

# Documentazione Progetto



## Università degli Studi di Bergamo

Laurea Magistrale di Ingegneria Informatica

A.A. 2021/2022

Corso di Progettazione, Algoritmi e Computabilità (9 CFU)

cod. Corso 38090

Carne Federico – 1059865

Gelosa Federico – 1060253

# Sommario

<b>1</b>	<b>Iterazione 0</b>	<b>5</b>
1.1	Toolchain	7
1.2	Casi d'uso	8
1.2.1	User stories	8
1.2.2	Requisiti non funzionali	12
1.3	Topologia	12
1.4	Stili architetturali e Design Pattern	14
1.4.1	Architettura a microservizi	14
1.4.2	Model-View-ViewModel e Single Source of Truth	15
1.4.3	Altri pattern	16
<b>2</b>	<b>Iterazione 1</b>	<b>17</b>
2.1	Casi d'uso - Fully Dressed Description	17
2.2	Component diagram	23
2.3	Class diagrams	25
2.4	Deployment diagram	26
2.5	Analisi	28
2.5.1	Analisi statica	28
2.5.2	Analisi dinamica	29
2.6	Documentazione API	30
<b>3</b>	<b>Iterazione 2</b>	<b>31</b>
3.1	Casi d'uso - Fully Dressed Description	31
3.2	Component diagram	39
3.3	Class diagrams	41
3.4	Deployment diagram	42
3.5	Analisi	44
3.5.1	Analisi statica	44
3.5.2	Analisi dinamica	44
3.6	Documentazione API	46
<b>4</b>	<b>Iterazione 3</b>	<b>47</b>
4.1	Casi d'uso - Fully Dressed Description	47
4.2	Component diagram	51
4.3	Class diagrams	53

4.4	Deployment diagram.....	54
4.5	Traveling Salesman Problem: definizione e algoritmi .....	56
4.5.1	2-Approximation Metric TSP Algorithm .....	57
4.5.2	Christofides Approximation Algorithm.....	57
4.6	Analisi.....	59
4.6.1	Analisi statica .....	59
4.6.2	Analisi dinamica.....	59
4.7	Documentazione API.....	60
<b>5</b>	<b>Conclusioni.....</b>	<b>62</b>
5.1	Guida installazione .....	62
5.2	Breve guida utente .....	64

## Indice delle figure

Figura 1:	Requisiti del sistema .....	6
Figura 2:	UML Use Case diagram.....	10
Figura 3:	Topology diagram.....	13
Figura 4:	Istanziamento del topology diagram .....	14
Figura 5:	Modello concettuale dell'architettura esagonale .....	15
Figura 6:	Modello concettuale del MVVM con SSOT.....	16
Figura 7:	Iterazione 1 - Component diagram .....	24
Figura 8:	Sequence diagram authentication e authorization.....	25
Figura 9:	Iterazione 1 - Struttura interna user-service .....	25
Figura 10:	Iterazione 1 – Interfacce.....	26
Figura 11:	Iterazione 1 - Entità.....	26
Figura 12:	Iterazione 1 - Transfer objects .....	26
Figura 13:	Iterazione 1 - Deployment diagram .....	27
Figura 14:	Iterazione 1 - Composition view user-service.....	28
Figura 15:	Iterazione 1 - Risultati test .....	29
Figura 16:	Iterazione 1 - Documentazione API.....	30
Figura 17:	Iterazione 2 - Component diagram.....	40
Figura 18:	Iterazione 2 - Struttura interna itinerary-service .....	41
Figura 19:	Iterazione 2 - Interfacce .....	41

Figura 20: Iterazione 2 - Entità.....	42
Figura 21: Iterazione 2 - Transfer objects .....	42
Figura 22: Iterazione 2 - Deployment diagram .....	43
Figura 23: Iterazione 2 - Composition view itinerary-service.....	44
Figura 24: Iterazione 2 - Risultati test .....	45
Figura 25: Iterazione 2 - Documentazione API.....	46
Figura 26: Iterazione 3 - Component diagram.....	52
Figura 27: Iterazione 3 - Struttura interna route-service .....	53
Figura 28: Iterazione 3 - Interfacce .....	53
Figura 29: Iterazione 3 - Entità.....	54
Figura 30: Iterazione 3 - Transfer objects .....	54
Figura 31: Iterazione 3 - Deployment diagram .....	55
Figura 32: Iterazione 3 - Composition view itinerary-service.....	59
Figura 33: Iterazione 3 - Risultati test .....	60
Figura 34: Iterazione 3 - Documentazione API.....	61
Figura 35: Schermata del discovery-service.....	63
Figura 36: Screen 1 - Login.....	64
Figura 37: Screen 2 - Schermata principale.....	64
Figura 38: Screen 3 - Navigation drawer.....	65
Figura 39: Screen 4 - Lista POI.....	65
Figura 40: Screen 5 - Generazione percorsi .....	66

## Indice delle tabelle

Tabella 1: Toolchain e tecnologie utilizzate.....	7
Tabella 2: Casi d'uso gestione utente.....	11
Tabella 3: Casi d'uso gestione itinerario.....	11
Tabella 4: Casi d'uso generazione percorso.....	12

# 1 Iterazione 0

Lo scopo di questo progetto è sviluppare un sistema che permetta agli utenti di organizzare i propri itinerari di viaggio. Il contesto di utilizzo principale del sistema è la pianificazione di itinerari di visita di città d'arte, in quanto spesso si tratta di viaggi di breve durata in cui si cerca di visitare la maggior parte delle attrazioni o monumenti della città. Il problema che si pone è quindi: “Come poter visitare tutto ciò che si vuole visitare?”. Una delle principali necessità che il sistema sviluppato cerca di soddisfare è proprio la creazione di un percorso di visita che sia ottimo, ossia il più breve possibile.

Il secondo obiettivo del progetto è invece l'applicazione dei pattern architetturali e di design in piccolo più diffusi oggi, ma anche di best practices, in modo tale da produrre un sistema di miglior qualità.

Lo sviluppo del software ha seguito l'approccio AMDD, il quale implica una prima fase di envisioning del sistema. Questa fase ha avuto inizio con l'analisi di un possibile documento dei requisiti del sistema (Figura 1), che descrive il sistema dal punto di vista dell'utente e dei suoi bisogni.

Si vuole progettare un sistema per l'organizzazione e la gestione di itinerari di viaggio. Ogni itinerario è composto da una lista di luoghi d'interesse per l'utente, un nome e dalla data nel quale è stato effettuato. Nel seguito del documento si farà riferimento ai luoghi di interesse utilizzando il termine “POI” (Points of Interest).

La ricerca dei POI può avvenire direttamente su una mappa oppure indicandone l'indirizzo. Un POI può anche essere cercato tramite nome o categoria (e.g. museo d'arte, ristorante, ...), in questo caso saranno presentati all'utente tutti i POI che rispettano il criterio di ricerca e sarà l'utente a scegliere quali aggiungere all'itinerario. Per poter cercare un POI non è necessario che l'utente abbia creato precedentemente un itinerario. Durante il salvataggio del POI sarà richiesto all'utente se definire un nuovo itinerario, nel caso in cui non ne sia già stato selezionato uno al quale aggiungere il POI. Un POI può anche essere spostato da un itinerario ad un altro oppure semplicemente rimosso. L'utente può anche visualizzare le informazioni dei POI già aggiunti direttamente sulla mappa oppure recandosi in un'apposita schermata.

Per poter generare un cammino, l'utente seleziona alcuni oppure tutti i POI salvati in un itinerario. L'utente può generare un percorso manualmente, ordinando i POI selezionati, oppure richiedere al sistema la generazione automatica di un cammino che minimizzi la "fatica" per percorrerlo. In una prima versione del sistema la "fatica" è semplicemente la distanza, in linea retta, tra un punto e un altro. In rilasci futuri è probabile che la definizione di "fatica" cambi. Si chiede pertanto che vengano attuate soluzioni per ridurre l'impatto di tale modifica sul sistema. In caso di generazione automatica, il POI di inizio percorso è il più vicino all'attuale posizione dell'utente, determinata attraverso servizi di localizzazione; l'utente può però indicare esplicitamente il POI da cui iniziare l'itinerario. Il percorso dev'essere presentato graficamente all'utente tramite una mappa.

Un cammino, generato sia manualmente sia automaticamente, può essere modificato dall'utente. In caso di rimozione di un POI appartenente a un cammino, il "predecessore" e il "successore" del punto in questione saranno collegati direttamente (anche se il percorso generatosi in questo modo non è più ottimo). Viene data la possibilità all'utente di visualizzare i dettagli del cammino, cioè i dati dei singoli POI, la distanza tra un POI e il successivo, il tempo di percorrenza di ogni tratta, la distanza e il tempo totali per percorrere il cammino. L'utente può decidere di avviare e seguire il cammino generato (in una prima versione del sistema ciò avviene marcando manualmente i POI una volta che sono stati visitati).

Ogni utente ha la possibilità di mantenere più itinerari contemporaneamente e di visualizzare lo storico degli itinerari passati. Un utente può inoltre condividere un itinerario con un altro utente attraverso un link permanente.

Visto il contesto, l'utente dev'essere in grado di usare il sistema mentre è in movimento e su dispositivi normalmente di dimensioni ridotte e con capacità computazionali limitate. Ogni utente deve inoltre poter disporre dei propri itinerari su più dispositivi, facendo in modo che i dati siano il più possibile sincronizzati. Solo un utente registrato potrà aver accesso ai servizi offerti dal sistema. Un utente può anche richiedere al sistema la completa rimozione dei propri dati.

*Figura 1: Requisiti del sistema*

Come si può intuire dai requisiti, il sistema da sviluppare sarà composto da due parti: una app che l'utente può installare sul proprio dispositivo e un backend per la persistenza e sincronizzazione dei dati e, soprattutto, per la generazione automatica del cammino ottimo.

Se interpretiamo un itinerario come un grafo, i cui nodi sono i POI e gli archi sono le strade che collegano i POI, la richiesta di generazione di un percorso ottimo si traduce nel problema di trovare un cammino che visiti tutti i nodi e che sia anche di costo minimo. Questo problema è noto in letteratura come il Problema del Commesso Viaggiatore (Traveling Salesman Problem, TSP) ed è considerato NP-completo. È necessario quindi che i calcoli non siano svolti da un dispositivo mobile, sia a causa della capacità computazionale dei dispositivi, sia per permettere all'utente di risparmiare batteria.

## 1.1 Toolchain

In Tabella 1 sono riportati le tecnologie e i tool utilizzati durante le varie iterazioni di progettazione e sviluppo del sistema.

*Tabella 1: Toolchain e tecnologie utilizzate*

<b>Attività/Utilizzo</b>	<b>Tool</b>
Modellazione UML	Astah UML
Linguaggio di programmazione backend	Java
Linguaggio di programmazione frontend	Kotlin
Framework backend	Spring Boot
DBMS	MongoDB Atlas
IDE backend	Eclipse
IDE frontend	Android Studio
Mapping API	OpenStreetMap
Analisi statica	STAN4J
Analisi dinamica - Unit test	JUnit 5 & Mockito
Analisi dinamica - Integration test	Spring Boot
Coverage	EclEmma
Documentazione codice	Javadoc
Documentazione API	Springdoc – OpenAPI 3
Versioning	Git & GitHub
Condivisione documentazione	Google Drive
Organizzazione e schedulazione attività	Trello
Team meetings	Discord

## 1.2 Casi d'uso

L'analisi dei requisiti è stata condotta attraverso l'uso delle User Stories, una metodologia agile per lo sviluppo delle specifiche software la cui particolarità è focalizzarsi sugli utenti e sui loro bisogni. Un tipico template è il seguente:

*As a < type of user >, I want < some goal > so that < some reason >*

Nel nostro caso il tipo di utente è uno solo e nel seguito della documentazione verrà indicato semplicemente come *utente*. Le user stories prodotte saranno dettagliate nelle iterazioni successive e serviranno per modellare il sistema

### 1.2.1 User stories

La prima versione delle user stories è stata stilata durante una sessione di brainstorming in cui, impersonando un possibile utente, sono stati formulati i bisogni da soddisfare e le funzionalità richieste. La lista di user stories è stata man mano perfezionata e successivamente suddivisa in tre macrogruppi.

Nel seguito sono riportate le user stories suddivise nei tre gruppi individuati.

#### **GESTIONE ACCOUNT UTENTE**

- Come utente, voglio potermi registrare al sistema;
- Come utente, voglio poter effettuare il login al mio account;
- Come utente, voglio poter effettuare il logout dal mio account;
- Come utente, voglio poter eliminare il mio account;
- Come utente, voglio poter recuperare le credenziali del mio account;
- Come utente, voglio poter sincronizzare le preferenze tra i miei dispositivi;
- Come utente, voglio poter condividere i miei itinerari attraverso un link;
- Come utente, voglio poter visualizzare un itinerario condiviso tramite link;

#### **GESTIONE ITINERARI E POI**

- Come utente, voglio poter cercare (e salvare) un POI sulla mappa;
- Come utente, voglio poter cercare (e salvare) un POI attraverso l'indirizzo;
- Come utente, voglio poter cercare (e salvare) un POI attraverso il nome;



- Come utente, voglio poter cercare (e salvare) un POI per categoria;
- Come utente, voglio poter vedere le informazioni riguardanti un POI;
- Come utente, voglio poter creare un nuovo itinerario;
- Come utente, voglio poter eliminare un itinerario;
- Come utente, voglio poter rimuovere un POI salvato;
- Come utente, voglio poter spostare un POI da un itinerario ad un altro;
- Come utente, voglio poter visualizzare lo storico dei miei itinerari passati;
- Come utente, voglio poter visualizzare gli itinerari ancora da effettuare;

## **GENERAZIONE PERCORSO**

- Come utente, voglio poter selezionare alcuni o tutti i POI salvati da includere in un percorso;
- Come utente, voglio poter creare un percorso manualmente ordinando i POI selezionati;
- Come utente, voglio poter chiedere al sistema la generazione di un percorso di costo minimo sulla base della mia posizione;
- Come utente, voglio poter chiedere al sistema la generazione di un percorso di costo minimo indicando il POI di inizio percorso;
- Come utente, voglio poter modificare/cancellare un percorso generato;
- Come utente, voglio poter visualizzare i dettagli di un percorso generato;
- Come utente, voglio poter percorrere il cammino generato;

A partire da questa lista è stato prodotto un diagramma UML dei casi d'uso (Figura 2), in cui è possibile notare la presenza di due ulteriori attori: un server di autenticazione e una Map API che offre servizi come il calcolo della distanza tra due punti geografici o la ricerca di un punto dato l'indirizzo.

Si è scelto di dividere i casi d'uso in due ulteriori gruppi: i casi d'uso “indispensabili” e che caratterizzano il sistema e quelli che invece “fanno comodo” e aumentano la facilità d'uso del software. Tutti i casi d'uso sono stati descritti più approfonditamente ma solo quelli del primo gruppo sono stati implementati, in quanto è già possibile soddisfare i bisogni espressi dai casi d'uso del secondo gruppo facendo leva sui casi d'uso

implementati. Per esempio “Sposta POI in itinerario” può essere realizzato eliminando il POI scelto da un itinerario e cercandolo manualmente nel nuovo itinerario.

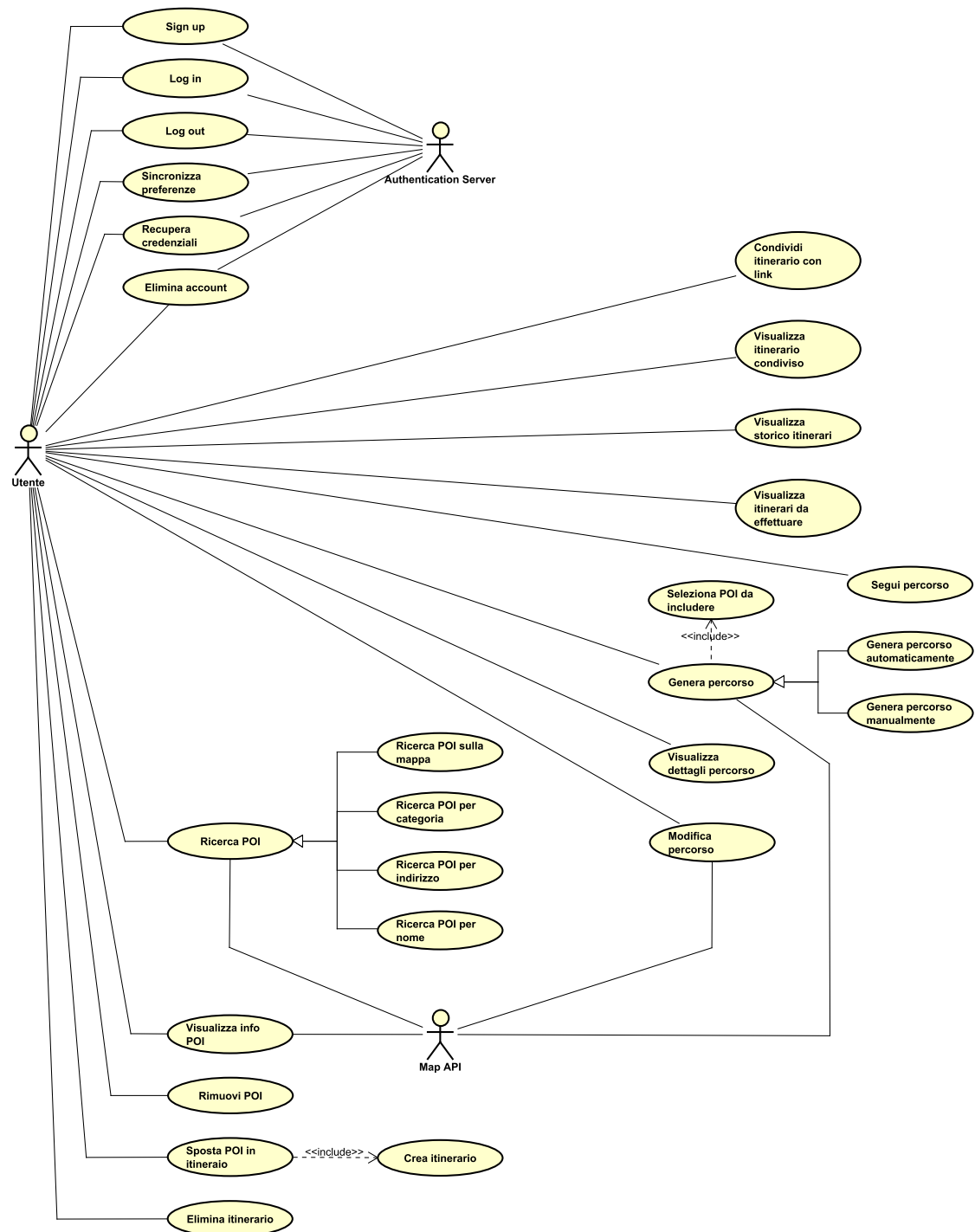


Figura 2: UML Use Case diagram

Nelle tabelle 2, 3 e 4 sono riportati i casi d’uso e la relativa scelta di implementazione.

