Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Final Report

An Intelligent Bike-Sharing Rebalancing System

Masters in Informatics Engineering

Author:

Diogo Manuel Matos Braga Lopes
braga@student.dei.uc.pt

Advisors:

Dr. Carlos Bento
DEI | FCTUC

Eng. Ricardo Machado
Ubiwhere

September 2015

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Final Report

JURIES:

Dr. Henrique Madeira

Dr. Luís Miguel Macedo

September 2015

Acknowledgements

I would like to express my gratitude to both my supervisors, Carlos Bento and Ricardo Machado, for all their support and time spent during the entire internship.

My sincere thanks to *Ubiwhere* for creating this opportunity and allowing me to work with such a great and committed team that taught me a lot.

A big thanks to all my friends that always been there for me independently of the situation, you surely will not be forgotten.

Finally, this report represents the end of one-year internship that wouldn't be possible without the full support of my parents, my sister and all my family. A special thanks to them.

Abstract

Bike-sharing offers users bikes-on-demand at an affordable price, improving urban mobility. It is currently the fastest growing way of transportation in the world, with an aggregated growth rate of around 45% since 2007. Nowadays there are more than 800 cities with bike-sharing programs.

Currently, one of the main problems and challenges in the management of Bike-Sharing Systems is to assure that users will be able to find available bicycles and parking slots in a station independently of the time. However, users behaviour causes bicycles to be asymmetrically distributed. Therefore a Rebalancing System is needed to maintain the adequate number of bikes at each station, in order to satisfy the demand. Rebalancing is a costly operation in terms of logistics and its inefficiency represents the main cause of dissatisfaction within customers. The design of algorithms that help and advise operators to redistribute bicycles accurately and efficiently through the cities is a very important step in terms of sustainability of these systems.

In this thesis an Intelligent Bike-Sharing Rebalancing System will be developed. This system represents an additional value to the project *Smart BikeEmotion* developed by *Ubiwhere*.

Keywords:

Bike-Sharing, Rebalancing Problem, Intelligent Rebalancing Algorithm, Smart BikeEmotion, Optimal Fleet Repositioning

Resumo

Os sistemas de *Bike-sharing* disponibilizam o aluguer de bicicletas aos utilizadores a um preço acessível, melhorando assim a mobilidade urbana. Atualmente este é o meio de transporte que mais rapidamente se está a expandir pelo mundo, com um crescimento agregado de 45% desde 2007. Hoje em dia existem mais de 800 cidades com programas de *Bike-sharing*.

Um dos principais desafios na gestão de sistemas de *Bike-sharing* é assegurar que os utilizadores não terão problemas em encontrar bicicletas e lugares de estacionamento em qualquer estação do sistema, independentemente do momento do dia. Porém, o comportamento dos utilizadores faz com que as bicicletas tenham uma distribuição assimétrica. Portanto é necessário um sistema de reposicionamento de bicicletas de modo a manter o número adequado de bicicletas em cada estação, para a satisfazer a procura por parte dos utilizadores. O reposicionamento de bicicletas é uma operação dispendiosa em termos de logística e sua ineficiência representa a principal causa de insatisfação nos clientes. O desenvolvimento de algoritmos que ajudem e aconselhem os condutores das carrinhas que fazem o reposicionamento das bicicletas a equilibrar as estações de forma eficaz e precisa é um passo muito importante em termos de sustentabilidade destes sistemas.

Nesta dissertação será apresentado o tema *An Intelligent Bike-Sharing Rebalancing System*. Este sistema representa um valor adicional para o projeto *Smart BikeEmotion* desenvolvido pela empresa *Ubiwhere*.

Palavras-chave:

Bike-Sharing, Reposicionamento de Bicicletas, Intelligent Rebalancing Algorithm, Smart BikeEmotion

Table of Contents

List of Tables

List of Figures

Acronyms

API Application Programming Interface

BSS Bike-Sharing System

CARET Classification and Regression Training

CQL Cassandra Query Language

CPU Central Processing Unit

CSV Comma Separated Values

EWGT EURO Working Group on Transportation

GBM Gradient Boosting Machines

GNU General Public License

LNS Large Neighborhood Search

TSP Traveling Salesman Problem

VRP Vehicle Routing Problem

RAM Random Access Memory

REST Representational State Transfer

RMSLE Root Mean Squared Logarithmic Error

SQL Structured Query Language

XML Extensible Markup Language

JSON JavaScript Object Notation

1 Introduction

In order to get the Master's Degree in Informatics Engineering from the Faculty of Science and Technology of the University of Coimbra, an internship takes place in the company *Ubiwhere* with the duration of one year. This Report documents the work done in the several phases of this internship. The internhip work was supervised by Carlos Bento, PhD professor at the Department of Informatics Engineering of the University of Coimbra, and Engineer Ricardo Machado, Project Manager at *Ubiwhere*.

This chapter is divided into 5 sections and provides an overall perspective of the project. Sections 1.1 and 1.2 start by clarifying what are the main reasons for the creation of this project and briefly analyse the state of the art of Bike-Sharing. Sections 1.3 and 1.4 give an overview of the company where this internship is integrated and explain what are the main objectives of this project. Section 1.5 presents a description of the global structure of the document.

1.1 Scope

With the continuous city population growth many challenges also emerged, such as high traffic levels, energy consumption, noise and pollution. Consequently, public authorities began to investigate new forms of mobility that could handle the continuous flow of people, but that did not had a negative impact on the environment. Bike-Sharing is an ecological transportation mode that provides bikes-on-demand in order to improve daily urban mobility.

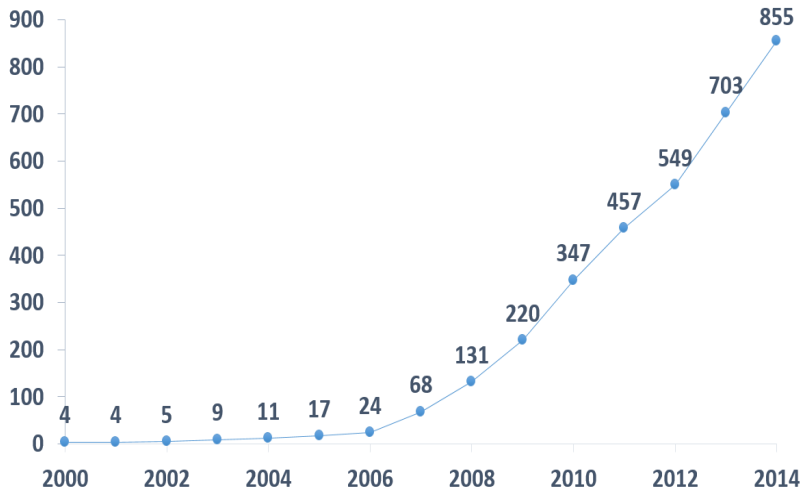

Figure 1: Number of cities with an Bike-Sharing System [2]

Over the years BSSs have become a popular form of transportation within cities all round the world thus promoting active types of transport, decreasing the dependence on automobiles and reducing greenhouse gas emissions. Bike-Sharing is currently the fastest growing way of transportation in the world, with an aggregate growth rate of around 45% since 2007. In the end of the year 2014 there were more than 800 cities with a bike-sharing program, as shown in the FIG. 1. [2]

In FIG. 2 is the distribution of the world BSSs by continent. The leader in number of BSSs programs is Europe with 414 programs (62%), followed by Asia with 164 programs (25%), North America with 50 programs (8%), South America with 22 programs (4%) and, finally, Africa and Australia have less than 1% of the world's BSSs, [2]. In Washington DC BSS (Capital Bikeshare) each member saves about 800 dollars on transportation costs annually, while each trip generates approximately 7 dollars for the local economy. [11]

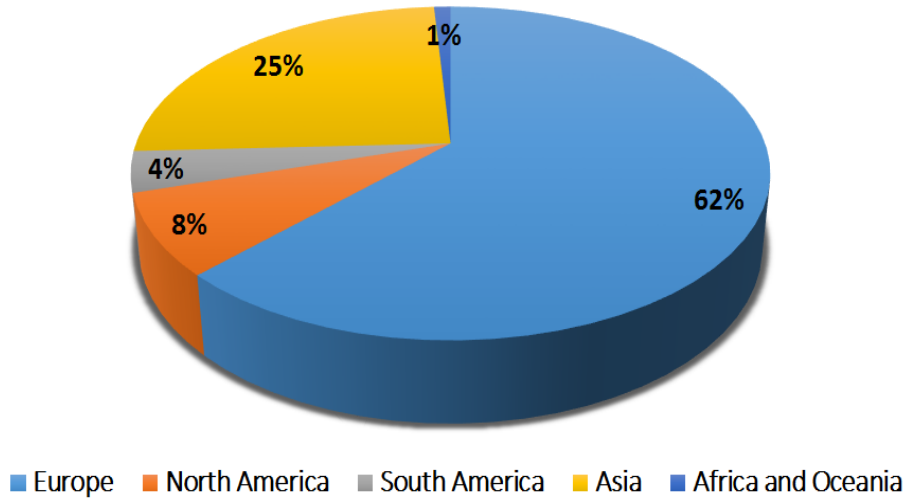

Figure 2: World Bike-Sharing Programs by Continent [2]

In BSSs, users can pick up and return bikes in designated bike-sharing stations with a finite number of docks. Unfortunately, users behaviour causes bicycles to be asymmetrically distributed, so some stations are full while others are empty. "The system equilibrium is often characterized by unacceptably low availability of bikes or open docks, for pick-ups or returns respectively" [12].

1.2 Motivation

Nowadays, one of the main problems in the management of BSSs is to ensure that people will be able to find available bicycles and to park them in any station, at any time. Therefore, a regulation system is needed for in order to maintain the optimal number of bikes at each station in order to fulfil the demand.

Rebalancing is a very costly operation in terms of logistics and its inefficiency represents one of the main causes of dissatisfaction within customers. Algorithms that manage the rebalancing of bicycles in BSSs help customers and operators to know where to drop or pick bicycles accurately and efficiently, so they are an important process in terms of sustainability from the perspective of both the user and the business stakeholder.

The innovative use of intelligent techniques in this particular matter can have some advantages when compared to real time algorithms used in some known Bike-Sharing Systems. One of those advantages is that, if the prediction is successful, it allows a preventive approach instead of a reactive approach. Rather than making corrections after stations get empty or full, it actually tries to anticipate and prevent that from happening. Another advantage is that intelligent algorithms evolve together with user needs, having as ultimate goal to get an optimal fleet repositioning.

1.3 Context

This internship takes place in *Ubiwhere*, a software company founded in Aveiro on 2007. In 2010 it achieved the prestigious ISO 9001 certification of the global quality management system and the NP 4457 certification.

On 2012 *Nesta* Magazine named *Ubiwhere* one of the Portugal's 50 Hottest Startups and *Ubiwhere* achieved more than one million Euros turnover.

On 2013 *Ubiwhere* won the European Seal of e-Excellence, became the youngest Portuguese company rated at CMMI (Capability Maturity Model Integration) with level 2 for development and also launched *BikeEmotion*, a fourth generation bike-sharing system integrated in the project *Tice.mobilidade* that invests in projects incorporated in urban mobility.

The Project being done on this thesis is the continuation of *BikeEmotion*. This upgraded version of *BikeEmotion* is called *Smart BikeEmotion* and has new features including the use of electric bicycles, a new intelligent rebalancing system and the integration of wearables.

1.4 Goals

From a professional angle, the goal is to consolidate the knowledge about Python, Django, Java, machine learning and other technologies, but above all to gain experience working in a corporate environment where team work, responsibility and commitment are essential to succeed.

Given the broad scope of the project, the objectives were classified as either Primary or Secondary. Primary objectives are the main focus of the project and where highest effort will be spent and possess a higher degree of interest from a scientific point of view. Secondary objectives, on the other hand, are meant to support the Primary objectives or improve the current state of the prototype.

The main goal of this thesis is to contribute for the enrichment of the *Smart BikeEmotion* project during this internship, developing an intelligent algorithm for the rebalancing of bicycles in BSSs. In order for this goal to be successful some functional requisites were defined:

1. [**Primary**] - *Prediction of the docks usage:* try to predict, through the help of an intelligent algorithm, the approximate number of bicycles in each dock.
2. [**Primary**] - *Dynamic Bike Redistribution:* instruct the employees how to distribute the bikes and the best sequence to do it.
3. [**Primary**] - *Develop an API:* develop an API to connect to other internal modules in *Smart BikeEmotion* system.
4. [**Primary**] - *Include location and time as a variable:* during the calibration of the system take in account the different locations of the stations and the current time.
5. [**Secondary**] - *Diagnose the most problematic stations:* do an overall evaluation of the stations hourly and analyse what are the stations that have an excess of bicycles or lack of bicycles.
6. [**Secondary**] - *Include weather as a variable:* during the calibration of the system take in account the current weather.
7. [**Secondary**] - *Include city topology as a variable:* while the system is being calibrated take in account the topology of each station.

1.5 Document structure

In this section a brief description of the structure of the document is presented and also what is discussed on each of its chapters.

- **Chapter 1** - Introduction to the internship, the context and motivation behind it, its scope and finally its main objectives.
- **Chapter 2** - State of the art analysis and comparison between the existing software and project being developed in this thesis.
- **Chapter 3** - Presents the software engineering methodologies followed to manage and control the development of the product and shows the planning of this project in form of *Gantt* diagrams and a risk analysis is performed.
- **Chapter 4** - Describes the main features supported by the Rebalancing Bike-Sharing Intelligent System specified through user stories.
- **Chapter 5** - Defines the overall architecture of the Smart BikeEmotion and the modules that are going to be developed.
- **Chapter 6** - Describes the modules developed on this internship with a higher level of detail.
- **Chapter 7** - Specifies the tests that were made to guarantee the quality of the software developed.
- **Chapter 8** - Future work and final remarks regarding all work done in this dissertation.

2 State of the art

In this chapter several possible solutions that allow the rebalancing of bicycles through intelligent systems will be analysed. Section 2.1 presents current solutions used by BSSs, section 2.2 analyses different techniques regarding intelligent prediction of the docks future states and section 2.3 analyses several tools and software regarding the redistribution of bicycles in BSSs.

2.1 Current Solutions

Presently the most important BSSs use a zone redistribution system with dedicated vehicles to try to ensure availability of bikes and spaces. To do this management, these systems use a central control room that shows when docks are empty or full in real-time , instructing employees to deliver and collect bikes around the city and distribute them as needed. This is done manually and requires a lot of resources that are expensive for the companies in charge of the BSS, although some stations fill and empty naturally without ever reaching their limits. Some examples of BSSs worth being mentioned are *Bicing* (Barcelona, Spain), *Divvy* (Chicago, United States), *Barclays Cycle Hire* (London, England), *CitiBike* (New York, United States), *Vélib* (Paris, France) and *Capital Bikeshare* (Washington, United States).

2.1.1 Barcelona (Bicing)

Bicing was launched in May 2007 with 750 bikes and 50 stations located near Metro stations and major parking areas. The system has over 170 000 subscribers. [3]

During high load hours the bikes are moved from the busiest stations to the emptiest, using trailers pulled by electric vehicles with zero carbon dioxide emissions and *Ford Transit* vans with specially-designed tail ramps. The vehicles used for doing the redistribution of bikes are showed in FIG. 3.

Barcelona topography causes a large number of vans to be required in order to re-distribute the bicycles. Ten vans re-distribute bicycles, the same number as in Paris but for less than a quarter the number of bicycles. They also try to incentive the user to help in rebalancing. For example, the user is given 15 additional free minutes if the docking station is full. The user can also call the operation centre who will dispatch a van to pick up or deliver additional bicycles.

In FIG. 4 is an example of a real-time information system. This web application is used in Barcelona (Bicing) informs the users and the employees in real-time about how many bikes each station has and how many are available in that same station. In this case this software also informs if the station is mechanical, electrical or if is currently out of service. The mechanical stations are represented in red, the electric stations are represented in blue and the services that are out of service, empty or

Figure 3: Example of Barcelona (Bicing) Relocation Vans [3]

full are represented in grey. The option of doing a search for one particular station is also available.

Figure 4: Real-time example used in Barcelona (Bicing) [3]

2.1.2 Chicago (Divvy)

This system is currently equipped with 4760 bicycles distributed across the city within 476 Stations. Chicago BSS has a medium of 2 million annual rides and does the distribution of its bikes using vans like the one in the FIG. 6. The rebalancing employees redistribute a medium of 1300 bicycles per day. [4]

Figure 5: Example of Chicago (Divvy) Relocation Vans [4]

Divvy uses a bike known as *Bixi*, used in many others BSSs. In FIG. 6 is an example of the real-time software being used in this system. System stations are solar-powered and are also modular in order to provide an easy installation. The stations also include a touch-screen that allows to obtain all kind of information in real-time about Divvy bikes and stations.

2.1.3 London (Barclays Cycle Hire)

It was launched with 15,000 bikes and 1,000 stations and aims to reduce traffic congestion and to clear out some of the smog in the city. [5]

In FIG. 7 are represented the type of vehicles used for reallocating the bicycles in this BSS. To incentive customers helping in rebalancing, users who find a station full can get an extra 15 minutes of free time to cycle to the next one. London uses a central control room that shows when docks are empty and full and afterwards employees are instructed to deliver and collect bikes around the city and distribute them as needed.

