IST Shuttle - Booking, management and monitoring service

Duarte Sérgio Ramalho Barbosa

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Teresa Maria Sá Ferreira Vazão Vasques

Examination Committee

Chairperson: Prof. Mário Rui Fonseca dos Santos Gomes
Supervisor: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Members of the Committee: Prof. João Pedro Faria Mendonça Barreto

April 2015

Acknowledgments

Michael J. Fox puts it best: ”Family is not an important thing. It’s everything.”

Abstract

The growing mobility problem between different points in the city and the Instituto Superior Técnico (IST) campi portraits the shuttle service as one of the most important solutions for current difficulties. It also poses new challenges related to ticketing, fraud-evasion and effective cost-wise transportation services.

IST provides its community a free shuttle service between Taguspark and Alameda Campi and other key points of the city ensured by an outsourced vehicle service. This shuttle has several static predefined routes, stops and schedules. All the management is done manually in a tedious and non-optimal way, not allowing the shuttle to effectively surpass its passengers necessities in real-time. Moreover, there are quite often problems with the seat occupation leading to either over or under-crowding of the vehicle.

This work proposes a dynamic and real-time Booking, Management and Monitoring (BMM) service for the IST shuttle service. Members of the community can access, in real-time, the central platform to book a seat or monitor the fleet of vehicles. An embedded system is deployed in each vehicle that grants access to authorized users, giving priority to the ones that have booked seats. This system periodically communicates with the central platform, to inform about the location, speed and seat occupancy in that journey. The BMM service was validated in the field with trial experiments and the results have shown that the system is suitable for being deployed and improves the user experience of the shuttle service.

This project draws its inspiration in empirical experience by the author and other shuttle passengers living these same problems.

Keywords: Mobility management, Real-time Booking system, Shuttle service

Resumo

O crescente problema de mobilidade entre vários pontos da cidade e os campi do IST evidenciam o serviço de shuttle como uma das mais importantes soluções para estas dificuldades. Traz ainda novos desafios relacionados com a emissão de bilhetes, evasão de fraudes e serviços de transporte eficientes no custo.

O IST oferece à sua comunidade um serviço de shuttle gratuito entre os campi do Taguspark e da Alameda e outros pontos da cidade. Este serviço é assegurado por uma transportadora contratada. O shuttle tem diversas rotas, pontos de paragem e horários estáticos. Toda a gestão é feita manualmente num processo entediante e não optimizado, impossibilitando que este serviço supere as necessidades dos seus passageiros em tempo real. Além disso, é frequente haver problemas na ocupação de lugares, levando a veículos sobrelotados ou por vezes, quase vazios.

Este trabalho propõe um serviço de Reservas, Gestão e Monitorização (BMM) dinâmico e em tempo real para o shuttle do IST. Os membros da comunidade poderão aceder, em tempo real, à plataforma central e reservar lugares ou monitorizar o estado dos veículos. Um sistema embebido é instalado em cada veículo e concede o acesso a utilizadores autorizados, dando prioridade aos que tiverem reservado um lugar. Este sistema comunica ainda periodicamente com a plataforma central, informando sobre a sua localização, velocidade e ocupação de lugares naquela carreira.

O serviço BMM foi validado no terreno com testes piloto e os resultados mostram que o sistema se encontra pronto para ser instalado e que melhora a experiência dos utilizadores do shuttle.

Este projecto é fruto da experiência empírica do autor e de outros passageiros com os mesmos problemas.

Palavras-chave: Gestão de mobilidade, Sistemas de Reserva em tempo real, Serviço Shuttle

Contents

List of Figures

List of Tables

List of Acronyms

IST Instituto Superior Técnico

ISIC International Student Identification Card

FOSS Free Open Source Software

JSON JavaScript Object Notation

CC Cartão do Cidadão

QR code Quick Response Code

TOC Table of Contents

ATR Answer to reset

INESC-ID Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento em Lisboa

API Application Programming Interface

e-ID Electronic identity

IAS Identification Authentication Signature

EMV-CAP Europay, MasterCard and Visa - Chip Authentication Program

PIN Personal Identification Number

RFID Radio-frequency Identification

NFC Near Field Communication

UPC Universal Product Code

EAN European Article Number

ATM Automated teller machine

ISO International Organization for Standardization

ICC Integrated Circuit Card

VANET Vehicular ad hoc Network

ST Science and Technology

LEIC Licenciatura em Engenharia Informática e de Computadores

GPS Global Positioning System

OBD-II On-Board Diagnostics, version II

DTC Diagnostic Trouble Codes

EOBD European On Board Diagnostic

YAEC Yet Another Embedded Client

Shuver Shuttle Server

UNIX Uniplexed Information and Computing System

KISS Keep it Simple Stupid

GSM Global System for Mobile

3G Third Generation of Mobile Telecommunications Technology

4G Fourth Generation of Mobile Telecommunications Technology

REST Representational State Transfer

GPIO General Purpose Input/Output

GPRS General packet radio service

ACL Access Control List

DDoS Distributed Denial of Service

GNU Gnu's Not Unix

LINUX Linus' Unix

SoC System On Chip

HTPC Home Theater Personal Computer

RAM Random-Access Memory

RPi Raspberrypi

ARM Acorn RISC Machines

CCID Chip Card Interface Device

PC/SC Personal Computer/Smart Card

FPU Floating-Point Unit

ZTE Zhong Xing Telecommunication Equipment Company Limited

ECU Engine Control Unit

MEAN MongoDB, Express, AngularJS, NodeJS

CGD Caixa Geral de Depósitos

AEIST Associação dos Estudantes do Instituto Superior Técnico

BMM Booking, Management and Monitoring

TP Taguspark

AL Alameda

JAR Java Archive

APDU Application Protocol Data Unit

AT Advanced Technology

TTS Text To Speech

USB Universal Serial Bus

stdin Standard input

SDK Software Development Kit

DIY Do it yourself

IAS Identification, Authentication, Signature

EMV Europay, MasterCard and Visa

CAP Chip Authentication Program

PIN Personal identification number

OEM Original Equipment Manufacturer

DTC Diagnostic Trouble Codes

VM Virtual Machine

ASR Authentication Success Ratio

NAU Number of Authenticated Users

UCR User Conversation Rate

BMMCR BMM platform Conversion Rate

BCR Booking Conversion Rate

BreqSR Booking request Success Rate

GbSR Granted booking Success Rate

ART Authentication Response Time

ICT Information and communications technology

PDF Portable Document Format

CAN Controller Area Network

UTF Universal character set Transformation Format

SHA Secure Hash Algorithm

FCFS First-come, first-served

Chapter 1

Introduction

Technology empowers the World. In an ever-changing Universe, technology is one of the tools materializing human ideas and concepts in more than thoughts or words. It also allows for building knowledge and to solve the problems we face in our daily life.

In the past few decades, there was an enormous evolution in the Information and communications technology (ICT) [1] [2]: cloud computing, low cost embedded systems equipped with sensing devices, GPS and advanced wireless communications are now the basis of the Internet. To face the sustainability challenges that we face nowadays, a lot of new technologies and products are emerging that will define the Internet of the future.

One of the most critical problems in modern societies is mobility and the technological revolution is creating, once more, the means to provide new approaches to the existing problems [3]. Different research and standardization initiatives are emerging in this area and a significant effort to develop new architectures are creating new market opportunities and products [4] [5].

With a growing mobility problem, the shuttle service is one of the most important solutions for current difficulties in moving between Alameda and Taguspark IST *campi*. However, the current service still has several limitations, since IST can not adjust it to the demand and users do not have any guarantee of being transported, specially at rush hours. This thesis aims at providing a contribution to this problem, by providing a **Booking, Management and Monitoring** (BMM) service for the IST Shuttle service.

This chapter presents the basis of this thesis. We briefly analyze the context and the problems we face today in the IST Shuttle service. This chapter also introduces the thesis objectives, expected contributions and a brief overview of our proposal as well. We will finish with the outline for the remaining document.

1.1 Context

Created in 1911, Instituto Superior Técnico is the largest and most reputed school of Engineering, Science and Technology and Architecture in Portugal [6]. The school started its activity in the old facilities of Instituto Industrial e Comercial de Lisboa and in 1937 a new *campus* was built in Alameda with

outstanding conditions to support its mission. Since the beginning, IST has been able to attract the best students and to adapt its curricula and courses portfolio according with the society needs and technology trends.

In the decade of 90's, the Alameda *campus* got overloaded. In fact, although the facilities were periodically renovated, they did not offer all the desirable conditions for state-of-art research and living quality. At the same time, seeking inspiration from the Silicon Valley model, the concept of having an IST campus located in a Science and Technology (ST) park gained momentum in the school.

Hence, the IST Taguspark *campus* opened doors in 2001. Although the main building was not completed (nor the students dorms and research institutes facilities), some professors, staff and about 30 students of Licenciatura em Engenharia Informática e de Computadores (LEIC) moved to the *campus*. Nowadays, IST Taguspark has more than 1700 students, about 50 resident staff members and a large number of professors with teaching or research activity in the *campus*. On the other hand, Taguspark park turned into the headquarters of some of the biggest and most influential Portuguese companies and startups and international ones as well. Besides the regular classes, IST Taguspark also houses part of INESC-ID and MIT-Portugal, where research activities take place. Conferences, talks and presentations from national and international lecturers, professors and influential people are increasingly common in both *campi*. Moreover, some of the courses are taught in both *campi* and/or need access to equipment or facilities which are scattered between them. The fact that, there a large community of people frequently moving between the two *campi* was a new problem that IST had to face.

Mobility is one of the most challenging problems of Taguspark; there are not many public transport services available and the existing ones offer a rather limited service. This problem poses a major limitation on the development of the *campus*, with a strong impact on students' enrollment and professors' and researchers' migration. In fact, most of the time, private transportation is the only mean of access to the *campus* for the school community. Some companies established in the park provide their own private shuttle, but neither the schedules nor the service characteristics are compatible with IST needs or legislation requirements.

The IST Shuttle service was born in 2005 with the single purpose of improving the mobility conditions of IST community to/from the Taguspark *campus*. The shuttle is a free transportation service with several schedules and intermediate stops, connecting other locations of the city (namely train/bus stations) to the Taguspark *campus*. The schedule and routes of the shuttle are set up on a commercial agreement, being defined based on empiric experience and observation. Each vehicle occupancy is manually tracked by the driver that registers in a piece of paper the number of people that enter or leave the vehicle in each stop. Schedule updates occur sporadically, based on the usage occupancy and users requests. Additional services are also performed upon request, mainly when there are tests or exams on Saturday morning. The schedule and routes information is summarized in a Portable Document Format (PDF) document available in the IST website.

The inaccessibility of Taguspark is a problem shared by the majority of the people that work there. IST is not the only institution with this kind of shuttle service. Actually, there are some companies with a quite similar service between locations closed to the ones where IST shuttle operates. It is unfortunate

that there is no way of taking advantage of all these facilities between institutions and to put it to work for the greater good of the community. For the remaining people not served by the shuttle, the only possibilities available are the use of private cars or the scarce transportation service by Vimeca.

1.2 Problem Statement

The Taguspark *campus* is steadily growing and as consequence, the mobility difficulty we face today is reaching new levels. Although the IST Shuttle service has been improving since the beginning, it is not yet able to offer a reliable transportation service. Increasing the number of vehicles or schedules is somehow limited by financial conditions and one needs to find other ways of improving the service.

A recurring problem occurs when there are more passengers than available seats in the shuttle. People who were not able to get on-board are left with two options: wait for the next shuttle (if it exists) or, more commonly, take other means of transportation. The passengers occupy the seats in a First-come, first-served (FCFS) basis. Therefore, there is no way to control who has the most urgency for a seat in the vehicle. If a student needs to use the shuttle for some important reason (as taking an evaluation) and happens to arrive a little bit later, he can be left outside of a vehicle full of people whom are just looking to arrive earlier. On the other hand, sometimes the shuttle leaves with almost no one on board. This portrays a scenario of either over or under-crowding for an expensive service payed by the institution. The current service is not able to dynamically manage the service capacity according to the demand.

Manually tracking the number of passengers per schedule and route does not reflect the needs of the community. It merely tracks the usage of the service in predefined models, albeit empirically optimized. Since its inception, the shuttle management relies on counting the number of transported people per journey as analytics. Unfortunately, this is accomplished in a tedious and error prone way: every time a vehicle leaves, someone from the IST staff counts and registers the number of passengers on the shuttle. Each time the shuttle stops for entry or exit of people, the vehicle's driver will do the same and logs are later crossed and merged. Frequently, these logs are either incomplete or even wrong. Although utopic, the schedules and routes should be dynamic so that the shuttle could better serve its users. Currently, IST lacks the metrics to achieve this. In fact, no information is known whatsoever about the journey or the vehicle. Also, although the shuttle is a service made available only for the IST community and other authorized individuals, there is no enforced authentication policy. Hence, anyone outside this community can use the shuttle with no barriers.

Most of the time, people use other type of transportation before entering the shuttle and need to plan the trip in advance with very limited information. End-users are not able to check the shuttle status and get the answer to: *is it late? do I still have a seat?* or *can I hop in the shuttle in one of the intermediate stops and then continue on it?* On the other hand, managing partners would be interested in information about the vehicle itself: *Is it possible to optimize the route (even with the same intermediate stops)? Are more shuttles required?* In spite of the importance of these questions, the current shuttle service is not able to provide them.

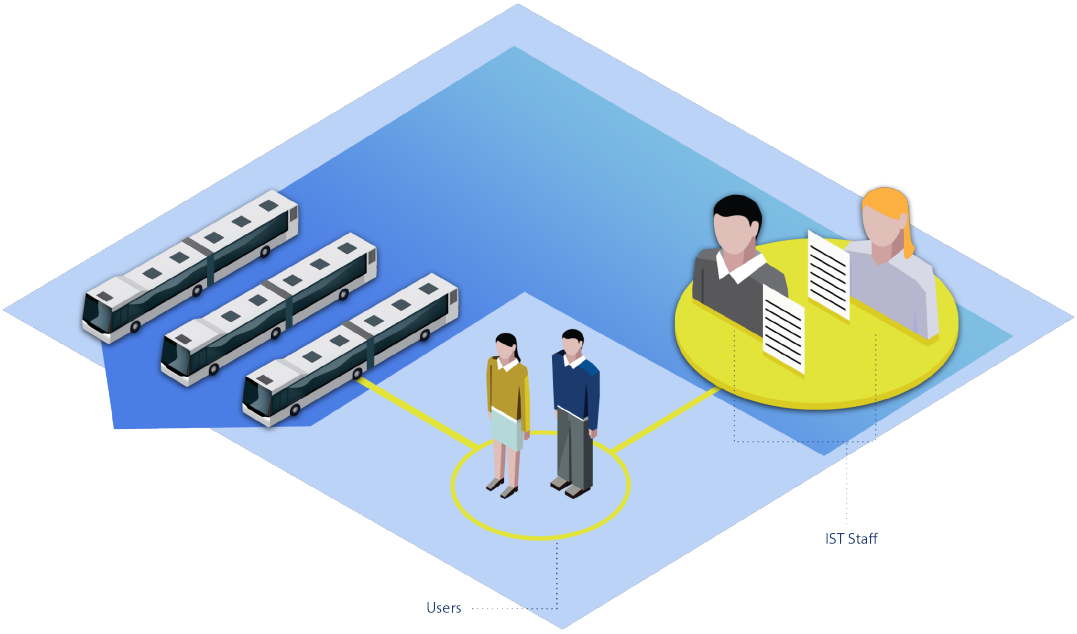

Figure 1.1: IST Shuttle service overview

The overview of the current service is summarized in figure 1.1.

Considering all these problems, the main question that we want to address in this thesis is:

Is it possible to devise a real-time booking system for the IST Shuttle service that is able to support a controlled access to the shuttle for the IST community and guests ?

1.3 Thesis objetives and expected contributions

The goal of this thesis is to design and implement a dynamic access management system for the IST Shuttle service, that provides access for authenticated users, bookings and real-time monitoring of each vehicle. With the introduction of this new system we intend to make the shuttle service more efficient and to provide the tools to manage it in a more optimal way.

The main contributions of this thesis are:

- A low cost and easy to install access management platform that provides all the functions needed to perform user access, booking accounting and gathering and dissemination of real-time vehicles status information. The platform, supported on a commercially available embedded system, will be placed on each shuttle near the front door so that it can be easily accessed by every passenger that enters the vehicle.

- A set of diverse authentication mechanisms to support different operation scenarios. These include contact and contactless cards and QR-codes. For students, professors and staff, this authentication is based on the IST identity card and Fenix credentials. For visitors and external authorized passengers, the authentication is provided through QR-Code. In order to allow the system evolution, other cards are also supported: the national e-ID CC, the old IST identity card and the AEIST student card.

- A complete back-end of the centralized access management platform that provides all the functions needed to manage the fleet of vehicles and the booking procedure and to process each vehicle status information.
- A fair reservation algorithm that takes into account user behavior and needs. Benefits will be given to well-behaved users or the ones that have an important reason to justify an exceptional booking.
- An Application Programming Interface (API) for accessing all the functionalities so that the service might be accessible through a corporate application, namely the Fenix mobile.
- A client so that the service might be used, even without being integrated into Fenix or other corporate application. This integration might happen if the system remains restricted to the IST community.

1.4 Solution overview

The system is composed by three components:

- a centralized system which manages the bookings and the fleet;
- a client application which allows users to book seats and check the shuttle information;
- embedded systems which are deployed in every shuttle and authenticate each passenger and provide information about the vehicle and the journey.

The overview of the proposed solution is summarized in figure 1.2.

All the communication is routed through the central system. Booking requests are made through the client application and managed by the central system. A list of bookings is fetched and stored every day in each of the embedded systems. This list contains booked seats information for each of the journeys and schedules. In the beginning of each journey, the driver pushes a button and it activates the authentication module. From this moment, passengers may enter the vehicle and authenticate themselves, independently of having booked a seat or not. The first users doing this should be the ones that booked a seat.

Periodically, the embedded system sends vehicle and shuttle information to the central system. Any user will be able to get real-time updates of this information by checking the client application. This allows users to decide if they may or not hop into the shuttle given its occupancy rate, the current traffic and the distance from it.

1.5 Outline

This thesis describes the research and work developed and in 6 chapters and 3 annexes. It is organized as follows:

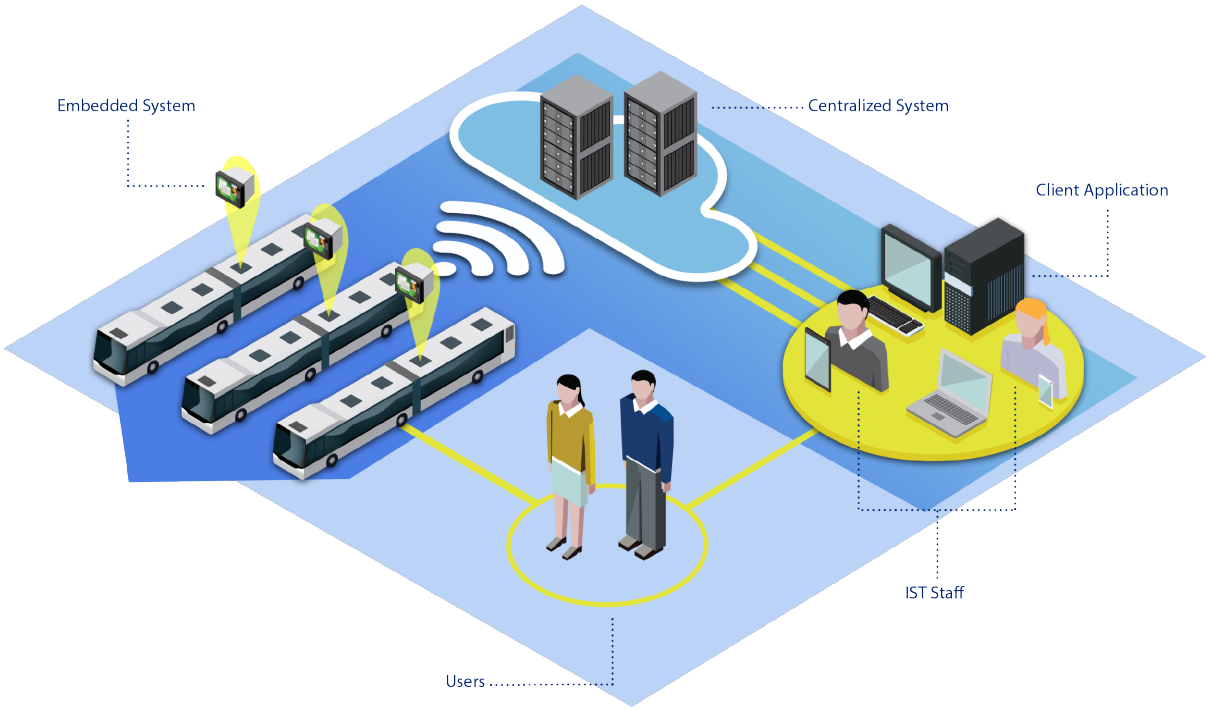

Figure 1.2: IST Shuttle service and BMM service overview

- **Chapter 1** presents the motivation, the problem, objectives and contributions, and overviews the proposed solution.
- **Chapter 2** describes the previous work in the field. Taking into account the most relevant topics of our work, it comprises three main topics: electronic entities, access control systems and vehicle's information. The chapter ends with a synthesis and discussion of the topics that have been presented.
- **Chapter 3** presents the system architecture. It starts by describing the requirements and the general architecture. After that, detailed descriptions of the architecture of the different components and interactions are presented. A final synthesis is provided at the end of the chapter.
- **Chapter 4** describes the implementation of the solution and the chosen technologies. Besides a discussion of the most relevant design options, a detailed description of each one of the components and interactions is presented. The chapter ends with the final synthesis.
- **Chapter 5** describes the evaluation tests performed and the corresponding results. After defining the objectives of the evaluation, two groups of tests are presented. The last section summarizes the chapter.
- **Chapter 6** summarizes the developed work and analyses future work.

Chapter 2

Related Work

This chapter describes the background and related work. It comprises four sections. In section 2.1, we present electronic identities, its characterization and some examples. After that, section 2.2 presents the characteristics of the available services and technologies for implementing access control. In section 2.3, the mechanisms and protocols used to access vehicle information, comprising geo-location and on-board information, are presented. A brief summary is provided at the last section.

2.1 Electronic Identification

Electronic identities are pretty common these days; what once started as a simple way of people communicating with each other using names, e-mail addresses or alias (also know as nicknames) has evolved into useful tools for asserting the real identity of a given person. This goal poses multiple security issues; a palpable effort is being made to fully address this problems and as a consequence, across all Europe Electronic identity (e-ID) cards are being developed and deployed [7]. Some other uses of e-ID include:

- Age verification;
- Personal data (national registry) verification;
- e-voting;
- Secure e-mail;
- Encryption.

e-ID cards offer three forms of information security functionality [8]:

- Identification (I);
- Authentication (A);
- Signature (S).

We are particularly interested in identification and authentication as it will provide the basis of our system. However, this can only be accomplished if the system is capable of verifying that both attributed identity and biographical identity for a given user are the same [9]. The ability to link a set of information to its owner and the effective and secure handling of entity-specific data are essential to numerous different interactions [10].

2.1.1 Identification Documents

Government electronic identities encompasses typically two different methods: electronic passports (contain a contactless chip in the booklet) and electronic ID cards (regular smart cards). Both share the same information; an ID card is personalized with a serial number, a photo and the owner's name and date of birth - like the data page of a passport.

Cartão do Cidadão (CC) The Portuguese e-ID Card is an example of a contact smart card. In the card itself, one can find several ID numbers such as civil identification, taxpayer, social security, health and it also replaces the electoral card. Moreover, it also includes the full name, gender, height, nationality, affiliation, photo and a digitalized signature of the person. Being a smart card with 72KB JavaCard chip [11], it also replicates that same information in the memory. Furthermore, the chip integrates a notepad, the full address of the person and contains also three different applications [12]:

- **Identification, Authentication, Signature (IAS)** - responsible for authentication and digital signing;
- **Europay, MasterCard and Visa (EMV) - Chip Authentication Program (CAP)** - responsible for generating unique token/passwords with alternative channels (as telephone);
- **Match-on-card** - responsible for biometric verification of fingerprints.

Figure 2.1 depicts an example of CC.

A variety of e-government services are available with the card. The card holder has a secret pin code to identify and authenticate himself, and the card generates a legally-binding digital signature for secure declarations and administrative procedures. This smart card also stores a set of private keys that can be used by its holder to authenticate himself electronically, e.g., to prove his identity to a website. The usage of these keys is protected by a Personal identification number (PIN) known as authentication PIN. Besides the authentication PIN, this e-ID card also has another PIN - the address PIN - which simply allows access to the holders address which is stored inside the card. The default address PIN is 0000 on all cards. Another application (Match-on-card), for which access is restricted to forensic and police authorities, will perform identity verification via a fingerprint check.

