Hugo
Silva

Integração de um planeador de rotas para veículos elétricos numa arquitetura em microsserviços: design e implementação

Integrating an electric vehicle route planner into a microservices architecture: design and implementation

Hugo
Silva

Integração de um planeador de rotas para veículos elétricos numa arquitetura em microsserviços: design e implementação

Integrating an electric vehicle route planner into a microservices architecture: design and implementation

"*It always seems impossible until it's done.*"

— Nelson Mandela

Universidade de Aveiro
2023

Hugo
Silva

Integração de um planeador de rotas para veículos elétricos numa arquitetura em microsserviços: design e implementação

Integrating an electric vehicle route planner into a microservices architecture: design and implementation

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Sérgio Matos, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este trabalho é exclusivamente dedicado ao meu falecido avô. Apesar de já não estar no meio de nós, era das pessoas que mais desejava que concluisse os meus estudos e que, por essa razão, me motivou durante os meus 5 anos deste percurso académico. Um muito obrigado.

o júri / the jury

presidente / president

Professor Doutor António Joaquim da Silva Teixeira
Professor Associado C/ Agregação, Universidade de Aveiro

vogais / examiners committee

Professor Doutor João André Pinto Soares
Investigador Auxiliar, Instituto Superior de Engenharia do Porto

Professor Doutor Sérgio Guilherme Aleixo de Matos
Professor Auxiliar em Regime Laboral, Universidade de Aveiro

agradecimentos / acknowledgements

Quero agradecer a todas as pessoas que fizeram parte deste percurso académico, principalmente, durante o mestrado. Um agradecimento especial à minha família por tornar este percurso possível, a todos os meus amigos, e à minha namorada que me acompanhou incansavelmente durante todo este percurso, principalmente pelo apoio em todos os momentos onde queria desistir e achava impossível a conclusão desta dissertação.

Quero agradecer ao meu orientador Professor Doutor Sérgio Guilherme Aleixo de Matos, por toda a disponibilidade que teve comigo durante este percurso, pela sua sabedoria, profissionalismo e orientação.

Também, ao meu orientador na minha atual empresa Miio electric, Me. Rafael Ferreira, quero agradecer por toda a ajuda, disponibilidade e principalmente por me introduzir ao mundo dos veículos elétricos. Obrigado pela oportunidade de poder colaborar na construção de um mundo melhor e mais sustentável todos os dias.

Um agradecimento especial, por fim, a todos os meus colegas de trabalho.

Muito obrigado!

Palavras-Chave

Mineração de Dados, Algoritmos de Decisão, Sistemas Distribuídos, Mobilidade Elétrica, Machine Learning, Arquitetura em Microsserviços, RESTful APIs, Planeamento de Rotas, Escalabilidade, Arquitetura Orientada em Serviços, Serviços Web

Resumo

O objetivo desta dissertação é desenvolver e integrar um planeador de rotas numa arquitetura em microsserviços. Nos últimos anos, a utilização da arquitetura em microsserviços tornou-se cada vez mais popular devido à sua capacidade de construir sistemas grandes e complexos, dividindo-os em serviços menores e independentes. Esta dissertação, propõe uma arquitetura em microsserviços composta por vários serviços, cada um responsável por uma tarefa específica no processo de planeamento de rotas. Os serviços comunicam entre si usando RESTFul APIs para fornecer a funcionalidade otimizada de planeamento de rotas. O sistema incorpora algoritmos de decisão e métodos de otimização para conseguir dar ao utilizador final, a melhor rota possível utilizando um pré-planeamento. O sistema é avaliado usando diferentes métricas, como, o tempo de resposta, escalabilidade e confiabilidade. Os resultados mostram que a arquitetura proposta em microsserviços fornece uma solução eficiente e confiável para o planeamento de rotas. O sistema também é comparado a outras ferramentas e arquiteturas de planemento de rotas para destacar alguns dos seus pontos fortes e limitações. Em conclusão, esta dissertação demonstra o potencial da utilização de uma arquitetura em microsserviços em Python, para construir um planeador de rotas para veículos elétricos escalável, eficiente e fiável. A arquitetura proposta pode ser aprimorada e expandida para incluir funcionalidades adicionais, tornando-se assim alvo de pesquisas futuras.

Keywords

Data Mining, Decision-Making Algorithms, Distributed Systems, Electric Mobility, Machine Learning, Microservices Architecture, RESTful APIs, Route Planning, Scalability Service-Oriented Architecture, Web Services

Abstract

The objective of this dissertation is to develop and integrate a route planner into a microservices architecture. In recent years, the use of microservices has become increasingly popular due to its ability to build large and complex systems by breaking them down into smaller and independent services. This dissertation suggests a microservices architecture comprising various services, each of which is in charge of a certain duty during the route planning process. The services interact with one another via RESTful APIs to offer the full capabilities of route planning. The system uses decision-making algorithms and optimization methods to give the user the best route it can, by using a pre-planned and calculated route. For evaluating the system, measures such as response time, scalability, and reliability, were used. The outcomes demonstrate that the suggested microservices architecture offers an effective, independent and scalable route planning solution. It is also contrasted with alternative route planning tools and architectures to highlight the system's advantages and disadvantages. In conclusion, this dissertation demonstrates the potential of using a microservices architecture with Python to build a scalable, efficient, and reliable route planning system. The proposed architecture can be further improved and expanded to include additional functionalities, making it a good area or starting point for future research.

Contents

List of Figures

List of Tables

Glossário

ABRP	A Better Route Planner
AHP	Analytic Hierarchy Process
API	Application Programming Interface
APIs	Application Programming Interfaces
DD	Decimal Degrees
EPA	Environmental Protection Agency
EPAF	Encoded Polyline Algorithm Format
EV	Electric Vehicle
EVDB	Electric Vehicle Database
EVs	Electric Vehicles
GIS	Geographic Information Systems
HTTPS	Hyper Text Transfer Protocol Secure
JSON	JavaScript Object Notation
kW	kilowatt
km	kilometer
ML	Machine Learning
NA	North America
NEDC	New European Driving Cycle
OCM	OpenChargeMap
OSM	Open Street Map
OSRM	Open Source Routing Machine
POI	Points of Interest
RAM	Random Access Memory
REST	Representational State Transfer
RESTful API	Representational State Transfer Application Programming Interface
SoC	State of Charge
UA	University of Aveiro
URI	Uniform Resource Identifier
WLTP	Worldwide Harmonized Light Vehicles Test Procedure
WSM	Weighted Scoring Method

CHAPTER 1

Introduction

The purpose of this dissertation is to present the development of a route planner using a microservices architecture for Electric Vehicles (EVs) users. In recent years, microservices have gained popularity due to their ability to simplify complex systems by dividing them into smaller and independent services [1]. This dissertation proposes a microservices architecture comprising multiple services, each with a specific responsibility in the route planning process. These services collaborate using Representational State Transfer (REST) Application Programming Interface (API)s, enabling users to benefit from the full range of route planning capabilities [2]. The system uses decision-making algorithms and optimization techniques to provide the user with the best possible route, using a pre-planned and calculated strategy.

1.1 Motivation

The objective of this dissertation is to develop a route planner using a microservices architecture that is specifically designed for electric vehicles. The proposed route planner will incorporate decision-making algorithms and optimization methods [3] to give users the best route possible.

The use of a microservices architecture will offer several advantages for the proposed route planner, including scalability, reliability, and efficiency [4]. By leveraging the microservices architecture and the latest software development practices, the proposed route planner can address some of the limitations that existing solutions have and offers a comprehensive and reliable route planning solution for EV drivers.

This dissertation intends to help the already existent community of EVs users, by developing and designing an open-sourced solution for EVs. The findings of this study may have consequences for a wide range of industries, such as logistics, urban planning, and transportation, and they may hasten the transition to a more sustainable transportation system [5].

1.2 Background

Nowadays, the popularity of electric cars (EVs) is growing, since they are a more practical and viable option when compared with internal combustion vehicles, encouraging the use of a more environment-friendly way of moving. But, the lack of practical route planning tools, range restrictions, energy price volatility, the infrastructure for charging on public roads, the amount of charging time, and the total range of a car, are some of the issues that are holding back the adoption of electric vehicles by some persons [6].

Route planning is an essential part when adopting an EV, since it may assist drivers in finding the most effective route to their destinations. When planning a route for an electric vehicle (EV), it is important to consider whether the user prioritizes the least energy-consuming route or the fastest one [7] [8]. Current most common route planning tools take into account the needs of conventional vehicles and do not take into account the special qualities of EVs, such as their range restrictions, energy usage, and the charging infrastructure (on public roads). Because of this, these solutions could not be appropriate for EV drivers, which could result in range problems when traveling and a lack of faith in EVs when doing a bigger trip.

For all of these reasons, there is an increasing need for EV specific route planning solutions to handle all of these challenges and limitations. An effective and accurate route planner for EVs would not only boost driver confidence when driving but would also aid in lowering carbon emissions by encouraging all other drivers to purchase an EV [9].

The development of such a route planner would require the use of advanced optimization algorithms and the latest software development practices. The use of microservices architecture has become increasingly popular in recent years as it enables the development of large and complex systems by breaking them down into smaller and independent services [10]. In this context, developing a route planner using this type of architecture can leverage the advantages of this project, facilitating future development in this research area since the route planner's main service could easily be integrated into other important services, such as a charging station locator service.

In conclusion, the development and study of an efficient, scalable, and comprehensive route planner for EVs is essential to address the challenges of this type of vehicle adoption and to promote sustainable ways of transportation in either the present or the future [5]. Using and integrating with a microservices architecture can enable the development of such a solution and can have implications for numerous fields.

1.3 Research problem and objectives

The problem addressed by this research is the lack of efficient and comprehensive route-planning solutions for electric vehicles. Traditional route planning solutions are designed with traditional vehicles in mind and do not take into account the unique characteristics of EVs, such as their range limitations and energy consumption [11]. This leads to limitations in existing route planning solutions and can cause a lack of confidence in EVs among drivers as already explained before.

The objective of this research is to develop a route planner specifically designed for EVs that can not only help address the limitations of existing route planning but also develop an open-sourced solution that can be easily integrated in the future in a potential microservices architecture.

This research intends to assess the suggested solution using metrics like response time, memory usage, scalability, and reliability in addition to developing the route planner. The performance of the suggested solution will be compared with the current route planning systems in the EV ecosystem, and its advantages and disadvantages will be highlighted in the evaluation.

The goal of this research overall was to develop an effective route planning solution for EVs and with this understanding of all the limitations and challenges that make route planning for EVs a complex topic, supporting this way environmentally friendly mobility and potential future research.

CHAPTER 2

Literature review

A potential approach to lowering carbon emissions and promoting environmentally friendly transportation has been made possible by EVs [5]. However, issues with range restrictions, charging infrastructure, the lack of knowledge by people about this electric ecosystem, and the lack of effective route planning solutions continue to slow the migration from conventional vehicles to EVs [11].

Route planning solutions' main objective is to provide drivers, with optimal routes that take into account factors like range, charging time, and energy consumption. The range of an electric vehicle is one of the most important factors that must be taken into account when planning a route [12]. Unlike conventional vehicles, which can be refueled in a matter of minutes at any gas station, EVs require access to charging stations to recharge their battery. The range of an EV determines the maximum distance that can be traveled before the battery needs to be recharged, and if the battery State of Charge (SoC) hits 0, there is no other good solution besides calling a trailer to get the car.

Weather, driving patterns, and the use of car features such as heated seats or air conditioning can all have an effect on an electric vehicle's range [13]. The efficiency and performance of a battery in an electric vehicle can also be affected by the temperature of the weather in general. If the weather temperature lowers, then the energy density of the battery may also lower while raising the internal resistance of the battery. It might call for the battery management system to heat up the battery to the optimum temperature of operation causing the SoC to lower faster than normal. Although most EVs are known for their instantaneous release of torque and power, a cold environment is certainly going to impact the overall responsiveness of the vehicle.

The driver has complete control over the heating system inside the vehicle, but driving in cold weather may be uncomfortable for the user, so it may need more frequent use of this system, resulting in the car requiring even more energy than usual. For this reason, it is crucial to take into account how weather conditions may affect an electric vehicle's range while making travel plans or while driving in bad weather.

Just like common cars (combustion engine), the driving style of an EV can affect its range [13]. Aggressive driving, high acceleration, hard braking, and high-speed driving, are all things that must be taken into consideration when trying to calculate the range of a vehicle. When an electric vehicle is driven at high speed, it requires more power from the battery to accelerate and maintain the speed. Sudden braking also wastes kinetic energy that could otherwise be used to recharge the battery through the regenerative braking system. [14] This is a feature commonly found in electric vehicles (EVs) that allows the vehicle to recover some of the energy that is normally lost during braking, by using the electric motor to slow down the vehicle and convert its kinetic energy into electrical energy, which is then stored in the battery for later use. When the driver brakes, the regenerative braking system engages, and the electric motor is used to slow down the vehicle. As the motor slows down the vehicle, it acts as a generator and converts the kinetic energy into electrical energy, which is then sent to the battery. Regenerative braking is particularly effective in traffic since we are always braking and accelerating or when driving downhill, where the vehicle is frequently braking. In these situations, the vehicle can recover a significant amount of energy that would otherwise be lost as heat during braking. The amount of energy that can be recovered through regenerative braking varies depending on the vehicle's design and the driving conditions. Some EVs allow the driver to adjust the regenerative braking settings to suit their driving style, while others automatically adjust the system based on driving conditions.

Another limitation of the range of an EV is battery degradation. Battery degradation is an inevitable process that occurs over time as the battery is charged and discharged. To minimize it, it is essential to maintain the battery properly, follow the manufacturer's recommendations for charging and discharging, and keep the battery at a consistent temperature range. In this case, this process has a significant impact on the vehicle's range and performance since the battery can no longer be charged to its maximum as it could when it was new [15].

In addition, every EV model has its own base range, depending on the size of the battery and the efficiency of the motor. This is why, it is essential to take into account the base range of the specific EV being used when planning a route. The base range of an electric vehicle is a critical factor for consumers when considering purchasing one. It refers to the distance a vehicle can travel on a single charge under ideal driving conditions [16].

Automakers use a variety of methods to calculate the base range of their electric vehicles. The most common approach is to conduct standardized tests under controlled conditions, such as the Worldwide Harmonized Light Vehicles Test Procedure (WLTP) or the New European Driving Cycle (NEDC). These tests simulate driving conditions and calculate the expected range based on a variety of factors, including the size of the battery, the weight of the vehicle, and the efficiency of the electric motor [17].

Automakers also use computational models to estimate the range of their electric vehicles. These models take into account a range of variables, such as the vehicle's speed, terrain, and weather conditions, and use mathematical formulas to calculate the expected range [18].

However, although we have a base range, it is important to note that the base range of an electric vehicle is just an estimate and can vary based on all the factors mentioned before.

In this section, I am going to name and talk about some already existing solutions that help users nowadays plan their trips while driving an electric vehicle. Some of them are:

- A Better Route Planner (ABRP) [19]
- ChargePoint [20]
- PlugShare [21]
- ChargeMap [22]

2.1.1 A Better Route Planner

The most known service used by EV users to plan their trips is called A Better Route Planner (ABRP) [19]. The solution developed by Iternio, allows EV drivers to plan their routes by simply adding their vehicle model and inserting the final destination. This service is already integrated with the electric vehicles' public charging infrastructure, which allows the planner to take into account the availability of charging stations along the route and plan for the necessary charging stops to complete the journey. Users can also customize their charging preferences, such as selecting a preferred charging network or type of charger. This feature is one of the most important ones since through that it is possible to show the user the number of stops that he needs to make to get to his final destination. 2.1

ABRP stores almost all EV models and uses that to provide customized route planning based on the vehicle's data, like its battery size and efficiency. With this data, the service can provide accurate range estimates and optimize the route based on the specific needs of each vehicle.

The service has unique features, for example, it allows users to customize their route preferences, like setting a custom average speed that the user will be traveling or choosing to avoid certain types of roads. The service also can take into account external factors, that are not controlled by the user itself, like the weather and the road elevation. With all of these features, users can have better control over the route planning process and it helps to achieve the most efficient and comfortable trip possible while reaching the destination with a good amount of battery.

Although ABRP has a lot of features nowadays, some disadvantages may cause some inaccuracy. Overall, ABRP has become a popular tool among EV drivers due to its ability to provide accurate and customized route planning that takes into account the unique characteristics of EVs. While there are other route planning solutions available for EVs, ABRP stands out due to its advanced features, user-friendly interface, and ability to integrate with various EV models and charging networks.

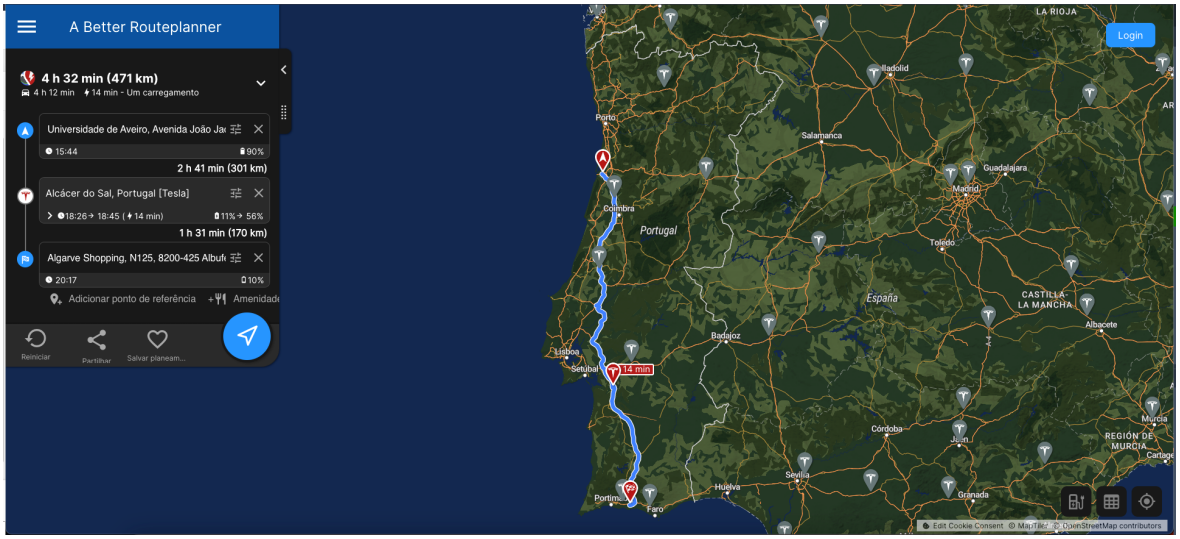

Figure 2.1: ABRP web application (Planning a route from University of Aveiro - Algarve)

2.1.2 ChargePoint

ChargePoint is a popular mobile application in Europe for electric vehicle users that provides a range of services in this area, including:

- Charging stations locations and reservation;
- Charging payments;
- Real-time data (availability and status of charging stations);
- Route planner;

ChargePoint's route planning solution is designed the same way as the ABRP, with one purpose only, to help EV drivers find the most convenient and efficient charging stations along their route, taking into account factors such as range, energy consumption, and charging time. From my research, the solution ChargePoint provides, uses a combination of algorithms and real-time data to provide accurate and up-to-date information about the location and availability of charging stations.

When comparing with ABRP, we can immediately see that one of the advantages of ChargePoint is that it provides users with a more reliable source of information about charging stations. Since ChargePoint integrates directly with charging networks and providers, they can provide users with real-time data about the availability and status of the charging stations, as well as information about the types of chargers and charging rates offered by each one. This factor can be a crucial one when EV drivers are choosing what service they are planning a route.

ChargePoint's route planning solution also offers other features, but these are not important for the range calculation itself, for example, being able to integrate with third-party mapping applications, such as Google Maps and Waze, and having a mobile application.

