Francisco José Rodrigues dos Santos

FLEET MANAGEMENT SYSTEM

UNIVERSIDADE D COIMBRA

1 2 9 0

1 2 9 0

UNIVERSIDADE D
COIMBRA

Francisco José Rodrigues dos Santos

Fleet Management System

Internship Report in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Pedro Furtado and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

January 2021

This page is intentionally left blank.

Abstract

The ever growing needs of fleet managers towards the management of their fleets increases the amount of data that needs to be tracked to fulfil said needs. Company mobility plans considering company employees, the sustainability of the fleet, the concerns regarding vehicle emissions and the rapid improvement of electric vehicles as a valid option, all contribute to the amount of data that fleet managers have to keep in mind when managing their company fleet.

With the new needs, come new systems that fulfil the necessities of fleet managers but end up decentralizing the data by the multitude of systems used. Thumbeo Corporate is the fleet management system developed by the hosting company, Ubiwhere, however at the start of this internship, this product does not solve the problem of data decentralization and as such, the main goal of this internship is to adapt Thumbeo Corporate to consume and harmonize data from external systems, as well as implement features that relate the data consumed with a set of Key Performance Indicators.

To develop the system proposed, this works starts by studying already existing bodies of work or products in the market, follows by presenting Thumbeo Corporate as a product, defines a set of requirements and designs the architecture of the system and then ends with a demonstration of the developed system along with closing remarks regarding all of the work produced.

The present document specifies the work produced by the student Francisco José Rodrigues dos Santos, in the context of the Masters of Informatics Engineering of the Department of Informatics Engineering of the Faculty of Sciences and Technology of University of Coimbra of the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

Keywords

Fleet Management, Fleet Management System, Data Harmonization, Telemetry, On-Board Diagnosis.

This page is intentionally left blank.

Resumo

A crescente necessidade apresentada pelos gestores de frota em relação à gestão das suas frotas aumenta diretamente com a quantidade de dados que precisam de ser monitorizados para cumprir essas mesmas necessidades.

Com estas novas necessidades, vem então novos sistemas para as cumprir, mas na verdade acabam por descentralizar os dados pela multiplicidade de sistemas utilizados pelos gestores de frota. Thumbeo Corporate é o sistema de gestão de frotas desenvolvido pela empresa que hospeda este estágio, a Ubiwhere, no entanto à data de inicio do estágio este produto não resolve o problema da descentralização de dados e como tal, o principal objectivo deste mesmo estágio é adaptar o produto Thumbeo Corporate para consumir e harmonizar dados vindos de sistemas externos, bem como implementar funcionalidades para relacionar os dados consumidos com um conjuto de *Key Performance Indicators*.

Para desenvolver o sistema proposto, este trabalho começa por estudar trabalhos relacionados e produtos existentes no mercado, seguindo este estudo com a apresentação do Thumbeo Corporate como um produto, de seguida define o conjunto de requisitos e desenha a arquitetura do sistema e termina como a demonstração do sistema desenvolvido em conjunto com considerações finais sobre todo o trabalho desenvolvido.

O presente documento especifica todo o trabalho produzido pelo aluno Francisco José Rodrigues dos Santos, no contexto do Mestrado em Engenharia Informática do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Palavras-Chave

Gestão de Frotas, Sistema de Gestão de Frotas, Harmonização de Dados, Telemetria, Diagnóstico *On-Board*.

This page is intentionally left blank.

Acknowledgments

This last year is the culmination of my progress since the start of my journey. A journey that was not made alone, and in retrospect it could not have been done alone. Words cannot describe how this journey was impacted by the people that, along with me, are part of it, and for all those people, my gratitude is unending.

To Ubiwhere, for enabling my development as a person and worker, and for arming me with the resources that I needed to complete this work.

To André Duarte, for the immense trust put on me and my work. No words can describe how thankful I am for having you as my advisor and, dare I say, as a friend.

To Professor Pedro Furtado, for the support and feedback given to the project, and for being available whenever any help was needed.

To my friends, for sharing these last five years with me, the good and the bad, the happy and the sad, without you all, my progress would not have been possible.

To my parents, for always being present whenever I needed the most, and for providing me with the opportunity to study, learn and improve myself to the best of my ability.

This page is intentionally left blank.

Acronyms

API	Application Programming Interface. vii, 70
CAN	Controller Area Network. 8
CMS	Content Management System. 48
DTC	Diagnostic Trouble Code. 9, 10, 11
DVIR	Driver Vehicle Inspection Report. 17, 18, 80
ECU	Engine Control Unit. 8
EU	European Union. 80
GDPR	General Data Protection Regulation. 36
KPI	Key Performance Indicator. , i, xi, 1, 5, 12, 13, 14, 52, 66, 73, 74
MIL	Malfunction Indicator Lamp. 11
OBD	On-Board Diagnostics. 1, 8, 9, 10, 11, 12, 14, 19, 34, 36, 64, 66, 71, 84, 99
OBD-II	Second version of the original On-Board Diagnostics unit. 9
ORM	Object-Relational Mapping. 46
RDBMS	Relational Database Management System. 45, 46
SAE	Service of Automotive Engineers. 8
TCM	Total Cost of Mobility. 81, 82
TCO	Total Cost of Ownership. 81, 82
ToS	Threshold of Success. 30

This page is intentionally left blank.

List of Figures

List of Tables

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Chapter 1

Introduction

The amount of data that needs to be tracked by fleet managers is ever growing and fleet management systems have to reflect the new challenges and the new data generated by fleets. The new data arises from recent technologies and strategies that are surfacing nowadays, such as the rising necessity for company mobility plans that have the employees in mind, the concerns of fleet managers to maintain sustainable fleets, the restrictions on vehicle emissions being increasingly imposed by politics and the advent of electric vehicles.

Thumbeo Corporate is a product developed by Ubiwhere, the hosting company for this internship, which offers a fleet management service in the form of a software system. This service, like the competitors studied, contributes to the decentralization of data in the fleet management industry, by creating yet another system, albeit with different features, that fleet managers must use when their features are necessary to comply with their company needs. In this context, the decentralization of data refers to the separation of data, generated by the fleet, across different system that must be consulted separately by fleet managers.

This internship proposes a solution to tackle the decentralization of data in the fleet management industry. At the beginning of this internship, Thumbeo Corporate had no way to integrate and harmonize data from external systems, since it only uses information gathered by the internal system itself. To improve on the current scope of the product this internship will adapt Thumbeo Corporate to consume and harmonize data from external systems, allowing fleet managers to import data from other fleet management systems used into the Thumbeo Corporate software system.

Following the consumption and harmonization of data from external systems, Thumbeo Corporate will be adapted to provide fleet managers with an overview of the users of it's fleet, and the interactions they perform with it, such as any rides taken or reports on any vehicle or ride; the vehicles registered and the data generated by their On-Board Diagnostics (OBD) system; and the relation of data stored by the system with a set of predefined Key Performance Indicators that fleet managers can use to measure availability, utilization and fuel consumption metrics of their fleet.

This report documents the study, design and development that took place during the academic internship. This internship is included in the curricular plan of the Informatics' Engineering Masters Degree in the Software Engineering branch. Ubiwhere is the hosting company for the internship that took place in its office in Instituto Pedro Nunes, in Coimbra.

1.1 Objectives

From a shallow perspective, the primary objective of this thesis is to develop a system that can consume and harmonize data from external system. Nevertheless, the previous perspective lacks in the sense that it is vague, therefore a deeper perspective follows. Concretely, the project has four objectives:

- Present academic research on fleet management topics and a competitors analysis on fleet management applications.
- Propose an architecture that can be used to develop a fleet management application that intends to consume data from external systems.
- Define and implement a Data Harmonization System capable of consuming data from different external systems and transform the data into a defined data structure.
- Define and implement a Fleet Management System that can correlate the data consumed and created with a set of established metrics.

1.2 Document Structure

This document is organized into four chapters, each of them with the following ambitions:

Chapter 2 — State of the Art provides the research methodology, the academic literature available regarding fleet management topics and the answers to questions initially posed about the problem; it also explores other available solutions in the market, providing a competitors analysis and lastly it defines what are the contributions of the internship.

Chapter 3 — The "Thumbeo Corporate" chapter details the core principles of Ubiwhere's solution and demonstrates the research done on the topics applied when building the product, along with the changes that will be made to the already existing system.

Chapter 4 — Project Management explains the methodology used in the whole internship, the planning done for both semester that compose the internship. This chapter also covers the risk management process and builds the success criteria that will be used as metric to evaluate the internship.

Chapter 5 — Requirements Specification presents all the user stories elicited while explaining the elicitation process. This chapter also exposes the business, legal and technical restrictions, and provides the quality attributes mandated at the system that will later impact the architectural decisions.

Chapter 6 — Architecture and Technology consolidates the technology background; describes the system architecture; reviews design decisions and illustrates the technologies in use by the system.

Chapter 7 — Project Development presents the implementation aspects of the different systems developed as well as the quality assurance performed on the systems implemented.

Chapter 8 — Final Product of the Internship shows the implementation of the product specified throughout the document.

Chapter 9 — Conclusions presents the thoughts regarding the entire internship, providing insight into the problems faced, the lessons learned and an overall critical analysis of the internship.

This page is intentionally left blank.

Chapter 2

State of the Art

The first step to develop a fleet management system that works with a data harmonization system is to research state-of-the-art work produced either in an academic environment or in an enterprise setting. This chapter is structured in such a way that the academic literature is tackled first, by posing and answering a set of questions. Next, a competitors analysis of the available solutions created by different companies follows the academic literature review. Lastly, the chapter ends by clearly defining the contributions of this internship.

2.1 Literature Review

The current section serves the purpose of reviewing the results found in the literature read, initially presenting the research methodology followed.

Research Methodology When defining the research methodology, the main goal established was that of answering the following questions: **Q0:** *What is fleet management?* **Q1:** *What data can be gathered about a fleet?* **Q2:** *Which data services have to be integrated to achieve a fleet management system?* **Q3:** *What fleet management Key Performance Indicators are relatable to the data made available by the data services?*

Researching academic literature is the basis from which to answer the proposed questions, and *DBLP [1]*, *Google Scholar [2]* and *Repositórios Científicos de Acesso Aberto de Portugal [3]* were the chosen search engines to find literature that could help to find answers to the previously posed questions. The following sections reference related work that provide insight into possible answers.

2.1.1 Fleet Management

Before delving deeper into the various aspects of fleet management, it is important to define the term itself in order to contextualize the research done, the decisions made, and the references made to "Fleet Management" as a concept.

In [4], Kaixiang Lin (et al) consider the fleet management problem as the *"problem of* *managing a large set of available homogeneous vehicles for online ridesharing platforms”*. It also presents what is the goal of the management aspect as *“maximize the gross merchandise volume (. . .) of the platform (. . .)”*. Regarding both statements made, the definition of fleet management provided lacks precision by not specifying a quantity to tie into the “large set” and by limiting how homogeneous the vehicles themselves must be. The definition given is also being limited to online ridesharing platforms, which in reality is a subset of the environments in which fleet management occurs.

In [5], Warren B. Powell (et al) define the fleet management problem as involving *“managing fleets of equipment to meet customer requests as they evolve over time”*. The equipment is being specified as *“containers which hold freight (. . .), locomotives, truck tractors, taxicabs or business jets”*; The customers are considered to be either (people or freight) and their requests are defined as wanting *“(. . .) to move from one location to the next.”*. This definition of fleet management as a concept is specific enough in identifying what is the fleet equipment and what are the customers and their goals. Nevertheless, the *“managing”* aspect is never accurately defined, nor are the actions taken to manage a fleet specified.

In [6], Falco (et al) points out that *“Fleet Management addresses transportation management and monitoring, including cars, trucks, airplanes, ships, among others (. . .)”*. This definition specifies which vehicles can be included in fleet management, and it also points out one goal of performing fleet management, it being to *“remove or minimize the risks associated with investing in vehicles as well as knowing the behavior of drivers”*. Nonetheless, the “management and monitoring” aspects are not scrutinized enough; what is being monitored? what is being managed and how is it being managed?

In [7], Monnerat (et al) express that *“Fleet management is a broad concept that incorporates decisions about fleet sizing and configuration, fleet allocation, vehicle routing, considering homogeneous or heterogeneous vehicles.”*. This definition identifies Fleet Management as a concept that encompasses a subset of other concepts and problems, being the problem of finding the right number of vehicles for a fleet and their respective jobs, or finding the best routes for each vehicle used, or how each vehicle should be allocated for a specific use case. The consideration for homogeneous and heterogeneous fleets is also important given that different companies might employ distinct types of vehicles in their business model.

2.1.2 Data Gathering — Human Generated Data

The human element of fleet management — the fleet managers, drivers and passengers — inherently produce data that can be used by a fleet management system. In this sense, the data that can be gathered is directly tied to the interactions each actor has with the company fleet; to give some examples: drivers and passengers schedule work trips, fleet managers plan vehicle maintenances and drivers report failures in fleet vehicles. The set of all interactions, when gathered to be displayed to fleet managers, should provide a more holistic view of the company fleet.

In [8], Calderón (et al) present a data mining approach towards the exploitation of a trip-based dataset, from the RideAustin application, and subsequent transformation into a dataset capable of enhancing a ridehailing service. As a body of work, it constitutes an example of the importance of mining human generated data — in this case, the data inherent to trips, the starting and finishing points, average speed and number of passengers, among other pieces of data.

In [9], Vogooshi (et al) mainly propose a new scoring process for a multi-agent transport simulation platform, along with a method to generate a synthetic population in a simulation. Specifically, the scoring process designed takes into account user generated data, such as the *willingness to use* factors of waiting time and in-vehicle travel time and the *travel distance* defined by a human agent. Although the work sets itself up to use human generated data, it is made clear by the conclusions that such data is a key factor when tracking operator costs and system profitability in fleet management system. The Robo-Taxi product analyzed in the work can also be considered a solution to the mobility needs of a company, and as such, if implemented to solve the corresponding problem, the user data referenced in this body of work has to be tracked.

In [10], Szczepański (et al) present an approach to fleet management, while accounting for the mobility needs of company employees and establishing a process through which the company can select different vehicles for different tasks. The mathematical model specified contains a vector that represents the expectations of the user that will use one of the fleet vehicles; the expectations contained in the vector, that represent examples of data generated by users, are the following:

- **q(zd,tp)** — type of implemented **zd** — th task by **tp** — th type of vehicle,
- **t(tp)** — assumed time of completing tasks by **tp** — th type of vehicle,
- **rp(tp)** — assumed yearly mileage **tp** — th type of vehicle being traversed during the implementation of tasks,
- **sf(tp)** — assumed method of financing **tp** — th type of vehicle,
- $\beta_{\mathbf{1}}$ — expected type of body,
- $\beta_{\mathbf{2}}$ — expected permissible gross vehicle weight,
- $\beta_{\mathbf{3}}$ — expected number of doors,
- $\beta_{\mathbf{4}}$ — expected type of fuel,
- $\beta_{\mathbf{5}}$ — expected displacement of the engine,
- $\beta_{\mathbf{6}}$ — expected engine power,
- $\beta_{\mathbf{7}}$ — expected emission standard,
- $\beta_{\mathbf{8}}$ — expected type of gearbox,
- $\beta_{\mathbf{9}}$ — expected average fuel consumption per 100 km,
- $\beta_{\mathbf{j}}$ — other expectations.

2.1.3 Data Gathering — Sensors and Devices

Vehicles, being complex systems have a variety of components from which data is gatherable, and the method through which the data gathering happens can vary differently from the type of data itself. This section presents the types of sensors and devices that can be used to gather data from vehicles, as well as some examples of what data each sensor and device can gather.

Smartphones

Nowadays, smartphones are highly available and are also ubiquitous devices, divided between Android and iOS operating systems. These devices are equipped with Internet connection, through Wi-Fi and 3G/4G technologies. Most smartphones also possess a variety of different components and sensors, such as a camera and microphone, GPS, accelerometers and gyroscopes.

These different sensors enable the smartphone to gather data about its current geographic position, its current speed and acceleration and its orientation. The Internet connection provides the ability to communicate with external services, sending the data previously gathered by the sensors, to a cloud system, or to a peripheral that will parse the data.

In [11], E. Türk (et al) develop an Android framework that is able to communicate with On-Board Diagnostics (OBD) system, described later in this section. This work demonstrates how a smartphone can also be integrated with other data gathering tools to improve the methods through which it is possible to gather data from vehicles. Furthermore, the work itself reviews and researches the communication protocol used by the OBD system and how and interface can be constructed to enable communications with Android smartphones. According to the work and it's references, the OBD system uses a Controller Area Network (CAN) protocol to communicate with vehicle's Engine Control Unit (ECU); the CAN itself acts only as means to provide a communication channel, without specifying a communication standard itself. Nevertheless, the Service of Automotive Engineers (SAE) developed a standard named J1939 that is used in most commercial vehicles.

Although the smartphone provides access to some of the data that could be useful for this internship, it proves to be limited in comparison to the two following options when it comes to the variety of data it can extract from vehicles. While the smartphone provides the geographical location as well as speed and acceleration, the following devices can provide not only the same information but a lot more. Given this limitation on the data that can be extracted with smartphones, another option will be chosen to gather data from vehicles.

Independent Sensors

The installation of independent sensors on important components is also a possibility for detecting maintenance needs. These independent sensors can be accelerometers, GPS, a tachograph, an oxygen sensor reader, among other sensors. Although the use of independent sensors is theoretically possible, the research done was unsuccessful in finding a body of work that would prefer to experiment with this approach instead of using either the aforementioned smartphone or the following device explored, the On-Board Diagnostics.

In theory it is possible to gather data from every component in a vehicle using independent sensors installed unto the components themselves, but in reality the usage of these independent sensors for every component imposes added costs in terms of buying, installing and maintaining the needed sensors. There is also another problem pertaining to the installation of independent sensors, there would be the need to relay all of the information of the sensors into a cloud service and even before that, the data from the sensors would have to be aggregated into a local system that could transmit the data into the cloud. All of these problems make it so that independent sensors are discarded as an option for this body of work.

On-Board Diagnostics

On-Board Diagnostics (OBD) is described as a standard for a vehicle's ability to diagnose itself and report failures in the system by the usage of Diagnostic Trouble Codes (DTCs). Said DTC can then be read by an OBD scanner and interpreted by anyone with the knowledge to do so; this includes a software system.

The current iteration of this mechanism, the OBD-II, was made mandatory for all cars produced for the United States in 1996 [12]. Meanwhile, the European Union made the OBD-II mandatory for petrol vehicles in 2001 and for diesel vehicles in 2004 [13]. Take in mind the regulations only applied to vehicles that are planned on being sold in said territories; meaning that European cars built to the US from 1996 onwards, were mandated to possess an OBD unit. This proves that if the implemented system uses the OBD-II standard, it will be capable of analyzing every vehicle that is suitable to be part of a company fleet.

OBD-II Devices The OBD-II system can be accessed with the help of some different types of devices. **Handheld scanners** are the most commonly used by automotive technicians. They offer a simple terminal to view the OBD system information available. **Data Loggers** are designed to spent most of their time connected to the vehicle collecting data for later analysis. ***Dongles*** are used to enhance the OBD system usually with features like Wi-Fi, Bluetooth or 4G modules, to communicate with the system over larger distances.

The aforementioned OBD devices have different methods of use from one another, nevertheless one aspect they all have in common is how they connect to the vehicle, which is through the OBD port present in vehicles. This port is located differently from vehicle to vehicle, in most it is located under the steering wheel, in some it is inside the glove compartment, in others it is located in the compartments below the radio, and there still many other locations for this port. Regarding their different methods of use, **Handheld scanners** have built-in screens that enable the user to read the information from the OBD; the **Data Loggers** record data from the vehicle and are able to write it to an internal storage that can be accessed later by connecting the logger to a computer or by wirelessly connecting to the logger itself; finally the ***Dongles*** are able to read data generated by the OBD in any given instance but in general must make use of an external system, like a smartphone, that sends the to a cloud server.

In [14], G. Signoretti (et al) evaluate how dependable are OBD Edge devices, analyzing the interface behavior during the experimentation process, as well as the percentage of failure of the communication interface. While traditional OBD devices require an intermediary device, a smartphone or an IoT device, in order to transfer data from the OBD to the Cloud, the OBD Edge devices are equipped with Wifi, 3G/4G or LPWAN that enable communication with cloud services that consume the data generated by these devices. The work developed by G. Signoretti (et al) designs a set of experiments to compare two Edge OBD and after obtaining the experimental results, the most modifiable scenario, measured by the experiments, is then altered to achieve a better dependability.

2.1.4 Data Services — Fleet Management

To design a fleet management system it is necessary to understand what data services actually serve such a system. This section aims to explore the research found on fleet management data services as a whole, focusing on what services must be present in a system for it to be considered a fleet management system.

In [10], Szczepański (et al), as mentioned previously, develop a mathematical model to approach the task of managing a fleet of vehicles. The model itself presents an equation to represent the "Model of Car Fleet Management in Enterprise" in which *MCFME* = <*TPOJ, QPOJ, QU,CWK, MFP, FKO, PWDFM*>, where: **TPOJ** — set of available vehicle types; **QPOJ** — vector of technical and operational parameters of vehicles taken during the assessment; **QU** — vector of task carried out by the enterprise; **MFP** — set of methods financing the replacement of vehicles in an enterprise; **FKO** — set of sub criteria for assessing the vehicles assignment for tasks; **PWDFM** — decision making procedure in car fleet management.

2.1.5 Data Services — Vehicle Data

With the knowledge acquired in the previous research of devices and sensors, it is clear that the data sources to explore are smartphones and the OBD system. The smartphone itself has been shown to be a reliable source of geographic location information regarding the vehicle, and the OBD system was purposefully made to provide detailed information about the different vehicle components and provides a means through which repairers can diagnose vehicles faults. As such, this section will provide an explanation about what information can these two devices convey, and examples as to how the information conveyed as been used in other bodies of work.

Technically, the On-Board Diagnostics (OBD) supports seven distinct services for operation [15], synthesized in table 2.1. Each service provides an intrinsic set of commands which return the information that is gathered by the sensors and made available by the OBD.

Modes	Description
1	Display current real-time vehicle data
2	Display vehicle data related to the last Diagnostic Trouble Code generated
3	Display stored Diagnostic Trouble Codes
4	Clear stored Diagnostic Trouble Codes and values
5	Request results for the O2 sensor monitor tests
6	Request results for continuous and non-continuous monitor systems tests
7	Request emissions-related DTCs generated in the last driving cycle
8	Enables third-party devices to control the operation of an on-board system.
9	Display vehicle information
0A	Request permanently store DTCs related to emissions of air pollutants

Table 2.1: OBD-II Services

The first service provided, as the description states, returns real-time data being generated by the vehicle; this involves, but is not limited to, information such as the current speed, revolutions-per-minute, time since the engine started and engine coolant temperature. The second service stores the information generated along with a specific Diagnostic Trouble Code; when a failure occurs and is registered by the OBD in a freeze frame, the system also stores the vehicle speed and the engine's revolutions-per-minute among other pieces of information that are relevant diagnose faults. The third service, simply put, displays the Diagnostic Trouble Codes recorded by the OBD system, independent of any time frame, meaning that if a fault has not been detected in a specific drive cycle but in a previous cycle instead, the fault code is still stored. The fourth service clears all fault codes from the system, along with any freeze frame data stored along with the generated DTC; this can be used to turn off the Malfunction Indicator Lamp. Services five and six return the results stored for different sensor monitor tests, specifically the tests performed by the O2 sensor monitor in case of service five, and in case of service six tests performed by continuous and non-continuous monitor systems.

The aforementioned Diagnostic Trouble Codes are the mechanism through which the OBD reports the faults detected by the sensors it has collecting data. Technically, the OBD system uses five character codes to report faults in specific components; all of the characters in each code provide deeper insight into the problem in question when read from left to right. The first character specifies the system reporting the fault: either the Powertrain system, the Body system, the Chassis system or the Network system. The second character, when zero, indicates if the code generated is a generic code, implemented by all manufacturers, or if the code is manufacturer specific, when it's value is one. The third character specifies which sub-system generated the fault; it can hold values from zero to nine, representing sub-systems such as the emission management system, the injector circuit system or the emission control system, among others. The last two characters are variable and represent particular problems detected by the OBD.

Powertrain Category In the Powertrain category, most of the components being analyzed are some of the most crucial for the vehicles normal functioning. The components in this category include, but are not limited to, the Engine Control Unit, the Fuel System, the Turbocharger/Supercharger System, the Cooling System, the Air Inlet Sensor, the Exhaust System, the Fuel Injection Circuit, the Ignition Circuit, the Catalyst Temperature Sensor.

Chassis Category The components analyzed in the Chassis category are passive security mechanisms or act as experience-enhancing for the vehicle users. In the category, some of the components included are the following: the Wheel speed sensor, the Brake and ABS system, the Traction control system, the Tire Pressure Monitor, the Vehicle tracking system, the Transfer Case.

Body Category In the Body category, most of the components are situated in the vehicle cabin, although not all. Some of the components included in this category are: the Airbags, the Seat Sensors, the Steering Column.

Network Category The Network category offers the most distinct faults from all the categories. The faults stored in this category simply portray communication failures between the Engine Control Unit and the respective component, along with software incompatibilities present in the whole system.

Although the data provided by the OBD is unparalleled, there is information that can not be registered and conveyed by the system, and as such, there is the need to obtain such information using other methods. Outside of the scope of data made available by the OBD, by using the smartphone's GPS, it is possible to pinpoint the location of the smartphone and, by extension, the vehicle it is currently riding on; along with information about the vehicle's current speed.

In [16], Malekian (et al) design and develop a wireless OBD system with the goal of providing information regarding vehicle fuel consumption. The measurement process and the data it gathers — as developed in the aforementioned work — was done in response to the statement that "approximately 27% of the total carbon dioxide(CO_2) emissions were as a result of the combustion of fuel from vehicles". This work, developed in 2016, sets an example for how the OBD technology can be used in the field of fleet management by acting as a source for the data generated the company vehicles.

2.1.6 Key Performance Indicators

In any business, measuring performance is an important part of managing and improving the work produced and the revenue gained. KPIs are a tool designed for this purpose, as they constitute a set of values against which to measure the performance of an organization or any specific activity.

In [17], Fleetio's content marketing specialist defines fifteen metrics that fleet managers should be tracking to evaluate their respective fleets. The metrics identified are divided into three different groups: Maintenance metrics, Cost metrics and Driver and Asset metrics. The following table 2.2 is a summary of the categories and the respective metrics, identified by Fleetio.

Category	KPI
Maintenance Metrics	Inspection Results
	Preventive Maintenance Practices
	Diagnostic Trouble Codes
	Repair Turnover
	Odometer Readings
Cost Metrics	Fuel Management
	Total Cost of Ownership
	Asset Utilization
	Vehicle Replacement
	Parts and Inventory
Driver and Asset Metrics	GPS and Telematics
	Accidents and Safety
	Driver Assignments
	Technician Productivity
	Performance Metrics

Table 2.2: Fleetio Metrics

In [18], the company Utilimarc surveyed more than one hundred utility companies hosted in North America, as to what the KPIs each company uses to manage their fleets. The results of the survey are not limited to the summary of all KPIs, listed by the surveyed, but also present the frequency of reporting for each indicator, the audience that reports each indicator, if the indicator is required by the customer department or by management department, if the indicator is used to evaluate fleet performance, and finally some of the business decisions that resulted from tracking each indicator. The following figure 2.1

visually represents the KPIs tracked by the surveyed companies.

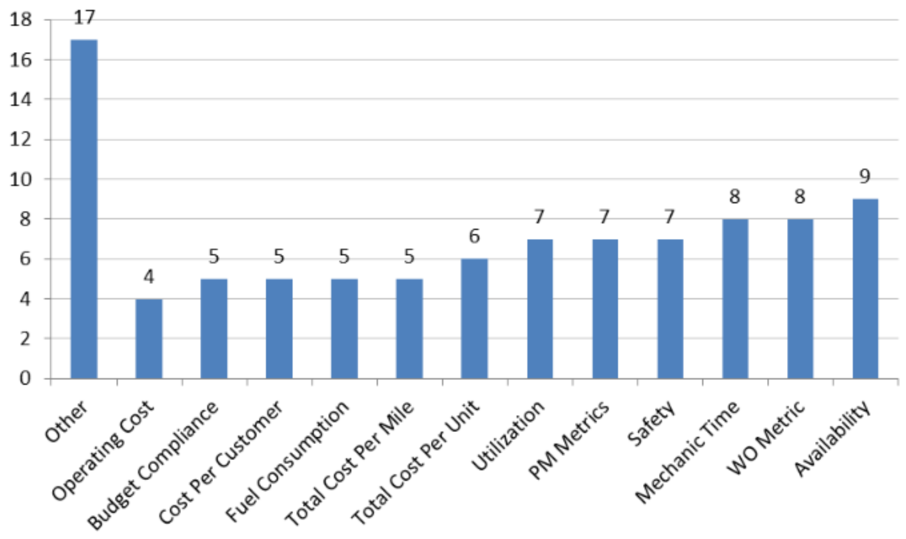


Figure 2.1: Utilitmarc Survey Results

In [19], Sanna-Mari sets out to solve two problems for the Valmet transportation team: First, establish a relevant KPI dashboard for the transportation team, and second study the relevance of an on-time delivery performance indicator. The conclusions of this body of work indicate that an organization should not approach KPI measurements from the perspective of what other organizations are measuring. Meaning that each organization should ponder what indicators are valuable from an internal point of view; nevertheless, KPIs established and measured by other companies serve as an example for what indicators can be tracked in a specific area of work.

The first two bodies of work mentioned previously in this section present examples about what KPIs can be measured in the field of fleet management, and the last work establishes a strategy about how to approach KPIs in general.