IST Card IST provides its community a smart card for identification purposes. This card is issued and sponsored by Santander Totta. Beyond offering normal banking functions, it also contains smart card capabilities which makes it particularly suited for our solution. Moreover, this same card is already used

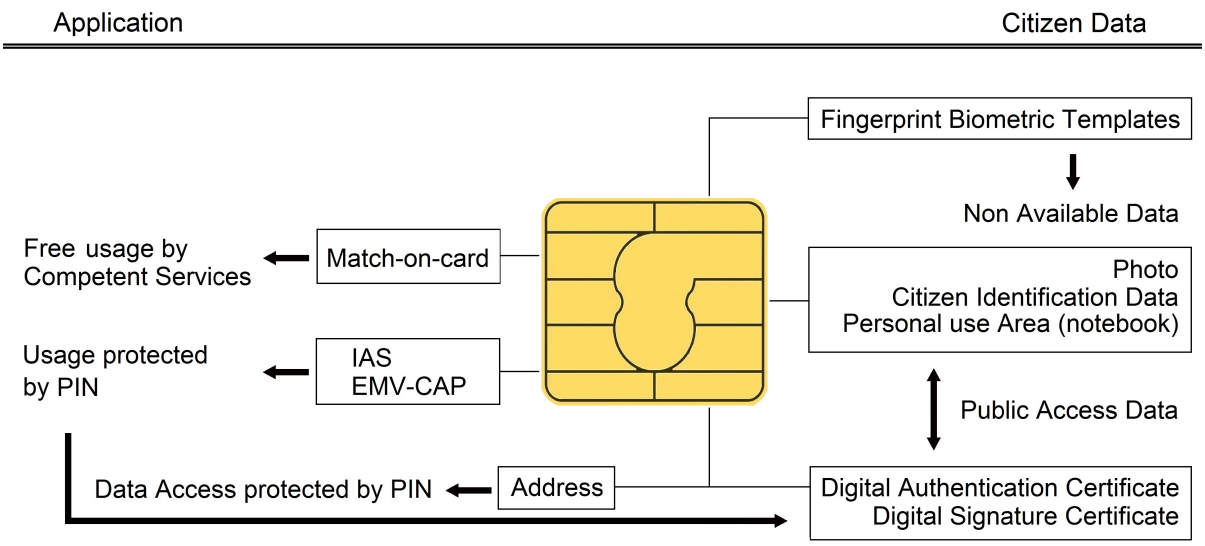

Figure 2.1: Applications and Data of CC

in other areas across the campi (as an example, to enter in the parking lot of the Alameda Campus) and there is a plan to make it support micro-payments and other functionalities as well.

The current card is an iteration of an array of older cards. Different cards were issued by other providers with a distinct shape and number of gold plated pins. This doesn't necessarily translates into incompatibility but the applications are likely different.

An example of different generation IST ID cards is depicted in figure 2.2.

Figure 2.2: Different generation IST Cards

2.2 Access Control

Access control can be defined as protection of system resources against unauthorized access [13].

Although the process of authorization is distinct from that of authentication, authorization presupposes authentication. Whereas authentication is the act of confirming the truth of an attribute of an

entity or piece of data, authorization is the process of specifying access rights to resources.

To perform adequate access control, one must be able to verify a claimed identity and confirm the access rights to that resource. Several methods exist; we will focus on electronic tickets and booking systems.

2.2.1 Electronic Tickets

An electronic or digital ticket is a way to claim some goods or services by means of digitalization of rights. It is a voucher or certificate that guarantees that the owner has the right to claim the services written on the electronic ticket [14].

An electronic ticket system involves two main transactions: issue and verification (consumption) of electronic tickets.

An issue transaction is an action in which the issuer gives ownership of an electronic ticket to the user. A consumption transaction is when an electronic ticket is verified by the service provider which fulfills the service represented by the electronic ticket; the ticket is then consumed by the checking machine. Both security and service quality are equally important for an effective electronic ticket system.

Electronic Tickets are commonly used for access control to public transportations as well as sports or cultural events; they are also massively used in highways or parking-lots because they offer a convenient way of charging for the service to the provider (and issuer of the ticket).

Some of the technologies used for implementing an Electronic Ticket system include barcode, Quick Response Code (QR code), smart card, RFID and NFC [15].

Barcode is a representation of data in one dimension with various parallel lines which width and space between them varies. It was conceived as an optical machine-readable representation.

QR code is a Matrix Barcode which encodes data in two dimensions and has fast readability and greater storage capacity than standard barcodes [16]. It was originally created by Denso-Wave - a subsidiary of the Toyota Group - in 1994 to be used to track parts in the vehicle manufacturing industry. Other 2D Barcode formats exist but QR codes are probably the most popular.

A standard 1D Barcode (UPC/EAN) stores up to 30 digits whereas a QR code can store up to 7,089. In addition, QR code also has error correction capability; Data can be restored even when substantial parts of the code are distorted or damaged. Also, being a two dimensional barcode, it can be read either vertically or horizontally.

Smart card Smart cards are pocket-sized cards (the size of a regular ATM card) with embedded integrated circuits available using either contactless and/or contact interface. ISO refers to them using the term Integrated Circuit Card (ICC) [17]. Contact cards include a gold connector plate with eight contacts although most of the time only six are actually used to communicate with the outside World.

Smart cards are placed in a special reader device for the duration of the transaction (reading from the card, processing, writing back to the card). While in the reader, the card's electrical contacts make

contact with the reader's electric connectors, through which data is read from and written to the card's chip. On the other hand, contactless smart cards [18] are supplied with their operating power by an inductive loop using low frequency electronic magnetic radiation.

They are used to provide identification, authentication, data storage and application processing in various domains; some of those include health, social protection, communication and payment. It is therefore necessary to be sure that there is no possibility of taking advantage of security issues. The application in the smart card shall respond only to requests sent by the authorized user, ensure the service continuity within respecting response time and ensure confidentiality of exchanged data. Our concerns are therefore data integrity, availability of services and confidentiality of information.

The tamper-resistant storage offered by smart cards protects electronic tickets from unauthorized exposure.

Regarding authentication, smart cards are able to provide enough information to prove an individual's personal identification using either token-based or knowledge-based approaches. Token-based systems use items such as a passport or a driver's license whereas knowledge-based rely on memorized information such as PIN numbers or passwords [19].

Smart card based systems can use two types of cryptography:

- **Shared key cryptography** - Provides high-speed authentication. However, the trade-off between the great damage that would affect the entire system if the keys were compromised and the huge cost of key management if each user was assigned a unique key is a great problem;

- **Public/Private key cryptography** - the Integrated Circuit Card (ICC) is equipped with public key mutual authentication and encrypted communication functions. This approach minimizes the damage that can be caused by a compromised key while keeping the key management cost extremely low.

Smart card based electronic ticket systems can prevent user electronic tickets from both malicious users and service managers by using tamper-resistant storage. Secure electronic-ticket systems using smart cards have already been proposed [20].

Radio-frequency Identification (RFID) relates to the use of radio-frequency to identify and track tags attached to objects. Specifically, it is wireless non-contact use of radio-frequency electromagnetic fields to transfer data.

RFIDs can be active, i.e. have power sources and transmitters, or more commonly passive using a reader electromagnetic field for power and communication over a short range [21].

The reader, typically installed somewhere where objects to be tracked pass, emits radio signals. When a RFID tag attached object comes within the range of radio signals emitted by the reader, the tag is activated and it sends the acknowledgement command. The reader captures this radio signal and then, decodes it to a byte stream, and sends the information for further processing to the host system connected to it. This system can be coupled to a smart card which reading process the verification and

validation of the tag will occur by following the protocol of smart card at the communication protocol layer.

They can be used as tokens for authentication, inventory control, electronic purses, anti-theft devices, e-passports, etc. However, since the information of RFID may easily unveiled in the air, there are multiple security and privacy issues. Efforts are being made [22] in order to circumvent this.

Near Field Communication (NFC) builds upon RFID systems by allowing two-way communication between endpoints. It's a set of short-range wireless technologies [23] to establish radio communication by touching devices together or bringing them into proximity. NFC operates at 13.56 MHz at rates ranging from 106 kbit/s to 424 kbit/s. As with proximity card technology, near-field communication uses magnetic induction between two loop antennas located within each other's near field, effectively forming an air-core transformer.

NFC always involves an initiator and a target; the initiator actively generates an RF field that can power a passive target. This enables NFC targets to take very simple form factors such as tags, stickers, key fobs or cards that do not require batteries. NFC peer-to-peer communication is possible, provided both devices are powered.

NFC tags contain data and are typically read-only, but may be rewritable.

There are two modes:

- **Passive communication** - The initiator device provides a carrier field and the target device answers by modulating the existing field. In this mode, the target device may draw its operating power from the initiator-provided electromagnetic field, thus making the target device a transponder.

- **Active communication** - Both initiator and target device communicate by alternately generating their own fields. A device deactivates its RF field while it is waiting for data. In this mode, both devices typically have power supplies.

2.2.2 Booking Systems

Many booking systems have been proposed over the years. Typically, they are developed as a part of a solution to a larger problem as flight/avionics [24], parking lots occupancy or even tied to authorization problems [25]. Our problem also encompasses this same scenario in a different context. Specifically, we take the seat booking as a way to manage the shuttle. In this field of study, we define two different types of booking systems:

Unmanaged booking systems are typical systems where any order will serve its purpose. In this example, a FCFS scenario is obviously unmanaged; the order in the queue is solely implied by the moment in time and space a given person made its way into the queue. The user itself has no meaning as long as the shuttle is completely filled because any order (and consequently any person) will do.

Managed booking systems on the other hand imply that a specific order is given by an algorithm rather than randomly or time based. Typical examples of these systems include the ones used to book a seat in a flight (example given: you may choose the class you want to travel in between a certain price-range; the system will take in account all the constraints of the existing state and the new proposed one to give you the best possible seat available) or the ones used in parking lots (finding the nearest available place for parking a vehicle). Obviously, this type of system is much slower but gives you endless opportunities to tweak the order in the queues.

In flight/airlines case, the booking systems help the customers search the availability and prices for various airlines, seats and destinations. They are the main source of distribution of tickets and therefore need to be highly scalable and reactive; the databases are typically big and updated information is supposed to be shown without any delay. The booking systems are mainly intended for two types of audiences; the customer or end user and the airlines themselves.

2.2.3 Examples

The access control systems can be applied to wide range of different scenarios; some of the applications include transportation systems, parking lots, highways [26] [27] or even in multimodal systems combining different transportation sub-systems.

Transportation systems Transportation systems help mitigate environmental and social problems, providing mobility to disadvantaged groups without access to own transportation means [28]. This poses uniques challenges for each of the different mean of transportation.

Flight systems [29] were designed to provide an efficient and easy way for people to book seats in a very constrained environment. As the performance of this type of system is clearly weak, flight companies started using aggressive marketing strategies leading to all sort of different new problems such as overbooking. These systems generally employ authentication of the ticket by barcode or QR code and an identifying document of the passenger, like a passport. One of the fundamental characteristics is the validation of the entries and booked seats (even if they are automatically issued). You can only buy a ticket to a trip which has not yet begun.

Regarding railroad transports, passengers are usually required to pay a booked seat upfront with their train tickets. The schedules are typically static and periodic.

The bus services on the other hand don't have any special constraint. You buy a ticket, find an empty place (or one you can fit in, not necessarily a seat) and enjoy your trip. Although there is a schedule, typically they suffer from the traffic on the streets and may not be on time [30]. As with trains, verification of the ticket is usually made on the entrance and during the journey.

Parking lots systems Parking lots are cleared areas intended for parking vehicles. Technology has improved the users experience by helping drivers finding unoccupied parking spaces and retrieving their vehicles. Several different use cases were therefore introduced, including indoor positioning systems, adaptive lighting, sensors and mobile payment options.

Smart applications using RFID have been researched [31]. VANET technology has also been used in this context [32]. Finding a vacant parking space is hard and taking advantage of the facilities provided by VANET may help you in that demand.

Intelligent parking services application may help to save fuel and drivers parking time. In some ways, our problem is quite similar to finding a vacant parking space; we need to be able to distribute in real-time and in an efficient way the remaining seats of a given shuttle. Reservation-based smart parking systems have also been proposed [33].

Multimodal systems Multimodal systems are used for storing and traveling within various transportations means. One of those systems is the Lisboa Viva card; a contact and contactless smart card used to store tickets and passes that provide access to public transports in the Lisbon Metropolitan Area. It is based on the Calypso platform [34], a set of technical specifications developed since 1990 by a consortium of European transport operators under the lead of Innovatron company.

Lisbon Viva Cards themselves were only introduced in 1997 but brought a great deal of innovation to the country using for the first time ever smart cards for the ticketing in large scale. In essence a Lisboa Viva smart card is an electronic medium used to store public transport tickets and passes/season tickets (all these will be referred to as contracts). Contracts can be bought using a variety of means: directly at the transport operator, at specialized vending machines or Automated teller machine (ATM) and more recently, on your own PC by visiting Portal VIVA and using a smart card reader.

2.3 Vehicle information

Current vehicles are effectively a master-piece of automotive and mechanic engineering coupled with all sorts of electronic devices. A lot of information can be extracted by querying sensors, actuators and microcontrollers [35]. We are interested in geolocation tracking and onboard information.

2.3.1 Geolocation Tracking

Several civilian usage location tracking systems are available since the 90s but arguably the best and most deployed is the GPS [36] technology. GPS is a space-based satellite navigation system that provides global coverage location and time information in all weather conditions, only hindered by line-of-sight issues. Most of new vehicles can be expected to have a Global Positioning System (GPS) unit and/or communication abilities. This information is very important as it is essential to know the location of the shuttle.

GPS is extensively used in Fleet management, tracking and composing tracks from subsequent vehicle's positions. Furthermore, for driving safety, many VANET applications require the vehicles to periodically broadcast their current position, velocity, acceleration, and direction.

2.3.2 On-board information

On-Board Diagnostics, version II (OBD-II) is a standard interface which gives access to the status of the various vehicle sub-systems relying on self-diagnostic and reporting capabilities. This system provides real-time data and Diagnostic Trouble Codes (DTC) about the vehicle. As of today, the OBD system is an integral element of every vehicle in the market and it's connected with the control system of the engine.

In late 70s, manufacturers started to use electronic means to control engine functions (mainly to meet EPA emission standards) and diagnose engine problems. Through the years on-board diagnostics and control became increasingly more sophisticated and nowadays it provides almost complete engine control and monitoring of the chassis, body and accessory devices.

Controller Area Network (CAN) bus is a message-based protocol designed to allow microcontrollers and other sensors to communicate with each other within a vehicle without a host computer. It was introduced in 1983 by Bosch and greatly simplified what had traditionally been an expensive and difficult problem if something went wrong: the amount of wires required for each feature in a vehicle. Commands are sent out over the CAN network and can be received by anything on the bus, but only used by those that can actually do something with them.

2.4 Synthesis and discussion

This chapter presented the related work. First, the electronic identification and the associated identification documents. A particular emphasis was given to the CC and IST cards, as both of them are used in our work. An interesting thing to remember is that these cards use different type of interfaces, meaning that a multipurpose reader must be considered.

A wide variety of technologies have been explored for electronic tickets: Bar-codes, QR code, contact and contactless smart cards, RFID and NFC. Both types of smart cards might be used for authentication purposes, depending on the identification document that is adopted by IST and will be used by the IST community. However, other types of authentication are also interesting to consider, as IST usually allows people outside its community to use the shuttle, if they were properly authorized to do it. From the above mentioned, QR code or NFC are the most interesting options.

Two different type of booking systems have been defined and studied: unmanaged and managed. Unmanaged booking system use a FCFS approach to reserve seats, while a managed system does this according to a given criteria. Clearly, both approaches have advantages in our case. Unmanaged systems are good for well-behaved users that deserve to know the result of their bookings immediately, while managed systems are more adequate for not so well-behaved users that, for some particular reason, failed to hop in a shuttle where they have a seat reserved. For this type of users, one needs to rank them and attribute the available seats based on that ranking.

Finally, as we want to monitor the shuttle and retrieve information in real-time, we also studied some of techniques used for communication with the vehicle: GPS, OBD-II and CAN bus.

Chapter 3

Architecture

This chapter describes the architecture of our solution. It comprises seven sections. In section 3.1, we start by listing the requirements that were taken into account in the design of the solution. After that, section 3.2 presents the characteristics of the services that are provided to the users. The general overview of the platform components and interactions is described in section 3.2.2. The details of each component are presented in section 3.3, 3.4 and 3.5. In section 3.6, we describe the interactions in the system. The chapter ends with a brief synthesis in section 3.7.

3.1 System requirements

Before analyzing our system requirements, we started by studying the use cases. After that, we performed the requirement analysis. We decided to split the analysis in three main groups:

- Functional requirements that define how is the service going to work;
- Architectural requirements that define how is the software designed;
- Performance requirements that define how is the system organized.

The next sections present them.

3.1.1 Use cases

The IST shuttle service may be used either for access to the BMM platform or for access to a vehicle of the fleet. There are three different stakeholders with distinct use cases. Those are defined by the user role, as illustrated in figure 3.1. Therefore, we have:

- All the IST community can occupy seats and access the BMM platform;
- Other authorized individuals may occupy seats;
- Vehicle providers may access the BMM for fleet management purposes.

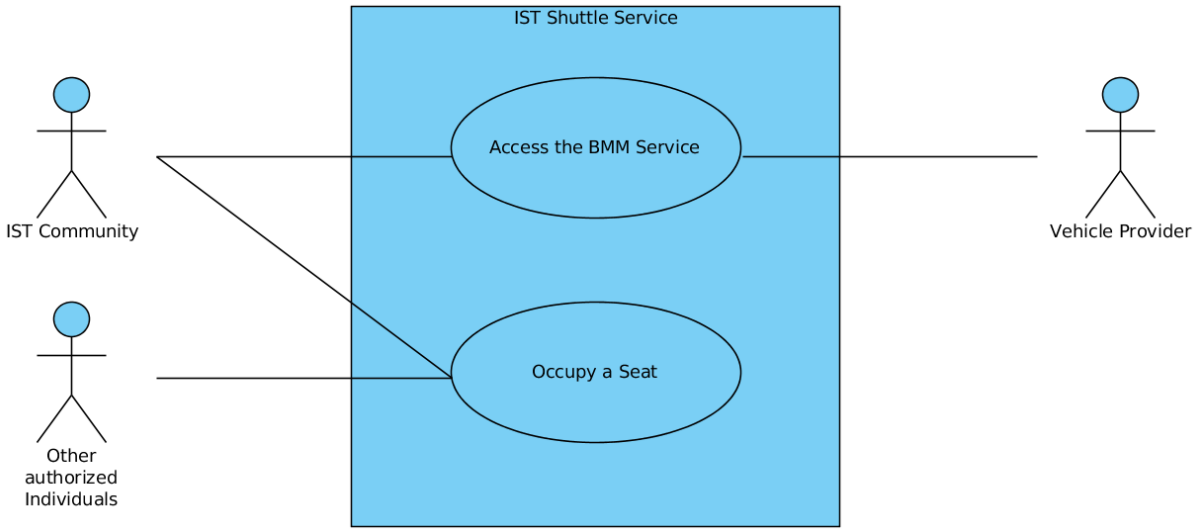

Figure 3.1: Service use cases

On the other hand, the BMM platform functionalities that are offered depending on user role. In particular:

- IST students, professors can book seats and monitor the shuttle status;
- IST staff can book seats for any authorized individual (including themselves) and perform administrative tasks (example given: adjust the number of reserved seats in each journey);
- Vehicle providers can manage the fleet used by the service.

Figure 3.2 illustrates in more detail the functionalities of the BMM platform.

3.1.2 Functional requirements

Regarding functional requirements, the most important aspect devised in the system was the access to the service. Dependent on this, different stakeholders may use the system in different ways and consequently perform distinct actions. One is able to:

Access to the booking, management and monitoring service The service shall provide a booking and monitoring platform for the IST community, granting all authorized users access to the shuttle. These users are the IST community (composed by students, professors and staff) and external individuals previously authorized by IST. The relationship with IST determines the available functionalities of the service. Management is offered only to the IST staff. To access the service, the system that supports it must be easily accessible in all kind of devices connected to the Internet.

Access to the vehicle Authenticating each person that hops in the vehicle requires that a manual or automatic identification verification take place inside (or even before entering) the vehicle. It also requires a transparent and hassle-free method to do this as we do not want to mess up the experience

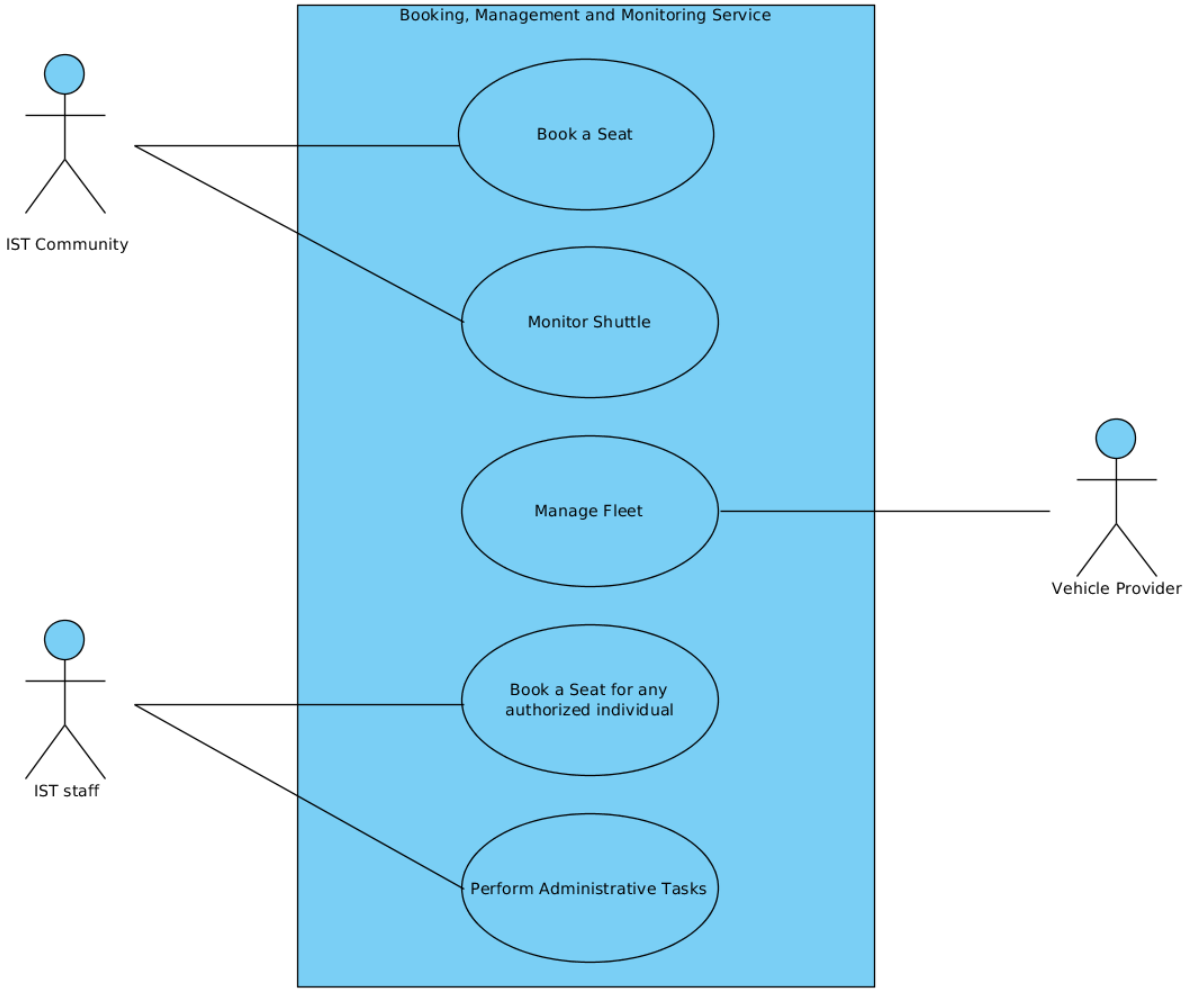

Figure 3.2: Use cases of the Booking, Management and Monitoring Service

people already have. Hence, the system should be able to verify the authenticity of the person and allow or deny them the entrance to the vehicle accordingly.

Booking It would be useful to book seats for the shuttle. If a person absolutely needs to travel between campi, he should be able to reserve a seat in the shuttle for a given day, route and schedule. This concept is similar to those on airlines. Given that IST does not control what vehicles will follow each route and they may have a different number of seated places, it is not possible to allocate a static hard-coded limit for the booked seats. The solution ought therefore to be dynamic to allow different vehicles to do the same task.

Management The service shall provide the IST staff tools to achieve what is currently done manually. Therefore, administrative tasks shall include options regarding the user roles, booking, fleet, schedules and routes. IST Staff shall also be able to book seats on behalf of external users interested in using the service.

Monitoring The service shall provide the IST community tools to monitor and audit the shuttle service. These include information regarding the current journey, the vehicle being used or analytics of the service.

3.1.3 Architectural Requirements

Given the functional requirements we analyzed earlier, the system that implements the service has specific requirements that justify the architecture of the solution. In detail:

Configurability The service shall be parameterizable in order to support its effective management. This includes changing on-the-fly options related to the administration of the service. Also of uttermost importance is the possibility of changing the ”rules of the game”. Most of the policies enforced nowadays in the IST Shuttle service are administration-related. The use of a technical solution to the problems we face today shall not pose a technical limitation or problem in the future. Given that contracts with providers are also finite, the system shall be adaptable to newcomers. Offering a specially targeted system with no configurability support would therefore be against the purpose of said system. The bright side of this is, that this turns the whole system into a standalone booking and management system that may be deployed to other types of vehicles with a complete different set of rules.

Modularity The system needs to have a modular architecture in the sense that each part and functionality shall be decoupled and free of dependencies. In consequence, if in any case one decides to re-implement a module of the system in a slightly different way (let it be programming languages, data structures or algorithms), it ought to abide to the proposed coupling interfaces of the system. The system shall follow the UNIX rule of doing just one thing and doing it correctly, in a KISS principle. This simplifies the system and favors a separation of concerns that allows easy upgrade and re-implementation as well.

