

## Iterazione 2

### Descrizione use case

#### UC4 Prenotazione utente

- UC4.1: Ricerca parcheggio
- UC4.2: Consultazione disponibilità posti
- UC4.3: Effettuare prenotazione
- UC4.4: Ricevere conferma prenotazione
- UC4.5: Consultazione storico prenotazioni

#### UC5 Analitiche parcheggio

- UC5.1: Monitoraggio stato parcheggio
- UC5.2: Gestione della manutenzione parcheggio

##### UC4.1: Ricerca parcheggio

Breve descrizione: Utente cerca, nella sezione dell'app apposita, i parcheggi presenti nel database collegati agli operatori esistenti.

Attori coinvolti: Utente e Database

Trigger: La ricerca ha prodotto dei risultati

Postcondizione: Dopo la ricerca l'utente seleziona il parcheggio, tra quelli proposti, a lui più conveniente.

Procedimento: 1) Selezione dell'area di ricerca

- 2) Click sulla barra di ricerca
- 3) Digitazione query
- 4) Presa visione dei risultati
- 5) Selezione della scelta

##### UC4.2: Consultazione disponibilità posti

Breve descrizione: Dopo aver eseguito la ricerca dei parcheggi l'utente vede accanto ad ognuno la disponibilità dei posti totali

Attori coinvolti: Utente e Database

Trigger: Corretta visualizzazione dei dati dei parcheggi

Postcondizione: Dopo aver preso visione dei posti disponibili l'utente seleziona il parcheggio a lui più congeniale

Procedimento: 1) Utente cerca i parcheggi

- 2) Prende visione della disponibilità dei singoli luoghi

##### UC4.3: Effettuare prenotazione

Breve descrizione: Utente una volta selezionato il parcheggio scelto, decide di effettuare un prenotazione presso quel luogo.

Attori coinvolti: Utente e Database

Trigger: Prenotazione avvenuta con successo

Postcondizione: Dopo aver effettuato la prenotazione, l'utente si dirigerà presso il parcheggio all'orario stabilito e potrà accedervi

Procedimento: 1) Utente cerca parcheggi

- 2) Vede la disponibilità
- 3) Sceglie la struttura
- 4) Effettua prenotazione

#### **UC4.4: Ricevere consegna prenotazione**

Breve descrizione: Utente dopo aver prenotato il posto auto riceve un QR Code della prenotazione.

Attori coinvolti: Utente e Database

Trigger: Ricezione codice avvenuto con successo

Postcondizione: Una volta arrivato alla struttura l'utente presenterà il QR Code alla colonnina di controllo

Procedimento: 1) Utente effettua prenotazione

2) Riceve codice di autenticazione

#### **UC4.5: Consultazione dello storico prenotazioni**

Breve descrizione: L'utente accede a una sezione dedicata dell'applicazione per visualizzare l'elenco completo di tutte le prenotazioni effettuate in precedenza.

Attori coinvolti: Utente e Database.

Trigger: L'utente seleziona la voce relativa allo storico dal menu a tendina dell'app.

Postcondizione: L'utente prende visione dei dettagli delle proprie prenotazioni passate (ID, orario, parcheggio e QR code).

Procedimento: 1. L'utente apre il menu a tendina dell'interfaccia principale.

2. Selezione della voce "Le mie prenotazioni".

3. Il sistema richiede i dati al database tramite l'ID utente.

4. Presa visione della lista dei risultati ottenuti.

#### **UC5.1: Monitoraggio stato parcheggio**

Breve descrizione: L'operatore può accedere alla piattaforma inserendo le credenziali che gli vengono date e il nome della struttura in cui lavora

Attori coinvolti: Operatore e Database

Trigger: Accesso avvenuto con successo

Postcondizione: Dopo l'accesso, l'operatore può controllare che nel parcheggio tutto funzioni correttamente

Procedimento: 1) Operatore accede alla piattaforma

2) Controlla che tutto funzioni nel parcheggio

#### **UC5.2: Gestione della manutenzione parcheggio**

Breve descrizione: I dati relativi alla registrazione degli utenti, ai pagamenti, agli accessi e alle uscite dal parcheggio vengono salvate nel Database

