



**UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO**

---

DIPARTIMENTO DI INGEGNERIA GESTIONALE, DELL'INFORMAZIONE  
E DELLA PRODUZIONE

Corso di Laurea Magistrale in Ingegneria Informatica

# Mountain App

Progetto del corso di Progettazione, Algoritmi e Computabilità

Prof.ssa  
**Patrizia Scandurra**

Candidati:  
**Federico Imberti**      **Cristian Tironi**  
Matricola: 1066358      Matricola: 1064464

**Stefano Gregis**      **Sabrin Maatoug**  
Matricola: 1073318      Matricola: 1065576



# Indice

<b>1 Iterazione 0</b>	<b>9</b>
1.1 Introduzione . . . . .	9
1.2 Requisiti e casi d'uso . . . . .	11
1.2.1 Use cases . . . . .	11
1.2.2 Priorità dei casi d'uso . . . . .	13
1.2.3 Use case diagram . . . . .	15
1.3 Architettura . . . . .	16
1.4 Toolchain . . . . .	18
<b>2 Iterazione 1</b>	<b>21</b>
2.1 Introduzione . . . . .	22
2.2 Component Diagram . . . . .	23
2.3 Data Class Diagram . . . . .	24
2.4 Interface Diagram . . . . .	25
2.5 Mockup dell'applicazione . . . . .	26
2.5.1 Homepage e Dettagli di un Evento . . . . .	27
2.5.2 Login e registrazione dell'utente . . . . .	28
2.5.3 Ricerca escursione e Filtraggio . . . . .	29
2.5.4 Visualizzazione del profilo utente . . . . .	30
<b>3 Iterazione 2</b>	<b>31</b>
3.1 Introduzione . . . . .	31
3.2 Component Diagram . . . . .	32
3.3 DataClass Diagram . . . . .	33

3.4	Interface Diagram . . . . .	34
3.5	Use Case . . . . .	35
3.5.1	UC-1 Login . . . . .	36
3.5.2	UC-2 Signup . . . . .	38
3.5.3	UC-3 Logout . . . . .	40
3.5.4	UC-7 Ricerca Evento . . . . .	41
3.5.5	UC-8 Visualizzazione risultati della ricerca . . . . .	42
3.5.6	UC-9 Visualizzazione di uno specifico evento . . . . .	43
3.5.7	UC-10 Prenotazione a un nuovo evento . . . . .	44
3.5.8	UC-11 Cancellazione da un evento . . . . .	45
3.5.9	UC-15 Algoritmo per selezionare i partecipanti . . . . .	46
3.5.10	UC-17 Eliminazione di un evento . . . . .	50
3.5.11	UC-18 Creazione di un evento . . . . .	51
3.6	Documentazione API . . . . .	52
3.7	Testing . . . . .	55
3.7.1	Analisi statica . . . . .	55
3.7.2	Analisi dinamica . . . . .	59
3.7.3	Unit test . . . . .	61
<b>4</b>	<b>Iterazione 3</b>	<b>63</b>
4.1	Introduzione . . . . .	63
4.2	Continuous Integration e Continuous Deployment (CI/CD) . . . . .	64
4.2.1	Continuous Integration (CI) . . . . .	64
4.2.2	Continuous Deployment (CD) . . . . .	66
4.3	API Gateway . . . . .	67
4.3.1	Creazione dell'API Gateway . . . . .	67
4.3.2	Testing del gateway . . . . .	68
4.4	DataClass Diagram . . . . .	69
4.5	Component Diagram . . . . .	70
4.6	Documentazione API . . . . .	71
4.7	Diagramma delle interfacce . . . . .	73

<b>INDICE</b>	<b>5</b>
4.8    Implementazione Algoritmo . . . . .	74
4.8.1    Sequence diagram invocazione algoritmo . . . . .	75
4.9    Casi d'uso . . . . .	76
4.9.1    UC-4 Visualizza iscrizioni . . . . .	77
4.9.2    UC-5 Visualizzazione Meteo . . . . .	78
4.9.3    UC-6 Visualizza eventi consigliati . . . . .	79
4.9.4    UC-13 Visualizza il profilo di un utente . . . . .	80
4.9.5    UC-14 visualizza il profilo di un organizzatore . . . . .	81
4.9.6    UC-6 Visualizza dettagli evento . . . . .	82
4.10   Testing . . . . .	83
4.10.1   Analisi statica . . . . .	83
4.10.2   Analisi dinamica . . . . .	87
4.10.3   Unit test . . . . .	89
4.11   Generazione dei mockup . . . . .	91
4.11.1   Generazione delle escursioni con ChatGPT . . . . .	91
4.11.2   Generazione dei profili utente con ChatGPT . . . . .	91
4.11.3   Generazione delle immagini delle escursioni con Crayon V3 . .	92
<b>5 Iterazione 4</b>	<b>99</b>
5.1    Introduzione . . . . .	99
5.2    Casi d'uso . . . . .	100
5.2.1    UC-4 Visualizzazione delle iscrizioni . . . . .	101
5.2.2    UC-23 Visualizzare profilo dell'organizzatore di un evento . .	102
5.3    Docker-compose, Continuous Integration e Continuous Deployment .	103
5.3.1    Immagini Docker e struttura finale del docker-compose . . .	103
5.3.2    Continuous Integration (CI) . . . . .	104
5.3.3    Continuous Deployment (CD) . . . . .	105
5.4    Guida per l'installazione . . . . .	106
5.4.1    Esecuzione locale . . . . .	106
5.4.2    Esecuzione remota . . . . .	107
5.5    Manuale utente . . . . .	108

5.6 Sviluppi Futuri . . . . .	111
5.6.1 Status dei casi d'uso . . . . .	111
5.6.2 Punti da trattare . . . . .	112

# Elenco delle figure

1.1	Use cases . . . . .	15
1.2	Deployment Diagram in versione free style . . . . .	17
2.1	Component Diagram. . . . .	23
2.2	Data Class Diagram. . . . .	24
2.3	Interface Diagram. . . . .	25
3.1	Component Diagram. . . . .	32
3.2	Class Diagram. . . . .	33
3.3	Interface Diagram. . . . .	34
3.4	FlowChart . . . . .	48
3.5	Aggiunta di un nuovo evento . . . . .	53
3.6	Inserimento nuova prenotazione . . . . .	53
3.7	Prenotazione dato uno specifico profilo . . . . .	53
3.8	Creazione nuovo profilo . . . . .	54
3.9	Test collection . . . . .	60
3.10	findById() . . . . .	62
4.1	Continuous Integration della branch main . . . . .	64
4.2	Continuous Deployment di un immagine . . . . .	66
4.3	Risultati testing gateway . . . . .	68
4.4	Class Diagram. . . . .	69
4.5	Component Diagram. . . . .	70
4.6	Profile level . . . . .	71
4.7	Ottenere tutte le prenotazioni confermate di un utente . . . . .	71

4.8 Profilo utente . . . . .	71
4.9 Selezione partecipanti . . . . .	72
4.10 Interface Diagram . . . . .	73
4.11 Sequence diagram invocazione algoritmo . . . . .	75
4.12 Test collection . . . . .	88
4.13 findById() . . . . .	90
4.14 Primo prompt per la generazione di escursioni . . . . .	93
4.15 Risposta parziale al primo prompt . . . . .	94
4.16 Secondo prompt e risposta . . . . .	95
4.17 Primo prompt per la generazione di utenti . . . . .	96
4.18 Risposta parziale al primo prompt . . . . .	97
4.19 Prompt per la generazione di immagini . . . . .	98
5.1 Login screen . . . . .	109
5.2 Signup screen . . . . .	109
5.3 Profilo utente . . . . .	109
5.4 Iscrizioni . . . . .	109
5.5 Homescreen . . . . .	110
5.6 New evento . . . . .	110
5.7 Suggerimenti . . . . .	110
5.8 Cerca . . . . .	110

# Capitolo 1

## Iterazione 0

### 1.1 Introduzione

Ad oggi la maggior parte delle persone che si organizzano per andare insieme in montagna fanno ciò attraverso passaparola, gruppi Telegram o gruppi Whatsapp. Manca quindi un modo per aggregare queste persone e permettergli di comunicare in maniera agevole e veloce per organizzare le escursioni che vogliono fare. L'obiettivo è quello di creare una app che permetta di fare proprio questo, mettendo a disposizione degli utenti un ambiente dove poter pubblicare escursioni che intendono compiere e permettendo ad altri utenti di iscriversi per prendervi parte. Gli attori coinvolti saranno essenzialmente di 2 tipi: Organizzatori: possono creare degli eventi e pubblicarli sulla piattaforma; Partecipanti: possono unicamente iscriversi agli eventi pubblicati dagli organizzatori e non possono crearne di loro. Sia organizzatori che partecipanti possono iscriversi a degli eventi pubblicati sulla piattaforma. Per tutti gli attori andranno salvate le informazioni anagrafiche di base (eg. nome e cognome) e un'immagine profilo. Sarà poi necessario poter distinguere a quale categoria di attore appartiene un dato utente, in modo da poter personalizzare l'esperienza sul client fornendo agli organizzatori i comandi per creare un evento. Le escursioni conterranno svariate informazioni sull'evento come l'ora e il giorno, oltre alla posizione. Dovranno inoltre contenere una lista di utenti iscritti e una seconda di organizzatori, dato che un evento può avere più organizzatori. Sarà inoltre necessario specificare un'ora e un luogo di ritrovo, la difficoltà del percorso e l'attrezzatura necessaria (per

esempio se sono presenti pareti da scalare) o consigliata (per esempio una quantità minima di acqua). Lato client è inoltre richiesto che ogni evento mostri le condizioni meteorologiche attese per il giorno dell'evento, tale informazione verrà recuperata tramite le API di Visual Crossing. La piattaforma è creata per rasentare un social media, dove gli organizzatori hanno dei loro profili che possono essere seguiti dagli utenti. Inoltre, al termine di ogni escursione, viene data all'utente la possibilità di esprimere un giudizio da 1 a 5 stelle all'escursione. La media di questi punteggi determinerà un giudizio in stelle assegnato all'organizzatore. Ogni attore disporrà inoltre di una lista di escursioni a cui ha preso parte, le informazioni contenute in questa lista potranno essere usate per consigliargli altre escursioni in base a criteri come il giorno, la difficoltà e il dislivello. Ogni utente dovrà registrarsi alla piattaforma tramite il client e dovrà fare l'accesso prima di poter visualizzare gli eventi. Nel momento dell'iscrizione verrà fornito un campo per permettere agli organizzatori di inserire una speciale chiave fornita dall'organizzazione responsabile della piattaforma. Questa gli permetterà di iscriversi come organizzatori e non come semplici partecipanti. Prevediamo inoltre un limite massimo di persone per ogni escursione, che verrà impostato dall'organizzatore. Nel momento in cui il numero di iscrizioni supera il numero massimo di persone consentite un algoritmo automatico scarta coloro che ritiene non essere all'altezza del percorso, basandosi sulle loro precedenti esperienze. Un esempio potrebbe essere un utente che non ha mai avuto esperienze su un percorso ad alta difficoltà.

## 1.2 Requisiti e casi d'uso

### 1.2.1 Use cases

Di seguito sono riportati i requisiti del sistema che si vuole implementare. Per ciascun caso si indica il codice del caso d'uso, il titolo e la user story al fine di facilitare la comprensione.

- **UC1 - Login.** Come utente voglio accedere al mio account;
- **UC2 - Signup.** Come utente voglio potermi registrare al sistema;
- **UC3 - Logout.** Come utente voglio potermi disconnettere dal sistema;
- **UC4 - Visualizzazione iscrizioni.** Come utente voglio visualizzare gli eventi ai quali sono iscritto;
- **UC5 - Visualizzazione meteo.** Come utente voglio visualizzare il meteo entro 5 giorni dall'evento;
- **UC6 - Visualizzazione consigliati.** Come utente voglio poter visualizzare gli eventi consigliati;
- **UC7 - Ricerca eventi.** Come utente voglio poter cercare dei nuovi eventi, scegliendo degli opportuni parametri di ricerca;
- **UC8 - Visualizzazione risultati ricerca.** - Come utente, voglio poter visualizzare tutti i risultati della ricerca;
- **UC9 - Visualizzazione specifico risultato.** - Come utente, voglio poter consultare il singolo risultato della ricerca eseguita, visualizzando nei dettagli informazioni quali la località, la data e l'organizzatore;
- **UC10 - Iscrizione ad un nuovo evento.** Come utente, voglio potermi iscrivere ad uno degli eventi disponibili;
- **UC11 - Cancellazione iscrizione.** Come utente, voglio poter cancellare la mia iscrizione ad un evento;

- **UC12 - Visualizzazione partecipanti.** Come utente, voglio poter visualizzare gli utenti che parteciperanno ad un evento;
- **UC13 - Visualizzazione profilo utente.** Come utente, voglio poter visualizzare sul mio profilo le esperienze a cui ho preso parte e quelle a cui sono iscritto;
- **UC14 - Visualizzazione profilo organizzatore.** Come utente organizzatore, voglio poter visualizzare sul mio profilo gli eventi passati da me organizzati e quelli futuri;
- **UC15 - Algoritmo iscrizione.** Come utente, voglio che solo i profili più adeguati possano partecipare ad un evento, e che solamente gli N profili più adeguati vengano accettati;
- **UC16 - Dettagli evento.** Come utente, voglio poter visualizzare tutte le informazioni relative all'evento per poter valutare se iscrivermi o meno;
- **UC17 - Eliminazione evento.** Come organizzatore voglio poter cancellare un evento che ho creato;
- **UC18 - Creazione evento.** Come organizzatore voglio poter creare un evento;
- **UC19 - Visualizzare profilo partecipanti.** Come utente voglio poter visualizzare il profilo di coloro che sono iscritti a un evento;
- **UC20 - Cambiare la foto del profilo.** Come utente voglio poter cambiare la mia foto di profilo;
- **UC21 - Cambiare la foto di copertina.** Come utente voglio poter cambiare la foto di copertina del mio profilo;
- **UC22 - Cambiare la foto di un evento.** Come organizzatore voglio poter cambiare la foto che descrive un mio evento;
- **UC23 - Visualizzare profilo dell'organizzatore di un evento.** Come utente voglio poter visualizzare il profilo di chi organizza un evento.

### 1.2.2 Priorità dei casi d'uso

I casi d'uso possono essere ripartiti all'interno di tre code, a seconda della loro priorità nel processo di sviluppo.

- Coda ad alta priorità: contiene i requisiti ritenuti fondamentali per il funzionamento dell'applicativo, ossia la creazione dei profili utente, la creazione di nuovi eventi ed il sistema d'iscrizione agli eventi stessi. Saranno i primi use case implementati sia lato client che server;
- Coda a media priorità: coda contenente funzionalità di supporto, principalmente legate alla visualizzazione d'informazioni ulteriori riguardo agli utenti e/o agli eventi;
- Coda a bassa priorità: coda nella quale vengono indicate le funzionalità meno rilevanti, non è prevista la loro implementazione tuttavia potremmo decidere di implementarli in base a come procederà il progetto.

In particolare, nelle tabelle di seguito [1.1,1.2,1.3] è possibile osservare come i casi d'uso precedentemente individuati possano essere suddivisi.

NUMERO	TITOLO
UC1	Login
UC2	Signup
UC3	Logout
UC7	Ricerca eventi
UC8	Visualizzazione risultati ricerca
UC9	Visualizzazione specifico risultato
UC10	Iscrizione ad un nuovo evento
UC11	Cancellazione iscrizione
UC15	Algoritmo iscrizione
UC17	Eliminazione evento
UC18	Creazione evento

Tabella 1.1: Coda ad alta priorità

NUMERO	TITOLO
UC5	Visualizzazione meteo
UC6	Visualizzazione consigliati
UC13	Visualizzazione profilo utente
UC14	Visualizzazione profilo organizzatore
UC16	Dettagli evento

Tabella 1.2: Coda a media priorità

NUMERO	TITOLO
UC4	Visualizzazione iscrizioni
UC12	Visualizzazione partecipanti
UC19	Visualizzare profilo partecipanti
UC20	Cambiare la foto del profilo
UC21	Cambiare la foto di copertina
UC22	Cambiare la foto di un evento
UC23	Visualizzare profilo dell'organizzatore di un evento

Tabella 1.3: Coda a bassa priorità

### 1.2.3 Use case diagram

I requisiti precedentemente elencati possono essere visualizzati graficamente tramite l'use case diagram riportato in figura.

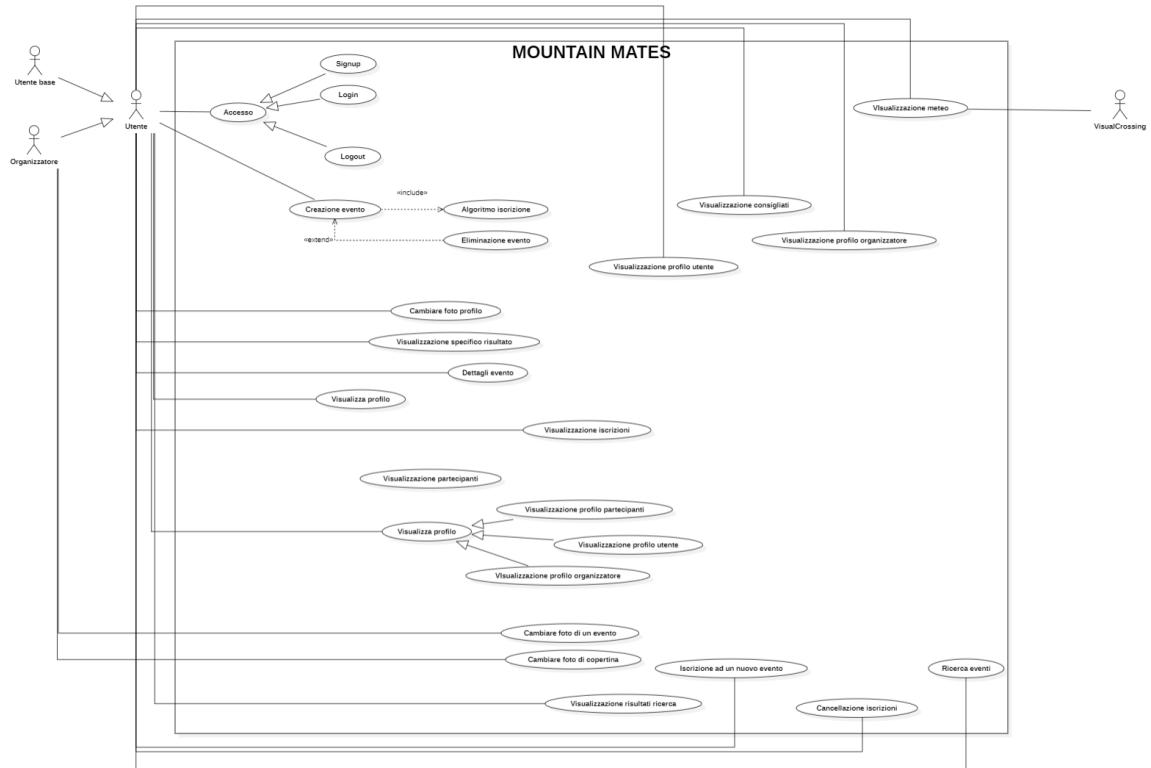


Figura 1.1: Use cases

È stata introdotta una generalizzazione sugli utenti per distinguere le funzionalità accessibili da un utente base rispetto a quelle di un organizzatore. L'entità Visual Crossing è coinvolta nella visualizzazione del meteo sul luogo dell'evento.