Extensibility Taking modularity into account, the system shall also be extensible in a sense that although this system was designed for the IST Shuttle service, there is no reason why it could not be used in any other transportation services as well. Although the main goal of this thesis is to mitigate the problems of the IST Shuttle service , one may argue that in fact those problems are transversal to the whole transportation industry. Moreover, IST is not the only institution crossing the city and various others points in order to bring their passengers. Therefore, this system shall be built considering possible expansions either in modular functionalities or cross utility for third parties. The system shall be extensible not only in a technical way (in which one may decide to implement new features) but also in an administrative way (in which new policies may be enforced or changed).

Portability The access system included in each bus shall also be portable; there are no guarantees regarding the used vehicle or even if the fleet remains the same. One shall be able to move freely this system to new vehicles and it must adapt transparently without hassle.

Security The system shall be able to verify the user identity whenever he accesses the BMM service or hops into the vehicle. Furthermore, as different users have access to a distinct set of functionalities (booking, delegated booking, administration...), authorization shall also be made accordingly to the user role.

Fairness An evil-minded user may exploit the booking service in order to prevent other people to take a seat. Likewise, other users may legitimately book seats and never occupy them. The system shall be fair in the sense that it must prioritize the booking service in order to penalize those who are bad-behaved and do not let others use the service. On the other hand, it shall also allow "good" users to take full advantage of the booking service.

Cost The system shall be as low-cost as possible. It must be inexpensive to deploy and operate. Furthermore, we want others to build upon our system, so it shall also allow inexpensive updates in either functionality or hardware.

3.1.4 Performance Requirements

The scalability and availability of our system are the most important performance requirements. In that regards, each was analyzed.

Scalability The system shall be scalable as we are not able to predict how many vehicles will travel simultaneously in the future nor the number of people accessing the system. This shall not be a bottleneck of the service.

Availability The system shall be fault-tolerant. If any of the nodes can not be reached, this fault alone shall not be considered as non-functional part of the system. On the other hand, the system shall adopt fail-safe methods so it may be possible to recover from faults.

3.2 General architecture

This section discusses the design options of our system and overviews the system architecture.

3.2.1 Design options

In order to meet the requirements and use cases defined earlier, some design options were taken.

The first design option that we faced was related with the type of architecture the system should have: distributed or centralized. By using a decentralized architecture, users may access the BMM platform connecting two different nodes. Given that every shuttle will have a computational node running, we may use this to host and serve our BMM platform. However, making a decentralized architecture for the BMM platform has several problems. Although it is by nature a more resilient design, it also requires an

extra effort for maintaining all the nodes synchronized. Solutions like Bitcoin's Blockchain [37] can be used to achieve this but they also require high computational power and bring communication overload. We also want this system to be low-cost; this is not feasible with a decentralized architecture. Therefore, one of our design options was to conceive a central server which takes the responsibility of keeping all the nodes synchronized and allowing access to its services through an API. Clients implementing this API will authenticate its users and expose the BMM functionalities.

Another important design option relates with the type of access that will be provided for end-users. Two major solutions were foreseen: creating an application that offers all the required functionalities or developing an API that might be accessible to any application developer. Creating an application will allow users to have a complete access to the functionalities of the platform, but providing an API will also allow the integration of the functionalities into existing applications. This seems an interesting possibility if, in the near future, the BMM platform needs to be integrated into Fenix Framework and end-users access Fenix Mobile for its services. Hence, to keep open all the possibilities we decided to provide an API. API Clients may implement the API in any platform of their choice with Internet connection. The central Server will be publicly accessible and answer all the client requests.

The type of equipment that is placed into the vehicles is also another concern. Two different set of possibilities have been studied: a dedicated hardware deployed in every vehicle, or portable and general purpose embedded system that can be easily deployed and removed from the vehicle. Given the fact that providers do not know in advance which vehicle will be allocated to the IST shuttle service, the use of a more flexible solution seems more convenient. Hence, we decided to deploy embedded system inside the shuttle. These nodes will authenticate the passengers that hop into the vehicle and provide a real-time stream of information from the vehicle to the central server using 3G/4G networks. To lower the cost of development, those embedded systems will be commercially available mass-market devices. Only one device will be deployed per vehicle; this is a restriction in the sense that it will only track the people that hop into the vehicle and not those who exit. Consequently, people that leave the vehicle on intermediate stop points will not be taken in account on the calculus of available seats. As such, someone who reserves a seat for the whole journey will be accounted as so even if he leaves before the end. Future work may take this in account and provide a way to track those who exit.

For all the authentication procedure inside the shuttle, the system will behave similarly to other multi-modal public transportation systems. Users will be able to authenticate themselves with a contact or contactless smart card. Non-smart card holders will be able to use a QR code instead as a fall-back option.

Fault-tolerance shall also be one of the main assets of this system. Too many moving gears can get you too many faulty pieces. This is particularly true for full-stack, multi-platform systems like this one. If the central server can not be reached, the embedded client shall be able to authenticate and authorize the users off-line and make decisions based on the available data. Then, whenever the communication is established again, it shall synchronize itself with the server and send usage statistics and the data of the shuttle users. The central server shall then match the data and come up with detailed information about who reserved and took the seat as it would in normal operation condition. This can also happen if the

server died instead for any reason. The embedded client shall be able to log the journey information as long as it shall be necessary until the central server comes alive. If the embedded client fails to execute properly, a more archaic alternative shall be used to balance its need; this can be achieved using today's method of gathering data using pen and paper. In this way, we shall ensure total availability of the service in most scenarios.

3.2.2 System overview

The BMM platform comprises three main modules, as depicted in figure 3.3.

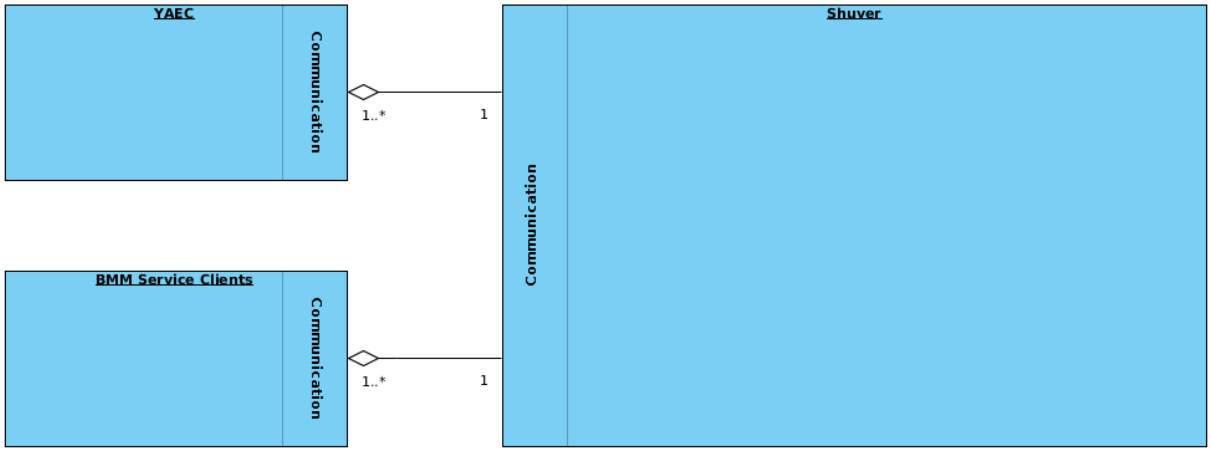

Figure 3.3: System architecture

The embedded system available in each vehicle is called Yet Another Embedded Client (YAEC) and is responsible for the access control upon entering the shuttle and provides vehicle and journey data.

The central node of our architecture is the Shuttle Server (Shuver). It collects and processes all the information flow from YAEC and it is responsible for the heavy lifting behind the BMM logic. Shuver offers an API that allows clients to access it and book seats, change configurations or analyze the data flow from YAEC.

BMM clients present and expose the available services and data. End-users do not have direct access to Shuver but to the BMM clients. This is the public facade of the BMM service and may be implemented in multiple platforms requiring only Internet access.

BMM clients and YAEC interact with Shuver in a client - server architecture. The deployed solution will be composed by several YAECs in the field and one Shuver in a 1-to-Many Architecture. Similarly, several BMM clients may connect to the same Shuver.

3.3 Central system - Shuver

This section describes the architecture of the Shuver. After a brief overview of the architecture, the different functionalities of the platform - booking, management and monitoring - are described. The database and the communication with the other components of the BMM platform are also presented.

3.3.1 Overview

The Shuver is responsible for managing the booking process, maintaining updated information for each route, providing reports of the status of the service and managing options to users and administrators.

This component acts as a sink node; it collects and provides information in real-time to whom requests it. Therefore, all the data must be updated and be available as soon as possible. There are no real-time constraints for the availability of the information; yet, if its severely outdated, it may lose any relevance.

The services and information provided by Shuver are exposed by an API. End-users cannot access it directly; only known clients are authorized to do it.

Figure 3.4 summarizes the architecture of Shuver.

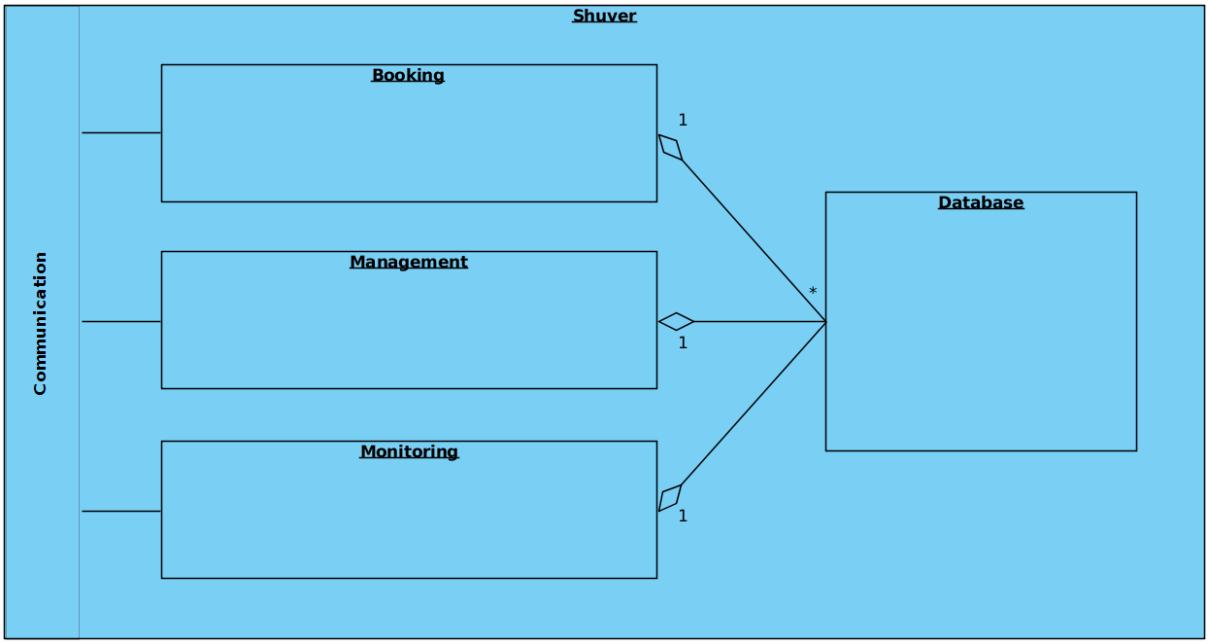

Figure 3.4: Shuver overview

As stated in the figure, the system is organized into five main components:

- **Booking** - allows users to book seats;
- **Management** - provides configurable parameters of the service;
- **Monitoring** - allows users to monitor the service;
- **Database** - stores all the information;
- **Communication** - offers an API and deals with the communication with YAEC and BMM Service Clients.

3.3.2 Booking component

The booking component provides a service that allows a user to book a seat for a given schedule and route. Using the service shall be a rewarding and non disruptive experience. As such, this concept was thought as similar to a airline booking procedure. Booking a seat guarantees that a seat will be available and one is expected to occupy it in order not to be penalized in further bookings. Booking a seat is considered a privilege. Therefore, users shall abide to a set of rules:

- Each booking is personal, non transferable and shall only be made using the BMM platform;
- Users are not allowed to book a set of seats to a group of people. Administrative staff may do this but each seat will be assigned to a person name or identity;
- One is able to book the seat for a given origin, destination and time;
- Booking of seats can not be done in real-time. A parameter in the system will dictate when the booking for each journey ends;
- For each booking a specifically crafted QR code is issued as a one-time-token. This QR code shall be used for access control if the IST Card is for any reason not available;
- The seats will not be assigned. A booking ensures one seat is available, not which one;
- No special enforcement will be made to make sure an evil-minded user would not occupy a booked seat. We trust the good will of the community.

Booking algorithm

We want to provide a useful and fair service to the shuttle users. As of now, the first passengers arriving at the shuttle are the ones that get a seat. Unmanaged systems are - by definition - fair only in the sense they privilege the first to arrive; an evil-minded person may therefore exploit the shuttle in order to prevent other people to take a seat. The proposed solution builds on top of a managed booking system instead, allowing booking of seats for whoever requires it, based on a set of conditions. If anyone exploits this system, he shall be penalized in order to not repeat.

This just implies that the queue would be transported from the location the shuttle stops to the seat in front of a computer where a user would frantically try to book a seat. Therefore, we need a way to prioritize which user will be able to take the seat based on ranking, available places and possibly, an exceptional reason.

Given these requirements, we proposed a new booking algorithm supporting FCFS and Ranking scenarios.

Each vehicle is divided upfront in two distinct parts: one for non-booked seats and another whose purpose is to supply the need of the passengers who booked a seat. Any of these sets may be zero seats; as such, the only seat limit of the booking algorithm is the actual limit of the vehicle.

For the booking procedure, two scenarios are possible. On a first stage, the booked seats are assigned in a first-come-first-served basis. Every user must access the Booking Platform component

```
availableSeats ← value
bookableSeats ← value
```

and confirm his interest in reserving such seat in a given shuttle trip. This procedure shall be completed until the departure day.

Algorithm 1 Book seats

```
function bookSeat(id)
    bookingQueue.append(id)
end function
```

When the occupation rate reaches the limit, the booking algorithm changes slightly, giving priority to whom needs it more. Consider two distinct sets m (users) and n (available seats): if $m <= n$, all the bookings will be considered valid. On the other hand, if by any chance, $m > n$, then only the first n ordered (by ranking and/or time) people of m are able to take the shuttle. In this situation, every one else is put on hold until the booking service closes for a given trip. In the eventual case the system finds a draw in this process, priority is given to the booking with lower booking time.

Algorithm 2 Results

```
function assignSeats
    if bookingQueue.length() > bookableSeats then
        bookingQueue.sortByRank()
    end if
    return list(bookingQueue).first(bookableSeats)
end function
```

Once the booking period ends, the results are considered final and all users are notified of the outcome of this procedure by means of notification of their choice. In the meantime, it will be possible to check the current results by accessing the booking platform which will show the current order for a given trip. One will be able to cancel its booking until the end of the period without any penalization.

Regarding rankings, first time users will receive a medium rank - the initial state - which can be lost (in the case of a booked seat which was not occupied) or even be rewarded (assume a passenger which is very well behaved). The initial ranking may be slightly different though; it will be determined by a bonus given the current statutes of the person in the institution (as an example, one that needs any special help due to sickness or an underprivileged person will be given an increased ranking).

This is a hybrid booking algorithm because it can behave both as unmanaged and as managed solution. The reason for choosing this complex approach relies on the need to offer a fair service to everyone using the shuttle. If the number of persons reserving seats does not surpass the number of available seats, the first-come-first-served method suffices because any order will be accepted as fair (in this case, the most interested users of the shuttle will surely be the first ones to reserve a seat). Otherwise, priority will be given to whom, on the (possibly evil or mistaken) eyes of the system, algorithmically deserves it more and be computed accordingly.

Effectively, users are not booking seats but instead applying for the right to occupy a seat. The algorithm prevents that bad-behaved users use the booking service if demand is high.

Algorithm 3 Ranking

```
rank ← initialRank

function updateRank
    if bookedSeat not occupied then
        rank ← rank − 1
    else if bookedSeat occupied then
        rank ← rank + 1
    end if
end function
```

Extensibility to other providers

Shuver's architecture must allow extensibility to other vehicle providers. Therefore, it makes sense to allow them to manage and perform administrative tasks as well. The system enforces a booking logic with internal (bonus/penalty) and external rules (3rd party passengers). This booking policy is enforced for each provider of the service directly into the system it is managing. The internal rules are devised by each provider and are used strictly into his own vehicles. The external rules are forced upon external users of the service.

Each user and provider are required to abide by these rules and agree on a set of external rules to be followed by every user of the service. Providers are also obliged to report on the misconduct of other party users to their own providers. This is required to properly maintain the service and make it available for free to all interested parties.

For each one of the vehicles, there will be a quota on the available seats each provider wants/is able to provide to other parties. This is enforced only in the booking process. In normal uses, the allocation is still based on a first-come-first-served approach; priority will be given to handicapped users and the provider of the vehicle's users though.

3.3.3 Management component

The management component provides a myriad of options to either administrative staff or vehicle providers and allows them to configure the system for the best of their interests. Configurability is one of the assets of this system; this functionality allows administrators to change the behavior and the policies enforced in the service.

Administrative management

Time has taught us what are the most usual routes and schedules for most of the people traveling between campi. Although the current schedules and routes are "empirically good" and are updated as needed, they are inherently static as they do not change that often. One reason for this is the difficulty of changing a route or schedule and the lack of proper logging and history management.

Currently, there is no way of "modeling" neither the schedules nor the routes to where or whomever needs it the most. It would be important to actively predict the routes and schedules for maximizing the utility of the shuttle service. This can only be achieved allowing the administrative staff to change

the rules and limits in order to optimize the service. Furthermore, the administrative management shall allow new policies to be enforced in the service and notify the user on-the-fly, example given: One will be able to define the number of seats available for booking and as consequence the number of non-bookable seats for each journey. Administrative management will also allow one to define new routes and schedules with ease as well as to impose limits on other parameterizable variables in the system.

Table 3.1 summaries the type of parameters that can be configured.

Parameter name	**Description**
Journey	Create, Read, Update and Delete journeys (routes, schedules, dates)
Initial Rank	Ranking attributed to new users
Rank range	The possible range for ranking levels
Booked seats	Booked seats per journey
Administrative booking	Booking service for every authorized individual

Table 3.1: Shuver's administrative management parameters

Fleet management

Shuver offers a fleet management functionality. The rationale is that once all the fleet becomes known to the system, it may be able to automatically suggest assignments between vehicles, routes and schedules, based on the availability and required resources. Although it would not force any behavior on the providers, it may help to optimize the service. Furthermore, we also collect vehicle specific data which may help to identify beforehand possible problems and alert the vehicle provider. Vehicle providers may use this functionality to add or remove vehicles to their fleet and impose specific rules on them (as the maximum capacity or available schedules).

Table 3.2 summaries the type of parameters that can be configured.

Parameter name	**Description**
Maximum capacity	The number of available seats per vehicle
License plate	Vehicle's license plate
Provider	Vehicle's provider
Availability	Schedule on which vehicle may operate

Table 3.2: Shuver's fleet management parameters

3.3.4 Monitoring component

The monitoring platform component provides detailed and meaningful insights about the service. It is very important to check the shuttle performance and current availability and status. This includes being able to answer: *Is the shuttle on time?* or *Are there any available seats in the current journey?*

Since we have access to the GPS data, we will also be able to locate the shuttle and even predict how long will it take to get to the destination.

Furthermore, vehicle providers may want to check the vehicle information and audit its performance based on the current trip. This is accomplished using the OBD-II data.

Analytics

Although we have logs for current routes, they are not maintained or analyzed whatsoever. There is a growing need for a proper way of keeping track of analytics regarding the IST Shuttle service. The system shall be also an administrative gateway for both providers of the vehicle and the administrative staff of IST. Using the BMM platform, they should be able to perform a full characterization of the fleet comprising the journeys, the schedules and other parameters of the system. This gateway will also be able to offer an insight view of the vehicle, provided that an OBD-II interface is available. With this system, one should also be able to track population movement more accurately and also optimize schedules and routes, maintaining or quite possibly improving the shuttle service.

Although defined at an architectural point of view, analytics is not the goal of this thesis so we will not discuss them in detail. Nevertheless, an optimized schedule, route and occupancy rate may be achieved if one pursues this goal.

3.3.5 Database component

The Shuver database stores all the data that flows through the system. In particular, this information relates to:

- **Users** - The ones that use the booking service (to store the rank);
- **Bookings** - To update and synchronize YAECs and BMM Service clients;
- **Journeys** - To maintain updated records of the available journeys and collect analytics;
- **Fleet** - To assign vehicles to journeys;
- **Service Logging** - To offer a black-box like service.

Once the booking expires, that database entry will be deleted. Since storing all the data that the service generates requires a huge amount of available space, the logs will be rotated periodically if no anomaly is detected. Everything else is stored for service use and later analysis. This does not mean the data will be mined but instead that we want to provide a useful service in the future and this data may be of some relevance. Vehicle data will be sampled; most of it loses importance once it is outdated. Only the Engine Control Unit (ECU) failures may be relevant even if outdated.

3.3.6 Communication component

BMM clients and YAEC interact with Shuver in a client - server architecture. All of Shuver's functionality is exposed by an API and offered with separate and different interfaces for each component.

Shuver exchange messages in a request-response messaging pattern with YAEC and BMM service clients. Once a request is received, Shuver returns a response. Shuver API is publicly accessible by devices with Internet connection; it only accepts connections from verified clients though.

Name	Predicate	Request Parameters	Response Parameters
Status	status	-	status, uptime, connected YAECs
Fleet	create	license plate, capacity, provider	return code, vehicle-id
	delete	vehicle-id	return code
	fleet	-	license plate and capacity per vehicle
Journey	create	schedule, route and vehicle	return code, journey-id
	delete	journey-id	return code
	journey	journey-id	schedule, route and vehicle
	journeys	-	list of journeys
Booking	book	schedule, route, user-id	return code, book-id
	delete	book-id	return code
	booking	book-id	schedule, route, user-id
	bookings	-	list of bookings
	bookingsByUser	user-id	list of bookings by user
	bookingsByJourney	journey-id	list of bookings by journey
YAEC	yaec	-	-

Table 3.3: Shuver's communication API for BMM clients

The functionality offered to BMM clients relates to:

- **Status** - Overall status of the BMM service;
- **Fleet** - Fleet information and management;
- **Journeys** - Journeys information and management;
- **Bookings** - Bookings information and management;

- **YAEC relay** - Relays all the API calls to a YAEC node.

Table 3.3 depicts Shuver's communication API for BMM clients. Please note that YAEC relay calls have different request and response parameters depending on the used predicate; these are described in 3.5.4.

On the other hand, the functionality offered to YAEC clients relates to:

- **Status** - Overall status of the BMM service;

- **Monitoring** - Real-time monitoring information.

Table 3.4 summarizes the Shuver's communication API for YAEC clients.

Name	Predicate	Request Parameters	Response Parameters
Status	status	-	status, uptime, connected YAECs
	update	YAEC-id	list of booked seats for the current day
Monitoring	occupation	list of current passengers	return code
	update	geo-location, vehicle data	-

Table 3.4: Shuver's communication API for YAEC clients

3.4 Booking, Management and Monitoring Service Client

This section describes the architecture of the BMM Service Client. It comprises the overview of the BMM client, the access control logic and the communication overview.

3.4.1 Overview

BMM clients present and expose the available services and data of Shuver. This is accomplished by interacting with Shuver, receiving information and sending requests. Those messages are then translated and presented to the end-user. There are no strict rules regarding implementation or interface; BMM clients may be web and mobile applications or even command line interfaces.

Figure 3.5 summarizes the architecture of the BMM Service clients.

As depicted in the figure, the system is organized into two main components:

- **Access Control** - authenticates and authorizes users;

- **Communication** - communicates with Shuver.

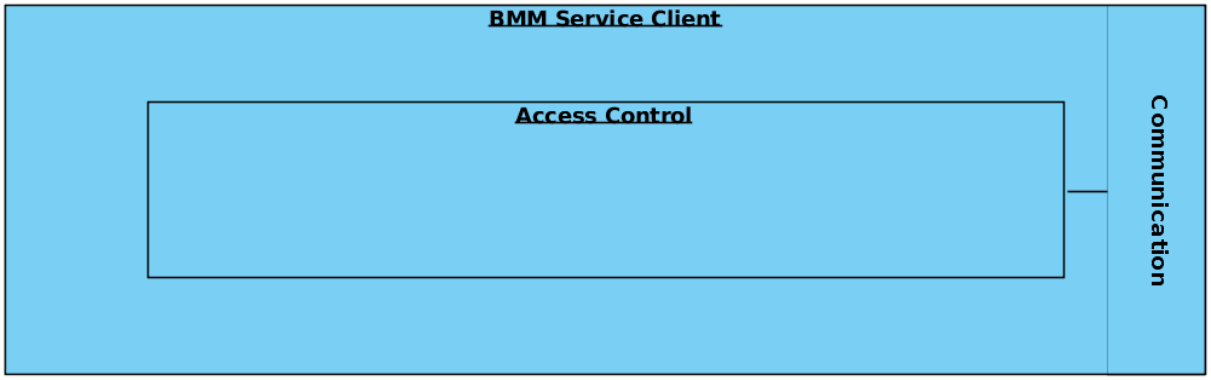

Figure 3.5: BMM Service client overview

3.4.2 Access control component

BMM Clients are one of the interfaces of the service to the end-user. Only the IST community and other authorized individuals are allowed access to these clients and therefore, we need to be able to authenticate and present them the functionalities related to their user role. The authentication mechanism shall allow a seamless use by its users by adopting already existent credentials. The authentication service may be external to the BMM client; the client may delegate this responsibility but relies on it for functioning.

