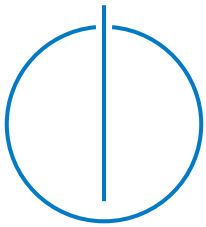
Department of Informatics
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Design and Implementation of a Bikesharing Service as part of an open Mobility-Ecosystem

Lucas Weidner



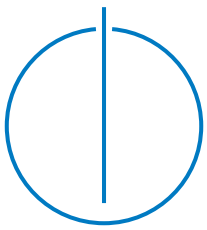
Department of Informatics
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Design and Implementation of a Bikesharing Service as part of an open Mobility-Ecosystem

Entwurf und Realisierung eines Bikesharing Dienstes als Bestandteil eines offenen Mobilitätsökosystems

Author:	Lucas Weidner
Supervisor:	Prof. Dr. rer.nat. Florian Matthes
Advisor:	Felix Michel
Submission Date:	15.11.2016



I assure the single handed composition of this Master's thesis only supported by declared resources.

Lucas Weidner

Acknowledgement

First and foremost, I would like to thank my supervisor Prof. Dr. rer.nat. Florian Matthes and the Technical University Munich for the opportunity to write this thesis.

I would also like to show my greatest appreciation to my advisor Felix Michel. He gave me great help and I learned a lot from him. His encouraging words, the discussions and all the comments on my thesis have been very beneficial.

Also a special thanks to the *iteratec GmbH*, specifically to Daniel Stahr and Anton Brass. I learned so much in the past year and I am happy that I could write my thesis together with you.

I am also grateful to my friends Marleen Vetter and Hammad Khan for their efforts in reading through this thesis.

Last but not least, I would like to thank my family. Especially my parents Astrid and Matthias Weidner and my sister Theresa for their endless love and support. They keep me grounded and I am blessed to have them.

Abstract

There is an influx of various sharing services in the market at the moment. Yet none of them have a general interface which can be used by various platforms. Each customer therefore ends up with many different applications on their smartphone just for traveling alone, examples being: public transport, carsharing, bikesharing etc. An open mobility ecosystem would be an ideal solution, which would allow the integrations of all these services in one application. This one application would be able to calculate the routing and combine other factors depending on the different means of transport and allowing the user to purchase tickets etc.
This thesis not only shed light on the proposed system, but also focuses on defining a generic REST interface and implementing it on a bikesharing service called Sharelock from the iteratec GmbH. This interface provides all the necessary gateways and functions that an open mobility ecosystem needs to integrate the service. Furthermore, the different needs and possibilities for locks which are online or offline are also characterized, since they each have their own advantages and disadvantages. Additionally, this thesis also covers the development of the backend as well as the administration frontend to operate the bikesharing and the client application (Android) to use it.
Finally, Sharelock was evaluated by working students from the iteratec GmbH and scored a high usability score. However, not all functionalities could be tested due to the lock technology still being under development. Nevertheless, this thesis shows that it is possible to build a system for an open mobility ecosystem.

Zusammenfassung

Immer mehr Sharing-Dienste kommen aktuell auf den Markt. Allerdings hat keiner dieser Dienste ein Interface, dass von mehreren Plattformen benutzt werden kann. Somit müssen Nutzer weiterhin für jeden Service eine eigene App auf dem Smartphone installieren. Allein zur Routenfindung innerhalb von Städten zum Beispiel Applikationen für die öffentlichen Verkehrsmittel, Car-Sharing und Bike-Sharing. Ein offenes Mobilitätsökosystem wäre die ideale Lösung um alle Dienste zu kombinieren. Der Nutzer könnte dann alle Services aus einer App benutzen. Diese würde das Routing, die Navigation und den Kauf der Services beinhalten.
Diese Arbeit ist allerdings nicht auf das beschriebene System fokussiert. Es geht vielmehr um die Definition eines generischen REST-Interfaces und die Implementierung dessen am Beispiel des Firmen-Bike-Sharing Sharelock der iteratec GmbH. Dieses Interface stellt alle benötigten Funktionen zur Verfügung, die ein offenes Mobilitätsökosystem benötigt, um den Service zu integrieren. Außerdem werden die verschiedenen Notwendigkeiten und Möglichkeiten von Schlössern die offline und online sind charakterisiert, da jedes seine eigenen Vor- und Nachteile besitzt. Zusätzlich behandelt diese Arbeit die Entwicklung eines Servers mit zugehörigem Administrations-Bereich zum Betreiben des Bike-Sharing-Dienstes. Ebenfalls wurde eine Android-Applikation zur Benutzung des Service entwickelt.
Zum Schluss wurde Sharelock von Werkstudenten der iteratec GmbH evaluiert und mit einer hohen Benutzerfreundlichkeit bewertet. Allerdings konnten nicht alle Funktionalitäten getestet werden, da die Entwicklung des Schlosses noch nicht beendet ist. Nichtsdestotrotz zeigt diese Arbeit, dass es möglich ist ein System für ein offenes Mobilitätsökosystem zu entwickeln.

Contents

List of Figures

List of Tables

1. Introduction

Sharing platforms get more and more popular every year. Recent companies like DriveNow, Car2Go or Flinkster are typical examples for carsharing in Germany. However, there are also other means of transportation which are proficient for sharing services.
A growth in one of these kinds of sharing systems can be seen for a few years now. As city councils are searching for ways to decrease air pollution and traffic jams inside cities bikesharing becomes more and more interesting for them. By investing a comparable small amount of money, they can influence different factors dramatically. That is why more and more cities discover the big advantages of bikesharing for themselves.
At the moment, there are approximately 1,328,100 bicycles and pedelecs in use for bikesharing in 1,029 cities around the world. 319 cities are in the planning phase or under construction of a bikesharing system[1]. There is a blog [Bib] which documents everything related to bikesharing at the moment. Additionally, there is a world map which shows bikesharing cities [Bic] globally.
Bikesharing is a comparatively new business and therefore under heavy development. Established companies went bankrupt and new ones are entering the field with new ideas on how to solve different problems.
There are several challenges with bikesharing. One of the main problems is the distribution of bikes. In other terms, bikes will agglomerate in some places whereas in other areas, no bikes are available. As this limits the availability of bikes for customers in certain areas, it reduces the income of the sharing provider. Additionally, costumer satisfaction is reduced. However, this thesis is not about the solution of this problem. References for this issue are discussed by Raviv, Tzur and Forma [TR11] and Contardo, Morency and Rousseau [CC12].

[1] According to `https://www.google.com/maps/d/viewer?mid=1UxYw9YrwT_R3SGsktJU3D-2GpMU&hl=en` from 18/05/2016

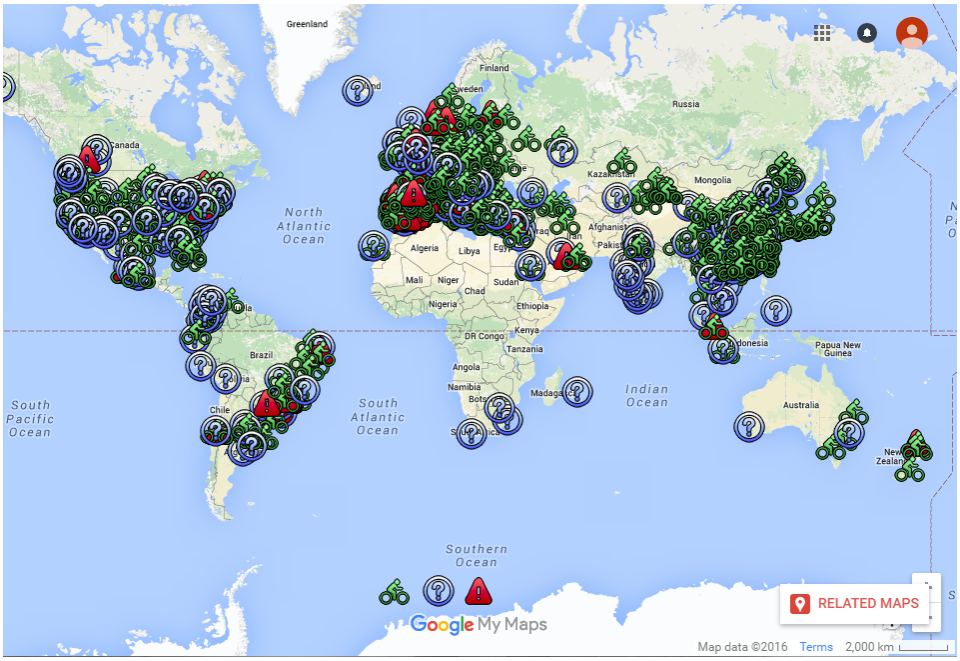


Figure 1.1.: Bikesharing provider on the world

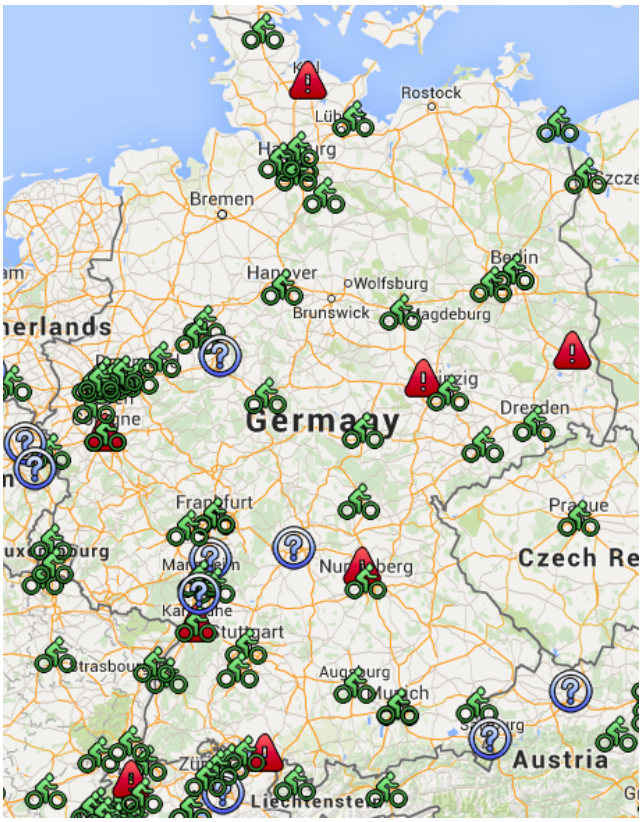


Figure 1.2.: Bikesharing provider in Germany

1.1. Challenges of an open mobility ecosystem

Nowadays, everyone has several applications on the smartphone. Some of these applications include navigation apps, renting cars or bikes or ticketing apps for public transport. For security reasons, each of these services should ideally have another login. At minimum, each service should have a different password to protect the customers from privacy or financial fraud. Yet, this means the user has to remember all this data, which is potentially a difficult task.
Another issue is switching between several applications. Using different applications parallel can be dangerous while riding a bike.
Having all these features in only one application would decrease the complexity of the usage enormously. In addition, it increases the user-friendliness and security while riding.
Therefore, the interfaces for all the different services have to be almost generic to implement them in an open mobility ecosystem. Users could then search for different routings, buy tickets or rent means of transportation. In the perfect case, the user just has one username and password to use all these services without being concerned about the digital security. Additionally, the billing would be integrated so that the user only has to enter their data once.

1.2. Motivation

Iteratec is a software development company near Munich. One of their internal projects is “Sharelock”. Sharelock is an electrical bike lock, which can be opened and closed via Bluetooth. Actually, it is in an early stage of development. However, when it is ready for operation the company will provide bikes to the employees with this lock for short rides. The company will behave like a bikesharing provider but with other goals then typical bikesharing companies. The objective is not to make money but to increase the health of employees and to decrease the time for typical or daily paths and commutes.
Even though Sharelock’s bikesharing will be used only within the company, it’s aim to build a sharing software to use for different means can easily be integraed into an open mobility ecosystem as it is designed as a generic sharing platform.

That the use of bicycles for the journey to work is increasing shows an article [Co] about company bicycles in Germany. More and more people are leasing bikes over the company. Since end of 2012, having a company bicycle does not interfere with the ownership of a company car. Furthermore, employees can even lease two bikes, for example like a normal one and an electric bicycle. Company bikes have several advantages for the employees and in fact also for the company. It is much more healthier and not as stressful as driving a car in the rush hour. In addition, the employees can use the bike for private reasons. Some companies even let their employees choose bikes they want. The company buys the bike for them and just decides about the painting, like the online language school Babbel.

1.3. Structure of this thesis

This document starts with a short introduction. It describes the challenges of an open mobility ecosystem and the motivation for myself to write this thesis. The next chapter is about related work. It outlines the types of sharing provider and gives several examples for sharing provider mainly in Germany. It also shows projects related to bikesharing and and the workflow from the bikesharing system in Munich. At the end are some examples for technical systems which are in some relation to bikesharing.
Subsequently is a chapter about the conceptual approach where the basic architecture is described. In the beginning is an evaluation for the lock technology. Then follows a list of basic use cases which are important for the Sharelock system. Last is a sketch of the developed workflow with the different interfaces.
In the fourth chapter is the description of the technical approach. It covers in depth the internal workflow of the Sharelock system. It also shows how the messages are sent from device to device and what these messages contain. In addition, the structure of the server software and of the client software are visualized. The developed REST interface for an open mobility ecosystem is displayed at the end.
Chapter 5 covers the evaluation of the system. Sharelock was tested by several working students at iteratec. Based on their feedback, the SUS (System Usability Score) for the frontend and for the smartphone application was calculated.

The discussion and outlook is in the last chapter. It starts with general observations. Last, a summary of further possibilities for improvement of the system is described.

2. Related Work

This chapter gives a short introduction to bikesharing. In the beginning is a small summary about the history and then a description of the different types. A list of various bikesharing provider with a comparison of them is in the next section followed by projects related to bikesharing. Afterwards, an example of the workflow of a bikesharing provider will be given. At the end is a list of related technical systems from hardware to software.

2.1. Types of Sharing Provider

There are several options to distinguish sharing providers. The first one is by assigning a generation. Following is a short summary of the different generations of sharing providers in history which are outlined in [De09].

Generations of bikesharing

There are four diffenent generations of bikesharing considered in literature. The first one was in Amsterdam in 1965. It was called "Witte Fietsen". This means White Bikes, because they were all white painted. All the bikes were placed in Amsterdam and everyone could use them for free. No locks or stations were needed. However, people did not use them appropriate, for example they stole or damaged it.
To reduce these problems, the second generation of bikesharing was introduced in 1991 in Farsø and Grenå, Denmark [De09]. This time, the bikes were more robust to resist the hard use. The system was station based with a coin deposit, through still free to use. Unfortunately, there was also ongoing theft.
Due to the anonymity of the users, there was a high theft and damage rate. That is why the third generation improved the tracking of the customer [De09] with different types of authorization methods. The first one of these was Bikeabout (1996 at Portsmouth University in England). The authorization was

done with magnetic stripe cards. Subsequently, different technologies were used like electronically locking racks, smartcards, mobile phone access and others.
Starting with this inventions, bikesharing systems became more and more popular all over Europe and the world. Currently, bikesharing is in its fourth generation with different additional functionalities like tracking of bike positions, mileages and further improvements.

Business Models

A second option to distinguish sharing providers is by differentiation of the business models or models of provision as in [De09]. First is the obvious one. Generating money by renting the bikes. There are various forms like renting per minute, per day or with special rates with free usage volumes.
Advertising companies like JCDecaux (see section 2.2.3) apply the second type. These companies typically own displays on billboards, bus shelters or other public spaces where they display their advertisements. Cities or regions who cannot run a bikesharing service themselves can provide public spaces to advertising companies in exchange for the service.
Another possibility is typically for regions which own more money. This means that they have enough money to afford building a system alone or with a partner. They can run the service alone and therefore have full control over the bikesharing. Nowadays, this model is rarely seen, because it is to expensive and hard to build. However, one example for a city which builds the system alone is BiCiBUR in Spain (see section 2.2.13). MVG Rad is an example for bikesharing with a partner. One partner is the MVG (public transport in Munich). It is partly owned by the city and includes the bikesharing in its ecosystem. Bikesharing provider nextbike is the other partner which brings the bikes and the know-how.
The next type is the transport agency model, which was done for example in Germany. Deutsche Bahn (German Railway) is a quasi-governmental organization with a bikesharing subsidiary called “Call A Bike” (see section 2.2.4).

2.2. Sharing Provider

In this section is a small selection of different bikesharing provider described. At the beginning are a few provider which are also retailers. Following, bikesharing providers from German cities are discussed. Last are different bikesharing models with other business models, other kinds of clients or from other countries. These models are not mainly focused on profit and have other main focuses such as the increased health of its users.

2.2.1. Open Source Bike Share

"Open Source Bike Share" [Opa] is a project which provides software for providing station-based bikesharing. This system is cheap and easy to install which makes it very interesting. It does not need any special equipment or special bikes. Any bike can be used and all that is needed are combination locks.
A customer who wants to use this system chooses a bike online. Then a message with the PIN code is sent to the user. Now, the customer just has to open the lock with the PIN code and can use the bike. At the end of the usage, the client gets another message with a new PIN code. Now, the user just has to change the combination lock to the new PIN code.
However, Open Source Bike Share has some disadvantages. Exact usage times cannot be tracked because the lock cannot send any message. Therefore, lock operations like opening and closing cannot be tracked. So the system cannot bill the usage correctly as a user can still use the bike after the change of the lock code. Additionally, the position of the bike cannot be sensed as there is no location tracker available at the lock or the bike. Trusting the customer is essential.
Open Source Bike Share is free to use, which means everyone can download the source code from GitHub and use it. Only a standard server which can run PHP and MySQL is needed.

2.2.2. Nextbike

A company which provides all necessary equipment and software solutions is "nextbike" [ne]. Founded in 2004, nextbike became one of the leading manufacturers and operators[2]. They have developed a bike with integrated technologies for the sharing purpose. Additionally, they have different kinds of return stations with or without screens or with solar battery operation for all urban situations. They provide the solution to possible partners who want to run a bikesharing system. These systems can be for different purposes like normal bikesharing, but also for campus, business or hotel bikes, for advertising or for events. Furthermore, they are not limited to these functions, but they also build tailor-made solutions for different types of customers. Usually, each customer who uses nextbike can rent up to four bikes at the same time. At present, they have 30,000 bikes in 18 countries on 4 continents. They are present in over 100 cities and are the largest international bikesharing network[3].