In FIG. 8 is possible to see another real-time software that informs the users and the employees what is the current state of each station. This information is given

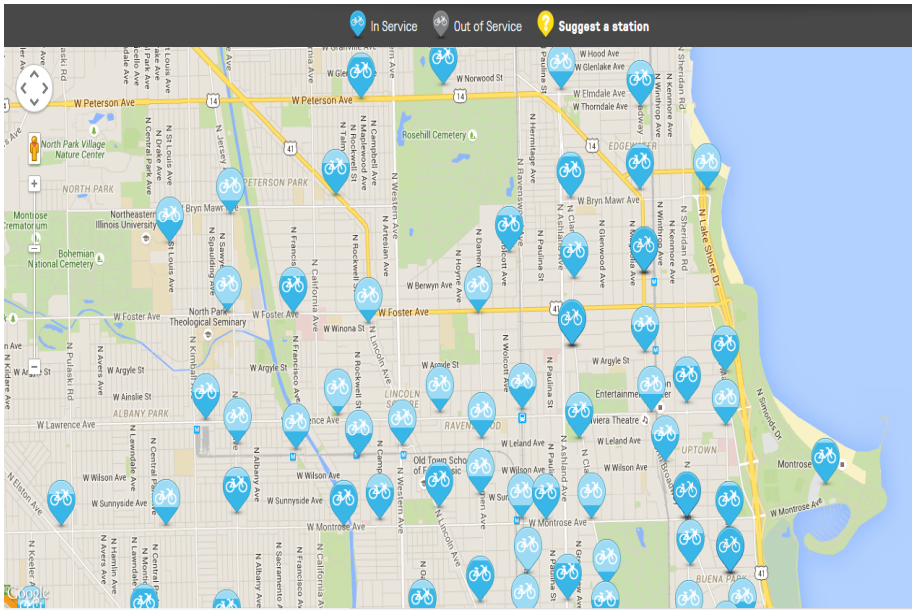

Figure 6: Real-time example used in Chicago (Divvy) [4]

Figure 7: Example of London (Barclays Cycle Hire) Relocation Vans [5]

by informing users and employees about how many bikes are available and how many docks are free. This system also presents a list of temporary suspended dock stations and the option for doing a personalized search for one particular station is also available.

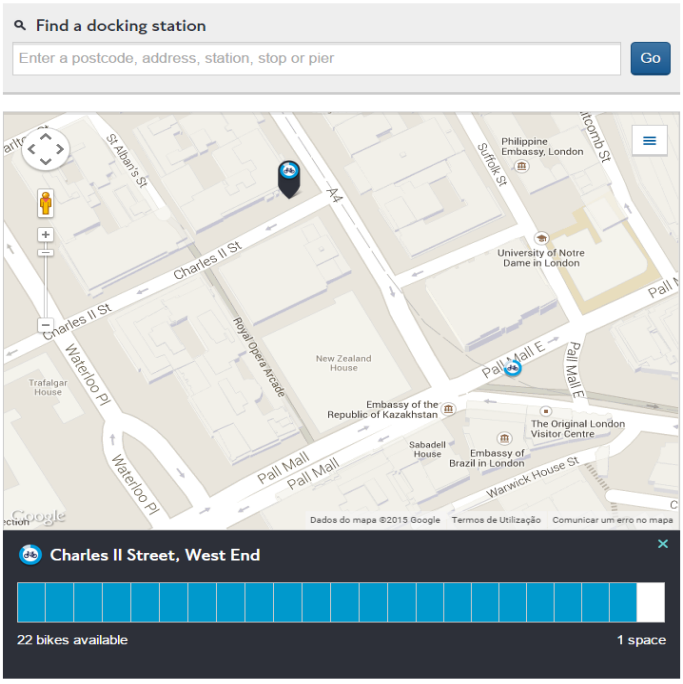

Figure 8: Real-time example used in London (Barclays Cycle Hire) [5]

2.1.4 New York (CitiBike)

New York (CitiBike) is anticipated to bring 10,000 bikes at 600 stations all over the city, from Brooklyn to Manhattan. [6] In Fig. 9 are represented the vehicles used for Rebalancing the BSS.

Figure 9: Example of New York (CitiBike) Relocation Vans [6]

In FIG. 10 is an example of real-time information about the state of the stations indicating the available docks of each station and the available bikes in each station at that particular time. There are different colours for each station in order to inform, in a simple way, their current state to the users. Blue means that the station is in service and with an acceptable number of available bikes, grey means that the station is currently out of service due to a particular reason and yellow means that the station is a part of the planned trip by the user.

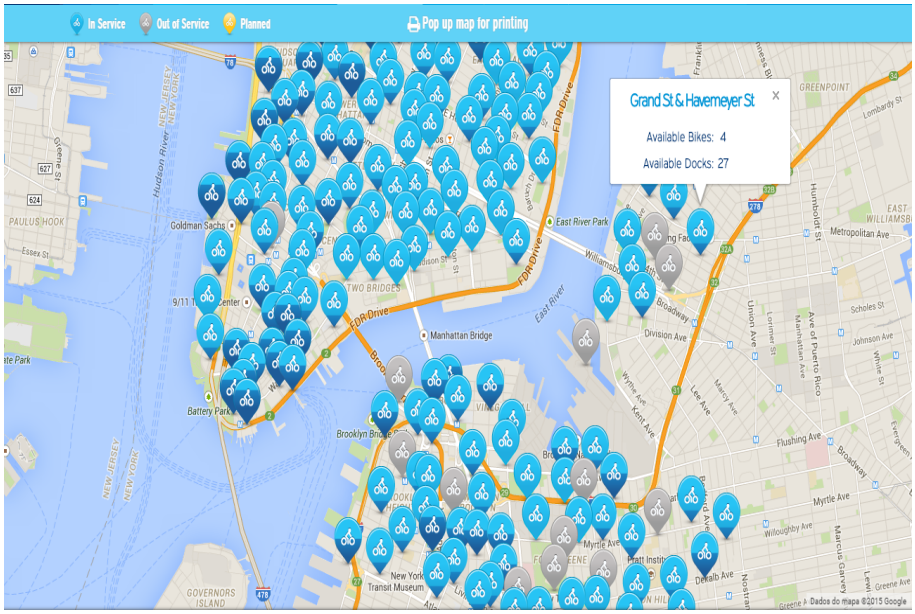

Figure 10: Real-time example used in New York (CitiBike) [6]

2.1.5 Paris (Vélib)

Paris (Vélib) is the biggest BSS in Europe and has over than 20000 bikes available for use and over 1800 dock stations. [7] In FIG. 11 is represented an example of the vehicles used in Paris for redistributing bikes.

This system has 48 redistribution agents working day and night to improve the distribution of bicycles in the stations, 23 trucks that operate with natural gas with a capacity for 20 bikes and two buses with a capacity of 62 bikes. Eighty per cent of maintenance is completed on site as there is an underground storage compartment at each docking station that holds maintenance equipment.

In FIG. 12 a similar method to the ones analysed before is found being used by *Vélib* in Paris that also shows the information in real-time. For each station in real-time the current available bikes and the free docks are displayed. In this example the option to add the station as favourite is also available. The Vélib stations are showed in purple, the searched stations by the users in green, the searched addresses

Figure 11: Example of Paris (Vélib) Relocation Vans [7]

in yellow and in grey are the stations that are close to the limit, either getting empty or full.

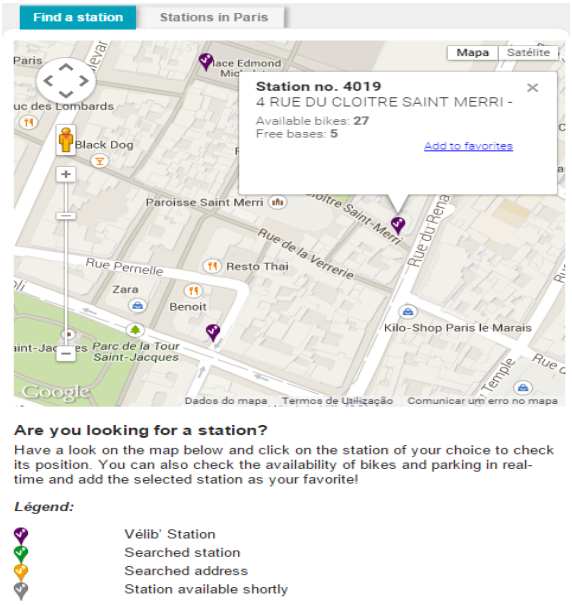

Figure 12: Real-time example used in Paris (Vélib) [7]

Washington, D.C. (Capital Bikeshare)

Capital Bikeshare is a public-private partnership with Alta Bicycle Share that has more than 300 stations and 3000 bikes available for the public. [8]

Figure 13: Example of Washington (Capital Bikeshare) Relocation Vans [8]

Rental stations are automated and powered by solar panels. A wireless data link connects the docks and kiosks to a central bike-tracking and billing database. In FIG. 13 is an example of the vans being used in this BSS to do the redistribution of the bikes. Five *Alta Bike Share* vans redistribute bikes among stations and pick up bikes for maintenance. In order to perform the rebalancing there is a team of 20 van drivers and four dispatches in this system. These van drivers move an average of 2600 bikes every weekday during peak season and it takes them 45 minutes to fill the van, with bikes from several stations, and then unload them where the bicycles are needed. [8]

In FIG. 14 is the software used by *Capital Bikeshare* to inform users and employees in real-time about the state of the stations. Red icons represent bikes that are in service, yellow represent new stations and grey represent stations that are currently out of service. This software also shows the available bikes at that moment in each station and what is the maximum capacity.

This system has an interesting method for helping maintenance services. Each bike has a repair button. Users press this button to report a damaged or malfunctioning bike and take it out of service. Riders are expected to notify *Capital Bikeshare* if a bike is unable to dock at a station or has any other problem.

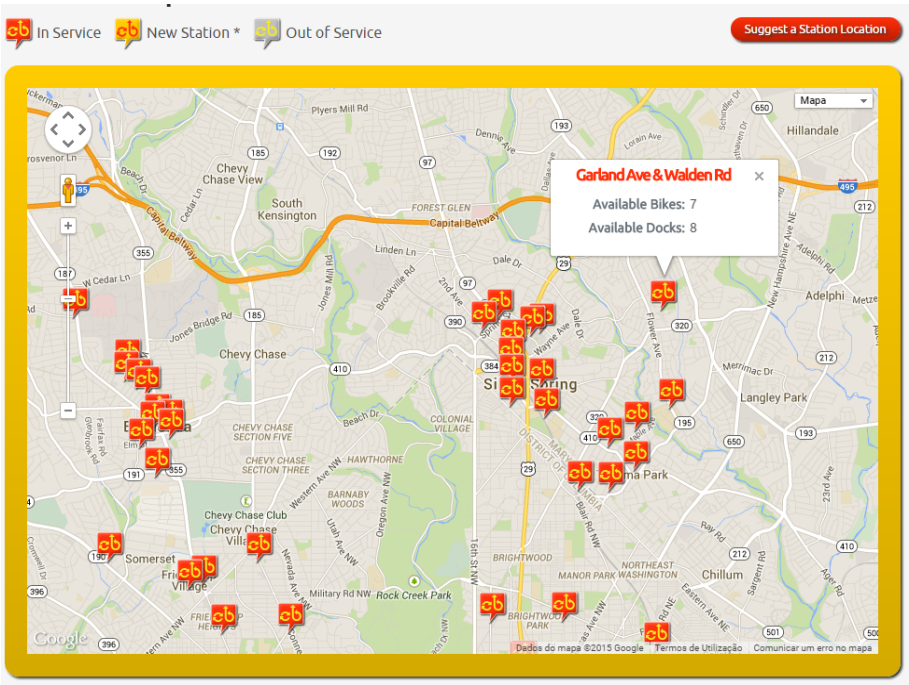

Figure 14: Real-time example used in Washington (Capital Bikeshare) [8]

2.2 Predicting Dock Usage

"The first essential step is to predict the future demand among stations in order to forecast bikes and free docking points requests." [13] When it comes to intelligent algorithms, they are an innovation in this area, so they are not presently being used to do the rebalancing of the bicycles in BSSs just yet.

"Bike stations have specific patterns regarding when they are empty or full. For instance, in cities where most jobs are located near the city centre, the commuters can cause specific peaks in the morning: the central bike stations are filled, while the stations in the outskirts are emptied. Furthermore, stations located on top of a hilly region are more likely to be empty, since users are less keen on cycling up a hill and thus less keen on returning a bike to such a station." [14]

There are several machine learning techniques worth mentioning in this particular area.

2.2.1 Bayesian Networks

These are graphical structures used to represent knowledge about uncertain domains. Each node in the graph represent a random variable while the edges probabilistic dependencies.

It has some advantages: The graphical nature clearly displays the links between different system components, it provides a way of overcoming data limitations by incorporating input data from different sources and Bayesian Networks can also yield good prediction accuracy using learning algorithms, even if the samples are small.

It also has some limitations too, for example Bayesian Networks have limited ability to deal with continuous data and feedback effects cannot be included in the network.

2.2.2 Extra Trees

This type of trees have two main differences in comparison with normal trees: they split at points chosen randomly and they use the whole learning sample to grow the tree and that way identifying the patterns present in that sample.

Extra Trees are also similar to random trees, but there two main differences. The first one is that instead of using different input training sets to different trees, it uses the same input training set for all trees. And the second difference is that Extra Trees pick a node split randomly whereas random trees find the best split among random subsets.

One of the advantages of the Extremely Randomized Trees is that they require less parameter tuning than boosting methods, which may prove has an advantage to predict the BSS docks usage. [15]

2.2.3 Gradient Boosting Machines

Gradient Boosting Machines (GBM) are regression based machine learning method that combines weak learners into a single strong learner. It trains many models repeatedly and each new model progressively minimizes the loss function of the system. Normally this is done by continually fitting a weak classifier, normally a decision tree, and join them to make the final prediction.

The main advantages of this method are robustness, performance, reduced data cleaning and preparation times. Over-fitting is avoided by carefully selecting the regularization parameters and it has great prediction accuracies. [16].

Regue and W. Recker tested three different techniques: Gradient Boosting Machines, Linear Regression and Neural Networks. Results show that GBM outperformed the

other two using Boston as a BSS and different time periods including 30 and 60 minutes. [17] It additionally offers several advantages that make it ideal for online applications because predictions are made as new data are being acquired and models are automatically retrained. With GBM a mean error of 3.6%, 9.5% and 11.2% of the capacity of the station for the 20, 40 and 60 minute windows respectively.

2.2.4 Linear Regression

In this technique data is modelled using predictor functions and unknown model parameters are estimated from the data. Linear Regression is widely used in biological, behavioural and social sciences to describe possible relationships between variables. The advantages of this method are that when relationships between the independent variables and the dependent variables are almost linear it shows optimal results.

Some disadvantages are that this technique is somehow limited to predicting numeric outputs, it is not recommended to model non-linear relationships and it has lack of boundary constraints. This technique also showed to have worse accuracy than Gradient boosting machines. [17].

2.2.5 Neural Networks

Neural Networks are inspired by Biological Neural Networks and are normally used to estimate functions that depend of a large number of unknown inputs. These have also the ability to learn in 3 different ways: Supervised learning, unsupervised learning and reinforcement learning.

This technique has some advantages such as it can be extremely robust and it learns directly from observed data so there is no need to assume explicit functions.

Some of the disadvantages of this method are that to implement large and effective neural networks, large processing and storage resources need to be committed and they require a large diversity of training for real-world operations. This method showed to have a 13.27% worse accuracy in 60 minutes windows than Gradient boosting machines. [17].

Caggiani and M. Ottomanelli used Neural Networks to predict arrivals and departures in their decision support system for the rebalancing of BSS but in this case they did not include external factors as independent variables. [13]

2.2.6 Poisson Regression

Poisson Regression is a type of regression useful for modelling counts of different variables and establish the relation between them. With that relation it tries to predict a certain outcome of another variable, in this case the number of bicycles in each station.

The advantages of this model are that it has discrete distribution and it has a restriction of predicted values to non-negative numbers and allows us to build more complicated models with additional explanatory variables and to model continuous variables using linear trends.

In 2013 a study was carried out by Data Science for Social Good in partnership with Divvy and Chicago Department of Transportation using several different variables to try to predict the outcome of the stations in the Washington DC BSS. The results from this study revealed that this technique may be promising. [18]

2.3 Bikes Redistribution

After predicting, the future state of the stations through an intelligent technique, an algorithm to show to the employees what stations need to be rebalanced and how to rebalance them is needed.

Fricker and Gast concluded that the equilibrium system performance collapses under heterogeneity of user behaviour and that a pressing need for rebalancing operations exists and that equilibrium normally has a specific pattern. [12]

Balancing bikes usually is done by employing a fleet of trucks that move the bicycles between stations in order to avoid empty or full stations. Finding an optimal solution is a challenging task, not only by complexity of the problem itself but also because of the large number o stations that normally BSS have in relation to a small fleet of trucks.

In TAB. 1 are the main differences between user-based relocation strategy and operator-based strategy. Although some user stimulation will be done to reduce the costs of rebalancing the bikes, this project will use an operator-based relocation strategy due to the reliability factor (clear definition of the balancing done in the past and that will be made the in the future) and due to the easy integration of electric vehicles in case of depots.