BMM clients do not need to implement any logic per se. Only after authentication will the users be able to access the functionalities provided by the client. These include booking seats, managing the system or monitoring the service. IST staff has access to a bigger set of functionalities; other users have a narrower use case, not being able to manage the system. Vehicle providers are only able to manage the fleet.

3.4.3 Communication component

BMM clients need to be able to communicate with Shuver. This is accomplished by establishing a connection with Shuver and use its communication component to send requests and receive responses. Being its main role, BMM clients are required to strictly consume Shuver's API for implementing their functionality. No other shuttle-related logic shall be implemented in this component.

3.5 Embedded system - YAEC

This section describes the architecture of YAEC. It comprises the overview of YAEC, the access control logic, the data gathering logic and the communication overview.

3.5.1 Overview

YAEC allows efficient authentication and authorization every time someone hops in the shuttle and provides journey and vehicle data. This information can be the occupation data and the location of the shuttle or his route; access to this information may extend the scope of applications and possibilities our underlying technology can provide.

YAEC synchronizes periodically with Shuver, receiving booking information and sending the collected data.

Figure 3.6 summarizes the architecture of YAEC.

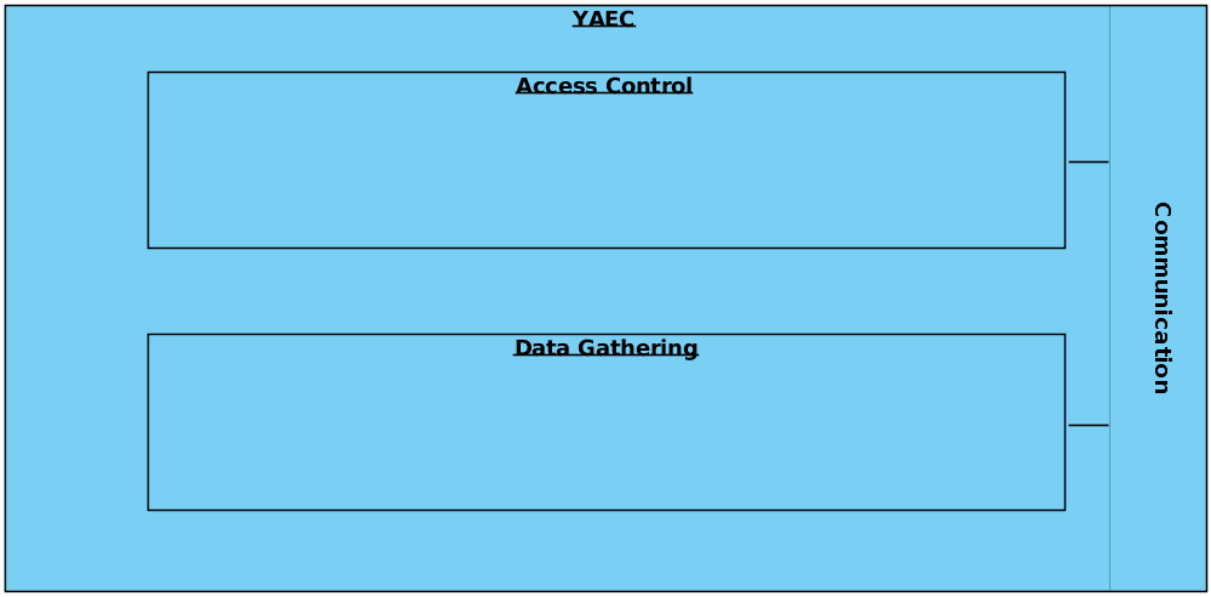

Figure 3.6: YAEC Overview

As depicted in the figure, the system is organized into three main components:

- **Access Control** - authenticates and authorizes the passengers;
- **Data Gathering** - collects data from the vehicle and the journey;
- **Communication** - offers an API and deals with the communication with Shuver and the vehicle.

3.5.2 Access control component

The access control component authenticates and authorizes users of the IST Shuttle service. This is required because this service is only available to the IST community and other authorized individuals. Anyone not contained in this set is therefore strictly forbidden to hop into the shuttle. The authentication procedure shall be as non-intrusive and non-disruptive as possible. This implies that the methods used need to be simple and already known to the user. Therefore, this authentication will be accomplished using one of the following methods:

- Checking a known smart card;

- Presenting a QR code specially crafted by the system.

Consequently, the authentication hardware is composed by two main components:

- Smart card reader (contact and contactless interfaces);
- QR code reader.

Upon the entrance to the vehicle, each user will have to authenticate himself occupy a seat. Assuming a smart card-holder, the user will pass an authorized card by the smart card reader which will extract the relevant data from the card. The ultimate goal is to verify someone's identity; in this context, retrieving the ID of the person identified by the card so we can associate it with something we already know. In IST related cards, the IST ID (already used in all the IST services), will be extracted from the card. However, if such a card is not available (foreign students as an example), that user may also authenticate himself using either an International Student Identification Card (ISIC) or the CC. Although the IST card can be used both as a contact and a contacless card, some only comprise the contact interface. Therefore, one must be able to use both interfaces to query the cards.

The known smart cards are:

- IST card;
- CC;
- ISIC;
- Associação dos Estudantes do Instituto Superior Técnico (AEIST) Card;
- Old IST ID Cards.

The IST shuttle's main users are the IST community. All remaining users are persons allowed by the School to use this service. Occupying seats on the vehicles by third party persons or entities is strictly forbidden. Consequently, it is desirable that an authorization mechanism takes place aboard to prevent abuse of the service by non-allowed users. On the other hand, another problem arises: before being authorized, one shall be able to identify a given person to make sure she is who claims to be. We will address both issues bellow.

Authentication

Every student and person from the IST staff has a IST-ID: a number which uniquely identifies him in the School. The IST application in IST card provides access to a handful of resources about the card holder. One of these resources is the ISD-ID. This is our main asset for providing authentication in this system; every IST card holder can use his card to provide in a seamless way access to this number. Given that the IST card is both a contactless and contact smart card and this information is available in either of the interfaces, the card holder only has to present his card through the card reader before hopping into the vehicle.

Other users (or users without a formal way of identifying themselves as IST personal) pose a new challenge. Unfortunately, it is not uncommon for people to lose access to their cards either because they lost their wallets, their IST card is damaged etc. On the other hand, allowed users of the shuttle may not be tied to IST in any way. Example given: a guest lecturer may want to use the service and be not affiliated with IST. These users will need to authenticate themselves using an alternative way.

Most of Portuguese citizens (quite probably anyone thinking on using this service) already has the not so new national identity card CC. However, the usage of this card poses some problems. As long as the card holder is also a member of IST, we should be able to query Fenix API to retrieve and verify his identity. This requires an explicit authorization on the usage of that binding between IST-ID and e-Identity number, as by the Portuguese law and the Privacy Policy of the Fenix service. On the other hand, if the user is not a member of IST, we must store the e-Identity number which is not legally permitted unless we have an explicit and formal authorization by the Portuguese Government.

International mobility students often have ISIC cards; this card may be used for authentication purposes. This requires registration on the ISIC portal of the card and implementation of client logic to gather specific data from the card and interchange it with ISIC online services.

Algorithm 4 depicts the authentication using smart cards.

Algorithm 4 YAEC Smart card Authentication

```
while waitForCard() do
    if known Answer to reset (ATR) then
        if IST then
            auth()
        else if CC then
            auth()
        else
            authOtherCards()
        end if
    end if
end while
```

Obviously, our authentication scheme only authenticates the smart card itself; an attacker claiming to be the owner of the card will be allowed to enter the bus if a proper identification is not made recurring, example given, to the photo present in the card. This problem has no trivial solution; biometrics could be the answer but an attacker may also take advantage of the false positives to corner the system. Other smart cards might be allowed in the same context if they take in consideration the support of cryptographic functions and offer some sort of Identification of the user. This would not be implemented in this thesis.

To simplify the authentication procedure, all non-IST card holders are treated in a special way. Given that it is hard to make generalizations and assume premises of these users regarding identities, authentication is made instead using a QR code. The goal is to allow the access to the service using a still efficient time-wise mechanism and easily extensible way to any other user. Therefore, to preserve authenticity, a QR code is issued as a one-time-token for each user/journey/schedule. The user will simply have to pass this code (either printed or on the smartphone display for example) through the QR code reader before hopping into the vehicle.

QR codes shall be issued in a timely and secure manner by the service. A QR code *per se* does not offer any kind of authentication or security. An attack can be successfully accomplished by copying the graphical matrix and simply authenticating first. A rightful owner of the seat would not even be able to travel in the shuttle (besides losing the booked seat, this user would not be able to authenticate himself again). Therefore, the matrix shall encode the identity of the user, additional information and a one-time-token issued by the central system.

Authorization

Every IST card holder (obviously, assuming the holder is the one the card identifies) is directly authorized to enter the vehicle.

Every other user requires a formal verification; this procedure ensures that the user once authenticated can effectively hop into the shuttle. Every time a QR code is generated, it provides enough details to identify the journey and the schedule. User information is also provided but YAEC may not be able to identify him: unless a seat has been booked previously and YAEC is aware of this, an extra communication needs to be made with the server to ensure that this user is authorized to travel in the vehicle. YAEC itself has no way of knowing whether a user is allowed or not to use the service. The server needs to implement this logic and store any relevant information.

More granular authorization schemes (like black-lists) may be implemented if necessity arises. If the seat was not booked, a simple verification of identity will suffice.

3.5.3 Data gathering component

YAEC will collect data from multiple sources during the journey. This data is pushed at known time intervals to Shuver. Regarding the vehicle information, all of the OBD-II data that can be fetched from the ECU will be saved in real-time in the permanent memory of YAEC. The idea is to be able to offer a black box of the vehicle. Shuver may decide to discard part of this data but it will remain locally saved nonetheless. These logs can occupy huge sizes so they will be rotated periodically.

Regarding the journey, data points for the location/time pair and all the passengers will be collected. This includes the number and identity of the passengers and location, booked seats and ranking as well.

3.5.4 Communication component

YAEC offers a read-only API. YAEC may be reached from outside networks; this feature was thought for local applications though: It allows debugger clients to directly query the system and whoever accesses it to get specific data from the current journey.

YAEC API follows the same client-server architecture of Shuver with request-response pattern communication.

The functionality offered by YAEC API relates to:

- **Status** - Overall status of YAEC;

- **Vehicle** - Vehicle information;
- **Journeys** - Journeys information.

Table 3.5 summarizes the interface.

Name	Predicate	Request Parameters	Response Parameters
Status	status	-	status, last-sync, license plate
Vehicle	status	-	velocity, duration, ecu-errors, consumption
	{OBD-II parameter}	id	value (not necessarily available)
	geo-location	-	latitude, longitude
Journey	booked	-	list of booked and occupied seats
	occupied	-	list of occupied seats

Table 3.5: YAEC's communication API

3.5.5 Physical/Hardware Constraints

YAEC is the only module of our system with specific hardware constraints. That being said, for implementing the functionalities described earlier, YAEC is composed by four distinct components, as depicted in 3.7:

- **Embedded System** - Executes the YAEC routines and manages system resources;
- **Access Control** - Access Control devices;
- **Communication** - Communication devices;
- **Power Source** - Powers up all the hardware.

The embedded system is composed only by the computational node that runs the YAEC routines and manages all other system resources. It's effectively the processing node.

The access control module is used upon authentication and is composed by:

- **Smart card reader** - used to read smart cards;
- **QR code reader** - used to read QR code.

The communication module is used throughout all the life cycle of YAEC and is composed by:

- **GPS Tracker** - used to determine the geolocation of the vehicle;
- **OBD-II reader** - used to collect data of the vehicle internal systems;

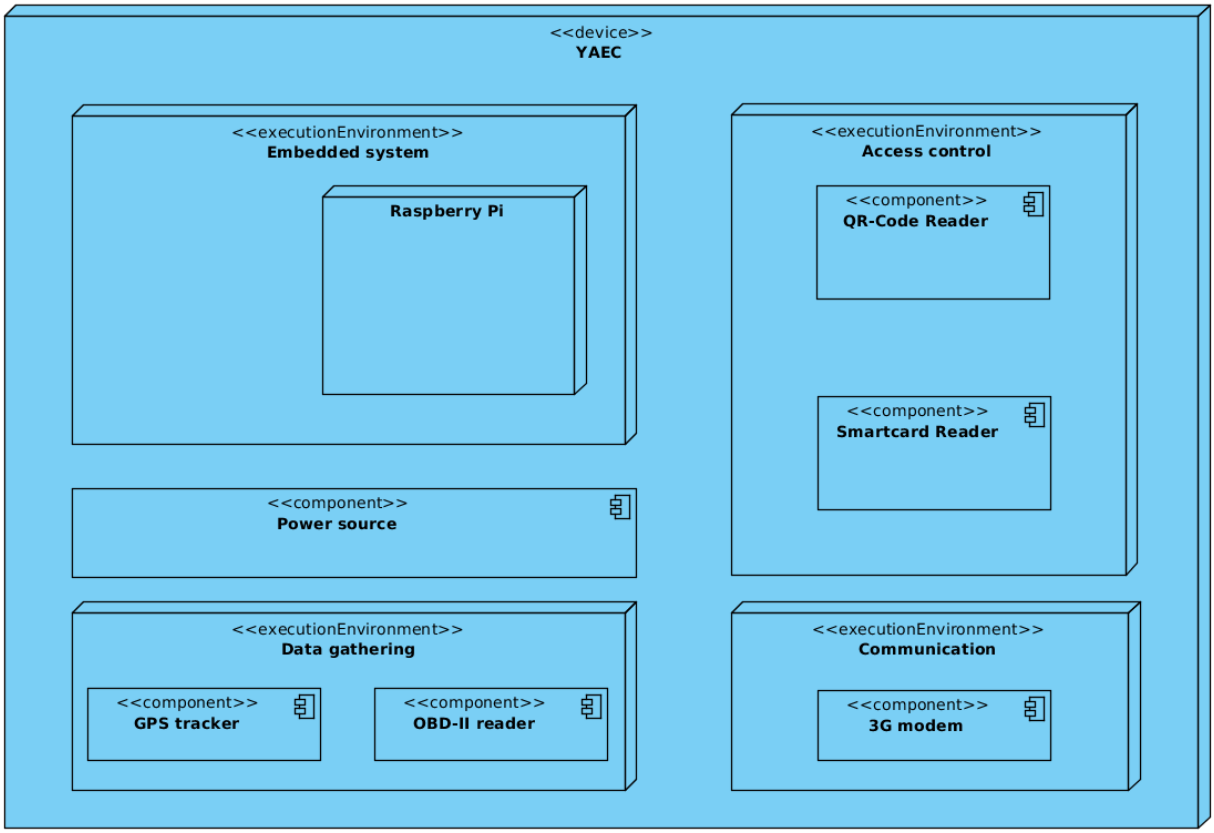

Figure 3.7: YAEC components

- **3G Modem** - used to communicate with Shuver.

All the devices described earlier need to be feed some current to operate; that's exactly the purpose of the power source.

3.6 Module interactions

This section details the communication between the different modules of the system. As the BMM service clients cannot access YAECs directly, interactions only comprise the BMM service clients and Shuver; and the YAEC and Shuver. All the messages interchanged between components of the system are in JavaScript Object Notation (JSON).

3.6.1 BMM service clients and Shuver

Prior to being able to access Shuver's services, BMM service clients need to authenticate the user. This is done requesting the (possibly external) authentication service to identify the user.

Once the users are authenticated, the BMM service clients are able to connect to Shuver and fetch the status of the service. From this moment on, the services are available to the end-user.

Booking, management and monitoring all follow the same pattern: BMM clients issue requests to Shuver and its responses are shown to the user. There are no middlemen in this case; the user is

already authenticated and the client is known to Shuver.

The messages interchanged between the BMM Service clients and Shuver are depicted in Figure 3.8.

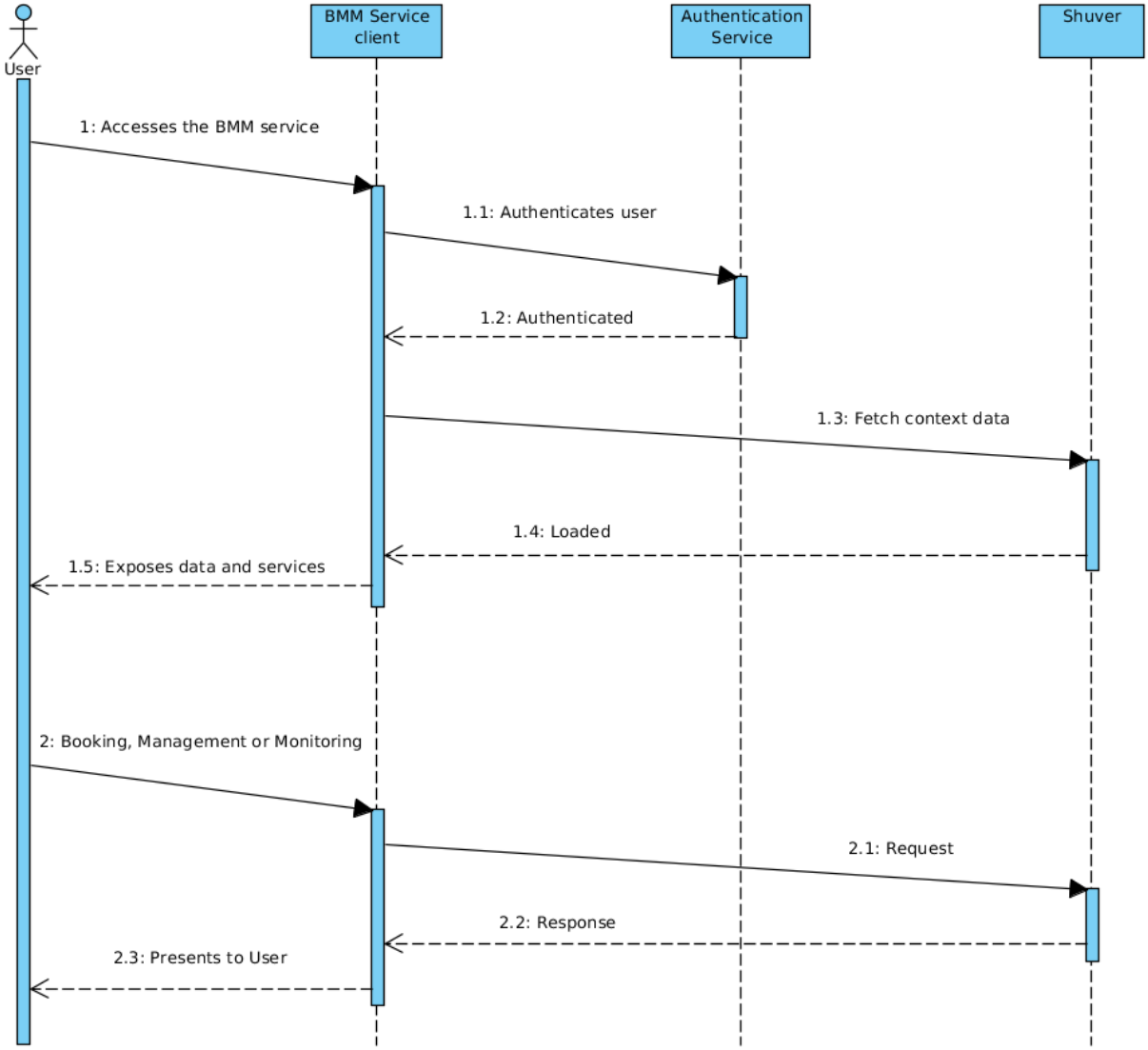

Figure 3.8: Messages between BMM service clients and the Shuver

3.6.2 YAEC and Shuver

Once YAEC is powered on, it connects to Shuver and retrieves an updated list of all the booked seats for that node and journey. This request takes an arbitrary large time depending on the number of booked seats and current YAEC state. As soon as the shuttle closes its doors and starts the journey, all the passengers have already been authenticated. YAEC is now able to notify Shuver who were the users that hop into the shuttle and the ones that occupied booked seats; YAEC will upload this list to Shuver and once it is acknowledged, it will be considered in a sane state. YAEC also sends monitoring updates periodically. These have a fixed size and are sent with the same Δt time interval.

It is important to note that delivery is not guaranteed. YAEC tries to synchronize itself to a sane state and once it successfully accomplishes this, it just broadcasts the monitoring messages.

Figure 3.9 summarizes the messages and their order between the BMM Service clients and Shuver.

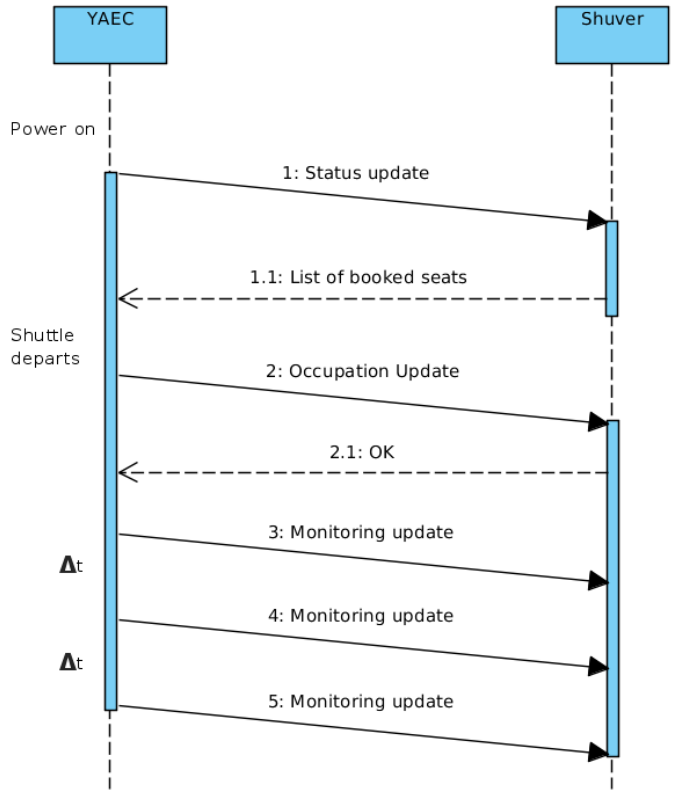

Figure 3.9: Messages between Shuver and YAEC

3.7 Synthesis

In this chapter we defined the architecture of our system. We propose an expansion of the current shuttle service to accommodate three extra components: a central server named Shuver, an embedded system per vehicle called YAEC and the specifications for the clients of the new BMM platform.

The BMM service extends the existing Shuttle service by allowing the IST community and other authorized individuals to book seats, manage different options regarding the configuration of the service and monitor the stats of the shuttle and the vehicle. Users do not have to access the BMM platform though; there are no changes for the current service except for the need to authenticate each time one hops into the vehicle. This is required to ensure that only authorized individuals use the service. All the rest is provided as a completely non-intrusive and non-disruptive experience.

Chapter 4

Implementation

This chapter describes the implementation of our solution. It comprises four sections. In section 4.1, we start by listing the implementation options of our solution. After that, section 4.1.2 presents the characteristics of the services provided in the prototype and its implementation details. The chapter ends with a brief synthesis in section 4.6.

4.1 Implementation Options

Implementation forces us to evaluate our decisions and make necessary compromises. In this section, we will present and explain our choices for each of the components - YAEC, Shuver and BMM Client - in both Software and Hardware areas and evaluate them. For each, we will state what were the available options and why we ended up choosing one or the other. It is important to note that given the full-stack nature of this solution, there were way more options than what we were able to fit in this document.

4.1.1 Hardware

On the hardware side, the only critical component in our system is the YAEC. Being an embedded system, much of the implementation decisions we made took in account physical, power and efficiency constraints. However, nowadays mainstream embedded systems are so capable that it would be inadequate not to put those resources in use. Therefore, we looked for upgradability options as well. On the other hand, both Shuver and BMM clients are able to run in normal computation devices with Internet connection. Past efforts in massifying cloud integration led to a generalized use of these solutions to run this type of systems; both Shuver and BMM clients take advantage of this by using hardware in the cloud.

Embedded System

Nowadays, with the advent of Do it yourself (DIY) movement and wearable computing paradigms, a myriad of highly advanced and inexpensive embedded systems (and its components) flooded the market.

One good example of this is the 35 USD Raspberry Pi (model B)[1]. This small computer packs enough juice for running a complete GNU/Linux distribution and it is quite inexpensive to maintain drawing less than 1 watt on idle. Besides USB, Ethernet and a couple of video-related ports, it also provides various GPIO pins which can be used to extend its functionality. Coupled to a Power Supply Unit or small battery, one is able to deploy an inexpensive and capable device for continuous use.