Despite all these features, there is a big disadvantage when comparing ChargePoint's solution with ABRP's. ABRP's primary focus is on providing the most optimized route planning (it is the only service they have), while the main ChargePoint focus is on providing a comprehensive network or charging stations for EV drivers, while still having a route planning solution. With this said, it is normal that ChargePoint's solution may not be as completed as the ABRP one.

Overall, ChargePoint's route planning solution is a useful tool for EV drivers who want to plan their routes based on the location of charging stations. While it may not be as comprehensive as other route planning solutions, such as ABRP, it is still a valuable resource for EV drivers who use the ChargePoint application for other purposes.

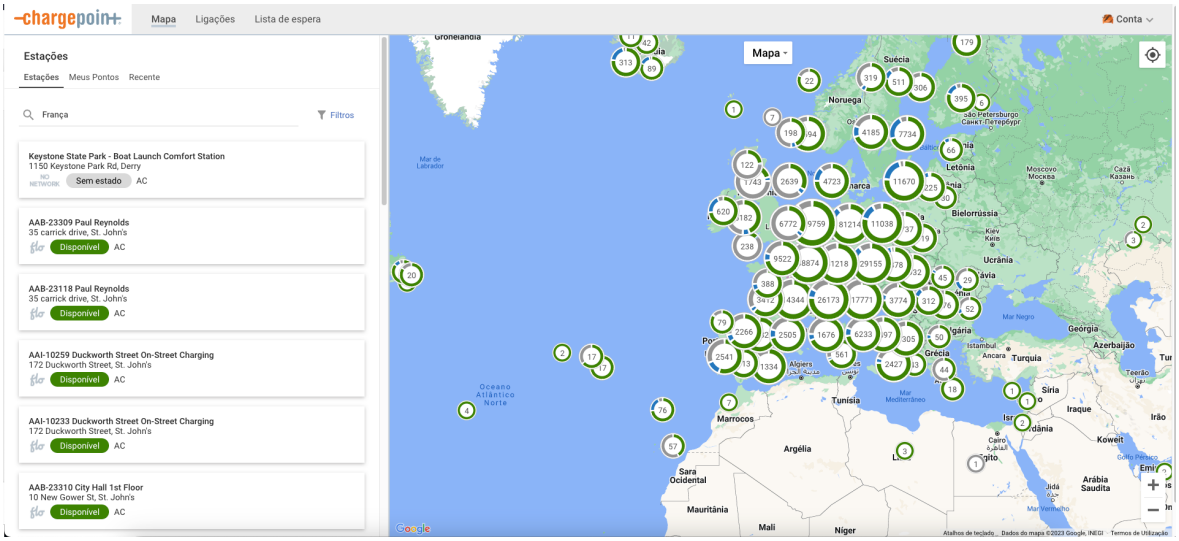

Figure 2.2: ChargePoint web application (Planning a route from Paris - Germany)

2.1.3 PlugShare

PlugShare is a mobile application and online service that provides charging stations around the world. The platform allows users to search for and locate charging stations, as well as view reviews and ratings from other users [21].

One of the advantages of PlugShare is its community-driven approach to collecting and sharing information about EV charging stations. The platform allows users to add and update information about charging stations, such as their location, type, and availability. This helps to ensure that the information on PlugShare is accurate and up-to-date. This can be a valuable resource for EV drivers who want to plan their route since it can provide real-time data, but, not only it needs user action to have this data, but the data can also be wrong or with errors.

PlugShare also offers a route planning feature that allows users to plan a route based on the location of charging stations. Users can input their starting point and destination, and the platform will provide a list of charging stations along the route that can be chosen by the user to recharge their vehicle's battery. The route planning feature also takes into account factors such as the vehicle's range and charging speed and can help users plan the most efficient route possible.

It is possible to use PlugShare without creating an account, but doing it can provide additional benefits and features.

Another advantage of PlugShare is its integration with various charging networks and providers across the world, having data on their directory of over 150000 charging stations. Although it has data from charging stations on 6 continents, PlugShare does not have data from all current charging stations in the world. This could be due to a variety of reasons, such as the availability of data from charging networks and providers, or agreements and contracts between PlugShare and those networks and providers [23].

PlugShare provides information about charging station locations and reviews from other users, but it does not offer a direct charging network integration, and with this, users are not able to charge their vehicles with this application. Nowadays, to charge on public charging stations, there is a need by the users to have an integrated payment solution on the application, so they can pay for their charging sessions, directly from their smartphone or other compatible device.

Both PlugShare and ChargePoint's main focus is to provide a comprehensive directory of charging stations, and this leads to the same problem as described before when comparing both solutions with ABRP. PlugShare does not offer the same level of customization or route optimization as ABRP, but it can be a valuable resource for EV drivers who want to find charging stations and plan their routes based on the location of those stations and the reviews from other users.

PlugShare can be a valuable resource for EV drivers who want to locate charging stations and plan their routes based on their location. Its community-driven approach, comprehensive directory, and integration with various charging networks make it a reliable and useful tool for EV drivers around the world, but, since the directory of charging stations data is completed with user information, it can sometimes lead to having wrong data or errors on the application.

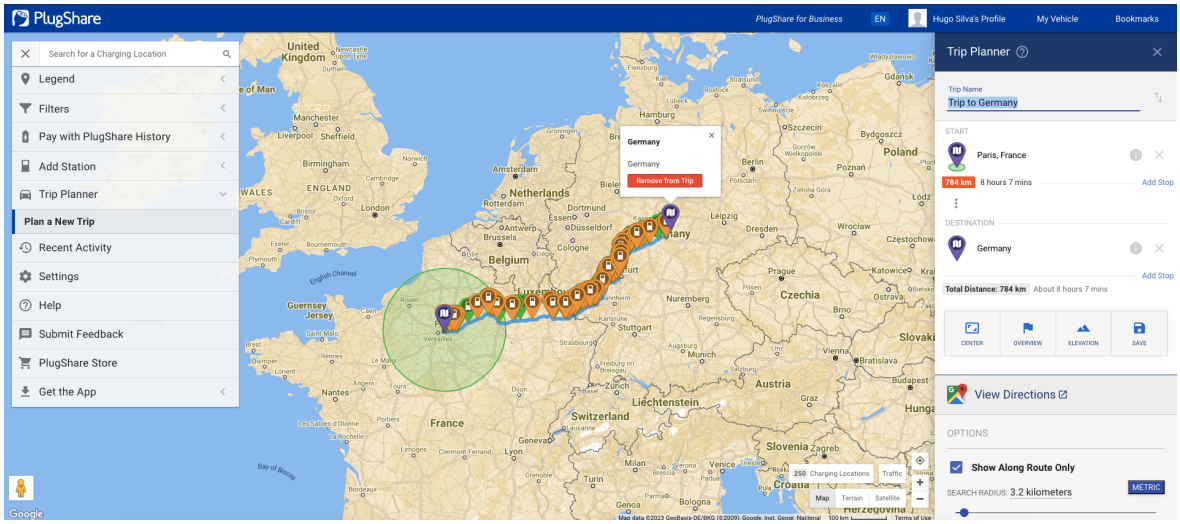

Figure 2.3: PlugShare web application (Planning a route from Paris - Germany)

2.1.4 ChargeMap

ChargeMap [22] is a community-driven platform that provides information and updates on charging stations for electric vehicles (EVs) across Europe. The platform was founded early in 2009 and has since grown to become one of the largest and most widely used charging station directories in Europe.

One of the main features of ChargeMap is its enormous database of charging stations from various networks and providers across Europe. Users can search for charging stations based on their location, EV model, and charging needs, and can view information such as charging speed, availability, and pricing for each station. The service integrates directly with the charging network [24] and has payments integration on the application. This allows users to charge their vehicle through its application while still being able to see the charging station's information, such as the price, location, charger types, and many others.

ChargeMap also includes route planning on their service. Users can plan their route based on normal factors such as their EV model and range. They simply can add their vehicle into the application, and then select the start and end location to calculate the route and the applications show the charging stations needed to stop, to complete the travel. Since they integrate directly with the charging network [24], they can optimize the route to include the most convenient and efficient charging stations along the way. The platform also provides real-time updates on charging station availability and pricing, making it easy for EV drivers to plan their route, avoid unexpected delays, and easily search for other alternatives if a specific charging station is occupied or with an error, for example.

Figure 2.4: Configuration on ChargeMap mobile application

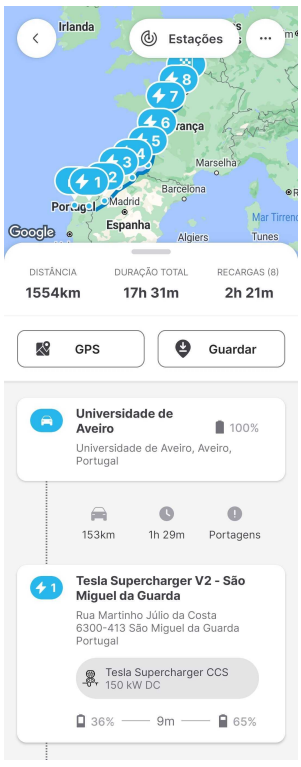

Figure 2.5: Route planned by ChargeMap

2.2 Limitations of existing solutions

Despite the advantages of existing route planning solutions, each one has its limitations. For example, while A Better Route Planner (ABRP) does take into account variables such as current weather, traffic, the availability and location of charging infrastructure, along with the vehicle's battery status and charging speed, some of the real-time data ABRP provides is reliant on user-reported making it susceptible to misinformation. If the reports from users are inaccurate or delayed, it could limit the accuracy of the route planner. However, not all route planning solutions have such capabilities. Some might have access to charging infrastructure data, but may not integrate it as effectively, or may not consider all the variables relevant to electric vehicles (EVs).

When examining the current solutions there are nowadays of route planning and charging solutions for electric vehicles (EVs), it is important to recognize the limitations that these may have. The following list highlights some of the key limitations associated with existing solutions:

- **Inaccurate or Outdated Information:** Existing solutions may suffer from inaccurate or outdated data on charging station availability, pricing, and status due to the dynamic nature of charging infrastructure [24], the limited resources for data updates (which includes user reported data), dependencies on charging station providers, and many others. This can lead to potential inconveniences and challenges either for EV drivers or for the route planer services to implement better solutions.
- **Limited Coverage:** Some solutions have limitations in terms of geographic coverage, with a focus on specific regions or countries. This results in limited availability and information for EV drivers in other areas of the world.
- **Lack of Integration:** Europe's charging infrastructure integration has a lot of issues since the charging networks have many different providers. This leads to inconsistencies in the user experience and restricted charging options since very often the EV driver needs multiple access cards or mobile apps to use charging stations from different networks. In Portugal, this problem slightly changes because every public charging station is connected to MOBI.E, as it is the most predominant network in the country [25].
- **Limited Customization:** Existing solutions may offer limited options for customizing route planning based on factors such as preferred charging networks, speeds, or vehicle parameters.
- **Reliance on User Input:** Solutions relying on user-generated data may be susceptible to potential inaccuracies or delays in information, as user-contributed data may not always be comprehensive or consistently reliable.

- **Lack of Real-time Updates:** Real-time updates on charging station availability and status can be limited in some solutions, leading to unexpected delays and inconveniences for EV drivers.
- **Charging Network Compatibility:** Some solutions may not provide information on specific charging networks or support certain charging connectors.

Understanding these limitations is crucial for identifying the areas that require further improvement and innovation in the development of more robust and comprehensive solutions for EV route planning and charging.

To address these limitations, this dissertation will include a microservices-based route planning solution, as it allows for the development of large and complex systems by breaking them down into smaller and independent services [26]. The microservices architecture can enable the development of a route planner that is specifically designed for EVs, accounting for their unique characteristics such as range limitations and energy consumption. By using this architecture, the proposed route planner can mainly offer scalability, reliability, efficiency, and simple integration with other useful existing services.

Overall, while existing route planning solutions for EVs have their strengths, they are not fully optimized for the unique needs of EV drivers.

CHAPTER 3

Methodology

3.1 Microservices architecture

The increased relevancy of microservices architecture in recent years has made it an ideal system for building intricate, reliable, scalable, and large software [4]. This includes route planning models suited to electric vehicles (EVs). This part examines the concept of microservices architecture and why it is important when developing a scalable and efficient system for EV routes.

3.1.1 Key Principles

A microservices architecture structures an app into independent and minor deployable services, each managing a separate function while using well-defined Application Programming Interfaces (APIs) as communication between themselves [27]

One major benefit of this format is its ability to promote modularity and scalability. The breakdown of monolithic applications into autonomous modules means quicker developmental cycles, easier maintenance activities, and specific component scaling flexibility by individual development groups [28].

In a business context, each required function is focused on by a given service enabling independent work from development groups who can optimize their code. This increases overall system agility without one service modification [27].

Well-defined APIs serve as a communication tool among services on a microservices architecture. The APIs create bidirectional standardized and independent data exchange between various modular services without updates or modifications affecting other services. As a result, simple testing, versioning, and scalability are achievable without having varying services impact each other's proper functioning [29].

Key principles such as well-defined APIs, scaling through component flexibility and modularity, and achieving robustness are all part of a microservices structure. Adopting this brings a lot of advantages to a development team and the system infrastructure.

3.1.2 Benefits

The incorporation of microservices in this dissertation enables faster and more continuous deployment practices. This is due to several key characteristics of microservices:

1. **Independent Operation:** Allows individual service operation, minimizing disruption to the system [27].
2. **Adaptability:** Microservices architecture is highly adaptable, which is effective for efficiently managing various changes [29].
3. **Scalability:** Resource efficiency can be enhanced by scaling services according to specific requirements [4], [26]
4. **Technology Diversity:** Different technologies (programming languages, databases, etc.) can be used for different services [29].
5. **Resilience:** If a service fails, the other services can continue to function [28].
6. **Ease of Understanding and Maintenance:** Each service is relatively small and fulfills a specific function, making the system easier to understand and maintain [29].
7. **Continuous Deployment and DevOps practices:** Services can be continuously updated and improved without causing system downtime or disruption [28].

These characteristics make microservices particularly suited for rapidly evolving and complex applications, for example, route planners for EVs [4].

Rolling updates can occur with minimal interruption to any other running operation in the system since every configured service operates independently. Such adaptive proves effectiveness at efficiently handling changes, and in a route planner context, it can be used on the charging infrastructure updates or the changes with the traffic [4].

Also, fault isolation characterizes another benefit related to microservices architecture implementations. Monolithic architectures expose systems to cascading sequence failures if one element fails, disrupting everything uniformly. On the other side, microservices operate individually. So, in a microservices architecture, a disruption or breakdown in one specified unit does not influence its indirect subordinated support (the other services). This effect encourages system monitoring and reliability enhancements through timely specific response mechanisms for quick attention to required due service.

Likewise, establishing inter-service communications ensures consistency across distributed services, but it needs to follow design and implementation standards since it is crucial in this type of architecture. It's how the individual services communicate and exchange information, enabling consistency and coordination throughout the service ecosystem when designed effectively.

3.1.3 Challenges and considerations

Several challenges arise regarding the complexity at multiple points in the system. This requires constant self-check to scout for potential channels affecting overall service performances. Robust and accurate mechanisms, such as effective logging protocols, can enhance fast issue identification and enable efficient data collection with real-time feedback capabilities [10].

Microservices architecture, while offering numerous advantages like referred before, also presents its own set of unique challenges and considerations:

1. **Data Consistency and Management:** Data consistency in a microservices architecture is more complex to maintain due to each service typically having its database, as opposed to a single shared one in a monolithic system.
2. **Inter-service Communication:** The communication between microservices can become increasingly complex with system evolution. Implementing synchronous protocols may cause mess and performance issues, while utilizing asynchronous protocols would add complexity to the system, making it harder to comprehend.
3. **Service Coordination:** With microservices, there is a need to coordinate multiple services, which could be developed by different teams and in different programming languages. Managing these services becomes a crucial task, and management platforms like Jira may be used to help manage this complexity.
4. **Deployment and Monitoring:** Sophisticated deployment, monitoring, and alerting services are must-have tools when deploying numerous microservices that take into account various environments since every single problem can be critical to the system. Besides the monitorization of the system as a whole, each service should also be monitored since this is vital to maintaining quality service standards.
5. **Security:** Deployable microservices require secure communication with the other internal services in the system. This necessitates the use of secure communication protocols and strategies to ensure the integrity and confidentiality of the data being exchanged. [30]

3.1.4 Design and implementation guidelines

It is important to consider some guidelines for design and implementation when building a microservices-based route planner for electric vehicles, ensuring its efficiency, scalability, and maintainability. In a general way, each implementation has its own use cases and these use cases can change the implementation guidelines and design. The following list represents standards and guidelines for implementation and design when dealing with a microservices architecture:

- **Define Clear Service Boundaries**: Each microservice should be designed around a specific business capability or function. In the context of a route planner for

electric vehicles, one might consider services such as Route Calculation, Charging Station Finder and Management, Application Management (including user and vehicle management), and State of Charge Management. Each of these services should have clear boundaries and should be independent of each other [30].

- **Use Domain-Driven Design (DDD)**: DDD is a method used in software engineering by which the design of the product is oriented around the problem, like for example, this dissertation is going to be used in the development of the route planner and its optimizations. It promotes the creation of applications that take into detail the practical concepts and processes [30].
- **Implement API-First Design:** Structuring the APIs of each microservice before beginning the implementation is one of the recommended practices when choosing this type of architecture. With this, we can consider how they all work together and discover any potential issues ahead of time [30].
- **Design for Failure**: In a microservices system, each component must be planned to manage malfunction efficiently. This can be achieved by introducing timeouts, circuit breakers, and backups in the services. Furthermore, services should strive for stateless configurations as much as possible, which makes them able to recover rapidly when catastrophes occur [30].
- **Use an Event-Driven Architecture as a pattern**: An event-driven architecture is a design paradigm that is used to facilitate communication between the services. Every time a service performs a task that may be useful on another service, it broadcasts an occurrence, which other services can enroll in if they wish to remain updated with the most recent data eliminating the necessity for a direct request. This creates an asynchronous communication between the services, which, when compared to traditional synchronous communication, reduces coupling since the sender is not blocked until the receiver responds [30].
- **Implement Continuous Integration/Continuous Delivery (CI/CD)**: Testing and deployment automation is vital for microservices architecture due to it having multiple services. CI/CD can secure quick, safe transitions from testing to production [28], [30].
- **Monitoring the services**: As discussed in the previous section, security is critical in a microservices architecture. Secure communication, authentication, authorization, API gateway security, and automated security testing are all aspects to consider. Some of those tools that can be used to monitor the system are Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana), Jaeger...[30].
- **Securing Microservices**: As already discussed in the previous section, security represents a critical role in a microservices architecture. Secure correspondence, ID validation, authorization, protection boundary security, and mechanized safety assessment are all things to take into account [30].

The aim when planning and developing this dissertation, was to follow the previous design and implementation guidelines as my time allowed. This type of planning provided a robust, scalable, and maintainable system that not only provided current needs but also simplified future expansion.

3.2 RESTful APIs

The rise in web services and web-based applications has created a need for efficient, scalable, and standardized communication between systems. Several architectural styles have emerged to serve this purpose in an influential manner and use Representational State Transfer Application Programming Interface (RESTful API)s to communicate.

A RESTful API is a kind of guideline and standard to create, read, update, and delete (CRUD) operations on resources using Hyper Text Transfer Protocol Secure (HTTPS) methods. The resource can be data or services that are found by a Uniform Resource Identifier (URI). Stateless architecture in the case of REST ensures that all the client information required to interpret and process a request will have to be sent together in every single request.

In our system, both internal and external services communicate predominantly via APIs, ensuring a uniform, efficient, and scalable interaction model.

Since our system was designed to use and to be integrated into a microservices architecture, RESTful APIs are even more relevant in this context. In this case, there is a synergy between microservices and RESTful APIs since the last one provides communication in a standardized way of the services regardless of their internal implementation. Both internal and external services mostly communicate through APIs in our system so a consistent, efficient, and scalable interaction model is ensured.

