Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea di Informatica per il Management

PROGETTAZIONE E SVILUPPO DI UN SOFTWARE DI SUPPORTO ALLA E-MOBILITY

Relatore:
Chiar.mo Prof.
MARCO DI FELICE

Correlatore:
FABRIZIO CRINÒ

Presentata da:
ALESSIA VASALLUCCIO

Sessione III
Anno Accademico 2019-2020

Alla mia famiglia...

Indice

Elenco delle figure

Introduzione

Uno degli oggetti più utilizzati in tutto il mondo oggi è l'automobile. Mezzo indispensabile se si vuole andare da una parte all' altra della città in poco tempo o da una parte all' altra della nazione. Ad oggi la maggior parte dei mezzi utilizzati, che siano pubblici o privati, viene alimentato grazie alla benzina o al diesel, due combustibili fossili esauribili e molto inquinanti per l' ambiente. Per questo nel mondo si sta cercando di sensibilizzare ogni giorno di più lo sviluppo della mobilità elettrica a emissioni zero.
In alcuni paesi, come ad esempio l'Europa, sono state varate alcune norme per rispettare l'ambiente e cercare di ridurre l'effetto serra, tra cui quella di limitare le emissioni di CO2. Un obbiettivo che l' Europa cercherà di raggiungere entro il 2030.

Attorno alla mobilità elettrica però ci sono vari approcci di pensiero e diversi studi al riguardo. Da chi pensa che l'auto green sia la soluzione migliore per il rispetto del pianeta a chi pensa che la produzione di auto elettriche sia più inquinante rispetto alla produzione di auto tradizionali, fino a chi non vuole acquistarle perché i tratti di strada percorribili sono relativamente brevi e i punti di ricarica sono troppo pochi per permettersi totale autonomia. É proprio questa la criticità maggiore della mobilità ecosostenibile, il sistema di distribuzione delle stazioni di colonnine elettriche.
In Italia infatti non c'è un' infrastruttura capillare che consenta all' automobilista di viaggiare in totale sicurezza e non rischiare di rimanere senza energia per alimentare la macchina. Nonostante sia un sistema poco sviluppato esistono vari software che gestiscono le poche stazioni esistenti. In questo elaborato ne verranno presentate due in particolare oltre a quella soggetto della tesi.

Fulcro della tesi quindi è ciò che riguarda le colonnine elettriche e in particolare lo sviluppo di un software di gestione di esse, sviluppato nel periodo di tirocinio da me svolto presso il gruppo FINMATICA S.P.A.
Altri protagonisti dell' applicazione creata sono l'ente che ha commissionato il progetto al gruppo, la Comunità solare dell' Emilia-Romagna e l'azienda Nrg4You di Napoli alla quale essa si appoggia per fornire il servizio di ricarica ai membri della Comunità.

Nello specifico l' applicazione consiste nella gestione del database, dove vengono inseriti gli utenti aderenti al servizio e nella gestione di attivazione delle colonnine.
Per quanto riguarda il database, per mezzo di tale software è quindi possibile aggiungere, visualizzare, apportare modifiche ed eliminare un abbonato al servizio. Mentre per quanto riguarda le colonnine esso permette la ricarica del veicolo, avviata dal servizio Restation di Nrg4You. Solo dopo aver verificato tutti i requisiti, come ad esempio l'esistenza di una tessera RFID con un codice appartenente a qualcuno presente nel DB e la scadenza di tale tessera, requisiti necessari per poter abilitare la ricarica, quest' ultima viene avviata. Tale progetto ha lo scopo di promuovere la mobilità elettrica nel territorio dell' Emilia-Romagna, in particolare nella zona di Bologna.

Di seguito è riportato un breve sunto del contenuto di ogni capitolo dell'elaborato.

Capitolo 1: Stato dell' arte
In questo capitolo viene descritto il contesto attorno a cui si muove la tesi.
In particolare viene trattato della situazione della auto elettriche in Italia, delle relative colonnine o dispositivi per la ricarica del veicolo ed i problemi ad essi connessi.
Cosa sono, come funzionano e lo sviluppo di tali colonnine nel tempo. Vengono inoltre descritti dei tools già esistenti per la gestione delle stazioni di ricarica.
Un altro aspetto fondamentale riportato è la descrizione del contesto attorno al quale si è sviluppato il software e cioè la Comunità Solare dell' Emilia-Romagna e il loro progetto in ambito green mobility.

Capitolo 2: Progettazione
In questo capitolo viene inizialmente descritto il contesto di lavoro in cui ho svolto il progetto. In seguito viene riportata una descrizione generale dell' applicazione e i requisiti richiesti dalla Comunità Solare per lo sviluppo del software.
In fine vengono descritte le varie tecnologie di cui mi sono servita per l' implementazione del progetto.

Capitolo 3: Implementazione
Nella parte di implementazione vengono riportate le funzionalità implementate per la realizzazione del software, sia la parte a lato server che la parte a lato client.
Sono stati riportati inoltre degli screen contenenti il codice implementato con la corrispondente spiegazione delle funzionalità svolte.

Capitolo 4: Validazione
Nel capitolo della validazione viene descritto come sono stati svolti i test di verifica del funzionamento dei metodi implementati. In particolare sono stati riportati alcuni esempi illustrativi al fine di presentare il metodo di validazione delle richieste http implementate.

Capitolo 5: Conclusioni & sviluppi futuri
In quest'ultimo capitolo ho riportato brevemente un riassunto di ciò che è stato descritto nei capitoli precedenti facendone il punto del discorso, di quali possono essere i sviluppi futuri del software e della situazione della mobilità elettrica e in particolare delle colonnine elettriche nel mondo.

Capitolo 1

Stato dell'arte

Le auto elettriche sono il futuro?

Quello delle auto elettriche è un settore sempre di più in crescita negli ultimi anni. In particolare in quest'ultimo, in cui le vendite di tali veicoli sono aumentate notevolmente in tutta Europa e ancora di più in Italia.
Infatti, secondo le analisi dell'Associazione europea dei costruttori automobilistici sono state costruite e vendute il triplo delle auto ad emissione zero rispetto agli anni precedenti. Ciò è stato reso possibile anche grazie agli incentivi dati da molti Stati per supportarne l'acquisto, visto che il costo di queste tipo di autovetture non è molto economico e quindi poco accessibile alla maggior parte della popolazione[1].
Per andare incontro agli automobilisti che vorrebbero contribuire alla riduzione dell'inquinamento del pianeta acquistando un veicolo elettrico, ma che non possono permetterselo, nel 2019 sono stati presi due provvedimenti chiamati Ecobonus ed Ecotassa, il cui scopo è quello di incentivare l'acquisto di nuove auto che abbiano un occhio di riguardo verso l'ambiente, che siano cioè a basso impatto ambientale.
Tali provvedimenti comportano l'agevolazione degli acquirenti che hanno acquistato o acquisteranno una nuova auto nel periodo che va dal 1° marzo 2019 al 31 dicembre 2021. In particolare l' Ecotassa prevede il pagamento di una tassa in base alla quantità di emissioni di CO2 per chilometro emessa dall'autovettura. Mentre l'Ecobonus consiste nello sconto sul prezzo d'acquisto di un'auto di nuova fabbricazione a patto che la casa costruttrice rilasci la dichiarazione di conformità all'omologazione dichiarata[2].

Oltre allo Stato, hanno contribuito anche le case produttrici di tali veicoli a questa iniziativa. Infatti queste ultime hanno applicato particolari promozioni per l'incentivazione dell'acquisto di auto elettriche, sia per pubblicizzare il loro marchio sia per cercare di rispettare la normativa Europea, il cui obbiettivo è quello di riuscire a mantenere le emissioni di CO2 entro il limite stabilito. L'unione Europea conta di raggiungere questo obbiettivo entro il 2030[1].

Figura 1.1: Auto Green

Ma qual è il motivo per cui l'Europa si è posta come obbiettivo quello di ridurre le emissioni di CO2 ed incentivare la mobilità elettrica?

La ragione principale di questo desiderio barra obbiettivo è spiegata dalla necessità di dover ridurre la diffusione di anidride carbonica nell'aria. Tale diffusione è causata in quantità molto elevata dai mezzi di trasporto, sia quelli di tipo stradale che di tipo ferroviario, aereo e marittimo[3].
Come tutti sappiamo il problema dell'inquinamento e in particolare dell'effetto serra è causato per la maggior parte dalla messa in circolazione di CO2, una componente molto importante dell'atmosfera terrestre.
Tutto ciò che rappresenta l'avanzamento e il progresso dell'uomo sulla terra come l'industrializzazione, la deforestazione e l'urbanizzazione, rappresenta sia un gran passo per l'umanità, ma anche una minaccia per il nostro pianeta.
L'aumento dei gas serra infatti sta danneggiando sempre di più la terra causando gravi danni come, l'aumento delle temperature, lo scioglimento dei ghiacciai, l' estinzione di alcune specie animali, l'innalzamento del livello del mare e molte altre conseguenze dannose sia per il pianeta che per chi lo popola[4].
Per questo motivo L'Europa e non solo sta cercando di porre dei rimedi ai danni causati dall'uomo e uno di questi potrebbe essere proprio quello di incentivare maggiormente la mobilità elettrica, non solo dal punto di vista del trasporto del singolo cittadino ma anche dal punto di vista del trasporto pubblico.

Ad oggi ci sono vari punti di vista riguardanti l'utilizzo e la produzione delle macchine elettriche. Per quanto riguarda la produzione, quest'ultima non è ritenuta proprio a emissioni zero a differenza del prodotto finale.
Infatti la produzione della macchina e dei suoi componenti comporta una emissione di CO2 quasi superiore rispetto alla produzione di macchine che utilizzano altri combustibili per l'alimentazione. Nonostante si hanno opinioni divergenti sull'utilizzo e la produzione di auto elettriche sembra che comunque l'impatto ambientale totale di queste ultime sia minore rispetto alle auto tradizionali.
Per quanto riguarda l'utilizzo invece possiamo trovare sia dei punti a favore che dei punti a sfavore.

Valutiamo i pro. Come affermato precedentemente chi compra un'auto elettrica gode di alcuni giovamenti economici sia da parte delle amministrazioni comunali che dal Governo e dalle case produttrici. Per di più, oltre ai consumi ridotti rispetto ai veicoli alimentati a benzina o a diesel, l'energia elettrica ha un costo minore rispetto a tali carburanti. Altro vantaggio è dato dalla possibilità di posizionare una colonnina di ricarica elettrica nella propria abitazione o luogo privato, usufruendo sempre di agevolazioni fiscali istanziate dal Governo.
Il costo dell'assicurazione si riduce del 10-30% rispetto all'assicurazione delle auto classiche, in quanto le macchine elettriche risultano più sicure rispetto alle altre.
Dal punto di vista ambientale come è stato detto più volte abbiamo una diminuzione delle emissioni dannose di circa il 30% in confronto alle auto con motore termico. Altro vantaggio è la riduzione dell'inquinamento acustico, in quanto le auto green risultano molto più silenziose, permettendo così di rispettare l'ambiente urbano e chi ci vive. Vantaggio del tutto non scontato è che, chi possiede questo tipo auto, ha l'agevolazione di poter circolare nelle zone a traffico limitato (ZTL) delle città e la possibilità barra diritto di non pagare il biglietto del parcheggio a strisce blu.

