****

**PROGETTO DI INGEGNERIA DEL SOFTWARE**

**SKI ONLINE**

**Sviluppo applicazione**

# Indice dei contenuti

[Indice dei contenuti 2](#_Toc122889298)

[Scopo del documento 3](#_Toc122889299)

[User flows 3](#_Toc122889300)

[Utente Anonimo 3](#_Toc122889301)

[Utente Registrato 4](#_Toc122889302)

[Utente Operatore/Gestore 5](#_Toc122889303)

[Application Implementation and Documentation 6](#_Toc122889304)

[Project Structure 6](#_Toc122889305)

[Project Dependencies 6](#_Toc122889306)

[Project Data or DB 7](#_Toc122889307)

[Project APIs 8](#_Toc122889308)

[Resources extraction from the class diagram 8](#_Toc122889309)

[Resources models 9](#_Toc122889310)

[Sviluppo API 12](#_Toc122889311)

[UTENTE 12](#_Toc122889312)

[API documentation 12](#_Toc122889313)

[FrontEnd Implementation 13](#_Toc122889314)

[GitHub Repository and Deployment info 13](#_Toc122889315)

[Testing 13](#_Toc122889316)

**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.

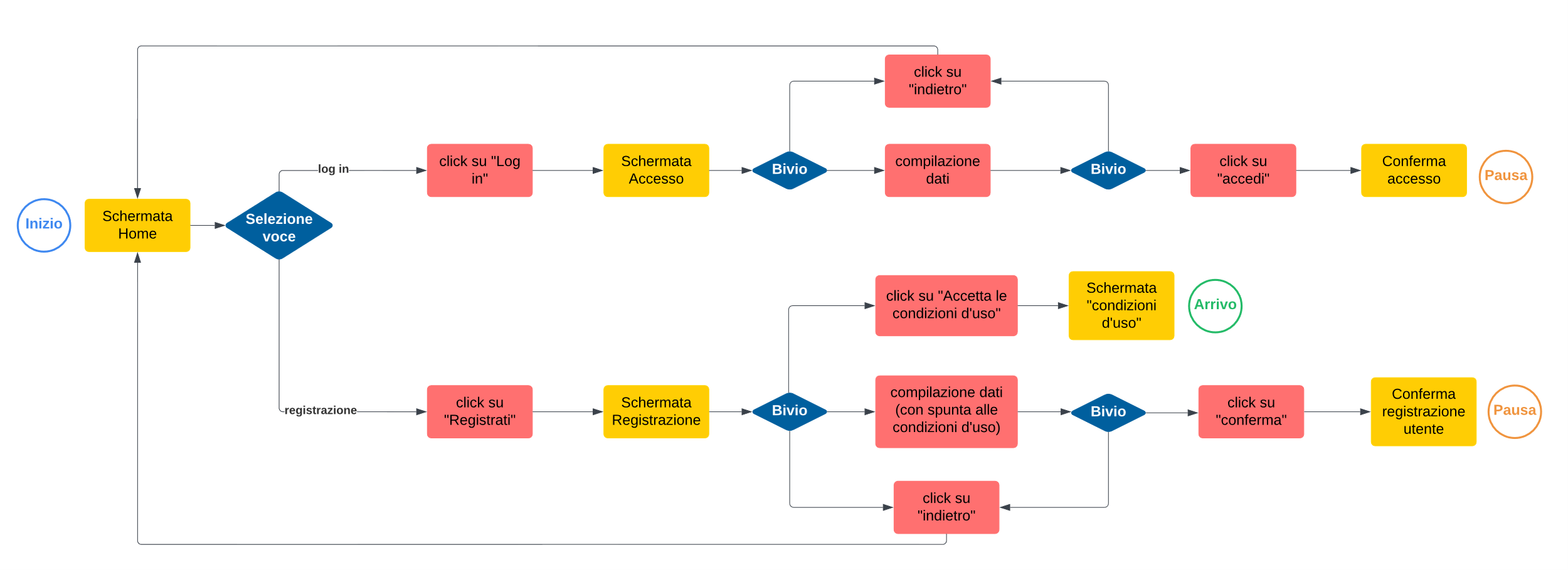
# User flows

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.

