;
- Il sistema avvisa l'utente che non ha compilato tutti i campi.

UC4: Autenticazione

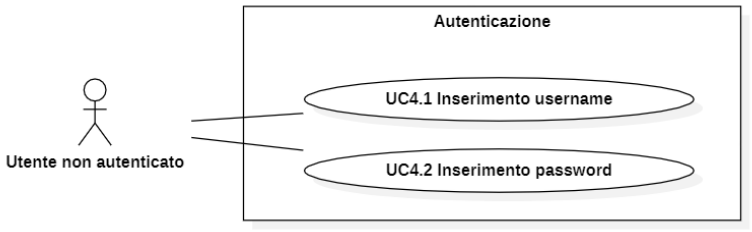


Figura 6.4: Use Case - UC1: Registrazione

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente vuole autenticarsi sull'applicazione web.

Postcondizioni: L'utente è autenticato sull'applicazione web.

Scenario Principale:

- L'utente inserisce il suo username (UC4.1);
- L'utente inserisce la sua password (UC4.2).

Estensioni:

- Visualizzazione errore username o password errati (UC5);
- Uno o più campi vuoti (UC3).

UC4.1: Inserimento username

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire il suo username per autenticarsi.

Postcondizioni: L'utente ha inserito il suo username.

Scenario Principale:

- L'utente inserisce il suo username.

UC4.2: Inserimento password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la sua password per autenticarsi.

Postcondizioni: L'utente ha inserito la sua password.

Scenario Principale:

- L'utente inserisce la sua password.

UC5: Errore username o password errati

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente vuole autenticarsi ma inserisce uno username e una password che non coincidono con nessun utente registrato.

Postcondizioni: L'utente visualizza un errore che lo avvisa che l'autenticazione è fallita.

Scenario Principale:

- L'utente cerca di autenticarsi con dati errati;
- Il sistema avvisa l'utente mostrando un errore di autenticazione fallita.

UC6: Mostra user

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non vede le informazioni di uno user.

Descrizione: L'utente vuole vedere le informazioni di uno user.

Postcondizioni: L'utente visualizza le informazioni di uno user.

Scenario Principale:

- L'utente seleziona uno user che vuole vedere;
- Il sistema mostra all'utente le informazioni dello user.

UC7: Cancella user

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e ha fatto l'autenticazione.

Descrizione: L'utente vuole cancellare il proprio account.

Postcondizioni: L'utente ha cancellato il proprio account.

Scenario Principale:

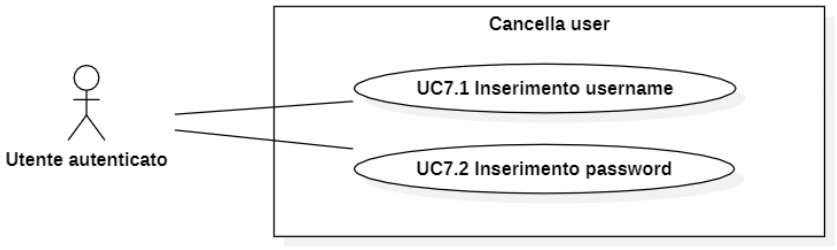


Figura 6.5: Use Case - UC7: Cancellazione

- L'utente inserisce il proprio username (UC7.1);
- L'utente inserisce la propria password (UC7.2).

UC7.1: Inserimento username

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e vuole cancellare il proprio account.

Descrizione: L'utente deve inserire il suo username cancellare l'account.

Postcondizioni: L'utente ha inserito il suo username.

Scenario Principale:

- L'utente inserisce il suo username.

UC7.2: Inserimento password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e vuole cancellare il proprio account.

Descrizione: L'utente deve inserire la sua password cancellare l'account.

Postcondizioni: L'utente ha inserito la sua password.

Scenario Principale:

- L'utente inserisce la sua password.

