Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Un servizio di route planning per scenari di mobilità elettrica

Relatore:
Chiar.mo Prof.
Luciano Bononi

Presentata da:
Marco Di Nicola

Correlatori:
Dott. Luca Bedogni
Dott. Marco Di Felice

Sessione I
Anno Accademico 2013/2014

Abstract

Nelle *smart cities* moderne, la mobilità di veicoli elettrici (EV) è considerata un fattore determinante nella riduzione del consumo di combustibili fossili e conseguenti emissioni inquinanti.
Tuttavia, nonostante gli interessi e investimenti a livello globale, l'accettazione da parte degli utenti è ancora bassa, principalmente a causa della mancanza di infrastrutture e servizi a supporto dei guidatori di EV. Queste mancanze sono la causa principale della cosiddetta *range anxiety* (timore che il veicolo non abbia autonomia sufficiente per raggiungere la destinazione) e hanno portato al preconcetto che gli EV siano adatti alla sola percorrenza di brevi tragitti.
Per contrastare questi problemi, in questo documento è proposta un'applicazione di *route planning* che supporti mobilità di EV anche su percorsi medio-lunghi, mediante utilizzo di un modello di predizione del consumo energetico e considerazione dell'eventuale necessità di ricarica.
Saranno descritte tecniche per determinare il tragitto che un EV sia in grado di percorrere per arrivare a destinazione, in considerazione di restrizioni energetiche, fattore altimetrico del percorso ed eventuali operazioni di ricarica necessarie.
Il modello di consumo e l'algoritmo che determina il miglior percorso (dal punto di vista energetico) sono implementati da un web service che interagisce con i servizi di Google Maps (per ottenere indicazioni stradali, dati altimetrici e informazioni in tempo reale sul traffico) e con servizi che offrono informazioni sulle stazioni di ricarica e relative posizioni.
Dopo aver descritto il modello di consumo e l'algoritmo per la ricerca del percorso, sarà presentata l'architettura del servizio implementato.
Sarà quindi fornita una valutazione del servizio, analizzandone performance e scalabilità, nonché l'efficacia nel supporto di percorsi di EV all'interno di scenari su larga scala (nello specifico la regione Emilia Romagna), attraverso tecniche di simulazione.

Indice

Introduzione

La mobilità veicolare elettrica costituisce uno degli elementi principali all'interno dell'ecosistema delle cosiddette *smart cities*: centri urbani che, mediante integrazione di tecnologie dell'informazione e comunicazione con infrastrutture a supporto dell'ambiente e dell'efficienza energetica, rappresentano la nuova frontiera della sostenibilità sociale ed ambientale.
Questo settore ha riscosso interessi ed investimenti da numerosi produttori di automobili [BCMW11], incentivati dalla prospettiva di ridurre le emissioni di gas clima-alteranti (CO2) e fornire una valida alternativa al mercato di combustibili fossili.
Secondo recenti rapporti del U.S. Department of Energy, il numero di veicoli elettrici (EV) registrati nel periodo che va dall'inizio del 2010 alla fine del 2012, negli Stati Uniti, è aumentato del 200%. Il grafico in figura 1 mostra come questo numero sia aumentato con la stessa velocità nel corso degli ultimi 2 anni.
Nonostante queste incoraggianti statistiche e l'introduzione sul mercato di una vasta gamma di modelli di EV da parte di diverse case automobilistiche, la scarsa presenza di infrastrutture e servizi a supporto del loro utilizzo [MLA+11] e la conseguente bassa accettazione da parte degli utenti rendono difficile immaginare una diffusione di queste tecnologie su larga scala.
In base ad un altro rapporto del U.S. Department of Energy [Lab11], il 70% di coloro che non possiedono un veicolo elettrico non lo acquisterebbero a causa della cosiddetta *driver range anxiety*, percepita come il timore che il veicolo non abbia autonomia sufficiente per raggiungere la destinazione e che lungo il tragitto non vi siano abbastanza stazioni di ricarica (o *Electrical Vehicle Supply Equipment*, EVSE) disponibili.

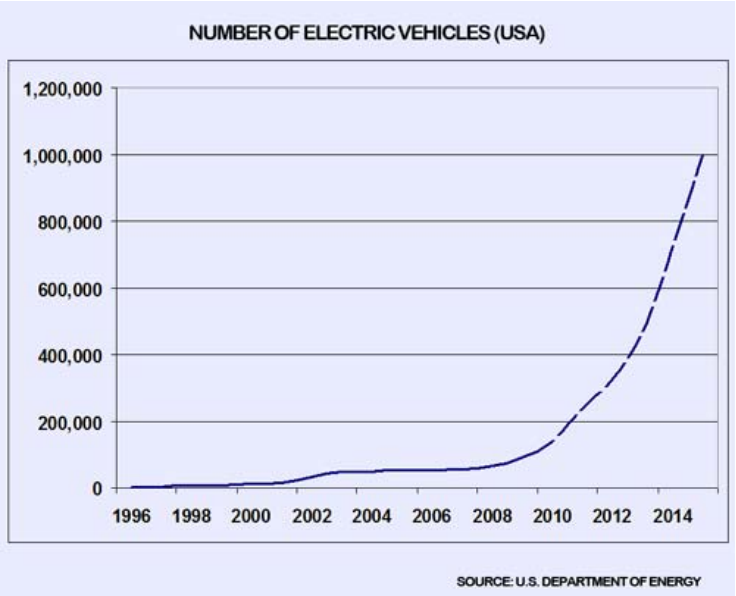

Figura 1: Numero di EV registrati negli USA nel corso degli ultimi anni

Un'altra indagine, condotta in [NHHB13], mostra come la mancanza di informazioni accurate sul consumo energetico del veicolo e la distanza percorribile con l'attuale stato di carica della batteria porti il guidatore di un EV ad alterare il proprio stile di guida e la selezione del percorso fino alla sua destinazione.
In aggiunta alla scarsità di infrastrutture e limitata autonomia delle batterie, uno dei più noti ostacoli alla diffusione della mobilità veicolare elettrica è costituito dai problemi relativi alle operazioni di ricarica, in particolar modo il tempo necessario a portarle a termine, di gran lunga superiore a quello richiesto per veicoli a combustibili fossili.
In merito a quest'ultimo aspetto, è possibile distinguere tre diverse modalità di ricarica, tra le più diffuse:

- *Level 1*: attraverso un cavo a 120 Volt per corrente alternata, richiede un tempo che va dalle 8 alle 12 ore per caricare completamente una batteria scarica.
- *Level 2*: attraverso un cavo a 240 Volt per corrente alternata, richiede dalle 4 alle 6 ore.
- *Level 3*: attraverso un cavo a 480 Volt per corrente continua, richiede in media 30 minuti per una ricarica dell'80%.

La quantità di corrente erogata varia dai 16 ai 32 Ampere.
Nonostante il Level 3 possa sembrare la migliore alternativa, non tutte le batterie dei veicoli attualmente in circolazione lo supportano e l'impiego di un tale voltaggio potrebbe comportare una riduzione della vita utile della batteria.
Una soluzione più recente risiede nelle *battery switch stations*: stazioni nelle quali la batteria del veicolo è semplicemente rimossa e rimpiazzata con una completamente carica (operazione che richiede un tempo inferiore ai 5 minuti); le soluzioni esistenti sono tuttavia proprietarie (Tesla Motors, Mitsubishi, Better Place) e non seguono uno standard preciso nel definire le modalità di scambio e caratteristiche delle batterie.
In sostanza, sono tre i fattori che incidono maggiormente sul comportamento del guidatore:

1. La limitata autonomia di un EV: in media un centinaio di chilometri (sebbene alcuni produttori, come Tesla Motors, stiano promettendo modelli in grado di percorrere $\approx$ 300 chilometri senza ricarica).

2. La bassa densità di stazioni di ricarica (EVSE) in scenari suburbani.

3. La considerevole durata delle operazioni di ricarica.

Queste limitazioni suggeriscono ai potenziali guidatori di EV l'idea che la mobilità elettrica costituisca una soluzione accettabile solo per brevi percorsi all'interno di ambienti urbani, nei quali le infrastrutture disponibili sono ben note e poste a distanze ravvicinate l'una dall'altra.
I problemi non si limitano alle attività dei singoli possessori di EV: diversi studi hanno dimostrato l'impatto critico che la ricarica simultanea di un numero significativo di EV abbia sulla rete energetica. Ne deriva il bisogno di coordinare efficacemente le necessita di ricarica degli EV con la quantità e la dislocazione territoriale degli EVSE, nonché con l'attuale carico della rete di distribuzione energetica.
Al fine di superare le limitazioni menzionate, numerose iniziative (su scala globale) hanno esaminato le potenzialità date dall'applicazione di tecnologie dell'informazione e comunicazione per la realizzazione di servizi specifici per il supporto alla mobilità elettrica.

Sono state proposte numerose applicazioni (mobili e/o accessibili via Web) per assistere i guidatori di EV attraverso diversi servizi di supporto, tra i quali figurano la ricezione di informazioni sulle stazioni di ricarica disponibili lungo il percorso [gre], il monitoraggio remoto dello stato di carica della batteria [car] e la prenotazione di slot temporali per le operazioni di ricarica presso determinate stazioni [BBDF+13].
In quest'ambito, le funzionalità fondamentali per facilitare l'utilizzo di EV su lunghi percorsi e mitigare la *driver range anxiety* sono quelle di predizione del consumo (e conseguente distanza percorribile) e pianificazione dei percorsi (*route planning*). Sono tuttavia pochi i progetti che forniscono una predizione del consumo in tempo reale ([DdV13], [FMA13], [OWB13]) e ancora meno quelli che integrano il *route planning* con considerazioni relative allo stato di carica del veicolo e le opzioni disponibili per la ricarica.
Al fine di colmare queste lacune, questo documento si propone di descrivere la progettazione ed implementazione di un *web service* che assista il guidatore di un EV pianificando il percorso e le operazioni di ricarica, così da permettere di raggiungere la destinazione e al contempo minimizzare il tempo impiegato o il consumo energetico.
Il servizio, realizzato come parte del progetto europeo Internet of Energy [ioe], prende in considerazione fattori quali il modello di EV e batteria in utilizzo, il consumo dei possibili percorsi (determinato in base ad un modello di predizione deterministico e informazioni sul traffico in tempo reale ottenute da Google), l'eventuale necessità di soste per ricarica lungo il tragitto e alcune preferenze utente (ad esempio lo stato di carica che si vuole avere a destinazione).
Il resto del documento è strutturato come segue.
Come prima cosa sarà brevemente presentato lo stato dell'arte di iniziative e servizi volti a promuovere la mobilità veicolare elettrica: il progetto Internet of Energy e i suoi obiettivi principali, nonché un approfondimento su alcune applicazioni analoghe sviluppate negli ultimi anni.
Successivamente sarà descritta la progettazione del servizio realizzato, fornendo una rappresentazione ad alto livello d'astrazione dell'architettura e suoi componenti, discutendo in dettaglio i più importanti aspetti concettuali e gli algoritmi che ne modellano il comportamento.

Seguirà quindi una descrizione del servizio da un punto di vista implementativo: gli strumenti utilizzati per realizzarlo e la struttura dell'applicativo sviluppato e sue funzioni, ponendo enfasi su alcune scelte effettuate e problematiche riscontrate.
Al fine di valutare la scalabilità del servizio realizzato e validarne l'utilizzo, ne saranno analizzate le performance nel supporto di percorsi di EV all'interno di scenari su larga scala (nello specifico la regione Emilia Romagna) e sarà svolto un confronto dei risultati con quelli ottenuti da una variante senza pianificazione, ricavati mediante tecniche di simulazione.
Infine, saranno presentati alcuni sviluppi futuri e proposte di integrazione con altri servizi a supporto della mobilità elettrica.

Parte I

Stato dell'arte

Capitolo 1

Internet of Energy

Avviato dall'Unione Europea, mantenuto dal gruppo *ARTEMIS* (Advanced Research and Technology for Embedded Intelligence and Systems) e comprensivo di 40 partner da 10 nazioni Europee, il progetto Internet of Energy (IoE) for Electrical Mobility [ioe] ha come obiettivo primario quello di realizzare e supportare sviluppo e diffusione su larga scala della mobilità veicolare elettrica in Europa.
Al fine di raggiungere questo obiettivo, il progetto si concentra sullo sviluppo di hardware, software e sistemi middleware per fornire comunicazione sicura e interoperabilità, connettendo le moderne reti elettriche (*smart grid*) con Internet, al fine di creare un'infrastruttura di supporto alla mobilità elettrica.
Sfruttando un budget di 45 milioni di Euro (parte del quale finanziata dalle 40 industrie partner), il progetto mobilita le migliori capacità industriali e di ricerca nei settori di produzione e stoccaggio dell'energia, sistemi embedded e produzione di veicoli elettrici.
La rappresentazione dell'architettura in figura 1.1 mostra alcuni degli attori principali coinvolti nel progetto IoE:

- La cosiddetta *smart grid*: una moderna rete di distribuzione elettrica che, con l'ausilio di tecnologie informatiche di comunicazione, garantisce maggiore efficienza e affidabilità nelle operazioni di stoccaggio e distribuzione dell'energia; comprende al suo interno l'infrastruttura di ricarica degli EV.

- I veicoli elettrici, i quali ottimizzano rendimento e consumo mediante apposite tecnologie e dispositivi hardware e software.

- Gli utenti che interagiscono con la grid e i servizi offerti, utilizzando applicazioni desktop, mobili o integrate a bordo dei veicoli.

- Internet come rete di comunicazione tra la smart grid, i veicoli elettrici e gli utenti, volta a permettere l'ottimizzazione delle operazioni di ricarica delle batterie in accordo ai vincoli imposti dal traffico cittadino e dal rapporto tra consumo e disponibilità di energia elettrica.

- La definizione di standard per garantire l'interoperabilità tra gli attori descritti.

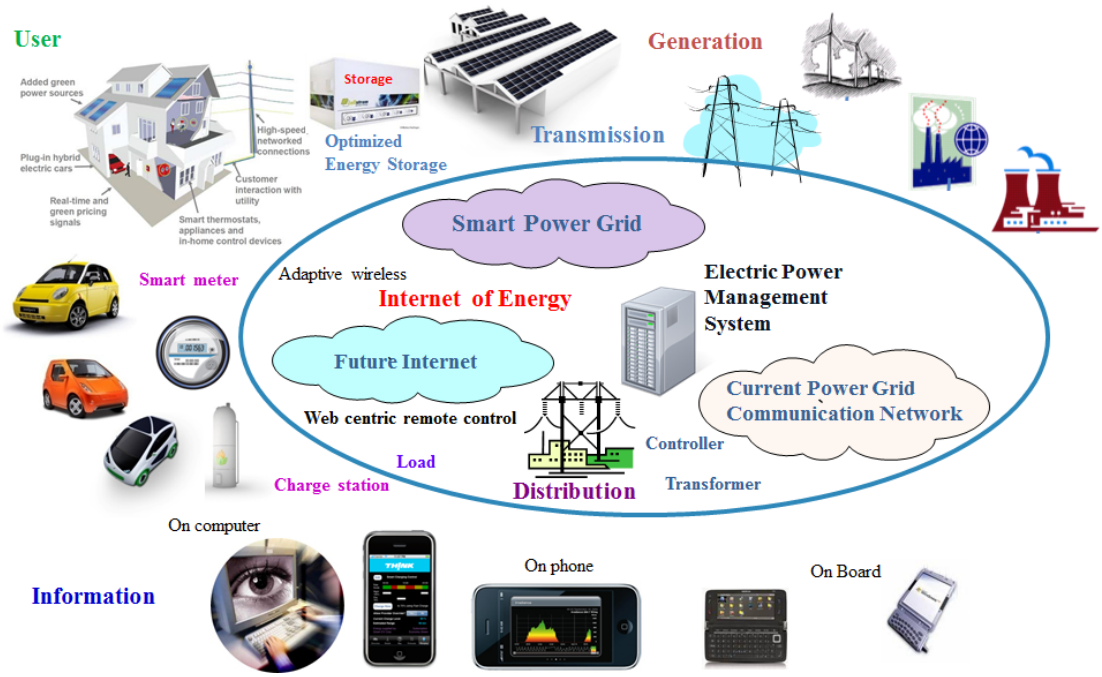

Figura 1.1: Architettura di Internet of Energy

1.1 Servizio di prenotazione delle ricariche

Il contributo al progetto Internet of Energy fornito dall'Università di Bologna, in congiunzione con il centro di ricerca *ARCES* (Advanced Research Center on Electronic Systems), consiste in un servizio di prenotazione di slot temporali per le operazioni di ricarica presso stazioni situate in ambiente urbano (specificatamente nella città di Bologna) e interazione con esso attraverso un'applicazione mobile [BBDF+13].
Questo contributo si compone di tre elementi principali, brevemente descritti a seguire.

1.1.1 City Service

Il *City Service* rappresenta la connessione tra i veicoli elettrici (e relativi guidatori) e la smart grid, ricevendo richieste di prenotazione di ricarica dagli smartphone che eseguono l'applicazione mobile e fornendo una lista delle stazioni disponibili, ordinata secondo le preferenze utente.
I dati necessari al conseguimento di queste operazioni sono rappresentati sotto forma di grafo semantico e definiti su un dominio concettuale di natura ontologica, garantendo agli attori in gioco (di natura eterogenea) la possibilità di sfruttare un vocabolario comune per processare informazioni e coordinare le proprie operazioni.
Il vocabolario ontologico include i concetti di EV, EVSE e relativi connettori di ricarica, batteria, prenotazione, zona geografica, utente, richiesta, risposta e altri, tutti utilizzati per espletare le operazioni di prenotazione delle ricariche.
La persistenza dei dati e la loro condivisione tra le parti interessate è gestita da un repository semantico denominato *Semantic Information Broker* (SIB), realizzato secondo il modello definito dal progetto *Smart-M3*.
Smart-M3 prevede che ogni entità attiva incorpori al suo interno un *Knowledge Processor* (KP), responsabile dell'accesso alle informazioni mantenute dal SIB (attraverso un protocollo applicativo denominato *Smart Space Access Protocol*, basato su messaggi XML) e delle sottoscrizioni che consentono di ricevere notifiche *push* da quest'ultimo al variare dei dati d'interesse.
Si distinguono due SIB fondamentali:

- *City* SIB: utilizzato dal City Service per interagire con gli utenti, fornendo informazioni circa la disponibilità di EVSE e loro caratteristiche, accettando richieste di prenotazione ed eventuali annullamenti, comunicando le opzioni disponibili per la ricarica e confermando la prenotazione.

- *Dash* SIB: utilizzato dalle applicazioni che interagiscono con il veicolo. Idealmente integrato a bordo di ogni veicolo, contiene informazioni relative ad esso quali posizione e stato della batteria.

1.1.2 Simulatore

Al fine di validare l'utilizzo del City Service descritto precedentemente, nonché fornire la possibilità di osservare le dinamiche di consumo dei veicoli in un ambiente virtuale, il terzo contributo è costituito da una piattaforma di simulazione.
Il simulatore ha lo scopo primario di valutare l'impatto dell'introduzione di mobilità veicolare elettrica in un contesto urbano (ad esempio, la città di Bologna), utilizzando tre strumenti principali:

- *Sumo* (Simulation of Urban Mobility): simulatore open source di traffico veicolare. Permette di simulare il movimento di veicoli su reti stradali di dimensioni arbitrarie, create ad hoc o importate a partire da rappresentazioni realistiche ottenute da diverse fonti (ad esempio OpenStreetMap).

- *Omnet++*: framework open source per simulazione ad eventi discreti, principalmente usato per modellare reti di comunicazione e sistemi informativi complessi.

- *Veins* (Vehicles in Network Simulation): framework open source per simulazione di reti veicolari, fornisce interazione bidirezionale tra Sumo ed Omnet++.

Fra gli aspetti modellati nel simulatore, oltre a quelli implicitamente forniti dagli strumenti appena descritti, sono inclusi la carica e scarica della batteria dei veicoli (dipende da parametri relativi ad essa e dalle caratteristiche del relativo veicolo elettrico, quali massa e area della superficie frontale, nonché dal fattore altimetrico del territorio), la necessità per il guidatore di ricaricare quando lo stato di carica scende sotto una certa

soglia (dettata dalla *range anxiety*) e la comunicazione con il City Service (mediante Omnet++) per effettuare prenotazioni.
Tramite gli strumenti per la raccolta di dati offerti da Omnet++, il simulatore consente di analizzare alcuni parametri utili: tempo medio necessario alla ricarica con e senza il servizio di prenotazione, percentuale di veicoli che rimangono senza autonomia e altri.

