Contents

List of Figures

0.1 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.

0.1.1 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 mentenere 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 richieta, 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".

0.1.2 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 realtivo 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 obbligatoru 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.

0.1.3 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''e nessuna sessione tra client e server, di conseguenza questo caso d'uso viene gestito intermamente 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 piú loggato nel sistema

Procedimento:

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

0.1.4 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;

0.1.5 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 scrorrere 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;

0.1.6 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;

0.1.7 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 speficiche dell'evento

Postcondizione: L'utente pronota 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".

0.1.8 UC-11 Cancellazione da un evento

L'utente può eliminare la propria partecipazione ad 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 l'icona "disiscrivimi";

Eccezioni:

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

0.1.9 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

Precondizione: Raggiungimento della data massima per iscriversi a un evento.

Postcondizione: Gli utenti non selezionati vengono eliminati dai partecipanti dell'evento

Procedimento:

1. Lanciare un trigger ogni giorno fino al raggiungimento della data ultima per l'iscrizione;

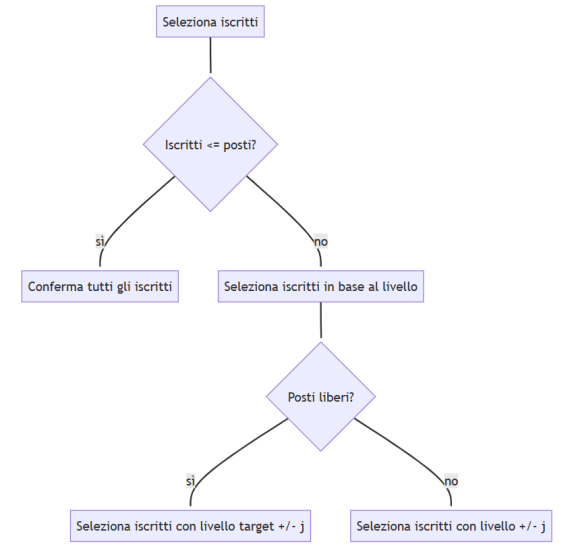
2. L'algoritmo viene applicato sulla lista degli iscritti all'evento.

FlowChart 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.

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 <= 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




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

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

- Caso Peggiore (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 peggiore 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(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(limiteMax).

0.1.10 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 voule 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".

0.1.11 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 schemata principale;
2. Completare i campi che vengono visualizzati;
3. Selezionare l'icona submit;

Eccezioni:

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

0.2 Component Diagram

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

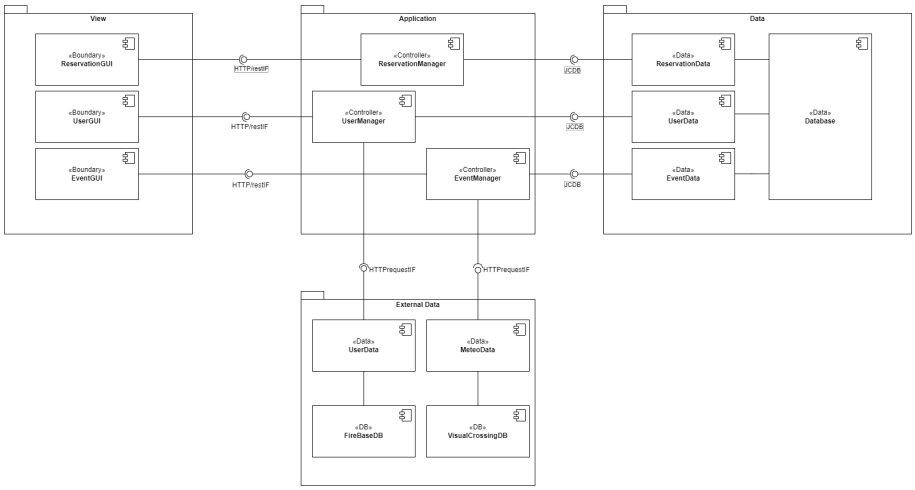


Figure 1: Component Diagram.

0.3 DataClass Diagram

Il Class Diagram relativo all'iterazione 2 è stato generato automaticamente a partire dal codice del Backend, tramite il tool integrato in INTELLIJ IDEA.