## Utente Anonimo

Nel seguente User Flow l’Utente Anonimo, a meno di selezionare la voce “indietro”, tramite la corretta compilazione dei campi richiesti può proseguire con il log-in oppure la registrazione di un nuovo utente.

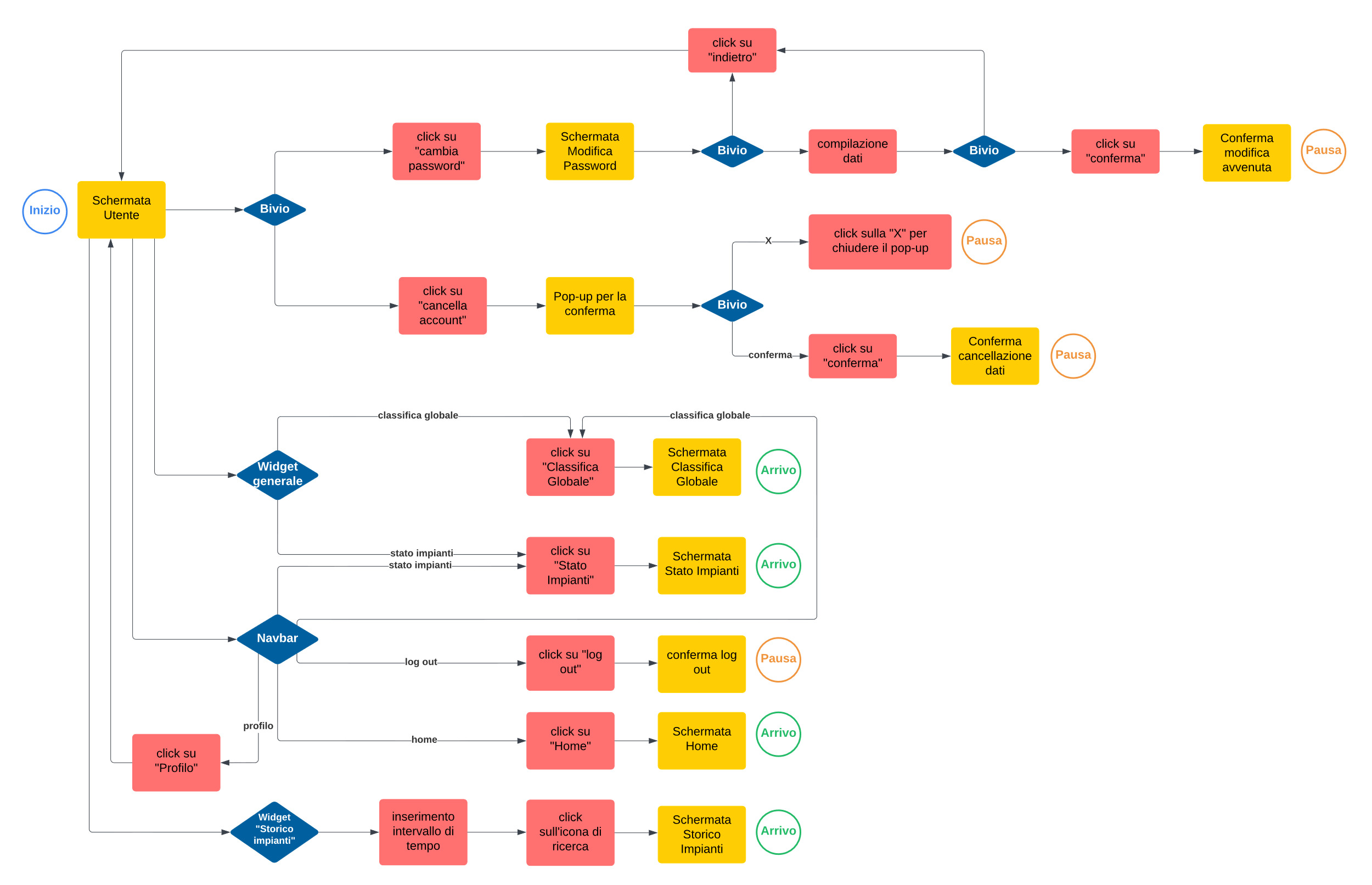


## Utente Registrato

Nel seguente User Flow sono descritte le principali funzionalità a cui può accedere, dalla schermata dedicata alla gestione dell’account, un utente di livello “Registrato”, “Gestore” oppure “Operatore”.