1.1.3 Applicazione mobile

L'applicazione mobile, sviluppata su piattaforma Android (versione 4.0.3 o superiori), consente di interagire con il City Service per eseguire le operazioni di prenotazione delle slot di ricarica e con il veicolo elettrico per monitorarne i parametri.
In particolare, le funzionalità dell'applicazione consistono nel monitoraggio dei suddetti parametri (stato di carica), possibilità di effettuare una richiesta di prenotazione (a cui segue una conferma o eventuale annullamento), visualizzazione della lista delle stazioni di ricarica disponibili (ordinata secondo parametri quali distanza dalla posizione attuale o da un determinato punto di interesse, prezzo e tempo richiesto) e dello storico di prenotazioni effettuate, lettura di un profilo altimetrico delle strade percorribili per raggiungere un EVSE (utile per selezionare il percorso con il maggior numero di discese o piani, al fine di minimizzare il consumo).
L'applicazione mobile può interagire con un veicolo reale tramite connessione Bluetooth o con un veicolo rappresentato all'interno del simulatore, al fine di poter osservare il consumo e distanza percorribili in un ambiente virtuale.

Capitolo 2

Lavori simili

In questi ultimi anni si è registrato un notevole aumento di servizi web e applicazioni di supporto alla mobilità veicolare elettrica. In particolar modo, in aggiunta a servizi di prenotazione come quello descritto in 1.1, il focus principale rimane quello di facilitare le operazioni di ricarica attraverso servizi di localizzazione di EVSE e fornire supporto nella pianificazione di itinerari, attraverso modelli di previsione del consumo energetico. Dal punto di vista della ricerca sono molti gli studi che puntano a gettare le giuste basi algoritmiche per costruire un adeguato modello di consumo energetico e pianificazione del percorso.
In [AHLS10] è descritta la possibilità di riformulare il problema dei cammini minimi in termini di consumo energetico su un grafo di strade, adattando alcuni dei noti algoritmi (Dijkstra e Bellman-Ford) che risolvono questo problema in tempo polinomiale.
Questo discorso è ampliato maggiormente in [EFS11], aggiungendo considerazioni sullo stato di carica iniziale, capacità massima della batteria e fattore altimetrico dei percorsi, dal quale può derivare un eventuale guadagno energetico (in fase di decelerazione) e conseguenti archi con peso negativo. Questo lavoro propone anche numerose ottimizzazioni nella potenzialmente complessa costruzione del grafo di strade, oggetto del problema.
Lo studio effettuato in [DdV13] aggiunge un sistema di vincoli e una componente stocastica al problema, determinando il cammino minimo come quello che massimizza la probabilità di arrivare a destinazione entro una certa quantità di tempo.

Un approccio simile viene utilizzato in [OWB13], definendo un framework stocastico che usa una catena di Markov per modellare il profilo del guidatore di un veicolo elettrico, al fine di stimare la distanza percorribile con l'attuale stato di carica della batteria.
Per quanto riguarda le applicazioni e i servizi implementati, ne esistono di open source e proprietari, creati e promossi da grandi o piccole aziende e consorzi.
Fra i progetti che offrono servizi web per localizzazione degli EVSE, i più noti e di maggiore diffusione sono CarStations [car], Green eMotion [gre], Enel Drive [ene], Open Charge Map [ope] e PlugShare [plu]. Persino Google, già da un paio di anni, consente di localizzare stazioni di ricarica sulle proprie mappe mediante l'uso di un pattern specifico nella query di ricerca ("ev charging station near x", con x un qualunque luogo espresso in coordinate geografiche o mediante indirizzo).
La stessa funzionalità è offerta da varie applicazioni mobili, tra le quali ChargePoint [Tec] (da Coloumbus Technologies) e Blink Mobile (da ECOItality) [ECO], che permettono di utilizzare la localizzazione tramite GPS dello smartphone per individuare le stazioni più vicine alla propria posizione.
In [SWN13] è fornita una panoramica completa delle applicazioni mobili a supporto dei guidatori di EV, identificando sei casi d'uso principali.
Nonostante la grande quantità di studi e progetti realizzati in questo contesto, il problema di route planning per veicoli elettrici con predizione del consumo rimane poco affrontato, con poche applicazioni pratiche che forniscono una tale funzionalità.
Uno degli sforzi maggiori in questa direzione è probabilmente costituito da quello dell'applicazione proposta in [FMA13]: V2Anything.
Le funzionalità di questa applicazione includerebbero il fornire informazioni circa la distanza percorribile dal veicolo, posizione di colonnine di ricarica, informazioni sul mercato dell'elettricità e, soprattutto, un route planner che consideri anche i trasporti pubblici e/o sistemi di condivisione di biciclette o automobili.
Il contributo proposto consiste fondamentalmente in una piattaforma che raccolga i dati necessari da diversi servizi e li integri per fornire le funzionalità sopracitate, eventualmente raccogliendo informazioni sui profili di guida degli utenti che ne fanno utilizzo.
Questa applicazione non è ancora stata implementata, ma le funzioni descritte e i problemi che si propone di risolvere sono simili a quelli del servizio realizzato nell'ambito di

questo progetto e descritto in questo documento.
A quanto risulta dalle informazioni possedute, nessun lavoro precedente fornisce quello che vuole essere il maggior contributo nella realizzazione del servizio descritto da questo documento: l'integrazione di route planning con la previsione del consumo e delle necessità di ricarica del veicolo, mediante l'interfaccia con servizi di localizzazione degli EVSE.

Parte II

Servizio

Capitolo 3

Progettazione

In questo capitolo sarà fornita una descrizione ad alto livello di astrazione del servizio di route planning realizzato, rimandando la discussione di dettagli implementativi e problematiche tecniche al capitolo 4.
Dopo una breve panoramica dell'architettura del servizio, saranno descritti i concetti e gli algoritmi che sono alla base delle operazioni svolte dai suoi componenti, ponendo enfasi su alcune scelte effettuate.

3.1 Architettura

L'architettura del servizio di route planning, rappresentata nel diagramma in figura 3.1, prevede comunicazione tra un front-end ed un back-end.
Attraverso il primo l'utente può immettere gli input per il planning del percorso (quali i punti di partenza e destinazione, il modello di veicolo guidato e la carica attuale della batteria) e visualizzare una rappresentazione grafica o testuale dei risultati ottenuti.
Il diagramma in figura 3.1 vuole evidenziare come il front-end sia un componente esterno al servizio vero e proprio, consentendo una certa flessibilità nella sua implementazione e nel definire il modo in cui elabori o visualizzi i risultati, a patto che riesca a comunicare con il back-end.

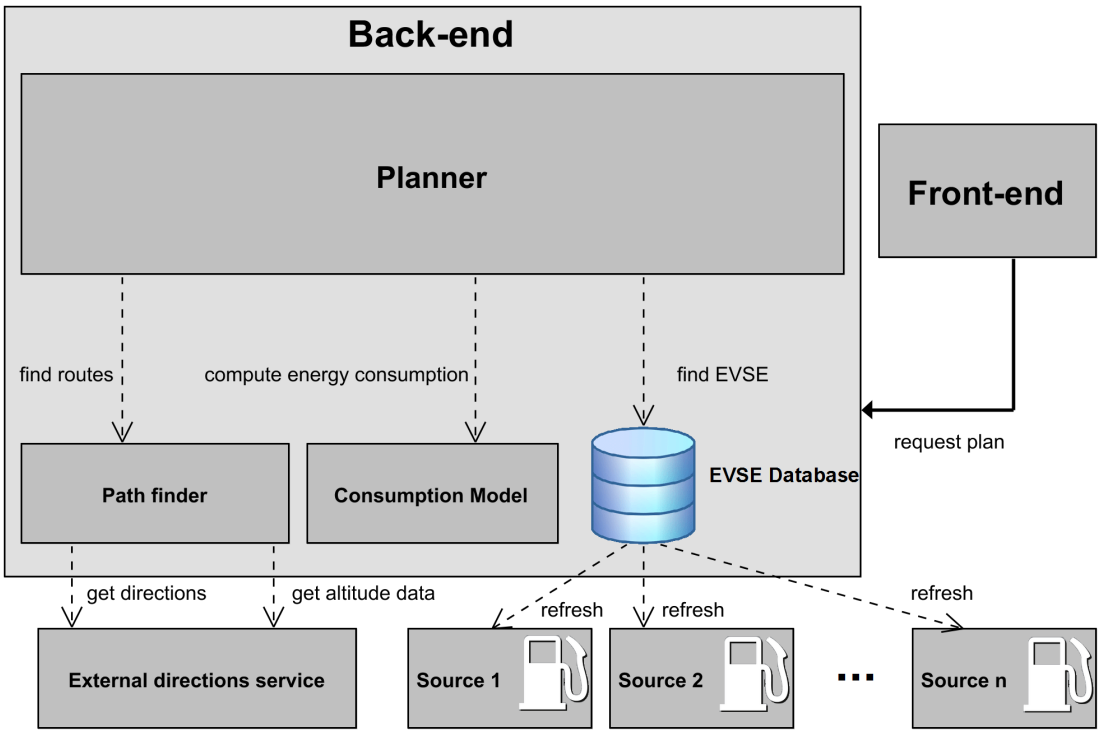

Figura 3.1: Architettura del servizio

Il back-end costituisce il planner vero e proprio e incorpora la logica del servizio, componendosi di tre elementi principali:

- Il modello di previsione del consumo energetico (*consumption model*).
- Il modulo (*path finder*) per il calcolo di percorsi stradali tra due punti e relativi dati altimetrici.
- Il *planner* che sfrutta le informazioni fornite dall'utente e le funzionalità degli altri moduli, nonché un database di EVSE, per determinare il percorso ottimale tra partenza e destinazione, comprensivo di eventuali soste per ricarica.

Le funzionalità e gli algoritmi implementati da questi tre componenti saranno descritti in dettaglio nelle sezioni 3.2 e 3.3.

Il *consumption model* fornisce una stima del consumo energetico (kWh) derivato dal percorrere un tratto di strada con un veicolo elettrico, date le sue caratteristiche (massa, area della massima sezione trasversale, ...).
Il *path-finder* interroga un servizio esterno per reperire informazioni sui percorsi stradali tra due punti arbitrari e determinare l'altitudine dei punti attraversati, utilizzata come supporto al calcolo del consumo.
Analogamente, il database di EVSE è costruito integrando i dati reperiti da diverse sorgenti esterne: fornitori di informazioni sulla posizione delle stazioni di ricarica e relative caratteristiche tecniche (numero di connettori, potenza e tensione in uscita, corrente erogata, ...).
I dati possono essere aggiornati con frequenza variabile, a seconda della sorgente: gli EVSE installati da compagnie elettriche o comuni sono tipicamente statici, mentre le stazioni di ricarica private, come quelle offerte da privati cittadini, possono variare dinamicamente la propria disponibilità (ad esempio in base alla disponibilità di energia prodotta attraverso impianti fotovoltaici).
Il *planner* utilizza le funzioni offerte dal framework sottostante, definito dagli altri componenti, per costruire un grafo dei possibili percorsi tra partenza e destinazione e determinare quello ottimo, considerando stato di carica del veicolo, capacità della batteria e una soglia minima di carica residua nel raggiungere un punto intermedio o la destinazione. Questi ultimi due parametri possono essere considerati come un margine sicuro che il guidatore dell'EV può regolare in base alle attività programmate o al suo livello di fiducia nel servizio.

3.2 Predizione del consumo

Questa sezione descrive il modello di predizione del consumo adottato dal servizio.
Il modello definisce una funzione, qui denominata $E_{Prediction}$, che dato un veicolo elettrico EV e relative caratteristiche (massa, area della massima sezione trasversale, ...) e un percorso r (route) di cui sono note determinate proprietà (distanza totale percorsa e una stima del tempo necessario a percorrerla), fornisce una valutazione del quantitativo di energia consumata nel percorrere quest'ultimo.

3.2.1 Suddivisione del percorso

Della route r è nota anche una sequenza ordinata di passi, qui definiti *steps*, che la compongono: $[s_1, s_2, \ldots, s_n]$.
Ogni step descrive una singola e specifica istruzione sul viaggio (es. "Proseguire per un chilometro e girare a destra all'incrocio") e i punti di giuntura tra i diversi step possono essere visti come deviazioni rispetto al tratto di strada percorso precedentemente: svolte, ingresso ed uscita da tratti autostradali, ingresso ed uscita da rotatorie o altro; un singolo step corrisponde quindi ad un sotto-percorso più o meno rettilineo. Per ogni step s_i sono note le coordinate geografiche dei punti che lo delimitano (p_i e p_{i+1}), la sua lunghezza d_i (distanza percorsa da p_i a p_{i+1}) e il tempo Δt_i necessario a percorrerlo.
La route e gli step in cui è suddivisa possono essere considerati come prodotti da un oracolo esterno al servizio (effettivamente un altro servizio), ragion per cui non è possibile selezionare arbitrariamente parametri quali il numero di step e/o la loro lunghezza.
Si può solo invocare una funzione offerta da questo oracolo, definita come $route(p, q)$ e tale che dati i punti geografici p e q essa restituisca una rappresentazione della route (o un insieme di più route possibili) fra i due punti, comprensiva di suddivisione in step e relative proprietà.
C'è tuttavia la garanzia che, dati n step, la distanza totale d percorsa nella route sia uguale a $\sum_{i=1}^{n} d_i$, il tempo totale Δt necessario a percorrerla sia uguale a $\sum_{i=1}^{n} \Delta t_i$ e i punti di partenza p e destinazione q siano tali che $p = p_1$ e $q = p_{n+1}$.
Ciò che manca è la considerazione del fattore altimetrico relativo ai punti attraversati.

Per includerlo viene nuovamente interrogato l'oracolo (o servizio esterno), invocando questa volta una funzione che potrebbe essere definita come $elevation(s, \alpha)$ e tale che dato lo step s, essa lo suddivida in una lista di punti $[p_1, \ldots, p_k]$, pressoché equidistanti, con $\alpha > 1$ lunghezza dei segmenti che uniscono coppie di punti consecutivi.
Ogni punto p_j è rappresentato come una tripla $< lat_j, lng_j, ele_j >$ di coordinate geografiche (latitudine, longitudine) e altitudine.
Il parametro α influenza le dimensioni della lista di punti creata: minore è il valore di α, maggiore sarà il numero di elementi all'interno della lista (con punti consecutivi separati da una minore distanza), permettendo di definire una precisione arbitraria nella suddivisione del segmento s, a patto che il numero k di punti prodotti sia inferiore ad una certa soglia k_{max} fissata dall'oracolo.
Qualora il numero di punti prodotti, calcolabile dividendo la lunghezza dello step per α prima di invocare *elevation*, superi k_{max}, è possibile semplicemente aumentare in maniera adattiva il valore di α (riducendo però la precisione del calcolo); nel caso in cui α sia minore della lunghezza dello step, si avrà una lista composta da due soli punti, coincidenti con i punti di partenza e destinazione.
É ai segmenti così prodotti (quelli che uniscono i punti in cui è suddiviso uno step) che viene applicata l'effettiva funzione di calcolo del consumo *consumption* (descritta in dettaglio nella sezione 3.2.2), la quale utilizza formule di fisica dinamica che considerano le caratteristiche del veicolo e del moto.
Rimane da chiarire come venga calcolata la velocità. Poiché per ogni step è nota la sua lunghezza d ed il tempo Δt necessario a percorrerlo, è possibile effettuare una stima semplicistica della velocità calcolandola come $v = \frac{d}{\Delta t}$.
Tuttavia questa velocità v è relativa alla percorrenza dell'intero step, il quale potrebbe essere lungo decine di chilometri (a discrezione del servizio esterno che fornisce indicazioni sui percorsi). Al fine di non valutare un semplice moto rettilineo uniforme sull'intero step (il quale fornirebbe probabilmente un valore di consumo ottimistico, privo della considerazione di accelerazioni e decelerazioni), si rende necessaria l'approssimazione delle variazioni di questa velocità lungo il percorso.
Per ottenere un valore di velocità relativo ai singoli segmenti che dividono lo step s, per ognuno di essi questo viene calcolato come un valore casuale con distribuzione normale

$\mathcal{N}(v, \frac{v}{k})$, usando come media la velocità sull'intero step (intesa quindi come velocità media) e come scarto quadratico medio il valore prodotto dividendo la velocità sull'intero step per il numero di punti che lo dividono.
Poiché, come accennato precedentemente, ad ogni step corrisponde un sotto-percorso rettilineo e alle connessioni tra essi delle deviazioni, usando lo scarto quadratico medio così calcolato si ottiene che a segmenti di step particolarmente lunghi (ad esempio lunghi tratti autostradali, nei quali è più facile mantenere una velocità pressoché costante) corrispondano minor variazioni rispetto alla velocità media; al contrario, uno step particolarmente corto (ad esempio parte di una strada in città) vedrà variazioni di velocità più brusche nei punti che lo suddividono.
Il tempo necessario a percorrere un singolo segmento è banalmente assunto come il tempo totale per percorrere lo step fratto il numero di segmenti in cui è diviso. Poiché i segmenti hanno pressoché la medesima lunghezza (α), l'assunzione che il tempo necessario a percorrerli sia il medesimo non è troppo forte.
Una volta ottenuti i punti che dividono ognuno degli n step che compongono la route r e calcolata la velocità del veicolo in ognuno di essi, utilizzando il metodo appena descritto, la funzione $E_{Prediction}$ risulta così definita:

$$E_{Prediction}(R, EV) = \sum_{s_i \in R} \sum_{j=1}^{k_i} consumption(p_j, p_{j+1}, v_j, v_{j+1}, \frac{\Delta t_i}{k_i}, EV)$$

con k_i numero di segmenti in cui è suddiviso lo step s_i:

$$k_i = \lfloor \frac{d_i}{\alpha} \rfloor - 1$$

La funzione calcola e restituisce il consumo totale come somma dei consumi relativi ad ogni singolo segmento, ottenuti mediante applicazione della funzione *consumption*, descritta nella sezione successiva.

3.2.2 Calcolo del consumo

Il calcolo del consumo energetico è definito dalla seguente funzione:

$$consumption : Point \times Point \times Speed \times Speed \times Time \times Vehicle \rightarrow Energy$$

la quale è applicata all'attraversamento del segmento di strada da p_i a p_j (punti rappresentati in coordinate geografiche, come triple <latitudine, longitudine, altitudine>), i valori di velocità del veicolo in tali punti (rispettivamente v_i e v_j), il tempo Δt necessario a percorrere il segmento e le caratteristiche del veicolo EV.
Le caratteristiche del veicolo EV comprendono:

- La sua massa m.
- L'area della sua massima sezione trasversale A_{cross}.
- Il coefficiente di forma C_{drag}, rappresentante la "levigatezza" della sua superficie frontale.
- Il coefficiente di attrito μ_a delle gomme su asfalto.
- L'efficienza $engine_{eff}$ del motore (nel trasformare energia elettrica in cinetica).
- Le caratteristiche della batteria del veicolo.

Dati questi parametri, la funzione *consumption* restituisce il quantitativo di energia consumata e, espresso in chilowattora (kWh), nel percorrere il segmento tra i due punti.

Fattore altimetrico

Al fine di considerare la pendenza del segmento di strada che connette p_i e p_j, viene calcolata la differenza di altitudine fra i due punti e relativo angolo di pendenza.
Dati i due punti espressi in coordinate geografiche, viene innanzitutto utilizzato un metodo di proiezione cartografica per convertirle in coordinate cartesiane e determinarne la distanza in metri:

$$d_{i,j} = \sqrt{((lng_j - lng_i) * \frac{\cos(lat_i + lat_j)}{2})^2 + (lat_j - lat_i)^2} * Radius_{earth}$$

considerando latitudine e longitudine convertiti in radianti e con $Radius_{earth}$ raggio medio del globo terrestre (distanza del centro della Terra dalla sua superficie al livello medio del mare), il cui valore è di 6,371.005076123 chilometri.
Data la distanza, è possibile utilizzarla per calcolare l'angolo $\gamma_{i,j}$ (in radianti) di inclinazione del segmento che connette i due punti, considerandolo come l'ipotenusa di un triangolo rettangolo e sfruttando il teorema di Pitagora:

$$\gamma_{i,j} = \arctan(\frac{alt_j - alt_i}{d_{i,j}})$$

Energia consumata

Al fine di determinare il lavoro necessario a spostare il veicolo attraverso il segmento, sono calcolate quattro diverse forze:

- F_{ma}: forza di inerzia.
- F_{mg}: forza peso, data dalla forza di gravità in salita o discesa.
- F_{roll}: forza d'attrito, data dalla resistenza generata dall'attrito dei pneumatici sulla strada.
- F_{drag}: forza data dalla resistenza aerodinamica dell'aria.

Poiché vale la seguente equazione:

$$F_{ma} = F_{tot} - (F_{mg} + F_{roll} + F_{drag})$$

ne deriva la possibilità di calcolare la forza totale F_{tot} applicata al veicolo come somma delle quattro componenti sopracitate.