Inoltre, l'iscrizione ad un evento richiede l'esecuzione del caso d'uso algoritmo d'iscrizione in quanto, come detto precedentemente, una volta raggiunto un massimo di iscritti solamente i primi N più qualificati saranno accettati all'evento.

## 1.3 Architettura

L'architettura del sistema è stata definita attraverso il deployment diagram mostrato in figura 1.2, espresso in notazione free style. Anzitutto si identifica un pattern architettonico multi-layer, dove:

1. Il Presentation Layer è costituito dalle applicazioni lato client che vengono utilizzate dagli utenti e hanno il solo compito di recuperare le informazioni dal sistema e visualizzarle, oltre che permettere all'utente d'interagire con il servizio;
2. L'Application Layer gestisce la logica di business. È costituito da tre macro-sistemi specializzati:
  - **Gestore profilo**, responsabile di tutto ciò che riguarda i profili degli utenti, come gestire la registrazione, il login e soddisfare le query inerenti all'utente. Questo gestore si connette ai servizi di Firebase per gestire l'autenticazione in fase di login;
  - **Gestore escursione**, responsabile di gestire le richieste inerenti alle escursioni, come ottenere una lista o i dettagli di uno specifico evento. Per fornire informazioni meteorologiche riguardo agli eventi, il gestore si connette ai servizi di Visual Crossing tramite API call. La risposta viene elaborata in loco e restituita al client in un formato più breve da definirsi in futuro;
  - **Gestore prenotazioni**, responsabile di gestire le richieste riguardanti le prenotazioni che gli utente possono far per ogni evento.
3. Il Data Layer si occupa della persistenza dei dati ed è costituito da un database relazionale dove sono conservate le informazioni di ogni evento e utente registrato alla piattaforma.

Inoltre identifichiamo pattern client-server dove il client è costituito dalla mobile app e si interfaccia con il server attraverso l'API gateway.

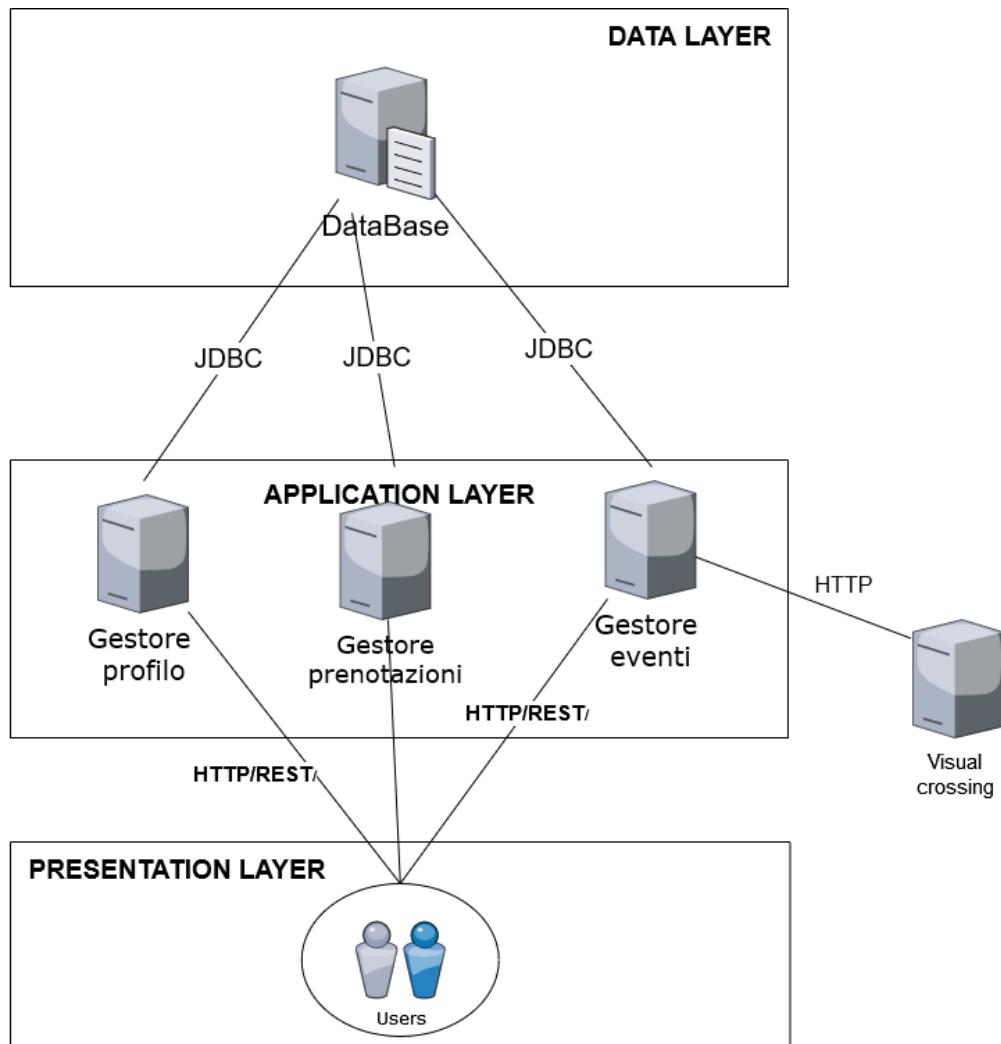


Figura 1.2: Deployment Diagram in versione free style

## 1.4 Toolchain

Per la realizzazione della piattaforma verranno utilizzati i seguenti strumenti:

- **Modellazione:**

- Diagramma dei casi d'uso, deployment diagram, component diagram, class diagram, diagrammi di flusso, diagramma entità-relazione: Draw.io e StarUML

- **Implementazione Applicazione Client:**

- Linguaggio di programmazione: Dart;
- IDE: Visual Studio Code;
- Interfaccia grafica: Flutter;
- Software per mockup: Figma;

- **Implementazione Web Server:**

- Linguaggio di programmazione: Java;
- IDE: Eclipse;
- Framework: Spring;
- Analisi statica: STAN4J;
- Analisi dinamica: JUnit;
- Deployment: Digital Ocean.

- **Implementazione Database:**

- Tipologia: Relazionale;
- Database: PosgreSQL;
- Provider: Digital Ocean.

- **Documentazione, versioning e organizzazione del team:**

- Documentazione: LaTeX;

- Versioning: GitHub;
- Git client: Github Desktop;
- Organizzazione del Team: Microsoft Teams e Trello.



# Capitolo 2

## Iterazione 1

## 2.1 Introduzione

Nell'iterazione 1 è stata perfezionata la specifica dei componenti progettati durante l'iterazione 0 in modo da definire meglio la struttura della piattaforma. Abbiamo creato lo scheletro del webserver, scritto in Spring, e lo scheletro dell'applicazione client in Flutter. È stata Inoltre definita una serie di mockup per modellare la parte grafica dell'app client.

## 2.2 Component Diagram

Il component diagram mostra come la componente application sia interfacciata tramite HTTP/REST con il client. È inoltre responsabile della comunicazione con il database e con i servizi esterni impiegati.

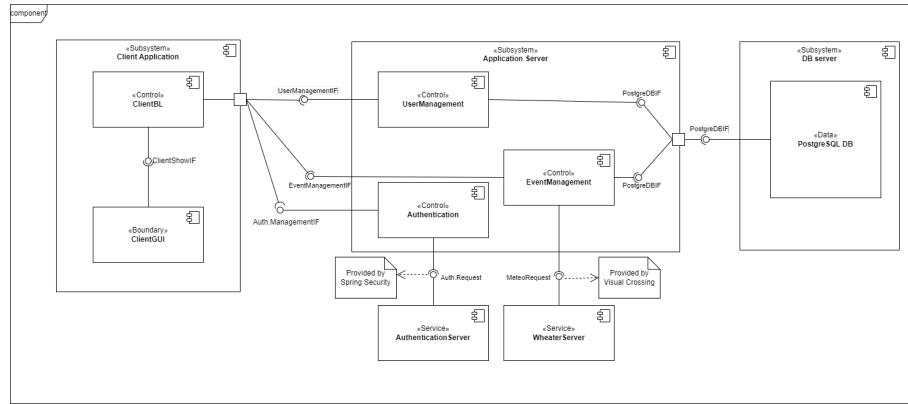


Figura 2.1: Component Diagram.

## 2.3 Data Class Diagram

Il diagramma delle classi in figura rappresenta le principali entità ed i relativi attributi.

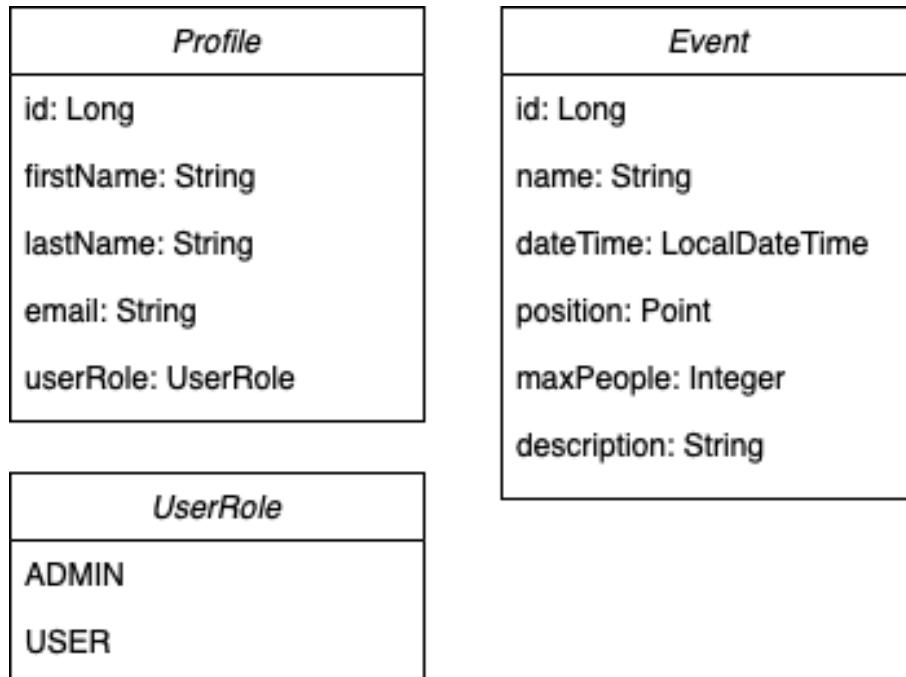


Figura 2.2: Data Class Diagram.

## 2.4 Interface Diagram

Le interfacce che verranno implementate nelle successive iterazioni sono rappresentate in figura 2.3

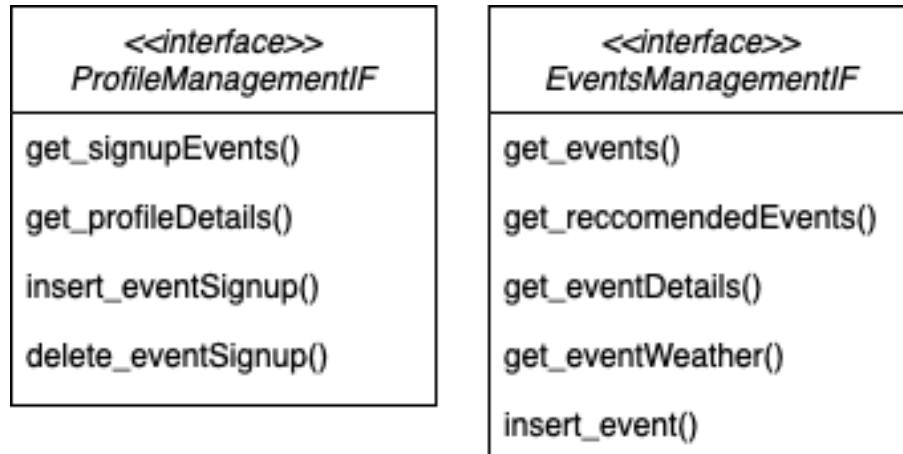
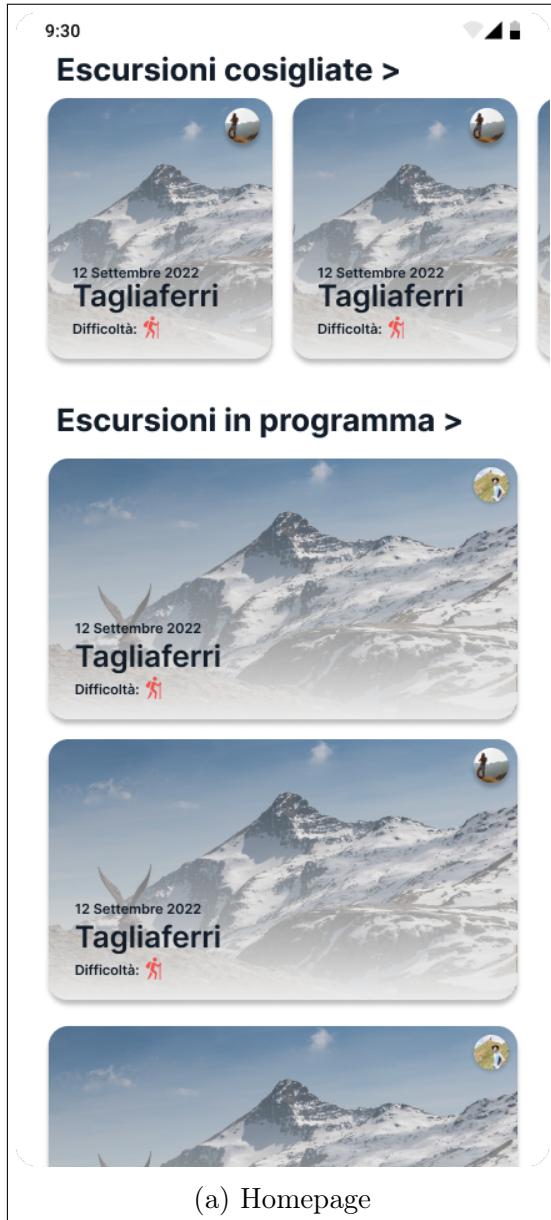


Figura 2.3: Interface Diagram.

## 2.5 Mockup dell'applicazione

Prima di realizzare l'app che permette all'utente di interagire con il servizio abbiamo pensato di creare dei mockup del design. In questo modo la scrittura dell'applicazione può proseguire basandosi su un design concordato con tutto il team e l'eventuale committente. Mettere a disposizione dei mockup e usarli per accordarsi sul design permette una progettazione più veloce e consistente rispetto alle modifiche fatte più avanti nel tempo direttamente sul prodotto software. Durante la realizzazione dell'app gli sviluppatori seguiranno al meglio i mockup, tuttavia alcune minime variazioni potrebbero emergere, per esempio a causa di differenti elementi grafici messi a disposizione da Figma rispetto che da Flutter.

### 2.5.1 Homepage e Dettagli di un Evento



(a) Homepage



## 2.5.2 Login e registrazione dell'utente

The image consists of two side-by-side screenshots of a mobile application interface, labeled (a) and (b).

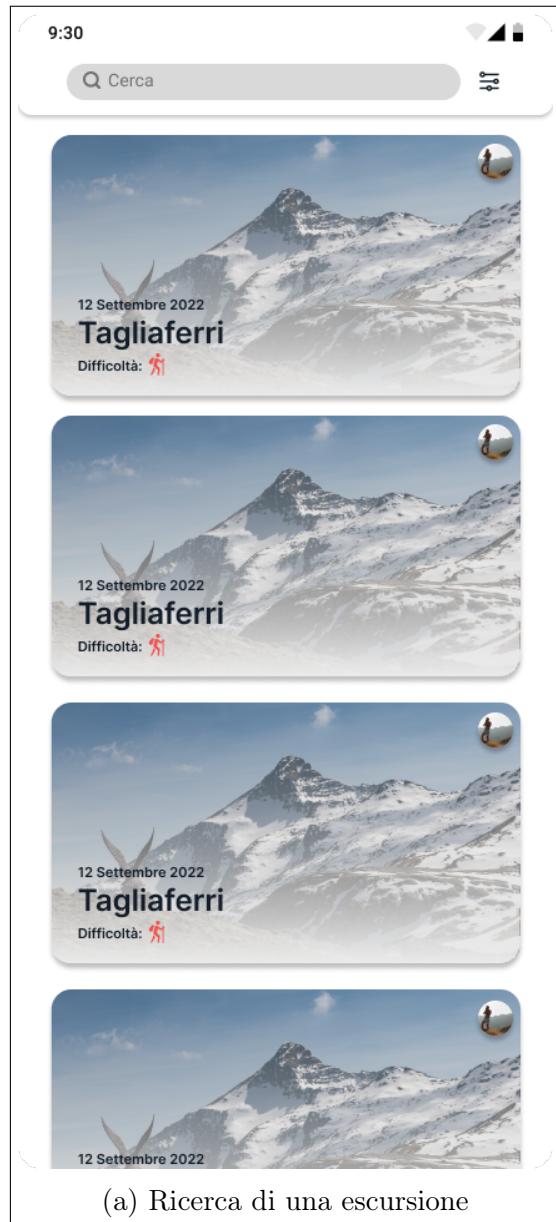
**Screenshot (a) Signup:** The screen title is "Crea Account". It includes fields for "Nome" (Name), "Email", "Telefono" (Phone), and "Password". Each field has an input placeholder and a clear icon. A large blue "Crea Account" button is at the bottom, and a link "Hai già un account? [Login](#)" is at the bottom right.

**Screenshot (b) Login:** The screen title is "Login". It includes fields for "Email Address" and "Password". Each field has an input placeholder and a clear icon. A "Forgot Password?" link is located below the password field. A large blue "Login" button is at the bottom, and a "Crea Account" link is at the bottom right.

**(a) Signup tramite mail**

**(b) Login**

### 2.5.3 Ricerca escursione e Filtraggio



#### 2.5.4 Visualizzazione del profilo utente

Cristian Tironi  
Livello: Marmotta  
17 amici

**Esperienze**

In programma Passate

12 Settembre 2022  
**Tagliaferri**  
Difficoltà: 🚶

12 Settembre 2022  
**Tagliaferri**  
Difficoltà: 🚶

12 Settembre 2022  
**Tagliaferri**  
Difficoltà: 🚶

(a) Profilo utente normale

Cristian Tironi  
Organizzatore 4.7 ★  
200 amici

**Esperienze**

In programma Passate

12 Settembre 2022  
**Tagliaferri**  
Difficoltà: 🚶 4.7 ★

12 Settembre 2022  
**Tagliaferri**  
Difficoltà: 🚶 4.7 ★

12 Settembre 2022  
**Tagliaferri**  
Difficoltà: 🚶

(b) Profilo organizzatore

# Capitolo 3

## Iterazione 2

### 3.1 Introduzione

Nella seconda iterazione sono stati implementati i seguenti casi d'uso:

- UC1 - Login;
- UC2 - Signup;
- UC3 - Logout;
- UC8 - Visualizzazione risultati della ricerca;
- UC9 - Visualizzazione di uno specifico evento;
- UC10 - Prenotazione a un nuovo evento;
- UC11 - Cancellazione da un evento;
- UC17 - Eliminazione di un evento;
- UC18 - Creazione di un evento.