2.2.3. JCDecaux

Yet another operator is JCDecaux [JC] from France. JCDecaux is one of the largest billboard advertising and street furniture design firms[4] in the world with nearly 2 billion Euro revenue in 2009[5].
JCDecaux is an example for the advertising business model. Cities provide the company space where they can build their stations for free or cheap. These stations and the bikes are painted with adverts. That means JCDecaux earns money from the companies.
Cyclocity [Bid] is the bikesharing subsidiary of JCDecaux. It started by operating the bikesharing system Vélo'V from Lyon (France). JCDecaux received the sole contract to use the transit shelters for advertisements in Lyon. In return, Cyclocity operates the bikesharing system.
Vélo'V is integrated into the public transport system of Lyon. Customer need a smart card which they swipe at the racks of a station to unlock a bike. Returning the bike is done by pushing the bike in the next free rack.

[2] http://www.nextbike.net/products/
[3] http://www.nextbike.net/about/
[4] http://www.jcdecaux.com/en/Outdoor-Advertising/Billboard
[5] https://de.statista.com/statistik/daten/studie/163549/umfrage/umsatz-der-fuehrenden-aussenwerber-weltweit-in-2009/

Using Vélo'V is very cheap. For 1 Euro per week or 5 Euro per year a customer can buy a smart card. Then, the first 30 minutes of each ride are free. If someone rides the bike longer, it costs an additional 0.50 Euro per half an hour. However, commuters can upgrade their transit pass to use Vélo'V for up to one hour for free.
Bikesharing was used extensively in Lyon. A bike was used by an average of 15 people per day with an average journey of 17 minutes[6].

2.2.4. Call A Bike

Another operator of bikesharing system is "Call A Bike" [Ca]. It is a subsidiary from the Deutsche Bahn company (German Railway). "Call A Bike" provides bicycles in many cities in Germany. After registration on the platform, bikes can be rent by calling a bike specific phone number. The caller receives a PIN code for opening the bike. Call A Bike is station-based in most cities but in a few there is also a flexible return zone. The price scheme starts at 1 Euro for half an hour with additional monthly or yearly rates to reduce the costs.
Compared to the bikesharing from JCDecaux, it costs more due to the fact that Call A Bike does not receive comparable benefits from the city.

2.2.5. StadtRAD Hamburg

StadtRAD Hamburg [St] uses the "Call A Bike" system and is only station-based. However, without paying a monthly or yearly fee, the bike can be used for free for the first half an hour after registration. The only fee is 5.00 Euro for registration which will go into a virtual account. Each minute you ride longer than the half an hour will be paid from the virtual account. For authentication purposes, the customer needs a credit or debit card. For a small fee, a key card can be purchased to make the booking even simpler.

2.2.6. Konrad

Next provider is "Konrad" [Ko]. It is the bikesharing system from Kassel/Germany. Customers, who are registered at "Call A Bike" or Stadtrad-Hamburg

[6] `http://www.bikesharephiladelphia.org/learn/history/`

do not have to register, everyone else has to register for the service when first using it. Afterwards, everyone can open one of the 500 provided bikes by calling a bike-specific phone number which is displayed on the lock. Returning the bike is as easy as closing the lock. The only important thing is that retrieving and returning can only be done at one of 56 special bike stations in the city. Students with a "Semester Ticket" can use the bikes one hour a day for free. The normal price is 1 Euro per hour.

2.2.7. metropolradruhr

"metropolradruhr" [meb] uses the nextbike system and provides the bike in the whole Ruhr region. Customers, who are already registered at nextbike do not have to register at metropolradruhr. It has similar prices as nextbike. Students from specific universities in the Ruhr region can use metropolradruhr the first hour every day for free. metropolradruhr is very attractive as it covers a big region.

2.2.8. NorisBike

Nextbike is also operator for NorisBike [No] in Nuremburg. It has same conditions as nextbike operated cities. NorisBike has over 800 bikes at over 70 stations. Inside the inner city, bikes can be returned everywhere in public (flexible zone).

2.2.9. MVG Rad

MVG Rad [MVa] is the bikesharing service from Munich. It is also operated by nextbike and has 1200 bikes and 125 stations. However, the bikes can be returned at special return stations or everywhere in the flexible return zone[7]. A customer receives a bonus of 10 free minutes on the account if the bike is returned to a station.

[7] `http://www.nextbike.de/en/news/`

2.2.10. Fächerrad

Another bikesharing system operated by nextbike is Fächerrad [Frad] from Karlsruhe. It started in May 2014 and has around 330 bicycles. Since June 2015, 16 pedelecs are also part of the system[8]. Two extra stations with charging possibilities where added too. Return options are similar to the MVG Rad system. There is a flexible return zone inside the city and additional return stations inside and around this zone. The pricing is also the same. Cheaper prices apply to students, customer from the public transport or local carsharing user.

2.2.11. MVGmeinRad

MVGmeinRad [MVb] is another station-based bikesharing service. It is in Mainz and a chip card is needed for the renting process. This chip card is necessary to open the lock at a station. There are over 100 stations with up to 1000 bicycles. The prices are a higher than nextbike prices, but with different rates.
Currently, MVGmeinRad is considering adding cargo bikes to the fleet. Customer experience is always tried to be increased and customer have been surveyed to improve the service.

2.2.12. Chemnitzer Stadtfahrrad

An additional concept is made by "Chemnitzer Stadtfahrrad" [Ch]. The station-based project from the city Chemnitz is mostly analog compared to all the other bikesharing services. Customers go to one of ten bike stations and show their identity card. For a small fee of 2 Euro per day or 20 Euro per month, they can use the bike for the respective time. The bikes are provided by different local shops and companies with special advertisement for themselves, therefore the costs for the project are very low which means that the small fee is more a representation allowance.

[8] http://www.karlsruhe-macht-klima.de/klimaschutzvorort/mobilitaet/faecherrad.de

2.2.13. BiCiBUR

BiCiBUR is the bikesharing system from Burgos [Bia]. It was started in 2006 with a free so-called "Loan system" as part of the European CiViTAS [Cib] project[9]. Citizens could hire a bike for two hours and tourists for three hours. There was also a lock that tourists could visit attractions.
BiCiBUR is station-based (23 stations) with a contactless card to open the bicycles. The availability can be checked in real-time over the internet.
In 2011, the system was joined with the local public transport to enable better access to the bikes. This also increased the number of customers (ca. 12.000 users with about 150.000 rides) [Bub].
One year later, the city council introduced a subscription fee of 15 Euro per year which dramatically decreased the number of customers and uses (ca. 500 user with around 5.000 rides). The new registration system was seen as too complicated on the one hand and too expensive compared to the quality of the bike (price of a bike was about 50 Euro) on the other hand. The new system is also a reason that tourists cannot use it anymore because of language barrier and the complexity.
BiCiBUR is part of the VeloCittà project [Bua] which will be described later in section 2.3.

2.2.14. Melbourne Bike Share

Melbourne Bike Share [Mea] is an example for a foreign bikesharing service. It is station based with about 50 stations and 600 bikes around Melbourne. Customers can buy a daily, weekly or annual pass with an unlimited number of 30 min rides (45 min for the annual pass) included. If the customer exceeds this time, additional fees depending on the overtime apply. Registering is not needed as the credit card hold for authorization. Every bike comes with a helmet for safety reasons. Furthermore, subscribers of an annual pass can purchase a personal helmet for just 5 $. If customers want to improve their bicycle skills, Melbourne Bike Share offers discounted courses from a licensed partner.

[9] CiViTAS (from City, Vitality and Sustainability) aims to create cleaner, better transport in cities

Figure 2.1.: Image of a station from Melbourne Bike Share

2.2.15. CERN

The CERN is a representative of bikesharing inside organizations and companies. It is located in switzerland and provides bikesharing and bike rental to the employees[10].

Bikesharing

CERN employees can borrow a Velopass bicycles for work-related purposes for free. Therefore, they just have to receive a Velopass subscription card [CEb], which is valid for one year. Opening and closing of the locks is done with the subscription card at the bike stations.

[10]Source: http://smb-dep.web.cern.ch/sites/smb-dep.web.cern.ch/files/documents/CarPool/CERN_Mobility_NewProcedures.pdf

Bike Rental

There is also the possibility to rent bikes for a longer period of time for 1 CHF per day [CEa]. This is applicable for members of the personnel in specific periods and only up to three months. Summer students at the CERN can borrow the bikes for free for the same period of time.

2.2.16. Google

Google provides bicycles on their campus in Mountain View for free [Go]. Not only employees can use them but any visitor. It started in 2007 with just 100 bikes[11]. In 2009, Google provided multi-colored bikes. Nowadays, the bikes are designed by engineers from Google.
They get maintained by a small division inside Google. The company does not make any money out of them, but keeps the employees happy and healthy. There is also the possibility to use special bikes with more seats for meetings. The advantage for Google is that it decreases the unproductive time of the employees while increasing mood.

2.2.17. Cargo Bikesharing

At the moment, bikesharing was just meant to provide normal bicycles for customers for the last mile from public transportation or at least for short time use. Actually, there is also a second possible use case for another type of bicycles. Cargo bikes are designed to carry loads[12]. Figure 2.2 shows an example for such a bike. As such bikes are more expensive than normal bikes and they are not in constant use for a specific person, it makes sense to provide cargo bikes also via bikesharing. Following is a short list of cities and providers in Europe which have cargo bikes. Konstanz, Norderstedhe (both Germany), the Lastenradkollektif in Vienna and Kasimir in Cologne, Hereford, London Bike Hub in Ealing, Cambridge, Norwich and Carvelo2go in Bern.

[11] http://www.businessinsider.com/here-are-the-crazy-colorful-bikes-google-employees-ride-around-campus-hq-2013-7?IR=T

[12] Source: http://www.shareable.net/blog/8-cargo-bike-sharing-programs-in-europe

Freie Lastenradler

Freie Lastenradler is an example for a provider of cargo bikes in Munich. At several stations in Munich, a cargo bike can be rented for up to three days for free. Customers only have to register on their website and book the bicycle for the preferred dates and the preferred station in advance. An image of such a cargo bike in Munich is figure 2.2.

Figure 2.2.: Cargo Bike from "Freie Lastenradler"

2.2.18. Comparison Sharing Provider

This section compares the several different sharing providers from above. The comparison is displayed in table 2.1. One feature to compare is the type of return. If it is just station-based or if it has a flexible return zone which makes bikesharing much more attractive. Second, the type of lock which is used. Some locks are more user-friendly than other locks. Third one is the business model together with the price for a usage of 30 minutes. This is the normal usage time for bikesharing and therefore the important factor. The last is the type of clients which use the service. Is it possible for a tourist to use it. Some are limiting the service to specific groups.
That Sharelock is a unique bikesharing service can be reasoned from table 2.1. It is the only service where the bikes can be opened via application. Another advantage is that bikes can be returned in a flexible zone. The pricing is not determined right now and can be customized. In the beginning, it starts only with the employees of iteratec. Later, it can be used for several other use cases.

	Station-based or flexible zone	Locks	Business Model (price 30 min)	Clients
Open Source Bike Share (B2B & B2C)	station-based	4 digit PIN locks	customizable	private users
Nextbike (B2B & B2C)	both possible	different types (app, smartcard or login at onboard computer)	customizable (1 Euro)	different types (private, events, hotels, business,...)
JCDecaux (Cyclocity) (B2B & B2C)	station-based	opening with membership card	customizable (free with 7 day ticket)	private users
Call A Bike (B2B & B2C)	both possible	call a number and receive an unlock code	pay per time (1 Euro)	private and train users
StadtRAD Hamburg (B2C)	station-based	call a number and receive an unlock code	pay per time (free)	tourists and locals
Konrad (B2C)	station-based	call a number and receive an unlock code	pay per time (1 Euro)	private users
metropolradruhr (B2C)	station-based	combination lock	pay per time (1 Euro)	tourists and locals
NorisBike (B2C)	both possible	combination lock	pay per time (1 Euro)	tourists and locals
MVG Rad (B2C)	both possible	on-board computer	pay per time (2,40 Euro)	tourists and locals
Fächerrad (B2C)	both possible	on-board computer	pay per time (1 Euro)	tourists and locals
MVGmeinRad (B2C)	station-based	opening with membership card on terminal	pay per time (1,40 Euro)	tourists and locals
Chemnitzer Stadtfahrrad (B2C)	station-based	opening on presentation of identity card	pay per day (2 Euro)	tourists and locals
BiCiBUR (B2C)	station-based	opening with membership card on terminal and entering PIN	flat rate for 15 Euro per year	locals
Melbourne Bike Share (B2C)	station-based	entering PIN code which you receive from terminal with your credit card	free	tourists and locals
CERN (B2C)	station-based	opening with membership card on bike station	free or 1 CHF per day between 01/06 and 30/09	employees
Google (B2C)	flexible zone	no locks	free	everyone
Cargo Bikesharing (Freie Lastenradler) (B2C)	station-based	opening on presentation of identity card and codeword	free	locals
Sharelock (B2C)	flexible zone	opening via app	pay per time (not stated)	iteratec employees

Table 2.1.: Comparison of different bikesharing provider against selected factors

2.3. Projects

There are several projects with the aim to improve and support bikesharing. These projects are not providers themselves. They are for example funded from the European Union with the goal to help existing provider.

One example for a project is VeloCittà [Ve]. It is a co-funded programm from the European Union with the aim to improve existing bicycle sharing systems. It runs from March 2014 until February 2017 and combines the five bikesharing system from Szeged (HU), Krakow (PL), Padua (IT), Burgos (SP) (see section 2.2.13) and London (UK).
Szeged's bikesharing is called CityBike[13]. It has a similar system to Call A Bike (see section 2.2.4) where the customer calls a phone number to receive a PIN code. CityBike is a station-based system.
Krakow had BikeOne for bikesharing. However, the contract ended in 2013 and a new system is under development. It will have a swipe card technology to unlock the bikes.
GOODBIKE PADOVA[14] is the bikesharing from Padua/Padova. It has several different subscription models and is station-based.
London's bikesharing, Santander Cycles[15], has over 11,000 bicycles at over 750 stations. It costs £2 per day with 30 min free per usage. Longer usages cost another £2 per 30 min. There is no need for registration as everything is done by credit or debit card.
VeloCittà wants to build an experience and knowledge base for the five systems, but also for others. The main goal is to adopt the most effective solutions for bikesharing.

[13] http://www.citybikeszeged.hu/en
[14] http://www.goodbikepadova.it/
[15] https://tfl.gov.uk/modes/cycling/santander-cycles

2.4. Best Practice workflow from existing bikesharing provider

Following sections explain the workflow of the existing bikesharing provider in Munich (Germany). First provider is MVG Rad. It is operated together with public services of Munich. Second provider is Call A Bike. It is the bikesharing subsidiary of Deutsche Bahn (German Railway).

2.4.1. MVG Rad

This section outlines the actual workflow of a booking from MVG Rad [MVa]. In the beginning, the customer receives a map with an overview of the locations of the bikes and the different stations (see fig. 2.4a). Now, the customer can choose a station or bike to receive further informations about the price or the availability of bikes (in case of a station) as you can see in fig. 2.4b. There are two possibilities if the user decides to use a bike. The first one is to reserve the bike for starting the ride in a few minutes (for example if the customer has to reach the bike first). Starting the usage immediately is the second one. The user receives the PIN code to open the lock (see fig. 2.4c) after the start of the usage. At the bottom, the app shows how long the user has the bike.
The usage ends when the customer parks the bike and locks it. He confirms the end at the local bike computer and waits until the application finishes the booking by emptying the view (fig. 2.4d). The customer can now see a history of all rides with their times and prices (fig. 2.4e).

2.4.2. Call A Bike

Using Call A Bike is a little bit different to the workflow of MVG Rad. At the beginning, the customer has to call the phone number which is displayed on each bike. Afterwards, the user receives a four digit PIN code for opening. This code has to be entered into the on-board computer (see figure 2.3). If it was successful, the lock opens and the bike is ready to ride.
Returning the bike is possible everywhere inside the flexible zone (at least in Munich). A customer just has to park the bike, push the locking button and the booking is finished.

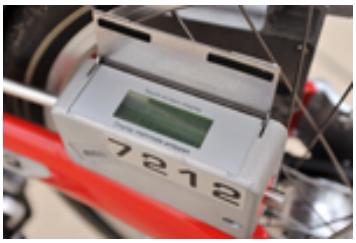


Figure 2.3.: Call A Bike on-board computer.[16]

2.5. Related technical systems

Sharelock is not only a digital solution. It also includes hardware parts like the physical lock. In the following sections, related technical systems from hardware to software are described. There are special bikes and locks. Furthermore, there is a project for an online database of bikesharing provider and bikes. At the end is a software kit for integration into a mobility system.

2.5.1. OpenBike

OpenBike [Opb] develops an operating system for bicycles. The goal is to build a bike with just one battery for all components. It should also have only one network to combine everything and one connection for access to apps. Therefore, the maintenance of the bike should be much easier. Overall, it will be easier to work with it for bicycle enthusiasts.
Right now, it is in a very early stage and no information about when the bikes will be released is available.