One of the notable applications of this approach in our system is the internal communication between the backend and frontend. Our frontend, which provides the user interface and user experience, constantly interacts with the backend to fetch, display, and manipulate data. This interaction is facilitated through RESTful APIs, ensuring that data is transferred in a structured, consistent, and reliable manner.

In summary, RESTful APIs are the heart of communication within our system. Its standardization coupled with being stateless and resource-oriented ensures that our microservices-based architecture works together to provide an improved user experience as well as efficiency within the system.

3.3 Charging Stops Strategies

The key aspect that distinguishes the experience of driving on a long trip an EV from conventional fuel ones is where and when to charge. For an EV route planner, this decision plays a critical role in ensuring smooth trips, minimal delays, and enhancing user satisfaction. The strategy used for making these decisions can vary and each has its unique benefits and challenges that are associated with it. In the context of this project, we primarily used the proactive strategy but it is also important to understand other available options.

In this section we will talk about the strategies researched and analyzed in this dissertation, giving a small overview of each of them and referring to their benefits and disadvantages of using each one of them.

3.3.1 Proactive Strategy

The proactive strategy is primarily concerned with anticipating charging needs before they become critical. Leaning on this approach, the route planner takes advantage of the scheduled stops based on projected battery depletion such that it is always within a comfortable range of a charging station before the battery level becomes critically low. [31]

The following diagram provides a visualization of this strategy in our system. It showcases a journey from a start to an end point, with three planned stop points that are pre-calculated according to the range of the vehicle and other criteria that will be explained in detail further on. Each of these stops represents an estimated SoC of 30% at these points according to our pre-calculations. At that point, following this strategy, it is recommended to charge, so, we start looking for charging stations around the vehicle at the given location.

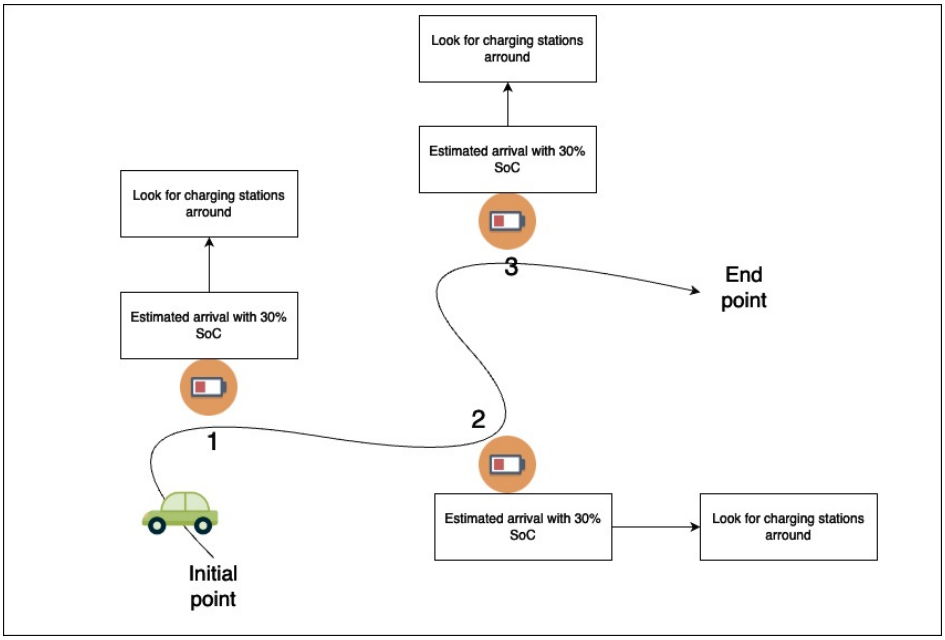

Figure 3.1: Representation of the proactive strategy with 30% SoC estimated stops

This visualization highlights the essence of the proactive strategy, focusing on pre-calculated charging stops to make sure the EV's battery remains within optimal thresholds on the journey.

Benefits:

- **Safety:** Avoiding very low battery levels reduces the risk of being in locations without a nearby charging station. This approach greatly reduces 'range anxiety' as drivers are kept constantly aware of their next charging stop with a considerable amount of SoC left. [23]
- **Optimized Charging:** By stopping to charge at predetermined SoC levels, the battery is kept in an optimal state. Lithium-ion batteries, commonly used in EVs, often have faster charging rates when they're not too empty or too full. [23]
- **Car Battery Health:** Recharging in pre-calculated stops, the car battery does not get very low and prevents it from degrading its health over time.
- **Cost Efficiency:** Some charging stations have variable pricing. By planning ahead, drivers can choose to charge at times or locations where prices are lower.

Disadvantages:

- **Overcaution:** This approach might suggest more frequent charges than necessary, potentially resulting in longer overall journey times.
- **Reliance on Predictions**: Proactive strategies depend on accurate predictions of energy consumption, route conditions, and charging station availability at a given time. Wrong predictions and inaccuracies can lead to unnecessary charging stops. [31]
- **Flexibility**: If something unexpected happens throughout the route with the vehicle SoC, the driver may need to recalculate the route again according to what happened.

We can conclude that this strategy has a lot of benefits when compared with its disadvantages. The consequences of these disadvantages result in more charging stops, which may not be a bad consequence overall if we think about safety and range anxiety.

3.3.2 Reactive Strategy

A reactive strategy in EV route planning is a strategy where charging decisions are based on the needs of the vehicle and other conditions such as weather, traffic patterns, or driving style in real-time. Unlike proactive strategies that are calculated with pre-planned stops, the reactive method focuses on calculating the route and deciding where to stop in real-time throughout the route. It allows the vehicle for example to dynamically adapt to changing scenarios like sudden changes in available charging infrastructure due to traffic congestion or unexpected battery consumption rates. This approach is very flexible since it deals with real-time data and changes, and works well for spontaneous trips since it does not require extensive previous planning with human interaction. However, since it uses the current available data at the time and this type of data may sometimes contain wrong information, this strategy must not compromise the range or convenience of the driver of the vehicle. [31]

To apply this strategy in an EV route planner, several factors should be taken into account since this method will not look for charging stations when a certain SoC is reached

like the proactive strategy. With this said, the following list represents some of the factors that should be taken into account when using a reactive strategy in an EV route planner:

- **Density areas:** In denser areas, it is easier to look for appropriate and available charging stations, since they usually offer more charging stations.
- **SoC & Safety Buffer:** Maintaining a minimum buffer, such as 20-25%, where we immediately should start to look for a charging station if the SoC reaches there is a safety approach. However, this value should also be dynamic according to not only the current area that the user is in but also the next area he is going to, since in denser areas like in the center of cities it is way easier to find an appropriate and available charging station than in less dense areas like outside cities. [31]
- **Driving Conditions:** Conditions like vehicle speed, road, and weather can influence the battery's consumption.
- **Future Route:** By knowing the full route and where the user is going next, we can adapt the search of charging stations according if we have fewer or more charging stations in the next "step" of the route. For example, if in the next 50 kilometers there are fewer charging stations than in the current location, then, a charging station should be searched at the current location so it is provided more charging options, which normally means better charging station characteristics like charging speed and also more availability.
- **User Preferences:** All drivers have their level of risk when talking about "range anxiety". Some prefer to charge only until the SoC is almost zero, while others like to charge the car when the SoC is almost full.
- **Availability, Speed, and Compatibility of Charging Stations:** As we have already seen before, availability, speed, and compatibility of the charging stations are crucial factors to take into account when deciding when to charge.

Figure 3.2 presents a hypothetical visualization of the reactive strategy in action over the course of a journey with a start and ending point, some charging stations represented in the route by numbers, and a loading symbol that indicates the real-time calculations and decisions made by the strategy throughout the route.

At Point 1, the EV SoC is almost full (e.g.: 75%), and this normally would mean that the vehicle can continue its journey without any problems. However, since we are dealing with a reactive strategy and due to the good availability, compatibility, and high charging speed of the charging station at point 1, the strategy might decide to initiate a short charging session depending on the future route, in this case, represented by Point 2. As we can see in Point 2, although the charging station is compatible and available, it has a low charging speed, one of the crucial criteria that should be taken into account when deciding which charging station to pick. For this case, the strategy would choose to initiate a charging session at point 1, since in nearby future routes there are no optimal charging stations.

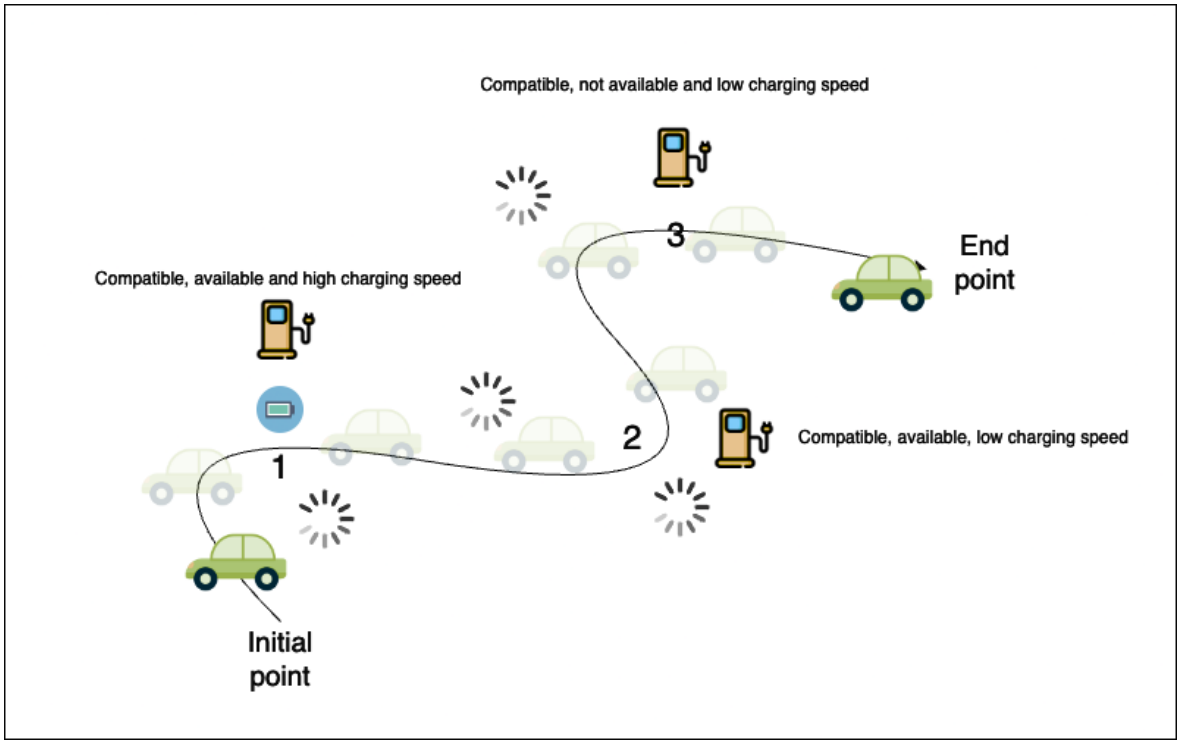

Figure 3.2: Representation of the Reactive Charging Strategy in the context of a EV Route Planner

As the EV continues its journey, it reaches Point 2. Here, as already referenced before the charging station has a lower charging speed, which can be a problem for most EV drivers, since a full charge can take hours to finish. At this point, the strategy should search for future charging stations throughout the route that can be optimal for charging, instead of choosing to charge in a low charging speed station.

By taking a look at the charging station represented by Point 3, we can check that although it is compatible, this charging station is not available and has a low charging speed. The availability of a charging station is represented by whether all charging ports are being used by another user or if the charging station is really out of service. At this point, we have a dilemma since although the charging station might not be optimal for charging, the user might be forced to wait for its availability (if it is being used by another EV) or start looking for others solutions if the vehicle range and SoC at this point is really low.

This image can provide a clear and understandable step-by-step on how the strategy works, but it does not represent every single situation that might happen throughout the calculations, decisions, and route since there are a lot of criteria being taken into account when calculating the whole route and deciding when and where to charge. Since this strategy deals with real-time data, even in the same route, different results and situations can happen over time.

Benefits:

- **Adaptability & Flexibility:** A reactive strategy can react to sudden changes in the route conditions, like traffic that suddenly appears, changes in weather, or any other changes in the road that could affect the route or the vehicle. [31]
- **Optimized Charging Stops:** By making real-time assessments, a reactive strategy can identify and prioritize charging stations that are both available and offer faster charging speeds, reducing total trip time.
- **Efficient Resource Usage:** By considering the current SoC of the vehicle in real-time and the conditions ahead, the strategy can make efficient use of the SoC and the charging resources on the route, avoiding this way unnecessary charging stops.
- **Dynamic Prices:** If the strategy is integrated with real-time dynamic prices data, it can lead drivers to charging stations that offer a cheaper price at a given time, however, not every charging station has dynamic prices since it depends on factors like the country, the network and the operators.

Disadvantages:

- **Risk of draining the battery:** Inaccurate information or error in the real data like inaccurate availability of a charging station can misdirect the EV driver to a location where he cannot charge his vehicle. This may lead to the vehicle running out of charge before reaching another viable charging point and leaving him without any SoC left. Such a risk emphasizes how accurate the data needs to be for a reactive strategy to be effective.
- **Increased Complexity and Computational Demands:** Real-time data analysis and decision-making may often introduce significant computational complexity in the system which might slow down response times.
- **Reliability on real-time data:** The success of a reactive strategy depends on constant access to real-time data, however, this is not always possible since it depends on the remote connection with not only the charging stations but also on all the criteria related before such as weather, traffic, vehicle SoC and all of its respective quality data. Also, any interruption in the connection, whether due to network problems or remote areas with poor coverage, can affect the performance of the strategy since it does not have any pre-calculated route like the proactive strategy. Even if the strategy chooses to deal with user feedback data, it not only can lead to returned wrong data from the user, but also the data might be outdated or even nonexistent since not every single user gives feedback when passing by or charging on a charging station. [31]
- **Frequent Changes:** If the strategy continuously reacts to the latest data, it may sometimes lead to frequent changes in the route which can be a problem for some drivers since this gives a sense of unpredictability.

3.3.3 Hybrid Strategy

The hybrid strategy is also a strategy that can be used to decide which charging stops and stations to pick in our route, by using a mix of both proactive and reactive strategies discussed before. It aims to maximize the route and charging decisions by integrating and merging the strengths of both strategies while nullifying the potential drawbacks inherent in each strategy. The strategy contains initial planning like in the proactive strategy but also contains real-time adjustments like in the reactive strategy, which makes it a mix of both approaches. [32]

In a hybrid strategy, at the beginning of the journey, based on known parameters (distance, charging stations, initial SoC, etc.), the system acts just like the proactive approach to suggest an optimal route and potential charging stops. This normally provides the EV driver a sense of comfort since he at least has an initial plan. As the journey goes on, the SoC starts dropping, and the system starts making requests in order to get real-time information about all the criteria that affect the range of an EV. After having the data, it starts to make adjustments in the route if possible. For instance, if the charging station initially planned becomes unavailable or a faster one comes online in the surroundings, the system would suggest a small change in the plan accordingly.

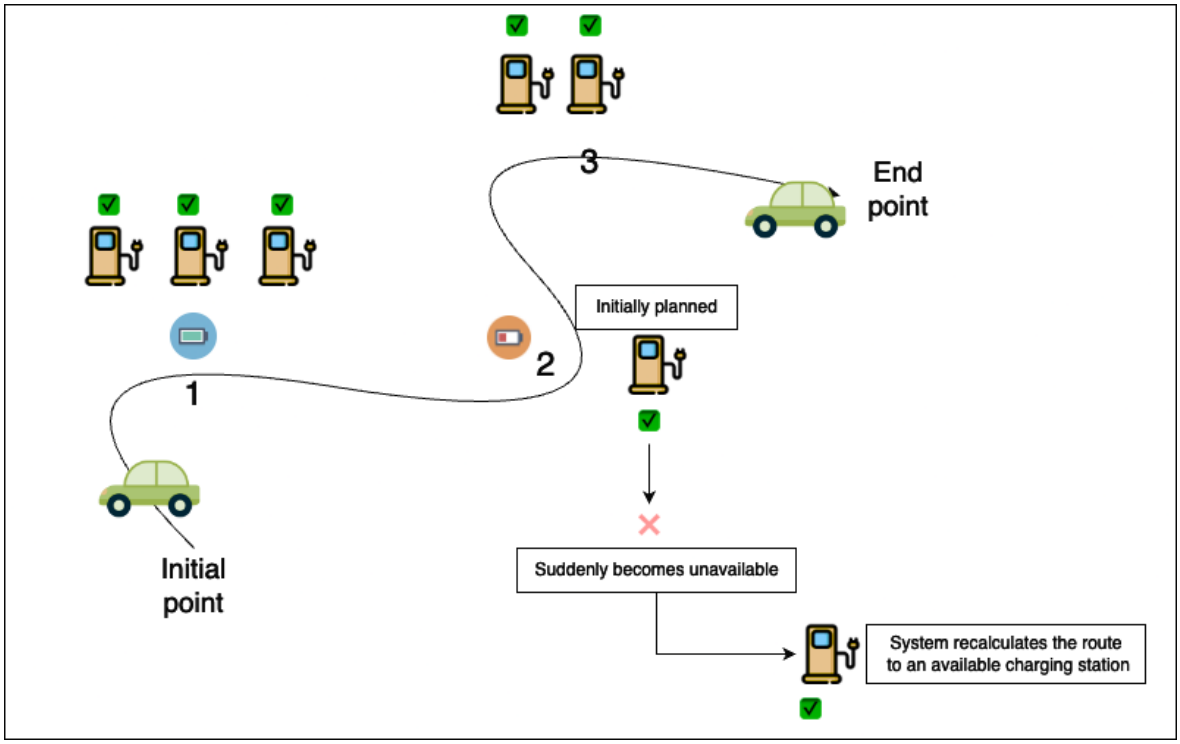

Figure 3.3: Representation of the Hybrid Strategy in the context of an EV Route Planner

Figure 3.3 shows a hypothetical representation of the hybrid charging strategy during an EV's journey from the start point to its destination. In the route we can see three critical points, marked as 1, 2, and 3, each indicating a place with charging stations just like the previous images representing the proactive and reactive strategies.

Since we are in a hybrid strategy, the entire journey starts with calculating an initial plan that is going to be used throughout the route. [32] After having a pre-calculated route, the EV user is now ready to depart at the starting point represented by the image.

As the EV proceeds on its route, it reaches Point 1 where it bypasses the charging station since it has a large amount of SoC left, and on the pre-calculated route, we are assuming that the first charge is going to be made on Point 2. Also, this charge is going to be skipped since the strategy did not find any critical sudden change on the route which could lead to a potential early charge.

At Point 2, the amount of power left in the vehicle's battery has dropped substantially, and it is close to a threshold that needs charging. Originally, the pre-calculated plan located a suitable charging station near Point 2, however, due to availability circumstances, the charging station is no longer available to charge. This is where the reactive part of the hybrid strategy kicks in. Instead of leaving the driver stranded or scrabbling around for solutions, the system recalculates immediately and finds another viable one nearby.

After charging, the EV proceeds on its journey towards Point 3 and then to the final destination without needing a second charge.

This simple hypothetical representation shows that with a hybrid strategy, we can have flexibility and adaptability but just like the reactive strategy, the image does not cover all of the numerous situations that can happen.

Benefits:

- **Balanced Approach:** By combining the strengths of both the proactive and reactive strategies, a hybrid approach can find a middle ground between long-term route planning and real-time adjustments. It makes sure that the decision-making is not too rigid nor too volatile. [32]
- **Safety:** The initial planning phase can offer drivers a sense of safety about their journey since they initially know when and where they will charge and with that adapt or customize their route according to their preferences. Also, the reactive adjustments can deal with unforeseen circumstances to make sure that drivers don't run out of charge.
- **Optimized stops & Energy usage:** This strategy combined can be very efficient since it already has an optimal pre-planning, and if something happens on the journey, the route is recalculated according to the unexpected change that occurred. For this reason, this strategy can potentially lead to more efficient use of energy considering that the pre-planned charging stops are optimal and by taking into account factors such as terrain and the driving style in real-time.