In figura 3.2 è rappresentata l'applicazione delle componenti nel moto di un veicolo su piano (strada) inclinato.
Per completezza segue la descrizione del calcolo delle quattro forze, utilizzando i parametri menzionati precedentemente (espressi mediante la medesima notazione).

Forza di inerzia:

$$F_{ma} = m \cdot a_{i,j}$$

considerando un moto uniformemente accelerato nel percorrere il segmento, l'accelerazione $a_{i,j}$ è calcolata semplicemente come

$$a_{i,j} = \frac{v_j - v_i}{\Delta t}$$

Forza peso:

$$F_{mg} = m \cdot g \cdot \sin(\gamma_{i,j})$$

Il valore di F_{mg} risulta negativo se il segmento percorso è in discesa.
Nelle formule seguenti è considerato un singolo valore di velocità $\overline{v_{i,j}}$, calcolato come media fra v_i e v_j.

Forza d'attrito:

$$F_{roll} = m \cdot g \cdot \cos(\gamma_{i,j}) \cdot \overline{v_{i,j}}$$

Resistenza dell'aria:

$$F_{drag} = \frac{1}{2} \cdot \rho_{air} \cdot C_{drag} \cdot A_{cross} \cdot \overline{v_{i,j}}^2$$

Con ρ_{air} coefficiente di densità dell'aria, pari a 1.2041.

A questo punto il lavoro necessario è calcolato come

$$W = F_{tot} \cdot \Delta t \cdot \overline{v_{i,j}} \cdot \frac{1}{engine_{eff}}$$

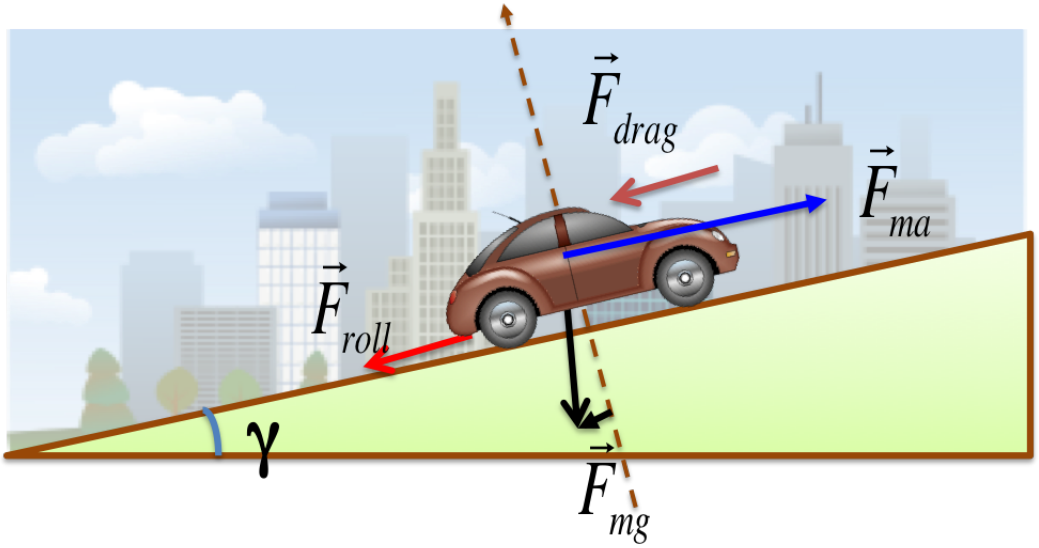

Figura 3.2: Dinamica del moto di un veicolo

da cui segue che l'energia consumata espressa in chilowattora, nonché output della funzione *consumption*, sia

$$e = \frac{W}{3,600,000}$$

Nonostante la sua semplicità, questo modello fornisce una ragionevole approssimazione del consumo medio di un veicolo elettrico, come dimostrato in lavori quali [MSAN11] e [BBDF+14].
In questi lavori viene mostrato, mediante tecniche di simulazione e confronto dei risultati relativi al consumo energetico con dati reali, come l'approssimazione fornita sia corretta. Difatti, le equazioni qui presentate fanno riferimento ad un moto traslatorio e non rotatorio; nel secondo tipo di moto, effettivamente corrispondente a quello di una ruota che gira continuamente intorno a un asse, dovrebbe essere calcolato l'attrito volvente e non radente, includendo quindi il raggio degli pneumatici del veicolo nell'equazione e considerando il momento meccanico della forza totale applicata al veicolo, eventualmente distribuita tra le quattro ruote.

Il modello prevede quindi una semplificazione del moto reale di un veicolo, ma gli studi dimostrano come questa possa essere comunque ritenuta sufficiente.

Frenata rigenerativa

Fino ad ora non è stato esplicitato cosa succede qualora l'accelerazione $a_{i,j}$ nell'attraversare il segmento sia negativa, rappresentando quindi una decelerazione/frenata.
In questo caso ne risulterà un guadagno energetico, considerando il funzionamento di un meccanismo di *Kinetic Energy Recovery System*, il quale evita la totale dispersione dell'energia cinetica in calore, durante la frenata, consentendone un parziale recupero sotto forma di energia elettrica.
In pratica, se l'accelerazione è negativa, viene calcolata l'energia cinetica prodotta nella decelerazione del veicolo, come

$$E_{kin} = \frac{1}{2} \cdot m \cdot (v_j^2 - v_i^2) \cdot \mathcal{U}([0, \eta_{regen}])$$

Il parametro $\eta_{regen} \in [0, 1]$, rappresenta l'efficienza rigenerativa: percentuale di energia elettrica recuperabile a partire da quella cinetica prodotta nella frenata.
Questo valore dipende da numerosi fattori, tra i quali l'efficienza del meccanismo di frenata e rigenerazione impiegati (distinti in seriali o paralleli) e vede un limite teorico del 30%, prodotto da recenti studi del IDSC (Institute for Dynamic Systems and Control) svizzero. Nell'ambito di questo modello l'efficienza rigenerativa viene calcolata come un valore casuale con distribuzione uniforme nell'intervallo $[0, \eta_{regen}]$.
L'energia cinetica prodotta viene infine sottratta al lavoro W precedentemente calcolato, determinando un potenziale output negativo della funzione *consumption* (rappresentante un guadagno energetico).
É stato comunque scelto un approccio "pessimistico" al calcolo del consumo, ignorando l'eventuale risultato negativo di *consumption* e portandolo a 0 (ragion per cui, percorrere un segmento in discesa e/o con una brusca decelerazione risulta semplicemente in un consumo nullo).

Ulteriori fattori

Il modello per il calcolo del consumo presentato in questa sezione tiene conto delle caratteristiche del veicolo, del percorso e del moto.
Esistono tuttavia molti altri fattori esterni da tenere in considerazione: il peso dei passeggeri o in generale del carico (da sommarsi a quello del veicolo) e le condizioni meteorologiche. Per quanto riguarda queste ultime, hanno un maggior impatto la velocità del vento e la presenza di pioggia o neve sul manto stradale, incrementando la resistenza al rotolamento. In aggiunta a queste andrebbe considerato anche l'eventuale utilizzo del condizionatore d'aria e/o dei fanali del veicolo.
Un possibile sviluppo futuro del servizio potrebbe includere la considerazione di tutti questi fattori nel calcolo della funzione *consumption*, così da fornire un valore di consumo più accurato.

3.3 Route planning

Questa sezione descrive l'algoritmo utilizzato per determinare il percorso "ottimo", in termini di consumo energetico, tra due punti s (sorgente) e d (destinazione), espressi in coordinate geografiche (eventualmente derivate a partire da un indirizzo).
Il primo passo dell'algoritmo consiste nella costruzione di un grafo orientato e pesato, i cui nodi sono:

- La sorgente s, punto di partenza del veicolo elettrico.
- La destinazione finale d.
- Un insieme di EVSE, presso i quali il veicolo può ricaricarsi.

Gli archi del grafo sono i percorsi stradali tra i nodi.
I pesi di ogni arco sono determinati come risultato della funzione $E_{Prediction}$ definita in 3.2, la quale restituisce il consumo energetico e_r (in kWh) nell'attraversamento del percorso (route) r.
In aggiunta ai due punti s e d, l'algoritmo considera quattro parametri:

1. Le caratteristiche del veicolo EV (massa, area della massima sezione trasversale, ...) utilizzate dal modello di predizione del consumo, descritto in 3.2.2. Tra queste è inclusa anche la capacità energetica (denotata in seguito con SOC_{max}) della batteria del veicolo.
2. Lo stato di carica iniziale SOC_s della batteria del veicolo nel punto s, espresso in kWh.
3. La soglia $threshold$ di carica minima che il guidatore vuole avere al raggiungimento di ogni punto intermedio del percorso.
4. La soglia $threshold_d$ di carica minima che il guidatore vuole avere al raggiungimento della destinazione d.

Dati questi parametri viene quindi calcolato il percorso ottimo per raggiungere la destinazione, applicando eventuali riduzioni al grafo di percorsi.

3.3.1 Grafo dei percorsi

Questa sezione descrive la costruzione del grafo $G = (V, E)$ di percorsi stradali fra l'origine s e la destinazione d.

Nodi

L'insieme V di nodi del grafo G contiene inizialmente l'origine s e la destinazione d.
Successivamente, vengono aggiunti i nodi relativi agli EVSE presso i quali il veicolo può ricaricarsi.
L'insieme $\mathbb{E}$ di stazioni di ricarica è costruito a partire da $\mathbb{E}_1 \cup \mathbb{E}_2 \cup \ldots \cup \mathbb{E}_n$, ottenuti interrogando n diversi servizi esterni, i quali forniscono informazioni circa la posizione (coordinate geografiche ed indirizzo) e caratteristiche tecniche (numero di connettori, corrente erogata, ...) delle stazioni. $\mathbb{E}$ è quindi costruito come unione degli insiemi ottenuti dai vari servizi, escludendo eventuali elementi replicati.
Al fine di escludere stazioni di ricarica potenzialmente irraggiungibili o troppo lontane dall'area di interesse, vengono incluse solo quelle nel sottoinsieme definito dalla seguente condizione:

$$\mathbb{E}_{final} = \{station \in \mathbb{E} \mid distance_{midpoint_{s,d},station} \leq distance_{s,d}\}$$

ossia le stazioni la cui distanza dal punto medio fra sorgente e destinazione $midpoint_{s,d}$ non superi la distanza fra sorgente e destinazione, considerando quella in linea d'aria (calcolabile semplicemente utilizzando le formule descritte in 3.2.2).
L'immagine in figura 3.3 vuole dare un'idea visuale dell'applicazione di tale filtro per determinare le stazioni di ricarica da aggiungere all'insieme dei nodi del grafo.
Sebbene questa condizione possa essere rilassata ulteriormente, aumentando il raggio del cerchio, si ritiene ragionevole assumere che il veicolo non necessiterà di deviare in tale misura dal suo percorso originale, per effettuare una ricarica (tale assunzione diventa più forte con l'aumentare di $distance_{s,d}$).

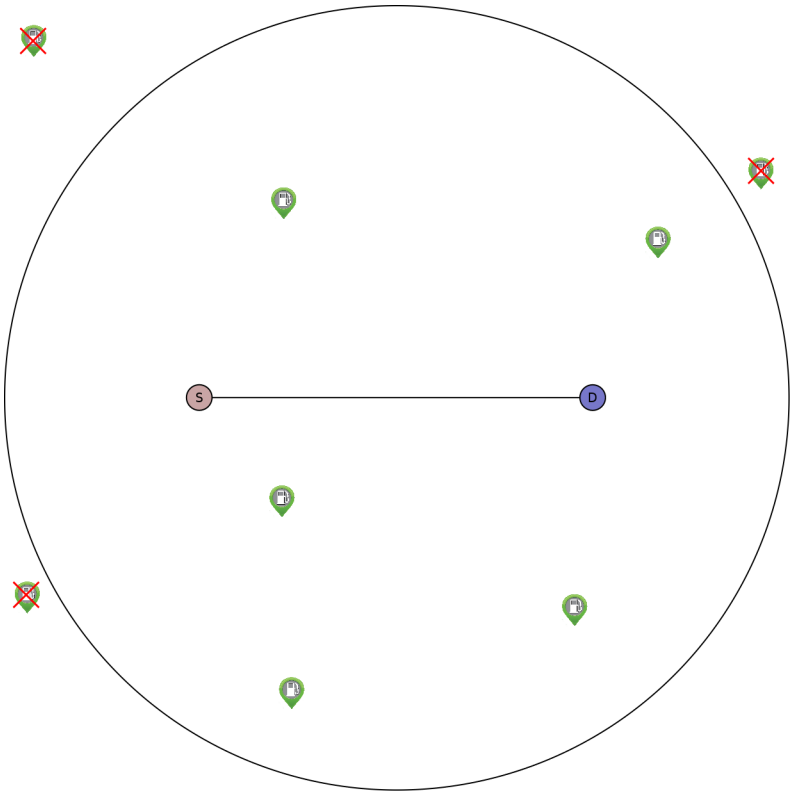

Figura 3.3: Stazioni di ricarica incluse nel grafo

Archi

Come accennato in 3.2.1, i percorsi che costituiscono gli archi del grafo sono ottenuti mediante l'invocazione di una funzione offerta da un servizio esterno, denominata *route*. Dati due punti u e v (nodi del grafo), $route(u, v)$ restituisce un insieme $R_{u,v}$ di percorsi tra i due punti.
Considerando un numero arbitrario di possibili percorsi tra u e v, la garanzia offerta dal servizio esterno è che gli n percorsi all'interno dell'insieme $R_{u,v}$ siano gli n percorsi più brevi (in termini di distanza percorsa guidando un veicolo) fra i due punti; questo è garantito dall'utilizzo di algoritmi di cammino minimo (Dijkstra) a opera del servizio esterno, sul proprio grafo di percorsi.
Poiché si è interessati a minimizzare l'energia consumata, risulta naturale definire il peso $w_{u,v}$ dell'arco che connette due punti u e v come il consumo minimo nel percorrere il

tragitto tra di essi:

$$w_{u,v} = min\{E_{Prediction}(r) \mid r \in R_{u,v}\}$$

Assumendo che ogni punto sia raggiungibile dagli altri, non è necessario costruire un grafo fortemente connesso: è sufficiente calcolare il percorso "ottimo" (con relativo peso energetico) da s a d, da s ad ogni stazione di ricarica $station \in \mathbb{E}_{final}$ e da ogni stazione a tutte le altre e a d.
Una volta costruito il grafo G, in cui il peso dell'arco tra ogni coppia di nodi rappresenta il minimo consumo energetico nel percorrere la strada che li unisce, è possibile utilizzarlo per definire il percorso tra s e d: tale procedimento è descritto in 3.3.2.

Riduzione degli archi

É possibile applicare un *pruning* (potatura) degli archi del grafo così costruito, basandosi sulla semplice considerazione che se una stazione di ricarica *station* ha una distanza $distance_{station,d}$ dalla destinazione minore di quella $distance_{station',d}$ della stazione $station'$, allora non ha senso calcolare il percorso che va da *station* a $station'$, in quanto ci si allontanerebbe da d senza beneficiare di un'ulteriore ricarica, già disponibile in *station*.

Date n stazioni di ricarica, il numero complessivo di nodi e archi rimanenti sarà dato da:

$$|V| = 2 + n \qquad\qquad |E| = 1 + 2 \cdot n + \sum_{i=1}^{n-1} i$$

con una complessità asintotica in spazio, per memorizzare gli archi, di $O(|E|^2)$.
L'algoritmo 1 riassume l'intera procedura di costruzione del grafo, comprensiva di pruning degli archi fra le stazioni di ricarica e assegnamento dei pesi.

Algorithm 1: Costruzione del grafo di percorsi

Input: $< s, d, EV >$

$V = \{s, d\}$
$E = \{(s, d)\}$

{Inizializzazione nodi}
$\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2 \cup \ldots \cup \mathbb{E}_n$
$\mathbb{E}_{final} = \{\ \}$
for all $station \in \mathbb{E}$ **do**
  if $distance_{midpoint_{s,d},station} \leq distance_{s,d}$ **then**
    $\mathbb{E}_{final} = \mathbb{E}_{final} \cup \{station\}$
  end if
end for
$V = V \cup \mathbb{E}_{final}$

{Inizializzazione archi}
for all $station \in \mathbb{E}_{final}$ **do**
  $E = E \cup \{(s, station)\}$
  $E = E \cup \{(station, d)\}$
  for all $station' \in \mathbb{E}_{final} \wedge station' \neq station$ **do**
    if $distance_{station,d} > distance_{station',d}$ **then**
      $E = E \cup \{(station, station')\}$
    end if
  end for
end for

{Calcolo pesi degli archi}
for all $(u, v) \in E$ **do**
  $R_{u,v} = route(u, v)$
  $w_{u,v} = min\{E_{Prediction}(r, EV) \mid r \in R_{u,v}\}$
end for

return (V, E)

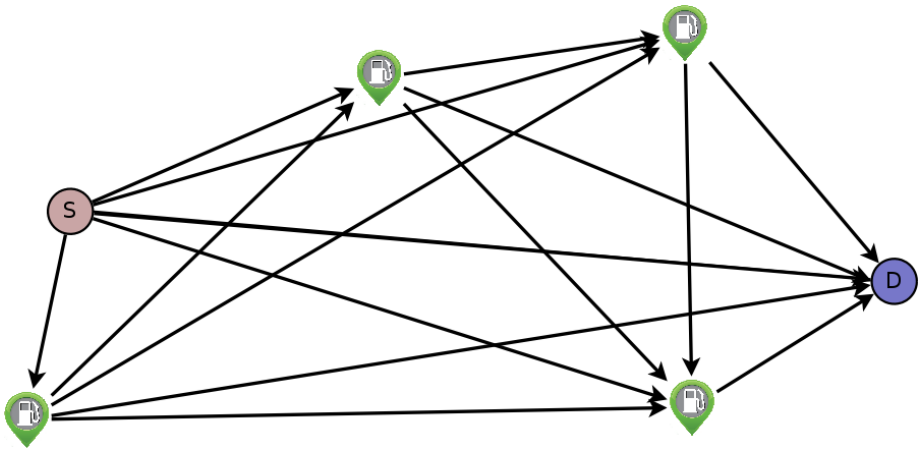

Figura 3.4: Grafo completo

3.3.2 Cammino minimo

Il grafo di percorsi costruito (vedi 3.3.1) utilizza come archi i cammini minimi (in termini di consumo energetico) tra ogni coppia di nodi; quello tra la sorgente s e destinazione d coincide semplicemente con l'arco che collega i rispettivi nodi.
Tale informazione non costituisce di per se un risultato: occorre innanzitutto determinare la *fattibilità* di tale percorso, in considerazione dello stato di carica iniziale SOC_s della batteria del veicolo.
Un percorso tra due punti u, v può essere definito come *fattibile* se il veicolo riesce ad arrivare in v senza esaurire la propria autonomia, ossia se soddisfa il seguente predicato:

$$doable(u, v) = SOC_u > w_{u,v}$$

Lo stato di carica è noto solo nel punto di partenza s.
Poiché gli altri nodi del grafo con archi uscenti sono le varie stazioni di ricarica, è invece possibile considerare la carica massima SOC_{max} della batteria del veicolo come stato di carica nei punti che rappresentano, seguendo il ragionamento che il guidatore avrebbe la possibilità di effettuare una ricarica completa in ognuno di essi.
Può inoltre risultare più utile fissare una soglia minima di carica residua nell'arrivo a

destinazione, da cui segue la forma più generale del predicato precedente:

$$doable(u, v) = \begin{cases} SOC_u - w_{u,v} > threshold_v & \text{if } u = s \\ SOC_{max} - w_{u,v} > threshold_v & \text{otherwise} \end{cases}$$

Come accennato in precedenza, la soglia di carica residua può essere distinta in quella al raggiungimento di un punto intermedio e quella che il guidatore vuole avere una volta arrivato in d ($threshold_d$).
La prima costituisce un margine di sicurezza: più alto è il suo valore, più l'esito del servizio è affidabile e le probabilità che il veicolo resti senza autonomia a causa di eventi non previsti diminuisce; in un certo senso, è inversamente proporzionale alla "fiducia" che il guidatore ripone nel servizio.
La seconda dipende invece dall'utilizzo che il guidatore vuole fare del veicolo, una volta arrivato a destinazione: riportando un veicolo in affitto presso l'autonoleggio non si è particolarmente interessati all'autonomia residua; al contrario, arrivando a lavoro si desidera un livello di carica della batteria sufficiente a ritornare a casa a fine giornata.
Date queste definizioni, si ha necessariamente che $\forall u \in V.\ threshold_u \leq threshold_d$.
A questo punto si può effettuare un *pruning* degli archi del grafo di percorsi, eliminando ogni arco (u, v) per cui valga $\neg doable(u, v)$, rappresentante un percorso che il veicolo non riuscirebbe a completare con la carica in u.
Sul grafo così ottenuto è quindi applicabile un qualunque algoritmo per trovare il cammino minimo e fattibile da s a d, considerando i pesi dei rimanenti archi.
Poiché gli archi non hanno pesi negativi (in quanto si è scelto di porre a 0 l'eventuale guadagno da frenata rigenerativa, come descritto in 3.2.2), è possibile utilizzare l'algoritmo di Dijkstra, la cui complessità in tempo è $O(|E| + |V| \cdot \log |V|)$.
Volendo considerare il guadagno energetico, sarebbe sufficiente utilizzare un algoritmo che preveda pesi negativi (ad esempio quello di Bellman-Ford, con complessità in tempo pari a $O(|E| \cdot |V|)$).