Nonostante i numerosi vantaggi, come per ogni cosa, anche le auto green hanno dei punti di criticità. Innanzitutto hanno un'autonomia limitata e quindi inferiore rispetto alle auto tradizionali o alle auto ibride.
Inoltre la poca disponibilità di colonnine elettriche nelle diverse città, maggiormente nei piccoli paesini e soprattutto in autostrada limita la voglia e l'interesse di acquistare auto elettriche ai viaggiatori. Per di più la procedura di smaltimento delle batterie utilizzate non è un procedimento del tutto ecologico[5].
Altro svantaggio di non poco conto è il tempo di ricarica di cui necessitano. Esso richiede qualche ora della giornata, quindi è fortemente consigliato avere sempre a disposizione una wallbox [1] o simili a casa e un parcheggio attrezzato di colonnine elettriche fuori casa,

[1]Dispositivo a parete che preleva la corrente di casa e la trasferisce all'auto tramite un apposito adattatore, permettendo anche di avviare una ricarica da remoto.

per poter sfruttare il tempo della sosta per la ricarica e non rischiare di rimanere a piedi.

Siamo quindi arrivati ad una possibile criticità della mobilità elettrica. Le colonnine di ricarica. In Italia siamo molto indietro con l'ubicazione di colonnine rispetto agli altri paesi dell'Europa e per di più, la concentrazione maggiore di esse la troviamo al Centro Nord, in particolare in Lombardia, seguita da Toscana, Piemonte, Emilia-Romagna, Lazio e Veneto.
Nel 2020 sono state registrati circa 13.176 punti di ricarica. Oltre il 70% dei sistemi è pubblico, il rimanente è privato, ma adibito ad uso pubblico. In aggiunta alla carenza di strutture si è notata la necessità di un sistema di monitoraggio e individuazione delle colonnine facile da utilizzare per gli automobilisti. Tra l'altro la localizzazione delle stesse è raggruppata negli stessi punti e al di fuori delle autostrade, togliendo quindi la possibilità ai viaggiatori di percorrere lunghi tragitti in autostrada[6].

Figura 1.2: Mappa della collocazione delle colonnine elettriche in Italia[32]

1.1 Sviluppo nel tempo

La prima macchina elettrica nasce nei primi anni 30' dell' ottocento con Robert Anderson un' imprenditore della Scozia, il quale progettò la prima carrozza elettrica della storia. Qualche tempo dopo un professore di origine olandese idealizzò il primo modello di auto elettrica vera e propria, la quale fu realizzata solo un paio di anni dopo dal suo costruttore Cristopher Becker.
In seguito alle sperimentazioni di due ingegneri francesi sulle batterie delle auto elettriche, esse iniziarono ad essere le più comode, silenziose e pratiche da guidare anche se la velocità del veicolo non era un punto forte del mezzo. Infatti il limite massimo di velocità raggiunto da un auto elettrica del fine 800' fu di 100km/h, grazie ad un veicolo a forma di razzo al suo pilota Camille Jenatzy.

Figura 1.3: La Jamais Contente - veicolo elettrico a forma di razzo

All' inizio del 1900 ci fu una situazione di equilibrio tra l'utilizzo di veicoli a benzina e quelli elettrici. Tanto che si iniziarono a ideare le prime forme di mobilità sostenibile, le quali consistevano in un servizio di car sharing con auto elettriche nelle città europee più importanti. Ma nella prima metà del 900 con l'innovazione tecnologica e il contemporaneo ritrovamento di diversi giacimenti petroliferi si capovolse il mercato automobilistico e l'auto a combustione predominò su quella elettrica.
Nella seconda metà del 1900 invece ci fu un ritorno di interesse per la mobilità sostenibile e nuovo interesse verso il cambiamento climatico, il quale si affermò sempre di più negli anni 2000. Anni in cui si diede molta importanza ai danni causati dal cambiamento climatico. Così le case automobilistiche iniziarono di nuovo e con maggiore interesse a darsi da fare nell' ambito dei trasporti sostenibili[37].

Arriviamo ad oggi. Il mercato delle auto elettriche è in forte crescita e in continuo miglioramento, ma la mancanza di infrastrutture per la ricarica di auto elettriche è un ostacolo molto importante da superare se si vuole promuovere ed incentivare la green mobility.
Le ultime statistiche di Motus-E[2], ci indicano che nel febbraio 2020 in Italia erano presenti circa 13.721 postazioni di ricarica, di cui il 73% rappresenta stazioni di tipo pubblico e il restante 27% di tipo privato, ma utilizzato per un uso pubblico.[14].
Si è verificata quindi una crescita del 33% rispetto a settembre 2019. Andando nello specifico nella situazione italiana e ancora di più nella situazione tra le varie regioni, possiamo notare che uno sviluppo maggiore si è verificato soprattutto al Nord seguito dal Centro e in percentuale molto minore dal Sud [13].

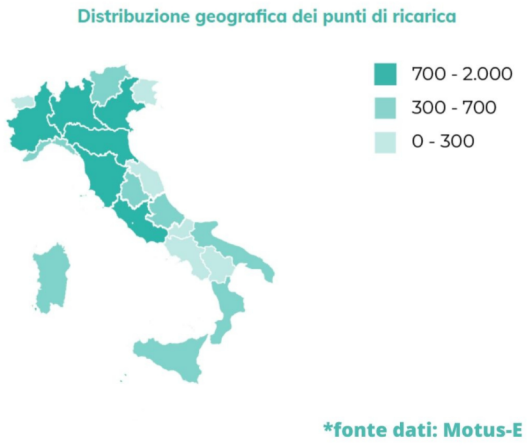

Figura 1.4: Rappresentazione della distribuzione delle colonnine per ogni regione[14]

1.2 Cosa sono le colonnine elettriche e la loro classificazione

La colonnina di ricarica elettrica è un distributore di energia che permette di ricaricare l'auto grazie ad un propulsore elettrico. Ad oggi esistono diverse tipi di colonnine, tra cui quella a palo, a parete e portatile. Come già descritto tali colonnine possono essere di due tipi: pubbliche e private.
Quelle pubbliche vengono installate in ambienti come i parcheggi pubblici, i parcheggi degli aeroporti e delle stazioni ferroviarie, direttamente dai diversi fornitori di energia. Mentre quelle private sono collegate alla rete locale e sono quelle che si trovano nelle

[2]MOTUS-E è la prima associazione Italiana composta da vari enti per la promozione e il desiderio del cambiamento verso la green mobility[15]

abitazioni, nei parcheggi dei negozi, dei ristoranti, delle aziende ecc...

Se consideriamo il fattore ricarica, ci sono due scenari possibili, la ricarica a casa e la ricarica fuori casa. La prima risulta molto più comoda ed economica rispetto alla seconda. Innanzitutto il costo dell'energia con un contratto domestico permette una ricarica con un costo che è quasi la metà rispetto ad una ricarica effettuata in una colonnina pubblica. Inoltre se si dispone di un pannello fotovoltaico il costo sarà ancor di più inferiore.
Ad uso domestico invece è consigliato l'utilizzo di una presa di tipo industriale e l'installazione della cosiddetta Wall-box, un dispositivo il cui compito è quello di coordinare in modo intelligente il rifornimento del mezzo in base ad alcuni parametri domestici, avendo anche la possibilità di poterlo fare da remoto.
Altro punto a favore dell'automobilista green è la possibilità di poter collegare la propria auto alla corrente durante le ore di inattività familiare, cioè durante la notte.
La ricarica fuori casa invece è più articolata. Per gestire il pagamento della corrente per esempio, oltre a costare molto di più, ci sono due opzioni, pagamento in base al consumo o tariffa "flat", cioè un abbonamento mensile o annuale.
Inoltre ci sono diversi operatori e associazioni adibite a tale servizio dislocate per tutta Italia, per cui per poter usufruire del servizio bisogna essere in possesso delle corrispettive tessere o applicazioni a differenza di un rifornimento di benzina o diesel tradizionale.
Oltre a questo, come già argomentato, le stazioni di ricarica non sono ancora abbastanza diffuse in Italia, soprattutto al sud, e ciò limita la possibilità di spostarsi fuori dal proprio paese per raggiungere una località lontana[25].
Per poter ricaricare la macchina bisogna quindi usufruire di uno di questi servizi sopra citati e tramite un caricatore collegato ad una colonnina di ricarica compatibile è possibile effettuare il caricamento del veicolo.[7]

Figura 1.5: Colonnina elettrica e Wallbox

1.3 Mobilità elettrica, tools per il funzionamento

Di seguito sono riportati alcuni esempi di software e il loro funzionamento per la gestione delle stazioni di ricarica, già esistenti da qualche tempo.

1.3.1 Thor

Una delle piattaforme più conosciute oggi per la gestione delle stazioni di ricarica dei veicoli elettrici si chiama THOR, una piattaforma utilizzabile mediante un sito web creato dalla startup, Smartbit srl, la quale opera maggiormente nel settore della mobilità elettrica e dell'IoT (Internet of Things[3])[8].

Figura 1.6: THOR

THOR si occupa quindi di gestire la parte più gravosa per quanto concerne la mobilità elettrica e cioè la gestione dei pagamenti delle ricariche e degli utenti.
THOR è un sistema SAAS - Software As A Service [4]. Per usufruire di tale sistema gli unici requisiti richiesti sono la possibilità di connessione a internet e la disposizione delle colonnine alla comunicazione tramite protocollo OCPP [5].
L'idea avuta da questa startup nasce dall'esigenza di mercato, in quanto la maggior parte dei software esistenti sono le legati direttamente ad un fornitore di colonnine. THOR invece utilizzando tale protocollo permette la gestione di qualsiasi colonnina di ricarica, utilizzando un sistema di pagamento che supporta ogni tipo di carta di credito ed eliminando così uno dei problemi relativi alla gestione della mobilità elettrica. Il software è diviso in lato backend e in lato frontend. Il primo è il lato amministrativo, il quale da la possibilità di inserire nuove colonnine e di monitorarne quelle esistenti.
Il monitoraggio prevede la possibilità di visualizzare lo storico delle ricariche effettuate

[3]Alla base dell'Internet of Things ci sono gli oggetti che fanno parte della vita quotidiana, che grazie ad internet acquistano una loro intelligenza per comunicare con altri oggetti[36].

[4]modello di distribuzione, dove il produttore di software sviluppa, opera e gestisce un' applicazione web, ponendola a disposizione dei clienti via internet[9].

[5]Open Charge Point Protocol - Protocollo che permette di pilotare da remoto, grazie ad un server OCPP, le colonnine. Tale protocollo è nato pochi anni fa grazie ai maggiori produttori di colonnine.

(importo della ricarica e tipo di pagamento), le ricariche in corso in tempo reale per poter agire su di esse, bloccando e sbloccando i connettori, i metodi di pagamento, le tariffe e le modalità.

Altro punto fondamentale è la gestione delle tessere RFID, le quali possono essere utilizzate su tutte le colonnine dell'infrastruttura. Per quanto riguarda il lato frontend invece, possiamo distinguere due interfacce. La prima è una mappa che fornisce la localizzazione delle colonnine ed insieme ad essa la potenza e il connettore disponibile e il suo stato in quel momento (libero/occupato). La seconda, a cui si può accedere solo dopo aver fatto accesso tramite login, la quale permette la visualizzazione dello storico delle ricariche, le fatture e l'area in cui è possibile modificare i dati personali[11]. Tale software permette quindi di far concentrare l'imprenditore sul proprio business senza pensare alla gestione[10].

