UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

Design and implementation of a mobile application for eco-routing and driving evaluation

Iván Sánchez Cordero

Enero, 2025

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

Design and implementation of a mobile application for eco-routing and driving evaluation

Autor: Iván Sánchez Cordero

Tutora: Cristina Vicente Chicote

Co-Tutor: José Ramón Lozano Pinilla

Resumen

Hoy en día, cada vez más personas se unen a la tendencia de emplear sus *smartphones* para obtener la mejor ruta posible (la más corta o la más rápida) entre dos puntos, uno de origen y otro de destino, a través de aplicaciones basadas en mapas interactivos.

Sin duda, la tecnología gratuita más extendida para llevar a cabo esta tarea es *Google Maps*, ofreciendo una interfaz interactiva muy intuitiva y fácil de usar, con la que se pueden seleccionar elementos del mapa, buscar localizaciones y obtener rutas en cuestión de segundos entre dos o más puntos y para distintos medios de transporte (en coche, transporte público, a pie o en bicicleta). Aún así, cada vez más usuarios se preocupan por su privacidad y la forma en que esta plataforma gestiona sus datos, por lo que buscan alternativas que les permitan obtener rutas de calidad sin tener que ceder su información personal, estando muchas basadas en otras plataformas libres como *OpenStreetMap*: un proyecto fundado en 2004 que ha ido creciendo a nivel mundial y busca ofrecer un servicio de mapas sin restricciones y gratuito.

Gran parte de las aplicaciones disponibles para calcular rutas terrestres se centran en obtener la ruta más corta o la más rápida, pudiendo ofrecer otras alternativas con otras propiedades dependiendo de las preferencias del usuario. Recientemente, Google Maps ha incorporado una nueva funcionalidad que permite a los usuarios seleccionar la ruta más ecológica, denominada *eco-ruta*, que tiene en cuenta la reducción de combustible y de emisiones de gases contaminantes. Sin embargo, tras varias pruebas, se ha comprobado que esta funcionalidad no es tan eficaz como se esperaba, ofreciendo una estimación respecto al consumo poco fiable en varios casos.

El cambio climático es un problema global que afecta a todos los seres vivos del planeta, y una de las causas principales de este fenómeno es la emisión de gases de efecto invernadero, siendo el transporte terrestre de vehículos de combustión uno de los principales responsables de este nivel de emisiones. Por ello, cada vez son más numerosos los trabajos de investigación que remarcan la importancia del *eco-routing* y del *eco-driving*. Las técnicas de *eco-routing* se centran en la obtención de rutas que minimicen la contaminación, teniendo en cuenta factores como la pendiente de la carretera, la congestión del tráfico o la velocidad media de circulación. En general, se priorizan recorridos con un menor número de rampas ascendentes y una mayor fluidez en el tráfico para reducir el consumo de combustible y el nivel de emisiones. Por otro lado, el *eco-driving* aboga por aplicar técnicas de conducción eficiente, promoviendo el uso adecuado de los cambios de marchas y evitando cambios bruscos en la aceleración.

En este contexto, el Trabajo Fin de Máster que aquí se presenta pretende desarrollar una aplicación móvil que ofrezca a los usuarios la posibilidad de calcular rutas desde su ubicación hasta el destino que elijan en un mapa interactivo, pudiendo seleccionar sus propios vehículos y especificar propiedades como el número de pasajeros o la carga del vehículo para que se tomen en cuenta, mostrando tres rutas alternativas: la más corta, la más rápida y la más eco-eficiente (eco-ruta). Además, se ha implementado una funcionalidad para evaluar la conducción de los usuarios en tiempo real, pudiendo obtener estadísticas útiles en función de la ruta seleccionada y de la forma de conducir. Con esto, se pretende lograr tres objetivos principales: (1) proporcionar a los usuarios una aplicación sencilla e intuitiva; (2) calcular, en un tiempo razonable, rutas de calidad, con mayor precisión (en términos de eco-eficiencia) que otras aplicaciones; y (3) que su uso promueva y fomente técnicas de *eco-driving* para lograr una reducción considerable de los niveles de contaminación.

Palabras clave: Cálculo de rutas terrestres para vehículos de combustión, *Eco-routing*, *Eco-driving*, Reducción de Combustible y de Emisiones de Gases de Efecto Invernadero, Mapas Móviles Interactivos y Geolocalización.

Abstract

Nowadays, an increasing number of people are joining the trend of using their *smartphones* to obtain the best possible route (the shortest or the fastest) between two points, origin and destination, through applications based on interactive maps.

Undoubtedly, the most widely used free technology for carrying out this task is *Google Maps*, offering an interactive interface that is very intuitive and easy to use, with which elements of the map can be selected, locations can be searched, and routes can be obtained in a matter of seconds between two or more points and for different modes of transportation (by car, public transportation, on foot, or by bicycle). However, an increasing number of users are concerned about their privacy and the way this platform manages their data, so they are looking for alternatives that allow them to obtain high-quality routes without having to provide their personal information, many of which are based on other free platforms like *OpenStreetMap*: a project founded in 2004 that has been growing globally and aims to offer a map service without restrictions and free of charge.

Most of the applications available for calculating terrestrial routes focus on obtaining the shortest or fastest route, and can offer other alternatives based on the user's preferences. Recently, Google Maps has incorporated a new functionality that allows users to select the most eco-friendly route, called *eco-route*, which takes into account the reduction of fuel consumption and greenhouse gas emissions. However, after several tests, it has been found that this functionality is not as effective as expected, offering an unreliable estimate of consumption in several cases.

Climate change is a global problem that affects all living beings on the planet, and one of the main causes of this phenomenon is the emission of greenhouse gases, with terrestrial transportation of combustion vehicles being one of the main causes for this level of emissions. Therefore, an increasing number of research works are highlighting the importance of *eco-routing* and *eco-driving*. *Eco-routing* techniques focus on obtaining routes that minimize pollution, taking into account factors such as road slope, traffic congestion, or average speed. In general, routes with a lower number of uphill ramps and a higher traffic flow are prioritized to reduce fuel consumption and emissions. On the other hand, *eco-driving* advocates for applying efficient driving techniques, promoting the proper use of gear changes and avoiding sudden changes in acceleration.

In this context, this Master's Thesis aims to develop a mobile application that offers users the possibility of calculating routes from their location to the destination they choose on an interactive map, allowing them to select their own vehicles and specify properties such as the number of passengers or the vehicle load, showing three alternative routes: the shortest, the fastest, and the most eco-efficient (eco-route). Additionally, a functionality has been implemented to evaluate the driving habits of users in real-time, allowing them to obtain useful statistics based on the selected route and driving style. This intends to achieve three main goals: (1) provide users with a simple and intuitive application; (2) compute, in a reasonable time, high-quality routes, more precise (in terms of eco-efficiency) than alternative applications; and (3) promote and encourage eco-driving techniques among users to achieve a significant reduction in pollution levels.

Keywords: Land Routes for Motor Vehicles, *Eco-routing*, *Eco-driving*, Fuel and Greenhouse Gas Emission Reduction, Interactive Mobile Maps and Geolocation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

The current lifestyle, characterized by constant mobility, has led to the expansion of cities and the increased use of terrestrial vehicles (public or private, individual and collective), surpassing walking as the primary mode of transportation. While residents of smaller cities often possess knowledge about the majority of the streets and routes, in larger cities, interurban mobility or unfamiliar locations typically require the use of guides for orientation and arrival at destination without major inconveniences.

Until recently, the only available guides for navigating unfamiliar locations were paper maps, which graphically represented various types of roads and notable buildings. However, with the rise of smartphones, geolocation, and digital cartography, route planning applications have proliferated, enabling users to easily and intuitively visualize routes on interactive maps and obtain directions to any

destination using various modes of transportation (on foot, by car, or using public transportation). Additionally, these applications provide access to relevant information for planning trips, such as nearby points of interest (restaurants, gas stations, etc.), traffic and road conditions, etc. The information managed by these applications is continuously updated, offering a significant advantage over traditional physical maps. This information is sourced from various channels, including application users, who can report incidents in real-time using their mobile devices, likely being one of the most relevant sources.

The development of cartographic navigation and visualization tools has undergone significant transformations in recent years. Although the exact origin of the first online map is unknown, it can be asserted that one of the earliest examples was the **PARC Map Viewer** developed by Xerox in 1993 [1]. This pioneering tool offered functionalities such as zooming and layering, which have been inherited by modern applications. To display data on the map, hyperlinks were employed, incorporating user request properties such as GPS coordinates of selected points, utilizing a geographic database to load an image of the result and returning it to the browser.

In 1996, **Mapquest** was launched, revolutionizing the way cartographic services were conceived and becoming one of the first services designed for consumer users. It introduced innovative features such as searching for points of interest, which were directly marked on the map, and obtaining driving directions between two locations. Google launched *Google Local* in 2003 as an attempt to stay competitive in this field, but it failed to gain significant popularity. However, Google acquired companies such as **Keyhole**, **Where2 Technologies**, and **ZipDash**, which specialized in cartography and navigation, to launch **Google Maps** in 2005 (along with a mobile version).

Besides, modern navigation applications are increasingly offering features that promote intelligent mobility and energy efficiency, such as route calculation that avoids congested or closed roads due to construction or accidents. Recently, some of these

applications have also begun to offer eco-friendly routes, allowing users to select the option with the lowest fuel consumption and greenhouse gas emissions.

The most dominant application in this field is *Google Maps*, which offers a wide range of features for route planning ahead of its competition, being one of the platforms with the largest amount of data thanks, precisely, to its large number of users. In fact, this application has been one of the first to offer eco-route calculation.

The calculation of eco-routes requires considering not only the distance between the trip origin and destination, but also factors such as the topography of the terrain (e.g., road slope) or the type of vehicle being used (e.g., weight, engine type, etc.), as these factors are crucial for determining fuel consumption and, therefore, greenhouse gas emissions. While *Google Maps* offers the option to select eco-routes, it does not provide detailed information on the accuracy of this result, nor does it allow users to modify factors that influence the calculation of the eco-route beyond the type of fuel used by the vehicle. Furthermore, although many of the data used by Google Maps to compute the requested routes is gathered from users, this information is not made publicly available (e.g., for other applications to use it), but rather provided through restricted "pay-per-use" services.

Another well-known platform is *OpenStreetMap* [2], which offers an open and free database of maps that can be used by anyone to create navigation applications. To achieve this, they employ a data model that groups road points into nodes, which are connected to each other through links, forming streets and roads and providing information about these elements, such as name, road or pavement type, speed limit, and whether it is located in a roundabout or not, among other features. However, since the data is maintained voluntarily by users, it is not always precise or complete.

Given that much of the information handled by commercial tools like *Google Maps* is not accessible, and the information offered by other open-source platforms is not always complete, it is essential to resort to third-party services that complement their functionality and offer this information for free and openly, such as *OpenTopoData*

or *Nominatim*, to obtain, respectively, topographic information (e.g., the elevation of a pair of GPS coordinates) and additional data on the type of road and speed limit in a specific area. The combined use of these technologies and their adaptation in an architecture designed to be used on mobile devices can contribute to the creation of an open and free application that allows for precise and efficient route estimation and, in particular, eco-route calculation.

In addition to the above, the use of mobile devices, typically equipped with a range of sensors useful to measure, e.g., distance travelled, speed, acceleration, etc., allows the application to monitor users' driving style and to provide them with useful recommendations to reduce environmental impact, if needed. This information can be stored in a database and used to calculate statistics on vehicle consumption and emissions, as well as to offer users a score that reflects their efficiency behind the wheel and encourages them to improve their habits.

1.1 Problem description

Despite the technological advancements in the automotive industry, energy efficiency of vehicles remains a significant concern. While there are tools that employ techniques to obtain routes between two points to save time in our travels, most of them do not take energy consumption into account or, if they do, they build on simplified models of the roads, vehicles and drivers.

Platforms such as *Google Maps* seem to gather and count on all the necessary information to plan a trip with the goal of prioritizing energy efficiency, but unfortunately, they do not provide these data for free nor seem to adequately use it to offer accurate fuel consumption estimations.

Although there are various open-source alternatives to *Google Maps* that offer the necessary data for calculating the most ecological routes (information on the road network, terrain topography, or real-time traffic status), to our knowledge, there is no free and publicly available application that uses these data in conjunction to calculate

efficient and accurate eco-routes.

The lack of awareness about the importance of efficient driving is another factor that contributes to increased fuel waste and, consequently, to an increase in greenhouse gas emissions.

To address some of these problems, initiatives such as the National **Eco-Traffic APP** Project have emerged, which proposes creating a mobile application for *Green Navigation* based on the calculation of eco-routes (*eco-routing*) and driver assistance through recommendations to achieve efficient driving (*eco-driving*). This Master's Thesis has been carried out within the framework of this Project, and it focuses on the development of the GRETA mobile application, described in detail in the following chapters.

1.2 Intended goals

The primary objective of this Master's Thesis is to design and develop an interactive, user-friendly, and intuitive mobile application capable of calculating and displaying alternative routes for a given journey, including the shortest, fastest, and most eco-friendly (eco-route) options. To do this, as will be described throughout this Report, it will be necessary to design and implement a mobile application architecture that allows: (1) the integration of different external services and components to manage the information necessary for each user; (2) the storage and updating of said information so that it can be consulted efficiently and accurately; and (3) to display the results through an application with which users can easily interact.

The primary objective is further broken down into the following sub-objectives:

- Investigate the functionality of existing interactive map applications and route-finding tools, with a focus on those that enable emission reduction and fuel savings.
- Based on the aforementioned study, define the application's requirements and

use cases, as well as design its architecture.

- Develop a data model that enables the representation of relevant user and route information, facilitating persistent storage and efficient retrieval for display on the interface.
- Implement the application's user interface, including interactive map visualization and route information management.
- Ensure that the application's route calculation and user information management mechanisms function correctly and within a reasonable time frame.
- Develop a component responsible for logging user routes, storing necessary information for fuel consumption calculations.
- Enhance the accuracy of fuel consumption and emission calculations for routes compared to similar services and applications.
- Enable the application to be used by any user in Spain, regardless of their technical expertise, and without platform limitations.
- Integrate all aforementioned components into a functional, interactive, and intuitive application, allowing users to request optimal routes to their chosen destinations, displayed on a map for enhanced visualization, and receive scores encouraging more efficient driving practices.

1.3 Methodology

The Project was carried out following an agile development methodology, with weekly iterations. The main reasons for choosing this approach were: (1) the flexibility to add or modify project goals and requirements; (2) the constant feedback from tutors provided by each iteration; (3) the rapid identification of functionality failures to correct; and (4) the increase in project documentation as it progresses, both internally in the code and in this Report.

Each week, sometimes in person and sometimes remotely, follow-up meetings were held with the project tutors to review the work done and plan the objectives of the next iteration. The topics discussed in these meetings were recorded in a shared document, which served as a guide thorough the whole project development. Additionally, *GitHub* was used as a version control system for code and documentation, and *Trello* was used for task management at each iteration.

1.4 Task planning

The Project was divided into several tasks to be completed:

- **Requirements analysis**: this task, along with the design of the architecture, are considered fundamental in laying the foundation for the proposed application. This task defines the application's requirements, along with its intended goals. To gain a better understanding of the problem definition, references were consulted on the current situation of pollutant emissions in the traffic sector, as well as the measures proposed in the frameworks of *eco-routing* and *eco-driving* to identify the problems and limitations of current systems. The time spent on this task was about 30 hours.
- **Review of related works and existing tools**: in order to better contextualize the development of the application, different approaches and tools for Android devices related to the management of route information on an interactive map, as well as other more general functions, were reviewed, helping to define the Project as it is currently proposed. The time spent on this task was about 50 hours.
- **Design of the architecture**: this task consisted of defining the components and services that would form part of the application, as well as selecting the necessary tools and libraries for its implementation. The time spent on this task was about 20 hours.

- **Development of the user interface**: this task entailed the preliminary design of each application screen, defining its main elements, and its implementation in Jetpack Compose. The time spent on this task was about 50 hours.
- **Development of the map visualization system**: this task involved integrating the chosen library for the map within the project, ensuring that the map was correctly displayed, requesting the necessary location permissions, displaying the user's location correctly, making sure that the routes and markers were placed on the map and managed according to screen events, among other functions. The time spent on this task was about 100 hours.
- **Development of the user authentication system**: this task encompassed planning and implementing the registration and login system, persistently storing the current user and ensuring that the data displayed belongs to them. The time spent on this task was about 20 hours.
- **Integration with the backend**: this task included implementing communication mechanisms with external data sources, as well as testing the application's interface after incorporating these elements to ensure they synchronize correctly. The time spent on this task was about 30 hours.
- **Development of the route recording function**: this functionality of the application is key to providing accurate results in the scores awarded at the end of the routes, ensuring that data is recorded correctly and that the task persists even when other applications are opened. The time spent on this task was about 30 hours.
- **Development of the vehicle factor database**: this addition to the application's functionality was recommended by project members to reduce the noise that mobile devices can introduce into consumption estimates, ensuring that for each vehicle, a correlation value between real consumption and estimated consumption using records is stored. The time spent on this task was about 30 hours.

- **Deployment of the system**: once all the application's components were implemented, and having tested that they integrate correctly, this task was carried out to launch a functional version of the application. The time spent on this task was about 10 hours.
- **Testing and validation of the main functionalities**: as part of this task, experiments were conducted using the application, observing whether it functioned correctly, extracting data from the results obtained to represent and analyse them, and drawing conclusions about aspects to be improved. The time spent on this task was about 30 hours.
- **Update and maintenance of the application**: this task consisted of correcting minor errors present in the application, as well as updating libraries to have more stable functionality. The time spent on this task was about 20 hours.
- **Documentation of the project**: this task was carried out concurrently with the previous ones throughout the Project, including external documentation and internal documentation embedded in the application's code. The time spent on this task was about 120 hours.

As can be seen in the Gantt chart of Figure 1.1, the project was divided into 13 main tasks, dedicating approximately 540 hours to its development. It is worth highlighting that during the month of August, progress was hindered until the final week due to vacation periods. However, significant milestones were achieved afterwards, including the execution of vehicular tests in the city of Cáceres with the application at the end of September. The resulting data was meticulously analyzed to validate its accuracy and subsequently compiled in a comprehensive test repository.

Throughout the months of January to July, the project underwent thorough internal and external documentation. Commencing in December, the synthesis of this extensive information was initiated, laying the groundwork for writing this document.

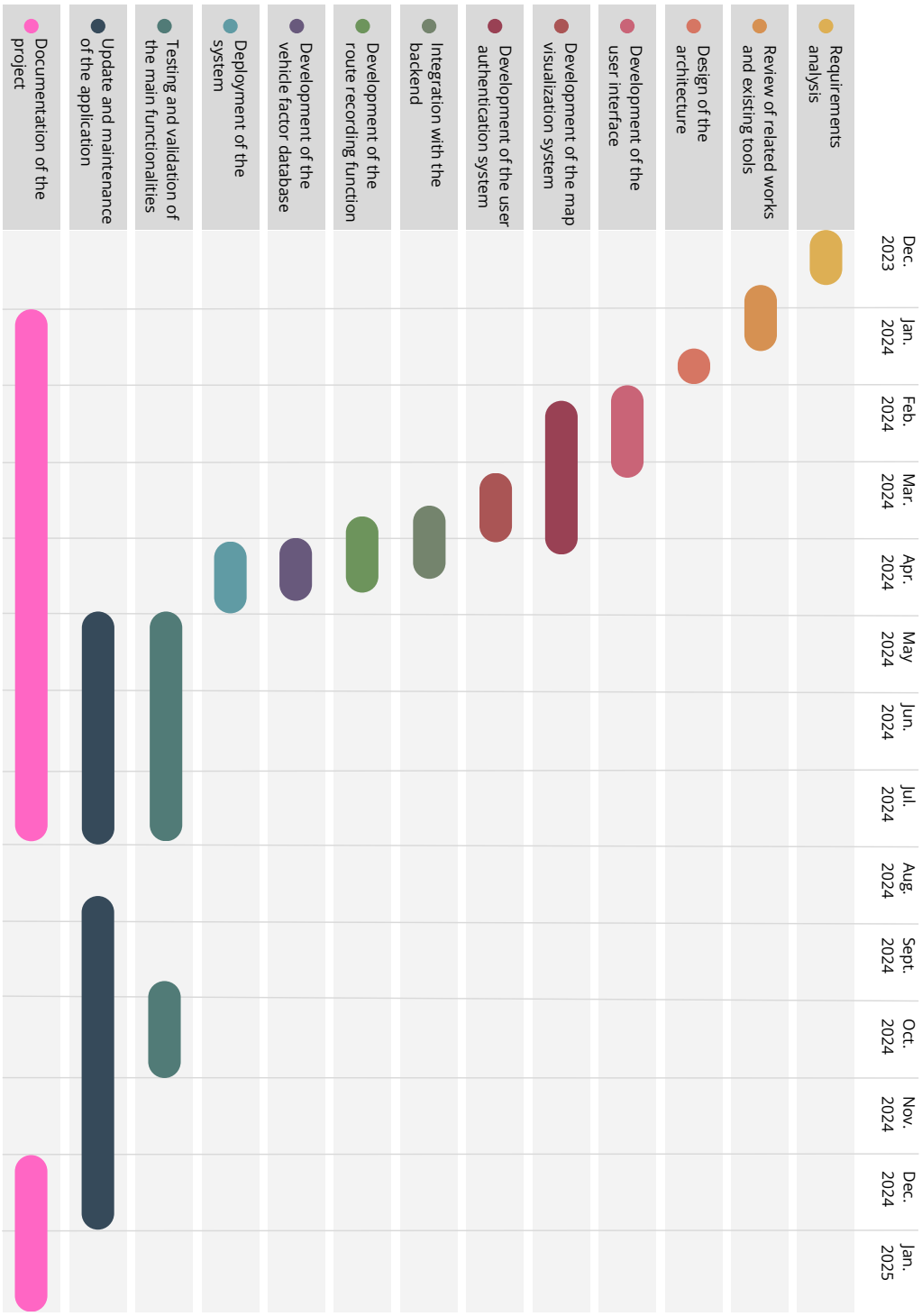


Figure 1.1: Gantt Diagram presenting the schedule of the tasks of the Project.

1.5 Document structure

The remainder of this Report is structured as follows:

- **Chapter 2.** This chapter provides an introduction to the fundamental concepts required to understand the purpose and approach employed in the development of the Project.
- **Chapter 3.** This chapter presents the results of the analysis and design phases of the project. The requirements and actors involved, along with their primary use cases, are described. The architecture design is also detailed, including its components, services, and tools used for implementation. Furthermore, the main patterns and principles considered for the architecture and design of the mobile application are outlined, along with the libraries employed to apply them.
- **Chapter 4.** This chapter describes the implementation of the application, providing a detailed account of its main components and the expected workflow of each screen.
- **Chapter 5.** Two exemplary use cases of the developed tool are presented, illustrating its functionality and application.
- **Chapter 6.** A brief overview of the tools available for obtaining routes is presented, with a focus on those that aim to reduce emissions and save fuel, comparing their features and functionalities with the developed system.
- **Chapter 7.** This chapter discusses the primary challenges encountered during the project, along with the solutions adopted to address them. The thesis concludes with a presentation of the technical and personal conclusions drawn from the project's development, and potential avenues for future work are proposed to enhance the developed tool.
- **Appendix A.** This appendix provides a detailed guide for installing the developed application on an Android device.
- **Appendix B.** This appendix provides a comprehensive guide on how to use the developed application on Android devices, thoroughly examining all the primary features and functionalities to ensure a seamless user experience.

Chapter 2

Background

Contents

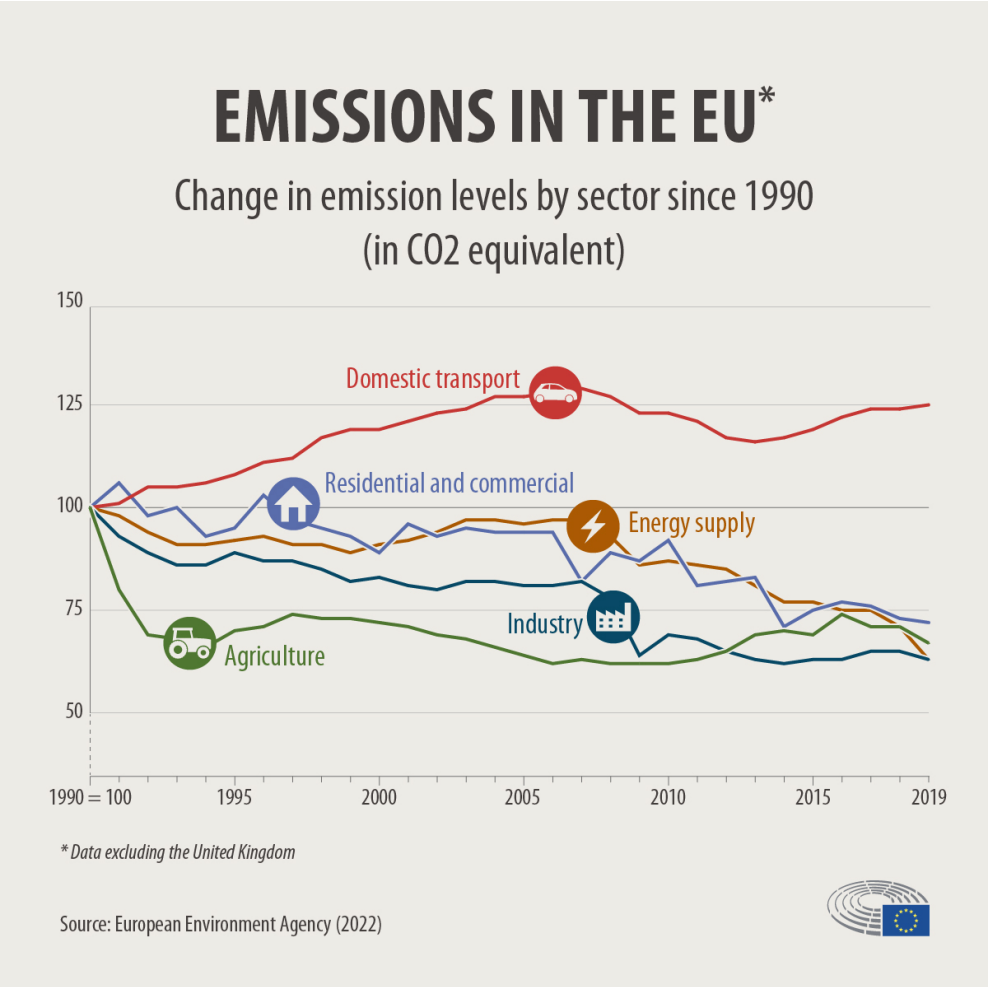
This chapter provides an overview of the context and significance of the thesis in the current landscape, outlining the background that led to its development, as well as some of the fundamental concepts explored throughout this document.

