Smart navigation system for electric vehicles charging

Francisco Antonio Peña Pérez

A Thesis presented for the degree of
Master by Research in Engineering

Department of Engineering
University of Durham
England

September 2019

Dedicated to

My parents for always supporting me to achieve my goals

Abstract

In the present time, there is still a lack of popularity in the use of electric vehicles, because of the actual disadvantages that they have. For this work presents the process of research and development of a web based application with the main purpose of helping Electric Vehicle owners decide the Charging Station that, by selecting it to go and charge their vehicles, represents the lowest cost in time or money (depending on their priorities) when they need to go to charge their electric vehicles and to give them less time or energy consuming route to follow in order to arrive to the charging station selected. This, to reduce the concern of the users about if the can or not arrive to a charging station.

To do this, the application has been developed with several features to help the users. First, the application has the feature of being accessed from multiple type of devices. Second, the application has the feature of detecting the users locations using Global Positioning System. Third, the application has the ability to find the charging stations and their coordinates that are near to the users. Fourth, the application has the ability to formulate the route with the lowest time or energy cost between the users locations and the charging stations. Fifth, after creating all the routes, the application shows the users the parameters of every route and charging station. Sixth, the application has the ability to let the users decide the priority to select the charging station. Seventh, the application let the users decide the battery percentage that they want their vehicles to have after charging them.

This application was created using mostly Javascript language, Expressjs as the framework and for the user interface jQuery. Moreover, MongoDB and PosgreSQL were used as databases. Furthermore, some web services like Amazon Web Services were used for server hosting, OpenStreetMap for obtaining GeoSpatial data, Open Charging Map for obtaining charging stations coordinates and data and Fuel Economy for obtaining vehicles data were used to complement the application. For the route formulation, Dijkstra's algorithm and pgRouting was used.

Results indicated that the application can successfully recommend routes and charging stations to the users with a reduction of 90% of time needed against the less time consuming cheapest option when time is the priority and a reduction of 27

times the money needed for the fastest option when price is the priority. Meaning that the navigation system can successfully reduce the time or costs to adjust to the users necessities.

Declaration

The work in this thesis is based on research carried out at the Department of Engineering of Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

Thanks to the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) for its financial support through out my studies for a Master Degree. I would also like to show my gratitude to my supervisor Dr. Hongjian Sun for accepting me in this project and his feedback when I needed it. Finally, but not least thanks to my family and friends for their support through my studies through for this degree.

Contents

Abstract iii

Declaration v

Acknowledgements vi

1 Introduction 2

2 Literature Review 4

2.1 Introduction 4

2.2 Shortest Path Algorithms 4

2.2.1 Graph Theory 4

2.2.2 Shortest Path Algorithms 6

2.2.3 Commonly used Shortest Path Algorithms 8

2.3 Navigation Systems 13

2.3.1 Description and Workflow 13

2.3.2 Time calculation of the path 15

2.3.3 Energy consideration in navigation systems 17

2.3.4 Software and Hardware technology used in navigation systems 19

2.4 Research Gaps 20

2.5 Conclusions 21

3 Navigation System Architecture 22

3.1 Introduction 22

3.2 Navigation System Workflow 22

3.3 Data displayed to the users 24

3.4 System Infrastructure Technologies 28
3.5 Application Technology and Data 31
3.5.1 Map Technology and Data 31
3.5.2 Charging Stations Data 32
3.5.3 Electric Vehicle Data 33
3.5.4 Route Formulation Technology 34
3.6 Limitations and disadvantages 36
3.7 Chapter Summary 36

4 Results **38**

5 Conclusions **53**
5.1 Future Work 53

Bibliography **55**

List of Figures

2.1 Graph representation 5
2.2 Weighted Graph representation 7
2.3 Dijkstra process, (a) Initial state, (b) Final state 9
2.4 A* process, (a) Initial state, (b) Final state 11
2.5 Basic Navigation System Workflow 14

3.1 General Navigation System Workflow. Arrows indicate the direction of the communication 23
3.2 Navigation System Workflow. Arrows indicate the direction of the communication 24

4.1 Map displayed in application, (a) User location, (b) Zoom in and out, (c) No Sleep Button, (d) Vehicle Selection, (e) Battery Percentage Selection 39
4.2 Brands selection in the application 40
4.3 Models selection in the application 41
4.4 List of EV manufacturers in database 42
4.5 List of EV models in database 43
4.6 List of Electric Vehicles in database 44
4.7 List of Charging Stations ordered by Price 46
4.8 List of Charging Stations ordered ordered by time 47
4.9 Query showing roads table in PostgreSQL 49
4.10 Result of query showing list of steps for a route 50
4.11 Route image format 51

List of Tables

2.1 Comparison of the three algorithms for route formulations presented . 13

4.1 Comparison of fig 4.7 and 4.8 48

4.2 Comparison between order by time and price 52

Nomenclature

API Application programming interface

AWS Amazon Web Services

CSS Cascade Style Sheets

DOE United States Department of Energy

DOM Document Object Model

EPA United States Environmental Protection Agency

HTML HyperText Markup Language

JSON JavaScript Object Notation

kWh Kilowatt hour

Chapter 1

Introduction

The use of electric vehicles has been becoming more common as time passes. In 2015 the worldwide use of electric vehicles reached a million [1]. It is a good change using electric vehicles more because of the substitution of combustion-based vehicles with electric vehicles can be beneficial to the planet due to the need of reducing of carbon emissions [2, 3]. As it is important to reduce the greenhouse gases emissions many countries have started to focus and making plans for the reduction of these gases, such as European Union that has set the target of reducing a 40% of them by 2030 [4] and the Chinese government project called "ten cities thousand vehicles program" to promote the Electric vehicle technology [5].

However, electric vehicles still have important disadvantages compared to combustion engine vehicles that decrease the progress in their popularity. First, they take much more time to charge their batteries, depending on the charging type they can take around 8 hours if they use a normal charger or half an hour to get to 80% if they use a quick charger. Second, charging stations are not that common to find in every city [6]. Because of these disadvantages, it can be unattractive to customers to opt for the use of electric vehicles as making long trips can be complicated [7].

Therefore, to help the electric vehicles owners, some research has been conducted for creating methods to formulate the most efficient routes for electric vehicles. Still, most of them have been theoretical and have not been implemented [8, 9, 7, 6, 1]. On the other hand, there has been not much research about creating applications for electric vehicle routing [10, 11]. However, there is still more data that can be

delivered to the users and options that can be considered to make the their trip more efficient and personalized to their needs.

For these reasons, it is necessary for the electric vehicle's owners to consider the locations of the charging stations, how much time they would take to get to the stations, the energy required for the trip (as the battery's charge can deplete before arriving at the station), how much money they would spend considering the energy needed for the trip and the price of the energy of the charging station. Therefore, this thesis proposes the original development of a web-based navigation system to help electric vehicles users decide to which station to go to charge their vehicles and what route follow to arrive there.

The contribution of this thesis is to provide a navigation system capable of providing three main features that actual navigation systems are lacking. The first is to be flexible with the users' and let them decide their priorities when charging their vehicles. The second is to let the users use the navigation system from different devices and not be tied to just one or two. The third is to use real data from existing charging station and therefore create a system that can be used in real life.

This thesis will be structured in the following way. The second chapter will provide a literature review about previous work and definitions; the third chapter will describe the system's architecture; the fourth chapter will present the results; the last chapter will be conclusions.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, definitions, information about the history and current state of the shortest path algorithms and navigation systems will be presented. In order to do this, in the first section, first some bases of the graph theory will be introduced, then description about what shortest algorithms are and how do they work will be presented and third some frequently used algorithms will be introduced. Furthermore, in the second section, information related to navigation systems, their features and factors that they have to take in consideration, will be presented. For this, first the concept and workflow of the navigation systems will be explained. Second, it will be explained how the navigation system considers many factors to calculate the time and energy of the routes. Finally, in the third section, conclusions of this chapter will be presented including the gaps found in research.

2.2 Shortest Path Algorithms

2.2.1 Graph Theory

In this section, before actually introducing what the shortest path algorithms are, some definitions of graph theory will be described because it is necessary to have a comprehension of this field to understand how it can explain the structure of shortest

path algorithms.

Using graphs is a common method to represent the road network data. In a graph, every node can represent an intersection, structure or interest point and every edge can represent a road. A graph is represented as $G = (V, E)$, where V refers to all the vertices in the graph and E refers to all the edges (also called arcs) in the graph [12]. Every node v in the graph is represented as a point and every edge e is represented as a line connecting two nodes. Each edge has a cost value, this value represents the cost of traversing the edge. Furthermore, a tuple (u, v) is the link between the node u and the node v. Moreover, Degree in this context means the number of edges connected to a node v. Hence, a graph can be used to represent network data using nodes, edges and tuples.

Fig. 2.1 displays the representation of a Graph. The circles with letters represent the nodes of a graph and the lines connecting them represent the edges of the graph. In this example, the tuple AB means the section including node A, node B and the edge connecting them. Finally, the Degree of node A is two, as two edges are connected to it.

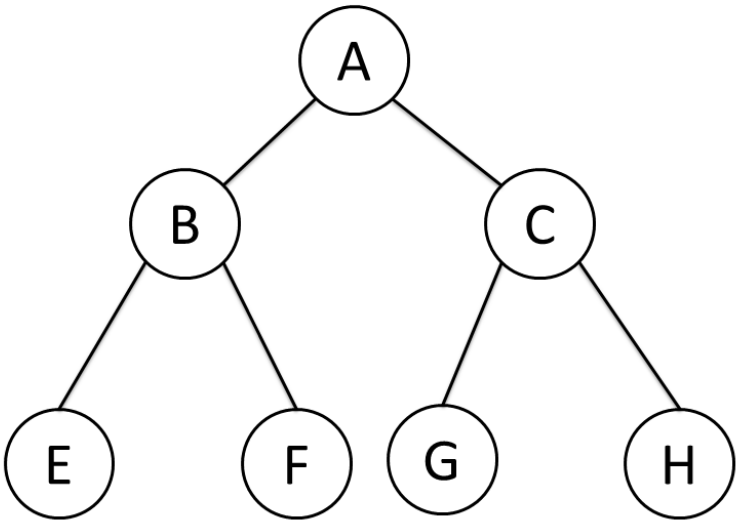


Figure 2.1: Graph representation

Additionally, there are two types of graphs used for shortest path algorithms. First, directed which is the type where the edges can only be traversed from node u to node v. And the second is undirected which is the type where the edges can be traversed from both directions whether it is from node u to node v or from node v to node u.

Moreover, with the previous definitions of graph structure, some terms commonly used in navigation systems using graphs are the following. It is considered a path P a sequence of nodes and edges that start from the source node s and end in the target node t. The weight of the path $w(P)$ means the sum of all the edges costs, giving, as a result, the total cost for traversing the path P.

Finally, with the bases obtained from the previous paragraphs, it can be proceeded to describe the shortest path algorithms.

2.2.2 Shortest Path Algorithms

Shortest Path Algorithms are the types of algorithms developed to find the shortest path from the source node n to the target node t in a graph with an input of (G, s, l), where $G = (V, E)$ is the graph, s is the source node and l is the length function [13].

Nevertheless, some basic concepts are needed to describe prior to comprehend the functioning of the shortest path algorithms. First, it is considered the shortest path, the path that has the minimum weight of all the possible paths between the source node s and the target node t [12], thus representing the weight of the shortest path as $d(s, t)$ as the minimum cost between source node s and target node t. Second, it is called edge relaxation, the process of exploring the edges and it is said that the node v has been reached through the process of edge relaxation of an edge (u, v). Third, when a node v is selected as part of the shortest path is called settled and the process of finding the shortest path is finished when the target node t is settled. Therefore, these concepts can help understand the functioning of the shortest path algorithms.

In Fig 2.2, it appears a weighted graph to help visualize what the Shortest Path is. In this figure, the nodes are labeled with letters and the weight of the edges are

labeled with numbers, these numbers represent the cost from one node to another. In this example, we want to know the shortest path between node B and node F. Node B is the source node and node F is the target node. For this, there are two options. The first path it to go from node B to node E and then to node F. The second path is to go from node B to node D, then to node D, then to node E and finally to node F. The sum of the costs of the edges of the first option is 4 and the sum of the costs of the edges for the second option is 5. Therefore, the Shortest Path is the first option, as it has the minimum cost.

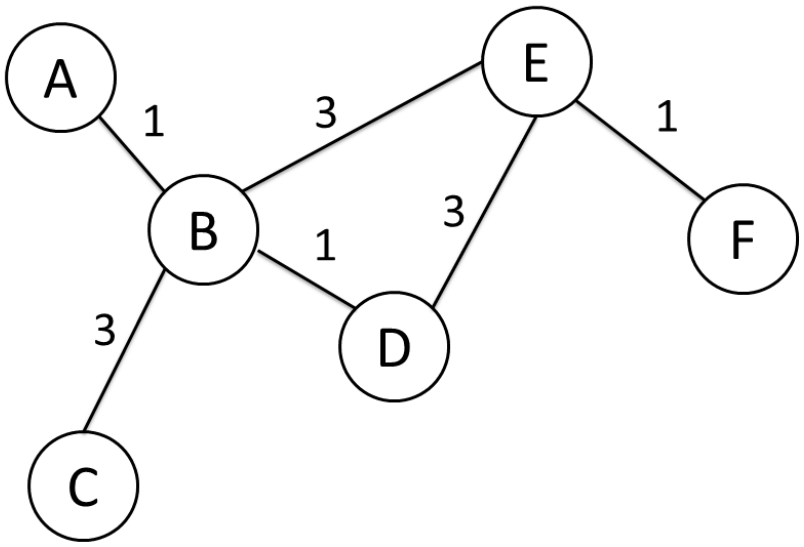


Figure 2.2: Weighted Graph representation

For the use of these types of algorithms, the process is usually divided into two phases [12]. First is the precomputing phase, used to speed up the process of finding the path by processing the original data to obtain useful information before the query phase. This phase is only carried out sparsely as the structure of the graph does not change frequently. However, the precomputing phase can be done frequently when the graph is enriched with dynamic data like traffic status or roads closed by maintenance. Moreover, some other causes of executing frequently the precomputing process can be caused by changing the metrics used for calculating

the cost of the path, for instance using travel time, energy or changing transport. The second is the query phase, in this phase, the algorithm is executed and, using the data obtained from the precomputing phase, the algorithm will follow the process to find the shortest route from the source node s to the target node t. Therefore, the process to use the shortest path algorithms is divided into the precomputing phase and the query phase.

2.2.3 Commonly used Shortest Path Algorithms

Dijkstra's algorithm

Dijkstra's algorithm is very old and frequently used for the formulation of routes [14]. The algorithm consists of finding the path that requires the lowest cost from one node to another in a graph, comparing every edge and selecting the ones that require the lowest cost from the source node s until it reaches the target node t. This algorithm can be used to formulate the routes for this thesis as it has low complexity and has been tested many times since its formulation.

According to Hector et al. [12], the process of Dijkstra's algorithm can be described as it follows:

First, Dijkstra's algorithm starts with all nodes been unreached and their distance is considered as unknown. An array D of tentative costs of the edges between the source node s and the target node t is created and the initial cost $D[s]$ set as 0. A pointer defines the source node $D[s]$ as the current node c.