1.3.2 Emobility

Un'altro cloud adibito alla gestione delle stazioni di ricarica è EMOBILITY. Esso permette il controllo e il monitoraggio delle postazioni di carica delle auto appartenenti ad un privato, che a sua volta può decidere di mettere a disposizione del pubblico la propria o le proprie colonnine di ricarica, così che tutti gli EV Drivers[6] possano usufruirne.
Queste ultime vengono collegate alla piattaforma tramite una SIM-Card che permette alla colonnina di essere online e operativa in qualsiasi momento, senza dover quindi utilizzare server e software locali con un costo di non poco conto.
Si ha quindi la possibilità di sapere in real-time lo stato della propria colonnina, se è libera o occupata o se ci sono eventuali guasti. Le ricariche effettuate vengono tutte registrate e archiviate così da fornire al proprietario un quadro generale del sistema di ricarica in base al periodo che si vuole analizzare. In caso di guasti o manutenzione tale software esegue la manutenzione da remoto rendendola veloce ed economica.
Il proprietario del punto di ricarica ha quindi la possibilità di abilitare o disabilitare utenti esterni ai quali viene eventualmente affidata una tessera RFID per identificarsi e attivare la colonnina. Questi ultimi possono trovare i punti di ricarica grazie alle mappe messe a disposizione sul web e il pagamento della ricarica può essere effettuato tramite diversi mezzi di pagamento messi a disposizione.
EMOBILITY si occupa inoltre di tutte le attività collegate alla gestione degli utenti esterni e alla remunerazione dovuta al proprietario, al quale vengono inviati bonifici bancari per rimborsarlo. Questa gestione permette al proprietario di ottenere tutti i vantaggi del suo punto di ricarica senza sforzi aggiuntivi [12].

[6]Electric Vehicle Drivers - Guidatori di auto elettriche

Figura 1.7: EMOBILITY

1.3.3 JOINON

JOINON è un sistema sviluppato dalla Gewiss, un'azienda Italiana del settore elettrotecnico molto attenta allo sviluppo della mobilità sostenibile. Esso consiste in un software che si articola in una parte di gestione per i proprietari delle stazioni di ricarica e in un' applicazione che viene utilizzata dall' utente che usufruisce del sistema.
Tale piattaforma consente al proprietario la gestione del sistema e il monitoraggio delle colonnine in tempo reale per ogni singola stazione. Inoltre permette di censire, avere informazioni e verificare il funzionamento di ogni stazione di ricarica. Ogni gestore può impostare le potenze erogabili, i prezzi dell' energia e le prese per le colonnine e le wallbox. Tramite un' interfaccia è possibile visualizzare lo stato delle colonnine, se sono libere, occupate o in manutenzione, inoltre è possibile attivarle o disattivarle da remoto, impostarne l'orario e gestirne i pagamenti e la fatturazione. Eventuali malfunzionamenti delle colonnine o delle wallbox si possono risolvere da remoto con un servizio di manutenzione più semplice ed economico. Gewiss aggiorna costantemente il firmware della stazione di ricarica all'ultima versione per fornire prestazioni migliori e compatibili con i veicoli elettrici del futuro. La piattaforma retribuisce direttamente i proprietari delle stazioni senza dover usufruire di risorse amministrative. L' App JOINON per gli utenti che utilizzano tale servizio è semplice ed immediata da utilizzare. Tramite una mappa si possono individuare e raggiungere i punti di ricarica disponibili. Dopo di che permette la gestione della ricarica che viene attivata tramite la scansione del codice QR posto sulla colonnina o sulla wallbox[44].

Figura 1.8: JOINON

1.3.4 CHARGE&GO

Il progetto che ho svolto per la Comunità Solare, presso il gruppo FINMATICA S.P.A. riguarda proprio le colonnine elettriche per la ricarica di un autoveicolo e si chiama CHARGE&GO.
Tale software permette la gestione delle colonnine poste nelle stazioni di ricarica. Il suo ruolo è quello di fare da tramite tra il cliente che si dirige presso la stazione per caricare il suo veicolo e i gestori di tali stazioni.
L'utente iscritto a tale servizio viene in possesso di una tessera RFID(Radio-frequency Identification), la quale utilizza una tecnologia per l'identificazione e/o memorizzazione di informazioni riguardanti oggetti, grazie all'utilizzo di etichette elettroniche nominate tag e alla loro capacità di interagire con dei reader(dispositivi fissi o portatili)[21].
Una volta entrato in possesso di tale tessera, quando l'utente si trova presso una delle colonnine mappate dal sistema, esso potrà attivare quest'ultima grazie al codice RFID del proprio bagde. La ricarica avrà inizio solo una volta che il sistema avrà verificato la registrazione dell'utente al servizio e verificato i suoi dati, come ad esempio la data di scadenza del suo badge in base al tipo di abbonamento stipulato.
Una volta verificato tali dati, il servizio di ricarica verrà attivato e l'utente potrà ricaricare la propria auto per poter viaggiare più green che mai.

Figura 1.9: Logo del software

1.3.5 Presentazione generale delle piattaforme esistenti

CHARX manage è un software per la gestione delle stazioni di ricarica, esso presenta un' interfaccia tra il conducente, il gestore della stazione di servizio e il fornitore dei servizi backend. Viene messo in funzione, configurato e monitorato tramite il web. Durante il caricamento del veicolo il software registra i dati energetici e le transazioni in una banca dati SQL locale e li invia al fornitore backend per poterne ricavare delle statistiche[46].
Altri esempi di software molto simili e disponibili per la gestione dei punti di ricarica sono: ViriCiti e ChargePoint.

Per quanto riguarda le applicazioni mobili invece, ne esistono varie che permettono di interagire con le colonnine e individuarle grazie all' utilizzo di un sistema di geolocalizzazione.

NEXTCHARGE è considerata l'app più completa in assoluto e permette di trovare colonnine elettriche in tutto il mondo sia pubbliche che private, visualizzare la stazione più vicina e il tempo di percorrenza per raggiungerla, i costi, i tipi ricarica e la prenotazione della colonnina.

E-CHARGE permette di trovare colonnine, valutarne lo stato, conoscere in anticipo il prezzo e controllare la ricarica tramite startstop. Inoltre per i pagamenti sono previsti di piani tariffari per gli abbonati, mentre per gli utenti occasionali il pagamento avviene tramite carta di credito.

PLUGSHARE conta circa 140000 stazioni pubbliche in tutto il mondo ed è consultabile anche tramite Apple Watch. Permette di trovare punti di ricarica in tempo reale anche grazie alle foto e recensioni degli utenti.

RECHARGE AROUND trova le stazioni di ricarica sia in Italia che all'estero, effettua ricerche in base alla disponibilità e potenza e fornisce informazioni base come orario di apertura e chiusura delle stazioni.

E-MOVING invece è un' app che presenta qualche limitazione in quanto visualizza solo le colonnine che si trovano a Milano e a Brescia.

E-GO Ricarica è un' app sviluppata da ENEL S.P.A. Essa permette di visualizzare le stazioni di Enel Energia, l'itinerario per raggiungerla, prenotare la presa di ricarica e pagamento tramite carta di credito[45].

ENEL X RECHARGE viene utilizzata per ricaricare veicoli nelle stazioni pubbliche o tramite ricarica domestica offerta dalla stessa Enel. Essa prevede quattro tariffe disponibili: profilo base, premium, flat small e large. Il suo vantaggio è che in caso di prenotazione si ha a disposizione 15 minuti per raggiungere la stazione e usufruire della prenotazione gratis.

Y-APP è l'app del gruppo Friem, il progetto Yess.Energy ha come scopo quello di "commercializzare soluzioni di energia integrata innovative nel settore delle infrastrutture di ricarica per veicoli elettrici". Essa segue il processo di ricarica dalla prenotazione al pagamento al controllo e gestione della ricarica.

EVWAY fornisce il suo servizio agli utenti privati, alle società e alla pubblica amministrazione. Essa implementa un tutorial che guida da ogni passo l'utente nel processo di ricarica oltre a fornire tutte le informazioni generali della stazione.

Esistono altre app che hanno la stessa funzionalità e fli stessi principi di funzionamento e sono: ENERMIA, D-Mobility, Plugsurfing, Carwow, New Motion.

App come On, ALPERIA MOBILITY, PlugGo, REPOWER, offrono noleggio di auto elettriche a lungo termine a differenza delle altre applicazioni. Inoltre PlugGo e E-MOVING danno una possibilità in più che è quella di installare una wall-box privata.

Per chi invece possiede automobili elettriche con i seguenti marchi, esistono per esse le relative applicazioni:

- EQ-Ready per le auto Mercedes Benz
- Go I-PACE per Jaguar
- Ready To Charge di Enel-X per Smart
- Tesla per l'omonimo marchio[47]

1.4 Contesto attorno al software

Tale software nasce da un progetto pensato e realizzato dalla Comunità Solare dell'Emilia-Romagna. Essa grazie alla collaborazione con un'altra azienda chiamata Nrg4You sta rendendo le località e in particolare la mobilità locale sempre più eco-sostenibile. Di seguito sono riportati gli ambiti in cui operano e gli obbietti prefissati di queste due realtà sopra citate.

1.4.1 Comunità Solare

Il Centro per le Comunità Solari Locali è un'associazione privata nata nel 2015, senza scopo di lucro da uno spin-off[7] dell'Università di Bologna. Il progetto è nato nel 2007 in seguito a delle attività del gruppo di ricerca scientifica nell'ambito dell'organizzazione energetica dei comuni della provincia di Bologna.
Tra il 2014 e il 2017 è stato creato un network per osservare e studiare il consumo di

[7]Lo Spin-off universitario è uno strumento per il trasferimento sul mercato di determinati progetti che nascono da iniziative prese da professori e ricercatori universitari, borsisti dell'università[23].

energia domestico per poterne sviluppare degli strumenti più consoni all'obbiettivo finale: un maggior utilizzo di energia rinnovabile.
Tale network è composto indicativamente da 300 famiglie facente parte delle 8 comunità locali presenti nella regione, che sono: Casalecchio di Reno, San Lazzaro di Savena, Medicina, Zola Predosa, Ozzano dell'Emilia, Bologna, Sasso Marconi e Castel Maggiore.

Lo scopo delle Comunità Solari Locali è quello di accompagnare la comunità verso un mondo solare la cui principale fonte di energia è un'energia rinnovabile e non inquinante. Mossi dai forti cambiamenti climatici, l'obbietto dei cittadini solari è di fare del risparmio energetico la base su cui costruire il nostro futuro.
Un mezzo per raggiungere tale scopo è promuovere la mobilità sostenibile. Oltre a spronare l' utilizzo della bicicletta per muoversi nel paese, scelta del tutto ecologica, ha iniziato da tempo a promuovere le cosiddette auto Green a zero emissioni.
Dal monitoraggio si è visto che metà dei consumi giornalieri di una famiglia è dovuto all'utilizzo di combustibili per le autovetture. Per questo motivo nasce un progetto chiamato CHARGE&GO.
Esso prevede la realizzazione di una rete per fornire un servizio di ricarica elettrica gratuita per le auto[24]. Diventando un cittadino solare infatti abbiamo la possibilità di acquistare auto elettriche, richiedendo un preventivo personalizzato, presso la rete ufficiale grazie alla convenzione con la CSL.
Per portare avanti questo progetto la Comunità solare si è servita di un' azienda chiamata Nrg4You e dei suoi servizi[22].