2.1.7 Conclusions

Answering the questions posed initially is the last step before concluding the literature review.

The first question **Q0**, and arguably the core question, of defining “Fleet Management” is answered with the following statement: Fleet Management is the set of actions that allows for an individual or a society of individuals to control and monitor a fleet of heterogeneous or homogeneous vehicles and the interactions with customers or fleet users (i.e drivers). Monitor assumes being able to gather data regarding the vehicles of the fleet, the users and customers of the fleet and the interactions between the vehicles and the users and customers (i.e trips). Controlling the fleet assumes being able to take actions based on the data presented by monitoring, be it changing the fleet size based on activity rates for each vehicle, or sending a vehicle to maintenance based on the data gathered from a vehicle.

Concerning question **Q1**, the gatherable data is comprised of two distinct groups — the data generated by the vehicles in the fleet, and the data generated by the human interactions with the vehicles themselves.

As for **Q2**, taking the work produced by Szczepański (et al) in [10] as an example, it is necessary to integrate services that: monitor the set of vehicles available, their corresponding technical and operational parameters and their performance in the tasks performed; analyze the tasks carried out by the company with the vehicles; record and monitor the financing methods for the replacing or maintaining the vehicles themselves. In reality which data services are integrated mostly depend on the intentions of the company that acquires a fleet management system; nevertheless the data services, formerly specified, act as a starting point or examples for which can be integrated in a fleet management system.

Lastly, before answering **Q3** it is necessary to specify that the KPIs tracked are completely dependent on the company, it's wants and needs, as well the tasks the company performs or business model it employs. Nevertheless, as mentioned previously, Utilimarc surveyed fleet owning companies and detailed the KPIs tracked, which enables for a matching between the resulting KPIs and the data services analyzed previously.

KPI	**Data Source**
Operating Cost	User Generated Data and OBD
Budget Compliance	User Generated Data
Cost per Customer	User Generated Data
Fuel Consumption	OBD — Fuel Readings
Total Cost per Mile	OBD
Total Cost per Unit	User Generated Data
Utilization	User Generated Data
Preventive Maintenance Metrics	OBD and User Generated Data
Safety	OBD and User Generated Data
Mechanic Time	User Generated Data
Work Order Metric	OBD
Availability	User Generated Data

Table 2.3: KPI — Data Source Matching

Although the previous table generalizes the data source for each KPI, it is not without reason; as the actual data required depends on the company specific process to measure each individual KPI. For the purposes of this internship the KPIs that will be tracked are **Fuel Consumption**, **Availability** and **Utilization** given their immediate availability.

2.2 Competitor Analysis

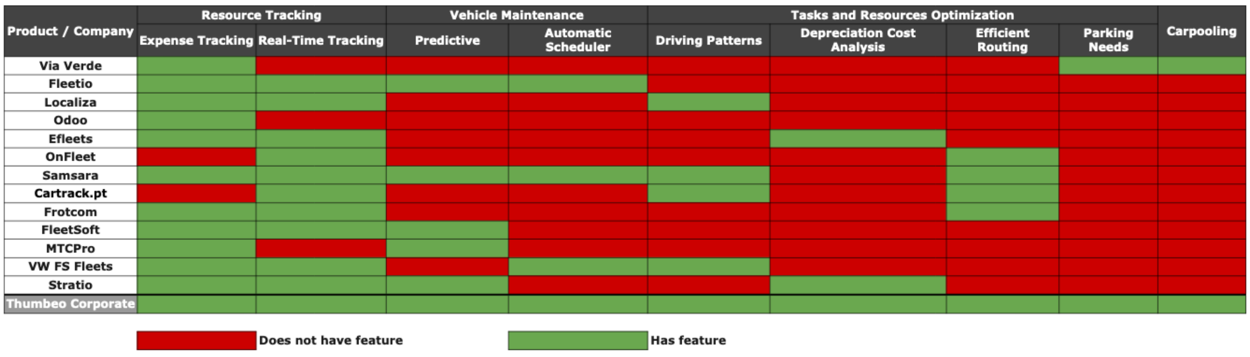
The present section serves the purpose of exploring and comparing possible *competitors* to Ubiwhere's solution. The process of analyzing competitors started by researching what products existed that focused on fleet management and a high-level comparison was performed. This high-level analysis was followed by a more in-depth comparison between the competitors that were deemed the most important and most influential in the field of fleet management in the context of Thumbeo Corporate's design and development.

2.2.1 High-Level Competitor Analysis

As a way to compare the different features present in Ubiwhere's solution, the high-level competitor analysis will approach four different categories for the main features of the product:

- **Resource Tracking** — Expense Tracking and vehicle real-time tracking.
- **Vehicle Maintenance** — Preventive and predictive maintenance.
- **Tasks and Resources Optimization** — Driving patterns, depreciation cost analysis, routing efficiency and matching, parking needs and fleet size analysis.
- **Carpooling** — The act of sharing the same vehicle in a trip with more than one person, in order to save resources by performing the same trip at the same time using different vehicles.

The high-level analysis of the competitors identified is presented in the following figure 2.2, followed by a general overview of how the different competitors stack up against the different categories specified previously.



Product / Company	Resource Tracking		Vehicle Maintenance		Tasks and Resources Optimization				Carpooling
	Expense Tracking	Real-Time Tracking	Predictive	Automatic Scheduler	Driving Patterns	Depreciation Cost Analysis	Efficient Routing	Parking Needs	
Via Verde	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩
Fleetio	🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥
Localiza	🟩	🟩	🟥	🟥	🟩	🟥	🟥	🟥	🟥
Odoo	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Efleets	🟩	🟩	🟥	🟥	🟥	🟩	🟥	🟥	🟥
OnFleet	🟥	🟩	🟥	🟥	🟥	🟥	🟩	🟥	🟥
Samsara	🟩	🟩	🟩	🟩	🟩	🟥	🟩	🟥	🟥
Cartrack.pt	🟥	🟩	🟥	🟥	🟩	🟥	🟩	🟥	🟥
Frotcom	🟩	🟩	🟥	🟥	🟥	🟥	🟩	🟥	🟥
FleetSoft	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥
MTCPro	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥
VW FS Fleets	🟩	🟩	🟥	🟩	🟩	🟥	🟥	🟥	🟥
Stratio	🟩	🟩	🟩	🟥	🟥	🟩	🟥	🟥	🟥
Thumbeo Corporate	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

🟥 Does not have feature 🟩 Has feature

Figure 2.2: High-Level Competitors Analysis

Regarding **Resource Tracking**, most of the competitors possessed both sub-categories of **Expense Tracking** and **Real-Time Tracking**. There were some exceptions: Via Verde [20], Odoo [21] and MTC Pro [22] do not implement real-time tracking; OnFleet and Cartrack [23] do not implement expense tracking.

The comparisons in the topic of **Vehicle Maintenance** were more sparse among the competitors analyzed, meaning there were not many competitors that implement this set of features. Fleetio [24] and Samsara [25] implement **Predictive Maintenance** and **Automatic Maintenance Scheduling** features into their products. FleetSoft [26], MTCPro [22] and Stratio Automotive [27] implement only **Predictive Maintenance** features. Finally, VW FS Fleets [28] implements only **Automatic Maintenance Scheduling**.

The categories present in the topic of **Tasks and Resources Optimization** were the most neglected by the majority of the competitors. **Driving Patterns** features were implemented by four competitors: Localiza [29], Samsara [25], Cartrack [23] and VW FS Fleets. **Depreciation Cost Analysis** was the second most neglected category, being implemented only by Efleets [30] and Stratio Automotive [27]. **Efficient Routing** was the most implemented feature of this topic, being present in OnFleet, Samsara [25], Cartrack [23] and Frotcom [31]. Lastly the most neglected category, **Parking Needs** was only implemented by Via Verde [20].

The **Carpooling** feature is the main difference between Thumbeo Corporate and the remaining solutions analyzed, as none of the solutions researched implemented features even closely related to ride-sharing.

After these conclusions, the analysis for some competitors deepened by focusing the features they implement and the business model they follow. These competitors are Via Verde [20], Stratio Automotive [27], Fleetio [24] and Samsara [25].

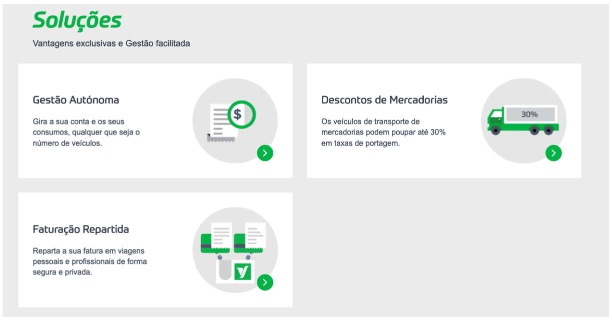
2.2.2 In-Depth Competitor Analysis

This section aims to break down the four direct competitors identified, by listing their features and defining their business models in order to compare them against Thumbeo Corporate's own features and business model.

Via Verde [20]

Important features:

- Expense tracking.
- Parking Needs
- Automatic shared invoices.
- Automatic toll payment.
- Discount for three types of commercial vehicles.


Soluções
Vantagens exclusivas e Gestão facilitada

Gestão Autónoma
Gira a sua conta e os seus consumos, qualquer que seja o número de veículos.

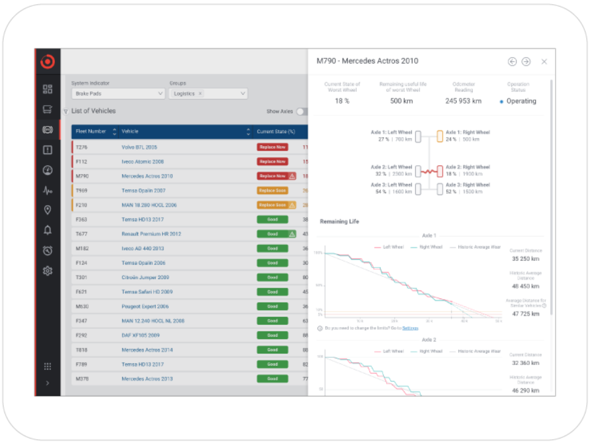
Descontos de Mercadorias
Os veículos de transporte de mercadorias podem poupar até 30% em taxas de portagem.

Faturação Repartida
Reparta a sua fatura em viagens pessoais e profissionais de forma segura e privada.

Competitor — Via Verde

Via-Verde as a company started by creating a service in Portugal that offered its clients an efficient method for automatically paying tolls through a device to be installed in the windshield of the vehicle. Nowadays, after further developing its service, Via-Verde now offers a service with some fleet management features in mind: the invoices generated paying a toll are now shared with the driver and the fleet manager; expenses are automatically tracked and Via-Verde shows them to their respective fleet managers to help them manage their expenses. Via-Verde also offers, through another one of it's service, an easy wait to pay for parking tolls, which is the feature designed as related to **Parking Needs**. This competitor was evaluated because it is has an already established position in the national market and provides products to help either a company or an individual manage one or more aspects of their vehicles. The data recorded by Via-Verde is distributed by their range of services, furthermore it is closed source and to be included in the development of this thesis an agreement would have to be made with Via-Verde.

Stratio Automotive [27]



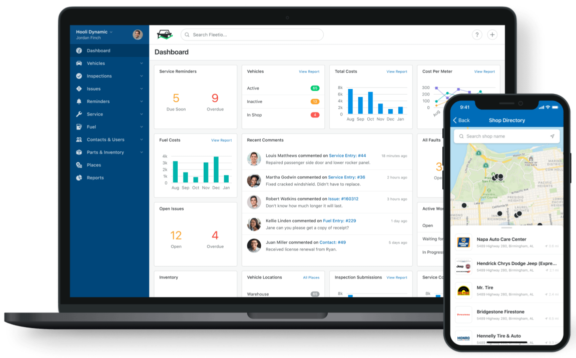
Competitor — Stratio Automotive

Important features:

- Vehicle Telemetry.
- Predictive Maintenance.
- Guided Decision Making.

Stratio Automotive offers an automated vehicle maintenance software directed at heavy-duty vehicles, designed to be used by fleet managers. This software is composed of three different components: ***stratio foresight***, a fault detector that utilizes predictive intelligence to detect anomalies and potential breakdowns before they occur; ***stratio pilot***, a fleet manager software component equipped with telematics intelligence that aims to optimize the different aspects of a fleet; ***stratio explorer***, a data navigator that helps fleet managers perceive the data gathered about their fleet from different perspectives to guide them to make the best decisions. Stratio Automative was evaluated for being a national newcomer to the fleet management market that provides a service based on components for different management needs using newer strategies to tackle the existing problems. Alike the previous competitor, the data stored by Stratio Automotive is also closed source and an agreement would have to be made with the competitor in order for the data to be used in the context of this thesis.

Fleetio [24]



Competitor — Fleetio

Important features:

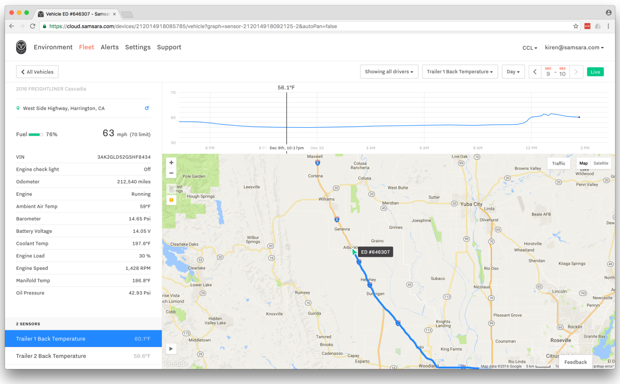
- Vehicle-Driver Assignments
- Preventive Maintenance Scheduling
- Fleet Reports
- Vehicle Tracking
- Total Cost of Ownership tracking

Fleetio is a fleet management software whose main selling point is comprised in four different aspects: the first aspect concerns the tracking of all equipment related to a given fleet, maximizing its lifespan while increasing profits, and generating reports about equipment usage and total cost of ownership; the second aspect is related to automating maintenance handling and consequent resolution, this is achieved by first identifying defects through methods like preventive maintenance, DVIR defects and recall and repair requests; the third aspect is directed to fuel optimization through simplifying fuel data collection and importing fuel costs directly from fuel cards; the fourth and final aspect relates to the integration of the various systems, that gather data related to the fleet, to improve the information shown to fleet managers. This competitor was evaluated because, alike the first, it is already established within the market of fleet management systems, albeit at an international level and tackles a different set of fleet management problems when compared to Via-Verde. Similarly to the previous services, the data collected by Fleetio is closed source and an agreement is necessary for a third-party to access said data.

Samsara [25]

Important features:

- GPS Fleet Tracking
- Fault Code Monitoring
- Usage-based Maintenance
- Real-time Route Tracking
- Historical Performance Analysis



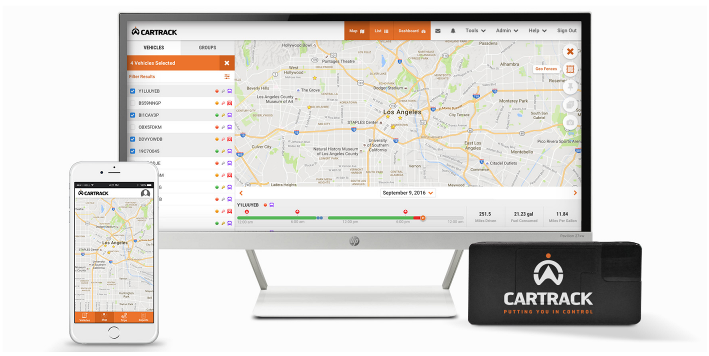
Competitor — Samsara

Samsara is a company that offers a product, of the same name, directed at fleets of heavy-duty vehicles, that employs a vast range of features related to fleet management. To name the most important: it has live vehicle location tracking, recording individual trips; the product also supplies fault code monitoring and Driver Vehicle Inspection Reports; it provides real-time route tracking and historical analysis of the routes taken. The less important features for this comparison include tracking trailers, providing dash cams to record driving, hosting in-cab Wi-Fi, and validating compliance through Electronic Logging Devices (ELD). All of these features are aimed at fleet managers of heavy-duty vehicles, which means that the product itself is not optimized primarily for light-duty vehicles. Samsara as a competitor was analyzed because alike the second competitor, it is a newcomer to the market of fleet management systems, but instead tackles an international market. Alike the previous competitors, the data harvested by Samsara is closed source and once again an agreement is necessary for a third-party to have access to the data.

Cartrack Portugal [23]

Important features:

- Real Time Vehicle Satellite Monitoring
- Stolen Vehicle Recovery
- Reports of Fleet Activity
- Driving Pattern Optimization Report



Competitor — Cartrack Portugal

Cartrack Portugal is the portuguese version of the Cartrack service provided by the company of the same name, co-founded and launched in 2004 in South Africa. The main features of Cartrack Portugal are providing stolen vehicle recovery solutions and top of the line vehicle tracking. As a fleet management system the product also provides reports regarding fuel consumption, tachograph data, OBD data and driver history. Cartrack was analyzed because it is an already established international product tackling very specific problems and a wider market when compared to Samsara. The specific problems are namely a more accurate real time vehicle tracking, with the ability to detect and track stolen vehicles. Similarly to the previous competitors, the data gathered by Cartrack is of a closed source nature and would required an agreement to be made in order for third-party entities to access the data.

Conclusions

In the competitors analysis performed it is clear that there are many fleet management systems provided to companies that have the need to manage their fleets. Many of these systems implement similar features, with arguably some implementation changes, but no system implements all the features identified in this analysis.

Concluding the competitors analysis it is worth noting that, if a new company needed a fleet management system to track the location of the vehicles with stolen vehicle recovery, along with expense tracking, parking needs, and a predictive maintenance system, this company would have to hire at least three distinct services. **Via-Verde** could provide the expense tracking and parking needs features, **Stratio Automotive** could provide the predictive maintenance, and **Cartrack** could provide the vehicle tracking the stolen vehicle recovery features. By hiring these three services, the company would also possibly duplicate expense tracking data, as both **Via-Verde** and **Stratio Automative** provide this functionality, and the company would fragment the data generated by the fleet through different systems. With solution proposed in this internship this new company could solve the problem of decentralized data in the systems hired for the different features.

2.3 Internship Contributions

Based on the work previously presented in this chapter, regarding the academic principles behind Fleet Management topics and the competitors analysis performed on fleet management applications, the following contributions are considered the key aspects of this internship in this chapter:

- The structure of a research methodology for *State of the Art*, consisting on a set of questions and respective answers pertaining to Fleet Management topics.
- The research performed on competitors available in the fleet management industry, pertaining to various types of competitors ranging from the established to the newcomers.

This chapter and its development also paved the way for the next two Chapters 5 and 6 in terms of the requirements to elicit and the architecture to define.

This page is intentionally left blank.

Chapter 3

Thumbeo Corporate

To improve an already existing solution, it is first necessary to study the concepts that lead to it's ideation, as well as the thought process behind it's first inception. As such, the present chapter will detail the aspects of Thumbeo Corporate as a product, along with the characteristics that differentiate it.

3.1 The Product

The present section presents a description of Thumbeo Corporate, the solution presented by Ubiwhere to help fleet managers in their decision-making process, and answers questions regarding different aspects of the product itself. The following questions will be answered: *What is the product? Who is the product for? Why is the product different?* After answering the previous questions, the expectations on the final product of the internship will be presented as a way to describe the contributions made by this internship in the context of Thumbeo Corporate.

What is *Thumbeo Corporate*? *Thumbeo Corporate* is Ubiwhere's fleet management system. As a product and a service, it focuses on providing features that enable company employees to share rides between themselves and enable fleet managers to control and optimize their fleets.

Who is *Thumbeo Corporate* for? The target audience for Ubiwhere's solution are companies owning small or medium-sized fleets, between 20 and 200 vehicles, that operate in Portugal. These companies generally concern themselves with at least one of the following sectors: distribution of people or goods, sales, construction or waste collection.

Why is *Thumbeo Corporate* different? Based on the results of the Competitor Analysis in Chapter 2, *Thumbeo Corporate* innovates by providing mobility as a service while covering for the fleet needed by the service itself. The product also provides a white label solution for companies that like their brand present on the products they use daily.

One of the core principles behind Thumbeo Corporate's innovation is the idea of building a system based on bundled features to better correspond to the needs of each company.

Some companies might want to keep their already established method of managing the fleet but need to give their collaborators a ride-sharing platform; others might need the whole set of features implemented by Thumbeo Corporate; while others might only need a way of managing their respective fleets but might choose to not adopt ride-sharing into their company's mobility policy.

3.2 The Problem

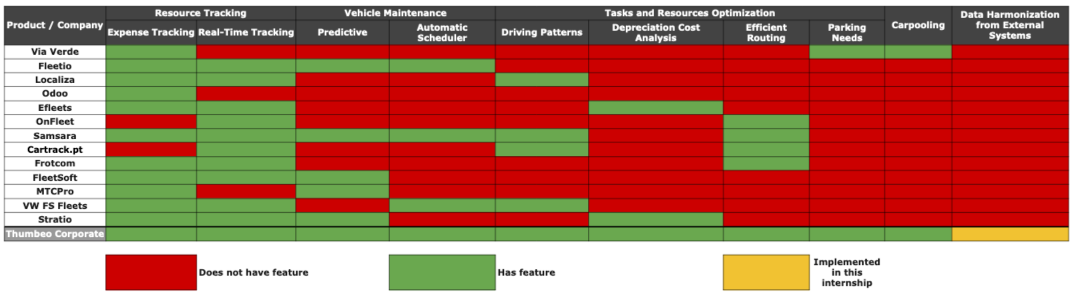
At the start of the internship, Thumbeo Corporate as a product had stagnated and it was not at all clear why. After the study of the topics regarding fleet management and the competitors analysis presented in Chapter 2 it was clear that Thumbeo Corporate, alike the competitors analyzed, did not offer a solution to import data from external systems, leaving fleet managers with yet another system that they must use to manager their fleets. As specified in the competitors analysis if a new company needed a fleet management system with different specifications it could have to hire services from more than one company offering a fleet management system, which would make it so that this new company would replicate the information of the fleet by the number of systems hired.

By studying the topics that led to Thumbeo Corporate's inception, present in Annex A, it became apparent that the product had innovate ideas on how to tackle problems for vehicle maintenance and depreciation, and expense management, but it nevertheless shared features with other fleet management systems. Moreover, the features that these systems share with one another make it so that fleet managers have to input similar data into the various systems used, which leads to the replication of data. Therefore, as previously explained, the internship aims to build upon a central solution that solves such problems as explained along this document.

3.3 Internship Contributions

Having already defined the product in this chapter, and having studied competitors in the previous Chapter 2, the problem with Thumbeo Corporate is clear. Despite the different features it provides, the product is yet another tool for fleet managers to use, which helps the decentralization of data regarding the company fleet. The core contribution of this internship to Thumbeo Corporate is that of developing a system containing a Data Harmonization System and a Fleet Management System that is both able to consume data from external systems and utilize it to perform Fleet Management tasks.

Revisiting the previously defined figure 2.2 in the competitors analysis, now altered to present the features that were added to Thumbeo Corporate.



Product / Company	Resource Tracking		Vehicle Maintenance		Tasks and Resources Optimization				Carpooling	Data Harmonization from External Systems
	Expense Tracking	Real-Time Tracking	Predictive	Automatic Scheduler	Driving Patterns	Depreciation Cost Analysis	Efficient Routing	Parking Needs		
Via Verde	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟥
Fleetio	🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥
Localiza	🟩	🟩	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥
Odoo	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Efleets	🟩	🟩	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥
OnFleet	🟥	🟩	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥
Samsara	🟩	🟩	🟩	🟩	🟩	🟥	🟩	🟥	🟥	🟥
Cartrack.pt	🟥	🟩	🟥	🟥	🟩	🟥	🟩	🟥	🟥	🟥
Frotcom	🟩	🟩	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥
FleetSoft	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MTCPro	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
VW FS Fleets	🟩	🟩	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥
Stratio	🟩	🟩	🟩	🟥	🟥	🟩	🟥	🟥	🟥	🟥
Thumbeo Corporate	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨

🟥 Does not have feature

🟩 Has feature

🟨 Implemented in this internship

Figure 3.1: High Level Competitors Analysis — New Thumbeo Corporate Features

The following Chapters 5 and 6 detail the requirements and the architecture of the system, and in Chapter 8 the results of the implementation are shown.

This page is intentionally left blank.

Chapter 4

Project Management

In the present chapter, the topics related to project management will be addressed. Starting with the software development methodology employed, moving onto the planning idealized, followed by risk identification and explanation, ending with the established success criteria.

4.1 Team Organization

The team involved in designing Thumbeo Corporate, as stated before, plays a part in studying the ever growing problem of fleet management and consequently designing and developing Ubiwhere's solution. This team is comprised of three members, each with their own role inside the project:

- **Product Owner:** André Duarte
- **Senior Developer:** João Garcia
- **Junior Developer:** Francisco Santos

4.2 Software Development Methodology

For the duration of the internship, the software development methodology followed is based on the Agile Manifesto Principles. The values proposed by the Agile Manifesto [32] that where fundamental for its choice as the methodology applied in the internship are the following:

- Individuals and interactions over processes and tools;
- Responding to change over following a plan.

As the product was idealized internally, the Agile paradigm is the best choice to deal with the rapid changes that will surely arrive with new breakthroughs in development or problems that might only appear at latter stages of the process. The first value mentioned corresponds to the individuals at Ubiwhere that worked on Thumbeo Corporate; their joined vision, knowledge and motivation are of greater importance to the product itself, rather than focusing on what tools the product should use or what processes should be employed because of their prestige or past usefulness to other projects. The second value is tied to the first in such as way that the changes that might occur, come from the individuals and their interaction along the project's development. Combining the investigation and knowledge of the people responsible for the project is bound to lead to changes and so, the development methodology applied should allow for the changes to happen fast and have their quality assured easily.

In the practical sense, regarding the implementation of the Agile methodology, there was one meeting between both developers every week. In this meeting, both developers would review the requirements elicited and the quality attributes defined, as well as the design of the architecture and the decisions regarding the technologies and its uses. Apart from these meetings between developers, the *Junior Developer* also had meetings every two weeks with the *Product Owner* in order to update it with the decisions made and to gather feedback that will be used to iterate the decisions made or the artifacts produced.

4.3 Planning

The planning of the project is structured in two different time periods corresponding with the respective university semesters. The first semester runs from February to June 2020, while the second semester runs from September 2020 to January 2021.

4.3.1 First Semester Plan

For the first semester, the plan built was divided into five different tasks, corresponding to this very report, the state of the art studied, the project management tasks, the requirements elicitation and the architecture design.

- **Report**: This stage is performed during the whole duration of the project. The subtasks involved in this stage are writing and reviewing the report;
- **State-of-the-Art Analysis**: This phase of the plan was important to understand the problem statement, study existing solutions and build a knowledge base to use during the whole project;
- **Project Management**: This stage was carried out to plan the different tasks needed to conclude the project. The success criteria and risk analysis were also defined during this stage;
- **Requirements**: During this phase the elicitation of functional requirements and the definition of the quality attributes was carried out;
- **Architecture Design**: This stage was crucial in development of the project. The following task were performed: research about architecture patterns to use when implementing the software; research possible technologies to use; define the architecture using diagrams; initial training on the technologies chosen.

4.3.2 Second Semester Plan

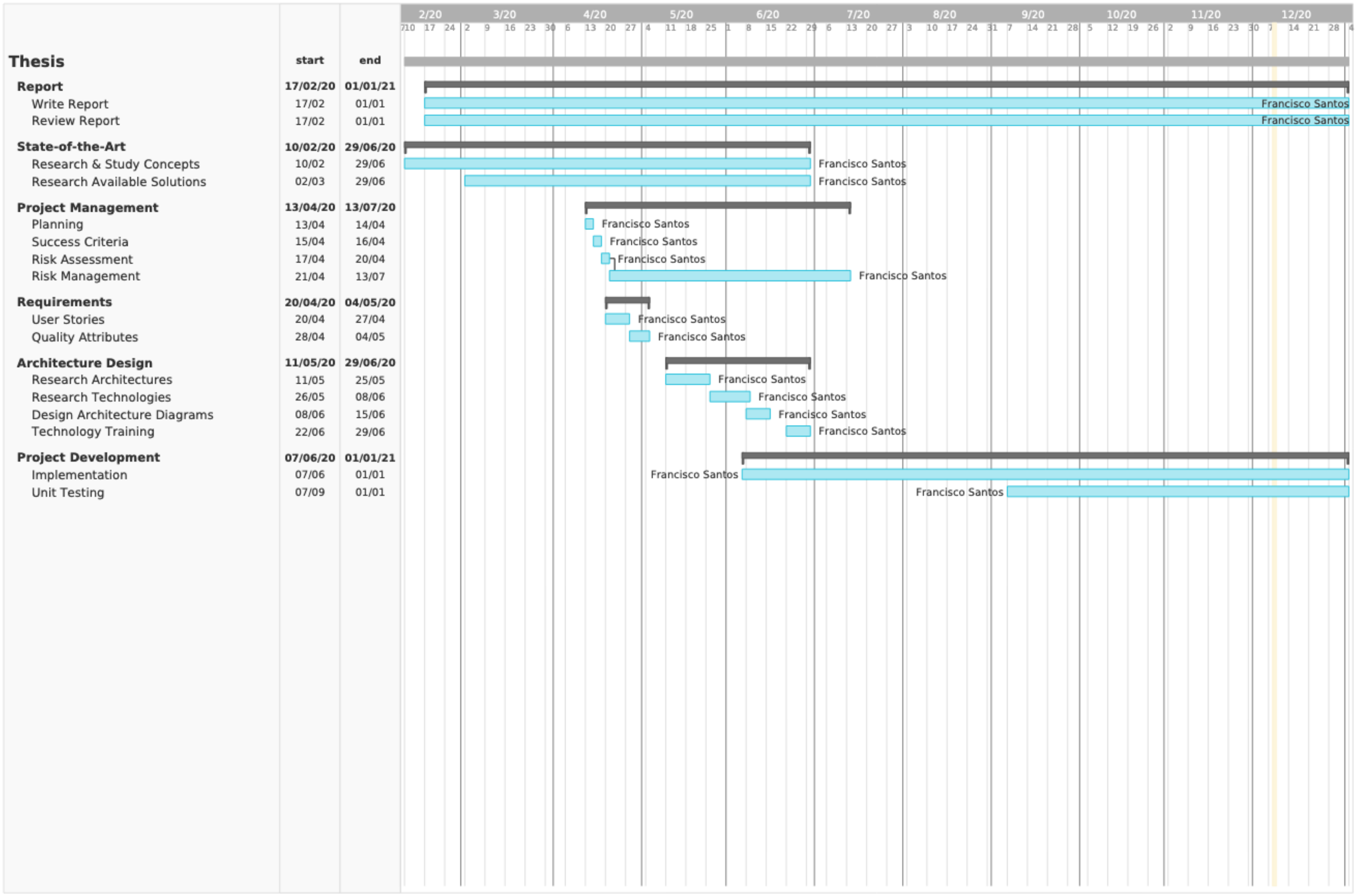
The plan designed for the second semester consisted on the completion of a new task — the development of the final product of the internship — while revisiting previously established phases, such as the State-of-the-Art definition, the project management, the requirements elicitation phase and the architecture design. This plan for the second semester was revisited based on the feedback provided in the intermediate presentation.