2.5.2. I LOCK IT

The bicycle lock "I LOCK IT" is build by haveltec, a startup company from Berlin. It is an electronical lock with GPS tracking, mounted to the frame of a bike. It opens and closes automatically when the smartphone of the owner arrives to the bicycle or leaves it. Furthermore, it is secured by an alarm system and sends a theft notifications, when someone wants to steal it. The communication to the lock is done via bluetooth signals. "I LOCK IT" can

[16]Image source: `https://www.callabike-interaktiv.de/index.php?id=397&f=500]`

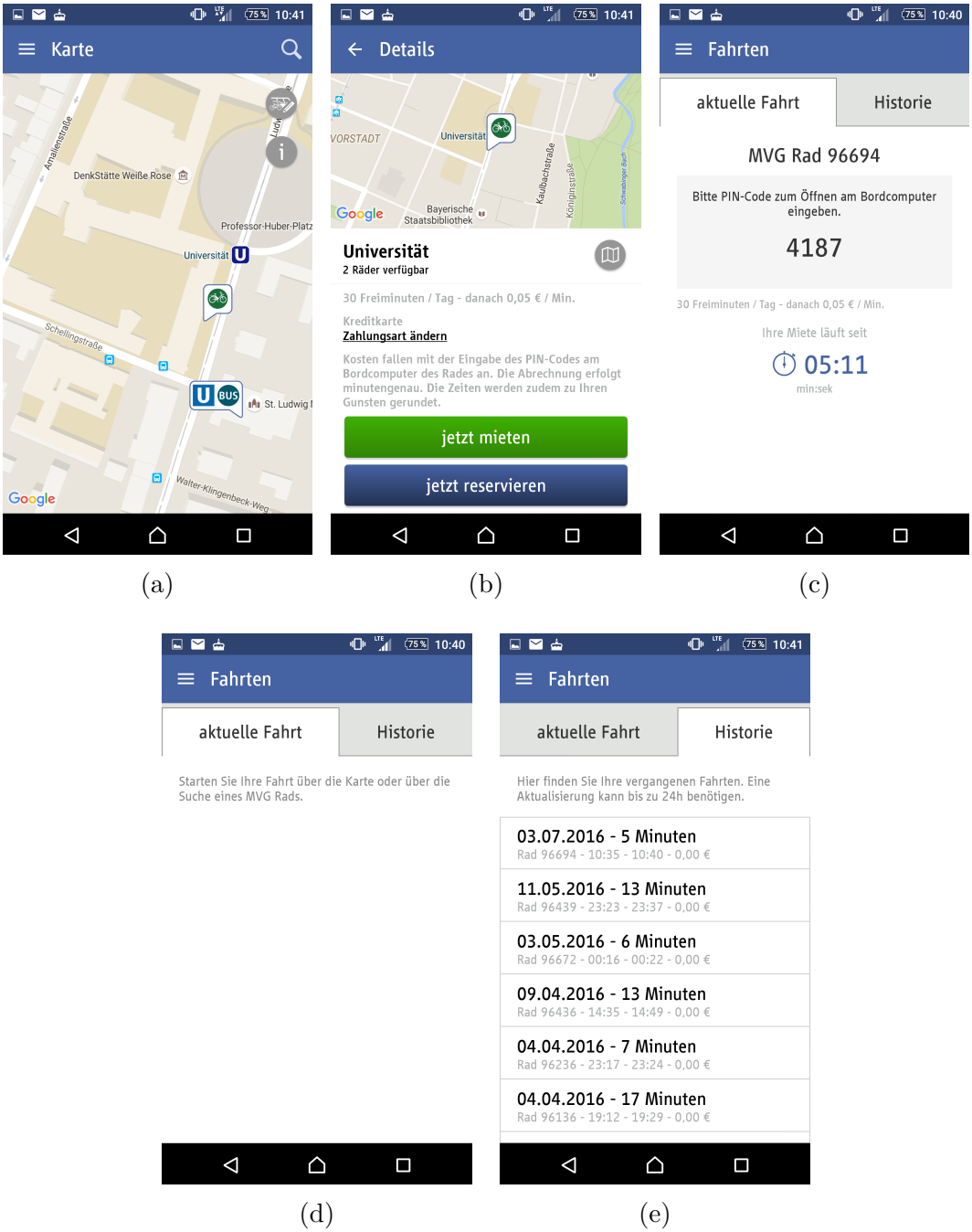


Figure 2.4.: Booking workflow from MVG Rad a) Map overview for bikes and stations. The customer has to click on a marker. b) Further information about bike or station like price or number of bikes. c) Usage of bike with PIN code and actual time of use. The PIN code has to be entered at the on-board computer of the bike. d) Overview over all actual rides. There are no rides after finishing a tour. e) Overview over all finished rides. This is the history.

also be used to share the bike with family and friends or for usual bikesharing by using the smartphone application.
Interestingly, "I LOCK IT" was successfully funded by a campaign on kickstarter[17].

2.5.3. BitLock

Another electronical bike lock is BitLock [Bie]. It also has a GPS tracker and communicates via bluetooth with the smartphone of the user. For opening and closing, the user just has to press a button on the lock when the smartphone is nearby. In comparison to "I LOOK IT", the main focus of BitLock lies on the track activity. Factors like trip length, duration and burnt calories are measured by the app.
BitLock is able to be used as a bikesharing system from the beginning. Sharing with friends and family does not cost any money, but you have to pay for extensive use. In return, you get a fleet management system for a bikesharing service.

2.5.4. CityBikes

CityBikes [Cia] is an API which provides data about the different bikesharing systems, their stations and the bikes. The received data are in JSON format and real-time. Therefore, you can easily integrate data about sharing systems into your own system or can display the location from bikes in specific cities.

2.5.5. RideTap

RideTap [Ri] is an SDK which combines APIs from different transportation services like public transportation, car and bikesharing or rideshare. It gives the possibility to integrate the functionality of finding a transportation solution inside an app immediatly. By adding a few lines of code, developer can implement RideTap into their app. The user of the app can find the right transportation which suits them with just one click.

[17]Kickstarter campaign: `https://www.kickstarter.com/projects/742922560/i-lock-it-the-worlds-first-fully-automatic-bike-lo/description`

3. Conceptual Approach

This chapter explains the different possibilities for the architecture of Sharelock (section 3.1). An overview about all the essential use-cases for bikesharing are described in section 3.2. Finally, a short description of how the workflow would look like is given.

3.1. Fundamental Architecture of the Service

When talking about the fundamental architecture, it is important to outline the different kinds which are possible for sharing services. They all differ mostly in their capability to connect to the internet. Many sharing services have all their objects directly connected to the internet. Nevertheless, sharing services with devices which are offline are still possible. Each of them has different advantages and disadvantages. This chapter will describe all possibilities and summarize their pros and cons. Afterwards, an evaluation will show what kind of architecture fits best for Sharelock.

3.1.1. Offline Architecture

The first possible architecture consists of devices which are offline. This means they do not have a direct communication to the internet. However, they contain communication technologies like Bluetooth, WLAN, NFC or others with which they can connect to the internet respectively the server indirect via the client.
This architecture implies that the client (smartphone) has to connect to the sharing server via internet and to the sharing device with one of its communication technologies (e.g. Bluetooth). The big advantage of this setup is that there are no running costs for mobile internet for the locks. Moreover, the lock

is more difficult to hack as a potential offender has to be nearby.
A big disadvantage is that the provider cannot maintain the lock from remote. Thus, he has to physically access the device which normally raises costs for transportation purposes. I addition the whole logic for the opening and closing has to be implemented in the application from the client (smartphone application). This makes it hard to integrate such a sharing service into an overall mobility platform, because this platform would always have to adapt to the logic if it changes.
Another one is that the exact position of the lock has to be send by the client and it has to be trusted that it is correct. Furthermore, inside cities the accuracy of GPS is often not very precise. Therefore, it is possible that the uploaded position is not precise enough.

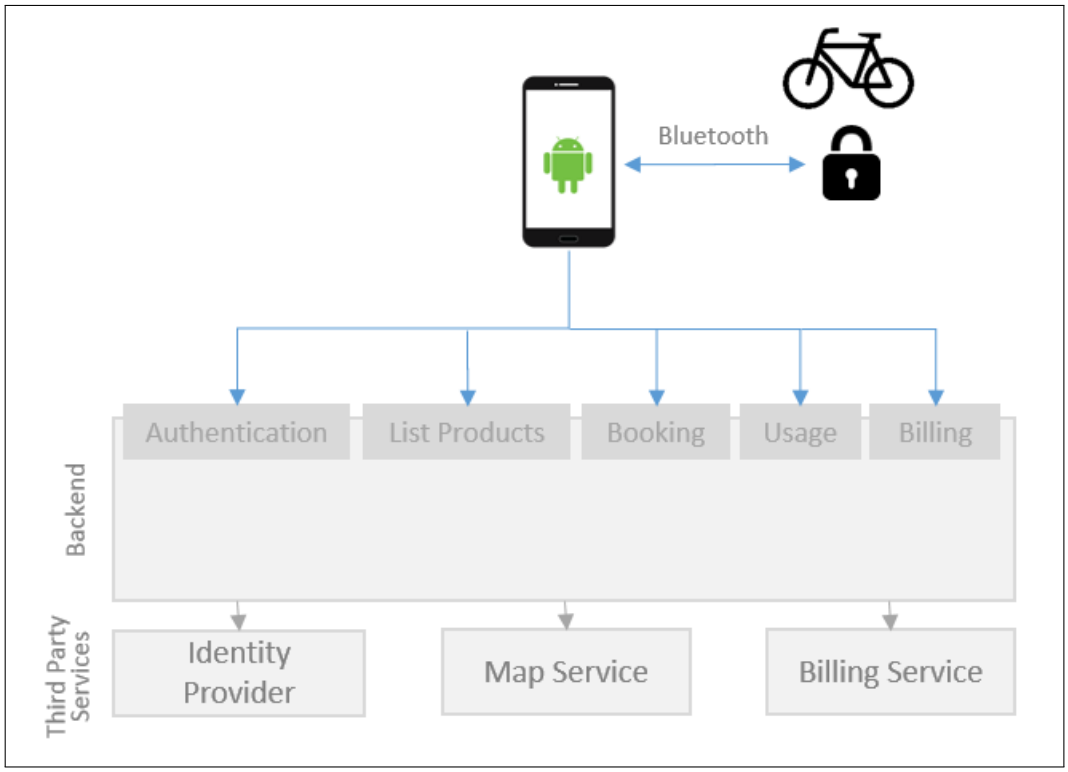


Figure 3.1.: Offline Architecture: The lock can only communicate with the smartphone. It cannot directly communicate with the application server.

3.1.2. Online Architecture

An online architecture is the second possible setup, which means that the lock is directly connected to the internet. This can be done for example with a GSM module attached to the hardware. Because of that, the lock could communicate over mobile internet.
Big advantage of this architecture is that the whole procedure of opening and closing is programmed on the server. The user does not need to know anything about the logic or the technology. He just knows the REST interface from the server with which he can open and close every lock. Hence, he only needs a device which is connected to the internet and can send data. Other communication technologies like Bluetooth, WLAN or NFC are unnecessary. Thus, a long list of supported devices for the client besides smartphones is possible like independent smartwatches, smart TVs, old mobiles and so on and so forth. All important routines and data is saved on the server side as well as all necessary algorithms for the lock procedure and for handling the user.
Another advantage is the maintainability. In a pinch, an administrator could control the lock from afar. In combination with a GPS tracker, he is also capable of receiving mostly accurate positions.
Disadvantages of this setup are the higher vulnerability to offender from the internet (hacker) and the running costs for the mobile internet.

3.1.3. Hybrid Architecture

Last possible setup is hybrid architecture. It combines the technology from the offline and the online setup. This means that the lock is directly connected to the internet, but also has bluetooth or a similar technology that the client can use to communicate with the lock. Thus, it is possible to make use of the lock over a generic REST interface for all kinds of independent services. However, the sharing provider can distribute a special application for clients which communicates directly with the lock (for example via bluetooh). This application could provide extra services for the user or just reduces the needed amount of internet volume.

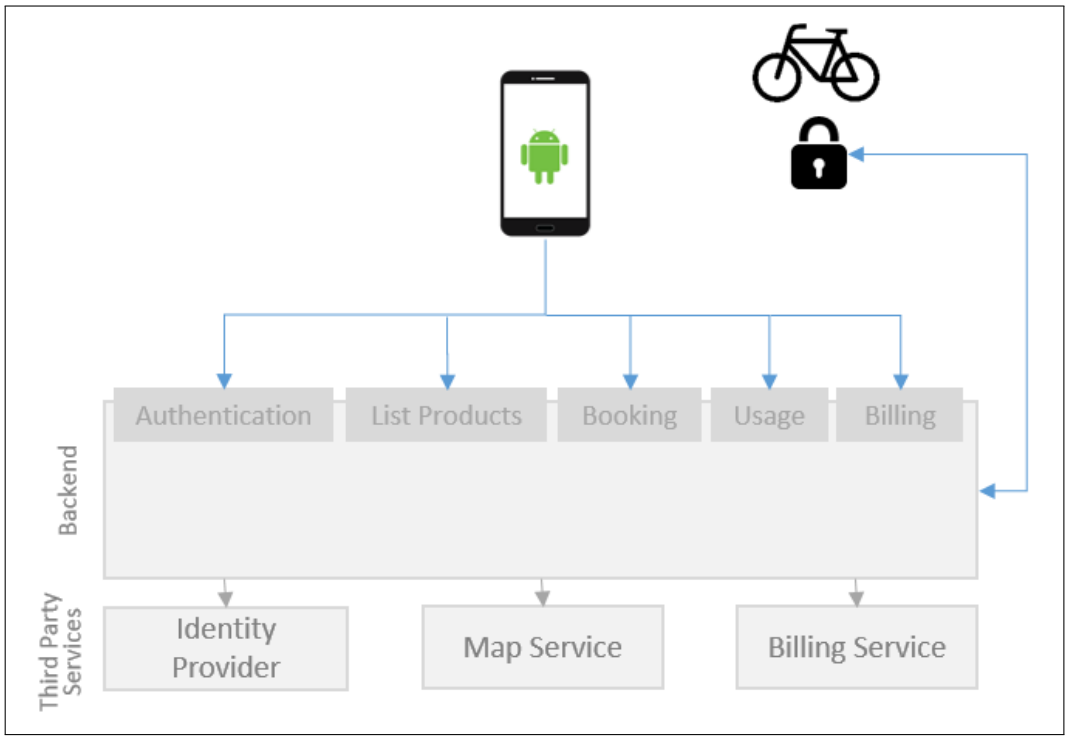


Figure 3.2.: Online Architecture: The lock can communicate directly with the application server.

3.1.4. Assessment of the different architectures

To choose an appropriate architecture it takes a complete assessment of all above mentioned types. Therefore, it needs a clear definition of special parameters for which the assessment will be performed.
First of all, it is important to judge the **number of needed communication technologies**. As more technologies are needed the worse is the solution, because not all are always available in every device. This means, that an online solution is highly preferred compared to a solution where the lock is not in the internet and has to be reached for example by bluetooth.
The next parameter is the **price** for such an architecture. An offline one is clearly better for that, because the locks do not generate costs for mobile internet traffic.
Sustainability is a next interesting factor for the assessment. It does not make sense to develop something which does not have a future. To rate the different architectures against this parameter is more difficult. It is not possible to predict how the future will proceed. There is a high chance that in the Internet of Things all devices are connected to the internet. However, it

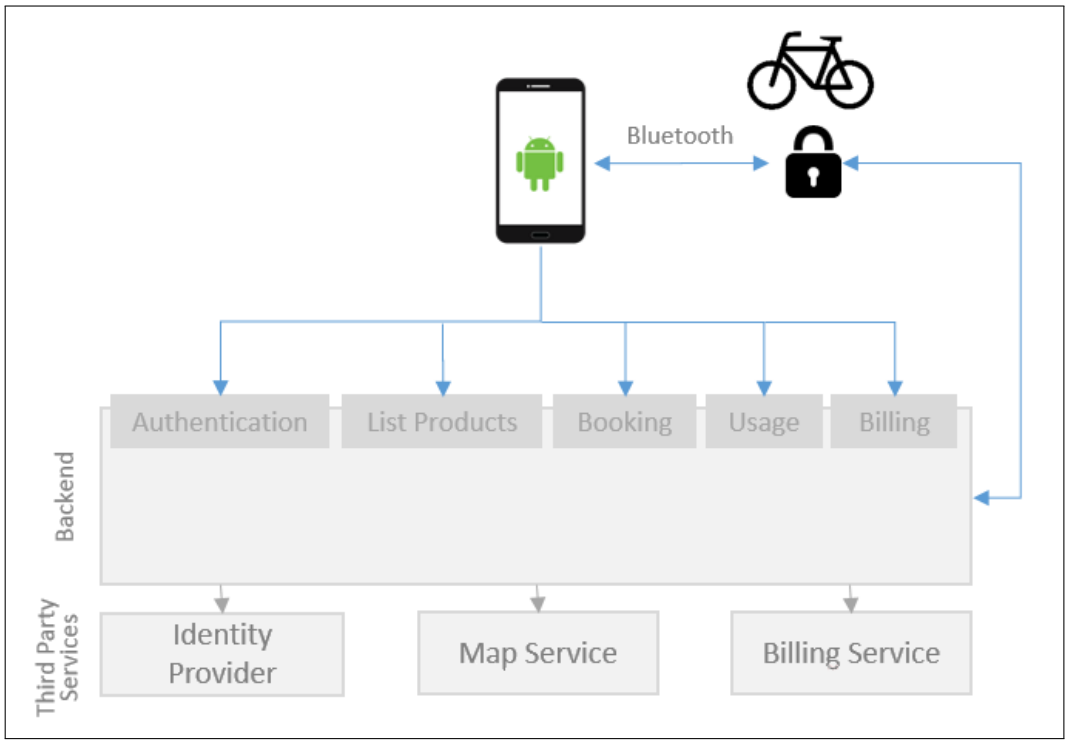


Figure 3.3.: Hybrid Architecture: The lock can directly communicate with the smartphone and the server.

is not certain how the devices are connected whether directly or indirect via Bluetooth, WLAN or other technologies. Therefore, this parameter does not give an advantage.

Furthermore, the **maintainability** should be as good and easy as possible. If errors get detected, the time to fix and update should be quick. The advantage of the online structure is that updates can be uploaded over the air (if the hardware allows that) and changes on the logic are just on the server. For the offline structure, the service provider has to physically access the device to update it. Besides, a new logic would have to be implemented on the client application. The distribution of it cannot be handled by the provider, because the client has to manually download the new release.

Last factor to evaluate is the **convenience** factor. That splits into the three aspects - convenience for the client, for the service provider and for a mobility platform provider. The first point is about the user (client). If the user owns a device, which can communicate with the lock in some way (over internet, bluetooth or something similar), he does not care about the rest. If the user cannot access the service, there is no customer and therefore the provider will not profit. With an architecture which is online, the user can utilize the service

Factor \Architecture	Online	Offline	Hybrid
Number of needed communication technologies	+	-	o
Price	-	+	-
Sustainability	o	o	+
Maintainability	+	-	+
Convenience for user	+	-	+
Convenience for service provider	o	o	o
Convenience for mobility platform provider	+	-	+
Result of aggregated factors	3	-3	3

Table 3.1.: Comparison of different possible architectures for Sharelock

and therefore the provider can obtain a profit. The next important party is the service provider. They mostly are not interested in the structure of the architecture. Depending on their goal they care more about factors like the overall profit, the customer satisfaction or others (see section about bikesharing providers). The last party is a mobility platform provider. They are interested in integrating all possible sharing services and other transportation solutions. Therefore, they need a well-designed generic interface for the sharing services. Integrating should be as easy as possible for them and their customers.