Figura 1.10: Comunità Solare Locale

1.4.2 Nrg4you-Restation

Nrg4You s.r.l. è un'azienda innovativa nata a Napoli, che si occupa della ricerca industriale, la produzione e commercializzazione di prodotti e servizi a sostegno della Mobilità

Eco-Sostenibile e dell'efficienza nel produrre energia proveniente da fonti rinnovabili.
L'azienda nasce nel marzo del 2014 e negli anni 2015-2017 si dedica alla ricerca nel settore della green mobility facendo grandi passi in avanti.
Nel 2019 si trasforma in una Scale-Up[8] entrando a far parte del mercato della mobilità eco-sostenibile. Tutto l'impegno che l'azienda impiega nasce dalla convinzione che nel futuro, ci sarà sempre più la necessità di individuare e sviluppare fonti alternative di energia che siano più sicure e affidabili, non solo per l'essere umano ma anche per l'ambiente.
Il loro obbiettivo è quello di valorizzare il sole, il vento e l'acqua come fonti naturali da cui poter ricavare energia eco-sostenibile. Perciò hanno creato un progetto chiamato Restation. [26]

Figura 1.11: Nrg4You

Restation è una stazione di servizio fornita di colonnine elettriche per la ricarica. Tali colonnine sono fornite di un'elettronica intelligente che viene controllata tramite un portale online. I servizi che offre sono diversi, tra cui: potenza di carica variabile, localizzazione e prenotazione di una colonnina, contabilizzazione e pagamento dell'energia utilizzata.[26]

Figura 1.12: Restation - stazione di ricarica per veicoli elettrici

[8]la scale-up è l' evoluzione di una start-up affermatasi nel mercato, pronta ad ambire ad una crescita internazionale[27]

Capitolo 2

Progettazione

2.1 Contesto di lavoro

Il contesto lavorativo in cui ho svolto il progetto presentato in questo elaborato è la realtà del gruppo FINMATICA S.P.A. Il progetto è stato portato avanti durante lo svolgimento del tirocinio della durata di 150 ore. Il gruppo FINMATICA S.P.A. è una delle società fra le principali produttrici di software sul territorio nazionale ed eroga servizi in ambito di information and comunication technology.
La maggior parte dell'ambito in cui operano è quello della sanità e della pubblica amministrazione oltre che degli enti privati, come per esempio la Comunità Solare Locale, per il quale insieme all'aiuto del team ho contribuito allo sviluppo della piattaforma [20].

2.2 Descrizione generale dell'applicazione

Come già annunciato precedentemente l'applicazione prevede la creazione di un software utile alla gestione delle colonnine elettriche messe a disposizione dalla Comunità Solare per la ricarica dei veicoli a zero emissioni.
L'applicazione è composta sia da un lato backend che uno frontend, utile per i gestori del servizio a tener traccia di tutti i movimenti riguardarti le azioni dei clienti e la situazione delle colonnine. Gli amministratori hanno la possibilità, tramite l'interfaccia implementata, di poter aggiungere e rimuovere un utente dal database, oltre che modificarne i dati personali e i dati riguardanti la propria tessera RFID per la ricarica.
Per quanto riguarda gli utenti, questi ultimi hanno la possibilità di scaricare l'App relativa chiamata Cittadino Solare per poter usufruire del servizio messo a loro disposizione.

2.3 Requisiti richiesti dall'azienda

L'Ente promotore del progetto ha richiesto lo sviluppo di alcuni casi d'uso necessari per il funzionamento della piattaforma.

2.3.1 Casi d'uso

Di seguito sono riportati i due casi d'uso richiesti per il corretto funzionamento della piattaforma.

- Caso d'uso 1: Registrazione di una nuova anagrafica per l'attivazione della tariffa flat.
  Al fine di poter attivare la ricarica della propria auto, presso le colonnine convenzionate con la Comunità Solare, gli utenti devono prima essersi registrati nel BackOffice previsto per l'attivazione della tariffa di ricarica. Al momento della registrazione l'utente riceve un badge RFID o un codice di attivazione per l'App.

  Attività di sviluppo previste:

  - Database: Progettazione e creazione della tabella per i dati anagrafici dell'utente e dei dati per l'attivazione della tariffa. I campi richiesti sono: Id, Nome, Cognome, Data di nascita, Codice fiscale, Codice badge, Codice App, Data attivazione, Data scadenza, Data aggiornamento, Utente aggiornamento.

  - Interfaccia web: Form con l'inserimento e la modifica dei dati anagrafici, Visualizzazione dell'elenco degli utenti.

  - Servizi Rest: Inserimento dell'anagrafica, modifica dell'anagrafica, cancellazione dell'anagrafica e query anagrafica.

- Caso d'uso 2: Ricarica con badge RFID.
  L'utente che ha effettuato la registrazione si può recare presso una colonnina convenzionata con la Comunità Solare e utilizzare il badge RFID per abilitare la ricarica. Il sistema installato sulla colonnina esegue una chiamata HTTP GET ad un servizio di verifica esposto dal BackOffice della Comunità solare. Se la verifica va a buon fine il BackOffice CSL esegue una richiesta di abilitazione porta Back-end Restation.

  Attività di sviluppo previste:

  - Servizi REST: Verifica del codice RFID, abilitazione della porta Back-end Restation.

2.4 Tecnologie utilizzate

Alla base del progetto c'è il Database, senza di esso il software non avrebbe modo di esistere. Del resto tutti i software che prevedono la gestione di una serie di dati, come per esempio quelli appartenenti ad un utente facente parte di un' organizzazione, hanno bisogno di un sistema di registrazione permanente di essi.

2.4.1 Database

Un database è un insieme di dati molto esteso, un archivio strutturato memorizzato in un sistema informatico per conservare informazioni che potranno poi essere utili in un momento successivo.
Tali dati vengono salvati in delle tabelle composte da righe e colonne per poter effettuare una migliore ricerca tra i record, grazie all'utilizzo di query[1]. Per poter scrivere, aggiornare i dati ed eseguire le query viene utilizzato il linguaggio SQL (Structured Query Language).
SQL è un linguaggio di programmazione per database che si basano su un modello relazionale. Tramite appunto l'utilizzo di query esso interroga e gestisce i dati presenti nel database.

Ci sono varie tipologie di database, tra le più utilizzate abbiamo[35]:

- Database relazionali - In un database relazionale i dati sono organizzati in tabelle suddivise in righe e colonne, offrendo così un metodo più efficiente per aver accesso alle informazioni salvate.
- Database orientati agli oggetti - In questo tipo di database le informazioni vengono rappresentate sotto forma di oggetti.
- Database distribuiti - In un database distribuito i database sono dislocati su più computer connessi tra loro.
- Data warehouse - In un data warehouse esiste una repository centralizzata ed è utilizzato per compiere eseguire query e analisi veloci.
- Database NOSQL - Un DB NOSQL è definito anche non relazionale e permette la manipolazione di dati non strutturati o semi-strutturati. Esso viene maggiormente usato con le applicazioni Web.

[1]Le query vengono utilizzate per l' interrogazione di un DB per poter ricavare informazioni o aggiornare/eliminare i dati presenti nel database.

Nel progetto svolto è stato utilizzato un database di tipo relazionale, in quanto fornisce un modo più intuitivo e semplice per inserire i dati necessari nelle tabelle. In tale sistema ogni riga della tabella rappresenta un record a cui è associata una chiave rappresentata da un Id univoco. Grazie a questa chiave le tabelle presenti nel database possono comunicare tra di loro per ricavare le informazioni di cui si ha bisogno.
Inoltre un database relazionale per far si che le transazioni avvengano in modo corretto supporta le seguenti proprietà fondamentali denominate con l' acronimo ACID:

- Atomicità
- Consistenza
- Isolamento
- Durabilità

Altro vantaggio sono le tecniche di blocco e concorrenza per garantire l' integrità dei dati. Secondo la tecnica di blocco viene impedito di aver accesso ai dati quando sono in fase di aggiornamento. Tale blocco può essere sia applicato a tutta la tabella oppure solo al record in fase di aggiornamento. La tecnica di concorrenza invece permette un corretto accesso al DB in base alle policy definite nel caso in cui più di un utente vuole accedere contemporaneamente ad esso [34].

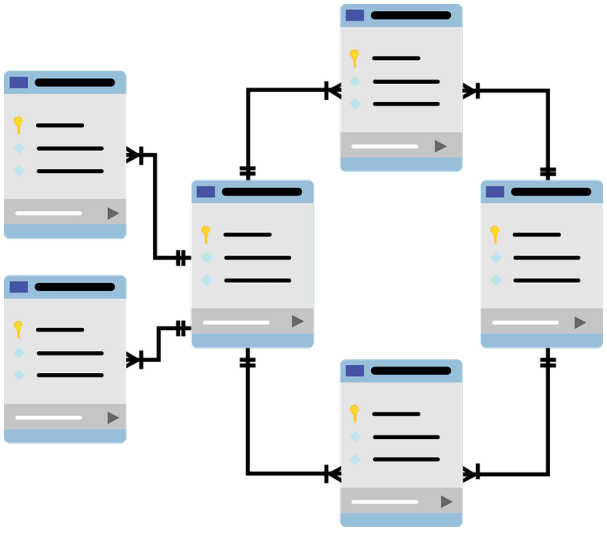

Figura 2.1: Schema di un database

Oltre al database, le tecnologie che sono state utilizzate per lo svolgimento del progetto sono:

- Spring Boot per i servizi middleware e gestione del database
- React js per la user interface
- PostregSQL per lo storage dei dati

2.4.2 Spring Boot

Spring Boot è una soluzione semplice e immediata per l'implementazione e il testing di un App su una piattaforma Spring[2]. Esso produce micro servizi in java, infatti facilita la progettazione di applicazioni stand alone[3] eseguibili. Il framework Spring Boot è stato pensato con lo scopo di ricavare il massimo potenziale dalle librerie di Spring per garantire un'ampia gamma di funzionalità. [17]

Figura 2.2: Logo Spring Boot

2.4.3 React.Js

React.Js è una libreria JavaScript nata per poter essere utlizzata per la progettazione di UI(User Interface) interattive per l'applicazione. Attraverso questa libreria è possibile creare componenti isolate a loro volta componibili per l'implementazione di interfacce complesse, utilizzando un approccio di tipo dichiarativo[18]. Ciò permette di rappresentare le componenti con una chiamata all' API di React, la quale interviene in maniera rapida sul DOM[4] per importare gli oggetti necessari[19].

Figura 2.3: Logo React.Js

[2]Framework open source per la creazione di applicazioni Java.[16]

[3]Oggetto o software capace di funzionare in autonomia o indipendentemente da altri oggetti o software ad esso collegati.

[4]Document Object Model(Modello a oggetti del documento) è un modo di rappresentare i documenti.

2.4.4 PostgreSQL

PostgreSQL è un sistema open source di database relazionali. Esso amplifica il linguaggio SQL con diverse funzionalità per lavorare in modo sicuro con i dati anche più complessi. Le peculiarità di tale sistema sono diverse: funziona su tutti i principali sistemi operativi, garantisce affidabilità, integrità dei dati ed estensibilità.

Figura 2.4: Logo PostgreSQL

2.4.5 Axios

Axios è una libreria Javascript per fare chiamate e ricevere risposte HTTP ed effettuare richieste verso un server. Essa oltre a restituire un oggetto già analizzato dal file Json, possiede altre funzionalità utili[28]. Esse sono:

- Intercettori - Utili per accedere alla configurazione ad elementi come le intestazioni di una richiesta o di una risposta.
- Istanze - Utili alla creazione di istanze utilizzabili con baseURL.
- Impostazioni predefinite - utile per impostare i valori delle intestazioni comuni delle richieste effettuate, sopratutto se per ogni richiesta viene prima effettuata un'autenticazione su un server.

Capitolo 3

Implementazione

3.1 Funzionalità implementate

Per lo svolgimento del progetto ho implementato una tabella utilizzando il framework Spring-boot, che a sua volta si appoggia sul sistema open source PostgreSQL per essere creata e visualizzata da lato backend. Inoltre tramite la libreria React.Js è stata implementata anche un'interfaccia per questa tabella, per far si che l'amministratore del servizio di ricarica per auto elettriche possa interagire con il software nel modo più semplice possibile. Infatti tale tabella possiede i comandi per permette all'amministratore di poter aggiungere, eliminare e modificare i dati relativi ad un'utente, in maniera del tutto veloce ed intuitiva.

3.2 Gestione dei dati

Per poter implementare la parte del progetto che riguarda il server ho quindi utilizzato Spring Boot per la parte implementativa e PostregSQL per la visualizzazione delle tabella nel server. Di seguito è riportato la maggior parte del codice scritto per l'implementazione con la sua relativa spiegazione, ma prima di tutto nel seguente sotto paragrafo verranno descritte le annotazioni di Spring-boot comuni alla maggior parte dei metodi creati.