- **Report**: For the second semester, this stage will consist of acting on the feedback received from the intermediate defense and in the writing of the implementation chapter of this report.
- **State-of-the-Art Analysis**: This phase of the plan is revisited to continuously study the problem in question and to continue to search for solutions in the academic or entrepreneurial world.
- **Project Management**: This stage was revisited to assert the state of the project planning and to review the tasks proposed in the initial planning.
- **Requirements**: The requirements elicitation will be consistently iterated during the project duration in order to reevaluate the elicited requirements or defined new ones.
- **Architecture Design**: The architecture design phase will also be iterated during the project in order to comply with new or updated quality attributes.
- **Product Development**: This phase of the plan will be assigned for the implementation and testing of the product of this internship.

4.3.3 Gantt Diagrams

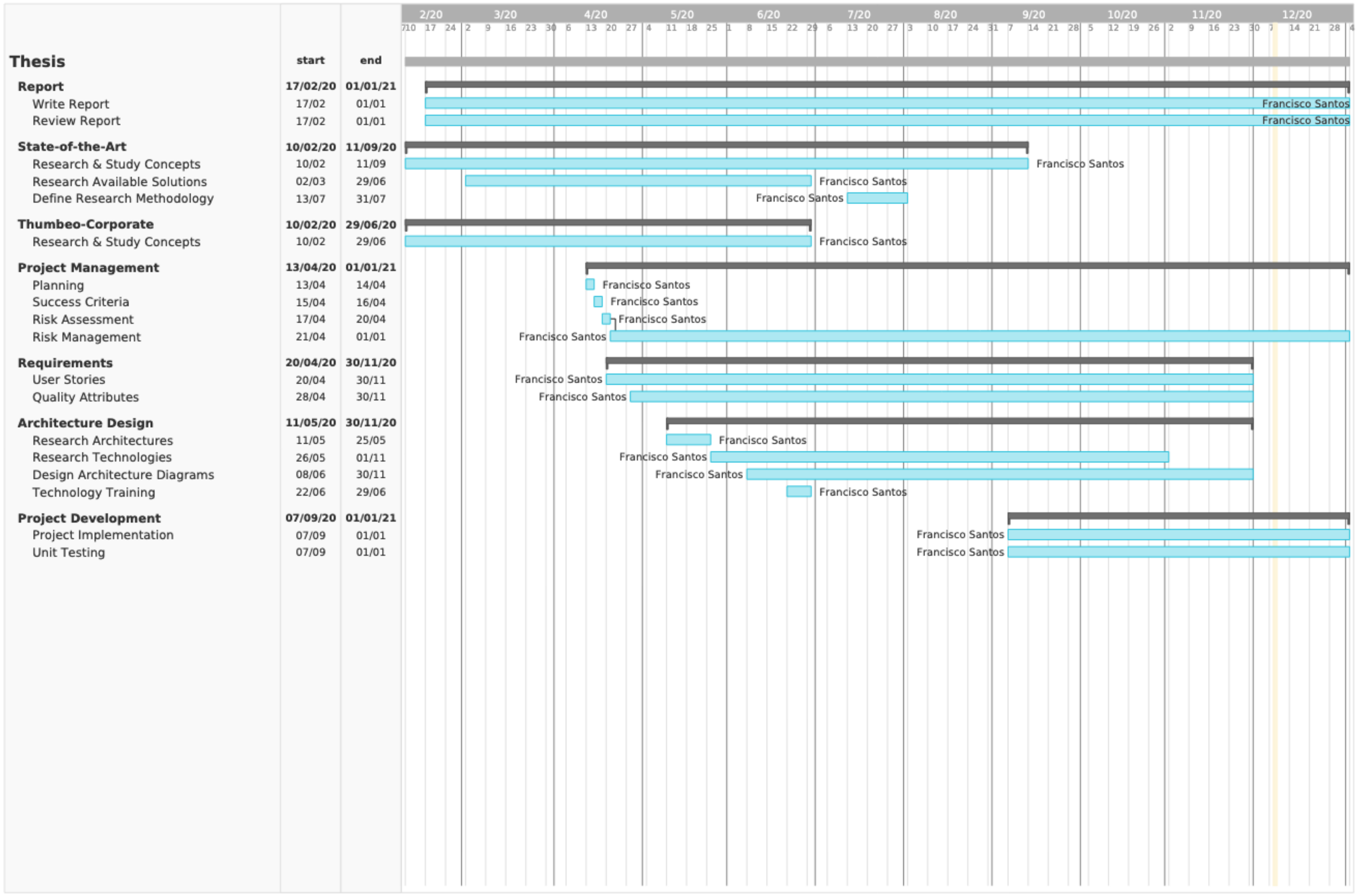
The first Gantt Diagram presented next is a visual representation of the effort that was expected from the internship and how that effort was split into the different tasks and subtasks mentioned previously. As for the second Gantt Diagram, it represents the real effort that was made and how that effort divided itself into the different tasks and subtasks, in which some of the subtasks are clearly different and arguably more detailed than expected.

The differences between the planned effort 4.1 and the real effort 4.2 exist for two main reasons, those being the faults existent on the initial plan, and the changes made to the plan in light of the feedback given in the intermediate presentation. The initial plan was faulty in the sense that it did not present the requirements elicitation process and the architecture design phases correctly; it presented the phases in a sequential manner, rather than the intended iterative manner of being revisited for the majority of the duration of the internship. The feedback provided in the intermediate presentation also required the State-of-the-Art research to be revisited and thought out in a different form. Although some of the previous performed work was still used in the new structure, it was mostly utilized for a new phase and a corresponding chapter named “Thumbeo Corporate” which explains the concepts behind the product developed by Ubiwhere.



Thesis	start	end
Report	**17/02/20**	**01/01/21**
Write Report	17/02	01/01
Review Report	17/02	01/01
State-of-the-Art	**10/02/20**	**29/06/20**
Research & Study Concepts	10/02	29/06
Research Available Solutions	02/03	29/06
Project Management	**13/04/20**	**13/07/20**
Planning	13/04	14/04
Success Criteria	15/04	16/04
Risk Assessment	17/04	20/04
Risk Management	21/04	13/07
Requirements	**20/04/20**	**04/05/20**
User Stories	20/04	27/04
Quality Attributes	28/04	04/05
Architecture Design	**11/05/20**	**29/06/20**
Research Architectures	11/05	25/05
Research Technologies	26/05	08/06
Design Architecture Diagrams	08/06	15/06
Technology Training	22/06	29/06
Project Development	**07/06/20**	**01/01/21**
Implementation	07/06	01/01
Unit Testing	07/09	01/01

Figure 4.1: Gantt Diagram — Planned Effort



Thesis	start	end
Report	**17/02/20**	**01/01/21**
Write Report	17/02	01/01
Review Report	17/02	01/01
State-of-the-Art	**10/02/20**	**11/09/20**
Research & Study Concepts	10/02	11/09
Research Available Solutions	02/03	29/06
Define Research Methodology	13/07	31/07
Thumbeo-Corporate	**10/02/20**	**29/06/20**
Research & Study Concepts	10/02	29/06
Project Management	**13/04/20**	**01/01/21**
Planning	13/04	14/04
Success Criteria	15/04	16/04
Risk Assessment	17/04	20/04
Risk Management	21/04	01/01
Requirements	**20/04/20**	**30/11/20**
User Stories	20/04	30/11
Quality Attributes	28/04	30/11
Architecture Design	**11/05/20**	**30/11/20**
Research Architectures	11/05	25/05
Research Technologies	26/05	01/11
Design Architecture Diagrams	08/06	30/11
Technology Training	22/06	29/06
Project Development	**07/09/20**	**01/01/21**
Project Implementation	07/09	01/01
Unit Testing	07/09	01/01

Figure 4.2: Gantt Diagram — Real Effort

4.4 Risk Management

The risk management plan constructed during the project planning phase was crucial to identify all the different factors that might lead to an unsuccessful internship, in this case, not reaching the Threshold of Success (ToS) criteria. To initiate this task, a way of classifying the risks, with respect to their probability and impact, was defined and is presented in table 4.1. After that, the risks themselves were gathered and written in table 4.2, and finally mitigation plans were created for all risks. The mitigation plans are present in table 4.3.

Scale	**Impact**	**Description**	**Probability**	**Probability Threshold**
1	Marginal	Can reach ToS without greater difficulty.	Low	<40%
2	Critical	Can reach ToS but with greater effort/cost.	Medium	between 40% and 70%
3	Catastrophic	Can not reach ToS.	High	>70%

Table 4.1: Risk Classification Scale for Impact and Probability

ID	**Risk**	**Probability**	**Impact**
R01	The technology being developed is state of the art; This might cause scope/requirements change based on new discoveries.	1	2
R02	The software being developed relies on data from external systems; This will affect the implementation in relation to what data can be obtained.	1	3
R03	The lack of resources and the changing in scope/requirements might delay or hinder the final delivery.	2	3

Table 4.2: Risk Management Analysis

Risk ID	**Mitigation Plan**
R01	Requirements and scope will be crafted to adequate changes that might appear at later stages in the development. The success criteria will also be written in a way that leaves some leeway for the requirements and scope to change.
R02	The data needed will be chosen based on its immediate availability and the system architecture is designed to ingest different types of data.
R03	The negotiations made on the scope or the requirements will consider importance and the value of the final delivery for all stakeholders involved.

Table 4.3: Risk Mitigation Plans

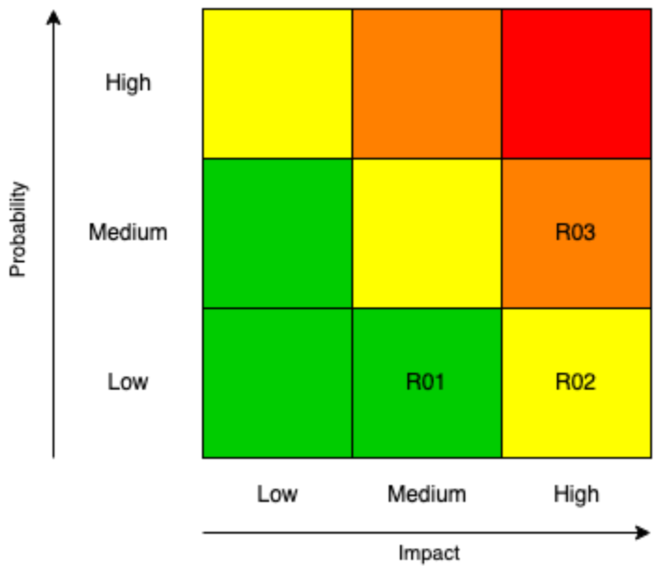


Figure 4.3: Risk Matrix

COVID Pandemic The pandemic that affected the world during this internship was pondered as a risk for the work, but given the fast adaptation of the team to a remote environment, combined with the immediacy at which the advisors responded with feedback when needed made it so that the impact of the pandemic was reduced to the point of it not being defined as a risk in the final version of this document.

4.5 Success Criteria

In order to evaluate the state of the project delivered, the following success criteria were established:

- The requirements specified and the architecture designed are approved by the existing stakeholders.
- Requirements classified as *"Must have"*, must be implemented at the time of the final delivery.
- The product developed satisfies the quality attributes proposed in their respective definitions in Chapter 5.

The previous success criteria represent the metrics through which the project will be deemed successful or not; to fail one of the success criteria means to fail the delivery itself.

This page is intentionally left blank.

Chapter 5

Requirements Specification

The present chapter will specify all requirements identified for the proposed work. Firstly the user personas identified are presented; next, the terminology used in the requirements and the structure of the functional requirements are identified and explained; this is followed by the restrictions imposed at the system. Lastly, the chapter ends with the elicitation of the functional requirements, along with the description of the quality attributes.

5.1 User Personas

Understanding the demographics targeted by the product is an important aspect of the requirements elicitation phase. To model the useful aspects of the target audience User Personas were used. These personas constitute a semi-fictitious representation of segments of the target audience; each representation is comprised of a name, important characteristics, and the goals and frustrations felt by the persona.

First Persona

Name:	**Albert Peterson**
Characteristics:	• Company CEO. • Owns a company fleet with 200 vehicles.
Goals:	• Obtain the optimal Total Cost of Ownership for the company's fleet. • Wants to include collaborators in the company's mobility plan.
Frustrations:	• Has to use multiple systems to analyze the fleet. • The multiple systems used are not always synchronized.

Table 5.1: First Persona — CEO

Second Persona

Name:	**Isaac Dawkins**
Characteristics:	• Fleet Manager • Manages a fleet of 50 vehicles
Goals:	• Wants to be able to view all aspects of the company fleet within a single system. • Wants to measure fleet performance.
Frustrations:	• Has to use multiple systems to manage the fleet. • The multiple systems used are not interoperable with each other.

Table 5.2: Second Persona — Fleet Manager

Third Persona

Name:	**Steven Piker**
Characteristics:	• Company Driver.
Goals:	• Wants to manage the company vehicle.
Frustrations:	• Has no way to track information generated by the vehicle. • Has no system to inform CEO or Fleet Manager about fleet related events.

Table 5.3: Third Persona — Driver

Fourth Persona

Name:	**Leonard Harris**
Characteristics:	• Company Collaborator.
Goals:	• Wants to reduce the time and money spent on everyday commuting.
Frustrations:	• Spends 30 to 40 minutes everyday in commuting. • Has no system to inform CEO or Fleet Manager about fleet related events.

Table 5.4: Fourth Persona — Collaborator

5.2 Terminology

In order to understand the requirements specified in the present work, it is necessary to define the terminology used in the specification of both functional requirements and quality attributes.

- **On-Board Diagnostics:** Vehicle system responsible for diagnosing faults in the vehicle's subsystems.
- **OBD Readers:** External device responsible for the gathering of data from the vehicle OBD system.
- **Data Harmonization System:** Set of components responsible for gathering and processing the data from the OBD system.

5.3 Requirement Structure

To allow for a more user-focused specification all functional requirements are specified using User Stories. The choice of this requirement elicitation technique also pairs well with the Agile Methodology used for project management, given that this technique focuses the requirements to be written from the perspective of the end user. As a result of the previous decision, the requirements specified have the following structure:

EPIC#ID

- **Epic:** User Story Epic
- **Description:** As a [role], I [want to], [so that]
- **Acceptance Criteria:**
  - **Scenario:** Predetermined business situation
    - **Given** a precondition
    - **When** an action is performed by an actor
    - **Then** a testable output is presented
- **Dependencies:** EPIC#ID
- **Priority:** Must Have, Should Have, Nice to Have

As a way to prioritize the requirements identified, the *MoSCoW* method was applied. This method applies the following scale:

- **Must have:** Requirements that are critical to the current delivery time frame in order for it to be a success.
- **Should Have:** Important but not necessary requirements.
- **Could have:** Desirable but no necessary requirements.
- **Won't have:** The least critical requirements agreed upon by the stakeholders.

5.4 Restrictions

The restrictions identified for this system present themselves in three different types, and impose constraints on how the system will be built. The present section will present said restrictions and justify their existence.

5.4.1 Business Restrictions

Business restrictions are described as constraints related to business resources such as the quantity and quality of the team in charge of the project, the time constraints imposed on the project.

- **R01 — Development Time Frame:** The system proposed in this report will be developed by a single developer for four months.

5.4.2 Legal Restrictions

Legal restrictions, like the name implies, are related to constraints imposed by European legislation.

- **R02 — GDPR Compliance:** The system to be developed must be compliant with the European General Data Protection Regulation (GDPR) [33].

5.4.3 Technical Restrictions

Technical restrictions can be described as the constraints bound to the technologies used, the systems that should be integrated with the project in development or the different platforms that should be able to interact with the system.

- **R03 — Integration with Thumbeo Corporate:** The system to be developed must be integrated with a pre-existent solution developed by Ubiwhere.
- **R04 — Open Source Technologies:** The system under development must make use of Open Source Technologies in order to allow for unrestricted updates and the availability of the source code in case modifications have to be done.

5.5 Functional Requirements

The functional requirements specified were idealized during the State-of-the-Art research, drafted for the first time once the Requirement Specification phase was undertaken and iterated when needed until the final version was achieved.

The requirements specified were divided by their respective Epics for various reasons. Needing to organize the different features based on their related implementations is the first reason; the second reason regards the need for sets of features that can be easily described when attached to a delivery; and lastly these epics provide agility when organizing large amounts of features to be developed. The following lists comprehend a summary of the requirements divided by their epics and roles, the full elicitation of the requirements along with their acceptance criteria can be reviewed in Annex B.

Epic: Data Harmonization, Role: Fleet Manager

1. Import vehicles from an external system, in order to centralize vehicle information in a single system.
2. Import users from an external system, in order to centralize user information in a single system.
3. Import rides from an external system, in order to centralize trip information in a single system.
4. Import driving patterns from an external system, in order to centralize driving pattern information in a single system.
5. Import OBD data from external OBD Readers, in order to centralize the data in a single system.

Epic: Statistics Collection, Role: Fleet Manager

1. See statistical data gathered concerning fuel consumption, to analyze fuel spending of the vehicles of the fleet.
2. See statistical data gathered concerning vehicle utilization, to analyze if the vehicles are being used according to the company's intent.
3. See statistical data gathered concerning vehicle availability, to analyze if the vehicles are available when they are needed.

Epic: Vehicle Management, Role: Fleet Manager

1. Add a vehicle to my fleet so that I can manage it using the web application.
2. Edit the information about a specific vehicle so that I can update the data kept on it.
3. See all my vehicles so that I can have a global view of my fleet.
4. Archive a vehicle of my fleet so that I can tell the system that the vehicle in question is no longer operational in my fleet.

Epic: Permission Management, Role: System Administrator

1. Give fleet management permissions to a user, so that the user can act as a fleet manager inside the web application.
2. Remove fleet management permissions from a user, so that the user can no longer act as a fleet manager inside the web application.

Epic: Permission Management, Role: Fleet Manager

1. Give driver permissions to a user, so that the user can act as a driver inside the web application.
2. Remove driver permissions from a user, so that the user can no longer act as a driver inside the web application.
3. Give a driver permissions to drive one of my vehicles, so that I can provide him with a means of transportation.
4. Remove a driver's permissions to use one of my vehicles, so that I can stop him from driving a specific vehicle.

Epic: Vehicle Evaluation, Role: Driver

1. Make an evaluation of the vehicle's condition after a trip, so that I can notify the fleet manager about occurrences during said trip.
2. Report a fault in vehicle that I am currently driving, so that I can inform the fleet manager.

Epic: Fleet Manager Notifications, Role: Fleet Manager

1. Is notified when one of my drivers starts a trip, so that I can be up to date with my vehicles state.
2. Is notified when one of my drivers finishes a trip, so that I can be up to date with my vehicles state.
3. Is notified when one of my drivers reports a vehicle fault, so that I can properly plan the maintenance for the reported fault.

Epic: Driver Notifications, Role: Driver

1. Is notified when I am assigned a new vehicle, so that I can be up to date with which vehicles I can use.
2. Is notified when my access to a vehicle has been removed, so that I can be up to date with which vehicles I can use.
3. Is notified when one of the vehicles assigned to me has been declared unavailable, so that I can be up to date with which vehicles I can use.

5.6 Quality Attributes

The quality attributes, often also called non-functional requirements, are defined as the requirements that will impact the architecture of the system itself. The quality attributes redacted for the system are defined using the following structure.

Priority: Set of priorities established based on the architecture impact and the business value. Scale will be High (H), Medium (M) and Low (L). The representation used is <architecture impact, business value>

Source of stimulus: Entity producer of the Stimulus

Stimulus: Condition that incites an action from the Artifact

Environment: Set of circumstances under which the Stimulus occurs

Artifact: Entity that is stimulated on a set Environment

Response: Action produced by the Artifact as a reaction to the Stimulus

Metric: Measurement criteria to evaluate the quality of the Artifact Response

Confidentiality

This quality attribute is present to provide drivers with the right to not have information, that is deemed confidential, disclosed to either other users in the system, that are not allowed to access the respective data; or to third party systems that are not required for the proper functioning of this system.

Priority: <L, H>

Source of stimulus: Drivers.

Stimulus: The drivers right to not have their sensitive information and information about their location to other users in the system or to third parties outside the system itself that are not defined as a requirement for the system itself to function properly.

Environment: Production.

Artifact: System.

Response: Sensitive information is only shared with the driver who defined it; or, if needed, with their respective fleet managers. Information about a driver's location is only disclosed to the fleet manager.

Metric: The system does not disclose any of the previously mentioned information with unauthorized users.

Data Integrity

Priority: <L, M>

Source of stimulus: Developers and System Integrators.

Stimulus: Need to update the harmonization system based on the a new system to integrate.

Environment: Development and maintenance.

Artifact: System.

Response: The harmonization system processes the data provided if the data follows the data structure set out by Ubiwhere.

Metric: The new system integrated is made available in one week of work.

Interoperability

Ubiwhere and its collaborators have shown great interest in designing a system that communicates seamlessly with various data collection devices. Therefore, this interest is defined as a quality attribute of the system to design and build.

Priority: <H, M>

Source of stimulus: Ubiwhere.

Stimulus: Need to change the acrshortobd data collection device if the problem requires it or a client demands it.

Environment: Production.

Artifact: System.

Response: The identification of vehicle faults is independent of the data collection device.

Metric: The identification of vehicle faults is based on the data available and not the device that collects the data.

Priority: <H, M>

Source of stimulus: Ubiwhere.

Stimulus: Need to aggregate data collection services that consume different types of data.

Environment: Production.

Artifact: System

Response: The data ingestion system is able to consume data from different sources.

Metric: At least two data collection services are integrated with the data ingestion system.

Modifiability

The ability to easily develop new features and maintain legacy ones is the driving force for this specific quality attribute. The developers at Ubiwhere need to maintain the system with the lowest amount of effort possible and develop new features with the assurance that new work will force the least amount of changes in the work already produced.

Priority: <H, L>

Source of stimulus: Developers.

Stimulus: Need to implement a new feature to the system.

Environment: Development and maintenance.

Artifact: Web Application.

Response: The system maintains the intended behavior after the new feature is added.

Metric: The feature can be successfully developed, integrated and tested within a week of work.

Priority: <H, M>

Source of stimulus: Developers.

Stimulus: Need to integrate a new data collection service with the data ingestion system.

Environment: Maintenance and Development.

Artifact: System.

Response: The system maintains the intended behavior after the new data collection service is integrated.

Metric: The new data collection service is integrated and tested within a week of work.

Scalability

This quality attribute is driven by the necessity to scale the already existing fleet management system in terms of the subsystems it can collect data from. The fleet managers or the CEO of any given company might need to integrate a new or pre-existent system with Thumbeo Corporate and the system should scale to integrate the new subsystem.

Priority: <H, M>

Source of stimulus: Fleet Managers and CEOs.

Stimulus: Need to integrate a new data collection service with the data ingestion system.

Environment: Maintenance and Development.

Artifact: System.

Response: The system maintains the intended behavior after the new data collection service is integrated.

Metric: The new data collection service is integrated and tested within a week of work.

Priority: <H, M>

Source of stimulus: System.

Stimulus: Need to deal with peaks of data being imported into the system.

Environment: Production

Artifact: System.

Response: The system maintains intended behavior while dealing with a peak of data being loaded into the database.

Metric: The messaging system auto-scales itself after the CPU usage threshold exceeds 80%.

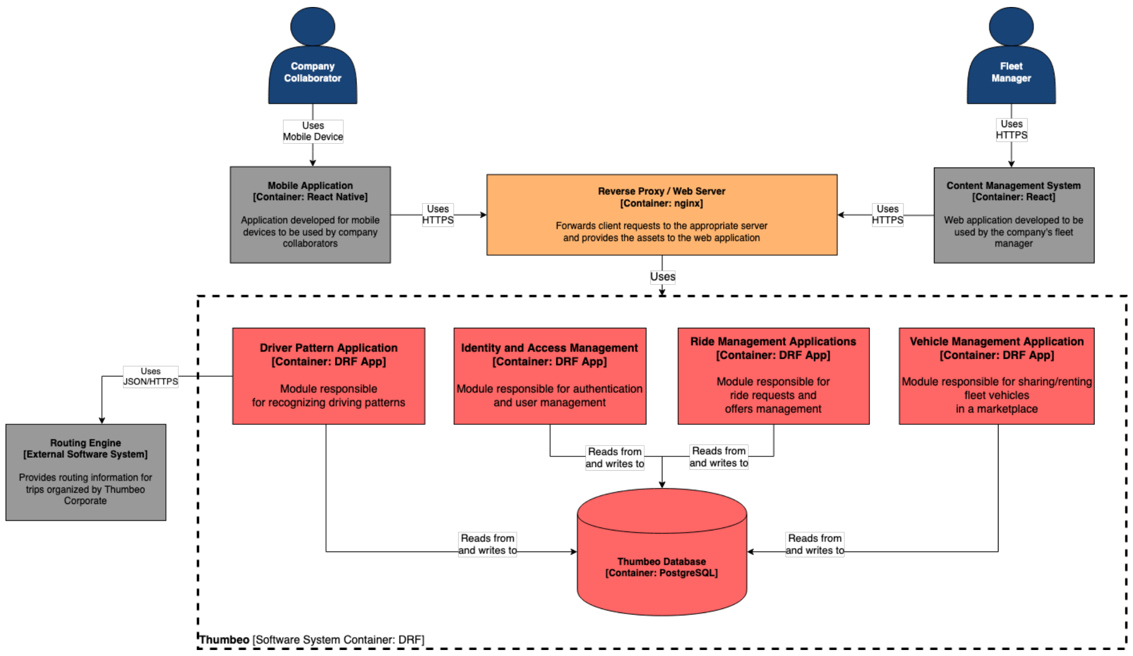
Chapter 6

Architecture and Technology

Following the elicitation of the functional requirements and quality attributes, this chapter will document the process through which the architecture of the system was built. The process relied on the research of what tools exist for the purposes of the system and how can said tools be orchestrated together to fulfill the goals of this internship. Firstly, the architectural decisions will be discussed and presented, followed by the technologies adopted, moving onto the architectural definition through diagrams and finalizing by validating the compliance of the architecture with the quality attributes established.

6.1 Project Background

Before introducing the architecture developed for the purpose of this internship it is valuable to present the already existing architecture for one of the two systems that will be integrated with the system to develop. This system is a product, also developed by Ubiwhere, named Thumbeo and it is a ride-hailing service.



Company Collaborator
Fleet Manager
Uses Mobile Device
Uses HTTPS
Mobile Application [Container: React Native]
Application developed for mobile devices to be used by company collaborators
Uses HTTPS
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses HTTPS
Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses
Uses JSON/HTTPS
Driver Pattern Application [Container: DRF App]
Module responsible for recognizing driving patterns
Identity and Access Management [Container: DRF App]
Module responsible for authentication and user management
Ride Management Applications [Container: DRF App]
Module responsible for ride requests and offers management
Vehicle Management Application [Container: DRF App]
Module responsible for sharing/renting fleet vehicles in a marketplace
Routing Engine [External Software System]
Provides routing information for trips organized by Thumbeo Corporate
Reads from and writes to
Reads from and writes to
Reads from and writes to
Reads from and writes to
Thumbeo Database [Container: PostgreSQL]
Thumbeo [Software System Container: DRF]

Figure 6.1: C4 Model — Existing System

As the previously figure 6.1 shows, the existing system was composed of four different applications within a *django-rest-framework* monolith. The **Identity and Access Management** application serves to manage user information and authenticate users into demanded and authorized resources. The **Vehicle Management Application** is responsible for managing the records of fleet vehicles, and provide functionalities to create, update, read or delete vehicle information. The **Ride Management Application** is responsible for managing the records of ride requests and offers, keeping track of the vehicle used, the passengers taking the ride, among other pieces of information. To provide driver pattern recognition functionalities, **Driver Pattern Application** implements a set of methods that analyzes incoming vehicle data in order to recognize intended or unintended driving patterns. All of the previously mentioned modules read from and write to the **Thumbeo Database** which is responsible for storing the information recorded by the respective modules.

6.2 Architecture Overview and Decisions

When designing Thumbeo Corporate's architecture, some decisions had to be made regarding the structure of the architecture, while taking into consideration the objectives specified in the Chapter 1 and the quality attributes specified in Chapter 5.

