Dynamic Pricing Communication

STEVEN LY
Dynamic Pricing Communication

Steven Ly

2018-06-07

Master’s Thesis

Examiner
Gerald Q. Maguire Jr.

Academic adviser
Anders Västberg

Industrial adviser
Robert Petersen, EasyPark

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science (EECS)
Department of Communication Systems
SE-100 44 Stockholm, Sweden
Abstract

Parking is an old concept, which fundamentally involves leaving a vehicle at a place. Parking has been considered as a subsidiary activity to owning a car. However, these days owning a car has become the norm, which leads to a greater demand for parking. Unregulated parking demand often leads to increased traffic congestion, when there are not enough parking spaces to keep up with the demand. Congestion itself has a negative impact on the environment and causes safety issues. A common solution to reduce congestion have been by influencing the demand for parking spaces through parking prices. During recent years, the existing pricing strategies have not been able to keep up with the daily changes in demand. Therefore, stakeholders in the parking industry have started to shift towards working for dynamic pricing.

Dynamic pricing utilizes a pricing strategy that sets the price according to the current demand and occupancy. However, the parking industry is missing a key feature to fully enable dynamic pricing. There is no communication standard in the parking industry. Thus, there is no efficient communication mean for the stakeholders to share their parking-related information (such as location, occupancy, and tariff data). This thesis has developed and proposes a protocol for sharing such parking-related information. The aim is that the protocol will be used as a communication standard in the parking industry. Due to limited time, the most focus was put on completing the protocol for tariff data. However, the developed protocol can be considered as a partial solution towards dynamic pricing. Because the protocol can still be used to properly share tariff data.

Based on the evaluation, the protocol could express a variety of tariffs. The tariffs that are expressible have use cases such as early bird, residential, or on-street parking. To make integration easier, for the parking industry, the protocol includes tools to aid integrations of the protocol. A future work will be to complete the support of location and occupancy related data. Additionally, it has been discussed that the protocol will onwards be developed as open-source.

Keywords

Data sharing, Dynamic pricing, Parking industry, Parking tariffs, Protocol
Sammanfattning


Nyckelord

Datakommunikation, Dynamisk prissättning, Parkeringstariffer, Protokoll
**Acknowledgments**

First of all, there is a couple of acknowledgment I would like to do. There are several people that have played a major role in helping me completing the thesis, by providing me with feedback and advice on the work.

The first person I like to express my gratitude for, is my examiner from KTH, Professor Gerald Q. Maguire Jr., without his support and advice I would not have been able to complete this thesis. Therefore, I want to thank him for always being there to help me and provide me with the feedback I need. His detailed and comprehensive feedback has been vital towards completing and improving the work.

I would also like to express my gratitude to EasyPark and my industrial adviser, Robert Petersen. It was Robert who proposed the project, and thus, the project would not have existed without him. He has always been available and providing me with advice and the necessary knowledge to complete the thesis. I also want to thank EasyPark at whole for providing me with a perfect working environment as well as a location to work on my thesis.

Furthermore, I also want to thank Rachel Yoka, at international parking institute, for providing me with access to information regarding IPI’s own protocol project. I am also thankful for IPI providing open access to their research database, which has been useful in providing me a better idea of the parking industry.

Stockholm, June 2018

Steven Ly
# Table of contents

Abstract .................................................................................................................. i  
Keywords.................................................................................................................. i  
Sammanfattning ................................................................................................... iii  
Nyckelord............................................................................................................... iii  
Acknowledgments ............................................................................................... v  
Table of contents ................................................................................................. vii  
List of Figures .................................................................................................... ix  
List of Tables ........................................................................................................ xi  
List of acronyms and abbreviations ................................................................... xiii  
1 Introduction ...................................................................................................... 1  
   1.1 Background .................................................................................................. 1  
      1.1.1 Sustainability ....................................................................................... 1  
      1.1.2 Privacy ................................................................................................ 2  
   1.2 Problem statement ....................................................................................... 2  
   1.3 Goals ........................................................................................................... 3  
   1.4 Methodology ............................................................................................... 3  
   1.5 Future work and limitations ....................................................................... 4  
2 Background ..................................................................................................... 5  
   2.1 Parking prices ............................................................................................... 5  
      2.1.1 Traditional tariffs ................................................................................ 6  
      2.1.2 Dynamic Pricing ................................................................................ 6  
   2.2 Technologies ............................................................................................... 8  
      2.2.1 Devices in the parking industry ......................................................... 8  
      2.2.2 Communication technologies ......................................................... 9  
      2.2.3 Parking data APIs .......................................................................... 14  
   2.3 Related work ............................................................................................... 15  
      2.3.1 The proposed standard for parking information sharing ............. 15  
      2.3.2 Added support for parking information sharing ......................... 20  
   2.4 Conclusions of the literature study ............................................................ 24  
      2.4.1 Data fields ......................................................................................... 25  
      2.4.2 Metrics .............................................................................................. 27  
      2.4.3 Roles ................................................................................................ 28  
      2.4.4 Use cases ......................................................................................... 28  
      2.4.5 Terminology ..................................................................................... 29  
3 Protocol development ...................................................................................... 31  
   3.1 Development Stages .................................................................................... 31  
      3.1.1 Initial format ...................................................................................... 31  
      3.1.2 Tools for writing and parsing ............................................................ 31  
   3.2 Development challenges ............................................................................ 32  
      3.2.1 Wrong data format .......................................................................... 32
# Table of contents

3.2.2 Time shortage .............................................................................. 32

4 Protocol description ........................................................................... 35

4.1 Format description ........................................................................... 35
  4.1.1 Location standard ....................................................................... 35
  4.1.2 Occupancy standard ................................................................... 36
  4.1.3 Tariff standard ........................................................................... 37

4.2 Tools description ............................................................................... 42
  4.2.1 Writing tool ................................................................................. 43
  4.2.2 Parsing tools ................................................................................. 43
  4.2.3 Conversion tools .......................................................................... 43

4.3 Interface description ......................................................................... 44
  4.3.1 Format and query schema ............................................................ 45
  4.3.2 Server application ........................................................................ 45

5 Evaluation and discussion ................................................................... 47

5.1 Format evaluation ........................................................................... 47
  5.1.1 Location and occupancy data ..................................................... 47
  5.1.2 Tariff data .................................................................................. 48
  5.1.3 Correctness tests ......................................................................... 48

5.2 Interface evaluation ......................................................................... 54
  5.2.1 Conversion speed test ................................................................. 55
  5.2.2 File compression test ................................................................. 57
  5.2.3 Aspects outside the protocol control ............................................ 58

6 Conclusions ......................................................................................... 61

6.1 Dynamic pricing and tariffs .............................................................. 61
6.2 The developed protocol .................................................................... 62
6.3 Ethics and Sustainability .................................................................... 63

6.4 Future work and challenges ............................................................. 63
  6.4.1 Challenges .................................................................................. 64
  6.4.2 Future work ................................................................................. 64

References .............................................................................................. 65

Appendix A: GraphQL schema ................................................................ 71
List of Figures

Figure 1: Example of a possible XML document ........................................ 10
Figure 2: Example of an external DTD ....................................................... 11
Figure 3: Example of a possible JSON object ........................................... 12
Figure 4: Example of a GraphQL schema ................................................. 13
Figure 5: Example of a GraphQL query ..................................................... 13
Figure 6: Class diagram of location data ................................................... 25
Figure 7: Class diagram of occupancy data .............................................. 26
Figure 8: Class diagram of tariff data ....................................................... 27
Figure 9: Class diagram of location data ................................................... 36
Figure 10: Class diagram of occupancy data .......................................... 37
Figure 11: Class diagram of tariff data ..................................................... 38
Figure 12: Example of a tariff defined for one day (SEK) ......................... 39
Figure 13: Example of a tariff with mixed rates (SEK) ......................... 40
Figure 14: Example of a tariff defined for one week (SEK) ..................... 40
Figure 15: Example of active schedules ............................................... 41
Figure 16: Example of valid schedules .................................................. 42
Figure 17: Example of a flatten schedule .............................................. 43
Figure 18: Chart showing conversion time taken from 100-1000 locations (incremented by 100 locations) ......................... 57
List of Tables

Table 1: Important elements in area domain (Facility location) [63] .......... 17
Table 2: Important elements in contact domain (Facility location) [63] ..... 17
Table 3: Important elements in location domain (Facility location) [63] .... 17
Table 4: Important elements in layout domain (Facility location) [63] ...... 17
Table 5: Important elements in demand domain (Occupancy) [64] .......... 18
Table 6: Important elements in demand space domain (Occupancy) [64] .. 18
Table 7: Important elements in location domain (Occupancy) [64] .......... 18
Table 8: Important elements in supply domain (Occupancy) [64] .......... 19
Table 9: Important elements in the area domain (Parking rate) [64] .......... 19
Table 10: Important elements in the ancillary domain (Parking rate) [64] .. 20
Table 11: Important elements in the rate table domain (Parking rate) [64] ................................................................. 20
Table 12: Important elements in rate construction (Parking rate) [64] ..... 20
Table 13: Important elements in GeoPointType (Location) [66] .......... 21
Table 14: Important elements in AdditionalGeoPointType (Location) [66] ............................................................... 21
Table 15: Important elements in AddressType (Location) [66] .......... 22
Table 16: Important elements in ChargePointInfo [66] .......... 22
Table 17: Important elements in ParkingSpotInfo [66] .......... 22
Table 18: Important elements in ParkingStatusType [66] .......... 23
Table 19: Important elements in TariffInfo [66] .......... 23
Table 20: Important elements in IndividualTariffType [66] .......... 24
Table 21: Important elements in TariffElementType [66] .......... 24
Table 22: Important elements in PriceComponentType [66] .......... 24
Table 23: Summary of location related word* [28, 62, 63, 66, 68, 69] ...... 29
Table 24: Summary of tariff related word* [26], [62], [64], [68], [69] ...... 30
Table 25: Summary of occupancy related word* [62, 64, 66, 68, 72] ...... 30
Table 26: Date time definitions ................................................................. 40
Table 27: Result from comparing the converted data with the original data (regular tariffs from EasyPark) ........................................ 51
Table 28: Result from comparing the converted data with the original data (bucket tariffs from EasyPark) ........................................ 52
Table 29: Result from comparing calculated fees between converted rates and the German rates ................................................. 54
Table 30: Computer specifications .......................................................... 55
Table 31: Result of the conversion speed test (note using a decimal comma.) .............................................................................. 56
Table 32: Number of tariffs in relation to the number of locations ....... 56
Table 33: Result from GZIP compressions tests using 3 different settings, with percentage compared to original size (note the use of the decimal comma.) .................................................. 58
Table 34: GraphQL symbol definitions .................................................... 71
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma separated values</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
<tr>
<td>GZIP</td>
<td>Gnu zip</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>KML</td>
<td>Keyhole Markup Language</td>
</tr>
<tr>
<td>MPPO</td>
<td>Mobile Phone Parking Operator</td>
</tr>
<tr>
<td>PO</td>
<td>Paring Operator</td>
</tr>
<tr>
<td>PP</td>
<td>Parking Provider</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter describes the subject area, the specific problem that this thesis addresses, the context of the problem, and the goals of this thesis project.

1.1 Background

The concept of parking has existed for a long time, starting when vehicles were introduced. In its fundamental form, the definition of parking, according to the Cambridge Advanced Learners Dictionary, is “leaving a vehicle in a particular place for a period of time” [1]. Parking has only been a subsidiary act when owning a vehicle, as alone parking itself has no meaning. Because the popularity of owning cars rose, the need for parking has also increased. These days there is a high demand for parking in the most densely populated and urbanized areas. Apart from the increased demand for parking, safety and security issues have emerged as important parts of parking [2].

In highly populated areas the demand for parking has continued to rise, while the supply of parking spots has had difficulties keeping up. For example, in Stockholm, the biggest and most densely populated city in Sweden, had roughly 65 800 parking spots 2017, while the number of cars numbered to roughly 356 000 [3, 4]. The number of cars in the City of Stockholm 2016, counting only cars registered in the municipality, was roughly 347 000 [5]. In the period from 2015 to 2016, the number of cars in the city of Stockholm increased by 7 000 (from 340 000) [6], while there was no increase in the number of parking spots [7, 8]. In the period from 2016 to 2017, the number of cars increased by additional 9 000 (from 347 000 to 356 000). In contrast, in the period from 2016 to 2017 the number of parking spots decreased by 5 200 (from 71 000 to 65 800) [9].

Other examples where there is a rapid increase in demand are Beijing and India [10]. During 2004, Beijing had an annual growth of 340 000 private cars, which outnumbered the 100 000 parking spots available. The increasing demand for parking in India is driven by the increasing number of cars sold in India. For example, in 2013 roughly 2.6 million cars were sold in India.

With such high demand, the interest and importance of parking have grown in the eyes of many companies and organizations. This can be seen in the number of parties now involved in the parking industry. Today, the global parking industry is estimated to be worth over US$30 billion [11, 12].

1.1.1 Sustainability

It is a common knowledge that motor-driven vehicles have a negative impact on the environment. More specifically, vehicles pollute the air when the engines are running. Unfortunately, even vehicles utilizing environmentally friendly fuel still negatively affect the environment. For example, vehicles can still pollute the air by
releasing particles into the air from tearing up particles from the roads. Inefficient parking (such as long search times, caused by few available parking spots) has a negative impact on the environment by creating traffic congestion [13, 14]. Traffic congestion is bad because vehicles are releasing emissions, despite standing still. According to a study conducted by INRIX [15], congestions brings additional indirect costs to drivers. Furthermore, a vehicle needs to be driven to and from a parking space.

By making parking more effective, parking will have a lower negative impact on the environment. The average time for finding a parking spot decreases when the occupancy of parking spaces is at or below 85% [16]. In addition to environmental issues, congestion may cause safety issues [13, 17]. This occurs because congestion disrupts the flow of traffic and thus initially increases the chance for collisions.

1.1.2 Privacy

Some parking systems, such as a smartphone application, can be used to collect user data. When collecting data, especially personal data, there are important responsibilities. For example, collecting more data than what is necessary, violates personal integrity and privacy. Moreover, data collection without users’ consent are unethical and might be illegal.

Recently a new data protection rule – General Data Protection Regulation (GDPR) [18], regarding collecting and processing personal data came into effect in the 25th of May 2018 in Europe. Basically, the rule defines how and what kind of data that is acceptable to collect or process. For example, consent must have been given from concerned users’. It is not allowed to collect an excessive amount of information, and there is a limit on how long sensitive data can be stored. Therefore, both from an ethical and legal perspective, it is important to ensure that a system does not violate personal integrity or privacy of its users.

1.2 Problem statement

Lack of parking brings problems such as congestion, which further exacerbates environmental and safety issues. The traditional solution to reduce congestion has been through keeping optimal levels of occupancy. Traditionally the means used for influencing the occupancy has been with parking fees and regulations. Parking fees affect the occupancy by manipulating the demand, while regulations try to affect the availability of parking spots through parking time limits. However, the use of parking fees has been the most efficient approach to manage occupancy. Because lowering the demand also influences the availability (fewer vehicles are parking due to low demand). In contrast, regulations only force a circulation which does not affect the actual demand or availability. For example, some drivers switch to another parking spot while other circulates but ends up in the same spot.
In recent times traditional parking fees have been insufficient to manipulate the demand. Therefore, companies have started to work towards dynamic pricing [10, 19]. Currently, dynamic pricing is already in use by other industries, such as the hotel and the taxi industries. Furthermore, for an experiment, a few cities have implemented dynamic pricing for parking on a small scale [11, 19, 20].

However, the parking industry is missing some key features, to fully utilize dynamic pricing. One of these missing key features is the need for communication standards. This is needed for different stakeholders (such as parking providers, operators, and cities) to be able to share parking-related information [21, 22]. While there exist plenty of APIs and protocols for sharing parking-related information (see Sections 2.2.3 and 2.3), none has been widely adopted.

1.3 Goals

The main goal of this project is to develop and evaluate a communication protocol for sharing parking-related information. The goal is also, that this protocol will become a standard for the parking industry. However, depending if the time is sufficient, the project might focus on additional intermediary goals. The highest priority will be for the protocol to fully support tariff data. The second priority would be on developing a functional interface and tools for writing data in the protocol’s format. The least priority goal is the full support for location and occupancy data.