Il cammino minimo così ottenuto consiste nel risultato r_{final} della concatenazione degli archi che vanno da s a d, ossia in una serie ordinata di *route* $[r_1, \ldots, r_n]$, con $n \geq 1$,

Algorithm 2: Calcolo del cammino minimo

Input: $< s, d, (V, E), SOC_s, SOC_{max}, threshold, threshold_d >$

{Pruning dei percorsi impraticabili}
for all $(u, v) \in E$ **do**
  if $\neg doable(u, v)$ **then**
    $prune(u, v)$
  end if
end for

$[r_1, \ldots, r_n] = ShortestPath\_Dijkstra((V, E), s, d)$
$e_{total} = \sum_{i=1}^{n} e_{r_i}$
$e_{recharge} = |min(0, SOC_s - e_{total} - threshold_d)|$

{Aggiunta degli step di ricarica}
for $i = 2;\ i < n;\ i++$ **do**
  $r_{final} = join(r_{final}, r_i)$
  $appendStep(r_{final}, step_{charging}(e_{r_i} + threshold_{i+1}))$
end for

return $< r_{final}, \mathbb{E}_{final}, e_{total}, e_{recharge} >$

tali che il punto di arrivo di ogni route r_i coincide con il punto di partenza di quella successiva r_{i+1}.
A partire da questa sequenza di route è possibile calcolare il consumo energetico totale del percorso come $e_{total} = \sum_{i=1}^{n} e_{r_i}$ (o somma dei pesi di ogni arco corrispondente), così come il tempo necessario e la distanza percorsa.
La descrizione del percorso pianificato è corredata dall'insieme di stazioni di ricarica $\mathbb{E}_{final}$, comprensivo di quelle presso le quali è necessaria una sosta per ricarica.
L'unica informazione che manca è di quanto il veicolo debba ricaricarsi presso ogni eventuale punto intermedio (stazione di ricarica) del percorso, al fine di raggiungere il successivo.
Dato un punto u di questo percorso, la carica necessaria ad arrivare al successivo v corrisponde semplicemente a $w_{u,v} + threshold_v$; è quindi possibile aggiungere uno *step* (vedi 3.2.1) aggiuntivo come collegamento tra ogni coppia di route, che specifichi il

quantitativo minimo di SOC che il veicolo deve avere uscendo da quel punto (stazione). In aggiunta, è anche possibile calcolare il quantitativo totale di energia che il veicolo dovrà immagazzinare durante il percorso, come $e_{recharge} = |min(0, SOC_s - e_{total} - threshold_d)|$, posta a 0 qualora la destinazione sia raggiungibile direttamente dal punto di partenza e senza oltrepassare la soglia data.
Date tutte queste informazioni, un guidatore che utilizza il servizio può decidere arbitrariamente come gestire le ricariche, ossia quanta energia recuperare presso ogni stazione intermedia indicata, considerando il quantitativo minimo necessario all'uscita da ognuna di esse e che complessivamente dovrà ricaricarsi di $e_{recharge}$.
Un tipico caso d'uso può comprendere il ricaricare più energia durante una pausa pranzo o la sosta in un motel, assicurandosi la carica necessaria all'uscita dalla stazione successiva. L'algoritmo 2 riassume queste ultime fasi del procedimento.
In aggiunta, qualora si volesse calcolare il cammino minimo in termini di tempo o distanza e non consumo energetico, risulterebbe sufficiente applicare l'algoritmo di Dijkstra considerando come pesi degli archi la distanza o il tempo necessari ad attraversare i relativi percorsi, una volta effettuato il pruning di quelli impraticabili da un punto di vista energetico.

Capitolo 4

Implementazione

In questo capitolo sarà descritta l'implementazione del servizio, le cui basi concettuali ed algoritmi sono stati presentati nel capitolo 3.

4.1 Strumenti utilizzati

Il servizio di *route planning* presentato in questo documento è realizzato come un *web service* che espone un'interfaccia RESTful.
L'architettura REST (**Re**presentational **s**tate **t**ransfer) prevede un dialogo *stateless* tra client e server, mediante concise richieste e risposte HTTP, trasferendo oggetti XML o JSON e senza utilizzare un formato di messaggi complesso, a differenza della diffusa alternativa SOAP (Simple Object Access Protocol).
La richiesta dei servizi del server da parte del client avviene principalmente specificando tutte le informazioni necessarie (metodi e parametri) all'interno del componente *path* dell'URL utilizzato per invocarlo, fornendo una modalità di comunicazione semplice e dal minimo overhead protocollare.
Per realizzare l'applicativo che fornisce il servizio è stato utilizzata la piattaforma *Node.js* [nod].
Creato da Ryan Dahl nel 2009 a partire dal motore Javascript di Google Chrome (*V8)*,

Node.js fornisce una piattaforma di sviluppo per software server-side (e non) prestante a scalabile, perfetta per applicazioni costrette a frequenti operazioni di input/output.
Infatti, il punto di forza di Node.js è il modello asincrono derivato dalla libreria cross-platform *libuv*: ogni operazione di input/output, quale una lettura di dati da database o richiesta HTTP verso altre macchine, è implicitamente svolta in maniera non bloccante per il processo principale, utilizzando un paradigma *event-driven* basato su esecuzione di funzioni di callback.
Tale modello permette di migliorare le prestazioni di applicativi che fanno input/output intensivo, eliminando la complessità aggiunta dalla gestione di operazioni concorrenti su più thread distinti.
Un altro vantaggio di Node.js è dato dall'utilizzo del linguaggio Javascript e conseguente gestione nativa e semplificata di oggetti *JSON* (**J**ava**S**cript **O**bject **N**otation), formato basato su struttura chiave-valore e particolarmente adatto per l'utilizzo da parte di un web service RESTful, per via della sua sintassi concisa e di semplice parsing.
Come supporto all'installazione di librerie di terze parti, Node.js offre un gestore dei pacchetti ufficiale: *npm*. Oltre al mantenimento di un repository di librerie e applicazioni, npm risolve automaticamente le dipendenze fra pacchetti e consente di installarli attraverso un'interfaccia a riga di comando.
Nello sviluppo del servizio qui presentato sono state utilizzate diverse librerie di terze parti, tutte registrate presso il repository di npm:

- restify: libreria per creare un server HTTP minimale e realizzare un'associazione tra il formato del path dell'URL utilizzato per invocarlo e diversi metodi.
- request: libreria per effettuare semplici richieste HTTP.
- node-googlemaps: libreria che semplifica le richieste ai web service di Google Maps.
- optimist: libreria per facilitare la lettura di parametri da riga di comando.
- domain: libreria per gestire diverse operazioni di I/O come un singolo gruppo, utilizzata per catturare le eccezioni sollevate da metodi asincroni.
- randgen: libreria per generazione di numeri casuali con distribuzioni di probabilità diverse da quella uniforme, unica implementata nativamente in Node.js.

- kdtree: libreria per costruzione e manipolazione di alberi k-dimensionali.
- mongoose: driver per interazione con MongoDB.

Per la persistenza dei dati (relativi a stazioni di ricarica e caratteristiche dei veicoli elettrici) è stato utilizzato il Database Management System open-source *MongoDB* [mon].
MongoDB è un database *document-oriented*: non utilizza un modello relazionale (NoSQL), ma rappresenta i dati con un formato binario derivato da JSON (*BSON*).
Le ragioni per l'utilizzo di MongDB sono la semplificata conversione nel formato JSON utilizzato dall'applicativo Node.js e la sua leggerezza.
Sebbene perda di espressività rispetto ai classici modelli relazionali (non esiste un'operazione di JOIN nativa che permetta di incrociare i dati di diversi insiemi di documenti) è particolarmente adatto a gestire dati non particolarmente complessi e potenzialmente reperiti con alta frequenza.

4.2 Struttura e funzioni

La struttura dell'applicazione è stata progettata seguendo un modello a classi, sfruttando le caratteristiche object-oriented del linguaggio Javascript combinate a quelle di Node.js, in particolar modo il supporto all'ereditarietà e l'incapsulamento implicito di attributi e metodi di ogni modulo. La struttura è mostrata nel diagramma UML di dominio in figura 4.1.

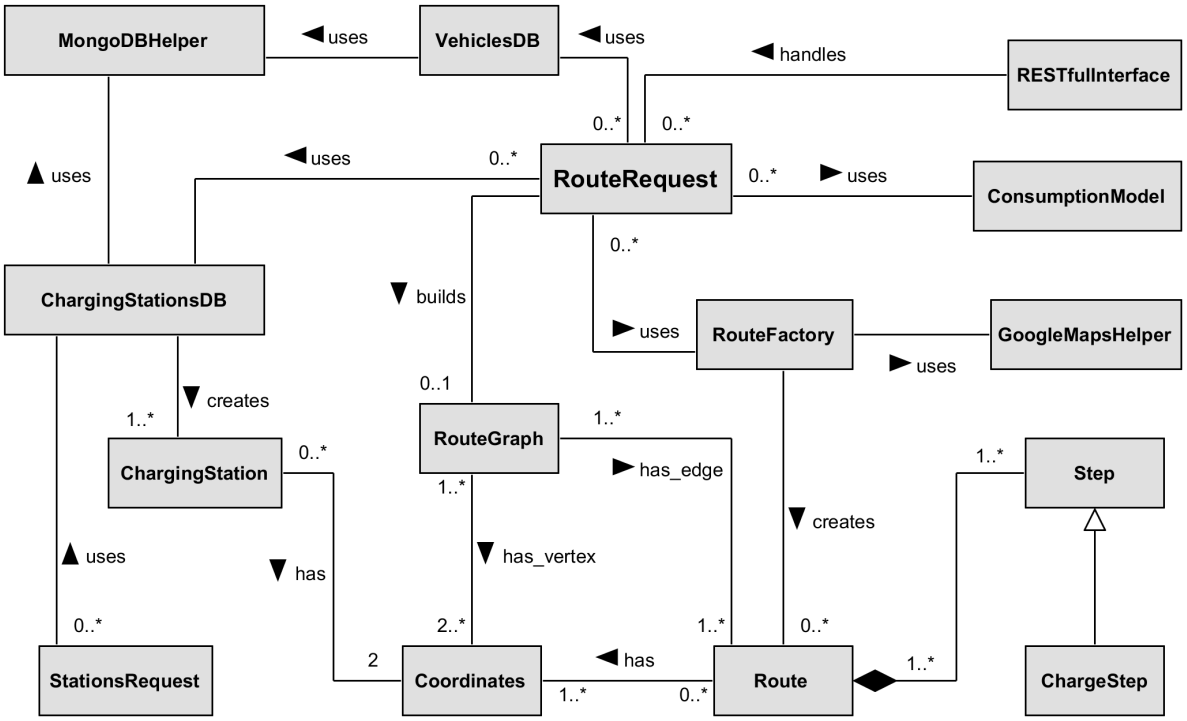

Figura 4.1: Struttura del software che realizza il servizio di route planning

Segue una descrizione dei componenti principali e relative funzioni, graficamente supportata da diagrammi UML di classe che evidenziano attributi e metodi principali.

4.2.1 RESTfulInterface

Il *singleton* RESTfulInterface rappresenta, come suggerisce il nome, l'interfaccia esposta ai client che fanno uso del web service.
Questo modulo costituisce il "main" e punto di ingresso dell'applicazione che realizza il servizio, accettando come parametro la porta TCP sulla quale il server è posto in ascolto di connessioni.
Il server è realizzato utilizzando il modulo Node.js di terze parti *restify*, particolarmente adatto a creare un server HTTP minimale e realizzare un'appropriata associazione tra l'interfaccia dei metodi esposti da quest'ultimo e l'implementazione dei relativi *handler*.
L'interfaccia di ogni metodo specifica:

- Il verbo HTTP utilizzato nella relativa richiesta (GET, POST, ...).
- Il formato del path dell'URL con cui il metodo del web service è invocato, comprensivo degli eventuali parametri (preceduti da ":" nella notazione che segue).

I metodi definiti dal web service sono i seguenti.

```
GET /route/:origin/:destination/:vehicle/:SOC/:threshold/:thresholdArrival/
:limit/:criterion
```

Il metodo *route* rappresenta il cuore del planner: la richiesta di un percorso tra due punti, calcolato tenendo conto del consumo, esigenza di carica e presenza di EVSE.
I parametri di questa richiesta sono il punto di partenza (*origin*) e destinazione (*destination*), il nome del modello del veicolo elettrico guidato (*vehicle*, attraverso il quale sarà possibile reperirne le caratteristiche), lo stato di carica attuale della batteria del veicolo (*SOC*), la carica minima che si vuole avere nel raggiungere ogni punto intermedio del percorso (*threshold*) o la destinazione (*thresholdArrival*).
Questi ultimi tre parametri sono accettati in valori percentuali e successivamente convertiti in kWh, usando come riferimento la capacità massima della batteria del veicolo.

Il *limit* pone un limite superiore al numero di stazioni di ricarica considerate nel planning (prediligendo quelle più vicine al centro della linea d'aria tra punto di partenza e destinazione, come descritto in 3.3.1); il valore "Infinity", effettivamente accettato come un numero intero in Javascript, rappresenta l'assenza di tale limite.
Il parametro *criterion* rappresenta invece il criterio con cui scegliere il percorso ottimo, ossia il parametro da minimizzare: il valore predefinito è *consumption* (consumo energetico), ma sono ammessi anche *distance* e *time*, corrispondenti alla distanza percorsa e al tempo necessario a percorrerla.

```
GET /stations/:origin/:distance/:limit
```

Il metodo *stations* consente di reperire un insieme di EVSE, senza alcun planning di percorsi, all'interno di un raggio di *distance* metri attorno al punto *origin* (espresso come indirizzo o in coordinate geografiche).
Il parametro *limit* permette di specificare un limite superiore al numero di stazioni restituite nella risposta, eventualmente assente (valore Infinity).

```
GET /vehicles
```

La risposta al metodo *vehicles* consiste in una semplice sequenza di nomi di modelli di veicoli elettrici, per i quali il planner conserva informazioni utili al calcolo del consumo (caratteristiche fisiche del veicolo e della batteria predefinita), utilizzabili nella richiesta *route*.

```
GET /schema
```

Il metodo *schema* restituisce un *JSON schema* che descrive il formato degli oggetti JSON inviati in risposta alle richieste precedenti. Lo schema comprende una descrizione della struttura dell'oggetto, nomi delle chiavi ed indicazione di quali siano opzionali, tipo dei valori ed eventuali restrizioni su di essi.

```
GET /planless/:origin/:destination/:vehicle/:SOC/:threshold/:thresholdArrival
```

Il metodo *planless*, aggiunto principalmente a scopo di validazione del servizio, consente di simulare il percorso intrapreso da un guidatore che non faccia uso del planner.
Il significato dei parametri è il medesimo di quelli del metodo *route*.
Per maggiori dettagli sulle operazioni svolte da questo metodo ed il suo utilizzo si rimanda alla sezione 5.2.

```
GET /
```

L'invocazione senza un path specifico risulta in una risposta contenente la pagina HTML che rappresenta il front-end predefinito dell'applicazione, comprensiva di codice Javascript client-side per le richieste AJAX (Asynchronous JavaScript and XML) al web service. Utilizzando la funzione *serveStatic* del modulo restify viene risolto automaticamente qualunque collegamento presente all'interno della pagina restituita (sorgenti Javascript, fogli di stile CSS, immagini, ...).

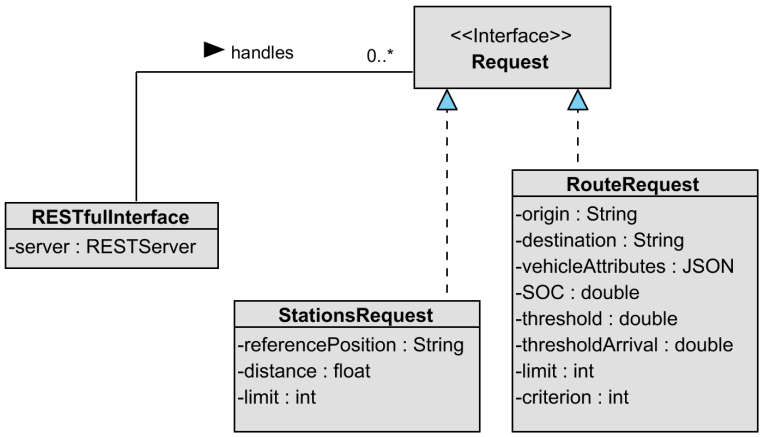

Figura 4.2: Diagramma di classi: Request

4.2.2 Request

Nel diagramma UML in figura 4.2 viene descritta la modellazione di diverse tipologie di richieste (corrispondenti ad alcune di quelle elencate precedentemente) come classi che realizzano la medesima interfaccia: *Request*. Nonostante l'utilizzo di un artificio (Javascript e Node.js non implementano un meccanismo nativo di interfacce), il diagramma descrive correttamente come le diverse tipologie di richieste siano gestite: istanziando un oggetto del giusto tipo in base al path dell'URL della richiesta.
Le responsabilità di ognuna di queste classi sono quelle di verificare la correttezza dei parametri forniti con la relativa richiesta, combinare le operazioni di altri componenti per produrre un risultato e costruire una risposta o un messaggio di errore da restituire al client.
La classe *RouteRequest* incapsula la gestione di richieste di tipo */route* o */planless*, implementando al suo interno gli handler associati ai relativi metodi esposti dall'interfaccia. Il controllo sui parametri comprende la conversione di SOC, threshold e thresholdArrival in unità di misura assolute (kWh) e l'eventuale assegnamento di valori di default (consumo energetico come parametro da minimizzare nel planning, soglie di carica residua pari all'1% della capacità). La conversione dai termini percentuali avviene dopo aver reperito informazioni circa la capacità massima della batteria nel veicolo. I punti di partenza e destinazione sono accettati come stringhe, indipendentemente dal fatto che esprimano

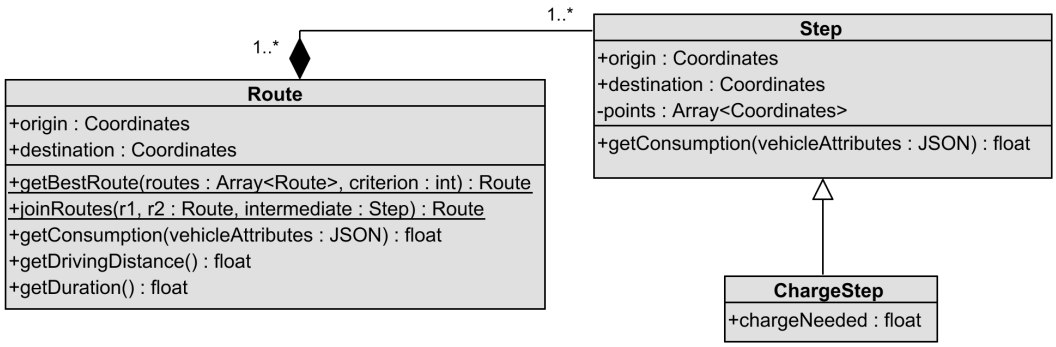

Figura 4.3: Diagramma di classi: Route

indirizzi o coordinate geografiche.
Una volta accettati i parametri, vengono utilizzati i metodi delle classi descritte in seguito per costruire il grafo di percorsi (vedi 3.3.1) e determinare quello ottimo fino a destinazione.
I metodi relativi al calcolo di un percorso "planless" svolgono le operazioni descritte nella sezione 5.2.
La variante *StationsRequest* è dedicata alla gestione delle richieste di tipo */stations*.
Il parametro che esprime il punto di riferimento viene convertito in coordinate geografiche, qualora sia stato fornito come indirizzo, sfruttando i metodi offerti da altri moduli.
Le altre classi (*VehiclesRequest* e *SchemaRequest*), non mostrate nel diagramma, svolgono operazioni analoghe per soddisfare le altre richieste.