Conclusion

As outlined above, there are several parameters to consider when deciding about a setup architecture. Online and offline have both their advantages and disadvantages. Therefore, it would make sense to develop a hybrid architecture. This means that the locks are connected to the internet directly, but also have a communication technology like bluetooth on-board. Hence, it is easy to integrate into an overall mobility platform in the future.
However, the Sharelock prototype will have an offline architecture to show the feasibility of the system. The online functionality can then be added later.

3.1.5. Summary

If the lock is online or offline has strong impact on the development of Sharelock. Considering the lock to be offline means that the client (the smartphone) would contain the whole opening logic. Also the smartphone would be the connection between backend and lock and has to establish the internet connection. From this follows that an open mobility platform would have to implement the logic for the locks from each sharing service. Otherwise, the user has to download the special application from each sharing provider. Changing something on the opening logic or adding a new service would in that case mean that the application has to be updated every time.
If the lock is online (directly connected with the internet) it would only have to communicate with the backend. This time the backend would be the interface between the lock and smartphone. Advantage of this type is that the whole opening logic would run on the server and can be updated right there. This means that an open mobility ecosystem does not have to change anything as long as the REST interface persits. The mobility ecosystem just needs to know the URL of the backend server and can use every method.

3.2. Use-Case Definition

Sharelock is a bikesharing service with an electrical bike lock which can be opened and closed with a smartphone application. It is meant to provide iteratec employees a cheap possibility to use bikes to facilitate transportation and healthy living. However, it is not the intention to generate a profit with it.
In detail, users have the possibility to search for bikes in a smartphonee application (Android App). If they want to rent one of the bikes, they walk to the bike and unlock the bike lock. After their usage, they return the bike to a station or somewhere inside a predefined business district. Then, they close the lock and the booking ends. Afterwards, they are charged for the usage automatically. In the application, the user can see a list of the last bookings in an overview.
The following use-cases describe the possibilities more comprehensive.

3.2.1. Register New User

This use-case is just theoretical. Iteratec will use its own list of employees as user database (LDAP). Hence, there is no need for iteratec to register user for Sharelock. All in all, connections to databases via LDAP or active directory are possible.

Name	Register new user in the system	
Actors	Client	
Description	A new client has to register first in the system to use Sharelock. Therefore, he or she has to provide his or her user details that a new user gets created.	
Precondition	Client has no account.	
Main Scenario	**Interaction**	**System Reaction**
	Client goes to a registering form	New form appears on the screen (on a smartphone or on a computer).
	Client provides information like name, address, date of birth, username, password, bank account ...	
	Administrator submits entry.	System saves new user.
Post Conditions	Client can now use Sharelock.	

Table 3.2.: Use-Case 1: Register new user

3.2.2. Register New Lock

Another use-case is for registering locks. Before someone can use a bike, it has to be registered in the system (see table 3.3). This lock cannot be deleted. However, it can be deactivated for further usages.
Before the administrator can register a lock, it has to be created by an external server. Then, the administrator can choose this lock in an administration frontend (see section 4.3). At the beginning, there are no information about the lock except of the id and name. Entering all information like fleet, price model and others puts the lock into the running stage. From now, the lock can be booked and used.

Name	Register new lock in the system	
Actors	Administrator	
Description	The administrator wants to register a new bike lock (Sharelock) in the database as part of a new shared bike. Therefore, the administrator has to fill out a form with all information about the lock. These information are: lock number, device address, price for usage, business district and information about the bike model.	
Precondition	Administrator has a Sharelock.	
Main Scenario	**Interaction**	**System Reaction**
	Administrator choose "Register New Lock"	New form appears on the computer window.
	Administrator enters all information (see Description)	
	Administrator submits entry.	System saves new bike in the database.
Post Conditions	Newly registered data are shown.	

Table 3.3.: Use-Case 2: Register new lock

3.2.3. Open Lock

Opening the lock is the main use-case for using Sharelock. First, the user has to find a bike and must connect to the lock. Opening the lock means always starting a booking process. It can only be opened if it was closed before and if the user is authorized for the bike.

Name	Open lock process.	
Actors	Client	
Description	User searches for a nearby bike and walks to it. At arrival, the client opens the lock and starts the booking.	
Precondition	The user wants to use a bike and is registered in the platform.	
Main Scenario	**Interaction**	**System Reaction**
	Client starts application on the mobile phone and searches for a nearby bike.	System sends all/nearby bikes.
	Client selects one bike in the application and walks to it.	System sends further information about the bike.
	Client starts booking.	System creates a booking.
	Client opens lock.	
	Client starts usage of the bike.	
Post Conditions	Start time of booking is shown.	

Table 3.4.: Use-Case 3: Open lock

3.2.4. Close Lock

Beside opening the lock, closing the lock is another essential use-case for Sharelock. Closing the lock can only be done by the user who is at the lock. Precondition for that case is that the lock is open. Also the user who wants to close the lock must be the one who opened it.

Name	Close lock after usage	
Actors	Client	
Description	Client ends the usage of the bike and the booking.	
Precondition	Lock is open and the client is the actual user of the bike.	
Main Scenario	**Interaction**	**System Reaction**
	The client closes the lock.	The system stops the booking.
Post Conditions	User sees all relevant information about the last usage.	

Table 3.5.: Use-Case 4: Close lock

3.2.5. Update Bike Location

There are several events where it is important to have the functionality of relocating a bike. For example when a mechanic has to repair a bike and places it at another position. Also in case that the bikes agglomerate in one region. Then, the operator may have to relocate them for balancing purposes.

Name	Update bike location	
Actors	Client	
Description	In the case that someone relocates the bike, it must be possible to update the position.	
Precondition	A client senses a bike at a position where it should not be. Proper positioning sensor must be available.	
Main Scenario	**Interaction**	**System Reaction**
	User senses a lock and asks for correct position.	System sends saved location of the lock.
	User checks if saved location is correct or if the lock is at another position from where it should be.	
	If lock is at another location, the client sends the new position.	
Post Conditions	User sees updated position on the map.	

Table 3.6.: Use-Case 5: Update bike location

3.2.6. Show bill

After using a bike, customers want to have the possibility to check their rides and bills. This use-case is essential when an application should fulfill all important ones. Bills are created automatically by the system and cannot be deleted.

Name	Show a bill	
Actors	User/Administrator	
Description	A user or an administrator wants to read a bill from a booking.	
Precondition	Booking must be created.	
Main Scenario	**Interaction**	**System Reaction**
	Client asks for a specific bill.	System sends bill with all relevant data.
Post Conditions	Client receives bill.	

Table 3.7.: Use-Case 6: Show a specific bill

3.2.7. Create/Delete Maintenance

Every bike needs some kind of maintenance. This maintenance has to be documented and created in a way. This can be done by the bikesharing operator or by customers when they realize broken parts (see table 3.8). After the maintenance is finished, the administrator can also delete the task.

Name	Create Maintenance	
Actors	Customer/Administrator	
Description	A user or an administrator creates a maintenance for a bike.	
Precondition	Bike must exist in the database.	
Main Scenario	**Interaction**	**System Reaction**
	Client sends reason for maintenance.	System shows maintenance task.
	Administrator or mechanic assigns the task.	
Post Conditions	System notifies mechanic.	

Table 3.8.: Use-Case 7: Create a maintenance for a bike

3.2.8. Create/Delete Fleet

Usually, products in sharing systems are joint together in fleets. In that way, changes to relevant data can be done once by changing the according fleet. An administrator can create and delete fleets accordingly. In table 3.9 is an example for the creating use-case.

Name	Create Fleet	
Actors	Administrator	
Description	The administrator creates a fleet for all bikes which belong together.	
Precondition		
Main Scenario	**Interaction**	**System Reaction**
	Administrator creates a fleet.	System shows created data.
Post Conditions		

Table 3.9.: Use-Case 8: Create a fleet for bikes

3.2.9. Create/Delete Price Model

Consistency is hard to maintain, especially when changing prices for the bikes. Therefore, there are different price models which you can assign to bikes. Changing the price of many bikes is then easily done by changing the underlying price model. Hence, it is possible to create and delete price models. An example for the creating is table 3.10.

Name	Create Price Model	
Actors	Administrator	
Description	An administrator creates a price model.	
Precondition		
Main Scenario	**Interaction**	**System Reaction**
	Administrator creates a new price model.	System shows the created price model.
Post Conditions		

Table 3.10.: Use-Case 9: Create a price model

3.2.10. Use-Case Overview

Figure 3.4 displays a use-case overview of Sharelock. There are different use-cases in the first row which are independent from each other.
Register a new lock is done once to create a new lock in the system. Another one is to update the position of a lock manually. That is needed, when for example a bike has to be moved to another location due to balancing of the system. Otherwise, the bikes agglomerate in some time on different hotspots. Third one is for showing the rides and bills of each customer that they get an overview.
Important for the administrator is the middle row. It shows the use-cases for creating maintenances, fleets and price models. However, the customer can also create maintenances when a defect is detected.
Usually, a customer follows the flow in the bottom row. Typically, a customer would have to register first which is the left use-case. In case of Sharelock from iteratec, there is no registering needed. Sharelock uses the existing employee database with LDAP. That is the reason why the box is highlighted.

After registering and log in, the customer can book and open a bike. This is the first usual use-case in the Sharelock application. At the end of the ride, the user just has to lock the bike which is the last box in the row.

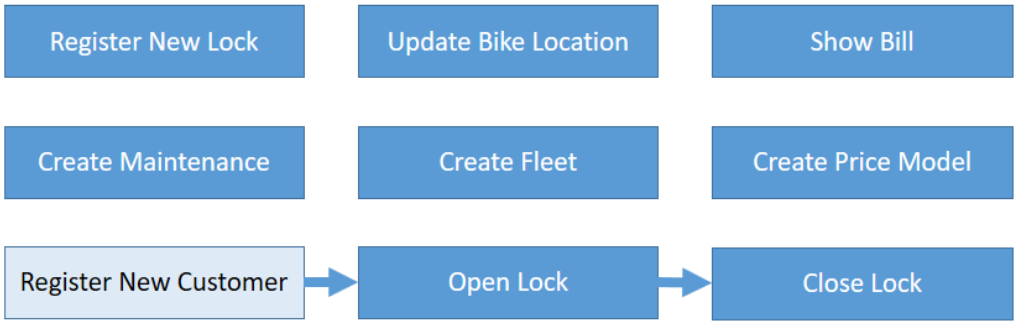


Figure 3.4.: Use-Case Overview of Sharelock. The first two rows show various use-cases. In the last is the typical flow of the customer. "Register New Customer" is highlighted because it is not needed at iteratec.

3.3. Sharelock User Workflow

Following is a general description about the main application workflow of Sharelock. It is also visualized in figure 3.5. A more specific description of the intenal message workflow can be read in section 4.1.
At the beginning (1), a potential user of the app has to login against the authentication system of the service to receive a token. With this token, access to relevant resources is now allowed.
The next step (2) of the app is to download all available locks and display them in the map view. The customer can now choose a lock from the map to load specific data like price or business district.
After decision to use the bike, the user sends a booking request to the server (3) and receives the opening code for the lock. This code will be forwarded to the lock via bluetooth (4).
While the customer uses the bike, the application can download and upload further information about and for the ride (5). These additional information can be locations for charging or return stations. Also the application could upload location points for calculating the driven distance.

For finishing the usage, the user requests the end of the booking (6) and receives the code to lock the bike. Forwarding the code to the lock (7) initiate the closing. At the end, the customer can download the bill for the ride (8).

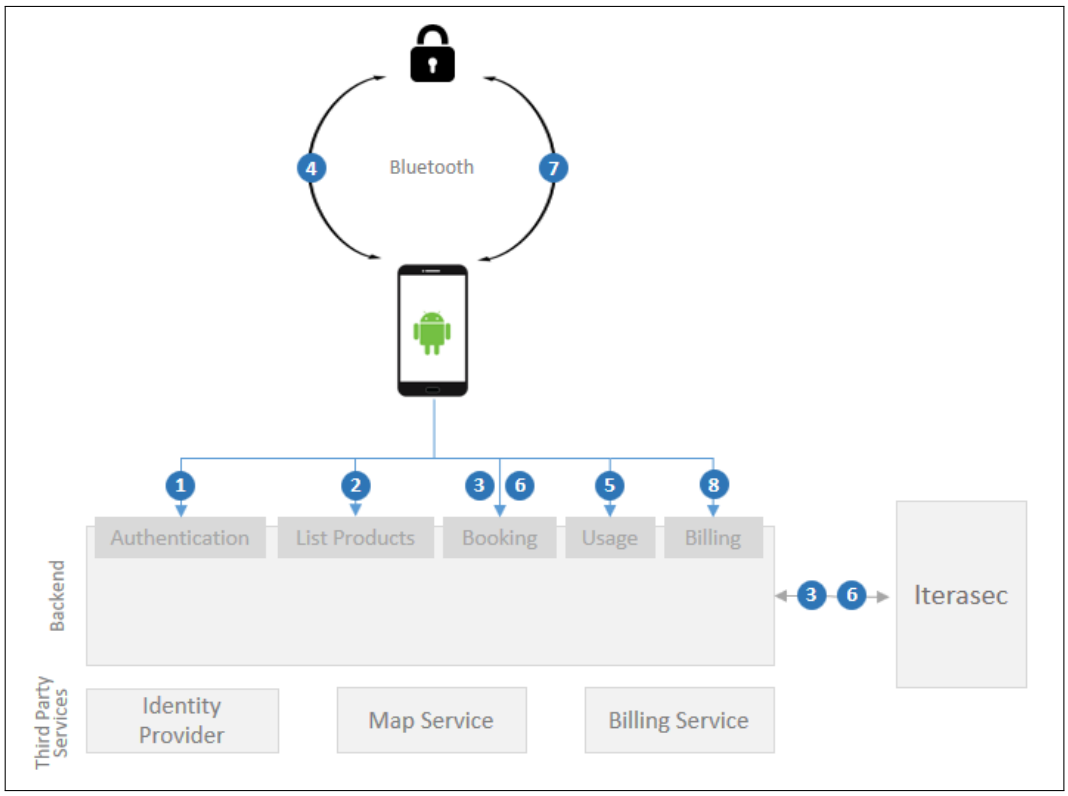


Figure 3.5.: Visualization of the basic Sharelock workflow. A smartphone uses the endpoints from the backend and communicates with the lock. Furthermore, the backend communicates with Iterasec (security backend).

3.4. Summary

This chapter is about the conceptual approach of the Sharelock bikesharing system. Firstly, an overview over the different possibilities of how to contact the lock are presented. In an online architecture, the lock is directly connected to the internet. In the offline case, the lock is just reachable via bluetooth or similar technology. Connection to the internet would then be done via smartphone.
The evaluation showed that an hybrid solution would have the advantages of both architectures. However, Sharelock will only be offline to check the feasibility of the system.
Section 3.2 outlines the different basic use-cases which have to be implemented. These use-cases are for example registering a new lock, opening and closing it or showing the bill. At the end was a little use-case flow for Sharelock displayed.
Last, section 3.3 describes the workflow which was developed for Sharelock. It displays the different interfaces of the system and which interface is used at which time.

4. Technical Approach

Following chapter describes the technical part of Sharelock. It starts with section 4.1 which explains the internal message flow between server, smartphone and lock. Afterwards, in section 4.2, 4.3 and 4.4 are the descriptions of the server, the administration frontend and the client structure. At the end, section 4.5 shows the generic REST interface for the open mobility platform.

4.1. Internal Workflow

Sharelock's whole software system architecture consists of two parts. It can be deduced from the workflow figure 4.1 on the right side. First part is the usual server which handles all requests from the client. It stores all information about the products, the customers and processes all data. It is the heart of the whole bikesharing system. Iterasec is the second system. It is the security backend which is on the right in figure 4.1. This is a specialized computer for cryptographic tasks like encryption or signing of messages. In the actual stage, it is just used for signing data or checking signed data for authentication purposes.
If a customer decides to book a bike and starts the usage, the displayed workflow from figure 4.1 begins. It starts by sending a request to the lock from the smartphone (1). The lock answers by sending a signed message for the backend (2). This message contains the request (for example for opening) and some additional data. Further information about the structure of the message can be read in section 4.2.2. This message has to be approved from the security backend (Iterasec). After receiving the message, the smartphone automatically forwards it to the backend (3). Before the backend forwards the message to the security backend, it checks if the user is allowed to use that bike (4). When the user is allowed to use the bike, the backend forwards the message to Iterasec (5). It checks the signed message if it comes from the lock. When

the check is passed, the security backend generates an approval message which is also signed (this time by Iterasec). Iterasec sends the generated message to the backend server (6) which forwards it to the smartphone (7). Next, the smartphone sends the message to the lock (8). At the end, the lock checks the message if it is from Iterasec. After passing that check, the lock opens and the usage starts. It is possible to send a maintenance task from the smartphone to the server if the customer realizes a broken part or an error at the lock or bike.