2.1 General concepts

In this section, essential concepts will be explained to facilitate an understanding of the characteristics of the proposed system.

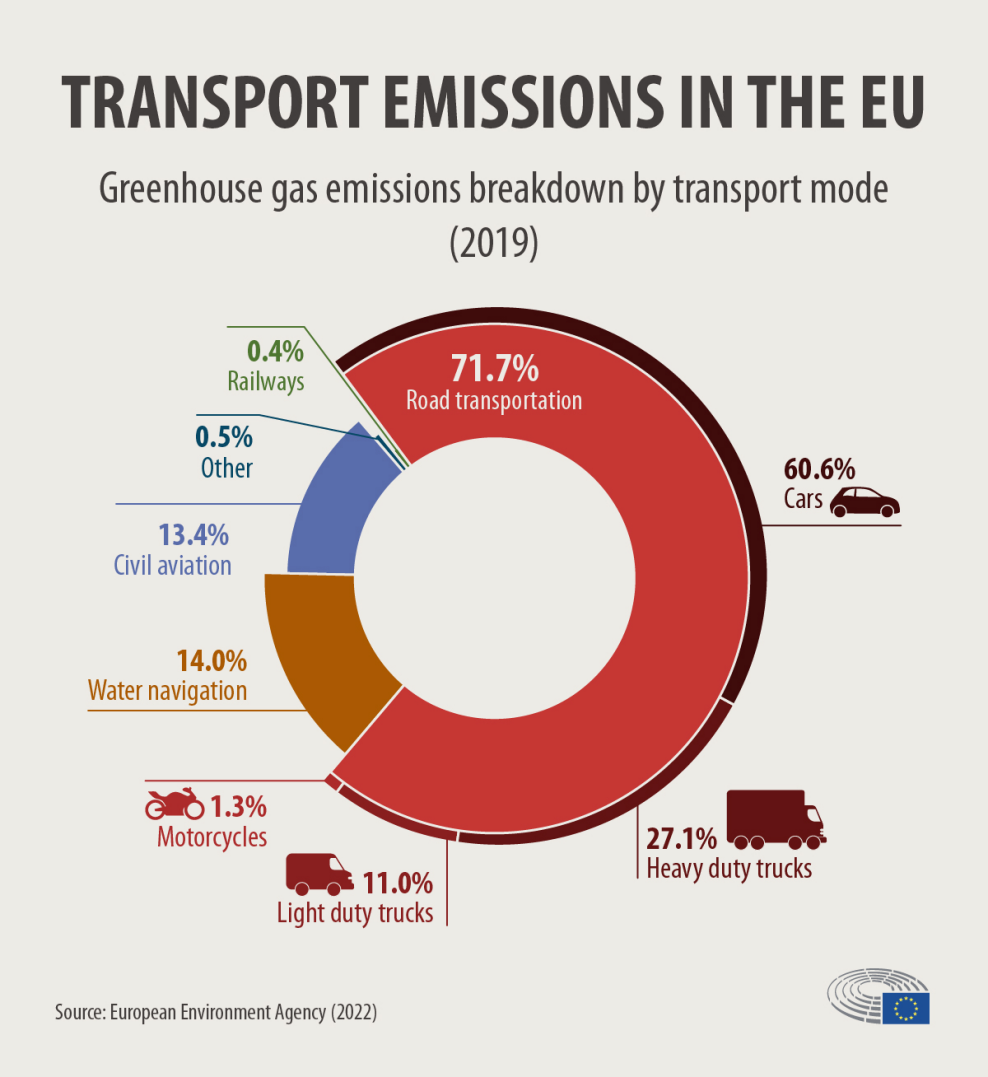
2.1.1 Current situation of road transport

Currently, greenhouse gas emissions (GHGs) and, consequently, global warming are increasing at an alarming rate. In this regard, the European Parliament has published information compiled over the past few years, which reveals that transportation is responsible for approximately one-fifth of GHG emissions in the European Union, making it the only sector where these numbers have not decreased since 1990. Furthermore, within this percentage, 72% of emissions in 2021 were attributed to road transportation, with cars accounting for 61% of these emissions (see Figure 2.1).



(a) Chart with the evolution of CO2 emissions in the EU by sector.

This data highlights the importance of taking measures to reduce fuel consumption and pollutant emissions in road transport. The European Union has proposed several



(b) Pie chart with the ratio of transport emissions by type in the EU.

Figure 2.1: Diagrams of transport emissions in the EU. Source: [3].

of them to address this issue, such as the ban on the sale of new internal combustion engine cars and vans from 2035 onwards [4], with the aim of reducing greenhouse gas emissions by 55% in 2030 and achieving climate neutrality by 2050 [5], in other words, even if some emissions are produced, they will be low enough to be offset by the carbon capturing effect of vegetation and soil. Other countries have also set similar goals to reduce their emissions [6].

In line with Spain's commitment to achieving a carbon-neutral transportation sector by 2040 [7], additional mitigation plans are being implemented, including the promotion

of urban mobility on foot, by bicycle, and other active transportation modes, as well as the enhancement and expansion of the public transportation network with the aim of introducing electrified vehicles or those that utilize greenhouse gas emission-free fuels, such as biomethane.

The implementation of low emission zones is a measure that has garnered significant attention in recent years as a means of mitigating the adverse effects of atmospheric pollution in urban areas. By restricting the circulation of more polluting vehicles in urban areas, low emission zones aim to reduce the exposure of the population to harmful gases, with its efficacy demonstrated through its success in various regions. In Europe, for instance, 73% of urban vehicle access regulations (UVARs) have been implemented in low-emission zones [8], with the primary objective of reducing nitrogen oxide levels (NO_X) in urban centres.

In Spain, the creation of low emission zones has been formalized through Royal Decree 1052/2022 [9]. This decree establishes the declaration of low emission zones in cities with more than 50,000 inhabitants, with a focus on delimiting the routes on which an intervention is considered essential. Furthermore, the decree prioritizes spaces such as the vicinity of schools, hospitals, and nursing homes, recognizing the heightened vulnerability of these sectors of population to the adverse effects of atmospheric pollution. By reducing exposure to air and noise pollution derived from traffic, low emission zones in Spain aim to improve air quality and public health in downtown areas.

The escalating demand for transportation has led to a growing interest in alternative modes of transportation, such as electric vehicles, which are gradually gaining widespread acceptance (see Figure 2.2) and with an increasing number of car brands expressing their intention to transition to these eco-friendly alternatives [11]. These and other zero-emission vehicles are anticipated to be a viable long-term solution for mitigating greenhouse gas emissions. Nevertheless, the current limitations in EV charging infrastructure and range pose significant barriers to their widespread

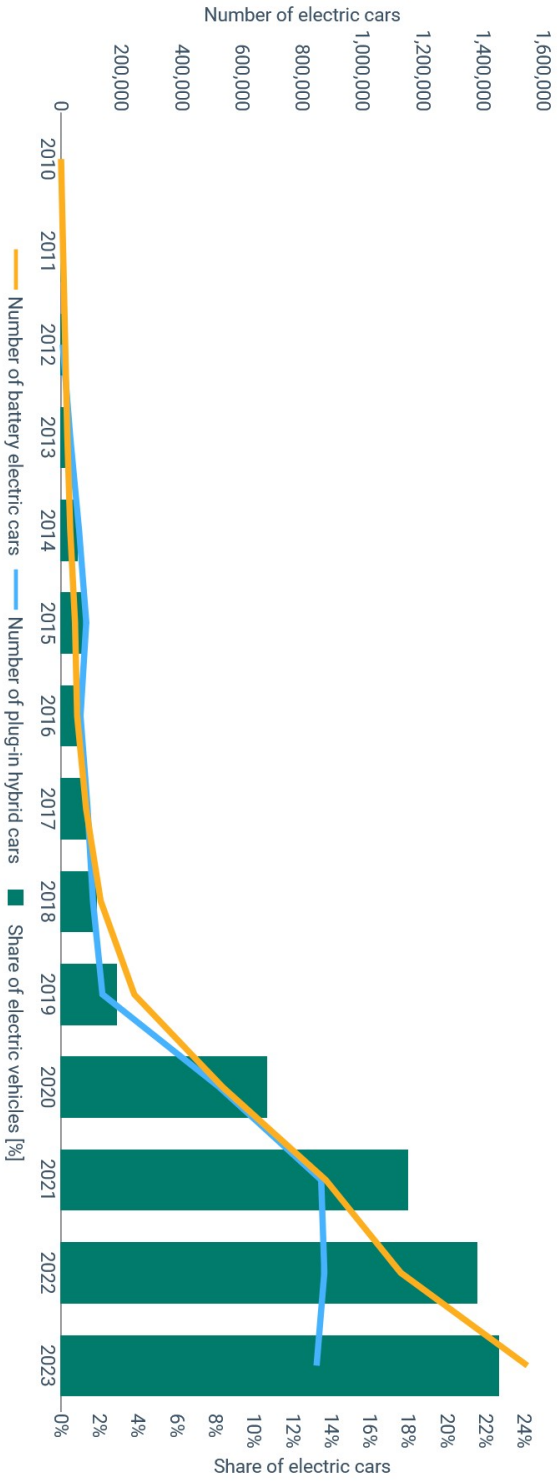


Figure 2.2: Increase in the registration of electric vehicles in the European Union. Source: [10].

adoption, so it is important to take some other measures while these new resources get settled down.

2.1.2 Eco-routing

This concept was first introduced in a study conducted by Ericsson *et al.* in 2006 [12], which proposed a navigation system aimed at reducing fuel consumption and CO_2 emissions by optimizing routes based on detected driving patterns. To achieve this, a collection of real trips was used to model the typical fuel consumption of each street in the city of Lund, Sweden, categorizing them according to various characteristics such as traffic flow outside and inside peak hours. To evaluate this approach, 109 trips longer than 5 minutes from the database were used as examples, obtaining the best route for each one and comparing it with the original route, revealing that 46% of the trips in the studied city can be optimized as they are not the most eco-efficient and that approximately 8.2% of fuel can be saved per trip using an optimized navigation system as a reference to provide this type of route.

Although this approach can be advantageous, it also has the drawback of potentially increasing travel time or resulting in routes that cover greater distances. In the work conducted by Kono *et al.* [13], a similar system was proposed that generated routes taking into account factors such as traffic information, topography, and vehicle data. By comparing the data of the most eco-efficient route with the fastest route, it was observed that an average fuel savings of 9% was achieved, although the arrival time increased by an additional 9%. Another study with similar characteristics to those previously described, conducted by Ahn and Rakha [14], concluded that the routes recommended by *eco-routing* systems can help save between 3.3 and 9.3% of fuel compared to conventional systems, and that while it is possible to reduce the distance of the route, it is not always possible to reduce travel time.

Navigation systems that employ *eco-routing* techniques to minimize the consumption of returned routes are also referred to in some studies as *Green Navigation*

systems. While these systems can play a crucial role in reducing consumption and, consequently, emissions, they also present another inherent problem, namely the management of traffic flow to avoid congestion in certain areas. A study conducted by Perez-Prada *et al.* [15] analyses the impact of using these systems on CO_2 and NO_X emissions in the city of Madrid, concluding that solely focusing on reducing emissions increases the population's exposure to NO_X by 20.2% and travel time by 28.7%. This is due to the fact that these systems propose shorter routes downtown, resulting in a 16.4% increase in traffic volume in these areas and a 13.5% reduction in the rest of the region.

2.1.3 Eco-driving

This term is used to describe an efficient management of vehicles to save fuel consumption by covering the same distance in trips, making better use of vehicle technology and offering among its main advantages the reduction of greenhouse gas emissions, fuel savings, and increased safety and comfort during driving [16].

According to the golden rules of *eco-driving* [17], there are 5 fundamental principles to respect: anticipating traffic flow, maintaining a constant speed with few revolutions per minute, using the highest gear possible in each situation, checking tire pressure monthly and before each long trip, and saving extra energy of the vehicle by turning off devices like the air conditioning if possible.

However, for drivers to be able to carry out this practice in their daily lives, it is necessary for them to receive specialized training beforehand. In a study conducted by Andrieu and Saint Pierre [18], a separation was made into two groups of drivers: those who only received general advice and those who received specific training by taking guided courses with experts in *eco-driving*, concluding that the latter achieved better results by being better oriented and motivated to carry out these practices.

In the work of Jeffreys *et al.* [19], the behaviour of 1056 drivers was observed with respect to their fuel consumption over 7 months in Australia, comparing the results

before and after being trained in *eco-driving* techniques, discovering that with this training, fuel consumption was reduced by 4.6%.

Another study conducted by Coloma *et al.* [20] on *eco-driving* in the city of Cáceres convened vehicle tests, knowing that it has little congested traffic, with the purpose of evaluating whether the result would be different from other cities with a higher volume of vehicles, concluding that the use of these techniques allows for a savings of 17% in gasoline vehicles and 21% in diesel vehicles, regardless of the type of route and road, although in exchange, the travel time increases by around 7.5%. It was also observed that the shortest route is usually considered the most ecological, implying that distance has a significant impact on fuel consumption.

On the other hand, this driving style also presents certain limitations. A study conducted by Garcia Castro *et al.* [21] reveals that, whereas the application of *eco-driving* yields positive effects in scenarios characterized by low or medium traffic demand levels and an increasing number of eco-drivers, the opposite holds true in scenarios marked by both a high number of eco-drivers and traffic demand, where emissions exhibit a significant increase.

2.2 Previous works

This thesis focuses on the significance of reducing fuel consumption in road transport in the current era, as it is a ubiquitous aspect of daily life and a primary source of greenhouse gas emissions. In this context, collaboration has been established with the **Eco-Traffic APP National Project** [22], a research initiative managed by the TRANSyT research center of Universidad Politécnica de Madrid in conjunction with other research groups from various Spanish universities, and funded by the Ministry of Science, Innovation and Universities of Spain. The primary objective of this project is to develop a free mobile application that assists car drivers during their routes, enabling them to enhance their energy efficiency and mitigate their environmental impact.

Prior to joining the main project team, one of the primary contributions made was

to establish the foundation of the application's routing operation, selecting the most suitable tools to integrate into the architecture, and developing a functional prototype of the application in web format. This prototype incorporated only the essential functions to request data on optimal routes and visualize them on an interactive map, and the results obtained from it were compiled into an article for the "7th EAI International Conference on Intelligent Transport Systems" (EAI INTSYS 2023 [23]), which was presented online by the co-supervisor of this work and published in Springer [24].

As a member of the main project team, the functionality of the application has been designed and implemented, refining the previous architecture in accordance with the indications of the rest of the team. This has resulted in a functional version capable of providing users with the ability to register and authenticate, specify their vehicles to be taken into account during the calculation of consumption, request optimal routes and visualize them on an interactive map, record their routes and obtain a score with tips to reduce emissions, visualize previous results in history and derived statistics, among other functionalities.

The primary contribution presented in this thesis is the design of the mobile application and the implementation of its functionality, synchronizing with the external server responsible for processing user requests. This approach has achieved a scalable, easy-to-maintain solution with high availability.

Chapter 3

Analysis and design

Contents

This chapter will provide an overview of the preliminary analysis conducted prior to the project development, including the architecture that was designed along with its individual components. It will further discuss the choice of tools evaluated for implementation, along with the design and architectural patterns considered to improve the overall quality.

3.1 Analysis

This section will outline both the functional and non-functional requirements, present a description of the main actors, and include use cases along with their diagrams.

3.1.1 Functional and non-functional requirements

Tables 3.1 and 3.2, respectively, provide lists of the principal functional and non-functional requirements identified for the system.

Table 3.1: Functional requirements.

Nº	Description
FR1	The system should enable the creation and access of user profiles, ensuring that their application activity is associated with their accounts.
FR2	The system will provide users with the capability to link their vehicles to their profile, either by specifying the exact model or through a broader category. Users will have the option to mark a vehicle as a favourite for easy preselection or remove it if it is no longer needed.
FR3	The system must enable users to acquire the optimal routes from their current location to their chosen destination, displaying pertinent information including travel time, distance in kilometres, and estimated consumption.
FR4	The system will present the data within its position on an interactive map, allowing users to interact with its elements.
FR5	The system should incorporate a feature to locate addresses and points of interest in order to display them on the map.

Continued on next page

Table 3.1: Functional requirements. (Continued)

FR6	The system will track the parameters of each user's chosen route and provide a score at the conclusion, along with suggestions for improvement based on their driving performance. All information will be recorded on a list for future reference.
FR7	As users log their routes, the system will generate statistics for each profile.
FR8	Users will have the ability to submit feedback to developers, offering suggestions for improvement across various elements.

Table 3.2: Non-functional requirements.

Nº	**Description**
NFR1	The system should be user-friendly for everyone, requiring no programming expertise.
NFR2	The graphical interface must be simple and intuitive to use.
NFR3	The application ought to be customizable according to the user's preferences, including their chosen language and interface theme.
NFR4	The system is required to retain and present information in the most realistic manner possible.
NFR5	The system is required to ensure the protection and confidentiality of its users is preserved.
NFR6	The system should be both scalable and adaptable, enabling the straightforward addition of new functionalities.
NFR7	The system needs to be resilient to faults.
NFR8	The system must deliver the intended results in a reasonable amount of time.

3.1.2 Description of the actors

In the development of the system, two distinct types of actors have been taken into account, which will now be outlined.

- **Users**: this type of actor is a person interested in obtaining information on the routes they take from a starting point to an endpoint, regardless of their level of knowledge based on maps and topography.

  For this objective, they will utilize the system to choose their own vehicle, which will enhance the accuracy of the estimation, and they will select the most suitable route from the optimal options provided by their query between their current position and their selected destination.

  After choosing a route or opting to bypass this step, users have the ability to register their journey and receive a performance-based score that serves to motivate their improvement. In addition, they gain access to continuously updated statistics to maintain engagement.

- **System administrators**: this category of actor is tasked with ensuring the system operates smoothly, by updating its components and rectifying errors. To achieve this, they incorporate new vehicles into the database, allowing for a more accurate application experience for users, perform component updates and maintenance tasks, and review user feedback to enhance the app's performance.

3.1.3 Description and Diagrams of Use Cases

Listed below are the identified use cases for the two actors discussed in the preceding section.

- **Users**: Table 3.3 displays the Use Cases linked to the user role.

Table 3.3: Use Cases for users.

Nº UC	Description
01	Create a profile
02	Login to the system
03	Close session
04	Manage vehicles
05	Complete a route
06	Review previous routes
07	View statistics
08	Send reviews

- **System administrators**: Table 3.4 presents the Use Cases related to the responsibilities of a system administrator.

Table 3.4: Use Cases for system administrators.

Nº UC	Description
09	Add new vehicle models
10	Restart the system
11	Review system statistics and user comments

Figure 3.1 illustrates the system's Use Case Diagram, while the accompanying tables present key information regarding the four most significant Use Cases: (1) UC01 (see Table 3.5); (2) UC02 (see Table 3.6); (3) UC04 (see Table 3.7) and (4) UC05 (see Table 3.8).

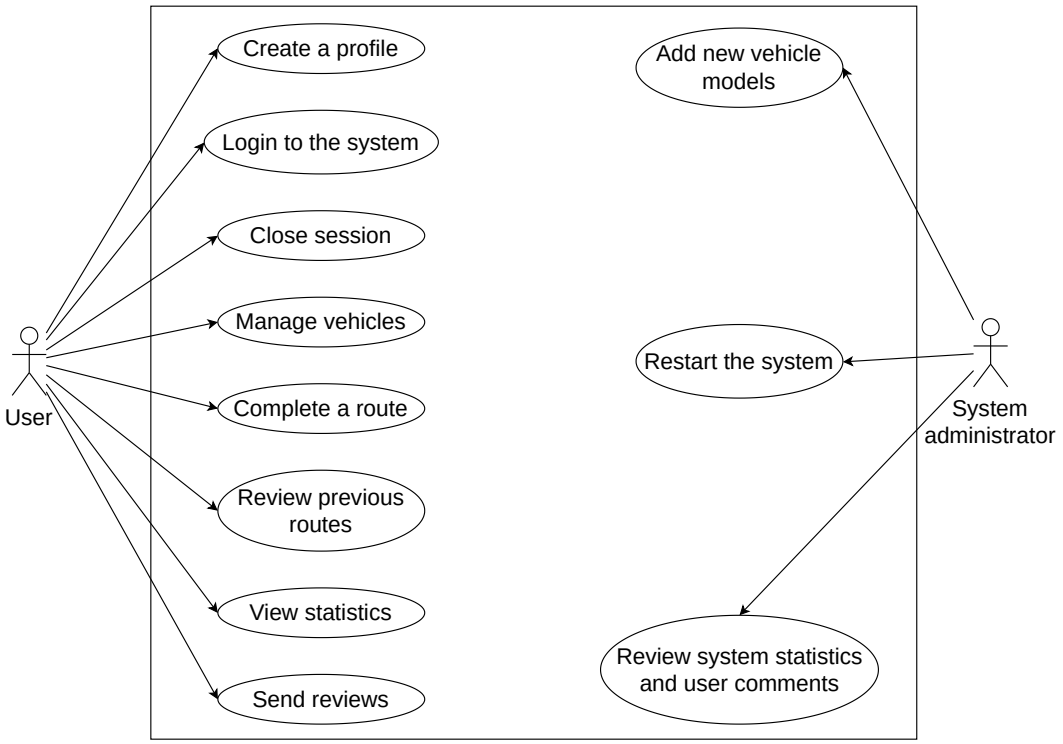


Figure 3.1: Use Case Diagram.

Table 3.5: Use Case 01's Description.

UC Nº01	**Create a profile**	
Actor	User	
Precondition	To have an internet connection for making the request to the server.	
	Not having any session started in the application.	
Normal sequence	**Step**	**Action**
	1	Open the application.
	2	Press the button with the label "Sign up" in the Welcome screen.
	3	Fill in the required fields (name, email, password, birthday, driving license year).

Continued on next page

Table 3.5: Use Case 01's Description. (Continued)

	4	Select the gender.
	5	Press the button with the label "Sign up" to complete the form.
Postcondition	The user is logged in on the current device, without having to log in again until they choose to log out.	
	An account is generated for the user, with which they can log back in and to which their activity data in the application will be linked.	
Comments	Step 4 can be skipped as it is an optional field.	

Table 3.6: Use Case 02's Description.

UC Nº02	**Login to the system**	
Actor	User	
Precondition	To have an internet connection for making the request to the server.	
	Not having any session started in the application.	
	Having an account previously created to login.	
Normal sequence	**Step**	**Action**
	1	Open the application.
	2	Press the button with the label "Login" in the Welcome screen.
	3	Fill in the email and password fields.
	4	Press the button with the label "Login" to complete the form.

Continued on next page

Table 3.6: Use Case 02's Description. (Continued)

Postcondition	If the email and password provided are correct, the user is logged in on the current device, without having to log in again until they choose to log out. Otherwise, an error message is shown.

Table 3.7: Use Case 04's Description.

UC Nº04	**Manage vehicles**	
Actor	User	
Precondition	To have an internet connection for making the request to the server.	
	Having a session started in the application.	
Normal sequence	**Step**	**Action**
	1	Open the application.
	2	Open the menu and select the button with the label "My vehicles" and a car icon.
	3	Press the button with a + symbol to add a vehicle.
	4	Select the model of the car.
	4.1	To choose a generic model, select one of the four available.
	4.2	To select a specific brand and model, use the search bar and choose the most appropriate one.
	5	Fill in the age and travelled kilometres fields.
	6	Press the button with the label "Save" to complete the form.
	7	Perform an operation to modify the state of the created vehicle.

Continued on next page

Table 3.7: Use Case 04's Description. (Continued)

		7.1	To mark it as favourite or unmark it, press the heart-shaped icon.
		7.2	To delete it, press the trash-shaped icon.
Postcondition	After creating the vehicle, it will appear on the screen and be eligible for routes.		
	After marking the vehicle as favourite, the heart-shaped icon will appear filled and the vehicle will be automatically selected for routes.		
	After deleting the vehicle, it will disappear from the list.		
Comments	If the application is already open, skip step 1, and if it is already on that screen, skip step 2.		
	Step 5 can be skipped as it is an optional field.		
	Step 7 can be skipped if the user does not want to perform an operation on a vehicle, and if they want to perform an operation on a vehicle already created, steps 3 to 6 can be skipped.		
	If a vehicle has been used on a registered route, step 7.2 can no longer be performed on that specific vehicle.		

Table 3.8: Use Case 05's Description.

UC Nº05	**Complete a route**	
Actor	User	
Precondition	To have an internet connection for making the request to the server.	
	Having a session started in the application.	
	Having at least one vehicle in the list of the screen "My vehicles".	
	Step	**Action**

Continued on next page

Table 3.8: Use Case 05's Description. (Continued)

Normal sequence	1		Open the application.
	2		Open the menu and select the button with the label "Map" and a map icon.
	3		Select the route parameters before starting a route, being the vehicle, the number of passengers and the number of bulks.
	4		Select the destination point.
		4.1	For selection from the map interface, click on the area the user wants to reach.
		4.2	For textual selection, type the address of the point in the search field and choose the closest available option.
	5		Click the button to generate the routes.
	6		Start recording a route.
		6.1	If routes were requested before, select one of the generated routes and click the button to start recording.
		6.2	To start recording without previously requesting routes, click the button with a camera icon.
	7		To pause the route, click the pause button.
	8		To continue with the route, click the resume button.
	9		To finish the route, click the button with a flag icon.
	10		View the results.
	11		Introduce the consumption indicated by the vehicle to improve accuracy in upcoming routes.

Continued on next page

Table 3.8: Use Case 05's Description. (Continued)

Postcondition	After viewing the results, the route will be saved along its results to be displayed in the history.
	After introducing the consumption of the vehicle in step 11, the correlation value of the estimated consumption for that vehicle will be adjusted.
	The vehicle associated with the route cannot be deleted after being saved in the history.
Comments	If the application is already open, skip step 1, and if it is already on that screen, skip step 2.
	For step 3, one passenger is taken into account by default, and if a vehicle was selected as favourite, it will already be selected.
	Steps 4 and 5 can be skipped if the user decides to start recording without requesting routes.
	Steps 7 and 8 are optional, and they can be repeated multiple times in the same sequence before step 9.
	Step 11 can be skipped, either by deciding to not introduce the consumption, or if the correlation value is well adjusted for that vehicle.