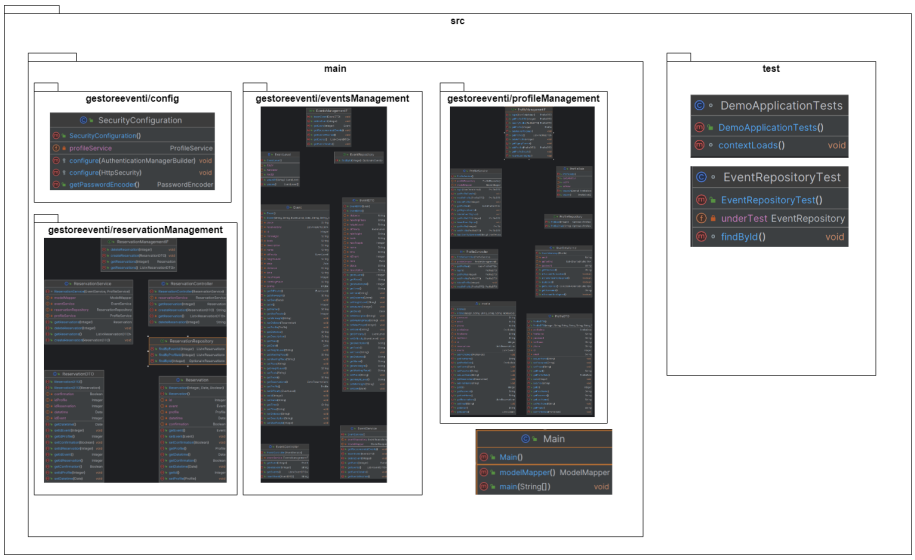


Figure 2: Class Diagram.

0.4 Interface Diagram

L'interfaccia 'Reservation Management IF' dichiara i metodi messi a disposizione per la gestione delle prenotazioni 3

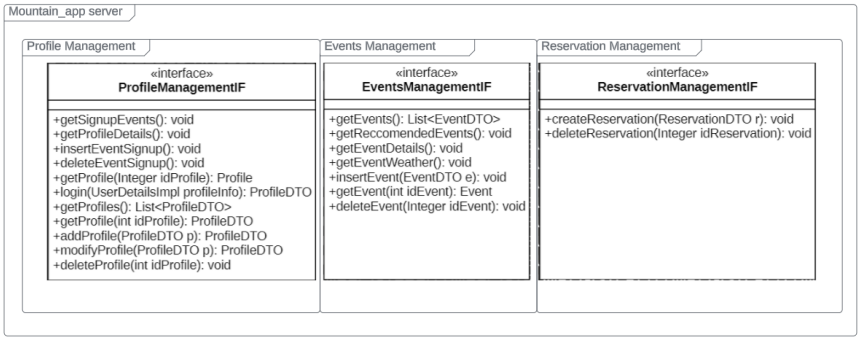


Figure 3: Interface Diagram.

0.5 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 APIs sviluppate, le piu significative offrono le seguenti funzionalità.

- API per la creazione di un evento.
- API per la creazione di una prenotazione.
- 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"
}
```


Figure 4: Creazione 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
}
```


Figure 5: Inserimento nuova prenotazione

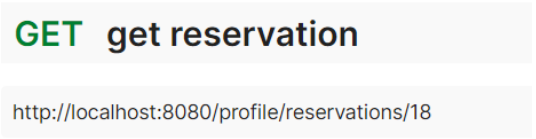


Figure 6: Prenotazione dato uno specifico profilo

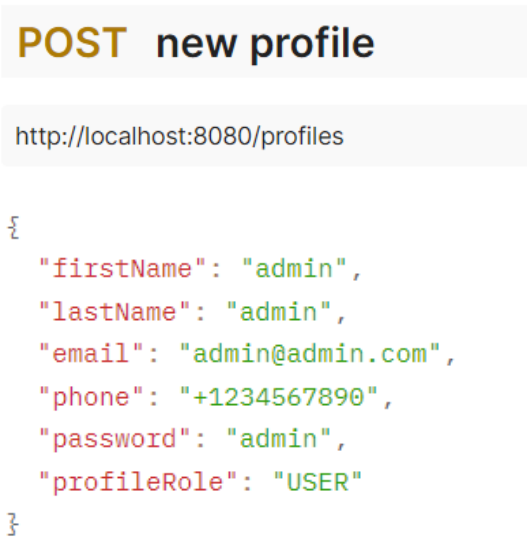


Figure 7: Creazione nuovo profilo

0.6 Generazione degli eventi mockup 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 adeguato alla situazione che "spiegasse alla macchina" il problema che volecamo risolvesse e le fornisse un esempio scritto da noi su cui lavorare (Figura 8 e Figura 9). Successivamente abbiamo affinato il prompt fornendo dettagli più precisi sulla forma di "difficulty" (Figura 10). Infine abbiamo chiesto che fossero generati i restanti mockup per arrivare a 10 (Figura 11)

SY **You**

You will be given a JSON file containing data about a famous Italian mountain. 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 10 other mockups formatted as the JSON example you will be given.

