****

**PROGETTO DI INGEGNERIA DEL SOFTWARE**

**SKI ONLINE**

**Sviluppo applicazione**

# Indice dei contenuti

[Indice dei contenuti 2](#_Toc122646042)

[Scopo del documento 2](#_Toc122646043)

[User flows 2](#_Toc122646044)

[Application Implementation and Documentation 4](#_Toc122646045)

[Project Structure 4](#_Toc122646046)

[Project Dependencies 5](#_Toc122646047)

[Project Data or DB 5](#_Toc122646048)

[Project APIs 7](#_Toc122646049)

[RESOURCES EXTRACTION FROM THE CLASS DIAGRAM 7](#_Toc122646050)

[RESOURCES MODELS 7](#_Toc122646051)

[Sviluppo API 10](#_Toc122646052)

[UTENTE 10](#_Toc122646053)

[API documentation 10](#_Toc122646054)

[FrontEnd Implementation 12](#_Toc122646055)

[GitHub Repository and Deployment info 12](#_Toc122646056)

[Testing 12](#_Toc122646057)

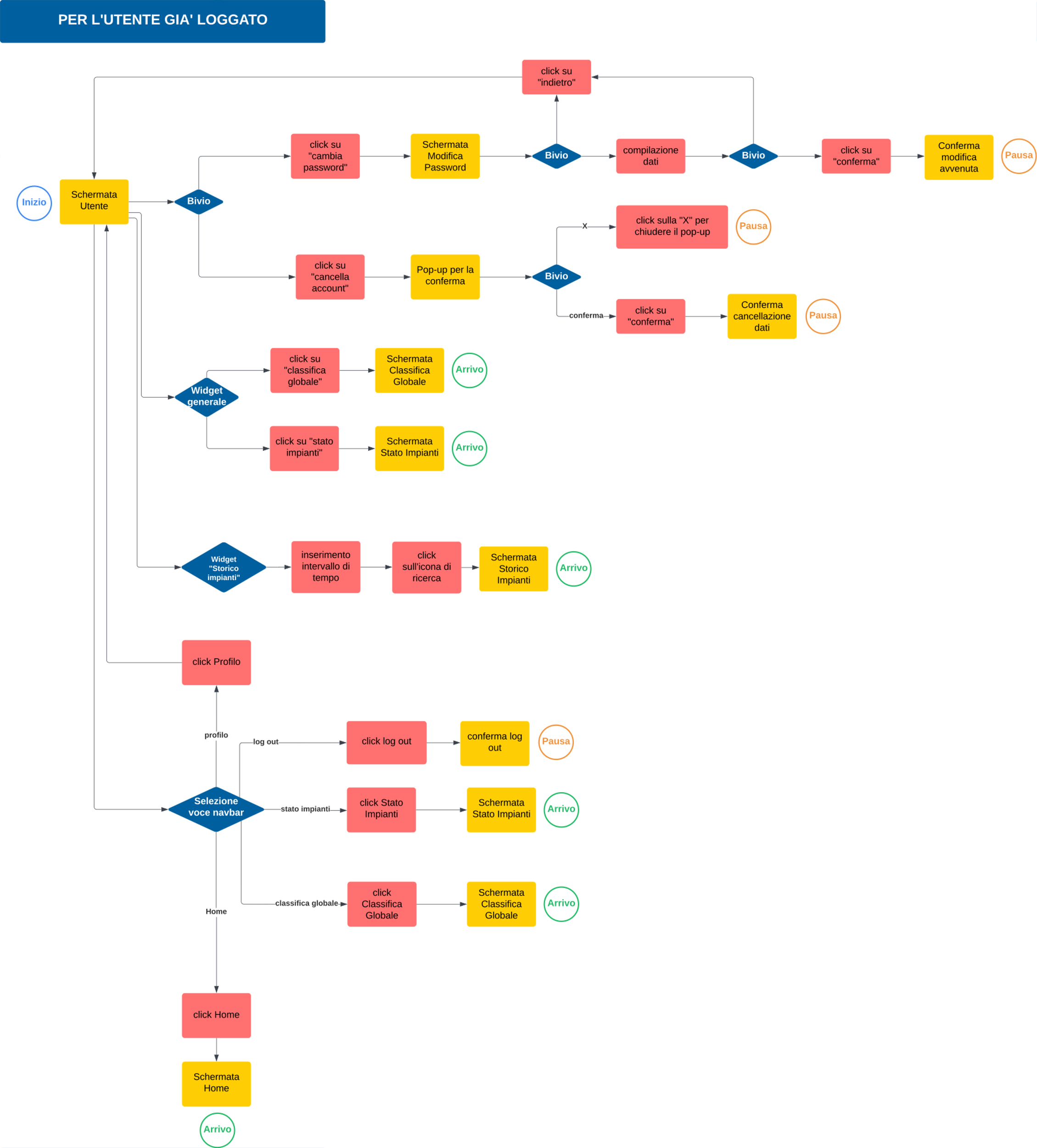
**Scopo del documento**

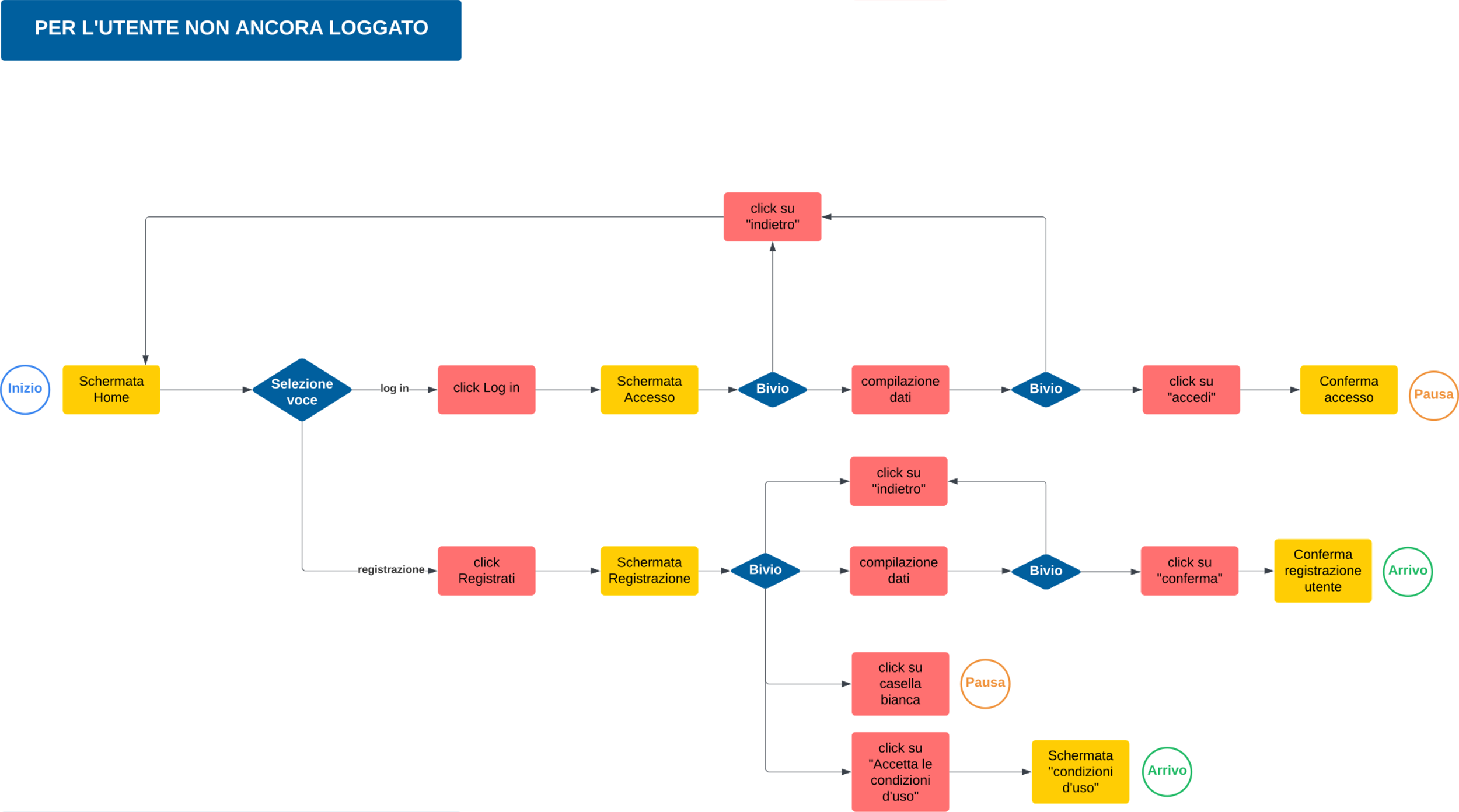
Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di una parte  
dell’applicazione Ski Online. In particolare, presenta tutti gli artefatti necessari per  
realizzare i servizi di gestione *degli utenti e degli impianti di risalita dell’applicazione Ski Online*.  
Partendo dalla descrizione degli user flow legate *alle azioni eseguibili dagli utenti anonimi, dagli utenti registrati e dagli utenti di sistema (operatore e gestore)*, il documento prosegue con la presentazione delle  
API necessarie (tramite l’API Model e il Modello delle risorse) per *effettuare il login, registrazione, cancellazione di un account, visualizzare lo stato di affollamento degli impianti e lo storico dei log degli accessi agli stessi* necessari all’applicazione Ski Online.  
Per ogni API realizzata, oltre ad una descrizione delle funzionalità fornite, il documento  
presenta la sua documentazione e i test effettuati. Infine una sezione è dedicata alle  
informazioni del Git Repository e il deployment dell’applicazione stessa.