3.2 Design

In this section, the design of GRETA App will be detailed by discussing the complete architecture, the primary patterns used to improve its coherence and operation, together with the tools considered and selected to be used in the development of the proposed system.

3.2.1 Architecture

Figure 3.2 illustrates the system's architecture. The functioning of each component will subsequently be described, along with the interactions occurring among them.

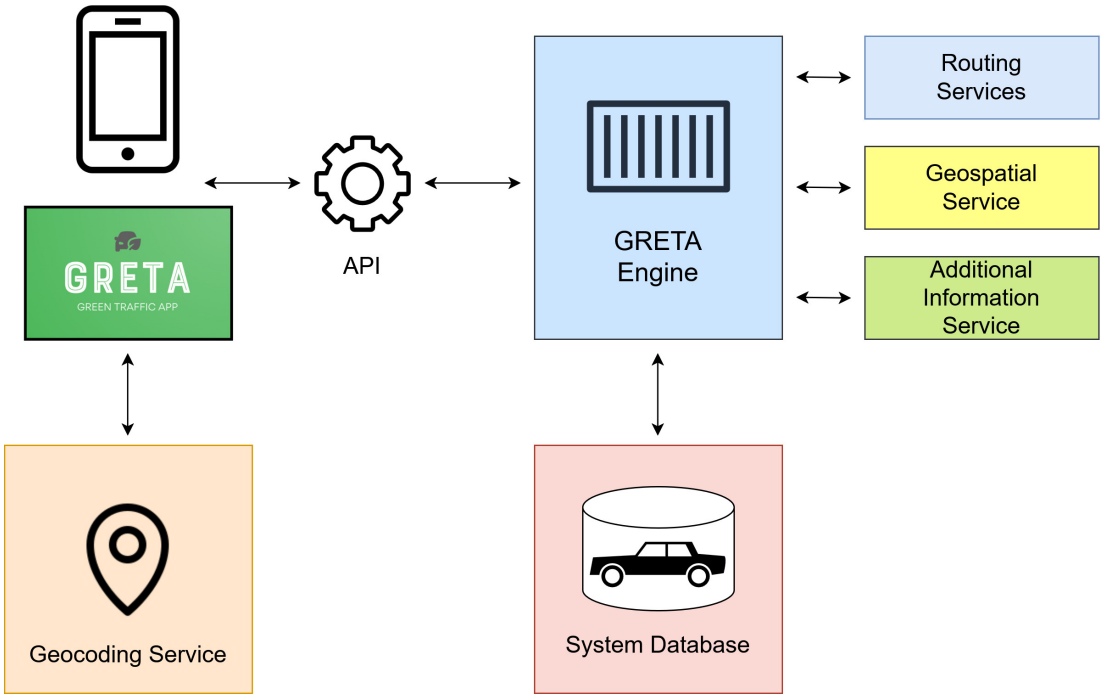


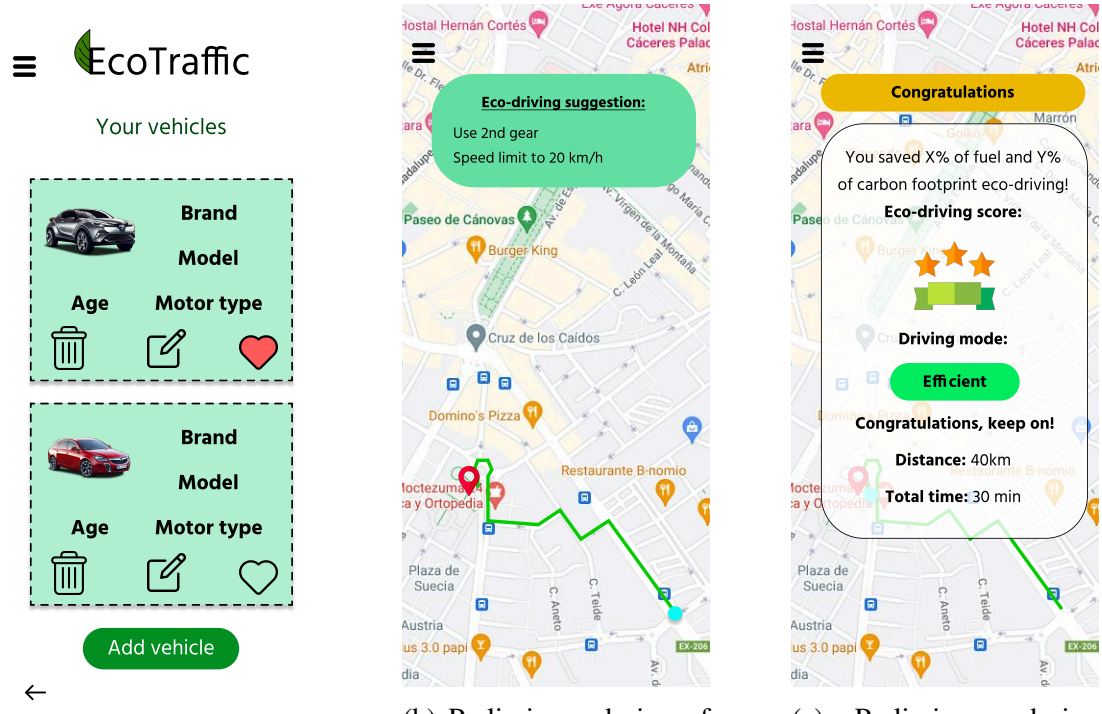
Figure 3.2: Architecture of the proposed system.

- **GRETA App**: this component is the primary focus of this thesis, representing the major contribution made. User interactions are managed through the app, enabling users to interact with the rest of the system.

  The application's interface is composed of multiple screens that enable users to create a profile, log in, and access the main features. These features include adding vehicles to their accounts, which can be selected for routes, and configuring additional travel parameters, such as the number of passengers in the car and the weight of additional bags or elements (approximately 5kg) to account for excess mass in calculations.

  Once these parameters are set, users can access the map screen, which resembles a Google Maps view, to select a destination point and obtain the best routes

and estimated consumption for the trip. By selecting a route, users can start recording their trip and obtain the results after completion, including guidance on how to improve and registering statistics. A rough representation of the intended interface is illustrated in Figure 3.3.



(a) Preliminary design of the vehicles screen.

(b) Preliminary design of the map screen showing the best route.

(c) Preliminary design of the map screen after finishing a route.

Figure 3.3: Initial mockup of GRETA App's main screens.

For the development of this component, the Jetpack Compose Android framework has been utilized, which provides a modern and efficient way to build user interfaces. Additionally, the Osmdroid library has been employed for the visualization of data on the map, allowing for the display of geographic information and routes. Furthermore, the Retrofit library has been used for communication with the server, enabling the application to send and receive data in a robust and scalable manner. Other libraries and tools have also been integrated to support the development of this component, which are explained in more detail in Subsection 3.2.3.

- **Geocoding Service**: this component is responsible for geocoding, which involves processing user queries about addresses to retrieve the closest matching locations along with their corresponding coordinates. The primary objective of this component is to facilitate the search for a route to a destination by automatically retrieving the coordinates of the desired location, thereby eliminating the need for manual placement on the map.

  To achieve this, the component utilizes the **Nominatim API** [25], a geocoding service that provides accurate and up-to-date location data. The component communicates directly with the GRETA application to handle user requests for addresses, and subsequently returns the retrieved location data to the app, which then displays the search results, allowing the user to select the most suitable location, and is then placed on the map to facilitate route planning.

- **Routing Services**: this component enables seamless interaction with various OpenStreetMap-based routing services, which accept multiple coordinate pairs to generate routes that pass through the specified points. In response to each request, the component returns a dictionary containing the processed route information, including distance and estimated time of arrival. Each route is represented as a list of coordinates that define the path in a sequential and linear manner.

  Currently, the system utilizes **openrouteservice** [26], **OSRM** [27], and **GraphHopper** [28] services, with the first two deployed within Docker containers [29, 30, 31] and the last one being accessed through their public API. The component is designed to accommodate additional services, provided they adhere to the standardized dictionary format for returned routes, thereby facilitating extensibility and flexibility.

- **Geospatial Service**: the service incorporates a database containing elevation data in meters for the terrain considered within the system, derived from a subset of NASA's **SRTM** dataset [32] that provides topographic information for

Spain. The primary function of this component is to process requests that retrieve elevation values based on provided coordinates, enabling the storage of this data at each node and its subsequent utilization in calculating road slopes.

Specifically, the 30-meter precision dataset files have been utilized, which are divided into separate files that cover distinct areas of the Iberian Peninsula. Each file represents the topography of a specific region, as illustrated in Figure 3.4.

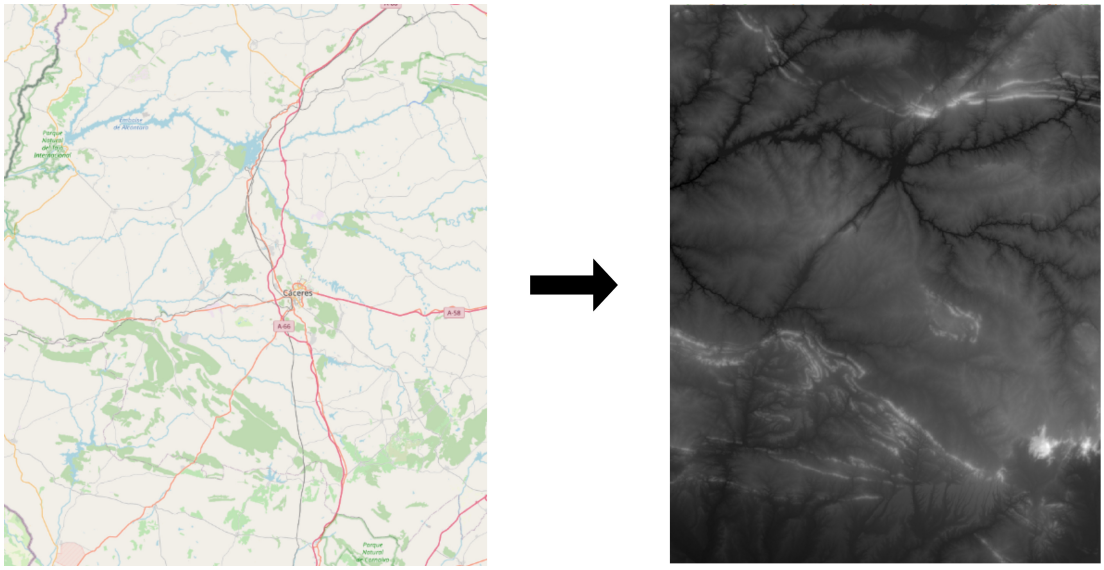


Figure 3.4: Example of the representation of a SRTM30m file.

To calculate the slope between two points on the route, the formula presented in Equation 3.1 has been employed.

$$slope = \frac{destination\_altitude - source\_altitude}{distance} * 100 \tag{3.1}$$

In this equation, *destination_altitude* and *source_altitude* represent the altitudes of the destination and origin points, respectively, while *distance* denotes the distance between the two points. By applying this formula, the percentage slope between two points is obtained, which is subsequently used to calculate fuel consumption based on the road's inclination.

To facilitate the operation of this component, the **Open Topo Data framework**

[33, 34] has been utilized, thereby enabling the seamless integration of new areas as required for future testing endeavours.

- **Additional Information Service**: this component is tasked with retrieving supplementary information about the points along the routes, including maximum speeds and lane counts, to augment the route data and enhance the accuracy of the consumption estimation.

  To achieve all of this, a version of **Nominatim** has been deployed within a Docker container [35], enabling the execution of coordinate-based queries that retrieve additional information from OpenStreetMap, which can be directly integrated into the graph to enrich the route data.

- **System Database**: the architecture's database serves as a central repository for storing all the information necessary to support the proper operation of the system. A detailed representation of the primary stored entities and their interrelationships is provided in Figure 3.5, which illustrates the main classes and their connections.

  – Firstly, the **User** class serves as a representation of a user's account, encapsulating essential attributes such as their name, email, password, birthday, and driving license year. Additionally, users have the option to provide their gender, which is not a mandatory field.

  – The **Vehicle** class represents a specific vehicle model, which can be selected by users to create a personalized instance in their profile. This class contains attributes such as the vehicle's name, engine type, empty mass, and other factors that influence the consumption calculation. Furthermore, the class allows for the optional inclusion of a URL containing the vehicle's data and additional links to display an image of the vehicle's appearance.

    The class incorporates a method specifically designed to recalculate the A-factor during the consumption calculation process. This is necessary

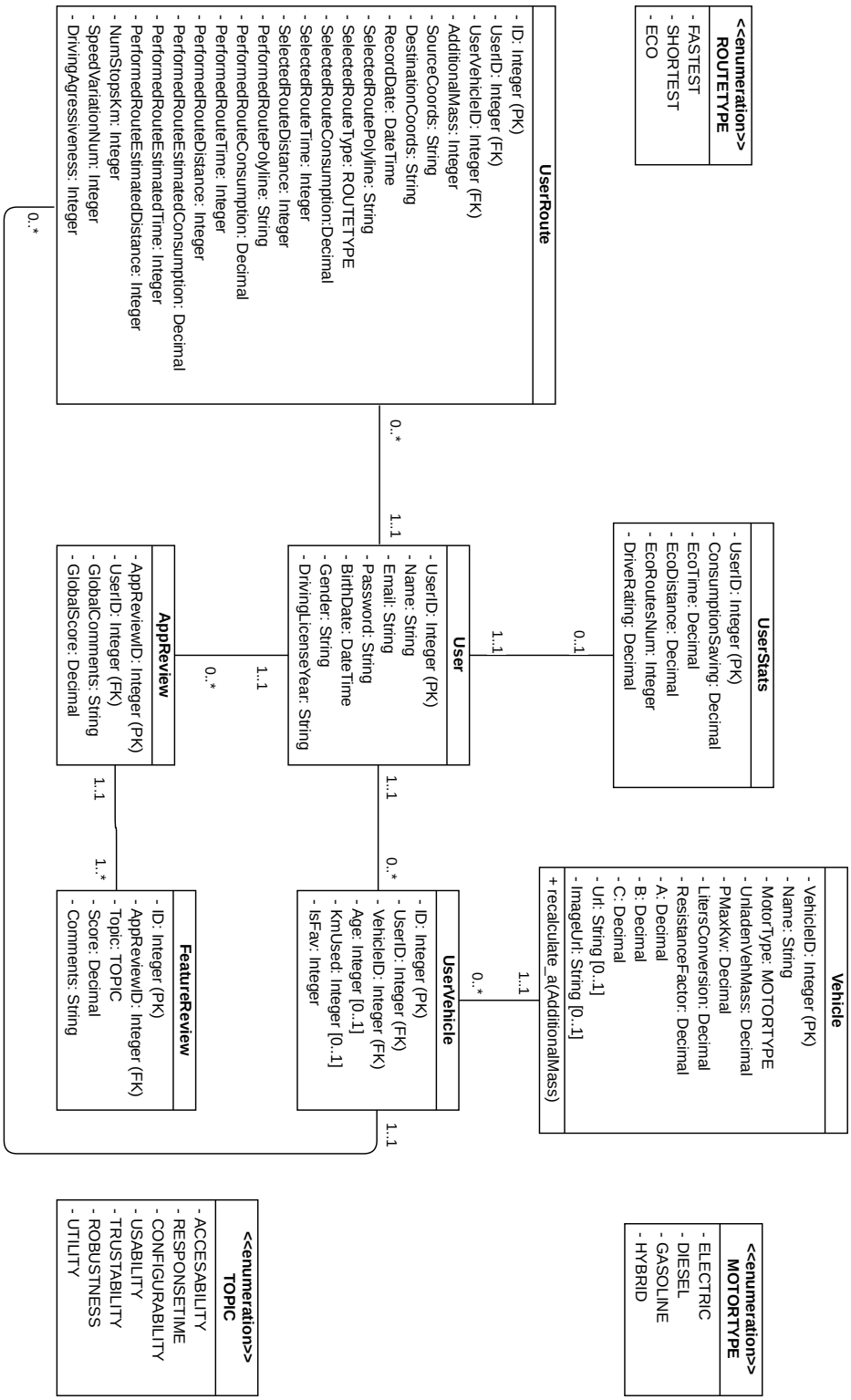


Figure 3.5: Class Diagram of the System Database for GRETA App.

because the A-factor is a dynamic parameter that changes in response to the additional mass loaded onto the vehicle, ensuring that the consumption calculation remains accurate and up-to-date.

- The **UserVehicle** class represents a specific instance of a vehicle that has been created within a user's profile, containing attributes such as the vehicle's age and kilometres travelled. Additionally, a flag is used to indicate whether the vehicle is marked as a favourite.

  A user can have zero or multiple UserVehicle instances associated with their profile, with a minimum of one required to successfully perform a route, while each UserVehicle instance is uniquely tied to a single user. In addition, a UserVehicle can only reference a specific vehicle, while multiple UserVehicle instances can be created from a vehicle model.

- The **UserRoute** class represents a route that has been registered by a user, storing relevant data such as the additional mass, source and destination coordinates, and the date of registration. If a route was previously selected before recording, its associated data is saved; otherwise, default values are used. Additionally, the class stores the data obtained from the route taken, including the route coordinates, estimated consumption, time taken, and distance travelled. Furthermore, an estimate of the last three values is also included, simulating a best-case scenario with ideal speeds and no braking, allowing for a comparison with the actual result obtained. Finally, the class stores the scores awarded by the system, providing a comprehensive record of the user's route.

  A **UserRoute** instance is uniquely associated with a single user, and a user can have zero or multiple instances recorded. Conversely, a **UserRoute** instance is linked to a single **UserVehicle** instance, which represents the vehicle used for that specific trip, while a **UserVehicle** instance can be used in zero or multiple **UserRoute** instances.

- The **UserStats** class represents the statistical data associated with a user's profile, which is updated after each journey to reflect the impact of the user's decisions and performance on their overall statistics.

  The statistics are tied to a specific user, and a user may or may not have associated statistics, as they are only recorded for the first time after the user completes a route. This allows the system to track the user's progress and provide insights into their performance over time.

- The **AppReview** class represents a review submitted by a user, containing data such as the user's overall rating and comments. A review is uniquely associated with a single user, and a user can submit zero or multiple reviews.

  This enables the system to collect and store user feedback, providing valuable insights into the user's experience and opinions about the application.

- The **FeatureReview** class represents a rating on a specific topic, as defined in the TOPIC enumeration, storing the score and comments provided by the user.

  A **FeatureReview** instance is uniquely associated with a single **AppReview** instance, and an **AppReview** instance can have multiple **FeatureReview** instances, with one instance for each topic type registered in the TOPIC enumeration. This allows for a detailed breakdown of the user's feedback, with separate ratings and comments for each specific feature or topic.

A MySQL Docker container [36] has been utilized for this component, providing a robust and reliable solution for data storage and management, ensuring data integrity, and facilitating communication between components.

- **GRETA Engine**: the *GRETA Engine* service constitutes the core component

of the architecture, serving as the primary interface for interactions with other components. This program is responsible for receiving and processing requests from the **GRETA application**, and subsequently responding to these requests by communicating with other components as necessary to fulfil the required functionality.

When a user request necessitates data retrieval or modification from the database, such as account creation, vehicle selection, marking a vehicle as a favourite, or retrieving past trips, the *GRETA Engine* service will establish communication with the **System Database** component to facilitate the required data access or updates.

Within the primary use case of the application, namely the calculation and recording of optimal routes to estimate consumption, various interactions occur with distinct components dedicated to this process, each serving a specific purpose.

1. Initially, a request is made to the **Routing Services** to retrieve all possible routes between the source and destination points specified by the application's request. The retrieved routes are then fragmented into nodes and stored in an in-memory graph, which contains the coordinates, heights (obtained from the **Geospatial Service**), and other relevant data for each resulting point.

2. Subsequently, the nodes are processed to establish direct connections, thereby shaping the routes. This process involves calculating the slope of each section (derived from the difference in heights of the two points comprising the junction), as well as obtaining additional road data such as maximum speeds and the number of lanes (obtained from the **Additional Information Service**).

3. Upon completion of the graph, an operation is performed to determine the optimal routes by minimizing the following criteria:

- Route distance.
- Journey time.
- Consumption during the route.

Although the first two criteria can be minimized using Dijkstra's algorithm [37], the calculation of consumption depends on a global understanding of the path, rather than solely on the data between immediate nodes or relationships.

Consequently, an alternative approach is employed, wherein all possible routes resulting from the graph are processed using a library developed by the transport research team of the University of Oviedo. This library estimates consumption as a function of road inclination, speed, and other factors, thereby classifying routes according to their parameters and returning the best options to the mobile device.

The resulting graph can be visualized in Figure 3.6, where the optimal paths are differentiated. Notably, some paths may possess multiple labels if they are considered optimal in more than one aspect.

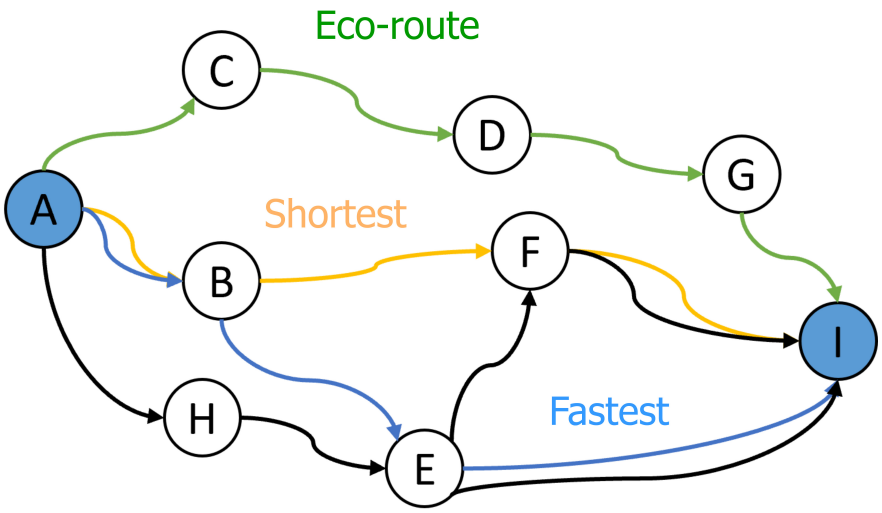


Figure 3.6: Example of the resulting graph after retrieving route data.

At this point, the user selects the desired route, and the system awaits the response from the mobile device to continue with the process.

4. Following the acquisition of driving data from the mobile device, a similar process is employed to determine the actual consumption of the route. Additionally, a simulation of the final route is conducted under ideal conditions, i.e., adhering to recommended speeds for each section and avoiding significant pauses, to compare the results with those obtained.

   Based on the acquired data, two metrics are evaluated and presented as the final score:

   (a) **Driving smoothness**: this parameter serves as an indicator of the number of sudden braking and acceleration events that occurred during the journey, measured according to the time spent travelling with an acceleration greater than 0.1m/s^2.

   (b) **Speed variability**: this parameter assesses the variability of the vehicle's speed during the journey, calculated as the percentage of time spent travelling with an acceleration greater than 2.5m/s^2.

   These metrics were carefully selected by TRANSyT members to ensure that the scoring system accurately reflected the *eco-driving* objectives and motivated users to improve their driving style.

   Upon reviewing the results, the route is saved in the history, and the user's statistics are updated to track their progress.

The component is deployed using Flask [38], a Python library that facilitates the creation of web services in a straightforward and efficient manner. Furthermore, the NetworkX library [39] is utilized for the creation and manipulation of in-memory graphs, in conjunction with several other libraries that enable the full operation of the application's functionalities.

This architecture is an evolution of the one presented in my Bachelor's Thesis [40],

with modifications made to facilitate its integration within an Android application. The decision to adopt a distributed approach was necessitated by the limited computational capabilities of mobile devices, which are incapable of executing complex operations independently. Consequently, the system's operations have been distributed across external services that complement the application, thereby enabling the processing and provision of necessary information to support user operations.

The subsequent subsections will concentrate on the design selections made for creating the GRETA App component.

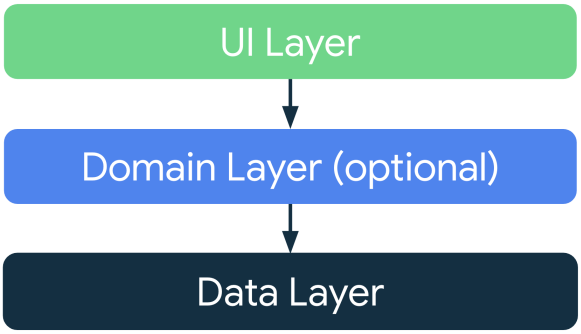
3.2.2 Architectural patterns and design principles

This section will provide an in-depth examination of the primary architecture and design patterns employed in the development of the GRETA App, as well as the rationale behind the key implementation choices.

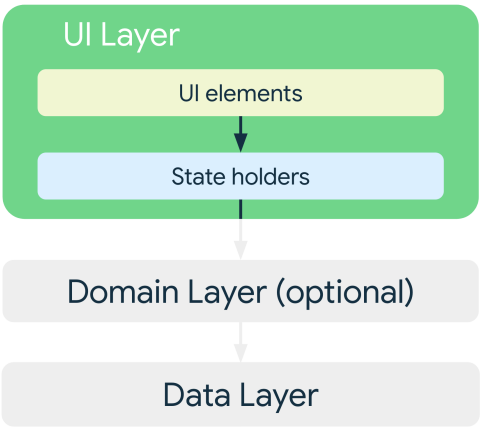
Initially, the decision was made to develop the application for the Android platform, driven by several factors. Firstly, the operational tests had already been planned to utilize Android devices, which would help reduce costs. Additionally, prior experience with developing similar applications on the Android platform had provided a foundation of familiarity with the language and tools used. Furthermore, considering a hybrid approach to create and compile applications for iOS devices would have required the use of the Xcode editor, which is exclusively available on MAC devices, a resource that was not available. Given these circumstances, the focus remained to developing a native Android application.

Android specialists recommend separating the logic of the application into at least two different layers, as can be seen in Figure 3.7.

