Exchange

ALEX CARAFFI

Sommario

Idea del progetto

L'idea del progetto è quella di creare una Web App che permetta ai suoi utenti registrati di simulare lo spostamento di denaro dal proprio Conto Corrente alla piattaforma per poi, eventualmente fare degli scambi di valuta da euro a dollari e viceversa.

Il software, quando riceve la richiesta di effettuare il cambio euro-dollari (o dollari-euro), contatta un servizio della BCE che offre un file XML contenente i tassi di cambio delle maggiori monete mondiali aggiornato in tempo reale. Il link del file è raggiungibile qui.

Un utente può registrarsi liberamente alla piattaforma inserendo nome, cognome, indirizzo email, password (quest'ultima ovviamente verrà cifrata per ragioni di sicurezza) e IBAN. Se per sbaglio l'utente dovesse inserire una email o un IBAN che sono già stati registrati, verrà segnalato il problema e l'utente dovrà cambiare questi campi.

Gli utenti inoltre possono impostare la loro foto di profilo scegliendo un qualsiasi file di tipo immagine (.jpg, .jpeg, .png, .gif) presente nei loro dispositivi. Al momento della registrazione, verrà assegnata di default un'immagine standard. In questo modo, si cerca di rendere l'esperienza dell'utente più piacevole e personalizzata.

Oltre agli utenti normali, è presente un amministratore che ha libero accesso a tutti i campi del database e ha la possibilità eventualmente di aggiungere, eliminare o modificare ogni informazione.

Durante lo sviluppo del codice, sono stati implementati una serie di test per controllare la qualità e la correttezza di esso che saranno trattati più avanti.

Principali tecnologie usate

ReactJS

ReactJS è una libreria open-source ed è stata usata per la realizzazione del front-end. Essa è basata sull'uso di JavaScript unito ad HTML per la creazione di interfacce utente altamente dinamiche. Inoltre, lavora a stretto contatto anche con CSS, per la parte prettamente grafica.

Alla base di React ci sono i componenti che sono un insieme di elementi HTML strettamente uniti da JS. L'unione di più componenti (e il riuso di essi adattandoli alle circostanze), crea la pagina web. Una funzionalità piuttosto peculiare è che quando si passa da una pagina all'altra della web app, essa non fa alcuna chiamata al server (non viene ricaricata) ma viene tutto gestito da React, accelerando notevolmente le operazioni.

Django

Per il back-end, l'applicativo è stato sviluppato completamente in Django. Tramite le sue tecnologie native, sono facilmente gestibili degli endpoint su cui fare delle richieste Http ed è facilmente creabile un database in cui inserire delle tabelle e fare delle query. Inoltre, Django mette a disposizione un super utente in grado di poter gestire l'intero applicativo da browser.

SQLite

Il database, per questioni di portabilità, è realizzato in SQLite. Oltre alle tabelle necessarie a Django, ha anche le relazioni per memorizzare gli utenti e le transizioni di essi. Tramite delle query, si riesce a ricavare facilmente le disponibilità economiche di ogni singolo utente, la foto profilo e i precedenti scambi di denaro. Nel database, per maggiore sicurezza, sono presenti vincoli molto forti sui dati, per esempio, le password devono essere lunghe 128 caratteri (perché sono state cifrate da una funzione di sha3-512) e che i valori non possono essere vuoti.

Framework Rest

Essendo back-end e front-end separati per simulare al meglio le applicazioni reali, la comunicazione tra i due avviene tramite delle chiamate Restful. Il back-end espone delle API su determinati endpoint raggiungibili solo attraverso determinati metodi del protocollo Http (tipicamente POST e OPTION). Il client invia un oggetto Json contenente i dati necessari per svolgere l'operazione e il server, dopo che ha svolto le sue operazioni, restituisce anch'esso un oggetto Json che verrà letto dal client il quale eseguirà le istruzioni al suo interno.

