Iterazione 1
Descrizione use case

UC1 Utente
- UC1.1: Login
- UC1.2: Registrazione
- UC1.3: Selezione preferenze
- UC1.4: Logout

UC2 Operatore
- UC2.1: Accesso alla piattaforma

UC3 Database
- UC3.1: Salvataggio dati

UC1.1: Registrazione
Breve descrizione: Utente accede all'app per la prima volta, seleziona tasto per la registrazione, compila i campi richiesti e i dati vengono inviati al database
Attori coinvolti: Utente e Database
Trigger: Registrazione avvenuta con successo
Postcondizione: Dopo la registrazione l'utente viene rimandato alla schermata di login così da poter validare le credenziali appena create e accedere alla piattaforma.
Procedimento: 1) Accesso all'app
2) Selezione bottone registrazione
3) Compilazione campi
4) Click sul tasto di invio
5) Reindirizzamento alla schermata di login

UC1.2: Login
Breve descrizione: Utente arriva sulla pagina di login dopo aver completato la registrazione e immette le sue credenziali così da accedere all'app e poter navigare; se non è il suo primo accesso l'utente viene automaticamente autenticato nella homepage
Attori coinvolti: Utente
Trigger: Login avvenuto con successo
Postcondizione: Dopo il login l'utente può accedere alla piattaforma e cercare / controllare i parcheggi
Procedimento: 1A) Utente compila i campi di login
2A) Entra nell'applicazione

1B) Utente entrando nella piattaforma si trova direttamente nella homepage

UC1.3: Selezione preferenze
Breve descrizione: Dopo aver eseguito l'accesso nella schermata preposta al profilo personale, l'utente può selezionare delle preferenze relative al parcheggio che agevoleranno la ricerca delle strutture in tempo di ricerca
Attori coinvolti: Utente e Database
Trigger: Preferenze salvate correttamente
Postcondizione: Dopo la selezione l'utente trova solo i parcheggi che matchano con le preferenze che ha selezionato

Procedimento: 1) Utente clicca sul suo profilo personale
2) Seleziona l'impostazione delle preferenze
3) Sceglie quelle che più gli corrispondono
4) Salva le impostazioni

UC1.4: Logout
Breve descrizione: Utente loggato nella piattaforma clicca sul tasto in alto a destra per seguire il logout dall'app.
Attori coinvolti: Utente
Trigger: Logout avvenuto con successo
Postcondizione: Dopo il logout l'utente può riaccedere alla piattaforma o uscire da quest'ultima
Procedimento: 1) Utente seleziona icona in alto a destra
2) Conferma di voler uscire dalla piattaforma

UC2.1: Accesso alla piattaforma
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

UC3.1: Salvataggio dati
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
Una volta scelti i casi d'uso per l'iterazione e definite le caratteristiche di design, è stata formalizzata una prima architettura software del nostro sistema.
I casi d'uso sono stati raggruppati e per ognuno è stato introdotto un componente *boundary, control* o *data*.

- <<*boundary*>> - componente front-end con cui gli attori si interfacciano;
- <<*controller*>> - componente back-end che gestisce la logica del programma ed espone le API;
- <<*service*>> - componente preposto alla logica applicativa che coordina le operazioni e decide "cosa fare";
- <<*repository*>> - componente di astrazione che incapsula query e operazioni CRUD verso il database;
- <<*database*>> - componente back-end che interagisce col database.

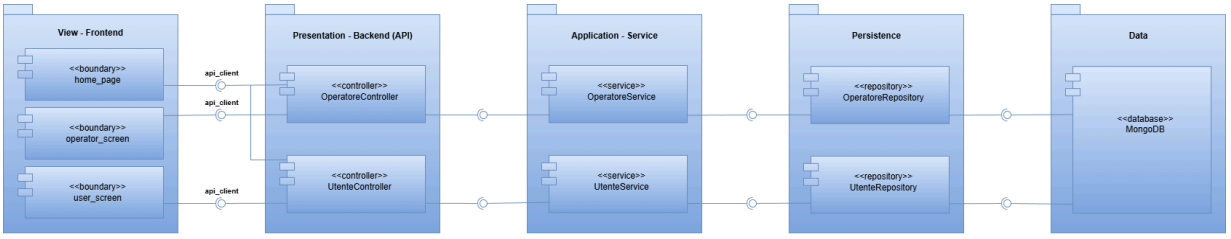


Diagramma delle classi per le interfacce
Il diagramma rappresentante le interfacce del sistema con le relative funzioni e i dati I/O.