Relocation strategy	**Advantages**	**Disadvantages**	**Major application area**
User-based	Low costs due to staff savings; Environmentally Sustainable (no additional vehicle movements)	Customer difficult to influence; Users contribution difficult to predict	Long term events with high destination traffic
Operator-based	Reliability due to clear definition; Easy integration of electric vehicles in case of depots	Cost for staff, transporters, and depots; Additional vehicle movements in case of relocation by staff	Short-term events with defined beginning and end and one-time high demand

Table 1: Types of Relocation strategy [1]

The operator-based rebalancing of the bikes of an BSS can be inspired in the Vehicle Routing Problem (VRP) and Traveling Salesman Problem (TSP) because it contains many of same problems. Many authors inspire their theories of bicycles redistribution in vehicles rebalancing. [14] [19]. This section covers some examples of software

inspired in VRP and TSP that can be used to rebalance the bikes.

2.3.1 ArcGIS for Transportation Analytics

ArcGIS for Transportation Analytics [9] is a paid software produced by Esri used for solve complex routing, scheduling, and mobile asset management problems. It contains a route planner, that plans and optimizes work for your fleet and a Status Dashboard, that is a web application built on Javascript that displays location data of the vehicles . This software is compatible with IOS and Android devices offering turn-by-turn driving directions and voice guidance for drivers via on-board.

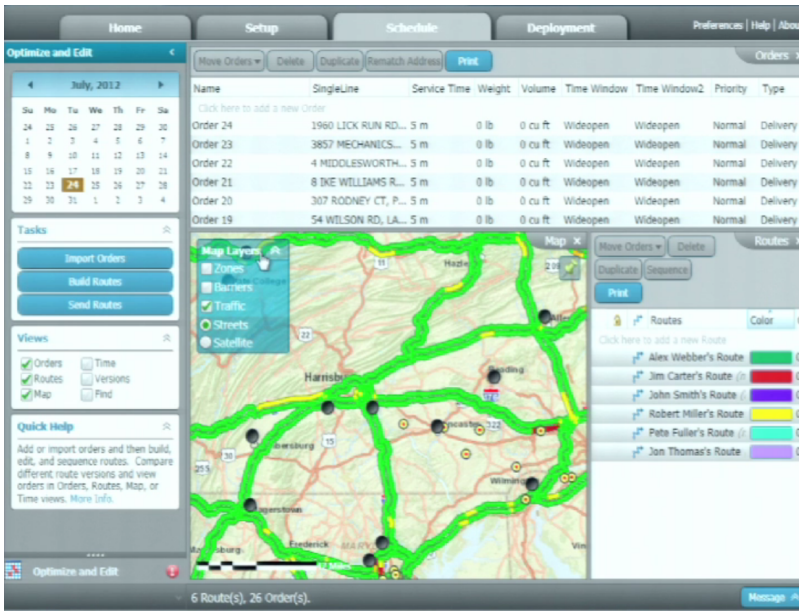

Figure 15: ArcGIS for Transportation Analytics Software [9]

FiG 15 presents an example of the route planner in the software ArcGIS for Transportation Analytics.

2.3.2 Disc

Misc [20] is a paid software developed by *Mjc2* designed for scheduling and optimizing large, complex logistics operations including: multi-depot logistics, delivery scheduling, driver/vehicle scheduling, integrated logistics, load planning and strategic planning.

Mjc2 delivery software tools optimize allocation of work to drivers and vehicles to maximise productivity. The delivery route planning system takes account of

constraints such as time windows, lorry type and capacity, driver hours, access restrictions, load compatibility and temperature control. The routing and scheduling algorithm is designed to cope with very large networks and can model integrated multi-depot operations to optimize resource allocation. This software is compatible with Android and IOS.

2.3.3 Jsprit

Jsprit [21] is Java based, open source toolkit for solving both TSP and VRP. It can solve problems with pickups and deliveries, heterogeneous fleets, multiple depots and different start and end locations so it is compatible with bike-sharing rebalancing problem. Its main advantages are the flexibility, the fact of being lightweight and being well documented.

The meta-heuristic algorithm that is applied to solve the various vehicle routing problems with Jsprit was developed by Gerhard Schrimpf. This algorithm is best suited for complex problems that have many constraints and a discontinue solution space. [22] There are only very few open source implementations and even fewer projects that can deal with real world problems that usually have many side-constraints, so this software is a very good choice when it comes to using an open-source software.

2.3.4 Open-VRP

Open-VRP [23] is an open-source framework to model and solve VRP-like problems for students, academics, businesses and hobbyist alike. The library is written in Common Lisp's CLOS. In this case Problem object (e.g. VRP) and the Algorithm object (e.g. Genetic Algorithm) are modelled separately and combined with the generic method. Different solution algorithms can be tested and compared against each other on the same problem. The documentation in this option is not reasonable, so it will be excluded from the final options.

2.3.5 OptaPlanner

OptaPlanner [24] is an lightweight, embeddable planning engine open-source software written in Java that solve cases including vehicle routing. OptaPlanner does not work out-of-the-box on Android yet.

This software is scalable (handles 50 000 variables with 5 000 variables each, multiple constraint types and billions of possible constraint matches), well documented and has a public API. Using a fleet of vehicles, each vehicle can service multiple customers, but it has a limited capacity. Although this is an open-source software, it

has an integrated graphical platform that limits the usage on this software in APIs or in micro-services.

2.3.6 Logvrp

Logvrp [10] is a fleet route optimization and scheduling software. This software is paid and is used for route optimization, transportation planning and fleet management. Logvrp is a Web-based software, scalable and uses cloud services to help do the calculations faster. Android and IOS devices are compatible with this option.

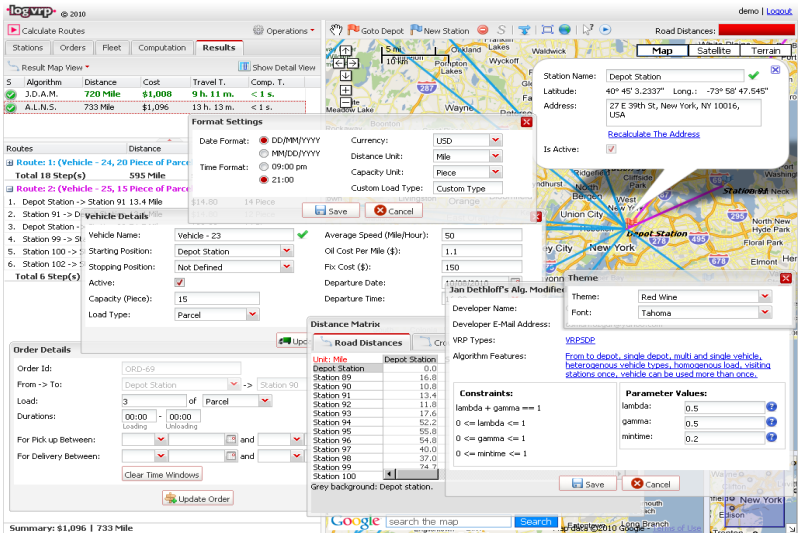

Figure 16: Logvrp Software [10]

In FIG. 16 is the route planner of Logvrp Software.

2.4 Summary

After analysing all the solutions above, there are several possibilites that need to be testes and explores. GBM shows to have an advantage in several phases of the learning process and better prediction accuracies in this matter. [17] This solution showed superiority against several other techniques including Neural Networks and Linear Regression. K-nearest Neighbours Regression was excluded due to the fact that it cannot handle more than 30 features. Bayesian Networks were also excluded because they have limited ability to deal with continuous data. From all the others techniques studied, Extra Trees, Poisson Regression and Random Forests also presented strong technical features to be applied to docks usage predictions. These four predictions techniques will be implemented, trained and tested in order to establish what is the best technique for this situation.

Regarding the dynamic rebalancing algorithms, the algorithm will be modified in order to fit in the bike-sharing scenario, so an open-source software is preferred and paid options will be excluded. From the open-source alternatives *Open-VRP* was excluded due the bad documentation provided and *OptaPlanner* was also excluded due to the fact that it brings an integrated graphical platform that limits the usage on this software in APIs or in micro-services. From the open-source options the one who has better documentation, can solve several types of VRPs and is lightweight is *Jsprit*. The meta-heuristic used in this software, developed by Gerhard Schrimpf is the best suited for complex problems that have many constraints and discontinue solution space. [22].

3 Proposed Approach

This chapter is divided in three parts and addresses methodology, planning and risk analysis. Section 3.1 explains the methodology used in this project. Section 3.2 shows how was the project divided and planned and section 3.3 presents the problems that can be encountered during this project and possible solutions attached.

3.1 Methodology

Although this project has a wide spectrum of possible requirements and the technologies that are used are complex, it is a independent module that requires good organization and planning of the several components being developed, hence the methodology that was adopted in this project was based on the Royce's Final Waterfall model. In FIG. 17 is a representation of this model.

Royce's Final Waterfall model is considered an improvement in relation to the original Royce Waterfall Model. The original Royce's Model didn't allow to go from testing to code again, a feature that is absolutely essential in this project because the dynamic redistribution module will depend on the testing results of the prediction module, so transitions from testing to code and from code to testing will occur frequently in this project.

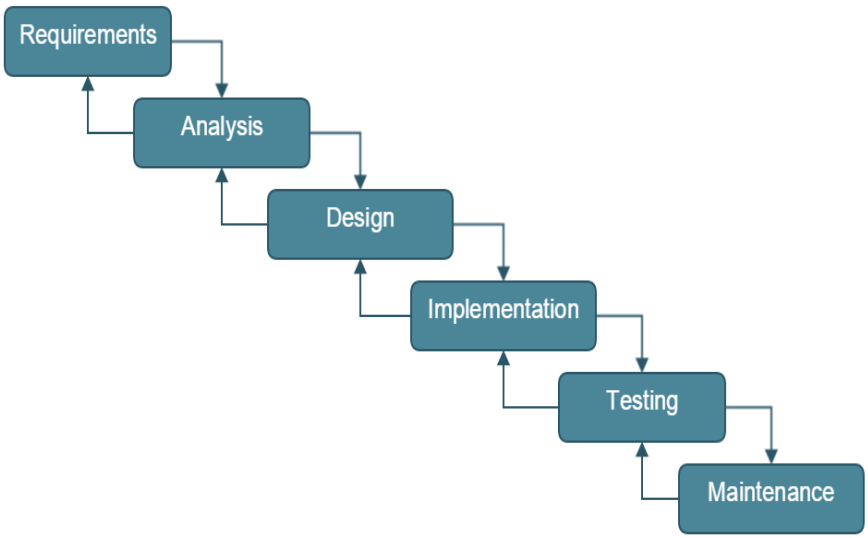

Figure 17: Royce's Final Waterfall Model

The module being addressed in this internship is connected to other modules that are currently being developed by *Ubiwhere* collaborators.

Redmine was used to consult the user stories and new tasks assigned. This tool registers the tasks that each user is going and what is the start date and the end date of that specific task. This tool is also being used to register the time that was dedicated to each task and what is the current percentage of the task already made.

For communication purposes the company uses *Slack*, which is a team communication tool that provides Mac, IOS and Android Apps, as well as web versions.

Fortnightly meetings took place with the advisor Carlos Bento, PhD professor at the Department of Informatics Engineering of the University of Coimbra. All these meeting reports can be found in Appendix A. Several other meeting took place with *Ubiwhere* Project Managers Ricardo Machado and Carlos Oliveira.

3.2 Planning

In order to have a better perspective of the work done in both semesters, two Gantt diagrams were made: a Gantt diagram for the first semester and one for the second semester. Both diagrams include the initial expectation for the time needed for each task and the real time used to accomplish that same task.

3.2.1 First Semester

ID	Task Name	Duration	Start	Finish
1	**1º Semestre**			
2	State of the Art	23.13 days	Mon 15/09/14	Wed 15/10/14
3	State of the Art	23.13 days	Mon 15/09/14	Wed 15/10/14
4	Requirements Analysis	22.13 days	Wed 15/10/14	Fri 14/11/14
5	Requirements Analysis	34 days	Wed 15/10/14	Sun 30/11/14
6	Arquitecture Definition	22 days	Sat 15/11/14	Mon 15/12/14
7	Arquitecture Definition	22 days	Sat 15/11/14	Mon 15/12/14
8	Test different technologies	14 days	Mon 15/12/14	Thu 01/01/15
9	Test different technologies	14 days	Mon 15/12/14	Thu 01/01/15
10	Definition of the risk plan	14 days	Mon 15/12/14	Thu 01/01/15
11	Definition of the risk plan	14 days	Mon 15/12/14	Thu 01/01/15
12	Definition of the test plan	11 days	Thu 01/01/15	Thu 15/01/15
13	Definition of the test plan	11 days	Thu 01/01/15	Thu 15/01/15
14	Write intermediate thesis	63 days	Sat 01/11/14	Tue 27/01/15
15	Write intermediate thesis	53 days	Sat 15/11/14	Tue 27/01/15

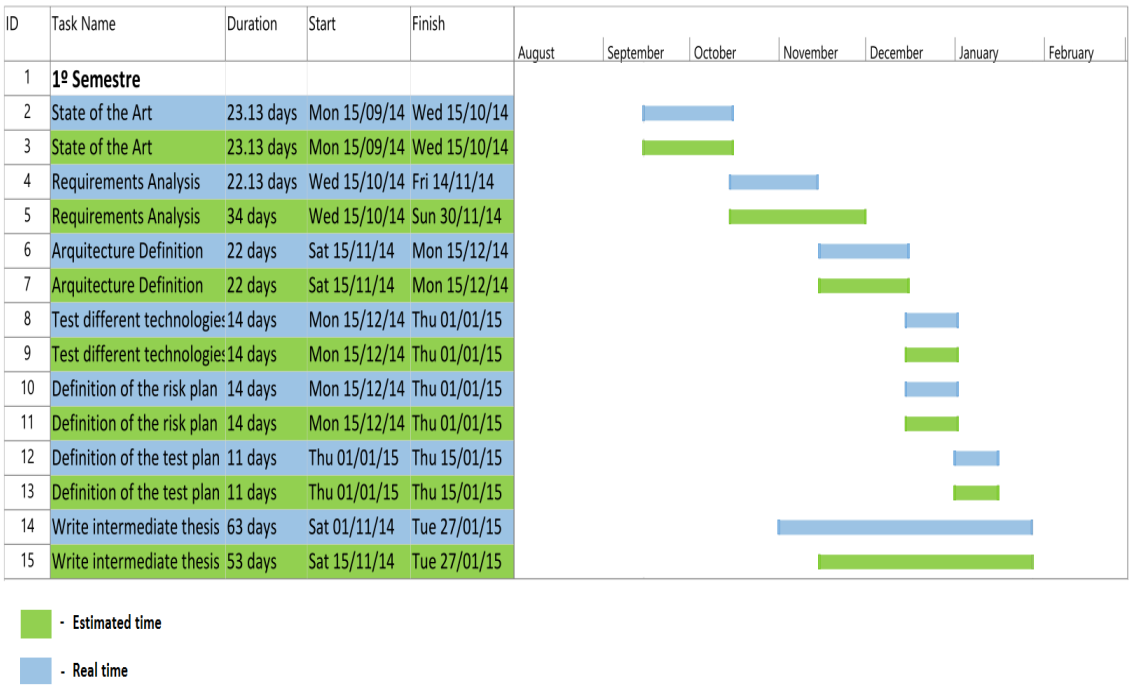

Figure 18: Final Gantt diagram for the first semester

In the first semester 16 hours were dedicated per week to the internship. This semester focused on the project planning. In FIG. 18 highlighted with green is a perspective of the initial plan for this semester and in blue is the real time spent in each task. The first step was an initial study of the state of art and the technologies being used nowadays in BSSs. Afterwards the Requirements Specification for this system were presented, specified in form of user stories. Then the architecture followed, where an overall view of the *BikeEmotion* developed by *Ubiwhere* was provided and its components and also a deeper look at the architecture that is going to be used for this specific project. The next phase was to test different technologies that will be used in this project, including GBM. Afterwards a risk mitigation plan and a test plan were defined in order to void some future problems in the development of this project. The final phase was dedicated to write the intermediate thesis. In the same diagram, highlighted with blue it is possible to see that there were not many changes in the initial plan. The only change worth mentioning was the requirement analysis that took more time than expected due to the complexity of the requirements and the undefined final client. The rest of the tasks were scheduled on time.