Json Web Token (JWT)

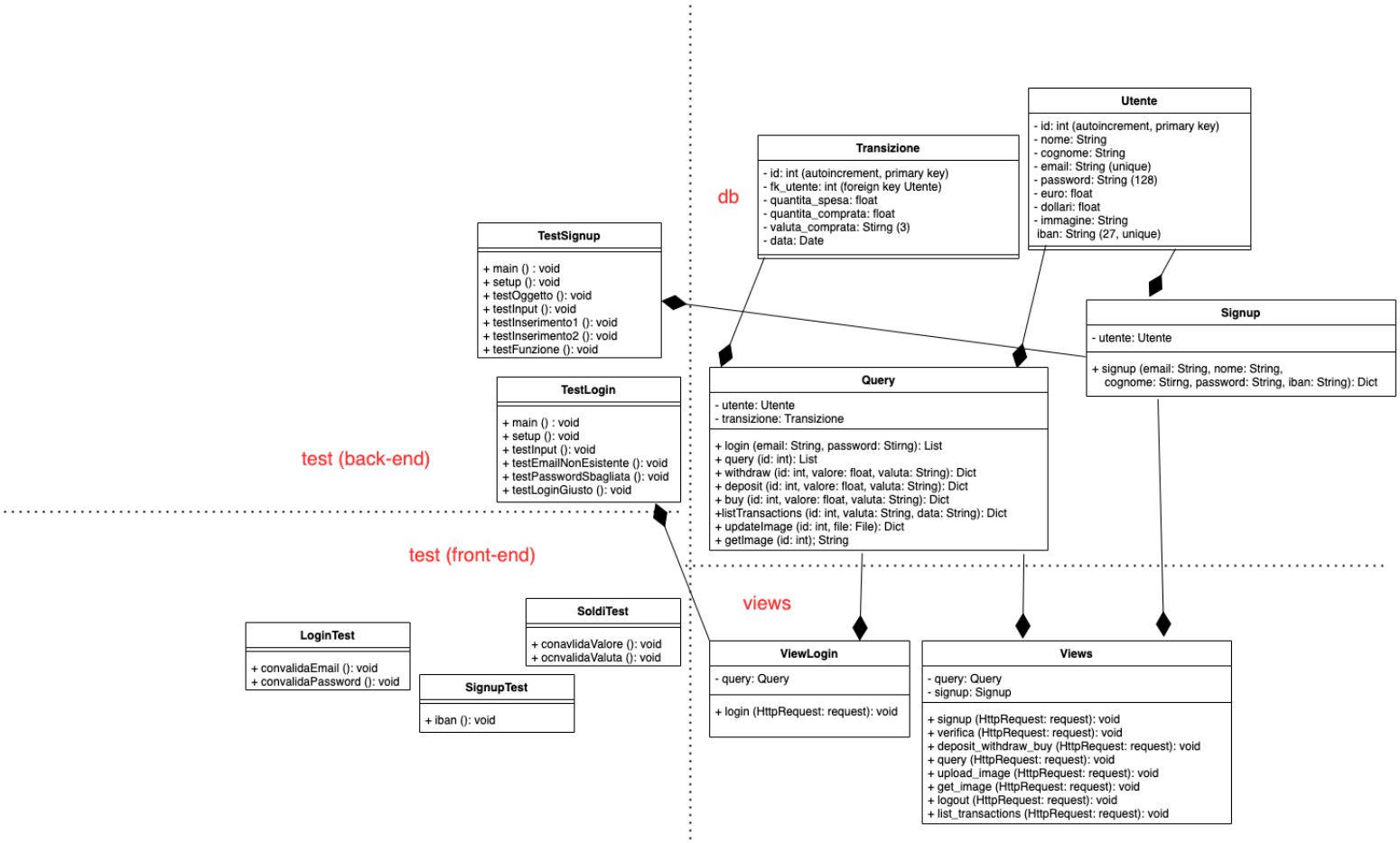
La necessità di usare il Json Web Token è data dall'esigenza di avere un login il più sicuro possibile. Quando le credenziali vengono verificate viene creato un token tramite l'algoritmo di cifratura HS256 che incapsula l'id dell'utente appena autenticato e la data attuale. Al momento della creazione, inoltre si specifica che il token sarà valido solo per 15 minuti, una volta scaduto questo tempo, l'utente dovrà autenticarsi nuovamente per avere un token aggiornato.

Il token viene salvato come cookie in modalità HttpOnly, in questo modo nessuno script eseguito in locale o nel browser può avere accesso a questo tipo di dato, solo il server potrà leggerlo e modificarlo.

Per accedere a tutte le API (tranne quella di login e signup), sarà necessario essere in possesso del token valido, altrimenti si verrà re direzionati sulla pagina di login.