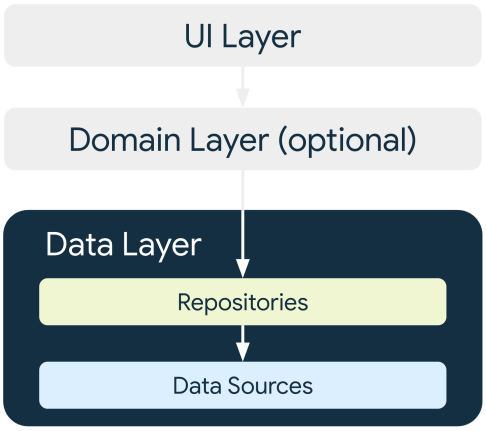
- The **UI Layer** is composed of the elements that enable the application's data to be visually represented in the interface, and ensures that any changes to the data, whether triggered by user interactions such as button clicks or external events like online requests, are accurately reflected in the graphical layout.



(a) Overview of the recommended architecture.



(b) Components of the UI Layer.



(c) Components of the Data Layer.

Figure 3.7: Diagram of the recommended app architecture to follow. Source: [41].

1. On one hand, the interface comprises various **elements that facilitate user interaction**, including, but not limited to, buttons, text boxes, and scrolling lists, which serve as the primary points of engagement between the user and the application. In the case of the app, these would be the methods annotated with the **@Composable** tag, which are displayed as part of the interface.

2. On the other hand, the **State holders** play a crucial role in maintaining the state of the interface by storing and managing relevant information. They are responsible for ensuring that the interface is updated in a timely manner, whenever possible, to reflect the current state. Furthermore, State holders facilitate communication with data providers, handling both

incoming and outgoing data transmissions, and subsequently updating the interface accordingly based on the outcome of these interactions. In the case of the application, these would be the **ViewModel** instances, which are responsible for managing the state of the corresponding screen and performing operations with external sources.

- The **Data Layer** is responsible for orchestrating the execution of external operations, encompassing methods that transcend the interface's immediate scope, and enabling the creation, storage, or modification of data in response to user interactions within the application.

  1. The data layer is comprised of **repositories**, which are interfaces that define a set of operations for managing specific types of data, and abstracting the data source from the rest of the app. For instance, the GRETA Repository is utilized to handle data related to the GRETA Engine. These repositories can be implemented to specify the underlying data source, allowing for flexibility and modularity in the data access.

  2. **Data sources** are specialized classes that serve as bridges between the application and external data sources, providing concrete implementations of the methods defined by the repositories. These classes encapsulate the logic necessary to interact with the external data sources, enabling the application to access and manipulate data in a standardized and efficient manner.

- The **Domain Layer**, although an optional component, typically serves as an intermediary layer that facilitates communication and data exchange between the layers described above, performing any necessary intermediate operations. However, in the case of GRETA App, this layer has been intentionally omitted from the architecture, thereby streamlining the overall design and minimizing complexity.

Some of the fundamental principles that this architecture model adheres to include:

- **Separation of Concerns**: this principle emphasizes the importance of modularizing the codebase by separating distinct concerns into individual components. By doing so, each class is responsible for a specific set of tasks, and it can request assistance from other components when necessary, rather than attempting to handle everything itself. This approach facilitates easier testing, debugging, and fault localization, as issues are isolated to specific components.
- **Data Persistence in UI**: the UI should reflect the state of the data models, and these should be designed to persist even when the UI is destroyed. Although data models are independent of the UI state, they can be terminated by the operating system under various circumstances. To mitigate this, it is recommended to save data models in memory, enabling the application to restore its state in case they are lost.
- **Unidirectional Data Flow**: this pattern ensures a unidirectional flow of data within the application, where the state of the interface is updated in a predictable and controlled manner. In this approach, the data flows in one direction, while the events that modify this data flow in the opposite direction. Specifically, in the context of the application, user interactions trigger events that flow from the interface to the data sources, and the resulting modifications to the state are then propagated from the data sources back to the interface, where the corresponding updates are applied. This one-way data flow pattern helps maintain a clear and predictable data flow, making it easier to reason about the application's behaviour and debug issues.

In addition to the architecture model previously described, other complementary patterns have been employed to further enhance the overall operation and performance of the application. These supplementary patterns work in conjunction with the existing architecture to provide a more robust, efficient, and scalable solution:

- **Dependency Injection**: this pattern has been utilized to simplify the management of dependencies within the application, allowing for the

decoupling of components and the seamless substitution of implementations [42]. Specifically, in the context of the application, this pattern has been employed to define the data sources of the repositories, providing a mechanism for their implementation during runtime.

The approach that has been followed is the one described by the Android specialists to manage dependencies manually [43], using an instantiated container for the entire application, which is responsible for defining the implementations of the repositories and exposing them to the ViewModels of the interface to call their operations. This approach was chosen due to the fact that the application does not rely on a diverse range of data sources, making a manual dependency injection approach a suitable and efficient solution.

Separating the code in this way allows that, for isolated tests, developers don't depend on the randomness that can arise from requests to online services, but they can define implementations of the repositories that return static results to see how they are processed and represented in the interface, as can be seen in Figure 3.8.

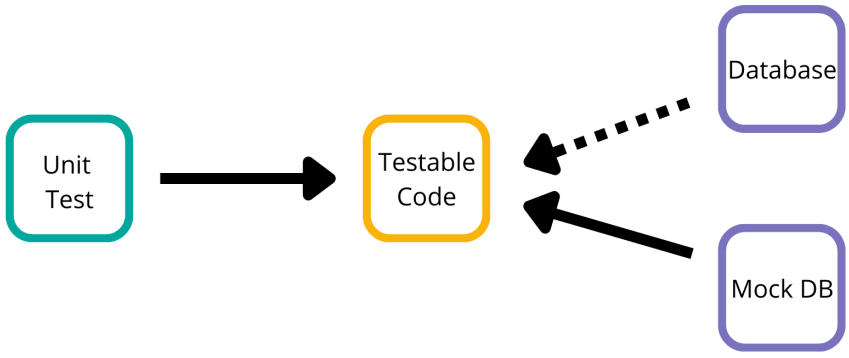


Figure 3.8: Example of dependency injection. Source: [44].

- **Data Access Object (DAO)**: the incorporation of this pattern facilitates a layer

of abstraction between the application and its respective data sources [45], thereby enabling the application to interact with them without necessitating direct access to their underlying structure. Through the definition of DAO classes that encapsulate their operations, the application is empowered to utilize these classes in a standardized and efficient manner, thus ensuring the preservation of data integrity and consistency.

A further notable advantage of employing this pattern lies in its utilization of serialization [46], whereby data is transformed into objects that can be directly manipulated by the application, facilitating effortless data manipulation and representation within the interface. Conversely, these objects can be readily converted back to their original source format, thereby enabling seamless storage in a database or transmission to external services.

- **State Hoisting**: this pattern has been employed to manage the state of the application in a centralized manner [47], enabling the easy sharing of state between different components. By hoisting the state to the lowest common ancestor component, the interface's state can be managed in a single location, reducing the complexity of the codebase and enabling the easy propagation of state changes throughout the composition, as illustrated in Figure 3.9.

  The use of the State Hoisting pattern has several benefits, including the improvement of maintainability, testability and the easy management of state changes.

In terms of the patterns and principles that have been applied in the design of the application's interface, the following are particularly noteworthy:

- **Material Design**: Google's Material Design system [49] has been adopted for the application's interface, leveraging its predefined visual and interaction elements to create a cohesive and visually appealing user experience. By adhering to the guidelines and principles outlined in Material Design, the application's interface has been designed to be intuitive and

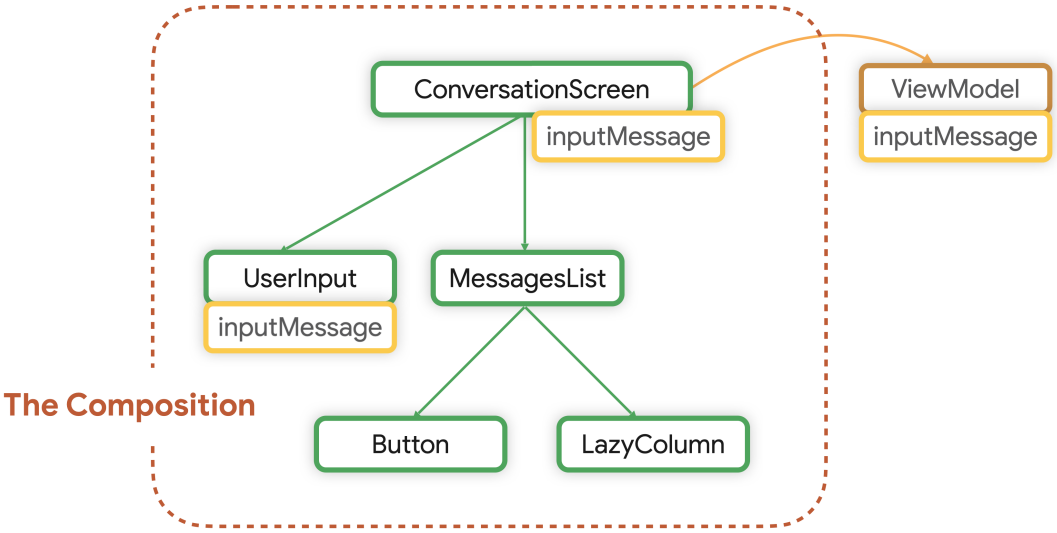


Figure 3.9: Example of state hoisting. Source: [48].

engaging. Specifically, a visual theme has been employed instead of static colours, ensuring a consistent appearance independently of the configuration of the interface for the device.

- **Text Resource Management**: To facilitate the localization of the application and enhance its accessibility, all text strings from the interface have been extracted from the code and consolidated into a separate resource file. This approach enables easy translation of the application into various languages, as Android automatically selects the most suitable resources for the device's current language [50], making it more adaptable to different regions.

3.2.3 Selection of tools

This section will provide an in-depth examination of the primary decisions made regarding the tools utilized during the development process. A comparative analysis of the key characteristics of the main alternatives will be presented, with a focus on ensuring that the selected tools align with the patterns and requirements outlined in the preceding sections.

- **Android Studio**: this is the official Integrated Development Environment (IDE)

created by Google for the development of Android applications [51], which includes multiple native functions such as a code editor with the option to visualize the resulting interface, a debugger, a mobile device emulator for testing application execution, and other essential tools for managing Android projects and compiling applications.

By utilizing Android Studio, developers can benefit from the functionalities it provides for programming Android applications in a more efficient and effective manner, and can also integrate other third-party tools and libraries to enhance the functionality it already provides.

- For the development of the user interface, there are two primary approaches that can be employed:
  - **XML Layouts** is the traditional approach for creating user interfaces in Android [52], which allows developers to define the structure and design of the interface using XML files, or programmatically through View or ViewGroup objects. This approach supports implementation in both Java and Kotlin programming languages.
  - **Jetpack Compose** is a new Android toolkit [53] that enables developers to create user interfaces in a more efficient and declarative manner, utilizing the Kotlin programming language.

  The use of Jetpack Compose is recommended by Android developers, as it provides a more intuitive and comprehensive way of creating user interfaces compared to XML Layouts. This method allows for the rapid and efficient definition of interactive elements, facilitating the creation and maintenance of the interface.

  Jetpack Compose enables developers so they can create user interfaces that are more dynamic, interactive, and visually appealing, thereby improving the overall quality of the application and the user experience.

- **Material Theme Builder**: the Material Theme Builder tool [54, 55] has been employed to create a custom theme for the application, enabling the definition of colours, typography, and other visual elements in accordance with the Material Design guidelines. This tool facilitates the creation of a visually appealing and consistent interface, ensuring a seamless user experience in both light and dark modes.

- For the visualization of geographic data and routes on the map, three main options have been considered:

  1. **Google Maps SDK**: this tool [56] provides a comprehensive set of APIs for integrating Google Maps into Android applications, enabling developers to display maps, customize map elements, and interact with map data.

  2. **Mapbox Maps SDK**: this tool [57] also offers a flexible and customizable mapping solution for Android applications, providing developers with the ability to create custom maps, add interactive elements, and integrate location-based services.

  3. **Osmdroid**: this library provides an open-source mapping solution for Android applications [58], enabling developers to display OpenStreetMap data and customize map elements. Osmdroid is designed to be lightweight and efficient, making it an excellent choice for applications that require basic mapping functionality without the need for additional features.

  A comparison of the main features of each tool to represent maps on Android is presented in Table 3.9. It can be noted that while the Google Maps SDK and Mapbox Maps SDK offer a wide range of features and customization options, in addition to native features for Jetpack Compose, both options are restricted by usage for each client of their respective platforms.

Table 3.9: Table with the main properties of each tool for representing maps in Android.

	Google Maps SDK	Mapbox Maps SDK	osmdroid
Free to use	✓[1, 2]	✓[2]	✓
Platform without restrictions of use	✗	✗	✓
Functions to represent the basic elements of the application on the map	✓	✓	✓
Native support with Jetpack Compose	✓	✓	✗

[1] Google Maps needs to introduce a billing method to use the SDK.

[2] Maps SDK require a token associated with an account to use the service.

In order to ensure that the application can be used completely free of charge and without restrictions, Osmdroid has been selected as the preferred mapping solution. Although this library does not offer native support for Jetpack Compose, it can be manually adapted for use with this toolkit, as it provides backwards compatibility functions with elements created for XML layouts.

- **Retrofit**: this library [59] was selected for communication with the server and other online data sources, as it provides a robust and scalable solution for sending and receiving information. Retrofit is a type-safe HTTP client for Android and Java that simplifies the process of making network requests and handling responses, and by using it, the application can interact with the server in a reliable and efficient manner, ensuring that data is transmitted securely and accurately.

- **WorkManager**: this library has been utilized for managing the recording of routes in the background [60], allowing the application to continue functioning

without interruptions while the route is being recorded.

WorkManager is an Android library that facilitates the execution of background tasks in an efficient and reliable manner, providing multiple methods for observing, cancelling, chaining, and even conditioning tasks to be carried out, for example, only if there is internet connectivity or sufficient battery life.

- **Room**: this library has been selected for managing the local database of the application [61], which stores individual vehicle adjustment data to improve the accuracy of consumption estimates, allowing for efficient and secure data storage and retrieval.

  Room is a data persistence library that allows the creation and management of local databases in Android applications, based on the DAO (Data Access Object) pattern and enabling the automatic implementation of CRUD (Create, Read, Update, and Delete) functions from an interface file with annotated methods.

- **DataStore**: this library [62] has been employed to store the active user in the application persistently after logging in, allowing the user's information to be retained even after the application is closed, eliminating the need to re-enter access credentials.

  DataStore is an Android library that provides a secure and efficient way to store information, offering an alternative to SharedPreferences and an internal database. This library facilitates the management of information in a simple and effective manner, enabling developers to store and retrieve data quickly and securely.

Chapter 4

Implementation and development

Contents

This chapter provides an in-depth examination of the development process of the GRETA application, encompassing the project's structural framework, its primary components, and the workflow dynamics that govern the interactions between each constituent part of the tool. Furthermore, this chapter will elucidate the testing procedures undertaken by the Project team to verify the application's optimal functionality.

4.1 Project structure

The project's structure adheres to the recommended architecture paradigm for Android projects, as outlined in Subsection 3.2.2, maintaining a clear distinction between the *data* and *ui* packages, which contain the classes that handle business logic and the user interface, respectively, as illustrated in Figure 4.1.

The UI package comprises the implementation of the user interface, encompassing screens and constituent elements, and with each screen accompanied by a corresponding ViewModel that orchestrates the interface state in response to user input and operational progress, while also being designed to mitigate the impact of unforeseen errors. For a comprehensive explanation of the purpose and functionality of each screen, the reader is referred to Appendix B.

Conversely, the *data* package contains classes that implement business logic, featuring repositories that encapsulate the implementation of corresponding data sources, thereby streamlining the information management process and promoting a more cohesive and reusable codebase. Among the data sources utilized by the application, some are **external** in nature (such as the *GRETA Engine API* or the *Geocoding Service* within the application's global architecture, as can be seen in Figure 3.2), while others are **internal**, meaning they are integral to the application itself and execute on the device, yet are managed externally as separate processes from the primary one to facilitate organizational simplicity. The internal application components that function as data sources are as follows:

- **User Session Manager**: this component is responsible for managing the user's session, storing the current account information and enabling access to it. Upon initial application use or when a user logs out, a new session can be initiated and stored in this component, which identifies the user in subsequent screens and accesses.
- **Background Recording Worker**: this component is in charge of managing route recording in the background, storing route information and enabling access

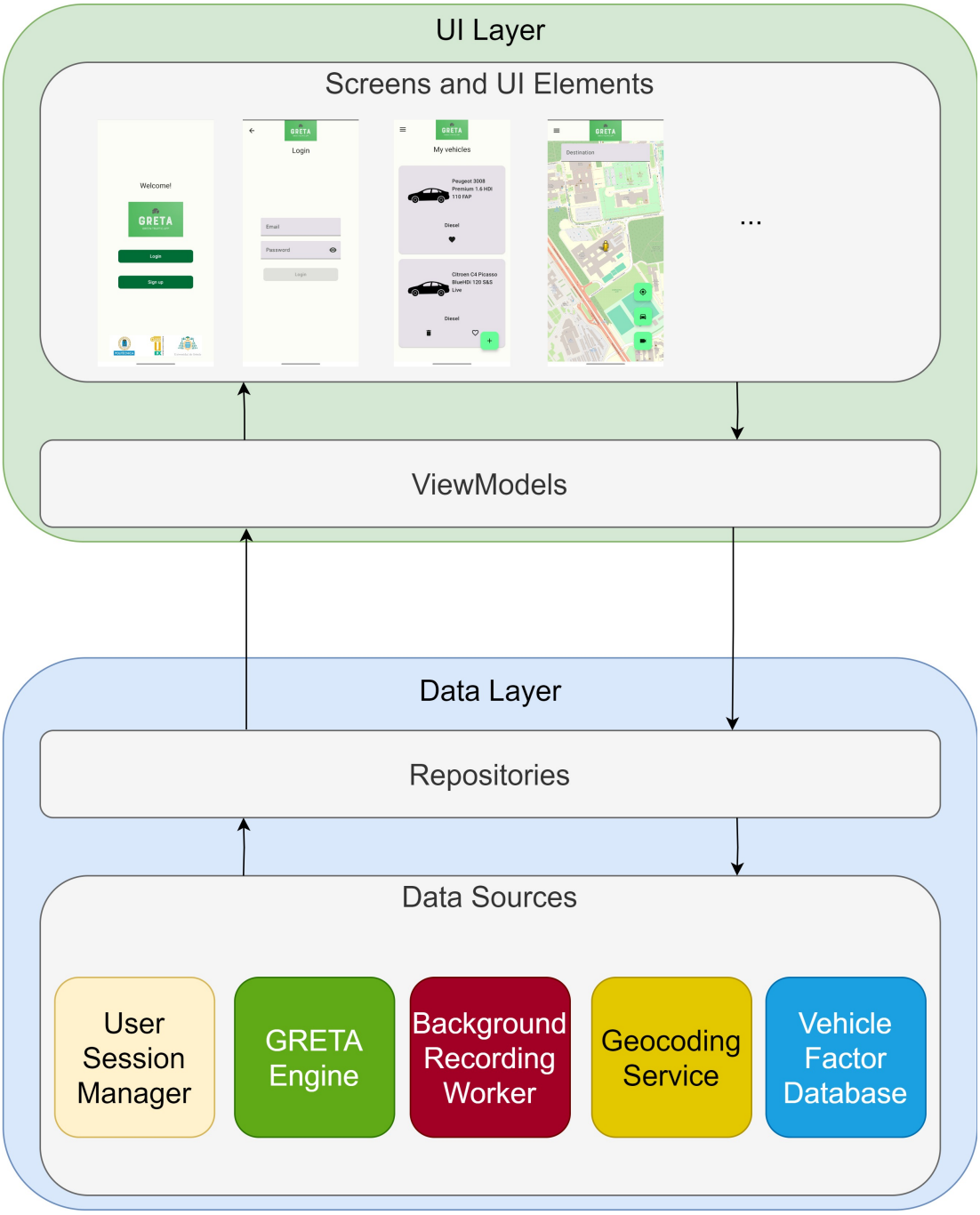


Figure 4.1: Diagram of the structure for GRETA App.

to it for transmission to the central server upon completion and retrieval of corresponding results.

When route recording is initiated, the interface changes to focus on the driver, and a call is made to this component to start recording the route in the

background. A notification is displayed on the phone indicating that the process is active, and the component manages the phone's sensors (GPS, accelerometer, etc.) to obtain relevant information during the journey.

Every second, the following data points are collected and stored in a local file on the phone: the timestamp, coordinates, altitude, speed in km/h, acceleration, and accelerometer positions, with the latter serving as a fallback for speed calculation in the event of unavailable GPS signals. Furthermore, the recording can be paused and resumed at will, and can be finalized by pressing a button when the user reaches their destination, which sends the information to the central server and displays the corresponding results.

- **Vehicle Factor Database**: this component takes care of storing and managing the adjustment factors for vehicles on the phone where the application is installed. When a route is completed and the result is displayed, the user is prompted to enter the fuel consumption displayed on the vehicle, and the adjustment factor for the vehicle in question is calculated. This factor is stored in this component and is used for both pre-route estimates, when displaying route data, and for calculating the actual fuel consumption of vehicles in future recorded routes.

  When the user-inputted vehicle result is very similar to the calculation made by the application, it is determined that further user input is no longer necessary, and the adjustment factor is considered definitive for that vehicle model. Conversely, if the results do not match, the user is continued to be prompted for fuel consumption data until a sufficient level of similarity is achieved.

The code developed for the application is hosted in the GitHub repository of the project [63], with a version control system that allows tracking its evolution and the features implemented in each version.

4.2 Workflow

This section provides an overview of the workflow of the GRETA application, spanning from the initialization of the application to the completion of a route recording and the visualization of the results.

4.2.1 Application startup

Figure 4.2 illustrates the complete workflow when the application is opened, and it is briefly explained below. Upon launching GRETA App, a request is sent to the *User Session Manager* to determine if an active user session exists. If not, the Welcome screen is displayed, where the user can choose to either log in by entering their credentials or register if they do not have an account. However, if the user has previously logged in to the application, they are directly redirected to the main screen.

When the user chooses to log in, a request is sent to the central server to verify the entered credentials. If they are incorrect, an error message is displayed on the interface. If the credentials are correct, a session is initiated for the user, and they are redirected to the application's main screen.

On the other hand, if a new account is created, a request is sent to the central server to register the new user. If the registration and all completed data are correct, a session is initiated for the user, and they are redirected to the main screen too.

4.2.2 Map screen

Figure 4.3 illustrates the workflow when using the Map screen to record a route in GRETA App, and is explained in greater detail below.

1. Upon initializing the screen, a request is sent to the *User Session Manager* to retrieve the current user, and a request is sent to *GRETA Engine API* to receive the data corresponding to the current user, including the vehicles that can be selected.

