Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Catarina Moura

Pervasive Services for Flexible Spaces

Master's Thesis
Espoo, February 12, 2015

Supervisor: Mario Di Francesco, Aalto University
Instructor: Ingrid Schembri, VTT Technical Research Centre of Finland

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Catarina Moura

Title:
Pervasive Services for Flexible Spaces

Date: February 12, 2015 **Pages:** x + 92

Professorship: Data Communication Software **Code:** T-110

Supervisor: Mario Di Francesco

Instructor: Ingrid Schembri

Shared spaces are increasingly being used in working environments to cope with the limitations in the available facilities, in terms of both square meters and costs. One important example of shared resource is represented by a meeting room that can be booked and used by several actors, for instance, companies co-located in a business hub. To this end, current reservation systems have several limitations. First, access control is not really enforced based on the owner of the booking. Second, it is difficult to monitor the utilization of resources unless occupancy sensors are deployed, thus incurring in additional costs.

In this thesis we have realized a cloud-based reservation and access system for shared rooms. Our solution is based on an electronic lock and a digital sign together with a reservation server. Users can book a room by using third-party authentication and can access the room by a simple and usable method that involves scanning a QR Code with a mobile phone. We have designed the system architecture and have implemented the service by using modern mobile web technologies. We have also analyzed the economic feasibility of our approach and developed a supporting business model. Our system has been piloted in the Learning Hub of the Computer Science library as part of the Flexible Spaces Service project sponsored by the EIT ICT Labs.

Keywords: Pervasive Services; Flexible Spaces; Smartphone; Authentication; Reservation System; Cloud; Database; Digital Sign; Sensor; Business Model;

Language: English

Acknowledgements

I would like to express my sincere and deepest gratitude to my supervisor, Dr. Mario Di Francesco for his essential and insightful mentoring and tireless patience. I am also grateful to my instructor, Ingrid Schembri.

Finally, I would like to thank my family and friends, specially my mother for the endless encouragement and support.

Espoo, February 12, 2015

Catarina Moura

Abbreviations and Acronyms

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AWS	Amazon Web Services
BSON	Binary Script Object Notation
CA	Certificate Authority
CAPEX	Capital Expenditure
CDI	Critical Design Issue
CPU	Central Processing Unit
CS	Computer Science
CSF	Critical Success Factor
CSS	Cascading Style Sheets
DB	Database
DBMS	Database Management System
DNS	Domain Name Server
DOM	Document Object Model
EBITDA	Earnings Before Interest, Taxes, Depreciation, and Amortization
ECMA	European Computer Manufacturers Association
EIT	European Institute of Innovation and Technology
FSS	Flexible Spaces Service
HR	Human Resources
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I/O	Input/Output
ICT	Information and Communication Technology
ID	Identification or Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
LAN	Local Area Network
NFC	Near Field Communication
NPV	Net Present Value
NoSQL	"Not Only SQL"
npm	Node Package Manager
OATH	Open Authentication
OPEX	Operational Expenditure
OTP	One-Time Password
OTT	Over-the-top
PID	Process Identifier
PDA	Personal Digital Assistant
P2P	Peer-to-peer
P&L	Profit and Loss
QoS	Quality of Service
QR Code	Quick Response Code
R&D	Research and Development
RDBMS	Relational Database Management System
REST	Representational State Transfer
RFC	Request for Comments
RFID	Radio Frequency Identification
ROI	Return on Investment
RPC	Remote procedure Call
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
STOF	Service, Technology, Organization and Finance
SWOT	Strengths, Weaknesses, Opportunities and Threats
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOTP	Time-Based One-Time Password
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VTT	Valtion Teknillinen Tutkimuskeskus (Technical Research Centre of Finland)
W3	World Wide Web

W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
WiFi	Wireless Network
WSDL	Web Services Definition Language
WWW	World Wide Web
XML	EXtensible Markup Language

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Progress in mobile devices, including smartphones and tablets, and the convergence with cloud computing technologies are making possible to realize a variety of innovative pervasive services, such as those in ubiquitous computing. The goal is to blend these services into our everyday physical environment so naturally that they will become invisible. Nowadays, pervasive services are a very important area in many application fields, thereby it introduces new challenges and research opportunities.

However, the effective design and development of pervasive services requires adaptive interactions among multiple devices and software components with properties that distinguish them from traditional information system.

Proposals from the early 2000s [45, 46, 50] try to account issues related to smart environments. Indeed, recent advances in sensing and communications technologies make it feasible to realize smart or intelligent environments that respond to the complexities of modern life and support new services. Cook and Das [46] defined a smart environment as an environment capable to collect data and apply it with usefulness, improving an inhabitant experience.

Others, considered flexible spaces to allow new social interaction patterns, such as shared spaces for working in collaborative environments. Among them, the Flexible Spaces Service (FSS) project sponsored by the EIT ICT Labs and coordinated by VTT [28].

Therefore, modeling integrated pervasive services and their execution environments must be able to gather information about users and their physical and digital environment to adapt the provided service to their current context. Consequently, these services will bridge the gap between digital or physical worlds by providing users with more personalized services. This is nowadays possible due to the proliferation of devices realizing the vision of ubiquitous computing that Mark Weiser proposed in the early 1990s [84]. The main idea was to spread computing capabilities that become familiar

and support a modern digital life.

This thesis aims at addressing the field of pervasive computing and more precisely the area of flexible spaces. At the time of writing, despite the success of pervasive services, there has been little research support for programming, i.e., every solution is still an ad-hoc solution. Additionally, shared spaces are increasingly being used in working environments to cope with the limitations in the available facilities, in terms of both square meters and costs.

The objective at the thesis is to realize a cloud-based reservation and access system for shared rooms. Our solution is based on an electronic lock and a digital sign together with a reservation server. Users can book a room by using third-party login and access the room by a simple and usable method that involves scanning a QR Code with a mobile phone. We have designed the system architecture and implemented the service by using modern mobile web technologies. We have also analyzed the economic feasibility of our approach and developed a supporting business model. Our system has been piloted in the Learning Hub of the Computer Science Library of Aalto University [1] as part of the Flexible Spaces Service project.