Second, the cost of every edge that has not been settled adjacent to the current node c is inspected and a new distance is defined using the cost of the path from the source node s to the current node c plus the weight of the edge $w(c, v)$ connected to the current node c. If the result is lower than the previous value of $D[v]$, the value of $D[v]$ is changed to $D[v]$ plus the weight from c to v.

Third, the pointer for current node c is changed to the next unsettled with the lowest weight. After that, the current node c is considered as settled.

Forth, the algorithm is finished if the current node c is the target node t. If this is not the case, the process continues to the second step.

The result is the sequence of the process is a path made by recovering the stored nodes from the target node t to the source node s.

In fig 2.3, it can be observed the process of this algorithm represented applying it from node A to node D. In section (a), we have the initial state where unreached nodes are set with a cost of infinity. The process starts in node A, the costs from this node to B, C and D are set from infinity to 1, 3 and 10 respectively. Because the cost to node B is lower than to C and to D, B is set as the current node and the cost of D is set as 6. However, as 6 is bigger than the cost to C. Consequently, node C is set as the current node. Then, the cost to node D is replaced from 6 to 4 in node D. As the cost of D is the lowest unreached node, D is set as the current node and the process ends because D is the target node. The final state is represented in section (b) where the shortest path is indicated with red colour.

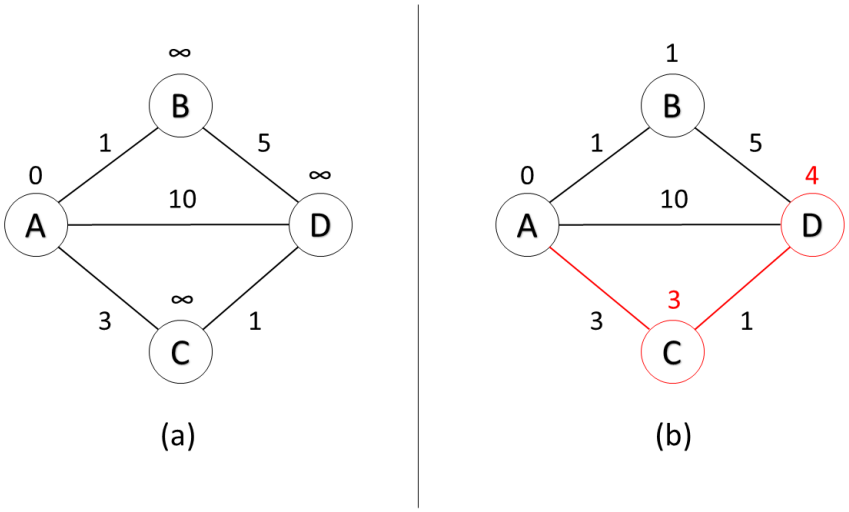


Figure 2.3: Dijkstra process, (a) Initial state, (b) Final state

A* Search algorithm

It is an algorithm that uses the Heuristic method, meaning that it is not 100% precise to find the shortest path. However, it proves to be faster to find the shortest path, with a certain range of imprecision, than other algorithms as it tries to examine the minimum amount of nodes possible to find the path with the minimum cost [15]. To do this, when deciding to settle a node, it uses an evaluation function to calculate

if the node leads to an shortest path. The steps of the algorithm are:

First, the source node s is marked as open and the evaluation function *f(s)* is applied to calculate the cost for the shortest path from the actual node n to the target node t.

Second, the open node n which lowest value of f is selected. If there are two with the same value the decision to choose between them is random.

Third, the algorithm is finished if the current node is the target node t and the current node n is marked as closed.

Fourth, the current node n is marked as closed and the successor operator is applied to n. The evaluation function is applied to every successor of n and every successor not already marked as closed is marked as open. If any successor of n is closed but the result of f is smaller now than it was when it was marked closed, it is marked as open. Go back to the second step.

The evaluation function consists in $f(n) = g(n) + h(n)$. Where $g(n)$ is the cost of the path from s to n, this should be calculated as the relaxation process proceeds. In the case of $h(n)$, it is the cost of the shortest path from n to t. To calculate $h(n)$, physical information is used to approximate the shortest possible distance between n and t, in the case of roads it is used the direct distance between n and t without considering the nodes in between.

In fig 2.4, the process of the A* search is represented, applying it to the search of the path from node A to node D. In section (a) we have the initial state. The estimated distance from every node to the target node D is represented as a number over each one. The distance from each of A neighbours is calculated by summing the edges costs plus the tentative designated cost of each node, giving 4, 5, 10 and 9 to nodes B, C, E and F respectively. Then B is selected as the current node as it has the lowest cost of all the neighbours. Then, the distance from A to following the path with node C to node D is calculated using the approximated cost of C that is 2. The result gives 5 for the route A-B-D. Node C is selected as the current node as the cost from A to C plus the estimated distance is 4, smaller than the previous path. Distance from C to D is evaluated and the cost of the path gives 4. Node D is considered the current node and the algorithm finishes. The final state

is represented in section (b) and the shortest path is indicated with red color.

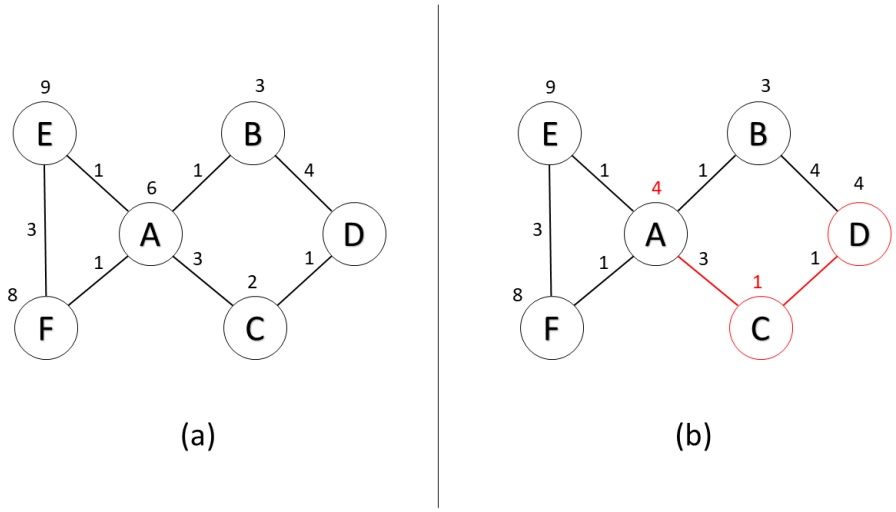


Figure 2.4: A* process, (a) Initial state, (b) Final state

Bidirectional search

This kind of search alternates between two shortest path algorithms working at the same time. The first algorithm starts from source node s and ends in target node t. The second, called backward search, starts from t and ends in s. If the graph is not directed only the cost from v to u should be considered for the backward search. The process finishes when both algorithms collide at the same node.

Some research has been conducted using this kind of process. Nicholson uses his own algorithm to find the shortest route while executing a bidirectional search [16]. Moreover, Andrew et al. made some experiments using a bidirectional search with the A* algorithm, the results indicated that the efficiency usually is better than the mean [17]. Therefore, evidence exists demonstrating the efficiency of this type of search.

Parallelization in shortest path algorithms

An alternative way to speed up the process of finding the shortest path is, with the use of multi-core processors, to execute multiple sections of the process at the same time using multiple threads.

Some research has been conducted regarding the parallelization of Shortest Path Algorithms and it has proved to have great results regarding efficiency. Jasika et al. research was conducted using 4 environments, some with multiple processors to use different cores to follow the path for different vertices with Dijkstra's algorithm, the results indicated that using parallelization the performance of the process of finding the shortest route incremented a 10% compared to the usual sequential way of implementing it [18]. Singla et al. conducted research using NVIDIA CUDA technology for Graphics Processing Units for graphics cards to parallelize the process Shortest Path Algorithms [19]. The experiment was conducted using a graph with over 2 million edges and the results indicated a speed-up of up to 5 times faster using parallel Dijkstra's algorithm than the sequential process. Moreover, experimenting with a graphs of over 23 million edges the speed up of the process was of 4 times. Therefore, the research indicated that the use of multiple cores to parallelize and speed-up the process was a great success.

Algorithms comparison

In table 2.1, it can bee seen a comparison between the three algorithms presented.

In terms of speed Dijkstra's has the lowest, A* is faster than Dijkstra's as it removed options by using estimated distance and Parallelization is the fastest of the three as it can implement simultaneous searches at the same time.

In terms of precision Dijkstra's has the best precision as it always finds the shortest route, A* has the lowest precision as it doesn't always finds the shortest route and Parallelization, depending of the algorithm implementations, can be as accurate as any of the other two algorithms.

In terms of resources required, Dijkstra's requires the lowest resources, A* requires more resources than Dijkstra's as it needs extra data sources and Parallelization requires the most resources as it uses much more processing power than the other two algorithms.

Algorithm	Speed	Precision	Resources Required
Dijkstra's	Low	High	Low
A*	High	Low	High
Parallelization	High	High	High

Table 2.1: Comparison of the three algorithms for route formulations presented

2.3 Navigation Systems

2.3.1 Description and Workflow

A navigation system is a combination of software and hardware that aids the user to find a route to follow to arrive at a destiny. Usually, the navigation systems display a map where the users can identify their location and decide their destiny. The navigation system displays instructions to the users about how to arrive at their destiny. These instructions can be visual, written or verbal.

In fig. 2.5 the typical workflow of a navigation system is displayed. Steps are indicated with numbers in parenthesis. In step 1, it starts with the roads database. This database is filled with nodes and every node represents a road or interest point. These nodes are records that contain useful information like distance, type of road, velocity limit and the like. Optionally, in step 2, there are other sources of information like other databases or web services where information about traffic, weather, inclination, etc. can be obtained. The data in these databases are extracted, in step 3, by the main application and then, in step 4, the application sends to the user device the map so the user can visualize it. After that, in step 5, the application obtains the coordinates of a start point and a target point. After that, in step 6, the application uses the data obtained from the databases to create a graph with the roads, relate the coordinates with nodes of the graph and then, in step 7, use an algorithm to formulate a path with the minimum cost. The minimum cost is indicated with red colour in the graph. After formulating the path, it is sent to the user device in step 8. Therefore, the navigation system workflow is usually followed in that order.

Considering the information above, the *algorithm* should be considered the core

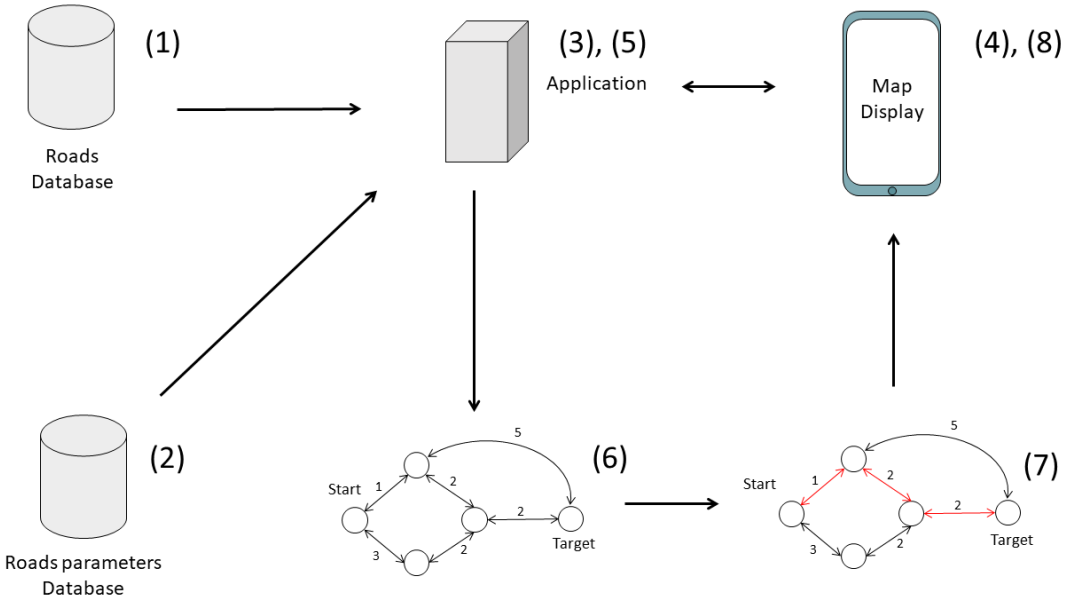


Figure 2.5: Basic Navigation System Workflow

of a navigation system, as it is the algorithm the part of the system that formulates the route to follow. The algorithm should consider the possible paths to follow from the start point (where the users is) to the target point (where the user should be at the end of the trip) and formulate the shortest path. However, the shortest path will be selected considering what the navigation system selects as the cost for the algorithm. Thus, when in past years, for combustion vehicles, the distance was normally used as the cost for these algorithms, for electric vehicles, there are many factors that the algorithm should consider to formulate a shortest path route e.g. electricity, time, money, CO2 emissions, etc. Therefore, the accuracy of the algorithm is very important to the navigation system.

Furthermore, it is important to know which algorithms are useful for navigation systems as they will be used for this thesis. A frequently used algorithm in research is Dijkstra's [4, 20, 21, 9, 6, 1] or some variation of it because this algorithm has proved for many years to find the lowest cost path. However, because of the high processing demand of Dijkstra's algorithm, other research [20, 10, 11] have opted for using A* search which is less processing costly and therefore faster. Nevertheless, A* search will not always find the route with the minimum cost by using a faster method. There are some other shortest path algorithms used in navigation systems like Bellman-

Ford [22] that is slower than Dijkstra's algorithm as it considers negative edges and LC search, a variant of Dijkstra's algorithm, used by Baum et al. to accept negative values as it is required for the consideration of battery recovery. Hence, it is more effective and common in navigation systems to use the Dijkstra's algorithm and A* Search to formulate routes.

As mentioned before, many factors are taken into account for the use of shortest path algorithms in navigation systems. Therefore, these will be described in the following sections of the chapter.

2.3.2 Time calculation of the path

One of the most focused metrics for navigation systems is the time. It is common in navigation systems to consider the shortest path as the one that consumes the lowest amount of time. However, there are many parameters that affect the amount of time required to go through a route for vehicles. Therefore, to calculate time, these parameters need to be taken into consideration.

An important parameter to measure the time that the vehicle would take to go through a section of the path is the speed limit of the section. This parameter is the core one to estimate the time of the route because it decides how much time the vehicle can take to go through the section in optimal vehicle and road conditions. However, as this estimation can only work in these optimal conditions there is a need for more parameters have to consider to complement adverse conditions.