3.2.1 Descrizione delle annotazioni di Spring-boot

Le Spring Boot Annotation vengono utilizzate per dare delle informazioni aggiuntive su un' applicazione.
Tali annotazioni però non incidono in maniera diretta sul funzionamento del programma, cioè non condizionano la sua compilazione.
Di seguito sono elencate le annotazioni utilizzate e il loro ruolo all'interno di un'applicazione in Spring Boot.[31].

- @Entity - Tale annotazione qualifica la classe come un' entità mappata in una tabella del database.
- @id - L' annotazione @id assegna la chiave primaria all' entità e tramite l' annotazione @GeneratedValue viene dettagliata la strategia per la generazione delle chiavi primarie della tabella.
- @JsonFormat - Questa annotazione viene utilizzate per specificare il formato con cui deve essere rappresentata la data, tramite la voce pattern[39].
- @PreAuthorize - Tale annotazione fa parte delle Spring Security Annotation ed è utilizzata per verificare la necessità di autorizzazione prima di poter eseguire un metodo, L'autorizzazione avviene in base al ruolo o argomento che viene passato al metodo[30].
- @RequestMapping - Questa annotazione è utile per mappare le richieste web. In particolare in questo progetto viene utilizzato sui metodi per fornire l'URI sul quale verrà eseguito il metodo creato.
- @PathVariable - E' una variabile di percorso che viene usata per estrapolare i valori dall' URI.
- @RequestBody - Tale annotazione mappa l'HttpRequest su un oggeto di tipo JSON inviato dal lato client che viene deserializzato in tipo Java, converte cioè il corpo della richiesta associandolo al paramentro annotato[40].
- @RequestHeader - Questa annotazione è utile per ottenere informazioni sulle intestazioni della richiesta.
- @RequestParam - Questa annotazione viene utilizzata per estrarre i parametri dall' URL e mapparli nell' argomento del metodo.

3.2.2 Codice implementativo

Prima di tutto per poter implementare il codice sono state aggiunte delle dipendenze nel file di configurazione pom.xml[1] del progetto. Tali dipendenze servono per la configurazione della build e includono librerie di terzi parti per la creazione dell'applicazione.
Le due dipendenze che ho usato per la creazione del database sono quella con la JPA Repositories[2] e PostgreSQL, di seguito rispettivamente riportate.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Figura 3.1: Configurazione a JPA

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.2.5</version>
</dependency>
```

Figura 3.2: Configurazione di postgreSQL

Il file UtenteRicaricaRepository contiene l'interfaccia UtenteRicaricaRepository, la quale estende la JPARepository e trova il nome dell' utente in base al codice badge ad esso appartenente, grazie al metodo findByCodiceBadge() della classe UtenteRicarica a cui viene effettivamente passato il valore da trovare.

[1]Il Project Object Model è il file di configurazione di Maven che la orienta nell'esecuzione definendo la struttura del progetto[33]

[2]JPA Repository è un estensione che contiene API per operazioni CRUD di base e anche API per impaginazione e ordinamento[42]

```java
public interface UtenteRicaricaRepository extends JpaRepository<UtenteRicarica, Long> {
    UtenteRicarica findByCodiceBadge(String codiceBadge);
}
```

Figura 3.3: Public interface UtenteRicaricaRepository

Dopo aver creato il file UtenteRicaricaRepository ho creato un file chiamato UtenteRicarica.java in cui ho istanziato la tabella con i campi descritti nel caso d'uso richiesto dalla Comunità Solare.
Grazie all'annotazione @Entity possiamo quindi definire la classe UtenteRicarica come un' entità mappata in una tabella. Le inizializzazioni che seguono servono per definire i nomi dei campi di questa tabella. Inoltre vengono utilizzate altre tre annotazioni: @Id, @GeneratedValue, @JsonFormat.
@Id e @GeneratedValue assegnano un'id univoco generato automaticamente, da assegnare ad ogni riga della tabella. @JsonFormat invece permette di stabilire il formato in cui vogliamo che la data sia salvata, in questo caso essa deve rispettare il seguente vincolo giorno/mese/anno. Oltre l'inizializzazione dei campi tale classe possiede i metodi get e set per ogni campo per poter ricavare e settare ogni campo appartenente alla tabella.

```java
@Entity
public class UtenteRicarica {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String nome;
    private String cognome;
    @JsonFormat(pattern = "dd/MM/yyyy")
    private Date dataDiNascita;
    private String codiceFiscale;
    private String codiceBadge;
    @JsonFormat(pattern = "dd/MM/yyyy")
    private Date dataAttivazione;
    @JsonFormat(pattern = "dd/MM/yyyy")
    private Date dataScadenza;
```

Figura 3.4: Tabella degli utenti aderenti al servizio

Il metodo viewUtenteRicarica è un metodo di tipo GET che tramite il comando .findAll() restituisce tutti gli utenti presenti nel database nelle apposite righe e colonne della tabella. Al metodo vengono passati alcuni argomenti, che sono: HttpServletRequest che è la classe che ingloba la richiesta Http passandola ai metodi post e get della servlet(motore al quale viene indirizzata la richiesta http)[43] e @RequestHeader che serve per prendere in input l' autorizzazione per l'esecuzione della richiesta.

```
@PreAuthorize("hasRole('ROLE_ADMIN') or hasRole('ROLE_CLIENT')")
@RequestMapping(value = "/getUtentiRicarica", method = RequestMethod.GET, produces = "application/json;charset=UTF-8")
public List<UtenteRicarica> viewUtenteRicarica(HttpServletRequest request,
                                    @RequestHeader("Authorization") String header) throws IOException {

    return utenteRicaricaRepository.findAll();

}
```

Figura 3.5: Metodo di visualizzazione del utenti

Il seguente metodo permette la registrazione di nuovi utenti al sistema. E' un metodo di tipo POST e prende come argomento l'HttpServletRequest il @RequestBody e il @RequestHeader. Viene istanziato un oggetto jsonObj di tipo JSONObject per ricevere i dati dell' utente che poi vengono salvati sul database.
Grazie al SimpleDateFormat posso impostare il formato della date nel modo in cui preferisco.

```
@PreAuthorize("hasRole('ROLE_ADMIN') or hasRole('ROLE_CLIENT')")
@RequestMapping(value = "/sendUtenteRicarica", method = RequestMethod.POST)
public ResponseEntity<String> sendUtenteRicarica(HttpServletRequest request,
                                                 @RequestBody String json,
                                                 @RequestHeader("Authorization") String header) throws IOException, ParseException {
    JSONObject jsonObj = new JSONObject(json);
    String nome = jsonObj.getString( key: "nome");
    String cognome = jsonObj.getString( key: "cognome");
    DateFormat format = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    format.setLenient(false);
    String codiceFiscale = jsonObj.getString( key: "codiceFiscale");
    String codiceBadge  = jsonObj.getString( key: "codiceBadge");

    Date dataN;
    Date dataA;
    Date dataS;
```

```
UtenteRicarica utenteRicarica = new UtenteRicarica();
utenteRicarica.setNome(nome);
utenteRicarica.setCognome(cognome);
utenteRicarica.setDataDiNascita(dataN);
utenteRicarica.setCodiceFiscale(codiceFiscale);
utenteRicarica.setCodiceBadge(codiceBadge);
utenteRicarica.setDataAttivazione(dataA);
utenteRicarica.setDataScadenza(dataS);
utenteRicarica=utenteRicaricaRepository.save(utenteRicarica);

try {
    utenteRicarica=utenteRicaricaRepository.save(utenteRicarica);
    return ResponseEntity.ok(utenteRicarica.getId().toString());
} catch (Exception e) {
    return ResponseEntity.badRequest().build();
}
```

Figura 3.6: Metodo per registrare un nuovo utente

Il metodo updateUtenteRicarica è di tipo PUT. Esso prende tre argomenti all'interno, che sono: @PathVariable, @RequestBody, @RequestHeader. Il @PathVariable viene utilizzato per estrarre l'id dell'utente selezionato dall'URI della richiesta. Il @RequestBody è stato utilizzato per legare la richiesta Http ad una stringa. @RequestHeader per prendere in input l'autorizzazione della richiesta. In tale metodo è stato creato un oggetto jsonObj di tipo JSONObject. Come il metodo utilizzato per l'inserimento di un nuovo utente, tale oggeto serve per ricevere i dati dell' utente per essere salvati sul DB.

```
@PreAuthorize("hasRole('ROLE_ADMIN') or hasRole('ROLE_CLIENT')")
@RequestMapping(value = "/updateUtenteRicarica/{id}", method = RequestMethod.PUT)
public ResponseEntity<Object> updateUtenteRicarica(@PathVariable("id") String id, HttpServletRequest request,
                                                   @RequestBody String json,
                                                   @RequestHeader("Authorization") String header) throws IOException, ParseException {
    DateFormat formatSlash = new SimpleDateFormat( pattern: "dd/MM/yyyy");
    formatSlash.setLenient(false);


    JSONObject jsonObj = new JSONObject(json);
    String nome = jsonObj.getString( key: "nome");
    String cognome = jsonObj.getString( key: "cognome");
    String codiceFiscale = jsonObj.getString( key: "codiceFiscale");
    String codiceBadge  = jsonObj.getString( key: "codiceBadge");

    Date dataN;
    Date dataA;
    Date dataS;
```

Figura 3.7: Metodo di aggiornamento dei dati degli utenti

Il metodo deleteUtenteRicarica prende anche esso come argomenti l'HttpServletRequest, @RequestHeader e il @PathVariable. Per le prime due diciture valgono le descrizioni fornite precedentemente, il @PathVariable invece prende in considerazione l'id dell'utente e tramite il metodo .deleteById("id") a cui viene appunto passato l'id per poter procedere con l'eliminazione dell'utente scelto.

```
@PreAuthorize("hasRole('ROLE_ADMIN') or hasRole('ROLE_CLIENT')")
@RequestMapping(value = "/deleteUtenteRicarica/{id}", method = RequestMethod.DELETE)
public ResponseEntity<Object> deleteUtenteRicarica(@PathVariable("id") Long id,  HttpServletRequest request,
                                        @RequestHeader("Authorization") String header) throws IOException {
    utenteRicaricaRepository.deleteById(id);
    return ResponseEntity.ok(new JSONObject().put("id", id ).toString());
}
```

Figura 3.8: Metodo per la cancellazione degli utenti

Tramite l'annotazione @Entity è stata mappata nel database la tabella con nome LogAttivazioneRFID. Tale tabella è stata creata con il fine di gestire lo stato delle tessere rfid e delle colonnine per poter abilitare la ricarica.

```
@Entity
public class LogAttivazioneRFID {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private long id;
    public String rfid;
    public String chargeStationID;
    public String stato;
    @Size(max = 1000)
    public String responeRestation;
    public Date date;
```

Figura 3.9: Tabella logAttivazioneRFID

Il metodo checkUUID prende come argomento l'HttpServletRequest e due @RequestParam a cui vengono passati l'UUID e il serialNumber, dove il primo corrisponde all' RFID della tessera dell' utente e il secondo al numero seriale della colonnina elettrica.

Dopo di che con il metodo .findByCOdiceBadge(UUID) attribuisco alla variabile utenteRicarica un utente se nel database è presente un utente che possiede una tessera RFID con quel codice. Con un controllo if-else verifico la presenza dell' utenteRicarica nel DB, se esiste tale utente vado a fare un ulteriore controllo sulla data di scadenza della tessera con un altro ciclo if-else. Verifico se la data di scadenza va oltre la data attuale, tramite il metodo .after(), se questo viene verificato passo al metodo attivaColonnina l'UUID(corrisponde all' RFID) e il serialNumber per poter permettere la richiesta di attivazione della colonnina. Altrimenti registro nella tabella LogAttivazione che l' abbonamento è scaduto. Se la tessera non è assegnata a nessun utente invece viene registrato che la tessera non è assegnata a nessuno.