The first hypothesis posed was that of the overall structure of the architecture; whether it would follow a Microservices, a Monolithic pattern or a Layered pattern. All the patterns will be explained next, in order to present the advantages and disadvantages inherently tied to Microservices, Monolithic or Layered patterns.

Monolithic Architecture Pattern The Monolithic Pattern considers the system as a self-contained single component. The philosophy behind this pattern is such that the application itself is responsible for every step necessary to complete a specific function. This pattern in particular lacks the application of the concept of modularity, where parts of the application logic are reused in order to ease development and maintenance. Nevertheless, this architectural pattern eases the development and deployment phases, given that the application consists of a single component, although it adds complexity in the testing phase, and makes the system harder to scale.

Microservices Architecture Pattern The Microservices Pattern[34] applies the notion of *separately deployed units* when organizing the components of the systems. These components, denoted as *service components* can vary in granularity, providing one or more modules, depending on the functions that the service component should perform. The main challenge of the Microservices Pattern is to find the right level of granularity that each service component should have; given that coarse-grained components will lead a bloated architecture that might lose the benefits proposed by the Microservices Pattern; and fine-grained components will lead orchestration problems that cause an increase in complexity, confusion and expense.

Layered Architecture Pattern The Layered Architecture Pattern[35] organizes the architecture constituents as horizontal layers with each layer having a specific role within the application itself. In general, most layered architectures present four standard layers:

presentation, **business**, **persistence** and **database**; nevertheless, the architectural pattern does not specify the number of layers that must exist and neither does it specify what are the roles and responsibilities of the layers. One of the key concepts present in layered architectures is the notion of *layers of isolation*, in which each layer is independent of the other layers and therefore have little to no knowledge of the inner workings of the other layers of the architecture.

Having the concepts of all architectural patterns in mind, it was decided that Thumbeo Corporate's architecture will follow a Layered Architectural pattern. This pattern allows for a clear division between the different layers of the system, hereby defined as the Messaging Layer, the Harmonization Layer, the Business Layer and the Data Layer. Another crucial idea behind the choice of a Layered pattern is the isolation between all layers, which allows for a separation of the concerns that the system will uphold. Also, in theory, updating the system to support a new external system requires only the development of a new interface in the Harmonization Layer and in the Business Layer.

6.3 Technology Overview and Decisions

Deciding the technologies that will be used to build Thumbeo Corporate is the next step of this chapter. In order to provide some deciding factors for the different technologies available some restrictions are imposed, either directly by Ubiwhere or by the stakeholders, or indirectly by the attributes the system should possess. These restriction are as follows:

- The technologies must be used in production-grade, scalable software systems.
- The technologies/frameworks used must be open-source.
- The chosen technologies/frameworks must have good community support and documentation.
- The time to learn each technology and implement the product must not exceed the existing delivery deadlines.

Having the previous restrictions in mind the choice for the technology to be employed was straightforward, the system will be implemented using the Python web framework, **Django**. This choice of technology meets all the restrictions imposed given that Ubiwhere already built some components present in Thumbeo Corporate using Django; which also indicates that the company already knows the technology, and discussions with other developers at Ubiwhere proved that the company is fluent in Django. The last restriction did not pose an issue, given that the developer of this internship is already well versed with the technology itself and any new challenges found could be discussed with the experienced team at Ubiwhere.

The choice of a Relational Database Management System (RDBMS) was also as straightforward as the choice of a web framework, given that Ubiwhere and the developer of this internship already had experience with the technology elected, **PostgreSQL**. This RDBMS also met all the open-source restriction posed initially, along with the community support and the existing documentation for the database itself and for its integration with Django. Given that there is a need to implement a message queue system, a technology that provides such features had to be chosen. **RabbitMQ** was elected as the ideal choice for its presence in Ubiwhere's knowledge-base, for the fact that it employs an open-source methodology and for its extensive documentation. Finally, the technology employed to orchestrate all of the services together is **Docker**; which was again chosen based on Ubiwhere's tech stack and the knowledge-base created around the technology.

Django: Defined as a high-level Python web framework that encourages rapid development, Django[36] has many of the tools needed to develop the product of this internship. Offering an abstraction on the REST pattern with the package *django-rest-framework*[37] and an imbued Object-Relational Mapping (ORM) that supports geographic queries when extended with *PostGIS*, along with many other open-source extensions developed by the community. The documentation provided by *Django* and by django-rest-framework is also more than sufficient to help with the work that must be produced.

PostgreSQL: Having over thirty years of open-source development, PostgreSQL[38] is considered to be a robust, reliable and performant RDBMS. It offers an official documentation, although it is not expected that the developer of this internship would use raw SQL queries, preferring instead to use of Django- ORM along with the geospatial extension *PostGIS*. PostgreSQL also offers the possibility of saving JSON objects or Array types directly to the database, which might become advantageous in case this functionality ever becomes needed.

Docker: The modern day technology used to orchestrate microservices involves utilizing images with the intended technologies configured, to create isolated containers that will communicate with one another through network, that can either internal or external depending on the wants and needs of the product. These containers are sometimes replicated at will in order to build elastic system or replicate existing system in another context. Docker[39] and it's extensions, *docker-compose*[40] and *docker-swarm*[41] are used to create containers using the images, orchestrate those containers together, and replicate those containers when needed, respectively.

RabbitMQ: This lightweight open source message broker is deemed as "easy to deploy on premises and in the cloud"[42]. RabbitMQ offers a multitude of features such as asynchronous messaging, compatibility with a wide range of programming languages and distributed deployment using clusters or the federation paradigm. The open-source community developing RabbitMQ also extended some functionalities [43] through a *plugin* system. For the purpose of this architecture, the most important feature provided by this technology is, without a doubt, the asynchronous messaging functionality, that will allow for a set of subsystems to gather and process data in a performant manner.

6.4 System Architecture

Having explained the hypothesis posed for the architecture and the decisions made, along with the having chosen the technologies that will be employed in the development of this internship, this section will show the architecture constructed and how it will integrate with the already existing components of Thumbeo Corporate.

The C4 Model was the elected method for visually representing the architecture, as it was already well known by the Ubiwhere's collaborators, meaning it would not pose an entry barrier for those who must grasp the architecture to critique it or implement it. This model also projects a natural order onto the architecture, first describing the *System Context* where a more high-level approach is taken, illustrating the users and the systems that will interact with Thumbeo Corporate. Diving deeper into the representation of the system presented in the *System Context* diagram, the *Containers* diagram aims to break down the high-level vision of the system into the separate entities that constitute deployable units that execute code or store data. Finally, the last diagram, the *Components* diagram, zooms in to the specific containers presented in the previous diagram, to provide a lower level view of architecture to implement in the context of this internship.

The following diagram, *System Context* will present a high-level view of the different actors that will interact with Thumbeo Corporate.

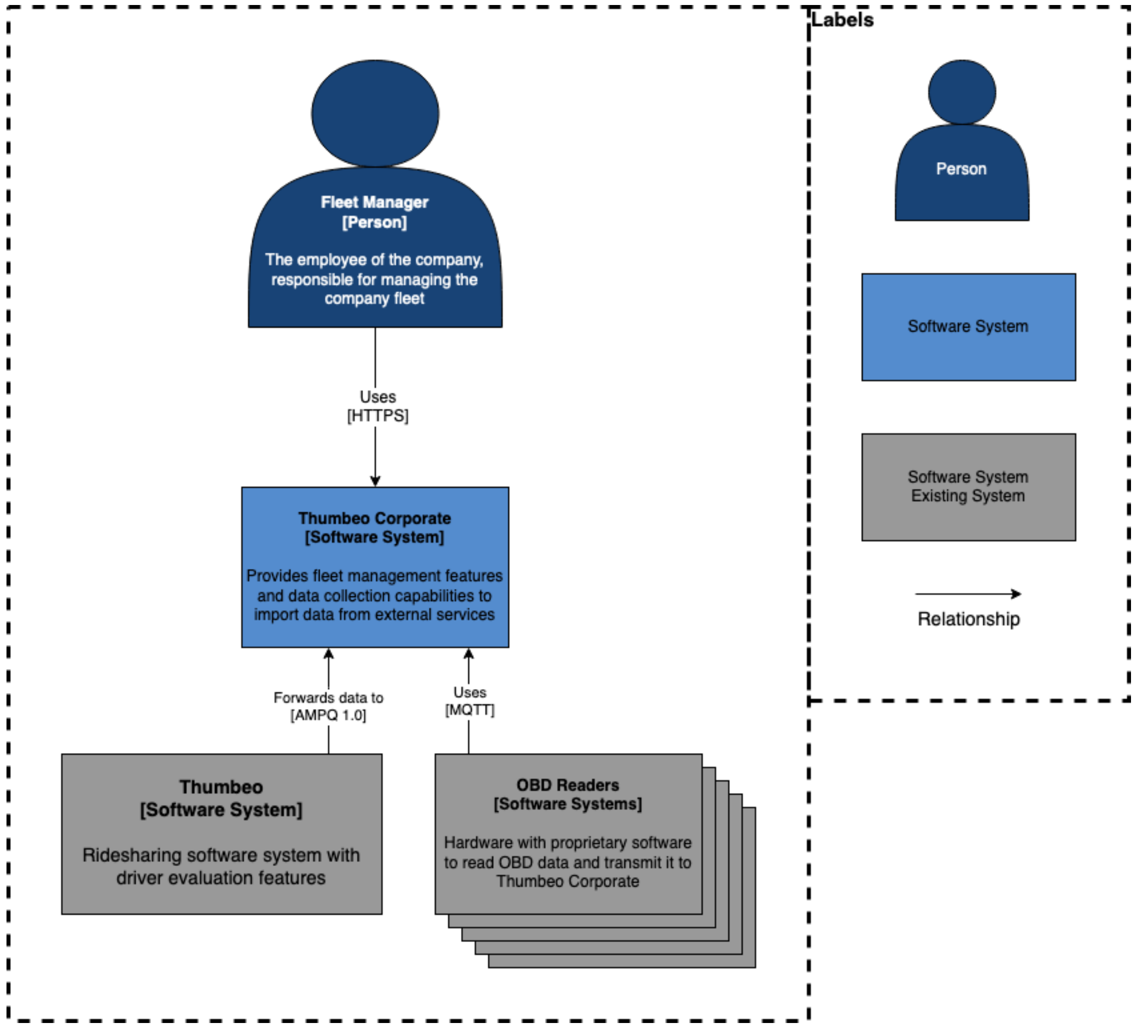

Labels
Fleet Manager
[Person]
The employee of the company, responsible for managing the company fleet
Uses
[HTTPS]
Thumbeo Corporate
[Software System]
Provides fleet management features and data collection capabilities to import data from external services
Forwards data to
[AMPQ 1.0]
Uses
[MQTT]
Thumbeo
[Software System]
Ridesharing software system with driver evaluation features
OBD Readers
[Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate
Person
Software System
Software System
Existing System
Relationship

Figure 6.2: C4 Model — System Context

The previous diagram shows both types of Software Systems that will interact with Thumbeo Corporate: the Thumbeo system is an external system that will supply data related to previously registered vehicles, users and rides; and the OBD Readers present in every vehicle that will supply data to Thumbeo Corporate. The figure also shows that the type of user that will interact with the system is the Fleet Manager.

The following *Containers* diagram visually demonstrates how the different constituents of Thumbeo Corporate are organized in order to provide the necessary functions to the product as a whole.

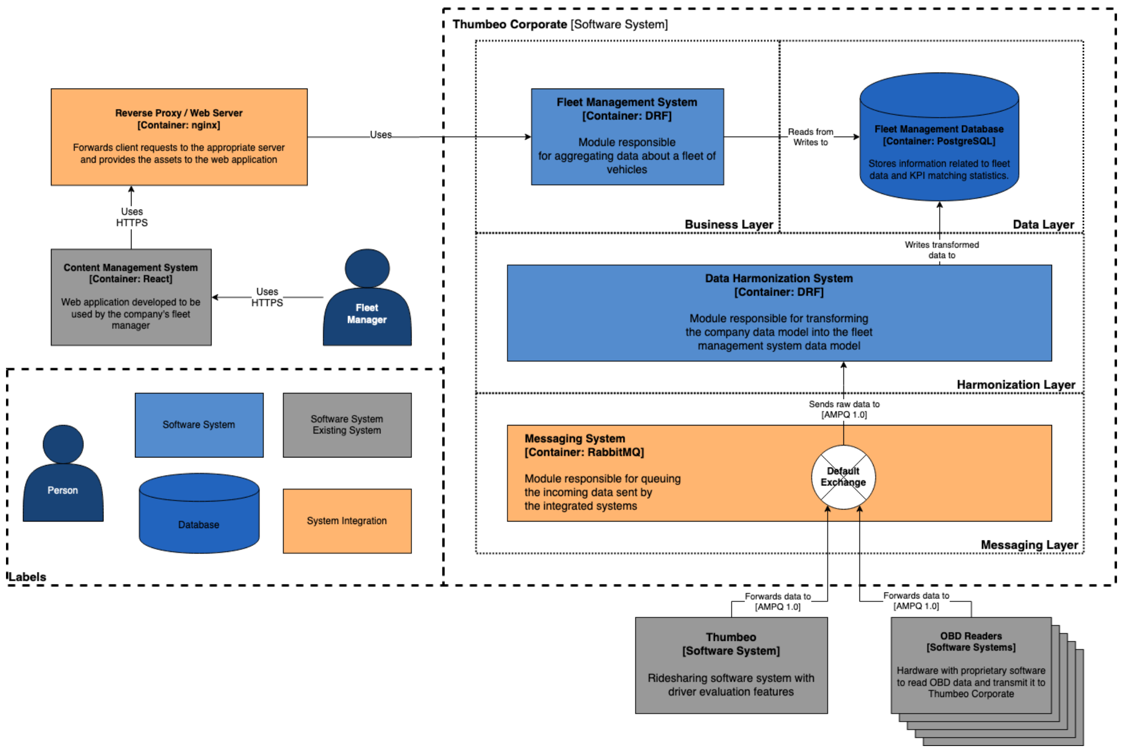

Thumbeo Corporate [Software System]
Reverse Proxy / Web Server
[Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses
Fleet Management System
[Container: DRF]
Module responsible for aggregating data about a fleet of vehicles
Reads from
Writes to
Fleet Management Database
[Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Business Layer
Data Layer
Uses
HTTPS
Content Management System
[Container: React]
Web application developed to be used by the company's fleet manager
Uses
HTTPS
Fleet Manager
Writes transformed data to
Data Harmonization System
[Container: DRF]
Module responsible for transforming the company data model into the fleet management system data model
Harmonization Layer
Sends raw data to
[AMPQ 1.0]
Messaging System
[Container: RabbitMQ]
Module responsible for queuing the incoming data sent by the integrated systems
Default Exchange
Messaging Layer
Person
Software System
Software System Existing System
Database
System Integration
Labels
Forwards data to
[AMPQ 1.0]
Forwards data to
[AMPQ 1.0]
Thumbeo
[Software System]
Ridesharing software system with driver evaluation features
OBD Readers
[Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate

Figure 6.3: C4 Model — Containers

Analyzing the previous diagram it is clear that the architecture is composed of a variety of different containers each with their respective responsibilities:

Entry Point The client-side system that acts as the user centered entry point is **Content Management System** (**CMS**), developed with *ReactJS*, and it provides the user interface that will forward the user requests to their respective system. This system will allow **fleet managers** to create, view, update or delete information regarding the company vehicles, as well as assign vehicles to drivers and visualize the data generated by their fleet. Although this system is being mentioned, it resides outside of the scope of this internship.

Messaging System This system resides within the Messaging Layer and it serves the purpose of receiving messages with data from the integrated external systems and forwards the messages into the Data Harmonization System. The existence of a messaging system is justified by the unknowable quantity of external systems that fleet managers will want to integrate with Thumbeo Corporate, and as such this system will enable multiple systems to be integrated without heavily compromising the system performance.

Data Harmonization System The purpose of this system it to transform the data received from the integrated external systems; it functions by using serializing methods to transform said data into the data structure defined by Ubiwhere. The serializing methods used are provided by *django-rest-framework* and use *Django* models to convert the data into Python objects. After the harmonization is performed, the data created is then written into the Data Layer in order to be accessible by the Fleet Management System.

Fleet Management System This system implements all the features related to fleet management, being the registration of users, vehicles, rides, the submission of reports and notifications to fleet managers. All of the data manipulated by this system is written into the Data Layer to the same database as the Data Harmonization system, this enables this system to read the data imported from external systems.

Before presenting the next level diagram, the *Components diagram*, it is important to represent the information flow of two different situations pertaining to this architecture. The first situation is that of importing data into the *Fleet Management System*.

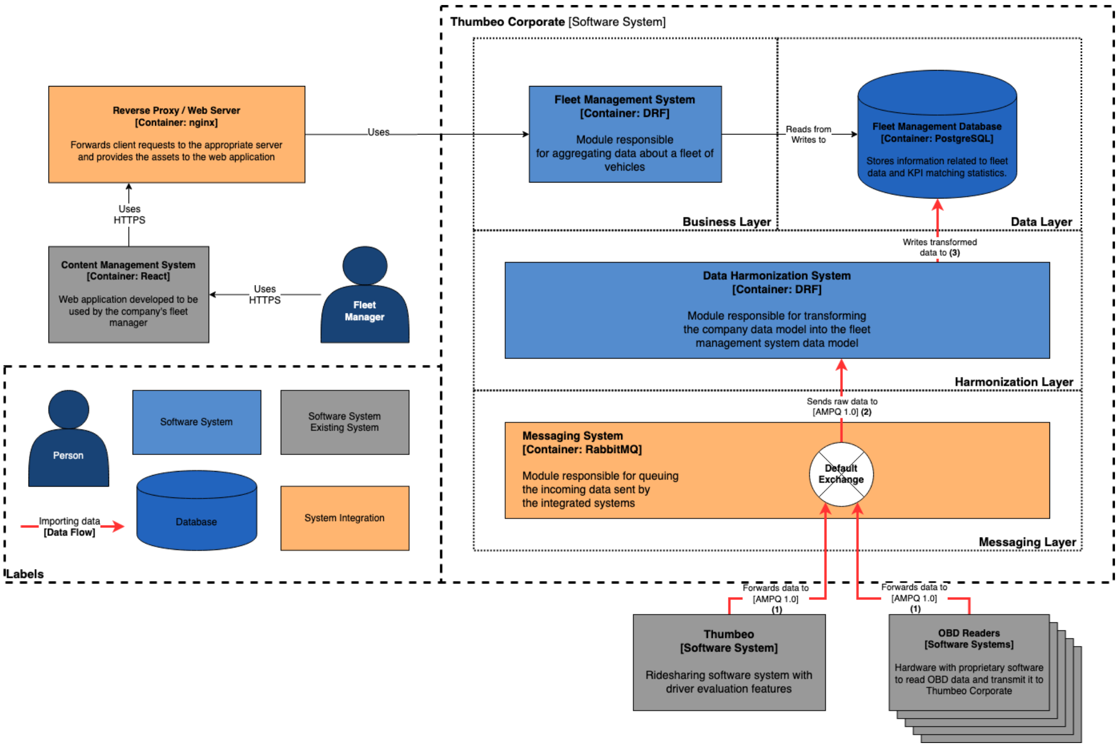

Thumbeo Corporate [Software System]
Reverse Proxy / Web Server
[Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses
Fleet Management System
[Container: DRF]
Module responsible for aggregating data about a fleet of vehicles
Reads from
Writes to
Fleet Management Database
[Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Business Layer
Data Layer
Uses
HTTPS
Content Management System
[Container: React]
Web application developed to be used by the company's fleet manager
Uses
HTTPS
Fleet Manager
Writes transformed data to (3)
Data Harmonization System
[Container: DRF]
Module responsible for transforming the company data model into the fleet management system data model
Harmonization Layer
Sends raw data to
[AMPQ 1.0] (2)
Messaging System
[Container: RabbitMQ]
Module responsible for queuing the incoming data sent by the integrated systems
Default Exchange
Messaging Layer
Person
Software System
Software System
Existing System
Importing data
[Data Flow]
Database
System Integration
Labels
Forwards data to
[AMPQ 1.0]
(1)
Forwards data to
[AMPQ 1.0]
(1)
Thumbeo
[Software System]
Ridesharing software system with driver evaluation features
OBD Readers
[Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate

Figure 6.4: Information Flow (Containers) — Importing Data

To import data into the *Fleet Management System*, the data is first sent in **(1)** by the external systems into the *Messaging System*, which then sends out the data in **(2)** to the *Data Harmonization System*. In this last mentioned system, the data is transformed from it's original format into the proprietary data structures defined by Ubiwhere. Upon finalizing the harmonization process, the data is then written into *Fleet Management Database* in **(3)**.

The next information flow to present is that of creating new data which up until the last step, takes a completely different route from the previous flow explained.

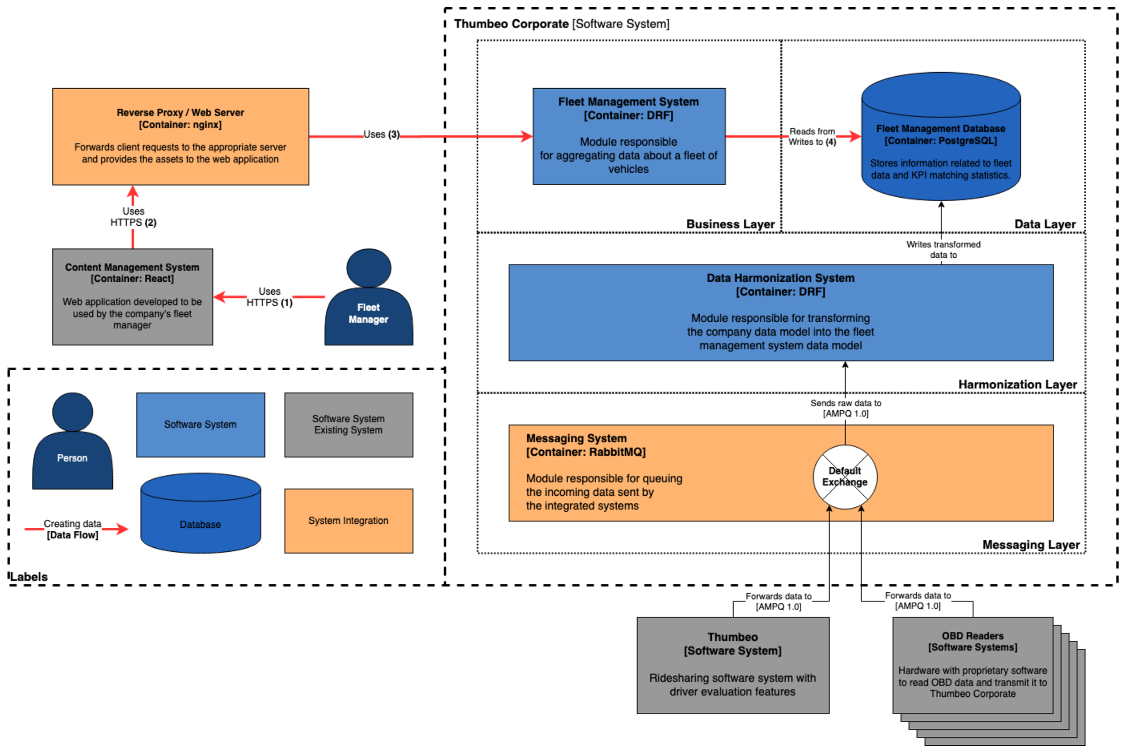

Thumbeo Corporate [Software System]
Reverse Proxy / Web Server
[Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses (3)
Fleet Management System
[Container: DRF]
Module responsible for aggregating data about a fleet of vehicles
Reads from Writes to (4)
Fleet Management Database
[Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Business Layer
Data Layer
Uses HTTPS (2)
Content Management System
[Container: React]
Web application developed to be used by the company's fleet manager
Uses HTTPS (1)
Fleet Manager
Writes transformed data to
Data Harmonization System
[Container: DRF]
Module responsible for transforming the company data model into the fleet management system data model
Harmonization Layer
Sends raw data to [AMPQ 1.0]
Person
Software System
Software System Existing System
Messaging System
[Container: RabbitMQ]
Module responsible for queuing the incoming data sent by the integrated systems
Default Exchange
Creating data [Data Flow]
Database
System Integration
Messaging Layer
Labels
Forwards data to [AMPQ 1.0]
Forwards data to [AMPQ 1.0]
Thumbeo
[Software System]
Ridesharing software system with driver evaluation features
OBD Readers
[Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate

Figure 6.5: Information Flow (Containers) — Creating Data

The information flow in the process of creating new data starts out by the *Fleet Manager* inputting data **(1)** into the *Content Management System*. This new data is then transmitted by the *CMS* into the *Reverse Proxy/Web Server* in **(2)**, which in turn forwards the request **(3)** into the *Fleet Management System*. Before writing the data **(4)** into the *Fleet Management Database*, the previously mentioned system validates the format of data sent in the request and only if the data is as intended, is it written into the database.

The *Components* diagram, presented next, zooms into the previously shown system as a way to demonstrate how each layer of the architecture is structured and how the information is communicated within the layers themselves and transmitted from one layer onto another. This diagram makes use of technology specific concepts and as such it may not reflect the actual architecture of systems that employ different technologies.

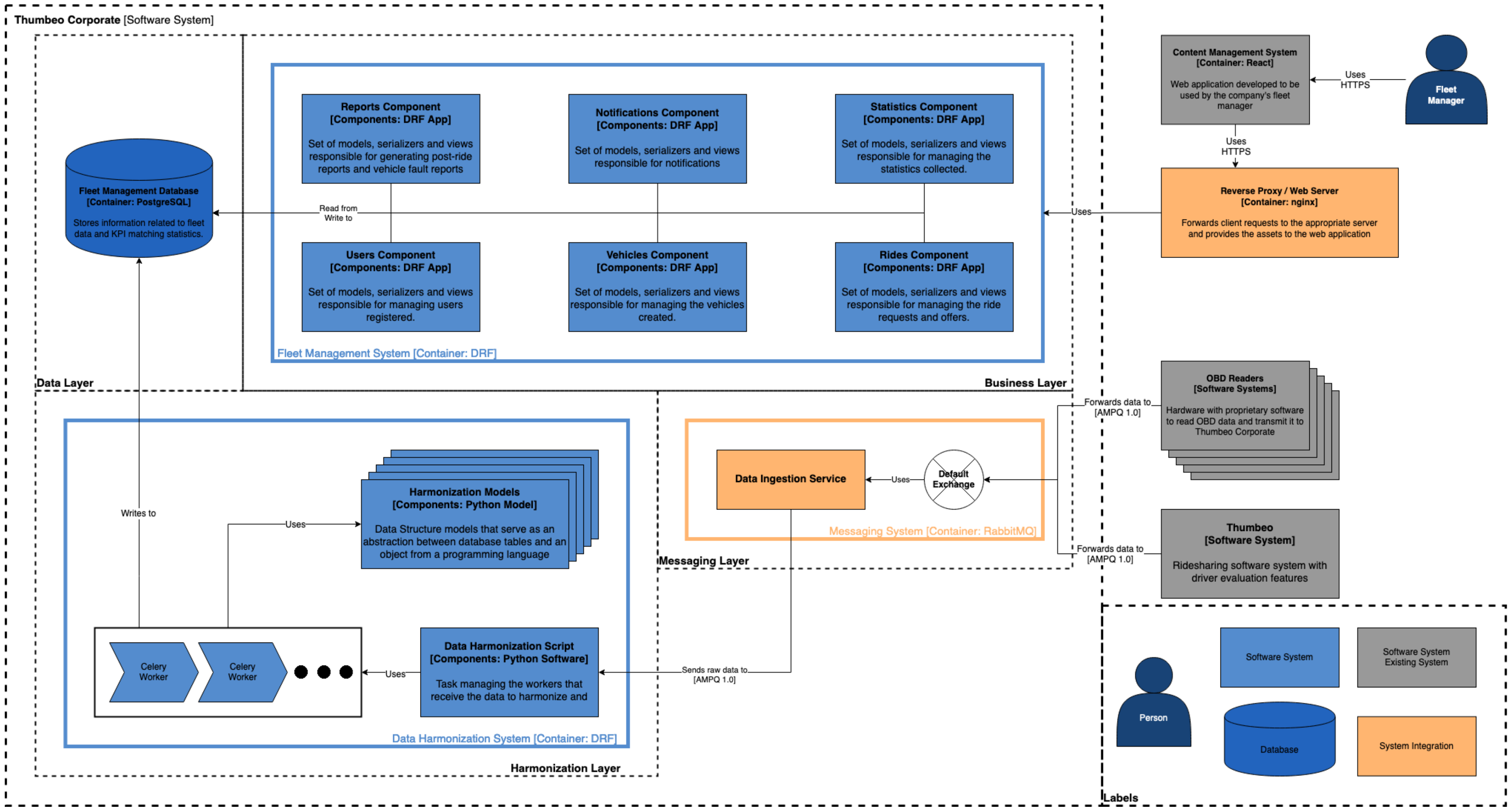

Thumbeo Corporate [Software System]
Reports Component [Components: DRF App]
Set of models, serializers and views responsible for generating post-ride reports and vehicle fault reports
Notifications Component [Components: DRF App]
Set of models, serializers and views responsible for notifications
Statistics Component [Components: DRF App]
Set of models, serializers and views responsible for managing the statistics collected.
Users Component [Components: DRF App]
Set of models, serializers and views responsible for managing users registered.
Vehicles Component [Components: DRF App]
Set of models, serializers and views responsible for managing the vehicles created.
Rides Component [Components: DRF App]
Set of models, serializers and views responsible for managing the ride requests and offers.
Fleet Management System [Container: DRF]
Fleet Management Database [Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Read from
Write to
Data Layer
Business Layer
Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses HTTPS
Fleet Manager
Uses HTTPS
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses
OBD Readers [Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate
Forwards data to [AMPQ 1.0]
Thumbeo [Software System]
Ridesharing software system with driver evaluation features
Forwards data to [AMPQ 1.0]
Data Ingestion Service
Uses
Default Exchange
Messaging System [Container: RabbitMQ]
Messaging Layer
Writes to
Uses
Harmonization Models [Components: Python Model]
Data Structure models that serve as an abstraction between database tables and an object from a programming language
Celery Worker
Celery Worker
Uses
Data Harmonization Script [Components: Python Software]
Task managing the workers that receive the data to harmonize and
Sends raw data to [AMPQ 1.0]
Data Harmonization System [Container: DRF]
Harmonization Layer
Person
Software System
Software System Existing System
Database
System Integration
Labels

Figure 6.6: C4 Model — Components

Components

- **Fleet Management System** — All of the following components encompassed by this system are *django-rest-framework* applications, that contain the models, the corresponding serializers and views responsible for managing the respective resource.
  - **Users Component** — This component is responsible for managing information pertaining to users, be it authentication credentials or profile data.
  - **Vehicles Component** — This component manages the information related to the vehicles registered in the platform.
  - **Notifications Component** — This component handles the notification process, sending notifications to users and allowing them to change their notification status.
  - **Reports Component** — This component is responsible for handling the reporting features of the system; allowing users to create report about rides or vehicle faults.
  - **Statistics Component** — This component handles the gathering of information pertaining to the KPIs specified in Chapter 2.
- **Data Harmonization System**
  - **Harmonization Models** — These models serve as the data structures used to transform the imported data into the intended format to be written into the *Data Layer*.
  - **Celery Workers** — The workers provided by the *Celery* package serve as the asynchronous runners that will execute the data harmonization tasks established for the respective data structure.
  - **Data Harmonization Script** — This script acts as the interface between the *Messaging System* and the *Data Harmonization System*; creating the **Default Exchange** in the system to which the external systems will send the information.
- **Messaging System**
  - **Default Exchange** — Pre-declared direct exchange with no name, that when used will deliver messages to the queue equal to the routing key of the message.
  - **Data Ingestion Service** — This service acts as queue for the messages that intend to be transmitted to a specific consumer.