A factor to consider when calculating the time travel of the trip is traffic, as this factor can reduce the capacity of the vehicle to travel through the road at the usual velocity and thus increase the time that would take to follow the path. There are two well-known ways to detect the traffic. One is to use traffic data from records obtained from some period of time and keep it in a database to use it afterward. A second one is to ask to web services that are continuously monitoring the traffic to obtain real-traffic information. Zhou et al. have mentioned using Google's and Yahoo's Application programming interfaces (APIs) to obtain traffic information of the roads and implement it on their navigation systems [1]. Zhihong et al. use a car to car communication system to measure the traffic by select the route that performs

the fewest number of switching between stop and start-up of the vehicles [6]. Guo considers using a technology named Intelligent Transport System to detect traffic congestion by sending real-time traffic information through FM radio broadcasts [23]. However, this technology is still in development and not usable for a software application. Hongming et al. made some tests comparing the time reduction in their system using traffic historical and real-time data [24]. The results indicated that even when the use of historical data was effective making a route that consumes less amount of time, the real-time data proved to be more effective at this task creating an even less time-consuming route. Cela et al. obtained real-time traffic data from the SYTADIN web service which provides data from Paris and the Ile-de-France region which is updated every 3 to 5 minutes [20, 21, 25]. This information represents how the focus is going more towards using real traffic data. However, some of this research states about the use of technology yet in development and the others use web services that require a frequent monetary cost. Nevertheless, it is evident that the use of traffic information helps to make a better route formulation by reducing costs.

The queue time is considered the amount of time that the users will spend waiting at the charging station to be able to start charging their vehicle. Guo et al. included the waiting time in the algorithm for their navigation system [23]. However, it was not stated how specifically they obtained this data and if it is possible at the present time to do this. Hongming et al. included it by using a mathematical model to estimate the waiting time, expecting infrared sensors installed at the charging stations entry to detect the number of vehicles and knowing the number of charging vehicles by sensors in the charge plugs [24] which is a good way to do it. Still, at the moment this type of monitoring is not used for the majority of the stations and thus not usable for an actual navigation system. Therefore, by analyzing the research at the moment, it can be concluded that the feature of obtaining the queue time is purely theoretical at the moment and not feasible for developing an actual navigation system.

Another factor that can be considered in the time calculation of a navigation system is the charging time, which refers to how much time is needed to charge the

vehicle to the desired level of battery charge which can vary between minutes and hours. Kobayashi et al. considered the charging time of the vehicle in the calculation of the total time of the route by calculating the type of charging depending on the type of the connector to differentiate the normal charging and the fast charging, this information is stored in a database as Charging Efficiency [9]. However, this information was hypothetical, not from real charging stations and therefore not useful for a real navigation system. Yang et al. equally considered the charging time in their research [5]. Nevertheless, the data for the charging time were simulated as well and not real data was used. Guo et al. considered the charging time as well, to include it in the total travel time. Moreover, they considered the possibility of using the service of swapping the battery instead which would require much less time. However, this service is used mainly for transport buses and not for regular electric vehicles. Thus, is not advisable to be considered for this project. Hence, it can be understandable the importance of the consideration of the charging time for the selection of the charging station as there is a great difference in time depending on the selected station.

Now with this information, it can be concluded that there are many factors that influence the amount of time that the users will spend depending on the station and route selected. Thus, it is important to take these parameters into consideration when calculating the time for the route. Moreover, it is also important to notice that at the present time, some of these parameters are not usually available and because of this they may need to be taken out of consideration.

2.3.3 Energy consideration in navigation systems

For feasibility purposes, is important to consider the remaining battery of the vehicle to decide if it is possible for it to arrive at the destination. Martin et al. considered the energy capacity of the vehicle's battery to detect if it is feasible to arrive at the station [10, 11]. Nonetheless, it is only considered the total capacity of the battery and not the remaining battery which can lead to a lack of precision on the futility to arrive to at the destination. Differently, Kobayashi et al. did consider the remaining battery charge of the vehicles to select stations in the way to the destination [9] which

resulted in a successful decision because the feasibility was always correct. Equally, Baum et al. research considered the vehicle remaining battery and energy necessary for the trip to consider the feasibility to arrive to the destination [8]. Hongming et al. considered it as well, selecting only the routes where the remaining battery was positive at the moment of arriving to the destination [24, 5]. Therefore, this evidence represents the importance of considering the remaining battery of the vehicle before recommending a route, as this may not be possible with the actual battery charge.

The battery recuperation factor is considered in some research [8, 10] to calculate the energy cost for a trip. It considers that as long as the electric vehicle is on the road, in some cases like going downhills, it can produce electricity and recover energy for its battery. For the purpose of this, some research considered altitude factor, for example using the NASA Shuttle Radar Topographic Mission data [10], to know when the vehicle is going downhill and as a result recovering energy. Baum et al. mentioned to handle the battery recuperation factor as a negative cost (discarding some algorithms like Dijkstra's). However, because of the time and resource limits of this thesis, this factor is not considered in the application of this thesis. Nevertheless, in a future version of this work, this factor should be considered as it would a more precise estimation of the battery charge at the end of the trip.

Some researchers have considered the road types to calculate the energy required for the trip. Liaw et al. decided to make a classification for driving styles depending on the type of road and assigned electric consumption profiles which were created using the data retrieved from a fleet of 15 electric vehicles and analysing it with fuzzy logic [26]. This classification gives the possibility of a better approximation of the energy required to follow a route. Adnane et al. calculated the energy of each segment of the route by considering the friction caused by the type of road giving different profiles of energy consumption depending on the type of road. Therefore, considering the road conditions can help to have a better approximation of the energy required for the route.

A seldom-used parameter in navigation systems for electric vehicles is the consideration of the weather. Zhou et al. mentioned the use of Google Maps and Yahoo! APIs to obtain weather data of the roads and estimate a mean energy consumption

[1]. In a different approach, Korosh et al. used weather data in their navigation system to consider how to regulate the vehicle AC system to reduce the energy consumption of the vehicle [22]. This, this evidence demonstrates ways to use weather information to reduce the energy consumption of the vehicle.

An important advantage to take into consideration the reduction of energy consumption is that the more energy spent on the trip would mean a rise in the price of charging as it would require to pay for more energy (which was consumed during the trip). Zhou et al. calculated the total cost of charging the vehicle by converting the distance of the route into energy and transforming it into money taking into account the charging station fee [1]. Similarly, Baum et al. convert the energy spent on the trip to money, also considering the recovering factor to reduce the amount of energy and thus money [8]. Yang et al. calculated the cost of charging using the energy spent during the trip, the remaining battery and they also considered different prices for fast and regular charging for a more precise calculation [24]. Therefore, the consideration of the energy consumption on the price should be considered as many users will benefit from cheaper routes and stations.

2.3.4 Software and Hardware technology used in navigation systems

Navigation systems need a way to identify the elements on the way from origin to destination. This means to take into account the roads' positions, connections and directions. Moreover, some other elements like structures, rivers, parks, etc. to help the user identify the surroundings. OpenStreetMap is constantly used for this matter in papers [27, 20, 21, 10, 22] as it is an open-source free to use. Furthermore, some navigation systems even consider elevation factor [27, 20, 21, 10, 11] which can be used to calculate increase and decrease in energy consumption. Thus, with this geospatial data, it is possible to identify the roads and intersections and therefore formulate routes for the vehicles.

For a navigation system, it is important to locate the destination of the users, in the case of this thesis the goal is to find charging stations. However, there is a lack of description in research papers about the sources of locations of charging

stations [8, 23, 9, 10, 11, 5, 1], mostly because their research is usually theoretical and not real data is used for charging stations. Though, knowing the location of real charging stations is an important feature for a real navigation system when real data must be used.

The interface of the navigation systems is very important, so the user can interact with it. Some researchers have opted for having their system installed in the vehicle [23, 22] which avoids the necessity of a third party device but includes the difficulties of compatibility and having to pay for an external product installed in the vehicle. On the other hand, some other researchers have opted for using tablets or smartphones to deploy their systems through applications [20, 21, 25, 10, 11, 24]. Using navigation systems through portable devices adds the possibility to use them without installing them on the users' vehicles, making them cheaper to use. Therefore, it is necessary to consider the necessities and background of the end-user before selecting an interface for the navigation system.

2.4 Research Gaps

Many features of the previously reviewed navigation systems are not realistic at the present time and have to be discarded. Calculating the queue time for navigation systems is not realistic at the moment as it requires that all charging stations calculate it and share that information. Moreover, most of the research conducted did not use real data to locate charging which is necessary for a real navigation system. Additionally, there is a lack of customization to the users needs as these navigation systems select only one option to optimize and do not let the users choose their priorities. Finally, there is also a lack of variety in regard of the device used, as the research reviewed is limited to only one device or operative system e.g. Android devices, iPads and iPhones or internal vehicle systems.

Thus, this work will have three main objectives to solve the gaps found by this literature review which are considered to be realistic at the time. The first objective of this work is to solve the lack of flexibility in navigation systems to let decide the user their priorities, this objective will be addressed in the third section of the next

chapter. The second objective is the solution to the lack of variety in the devices compatible with the navigation system, in the first section of the next chapter this problem will be addressed. The third objective is the use of real data from the charging station in the navigation system, this objective will be addressed in the second section of the next chapter.

2.5 Conclusions

In this chapter, information about the basics of shortest path algorithms has been presented, including graph theory, shortest path algorithm description and some commonly used algorithms. Furthermore, it has been reviewed many factors that navigation systems have taken into consideration for selecting the route with the lowest cost and how some of these are realistic to take into consideration at the present time and how some few are not.

The information presented suggests that each algorithm have advantages and disadvantages. Therefore, the decision of the selection of the algorithm was made considering the advantages and disadvantages to choose the one that is more suited for this thesis application. Considering that Dijkstra's algorithm can always provide the shortest path and consumes the lowest resources, it was chosen as the most suited for this thesis.

Furthermore, the research suggested that time and energy consumption considerations are important to consider when developing navigation systems. Moreover, there are many variables that influence the result value of these two factors. Thus, this data will be considered when calculating time and energy consumption in the navigation system.

Chapter 3

Navigation System Architecture

3.1 Introduction

This chapter will be dedicated to explaining how this navigation system is structured and how it works. To do this, the first section will describe the workflow of the navigation system of this thesis. The second section will include the technology used involved in the construction of the infrastructure of the system, mentioning the features needed for the infrastructure and the reasons for selecting them. The third section will describe the data obtained, including the map and vehicle data, and the technologies used for the main application, including the technology for the map and for formulating the routes. The fourth section will present the information that will be displayed to the users in order for them to be able to know the costs to charge their vehicles in each charging station and decide their priorities for the trip. Finally, a summary of the highlights of the chapter will be presented at the end.

3.2 Navigation System Workflow

Fig 3.1 represents a general workflow of the system process. As it can be seen, the flow starts with the user sending a request to the application server for a route to a charging station. Next, the application server asks for additional data to the databases and web services. After that, the application server uses all the data to generate a graph and, using Dijkstra's algorithm, formulates a route. Finally, the

route is sent to the user.

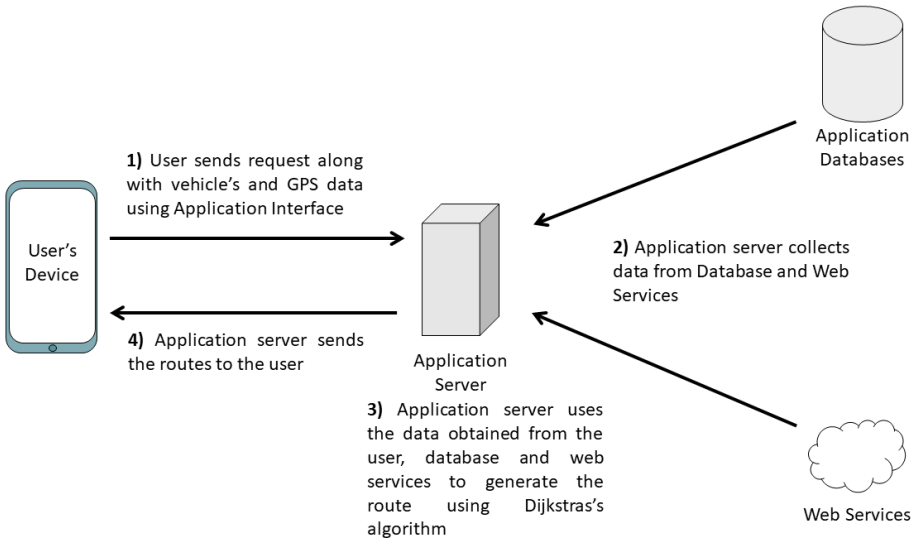


Figure 3.1: General Navigation System Workflow. Arrows indicate the direction of the communication

In fig 3.2, it can be seen, in a more specific manner, how the communication flows in the system. The process of the system starts when the users connect to the application server from their devices' web browser. After that, the application server sends the user-side application to the users' devices so the devices can be able of displaying the map and the users capable of interacting with the Application and Geospatial servers. Then, the users select and send their vehicles' manufacturers, models and coordinates (obtained by the devices GPS) to the Application Server. Next, the application server sends a request to a Web Service to obtain data, such as name, location and prices, of charging station near to the users' locations. The web service responds with the charging stations data. Afterwards, the Application Server, with the manufacturer and model selected by the users, searches for the vehicles' data in the Vehicles Database. After the petition is answered by the Vehicles Database, the Application Server uses the Roads Database data, in conjunction with the users' vehicles data, to calculate data from the userss location to each charging station (including energy and time required). When the process is finished, the Application Server sends the data to the users' Web Browser to display the data in the

users' devices.

Then, having the charging stations's data displayed in their devices, when the users select a charging station from the options appearing, the web browser sends the users' and the charging station coordinates to the Geospatial Server to show the route in the userss devices. After that, the Geospatial Server sends the coordinates to the Application Server asking route. The Application Server, using the coordinates, starts formulating the route using Dijkstra's algorithm and the Roads Database data. To do this pgRouting implements the algorithm with the Roads Database data. After finishing the process, the Application Server sends the route data (streets, avenues, distance) to the Geospatial Server. Next, the Geospatial Server creates an image of the route and sends it to the users' web browser. Finally, the route is displayed in the users' devices.

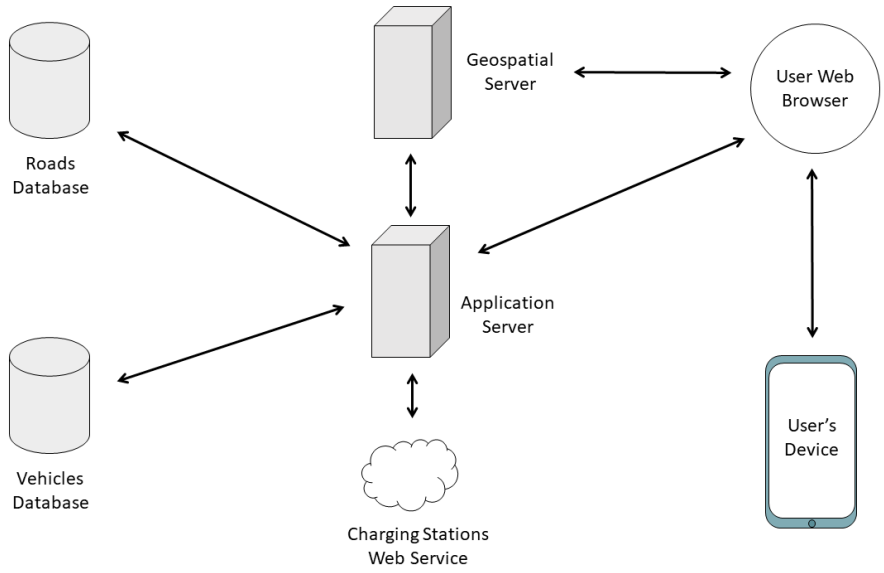


Figure 3.2: Navigation System Workflow. Arrows indicate the direction of the communication

3.3 Data displayed to the users

The first objective of this thesis is to solve the lack of flexibility in navigation systems to let decide the user their priorities. For this, the purpose of the application for this thesis is to give the users the data needed to be able to choose the charging

station that best suit them when selecting where to charge their vehicles. In order to do that, the application should be able to display in the users' devices screen a list of charging stations with the mentioned data ordered by the priority of the users. Therefore, this information should be obtained or calculated when the users make requests to be able to display it on the devices' screens.