Tabella 2: Casi d'uso gestione utente

<b>ID</b>	<b>CASO D'USO</b>	<b>IMPLEMENTATO</b>
<b>UC01</b>	Sign up	Sì
<b>UC02</b>	Log in	Sì
<b>UC03</b>	Log out	Sì
<b>UC04</b>	Sincronizza preferenze	Sì
<b>UC05</b>	Elimina account	Sì
<b>UC06</b>	Recupera credenziali	No
<b>UC07</b>	Condividi itinerario con link	No
<b>UC08</b>	Visualizza itinerario condiviso	No

Tabella 3: Casi d'uso gestione itinerario

<b>ID</b>	<b>CASO D'USO</b>	<b>IMPLEMENTATO</b>
<b>UC09</b>	Ricerca POI sulla mappa	Sì
<b>UC10</b>	Ricerca POI per indirizzo	Sì
<b>UC11</b>	Ricerca POI per nome	Sì
<b>UC12</b>	Ricerca POI per categoria	No
<b>UC13</b>	Crea itinerario	Sì
<b>UC14</b>	Elimina itinerario	Sì
<b>UC15</b>	Rimuovi POI salvato	Sì
<b>UC16</b>	Sposta POI in itinerario	No
<b>UC17</b>	Visualizza info POI	Sì
<b>UC18</b>	Visualizza itinerari da effettuare	Sì
<b>UC19</b>	Visualizza storico itinerari	Sì

Tabella 4: Casi d'uso generazione percorso

ID	CASO D'USO	IMPLEMENTATO
UC20	Seleziona POI da includere	Sì
UC21	Genera percorso manualmente	Sì
UC22	Genera percorso automaticamente	Sì
UC23	Modifica percorso	Sì
UC24	Visualizza dettagli percorso	Sì
UC25	Effettua percorso	No

Unico caso d'uso che fa eccezione è il caso d'uso “Segui percorso” che, seppur caratterizzante, si è deciso di non implementare vista la difficoltà di creare un servizio di routing.

### 1.2.2 Requisiti non funzionali

Nel documento dei requisiti sono presentati anche alcuni requisiti non funzionali, per esempio:

- la manutenibilità, espressa dalla richiesta di poter cambiare il modo in cui si calcola la “fatica” senza impattare eccessivamente sul sistema;
- l'usabilità, visto che l'utente si troverà a utilizzare l'app attraverso un dispositivo mobile con dimensioni ridotte e batteria limitata;
- la security, dal momento che dati relativi all'utente sono mantenuti sui database propri del sistema.

## 1.3 Topologia

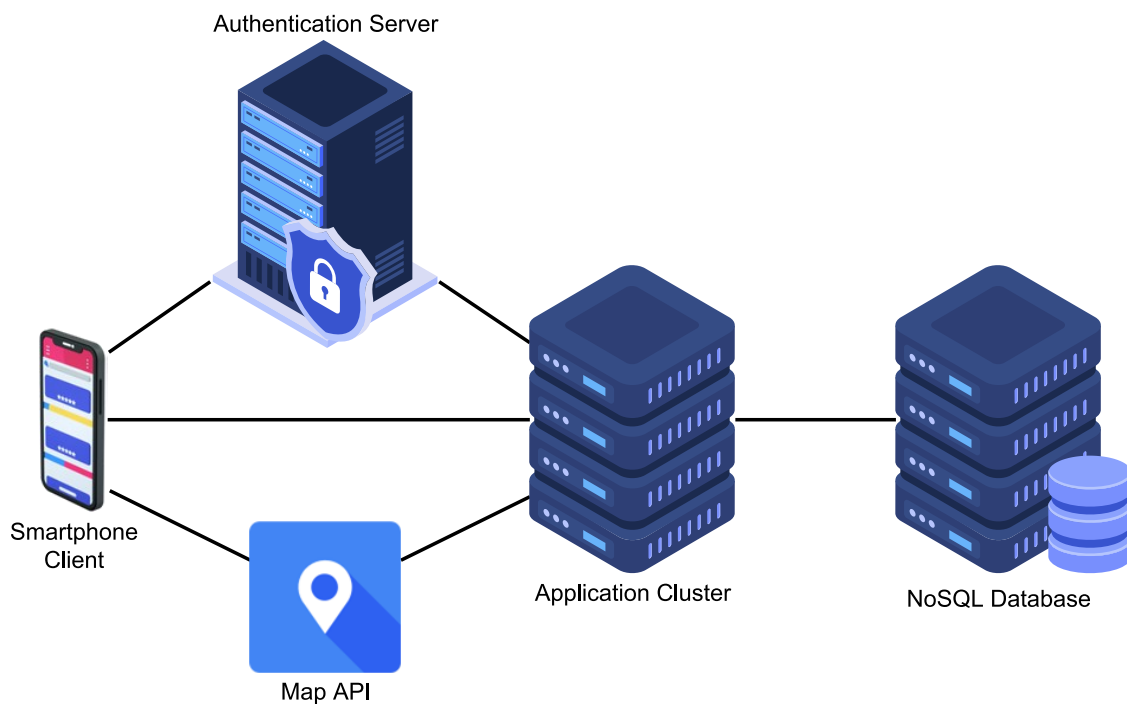
La topologia del sistema da sviluppare è mostrata in Figura 3 mentre la Figura 4 rappresenta l'istanziatura scelta, ovvero sono stati selezionati i provider e i servizi veri e propri con cui il sistema si interfacerà. Come si può notare dal diagramma, il sistema è diviso in due parti:

- 1) il backend, formato da uno o più server fisici, che espone servizi attraverso API REST;
- 2) un'applicazione, da installare sui propri dispositivi, con cui poter accedere ai servizi offerti dal backend e dagli altri provider.

Sia il cluster sia lo smartphone comunicano con l'authentication server, per il quale si è scelto di affidarsi a Firebase Auth, e con la Map API. Tre alternative sono state prese in considerazione per l'API di mapping:

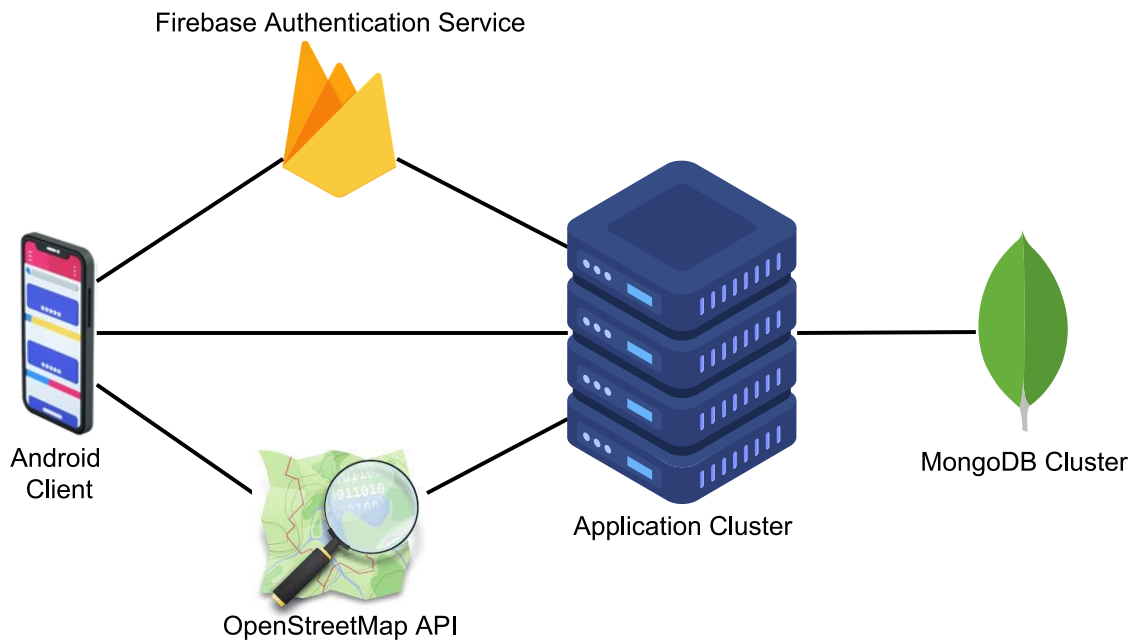
1. Google Maps
2. Mapbox
3. OpenStreetMap

Si è deciso di usare OpenStreetMap (OSM), e altri servizi basati su di essa, visto che è l'unica tra le alternative ad offrire API gratuite e ad uso illimitato.



*Figura 3: Topology diagram*

Infine, il backend comunica con un cluster di MongoDB Atlas. La scelta di un database NoSQL è stata presa in conseguenza ai tipi di entità che saranno ospitate nel database (insiemi e liste ordinate di punti geografici) e alla nota efficienza di MongoDB nella gestione di dati geografici.



*Figura 4: Istanziamento del topology diagram*

## 1.4 Stili architetturali e Design Pattern

Il sistema sviluppato, come già detto, si divide in due parti. Questo permette di poter scegliere gli stili architetturali più congeniali per ogni componente. Infatti, il backend è stato sviluppato secondo l'architettura a microservizi, per l'applicazione si è invece deciso di usare il pattern Model-View-ViewModel consigliato da Android.

### 1.4.1 Architettura a microservizi

Come già osservato, i casi d'uso si dividono in tre macrogruppi. La naturale conseguenza di questo fatto è la divisione del backend in tre servizi, responsabili di un solo aspetto del sistema, quindi molto coesi al loro interno, ma poco accoppiati tra di loro. I tre servizi possono allora essere implementati seguendo l'architettura a microservizi. Ciò significa che ognuno dei tre servizi può essere sviluppato e distribuito indipendentemente dagli altri, a tal punto da non sapere nemmeno dove si trovino gli altri microservizi su cui sta facendo affidamento. Di questo aspetto si occupa un altro servizio, il servizio di discovery, tra i cui compiti vi è quello di registrare le corrispondenze tra il nome del microservizio e l'indirizzo del server che lo sta offrendo e risolvere le richieste (in modo simile a ciò che fa un server DNS). Infine, un API Gateway fa da unico access point per le richieste e gestisce alcuni aspetti comuni, come la security.

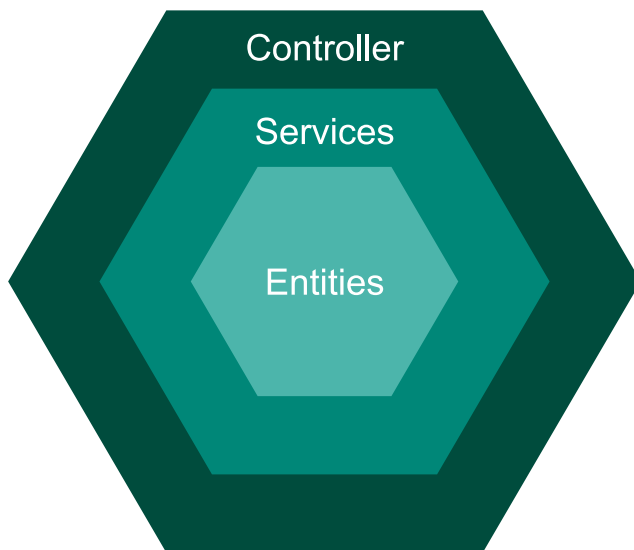


Figura 5: Modello concettuale dell'architettura esagonale

Al loro interno i tre servizi seguono il pattern esagonale: il servizio offerto viene implementato indipendentemente dal modo e dalla tecnologia con cui verrà poi esposto; nello strato esterno sono implementati i controller (o adapter) che “traducono” il servizio e lo espongono all'esterno. Nel nostro caso i controller espongono i servizi attraverso API REST.

Un altro dei vantaggi di questa architettura è la possibilità di scegliere lo stack tecnologico più adatto per ogni servizio, quindi anche il database (proprio di ogni microservizio). Nel nostro caso si è scelto di seguire il pattern *Private-tables-per-service*, secondo cui un servizio è proprietario di una o più tabelle (o collezioni) ed è l'unico a potervi accedere, mentre il database server è condiviso tra tutti i microservizi. Per fare in modo che ogni microservizio rispetti i propri confini, ad ognuno è stato associato uno specifico user, con permessi di lettura e scrittura solo per le collezioni di cui è considerato owner.

### 1.4.2 Model-View-ViewModel e Single Source of Truth

L'architettura dell'applicazione segue le raccomandazioni di Android e si basa sul pattern *Model-View-ViewModel (MVVM)*, variante del *Model-View-Presenter (MVP)*, le cui componenti principali sono le seguenti:

- **Model:** layer responsabile dell'astrazione delle sorgenti dei dati. Comunica con il ViewModel per la gestione dei dati;
- **View:** si occupa di notificare al ViewModel le azioni dell'utente. Osserva il ViewModel e non contiene nessun tipo di application logic. Può anche fare riferimento a più di un ViewModel;
- **ViewModel:** espone i dati che sono rilevanti per la View e fa da collegamento tra View e Model. Non ha nessun riferimento alla View.

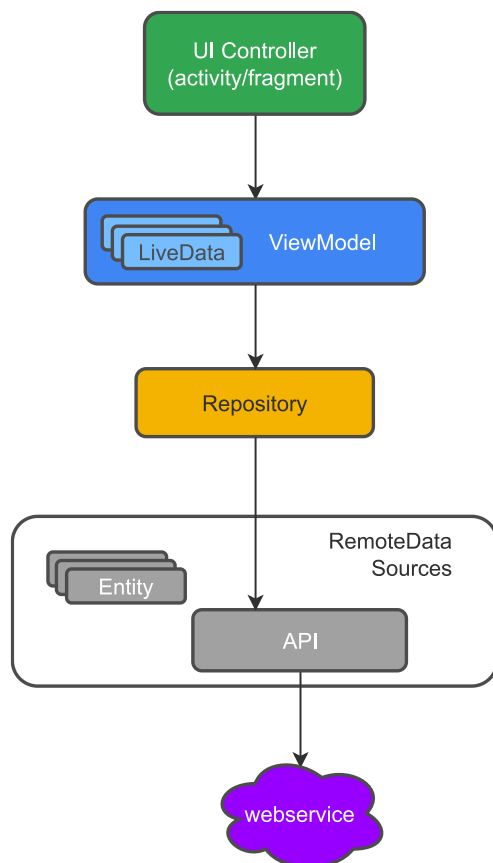


Figura 6: Modello concettuale del MVVM con SSOT (solo sorgenti remote)

### 1.4.3 Altri pattern

Un pattern di design in piccolo che è stato molto utilizzato durante lo sviluppo del codice è quello delle *Dependency Injection (DI)*. In questo pattern una classe, detta client, non si occupa di inizializzare le proprie dipendenze (definite attraverso interfacce) ma si aspetta che un altro componente, detto *injector*, le risolva al posto suo. Le dipendenze possono poi essere iniettate al client attraverso il costruttore o metodi setter. Questo pattern diminuisce l'accoppiamento, favorisce l'estensibilità e la testabilità del codice, permettendo il mocking durante lo unit testing. Il framework Spring offre nativamente l'annotazione `@Autowired`, che permette l'injection di tutti i componenti dichiarati come `@Bean`.

Un altro pattern utilizzato, di nuovo per ridurre l'accoppiamento, è lo *Strategy Pattern* per quanto riguarda l'implementazione dell'algoritmo risolutivo del TSP.



## 2 Iterazione 1

Vista la divisione dei casi d'uso in tre gruppi e la scelta di sviluppare un'architettura a microservizi, in ogni iterazione si è scelto di implementare un microservizio e la rispettiva controparte nell'applicazione. Nella prima iterazione si è scelto di implementare i casi d'uso, e quindi il servizio, relativi alla gestione utente (Tabella 2).

### 2.1 Casi d'uso - Fully Dressed Description

Nel seguito verranno descritti dettagliatamente, uno per uno, i casi d'uso. Il template scelto è simile a quello descritto da Cockburn in *Writing Effective Use Cases*. Per quanto riguarda la numerazione dei passi da eseguire in caso di eccezioni o alternative, la struttura è la seguente:

1. numero del passo al quale si verifica un'eccezione o un'alternativa;
2. una lettera per identificare l'eccezione o l'alternativa;
3. un numero progressivo che identifica i passi da eseguire per reagire all'eccezione o all'alternativa.

Per esempio, se si può verificare un'eccezione al passo 3 di un caso d'uso, i passi da eseguire saranno così numerati: 3a1, 3a2, 3a3, ... Se si può verificare un'altra eccezione sempre al passo 3, i passi da eseguire per rispondere all'eccezione sono così numerati: 3b1, 3b2, 3b3...