1.4 Methodology

Overall, the types of research methods used in this project are mainly qualitative methods. This project was divided into two major parts. The first part consists of a literature study to widen my knowledge of the subject area. During the literature study, information was collected to identify important metrics and parameters. These metrics and parameters mainly come from earlier works and opinions from experts in the parking industry. Therefore, a qualitative method suits this thesis the best. However, due to the need for a correctness and performance evaluation of the protocol, a quantitative method based on experiments will also be used.

The second part consists of the design, development, and evaluation of a communication protocol. During this second part, an engineering method based on multiple iterations of a prototype and its evaluation was used. The initial iteration developed the first prototype of the protocol. This protocol was developed based on the previously collected information. The next iteration follows testing of the prototype with respect to the metrics, found earlier. Depending on the results from these tests, the protocol was modified, to better realize the desired metrics in the next iteration of the prototype. This process continued until the protocol fully satisfied the metrics and was approved by the parking industry experts.
1.5 Future work and limitations

This project focused on the design, development, and evaluation of a communication protocol for sharing parking-related information. Additionally, some tools for aiding the integration of this protocol with other systems were developed to facilitate the adoption of the proposed protocol.

This project will not develop a complete parking system, collect parking-related information, or implement a method for calculating dynamic parking prices. Instead, the design, development, implementation, and evaluation of a complete parking system, capable of collecting parking-related information and then utilizing dynamic pricing, is seen as future work. However, it is in the intention of the project to define a generic method for how to express tariffs. A complete description and additional future work can be found in Section 6.4 (page 63).
2 Background

This chapter provides basic background information regarding parking prices, tariffs, and dynamic pricing. Additionally, this chapter describes the concept of dynamic pricing and its use cases. This chapter also describes work related to sharing parking-related information. Lastly, this chapter presents the conclusions made from the literature study.

In this chapter, there are some words that are used interchangeably. A parking price is the same as a parking fee, parking rate, fee, price, and rate. The word driver is used equally to the word parker.

2.1 Parking prices

Parking prices exist for more reasons than simply earning money. Past experience indicates that a good balance between price and demand leads to optimal parking space utilization [13, 16, 19, 23]. Optimal parking space utilization can maximize profits while maintaining the availability of parking spaces at an optimal level can prevent congestion. Moreover, simply raising the parking rates can both increase and decrease the overall revenue of a certain parking area. This occurs because increasing the rate while maintaining the number of customers will lead to increased revenue. However, if the increased price leads to fewer customers, the outcome might be a loss of revenue. Furthermore, changing the parking rates may affect the demand, by attracting or deterring customers. For these reasons, parking prices can and are used as means for managing demand.

Parking prices are usually determined based on a business model or by adopting a strategic approach [13, 23, 24]. Most commonly, prices are set based upon a combination of a strategic approach and a business model. In a business model approach, prices are set to maximize revenue, while lowering costs. In a strategic approach, prices are set according to supply and demand. The strategic approach is the most fundamental way to determine parking prices [13].

Ring structured pricing is a strategic approach to determine parking rates, as well as a way to apply these rates to larger areas, such as a city or municipality, by dividing the area into zones [16, 25]. Large areas are divided into smaller zones, based on the popularity of the zone. More central parking areas, generally offer shorter walking distances, hence they tend to be more popular than outer areas. Then based upon the popularity of a given zone, parking rates are set. Less popular zones are usually set with low rates to attract more customers (to better utilize the parking capacity of this zone). In the opposite way, more popular zones are set to higher rates, to deter drivers from attempting to park in these zones. The basic idea is to try to keep the occupancy of all parking spaces at an optimal level, which according to M. Darst is 85% [23]. The optimal level of 85% occupancy was a theory made popular by Professor D. Shoup, a Research Professor in urban planning [26, 27]. Traditional tariffs and dynamic pricing will be presented in the following subsections.
2.1.1 Traditional tariffs

Parking prices need not be constant during all hours of a day [23, 25, 28]. Instead, parking prices might change depending on the time of the day or the day of the week. The prices might even change depending on the elapsed parking time. Therefore, some areas might have a collection of parking prices, to fully describe the parking fees. The collection of parking prices is often referred to as a tariff. Some tariffs might be based on historical occupancy conditions and the prices then change accordingly. However, these types of tariffs are often still considered static, because even if the prices do change, the prices are not based on current conditions.

A single location might have multiple tariffs, which are separated by some constraints (such as time or day). These constraints regulate which drivers the tariffs are applicable for. Tariffs need not only be separated by current parking time, tariffs might also be separated depending on the time a parking started. For example drivers parking during a targeted period might get cheaper prices compared to a driver parking outside the targeted period. This way parking providers can encourage drivers to park during specific time periods. This kind of parking is also known as early bird parking [29–31]. Tariffs can also be restricted by other constraints, such as by certain groups (such as public, residential, or personnel), or types of vehicles (such as a car, motorbike, bus, truck, or electric car).

Another type of restriction is constraints of the tariff, such as a max fee. The max fee regulates how much a parker can be charged at most, for a successive period. There can be a max fee per day, week, month, span (such as 3 hours, 8 hours, or 5 days), or parking (no time limit). Furthermore, the date time units can have different meanings depending on the wording. For example, a day can refer to 24 hours, or a calendar day (which are additionally defined by a daily reset time). Continuously, a week can refer to 168 hours, 7 calendar days, or a calendar week (which resets at the end of the week). Lastly, a month have even more ways to be defined in, especially since every month does not have the same amount of days.

Tariffs can further mainly be divided into two types: regular and fixed [28]. Regular tariffs are prices that are charged depending on the elapsed or expected parking time. In contrary, fixed tariffs are prices that are charged fully for a predefined period, usually as the tariff is defined. A tariff can be defined as hourly, daily, weekly, monthly, or as another predefined period (such as 20 minutes, 3 days, or 4 weeks periods). In addition to the two tariff types, parking fees might be charged in advance, which is known as prepaid parking. Fixed tariffs that are prepaid are commonly used for predefined parking permits, such as residential or personnel parking permits.

2.1.2 Dynamic Pricing

Dynamic pricing means that parking prices change based on current conditions, in contrary to traditional parking which changes according to old conditions. Moreover, this change can occur monthly, weekly, daily, or even on an hourly basis. However,
the important factor is that the changes can be made in real-time based on current conditions. Compared to the traditional tariffs, which often are based on old conditions. In other industries, such as hotel, retailing, and taxi industries, dynamic pricing is already widely used [19]. In these industries, prices might be individual, or change based on the status of current inventory (versus demand). Fundamentally, dynamic pricing sets prices in real-time, usually based on the current supply and demand conditions.

In the taxi industry, more specifically Uber, prices are based on a business model using dynamic pricing. Taxi price changes based on the availability of drivers in the surrounding area. During high demand, the prices favor drivers, to attract drivers to match the demand. The most famous implementation of dynamic pricing is the pricing algorithm used by Airbnb [32]. The algorithm calculates prices for renting out rooms based on other prices in the location, demand compared to supply, and what a room includes.

Dynamic pricing can be used to ensure that parking occupancy is kept at an optimal level [14, 23]. Today, some companies have implemented dynamic parking into their system [11, 19, 20]. Dynamic pricing is considered to be a part of smart parking [14]. Moreover, dynamic pricing does not simply mean a change of the pricing strategy, but rather that prices will be based on demand and supply. The differences between truly dynamic pricing and a varying pricing strategy are how often and long it takes to change parking rates. Because changing parking rates today takes too many steps, with each step taking a long time [33], parking rates are relatively static or at least change very slowly. The reason for the slow rate of changes is the number of steps. The steps required involve meetings and conversations, to understand the parking situations, propose new rates, get feedback from the community about these new rates, make final decisions based on this feedback, and then finally announcing the new rates to the public. In contrast, dynamic pricing would automate many of the steps, while skipping some of these steps. Therefore, being able to keep up with fluctuating demands.

Companies that want to utilize dynamic pricing, must have a system capable of collecting the necessary information and this information must be accurate [14, 19]. Moreover, these systems must be robust against failures, as system downtime leads to loss of revenue. Lastly, the system must have sufficient computing power and efficient communication to support all the desired features of dynamic pricing.

However, there are some disadvantages using dynamic pricing. For example, prices that fluctuate excessively might confuse and frustrate drivers [14]. Customers might not get the parking fee, they first were promised. Which might have been caused by increased occupancy by the time the parking is made. Consequently, dynamic pricing might instead deter drivers, reducing the demand too much. Therefore, negatively affect customers visiting surrounding businesses [20]. There is also a risk that dynamic pricing might be misused for personal gain. The misuses can be made by means, such as: announcing higher, or false demand to systems. Thus, creating higher parking prices than a current condition might require. Some
companies might even announce false demand to the other systems, which might increase the overall parking prices.

2.2 Technologies

Currently, in the parking industry, there exist many different technologies for collecting parking information. Furthermore, there exist several application programming interfaces (API) which provide information regarding parking occupancy for specific areas.

2.2.1 Devices in the parking industry

There exist both new and old devices, where some of the old devices have been improved over time. In the most cases, these devices collect occupancy data or can provide services to retrieve such information.

2.2.1.1 Parking meter

The first working parking meter was installed in the year 1935 [14, 34]. The main purpose of a parking meter is to enable unattended acceptance of payments for parking spaces [35]. In earlier days, a parking meter was an analog device that had a clock mechanism driving a timer [36]. These traditional parking meters were individual for each parking spot, also known as single-space meters, as the prevailing concept was to pay at each parking space. Additionally, these traditional parking meters could only accept coins and required payment in advance. If more parking time was paid for than required, the customer would have paid more than necessary (hence the customer lost money).

Today, most parking meters have been digitalized and they can accept multiple kinds of payments. Instead of only single-space meters, multi-space meters are widely used to print permits, following the pay-and-display concept [35]. Some parking meters are capable of keeping track of active parking permits and therefore are able to provide some occupancy data [37].

2.2.1.2 Sensor for detecting occupancy

One of the newer additions to the parking industry is occupancy sensors [14]. These occupancy sensors track the parking occupancy of parking areas [37]. There exist different types of sensors and different methods to get a precise approximate view of current parking occupancy. Currently, it is possible to install sensors in car parks, and it has proven to be difficult in integrating such sensors into on-street parking in cities. In cities, there might be interference from under- and overground cables. Additionally, the data might be influenced by extreme weathers, such as cold and snow.

One approach for achieving a 100% accuracy of parking occupancy, is to install a sensor at each parking spot [37]. This is expensive, but the most accurate approach
to collect parking occupancy data. Cheaper methods consist of reducing the number of sensors and pairing the sensor data with sampling methods. Using spatial sampling coupled with a fraction of sensors gives an approximate view of parking occupancy. Lastly, using temporal sampling, it is possible to use a small number of mobile camera sensors and then use a computer vision algorithm to determine the availability of parking spaces. Both sampling methods can also make use of the information from parking meters, thus making the approximation more accurate.

### 2.2.1.3 Smartphone applications

These days, almost everyone has a smartphone which they take with them everywhere, and there is an application for almost everything [38]. Furthermore, a smartphone has become a tool to assist people with most of their activities. Smartphones provide useful features, such as various sensors and GPS [39]. Therefore, many industries have integrated smartphones into their environment to enhance and give their customers a richer experience. Currently, some features of a smartphone are already employed in the parking industry. The most common is the use of mobile payment. Mobile payments often offer more than one type of payment [40]. One result is that customers can pay for their parking time via their smartphones. Another result is that drivers can pay for only the actual parking time after the parking, in contrary to the expected before.

Other features that a smartphone can bring to the parking industry are real-time occupancy tracking, locator services, and acting as an information source [39–41]. Tracking currently valid parking permits can give an approximate status of current occupancy. For example, knowing that a vehicle with a currently active parking permit is moving, hence no longer parked, can indicate that the parking place that it had been using is now available. However, for this to work, the companies need access to location data of the user. Smartphones can also be used as a locator. For example, a smartphone can be used to locate available parking spots, a parked car, or an electric charging station. Parking information, such as opening hours, tariffs, and occupancy, can be announced through an application or webpage, thus making the information accessible via a smartphone.

However, apart from operators getting access to occupancy and location data, the operators might also get access to more sensitive data, such as private data. Private data may include information such as age, gender, phone number, information regarding family and friends, and users’ whereabouts. Collecting sensitive data from users therefore naturally brings issues with personal integrity and privacy. Especially, if the information is collected without the knowledge or permission of the user. Which is illegal since GDPR came into effect (described in Section 1.1.2).

### 2.2.2 Communication technologies

There are a couple of data formats commonly used for communication. Some might complement each other in different aspects, while others are used to achieve the
same purpose. The data formats used in this project are JavaScript Object Notation (JSON), Keyhole Markup Language (KML), Comma-separated values (CSV) file, and Gnu zip (GZIP). The two data formats that have been examined are eXtensible Markup Language (XML) and Web Services Description Language (WSDL). The two platforms GraphQL and Nodejs will be used for the development of the protocol. And lastly the protocol Transport Layer Security (TLS) will be used to ensure reliability and security. Each of these formats, platforms and protocol will be described in the paragraphs that follow.

2.2.2.1 XML

XML is a markup language mainly used for storing and transporting data [42]. The syntax of XML is similar to that of HTML. However, XML has the additional feature of being self-descriptive (although an external DTD may be necessary). XML is a straightforward language with no predefined tags, unlike HTML. It is up to the developer to decide on both the tags and document structure, making XML a highly customizable language.

The core structure of XML is based upon using tags to open and close element areas [43]. Where each XML element is defined with opening and closing tags. XML elements allow internally nested XML elements. Therefore, there can be parent and child relations between XML elements. Every XML document requires at least one global parent element, known as the root element, which binds all elements in the document. Additionally, XML supports the use of attributes, which allows more descriptive information to be encoded regarding the stored element variables [44]. Furthermore, a DTD can be referred to using a DOCTYPE definition tag. An example of an XML document is shown in Figure 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rates SYSTEM "rates.dtd">
<rates>
  <rate order="1">
    <value>20</value>
    <interval>60</interval>
    <intervals>1</intervals>
  </rate>
  <rate order="2">
    <value>15</value>
    <interval>60</interval>
    <intervals>1</intervals>
  </rate>
  <rate order="3">
    <value>10</value>
    <interval>60</interval>
    <intervals>1</intervals>
  </rate>
</rates>
```

**Figure 1:** Example of a possible XML document
The strongest points of XML are that it is easily readable by both computers and humans, and it is easily customizable so that the developers can achieve what they want.

2.2.2.2 DTD

DTD, as its name suggests, is a definition document. DTD is used to define the structure, legal elements, and attributes of an XML document. Basically, DTD is used to impose a standard for interchanging data and to validate data in XML documents. From the viewpoint of DTD, XML documents consist of 5 types of blocks: elements, attributes, entities, PCDATA, and CDATA. PCDATA defines parsed character data and the element value is found between the start tag and end tag of an XML element. CDATA is character data which does not need to be parsed and such data is usually the value of XML attributes. Elements, attributes, and entities are defined in DTD using the tags !ELEMENT, !ATTLIST, and !ENTITY. Additionally, element names can be accompanied with the symbols: plus (+), asterisk (*), question mark (?), and pipe bar (|). Asterisk is used to define that an element can occur zero, once, or many times. Plus is used to define that an element can occur at least once (once or many). Question marks are used to define that an element can occur at most once (once or not at all). Pipe bar is used to define that only one element can be chosen between two provided elements.

A DTD must be declared inside an XML document and must be wrapped inside the DOCTYPE definition. The DTD can also be defined using an external file which then must be referred to by the DOCTYPE definition in an XML document. An example of an XML document referring to an external DTD was shown in Figure 1, while Figure 2 shows the contents of this DTD file.