Here is the example:


```
{
 "id": "123",
 "name": "Tagliaferri",
 "place": "Bergamo",
 "difficulty": "EASY",
 "date": "20-09-2023",
 "description": "Dalla località Ronco di Schilpario (1075 m) si prende la mulattiera che sale in modo regolare lungo la valle del Vò. In prossimità delle cascate (deviazione sulla destra) il sentiero sale con svolte nel bosco e prosegue sempre lungo il lato sinistro (orografico destro). A quota 1650 (Venano di Mezzo) si attraversa il torrente (ponte di legno) e si sale con più decisione alla conca di Venano di Sopra (1850 m), che si attraversa per riprendere la salita sulla mulattiera, con pendenza costante e con ampi tornanti fino a quota 2202 m. Da qui si svolta in direzione nord e, con percorso più dolce, si arriva al rifugio Nani Tagliaferri (2328 m).",
 "distance": "2.3km",
 "heightLevel": "956m",
 "minHeight": "1629",
 "tools": "Tanta buona volontà",
 "meetingPlace": "Fiumenero, BG",
 "maxPeople": 100,
 "time": "7:30"
}
```


Figure 8: Primo prompt

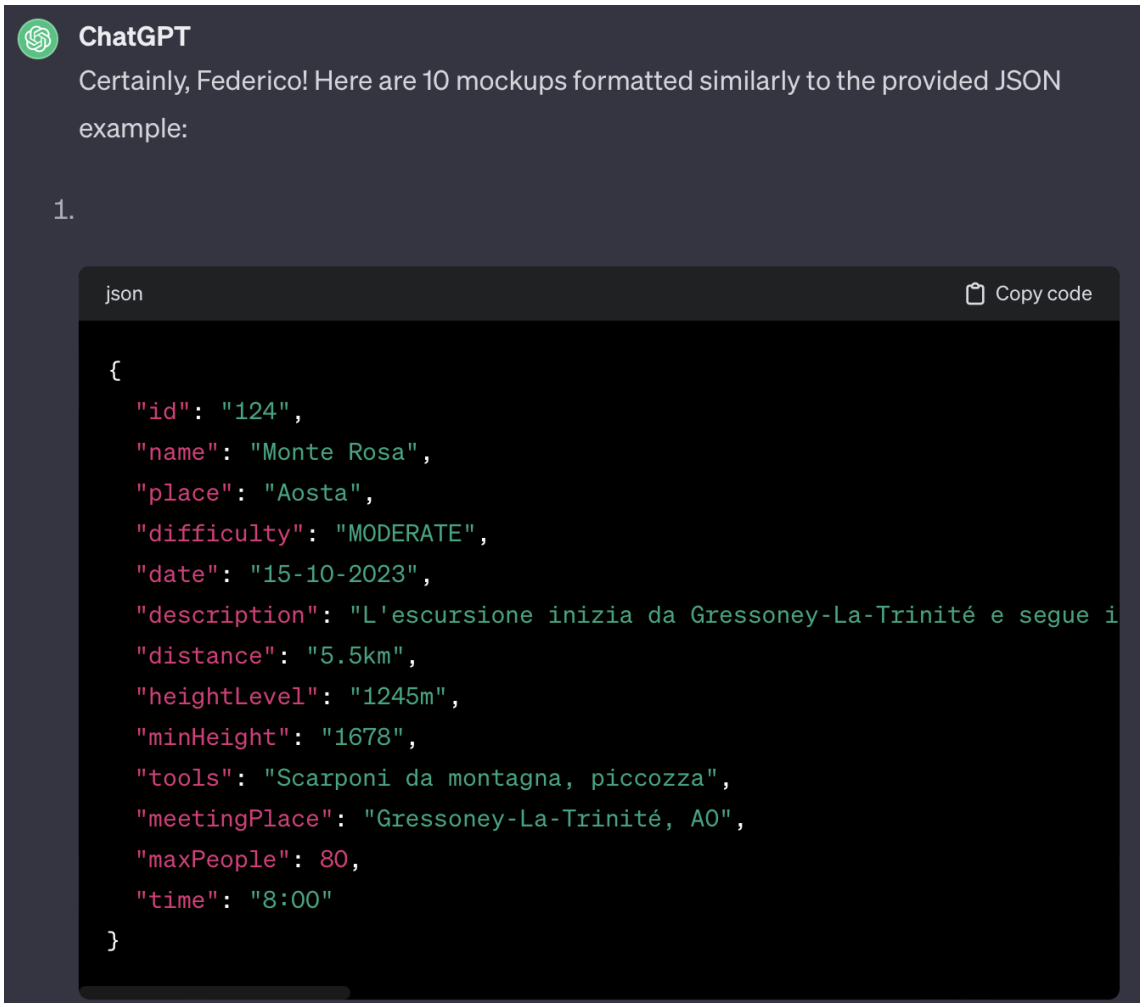


Figure 9: Risposta parziale al primo prompt

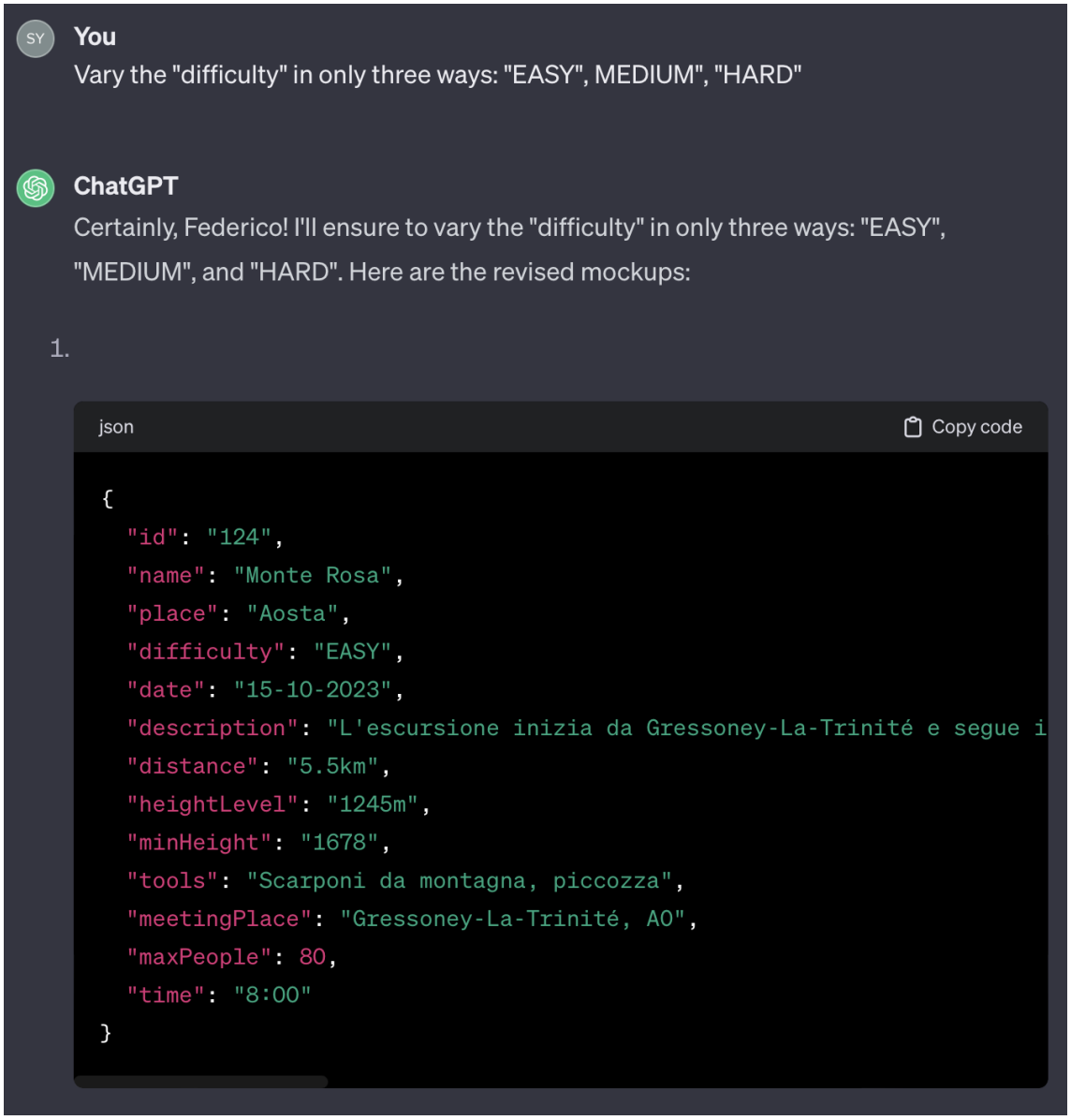


Figure 10: Secondo prompt e risposta parziale

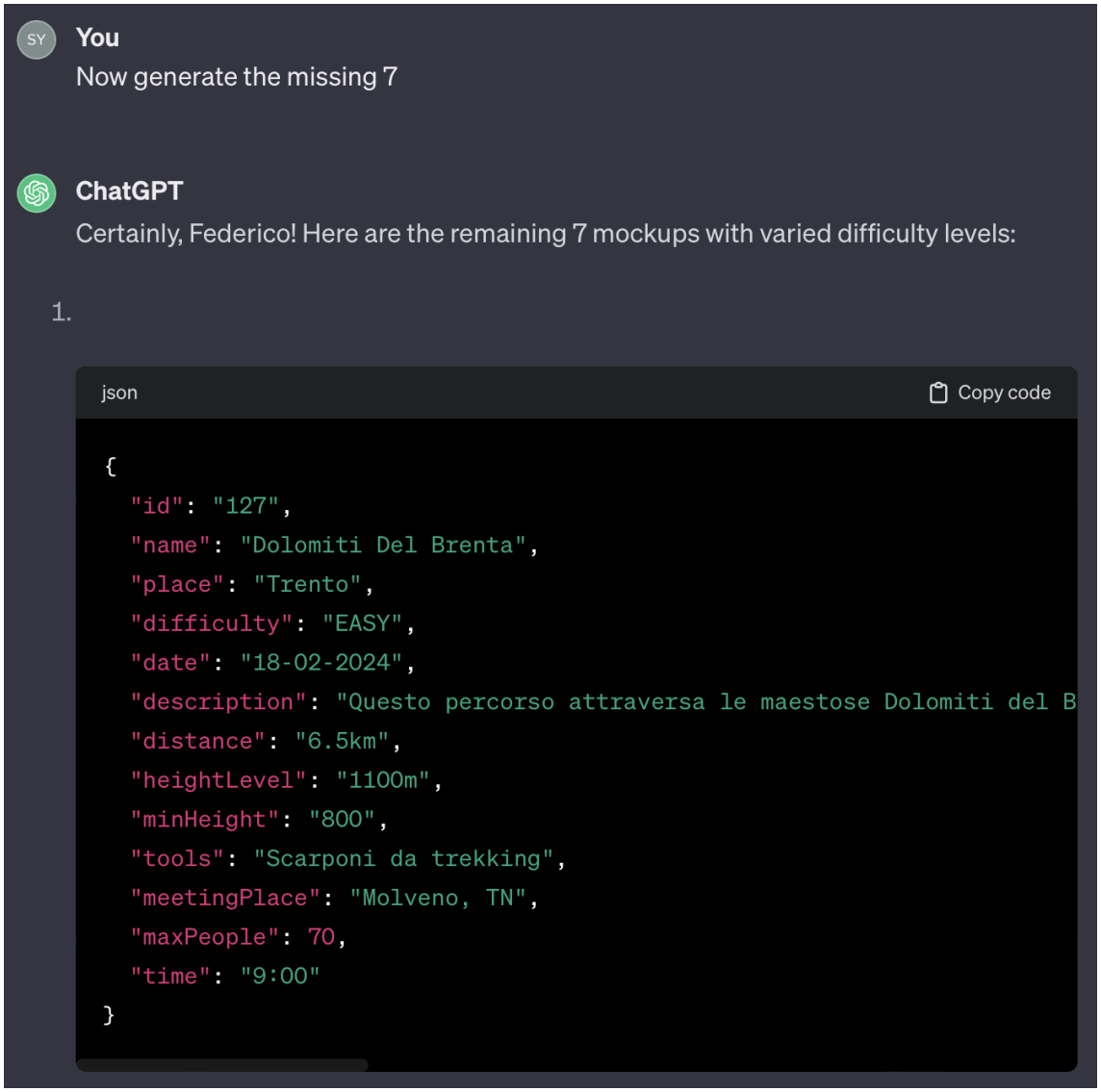


Figure 11: Terzo prompt e riposta parziale

0.7 Testing

0.7.1 Analisi statica

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

Quality Report

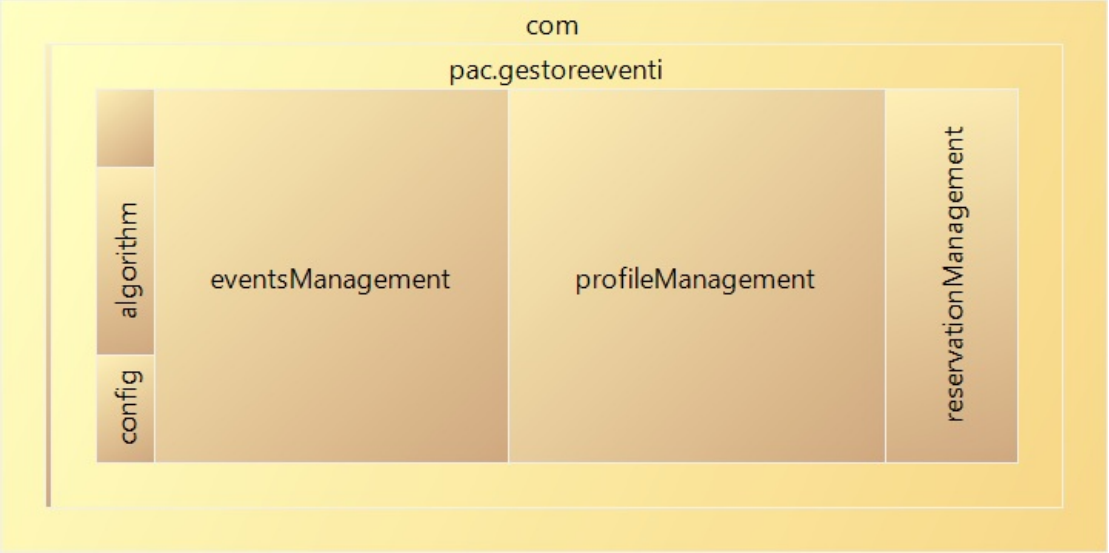
Creation Date 2024-01-17
Level of Detail Member