Disadvantages:

- **Complexity:** This strategy can introduce significant complexities in terms of algorithm solutions, development, implementation, testing, and scaling. Logically, the higher the level of complexity introduced by combining both strategies, the greater will be the cost of development and maintenance of the whole system. Since it uses real-time data and we also need the current information about the EV at almost all the points on its route, continuous connectivity and a reliable infrastructure are needed to ensure accurate decision-making throughout the route. [32]
- **Conflicting decisions:** There might exist situations where the reactive components conflict with the already optimal suggestion from the proactive component.
- **Higher response times:** Continuously merging data from proactive and reactive sources might introduce computational overhead, which can potentially affect the system response times.

3.3.4 Evaluation and selection

With all the strategies analyzed, we concluded that the best solution for a route planner system would be a hybrid strategy. A hybrid approach has the benefits of both proactive and reactive strategies with huge potential for optimizing efficiency and user experience for EV route planning since it deals with data in real-time.

However, as we already have seen before, the hybrid strategy has some issues and challenges. It requires a big infrastructure to deal with the real-time connections between the system and the user, constant and complete real-time data which also requires connecting with internal or external services, connection with the charging stations that may require a contract with their operator, and many more complexities. Due to the current limitations of this dissertation, and the resources available for it, it is not possible to integrate and carry out this hybrid strategy's reactive aspect.

With this, the proactive strategy proved to be the better choice for the scope of this work. It might not have the complete level of adaptability that a hybrid strategy could offer, but it is a well-planned strategy, systematic, effective, and can return good results for EVs users. The way to prove this is by looking to the ABRP service that was already referenced before in this dissertation, because their system also uses a proactive strategy, and it is the current most used EV route planner in the world.

Selecting charging stations is a crucial step in the developing process of any route planner for electric vehicles. Choosing where the final user should stop to charge requires using a well-defined approach since it can draw many consequences if chosen wrong, such as not having enough battery to reach another charging station because the one chosen before was out of service. Several different decision-making algorithms could be applied to the selection of charging stations but many of them may not be optimal for our context. In this section, we will take a look at some potential algorithms and methods that were researched and investigated, to pick what was the one best suitable for our project.

3.4.1 Weighted Scoring Method

The Weighted Scoring Method (WSM) is a decision-making tool that evaluates options based on multiple criteria. By assigning specified weights for each criterion, this technique provides an ordered way of ranking or scoring choices, from the best to worst. After the options are scored according to the performance across the weighted criteria, typically, the option that is ranked highest based on the final cumulative score is selected as the best of all available choices.

In the context of our route planner, we can assume criteria options such as charging station availability, distance, elevation, number of chargers, charging speed, and many others. This methodology offers a clear method for making a choice according to the criteria given by the developer. However, it should also be noted that priorities can change over time, over the route, and even the final user might prefer having few but long stops or having short but many stops. With this said, the assigned weights might need to be changed and revised over time, and be customizable to the final user according to its preferences.

The following table (3.1) shows an example of possible assigned weights to each of the most pertinent features based on EV drivers' expectations when planning a route. These weights were determined after considering the basic demands and expectations of EV drivers, that is reaching the final route with a considerable amount of SoC.

Table 3.1: Weighted Scoring Method for EV Route Planner Features

Criteria	Weight (%)	CS 1 (Score)	CS 2 (Score)	CS 3 (Score)
Distance	45	9	8	7
Availability	35	8	7	9
Charging speed	5	6	5	6
Elevation	10	6	7	8
Traffic	5	8	7	6
Weighted Total	100	7.85	7.45	7.7

Note: CS stands for Charging Station.

In Table 3.1, the three hypothetical charging stations (CS 1, CS 2, and CS 3) were presented to explain how the scoring might go in a development decision scenario. The criteria cover some important elements like the distance between the user and the charging station in the moment of looking for a charging station to charge when calculating the route, availability of the charging station, its charging speed (if available), the elevation of the road to the charging station and the traffic.

All three presented charging stations have their strengths, for example, CS 3 has a 9 out of 10 score on availability criteria, however, it is not the highest weighted total, since it has the worst score on the others criteria. CS 1 beat out with the highest Weighted Total of 7.85 out of 10, and for this reason, a system using this decision method would probably recommend this charging station to the EV driver among these three options given these weights and scores.

Distance and availability have the highest weightings since their importance in the decision-making process for EV drivers is higher when compared to the other criteria. For example, although charging speed is essential and also important it doesn't carry as much weight as the distance between the driver and the charging station, which can directly impact the range calculation. Normally, drivers might prefer to reach a nearby available station with a considerable amount of SoC left, over a slightly faster charge, however, it depends on the user profile.

This method offers a clear approach to the decision-making process of picking a charging station when deciding to charge, but since it is also flexible it needs to be adjusted according to the driver requirements. Also, as new data becomes available, the weights and scores can and should be adjusted accordingly to ensure the most accurate and useful recommendations.

3.4.2 Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) is a structured methodology developed especially for complex decision-making. Developed in the 1970s by Thomas L. Saaty, [33] it breaks down a large problem into a series of simple comparisons allowing qualitative and quantitative aspects to be considered. This method uses a hierarchical structure of objectives, criteria, sub-criteria, and alternatives for the objects that are chosen to decide. It is often used when decisions involve conflicting criteria which is the case for our route planner, however, it heavily depends on human intervention, which makes it a bad option in the context of our system. Although we started to research and investigate this method, as soon as we checked that it relied on user feedback, we almost immediately discarded this method as an option for our decision-making process.

3.4.3 Dijkstra's Algorithm

In 1956, Edsger W. Dijkstra created the famous graph algorithm, known as Dijkstra's Algorithm. The algorithm finds the shortest path to a destination among a set of nodes in a graph which can be cities, waypoints, or even in our EV route planner's context, charging stations. It works by recursively selecting the node with the smallest known distance from the start node and then it examines its neighbors to update their shortest distances until one reaches the end location or all nodes have been considered by the algorithm. [34]

Regarding electric vehicle route planning, Dijkstra's Algorithm is an algorithm that can be applied to calculate the shortest path between a user and nearby charging stations. Since this algorithm uses graphs, it is possible to give a weight for each edge, combining criteria such as distance, charging speed, availability of the charging station, and many others to make the algorithm more flexible and robust. However, such incorporation of weights in the graph means we are moving away from the purpose of standard Dijkstra's algorithm.

Dijkstra's is a proven algorithm, known for its efficiency in finding the shortest path, and it can be adapted to consider other constraints, making it possible for a decision method for our EV route planner. However, the algorithm has some limitations since it was not designed to take into account other important factors such as the charging station availability, charging speed, and many others. For this reason, we also decided to discard this algorithm.

3.4.4 Greedy Algorithm

The Greedy Algorithm is a generic algorithmic paradigm that proceeds after the problem-solving heuristic making locally optimal choices at each stage with the hope of finding the global optimum. In other words, it makes its best decision from its current position without concern about other implications this decision might have for future steps. For many problems, the Greedy Algorithm has an optimal solution but there are scenarios where it can result in suboptimal solutions.

This could be a straightforward solution to our problem. At each decision point in our route planner (when the EV needs charging), the algorithm simply chooses the nearest charging station without much consideration for future implications. Simple and easy to implement but it might not always give an optimized route because of discarding others' charging points and other important factors that need to be taken into account such as charging speed or availability. For this reason, we immediately discarded this option as well.

3.4.5 Machine Learning

Machine Learning (ML) is a part of artificial intelligence, that uses existing algorithms and statistical models to allow computers to do tasks without having explicit programming. In essence, ML models are trained to recognize patterns and autonomously make a decision based on the data provided to the model. With more data available over time, the model can learn and improve, and with this, it can be particularly effective for tasks that require complex decision-making. [35]

Different methodologies exist in the ML field, and this main field can be divided into many subfields, each one serving particular kinds of tasks and data structures. From the structured decision-making of decision trees to the continuous feedback mechanism of reinforcement learning, ML provides a lot of techniques and methodologies that should be analyzed according to the different needs and complexities of each system. [36]

In EV route planning the use of ML can be applied in multiple ways, but it needs the right methodology, tools, and data to create an optimal solution. For example, regression models can predict the expected usage of a certain charging station at any given time, while decision trees can be used to classify charging stations based on their fitting for a specific route. Deep learning can combine multiple features, such as distance between the user and the charging station, charging speed, and availability, providing this way the best possible charging options. Also, reinforcement learning can be used in a more robust and dynamic approach to the system, since it continuously learns and refines its strategy over time, while still choosing the best charging stations throughout the route.

There are many benefits of using ML models in an EV route planner system, but the main advantage is its adaptability. Unlike conventional algorithms, ML models can evolve with time and adapt towards changing patterns, user behaviors, or even the charging infrastructure development. Such adaptability makes it possible for these models to discover patterns from data that might be missed by other methods. However, such success depends heavily on the availability of high-quality data. Otherwise, without it, models might suffer from overfitting, a term used to describe when a model performs really well on the training data but performs poorly on new unseen data. Since the EV field and all of the ecosystems around it are in constant evolution, this disadvantage can be a huge problem. Another challenge is interpretability, particularly when using neural networks like deep learning models, it can be hard to understand what is the reason behind a certain choice, and this can be a problem either for developers or for the final user when a bad decision is made by the system. [35]

To conclude, the potential for ML to increase the efficiency of EV's route planning is quite evident. It offers the ability to evolve and adapt to the continuously changing landscape of electric vehicles and the ecosystem around them. However, a lot of work is needed when choosing ML as a solution to our decision-making algorithm since to choose it, we need a clear understanding of all the prerequisites needed to succeed and the risks of it. The success of a ML model heavily relies on the quality and quantity of data available, but, when there is a scenario where the data is limited and not easy to get, simpler methodologies like decision trees are often a better choice.

Compared to more complex models, decision trees offer transparency, easy interpretation, and less susceptibility to overfitting. [36] For this reason, for effective integration of ML in EV route planning, decision trees are a balanced choice, offering clarity, and adaptability and ensuring that the benefits of ML are harnessed without compromising reliability.

3.4.6 Decision-making Methodology: A Summary

Two main approaches were clear in the search for decision-making methodologies appropriate for EV route planning:

- Weighted Scoring Method
- Machine Learning

WSM provides a structured and straightforward method of rating and ranking the charging stations based on some pre-defined criteria with weights. With the weights defined, we can easily decide on which charging station to choose.

Machine learning can be flexible and can find different patterns in the data which also makes it a viable option, however, it depends on the quality and quantity of input data trained on it. While there are many advantages with ML, it was really hard throughout our whole research to find any open-source dataset that contains enough quality data to train a model. For that reason, and specifically for this dissertation, we decided to not choose this methodology as the decision-making algorithm for picking the charging stations in the route.

With that, the Weighted Scoring Method became our final choice. For getting charging station data, we used OpenChargeMap API [37], which is a completely open-source platform that gives details for many charging stations. Integration and usage of this service will be covered in the next chapter of this dissertation.

CHAPTER 4

Implementation

4.1 Overview

Here, we will discuss how we achieved the practical aspects of our dissertation by describing the existing methods and technologies used to build an EV route planner based on a microservices architecture.

First, we discuss the overall system architecture as well as the justification behind choosing a microservices-based approach and every technology used. The core functionality of our system relies on a proactive strategy that is used to search for charging stations when a vehicle gets below 30 % of SoC. This strategy will be explained in more detail further on. Our system uses external services like OpenChargeMap (OCM) API to search for nearby charging stations within a radius that can expand according to the results. We filter the results according to car connector type and fast-charging EV connectors. Although every charging station can be searched by our system, it prioritizes the ones that have fast charging (>50 kilowatt (kW)).

Subsequently, we briefly describe some of the main technologies that were used to make our solution possible. These include tools made in Python 3, and databases, among others combined using external services like Open Source Routing Machine (OSRM) for route planning. Lastly, we used Folium for interactive mapping while Streamlit helped with the creation of our interface.

All things considered, we think the practical implementation of our theoretical constructs discussed here contributes significantly to the broader discourse on EV logistics and infrastructure. Through technical combinations and employing more technology than usual, we can deliver a starting point for proper solutions in EV route planning and charging station locating.

4.2 System Architecture

The architecture developed and designed for this dissertation uses a microservices approach, having completely functional components responsible for specific tasks that add up to efficient development, testing, and scaling. In this current version, the system only contains two main services, where our main one is the route planner service (backend) as the central hub where various connections are made externally to APIs and internally to our frontend service. In the future, we intend to integrate more services to optimize our existing system.

The system uses external services via custom-made wrappers in Python [38]. These include the Open Source Routing Machine (OSRM) for route planning and OpenChargeMap for locating nearby charging stations. Each of these external services interacts directly with our backend service.

Internally in our system, there are numerous classes in Python that are responsible for each designed task, such as filtering and selecting charging stations, calculating routes, recalculating routes, decision-making methods, SoC management, and many more. Each of these classes works with the others to be able to build an efficient route planner system.

The communication between our internal repositories (backend and frontend) is made through HTTPS requests, using FastAPI [39] to expose an endpoint that accepts varied inputs required for route calculation such as vehicle model, brand, year, start location, end location, and the initial SoC.

MongoDB [40] was the choice for data persistence, storing the necessary across our different services.

The overall architecture of the system explained above is represented by figure 4.1. The diagram shows how different parts and services from our system interact with each other. The final user interacts directly with the Frontend Service which in turn communicates with the Backend Service, labeled as "Route Planner Service". The returned data from the backend service is possible due to all of the internal management and services in the system and to all of the connections to several external APIs like OSRM and OCM.

In conclusion, our application integrates many technologies and methodologies in the best way to face the primary challenges of electric vehicle route planning and charging station location. Having a microservices architecture, the system is flexible and scalable, being ready for all future upgrades and requirements.

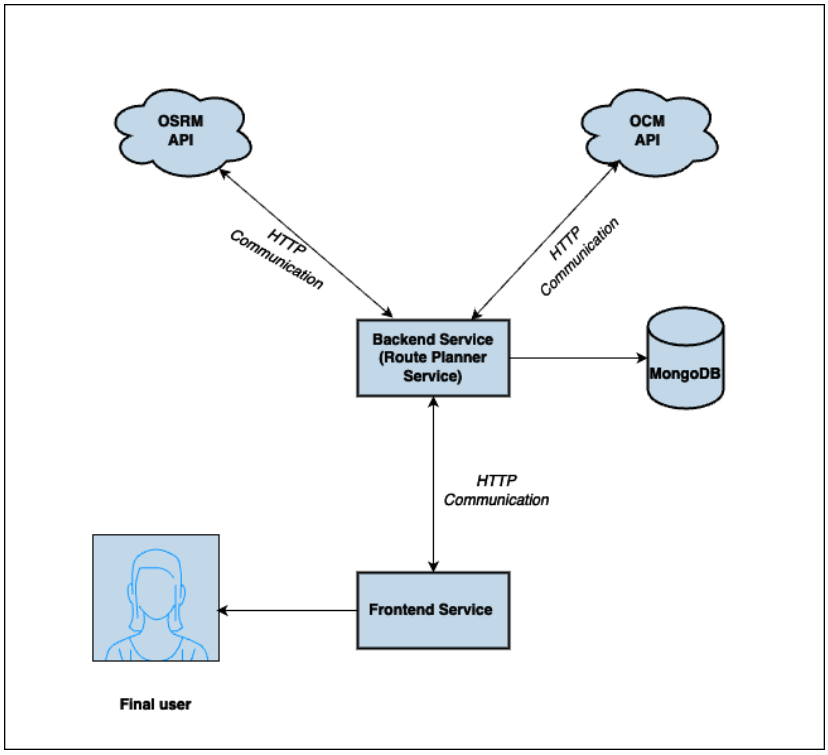

Figure 4.1: Ilustration of the System Architecture

4.3 External Services

The electric vehicle route planner implementation was only possible due to the usage of multiple external services. Building a route planner from scratch is not an easy task, and without using these external services, the development of this system may have not been possible, since the amount of effort to build everything from zero is enormous. The services have important roles in providing certain features that are crucial to the entirety of the system's functionalities. Every single external service used in the practical part of this dissertation will be detailed and explained in the following sections. It will include a general overview of each service, its benefits in using it on the system, its implementation, and the challenges associated with it.

We also incorporated good software practices in the development and integration of our application. Particular emphasis was on using API wrappers to ensure the robustness, maintainability, and scalability of our system.

For every external service used in this dissertation, it was easier to add its own functionality to our system by wrapping them in customized Python classes.

All the customized wrappers simplified the interaction with each external API. Additionally, with the usage of this practice, we were able to reduce the potential for errors that might occur since we minimized unnecessary coding and requests, or allowed different APIs to be used within applications. Also, even if an error occurred, we could easily detect it and start the debugging from that part of the wrapper.

4.3.1 OSRM

Overview

OSRM [41] is an extremely fast routing engine built with Open Street Map (OSM) [42] data. It offers directions from one point to another worldwide and can calculate long routes quickly for any type of vehicle travel.

Built around high scalability, the system can handle requests either for small areas like cities or larger regions like countries. It uses complex algorithms like contraction hierarchies to offer fast and accurate route calculation. [43]

Although it is a recent technology, OSRM already provides an open API making it easy for developers to integrate its features into their apps or services. This service offers integration with calculating routes, mapping GPS traces, and calculating distance tables through an API. In this dissertation, the integration only included the route calculation feature, which is necessary for calculating the routes between the starting point, and the location of the vehicle alongside the route, charging points, and the final destination.

Another great benefit of using OSRM is that it is an open-source solution, meaning that the source code is freely available for anyone to read, change, or even make suggestions. There's a large community of active development surrounding the tool, which makes it better and more optimized every day. It is used worldwide by different organizations in various mapping and routing services, such as Geographic Information Systems (GIS) companies. [44]

Integration with the system

The OSRM [41] is one of the main services used for the route planning system developed for this dissertation. OSRM was selected as the service to calculate the routes because it is an open source that offers precise route calculations, flexibility, ease of integration, and customization. It delivers fast route calculations with optimal results in terms of distance and time traveled, which are particularly important factors when planning to build a route planning service, especially in the context of electric vehicles.

OSRM uses an incredibly powerful method called contraction hierarchies, which is a speedup technique for shortest path queries in road networks. Combined with the OSM [42] data provided freely, OSRM is able to deliver accurate and efficient route calculations.

The OSRM service interacts with our application via its RESTful API, where we specifically call the endpoints about the route calculation from our internal Python wrapper. The endpoint will receive starting coordinates and ending coordinates in Decimal Degrees (DD) format given by our system through an HTTPS request, and a detailed route is returned by the OSRM side on the response. The returned coordinate data is encoded into an Encoded Polyline Algorithm Format (EPAF) [45], which is a compact format for storing a list of n-dimensional points. It uses variable-length integers to greatly reduce the size of the encoded data, thus making it suitable for web-based applications where resource usage is a significant consideration. [45]

In this type of encoding, each point is defined by its latitude and longitude, which are encoded as signed values. The first pair of latitude and longitude is absolute, and subsequent pairs are relative to the previous point, which makes the data even more compact. Since the returned coordinate is returned into this format, we needed to decode it to be able to work with the route, and for that, a Python library named polyline [46] was used. What this library basically does is reverse the polyline algorithm, translating the encoded string back into a list of coordinates that can be easily read and worked by us. After understanding the algorithm and the process of encoding/decoding, the code implementation was very simple as we can see in the following code 4.1.

```
1    def decode_route(self, route):
2        # Use the polyline library to decode the route
             geometry
3        route_decoded = polyline.decode(route)
4        return route_decoded
```

Listing 4.1: Decoding a route using polyline library

```
Given an example of a decoded route given by the following polyline string

route = _p~iF~ps|U_ulLnnqC_mqNvxq`@

The decoded route output is: [(38.5, -120.2), (40.7, -120.95), (43.252, -126.453)]
```

With the decoded geographical points, we now have some waypoints (represented by each point) that we can follow and use another library to draw it on the map if needed. However, only having these points on our map without knowing the distance between each pair of them, gives us almost nothing about how to use it in the electric vehicle route planner context.