Si è deciso di sviluppare anzitutto quei casi d'uso identificati come ad alta priorità in modo da definire le fondamenta della piattaforma, sia lato client che server. Segue una presentazione dettagliata di ognuno di loro. L'integrazione del backend con il frontend attraverso un API Gateway è stata riservata per la successiva iterazione.

## 3.2 Component Diagram

Il component diagram aggiornato mostra i componenti e le interfacce relative al sistema di gestione degli eventi.

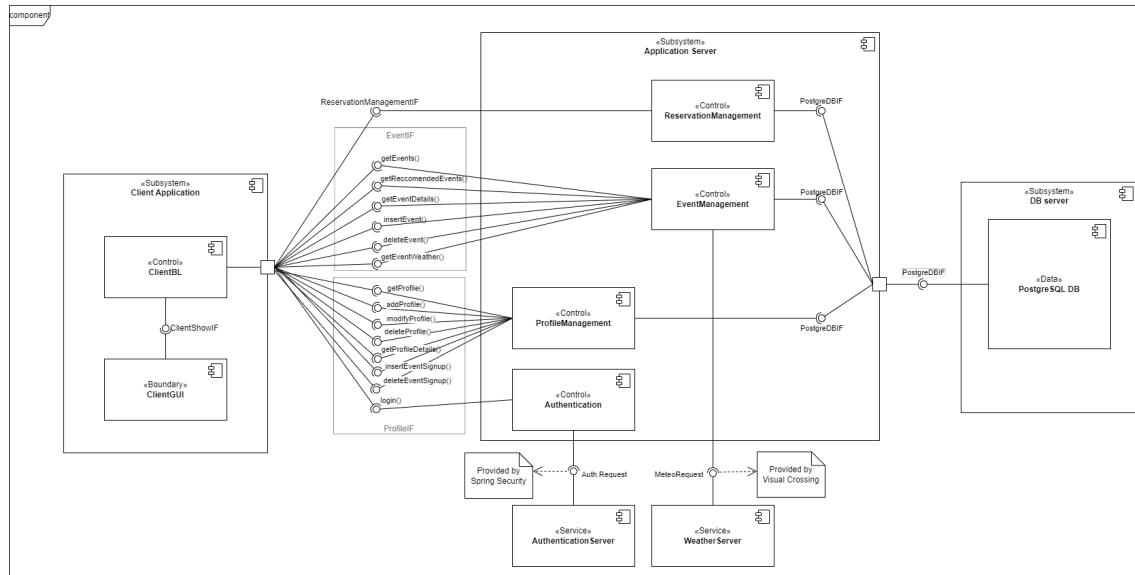


Figura 3.1: Component Diagram.

### 3.3 DataClass Diagram

Il Class Diagram relativo all'iterazione 2 mostra come le principali entità si sono evolute nel corso dell'iterazione 2 a seguito dell'introduzione degli use case ad alta priorità. In particolare, per ciascuna entità è stato aggiunto il relativo Data Transfer Object, impiegato nella comunicazione tra client e server.

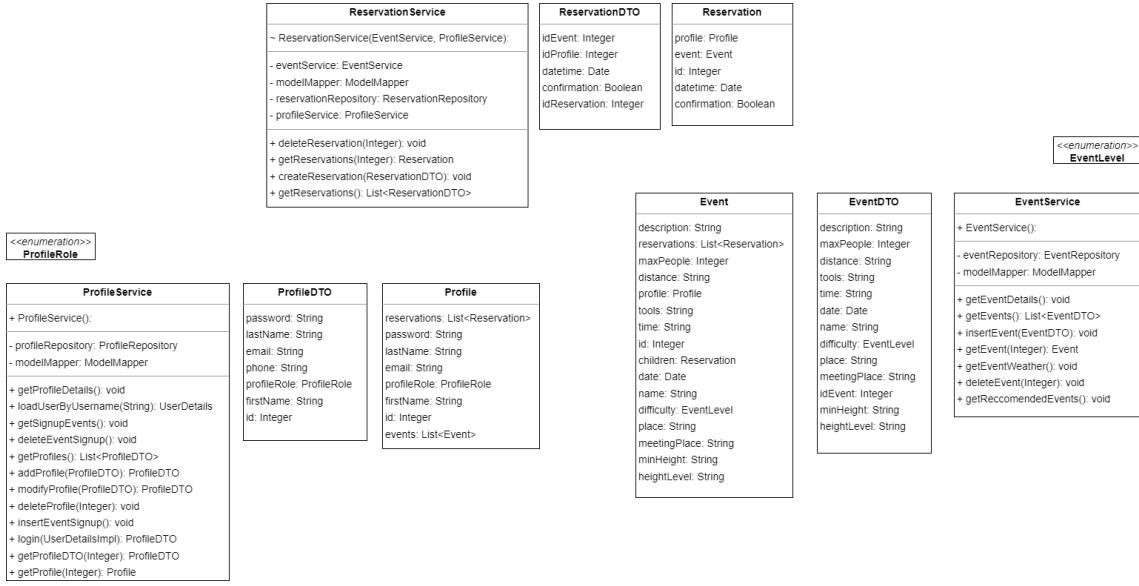


Figura 3.2: Class Diagram.

## 3.4 Interface Diagram

Il diagramma in Figura 3.3 mostra l’evoluzione delle interfacce e delle funzionalità offerte. Nell’iterazione 2 è stata aggiunta l’interfaccia ReservationManagementIF, mentre EventsManagementIF e ProfileManagementIF hanno subito modifiche legate all’implementazione degli use case previsti per l’iterazione 2.

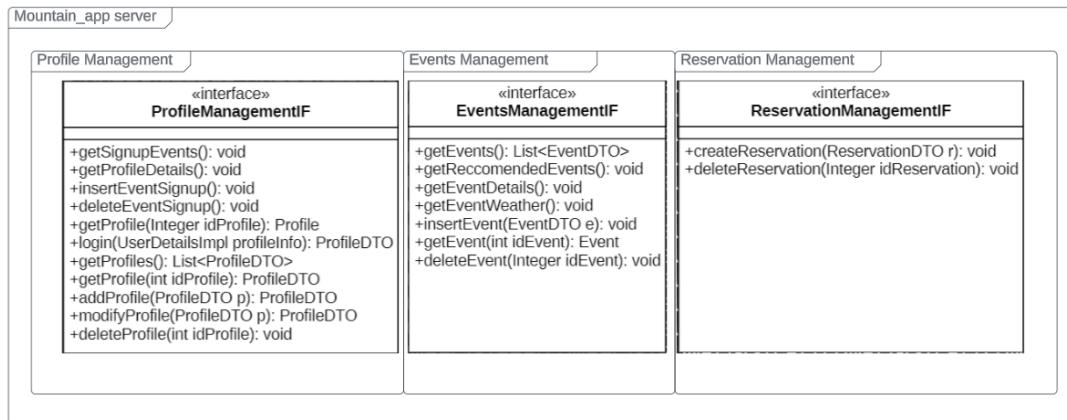


Figura 3.3: Interface Diagram.

## 3.5 Use Case

Dettaglio dell'implementazione dei requisiti ritenuti fondamentali per il funzionamento dell'applicativo, ossia la creazione dei profili utente, la creazione di nuovi eventi ed il sistema di iscrizione agli eventi stessi. Saranno i primi use case implementati sia lato client che server.

### 3.5.1 UC-1 Login

Per l'autenticazione abbiamo utilizzato il framework Spring Security con il metodo “Basic Access Authentication”: si tratta di una tecnica che non necessita dell'utilizzo di cookie o di mantenere una sessione tra client e server, ma utilizza gli header HTTP per fornire le informazioni di accesso. I campi username e password vengono codificati con base64 e sono poi trasmessi nell'header ogni volta che viene chiamata una API. Il sistema, prima di elaborare una richiesta, verifica che lo username e la password trasmessi appartengano effettivamente ad un utente presente nel database.

*Breve descrizione:* l'utente compila il form per eseguire il login: in caso di credenziali corrette il sistema consente l'accesso ai servizi, altrimenti notifica l'utente della non correttezza delle credenziali.

*Attori coinvolti:* Utente, sistema.

*Precondizione:* l'utente è registrato nel sistema e apre la app.

*Postcondizione:* l'utente accede alla app (in caso le credenziali siano corrette) oppure viene avvertito che le credenziali sono sbagliate.

*Procedimento:*

1. il sistema richiede all'utente le informazioni di accesso: username e password;
2. l'utente inserisce le informazioni di accesso;
3. il sistema controlla le informazioni fornite;
4. le informazioni sono corrette. [E1: le informazioni sono sbagliate].
5. l'utente viene indirizzato alla homepage dell'applicazione.

*Eccezioni:*

- E1:
  1. le informazioni sono sbagliate;
  2. il sistema comunica all'utente che le informazioni inserite non sono corrette;
  3. ritorno al passo 1 di “Procedimento”.

### 3.5.2 UC-2 Signup

*Breve descrizione:* Un nuovo utente compila il form per iscriversi alla piattaforma. Se è in possesso di codice organizzatore è necessario che lo inserisca affinché venga registrato come organizzatore.

*Attori coinvolti:* Utente, Sistema.

*Precondizione:* Il nuovo utente è nella pagina di registrazione raggiungibile dalla pagina di login mostrata all'apertura della app.

*Postcondizione:* Il nuovo utente viene inserito nel database con il relativo ruolo e può accedere ai servizi.

*Procedimento:*

1. Il nuovo utente fornisce le seguenti informazioni:
  - Nome;
  - Cognome;
  - Email;
  - Telefono;
  - Codice organizzatore (se ne possiede uno);
  - Password;
2. Dalla pagina di login il nuovo utente seleziona “registrati”;
3. L'utente compila i campi obbligatori;
4. Il sistema verifica che tutti i campi obbligatori siano stati compilati [E1: ci sono dei campi vuoti];
5. Il sistema aggiunge l'utente con i rispettivi dati nel database [E2: la mail è già associata ad un account];

6. Se l'utente inserito dispone di un codice organizzatore valido, il sistema imposta l'utente come caposquadra della squadra selezionata [E3: codice organizzatore non è valido].

*Eccezioni:*

- E1:
  1. Il form non viene mandato;
  2. Viene notificato un errore all'utente;
  3. Ritorno al passo 3 di “Procedimento”.
- E2:
  1. il sistema comunica al nuovo utente che i dati sono già associati ad un altro utente;
  2. ritorno al passo 3 di “Procedimento”.
- E3:
  1. il sistema comunica al nuovo utente che il suo codice organizzatore non è valido;
  2. ritorno al passo 3 di “Procedimento”;
  3. L'utente può inserirne uno corretto o registrarsi come normale utente.

### 3.5.3 UC-3 Logout

Il logout dell'utente consiste nel rimandarlo alla pagina di login per richiedere nuovamente username e password. In questo caso non c'è nessuna sessione tra client e server, di conseguenza questo caso d'uso viene gestito interamente lato client.

*Breve Descrizione: Il sistema esegue il logout dell'utente rimandandolo alla pagina di login/registrazione*

*Attori Coinvolti: Sistema, utente*

*Precondizione: L'utente è autenticato nella app*

*Postcondizione: L'utente viene rimandato alla pagina di login e non è più loggato nel sistema*

*Procedimento:*

1. L'utente naviga alla pagina del suo profilo;
2. L'utente tocca il tasto in alto a destra;
3. Il sistema rimanda l'utente alla pagina di login.

### 3.5.4 UC-7 Ricerca Evento

Dalla app l'utente può cercare un evento in base al nome, viene mostrata una vista con l'evento dal nome scelto, se viene trovato.

*Breve Descrizione: Il sistema cerca nella lista degli eventi scaricati non ancora conclusi per una corrispondenza del nome. Tutto Lo use case si sviluppa offline*

*Attori Coinvolti: Sistema, utente*

*Precondizione: L'utente è registrato nell'app*

*Postcondizione: L'utente visualizza l'evento che ha cercato*

*Procedimento:*

1. Selezionare l'icona di ricerca in alto a destra;
2. Scrivere il nome dell'evento da cercare;
3. Se il nome è presente viene mostrato dal sistema.

*Eccezioni:*

- E1: L'evento con il nome dato non esiste
  1. Viene mostrato all'utente un messaggio di errore;
  2. Viene data la possibilità di riprovare o tornare alla homepage;

### 3.5.5 UC-8 Visualizzazione risultati della ricerca

Quando vengono trovati uno o più eventi tra quelli in programma viene mostrata una lista di tutti loro.

*Breve Descrizione:* L'utente effettua la ricerca di uno o più eventi in base al nome e gli viene mostrata una lista con le corrispondenze

*Attori Coinvolti:* Sistema, utente

*Precondizione:* L'utente è registrato nell'app

*Postcondizione:* L'utente visualizza una lista dei risultati

*Procedimento:*

1. Cercare un evento così come descritto in UC-7;
2. La lista degli eventi viene mostrata nella parte inferiore dello schermo;
3. Se la lista contiene molti elementi può essere fatta scorrere in verticale.

*Eccezioni:*

- E1: L'evento con il nome dato non esiste
  1. Viene mostrato all'utente un messaggio di errore;
  2. Viene data la possibilità di riprovare o tornare alla homepage;

### 3.5.6 UC-9 Visualizzazione di uno specifico evento

L'utente può visualizzare informazioni dettagliate riguardo al singolo evento

*Breve Descrizione:* Dalla schermata principale, è possibile visualizzare nel dettaglio i singoli risultati prodotti dalla ricerca.

*Attori Coinvolti:* Sistema, utente

*Precondizione:* L'utente è registrato nell'app

*Postcondizione:* Vengono mostrati a schermo tutti i dettagli del singolo evento

*Procedimento:*

1. Cercare un evento;
2. Selezionare quello d'interesse tra i risultati mostrati;

*Eccezioni:*

- E1: La prenotazione risulta già eliminata
  1. Viene mostrato all'utente un messaggio di errore;

### 3.5.7 UC-10 Prenotazione a un nuovo evento

L'utente può prenotarsi a un evento dalla pagina che dettaglia le specifiche di quello a cui è interessato.

*Breve Descrizione: Aprendo un evento l'utente visualizza un bottone in alto a destra che gli permette di iscriversi a un evento. Il suo codice utente viene inviato e viene aggiunto alla lista delle prenotazioni.*

*Attori Coinvolti: Utente, Sistema*

*Precondizione: L'utente è loggato nella app e si trova sulla pagina che dettaglia le specifiche dell'evento*

*Postcondizione: L'utente prenota il suo posto per un evento*

*Procedimento:*

1. Dalla homepage selezionare un evento a cui si è interessati;
2. Dalla pagina del dettaglio, in alto a destra, premere il bottone “Prenotati a questa escursione”;
3. Viene visualizzata la conferma o un errore nel caso di problemi di connessione  
[E1: problemi di connessione]

*Eccezioni:*

- E1:
  1. L'utente viene notificato del problema;
  2. Viene proposto di riprovare;
  3. Ritorna al passo 2 di “Procedimento”.

### 3.5.8 UC-11 Cancellazione da un evento

L'utente può eliminare la propria partecipazione a un evento.

*Breve Descrizione:* L'utente visualizza lo specifico evento dalla sezione dedicata alle iscrizioni. Aperta la singola iscrizione, se ne può richiedere l'eliminazione

*Attori Coinvolti:* Sistema, utente

*Precondizione:* L'utente è registrato nell'app

*Postcondizione:* L'utente viene eliminato dai partecipanti all'evento

*Procedimento:*

1. Selezionare la prenotazione;
2. Selezionare il bottone "Disiscriviti da questa escursione";

*Eccezioni:*

- E1: La prenotazione risulta già eliminata
  1. Viene mostrato all'utente un messaggio di errore;

### 3.5.9 UC-15 Algoritmo per selezionare i partecipanti

Selezione degli utenti iscritti ad un evento con strategia Greedy.

*Breve Descrizione:* Se il numero di iscritti ad un evento eccede il numero massimo di partecipanti, vengono selezionati in base alla loro esperienza.

*Attori Coinvolti:* Sistema, Organizzatore

*Precondizione:* L'utente organizzatore avvia la funzione di selezione degli utenti iscritti ad un proprio evento.

*Postcondizione:* Gli utenti non selezionati vengono eliminati dai partecipanti dell'evento e viene restituita una lista degli utenti confermati.

*Procedimento:*

1. L'utente organizzatore accede all'app e seleziona un evento che ha creato;
2. L'opzione 'Seleziona partecipanti' viene selezionata.
3. L'algoritmo viene applicato sulla lista degli iscritti all'evento.