## UC01: SIGN UP

<b>ID</b>	<b>UC01</b>
<b>NAME</b>	Sign up
<b>SUMMARY</b>	L'utente compila il form di registrazione e viene aggiunto al database
<b>ACTORS</b>	Utente, Authentication Server
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per la registrazione
<b>PRECONDITION</b>	
<b>POSTCONDITION</b>	L'utente viene aggiunto al database dell'Authentication Server e al database del sistema
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente compila il form di registrazione, inserendo indirizzo e-mail e password</li><li>2. Il sistema verifica che le credenziali non siano già presenti nel database</li><li>3. Il sistema invia le credenziali all'Authentication Server</li><li>4. L'Authentication Server crea un nuovo utente con le credenziali inviate</li><li>5. Il sistema crea un nuovo account per l'utente appena registrato</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>L'utente seleziona il sign up tramite un identity provider</i></p> <ol style="list-style-type: none"><li>1a1. Il sistema mostra all'utente la schermata di signup propria dell'identity provider scelto</li><li>1a2. L'utente segue la procedura di signup propria dell'identity provider scelto</li></ol> <p><i>Le credenziali inserite sono già presenti</i></p> <ol style="list-style-type: none"><li>2a1. Il sistema notifica all'utente l'impossibilità di effettuare la registrazione</li></ol>

## UC02: LOG IN

<b>ID</b>	<b>UC02</b>
<b>NAME</b>	Log in
<b>SUMMARY</b>	L'utente compila il form di login e viene reindirizzato alla schermata principale del sistema
<b>ACTORS</b>	Utente, Authentication Server
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per il login
<b>PRECONDITION</b>	L'utente ha effettuato la registrazione
<b>POSTCONDITION</b>	L'utente possiede un token da inviare al sistema per l'autenticazione delle richieste
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente compila il form di login</li><li>2. Il sistema invia le credenziali all'Authentication Server</li><li>3. L'Authentication Server verifica che l'utente sia registrato</li><li>4. L'Authentication Server invia un token di autenticazione al sistema</li><li>5. Il sistema salva localmente al dispositivo dell'utente il token</li><li>6. Il sistema reindirizza l'utente alla schermata principale</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>L'utente seleziona il login tramite un identity provider</i></p> <p>1a1. Il sistema mostra all'utente la schermata di login propria dell'identity provider scelto</p> <p>1a2. L'utente segue la procedura di login propria dell'identity provider</p> <p><i>Le credenziali inserite non sono presenti</i></p> <p>2a1. L'Authentication Server notifica al sistema l'assenza di un account relativo alle credenziali inviate</p> <p>2a2. Il sistema notifica all'utente che le credenziali inserite sono errate</p>

### UC03: LOG OUT

<b>ID</b>	<b>UC03</b>
<b>NAME</b>	Log out
<b>SUMMARY</b>	L'utente effettua il logout dal sistema e viene reindirizzato alla schermata di login
<b>ACTORS</b>	Utente, Authentication Server
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per il logout
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Il sistema cancella il token salvato localmente
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema invia all'Authentication Server la richiesta di logout</li><li>2. L'Authentication Server effettua il logout dell'utente</li><li>3. Il sistema cancella il token di autenticazione salvato localmente al dispositivo dell'utente</li><li>4. Il sistema reindirizza l'utente alla schermata di login</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

### UC04: SINCRONIZZA PREFERENZE

<b>ID</b>	<b>UC04</b>
<b>NAME</b>	Sincronizza preferenze
<b>SUMMARY</b>	Le preferenze dell'utente vengono salvate nel database del sistema
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente apre la schermata delle preferenze
<b>PRECONDITION</b>	L'utente ha effettuato il login e l'impostazione per la sincronizzazione delle preferenze è attiva
<b>POSTCONDITION</b>	Le preferenze dell'utente sono presenti nel database
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente modifica le proprie preferenze</li><li>2. L'utente chiude la schermata di modifica delle preferenze</li><li>3. Il sistema registra le preferenze dell'utente</li><li>4. Alla prossima apertura della schermata il sistema aggiorna le preferenze locali presenti sul dispositivo dell'utente</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

## UC05: ELIMINA ACCOUNT

<b>ID</b>	<b>UC05</b>
<b>NAME</b>	Elimina account
<b>SUMMARY</b>	L'utente richiede al sistema l'eliminazione del proprio account
<b>ACTORS</b>	Utente, Authentication Server
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per l'eliminazione del proprio account
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	L'utente viene rimosso dal database dell'Authentication Server e dal database del sistema
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema chiede conferma all'utente di voler eliminare il proprio account</li><li>2. L'utente conferma la scelta</li><li>3. Il sistema elimina l'account dell'utente dal proprio database</li><li>4. Il sistema chiede all'Authentication Server l'eliminazione dell'account</li><li>5. L'Authentication Server elimina l'account dal proprio database</li><li>6. Il sistema indirizza l'utente alla schermata di login</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

## UC06: Recupera credenziali

<b>ID</b>	<b>UC06</b>
<b>NAME</b>	Recupera credenziali
<b>SUMMARY</b>	L'utente chiede il recupero delle credenziali per il proprio account
<b>ACTORS</b>	Utente, Authentication Server
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per il recupero delle credenziali
<b>PRECONDITION</b>	L'utente ha effettuato la registrazione
<b>POSTCONDITION</b>	La password dell'account dell'utente è stata modificata
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema richiede all'Authentication Server il recupero delle credenziali</li><li>2. L'Authentication Server procede all'attività di recupero delle credenziali</li><li>3. L'utente effettua il login con le nuove credenziali</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

**UC07: CONDIVIDI ITINERARIO CON LINK**

<b>ID</b>	<b>UC07</b>
<b>NAME</b>	Condividi itinerario con link
<b>SUMMARY</b>	L'utente condivide i dati di un itinerario di cui è il proprietario
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca l'apposito pulsante per la condivisione di un itinerario
<b>PRECONDITION</b>	L'utente ha effettuato il login e sta visualizzando un itinerario
<b>POSTCONDITION</b>	Un link contenente i dati dell'itinerario da condividere è stato inviato a un altro utente
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"> <li>1. Il sistema genera un link da poter inviare</li> <li>2. Il sistema notifica al sistema operativo l'intenzione di condividere un messaggio testuale</li> <li>3. L'utente invia il messaggio</li> </ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

**UC08: VISUALIZZA ITINERARIO CONDIVISO**

<b>ID</b>	<b>UC08</b>
<b>NAME</b>	Visualizza itinerario condiviso
<b>SUMMARY</b>	L'utente visualizza un itinerario che gli è stato condiviso tramite link da un altro utente
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente apre il link inviatogli
<b>PRECONDITION</b>	L'utente ha installato sul proprio dispositivo l'applicazione del sistema
<b>POSTCONDITION</b>	L'utente viene reindirizzato alla schermata della mappa, in cui sono marcati i POI contenuti nell'itinerario condiviso
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"> <li>1. Il sistema intercetta l'evento di apertura del link</li> <li>2. Il sistema recupera i dati dell'itinerario condiviso</li> <li>3. Il sistema apre la schermata contenente la mappa e la popola con i POI dell'itinerario</li> </ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>L'utente non ha effettuato il login</i>  3a1. Il sistema reindirizza l'utente alla schermata di login</p> <p><i>L'utente non è registrato</i>  3b1. Il sistema reindirizza l'utente alla schermata di sign up</p>

## 2.2 Component diagram

Partendo dai casi d'uso appena descritti e seguendo delle best practices per la modellazione del sistema, è stato prodotto il component diagram del sistema. Come si può vedere dalla Figura 7, e come già anticipato, lato backend sono presenti anche una componente per il servizio di discovery, implementato tramite Spring Eureka e un API Gateway che, nel nostro caso, fa da relay per le chiamate.

Il ruolo di authentication server è svolto da Firebase Authentication Server e si basa sulla verifica di Token JWT:

1. un utente effettua il login attraverso Firebase;
2. Firebase, se il login è avvenuto correttamente, invia all'utente un token JWT;
3. quando l'app invia una richiesta http al backend, inserisce un header contenente il token;
4. prima di accettare la richiesta, il microservizio destinatario verifica che il JWT ricevuto sia valido.

In Figura 8 è visibile un sequence diagram di una tipica interazione tra app, backend e authentication server.

La struttura interna del componente `user-service` è descritta in Figura 9. Come già detto, sono presenti una componente `service`, cioè il servizio vero e proprio, e una componente `controller` che sfrutta le funzionalità offerte del `service` per rispondere alle richieste http. La componente `repository` si occupa invece di comunicare con il cluster di MongoDB.

Lato app, invece, sono presenti due componenti: la prima si occupa di gestire il processo di autenticazione, la seconda consuma invece le API REST esposte dal backend. Quest'ultima componente è internamente realizzata secondo lo stile MVVM.

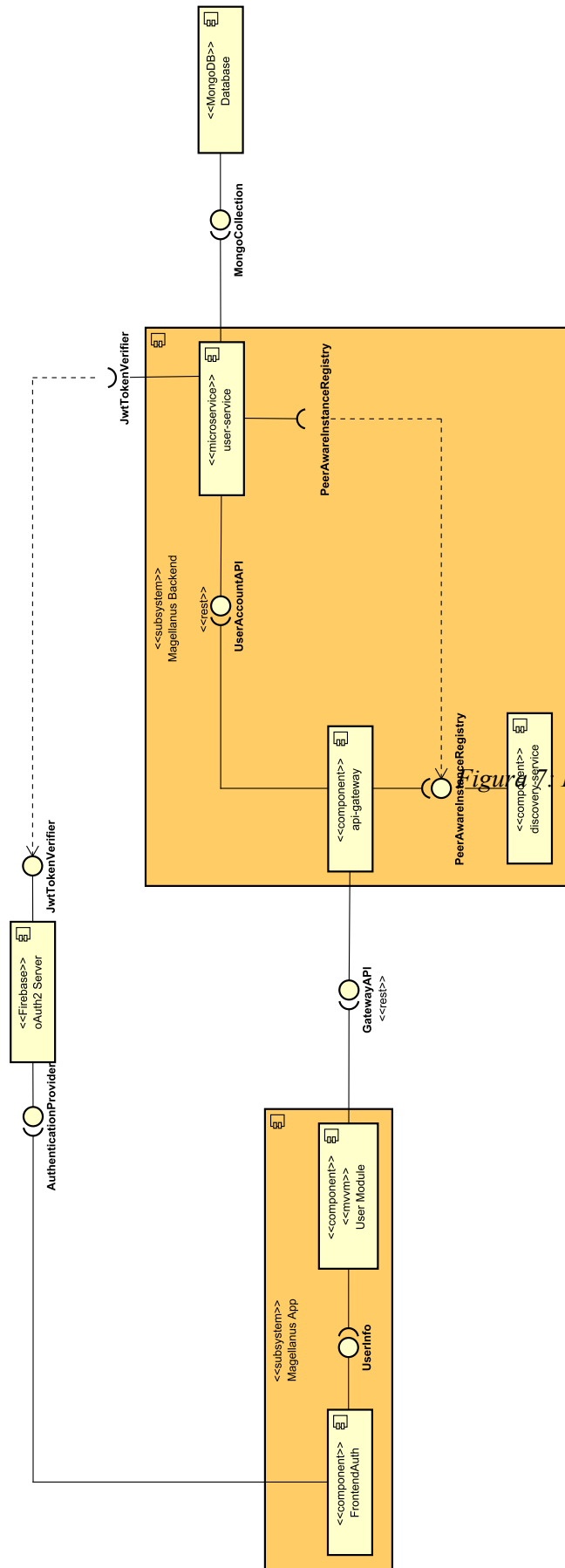


Figure 7 Iterazione 1 - Component diagram



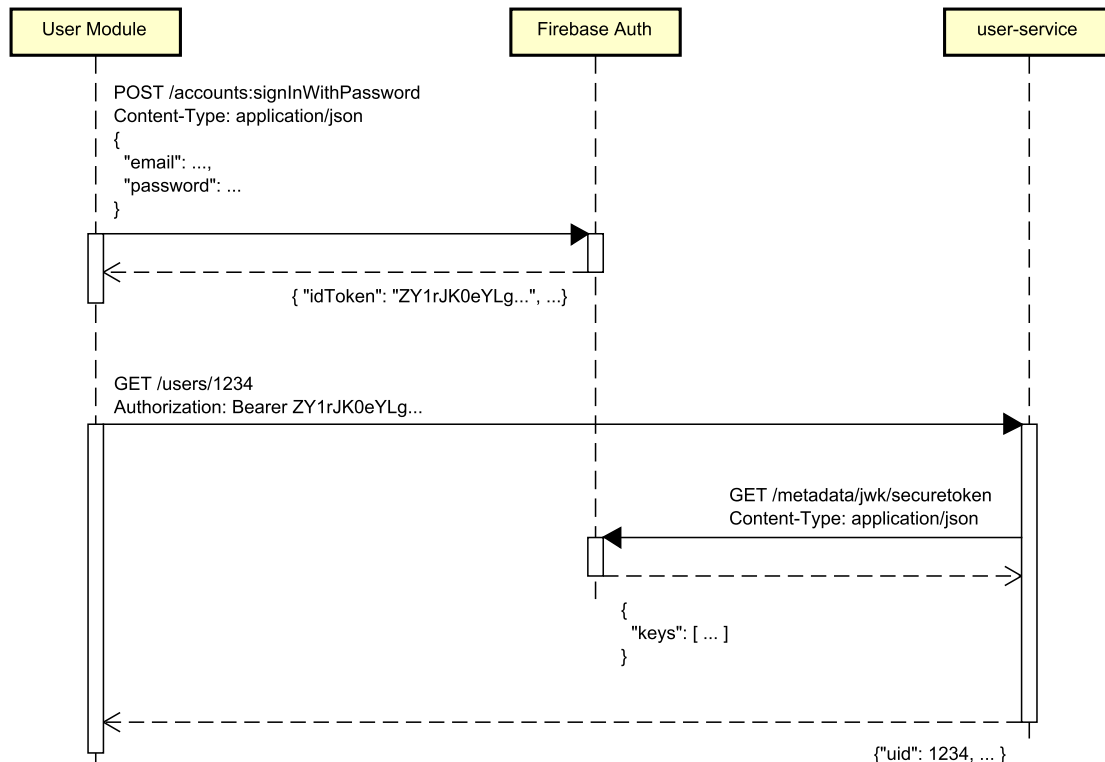


Figura 8: Sequence diagram authentication e authorization

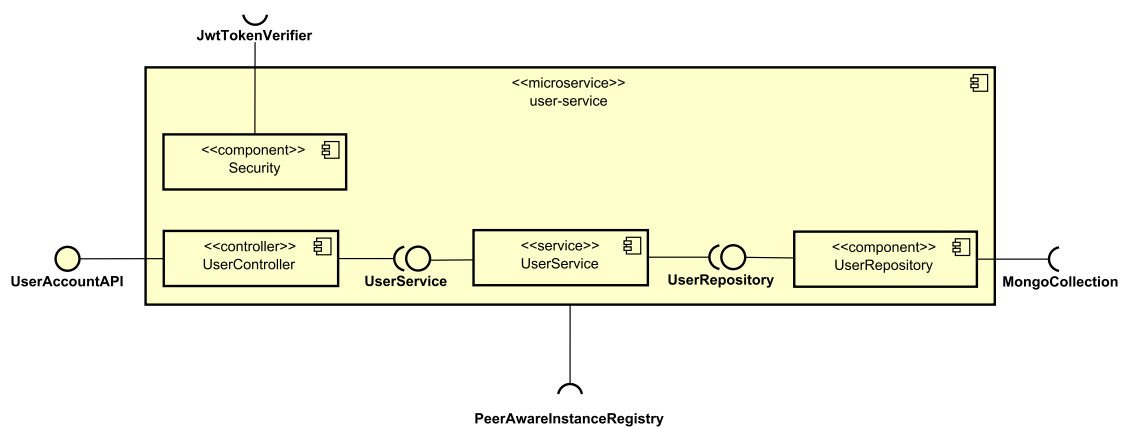


Figura 9: Iterazione 1 - Struttura interna user-service

## 2.3 Class diagrams

Le interfacce esposte e richieste dalle componenti realizzate nell'iterazione 1 sono descritte in Figura 10. Le interfacce che si trovano al di fuori dei package sono già presenti nel framework Spring (per questo motivo la loro segnatura è incompleta). GatewayAPI eredita dall'interfaccia UserAccountAPI per mostrare che tutto ciò che espone l'API viene esposto anche dal gateway.

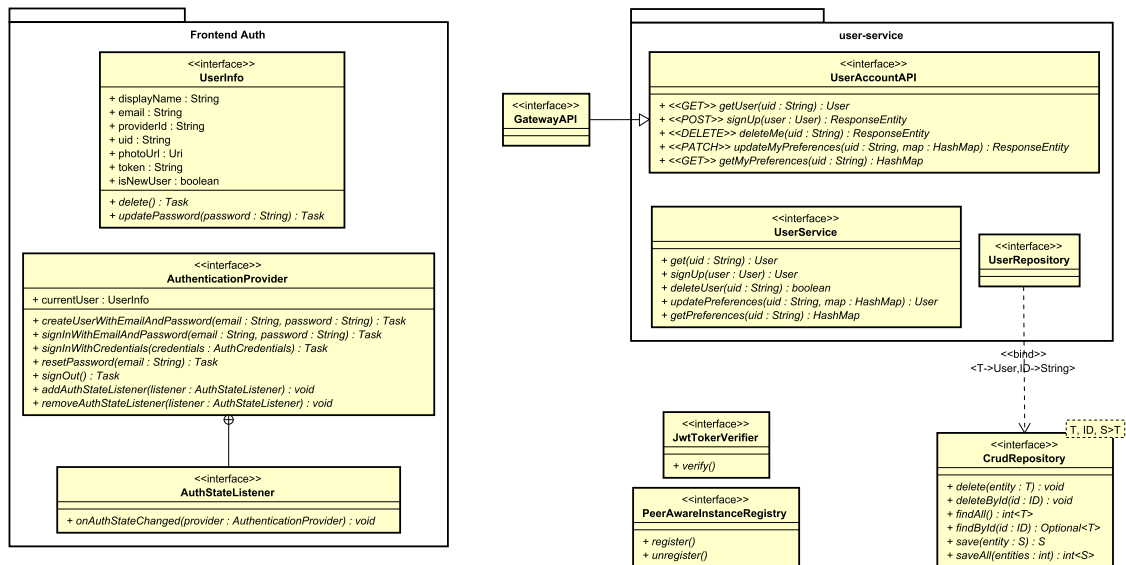


Figura 10: Iterazione 1 – Interfacce

Le entità memorizzate nelle collections di MongoDB sono presentate in Figura 11, queste rappresentano un utente e le sue preferenze. Il transfer object scambiato, in formato json, tra l'applicazione e il backend è descritto in Figura 12 (i transfer objects non è detto siano implementati nel codice, possono anche essere generati automaticamente dalle librerie).