6.4 Casi d'uso - servizio viaggi

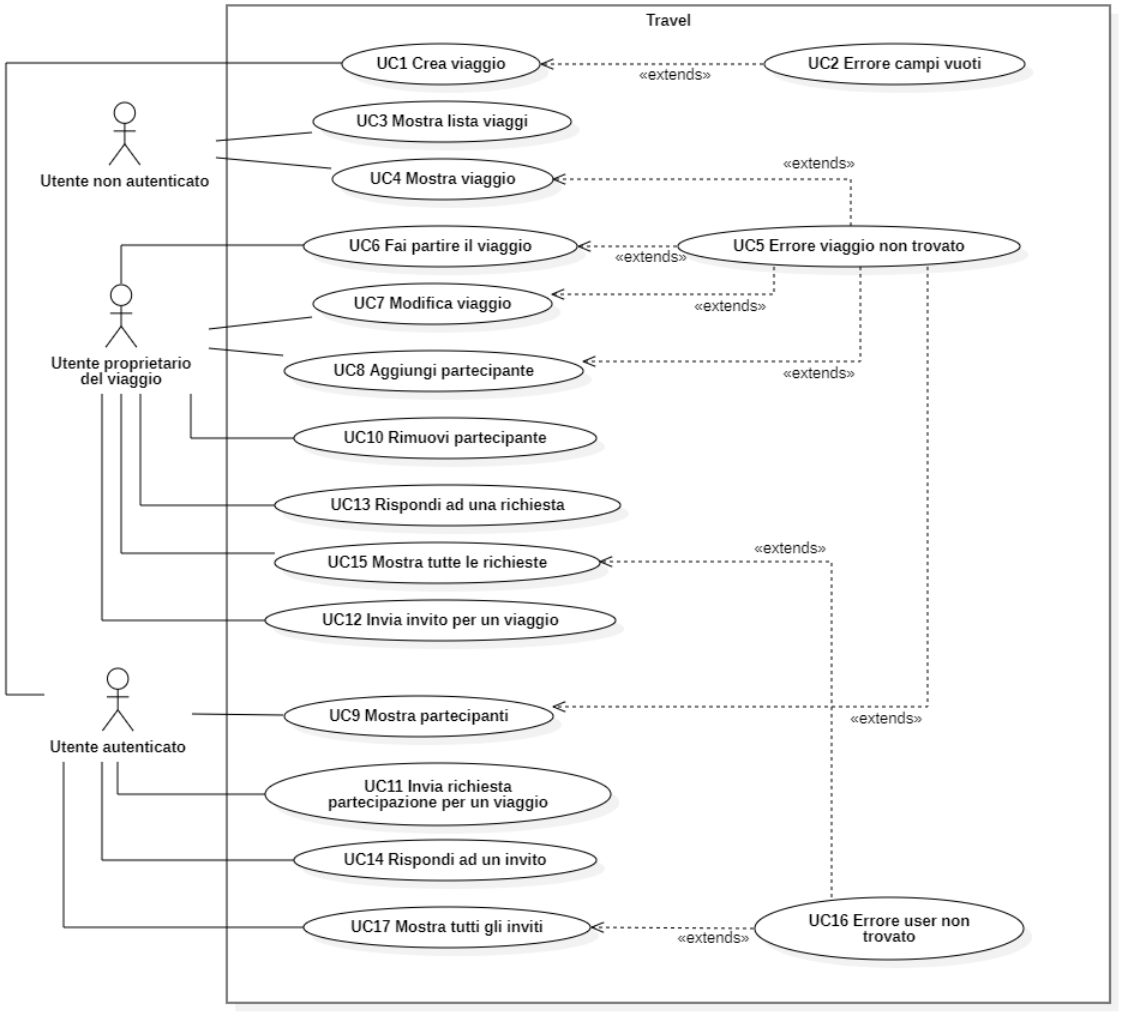


Figura 6.6: Use Case - Travel

UC1: Crea viaggio

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non c'è il suo viaggio.

Descrizione: L'utente vuole creare il suo viaggio.

Postcondizioni: L'utente ha creato il suo viaggio.

Scenario Principale:

- L'utente entra in crea viaggio;
- L'utente inserisce il luogo di partenza;

- L'utente inserisce la destinazione;
- L'utente inserisce la descrizione;
- L'utente inserisce la data di partenza;
- L'utente seleziona i vari flag per indicare quali esperienze sono previste durante il viaggio (cena, bere qualcosa ecc.).

Estensioni:

- Errore campi vuori (UC2).

UC2: Errore campi vuoti

Attori Principali: Utente autenticato.

Precondizioni: L'utente sta creando il suo viaggio.

Descrizione: L'utente vuole creare il suo viaggio lasciando dei campi vuoti.

Postcondizioni: Il sistema manda un errore all'utente dicendo che non ha compilato tutti i campi.

Scenario Principale:

- L'utente prova a creare un viaggio lasciando dei campi vuoti;
- Il sistema manda un errore all'utente avvisandolo che alcuni campi sono vuoti.

UC3: Mostra lista viaggi

Attori Principali: Utente non autenticato.

Precondizioni: L'utente non vede i viaggi disponibili.

Descrizione: L'utente vuole vedere i viaggi disponibili.

Postcondizioni: L'utente vede i viaggi disponibili.

Scenario Principale:

- L'utente entra nella sezione lista viaggi disponibili;
- L'utente visualizza la lista dei viaggi disponibili.

UC4: Mostra viaggio

Attori Principali: Utente non autenticato.

Precondizioni: L'utente non vede le informazioni complete di un viaggio.

Descrizione: L'utente vuole vedere i viaggi disponibili.

Postcondizioni: L'utente vede i viaggi disponibili.

Scenario Principale:

- L'utente seleziona il viaggio che vuole vedere;
- L'utente visualizza tutte le informazioni del viaggio.

Estensioni:

- Errore viaggio non trovato (UC5).

UC5: Errore viaggio non trovato

Attori Principali: Utente generico.

Precondizioni: L'utente sta selezionando un viaggio.

Descrizione: L'utente l'utente vuole selezionare un viaggio che non esiste.

Postcondizioni: Il sistema mostra all'utente un errore per avvisarlo che il viaggio selezionato non esiste.

Scenario Principale:

- L'utente sta selezionando un viaggio;
- Il sistema mostra all'utente un errore per avvisarlo che il viaggio selezionato non esiste.

UC6: Fai partire il viaggio

Attori Principali: Utente proprietario del viaggio.

Precondizioni: L'utente ha creato un viaggio.

Descrizione: L'utente vuole far partire il viaggio.

Postcondizioni: L'utente ha fatto partire il viaggio.

Scenario Principale:

- L'utente seleziona il viaggio che vuole far partire;
- L'utente fa partire il viaggio.

Estensioni:

- Errore viaggio non trovato (UC5).

UC7: Modifica viaggio

Attori Principali: Utente proprietario del viaggio.

Precondizioni: L'utente vede un viaggio che ha creato.

Descrizione: L'utente vuole modificare il viaggio.

Postcondizioni: L'utente modifica il viaggio.

Scenario Principale:

- L'utente seleziona il viaggio che vuole modificare;
- L'utente visualizza tutte le informazioni del viaggio;
- L'utente decide quali campi del viaggio modificare;
- L'utente modifica il viaggio.

Estensioni:

- Errore viaggio non trovato (UC5).

UC8: Aggiungi partecipante

Attori Principali: Utente proprietario del viaggio.
Precondizioni: L'utente ha creato un viaggio.
Descrizione: L'utente vuole aggiungere un partecipante al viaggio.
Postcondizioni: L'utente ha aggiunto un partecipante al viaggio.
Scenario Principale:

- L'utente seleziona il viaggio a cui vuole aggiungere un partecipante;
- L'utente seleziona il partecipante che vuole aggiungere;
- L'utente aggiunge il partecipante al viaggio.

Estensioni:

- Errore viaggio non trovato (UC5).

UC9: Mostra partecipanti

Attori Principali: Utente autenticato.
Precondizioni: L'utente sta guardando un viaggio.
Descrizione: L'utente vuole vedere i partecipanti al viaggio.
Postcondizioni: L'utente vede i partecipanti al viaggio.
Scenario Principale:

- L'utente sta guardando un viaggio;
- L'utente seleziona mostra partecipanti;
- L'utente vede una lista di partecipanti;
- L'utente visualizza il singolo partecipante UC6.

