PROGETTAZIONE

Suddivisione in package del sistema:

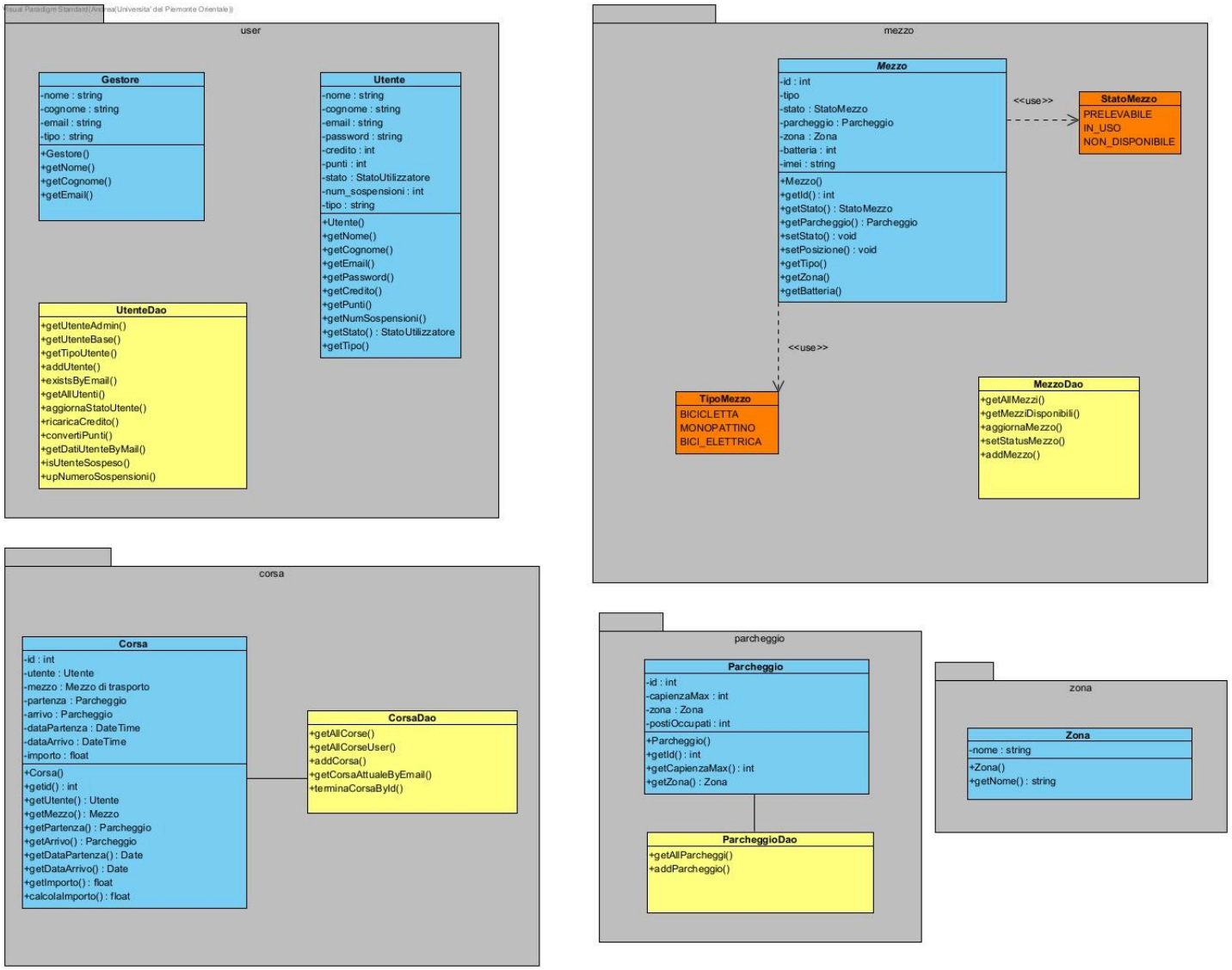
```
Codice
    |__ Client
    |       |__ src
    |               |__ main
    |                       |__ java
    |                       |       |__ it.uniupo.pissir.controller
    |                       |               |__ corsa
    |                       |               |__ gestore
    |                       |               |__ Index
    |                       |               |__ login
    |                       |               |__ logout
    |                       |               |__ mezzo
    |                       |               |__ pagamento
    |                       |               |__ parcheggio
    |                       |               |__ profilo
    |                       |               |__ registrazione
    |                       |               |__ utente
    |                       |
    |                       |__ resources
    |                               |__ public
    |                               |__ templates
    |
    |__ Server
            |__ src
                    |__ main
                    |       |__ java
                    |               |__ it.uniupo.pissir
                    |                       |__ mqtt
                    |                       |
                    |                       |__ oggetti
                    |                               |__ corsa
                    |                               |__ mezzo
                    |                               |__ parcheggio
                    |                               |__ user
                    |                               |__ utils
                    |                               |__ zona
                    |
                    |__ resources
```