The crucial key for building an EV route planner is ensuring that the vehicles have enough charge to navigate through all the routes, making (if needed) charges along the way. With this said, the next crucial step in our route planning process involved calculating the distance between each pair of waypoints. Having this distance between each waypoint, we could estimate which waypoints the electric vehicle could reach without needing to recharge, and if so, we would search for charging locations using that distance.

To determine this distance, our system uses the Haversine formula [47]. The Haversine formula is a mathematical expression that helps in determining the shortest length from point A to point B on the surface of a sphere. Since Earth has a sphere shape, we could use this formula to calculate the waypoints given by OSRM API (after decoding) returning it on meters.

The following is the version of the Haversine formula's implementation in our system:

$$D = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{4.1}$$

where:

- D is the distance between the two points (along the surface of the sphere),
- r is the radius of the Earth (approximately 6,371kilometer (km)),
- ϕ_1, ϕ_2 are the latitude of point 1 and latitude of point 2 (in radians), respectively,
- λ_1, λ_2 are the longitude of point 1 and longitude of point 2 (in radians), respectively.

This formula is implemented in our application as a method of the OSRM Wrapper [48], [49], with the following code:

```
def calculate_distance(lat1, lon1, lat2, lon2):
    R = 6371.0 # approximate radius of the earth in km

    lat1_rad = math.radians(lat1)
    lon1_rad = math.radians(lon1)
    lat2_rad = math.radians(lat2)
    lon2_rad = math.radians(lon2)

    dlon = lon2_rad - lon1_rad
    dlat = lat2_rad - lat1_rad

    a = (
math.sin(dlat / 2) ** 2
        + math.cos(lat1_rad) * math.cos(lat2_rad) * math.
           sin(dlon / 2) ** 2
    )
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

    distance = R * c
    return distance * 1000  # in meters
```

Listing 4.2: Python function to calculate distance between two points

The integration of OSRM into our system presented some challenges in understanding, decoding, and customizing the returned route data. However, these challenges were overcome by reading and understanding OSRM's documentation and searching for possible solutions to the faced problems.

4.3.2 OpenChargeMap

OCM is another external service used in our system that complements our route planning system. The service can provide a detailed directory of charging stations, covering numerous countries throughout all continents around the globe, which makes it a great tool to integrate with our system. It also provides necessary information on charging stations and other important metrics such as their location, types, number of chargers, charging speed, and user ratings and reviews just like we checked before on this dissertation [22].

Furthermore, OCM allows third-party integrations through its API. Since we can have charging stations' important information, by making requests into their API, we chose to integrate this service into our route planner.

Within our system, we have created a wrapper class for the OCM API which is responsible for all the requests made to this service. Using this customized wrapper, enables us to access and use the detailed data available on OCM internally on our system with more organization and simplicity.

To understand what functionalities and information their API can provide, a research into the service API documentation was made [37]. The table below (4.1) shows a set of services that can be obtained from the OCM API, indicating what is the provided service, its description, and if they were integrated or not in our system:

Service	Description	Used
Points of Interest (POI)	Provides access to the database of charging locations	✓
Reference Data	Provides lists of reference data	✗
Media	Allows retrieval of media associated with locations	✗
Comments	Provides access to submitted comments	✗
User Profile	Manages a user's profile	✗

Table 4.1: OpenChargeMap API Services and their usage in our system

Although OCM provides multiple services through its API, we only needed to use the POI service, which gave us access to OCM charging locations database and a lot of details about each one. In the following subsection, we will explain some of the reasons that led to the selection of this particular service to obtain the charging points data.

Obtaining Charging Points Data

In the initial phase of this dissertation, one core component that we needed to decide on, was where to obtain the charging points data, having a trustworthy source that could provide expansive and accurate data related to the number of charging stations around the globe and providing detailed information needed for the route planner to work properly. Several potential providers of data were considered based on the level of detail, quality, and ease of integrating them with our system.

The first service found was OCM, the platform already described before that provides extensive and global coverage of charging stations through its API. This service immediately filled all the requirements we needed to integrate with our route planner since it had extensive global coverage, frequent updates on the data, and large levels of detail available for each charging station.

Another provider is the ChargePoint API. Although it also has extensive information through an API, it comes with some disadvantages:

- **Restricted Access:** Access to the ChargePoint API was not openly accessible, requiring an agreement with the company. This limited the accessibility of their API in this dissertation.
- **Limited Geographic Coverage:** ChargePoint has a large network, but its presence is stronger in certain regions than others, such as in Europe or North America (NA).
- **Complex Integration and compliance requirements:** From the little documentation and terms and conditions we could have access to, the integration seemed more complex than with other providers.

Lastly, PlugShare [21] is another provider of charging points data and also uses user reviews and ratings. But, the access to its data was not practical and simple, since they do not have an open API.

From OpenChargeMap, we specifically used the function that provides charging stations within a defined bounding box, referred to in the table 4.1.

The interaction with OCM API is performed through HTTPS requests, using a GET method. To retrieve the charging stations inside a predetermined bounding box, the endpoint we are using in the API calls is as follows:

`https://api.openchargemap.io/v3/poi/`

We used the specific parameters defined on OCM documentation that are used to define a bounding box. This allows us to retrieve all charging stations located within this defined area, by giving 4 charging points as parameters. To use them, we need to set in the form of `?boundingbox=[lat1],[lon1],[lat2],[lon2]`, and it will return the charging stations within the latitude and longitude bounds defined by `lat1`, `lon1`, `lat2`, and `lon2`.

To illustrate the concept of a bounding box in the context of our request to the OCM API, an example is provided. 4.2 showcases a geographical area around Madrid, Spain, which is defined by four geographical coordinates. These coordinates, represented by the two black dots on the figure, form a 'box' around the city. By providing these two coordinates to the OCM API, we can retrieve all the charging stations available within this defined area.

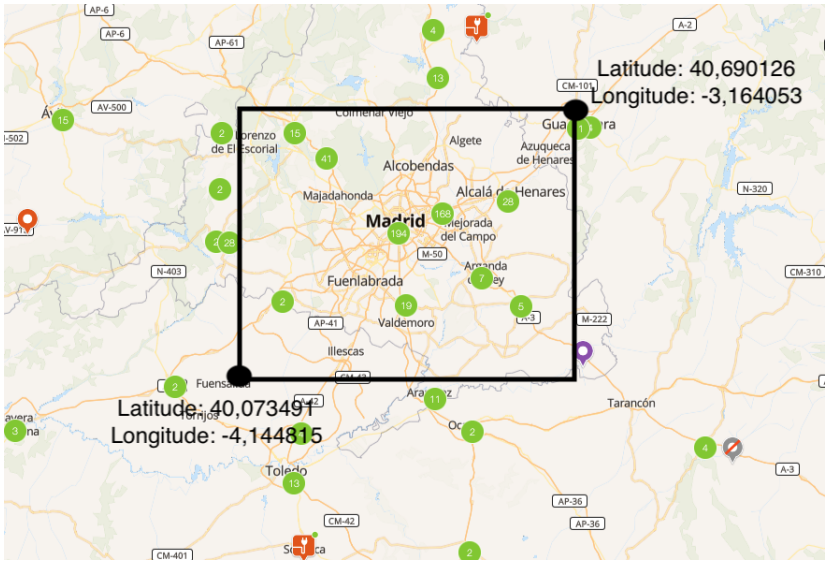

Figure 4.2: Illustration of a Bounding Box defined around Madrid for charging station retrieval

To enhance the precision and relevance of the results, and also to not create a big offset between the route and the local that the vehicle user will charge, we created an internal method that calculates a bounding box around a specific location, with a given offset of approximately 2 kilometers. In the context of our route planner, this method is called every time we need to make a request into OCM API, to get the charging points so we are able to calculate every stop and recalculate the route in the end.

In our route planner, the way we search for charging points is when the SoC of the vehicle starts getting low. A bounding box is calculated when our system checks that the total distance traveled until that point converted into SoC, is less than 35%. When this bounding box is calculated by our system, we need to make sure that it covers a wide area, so it can catch various charging stations in one request to OCM. Although it can be changed in some situations, due to not finding enough or appropriate charging stations, we considered the default and good offset as an area of 2 kilometers, with the center point being the vehicle (on route calculation). This bounding box is created around the user vehicle's geographical point, with the usage of a customized method in our system. This method considers the spherical geometry of the Earth since we need to expand two coordinates (given by the waypoints response from OSRM) into a bounding box with four geographical coordinates. The conversion of the offset from kilometers to degrees is critical, and this conversion varies for latitude and longitude due to the shape of the Earth.

Given the offset in kilometers from the user vehicle point to the edge of the bounding box (*offset_km*), which is never less than 2 kilometers in our system, along with the latitude (*center_lat*) and the longitude (*center_lon*) of the user vehicle location at a specific point in the route calculation process, we first convert *offset_km* to degrees for both latitude and longitude. This conversion uses an approximate value of 111, representing the number of kilometers in one degree of latitude on Earth's surface:

$$lat\_offset\_deg = \frac{offset\_km}{111} \tag{4.2}$$

$$lon\_offset\_deg = \frac{offset\_km}{111 \cdot |\cos(center\_lat)|} \tag{4.3}$$

Then, the bounding box is calculated by subtracting and adding the offset in degrees from *center_lat* and *center_lon*:

$$\begin{aligned} lat1 &= center\_lat - lat\_offset\_deg \\ lat2 &= center\_lat + lat\_offset\_deg \\ lon1 &= center\_lon - lon\_offset\_deg \\ lon2 &= center\_lon + lon\_offset\_deg \end{aligned}$$

In these equations:

- `offset_km` is the offset in kilometers from the center to the edge of the bounding box,
- `center_lat` is the latitude of the central location,
- `center_lon` is the longitude of the central location,
- `lat_offset_deg` is the latitude offset in degrees,
- `lon_offset_deg` is the longitude offset in degrees,
- `lat1, lat2, lon1, lon2` are the latitude and longitude bounds of the bounding box.

With this, we can successfully make requests into OCM API, using this internal method on the wrapper of our system, and we can easily customize the offset of the bounding box.

Processing and filtering the data

Once the request is sent to the OCM API, the server returns a response that contains data about all the charging stations located within the defined bounding box. This includes information about the provider, operator info, charging station usage type, status, address, and connection details.

To exemplify, we extracted only a fraction of the total response structure. The actual response contains much more data and can be consulted on `https://openchargemap.org/site/develop/api#/operations/get-poi`.


```
{
  "Example Results": [
    {
      "DataProvider": {
        "WebsiteURL": "https://openchargemap.org",
        "Comments": null,
        "DataProviderStatusType": {
          "IsProviderEnabled": true,
          "ID": 1,
          "description": "Manual Data Entry"
        },
        ...
      },
      "OperatorInfo": {
        "WebsiteURL": "https://www.afconev.co.il/",
        "Comments": null,
        ...
      },
      "AddressInfo": {
        "ID": 190212,
        "description": "Manual Data Entry",
        "AddressLine1": "10",
        "Town": "Ashkelon",
        ...
      },
      "Connections": [
        {
          "ID": 307235,
          "ConnectionTypeID": 25,
          ...
        },
        ...
```

```
      ],
    ...
    },
  ]
}
```

While all these details can be very important in specific contexts, for our purposes at the route planner, we focused on extracting the "location", "id", "status type", and "connections" for each charging point.

Given the complexity of the response, we decided to process and handle it first. So, the first step in our data processing pipeline was to extract the needed information from this response. This is done with a customized parser that can operate within the complex JavaScript Object Notation (JSON) structure of this API response and retrieve only the fields that are required. The data is then stored inside an internal data structure for further processing.

Further steps involve filtering the parsed data to select the most appropriate charging stations according to a given vehicle (previously entered in our system). Our criteria for selection prioritizes compatibility with the vehicle's port of charging and the fast charging capabilities of the charging station and the vehicle. Compatible charging station connectivity is established by comparing the "connections" field in the API response with the specification of the vehicle's charging port. Fast chargers are simply preferred since they minimize the time of charging and hence the total travel time, but some users might not prefer this which makes the route planning process more complex for each individual one. [50]

As you can see in the JSON example provided above, some of the values provided by OCM on its response, are represented by internal IDs used on their system. Although those IDs are explained with the correct values on their documentation, we decided to transform them in our system to the correct values to have better readability and to make it easier to deal with the data.

With all these steps done, our system can select charging stations that are not just within the vehicle's range but also compatible with its charging specifications and capable of rapid charging, if possible.

One challenge we faced during this integration was the clarity of the documentation provided by OpenChargeMap. Although the development of the customized wrapper helped to deal with some of the problems, the documentation provided by them could be clearer. Some of the request examples provided by them were not working, there were no examples of query parameters and the documentation does not explain well every part of the detailed response.

In the electric vehicle field, route planning is way more than distance and time calculation because numerous factors need to be taken into account. It delves into the intricate dynamics of energy consumption, state of charge management, and making prudent decisions on charging stops. In this section, we discuss the major processes that we have come up with in order to have an efficient, and reliable route planner for EV users. We will be discussing details about the intricacy of energy consumption, how the SoC management was made in our system and how crucial can it be if made wrong. After this, we are describing the decision-making process to come up with the best stops for charging using the Weighted Scoring Method algorithm. Concluding this section, we describe how routes need to be recalculated and refined at the end so that the final users will always have the most optimal and efficient routes.

4.4.1 Energy Consumption and Managing SoC

Energy consumption in the context of our route planner is the energy that an EV uses to travel some distance. The accurate prediction of the EV energy consumption is an issue for efficient route planning not only because their range is significantly lower when compared to combustion engine vehicles but also because they require a different approach to recharge its range, this is, using a charging station. Knowing what to expect from the charging stations present throughout the route is one of the biggest challenges the current landscape of EV route planners deals with.

Besides having to deal with the charging station details, the energy consumed by the electric vehicle can also be affected by external factors such as the terrain, road conditions, driving style, and weather.

Our system uses the Environmental Protection Agency (EPA) data provided by Electric Vehicle Database (EVDB) [51] as a baseline to predict the vehicle's energy consumption across different terrains and routes. After having the baseline distance, we also need a crucial input for our route planner to work properly, this is, the initial SoC that the vehicle has when starting its route.

This SoC is afterward transformed into a total distance by using a simple formula:

$$d = V_r - \left(\frac{\text{SoC} \times V_r}{100}\right) \tag{4.4}$$

Where:

d is the distance

V_r is the vehicle range

SoC is the state of charge (in percentage)

Having the SoC transformed in meters is then way easier to deal with the calculations between the EPA total range, the distance traveled, and the current SoC of the user. If our system needs to transform again the meters distance into SoC percentage (e.g. providing it to the user at the charging stops), it simply inverts the formula.

4.4.2 SoC in Charging Decisions:

Our service uses the proactive strategy that was described in detail in the previous chapter, which means it will start looking for charging stations when the SoC of the vehicle drops below a pre-established value in the system. For the system developed in this dissertation, we decided that a good value was 35%. This means that when the SoC falls below this value, our system starts seeking potential charging stops within a 2 km radius. If no optimal charging stations are found within this proximity, the system allows for a minor drop in the SoC (around 5-10%) but intensifies the search for charging options by expanding the search radius, providing a larger scope to locate optimal charging points. We considered optimal charging stops when the charging points are available and not out of service, compatibility with the vehicle previously inserted in the system (charging port), high charging speed, and located near the calculated route. Even though this approach had good results mainly when testing in Europe, it was still made using a proactive strategy with pre-calculated routes and charging stops, which means real-time adjustments based on live SoC and charging station data aren't possible in the current system. The use of this strategy emphasizes the importance of user flexibility and the system's adaptability to ensure optimal charging stops are selected, which can explain some of the problems current solutions like ABRP may have since they also use a proactive strategy.

Managing and predicting the SoC of a vehicle is essential for ensuring a good journey since bad management or calculation can lead to the vehicle getting out of range and no charging stations being located in the nearby area. Also, in a proactive approach, inaccurate prediction can occasionally lead to not optimal charging stop recommendations, since users can also stop when they have enough SoC to reach their destination. By not having real-time SoC data from the vehicle, our estimations calculations were crucial during the process of decision-making for charging stops.

4.4.3 Charging Station Selection using WSM

Selecting the best charging station among multiple ones is a crucial step in our system to ensure an efficient trip for electric vehicle users. For most cases in our system, the requests made into the ChargeMap API in a given radius usually returned multiple charging stations, which led us to build an efficient method for picking those charging stations properly. Upon receiving the list of potential charging stations within the bounding box from the ChargeMap API, our system starts by parsing the data and filtering the charging stations based if they are operational and according to the charging port of the vehicle, since without any of those two factors, the user is not able to charge at the given charging point. Even with this filter applied, multiple charging stations are still returned by our system, which means we still need to apply a decision-making method.

In our system, we used the Weighted Scoring Method to help us go through multiple charging stations available and pick only those that best fit the criteria. This method was able to provide multi-criteria decision-making, by taking into account multiple factors (some more important for specific users than others).

The charging stations were scored based on:

- Distance from the vehicle's current location (how far is the charging station from the current estimated stop)
- Charging Speed
- Number of Plugs (more plugs might indicate less waiting time)
- Availability (If the charging station is currently being occupied by another EV user)

In order to develop an efficient scoring mechanism to score the charging stations, we utilized a linear scaling technique, which is a popular normalization method used in transforming raw metrics into standard scores. By using linear scaling, we could easily define which values would have a great score and which values would have a bad score. For example, when evaluating the distance criterion, a good value would be less than 1 kilometer, which means our system would rate almost the maximum value.

Linear scaling maps an original value over a wide range onto a pre-determined mathematical scale of typically between 0 and 1 or 0 through 100. In our system, we decided to use a scale of 0 to 100, which can be mathematically described as:

$$\text{Score} = \left(\frac{\text{Value} - \text{Min Value}}{\text{Max Value} - \text{Min Value}} \right) \times 100$$

Where:

- **Value** is the raw metric value to be scored.
- **Min Value** and **Max Value** are the minimum and maximum values possible on the defined bounds, respectively.

To illustrate the application of the technique in our system, let's assume a criteria for the charging speed for the charging station and its compatibility with the vehicle. Assuming 120kW maximum and 20kW minimum charging speeds of a certain vehicle, applying linear scaling would return a score of 50 by a charger that has a speed of 60 kW after linear scaling is applied.

The same can be used for the distance criterion, where we transform kilometers in score by defining what is a good or bad value (max and minimum). Evaluating the proximity of charging stations, we considered that a station that was less than 1 kilometer away has got a score of 100 (optimal proximity) and a station 20 kilometers away has got a score of 0 (farthest reasonable distance). Stations at intermediate distances have their scores linearly interpolated between these bounds.

After the calculation and normalization of the calculated scores, the system proceeds by calculating each weight on the scores. The final score for a charging station S_i for each retrieved charging station i is calculated as:

$$S_i = 0.45 \times D_i + 0.35 \times C_i + 0.10 \times A_i + 0.10 \times P_i$$

Where:

- D_i: Distance score of station i
- C_i: Charging speed score of station i
- A_i: Availability score of station i
- P_i: Plug count score of station i

After scoring each one, the charging station selection is quite simple since we only have to choose the one with the highest score. However, to ensure reliability and efficiency, we perform secondary checks. These checks involve reconfirming the service status of the charging station, its compatibility with the vehicle, and the charging speed. Since we are using a method that calculates scores for each charging station, and with the charging speed having only 20% of impact, we decided to make additional verifications according to this, assuring that the vehicle will reach a charging station that is not only compatible but also functional and has good charging speed.