Similarly to the Raspberry devices, there are a couple of other embedded systems in the market with different specifications, namely: combination of SoC, memory or expansion capabilities. Of those, and keeping the Chinese knock-off clones out of the list, the most popular are the media-oriented ODROIDs and similar, and DIY-platforms such as Beaglebone or Arduino-based devices. ODROIDs and the like are extremely well known for their use in HTPC builds. They provide a more capable SoC and RAM but cost much more. It is overkill to use this kind of system if no media-related functionalities are required; it draws considerably more power and the extra performance may not be noticeable. On the other hand, Arduino-based devices are well known for their extremely low prices but the hardware itself is quite underpowered. In fact, Arduino-based devices come loaded with ATmega microcontrollers instead of a CPU and are therefore not suitable for running a full GNU/Linux distribution (either because of the instruction set or raw processing power). Given that the full implementation of device drivers, network protocols and a basic operating system are out of the scope of this system, Arduino-based devices are simply not feasible. Beaglebone Black[2] offers roughly the same as the RPi albeit with upgraded specifications and a higher price target. Given that IST already had RPis available for use by its students and professors, the choice was relatively easy. Table 4.1 summarizes the different options that were studied. ODROID X2[3] and Arduino Uno[4] were used for comparison.

Specifications / Device	**Raspberry Pi (B)**	**Beaglebone Black**	**ODROID X2**	**Arduino Uno**
CPU (or μcontroller)	ARM11 (v6) @ 700 MHz	ARM Cortex-A8 @ 1 GHz	4x ARM Cortex-A9 @ 1.7 GHz	ATmega328 @ 16 MHz
RAM	512 MB SDRAM	512 MB DDR3	2 GB DDR2	2 KB
Expandability (GPIO pins / USB ports)	17 / 2	69 / 1	50 / 6	26 / 0
OS Support	GNU / Linux	GNU / Linux	Android or Ubuntu	-
Power (idle)	700 mA	460 mA	1 A	50 mA
Cost	35 USD	45 USD	89 USD	25 USD

Table 4.1: Comparison between embedded systems

[1] http://www.raspberrypi.org/products/model-b/
[2] http://beagleboard.org/BLACK
[3] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G135235611947
[4] http://arduino.cc/en/main/arduinoBoardUno

Access Control Hardware

YAEC authenticates passengers using either smart cards or QR codes. Therefore, we need to couple a smart card and QR code readers to the embedded system. For the smart card reader, contact and contactless interfaces are both interesting options. The reason is that although the IST card supports both interfaces and we would not be using the CC anytime soon (due to legislation issues), we do not know whether a future card will be allowed for authentication. Moreover, supporting various different interfaces ensures some redundant fail-safe measure.

The market has lots of either contact or contactless readers but devices that implement both interfaces are not that common. Furthermore, any devices that we end up using needs also to have specific drivers for proper and correct execution. As usual in most of commercial manufacturers, only proprietary drivers are available. Obviously, those drivers are invariably compiled for popular architectures such as x86 as amd64 and without reverse engineering, we can not use the binary blobs in ARMv6 devices (the architecture of the RPi). Turns out that there is a free open-source driver for Chip Card Interface Device (CCID) (also called CCID) which is able to ”talk” with a number of different smart card readers. We are left with a search for a contact/contactless smart card reader supported for GNU/Linux ARMv6-based systems using CCID.

To ensure both the IST and future cards are read, we opted out for an Identive Cloud 4700F [38]. The manufacturer made available one free sample device that besides offering these interfaces, also comes with support for NFC Technologies. Both Proprietary and open-source (CCID) drivers are available. Furthermore, the reader can be normally queried using the de facto standard PC/SC implementation in Linux, pcscd. This device was chosen because it supports all the technologies we were aiming and actually allows us to work with new ones as they become supported in the system. The hardware is more capable than the software and that is a good thing because it ensures upgradability.

Regarding the QR code scanner, standard cameras can be used to accomplish this. The greatest problem is that they do not process the QR code but only the image. This offloads part of the computation to YAEC and image-related stuff is traditionally hard and expensive to process (even worse in such low powered embedded systems). Furthermore, standard cameras do not provide good resolution and taking photos and analyze them afterwards may take a long time. One of the requirements is also that it is sturdy so it can withstand the shocks and the wear. Cameras are not known for their sturdiness. A dedicated solution is desirable in this case. Once again, the same problem of smart card readers drivers arises. This led us to chose the Motorola DS9208. It is well supported out of the box and ticks all the requirements we stated before.

Both authentication devices are coupled to the embedded system with the USB ports.

Communication Hardware

YAEC communicates periodically with Shuver using a 3G link. Hardware-wise, it involves coupling a 3G modem with the Processing Unit. There are no big requirements regarding this device; the Linux kernel supports out of the box most of this kind of modems. Taking this into account, we choose an old ZTE

MF620 modem to perform tests and they all worked flawlessly. Regarding the communication with the vehicle, an OEM OBD-II reader dongle (based on the ELM327 chip) is coupled with the OBD-II port of the vehicle. This dongle is powered directly by the port and supports Bluetooth in order to query the ECU remotely. Obviously, the embedded system needs to have a Bluetooth link to communicate with the OBD-II adapter; this led us to also couple an OEM Bluetooth adapter to the system. Both the modem and the Bluetooth adapter are USB devices.

Other Hardware

Since the access control and communication hardware is coupled to the embedded system using the USB ports, we need a hub for extra available ports. However, all these devices draw a lot of extra current from the embedded system which may change its lifetime and/or even cause availability failures. With this in mind, we decided to deploy a powered USB hub to connect all the hardware. This allows all the devices to be powered by the hub and the embedded system to deal only with the data stream.

Regarding the power supply unit, the obvious solution is to use the cigarette lighter socket. We deployed a standard OEM charger with 2 USB ports. The charger itself is capable of providing 2A at 5V (maximum), which is enough to power the embedded system on one and the hub on the other. This poses a problem though; the cigarette lighter socket may only be receiving current if the vehicle's engine is turned on. This can cause failure or unavailability of the service. Future work shall take this into account and provide a battery for continuous use.

4.1.2 Software

All the components of the BMM system contain specially crafted software. Most of the software we ended up using is Free Open Source Software (FOSS) which, in some licenses may require it to be used only with other FOSS.

YAEC

Although not critical, YAEC performance shall not be a reason for users abandoning the system. This means that, although the hardware can keep up with some load, the used software should take in account the constraints due to the low performance embedded nature of YAEC. Software is obviously dependent on whatever the hardware choices were.

Operating Systems There are multiple Operating Systems supported either officially or non-officially (mostly ported experiments) for Raspberry Pi. We chose a well known GNU/Linux distribution: Raspbian, an ARMv6 optimized version of Debian specially compiled to Raspberry Pi platform. Raspbian has gained a lot of attention and is also one of the officially supported and recommended Operating Systems by the Raspberry Pi Foundation [5].

[5] http://www.raspberrypi.org/documentation/raspbian/README.md

Programming Languages Regarding Programming Languages, the choices are relatively straightforward. System programming languages as the name implies are particularly well suited for this kind of work. We need to find a balance between performance and suitability. Python is well supported in Debian and as a Programming Language is well suited to this.

Personal Computer/Smart Card (PC/SC) implements a C API for direct querying of smart cards and several wrappers and bindings exist for other languages. One of those is Python, namely pyscard. Pyscard builds on top of a number of useful abstractions and although it does not perform like its C counterpart, the query time is still manageable for the authentication of the users. In fact, the smart card operations are much faster than the entrance of passengers in the vehicle.

We still require the system to be scalable and sufficiently fast so that users do not end up not using it. YAEC may need to query a lot of low-level details from the smart cards and it inherently takes time. The C Programing Language makes itself really suitable for this use-case so it was chosen to implement most of the low-level functionality in the embedded side. Everything else is coded directly in Python; it does not make sense to implement a web server in C nor not to use wrappers for time-consuming and error-prone tasks that are easily achieved using higher level language. Furthermore, pyscard implements pcscd smart card API with automatic error correction.

Shuver

Operating Systems Any operating system with a working install of Python is enough for executing Shuver. In this case, we chose the GNU / Linux distribution Debian.

Programming Languages Although performance is a good aim, we do not need to be lightning fast and managing the databases can take advantage of a higher-level language. Given that our constraints are fairly low and it provides a seamless transition, the choice for implementation was again Python, specifically version 2.7.

Database There are a multitude of different databases (and types) in the market. Of those, the two most interesting options are relational (SQL) and document-based databases. One of the implementation options was to use the same format for every message in the system and that it should be able to scale horizontally. A non-sql database was used to allow horizontal scaling. In architectural terms, this is quite feasible because the data is balanced and scattered between various documents. Since all the messages interchanged in the system are JSON, a good candidate for database is MongoDB; it favors JSON-like documents as schemas and allows messages to either of the systems to be well understood (and parsed) and saves some work when building and routing complex messages.

BMM service client

Operating Systems Any operating system with a working install of NodeJS and MongoDB is enough for executing the BMM service client prototype. To provide a seamless transition, we chose again the GNU / Linux distribution Debian.

Programming Languages The BMM client was developed in Javascript (ECMAScript 5) on both client and server-side. Javascript is the de facto standard programming language for web applications. It grew in popularity in the last couple of years outshining other relevant technologies such as PHP or Ruby. NodeJS allows you to use Javascript in both client and server-side and has seen a rising number of projects being built on top of it. It gave us confidence it would be an interesting experiment and we got to develop our skills in yet another programming language.

4.2 Prototype - Shuver

This section describes the implementation of Shuver. Shuver implements the BMM logic and deals with the communication and information synchronization in the BMM service.

4.2.1 Overview

Shuver is the central node of our architecture and offers an API that exposes access to the BMM functionalities.

Shuver was implemented in Python 2.7 and runs as a standalone process which spawns a thread for each of the BMM components. The database is external though; a MongoDB instance is launched upon system boot and we access it directly using a custom driver and helper wrapper.

Shuver exposes its services as a publicly available RESTful API. Clients that want to consume this API, need only to connect to Shuver and send GET and POST requests to the desired endpoints. More details on the communication can be found in section 4.2.6. Shuver runs as a headless server; there are no GUI's to describe. Shuver only accepts requests to its endpoints from known Clients. As such, any Client must be registered in Fenix and its keys need to be manually introduced in Shuver. Any user that wants to access Shuver services needs also to authorize the client to access his own personal information inside Fenix. Once everything is set up, the client will be able to send requests to Shuver.

Shuver lives at http://shuttle.tecnico.ulisboa.pt:8080.

4.2.2 Booking

The booking functionality allows users to book seats for specific journeys. Booking requests are issued by BMM clients and include:

- the identity of the booker;
- the date;
- the journey.

To request a booking, the following message is sent as a payload of a POST request to /booking/create:

```
{
        "user": "",
        "date": "",
        "journey": ""
}
```

Upon receiving a booking request, Shuver verifies if the users ranking is below the booking threshold rank (the minimal rank that one is able to book seats); in that case, it will deny the booking and alert the user.

On the other hand, if the ranking is equal or higher, the user is allowed to apply for a booking (even if he later loses it). Shuver accomplishes this by inserting into the database an entry with the request information. A response is then returned with a 200-OK with an optional payload describing the current status of that booking as:

- **Attributed** - if the booking is guaranteed;
- **Pending** - if the algorithm cannot determine yet the availability of the booking;
- **Denied** - if the booking is not allowed.

The default ranking is 3 out of 5 possible levels.

The data taken in account for the booking process is the IST-ID, the date for the booking and the journey (which includes the route and schedule). Each of these parameters is encoded as an Universal character set Transformation Format (UTF)-8 string and concatenated in the order given earlier. This information is then hashed with Secure Hash Algorithm (SHA)-256. This same hash is what is encoded in the QR code for authenticating upon entering the shuttle. The database encodes the timestamp for the booking process automatically as each of the booking entries is added. This is used when evaluating draws in the booking algorithm.

It's not possible to remove or change bookings; that functionality has not been implemented.

4.2.3 Management

The management functionality allows system administrators (IST Staff) to change and configure different parameters related to the service. In particular, it allows one to:

- Manage journeys (including routes and schedules);
- Manage fleet;
- Book seats for other individuals (delegated booking).

For each of these functionalities, different endpoints were created. Upon receiving requests to those same endpoints, Shuver will access the database and perform the required actions. More details on the available endpoints are present in appendix A.

Journey Management

To create or remove journeys, Shuver inserts or removes relevant information in the database. This information is received from the BMM client in JSON format and contains all the required data for this operation. Shuver validates this information ensuring that there are no conflicts in this operation (as an example, creating a journey which collides with others previously made).

For the creation operation, the following message is sent as a payload of a POST request to /journey/create:

```
{
        "vehicle": "",
        "schedule": "",
        "stop-points": ["", "", "", ...],
        "available-for-booking": ""
}
```

If this operation proves successful, Shuver will answer the request with a 200-OK with no payload and the journey will be automatically available for the remaining functionalities. In any other case, a error will be thrown and sent.

On the other hand, if one wants to remove a journey, a POST request to /journey/destroy is sent. This request contains the $journeyID$ which identifies the journey.

It's important to note that removing journeys may leave the system in an inconsistent state. No guarantees are made regarding the bookings made for journeys that no longer exist. Future work may take this in account and also provide an update functionality.

Regarding the stop points, the available ones are hardcoded. The reason is that those do not change that often (since it's a contractually-agreed term) and it did not impose any limitation in the current service. On the other hand, there are no default schedules and one is required to input a desired one upon create a new journey.

Fleet Management

The fleet management functionality allows one to create and remove vehicles from the available fleet. These vehicles are later assigned to journeys and one can correlate them with the vehicle data. This functionality is analogue to the management of journeys.

To add a new vehicle to the fleet, the following message is sent as a payload of a POST request to /fleet/create:

```
{
        "license-plate": "",
        "capacity": "",
        "provider": ""
}
```

Delegated booking

The delegated booking functionality allows IST staff to create a booking on behalf of some other user. This is particularly important for non-IST community since they lack the access to the BMM service.

On the implementation side, this is similar to a regular booking. The only difference being that instead of the $user$ field identifying an IST community member, it contains instead the e-mail address of the foreign user. After booking, an e-mail with a QR code is sent to this address. The request this time is made to /management/booking/create.

4.2.4 Monitoring

Shuver provides a monitoring functionality accessible by the monitoring endpoint. In the implemented version, no location or vehicle data is available from YAEC. Consequently, the monitoring functionalities of Shuver are mostly a skeleton for further development, offering just the occupation data. This information includes a list of everyone onboard and periodic updates.

Shuver discards most of the received data; the only important information are the bookings that were effectively occupied. After the algorithm assignment of new rankings, this data can be safely removed.

Requests to its endpoints return the most updated available information about the journey and the vehicle. As an example, a GET request to /monitor will return:

```
[{
        "yaec": "",
        "occupation": ""
},
...
]
```

4.2.5 Database

The database encodes all the information as JSON schemas. There is nothing relevant about the implementation of the database. More information on the particular schemas is available in appendix C.

4.2.6 Communication

Shuver exposes a publicly available Representational State Transfer (REST) API to which each of the YAECs and BMM clients directly communicate.

Webpy[6] was used to implement the REST API. Webpy was created by the late Aaron Swartz and makes it easy to semantically expose a REST API in Python. To define an endpoint, one must create a Python class and assign it to a string representing the path of the endpoint. Then, for each HTTP method one wants to implement in that endpoint, a class function with the same name must be created.

[6] http://webpy.org/

Each HTTP request to that endpoint is translated to a class function bearing the same name as the method. If one wants to deploy a Web Server which answers GET requests made to IP/booking, one must create a ”booking” class and a class method named GET returning the response. This is exactly what is implemented in both Shuver and YAEC APIs.

An example GET method for the /shuttle/location endpoint looks like this:

```
urls = (
        '/shuttle/location', 'location'
)
class location:
        def GET(self):
                return json.dumps({"lat": self.lat, "lon": self.lon})
```

The full API endpoints and an example request and response for each are available in Appendix A.

4.3 Prototype - BMM Client

This section describes the implementation of the BMM service client, the interface of the BMM service to the end-user.

4.3.1 Overview

The BMM client was developed as a Web (Single Page) Application. The goal was to build a fluid experience akin to the ones most users are already used to. AngularJS allows one to declarative program the DOM. From a myriad of different web frameworks, the MongoDB, Express, AngularJS, NodeJS (MEAN) stack was used because it was conceived exactly for this kind of application. As of February 2015, this is a popular stack for responsive web development. Since we are not building native applications but rather a responsive web application, this stack abstract lots of required work. Although we do not actually need MongoDB running in the prototype, it was originally used as the DB of the system and remains as a useful leftover (it allows us to verify requests prior to being sent to Shuver).

The BMM clients run completely in client-side and thus, do not store any user data. An user is forced to authenticate in Fenix and all consequent data exchanges are performed just to identify the user (between the BMM platform and Fenix Framework).

The BMM client lives at http://shuttle.tecnico.ulisboa.pt/

On implementation, no special caution was taken in the usability or design of the solution as it is mostly out of the scope of this thesis.

4.3.2 Booking

The booking functionality allows one to book a seat for a specific journey. In the prototype client, this was implemented creating an endpoint (/book) which points to a different web page. In this view, a

user is required to input the date, the schedule, the entry and the exit stop point. Keep in mind that the identification of the user is already known per the Fenix credentials and is thus automatically injected in the request. The user must then click on the green ”Book Seat” button and the request will be sent to Shuver. Upon confirmation, the status code is returned and shown to the user. At the same time, the page is redirected to home, showing all current bookings.

Figure 4.1 depicts an actual screenshot of the booking functionality.

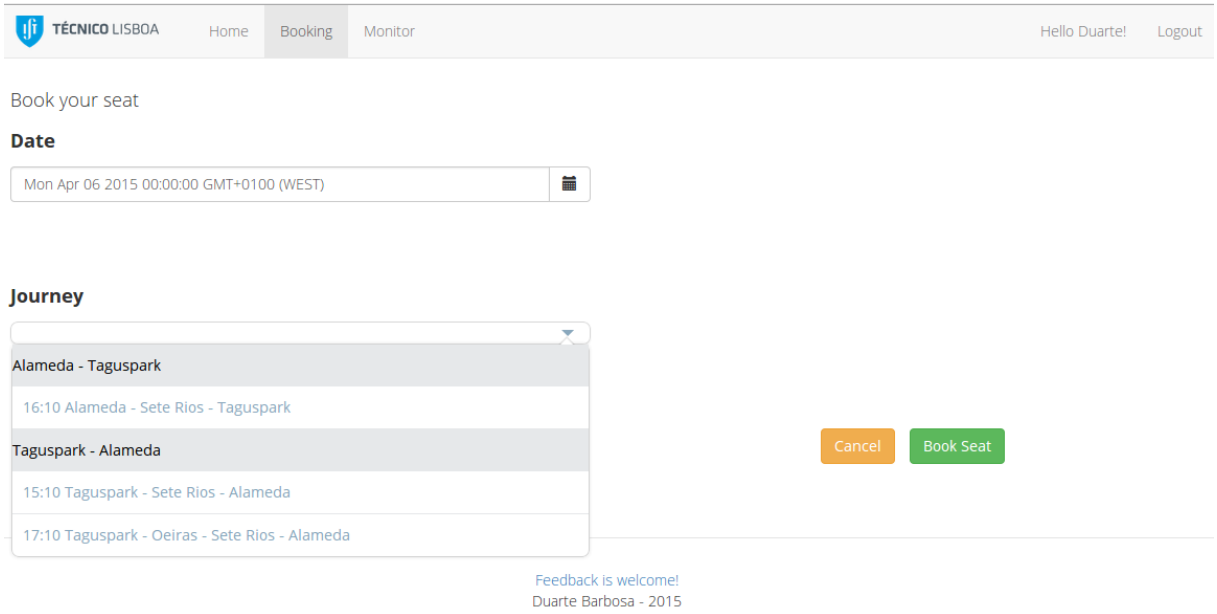

Figure 4.1: Booking area

4.3.3 Management

The management functionality allows IST staff to manage important parameters of the system. This was implemented by creating an endpoint (/admin) which shows the user a different web page. This page contains three buttons which link to the Journey Management, Fleet Management and Booking for other individuals functionalities.

Since the only implemented authentication model was Fenix credentials, we do not support external parties to access the system. Therefore, vehicle providers are not able to manage the fleet themselves. Only the IST Staff will be able to do it.

One small caveat: since this functionality is not considered stable or user-friendly yet, this administrative view will only appear to selected peers (hardcoded).

Journey management

Upon accessing the journey management functionality one is able to create or delete journeys.

To create a journey, one is required to input the schedule and select a route from a set of predefined ones. Journeys are assumed to be daily; this is a limitation of the system in order to simplify the logic

(which translates to the current service).

The journey update involves a small hack; in fact, there is no update functionality implemented in Shuver. Upon updating, the BMM client will instead delete the previous journey and add a new one with the new fields. This implies that previous bookings in that journey would not ever be honoured and shall be made again in the new journey.

Figure 4.2 depicts an actual screenshot of the journey management functionality.

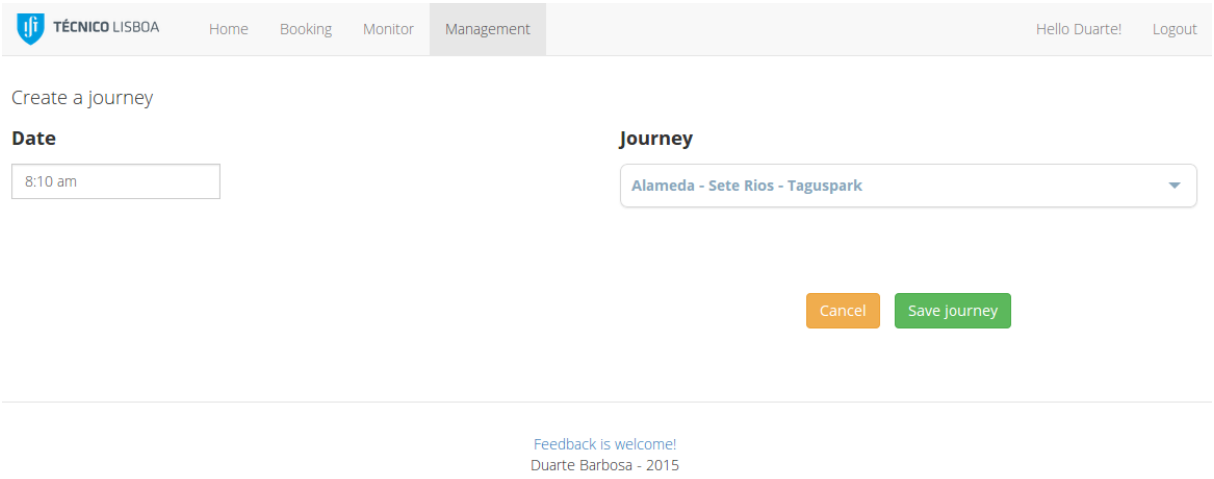

Figure 4.2: Journey management area

Fleet management

This functionality is similar to the journey management in look and feel. The only difference is the data that is exchanged. In a nutshell, to create and delete vehicles, the procedure is exactly the same. This time, you are required to input vehicle details though. The required parameters are the license-plate, the capacity and the provider of the vehicle.

Booking for other individuals

The booking functionality is virtually identical to the one offered to normal users. The only difference is an extra text-box where an e-mail address is required. This identifies the user that requested the booking. After a successful booking, the corresponding QR code is sent to this e-mail address.

Any bookings made in this way are considered priority ones and are therefore requested with maximum ranking. Please note that the insertion in the database is made in the exact same way as normal bookings. Since the booking algorithm takes the insertion timestamps in account in case of a draw, this implies that bookings made by the administrative staff may still be left outside of the vehicle in the extreme scenario of it being fulfilled with other maximum rank users first.

4.3.4 Monitoring

The monitoring functionality allows one to acknowledge current vehicle position and the occupation ratio. This is accomplished by requesting Shuver information of the current operating shuttles and their journeys. Please keep in mind that this information is not always available or updated. In fact, this service operates in a best-effort mode and it is conditioned by various other external variables.

In this view, normal users are presented a map with the coordinates of the vehicle and the number of empty seats. Administrators may also view current vehicle data collected by the OBD-II component.

Figure 4.3 depicts an actual screenshot of the monitoring area.

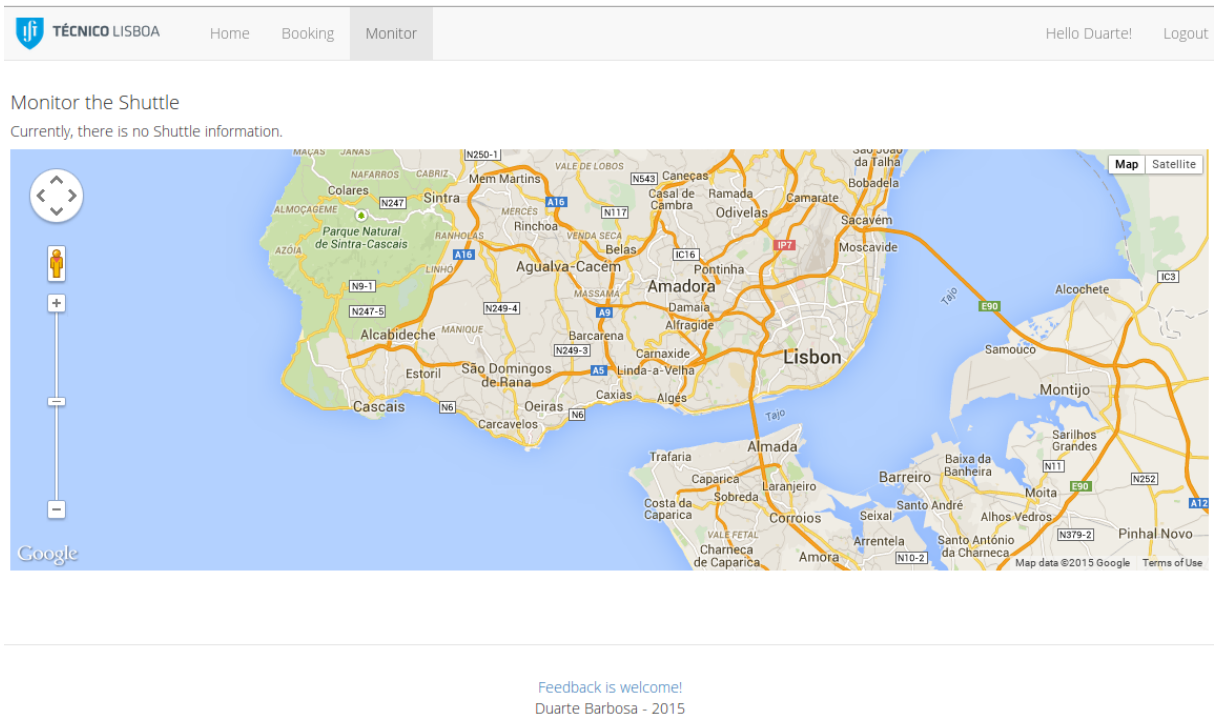

Figure 4.3: Monitoring area

4.3.5 Access Control

The access control in the BMM clients is required to ensure that no unauthorized individuals access the service.