4.2.3 Route

La classe *Route* rappresenta un percorso stradale tra due punti, espressi con coordinate geografiche e indirizzi corrispondenti.
Le coordinate di un punto sono rappresentate da un'istanza della classe *Coordinates*, la quale include proprietà come latitudine, longitudine ed altitudine, oltre ad una serie di metodi che permettono di calcolare la distanza in metri fra due punti, determinare che

due punti "coincidano" (la loro distanza non superi una certa soglia) e convertire un punto in coordinate cartesiane.
Una Route è suddivisa in un array ordinato di *Step*, composti a loro volta da una successione di punti, così come descritto in 3.2.1.
La classe Route comprende metodi di istanza che forniscono i valori di consumo energetico totale, distanza percorsa e tempo necessario stimato a percorrerla. Il consumo totale viene calcolato iterando sull'array di Step nella Route e sommando il consumo relativo ad ognuno di essi.
Per ogni Step viene calcolato il consumo nel percorrere ognuno dei segmenti che uniscono i punti che lo suddividono (metodo *getConsumption*()), applicando il modello di consumo energetico descritto in 3.2.2.
La classe Route implementa anche due metodi statici:

- *getBestRoute*(*routes, criterion*): dato un array di oggetti di tipo Route e un intero che rappresenta un criterio (tra quelli discussi in precedenza, anch'essi espressi come parametri statici della classe), restituisce la Route "migliore".

- *joinRoutes*(*route1, route2, intermediateStep*): unisce due oggetti di tipo Route, concatenando (in termini di percorso stradale) il primo al secondo e impostando di conseguenza gli attributi del risultato prodotto. La concatenazione avviene solo se la destinazione della prima Route coincide con la partenza della prima. É anche possibile fornire un terzo parametro al metodo, rappresentante uno Step intermedio da interporre tra le due Route.

Il secondo metodo viene impiegato nella costruzione del percorso finale dal grafo, aggiungendo eventuali Step di ricarica presso le stazioni.
Gli step di ricarica sono definiti dalla classe *ChargeStep*, la quale espone la stessa interfaccia di Step, fornendo però informazioni relative alla carica necessaria (in termini assoluti e percentuali) al veicolo in quel punto, al fine di riuscire a completare il percorso.

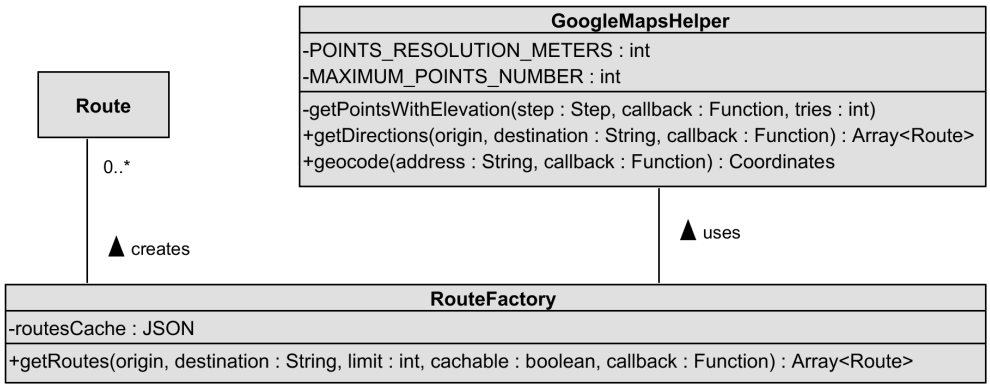

Figura 4.4: Diagramma di classi: RouteFactory

4.2.4 RouteFactory

La *RouteFactory* è responsabile della creazione di oggetti di tipo Route e inizializzazione dei relativi array di Step, suddivisi a loro volta in punti.
Per costruire le Route, la Factory si appoggia ad un altro modulo: *GoogleMapsHelper*, il quale ottiene da Google le indicazioni stradali tra due punti e i dati altimetrici del percorso. Maggiori dettagli su come avviene questa interazione e le relative problematiche saranno forniti in 4.3.1.
Al fine di ridurre le richieste a Google, la Factory incorpora al suo interno una cache di percorsi; solo i percorsi (statici) tra due stazioni di ricarica sono aggiunti alla cache (non avrebbe senso memorizzare quelli tra due punti arbitrari).
Ad ogni Route nella cache viene associato un timeout (sfruttando la funzione Javascript nativa *setTimeout*) della durata di 24 ore, oltre le quali questa è rimossa, in accordo ai termini del servizio delle Maps API di Google.

ConsumptionModel

Il modulo *ConsumptionModel* consiste in una raccolta di variabili e funzioni necessarie al calcolo del consumo energetico su un segmento stradale e produzione di valori casuali di velocità; contiene l'implementazione delle funzioni descritte in 3.2.2.

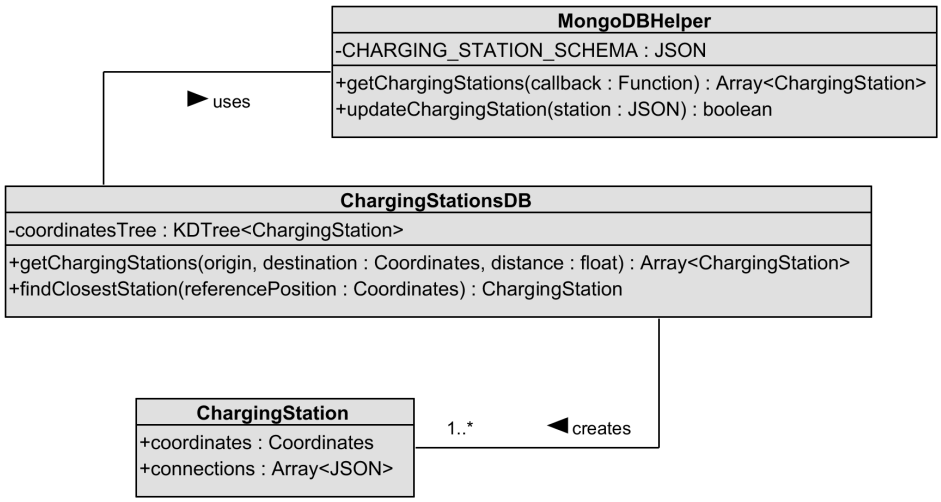

Figura 4.5: Diagramma di classi: ChargingStationsDB

4.2.5 ChargingStationsDB

Il *ChargingStationsDB* è responsabile della creazione di oggetti di tipo *ChargingStation*, i quali consistono in semplici strutture che rappresentano stazioni di ricarica (comprensive di coordinate geografiche, denominazione e caratteristiche fisiche dei connettori).
I metodi che questo modulo espone all'esterno sono *getChargingStations*(), per reperire le stazioni di ricarica lungo il percorso tra due punti (come descritto in 3.3.1) e *findClosestStation*(), per trovare la stazione più vicina ad un dato punto.
Il database interno di EVSE è caricato in memoria da MongoDB all'avvio del servizio e inserito all'interno di un *3-dimensional tree*.
Questa struttura dati indicizza gli elementi al suo interno con un valore di coordinate (tridimensionali) ed è particolarmente adatto per estrarre elementi in base alla loro vicinanza ad un dato punto, richiedendo una complessità in tempo di $O(logn)$ nel caso medio. Gli indici degli EVSE all'interno dell'albero sono le loro coordinate geografiche convertite in coordinate cartesiane attraverso un metodo di proiezione cartografica.
Il modulo *MongoDBHelper* implementa le operazioni *CRUD* (Create, Read, Update, Delete) sul database delle stazioni di ricarica in MongoDB, definendo il formato dei documenti che rappresentano le stazioni con uno schema JSON; come identificativo (campo *_id*) per le stazioni sono usate le loro coordinate geografiche.

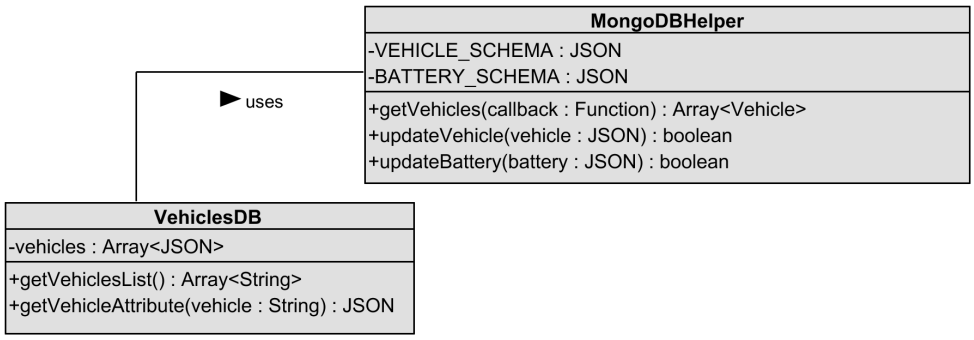

Figura 4.6: Diagramma di classi: VehiclesDB

Il database interno al servizio viene integrato con i dati su EVSE ottenuti da diverse sorgenti esterne (vedi 4.3.2).

4.2.6 VehiclesDB

Il modulo *VehiclesDB* contiene un array associativo tra i nomi dei modelli di veicoli elettrici e le relative caratteristiche fisiche e della batteria.
L'array è il risultato del caricamento in memoria dei dati conservati su MongoDB, ottenuti attraverso i metodi del wrapper MongoDBHelper.
Sono definiti due schemi distinti per veicoli e batterie, sebbene per il momento sia semplicisticamente assunta una relazione uno-a-uno tra di essi: all'interno del documento che descrive un veicolo, nel database MongoDB, vi è un riferimento all'identificativo della batteria.

4.2.7 RouteGraph

La classe *RouteGraph* costituisce il grafo dei percorsi descritto in 3.3.1.
Gli archi del grafo sono oggetti di tipo Route e i nodi sono JSON composti da:

- Coordinate geografiche del punto.

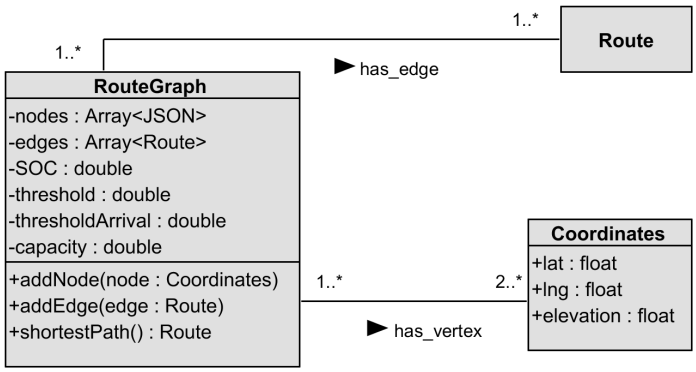

Figura 4.7: Diagramma di classi: RouteGraph

- Costo totale del percorso dal nodo di partenza a questo; la proprietà valutata dipende dal criterio passato come parametro alla richiesta e può essere il consumo, il tempo o la distanza.

- Nodo predecessore (nel cammino dal nodo di partenza).

- Riferimento alla Route che unisce il punto rappresentato dal nodo predecessore a questo.

Nodi ed archi del grafo vengono aggiunti da un oggetto RouteRequest man mano che le informazioni (indicazioni stradali e posizione delle stazioni di ricarica) diventano disponibili.
Una volta che il grafo è completato, viene invocato il metodo *shortestPath*(), il quale applica l'algoritmo di Dijkstra per restituire il percorso a costo minimo (consumo, tempo, distanza) tra partenza e destinazione, o *null*, se questo non esiste.
Il percorso viene costruito applicando ripetutamente il metodo static *joinRoutes*() della classe Route a quelle che collegano partenza e destinazione, partendo da quest'ultima e sfruttando il campo predecessore associato ad ogni nodo.
Come giuntura tra ogni coppia di Route che hanno come punto intermedio una stazione di ricarica viene aggiunto un ChargeStep, il cui valore di carica richiesta è prodotto sommando il consumo della seconda Route alla threshold di carica per l'arrivo nella destinazione di questa.

Tutte le considerazioni relative alla *fattibilità* dei singoli percorsi (descritte in 3.3.2) vengono applicate all'interno del metodo per l'aggiunta di nuovi archi, evitando di includere quelli impraticabili a causa dei vincoli energetici.

4.2.8 Front-end

A ricezione di una richiesta GET con path vuoto, il web service restituisce una pagina HTML, comprensiva di script Javascript e fogli di stile CSS, la quale costituisce il front-end predefinito per interagire con il planner.
La schermata principale del front-end si compone di un semplice form per la formulazione delle richieste e di un *canvas* contenente una mappa di Google. Attraverso il form è possibile specificare parametri quali il punto di partenza, la destinazione, il veicolo guidato, lo stato di carica iniziale (in termini percentuali), la soglia di carica residua desiderata nel raggiungere ogni punto intermedio o la destinazione, il criterio per ottimizzare i percorsi ed un limite alle stazioni da considerare nel planning.
Il veicolo viene scelto da una lista, caricata all'avvio del front-end con i risultati di una richiesta */vehicles* al servizio.
In aggiunta a questi parametri, il front-end consente di effettuare richieste */planless* (vedi 5.2) o */stations*, permettendo per quest'ultima di specificare il raggio (in chilometri) attorno al punto scelto.
Una volta completata la richiesta, i risultati sono visualizzati sulla mappa attraverso una linea che descrive il percorso (con dei marker che rappresentano stazioni di ricarica e punti attraversati). Nel lato sinistro dello schermo sono inoltre fornite le indicazioni stradali in formato testuale (in maniera simile a quanto fa Google), aggiungendo istruzioni per le eventuali ricariche e informazioni quali il consumo energetico totale, la quantità di ricarica complessiva necessaria, la distanza percorsa e il tempo impiegato.
Nelle figure 4.8 e 4.9 sono mostrate due schermate del front end: rispettivamente il form per formulare richieste di planning e relativa finestra dei risultati.
Chiaramente, poiché il servizio espone un'interfaccia RESTful facilmente interrogabile, ogni genere di front-end esterno può comunicare con esso e fornire modalità alternative di elaborazione e visualizzazione dei risultati.

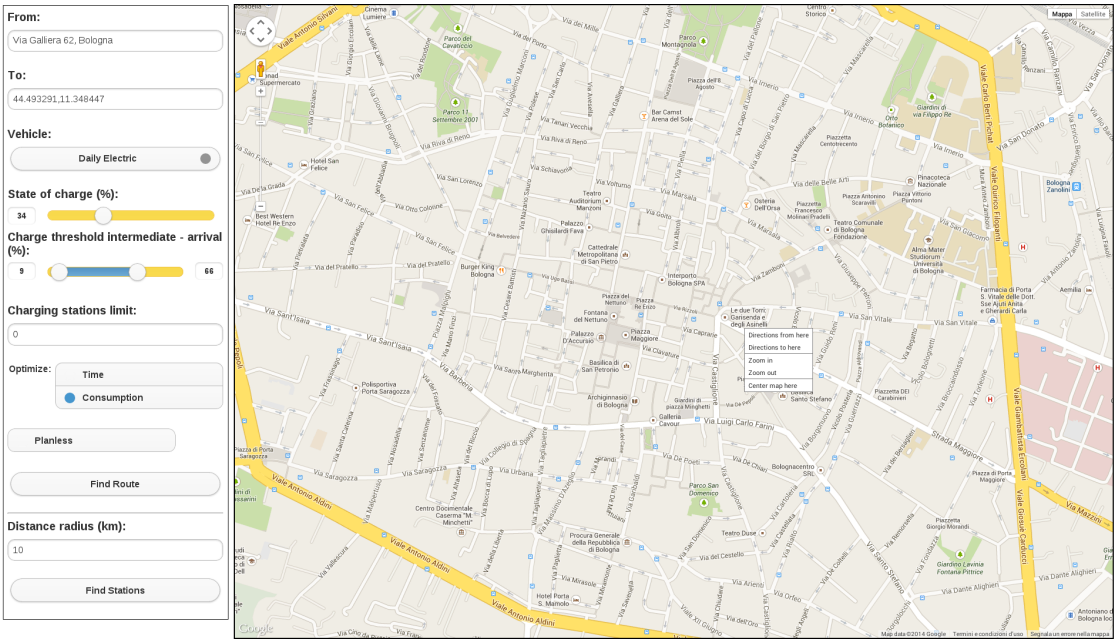

Figura 4.8: Screenshot del front-end predefinito dell'applicazione: formulazione della richieste

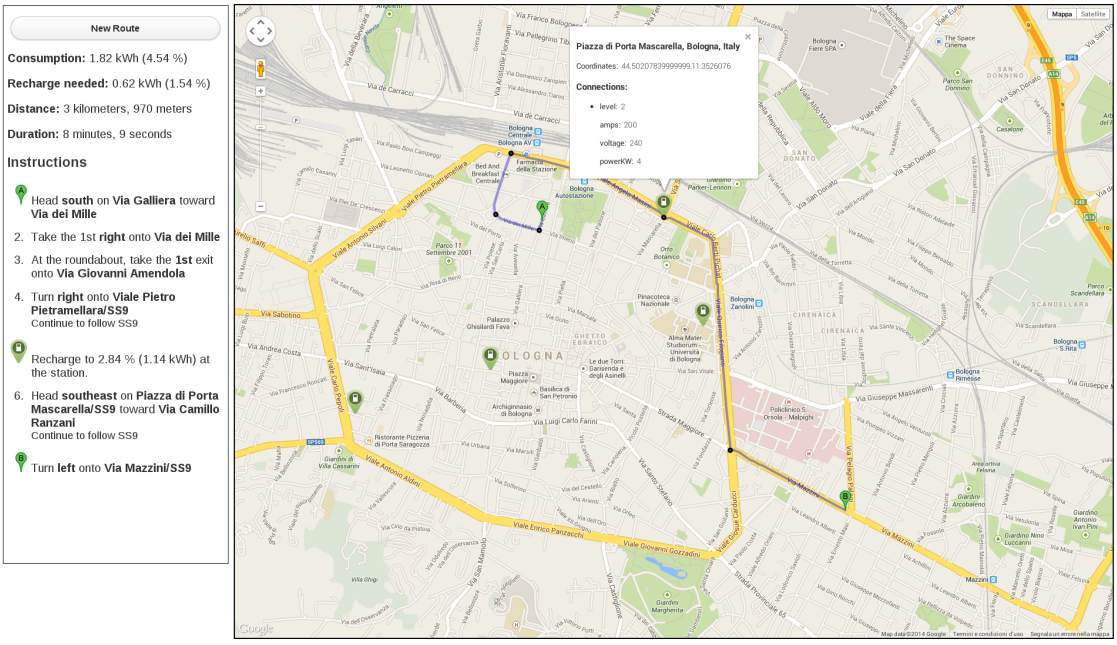

Figura 4.9: Screenshot del front-end predefinito dell'applicazione: visualizzazione del percorso

4.3 Servizi esterni

Il web service descritto in questo documento consiste in un *mashup* che usa i dati ottenuti da più sorgenti: Google Maps e i fornitori di informazioni sulla presenza di EVSE.

4.3.1 Google Maps

Al fine di ottenere indicazioni stradali, dati altimetrici e *geocoding* di indirizzi (determinare le rispettive coordinate geografiche), il servizio comunica con i *Google Maps API Web Services* [goo]: una serie di interfacce HTTP esposte da Google per fornire informazioni geografiche ad applicazioni che interagiscono con le mappe.
L'utilizzo di queste API è incapsulato all'interno del modulo *GoogleMapsHelper*, il quale a sua volta utilizza la libreria Node.js di terze parti *node-googlemaps*.
Le funzioni svolte dal modulo sono le seguenti:

- Produrre una o più Route tra due punti, espressi come coordinate geografiche o indirizzi. Sono utilizzate le *Directions API* con alcuni parametri fissati: modalità *driving*, strade a pedaggio non escluse, unità di misura nel sistema metrico decimale e parametro *alternatives* impostato a *true*. La risposta di Google è convertita in un array di Route, ognuna suddivisa in uno o più Step.

- Scomporre uno Step in una sequenza di punti. Sono utilizzate le *Elevation API* per ottenere i punti e relativi dati altimetrici, memorizzati in sequenza all'interno di un vettore, variabile d'istanza della classe Step; come parametri alla richiesta sono dati i punti che delimitano lo Step.

- Fare geocoding di indirizzi attraverso le *Geocoding API*, al fine di localizzare le stazioni di ricarica attorno ad un singolo punto (richiesta */stations*); nel caso di più possibili coordinate associate all'indirizzo, solo la prima è selezionata.

I servizi di Google restituiscono risposte in formato JSON o XML (a discrezione del richiedente), elaborate e convertite in oggetti Route, Step o Coordinate dai metodi di GoogleMapsHelper.