To make this validation happen in our system, we first start by looking at the charging speed of the charging station. If the charging speed is the highest compatible with the vehicle, the algorithm continues for its next phase, which is returning the best charging station previously calculated. However, if the charging speed is not optimal for the inputted vehicle, we look for the other top 3 scored charging stations and check if their charging speed is higher than the highest-scored one. If so, the system evaluates the distance which is the criterion that impacts more the scoring weights. If the difference in distance is less than 5km, then, the charging with the higher charging speed is selected. We considered that although the distance is a crucial criterion that directly impacts the SoC of the vehicle, the charging speed should be more relevant in certain cases.

So, for better understanding let's take an example of a journey from Lisbon, Portugal to Milan, Italy. As the vehicle approaches Salamanca in Spain, the SoC shows that a charging stop must be made. The system pulls data for charging stations using OCM API in proximity and scores them. A station that is 5 km away but has good charging speed and multiple plugs available might score more than a closer one with fewer plugs and slower charging speed. Thus although the first station is slightly farther it gets selected as it provides quicker and reliable charging compared to the second one.

With all of this, the system guarantees that the final user will reach a charging station that not only is compatible and functional but also has a good charging speed. We selected the Weighted Scoring Method for its simplicity, efficiency, and easiness to be customized as we

can see by these last validations referenced. Through the assignment of weights, the system can easily adapt to different preferences of users or new insight into utility charging stations being used. However, though it may be effective, other machine learning approaches that are more dynamic and adaptive might give better results in certain cases. The limitation of such an approach especially in our case is that large datasets would be needed for training purposes, resources which at this moment are still limited due to operational constraints with the charging station network overall. In future system improvements, machine learning can be adopted as more data on the routes and charging stations becomes available.

4.4.4 Recalculating the final route

Considering the necessities of electric vehicles and all the things described before, recalculating the final route after having the charging stops is a must-step. After identifying potential charging stations, scoring, and filtering them out, the original route from OSRM API needs adaptations to guarantee the integration of these charging stops into the final route.

The system starts by making the initial request into the OSRM API to retrieve the initial and base routes by giving a start and end location. This primary route is calculated without taking into account the electric vehicle's charging needs or the availability of charging stations along the way. This initial route is optimal in terms of distance and time to the final destination since the OSRM service already does that for us.

After the initial route generation, the possible charging stops are suggested by the system through querying for potential charging points over OpenChargeMap API and scoring them with the Weighted Scoring Method described in previous subsections.

Once the charging stops have been identified, routes are regenerated with several requests to OSRM based on the number of charging stops required. The process involves:

1. Breaking down the initial route into segments: Before and after each selected charging stop.
2. Re-querying OSRM for each segment to generate a new route that includes the charging stop.
3. Merging all segments to produce the final route that the user will follow and is displayed in our system.

The length of the recalculated route could be a bit longer than the first calculated one because it makes sure that the vehicle stops in the identified charging stations which as we have seen before can depend on the charging speed of it and not only their distance of them with the vehicle along its calculated route.

In some very rare cases, after the completion of the recalculation has been done successfully, if the chosen charging station suddenly becomes unavailable (e.g., goes out of service), the system may need to make another series of recalculations, since it will pick another charging station from the selection described in the previous section.

Data visualization plays a crucial role in the comprehension of large volumes of information. In the context of our Route Planner, the picture is a graphical representation of the data retrieved and processed by our services, that serves as an essential element to bring clarity and provide insights to the final user.

With the use of effective data visualization tools, complex geospatial data can be represented as simplified maps and plots where it becomes easily understandable. Its main advantage lies in providing users the ability to visualize the planned routes and the charging station's locations to charge along with other related data along the way. [52]

In the following sections, we will explore the use of MongoDB to store necessary data in our system and two powerful data management and visualization tools, Folium [53] and Streamlit [54], that were used in this dissertation. Their application will be demonstrated in handling and presenting geospatial data related to route planning and electric vehicles.

4.5.1 MongoDB

MongoDB [40] plays a key role in handling the large amount of loaded data associated with mapping, route planning, and vehicle management.

In our project, MongoDB is used to manage and store a variety of data pertinent to route planning and vehicle management. This information is structured into three main collections, each serving a specific purpose and containing unique types of data. A summary of these collections and their respective roles is provided in the table below.

Collection Name	Description	Key Fields
Vehicles	Electric vehicle data	range, type, brand
User_Vehicles	Vehicles inserted by users	user_id, vehicle_id, custom_attributes
Routes	Calculated user routes	route_id, user_id, waypoints, estimated_travel_time

Table 4.2: Summary of the used MongoDB collections

Starting with the Vehicles collection, it was used by our system to store all types of information about all electric vehicles. To populate this collection, an open-source dataset from the EVDB [51] was used. The dataset contains a great amount of information on electric vehicles available in the market, including details and specifications such as battery capacity (in kWh), vehicle range, brand, and many others. When a user chooses a car in our system, we immediately check the vehicle's existence in our database by retrieving the brand, model, and year from the user input and querying it into our Mongo collection. If it exists, we can immediately get crucial info for our route planner calculations, such as the vehicle estimated range calculated by EPA.

Another use of MongoDB within our project is storing different routes, for every calculation made. Every route represented as multiple waypoints can be held by including them as documents within a collection called "routes" within MongoDB. The flexibility of MongoDB's document model allows us to hold additional information along with each type of route, for instance, time for travel, other alternative routes, the vehicle used, and many other fields.

This saved data is only being saved at the current state of the system, however, Mongo has a geospatial indexing feature that can be exceptionally useful in the future. This feature provides the capability to execute location-specific queries efficiently and can enable carrying out operations like searching all routes over a certain distance from any given point. This can be very helpful for providing users with suggestions regarding their current positions, possible routes they could take, faster responses with pre-calculated routes saved on our database, and many more.

For last, we also used Mongo for the storage and management purposes of user vehicles. For every user input action with a different vehicle, we create a document in our collection, saving some user details and their vehicles, so they don't need to insert it every time they want to calculate a route.

4.5.2 Communication between services

In our architecture, the route planner and the frontend are two independent services that communicate with each other using HTTPS requests. By making these two services independent, where each service has a single, well-defined task, we were able to enhance our system's scalability and resilience, making it easier to develop, maintain, scale, and reutilize it if necessary in the future.

The Route Planner service (backend) calculates the optimal routes for a given start and end point, considering factors like distance, battery range, charging stations, and user preferences as already described before. These routes are then stored internally and made accessible to the frontend via an internal API.

Through the use of the 'requests' library, our system (frontend) sends HTTPS requests to the backend open API, retrieving the calculated routes when requested by the user. The routes, along with the other data such as charging stations, are then visualized using Streamlit to run a web app and Folium to display the map with the route.

By using this type of architecture and communication we made sure our frontend service remains light and focused on its main role: providing an excellent user interface for visualization.

Thus, the microservices architecture facilitates a separation of concerns, where each service does what it does best. The route planner focuses on the complex task of route calculation, while the frontend focuses on presenting the results in an accessible, user-friendly way. The communication between these services, using HTTPS requests ensures that data is passed accurately and efficiently, contributing to a great and complete user experience.

4.5.3 Folium

Folium is an open-source Python library used for creating interactive maps [53]. It provides a high-level interface to the Leaflet JavaScript library [55] and is capable of creating sophisticated visualizations with just a few lines of code.

This powerful mapping library translates the complex, abstract arithmetic of routes into concrete, intuitive visual representations, enhancing the system's usability and the users' understanding of the calculated route.

The use of Folium in our system runs through three steps: First, we instantiate a map centered at the first waypoint of the previously calculated route by the system, setting an initial geographic point for our visualization. In the following image, the route starting point is located in Paris.

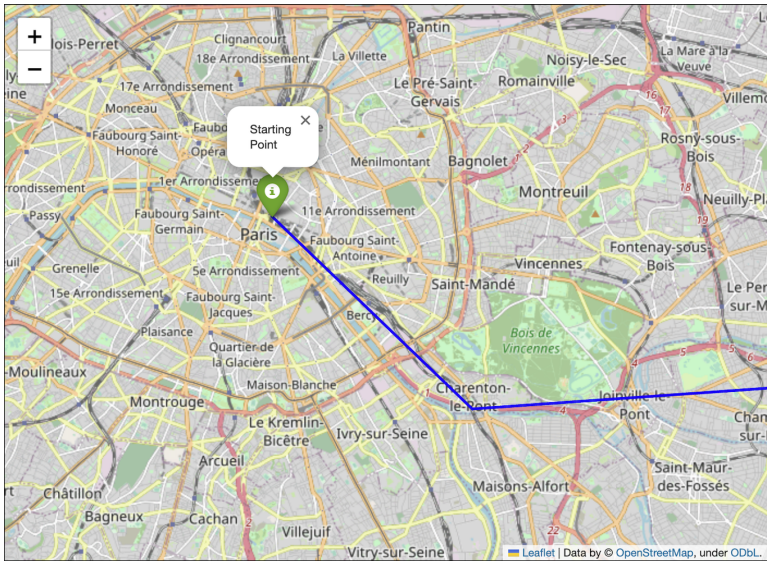

Figure 4.3: Initial Folium map centered at the starting waypoint of the calculated route, marked with a green icon.

Next, we iterate through each waypoint using Folium's PolyLine feature to draw a line from each point to its successor. With this function, we were able to visualize the calculated route, showing the path from the start to the end point, passing through all intermediate waypoints.

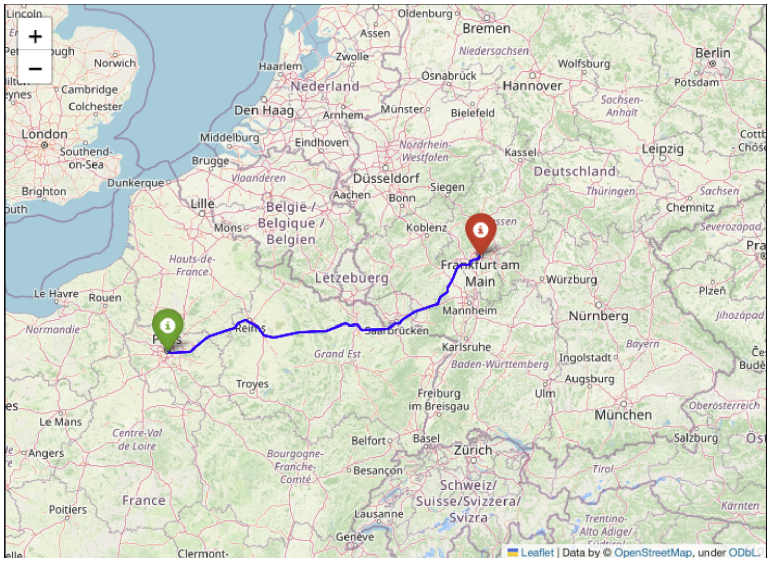

Figure 4.4: Visualization of the calculated route with blue polylines connecting each waypoint.

Besides drawing the start-to-end route, Folium is also used to mark off charging points along the specified route. The charging points location, also comes in the response as waypoints, but there is a tag separating them, so we know when to mark a point on our map. The charging stops depend on several factors as we discussed in the previous section, but in this part of the system, its only job is to draw the charging stops according to the response from our system backend. The required data about the charging points comes from a request to our internal API route planning service. After getting this data, these charging stops at the locations can be colored over a map using the 'folium.Marker' function so that it can be seen by the user through a visual guide, and they know where they need to stop their vehicle to recharge it during its journey.

Figure 4.5 represents our completed interactive map with the drawn route, which is rendered within the Streamlit user interface using the folium_static function from the 'streamlit_folium' [56] package:

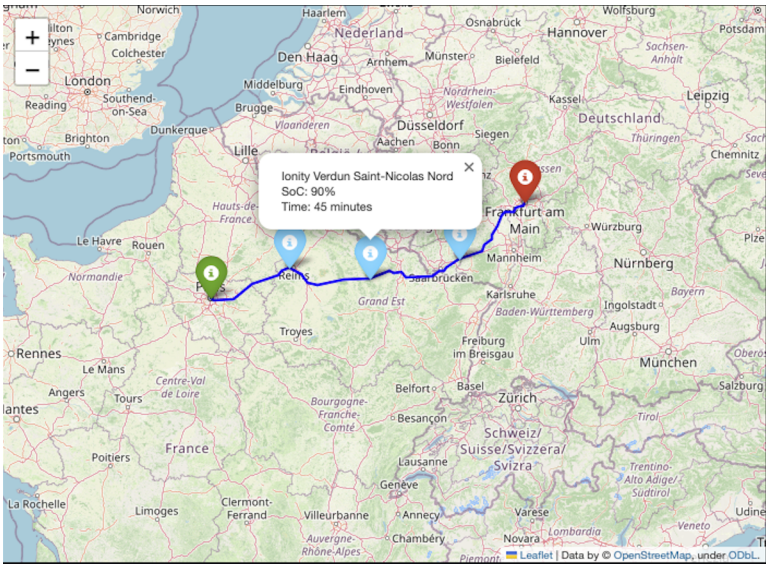

Figure 4.5: Visualization of the calculated route with the charging stops.

By visualizing the routes, Folium assists our system in transforming complex routing data, into comprehensible and interactive images. Through graphical representations of the calculated routes, users are given an overview of the start and end points, waypoints, charging points and its details, the route progression, and many details about the trip.

As the backend development was complex and time-consuming, we needed to choose a tool that would not only facilitate the integration with the data provided by our route planner but could also be able to properly show the final route with all of its details.

For this reason, Folium over other tools was chosen, not only because of its ease of integration but also because of my familiarity and comfort with the Python programming language. Also, by using this language we could use other data manipulation tools libraries like pandas [57] and NumPy[58] that I am also familiar with.

However, it's important to be clear that there are other mapping tools available for similar purposes. Libraries like Leaflet.js [55] or even Google Maps API [59] could also have been

used and be a good fit for our system. Other alternatives in Python also exist, like Plotly [60] and Bokeh [61], but their main functionalities are focused on plots and not geographical data. In conclusion, Folium's simplicity, direct integration with Python, and great geographical data visualization made it the best fit for our system.

4.5.4 Streamlit

Streamlit is an open-source Python library that makes it fast to develop data applications. It simplifies the process of developing and deploying custom web apps dealing with data, including geographical ones. [54]

In our route planning system, Streamlit serves as the core of our user interface, providing a simple and customizable interface. Before the route calculation is initiated, users need to supply the necessary data through this interface, such as the initial and final locations of the route, detailed information about the vehicle being used (brand, model, and year), and the initial SoC of the vehicle. After inputting this information, we can make an HTTPS request to our route planner system to calculate the route and display it when the response comes with the help of Folium. Integrating with other Python libraries like Folium and pandas, allowed us to manipulate the necessary data alongside an interactive visualization.

The main advantage of Streamlit in use for this project is once again the simplicity that this tool could give us for a project that focuses mainly on the backend.

Comparing Streamlit with other similar tools for creating web-based data applications, such as Dash (Plotly), Flask, or Django - none shows the simplicity and fast deployment of Streamlit, and since we only needed a simple way of making a web app, this tool was the better fit for this dissertation.

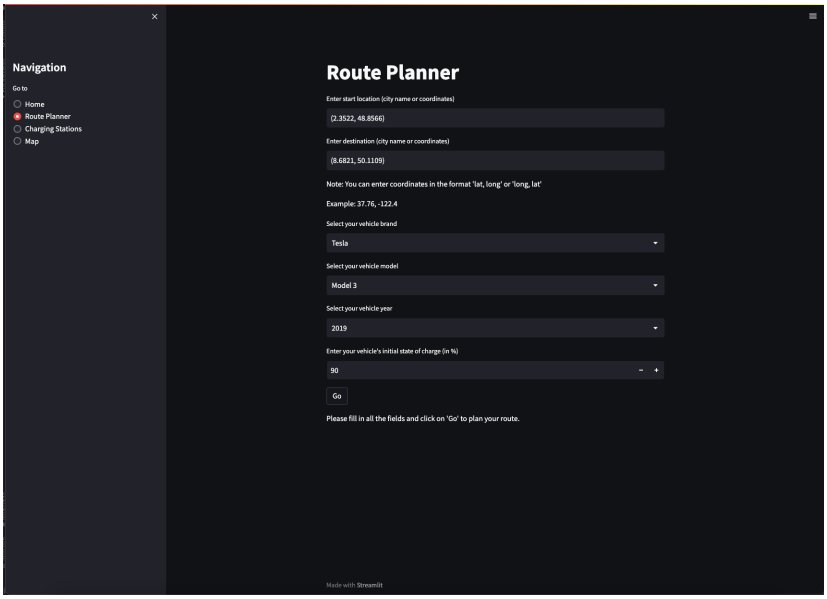

Figure 4.6: Input in our app using Streamlit

We also used containerization during the development of our route planning system to ensure that the system has consistent behavior in different environments. To that effect, Docker [62] was used to containerize services like our database, route planner, and frontend. This encapsulation made it easier for the services to be independent and flexible, allowing subsequent developers to set up the project regardless of their specific system settings or software installations.

One of the major goals that lay on our roadmap to the future is migrating our system to a cloud-based platform. Although it has not been made yet, we should consider this migration for several reasons:

1. **Scalability:** Cloud platforms offer the benefit of easily scaling up or down as we need, according to our traffic and computation resources. As our route planning system grows and gets increasingly used by more and more people, we will be comfortably increasing load by scaling capacity without drastic architectural modifications.
2. **Reliability and Redundancy:** Cloud services usually have multiple data centers over different regions. This in turn means our application can have its redundancy built ensuring high availability and resilience against, for example, unexpected disasters.
3. **Security:** Cloud providers usually make heavy investments to secure their infrastructure following international security standards. This could assure us of a robust and secure deployment environment.

CHAPTER 5

Results and evaluation

In this chapter, an evaluation of the developed route planning system for electric vehicles will be presented. The chapter focuses on the system's performance, scalability, and comparison with the other existing solutions, understanding the strengths and limitations of the system, as well as seeing where it stands in the current landscape of EV route planners.

The evaluation of any system is a crucial part of any development lifecycle, and it wasn't an exception in this dissertation. By doing it, we had an analysis of all the system's functionalities and had a conclusion on the areas of potential improvement. With this, we were able to have a list of possible upgrades regarding future developments and research that will be discussed in detail in the next chapter.

We will start by analyzing the performance of our system by using some metrics like processing and response time, memory usage, throughput, and load testing. We looked into the accuracy with which route calculations are done, the time taken for these computations, and conditions that affect the system's performance such as the distance between start and end points or the lack of charging stations in some routes.

After that, we will talk about the scalability and reliability of our system and its capacity for handling increasing workloads, its consistency in realizing accurate results, and explore how our system performs under increasing demand, as well as what that tells us about its potential in a real world application.

Lastly, we will compare our system against other route planning services available and already referenced in detail before. By comparing it with other services, we can look for possible next steps and check where our service can be improved.

Performance analysis is essential in any software development and it was even more relevant in our system given that real-time processing and efficient routing algorithms are needed in order to provide a good and fast user experience for the final user. The goal of this analysis is to identify any bottlenecks in the system, and areas for improvement, as well as making sure it is scalable in the future.

5.1.1 Processing and response time

The response time is the time taken for a system to respond to a user request. This metric is normally used for having a good user experience since a system that takes too much time to respond can lead to user frustration and consequently exiting it.

The response time of our system includes the processing time which includes the time it takes for our internal API to respond to a request from our frontend. The response time is similar to the processing time as the additional latency introduced by an API request is minimal since we are using an internal one.

Although using external services like OSRM and OCM to calculate the routes and get charging station data respectively, helped a lot in the development of this dissertation, using them as crucial pieces of our system, created a dependency on their APIs to make our system work properly. If for some reason we can't get a response from any of the APIs or they are taking too much time to give a response, this will directly affect our system.

With this said, the processing time of our system, which is how long it takes from when a request is initiated until a route is fully calculated, depends on the complexity of the route. For really short routes with 0 charges needed, it takes less than a second to make the request since the system immediately checks that the current SoC of the vehicle is enough to perform the route, and with this, making no external requests to the services referenced before. For routes that need 2 or 3 charges, it takes approximately 3 to 4 seconds to calculate the whole route with the necessary stops at charging stations. For longer routes (more than 10 charges needed) that demand more processing and calculations on our internal services, multiple requests to OSRM to recalculate the routes, and OCM to find the optimal charging stations it takes more than 12 seconds to provide the final route.