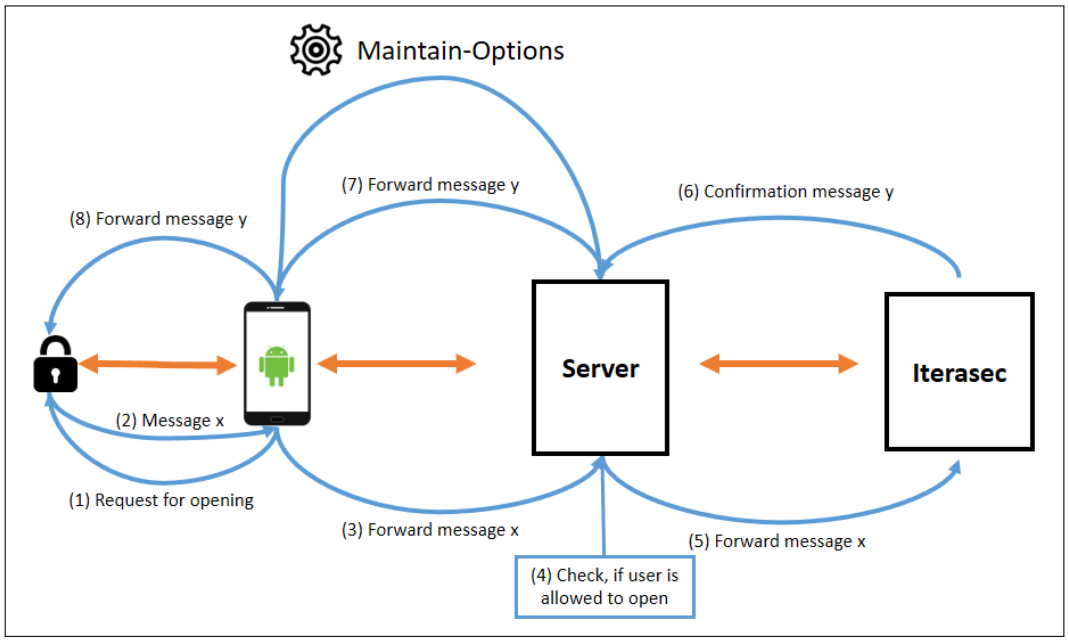


Figure 4.1.: Intenal Sharelock workflow. 1) Smartphone requests lock to open. 2) Lock sends signed message to smartphone. 3) Smartphone forwards message to server. 4) Server checks if user is allowed to open and use the lock. 5) Server forwards message to Iterasec if check passed. 6) Iterasec sends confirmation message to server if message from lock is ok. 7) Server sends confirmation message to smartphone. 8) Smartphone forwards message to lock which opens if the message is ok. Smartphone can also send a maintenance task to the server.

4.2. Server Structure

This section is divided into two parts. The first one describes the normal server and the second one the security backend (Iterasec).

4.2.1. Server

Sharelock's backend side is based on Spring Boot[18] which creates runnable Spring Applications. This system consists of three layers (API, Controller, Model & Database) which can be seen in figure 4.2.
On the bottom is the layer for the interface to the database. It handles all methods to create, alter and delete data in the database.
Incoming HTTP requests treats the top layer. It is the controller. Each request will be checked and filtered for security and authentication purposes. When the request passes through the filters, it will be processed by the controller.
Between both layer is the service layer. It is called by the controller and checks all calls for correctness. For example if data are in the right type or if they make logic sense.
Illustrated in figure 4.2 is that there is a connection to a payment service. This is not implemented yet but needs to be there in the future. It bills the customer after a ride.

[18] `https://projects.spring.io/spring-boot/`

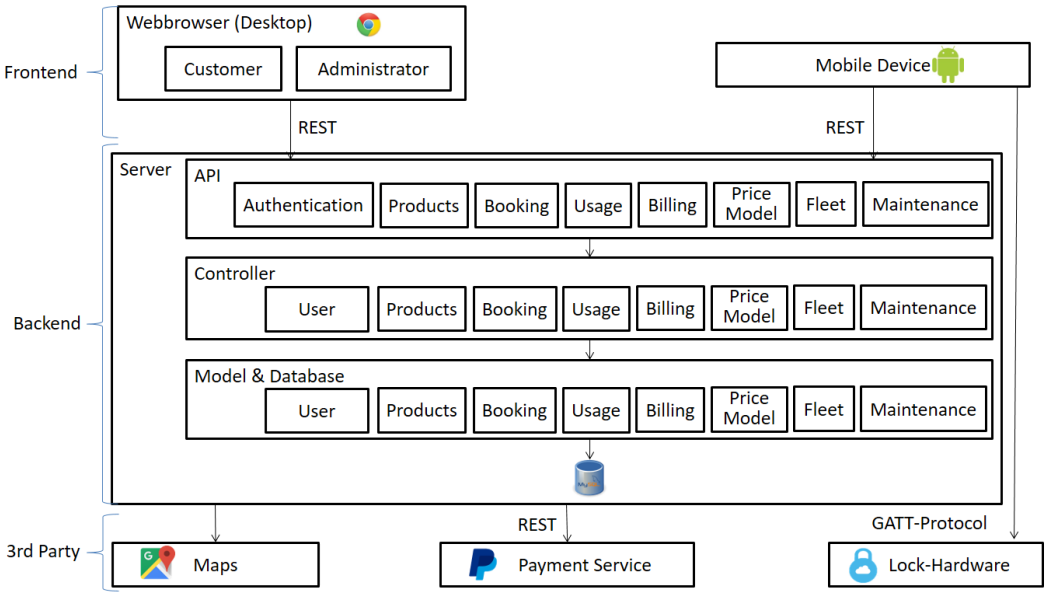


Figure 4.2.: System Structure of Sharelock. On top are the clients who connect to the backend. Bottom row shows external parties. Google Maps is used to display maps in the frontend. A payment service is not yet implemented but would be used to bill the customer. Lock-Hardware is the actual lock. In the middle are the three layers of the backend displayed.

A deeper insight into the system can be seen in figure 4.3 and figure 4.4. It is divided into two separate figures for better visualization purposes. Figure 4.3 displays the relevant classes for the generic REST interface.
Both class diagrams do not show all but the important classes of the backend. Displaying all classes would radically decrease the readability and understandability. Therefore, only relevant files are presented.
It can be seen in both diagrams that the structure is divided in four layers. The first layer is the controller (resource classes) for all incoming REST calls. At the bottom are the repository classes. They are the connection to the database. In between are the service files. They are all divided into the interface (second layer) and the implementation (third layer).

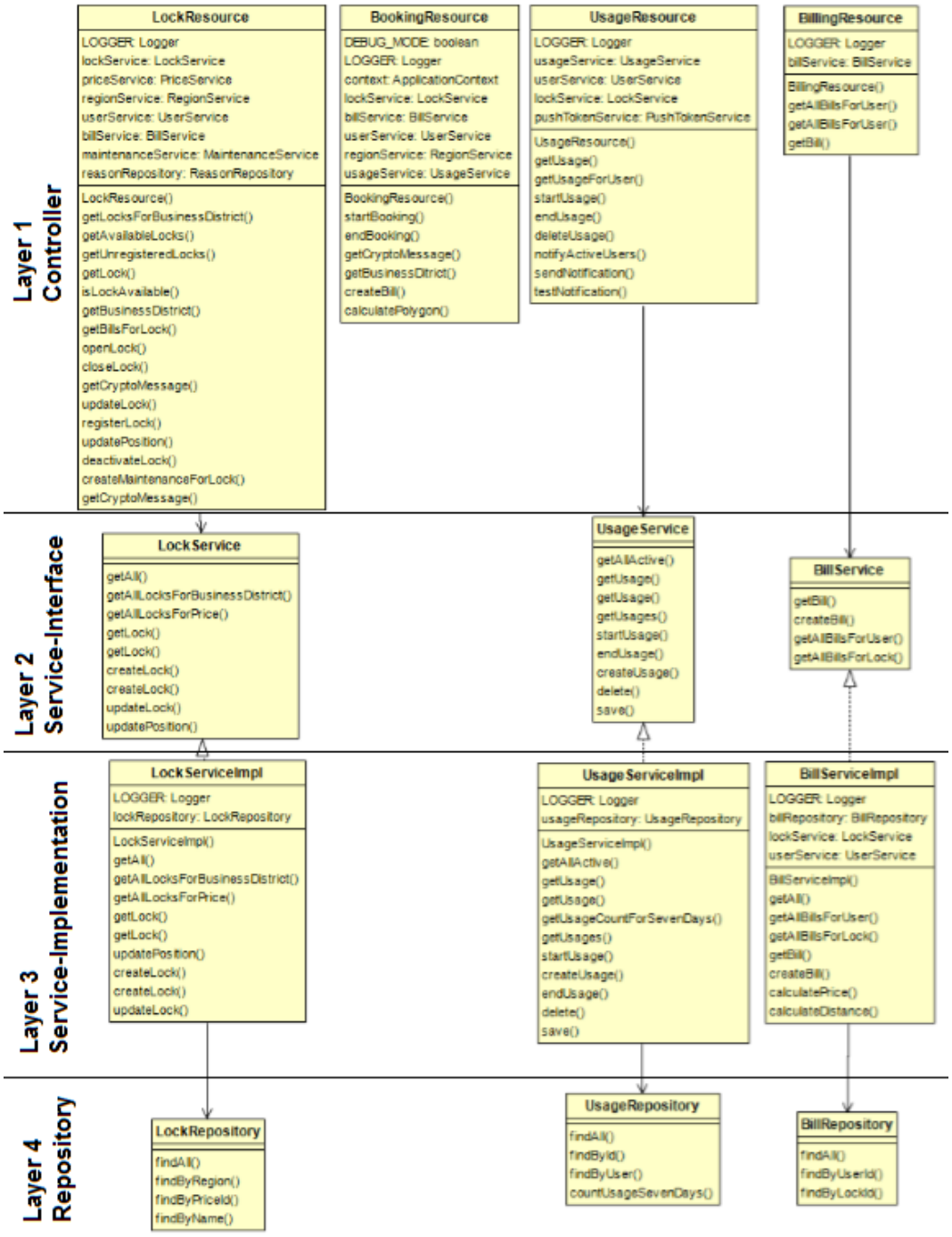


Figure 4.3.: First part of the server class diagram from the backend. It shows all relevant classes for the generic interface. It is divided in four layers. The top layer is the controller for all incoming REST calls. At the bottom are the repository classes. They are the connection to the database. In between are the service files. They are all divided into the interface (second layer) and the implementation (third layer).

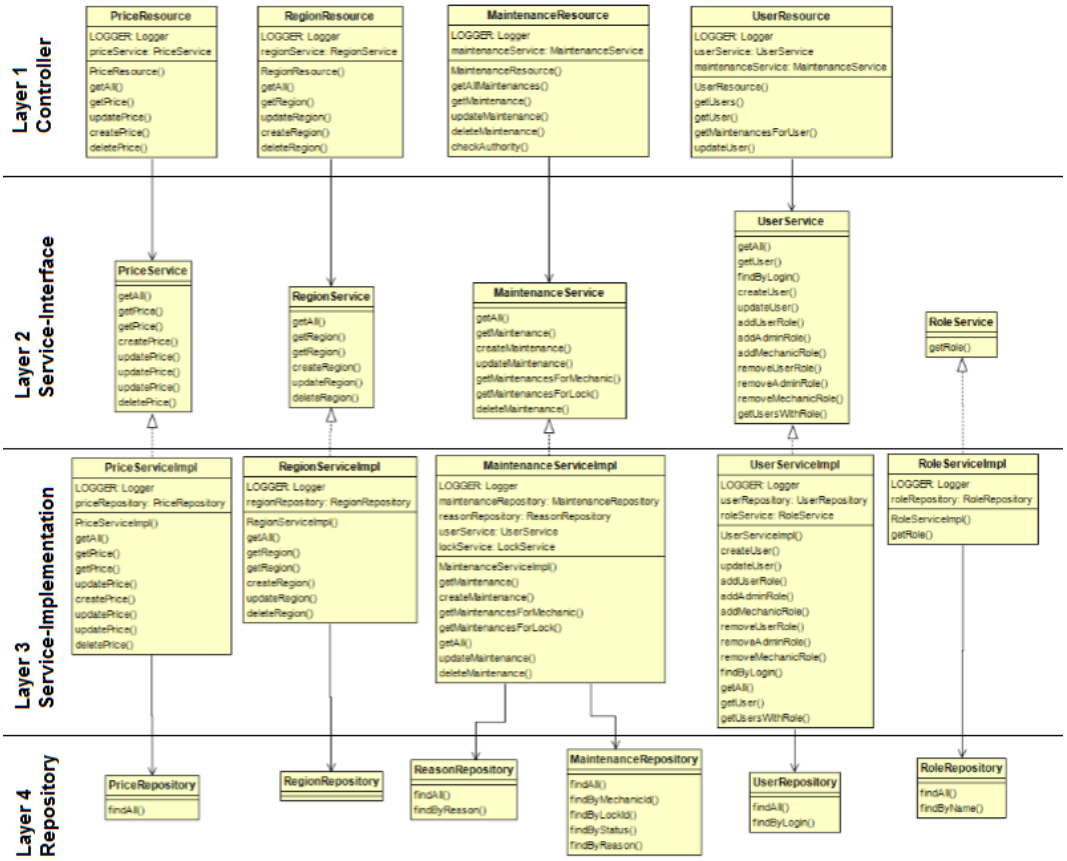


Figure 4.4.: Second part of the server class diagram from the backend. It is divided in four layers. The top layer is the controller for all incoming REST calls. At the bottom are the repository classes. They are the connection to the database. In between are the service files. They are all divided into the interface (second layer) and the implementation (third layer).

Sharelock's database entity model is presented in figure 4.5. Entities have an own table in the database. All attributes of the entities are in the diagram. For communication with the database are the repository classes from figure 4.3 and figure 4.4.

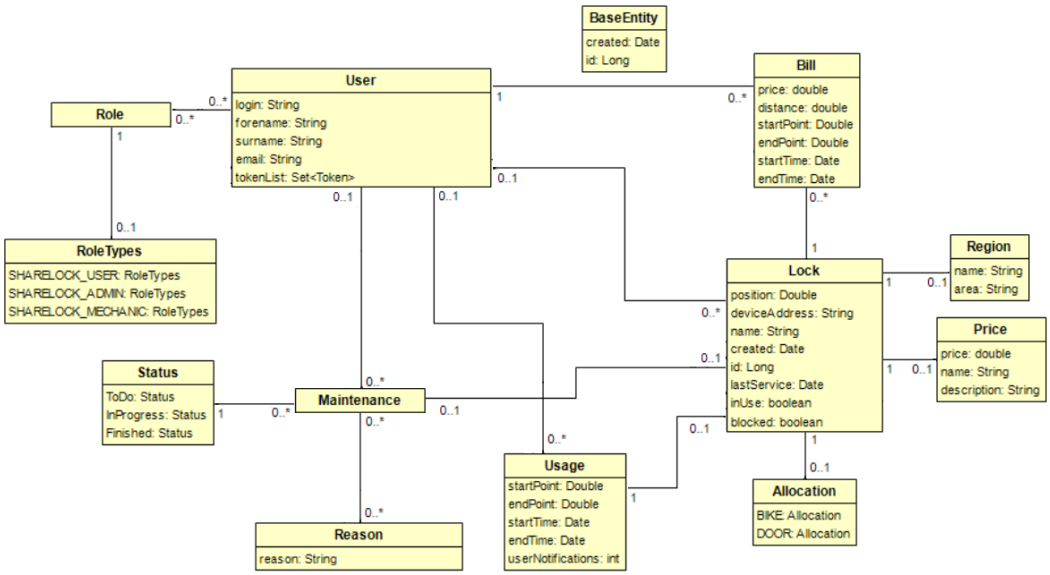


Figure 4.5.: Database Entity Model of Sharelock. All major entities extend BaseEntity and therefore have an id and a creation date.

4.2.2. Security Backend

The security backend runs on another computer (Raspberry Pi[19]). It is only used to check signatures of messages and create confirmation messages for requests. The responses are signed with the key from the security backend.
Iterasec (the security backend of Sharelock) receives the lock messages from the Sharelock server. A message consists of five different parts.
First part (4 bytes) is the ID of the sender of the message. 0x0000 is reserved for the security backend and the rest for all locks.
The next part (2 byte) is the type of the message. For example, if it is a status message or a request. In addition, it can contain the information that the message is a request for a key generating or distribution.
Third part (2 byte) holds the data field. It depends on the type of the message (status or request) if the data describe that the lock is open or closed or if it should lock or unlock. Furthermore, it can signalize that the lock has an internal error.
At the end of the main data message is a random number attached to make it unique. The reason for that lies in the need for security. Sharelock would work with just that. However, everyone could write messages to open and close the

[19] `https://www.raspberrypi.org/`

lock without any restrictions. Also without paying for the service. That is why a signature is added to the main data message. This signature authenticates the sender of the message. Therefore, nobody can fake a message to illegal use or change a lock.
The signature is based on an elliptic curve algorithm which uses public and private keys. The private key are only stored at the specific sender to maintain the correct authentication. Thus, Sharelock can just be used from customers and no one can use the locks for somebody else.
Further information about the security backend can be found in the master thesis from Rolf Sotzek from iteratec [So].

4.3. Administration Frontend Structure

The administration frontend is written with Angular 2.0[20] which was being highly developed at the time of the thesis. In September 2016 was the first stable release published. There are several advantages over the older Angular version which can be found on the official website[21].
Actually, the frontend is just for the administrator. In the beginning, the administrator has to login to the system with the login credentials. This is visualized in figure 4.6.

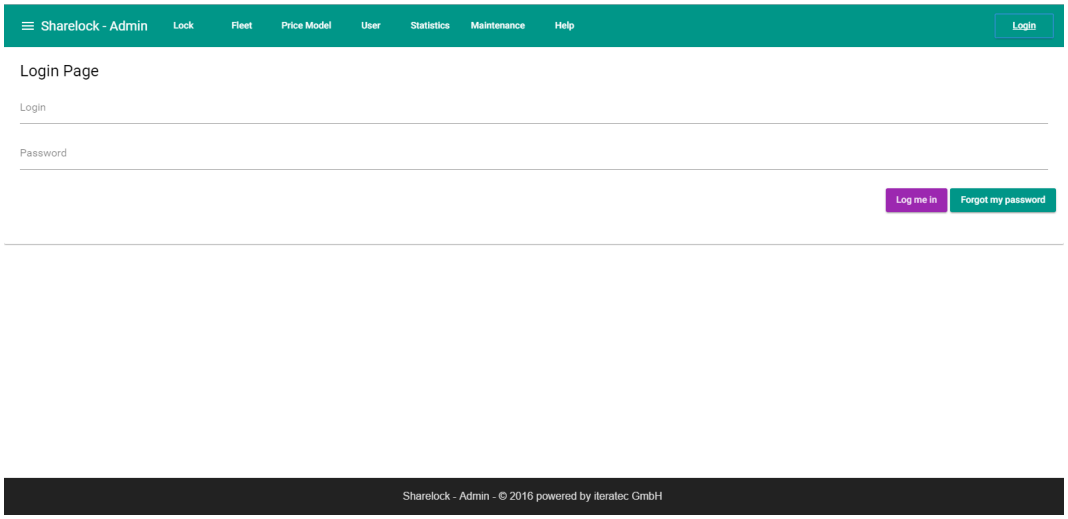
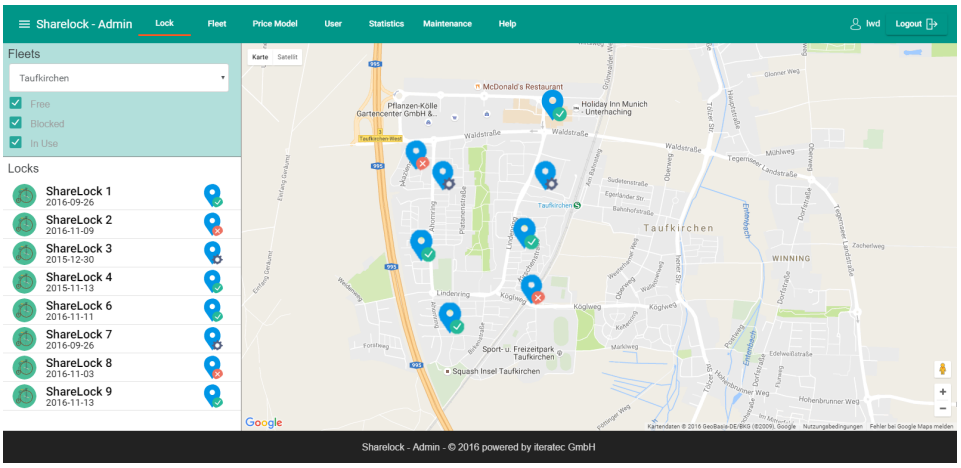


Figure 4.6.: Login to the administration frontend with the login credentials.