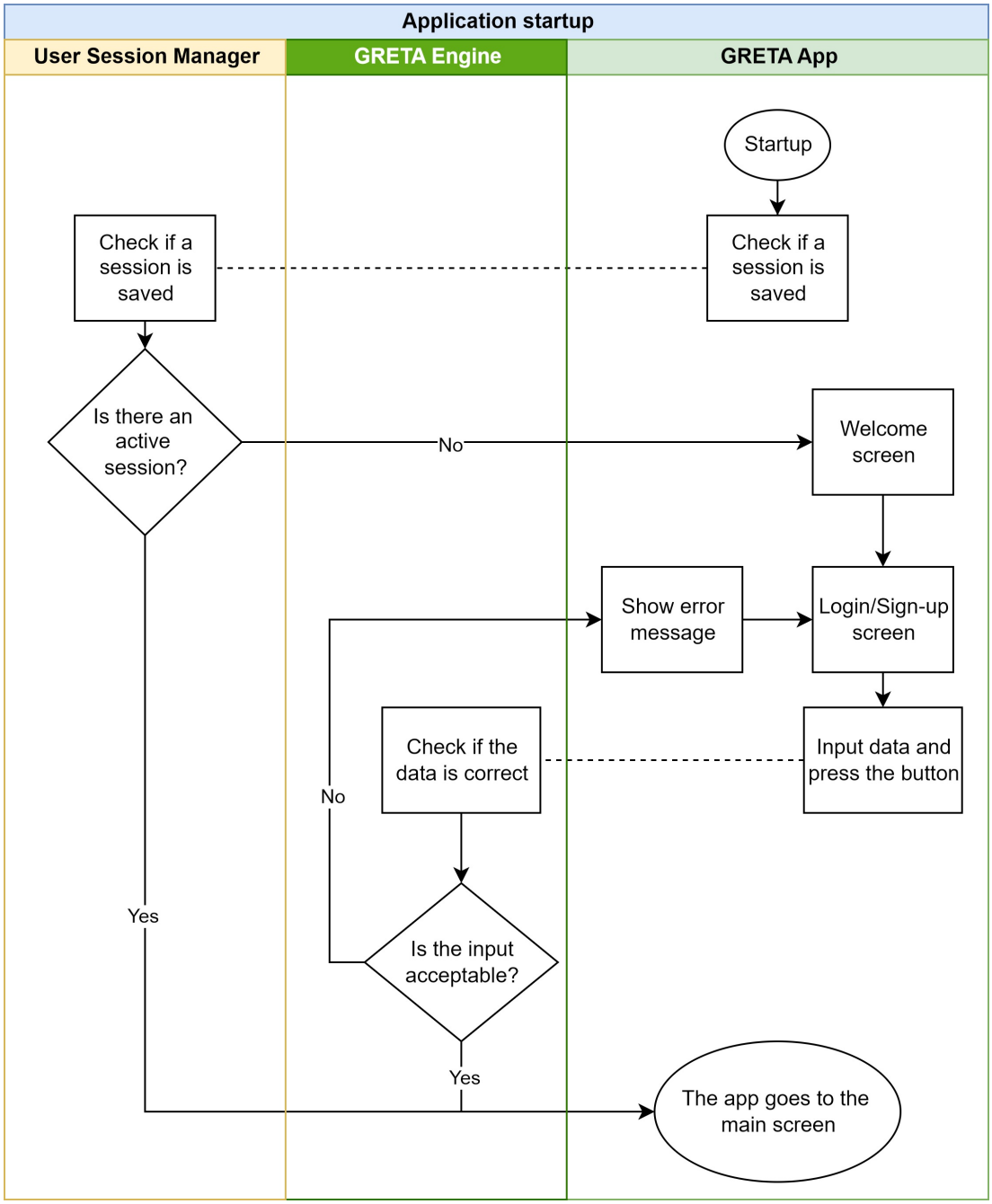
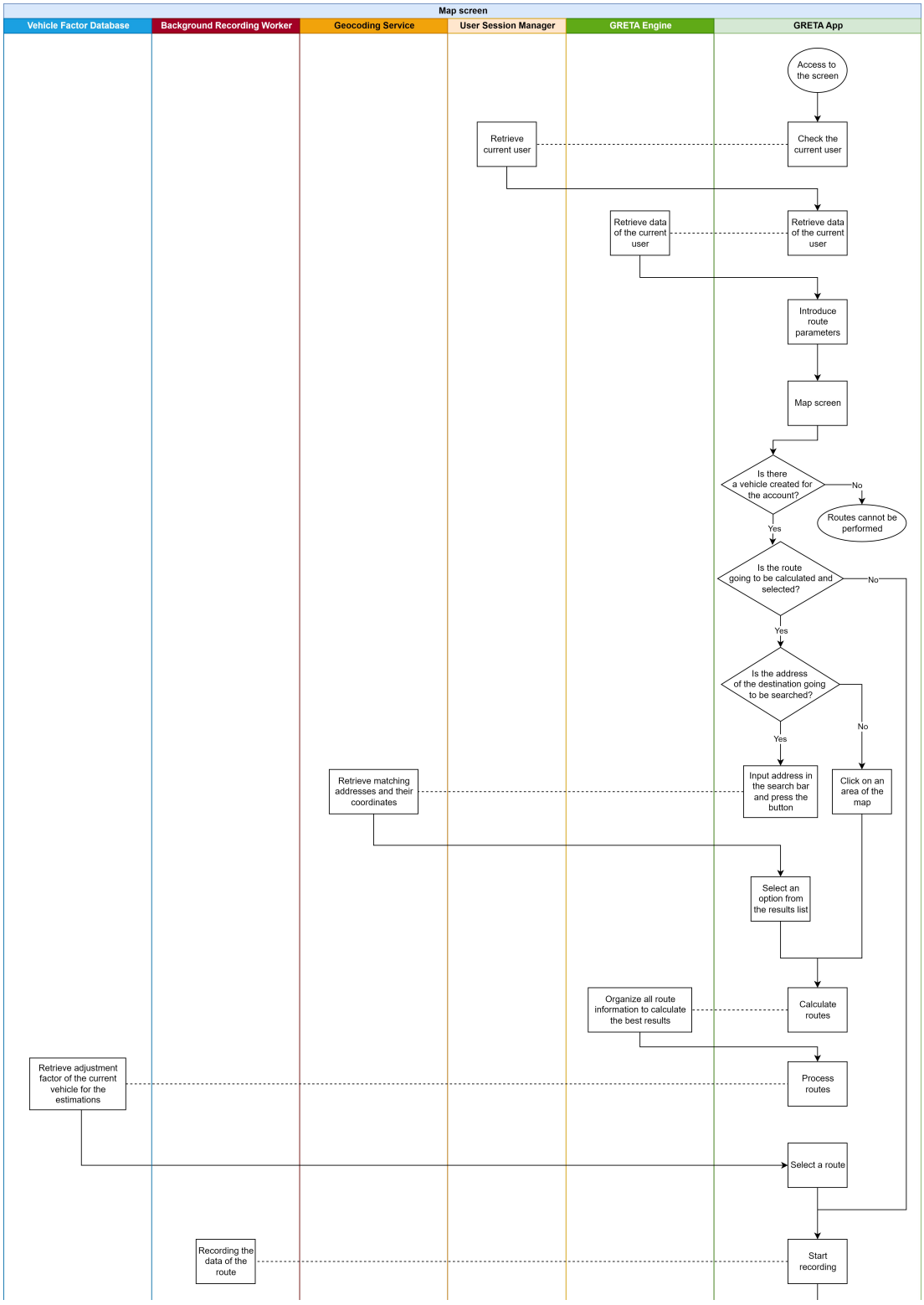
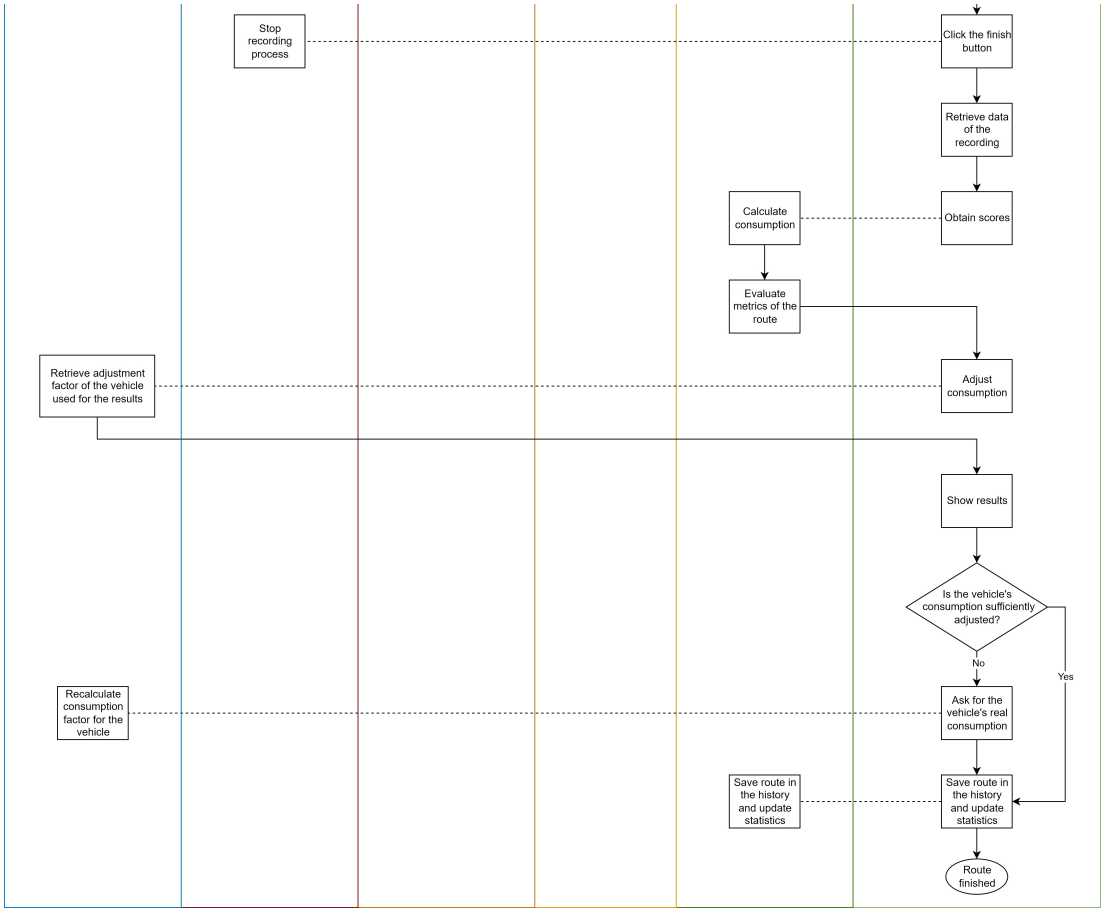


Figure 4.2: Diagram of the workflow when GRETA App is opened.

2. Once the user data is retrieved, the user is prompted to enter the route parameters, including the vehicle, passengers, and luggage being transported. If no vehicles are registered, a message is displayed indicating that a route cannot be recorded without a registered vehicle, prompting the user to go to the vehicle registration



(a) First part of the diagram.



(b) Second part of the diagram.

Figure 4.3: Diagram of the workflow when using the Map screen in GRETA App.

screen.

3. After entering the route parameters, the user can request a route by selecting a destination or start recording directly. If a direction is searched in the search bar, a request is sent to the *Geocoding Service* to obtain the coordinates of the addresses that match the entered address, displaying them in a list, and upon selecting one, the map view is centred on the corresponding area and a marker is placed. A destination can also be selected by clicking on the map. After selecting the destination, a request is sent to *GRETA Engine API* to obtain the best routes to follow, and then the vehicle consumption factor is requested from the *Vehicle Consumption Database* to adjust the estimated consumption, displaying all of the routes on the map and allowing the user to select one, after which the

recording process begins.

4. Upon starting the route, a request is sent to the *Background Recording Worker* to start the recording in the background, and a notification is displayed on the phone indicating that the process is active. Every second, the route data is collected and stored in a local file on the phone, and the recording can be paused and resumed at will. Upon completing the recording, another request is sent to the *Background Recording Worker* to finalize the process, the driving data is collected from the file, and the results are obtained with a request to *GRETA Engine*, adjusting the displayed consumption with the factor of the selected vehicle and making a request to the *Vehicle Factor Database* prior to displaying them.

5. After displaying the route results, it is checked if the vehicle consumption is already sufficiently adjusted, and if not, the user is prompted to introduce the consumption provided by the vehicle, adjusting the vehicle factor in the *Vehicle Factor Database*. Finally, the results of the recorded route are saved in the history and the user's statistics are updated.

The internal workflow of the central server for route processing is described in the **GRETA Engine** component of Subsection 3.2.1. It is worth noting that, due to the load that the application's requests can have on network capacity, the coordinate compression algorithm described in Chapter VI of the work of Ghosh *et al.* [64] has been implemented on the routes to reduce the size of the application's requests and responses, allowing for greater efficiency in data transmission.

4.2.3 Other screens

The remaining screens follow a very similar workflow, utilizing the *User Session Manager* to retrieve the current user, and the *GRETA Engine API* to receive data corresponding to the account, as well as perform other operations from the central server. The main difference between the screens lies in the operations that can be performed on each one.

4.3 Tests performed

To validate the correct functioning of the GRETA application and gather general information, various types of tests were conducted for the National Project, which are described below:

1. In July 2023, driving tests were conducted in the city of Gijón with the objective of collecting driving data, utilizing a preliminary version of the application that solely included a route recording screen. During these tests, driving data were collected under diverse conditions, and the fuel consumption of the vehicles used in the tests was recorded to facilitate comparisons with estimated values.

2. In November 2023 [65], an extensive testing program was implemented with different drivers and vehicles in the cities of Cáceres and Madrid, with the objective of collecting driving data and fuel consumption in cities with varying conditions, distinguishing between drivers following a normal driving style and those adhering to the recommended practices of *eco-driving*, as well as utilizing vehicles with different engine types, to enable comparisons of the results of the same routes in each scenario.

   For these tests, a preliminary version of the application was employed, which included the route recording function, selecting the route from textual identifiers included in each driver's notebook.

3. Between May and July 2024, members of the Project carried out driving tests in various cities in Spain, with the objective of testing the functioning of the GRETA application following its initial deployment and providing feedback to the development team to improve various aspects.

4. At the end of September 2024, a series of driving tests took place in the city of Cáceres with similar objectives to the tests conducted in November 2023, with the distinction that the final version of the GRETA application was utilized, which included all the functionalities and improvements implemented up to

that date. The results obtained from these tests were sufficiently accurate and satisfactory to be utilized in the Project.

Chapter 5

Working examples

Contents

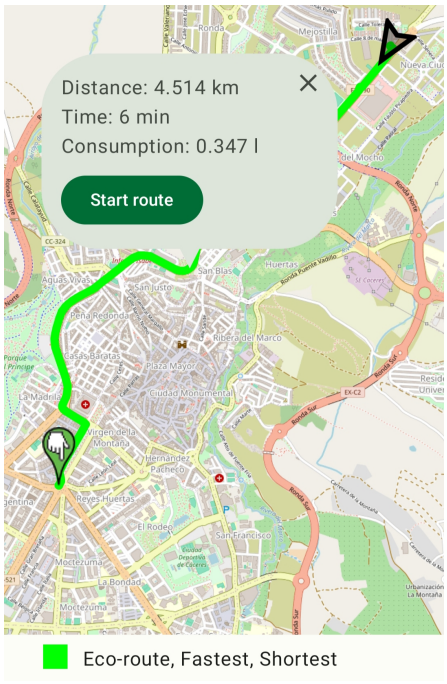
This chapter presents two practical examples of the application's usage, focusing on the route request process and the processing of driving data to visualize the results. The analysis will examine some of the application's key features and compare the final values obtained with those provided by the vehicle and other services provided by the architecture.

The experiments were conducted using a 26-year-old Volkswagen Golf gasoline-powered vehicle, equipped with a display that shows the average fuel consumption per 100 km, and loaded with two passengers and approximately 10 kg of additional luggage, which were introduced into the application as two separate pieces of luggage. Moreover, the recordings were carried out using a mid-range mobile phone, and the accuracy of the calibration data from the phone's sensors may be subject to variability, which could potentially impact the reliability of the results. For the analysis of the recorded data, the Python Folium library [66] was utilized to visualize the routes, while the matplotlib library was employed to generate graphs

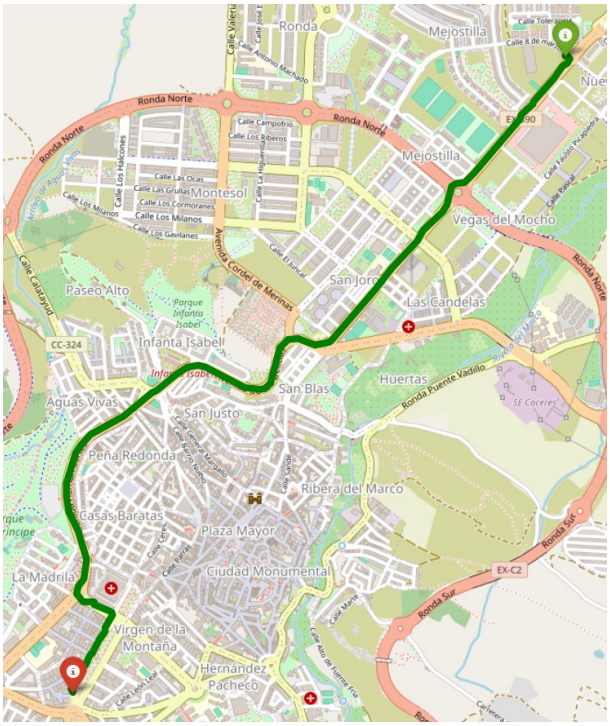
[67], thereby facilitating a comprehensive examination of the resulting information.

5.1 Example 1

This initial example was conducted by requesting a route in Cáceres through the application, specifically from the examination centre of the General Traffic Council (DGT) to the southern end of Avenida de España. The proposed eco-route option was selected and followed, and data were collected from the vehicle's route record, including the actual route taken, which in this case coincided with the proposed route (see Figure 5.1).



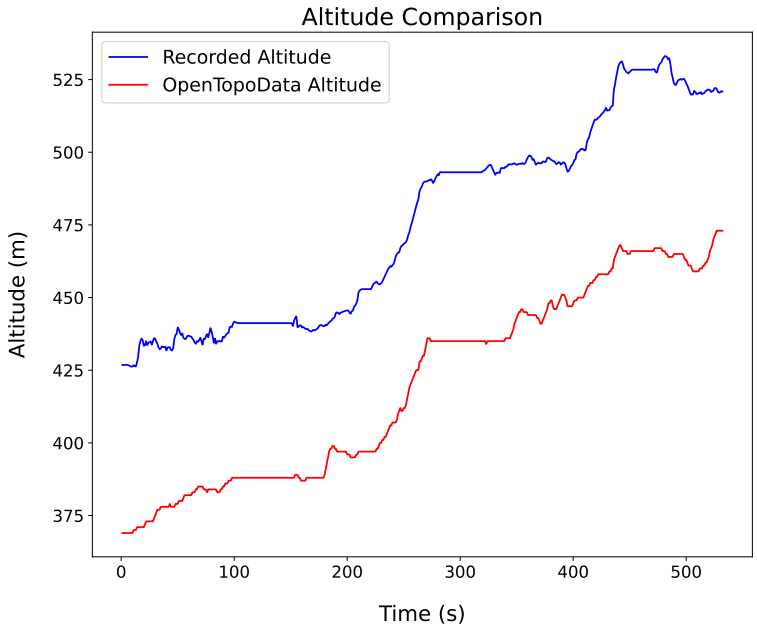
(a) Proposed route for the first example by GRETA App.



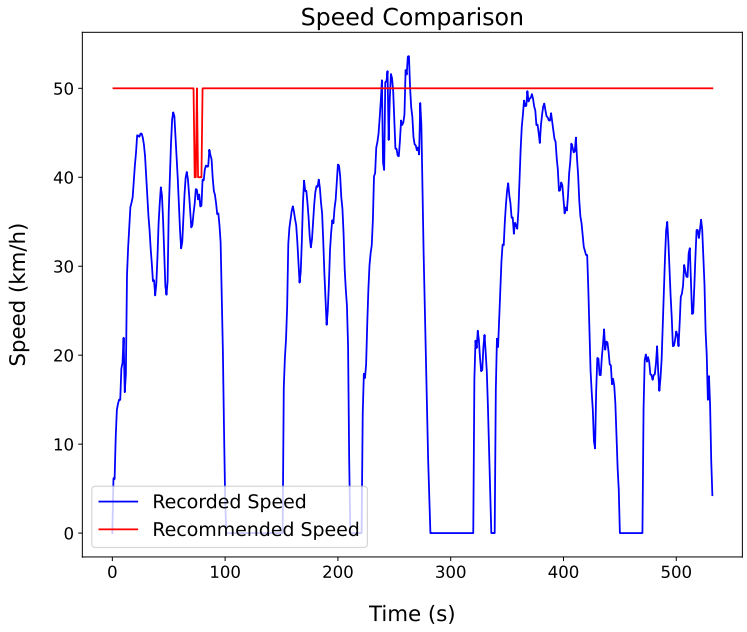
(b) Route taken by the vehicle for the first example.

Figure 5.1: Representations of the route for the first example.

As illustrated in Figure 5.2, the elevation data recorded by the phone, although not identical to those provided by the topography component of the architecture, exhibit a remarkably similar curve when plotted. Furthermore, the recorded speeds display reasonable values given the circumstances of the example, and have been graphed



(a) Graph with the comparison between altitudes of the first example.

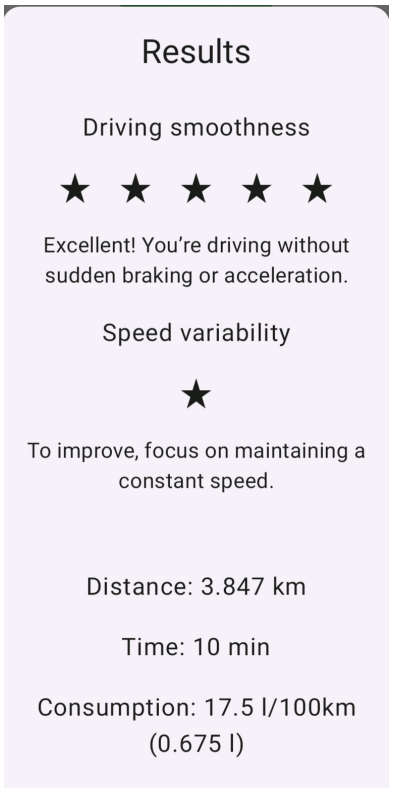


(b) Graph representing the speeds of the first example.

Figure 5.2: Graphs with the main information subtracted from the recording of the first example.

alongside the maximum speed of the road obtained from OpenStreetMap to further validate this finding.

Upon completion of the route, the results were obtained and the estimated fuel consumption by the vehicle was observed for comparison purposes, as shown in Figure 5.3.



(a) Window with the results for the first example.

(b) Car meter with the average consumption of the first example.

Figure 5.3: Results and estimated consumption of the first example.

As can be seen in the scores, the smoothness of driving was positively evaluated, but there is room for improvement in terms of speed variability. One aspect that stands out in comparison to the initially expected result is the difference in time taken and fuel consumption, which may be related to both traffic congestion at the time the route was taken and the flow of traffic taking into account traffic lights, which can limit progress compared to the ideal situation.

On the other hand, it can be observed that there is a difference of 4 litres/100km

between the consumption estimated by the application and that estimated by the vehicle itself. Taking into account the distance recorded by the mobile device, this would result in a difference of 0.163 litres between estimates. To improve the result of the next example, the consumption provided by the vehicle has been introduced to be taken into account when adjusting the calculations.

5.2 Example 2

This route was carried out using the application's free recording function, following the individual driver's criteria to decide on the optimal route, so no prior estimates were obtained before driving.

In Figure 5.4, the route taken can be observed, passing from Catedrático Antonio Silva Street to Cueva de Santa Ana Street. It is worth noting that a detour was made from Ronda del Carmen to Plaza de los Conquistadores via the opposite lane on Antonio Reyes Huertas Street, as the usual route was under construction and a provisional access was opened.

Figure 5.5 shows the results obtained from the route recording, observing that, as in the previous example, the elevations coincide to a large extent, although it can be seen that there is a greater variation in the records of the topographic component. This may be due to significant changes in the layout of the roads travelled with respect to the data used to build the employed dataset, or due to interference or noise with the GPS signal of the phone during recording. On the other hand, in this route, driving was done at a lower speed than in the previous example, as several of the intermediate roads were single-lane for each direction and the speed limit is more strict.

Similarly to the previous example, upon completing the route, the results were sent and a score was obtained, in addition to reviewing the estimated fuel consumption by the vehicle for comparison, as can be seen in Figure 5.6.

Although the scores have not changed compared to the first example and there is still

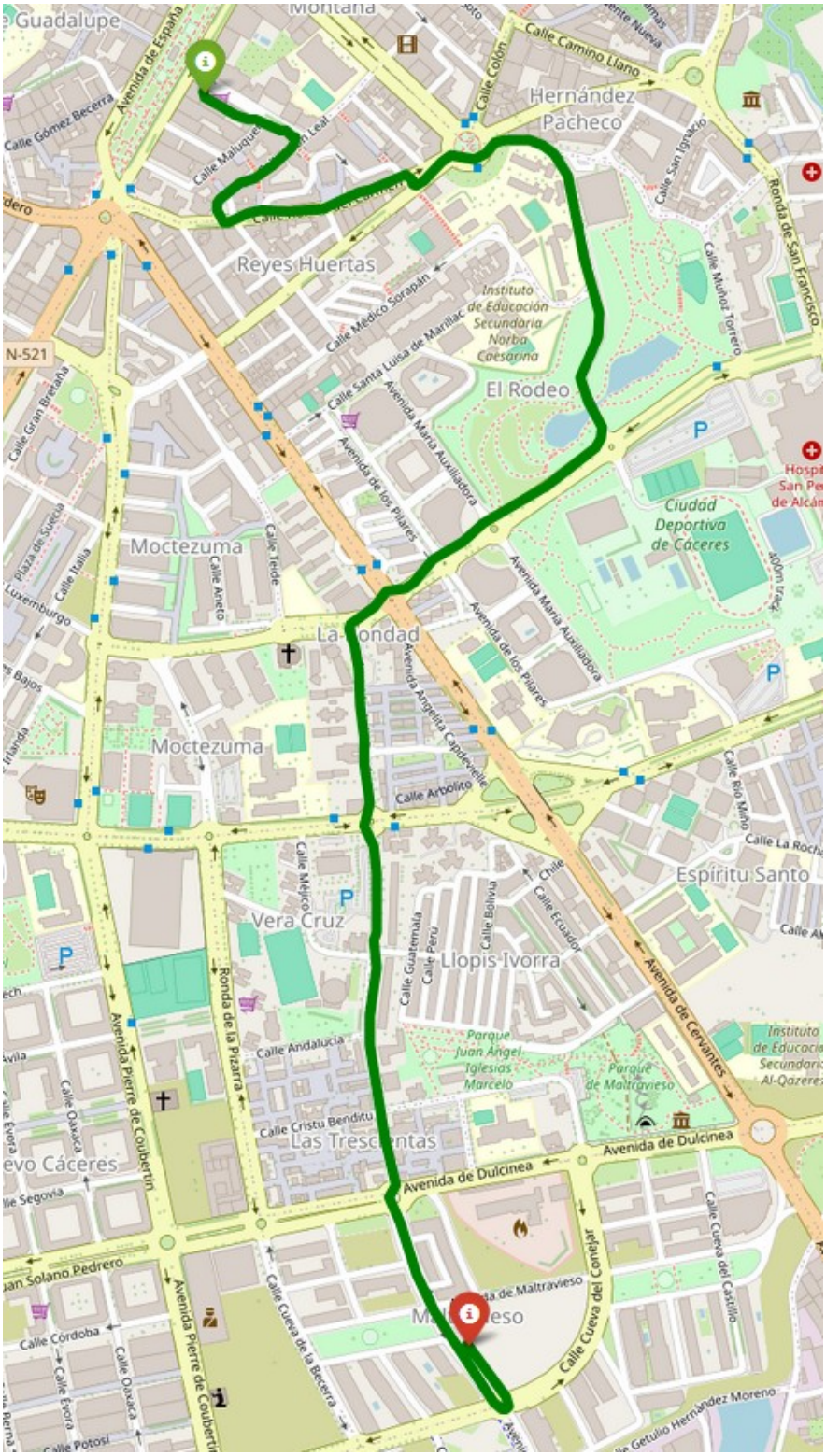
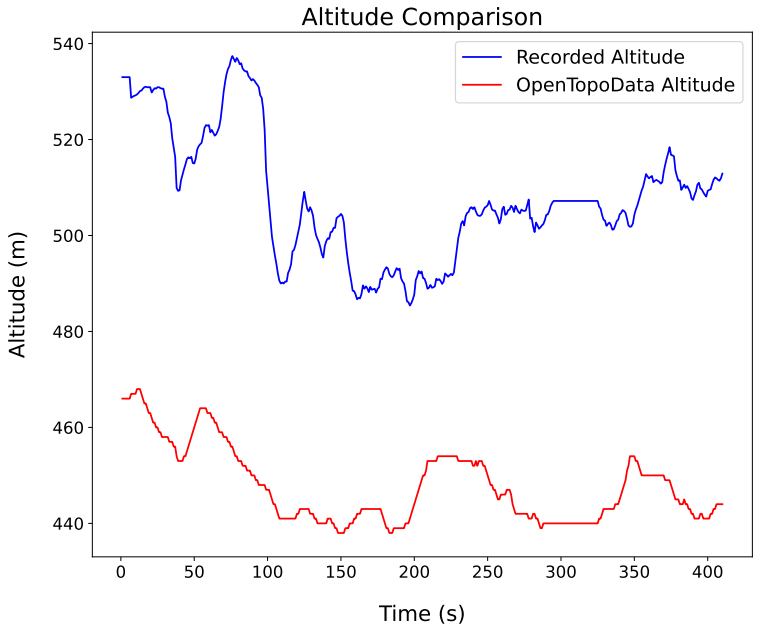
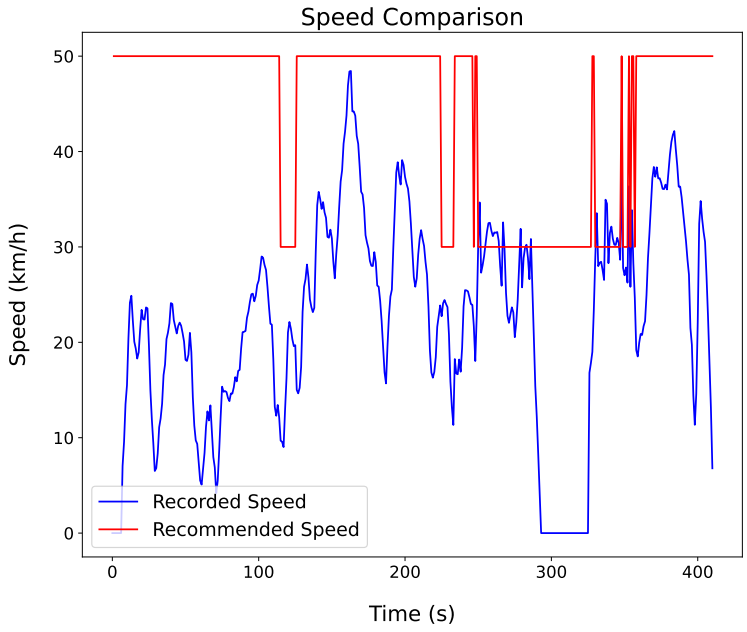


Figure 5.4: Route taken by the vehicle for the second example.