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

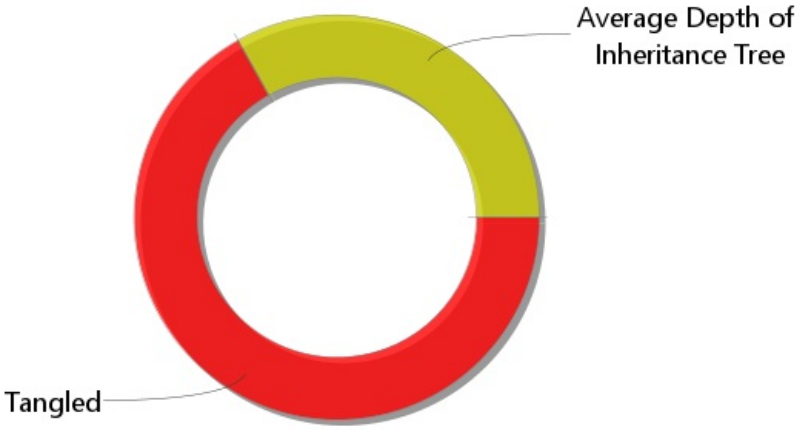
Top Violations (2 of 2)

Artifact	Metric	Value

com.pac.gestoreeventi	Tangled	26.42%
Tmp_240117_134748	DIT	0.88

Pollution Chart

Pollution 1.12



Violations by Metric

Tangled

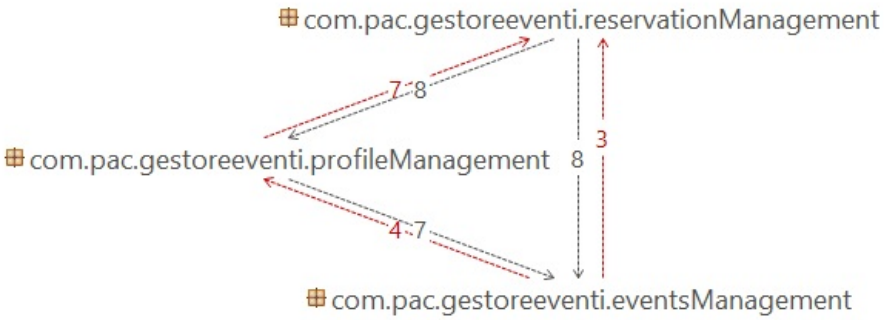
Artifact	Value
com.pac.gestoreeventi	26.42%

Average Depth of Inheritance Tree

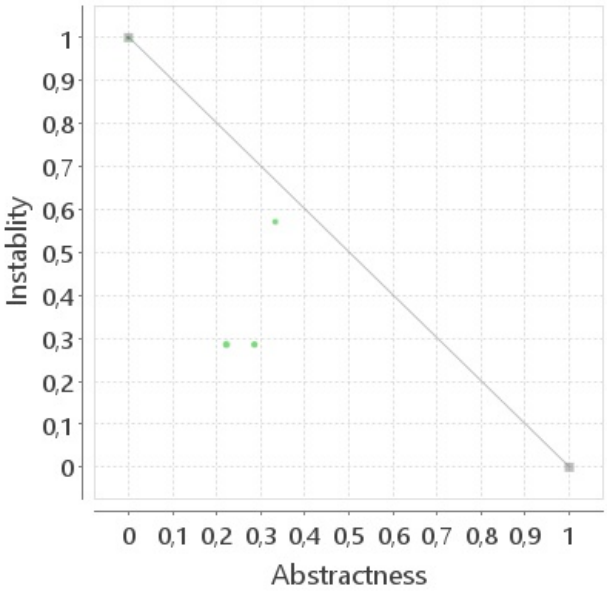
Artifact	Value
Tmp_240117_134748	0.88

Design Tangles

Tangle inside com.pac.gestoreeventi *(#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	200 300 400 500	✓
Estimated Lines of Code	30 60 120 240	✓