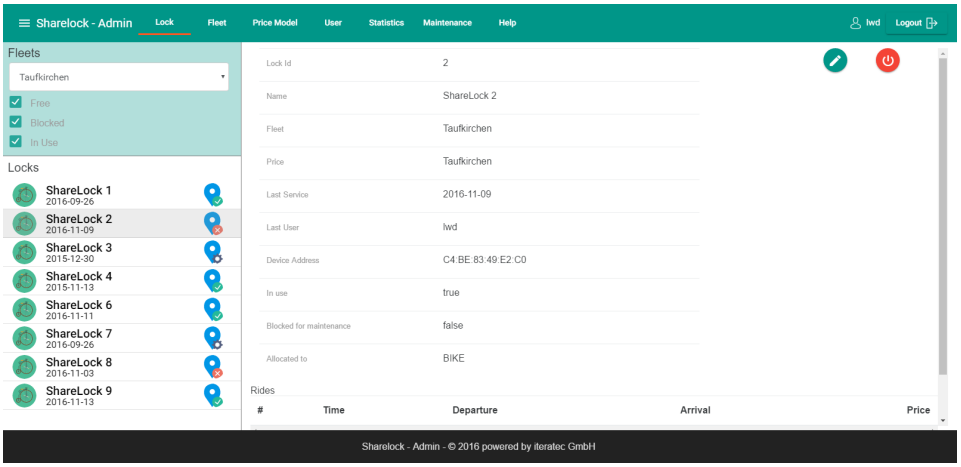
[20] https://angular.io/
[21] Angular 2: https://angular.io/

Afterwards, it shows different information about the products (figure 4.7), fleets (figure 4.8) and price models (figure 4.9). According to the relevance, these information can be changed and deleted. For example locks cannot be deleted. They are just deactivated for the system to keep all relevant data. As the bikes can only be returned in a specific region, they get assigned to fleets. Fleets have a business districts which is a polygon in which the bikes can be returned. For adding polygons, KML[22] files are needed. Additionally, the administrator can see a list of users (figure 4.10) with the bills of each ride and maintenance tasks for the locks (figure 4.11). There is the possibility to change information about the maintenance or giving the user various rights.

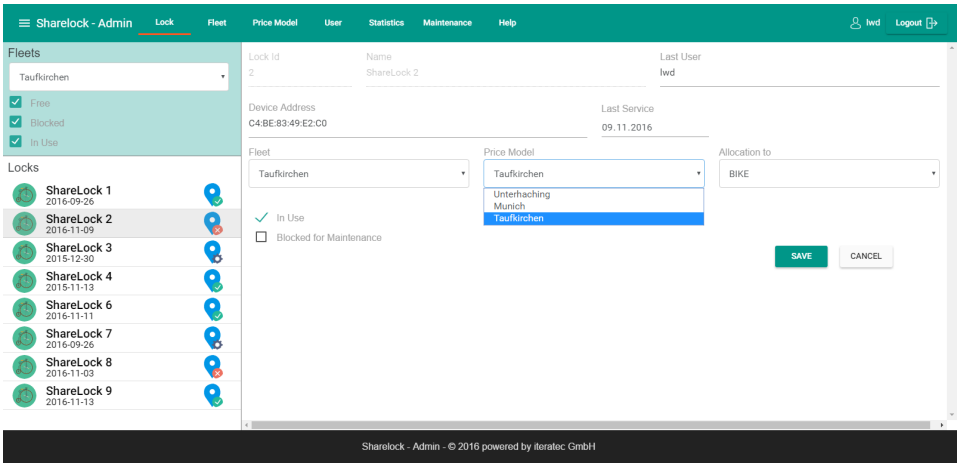
[22]KML (Keyhole Markup Language) is the standardized file type for geographic data and information.



(a) Map overview of all bikes which are belong to the fleet "Taufkirchen"

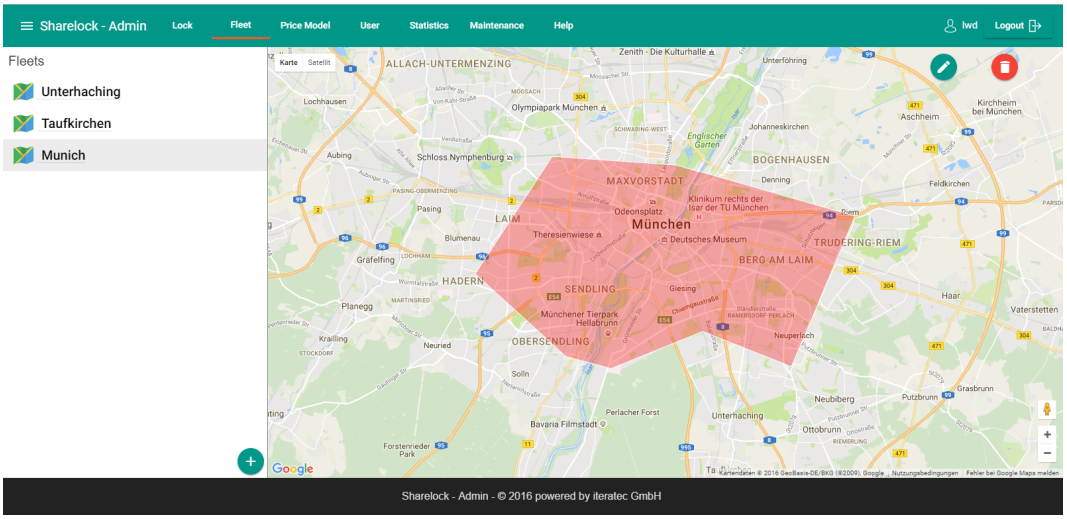


(b) All information to the selected bike on the right side. Performed rides would be shown at the bottom. Editing or deactivating of the lock can be done with the button in the top right corner.

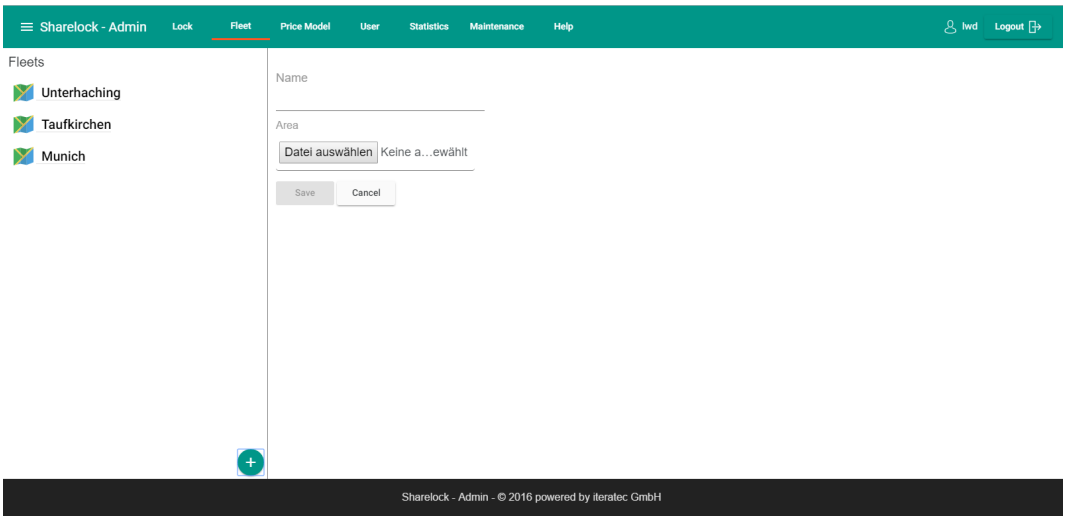


(c) Editing information about the bike. Last user is the customer who used the bike before. Last service is the date of last maintenance. Price, fleet and allocation can be choosed from a dropdown.

Figure 4.7.



(a) Map which shows the business district of the fleet "Munich". Pressing the add button in the bottom left corner leads to the next screen.



(b) Adding a new fleet. It needs a name for the fleet and an uploaded KML file.

Figure 4.8.

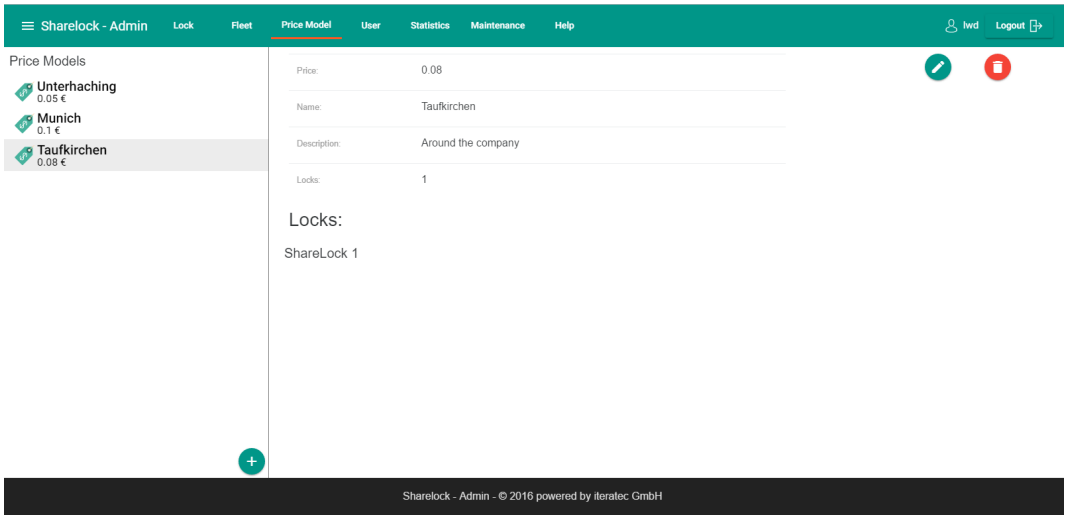


Figure 4.9.: Information about the price model. These information can be edited or deleted by the buttons in the top right corner. A new price model can be added with the button in the bottom left corner.

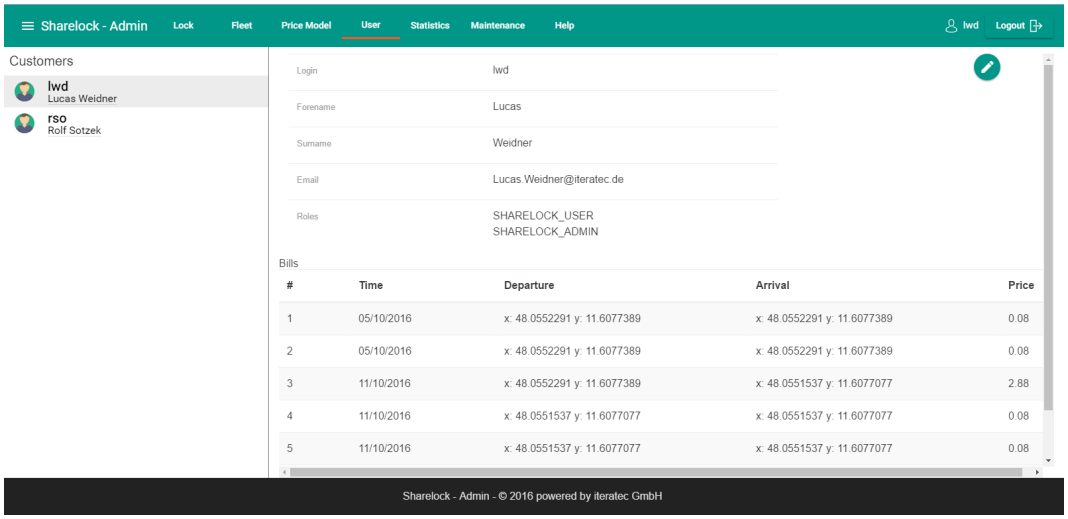
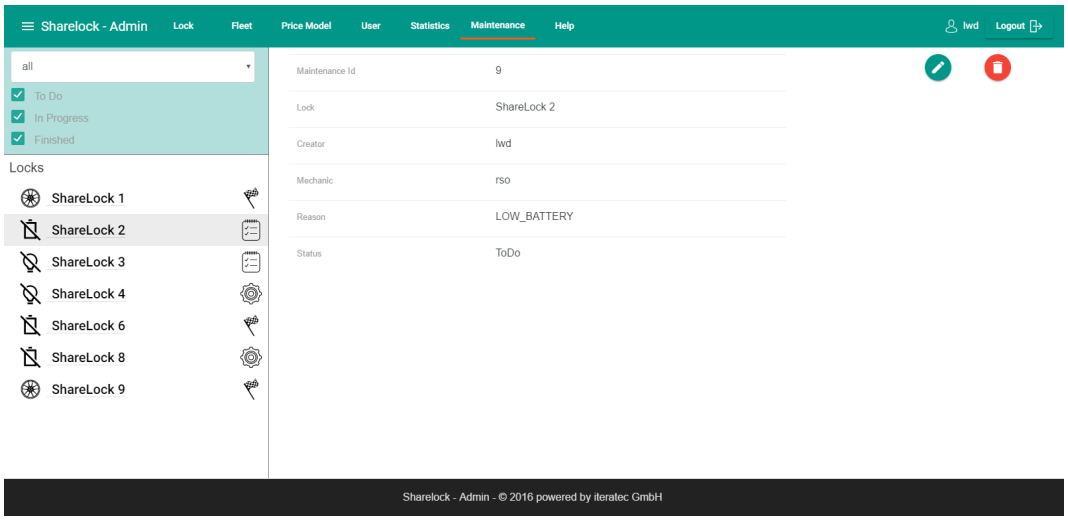
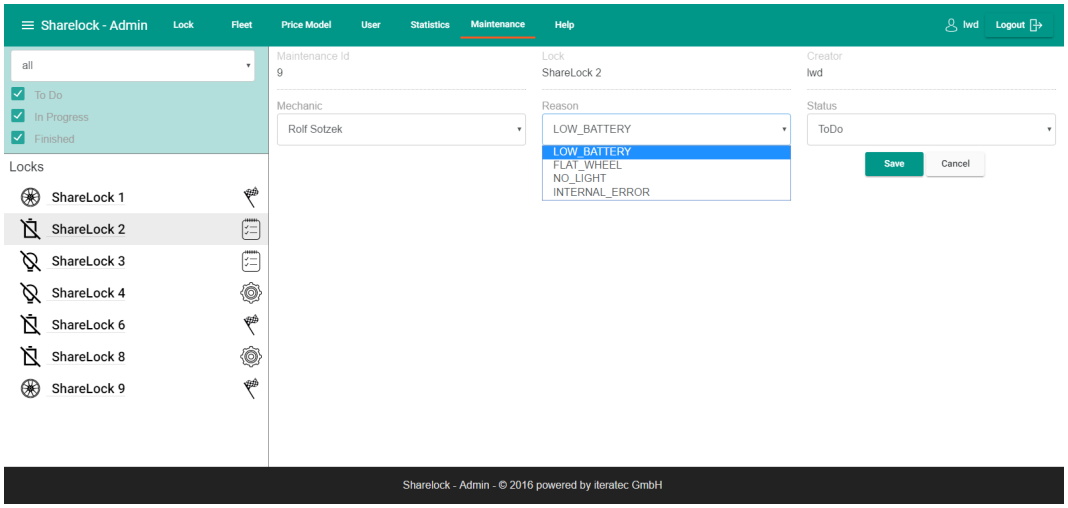


Figure 4.10.: Relevant information about the customer. At the bottom is a table which shows the performed rides of the user. It displays the bill number, starting time, departure and arrival point and the price for the ride.



(a) Actual information about a maintenance. On the left side is a list with all maintenance tasks.



(b) Changing relevant information of a maintenance like reason, status or mechanic.

Figure 4.11.

There are two different map providers which were considered for visualization of the bike locations and the business districts. The first one is HERE and the second one is Google Maps. A comparison of the prices[23] from both provider is in table 4.1. At the end, Google Maps is used because it is cheaper and has all the needed functionality. In addition, it has the better support.

[23]Source of the Google maps prices: `https://developers.google.com/maps/pricing-and-plans/`

	Today	Future (B2B)	Future (B2C)
Who pays for the application?	iteratec	customers or company	customers
How will the application be distributed?	online storage	App Store or online storage	App Store
How much users do the application has?	< 100	> 1000	> 1000
Online or offline usage of the map?	online	online	online
Do we need navigation?	no	maybe	maybe
Is it B2B or B2C?	B2C	B2B	B2C
Price HERE map without navigation	> 1.500 €	> 16.000 €	> 1.900 €
Price HERE map with navigation	> 1.900 €	> 31.000 €	> 5.900 €
Price Google maps without navigation	~ 0 €	~ 375 €	~ 375 €
Price Google maps with navigation	~ 0 €	~ 750 €	~ 750 €

Table 4.1.: Price Comparison between Google and HERE

4.4. Mobile App Structure

This section covers all relevant information about the Android mobile application. It starts with a short introduction about the Android system. Afterwards, the structure of the app is described. At the end is the workflow of the application visualized.