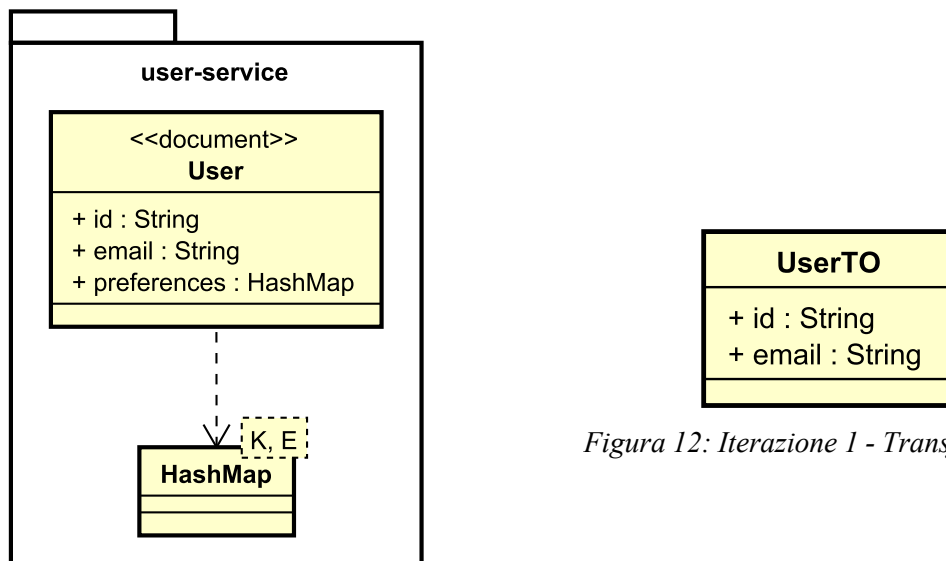


Figura 12: Iterazione 1 - Transfer objects

Figura 11: Iterazione 1 - Entità

## 2.4 Deployment diagram

Il deployment diagram del sistema (Figura 13) mostra la suddivisione dei vari componenti nei dispositivi che formano il sistema. Da notare come, data la struttura a microservizi, ogni servizio sia contenuto in un container.

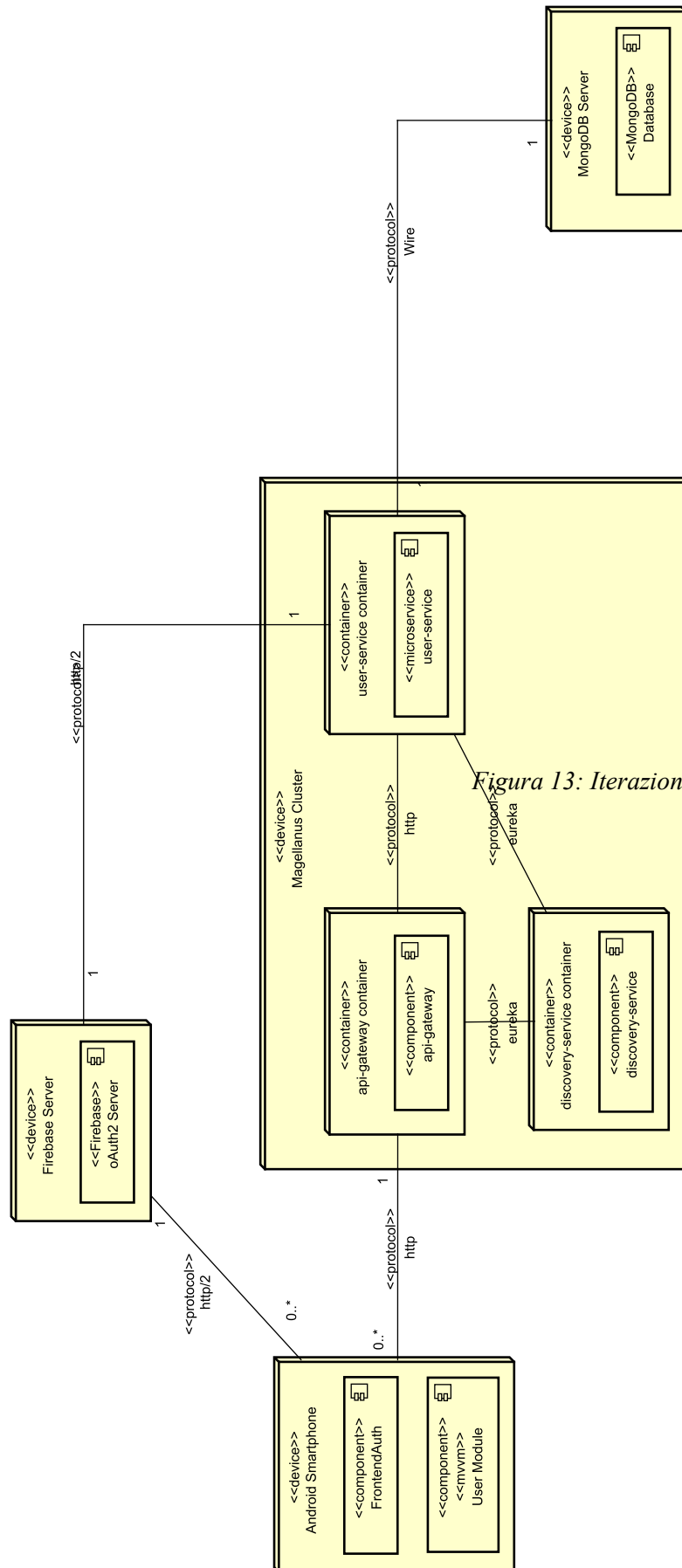


Figura 13: Iterazione 1 - Deployment diagram

## 2.5 Analisi

Il backend, in tutte le iterazioni, è stato sottoposto ad analisi statiche e dinamiche. Per quanto riguarda l'analisi dinamica si è scelto di fare uno unit test per la classe `@Service` e di fare un integration test dell'intero microservizio usando gli strumenti offerti da Spring stesso. La struttura dei test segue il pattern *Arrange, Act, Assert (AAA)*:

- **Arrange**: si costruiscono gli oggetti necessari all'esecuzione del test;
- **Act**: si esegue l'azione sul soggetto del testing;
- **Assert**: si controlla che l'azione abbia portato i risultati attesi.

I nomi dei test invece seguono la seguente struttura:

`metodoTestato_caratteristicaInput_condizioneRisultato`

Per esempio, `get_exists_returnsUser` testa il metodo `get`, passandogli l'id di un utente presente nel db e si aspetta che ritorni l'utente corrispondente. Per disaccoppiare il servizio testato dalle proprie dipendenze si è scelto di usare componenti mock.

### 2.5.1 Analisi statica

L'analisi condotta con STAN (disponibile nella repository GitHub) non mostra particolari anomalie e quindi il codice è di qualità soddisfacente per procedere con l'iterazione successiva.

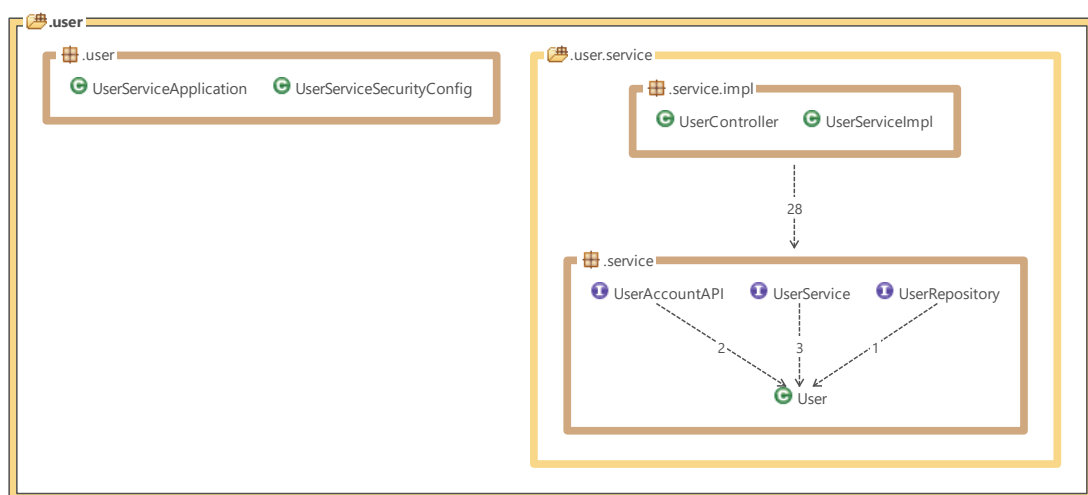


Figura 14: Iterazione 1 - Composition view user-service

## 2.5.2 Analisi dinamica

I test sono stati condotti sulla classe `UserServiceImpl` e sul microservizio nella sua interezza. Tutti i test sono stati superati e il coverage è del 92%. Si può affermare con elevata fiducia che il microservizio è corretto.

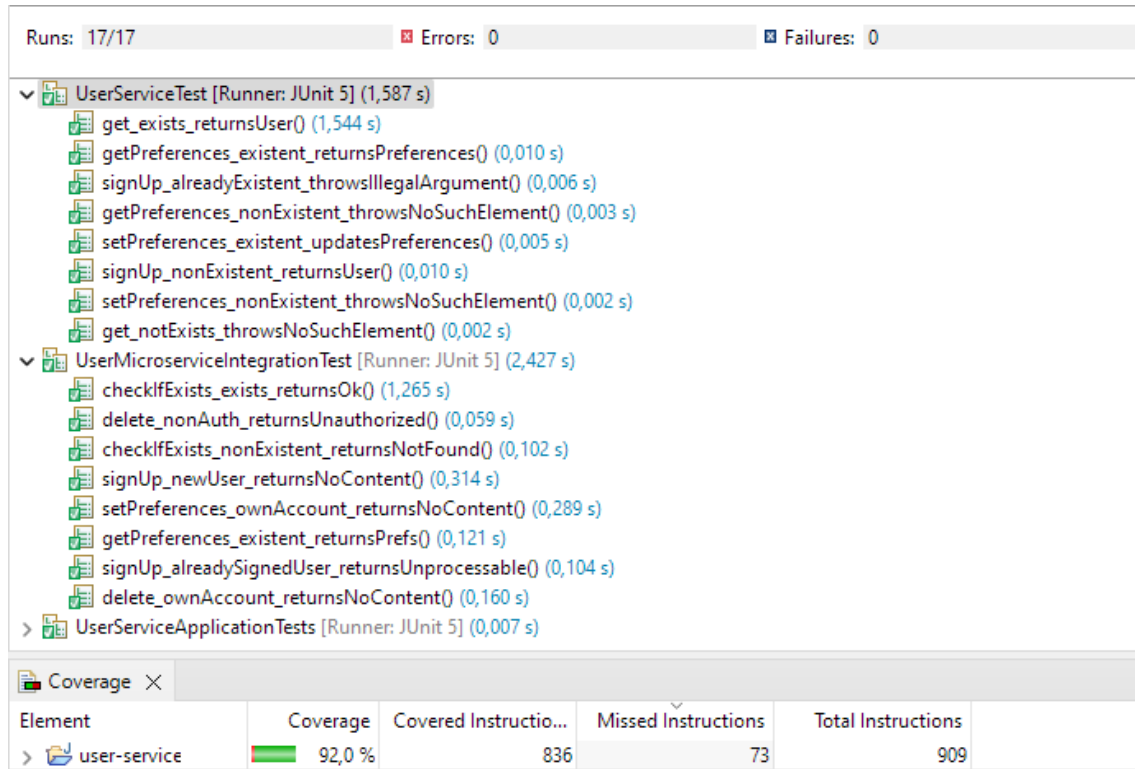


Figura 15: Iterazione 1 - Risultati test

## 2.6 Documentazione API

La documentazione dell'API è stata generata da Springdoc e segue la specifica di OpenAPI 3. È disponibile sia in formato json, sia yaml, sia grafico (Figura 16).

The screenshot displays the Swagger UI for the 'User API' (v1.0, OAS3). The top navigation bar includes the Swagger logo and a dropdown menu for 'Select a definition' with 'User API' selected. Below the header, the API title 'User API' is shown with version and OAS3 tags, along with a link to the API docs and the version number. The 'Servers' section shows a default server URL. An 'Authorize' button is present. The main content area lists endpoints for the 'user-controller'. The endpoints are: POST /, GET /me/preferences, PATCH /me/preferences, GET /{uid}, and DELETE /me. The DELETE /me endpoint is expanded, showing its details. The 'Parameters' section indicates 'No parameters'. The 'Responses' section shows a 200 status code with the description 'OK' and 'No links'. A 'Try it out' button is located next to the parameters section. The 'Schemas' section at the bottom shows a 'User' schema.

Figura 16: Iterazione 1 - Documentazione API

## 3 Iterazione 2

Nella seconda iterazione si è scelto di implementare i casi d'uso, e quindi il servizio, relativi alla gestione degli itinerari (Tabella 3).

### 3.1 Casi d'uso - Fully Dressed Description

#### UC09: RICERCA POI SULLA MAPPA

<b>Id</b>	<b>UC09</b>
<b>NAME</b>	Ricerca POI sulla mappa
<b>SUMMARY</b>	L'utente ricerca un POI spostandosi sulla mappa
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente apre la mappa di ricerca dei POI
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Viene marcato sulla mappa il POI cercato dall'utente
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente si sposta sulla mappa</li><li>2. L'utente tiene premuto il POI cercato</li><li>3. Il sistema invia a Map API le coordinate del punto premuto</li><li>4. La Map API invia al sistema i dati del POI cercato</li><li>5. Il sistema mostra le informazioni del POI cercato</li><li>6. L'utente conferma il salvataggio del POI</li><li>7. Il sistema aggiunge il POI all'itinerario selezionato</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

## UC10: RICERCA POI PER INDIRIZZO

<b>ID</b>	<b>UC10</b>
<b>NAME</b>	Ricerca POI per indirizzo
<b>SUMMARY</b>	L'utente ricerca un POI digitando l'indirizzo nell'apposita barra di ricerca
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente clicca sulla barra di ricerca
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Viene marcato sulla mappa (ed eventualmente salvato) il POI associato all'indirizzo cercato
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente digita nella barra di ricerca l'indirizzo del POI</li><li>2. Il sistema invia a Map API l'indirizzo digitato</li><li>3. La Map API invia al sistema i dati del POI cercato</li><li>4. Il sistema mostra il POI cercato sulla mappa</li><li>5. L'utente conferma il salvataggio del POI</li><li>6. Il sistema aggiunge il POI all'itinerario selezionato</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>Non esiste nessun POI con l'indirizzo cercato</i></p> <p>4a1. Il sistema notifica all'utente l'assenza del POI cercato</p> <p><i>Il POI non è quello effettivamente voluto dall'utente</i></p> <p>5a1. L'utente scarta il POI e non conferma il salvataggio</p> <p><i>L'utente non ha selezionato nessun itinerario</i></p> <p>6a1. L'utente sceglie in quale itinerario salvare il POI</p> <p>6b1. L'utente crea un nuovo itinerario</p>



## UC11: RICERCA POI PER NOME

<b>ID</b>	<b>UC11</b>
<b>NAME</b>	Ricerca POI per nome
<b>SUMMARY</b>	L'utente ricerca un POI digitando il nome del luogo di suo interesse nell'apposita barra di ricerca
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente clicca sulla barra di ricerca
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Viene marcato sulla mappa (ed eventualmente salvato) il POI associato al nome del luogo cercato
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente digita nella barra di ricerca il nome del POI</li><li>2. Il sistema invia a Map API il nome digitato</li><li>3. La Map API invia al sistema i dati dei POI corrispondenti</li><li>4. Il sistema mostra i POI trovati sulla mappa</li><li>5. L'utente conferma il salvataggio di un POI</li><li>6. Il sistema aggiunge il POI all'itinerario selezionato</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>Non esiste nessun POI con il nome cercato</i> 4a1. Il sistema notifica all'utente l'assenza di POI corrispondenti</p> <p><i>Il POI non è quello effettivamente voluto dall'utente</i> 5a1. L'utente scarta il POI e non conferma il salvataggio</p> <p><i>L'utente non ha selezionato nessun itinerario</i> 6a1. L'utente sceglie in quale itinerario salvare il POI 6b1. L'utente crea un nuovo itinerario</p>

## UC12: RICERCA POI PER CATEGORIA

<b>ID</b>	<b>UC12</b>
<b>NAME</b>	Ricerca POI per categoria
<b>SUMMARY</b>	L'utente ricerca un POI digitando il nome della categoria nell'apposita barra di ricerca e il sistema restituisce una lista di luoghi vicini all'ultimo POI cercato appartenenti a quella categoria
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente clicca sulla barra di ricerca
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Viene marcato sulla mappa (ed eventualmente salvato) il POI associato alla categoria cercata
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente digita nella barra di ricerca la categoria di luoghi da cercare</li><li>2. Il sistema invia a Map API la categoria digitata e le coordinate dell'ultimo POI salvato</li><li>3. La Map API invia al sistema i dati dei POI corrispondenti</li><li>4. Il sistema mostra i POI trovati sulla mappa</li><li>5. L'utente conferma il salvataggio di un POI</li><li>6. Il sistema aggiunge il POI all'itinerario selezionato</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>Non sono disponibili le coordinate dell'ultimo POI salvato</i> 2a1. Il sistema invia a Map API la categoria digitata e l'attuale posizione dell'utente</p> <p><i>Non esiste nessun POI con la categoria cercata</i> 4a1. Il sistema notifica all'utente l'assenza di POI adatti</p> <p><i>I POI non corrispondono a quanto effettivamente voluto dall'utente</i> 5a1. L'utente scarta i POI e non conferma il salvataggio</p> <p><i>L'utente non ha selezionato nessun itinerario</i> 6a1. L'utente sceglie in quale itinerario salvare il POI 6b1. L'utente crea un nuovo itinerario</p>

### UC13: CREA ITINERARIO

<b>ID</b>	<b>UC13</b>
<b>NAME</b>	Crea itinerario
<b>SUMMARY</b>	L'utente crea un nuovo itinerario, indicandone il nome
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per la creazione di un itinerario
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Il nuovo itinerario è presente nel database
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema mostra all'utente un form da compilare con i dati dell'itinerario</li><li>2. L'utente compila il form</li><li>3. L'utente conferma la creazione dell'itinerario</li><li>4. Il sistema salva il nuovo itinerario nel database</li><li>5. Il sistema conferma all'utente la creazione dell'itinerario</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

### UC14: ELIMINA ITINERARIO

<b>ID</b>	<b>UC14</b>
<b>NAME</b>	Elimina itinerario
<b>SUMMARY</b>	L'utente elimina un itinerario precedentemente creato e i dati ad esso relativo
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per l'eliminazione di un itinerario
<b>PRECONDITION</b>	L'utente ha effettuato il login, ha creato l'itinerario che vuole eliminare e si trova nella schermata di visualizzazione degli itinerari
<b>POSTCONDITION</b>	I dati dell'itinerario eliminato non sono più presenti nel database
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema chiede conferma all'utente di voler eliminare l'itinerario</li><li>2. L'utente conferma la scelta di voler eliminare l'itinerario scelto</li><li>3. Il sistema cancella i dati relativi all'itinerario scelto</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