```
@RequestMapping(value = "/check", method = RequestMethod.GET)
public ResponseEntity checkUUID(HttpServletRequest request,
                                @RequestParam String UUID,
                                @RequestParam String serialNumber) {
    System.out.println(new Date());
    System.out.println(request.getRemoteAddr());
    System.out.println("uuid: "+UUID + " Serial number:" + serialNumber);

    Calendar calendar = Calendar.getInstance();
    UtenteRicarica utenteRicarica = utenteRicaricaRepository.findByCodiceBadge(UUID);
    if (utenteRicarica != null) {
        Date dataScadenza = utenteRicarica.getDataScadenza();
        if (dataScadenza.after(calendar.getTime())) {
            //OK, TESSERA ESISTENTE ED ABBBONAMENTO NON SCADUTO, ATTIVO LA COLONNINA..
            attivaColonnina(UUID, serialNumber);
            return ResponseEntity.ok("{OK}");
        }
        LogAttivazioneRFID log = new LogAttivazioneRFID(UUID, serialNumber,  stato: "ABBONAMENTO SCADUTO",  responeRestation: "", new Date());
        logAttivazioneRFIDRepository.save(log);
        return ResponseEntity.ok("{OK}");
    } else {
        //tessera non esistente
        LogAttivazioneRFID log = new LogAttivazioneRFID(UUID,serialNumber, stato: "TESSERA NON ASSEGNATA A NESSUN ABBONATO", responeRestation: "",new Date());
        logAttivazioneRFIDRepository.save(log);
        return ResponseEntity.ok("{OK}");
    } }
```

Figura 3.10: Metodo checkUUID per verificare il codice RFID

Con il metodo attivaColonninaByGetRequest(), dopo aver passato tutti i controlli del metodo precedentemente descritto, viene effettuata una chiamata http al servizio di attivazione di ricarica, attivabile grazie al token di autorizzazione.

```java
@RequestMapping(value = "/attivaColonnina", method = RequestMethod.GET)
public ResponseEntity attivaColonninaByGetRequest(HttpServletRequest request,
                                @RequestParam String rfid,
                                @RequestParam String chargeStationId) {
    RestTemplate restTemplate = new RestTemplate();
    HttpHeaders headers = new HttpHeaders();
    String apiKey = getAutenticationToken();
    headers.set("x-auth-apiKey", apiKey);
    headers.setContentType(MediaType.APPLICATION_JSON);
    String urlPostAttivazione = "https://restation.eu/restation-be/api/private/cs/reservations/reserve";
```

Figura 3.11: Metodo attivaColonninaByGetRequest

La chiamata sopra riportata richiede quindi un metodo di autenticazione attivabile mediante un token che deve essere recuperato tramite la seguente chiamata per effettuare la richiesta di attivazione.

```java
public String getAutenticationToken() {
    RestTemplate restTemplate = new RestTemplate();
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    String urlLogin = "https://restation.eu/restation-be/api/v1/auth/login";
```

Figura 3.12: Metodo getAutenticationToken

Il metodo attivaColonnina(), richiamato dal metodo checkUUID, prende come argomento il codice RIFD e l' id della stazione di ricarica. Dopo di che viene fatto un controllo if-else dove se la colonnina è in stato KO, registro il problema di autenticazione a Restation e quindi l'impossibilità di effettuare la ricarica. Se invece lo stato è OK effettuo la chiamata http e la colonnina viene attivata.

```java
public void attivaColonnina(String rfid, String chargeStationId) {
    RestTemplate restTemplate = new RestTemplate();
    HttpHeaders headers = new HttpHeaders();
    String apiKey = getAutenticationToken();
    if (apiKey.equals("KO")) {
        logAttivazioneRFIDRepository.save(new LogAttivazioneRFID(rfid, chargeStationId, stato: "KO",
                                    responeRestation: "Problemi nell'autenticazione a Restation", new Date()));
        return;
    }

    headers.set("x-auth-apiKey", apiKey);
    headers.setContentType(MediaType.APPLICATION_JSON);
    String urlPostAttivazione = "https://restation.eu/restation-be/api/private/cs/reservations/reserve";
```

Figura 3.13: Metodo attivaColonnina

3.3 Implementazione dell'interfaccia

L'interfaccia del software è stata creata con la libreria Javascript React.js. Il file che ho creato per l'implementazione è GestioneUtenteRicarica.js
All'interno di questa classe sono andata a creare un metodo per la rappresentazione grafica della tabella, inoltre ho richiamato i metodi creati nel file ServiceProvider.js per poter prelevare i dati dal database e inserirli nelle rispettive righe e colonne della tabella di seguito riportata per una corretta e completa visualizzazione dei dati.

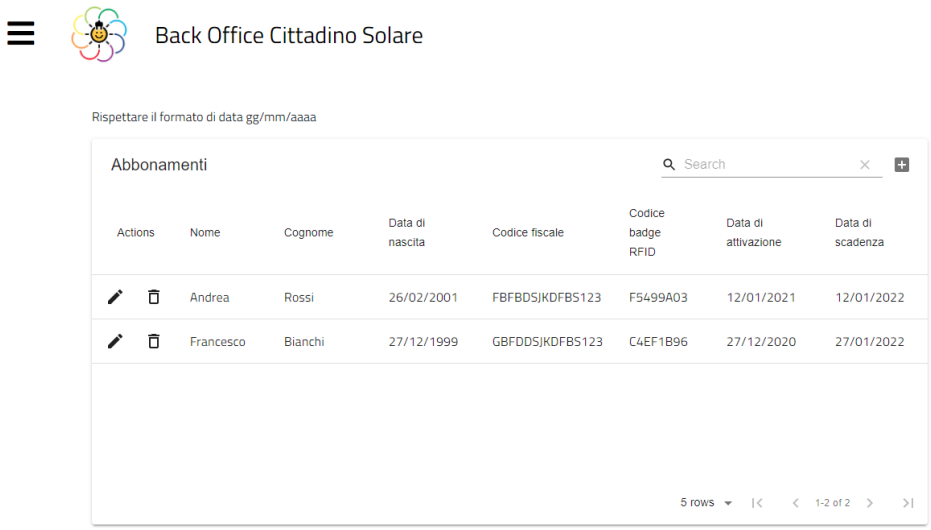

Figura 3.14: Tabella della visualizzazione utenti

Cliccando su questo pulsante l'amministratore ha la possibilità di aggiungere un nuovo utente abbonatosi al servizio di ricarica dei veicoli elettrici.

Cliccando su questo pulsante l'amministratore ha la possibilità di modificare i dati di un utente precedentemente inseriti nella tabella, cliccando sul campo che decide di modificare. Dopo di che ha due possibilità:

- Cliccando sulla spunta i campi modificati verranno salvati nel database e andranno a sostituire quelli precedentemente inseriti.

- ✕ Cliccando sulla X le modifiche che si desiderava effettuare verranno annullate e i dati inseriti rimarranno gli stessi.

🗑 Cliccando su questo pulsante l'amministratore ha la possibilità di eliminare i dati di un utente presente nella tabella ed escluderlo quindi dal servizio di ricarica.

PARTE IMPLEMENTATIVA

Come prima cosa dichiariamo nel costruttore della classe l' API_ENDPOINT, cioè l'indirizzo di destinazione dei metodi. Tale indirizzo è quello di produzione del software. Inizializziamo quindi la libreria axios con l' url dei servizi.

```
constructor() {
  let API_ENDPOINT = "https://app-csl.e-pal.it/api/";

  this.log = (prompt, log) => {
    if (process.env.NODE_ENV === "development") {
      console.log(prompt, log);
    }
  };

  this.axiosCSL = axios.create({
    baseURL: API_ENDPOINT,
  });
```

Figura 3.15: Costruttore della classe GestioneUtenteRicarica.js

Il metodo che segue viene utilizzato per la visualizzazione degli utenti presenti nel database. Tramite la libreria JavaScript Axios vado a richiamare il metodo viewUtente-Ricarica presente nella parte backend.
Grazie al metodo .get della libreria richiamo il valore "getUtentiRicarica" dell'annotazione @RequestMapping del metodo viewUtenteRicarica precedentemente descritto e gli passo il token di autorizzazione. Se l'esecuzione di tale metodo va a buon fine verrà restituita la risposta alla chiamata tramite il metodo .then, altrimenti catturo l'errore e restituisco il log("errore caricamento utenti ricarica", error) tramite la dicitura .catch.

```
getUtentiRicarica(token){
  return this.axiosCSL
  .get("getUtentiRicarica", {
    headers: {
      Authorization: token,

    },
  })
  .then((response) =>{
    return response;
  })
  .catch((error) => {
    this.log("errore caricamento utenti ricarica", error);
    alert("errore");
  })
}
```

Figura 3.16: Metodo dell'interfaccia per visualizzare utenti

Il seguente metodo viene utilizzato per l'inserimento di un nuovo utenti nel database. Tramite il metodo .post("sendUtenteRicarica, data, ...) richiamiamo il metodo sendUtenteRicarica del backend e gli passiamo i dati. Inoltre come headers gli passiamo il token di autorizzazione. Se la chiamata del metodo è autorizzata e va a buon fine il nuovo utente viene registrato nel database altrimenti verrà restituito l'avviso di cattura dell'errore con il log("errore invio utente", error).

```
sendUtenteRicarica(token,data){
    return this.axiosCSL
  .post("sendUtenteRicarica",data, {
      headers: {
          Authorization: token,
      },
  })
  .then((response) =>{
    return response;
  })
  .catch((error) => {
    this.log("errore invio utente", error);
    alert("errore");
  })
}
```

Figura 3.17: Metodo dell'interfaccia per la registrazione di nuovi utenti

Tramite il seguente metodo invece posso modificare i dati presenti nel database. Con il metodo .put della libreria axios prendo il metodo del backend "updateUtenteRicarica", l'id relativo all'utente selezionato e i dati in formato JSON. Dopo aver verificato il token di autorizzazione con il metodo .then facciamo ritornare la risposta e quindi l'aggiornamento effettivo dei dati. Al contrario viene restituito l'errore di aggiornamento.

```
updateUtenteRicarica(token,id,data){
  return this.axiosCSL
  .put("updateUtenteRicarica/"+id, data,{
    headers: {
      Authorization: token,

    },
  })
  .then((response) =>{
    return response;
  })
  .catch((error) => {
    this.log("errore update", error);
    alert("errore");
  })
}
```

Figura 3.18: Metodo dell'interfaccia per aggiornare i dati dell'utente

Il seguente metodo è utile alla cancellazione dell'utente dal database. La dicitura .delete() permette di cancellare l'utente dal database, grazie al metodo del backend "deleteUtenteRicarica" e all' id dell'utente selezionato. Come per gli altri metodi se l'autorizzazione va a buon fine avviene la cancellazione, al contrario viene mostrato un messaggio di errore per notificare la mancata eliminazione dei dati dal DB.

```
deleteUtenteRicarica(token,id){
  return this.axiosCSL
  .delete("deleteUtenteRicarica/"+id, {
    headers: {
      Authorization: token,

    },
  })
  .then((response) =>{
    return response;
  })
  .catch((error) => {
    this.log("errore update", error);
    alert("errore");
  })
}
```

Figura 3.19: Metodo dell'interfaccia per la cancellazione degli utenti

Con il codice riportato nelle immagini successive procedo con la costruzione effettiva della tabella. Per prima cosa dichiaro i nomi dei campi della tabella con l' inizializzazione di var columns in cui vado a dichiarare il titolo della colonna e il relativo valore con cui deve essere riempita la tabella.