Estensioni:

- Errore viaggio non trovato (UC5) .

UC10: Rimuovi partecipante

Attori Principali: Utente proprietario del viaggio.
Precondizioni: L'utente sta guardando i partecipanti al viaggio.
Descrizione: L'utente vuole rimuovere un partecipante dal viaggio.
Postcondizioni: L'utente ha aggiunto un partecipante al viaggio.
Scenario Principale:

- L'utente seleziona il viaggio che vuole far partire;
- L'utente seleziona il partecipante che vuole rimuovere;
- L'utente rimuove il partecipante dal viaggio.

Estensioni:

UC11: Invia richiesta partecipazione al viaggio

Attori Principali: Utente autenticato.
Precondizioni: L'utente sta guardando un viaggio.
Descrizione: L'utente vuole partecipare a un viaggio.
Postcondizioni: L'utente ha inviato la richiesta di partecipazione al viaggio.
Scenario Principale:

- L'utente seleziona il viaggio che vuole a cui vuole partecipare;
- L'utente invia la richiesta di partecipazione a un viaggio;

UC12: Invia invito per un viaggio

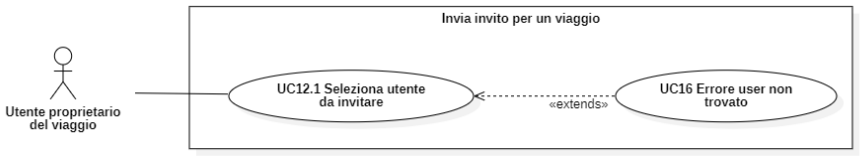


Figura 6.7: Use Case - UC12 Invia invito per un viaggio

Attori Principali: Utente autenticato.
Precondizioni: L'utente sta guardando un viaggio.
Descrizione: L'utente vuole invitare un partecipante dal viaggio.
Postcondizioni: L'utente ha inviato un invito ad un utente.
Scenario Principale:

- L'utente seleziona il viaggio a cui vuole invitare un utente;
- L'utente cerca e seleziona un utente;
- L'utente invia l'invito di partecipazione all'utente selezionato (UC12.1).

UC12.1: Seleziona utente da invitare

Attori Principali: Utente autenticato.
Precondizioni: L'utente sta invitando un altro utente ad un viaggio.
Descrizione: L'utente vuole selezionare un utente.
Postcondizioni: L'utente ha selezionato un utente.
Scenario Principale:

- L'utente sta invitando un utente ad un viaggio;
- L'utente cerca l'utente tramite username;
- L'utente seleziona l'utente che gli interessa.

Estensioni:

- Errore user non trovato (UC16).

UC13: Rispondi a una richiesta

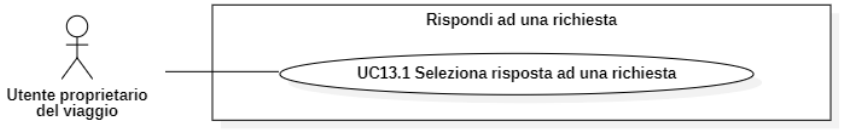


Figura 6.8: Use Case - UC13 Rispondi a una richiesta

Attori Principali: Utente proprietario del viaggio.
Precondizioni: L'utente sta controllando le richieste ricevute per i propri viaggi.
Descrizione: L'utente vuole rispondere ad una richiesta.
Postcondizioni: L'utente risponde ad una richiesta.
Scenario Principale:

- L'utente sta controllando le richieste ricevute;
- L'utente seleziona la risposta alla richiesta (UC13.1);
- L'utente risponde alla richiesta.

UC13.1: seleziona risposta ad una richiesta

Attori Principali: Utente proprietario del viaggio.
Precondizioni: L'utente sta rispondendo ad una richiesta.
Descrizione: L'utente vuole selezionare una risposta.
Postcondizioni: L'utente seleziona una risposta.
Scenario Principale:

- L'utente visualizza una richiesta;
- L'utente seleziona se accettare o rifiutare la richiesta.

UC14: Rispondi a un invito

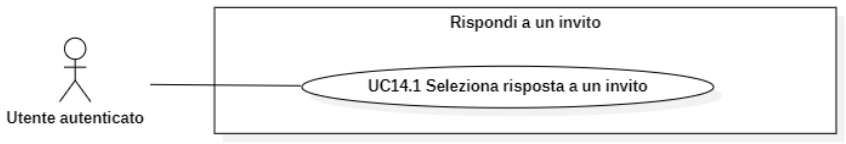


Figura 6.9: Use Case - UC14 Rispondi a un invito

Attori Principali: Utente autenticato.
Precondizioni: L'utente sta controllando gli inviti ricevuti.
Descrizione: L'utente vuole rispondere ad un invito.
Postcondizioni: L'utente risponde ad un invito.
Scenario Principale:

- L'utente sta controllando gli inviti ricevute;
- L'utente seleziona la risposta a un invito (UC14.1);
- L'utente risponde all'invito.

UC14.1: Seleziona risposta a un invito

Attori Principali: Utente proprietario del viaggio.
Precondizioni: L'utente sta rispondendo ad un invito.
Descrizione: L'utente vuole selezionare una risposta.
Postcondizioni: L'utente seleziona una risposta.
Scenario Principale:

- L'utente visualizza un invito;
- L'utente seleziona se accettare o rifiutare l'invito.

UC15: Mostra tutte le richieste

Attori Principali: Utente proprietario del viaggio.

Precondizioni: L'utente è entrato nell'applicazione web.

Descrizione: L'utente vuole vedere tutte le richieste inviate.

Postcondizioni: L'utente visualizza tutte le richieste inviate.

Scenario Principale:

- L'utente visualizza l'applicazione web;
- L'utente clicca su mostra le richieste;
- L'utente visualizza le richieste ricevute.

Estensioni:

- Errore user non trovato (UC16)

UC16: Errore user non trovato

Attori Principali: Utente autenticato.

Precondizioni: L'utente sta selezionando un altro utente.