Furthermore, the system and its user interface can be suitably adapted to cover other applications to workspaces, such as, for instance, enterprises and public organizations or other smart environments. For these reasons, the interest of this work extends well beyond the limits of the specific, though an important service described in the thesis.

1.1 Research Topic

The rising popularity of pervasive computing and a growing desire for successful projects in the marketplace, motivate an urgent need to create pervasive services to facilitate the construction of context-aware platforms to optimize the operation of the devices that populate smart and flexible environments. On the other hand, shared spaces and resources in working places are increasing, which brings difficulties in terms of facilities, both square meters and costs. There are studies [33, 35] showing that facilities in terms of total business costs is the second highest cost after salaries. Besides this, another limitation is the space area, because it is hard to move, increase or reduce people.

Nevertheless, accordingly to a recent study conducted in Aalto University about work environments [36], the average rate of the utilization of Aalto spaces is below 20%; meeting rooms even have utilization rates lower than 10%.

These are some of the constraints affecting shared or flexible spaces, which

motivate us to design a valuable solution to efficiently use available spaces. For instance, providing in real time the information where is an empty room and simplifying the access to the flexible environment. These solutions create a better atmosphere for the people.

One important example of shared resources is represented by a meeting room that can be booked and used by several actors, for instance, companies co-located in a business hub. To this end, current reservation systems have several bottlenecks. First, access control (identifying the user), hence unauthorized person can occupy easily the room (room squatting). Second, it is difficult to monitor the utilization of resources unless occupancy sensors are deployed, although this approach is still not reliable without a valid user identification. Third, primitive procedures still remains such as manual room booking.

The objective of the work is to realized a cloud-based reservation and access system for shared rooms. The fight integration with the physical environment – sensors, mobility devices – and software is what allow us to take appropriate actions. By making this relationship explicit we derived a framework. This study was driven by the need for a practical pervasive service for flexible space with potential for commercial exploitation. We considered different ways to design the system and to implement the service, then we have chosen the technological communication solution that performs better than existing ones. We have also derived a business model to realize the service offering.

The design and modelling was focused in the flexible space of Aalto Computer Science Library to support students, professors, and visitors wherein one of the activities is "User requests a room in the library for a specific date and time". The concept that our work proposed is a web service-based solution that comes when the user navigates from his physical world to the Web to pick up the URL of the reservation system to activate "Book Room", also to scan QR Code using the smartphone to activate "Door Open".

Traditionally, a lot of work have been developed in pervasive computing on behalf of users concerning devices and applications. However, the evolution of pervasive computing with service-oriented computing is less reported. In addition, there is a need to provide transparent interfaces between disparate entities in the environment. The propose is to study this pervasive (intelligent) environment focusing on services to support user's basic activities.

We have built our cloud-based distributed architecture using the Node.js/Hapi.js frameworks for fast and scalable web applications written in JavaScript. We have developed the system with a generic mechanism for context management, using MongoDB for storing the application state, and OAuth to allow

secure authorization. We have also adopted HTML5, CSS, and jQuery as a basis for our responsive website design. For the local network the system has a digital signage based on TOTP algorithm and an advertising URL through a Quick Response (QR) Code.

A new service can be defined as "an offering not previously available to customers that they perceive as being new"[42]. Therefore, we provided a practical approach to link the technological value to customer value by business models formulation, based on Strengths, Weaknesses, Opportunities and Threats (SWOT) analysis, Service, Technology, Organization and Finance (STOF) method and financial considerations.

1.2 Contributions

This study has contributed to the research literature by presenting important issues related to pervasive services focus on flexible context programming capabilities, and also examining features for customer and network value by formulating the business model.

On the one hand, service providers and users/consumers will benefit about the success of pervasive services implementation. The research for innovation may be conducted in real-life context, such as everyday life, work or leisure contexts, which is the environment of this thesis.

The major features of our approach are the following:

- a solution that combines a smart electronic lock and a cloud-based reservation system for shared spaces;
- the electronic lock as an enabling technology that allows to enforce access control and to indirectly measure the space utilization;
- the reservation system is flexible to implement different policies and incentives;
- shared spaces with an efficient use by maximizing the square meters and reducing the costs;
- meeting rooms were the focus, because they have the lowest rate of space utilization among the other spaces in Aalto University [36].

On the other hand, the contribution to research agenda will be the looking into some aspects of established unified and ubiquitous communications services academic discipline.

At the end, the contribution that we can obtain facilitates other research developments on pervasive services computing.

1.3 Structure

The remainder of the thesis is organized in five chapters. Chapter 2 focuses on the background aspects of ubiquitous computing as well as concepts related to smart environments, then summarizes the related work about technologies for pervasive services.

Chapter 3 provides the case study scenario and the design and implementation of the proposed system, which realizes the reservation and access control service. Chapter 4 details the business models and analyzes the economic feasibility of the proposed solution.

Finally, Chapter 5 presents some concluding remarks with directions for further research.

Chapter 2

Background

In this chapter we begin by looking at pervasive services in flexible spaces and defining the core concepts, after which we discuss the importance of technologies that are relevant from a service perspective.

2.1 Pervasive and Ubiquitous Computing

The term ubiquitous means everywhere. In computer science it refers to devices that are distributed into the physical world, giving users access to computing and communication capabilities. So, ubiquitous computing integrates technology into the environment.

The ubiquitous computing devices can be computers, sensors or computers embedded in everyday objects. At the same time, ubiquitous computing involves the necessary information and technology infrastructures (ICT).

The ICT sector is assisting a tremendously evolution since the years 1990s, with Mark Weiser's original vision of calm computing [85], to the actual second decade of the 21st century, with the emergence of ubiquitous deployment of public display systems using techniques to build large-scale networks [47].