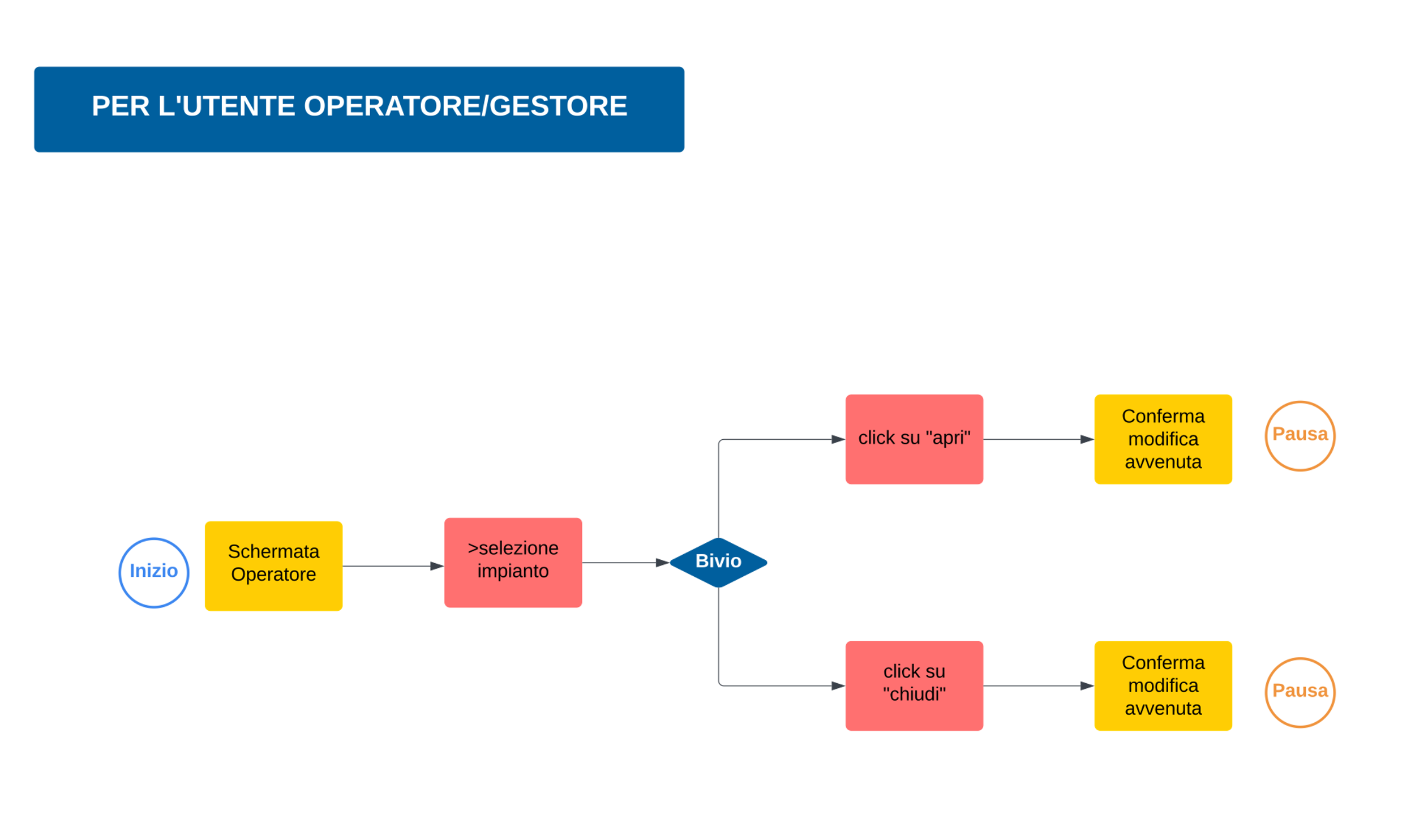
# User flows

In questa sezione del documento di sviluppo riportiamo gli “user flows” per il ruolo sia dell’utente registrato che per quello anonimo.

La prima figura descrive gli user flows relativi alle operazioni di log-in e registrazione da parte dell’utente anonimo. Diversamente, la seconda si riferisce all’utente registrato e alle azioni che può effettuare a partire dalla “Schermata Utente” a lui dedicata; mentre, la terza descrive lo stato di apertura degli impianti gestita dall’Utente Gestore o Operatore.







# Application Implementation and Documentation

## Project Structure

La struttura del progetto è presentata in figura ed è composta di una cartella API per la  
gestione delle API locali, di una cartella static per la parte del front-end, dei file di sistema come il file .env, .gitignore e package.json e del file swagger.json per la documentazione delle API che verrà discussa in seguto

Immagine che contiene testo

Descrizione generata automaticamente

## Project Dependencies

I seguenti moduli Node sono stati utilizzati e aggiunti al file package.Json

* Bcrypt
* Dotenv
* Express
* Jsonwebtoken
* Mongoose
* Mongoose-express-api
* Swagger-ui-express

## Project Data or DB

Per la gestione dei dati utili all’applicazione abbiamo definito tre principali strutture dati come illustrato in figura. Una collezione di “Utenti”, una collezione di “Impianti” e una collezione di “Log” che lega il passaggio di un Utente per un Impianto.