3.2.2 Second Semester

	Task Name	Duration	Start	Finish
1	**2º Semester**			
2	Correct Report Errors	9 days	Tue 03/02/15	Fri 13/02/15
3	Correct Report Errors	7 days	Tue 03/02/15	Wed 11/02/15
4	Data Preparation	8 days	Mon 16/02/15	Wed 25/02/15
5	Data Preparation	7 days	Thu 12/02/15	Fri 20/02/15
6	Data Processing	10 days	Thu 26/02/15	Wed 11/03/15
7	Data Processing	10 days	Sat 21/02/15	Thu 05/03/15
8	Write EWGT 2015 intermediate paper	15 days	Thu 12/03/15	Wed 01/04/15
9	Write EWGT 2015 intermediate paper	15 days	Fri 06/03/15	Thu 26/03/15
10	Write Prediction algorithm	16 days	Thu 02/04/15	Thu 23/04/15
11	Write Prediction algorithm	32 days	Fri 27/03/15	Mon 11/05/15
12	Dynamic Bike Rebalancing algorithm development	18 days	Fri 24/04/15	Tue 19/05/15
13	Dynamic Bike Rebalancing algorithm development	25 days	Tue 12/05/15	Mon 15/06/15
14	Test and validate algorithms	28 days	Fri 24/04/15	Tue 02/06/15
15	Test and validate algorithms	45 days	Mon 11/05/15	Fri 10/07/15
16	Develop API	10 days	Wed 03/06/15	Tue 16/06/15
17	Develop API	16 days	Sat 11/07/15	Fri 31/07/15
18	Write final thesis	19 days	Wed 17/06/15	Mon 13/07/15
19	Write final thesis	24 days	Sat 01/08/15	Wed 02/09/15

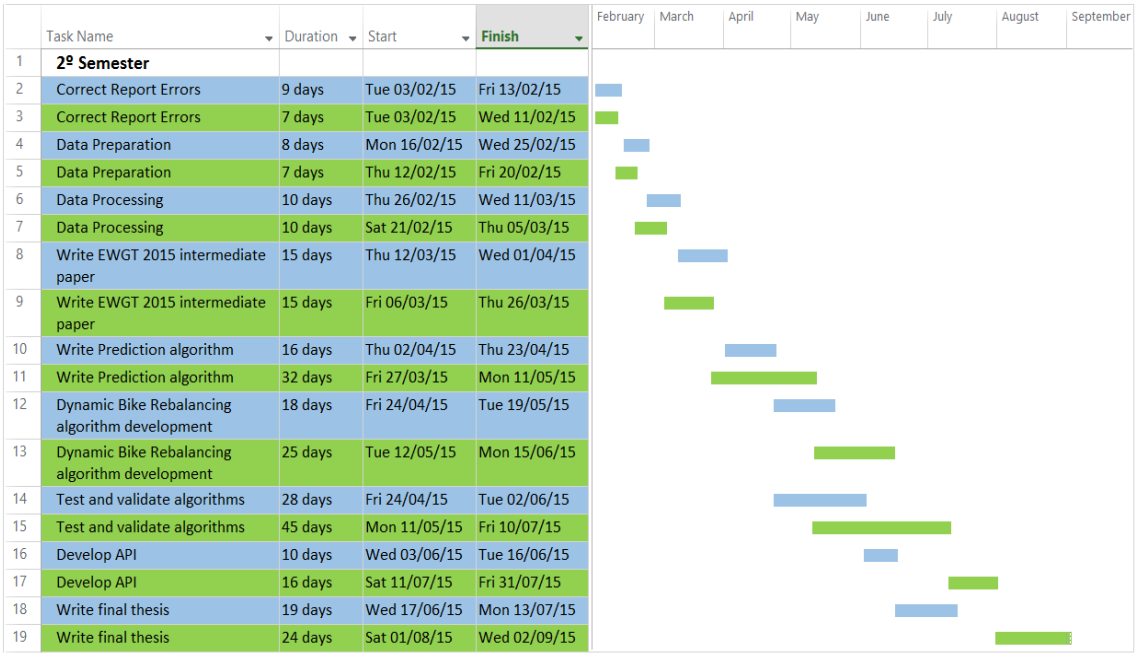

Figure 19: Final Gantt diagram for the second semester

In the second semester 40 hours were dedicated per week to the internship. This

semester focused on the implementation of the intelligent rebalancing system, in the test and validation phase and in the writing of the final thesis.

The second semester started with the correction of some errors in the report pointed out by the Juries and the Advisors in the intermediate defence. Afterwards the Data preparation and Data Processing followed. This part consisted on obtaining all the data regarding all trips made in each BSS, access to live feeds from the respective websites, use of several APIs to obtain data about the altitude, windspeed, type of weather, humidity and other variables. After obtaining the data it had to be processed so data of the *traces* of the bicycles had to be transformed and adapted so that they could be interpreted by the machine learning techniques. Next the *EWGT 2015* intermediate paper was written and the development of the prediction algorithms also started. In this phase four machine learning techniques were developed: Extra Trees, Gradient Boosting Machines, Poisson Regression and Random Forests.

Afterwards Dynamic Bike Rebalancing development begun by using *Jsprit Toolkit* and adapting it to this particular problem, this way it was possible to develop the best routes to do the distribution of the bicycles having in account real distances and times between stations. At the same time tests were executed to all the code including several types of functional and non-functional tests. This phase served to validate all work done and evaluate the quality of the algorithms developed. The preparation of data and tests for Chicago BSS feature was added to the original plan due to the advisement of both Juries and Advisors, having has final objective to prove that the predictions and distribution systems can adapt to several different types of cities and BSSs. The development of the API followed. The API comes has a way for the company use this software in a easy and secure way and also to make the integration with other software easier since *Ubiwhere* deals with a lot of micro-services. Finally the final thesis was written in order to document all the work done.

In FIG. 19 the colour blue represents estimated time and green represents the real time for each activity in the second semester. Initially all tasks were ahead of schedule but the development of intelligent prediction algorithms was delayed. This delay was due to the fact that having more predictions techniques to develop brought more complications, especially Poisson Regression, because R language had to be learned in order to be developed. The preparation of data for Chicago and applying both the prediction and distribution for this system was also added to the original plan. The test and validation also took more time than expected due the complexity and variety of tests performed. Since the features that were added took more time to develop and analyse, but also added value to the final project, the decision to delay the final delivery to September was made.

3.3 Risks Analysis

It is important to try to anticipate some uncertain events that may affect this project in order to be able to overcome them or even prevent them. This can be achieved by doing a risk mitigation plan.

TAB. 2 contains the risks identified during the internship as well as some strategies to overcome that possible problems.

Risk	Likehood	Impact	Mitigation Strategy
The proposed features are innovative and are not defined in any known BSS at the time so unknown problems can arise with the main intelligent algorithm.	Medium	High	Have a second option intelligent algorithm ready to take place
This project module also depends on others APIs so it is not possible to guarantee services will be available all the time.	Low	High	The application will monitor the state of the APIs and in case one of them is offline, it will be disabled so that no crash can occur.
Prediction of BSS docks with very high percentage of error.	Low	Medium	Explore other variables being used to calibrate the algorithm
Not knowing the communication technologies such as SOAP, REST, XML, JSON, etc.	Medium	Medium	Study and explore used technologies and tools.
Data acquired from an BSS not enough to calibrate the algorithm	Medium	Low	Have data and *traces* from different BSSs available to use

Table 2: Risk Mitigation Plan

4 Requirements

In this chapter the Requirements Specification for this system are presented, specified in form of user stories.

User Stories

This phase is essential so the main requirements of the project may be defined and planned and only after implemented and tested. These were obtained in conjunction with project managers Ricardo Machado and Carlos Oliveira.

The requirements are specified in the form of user stories - A simple, effective and non-formal way to express the requirements projects. Basically user stories are a quick way of handling the product requirements without creating formal requirement documents or performing administrative maintenance tasks. The user stories are define as:

"As a (role), I want (goal/desire)"

In each user story the requirement can be considered Primary or Secondary. Primary requirements are the main focus of the project and where highest effort will be spent and also possess a higher degree of interest from a scientific point of view. Secondary requirements, on the other hand, are meant to support the Primary objectives or improve the current state of the prototype.

4.1 User Story 1 - Prediction of Bike Stations Future States

Role	Employee
Priority	Primary
Description	As an Employee I want to know what stations will be full or empty in the future in order to focus them.
Acceptance Criteria	1. Employee must receive a prediction for 1 hour ahead in order to have time to the redistribution of the bikes. 2. Predictions must be based on the past trips and real-time information. 3. The trips with less than 60 seconds and trips made by employees must be filtered from the data. 4. Only include information of the direct routes of the BSS Users. 5. The information must be stored in Cassandra Database.

Table 3: User Story 1 - Prediction of Bike Stations Future States

4.2 User Story 2 - Dynamic Bike Redistribution

Role	Employee
Priority	Primary
Description	As an Employee I want to know how to execute the redistribution and how many bikes should I carry.
Acceptance Criteria	1. Predictions must be loaded from Cassandra Database. 2. Employee must get the number of bikes to reallocate between stations. 3. Employee must get the sequence of stations during the redistribution of the bicycles. 4. Available vehicles for the redistribution must be taken in account.

Table 4: User Story 2 - Dynamic Bike Redistribution

4.3 User Story 3 - Include Time and Location

Role	Employee
Priority	Primary
Description	As an Employee I want to include the time and the location as an factor that influences the user in, order to get more efficient predictions of the docks future state.
Acceptance Criteria	1. Time and Location must be included in the variables used to predict docks future state. 2. The impact that this variable has on the prediction must be tested.

Table 5: User Story 3 - Include Time and Location

4.4 User Story 4 - Make an API

Role	Developer
Priority	Primary
Description	As an Developer I want to implement a mechanism to connect with other modules from *BikeEmotion* with security and flexibility.
Acceptance Criteria	1. API must be able to connect with the other modules of BikeEmotion through Representational State Transfer (REST). 2. Application must verify connection before connection with external live feeds.

Table 6: User Story 4 - Make an API

4.5 User Story 5 - Diagnose the most problematic stations

Role	Employee
Priority	Secondary
Description	As an Employee I want to know what are the most problematic stations in order to focus them.
Acceptance Criteria	1. Employee must be warned of the most problematic stations. (Stations that get empty or full in that moment) 2. This analysis must be taken in account for doing the redistributions. 3. Past stations where bikes were redistributed must be saved in database.

Table 7: User Story 5 - Diagnose the most problematic stations

4.6 User Story 6 - Include Weather

Role	Employee
Priority	Secondary
Description	As an Employee I want to include the weather as an factor that influences the user, in order to get more efficient predictions of the docks future state.
Acceptance Criteria	1. Weather must be included in the variables used to predict docks future state. 2. The impact that this variable has on the prediction must be evaluated. 3. Weather information must be acquired through an API.

Table 8: User Story 6 - Include Weather

4.7 User Story 7 - Include City Topology

Role	Employee
Priority	Secondary
Description	As an Employee I want to include the city topology as an factor that influences the user in order to get more efficient predictions of the docks future state.
Acceptance Criteria	1. City Topology must be included in the variables used to predict docks future state. 2. The impact that this variable has on the prediction must be evaluated. 3. Weather information must be acquired through an API.

Table 9: User Story 7 - Include City Topology

5 Architecture

This chapter delineates the overall architecture of the Smart BikeEmotion and the specific module that is going to be developed. On section 5.1 is provided an overall view of the *BikeEmotion* developed by *Ubiwhere* and its components. Section 5.2 provides a more deep look at the architecture that is going to be used for predict the docks future state. In Section 5.3 is the architecture of the algorithm that is going to be used for redistributing the bicycles.

5.1 Smart BikeEmotion Overview

In FIG 20 are represented all the components of the project *Smart BikeEmotion*, a project being developed by three different companies that are *Ubiwhere*, *Micro I/O e* Ponto.c.

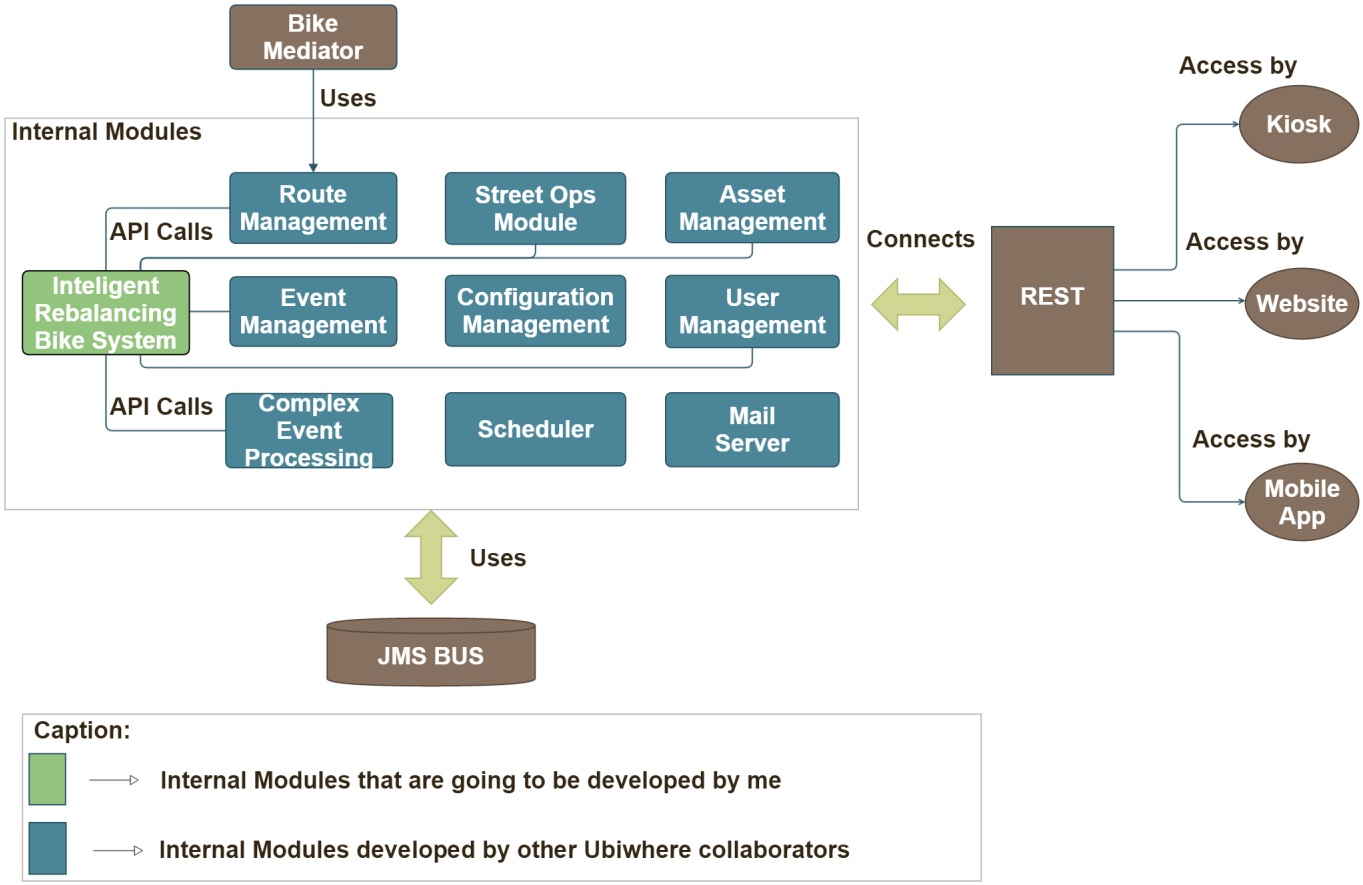

Figure 20: Smart BikeEmotion Architecture Overview (Level 0)

In FIG. 20 is represented the Intelligent Rebalancing Bike System that is directly connected to several other modules. The module represented in green is the one that is going to be developed during this internship and the modules in blue are the modules that are already developed or are currently being developed by other staff.

Route Management is a module that plans the routes being taken by distributors, Event Management module manages every event that might occur (e.g. departure of bikes, arrival of bikes) and Complex Event Processing processes that same events and informs other modules about that situations. Asset Management module manages the physical assets of the BSS like bikes and stations and User Management manages the users and employees information and related data. Finally the Street Ops Module manage the operation of the employees and the data related to the redistribution of the bikes. All the information is accessed through REST by Kiosk, Website or Mobile Application.

5.2 Intelligent Dock State Prediction

In FIG 21 is represented the architecture of the intelligent prediction of docks future state algorithm.