### UC15: RIMUOVI POI SALVATO

<b>ID</b>	<b>UC15</b>
<b>NAME</b>	Rimuovi POI salvato
<b>SUMMARY</b>	Un POI salvato viene rimosso da un itinerario
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per rimuovere un POI
<b>PRECONDITION</b>	L'utente ha effettuato il login e si trova sulla mappa o nella schermata di riepilogo di un itinerario
<b>POSTCONDITION</b>	Il POI non è più presente nella lista dei POI di un itinerario
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema chiede conferma all'utente di voler rimuovere il POI</li><li>2. L'utente conferma di voler rimuovere il POI scelto</li><li>3. Il sistema rimuove il POI dall'itinerario</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

### UC16: SPOSTA POI IN ITINERARIO

<b>ID</b>	<b>UC16</b>
<b>NAME</b>	Sposta POI in itinerario
<b>SUMMARY</b>	L'utente seleziona un POI e lo sposta in un altro itinerario
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca l'apposito bottone per lo spostamento di un POI
<b>PRECONDITION</b>	L'utente ha effettuato il login, ha creato almeno un itinerario e si trova sulla mappa o nella schermata di riepilogo di un itinerario
<b>POSTCONDITION</b>	Il POI spostato non è più presente nell'itinerario corrente ed è stato aggiunto all'itinerario scelto
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema mostra all'utente la lista di tutti gli itinerari creati</li><li>2. L'utente sceglie l'itinerario nel quale spostare il POI scelto</li><li>3. L'utente conferma la scelta di voler spostare il POI</li><li>4. Il sistema aggiunge all'itinerario scelto il POI da spostare</li><li>5. Il sistema rimuove dall'itinerario corrente il POI scelto</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

**UC17: VISUALIZZA INFO POI**

<b>ID</b>	<b>UC17</b>
<b>NAME</b>	Visualizza info POI
<b>SUMMARY</b>	Vengono mostrati all'utente le informazioni riguardanti un POI, quali nome (se esiste), indirizzo e orari di apertura
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente seleziona un itinerario da visualizzare
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Il sistema mostra un riepilogo delle informazioni di un POI
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. L'utente si sposta sulla mappa</li><li>2. L'utente clicca su un POI già aggiunto all'itinerario</li><li>3. Il sistema mostra le informazioni relative al POI scelto, tra le quali:<ul style="list-style-type: none"><li>• nome</li><li>• indirizzo</li><li>• orari di apertura</li></ul></li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<p><i>Visualizzazione delle informazioni sulla schermata di riepilogo dell'itinerario</i></p> <p>1a1. L'utente clicca sul bottone per l'apertura della schermata di riepilogo di un itinerario</p> <p>1a2. Il sistema mostra la lista dei POI aggiunti all'itinerario</p> <p>1a3. L'utente sceglie un POI dalla lista</p> <p>1a4. Il sistema mostra le informazioni relative al POI scelto</p> <p><i>Il sistema non dispone delle informazioni di un POI</i></p> <p>3a1. Il sistema invia a Map API le coordinate del POI scelto</p> <p>3a2. Map API invia al sistema le informazioni del POI</p> <p>3a3. Il sistema mostra le informazioni relative al POI scelto</p>

**UC18: VISUALIZZA ITINERARI DA EFFETTUARE**

<b>ID</b>	<b>UC18</b>
<b>NAME</b>	Visualizza itinerari da effettuare
<b>SUMMARY</b>	Il sistema mostra la lista degli itinerari che non hanno ancora una data di completamento
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per la visualizzazione degli itinerari da effettuare
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Il sistema mostra all'utente la lista degli itinerari non ancora completati
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"> <li>1. Il sistema filtra gli itinerari creati dall'utente sulla base della data di completamento, selezionando solo quelli che ancora non hanno una data associata</li> <li>2. Il sistema mostra all'utente la lista degli itinerari filtrati</li> </ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

**UC19: VISUALIZZA STORICO ITINERARI**

<b>ID</b>	<b>UC19</b>
<b>NAME</b>	Visualizza storico itinerari
<b>SUMMARY</b>	Il sistema mostra la lista degli itinerari la cui data di completamento è precedente alla data corrente
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per la visualizzazione dello storico degli itinerari
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Il sistema mostra all'utente la lista degli itinerari passati
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"> <li>1. Il sistema filtra gli itinerari creati dall'utente sulla base della data di completamento, selezionando solo quelli che hanno una data precedente alla data corrente</li> <li>2. Il sistema mostra all'utente la lista degli itinerari filtrati</li> </ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

## 3.2 Component diagram

I casi d'uso appena descritti portano alla creazione di due nuovi componenti, uno lato app e uno lato backend. Il nuovo component diagram è presentato in Figura 17.

La struttura interna del componente `itinerary-service` è descritta in Figura 18 ed è pressoché identica alla struttura di `user-service`.

Il componente aggiunto all'app si occupa di consumare la nuova API esposta dal backend e l'API di Geocoding offerta da Photon, un servizio basato su OpenStreetMap.

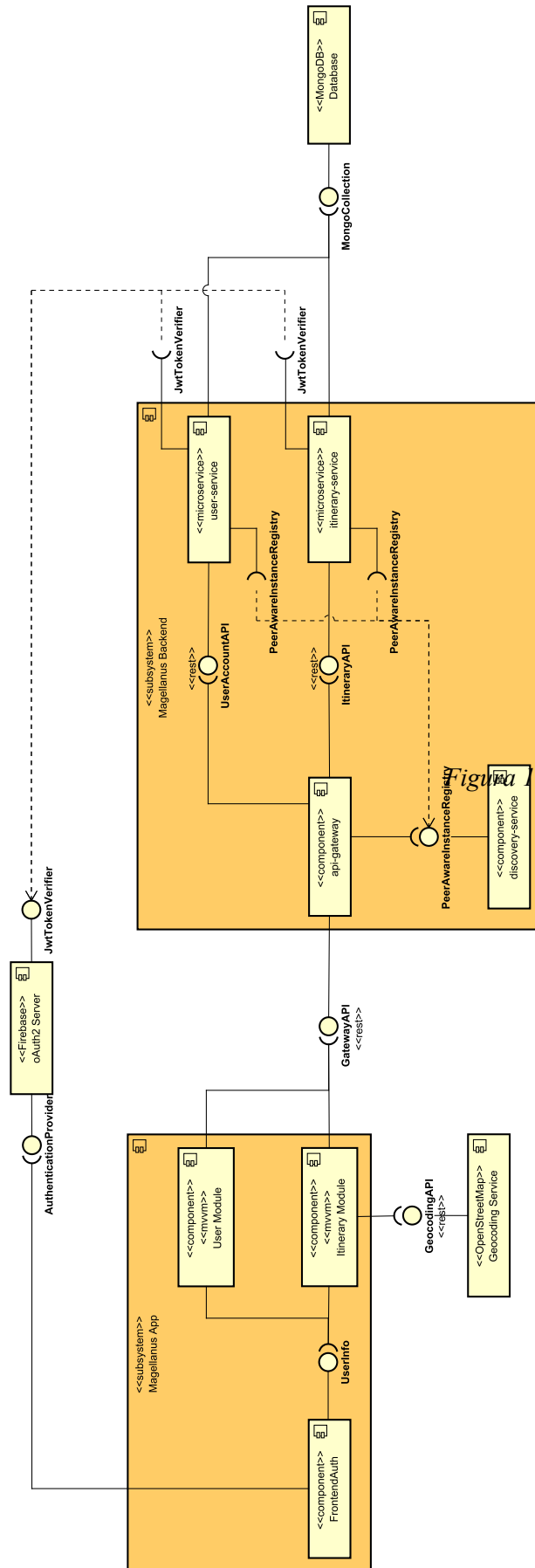


Figura 17: Iterazione 2 - Component diagram



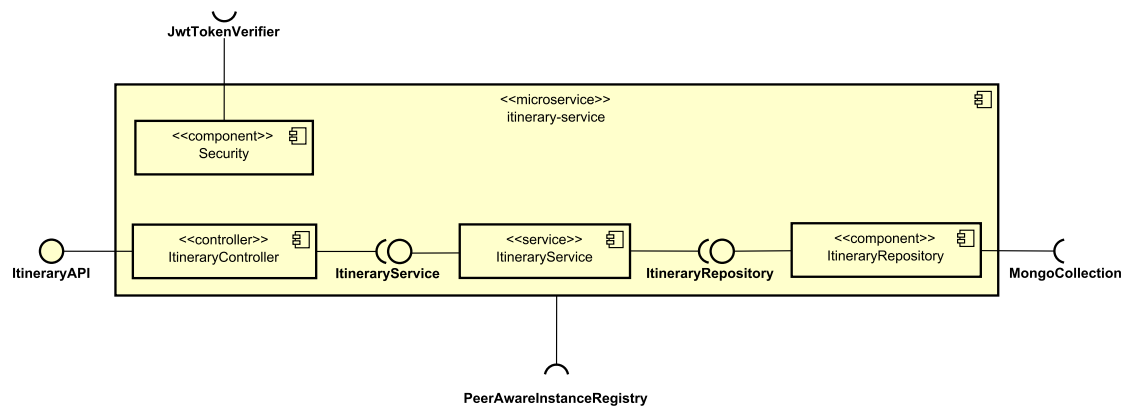


Figura 18: Iterazione 2 - Struttura interna itinerary-service

### 3.3 Class diagrams

Le nuove interfacce derivate dai casi d'uso scelti nell'iterazione 2 sono descritte in Figura 19. Si è deciso di riportare nuovamente la `UserAccountAPI` per mostrare che il gateway continua a esporre anche lo `user-service`.

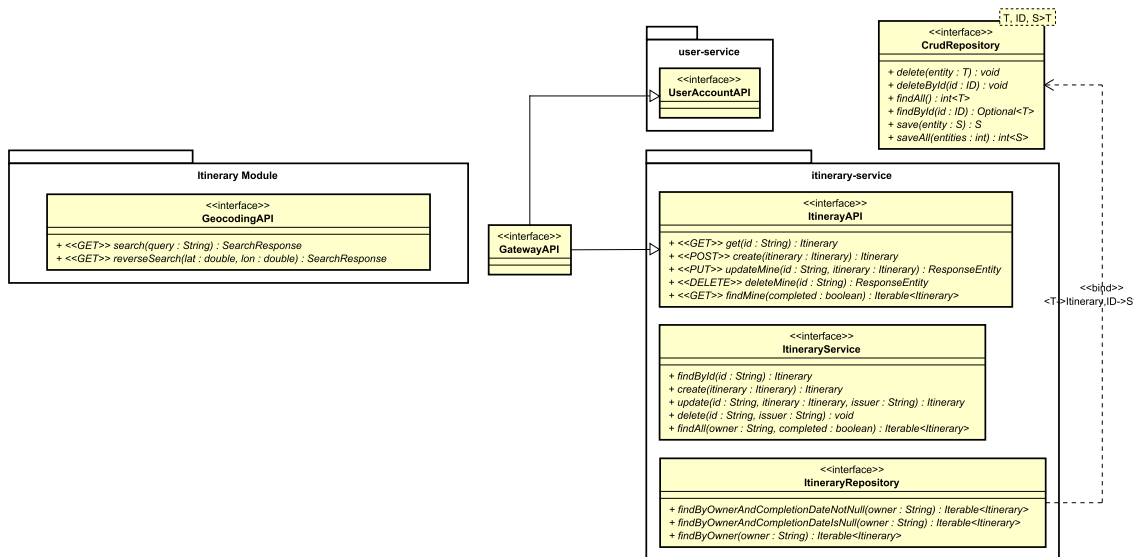


Figura 19: Iterazione 2 - Interfacce

In Figura 20 sono presentati i data types introdotti con i nuovi casi d’uso. Rappresentano entrambi il concetto di itinerario ma con grado di dettaglio diverso: si è deciso di non persistere nel database tutte le informazioni riguardo un POI ma solamente le coordinate. Questa scelta è stata presa in conseguenza al fatto che i dati mancanti di un POI possono essere recuperati rivolgendosi alla Geocoding API.

I nuovi transfer objects (Figura 21) mostrano i dati scambiati tra backend e app e anche la risposta fornita dall’API di Geocoding.

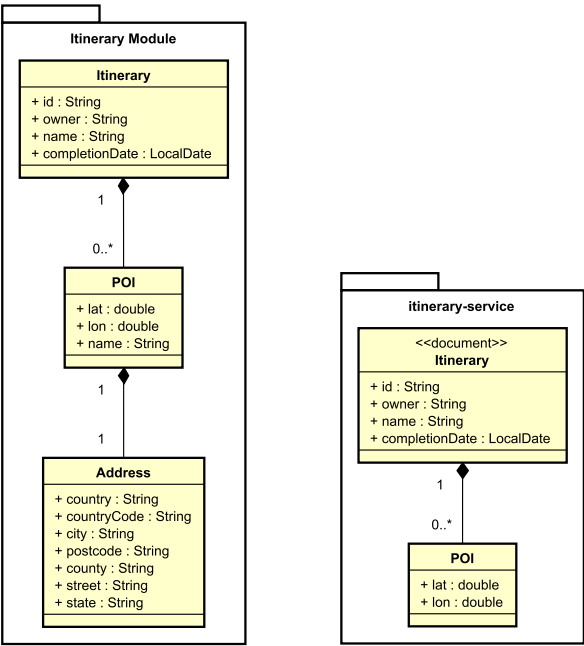


Figura 20: Iterazione 2 - Entità

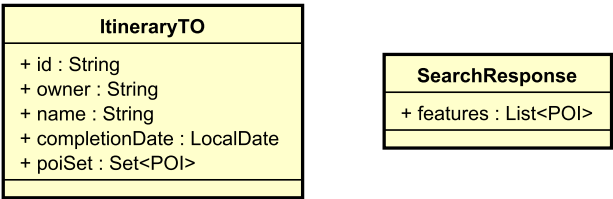


Figura 21: Iterazione 2 - Transfer objects

### 3.4 Deployment diagram

Il nuovo deployment diagram del sistema (Figura 22) è ottenuto dal precedente aggiungendo un nuovo container per il microservizio appena implementato e il server di OpenStreetMap che si occupa di fare Geocoding.

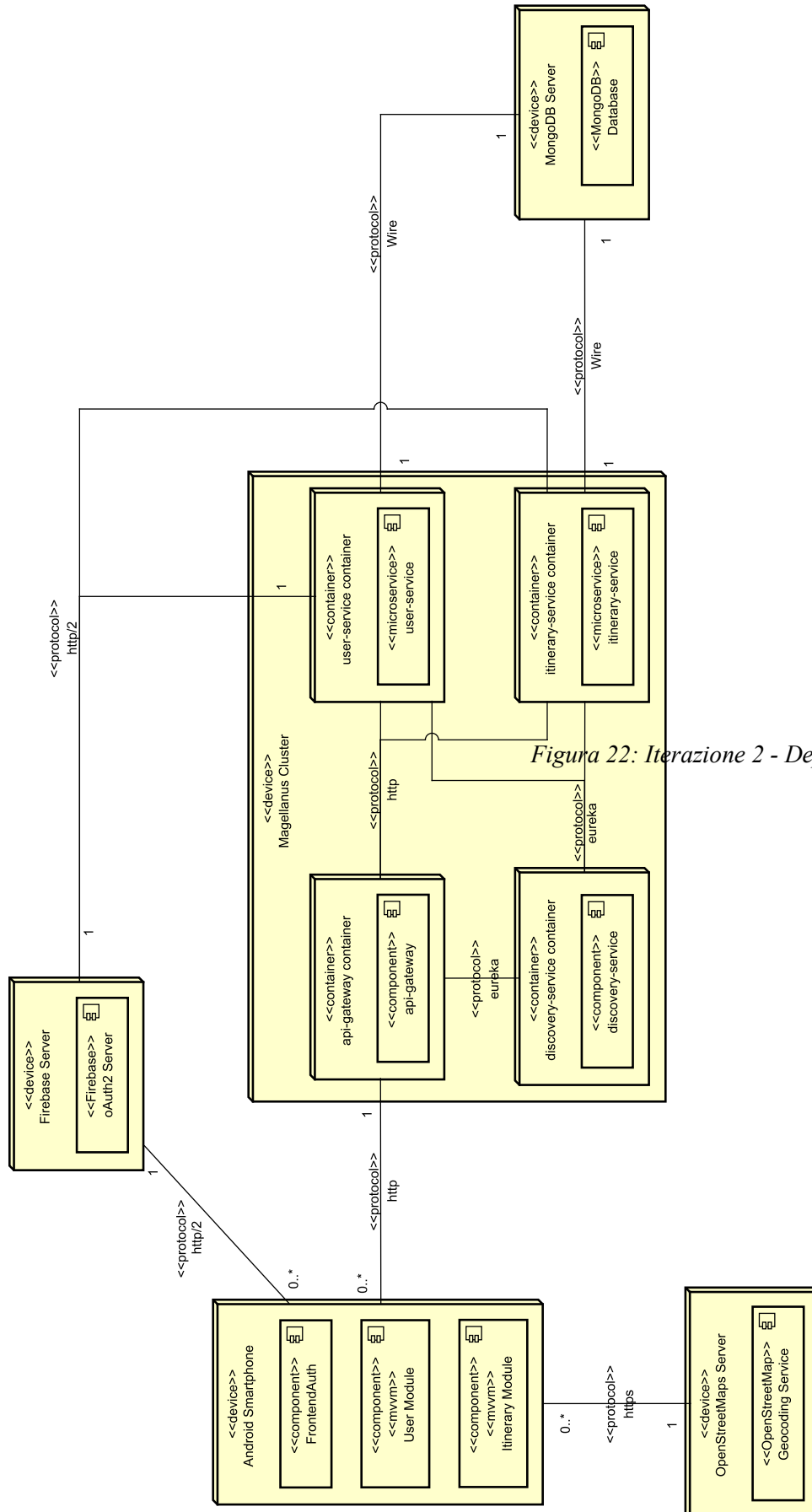


Figura 22: Iterazione 2 - Deployment diagram

## 3.5 Analisi

### 3.5.1 Analisi statica

L'analisi condotta con STAN non mostra particolari anomalie e quindi il codice è di qualità soddisfacente per procedere con l'iterazione successiva.