Immagine che contiene tavolo

Descrizione generata automaticamente

Per rappresentare gli Utenti, Impianti e Log abbiamo definito i seguenti tipi di dati di esempio:

Immagine che contiene testo

Descrizione generata automaticamente

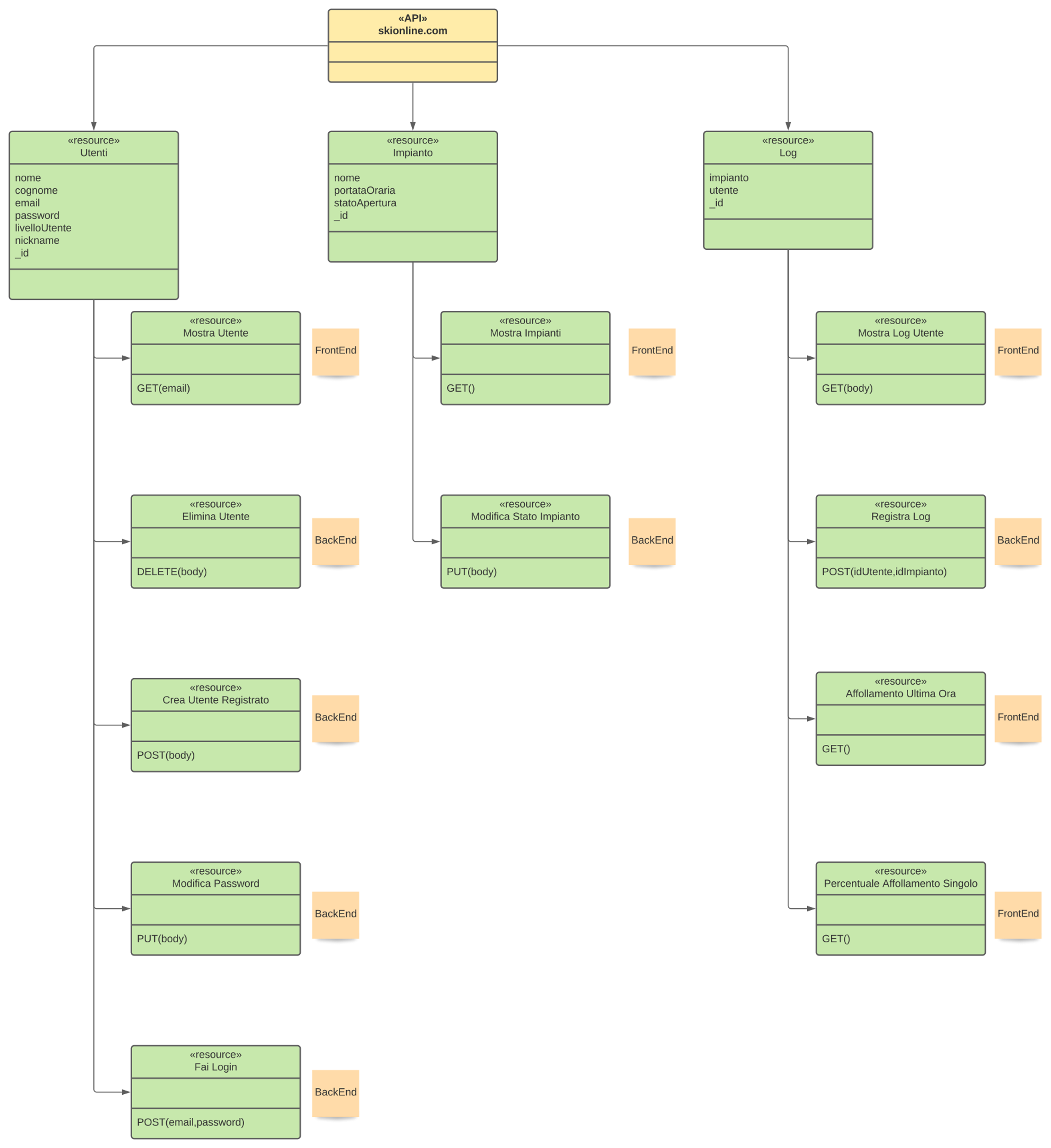
Immagine che contiene testo

Descrizione generata automaticamente