More recently, appears pervasive service computing concept enabling technology (like mobile phones, portable computers and smart jackets) and human behavior interactions within environment by – always available – services. According to Zhou et al. [86] nowadays many areas like remote communication; remote information access; fault tolerance; high availability; and security and privacy are foundational to construct pervasive computing. Nonetheless, drivers like: (i) mobility in the systems; (ii) service-oriented systems developed around business processes; (iii) context-awareness systems that can sense the physical environment; (iv) smart devices, appliances specialized in information; are the crucial factors that make evolution for the

pervasive computing toward Pervasive Service Computing [86]. Thereby, it is expected that these services will blend into user's live as computing devices.

The challenge component in pervasive service computing (and the difference for pervasive computing) is the adaptation and the full integration of human behaviors into a service.

An important technological pervasive services enabler is context-awareness, which means the user's situation (that is *context*). Context-aware services use such information to adapt services automatically. The power of pervasive services becomes apparent when it is integrated with user's daily activities.

Nowadays computing systems are based on a non-centralized approach characterized by user needs (intentions) and context sensitiveness. Formerly, computing was focused on centralized web services which means that users can interact with systems through the Internet or a particular network, not supposed to take into account its context [41].

So, following this modern computing systems approach, the main goal of pervasive computing is to design services that minimize the barriers between the humans cognitive model of what they want to accomplish and the system's understanding of the user's task.

An increasingly important example of pervasive system is given by digital signage as displays becoming ubiquitous. According to Want and Schilit [83] the recent growth in the digital signage market is expected to increase significantly in the next few years.

Interactive digital signage brings great opportunities for personalization of information presented on digital signs that are enabled by mobile devices interactions with displays.

2.2 Smart Environments

Smart environments are spaces managed by automatic systems providing people with an intelligent environment that facilitates everyday practices. Examples of smart spaces are given by homes, workplaces, classrooms, vehicles, and other places densely instrumented with sensors, displays, computers, mobile devices, wireless networks which enable users to control and monitor the installed devices.

According to Cook and Das [46], designing smart environments is a goal that appeals to researchers in a variety of disciplines including wireless networking and mobile computing. Here, the goal is to design an automated system that users do not need to explicitly configure or even know operating parameters for the smart space. The major requirements in building smart environments are the followings [46]:

1. *Physical properties.* Perception of the environment in real time and in real world objects, which means identifying the context by attributes of: (i) People (e.g. user's personal factors); (ii) Location, proximity (e.g. noise, temperature, light intensity); (iii) Hardware interfaces (mobile, wireless, sensors, other computing entities) that collect the space attributes.

2. *Information.* A description of about the environment that is stored in system components (e.g database).

3. *Action execution.* Information is processed by the system components and automatically executed for the smart environment.

Smart environments have applications in several domains: smart spaces; smart homes; smart energy systems; health and wellbeing assistance; digital cities; future media and content delivery; intelligent transportation systems; education; hospitals; emergency services. Among these applications, pervasive public display systems and the associated applications have grown spectacularly in recent years.

From Flexible Spaces Service project we have analyzed its applications and we proposed to characterize smart spaces when the intelligent environment addresses the workplace focusing on the following attributes:

- Optimization of resources and facilities in office space;
- Promoting work efficiency through the learning environment;
- Providing new office setups through digital services;
- Providing business through office rental.

Co-working spaces is increasing due to the fact that home is proving to be unpopular as a long-term work arrangement and nowadays we assist the return flow of flexible workers. So, for instance, VTT has been exploring new ways of working [62, 65] to: (i) address the lack of space; (ii) reduce real estate costs; (iii) improve the support for employees in a distributed work environment; (iv) promote interaction (collaborative) work.

In this context, smart spaces leads to "flexible spaces" when shared resources are provided for purposes related to work.

Flexible Spaces endeavors to monetize the space use, therefore makes the space more cost-efficient. The value comes when with users experiences and services enhance the users awareness of resources utilization.

We have developed a pattern of Flexible Spaces characteristics:

- Reconfigure spaces for support collaborative work;
- Reuse shared resources;
- Promote self-service use;
- Provide real-time and fast services;
- Address automated user authentication that protect data from unauthorized access;
- Reduce real estate costs, reclaiming unused space or tracking space utilization.

Some environments at first glance do not seem to have smart objects though they can be made smarter, as the use case of controlling a display in a shared meeting room [69].

Pervasive displays have become increasingly significant in the last decade, which requires substantial innovation that contributes to the creation of wide range of new services.

Oat et al. [69] states that public displays have the potential to enable unprecedented applications and services, also Davies et al. [47] refers to open display networks as a new communications medium for the 21st century.

The vision of flexible services, is that in the future the Internet will be a Web of Services which users can create their own services through small or individual service components, and can combined with larger service entities due to their open modular design. Even, if this vision would not be entirely accomplished, it is very probable that the interaction of services with cloud computing systems will increase the demand for open technologies, which is the main technological enabler for this development path.

2.3 Technologies for Pervasive Services

In this section related to pervasive computing services we present an important feature, technology.

Pervasive computing helps practical deployment of advanced technologies, and particularly, the related applications that could be delivered as services. In this sense, when these new technologies have a disruptive impact they become a radical innovation as it changes the people behavior and it impacts the environment, then they are the backbone of a new communications era.

Technology will naturally move from a close to an open model and for Davies et al. [47] to this happen, relatively to the open public displays

paradigm, five key elements should be achieved: (1) open architecture; (2) information; (3) privacy; (4) user control; (5) business models.

For instance, a transformation can only occurs if open display networks are comparable to an app store [47]. The idea behind consists in the owner registers his display, then he/she is able to develop or install applications on it. In the user side, with a matching application installed on the mobile activates a personalized display functionality. However, this paradigm will bring a drawback concerning privacy.

Our pervasive service comprises this open vision by applying modern web technologies which are described in this section. Specifically technologies for programming in flexible spaces.