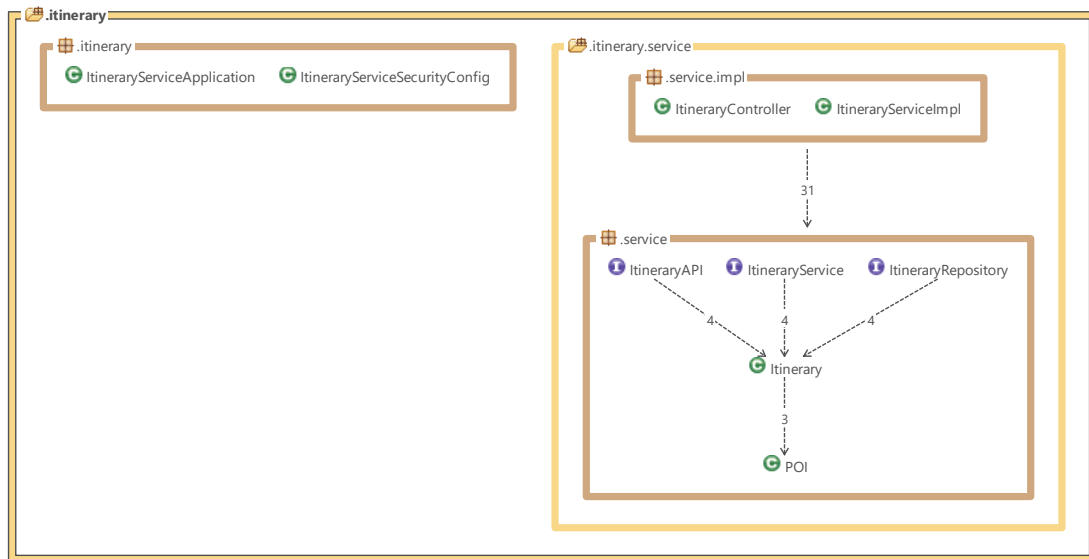


Figura 23: Iterazione 2 - Composition view itinerary-service

### 3.5.2 Analisi dinamica

I test sono stati condotti sulla classe `ItineraryServiceImpl` e sul microservizio nella sua interezza. Tutti i test sono stati superati e il coverage è del 95.9%. Si può affermare con elevata fiducia che il microservizio è corretto.

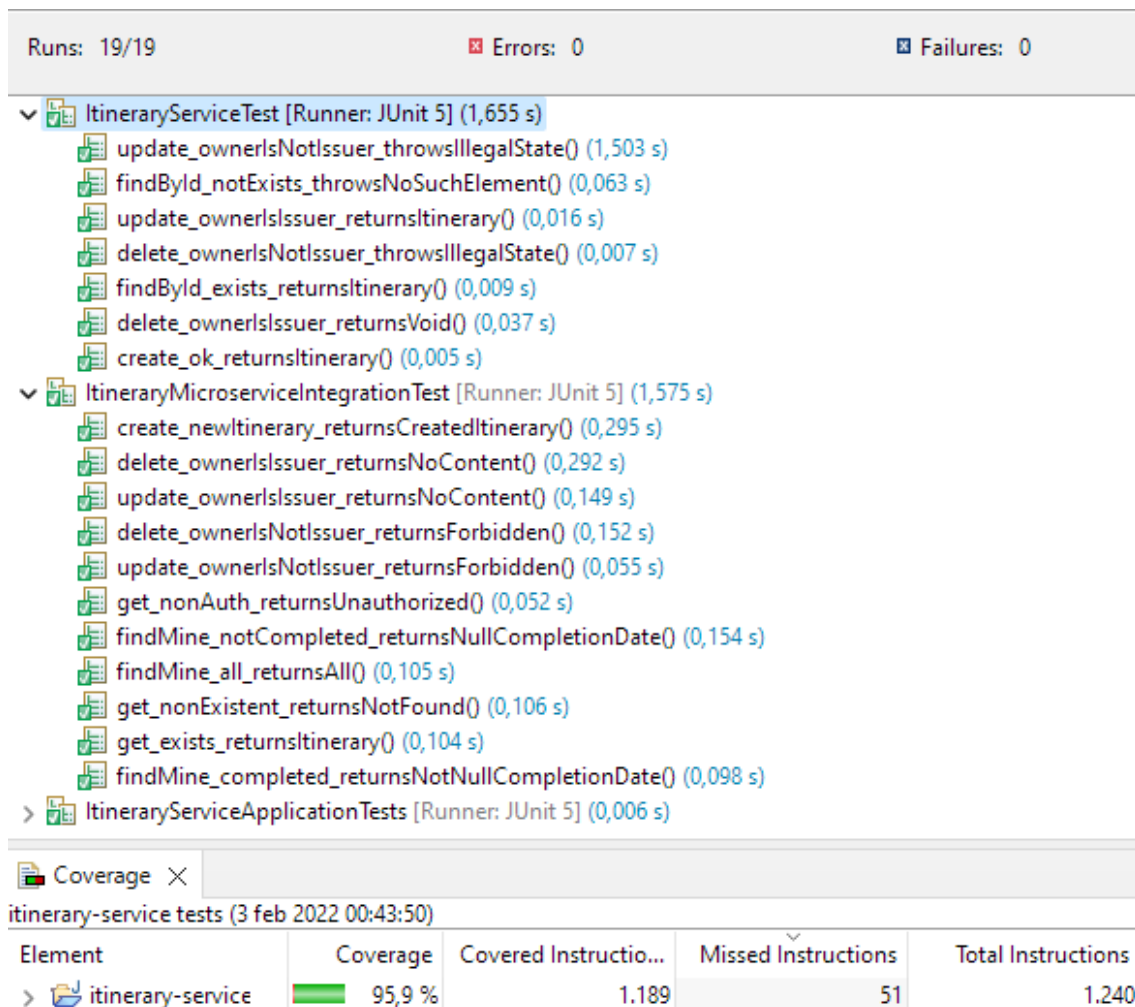


Figura 24: Iterazione 2 - Risultati test

## 3.6 Documentazione API

La documentazione dell'API è stata generata da Springdoc e segue la specifica di OpenAPI 3. È disponibile sia in formato json, sia yaml, sia grafico (Figura 25).

The screenshot displays the Swagger UI for the 'Itinerary API' (v1.0, OAS3). The top navigation bar includes the Swagger logo and a dropdown menu for 'Select a definition' set to 'Itinerary API'. Below the header, the API title 'Itinerary API' is shown with version and specification details. A 'Servers' section lists the base URL 'http://192.168.1.25:8082 - Generated server url'. An 'Authorize' button is present in the top right.

The main content area is titled 'itinerary-controller' and features a list of endpoints. The selected endpoint is 'PUT /me/{id}'. Its details include:

- Parameters:** A table with one parameter: 'id' (string, path, required).
- Request body:** A dropdown menu set to 'application/json'.
- Example Value:** A JSON object: 

```
{  "id": "string",  "owner": "string",  "name": "string",  "completionDate": "2022-02-02",  "poiSet": [    {      "lat": 0,      "lon": 0,      "inRoute": true    }  ]}
```
- Responses:** A table with one response: '200 OK' with no links.

Below the endpoint details, there are links to other endpoints: 'DELETE /me/{id}', 'POST /', 'GET /{id}', and 'GET /me'. At the bottom, there are links to the 'Schemas' section, specifically 'Itinerary' and 'POI'.

Figura 25: Iterazione 2 - Documentazione API

## 4 Iterazione 3

Nella terza e ultima iterazione si è scelto di implementare i casi d'uso, e quindi il servizio, relativi alla generazione dei cammini (Tabella 4).

### 4.1 Casi d'uso - Fully Dressed Description

#### UC20: SELEZIONA POI DA INCLUDERE

<b>Id</b>	<b>UC20</b>
<b>NAME</b>	Seleziona POI da includere
<b>SUMMARY</b>	L'utente seleziona i POI che vuole includere nel cammino da generare
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente clicca l'apposito bottone per la creazione di un cammino
<b>PRECONDITION</b>	L'utente ha effettuato il login e sta visualizzando un itinerario nell'apposita schermata
<b>POSTCONDITION</b>	Viene creata una lista di POI da cui generare un cammino
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema mostra all'utente la lista di tutti i POI salvati nell'itinerario</li><li>2. L'utente seleziona i POI che desidera facciano parte del cammino da generare</li><li>3. L'utente conferma i POI selezionati</li><li>4. Il sistema crea una lista contenente tutti e soli i POI selezionati dall'utente</li><li>5. Il sistema invia a Map API le coordinate dei POI selezionati</li><li>6. Map API invia la matrice delle distanze tra i POI al sistema</li><li>7. Il sistema setta le distanze tra ogni POI e il successivo</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	<i>Sono stati selezionati meno di tre POI</i> 4a1. Il sistema notifica all'utente la necessità di selezionare almeno tre POI per poter continuare

## UC21: GENERA PERCORSO MANUALMENTE

<b>ID</b>	<b>UC21</b>
<b>NAME</b>	Genera percorso manualmente
<b>SUMMARY</b>	L'utente manipola la lista di POI selezionati al fine di generare un percorso di suo gradimento
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente clicca l'apposito bottone per la creazione di un cammino
<b>PRECONDITION</b>	L'utente ha selezionato i POI da includere nel cammino
<b>POSTCONDITION</b>	Il cammino generato è presente nel database
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema mostra all'utente la lista dei POI inclusi nel percorso</li><li>2. L'utente cambia l'ordine dei POI nella lista</li><li>3. Il sistema invia a Map API le coordinate dei POI la cui posizione originaria è cambiata</li><li>4. Map API invia la matrice delle distanze al sistema</li><li>5. Il sistema aggiorna le distanze tra i POI</li><li>6. L'utente conferma di voler salvare il cammino generatosi</li><li>7. Il sistema salva il cammino</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

## UC22: GENERA PERCORSO AUTOMATICAMENTE

<b>ID</b>	<b>UC22</b>
<b>NAME</b>	Genera percorso automaticamente
<b>SUMMARY</b>	Il sistema genera un cammino ottimo su richiesta dell'utente
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente clicca il pulsante per la generazione automatica di un cammino
<b>PRECONDITION</b>	L'utente ha selezionato i POI da includere nel cammino
<b>POSTCONDITION</b>	Il sistema salva il nuovo cammino e lo mostra all'utente
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema invia a Map API le coordinate dei POI selezionati</li><li>2. Map API restituisce al sistema la matrice di distanze dei POI</li><li>3. Il sistema calcola un cammino ottimo sulla base della matrice delle distanze (o su una loro manipolazione) prendendo come punto iniziale il POI più vicino all'attuale posizione dell'utente</li><li>4. Il sistema salva il cammino generato</li><li>5. Il sistema mostra all'utente il nuovo cammino</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	



**UC23: MODIFICA PERCORSO**

<b>ID</b>	<b>UC23</b>
<b>NAME</b>	Modifica percorso
<b>SUMMARY</b>	L'utente modifica un cammino precedentemente generato
<b>ACTORS</b>	Utente, Map API
<b>TRIGGER</b>	L'utente si trova nella schermata di visualizzazione di un percorso
<b>PRECONDITION</b>	L'utente ha effettuato il login e ha già generato un cammino
<b>POSTCONDITION</b>	Il sistema salva il cammino modificato
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"> <li>1. L'utente cambia l'ordine dei POI nel cammino o li rimuove</li> <li>2. Il sistema invia a Map API le coordinate dei POI interessati dalla modifica del percorso</li> <li>3. Map API invia la matrice delle distanze al sistema</li> <li>4. Il sistema aggiorna le distanze tra i POI</li> <li>5. L'utente conferma di voler salvare il cammino aggiornato</li> <li>6. Il sistema aggiorna il cammino presente nel database</li> </ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

**UC24: VISUALIZZA DETTAGLI PERCORSO**

<b>ID</b>	<b>UC24</b>
<b>NAME</b>	Visualizza dettagli percorso
<b>SUMMARY</b>	Il sistema mostra all'utente la lista dei POI inclusi nel cammino, nell'ordine in cui sono stati salvati
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca sull'apposito bottone per la visualizzazione del percorso relativo a un itinerario
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	Il sistema mostra all'utente i dettagli del cammino
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"> <li>1. Il sistema recupera i dati del percorso relativo all'itinerario selezionato dall'utente</li> <li>2. Il sistema mostra la lista dei POI, nell'ordine definito dal percorso, e la distanza tra ogni POI e il successivo</li> </ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

**UC25: EFFETTUA PERCORSO**

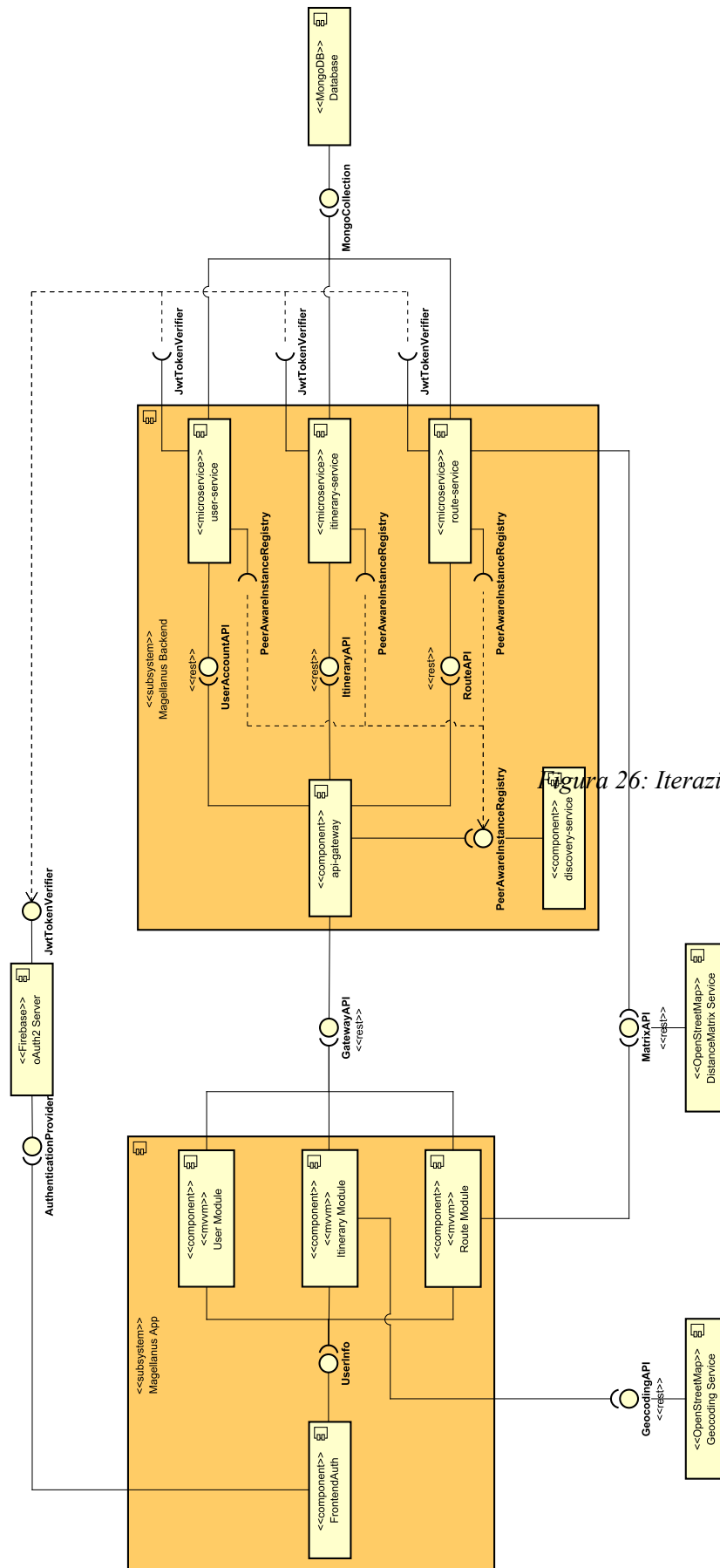
<b>ID</b>	<b>UC25</b>
<b>NAME</b>	Effettua percorso
<b>SUMMARY</b>	L'utente percorre il cammino, seguendo l'ordine dei POI
<b>ACTORS</b>	Utente
<b>TRIGGER</b>	L'utente clicca l'apposito pulsante per l'avvio e realizzazione del percorso
<b>PRECONDITION</b>	L'utente ha effettuato il login
<b>POSTCONDITION</b>	All'itinerario relativo il percorso appena terminato viene assegnata come data di completamento la data odierna
<b>BASE SEQUENCE</b>	<ol style="list-style-type: none"><li>1. Il sistema mostra all'utente il POI da raggiungere</li><li>2. L'utente, una volta raggiunto il POI, lo segnala al sistema</li><li>3. Il sistema mostra il POI successivo</li><li>4. Il sistema, terminati i POI, aggiorna l'itinerario corrispondente, assegnandogli come data di completamento la data odierna</li></ol>
<b>EXCEPTION/ALTERNATIVE SEQUENCE</b>	

## 4.2 Component diagram

I casi d'uso appena descritti portano alla creazione di due nuovi componenti, uno lato app e uno lato backend. Il nuovo component diagram è presentato in Figura 26.

La struttura interna del componente `route-service` è descritta in Figura 27. Oltre alle solite componenti (`repository`, `service`, `controller`), questo microservizio ha un ulteriore componente che si occupa di calcolare la soluzione del TSP.

Il componente aggiunto all'app si occupa di consumare la nuova API esposta dal backend. Entrambi i componenti inoltre consumano un'API di calcolo della matrice delle distanze offerta da Open Source Routing Machine (OSRM), un servizio basato su OpenStreetMap.