Tramite due bottoni “Cancella Account” e “Cambia Password” può, come suggeriscono i nomi, eliminare definitivamente l’utenza oppure modificare la propria password.

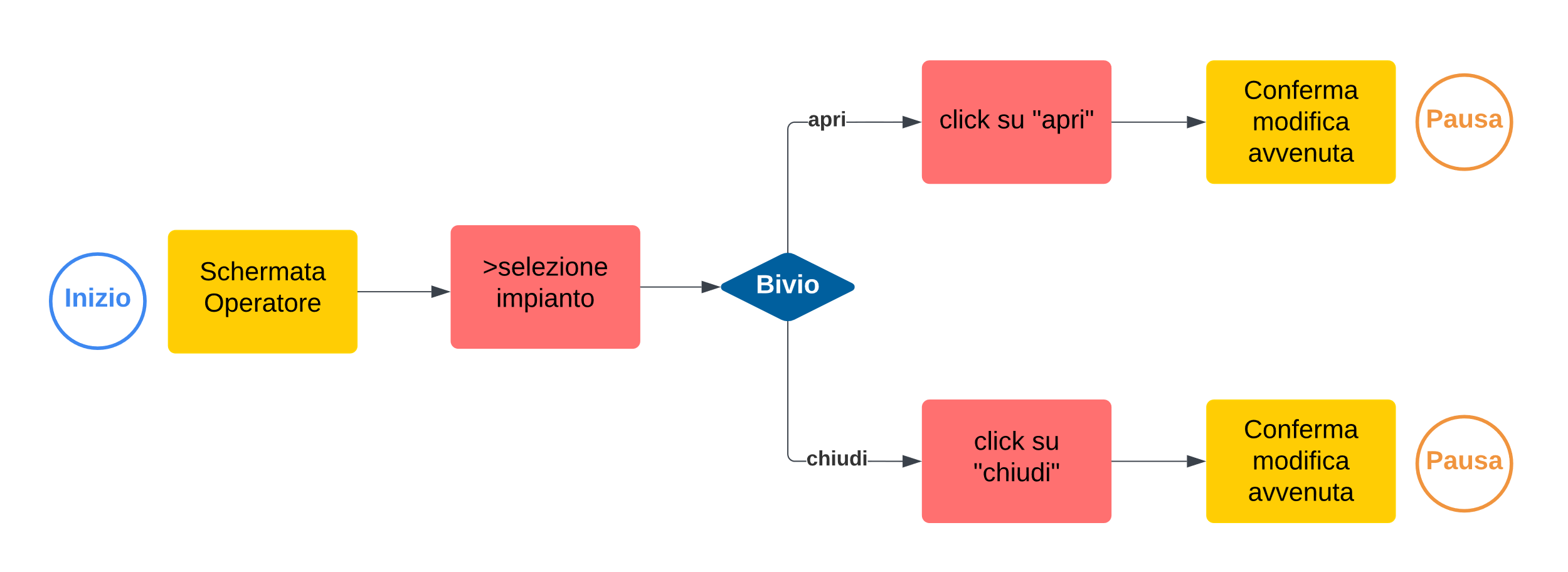
Inoltre, tramite un widget generale può visualizzare il meteo, lo stato degli impianti e la classifica globale. Queste ultime due funzioni sono accessibili anche dalla navbar. In aggiunta, da quest’ultima è possibile effettuare il log-out ed essere reindirizzati alla pagina “Home”.



## Utente Operatore/Gestore

Nel seguente User Flow è descritta una delle funzionalità principali dell’Utente di livello “Operatore” e “Gestore” a cui solo loro possono accedere.

Tramite una schermata apposita l’utente, in seguito alla selezione di un impianto, può modificarne lo stato di apertura o chiusura.