2.3.1 Web Communications

Web communications, including protocols, standards and other frameworks, have proven themselves adaptable to construct a virtual bridge between computing systems and physical entities (people, places, or things). However, the communications on the web between browser (client) and server can be quite complex in ways that client is not aware of. Despite of, there is a great supply for programming frameworks.

Among web communications that can be used in pervasive services, to target specifically pervasive and flexible spaces, we leverage not only the well-known HTTP protocol, but also technologies such as REST and WebSockets. Oat et al. [69] provided a set of comments favoring WebSocket protocol for pervasive and public displays, in contrast with MagicBroker RESTful choice. In this section, we present the interaction through the Web and the associated technologies.

HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative and hypermedia information systems [52]. Any device that supports the HTTP standard can access resources on the Web.

HTTP is the foundation of data communication in the World Wide Web, where the interactions take place between a browser and a web server. It is a stateless protocol, that is, each HTTP request/response is independent from the previous one. The HTTP stateless characteristic demands additional mechanisms to handle user sessions – one of them is represented by browser cookies i.e. small text files stored in the user's browser computer to track and to remember the stateful information of the particular user. The

HTTP protocol uses Transmission Control Protocol (TCP) connections, and by default the web traffic utilizes port 80.

HTTP communication is characterized by messages (request/response) always initiated by the client as a request to the server, followed by a response from the server to the client. HTTP uses the request methods – GET, HEAD, POST, DELETE, PUT – and the status code acting as a response to the request. The most popular codes are the "200", OK; "400", Bad Request; "404", Not Found; "500", Internal Server Error; and "502", Bad Gateway.

For instance, the representation for HTTP URLs must be as follows:

"http:" "//" host [":" port][abs_path ["?" query]]

HTTP Secure (HTTPS) protocol provides a secure connection on the Web. The secure component involves adding a encryption/decryption layer between the HTTP and TCP called Secure Sockets Layer (SSL) or Transport Layer Security (TLS). The implementation of a secure connection that prevents man-in-the-middle attacks by checking the server's identity i.e., it verifies if the web server has a digital certificate issued by a Certificate Authority (CA) and additionally prevents eavesdropping and tampering attack through bidirectional encryption of communications between the server and the client.

The HTTP Internet standard protocol was developed mainly by Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) leading to multiples versions Request for Comments (RFCs) to improve. However, in scenarios of pervasive computing services it is need to combine with others web technologies.

REST

The REpresentational State Transfer (REST) is a well defined way of interacting with web servers. It has received a considerable adoption among web developers because it is simple to implemement and utilize. REST was introduced in 2000 by Roy Fielding in his doctoral dissertation at University of California [53].

REST is a web architectural style of networked systems relying on HTTP communication protocol which a client requests a resource (URIs) and receives the respective resource representation (another resource, a text, an image, etc). Thus, REST is characterized by: an application programming interface (API) that uses various HTTP methods (GET, POST, PUT, DELETE) as an interface to access the resources; an Uniform Resource Identifier (URI) as the resource identifier; and a data representation linked to other resources or data formats as the response from the server. REST supports a variety of data formats including plain text, HyperText Markup

Language (HTML), JavaScript Object Notation (JSON), EXtensible Markup Language (XML).

If a web service has a REST API, it is then defined as "RESTful".

The service is built on the web server, providing programmatic access to read and write from/to a website or application (client). In spite of, the request is always initiated by the client to the server like any other HTTP communication. The client does not require to know any details about the implementation of the process, all it is managed by the server.

This design (Figure 2.1) provides the mechanism client/server for computers or devices beginning with the Internet Protocol (IP) identifying each computer or device by an unique number (IP address), and addressing communication by IP packets encapsulated in TCP.

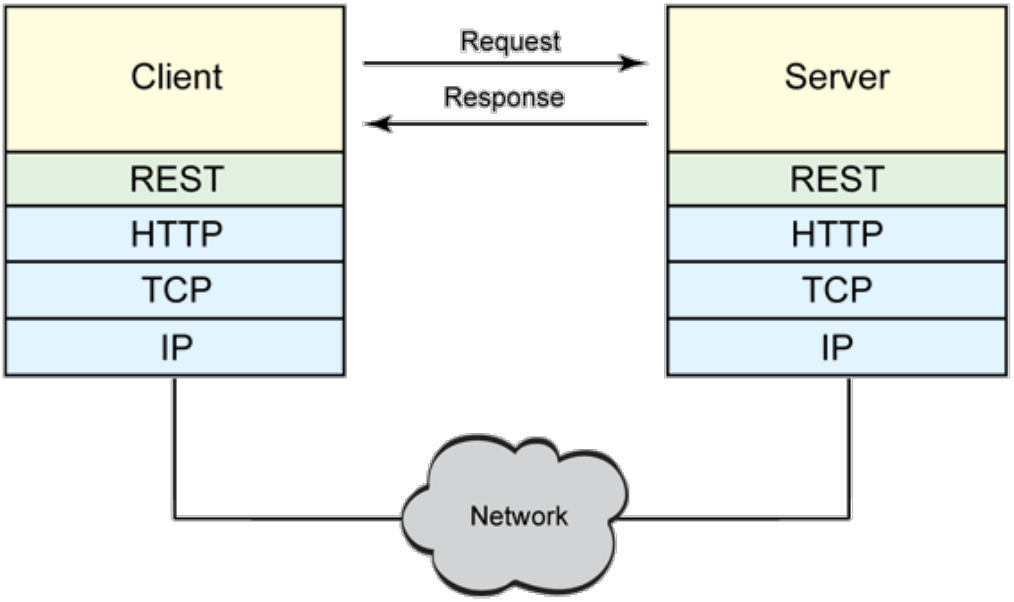


Figure 2.1: Layered architecture of RESTful interactions [80]

Restful services apply HTTP methods representing the actions that are used to operate in the resources, for example: the GET method is utilized to read and retrieve data from a resource; the POST method is used to create data; while the PUT method is to update existing data; the DELETE method will remove the data from the data store; even thought the verbs POST

and PUT, can be both used to create and update [52]. In fact, RESTful web services is the most simple and lightweight in comparison to others technologies for web services. Alternatives are represented by XML-RPC and SOAP-based web services [39, 44, 73].