The application should be able to display the charging station's name in the list to let the users identify the establishment when they arrive at it. This data can be obtained through the Open Charge Map's API [28]. After obtaining the data from the API, it should be sent to the users from the main application server and then displayed in the list. Thus, the users will have access to the name of the charging station when using the application.

The application should be able to show to the users how much distance they have to travel from their position to the charging station. For this purpose, a sum of all the sections of the route is made using the OpenStreetMap's geographic data and pgRouting to quantify the distance of the trip, this is done for every station near to the users' positions. Initially, the distance comes using miles as a unit. However, it is transformed to meters and if the amount is equals or greater than 1000 meters, it is transformed into Kilometres for reading facilitation for the users. As a result, the users will be able to access the distance from their position to every charging station that is presented in the application's list.

The application should display the time that would be required to arrive from their positions to the charging stations to the users. To do this, for every charging station near to the user, it was required to identify the time required to go through every section of the route to the charging station, this was possible using the geographic data from OpenStreetMap. Then, sum each of the sections to know how much time is required to go through all the route. This data is sent to the user-side of the application and displayed to the users in the list of charging stations. Thus, the users are able to know how much time would be required to arrive at each of charging stations from their positions.

The application should display how much time it would take to charge the vehicle from the battery level when arriving at the charging station to the battery percentage

required. To do this, it was required to know the battery capacity of the vehicle, the battery percentage at the start of the trip, calculate the energy required for the trip and with that information estimate the battery left at the end of the trip. Knowing the battery left, it can be calculated how much energy is required to get to the target battery percentage. After knowing how much energy is required from the charging station, it has to be calculated how much time it would require to transfer that energy to the vehicle. To calculate that time, it was required to know the capacity of the connector of the charging station. Fortunately, Open Charge Map [28] provides the connector data from its API. Finally, knowing the energy required and the connector transfer capacity it is possible to calculate the time required to charge the vehicle and send it to the users' devices. Therefore, the users will be able to know how much is required to charge their vehicles considering the energy lost during the trip.

The application should be able to provide the users with the information of how much would be the monetary cost of charging the vehicle at the charging station. In this matter, there are three possibilities considered. First, it can be a free service and no more calculations would be needed. Second, it can be a fixed price and thus, no more factors would be needed to be considered. Third, it can be a price per Kilowatt and for this, it is necessary to know how much energy would be required to charge the vehicle from its starting battery percentage to the target battery percentage. This data was calculated by obtaining the battery capacity of the vehicle from FuelEconomy [29], the battery percentage at the start of the trip introduced by the users, calculating the energy required for the trip considering the distance and the vehicle efficiency, and with that information calculate the battery left after arriving at the charging station. Knowing the battery left, it can be calculated how much energy is required to get to the target battery percentage. Finally, after obtaining the number of Kilowatts required, it should be multiplied by the fee which is obtained by the Open Charge Map API [28]. After that, the total cost would be sent from the application server to the user-side application in the users' devices. Therefore, 3 factors were considered to obtain the cost of charge the users' vehicle and the user would be able to obtain this information through the application.

The application should be able to inform to the users how much energy is required to make the trip from their position to the charging stations to know if it is viable for them to go to their selected charging station. To calculate this data, it is necessary to know the efficiency of the vehicles' motors. Fortunately, the motor efficiency is included in the FuelEconomy database [29]. Knowing the vehicles motors efficiency, the next step is to obtain the distance between the users' locations and the charging stations. To obtain the distance between the users' locations and the charging stations, pgRouting [30] was used to go over every edge between these locations and obtain the distance of every edge in the route. After obtaining the distances, a summation is made to obtain the total distance of the trip. The total distance is multiplied by the vehicles' efficiency factor and the result is the total energy required for the trip. This data is calculated in the application server and is sent to the user-side application in the users' devices. Therefore, the users will be able to know the amount of energy required for the trip before making it using this thesis application.

However, not all the users comprehend the energy required in terms of Kilowatt hour(kWh) and would prefer a more common metric for the energy required, for that reason an alternative should be displayed. The decision was made to display the energy required for the trip as a percentage of the battery. Therefore, if the users are not familiar with the units to measure energy, they would be able to know how much percentage of the battery is required to make the trip to the charging station and thus know if their vehicles have the energy required for it. To be able to display the battery percentage required, first the energy required is calculated as explained in the previous paragraph. Next, it is necessary to know the vehicle motor capacity which was calculated previously. Finally, with those two parameters, it is calculated the percentage of the vehicle battery capacity required for the trip and the result is sent to the user side of the application. Hence, the users will be able to comprehend how much energy is required to arrive to the charging stations even if they do not understand metrics to measure energy.

The application should be able to display a list of the charging stations near to the users displaying first the ones that are better for the users considering their preference. In order to sort them, first, the users need to decide their priority,

whether it is the cost of charging their vehicles or the time spent to charge them. After deciding the users' priority there are two options. The first option, if the users decide to give priority to the price for charging the vehicle, after obtaining the price for charging the vehicle as mentioned in previous paragraphs, the list is ordered by the minimum price first and the most expensive at last. If the price is the same, the charging stations with the same price are ordered by time to decide which one goes first. The second option, if the users decide to give priority to the time, after obtaining the travel time and charging time as mentioned before, these two parameters are summed for every station and the ones with the lower total time are displayed first. By default, the list is ordered by the price for charging the vehicle. Therefore, the users will be able to select their priority and the charging stations that more suit their preferences will appear first.

3.4 System Infrastructure Technologies

The second objective of this thesis is to build a navigation system capable of being used in different types of devices e.g. tablets, smart-phones and desktop computers. In order to do this, the solution opted in this thesis is to create a web application and thus use the web browser in every device to execute it and have access to the system.

For this, first, the application requires a mean to display to the users an interface to communicate and receive information, for this purpose a technology capable of this was investigated. HyperText Markup Language (HTML) 5 is a cross-platform language, meaning that a specific Operative System nor a specific device (which implies that it can be used from desktop computers, laptops, tablets, etc) is required to work with it [31]. It only needs a modern web browser to use HTML5 to process and display information. HTML5 provides new features from its predecessors useful for developing modern web applications. One of this features is the Canvas element, which is used for dynamically render visual content, a feature useful for the map in a navigation system that requires changing its contents e.g. charging station, routes and other elements. Therefore, HTML5 was selected as the base technology

responsible for displaying to the users the interface to interact with the application.

As mentioned in the previous paragraph, Canvas would represent an important feature for displaying the map to the user and there should be a manner to control what appears inside this element. In order to obtain this control over the contents of the canvas element, another technology would be necessary. The Canvas element can be controlled by the JavaScript technology to be able to draw and display contents inside of it [31]. Moreover, the purpose of JavaScript is to manipulate the DOM (Document Object Model) elements of a website [32] which would be useful for this thesis application as the Canvas element can be controlled with it and it is not the only element needed to be controlled as other DOM elements should be used for the users to introduce information about their vehicles. Thus, JavaScript should be used in this thesis application to interact with the Canvas and other DOM elements.

Furthermore, for this application to be able to display the elements correctly to the user, the use of CSS (Cascade Style Sheets) will be necessary. CSS allows developers to attach style to the elements of their HTML or XML documents by changing size, colour, fonts, transitions and other attributes [33]. Moreover, to be able to add some features that will be essential in this project, like responsive design or Canvas, the use of CSS would be necessary. Hence, it was decided the use of CSS on this application to fulfil the requirements for other features and make it easier to navigate through it.

Another factor to take in consideration is that, because the application is web-based, it is needed to be stored in a web server so it can be accessed from multiple devices through the internet. At the present time, AWS (Amazon Web Services) is constantly used in the field of web development as they are secure and scalable. AWS is a platform that provides cloud services including website storage [34]. Moreover, AWS website provides good quality documentation and tutorials for the purpose of learning how to use their products. Additionally, AWS provides a free period for students to test their products and get used to them. Thus, after some tests were conducted using an AWS Ubuntu instance, it was proven that their products were compatible with the technologies required for developing the application. Therefore, AWS was selected to be used for this thesis for the purposes of storing the application

on a web server and making it accessible to the users.

Another consideration to have, because of the application for this project is web-based and is expected to be used in different types of devices, is that it is important to include the feature of being responsive. This means that the application content can adapt its position and size depending on the resolution and aspect ratio of the device that is using it, otherwise the content of the application can appear too small to be visible or too big to fit in the screen and it would not be comfortable or efficient to use. Additionally, making a responsive application avoids the need for creating multiple instances of the application for every different device. Therefore, as a consequence of the previous statements the decision was made to include the feature of being responsive for the application of this thesis.

Additionally, an efficient approach to include the feature of responsive design to the application is with the inclusion of a framework. Materialize is a front-end framework which function is to include responsive styles [35] following material design principles [36] to speed up the development of software by using its components with custom styles, animation and transitions. Moreover, Materialize website includes documentation and examples to learn how to include the framework and its components. Furthermore, Materialize is open-source, which means that it can be included in this software without any problem. Thus, the framework Materialize was included in the development of this application thesis to facilitate the inclusion of responsive design without consuming much time.

Moreover, a good practice to develop software is to maintain version control and as a consequence in case of making errors and saving the code, it would be possible to go back to previous versions that do not contain the error without losing major problems. According to its website, Git is an open-source version control system that permits to maintain version control of software development having the features of multiple branches of development to test new ideas and the capability of going back from any change made [37]. Furthermore, the Git website offers wide documentation about the user of this software and good practices. Additionally, Github offers the feature of uploading for free your software online [38] for the purposes of easily install your software in another computer and also to have an up to date backup

in case of disaster. Besides, the Github website also has an extensive section for documentation in case it is needed during the development of the application. As a result, Git, in combination with Github, was chosen to be used in this project to be used as the version control technology as they have much documentation and having a version control has many advantages while developing software.

3.5 Application Technology and Data

The third objective of this thesis is to build a navigation system that uses real data from roads, charging stations, vehicles and roads to provide accurate values to the user. To do this, in this section, first is explained what data was used for the identifying and displaying elements in the map, like roads, building and other structures. Second, the charging stations data source is presented. Third, the selection and justification for vehicles data are described. The final part of this section explains how the route is formulated using the data obtained from the previous sources.

3.5.1 Map Technology and Data

An important feature considered in making the application that needs to display routes for navigation purposes was to be able to illustrate a map to the users so they can know where to go and what is their position in relation to the charging station. For this reason, before choosing a technology to use, research about what capabilities it would require was conducted.

First, since this is a web-based application, the technology used to develop this application should be compatible with web browsers. Documentation about how to implement it with front-end technologies such as HTML, JavaScript and CSS should exist to be able to learn how to use it and develop the application.

Second, the map technology should be able to display an environment easy for the user to understand the environment. It should be able to display roads, buildings, parks, lakes and other places that would be useful to the users to orientate in the region that they would travel.

Third, this technology should be flexible enough to let the developer add features to the map dynamically. The flexibility refers to the ability to draw the routes, mark the user's position and point the destination where the charging station is. Therefore, with the capabilities for the technology being selected, a selection for this technology can be continued.

In other to obtain the data to display in the map, OpenStreetMap was selected for this project thesis. Previous research has been made using this technology for routing purposes for Electric Vehicles to find the most optimum cost-effective route [11]. OpenStreetMap is a database with data obtained by a big community composed of local people that decided to support the project [39]. The project is open-source, therefore it is possible to access its data without having to pay large amounts of money that normally would have been the case. Furthermore, this technology fulfils the second capability previously described, as it contains a vast amount of data related to roads, buildings and other structures. However, it is necessary to complement this data with a technology able of displaying it and fulfil the other two capabilities.

The library OpenLayers was selected as a very useful technology to complement the OpenStreetMap data for this project's purposes. OpenLayers is an open-source library developed with JavaScript language, compatible with OpenStreetMap and easy to customize with technologies such as HTML, JavaScript and CSS [40]. Furthermore, it permits to display vector and markers on the map that would let the developer render the routes and charging stations. Therefore, OpenLayers was selected to complement OpenStreetMap's data as it would fulfil the first and third capabilities that were pending.

3.5.2 Charging Stations Data

Another concern for this project was to get the charging stations data, as the purpose of this project is to guide the user to charging stations. For this purpose research was conducted to identify a suitable database containing the data necessary for this project necessities. Information such as the location of the stations, price of the energy and service hours were necessary.

Research conducted to the finding of Open Charge Map which appeared as a suitable option for getting the charging stations data. Open Charge Map provides a service for applications that allow requests that return information in JSON (JavaScript Object Notation) format about the charging stations in a region specified. Such information contains data such as coordinates, price and address [28]. Moreover, its website provides documentation about the parameters that can be sent in the requests. Furthermore, this is free to use service, therefore it would save the necessity to pay for obtaining this data. Thus, because of its features, Open Charge Map service was selected as the source of data for charging stations in this project.

3.5.3 Electric Vehicle Data

Obtaining data about the users' vehicles is important for this project, as data like the vehicles' efficiency factor is needed to calculate the energy cost of the trip to the charging station. Therefore, in order to get the vehicles' data, two options were considered.

The first option was to install a device in the electric vehicle to get its information from the vehicle's integrated computer. However, for this method, it would require the users of the application to buy a product that would be installed in their vehicles. Furthermore, it would require to find a suitable device, compatible with the majority of the electric vehicles and make research in other to find ways to develop software for this device. Therefore, this option was discarded as it would require too many resources and time to develop.

The second option was to search for a database of electric vehicles that could provide the data required for this application to calculate the energy consumption and formulate the lowest cost route for the vehicle. Thus, no specialised device would be needed for the users to install in their vehicles and instead the users could select their vehicles from a list that appears in the application. The data would be retrieved from the database after the users select their vehicles from the list and the calculations would be made afterwards. Hence, this option was more suitable to appeal to a major quantity of users than the first option.

Consequently, for this project, the second option was selected as it would be more convenient for the users to just open and select their vehicles from a list than having to buy and install a device into their vehicle. Thus, research was conducted to find a suitable database from which to obtain the electric vehicles' data.

The research resulted in the finding of a good source for electric vehicles' data, it was found at FuelEconomy.gov. This website is maintained by the DOE (United States Department of Energy) Office of Energy Efficiency and Renewable Energy and the data is provided by the EPA (United States Environmental Protection Agency) [29]. From this website, an up to date database with specification about vehicles like the model, manufacturer, energy efficiency and others can be downloaded [29]. As a result, the database obtained from FuelEconomy.gov was selected for the use of retrieving data about electric vehicles for this project.