4.4.1. Android Introduction

Android applications are based on activities which are the corresponding file to the open window on the smartphone. An activity handles all important events and updates the window respectively. For example, it handles button events or screen loading. Methods inside activities should not block, because the application would crash.
There are other possibilities when an application runs tasks which are blocking. Async tasks are an usual approach for blocking tasks like waiting for a response from a website. They run on another thread and do not block the UI thread (the activity).
A third thing which is often used is a fragment. An activity can load different fragments which are running inside the activity. They are easily to exchange and are faster to load compared to an activity.

4.4.2. Application Structure

Last section introduced the three main things which are needed in Sharelock's Android application. In figure 4.12 are the important classes displayed, together with their flow. All tasks which are at the bottom of each activity or fragment are AsyncTasks. They are used for non-blocking calls to the backend of Sharelock.
At the beginning is the StartScreenActivity. It checks if there is an ongoing usage or if the customer is still logged in. When the user is not logged in, the next activity is automatically the LoginActivity. Actually at iteratec, a user can login with the own employee credentials. When the user is still logged in without any actual ride, the application goes to the MainActivity which instantly loads the MapFragment. If the customer is still logged in and has an actual ride, the application jumps to the LockControlActivity.
The MapFragment displays the map with all available bikes. Therefore, it loads all bikes with the LoadLocksTask. If the application realizes that a bike is located on a wrong position, it updates the location with the UpdateLockPositionTask.
For switching purposes, a navigation drawer is included in the MainActivity.

From there, the ScanActivity (scanning for nearby bikes) and the BillListActivity (history of all rides/bills) can be reached. By clicking on a bike on the map or by choosing one in the ScanActivity, the user comes to the LockControlActivity. This activity handles all communication between the lock and the backend server. It starts and ends bookings and displays the actual usage time. It can also send maintenance tasks to the backend over a dialog. In this dialog can the user choose a reason for the maintenance from a list. Uploading is done by the BlockLockTask.
A list of all bikes which are nearby will be created by the ScanActivity. It checks with the CheckLockTask if surrounding bluetooth devices are locks from Sharelock and if they are free to use.
At the end of each booking, the application shows all bills automatically. The list displays important information like the start and end time or the price.

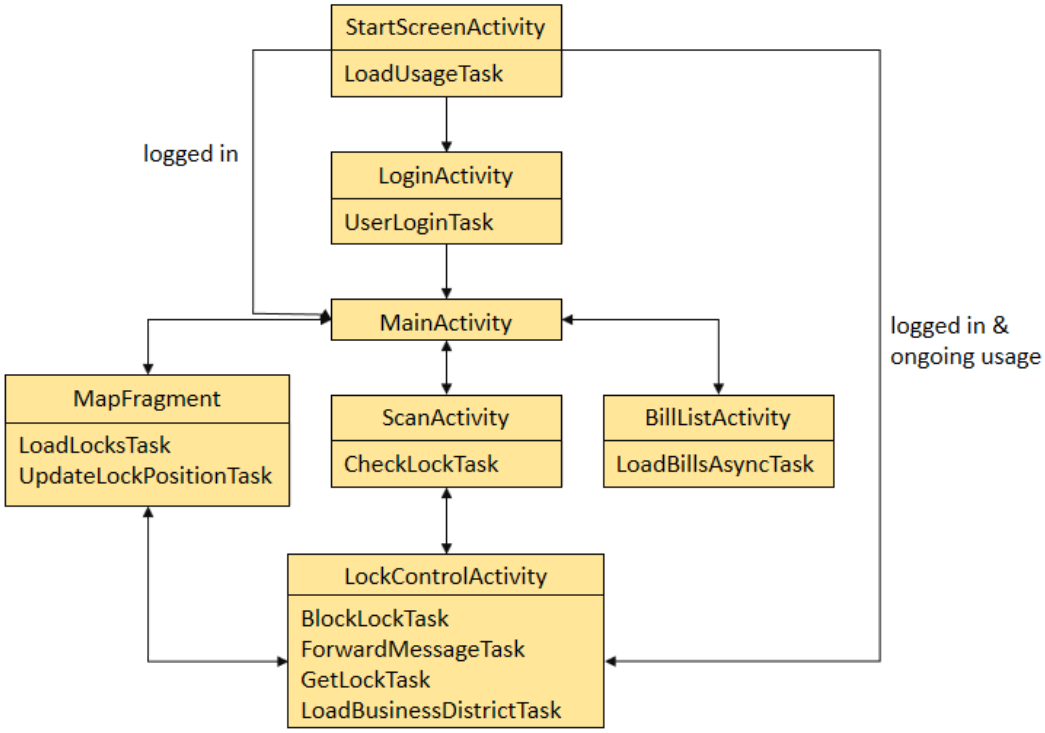
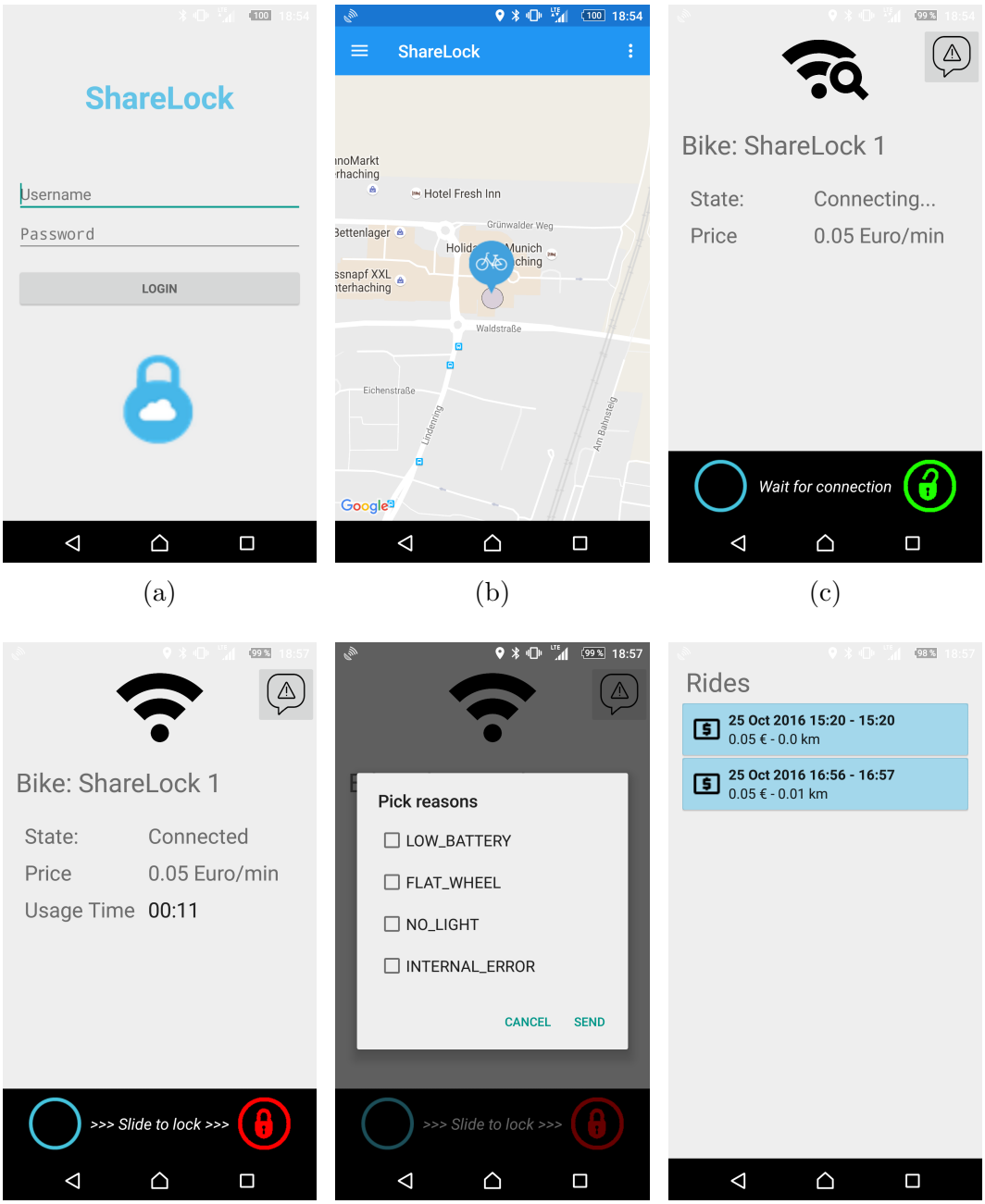


Figure 4.12.: Activity flow of the Android application. All tasks inside an activity or fragment are AsyncTasks.

4.4.3. App Workflow

The final version of the Sharelock Android application can be seen in figure 4.1. In the beginning, the customer has to login to the application (4.13a). If that was successful, the user gets forwarded to the map of bikes. While the smartphone searches for the GPS location, the customer starts with a default position. Finding the GPS location changes the view to the actual position (4.13b). On the map view are all bikes displayed which can be rented at this time. Bikes which are in use from someone else are not visualized. If there are too many bikes nearby, the customer can change to a list view.
After choosing a bike, a new activity is loaded which displays the price of the bike. In the background, the smartphone tries to connect to the bike via bluetooth (4.13c). In the top right corner is a button for sending information about bike issues like a flat wheel, no light or other problems (4.13e).
For booking and unlocking the bike, the user just has to use the slider at the bottom. This is only possible, if the connection to the bike is successfully established. For better explanation what to do changes the text on the slider from “Wait for connection” to “Slide to unlock” .
As visualized in figure (4.13d), the actual usage time is displayed during the whole journey. The customer can decide to close the app without any problems. After restarting, the user gets directed to the actual usage. Furthermore, the customer receives a reminder after every 30 min about the running usage.
For ending the usage, the customer just has to use the slider again for closing the lock. Sharelock updates the position of the lock with the position of the user. That is why the lock has no internal GPS chip. If the ending was successful, the lock closes and the user gets to the next screen which shows a history of all rides in a list (4.13f).
At the end, the customer could logout to delete the access token which causes a new login. Or the user just closes the app. In that case, the customer is remembered next time in the application.



(a) (b) (c)

(d) (e) (f)

Figure 4.13.: a) Login screen b) Map overview for all bikes c) Connecting to a bike and opening via sliding at the bottom d) Already running usage which can be ended by sliding again at the bottom e) Maintenance dialog f) Overview over all finished rides (history)

4.5. REST interface

One core element of Sharelock is the possibility to integrate it in an open mobility ecosystem. It was special designed for a clear REST structure.
Sharelock's REST interface is roughly divided into two parts. The first one is the generic interface for the open mobility ecosystem. Second part contains the remaining endpoints which are needed to fully operate the bikesharing. For example creating price models or fleets are important operations. However, they are not important for the open mobility ecosystem. This platform needs the essential possibility to load available bikes, create bookings and to load generated bills.
First important REST endpoint is for authentication purposes. As response comes a JSON Web Token (JWT)[24]. This token handles the authentication of the user for all other endpoints.
Another important interface is the list of products. Amongst others, it provides methods to register locks, update their information and of course receiving a list filtered against different options.
The next endpoint is for booking. This is used when the customer decides to ride the bike. It starts or ends a booking depending on the request method.
Fourth interface (usage) is for further information while the usage is running. This can be information to download like the next return station, the business district or points of interest. Though it is also possible to upload data like the actual position that the backend can calculate the driven distance.
Last interface for the generic purpose is billing. Obviously, it makes sense to provide the customer all bills for past rides. A concrete list of all methods and endpoints is displayed in figure 4.14 and figure 4.15. It is automatically generated by Swagger[25]. They are divided into two parts to increase the readability. Please refer to the enclosed CD for a complete documentation of all the listed methods and endpoints.

[24] `https://jwt.io/`
[25] Source: `http://swagger.io/`

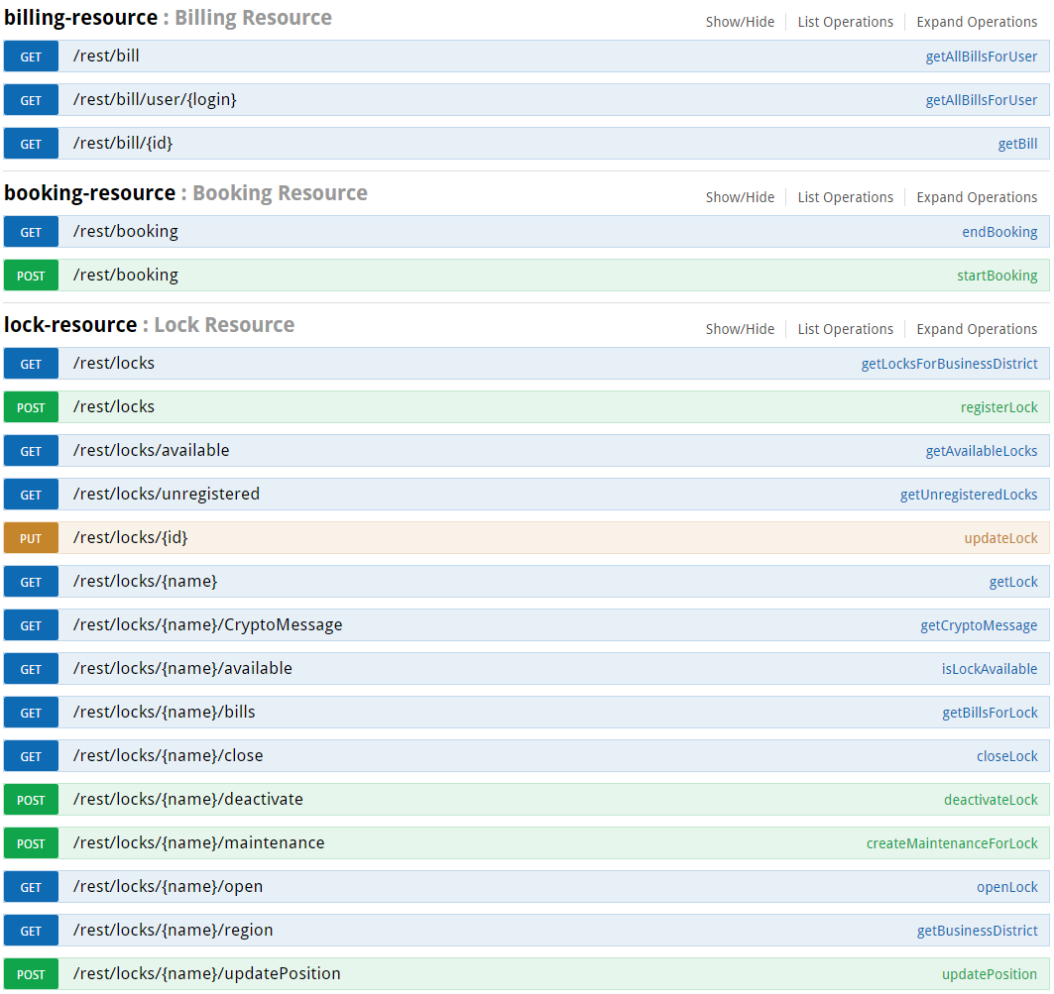


Figure 4.14.: First part of the swagger generated output of the REST interface from Sharelock.

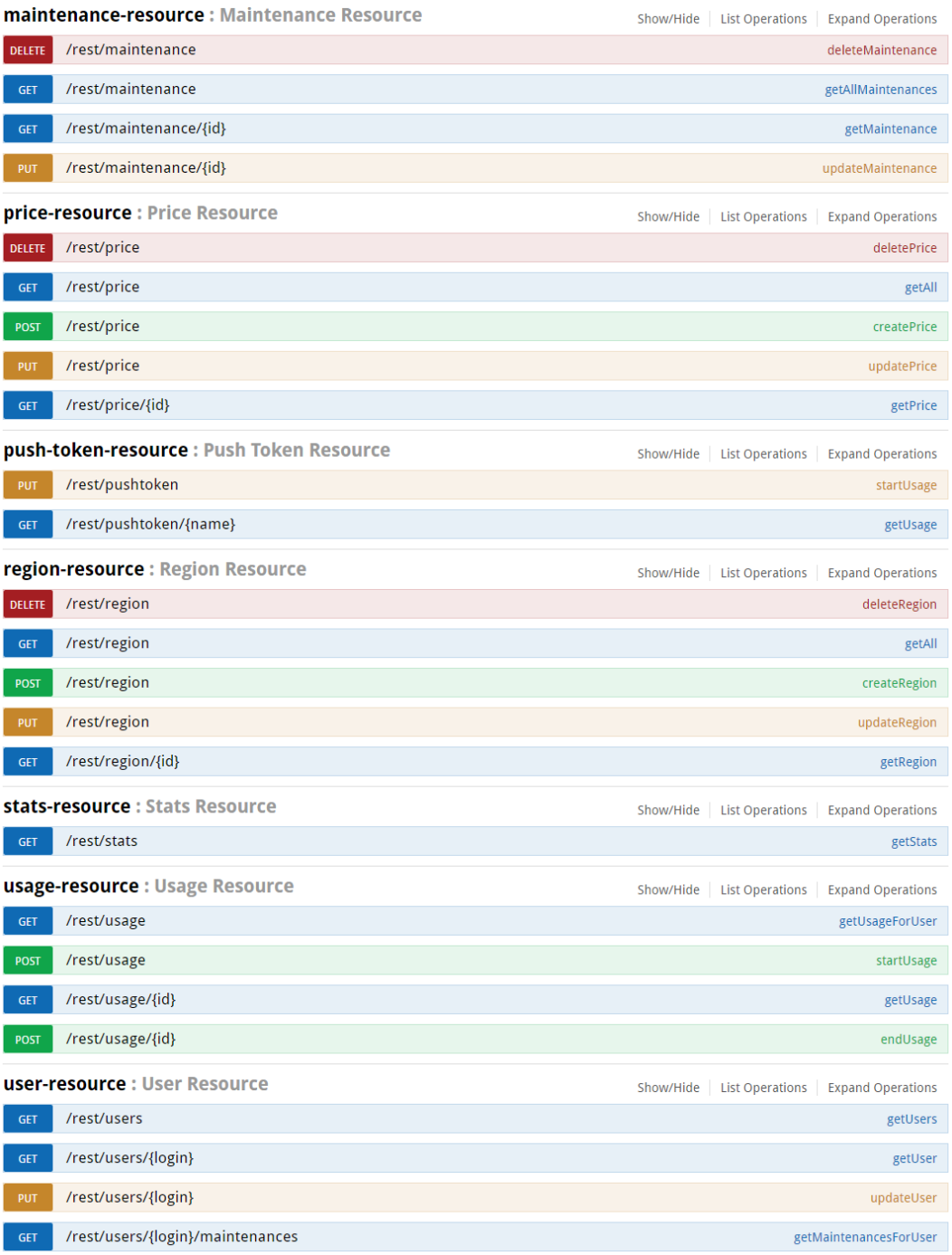


Figure 4.15.: Second part of the swagger generated output of the REST interface from Sharelock.