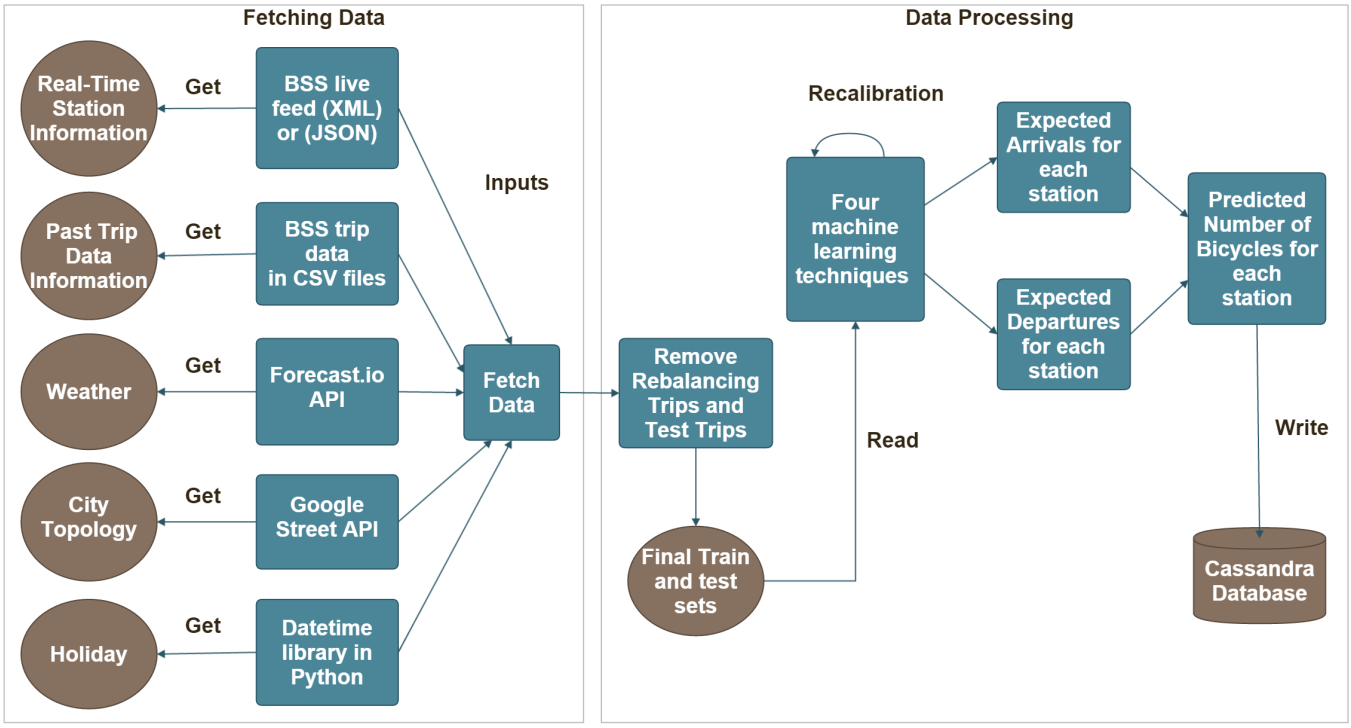

Figure 21: Intelligent Dock State Prediction Architecture (Level 1)

This Module represents the investigation and research part of the internship. Python and R will the languages used in this part because they both have good libraries to work with machine learning techniques and they are also very well documented. A test case was also defined because a final Client is yet to be defined by *Ubiwhere*. The test case that was chosen was *Capital Bikeshare* (Washington D.C.) because it has a reliable live feed information, it has a big and dynamic population and it has a big number of information of the routes taken by the users specified in form of Comma Separated Values (CSV) files. *Divvy* (Chicago) was also used as secondary test case, in order to test the prediction system in time but also in space. Chicago has completely different user patterns from Washington so it will be interesting see the final results.

In order to predict the number of bikes in each station one must get several information first, that will be used as variables in the algorithm to calculate the pattern in the movement of the bikes in the BSS. Among these variables are the weather in that day, the current time and location, the topology of each station and data information about the several trips that were made in the past by the users. Real-time Station Information and Past Trip Data Information will be acquired through a Extensible Markup Language (XML) or JavaScript Object Notation (JSON) live feed and CSV files respectively. The weather will be collected by the *Forecast.io API*, which provides accurate short-term and long-term predictions to current and past weather. City topology will be gathered with *Google Elevation API* and holiday information will be gathered through the Datetime library of Python that indicates if the day in question is a day of the week or not.

After obtaining this information, this data has to be filtered by removing the trips with less than 120 seconds and the trips made by staff to rebalance the system. Afterwards this filtered information will be stored in a Apache Cassandra Database, the database chosen by *Ubiwhere* due to its high performance, elastic scalability and fault tolerant platform. After this step four machine learning techniques including Extra Trees, Gradient Boosting Machines, Poisson Regression and Random Forests will try to learn the pattern of the BSS. This prediction will tell us the expected arrivals and departures for each station, that consequently will be added or subtracted to the real-time number of bikes in that current moment, which will give us the predicted number of bicycles for each station. This prediction will hopefully help the employee to know the stations where rebalancing will be needed.

5.3 Bike Redistribution

In FIG. 22 is the architecture of the redistribution algorithm of the bikes, which will help the employee to know what are the suggested corrections to be made and what is the best sequence to do them.

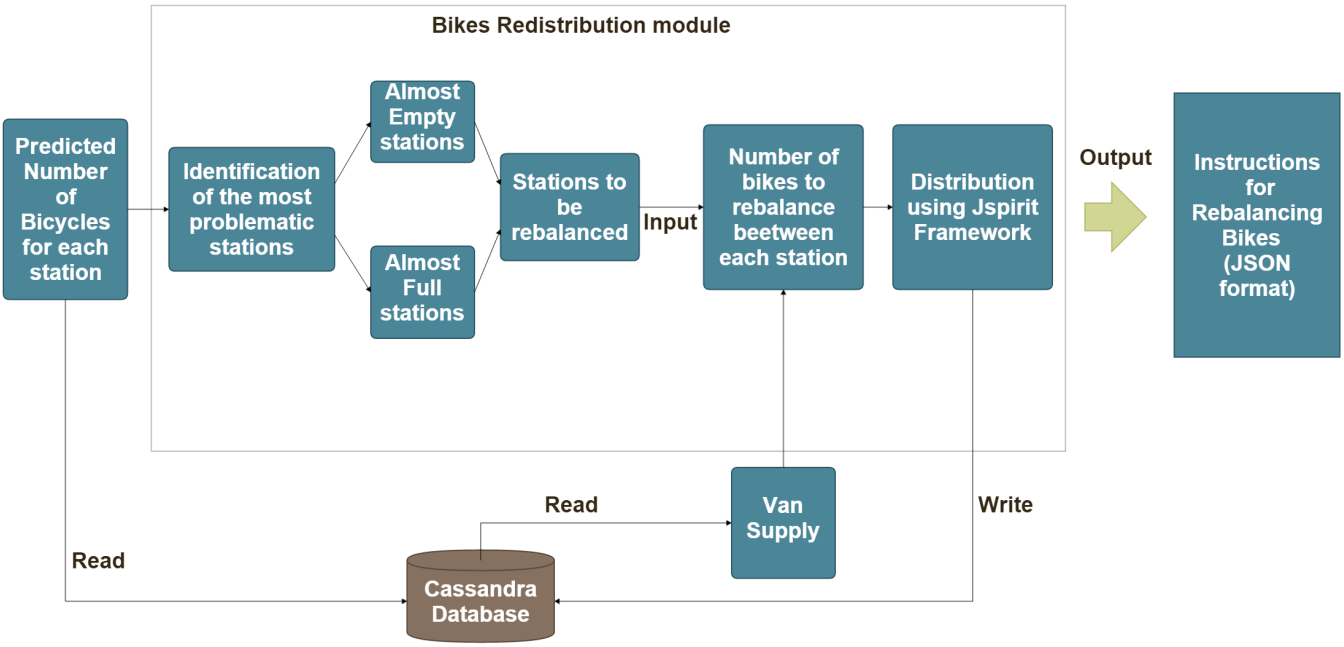

Figure 22: Redistribution Bike Algorithm Architecture (Level 1)

This Module represents the engineering part of the internship. The first step is to get the data calculated of the algorithm in the sector 5.2 from Apache Cassandra Database. Afterwards an identification of the most problematic stations is made, selecting the stations that will be empty or full in the future. Then the selected problematic stations and the suggested number of bikes being relocated in each station are calculated and sent to *Jspirit* Framework, referenced in section 2.3, by using the current available distribution vehicles. This framework is inspired in the VRP, which uses the best algorithms to compute the best sequence to do the redistribution of the bikes. This instructions are sent to other modules of the *BikeEmotion* system through a REST API in JSON format. All distribution and predictions data is stored in Cassandra Database, including past requests. Each BSS own van supply is also stored in the database.

6 Implementation

This chapter addresses some implementation details about each feature and problems that were faced during the implementation.

6.1 Intelligent Prediction Module

This Module represents the investigation and research part of the internship due to fact that according to our research intelligent prediction systems are not being used in the market in any real BSS yet. It emphases on developing software that makes hourly predictions of the number of departures and arrivals for each station and then uses the information from the live feed to present a real number.

6.1.1 Data Acquisition

The first step to develop the Intelligent Prediction Module, was to obtain information about all trips performed by the customers in both the Washington BSS and in the Chicago BSS. This information was obtained in form of CSV files in the respective websites of the BSSs.

Besides this information, it was also acquired information about the weather in each location in intervals of 1 hour, the altitude of each station, and if each day in particular was a workday or a holiday. All this information was gathered and joined together with the information of all trips in order to create train files and test files to make the final prediction of how many bicycles arrive and leave each station.

In order to get the information about the weather in intervals of 1 hour the *Forecast.io API* was used. Several other APIs were analysed but many did not allow access to weather from past dates and others had a low daily requests limit. This API in particular allows us to gather information up to 60 years in the past, such as the temperature, humidity, wind speed and the actual weather conditions (if it was a sunny day or cloudy day). This API has a limit of 1000 requests a day and it can gather information about all locations in the world, so it would fulfil all our project demands.

The altitude of each station was collected using the *Google Elevation* API. This API was chosen due to the hight daily limit requests permissions and to the very well organized documentation made available by *Google*. With this API is possible to make 2500 request per day for all locations on the surface of the earth, including oceans. It was already proven that the elevation of the stations influences the behaviour of the BSSs customers, tending to ride the bicycles from the high altitude stations to low elevation stations but not the other way around. This leads to

high amount of bicycles in low altitude stations and low number of bicycles in high elevation stations. [25] Elevation will only affect predictions if the stations are mobile, otherwise it will be a constant number.

The real number of bicycles is accessed through a live feed. The feed is presented in XML by Washington and in JSON by Chicago. This real live feeds inform us about the status of each stations and the number of bikes in each particular stations. All this information is updated every minute.

6.1.2 Data Processing

All the raw data obtained has no meaning if is not manipulated and well organized. The data regarding the traces of all the trips made by the costumers was filtered, by removing all trips that have less than 2 minutes of riding time, which represent mostly test trips or malfunctioning bikes and by removing all trips where the station of origin and destiny is the same and the riding time is less than 1 hour simultaneously, which do not have impact in the prediction since they compensate themselves. All this data is also ordered, firstly by station and secondly by date, in order to have a faster analysis of all the traces and transform that traces into organized data.

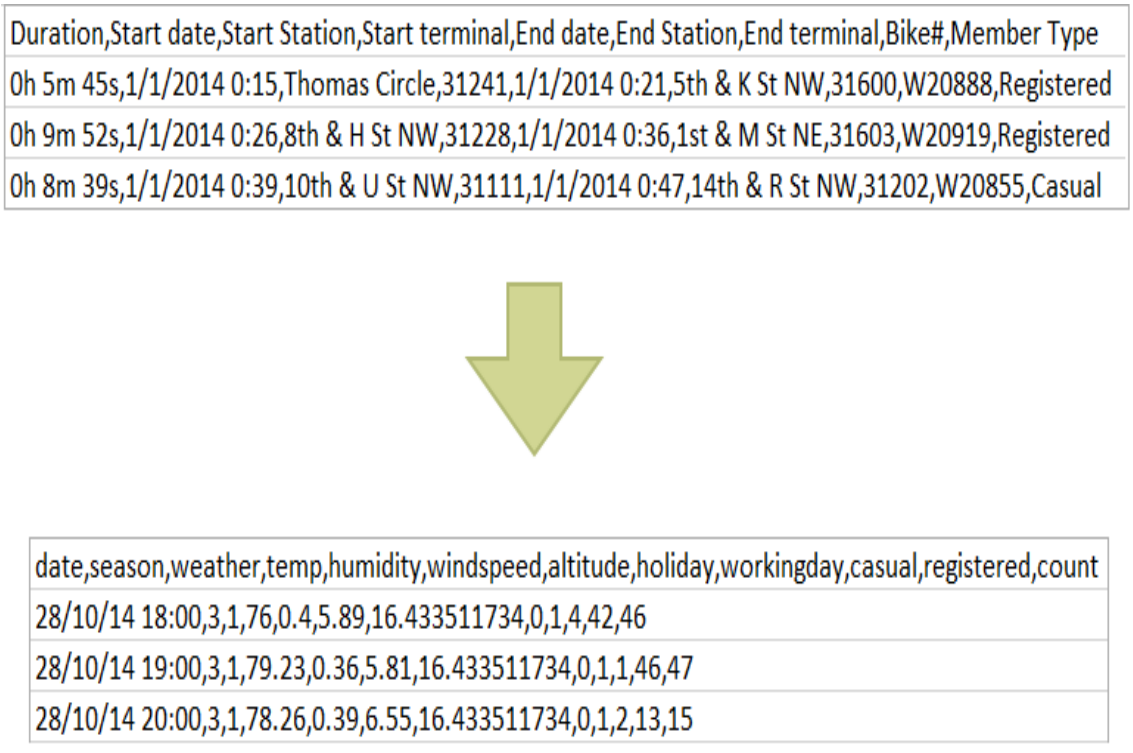

Figure 23: Data Processing

In FIG. 23 is possible to detect that all data mentioned before is then divided by hourly time periods displaying the season, type of weather, temperature, humidity,

altitude, type of day, type of user and the total count of arrivals or departures depending of the situation.

Casual users and customers are trained separately in every machine learning technique due to the fact that these two type of users have very different routines and patterns. Arrivals and departures are also trained separately and then summed at the end for the same reason. These two training methods offer more precise predictions in general.

6.1.3 Prediction

In order to test and evaluate the best prediction technique for Chicago and Washington BSSs, four techniques were developed and tested. *Weka*, a collection of data mining of machine learning algorithms, was not used in this case because it lacked several of the algorithms that needed to be tested and it would not be possible to apply some personal enhancements to the techniques. Consequently Extra Trees, Gradient Boosting Machines, Poisson Regression and Random Forests were fully developed.

Extra Trees, Gradient Boosting Machines and Random Forests were all developed in python. Some Python resources were used to operate with machine learning techniques such as the main libraries *scikit-learn* and *numpy*. *Scikit-learn* is a library for data mining and data analysis. This tool has resources to several machine learning regression techniques that help making machine learning predictions. *Numpy* is used by *scikit-learn* to store the datasets into N-dimensional array objects and to calculate exponentials inside the arrays. Logarithmic function was applied to the several features in order to be more symmetric and unimodal distributions. Transforming the features in this way in a very important step in order to make the learning variables meaningful. [26].

Poisson Regression was developed in R language due to many problems encountered when using python to apply this technique. The basics of this language had to be learned although this step was not very difficult because this language is intuitive and simple. R language, along side with python, is one of the most popular programming languages to develop machine learning algorithms. In this case the *CARET* library was used for data splitting, feature selection, training and predicting. The package *Metrics* was used in order to help in the evaluation of this machine learning technique. *Grid Search* function was used in the project to calibrate the parameters of each used learning technique, this way it was possible to get the best results for each technique used and then compared them.

Several feature engineering were tested and applied in order to find the best combination of variables to achieve the most accurate predictions. Several variables were extracted from the initial features, taken with APIs.

Here are the main variables extracted:

- *year:* year extracted from the original date.
- *month:* month extracted from the original date.
- *hour:* hours and minutes extracted from the original date.
- *temp:* temperature on that particular time.
- *windspeed:* the speed of the wind on on that particular time.
- *humidity:* the humidity in the air on that particular time.
- *season:* what time of season from 1 to 4 (being 1 spring and 4 winter).
- *day:* day of the week extracted from the original date.
- *peakhour:* reveals if that particular time has a lot of traffic or not.
- *weekday:* all days excluding weekends.
- *working:* reveals if it is a working day.
- *holiday:* reveals if the day is in a holiday.

The function *feature importances*, which is a function from Python's library *scikit-learn*, was used to measure the level of impact of each feature in the outcome of the predictions, as shown in FIG. 24, FIG. 25, FIG. 26 and FIG. 27.