```
<!ELEMENT rates (rate*)>
<!ELEMENT rate (value, interval, intervals)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT interval (#PCDATA)>
<!ELEMENT intervals (#PCDATA)>
<!ATTLIST rate order CDATA #REQUIRED>
```

**Figure 2: Example of an external DTD**

2.2.2.3 WSDL

WSDL is a format and language used for describing functionalities of web services [45]. WSDL is written in XML; however, WSDL provides some major elements with their own pre-defined tags.

These major elements, their tags, and what they define are: *types* - the element which defines which data types are to be used by the web service, *message* - the element that defines the data used for each operation, *portType* - the element that describes the available operations and their coupled message, and lastly *binding* - the element which defines which protocol and data format each port type uses.
2.2.2.4 KML

KML is a file format used for displaying and sharing geographical data [46]. KML is fully compatible with Earth browsers such as Google Earth. KML is based on the XML standard, therefore KML supports the use of tags, nested elements, and attributes. However, there exist pre-defined tags in KML, which are coupled with some of its key features, such as placemark, paths, and polygons.

A placemark pinpoints a position on the earth using coordinates. By using the paths feature it is possible to mark up a path (route or series of waypoints) which can be displayed using an Earth browser. It is possible to mark simple or advanced geometric areas using polygons.

2.2.2.5 JSON

JSON is a format for storing and exchanging data [47]. JSON is similar to XML in that it uses plain text, but the data is written according to how objects are defined in JavaScript. JSON is a lightweight data-interchange format which is both self-describing and easy to understand.

The syntax of JSON follows the JavaScript Object Notation, where data are stored as name-value pairs [48]. Each name-value pair can be further subdivided by commas. Curly brackets define where JSON object starts and ends. JSON allows nested objects, which means that a single object may contain other objects. JSON supports the use of arrays, making it possible to nest multiple objects or values in a single object. JSON arrays are defined using square brackets. JSON does not require tags and the encoding of data is shorter compared to using XML. An example of a JSON object is shown in Figure 3.

```
{"rates": [
    {"order":1,"value":20,"interval":60,"intervals":1},
    {"order":2,"value":15,"interval":60,"intervals":1},
    {"order":3,"value":10,"interval":60,"intervals":1}]
```

**Figure 3: Example of a possible JSON object**

The strongest points of JSON are that it is lightweight and easy to understand, with the addition that it is easy to use with any programming language.

2.2.2.6 CSV

A CSV is a common format for storing or exchanging tabular data [49] using a text file. The fundamental idea is to separate the values of the table using a separator character, most commonly a comma as the name suggests. However, as there are no standard regulations on which separator to use, many implementations with other characters exist. Anyhow, to further distinguish the values from the separators, quotations may be used to emphasize the values. This also allows the use of separators in the values.
2.2.2.7 **GraphQL**

GraphQL is a platform for building APIs. GraphQL uses a query language that makes it possible to retrieve data from a system using queries [50]. GraphQL also provides its own schema language, to define data fields of stored data objects. Additionally, GraphQL is independent and works with most languages and systems. A GraphQL service is strongly recommended to respond with data in JSON format, but this is not required [51]. However, the chosen format used for responses must be able to support the major primitives that GraphQL provides, these are *Map, List, String,* and *Null.*

The syntax of the GraphQL schema language has its own basic components and format to define data fields [52]. An object is defined using the tag *type* followed by an object name. The object content (data fields) are encapsulated using curly brackets. A data field is defined by a name-value pair. An exclamation (!) is used to define that the value of a field is non-nullable, meaning that a value is guaranteed when requested. In addition to the major primitives, the schema language also supports the primitives: *Boolean, Int, Float,* and *Enum Value.* GraphQL also supports the use of arguments for data fields. The arguments of a data field are encapsulated after the name using round brackets. Furthermore, arguments can be required or optional, if optional a default value can be defined. Entry points are defined using the *query* and *mutation* types, which are defined in the same way as regular objects. An example of an object defined using GraphQL schema language is shown in Figure 4.

```
type Tariff{
  zone: Int!
  value (currency: Currencies): Float!
  period: Int!
}
```

**Figure 4: Example of a GraphQL schema**

A simple GraphQL query is encapsulated by curly brackets. The desired data fields are defined using their names, as they are defined in the schema. Arguments can be included in the query to specify what stored data to retrieve. An example of a query for requesting tariffs with values expressed in SEK is shown in Figure 5.