L'utilizzo di questi servizi presenta due limitazioni: un numero massimo di richieste per secondo e giornaliere.
Il primo caso è gestito applicando, qualora una "collisione" venga rilevata (tramite un apposito messaggio di errore in risposta alla richiesta: "OVER_QUERY_LIMIT"), una politica di *backoff esponenziale binario*, in maniera simile a quanto accade nel protocollo di accesso a canale condiviso CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).
In sostanza, se il limite di richieste per secondo è raggiunto, la prima richiesta che fallisce verrà ripetuta dopo $2^{tries} * 1000 + \mathcal{U}([0, 1000])$ millisecondi, con *tries* incrementato ad ogni successivo errore restituito per la medesima richiesta ed un'aggiunta di un numero casuale di millisecondi. Questo meccanismo è facilmente implementabile grazie alla funzione Javascript nativa *setTimeout*.
Se un numero massimo di tentativi è raggiunto, la richiesta a Google viene annullata ed al client viene restituito un errore.
Per quanto riguarda invece il limite giornaliero, l'unica soluzione è quella di minimizzare le richieste a Google, tramite politiche di caching dei percorsi ottenuti e relativi dati altimetrici (vedi sezione 4.2.4) ed evitando di inviare quelle non strettamente necessarie.
I valori di quota massima per i servizi utilizzati, diversi per l'utilizzo di API Business o gratuite, sono mostrati in tabella 4.1.

Servizio	**Free API quota**	**Business API quota**
Geocoding	2,500 richieste	100,000 richieste
Elevation	2,500 richieste, 512 punti per richiesta, 25,000 punti al giorno	100,000 richieste, 1,000,000 punti al giorno
Directions	2,500 richieste	100,000 richieste

Tabella 4.1: Quota di utilizzo dei servizi di Google Maps (Free e Business)

Il limite di punti per richiesta è eliminato nella versione Business dell'Elevation service, permettendo di ottenere maggiore precisione nel calcolo del consumo (tramite una suddivisione più fine del percorso in punti, vedi 3.2.1). Un'ulteriore limitazione, trascurabile, consiste nell'accettazione di richieste invocate tramite URL lunghi al più 2048 caratteri.

Al fine di minimizzare il numero di richieste a Google, è possibile suddividere in tre fasi consecutive la ricerca del cammino minimo fra partenza e destinazione, ognuna delle quali applicata a versioni "incrementali" del grafo di percorsi:

1. Considerando solo la partenza e la destinazione nella richiesta */route* (in questo caso non viene neanche costruito il grafo, ma ci si limita a verificare che il consumo della singola Route fra i due punti non superi il parametro thresholdArrival).

2. Aggiungendo i percorsi che vanno dalla partenza ad ogni stazione di ricarica e da queste a destinazione (in sostanza, considerando cammini con un solo hop intermedio).

3. Aggiungendo i percorsi tra le stazioni (considerando quindi i cammini ad un numero arbitrario di hop intermedi).

Nel momento in cui una delle tre fasi produce una soluzione, non sarà necessario proseguire con le altre: si ricorda infatti che i percorsi diretti tra ogni coppia di punti sono sempre quelli ottimi, in termini di tempo, distanza o consumo (come accennato in 3.3.1), da cui segue che aggiungervi una deviazione non porterebbe alcun beneficio.
La complessità, in termini del numero di richieste effettuate al servizio *Directions*, risulterebbe di $O(1)$ nel caso ottimo, $O(n)$ nel caso medio e $O(n^2)$ nel caso pessimo, con n definito come il numero di stazioni lungo il percorso tra partenza e destinazione e incluse tra i nodi del grafo di percorsi. L'algoritmo arriverebbe alla terza fase (caso pessimo) solo per percorsi molto lunghi, batterie dalla scarsa capacità e/o alti valori di threshold, risultando in una notevole riduzione del numero di richieste per gli altri due.
Per quanto riguarda invece le richieste al servizio *Elevation*, poiché ne viene effettuata una per ogni Step in cui ognuna delle Route ottenute è suddivisa, il loro numero cresce più rapidamente delle precedenti, sebbene in termini asintotici si possa far riferimento alla medesima complessità. In questo senso, una possibile manovra per ridurre il numero di richieste è quella di effettuarne una per Route, passando come parametri i punti che la delimitano; ne conseguirebbe una riduzione della precisione del profilo altimetrico, a meno che non si utilizzino le Business API, eliminando quindi il limite di punti per richiesta.

Infine, vale la pena menzionare che interrogando il servizio Directions con le Business API è possibile ottenere informazioni sul tempo necessario a percorrere una Route che siano comprensive del fattore traffico, del quale Google fornisce aggiornamenti in tempo reale.
Infatti, occorre semplicemente specificare il parametro *departure_time* nella richiesta, impostandolo al più a qualche minuto dall'istante attuale, affinché sia aggiunto il campo *duration_in_traffic* all'interno degli oggetti JSON che descrivono Route o Step, ottenuti come risposta dalle Directions API. Nel caso di percorsi molto lunghi questa informazione potrebbe comunque perdere rilevanza, richiedendo molteplici interrogazioni del planner per ottenere dati aggiornati sul traffico.
Nella sezione conclusiva sarà proposta una possibile alternativa all'utilizzo di Google Maps, finalizzata al superamento delle limitazioni menzionate.

4.3.2 Localizzazione di EVSE

Il database di EVSE mantenuto dal servizio è regolarmente aggiornato con i dati forniti da diversi servizi esterni. I dati comprendono la posizione delle stazioni di ricarica (coordinate geografiche ed indirizzo), caratteristiche tecniche (numero e tipologia dei connettori), azienda fornitrice ed eventualmente un indicazione della disponibilità.
In merito a quest'ultimo aspetto, il database costruito non mantiene informazioni circa la disponibilità di un punto di ricarica: presenza o meno di un veicolo in carica, interruzione del servizio per manutenzione o disponibilità di un EVSE privato (presenti a intermittenza e non gestiti da questo servizio).
Sebbene queste informazioni siano fornite da alcuni servizi, la loro natura dinamica e il fatto che spesso non siano completamente affidabili ha portato alla scelta di escluderle dai fattori considerati nel planning del percorso.
Va anche considerato che la maggioranza dei servizi web che forniscono queste informazioni non espone interfacce RESTful (o SOAPful), o che permettano una qualunque interazione automatizzata da parte di altri servizi (tra questi [ene], [car] e [gre]). Alcuni di quelli che lo fanno non consentono l'utilizzo delle loro interfacce a terze parti che non siano partner autorizzati (quali possono essere le case automobilistiche) o utenti paganti

(un esempio di questi servizi è [plu]).
Tra i rimanenti figura in particolare *Open Charge Map* [ope], interrogato dal modulo ChargingStationsDB al fine di integrarne le informazioni (eventualmente aggiornate) con quelle memorizzate dal database. Open Charge Map fornisce una lista di stazioni, posizionate nell'intorno di un dato punto, in formato JSON o XML, insieme alle informazioni descritte precedentemente.
Il problema di Open Charge Map è che il suo database non riflette neanche una piccola frazione del numero di EVSE in Europa, concentrando le sue attenzioni sul suolo Americano.
Per questa ragione, la maggior parte delle stazioni memorizzate nel database interno del servizio è stata reperita attraverso tecniche di *crawling* (analisi ed estrazione di dati da pagine HTML scaricate dal Web) sui portali dei maggiori fornitori di informazioni e aggiornata periodicamente.

Capitolo 5

Analisi

In questo capitolo sono presentate le valutazioni di due fondamentali proprietà del servizio: performance ed efficacia.
La prima consiste nei tempi di risposta del servizio a fronte di richieste di diversa complessità (in termini di lunghezza del percorso e/o vincoli energetici), per i quali un fattore critico è sicuramente costituito dal numero di richieste HTTP a Google.
La seconda risiede nella riduzione del rischio di non raggiungere la destinazione a causa di problemi di autonomia energetica: rappresenta quindi una validazione del servizio e sarà dimostrata mediante tecniche di simulazione.

5.1 Performance

Performance e scalabilità del servizio sono valutate misurando il tempo di risposta TR, definito come il tempo necessario ad ottenere i risultati di una richiesta di tipo */plan* (vedi sezione 4.2.1).
I dati ottenuti sono i risultati di esperimenti che prevedono richieste del planning di percorsi fra diverse coppie di punti, ottenute casualmente su una superficie di più di 400 chilometri (la regione Emilia Romagna). Per ogni coppia di punti sono utilizzati diversi valori dello stato di carica SOC iniziale della batteria: 40, 60, 80 e 100%; le caratteristiche del veicolo e la soglia di carica minima nel raggiungimento di punti intermedi e

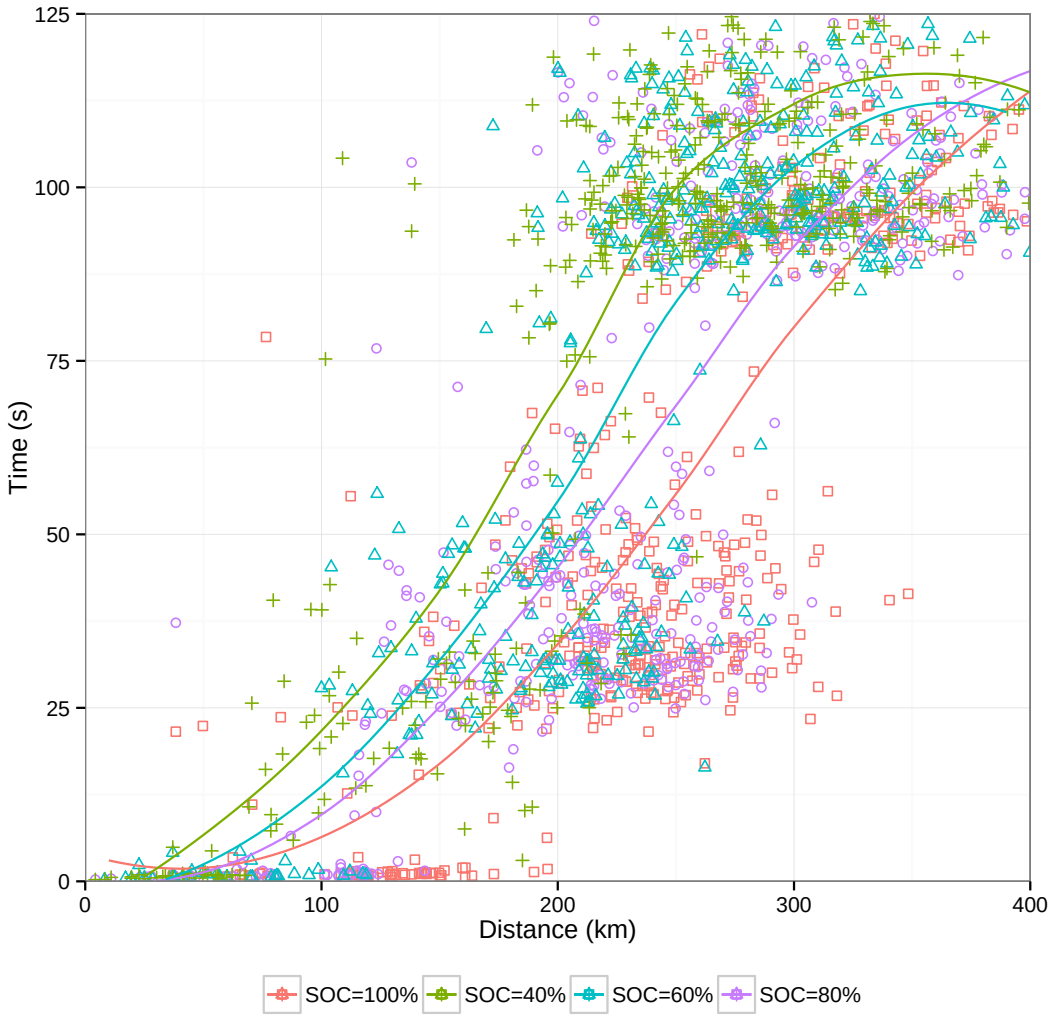

Figura 5.1: Tempi di risposta del servizio al variare della distanza percorsa

destinazione sono mantenuti costanti (modello Daily Electric e 10%). Sono stati considerati solo i risultati in cui il veicolo riesce effettivamente a raggiungere la destinazione selezionata.

Nella figura 5.1 è mostrato il valore di TR in funzione della distanza totale percorsa nella Route prodotta: ogni punto rappresenta il risultato di un esperimento e il suo colore e forma denotano il SOC iniziale; per ogni valore di SOC considerato sono anche mostrate delle curve approssimanti dei risultati.

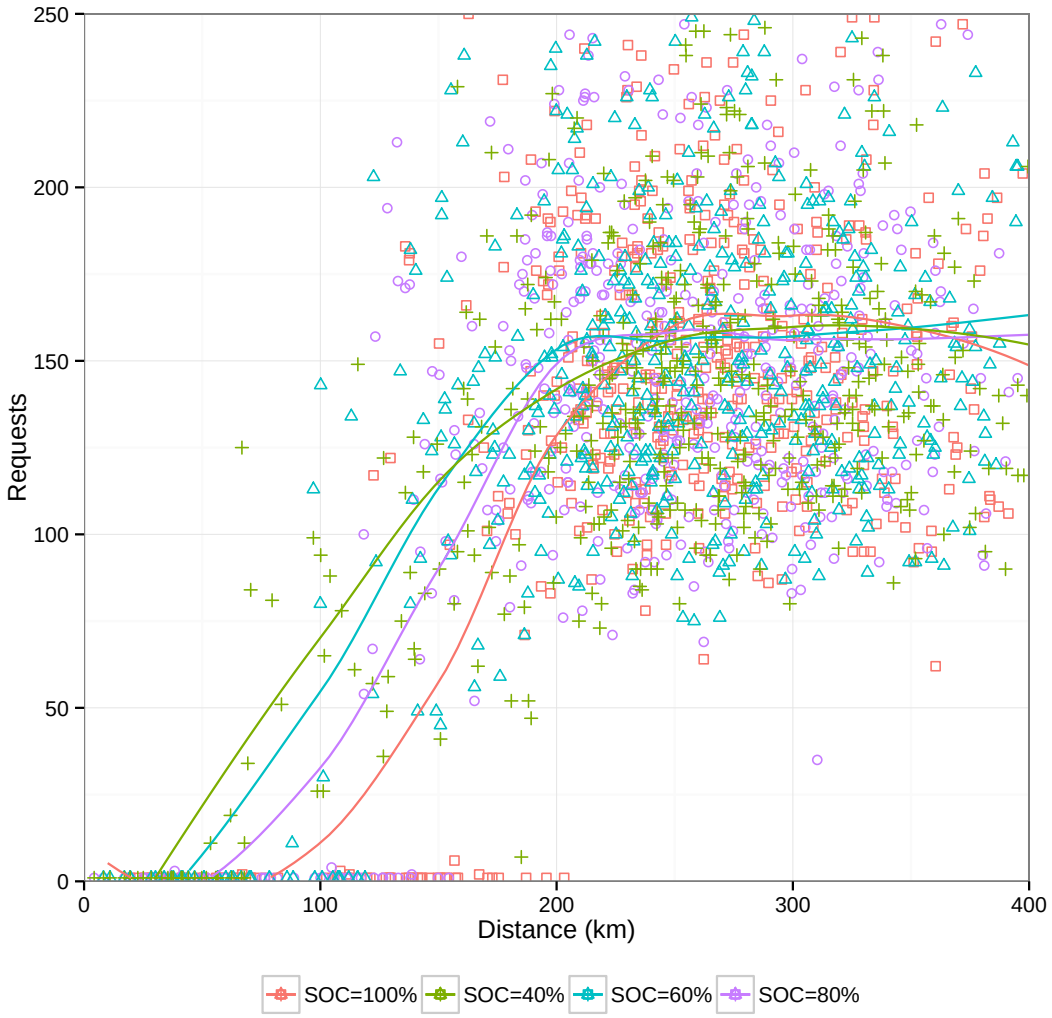

Figura 5.2: Numero di richieste alle Google Maps API al variare della distanza percorsa

É possibile osservare come il TR diminuisca con l'aumentare del valore di SOC (mediamente) e aumenti con la distanza, in quanto sarà maggiore il numero di EVSE inclusi nel grafo dei percorsi e più alta la probabilità di doverli aggiungere.
Infatti, dalla costruzione incrementale del grafo dei percorsi descritta in 4.3.1 deriva una suddivisione dei risultati sperimentali mostrati nel grafico in tre gruppi, relativi al raggiungimento della destinazione con necessità di nessuna, una o due ricariche rispettivamente. Nel primo caso viene effettuata una singola richiesta alle Directions API di

Google, relativa al percorso fra partenza e destinazione. Nel secondo caso le richieste crescono linearmente con il numero di stazioni di ricarica incluse nel grafico. Nel terzo caso il tempo aumenta considerevolmente, in quanto sarà necessario calcolare il percorso fra diverse coppie di stazioni.
Sebbene il caching dei percorsi fra EVSE ridurrebbe il TR nel terzo caso, assumendo la loro posizione come statica, ciò comporterebbe una perdita del fattore relativo alle condizioni del traffico (fornite in tempo reale da Google) nel calcolo del tempo necessario, ragione per cui una richiesta alle Directions API è comunque effettuata, negli esperimenti.
Nella figura 5.2 è mostrato il numero di richieste HTTP alle Directions API di Google Maps in funzione della distanza, evidenziando (in base all'analogia dei valori con quelli nel grafico 5.1) come queste siano la maggiore fonte di ritardo, specialmente se si considera il limite di richieste per secondo e le politiche di backoff adottate (vedi sezione 4.3.1); in questo secondo grafico i gruppi di risultati sono due: quelli ad una singola richiesta (fattibilità del percorso diretto) e i rimanenti.
In ogni caso il TR non sembra superare i due minuti, anche per percorsi di 400 chilometri, fornendo un risultato più che accettabile per un servizio di route planning (tipicamente utilizzato una singola volta, nell'atto di partire).
Nelle figure 5.3 e 5.4 sono mostrati il tempo di risposta e il numero di richieste in funzione dell'energia consumata durante il percorso. La variazione dei valori è analoga a quella nei grafici per la distanza, confermando come sia effettivamente il consumo dato dall'aumentare di essa a portare ad un aumento della complessità in tempo dell'algoritmo, in termini asintotici quanto effettivi.
Generalmente i valori maggiori sull'asse delle ascisse corrispondono a percorsi più lunghi o caratterizzati da un consumo più alto dovuto a profili altimetrici meno regolari (con più salite e discese).
Per ognuno dei grafici mostrati è valida la considerazione che, poiché i percorsi sono calcolati casualmente e il fattore altimetrico non è mostrato nei risultati, alcune fluttuazioni nei valori, indipendenti da SOC e distanza, sono più che plausibili.