We evaluated the response time of the system by executing five different routes in Europe. For the results shown bellow, the calculations were made with a 2021 Tesla Model 3, starting at 90% SoC. To make the test, ten sequential requests with the same initial and end point were made into our system. To measure the time, we used Python's 'time' library for timing each request. The results provided a perspective on how the system performs when computing a complex route with multiple charging stops or a simpler route with fewer charging stops.

As evidenced by the chart 5.1, the boxplot shows the variability in response times across the different routes. It is worth mentioning that these were 10 sequential requests, so the chances of any of the initially pre-planned charging stations failing to be available between one request and the next were pretty minimal. In case it had occurred, not only the routes could have differed but also the response times, if it was the only available charging station nearby. If this situation happened, more recalculations and requests would have been made because the system would have tried to look for a substitute charging station, especially if the charging station was the only one available within the default 2 kilometers area. However, as already referenced this is a very unlikely situation to happen.

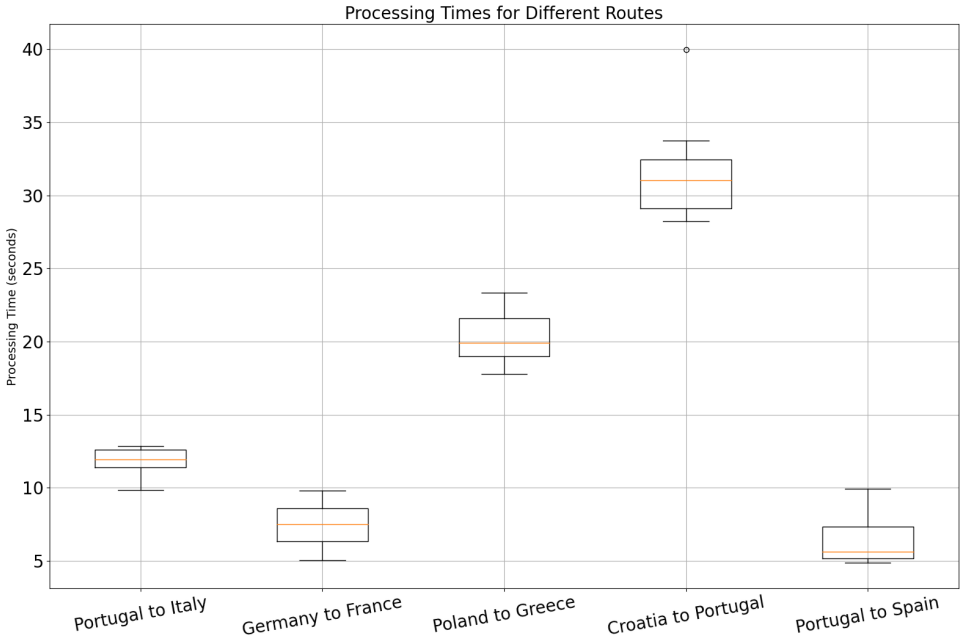

Figure 5.1: Response times for ten sequential requests from different routes in Europe.

5.1.2 Memory usage

This test measures the amount of memory used by the system during its processing time. In the process of evaluating the memory performance of our system, we used our local environment to perform this test. This test was run on a MacBook Pro from 2022 with 16GB of Random Access Memory (RAM), with all our docker container services running. To monitor the test, we used a small Python script that was able to track the memory consumption over one minute by using the library psutil [63], while doing 10 sequential requests to our system.

The consistent performance from our results shows that the system is capable of producing a constant output every time, under a given set of conditions which shows stability for the system. In the course of the test run, the amount of memory used hardly changes from one moment to another. Interestingly, it kept hovering in a range of barely 3.5% of its maximum and minimum values.

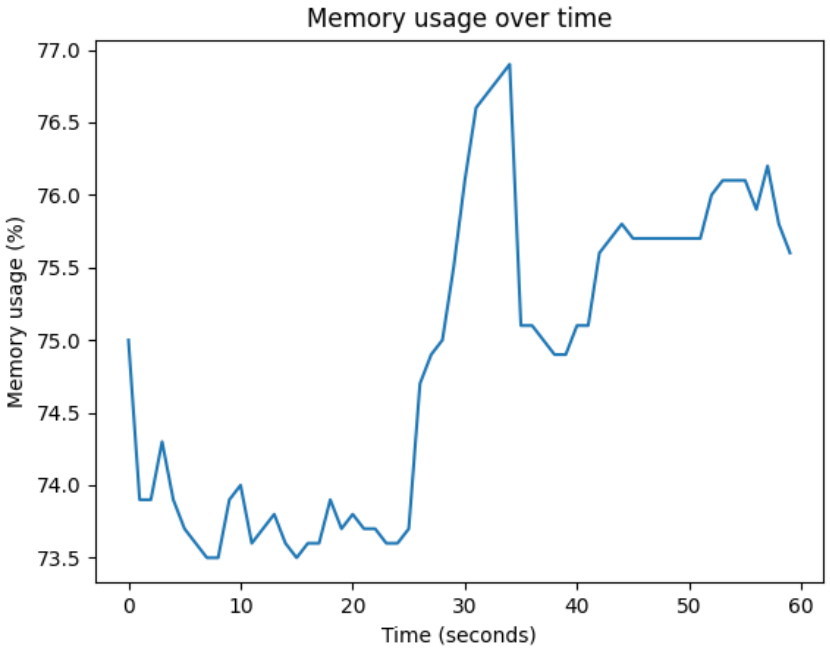

Figure 5.2: Memory usage for ten sequential requests from Portugal to Italy with 14 charging stops.

Nevertheless, while the numbers in this local case are good, we should still keep in mind that there might be variances in the performance on different setups or under different types of loads, especially when testing this on a high-end machine and locally.

5.1.3 Throughput and load testing

Modern web applications and services are evaluated through throughput testing which measures the throughput of a system, i.e., the rate at which the system can handle requests within a given time.

However, the current scope of our project was to deploy our system in a local environment. This local deployment does not have the real-world network conditions of latency, variable loads, and user accesses to put up a challenge. With all these limitations, the performance characteristics of the system, and their responses to the above real-world conditions would not be represented correctly in a local test. For this reason, we decided to skip the throughput and load testing for this iteration of the research.

In this section, we discuss two key aspects that also define the success of a software system: scalability and reliability. The first deals with how our system can be expanded to handle growth by either serving an increased number of users or dealing with a greater dataset. Reliability means the system should be able to provide accurate results over time while having a consistent performance, for example, in the context of our route planner, it should provide the most optimal and correct route for the final user.

Both these parameters are essential to our route planning system. For scalability, as the usage of electric vehicles increases worldwide, the system should be able to deal with an increasing number of users without sacrificing performance. For reliability, the system should also be able to expand the data related to routes, charging stations, vehicles, and user feedback, as time goes, so it is able to provide even more accurate and optimal routes.

In the subsections that follow, we discuss in more detail the analysis of the scalability and reliability of our system.

5.2.1 Scalability analysis

Scalability refers to a system's capacity to manage an increasing number of tasks or requests over time. Although our route planning system's scalability was only evaluated by load testing the system, by choosing a microservices architecture as our initial design, it is way easier to scale the system in the future. Microservices architecture as we have already seen before, allows the distribution of different services across various servers or machines, enhancing the system's load management. However, to increase the system's scalability further, future work might involve incorporating additional technologies and internal services to our system, strategies like load balancing, distributed computing, and even auto-scaling tools.

5.2.2 Reliability analysis

Reliability in the context of a system, refers to the system's ability to perform a required function under stated conditions for a specified period. For our route planning system, reliability is mostly about producing correct and optimal routes consistently. In its current state, the system runs into errors when attempting to find routes in areas with low charging station coverage, or when the number of requests per minute exceeds a certain limit. These issues are related to the worldwide lack of public charging stations, as already described previously at the beginning of this dissertation.

The improvement in our system reliability has involved error handling mechanisms in which the user gets feedback if there is an error occurring while using the system. If the system identifies that there are not enough charging stations in the calculated route, the system warns the user that the calculated route might not be optimal due to the lack of charging stations found between the start and end points. If the system exceeds the limit of requests per minute, it alerts a try later message to be sent by the user. Such feedback can help in understanding what was wrong and allows either correction or waiting for the availability of the system.

However, further enhancement in reliability comes through the use of more customized and internal services, rather than relying on external ones. Having our system designed as a microservices architecture, we can scale it to use other internal services, in order to provide even more accurate data, better results, and less dependability with external services, which gives us more control over the system. An example of this potential scalability could be to transition from using OCM API to building our own dataset and system of charging stations. This internal service would communicate with our main route calculator service via HTTPS requests, similar to how our frontend service does. Although OCM provides great data about the charging stations, it may not cover all the charging stations worldwide. This lack of charging stations might be because the contracts and agreements between the charging station operators and this type of service like OCM are complex.

Nonetheless, more formal methods and techniques for quantifying this reliability are yet to be used and give room for future work.

For the improved reliability of the system, fault tolerance and redundancy strategies could be implemented. These would ensure that even when problems arise, the system continues to function intact. Strategies like retrying failed operations, and providing backup services that can take over if a primary service fails are examples.

5.3 Comparison with existing solutions

In this section, we will be comparing the already existing solutions referenced before in this dissertation like ABRP, PlugShare, ChargePoint, and ChargeMap with our system at its current state. We will be using the results and the analysis made in the previous section for comparison and also making an overall perspective of the system checking what still can be improved in the future.

In the field of route planners for electric vehicles, the current solutions have evolved when compared with a few years ago, providing features with unique capabilities as we already discussed before. ABRP, PlugShare, ChargePoint, and ChargeMap all made considerable contributions in the area of route planners for electric vehicles. However, just like any standard technology system, there is a constant evolution with a lot of room for improvement.

Every existing solution, despite its great algorithms and robust databases, has its limitations:

- **Coverage Limitations:** Not all solutions have comprehensive worldwide coverage. Certain areas might have sparse data or might not be covered at all.
- **Accuracy:** There can be occasions when the routes suggested are suboptimal because of various reasons such as limited and outdated information about the charging points' details and availability, dynamic traffic conditions, road infrastructure changes, weather conditions, system glitches, algorithm limitations and many more.

- **User Interface and Experience:** Although the final user will usually prefer an optimized solution instead of a pretty one, the user interface and experience of an application can be a factor in choosing between one system or another.

An evaluation was performed to compare the performance of our system with ABRP, ChargeMap, and PlugShare. Since ChargePoint only includes a system that localizes the charging points and the user needs to create the route manually, we decided to exclude this service from the results below. The primary concern for this comparative analysis was the response time (RT) for route planning requests, which tends to be one of the critical metrics for deciding how efficient and usable this type of system is.

Ten individual requests were made on all systems with different routes across Europe. As we can see in Table 5.1 and Table 5.2, our system's respective response times varied between 6.43 average seconds and 31.61 seconds, depending on the route length.

An important aspect to consider in this analysis is the unique functionality of PlugShare. Unlike the other systems evaluated, PlugShare primarily focuses on returning multiple charging stations along a given route and does not incorporate the critical aspects of calculating the necessary charging stops or selecting an optimal charging station. This way it does not offer an optimized route plan based on the specific charging needs or preferences, and for this reason, it returned faster response times when compared to the other services.

ABRP showed an average response time that varies from 7.90 seconds for shorter routes to as much as 16.24 seconds for longer ones making it the most optimized solution of all services, considering that PlugShare does not take into account the charging station selection or charging stops, which makes it an inaccurate solution. It is important to note that the first request to ABRP for longer routes takes a considerable amount of time. This is due to ABRP's caching system for its routes, which allows for much quicker responses for subsequent requests. However, since ABRP represents a proprietary service and is not open-sourced, details about particular optimizations used in their system are unavailable, so this is only speculation.

Route	Local System	ABRP	ChargeMap	PlugShare
Portugal to Italy	11.8	8.5	10.8	2.2
Germany to France	7.4	6.4	9.0	2.5
Poland to Greece	20.2	16.2	22.5	2.2
Croatia to Portugal	31.6	12.0	30.6	3.0
Portugal to Spain	6.4	7.9	7.5	2.6

Table 5.1: Average response times for different routes and services

ChargeMap does not have an included caching system like ABRP, and for this reason it returned longer response times just like our local system.

It is also important to highlight that all of these tests were made manually over the services' web page or mobile application, and this reason might explain the difference between some of the results shown.

Route	Local System	ABRP	ChargeMap	PlugShare
Portugal to Italy	11.9	8.4	11.2	2.1
Germany to France	7.5	6.5	8.7	2.3
Poland to Greece	19.9	12.6	21.2	2.0
Croatia to Portugal	31.1	10.1	29.8	3.0
Portugal to Spain	5.6	7.9	7.14	2.4

Table 5.2: Median response times for different routes and services

All the systems returned good response times with only ABRP performing better in longer routes than the others for subsequent requests since it uses a caching system. For this reason, we need to consider other factors such as features, integrations, and user experience and not merely confined by raw performance metrics alone to pick a better system.

While ABRP may have a wider set of features since it is a commercial company that develops the system, the system developed in this dissertation is also really accurate with its returned charging points on the route when compared with ABRP on its core functionality. However, it's not possible to make a one-to-one comparison with ABRP since the details of their various technologies and integrations are not open for analysis.

In terms of usability ABRP, PlugShare, ChargePoint, or ChargeMap typically offer a more refined user experience since they have a commercial product currently on the market. This is not very astonishing since our primary aim was for academic purposes building an accurate core functionality in terms of decision-making and picking the correct route and not really refining the user interface.

Although every solution has its strong sides and each of them can be a useful tool in real-world situations, users should still be careful with any of the systems. Both ours and commercial ones like ABRP are based on a proactive strategy, which means the route isn't changed according to real-time events, and unexpected situations that directly affect the route might happen.

When talking about accessibility or scaling, our solution was specifically developed for academic purposes and therefore will be free to access to everyone. Solutions like ABRP work in a commercial setup and may thereby introduce pricing or usage considerations when trying to access its product.

With all of this said, while raw performance metrics can be used to understand the performance of the system when compared to the others, the essential basis factors such as the system features, reliability, accessibility, and user experience complete the picture.

5.3.1 Accuracy and Route Optimization

A primary advantage of our route planning system lies in its commitment to delivering accurate and optimal routes. While speed might not be a crucial metric, our system significantly shines in its ability to generate the best routes considering the most important factors that users highly value. Our system provides optimal routes, thanks to our sophisticated route calculation algorithm that takes into account factors such as the current state of charge of the

vehicle, battery efficiency, the locations of charging stations, and its availability, accessibility, and charging speed.

When compared to the global market, our route planning system does quite well offering a range of functionalities that in many aspects equal those of the services in the field where accuracy and optimal route calculation are the strongest features.

However, when only compared with A Better Routeplanner (ABRP), the most complex and complete service in the current market, some differences can be noted. Because of its consideration for meteorological conditions in its premium version, this does have the potential to influence electric vehicle route plans simply because it may impact the SoC of the car. Further, one factor that is often ignored but also plays a significant role in battery consumption is heat/cold, something that might also be considered by ABRP along with the road elevation.

However, it is important to underscore that even as ABRP incorporates some of these advanced functionalities in its paid service, quite a number of the solutions available in the market do not have such features. Our system for example while it might not have the factor of weather conditions or road altitude today still competes favorably thanks to its principle when planning routes and considering core essential factors. Also, one of the reasons it is favorable is its potential to scale due to the choice of a microservices architecture.

Currently, the EV route planning ecosystem offers some services that were already described at the beginning of this dissertation. Though particular features perhaps brought in by some solutions are not seen elsewhere, a good route planning system actually comes down to delivering accurate and efficient routes consistently. Our system demonstrates this capability while still keeping open for future development and integrations such that it remains versatile and relevant in the context of the EV field.

5.3.2 Scalability

Because of proprietary restrictions and the closed nature of many of the services referenced before, it's hard to make a direct assumption of their internal architectures, and with this, it is consequently hard to compare their architecture with our system in terms of scalability. However, what we can conclude with certainty is the scalability of our system. From the beginning, our solution has been carefully created to embrace growth and adaptability. The use of a microservices architecture makes sure that each component or module in our system works independently, so scaling can be easier in the future, by supporting an increased number of users and facilitating other integrations that can make the system work better and even more accurately. Beyond that, the routing algorithms at the core of our system are also scalable in order to be even more improved as time goes on. They can be refined and adapted as more data becomes available and as new technologies surge and advance. One possible future enhancement we are thinking of is the direct integration with charging networks by having another independent service for this, which could offer even greater results overall in our route suggestions.

5.3.3 Worldwide Coverage

Comparing the worldwide coverage with the other existing services, we could conclude that our system was able to work properly in every single part of the world unless it is given a start and final destination that is impossible to drive by with a car (e.g. From America to Europe) and so impossible to calculate the route. The global reach in our system was possible due to the integration with the OSRM and the OpenChargeMap API since both of them work worldwide either by retrieving routes or retrieving charging station details.

One of the services that have no coverage in the context of a route planner worldwide is ChargeMap. [22] This service provides charging station details throughout the whole world, however, when trying to plan a route outside of Europe through their app, an error is returned. It is hard to conclude why their system is not able to provide routes outside Europe, even though they have contracts with all charging stations throughout the world while still being able to provide real-time data from the charging stations. We believe this is a work in progress, and in the near future, users are going to be able to search for routes in this application worldwide.

It is also important to note that the adoption of electric vehicles is way more prominent in Europe or North America than in the rest of the world, and this directly impacts not only the construction of the whole public infrastructure needed in order to charge the vehicles but all the technology and applications related to the electric vehicles ecosystem.

5.4 Implications of the results

The coming age of EVs literally, means a new era in our global movement toward sustainable transport. The technology behind EVs is not only about the cars but also the ecosystem supporting their operation including the charging stations, all the software and technologies behind it, applications that support the user to find the best charging stations and their locations, the route planning, and many others. The current project, to develop an optimized route planner for EV was really ambitious. This effort led us to several insights, that were concluded from all the work done until now.

One of the biggest problems nowadays and when building a route planner for EVs is that there are a lot of complexities involved in coming up with an ideal route planner for this type of vehicle since a lot of criteria have to be taken into account. The driving style, road elevation, weather, availability, charging speed and accessibility of charging points, the amount of charging points near the route, and the type of algorithm used to get them on the route all add layers of intricacy to the situation, but especially the lack of charging stations in the public road. Although our algorithm did a great job and the best possible result given the algorithm we used and the amount of data we had, human intervention cannot be entirely discarded in the end. This is, even in real-world apps like ABRP, errors, and bad results can happen, so, it is recommended to the final user to check not only the final route by himself but also the returned charging stations and look for an alternative for each one if possible. With

this, we can conclude that when it comes to charging station availability and accessibility, real-time and high-quality data is very important.

One of the constraints we had while developing the system, was the lack of data that is available from open source APIs like OpenChargeMap. Also, many of the services available provide outdated data and details about the charging stations. The ideal scenario for our system is to have our own independent charging station retrieval service, with the data being updated regularly with an indirect connection to each charging station provider or network. The charging station data in our system comes from the OCM API, which not only creates a dependency in our system on this service but also directly affects its performance by making multiple requests on their API throughout the calculation of the route. Having our own service with its independent database, the system could directly get the data, instead of using and depending on an external service to get it.

Summarizing the discussions in the sections and chapter, it can be seen that opportunities and challenges exist in the EV landscape, mainly in the quality and availability of data. Even though innovations and new technologies have played a great role in our system, human intervention such as final checks has still remained valuable. Our choice to design our system using microservices architecture not only complied with current best practices but also provided future scalability and adaptability necessary for this type of system.