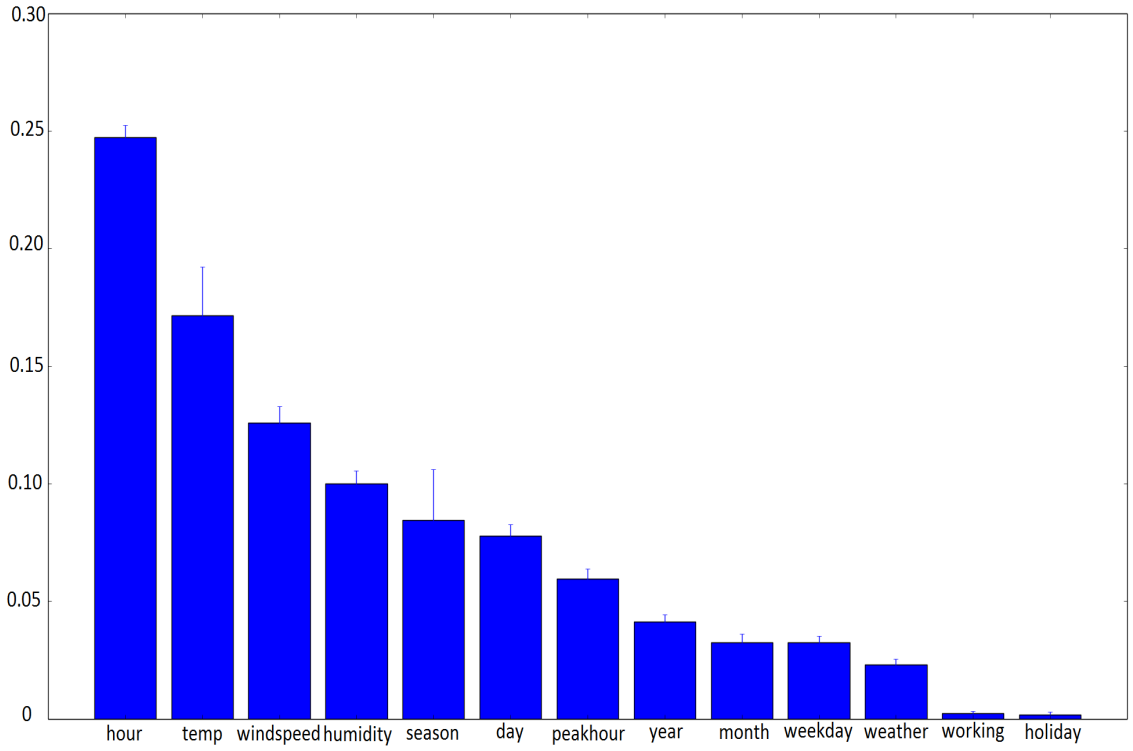

Figure 24: Importance of each feature being trained in Washington for casual users

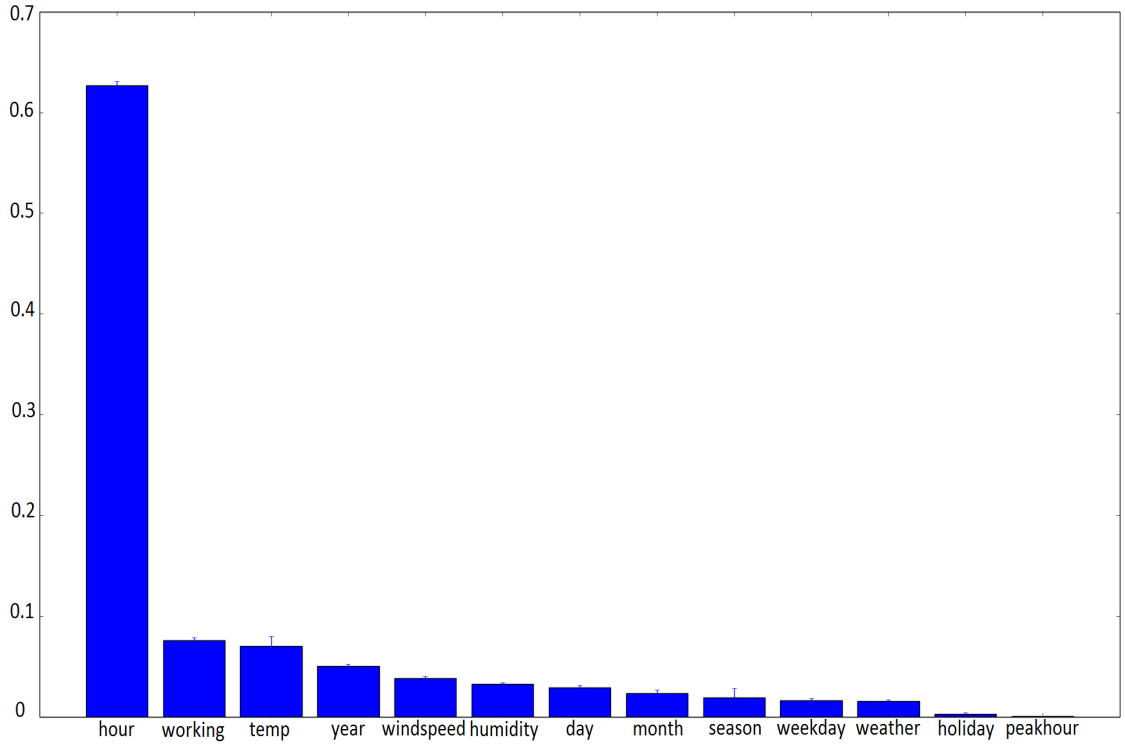

Figure 25: Importance of each feature being trained in Washington for subscribers

In FIG. 24 and FIG. 25 is expressed the importance of each feature that was separated from the initial variables that affect directly the behaviour of the customers and casual users of Washington BSSs. The hour variable is clearly the one that most affects the behaviour of the customers and casual users of this BSS. In casual users case temperature, wind-speed, humidity and the type of season follow as the most important variables to predict their behaviour. In the case of regular customers the fact that it is a working day, the temperature, year and wind-speed are the ones that most affect their behaviour.

FIG. 26 and FIG. 27 show the importance of the features for customers and usual users but in this case for Chicago BSS. It is possible to notice the interesting fact that for Chicago casual users the temperature is even more important the actual hour when it comes to riding a bicycle. The hour, wind-speed and the fact that it is a peak hour or not follow this main feature. For Chicago customers the most important features are the hour of the day, the temperature, wind-speed and the humidity in that moment.

Taking the analysis even to a deeper level, it is possible to see a clear distinction between the two types of users in both Chicago and Washington BSSs. One of the possibilities is that casual users dot not pay annual fees, so they do not feel so obligated to use the service and whenever they do, the time spent riding the bikes has to be paid so many other variable other than hour have a significant importance too. As for the customers, because they pay an annual fee, the most important

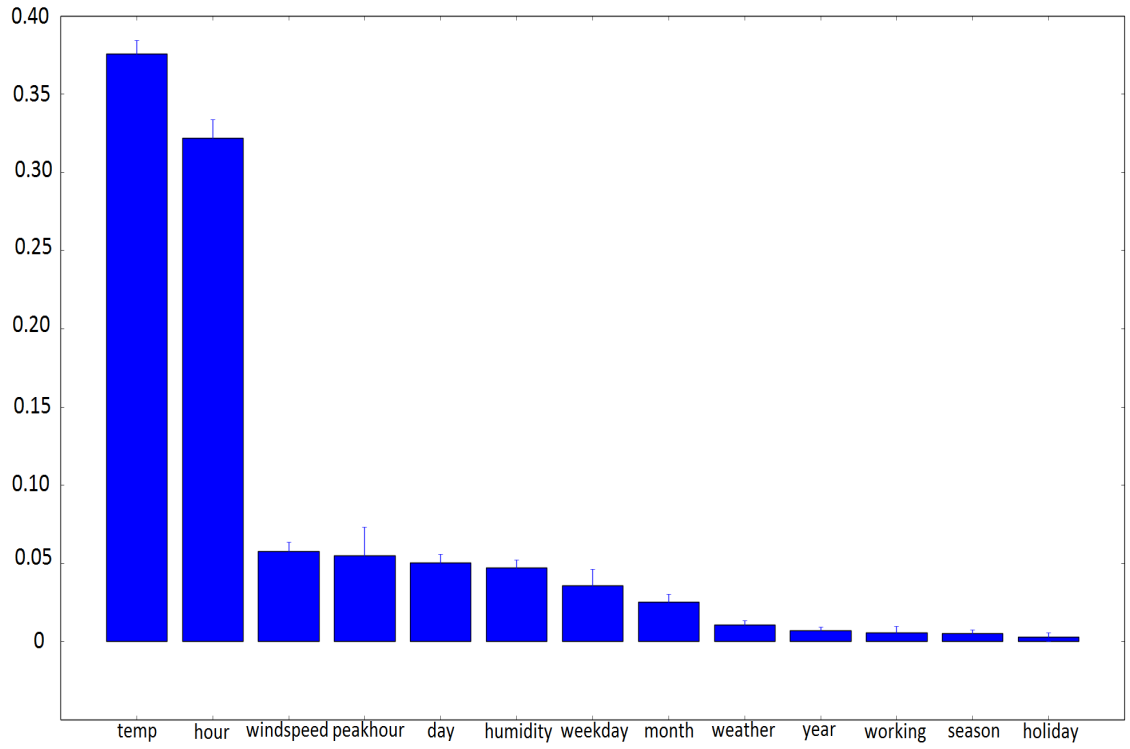

Figure 26: Importance of each feature being trained in Chicago for casual users

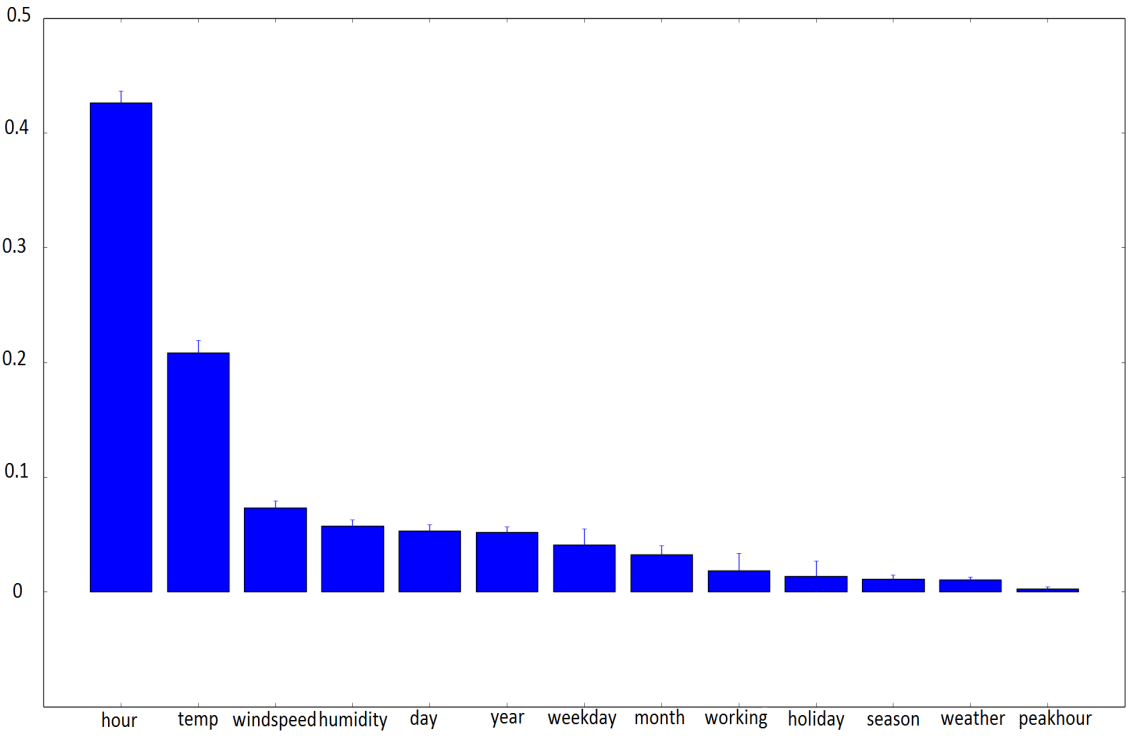

Figure 27: Importance of each feature being trained in Chicago for subscribers

feature is always the hour and others variables are much less meaningful.

For creating these diagrams regarding the importance of each feature in users behaviour, the function *feature importances* from Python's library *scikit-learn* was used. This procedure also helped in the process of choosing and filtering the most important features for each type of user belonging to both BSSs. This technique improved the quality of the finals predictions, for both casual users and customers, and adapted the variables in conformance with Chicago and Washington distinct patterns.

6.2 Dynamic Distribution Module

This Module represents the engineering part of the internship. It emphases on developing a software that after getting the predictions from the prediction module, it searches the stations that need to be rebalanced. Then it suggests the number of bikes that should be transferred from station to station and calculates the best routes in terms of distance and costs to the company responsible for the distribution of bicycles. At the same time it assures that efficiency will not be neglected. This distribution uses real distances and times between stations obtained with *Mapquest API*.

6.2.1 Data Acquisition

In this module all the distances and times between Chicago and between Washington stations were obtained. For that purpose *Mapquest API* was used due to the large number of requests this API endures and the good documentation and information it provides. Other rival APIs were discarded due to the fact that the majority had very a low limit of requests allowed per day. With this API 61425 requests were made for Washington and 44850 requests were made for Chicago. The number of requests needed were calculated using the following formula:

$$Requests = \sum_{n=1}^{n}(n-1) \quad (1)$$

In this equation n represents the number of stations.

6.2.2 Identification of the stations that need redistribution

The process to identify the stations that need redistribution on a BSS relies on six main conditions for each station:

- Prediction of departures and arrivals for one hour in the future.
- Future tendency for obtaining more departures or arrivals in the next two hours.
- Current number of bikes.
- Total number of available docks to park the bikes.
- Medium number of docks per station for each BSS.
- Mean Absolute Error for the time period of one hour for each BSS.

The first and the second conditions are trained and calculated through the Gradient Boosting Machines technique as explained in section 6.1.3. The third and the fourth variables are accessed through a XML feed in Washington BSS and through a JSON live feed in Chicago case. Finally the fifth and the sixth were calculated by quality tests elaborated in Chapter 7.

Two main lists are selected using the conditions mentioned above, the stations that need bikes and are in risk of getting empty and the stations that have too many bikes and are getting completely full. Several limits percentages were tested to know what would be the best lower and higher limit for the stations in order to execute the rebalancing of the bicycles and select the right stations that needed to be rebalanced. The best results were obtained by applying equations 2 and 3 for each station of the BSS.

$$HigherLimit = (x - x\frac{\frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|}{\bar{d}}) - 0.1x \tag{2}$$

$$LowerLimit = (x\frac{\frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|}{\bar{d}}) + 0.1x \tag{3}$$

In these formulas n represents the number of tested values, x is the number of total docks of that stations, f_i represents our prediction for the number of bikes in a station and y_i is the real number of bikes in the station. $\bar{d}$ is the medium number of docks for that particular BSS.

These limits are only applied to each station if the tendency for the next 2 hours remains the same. For example, if a station is almost reaching the lower limit but the tendency will be to gain bicycles naturally, it will not be rebalanced. Only if the prediction for the next 2 hours shows that it will continue losing bikes, the system will rebalance this station. This feature not only lowers the rebalancing costs, but it also lets the users contribute for the own stability of the BSS.

The overcrowded and under-crowded stations normally compensate each other, but in the case that there is only either full stations or empty stations left, the *depot* gives or takes bikes from and to the stations. *Depots* are big spaces that BSSs have to store substitute bikes for any case needed. The number each station will give or receive will estimated by the number of predicted bikes for the next two hours plus the number of bikes that station has below the lower limit or above the higher limit. Afterwards, the stations in need of redistribution and the respective number of bikes needed, are sent to the redistribution algorithm based on the *Jsprit Framework* in order to calculate the best routes possible to avoid big costs and assure good efficiency and availability.

6.2.3 Routes selection and costs/time calculation

Jsprit toolkit has been adapted to this problem in order to distribute the bicycles belonging to the BSS through out the stations, so that stations do not get empty or full. This way costumers can drop and pick bicycles from almost every station. This toolkit uses the algorithm created and tested by G. Schrimpf, which was proven to use one of the best algorithms to solve problems of Vehicle Routing Problems like this one. [22] This open-source toolkit is under General Public License (GNU), which means everyone is permitted to copy, use and change this algorithm.

One of the changes made in this toolkit was due to the fact that *Jsprit* only solved problems based on Euclidean distances, instead of taking in account real distances and real times between stations. Euclidean distances provided inaccurate information so in order to solve this problem *Mapquest API* was used to obtain all real distances and times between stations, simulating the use of a car or a van. A cost and time matrix was created with all the values of times and distances, this way a much more accurate information was attained.

Another feature added to this *toolkit* was that although the first concern remains to do the distribution of the bikes in an inexpensive way, saving in costs and in the number of vans distributing bikes, a function was created to ensure the van distributing the bikes must not take more than three times the time it would get the van to reach that station directly. This way the distribution algorithm will focus on saving costs but also ensures that time and fast distribution will not be neglected.

6.3 API Rest Module

An API module was developed with objective to create a easy and secure way of connecting all the Intelligent Rebalancing Bike System developed in this internship to all the other modules created by *Ubiwhere* staff. This API was developed in the programming language *JAVA* using *Spring Framework* and was documented using *Swagger*. The full documentation of this API is located in Appendix B.

There were a total five endpoints defined by the API to provide access to predictions and distributions made by the rebalancing system. All functions in this database return the predictions and distributions in the form of a JSON object. JSON has the advantage of being compatible with many languages, it is a lightweight format and very fast when being parsed.

There are three different types of functions to access the predictions. The user can access predictions made only with GBM with the *newpredictiongbm* function, which only include the sum between the predicted arrivals and the predicted departures for one hour in the future. The user can also access predictions where GBM predictions are joined with the real-time dock usage values and make a prediction based on that value for the future hour with the *newpredictionreal* function. All this predictions first verify if the date the user wants to predict is already stored in the database in order to avoid unnecessary calculations. Only if that date is not store in the database, the algorithm proceeds to the prediction algorithm. Lastly the user can also access past predictions for any specific date using *pastprediction* function. All predictions made with this API will be saved in Cassandra Database for statistical purposes.

Distributions can be accessed with two different types of functions. The *newdistribution* function first verifies if that distribution is already stored in the database. Only if the distribution is not stored in the database, the algorithm will advance to verify if the prediction for that particular date is already stored in database due to the fact that distributions are completely dependent on predictions. Afterwards, it makes the necessary calculations to provide the full routes suggested to the several vans doing the distribution in that particular BSS. In end the quantity of vans needed for distribution, total costs, total times and the routes details to be followed by each van will be displayed. All distributions made with this API will be saved in Cassandra Database for statistical purposes.