Having explained the roles of each component present in the system, the next step is that of demonstrating how the information flows within the system, alike what was done with the previous *Containers Diagram*. The two following images pertain respectively the information flow when a fleet manager wants to import data and when it wants to create new data.

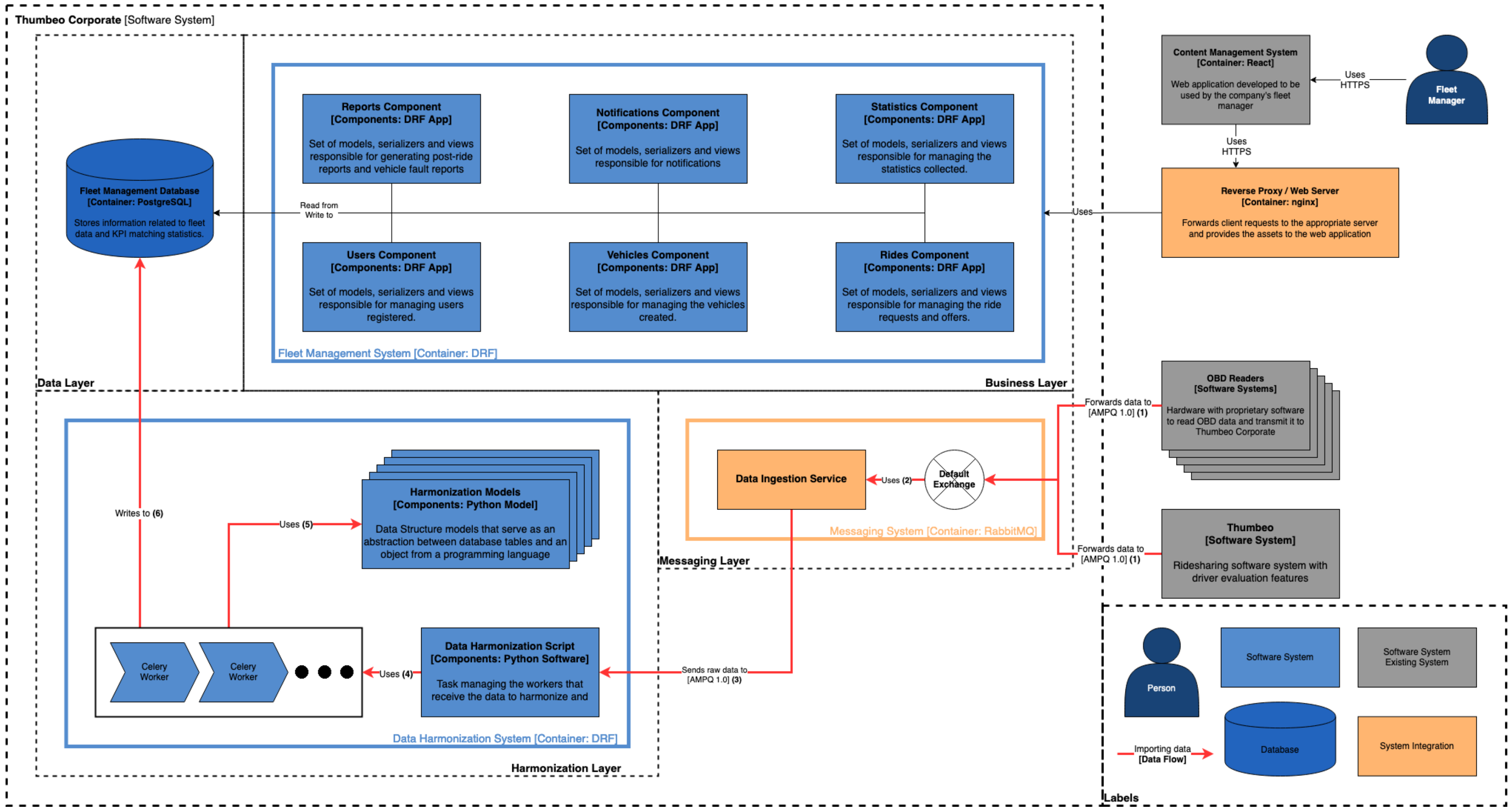
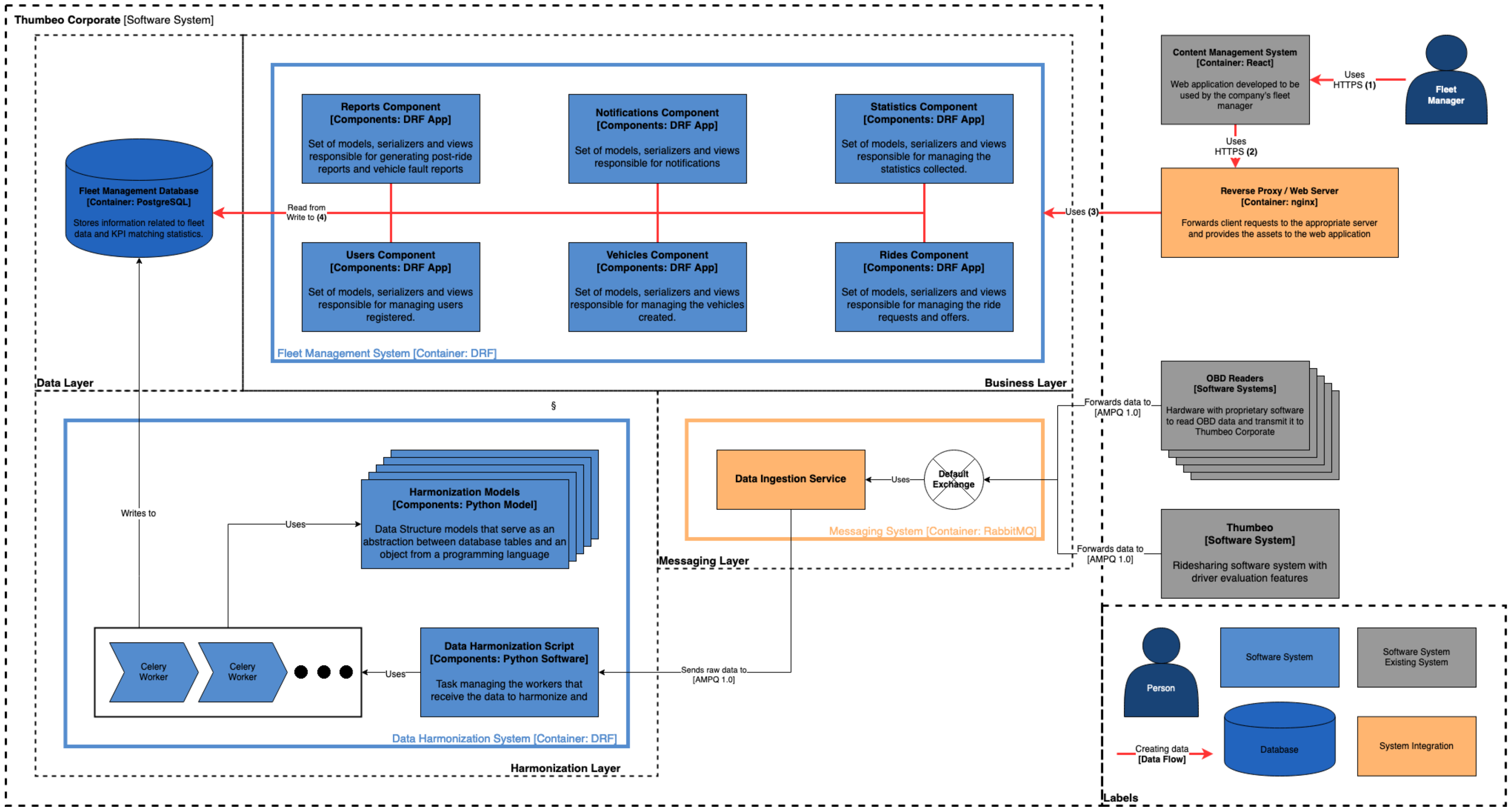

Thumbeo Corporate [Software System]
Fleet Management Database [Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Read from
Write to
Reports Component [Components: DRF App]
Set of models, serializers and views responsible for generating post-ride reports and vehicle fault reports
Notifications Component [Components: DRF App]
Set of models, serializers and views responsible for notifications
Statistics Component [Components: DRF App]
Set of models, serializers and views responsible for managing the statistics collected.
Users Component [Components: DRF App]
Set of models, serializers and views responsible for managing users registered.
Vehicles Component [Components: DRF App]
Set of models, serializers and views responsible for managing the vehicles created.
Rides Component [Components: DRF App]
Set of models, serializers and views responsible for managing the ride requests and offers.
Fleet Management System [Container: DRF]
Data Layer
Business Layer
Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses HTTPS
Fleet Manager
Uses HTTPS
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses
OBD Readers [Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate
Forwards data to [AMPQ 1.0] (1)
Thumbeo [Software System]
Ridesharing software system with driver evaluation features
Forwards data to [AMPQ 1.0] (1)
Data Ingestion Service
Uses (2)
Default Exchange
Messaging System [Container: RabbitMQ]
Messaging Layer
Writes to (6)
Uses (5)
Harmonization Models [Components: Python Model]
Data Structure models that serve as an abstraction between database tables and an object from a programming language
Celery Worker
Celery Worker
Uses (4)
Data Harmonization Script [Components: Python Software]
Task managing the workers that receive the data to harmonize and
Sends raw data to [AMPQ 1.0] (3)
Data Harmonization System [Container: DRF]
Harmonization Layer
Person
Software System
Software System Existing System
Database
System Integration
Importing data [Data Flow]
Labels

Figure 6.7: Information Flow (Containers) — Importing Data

In the previous figure 6.7, analogous to the containers diagram, the information flow of importing data starts when an external systems sends data to the *Messaging System* in **(1)**, more precisely this time, to the *Default Exchange*. This exchange then sends the transmitted data into the *Data Ingestion Service* in **(2)** that will queue the data to be consumed by the *Data Harmonization System* in **(3)**. This script will make use **(4)** of *Celery Workers* to start asynchronous tasks that transform the data imported into the their respective data structure **(5)** specified by the *Harmonization Models*. After the harmonization process is concluded, the *Celery Workers* will write **(6)** the new data into the *Fleet Management Database*.

The following figure 6.8, again analogous to the containers diagram, the information flow of creating data starts by the *Fleet Manager* inputting data **(1)** into the *Content Management System*. The *CMS* then sends the data **(2)** into the *Reverse Proxy/Web Server* that will transmit the data **(3)** into the corresponding component present into the *Fleet Management System*. Finally, any of the components in the *Fleet Management System* will write the data **(4)** inputed by the *Fleet Manager* into the *Fleet Management Database*.



Thumbeo Corporate [Software System]
Fleet Management Database [Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Read from
Write to (4)
Reports Component [Components: DRF App]
Set of models, serializers and views responsible for generating post-ride reports and vehicle fault reports
Notifications Component [Components: DRF App]
Set of models, serializers and views responsible for notifications
Statistics Component [Components: DRF App]
Set of models, serializers and views responsible for managing the statistics collected.
Users Component [Components: DRF App]
Set of models, serializers and views responsible for managing users registered.
Vehicles Component [Components: DRF App]
Set of models, serializers and views responsible for managing the vehicles created.
Rides Component [Components: DRF App]
Set of models, serializers and views responsible for managing the ride requests and offers.
Fleet Management System [Container: DRF]
Data Layer
Business Layer
Uses (3)
Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses HTTPS (1)
Fleet Manager
Uses HTTPS (2)
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
§
Writes to
Uses
Harmonization Models [Components: Python Model]
Data Structure models that serve as an abstraction between database tables and an object from a programming language
Celery Worker
Celery Worker
Uses
Data Harmonization Script [Components: Python Software]
Task managing the workers that receive the data to harmonize and
Data Harmonization System [Container: DRF]
Harmonization Layer
Sends raw data to [AMPQ 1.0]
Data Ingestion Service
Uses
Default Exchange
Messaging System [Container: RabbitMQ]
Messaging Layer
Forwards data to [AMPQ 1.0]
OBD Readers [Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate
Forwards data to [AMPQ 1.0]
Thumbeo [Software System]
Ridesharing software system with driver evaluation features
Person
Software System
Software System Existing System
Creating data [Data Flow]
Database
System Integration
Labels

Figure 6.8: Information Flow (Containers) — Creating Data

Lastly, to fully explain the architecture is necessary to present how the process of harmonizing data occurs within the system. When the *Data Harmonization System* receives data coming from the *Messaging System*, it executes a *Celery Worker*. This worker will use the *Harmonization Models* specified to convert the incoming data into objects that can the be inserted into the database, given that these models will be specified using Python Classes, the objects created are Python Objects. The following diagram is an overview of what models exist and in what circumstances are these models used.

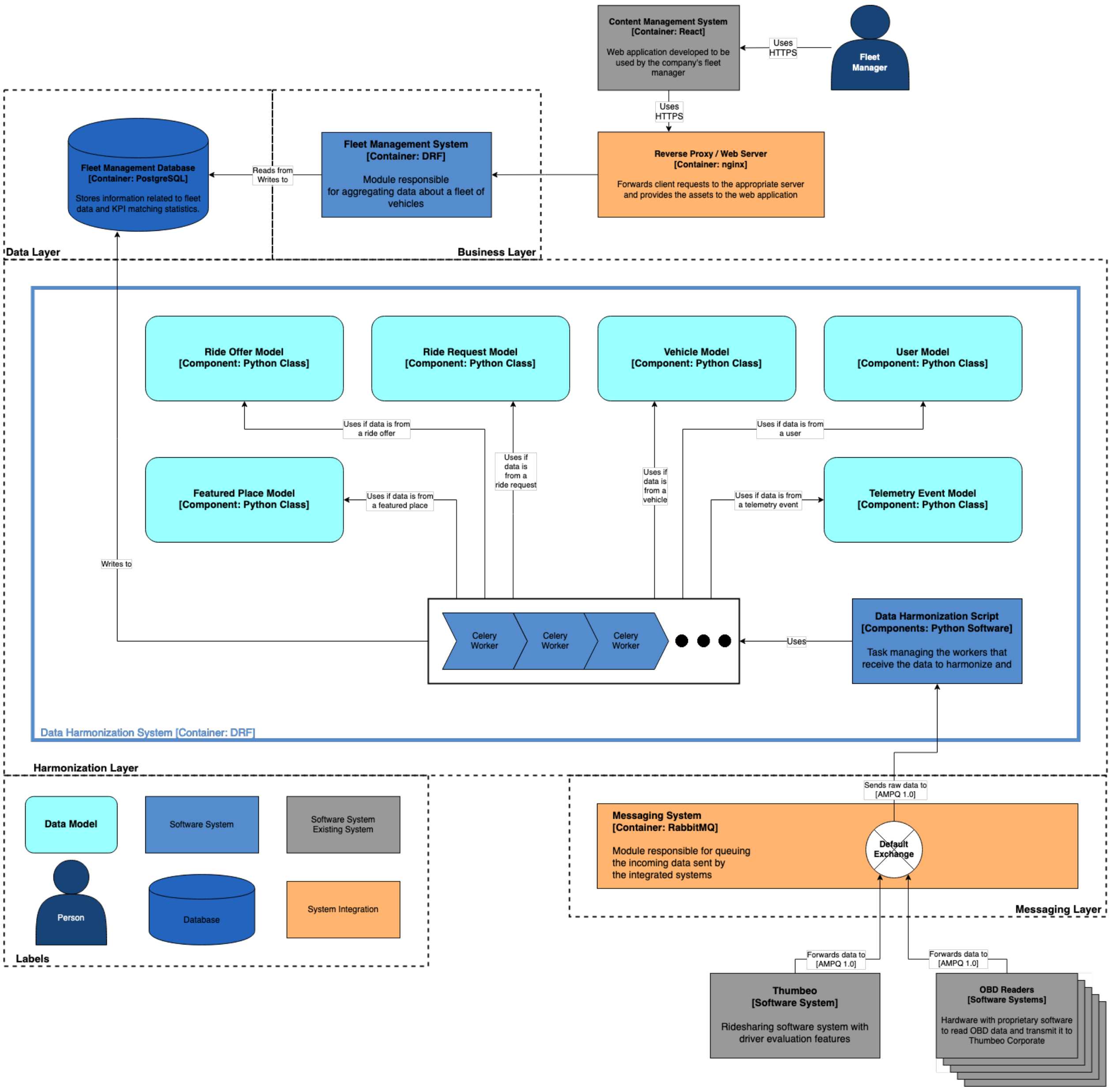

Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses HTTPS
Fleet Manager
Uses HTTPS
Fleet Management Database [Container: PostgreSQL]
Stores information related to fleet data and KPI matching statistics.
Reads from Writes to
Fleet Management System [Container: DRF]
Module responsible for aggregating data about a fleet of vehicles
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Data Layer
Business Layer
Ride Offer Model [Component: Python Class]
Ride Request Model [Component: Python Class]
Vehicle Model [Component: Python Class]
User Model [Component: Python Class]
Uses if data is from a ride offer
Uses if data is from a ride request
Uses if data is from a vehicle
Uses if data is from a user
Featured Place Model [Component: Python Class]
Uses if data is from a featured place
Uses if data is from a telemetry event
Telemetry Event Model [Component: Python Class]
Writes to
Celery Worker
Celery Worker
Celery Worker
Uses
Data Harmonization Script [Components: Python Software]
Task managing the workers that receive the data to harmonize and
Data Harmonization System [Container: DRF]
Harmonization Layer
Data Model
Software System
Software System Existing System
Person
Database
System Integration
Labels
Sends raw data to [AMPQ 1.0]
Messaging System [Container: RabbitMQ]
Module responsible for queuing the incoming data sent by the integrated systems
Default Exchange
Messaging Layer
Forwards data to [AMPQ 1.0]
Forwards data to [AMPQ 1.0]
Thumbeo [Software System]
Ridesharing software system with driver evaluation features
OBD Readers [Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate

Figure 6.9: Data Harmonization System

As seen in the previous diagram 6.9, the *Data Harmonization Script* recognizes the type of data it is importing, and uses the corresponding data model to transform the original data structured into a Python Object that is then written into the *Fleet Management Database*. It is necessary to clarify that the previous diagram is specific for this internship only and it is not representative of what can be achieved by the *Data Harmonization System* in the case that the external systems where different than those specified.

6.5 Architecture Representation Phases

This section aims to help maintain a cohesive narrative of the design of the architecture of the project, showing the different architecture diagrams that lead to the final diagrams shown previously. For the purposes of a establishing a timeline, this section is divided between the first semester, prior to the intermediate presentation, and the second semester, after the intermediate presentation.

6.5.1 First Semester

In the first semester, the basis for the architectural pattern was a microservices architecture. This choice was made because the perspective of the project was different at the time; given the knowledge about the Thumbeo Corporate product as a whole and about Ubiwhere's objectives for the Fleet Management topic, the decision of choosing a microservices pattern was indeed the correct one. The initial idea was to develop Thumbeo Corporate, to be integrated with other Ubiwhere products that implement other fleet related functionalities. To achieve this idea, the remaining products and the fleet management system, that is the goal of this work, where defined as plugins, that could be toggled on or off based on customers decision.

In hindsight, this decision was not necessarily wrong for the product as a whole, but it was not correct approach for the work of this internship, and the objective of developing a data ingestion hub to collected data from external systems along with a system to provide fleet management functionalities was not correctly expressed in the diagram presented at the time.

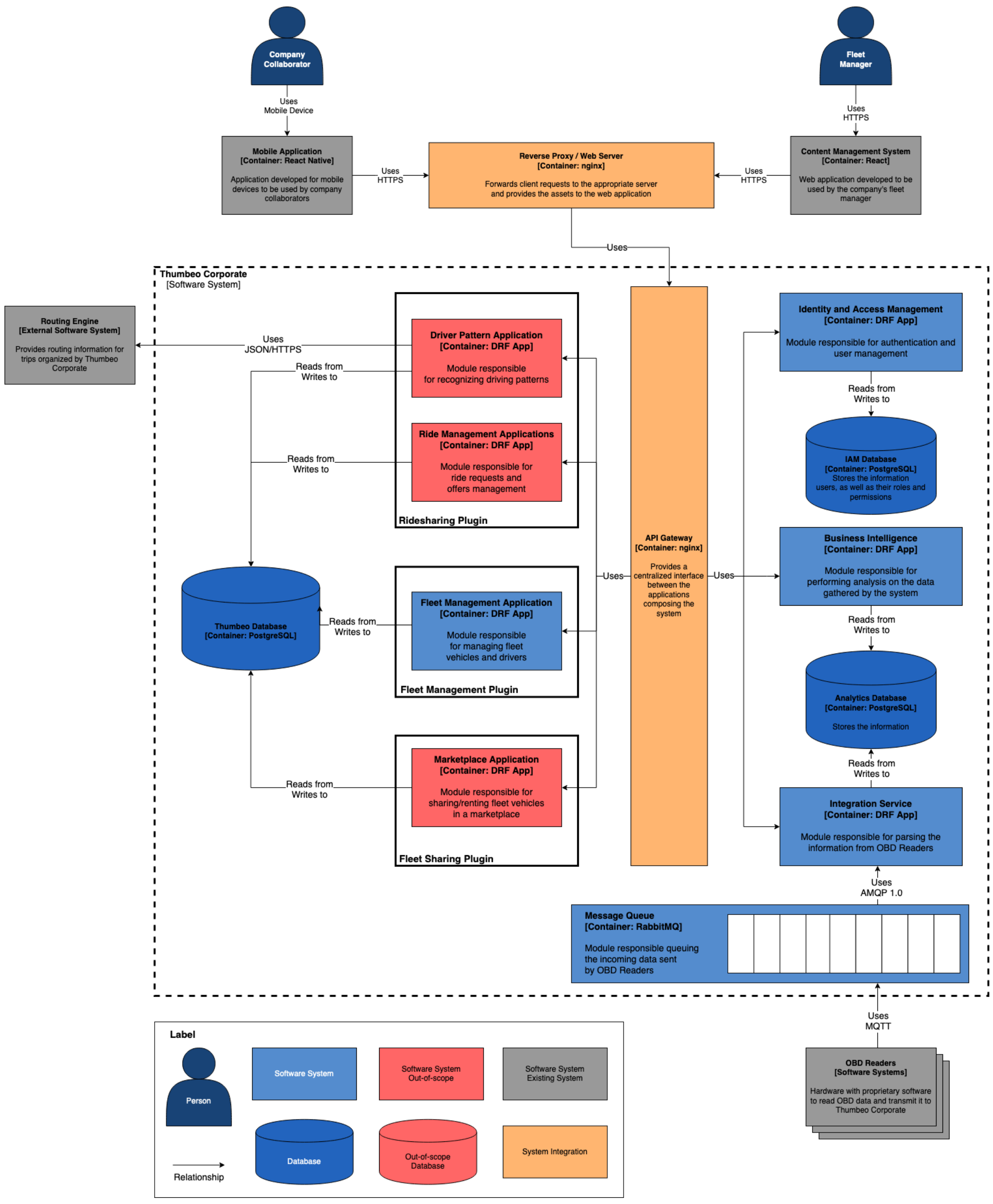

Company Collaborator
Fleet Manager
Uses Mobile Device
Uses HTTPS
Mobile Application [Container: React Native]
Application developed for mobile devices to be used by company collaborators
Uses HTTPS
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses HTTPS
Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses
Thumbeo Corporate [Software System]
Routing Engine [External Software System]
Provides routing information for trips organized by Thumbeo Corporate
Uses JSON/HTTPS
Reads from Writes to
Driver Pattern Application [Container: DRF App]
Module responsible for recognizing driving patterns
Ride Management Applications [Container: DRF App]
Module responsible for ride requests and offers management
Ridesharing Plugin
Thumbeo Database [Container: PostgreSQL]
Fleet Management Application [Container: DRF App]
Module responsible for managing fleet vehicles and drivers
Fleet Management Plugin
Marketplace Application [Container: DRF App]
Module responsible for sharing/renting fleet vehicles in a marketplace
Fleet Sharing Plugin
API Gateway [Container: nginx]
Provides a centralized interface between the applications composing the system
Uses
Identity and Access Management [Container: DRF App]
Module responsible for authentication and user management
IAM Database [Container: PostgreSQL]
Stores the information users, as well as their roles and permissions
Business Intelligence [Container: DRF App]
Module responsible for performing analysis on the data gathered by the system
Analytics Database [Container: PostgreSQL]
Stores the information
Integration Service [Container: DRF App]
Module responsible for parsing the information from OBD Readers
Uses AMQP 1.0
Message Queue [Container: RabbitMQ]
Module responsible queuing the incoming data sent by OBD Readers
Uses MQTT
OBD Readers [Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate
Label
Person
Software System
Software System Out-of-scope
Software System Existing System
Database
Out-of-scope Database
System Integration
Relationship

Figure 6.10: First Semester — Containers

6.5.2 Second Semester

As of the second semester, based on the feedback given in the intermediate presentation, the goal shifted to create an architecture more aligned with a standalone Fleet Management System, that is also able to import data from external systems.

The first diagram shows the two external systems that will be integrated by the end of this work, along with the relationships between the Data Harmonization System, the Fleet Management System, and the Statistics application.

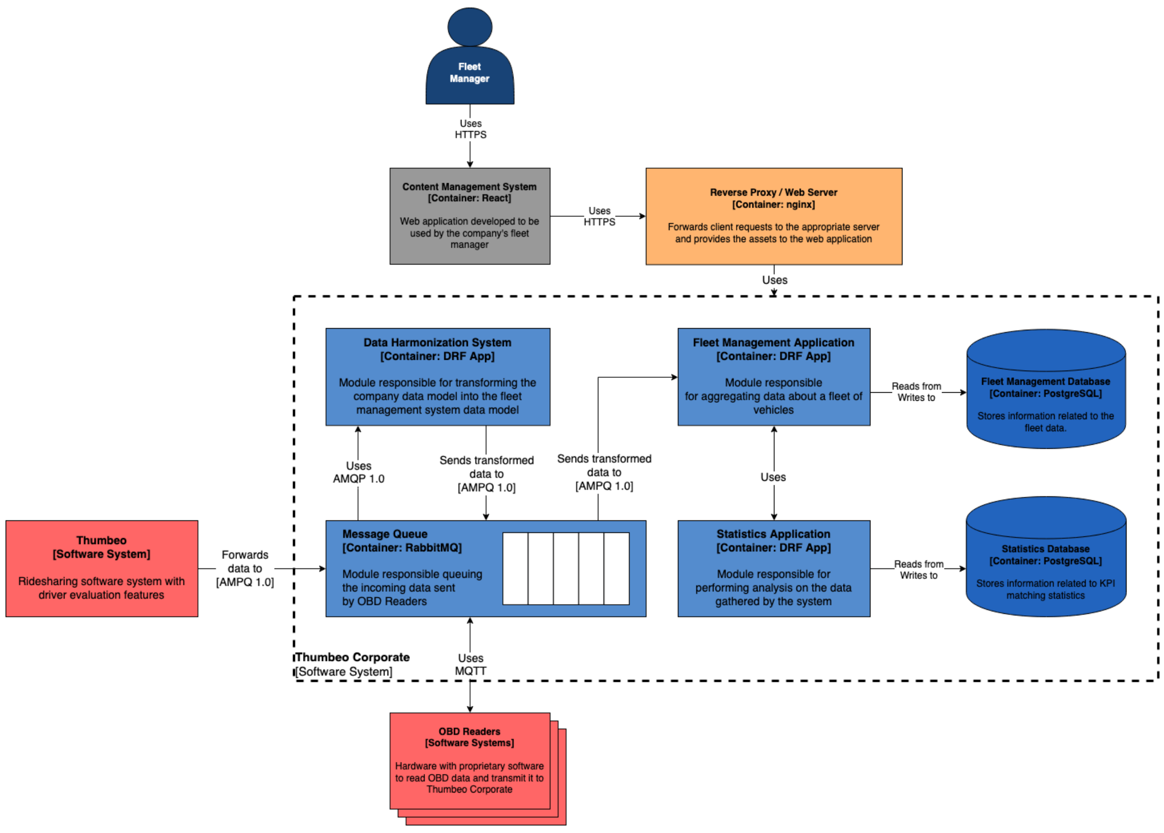

Fleet Manager
Uses HTTPS
Content Management System [Container: React]
Web application developed to be used by the company's fleet manager
Uses HTTPS
Reverse Proxy / Web Server [Container: nginx]
Forwards client requests to the appropriate server and provides the assets to the web application
Uses
Data Harmonization System [Container: DRF App]
Module responsible for transforming the company data model into the fleet management system data model
Fleet Management Application [Container: DRF App]
Module responsible for aggregating data about a fleet of vehicles
Reads from Writes to
Fleet Management Database [Container: PostgreSQL]
Stores information related to the fleet data.
Uses AMQP 1.0
Sends transformed data to [AMPQ 1.0]
Sends transformed data to [AMPQ 1.0]
Uses
Thumbeo [Software System]
Ridesharing software system with driver evaluation features
Forwards data to [AMPQ 1.0]
Message Queue [Container: RabbitMQ]
Module responsible queuing the incoming data sent by OBD Readers
Statistics Application [Container: DRF App]
Module responsible for performing analysis on the data gathered by the system
Reads from Writes to
Statistics Database [Container: PostgreSQL]
Stores information related to KPI matching statistics
Thumbeo Corporate [Software System]
Uses MQTT
OBD Readers [Software Systems]
Hardware with proprietary software to read OBD data and transmit it to Thumbeo Corporate

Figure 6.11: Second Semester — Containers

6.6 Final Considerations

Given that the idea of conceptualizing an architecture to define the representation of the system predicates the necessity of meeting the quality attributes defined in Chapter 5, this section will explain how these attributes were met by the architecture defined.