XML-RPC is simple to implement unlike SOAP, briefly is a protocol that performs remote procedures calls (RPCs) from a client to the server and subsequently server to client by sending/receiving XML messages via HTTP Post.

Soap-based web services is more powerful and capable – add more information to the message, defined data types, set the specified character and named structures and arrays – than XML-RPC, shortly defines three entities: service registry – links the client to the server – ; service provider or server – provides the service – ; and service requestor or client – requests the service – . Soap-based web services comprises: Universal Description, Discovery and Integration (UDDI) – a platform-independent framework XML base registry where businesses can register and locate or search for web services – ; Web Services Definition Language (WSDL) – a language written in XML for describing web services and how to access them – ; and Simple Object Access Protocol (SOAP) – a protocol for accessing web services based on XML – .

For instance, XML-RPC and SOAP-based web services are cumbersome, complex and hard to use specially if the programming language is JavaScript because from the programmer perspective these technologies require to write many lines of code just to perform a simple task, because it is necessary to parse and create a XML structure every time. Conversely to REST, each object its is own URI and the data format is not rigid in this case JSON was the most suitable due to be a subset of JavaScript.

Furthermore, contrary to other systems which the servers maintain complex sessions with clients, REST API reduces to a simple response. Hence, RESTful architecture is stateless being more reliable and scalable [53].

These attributes motivates the selection of RESTful for web service provisioning [39].

Websocket

Websocket was standardized by IETF in 2011.

The Websocket protocol allows real time bidirectional communication between a client and a remote host [51]. WebSocket was designed to provide full duplex – bidirectional – communications channels layered over a single Transmission Over Protocol (TCP) connection. In addition, to the communication be stateful that is, a persistent connection between the web browser and the web server until one of them decide to close it, both client and server

can initiate the communication.

This characteristic, open connection, is what allows a real-time data exchange, which the client or the server can send a message at any given time to the other and receive event-driven responses without having to poll the server for a reply.

It represents a fundamental paradigm shift in a client/server technology because of the open and bidirectional communication capability. Lubbers and Greco [64] go further proclaiming that HTML5 Web Sockets represents the next evolution of web communications, a full-duplex, bidirectional communications channel, operating through a single socket over the web.

Furthermore, Websocket reduces overhead – network traffic –, increases the efficiency, and reduces dramatically the complexity, thus being a better alternative to Asynchronous JavaScript and XML (AJAX), and HTTP polling [56, 64, 72]. WebSockets have been used for communications between browser and server or between native mobile applications and servers or between devices creating functionality that works consistently across multiple platforms.

2.3.2 Backend

There are different technologies options for server-side provisioning of a service. In the following we only focus on the most relevant for our work: Javascript, Node and Hapi. For primary data store different solutions can be adopted, though should be consider the service software requirements in order to achieve better performances and less complexity. Therefore, the database management system (DBMS) chosen for the pervasive service was MongoDB and JSON for data-interchange.

JavaScript

JavaScript (frequently shortened to JS) standard [15] is the web's *lingua franca* (trade language) across all browsers, and it is a dual side programming language that can be used by both, client and server. Regardless, the JavaScript language was introduced in 1995 by Brendan Eich, the server-side approach has only recently gained massive popularity with Node.js, though from the front-end perspective it has been a reference for a long time.

JavaScript has been standardized in the ECMAScript language specification (ECMA-262) [48]. JavaScript is a lightweight scripting language, read and executed by an interpreter. The basic syntax and structure is C++ and Java like, which the reason was to reduce new concepts required to learn a

new language. Though, JavaScript is not Java nor part of Java Platform, it is completely divergent.

Furthermore, JavaScript is classified an object-oriented language which the inheritance model is prototype-based instead of class-based. Also, JavaScript is dynamic taping meaning that types are associated with values instead of variables, for example a variable "i" can be bound to a number and later on to a string.

Moreover, two of defining characteristics of JavaScript is that has closures and first-class functions. First-class functions are functions that are passed as arguments to other functions and a function defined in the closure remembers the environment in which it was created.

JavaScript is one of the three layers pillars of a web page (the others are HTML and CSS) adding the behavioral layer, which brings: interactivity with user; responsiveness web page; instantly content change; animation; effects, among others.

On the one hand, Javascript in the backend competes with other alternatives programming languages like Python, Ruby on Rails, Django, PHP.

On the other hand, Javascript in the front-end can result in several line of codes and be hard to implement, also across web browsers each browser can interpret the program differently. To overcome these two difficulties, jQuery emerged as a feature-rich JavaScript library, and jQuery will be detailed forward in the 2.3.3 Front-end section.

The JSON option is a subset of JavaScript that uses the same syntax, but the JSON format is text only which can be read and used as a data format by any programming language. Because of the similarity, a JavaScript program can easily convert JSON data into native JavaScript objects. JSON will be detailed forward in this section.

Node

Node.js (also known as Node), is a server-side JavaScript platform built on Chrome's JavaScript runtime that is based on Google's V8 engine [24].

Node was first presented on JSCONF Berlin 2009 by the creator Ryan Dahl [34]. It was designed to build high fast, scalable network applications focusing on performance and low memory consumption. The applications are lightweight, efficient data intensive, real time, and across distributed devices. The important feature of Node.js is the event-driven non-blocking Input/Output (I/O) model, also called asynchronous.

Formerly, the I/O characteristic was executed like a calling function i.e. it did not advance until the operation was finished, this blocking was problematically and not efficient. The alternative appeared with the multithreading,

sharing memory among threads within the same process, which solved the blocking issue because when one thread was waiting for a I/O operation another could take over the Central Processing Unit (CPU). However, the programmer is not aware which threads are being executed at a given time, therefore the developer needs to be highly careful with the concurrent access to the shared memory state, otherwise can easily lead to random and difficult bugs to discover.