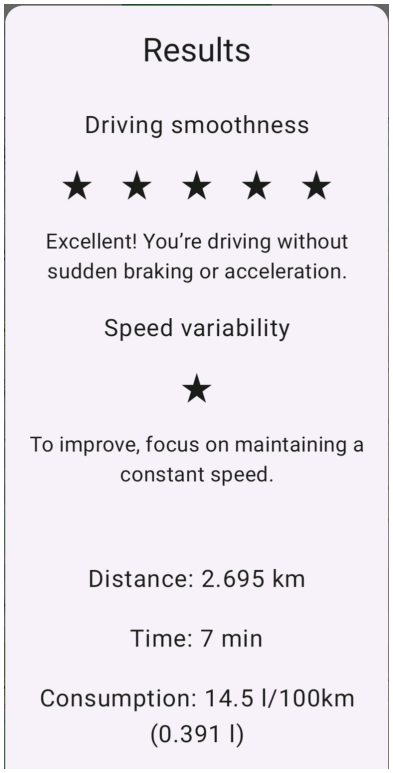


(a) Graph with the comparison between altitudes of the second example.



(b) Graph representing the speeds of the second example.

Figure 5.5: Graphs with the main information subtracted from the recording of the second example.



(a) Window with the results for the second example.

(b) Car meter with the average consumption of the second example.

Figure 5.6: Results and estimated consumption of the second example.

much room for improvement with respect to the speed variability metric, it can be observed that the consumption result provided by the application is closer to that of the vehicle, with a difference of only 0.05 litres this time (1.9 l/100km), approximating better based on the estimated consumption by the car that was introduced previously at the end of the route.

With these two examples, it can be observed that the data provided by the phone's sensors are sufficiently precise to be taken into account, and the application, although it has a well-defined functionality and offers realistic results, has a lot of potential to be explored in order to offer a better experience to the target users.

Chapter 6

Related works

Contents

This chapter undertakes a comprehensive review of the most pertinent existing works related to the proposed system, critically examining their key characteristics and inherent limitations, and subsequently drawing comparisons with the proposed system to contextualize its contributions and innovations.

6.1 Commercial systems

Currently, there is a wide variety of commercial applications offering navigation and route planning services, among which three of them are the most notable.

6.1.1 Waze

This application [68] offers GPS navigation and focuses on providing alerts with information collected from the community through a collaborative network, updating its metrics in each area based on the alerts sent by nearby users, whether about adverse events on the road or statistics such as average road speed, with the purpose of improving the user experience along with the indications for future routes.

In addition to the above, the application has a scoring system [69] that rewards users who complete routes, as well as other actions that can be useful to the community, such as updating map data and correcting errors. Daily statistics are also compiled for each profile, showing a ranking of the most active users to encourage participation.

The operation of the routing service and the algorithms employed are private, and only some clarifications have been made for interested parties. Although this application does not directly offer the possibility of displaying eco-routes, it has other functions such as generating routes based on the ECO label type of the vehicle used to avoid restricted areas on the route [70].

6.1.2 TomTom

This navigation system [71] encompasses a range of GPS devices, integrated vehicle accessories, mobile applications, and online services. According to the official API documentation, in addition to the fastest and shortest routes, it is also possible to select the most eco-efficient route among the available options [72, 73]. Furthermore, users can request the fuel consumption based on the vehicle's engine type and fuel efficiency at 100 km, although the internal process followed is unknown and they only mention

that real-time traffic and historical traffic data are taken into account to obtain these routes.

Although few details are provided about the functioning of this service, TomTom offers another utility designed to mitigate the effects of climate change, for which more information has been revealed. This functionality enables users to request routes for electric vehicles that pass through charging stations, minimizing arrival time while charging the vehicle.

In the paper published about this service [74], the problem of calculating the range of an electric vehicle is explained, which depends on both vehicle parameters such as weight, battery, efficiency, and acceleration, as well as map data, including slopes, traffic, and curvature. Additionally, the problem of selecting charging stations along the route to minimize the overall travel time, taking into account the time spent charging and ensuring that the vehicle does not run out of battery, is addressed.

To solve this problem, a graph has been created to precompute data for route requests, using information from frequent routes between charging stations, and employing a variant of Dijkstra's algorithm to optimize travel time and energy consumption in a multi-objective problem. The current situation is taken into account, and the Pareto front of each node is propagated to its neighbour until the destination is reached. Finally, a route is obtained with indications of arrival and charging times, and the battery level upon arrival at the destination, emphasizing that the route can be recalculated dynamically if the situation requires it.

6.1.3 Google Maps

This is one of the most well-known interactive route services [75], allowing users to select routes on a map for different transportation profiles, such as walking or driving, and also modify the places to pass through or avoid. In a standard search, the algorithm employed prioritizes the fastest route, highlighting it in blue and displaying information about traffic conditions in nearby areas to support this decision.

Since 2021, Google Maps has offered an eco-routing service available in most of Europe and North America, enabling users to obtain eco-friendly routes by selecting the vehicle profile used during the trip. According to the official documentation of the application and its API [76, 77], statistical calculations provided by the **National Renewable Energy Laboratory (NREL) of the United States** are employed to estimate fuel consumption and CO_2 emissions for each route, taking into account factors such as average fuel consumption per vehicle in each region, road gradients, traffic patterns, and the type of route, whether urban or interurban.

In their official sustainability report on Google Maps [78], more information is provided on the functioning of this service and their vision for measures to reduce greenhouse gas emissions in transportation. Here, they state that while their algorithm for returning the fastest route already helped reduce consumption by avoiding congested areas, they have been forced to take additional measures. Therefore, an eco-efficient route is now displayed alongside the fastest route, which may not be the optimal one, as options that take the most time may be eliminated.

To estimate fuel consumption based on vehicle type and route, NREL's **FASTSim** [79, 80] and **RouteE** [81, 82] technologies have been used, respectively. FASTSim is a vehicle simulation model that calculates fuel consumption and greenhouse gas emissions of a vehicle based on its design, technology, and driving behaviour, while RouteE is an energy prediction model that estimates a vehicle's energy consumption based on the route, traffic, and weather conditions.

The eco-routing service of Google Maps follows these steps:

1. A simulation is performed in FASTSim based on the vehicle type and region.
2. Once the results of this simulation are obtained, they are used as input in RouteE to obtain the differences in consumption between several alternative routes defined.
3. A Machine Learning algorithm created by Google is used to predict the most

eco-efficient route, integrating traffic speeds with the NREL consumption model to determine consumption on representative routes.

4. Additional parameters, such as the additional time added by eco-efficient routes compared to the consumption saved, are applied to eliminate non-viable options.

Although this service is a significant contribution to the community, it has some limitations, such as the inability to personalize fuel consumption data for vehicles, as it is limited to using generic models by engine type, or the lack of information about the source of traffic data and their quality. Additionally, the filtering performed based on the additional time taken to reach the destination may be a drawback, as it may not display the most beneficial options without considering time.

6.2 Open-source and academic research systems

There are other approaches to navigation systems that are not as focused on a business model, but rather on research and collaboration with the community.

6.2.1 Open-source systems

On the one hand, there are services that rely on the information provided by OpenStreetMap [83], a free and collaborative geographic database with a comparable amount of information to that utilized by other commercial systems. Notable examples of applications that leverage this database include openrouteservice, GraphHopper, and Open Source Routing Machine, which offer online APIs for querying routes and can be seamlessly integrated with interactive interfaces. Among these three services, openrouteservice particularly stands out due to its defined speed profiles and a feature for retrieving terrain topography and associating heights with the route [84]. Other than that, these applications share a similar functionality and, being open-source, enable users to modify and adapt the code to suit their specific needs.

6.2.2 Academic research systems

On the other hand, there are research-based systems that focus on the development of new algorithms and techniques to improve the efficiency of routing services. In many of these research works, they use modified versions of the open-source systems previously mentioned that incorporate new algorithms or metrics to meet previously unaddressed objectives. Some of these works are going to be briefly described below.

Most of these studies follow the line of *eco-routing*, as it is a topic of interest in the current context of environmental conservation. Zeng *et al.* [85] developed an algorithm for navigation systems, creating consumption models based on vehicle dynamics and proposing a heuristic to solve the NP-complete problem, finding routes that do not exceed the maximum time established by the user and that consume the minimum possible amount of fuel. Another notable work is that of Boriboonsomsin *et al.* [86], which presents an architecture for a navigation system consisting of a database of roads with real-time traffic information, a set that groups consumption factors of different vehicles taking into account topography and traffic, a navigation engine with algorithms to calculate optimal paths, and the user interface that receives user inputs and displays the calculated routes on a map. *MobiWise* was presented by Aguiar *et al.* [87] as a decision-support system for eco-routing, which benefits from the use of *IoT* technologies along with real-time traffic data and a digital twin, being designed for use in Porto, Portugal as well as other medium-sized European cities, concluding that a reduction in emissions of 2.1% can be achieved in groups of vehicles among which 5% of them use *eco-routing*.

Some studies are focused on the **Internet of Vehicles** (IoV) scenario, a topic close to *eco-routing* that is based on the use of sensors and other *Internet of Things* (IoT) devices that exchange information with each other, achieving an increase in collective efficiency and safety [88]. Although this technology is an improvement over **Vehicular Ad Hoc Networks** (VANETs), being compatible with each other under certain circumstances, both face the problem that due to their high prices, some

researchers predict that they will not be accessible to the majority of the population in the near future [89]. Jabbarpour *et al.* [90] proposed a Vehicle Traffic Routing System (VTRS) that reduces emissions and noise by employing an ant-based system similar to a *VANET*, and also incorporating intelligent traffic lights. They conducted experiments using various traffic simulation tools and concluded that this approach can improve the average travel time, speed, and consumption index by 25.5, 19.5, and 17%, respectively.

Another similar approach is that of **Connected and Autonomous Vehicles** (CAVs), which are interconnected vehicles that assist during driving with the help of special technologies [91], and could potentially reduce the number of traffic accidents and save costs in public transportation. Alfaseeh *et al.* [92] proposed a multi-objective eco-routing system for autonomous vehicles that analyses its efficiency in several aspects, such as travel time or emissions generated, showing that when only one aspect is considered, it is better to minimize travel time, while the multi-objective strategy produces better results than those that only focus on minimizing one aspect, also capturing traffic characteristics to increase precision.

A study that focuses on the *eco-driving* approach is the one developed by Orfila *et al.* [93], which proposes a mobile application that provides feedback to users based on their driving mode, adding novel features such as reacting to real-time events like curves, uphill slopes, or intersections based on the information provided by various sensors and data sources, evaluating driving while the vehicle is in motion, and analysing the record of actions performed by the user. To achieve this, an interface connected to an OBD2 is offered to obtain more accurate vehicle data, although it is not mandatory and data can be estimated from the phone's sensors if not possible, and map data is also used to react to events related to the route itself.

Finally, there are some studies that use various technologies with the goal of reducing emission levels, with some expanding the criteria used by other *eco-routing* studies. The framework developed by Vamshi and Prasad [94] uses *IoT* technology

along with geographic data to propose routes in urban areas to distribute traffic density in real-time to areas with lower concentration, thereby reducing congestion and air pollution levels. A path-finding algorithm designed by Ghaffari *et al.* [95] takes into account air quality along with traffic congestion, creating a graph from the information provided by Google Maps along with data recorded by GPS devices in the cloud, to solve the optimization problem and find the best route, using a linear programming method.

Although most of these studies are truly innovative and employ approaches that benefit the vision for future studies, many of them are limited by the information they use or the technologies they are based on, potentially ignoring some relevant aspects of the resulting topology and obtaining less optimized results as a consequence. In addition to the above, many of these systems are not currently applicable for use by an average user, nor do they offer a way to install or test a demo with basic functionality.

6.3 Comparison with the proposed system

After reviewing all the works described above, the comparison has been focused on routing and consumption estimation services, as these are the functionalities that bring the most novelty to this proposal. Academic research works have not been included in this comparison because most of them do not offer a link to an actual application that can be tested and thus compared.

Table 6.1 shows the results of the comparison among GRETA, the main commercial similar services, and openrouteservice. Although the latter does not directly provide eco-routing mechanisms, it has been included as the main representative of the open-source routing apps. The aspects in which GRETA stands out are evaluated, along with some others that, although not yet implemented in GRETA, could be interesting to be included in future versions.

As can be seen in the table, the GRETA App stands out in several aspects compared to the other systems, such as the use of personalized cycles and the possibility

of integrating other routing systems to extract information from the routes they provide. The most notable is the personalization of the vehicle, which allows users to enter specific data about their vehicle, such as the mass, maximum engine power, resistance factor and other coefficients, which are essential for obtaining accurate consumption estimates. This feature is not available in any of the other systems, only allowing the user to select the type of engine or the fuel consumption at 100 km, a generalization that does not take into account other factors that may significantly affect consumption.

Table 6.1: Comparison of the main characteristics of the proposed system with other similar systems.

	GRETA App	Google Maps	TomTom	Openrouteservice
Vehicle personalization	✓	✗[1]	✓[2]	✗
Consideration of additional charge (passengers, equipment)	✓	✗	✗	✗
Consideration of topography	✓	✓	✓	✗[3]
Speed limits	✓	✓	✓	✓
Speed profiles/cycles	✓	✓	✓	✓
Estimated consumption calculation	✓	✓	✓	✗
Individualized consumption adjustment	✓	✗	✗	✗
Consideration of real traffic	✗	✓	✓	✗
Recalculate route in real time[4]	✗	✓	✓	✗
User interface	✓	✓	✓	✓[5]

Continued on next page

Table 6.1: Comparison of the main characteristics of the proposed system with other similar systems. (Continued)

	GRETA App	Google Maps	TomTom	Openrouteservice
Currently applicable	✓	✓	✓	✓
Available	Mobile	Mobile, web	GPS, mobile, API	API, Docker, open-source
Alternative routes	✓	✓	✓	✓
Integration with other routing systems	✓	✗	✗	✗

[1] Google Maps only allows to select the type of engine, generalizing the rest of the vehicle features.

[2] TomTom only allows to enter the constant consumption at 100 km of the vehicle, using this value to calculate the result and ignoring other factors like the mass.

[3] Although Openrouteservice counts on road elevation data (actually, they offer these data as a separate service), it is unclear whether it uses this information in route calculations or not, as it does not offer eco-routing capabilities.

[4] Recalculation based on, for example, traffic (accidents, roadworks), weather conditions or the route followed by the user.

[5] The user interface is implemented as a separate component [96].

Source: Own elaboration based on the information available at the time of writing, provided by the official documentation of each service.

Chapter 7

Conclusions and future works

Contents

This chapter outlines various challenges encountered throughout the Project's development, presents the conclusions reached upon its completion, and suggests potential future improvements.

7.1 Problems found

Outlined below are various challenges encountered during the Project, along with the measures implemented to resolve them.

- **Map library integration**: employing Osmdroid as a library to build the interactive map of the application offered the benefit of unrestricted, concurrent access by multiple users since it does not require an API key. However, this necessitated a manual adjustment for compatibility with the development framework, as it was not initially tailored for use within it.

The initial modification involved handling lifecycle events, since the library was originally intended for invocation within the application's main activity calls. The simplest adaptation involved using a listener from Jetpack Compose to manage these map events, ensuring that appropriate calls are made when needed.

One more issue encountered was managing the objects displayed on the map, necessitating additional data structures and processes to generate these depictions based on the data obtained from events, such as responding to route requests. Furthermore, alterations were essential for managing screen events, particularly when the application remains open for extended periods or undergoes a configuration change during operation. In such cases, the application's state is reinstated. While most Android libraries are equipped to tackle these scenarios and restore the pre-interruption state, Osmdroid lacks these built-in mechanisms, requiring manual additions.

- **Route directions during the route recording process**: in the initial stages of the development of the app, a key idea was to provide navigational guidance to drivers along their selected route, similar to platforms such as Google Maps. This involves tracking the driver's position and providing directional cues either visually on-screen or via text-to-speech, minimising driver distraction.

  Originally, some components were found too complex to develop independently; hence, it was decided to utilise Google Maps for route visualisation once recording began, leveraging its pre-installed presence on Android devices. However, this method posed a downside; replicating the app's exact route required acknowledging rest points, where immediate confirmation upon arrival was necessary, potentially distracting and endangering drivers.

  Ultimately, it was decided to manage the entire route within the app, tracking the driver's position along the marked path, allowing Google Maps usage solely at the user's discretion when opting to record without preset route calculations.

- **Limitations of the recording service**: in its initial configuration, the recording

service encountered an issue due to the limitation of Android, which does not allow background tasks to exceed 10 minutes. When this limit is reached, the system terminates the process, which results in it being restarted if it hasn't fulfilled its goal. For the application in question, this led to incorrect assumptions of session durations of less than 10 minutes and data loss.

To address this, two strategies were adopted and combined. Initially, data collected from the ongoing recording were compiled into a file. This step ensured continuity in data accuracy even if the recording was restarted, thus capturing the entire session flawlessly. Subsequently, after researching ways to bypass the background task limitation, the solution was to switch to a background service, which offers different usage privileges and is not constrained by such limits.

- **Discrepancies among consumption values**: upon completing the preliminary development, initial testing of the application revealed discrepancies between the consumption values obtained using specialized calculations and those reported by the vehicles. It was determined that certain mobile devices contributed to data noise, attributed to the low quality of their internal sensors, reducing accuracy.

  To address this, a smoothing technique was incorporated into the calculations to mitigate the impact of outliers. Additionally, the application now includes a feature for inputting the vehicle's fuel consumption after the trip, which allows for the calculation of a correlation ratio, thereby enhancing the accuracy of future results.

7.2 Conclusions

Once the system for the Project has been implemented and fine-tuned, specific conclusions will be drawn from the outcomes and insights gained throughout the process.

The implementation of a new system has been successfully achieved, thereby enabling

registered users to request optimal routes tailored to their individual preferences through an interactive interface that simplifies the search for designated destination points and the visualization of the routes on a map. As a result, users are provided with personalized feedback regarding their driving habits which, in turn, encourages them to adopt more environmentally friendly driving practices and reduce pollution.

Moreover, the system enables users to select their own vehicles and specify additional parameters, utilizing personalized cycles and adjusting consumption based on individualized data, a feature that few related services currently offer, as pointed out in Chapter 6. This functionality further enhances the accuracy of trip consumption calculations, ultimately fulfilling the objectives outlined in Section 1.2.

The outcomes of this research have been disseminated through two articles submitted to the Sistedes Conference in 2024 [97, 98], specifically within the realm of the Conference on Service Science and Engineering (JCIS [99]), which were presented at the conference held in A Coruña. Furthermore, efforts are currently underway to publish two journal articles, pending acceptance, and an abstract has been submitted for potential publication at the 2025 Transportation Engineering Congress (CIT [100]).

The realization of this work has been a valuable learning experience, allowing me to acquire new knowledge and apply previously learned concepts in the context of software development from my university studies. Specifically, I have gained a deeper understanding of architectural organization and the practical application of design patterns in Android application development. Additionally, I have had the opportunity to explore various tools that facilitate the system implementation process, and I have been impressed by their capabilities. Overall, I believe that this field holds significant potential for future growth and innovation, and I am eager to continue contributing to it in the future.

7.3 Future works

This section gathers some of the potential future enhancements that could be implemented on the developed system to refine and improve some of its current features.

- **Integration with iOS**: while the current application is exclusively developed for Android platforms, it would be advantageous to extend implementation to Apple's devices to broaden its audience and attract additional potential users.
- **Support for more languages**: presently, the app is equipped with English by default and Spanish translations. Thus, incorporating additional languages could be beneficial to capture the interest of a broader range of potential users.
- **Improvement of the user interface**: during various evaluations, unfavourable comments have emerged regarding certain elements of the application's graphical interface. Although some adjustments were implemented in response to this feedback, numerous additional elements may require attention.

  Moreover, on the map interface, it could be beneficial to incorporate additional features, like displaying relevant information about selected destinations or enabling address auto-completion in the search bar to expedite their selection. Turn-by-turn directions could also be incorporated into the routes, with the aim of delivering them via audio to ensure the driver remains focused while travelling.

  Therefore, it would be beneficial to reevaluate the design of the application's interface to enhance its usability ahead of the public launch.
- **Optimization of the processes of the app**: although the architecture of the app has been fine-tuned for mobile use, some delays remain in processing requests like route queries or retrieving results. Despite efforts to limit server resource consumption, further optimization is possible to reduce response times and enhance user comfort.

Appendices

Appendix A

Installation Manual

This appendix will explain the procedure for installing the GRETA application. First, it is required to have access to the APK file, obtained after compiling the source code in Android Studio. Once the file has been downloaded within the mobile device that is going to use the application, it must be checked that it is sufficiently updated to be used, being between Android versions 7.0 and 15.0.

When the file is opened, a dialog opens asking the user to approve new sources of installation, as an Android security measure to avoid installing possible malicious programs. To install successfully, open the option in the "Installation Sources" settings and select the application from which the file was downloaded, being Google Drive in the case of Figure A.1.

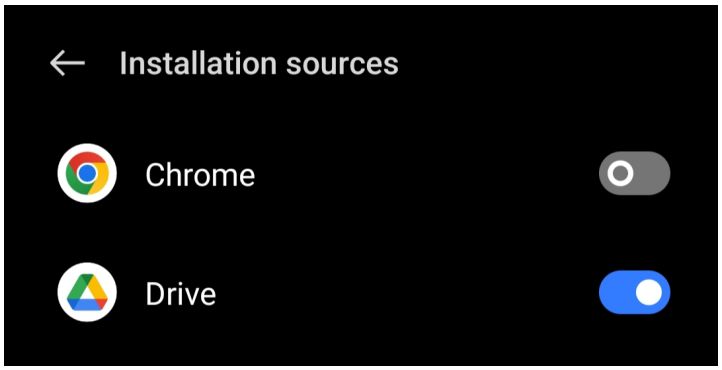


Figure A.1: Dialog for selecting trusted sources to install new apps.

Once the new source is accepted, the user is asked if they want to install the new application (see Figure A.2), selecting “INSTALL” to begin the installation process, and thereby enabling GRETA to be used as any of the other applications on the device from now until it gets uninstalled.

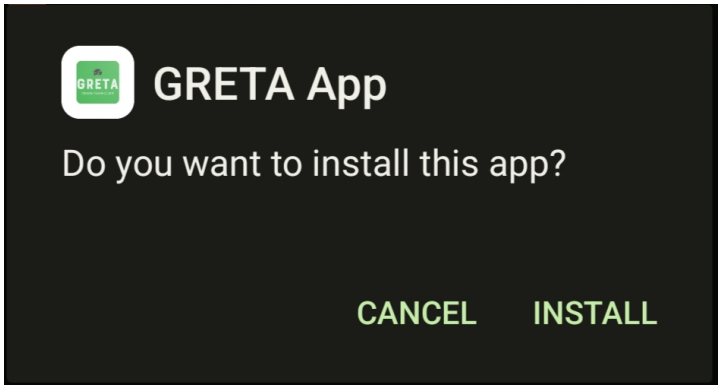


Figure A.2: Dialog for installing GRETA App.

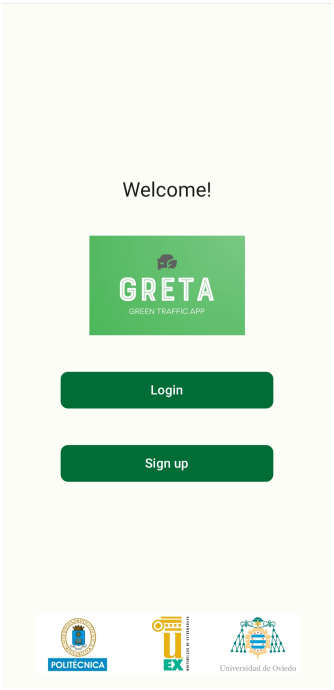
If a new version of the app becomes available, it is advisable to remove the previous version before updating, as certain data might not function correctly on some devices following the update.