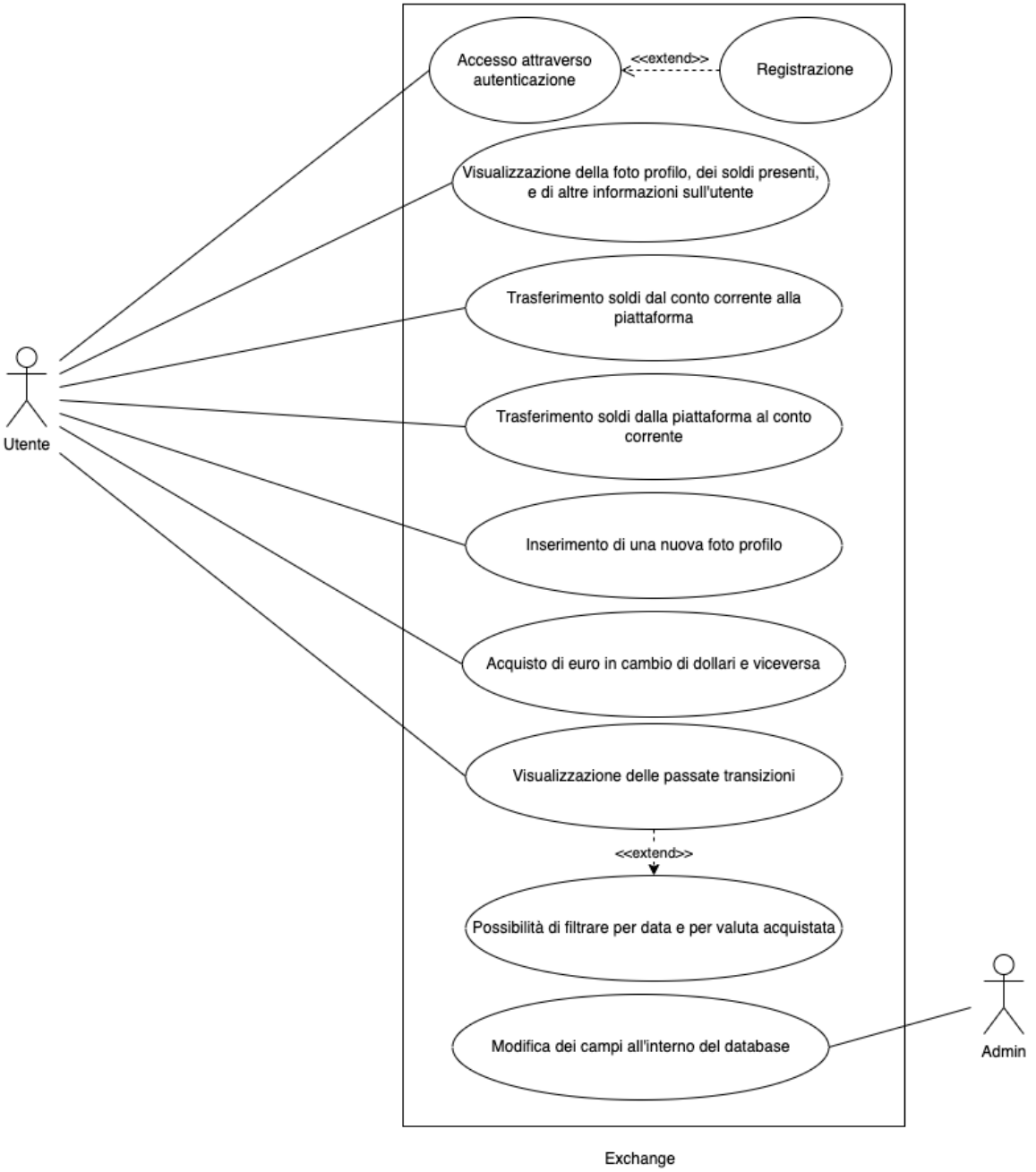
Descrizione dell'applicazione

Diagramma UML



La figura rappresenta il diagramma UML delle classi e delle funzioni usate. Si è diviso il tutto in quattro parti: test (back-end), test (front-end), views, db. I test del back-end sono presenti nella cartella root del back-end mentre i test del front-end sono in src/test. Nella parte delle views sono stati creati due file che contengono tutte le funzioni da eseguire quando il front-end chiama il back-end. Si è optato per tenere tutto su un unico file (a parte il login, su cui sono stati effettuati i test) per semplicità. Mentre nell'applicazione db sono presenti i modelli delle tabelle nel database Utente e Transizione con due file per eseguire le query necessarie al funzionamento del progetto. Anche in questo caso si è optato per semplicità l'uso di un unico file tranne per la signup su cui vengono effettuati altri test.