# 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 seguito

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

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

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

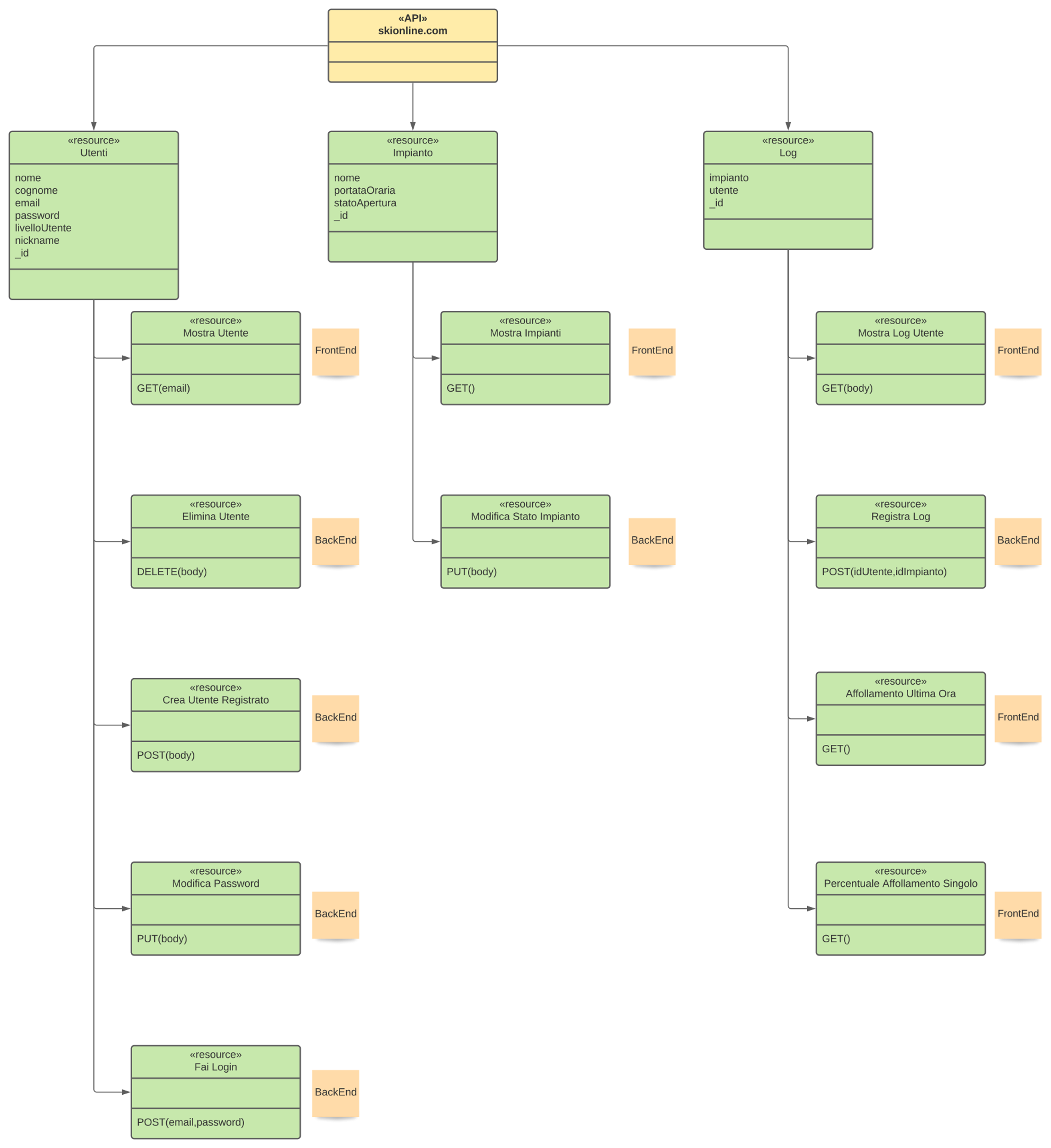
## Project APIs

### Resources extraction from the class diagram

Nel seguente digramma sono presenti le risorse estratte dal Class diagram realizzato nel documento precedente.

A partire dal root del diagramma, dove è presente l’URL del nostro sito skionline.com, si diramano tutte le relazioni che gestiamo a livello di API, ovvero le seguenti tre risorse:

* Utenti;
* Impianto;
* Log.