The event-driven non-blocking I/O, one of the defining characteristics of Node, offers a more efficient, scalable and control environment alternative for switching the context between function calls [24, 78, 79]. The conceptual model is depicted in Figure 2.2 and handles many concurrent requests on a single process/thread-deadlock free. Node.js has three main components: the Event Queue, the events to be executed on the Event Loop; the Event Loop, the main interaction of the program; and the Thread Pool, asynchronous I/O API. The program starts and the Event Loop pulls out the code from the top of the Event Queue and execute it. If the code needs I/O operations it will be delegate to the Thread Pool and the Event Loop proceeds taking the next code on the top of the Event Queue. In the meantime when the I/O is done, the Thread Pool sends the results to the bottom of the Event Queue. To sum up, the Event Loop is a single thread running, that takes events of the queue and which can be code to be executed or callbacks from I/O.

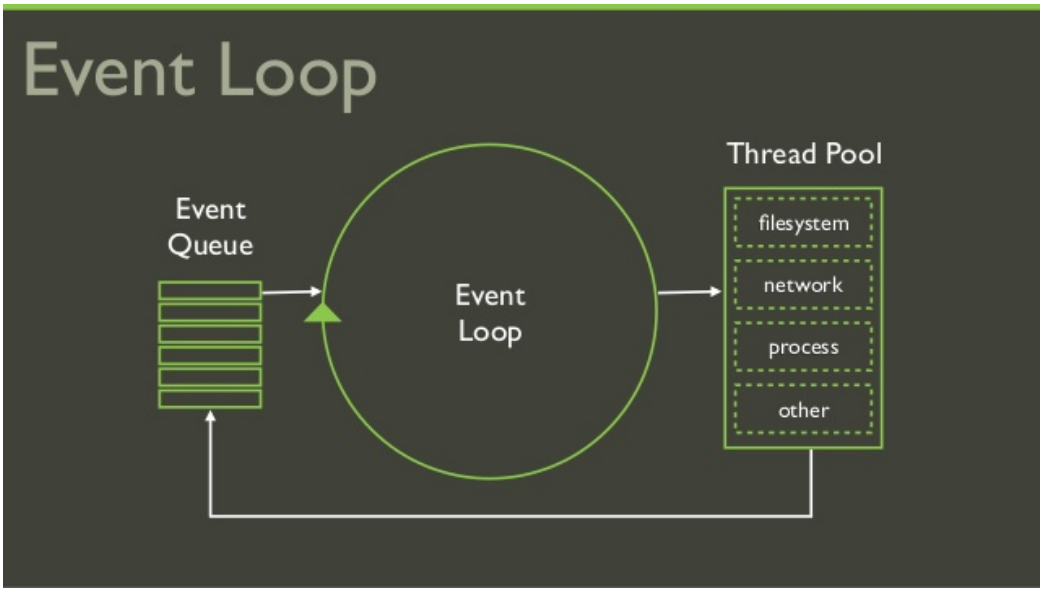


Figure 2.2: Node.js Jeff Kunkle presents the Node.js event-driven conceptual model [78]

Node embeds JavaScript as the programming language, because JavaScript has closures and first-class functions which assembles a powerful tool for event-driven programming.

The power of JavaScript and the unique characteristic of asynchronous programming is why Node become so popular and successful, replacing Rails, in its short time of existence, from the most popular GitHub repository [78]. Moreover, Node has integrated Node Package Manager (npm) [25] which is an online repository of open-source of Node.js projects. It has several node.js libraries as seen as applications, additionally, it is a command-line utility that aids in package installation, version and dependency management.

Hapi

Hapi.js (usually called Hapi) [10], is an open-source framework for building web applications and services that is built on top of Node.js. Hapi allows to build a server quickly because of its wide range of configurable options. Hapi enables developers to focus on writing reusable application logic instead of spending time building infrastructure [10, 71].

Hapi.js can easily implement all RESTful services using the different HTTP methods to manipulate data. It has 100% code coverage, integrating small modules allowing rich and highly specialized functionality.

Hapi.js was created in Walmart by the team of Mobile Platform under the leading of Eran Hammer. It is interesting that Walmart team behind hapi.js has also been contributing to Node.js development in the past. Consequently, developers fully understand how to control node power in their framework from the ground up. An important aspect of Hapi.js is the community of contributors, so that the framework can expand and evolve for other organizations under more open governance. In fact, Hapi.js is widely use in m-commerce, and in companies like Mozilla, PayPal, Disney, Node Packaged Modules, Walmart, Yahoo.

This year, on EmpireNode 2014 conference, Ben Acker talks about Hapi [49] and reveals that the developer team was only defined in 2011 and the first version of Hapi came out on April 13, 2013. From day one, Hapi redid all mobile services of Walmart, but the deployment peak was on the black friday when Hapi handle smoothly all the mobile traffic in that congested day generating much lesser problems than other deploys.

MongoDB

Web-based systems require storing data for sessions, state handling and object persistence. DBMS is divided in two groups: relational – Structured

Query Language (SQL) – and non-relational – "Not Only SQL" (NoSQL) – . NoSQL appeared to address the weak points of relational databases by being schema-less, non-relational (eradicates relational models) and horizontal scaling (scale out). It can be classified by four categories [66]: Key-Value stores; Document databases or stores; Wide-Column or Column-Family stores; and Graph databases.

MongoDB is the leading NoSQL database [20] and it is identified as simple to use and perceive yet, other alternatives prevail such as: Redis, Apache CouchDB, Cassandra, and Neo4J among others, categorize by Key-Value, Document, Wide-Column and Graph, respectively. Though, it does not exists a better DBMS than other. Many studies and tests have been done around which database gets the best performance, yet the conclusion obtained depends on the service requirements so, the software architect must carefully evaluate it and choose the data model accordingly.