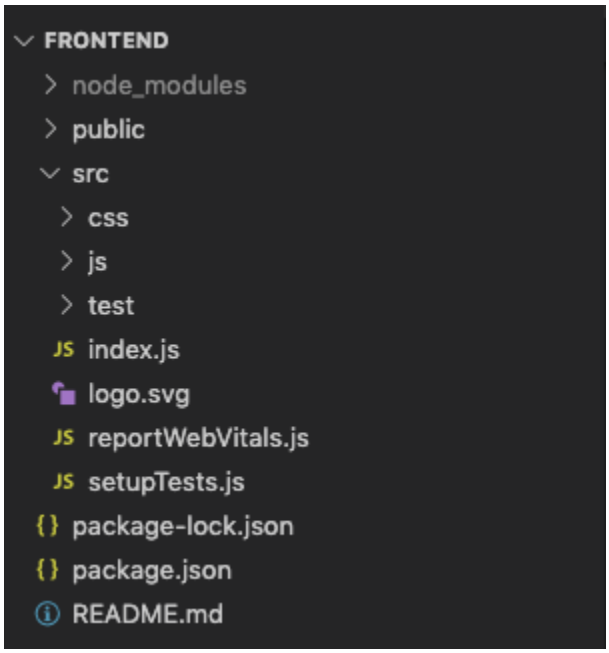
Diagramma dei Casi d'Uso



La precedente figura rappresenta il diagramma dei casi d'uso. L'utente, una volta effettuata la registrazione, potrà accedere tramite un login al suo portafoglio virtuale in cui può vedere la sua foto profilo e la sua situazione economica, mostrando la quantità di euro e dollari presenti nel suo profilo. In seguito, può a piacimento fare dei versamenti di denaro sul suo conto oppure fare dei prelievi, specificando la valuta e la quantità scelta di denaro. Può inoltre effettuare il cambio valuta in base al tasso attuale e può vedere le sue precedenti transizioni le quali potranno essere filtrate per data o per valuta comprata. Infine, è presente l'utente admin che può accedere alla pagina web di Django in cui può liberamente visualizzare, modificare ed eliminare delle tuple del database.