```
{ tariff{
    zone
    value (currency: SEK)
    period
  }
}
```

**Figure 5: Example of a GraphQL query**
A strong point of GraphQL is that the interface can be developed using the most programming languages and environments, more importantly, it supports JavaScript. GraphQL also provides its own languages and runtime for building an API. Additionally, it is possible to use GZIP (see paragraph 2.2.2.9), which is also recommended, to decrease the package sizes sent by a GraphQL interface.

### 2.2.2.8 Nodejs

Nodejs is a platform running on the JavaScript runtime, commonly used to build server applications [53]. In contrast to traditional usage of JavaScript (client-side scripting), Nodejs runs JavaScript on the server-side.

Nodejs is asynchronous event-driven and is suitable for developing scalable network applications. Nodejs also uses a non-blocking I/O model and is considered to be both lightweight and efficient. Nodejs also provides its own ecosystem, npm, which provides access to many open-source libraries, such as libraries for building GraphQL interfaces.

### 2.2.2.9 GZIP

GZIP is a file format and an application used to compress and decompress other files [54]. GZIP is a standard application in the Unix environment systems. GZIP uses the deflate algorithm, for file compression, which is a combination of the algorithms LZ77 and Huffman coding. GZIP uses the file extension .gz and tar.gz for old versions.

### 2.2.2.10 TLS

TLS is a protocol that provides communication security on the application layer [55]. TLS was developed as the successor of Secure Socket Layer (SSL). The main purpose of TLS is to provide security and data integrity between the communication for two applications. Basically, a connection is made secure by using symmetric cryptographic and handshakes to establish a shared secret. Public key cryptography is used to authenticate the parties over the connection. Additionally, a connection is made reliable by using message integrity checks throughout the communication.

### 2.2.3 Parking data APIs

There are several application programming interfaces (APIs) available online that shares parking information. Each of these provides different kinds of parking information. Some examples are ParkRight API, SmartPark API, Streetline API, INRIX API, and ParkAPI.

The ParkRight API [56] is based on the SmartPark API [57], therefore the two APIs have many similarities. The difference is that ParkRight provides information for a smaller area, specifically central London, while the SmartPark API is not limited to any specific area. Apart from this difference, both APIs provide the same types of
information: parking lot details, operating hours, tariffs, occupancies, and unavailable bays. The information is retrieved by using HTTP requests. The requested information can be for a specific area by defining a zone, subzone, parking lot, or geospatial coordinates. Additionally, the API can provide the information in either JSON or XML format.

The Streetline API collects parking occupancy information using sensors and cameras [58]. The accuracy of the occupancy information is further improved by using mobile devices, payments, cameras, and GPS. The Streetline API consists of two separate APIs: Basic API, and Availability API [59]. The Basic API provides static parking information regarding facility location, rates, hours, and other policy details. The Availability API provides real-time information regarding on-street and off-street occupancy. Unfortunately, there is no documentation available for the Streetline API.

INRIX Connected Services consists of multiple applications and services [60]. Some of these services focus on providing parking information, such as off-street parking, on-street parking, and occupancy [61]. Information about on-street and off-street parking can be requested for specific parking lots, blocks, or a geographic area. Information about occupancy can be provided as either a number of available parking spaces, the percentage of available spaces, or simply as an indicator of whether the parking area is full or not. The INRIX API utilizes HTTP requests and provides information in JSON format.

ParkAPI collects parking data by reading occupancy data published by cities [62]. The main idea is to retrieve and gather data from multiple sources and make it accessible at one single site, thus reducing the load on the original sources. ParkAPI provides occupancy data for every parking lot in a city when requesting data. Therefore, it is up to the user to filter and parse the information for a specific parking lot. A city is specified by providing a city identification with in the HTTP request. The ParkAPI provides occupancy data in JSON format. ParkAPI also supports hosting ones’ own server, by providing instructions and downloadable source code.

2.3 Related work

There exist a few organizations who mainly work with parking. A few of these have developed protocols closely related to the protocol this thesis intends to develop. In this section, we will explore this related work.

2.3.1 The proposed standard for parking information sharing

The International Parking Institute (IPI) has the intention to develop a protocol, as an open source project, for exchanging data regarding parking information [21]. The project is called IPI Data Exchange Standard, IPI-DataEx for short. IPI has a vision that data sharing is essential for enabling several concepts involving vehicles and parking. Some of these concepts are car sharing, ride sharing, prepaid parking, and
dynamic pricing. The intention of their standard is to provide a common language and data definition, to make it easier for North American companies to exchange data.

The vision of the IPI-DataEx is similar to the goal of this thesis, specifically the desire to develop a standard protocol for sharing parking-related information. Moreover, as IPI-DataEx has progressed it is an interesting project to investigate further and take inspiration from.

Because the protocol is still under development, all its intended features do not exist yet. The information that is regarded to be important, is categorized into different sets and further into subsidiary domains. The sets of information that are currently included in the protocol proposal are facility location, parking rate, and occupancy data [21, 22]. There is a standard document for each information set where the data fields are described. The data fields include a data field (name), domain affiliation, field description, data type if the field is required or not, suggested list items, and notes. Facility location related information are proposed to be shared via comma-separated values (CSV) files, as facility data is seemingly static. Dynamic data such as rate and occupancy are proposed to be shared via JSON/XML schemas. Unfortunately, there are currently no API schemas available for dynamic data.

Facility location is currently the only information set with a complete standard definition [63]. Parking rate and occupancy data are still under development [64]. The information that is included in each set will be described in the paragraphs that follow.

2.3.1.1 Facility location

The facility location related information describes a facility, such as a parking lot, garage, or street area. This facility data includes information regarding the location and facility-specific data (such as address, parking spaces, and contact information) [63]. The domains related to facility location are an area, contact, location, layout, operating hours, payment, record, and space. The domains with required data fields are an area, contact, location, layout, and record.

An area domain describes the surrounding area of the facility, for example: if there are any restrictions and the geospatial coordinates of the facility. The contact domain includes information necessary for contacting the facility owner or manager. The location contains information regarding a facility’s name, address, relations to other facilities, and time zone. The layout domain information describes the facility’s physical form, appearance, type, and attributes (such as if there is a valet, overnight parking, gated, or robotic parking). For detailed information regarding facility location, related data see Tables 1 to 4.
Table 1: Important elements in area domain (Facility location) [63]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Detail</td>
<td>Text</td>
</tr>
<tr>
<td>Area Type</td>
<td>Text</td>
</tr>
<tr>
<td>Area – Geo location</td>
<td>Number string</td>
</tr>
</tbody>
</table>

Table 2: Important elements in contact domain (Facility location) [63]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point of contact</td>
<td>-</td>
</tr>
<tr>
<td>Contact Type</td>
<td>Text</td>
</tr>
<tr>
<td>Name</td>
<td>Text</td>
</tr>
<tr>
<td>Organization</td>
<td>Text</td>
</tr>
<tr>
<td>Email</td>
<td>Text</td>
</tr>
<tr>
<td>Phone Number</td>
<td>Text</td>
</tr>
<tr>
<td>Primary Contact</td>
<td>Text</td>
</tr>
</tbody>
</table>

Table 3: Important elements in location domain (Facility location) [63]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking Location Name</td>
<td>Text</td>
</tr>
<tr>
<td>Location address Country</td>
<td>List (ISO 3166-1 alpha-2 country code)</td>
</tr>
<tr>
<td>Location address Locality</td>
<td>Text</td>
</tr>
<tr>
<td>Location address Region</td>
<td>Text</td>
</tr>
<tr>
<td>Location address Postal Code</td>
<td>Text</td>
</tr>
<tr>
<td>Time zone</td>
<td>Text</td>
</tr>
</tbody>
</table>

Table 4: Important elements in layout domain (Facility location) [63]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facility Structure Type</td>
<td>List (On Street, Off Street Structure, Off Street Surface)</td>
</tr>
<tr>
<td>Open to public</td>
<td>List (Public, Not public)</td>
</tr>
</tbody>
</table>
2.3.1.2 Occupancy data

Occupancy data include location data and occupancy data enabling an analysis of demand and supply [64]. The domains related to occupancy data are an area, demand, demand space, location, record, space, and supply. All domains, except for area, have required data fields.

Demand consists of occupancy data and information regarding when the data was collected. Demand includes specific information regarding each parking spot such as if the parking spot is occupied or when the parking spot was (last) occupied. Location-related information defines which facility the occupancy data relates to. Record information describes when the data was created or updated. Space data consists of physical information about every individual parking spot, such as dimensions, size, and how occupancy data are collected. Supply is static data describing how many parking spots a facility has, it also states the accuracy of the occupancy data. For detailed information regarding occupancy-related data see Tables 5 to 8.

Table 5: Important elements in demand domain (Occupancy) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand Table</td>
<td>-</td>
</tr>
<tr>
<td>Data as of</td>
<td>Date/Time</td>
</tr>
<tr>
<td>Counted or Derived or Verified</td>
<td>List (Counter, Derived, Verified)</td>
</tr>
<tr>
<td>Collection Frequency</td>
<td>Number</td>
</tr>
</tbody>
</table>

Table 6: Important elements in demand space domain (Occupancy) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand Space Table</td>
<td>-</td>
</tr>
<tr>
<td>Space ID</td>
<td>Text</td>
</tr>
<tr>
<td>Occupancy Start Time</td>
<td>Date/Time</td>
</tr>
<tr>
<td>Occupancy End Time</td>
<td>Date/Time</td>
</tr>
<tr>
<td>Detection Update Time</td>
<td>Date/Time</td>
</tr>
</tbody>
</table>

Table 7: Important elements in location domain (Occupancy) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking Location Name</td>
<td>Text</td>
</tr>
</tbody>
</table>
Table 8: Important elements in supply domain (Occupancy) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking Supply</td>
<td>-</td>
</tr>
<tr>
<td>Space or Vehicle View</td>
<td>List (Space, Vehicle)</td>
</tr>
<tr>
<td>Space Count</td>
<td>Number</td>
</tr>
<tr>
<td>Supply Expiration Valid Start</td>
<td>DateTime</td>
</tr>
<tr>
<td>Supply Expiration Valid End</td>
<td>DateTime</td>
</tr>
</tbody>
</table>

2.3.1.3 Parking rate

Parking rate includes information regarding the location and specific information (such as description, availability, and expiration) about the rates [64]. The domains related to parking rate are ancillary, area, location, rate table, record, rate construction(RC), and surcharge. All domains in parking rate have at least one required field.

Location-related information defines which facility the parking rate data belongs to. Rate tables are basically tariffs which include an identification string, type of rate, restrictions on the rate (public or private), and when the rates are active. Record information describes when the data was created or updated. RC contains more specific information regarding a rate such as value, period, unit time, and if it is repeatable or not. Surcharge describes additional fees associated with the parking rate, such as if taxes are included in the rate. Ancillary information is any subsidiary information not associated with any other domain. Ancillary information includes additional rules or filters for the parking rates. For example, different types of vehicles might have different rates. For detailed information regarding required data fields and their types related to parking rate, see Tables 9 to 12.

Table 9: Important elements in the area domain (Parking rate) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area name</td>
<td>Text</td>
</tr>
</tbody>
</table>

Table 10: Important elements in the ancillary domain (Parking rate) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate Ancillary</td>
<td>-</td>
</tr>
<tr>
<td>Validation Accepted</td>
<td>List (Yes, No)</td>
</tr>
</tbody>
</table>
Table 11: Important elements in the rate table domain (Parking rate) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate ID</td>
<td>Text</td>
</tr>
<tr>
<td>Rate Type</td>
<td>List (Contract, Hourly, Event, Daily)</td>
</tr>
<tr>
<td>Rate Availability</td>
<td>List (Private, Restricted, Public)</td>
</tr>
<tr>
<td>Rate Active Times</td>
<td>Text</td>
</tr>
<tr>
<td>Rate Expiration Valid Start</td>
<td>DateTime</td>
</tr>
<tr>
<td>Rate Expiration Valid End</td>
<td>DateTime</td>
</tr>
</tbody>
</table>

Table 12: Important elements in rate construction (Parking rate) [64]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1value</td>
<td>Number</td>
</tr>
<tr>
<td>B1period</td>
<td>Number</td>
</tr>
<tr>
<td>B1unitime</td>
<td>List (Sec, Min, Hr, Day, Month)</td>
</tr>
<tr>
<td>Bxxvalue</td>
<td>Number</td>
</tr>
<tr>
<td>Bxxperiod</td>
<td>Number</td>
</tr>
<tr>
<td>Bxxunitime</td>
<td>List (Sec, Min, Hr, Day, Month)</td>
</tr>
</tbody>
</table>

2.3.2 Added support for parking information sharing

The Open Clearing House Protocol (OCHP) is an open source project focused on electric vehicles, specifically information related to charging of electric vehicles [65, 66]. The purpose of the OCHP project is to enable communication and cooperation between different charging stations. During the latest update, the protocol added support to share parking information [67]. The protocol is currently able to share information regarding facility location, parking space occupation, and tariffs. Therefore, the OCHP project is also interesting and worth taking inspiration from.

OCHP provides definitions and APIs for sharing parking-related information [66]. The different APIs functionalities are divided into different roles and they are navigation service provider (NSP), electric vehicle user (EV User), electric vehicle service provider (EVSP), electric vehicle service equipment operator (EVSE Operator), and parking spot operator (PSO). The NSP role is used as an overall master which encompasses the different services available to users. An EV User acts as the client of services. The EVSP role is used to authenticate users and handle
billing processes. EVSE is used to execute requested services. The PSO collects and provides occupancy data.

OCHP provides tariff data, which essentially is the same as parking rate information. OCHP does not provide a service specifically for (facility) location related information. Instead, location related information is shared as subsidiary information under charge point related information. All the data are shared in XML format. OCHP also provides publicly available API schemas in WSDL format. The shared information is divided into different type classes and enumerated types.

The relevant location related information is split into different classes, specifically: GeoPointType, AdditionalGeoPointType, AddressType, ParkingSpotInfo, and ChargePointInfo classes. The GeoPointType consist of geospatial coordinates (expressed as latitude and longitude). AdditionalGeoPointType consist of the same information as the GeoPointType, but with the addition of a name and a category type for the point (such as entry or exit). AddressType consist of location information, such as country, city, and street name. ParkingSpotInfo is information about specific parking spots. ChargePointInfo includes information for specific charge points such as location, address, coordinates, and name. For more detailed information see Tables 13 to 17.

Table 13: Important elements in GeoPointType (Location) [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lat</td>
<td>String (10)</td>
</tr>
<tr>
<td>Lon</td>
<td>String (11)</td>
</tr>
</tbody>
</table>

Table 14: Important elements in AdditionalGeoPointType (Location) [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lat</td>
<td>String (10)</td>
</tr>
<tr>
<td>Lon</td>
<td>String (11)</td>
</tr>
<tr>
<td>Name</td>
<td>String (255)</td>
</tr>
<tr>
<td>Type</td>
<td>GeoClass*</td>
</tr>
</tbody>
</table>

* entrance, exit, access, ui, other
Table 15: Important elements in AddressType (Location) [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>houseNumber</td>
<td>String (6)</td>
</tr>
<tr>
<td>Address</td>
<td>String (45)</td>
</tr>
<tr>
<td>City</td>
<td>String (45)</td>
</tr>
<tr>
<td>zipCode</td>
<td>String (45)</td>
</tr>
<tr>
<td>Country</td>
<td>String (3, ISO 3166 country code)</td>
</tr>
</tbody>
</table>

Table 16: Important elements in ChargePointInfo [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>evseld</td>
<td>Evseld*</td>
</tr>
<tr>
<td>locationId</td>
<td>String (15)</td>
</tr>
<tr>
<td>Timestamp</td>
<td>DateTimeType</td>
</tr>
<tr>
<td>locationName</td>
<td>String (100)</td>
</tr>
<tr>
<td>locationNameLang</td>
<td>String (3, ISO-639-3 language code)</td>
</tr>
<tr>
<td>chargePointAddress</td>
<td>AddressType</td>
</tr>
<tr>
<td>ChargePointLocation</td>
<td>GeoPointType</td>
</tr>
<tr>
<td>telephoneNumber</td>
<td>String (20)</td>
</tr>
<tr>
<td>Location</td>
<td>GeneralLocationType**</td>
</tr>
<tr>
<td>parkingSpot</td>
<td>ParkingSpotInfo</td>
</tr>
</tbody>
</table>

* a globally unique identifier
** on-street, parking-garage, underground-garage, parking-lot, private, other, unknown
*** Payment and/or authorization options

Table 17: Important elements in ParkingSpotInfo [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>parkingId</td>
<td>ParkingId*</td>
</tr>
<tr>
<td>Floor level</td>
<td>String (4)</td>
</tr>
<tr>
<td>parkingSpotNumber</td>
<td>String (5)</td>
</tr>
</tbody>
</table>

* a unique identifier for parking spots
Table 18: Important elements in ParkingStatusType [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>parkingId</td>
<td>ParkingId*</td>
</tr>
<tr>
<td>Status~</td>
<td>MajorType**</td>
</tr>
</tbody>
</table>

* a unique identifier for a parking spot  
** available, not-available, unknown

Occupancy data is included in the ParkingSpotInfo and ParkingStatusType classes, which further are made available through ChargePointInfo data. ParkingStatusType contains information to bind a status (if occupied or not) to a specific parking spot, see Table 18. Tariff data is divided into the main class with multiple sub-classes, which are encapsulated in each other. The main class which encapsulates all tariff information is TariffInfo. The IndividualTariffType class is directly under TariffInfo and contains information regarding currency, payment recipient, and encapsulates the TariffElementType class. TariffElementType contains the two classes PriceComponent and TariffRestrictionType. PriceComponent contains more specific information regarding tariffs, such as billing item, price, and step size. TariffRestrictionType describes restrictions of a tariff and is not mandatory for tariff data. Lastly, ChargePointInfo contains information for payment options, which is relevant for tariff data. See Tables 19 to 22 for the fields of mandatory classes.

Table 19: Important elements in TariffInfo [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>tariffId</td>
<td>TariffId*</td>
</tr>
<tr>
<td>individualTariff</td>
<td>IndividualTariffType</td>
</tr>
</tbody>
</table>

* a unique identifier for tariffs

Table 20: Important elements in IndividualTariffType [66]

<table>
<thead>
<tr>
<th>Data field</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>currency</td>
<td>String (3)</td>
</tr>
<tr>
<td>tariffElement</td>
<td>TariffElementType</td>
</tr>
<tr>
<td>recipient</td>
<td>String (5)</td>
</tr>
</tbody>
</table>
2.4 Conclusions of the literature study

Parking prices are useful for more than earning profits, as they are also meant to manage demand. Finding an optimal balance between price and utilization enables profits to be maximized while preventing congestion. There exist multiple ways to set prices, but the most fundamental approach is based on demand and supply. Parking prices can change dynamically with the different hours in a day or between the different days in a week. Currently, traditional parking prices are still largely considered to be static and are based on demand conditions. Therefore, there is a risk that traditional parking prices might have a negative effect on congestion. For example, the parking prices might change accordingly outdated demand conditions, which no longer are relevant. This would especially be an issue if the outdated demands are entirely opposite of the current demand. The result would be increasing the prices during low demand while decreasing the prices during high demand.

Dynamic pricing is an approach where prices change dynamically and are set based on the current demand and supply. Dynamic pricing is expected to result in prices adapting more optimally to current demands. For dynamic pricing to work properly, there must be an efficient way for parking systems to share information related to occupancy and tariffs. Additionally, to make dynamic pricing possible, the systems must be capable of handling dynamic prices and occupancy data. Luckily, there exist many services able to provide parking occupancy information.

During the literature study some data fields, metrics, use cases, and roles have been identified, these were used during the development phase of this project. The identified information will be described in the following paragraphs. Additionally, the terminology used by the different APIs and project have been analyzed and are presented in Section 2.4.5.
2.4.1 Data fields

There exist two major projects which provide inspiration, for a new protocol. The first project IPI-DataEx offers a proposed standard protocol, for sharing parking information. The second project OCHP offers an open source protocol that supports sharing of parking information. A similarity between both projects is that parking data can be divided into at least 3 major categories: location, tariff (or parking rate), and occupancy data. Moreover, the information available for each category might be different.

With regard to location data, both protocols provide basic location information: an address and geospatial coordinates of the facility. IPI-DataEx also provides information about the structure and layout of the facility. In contrast, OCHP provides information regarding electric vehicle charging stations, as this is the main purpose of OCHP. Both protocols provide contact information. For a summary of the location data from both protocols see Figure 6.

![Class diagram of location data](image)

**Figure 6: Class diagram of location data**

When looking at occupancy data, both protocols support the idea of data for individual parking spots. However, IPI-DataEx provides additional information regarding parking spot supply, such as the number of parking spots, and data for occupancy prediction. In the case of tariff data, both protocols basically provide the