For instance, Li and Manoharan [63] proved that, not all NoSQL perform better than SQL databases, differs from each operation. These authors tested on MongoDB, RavenDB, CouchDB, Cassandra, Hypertable, Couchbase and MS SQL Express databases. They acquired on the operation reading that, when it was required 10 readings, MongoDB was the fastest but, when the number of operations increased to 100000, Couchbase turned to be the fastest. MongoDB and Couchbase were in average the fastest two for the operations read, write and delete. Nonetheless, to fetch all keys MS SQL EXPRESS was in average the fastest.

Yet, van der Veen et al. [81] compared one SQL database (PostgresSQL) and two NoSQL databases (Cassandra and MongoDB) and stated no single database was the best in the two proposed cases scenarios. MongoDB was better in single writes and PostgresSQL was better in multiple reads. The tests were done in a single machine thus, do not test database scalability which probably Cassandra would performed better than the other two databases.

In addition, Parker et al. [70] examined SQL versus MongoDB (NoSQL) and obtained that MongoDB has better runtime performance than SQL regarding to simple queries, inserts and updates. However, the paradigm changes it and SQL performs better than MongoDB when it is necessary to update and query non-key attributes and aggregates queries.

So, after meticulous consideration MongoDB was the preferential DBMS. MongoDB (from *hu**mongo**us*) is an open-source powerful cross-platform database, which provides high performance, high availability and automatic scaling [19].

High-performance data persistence through the feature indexes and the support for embedded data models; high availability by the replication facility

(called replica sets) that provides data redundancy and automatic failover; and automatic scaling offers automatic sharding and replica sets as a solution for horizontal scaling.

The core difference between MongoDB and a relational database management system (RDBMS) is the underlying data model. A document-oriented is distinguished from a relational database because it replaces the concepts of "table" and "row" to "collection" and "document", respectively. This allows establishing a more flexible model and simplifies the process of splitting up stored data across multiple servers. Moreover, MongoDB does not have predefined schema per collection or per document, allowing document's keys and values of variables types and sizes thus, enables a rapid development and ease of schema modification.

A document is the basic unit of data; a collection consists of its own JSON documents; and a database contains multiple collections. A single instance of MongoDB can host multiple independent databases. Thus, MongoDB database structures data into collections of JSON documents which are represented in a binary-encoded format called Binary Script Object Notation (BSON) for efficiency in encoding and decoding data. BSON is a binary-encoded serialization of JSON [4].

MongoDB was released in 2009 by the founders Kevin P. Ryan, Dwight Merriman and Eliot Horowitz.

JSON

JavaScript Object Notation commonly known as JSON is a lightweight simple open standard for data-interchange format [16, 43].

JSON text is serialized on data objects and arrays. Object structure comprises zero or more name/value pairs and an array structure involves zero or more values. The value is a string or an number or an object or an array or one of the three literal names: true, false or null.

This simple minimal self-describing textual data-interchanging format rapidly become popular among web services and document databases, that use, store and return data in JSON format. As it is subset of JavaScript, it is widely used in JavaScript programming. Moreover, JSON is uncomplicated and simple for both humans – to read and write – and computers to parse and generate.

JSON offers a good alternative to XML because it is faster, lighter and less verbose.

The creator was Douglas Crockford and in 2011 become a standard and an official document.

2.3.3 Front-end

As for the technologies at the client side, are widely deployed: HTML, CSS, and Javascript. Their availability is a powerful combination to develop Web applications.

HTML

The HyperText Markup Language (HTML) is the de facto standard publishing language for the World Wide Web (WWW or W3) [12, 40]. Also, HTML is one pillar of web design, providing the structural layer which describes the content of a web page.

HTML consists of two major components: hypertext, that is text showed on electronic device with references/hyperlinks to one another; and markup language, that is, a mark or a tag in a document, indicating a logical structure. Therefore, HTML documents are strictly organized in which every part of the document is differentiated, declared and enclosed in specific tags in such a way that all web browsers are able to read and display them correctly. HTML supports images, audio, video and allows objects to embedded and interactive forms.

HTML was established in 1993 by World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG) and currently both of the creators have been working jointly on new HTML specifications namely HTML5.

CSS

Cascading Style Sheet known as CSS [5], is another pillar of web design. It provides the presentation layer and specifies how the content is displayed to the user, through a clear separation between document content and layout.

CSS consists of two major components: the styling rules, concerning about the style, font, colours, layout, and line spacing among others; and the selectors for binding these styling rules. In addition, it provides flexible methods for controlling presentation, such as the width of the screen distinguishing between a smartphone, a tablet or a traditional desktop browser.

Using only HTML for a web page makes its format unattractive and sometimes unintelligible, because the default styles of the browser are applied. Instead, CSS enables complex and consistent styling, for example, multiple HTML pages can share the same formatting, reducing the complexity and avoiding repetition in the structural content.

CSS is also a W3C standard, developed by Håkon Wium Lie and Bert Bos in 1996, turning up a robust and powerful web design tool.

jQuery

jQuery [14] is a fast, small, cross-browser and concise JavaScript Library that simplifies the client-side scripting of HTML documents by reducing the program complexity and using less code. With jQuery is possible to select Document Object Model (DOM) elements; create animations; handle events; and develop AJAX applications.

Additionally, jQuery has an easy and robust built-in method for extending its functionality via plugins, i.e. any programmer can create a useful function and write a plugin for further reuse.

jQuery is a free open source licensed by MIT and was developed by John Resig in 2006. Its rapidly success comes from companies like: IBM, Amazon, Dell, Twitter, among others. For instance, Microsoft has included jQuery in Visual Studio and Nokia has integrated it into the Web Runtime widget development platform.

jQuery changed the manner how developers write JavaScript.

2.3.4 Contactless Technologies

This section discusses contactless communication technologies providing new services with just a simple tap, which can provide remote and local access to everyday things in the physical space.

QR Code

Barcode is a fast and simple method to track accurately and efficiently products. Quick Response Code (QR Code) [29] is a two-dimensional barcode that encodes and decodes data at a rapid rate in both, vertical and horizontal direction.