Descrizione: L'utente vuole selezionare un utente.

Postcondizioni: L'utente seleziona un utente inesistente.

Scenario Principale:

- L'utente vuole selezionare un untente inesistente;
- Il sistema manda un messaggio di errore all'utente dicendo che ha selezionato un utente inesistente.

UC17: Mostra tutte gli inviti

Attori Principali: Utente autenticato.

Precondizioni: L'utente è entrato nell'applicazione web.

Descrizione: L'utente vuole vedere tutti gli inviti ricevuti.

Postcondizioni: L'utente visualizza tutti gli inviti ricevuti.

Scenario Principale:

- L'utente visualizza l'applicazione web;
- L'utente clicca su mostra inviti;
- L'utente visualizza gli inviti ricevuti.

Estensioni:

- Errore user non trovato (UC16)

6.5 Tracciamento dei requisiti

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[Importanza][Tipologia][Codice]

- **Importanza:** Ogni requisito può appartenere solo a una delle classi di importanza elencate di seguito:
  - **O** (Requisito Obbligatorio): requisito fondamentale per la corretta realizzazione del progetto;
  - **D** (Requisito Desiderabile): requisito non fondamentale al progetto ma il cui soddisfacimento comporterebbe una maggiore completezza del prodotto;
  - **F** (Requisito Facoltativo): requisito non richiesto per il corretto funzionamento del prodotto ma che se incluso arricchirebbe il progetto. Prima di soddisfare il requisito è necessaria un'analisi di tempi e costi per evitare ritardi nella consegna e/o costi superiori a quelli preventivati.
- **Tipologia:** Di seguito sono riportate le tipologie di requisito:
  - **V:** Identifica un requisito di vincolo;
  - **F:** Identifica un requisito funzionale;
  - **Q:** Identifica un requisito di qualità.
- **Codice:** Ogni requisito è formato da un codice numerico che lo identifica in modo univoco.

Tabella 6.1: Tabella del tracciamento dei requisiti funzionali

Requisito	**Descrizione**	**Stato**
ROF-1	L'utente non autenticato può registrarsi	Completato
ROF-1.1	L'utente non autenticato deve poter inserire il proprio nome	Completato
ROF-1.2	L'utente non autenticato deve poter inserire il proprio cognome	Completato
ROF-1.3	L'utente non autenticato deve poter inserire la propria email	Completato
ROF-1.4	L'utente non autenticato deve poter inserire il proprio username	Completato
ROF-1.5	L'utente non autenticato deve poter inserire la propria data di nascita	Completato
ROF-1.6	L'utente non autenticato deve poter inserire la propria password	Completato
ROF-2	L'utente non autenticato deve poter effettuare il login	Completato
ROF-2.1	L'utente non autenticato deve poter inserire il proprio username	Completato
ROF-2.2	L'utente non autenticato deve poter inserire la propria password	Completato
ROF-3	L'utente autenticato deve poter vedere il proprio profilo	Completato
ROF-4	L'utente autenticato deve poter modificare le informazioni sul proprio profilo	Completato
ROF-4.1	L'utente autenticato deve poter modificare la foto del proprio profilo	Completato
ROF-5	L'utente autenticato deve poter modificare la propria password	Completato
ROF-6	L'utente autenticato deve poter cancellare il proprio account	Completato
RDF-6.1	Come conseguenza della cancellazione di un account devono essere cancellati anche tutti i viaggi creati dallo user	Completato
RDF-6.2	Come conseguenza della cancellazione di un account devono essere cancellate anche tutte le richieste di partecipazione ad un viaggio inviate dallo user	Completato
RDF-6.3	Come conseguenza della cancellazione di un account devono essere cancellate anche tutti gli inviti di partecipazione ad un viaggio collegate allo user	Completato
RDF-6.4	Come conseguenza della cancellazione di un account devono essere cancellate anche tutte le recensioni effettuate dallo user	Completato
ROF-7	L'utente autenticato deve poter creare un viaggio	Completato
ROF-7.1	L'utente autenticato deve poter inserire il luogo di partenza del viaggio	Completato
ROF-7.2	L'utente autenticato deve poter inserire la destinazione del viaggio	Completato
ROF-7.3	L'utente autenticato deve poter inserire la descrizione del viaggio	Completato

ROF-7.4	L'utente autenticato deve poter inserire la data di partenza del viaggio	Completato
ROF-7.5	L'utente autenticato deve poter selezionare le varie attività presenti nel viaggio	Completato
ROF-8	L'utente non autenticato deve poter visualizzare la lista viaggi	Completato
ROF-9	L'utente non autenticato deve poter visualizzare i singoli viaggi	Completato
ROF-10	L'utente proprietario del viaggio deve poter modificare un viaggio	Completato
ROF-11	L'utente proprietario del viaggio deve poterlo cancellare	Completato
ROF-11.1	In seguito alla cancellazione di un viaggio devono essere cancellate tutte le partecipazioni al viaggio	Completato
ROF-11.2	In seguito alla cancellazione di un viaggio devono essere cancellati tutti gli inviti di partecipazione collegati al viaggio	Completato
ROF-11	In seguito alla cancellazione di un viaggio devono essere cancellati tutte le richieste di partecipazione collegate al viaggio	Completato
ROF-12	L'utente proprietario del viaggio deve poter invitare persone nel viaggio	Completato
ROF-13	L'utente proprietario del viaggio deve poter accettare chi chiede di partecipare al viaggio	Completato
ROF-14	L'utente autenticato deve poter richiedere di partecipare a un viaggio	Completato
ROF-15	L'utente autenticato deve poter vedere i partecipanti a un viaggio	Completato
ROF-16	L'utente autenticato deve poter vedere le informazioni di un singolo partecipante al viaggio	Completato
ROF-17	L'utente autenticato deve poter vedere gli inviti per partecipare ai viaggi ricevuti	Completato
ROF-18	L'utente autenticato deve poter rispondere agli inviti ricevuti	Completato
ROF-19	L'utente autenticato deve poter inviare una richiesta di partecipazione ad un viaggio	Completato
ROF-20	L'utente proprietario del viaggio deve poter rispondere alle richieste di partecipazione al viaggio	Completato
ROF-21	L'utente proprietario del viaggio deve poter rimuovere un partecipante dal viaggio	Completato