## Project APIs

### RESOURCES EXTRACTION FROM THE CLASS DIAGRAM



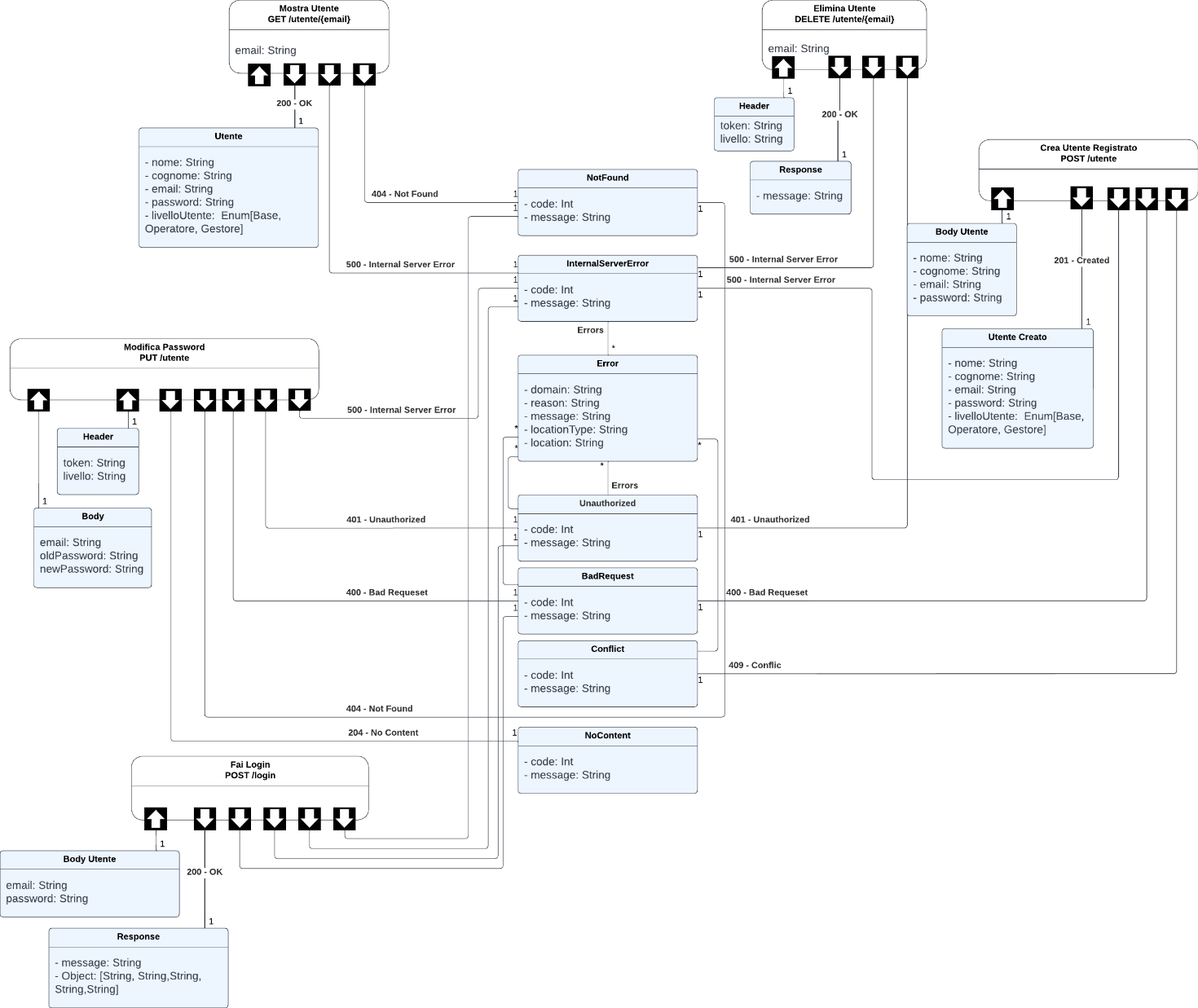
Nota bene: il parametro \_id indicato nelle risorse si riferisce all’identificativo univoco generato da MongoDB all’atto di registrazione dei dati

### RESOURCES MODELS

Basandoci sulla sezione precedente sono state realizzate cinque Resources Models suddivisi per tipologia.