The chosen authentication mechanism were Fenix credentials. Fenix framework integration comes with a cost: people that lack Fenix credentials (and are still authorized to use the shuttle) are not able to access the service for themselves.

Effective authentication is accomplished by authenticating first with Fenix services and using their API to access the desired functionalities (namely, retrieving the IST-ID).

OAuth has emerged as a popular mechanism for delegated authentication. Fenix implements the protocol allowing one to use those credentials in other applications. OAuth integration implies that some messages are exchanged between Fenix and the BMM client prior to being able to access the service.

Also, it requires an extra effort to maintain the session and ensure no data is leaked.

The BMM service clients are identified by an ID and secret, provided by Fenix upon registering the application access to its API. Since the available SDKs did not match the programming language of the BMM client, we were forced to implement by ourselves the OAuth authentication flow. This requires redirecting the user to Fenix, letting him authenticate himself and authorizing the application, redirecting to a callback URL of the BMM client and from now on, request and extend the access token as required.

The initial version did this purely on client side. The problem is that this exposes the application keys and may be used to forge an elicit application on a phishing attack. All the OAuth flow was then implemented in server side. Given that we are using NodeJS and there is a vibrant community of developers, we searched for modules implementing OAuth strategies. Passport [7] implements an array of different authentication methods including OAuth. After some implementation tries, we found a proof-of-concept Fenix strategy from a fellow IST hacker [8]. After minor changes, the module was working as expected and the authentication using Fenix credentials was complete and successful.

Session secrets are stored server-side using Mongoose[9] in the BMM client mongodb instance.

Partial implementation of authentication with Google, Facebook and Twitter exists; it is based on the same OAuth protocol but it's not supported as currently there is no way of proving an identity using those credentials. Later, if the service gets extended to other providers (such as the ones that also have vehicles transporting people in the Taguspark), a dedicated authentication platform, external to Fenix and even including these services, may be implemented.

4.3.6 Communication

BMM service clients communicate with Shuver using standard HTTP REST methods. All the requests are implemented using AngularJS's core $http service (and consequently its promises API).

The $http service provides easy communication with remote HTTP servers via the browser's XMLHttpRequest object or JSONP. It generates an HTTP request and returns a promise with two $http specific methods: success and error. This promise can be used to register callbacks methods, receiving an object representing the response.

An example POST request for a booking looks like:

```
service.book = function(data){
        var request = $http.post('/booking/create', data);
        var deferred = $q.defer();
        deferred.resolve(request);
        return deferred.promise;
};
```

All the requests sent by the BMM client are well formed to be inserted in the database. This is a leftover characteristic of the initial implementation of Shuver and BMM client in a single piece of software.

[7] http://passportjs.org/
[8] https://github.com/nunofmaia/passport-fenixedu
[9] http://mongoosejs.com/

4.4 Prototype - YAEC

This section describes the implementation of YAEC, the embedded system available in each vehicle.

4.4.1 Overview

YAEC ensures an efficient access control every time someone hops in the shuttle and provides journey data. YAEC was developed as an embedded system to be deployed at the entrance of the shuttle. Upon entering, users authenticate themselves using either the official IST card or a specially crafted QR code. YAEC ensures that booked seats are available and manages the data stream to Shuver, requesting the list of booked seats and notifying Shuver of the current passengers as well.

YAEC was implement in Python 2.7 and includes code in C and bash scripting. Makefiles are available for building and deploying the system.

YAEC initial process spawns a different thread for each module. All the synchronization is made using a message queue that each of the modules listens too.

Figure 4.4 depicts one of the first assembled prototypes of YAEC. Figure 4.5 depicts the actual physical configuration of YAEC.

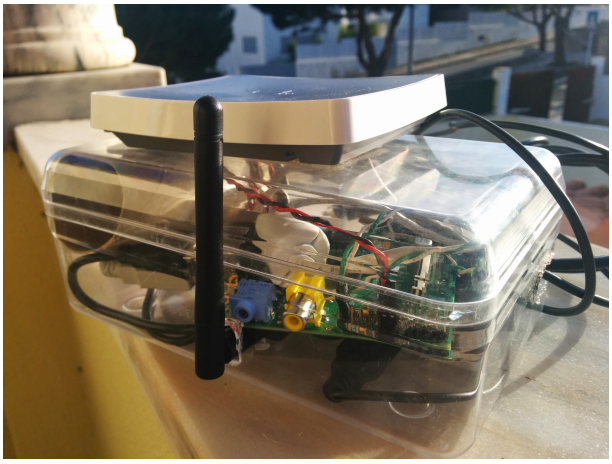

Figure 4.4: YAEC Prototype

4.4.2 Access Control

Although the smart card reader manufacturer provides proprietary driver blobs for x86 and amd64 architectures, the source-code is not available. The choice for the reader took in account the support list of the open-source driver, CCID. However, Raspbian repositories have no binary available for CCID, so we compiled directly the source from the original subversion repository. As the time of writing, the available version of pcscd in the Raspbian Repositories is not the latest available. Since the smart card reader driver was specifically compiled for the Raspberry Pi, we went ahead and compiled the whole smart card-related stack to ensure all the versions are compatible between them. This required the installation of several new development packages.

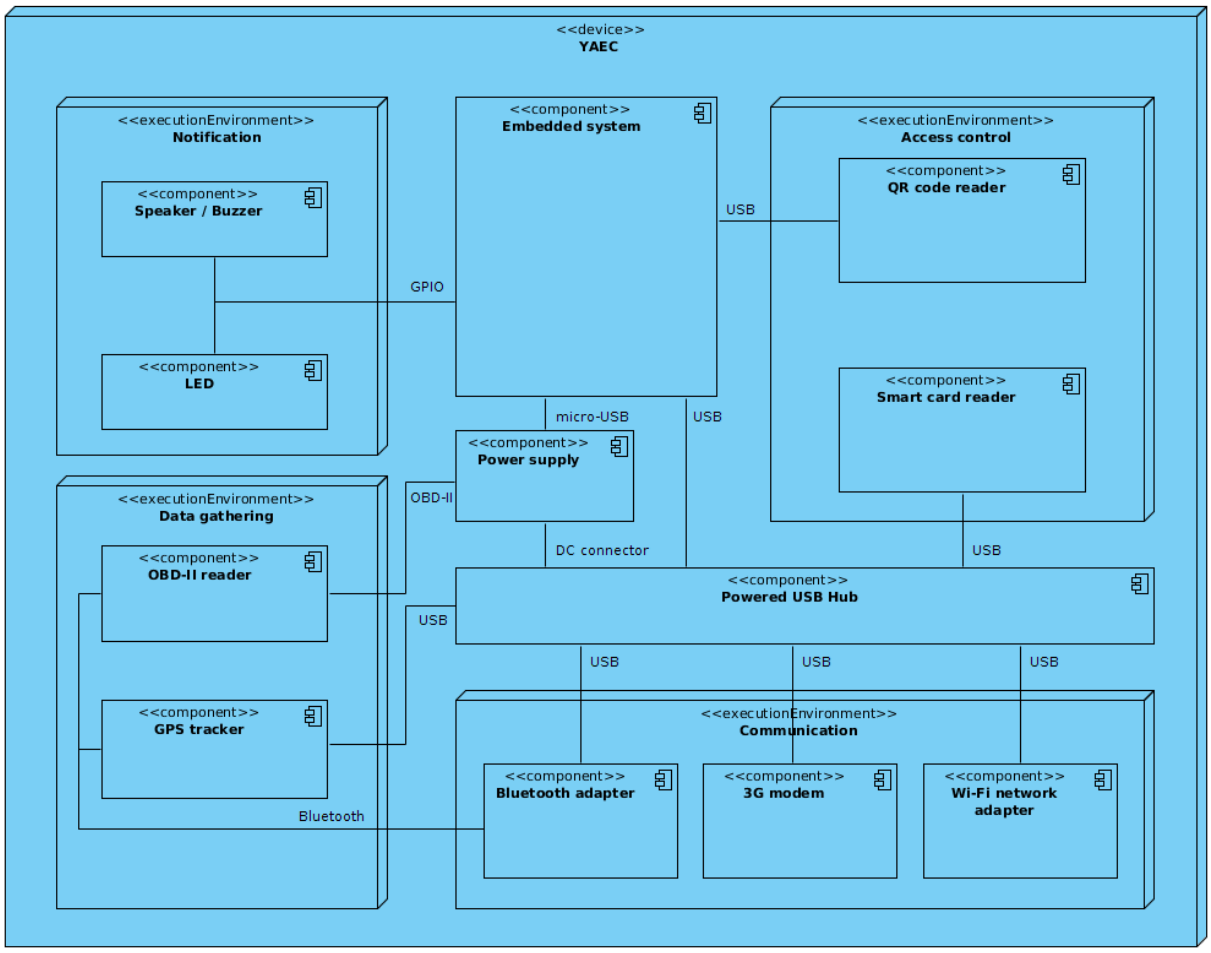

Figure 4.5: YAEC's physical configuration

The authentication module uses pyscard to create a client for pcscd. This client queries all the smart cards that get into the near field of the reader. The authentication consists in recognizing the card and querying it (taking in account its internal structure) to identify its holder. This procedure consists in sending specific APDUs to the smart card to retrieve relevant data.

The usage of the IST card for authentication allows a secure and robust identity check because it unequivocally assigns a user from IST to the service. However, not all the smart cards encode the IST-ID in its memory. In fact, from all the considered cards, to the best of our knowledge only the IST card does it. Therefore, checking the identity of non-IST card holders is considerably more difficult because we need to maintain some sort of virtual identity, comprising all the information required for distinct cards, full name, ATRs, etc. Hence, for non-IST community (or the ones that belonging to IST, are not holders of the IST card), the QR code shall be used instead.

IST card authentication

The IST card is issued by Santander Totta and there is no publicly available documentation regarding its contents. Given that it is also a banking card, our first approach to extract the IST-ID was to query the card with some well known EMV Application Protocol Data Unit (APDU)s. If the IST-ID was stored in the

EMV application, we would be able to search for it in the card's response. Unfortunately, no IST-related data was found. Most of it was obviously related to the banking functionalities the card provides. The only notable piece of information useful for authentication was the name of the card's holder; given that there are multiple persons with the same name, we cannot use this data as it does not uniquely identify the holder.

Another approach was to brute-force search the card but the effort rendered useless. Given that command APDUs may range from 4 bytes (just the header) to 259 (header + data), the universe of possible combinations rises to 2^{8*259}. Even assuming a very optimistic (an unrealistic) time of $1ms$ per query and that the sent APDU was always only the header, it would still take us almost 50 days to dump the whole card. Given this constraint, we tried to use a dictionary attack instead. This dictionary was made of public known APDUs used in other applications. No evidence was found.

Unsure if the card would contain an IST-ID at all, we asked the help of the IST DSI. Fortunately, they had technical documentation regarding the IST application in the IST card. This documentation not only contained the required APDUs but also the structure of the responses. Implementing the IST card authentication was a matter of retrieving the IST-ID from the card. Curiously though, the implementation of the IST application is not conformant with the protocol standards. In order to access it, a specific APDU is sent; the response for this request is different from the standard success code. Even if we had stumbled on the right APDU in the brute force approach, we would have discard it as valid since the response code was different from the success one.

Another curious fact is that most of the fields in the IST application are blank. In fact, the only available information is the IST-ID, the academic year and a bit stating if the card has been activated or not. Interesting data, like the role or the department, although foreseen in the response structure, are stored blank as well.

QR code authentication

The QR code reader behaves like a normal keyboard ie. every time the QR code detects a barcode, it outputs to the Standard input (stdin) the code it just read. This turns the QR code authentication a matter of reading the code, checking if it is present on the booking list and, if not, reading the identity encoded in the QR code.

CC authentication

The CC technical documentation does not provide any useful information regarding the smart card itself. Instead, it can be read how to interact with the client application or how to digitally sign documents as an example. All the technical information regards the (restricted) usage of the card and not on the implementation of the system. Documentation for an API with bindings for several languages exists but it requires loading a heavy Software Development Kit (SDK). Since we only had to know the CC number, no documentation was available and it was not acceptable to run CC's middleware in YAEC (which would not even work since it is an X server application and we are not running X server at all), we decided to

reverse engineer the relevant parts in order to extract the CC's number.

The CC's middleware is distributed as a Java Archive (JAR) package which can be opened using tar. Although most classes were obviously compiled, some tests were included in plain text. However, these only evaluated the middleware API. It was though enough to build a client application that when linked to the same library would produce the same results. However this brought extra complexity to the project, yet another programming language and we would waste a lot of time just guessing how to interpret the API response.

After successfully decompiling part of the middleware, it struck us that we could instead implement an effective man-in-the-middle attack in the bare metal. Turning to pcscd, we carefully logged all the APDU's sent to the card and interpreted their response. Building on previous knowledge, we were able to discover the relevant APDU's for both initializing the card and query its assigned number. This was directly implemented on our smart card authentication module offloading all the processing of CC's middleware and saving us some major headaches.

Other smart cards

Although other smart cards were not considered for authentication, some functionalities were implemented. For the old IST cards (issued by CGD), AEIST cards and ISIC, it is possible to query the name of the card's holder. This is done by initializing the card and querying it using the EMV standard APDUs. No further development was made with these cards since they are not the official ID in IST, exception made for the ISIC. This card in particular may be used to authenticate an international student but lack of documentation and low usage made it an unattractive option.

4.4.3 Data Gathering

YAEC collects data from multiple sources. The database in YAEC are plain Python sets. Although JSON is effectively a Javascript object, it can be strictly evaluated in a subset of Python (as a dictionary). This data format is therefore the main one and is used in all the communication with Shuver.

Modern vehicles offer a peak view into its sensors and ECU data through the OBD-II port. This part was paired with an OEM adapter sporting the widely known ELM327. This adapter provides a Bluetooth interface that other devices may access. That being said, an OEM Bluetooth adapter was coupled to YAEC through one of the USB ports.

All the data is collected into the filesystem in plain-text (ASCII encoded):

- **Vehicle** - information regarding the vehicle gets the name ”vehicle-*”;
- **Location** - geolocation data is stored in kml files named ”location-*”;
- **Journey** - shuttle journeys information is stored in ”journeys-*”.

where * indicates a numeral ranging from 0 to 4 (the days of the week) or ”s” for a special Shuttle (like those on Saturdays).

No special precaution was made in securing the files as tampering with the system is not expected. All these files are stored in the current/ directory. Once a new week starts, this directory is renamed previous/ and a new current/ directory is created. If there is already a previous/ directory, it will be removed. This is a naive logrotate. Since the most important data has already been stored in Shuver, the only purpose of this is to maintain a more complete dataset of all the events. This requires a possibly large amount of space, reason why it is rotated after a while.

Once YAEC is powered on, it will verify if it is the first time it is powered on in that day. If it is, it will fetch status information from Shuver and the booked seats for that day.

YAEC may confirm this if there are "booked" files in current/ and there is no .lock file. On the other hand, if those files exist, it's the first time YAEC is powered on. Please keep in mind that YAEC is powered by the cigarette socket, which may cut the power once the vehicle engine is turned off.

The OBD-II data consists in the raw readings of the OBD-II adapter. Those are stored in JSON format with the following specification:

$$"sensor - key" : "value", ...$$

for each of the timestamps it is queried.

Geolocation data is collected with a GPS receiver coupled in one USB port. It allows us to receive a (latitude, longitude) pair each time we query it.

Journey information collects the identities of each individual that hops into the vehicle. This information is the exact same that is sent to Shuver.

4.4.4 Communication

YAEC communicates with Shuver through its REST API. For providing HTTP capabilities, we used Requests [10], a library that offers a more Pythonic way of dealing with REST APIs.

As an example, the initial fetch of the booked seats is accomplished with:

```
r = requests.get('http://shuttle.tecnico.ulisboa.pt:8080/bookings')
bookedSeats = r.json()
```

YAECs API was implemented in the exact same way as Shuver's one.

YAEC was deployed with a 3G modem for being able to communicate. Later, it used instead a Wi-Fi network adapter connected to tethered 4G connection.

4.4.5 Powering up

Our solution employs a lot of distinct devices. All this hardware requires power to operate and unfortunately, inside a vehicle, power is quite limited. Since we are powering YAEC through the cigarette lighter socket, we purposively got an adapter that provided two USB ports. The rationale is one is used for powering the embedded system and the other for providing extra required current. Since the RPi is

[10] http://docs.python-requests.org/en/latest/

quite limited in the number of USB ports and our devices are all powered by USB, a hub was coupled to YAEC. Being a powered hub, the free USB port in the cigarette lighter socket was used to power it. However, the employed hub can only be powered by two ports; a standard DC cylindrical connector jack or an USB B port. Worse than that is the fact that to connect to the RPi, we need to use a standard USB B to USB A cable which may power the hub automatically. Given that there were no USB to DC connector adapters available, we chose to weld directly a salvaged USB cable to the hub, disconnecting the DC connector in the process. A previous attempt at welding a DC to a USB cable proved useless as the cable was partially broken inside.

The hub did not help us as some of the USB ports were also malfunctioning but we got the system working for the evaluation.

These power constraints prevented us from deploying the 3G modem with YAEC. Every time it was plugged in, it brought the system to a halt, either rebooting or even not being able to power at all. A lot of work needs to be made in this area, deploying better quality devices, lowering the consumption of the system and perhaps including a battery for preventing power issues.

Figure 4.6 illustrates the last prototype of the system, before getting assembled.

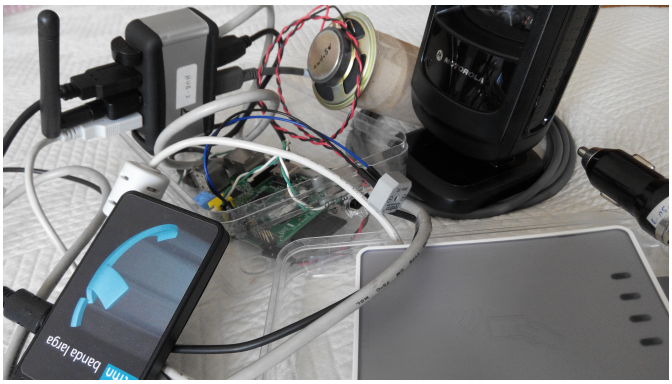

Figure 4.6: Last YAEC prototype

4.4.6 Notifications

We need to notify the user of the outcome of the authentication procedure. As such, a new module was required: something audible and observable that behaved as a notification.

As part of the notification module, we plugged an old speaker to the GPIO pins. This speaker was a salvaged part from an old Advanced Technology (AT) chassis and the only inscription it had was “5W”. To output some sound, a naive square wave generator was implemented by changing the levels on one of the pins. However, the sound was almost inaudible. A brief experience with Text To Speech (TTS) was also implemented but we could not drive the speaker enough to hear anything meaningful. Clearly, a driver was needed; however, drivers also consume current and given our previous power issues, we chose to build an analog acoustic amplifier instead. This was a matter of taking our speaker and searching for something that allowed the sound to be amplified. A roll of toilet paper fitted perfectly and coupled to the speaker was enough to provide a more audible sound. Still, our naive square wave

generator generator was not really appropriate for outputting sound so we began searching something else.

WiringPi is small library written in C that offers access to the GPIO pins with high-level methods similar to Arduino ones. The one big advantage of WiringPi is that it offers a better performance that RaspberryPi foundation own library [11]. Using the software tone generator of WiringPi, we implemented some notification methods, running a A (at 440Hz) through the speaker and a light though a LED.

After understanding that our notification module was not really useful in the evaluation, we changed the speaker for a proper buzzer and the LED to a high intensity one. The results are clearly better but the buzzer still lacks a driver for improved performance.

4.5 Modules interactions

This section details the communication between the different modules of the system. No relevant implementation detail exists for the interactions between YAEC and Shuver. All the messages interchanged between components of the system are in JSON.

4.5.1 BMM service clients, Shuver and Fenix

BMM service clients rely on Fenix for authenticating the user. This is done by following the OAuth authentication flow.

The user is first redirected to Fenix to authorize access. After authorization has been granted, the user is redirected back to the BMM service client with a code (encoded in the URL) that can be exchanged for an access token.

From this moment on, the BMM service clients are able to connect to Shuver and fetch the status of the service. The services are now available to the end-user. BMM clients issue booking, management and monitoring requests to Shuver using a RESTful API.

The messages interchanged between the BMM Service clients, Shuver and Fenix are depicted in Figure 4.7.

4.6 Synthesis

In this chapter we described the implementation of our system. Three distinct components were implemented: a central server named Shuver, an embedded system called YAEC and one client of the BMM platform.

The Shuver receives and manages all the communications, being the central node of our system. The Shuver exposes a public available REST API which is used by the YAEC and BMM clients, but other entities may also consume it to communicate with the Shuver. The use of a Mongo DB allowed us to develop a very flexible and scalable solution.

[11] http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/

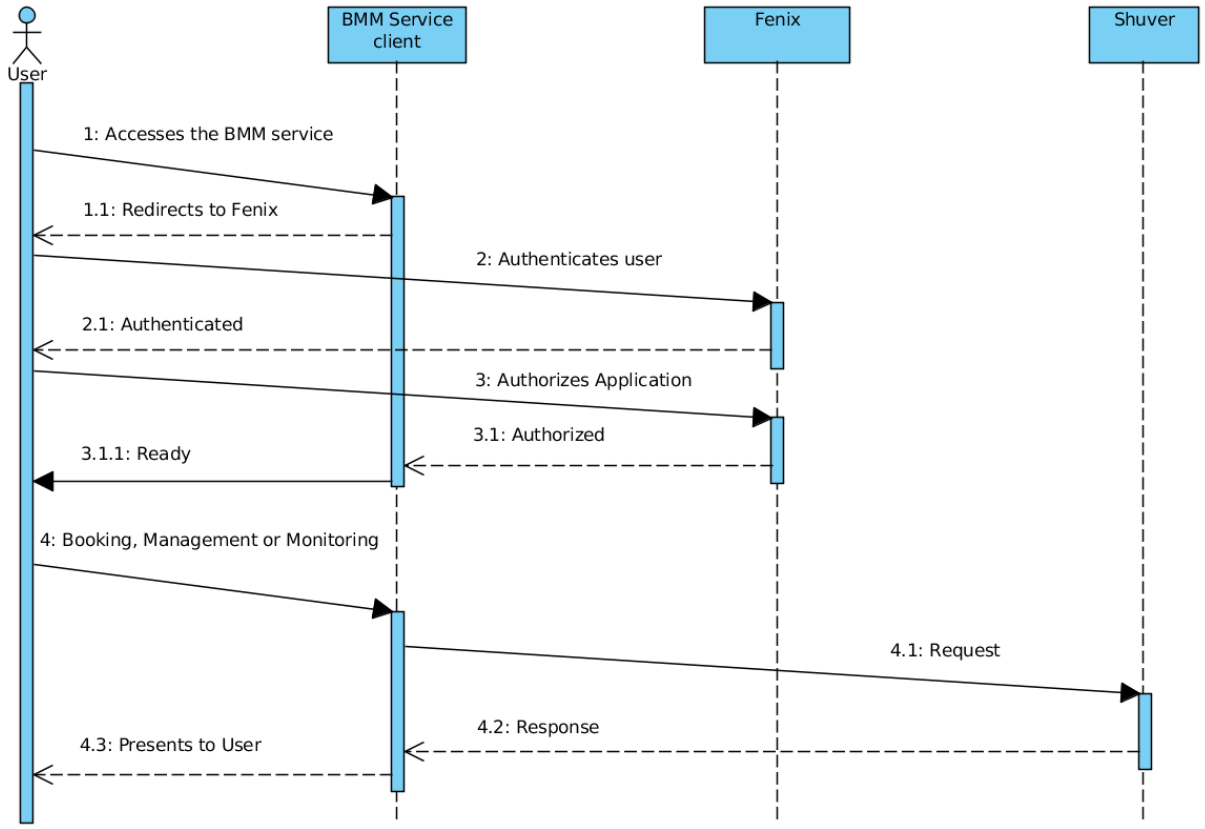

Figure 4.7: Messages between BMM service clients, Shuver and Fenix

There is one YAEC per shuttle. This system is implemented on a Raspberry Pi, coupled with different set of devices that are used to: access control, notifications, data gathering and communication. The access control supports an array of different cards, both contact and contactless; the communication supports different technologies as well: 3G and Wifi to communicate with the Shuver and Bluetooth to communicate with the OBD-II reader and GPS tracker.

The BMM service client supports authentication through Fenix and provides a graphical and responsive interface.