*FlowChart (3.4) e pseudocodice: L'approccio si basa sull'idea di selezionare in modo iterativo gli utenti con il livello più adatto, iniziando dal livello dell'evento e ampliando la ricerca ai livelli adiacenti solo se necessario.*

```

1 algoritmo selezionaIscritti(Lista listaIscritti)
2     S <- {}
3     limiteMax = Event.maxPeople
4
5     if (listaIscritti.dim <= limiteMax) then S <-
6         all(listaIscritti)
7     else
8         livello = Event.level
9         int i = 0 //posti occupati
10        int j = 0 //distanza dal livello evento
11
12        while((i != limiteMax) and (listaIscritti != {})) do
13            utenteConfermato = seleziona(listaIscritti,
14                livello, j)
15
16            if(utenteConfermato != null) {
17
18                listaIscritti <- listaIscritti /
19                    {utenteConfermato}
20
21                S <- S U {utenteConfermato}
22
23                i + 1
24
25            }
26
27            if(not listaIscritti.contains(anotherUser con
28                userLevel = livello +- j)) then j+1
29
30
31        return S
32
33
34 //prima vengono considerati gli utenti con lo stesso livello
35 //dell'evento
36
37 //poi quelli ai livelli +-1, +-2 fino a esaurimento posti

```

```

26  funzione seleziona(listaIscritti, livello, distanza)
27      return (User con userLevel = livello + distanza) or (User
        con userLevel - distanza)

```

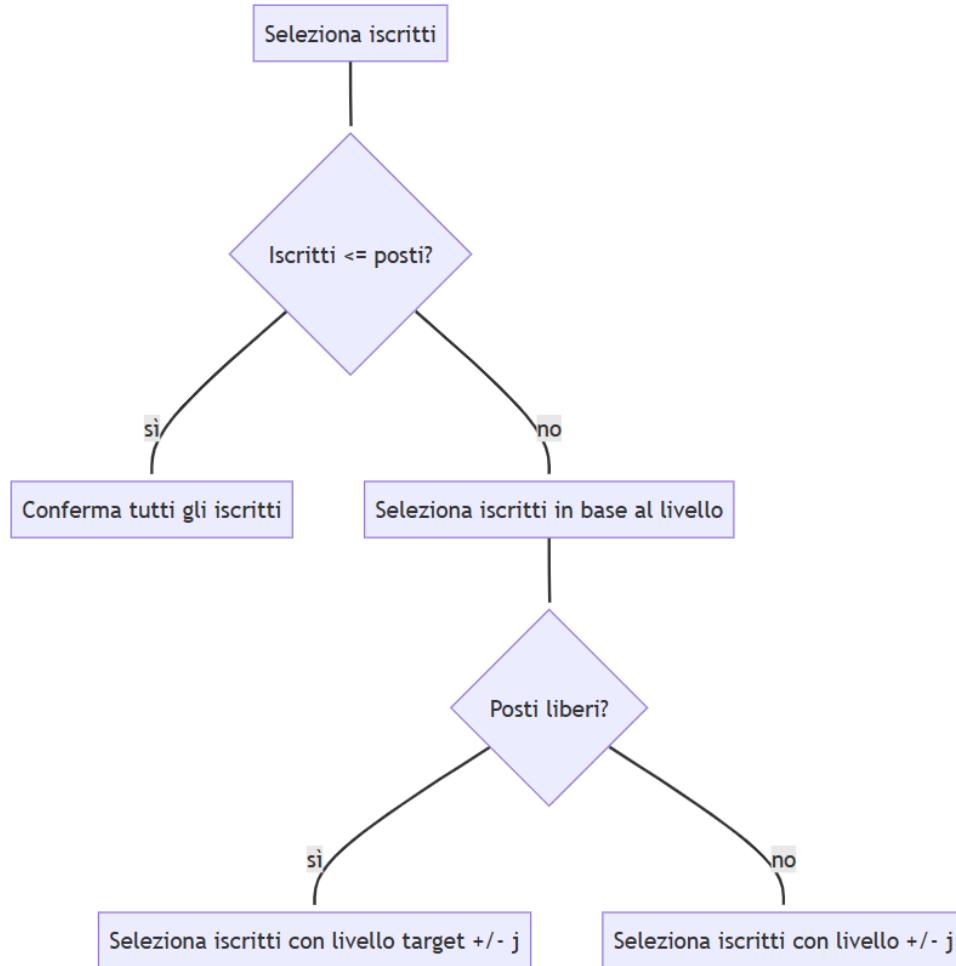


Figura 3.4: FlowChart

*Analisi di Complessità:* l'algoritmo ha una complessità temporale più significativa rispetto a quella spaziale, la sua efficienza dipende dal rapporto tra il numero massimo di posti disponibili e la dimensione totale della lista degli iscritti.

### 1. Complessità temporale:

- Caso Migliore (Numero di iscritti  $\leq$  Numero massimo di posti): Nel caso in cui il numero di iscritti sia inferiore o uguale al numero massimo di posti disponibili, l'algoritmo esegue una copia di tutti gli iscritti nella

lista ‘S‘. Complessità Temporale:  $O(n)$ , dove n è la dimensione della lista degli iscritti.

- Caso Peggio (Numero di iscritti > Numero massimo di posti): Nel caso in cui il numero di iscritti superi il numero massimo di posti, l’algoritmo utilizza due cicli while annidati. Il ciclo esterno viene eseguito fino a quando non vengono occupati tutti i posti desiderati o la lista degli iscritti è vuota, mentre il ciclo interno verifica la presenza di utenti con il livello desiderato nella lista degli iscritti. Nel peggio dei casi, il numero di iterazioni del ciclo esterno è limitato dal numero massimo di posti e dalla dimensione della lista degli iscritti. Complessità Temporale:  $O(\text{limiteMax} * n)$ , dove n è la dimensione della lista degli iscritti.

2. Complessità spaziale:

- Spazio Ausiliario (Variabili e Strutture Dati): La lista ‘S‘ contiene gli iscritti selezionati. Nel caso peggiore, sarà di dimensione ‘`limiteMax`‘.
- Altre variabili ausiliarie occupano uno spazio costante.

Complessità Spaziale:  $O(\text{limiteMax})$ .

### 3.5.10 UC-17 Eliminazione di un evento

L'organizzatore di un evento può cancellarlo

*Breve Descrizione: Dalla scritta in basso al centro della pagina che dettagli un evento, il relativo organizzatore può eliminarlo*

*Attori Coinvolti: Utente, Sistema*

*Precondizione: L'utente è loggato nell'app ed ha in passato creato l'evento considerato*

*Postcondizione: L'evento viene cancellato*

*Procedimento:*

1. L'organizzatore naviga alla pagina dei dettagli dell'evento che vuole cancellare;
2. L'utente scorre la pagina del dettaglio fino in fondo;
3. L'utente preme “Cancella evento”;
4. L'evento viene rimosso dal database a meno di problemi di connessione [E1: Problemi di connessione];
5. Viene cancellata la prenotazione per ogni utente che aveva espresso interesse

*Eccezioni:*

- E1:

1. L'utente viene notificato del problema;
2. Viene proposto di riprovare;
3. Ritorna al passo 2 di “Procedimento”.

### 3.5.11 UC-18 Creazione di un evento

L'utente di tipo organizzatore ha la possibilità di organizzare un nuovo evento, a cui altri utenti potranno richiedere di partecipare.

*Breve Descrizione:* Dalla schermata principale è possibile selezionare l'icona crea e completare il form che viene visualizzato. Cliccando "submit" verrà aggiunto un nuovo evento

*Attori Coinvolti:* Sistema, utente

*Precondizione:* L'utente è registrato nell'app ed è un organizzatore

*Postcondizione:* L'evento viene pubblicato sulla piattaforma

*Procedimento:*

1. Selezionare l'icona crea dalla schermata principale;
2. Completare i campi che vengono visualizzati;
3. Selezionare il pulsante "Aggiungi Evento";

*Eccezioni:*

- E1: Esiste già un evento con il medesimo nome
  1. Viene mostrato all'utente un messaggio di errore;

## 3.6 Documentazione API

In questa sezione vengono mostrate alcune delle API sviluppate per l'iterazione 2. È possibile visualizzare tutte le API mediante la collezione di Postman presente nella repository GitHub. Tra tutte le API sviluppate, le più significative offrono le seguenti funzionalità.

- API per la creazione di un evento.
- API per la creazione di una prenotazione e la relativa eliminazione.
- API per la visualizzazione di prenotazioni di uno specifico utente.
- API per la registrazione di un profilo.

**POST new event**

```
http://localhost:8080/events/new
```

```
{
  "id": "1",
  "name": "test",
  "place": "test",
  "difficulty": "MEDIUM",
  "date": "2024-01-19",
  "description": "test",
  "distance": "test",
  "heightLevel": "test",
  "minHeight": "test",
  "tools": "test",
  "meetingPlace": "test",
  "maxPeople": 4,
  "time": "test"
}
```

Figura 3.5: Aggiunta di un nuovo evento

**POST new reservation**

```
http://localhost:8080/events/reservation
```

```
{
  "idProfile": 2,
  "idEvent": 39,
  "datetime": "2024-01-14T19:21:07.000+00:00",
  "confirmation": false
}
```

Figura 3.6: Inserimento nuova prenotazione

**GET get reservation**

```
http://localhost:8080/profile/reservations/18
```

Figura 3.7: Prenotazione dato uno specifico profilo

**POST new profile**

<http://localhost:8080/profiles>

```
{  
    "firstName": "admin",  
    "lastName": "admin",  
    "email": "admin@admin.com",  
    "phone": "+1234567890",  
    "password": "admin",  
    "profileRole": "USER"  
}
```

Figura 3.8: Creazione nuovo profilo

## 3.7 Testing

### 3.7.1 Analisi statica

Per l'analisi statica del codice Java è stato utilizzato il tool STAN4J, il seguente è il report generato a partire dal nostro codice.

# Quality Report

Creation Date 2024-01-17

Level of Detail Member

## Tmp\_240117\_134748

### Library Dependency Graph

/project/src/main/java /project/src/main/resources /project/src/test/java

### Treemap Overview



### Metrics Summary

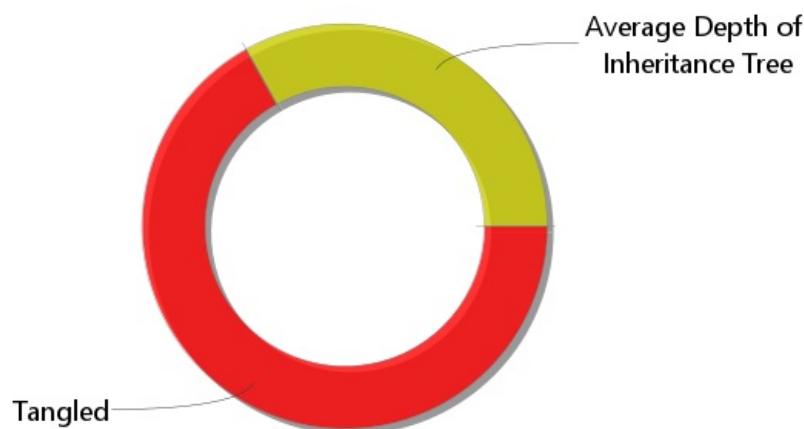
Metric	Value
Number of Libraries	3
Number of Packages	7
Number of Top Level Classes	26
Average Number of Top Level Classes per Package	3.71
Average Number of Member Classes per Class	0
Average Number of Methods per Class	8.42
Average Number of Fields per Class	3.38
Estimated Lines of Code	922
Estimated Lines of Code per Top Level Class	35.46
Average Cyclomatic Complexity	0.96
Fat for Library Dependencies	0
Fat for Flat Package Dependencies	9
Fat for Top Level Class Dependencies	58
Tangled for Library Dependencies	0%
Average Component Dependency between Libraries	0%
Average Component Dependency between Packages	28.57%
Average Component Dependency between Units	26.15%
Average Distance	-0.15
Average Absolute Distance	0.15
Average Weighted Methods per Class	8.12
Average Depth of Inheritance Tree	0.88
Average Number of Children	0
Average Coupling between Objects	1.15
Average Response for a Class	10.15
Average Lack of Cohesion in Methods	42.77

### Top Violations (2 of 2)

Artifact	Metric	Value
----------	--------	-------

## Pollution Chart

Pollution 1.12



## Violations by Metric

### Tangled

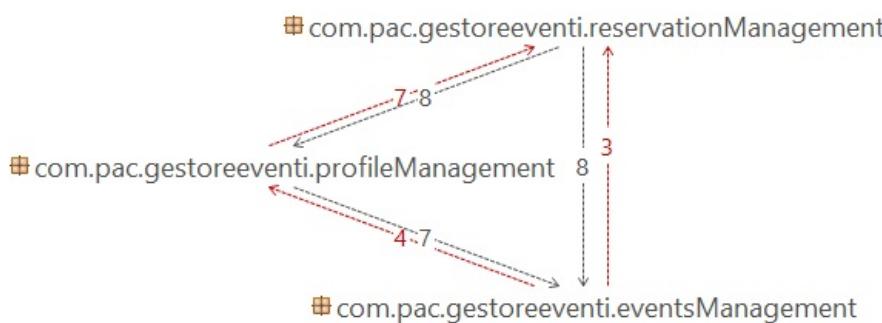
Artifact	Value
com.pac.gestoreventi	26.42%

### Average Depth of Inheritance Tree

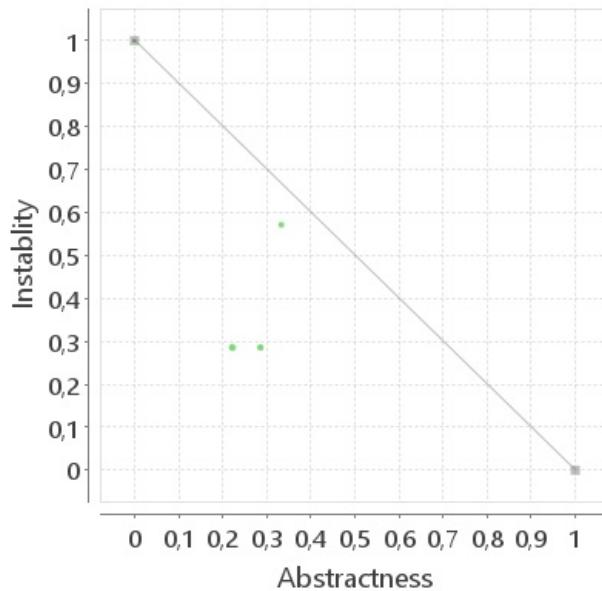
Artifact	Value
Tmp_240117_134748	0.88

## Design Tangles

File Tangle inside com.pac.gestoreventi (#nodes=3, #edges=6, weight=37, fas size=3, weight=14)



## Package Distance Chart



## Metric Ratings

### Count Metrics

Metric	Rating	Linear
Number of Top Level Classes	20 40 60 80	✓
Number of Methods	25 50 100 200	✓
Number of Fields	10 20 40 80	✓
Estimated Lines of Code	200 200 400 500	✓
Estimated Lines of Code	20 50 100 200	✓

### Complexity Metrics

Metric	Rating	Linear
Cyclomatic Complexity	10 15 20 20	✓
Fat	20 60 120 240	✓
Fat	20 60 120 240	✓
Tangled	0 1 1 1	✓
Tangled for Library Dependencies	0 1 1 1	✓
Average Component Dependency between Libraries	0 5 1 1	✓
Average Component Dependency between Packages	0 5 1 1	✓

### Robert C. Martin Metrics

Metric	Rating	Linear
Distance	1 5 0 5 1	✓
Average Absolute Distance	0 1 5 1	✓

### Chidambar & Kemerer Metrics

Metric	Rating	Linear
Weighted Methods per Class	0 100 200 300	✓
Depth of Inheritance Tree	1 5 0 10	✓
Average Depth of Inheritance Tree	2 1 0 0	✓
Coupling between Objects	0 25 250 1000	✓
Response for a Class	0 100 1000	✓

### 3.7.2 Analisi dinamica

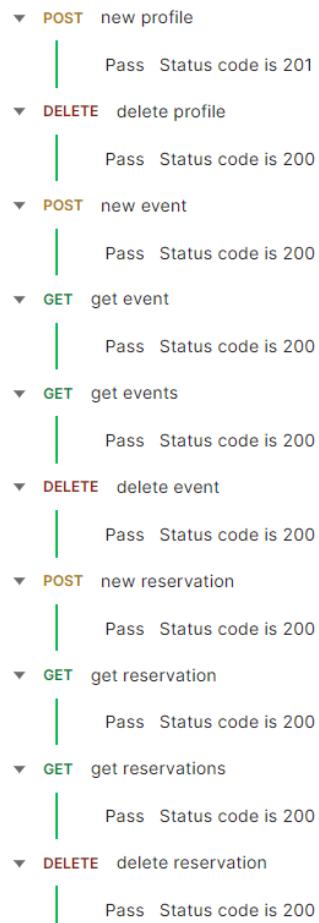
Nell’iterazione 2 sono state testate tutte le API REST implementate, utilizzando Postman . In particolare sono state testate le seguenti funzionalità:

- ProfileController:
  - Eliminazione profilo;
  - Inserimento di un nuovo profilo e verifica che i dati del profilo siano stati inseriti in modo corretto;
- EventController:
  - Visualizzazione di un evento e verifica che le informazioni ricevute siano corrette;
  - Visualizzazione di tutti gli eventi inseriti nel sistema;
  - Eliminazione di un evento;
  - Inserimento di un evento;
- ReservationController:
  - Visualizzazione di una reservation e verifica che le informazioni ricevute siano corrette;
  - Visualizzazione di tutte le reservation inserite nel sistema;
  - Inserimento di una nuova reservation;
  - Eliminazione di una reservation;

## RUN SUMMARY

		1
▶ <b>POST</b>	new profile	1   0
▶ <b>DELETE</b>	delete profile	1   0
▶ <b>POST</b>	new event	1   0
▶ <b>GET</b>	get event	1   0
▶ <b>GET</b>	get events	1   0
▶ <b>DELETE</b>	delete event	1   0
▶ <b>POST</b>	new reservation	1   0
▶ <b>GET</b>	get reservation	1   0
▶ <b>GET</b>	get reservations	1   0
▶ <b>DELETE</b>	delete reservation	1   0

Figura 3.9: Test collection



### 3.7.3 Unit test

Per questa iterazione è stata testata la funzione findById presente nel backend, che serve a recuperare un evento tramite id.

```
1  @DataJpaTest
2
3
4  @Autowired
5  private EventRepository underTest;
6
7  @Test
8  void findById() {
9
10    String stringaTest = "test";
11    //given
12    Date date = new Date(100);
13    Event expected = new Event(stringaTest, stringaTest,
14      EventLevel.MEDIUM, date, stringaTest, stringaTest,
15      stringaTest, stringaTest, stringaTest, stringaTest,
16      stringaTest, new Integer(4));
17    underTest.save(expected);
18
19    // when
20    Event result = underTest.findById(expected.getId()).get();
21
22    // then
23    assertThat(expected.getDescription())
24      .isEqualTo(result.getDescription());
25  }
26 }
```

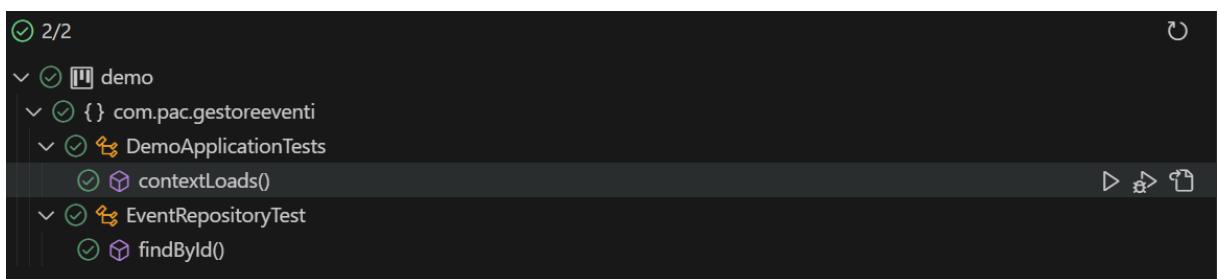


Figura 3.10: findById()

# Capitolo 4

## Iterazione 3

### 4.1 Introduzione

Durante la terza iterazione ci siamo concentrati principalmente sulla creazione di un gateway che mettesse in comunicazione l'applicazione mobile con il backend creato nella precedente iterazione. Abbiamo inoltre implementato, attraverso le GitHub Actions, un meccanismo di Continuous Integration e Deployment per gestire in automatico le release della nostra architettura. Infine abbiamo lavorato sugli use case a media priorità e abbiamo aggiornato alcune parti della documentazione e dei diagrammi scritti nelle precedenti iterazioni.