Nevertheless, after reviewing the data, it was found that there were some complications with the vehicle data obtained and adjustments were made to obtain some lacking information. First, the vehicle's motor capacity was not explicitly displayed in the data. However, in the same variable as the motor's name, it was included the energy capacity for the motor of each vehicle. Because of this, a script was created and implemented to separate the motors' capacity from the name of the motor automatically to each vehicle, to void doing it manually and thus be able to use this data.

3.5.4 Route Formulation Technology

For this project, two main factors were considered to decide the cost of the route. The first one is the time that would take for the user to travel across the route. The second one would be the monetary cost that would take to travel through the route instead than the distance that is common for combustion engine vehicles [6]. For the second factor, some researchers have considered that it would be necessary to consider the energy spent by the vehicle to formulate routes for electric vehicles as electric vehicles can lose energy very fast when changing between start-up and stop, especially in cities [6]. It was chosen for this application to let the users decide which factor that they want to take as a priority and consider the cost of the route

based on that selection. After deciding the factors to measure the cost of the route, research about algorithms for the routes formulation had to be conducted.

As mentioned before in the literature review chapter, for shortest path algorithms, most are based on Dijkstra's algorithm. Dijkstra's algorithm is based on finding the path with the minimum length between two nodes in a graph considering the cost from each node to another node [14]. Many researchers have used Dijkstra's algorithm before to manage the formulation of efficient routes for electric vehicles. Moreover, it provides the shortest path considering the factor selected as the cost. As this is an efficient and well-tested algorithm, it was selected as the base to the planning of routes for this thesis application.

As Dijkstra's algorithm was selected to be used for making the routes for this application, it was necessary to find a technology that permitted to implement this algorithm with the geospatial data obtained from OpenStreetMap. After the research was conducted, a library named pgRouting was found to be compatible with the data previously mentioned. The pgRouting library provides geospatial routing functionality using PostgresSQL databases (making it compatible with OpenStreetMap as it can be transferred to PostgresSQL databases), has a variety of algorithms available for routing including Dijkstra's and the cost parameter for the algorithms can be calculated dynamically (which leaves space to adapt the process for electric vehicles) [30]. After conducting extensive reading from the official documentation provided by pgRouting website [30] and making some tests with the laboratory equipment, it was decided that pgRouting would be able to use OpenStreetMap data for the purposes of formulating routes. Therefore, pgRouting was selected as the technology that would be used for this thesis to proceed with the development of the Electric Vehicle's Routing Application.

Despite the efficiency of the pgRouting library, it was detected afterwards that a way to communicate between the mentioned library and OpenLayers was necessary as geospatial data should not be transmitted in a conventional way as for example using NodeJS server which is used for this thesis application. Hence, research was conducted to find a technology capable of connecting these two libraries. Hinted by a workshop provided at the pgRouting website [30], a geospatial data server was

found named Geoserver. According to its website, this server allows to view, edit and share geospatial data [41], which is useful for the purpose of this work as it needs to share the OpenStreetMap's geospatial data, obtained through GEOFABRIK's servers [42], to OpenLayers and also the routes constructed using pgRouting. With the help of the documentation found at GeoServer's website [41] and pgRouting's workshop [30], some tests were conducted to check the compatibility between them and results were satisfactory. Therefore, GeoServer was selected as the software in charge to transfer data between pgRounting routes and OpenLayers to display routes through the users' browser.

3.6 Limitations and disadvantages

One of the main disadvantages of this architecture is that the users need their devises to be connected to the internet. The reason for this is that there is no map or application installed in the users' devices. Therefore, all the route formulation and calculations are made in the server. Thus, this application cannot be accessed without internet connection.

Moreover, all the data is calculated using the vehicles' technical information, roads data and charging stations data. Therefore, the information presented to the users may not be always be completely accurate. Factors such as the vehicles and roads lack of maintenance can influence the accuracy of the application. Therefore, the lack of real time updated data can decrease the accuracy of the data provided to the users.

3.7 Chapter Summary

This chapter has presented the navigation system structure and how it solves the objectives presented in the previous chapter. This was done by, first, being able to be used by many types of devices. Second, using real data for obtaining values of existent location, roads, charging stations and vehicles. Third, by giving information about the trip to the users and letting them decide their priority to reduce

when charging their vehicles (time or money). Forth, describing the limitations and disadvantages of this architecture. Therefore, with the knowledge of the architecture of the system, the next chapter will proceed to present the results obtained from this work.

Chapter 4

Results

Fig. 4.1 displays the map as it appears in the application. The map contains roads, parks, rivers and structures that will help the users to identify their position. Moreover, the roads, rivers, parks and other structures are drawn with different colours to make it easier for the users to distinguish them. Furthermore, the map has the option to zoom in or out (b) to make it easier for the users to identify the structures and names on the map. Therefore, the map in the application contains useful features for the users to orientate themselves and facilitate the elements inside of it. The location of the users (a) is displayed on the map as a blue filled circle. When the users start the web application dialogue box appears and ask their approval to send their location as it is used to search the charging stations that are near to them. Additionally, the location updates frequently on the map, thus as the users move through the route they can always know their actual location to take as reference. Hence, the users' location is a helpful addition for the application to help the users arriving at their destination. The screen contains drop-lists (d) to select the brand and model of the vehicle and selectors (e) for the remaining battery percentage and target energy percentage.

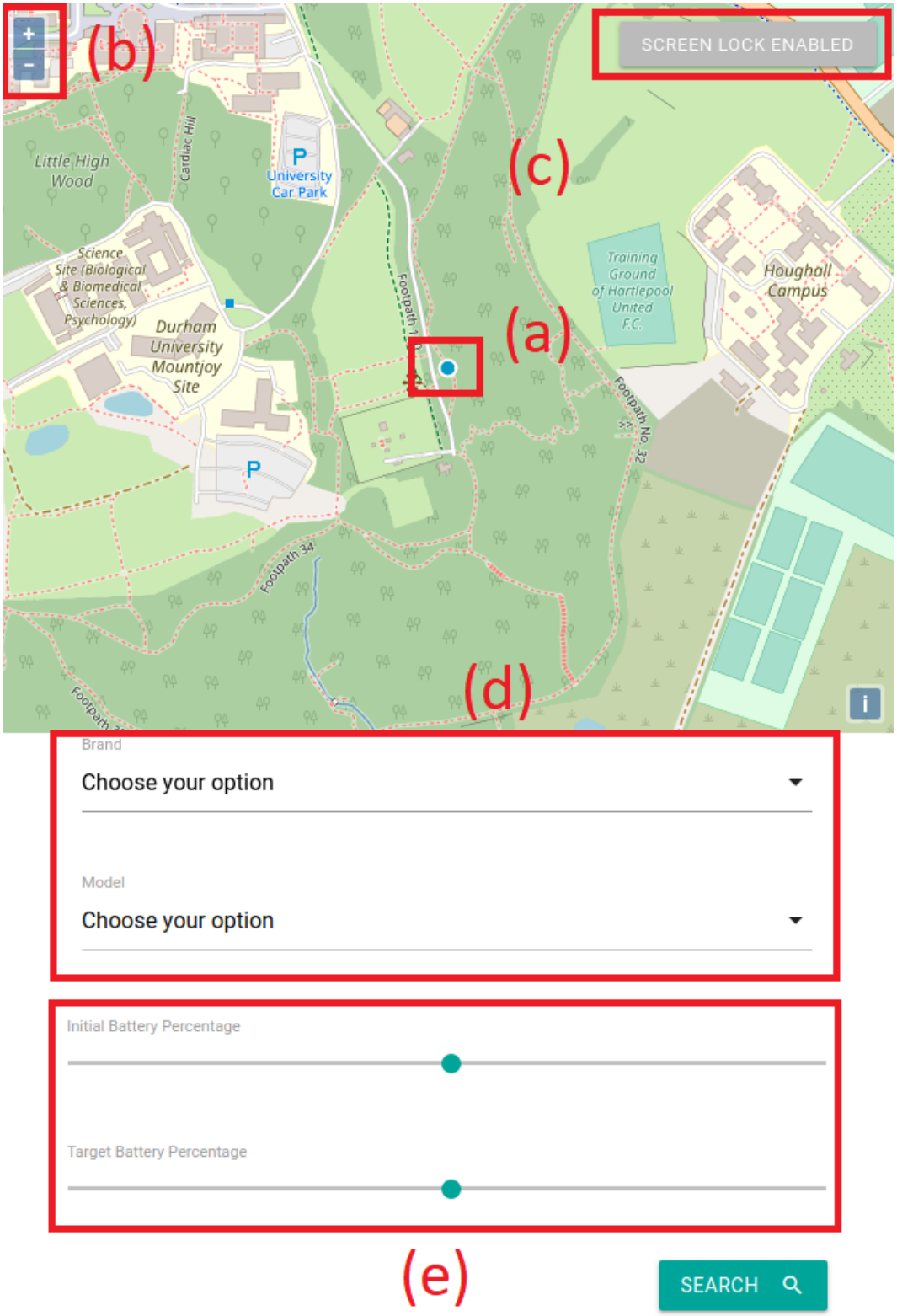


Figure 4.1: Map displayed in application, (a) User location, (b) Zoom in and out, (c) No Sleep Button, (d) Vehicle Selection, (e) Battery Percentage Selection

In fig. 4.2, we can see how the vehicle brand selection appears so the user can select it from a drop-down menu. The list is alphabetically ordered. Moreover, it is recovered from a MongoDB database which is presented in fig. 4.4. The selection of a manufacturer is required and it was developed like this for the purpose of speeding up the search of vehicles. Hence, the inclusion of the manufacturers' list is useful for the application to help the users find their vehicle faster. It works the same way for models in fig. 4.3, which is alphabetically ordered and the data is retrieved from the MongoDB database presented in fig. 4.5.