Attori coinvolti: Utente, Operatore e Database

Trigger: Registrazione avvenuta con successo/Pagamento avvenuto con successo/Ingresso/Uscita

Postcondizione: I dati vengono registrati nel Database per conservare tutte le operazioni e per effettuare a posteriori analisi sui dati (per esempio: previsioni sull'anno successivo)

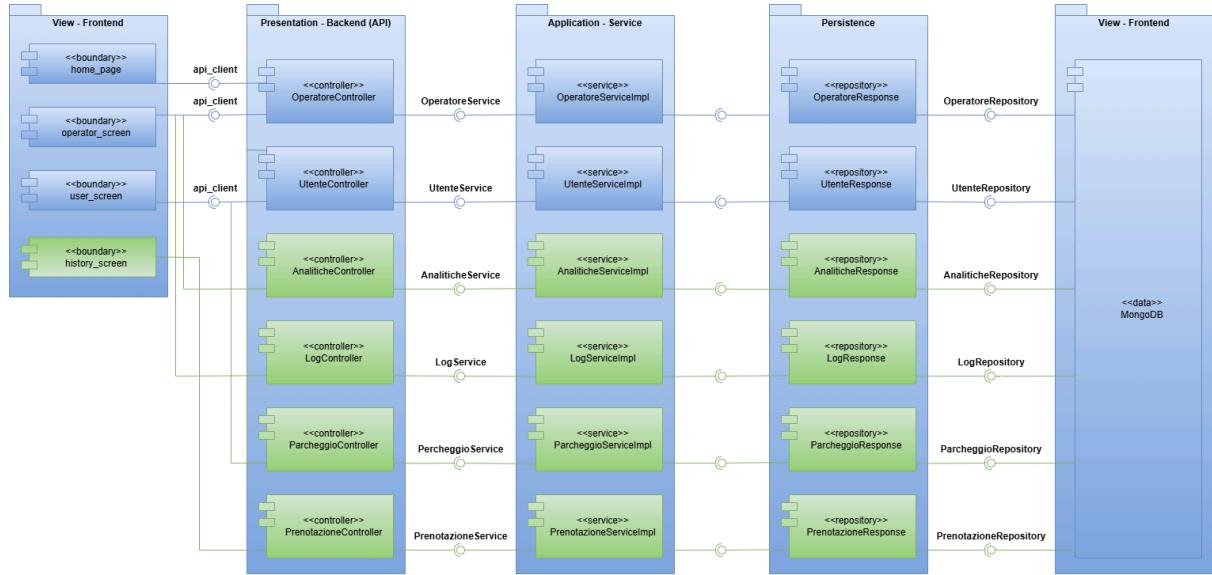
Procedimento: 1) Registrazione utente/Pagamento/Ingresso/Uscita

2) Salvataggio dati

## Diagrammi

### Diagramma delle componenti UML

I casi d'uso scelti per la seconda iterazione sono stati aggiunti al component diagram della prima iterazione



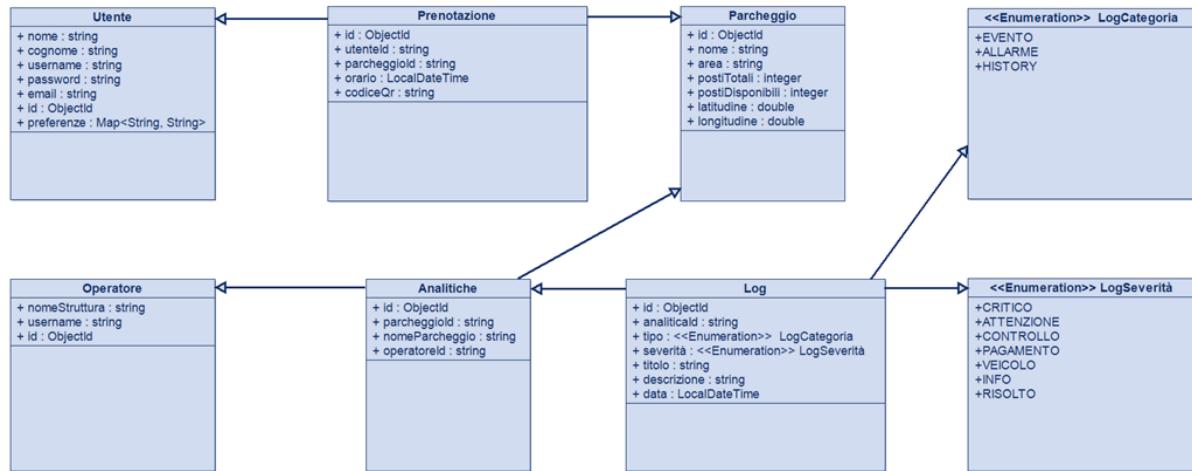
### Diagramma delle classi per le interfacce