Tabella 6.3: Tabella del tracciamento dei requisiti qualitativi

Requisito	**Descrizione**	**Stato**
ROQ-1	Deve essere implementato un sistema di log	Completato
ROQ-2	La base di dati e le interazioni con essa devono essere gestite unicamente con le funzioni fornite da Spring Data JPA	Completato
ROQ-3	Devono essere effettuati test con Postman di tutti i metodi esposti	Completato
RDQ-3	Il codice deve essere formattato in modo chiaro e deve essere autoesplicativo seguendo il più possibile i principi del Clean-Code	Completato

Tabella 6.5: Tabella del tracciamento dei requisiti di vincolo

Requisito	**Descrizione**	**Stato**
ROV-1	Il database deve essere configurato per utilizzare PostgreSQL	Completato
ROV-2	I microservizi non devono accedere a tabelle del database che non gli appartengono	Completato

Capitolo 7

Progettazione e codifica

7.1 Tecnologie e strumenti

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

Java

Java è un linguaggio di programmazione ad alto livello orientato agli oggetti. Nel progetto è stato utilizzato anche Jakarta EE.

Spring Core

Spring Core si occupa della gestione di: Inversion of Control, Dependency Injection, transazioni e accesso ai dati

Spring Boot

Spring Boot introduce convenzioni predefinite e configurazioni automatiche volte a semplificare notevolmente la configurazione e lo sviluppo. Inolte Embedded Web Server è un modulo di Spring Boot che permette l'utilizzo di web server integrati e Spring Boot Strarter contiene pacchetti preconfigurati che semplificano l'iterazione con tecnologie specifice.

Spring Data Jpa

Spring Data Jpa è un modulo del progetto Spring Data e fornisce un livello di astrazione superiore per semplificare l'accesso ai dati attraverso la JPA.

Lombok

Lombok è una libreria Java che aiuta a snellire il codice, in particolare si occupa di scrivere i costrutti base di alcune classi in base alle istruzioni che gli vengono date tramite delle annotazioni.

Loh4j

Log4j è una libreria Java che si occupa di facilitare la gestione nei log all'interno dei progetti. Offre dei pmetodi per tenere traccia del flusso dei codice tramite i log e offre anche la possibilità di dividere i log in base alla loro importanza in modo da tenere separati ad esempio i log di errore dai log normali.

Http request

Le richieste HTTP sono messaggi utilizzati per la comunicazione tra un client e un server su una rete, come ad esempio l'Internet. Queste richieste consentono al client di richiedere risorse dal server o di inviare dati al server.

Hibernate

Hibernate è una piattaforma Java open source che attraverso il relativo framework facilita la gestione dei dati tra un'applicazione Java e un database relazionale. Hibernate è un'implementazione del Java Persistence API (JPA) e fornisce un modo per mappare gli oggetti Java alle tabelle del database.

PostgreSQL

PostgreSql è un open source relational database management system, permette di creare e gestire basi di dati.

Postman

Postman è un ambiente di sviluppo API che consente agli sviluppatori di salvare e organizzare richieste ai servizi e di analizzare le risposte, grazie a questo rende possibile fare test di integrazione in modo molto veloce.

7.2 Progettazione

7.2.1 Architettura del progetto

L'architettura del progetto è composta da tre parti principali:

- Front end: si occupa di chiedere e inviare i dati al Back end e di mostrarli all'utente tramite un'interfaccia grafica.
- Microservizi: sviluppati usando Spring, si occupano di gestire le richieste del Front end, il Front end e il Back end comunicano esclusivamente attraverso l'Application Program Interface.
- Database: configurato su PostgreSQL, contiene sia i dati degli utenti che dei viaggi.

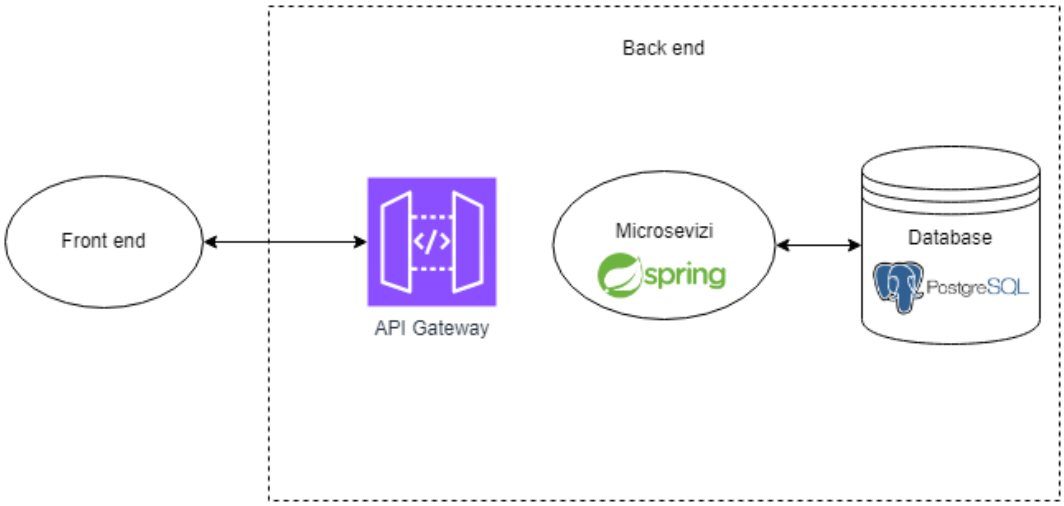


Figura 7.1: Architettura generale del progetto

7.2.2 Database

Il database è stato sviluppato tramite Spring Data JPA ed è stato configurato un database PostgreSQL. Si tratta di un database relazionale, è stato scelto un approccio con database relazionale in quanto le entità sono estremamente connesse tra loro e quindi un database non relazionale avrebbe portato una maggiore complessità di gestione. Allo schema mostrato nella figura 7.2 si possono vedere le varie entità del database con le chiavi primarie e le relazioni presenti tra le entità.