- **Confidentiality:** The **User Component** is responsible for validating the authorization of a user that intends to access a resource.
- **Data Integrity:** The models established in the **Data Harmonization System** assert the integrity of the data received; if the data does not follow the structure define by Ubiwhere, it will get rejected.
- **Interoperability:** The **Data Harmonization System** is be able to harmonize the data received from the **OBD Readers**, independently of how those systems operate. Obtaining the data and processing it into a usable format by the system is the **Data Harmonization System**'s responsibility.
- **Modifiability:** The structure of the **Data Harmonization System** assures that creating new data structures to import new data requires simply defining the new models and letting the **Data Harmonization Script** know it needs to use the new models to import the data.
- **Scalability:** The system will automatically scale itself when needed with the help of

technologies present in the environment it resides on. Technologies like Docker and the usage of RabbitMQ will assure that the system can deal with peaks of data and can scale itself based on the resources it is using.

Chapter 7

Project Development

This chapter explains the details of the implementation of the previously defined requirements and architecture in chapters 5 and 6. The first aspect covered are the configurations used in the Docker Virtualization system; followed by the configurations in the messaging system implemented using RabbitMQ; next, the implementation of the models, the serializers and the celery workers from the data harmonization system will be shown; finally the fleet management system models and the statistics collection mechanism will be explained.

7.1 Django Concepts

This sections serves to explain some of the main concepts used by the *django-rest-framework* used in the development of the *Fleet Management System*, which are key to understanding the implementation details explained further.

7.1.1 ViewSets

A *django-rest-framework* viewset, as defined by the documentation, is used to "combine the logic for a set of related views in a single class" [44]. The documentation also states that other frameworks also provided conceptually similar implementations named along the lines of "Resources" or "Controllers". In short these components allow the programmer to quickly define REST endpoints in few lines of code.

Using the **ModelViewSet** provided by *django-rest-framework* it is possible to define CRUD methods with their corresponding REST endpoint in the following manner.

```
class AccountViewSet(viewsets.ModelViewSet):
"""
A simple ViewSet for viewing and editing accounts.
"""
queryset = Account.objects.all()
serializer_class = AccountSerializer
permission_classes = [IsAccountAdminOrReadOnly]
```

The previously defined code generates the following endpoints:

Protocol	**Function**	**Description**
GET	list	Lists all available objects for the corresponding model
GET	retrieve	Retrieve the single instance that corresponds to the attribute sent in the request
POST	create	Create a single instance of the model defined in the viewset, using the attributes sent in the request
PUT	update	Updates the instance specified by a parameter in the endpoint, using the attributes sent in the request
DELETE	delete	Deletes the instance specified by a parameter in the endpoint

Table 7.1: *django-rest-framework* — Endpoints defined by ModelViewSet

This tool defined by the framework eases the implementation of basic functionalities, since the behavior of the *viewset* functions was only overridden when more complex actions arouse.

7.1.2 Serializers

The *django-rest-framework* serializers are a tool to translate data structures sent by requests, made to the previously mentioned, *viewsets*, into Python data structures. In these serializers it is also possible to define validation criteria that must be followed by the incoming data to transform, if these criteria are not met, the viewset will issue a response error with the corresponding messages relating to the error that occurred. As seen in the documentation [45], the serializers are defined as follows:

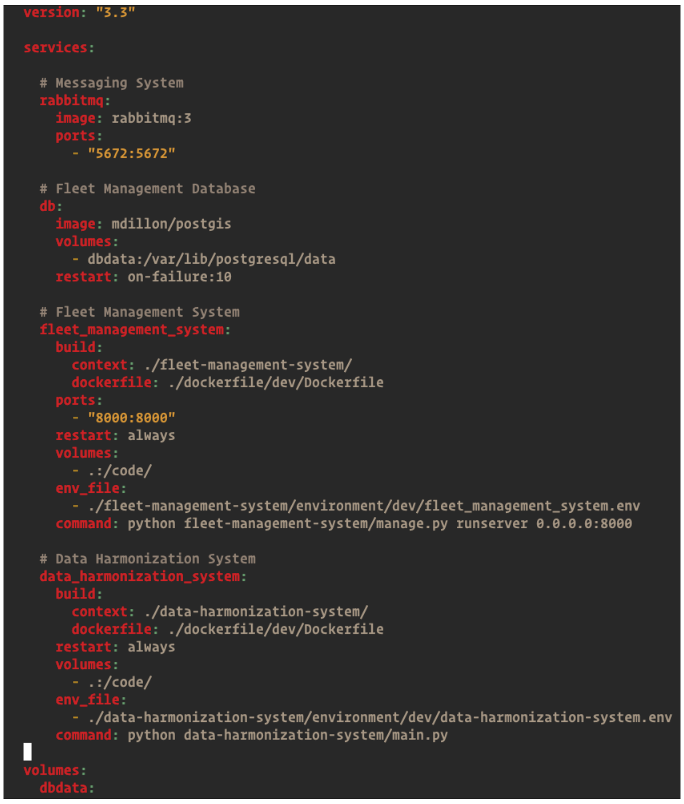
```
from rest_framework import serializers

class CommentSerializer(serializers.Serializer):
    email = serializers.EmailField()
    content = serializers.CharField(max_length=200)
    created = serializers.DateTimeField()
```

In the previous example it is possible to see how the validation criteria are defined for each field in the serializers, as well as some of the available parameters for the validation criteria. These serializers are used in the project to transform incoming **json** data, from the requests made to the endpoints defined by viewsets, into Python data structures to be stored into the database.

7.2 Docker

As previously specified in chapter 6, Docker is used to orchestrate Thumbeo Corporate's environment. The following image refer to the configurations developed to allow Docker to orchestrate the system. Note that both the *Fleet Management System* and the *Data Harmonization System* are built from their own contexts using their respective *Dockerfile*, this is intended not only to isolate the systems but also to make it possible to alter the environment from each system separatly.



```yaml
version: "3.3"

services:

  # Messaging System
  rabbitmq:
    image: rabbitmq:3
    ports:
      - "5672:5672"

  # Fleet Management Database
  db:
    image: mdillon/postgis
    volumes:
      - dbdata:/var/lib/postgresql/data
    restart: on-failure:10

  # Fleet Management System
  fleet_management_system:
    build:
      context: ./fleet-management-system/
      dockerfile: ./dockerfile/dev/Dockerfile
    ports:
      - "8000:8000"
    restart: always
    volumes:
      - .:/code/
    env_file:
      - ./fleet-management-system/environment/dev/fleet_management_system.env
    command: python fleet-management-system/manage.py runserver 0.0.0.0:8000

  # Data Harmonization System
  data_harmonization_system:
    build:
      context: ./data-harmonization-system/
      dockerfile: ./dockerfile/dev/Dockerfile
    restart: always
    volumes:
      - .:/code/
    env_file:
      - ./data-harmonization-system/environment/dev/data-harmonization-system.env
    command: python data-harmonization-system/main.py

volumes:
  dbdata:
```

Figure 7.1: Docker Configurations

7.3 Messaging System

This system, as specified in chapter 6, receives and queues messages containing the data pertaining to the external systems integrated with Thumbeo Corporate. Given that the messaging system is implemented using RabbitMQ, the developed done on this specific system was that of installing and configuring the software to enable its use in the system as whole. As seen in the previously mentioned Docker configuration 7.1 the Messaging System is declared as *rabbitmq*.

7.4 Data Harmonization System

This system in specific, responsible for transforming incoming data from external systems into their respective format, was implemented with using Celery Workers and Models that correspond to the intended data structure.

7.4.1 Models

The models implemented in the Data Harmonization System are used by the Celery Tasks to transform the data consumed by the system into the data to be transmitted into the Fleet Management System. Given that models implemented in the Data Harmonization System have no relationships with one another, it is not possible to construct a diagram showing how they are related.

OBD Readers Models

In order to comply with the Fleet Management System models for recording OBD data, the Data Harmonization System implements models that are analogous to the models present on that system.

Telemetry Event

- **Event Type:** Type of event recorded by the OBD;
- **Event Value:** Value of the event recorded by the OBD;
- **Occurrence Datetime:** Date and time at which the event occurred;
- **Vehicle**: Vehicle that generated the event;
- **Completed Ride**: Ride at which the event was generated;

Thumbeo Models

The following models served as the intermediary between the data sent by the Thumbeo software system and the data transmitted to the *Fleet Management System*.

User

- **Email:** Email of the user;
- **Name:** Name of the user registered on the system sending the data;
- **Phone:** Phone number of the user registered on the system sending the data;
- **Profile Photo:** Profile photo submitted by the user on the system sending the data.

Vehicle

- **Make:** Vehicle manufacturer name;
- **Model:** Vehicle model name;
- **License Plate:** Identification number in the vehicle license plate;
- **Number of Seats:** Number of seats available in the vehicle;

Ride Request

- **User:** User that issued the ride request;
- **Vehicle:** Vehicle to be used in the ride request;
- **Departure Datetime:** Date and time at which the user requesting the ride wants to be picked up;
- **Origin:** Place of origin where the ride should start;
- **Destination:** Place at which the ride should end;
- **Notes:** Miscellaneous information provided by the user that created the ride request;
- **Status:** Status of the ride.

Ride Offer

- **User:** User that issued the ride offer;
- **Vehicle:** Vehicle to be used in the ride offer;
- **Departure Datetime:** Date and time at which the offer will depart;
- **Origin:** Place of origin where the ride should start;
- **Destination:** Place at which the ride should end;
- **Notes:** Miscellaneous information provided by the user that created the ride offer;

Featured Place

- **Name** — Name of the place frequently used;
- **Description** — Description of the place;
- **Coordinates** — Set of coordinates describing the location of the place.

7.4.2 Celery Tasks

A Celery Task, in this project, is the set of instructions that will be used by a Celery Worker to transform the data received into the intended data structure. For the development of the project, a total of four tasks were created, each one imports respectively the **users**, the **vehicles**, the **rides** and the **telemetry data**. As seen in the previous Chapter 6, these tasks are executed by the workers instantiated in the *Data Harmonization Script* and will make use of the models defined in the previous section to transform the raw data coming from the *Messaging System* into Python objects that are written into the database.

7.5 Fleet Management System

For the development of the Fleet Management System the important aspects worth mentioning are the models created for the records that will be kept in the system, and the models that will be stored for the purpose of keeping statistics to show to fleet managers. The extensive definition of the models is present in Annex C, along with a low level entity-relationship diagram showing the attributes.

7.5.1 Models

The following diagram reflects the developed models for the Fleet Management System. Most of the models included were developed with the models of the existing system 6.1 in mind given that the data to be imported was first generated by the system in question, and the development team decided that the work done on the previously mentioned system could and should be taken advantage of when developing this new iteration.

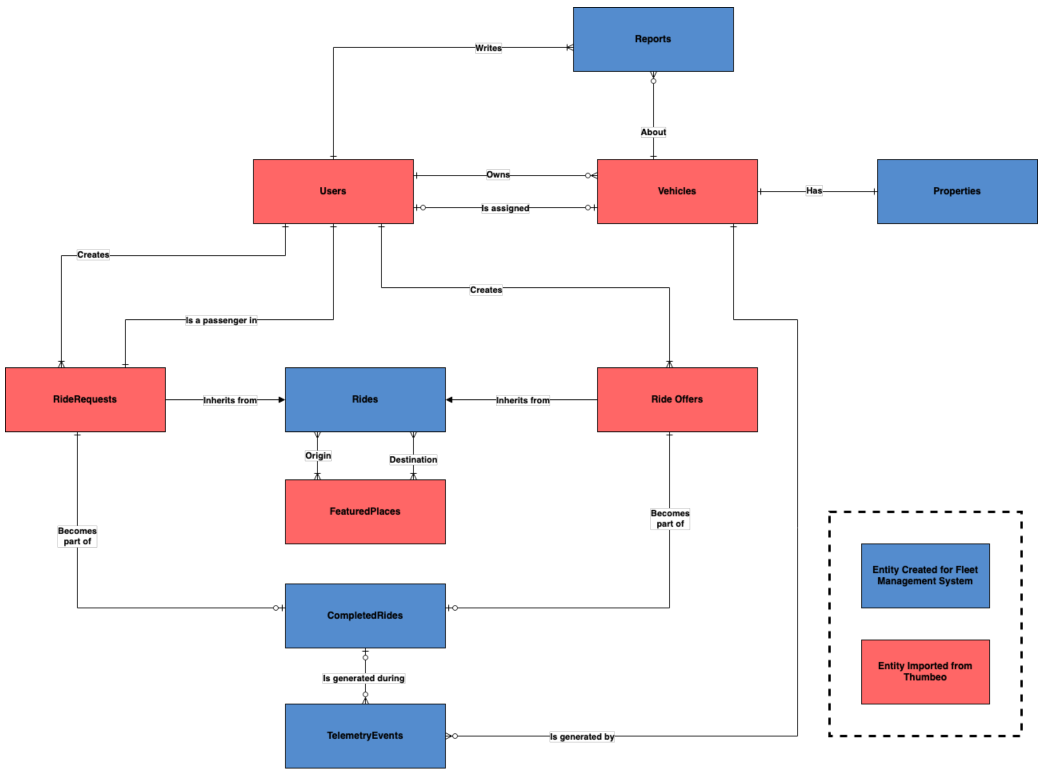


Figure 7.2: Fleet Management System — Entity Relationship Diagram

The full data model developed for the Fleet Management System, complete with all the attributes of each model is present in Annex C.

7.5.2 Statistics

To enable the collection of statistical data, the three different KPIs defined in Chapter 2 of **Fuel Consumption**, **Utilization**, **Availability**.

To implement the **Fuel Consumption** KPI it is necessary to fetch all *TelemetryEvents* that correspond to the type **FUEL_RATE**. This type of Telemetry Event, generated by the OBD Readers, is a record of the instantaneous fuel injection rate into the engine; stored using liters per hour as a unit of measure. Using a ViewSet function that calculates the average fuel rate spent per ride produces the average fuel consumption for a given vehicle; this function then returns the result of this KPI using a serializer to transform the python object into a *json* data structure.

Regarding the **Utilization** KPI, it's implementation consisted of a ViewSet that filters the **Completed Rides** in the database by the Vehicle identifier sent in the request. This restricts the set of all Completed Rides into the subset that was performed using the specified vehicle, allowing the fleet manager to compare the utilization of a given vehicle with any other. The results filtered by the ViewSet are then transformed into a *json* data structure by a serializer and sent in the response body.

The **Availability** KPI is implemented by filtering the set of all vehicles by those who are currently not performing a ride using the **Is Active** field specified in the models C. Alike the previous KPI, a ViewSet implements a function that receives the parameter by which it should filter the fleet of vehicles and returns a *json* data structure with the results of the filtering.

7.6 Testing

In order to assess the quality of the features implemented, a unit testing suite was developed. Further testing strategies were considered, **end-to-end testing** being one of the strategies that made sense to implement into a finalized product, nevertheless, it was decided that it is not worth addressing this type of tests at this stage of development. Therefore, unit testing is the only strategy implemented in the system as a means to validate the implementation of the requirements specified in Chapter 5.

7.6.1 Unit Testing

The main goal of performing unit tests is to isolate each part (or unit) of a particular software in order to show that the individual parts are correct. In the specific context of this system, the units are defined as the different components of the Fleet Management System: *User Component*, *Vehicle Component*, *Rides Component*, *Statistics Component*.

The tests performed and their respective results are described in the tables D.1, D.2, D.3, D.4 and D.5 of Annex D

7.6.2 Data Harmonization System

Regarding the testing of the components of the *Data Harmonization System*, the main concern was testing the transmission of data incoming from the Messaging System, the transformation of the data into the intended data model and ultimately the process of writing the data into the database.

- **User Data**
  - Received data that is valid, is transformed into a User Model;
  - Received data that is invalid is discarded;
  - User Models can be written into the Fleet Management Database.
- **Vehicles Data**
  - Received data that is valid, is transformed into a Vehicle Model;
  - Received data that is invalid is discarded;
  - Vehicle Models can be written into the Fleet Management Database.
- **Rides Data**
  - Received data that is valid, is transformed into a Ride Request or Ride Offer;
  - Received data that is invalid is discarded;
  - Vehicle Models can be written into the Fleet Management Database.
- **OBD Data**
  - Received data that is valid, is transformed into a Telemetry Event;
  - Received data that is invalid is discarded;
  - Telemetry Events can be written into the Fleet Management Database.

7.6.3 Fleet Management System

When unit testing the different components of the Fleet Management System, the concern was in testing actions of creating, reading, updating or deleting the specific models defined for each component. A summary of the unit tests performed for each component is as follows:

- **Users Component**
  - Create a new user along with a profile;
  - Fetch a user profile or a set of user profiles;
  - Update a user and user profile information;
  - Delete user information;
- **Vehicles Component**
  - Create a new vehicle;
  - Fetch a vehicle or a set of vehicles;
  - Update vehicle information;
  - Delete vehicle information;
  - Filter vehicles by their utilization in rides;
  - Filter vehicles by their immediate availability;
- **Rides Component**
  - Create ride requests and ride offers;
  - Fetch ride requests or ride offers;
  - Update information in ride offers and ride requests;
  - Delete ride offers and ride requests.
- **Statistics Component**
  - Fetch the average fuel consumption for a given vehicle.
  - Fetch the average fuel consumption for all vehicles in the fleet.

Chapter 8

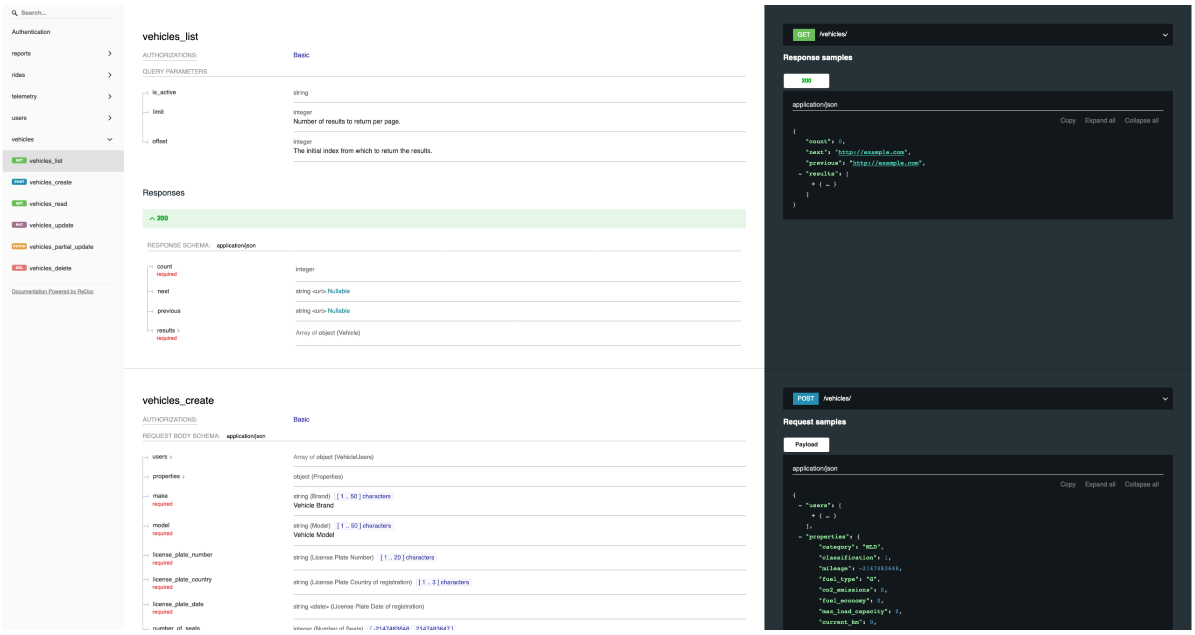
Final Product of the Internship

In this chapter, Thumbeo Corporate is presented following the implementation phase of this internship. Given that no front-end application or web interface was within the scope of the project most of the demonstration will occur with the help of pictures from the documentation developed for the *Fleet Management System*, the administration interface that comes bundled with *django-rest-framework* or requests sent to the *Data Harmonization System*.

At the start of this internship, Thumbeo Corporate as a product had stagnated it's development. The study of the concepts in Chapter 2 were essential to understand that the product had stagnated because it created yet another service for fleet managers to use. After studying the concepts that were the basis for Thumbeo Corporate's initial ideation, present in Chapter 3, the focus of the internship became the creation of a system capable of consuming data from other systems along with the fleet management features defined in Chapter 5. The idea behind creating a system that consumes data from other systems serves the purpose of helping fleet managers unify the systems already in use into a single software, and the architecture of this system is defined in Chapter 6.

8.1 Fleet Management System

Regarding the *Fleet Management System*, there are some aspects that can be shown, namely the documentation developed for the system, along with a administration interface. Initially, the documentation developed for the *Fleet Management System* allows for anyone to explore the endpoints implemented along with the attributes required and allowed by each one.



vehicles_list

AUTHORIZATIONS: Basic

QUERY PARAMETERS

is_active	string
limit	integer Number of results to return per page.
offset	integer The initial index from which to return the results.

Responses

200

RESPONSE SCHEMA: application/json

count required	integer
next	string <uri> Nullable
previous	string <uri> Nullable
results › required	Array of object (Vehicle)

application/json

```
{
    "count": 0,
    "next": "http://example.com",
    "previous": "http://example.com",
  - "results": [
      + { … }
    ]
}
```

vehicles_create

AUTHORIZATIONS: Basic

REQUEST BODY SCHEMA: application/json

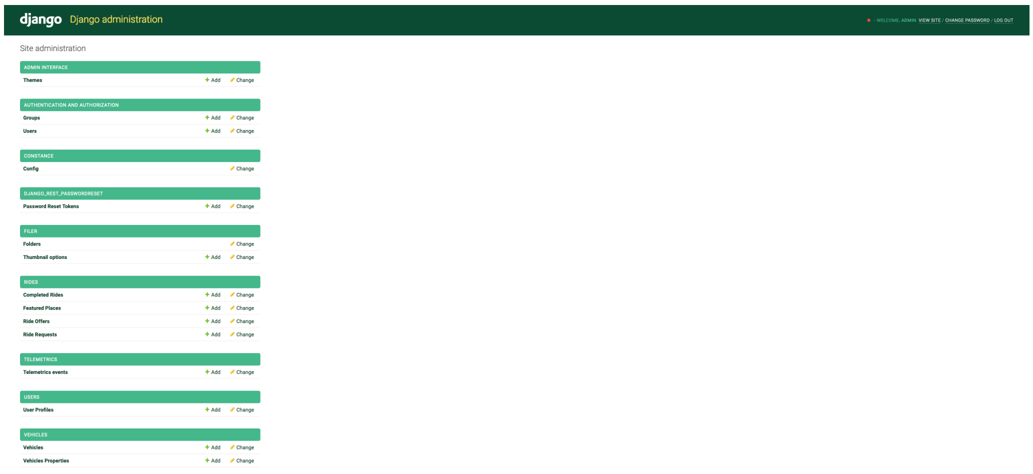
users ›	Array of object (VehicleUsers)
properties ›	object (Properties)
make required	string (Brand) [1 .. 50] characters Vehicle Brand
model required	string (Model) [1 .. 50] characters Vehicle Model
license_plate_number required	string (License Plate Number) [1 .. 20] characters
license_plate_country required	string (License Plate Country of registration) [1 .. 3] characters
license_plate_date required	string <date> (License Plate Date of registration)
number_of_seats	integer (Number of Seats) [-2147483648 .. 2147483647]

application/json

```
{
  - "users": [
      + { … }
    ],
  - "properties": {
        "category": "MLD",
        "classification": 1,
        "mileage": -2147483648,
        "fuel_type": "G",
        "co2_emissions": 0,
        "fuel_economy": 0,
        "max_load_capacity": 0,
        "current_km": 0,
```

Figure 8.1: Fleet Management System — API Documentation

Lastly, the administration interface provided by *django-rest-framework* allows anyone with the administration credentials to view and change the records of the system, nevertheless some of the functional aspects provided by the API are not available from this interface.

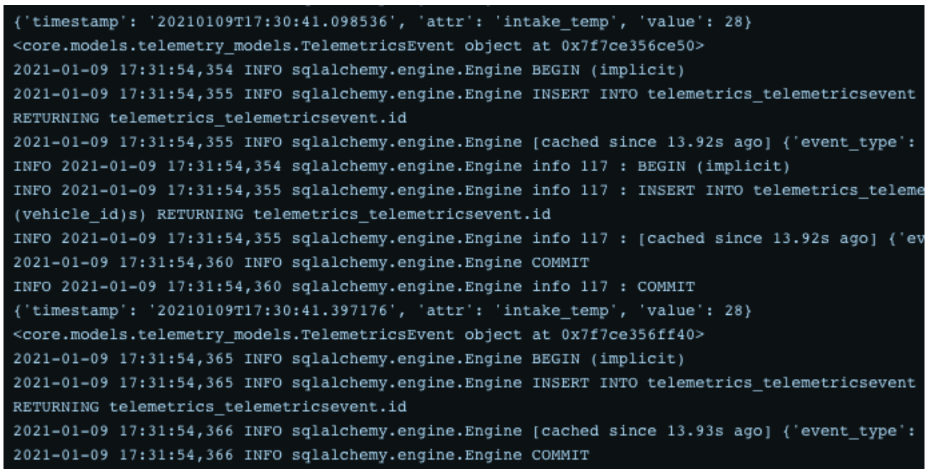

django Django administration
- WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration

ADMIN INTERFACE		
Themes	+ Add	Change
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change
CONSTANCE		
Config		Change
DJANGO_REST_PASSWORDRESET		
Password Reset Tokens	+ Add	Change
FILER		
Folders		Change
Thumbnail options	+ Add	Change
RIDES		
Completed Rides	+ Add	Change
Featured Places	+ Add	Change
Ride Offers	+ Add	Change
Ride Requests	+ Add	Change
TELEMETRICS		
Telemetrics events	+ Add	Change
USERS		
User Profiles	+ Add	Change
VEHICLES		
Vehicles	+ Add	Change
Vehicles Properties	+ Add	Change

Figure 8.2: Fleet Management System — Administration Interface

8.2 Data Harmonization System

The *Data Harmonization System*, as demonstrated in figure 6.9 in Chapter 6, transforms incoming data into data that the *Fleet Management System* can understand the following demonstrations present the harmonization process in action, as well as the transformed data now included in the product *Thumbeo Corporate*.