Il package “**Interfaccia**” contiene quattro package principali:

- **Client**: contiene gli elementi che si interfacciano direttamente con l’utente, i **controller** dei componenti del sistema e la **Gui**, ovvero l’interfaccia grafica che permette all’utente di interagire con il sistema intero;

- **Server**: contiene, nel package “**Oggetti.src**” i vari oggetti del sistema, con le relative classi che li definiscono e le loro classi DAO che permettono la comunicazione con il database.
  Questo package contine inoltre un classe “service.java” che definisce le varie API REST del sistema;
  Il Database è contenuto all’interno della cartella “**resources**”.

Diagramma delle classi “Interfaccia.Rest.Oggetti.src”:



Utente:

- Gestore.java: definisce l'oggetto gestore;
- Utilizzatore.java: definisce l'oggetto utilizzatore;
- UtenteDao.java: classe che si interfaccia con il DB;

Mezzo:

- Mezzo: definisce l'oggetto mezzo;
- StatoMezzo: (*enum*) definisce lo stato del mezzo {IN_USO, PRELEVABILE, NON_DISPONIBILE};
- TipoMezzo: (*enum*) definisce il tipo di mezzo {BICICLETTA, BICI_ELETTRICA MONOPATTINO};
- MezzoDao: classe che si interfaccia con il DB;

Corsa:

- Corsa: definisce l'oggetto corsa;
- CorsaDao: classe che si interfaccia con il DB;

Parcheggio:

- Parcheggio: definisce l'oggetto parcheggio;
- ParcheggioDao: classe che si interfaccia con il DB;

Zona:

- Zona: definisce l'oggetto parcheggio;
- ZonaDao: classe che si interfaccia con il DB

Descrizione API ed End-Point REST:

Utente:

Metodo HTTP	ENDPOINT	Descrizione	INPUT	OUTPUT	Messaggio Errore
POST	/login	Ricerca un utente specifico	email, password	Stato: 200 Body: Utente	Stato: 404, 401, 500
POST	/register	Aggiunge un nuovo utente	nome, cognome, email, password	Stato: 200 Body: Utente	Stato: 409, 500
GET	/users	Ricerca tutti gli utilizzatori		Stato: 200 Body: ArrayList< Utente>	Stato: 404
POST	/updateUserStatus	Aggiorna i dati di un utente	email, stato, numSospensioni	Stato: 200 Body: Utente	Stato: 400, 404, 500
POST	/updateUtente	Aggiorna il credito e punti degli utilizzatori	email	Stato: 200	Stato: 404
POST	/ricaricaCredito	ricarica il credito dell'utilizzatore	email, importo	Stato: 200	Stato: 400, 404
POST	/conversionePunti	converte i punti in credito	punti, credito	Stato: 200	Stato: 400, 404

Mezzo:

Metodo HTTP	ENDPOINT	Descrizione	INPUT	OUTPUT	Messaggio Errore
GET	/mezziAdmin	Ritorna la lista di tutti i mezzi		Stato: 200 Body: ArrayList<Mezzo>	Stato: 404, 500
POST	/updateMezzo	Aggiorna i dati di un mezzo	datiAggiornamento	Stato: 200 Body: datiAggiornamento	Stato: 400, 404, 500