same information. An obvious difference is how the data is divided and structured in these formats. OCHP divides data into classes and shares them in XML format. While IPI-DataEx divides the data into domains (which do not mean that the data must be separated into different objects) and shares location related information as comma separated value files (CSV files). Additionally, occupancy and tariff data are proposed to be shared in either JSON or XML format. For a summary of both protocols see Figure 7 for occupancy data and Figure 8 for tariff data. These summaries also served as the foundation for my own protocol development.

![Class diagram of occupancy data](image)

**Figure 7: Class diagram of occupancy data**
2.4.2 Metrics

A few metrics were identified which can be used to evaluate a protocol intended for dynamic pricing. To dynamically set prices, a system requires sufficient comprehensive information to base the prices on. Therefore, the protocol must provide comprehensive information. Because the protocol is intended to be used in a real-time system, it is also important that the protocol is fast. Fast in terms of transferring, producing, and consuming the protocol data. For a protocol to be both comprehensive and fast, the protocol must only include the most relevant information. However, what is relevant can change between different use cases. Therefore, the protocol must also be evaluated with consideration of these different use cases.

Some subsidiary metrics that also should be taken into consideration, during the development, are reliability and security. Organizations must ensure that the received data, is reliable and not tampered with. Furthermore, if sensitive data are being handled, it is important that such information is secure. The protocol itself is not required to satisfy these subsidiary metrics, because they can be solved using other means, such as using the transport layer security protocol.
The metrics identified for protocol evaluation are:

**Comprehensive** Does the protocol include enough information to express the data from existing systems?

**Relevance** Does the format include any irrelevant elements?

**Speed** Can conversions of data to and from the specified format be done fast enough?

The subsidiary metrics with less importance are:

**Reliability** Is the content of the data reliable and not tampered with?

**Security** Is the sensitive data safe from being stolen?

### 2.4.3 Roles

Prior to the start of the development of a new protocol some roles and use cases were identified. These use cases are then used as goals of the protocol, to make the development easier.

Some possible roles that have been identified and will be referred to subsequently in the report are:

- **Parking Provider (PP)** The owner role set prices and provides with parking locations, such as facilities or surface lots.
- **Parking Operator (PO)** The operating role that helps a PP with different tasks. The PO might enforce payments and rules.
- **Mobile Phone Parking Operator (MPPO)** The role that helps a PO to provide additional services such as mobile payment.
- **Driver** The customer role, which utilizes services from PO and MPPO.
- **Operator** Refers all operators such as PO, and MPPO.

### 2.4.4 Use cases

The identified use cases with regards to the above roles are:

- The PP should be able to retrieve occupancy data to be able to base parking tariffs on actual occupancy.
- The PP should be able to provide the PO with tariff data.
- The PO should be able to provide occupancy data to the PP.
• The PO should be able to retrieve tariff data from the PP.
• The PO should be able to provide the MPPO with tariff data.
• The MPPO should be able to retrieve tariff data from a PO, PP, or other MPPO. This data is required to be able to charge clients the correct fees.
• Drivers should be able to retrieve location data from the parking systems, to find parking locations.
• Drivers should be able to retrieve occupancy data from the parking systems to find parking locations with available parking spaces.
• Drivers should be able to retrieve tariff data from the parking systems to be aware of the parking fees before starting to park.

2.4.5 Terminology

There are a variety of words which are used differently in the parking industry for the same purpose. For example, EasyPark uses the word parking area to refer to a parking location, while INRIX and Park API use the word parking lot, and the rest (IPI-DataEx, OCHP, and Smart Parking) uses the word location. A summary of the terminology used in the parking industry is presented in Tables 23 to 25. The summary of location-related words (see Table 23) shows that a majority uses the word street to refer to the name of a street. Similarly, the city is used by a majority to refer to the name of a city. However, the other words (locality and region) that are also used, refers to more than the name of a city. Locality and region might additionally refer to other area names (such as county name, or the name of the state).

Table 23: Summary of location related word* [28, 62, 63, 66, 68, 69]

<table>
<thead>
<tr>
<th>EasyPark</th>
<th>IPI-DataEx</th>
<th>OCHP</th>
<th>INRIX</th>
<th>Smart Parking</th>
<th>Park API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking area</td>
<td>Location</td>
<td>Location</td>
<td>Parking lot</td>
<td>Location</td>
<td>Parking lot</td>
</tr>
<tr>
<td>Street</td>
<td>Street</td>
<td>Street</td>
<td>Address</td>
<td>Street</td>
<td>Address</td>
</tr>
<tr>
<td>Postal code</td>
<td>Postal code</td>
<td>Zip code</td>
<td>zip</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>City</td>
<td>Locality</td>
<td>City</td>
<td>City</td>
<td>City</td>
<td>Region</td>
</tr>
</tbody>
</table>

* Bold words are those that occurred the most, while cursive words only occurred once.

The summary of tariff related words (see Table 24) shows that tariff is the most commonly used word to refer to a collection of parking prices. However, IPI-DataEx also uses tariff in its vocabulary, but in this case, are referring to the charged fee. Other words that are used by a majority are Start Time, End Time, and Day of week. EasyPark was alone in using the term day type. However, the usage of day type also includes other days such as holiday, and day_before_a_red_day (the day before a
holiday). Another difference (not included in the summary) is that EasyPark alone uses the word buckets to describe fixed prepaid prices.

**Table 24: Summary of tariff related word** [26], [62], [64], [68], [69]

<table>
<thead>
<tr>
<th>EasyPark</th>
<th>IPI-DataEx</th>
<th>OCHP</th>
<th>INRIX</th>
<th>Smart Parking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tariff</td>
<td>Rate</td>
<td>Tariff</td>
<td>Rate</td>
<td>Tariff</td>
</tr>
<tr>
<td>Price(^1)</td>
<td>Rate(^1)</td>
<td>Price(^1)</td>
<td>Rate(^1)</td>
<td>Rate(^1)</td>
</tr>
<tr>
<td>Price(^2)</td>
<td>Value(^2)</td>
<td>Price(^2)</td>
<td>Rate(^2)</td>
<td>Rate(^2)</td>
</tr>
<tr>
<td>Interval</td>
<td>Period</td>
<td>Step Size</td>
<td>Increment</td>
<td>Duration</td>
</tr>
<tr>
<td>Max time</td>
<td>Max duration</td>
<td>-</td>
<td>-</td>
<td>Max stay period</td>
</tr>
<tr>
<td>Valid from</td>
<td>Valid start</td>
<td>Start date</td>
<td>-</td>
<td>Inclusion</td>
</tr>
<tr>
<td>Valid to</td>
<td>Valid end</td>
<td>End date</td>
<td>-</td>
<td>Exclusion</td>
</tr>
<tr>
<td>Start time</td>
<td>Start time</td>
<td>Period begin</td>
<td>Time(_{in})</td>
<td>Start time</td>
</tr>
<tr>
<td>End time</td>
<td>End time</td>
<td>Period end</td>
<td>Time(_{out})</td>
<td>End time</td>
</tr>
<tr>
<td>Day type</td>
<td>Day of week</td>
<td>Week day</td>
<td>dow (day of week)</td>
<td>Day of week</td>
</tr>
</tbody>
</table>

* Bold words are those that occurred the most, while cursive words only occurred once.

\(^1\) These words are equivalent to the price in parking price.

\(^2\) These words refer to the cost value of a parking price.

Regarding occupancy (see Table 25) there were few words that were used by a clear majority. This was probably caused by the differences in the approach of the different APIs. For example, the approach used by INRIX was focused on probabilities and showing an approximation of the current occupancy. The approach of OCHP was focused on individual statuses, which reflects their focus on electric charging stations. The approach of IPI-DataEx was more of a mix of the others, which reflects their aim towards being a standard.

**Table 25: Summary of occupancy related word** [62, 64, 66, 68, 72]

<table>
<thead>
<tr>
<th>IPI-DataEx</th>
<th>OCHP</th>
<th>INRIX</th>
<th>Smart Parking</th>
<th>ParkAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand</td>
<td>Occupancy</td>
<td>Occupancy</td>
<td>Occupancy</td>
<td>-</td>
</tr>
<tr>
<td>Number of Spaces</td>
<td>-</td>
<td>Space Total</td>
<td>-</td>
<td>Total</td>
</tr>
<tr>
<td>Space Count</td>
<td>-</td>
<td>Available</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>(Demand) Count</td>
<td>-</td>
<td>Spots took</td>
<td>Occupied</td>
<td>-</td>
</tr>
<tr>
<td>Space</td>
<td>Spot</td>
<td>Spot</td>
<td>Bay</td>
<td>-</td>
</tr>
</tbody>
</table>

* Bold words are those that occurred the most, while cursive words only occurred once.
3 Protocol development

This chapter presents the development process of the proposed protocol. Additionally, this chapter mentions the evaluations which have been made as part of the development.

3.1 Development Stages

The development consisted of three stages which focused on different parts of the protocol. Furthermore, each stage consisted of multiple iterations. Mainly, the task in each iteration was to modify the protocol’s format to resolve any deficiencies detected during the evaluations of the previous iteration. Therefore, each iteration ended with an evaluation of the protocol. For the development to proceed from any given stage, the protocol had to meet the criteria for that stage.

The different stages of development and evaluation are presented in Sections 3.1.1 and 3.1.2. The evaluations and results are presented in Section 5. The metrics used during the evaluation were described in Section 2.4.2. The roles and use cases used during the development were described in Sections 2.4.3 and 2.4.4.

3.1.1 Initial format

The focus of the first stage was to build an initial structure for the protocol’s format. This served as a base for subsequent development. The process of defining the initial format started with identifying uses cases and roles. Once the use cases and roles were identified, the relevant data fields from the literature study (see Section 2.3) could be picked out. Then UML classes were made to formally describe the data fields, making the development of the protocol’s schemas easier. The protocol schemas are available in Appendix A.

The evaluation during the first stage was primarily a theoretical evaluation in which the defined elements were compared to the elements used by existing APIs and systems. During this stage, the protocol was mainly evaluated by considering the metric comprehensive (i.e., are the elements comprehensive with respect to the information that might be desired for all the use cases and by all the roles). Basically, if an element was missing or cannot be expressed equivalently with defined elements, the protocol’s format was considered to be deficient. The stage was considered complete when missing elements could no longer be identified.

3.1.2 Tools for writing and parsing

During the second stage, the focus was still mainly on the element fields of the format. However, the focus was also to develop tools for parsing and writing data, to the protocol’s format. Additionally, to be able to further test the protocol’s format, conversion tools were developed. The evaluation for the second stage focused on the metrics comprehensive and relevant. The process of developing these tools was also
part of the evaluation of the format, as strange words could be identified during the development. The format was also evaluated by converting data in other formats to the protocol format. If a conversion did not maintain the original meaning or there was a loss of information, then the format was considered deficient. Similarly, the converted data should be able to be reverted to the original format, without any loss of or generation of residual information. The second stage was considered complete when several formats could be losslessly converted between their format and the protocol format.

For the final stage, in addition, to further develop the protocol’s format, the focus was in developing an interface for the protocol. It was also during the final stage where any flaws with the tools could be identified and resolved. The final stage tried to evaluate the protocol according to all metrics (comprehensive, relevant, and speed).

The protocol was evaluated with a test consisting of format conversions. The main purpose of the test was to programmatically find any flaws in the protocol’s format. The test mainly consisted of converting a set of tariffs from their format into the protocol’s format and then converting the tariffs back into their original format. Mistakes were then shown by any information being lost or wrong when comparing the converted data with the original data. The tariff data, evaluation, test, and results are presented in Section 5.

### 3.2 Development challenges

Throughout the development, many issues have been. Some issues were corrected as soon as they were identified, while other caused a smaller delay in the development process. However, 2 issues that caused major changes and delay in the development process are the choice of wrong data format and time shortage. These 2 issues are described and presented in the following subsections.

#### 3.2.1 Wrong data format

One of the mistakes made was choosing the wrong data format. Initially, XML was the target data format for the protocol. However, after half of the development time had passed, it was determined that GraphQL only responds in JSON format. Therefore, the tools and schemas had to be converted for JSON. The lesson learned is that it is important to ensure that the chosen languages, formats, and frameworks work together. On the other hand, there are now some prior works that make it easier to also support XML format.

#### 3.2.2 Time shortage

Another issue with the development was lack of time. The development of the protocol required more time than initially planned. Therefore, most of the focus was put on developing the protocol for tariff data. However, some development of
location and occupancy data have still been made. The choice to prioritize Tariff data was because tariff data was determined to be the more doable part of dynamic pricing.

Some occupancy tracking solutions might work excellent for parking facilities, which on the other works poorly for on-street parking. Due, not being able to withstand: wear and tear from harsh weather conditions or traffics, or interference from the surrounding. Additionally, there currently exist multiple approaches for storing and sharing occupancy, where none are widely used. From the APIs and protocols (presented in Sections 2.2.3 and 2.3) there were already more than 4 different approaches. The approach can be towards collecting data from individual parking spots, the overall occupancy status (free in contrast to taken spots), the average overall occupancy status (throughout the day, week, or month), or as the probability to find free parking spots. However, these approaches, apart from individual parking spot status, mostly meant a difference of a single element field (see Section 4.1.2).

In the case of location data, it did not seem complex enough for any solution to cause any major improvements. The biggest difference, from existing solutions, is mostly a structure difference because the included elements are mostly the same (see Sections 2.3 and 4.1.1). However, the terminology might be different. Additionally, because the location data is static and not expected to frequently change, location data becomes simpler than tariff data. Therefore, the most focus was decided to be put on tariff data, as it needs a standardization.
4 Protocol description

This chapter presents the protocol and its different parts. Section 2.4.5 presented the terminologies of the protocols. Section 4.1 presents the protocol’s format with examples of how to express some parking data. Section 4.2 describes how the tools for writing and parsing the protocol’s format works. Section 4.3 presents the protocol’s interface.

4.1 Format description

The protocol’s format is based on JSON and therefore consist of the same components and features that JSON provides. More specifically the protocol uses elements, objects, and arrays. An element is defined as a simple key-value pair. An object is an element that encapsulates a variety of elements and is defined as JSON objects. Lastly, an array (or list) is an element that encapsulates multiple occurrences of a single element and is defined using JSON array.

The parking data included in the protocol’s format, are referred to as elements. The elements are further divided into three categories: location, occupancy, and tariff. The three categories act as root objects and hold the other elements together. The root elements are separated into three different arrays under a parent object. Arrays are used for the root element as there might be multiple occurrences of each type. Any object that has multiple occurrences, such as the root objects, include a unique identification string. The three categories and what they include is explained in the following subsections.

4.1.1 Location standard

The purpose of location data is to map parking-related information to a location. Therefore, location data includes information regarding the physical address, contact information, geospatial coordinates (geoLocation), and other location related information such as the elements parentLocation – an identifier to bind a sublocation to a major location, and sublocations – a list of identifiers that link the major location to its sublocations. Compared to the other two categories (occupancy and tariff) location data are more static and is not expected to change often. Mandatory for the location data is to include a globally unique identifier for each location. For a class diagram of location data see Figure 9.

Some PO or PP might want more advanced geographical data such as polygon markups in a map. A commonly used format which provides such functionalities is KML. However, KML is a format based on XML with their own definition for file structures and might be hard to include in the JSON representation. Additionally, the companies might use other formats than KML. Therefore, the format includes an object for attaching a reference providing the location of the polygon files. The polygon object also supports the use of encoded polygon data as an encrypted string.
Location data also includes log data to keep track of changes. Information that are included in the log data are: updated – date and time of when the data was latest updated, user – a string to identify the user who made the update, created – date and time of when the data was initially created, and creator – a string to identify the user who created the initial data.

![Class diagram of location data](image)

**Figure 9: Class diagram of location data**

### 4.1.2 Occupancy standard

Occupancy data are the information that describes the occupancy status of a parking location. The purpose of shareable occupancy data is to provide PPs (such as cities) with information on which to base new tariffs on. For a class diagram of occupancy data see Figure 10.

Occupancy includes information about how many parking spaces there are in total (supply) and how many that are occupied (both as a percentage and a total number). Both supply and occupied are part of the core information of occupancy data. Therefore, supply and occupied are mandatory elements for the occupancy data. For additional features, the occupancy data also includes the elements: average occupancy, the probability of finding a parking space, and individual parking space information. The purpose of the average occupancy information is to give a major idea of how well a pricing strategy has performed and the occupancy status. Like average occupancy, the purpose of probability element is to give a major idea of the current occupancy status. The purpose of individual parking space data is to give an accurate occupancy status. Occupancy data also include a log data
(described in Section 4.1.1). Occupancy data are identified by a location ID, as there is at most one occupancy object per location or sublocation.

The elements included for an individual parking space are: a local unique space identifier (for systems), a local space number (more understandable for humans), status (if it is occupied or not), how frequent the status is updated, parking space type, when an occupation started and its approximate ending time, and when the status was latest updated.

![Diagram of occupancy data](image)

**Figure 10: Class diagram of occupancy data**

### 4.1.3 Tariff standard

Tariff data is the object that describes the parking fees of a parking location. Tariff data is included in the protocol to enable fast and easy sharing of new tariffs. Parking tariffs are harder to express than occupancy and location data, as there exists a wide variety of how a tariff is currently expressed. The proposed protocol aims to express both the simplest and the most complex tariff while maintaining a generic method of expressing tariffs. For a class diagram of tariff data see Figure 11.

The tariff information is divided into three parts: *restriction, rate*, and *schedule* data. Tariff data also includes log data (described in Section 4.1.1). Because each location can have multiple tariff objects, each tariff must be provided with a globally unique tariff ID and mapped with a location ID. Each of these parts of the tariff is described in the following paragraphs.
4.1.3.1 Restrictions

Restrictions object includes information that limits or provide additional information regarding a tariff. The elements that restrict tariffs are: targetGroup – a token that restricts the tariff to a targeted group (such as residential, or personnel), vehicles – a token that restricts the tariff to a specific vehicle type (such as a car, truck, or bus), maxParkingTime – a number that defines the max allowed parking time in minutes, maxPaidParkingTime – a number that defines the max allowed parking time during paid hours, maxFee – a number that defines the overall maximum fee of the tariff, and minFee – a number that defines the minimum fee a parker is expected to pay. All these elements are optional to include in the format. Therefore, if any of the elements are unassigned it means that the tariff is not restricted in any of the specific ways. For example: if targetGroup is not defined, it means that the tariff is available to the public, and if maxParkingTime is not defined, it means that there is no parking time limit. The only exception is the element maxFee – which might still be restricted under rates (see paragraph 4.1.3.2) as daily or weekly maximums.

The elements that provides additional information for a tariff are: tariffType – a token that defines how prices are charged/applied (such as fixed or regular), prepaid – a Boolean that defines if the tariff requires advanced payment, and resetTime – a number that defines the time during a day when a tariff resets the fee calculation to the first rate. The optional elements are prepaid, and resetTime. The prepaid
element is false by default and if no resetTime is defined, it means that the rates will never reset unless the driver restarts the parking themselves.

4.1.3.2 Rate

The rate is the smallest component of a parking fee, which describes the price development. The rate component describes the fee as a value over a time interval in minutes. For more advanced price strategies, a tariff can consist of multiple rate components. Therefore, each rate must a number defining in which order the rates applies.

A rate object includes the elements: order – a number that defines in which order the rates applies (lowest first), value – a number that defines the rate to charge drivers, interval – a number that defines how long the rate applies for, intervals – a number that defines how many times the rate applies before next rate applies, repeat – a Boolean that defines if the rate repeats (infinitely unless restricted in other ways), unit – token that defines the date time unit the interval is expressed in, max – a Boolean which defines if the rate is a max fee, and countOnlyPaidTime – a Boolean used by fixed prices that defines if the parking time should deduct paid parking time only. The mandatory elements are order, value, and interval, where order must be locally unique.