```
var columns = [
  {title: "id", field: "id", hidden: true},
  {title: "Nome", field: "nome"},
  {title: "Cognome", field: "cognome"},
  {title: "Data di nascita", field: "dataDiNascita"},
  {title: "Codice fiscale", field: "codiceFiscale"},
  {title: "Codice badge RFID", field: "codiceBadge"},
  {title: "Data di attivazione", field: "dataAttivazione"},
  {title: "Data di scadenza", field: "dataScadenza", placeholder: "gg/mm/aaaa"},

]
```

Figura 3.20: Inizializzazione colonne della tabella

Dopo di che con il tag <MaterialTable> vado a costituire la parte grafica della tabella e a popolarla con i dati prelevati dai metodi precedentemente descritti, i quali a loro volta vengono utilizzati in onRowUpdate, onRowAdd e onRowDelete, metodi precedentemente creati. Prima di richiamare questi ultimi metodi citati vado a riempire la tabella con gli abbonati presenti nel database tramite il metodo useEffect() che richiama il metodo getUtentiRicarica dal serviceProvider, utile per ricavare tutti gli utenti presenti nel DB.

```
useEffect(() => {
  serviceProvider.getUtentiRicarica(props.token)
      .then(res => {setData(res.data)})
      .catch(error=>{console.log("Error")})
}, [])
```

Figura 3.21: Riempimento tabella