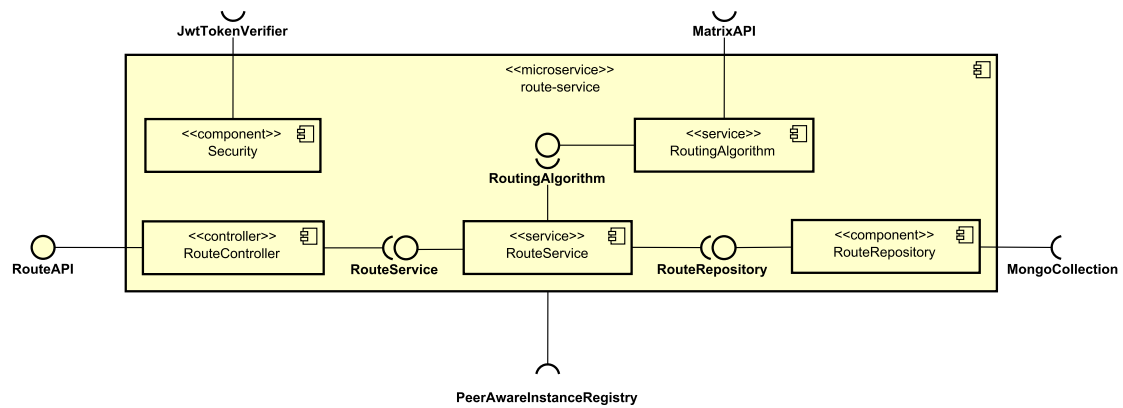


Figura 27: Iterazione 3 - Struttura interna route-service

## 4.3 Class diagrams

Le interfacce derivate dai casi d'uso scelti nell'iterazione 3 sono descritte in Figura 28.

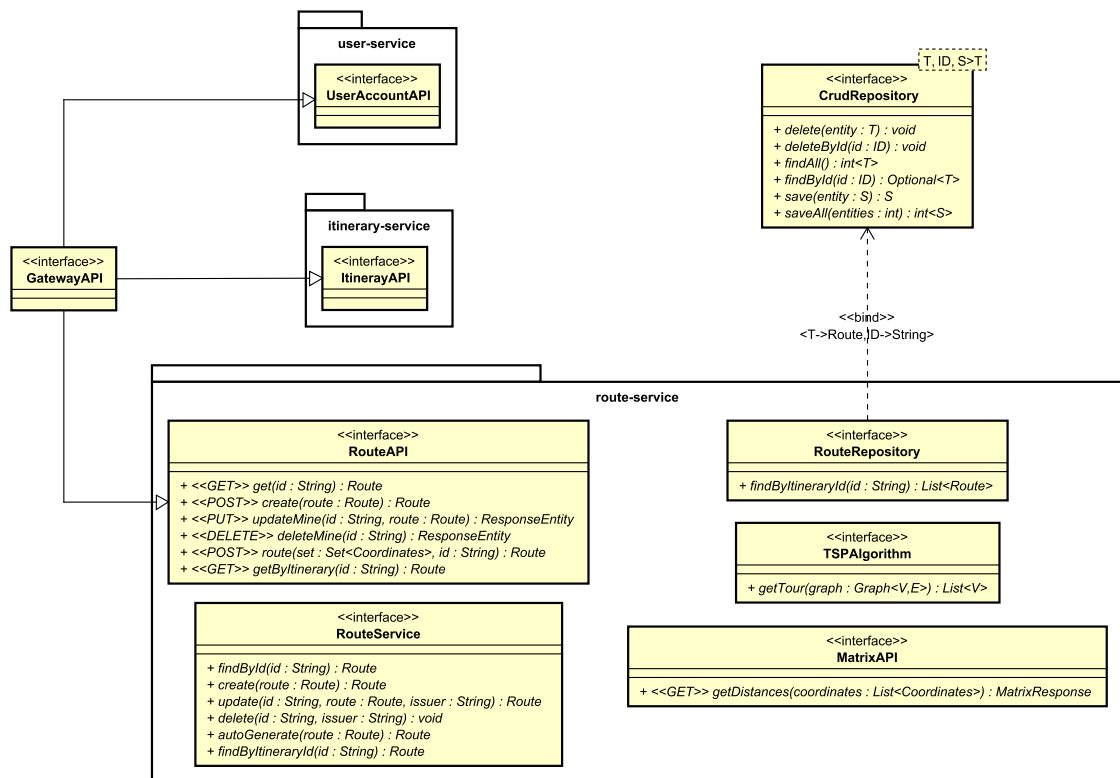


Figura 28: Iterazione 3 - Interfacce

In Figura 29 sono mostrati i data types introdotti con i nuovi casi d’uso. Rappresentano entrambi il concetto di percorso e in questo caso le entità sono identiche.

I nuovi transfer objects (Figura 30) mostrano i dati scambiati tra backend e app e la risposta fornita dall’API di matrice delle distanze con cui si interfacciano entrambe le nuove componenti.

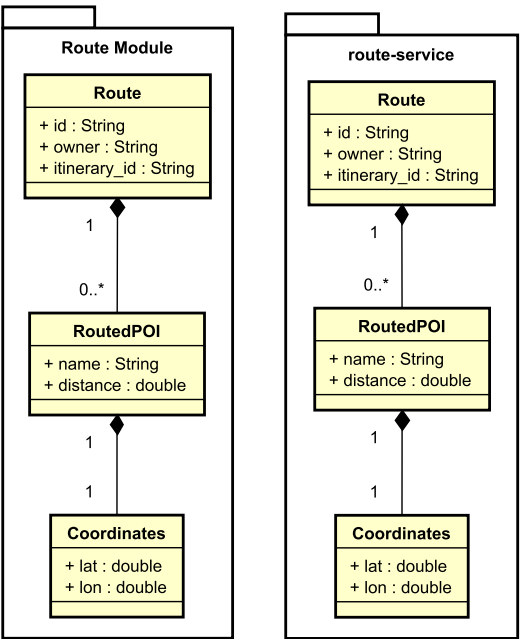


Figura 29: Iterazione 3 - Entità

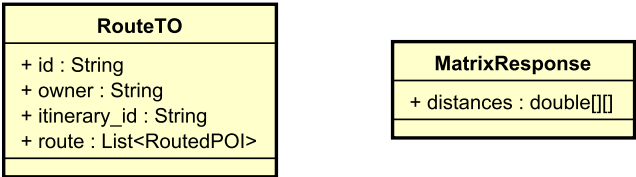


Figura 30: Iterazione 3 - Transfer objects

## 4.4 Deployment diagram

Il nuovo deployment diagram del sistema (Figura 31) è ottenuto dal precedente aggiungendo un nuovo container per il microservizio appena implementato e la nuova componente di OpenStreetMap che si occupa di calcolare la matrice delle distanze.

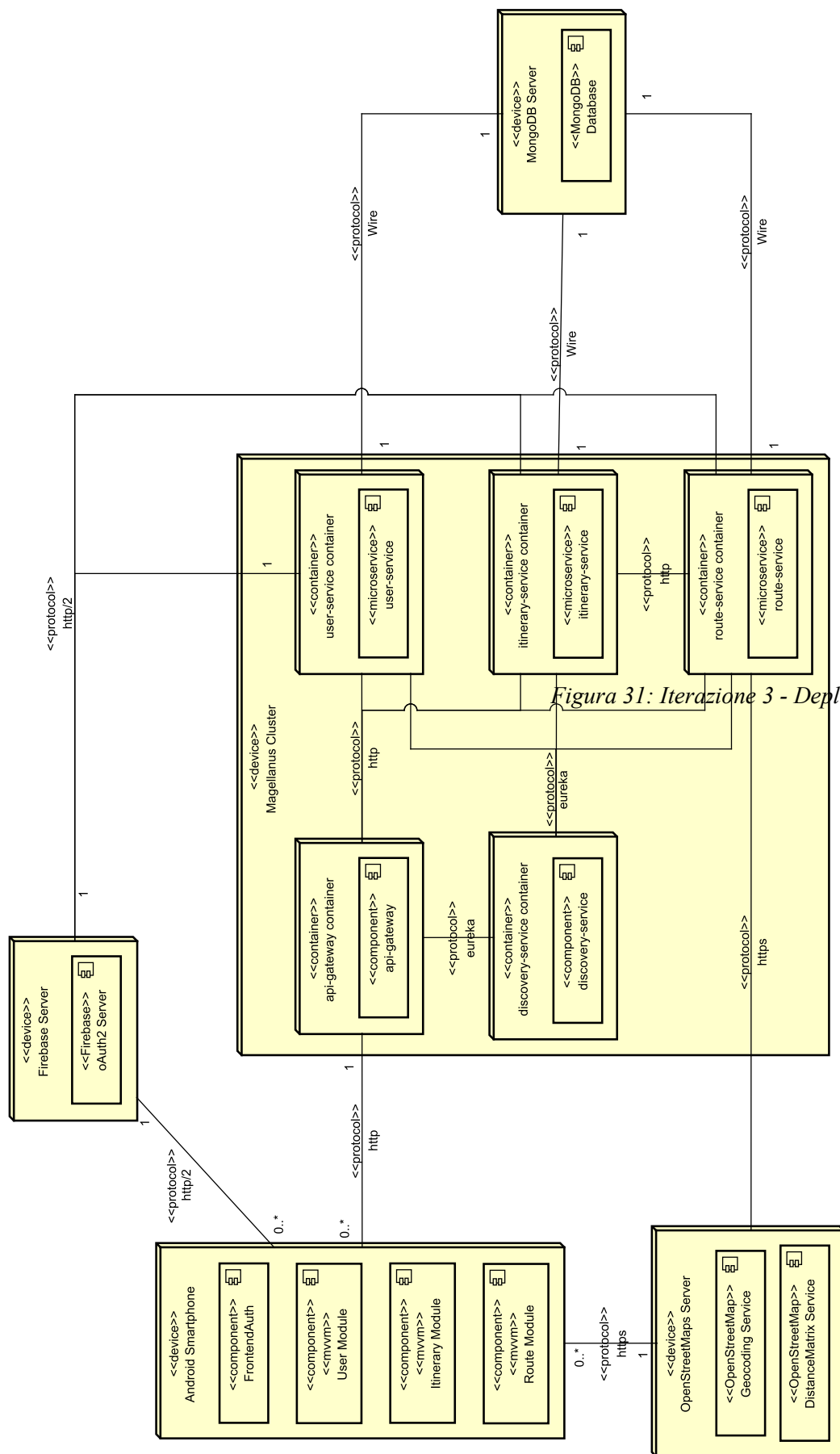


Figura 31: Iterazione 3 - Deployment diagram

## 4.5 Traveling Salesman Problem: definizione e algoritmi

Il problema del commesso viaggiatore (TSP) può essere posto nel seguente modo: “Data una lista di città e la distanza tra ogni possibile coppia di città, qual è il percorso più corto che visita ogni città esattamente una volta e termina nella città di partenza?”

Rispondere a questa domanda significa determinare un ciclo Hamiltoniano che sia anche di costo minimo. Il problema di determinare l’esistenza di un circuito Hamiltoniano è NP-hard mentre il TSP è anche un problema di ottimizzazione, per cui è considerato NP-completo.

Esistono diverse versioni del TSP, che si differenziano per il grafo su cui si sta cercando la soluzione. La formulazione del TSP in caso di grafo non orientato è la seguente:

dato un grafo non orientato  $G = (V, E)$  con pesi  $c_{i,j} = c_{j,i}$  associati agli archi

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} c_{i,j} x_{i,j} \\ \text{s.a.} \quad & \sum_{(j,i) \in S(i)} x_{i,j} = 2, \quad i \in \mathbb{N} \\ & \sum_{i \in Q} \sum_{j \in N \setminus Q} x_{i,j} \geq 1 \quad \forall Q \subset V, |Q| \geq 1 \\ & x_{i,j} \in \{0,1\} \end{aligned}$$

Nel nostro caso il TSP viene detto *metrico* in quanto risultano rispettate le seguenti condizioni:

- il grafo costruito a partire dai POI è completo, ovvero si può andare da ogni nodo verso qualunque altro nodo;
- il grafo è non orientato: la strada (l’arco) che collega due POI è percorribile in entrambe le direzioni;
- il grafo rispetta la disuguaglianza triangolare: dati tre nodi qualsiasi  $i, j$  e  $k$  vale

$$c_{i,k} + c_{k,j} \geq c_{i,j}$$

dove  $c_{i,j}$  rappresenta il costo di attraversamento dell’arco  $(i, j)$ , nel nostro caso il costo può essere rappresentato dalla distanza o dal tempo di percorrenza di un tratto di percorso.



Esistono molti algoritmi euristici per il TSP, se il problema è metrico si possono però applicare algoritmi specifici, più efficienti e migliori dal punto di vista della qualità della soluzione. Infatti, i due algoritmi che si è scelto di implementare danno una garanzia sull'upper bound dell'errore commesso, cioè quanto dista la soluzione trovata dall'ottimo.

#### 4.5.1 2-Approximation Metric TSP Algorithm

Il primo algoritmo implementato è un algoritmo approssimato per la risoluzione di un TSP metrico. I passi dell'algoritmo sono i seguenti:

- 1 Create a minimum spanning tree  $\mathcal{T}$  of  $\mathcal{G}$
- 2 Duplicate each edge in  $\mathcal{T}$  to obtain a Eulerian graph  $E$
- 3 Form a Eulerian circuit in  $E$
- 4 Make the circuit found in the previous step into a Hamiltonian circuit by skipping repeated vertices (shortcutting)

Lo pseudocodice è molto sintetico perché si basa sugli algoritmi di ricerca di un albero di copertura di costo minimo e di ricerca di un ciclo Euleriano. I passi da 2 a 4 possono essere sintetizzati con una semplice visita in preordine dei nodi dell'albero  $\mathcal{T}$ , si può dimostrare infatti come una visita di questo tipo formi un ciclo Hamiltoniano.

Ricordiamo che il grafo è completo:  $m = |E| = |V|^2 = n^2$ . Per calcolare il MST si possono usare gli algoritmi di Kruskal o Prim, entrambi con complessità  $O(m \cdot \log n) = O(n^2 \cdot \log n)$ . Una visita DFS di un albero ha invece complessità  $O(n)$ . L'algoritmo ha quindi complessità  $O(n^2 \cdot \log n) + O(n) = O(n^2 \cdot \log n)$ .

Dal momento che il ciclo Hamiltoniano è costruito a partire dal MST, a cui sono stati duplicati gli archi, si può dimostrare che il costo della soluzione trovata sarà sempre minore del doppio del costo della soluzione ottima.

#### 4.5.2 Christofides Approximation Algorithm

L'algoritmo di Christofides è un miglioramento del precedente e si basa sul lemma della stretta di mano: l'insieme dei vertici con grado dispari nel MST ha cardinalità pari. Ciò significa che è possibile trovare un accoppiamento perfetto nel sottografo definito a partire dall'insieme dei vertici con grado dispari. L'unione dell'albero di copertura e dell'accoppiamento è Euleriano.

I passi dell'algoritmo sono i seguenti:

- 1 Create a minimum spanning tree  $\mathcal{T}$  of  $\mathcal{G}$
- 2 Let  $\mathcal{O}$  be the set of vertices with odd degree in  $\mathcal{T}$
- 3 Find a minimum-weight perfect matching  $\mathcal{M}$  in the induced subgraph given by the vertices from  $\mathcal{O}$
- 4 Combine the edges of  $\mathcal{M}$  and  $\mathcal{T}$  to form a connected multigraph  $\mathcal{H}$  in which each vertex has even degree
- 5 Form a Eulerian circuit in  $\mathcal{H}$
- 6 Make the circuit found in the previous step into a Hamiltonian circuit by skipping repeated vertices (shortcutting)

La descrizione dell'algoritmo è, come nel caso precedente, estremamente semplice in quanto si tratta di applicazioni di altri algoritmi noti. Il costo di ogni passo è:

- 1 calcolo di un MST tramite Prim o Kruskal:  $O(m \cdot \log n) = O(n^2 \cdot \log n)$ ;
- 2 ricerca dei nodi con grado dispari (è un filtraggio):  $O(n)$ ;
- 3 ricerca di un accoppiamento di costo minimo con l'algoritmo ungherese o l'algoritmo di Edmonds:  $O(t^2 \cdot n) = O(n^2 \cdot n) = O(n^3)$ , dove  $t$  è il numero di archi presenti in  $\mathcal{T}$ ;
- 4 aggiunta degli archi di  $\mathcal{M}$  a  $\mathcal{T}$ :  $O(n/2) = O(n)$  in quanto il numero di archi di  $\mathcal{M}$  nel peggiore dei casi è pari alla metà del numero dei nodi da cui è costruito;
- 5 ricerca di un circuito Euleriano tramite Hierholzer:  $O(m) = O(n^2)$ ;
- 6 calcolo del circuito Hamiltoniano: bisogna attraversare tutti gli archi del circuito Euleriano,  $O(m) = O(n^2)$ .

Il costo dominante è dato dal passo 3, quindi il costo dell'algoritmo è  $O(n^3)$ . Il costo della soluzione trovata con l'algoritmo di Christofides è al massimo  $\frac{3}{2}$  il costo della soluzione ottima, questo è il miglior risultato raggiunto da un algoritmo approssimato che risolve il TSP.

## 4.6 Analisi

Per questa iterazione si è ovviamente deciso di testare anche la correttezza dei due algoritmi, oltre alla componente `@Service` e al microservizio.

### 4.6.1 Analisi statica

L'analisi condotta con STAN riporta un valore pari a -0.66 per la Distance del package relativo agli algoritmi. Questo valore è conseguenza del fatto che nel package ci siano due classi concrete e che l'*afferent coupling* sia pari a 1. Ciò nonostante, il codice è di qualità soddisfacente per terminare il progetto.