```
{
  "rates": [
    {"order":1,"value":10,"interval":60,"intervals":1},
    {"order":2,"value":0,"interval":60,"intervals":1},
    {"order":3,"value":20,"interval":60,"intervals":22}
  ]
}
```

Figure 12: Example of a tariff defined for one day (SEK)

The simplest tariff with a constant cost per time units can be expressed using one rate component. However, a tariff can be more advanced than a constant cost per time unit, which requires more than 1 rate component to express the tariff. For example: see Figure 12, which describes a pricing strategy: the first-hour costs 10 SEK, the second hour is free, and thereafter the cost is 20 SEK per hour. Because no unit was provided in the rates, the interval is assumed to be in minutes. Apart from consisting of several rate components, a complex tariff might also include both regular and fixed rates. In such case, the fixed rates must be expressed differently than the regular rates. For example: see Figure 13, which describes the pricing strategy: the first hour has a fixed rate of 15 SEK, and thereafter the cost is 10 SEK per hour.
Figure 13: Example of a tariff with mixed rates (SEK)

Table 26: Date time definitions

<table>
<thead>
<tr>
<th>Value unit</th>
<th>Time (min)</th>
<th>Cal. Day</th>
<th>Cal. Week</th>
<th>Cal. Month</th>
<th>Cal. Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hour</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1 day</td>
<td>1 440</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1 week</td>
<td>10 080</td>
<td>7</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1 month</td>
<td>40 320 to 44 640</td>
<td>28 to 31</td>
<td>4</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1 year</td>
<td>525 600 to 366</td>
<td>365 to 366</td>
<td>52 to 366</td>
<td>12 to 53</td>
<td>1</td>
</tr>
</tbody>
</table>

Apart from complex tariffs, a tariff might also have maximum fees such as daily, weekly, or monthly. However, a day, week, or month can be defined differently, depending on the PP, PO, or the country. Therefore, the rate object includes the element unit which allows a PP to define the date time unit. Some possible definitions and values for the different units were shown in Table 26. Using the rate objects, it is, therefore, possible to define different types of max fees. Max fees are recognized by the element max being set to true. For example: see Figure 14, which describes the pricing strategy: 10 SEK per hour, with a daily maximum fee of 60 SEK (24 hours), and a weekly maximum fee of 200 (full week i.e. 168 hours), expressed in minutes.

Figure 14: Example of a tariff defined for one week (SEK)

4.1.3.3 Schedule

There are two types of schedule objects: active- and validSchedule. The activeSchedule is used to define when a tariff is active, and restricts a tariff based on the elapsed- and current time. An activeSchedule can be interpreted as rules to when a tariff is applicable. This is necessary to be able to define time periods when the parking is free. The activeSchedule are also used to differentiate which tariff to use when a single location has multiple tariffs. For example, parking after a specific time might have a cheaper tariff or be free. The validSchedule, on the other hand, restrict
when a tariff is valid and usable based on when the parking started. The \textit{validSchedule} is necessary when defining early bird tariffs (described in 2.1.1), where a driver get cheaper rates if the driver starts parking inside a \textit{specific} time period. In this case, there would be two tariffs (one early bird and one regular), which would have the same \textit{activeSchedule} but different \textit{validSchedules}. More specifically, the early bird tariff is valid inside the time period, while the regular is valid outside the time period. Thus, the two types of schedules can be interpreted as two layers of time regulations, where the \textit{validSchedule} overrules the \textit{activeSchedule}.

The \textit{activeSchedule} object includes the elements: \textit{startTime} and \textit{endTime} – two numbers that define when the tariff is active, and \textit{days} – list of tokens that define which days the tariff is active. An \textit{activeSchedule} might need more than one schedule unit to express the intention because the active time might vary per day. Therefore, each schedule requires a locally unique ID. For example, Figure 15 shows a tariff is active during 7-17 on weekdays and 7-14 on Saturdays, and thus it requires at least two schedule units to describe. Note that there is no time defined for Sundays, hence, there is no parking fee during Sundays.

\begin{verbatim}
{"activeSchedules":[
{"startTime":420,
 "endTime":1020,
 "days": ["MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY"]
 },
{"startTime":420,
 "endTime":840,
 "days": ["SATURDAY"]
 }]
\end{verbatim}

\textbf{Figure 15: Example of active schedules}

The \textit{validSchedule} object includes the elements: \textit{validFrom} and \textit{validTo} – mandatory field of dates that define a period the tariff is valid, \textit{validTimeFrom} and \textit{validTimeTo} – numbers that define when during a day the tariff is valid, and \textit{validDays} – a list of days which defines the days the tariff is valid. Like \textit{activeSchedules}, multiple \textit{validSchedules} might be necessary, and therefore locally unique IDs is required. For example: see Figure 16, which says that a tariff is valid between 1 January 2018 to 30 June 2018, Mondays to Fridays, from midnight to 09:00, and from 21:00 to 00:00.
A schedule does not require every element to be defined. Therefore, some assumptions are to be made when these elements are missing (i.e. default values). For active schedules, if no days are defined the schedule should not be restricted to a specific day (i.e. the tariff should be active every day). Similarly, if no times are defined it should be active for the whole day (start and end are defined by reset time or 00:00 local time). In contrary to the other elements, validFrom and validTo must be assigned, or be considered invalid. On the other hand, if a day or time slot is not covered by any schedule for any tariff, the parking time is to be assumed to be free of charge.

The time values (startTime, endTime, validTimeFrom, and validTimeTo) have a time accuracy in the minute range. The minute range was decided upon because an assumption was made, that no PP or operators would define time values within the range of seconds. Additionally, to prevent the time values from clashing some standard definitions were made. The startTime and validTimeFrom start defined exactly on the second as the minutes define, for example, 60 (01:00) defines the start time 01:00:00. In contrary, the endTime and validTimeTo is defined as 1 second before what the minutes define, for example, 1380 (23:00) defines the end time 22:59:59.

4.2 Tools description

The protocol tools are written in the programming language JavaScript for the Nodejs runtime. JSON is native to JavaScript and Nodejs provides several libraries for developing server applications with the GraphQL interface. Therefore, JavaScript was a good choice for writing the protocol tools. These tools are divided into three categories: writing tools, parsing tools, and conversion tools. These tools are presented in the following paragraphs.
4.2.1 Writing tool

The purpose of the writing tool is to automate as much work as possible to make future integrations easier. The writing tools mainly provide two kinds of functionalities: the creation of objects (create) or adding an element to an object (add). The objects that can be created are location, occupancy, tariff, rate, and schedule.

Utilizing a create function creates an empty template. Depending on which object, the function might require some additional information, such as an object ID or a parent ID. The add function requires the parameters: an object, tag of the parent element, the tag of the element, and value of the element. If the parent element tag is either not provided or does not exist for the object, then the tools automatically find the parent tag or creates the parent element as well as adds the parent to the root element. If an element already exists, then the value is simply overwritten. However, if the parent element is an array, the value is appended to the list.

4.2.2 Parsing tools

The main purpose of the parsing tool is to make the protocol’s format easier to parse (if the structure is not known) and to make it easier to integrate into existing systems. The elements of the protocol’s format are complex to parse, because the data may contain many nested arrays and objects. Therefore, the main function of the parser flattens the hierarchy of the elements. Flattening results in a list of key-value pairs that maintain the original order of elements, making the parsing more straightforward. For an example: by flattening the example schedule (shown in Figure 15) results in a flat list with no hierarchies, see Figure 17.