Il diagramma rappresentante le interfacce dei nuovi casi d'uso introdotti.

<b>&lt;&lt;Interface&gt;&gt; AnaliticheService</b>	<b>&lt;&lt;Interface&gt;&gt; AnaliticheRepository</b>
+ getById(in id: string): Analitiche + getByOperatoreId(in operatoreId: string): Analitiche + save(in request: AnaliticheRequest): Analitiche	+ findByParcheggiIdAndOperatoreId(in parcheggiId: string, in operatoreId: string): List<Analitiche> + findByOperatoreId(in operatoreId: string): Optional<Analitiche> + findByParcheggiId(in parcheggiId: string): Optional<Analitiche>
<b>&lt;&lt;Interface&gt;&gt; LogService</b>	<b>&lt;&lt;Interface&gt;&gt; LogRepository</b>
+ salvaLog(in request: LogRequest): Log + getLogById(in id: string): Optional<Log> + getLogByAnalitichId(in analitichId: string): List<Log> + getLogByAnalitichIdAndTipo(in analitichId: string, in tipo: string): List<Log> + salvaLog1(in log: Log): Log	+ findByAnalitichId(in analitichId: string): List<Log> + findByAnalitichIdAndTipo(in analitichId: string, in tipo: string): List<Log> + findByAnalitichOrderbyDataDesc(in analitichId: string): List<Log>
<b>&lt;&lt;Interface&gt;&gt; ParcheggioService</b>	<b>&lt;&lt;Interface&gt;&gt; ParcheggioRepository</b>
+ cercaPerArea(in area: string): List<ParcheggioResponse> + effettuaPrenotazione(in req: PrenotazioneRequest): PrenotazioneResponse + cercaVicini(in lat: double, in long: double, in radius: double): List<ParcheggioResponse>	+ findByAreaContainingIgnoreCase(in area: string): List<Parcheggio>
<b>&lt;&lt;Interface&gt;&gt; PrenotazioneService</b>	<b>&lt;&lt;Interface&gt;&gt; PrenotazioneRepository</b>
+ getStoricoUtente(in utenteId: string): List<PrenotazioneResponse>	+ findByUtenteId(in utenteId: string): List<Prenotazione>