Complexity Metrics

Metric	Rating	Linear
Cyclomatic Complexity	10 15 20 30	✓
Fat	30 60 120 240	✓
Fat	30 60 120 240	✓
Fat	30 60 120 240	✓
Tangled	0 1	✓
Tangled for Library Dependencies	0 1	✓
Average Component Dependency between Libraries	0 5 1	✓
Average Component Dependency between Packages	0 5 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	0 100 200 300	✓
Depth of Inheritance Tree	4 6 8 10	✓
Average Depth of Inheritance Tree	2 1 0	✓
Coupling between Objects	0 25 250	✓
Response for a Class	0100 1000	✓

0.7.2 Analisi dinamica

Nell'iterazione 2 sono state testate tutte le API REST implementate, utlizzando Postman . In particolare sono state testate le seguenti funzionalita:

- ProfileController:
  - Visualizzazione di tutti i profili registrati nel sistema;
  - Visualizzazione di un profilo specifico e verifica della correttezza dei dati ritornati;
  - 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;

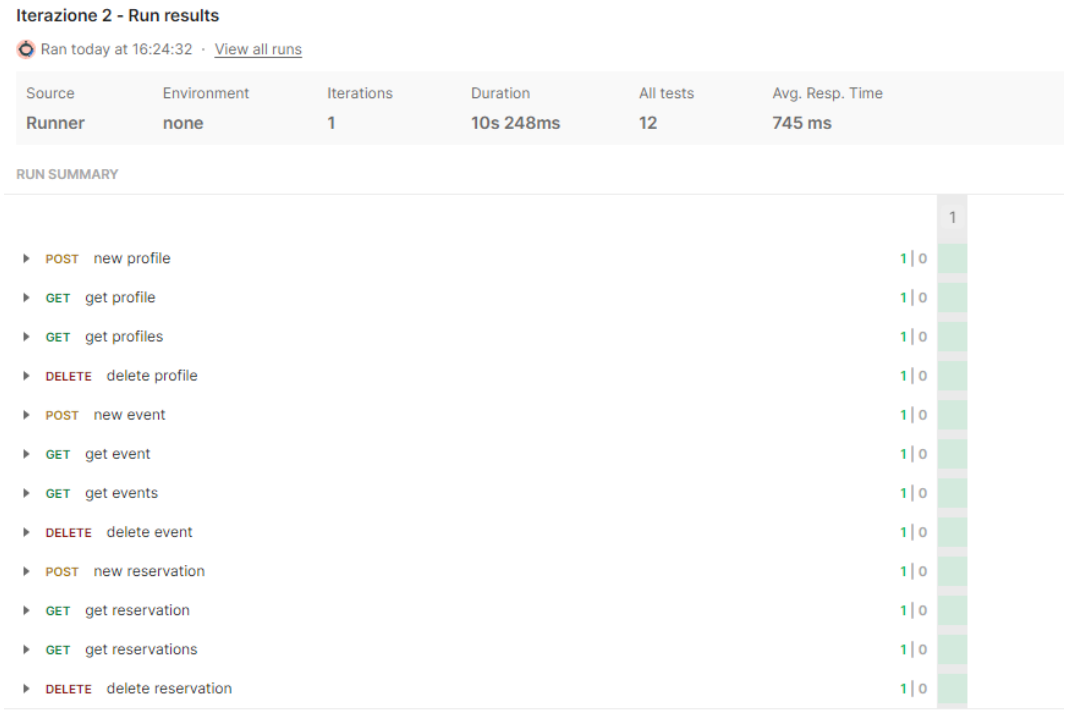
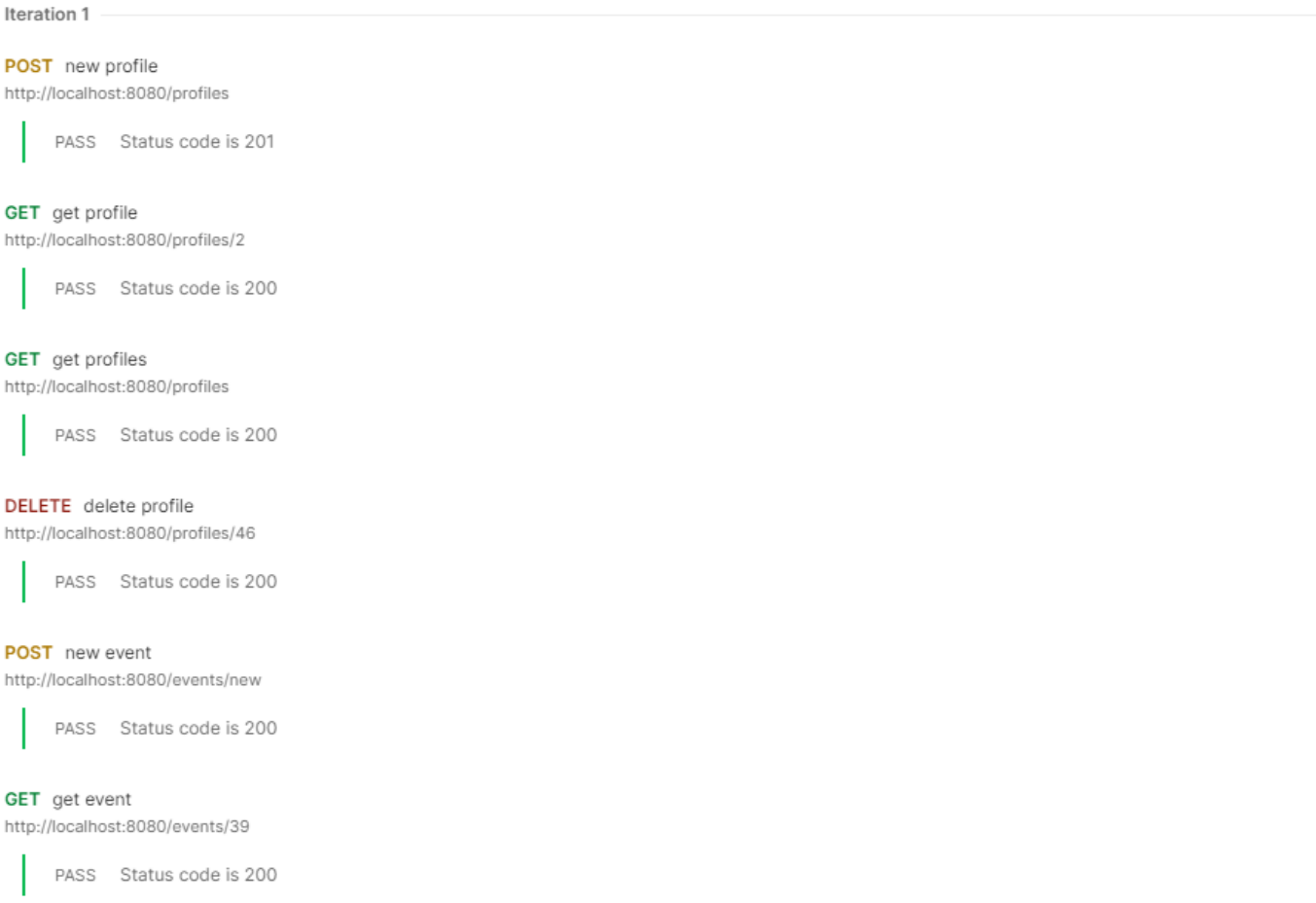