```json
[{
  "key": "type",
  "value": "activeSchedules"
},
{
  "key": "startTime",
  "value": 420
},
{
  "key": "endTime",
  "value": 1020
},
{
  "key": "days",
  "value": "MONDAY"
},
{
  "key": "days",
  "value": "TUESDAY"
},
{
  "key": "days",
  "value": "WEDNESDAY"
},
{
  "key": "days",
  "value": "THURSDAY"
},
{
  "key": "days",
  "value": "FRIDAY"
},
{
  "key": "startTime",
  "value": 420
},
{
  "key": "endTime",
  "value": 840
},
{
  "key": "type",
  "value": "days"
},
{
  "key": "days",
  "value": "SATURDAY"
}]
```

Figure 17: Example of a flatten schedule

4.2.3 Conversion tools

The conversion tools are written in Nodejs like the protocol tools. Additionally, the conversion tools are built using the protocol tools. The intention of the conversion
tools is to provide developers with rough examples of how conversions into the protocol’s format can be made. Developers can, therefore, use the conversion tools as suggestions towards their own implementation. There were mainly 2 kinds of tools developed: tools for converting data into the protocol’s format, and tools to convert from the protocol’s format. Where the first have been developed for 2 sets of tariff data, initially having 2 different structures and format (the formats are presented in Section 5.1.3)

For the first set of tariffs (EasyPark as source), the conversion tools are more comprehensive than the tools for the second set (acquired company as source). This, because the first data set include all information that corresponds to the whole format of the protocol (see Section 4.1.3), while the latter only includes rate data. Additionally, the conversion tools for the first set include two kinds of tools (conversion into the protocol’s format and conversion back into EasyPark’s format), while the latter only includes the first kind (conversion into the protocol’s format).

The conversion tools mainly follow the strategy: divide and conquer. A small part of the tariff is converted at a time. Where each later conversion adds additional information. Similarly, each tariff is converted one at a time, for one location at a time. Therefore, the tariff data is first separated by location and tariff. Additionally, the conversions begin with converting all information as it is and then adjusts the converted data, to better fit the protocol’s format. Some adjustments made are: merging of similar rates, duplicate or irrelevant data being removed, or reductions of value sizes (while maintaining the meaning).

For some tariffs, the rate for every hour is defined, even if each hour has the same rate. These rates that are repeated two or more times, can be merged into a single rate. Therefore, removing 2 duplicate rates. For example, the rates: 10 SEK first hour, 10 SEK second hour, and 10 SEK third hour, can be adjusted into the single rate: 10 SEK per hour, for 3 hours.

Value reductions made are for those cases where the rate is not fixed but are defined with an unnecessarily long period, which can be difficult to associate. These rates can be reduced to make it more understandable. For example, the unfixed parking rate: 170 SEK per 1020 minutes, can be reduced to 10 SEK per hour (60 minutes), for 17 hours. However, it is important that the rates do not become more complex (resulting in an irrational number). It is even more important, that the meaning of the rates do not change.

4.3 Interface description

The protocol’s interface is developed as a GraphQL interface, running on the express server application on top of the Nodejs runtime. Similarly, to the conversion tools, the intention of the interface is to provide developers with some guidelines on how the interface can be implemented. However, because it is hard to predict how every system and database is used, the protocol’s interface should be used as a template or as an example for the development of an interface.
4.3.1 Format and query schema

The main purpose of the format schema is to impose a standard for the protocol’s format. The schema describes the query examples and the elements that are to be accessible through the interface. The format schema describes the elements that are included in the protocol’s format. The schema also provides more detailed information which described the hierarchies and structures of the format objects. The GraphQL schema can be found in Appendix A.

4.3.2 Server application

The server is a simple express server implementation. The main functionality of the server is to provide clients with schema information and handle client queries by invoking the right logic functions.

The logical functions consist of empty function which hints towards how they should resolve. fetches stored data and depending on the format they also translate the data into the protocol’s format. The logics are further split into two parts: query and mutation logics. The query logics consist of those functions who are responsible for fetching and provide the client with the right data. Mutation logics are those function that is responsible for updating databases.
5 Evaluation and discussion

The evaluation of the protocol was divided into two parts: the protocol’s format, and the protocol’s interface. The evaluations of the protocol’s format are presented in Section 5.1. The evaluation of the protocol’s interface is presented in Section 5.2. The metrics used during the evaluation were first presented in Section 2.4.2. The protocol developed in this thesis, will from here on be referred to as “the protocol”. The protocols which were not developed in this project will be referred to as “other protocols”, often with a reference to Section 2.3 (page 15).

5.1 Format evaluation

The evaluation of the format has been divided into 2 parts: location and occupancy data (see Section 5.1.1), and tariff data (see Sections 5.1.2 and 5.1.3). The protocol’s format was mostly evaluated according to the metrics comprehensives and relevance. The evaluation of the metric speed was omitted for the protocol’s format. This because it was determined that the format would only have a minor impact towards the speed. However, by ensuring the format only include relevant information, the format might have some minor impact on the speed. The metrics reliability and security were omitted from the evaluation of the protocol’s format. This because any changes to the protocol’s format would have a minimal impact on these metrics.

5.1.1 Location and occupancy data

The evaluation of the location and occupancy data were much simpler compared to the evaluation of tariff data. This, because location data and occupancy data have less complex definitions than tariff data (described in Section 3.2.2). Therefore, the evaluation of location and occupancy data have been more focused on satisfying the metric comprehensive. Additionally, with the use of GraphQL (described in Section 2.2.2, paragraph 2.2.2.7 and Section 4.3), sent data are managed by the client-side through queries.

In comparison to existing APIs (see Sections 2.2.3 and 2.3) the protocol covers the most location related elements that are important. One major difference is that instead of including geospatial polygon information directly in the format, the protocol makes use of the already well developed KML format. Another difference lies in the terminology and structure. Anyhow, any elements that might be missing can simply be added to the protocol whenever it is discovered.

With occupancy data, it is difficult to determine which elements that are the most relevant or required (described in Section 3.2.2). Therefore, the protocol’s format should cover all elements that might be relevant for occupancy data. The relevant elements are those that are in use by any APIs or other protocols (see Sections 2.2.3 and 2.3). Anyhow, disregarding individual space data, the ID, and log object, occupancy data consist of considerably few elements (7 unique elements).
Additionally, when an occupancy approach has been proven to be better than the others, the protocol might already support the required elements. Furthermore, any irrelevant element can then simply be ignored.

5.1.2 Tariff data

Tariff data has shown to be more complex to define than it initially seemed. There might be a big difference on how they are defined, even for a single operator, depending on which person that set up the tariff. The differences might also be different depending on the approach of a provider and what they try to achieve. Therefore, the evaluation of tariff data has been focused on the metrics comprehensive and relevance (described in section 2.4.2). Comprehensive is important because the format must be able to cover all possible tariffs while staying as generic as possible. This also makes the relevance of the included elements important. With too many irrelevant elements, the complexity of the format would be unnecessary high. However, tariff data is complex, and we cannot know if we have covered all kind of tariffs. Therefore, we can only cover every kind we have access to and increase the coverage with future developments.

It was possible to cover many kinds of tariffs using the format. This was verified by manually converting as many kinds of tariffs as possible. However, doing it manually was slow and could only be used to verify a limited number of tariffs. Therefore, we developed conversion tools (described in Section 4.2.3) to automate the conversion process. But, due to a large number of tariffs, it was hard to determine the correctness of the converted data. Therefore, the correctness was verified programmatically by conducting few correctness tests, which are presented in Section 5.1.3.

A minor flaw of the protocol is that the terminology and definitions might be too influenced by the terminology of EasyPark. This is caused by whenever a missing element is added, the names come in firsthand from EasyPark. However, this issue can be solved by keeping the future development as open-source. By taking opinions from other parties in the parking industry (such as operators and PP), counterintuitive elements can be identified and adjusted.

5.1.3 Correctness tests

The tariff data used in the tests were provided by EasyPark. The tariff data includes tariffs from multiple parking locations around the world. Furthermore, any tariff data used was not handpicked, instead, the tariff data was randomly chosen, to cover as many countries as possible. This leads to a variety of tariffs that are set with different approaches and pricing strategies. For example, some tariffs in Italy and France have irregular rates that change every minute. In contrary, some tariffs in Norway are defined with a single hourly rate that is regulated by different maximum fees. With this, we aim to include edge cases which otherwise would have been overlooked. The initial format of all tariff data is tabular data, stored in a database.
However, to make the testing easier and eliminate the need for a database, the data was fetched and exported as CSV-files in beforehand.

The first set of tariff data origins from EasyPark and is divided into two types: bucket, and regular. In this context, bucket tariffs refer to tariffs that are fixed and paid in advanced. The rest of the tariffs fall into the category regular tariffs. Regular tariffs include both regular rates, fixed rates, irregular rates and linear rates. The key point is that regular tariffs are only charged for the actual parking time, or at least for the expected parking time. In contrary to bucket tariffs, where a set of parking times are paid for in advance. The regular tariffs used 4 CSV-files which corresponds to the parts: valid schedule, active schedule, rates, and restrictions, of the protocol’s format. Bucket tariffs had only 3 CSV-files that corresponds to the parts: valid schedule, active schedule, and rates. More specifically, the tariffs used were around 2 000 in number, from around 2 300 different parking locations in 11 countries. The reason the number of tariffs is lower than the number of parking locations is due to tariffs are reused for multiple locations (i.e., each tariff can belong to multiple locations). Similarly, each location can have multiple tariffs. The test using the first set of tariff data are presented in the paragraphs 5.1.3.1 and 5.1.3.2.

The second set of tariff data are rates acquired from a German company. The second set of data only consist of tariff rate information, stored in a single file. In contrary to the first set of tariff data, the second set has another format and structure. The parking rates are defined using spans of times and values. For example, a regular parking rate of 10 SEK per hour is defined as: from the time 0 – 1440, the rates are 0 – 240. At whole, there are tariff rates for 1000 different tariffs. The test using the second set of tariff data are presented in paragraph 5.1.3.3.

5.1.3.1 First test specifications

The main method for the first correctness tests was to first convert the tariff data into the protocol’s JSON format. Secondly, the tariffs were converted back into the initial format (CSV). In the last step, the converted CSV-files were programmatically compared to the original CSV-files. The code comparing the CSV-files are written in JavaScript for Nodejs runtime.

The actual comparison was made by comparing 2 blocks of rows at a time, 1 block of the original files with 1 block from the converted files. More specifically, a block consists of multiple rows, whereas a row consists of multiple element fields. A block is determined by each row having the same specific ID fields. The 2 blocks are then compared depending on the ID fields. For valid information the location ID is used, while for restrictions, active, and rate information tariff ID is used. For each block and row, there are four kinds of statuses: correct, incorrect, duplicated and missing. However, a block can only have one status at a time.

A correct block means that every row in the block is correct. A correct row means that every meaningful element of the row is correct compared to the original files. A meaningful element is an element that provides the tariff with useful information.
Basically, all information is useful, except for IDs that do not bind a row of information to another row of the same or another file. A duplicated block means that every row in the block is correct but at least one of the rows was duplicated after the conversion. However, the number of duplicated blocks only counts unique blocks that have been duplicated. A duplicated row means that the row is a duplicate of another row. In contrary to a duplicate block, a duplicate row may contain both correct and wrong elements and every duplicated row is counted towards the total number.

An incorrect block means that at least one row in the block is incorrect. An incorrect row means that at least one meaningful element of the row was wrong compared to the original files. An incorrect block may also contain duplicate rows. A missing block is basically an incorrect block where at least one row is missing. A missing row is a row that was lost during the conversions and therefore not found in the converted files. However, a missing block may contain both incorrect and duplicate rows.

5.1.3.2 First test result

The conversion test of the tariffs (see Tables 27 and 28) showed that no meaningful information was lost after the conversions. However, some of the data were duplicated, and some information was incorrectly converted. The most common incorrect conversions were incorrect conversions of start (also from) and end (also to) times. Of all 5956 incorrect rows, 5955 rows were incorrect converted time values. However, the time difference was at most 59 seconds off. This initially indicates that there is an issue with the time accuracy.

When the issue was further investigated, it was determined that the main cause was a lack of a standard for the tariff data. Depending on which person who set up the tariffs, the set times could be between 0-59 seconds off from the intended time. More specifically, start times were at most 1 seconds behind the intended time. While end times were up to 60 seconds before the intended time. For example, a start time of 06:00 was defined as either 06:00:00, or 06:00:01. While an end time of 22:00 could be defined as 21:59:00, 21:59:59, or 22:00:00. Because the format follows a set of standards (see Section 4.1.3, paragraph 4.1.3.3), this issue appeared. However, the standard set for the format must have been the same as the most commonly used approach. Otherwise, the number of incorrect rows would have been more than the number of correct rows (see Tables 27 and 28, Valid and Active columns). On the other hand, this shows that the format can be used to impose format standards on existing tariff data (because all time values after the conversion followed the standard definition).
Table 27: Result from comparing the converted data with the original data (regular tariffs from EasyPark)

<table>
<thead>
<tr>
<th>Status</th>
<th>Valid</th>
<th>Restrictions</th>
<th>Active</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total blocks:</td>
<td>1361</td>
<td>1106</td>
<td>1106</td>
<td>467</td>
</tr>
<tr>
<td>Correct blocks:</td>
<td>1213</td>
<td>929</td>
<td>553</td>
<td>282</td>
</tr>
<tr>
<td>Correct rows:</td>
<td>15326</td>
<td>929</td>
<td>4489</td>
<td>843</td>
</tr>
<tr>
<td>Incorrect blocks:</td>
<td>32</td>
<td>1</td>
<td>504</td>
<td>0</td>
</tr>
<tr>
<td>Incorrect rows:</td>
<td>737</td>
<td>1</td>
<td>3978</td>
<td>0</td>
</tr>
<tr>
<td>Duplicated blocks:</td>
<td>116</td>
<td>176</td>
<td>49</td>
<td>185</td>
</tr>
<tr>
<td>Duplicated rows:</td>
<td>1819</td>
<td>624</td>
<td>3870</td>
<td>2134</td>
</tr>
<tr>
<td>Missing blocks:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Missing rows:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The last incorrect row (Restrictions) was after some investigations determined to be caused by the tariff been set up wrongly. A parking time limit of 0 seconds had been assigned. This was determined to be a mistake because a parking time limit of 0 seconds is not likely. However, the conversions were instead able to remove this mistake in the initial format.

Duplicated blocks and rows were caused by some locations being bound to the same tariff. However, in the protocol’s format, each tariff is considered unique for each individual location. Therefore, when converting the tariff back into EasyPark’s format, the individual tariffs are kept which results in multiple occurrences for some tariff. The duplicated blocks do not cause any issues for the tariffs, because the information is still correct, and the correlations are intact. When converted to the protocol’s format, the data consisted of 2175 locations and 2566 tariffs.
Table 28: Result from comparing the converted data with the original data (bucket tariffs from EasyPark)

<table>
<thead>
<tr>
<th>Status</th>
<th>Valid</th>
<th>Rate</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total blocks:</td>
<td>955</td>
<td>825</td>
<td>122</td>
</tr>
<tr>
<td>Correct blocks:</td>
<td>895</td>
<td>693</td>
<td>52</td>
</tr>
<tr>
<td>Correct rows:</td>
<td>12 678</td>
<td>689</td>
<td>375</td>
</tr>
<tr>
<td>Incorrect blocks:</td>
<td>60</td>
<td>0</td>
<td>57</td>
</tr>
<tr>
<td>Incorrect rows:</td>
<td>729</td>
<td>0</td>
<td>511</td>
</tr>
<tr>
<td>Duplicated blocks:</td>
<td>0</td>
<td>132</td>
<td>13</td>
</tr>
<tr>
<td>Duplicated rows:</td>
<td>0</td>
<td>434</td>
<td>302</td>
</tr>
<tr>
<td>Missing blocks:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Missing rows:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

5.1.3.3 Second test specification

For the second test, we used the second set of tariff data (the German tariffs presented in Section 5.1.3). The second test was run 10 times, and the result presented in Table 29 shows the average and median values from these tests. The test was conducted 10 times because the first two tries, produced results with some differences. The test was then reconducted and stopped after the 10th try. Where it was determined, that the differences occurred randomly and by no specific cause. The main method for the second correctness test was similar to the first test (presented in the earlier paragraphs). The test begins by converting the tariff rates into the protocol’s format. The 2 sets of rates (one in the German format and the other in the protocol’s format) are then compared pairwise, the initial rate with the converted counterpart. However, instead of converting the data back into the German format, the expected parking fees are calculated and compared. If the fees are different for 2 corresponding tariffs, the difference proves that information was either lost or incorrectly converted. Each pair of rates were compared for 1000 random parking times (between 0 – n minutes, where n is the maximum allowed parking time + 100). To make sure the test included the 2 edge cases, minimum and maximum parking fees, the parking times 1 and n minutes were injected as the 2 first fee calculations for every pair. The maximum parking time was substituted with n (maximum parking time + 100) to ensure the max parking time was reached.

Some knowledge was lacking regarding the German rates. Therefore, some assumption had to be made during the fee calculations. Any parking time exceeding the maximum parking time allowed was interpreted as the maximum parking time allowed. Hence, n was used as the substitute for the maximum parking time. Any starting fee (greater than 0) is assumed to correspond to the first minute of the
period. For example, the rate with the period: 0 - 100 minutes, and the fee: 10 - 100 SEK, is calculated as the first-minute costs 10 SEK, while the rest (99 minutes) costs 80 SEK. The formula presented in Equation (1) was used to calculate the expected fees whenever a parking time is in the middle of a rate period. However, if the parking time has reached the end time of the period (i.e. parking time = end time of the rate period), then the maximum fee is the total fee.

\[
	ext{Fee} = \frac{T - P_{\text{start}}}{P_{\text{end}} - P_{\text{start}}} \times (F_{\text{max}} - F_{\text{min}}) + F_{\text{min}}
\]

where

- \( T \): Total parking time
- \( P_{\text{start}} \): The start time of the rate period
- \( P_{\text{end}} \): The end time of the rate period
- \( F_{\text{min}} \): The minimum fee for the rate period
- \( F_{\text{max}} \): The maximum fee for the rate period

The calculations used for the rates in the protocol’s format (described in Section 4.1.3, paragraph 4.1.3.2) followed another method. The fees are calculated as being additive, starting from the first rate. If the parking time is greater than the interval, a rate component is defined for, the full value is added to the total fee, while the interval is deducted from the parking time. Each rate is reused at most as many times as the element intervals is defined. If any rate component would cover a larger period than what is left of the parking time, then Equation (2) is used to calculate the part of the value added to the total fee. However, if the rates are fixed, then the full value is added to the total fee.

\[
F_{\text{part}} = \frac{T}{I} \times F
\]

where

- \( F_{\text{part}} \): The part of the rate value added to the total fee
- \( T \): Remaining parking time which has not been counted for in total fee
- \( I \): The time interval of the rate
- \( F \): The fee value of the rate

5.1.3.4 Second test result

Out of the 1000 tariffs converted, on average 585 tariffs resulted in fees that had some error (see Table 29). In total, roughly 75 000 of the calculation resulted in a difference in fee. However, none of the error was greater than 10⁻⁹ or was consistent, which shows that the issue is mainly a rounding issue.

Additionally, the tariffs could be converted into EasyPark’s format for tariffs, by using the conversion tools for the first test. This shows that it is possible to use the protocol as an intermediate step to convert tariffs to another format. Therefore, each
party (such as an operator) would only need to implement and learn about 1 format, to be able to communicate tariff data.

Table 29: Result from comparing calculated fees between converted rates and the German rates

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Average</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tariffs</td>
<td>1 001</td>
<td>1001</td>
</tr>
<tr>
<td>Tariffs that result in at least one error</td>
<td>585,4</td>
<td>585,5</td>
</tr>
<tr>
<td>Comparisons made</td>
<td>1 001 000</td>
<td>1 001 000</td>
</tr>
<tr>
<td>Incorrect fees with an error margin greater than 1E-09</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Incorrect fees</td>
<td>74 883</td>
<td>74 991</td>
</tr>
<tr>
<td>Greatest error found</td>
<td>$9.0949 \times 10^{-13}$</td>
<td>$9.0949 \times 10^{-13}$</td>
</tr>
<tr>
<td>Smallest error found (greater than 0)</td>
<td>$5.5511 \times 10^{-17}$</td>
<td>$5.5511 \times 10^{-17}$</td>
</tr>
</tbody>
</table>

5.2 Interface evaluation

The interface was mainly evaluated regarding the metrics relevance, speed, security, and reliability. The server-side of the interface has a small impact regarding the protocol being comprehensive, instead of being comprehensive is the responsibility of the format itself. However, whether the sent data is relevant is regulated by the client-side queries. Basically, the interface can only respond with those elements that a client has requested. Security and reliability are mostly solved utilizing TLS. Additionally, by outsourcing the security and reliability to TLS, there are fewer problems with the protocol. However, it is still the responsibility of the protocol to choose good applications and techniques.

For the evaluation of the interface’s speed performance, aspects that have an actual impact were identified. The aspects that were determined to affect the performance in terms of speed are: how fast the formats are converted, package size, link speed, and database. The first two aspects (conversion speed and package size) are dependent on the implemented interface, whereas the latter two aspects (link speed and database) can only be affected by the users, such as PP, operators, or drivers. Basically, the interface cannot affect the choice of links, especially as links are also restricted by what is available in different areas. Additionally, the interface does not impose any restrictions on the choice of database or the locality of the database (i.e. whether the database is on a local or remote network). Link speed and database are further described in the Section 5.2.3.

The aspects conversion speed and package size were evaluated with 2 tests. The first test, that measures the conversion speed is presented in Section 5.2.1. The second test which measures package and compressions sizes is presented in Section 5.2.2.
5.2.1 Conversion speed test

The specification of the computer used for the test is provided in Table 30. The test measured the fastest, slowest, average, and median time taken to convert tariffs. The tests were done in four steps: the first step converted tariffs for a single parking location, the second step converted tariffs for 10 locations, the third step converted tariffs for 100 locations, and the last step converted tariffs for 1000 locations. Each step of the test was reproduced 10 times and each location had at least 1 tariff. More information regarding the actual number of tariffs for each test is provided in Table 32. The tariff data was chosen according to the number of locations, instead of tariffs, mainly because the tariff data was mapped by location. Therefore, it was hard to determine the actual number of tariffs each location has, before converting the tariffs. Thus, it was not possible to restrict the test according to the number of tariffs.

Table 30: Computer specifications

<table>
<thead>
<tr>
<th>Type</th>
<th>Desktop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operative System</td>
<td>Windows 10 Pro</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel i7 3770 [73]</td>
</tr>
<tr>
<td>Base frequency</td>
<td>3.4 GHz</td>
</tr>
<tr>
<td>Max turbo frequency</td>
<td>Up to 3.9 GHz</td>
</tr>
<tr>
<td>Cores</td>
<td>4</td>
</tr>
<tr>
<td>Threads</td>
<td>8</td>
</tr>
<tr>
<td>Cache</td>
<td>8 MB Smart Cache</td>
</tr>
<tr>
<td>Bus Speed</td>
<td>5 GT/s DMI</td>
</tr>
<tr>
<td>Memory</td>
<td>16 GB DDR3</td>
</tr>
<tr>
<td>Motherboard</td>
<td>Sabertooth z77</td>
</tr>
</tbody>
</table>

The conversion speed was measured by conducting some evaluation tests. Basically, the test measures the time taken to convert tariffs into the protocol’s format. The tariff data used for the speed test is the same data used in the correctness test (see Section 5.1.3). The purpose of the test was to only measure the performance of the conversion. Therefore, the time taken to set up and get necessary tariff data was omitted from the time measurements. This was ensured by starting the timer only when all necessary data was prepared and stopping the timer as soon as the conversion was completed. The time was measured programmatically, by calling the JavaScript native functions `console.time` (at the start of the conversion) and `console.timeEnd` (at the end of the conversion). The time function had an accuracy of microseconds.
**Table 31: Result of the conversion speed test (note using a decimal comma.)**

<table>
<thead>
<tr>
<th>Locations</th>
<th>Fastest (ms)</th>
<th>Average (ms)</th>
<th>Median (ms)</th>
<th>Slowest (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.147</td>
<td>0.766</td>
<td>0.243</td>
<td>3.631</td>
</tr>
<tr>
<td>10</td>
<td>2.524</td>
<td>4.323</td>
<td>3.731</td>
<td>7.252</td>
</tr>
<tr>
<td>100</td>
<td>29.105</td>
<td>38.261</td>
<td>33.034</td>
<td>88.947</td>
</tr>
<tr>
<td>1000</td>
<td>326.135</td>
<td>376.260</td>
<td>344.679</td>
<td>641.709</td>
</tr>
</tbody>
</table>

The result of the test was presented in Table 31 and the approximate number of tariffs are presented in Table 32. Do note that there was no correlation between the slowest run in Table 31 and the most number of tariffs in Table 32. From these results, it was determined that the conversion time has a linear growth. Basically, the time taken grows with the number of tariffs converted.

**Table 32: Number of tariffs in relation to the number of locations**

<table>
<thead>
<tr>
<th>Locations</th>
<th>Fewest</th>
<th>Average</th>
<th>Median</th>
<th>At most</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>14</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>100</td>
<td>124</td>
<td>130</td>
<td>131</td>
<td>140</td>
</tr>
<tr>
<td>1000</td>
<td>1276</td>
<td>1316</td>
<td>1321</td>
<td>1340</td>
</tr>
</tbody>
</table>

To further investigate the linear growth, another test was conducted. In contrast to the first test, the number locations, for the second test, were consistently increased by 100 locations (starting from 100 and then up to 1000). The results from the second test are presented in Figure 18. The equation from the fitted line also indicates that the conversion times have a linear growth, with few irregularities.
In terms of speed, the conversion tools perform well, especially when only a few tariffs are converted. However, the time to convert tariffs increases linearly with the number of tariffs. Therefore, given these test results, a conversion of 1,000,000 tariffs would take roughly 9 minutes at most (on a computer with similar specifications). The rough estimation, 9 minutes, was calculated using the slowest time and the fewest number of tariffs. However, the only time when a conversion of 1,000,000 tariffs would be of interest, is during acquisitions of tariffs. In which case, 9 minutes is an acceptable time, not taking the time to modify the tools in consideration.

5.2.2 File compression test

Larger package sizes (where a package consists of a set of tariffs) take a longer time to transfer, compared to smaller package sizes. Depending on how the client-side queries are implemented, irrelevant elements can be omitted and thereby further decrease the package sizes. However, if only a few tariffs are transferred, additional elements would not have a large effect on the package size. In contrast, if many tariffs are transferred, additional elements would have a large impact, but in any case, the package would still be large. For example, a file containing 1000 tariffs have an average size of 1049 kB (see Table 33). GraphQL supports and recommends the use of GZIP to compress files. Therefore, a compression test has been made, to observe the compression provided by GZIP.
The tariff data used for the compression tests are the same tariff data used in the correctness test (see Section 5.1.3). Because each tariff is set up differently they also have different sizes. Therefore, to make the test fair, a random set of tariffs were used. In total there were 50 different files generated containing either 10, 50, 100, 500, or 1000 random tariffs (both regular and bucket tariffs), with 10 files for each amount.

The files were compressed using the Cygwin (a Unix environment for windows) implementation of GZIP using the compression settings: best (9), balanced (5), and fastest (1). These compression results compared with the files original sizes are presented in Table 33. Each column contains the average size for each test. Additionally, the resulting values are separated based upon the number of tariffs and which setting that was used for compression.

Table 33: Result from GZIP compressions tests using 3 different settings, with percentage compared to original size (note the use of the decimal comma.)

<table>
<thead>
<tr>
<th>Number of tariffs</th>
<th>Original size (kB)</th>
<th>Fastest (kB)</th>
<th>Fastest (%)</th>
<th>Balanced (kB)</th>
<th>Balanced (%)</th>
<th>Best (kB)</th>
<th>Best (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10,26</td>
<td>1,38</td>
<td>13,47%</td>
<td>1,10</td>
<td>10,75%</td>
<td>1,05</td>
<td>10,25%</td>
</tr>
<tr>
<td>50</td>
<td>52,27</td>
<td>5,00</td>
<td>9,56%</td>
<td>3,49</td>
<td>6,68%</td>
<td>2,94</td>
<td>5,63%</td>
</tr>
<tr>
<td>100</td>
<td>104,05</td>
<td>9,22</td>
<td>8,87%</td>
<td>6,22</td>
<td>5,98%</td>
<td>5,02</td>
<td>4,83%</td>
</tr>
<tr>
<td>500</td>
<td>528,08</td>
<td>44,17</td>
<td>8,36%</td>
<td>28,79</td>
<td>5,45%</td>
<td>22,23</td>
<td>4,21%</td>
</tr>
<tr>
<td>1000</td>
<td>1048,86</td>
<td>87,34</td>
<td>8,33%</td>
<td>56,32</td>
<td>5,37%</td>
<td>43,28</td>
<td>4,13%</td>
</tr>
</tbody>
</table>

From these test results, we can see that GZIP is able to compress JSON files to less than 14% of their original size. If the JSON files contain more than 50 tariffs, then the compression is even better, down to less than 10% of the original size, regardless of the setting. The balanced and best compression settings were significantly better, by at least 2% lower than the fastest setting. Additionally, the more tariffs the files contained, the better the compression was. The best setting for compression almost reached 4% of the original file size with 1000 tariffs.

5.2.3 Aspects outside the protocol control

Faster links perform better than slower links. However, the protocol does not regulate what kind of links are used. Furthermore, if fast links are used on the server-side, slow links might still be used on the client-side. However, by choosing fast links on the server-side, situations where both sides slow down the processing can be avoided. Additionally, the choice of links is also affected by the availability of the links.
The choice of the database also influences the performance of the interface. An incorrect choice of the database might have a negative effect on the overall performance. However, the choice of database is not regulated by the protocol, as it would unnecessarily put more responsibilities on the protocol. Moreover, carrying out research on databases was not the intention of this project. However, this flexibility allows PPs and operators to choose technologies they are more comfortable and experienced with. Therefore, in the end, improving the overall performance.
6 Conclusions

This project has mainly focus (1) dynamic pricing and (2) developing a protocol for sharing parking-related data. Dynamic pricing has proven to be a complex subject, as tariffs alone are a major subject itself. A summary of dynamic pricing is presented in Section 6.1.

The end goal of the project was not reached; however, a major step towards reaching this goal was taken. The developed protocol can be used to convert and share tariff data. The work that was completed is presented in Section 6.2. The identified ethical and sustainability issues are described in Section 6.3. The work that was not completed and possible future work, are presented in Section 6.4.

6.1 Dynamic pricing and tariffs

For dynamic pricing to be possible, the parking industry must be able to share location, occupancy, and tariff data. Location data is important for relating parking prices and occupancy data to a physical place or area. Occupancy data is also important, it is one of the key features in making dynamic pricing possible. Occupancy data (demand and supply) is the information that parking prices fundamentally are based upon (see Section 2.1). Additionally, being able to share tariff data is also important to utilize dynamic pricing. For example, when new parking fees have been determined, all the relevant parties, such as parking operators, must be notified of the changes.

However, a protocol for sharing parking-related data is insufficient to fully adopt dynamic pricing. Dynamic pricing requires better methods for collecting and displaying accurate occupancy data. There exist many methods for collecting occupancy data (see Section 2.2.1, paragraph 2.2.1.2). However, none of the methods for collection occupancy data works in every parking environment. Additionally, there exists some uncertainty in how occupancy data should be perceived (see Section 4.1.2). An equally important factor is better methods for calculating new parking fees. Currently, the existing methods are slow (see Section 2.1.2) and would end up becoming a weakness for a dynamic pricing system. Therefore, the methods for calculating parking fees must be automated and adapted to support real-time occupancy data.

Tariff data is a complex subject by itself. This data currently exists in a variety of formats and structures used for describing a tariff (see Section 4.1.3, paragraph 4.1.3.2). Thus, it is possible to define a tariff using different formats, which at first seems different. However, by analyzing these tariffs, it becomes obvious that they describe the very same parking prices. Additionally, there exist lots of workarounds that organizations are using, to express special cases of tariffs. Therefore, it has been hard (but not impossible) to develop a universal method for describing all possible tariffs. This has become harder since the purpose is also to remove the usage of these workarounds.
6.2 The developed protocol

Due to the limited time of this project, the focus of the development was mostly on tariff data. The reason, apart from limited time, is that the technologies and research regarding occupancy are not sufficiently mature. However, location and occupancy data were still included in the protocol as they are still critical parts towards dynamic pricing. In the end, even if the main goal would have been reached and the protocol was fully completed, it would still not be the complete solution toward dynamic pricing (mentioned in Section 6.1). However, the protocol can be considered as a partial solution toward dynamic pricing since different parties in the parking industry (such as PPs, and operators) can now properly share tariff data. Additionally, the protocol also defines a format for sharing location and occupancy data.

From the evaluations (see Section 5.1.3, paragraph 5.1.3.2), we were able to show that we can describe many kinds of tariffs without losing any meaningful information. However, the tariffs used were restricted to those tariffs we could acquire. Therefore, there might exist some tariffs that have been overlooked. The testing also showed that the protocol can be used to impose standards on existing data. The evaluation (see Section 5.1.3, paragraph 5.1.3.4) also showed that it was possible to use the protocol as intermediate-step when converting tariff data to a specific format. This means that it is possible to share tariff data between multiple parties, without prior knowledge regarding the other parties’ formats. The use of the protocol might cause duplication of data. However, the data relationship is kept intact. Additionally, with the surprisingly effective compression with GZIP, i.e., up to 95% compression rate (see Section 5.2.2), the duplicated data will not lead to significantly larger package sizes.

On the other hand, the terminology used may be too influenced by the approaches and methods of EasyPark. However, it has been discussed that the protocol will be onwards developed as an open source project. Therefore, this issue might be solved by gathering opinions from other parties during the development. Moreover, it might be hard to convince the whole parking industry to accept the protocol as a standard, especially since it is not a complete solution. However, the protocol is at least useful for EasyPark and their partners, as they now can uniformly share tariff data. Furthermore, EasyPark can recommend the protocol to their partners. Thus, creating an initial userbase for the protocol, which might further influence the use of the protocol by others.

Lastly, the protocol still does not have a name. Basically, it has been hard to determine a name which is intuitive, short, easy to remember or associate, and is not already taken. Therefore, it has been decided to not rush with the naming process, as bad names might have a negative impact on the perception of the protocol. However, it is important that the protocol is named before the protocol is released.
6.3 Ethics and Sustainability

Unregulated parking occupancy has a negative impact on the environment, by causing traffic congestion (see Section 1.1.1). Traffic congestion might further lead to environmental, and safety issues. The traditional mean to regulate occupancy (and thus lower traffic congestion) has been by adjusting parking prices (see Section 2.1). However, traditional parking prices cannot keep up with fluctuating demands. Additionally, traditional tariffs might instead have a negative impact on congestion. Therefore, many parties (such as PP, and operators) have started working towards a dynamic pricing model (see Section 2.1.2). Dynamic pricing should be able to maintain occupancy at an optimal level. Therefore, lowering the overall amount of traffic congestions and decrease its negative impact on the environment.

However, dynamic pricing also has negative sides, most of which can be considered unethical. Utilizing dynamic pricing, enables PPs to adapt their parking prices according to current conditions (see Section 2.1.2). However, such pricing model could be abused, by announcing false demand conditions. Therefore, creating unnecessarily high parking prices, either to increase their own revenues or to provide competitors with false demand conditions. This is unethical as it would create an unfair market in the parking industry, while in the end it mostly affects the drivers negatively.

Occupancy data plays a major role in dynamic pricing. Smartphones are one of the tools that can be used to collect occupancy data (see Section 2.2.1, paragraph 2.2.1.3). However, operators might also get access to location data and other sensitive data. Therefore, it is important that companies not abuse the permissions they might gain. Additionally, apart from being unethical, collecting excessive information without the users’ permission or knowledge, is illegal in Europe (since GDPR came into effect). Like the unethical pricing issue, mentioned above, the drivers are those who in the end are negatively affected the most by this type of abuse.

6.4 Future work and challenges

At the end of the project, there is naturally some work that has been forgotten or omitted because of limited time. Additionally, apart from future work, there are some challenges towards making the protocol a standard. All the planned work was not completed, such as the full support for location and occupancy data in the protocol. However, the elements that might be useful, for location and occupancy data, are included in the protocol’s format. Therefore, one of the next steps is to conduct research regarding the location and occupancy data. Additionally, some standards and definition of location and occupancy data element should be made and then added to the protocol. Another step required is to provide the protocol with a suitable name, as mentioned in Section 6.2. Some challenges with the project will be presented in Section 6.4.1. Some more identified future work will be presented in Section 6.4.2.
6.4.1 Challenges

The biggest challenge left is to have the parking industry adopt the protocol. The first target is to get adoption by commercial parking providers, as they usually have less strict rules and regulations than municipal operators. The more challenging part is to reach out to governmental providers such as cities, who may have strict rules and regulations that would affect what standards they can adopt.

Another challenge is to make the tariff rates readable by a human. The protocol’s format is comprehensive and covers a wide variety of tariffs. However, from a drivers’ perspective, a complex tariff might be hard to understand and worse, confusing. Therefore, one future task is to develop a method to parse and make tariffs more readable and understandable by humans. For example, a simple tariff can easily be described as a single value per hour, such as 10 SEK per hour. However, an irregular and complex tariff, that changes every minute, cannot be simply expressed as a value over a time period. An approach for such complex tariffs could be to display the expected fee to be charged, for a span of time that the driver chooses.

6.4.2 Future work

A future work that was thought about early but omitted, is the development of a system utilizing dynamic pricing. Such a system would take the responsibility of calculating parking tariffs and sharing parking-related data. However, two different approaches to implementation have been identified: a centralized or decentralized approach.

With a centralized hub, the full responsibility for calculating and distributing tariffs are placed on a single actor (such as a PP). This would mean that this PP would have full control over pricing. However, a major flaw would be the introduction of a single point of failure. If this PP’s systems go down, then all connected operators and parking systems would no longer be able to acquire tariff data.

With a decentralized system, the responsibility can be split between multiple actors (such as PP and operators). This would mean that the PP can share the responsibility of sharing tariffs with other parties, such as operators. However, by allowing a leader, the PP may still maintain control over pricing, while the single point of failure is removed. As a result, even if the PP’s system is down, connected operators and parking systems can still share the last known tariff data, thus minimizing the loss of functionality. Additionally, this approach is thought to be the better approach.

Another future work is to conduct research on how often prices should be changed in a dynamic priced environment. As mentioned in Section 2.1.2 (page 6), if prices change too frequently, these parking fees might annoy and confuse drivers. However, if the prices do no change frequent enough, then the benefits of dynamic pricing might not be achieved. Additionally, there might be an interest in providing individual pricing, which would add to the complexity of calculating parking fees.
References


References


Appendix A: GraphQL schema

Table 34: GraphQL symbol definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>A list of elements A. A may have 0 to many occurrences</td>
</tr>
<tr>
<td>A!</td>
<td>Element A is mandatory and not nullable</td>
</tr>
<tr>
<td>type A</td>
<td>Object definition of the element A</td>
</tr>
<tr>
<td>type Query</td>
<td>Query definition, describing the available queries</td>
</tr>
<tr>
<td>String, Int, Boolean, Float</td>
<td>Default predefined types</td>
</tr>
<tr>
<td>ID</td>
<td>Represents a unique identifier, basically a string</td>
</tr>
<tr>
<td>enum A</td>
<td>A is an element type that is limited to a predefined set of values.</td>
</tr>
<tr>
<td>scalar A</td>
<td>A is an open element type that is defined by the implementation</td>
</tr>
<tr>
<td># aaa</td>
<td>The following string is a comment</td>
</tr>
</tbody>
</table>