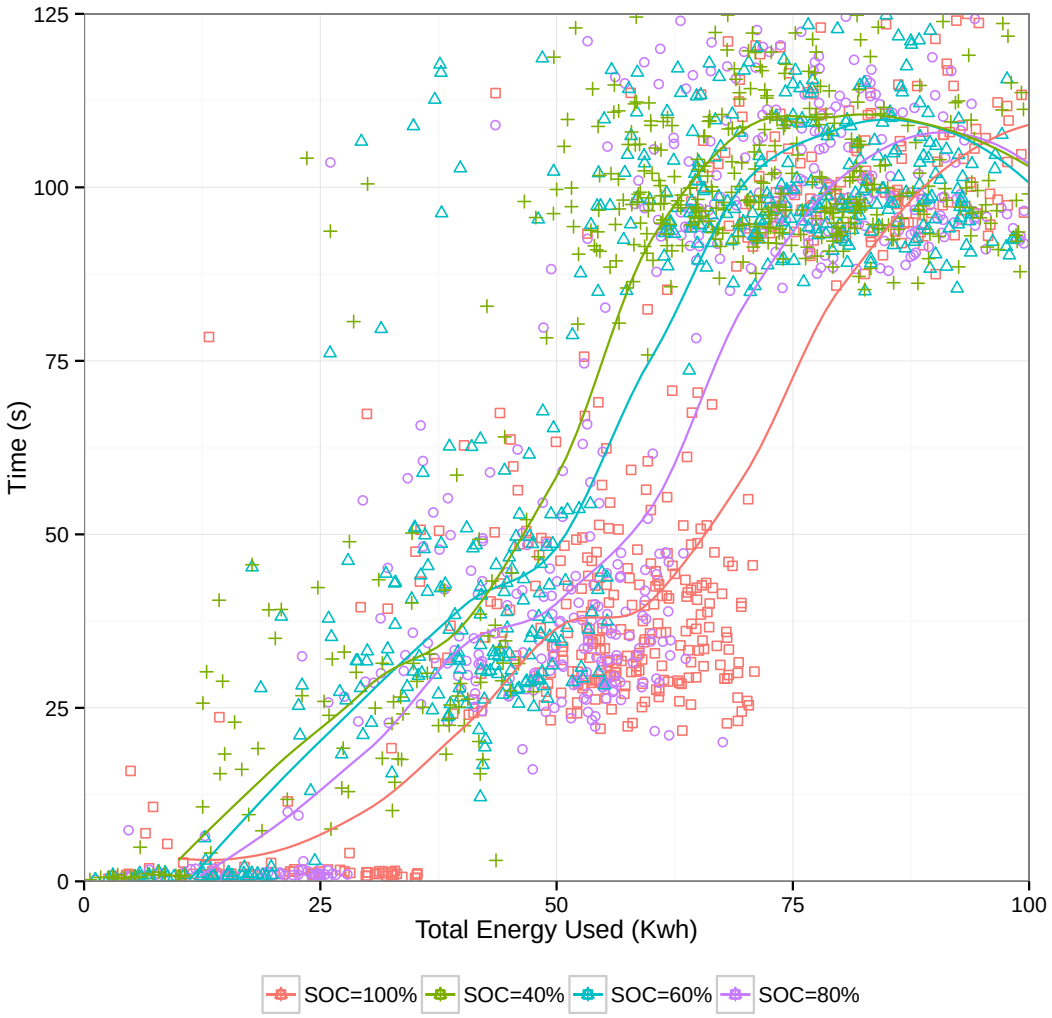

Figura 5.3: Tempi di risposta del servizio al variare dell'energia consumata nel percorso

5.2 Validazione

Per dimostrare che il servizio di planning sia effettivamente di supporto alla riduzione dei problemi derivanti da ristrettezze energetiche nei lunghi percorsi, è stato svolto un confronto fra i risultati ottenuti da due diverse configurazioni:

1. Servizio utilizzato: il guidatore di EV segue le sue istruzioni (percorso e fermate per ricarica).

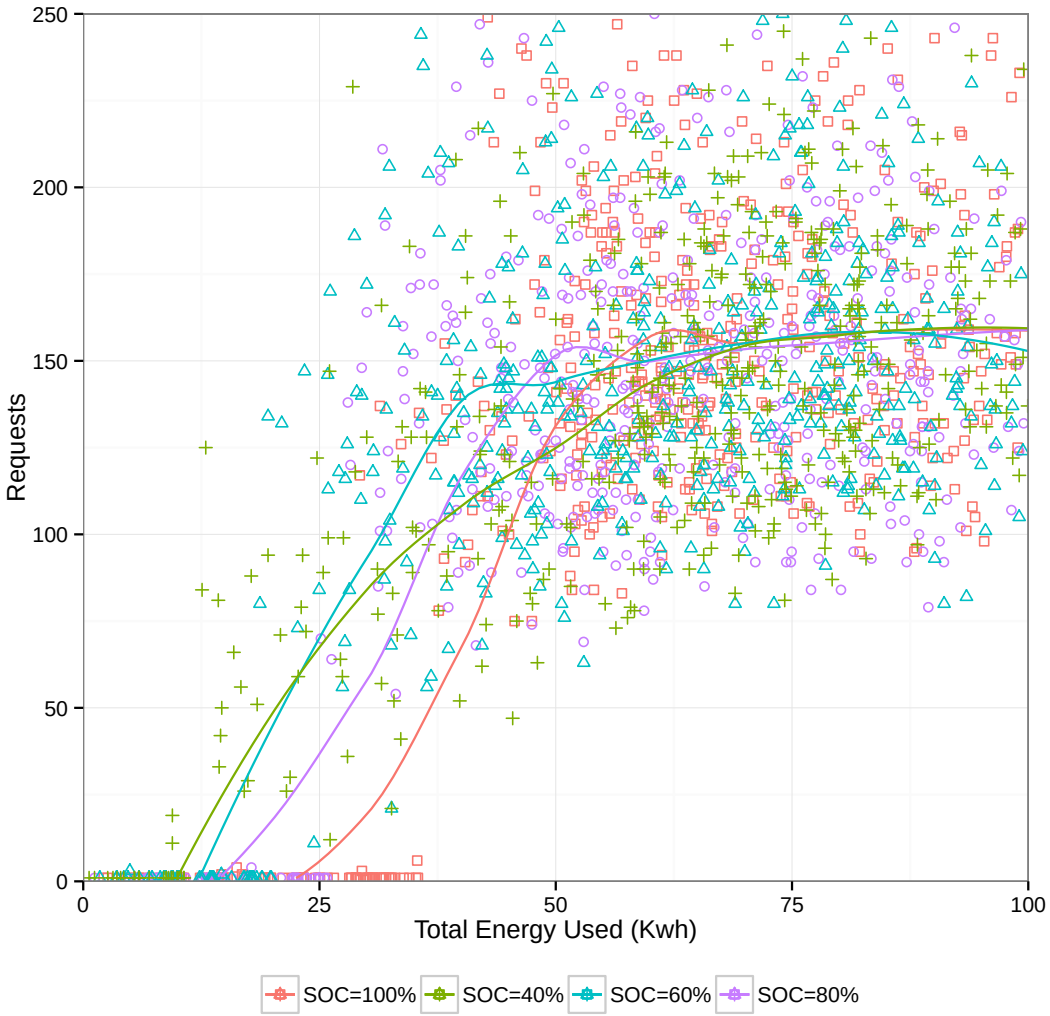

Figura 5.4: Numero di richieste alle Google Maps API al variare dell'energia consumata nel percorso

2. Servizio non utilizzato: il guidatore di EV segue il percorso più breve (in termini di distanza) per raggiungere la destinazione (è possibile immaginare questa informazione come fornita da un navigatore), alterando la sua rotta per ricaricarsi presso la stazione più vicina quando lo stato di carica scende sotto una certa threshold.

La seconda configurazione può essere considerata come un modello realistico del reale comportamento dei guidatori di EV.

Al fine di simulare scenari di guida senza supporto di planning è stata aggiunta la richiesta */planless* ai metodi nell'interfaccia esposta dal servizio, i cui *handler* sono implementati dalla classe Request (4.2.2).
Il meccanismo è semplice: data la Route più breve (in termini di distanza) fra partenza e destinazione, si controlla che il consumo totale non porti lo stato di carica sotto la threshold data (fissata al 10% della capacità della batteria, per questa analisi); se ciò accade, si ricalcola una Route a partire dal primo punto raggiunto che viola tale condizione e verso la stazione di ricarica più vicina. Nel caso in cui la stazione non sia raggiungibile con lo stato di carica in quel punto, il servizio restituisce un errore, altrimenti il procedimento viene ripetuto ripartendo dalla stazione raggiunta e considerando come SOC la capacità massima della batteria (assumendo una potenziale piena ricarica), finché non si raggiunge la destinazione.
Onde evitare che il veicolo ritorni più di una volta nella medesima stazione, risultando in un potenziale ciclo infinito fra due o più punti, la procedura mantiene un insieme di stazioni già visitate e le esclude dalla ricerca.
Nelle figure 5.5 e 5.6 sono confrontati i risultati ottenuti con le due configurazioni, utilizzando gli stessi parametri indicati nell'analisi di performance (5.1), in termini di percentuale di successo nel raggiungimento della destinazione (PS); nel caso della configurazione con planning, il fallimento equivale alla costruzione di un grafo dei percorsi in cui partenza e destinazione non sono collegate da un cammino (a causa di ristrettezze energetiche).
Le linee tratteggiate e piene rappresentano rispettivamente la configurazione senza e con ausilio del planning. Le curve mostrate sono relative a diversi stati di carica iniziale della batteria.
Il grafico 5.5 mostra la PS al variare della distanza. Come ci si potrebbe aspettare, la PS è direttamente proporzionale al valore di SOC negli esperimenti per entrambe le configurazioni, poiché da questo dipende la possibilità o meno di raggiungere la prima stazione.
In media, l'utilizzo del servizio di planning incrementa notevolmente la PS per ogni valore di SOC. In particolare è possibile osservare come la PS decada molto velocemente con l'aumento della distanza, nei risultati senza planning, portando ad una differenza sempre

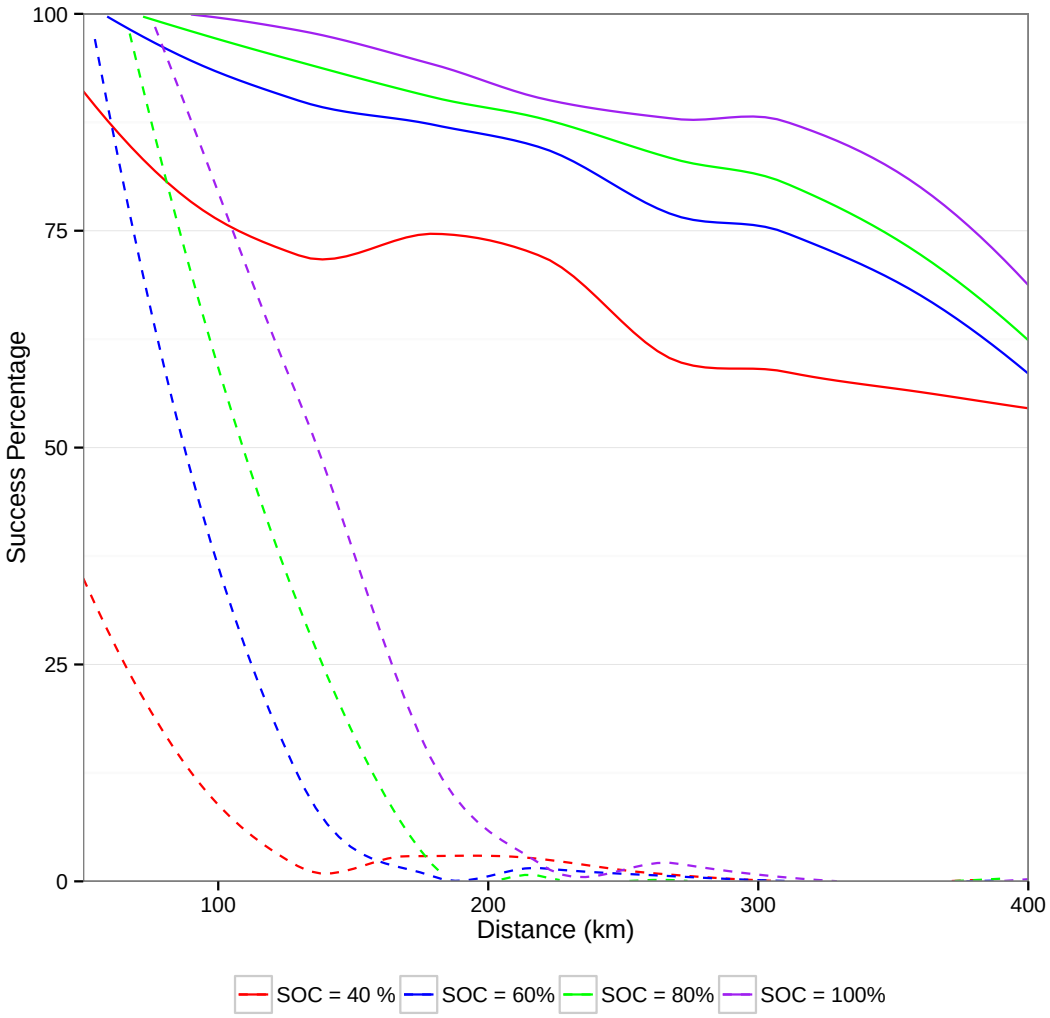

Figura 5.5: Percentuale di successo al variare della distanza percorsa

più marcata con quelli che lo includono.

Nel grafico in figura 5.6 sono mostrati i diversi valori della PS al variare del consumo energetico complessivo: valgono le medesime considerazioni fatte nella sezione 5.1.

Questi risultati forniscono una prova dell'efficacia del servizio, la cui realizzazione mira effettivamente al supporto della mobilità veicolare elettrica su percorsi di media/lunga distanza.

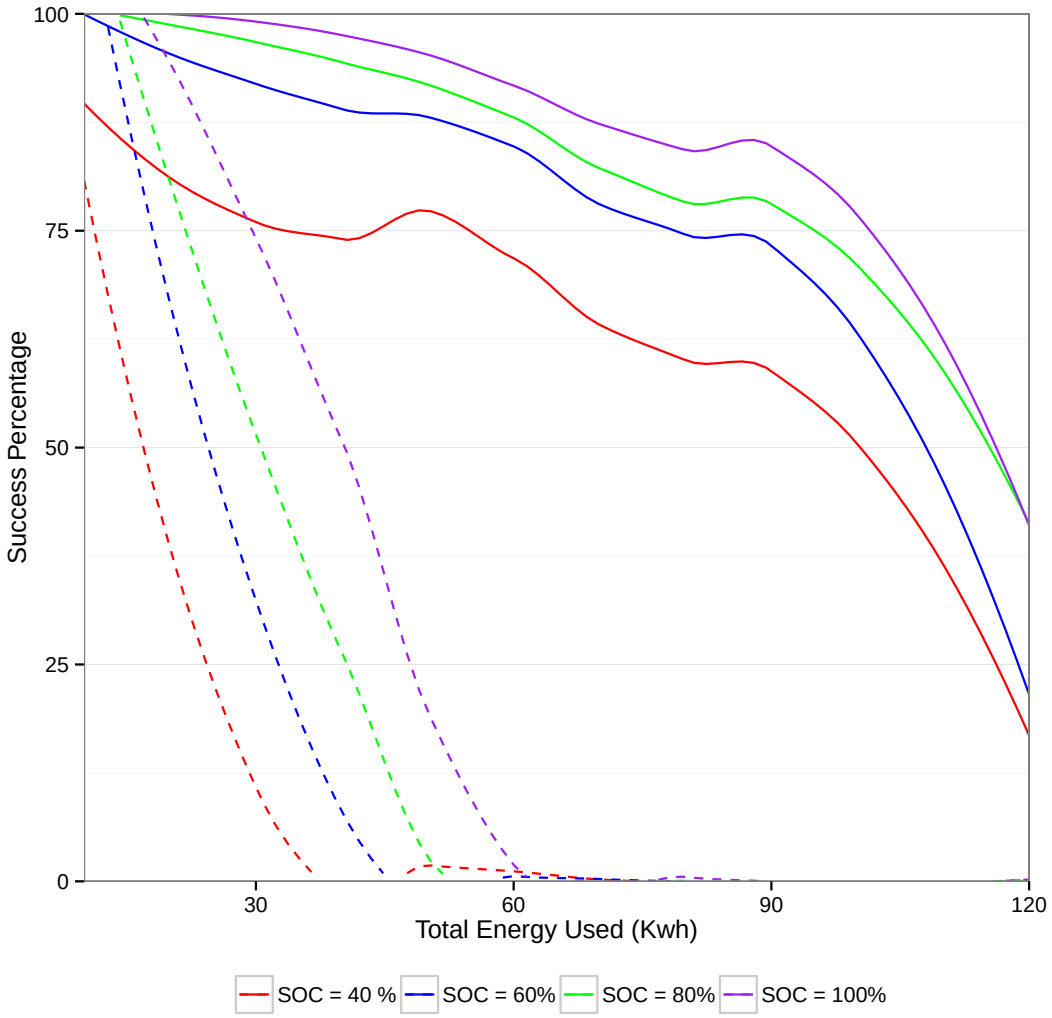

Figura 5.6: Percentuale di successo al variare dell'energia consumata nel percorso

5.2.1 Modello di consumo

Sfruttando la piattaforma di simulazione descritta in sezione 1.1.2, sono state svolte alcune analisi del modello di consumo implementato al suo interno (molto simile a quello utilizzato dal servizio), al fine di ottenerne una validazione.
Il grafico in figura 5.7 mostra il risultato di diverse *run* di simulazione sullo scenario urbano di Bologna.

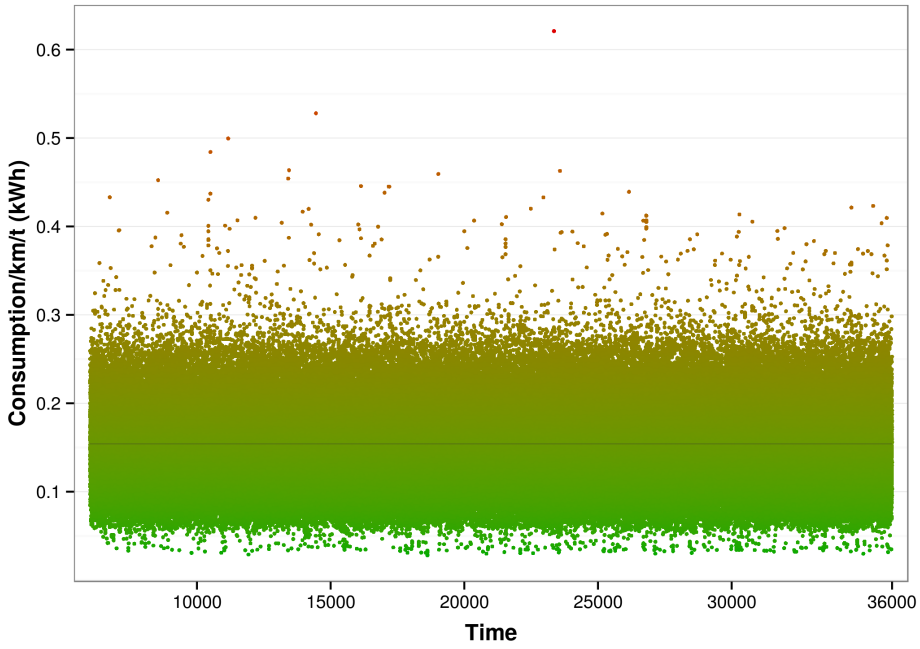

Figura 5.7: Risultati per consumo kWh/km/t in ambiente urbano

Ogni punto indica il consumo medio (kWh per chilometro per tonnellata) per un singolo veicolo sulle diverse run; i veicoli all'interno della simulazione hanno le medesime caratteristiche fisiche e percorrono diversi tragitti selezionati casualmente. La sottile striscia nera rappresenta gli intervalli di confidenza all'interno dei quali è situata la media globale del consumo, con un livello di confidenza del 95%, pari ad un valore di circa 0.15416 kWh/km/t.

L'obiettivo di questa serie di esperimenti è stato quello di dimostrare come il consumo medio calcolato convergesse effettivamente su un valore vicino a quello reale, prodotto da risultati raccolti da veicoli reali.

Il grafico in figura 5.8 mostra il consumo medio (kWh/km) al variare di diverse caratteristiche del veicolo e pendenza della strada, ma mantenendo una velocità pressoché costante di 50 km/h; lo scenario, in questo caso, è un semplice rettilineo di 200 chilometri prodotto utilizzando gli strumenti di SUMO (vedi 1.1.2).

Questa seconda analisi permette di osservare come il consumo energetico, calcolato utilizzando il modello implementato, vari in modo significativo con i parametri del veicolo

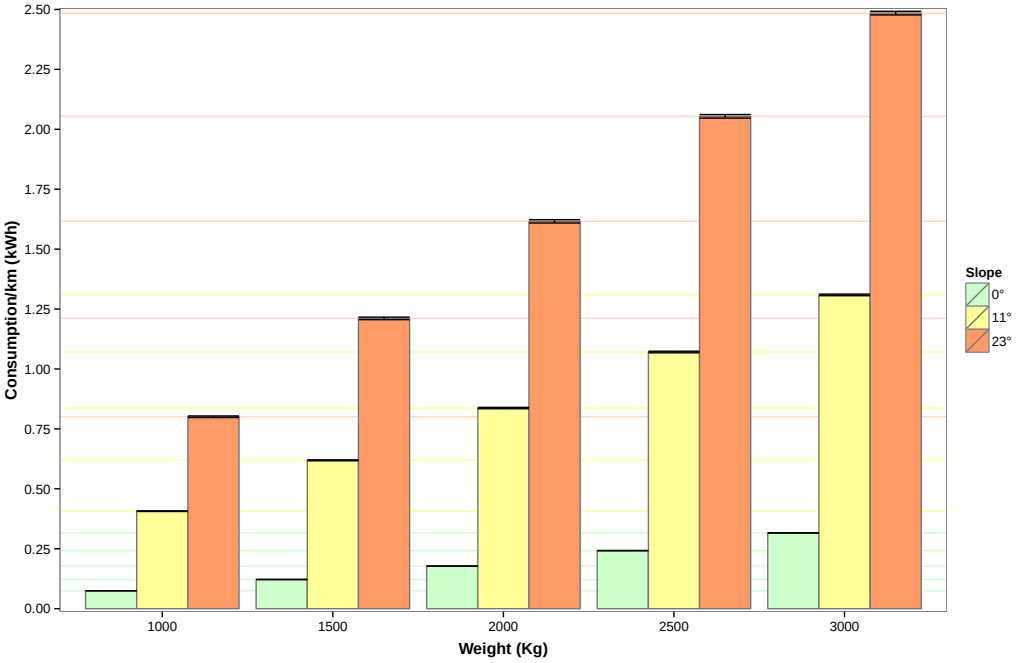

Figura 5.8: Consumo per diverse caratteristiche del veicolo e pendenze di 0, 11 e 23 gradi

(massa, superficie della massima sezione trasversale e coefficiente di forma) e con la pendenza della strada, come ci si aspetterebbe. Sebbene i grafici riportino solo il peso (per motivi di esposizione), i parametri variano da quelli che potrebbe avere una Fiat 500e (peso poco maggiore di 1 tonnellata, area della sezione trasversale pari a 1,4 m^2 circa) fino ad un modello Electric Daily e poco oltre (fino a 3 tonnellate di peso).