CHAPTER 6

Conclusion

As the world transitions towards a greener way of transportation, electric vehicles have quickly moved from being a small product used by a small group of persons into a big industry having millions of users nowadays. More than their environmental benefits, EVs have a promising future with all the technological advances and they can also be a cheaper way of transportation when compared with the combustion engine ones. It is in this context of the growing popularity of EVs and its expected rise in transport use in the coming decades that our project has been proposed and set out.

In this final chapter, we will reflect on the milestones we have made along this journey, referring to which methodologies were used and what were the major findings and conclusions. After that, we will discuss the existing limitations within this context, not only in this dissertation but in the EV field overall, and the future work that needs to be done in order to surpass some of these limitations. Finally, we will highlight this work's contribution to the electric vehicle field theme, especially in the context of a route planner.

6.1 Summary of the research

As EVs are only going to grow in popularity over time, so the whole ecosystem around them also needs to grow. Currently one of the major challenges is route planning since even the more popular services that exist may not be ideal for every route and sometimes need human intervention after the route is planned. The effectiveness of a route planner not only affects the user's convenience but also directly impacts its journey since if something goes wrong during it, the user's vehicle can reach out of charge with no nearby solutions to recharge it. This research targeted to understand the intricacies, limitations, and complexities that make EV route planning a complex work.

During this dissertation, we were motivated and had a point of comparison by solutions like A Better Route Planner (ABRP), which represents the current landscape of route planning for EVs since it is the most used and optimal solution available nowadays. While a lot has

been done over the years by the service, perfection has not been achieved yet. By developing a route planner system in this dissertation from scratch, we were able to understand what are the challenges and complexities that can impact directly the accuracy of an EV route planner.

Though the study was primarily focused on uncovering all the challenges that make EVs route planning a complex problem, it also had a secondary objective of developing a system based on a microservices architecture. This system provides a great starting point for future researchers and for the whole EV society in general, on which they can have a solid base, pull out any particular components that they find useful, or even integrate this system into their own existing infrastructures with relative ease. All of this is possible due to the choice of using this type of software design, which is flexible, independent, and scalable.

Our primary work involved extensive research into various areas of the electric vehicle field, however, there were several primary findings that emerged from our research, which include the available solutions nowadays for route planners, the necessary connection with the charging stations, the route planning strategies and the decision-making algorithms essential for choosing the charging stations throughout the route.

Although the real-world testing was not within our scope, we have performed several local tests for which there are results that include: response times, the system's capacity to keep up under stress conditions, and a final comparison with the existent solutions.

6.2 Limitations and challenges

The implemented system has allowed the route planner to be flexible and scalable through the use of a microservices architecture, while still providing accurate and good results overall. Despite the implementation and results shown, several number of limitations were found during the process:

1. **Lack of scientific resources**: One of the greatest challenges was the lack of academic and scientific resources in the domain of EV, especially in the context of route planning. Since this is a recent sector that is gaining popularity over time, there aren't many detailed academic studies and/or articles that might be referred to for our research and development.

2. **Proactive Strategy**: While having many advantages and being more efficient than the conventional reactive strategy, it does not deal with the dynamic changes and unexpected circumstances during the route such as traffic congestion, bad weather conditions, or road closures.

3. **Data Access Challenges:** Finding open-source datasets of the charging stations was also a major challenge during the development of this dissertation. Most of the charging station operators have exclusive contracts and are not open-source data. Also, having static datasets can be a good idea, however, it is not an optimal solution since this type of system requires updated data from the charging stations.

4. **Solo development:** The vast complexity of the domain required choices to be made regarding what elements should be principally covered in this dissertation. While our focus was on charging station data retrieval, decision-making methodologies, and algorithms, there are other factors such as weather variations, road elevation, and individual driving styles that affect an EV's range that consequently can affect the planned route directly. Developing such a system single-handedly has reiterated the challenges faced by independent researchers and developers. It emphasizes the significance of collaborative development, especially when navigating projects of this magnitude. Every successful route planning software has a team of dedicated developers, data scientists, and analysts behind it who see no rest. They work together in a collaborative and complex process where the intricacies and subtleties involved by each person doing the work seldom get fully understood.

6.3 Future Work

Despite the results and the development, the system could be improved in several ways and a lot of new features can be implemented in subsequent work:

- A evolution of our system would be to switch over to a hybrid strategy. This would involve a lot of work such as vehicle integration, contracts with car manufacturers for data access, big infrastructural development and changes, contracts and agreements with charging station operators all over the world, and a system that is capable of handling and processing real-time data.
- The use of other external services like the Google Elevation API [64] can help enhance our system's decision-making by taking into account the elevation of the road throughout the route. By considering elevations while making predictions, the system could make better and more accurate estimations about the vehicle's range and its charging needs.
- An internal service responsible for retrieving detailed information about the charging stations should be developed to achieve higher accuracy and more responsiveness. This new service would deal with all of the connections made with the charging stations as a whole. By doing this, we can take all the information needed directly from the charging stations and centralize it into this new service, and with this, we can reduce our dependence on external services and have better processing times in general.
- Isolating SoC management into a separate service alone would streamline the whole energy prediction process. This service would monitor continuously the vehicle's energy levels, adjusting estimations in real-time and giving immediate feedback to route calculation service if a hybrid strategy was also implemented. Even on a proactive strategy alone, isolating SoC management into its own service, it becomes feasible to integrate advanced algorithms, machine learning models, and integration with other

external services such as Google Elevation API to further refine and improve the accuracy of SoC predictions according to the route. Also, on future versions of this system, an integration with some weather API could be made into this internal service. All of this would not only optimize the route charging stops and calculations but also offer a more personalized and efficient final route for the driver.

- With a great data load, especially if we have an internal charging station service implemented, we can upgrade the system to more precise decision-making methodologies when picking charging stations. One possible way would be for machine learning algorithms to come into play here as already referenced before in Chapter 2. Machine learning models are able to learn using huge amounts of data together with complex patterns, and this can be used to select optimal charging stations throughout our planned route.
- To improve system performance and reduce latency in general, we should consider hosting services like OSRM using Docker containers locally instead of using its external API as recommended by OSRM. [41]

Although the currently deployed system provides an already efficient platform for planning EV routes, there is a lot of room for improvement. From including more holistic strategies to incorporating various external services and refining internal mechanisms, there is ample scope for accuracy and efficiency enhancements, however, these changes aren't as simple as they look as we have already seen before. Moving into machine learning methodologies may drastically change the way decisions are taken making it more intuitive and responsive for the system and the final user.

Researchers and developers, with the help of this project's algorithms and methodologies, can not just expand and modify it but also take the project as a foundation for their own projects.

Our initial results have been highly promising, but there is a lot of room for improvement, and our system also has its own set of limitations like all the current solutions that are in the current EV market. As electric vehicles become common and their use is way more common in our daily lifestyle, systems like the one we built, which provide essential tools for the EV user and give it an overall good user experience certainly play an imperative role in the migration process from combustion engine vehicles to electric ones.

6.4 Contributions

This article proposes a highly scalable system for the world of EV route planning, providing an in-depth analysis of all the intricacies associated with this area while simultaneously forging a new system designed to try to tackle these issues. Our journey during this research has been intense and enlightening, so here we will detail how many ways this dissertation is contributing to this field

As the only and main researcher for this project, my responsibilities were not only constrained to theoretical exploration and research. Right from its conception to its final implementation, I managed every detail of the lifecycle of the project, which included the architecture, algorithms, methodologies, plans, and infrastructure. My professional experience as a software engineer in the "miio" [65] app proved to have a lot of value in both the practical and theoretical aspects of this dissertation.

When comparing our solution to the existing ones, it is hard to compare the methodologies, algorithms, and other internal aspects, since these systems are not open-sourced and do not openly share how they work. However, the introduction of the hybrid strategy in this dissertation is something that is not used by any of the already existing solutions and can be considered in the future.

Taking into account the issues of data collection in this context, where private charging station information is involved and any of the operators does not provide it in a free way, the system had to depend on the OCM API. However, the data regarding the EVs needed to get the total range and the manufacturer details, was successfully retrieved by the open-source platform EVDatabase [51].

The research was conducted in collaboration with the University of Aveiro (UA) as part of my Masters degree in Informatics Engineering. In addition, "miio," [65] an EV application also provided constant support and collaborated to ensure that this research remained grounded on real-world applicability as well as its relevance to the industry.

In summary, this dissertation shows potential advances that can be made in EV route planning while at the same time highlighting all the potential challenges and limitations that current solutions have to go through to have a better and more accurate system.

References

[1] N. Dragoni, S. Giallorenzo, A. L. Lafuente, *et al.*, "Microservices: Yesterday, Today, and Tomorrow", in 2017. DOI: 10.1007/978-3-319-67425-4_12.

[2] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful web services vs. "Big" web services: Making the right architectural decision", *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*, pp. 805–814, 2008. DOI: 10.1145/1367497.1367606.

[3] S. Pettie and V. Ramachandran, "A Shortest Path Algorithm for Real-Weighted Undirected Graphs", *SIAM Journal on Computing*, vol. 34, no. 6, pp. 1398–1431, 2005. DOI: 10.1137/S0097539702419650.

[4] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, "From monolithic systems to Microservices: An assessment framework", *Information and Software Technology*, vol. 137, 2021, ISSN: 09505849. DOI: 10.1016/j.infsof.2021.106600.

[5] G. Hill, O. Heidrich, F. Creutzig, and P. Blythe, "The role of electric vehicles in near-term mitigation pathways and achieving the UK's carbon budget", *Applied Energy*, vol. 251, Oct. 2019, ISSN: 03062619. DOI: 10.1016/j.apenergy.2019.04.107.

[6] T. T. Lie, K. Prasad, and N. Ding, "The electric vehicle: a review", *International Journal of Electric and Hybrid Vehicles*, vol. 9, p. 49, Jan. 2017. DOI: 10.1504/IJEHV.2017.10003709.

[7] K. N. Genikomsakis and G. Mitrentsis, "A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications", *Transportation Research Part D: Transport and Environment*, vol. 50, pp. 98–118, Jan. 2017, ISSN: 13619209. DOI: 10.1016/j.trd.2016.10.014.

[8] Y. Lu, J. Gu, D. Xie, and Y. Li, "Integrated Route Planning Algorithm Based on Spot Price and Classified Travel Objectives for EV Users", *IEEE Access*, vol. 7, pp. 122 238–122 250, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2937910.

[9] R. Raeesi and K. G. Zografos, "The electric vehicle routing problem with time windows and synchronised mobile battery swapping", *Transportation Research Part B: Methodological*, vol. 140, pp. 101–129, 2020, ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2020.06.012.

[10] M. Söylemez, B. Tekinerdogan, and A. Kolukısa, "Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review", *Applied Sciences*, vol. 12, p. 5507, May 2022. DOI: 10.3390/app12115507.

[11] K. Poornesh, K. P. Nivya, and K. Sireesha, "A Comparative Study on Electric Vehicle and Internal Combustion Engine Vehicles", in *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, 2020, pp. 1179–1183. DOI: 10.1109/ICOSEC49089.2020.9215386.

[12] J. Wang, I. Besselink, and H. Nijmeijer, "Electric Vehicle Energy Consumption Modelling and Prediction Based on Road Information", *World Electric Vehicle Journal*, vol. 7, pp. 447–458, Sep. 2015. DOI: 10.3390/wevj7030447.

[13] A. Donkers, D. Yang, and M. Viktorović, "Influence of driving style, infrastructure, weather and traffic on electric vehicle performance", *Transportation Research Part D: Transport and Environment*, vol. 88, p. 102 569, 2020, ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2020.102569.

[14] S. S. Bhurse and A. Bhole, "A Review of Regenerative Braking in Electric Vehicles", in *2018 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, 2018, pp. 363–367. DOI: 10.1109/ICCPEIC.2018.8525157.

[15] O. Juhlin, "Modeling of Battery Degradation in Electrified Vehicles", Master of Science Thesis in Electrical Engineering, Linköping University, Department of Electrical Engineering, Division of Vehicular Systems, SE-581 83 Linköping, Sweden, 2016.

[16] H. Helms, M. Pehnt, U. Lambrecht, and A. Liebich, "Electric vehicle and plug-in hybrid energy efficiency and life cycle emissions", *18th International Symposium Transport and Air Pollution*, Jan. 2010.

[17] G. Wager, M. McHenry, J. Whale, and T. Braunl, "Testing energy efficiency and driving range of electric vehicles in relation to gear selection", *Renewable Energy*, vol. 62, pp. 303–312, Jul. 2013. DOI: 10.1016/j.renene.2013.07.029.

[18] D. George and S. P, "Driving Range Estimation of Electric Vehicles using Deep Learning", in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2021, pp. 358–365. DOI: 10.1109/ICESC51422.2021.9532912.

[19] A Better Routeplanner AB, *A Better Route Planner (ABRP)*, Accessed: 2023-01-05, 2023. [Online]. Available: https://abetterrouteplanner.com.

[20] *ChargePoint: EV Charging Stations*, https://www.chargepoint.com/, Accessed: 2023-06-04.

[21] *PlugShare - Find Electric Vehicle Charging Locations Near You*, https://www.plugshare.com/, Accessed: 2023-01-06.

[22] *ChargeMap*, Accessed: 2023-12-17. [Online]. Available: https://chargemap.com/.

[23] H. S. Das, M. M. Rahman, S. Li, and C. W. Tan, "Electric vehicles standards, charging infrastructure, and impact on grid integration: A technological review", *Renewable and Sustainable Energy Reviews*, vol. 120, Mar. 2020, ISSN: 18790690. DOI: 10.1016/j.rser.2019.109618.

[24] Z. Garofalaki, D. Kosmanos, S. Moschoyiannis, D. Kallergis, and C. Douligeris, "Electric Vehicle Charging: A Survey on the Security Issues and Challenges of the Open Charge

Point Protocol (OCPP)", *IEEE Communications Surveys and Tutorials*, vol. 24, pp. 1504–1533, 3 Sep. 2022, ISSN: 1553877X. DOI: 10.1109/COMST.2022.3184448.

[25] *MOBI.E - Entidade Gestora da Rede de Mobilidade Elétrica*, Accessed: 2023-02-06, 2023. [Online]. Available: https://www.mobie.pt/.

[26] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study", Jan. 2016, pp. 137–146. DOI: 10.5220/0005785501370146.

[27] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 1st. O'Reilly Media, Feb. 2015, p. 280, ISBN: 978-1491950357.

[28] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture", *IEEE software*, vol. 33, no. 3, pp. 42–52, 2016.

[29] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: A systematic mapping study", *CLOSER 2018 - Proceedings of the 8th International Conference on Cloud Computing and Services Science*, vol. 2018-January, 2018. DOI: 10.5220/0006798302210232.

[30] Microsoft, *Microservices architecture style*, Accessed: 2023-02-22, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices.

[31] J.-A. Orozco and J. Barceló, "Reactive and Proactive Routing Strategies with Real-Time Traffic Information", *Procedia - Social and Behavioral Sciences*, vol. 39, pp. 633–648, 2012, Seventh International Conference on City Logistics which was held on June 7- 9,2011, Mallorca, Spain, ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2012.03.136.

[32] S. Ichoua1, M. Gendreau, and J.-Y. Potvin, "Planned route optimization for real-time vehicle routing", in *Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies*, V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, and I. Minis, Eds. Boston, MA: Springer US, 2007, pp. 1–18, ISBN: 978-0-387-71722-7. DOI: 10.1007/978-0-387-71722-7_1.

[33] T. L. Saaty, *The analytic hierarchy process : planning, priority setting, resource allocation*. New York; London: McGraw-Hill International Book Co., 1980, ISBN: 0070543712978007054371 3.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd. The MIT Press, 2001, ISBN: 0262032937.

[35] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications", in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857.

[36] B. Mahesh, "Machine Learning Algorithms-A Review", *International Journal of Science and Research*, 2018, ISSN: 2319-7064. DOI: 10.21275/ART20203995.

[37] *Open Charge Map API Documentation*, Accessed: 2023-02-25, 2023. [Online]. Available: https://openchargemap.org/site/develop/api.

[38] *Python Version 3.9*, https://www.python.org, Accessed: 2023-03-01.

[39] S. Ramírez, *FastAPI*, 2018. [Online]. Available: https://fastapi.tiangolo.com.

[40] MongoDB, *MongoDB*, 2009. [Online]. Available: https://www.mongodb.com.

[41] *Project OSRM*, Accessed: 2022-11-10. [Online]. Available: http://project-osrm.org/.

[42] *OpenStreetMap*, Accessed: 2022-11-10. [Online]. Available: https://www.openstreetmap.org/.

[43] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks", May 2008, pp. 319–333, ISBN: 978-3-540-68548-7. DOI: 10.1007/978-3-540-68552-4_24.

[44] D. Luxen and C. Vetter, "Real-time routing with OpenStreetMap data", in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '11, Chicago, Illinois: ACM, 2011, pp. 513–516, ISBN: 978-1-4503-1031-4. DOI: 10.1145/2093973.2094062.

[45] Google, *Encoding and Decoding Paths*, Accessed: 2023-02-10, 2023. [Online]. Available: https://developers.google.com/maps/documentation/utilities/polylinealgorithm?hl=en.

[46] *Polyline library*, Accessed: 2023-01-05. [Online]. Available: https://pypi.org/project/polyline/.

[47] Eric W. Weisstein, *Haversine – from Wolfram MathWorld*, Accessed: 2023-02-25, 2023. [Online]. Available: http://mathworld.wolfram.com/Haversine.html.

[48] M. Kennedy, *Distance on a Sphere: The Haversine Formula*, Accessed 2023-02-26], 2011. [Online]. Available: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128.

[49] S. O. User, *Haversine formula in Python (Bearing and distance between two GPS points)*, Accessed: 2023-02-25, 2011. [Online]. Available: https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-two-gps-points.

[50] S. Mateen, M. Amir, A. Haque, and F. I. Bakhsh, "Ultra-fast charging of electric vehicles: A review of power electronics converter, grid stability and optimal battery consideration in multi-energy systems", *Sustainable Energy, Grids and Networks*, vol. 35, p. 101 112, 2023, ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2023.101112.

[51] *EV Database*, Accessed: 2023-03-04, 2023. [Online]. Available: https://www.ev-database.org/.

[52] J. Heer, M. Bostock, and V. Ogievetsky, "A Tour Through the Visualization Zoo", *Communications of the ACM*, vol. 53, no. 6, pp. 59–67, 2010. DOI: 10.1145/1743546.1743567.

[53] *Folium*, 2013. [Online]. Available: https://python-visualization.github.io/folium/.

[54] A. K. Adrien Treuille *et al.*, *Streamlit*, Python Library, 2018. [Online]. Available: https://streamlit.io.

[55] V. Agafonkin and contributors, *Leaflet*, JavaScript library for interactive maps, 2011. [Online]. Available: https://leafletjs.com.

[56] *Streamlit Folium*, Accessed: 2023-03-18, 2023. [Online]. Available: https://folium.streamlit.app/.

[57] T. pandas development team, *pandas: Python Data Analysis Library*, Accessed: 2023-03-18, 2023. [Online]. Available: https://pandas.pydata.org/.

[58] T. N. community, *NumPy – Numerical Python*, Accessed: 2023-02-18, 2023. [Online]. Available: https://numpy.org/.

[59] *Google Maps API*, Accessed: 2023-03-19. [Online]. Available: https://developers.google.com/maps.

[60] P. T. Inc., *Plotly*, 2021. [Online]. Available: https://plotly.com/.

[61] B. D. Team, *Bokeh*, 2021. [Online]. Available: https://bokeh.org/.

[62] I. Docker, *Docker*, Software. Available at https://www.docker.com, 2013.

[63] G. Rodolà, *psutil*, Accessed: [2023-05-26], 2022. [Online]. Available: https://pypi.org/project/psutil/.

[64] Google, *Google Elevation API*, Accessed: [2023-05-28], 2021. [Online]. Available: https://developers.google.com/maps/documentation/elevation/start.

[65] Miio, *Miio: Application for Electric Vehicles*, Accessed: 2023-06-29, 2019. [Online]. Available: https://www.miio.pt/.