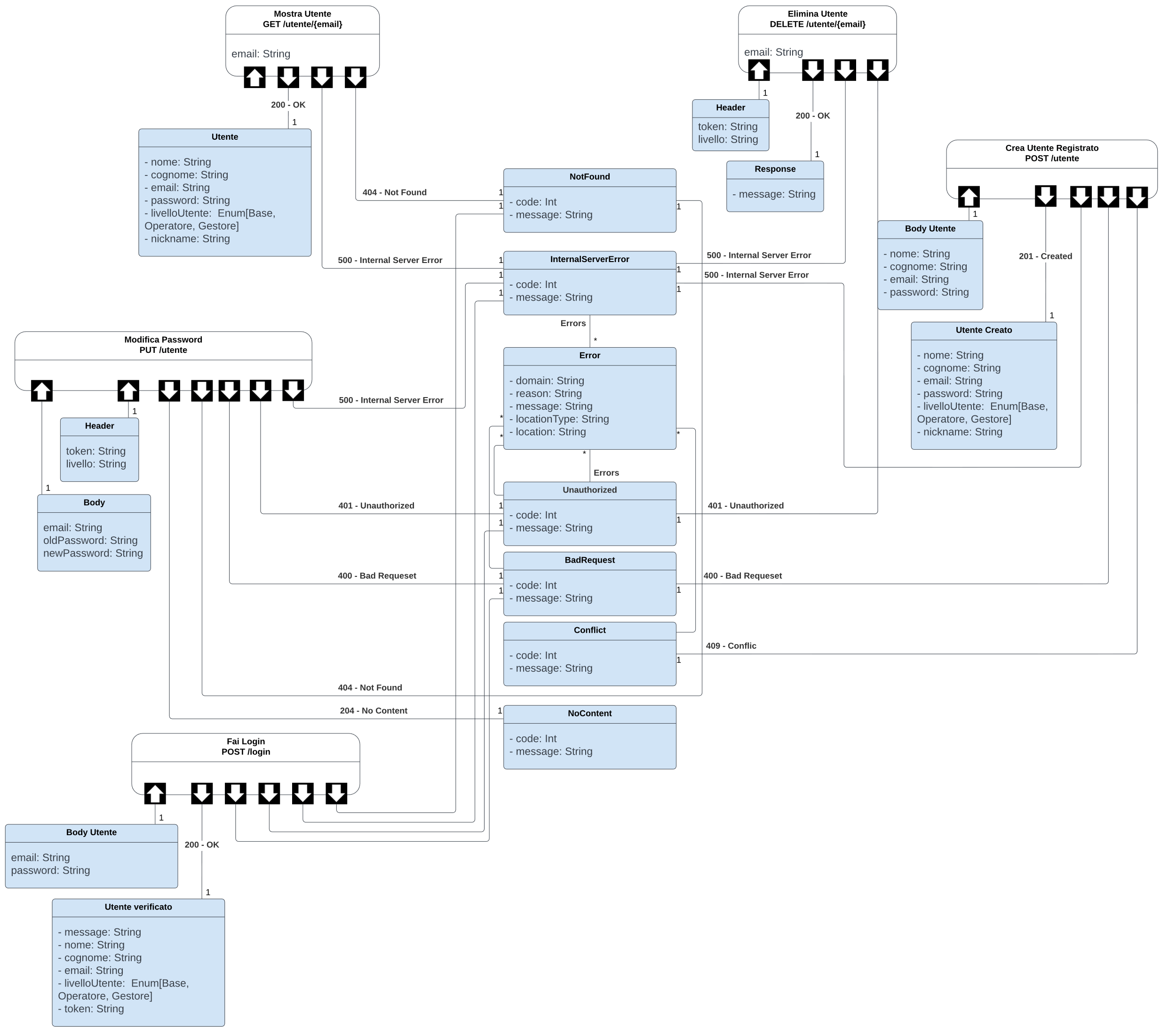
*Nota bene: il parametro \_id indicato nelle risorse si riferisce all’identificativo univoco generato da MongoDB all’atto di registrazione dei dati*

### Resources models

Basandoci sulla sezione precedente sono stati realizzati quattro Resources Models suddivisi per risorsa.

#### Utenti

Nel seguente diagramma sono presenti le risorse precedentemente descritte per “Utenti” attraverso l’API resource. Ognuna, oltre al nome e al metodo, presenta anche l’URI tramite il quale è possibile accedere al servizio specifico.