Figure 12: Test collection



0.7.3 Unit test

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

```
@DataJpaTest
class EventRepositoryTest {

        @Autowired
        private EventRepository underTest;

        @Test
        void findById() {

        String stringaTest = "test";
        //given
        Date date = new Date(100);
        Event expected = new Event(stringaTest, stringaTest,
        EventLevel.MEDIUM, date, stringaTest, stringaTest,
        stringaTest, stringaTest, stringaTest, stringaTest,
        stringaTest, new Integer(4));
        underTest.save(expected);

        //when
        Event result = underTest.findById(expected.getId()).get();

        //then
        assertThat(expected.getDescription())
        .isEqualTo(result.getDescription());
        }
}
```

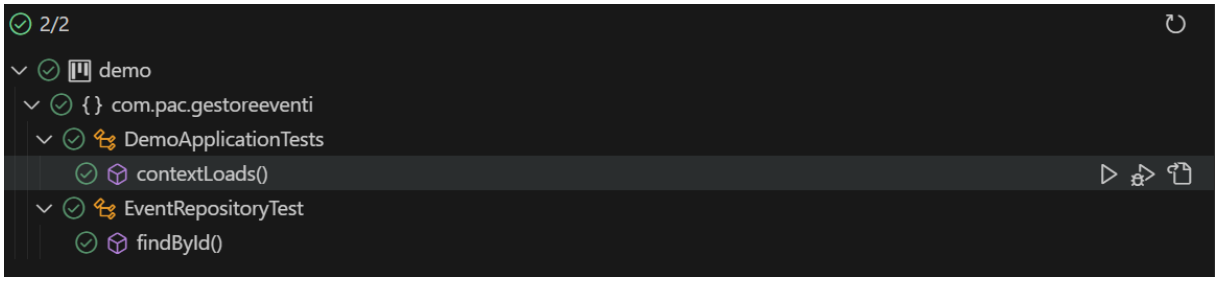


Figure 13: findById()