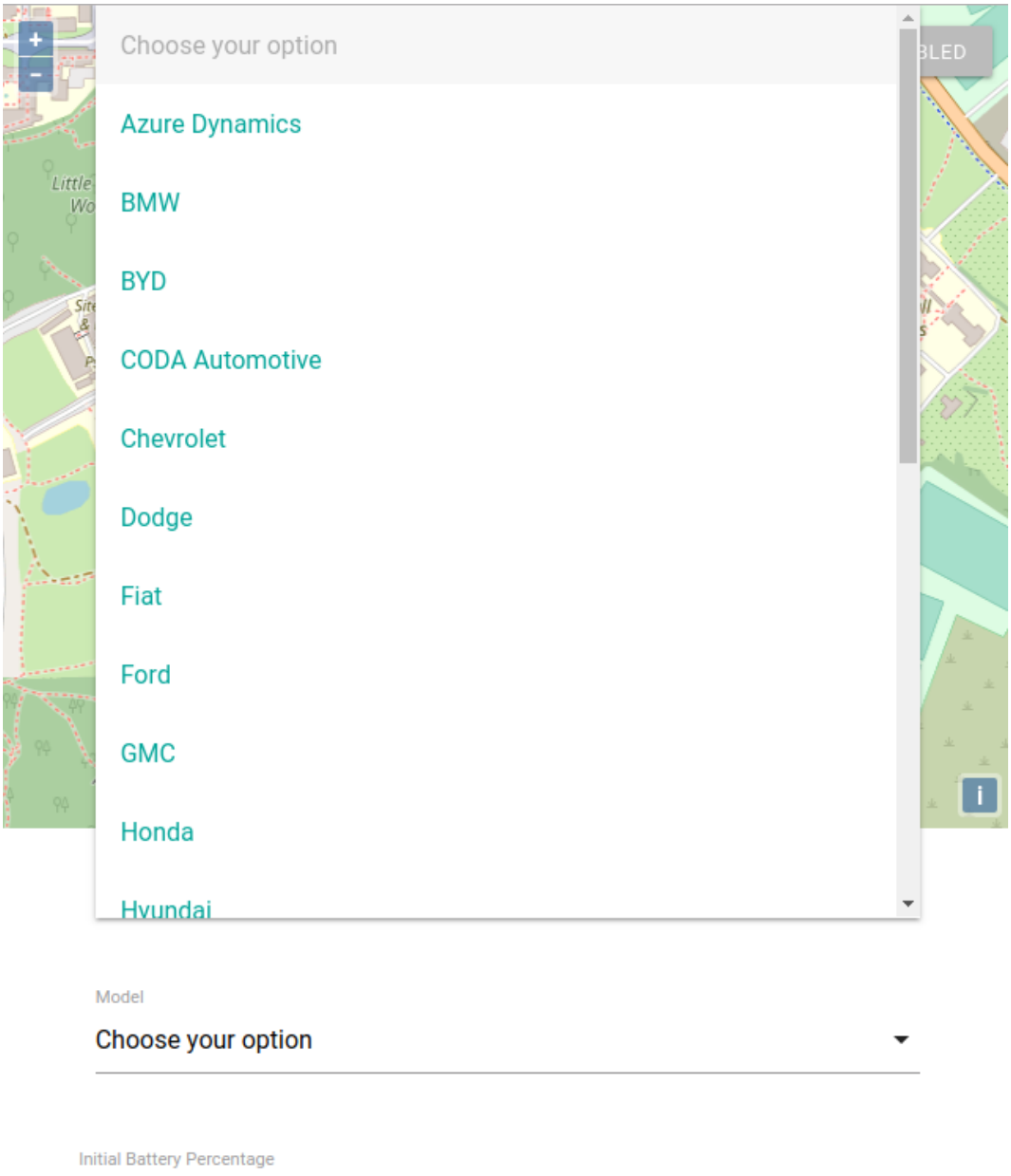


Figure 4.2: Brands selection in the application

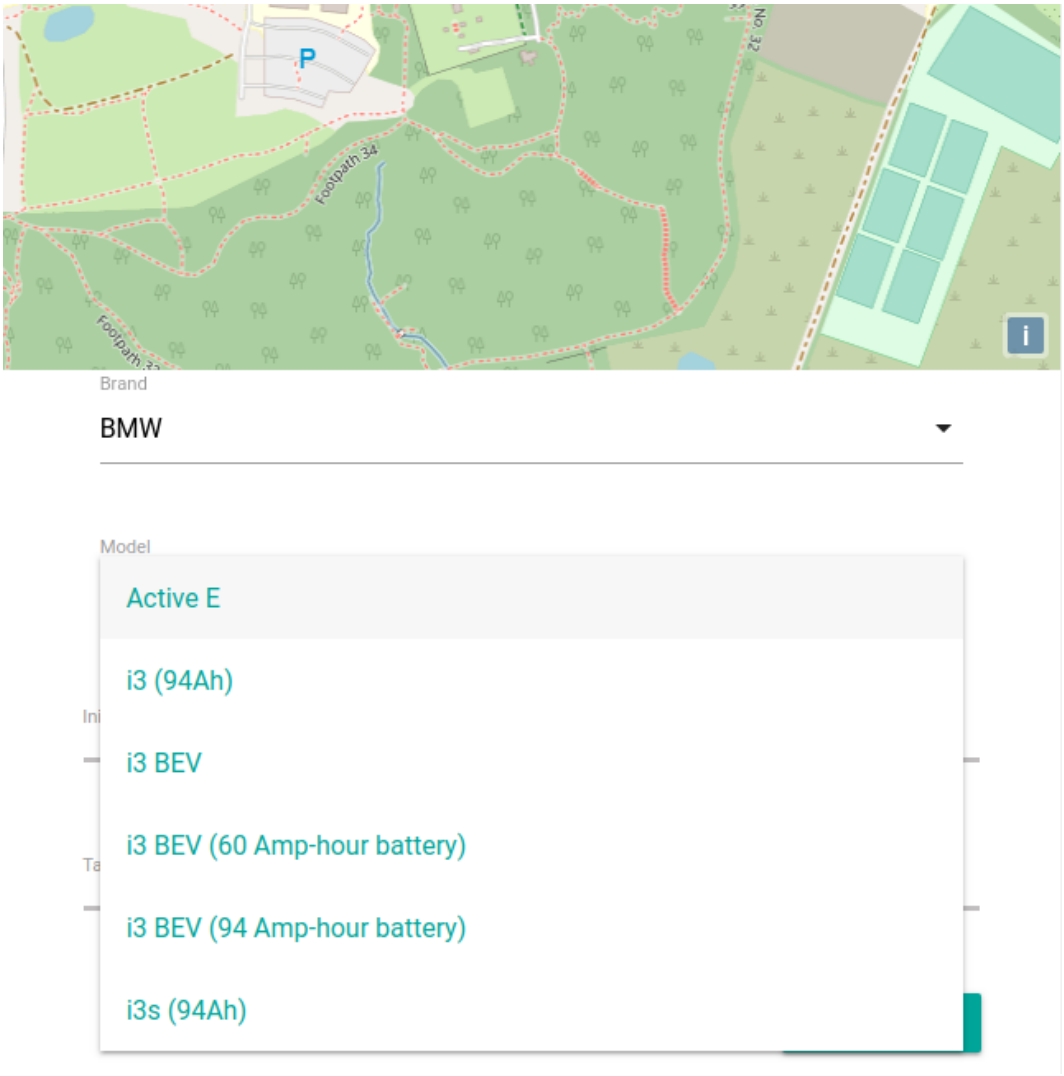


Figure 4.3: Models selection in the application

In Fig. 4.4 appears a portion of the list of the manufacturers contained in one of the databases of the application. This list is obtained after making a query to get all the manufacturers in the database using the application Robomongo which is connected to the local MongoDB database. This data was originally obtained from the DOE website [29] with only a collection of the electric vehicles and their attributes. However, a separated collection was created only with the manufacturers for purposes of displaying a list of them in the application as it would be more efficient. Thus, as it can be seen in figure 3, the application has a source from where to obtain a list of manufacturers of electric vehicles to display in the user side of the application.

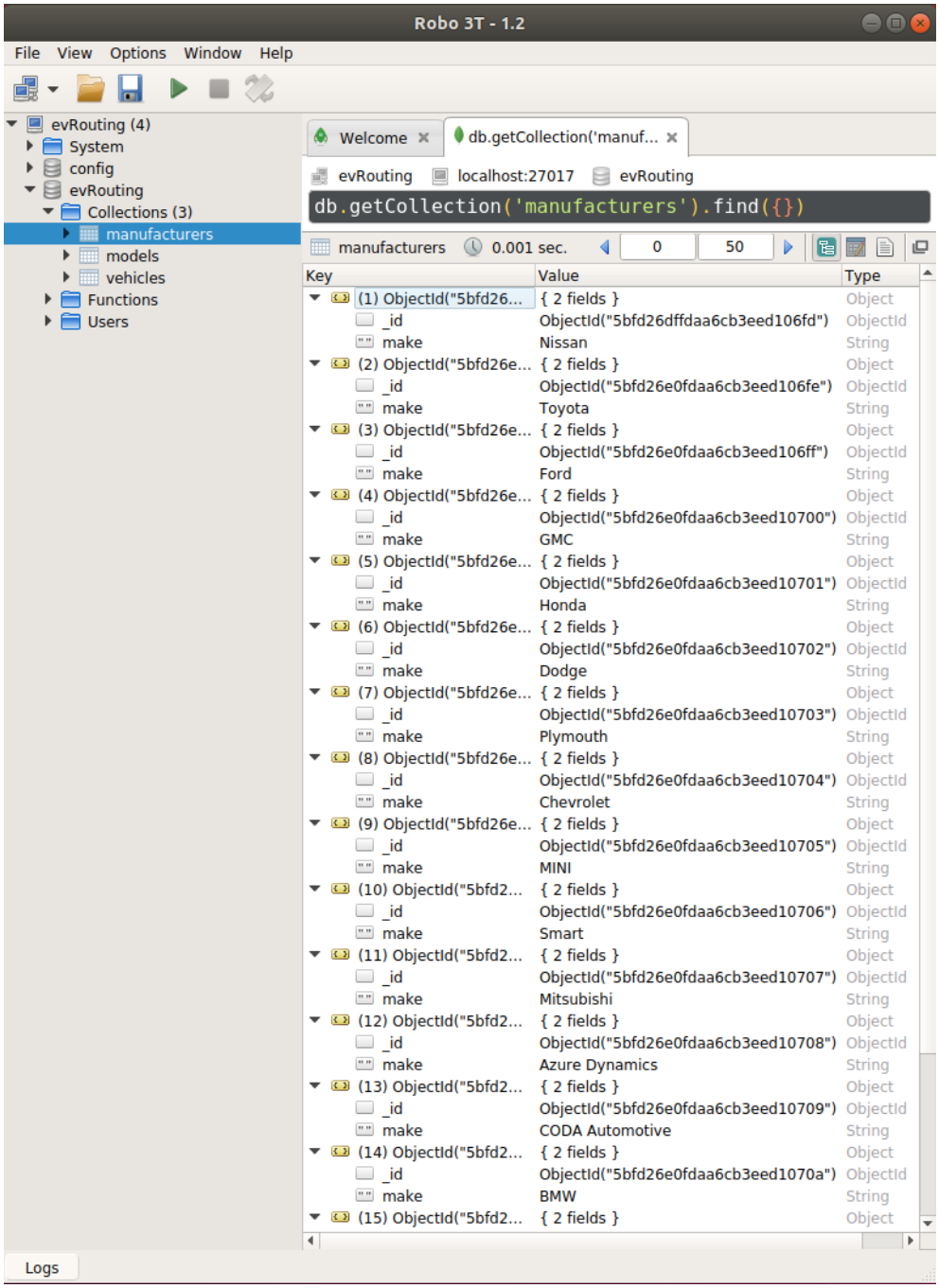


Figure 4.4: List of EV manufacturers in database

In Fig 4.5 the list of the Electric Vehicle's models contained in the database is displayed. This collection of models was created in the same way as the manufacturers collection, however, a relationship with the Manufacturer's collection was registered in models collection to make it possible relation between manufacturers and vehicles. Therefore, this collection is useful for the purpose of obtaining the

vehicles models and making a relation between manufacturers and vehicles.

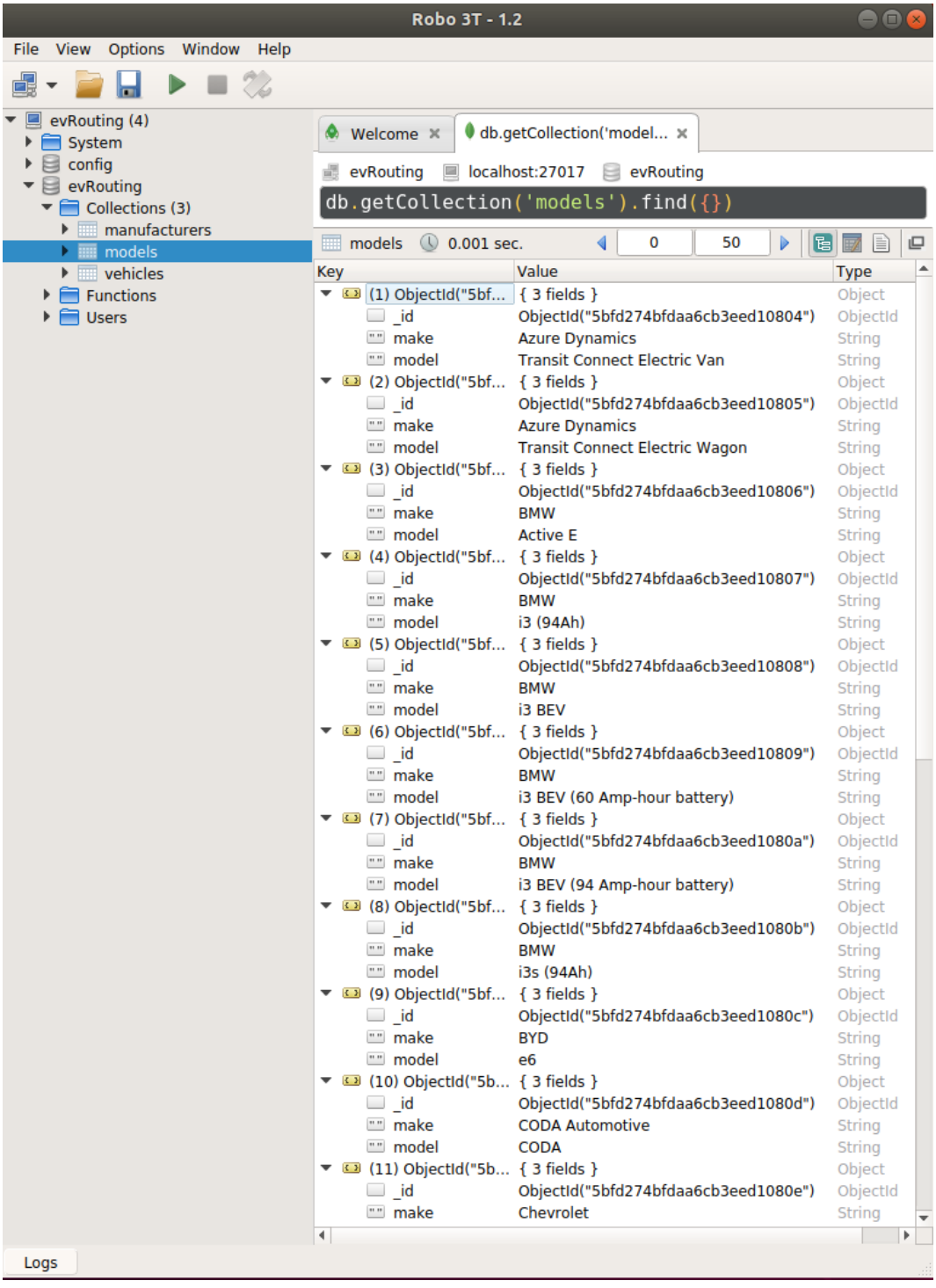


Figure 4.5: List of EV models in database

In Fig 4.6 the list of the Electric Vehicles records contained in the database is displayed including attributes of each vehicle. These attributes are displayed as

they were obtained from the DOE website [29]. Moreover, these attributes contain useful information of the vehicles including energy efficiency which can be used for the purpose of calculating the energy spent on the route. Thus, the vehicle's data in the database is useful for the application as it includes the required data for the user side and also data to calculate the energy consumption of the vehicle.

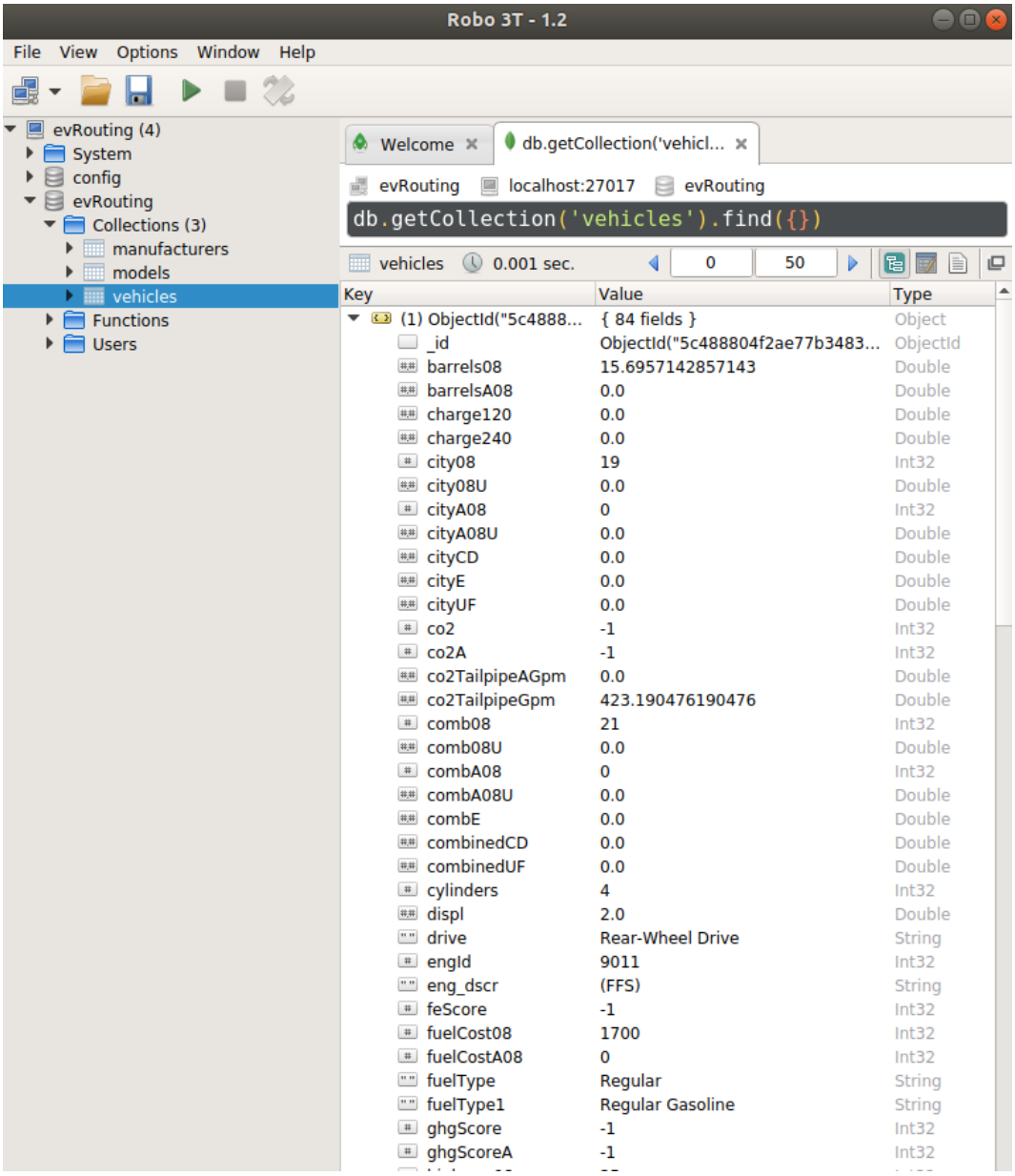


Figure 4.6: List of Electric Vehicles in database

In fig. 4.7, the charging stations are displayed on the map as green filled circles after making a request to the server and obtaining the data of location for the

charging stations that are close to the users' location. In this way, the users can visualize the location of the charging stations to consider their options as a station can be closer to other destination to consider after charging the vehicle. This data was obtained from the Open Charge Map website [28]. Thus, the inclusion of the charging station on the map is useful for the application to help the users make a decision about their destination.