Chapter 5

Evaluation

In order to validate the assumptions that led to the design and architecture of our solution, we developed a proof-of-concept system which we evaluated in different scenarios. In this chapter we present the evaluation of our solution. Instead of evaluating each one of the components independently, we assessed the system from a global perspective that takes into account the objectives that led to its design.

5.1 Evaluation objectives

Our main objective is to solve some of the problems of the IST shuttle service, namely, the booking of seats in scheduled journeys and retrieval of valuable information of the vehicle in real-time. To ensure the efficiency and correctness of our solution, we deployed a prototype in one of the running vehicles.

In a first stage, authentication is the problem to solve; therefore, the first evaluation set will assert that every user in the shuttle can be unequivocally identified. Then, it is required that users are able to book seats and occupy them as such. We need to assert that the booking process is correct and the system acknowledges that. One of our goals is also to being able to present the user with insightful information; this requires that we can indeed retrieve this data from the vehicle and therefore we should be able to test it. On a final stage, we will test the communications between the embedded system and the central system and ensure they behave in real-time.

Hence, in order to tests these different stages, two independent test scenarios have been defined and a set of tests have been realized in each one of them. The next sections will detail them.

5.2 Authentication test

5.2.1 Goals of the test

The purpose of this test was to gather information about the different smart cards used by the IST shuttle community and understand if there were any problems reading and authenticating the people hopping into the vehicle. Also, as a first field-test, we tried to engage people and to explain them the context of

this experiment.

5.2.2 Test preparation and schedule

The success of this type of tests deeply depends on the help of the users. It is of uttermost importance that the community is aware of the significance of the project and actively participates in the field trials. Hence, in order to prepare the IST shuttle community and promote its engagement, we informed them about the project and announced the realization of the test in advance. An e-mail was sent to all IST Taguspark community (students, professors and staff) to let them know about the project and the test that was going to be realized. A copy of this email can be seen in appendix C. The IST community was also notified with a post in the IST Taguspark Facebook profile. With this procedure, we hoped that every user was aware of the reason of the test and voluntarily participated in it, by having a card available (for authentication purposes) upon hopping into the vehicle.

The date of the test was chosen to minimize service interruption and typically, the last day before holidays brings a lower number of passengers to the shuttle. Hence, the test was scheduled to 17/12/2014, the last day before Christmas holidays. The test comprised several journeys of the shuttle that departed from Taguspark to Alameda during the afternoon. The chosen journey schedules were 13h10, 14h10, 15h10 and 16h10.

5.2.3 Scenario description

A proof of concept authentication device was used in the test. This authentication device consisted of YAEC's smart card reader coupled to a development laptop. The smart card reader was the *Identive Cloud 4700F* [38] and the laptop a late 2012 *Toshiba Portegé Z930-118*, comprising an Intel i5-3317U, 4GB DDR3@1600MHz and a 128GB SSD. Concerning the software, an early version of YAEC's code, solely designed for authentication purposes, was used. In this version, a query was made to the smart card for identification purposes and an answer was generated, after correctly guessing what card was inserted.

Figure 5.1 illustrates this first prototype system.

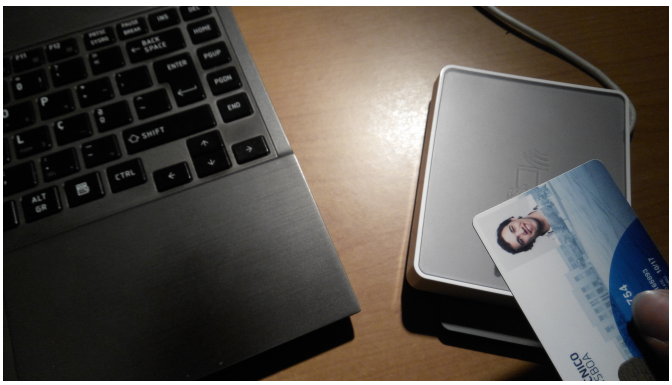

Figure 5.1: First authentication prototype

To the extent of collecting smart card data and understanding what were the major hurdles, the accepted smart cards for authentication were:

- **IST Card** - using either contactless or contact interfaces, querying IST-ID.
- **old IST Cards** from CGD - querying holders name using the contact interface.
- **CC** - querying identification number and performing checksum test.
- **ISIC** - querying holders name using the contact interface.
- **AEIST** card - acknowledging just the type of the card.

Initially, the system was placed in the entrance of the building nearest to the vehicle departure point. However, during the tests we observed that several passengers hoped in the vehicle without participating in the test. Hence, on the last journey, we purposely moved all the equipment to entrance of the vehicle instead.

5.2.4 Results and analysis

In order to assess the results of these tests, two simple metrics were used:

- **Number of Authenticated Users (NAU)** - represents the number of users that successfully completed their authentication.
- **Authentication Success Ratio (ASR)** - represents the percentage of successful authentication requests over the total number of authentication requests made.

Figure 5.2 depicts the value of NAU per type of card. Table 5.1 depicts additional information about the number of authenticated users, by describing the results achieved in each one of the journeys. In both cases, the valid results comprises all the cards that have been identified previously and the field 'Other' indicates cards that could resemble a connection with IST but could not be identified in runtime. This includes even older IST and AEIST cards or cards from Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento em Lisboa (INESC-ID). Errors in the authentication were left out as they did not represent meaningful data for this test.

The analysis of the figure showed that the IST community used a diversity of identity cards, ranging from the CC card to the AEIST card. Two important things must be mentioned: first, around 12% of the users had an invalid identity card, the old IST card; second, for the purpose of the IST shuttle service the only valid identification is the IST card and less than 30% of the users had it. The analysis of the table showed also that this value is even smaller in the last test, that comprises almost 50% of the users, as less than 13% of the users authenticated themselves using this card. This might compromise the success of the service when its deployment starts if no enforcement policy to have the card is made by IST in advance.

The authentication test managed to identify 86 out of 90 inquired people during 4 journeys. This represents an ASR of 95.5%, which is a very good result.

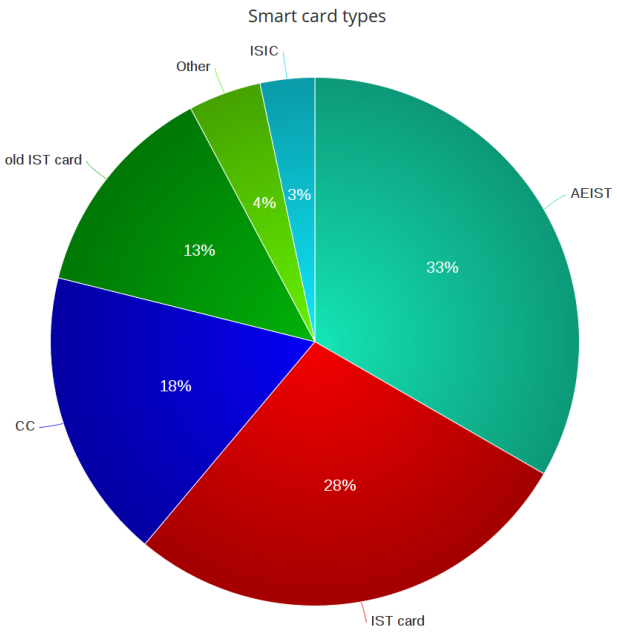

Figure 5.2: NAU per type of card

Journey / Card	IST Card	CC	old IST card	AEIST	ISIC	Other	Total - journey
13:10 (TP - AL)	5	0	2	6	0	1	14
14:10 (TP - AL)	9	7	0	1	1	0	18
15:10 (TP - AL)	5	5	1	0	0	0	11
16:10 (TP - AL)	6	4	9	23	2	3	47
Total - card type	25	16	12	30	3	4	90

Table 5.1: NAU per type of card

5.2.5 Additional comments

We were expecting more people to authenticate through our system during these first set of trial tests.

Although some did not acknowledge what was the equipment for or have not really read the e-mail, some people were deliberatively running away from the bench. Some of the reasons may include missing identification or misuse of the service. In the two first journeys, we were not able to gather as much data as we were expecting so we decided to move the equipment to the vehicle instead. Even though there were still a bunch of people who did not really cooperate. Also, we managed to find two people who were not allowed to travel and do not have any valid identity card. Even though we did not enforce any policy as this is a matter outside of the scope of this thesis.

Another interesting aspect of this experiment was dealing with the collisions between old IST and newer AEIST cards. Both cards share the same ATR for contactless interface but differ in the installed applications. This implies that we were not able to correctly identify a card without making extra queries,

taking longer time. Although these cards were not authorized for authentication purposes, they showcased some of the problems with this kind of system.

5.3 Full operational test

In order to start deploying the system in the shuttles, we performed a full test where the different aspects of the platform were assessed in real operational conditions. The next section details the tests that were realized for this purpose.

5.3.1 Goals of the test

The aim of these tests is to perform a full evaluation of the BMM service so that one can identify whether the system is ready to be deployed or not. As the QR-Code reader had not arrived at the date of the test, we were not able to perform an authentication test using QR codes.

The tests were divided in three distinct groups that correspond to the way the service is implemented. They comprise: the booking in the BMM service client; the authentication and access control in the shuttle; and the monitoring of service. A fourth group comprises an additional set of tests aims at assessing the users satisfaction with the service, based on this preliminary experiment.

5.3.2 Test preparation and schedule

In order to run these tests, an YAEC device was deployed in a vehicle of the IST shuttle service. The access was granted by Barraqueiro, the fleet operator that owns and provides the shuttle service to IST, during a meeting specially scheduled with the purpose of presenting the project and ask for permission to execute the tests. Unfortunately, the vehicle did not had an OBD-II port. The vehicle used during the test is the one that is permanently allocated to IST and the driver was informed in advance why, when and how the test was going to be realized and what type of cooperation was expected.

The results of the previous test have shown that users were not really engaged with the service and a significant number of them, in spite of using the shuttle, did not participated in the test. So, a special attention was dedicated to the communication within the IST community and an institutional e-mail was sent the day before and a post was published in the Facebook profile. A copy of this email can be seen in appendix C. A misunderstanding of the communication led to a delay of the Facebook post, that was published only in the day of the test. This fact caused confusion on several users, as they continued to book seats after the date of the test, as it will be shown later.

This time, a more detailed information was sent regarding the service, booking policies and access conditions to the vehicle, as well as the scheduled date and journeys of the test. It was clearly mentioned that:

- There would be only one booking per journey and per user;
- The booking was not transmissible;

- The bookings were only available to the IST community using Fenix identity;
- For evaluation purposes, only 10 seats per journey would be available for booking;
- Every passenger that hopped into the vehicle had to authenticate himself, on a card reader available at the entrance of the vehicle;
- For this purpose the passenger had to have a valid IST card, otherwise, exceptionally, he could use the CC.

To restrict the impact of the test on the shuttle service, we decided to evaluate the system in a less busy day. Hence, once again, we selected the last day before holidays to run the tests: 31/03/2015, which is the last day before Easter holidays. We chose three distinct journeys on the same day for evaluation purposes: from Taguspark to Alameda at 15:10 and 17:10 and from Alameda to Taguspark at 16:10. Those journeys were chosen based on the occupation information we gathered from the IST staff and in order to test the system in both rush-hour and in schedules with less passengers.

5.3.3 Scenario description

The preparation of a full operational test required the entire system working properly. To achieve this, both Shuver and BMM service clients were installed and tested in advance, in order to being used without interruptions by the IST community during the test.

The Shuver and BMM service client systems were deploying in a Virtual Machine (VM) of "tecnico.ulisboa.pt" domain, being the service accessible through the URL: shuttle.tecnico.ulisboa.pt.

The BMM service client went live some days before and was extensively tested, privately, in order to overcome most UI and UX problems. All the bookings made in this phase were taken in account so the booking algorithm could seed the database with different rankings, other than initial default ones. Hence, when the email was sent, the BMM service client and Shuver were fully available to manage booking requests. Unbooked seats were available to any user that presented a valid identity when hopping in the vehicle.

In the day of the test, before passengers hopping into the shuttle, YAEC was deployed in the vehicle by connecting the power source to the cigarette lighter socket. In order to guarantee that every passenger is authenticated, the card-reader is placed in a visible place, near the entrance. Figure 5.3 illustrates the YAEC system in the vehicle, at the beginning of the test.

Since there were some problems with the power consumption and current stability, the YAEC used an Wi-Fi network adapter instead of the 3G modem. A 4G enabled smartphone (LG-Google Nexus 5) provided a tethering connection to YAEC which was used for all the communication purposes. Given that the QR code reader had not arrived by the evaluation date, all the authentication was made using smart cards instead. The smart card reader was initially coupled to the YAEC prototype box but since the power cables were getting in between the driver and the steering wheel, we decoupled the device and unfolded the cable.

Figure 5.3: YAEC in operation inside the shuttle

At the same time passengers hopped in, they also authenticated themselves. The allowed smart cards were IST Card, CC and ISIC. Every time a smart card was recognized and authenticated correctly its holder, a led lit and a buzzer outputted a beep. On the other hand, if the card could not be identified or if there was any error in the authentication process, both led and buzzer would beep 10 times periodically.

The first two journeys (round-trip) also evaluated the entrance of passengers in intermediate stops. There were no passengers hopping in Sete-Rios in the 15:10 journey. Unfortunately, due to power issues, the initial authentication of the 16:10 shuttle was not performed. Since it began to work properly shortly after, it did manage to authenticate passengers hopping in Sete-Rios and a full authentication was repeated in the end of the journey. The test ended with the departure of the 17:10 shuttle from Taguspark.

At the end of each journey, a questionnaire was made to get feedback from the users.

5.3.4 Results and analysis - BMM service client

The first step of our analysis of the full operational test is focused on the first interaction of the users with the system: they need to access the BMM service client to book seats.

The evaluation of the BMM service client comprises the two stages that are needed to fulfill this task: the access to the BMM service client and the booking process. The next sections details each one of them.

Access to the BMM service client

To assess how users interact with our platform some raw information was retrieved. Google analytics was used to gather the number of pageviews, sessions and users. The logs of the BMM platform were used to retrieve the number of BMM authorized users and bookings. Based on that, several different metrics have been defined that measure distinct levels of interactions with the systems, as follows:

- **User Conversation Rate (UCR)** - defines how many unique users access the service:

$$UCR = Users/Sessions$$

- **BMM platform Conversion Rate (BMMCR)** - defines how many users are converted into BMM platform users and is given by:

$$BMMCR = BMMUsers/Users$$

- **Booking Conversion Rate (BCR)** - defines how many users perform booking requests and is given by

$$BCR = Booking/BMMUsers$$

Table 5.2 depicts the raw information registered at four different instants of time:

- t_0 - immediately before the email was sent;
- $t_0 + 6h$ - 6 hours after the email was sent;
- $t_0 + 24h$ - the day after the tests;
- $t_0 + 48h$ - two days after the tests.

Metric / time	t_0	$t_0 + 6h$	$t_0 + 24h$	$t_0 + 48h$
Pageviews	-	174	634	869
Sessions	-	102	373	436
Users	-	90	189	278
BMM authorized users	5	78	135	205
Booking requests	12	23	48	66

Table 5.2: BMM client data

Using the information of this table, the UCR, BMMCR and BCR were computed. Figure 5.4 depicts the results that have been achieved.

The reason why there are no values for pageviews, sessions and users in t_0, is that Google analytics only started to track metrics after the BMM service going live. It would make no sense to track them before as it would consist of our own accesses to the service. Similarly, the authorized users and booking requests were made by ourselves to test various parameters of the system.

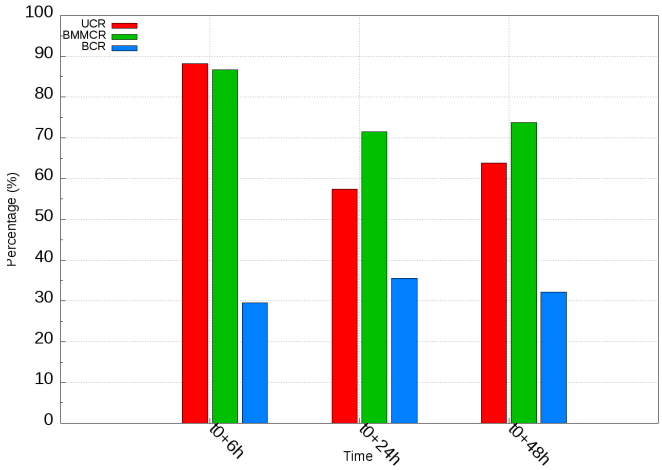

Figure 5.4: Assessment of user interaction with the BMM client platform

Things got interesting shortly after; as the service went live, all the metrics skyrocketed, managing to achieve 78 authorized users with 23 booking requests in just 6 hours.

As stated in the figure, the UCR was fairly high before the field test and most of those users were actually converting into BMM service users. As the time passed by, the rate of new users dropped but the BMMCR is constantly high; this indicates that there is interest in the service. Still, this interest did not manifest itself in the number of booking requests which remained fairly low. Surprisingly, even after the booking period, the rate of booking requests remained constant.

Booking process

The next stage of the evaluation of the BMM service client is related with the booking process. Using the logs of the BMM platform, three different types of data were retrieved:

- **Valid booking request** - the number of valid booking requests that have been performed by BMM authorized users. This number accounts for one booking per BMM authenticated user per journey.
- **Invalid booking request** - the number of invalid booking requests that have been performed by BMM authorized users. This number accounts for all the remaining cases.
- **Granted booking** - the number of booking requests that have been granted. This accounts for the number of valid requests of BMM users that have credits to book seats, under the maximum allowed number of bookings that was defined.

Table 5.3 depicts the information of the booking process that have been gathered at the BMM platform. These results details the different journeys that have been selected by the users.

The analysis of the table showed that, in all the journeys, the booking requests were less than the available seats. We feel that this may have happen because of the short notice of the sent e-mail and the late Facebook post. Users were not able to book seats for the same day and most of them probably read both of the notifications in the following day (which may also explain why the booking requests remained constant).

Journey / booking	Valid bookings	Invalid bookings	Total	Granted bookings
15:10 (TP - AL)	4	8	12	8
16:10 (AL - TP)	1	4	5	4
17:10 (TP - AL)	8	10	18	10
Other schedules	21	10	31	—
Total	34	32	66	32

Table 5.3: Bookings made

Surprisingly, plenty of bookings were made to schedules in which YAEC was not being tested. Maybe due to a miscommunication, a lot of those bookings also reflected the most sought after journeys in rush-hour. After some tweaking in order to ensure incorrect bookings would not be available, the service stabilized and the bookings made declined considerably as it became evident that booking would not have any effect unless YAEC was deployed in that journey.

The booking process is one of the most important aspects for the success of our service. In order to assess it, two different metrics were defined:

- **Booking request Success Rate (BreqSR)** - defines the percentage of valid booking requests and is given by:

$$BrSR = Validbookingrequests/Totalbookingrequests$$

- **Granted booking Success Rate (GbSR)** - defines the percentage of valid booking requests that converts into granted bookings and is given by:

$$GbSR = Grantedbooking/Validbooking$$

Figure 5.5 depicts the BreqSR and GbSR.

The analysis of figure showed that more than 50% of invalid booking requests have been registered. This means that, either the users failed to set the date, or just repeated the same request. Considering just the valid requests, almost all the users had their request accepted, as the value GbSR is close to 100%. This happened because for most of them, there is no past history to consider and all the users that booked seats had enough ranking to do it. Therefore, none of them lost ranking because they failed to authenticated in a previous journey where he booked a seat; neither gained more credits due to his very good behavior. The only two exception were ourselves, who previously seeded the database in order to get a failed booking request.

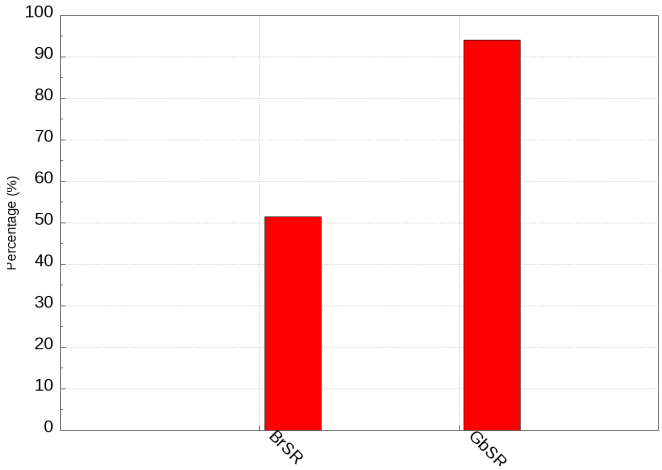

Figure 5.5: Assessment of the booking process at the BMM client platform

5.3.5 Results and analysis - Access control in the vehicle

After deploying our service, users need to authenticate themselves in order to access the IST shuttle, as access will be granted only to authorized users. Hence, the next step of our evaluation is focused on the shuttle access control and authorization processes. The next section details both tests.

Access Control

The authentication module of YAEC ensures shuttle passengers are who they claim to be. This means that, when using our system, access to the vehicle should only be granted to authenticated users. However, to ensure that they do not have reasons to complain about the service, the process must be fast enough to avoid the creation of long queues of users that are waiting for their time to do it and hop in the vehicle. Hence, the authentication must be correctly performed and fast. These are the two aspects that were validated in the next sections.

Type of cards used for authentication process In order to perform an effective access control, every user that hops in the vehicle must use the IST Card for authentication purposes. However, given the results of the previous tests, we were not quite convinced about the possibility of having the great majority of users with this card. Hence, we decided to repeat the authentication tests to be sure of the basic assumptions that need to be guaranteed before the deployment of the service. Figure 5.6 depicts the overall smart card types used for authentication in the whole scenario.

Although IST, CC and ISIC smart cards were allowed for authentication purposes and there were some Erasmus students hopping in the shuttle, there were no passengers using ISIC. More surprisingly was the use of the CC though; as the QR code reader had not arrived yet, the CC was thought as a fall-back option for passengers who were not holders of the IST card. Since we are neither able to store the CC number nor query Fenix for identification purposes, the use of the CC is not suitable for a complete authentication. Still, (albeit with a low advantage - 43 out of 85) most of the passengers used it instead of the IST card. From several reasons, the most common were *I do not have it yet!* or *Is there other*

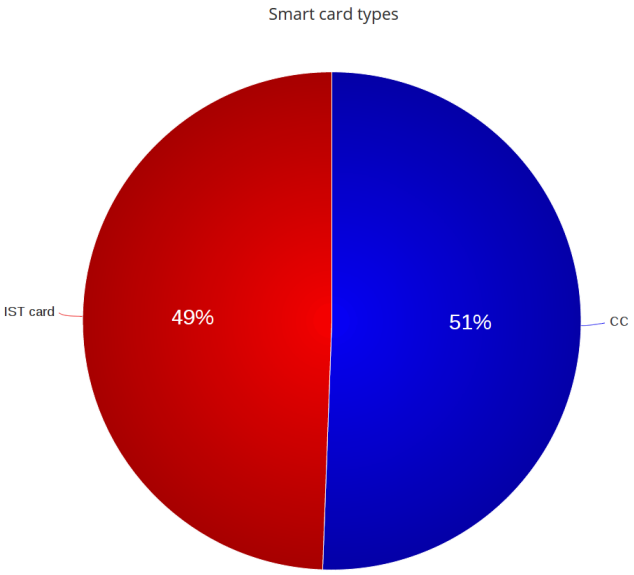

Figure 5.6: Smart card types

card? (holding the previous IST card).

Table 5.4 depicts the usage of the smart cards per journey. As it can be easily stated, the previous conclusions are valid for all the journeys. Clearly, a lot of effort needs to be made by administrative services to bring those cards to their rightful owners as this is a major obstacle in the deployment of this solution.

Journey / Card	**IST card**	**CC**	**ISIC**	**Total - journey**
15:10 (TP - AL)	16	18	0	34
16:10 (AL - TP)	8	8	0	16
17:10 (TP - AL)	18	17	0	35
Total - card type	42	43	0	85

Table 5.4: NAU in full operation conditions

Figure 5.7, figure 5.8 and figure 5.9 illustrates three different situations that were registered during the field trial and shown the different authentication situations that have been reported.

Latency of the authentication process The authentication at the entrance of the shuttle introduces an additional overhead and might delay the users entrance in the vehicle. Therefore, the goal was to validate that the new procedure would neither deteriorate passengers experience nor cause any unsustainable overhead. As part of the evaluation process, each time a smart card was used for authentication and some time measurements were taken. Specifically:

- **Detected** - When a card was detected;

Figure 5.7: Field trial - authentication with IST card

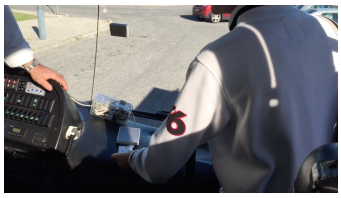

Figure 5.8: Field trial - authentication with CC card

Figure 5.9: Field trial - authentication with an invalid card

- **Authenticated** - When a card was identified and authenticated;
- **Error** - When any error occurred.

This information was used to compute the Authentication Response Time (ART) - the time since the card is detected until the result of the detection is generated. This result may represent a card that was identified and authenticated or an error.

The values are registered in tables 5.5, 5.6 and 5.7, which illustrate the maximum, average and minimum value for each one of the journeys. Please note that these values are potentially (and should be) higher than reality; the reason is that all the logging code is extra added execution time which will not be run in production and thus lower the effective times our solution can provide.