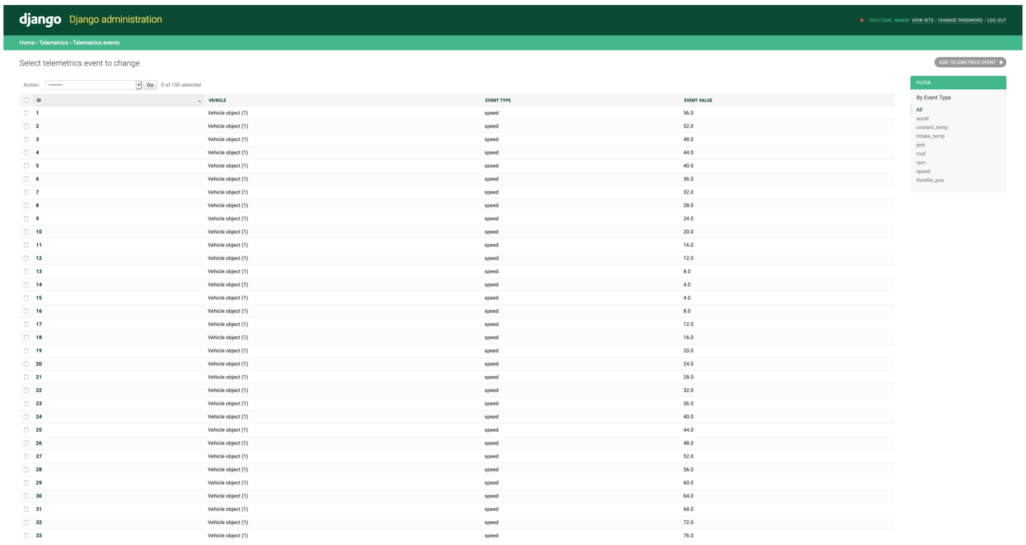


```
{'timestamp': '20210109T17:30:41.098536', 'attr': 'intake_temp', 'value': 28}
<core.models.telemetry_models.TelemetricsEvent object at 0x7f7ce356ce50>
2021-01-09 17:31:54,354 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2021-01-09 17:31:54,355 INFO sqlalchemy.engine.Engine INSERT INTO telemetrics_telemetricsevent
RETURNING telemetrics_telemetricsevent.id
2021-01-09 17:31:54,355 INFO sqlalchemy.engine.Engine [cached since 13.92s ago] {'event_type':
INFO 2021-01-09 17:31:54,354 sqlalchemy.engine.Engine info 117 : BEGIN (implicit)
INFO 2021-01-09 17:31:54,355 sqlalchemy.engine.Engine info 117 : INSERT INTO telemetrics_teleme
(vehicle_id)s) RETURNING telemetrics_telemetricsevent.id
INFO 2021-01-09 17:31:54,355 sqlalchemy.engine.Engine info 117 : [cached since 13.92s ago] {'ev
2021-01-09 17:31:54,360 INFO sqlalchemy.engine.Engine COMMIT
INFO 2021-01-09 17:31:54,360 sqlalchemy.engine.Engine info 117 : COMMIT
{'timestamp': '20210109T17:30:41.397176', 'attr': 'intake_temp', 'value': 28}
<core.models.telemetry_models.TelemetricsEvent object at 0x7f7ce356ff40>
2021-01-09 17:31:54,365 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2021-01-09 17:31:54,365 INFO sqlalchemy.engine.Engine INSERT INTO telemetrics_telemetricsevent
RETURNING telemetrics_telemetricsevent.id
2021-01-09 17:31:54,366 INFO sqlalchemy.engine.Engine [cached since 13.93s ago] {'event_type':
2021-01-09 17:31:54,366 INFO sqlalchemy.engine.Engine COMMIT
```

Figure 8.3: Data Harmonization System — Harmonization Process

In the previous image 8.3 the harmonization process is demonstrated, the first line of the image shows the data received by the *Data Harmonization System* of an event generated by an external system, the second line shows the data transformed into a model to be stored into the database; a process which is shown to take process in the lines that follow, with the process repeating for the next event harmonized.

In the next image, it is possible to see that the data harmonized by the *Data Harmonization System* is now accessible in *Thumbeo Corporate* and is visible in the administration interface.


django Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Telemetrics › Telemetrics events
ADD TELEMETRICS EVENT
Action: Go
FILTER
By Event Type
All
accel
coolant_temp
intake_temp
jerk
maf
rpm
speed
throttle_pos

Select telemetrics event to change

0 of 100 selected

ID	VEHICLE	EVENT TYPE	EVENT VALUE
1	Vehicle object (1)	speed	56.0
2	Vehicle object (1)	speed	52.0
3	Vehicle object (1)	speed	48.0
4	Vehicle object (1)	speed	44.0
5	Vehicle object (1)	speed	40.0
6	Vehicle object (1)	speed	36.0
7	Vehicle object (1)	speed	32.0
8	Vehicle object (1)	speed	28.0
9	Vehicle object (1)	speed	24.0
10	Vehicle object (1)	speed	20.0
11	Vehicle object (1)	speed	16.0
12	Vehicle object (1)	speed	12.0
13	Vehicle object (1)	speed	8.0
14	Vehicle object (1)	speed	4.0
15	Vehicle object (1)	speed	4.0
16	Vehicle object (1)	speed	8.0
17	Vehicle object (1)	speed	12.0
18	Vehicle object (1)	speed	16.0
19	Vehicle object (1)	speed	20.0
20	Vehicle object (1)	speed	24.0
21	Vehicle object (1)	speed	28.0
22	Vehicle object (1)	speed	32.0
23	Vehicle object (1)	speed	36.0
24	Vehicle object (1)	speed	40.0
25	Vehicle object (1)	speed	44.0
26	Vehicle object (1)	speed	48.0
27	Vehicle object (1)	speed	52.0
28	Vehicle object (1)	speed	56.0
29	Vehicle object (1)	speed	60.0
30	Vehicle object (1)	speed	64.0
31	Vehicle object (1)	speed	68.0
32	Vehicle object (1)	speed	72.0
33	Vehicle object (1)	speed	76.0

Figure 8.4: Thumbeo Corporate — Harmonized Data

The vehicle telemetry data was only accessible through a mobile application the connected to the OBD devices and now, through the *Data Harmonization System*, it is accessible in the product *Thumbeo Corporate*, allowing fleet managers to track this data in single system, where the remaining data of their vehicles is centralized.

This page is intentionally left blank.

Chapter 9

Conclusions

This chapter concludes this report of the internship, starting by summarizing the conclusions withdrawn from the work done, following with the tasks planned for the future work of this internship and ending with the lessons learned during the internship.

9.1 Work Done

During this internship the problem identified was that of the decentralization of data belonging to fleet management systems; that is, fleet managers who use multiple systems to manage their fleets, have their data scattered throughout the systems used. This internship proposes a solution to this problem by implementing a fleet management system that is capable of consuming data from external systems, using a data harmonization system. Along with the implementation of this data harmonization system in Ubiwhere's fleet management system, Thumbeo Corporate.

Reflecting upon the work done during this internship, considering the success criteria established in Chapter 4, the overall internship is classified as success. The system is developed with all requirements with "Must Have" priority implemented and it satisfies all the quality attributes established in Chapter 5.

The objectives specified in Chapter 1 are also met. The academic research presented encompassed fleet management topics from the definition of what is fleet management in the context of this thesis, the specification of what data should be present in a fleet management system and finally what metrics can be tracked from the data present in such systems. As specified before, the architecture proposed also fulfills the need of unifying data from various systems into a single fleet management system, and the implementation of said system is also in line with the requirements and quality attributes defined.

9.2 Future Work

Although the success criteria are met and the system is developed, it can still be improved in several ways. First, a graphical interface is necessary for users to interact with the system developed. Next, focusing more on implementing more KPI related features would allow fleet managers to have access to more metrics to manager their fleets. Lastly, partnering with companies can allow the development of more models for the *Data Harmonization System* and successively the implementation of more features into the fleet management system, that are shown to be desired by fleet managers.

All of the previous aspects contribute for the improvement of Thumbeo Corporate not only as Fleet Management System, but also as a system that is capable of centralizing the information regarding fleets into a single source of truth for fleet managers to analyze when evaluating their fleets, which will improve their day to day operations since all the data regarding their fleet is centralized. More KPIs allow fleet managers to analyze the fleet with different perspectives in mind and making decisions with more information at hand; more models for the *Data Harmonization System* allows the system itself to respond to a wider range of needs by fleet managers; and lastly a graphical interface would tie all these aspects together and bring them to fleet managers in such a way that the information stored by the system can be interpreted by those visualizing the data.

9.3 Lessons Learned

The internship provided opportunities to learn new skills and solidify already existent knowledge. Ubiwhere, directly and indirectly, influenced the learning process of the junior developer, directly by the feedback provided on the work produced and indirectly by entrusting the developer with the responsibilities inherent to the internship.

The skills learned belonged to two different categories, soft-skills and hard-skills. In terms of soft-skills, the internship provided a full semester worth of communication experience inside a company, which taught the junior developer how to professionally negotiate his opinion whenever the decision was important enough to be discussed. On the category of hard-skills, the writing of an academic report was the most important skill learned; a skill that involves reading and analyzing how the report delivers its message.

The preexistent skills consolidated during this internship consisted of hard-skill developed during the Masters course. The elicitation of requirements, the design of an architecture for a system and the planning of a project are part of the hard-skills improved during the internship.

References

[1] "Dblp — computer science bibliography." https://dblp.uni-trier.de/.

[2] "Google scholar." https://scholar.google.com/.

[3] "Repositórios científicos de acesso aberto de portugal." https://www.rcaap.pt/.

[4] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," *CoRR*, vol. abs/1802.06444, 2018.

[5] W. B. Powell and H. Topaloglu, "Fleet management," in *Applications of Stochastic Programming, Math Programming Society - SIAM Series on Optimization*, 2005.

[6] M. Falco, I. Núñez, and F. Tanzi, "Improving the fleet monitoring management, through a software platform with iot," in *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, 2019.

[7] F. Monnerat, J. Dias, and M. J. Alves, "Fleet management: A vehicle and driver assignment model," *European Journal of Operational Research*, vol. 278, 2019.

[8] F. Calderón and E. J. Miller, "Bridging the knowledge gap in ridehailing service provision with human-driven fleets: A data mining approach," *Procedia Computer Science*, vol. 170, 2020. The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[9] R. Vosooghi, J. Kamel, J. Puchinger, V. Leblond, and M. Jankovic, "Robo-taxi service fleet sizing: assessing the impact of user trust and willingness-to-use," *Transportation*, vol. 46, p. 1997–2015, May 2019.

[10] E. Szczepański, M. Nivette, I. Jacyna-Gołda, and M. Izdebski, "The car fleet management model with including expectations of the users," *Journal of KONES*, vol. 25, no. 4, pp. 489 – 500, 01 Dec. 2018.

[11] E. Türk and M. Challenger, "An android-based iot system for vehicle monitoring and diagnostic," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, 2018.

[12] Environmental Protection Agency (EPA), "(...) Regulations Requiring Availability of Information for Use of On-Board Diagnostic Systems (...)." [Available at: https://www.govinfo.gov/content/pkg/FR-1995-08-09/pdf/95-18867.pdf; Online; accessed 15-February-2020].

[13] European Parliament, "DIRECTIVE 98/69/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL." [Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:31998L0069&from=en; Online; accessed 16-February-2020].

[14] G. Signoretti, M. Silva, J. Araujo, I. Silva, D. Silva, P. Ferrari, and E. Sisinni, "A dependability evaluation for obd-ii edge devices: An internet of intelligent vehicles perspective," in *2019 9th Latin-American Symposium on Dependable Computing (LADC)*, 2019.

[15] I. O. for Standardization, "Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics," standard, International Organization for Standardization, Oct. 2015.

[16] R. Malekian, N. R. Moloisane, L. Nair, B. T. Maharaj, and U. A. K. Chude-Okonkwo, "Design and implementation of a wireless obd ii fleet management system," *IEEE Sensors Journal*, vol. 17, p. 1154–1164, Feb 2017.

[17] L. Flowers, oct 2019. [Available at: `https://www.fleetio.com/blog/15-metrics-every-fleet-manager-should-be-tracking`; Online; accessed 14-September-2020].

[18] F. Nicole H, "Key performance indicator survey," 2013. [Available at: `http://www.fleetanswers.com/sites/default/files/KPI_Report_0.pdf`]; Online; access 14-September-2020.

[19] S.-M. Juntunen, "Key performance indicators of transportation category management," 2017. [Available at: `https://www.theseus.fi/bitstream/handle/10024/132096/Juntunen_Sanna-Mari.pdf?sequence=1`].

[20] V. Verde, "Via Verde - Soluções para Empresas." `https://www.viaverde.pt/empresas/Solu%C3%A7%C3%B5es`, accessed 21-Jun-2020.

[21] Odoo, "Odoo." `https://www.odoo.com/`, accessed 21-Jun-2020.

[22] M. Pro, "MTC Pro." `https://www.mtcpro.com/`, accessed 21-Jun-2020.

[23] Cartrack, "Cartrack." `https://www.cartrack.pt/`, accessed 21-Jun-2020.

[24] Fleetio, "Fleetio." `https://www.fleetio.com/`, accessed 21-Jun-2020.

[25] Samsara, "Samsara." `https://www.samsara.com/`, accessed 21-Jun-2020.

[26] FleetSoft, "FleetSoft." `https://fleet-maintenance.com/`, accessed 21-Jun-2020.

[27] S. Automotive, "Stratio Automotive." `https://stratioautomotive.com/`, accessed 21-Jun-2020.

[28] Volkswagen, "Volkswagen Financial Services Fleets." `https://www.vwfsfleet.co.uk/`, accessed 21-Jun-2020.

[29] L. Hertz, "Localiza Hertz." `https://www.localizahertz.com/others/en-us`, accessed 21-Jun-2020.

[30] E. F. Management, "Enterprise Fleet Management." `https://www.vwfsfleet.co.uk/`, accessed 21-Jun-2020.

[31] Frotcom, "Frotcom - Intelligent Fleets." `https://www.frotcom.com/pt-pt/gestao-de-frotas`, accessed 21-Jun-2020.

[32] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001.

[33] European Parliament and Council of European Union, “Regulation (...) on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (...).” [Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679; Online; accessed 22-May-2020].

[34] M. R. @O’Reilly, “Microservices Architeture Pattern.” https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch04.html, accessed 21-Jun-2020.

[35] M. R. @O’Reilly, “Layered Architeture Pattern.” https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html, accessed 3-Nov-2020.

[36] “Django.” https://www.djangoproject.com/, accessed 21-Jun-2020.

[37] “django-rest-framework.” https://www.django-rest-framework.org/, accessed 21-Jun-2020.

[38] PostgreSQL. https://www.postgresql.org/, accessed 21-Jun-2020.

[39] “Docker.” https://www.docker.com/, accessed 21-Jun-2020.

[40] “docker-compose.” https://docs.docker.com/compose/, accessed 21-Jun-2020.

[41] “Docker Swarm.” https://docs.docker.com/engine/swarm/, accessed 21-Jun-2020.

[42] RabbitMQ. https://www.rabbitmq.com/, accessed 24-Jun-2020.

[43] RabbitMQ, “Clients Libraries and Developer Tools.” https://www.rabbitmq.com/devtools.html, accessed 24-Jun-2020.

[44] django-rest framework, “ViewSets.” https://www.django-rest-framework.org/api-guide/viewsets/#viewsets, accessed 04-Jan-2021.

[45] D. R. Framework, “Serializers.” https://www.vwfsfleet.co.uk/, accessed 04-Jan-2021.

[46] European Parliament, “DIRECTIVE 2014/45/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL.” [Available at: https://op.europa.eu/en/publication-detail/-/publication/b560a38a-cf66-11e3-b682-01aa75ed71a1/language-en; Online; accessed 16-February-2020].

[47] Federal Motor Carrier Safety Administration, “Section § 396.17: Periodic inspection..” [Available at: https://www.fmcsa.dot.gov/regulations/title49/section/396.17; Online; accessed 16-February-2020].

[48] Federal Motor Carrier Safety Administration, “Section § 396.11: Driver vehicle inspection report(s).” Available at: https://www.fmcsa.dot.gov/regulations/title49/section/396.11; Online; accessed 28-February-2020.

[49] Appa’s Facilities Manager, “ Total Cost of Ownership.” [Available at: https://web.archive.org/web/20180215023821/https://www.appa.org/documents/TCOArticleJun-Jul2016.pdf; Online; accessed 20-February-2020].

[50] S. P. C. L. F. T. P. Enderle, “Fleet management in europe,” tech. rep., Deloitte Touche Tohmatsu Limited, 2017. [Available at: https://www2.deloitte.com/content/dam/Deloitte/cz/Documents/consumer-and-industrial/cz-fleet-management-in-europe.pdf; Online; Accessed 29-February-2020].

[51] "Depreciation Tables." [Available at: https://bit.ly/36tgd76 , http://www.lra.pt/ficheiros/automovel/fidelidademundial_liber.pdf, https://www.caravelaseguros.pt/wp-content/uploads/2019/03/1.CGerais-Auto.pdf, https://www.ocidental.pt/media/2009/au-frotas-cge.pdf; Online; accessed 20-February-2020].

[52] V. Verde, "Vehicle classification." https://www.viaverde.pt/particulares/Ferramentas/classe-veiculos.

Appendix A

Thumbeo Corporate — Applied Concepts

A.1 Applied Concepts

Fleet Management is a topic comprised of many important concepts that must be researched to understand the whole for its parts. The subtopics that constitute Fleet Management, and that were applied by Thumbeo Corporate, are explained in the present section. The topics studied belong to various levels of fleet management work, such as expense management to minimize client costs; vehicle maintenance to improve fleet conditions and maintain vehicle value, and vehicle depreciation to maximize company revenue.

A.1.1 Vehicle Maintenance

Within the context of Fleet Management, the maintenance of every vehicle is a key aspect in optimizing fleet expenses and minimizing fleet down-time. The focus of the research lied on identifying when maintenance must be done and what components can and should be actively observed to attain the goals posed by fleet maintenance. Meanwhile, although the previous topics are the most important, it is also worth mentioning how vehicle faults are detected nowadays; what instruments are used to disclose these faults, and what faults are most impactful for a vehicle's condition. Regarding **how** vehicle faults are currently detected, this can be the following ways: brand recommended inspections; government-regulated audits or driver reports.

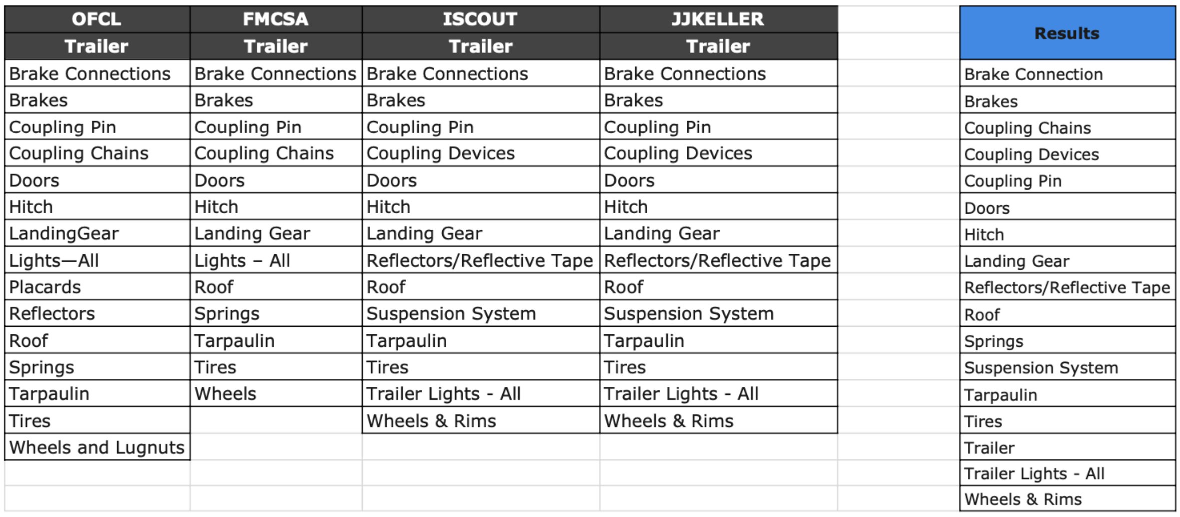
Brand Recommended Inspections In respects to brand recommended inspections, they translate into how and when a certain brand recommends its customers to audit their vehicles in brand-specific workshops. This step is especially important, given that most government regulations are trying to match current market trends, and, as such, owners must be able to inspect and repair their vehicles according to their needs. An example of this is the lack of inspection regulations on auto-gas in Portugal; to this day, there are currently no laws regulating fuel emissions of auto-gas, therefore it isn't inspected on the mandatory periodical inspections. Meanwhile, a user of auto-gas would certainly need to maintain its engine, and so it must go to a brand audit or a third-party workshop to accomplish said goal.

Government-Regulated Audits Analyzing government-regulated audits poses a different problem altogether. Depending on the continent, regulations vary from mandatory inspections every six months to some vehicles, to no inspections at all for most vehicle types. Looking at the European Union regulation, the EU Directive 2014/45 [46] obliges all member states to perform periodic inspections to validate vehicle safety and emission values. In the American continent, Canada and the United States vehicle inspections are regulated on a state basis [47], meaning that each state is free to mandate or forego inspections as they see fit, while Brazil has yet to mandate vehicle inspections at all, let alone regulate the process.

Driver Vehicle Inspection Report These types of inspections were legally required by the United States law upon the approval of Federal Regulation 49 CFR§396.11 [48]. Companies can have their vehicles hindered from performing their duty when these inspections are lacking or failing. Driver Vehicle Inspection Report (DVIR)s are formal records concerning a vehicle's current working state; to be performed before and after every work with the vehicle. The importance of DVIRs in the context of this thesis relies on the fact that it presents some guidelines for which components could be inspected, disregarding the fact that these might not be inspectable by current-day software and hardware. These inspection report forms will also serve as the base for building a template for the report to be filled by drivers after each trip.

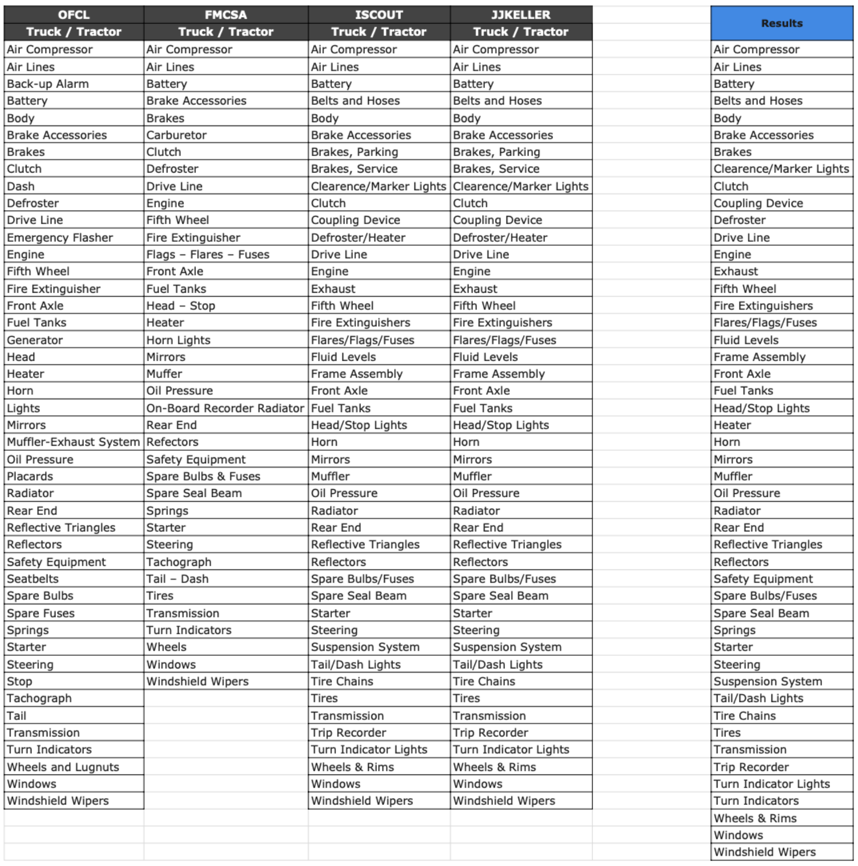
Moving on from **how** faults are identified, the next logical step is to understand **what** faults are searched for, and what are their consequences for vehicles in general. Using the DVIR examples analyzed, it is possible to identify the vehicle components that could be searched for faults.

This analysis was done by utilizing the DVIRs provided to their employees from four different companies and gathering all the different components queried in all inspection reports. The results of the analysis can be found in following images.



OFCL	FMCSA	ISCOUT	JJKELLER		Results
Trailer	Trailer	Trailer	Trailer		
Brake Connections	Brake Connections	Brake Connections	Brake Connections		Brake Connection
Brakes	Brakes	Brakes	Brakes		Brakes
Coupling Pin	Coupling Pin	Coupling Pin	Coupling Pin		Coupling Chains
Coupling Chains	Coupling Chains	Coupling Devices	Coupling Devices		Coupling Devices
Doors	Doors	Doors	Doors		Coupling Pin
Hitch	Hitch	Hitch	Hitch		Doors
LandingGear	Landing Gear	Landing Gear	Landing Gear		Hitch
Lights—All	Lights – All	Reflectors/Reflective Tape	Reflectors/Reflective Tape		Landing Gear
Placards	Roof	Roof	Roof		Reflectors/Reflective Tape
Reflectors	Springs	Suspension System	Suspension System		Roof
Roof	Tarpaulin	Tarpaulin	Tarpaulin		Springs
Springs	Tires	Tires	Tires		Suspension System
Tarpaulin	Wheels	Trailer Lights - All	Trailer Lights - All		Tarpaulin
Tires		Wheels & Rims	Wheels & Rims		Tires
Wheels and Lugnuts					Trailer
					Trailer Lights - All
					Wheels & Rims

Truck/Tractor DVIR Analysis



OFCL	FMCSA	ISCOUT	JJKELLER		Results
Truck / Tractor	Truck / Tractor	Truck / Tractor	Truck / Tractor		
Air Compressor	Air Compressor	Air Compressor	Air Compressor		Air Compressor
Air Lines	Air Lines	Air Lines	Air Lines		Air Lines
Back-up Alarm	Battery	Battery	Battery		Battery
Battery	Brake Accessories	Belts and Hoses	Belts and Hoses		Belts and Hoses
Body	Brakes	Body	Body		Body
Brake Accessories	Carburetor	Brake Accessories	Brake Accessories		Brake Accessories
Brakes	Clutch	Brakes, Parking	Brakes, Parking		Brakes
Clutch	Defroster	Brakes, Service	Brakes, Service		Clearence/Marker Lights
Dash	Drive Line	Clearence/Marker Lights	Clearence/Marker Lights		Clutch
Defroster	Engine	Clutch	Clutch		Coupling Device
Drive Line	Fifth Wheel	Coupling Device	Coupling Device		Defroster
Emergency Flasher	Fire Extinguisher	Defroster/Heater	Defroster/Heater		Drive Line
Engine	Flags – Flares – Fuses	Drive Line	Drive Line		Engine
Fifth Wheel	Front Axle	Engine	Engine		Exhaust
Fire Extinguisher	Fuel Tanks	Exhaust	Exhaust		Fifth Wheel
Front Axle	Head – Stop	Fifth Wheel	Fifth Wheel		Fire Extinguishers
Fuel Tanks	Heater	Fire Extinguishers	Fire Extinguishers		Flares/Flags/Fuses
Generator	Horn Lights	Flares/Flags/Fuses	Flares/Flags/Fuses		Fluid Levels
Head	Mirrors	Fluid Levels	Fluid Levels		Frame Assembly
Heater	Muffer	Frame Assembly	Frame Assembly		Front Axle
Horn	Oil Pressure	Front Axle	Front Axle		Fuel Tanks
Lights	On-Board Recorder Radiator	Fuel Tanks	Fuel Tanks		Head/Stop Lights
Mirrors	Rear End	Head/Stop Lights	Head/Stop Lights		Heater
Muffler-Exhaust System	Refectors	Horn	Horn		Horn
Oil Pressure	Safety Equipment	Mirrors	Mirrors		Mirrors
Placards	Spare Bulbs & Fuses	Muffler	Muffler		Muffler
Radiator	Spare Seal Beam	Oil Pressure	Oil Pressure		Oil Pressure
Rear End	Springs	Radiator	Radiator		Radiator
Reflective Triangles	Starter	Rear End	Rear End		Rear End
Reflectors	Steering	Reflective Triangles	Reflective Triangles		Reflective Triangles
Safety Equipment	Tachograph	Reflectors	Reflectors		Reflectors
Seatbelts	Tail – Dash	Spare Bulbs/Fuses	Spare Bulbs/Fuses		Safety Equipment
Spare Bulbs	Tires	Spare Seal Beam	Spare Seal Beam		Spare Bulbs/Fuses
Spare Fuses	Transmission	Starter	Starter		Spare Seal Beam
Springs	Turn Indicators	Steering	Steering		Springs
Starter	Wheels	Suspension System	Suspension System		Starter
Steering	Windows	Tail/Dash Lights	Tail/Dash Lights		Steering
Stop	Windshield Wipers	Tire Chains	Tire Chains		Suspension System
Tachograph		Tires	Tires		Tail/Dash Lights
Tail		Transmission	Transmission		Tire Chains
Transmission		Trip Recorder	Trip Recorder		Tires
Turn Indicators		Turn Indicator Lights	Turn Indicator Lights		Transmission
Wheels and Lugnuts		Wheels & Rims	Wheels & Rims		Trip Recorder
Windows		Windows	Windows		Turn Indicator Lights
Windshield Wipers		Windshield Wipers	Windshield Wipers		Turn Indicators
					Wheels & Rims
					Windows
					Windshield Wipers

Trailer DVIR Analysis

A.1.2 Expense Management

When managing a fleet, it is important to track all costs involved in its existence. Subsequently, two methods of tracking expenses are presented. Although one method can be considered a subset of the other, both have their place and time to be used, and as such both deserve to be explained and compared.

Total Cost of Ownership TCO is defined as the purchase price of an asset plus the costs of its operations [49]. The importance of this concept relies on the necessity of calculating the TCO for all vehicles in fleet. This metric is important for the product to develop because it is one of its main selling points. Clients need to know how much their fleet costs and how much Ubiwhere's product can help them save. Showing this information and making it the main selling point can help the solution garner the clients it needs.

Total Cost of Mobility TCM is a term applied to describe the total sum of the costs related to every part of mobility in a company. Besides calculating solely the cost per vehicle, it also takes into account other mobility options used by employees, that is taxis, flights, carpooling or rented/leased vehicles[50]. "Today, more and more companies tend to analyze and optimize the total cost of mobility (TCM) rather than the TCO"[50].

TCO vs TCM Comparing both methods, as said previously, both have a time and a place to be used. There is no point in using TCM if the company does not plan to refund trips undertaken by employees using other mobility methods outside the companies fleet. In contrast, there is no point in calculating TCO if the company does not own a fleet at all. Nevertheless, both calculation methods can be used in conjunction with one another; to provide clients with another layer of metrics to their management needs.