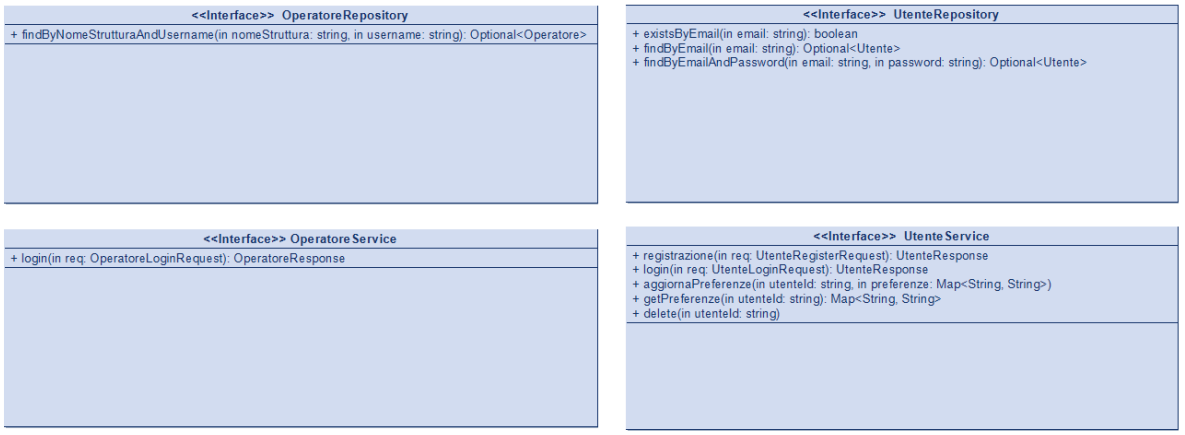
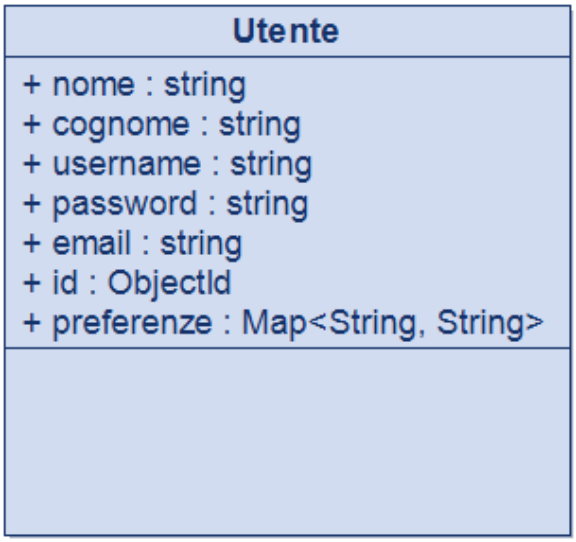


Diagramma delle classi per tipi di dato
Diagramma con i tipi di dato necessari allo sviluppo dell'applicazione.



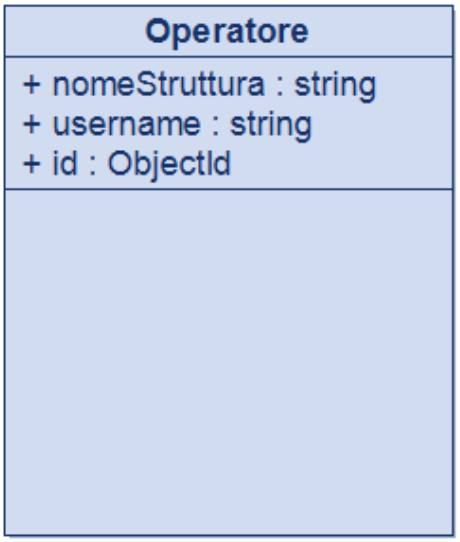
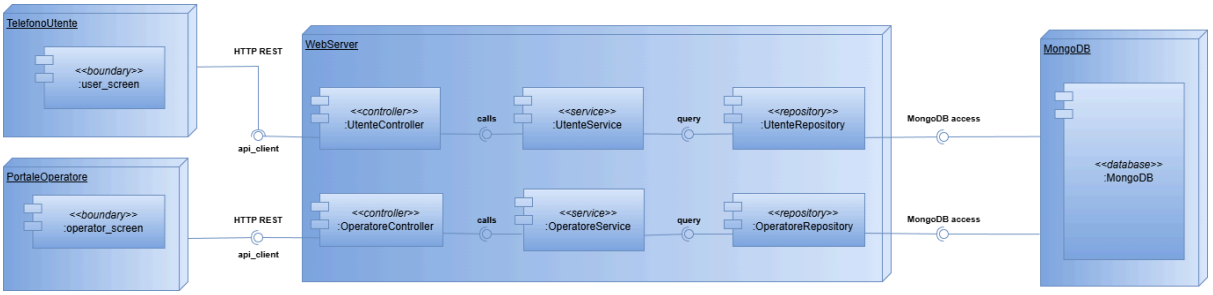