ART [s]	**15:10 (TP - AL)**	**16:10 (AL - TP)**	**17:10 (TP - AL)**	**Maximum per card type**
IST Card	0.170	0.190	0.270	0.210
CC	0.210	0.190	0.190	0.197
Error time	2.000	1.550	1.840	1.797
Average per journey	0.793	0.643	0.767	

Table 5.5: Maximum smart card Authentication Response Time

Based on the information that was gathered, we computed the ART for each one of values for the two cards and for the errors situations. The results are shown in figure 5.10 and figure 5.11.

According to the figures, the time taken to authenticate when using the IST Card or the CC card are quite similar, although the former is slightly lower. This can be explained by the extra number of queries made for CC. Nevertheless, the variation of the ART for the IST Card is higher than the one observed at the CC. This is caused by the usage of the contactless interface instead of the contact one.

ART [s]	15:10 (TP - AL)	16:10 (AL - TP)	17:10 (TP - AL)	Average per card type
IST Card	0.140	0.145	0.152	0.146
CC	0.181	0.180	0.181	0.181
Error time	1.699	1.550	1.571	1.624
Average per journey	0.609	0.317	0.626	

Table 5.6: Average smart card Authentication Response Time

ART [s]	15:10 (TP - AL)	16:10 (AL - TP)	17:10 (TP - AL)	Average per card type
IST Card	0.110	0.130	0.130	0.123
CC	0.170	0.170	0.170	0.170
Error time	1.510	1.550	1.510	1.523
Average per journey	0.597	0.617	0.603	

Table 5.7: Minimum smart card authentication time

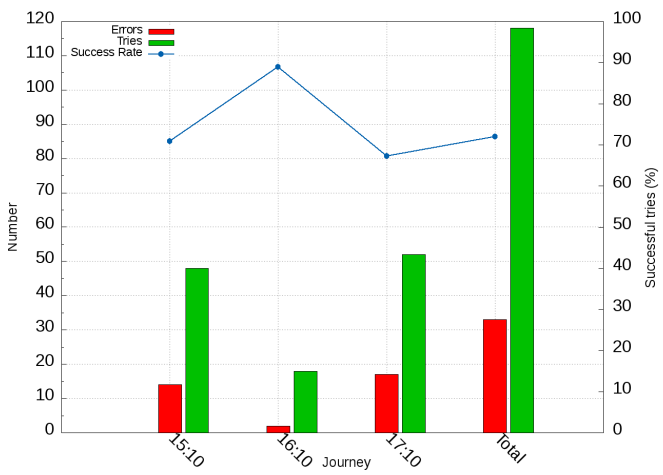

Figure 5.10: Smart card authentication errors and tries

Concerning the detection of error situations, the value increases significantly, reaching near 2 s. Again, the variation is higher because of the same interface issue. Moreover, the IST card also offers a contact interface, which users sometimes tried to use instead of the contactless one. Given that approaching the slot is enough to power up the card and we were constantly helping users, these values although high may be considered normal.

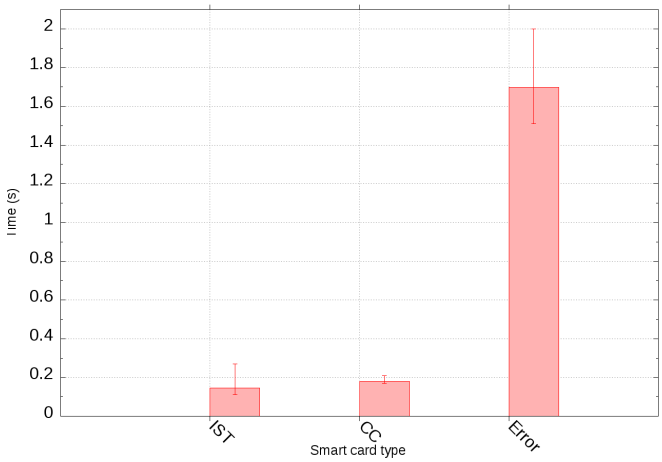

Figure 5.11: Smart-card ART

Authorization

Once the booking period ends, a set of users will be automatically authorized upon authentication. On the other hand, not only those users are authorized to enter the vehicle but a handful of others. This experiment took in account the already authenticated users and the ones that lacking the booking were also authorized to enter the vehicle.

Figure 5.12 depicts the overall results of this process.

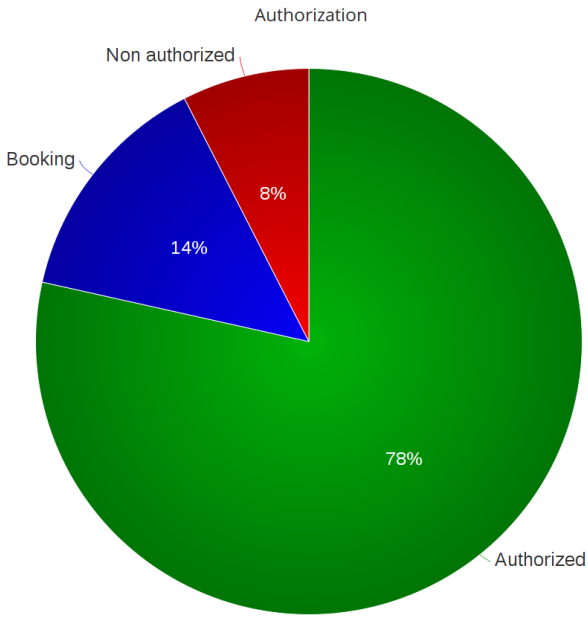

Figure 5.12: Authorization process

Every user got access to the vehicle. Although there were few granted booking requests for each journey, the number of users with bookings that hopped in the shuttle do not match. In fact, this number was much lower maybe due to the fact that the majority tried to authenticate with the CC instead of the IST card. Moreover, a number of people had no identification at all or hopped in without using the system.

A detailed information about the authorization process is shown in table 5.8, that depicts also the authorization results per journey.

Access try / Journey	**15:10 (TP - AL)**	**16:10 (AL - TP)**	**17:10 (TP - AL)**	**Total**
Authorized	30	15	27	73
Authorized with booking	4	1	8	13
Non Authorized	2	0	5	7
Total	36	16	40	94

Table 5.8: Authorization test

5.3.6 Results and analysis - Monitoring of the service

A lot of information can be retrieved from the shuttle during a journey and used for monitoring purposes. Hence, the last step of our evaluation is focused on different types of information that might be retrieved from the YAEC. The next section details these results.

Journey duration

Given that passengers hopping in the shuttle need to authenticate themselves, it is possible to extrapolate a lot of information just by looking at the time those authentications were made. Every-time YAEC was powered on, a “begin” event was recorded with a timestamp. The shuttle departs only after all passengers get on board (obviously, with a schedule constraint); since an authentication is made for each one, we can safely say that the departure time is later than the moment the last passenger authenticated himself. On the other hand, passengers may hop in the vehicle on intermediate stops; this allows us to extrapolate the moment the shuttle stops in those points just by following the authentication timestamps and finding a larger gap than usual (typically around minutes). YAEC records an “end” event once the authentication module turns off the devices. This allows us to box all the values in a given time range.

Figure 5.13 depicts the YAEC times for standby, authentication, journey and intermediate stops. A more detailed view is presented in table 5.9.

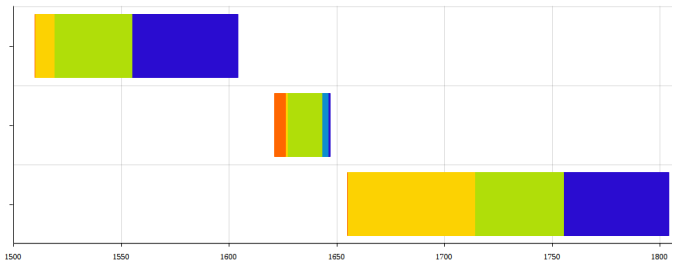

Figure 5.13: Journey’s times

Journey	Begin	Departure	Stop	End
15:10 (TP - AL)	15:10:11.152	15:19:34.93	-	-
16:10 (AL - TP)	16:21:01.798	-	16:26:54.9	16:46:49.52
17:10 (TP - AL)	16:55:43.747	17:14:17.59	-	-

Table 5.9: Journey's times

YAEC was powered on slightly late on the 15:10 journey; in fact it was powered on the moment the shuttle should have departed. The last authentication was made nine minutes later and the vehicle departed shortly after. No one hopped in the vehicle in Sete-Rios (the intermediate stop) and given that there was no external input, the shuttle arrived Alameda and YAEC was promptly shut down.

Unfortunately, due to power issues no authentication was made on the 16:10 journey. Once YAEC got operational, the "begin" event was recorded (eleven minutes after the scheduled 16:10 departure time) and just six minutes later, the vehicle arrived to Sete-Rios. Two passengers hopped in and the shuttle continued his journey to Taguspark. To ensure remaining passengers would get registered, we asked them to authenticate on the end of the journey. This procedure took four and a half minutes for 16 people and the last authentication time was 36 minutes after the scheduled time.

On the 17:10 journey, only the "begin" and the last authentication were recorded as we didn't hopped in the vehicle. That being said, the authentication for 35 people took only seven minutes, a much better figure than the previous ones. We are confident that once this system gets well known and understood by the community this time can be even improved for a full vehicle.

Traffic generated

Every time YAEC is powered on, a small amount of data is exchanged with Shuver. This data is mainly comprised of the booking details for each one of the users. The booking details is only fetched once per day to minimize communication costs.

Unfortunately, power issues made it impossible to use the 3G modem and all the other devices. The solution was therefore to use instead a Wi-Fi link to a 4G tethered connection from a smartphone. Although the network strength was far from ideal, we still used the 4G network instead of the 3G. Given that the modem was relatively outdated, it's expectable that a future prototype will deploy a 4G capable modem instead.

The booking list for the whole field test day occupied 4KB and took less than 1ms to fetch in our tests. This time is largely due to the quality of network connections.

All the periodic messages (weighting at) were delivered. There is no data regarding its time.

5.3.7 Additional comments

The users feedback regarding the usefulness of this service was unanimous; the growing mobility problem of Taguspark requires an optimized service and this solution looked like an effort in the right direction.

However, YAEC's notifications were considered quite poor and criticized by the passengers. The smart card reader status LEDs provided a confusing experience; using a contact card, once communication is successfully established, a green LED will appear. On the other hand, using a contactless card lights up a red LED instead due to the powering of the card and multiple failed queries made to identify the protocol to use. This caused quite a bit of confusion between passengers. Moreover, the extra led and buzzer proved inefficient to notify the user of anything: the light coming from the window and the engine/ambient noise was more than enough to nullify the extra added parts.

5.4 Synthesis and discussion

In this chapter we described the tests that we made to assess our system. After an initial authentication test designed to test different types of smart cards, we performed a full evaluation of the system. This evaluation comprised the different aspects of our system: BMM service client, access control to the vehicle and service monitoring.

The tests of the BMM service client have shown a fairly high user conversion rate (almost 90% after the test), although the number of bookings is less impressive. This indicates that there is interest in the service.

Concerning the bookings, the majority of them have been granted, as more seats were available for reservation than the number of valid requests that have been received. Nevertheless, the booking algorithm has given a low rank to two users, that repeatedly book seats for previous journeys.

Authentication tests have shown a very important problem: more than 50% of the users do not have the IST card. This problem must be solved before the service is in use, since the management of the bookings is supported on the information available at Fenix. The tests have also shown that authentication and access control would not disrupt the user experience, since the latency of these process is very small (200 ms).

Using the information retrieved in the shuttle it is possible to get interesting and meaningful information, such as number of accesses, number of bookings, duration of the journey and time between intermediate stops.

Users gave a positive feedback of this service, but they claim for a more effective way of identifying the result of the access control, as the colors of the LEDs were not visible and the buzzer was difficult to hear.

Chapter 6

Conclusions

Is it possible to devise a real-time booking system for the IST Shuttle service that is able to support a controlled access to the shuttle for the IST community and guests ?

To address this question, we proposed the Booking, Management and Monitoring service. The BMM service extends the existing Shuttle service by allowing the IST community and other authorized individuals to book seats, manage different options regarding the configuration of the service and monitor the stats of the shuttle and the vehicle.

With the introduction of this new system we intend to make the shuttle service more efficient and to provide the tools to manage it in a more optimal way. This is achieved by a set of novel components:

- An embedded system available in each vehicle (YAEC), responsible for the access control upon entering the shuttle and providing vehicle and journey data.
- A central node (Shuver), which collects and processes all the information flow from YAEC and it is responsible for the heavy lifting behind the BMM logic.
- BMM clients which present and expose the available services and data.

The field tests gave us a very positive feedback, with high conversation rates and proving this service is non-disruptive and an effort in the right path. We can conclude that the answer to the original question is indeed possible. As always, compromises were made; however, we feel this solution has more advantages than disadvantages and brings new power to the management entities. Furthermore, this thesis has set the foundations and enabled future work to take place, specifically:

- YAEC suffers a lot from power issues. Support for other power sources like solar or battery and improved energy consumption shall be researched. The notifications from YAEC shall be improved in order to overcome user experience issues. YAEC may also provide Wi-Fi onboard.
- The BMM client shall be extended with analytics of the service and more monitoring and management options. It may even be integrated with Fenix and Fenix Mobile.
- The service may use VANETs as a routing system for network messages.
- There is no reason why this work can not be extensible to other vehicle providers and institutions.

Appendix A

Communication

A.1 Shuver

All the clients will consume this API and work on top of it. For each of the API requests, an example response is given.

Status retrieves the status of the whole system

GET /status

```
{
        "online": "healthy",
        "uptime": "14 days",
        "yaec": "4 connected"
}
```

Fleet Information retrieves the Fleet Information (dump)

GET /fleet

```
[
        {
                "license-plate": "",
                "capacity": "",
                "provider": ""
        },
        ...
]
```

Journeys retrieves all the known Journeys

GET /journeys

```
[
        "07:15 Alameda - Sete Rios - Taguspark",
        "07:15 Alameda - Sete Rios - Cacem - Taguspark",
        "07:20 Sete Rios - Oeiras - Taguspark",
        "08:00 Alameda - Sete Rios - Taguspark",
        "09:10 Alameda - Sete Rios - Taguspark",
        "10:10 Alameda - Sete Rios - Taguspark",
        "11:10 Alameda - Sete Rios - Taguspark",
        ...
]
```

Booking retrieves all the bookings for a specific journey

GET /bookings/{id}

```
{
        "ist": [
                "65893":"16:10 Alameda - Sete Rios - Taguspark",
                "65933":"16:10 Alameda - Sete Rios - Taguspark",
                "65936":"16:10 Alameda - Sete Rios - Taguspark",
                "65943":"16:10 Alameda - Sete Rios - Taguspark"
        ],
        "other": ""
}
```

YAEC relay relays all the API calls to the YAEC node identified by id. Example shown for the status query.

GET /yaec/{id}

```
{
        "status": "healthy",
        "last-sync": "14-January-1990",
        "license-plate": "92-18-VJ"
}
```

Book new journey books a new journey

POST /booking/create

```
{
        "user": "",
        "date": "",
        "journey": ""
}
```

```
{
        "result": "Attributed",
        "qr-code": "647561727465626172626f73612d697374313635383933 2d416c
           616d6564612d54616775737061726b"
}
```

A.2 YAEC

For each of the requests, an example response is given.

YAEC Status retrieves the status of YAEC

GET /status

```
{
        "status": "healthy",
        "last-sync": "14-January-1990",
        "license-plate": "92-18-VJ"
}
```

Vehicle information retrieves overview of OBD-II data (this may change due to available sensors in the vehicle)

GET /vehicle/obd2

```
{
        "velocity": "78km/h",
        "duration": "7 minutes",
        "ecu-errors": "none",
        "consumption": "23l/100km"
}
```

Geo-location retrieves location of the vehicle

GET /vehicle/location

```
{
        "latitude": "38.7370420",
        "longitude": "-9.3036060"
}
```

Journey retrieves information about that journey

GET /journey

```
{
        "date": "14-February-2015",
        "time": "13:10",
        "route": "TP-AL",
        "occupation-rate": "47/49",
        "other": ""
}
```

Booked Seats retrieves all the booked seats in that journey

GET /seats/booked

```
{
        "ist": ["65893", "65933", "65936", "65943"],
        "other": ""
}
```

Occupied Seats retrieves all the occupied seats in that journey

GET /seats/occupied

```
{
        "ist": ["65893", "65933", "65936", "65943"],
        "other": ""
}
```

Appendix B

Database

B.1 fleet

```
{
        "license-plate": "",
        "capacity": "",
        "provider": ""
}
```

B.2 journey

```
{
        "vehicle": "",
        "schedule": "",
        "stop-points": ["", "", "", ...],
        "available-for-booking": ""
}
```

B.3 booking

```
{
        "user": "",
        "journey": "",
        "date": ""
}
```

B.4 user

```
1 {
2         "ist-id": "",
3         "cc": "",
4         "other": ""
5 }
```

Appendix C

Notifications to the community

C.1 E-mail - 16th of December, 2014

De: Técnico Lisboa (Núcleo de Apoio Geral do Taguspark) <noreply@tecnico.ulisboa.pt>
Data: 16 de dezembro de 2014 às 15:40
Assunto: Teste - Shuttle
Para:

Boa tarde,

Amanhã, dia 17 de dezembro, irá realizar-se no campus um teste de um sistema de reservas para o serviço de Shuttle no âmbito de uma tese de mestrado.

O teste, dirigido a toda a comunidade do IST-TP, irá decorrer entre as 12:00h e as 17:00h e consistirá na autenticação dos utentes antes da entrada no autocarro. Para este efeito, será colocado um receptor junto ao bar no topo sul (junto ao restaurante).

Pedimos a todos que passem o Cartão do IST ou o Cartão de Cidadão por esse receptor antes de entrarem no autocarro. Este procedimento é relativamente rápido mas pede-se a todos que se apresentem ligeiramente mais cedo para evitar atrasos.

Agradecemos desde já a ajuda e compreensão de todos.

Obrigado.

—

Esta mensagem foi enviada por meio do sistema Fénix, em nome de(o) Técnico Lisboa (Núcleo de Apoio Geral do Taguspark), para os seguintes destinatários:
Todos os alunos do campus Taguspark
Todos os funcionários do campus Taguspark
Todos os docentes do campus Taguspark

C.2 E-mail - 30th of March, 2015

From: Técnico Lisboa (Conselho de Gestão) <noreply@tecnico.ulisboa.pt>
Date: 2015-03-30 19:29 GMT+01:00
Subject: Teste ao sistema de reservas do shuttle
To:

Está em curso o desenvolvimento dum sistema que permite a reserva de lugares no serviço de shuttle. A reserva é pessoal e intransmissível e está disponível para membros da comunidade IST (com identidade Fénix).

O primeiro teste do sistema vai ser realizado durante o dia de amanhã, dia 31 de Março, nas seguintes carreiras:
15:10 - Taguspark - Alameda
16:10 - Alameda - Taguspark
17:10 - Taguspark - Alameda

Os utentes do shuttle devem usar o novo cartão do IST (cartão Santander) para aceder ao shuttle, procedendo à autenticação num leitor que se encontra disponível dentro do autocarro, junto à entrada. Caso não tenham o referido cartão, excepcionalmente, podem usar o cartão do cidadão.

É ainda possível efetuar a reserva de lugares através duma plataforma de teste, criada para o efeito, acessível através de: http://shuttle.tecnico.ulisboa.pt/ Para efeitos de teste, em cada viagem, o número de lugares que é possível reservar está limitado a 10. Será enviado um email com o resultado da reserva.

Contamos com a colaboração de todos os utentes do shuttle.

Teresa Vazão

—

Esta mensagem foi enviada por meio do sistema Fénix, em nome de(o) Técnico Lisboa (Conselho de Gestão), para os seguintes destinatários:
Todos os alunos do campus Taguspark
Todos os funcionários do campus Taguspark
Todos os docentes do campus Taguspark

Bibliography

[1] Kleinrock, L.: History of the internet and its flexible future. Wireless Communications, IEEE **15**(1) (2008) 8–18

[2] Clark, D.D., Partridge, C., Braden, R.T., Davie, B., Floyd, S., Jacobson, V., Katabi, D., Minshall, G., Ramakrishnan, K., Roscoe, T., et al.: Making the world (of communications) a different place. ACM SIGCOMM Computer Communication Review **35**(3) (2005) 91–96

[3] Toor, Y., Muhlethaler, P., Laouiti, A.: Vehicle ad hoc networks: applications and related technical issues. Communications Surveys Tutorials, IEEE **10**(3) (2008) 74 –88

[4] Uzcátegui, R., Acosta-Marum, G.: WAVE: A Tutorial. Communications Magazine, IEEE **47**(5) (May 2009) 126 –133

[5] Hartenstein, H., Laberteaux, K.: A tutorial survey on vehicular ad hoc networks. Communications Magazine, IEEE **46**(6) (June 2008) 164–171

[6] Técnico, I.S.: Sobre o ist: O instituto & história webpages. `http://tecnico.ulisboa.pt/` (2013)

[7] Arora, S.: National e-id card schemes: A european overview. Information Security Technical Report **13**(2) (2008) 46–53

[8] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Volume 263 of Lecture Notes in Computer Science., Springer (1986) 186–194

[9] Price, G.: The benefits and drawbacks of using electronic identities. Information Security Technical Report **13**(2) (2008) 95–103

[10] Collings, T.: Some thoughts on the underlying logic and process underpinning electronic identity (e-id). Information Security Technical Report **13**(2) (2008) 61–70

[11] : Portuguese citizen ID card roll out underway. Biometric Technology Today **15**(2) (2007) 3

[12] Agência para a Modernização Administrativa: Cartão de cidadão, o novo documento de identificação dos cidadãos portugueses. (2007) 9

[13] Shirey, R.: Internet Security Glossary, Version 2. RFC 4949 (Informational) (August 2007)

[14] Fujimura, K., Nakajima, Y.: General-purpose digital ticket framework. In: Proceedings of the 3rd Conference on USENIX Workshop on Electronic Commerce - Volume 3. WOEC'98, Berkeley, CA, USA, USENIX Association (1998) 15–15

[15] Ghìron, S.L., Sposato, S., Medaglia, C.M., Moroni, A.: Nfc ticketing: A prototype and usability test of an nfc-based virtual ticketing application. In: Near Field Communication, 2009. NFC'09. First International Workshop on, IEEE (2009) 45–50

[16] ISO: ISO/IEC 18004 - Information technology - Automatic identification and data capture techniques-Bar code Symbology - QR Code (2000)

[17] ISO: ISO/IEC 7816 - Identification cards - Integrated circuit cards (2004)

[18] ISO: ISO/IEC 14443 - Identification cards - Contactless integrated circuit cards - Proximity cards (2008)

[19] Shelfer, K.M., Procaccino, J.D.: Smart card evolution. Commun. ACM **45**(7) (July 2002) 83–88

[20] Khan, M., Takeshi, Y., So, I., Bessho, M., Sakamura, K.: A secure and flexible electronic-ticket system. In: Computer Software and Applications Conference (COMPSAC '09). 33rd Annual IEEE International. Volume 1. (2009) 421–426

[21] Mayes, K.E., Cid, C.: The MIFARE classic story. Information Security Technical Report **15**(1) (2010) 8–12

[22] Lee, Y.S., Park, Y., Lee, S., Kim, T., Lee, H.J.: Rfid mutual authentication protocol with unclonable rfid-tags. In: Mobile IT Convergence (ICMIC), 2011 International Conference on. (Sept 2011) 74–77

[23] ISO: ISO/IEC 18092 - Near Field Communication Interface and Protocol (2013)

[24] Belobaba, P.P.: Survey paper-airline yield management an overview of seat inventory control. Transportation Science **21**(2) (1987) 63–73

[25] Katz, J.: Reservation-based preauthorization payment system (January 7 2014) US Patent 8,622,292.

[26] Tsubouchi, K., Yamato, H., Hiekata, K.: Innovative on-demand bus system in japan. Intelligent Transport Systems, IET **4**(4) (2010) 270–279

[27] Edara, P., Teodorović, D.: Model of an advance-booking system for highway trips. Transportation Research Part C: Emerging Technologies **16**(1) (2008) 36–53

[28] Liu, K., Chan, E., Lee, V., Kapitanova, K., Son, S.H.: Design and evaluation of token-based reservation for a roadway system. Transportation Research Part C: Emerging Technologies **26** (2013) 184–202

[29] Hornick, S., Hong, D., Gupta, S., Mehrotra, R.: Airline seat inventory control method and apparatus for computerized airline reservation systems (October 19 1993) US Patent 5,255,184.

[30] Armstrong-Wright, A., Thiriez, S.: Bus services: Reducing costs, raising standards. Number TP-68. (1987)

[31] Pala, Z., Inanc, N.: Smart parking applications using rfid technology. In: RFID Eurasia, 2007 1st Annual. (Sept 2007) 1–3

[32] Bavkar, D.M., Khairnar, V.D., Gaikwad, V.B.: Pav: Parking allotment for vehicles using vanet. International Journal of Latest Trends in Engineering and Technology (IJLTET) **2** (2013) 68–76

[33] Wang, H., He, W.: A reservation-based smart parking system. In: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. (2011) 690–695

[34] Calypso Networks Association: Calypso handbook. `http://calypsotechnology.net/index.php/documents/specifications/public-documents/79-100324-calypso-handbook-v11` (2010)

[35] Mikulski, J.: Using telematics in transport. In: Transport Systems Telematics. Springer (2010) 175–182

[36] NMEA: http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp Visited in: Jan 2014.

[37] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System

[38] Identive Group, Inc: Cloud 4700 f, dual interface desktop reader datasheet. `http://www.identive-infrastructure.com/pdf/usa/DAT_CLOUD_4700F.pdf` (2012)