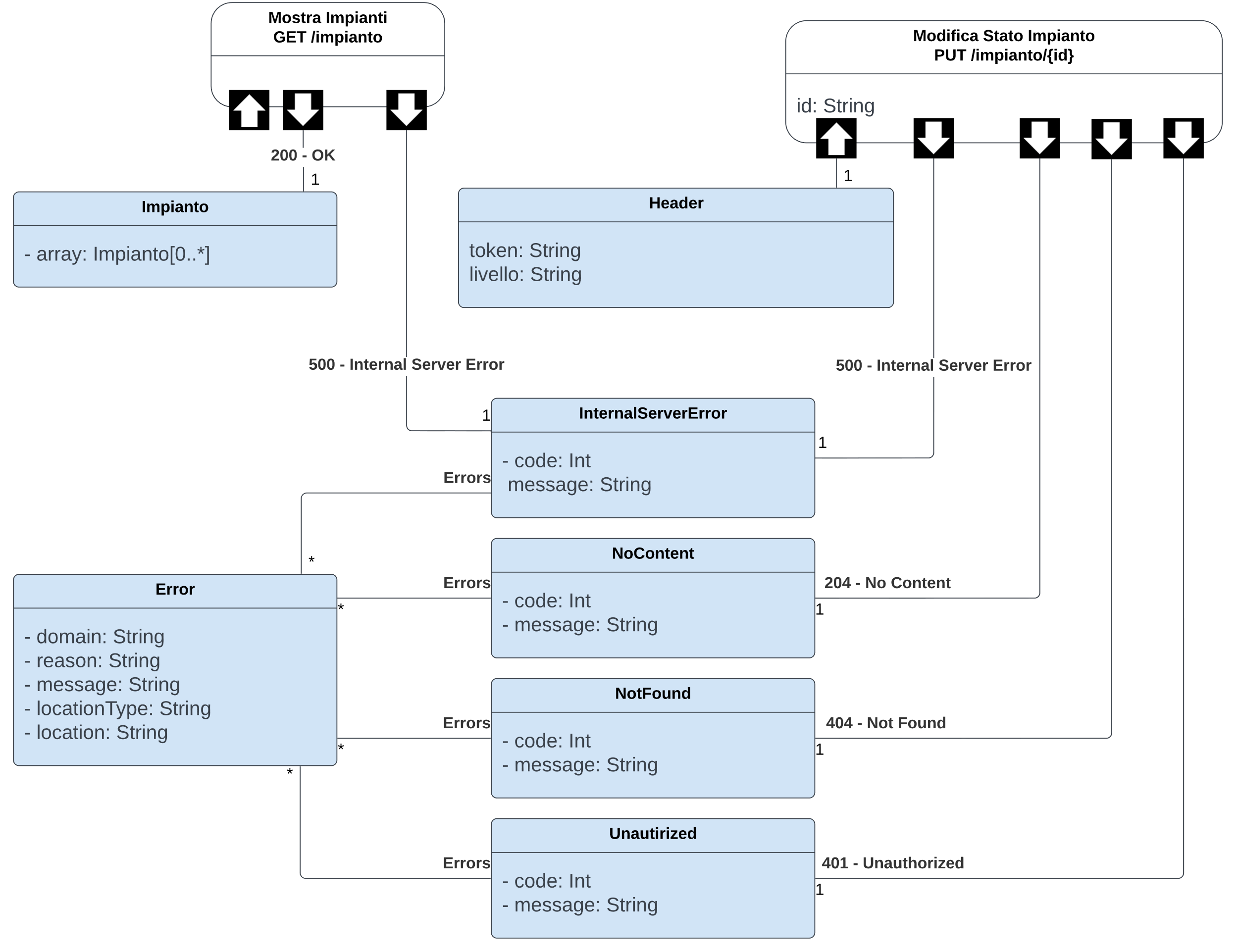


In questo caso abbiamo:

|  |  |  |
| --- | --- | --- |
| Mostra Utente | GET | /utente/{email} |
| Elimina Utente | DELETE | /utente/{email} |
| Crea Utente Registrato | POST | /utente |
| Modifica Password | PUT | /utente |
| Fai Login | POST | /login |

#### Impianti

Nel seguente diagramma sono presenti le risorse precedentemente descritte per “Impianti” attraverso l’API resource. Ognuna, oltre al nome e al metodo, presenta anche l’URI tramite il quale è possibile accedere al servizio specifico.