Diagramma di deployment
Mediante l'utilizzo di un Deployment Diagram UML viene rappresentate la gestione hardware e software del sistema, di seguito sono presentati i componenti nei seguenti nodi:

- *TelefonoUtente*, nodo su cui un utente può registrarsi e loggarsi alla piattaforma
- *PortaleOperatore*, nodo su cui un operatore può loggarsi nella piattaforma per avere una panoramica dei servizi;
- *WebServer*, espone le API richieste lato front-end;
- *MongoDB*, si occupa dello storage dei dati.



Testing

Analisi statica (SonarLint)

Per garantire l'alta qualità, la manutenibilità e la sicurezza del codice sviluppato durante l'Iterazione 1 del progetto, è stata eseguita un'analisi statica integrata.
Il tool scelto è SonarLint (plugin di SonarQube), utilizzato in Eclipse.
Durante la procedura ha identificato e guidato la risoluzione di Code Smells, potenziali Bug e violazioni delle best practice di programmazione.
Di seguito sono riepilogate le modifiche eseguite, suddivise per package e file.

Analisi dinamica (Postman e JUnit)

L'analisi dinamica è stata effettuata tramite:

- JUnit 5 per l'esecuzione dei test
- Interazione reale con MongoDB
  Nessuna simulazione o mock del database:
  → i comportamenti riflettono il funzionamento effettivo del sistema.
- Inserimento e rimozione temporanea dei dati
  Ogni test genera dati unici (basati su System.nanoTime()) per evitare collisioni tra esecuzioni.

Questa metodologia garantisce risultati ripetibili e validi anche in presenza di test multipli.

Componenti Coinvolte

1) Modulo Utente
2) Modulo Operatore
3) Modulo Connessione

La correttezza di tali connessioni è fondamentale per l'esito positivo dell'analisi dinamica.

Struttura dei Test Dinamici

Sono stati realizzati test JUnit 5 sulle seguenti classi:

1) ConnessioneTest
2) UtenteTest
3) DatiUtentiTest
4) OperatoreTest
5) DatiOperatoriTest

Risultati dell'Analisi Dinamica

Tutti i test progettati hanno confermato che:

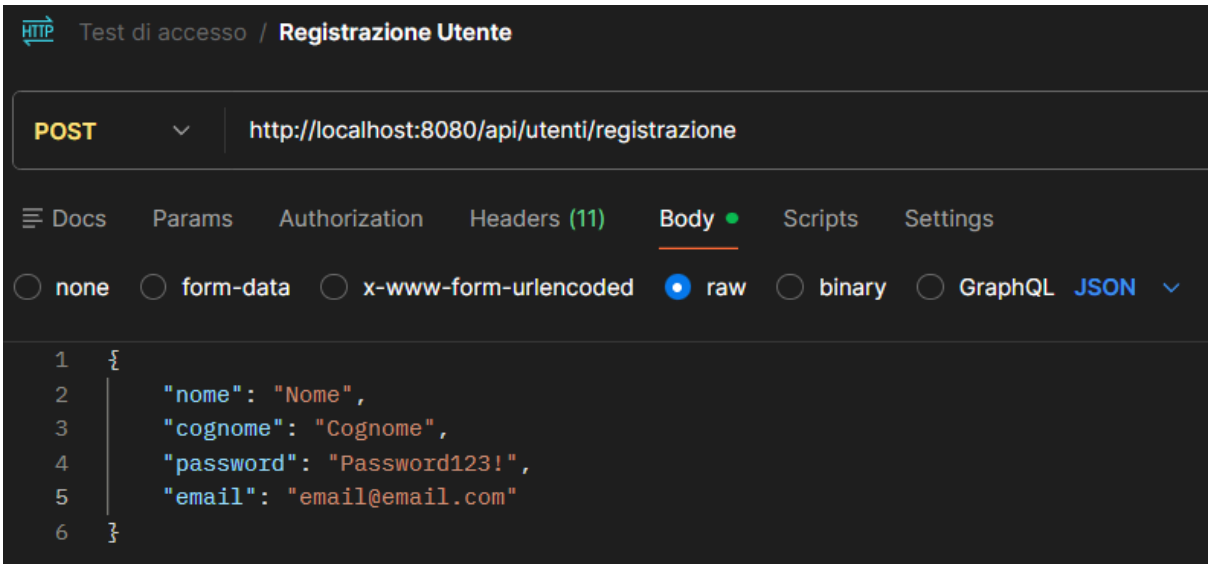
- il sistema interagisce correttamente con il database
- la logica applicativa rispetta il comportamento atteso
- eccezioni e condizioni di errore sono correttamente rilevate
- la separazione tra interfacce e implementazioni è coerente
- i metodi critici (loginDB, registrazioneDB, esisteOperatore) reagiscono correttamente sia a input validi che non validi

Non sono emersi malfunzionamenti critici.