7.2.3 Microservizi

I microservizi sono stati sviluppati come API RESTful, consentendo una comunicazione flessibile. Inoltre, ogni servizio è accessibile solamente attraverso un Application Program Interface, il che crea un unico punto di accesso per semplificare l'iterazione con l'intera architettura. Si è scelto un approccio a microservizi con database unificato. Questo tipo di approccio deve essere accompagnato da una limitazione di accessi alle tabelle ad ogni servizio, permettendo ai servizi di accedere solo alle tabelle di loro competenza.

7.2.4 Servizio User

Il servizio User si occupa della gestione di: registrazione, login, mostra profilo utente, modifica utente, cancella utente. User copre quindi l'intero dominio riguardante gli user e le funzioni a essi collegati.

UserController: Si occupa di esporre gli API endpoint di login e registrazione.

UserDetailsController: Si occupa di esporre tutti gli API endpoint relativi al servizio User che vanno usati solo dopo essersi autenticati.

LoginService: Si occupa di restituire le informazioni di uno user e di verificare il login.

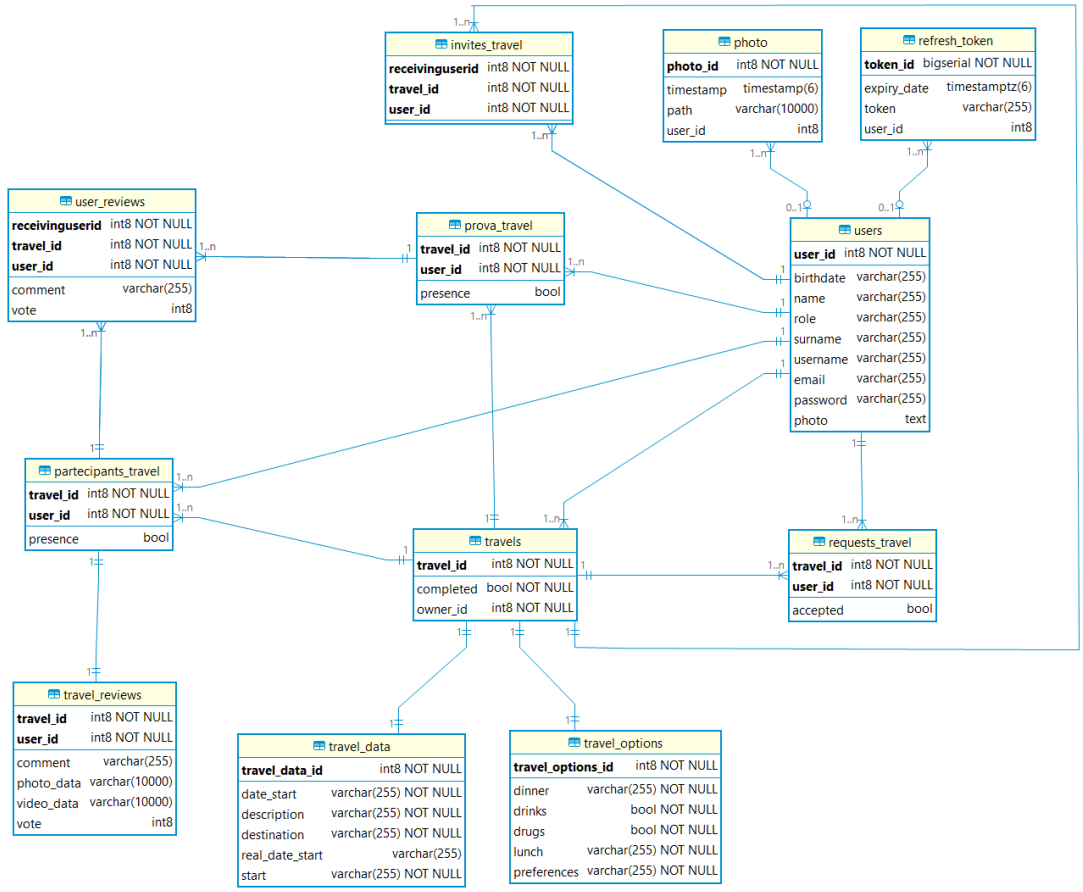


Figura 7.2: Diagramma Database

RegistrationService: Si occupa di effettuare la registrazione di un nuovo user.

UserManagementService: Si occupa di fornire tutti i metodi per la gestione di uno User (come modificare un campo).

DbUtils: Si occupa di trasformare i Dto in entità e viceversa.

ReviewClient: Si occupa di inviare le richieste necessarie al servizio Review.

TravelClient: Si occupa di inviare le richieste necessarie al servizio Client.

Dto: Tutte le classi dentro il package dto si occupano di fornire i dto per gli API endpoint esposti. I dto seguono la nomenclatura [nome del metodo]+Dto.

7.2.5 Model

Il Model contiene l'implementazione della JPA e funge quindi da base per il database. Il database è condiviso tra i servizi quindi è necessario che le classi del modello siano uguali in tutti i servizi altrimenti generano conflitti sul database.

DoubleId: Viene usata come id per tutte le classi a cui serve un id composto da userid e travelid.

Id: Classe astratta per il doppio id con userid e travelid.

TipleId: Viene usata come id da tutte le classi che hanno bisogno di un id composto da due userid e un travelid.

Invite: Rappresenta un invito a un viaggio che uno user riceve.

Partecipant: Rappresenta la partecipazione a un viaggio da parte di uno user.

Request: Rappresenta la richiesta di partecipazione ad un viaggio fatta da uno user.

Travel: Rappresenta la classe principale di viaggio.

TravelData: Rappresenta i dati di un viaggio come partenza, arrivo, ora di partenze ecc.

TravelOption: Rappresenta le opzioni di attivita che si possono svolgere durante il viaggio.

User: Rappresenta un utente iscritto all'applicazione e contiene le informazioni di uno user.