4.6. Summary

This chapter outlines the main structure of Sharelock. Firstly, an overview about the internal workflow is described. It shows how the messages from the lock and the backend are send between all devices.
Afterwards is the description of the server structure. This is divided into the backend and the security backend side.
Section 4.3 focused on the administration frontend structure. A description of the mobile application is in section 4.4. It begins with a short introduction about the kind of files which were used in the Android application. Then follows the description of the Sharelock application with smartphone screenshots. At the end is a short summary of how a generic REST interface looks like which can be used inside an open mobility ecosystem.

5. Evaluation

The developed systems were evaluated with the system usability scale (SUS) [Br13]. Furthermore, a questionnaire were taken to ask about the pros and cons of the application and the administration system.

5.1. Usability

All in all, nine working students from iteratec tested the Sharelock application and the administration frontend. They received a task which they should fulfill without any prior knowledge. First one was to use the application for renting a bike. While using the bike, the user would recognize that a wheel is flat and has to communicate the issue. The second task was to use the administration frontend to assign the just created maintenance to a mechanic. Afterwards, the status of the lock has to be put to unblocked. At the end, the students had to fill out a questionnaire to calculate the system usability score. The result was a score of around 80 points for the application and approximately 79 for the frontend. In comparison, a score above 68 is considered better than average with good usability.

5.2. Feedback

All working students from iteratec were asked several questions about the system and about bikesharing in general. First question was, if they used bikesharing before which all participants negated. Next one asked about how the perfect bikesharing should be. There were a lot of different answers. Most students said, that it has to be easy to use. For example, it should only need a few clicks to book a bike. Furthermore, the registration should be quick and easy and the business district should have enough bikes with return stations at

points of interest. Also mentioned was the possibility to reserve bikes, security issues, a nice layout, a fast and easy process to unlock (maximum 10 seconds until start cycling), that it is mostly automatically and easy to find bikes, that it is clear when the usage starts or ends and that it is reliable.
Next question was if they would use it at iteratec. Seven out of nine would use it. The two remaining students would not use it because they do not need it or do not cycle at all.
If it would be more interesting if Sharelock is integrated in another application such as from the public transport was the next question. Most students agreed on that. One student disagreed and wanted to separate both and another one was unsure.
Last question asked for ideas for further possibilities to use Sharelock. At the moment, Sharelock is just a lock which can be opened and closed via a smartphone application. It is not limited to use it with bikes. There was a wide range of answers. Suggestions were to use it with wheel chairs, cars, doors, lockers, rooms/offices, fitness studios, trailer and parcel services.

In the second part of the survey, the students were asked for feedback to the application and the frontend. What was good, what has to be improved, what was missing and what was unclear.
Related to the smartphone application, it was good that it was fast and easy to choose and rent a bike. Additionally, it was noted that the layout was nice and focused on the main parts of the application. Also positive mentioned was the showed usage time, that the bikes can be seen on a map and that maintenance tasks can be send. All in all, it was perceived to be very intuitive.
Nevertheless, there are things which can be improved like showing the status of the lock or the slider. In addition, a better feedback was mentioned and that the application start at the last known location. One student criticized the layout of the usage screen and that the maintenance dialog looks like an error. When the smartphone is offline or GPS is turned off, the application was not warning the user. Therefore, the customer could not rent or return a bike and does not know the reason for that. Last point that was made was that it is hard to come back to the usage screen when the user went back to the map.
Missing elements are an introduction screen for the application or the actual price like a taximeter. Additionally, it was noted that the price per minute could be shown on the map. An important feedback was also to visualize

the actual business district in the application to know where it is allowed to return the bike. Two other missing elements are the possibility to change account settings and that in the nearby bikes screen a message appears if there are no bikes.
Unclear was sometimes the slider and where the bikes can be used. Also that there was no feedback from the slider or that no internet or GPS is available. In addition, it is unclear which marker on the map belongs to which bike. Last was that the nearby bikes screen showed no information at all when there are no bikes.

For the administration frontend, most students liked the design and that it is easy to use. They could see all bikes, it is clearly arranged and consistent.
Nonetheless, some students found it could be improved by rearranging elements and put related information together. For example that a bike can be unblocked directly in the maintenance view. A few students would change the design completely or at specific sides like the editing views.
Just one student was missing something in the frontend. One thing was a small introduction feature and second one was the possibility to edit field directly in the view.
Unclear was for some students where to find specific elements and how to unblock a bike. Sometimes, the student found it unclear which maintenance they have to choose when there were multiple with the same name. It was also found to be information overloaded and unclear that the administrator has to choose a fleet before seeing a list of locks.

5.3. Summary

Sharelock was evaluated with nine working students from iteratec. They had to use the smartphone application and the administration frontend with a usual case (using a bike and maintaining). The students gave positive and negative feedback to both systems which helps for later improvement.
All in all, both systems scored a very high SUS (system usability scale) of 80 points for the application and 79 for the frontend. A score above 68 is considered above average. However, the students are smart developers at iteratec with a highly technological background. Therefore, the score has a bias and is in reality maybe lower.

6. Discussion and Outlook

6.1. Contribution

This thesis was not started from scratch. However, most of Sharelock was developed in this thesis. Only the security backend and its message structure were developed by Rolf Sotzek (working student at iteratec) in his master thesis [So]. A skeletal structure with Spring Boot was the basis from which Sharelock was developed. This basic structure involved a few entities like the lock and their connections to the database.
Certainly, this basic structure was expanded and improved in this thesis. Many important functionalities were missing like booking and billing. However, to correctly use the bikesharing, fleets, price models and maintenance were also missing. All these functionalities were implemented in this thesis. In addition, a cloud messaging system (firebase) was integrated to notify a user after each 30 min of a ride. Therefore, if the customer forgets to return the bike the smartphone receives a notification. A short description of the notification workflow from firebase is in appendix A. Also the REST interface was newly modeled. It should easy integrate into an open mobility ecosystem.
In the beginning, the frontend was based on some bootstrap files which was obsolete. So the new system is based on Angular 2 with material design which was completely new created during this thesis.
On the smartphone side was also a basic application provided. This application was able to scan and connect to the bluetooth locks. It could also send and receive messages. Nevertheless, the application was changed to a mostly new design, a better user experience and some additional features. For example the map, the navigation drawer or the list of bills. In addition, the whole communication with the backend had to be implemented as well as the login was integrated later.

6.2. Conclusion

Developing a bikesharing system is a complex task. There are different open questions like what kind of lock is needed or which technology should be used. Sharelock was specifically developed for iteratec employees. However, Sharelock should be also customizable for other needs. On the one hand, it should be useable for different purposes. Not only for means of transportation but also for offices, lockers or something else which could be shared. On the other hand, it should be usable for different providers. Examples are companies or universities (on campus).
Certainly, using bikesharing requires usually another smartphone application. This includes typically a new account on a new platform. Routing for public transportation from one point to another is then often difficult or not complete. This is due to the fact that not every means of transportation is integrated in the routing application.
Sharelock is a prototype for a system which can be easily integrated into an open mobility ecosystem. Therefore, a generic REST interface was developed. Staying on the convention of this interface enables the possibility to use Sharelock.
However, the actual system is developed for a lock which is not connected to the internet. Therefore, an extra smartphone application is needed to communicate with the lock (for opening and closing). Anyway, a routing application can integrate the data from Sharelock to show the locations of the bikes. It could use this information while calculating the best route. Just the last step of using the bike has to be executed from Sharelock's smartphone application. This is due to the fact that the messaging logic is programmed on the client side.
Indeed, the feature to open and close the bike from insight a mobility application is possible. Therefor, the lock only has to receive an internet connection. Then the lock can communicate directly with the backend server which sends the commands for opening and closing. This means that the complete logic would be programmed backend side. Commands for the backend server would in that case come from the mobility platform.
All in all, Sharelock is in its current state an executable project. It is deployed in the company (iteratec GmbH) and can be used. However, there are currently no bikes and billing is not integrated. Though, all rides are saved and

can be listed in the administration frontend. Therefore, all customers can be billed manually. Thus only the bikes are missing. This is due to the fact that the development of the physical locks is not finished right now.

6.3. Outlook

There are several features which can be implement into a bikesharing system, but which are out of scope for this thesis. These features can improve the usability or satisfaction of the customer or the provider.
One example is to monitor the state or condition of the bike by for example asking the customer before or after the usage about it. Optionally, the customer should also have the opportunity to give feedback.
Monitoring the battery of the bike lock should always occur in the background after each ride to make sure that the lock will be charged sufficiently early.
There are many possibilities to increase the maintainability. The first one is to add an email notification when a new maintenance is created. In the best case, the email would only be send to the nearest mechanic who can repair the bike quickly. Improving the maintainability for the fleet management can also be done by calculating the best route through all bikes which are blocked for maintenance. Also machine learning techniques could predict when a bike needs maintenance that a mechanic could maintain the bike earlier to reduce the unused time. Another improvement would be to declare a service phone number for every fleet where customers get further help of any kind.
There are also different extensions for the central office. In the administrator frontend are several possibilities to not only show data, but also to change and delete them if the customer had problems.
At the moment each ride is saved with the start and end position. These positions are latitude and longitude which make them difficult to read for humans. Therefore, displaying the locations as addresses would increase the readability enormously.
Actually, every lock represents a bike. However, there is no information about the bike itself. In the future, it makes sense to add information to the lock about its assigned object. Like what kind of bike it is or what color it has. It also can contain images of the object to display them in the frontend or in the smartphone application.

An interesting feature for the provider of Sharelock is a statistic. There are different possibilities what can be measured like how much usages per day or month in total or by bike. Also how much maintenance is needed, which bike was least used, etc. All these parameters can be used to improve the service for the customer, reduce costs or increase the income.
All in all, Sharelock is a runnable system which is deployed. It is stable and can be used. There is an administration frontend and an application for the smartphone to use the system and it has the capability to implement the previously discussed improvements.

Bibliography

[Bia] BiCiBUR: http://www.bicibur.es/. Accessed: 2016-09-11. 2.2.13

[Bib] Bike Sharing Blog: http://bike-sharing.blogspot.de/. Accessed: 2016-05-12. 1

[Bic] Bike Sharing Google Map: https://www.google.com/maps/d/viewer?mid=1UxYw9YrwT_R3SGsktJU3D-2GpMU&hl=en. Accessed: 2016-05-12. 1

[Bid] BikeSharePhiladelphia: http://www.bikesharephiladelphia.org/learn/history/. Accessed: 2016-09-11. 2.2.3

[Bie] BitLock: http://bitlock.co/. Accessed: 2016-09-11. 2.5.3

[Br13] Brooke, J.: *SUS - A quick and dirty usability scale.* Technical report. February 2013. http://uxpajournal.org/wp-content/uploads/pdf/JUS_Brooke_February_2013.pdf; Accessed: 2016-11-03. 5

[Bua] Burgos: http://velo-citta.eu/cities/burgos/. Accessed: 2016-09-11. 2.2.13

[Bub] Burgos Case Study: http://mobility-workspace.eu/geosearch/burgos/. Accessed: 2016-09-11. 2.2.13

[Ca] Call A Bike: https://www.callabike-interaktiv.de/. Accessed: 2016-05-18. 2.2.4

[CC12] Claudio Contardo, Catherine Morency, L.-M. R.: *Balancing a Dynamic Public Bike-Sharing System.* Technical report. CIRRELT - Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport. March 2012. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf; Accessed: 2016-05-12. 1

[CEa] CERN bikes rental: `https://smb-dep.web.cern.ch/en/Mobility/CERN_bikes_rental`. Accessed: 2016-09-11. 2.2.15

[CEb] CERN Velopass: `https://smb-dep.web.cern.ch/en/Mobility/Bike_sharing`. Accessed: 2016-09-11. 2.2.15

[Ch] Chemnitzer Stadtfahrrad: `http://www.chemnitzer-stadtfahrrad.de/`. Accessed: 2016-05-18. 2.2.12

[Cia] CityBikes API: `http://api.citybik.es/v2/`. Accessed: 2016-09-11. 2.5.4

[Cib] CiViTAS: `http://www.civitas.eu/`. Accessed: 2016-09-11. 2.2.13

[Co] Company bike leasing: `http://www.spiegel.de/karriere/dienstrad-statt-dienstwagen-radfahren-mit-steuervorteil-a-974881.html`. Accessed: 2016-09-11. 1.2

[De09] DeMaio, P.: *Bike-sharing: History, Impacts, Models of Provision, and Future. Journal of Public Transportation.* 12(4). 2009. `http://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=1196&context=jpt`; Accessed: 2016-05-12. 2.1

[Frad] Fächerrad: `http://www.faecherrad.de/de/karlsruhe/`. Accessed: 2016-05-18. 2.2.10

[Go] Google Bike Share: `http://www.wired.com/2013/04/google-bikes/`. Accessed: 2016-06-27. 2.2.16

[JC] JCDecaux: `http://www.jcdecaux.com/en/`. Accessed: 2016-05-19. 2.2.3

[Ko] Konrad: `https://konrad.dbcarsharing-buchung.de/kundenbuchung/`. Accessed: 2016-05-18. 2.2.6

[Mea] Melbourne Bike Share: `http://www.melbournebikeshare.com.au/`. Accessed: 2016-08-13. 2.2.14

[meb] metropolradruhr: `http://www.metropolradruhr.de/de/`. Accessed: 2016-05-18. 2.2.7

[MVa] MVG Rad: `https://www.mvg.de/services/mobile-services/mvg-rad.html`. Accessed: 2016-05-20. 2.2.9, 2.4.1

[MVb] MVGmeinRad: `http://www.mvg-mainz.de/mainzigartig-mobil/mit-mvgmeinrad`. Accessed: 2016-05-18. 2.2.11

[ne] nextbike: `http://www.nextbike.net/`. Accessed: 2016-05-11. 2.2.2

[No] NorisBike: `http://www.norisbike.de/de/nuernberg/`. Accessed: 2016-05-18. 2.2.8

[Opa] Open Source Bike Share: `http://opensourcebikeshare.com/`. Accessed: 2016-05-11. 2.2.1

[Opb] OpenBike: `https://openbike.com/`. Accessed: 2016-09-11. 2.5.1

[Ri] RideTap: `http://ridetap.io/`. Accessed: 2016-09-11. 2.5.5

[So] Sotzek, R.: *Development of a Security Solution for a Smart Bike Lock of a Bike-Sharing Project.* not published (Master Thesis at TUM). 4.2.2, 6.1

[St] StadtRAD Hamburg: `http://stadtrad.hamburg.de/kundenbuchung/`. Accessed: 2016-05-18. 2.2.5

[TR11] Tal Raviv, Michal Tzur, I. A. F.: *Static Repositioning in a Bike-Sharing System: Models and Solution Approaches.* Technical report. Industrial Engineering Department. August 2011. `http://nacto.org/wp-content/uploads/2012/02/Static-Repositioning-in-a-Bike-Sharing-System.pdf`; Accessed: 2016-05-12. 1

[Ve] VeloCittà: `http://velo-citta.eu/`. Accessed: 2016-09-11. 2.3

A. Appendix

Firebase Notification Process

Following is a short description of how the user receives a notification by the server with firebase[26]. Figure A.1 displays the procedure with the application server, the smartphone application and the firebase server.
(1) In the beginning sends the mobile application information about the smartphone to the firebase server. (2) Then, it receives a token which is dedicated to this device. (3) Afterwards, the application sends this token to Sharelock's server which saves the token together with the user details. If the application server wants to send information to the user, it searches for the token in the database. (a) Subsequently, it sends the notification together with the token to the firebase server. (b) Firebase sends the notification to the smartphone.

[26] https://firebase.google.com/
[27] Image Source: http://tguerin.github.io/cloud-messaging-gdg-2014/img/data_flow_schema.png

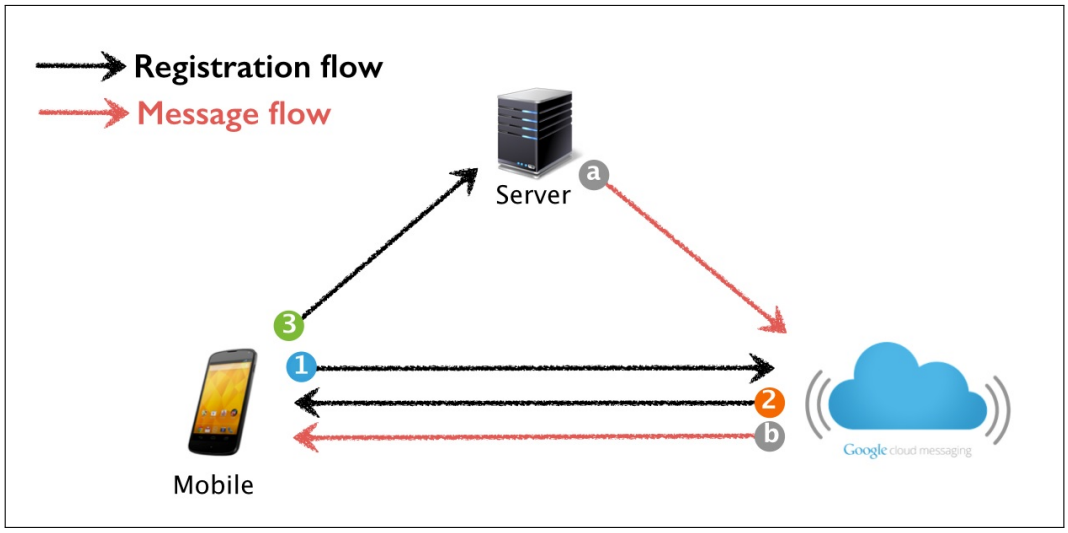


Figure A.1.: Firebase Notification Process[27]. 1) App sends information about smartphone to firebase. 2) App receives token from firebase. This token is dedicated to the smartphone. 3) Application sends the token to the application server. a) Server sends message and token to firebase. b) Firebase sends message to the smartphone which belongs to the token.