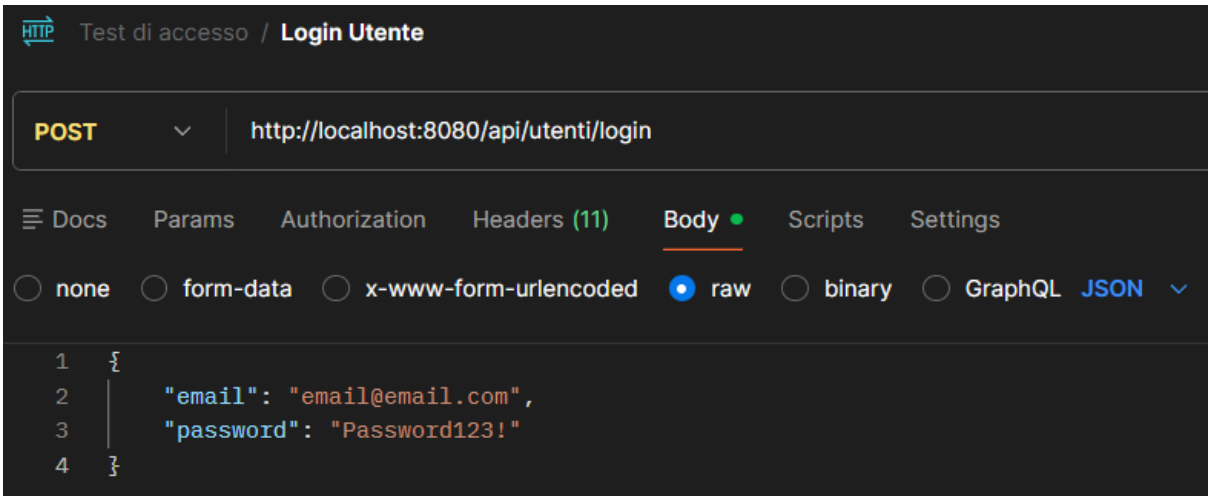
Documentazione API

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

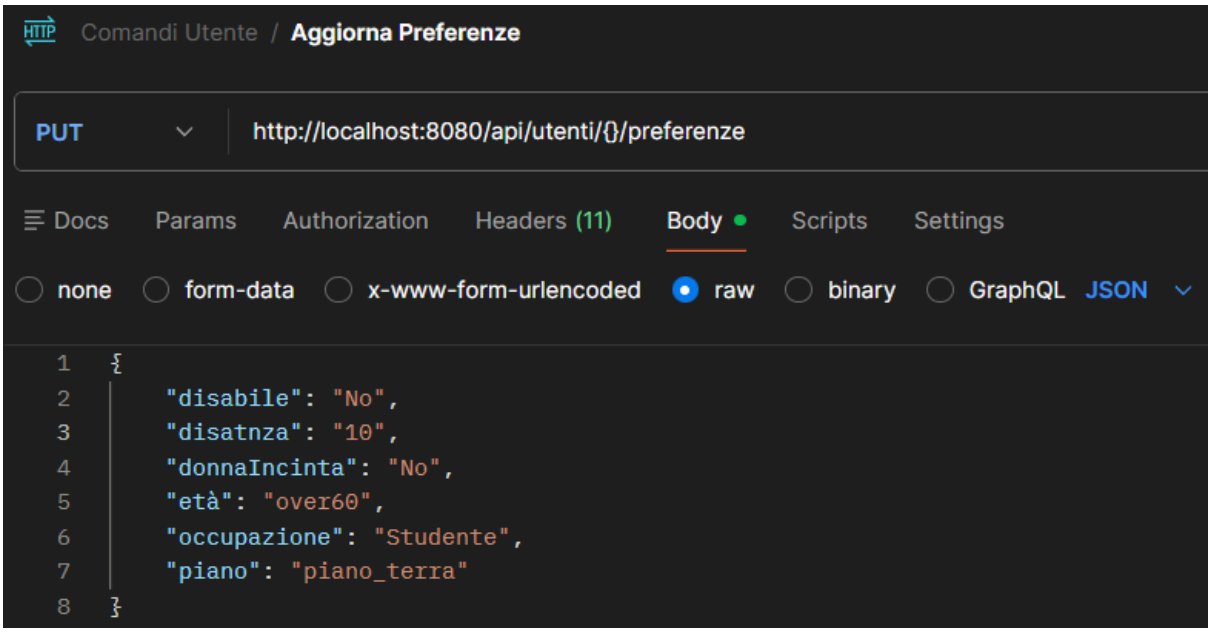
- API per la registrazione e il login dell'utente
- API per la selezione delle preferenze dell'utente
- API per l'accesso dell'operatore



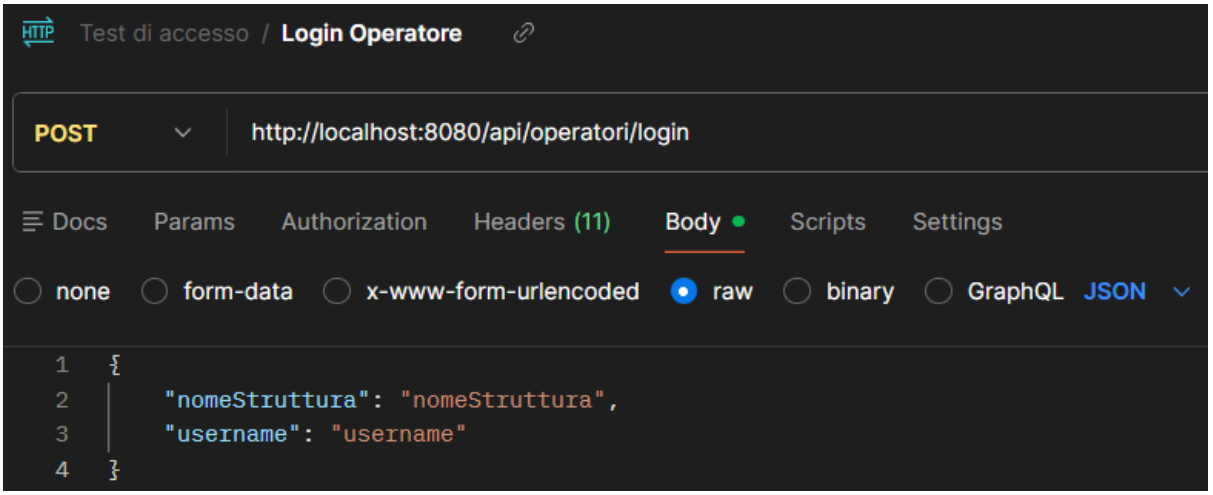
API per la registrazione dell'utente



API per il login dell'utente



API per aggiornare le preferenze



API per il login dell'operatore