Organizzazione logica dell’applicazione

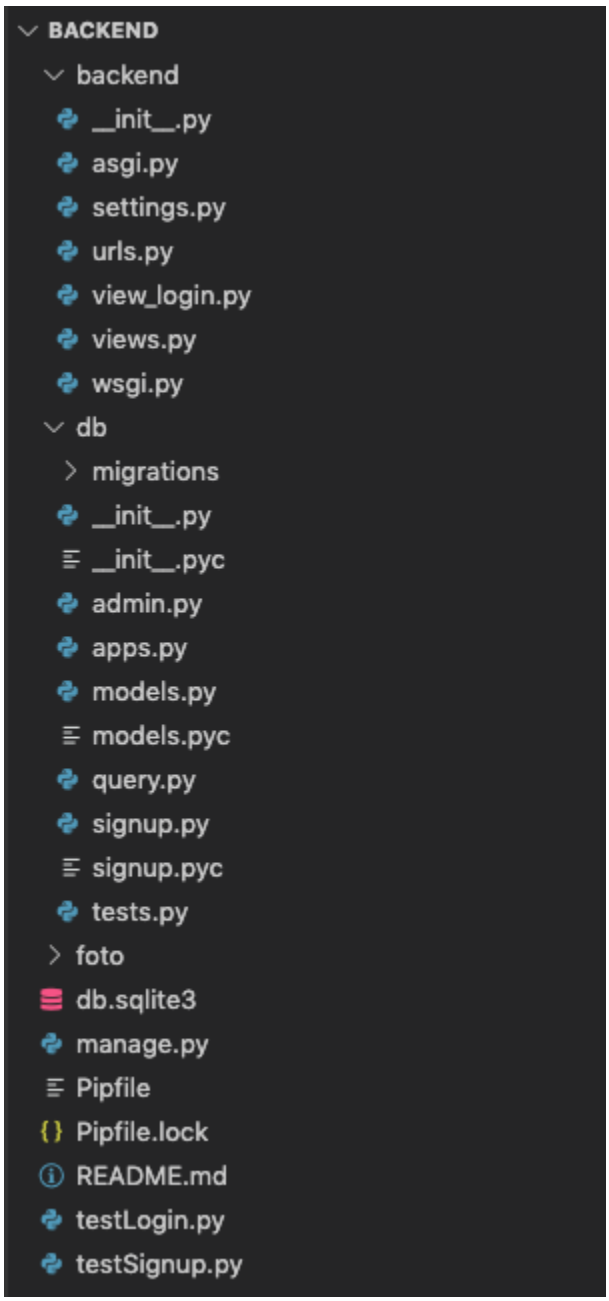
Front-end



La precedente, rappresenta la cartella principale del front-end. Nella cartella node_modules (che sarà presente solo dopo aver eseguito il comando `npm install`) sono presenti tutte le dipendenze di React e le funzioni esterne (come, per esempio, quella che cifra la password); nella cartella public sono presenti tutti i file statici essenziali del progetto come il logo e nella cartella src sono presenti i file di costruzione della web app. I file package.json servono per tenere segnate tutte le dipendenze dell’applicazione. È stata fatta questa scelta di organizzazione perché è la più semplice e immediata possibile, anche per un futuro ampliamento dell’applicazione.

Nel file README.md sono presenti tutte le istruzioni per lanciare il front-end

Back-end



Nella cartella principale sono presenti due applicazioni: backend che gestisce il cuore del server e db che gestisce i modelli e le query al database. Nella root sono anche presenti i test, il database e i Pipfile per installare le dipendenze. In backend sono presenti tutti gli endpoint del server nel file urls.py a cui sono associate le relative funzioni nel file views. In db invece ci sono i modelli Utente e Transizione nel file models.py, mentre nel file query.py è presente una classe che ha come metodi, tutte le query necessarie.

Per future implementazioni e aggiunte, verranno separate le varie funzioni in diversi file per migliorare la struttura del progetto e per renderlo più scalabile.

Nel file README.md sono presenti tutte le istruzioni per lanciare il back-end

Test eseguiti

Front-end

All'interno della cartella src/test del front-end sono presenti quattro file test: Login.test.js, Signup.test.js e Soldi.test.js. Nel file del login, controllo che la stringa che si sta inserendo sia effettivamente un indirizzo email (deve quindi contenere il carattere @) e che la password appena cifrata sia di 128 caratteri.

Nel file di signup si controlla solo che l'iban sia di 27 caratteri (perché la password e l'email vengono già controllate dalla stessa funzione che li controlla nel login).

Nel file Soldi.test.js invece si controllano gli input di quando si vuol fare uno spostamento di denaro. Si controlla che il denaro che si vuole prelevare non sia un numero minore di zero o maggiore del massimo prelevabile (tranne nel caso di quando si vuole caricare il saldo, li non c'è limite) e che la valuta possa assumere solo i valori EUR o USD.

Nell'ultimo file, infine, si controlla che la data presa in input quando si vogliono filtrare le transizioni sia effettivamente una data presa nel formato corretto.