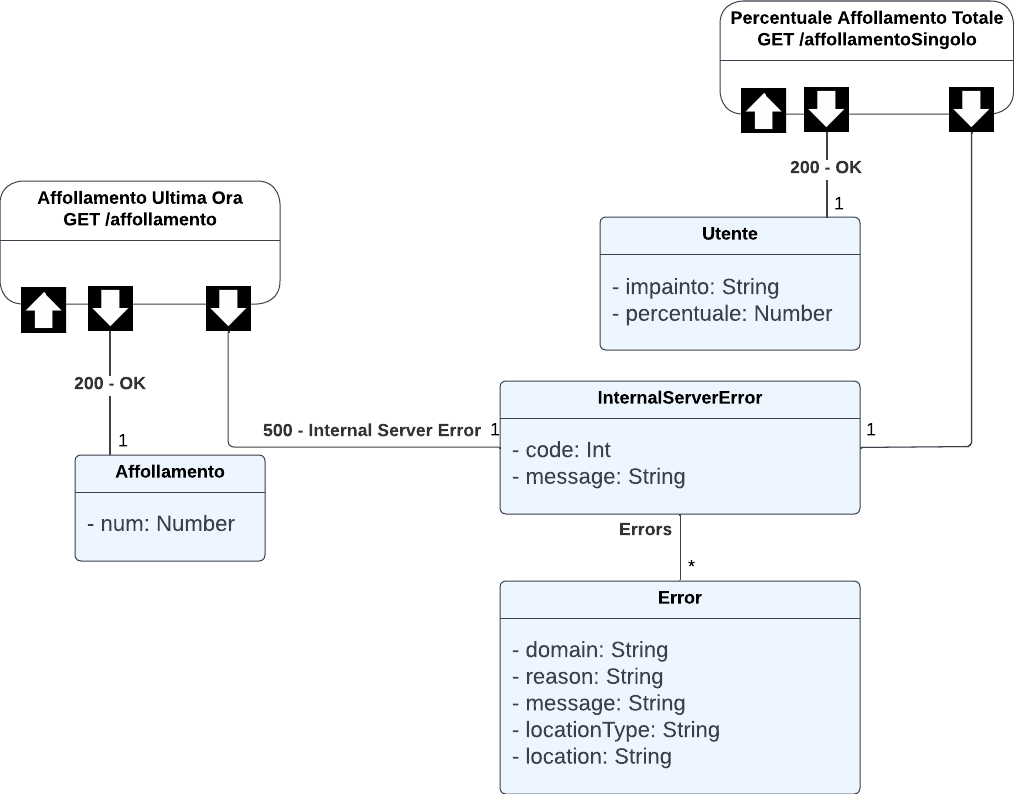
#### UTENTI

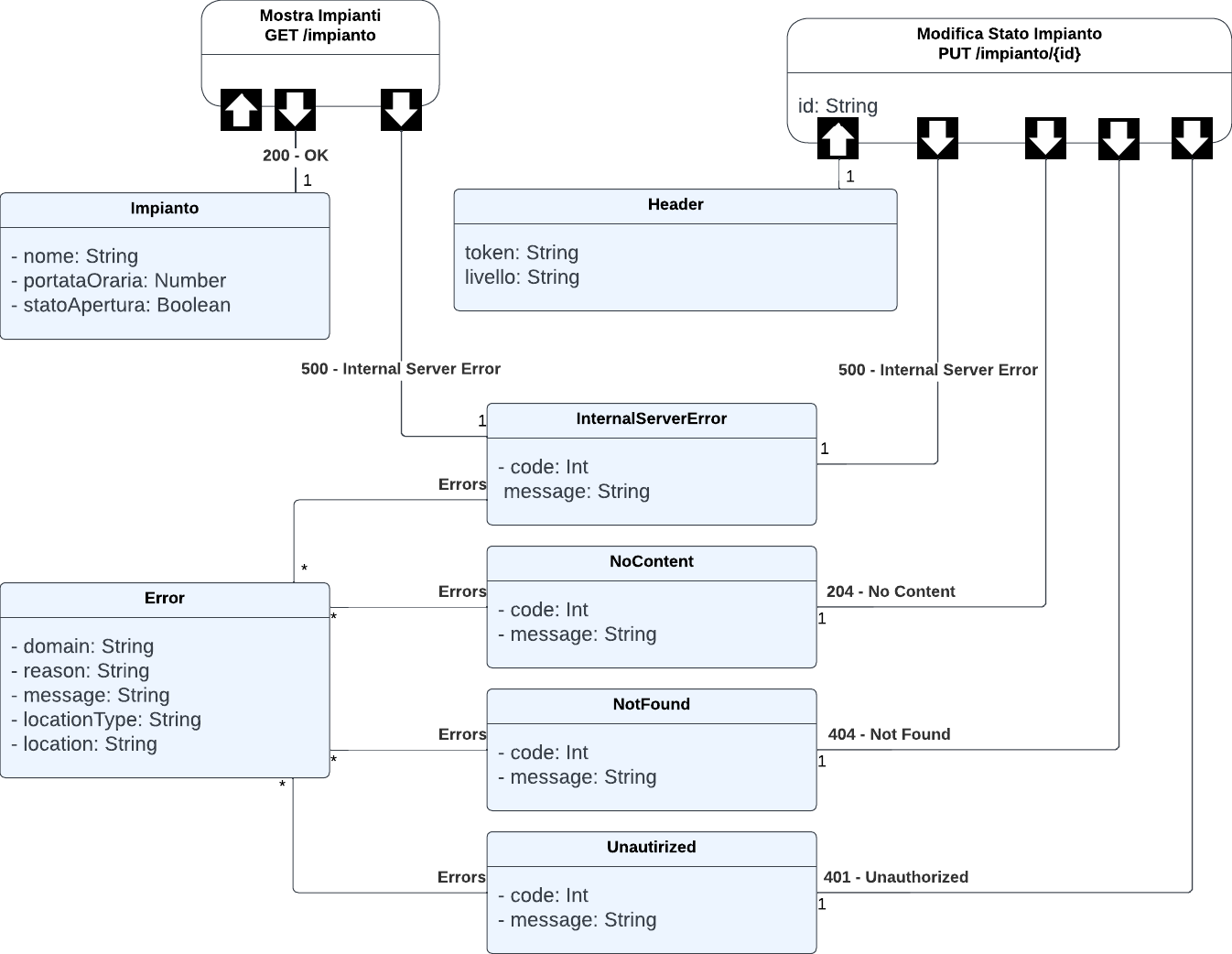
<descrizione>



#### IMPIANTI

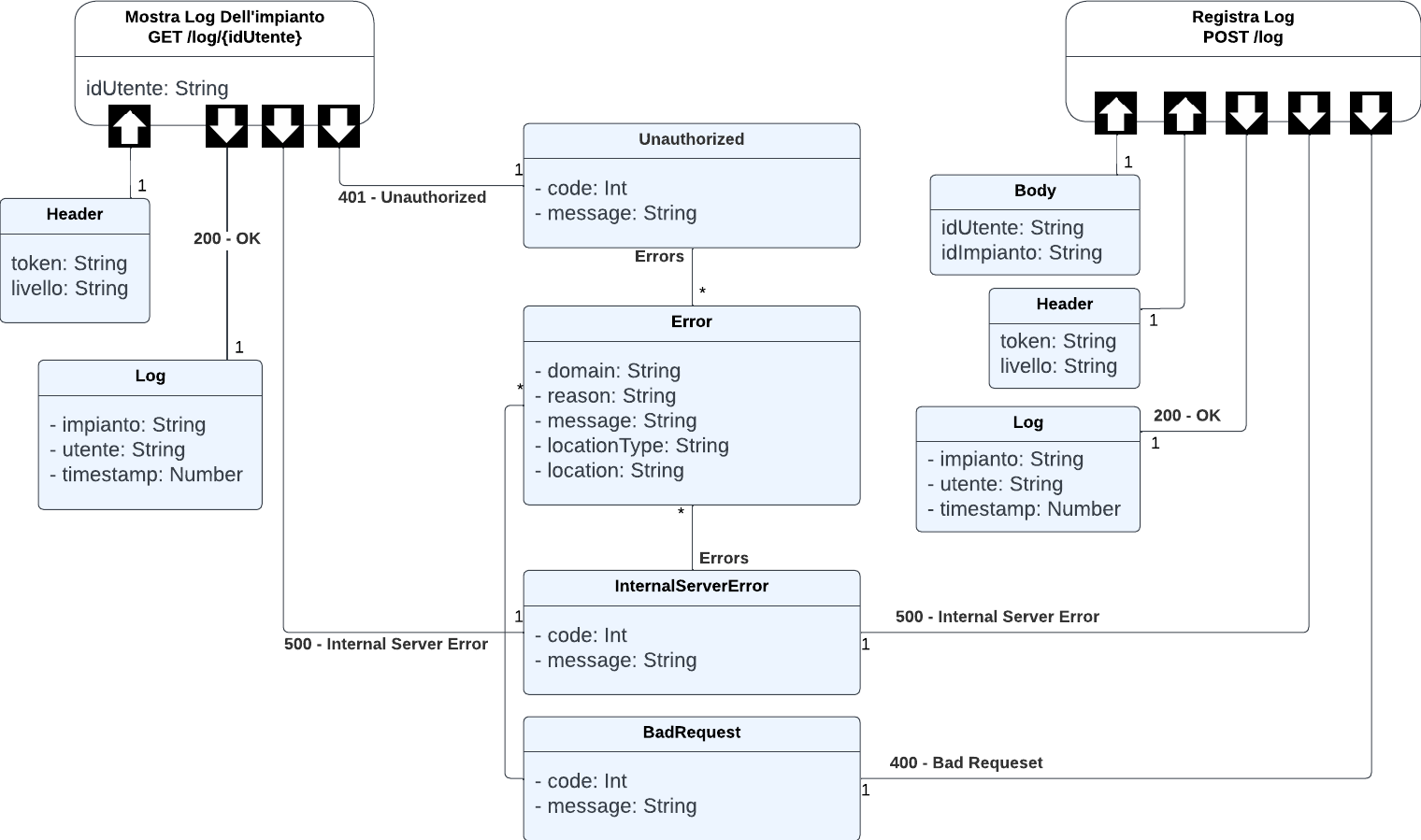
<descrizione>





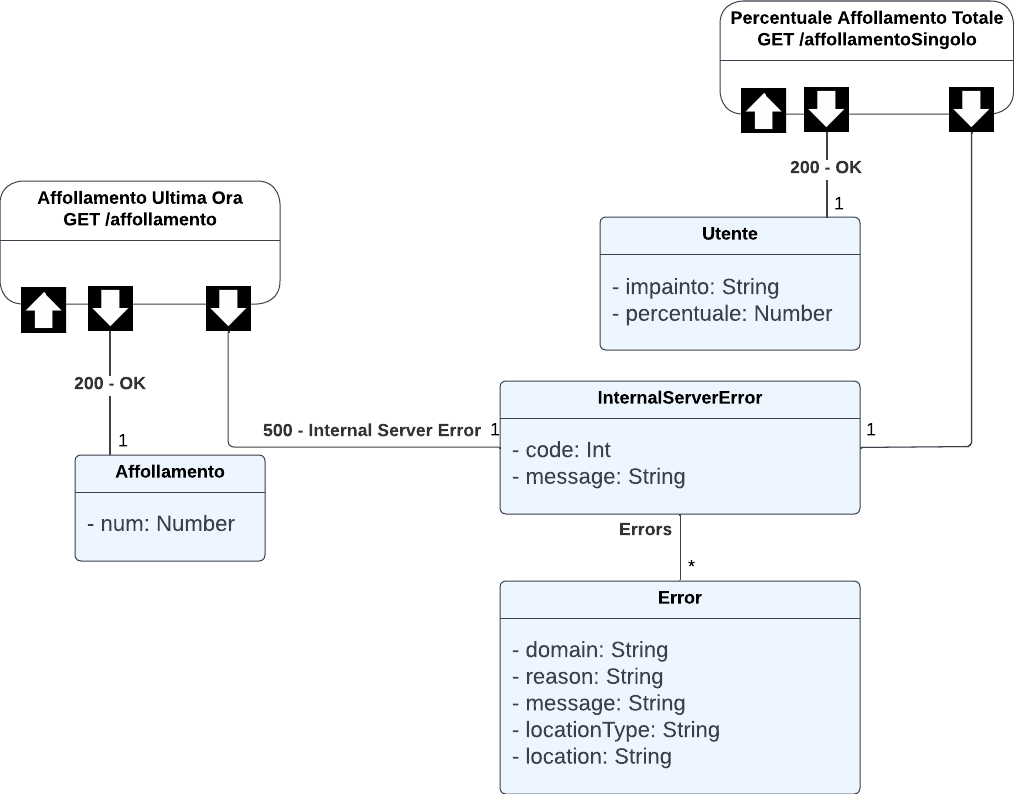
#### LOG

<descrizione>



#### AFFOLLAMENTO

<descrizione>

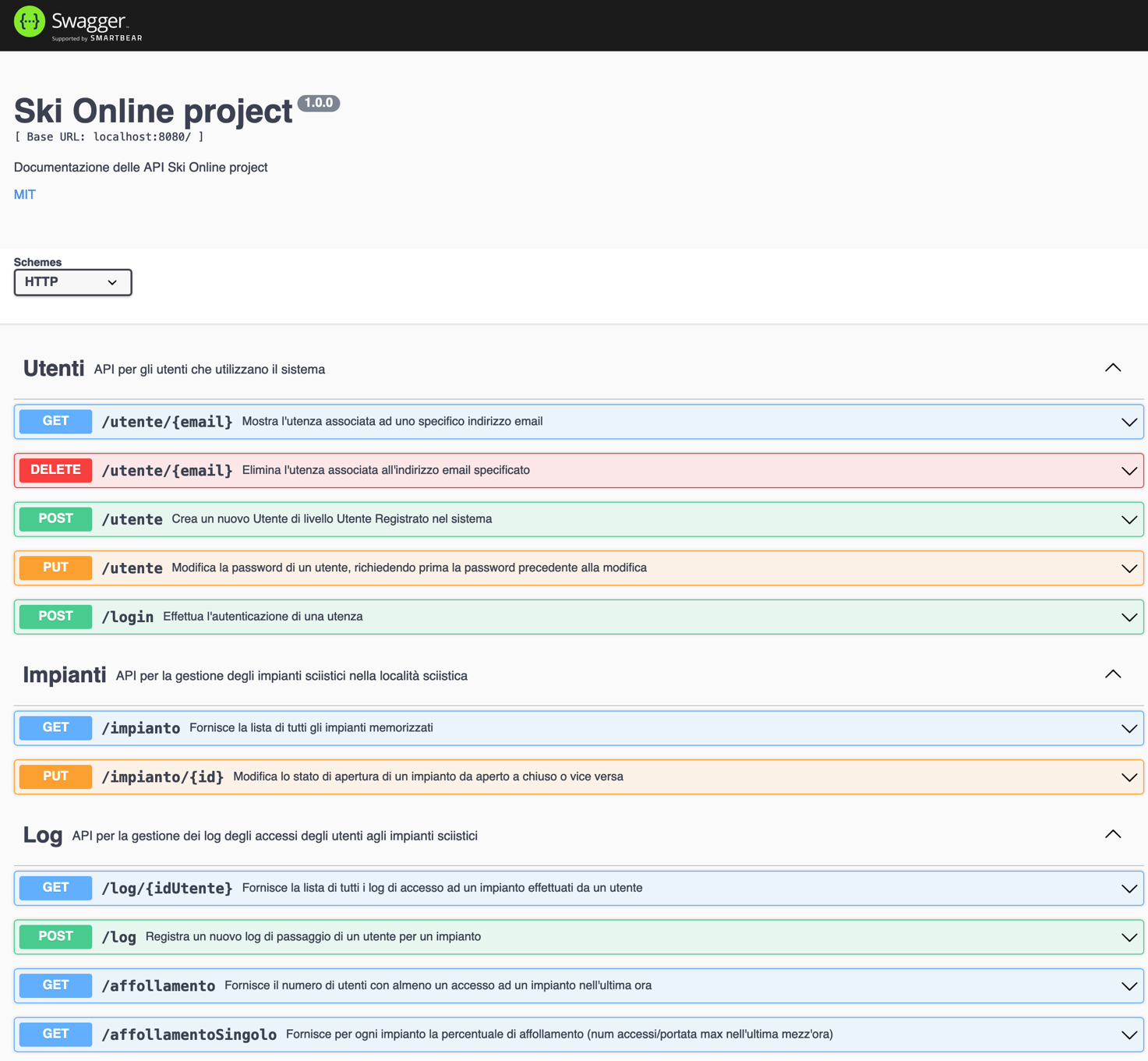


## Sviluppo API

### UTENTE

# API documentation

Le API Locali fornite dall’applicazione Ski Online e descritte nella sezione precedente sono state documentate utilizzando il modulo NodeJS chiamato Swagger UI Express. In questo modo la documentazione relativa alle API è direttamente disponibile a chiunque veda il codice sorgente. Per poter generare l’endpoint dedicato alla presentazione delle API abbiamo utilizzato Swagger UI in quanto crea una pagina web dalle definizioni delle specifiche OpenAPI. In particolare, di seguito mostriamo la pagina web relativa alla documentazione che presenta le 11 API (GET, POST, DELETE e PUT) per la gestione dei dati della nostra applicazione. La GET viene utilizzata per visualizzare i dati in una pagina HTML. La POST per inserire un nuovo dato nel nostro sistema o per l’operazione di login. La DELETE per cancellare un dato dal nostro sistema. La PUT per modificare un dato già presente nel nostro sistema. L’endpoint da invocare per raggiungere la seguente documentazione è: <http://localhost:8080/api-docs>



# FrontEnd Implementation

# GitHub Repository and Deployment info

# Testing