Conclusioni

In questo documento è stato descritto un servizio di route planning per veicoli elettrici, volto a supportare la guida di EV su percorsi medio-lunghi e mitigare la *range anxiety* dovuta a vincoli energetici e conseguente autonomia limitata.
Il servizio, realizzato come un *web service* con interfaccia RESTful, utilizza tre componenti fondamentali:

- Le API di Google Maps per ottenere indicazioni stradali tra due punti e dati altimetrici del percorso.
- Un modello di predizione del consumo energetico nell'attraversare un tratto di strada.
- Un database di EVSE, aggiornato con i dati offerti da diversi servizi.

Aggregando le funzionalità di questi componenti il planner è in grado costruire un grafo dei possibili percorsi tra due punti e le vicine stazioni di ricarica, calcolare una stima di consumo e tempo impiegato per ognuno di essi e pianificare il tragitto che consenta al veicolo di arrivare a destinazione con un dato livello di carica residua, minimizzando consumo o tempo.
Seguono adesso le proposte di alcuni sviluppi futuri, atti a rendere più accurate le valutazioni del planner o migliorarne le prestazioni.
Il primo sviluppo consiste nell'affinare l'algoritmo per il calcolo del percorso più economico in termini di tempo.
Attualmente l'unico fattore che l'algoritmo considera è il tempo totale necessario a percorrere il tragitto, anche in presenza di eventuali ricariche. Al fine di rendere la stima

di tempo più accurata, occorrerebbe considerare anche il tempo necessario alle ricariche stesse; naturalmente questo dipende dalla quantità di energia ricaricata, ma anche dalle modalità di ricarica: da un tempo inferiore ai 40 minuti necessari alla ricarica di terzo livello fino alle ore necessarie per ricaricare a 120 Volt/16 Ampere con una ricarica di primo livello.
In prima istanza ci si potrebbe limitare ad aggiungere i tempi necessari a ogni ricarica, considerando un quantitativo di energia ricaricata minimo (sufficiente a raggiungere la destinazione da quel punto), modalità di ricarica disponibili presso la stazione, caratteristiche fisiche della batteria e programmi dell'utente (una ricarica "overnight" potrebbe utilizzare il lungo, ma "salutare", primo livello).
Usando il planner in combinazione con un servizio di prenotazione delle ricariche (vedi sezione 1.1) sarebbe possibile aggiungere l'ulteriore considerazione della disponibilità della stazione scelta, dirigendo l'utente verso stazioni meno richieste o consentendo di prenotare una ricarica automaticamente, come parte della selezione del percorso.
Da migliorie nel calcolo del tempo impiegato deriverebbero anche maggiori potenzialità nello sfruttare le informazioni sul traffico in tempo reale fornite da Google. Potrebbe anche essere aggiunta la metrica del costo monetario, considerando il prezzo dell'energia in relazione alle necessità di ricarica.
Il calcolo del consumo energetico stesso potrebbe essere affinato, prendendo in considerazione i fattori discussi in sezione 3.2.2.
Un altro utile sviluppo consiste nel cambiare il metodo con cui le indicazioni stradali e dati altimetrici sono ottenuti: utilizzare servizi alternativi a Google Maps.
Come descritto in 4.3.1, la limitazione più grande nell'utilizzo delle API di Google è data dalla quota massima di richieste per secondo e giornaliere. Questi limiti impongono l'utilizzo di politiche di backoff delle richieste con aggiunta di ritardi casuali ed euristiche per ridurre il più possibile la dimensione del grafo dei percorsi.
Una possibile alternativa consiste nei *MapQuest Open Platform Web Services* [map], i quali utilizzano i dati di *OpenStreetMap* per fornire una serie di servizi analoghi a quelli offerti da Google Maps: Geocoding, Directions, Elevation e altri.
La qualità dei servizi potrebbe risultare diversa da quelli offerti da Google (considerando la natura open dell'accesso ai dati), ma le limitazioni menzionate sarebbero eliminate.

Eventualmente si potrebbe combinare l'utilizzo di entrambi i servizi, così da sfruttare MapQuest qualora Google non fosse disponibile o le richieste eccedano i limiti imposti (ad esempio in termini del numero di punti consecutivi su un percorso, dei quali è necessaria l'altitudine).
Infine, sarebbe possibile sviluppare applicazioni mobili come front-end per il servizio; tra i vantaggi dell'architettura REST figurano la natura stateless della comunicazione e la facile invocazione dei metodi offerti, rendendola particolarmente adatta all'interazione con dispositivi mobili.
Per concludere, è possibile affermare che un servizio di planning efficace e altamente disponibile costituirebbe un notevole supporto all'accettazione della guida di veicoli elettrici, riducendo le problematiche date dalla scarsa densità di infrastrutture nel territorio e autonomia inferiore ai veicoli a combustibili fossili; questo obiettivo sarà consolidato avendo a disposizione maggiori informazioni sugli EVSE disponibili e l'ausilio di servizi per facilitare le operazioni di ricarica, risultando in un ulteriore passo avanti verso la sostenibilità ambientale e sociale.

Appendice A

Formato delle risposte

In questa appendice sono dati alcuni esempi di risposte che il servizio fornisce alle richieste descritte nella sezione 4.2.1, inclusi eventuali messaggi di errore.
Il formato JSON dei messaggi è descritto e validato da un JSON Schema, ottenibile con la richiesta GET */schema*; per alleggerire la notazione, il valore intero Javascript "Infinity" è espresso con il simbolo matematico e viene omesso l'URL encoding dei caratteri, nel path degli URL.

Richiesta:

/route/Via Galliera 62 Bologna/Via Giuseppe Petroni Bologna/Daily Electric/20/1/1/∞/consumption

Risposta:


```
{
  status: "OK",
  route: {
    summary: "SS9",
    origin: {lat: 44.5019273, lng: 11.3437333, elevation: 60.79116544149412 },
    destination: {lat: 44.4954948, lng: 11.3511191, elevation: 61.67819213867188 },
    startAddress: "Via Galliera, 62, 40121 Bologna, Italy",
    endAddress: "Via Giuseppe Petroni, Bologna, Italy",
    drivingDistance: 2649,
    duration: 472,
    consumption: 2.0877618448721273,
    consumptionPercentage: 5.219404612180318,
    rechargeNeeded: 0,
    rechargeNeededPercentage: 0,
```

```
    steps: [{
      origin: {lat: 44.5019273, lng: 11.3437333, elevation: 60.79116544149412 },
      destination: {lat: 44.5014037, lng: 11.3434943, elevation: 61.86153042181403 },
      drivingDistance: 61,
      duration: 6,
      htmlInstructions: "Head <b>south</b> on <b>Via Galliera</b> toward <b>Via dei Mille</b>",
      encodedPolyline: "axrnGiqfdAhBn@",
      consumption: 0.35233170656676549
    },
    ...,
    {
      origin: {lat: 44.4961747, lng: 11.3504443, elevation: 62.16114044189453 },
      destination: {lat: 44.4954948, lng: 11.3511191, elevation: 61.67819213867188 },
      drivingDistance: 93,
      duration: 18,
      htmlInstructions: "Turn <b>right</b> onto <b>Via Giuseppe Petroni</b>",
      encodedPolyline: "atqnGg{gdAVYhBiBDC",
      consumption: 0.4966770656718548
    }]
  },
  chargingStations: []
}
```


Sono fornite informazioni circa i punti di partenza e destinazione (coordinate geografiche ed indirizzo), distanza totale percorsa e tempo impiegato, quantitativo di energia consumata (in kWh e termini percentuali sulla capacità massima della batteria), eventuale ricarica necessaria e la sequenza ordinata di step in cui la route è suddivisa.
Le informazioni su ogni step comprendono le istruzioni per attraversare il tratto di strada relativo ad esso (con markup HTML, così come sono restituite da Google Maps) e un'*encoded polyline* che codifica i punti attraversati dallo step in un formato testuale e conciso; la polyline può essere utilizzata per visualizzare lo step su mappa, come una linea che interpola i punti al suo interno.
La route pianificata non prevede ricariche, ragione per cui l'array *chargingStations* è vuoto.

Richiesta:

/route/Via Galliera 62 Bologna/44.4954948,11.3511191/Daily Electric/2/1/1/∞/consumption

Risposta:


```
{
  status: "NO_ROUTE_AVAILABLE",
  route: {
    summary: "SS9",
    origin: {lat: 44.5019273, lng: 11.3437333, elevation: 60.79116544149412 },
    destination: {lat: 44.4954948, lng: 11.3511191, elevation: 61.67819213867188 },
    startAddress: "Via Galliera, 62, 40121 Bologna, Italy",
    endAddress: "Via Giuseppe Petroni, 6-14, 40126 Bologna, Italy",
    drivingDistance: 2649,
    duration: 472,
    consumption: 2.128882755017942,
    consumptionPercentage: 5.322206887544855,
    rechargeNeeded: 1.728882755017942,
    rechargeNeededPercentage: 4.322206887544855,
    steps: [... ]
  }
}
```


I punti di partenza e destinazione sono i medesimi della richiesta precedente (con il secondo espresso in coordinate geografiche). Il codice *NO_ROUTE_AVAILABLE* rappresenta l'impossibilità di arrivare a destinazione con i vincoli energetici forniti (stato di carica iniziale e threshold intermedia o di arrivo). Le informazioni sulla route diretta da partenza a destinazione sono restituite in ogni caso.

Richiesta:

/route/Via Galliera 62 Bologna/44.532432,11.287664/Daily Electric/25/1/75/∞/consumption

Risposta:


```
{
  status: "OK",
  route: {
    summary: "SS9 | Via del Triumvirato",
    origin: {lat: 44.5019273, lng: 11.3437333, elevation: 60.79116544149412 },
    destination: {lat: 44.531467, lng: 11.2883235, elevation: 62.768216732811 },
    startAddress: "Via Galliera, 62, 40121 Bologna, Italy",
```

```
    endAddress: "Via dell'Aereoporto, 12, 40132 Bologna, Italy",
    drivingDistance: 10458,
    duration: 1131,
    consumption: 9.468216040646698,
    consumptionPercentage: 23.670540101616744,
    rechargeNeeded: 29.4682160406467,
    rechargeNeededPercentage: 73.67054010161675,
    steps: [{
      origin: {lat: 44.5019273, lng: 11.3437333, elevation: 60.79116544149412 },
      destination: {lat: 44.5014037, lng: 11.3434943, elevation: 61.86153042181403 },
      drivingDistance: 61,
      duration: 6,
      htmlInstructions: "Head <b>south</b> on <b>Via Galliera</b> toward <b>Via dei Mille</b>",
      encodedPolyline: "axrnGiqfdAhBn@",
      consumption: 0.35233170656676549
    },
    ...,
    {
      chargeNeeded :34.77767544015039,
      chargeNeededPercentage :86.94418860037597,
      coordinates: {lat: 44.49186, lng: 11.33006, elevation: 61.5342042111437 },
      htmlInstructions: "Recharge to 5.01 % (2.00 kWh) at the station."
    },
    ...,
    {
      origin: {lat: 44.5261344, lng: 11.2850271, elevation: 62.5483567110913 },
      destination: {lat: 44.531467, lng: 11.2883235, elevation: 62.768216732811 },
      drivingDistance: 649,
      duration: 64,
      htmlInstructions: "Turn <b>right</b> to stay on <b>Via dell'Aereoporto</b><div
          style=\"font-size:0.9em\">Partial restricted usage road</div>",
      encodedPolyline: "iownGmb{cAwCoAuAm@_@QOI{Am@sAu@iAq@sAy@_Ai@eGeD_BgAYO",
      consumption: 0.6000756359213775
    }],
  },
  chargingStations: [{
    summary: "Via Ugo Foscolo, Bologna, Italy",
    coordinates: {lat: 44.49186, lng: 11.33006 },
    connections: [{level: 2,amps: 200, voltage: 240, powerKW: 4}]
  }]
}
```


Nel caso di una o più soste per ricarica previste dal piano formulato, la risposta comprenderà due aggiunte:

- Uno step di ricarica in corrispondenza di ogni stazione presso la quale il veicolo

deve sostare, indicante il quantitativo minimo di carica che il veicolo deve avere prima di lasciarla per proseguire il viaggio.

- Informazioni sulle stazioni attraversate all'interno dell'array chargingStations, corrispondenti agli step di ricarica introdotti nella descrizione del percorso.

Richiesta:

/stations/Via Galliera 62 Bologna/10000/∞

Risposta:

```
{
  origin: {lat: 44.5019232, lng: 11.3437509},
  chargingStations: [{
    summary: "Via Ugo Foscolo, Bologna, Italy",
    coordinates: {lat: 44.49186, lng: 11.33006 },
    connections: [{level: 2, amps: 200, voltage: 240, powerKW: 4}]
  },
  ...
  ]
}
```

Una risposta alla richiesta */stations* consiste nel riferimento al punto fornito (convertito in coordinate geografiche) e nel vettore di stazioni trovate.

Richiesta:

/route/abc/def/Daily Electric/25/1/75/∞/consumption

Risposta:

```
  404 Not Found
{
  code: "NotFoundError"
  message: "Couldn't find any route to destination."
}
```

Al fine di distinguere il caso in cui il percorso fra partenza e destinazione esiste ma non è fattibile a causa di ristrettezze energetiche, da quello in cui i due punti non sono collegati

da rete stradale (ad esempio perché uno dei due non esiste), in quest'ultima evenienza il servizio restituisce un messaggio di errore con codice HTTP 404.

Richiesta:

/route/abc

Risposta:

```
  400 Bad Request
{
  code: "BadRequestError"
  message: "Missing one or more required parameters."
}
```

Se il path della richiesta non comprende tutti i parametri necessari, è restituito un messaggio di errore con codice HTTP 400, a indicare la mancanza di parametri necessari (primo fra tutti la destinazione, in questo esempio).

Elenco delle figure

Bibliografia

[AHLS10] Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. *The shortest path problem revisited: Optimal routing for electric vehicles*. In KI 2010: Advances in Artificial Intelligence, pages 309–316. Springer, 2010.

[BBD+14] Luca Bedogni, Luciano Bononi, Alfredo D'Elia, Marco Di Felice, Marco Di Nicola, and Tullio Salmon Cinotti. *Driving without anxiety: A Route Planner Service with Range Prediction for the Electric Vehicles*. Submitted for conference publication, 2014.

[BBDF+13] Luca Bedogni, Luciano Bononi, Marco Di Felice, Alfredo D'Elia, Randolf Mock, Federico Montori, Francesco Morandi, Luca Roffia, Simone Rondelli, Tullio Salmon Cinotti, et al. *An interoperable architecture for mobile smart services over the internet of energy*. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a, pages 1–6. IEEE, 2013.

[BBDF+14] L. Bedogni, L. Bononi, M. Di Felice, A. D'Elia, R. Mock, F. Montori, F. Morandi, L. Roffia, S. Rondelli, T. Salmon Cinotti, and F. Vergari. *An Integrated Simulation Framework to Model Electric Vehicles Operations and Services*. Submitted for journal publication, 2014.

[BCMW11] Albert G Boulanger, Andrew C Chu, Suzanne Maxx, and David L Waltz. *Vehicle electrification: Status and issues*. Proceedings of the IEEE, 99(6):1116–1138, 2011.

[DdV13] Yuhuan Du and Gustavo de Veciana. *Mobile applications and algorithms to facilitate electric vehicle deployment.* In Consumer Communications and Networking Conference (CCNC), 2013 IEEE, pages 130–136. IEEE, 2013.

[EFS11] Jochen Eisner, Stefan Funke, and Sabine Storandt. *Optimal Route Planning for Electric Vehicles in Large Networks.* In AAAI, 2011.

[FMA13] J Ferreira, Vítor Monteiro, and J Afonso. *Vehicle-to-Anything Application (V2Anything App) for Electric Vehicles.* IEEE Transactions on Industrial Electronics, 99(1), pp.1-11, 2013.

[Lab11] U.S. National Energy Technology Laboratory. *Assessment of Future Vehicle Transportation Options and Their Impact on the Electric Grid.* Report DOE/NETL-2010/1466, 2011.

[MLA+11] Nils Masuch, M Lutzenberger, Sebastian Ahrndt, Axel Heßler, and Sahin Albayrak. *A context-aware mobile accessible electric vehicle management system.* In Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on, pages 305–312. IEEE, 2011.

[MSAN11] Ricardo Maia, Marco Silva, Rui Araújo, and Urbano Nunes. *Electric vehicle simulator for energy consumption studies in electric mobility systems.* In Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on, pages 227–232. IEEE, 2011.

[NHHB13] Myriam Neaimeh, Graeme A Hill, Yvonne Hübner, and Phil T Blythe. *Routing systems to extend the driving range of electric vehicles.* IET Intelligent Transport Systems, 7(3):327–336, 2013.

[OWB13] Javier A Oliva, Christoph Weihrauch, and Torsten Bertram. *A Model-Based Approach for Predicting the Remaining Driving Range in Electric Vehicles.* In Proc. of PHM Society Conference, New Orleans, USA, 2013.

[Sna02] Matthew C Snare. *Dynamics model for predicting maximum and typical acceleration rates of passenger vehicles.* PhD thesis, Virginia Polytechnic Institute and State University, 2002.

[SWN13] Tai Stillwater, Justin Woodjack, and Michael Nicholas. *Mobile App Support for Electric Vehicle Drivers: A Review of Today's Marketplace and Future Directions.* In Human-Computer Interaction. Applications and Services, pages 640–646. Springer, 2013.

Sitografia

[car] *The CarStations project*. http://www.carstations.com.

[ECO] Inc. ECOtality. *The Blink Mobile Application*. http://www.ecotality.com/.

[ene] *The Enel Drive project*. http://www.eneldrive.it/.

[goo] *Google Maps API Web Services*. https://developers.google.com/maps/documentation/webservices/.

[gre] *The Green eMotion project*. http://www.greenemotion-project.eu/.

[ioe] *Internet of Energy (IoE) - ARTEMIS European Project*. Project Website: http://www.artemis-ioe.eu/.

[map] *MapQuest Open Platform Web Services*. http://open.mapquestapi.com/.

[mon] *MongoDB*. http://www.mongodb.org/.

[nod] *Node.js*. http://nodejs.org/.

[ope] *The Open Charge Map project*. http://openchargemap.org/site/.

[plu] *The PlugShare project*. http://www.plugshare.com/.

[Tec] Inc. Coulomb Technologies. *The ChargePoint Application*. https://www.chargepoint.net.au/.

Ringraziamenti

Desidero innanzitutto ringraziare il Professor Luciano Bononi per avermi dato l'opportunità di lavorare a questo progetto di tesi e di dare un piccolo contributo al progetto Internet of Energy.
Voglio ringraziare Luca Bedogni e Marco Di Felice, miei correlatori, per aver lavorato a questo progetto insieme a me e per avermi fornito tutto il supporto di cui necessitavo. I brain-storming per discutere degli aspetti concettuali del lavoro e suoi potenziali sviluppi, svolti in questo o quell'ufficio, sono stati insieme stimolanti e divertenti, così come il tempo impiegato per configurare e realizzare (con una sana dose di buchi nell'acqua) le simulazioni necessarie all'analisi. Ringrazio anche Simone Rondelli, per avermi introdotto al progetto IoE e, di conseguenza, spinto a scegliere questo argomento di tesi.
Ringrazio Eugenia per essermi stata vicina in quest'ultimo periodo universitario e per tutta la disponibilità ad ascoltare, di tanto in tanto, i dettagli di un lavoro che (in fondo e comprensibilmente) non destava il suo massimo interesse.
Ringrazio i miei amici e compagni di studio per tutto il supporto dato e la continua disponibilità a chiacchierare del mio lavoro e fornire pareri o spunti interessanti.
Per ultima ma non meno importante, ringrazio anche la mia famiglia, per esserci stata sempre e comunque.