Appendix B

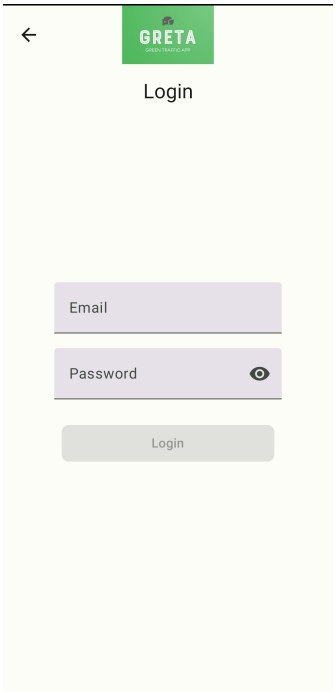
User Manual

Contents

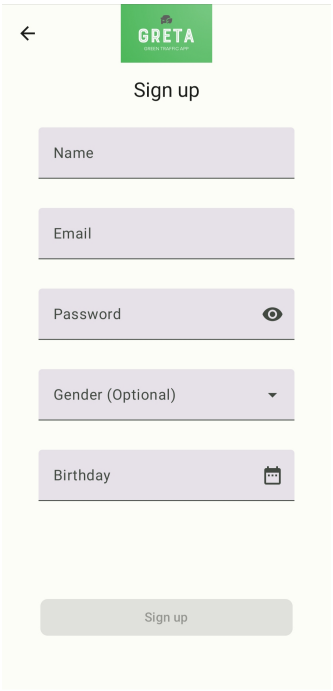
This appendix provides guidance on using the app after its installation on a smartphone. Figure B.1 offers a visual representation of all the primary screens of the app.



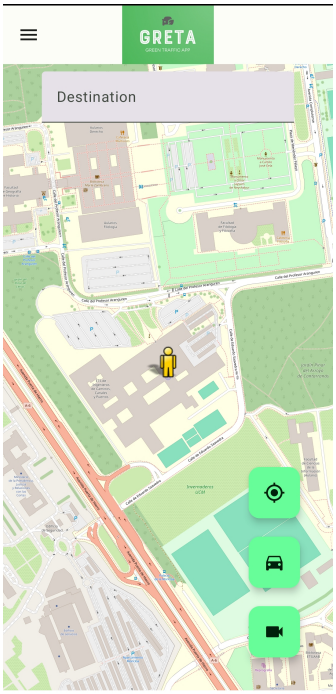
(a) Initial screen.



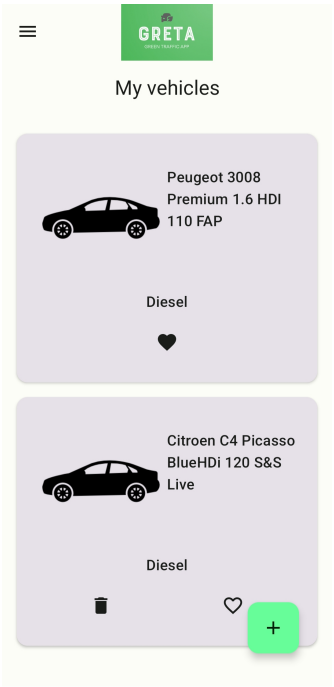
(b) Login screen.



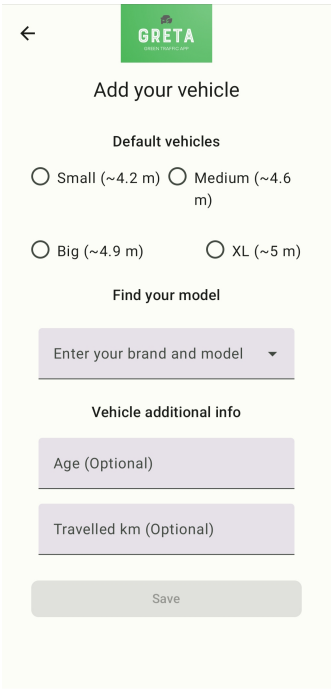
(c) Sign-up screen.



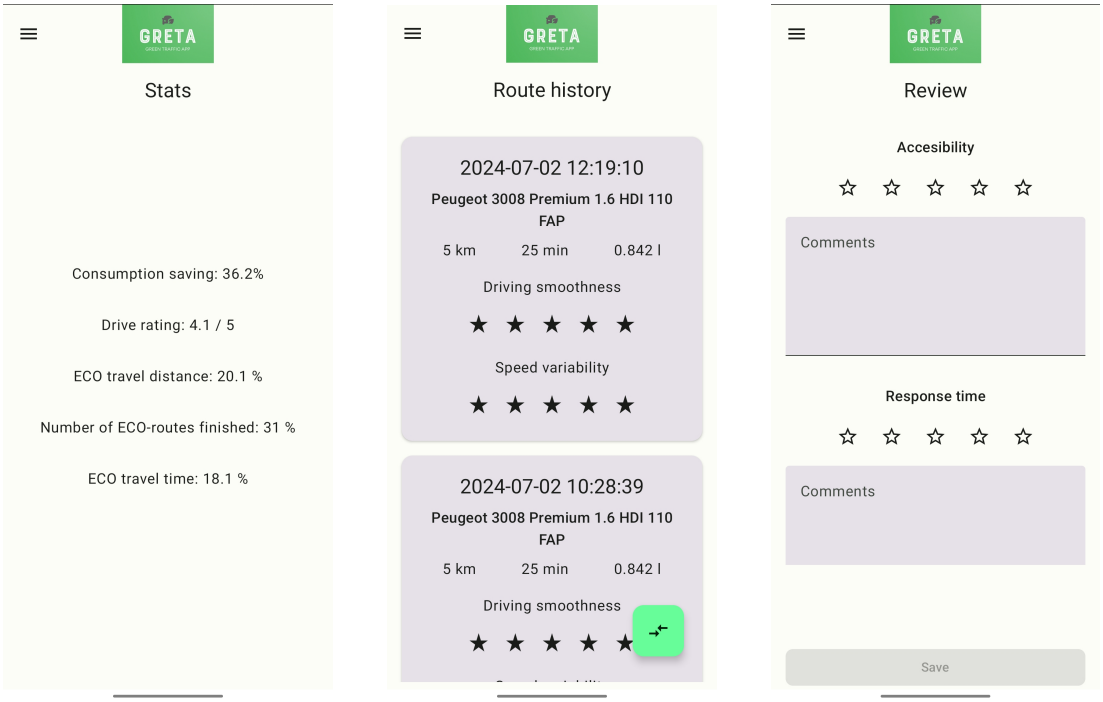
(d) Map screen.



(e) Vehicles screen.



(f) Add vehicle screen.



(g) User stats screen. (h) Route history screen. (i) App review screen.

Figure B.1: Main screens of GRETA App.

B.1 First user access

Upon initially launching the app (see Figure B.1a), the user is asked to access their account by logging in or by registering a new account. If the user opts to log in (see Figure B.1b), they only need to provide their email and password, granting access upon successful entry or showing an error message if unsuccessful. In the event that a new account is created (see Figure B.1c), it is requested to fill in 6 fields (if you do not see any of them, just slide the screen down), being the name, email, password, gender, date of birth, and year of issuance of the driver's license, with the selection of these attributes of each user having statistical purposes. The password field features a button to toggle the visibility of the input text, while the birthday and license year fields provide a dialog to select a date, allowing users to choose a day from the calendar or enter it manually (see Figure B.2).

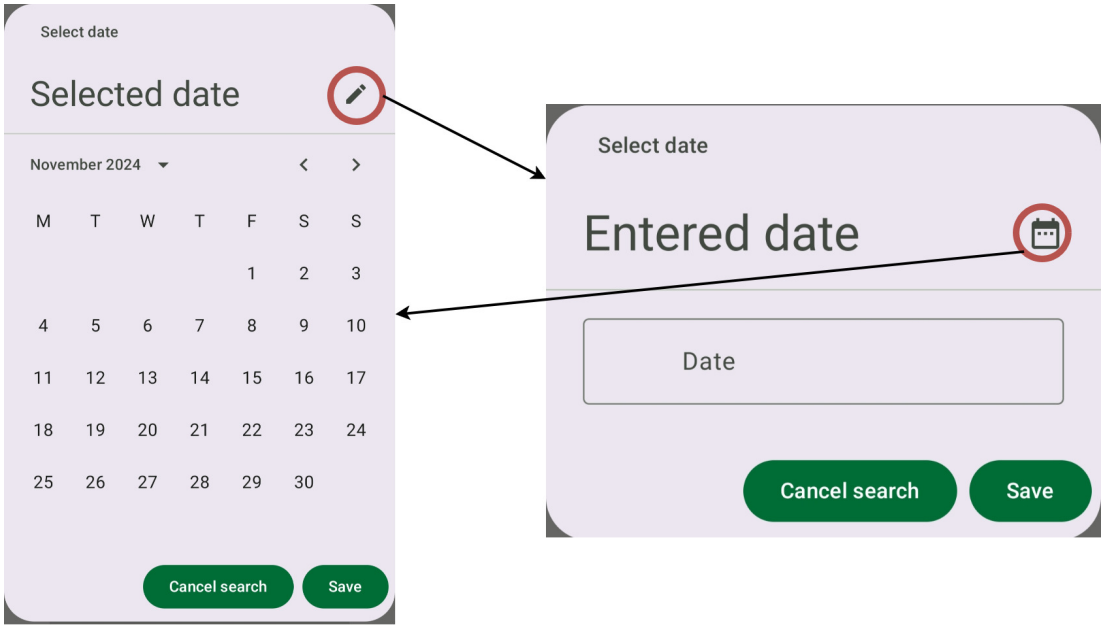


Figure B.2: Date dialog for selecting a date.

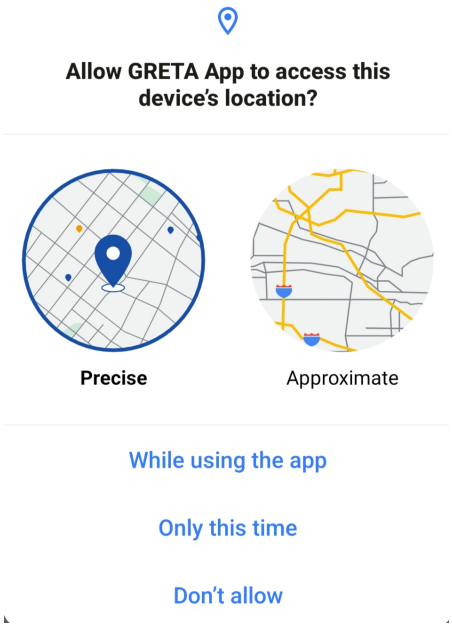
B.1.1 Location permission

When the user logs in for the first time, the app displays the map screen (see Figure B.1d) and asks permission to use the location of the device. To improve the precision of the results, it is suggested to opt for the precise mode (see Figure B.3a). The user will then need to confirm access to location data in the background (see Figure B.3b), allowing route tracking to function even when the app is not active.

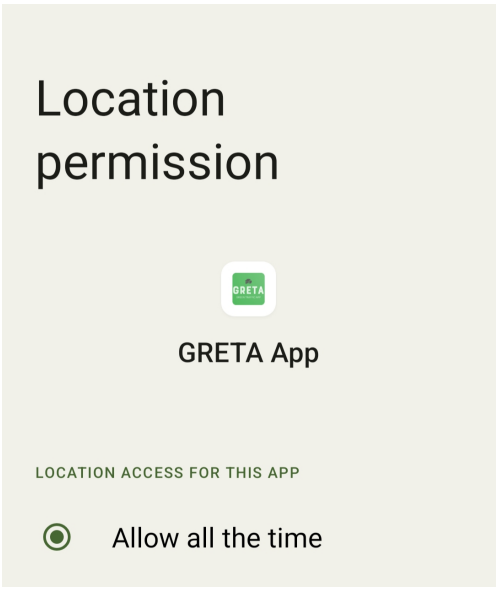
A different dialog will appear showing the route parameters; however, detailed instructions will be given later, as it is presumed that the user currently does not have vehicles. Consequently, the process of incorporating a vehicle will first be outlined.

B.1.2 Vehicle management

After logging in, almost every screen features a button in the top-left corner with three horizontal lines to open the menu (see Figure B.4), which facilitates navigation between the different screens in the app. In addition, there is an option to log out, which brings the app back to the welcome screen (see Figure B.1a). From the menu,



(a) Interface for choosing the accuracy of the recorded position.



(b) Prompt for granting location permissions when the app runs in the background.

Figure B.3: Dialogs requesting app location access permissions.

the user should navigate to the vehicle screen (see Figure B.1e), where they will notice the unavailability of any vehicles for selection, and a message indicates that a new vehicle can be added by clicking the button with a + symbol, which will navigate to the screen for adding a vehicle (see Figure B.1f).

In this interface, the user has the option to add either one of two vehicle types: a vehicle from the four generalised types classified by size to estimate consumption, selectable via the buttons at the top, or a specific vehicle from the database. To do this, enter the brand or model in the central search bar, then click the search icon (see Figure B.5) on the keyboard, and choose the correct option from the drop-down list. After choosing the type of vehicle, the age and distance covered in kilometres can optionally be added, also serving statistical purposes.

Once a vehicle has been added for the current user, it will appear on the Vehicles screen (see Figure B.1e) as a card that displays its details along with some buttons. The heart button allows the user to set the vehicle as their favourite; for this vehicle, the icon will

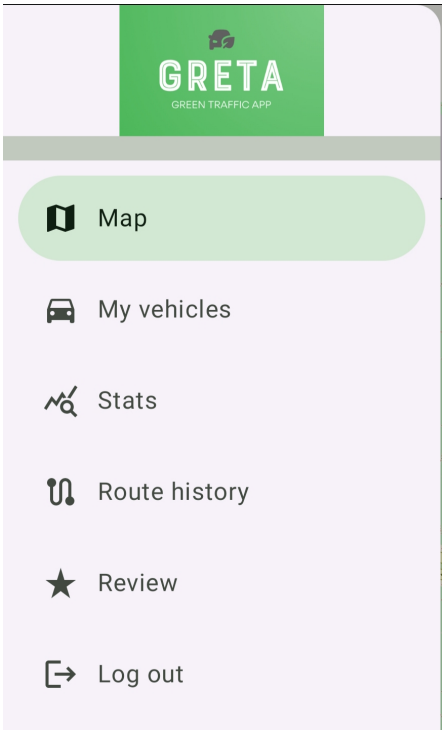


Figure B.4: Side menu of the GRETA application.

Figure B.5: Keyboard featuring a search symbol designed like a magnifying glass.

remain filled while only the outlines will be visible for the other vehicles, indicating that it will be automatically chosen for trips unless modified. Pressing that button again when it is already selected will remove its favourite status. The trash button will delete the vehicle from the list only if it has not been previously used in the app. If it has been

used, the button will not be displayed.

B.2 Routes selection and recording

With a vehicle available, the user is ready to utilize the app. On the Map screen (see Figure B.1d), once opened, a dialog box will appear to choose the route parameters (see Figure B.6), such as the vehicle being used, the number of passengers in the car, and the estimated quantity of bags or items, each with a weight of 5 kg. When a user selects a preferred vehicle on the Vehicles screen, it will be automatically chosen. If no preferred vehicle is set or if the user wants a different one, they can pick another option from the drop-down menu that lists all vehicles associated with them. By default, the number of passengers is set to 1, assuming initially that there is no luggage. There is also an option to share previous results from recorded routes, though this feature is mainly beneficial for those involved in data examination for the project. The dialog will close if the user clicks the "Accept" button or taps anywhere outside the box.

After obtaining location permissions, an icon appears on the map to show the user's position: represented by a person when stationary and an arrow when in motion. This spot acts as the starting point for any routes chosen. Below on the right, there are three extra buttons: the top button (with a reticle icon) re-centres the map to the user's position, the middle button (with a car icon) reopens the route parameters dialog, and the bottom button (with a camera icon) allows for route recording without prior selection; this will be clarified in more detail subsequently.

B.2.1 Destination selection

At the top of the screen, there is a search bar to enter the destination address; it is recommended to include the city at the end to prevent confusion with similar addresses from different countries. Once the address is entered and the search button on the keyboard is pressed (see Figure B.5), various options will be displayed in a drop-down menu (see Figure B.7a). Selecting one will place a marker on the map, as can be seen

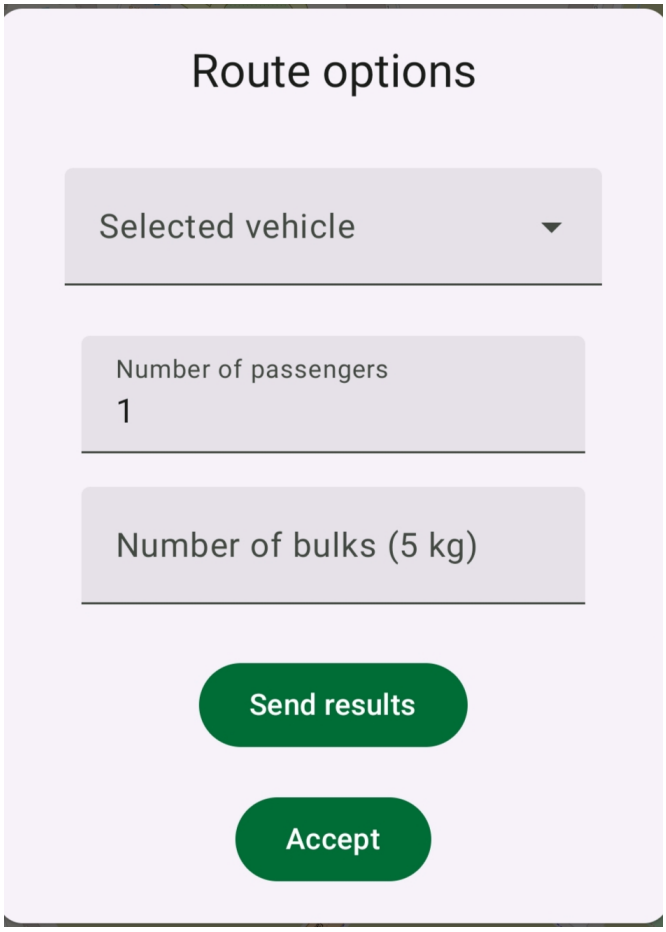
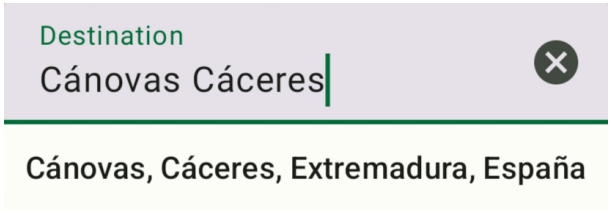


Figure B.6: Box with the parameters that can be configured for a route.

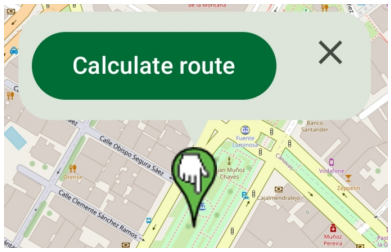
in Figure B.7b, and clicking the cross button on the right will clear the bar's contents and remove the marker. An alternative method for choosing the destination point is by clicking on the map; this action will automatically add a marker and populate the search bar with the corresponding coordinates.

B.2.2 Route calculation and selection

Once the destination marker is set on the map, the user is able to determine the optimal paths between their location and that target. They can achieve this by selecting the "Calculate route" button (see Figure B.7b) that appears after placing the marker or clicking it, which prompts a loading dialog box featuring a spinning wheel and a changing message encouraging the user to choose eco-friendly driving options (see



(a) Search menu for selecting the correct address to place a marker.



(b) Marker that represents the destination, with a button to search routes.

Figure B.7: Interface elements for selecting destination.

Figure B.8).

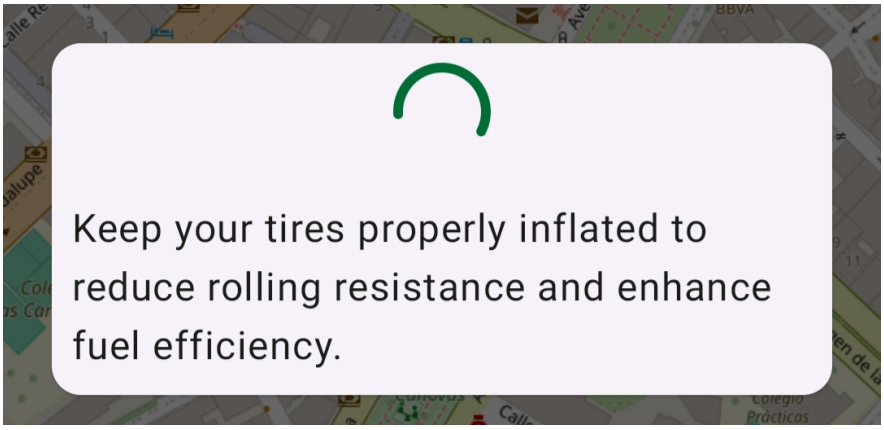
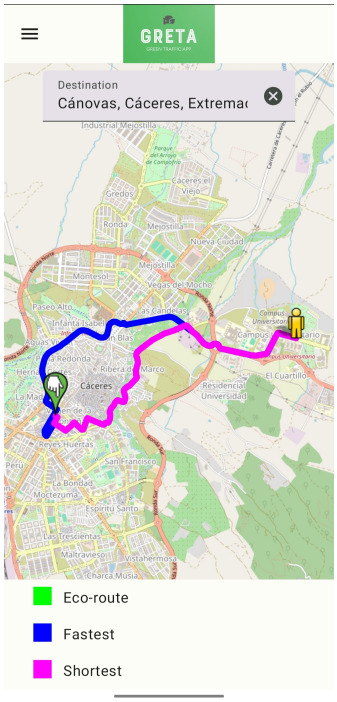


Figure B.8: Dialog box when a petition on the map is loading.

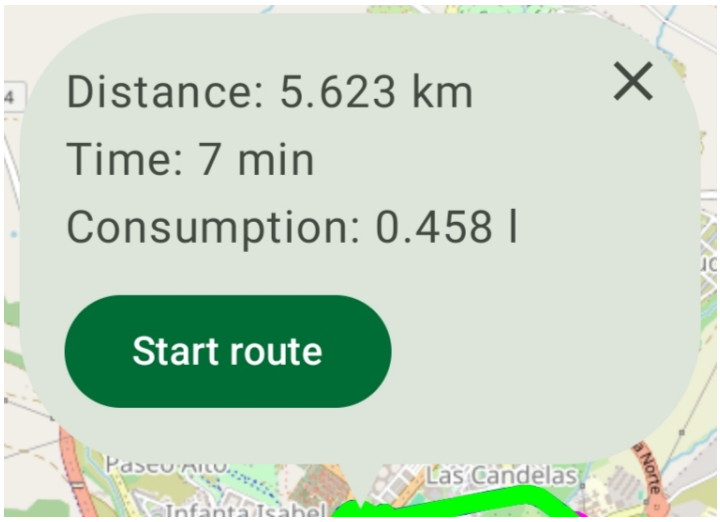
After the loading process is completed, the optimal routes are displayed on the map, accompanied by a legend at the bottom (see Figure B.9a). Clicking on the name of any route will highlight it and open a window displaying its specific details (see Figure B.9b).

B.2.3 Route recording

When the user has chosen their desired route, they can view its details by selecting the name in the legend or clicking on the route itself. By pressing the “Start route” button (see Figure B.9b), the screen will switch to recording mode (see Figure B.10), removing the other routes from the map and altering the arrangement of certain



(a) Screen with the optimal route choices to select.



(b) Details of a route, with a button to start recording it.

Figure B.9: Interface elements for selecting a route to record.

elements.

The screen now consistently tracks the user's position, permitting only adjustments to the zoom level, while the camera is continually angled upwards as they move. The app's menu remains inaccessible unless recording is paused, and the menu button is replaced by an arrow icon to return to the route selection interface.

In this mode, a background process continuously records all user movements, and the phone provides a notification to indicate this activity (see Figure B.11), with a button to stop it. Pressing the cancel button on the search bar will also cancel the recording process and remove the destination marker, and selecting the route will reopen its details, this time with the "Calculate route" button substituted with an option to cancel it as well.

The buttons in the bottom right corner have also been updated. The top-most button, identified by a flag icon, completes the route and submits the recording data for

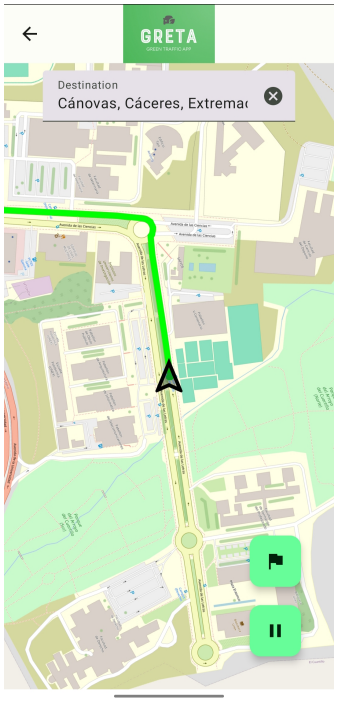


Figure B.10: Map screen when a recording is taking place.

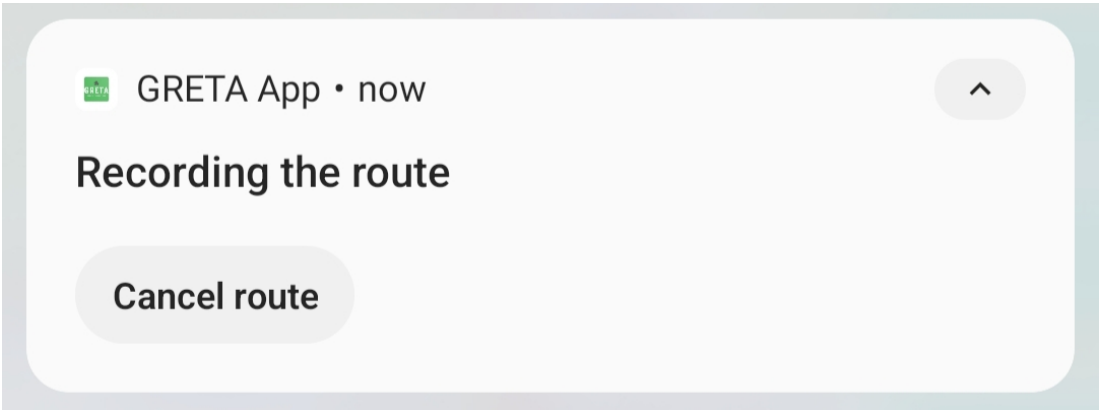


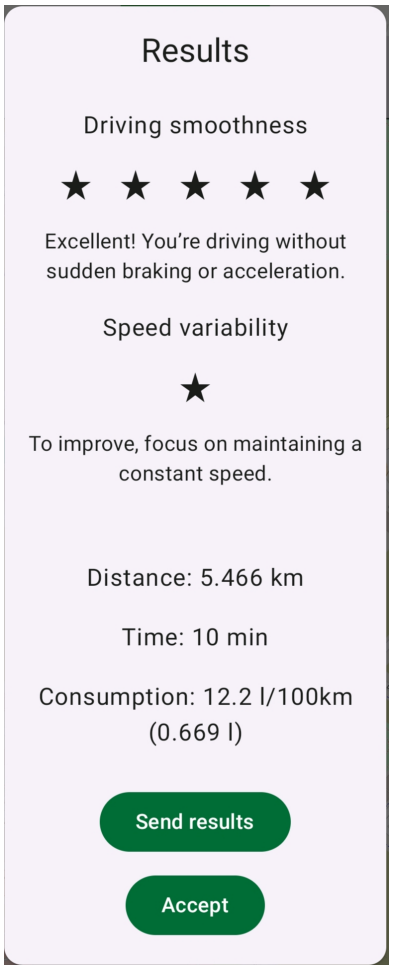
Figure B.11: Notification of the phone to indicate GRETA App is recording a route.

scoring. Meanwhile, the second button provides the option to pause and resume the recording, useful for taking rest breaks.