## 4.2 Continuous Integration e Continuous Deployment (CI/CD)

Attraverso le GitHub Actions abbiamo implementato un meccanismo di Continuous Integration e Deployment che ci permette di lavorare sul nostro codice sorgente e non dover provvedere manualmente alla compilazione e pubblicazione dell'immagine Docker con la quale abbiamo pubblicato il nostro backend. La parte di CI avviene direttamente su GitHub, appunto grazie alle Actions, mentre la parte di CD avviene sulla macchina virtuale di Digital Ocean che abbiamo usato per pubblicare l'architettura.

### 4.2.1 Continuous Integration (CI)

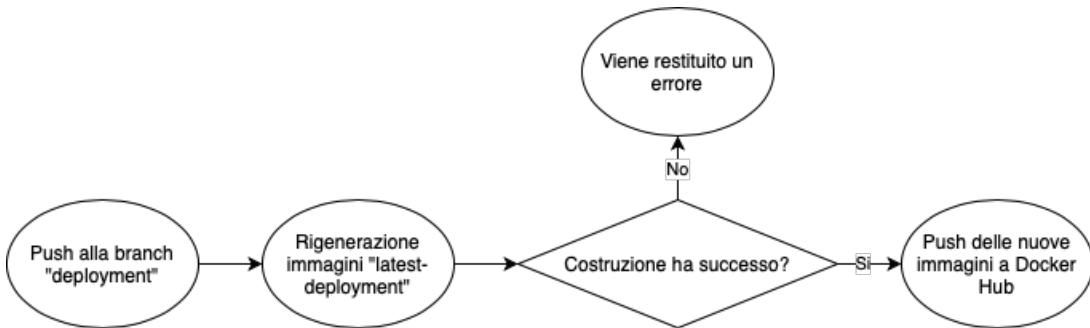


Figura 4.1: Continuous Integration della branch main

Il nostro backend è organizzato in 2 immagini:

1. “ventura-boulevard”, contenente il manager degli utenti, degli eventi e delle prenotazioni;
2. “radio-nowhere”, contenente l’API Gateway usato per comunicare con la mobile app.

Entrambe le immagini sono organizzate in 2 tag:

1. “latest-development”, l’ultima versione del codice usato per lo sviluppo;
2. “latest-deployment”, l’ultima versione del codice usato in produzione e pubblicato su Digital Ocean.

## 4.2. CONTINUOUS INTEGRATION E CONTINUOUS DEPLOYMENT (CI/CD)65

Infine la nostra repository è organizzata attorno a 2 branch principali (oltre a quelle sviluppate al bisogno da ogni membro, ma che non vengono gestite dal sistema CI/CD):

1. “main”, rappresentante la nostra branch di sviluppo, dove poniamo il codice a cui stiamo ancora lavorando ma che non vogliamo pubblicare sul server di produzione;
2. “production”, la nostra branch di produzione, dove è contenuto il codice da cui sono generate le immagini attualmente pubblicate sul server di produzione.

Ad ogni push alla branch main o production vengono rigenerate e ripubblicate su Docker Hub, rispettivamente, l’immagine taggata come “latest-development” e “latest-production”. Per pubblicare l’immagine compilata sul relativo registry di Docker Hub è necessario accedere con le credenziali del proprietario. Per permettere di usarle senza che vengano mai divulgare al pubblico, GitHub mette a disposizione “Secrets”, che possiamo vedere come un dizionario dove ogni coppia chiave-valore rappresenta una costante che può essere richiamata all’interno delle Actions attraverso la sua chiave. Una volta creata una entry non è più possibile, neppure per l’autore, vederne il contenuto, ma unicamente modificarla o eliminarla.

#### 4.2.2 Continuous Deployment (CD)

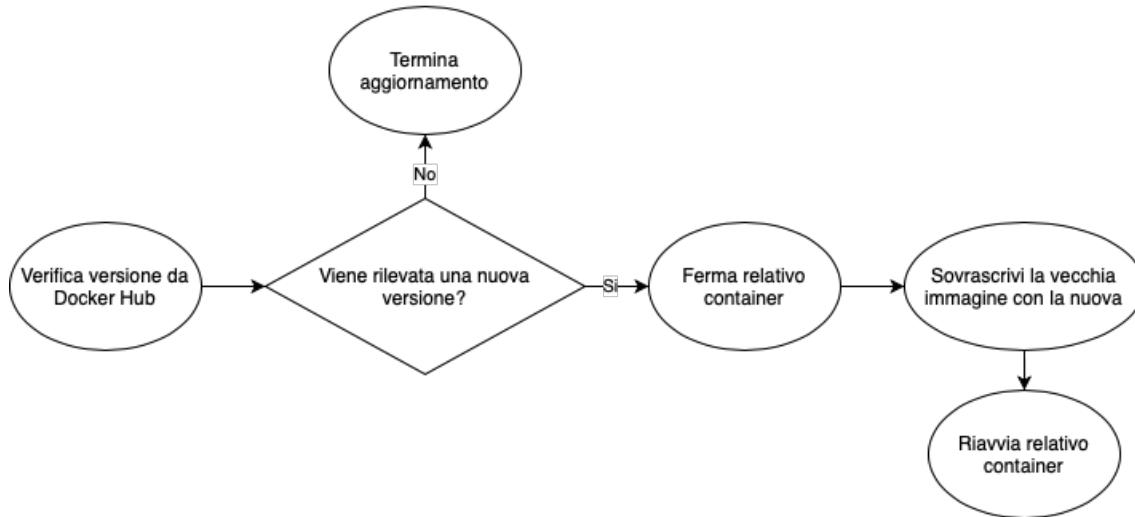


Figura 4.2: Continuous Deployment di un immagine

Una volta che l’immagine su Docker Hub è stata rigenerata è necessario che questi cambiamenti vengano rispecchiati sul server di produzione. Per fare ciò abbiamo usato “Watchtower”, un programma open-source usato per gestire le immagini Docker salvate su una macchina aggiornandole in automatico quando viene pubblicata una nuova versione. Il controllo e l’aggiornamento avvengono a intervalli regolari impostabili dall’utente. Per evitare che il container venisse ricreato troppe volte abbiamo deciso di limitare l’aggiornamento ad una volta al giorno, verso mezzanotte. Rimane poi la possibilità di accedere alla macchina virtuale e riavviare manualmente il container, oppure, scelta meno consigliata, riavviare per intero il server.

## 4.3 API Gateway

### 4.3.1 Creazione dell'API Gateway

Per completare la nostra architettura e disporre di un modo per rispondere in maniera più efficace ed efficiente alle richieste del client abbiamo deciso di creare un API Gateway che facesse da tramite tra il backend scritto in Spring (il “sistema complesso”) e l’applicazione mobile scritta in Dart. Per fare ciò abbiamo optato per un terzo linguaggio di programmazione, NodeJS, poiché volevamo realizzare un componente che fosse veloce e leggero. Il maggior compito del nostro gateway è quello di interrograre il sistema complesso e comporre le risposte da inviare al client. Grazie al gateway abbiamo riscontrato molti vantaggi mentre lavoravamo all’architettura, tra questi troviamo:

- La possibilità di modellare la risposta proveniente dal server in modo da fornire particolari header necessari affinché le risposte fossero accettate dal client. Per esempio includendo l’header ‘Access-Control-Allow-Origin’, richiesto da Dart per accettare una risposta, ma che non era fornito dal sistema complesso;
- Permettere a client di ottenere oggetti composti dal sistema complesso che senza l’utilizzo del gateway avrebbero necessitato di numerose chiamate da parte del client. Un esempio è la chiamata di login, la quale, quando l’utente viene autenticato, restituisce informazioni su di esso come nome e cognome. Nella risposta fornita dal Manager degli Utenti tuttavia non viene fornito il livello di esperienza. È però disponibile una chiamata che soddisfa questa richiesta, dato l’id di un utente. Chiamando questo secondo metodo, dopo aver ottenuto i dati dell’utente, ci ha permesso di includere il campo “experience” nella risposta inviata al client. In maniera simile abbiamo popolato la lista delle escursioni passate e delle iscrizioni.

### 4.3.2 Testing del gateway

Per verificare il corretto funzionamento del gateway abbiamo scritto una serie di test per ognuna delle chiamate verificando che la risposta attesa fosse corretta. Seguono i risultati della test suite (4.3).

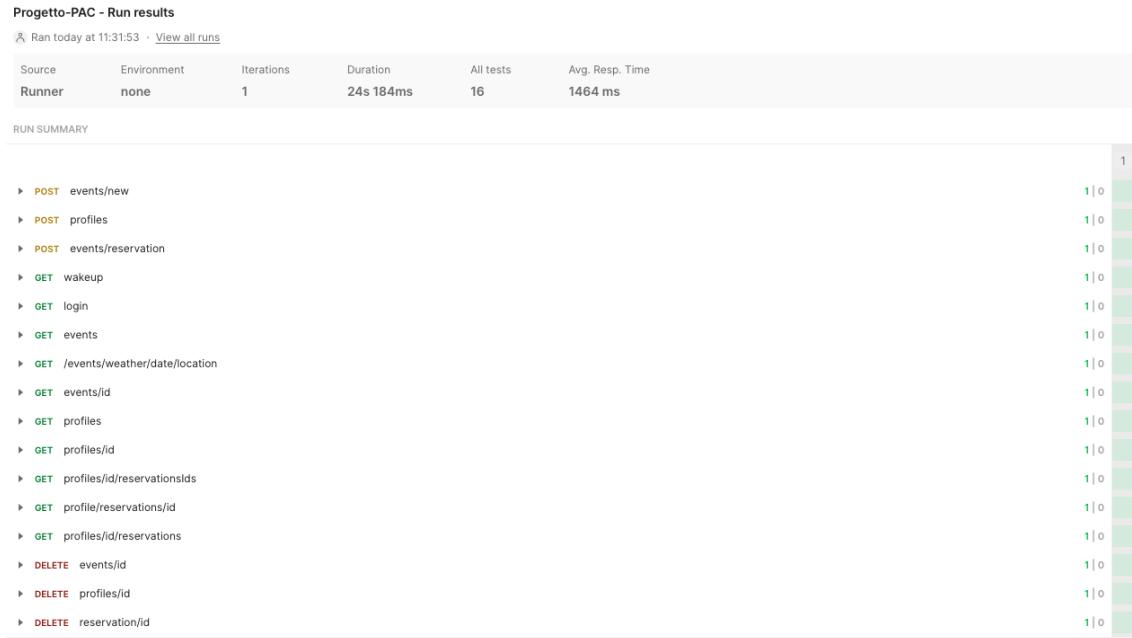


Figura 4.3: Risultati testing gateway

## 4.4 DataClass Diagram

Il Class Diagram relativo all'iterazione 3 mostra le modifiche relative all'implementazione nel Backend dell'algoritmo.

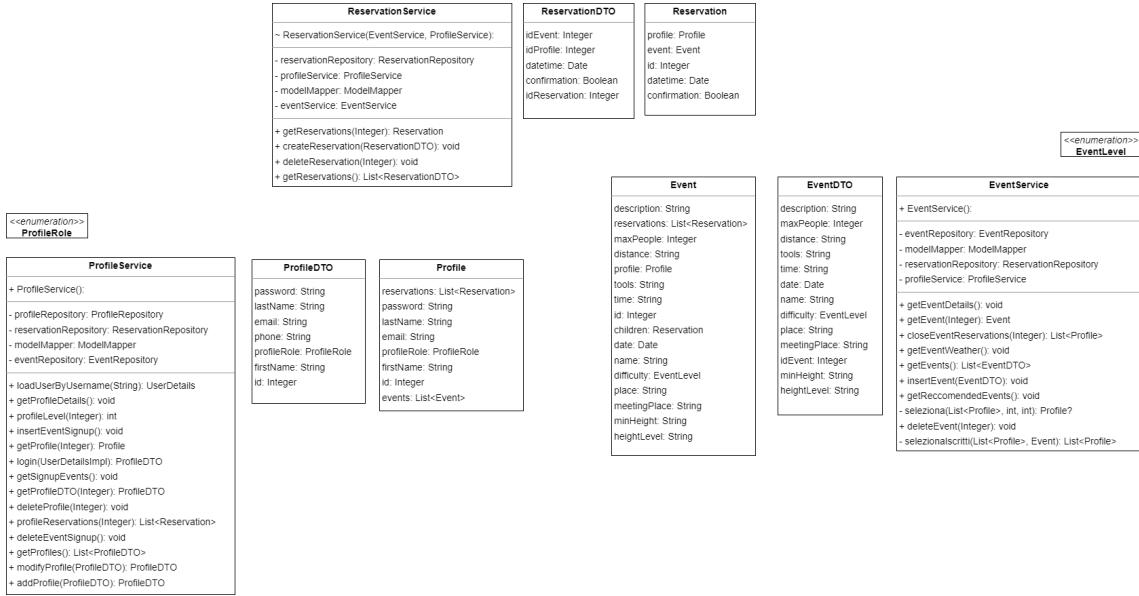


Figura 4.4: Class Diagram.

## 4.5 Component Diagram

Il component diagram mostra la struttura dell'applicativo e le componenti principali. In particolare, le API sono posizionate su un API gateway realizzato tramite un server NodeJS.

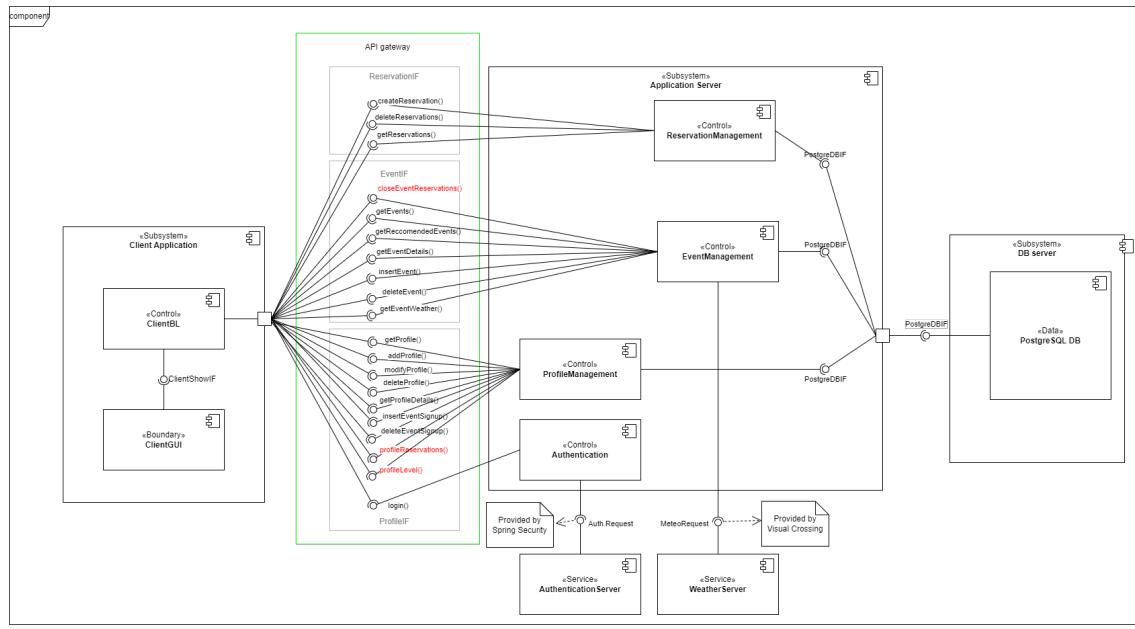


Figura 4.5: Component Diagram.

## 4.6 Documentazione API

In questa sezione vengono presentate le API sviluppate nel corso della terza iterazione. Come nell'iterazione precedente, le APIs sono visualizzabili nella collezione di postman esportata. In questa iterazione, le aggiunte principali sono state relative agli use case UC13, UC14 e all'implementazione dell'algoritmo. Infatti, come detto in precedenza, l'integrazione nel sistema ha comportato l'aggiunta di due funzionalità.

- API per la visualizzazione dei profili (utente/organizzatore)
- API per ottenere il livello di ogni utente
- API per chiudere le iscrizioni ad un evento e selezionare i partecipanti
- API per ottenere tutte le prenotazioni confermate di un partecipante

### GET profile level

```
http://localhost:8080/profiles/level/2
```

Figura 4.6: Profile level

### GET reservations

```
http://localhost:8080/profiles/2/reservations
```

Figura 4.7: Ottenere tutte le prenotazioni confermate di un utente

### GET get profile

```
http://localhost:8080/profiles/2
```

Figura 4.8: Profilo utente

**GET selected participants**

```
http://localhost:8080/events/115/close
```

Figura 4.9: Selezione partecipanti

## 4.7 Diagramma delle interfacce

Il diagramma delle interfacce in figura mostra i metodi offerti al termine della terza iterazione da EventsManagementIF, ProfileManagementIF e ReservationManagementIF. Si può notare come le interfacce siano state modificate per l'implementazione dei casi d'uso, in particolar modo con l'aggiunta dei metodi per la gestione delle prenotazioni.

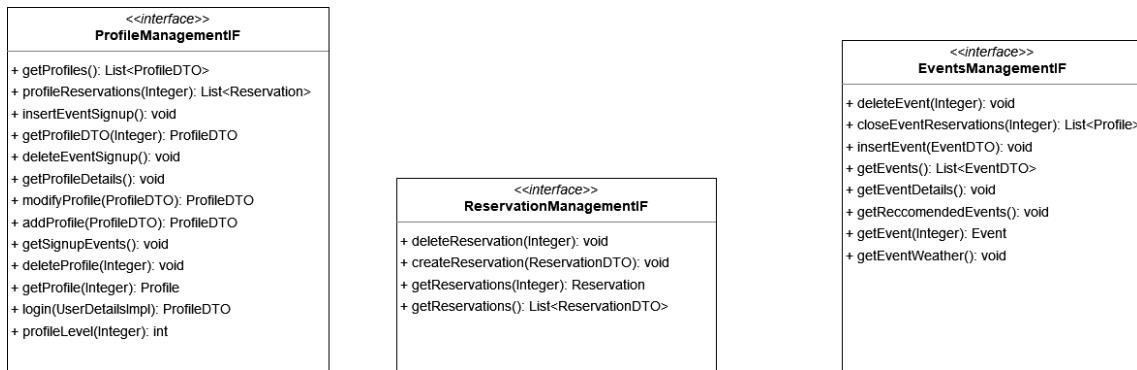


Figura 4.10: Interface Diagram

## 4.8 Implementazione Algoritmo

Nel capitolo precedente si è descritta la fase di sviluppo dell'algoritmo, ed il relativo pseudocodice. Nel corso dell'iterazione 3 l'algoritmo è stato implementato in Spring. In particolare, la sua implementazione è strettamente legata all'introduzione di nuove API quali quella per la selezione dei partecipanti e la gestione del profilo dell'utente. Innanzitutto, per l'algoritmo è stato aggiunto un metodo nell'interfaccia EventsManagementIF responsabile della chiamata all'implementazione effettiva dell'algoritmo, spezzata in due metodi separati (privati) per garantire una maggiore leggibilità. Il nuovo metodo in EventService consente, oltre che di chiamare l'algoritmo, anche di aggiornare se ogni singola prenotazione per quell'evento è confermata o meno, andando ad aggiornare il campo "confirmation". Successivamente, si è introdotto il calcolo del livello del profilo dell'utente, che viene eseguito considerando la media delle difficoltà associate a ciascun evento al quale l'utente ha preso effettivamente parte.