Moreover in fig. 4.7, a list of charging stations appears showing useful data to the users. The data provided to the users contains the name of the charging station, distance in metric units to the station, energy necessary to get to the station and time to arrive at the station from the users' location. This list allows the users to select the station desired by clicking on it, the route to the station appears and the route is zoomed in to make it easier for them to focus on the route. Hence, this confirms that the users have the option of selecting other stations to use for the purpose of going to a station that suits their necessities.

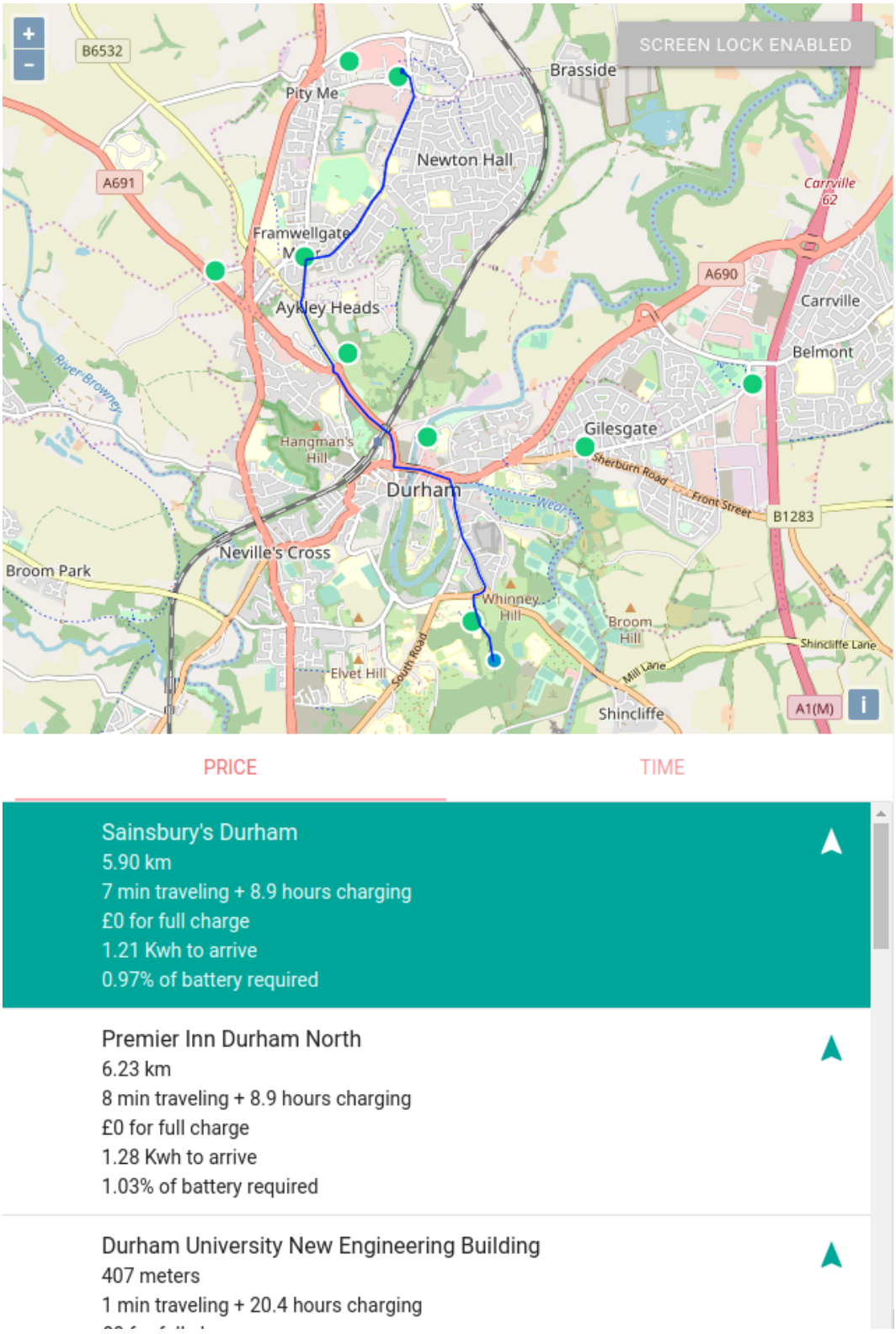


Figure 4.7: List of Charging Stations ordered by Price

In fig. 4.8, it can be observed that the list of charging stations can be ordered

by time too. If the price is selected, the first charging station that would appear are the ones that would present a cheaper monetary cost for charging the electric vehicle. However, if time is selected, the first charging station that will appear are the ones with the lowest time required to charge considering the time of travel and the time required for charging to the percentage selected.

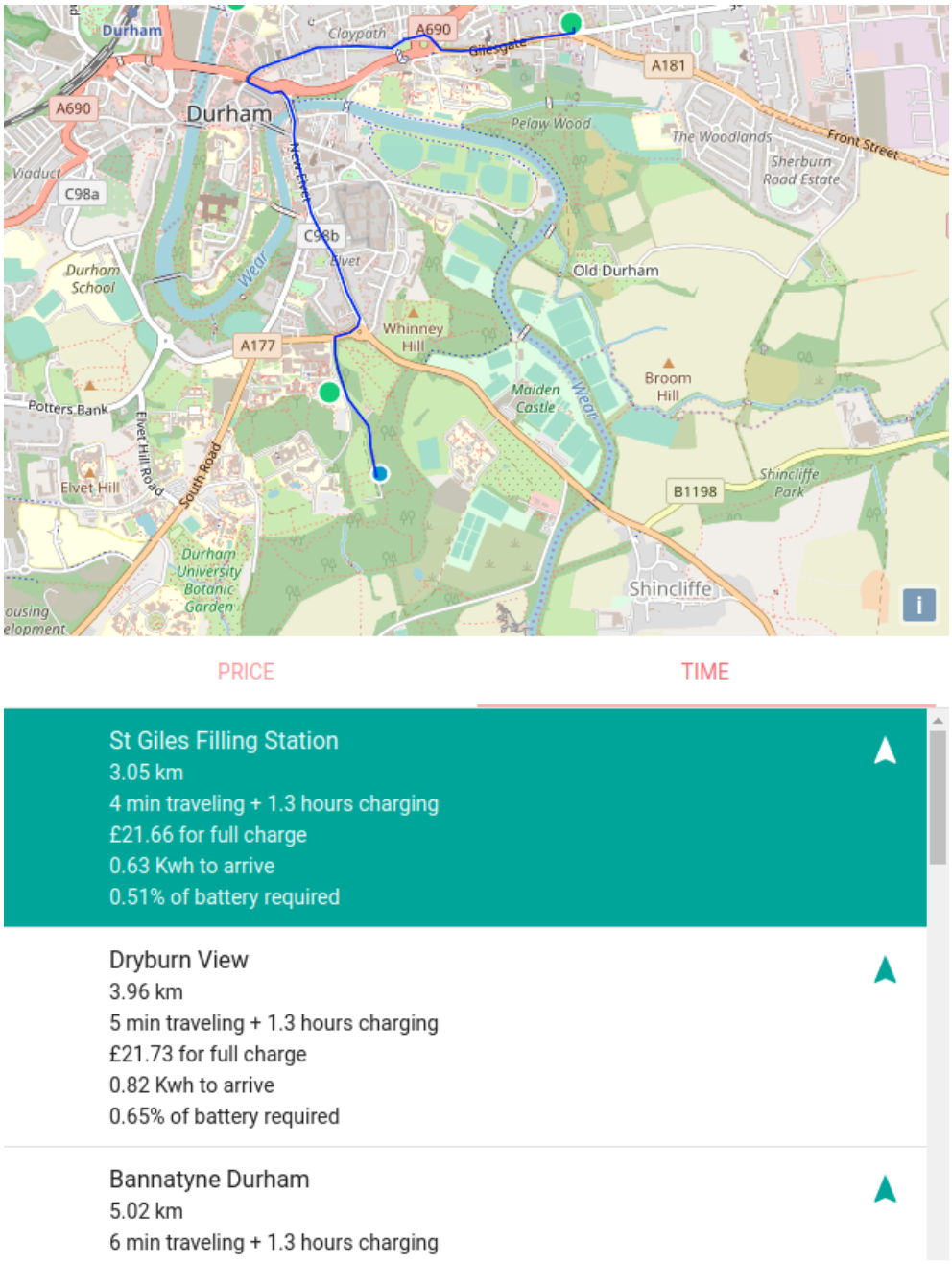


Figure 4.8: List of Charging Stations ordered ordered by time

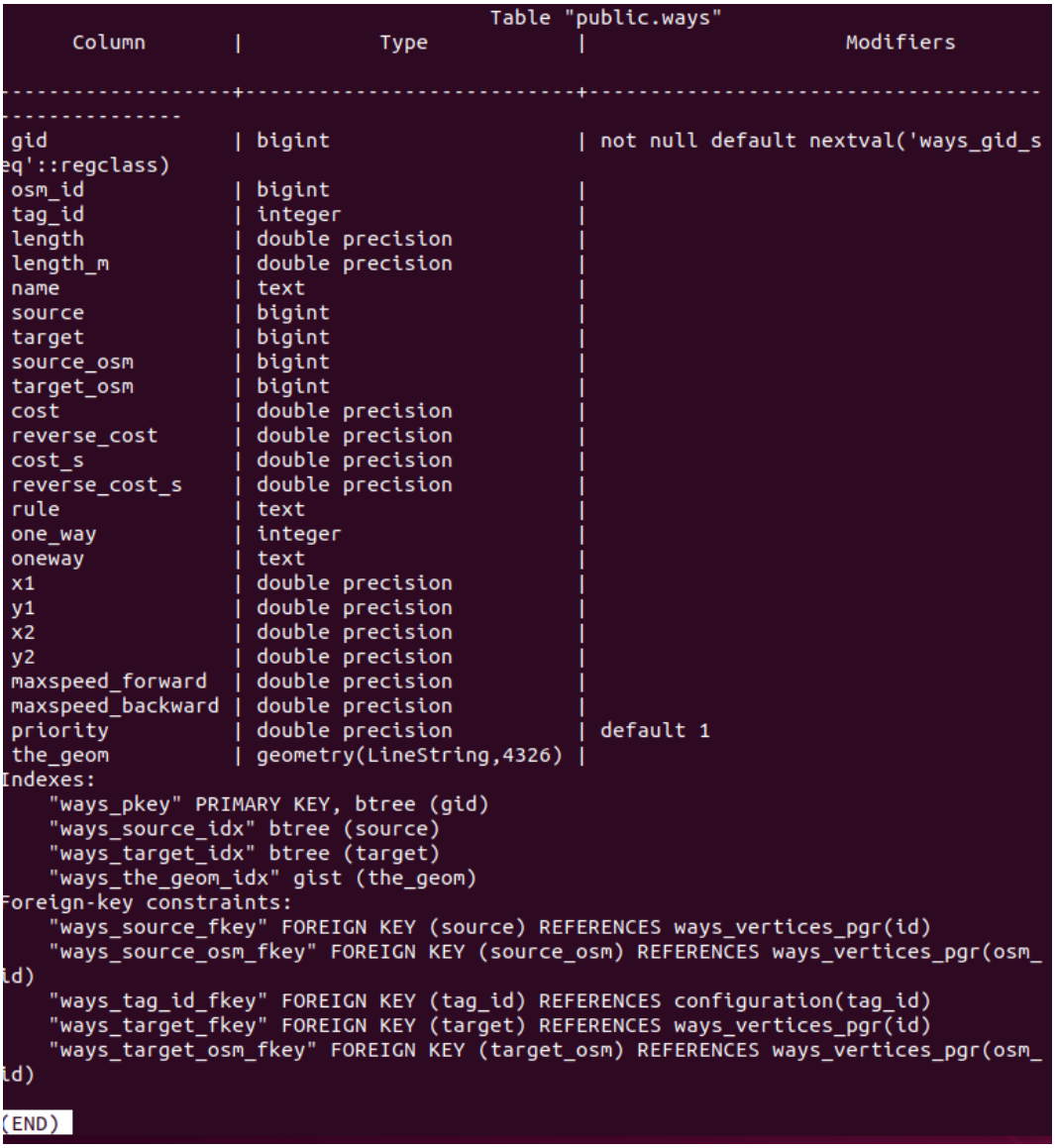
To explain the difference between the two order options, in table 4.1, the data

is presented. Here, it can be noticed that when the option to order by time was selected, the options provided are significantly less time consuming that when ordered by price e.g. First option when ordered by time required approximately 1.3 hours to arrive to the station and charge the vehicle and first option when ordered by price required approximately 8.9 hours. However, when the option to order by price was selected, the options provided were the ones that required the least amount of money to charge the vehicle e.g. First and second options when ordered by price provided charging station when charging was free and first option when ordered by time required £21.66 to charge the vehicle to the battery percentage desired. This data corroborates that the application will provide the users the options that are most suited for them.