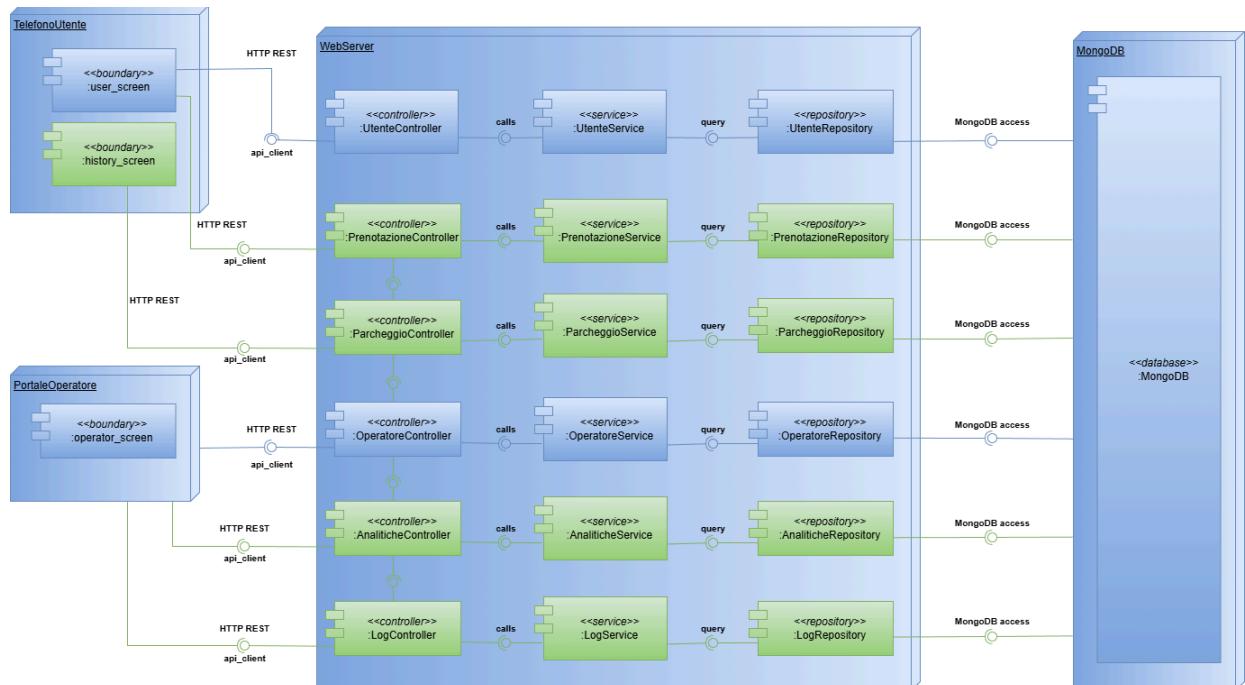
## Diagramma delle classi per tipi di dato

Il tipo di dato Analitiche, Log, Parcheggio e Prenotazione vengono inseriti nel Diagramma per tipo di dato dell'iterazione precedente, il nuovo diagramma dispone le nuove interazioni tra i dati.



## Diagramma di deployment

I nuovi componenti definiti in questa seconda iterazione sono stati inseriti nel Deployment Diagram dell'iterazione numero uno.



## Testing

### Analisi statica

Per garantire l'alta qualità, la manutenibilità e la sicurezza del codice sviluppato durante l'Iterazione 2 del progetto, è stata eseguita un'analisi statica integrata.

SonarLint ha identificato e guidato la risoluzione di *Code Smells*, potenziali *Bug* e violazioni delle *best practice* di programmazione.

### Analisi dinamica

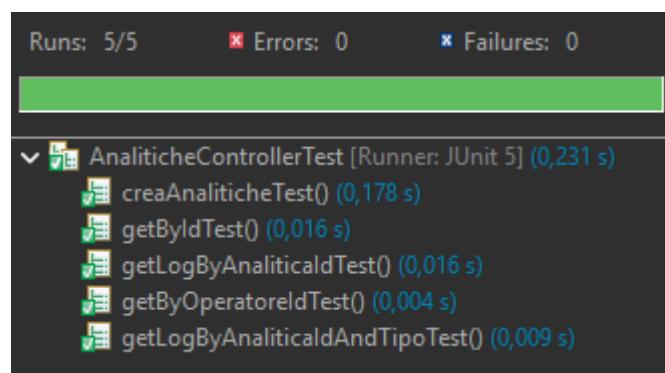
Per l'iterazione 2 sono stati effettuati i test delle seguenti funzioni:

- Creazione, recupero e salvataggio delle Analitiche
- Creazione, recupero, salvataggio e aggiornamento dei Log
- Ricerca e prenotazione di un Parcheggio
- Recupero storico delle Prenotazioni

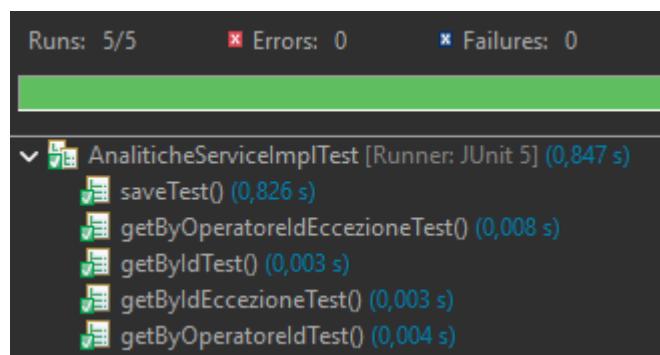
### JUnit test

Sono stati realizzati test JUnit 5 sulle seguenti classi:

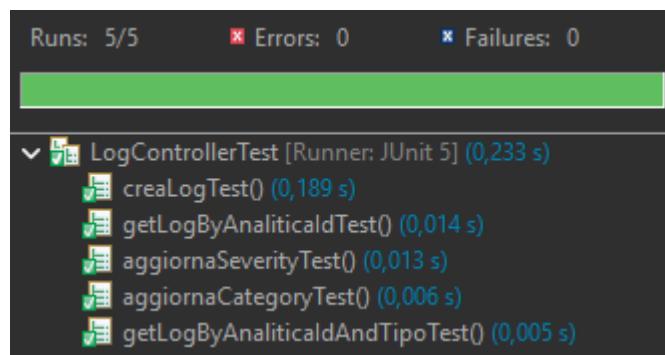
- 1) AnaliticheController



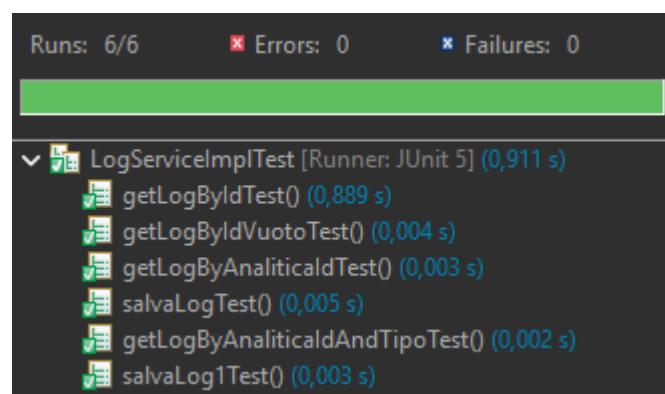
- 2) AnaliticheServiceImpl



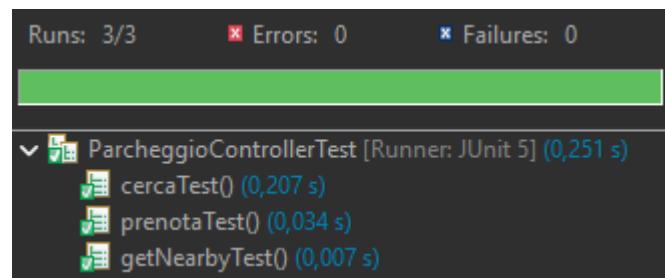
### 3) LogController



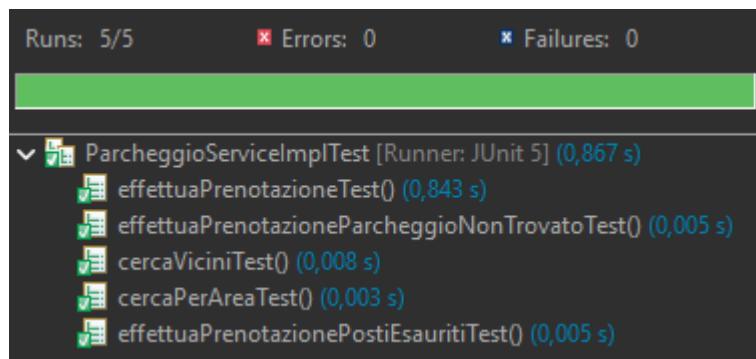
### 4) LogServiceImpl



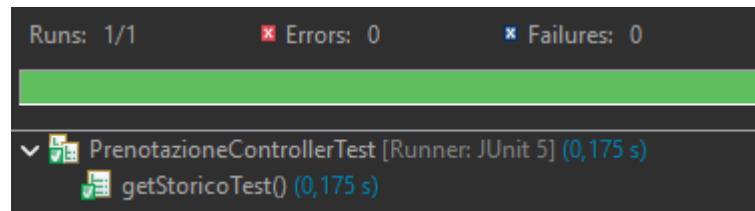
### 5) ParcheggioController



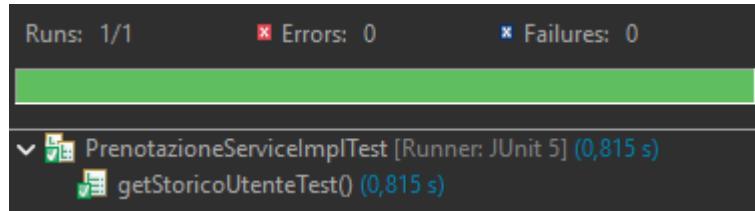
### 6) ParcheggioServiceImpl



## 7) PrenotazioneController



## 8) PrenotazioneServiceImpl



## Coverage:

PMG-backend	94,5 %	4.631	272	4.903
> src/main/java	88,0 %	1.619	221	1.840
> src/test/java	98,3 %	3.012	51	3.063

## Documentazione API

Queste sono le API sviluppate nella seconda iterazione, è possibile prenderne visione tramite la collezione Postman presente nel repository GitHub:

- API per la ricerca dei parcheggi in base all'area
- API per la prenotazione di un posto nel parcheggio
- API per recuperare lo storico delle prenotazioni dell'utente
- API per recuperare le analitiche del parcheggio
- API per recuperare i log di un'analitica

Comandi Utente / Ricerca parcheggi per area

GET http://localhost:8080/api/parcheggi/cerca?area=Milano

Body

This request does not have a body

200 OK

Body JSON

```
[{"id": "69442f7d8ee99a6e7d22a3df", "nome": "Parcheggio Duomo", "area": "Milano Centro", "postiTotali": 120, "postiDisponibili": 43, "latitudine": 45.4642, "longitudine": 9.1916}]
```

## Ricerca dei parcheggi in base all'area

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/api/parcheggi/prenota
- Body (JSON):**

```
1 {
2     "utenteId": "692f29200205c708abbd5494",
3     "parcheggioId": "69442f7d8ee99a6e7d22a3df"
4 }
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2     "id": "696a843cb9f80b5396889160",
3     "utenteId": "692f29200205c708abbd5494",
4     "parcheggioId": "69442f7d8ee99a6e7d22a3df",
5     "orario": null,