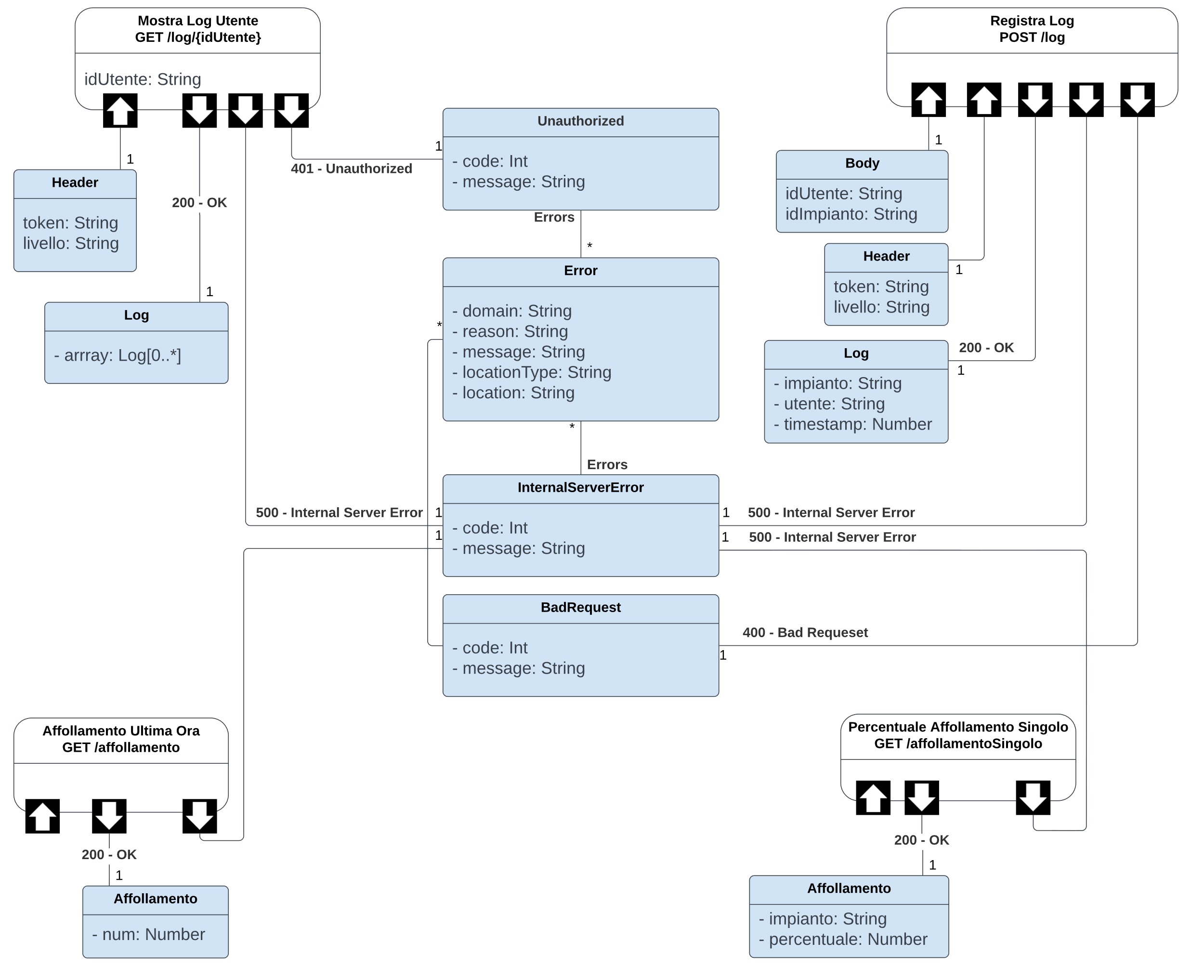


In questo caso abbiamo:

|  |  |  |
| --- | --- | --- |
| Mostra Impianti | GET | /impianto |
| Modifica Stato Impianto | PUT | /impianto/{id} |

#### Log

Nel seguente diagramma sono presenti le risorse precedentemente descritte per “Log” attraverso l’API resource. Ognuna, oltre al nome e al metodo, presenta anche l’URI tramite il quale è possibile accedere al servizio specifico.