	Ordered by Time		**Ordered by Price**	
Option	**Time**	**Price**	**Time**	**Price**
First	4min+1.3hours	£21.66	7min+8.9hours	Free
Second	5min+1.3hours	£21.73	8min+8.9hours	Free

Table 4.1: Comparison of fig 4.7 and 4.8

In fig. 4.9, a query was sent to the roads database to obtain the structure of the table where the roads are stored. It can be seen that each record includes useful data like the name of the road, initial and ending coordinates, maximum speed permitted, its relation with its point on the map and others.



```
                                    Table "public.ways"
     Column       |           Type            |                      Modifiers
------------------+---------------------------+-----------------------------------
--------------
 gid              | bigint                    | not null default nextval('ways_gid_s
eq'::regclass)
 osm_id           | bigint                    |
 tag_id           | integer                   |
 length           | double precision          |
 length_m         | double precision          |
 name             | text                      |
 source           | bigint                    |
 target           | bigint                    |
 source_osm       | bigint                    |
 target_osm       | bigint                    |
 cost             | double precision          |
 reverse_cost     | double precision          |
 cost_s           | double precision          |
 reverse_cost_s   | double precision          |
 rule             | text                      |
 one_way          | integer                   |
 oneway           | text                      |
 x1               | double precision          |
 y1               | double precision          |
 x2               | double precision          |
 y2               | double precision          |
 maxspeed_forward | double precision          |
 maxspeed_backward | double precision          |
 priority         | double precision          | default 1
 the_geom         | geometry(LineString,4326) |
Indexes:
    "ways_pkey" PRIMARY KEY, btree (gid)
    "ways_source_idx" btree (source)
    "ways_target_idx" btree (target)
    "ways_the_geom_idx" gist (the_geom)
Foreign-key constraints:
    "ways_source_fkey" FOREIGN KEY (source) REFERENCES ways_vertices_pgr(id)
    "ways_source_osm_fkey" FOREIGN KEY (source_osm) REFERENCES ways_vertices_pgr(osm_
id)
    "ways_tag_id_fkey" FOREIGN KEY (tag_id) REFERENCES configuration(tag_id)
    "ways_target_fkey" FOREIGN KEY (target) REFERENCES ways_vertices_pgr(id)
    "ways_target_osm_fkey" FOREIGN KEY (target_osm) REFERENCES ways_vertices_pgr(osm_
id)

(END)
```

Figure 4.9: Query showing roads table in PostgreSQL

In fig. 4.10, a query was sent to the geospatial database using pgRouting, PostgreSQL and Dijkstra's algorithm is displayed. The server answered with records that describe the route to go from Heaviside Place to Glue Garth at Durham, United Kingdom step by step in order from the start point to the endpoint. The route was corroborated successfully to determine if it was an accurate path to follow. Therefore, this evidence confirms that pgRouting was successfully included in the application to formulate useful routes for the user.

```
 seq |  gid  |       name       |       length        |      the_time
-----+-------+------------------+---------------------+---------------------
   1 | 27222 | Heaviside Place  |  0.0478097731371507 |  0.0594151576319209
   2 | 15910 | Heaviside Place  |  0.0155954577318041 |  0.0193811122428665
   3 | 19609 | Heaviside Place  | 0.00903878250708809 |  0.0112328641660502
   4 | 25404 | Long Acres       |   0.044348466857425 |  0.0551136509581221
   5 | 27785 | Long Acres       |  0.0193148992091827 |  0.0240034140690476
   6 |  9090 | Long Acres       |  0.0139981729385721 |  0.0173961011970975
   7 | 23365 | Long Acres       |  0.0313370602028589 |  0.0389438445217609
   8 | 29576 | Wakenshaw Road   |  0.0593389996229559 |  0.0737429982402242
   9 | 12601 | Wakenshaw Road   |   0.097428117220868 |    0.1210779002413
  10 | 21501 | Wakenshaw Road   |  0.0172729852722238 |  0.0214658442276848
  11 | 13426 | Wakenshaw Road   |  0.0183266571850745 |  0.0227752853458169
  12 | 13425 | Churchill Avenue |  0.0432350073680081 |  0.0537299093768625
  13 | 33784 |                  |  0.0463717067747055 |  0.0576280138325823
  14 |  9507 |                  |  0.0365152536634407 |  0.0453789970129987
  15 |  9790 | Edward Street    | 0.00995958091253323 |  0.0123771779499663
  16 | 24524 | Edward Street    |  0.0477926466573677 |  0.0593938738560479
  17 |  5249 | Sunderland Road  | 0.00361901002171626 | 0.00449749155460201
  18 | 30086 | Sunderland Road  |  0.0275496738642561 |  0.0342371048416637
  19 | 35613 | Glue Garth       |  0.0189244337354799 |  0.0235181666782406
(19 rows)

evrouting=#
```

Figure 4.10: Result of query showing list of steps for a route

In figure 4.11, it can be seen the route being drawn in the application. Using the development tools of the browser, it can be seen that the route is sent as an image. This image of the route is sent by Geoserver, which uses pgRouting to formulate the route and it is created using the data from OpenStreetMap, considering the coordinates of the user and the charging station. Thus, the previous statements indicate that routes are correctly sent to the application to indicate the user how to arrive at a charging station.

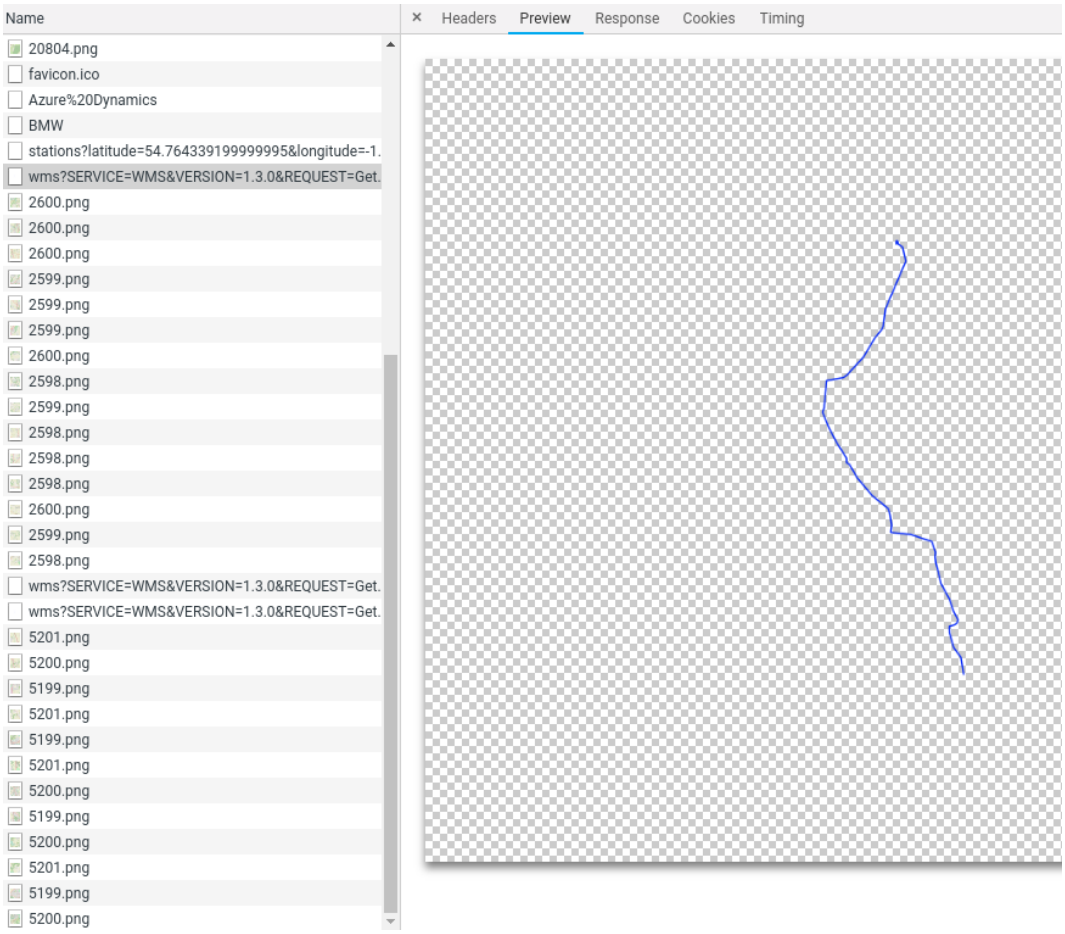


Figure 4.11: Route image format

Table 4.2 compares options given in the application when searching for a charging station. The first column shows the options when the priority is the time and the second column shows the options ordered when the priority is the price. In the first option, when ordered by time, the amount of time required to travel and charge the vehicle is approximately ten hours less than when ordered by price, which is a decrease in 90% from the cheapest option, but when the priority is price, the first options are free or £1 per session, which much cheaper than the options when ordered by time with prices of £28 approximately, which is 28 times more expensive than the fifth option when ordered by price. Moreover, it can be observed that when ordered by the time, first, is ordered by the time and then the stations with the same time are ordered by price. Inversely, when ordered by price first, the options with the same price are afterwards ordered by time. This order is done in order to assure that the options that the users would prefer are shown first. Therefore, these

results indicate the application successfully selects and show first the options more appropriated for the user interests.

	Ordered by Time		**Ordered by Price**	
Option	**Time**	**Price**	**Time**	**Price**
First	1.6hours+3min	£28.26	11.6hours+6min	Free
Second	1.6hours+4min	£28.32	11.6hours+7min	Free
Third	1.6hours+5min	£28.36	26.8hours+1min	Free
Fourth	1.6hours+6min	£28.41	11.5hours+2min	£1
Fifth	11.5hours+2min	£1	11.5hours+2min	£1

Table 4.2: Comparison between order by time and price

Chapter 5

Conclusions

This thesis has presented the development of a web-based application for electric vehicles routing capable of sharing with the users the information necessary for them to make decisions and select the lowest cost route considering the users' priority, whether it can be monetary cost or time required for the trip. Evidence has been presented indicating that the application is capable of displaying a map, formulate routes for the users' trip to the charging stations and select the one with the lowest cost for the users. Moreover, the application can switch between users priorities (reducing time or money).

Finally, results indicated that the application can successfully recommend routes and stations to the users, that can reduce greatly the costs in time or money for them, considering the price, time of travel, time of charging, energy spent, the vehicles' battery capacity, remaining energy and desired charge percentage.

5.1 Future Work

As this work was made for a master degree, time and resources were limited to be able to deliver in time. Therefore, in future versions of this system more features can be included to make it more accurate.

One feature that can be included is the consideration of traffic to make the trip time more precise. The consideration of road inclination can be helpful to better calculate the energy consumption and recuperation. The consideration of the

weather can help to calculate the energy consumption. The battery recuperation factor when the vehicle brakes can be considered to calculate the energy necessary for the trip.

Moreover, another kind of features can be included to expand the purpose of this system as for example, the destination can be not only a charging station but it can be a specific target on the map too. Furthermore, it can be included the feature of selecting a different starting point than the users' locations.

Bibliography

[1] Zhou, Z., et al. Charging station selection optimization based on electric and traffic information. in 2017 IEEE Power & Energy Society General Meeting. 2017.

[2] Marano, V., et al., Intelligent Energy Management for Plug-in Hybrid Electric Vehicles: The Role of ITS Infrastructure in Vehicle Electrification. Dossier, 2012. 67: p. 575-587.

[3] Tianheng, F., et al., A Supervisory Control Strategy for Plug-In Hybrid Electric Vehicles Based on Energy Demand Prediction and Route Preview. IEEE Transactions on Vehicular Technology, 2015. 64(5): p. 1691-1700.

[4] Newbery, D.M., Towards a green energy economy? The EU Energy Union 's transition to a low-carbon zero subsidy electricity system – Lessons from the UK's Electricity Market Reform. Applied Energy, 2016. 179: p. 1321-1330.

[5] Yang, H., et al., Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm. 2015. 6(2): p. 657-666.

[6] Zhihong, W., et al. A New Route Searching Method for EVs Considering Electric Motor Efficiency and Charging Stations. in 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics. 2013.

[7] Sweda, T.M. and D. Klabjan. Finding minimum-cost paths for electric vehicles. in 2012 IEEE International Electric Vehicle Conference. 2012.

[8] Baum, M., et al. Energy-optimal routes for electric vehicles. in Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems. 2013. ACM.

[9] Kobayashi, Y., et al. A route search method for electric vehicles in consideration of range and locations of charging stations. in 2011 IEEE Intelligent Vehicles Symposium (IV). 2011.

[10] Sachenbacher, M. The Shortest Path Problem Revisited: Optimal routing for Electric Vehicles. in 2nd International Conference on Computational Sustainability (CompSust). 2010.

[11] Sachenbacher, M., et al. Efficient Energy-Optimal Routing for Electric Vehicles. in AAAI. 2011.

[12] Ortega-Arranz, H., D.R. Llanos, and A. Gonzalez-Escribano, The shortest-path problem: Analysis and comparison of methods. Synthesis Lectures on Theoretical Computer Science. Vol. 1. 2014. 1-87.

[13] Cherkassky, B.V., A.V. Goldberg, and T. Radzik, Shortest paths algorithms: Theory and experimental evaluation. Mathematical programming, 1996. 73(2): p. 129-174.

[14] Dijkstra, E.W., A note on two problems in connexion with graphs. Numerische mathematik, 1959. 1(1): p. 269-271.

[15] Hart, P.E., N.J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 1968. 4(2): p. 100-107.

[16] Nicholson, T.A.J., Finding the shortest route between two points in a network. The computer journal, 1966. 9(3): p. 275-280.

[17] Goldberg, A.V. and C. Harrelson. Computing the shortest path: A search meets graph theory. in Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. 2005. Society for Industrial and Applied Mathematics.

[18] Jasika, N., et al. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. in 2012 proceedings of the 35th international convention MIPRO. 2012. IEEE.

[19] Singla, G., A. Tiwari, and D.P. Singh, New approach for graph algorithms on GPU using CUDA. International Journal of Computer Applications, 2013. 72(18).

[20] Cela, A., et al., Energy optimal real-time navigation system. IEEE Intelligent Transportation Systems Magazine, 2014. 6(3): p. 66-79.

[21] Hrazdira, A., et al. Optimal real-time navigation system: Application to a hybrid electrical vehicle. in 2012 15th International IEEE Conference on Intelligent Transportation Systems. 2012. IEEE.

[22] Vatanparvar, K. and M.A. Al Faruque. Eco-friendly automotive climate control and navigation system for electric vehicles. in 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS). 2016. IEEE.

[23] Guo, Q., et al., Rapid-charging navigation of electric vehicles based on real-time power systems and traffic data. IEEE Transactions on smart grid, 2014. 5(4): p. 1969-1979.

[24] Yang, H., et al., Electric vehicle route selection and charging navigation strategy based on crowd sensing. IEEE Transactions on Industrial Informatics, 2017. 13(5): p. 2214-2226.

[25] Jurik, T., et al. Energy optimal real-time navigation system: application to a hybrid electrical vehicle. in 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). 2013. IEEE.

[26] Liaw, B.Y. and M. Dubarry, From driving cycle analysis to understanding battery performance in real-life electric hybrid vehicle operation. Journal of power sources, 2007. 174(1): p. 76-88.

[27] Cabani, A., et al. Intelligent navigation system-based optimization of the energy consumption. in 2015 IEEE Intelligent Vehicles Symposium (IV). 2015. IEEE.

[28] Open Charge Map. Open Charge Map. 2018 [cited 2018 26 Sept 2018]; Available from: https://openchargemap.org/site/.

[29] FuelEconomy. Fuel Economy. 2018 [cited 2018 26/09/2018]; Available from: https://www.fueleconomy.gov/.

[30] pgRouting. pgRouting. 2018 [cited 2018 06/11/2018]; Available from: https://pgrouting.org/.

[31] Pilgrim, M., HTML5: up and running: dive into the future of web development. 2010: " O'Reilly Media, Inc.".

[32] Doernhoefer, M., JavaScript. SIGSOFT Softw. Eng. Notes, 2006. 31(4): p. 16-24.

[33] Bos, B., et al., Cascading style sheets, level 2 CSS2 specification. 1998: p. 1472-1473.

[34] Amazon Web Services. Amazon Web Services. 2018 [cited 2018 21/11/2018]; Available from: https://aws.amazon.com/.

[35] Materialize. Materialize. 2019 [cited 2019 16/01/2019]; Available from: https://materializecss.com/.

[36] Google. Introduction - Material Design. 2019 [cited 2019 16/01/2019]; Available from: https://material.io/design/introduction/#principles.

[37] Git. Git. 2018 [cited 2018 21/11/2018]; Available from: https://git-scm.com/.

[38] Github. Github. 2019 [cited 2019 16/01/2019]; Available from: https://github.com/.

[39] OpenStreetMap. OpenStreetMap. 2018 [cited 2018 25 Sept 2018]; Available from: https://www.openstreetmap.org/about.

[40] OpenLayers. OpenLayers. 2018 [cited 2018 25 Sept 2018]; Available from: https://openlayers.org/.

[41] GeoServer. GeoServer. 2018 [cited 2018 06/11/2018]; Available from: http://geoserver.org/.

[42] Geofabrik GmbH Karlsruhe. GEOFABRIK. 2018 [cited 2018 15/10/2018]; Available from: http://www.geofabrik.de/.