A.1.3 Vehicle Depreciation

A key topic within Fleet Management is the concept of vehicle depreciation. As a fleet manager, it is crucial to know when vehicles are starting to lose their value and should be sold to minimize fleet expenses while maximizing their net worth.

Vehicle depreciation can be defined as the value lost by a vehicle caused by its usage over its life-cycle. Research proved that there are two distinct ways to calculate this depreciation, depending on each country's laws on vehicle value, mobility taxes and market interests.

Depreciation Tables

One way of calculating vehicle depreciation, specifically in Portugal, is using Depreciation Tables provided by insurance companies [51]. These tables provide an average value for every vehicle; with the objective of helping the insurance company evaluate a vehicles worth in case of a casualty. However, this method presents a problem; it doesn't take into account the market value of a vehicle, and understandably so. In case of a casualty, the company's interest is paying the least amount of money possible, and the insured's interest is paying the least possible quota for the insurance during the contract.

Market Analysis

An alternative way to calculate vehicle depreciation is by analyzing market values and calculating the average worth based on the actual data available. This method presents some initial challenges; firstly it is necessary to define how many markets to explore and what sellers should be analyzed; should independent/third-party sellers be considered? Must we only consider official stands? Thereafter, the challenge of mounting an infrastructure capable of observing these sellers, while evaluating certain vehicles, has to be taken into account. Ultimately, it is also of utmost importance to be able to distinguish correct vehicle models and their minute differences to evaluate a vehicle correctly. In practice, this last challenge translates into identifying the differences between all the versions of the same make and model of a vehicle.

Appendix B

Requirement Specification — Functional Requirements

Data Harmonization

DH01

- **Epic:** Data Harmonization
- **Description:** As a fleet manager, I want to import data from vehicles registered in an external platform to Thumbeo Corporate, in order to centralize vehicle information in a single system.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager imports vehicle to Thumbeo Corporate.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the fleet page
      And select the "import vehicles" button
    - **Then** the system communicates with the external system to import the vehicles present in the system itself.
- **Dependencies:** None
- **Priority:** Must Have

DH02

- **Epic:** Data Harmonization
- **Description:** As a fleet manager, I want to import data from users registered in an external platform to Thumbeo Corporate, in order to centralize user information in a single system.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager imports users to Thumbeo Corporate.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the fleet page
      And select the "import users" button

  * **Then** the system communicates with the external system to import users present in the system itself.

- **Dependencies:** None
- **Priority:** Must Have

DH03

- **Epic:** Data Harmonization
- **Description:** As a fleet manager, I want to import data from ride offers and requests registered in an external platform to Thumbeo Corporate, in order to centralize trip information in a single system.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager imports ride offers and requests to Thumbeo Corporate.
    * **Given** that I am authenticated
      And have fleet manager permissions
    * **When** I access the fleet page
      And select the "import ride offers and requests" button
    * **Then** the system communicates with the external system to import ride offers and requests present in the system itself.
- **Dependencies:** None
- **Priority:** Must Have

DH04

- **Epic:** Data Harmonization
- **Description:** As a fleet manager, I want to import data from driving pattern data registered in an external platform to Thumbeo Corporate, in order to centralize driving pattern information in a single system.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager imports driving pattern data to Thumbeo Corporate.
    * **Given** that I am authenticated
      And have fleet manager permissions
    * **When** I access the fleet page
      And select the "import driving patterns" button
    * **Then** the system communicates with the external system to import driving pattern data present in the system itself.
- **Dependencies:** None
- **Priority:** Must Have

DH05

- **Epic:** Data Harmonization
- **Description:** As a fleet manager, I want to import data from OBD readers to Thumbeo Corporate, in order to centralize the gathered information in a single system.

- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager imports driving pattern data to Thumbeo Corporate.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the fleet page
    - **Then** the information collected from OBD readers should be available in each vehicle and for each ride.
- **Dependencies:** None
- **Priority:** Must Have

Statistics Collection

SC01

- **Epic:** Statistics Collection
- **Description:** As a fleet manager, I want to see statistics gathered concerning the fuel consumption of the vehicles in my fleet, to analyze the cost and spending of fuel of the vehicles and of the fleet.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager sees statistical data regarding fuel consumption.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the statistics page
    - **Then** the statistical data gathered about fuel consumption of the fleet and of each vehicle should be available.
- **Dependencies:** None
- **Priority:** Must Have

SC02

- **Epic:** Statistics Collection
- **Description:** As a fleet manager, I want to see statistics gathered concerning the utilization of vehicles in the fleet, to analyze if said vehicles have their intended utilization.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager sees statistical data regarding vehicle utilization.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the statistics page
    - **Then** the statistical data gathered about utilization of the vehicles of the fleet should be available.
- **Dependencies:** None
- **Priority:** Must Have

SC03

- **Epic:** Statistics Collection
- **Description:** As a fleet manager, I want to see statistics gathered related to the availability of the vehicles in the fleet, to analyze if the vehicles are available when they are needed.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager sees statistical data regarding vehicle availability.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the statistics page
    - **Then** the statistical data gathered about the availability of the vehicles of my fleet should be available.
- **Dependencies:** None
- **Priority:** Must Have

Vehicle Management

VM01

- **Epic:** Vehicle Management
- **Description:** As a fleet manager, I want to add a vehicle to my fleet so that I can manage it using the web application.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager opens the page to add a vehicle
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the fleet page
      And select the "new vehicle" button
    - **Then** I see a form to add a new vehicle
  - **Scenario:** Fleet Manager provides the required information to add a vehicle to the fleet
    - **Given** that I am authenticated
      And have fleet manager permissions
      And I am in the "add vehicle" form
    - **When** I fill the required fields with valid information
    - **Then** I see a success message
      And the vehicle now appears in my fleet
  - **Scenario:** Fleet Manager does not provide the required information to add a vehicle to the fleet
    - **Given** that I am authenticated
      And have fleet manager permissions
      And I am in the "add vehicle" form
    - **When** I fill the required fields with invalid information or leave required fields blank
    - **Then** I see an error message
- **Dependencies:** None
- **Priority:** Must Have

VM02

- **Epic:** Vehicle Management
- **Description:** As a fleet manager, I want to edit the information about a specific vehicle so that I can update the data kept on it.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager open the page to update a vehicle
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I access the fleet page
      And select the "edit vehicle" button in a specific vehicle
    - **Then** I see a form to edit the information of a specific vehicle
  - **Scenario:** Fleet Manager provides the required information to edit a vehicle in the fleet
    - **Given** that I am authenticated
      And have fleet manager permissions
      And I am in the "edit vehicle" page
    - **When** I fill the required fields with valid information
    - **Then** I see a success message
      And the changes should be reflected in the fleet information
  - **Scenario:** Fleet Manager does not provide the required information to edit a vehicle in the fleet
    - **Given** that I am authenticated
      And have fleet manager permissions
      And I am in the "edit vehicle" page
    - **When** I fill the required fields with invalid information or leave the fields blank
    - **Then** I see an error message
- **Dependencies:** VM01
- **Priority:** Must Have

VM03

- **Epic:** Vehicle Management
- **Description:** As a fleet manager, I want to see all my vehicles so that I can have a global view of my fleet.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager opens the page to see the fleet
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** I select "My Fleet" button
    - **Then** I am redirected to the page containing all the vehicles present in my fleet
- **Dependencies:** VM01
- **Priority:** Must Have

VM04

- **Epic:** Vehicle Management
- **Description:** As a fleet manager, I want to archive a vehicle of my fleet so that I can tell the system that the vehicle in question is no longer operational in my fleet.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager archives a vehicle
    - **Given** that I am authenticated
      And have fleet manager permissions
      And I am in "My fleet" page
    - **When** I select the "Archive" option in a specific vehicle
    - **Then** I see a success message
      And the vehicle is removed from my fleet
- **Dependencies:** VM01
- **Priority:** Must Have

Permission Management

PM01

- **Epic:** Permission Management
- **Description:** As a system administrator, I want to give fleet management permissions to a user, so that the user can act as a fleet manager inside the web application.
- **Acceptance Criteria:**
  - **Scenario:** System administrator grants fleet manager permissions to a user
    - **Given** that I am authenticated
      And have system administrator permissions
    - **When** I select a specific user
      And grant him permissions to act as a fleet manager inside the application
    - **Then** I see a success message
      And the user can now perform actions reserved to fleet managers inside the application.
- **Dependencies:** None
- **Priority:** Must Have

PM02

- **Epic:** Permission Management
- **Description:** As a system administrator, I want to remove fleet management permissions from a user, so that the user can no longer act as a fleet manager inside the web application.
- **Acceptance Criteria:**
  - **Scenario:** System administrator revokes fleet manager permissions from a fleet manager.
    - **Given** that I am authenticated

And have system administrator permissions
    * **When** I select a specific fleet manager
      And revoke his fleet manager permissions
    * **Then** I see a success message
      And the user can no longer perform actions restricted to fleet managers inside the application.
- **Dependencies:** PM01
- **Priority:** Must Have

PM03

- **Epic:** Permission Management
- **Description:** As a fleet manager, I want to give driver permissions to a user, so that the user can act as a driver inside the web application.
- **Acceptance Criteria:**
  - **Scenario:** Fleet manager grants driver permissions to a regular user
    * **Given** that I am authenticated
      And I have fleet manager permissions
    * **When** I select a specific user
      And grant the user permissions to become a driver in my fleet
    * **Then** I see a success message
      And the user becomes a driver in my fleet.
- **Dependencies:** PM01
- **Priority:** Must Have

PM04

- **Epic:** Permission Management
- **Description:** As a fleet manager, I want to remove driver permissions from a user, so that the user can no longer act as a driver inside the web application.
- **Acceptance Criteria:**
  - **Scenario:** Fleet manager removes driver permissions from a driver.
    * **Given** that I am authenticated
      And I have fleet manager permissions
    * **When** I select a specific driver in my fleet
      And remove his permissions to drive vehicles in my fleet
    * **Then** the driver no longer belongs to my fleet
      And the driver can no longer use the vehicles in my fleet.
- **Dependencies:** PM01, PM03
- **Priority:** Must Have

PM05

- **Epic:** Permission Management
- **Description:** As a fleet manager, I want to give a driver permissions to drive one of my vehicles, so that I can provide him with a means of transportation.

- **Acceptance Criteria:**
  - **Scenario:** Fleet manager grants access to a driver to use a specific vehicle.
    - **Given** that I am authenticated
      And I have fleet manager permissions
    - **When** I select one of the vehicles in my fleet
      And grant access to a driver to use that specific vehicle
    - **Then** I see a success message
      And the driver receives a notification about having been granted access to the vehicle.
- **Dependencies:** PM01, PM03, VM01
- **Priority:** Must Have

PM06

- **Epic:** Permission Management
- **Description:** As a fleet manager, I want to remove a driver's permissions to use one of my vehicles, so that I can stop him from driving a specific vehicle.
- **Acceptance Criteria:**
  - **Scenario:** Fleet manager revokes driver access to a vehicle.
    - **Given** that I am authenticated
      And I have fleet manager permissions
    - **When** I select one of the vehicles in my fleet
      And revoke the access a specific driver of that vehicle
    - **Then** I seed a success message
      And the driver receives a notification about having his access revoked to the vehicle
- **Dependencies:** PM01, PM03, PM05, VM01
- **Priority:** Must Have

Vehicle Evaluation

VE01

- **Epic:** Vehicle Evaluation
- **Description:** As a driver, I want to make an evaluation of the vehicle's condition after a trip, so that I can notify the fleet manager about occurrences during said trip.
- **Acceptance Criteria:**
  - **Scenario:** Driver submits a trip report.
    - **Given** that I am authenticated
      And I have driver permissions
      And I have just finished a trip
    - **When** I fill the trip report with departure and arrival date and time, stops made along the way and details about certain vehicle components.
    - **Then** I see a success message
      And the report is sent to the corresponding fleet manager.
- **Dependencies:** PM01, PM03, PM05, VM01
- **Priority:** Must Have

VE02

- **Epic:** Vehicle Evaluation
- **Description:** As a driver, I want to report a fault in vehicle that I am currently driving, so that I can inform the fleet manager.
- **Acceptance Criteria:**
  - **Scenario:** Driver submits fault report
    - **Given** that I am authenticated
      And I have driver permissions
      And the vehicle I am driving presents a fault
    - **When** I submit the fault report with the location of the fault and a description of the fault
    - **Then** I see a success message
      And the report is sent to the corresponding fleet manager.
- **Dependencies:** PM01, PM03, PM05, VM01
- **Priority:** Must Have

Fleet Manager Notifications

FMN01

- **Epic:** Fleet Manager Notifications
- **Description:** As a fleet manager, I want to be notified when one of my drivers starts a trip, so that I can be up to date with my vehicles state.
- **Acceptance Criteria:**
  - **Scenario:** Fleet manager receives a notification about the start of a trip by one of the fleet's drivers
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** one of my fleet's drivers initiates a trip
    - **Then** I will receive a notification about the starting point, destination point and estimated duration of the travel.
- **Dependencies:** None
- **Priority:** Nice to Have

FMN02

- **Epic:** Fleet Manager Notifications
- **Description:** As a fleet manager, I want to be notified when one of my drivers finishes a trip, so that I can be up to date with my vehicles state.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager receives a notification about the ending a trip.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** a driver finishes a trip
    - **Then** I will receive a notification with a report of the trip.

- **Dependencies:** PM01, PM03, PM05, VM01, VE01
- **Priority:** Nice to Have

FMN03

- **Epic:** Fleet Manager Notifications
- **Description:** As a fleet manager, I want to be notified when one of my drivers reports a vehicle fault, so that I can properly plan the maintenance for the reported fault.
- **Acceptance Criteria:**
  - **Scenario:** Fleet Manager receives a notification about a vehicle fault.
    - **Given** that I am authenticated
      And have fleet manager permissions
    - **When** a driver reports a vehicle fault
    - **Then** I will receive a notification with the location where the fault occurred and the details of the fault itself.
- **Dependencies:** PM01, PM03, PM05, VM01, VE02
- **Priority:** Should Have

Driver Notifications

DN01

- **Epic:** Driver Notifications
- **Description:** As a driver, I want to be notified when I am assigned a new vehicle, so that I can be up to date with which vehicles I can use.
- **Acceptance Criteria:**
  - **Scenario:** Driver receives a notification about a new vehicle assigned to himself
    - **Given** that I am authenticated
      And have driver permissions
    - **When** the fleet manager assigns me with a new vehicle
    - **Then** I will receive a notification with the make and model of the vehicle and also the license plate number.
- **Dependencies:** PM01, PM03, PM05
- **Priority:** Nice to Have

DN02

- **Epic:** Driver Notifications
- **Description:** As a driver, I want to be notified when my access to a vehicle has been removed, so that I can be up to date with which vehicles I can use.
- **Acceptance Criteria:**
  - **Scenario:** Driver receives a notification about having the access to a specific vehicle revoked.
    - **Given** that I am authenticated

        And have driver permissions
    - ∗ **When** the fleet manager revokes my access to a vehicle
    - ∗ **Then** I will receive a notification with the make and the model of the vehicle and also the license plate number.

- **Dependencies:** PM01, PM03, PM05, PM06
- **Priority:** Nice to Have

DN03

- **Epic:** Driver Notifications
- **Description:** As a driver, I want to be notified when one of the vehicles assigned to me has been declared unavailable, so that I can be up to date with which vehicles I can use.
- **Acceptance Criteria:**
  - **Scenario:** Driver receives a notification about having a vehicle assigned to him archived.
    - ∗ **Given** that I am authenticated
      And have driver permissions
    - ∗ **When** the fleet manager archives a vehicle that I am assigned to
    - ∗ **Then** I will receive a notification with the make, model and license plate number of the vehicle
- **Dependencies:** PM01, PM03, PM05, VM01, VM04
- **Priority:** Nice to Have

Appendix C

Project Development — Fleet Management Models

Entity Relationship — Low Level Diagram

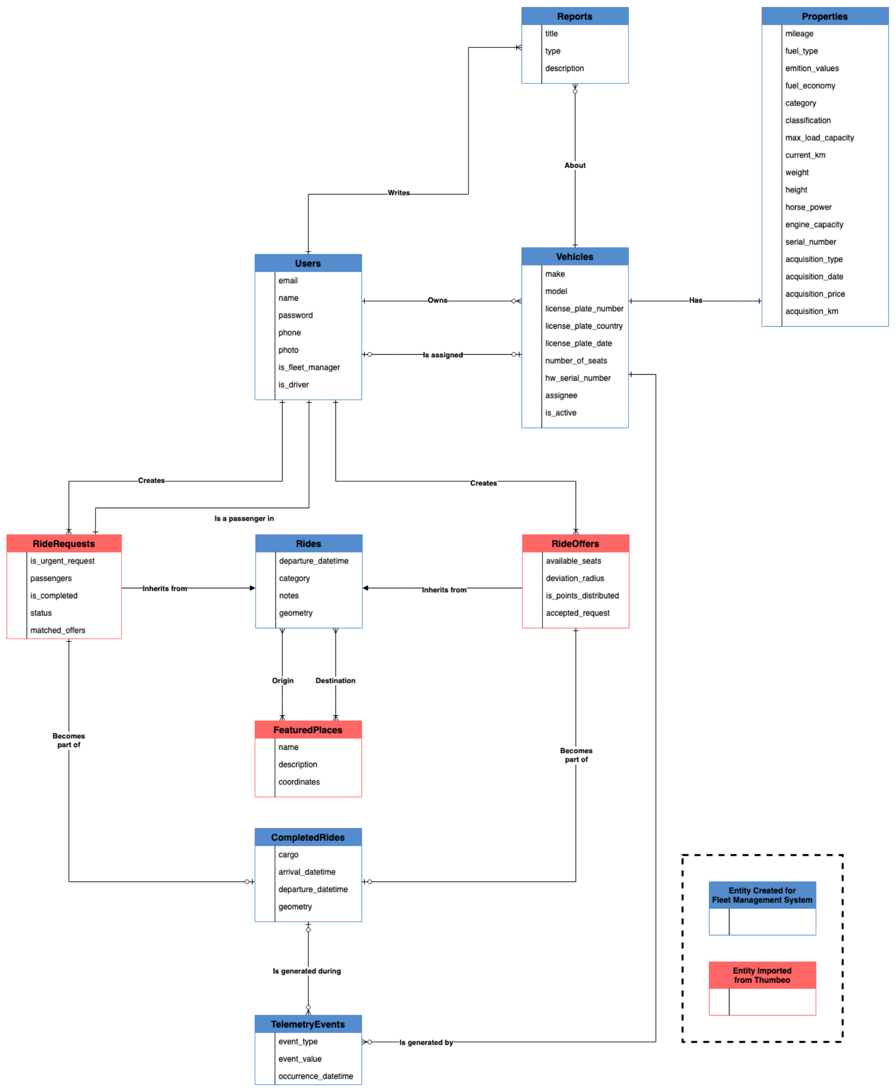


Figure C.1: Fleet Management System — Low Level Entity Relationship Diagram

User

The following model pertaining to the *User Profile* was created to extend Django's default *User* model in order to enable additional user information to be stored. The default *User* model is used in the authentication process of the Fleet Management System.

User Profile

- **Name** — Name of the user register;
- **Job** — Profession the register user performs in the company;
- **Phone Number** — Contact of the user registered;
- **Profile Photo** — Photo submitted by the user;
- **Password** — Encrypted password of the user;
- **Is Driver** — True if the user has driver permissions, else is false;
- **Is Fleet Manager** — True if the user has fleet manager permissions, else is false.

Vehicle

Two distinct models where created in the Vehicle component. The first model, *Vehicle*, stores mandatory vehicle information that is necessary to maintain the registration of a vehicle, and the second model, *Properties*, stores additional vehicle information that fleet managers might want to keep track of.

Vehicle

- **Make** — Vehicle manufacturer name;
- **Model** — Vehicle model name;
- **License Plate Number** — License plate identification number;
- **License Plate Country** — License plate country of legalization;
- **License Plate Date** — License plate date of legalization;
- **Number of Seats** — Number of seats available in the vehicle;
- **Properties** — Relationship with the following *Properties* model;
- **Assignee** — Entity assigned to the vehicle;
- **Users** — Users responsible for the vehicle.
- **Hardware Serial Number** — Serial Number of the system that is gathering data for this vehicle;
- **Is Active** — True if the vehicle is currently being used.

Properties

- **Category** — One of the following options:
  - Light Duty Vehicle Carrying Passengers;

  - Heavy Duty Vehicle Carrying Passengers;
  - Light Duty Vehicle Carrying Goods;
  - Heavy Duty Vehicle Carrying Goods;
  - Two and Three Wheel Vehicle or Quadricycle;
  - Agricultural and Forestry Tractor and Trailer.
- **Classification[52]** — One of the following options:
  - Class 1 — Vehicles with height inferior to 1.1 meters;
  - Class 2 — Vehicles with two axis and height superior to 1.1 meters;
  - Class 3 — Vehicles with three axis and height superior 1.1 meters;
  - Class 4 — Vehicles with four axis and height superior 1.1 meters.
- **Mileage** — Fuel economy tracked in liters per 100 kilometers;
- **Vehicle Fuel** — One of the following options:
  - Petrol;
  - Diesel;
  - Electric;
  - LPG;
  - Hybrid.
- **CO2 Emissions** — Carbon Dioxide emissions measured in grams per kilometers;
- **Max Load Capacity** — Max Load Capacity as stated in the vehicle registration documents;
- **Current kilometers** — Kilometers currently registered in the vehicle;
- **Weight** — Weight as stated in the vehicle registration documents;
- **Height** — Height as stated in the vehicle registration documents;
- **Horse Power** — Horse Power as stated in the vehicle registration documents;
- **Engine Capacity** — Engine Capacity as stated in the vehicle registration documents;
- **Vehicle Identification Number** — Unique code used to identify the vehicle;
- **Acquisition Type** — One of the following options:
  - Rented;
  - Leased;
  - Purchased;
  - Other.
- **Acquisition Date** — Date at which the vehicle was acquired;
- **Acquisition Price** — Price at which the vehicle was acquired;
- **Acquisition kilometers** — Kilometers registered in the vehicle when it was acquired.

Ride

To register rides, four different models were developed: *Ride* was created to be extended by the following models. *Ride Offer* is the model that shapes ride offers created by users in the fleet management system or the model followed by ride offers when imported from external systems. *Ride Request* is the model used by users to create ride requests in the platform or the model followed by externally imported data. *Completed Ride* is the model used to create the report of a finalized trip, making use of the *Ride Offer* and *Ride Request* that generate it. The *Featured Place* model is used by fleet managers to create a place frequently ridden to and from.

Featured Place

- **Name** — Name of the place frequently used;
- **Description** — Description of the place;
- **Coordinates** — Set of coordinates describing the location of the place.

Ride

- **User** — Person creating the ride request or offer;
- **Origin** — Origin of the ride;
- **Destination** — Destination of the ride;
- **Departure Datetime** — Datetime that the trip is suppose to start in;
- **Category** — One of the following categories:
  - People — If the ride is suppose to carry only people;
  - Cargo — If the ride is suppose to carry only cargo;
  - All — If the ride is going to take both cargo and people.
- **Notes** — Miscellaneous information provided by the user creating the ride;
- **Geometry** — Geographical geometry of the route used in the ride.

Ride Offer

- **Ride** — Relationship with *Ride* model;
- **Available Seats** — Number of currently available seats in the ride offer;
- **Deviation Radius** — Deviation radius allowed by the offerer in the ride;
- **Accepted Request** — Relationship with the ride request accepted by this offer.

Ride Request

- **Status** — One of the following options:
  - Pending;
  - Standby;
  - Accepted;
  - Rejected;
  - Canceled.
- **Is Urgent Request** — True if the ride request is urgent;
- **Passengers** — Passengers of the ride request;
- **Matched Offers** — The offers that matched this ride request.

Completed Ride

- **Ride Request** — Relationship with the ride request;
- **Ride Offer** — Relationship with the ride offer;
- **Departure Datetime** — Datetime at which the ride departed;
- **Arrival Datetime** — Datetime at which the ride arrived;
- **Cargo** — Cargo transported during the ride;
- **Geometry** — Ride geometry collected for the ride;
- **Passengers** — Passengers transported during the ride.

Reports

In the development of the reporting features the following model is used to store the information concerning the reports created by users.

Reports

- **User** — User submitting the report;
- **Vehicle** — Vehicle about which the report is about;
- **Title** — Title of the report;
- **Description** — Description of the report;
- **Report Type** — One of the following options:
  - Fault Report;
  - Post-Trip Report;

Telemetry Event

The data generated by the OBD Readers is stored by the following model. Telemetry Event was chosen as a name for the model rather than OBD Event given that OBD is a standard which could be renamed, and telematics is the concept of gathering information from vehicles.

Telemetry Event

- **Vehicle** — Vehicle which generated the event;
- **Completed Ride** — Ride at which the event was generated;
- **Event Type** — Type of event generated;
- **Event Value** — Value of the event generated;
- **Occurrence Datetime** — Date and time at which the event was generated.

Appendix D

Project Development — Unit Tests

Data Harmonization System

ID	Component	Description	Passed
U01	User Data	Receiving user data that is valid should transform it into a user model	Yes
U02	User Data	Receiving user data that is invalid should discard it	Yes
U03	User Data	After a user model is created it should be written into the database	Yes
V01	Vehicle Data	Receiving vehicle data that is valid should transform it into a vehicle model	Yes
V02	Vehicle Data	Receiving vehicle data that is invalid should discard it	Yes
V03	Vehicle Data	After a vehicle model is created it should be written into the database	Yes
R01	Rides Data	Receiving ride request data that is valid should transform it into a Ride Request model	Yes
R02	Rides Data	Receiving ride offer data that is valid should transform it into a Ride Offer model	Yes
R03	Rides Data	Receiving ride request data that is invalid should not discard it	Yes
R04	Rides Data	Receiving ride offer that is invalid should discard it	Yes
R05	Rides Data	After a ride request model is created it should be written into the database	Yes
R06	Rides Data	After a ride offer model is created it should be written into the database	Yes
TD01	Telemetry Data	Receiving OBD data that is valid should transform it into a Telemetry Event	Yes
TD02	Telemetry Data	Receiving OBD data that is invalid should discard it	Yes
TD03	Telemetry Data	After a telemetry event is created it should be written into the database	Yes

Table D.1: Data Harmonization System — Unit Tests

Fleet Management System

ID	Component	Description	Passed
U01	Users Component	Creating a user with all fields provided should be accepted	Yes
U02	Users Component	Creating a user with required fields provided should be accepted	Yes
U03	Users Component	Creating a user without fields provided should be rejected	Yes
U04	Users Component	Fetching a user profile should return user and profile attributes	Yes
U05	Users Component	Updating a user profile with new attributes should be accepted	Yes
U06	Users Component	Updating a user and profile with new attributes should be accepted	Yes
U07	Users Component	Updating a user email and password should be accepted	Yes
U08	Users Component	Deleting a user should erase the information from the system	Yes

Table D.2: Fleet Management System — User Component Unit Tests

ID	Component	Description	Passed
V01	Vehicles Component	Creating a vehicle with all fields provided should be accepted	Yes
V02	Vehicles Component	Creating a vehicle with required fields provided should be accepted	Yes
V03	Vehicles Component	Creating a vehicle without any fields provided should be rejected	Yes
V04	Vehicles Component	Fetching a vehicle that is not created should return an empty result	Yes
V05	Vehicles Component	Fetching a vehicle that exists should return its information	Yes
V06	Vehicles Component	Updating a vehicle with new attributes should be accepted	Yes
V07	Vehicles Component	Updating a vehicle with new required attributes should be accepted	Yes
V08	Vehicles Component	Updating a vehicle with empty attributes should be rejected	Yes
V09	Vehicles Component	Deleting a vehicle should erase the information from the system	Yes
V10	Vehicles Component	Deleting a vehicle that does not exist should not crash the system	Yes

Table D.3: Fleet Management System — Vehicle Component Unit Tests

ID	Component	Description	Passed
R01	Rides Component	Creating a ride request with all fields provided should be accepted	Yes
R02	Rides Component	Creating a ride request with required fields provided should be accepted	Yes
R03	Rides Component	Creating a ride request without any fields provided should be rejected	Yes
R04	Rides Component	Fetching a ride request that is not created should return an empty result	Yes
R05	Rides Component	Fetching a ride request that exists should return its information	Yes
R06	Rides Component	Updating a ride request with new attributes should be accepted	Yes
R07	Rides Component	Deleting a ride request should remove its information from the system	Yes
R08	Rides Component	Deleting a ride request that does not exist should not crash the system	Yes
R09	Rides Component	Creating a ride offer with all fields provided should be accepted	Yes
R10	Rides Component	Creating a ride offer with required fields provided should be accepted	Yes
R11	Rides Component	Creating a ride offer without any fields provided should be rejected	Yes
R12	Rides Component	Fetching a ride offer that is not created should return an empty result	Yes
R13	Rides Component	Fetching a ride offer that exists should return its information	Yes
R14	Rides Component	Updating a ride offer with new attributes should be accepted	Yes
R15	Rides Component	Deleting a ride offer should remove its information from the system	Yes
R16	Rides Component	Deleting a ride offer that does not exist should not crash the system	Yes

Table D.4: Fleet Management System — Rides Component Unit Tests

ID	Component	Description	Passed
F01	Statistics Component	Fetching the average fuel consumption for a specific vehicle should return the intended result	Yes
F02	Statistics Component	Fetching the average fuel consumption for all vehicle should return the intended result for all vehicles	Yes

Table D.5: Fleet Management System — Statistics Component Unit Tests