API security features were not implemented in this API because in this production environment the system of authentication, security and authorization will be developed only after the integration of all *Smart Bike Emotion* modules, and only then will be implemented common security measures for all the modules integrated in this Bike-Sharing Software being produced by *Ubiwhere*.

6.4 Database Management System

The Database shared by all modules already developed in *Smart BikeEmotion* is Apache Cassandra Database, so in order to maintain the same structure as the other modules Cassandra was also used in the modules being developed in this internship. Cassandra was also chosen because it is an open-source NoStructured Query Language (SQL) and Non-relational Database that follows a peer-to-peer architecture and has elastic scalability, hight availability and fault tolerance. In terms of scalability Cassandra proved to be the best between the NoSQL databases. [27]

Cassandra uses Cassandra Query Language (CQL) formal language in order to represent the queries for information retrieval systems.

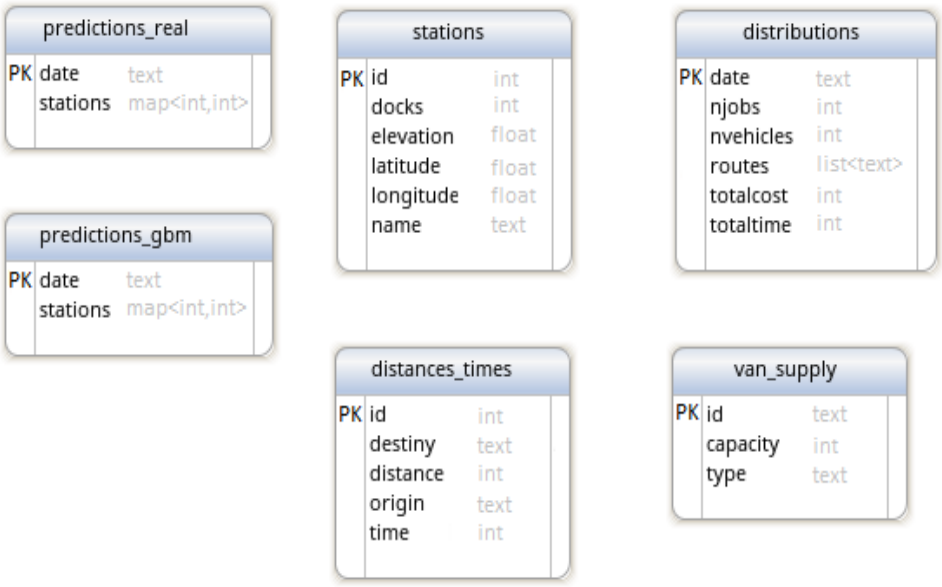

Figure 28: Cassandra Entity-Relationship Diagram for each Keyspace that represents one BSS

In FIG. 28 is possible to see the tables created in order to accommodate all the data created by the predictions and the distributions in a reliable and fast way. In this FIG the *PK* indicates the primary keys of each table. The table *stations* has all the information regarding the stations information of that BSS , *predictions_real* has the information of the prediction of bikes for each station joined with the real-time number of bikes at the moment, *predictions_gbm* stores information about the sum of predicted arrivals and departures for each station, *distances_times* stores all distances and times between the stations of that particular BSS, distributions, like the name suggests, stores all the distributions made in the past by the system so

unnecessary calculation would have to be made. Finally *van_supply* table provides us the quantity of vans that BSS has and the capacity of each van.

The database was designed in order to each *keyspace* represents one different BSS. So a total of two different *keyspaces* for Washington and Chicago were created. *Keyspaces* allow to the full structure to be copied in a fast and easy way.

7 Tests and Validation

This chapter contains all the tests performed regarding each module, in order to guarantee the quality of the final product. Tests are divided into two main groups. Firstly functional tests are presented, where results obtained by the system are analysed and compared to what was expected from the system. In a second phase non-functional tests are performed. These tests aim to validate the behaviour of the system when it comes to non-functional requirements.

7.1 Functional Tests

7.1.1 Unit and Integration Tests

Unit tests consist on individual tests of source code to ensure that code meets its design and behaves as intended. This type of testing provides several benefits such as finding problems early, simplifies integration and facilitates change within the code.

During the development phase, several modules were tested. Many errors were corrected in methods in sub-routines during the development. This corrections avoided errors to become even bigger flaws and have a greater impact in the final product. For the realization of this unit tests an input and an expected output were prepared for each module. In the cases that the output did not match the expected output, necessary corrections were made in the code.

In a more advanced stage of development, when many modules were already implemented, it was necessary to test how modules interacted between them and if that interaction was made without any flaws and in accordance with the expected. For this stage an *bottom-up* approach was adopted where modules from previous stages were integrated and tested consecutively. These tests were not extended to other external modules of *Smart Bike Emotion* due to the fact that many modules are still being developed by others collaborators in *Ubiwhere* and are not ready to join and make a full integration test.

7.1.2 System Tests

After the algorithms have been implemented, it is possible to skip to the next phase of validation, the system tests. In this type of tests the requirements defined initially are confronted with the functionalities of the developed system. Each system test was made having the Table 10 as template.

[TC01] - Example Test Case

Test case	TC01
Description	Describes what is going to be tested.
Requirement	The functional requirement that is attached to this system test
Expected Results	Results that the system is expected to produce during the test.
Final Result	Pass or Failed

Table 10: [TC01] - Example Test Case

In Appendix C are gathered all system tests performed in accordance with the same structure of this diagram. The completion of this validation process allowed to conclude that the features implemented meet the requirements stipulated and only finding small flaws in some features that were corrected and that same system test was executed again.

7.2 Non-Functional Tests

7.2.1 Quality Tests

To assure that all the implemented software would create reliable predictions and meaningful redistribution recommendations, quality tests were performed to the system. In Appendix D are located all the quality tests that were performed to the system.

Bike Prediction Module

In this module quality tests were made using different datasets, from different locations and from different time periods. In order to reduce over-fitting and to make the validation of the machine learning techniques more accurate a 10-fold Cross-Validation validation technique was used. A 10-fold Cross-Validation divides the data into 10 training datasets and 10 testing datasets. Then the system is trained using each of these combination of datasets to make a prediction in the locations, Washington and Chicago, and in all time periods that are one, two and three hours

in the future. The mean value of the difference between the real values and the predicted values for each one of those 10 prediction datasets produced the results shown in the tests that will follow. This validation technique was made in order to improve the final statistics.

Due to the fact that Chicago BSS is a younger system, it had less information regarding the past trips of the users than Washington. In Chicago the past trips training information holds two years of trips and for the test cases three months were applied. Washington BSS had much more information regarding user past trips, so four years of past trip information were trained and also three months were applied for the tests.

The quality tests for prediction were made for one, two and three hours of intervals. Lower time periods were excluded due to the fact that in big cities like Washington and Chicago each van would not have sufficient time to make the distributions.

The metrics used to measure the accuracy of the four techniques of prediction were Root Mean Squared Logarithmic Error (RMSLE) and Mean Absolute Error.

RMSLE is calculated using the following equation:

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(r_i+1))^2} \quad (4)$$

In this Equation n represents the number of tested values, p_i represents our prediction for the number of bikes in a station and r_i is the real number of bikes in the station.

Mean Absolute Error is calculated using the following equation:

$$MeanAbsoluteError = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| \quad (5)$$

In this Equation n represents the number of tested value, f_i represents our prediction for the number of bikes in a station and y_i is the real number of bikes in the station.

Due to not having past real-time values of the number of bikes in each station at each different date, that data was created through the number of departures and arrivals in each station. The starting number of bikes for the all stations started at half the capacity of each station, so that no differences between them would emerge.

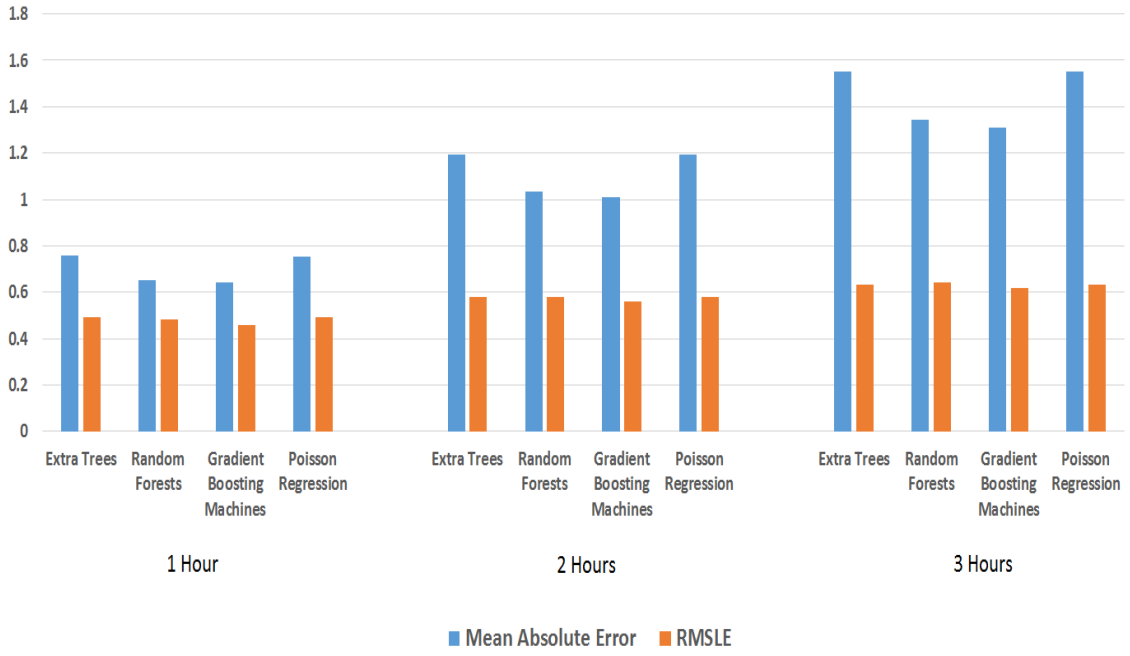

Figure 29: RMLSE and Medium Mean Error for every station with four prediction techniques for 1, 2 and 3 hours in Chicago

In FIG. 29 and in FIG. 30 are the results obtained using the selected metrics to evaluate the prediction accuracy to one, two and three hour predictions both in Chicago and Washington BSSs. In these diagrams is possible to analyse that the two techniques that stand out with betters results, both in the Mean Absolute Error and in the RMSLE, are the Gradient Boosting Machines and the Random Forests. GBM had the best performance overall and Extra Trees has the worst performance overall. It is also possible to see that as the hour prediction gets higher, the Mean Absolute Error and the RMSLE also get theirs values aggravated. RMSLE penalize under-predictions greater than over-predictions and makes a prediction on a logarithmic scale, which means whatever the outcomes possible in the prediction this value is not affected by it. The value is used as a reference is several prediction studies.

Taking into account that 16 bikes per station is the average number of docks for each Washington station, Mean Absolute Error results shows that a prediction made with GBM for 1 hour only misses the prediction by 0.89 bikes by average, which represents a prediction mean error percentage of around 5.6%. Chicago BSS has an average number of 17 bikes for each station. With average Mean Absolute Error of 0.6 for GBM prediction for 1 hour this represents a mean error percentage of around 3.53%, as represented in FIG. 31 and FIG. 32.

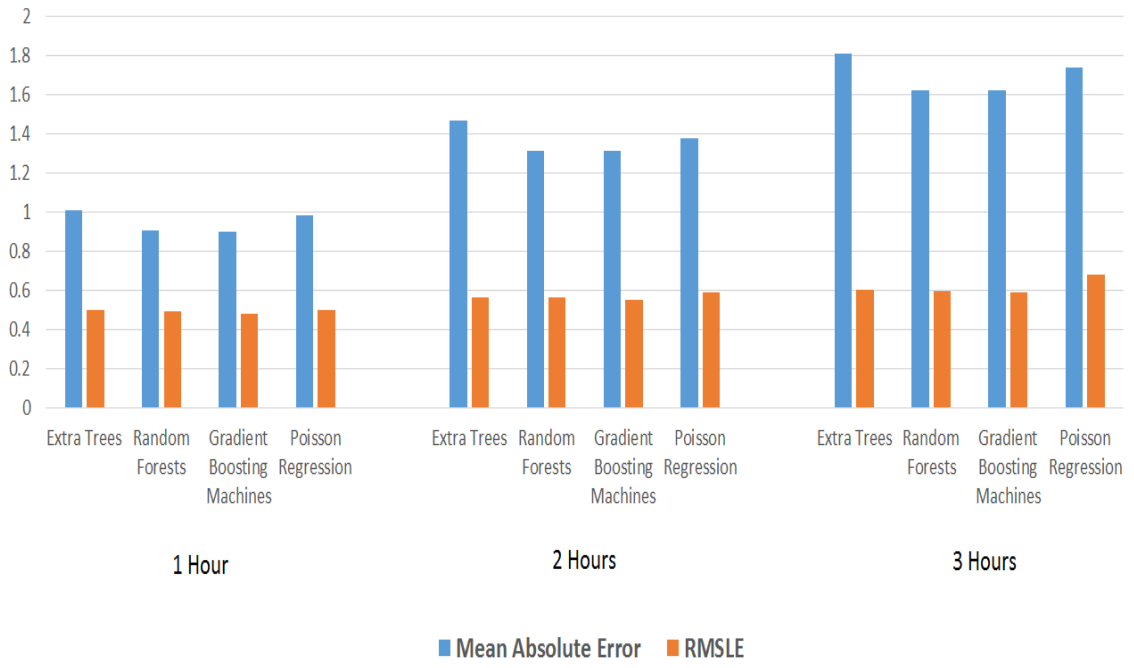

Figure 30: RMLSE and Medium Mean Error for every station with four prediction techniques for 1, 2 and 3 hours in Washington

Figure 31: Mean Error Percentage for the several prediction techniques for 1, 2 and 3 hours in Chicago

Chicago and Washington BSSs despite being about the same size, they have very different patterns of learning. After analysing all the quality test statistics is possible to see that overall Chicago BSS dock usage prediction got better results in all metrics.

When it comes to the interpretation of this results, it is possible to compare them to

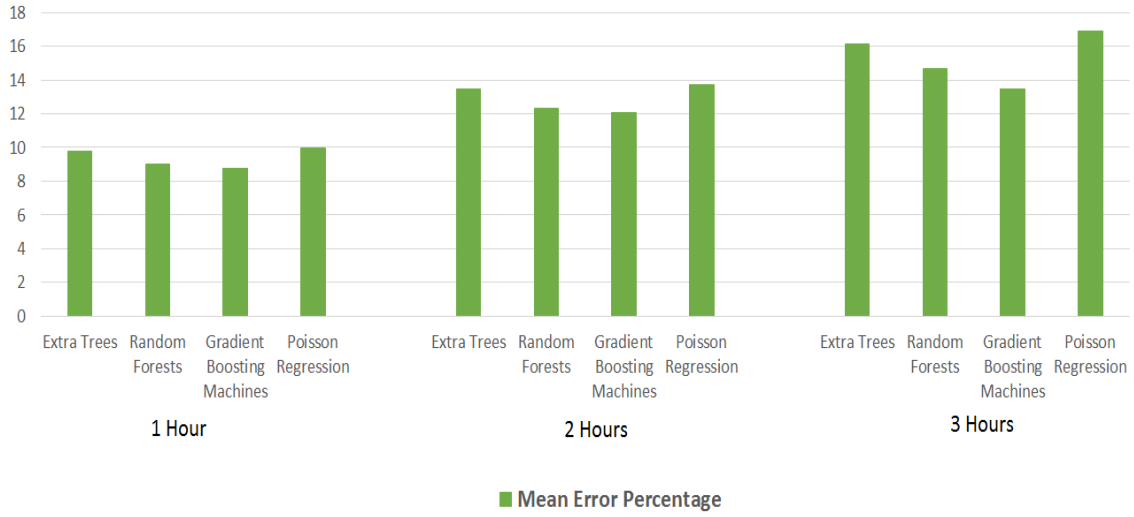

Figure 32: Mean Error Percentage for the several prediction techniques for 1, 2 and 3 hours in Washington

other studies made in this Bike-Sharing dock usage prediction theme. In terms of 1 hour prediction it is possible to conclude that they are in line with the studies already made. Regue and W. Recker made a 60 minute dock usage prediction using GBM for Boston BSS and got a mean error percentage of 11.2% [17]. So a improvement of 7.67% when comparing with our tests for 1 hour period in Chicago BSS and an 5.6% improvement when comparing to our tests for 1 hour period in Washington BSS. With this results in mind is possible to say these predictions are in fact a positive and consistent result, for both *Chicago* and *Washington* BSSs, and that the better machine learning for dock usage prediction in both BSS is GBM and the best time period is 1 hour.

Bike Rebalancing System

After the best prediction technique and the best time period were selected in the prediction quality tests, more quality tests involving the prediction module and the distribution module together were made to evaluate if the hole system would provide good rebalancing instructions to the employees. Due to not having access to real-time values from the past in Washington and Chicago BSSs, the data of past trips was used to simulate how many bikes would arrive and departure from each station every hour and how many bikes were in each station. This way, there would be no conflicts with the own redistribution system each BSS already had. These tests were simulated for 3 months using the 10 prediction datasets created in Bike Prediction Module quality testing for each location, but now only using GBM as the prediction technique for docks usage and 1 hour time as the standard prediction because they obtained better results in the quality tests performed.