```graphql

type Query{
  location(locationId: [ID], areaNumber:ID, name: String, country: String, city: String, geoLocation: GeoLocation): [Location]
    occupancy(locationId: [ID]): [Occupancy]
    tariff(locationId: [ID], valid: Boolean): [Tariff]
}

# Location related objects

type Location {
  locationId: ID!
  parentLocation: ID
  sublocations: [ID]
  name: String
  areaNumber: ID
  address: Address!
  contact: Contact!
  geoLocation: [GeoLocation]
  advancedLocation: String
  auxiliary: Auxiliary
  log: Log!
}

type Address {
  number: String
  street: String!
  postalCode: String!
  city: String!
  country: String! # ISO-3166 Alpha-2
}
```
# GraphQL schema

```graphql
// Contact type

type Contact {
  contactName: String!
  organization: String!
  email: String!
  phoneNumber: String
}

// GeoLocation type

type GeoLocation {
  latitude: String!
  longitude: String!
  geoType: GeoType!
  locationName: String
}

// Auxiliary type

type Auxiliary {
  currency: Currency
  timeZone: String
  public: Boolean
  paid: Boolean
  locationType: LocationType
  operatingHours: [Schedule]
  surcharges: Surcharges
}

// Surcharges type

type Surcharges {
  taxIncluded: Boolean
  tax: Float
  other: Float
}

# Occupancy related objects

type Occupancy {
  locationId: ID!
  supply: Int!
  occupied: Int!
  occupiedPct: Int
  average: Int
  updateFrequency: Int
  detectionMethod: DetectionType
  parkingSpace: [ParkingSpace]
  log: Log!
}

// ParkingSpace type

type ParkingSpace {
  spaceId: ID!
  spaceNumber: Int
  available: Boolean
  frequency: Int
  occupiedFrom: DateTime
  occupiedTo: DateTime
  spaceType: SpaceType
  updated: DateTime
}
```
# Tariff related objects

type Tariff {
  tariffId: ID!
  locationId: ID!
  supersedes: String
  restrictions: Restrictions!
  rates: [Rate]!
  activeSchedules: [activeSchedule]
  validSchedules: [validSchedule]!
  log: Log!
}

type Restrictions {
  tariffType: TariffType
  rateType: RateType
  maxFee: Float
  minFee: Float
  maxPaidParkingTime: Boolean
  maxParkingTime: Int
  prepaid: Boolean
  resetTime: Int
  targetGroup: [ParkingType]
  vehicles: [VehicleType]
}

type Rate {
  order: Int!
  value: Float!
  interval: Int!
  intervals: Int
  unit: UnitType
  repeat: Boolean
  maxFee: Boolean
  countOnlyPaidTime: Boolean
}

type validSchedule {
  ID: ID!
  validFrom: DateTime!
  validTo: DateTime!
  startTime: Int
  endTime: Int
  days: [Days]
}

type activeSchedule {
  ID: ID!
  startTime: Int
  endTime: Int
  days: [Days]
}

# Log object used by location, occupancy, and tariff.
type Log {
  updated: DateTime!
  user: String!
  created: DateTime!
  creator: String!
}
enum GeoType {
    ENTRY
    EXIT
    ACCESS
    UI
    POINT
    OTHER
}
enum LocationType {
    ABOVE_GROUND_GARAGE
    UNDERGROUND_GARAGE
    ON_STREET
    SURFACE_LOT
    PRIVATE
    OTHER
}
enum DetectionType {
    MANUALLY
    SPACE_SENSOR
    VIDEO_COUNT
    IMAGE_ANALYTIC
    VIDEO_ANALYTIC
    CROWD_SOURCING
}
enum SpaceType {
    RESERVED
    UNRESERVED
    EVENT
    RESERVATION
    VALET
    OTHER
}
enum Days {
    MONDAY
    TUESDAY
    WEDNESDAY
    THURSDAY
    FRIDAY
    SATURDAY
    SUNDAY
    DAY_BEFORE_RED_DAY
    HOLIDAY
}
enum TariffType {
    REGULAR
    FIXED
    EVENT
    CONTRACT
    INDIVIDUAL
}
enum Currency {  # ISO 4217 currency codes
    AUD
    EUR
    USD
    GBP
    SEK
    NOK
    ...
}

enum ParkingType {
    PRIVATE
    PERSONNEL
    RESIDENTIAL
    INDIVIDUAL
    PUBLIC
}

enum VehicleType {
    CAR
    VAN
    MC
    TRUCK
    ELECTRIC
    BUS
}

enum UnitType{
    MINUTES
    DAY
    WEEK
    MONTH
    YEAR
}

scalar DateTime # ISO date time (YYYY-MM-DDThh:mm:ss±hh:mm)
scalar Date # ISO date (YYYY-MM-DD)