In questo caso abbiamo:

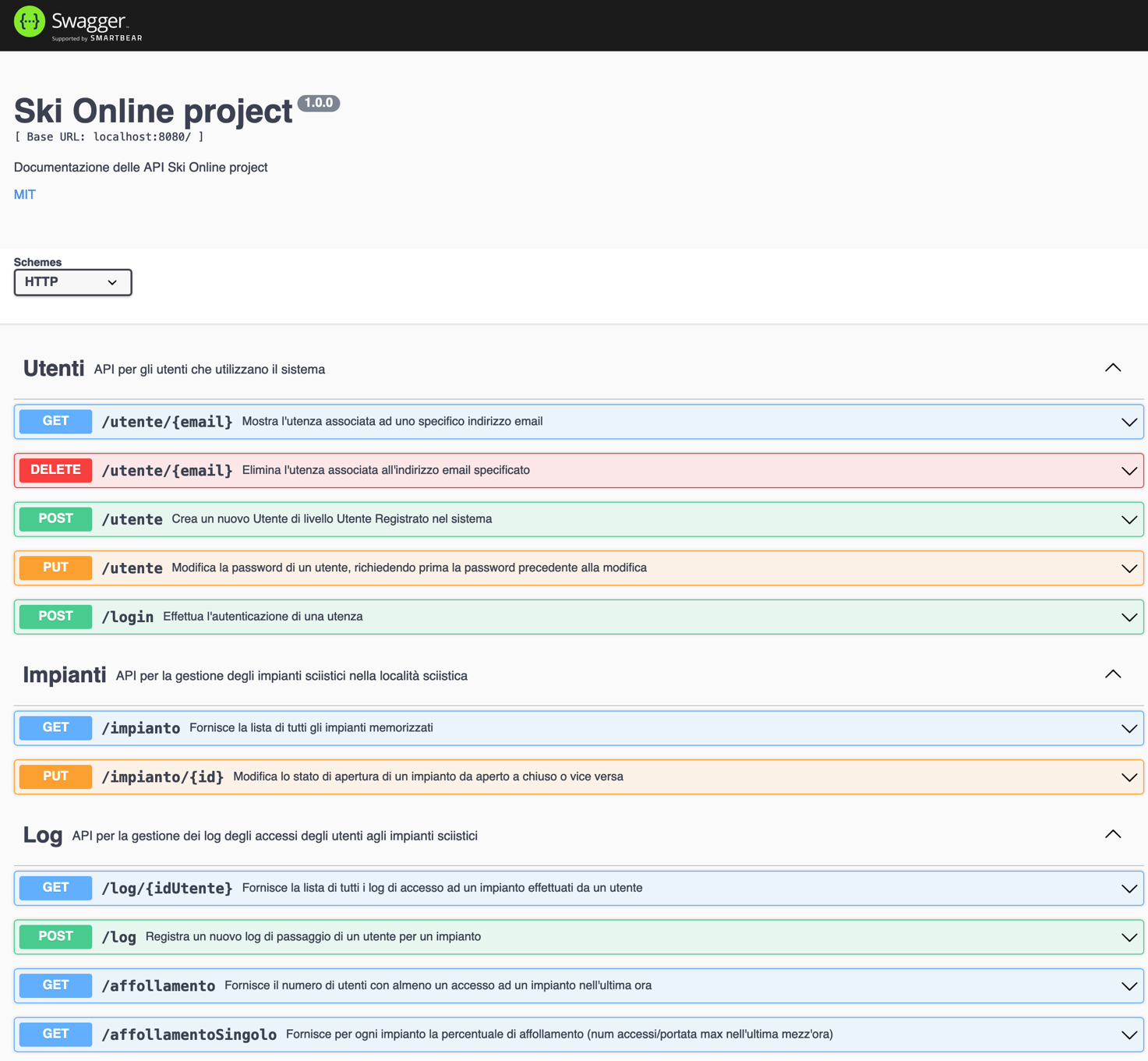
|  |  |  |
| --- | --- | --- |
| Mostra Log Utente | GET | /log/{idUtente} |
| Regista Log | POST | /log |
| Affollamento Ultima Ora | GET | /affollamento |
| Percentuale Affollamento Singolo | GET | /affollamentoSingolo |

## 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