7.2.6 Servizio Travel

Il servizio Travel si occupa di gestire tutto quello che riguarda i viaggi, quindi: creazione di un viaggio, cancellazione, modifica, aggiunta partecipanti, invio e ricezione di inviti e richieste. Il servizio Travel copre tutto il dominio riguardante i viaggi inviti e richieste compresi.

TravelController: Si occupa di esporre gli API endpoint relativi alla creazione e gestione di un viaggio.

InviteController: Si occupa di esporre gli API endpoint relativi all'invio e risposta di inviti e richieste di partecipazione ai viaggi.

InviteService: Si occupa dell'invio e e di gestire la risposta agli inviti.

RequestService: Si occupa dell'invio e di gestire la risposta delle richieste.

TravelService: Si occupa della creazione ecancellazione dei viaggi, inoltre si occupa anche di ritornare i viaggi cercati.

TravelManagementService: Si occupa della gestione dei viaggi quindi modifiche e farli partire.

PassengerManagementService: Si occupa della gestione dei passeggeri nei viaggi aggiungendo e togliendo passeggeri.

UserService: Si occupa di recuperare le informazioni degli user invocando la chiamata a UserClient per chiedere le informazioni al servizio user.

UserClient: Si occupa di inviare le richieste necessarie al servizio user e di restituire le informazioni sullo user richiesto a chi lo ha invocato.

ReviewClient: Si occupa di inviare le richieste necessarie al servizio review.

DbUtils: Si occupa di trasformare i Dto in entità e viceversa.

Dto: Dentro il package Dto sono presenti delle classi chiamate [nome del metodo]+Dto le quali servono agli API endpoint per ricevere e inviare informazioni.

7.3 Design Pattern utilizzati

I design pattern sono soluzioni progettuali generali e ricorrenti per problemi che si verificano molto spesso nella progettazione del software. I design pattern approcciano il problema e lo risolvono in modo strutturato e offrono una soluzione che può essere utilizzata più volte.

Dto: Serve a trasferire dati da un sistema ad un altro. Un DTO è un oggetto con una strutture semplice e senza logica dentro.

Inversion of Control: L'Inversion of Control (IoC) è un principio di progettazione del software che ribalta il controllo della gestione del flusso dell'applicazione rispetto alle tecniche classiche della programmazione. Laa gestione del flusso dell'applicazione viene spostata dall'applicazione a un framework o un contenitore IoC..

Dependency Injection: Tutti gli oggetti sono creati in modo da applicare una dependency injection tramite costruttore così da lasciare a Spring il compito di istanziare gli oggetti quando serve.

Repository Pattern: Il Repository Pattern è un design pattern che fornisce un'astrazione per l'accesso ai dati in un'applicazione. L'obbiettivo del pattern è quello di separare la logica di accesso ai dati dalla parte di business logic..

7.4 Codifica

La codifica è stata divisa in due parti una relativa al servizio User e una relativa al servizio Travel. Si è quindi proceduto prima allo sviluppo del servizio user.
Durante la fase di verifica si è proceduto tenendo ogni servizio su una repository GitHub separata in modo da poter proseguire con uni sviluppo in parallelo di più parti del progetto.

7.4.1 Servizio User

Il servizio user espone diversi metodi relativi alla gestione e creazione degli user.

login: Serve per effettuare il login di uno user già registrato.

signUp: Serve per effettuare la registrazione di un nuovo user.

modifyEmail: Serve per modificare l'email di uno user.

modifyUser: Serve a modificare tutti i campi user insieme.

modifyPassword: Serve a modificare la password di uno user.

modifyUsername: Serve a modificare uno username di uno user.

modifyPhoto: Serve a modificare la foto di uno user.

takeUserById: Serve a mostrare i dati di uno user dato il suo id.

takeUserByUsername: Serve a mostrare i dati di uno user dato il suo username.

deleteAccount: Serve a cancellare uno user.

7.4.2 Servizio Travel

Il servizio travel esponde diversi metodi relativi alla creazione e gestione dei viaggi.

addTravel: Serve a creare un nuovo viaggio.

travelList: Serve a mostrare la lista dei viaggi.

showTravel: Serve a mostrare il singolo viaggio.

showTravelPartecipant: Serve a mostrare i partecipanti ad un viaggio.

modifyTravelOption: Serve a modificare le opzioni di un viaggio.

modifyDescription: Serve a modificare la descrizione di un viaggio.

startTravel: Serve a far partire un viaggio.

addPassenger: Serve ad aggiungere un passeggero al viaggio.

removePassenger: Serve a rimuovere un passeggero dal viaggio.

deleteTravel: Serve a cancellare un viaggio.

deleteTravelFromOwnerId: Serve a cancellare tutti i viaggi creati da uno user.

sendTravelInvite: Serve a inviare un invito per un viaggio.

sendTravelPartecipationRequest: Serve a inviare una richiesta per un viaggio.

responseInvite: Serve a rispondere ad un invito.

responsePartecipationRequestPartecipation: Serve a rispondere a una richiesta.

showAllInvite: Serve a mostrare tutti gli inviti ricevuti da uno user.

showAllRequest: Serve a mostrare tutte le richieste ricevute da un viaggio.

7.5 Problematiche riscontrate

Durante lo sviluppo della funzione di cancellazione dell'account da parte di un utente si è riscontrato un problema in quanto la tabella utente risultava avere diverse dipendenze entranti le quali ne bloccavano la cancellazione. Per risolvere questo problema si è pensato a due possibili soluzioni: cancellare tutte le tabelle generate dall'utente e mettere a nullo tutti i riferimenti ad esso da tabelle non create dall'utente stesso oppure creare un utente speciale (cancellato) con il quale rimpiazzare tutti i riferimenti al vecchio utente. In seguito ad un confronto con i colleghi si è deciso di seguire una politica di cancellazione totale dell'utente quindi la prima opzione.
Nello sviluppo della funzionalità del profilo utente per il servizio user, abbiamo dovuto affrontare sfide legate alla gestione delle immagini. Un mio collega ha già riscontrato questo problema durante lo sviluppo di un servizio di review. Inizialmente pensavo di elaborare l'immagine utilizzando un tipo di dati specifico o semplicemente di archiviare il percorso del file in un database. A seguito di aulcure ricerche è stata individuata come soluzione la codifica delle immagini in formato Base64. Questo approccio consente di trattare le immagini come stringhe e di memorizzarle direttamente nel database. La codifica Base64 delle immagini si è rivelata una scelta vantaggiosa poiché semplifica la gestione e l'integrazione dei dati e riduce la complessità associata all'archiviazione dei file binari tradizionali.