```
<MaterialTable
title="Abbonamenti"
columns={columns}
data={data}
icons={tableIcons}
editable={{
  onRowUpdate: (newData, oldData) =>
    new Promise((resolve) => {
      handleRowUpdate(newData, oldData, resolve);
    }),
  onRowAdd: (newData) =>
    new Promise((resolve) => {
      handleRowAdd(newData, resolve)
    }),
  onRowDelete: (oldData) =>
    new Promise((resolve) => {
      handleRowDelete(oldData, resolve)
    }),
  }}
/>
```

Figura 3.22: Creazione tabella

Capitolo 4

Validazione

Per verificare l'effettivo funzionamento dei metodi implementati ci siamo serviti di Swagger, un linguaggio di descrizione dell'interfaccia utilizzato per descrivere le API RESTful espresse utilizzando JSON[29]. Esso viene quindi utilizzato per creare, progettare, documentare e utilizzare i servizi Web RESTful. Ma il mio utilizzo principale è stato quello di interazione e documentazione con le API.

Grazie a tale linguaggio abbiamo potuto verificare volta per volta l'effettivo funzionamento dei metodi creati nel lato server tramite l'interfaccia dei test.
I metodi implementati non sempre sono risultati corretti al primo colpo, infatti sono stati diversi i codici di stato del protocollo di HTTP[1] che si sono presentati durante la fase di testing.
Tra cui: 400 Bad Request, il quale si verifica quando la richiesta genera un errore in seguito ad errori di sintassi. 401 Unauthorized, appare quando l'autenticazione è possibile ma fallisce o non può essere fornita. 403 Forbidden, quando la richiesta è consentita ma il server non approva la sua soddisfazione. 500 Internal Server Error, si verifica quando è presente un errore generico senza fornirne i dettagli. Se invece l'esecuzione del test è andato a buon fine verrà restituita la risposta 200 OK.

Di seguito sono riportati due esempi di verifica di funzionamento dei metodi implementati a lato server, che sono l'inserimento di un nuovo utente e la cancellazione di un utente già esistente nel database.

Come prima cosa per effettuare i test bisogna fare login tramite l'apposito metodo per ricavare il token di autenticazione tramite il quale possiamo effettuare i test, in quanto è richiesto dalle altre chiamate http per poter autorizzare l'operazione.

[1]Risposta generata dal server ad una richiesta del broswer

Figura 4.1: Login per l'autenticazione

Poichè la richiesta è andata a buon fine ci verrà restituito il token di autenticazione nel Response Body e il Responde Code di 200.

4.1 Test inserimento nuovo utente

Per effettuare il test di inserimento di un nuovo utente inserisco nel value i dati del nuovo utente che si vuole aggiungere in formato JSON, dopo di che inserisco il token di autenticazione, precedentemente ricavato dal login e verifico il funzionamento del metodo.

Figura 4.2: Metodo dell'interfaccia per la cancellazione degli utenti

Come possiamo notare dall'immagine seguente il Responde Code della richiesta è 200, quindi si può affermare che la richiesta di inserimento è andata a buon fine e l'utente è stato aggiunto alla tabella del DB.

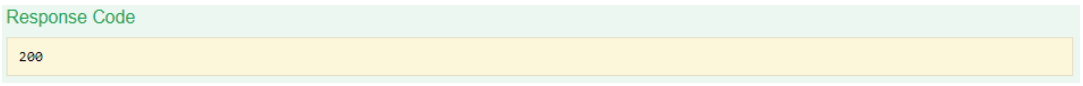

Figura 4.3: Status code dell'inserimento di un nuovo utente

4.2 Test cancellazione utente

Per effettuare il test di cancellazione di un utente dal database inserisco il numero del'id dell'utente che si vuole eliminare nel Value e il token di autorizzazione fornitoci dal metodo di login, nei corrispettivi spazi.

Figura 4.4: Metodo dell'interfaccia per la cancellazione degli utenti

Dall'immagine che segue possiamo capire che la cancellazione dell' utente che ha come id quello descritto nel Response Body è andata a buon fine, in quanto il Response Code della richiesta http è 200.

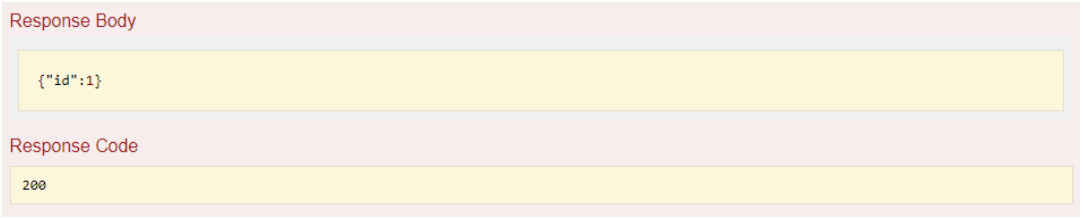

Figura 4.5: Metodo dell'interfaccia per la cancellazione degli utenti

4.3 Test di attivazione colonnina

Per verificare i metodi di richiesta di attivazione delle colonnine, da parte del cittadino solare, è stata utilizzata la piattaforma Restation, poichè è tramite questo servizio esterno fornito dall'azienda Nrg4You che la colonnina viene attivata.

Utilizzando il metodo checkUUID del server precedentemente descritto verifico i stati della tassera RFID posseduta da un'utente registrato al servizio e lo stato della colonnina di ricarica in quel momento.
Le due immagini seguenti mostrano che la colonnina è attiva e che nella prima immagine lo stato della colonnina è FREE cioè libera e non richiesta per l'uso.

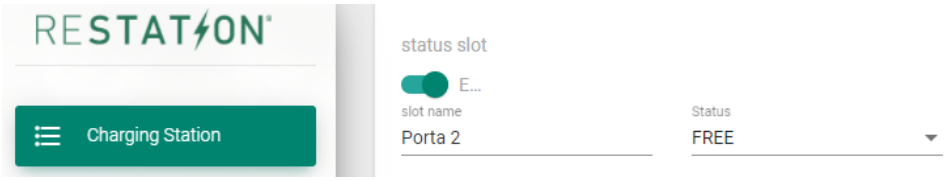

Figura 4.6: Colonnina attiva con stato FREE

Mentre nella seconda immagine, la colonnina è sempre accesa ma lo stato è ENABLE_CHARGE. Ciò significa che è pronta per abilitare la ricarica.

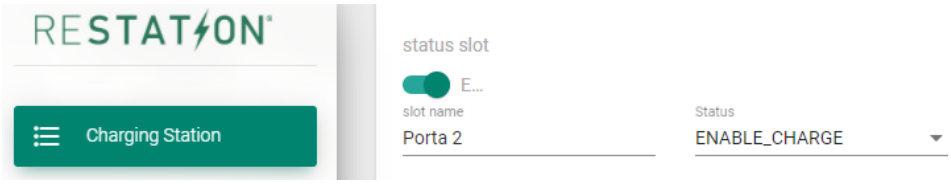

Figura 4.7: Colonina attiva con stato ENABLE-CHARGE

Quindi quando lo stato è settato a FREE il metodo da l'OK per passare ad ENABLE_CHARGE e abilitare l'uso della colonnina. Se invece lo stato è già settato ad ENABLE_CHARGE quando si vuole attivare la colonnina il server darà una risposta negativa per la sua attivazione in quanto è già in uso. Di seguito è riportata la tabella con le possibili risposte fornitoci dal server in seguito alla richiesta di attivazione di una colonnina per la ricarica del veicolo.

id [PK] bigint	charge_stationid character varying (255)	date timestamp without time zone	respone_restation character varying (1000)	rfid character varying (255)	stato character varying (255)
23	RS2021CSLTEST2	2021-02-23 13:04:49.444		F5499A03	ABBONAMENTO SCADUTO
24	RS2021CSLTEST2	2021-02-23 13:05:55.536	org.springframework.web.clien...	F5499A03	KO
25	RS2021CSLTEST2	2021-02-23 13:07:02.986	{"reservationKey":"b23c5ee3-0a...	F5499A03	OK
26	RS2021CSLTEST2	2021-02-23 15:32:19.861		F5499A043	TESSERA NON ASSEGNATA A...

Figura 4.8: Tabella attivazione colonnina

La tabella sopra riportata ci fornisce i risultati della richiesta di attivazione della colonnina nella colonna 'stato'. Nel primo caso possiamo notare che la risposta è ABBONAMENTO SCADUTO, ciò significa che la tessera RFID che si vuole utilizzare per la ricarica è scaduta.
Nella seconda riga notiamo invece che la risposta è KO. Ciò vuol dire che lo stato della colonnina è in ENABLE_CHARGE e quindi la colonnina è già occupata e se ne dovrà richiedere un'altra se si vuole effettuare una ricarica.
Nella terza riga invece la risposta ricevuta dal server è OK, quindi la ricarica può essere effettuata in quanto lo stato della colonnina è a FREE e può passare a ENABLE_CHARGE. Nell' ultima riga invece lo stato risultante è "TESSERA NON ASSEGNATA A NESSUN ABBONATO" , ciò significa che la tessera RFID con quel codice non esiste all'interno del database e di conseguenza non è assegnata a nessuno.

Capitolo 5

Conclusione

Questo elaborato tratta dello sviluppo di un software per la gestione delle colonnine elettriche per la ricarica di veicoli green. Tale software, commissionato dalla Comunità Solare è stato creato per fornire un servizio da parte di questi ultimi per chi entra a far parte delle Comunità Solari Locali nel territorio dell' Emilia-Romagna.
I membri di questa associazione oltre ad usufruire di queste colonnine di ricarica per caricare il proprio mezzo elettrico, fanno parte di una grande community il cui obbiettivo principale è quello di vivere in un mondo alimentato da energia rinnovabile.

Durante la stesura dell'elaborato ho trattato diversi temi appartenenti all' ambito della mobilità elettrica. Per lo più i motivi riguardanti la necessità di una conversione verso un mondo green, la situazione delle stazioni di ricarica in Italia e la precarietà a cui vanno incontro gli automobilisti che decidono di sostenere la mobilità elettrica. Sono state esposte le tipologie di dispositivi presenti sul mercato per la ricarica del veicolo e il loro tipo di uso, che può essere pubblico o privato. Inoltre vengono presentati alcuni software di gestione già presenti nel mercato con il loro funzionamento.
In particolare e più nel dettaglio è stata esposta la progettazione, l'implementazione e la validazione del progetto svolto da me e il team del gruppo FINMATICA S.P.A. a cui sono stata assegnata per svolgere le ore di tirocinio per la produzione dell'applicazione della gestione delle colonnine elettriche.

5.1 Sviluppi futuri

Per quanto riguarda i sviluppi futuri sulle colonnine elettriche ci sono vari progetti in ballo, uno di questi è quello ideato dalla Germania, il progetto "FastCharge", il quale prevede la diffusione di un sistema già messo alla prova, di una stazione di ricarica super veloce che permette di caricare il veicolo in in pochissimo tempo.

É stato già provato, su delle macchine marchiate BMW e Porsche, che tale sistema permette di portare l' autonomia della macchina da 0km a 100km in soli 3 minuti.
Altro progetto è quello di un sistema di ricarica wireless, possibile grazie ad una tecnologia induttiva però ancora non molto diffusa.
Direttamente dal politecnico di Torino invece è nato un progetto chiamato "Charge While Driving", il quale consiste in un impianto fatto di piastre induttive posizionate sulla pavimentazione stradale che permette il caricamento della vettura durante la marcia.
Altro progetto ancora è ideato da un start-up di Bologna chiamata Battery che con un progetto chiamato Nessox (New Semi-Solid flow lithium Oxygen Battery) sta ideando un fluido caricato di elettricità utilizzabile come un normale carburante ma che dopo essersi scaricato potrebbe essere riutilizzabile per essere di nuovo caricato [41].
Tali innovazioni richiedono tempo e denaro per la loro realizzazione, per ciò ci vorrà molto tempo perché esse vengano messe in funzione e a disposizione di tutta la popolazione.

Per quanto riguarda lo sviluppo del software nel tempo si sta progettando di costruire un sistema che permetta il pagamento della ricarica direttamente dall' applicazione della Comunità Solare creata dall' azienda. Tale sistema permetterà quindi di pagare l'energia direttamente nel momento della ricarica senza dover possedere necessariamente un abbonamento.

Ringraziamenti

Vorrei ringraziare in primis la mia famiglia che mi ha sostenuto in tutto e per tutto sin dal primo anno in cui ho deciso di intraprendere questo percorso, poi vorrei ringraziare il professore Marco Di Felice e il gruppo FINMATICA S.P.A., in particolare il mio tutor Fabrizio Crinò, che mi hanno sostenuto e permesso di svolgere questo elaborato.
In fine non per meno importanza i miei colleghi di corso, con cui ho legato ogni anno sempre di più, i miei amici di sempre e le mie coinquiline.

Questi tre anni sono stati un mix di gioie, risate e vittorie, ma anche sconfitte e pianti. Non sempre sono stati facili, tra le mille ansie, pensieri, angosce e arrabbiature c'era sempre qualcuno di loro a sostenermi e a darmi il coraggio di andare avanti ed è grazie anche a loro se ho raggiunto questo importantissimo traguardo.

Ringrazio, e non per ordine di importanza, Hajar per il suo umorismo la voglia di far festa e la disperazione condivisa che non ti fa sentir sola, Stefania che anche se ci siamo conosciute solo un anno fa ha saputo farmi forza soprattutto in questo ultimo periodo con la sua positività e gentilezza. Geralda con cui ho condiviso molti momenti sia tristi che felici e ha sempre avuto una parola di conforto nei mie confronti, il mio compagno fedele di studi Alessandro che ha sempre trovato un modo per farmi ridere nei momenti tristi e di particolare ansia, insegnandomi a dare il giusto peso alla cosa. Alex che con il suo buon umore e buon cuore ha saputo rendere divertente anche una situazione di particolare tensione. Giovanni che con la sua parlantina ti coinvolge nella sua vita e con cui ho condiviso le ansie degli ultimi giorni.

Vorrei ringraziare Sam, Eugenio, Andrea e Jimmy perchè ci sono sempre stati quando avevo bisogno di svagarmi e con la loro simpatia e voglia di vivere hanno saputo tirarmi su di morale.

Ringrazio le mie amiche di sempre, Giulia, Lucia, Caterina e Eleonora che hanno intrapreso l' avventura universitaria con me decidendo di cambiare città e andare a vivere insieme, rendendo l'inizio di una nuova vita più facile e divertente. Le ringrazio perché sono e saranno sempre un pilastro della mia vita su cui posso contare.

Ringrazio le mie coinquiline Asia e Alice che hanno dovuto sopportare i miei sbalzi d'umore di questi ultimi mesi cercando di rallegrarmi le giornate.

Infine ringrazio tutti coloro che nel loro piccolo hanno reso questi ultimi anni speciali.

Bibliografia

[1] *Auto elettriche: i modelli e le tecnologie*
https://www.ilsole24ore.com/art/auto-ibride-raddoppio-mentre-elettriche-triplicano-vendite-europa-ADzed1IB

[2] *Come funzionano ecobonus ed ecotassa sulle auto*
https://www.altroconsumo.it/auto-e-moto/automobili/news/ecotassa-ecobonus-incentivi-auto

[3] *Emissioni di CO2 delle auto: i numeri e i dati. Infografica*
https://www.europarl.europa.eu/news/it/headlines/society/20190313STO31218/emissioni-di-co2-delle-auto-i-numeri-e-i-dati-infografica

[4] *Effetto serra: cos'è, cosa comporta e come contenerlo*
https://anteritalia.org/effetto-serra-cose-cosa-comporta-contenerlo/

[5] *Auto elettriche: pro e contro*
https://www.sorgenia.it/guida-energia/auto-elettriche-pro-e-contro

[6] *Colonnine elettriche: situazione aggiornata in Italia*
https://www.leaseplan.com/it-it/news-auto/colonnine-elettriche-italia/

[7] *Colonnina di ricarica elettrica, come funziona*
http://motori.quotidiano.net/comefare/colonnina-ricarica-elettrica-funziona.htm#:~:text=La%20colonnina%20per%20la%20ricarica,palo%2C%20a%20parete%20o%20portatili.

[8] *Smartbit*
https://smartbit.io/

[9] *Software as a service*
https://it.wikipedia.org/wiki/Software_as_a_service

[10] *THOR, Il sistema gestione stazioni di ricarica veicoli elettrici*
https://thor.tools/

[11] *Smartbit - thor sistema gestione ricarica veicoli elettrici*
https://www.autoyas.com/IT/Cuneo/1017844198322246/Smartbit---THOR-sistema-gestione-ricarica-veicoli-elettrici

[12] *Cloud EMOBITALY*
https://www.stazioni-di-ricarica.it/emobitaly/

[13] *Auto elettriche: ecco la situazione delle colonnine per la ricarica*
https://www.autoblog.it/post/auto-elettriche-ecco-la-situazione-delle-colonnine-per-la-ricarica

[14] *Infrastrutture di ricarica elettrica in Italia: a che punto siamo?*
https://www.dealerlink.it/infrastrutture-ricarica-auto-elettrica-italia-dati-diffusione-motus-e/

[15] *MOTUS-E*
https://www.motus-e.org/chi-siamo/mission

[16] *Spring Framework*
https://it.wikipedia.org/wiki/Spring_Framework

[17] *Spring Boot*
https://www.nextre.it/spring-boot-un-nuovo-livello-di-semplicita-per-produrre-e-testare-java-app/

[18] *React*
https://it.reactjs.org/

[19] *React.Js*
https://www.html.it/pag/55052/react-introduzione/

[20] *FINMATICA S.P.A.*
https://www.ads.it/

[21] *Radio-frequency identification*
https://it.wikipedia.org/wiki/Radio-frequency_identification

[22] *Comunità Cittadino Solare Locale*
https://comunitasolare.eu/

[23] *Spin-off universitario*
https://www.imprenditoreglobale.com/spin-off-aziendale-spin-off-universitario/

[24] *EmiliaRomagnaSTARTUP*
https://www.emiliaromagnastartup.it/it/innovative/imprese/comunit-solare-locale

[25] *Ricaricare l'auto elettrica: dove, come e quanto costa*
Rhttps://www.qualenergia.it/articoli/ricaricare-lauto-elettrica-dove-come-e-quanto-costa/

[26] *Nrg4You*
https://www.nrg4you.it/

[27] *Cos'è la scale-up e quali sono le sue caratteristiche*
https://www.startupbusiness.it/cose-la-scaleup/88836/

[28] *Axios*
https://rapidapi.com/blog/axios-react-api-tutorial/#:~:text=%20How%20to%20Build%20an%20Application%20with%20Axios,or%20follow%20this%20link%20to%20the...%20More

[29] *Swegger(Software)*
https://en.wikipedia.org/wiki/Swagger_(software)

[30] *@PreAuthorize e @PostAuthorize in Spring Security*
https://www.concretepage.com/spring/spring-security/preauthorize-postauthorize-in-spring-security

[31] *Annotazioni Spring Framework*
https://springframework.guru/spring-framework-annotations/

[32] *Auto elettrica: la mappa dei punti rifornimento in Italia. Foto*
https://motori.virgilio.it/auto/auto-elettrica-mappa-dei-punti-rifornimento-in-italia-foto/344/

[33] *Organizziamoci con Maven*
https://codingjam.it/organizziamoci-con-maven-parte-i/#:~:text=Il%20POM%20%C3%A8%20un%20file,come%20la%20generazione%20di%20documentazione.

[34] *Che cos'è un database relazionale?*
https://www.oracle.com/it/database/what-is-a-relational-database/

[35] *Che cos'è un database?*
https://www.oracle.com/it/database/what-is-database/

[36] *INTERNET of THINGS (IoT) Significato, esempi, ambiti applicativi e prospettive di mercato in Italia*
https://blog.osservatori.net/it_it/cos-e-internet-of-things#iot-significato

[37] *STORIA DELLA MACCHINA ELETTRICA: DAL 1800 AD OGGI*
https://www.e-vai.com/storia-della-macchina-elettrica-dal-1800-ad-oggi/

[38] *Wall Box: cos'è, quanto costa e dove installarla*
https://www.leaseplan.com/it-it/news-auto/wall-box-cos-e/

[39] *Guide to @JsonFormat in Jackson*
https://www.baeldung.com/jackson-jsonformat

[40] *Spring's RequestBody and ResponseBody Annotations*
https://www.baeldung.com/spring-request-response-body

[41] *Colonnine di ricarica e auto elettriche: un futuro strettamente connesso*
https://smartparkingsystems.com/colonnine-di-ricarica-e-auto-elettriche-un-futuro-strettamente-connesso/

[42] *Tutorial su Spring Boot JpaRepository*
https://zetcode.com/springboot/jparepository/

[43] *Pdf sul servlet*
http://www.ce.uniroma2.it/~lopresti/Didattica/RetiWeb/RetiWeb0809/Servlet.pdf

[44] *Gewiss JOINON, una soluzione a 360° per la e-mobility*
https://www.pcprofessionale.it/news/tech/automotive/gewiss-joinon-soluzione-360-per-e-mobility/

[45] *Le 5 migliori App per trovare le colonnine elettriche di ricarica*
https://mobilitasostenibile.it/le-5-migliori-app-per-trovare-le-colonnine-elettriche-di-ricarica/

[46] *Software per la gestione della ricarica e dei carichi*
https://www.phoenixcontact.com/online/portal/it?1dmy&urile=wcm:path:/itit/web/main/products/subcategory_pages/Charging_management_software_P-29-05/9e6b10f4-6bd8-4275-9c30-143f3e0d0690

[47] *https://www.vaielettrico.it/ricarica-caccia-al-tesoro-in-dodici-app/*
https://www.vaielettrico.it/ricarica-caccia-al-tesoro-in-dodici-app/