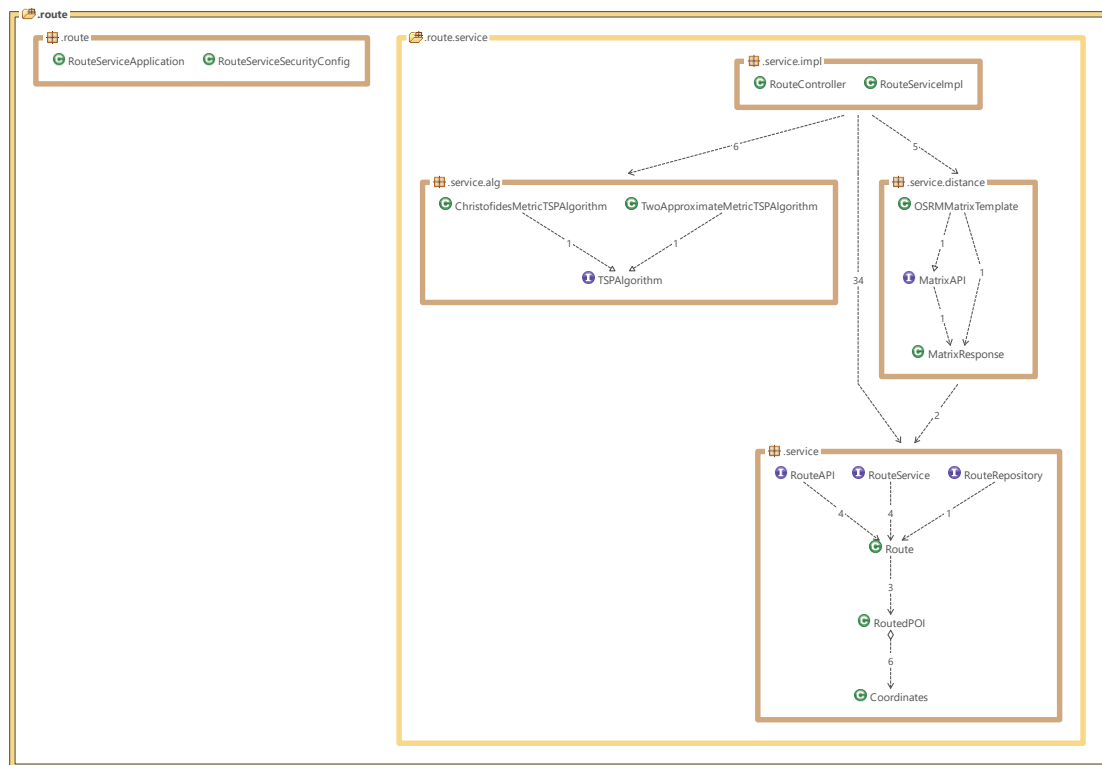


Figura 32: Iterazione 3 - Composition view itinerary-service

### 4.6.2 Analisi dinamica

I test sono stati condotti sulla classe `RouteServiceImpl`, sui due algoritmi e sul microservizio nella sua interezza. Tutti i test sono stati superati e il coverage è del 97.6%. Si può affermare con elevata fiducia che il microservizio è corretto.

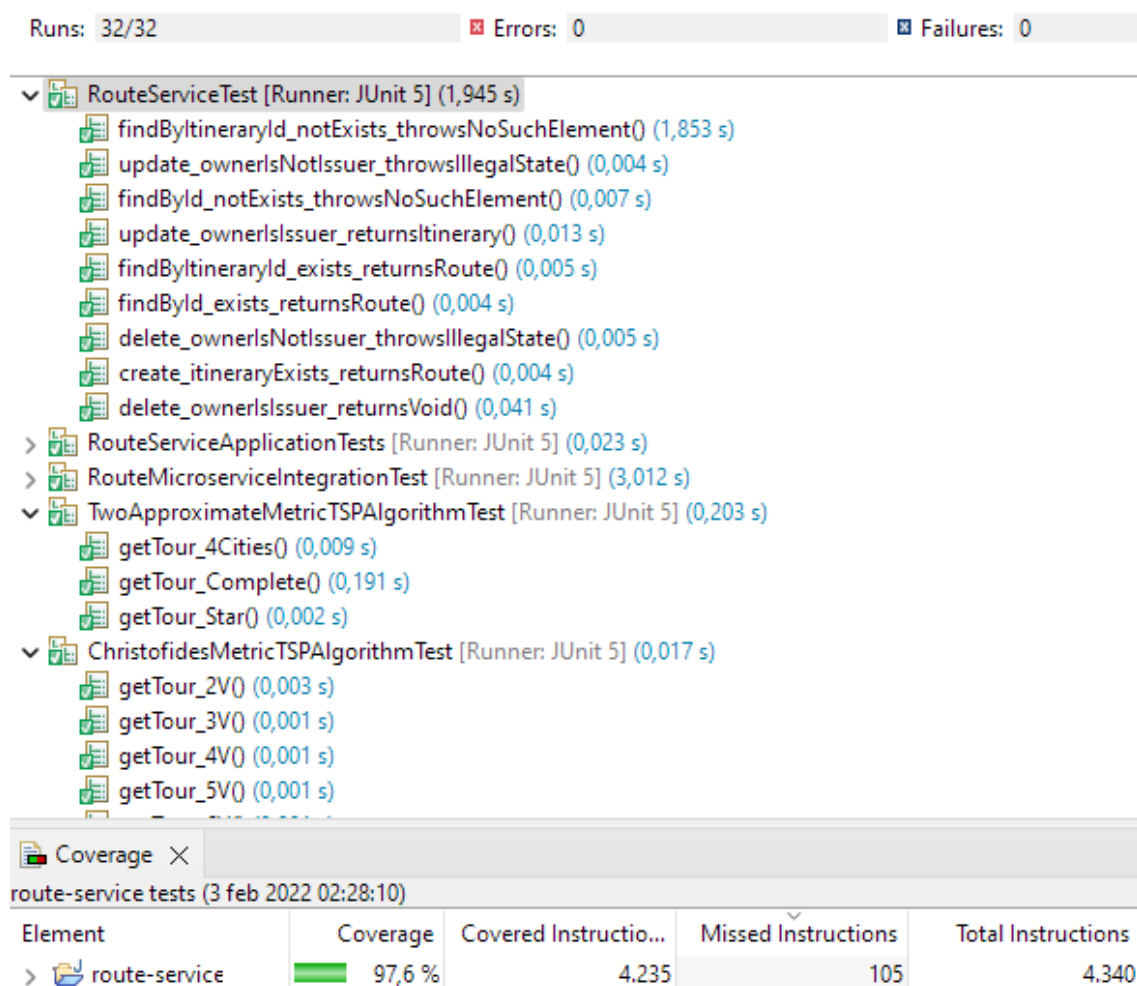



Figura 33: Iterazione 3 - Risultati test

## 4.7 Documentazione API

La documentazione dell'API è stata generata da Springdoc e segue la specifica di OpenAPI 3. È disponibile sia in formato json, sia yaml, sia grafico (Figura 34).


Swagger
powered by SMARTBEAR

Select a definition
Route API

# Route API

v1.0
OAS3

[/3api-docs/routes](#)

Documentation Route API v1.0

Servers

http://192.168.1.25:8083 - Generated server url

Authorize

## route-controller

PUT
/me/{id}

DELETE
/me/{id}

POST
/auto

Parameters

Try it out

No parameters

Request body required

application/json

Example Value
Schema

```

{
  "id": "string",
  "owner": "string",
  "itineraryid": "string",
  "route": {
    "coordinates": {
      "lat": 0,
      "lon": 0
    },
    "name": "string",
    "distance": 0
  }
}

```

Responses

Code	Description	Links
200	OK	No links

Media type

\*/

Controls Accept header

Example Value
Schema

```

{
  "id": "string",
  "owner": "string",
  "itineraryid": "string",
  "route": {
    "coordinates": {
      "lat": 0,
      "lon": 0
    },
    "name": "string",
    "distance": 0
  }
}

```

POST
/

GET
/{id}

GET
/itinerary/{id}

## Schemas

Coordinates >

Route >

RoutedPOI >

Figura 34: Iterazione 3 - Documentazione API

61

## 5 Conclusioni

Il codice, i diagrammi UML, i report di STAN, i risultati dei test e la documentazione sono disponibili nella repository all'indirizzo <https://github.com/fcarne/Magellanus.git>.

Alcuni aspetti del sistema sviluppato che possono essere migliorati per aumentarne la qualità sono:

- Annotare le classi controller del backend con gli appositi costrutti per poter produrre una documentazione delle API più estesa ed esauriente.
- Introdurre un *configuration server* nel backend in cui centralizzare i file di configurazione dei vari microservizi.
- Scomporre l'applicazione in veri e propri moduli, piuttosto che in semplici package, per facilitare i processi di building e la gestione delle dipendenze.
- Il codice dell'app segue il pattern Dependency Injection ma non fa uso di alcun injector; a tal fine, si può fare uso di librerie apposite come Dagger o Hilt.
- Lato applicazione non è stato implementato nessun meccanismo di persistenza dei dati, solo le chiamate alle API di Photon e OSRM sono sottoposte a caching vista l'invarianza dei dati che inviano; per ridurre il consumo dei dati si potrebbe introdurre un db sqlite locale, per esempio con Room. Le modifiche da apportare riguarderebbero solo le classi repository, grazie al pattern SSOT, per garantire la consistenza dei dati.

### 5.1 Guida installazione

Per installare l'applicazione su un dispositivo Android basta scaricare il file .apk presente nella release di GitHub e avviare l'installazione.

Per fare uso dei microservizi è necessario eseguire i file .jar corrispondenti, digitando in un terminale il comando: `java -jar <nome_servizio>.jar`

I microservizi `user-service`, `itinerary-service` e `route-service` saranno accessibili rispettivamente alle porte 8081, 8082 e 8083. Avviando anche il `discovery-service` e l'`api-gateway` tutte le API esposte saranno accessibili alla porta 8080.

Il servizio di discovery, in ascolto sulla porta 8761, mette a disposizione un endpoint all'indirizzo `localhost:8761/` per visualizzare i servizi attualmente registrati.

The screenshot displays the Spring Eureka web interface. At the top, there's a header with the Spring Eureka logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content is divided into several sections:

- System Status:** A table showing environment details (Environment: N/A, Data center: N/A) and system metrics (Current time: 2022-02-03T13:46:06 +0100, Uptime: 00:01, Lease expiration enabled: false, Renew threshold: 8, Renew (last min): 0).
- DS Replicas:** A section showing the local host as a replica.
- Instances currently registered with Eureka:** A table listing four services: GATEWAY-SERVICE, ITINERARY-SERVICE, ROUTE-SERVICE, and USER-SERVICE. Each service is shown with its AMIs, Availability Zones, and Status (UP).
- General Info:** A table showing system metrics like total-avail-memory (64mb), num-of-cpus (8), current-memory-usage (36mb (56%)), server-uptime (00:01), registered-replicas (http://localhost:8761/eureka/), unavailable-replicas (http://localhost:8761/eureka/), and available-replicas.
- Instance Info:** A table showing the instance's IP address (192.168.1.25) and status (UP).

*Figura 35: Schermata del discovery-service*

Infine, la documentazione è disponibile in formato grafico all'indirizzo `localhost:8080/swagger-ui.html` (Figura 16, Figura 25, Figura 34).

Se si dispone di Docker sul proprio sistema è possibile avviare il backend scaricando il file `docker-compose.yml` presente nella repository GitHub e, ponendosi nella stessa directory, avviare un terminale e digitare il comando: `docker-compose up -d`. In questo caso saranno esposti solo i servizi di discovery e di gateway.

## 5.2 Breve guida utente

Al primo accesso all'applicazione, verrà richiesto di effettuare il login (e di registrarsi nel caso non lo si avesse già fatto, Figura 36).

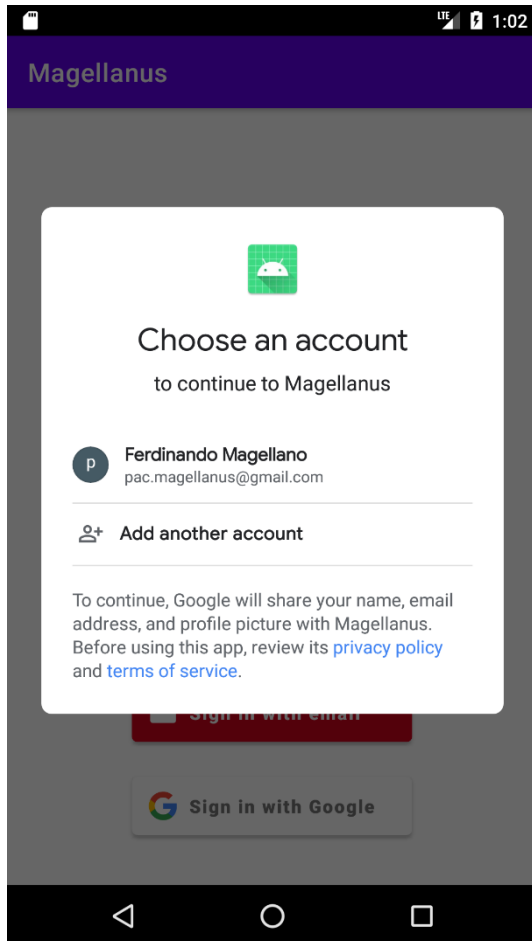


Figura 36: Screen 1 - Login



Figura 37: Screen 2 - Schermata principale

Se il login è andato a buon fine si aprirà la schermata principale, contenente una mappa nella quale è possibile cercare, salvare e rimuovere i POI da un itinerario (Figura 37). Cliccando sul menù in alto a sinistra si aprirà un Navigation Drawer dal quale è possibile accedere alla schermata delle impostazioni ed effettuare il logout (Figura 38). I 3 bottoni presenti nella schermata principale, dall'alto verso il basso, permettono di:

- aprire la lista dei POI salvati;
- aprire la lista di tutti gli itinerari;
- riposizionare la mappa.



Cliccando sul primo dei tre bottoni si aprirà la schermata in cui sono contenuti i dettagli dei POI salvati (Figura 39). Cliccando sul bottone presente in ogni voce della lista è possibile tornare alla mappa, centrata sul POI cliccato. Il bottone in basso a destra permette invece di generare un percorso, a partire dai POI che sono stati selezionati. Se questi sono più di tre si aprirà la schermata di generazione e gestione del percorso.

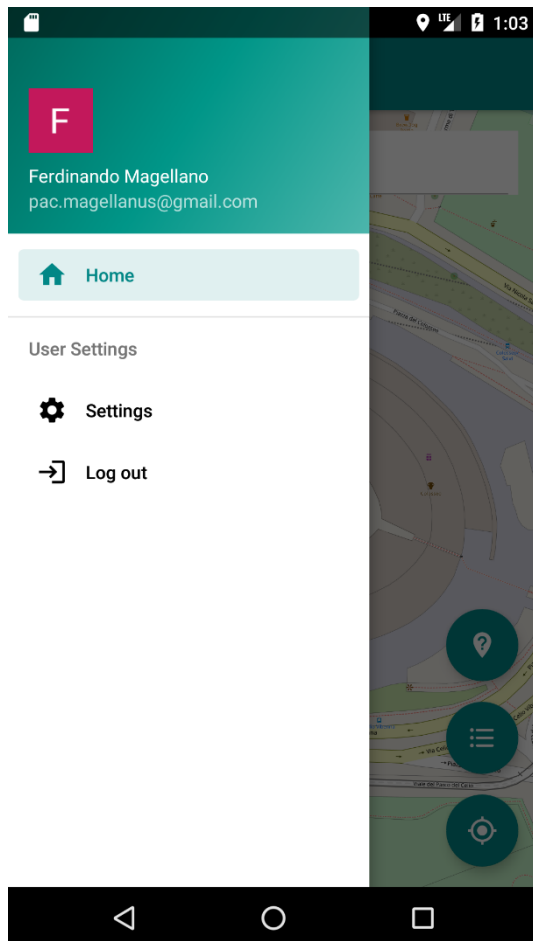


Figura 38: Screen 3 - Navigation drawer

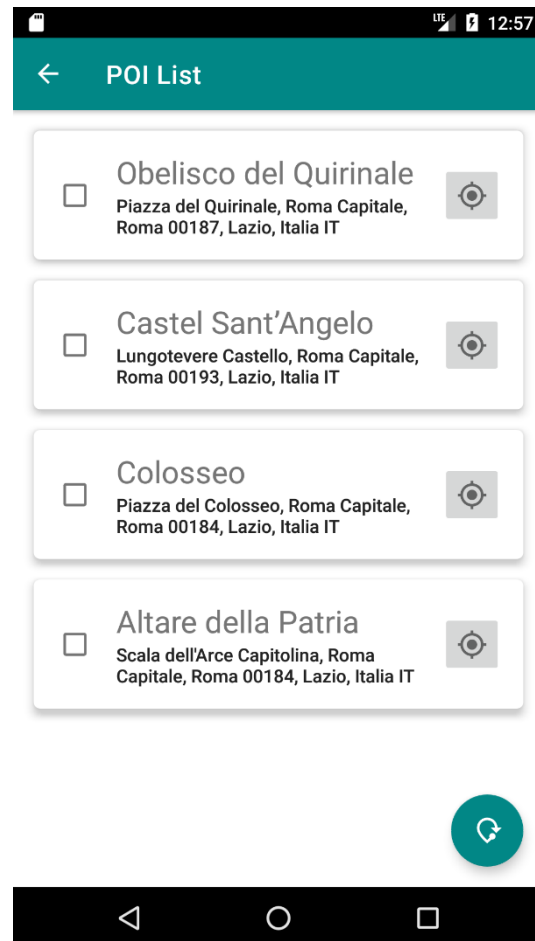
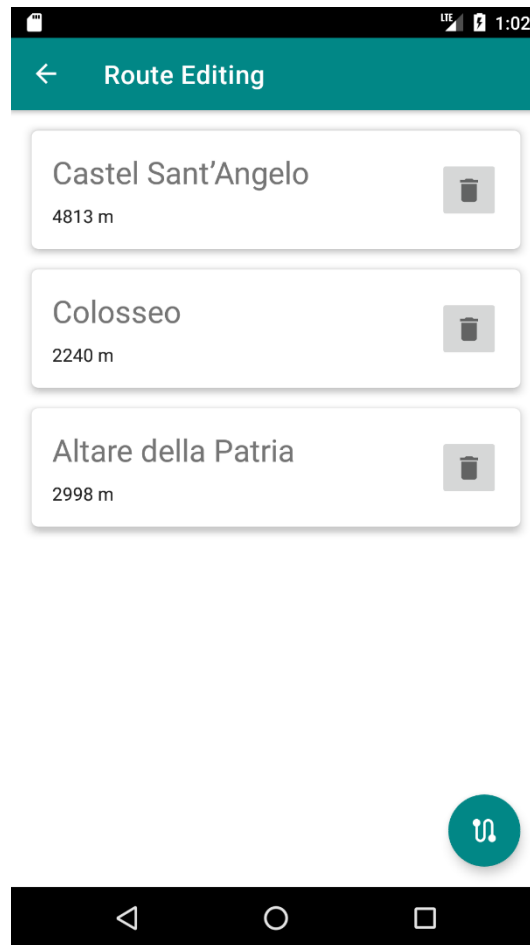


Figura 39: Screen 4 - Lista POI

In questa schermata è possibile modificare il percorso tenendo premuto l'elemento che si vuole spostare e riordinando i POI. Una volta lasciato, l'elemento comparirà alla posizione assegnata e le distanze verranno aggiornate di conseguenza. Il bottone in basso a destra permette invece di riordinare il percorso automaticamente.



*Figura 40: Screen 5 - Generazione percorsi*