### 4.8.1 Sequence diagram invocazione algoritmo

E' possibile visualizzare le interazioni tra i metodi sviluppati per l'implementazione dell'algoritmo tramite un UML Sequence diagram.

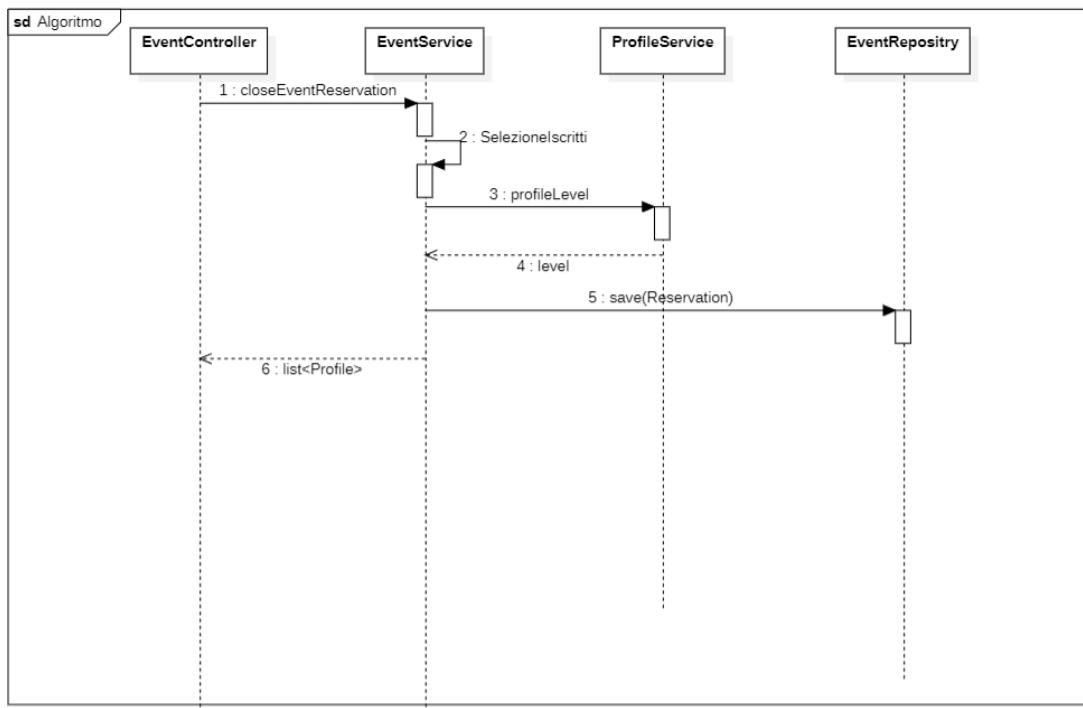


Figura 4.11: Sequence diagram invocazione algoritmo

## 4.9 Casi d'uso

Dettagli dell'implementazione della coda contenente funzionalità di supporto, principalmente legate alla visualizzazione di informazioni ulteriori riguardo agli utenti e/o agli eventi.

#### 4.9.1 UC-4 Visualizza iscrizioni

*Breve Descrizione: il sistema consente all'utente di visualizzare gli eventi a cui è attualmente iscritto.*

*Attori Coinvolti: Sistema, Utente*

*Precondizione: L'utente è registrato e ha effettuato l'accesso.*

*Postcondizione: L'utente visualizza le proprie iscrizioni agli eventi*

*Procedimento:*

1. L'utente accede alla app e visualizza la homepage;
2. L'utente entra nel proprio profilo e seleziona gli eventi in programma;
3. Il sistema mostra una lista contenente gli eventi a cui l'utente è iscritto e a cui non ha ancora partecipato. [E1: Pagina vuota];

*Eccezioni:*

- E1: Pagina vuota
  1. Viene mostrato all'utente un messaggio indicante che al momento non ci sono iscrizioni in corso;

#### 4.9.2 UC-5 Visualizzazione Meteo

Per ogni evento, entro 5 giorni dalla sua data, l'utente può visualizzare una breve del meteo.

*Breve Descrizione:* Usando la posizione e il giorno dell'evento si ottengono le condizioni meteo da mostrare per ogni evento solo prima che manchino 5 giorni allo stesso.

*Attori Coinvolti:* Sistema, Visual Crossing Weather

*Precondizione:* Mancano almeno 5 giorni da un evento

*Postcondizione:* Viene mostrato il meteo per il giorno e il luogo dell'evento

*Procedimento:*

1. Quando l'escursione viene preparata dal gateway si verifica se manchino meno di 5 giorni;
2. In caso affermativo viene invocata l'API di Visual Crossing Weather per ottenere le previsioni [E1: Errore nell'ottenere il meteo];
3. In caso negativo il campo del meteo viene lasciato vuoto.

*Eccezioni:*

- E1: Errore nell'ottenre il meteo
  1. Viene rilevato l'errore attraverso lo statusCode ritornato dall'API;
  2. Il campo del meteo viene lasciato vuoto, come se mancassero più di 5 giorni all'evento.

#### 4.9.3 UC-6 Visualizza eventi consigliati

Da una pagina dedicata sulla app, l'utente può visualizzare una lista di escursioni curata automaticamente in base alla sua esperienza.

*Breve Descrizione:* Visualizzando la pagina “per te” viene presentata una lista di tutti gli eventi disponibili filtrata in base alla loro difficoltà

*Attori Coinvolti:* Sistema, Utente

*Precondizione:* L'utente è loggato nella app

*Postcondizione:* L'utente visualizza gli eventi consigliati

*Procedimento:*

1. L'utente accede alla app e si ritrova sulla homepage;
2. L'utente seleziona la pagina “per te” dal menu in basso nella pagina;
3. Il sistema filtra la lista di eventi scaricata dalla homepage;
4. Il sistema mostra una lista contenente solo gli eventi con una difficoltà $\leq$ all'esperienza dell'utente [E1: Pagina vuota];

*Eccezioni:*

- E1: Pagina vuota

1. Viene mostrato all'utente un messaggio indicante che al momento non ci sono eventi consigliati;

#### 4.9.4 UC-13 Visualizza il profilo di un utente

*Breve Descrizione:* Il sistema consente la visualizzazione dei profili del singolo utente, al fine di verificarne il livello e le escursioni svolte in precedenza.

*Attori Coinvolti:* Utente, sistema

*Precondizione:* L'utente deve essere registrato e deve aver eseguito l'accesso alla piattaforma.

*Postcondizione:* Il sistema mostra una nuova schermata in cui le informazioni generali dell'utente vengono mostrate.

*Procedimento:*

1. L'utente seleziona un profilo;
2. Il sistema mostra il profilo dell'utente selezionato, con annessa tipologia di utente;

*Eccezioni:* Durante il procedimento non si possono verificare eccezioni in quanto solamente i profili accessibili potranno essere visualizzati.

#### 4.9.5 UC-14 visualizza il profilo di un organizzatore

Similmente ad UC-13 (4.9.4) è possibile visualizzare il profilo di un utente.

*Breve Descrizione:* La differenza con UC-13 sta nei dati aggiuntivi che vengono mostrati se l'utente è organizzatore, per esempio il badge di “organizzatore”

*Attori Coinvolti:* Sistema, Utente

*Precondizione:* L'utente deve essere registrato e deve aver eseguito l'accesso alla piattaforma.

*Postcondizione:* Il sistema mostra una nuova schermata in cui le informazioni generali dell'utente vengono mostrate.

*Procedimento:*

1. L'utente seleziona un profilo;
2. Il sistema mostra il profilo dell'utente selezionato, con annessa tipologia di utente;

*Eccezioni:* Durante il procedimento non si possono verificare eccezioni in quanto solamente i profili accessibili potranno essere visualizzati.

#### 4.9.6 UC-6 Visualizza dettagli evento

L'utente può visualizzare informazioni dettagliate riguardo ad un singolo evento.

*Breve Descrizione:* Dalla pagina dell'evento, è possibile consultare dettagli come il luogo, l'ora, la distanza, ecc.

*Attori Coinvolti:* Sistema, utente

*Precondizione:* L'utente è registrato e ha effettuato l'accesso all'app

*Postcondizione:* Vengono mostrati tutti i dettagli dell'evento

*Procedimento:*

1. L'utente accede alla app;
2. L'utente accede alla sezione 'per te' e seleziona un evento di interesse o ne ricerca e seleziona uno dalla barra di ricerca;
3. L'utente visualizza la pagina dell'evento con tutti i dettagli;

*Eccezioni:*

- E1: Problema di connessione

1. Messaggio di errore;
2. Viene chiesto all'utente di riprovare;

## 4.10 Testing

### 4.10.1 Analisi statica

Il seguente report STAN descrive l'analisi statica del codice definitivo relativo al BackEnd.

# Quality Report

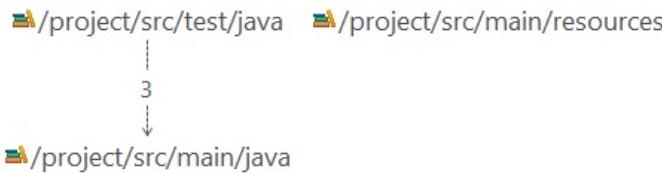
Creation Date 2024-02-04

Package Prefix com.pac.

Level of Detail Class

## Tmp\_240117\_134748

### Library Dependency Graph



### Treemap Overview



### Metrics Summary

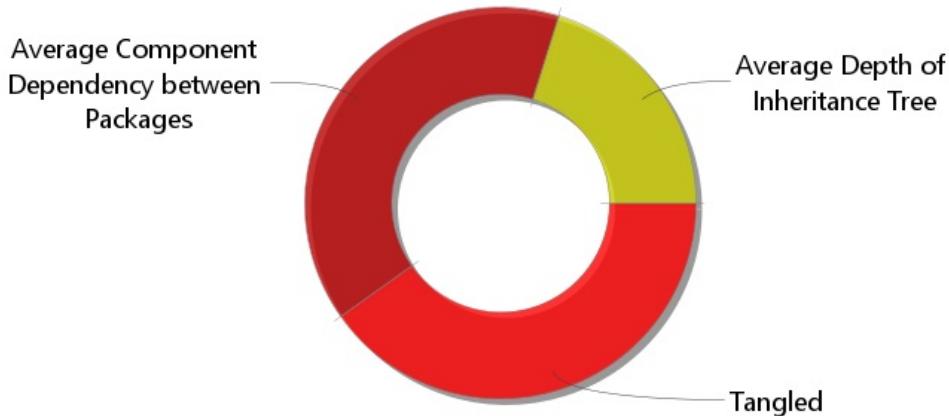
Metric	Value
Number of Libraries	3
Number of Packages	5
Number of Top Level Classes	25
Average Number of Top Level Classes per Package	5
Average Number of Member Classes per Class	0
Average Number of Methods per Class	8.48
Average Number of Fields per Class	3.04
Estimated Lines of Code	852
Estimated Lines of Code per Top Level Class	34.08
Average Cyclomatic Complexity	0.92
Fat for Library Dependencies	1
Fat for Flat Package Dependencies	8
Fat for Top Level Class Dependencies	55
Tangled for Library Dependencies	0%
Average Component Dependency between Libraries	16.67%
Average Component Dependency between Packages	60%
Average Component Dependency between Units	24.83%
Average Distance	-0.17
Average Absolute Distance	0.17
Average Weighted Methods per Class	7.76
Average Depth of Inheritance Tree	0.88
Average Number of Children	0
Average Coupling between Objects	n/a
Average Response for a Class	n/a
Average Lack of Cohesion in Methods	n/a

## Top Violations (3 of 3)

Artifact	Metric	Value
gestoreventi	Tangled	23.08%
Tmp_240117_134748	ACD - Package	60%
Tmp_240117_134748	DIT	0.88

## Pollution Chart

Pollution 1.85



## Violations by Metric

### Tangled

Artifact	Value
gestoreventi	23.08%

### Average Component Dependency between Packages

Artifact	Value
Tmp_240117_134748	60%

### Average Depth of Inheritance Tree

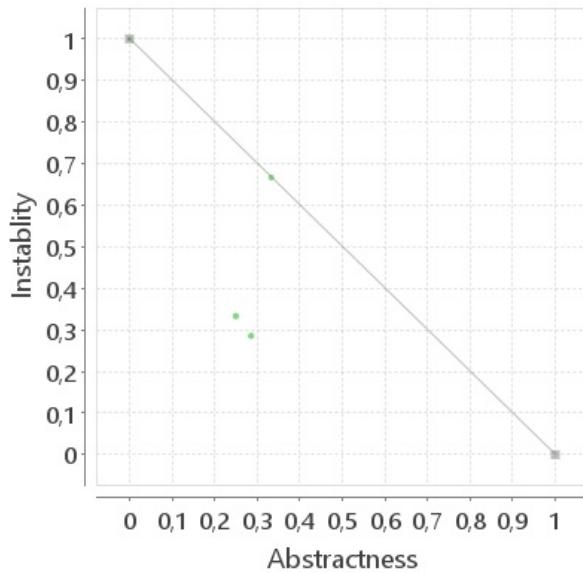
Artifact	Value
Tmp_240117_134748	0.88

## Design Tangles

Tangle inside gestoreventi (#nodes=3, #edges=6, weight=9, fas size=3, weight=3)



## Package Distance Chart



## Metric Ratings

### Count Metrics

Metric	Rating	Linear
Number of Top Level Classes	20 40 60 80	[?]
Number of Methods	25 50 100 200	[?]
Number of Fields	10 20 40 80	[?]
Estimated Lines of Code	200 300 400 500	[?]
Estimated Lines of Code	200 600 1200 2400	[?]

### Complexity Metrics

Metric	Rating	Linear
Cyclomatic Complexity	10 15 20 20	[?]
Fat	20 60 120 240	[?]
Fat	20 60 120 240	[?]
Fat	20 60 120 240	[?]
Tangled	0 1 1	[?]
Tangled for Library Dependencies	0 1 1	[?]
Average Component Dependency between Libraries	0 5 1 1	[?]
Average Component Dependency between Packages	0 5 1 1	[?]

### Robert C. Martin Metrics

Metric	Rating	Linear
Distance	1 5 0 5 1	[?]
Average Absolute Distance	0 4 5 1	[?]

### Chidamber & Kemerer Metrics

Metric	Rating	Linear
Weighted Methods per Class	0 100 200 200	[?]
Depth of Inheritance Tree	1 5 0 10	[?]
Average Depth of Inheritance Tree	2 1 0 0	[?]
Coupling between Objects	0 25 250 250	[?]
Response for a Class	0 100 1000 1000	[?]

#### 4.10.2 Analisi dinamica

Nell’iterazione 3 sono state testate le seguenti API utilizzando Postman:

- Visualizzazione del profilo di un utente;
- Visualizzazione del profilo di un organizzatore;
- Visualizzazione degli eventi consigliati all’utente;
- Visualizzazione dei dettagli di un evento;
- Visualizzazione delle informazioni meteorologiche;
- Visualizzazione degli utenti iscritti ad un evento;

Iteration 1

```

GET get profile
http://localhost:8080/profiles/2
    PASS Status code is 200
                                                200 OK 90 ms 474 B

GET get profiles
http://localhost:8080/profiles
    PASS Status code is 200
                                                200 OK 72 ms 2.25 KB

GET profile level
http://localhost:8080/profiles/level/2
    PASS Status code is 200
                                                200 OK 135 ms 348 B

GET selected participants
http://localhost:8080/events/115/close
    PASS Status code is 200
                                                200 OK 1332 ms 578 B

GET reservations
http://localhost:8080/profiles/2/reservations
    PASS Status code is 200
                                                200 OK 98 ms 896 B

```

Figura 4.12: Test collection

**Iterazione 3 - Run results**

 Ran today at 07:49:43 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests
Runner	none	1	2s 379ms	5

RUN SUMMARY

Test	Status	Count
▶ <b>GET</b> get profile	PASS	1   0
▶ <b>GET</b> get profiles	PASS	1   0
▶ <b>GET</b> profile level	PASS	1   0
▶ <b>GET</b> selected participants	PASS	1   0
▶ <b>GET</b> reservations	PASS	1   0

### 4.10.3 Unit test

Nell'iterazione 3 è stata testata la funzione `findById()` della classe `ReservationRepository`.

```
1  @DataJpaTest
2  class ReservationRepositoryTest {
3
4      @Autowired
5      private ReservationRepository underTest;
6
7      @Test
8      void findById() {
9
10         // given
11         Date date = new Date(100);
12         Reservation expected = new Reservation(1,
13             date, new Boolean(true));
14         underTest.save(expected);
15
16         // when
17         Reservation result =
18             underTest.findById(expected.getId()).get();
19
20         // then
21         assertEquals(expected.getId(),
22             result.getId());
23     }
24 }
```

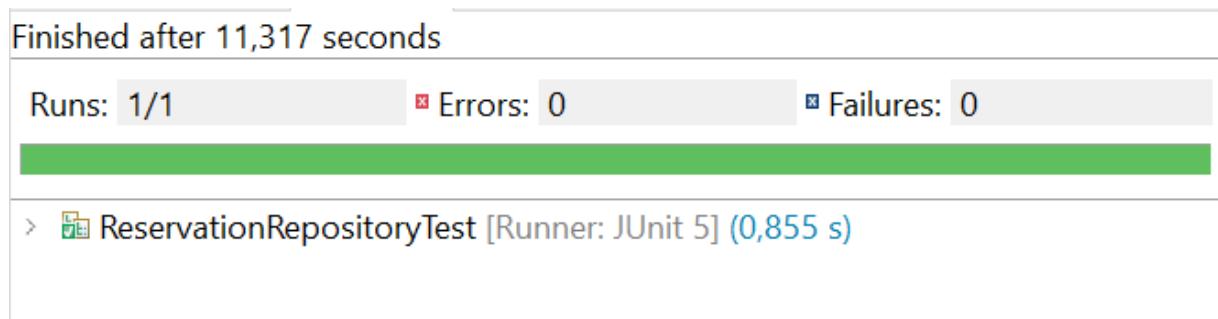


Figura 4.13: findById()

## 4.11 Generazione dei mockup

### 4.11.1 Generazione delle escursioni con ChatGPT

Per fornire una dimostrazione del sistema abbiamo caricato nel database una decina di escursioni di esempio. La generazione di mockup di buona qualità generalmente richiede molto lavoro manuale affinché rispecchino la forma dei dati che nel mondo reale verranno forniti alla piattaforma. Se si usano, per esempio, stringhe troppo corte o troppo lunghe si rischia di creare un prodotto non adatto al mondo reale. Nell'ultimo anno sono emersi numerosi Large Language Model (LLM) che si rivelano molto efficaci nella rielaborazione di testi, specialmente quando gli viene fornito un esempio su cui lavorare. Il nostro problema di generazione dei mockup è appunto un'istanza di questo problema che si presta molto bene ad essere risolta dal un LLM. Abbiamo scelto di lavorare con ChatGPT essendo uno dei modelli più accessibili e prestanti al momento disponibili. Il primo passo è stato elaborare un prompt che istruisse la macchina sul problema che volevamo risolvesse e le fornisse un esempio scritto da noi su cui lavorare (Figura 4.14). Abbiamo quindi ricevuto una risposta con alcuni dei mockup richiesti (Figura 4.18). Avendo verificato che tutti i vincoli venissero rispettati abbiamo chiesto di formulare anche i mockup mancanti per arrivare a 10 (Figura 4.16).