6     "codiceQr": "002d1a77-4d4e-47ed-8722-9fa38a227ead"
7 }
```

## Prenotazione di un posto nel parcheggio

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/api/prenotazioni/utente/69315407f917a04071dbafc1
- Body:** This request does not have a body
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 [
2     {
3         "id": "695a465d7c5c7356ee334045",
4         "utenteId": "69315407f917a04071dbafc1",
5         "parcheggioId": "69442f7d8ee99a6e7d22a3df",
6         "orario": "2026-01-04T11:52:13.491",
7         "codiceQr": "6f024a3b-33a8-4ae3-a68c-fe979b1e3cab"
8     },

```

## Recupero storico delle prenotazioni dell'utente

HTTP Comandi Operatore / Recupero analitiche parcheggio

GET http://localhost:8080/api/analitiche/694aa3eeb7b5590ae69d9379

Docs Params Authorization Headers (7) Body Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

This request does not have a body

Body Cookies Headers (8) Test Results 200 OK

{ } JSON ▾ ▷ Preview Visualize ▾

```
1 {  
2   "id": "694aa3eeb7b5590ae69d9379",  
3   "parcheggioId": "69442f7d8ee99a6e7d22a3df",  
4   "nomeParcheggio": "Parcheggio Duomo",  
5   "operatoreId": "692d6d74909d86ebac14489c"  
6 }
```

## Recupero delle analitiche del parcheggio

HTTP Comandi Operatore / Recupero logs analitica

GET http://localhost:8080/api/log/analitiche/694aa3eeb7b5590ae69d9379/log

Docs Params Authorization Headers (7) Body Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

This request does not have a body

Body Cookies Headers (8) Test Results 200 OK

{ } JSON ▾ ▷ Preview Visualize ▾

```
1 [  
2   {  
3     "id": "694aa8f7b7b5590ae69d938b",  
4     "tipo": "HISTORY",  
5     "titolo": "Sbarra ingresso bloccata",  
6     "descrizione": "Motore in overload",  
7     "data": "2025-12-29T12:25:55.198",  
8     "severita": "RISOLTO"  
9   },
```

## Recupero dei log di un'analitica