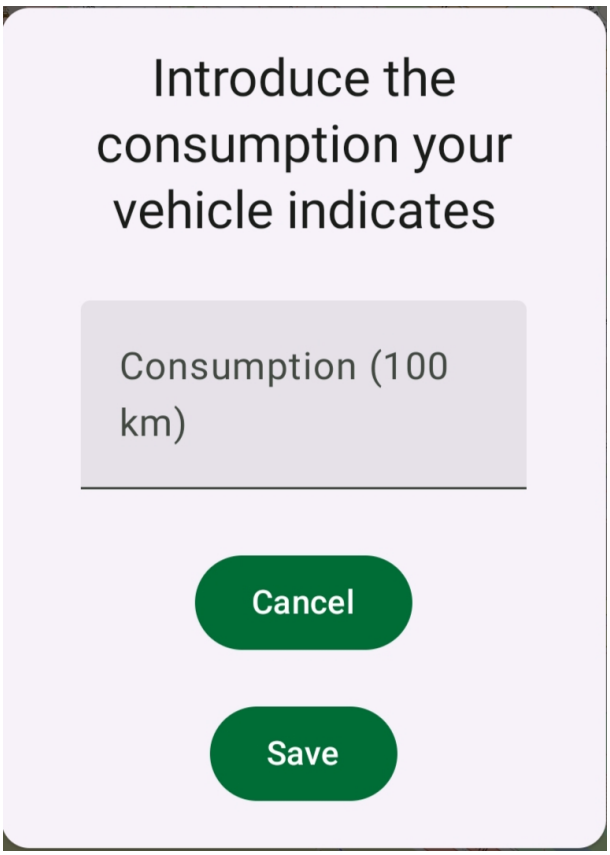
After completing the route, the user needs to click the flag icon to send the results and receive a score. Once the loading screen has finished, a dialog box will appear showing the score, as depicted in Figure B.12a. The evaluation will focus on two criteria: the smoothness of driving and the variability of speed. Each criterion will be rated from 1

to 5 stars, accompanied by a concise summary of its performance and, if necessary, an improvement suggestion. The details below the score outline the duration and distance of the trip, along with the anticipated consumption of the journey. At the bottom, there are two buttons available: one to send the recording files, which is not particularly relevant for ordinary users, and another to confirm and store the results in the profile.

In the absence of data on the correlation between the consumption computed by the app and that reported by the vehicle, an additional window will appear to request the consumption data directly from the vehicle (see Figure B.12b). This assists in refining future estimations until satisfactory calibration is achieved.



(a) Dialog box with the results from the route.



(b) Window for introducing the consumption registered by the vehicle, to calibrate values provided by the app.

Figure B.12: Interface elements when a route is completed.

Completing the route returns the interface to its original state. As mentioned at the start of the explanation, the bottom right icon permits starting a recording without choosing a route, automatically enabling recording mode. In this case, only the recorded trip's values are estimated.

B.3 Other screens

With the primary functionality of the app now clarified, the next step is to discuss the additional screens that display related information.

B.3.1 User stats

The stats screen (see Figure B.1g) presents various details about the user, including:

- The savings in consumption achieved by choosing the eco-route and optimal driving conditions, as opposed to the worst-case scenario.
- The mean score computed from all the journeys recorded.
- The proportion of the total distance driven that was covered via eco-routes.
- The count of eco-routes selected among all the logged trips.
- The proportion of the overall time spent using the app that was dedicated to eco-routes.

This information is refreshed after the completion of each route, allowing users to enhance their profiles and set new goals.

B.3.2 Route history

Within the Route history screen (see Figure B.1h), there is a scrolling list featuring cards that detail the most recently recorded routes, including information such as the date, the vehicle driven, the duration, travel time and consumption, and the star rating

received. In the lower-right corner, there is a button that allows the user to change the list's order, from the oldest entries to the latest.

B.3.3 App review

The App review screen (see Figure B.1i) serves to send feedback to the development team for app enhancement. The key elements for evaluation include accessibility, response time, configurability, usability, trustworthiness, robustness, and utility. Each aspect offers a scoring field ranging from 1 to 5, accompanied by a comment box. In addition, there is a section to provide an overall rating of the app. Once scores have been assigned to all fields, the review is ready to be submitted.

Bibliography

[1] Matt Forrest. *A Brief History of Web Maps*. https://forrest.nyc/a-brief-history-of-web-maps/, July 2021. Accessed: 2025-01-28.

[2] OpenStreetMap. *OpenStreetMap*. https://www.openstreetmap.org/about. Accessed: 2025-01-29.

[3] European Parliament. *CO2 Emissions from Cars: Facts and Figures (Infographics)*. https://www.europarl.europa.eu/topics/en/article/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics, December 2024. Accessed: 2025-01-09.

[4] European Parliament. *EU Ban on Sale of New Petrol and Diesel Cars from 2035 Explained*. https://www.europarl.europa.eu/topics/en/article/20211019STO44572/eu-ban-on-sale-of-new-petrol-and-diesel-cars-from-2035-explained, June 2023. Accessed: 2025-01-09.

[5] European Environment Agency. *Decarbonising Road Transport: The Role of Vehicles, Fuels and Transport Demand*. EEA Report. Publications Office of the European Union, 2022. doi:10.2800/68902.

[6] The International Council on Clean Transportation (ICCT). *Update on Government Targets for Phasing Out New Sales of Internal Combustion Engine Passenger Cars*. http://theicct.org/sites/default/files/publications/update-govt-targets-ice-phaseouts-jun2021_0.pdf, June 2021. Accessed: 2025-01-14.

[7] Jefatura del Estado, Gobierno de España. Ley 7/2021, de 20 de mayo, de cambio climático y transición energética. *Agencia Estatal Boletín Oficial del Estado (BOE)*, 121:62009–62052, 2021. URL: https://www.boe.es/buscar/pdf/2021/BOE-A-2021-8447-consolidado.pdf.

[8] EIT Urban Mobility. Urban Vehicle Access Regulations: From Design to Implementation. *Urban Mobility Next*, 6, October 2022. URL: https://www.eiturbanmobility.eu/wp-content/uploads/2022/10/EIT-UrbanMobilityNext6_HD2.pdf.

[9] Ministerio de la Presidencia, Relaciones con las Cortes y Memoria Democrática, Gobierno de España. Real Decreto 1052/2022, de 27 de diciembre, por el que se regulan las zonas de bajas emisiones. *Agencia Estatal Boletín Oficial del Estado (BOE)*, 311:185962–185981, 2022. URL: https://www.boe.es/buscar/pdf/2022/BOE-A-2022-22689-consolidado.pdf.

[10] European Environment Agency. *New Registrations of Electric Vehicles in Europe*. https://www.eea.europa.eu/en/analysis/indicators/new-registrations-of-electric-vehicles, October 2024. Accessed: 2025-01-09.

[11] Firstpost. *EVs are the Future: A List of All Carmakers Who Have Decided to Phase Out ICE Vehicles and Go Electric*. https://www.firstpost.com/tech/news-analysis/evs-are-the-future-a-list-of-all-carmakers-who-have-decided-to-phase-out-ice-vehicles-and-go-electric-9744401.html, 2021. Accessed: 2025-01-14.

[12] Eva Ericsson, Hanna Larsson, and Karin Brundell-Freij. Optimizing route choice for lowest fuel consumption – Potential effects of a new driver support tool. *Transportation Research Part C: Emerging Technologies*, 14(6):369–383, 2006. URL: https://www.sciencedirect.com/science/article/pii/S0968090X06000799, doi:10.1016/j.trc.2006.10.001.

[13] Toshiaki Kono, Takumi Fushiki, Katsuya Asada, and Kenjiro Nakano. Fuel consumption analysis and prediction model for “eco” route search. In *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual MeetingITS AmericaERTICOITS JapanTransCore*, 2008. URL: https://www.itskrs.its.dot.gov/2014-b00900.

[14] Kyoungho Ahn and Hesham A. Rakha. Network-wide impacts of eco-routing strategies: A large-scale case study. *Transportation Research Part D: Transport and Environment*, 25:119–130, 2013. URL: https://www.sciencedirect.com/science/article/pii/S1361920913001259, doi:10.1016/j.trd.2013.09.006.

[15] Fiamma Perez-Prada, Andres Monzon, and Cristina Valdes. Managing Traffic Flows for Cleaner Cities: The Role of Green Navigation Systems. *Energies*, 10(6):791, 2017. URL: https://www.mdpi.com/1996-1073/10/6/791, doi:10.3390/en10060791.

[16] ecodrive. *What is Ecodriving?* https://www.ecodrive.org/en/what_is_ecodriving/. Accessed: 2025-01-10.

[17] ecodrive. *The golden rules of ecodriving*. https://www.ecodrive.org/en/what_is_ecodriving/the_golden_rules_of_ecodriving/. Accessed: 2025-01-10.

[18] Cindie Andrieu and Guillaume Saint Pierre. Evaluation of ecodriving performances and teaching method: comparing training and simple advice. *European Journal of Transport and Infrastructure Research*, 14(3), June 2014. URL: https://journals.open.tudelft.nl/ejtir/article/view/3030, doi:10.18757/ejtir.2014.14.3.3030.

[19] Ian Jeffreys, Genevieve Graves, and Michael Roth. Evaluation of eco-driving training for vehicle fuel use and emission reduction: A case study in Australia. *Transportation Research Part D: Transport and Environment*, 60:85–91, 2018.

Special Issue on Traffic Modeling for Low-Emission Transport. URL: https://www.sciencedirect.com/science/article/pii/S1361920915002278, doi:10.1016/j.trd.2015.12.017.

[20] Juan Francisco Coloma, Marta Garcia, Yang Wang, and Andres Monzon. Green Eco-Driving Effects in Non-Congested Cities. *Sustainability*, 10(1):28, 2018. URL: https://www.mdpi.com/2071-1050/10/1/28, doi:10.3390/su10010028.

[21] Alvaro Garcia-Castro, Andres Monzon, Cristina Valdes, and Manuel Romana. Modeling different penetration rates of eco-driving in urban areas: Impacts on traffic flow and emissions. *International Journal of Sustainable Transportation*, 11(4):282–294, 2017. doi:10.1080/15568318.2016.1252972.

[22] TRANSyT, Universidad Politécnica de Madrid. *ECOTRAFFIC-APP*. https://transyt.upm.es/en/proyectos/ecotraffic-app/. Accessed: 2025-01-09.

[23] INTSYS. *EAI INTSYS 2023 - 7th EAI International Conference on Intelligent Transport Systems*. https://futuretransport.eai-conferences.org/2023/, 2023. Accessed: 2025-01-16.

[24] José R. Lozano-Pinilla, Iván Sánchez-Cordero, and Cristina Vicente-Chicote. Smart-Routing Web App: A Road Traffic Eco-Routing Tool Proposal for Smart Cities. In Ana Lucia Martins, Joao C. Ferreira, Alexander Kocian, Ulpan Tokkozhina, Berit Irene Helgheim, and Svein Bråthen, editors, *Intelligent Transport Systems*, pages 247–258, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-49379-9_14.

[25] Nominatim. *Nominatim*. https://nominatim.org/. Accessed: 2025-01-03.

[26] openrouteservice. *Services*. https://openrouteservice.org/services/. Accessed: 2025-01-07.

[27] OSRM. *Project OSRM*. https://project-osrm.org/. Accessed: 2025-01-07.

[28] GraphHopper. *Developers - GraphHopper Directions API*. https://www.graphhopper.com/developers/. Accessed: 2025-01-07.

[29] openrouteservice. *Running with Docker*. https://giscience.github.io/openrouteservice/run-instance/running-with-docker, November 2024. Accessed: 2025-01-08.

[30] GIScience Research Group. *docker-compose.yml at main*. https://github.com/GIScience/openrouteservice/blob/main/docker-compose.yml, 2024. Accessed: 2025-01-08.

[31] Project OSRM. *OSRM Backend*. https://github.com/Project-OSRM/osrm-backend, 2024. Accessed: 2025-01-08.

[32] NASA's Earth Observing System. *Shuttle Radar Topography Mission (SRTM)*. https://eospso.nasa.gov/missions/shuttle-radar-topography-mission, October 2019. Accessed: 2025-01-07.

[33] Open Topo Data. *Open Topo Data*. https://www.opentopodata.org/. Accessed: 2025-01-07.

[34] ajnisbet. *opentopodata: Open alternative to the Google Elevation API!* https://github.com/ajnisbet/opentopodata, 2024. Accessed: 2025-01-07.

[35] Mediagis. *Nominatim Docker*. https://github.com/mediagis/nominatim-docker, 2024. Accessed: 2025-01-08.

[36] Docker Hub. *MySQL Official Image*. https://hub.docker.com/_/mysql/. Accessed: 2025-01-07.

[37] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

[38] Flask. *Welcome to Flask*. https://flask.palletsprojects.com/en/stable/. Accessed: 2025-01-08.

[39] NetworkX Developers. *NetworkX*. https://networkx.org/, 2024. Accessed: 2025-01-08.

[40] Iván Sánchez. *Eco-Traffic Web APP*. https://github.com/isanchezdy/eco-traffic-web-app, 2023. Accessed: 2025-01-07.

[41] Android Developers. *Android Architecture Guide*. https://developer.android.com/topic/architecture, December 2023. Accessed: 2025-01-06.

[42] Android Developers. *Dependency injection in Android*. https://developer.android.com/training/dependency-injection/manual, January 2024. Accessed: 2025-01-07.

[43] Android Developers. *Manual dependency injection*. https://developer.android.com/training/dependency-injection/manual, April 2024. Accessed: 2025-01-07.

[44] .cult by Honeypot. *5 Reasons to Use Dependency Injection in Your Code*. https://cult.honeypot.io/reads/5-reasons-to-use-dependency-injection-in-your-code/, July 2022. Accessed: 2025-01-07.

[45] Oracle. *Design Patterns: Data Access Object*. https://www.oracle.com/java/technologies/data-access-object.html. Accessed: 2025-01-29.

[46] Kotlin. *Serialization*. https://kotlinlang.org/docs/serialization.html. Accessed: 2025-01-29.

[47] Android Developers. *State and Jetpack Compose*. https://developer.android.com/develop/ui/compose/state. Accessed: 2025-01-29.

[48] Android Developers. *Where to hoist state*. https://developer.android.com/develop/ui/compose/state-hoisting. Accessed: 2025-01-29.

[49] Google. *Material Design 3*. https://m3.material.io/. Accessed: 2025-01-08.

[50] Android Developers. *Supporting Different Languages and Cultures*. https://developer.android.com/training/basics/supporting-devices/languages. Accessed: 2025-01-23.

[51] Android Developers. *Android Studio*. https://developer.android.com/studio. Accessed: 2025-01-25.

[52] Android Developers. *Layouts in views*. https://developer.android.com/develop/ui/views/layout/declaring-layout. Accessed: 2025-01-25.

[53] Android Developers. *Jetpack Compose UI App Development Toolkit*. https://developer.android.com/compose. Accessed: 2025-01-25.

[54] Material Design. *Material Theme Builder*. https://material-foundation.github.io/material-theme-builder/. Accessed: 2025-01-26.

[55] Yasmine Evjen. *Introducing Material Theme Builder*. https://m3.material.io/blog/material-theme-builder, October 2021. Accessed: 2025-01-26.

[56] Google for Developers. *Maps SDK for Android*. https://developers.google.com/maps/documentation/android-sdk. Accessed: 2025-01-26.

[57] Mapbox. *Maps SDK for Android*. https://docs.mapbox.com/android/maps/guides/. Accessed: 2025-01-26.

[58] Osmdroid. *Osmdroid: OpenStreetMap-Tools for Android*. https://github.com/osmdroid/osmdroid. Accessed: 2025-01-26.

[59] Square. *Retrofit: A type-safe HTTP client for Android and Java*. https://square.github.io/retrofit/. Accessed: 2025-01-26.

[60] Android Developers. *WorkManager*. https://developer.android.com/reference/androidx/work/WorkManager. Accessed: 2025-01-26.

[61] Android Developers. *Save data in a local database using Room*. https://developer.android.com/training/data-storage/room/. Accessed: 2025-01-26.

[62] Android Developers. *DataStore*. https://developer.android.com/topic/libraries/architecture/datastore. Accessed: 2025-01-26.

[63] Iván Sánchez. *GRETA App*. https://github.com/isanchezdy/GRETA_App. Accessed: 2025-02-03.

[64] Ratan K. Ghosh, Vinay R, and Arnab Bhattacharyya. Eco-Routing Using Open Street Maps. *CoRR*, abs/2011.13556, 2020. arXiv:2011.13556.

[65] TRANSyT, Universidad Politécnica de Madrid. *Investigadores de TRANSyT-UPM colaboran en un experimento de conducción verde*. https://transyt.upm.es/investigadores-de-transyt-upm-colaboran-en-un-experimento-de-conduccion-verde/, 2023. Accessed: 2025-01-30.

[66] Folium. *Folium 0.19.4 documentation*. https://python-visualization.github.io/folium. Accessed: 2025-01-21.

[67] The Matplotlib development team. *Matplotlib: Visualization with Python*. https://matplotlib.org/. Accessed: 2025-01-21.

[68] Waze. *Driving Directions, Traffic Reports & Road Alerts by Waze*. https://www.waze.com/company. Accessed: 2025-01-13.

[69] Waze. *Your Rank and Points*. https://www.waze.com/wiki/USA/Your_Rank_and_Points. Accessed: 2025-01-13.

[70] Waze Help. *Navigate with a clean air pass*. https://support.google.com/waze/answer/7670108. Accessed: 2025-01-12.

[71] TomTom. *TomTom - Maps and Location Technology*. https://www.tomtom.com/. Accessed: 2025-01-13.

[72] TomTom. *Calculate Route — Routing API — TomTom Developer Portal*. https://developer.tomtom.com/routing-api/documentation/tomtom-maps/calculate-route. Accessed: 2025-01-13.

[73] TomTom. *Common Routing Parameters — Routing API — TomTom Developer Portal*. https://developer.tomtom.com/routing-api/documentation/tomtom-maps/common-routing-parameters. Accessed: 2025-01-14.

[74] TomTom. *Long-Distance EV Routing - Accelerating the adoption of electric mobility for a cleaner world*. https://www.tomtom.com/products/ev-routing-and-range/#downloads. Accessed: 2025-01-14.

[75] Google. *Google Maps*. https://www.google.com/maps. Accessed: 2025-01-12.

[76] Google Maps Help. *Use eco-friendly routes on your Google Maps app*. https://support.google.com/maps/answer/11470237. Accessed: 2025-01-12.

[77] Google for Developers. *Get an eco-friendly route*. https://developers.google.com/maps/documentation/routes/eco-routes, 2025. Accessed: 2025-01-11.

[78] Google. *Google Maps Eco-Friendly Routing*. https://www.gstatic.com/gumdrop/sustainability/google-maps-eco-friendly-routing.pdf, November 2021. Accessed: 2025-01-10.

[79] National Renewable Energy Laboratory (NREL). *FASTSim: Future Automotive Systems Technology Simulator*. https://www.nrel.gov/transportation/fastsim.html. Accessed: 2025-01-11.

[80] Aaron Brooker, Jeffrey Gonder, Lijuan Wang, Eric Wood, Sean Lopp, and Laurie Ramroth. FASTSim: A Model to Estimate Vehicle Efficiency, Cost and Performance. In *SAE 2015 World Congress & Exhibition*. SAE International, April 2015. URL: https://www.nrel.gov/docs/fy15osti/63623.pdf, doi:10.4271/2015-01-0973.

[81] National Renewable Energy Laboratory (NREL). *RouteE: Route Energy Prediction Modeling Tools*. https://www.nrel.gov/transportation/route-energy-prediction-model.html. Accessed: 2025-01-11.

[82] Jacob Holden, Nicholas Reinicke, and Jeff Cappellucci. RouteE: A Vehicle Energy Consumption Prediction Engine. *SAE International Journal of Advances and Current Practices in Mobility*, 2(5):2760–2767, April 2020. doi:10.4271/2020-01-0939.

[83] OpenStreetMap. *Routing*. https://wiki.openstreetmap.org/wiki/Routing. Accessed: 2025-01-12.

[84] openrouteservice. *Directions Service — ORS API*. https://openrouteservice.org/dev/#/api-docs/directions%20service. Accessed: 2025-01-17.

[85] Weiliang Zeng, Tomio Miwa, and Takayuki Morikawa. Eco-routing problem considering fuel consumption and probabilistic travel time budget. *Transportation Research Part D: Transport and Environment*, 78:102219, 2020. URL: https://www.sciencedirect.com/science/article/pii/S136192091530153X, doi:10.1016/j.trd.2019.102219.

[86] Kanok Boriboonsomsin, Matthew J. Barth, Weihua Zhu, and Alexander Vu. Eco-Routing Navigation System Based on Multisource Historical and Real-Time Traffic Information. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1694–1704, 2012. doi:10.1109/TITS.2012.2204051.

[87] Ana Aguiar, Paulo Fernandes, Andreia P. Guerreiro, Ricardo Tomás, João Agnelo, José Luís Santos, Filipe Araújo, Margarida C. Coelho, Carlos M. Fonseca, Pedro M. d'Orey, Miguel Luís, and Susana Sargento. MobiWise: Eco-routing decision support leveraging the Internet of Things. *Sustainable Cities and Society*, 87:104180, 2022. URL: https://www.sciencedirect.com/science/article/pii/S2210670722004930, doi:10.1016/j.scs.2022.104180.

[88] Medium. *Internet of Vehicles Explained: What Is It And Why Is It Important?* https://medium.com/the-research-nest/internet-of-vehicles-explained-what-is-it-and-why-is-it-important-3d3fd72038ff, 2019. Accessed: 2025-01-19.

[89] Rydzewski Aleksander and Czarnul Paweł. Recent advances in traffic optimisation: systematic literature review of modern models, methods and algorithms. *IET Intelligent Transport Systems*, 14(13):1740–1758, 2020. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-its.2020.0328, doi:10.1049/iet-its.2020.0328.

[90] Mohammad Reza Jabbarpour, Rafidah Md Noor, and Rashid Hafeez Khokhar. Green vehicle traffic routing system using ant-based algorithm. *Journal of Network and Computer Applications*, 58:294–308, 2015. URL: https://www.sciencedirect.com/science/article/pii/S1084804515001885, doi:10.1016/j.jnca.2015.08.003.

[91] Ferrovial. *Connected Autonomous Vehicles*. https://www.ferrovial.com/en/innovation/technologies/connected-autonomous-vehicles/. Accessed: 2025-01-19.

[92] Lama Alfaseeh, Shadi Djavadian, Ran Tu, Bilal Farooq, and Marianne Hatzopoulou. Multi-objective Eco-routing in a Distributed Routing Framework. In *2019 IEEE International Smart Cities Conference (ISC2)*, pages 747–752, 2019. doi:10.1109/ISC246665.2019.9071744.

[93] Olivier Orfila, Guillaume Saint Pierre, and Mickaël Messias. An android based ecodriving assistance system to improve safety and efficiency of internal combustion engine passenger cars. *Transportation Research Part C: Emerging Technologies*, 58:772–782, 2015. Technologies to support green driving. URL: https://www.sciencedirect.com/science/article/pii/S0968090X15001679, doi:10.1016/j.trc.2015.04.026.

[94] B. Vamshi and Raja Vara Prasad. Dynamic route planning framework for minimal air pollution exposure in urban road transportation systems. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 540–545, 2018. doi:10.1109/WF-IoT.2018.8355209.

[95] Elham Ghaffari, Amir Masoud Rahmani, Morteza Saberikamarposhti, and Amir Sahafi. An Optimal Path-Finding Algorithm in Smart Cities by Considering Traffic Congestion and Air Pollution. *IEEE Access*, 10:55126–55135, 2022. doi:10.1109/ACCESS.2022.3174598.

[96] GIScience Research Group. *ORS Map Client*. https://github.com/GIScience/ors-map-client/. Accessed: 2025-01-17.

[97] I. Sánchez-Cordero, J. R. Lozano-Pinilla, D. Álvarez Mantarás, J. Benavente, J. F. Coloma, M. García, P. Luque, A. Monzón de Cáceres, A. M. Rivadeneira Muñoz, and C. Vicente-Chicote. Eco-Traffic APP: una arquitectura orientada a servicios para la identificación de eco-rutas y de tramos adecuados para eco-driving. In M. Resinas, editor, *Actas de las XIX Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2024)*. Sistedes, 2024. URL: https://hdl.handle.net/11705/JCIS/2024/32.

[98] J. R. Lozano-Pinilla, I. Sánchez-Cordero, and C. Vicente-Chicote. Advanced Eco-Routing Planning Tool: herramienta de planificación de rutas en Ciudades Inteligentes. In M. Resinas, editor, *Actas de las XIX Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2024)*. Sistedes, 2024. URL: https://hdl.handle.net/11705/JCIS/2024/26.

[99] SPILab, Universidad de Extremadura. *SISTEDES 2024*. https://sistedes2024.spilab.es/, 2024. Accessed: 2025-01-02.

[100] CIT. *CIT 2025*. https://cit2025.com/CIT2025. Accessed: 2025-01-02.