In questo modo, vengono quindi controllati tutti i valori dei form della web app

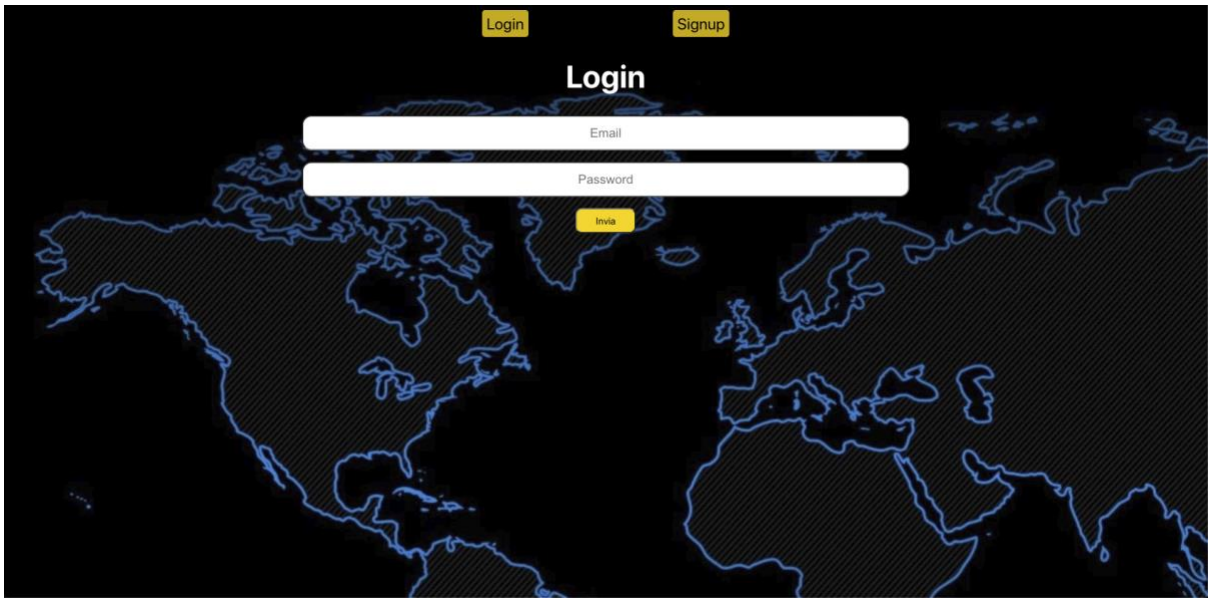
Back-end

Nel back-end sono presenti due file per i test Unit di Django: testLogin.py e testSignup.py. Nel test del login si controlla la correttezza dell'endpoint del login in generale, ovvero si controllano i metodi della richiesta (che possono solo essere POST o OPTION per il preflight), il corpo della richiesta che deve rispettare certe specifiche (deve essere un oggetto Json che ha i campi login e password). Bisogna inoltre controllare la risposta del database in base agli input, ovvero il login risulta effettivamente riuscito solo se esiste una tupla che abbia email e password uguali a quelle richieste. In ogni altro caso, si otterrà un errore che verrà segnalato all'utente.

Il test sulla signup è invece incentrato sul database: si controllano gli input (l'iban deve essere di 27 caratteri, la password di 128 e l'email deve contenere il carattere @) e i risultati ricevuti dal database quando si prova ad inserire una email o un iban già presente, cosa che violerebbe il principio Unique di questi campi.

Risultati finali

La prima volta che si apre l'applicazione ci si trova davanti alla schermata di login.



Una volta effettuato l'accesso (o la signup), si viene reindirizzati sulla pagina principale.

Da qui si può andare in tutte le sezioni dell'applicativo, effettuare il logout ed eventualmente cambiare la foto profilo. Nello storico troviamo tutte le passate transizioni filtrabili per data e/o per valuta comprata.


Home
Carica
Deposito
Converti
Storico
Logout
2.96 €
49.01 $
Storico delle transizioni
Quantita Spesa
Quantita Comprata
Data
1.00 $
0.95 €
19-06-2022
2.00 €
2.10 $
19-06-2022
3.10 $
2.96 €
19-06-2022
1.00 $
0.95 €
19-06-2022
2.00 $
1.90 €
21-06-2022
3.00 $
2.85 €
23-06-2022
4.04 $
3.84 €
23-06-2022
15.15 €
15.94 $
23-06-2022
1.00 $
0.95 €
25-06-2022
Filtri
gg/mm/aaaa
Nessuna
Invia

Problemi e migliorie future

Il problema principale è stato far comunicare il client con il server: settare correttamente tutti gli url, gli endpoint, le views e le chiamate da parte del client ma soprattutto rispettare le politiche di CORS e le chiamate Restful. Il bug principale è stato il capire che quando il client chiama l'endpoint viene inviata una richiesta con il metodo OPTION all'url indicato che deve essere necessariamente gestita in modo diverso dal server e che senza la quale la comunicazione effettiva non può avvenire. Inoltre, è stato anche necessario settare il client in modo che potesse ricevere sia delle risposte in formato Json, sia dei cookie salvati nella risposta del server e che serviranno successivamente per garantire l'avvenuta autenticazione.

Migliorare la politica del JWT in quanto per ora ha un tempo di vita indipendente dal fatto che l'utente usi l'applicazione o meno, questo implica che non viene ricaricato automaticamente se l'utente fa delle operazioni che riguardano il server. Inoltre, quando l'utente effettua il logout, il token rimane usabile, nonostante venga cancellato dai cookie.

Applicare ulteriori controlli sia sul back-end sia sul front-end per evitare SQL Injection.

Rendere più scalabile il back-end, dividendo le views su più file e app.