### 4.11.2 Generazione dei profili utente con ChatGPT

Per gli stessi motivi esplicati in 4.11.1abbiamo generato anche 4 profili utente: 2 organizzatori e 2 utenti normali (Figura 4.17).

### 4.11.3 Generazione delle immagini delle escursioni con Crayon V3

Fornendo il prompt “A fictitious mountain range in northern Italy” abbiamo istruito un modello per la generazione di immagini di creare alcune immagini che potessimo creare per arricchire la presentazione dei nostri mockup (Figura 4.19). Con la funzione di upscaling abbiamo infine aumentato la qualità delle immagini che più ci piacevano e le abbiamo scaricate. Tutte le immagini selezionate sono disponibili sulla repository del progetto.

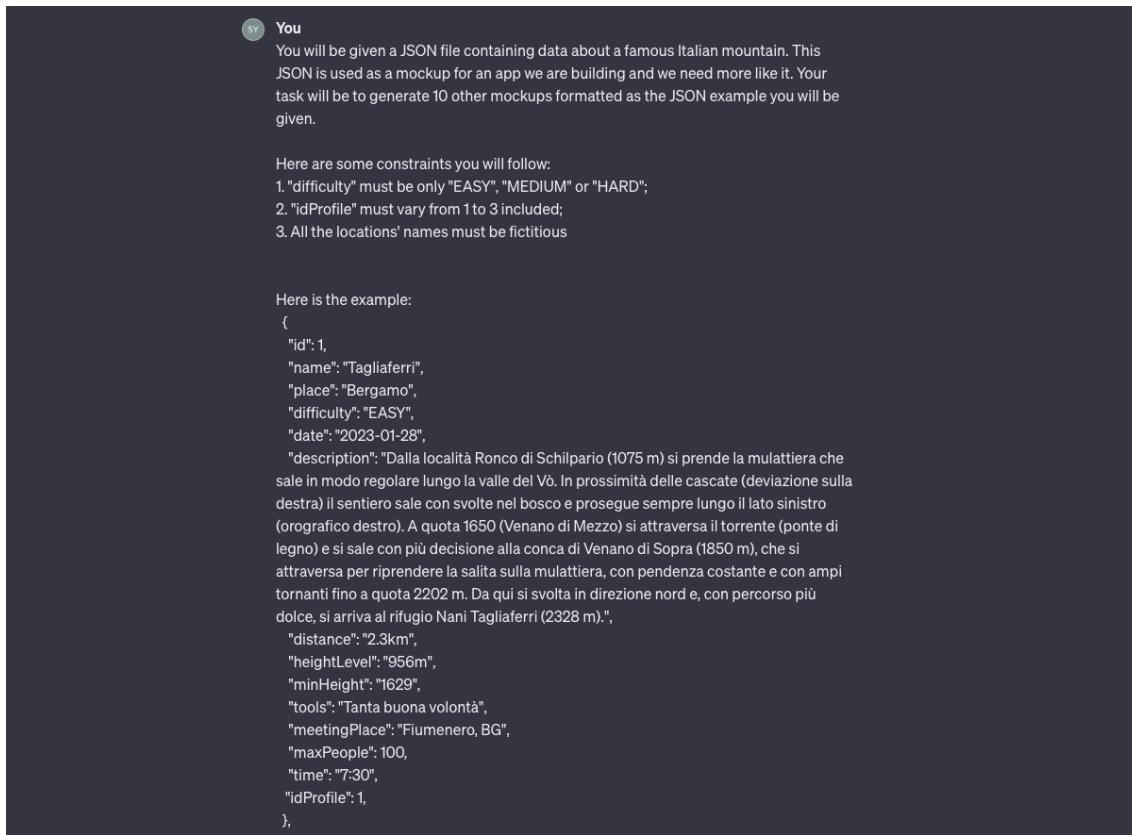


Figura 4.14: Primo prompt per la generazione di escursioni

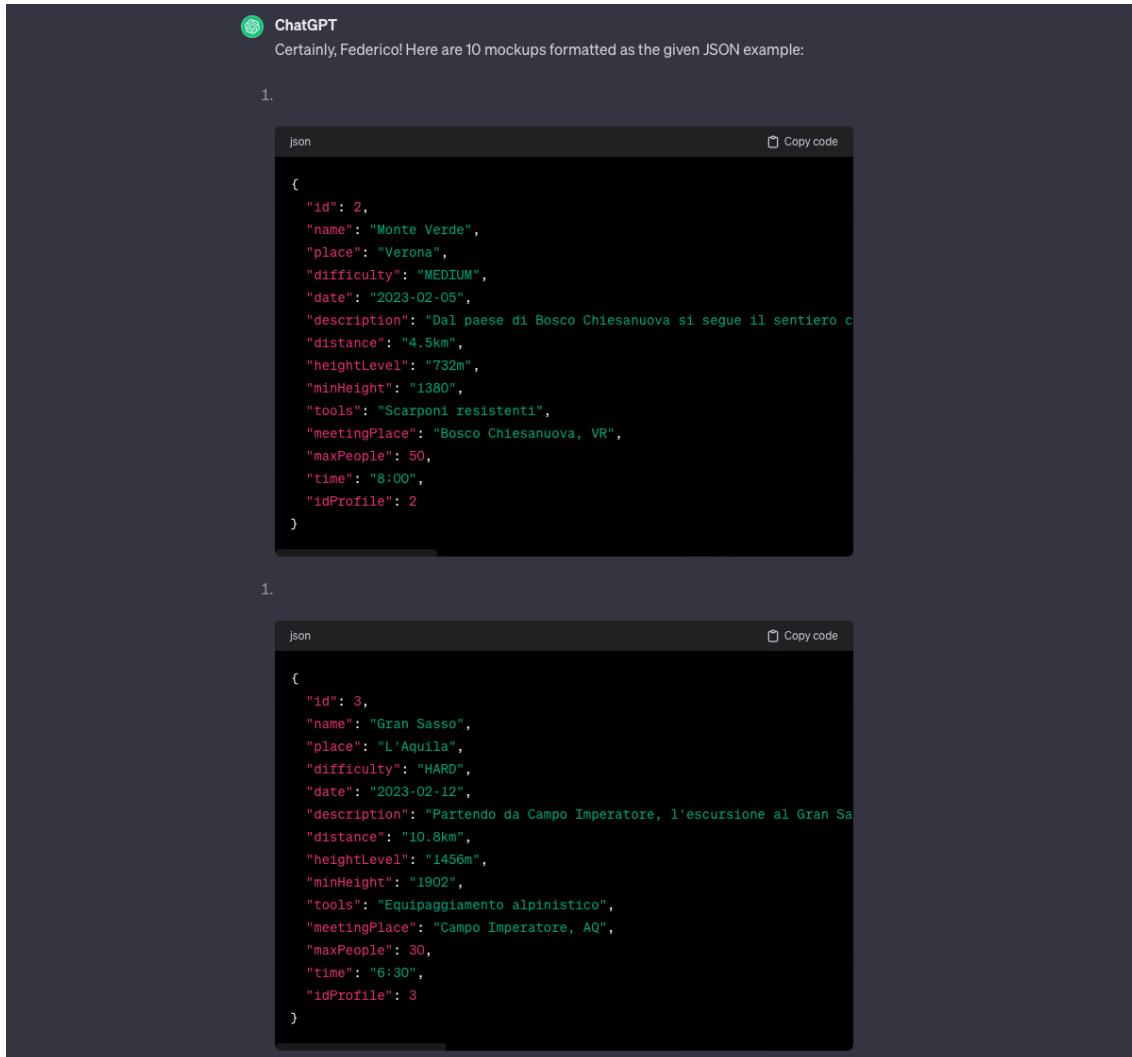


Figura 4.15: Risposta parziale al primo prompt

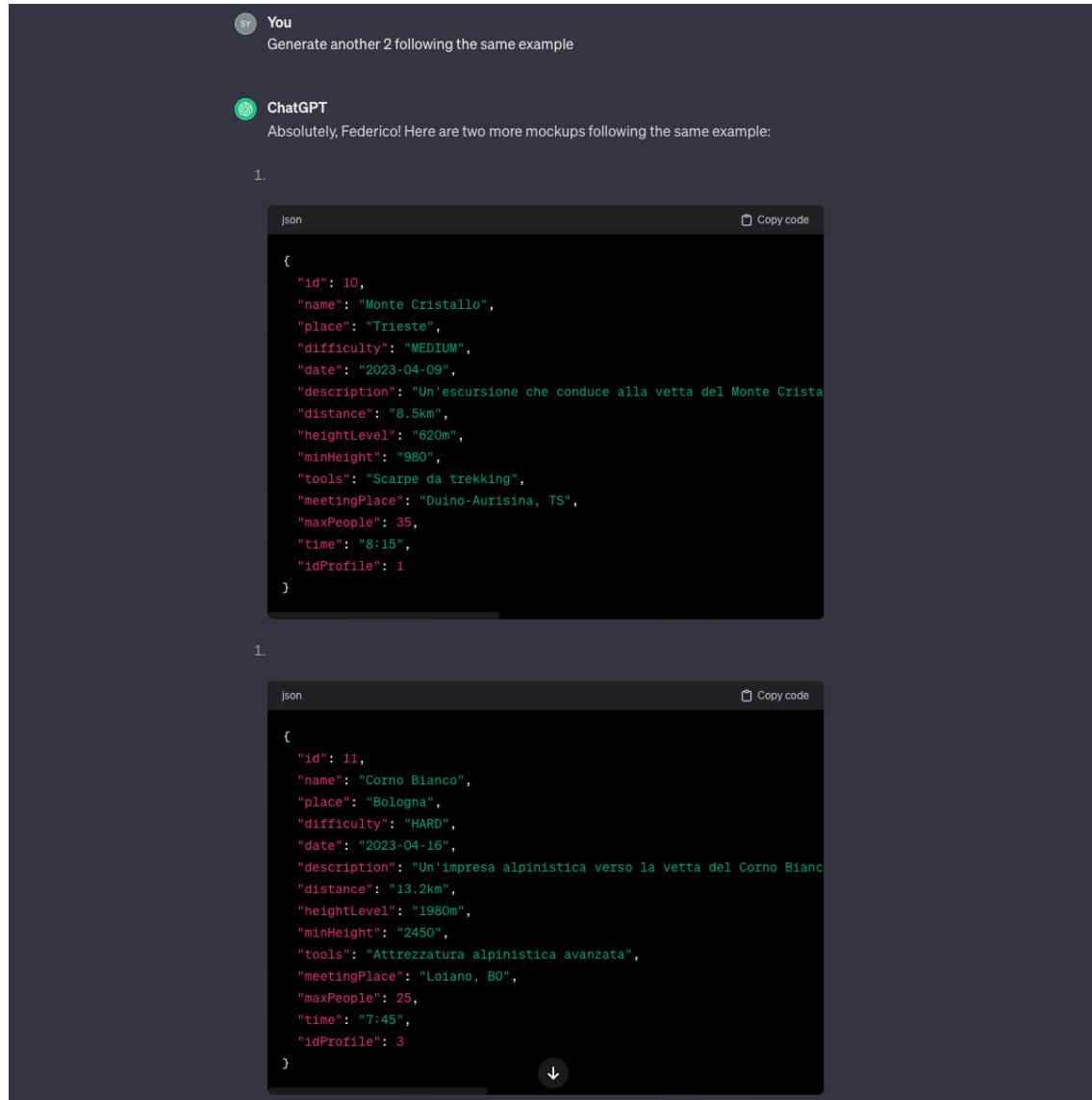


Figura 4.16: Secondo prompt e risposta

SY

**You**

You will be given a JSON file containing data about users to a platform. This JSON is used as a mockup for an app we are building and we need more like it. Your task will be to generate 5 other mockups formatted as the JSON example you will be given.

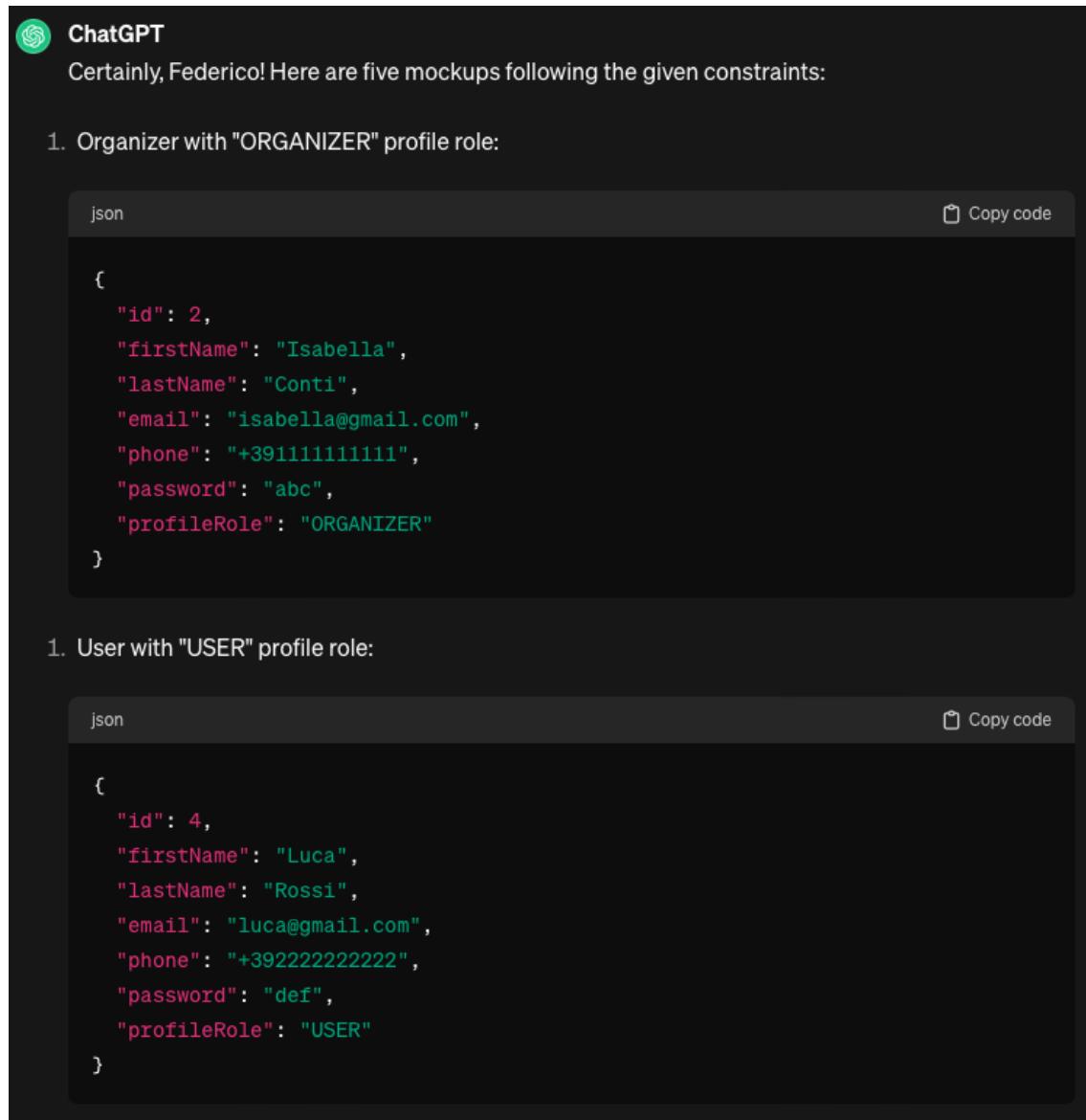
Here are some constraints you will follow:

1. "id" must be within 1 and 5;
2. "email" and "password" must be simple;
3. "profileRole" will only be either "USER" or "ORGANIZER";
4. If the "profileRole" is "ORGANIZER" then its "id" must be either 1 or 2 or 3;
5. If the "profileRole" is "USER" then its "id" must be either 4 or 5;
6. There must only be 2 profiles having "profileRole" as "USER" and only 3 with "profileRole" as "ORGANIZER";
7. "id" must be unique for each object.