GET	/mezziUsser	ritorna tutti i mezzi disponibili		Stato: 200 Body: ArrayList<Mezzo>	Stato: 404, 500
POST	/addMezzo	aggiunge nuovo mezzo	Mezzo	Stato: 200	Stato: 404

Parcheggio:

Metodo HTTP	ENDPOINT	Descrizione	INPUT	OUTPUT	Messaggio Errore
GET	/parcheggiAdmin	ritorna tutti i parcheggi		Stato: 200 Body: ArrayList<Parcheggio>	Stato: 404, 500
POST	/addParcheggio	aggiunge un nuovo parcheggio	Parcheggio	Stato: 200	Stato: 404

Corsa:

Metodo HTTP	ENDPOINT	Descrizione	INPUT	OUTPUT	Messaggio Errore
GET	/corseAdmin	Ritorna tutte le corse		Stato: 200 Body: ArrayList<Corsa>	Stato: 404, 500
GET	/corseUser	Ritorna tutte le corse di un utnte	email	Stato: 200 Body: ArrayList<Corsa>	Stato: 404, 500
POST	/corsaAttuale	Ritorna la corsa attuale attiva	email	Stato: 200 Body: Corsa	Stato: 404

POST	/terminaCorsa	Termina la corsa attuale	idCorsa, parcheggio_fine	Stato: 200 Body: ArrayList<Corsa>	Stato: 404

Descrizione Topic MQTT:

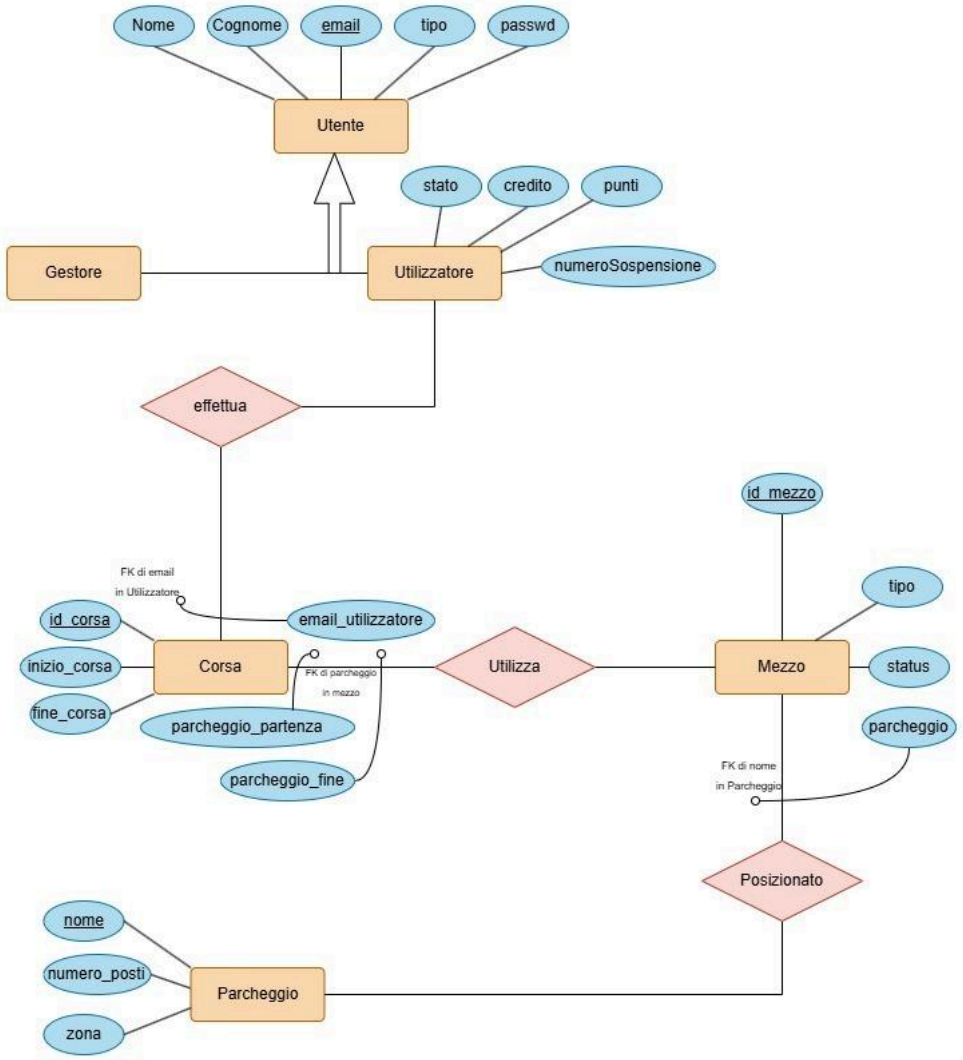
Carica/Scarica Batteria:
mezzo/batteria/{id_mezzo} (da Gestore IoT a Broker) invio l'id del mezzo ed il livello della batteria.