Capitolo 8

Conclusioni

8.1 Consuntivo finale

L0oggetto di questo tirocinio è stato lo sviluppo di parte del back end di una web app. Per realizzare questo obiettivo, si è iniziato con un'analisi del problema e uno studio individuale delle tecnologie, per poi proseguire con lo sviluppo finale del progetto. Il progetto è stato realizzato seguendo una architettura a microservizi sviluppata con Java Spring.
Il caso d'uso individuato è stata la crazione di due servizi per la gestione della parte dei viaggi e la gestione degli utenti.

8.2 Raggiungimento degli obiettivi

In merito al raggiungimento degli obiettivi prefissati per il tirocinio si, si può confermare la soddisfazione di tutti gli obiettivi. Complentando sia gli obiettivi obbligatori sia quelli desiderabili e facoltativi, l'unica cosa mancante sarebbe stata la dockerizzazione dei servizi tuttavia questa attività non rientrava tra gli obbiettivi e se ne è discusso con l'azienda solo alla fine del periodo di tirocinio.

8.3 Conoscenze acquisite

Durante il periodo di tirocinio ho approfondito la conoscenza del linguaggio di programmazione Java, con un focus particolare sulla piattaforma Jakarta EE. In questo contesto ho acquisito una conoscenza approfondita delle tecnologie e delle pratiche utilizzate per sviluppare applicazioni aziendali basate su Java. Inoltre, mi sono dedicato ad uno studio approfondito del framework Spring, concentrandomi principalmente sulla gestione dell'inserimento delle dipendenze e sull'utilizzo dei moduli Spring Boot e Spring Data. Dopo questa esperienza ho una comprensione più approfondita di come Spring semplifica lo sviluppo di applicazioni Java promuovendo la progettazione modulare e facilitando l'integrazione di vari componenti.
Nel periodi di tirocinio, ho dedicato tempo ad approfondire e ad analizzare in dettaglio l'architettura dei microservizi. Attraverso questa ricerca, siamo stati in grado di comprendere in che modo le architetture basate su microservizi differiscono dagli approcci monolitici. In particolare, ci siamo concentrati sul processo di passaggio da un'archi-

tettura monolitica a un'architettura basata su microservizi ed abbiamo esplorato le sfide, le best practice e gli strumenti associati a questo processo di migrazione. Questa esperienza mi ha fornito una solida base di conoscenze pratiche e teoriche nel campo dello sviluppo software, con un focus particolare sulla progettazione e implementazione di soluzioni basate su architetture Java, Spring e microservizi.

8.4 Valutazione personale

L'attività di stage si è rivelata molto utile per la crescita personale e professionale, permettendo di acquisire nuove conoscenze e competenze in ambiti di sviluppo e integrazione di microservizi, che sono stati approfonditi nel modo mirato nel progetto realizzato. Il progetto svolto ha rappresentato per me l'opportunità di mettere in pratica molto di quello studiato negli ulti anni, Inoltre l'aver svolto il progetto dentro una realtà lavorativa ha dato modo di conoscere nuove persone e vedere com'è l'ambiente di lavoro dell'ambito che sto studiando.

Acronimi e abbreviazioni

API Application Program Interface. 1, 8, 30, 31, 40

UML Unified Modeling Language. 10, 41

Glossario

API In informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 39

API endpoint In informatica il termine *API endpoint* denota un metodo che l'*API* espone e dal quale riceve le richieste e invia le risposte. Ogni *API endpoint* è caratterizzato da un URL che lo distingue.. 3, 31–34, 40

API Gateway Un *API gateway* è uno strumento di gestione delle API che si situa tra un client e una raccolta di servizi back end. Un *API gateway* si comporta come un proxy inverso per accettare tutte le chiamate API, aggregare i vari servizi richiesti per gestirle e restituire i risultati appropriati.. 30, 31, 40

Back end In informatica il termine *back end* denota la parte non visibile all'utente di un programma e che permette l'effettivo funzionamento delle interazioni con il front end.. ii, 3, 30, 40

framework In informatica con il termine *framework* si indica un'architettura logica di supporto su cui un software può essere progettato e realizzato, facilitandone lo sviluppo da parte del programmatore. Questo permette di avere un'astrazione del codice, permettendo di concentrarsi sulla logica dell'applicazione, senza dover gestire le parti più complesse. ii, 5, 6, 30, 34, 37, 40

Front end In informatica il termine *front end* denota la parte visibile all'utente di un programma e con cui egli può interagire, tipicamente un'interfaccia utente. ii, 3, 30, 40

open source In informatica con il termine *open source* si indica un software di cui viene reso pubblico il codice sorgente, permettendo a chiunque di poterlo utilizzare, studiare, modificare e distribuire liberamente. 6, 30, 40

repository In informatica con il termine *repository* si indica un ambiente di un sistema informativo, in cui vengono gestiti i metadati, attraverso tabelle relazionali, e i file, attraverso un file system gerarchico. 6, 34, 40

restful Una API RESTful è un'interfaccia di programmazione delle applicazioni (API o API web) conforme ai vincoli dello stile architetturale REST, che consente l'interazione con servizi web RESTful. Il termine REST è l'acronimo di REpresentational State Transfer.. 31, 41

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 39

Bibliografia

Riferimenti bibliografici

Robert C. Martin, Dean Wampler. *Clean Code: A Handbook of Agile Software Craftsmanship, First Editon*. Prentice Hall, 2008.

Siti web consultati

Documentazione Spring. URL: https://docs.spring.io/spring-framework/reference/index.html.

Manifesto Agile. URL: https://agilemanifesto.org/iso/it/manifesto.html (cit. a p. 1).

microservice-material. URL: https://microservices.io/.