Here is the example:

```
{  
  "id": 1,  
  "firstName": "Federico",  
  "lastName": "Imberti",  
  "email": "abc@gmail.com",  
  "phone": "+39000000000000",  
  "password": "123",  
  "profileRole": "ORGANIZER"  
}
```

Figura 4.17: Primo prompt per la generazione di utenti



ChatGPT

Certainly, Federico! Here are five mockups following the given constraints:

1. Organizer with "ORGANIZER" profile role:

```
json
Copy code

{
  "id": 2,
  "firstName": "Isabella",
  "lastName": "Conti",
  "email": "isabella@gmail.com",
  "phone": "+391111111111",
  "password": "abc",
  "profileRole": "ORGANIZER"
}
```

1. User with "USER" profile role:

```
json
Copy code

{
  "id": 4,
  "firstName": "Luca",
  "lastName": "Rossi",
  "email": "luca@gmail.com",
  "phone": "+392222222222",
  "password": "def",
  "profileRole": "USER"
}
```

Figura 4.18: Risposta parziale al primo prompt

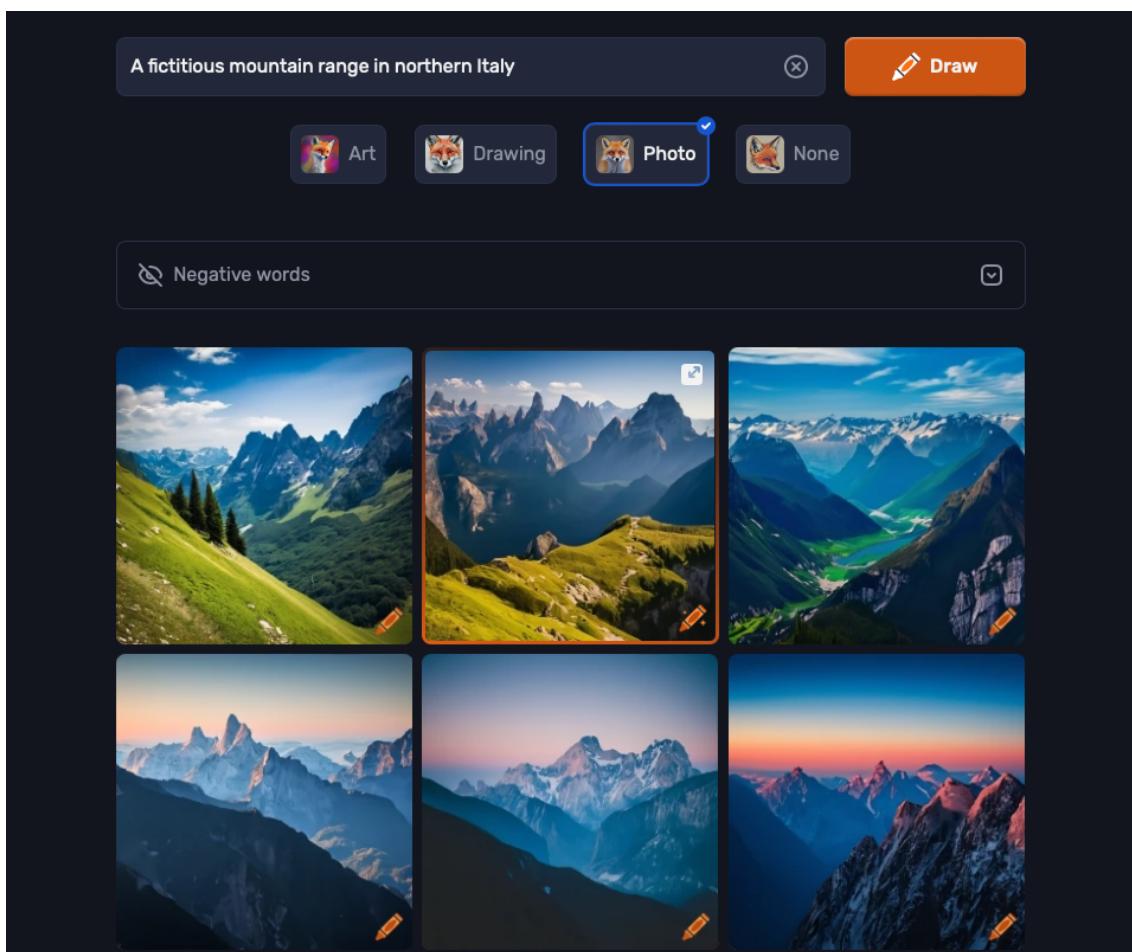


Figura 4.19: Prompt per la generazione di immagini

# Capitolo 5

## Iterazione 4

### 5.1 Introduzione

Durante la quarta e ultima iterazione ci siamo concentrati principalmente su attività di refactoring del codice, sia del backend che del frontend, e della documentazione. Abbiamo inoltre implementato alcuni casi d'uso a bassa priorità. Come ultima attività assembleremo la documentazione finale unendo i precedenti documenti.

## 5.2 Casi d'uso

Dettaglio dell'implementazione dei casi d'uso a bassa priorità che sono stati implementati come parte dell'iterazione finale.

### 5.2.1 UC-4 Visualizzazione delle iscrizioni

L'utente può visualizzare tutti gli eventi a cui è iscritto.

*Breve Descrizione:* Da una pagina dedicata sulla app vengono scaricate tutte le iscrizioni, presenti e passate, a cui ha partecipato l'utente.

*Attori Coinvolti:* Sistema, Utente.

*Precondizione:* L'utente è iscritto alla piattaforma e loggato nella app.

*Postcondizione:* L'utente visualizza gli eventi a cui è iscritto.

*Procedimento:*

1. L'utente visita la pagina "Iscrizioni" dal menu in basso nella app;
2. Il sistema scarica solo quegli eventi a cui l'utente è iscritto;
3. I suddetti eventi vengono mostrati [E1: Nessun evento da mostrare].

*Eccezioni:*

- E1: Nessun evento da mostrare
  1. Viene mostrata una schermata che spiega all'utente che non è stato iscritto a nessun evento.

### 5.2.2 UC-23 Visualizzare profilo dell'organizzatore di un evento

L'utente può visualizzare il profilo di chi sta organizzando un evento.

*Breve Descrizione: Cliccando sulla foto profilo dell'organizzatore di un evento si visualizzano i dettagli del suo profilo.*

*Attori Coinvolti: Sistema, Utente.*

*Precondizione: L'utente è iscritto alla piattaforma, loggato nella app e visualizza una lista di eventi*

*Postcondizione: L'utente visualizza il profilo del creatore di un dato evento.*

*Procedimento:*

1. L'utente clicca l'immagine dell'organizzatore in alto a sinistra di una finestra che descrive un evento
2. Il sistema scarica i dati relativi all'utente [E1: Utente non trovato]
3. La pagina con le informazioni dell'utente organizzatore viene mostrata

*Eccezioni:*

- E1: Utente non trovato
  1. Viene mostrata una schermata di errore che invita l'utente a riprovare.

## 5.3 Docker-compose, Continuous Integration e Continuous Deployment

Durante l'iterazione 4 abbiamo espanso e migliorato l'uso delle Github Actions per implementare l'approccio CI/CD che è stato delineato in 4.2. La nostra architettura è stata inoltre completata dall'aggiunta di 2 immagini. Per coordinarle sono state assemblate nel file ‘docker-compose.yml’ reperibile alla root della repository. Così facendo è possibile lanciare l'intera architettura con un semplice ‘docker-compose up’.

### 5.3.1 Immagini Docker e struttura finale del docker-compose

Rispetto a quanto descritto in 4.2 abbiamo espanso la nostra containerizzazione a 3 immagini compilate dal nostro codice:

- “ventura-boulevard”, contenente il manager degli utenti e degli eventi;
- “radio-nowhere”, contenente l’API Gateway usato per comunicare con la mobile app;
- “zombie-zoo”, contenente il frontend della mobile app.

Con altre 3 disponibili liberamente:

- “watchtower”, usata per la implementare Continuous Deployment (maggiori dettagli a 5.3.3);
- “postgres”, l’immagine ufficiale di PostgreSQL, usata per creare e gestire il database. Abbiamo fatto in modo che lo schema venisse inizializzato quando l’immagine viene costruita in un container per la prima volta, inserendo inoltre l’utente admim. Per fare ciò abbiamo scritto uno script in SQL e impostato PostgreSQL affinché lo lanciasse al primo avvio. Le configurazioni di username, password e nome del database sono scritte in docker-compose.yml.
- “pgadmin4”, l’immagine ufficiale di “pgAdmin”, usata per interagire con il database attraverso una UI. Le configurazioni di username, password e nome del database sono scritte in docker-compose.yml.

### 5.3.2 Continuous Integration (CI)

Abbiamo riscritto le Actions dividendo la generazione di ognuna delle immagini e mantenendo la suddivisione "deployment" e "development" per distinguere le immagini pubblicate sull'ambiente di produzione da quelle sull'ambiente di sviluppo. In questo modo abbiamo attenuto 6 Actions che vengono eseguite in base a quale cartella della repository subisce una push. Di seguito riassumiamo ogni Actions, con la relativa descrizione e trigger.

#### Github Actions

**Dep - Frontend** Costruisce e pubblica l'immagine del frontend dell'ambiente di produzione ad ogni Push sulla branch "production" all'interno della cartella "ClientApp/Iterazione\_4\_finale/mountain\_app";

**Dep - Manager** Costruisce e pubblica l'immagine dei manager dell'ambiente di produzione ad ogni Push sulla branch "production" all'interno della cartella "Backend/Iterazione3/project";

**Dep - Gateway** Costruisce e pubblica l'immagine del gateway dell'ambiente di produzione ad ogni Push sulla branch "production" all'interno della cartella "Backend/Iterazione3/gateway";

**Dev - Frontend** Costruisce e pubblica l'immagine del frontend dell'ambiente di sviluppo ad ogni Push sulla branch "main" all'interno della cartella "ClientApp/Iterazione\_4\_finale/mountain\_app";

**Dev - Manager** Costruisce e pubblica l'immagine dei manager dell'ambiente di sviluppo ad ogni Push sulla branch "main" all'interno della cartella "Backend/Iterazione3/project";

**Dev - Gateway** Costruisce e pubblica l'immagine del gateway dell'ambiente di sviluppo ad ogni Push sulla branch "main" all'interno della cartella "Backend/Iterazione3/gateway".

### 5.3.3 Continuous Deployment (CD)

Già in 4.2.2 avevamo introdotto Watchtower e il meccanismo usato per verificare se un’immagine sulla quale viene costruito un container sia da aggiornare. Nell’iterazione 4 abbiamo migliorato l’uso che ne abbiamo fatto modificandone alcune impostazioni e includendolo nel docker-compose di tutta l’architettura. Abbiamo inoltre aggiunto un secondo container denominato “manual-update”. Essenzialmente si tratta di un’istanza di Watchtower programmata per eseguire un solo update delle immagini nel momento in cui viene lanciata per poi spegnersi. Come suggerisce il nome questo container aggiuntivo risulta molto comodo per anticipare un update rispetto all’intervallo di 24 ore impostato sull’updater automatico. È bene notare che all’esecuzione di un aggiornamento manuale il timer dell’aggiornamento automatico non viene resettato. Per fare in modo che le due istanze della stessa immagine non entrassero in conflitto abbiamo programmato, nel docker-compose, l’updater manuale per non avviarsi in corrispondenza di ‘docker compose up’ ma di aspettare che venga invocato manualmente con il comando ‘docker start manual-update’.

## 5.4 Guida per l'installazione

Il progetto che consegniamo può essere eseguito sia in locale che in remoto. L'istanza remota verrà disattivata una volta terminata la correzione e presentazione del progetto mentre l'esecuzione locale resterà disponibile indefinitamente.

### 5.4.1 Esecuzione locale

Per eseguire il progetto sulla propria macchina si dovrà procedere in 6 passaggi:

1. Verificare di aver installato Docker, per esempio verificando che il comando ‘docker –version’ non restituisca errori;
2. Lanciare Docker tramite comando oppure tramite Docker Desktop;
3. Scaricare il file docker-compose.yml alla root della repository, posizionarsi nella stessa cartella dove è stato scaricato e assicurarsi che sia l'unico file chiamato ‘docker-compose.yml’;
4. Eseguire il file con il comando ‘docker compose up’ da una finestra del terminale aperta nella cartella dove è stato scaricato il file;
5. Attendere l'installazione di tutti i servizi (potrebbe richiedere un po' di tempo dipendentemente dalla velocità della propria rete internet);
6. È ora possibile utilizzare il servizio:
  - Collegarsi a <http://localhost:8080> se si vuole interagire direttamente con i manager (per esempio per debugging);
  - Collegarsi a <http://localhost:8085> se si vuole interagire direttamente con il gateway (per esempio per debugging);
  - Collegarsi a <http://localhost:9000> se si vuole interagire con il frontend della app mobile. Essendo l'app pensata per essere eseguita unicamente su dispositivi mobili consigliamo di **restringere in larghezza la pagina del browser** in modo da simulare le dimensioni di uno smartphone.
  -

- Collegarsi a `http://localhost:5050` se si vuole interagire in maniera grafica con il database attraverso pgAdmin (per esempio per debugging);

Alla sezione QuickStart del `readme.md` della repository è possibile trovare un comando che velocizza l'esecuzione in locale dell'architettura scaricando il `docker-compose.yml` attraverso curl ed eseguendolo subito dopo.

#### 5.4.2 Esecuzione remota

Per facilitare la correzione del progetto abbiamo pensato di pubblicare su di una macchina virtuale (messa a disposizione da Digital Ocean con un credito di 200\$ dedicato agli studenti) il nostro progetto in modo che ci si possa collegare facilmente senza il bisogno di installare Docker o scaricare nulla. Per accedere all'istanza remota è possibile collegarsi a:

- `http://165.227.152.216:8080` se si vuole interagire direttamente con i manager (per esempio per debugging);
- `http://165.227.152.216:8085` se si vuole interagire direttamente con il gateway (per esempio per debugging);
- `http://165.227.152.216:9000` se si vuole interagire con il frontend della app mobile. Essendo l'app pensata per essere eseguita unicamente su dispositivi mobili consigliamo di **restringere in larghezza la pagina del browser** in modo da simulare le dimensioni di uno smartphone.
- 
- Collegarsi a `http://165.227.152.216:5050` se si vuole interagire in maniera grafica con il database attraverso pgAdmin (per esempio per debugging);

Abbiamo deciso di lasciare aperte all'esterno sia la porta 8080 che la porta 8085 per soli scopi didattici, sottolineiamo che se si non si trattasse di un progetto dimostrativo avremmo lasciato solo la porta 9000 per il frontend e 5050 per pgAdmin.

## 5.5 Manuale utente

L'utente può interracciaiarsi con l'applicazione tramite diverse schermate. Innanzitutto, è possibile registrarsi creando un nuovo account o eventualmente eseguire direttamente il login. Per fare ciò sono state predisposte le due interfacce 5.1 e 5.2. Nel database reso a disposizione per la presentazione del progetto sono già inseriti 5 utenti (quelli generati dal mockup e l'admin generato automaticamente). Per esempio è possibile accedere come utente normale usando:

- username: luca@gmail.com
- password: def

E come organizzatore usando:

- username: martina@gmail.com
- password: ghi

Eseguito l'accesso tramite le proprie credenziali utente, si viene reindirizzati alla schermata principale che mostra gli eventi disponibili 5.5. In alto a sinistra è possibile accedere al proprio profilo 5.3, mentre tramite l'apposita icona si apre la sezione cerca 5.8, da cui si possono cercare più rapidamente specifici eventi. La creazione di una nuova escursione è possibile solo se loggati come organizzatore e avviene facendo click sul bottone "nuova escursione", in quanto si verrà rimandati alla schermata da cui inserire le informazioni del nuovo evento 5.6 (qualora l'utente sia un organizzatore). Inoltre, tramite la barra di navigazione è possibile accedere alla sezione dedicata ai consigliati 5.7 ed alle iscrizioni 5.4, in cui vengono mostrati gli eventi ai quali si è iscritti. Infine, da qualsiasi schermata è possibile accedere al profilo di ogni evento, nel quale vengono mostrate tutte le specifiche inserite dall'utente organizzatore.

**Login**  
Accedi per iniziare

Indirizzo Email

Password

**Accedi**

Ottieni [Registrati](#)

Figura 5.1: Login screen

**Crea Account**  
Registrati per iniziare

Nome\*

Cognome\*

Indirizzo Email\*

Numero di telefono\*

Codice Organizzatore

Password\*

**Registrati**

Figura 5.2: Signup screen

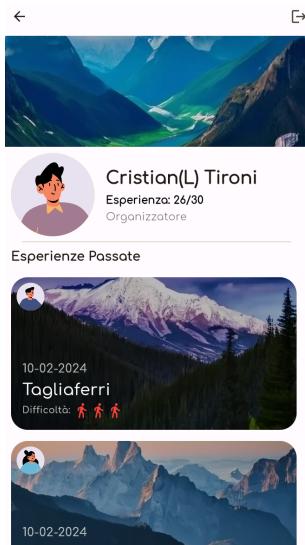


Figura 5.3: Profilo utente



Figura 5.4: Iscrizioni



Figura 5.5: Homescreen

**Crea Evento**  
Compila i campi per creare un nuovo evento

**Nome\***

**Luogo\***

**Data dell'escursione\***

**Descrizione\***

**Difficoltà:** medium

Figura 5.6: New evento

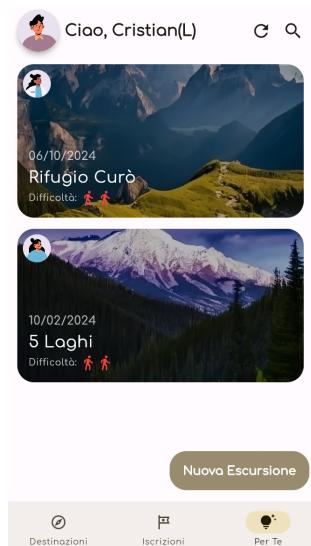


Figura 5.7: Suggerimenti

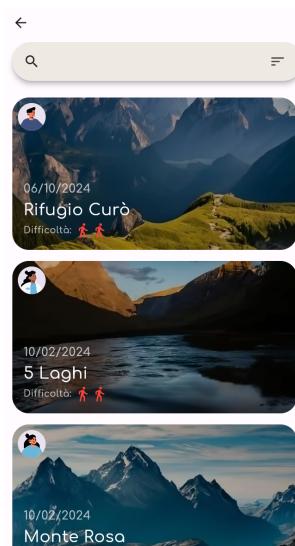


Figura 5.8: Cerca

## 5.6 Sviluppi Futuri

### 5.6.1 Status dei casi d'uso

Non tutti i casi d'uso definiti sono stati implementati. Di seguito è riportata una tabella riassuntiva dello status di ognuno.

Codice	Caso d'uso	Implementato
Alta Priorità		
<b>UC1</b>	Login	Sì
<b>UC2</b>	Signup	Sì
<b>UC3</b>	Logout	Sì
<b>UC7</b>	Ricerca eventi	Sì
<b>UC8</b>	Visualizzazione risultati ricerca	Sì
<b>UC9</b>	Visualizzazione specifico risultato	Sì
<b>UC10</b>	Iscrizione ad un nuovo evento	Sì
<b>UC11</b>	Cancellazione iscrizione	Sì
<b>UC15</b>	Algoritmo iscrizione	Sì
<b>UC17</b>	Eliminazione evento	Sì
<b>UC18</b>	Creazione evento	Sì
Media Priorità		
<b>UC5</b>	Visualizzazione meteo	Sì
<b>UC6</b>	Visualizzazione consigliati	Sì
<b>UC13</b>	Visualizzazione profilo utente	Sì
<b>UC14</b>	Visualizzazione profilo organizzatore	Sì
<b>UC16</b>	Dettagli evento	Sì
Bassa Priorità		
<b>UC4</b>	Visualizzazione iscrizioni	Sì
<b>UC12</b>	Visualizzazione partecipanti	No
<b>UC19</b>	Visualizzare profilo partecipanti	No
<b>UC20</b>	Cambiare la foto del profilo	No
<b>UC21</b>	Cambiare la foto di copertina	No
<b>UC22</b>	Cambiare la foto di un evento	No

Tabella 5.1: Casi d'uso implementati e mancanti.

### 5.6.2 Punti da trattare

Se si volesse procedere oltre nel progetto, i seguenti punti andrebbero certamente trattati:

- Implementare i restanti casi d'uso a bassa priorità;
- Registrare un dominio e associarlo alla macchina virtuale che pubblica il backend;
- Acquistare un certificato SSL per il suddetto dominio e implementare un reverse proxy sul server in modo da poter usare HTTPS;
- Usare un sistema di login più sicuro: al momento username e password sono codificati in Base64 vengono inviati per ogni richiesta come parte del corpo della richiesta;
- Rendere automatica l'esecuzione dell'algoritmo di selezione dei partecipanti;
- Rafforzare la sicurezza del database e rendere il trattamento dei dati conforme al GDPR.