Iscritto a "mezzo/batteria/+"

Cambio Stato:
mezzo/status/{id_mezzo}/{nuovo_stato} (da Gestore IoT a Broker) invio l'id del mezzo ed il nuovo stato da impostare.

Iscritto a "mezzo/status/+/+"

DataBase:



Funzionamento:

Quando un utente desidera usufruire del servizio di mobilità sostenibile, si registra nel sistema e carica del credito sul proprio account. Quando effettua una corsa, il sistema registra l'uso del mezzo e addebita l'importo dovuto tramite la tabella Corsa.
I mezzi sono distribuiti in vari parcheggi, che sono monitorati dalla tabella Parcheggio. Ogni mezzo ha una propria posizione, e questa viene aggiornata ogni volta che il mezzo viene prelevato o restituito in un parcheggio. Se un mezzo presenta un guasto, l'utente può segnalarlo e il gestore del sistema può intervenire tempestivamente per la manutenzione dei mezzi.
Il sistema può anche gestire lo stato degli utenti: se un utente non ha abbastanza credito per completare una corsa, lo stato dell'utente può essere cambiato in "sospeso", impedendo ulteriori utilizzi finché non viene riattivato dal gestore.

Tabella Utente:
La tabella Utente è fondamentale per gestire i dati relativi agli utenti del servizio. Ogni utente è identificato in modo univoco tramite la sua email, garantendo. La tabella contiene anche informazioni personali dell'utente, come il nome, il cognome e password. Inoltre, l'utente ha un credito, che rappresenta il saldo disponibile per usufruire dei mezzi di trasporto, e uno stato, che può essere "attivo" o "sospeso". L'utente può essere sospeso, ad esempio, in caso di credito insufficiente o di altri problemi legati al suo comportamento.

Tabella Parcheggio:
La tabella Parcheggio tiene traccia dei vari punti di parcheggio dei mezzi sparsi nella città. Ogni parcheggio ha un id_parcheggio univoco, un nome, la sua localizzazione (zona) e la capienza massima. Ogni parcheggio può ospitare diversi mezzi di trasporto, che possono essere prelevati o restituiti dagli utenti.

Tabella Mezzo:
La tabella Mezzo è utilizzata per registrare i mezzi di trasporto disponibili per il servizio di sharing, che possono essere biciclette muscolari, biciclette elettriche o monopattini elettrici. Ogni mezzo ha un identificatore univoco id_mezzo, un tipo che indica se è una bicicletta muscolare, elettrica o un monopattino elettrico, e uno stato, che può essere "prelevabile", "non disponibile" o "in uso". Inoltre, la tabella tiene traccia della posizione del mezzo, ossia il parcheggio in cui si trova, utilizzando una chiave esterna che fa riferimento alla tabella Parcheggio. Per i mezzi elettrici, c'è anche un campo batteria che indica lo stato della carica, e un codice IMEI univoco (codice_IMEI) che consente di identificare il mezzo in modo univoco anche al di fuori del sistema.

Tabella Corsa:
La tabella Corsa registra le informazioni relative a ogni corsa effettuata da un utente. Ogni corsa ha un id_corsa univoco e viene associata a un utente (id_utente) e a un mezzo (id_mezzo) tramite chiavi esterne. Inoltre, vengono registrati la data e l'ora di inizio (data_inizio) e di fine (data_fine) della corsa, nonché il costo della corsa, calcolato in base alla durata.