In FIG. 33 there is a graph that provides the percentage of empty and full stations measured every hour during three months. Using our rebalancing method the per-

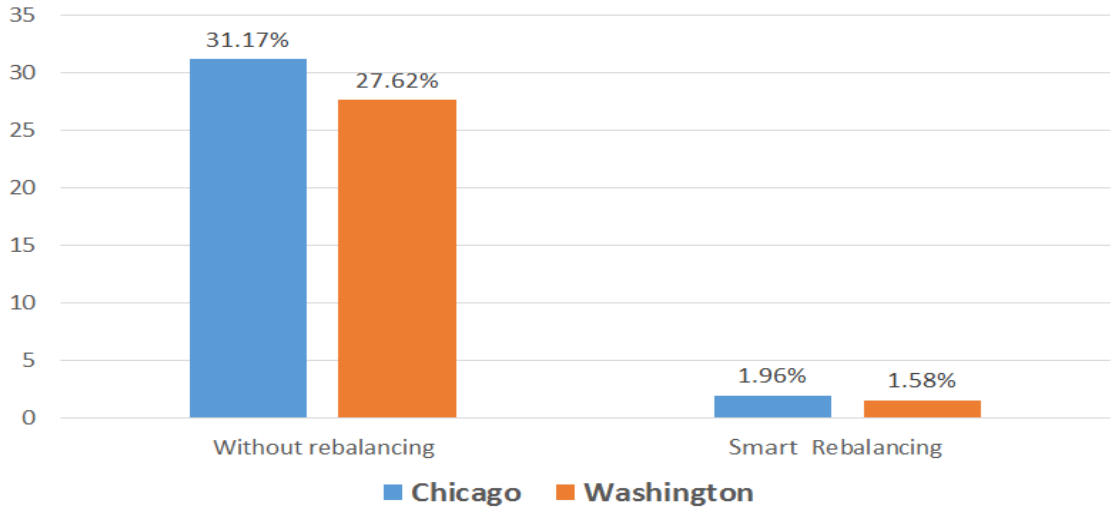

Figure 33: Mean percentage of empty and full stations measured every hour in Chicago and Washington BSSs

centage of empty and full stations dropped dramatically. These numbers include all stations in both BSSs and reveal the stations that were unbalanced and lacked bikes or were completely full, making theses stations inoperable. The objective here was not to know the percentage of time each station was empty or full but to have a global perspective of the stations that remain unbalanced for longs periods of time.

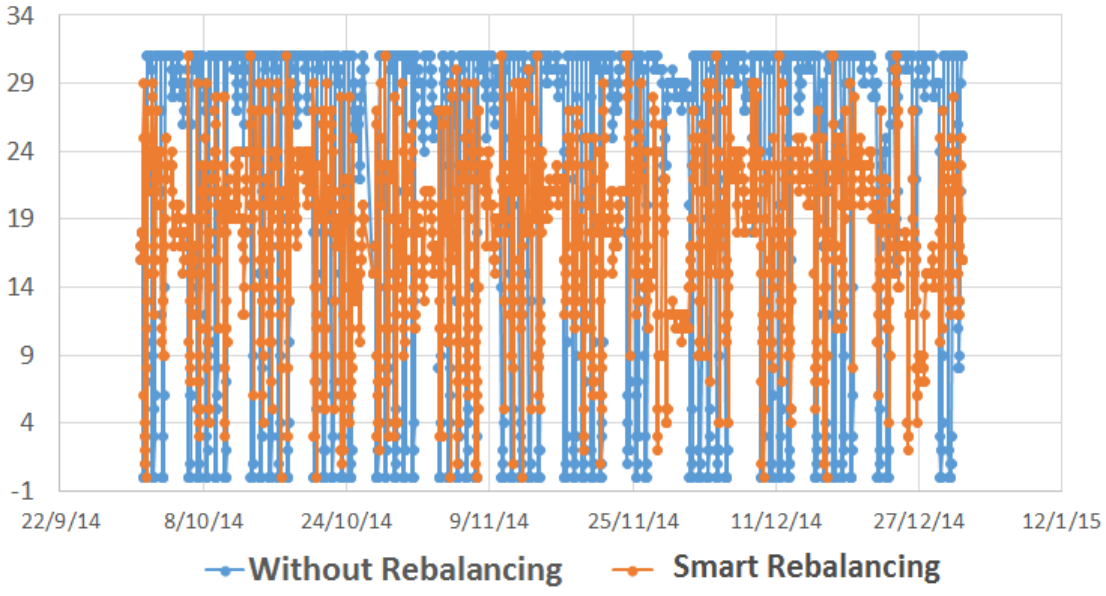

Figure 34: Number of bikes measured every hour in one the the busiest station in Chicago, Clinton Street & Washington Boulevard Station on specific dataset

In order to do the rebalancing of the bikes within the cities, the system suggested a

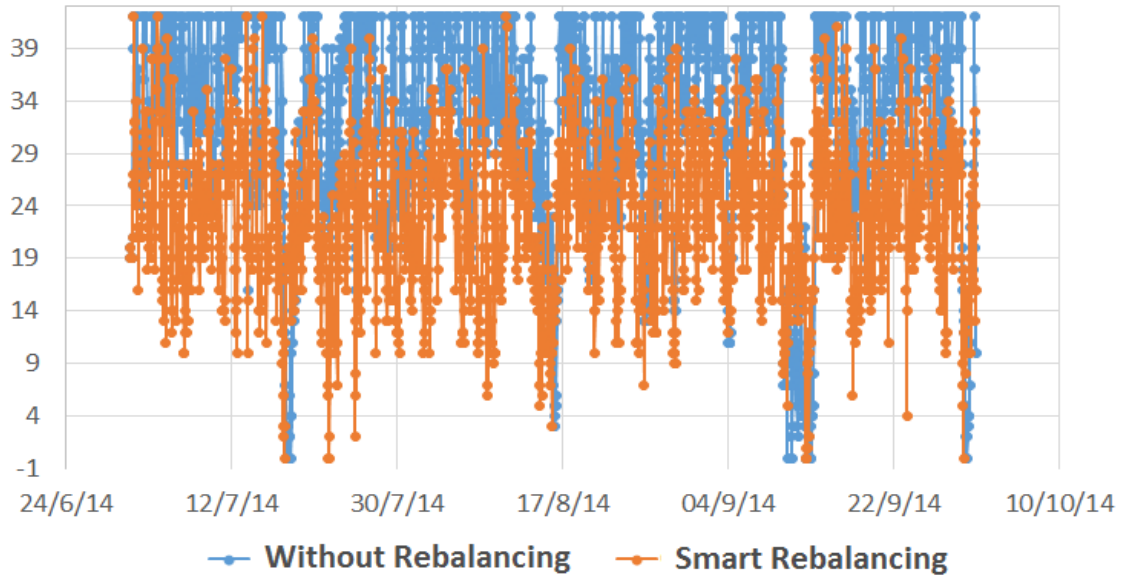

Figure 35: Number of bikes measured every hour in one the the busiest station in Washington, Dupont Circle Station on specific dataset

mean value for each hourly distribution of 10.3 vans in Washington and 8.8 vans in Chicago, the medium quantity of stations rebalanced per hour was 41.2 stations for Washington BSS and 35.9 stations for Chicago BSS and the medium distance covered during each distribution was of 27357.4 meters in Washington and 23875.1 meters in Chicago. All these hourly distributions were made in less than one hour.

In FIG. 34 and in FIG. 35 is the comparison of the number of bikes in two of the busiest stations in both Chicago and Washington BSSs without using any rebalancing system and the rebalancing system developed in this internship. In *Clinton Street & Washington Boulevard Station* the percentage of empty and full stations was of 3.2% and in *Dupont Circle Station* was of 2.1%. Both this stations despite having a very big number of departures and arrivals, they managed to present good results with our Smart Rebalancing System. *Clinton Street & Washington Boulevard Station* if it was not rebalanced it would have a 57.6% of empty and full stations, making it almost inoperable.

When comparing this rebalancing system with real-time systems of rebalancing being used in Washington and in Chicago. Only Washington BSS provides the information that in May of 2015 had a 17.08% percentage of stations that remain empty more than 60 minutes and a 11.27% percentage of stations that remain full also more than 60 minutes. [11] So despite the fact that this results are in different time periods, our system seems to keep stations operable for almost more than 12.04% of the time in relation to the system currently being used in Washington.

Test id		
Number of runs:	Times each test is executed	
Number of errors:	Number of errors during execution	
Number of Pickups and Deliveries:	Number of trips present in the test	
	Jsprit	**Best Know Value**
Vehicles used	x	y
Distance travelled	x	y
Running time	x	y

Table 11: Lim benchmark performance test example

7.2.2 Performance Tests

In order to validate some non-functional features of the application, such as scalability, some performance testes were performed. These tests were performed on the redistribution module because it tends go get heavy when there are a big numbers of stations to rebalance. The complete performance tests that were executed are located in Appendix E.

To test the redistribution module benchmark instances defined by Li, H. and Lim, A were used. This benchmark is "A meta-heuristic for the pickup and delivery problems with time windows." [28] These tests also include the best know solutions for each test on this benchmark until now. This benchmark gives priority to two main features: minimize number of vehicles doing the redistribution and minimize the total travelled distance by the vans.

TAB. 11 represent an example of template used to perform these benchmark tests. Each test was executed 10 times for 100 , 400 , 800 and 1000 trips. The mean results were added to these tests. This benchmark tests were performed with a i7 - 3635 CPU 2.40GHZ Central Processing Unit (CPU) with 8 Gigabytes of Random Access Memory (RAM).

The results of these tests prove that *Jsprit Toolkit* is very robust tool and although it needs many resources to run, it is scalable until a good fair amount of trips. To have a better idea the total number of stations of Washington BSS is 350, and the number of corrections is always much lower, so this tool would work for a much bigger BSS without any problem. This benchmark tested the system until 1000 trips of redistribution, which means the system will have to have at least 2000 stations, and performed very well taking in mind that this type of distribution will only be performed 1 time per hour in a server with much better specifications than the one used in this tests. Using a i7 - 3635 CPU 2.40GHZ CPU with 8 Gigabytes of RAM the maximum time spent running the distribution within all tests was of 605 seconds.

8 Conclusion

This chapter marks the end of one year internship at *Ubiwhere*. The first section presents the final remarks about the work done during this internship and the second section explains the future work that can be done to improve certain features.

8.1 Final Remarks

This internship has been a whole new experience, full of challenges and lessons learned. Since the beginning of the internship the goal was to present a innovative way to do the rebalancing of bikes in BSS by exploring machine learning techniques and combine it with some fast redistribution methods. With this goal in mind several functional requisites were presented in the initial phase of the project:

- *Prediction of Bicycles Stations Future State:* try to predict, through the help of an intelligent algorithm, the approximate number of bicycles in each dock on different time windows.
- *Dynamic Bike Redistribution:* instruct the employees how to distribute the bikes and the best sequence to do it.
- *Develop an API:* develop an API to connect to other internal modules in order to retrieve information from the bikes, employees and stations.
- *Include location and time as a variable:* during the calibration of the system take in account the different locations of the stations and the current time.
- *Diagnose the most problematic stations:* do an overall evaluation of the stations hourly and analyse what are the stations that have an excess of bicycles or lack of bicycles .
- *Include weather as a variable:* during the calibration of the system take in account the current weather.
- *Include city topology as a variable:* while the system is being calibrated take in account the topology of each station.

The first step was to analyse several machine learning techniques, vehicle routing distribution algorithms and software that could help to build an efficient and productive Bike-Sharing System, lowering the high costs that are associated with such operations. Then a methodology was adopted and planning strategy was defined based on the Royce's Final Waterfall model. In this phase a risk mitigation plan was also made to anticipate some uncertain events that could affect this project and be able to overcome them or even prevent them.

The implementation phase followed where three main modules were defined: Dock usage prediction module, dynamic distribution module and the implementation of an API. In the dock usage prediction module,which represented the investigation and research part, several machine learnings were implemented, trained and tested in order to compare them in different locations, different hour periods and with several datasets for each test. This involved the implementation of Extra Trees, Gradient Boosting Machines, Poisson Regression and Random Forests. Dynamic prediction module, which represents the engineering part, makes the identification of the stations that need redistribution and how many bikes each station needs. *Jsprit Toolkit* was also integrated and adapted to this module so that distribution would take the best routes possible to minimize operating costs while keeping the stations operable. Both this modules were combined together in a API based in REST so that all this features can be accessed easily and in a secure way.

The last phase of this project was the test and validation of the system. Part of this validation occurred during the implementation of the code through the execution of unit and integration tests to each unit module developed. To ensure that the system fulfilled every functional requisites system tests were performed. Then several quality tests were made to the system to assure that all the implemented software would create reliable predictions and meaningful redistribution recommendations. Finally performance tests were performed to validate some non-functional features of the application such as scalability.

By performing an brief analysis of all the work done during this internship it is possible to perceive that the objectives that were set in the beginning were fully achieved. All the tests performed proved that the prediction of bikes can be done for 1 hour period using GBM machine learning technique with low percentage of error giving the number of bikes in each station. As for the redistribution system the tests responded positively both in quality and performance by getting very good numbers in the benchmarks done. This modules were also quality tested together and results proved that the Bike Rebalancing System developed is able to keep the system stable with very reduced numbers of empty or full stations.

In addition to having accomplished the defined functional objectives, it is considered that the work developed in this internship will help to improve the *Smart Bike Emotion* project by giving it a innovative functionality.

8.2 Future Work

BSSs are highly dynamic and riders behaviour is difficult and complex to predict so there all always existing opportunities to improve certain features.

Firstly all the work developed in this internship was back-end software, so all the implemented code will have to be joined with a front-end code that will have to find creative and easy ways to present to the employee to interpret all the data created.

The Bike Rebalancing System developed in this is only a part of a very big software production, *Smart BikeEmotion*, that involves a lot of different developed modules and modules yet to be developed, so a big integration work will have to be done to join all these modules.

It would be very interesting to try to adapt the developed Rebalancing System in a real environment. This would allow to perform more tests and would give the opportunity to improve the prediction and distribution algorithms even further.

References

[1] J. Longhurst and C. Brebbia, "*Urban Transport and the Environment in the 21st Century*," 2012.

[2] R. Meddin, "`www.bikesharingworld.com`," January, 2015.

[3] B. V. Bicing, "`https://www.bicing.cat/`," 2015.

[4] C. D. Bikes, "`https://www.divvybikes.com/`," 2015.

[5] T. for London, "`https://tfl.gov.uk/modes/cycling/`," 2015.

[6] N. Y. CitiBike, "`https://www.citibikenyc.com/`," 2015.

[7] P. V. System, "`http://www.velib.paris/`," 2015.

[8] W. C. Bikeshare, "`https://www.capitalbikeshare.com/`," 2015.

[9] Esri, "`http://www.esri.com/`," 2012.

[10] Netakil, "`https://logvrp.com/logvrpsite/Default.aspx`," 2010.

[11] L. Consulting, "*Capital Bikeshare Member Survey Report*," 2014.

[12] Fricker and Gast, "*Incentives and regulations in bike-sharing systems with stations of finite capacity*," 2012.

[13] L. Caggiani and M. Ottomanelli, "*A Modular Soft Computing based Method for Vehicles Repositioning in Bike-sharing Systems*," 2012.

[14] L. Gaspero, A. Rendl, and T. Urli, "*Balancing Bike Sharing Systems with Constraint Programming*," 2014.

[15] P. Geurts and G. Louppe, "*Learning to rank with extremely randomized trees*," 2011.

[16] J. H. Friedman, "*Greedy Function Approximation: A Gradient Boosting Machine*," 2001.

[17] R. Regue and W. Recker, "*Using Gradient Boosting Machines to Predict Bike-sharing Station States*," 2014.

[18] C. D. of Transportation, "Data science for social good," 2013.

[19] M. Dell'Amico, E. Hadjicostantinou, and N. S. Iori, M., "*The bike sharing rebalancing problem: Mathematical formulations and benchmark instances*," 2014.

[20] Mjc2, "`http://www.mjc2.com/`," 1990.

[21] S. Schröder, "`http://jsprit.github.io/`," 2014.

[22] G. Schrimpf, J. Schneider, H. Stamn-Wilbrandt, and G. Dueck, "*Record Breaking Optimization Results Using the Ruin and Recreate Principle*," 2000.

[23] M. Kuo, "`https://github.com/mck-/Open-VRP`," 2012.

[24] R. Hat, "`http://www.optaplanner.org/`," 2006.

[25] P. Midgley, "*Bicycle-sharing schemes: Enhancing Sustainable Mobility In Urban Areas*," 2011.

[26] M. Kuhn and K. Johnson, "*Applied Predictive Modeling*," 2013.

[27] T. Rabl, M. Sadoghi, H.-A. Jacobsen, S. Gomez-Villamor, V. Muntes-Mulero, and S. Mankovskii, "*Solving Big Data Challenges for Enterprise Application Performance Management*," 2012.

[28] H. Li and A. Lim, "*A Metaheuristic for the pickup and delivery problem with time windows*," 2001.

Appendix A - Internship Meetings (5/15 reports)

Appendix B - REST API Documentation

Appendix C - System tests

Appendix D - Quality tests

Appendix E - Performance Tests