The information stored in a QR Code can be a URL to a website, a contact information, a image, a video, a product information, an advertising, an email, and so forth.

QR Code is a matrix-type with a cell structure arranged in a square which is usually presented with black and white dots (Figure 2.3), capable of encoding the same amount of data in approximately one-tenth the space of a traditional barcode. Furthermore, QR Codes are readable from any direction, omnidirectional, even if the symbol is partially dirty or damaged. The error correction feature based on Reed-Solomon Code [75] is capable to restore the data until 30% of the dirty or damaged symbol .

Nowadays QR Codes have become so common that it is possible to find them everywhere, on billboards, vending machines, magazines, food prod-

ucts, airplane tickets and so on.

The versatile use of QR Codes have been also implemented as a safe authentication system by combining the QR Code and the OTP or TOTP [59, 69]. Our pervasive service adopts this approach.

General public interact with QR Codes via the smartphone which behaves as a barcode scanner, specifically, through a mobile app that uses the camera of the mobile device.

QR Code was created by the Japanese corporation Denso Wave in 1994, in order to aid the manufacturers tracking vehicle parts. However, the QR Code become popular outside the automotive industry then in June 2000, QR Code was approved as an ISO international standard (ISO/IEC18004).

Figure 2.3: QR Code for the URL of the service "Book Room & Unlock Door"

NFC

Near Field Communication (NFC) [23, 27] is a standard-based short-range wireless two-way communication, and a successor of Radio Frequency Identification (RFID).

NFC technology has been growing rapidly because of its characteristics to exchange safely contactless data between devices in close proximity using magnetic induction, which in a simple way enables to exchange digital content without difficulties for standard people. Contactless payments and ticketing applications are the most popular use cases of NFC.

There are two defining attributes of NFC. First, a short range connection that is low-power and avoids interference with radio signals received by the device. Second, it is designed for short messages and not for high-speed communications, contrary to WiFi and Bluetooth.

Nevertheless, NFC avoids the inconvenience of pairing, exchanging passwords and the binding steps common from other protocols. Moreover, NFC can identify and initiate larger amounts of data transfer between two NFC-enabled devices by switching or handover to Bluetooth or WiFi. NFC Connection Handover Specification is designed to handle larger transfer files by seeking for better transfer medium like WiFi or Bluetooth through, only exchanging few messages without pairing or passwords needed.

NFC devices can be active or passive. An active device is considered when has internal power supply so can send and receive data, like a smartphone. A passive device does not have internal power supply thus only can send data, such as a smart card or tag.

The NFC communication is accomplished when a device creates a low frequency radio-wave field and another device gets close enough to contact the field, magnetic inductive coupling. The inductive coupling causes a passive device to absorb energy (power itself) from an active device, once powered up the passive device communicate and exchange the data. However, NFC communications can involve two active (powered) devices so in this case both generate their own radio frequency fields and act as active and passive devices

Also, the NFC technology harmonizes diverse contactless technologies allowing broad solutions supporting three operational modes based on ISO/IEC 18092, NFC IP-1 and ISO/IEC 14443 contactless smart card standards. First, the read/writer mode where the device reads the NFC tag. Second, the peer-to-peer (P2P) mode, where two NFC-enabled devices communicate with each other. Third, the card emulation mode where the NFC-enabled device acts like a NFC tag. The examples are: reading a product information; exchange photos or business cards; and updating seat information while boarding, respectively.

NFC Forum [27] was founded in 2004 by Sony, Nokia and Philips to promote and develop the technology. Later, in 2006, the group produced the first set of specifications for NFC tags.

2.4 Authentication

Pervasive services need to be secured and support by suitable authentication and authorization mechanisms in order to improve their user experience.

2.4.1 TOTP

One-Time Password (OTP) [54] is a password that can be used only once. OTP blocks passive replay attacks or eavesdropping – the attacker eavesdrop the authentication information though is not able to reuse it –, a major advantage in contrast to static passwords.

Time-Based One-Time Password (TOTP) is a special form of OTP, i.e., an algorithm that generates a password that can only be used once on a short period of time [68]. TOTP is an extension of One-Time Password (OTP) that is, adds the time-based moving factor meaning short-lived OTP values, which enhances a stronger security.

The algorithm computes the time-based one-time password from a shared secret key and the current time having as options the encoding and length of the password and as well the time-step expiration (the time between new passwords). This last option is recommended to be 30 seconds to achieve a balance between security and usability.

One objective for the designed algorithm is to be used as a two-factor authentication, that has seen growing adoption by cloud-based systems and used by banks and major Internet companies like Facebook, Twitter, Google, Amazon and Paypal strengthening the authentication.

The method to deliver the TOTP as a two-factor authentication could be by: (1) a phone call or sms, the server generates the TOTP and after calls or sends to the user's cellphone; (2) a mobile phone application like Google Authenticator, the user's application device generates the TOTP and shows it; (3) hardware tokens, a dispositive with a display that generates the TOTP and displays it.

The last two deliver methods have a prior sharing of the secret seed with the server.

TOTP was developed by Initiative For Open Authentication (OATH) became officially in 2008, it is an approved standard of the Internet Engineering Task Force (IETF) since 2011.

2.4.2 OAuth

OAuth is an open standard enabling secure authorization across the web, that is, it coordinates a secure authorization between the end user and the service provider to allow third-party applications, such as desktop and mobile applications, to access the protected resource [26, 55]. For example, a user grants to a phone application access to the Google Contacts without inserting the Google credentials in the phone application.

OAuth has also been used for authentication. Formerly, the traditional

client-server authentication model, end users shared their credentials username and password with the third-party applications, OAuth emerges to address compromisingly credentials.

OAuth have been widely used by popular companies like Facebook, Google, Twitter, Yahoo, GitHub for user authentication and authorization. For that reason both, service providers and end users, save time and achieve secure authorization, because if the service provider, namely websites or applications, provides via OAuth a login-plugin for user authentication, users in fact, can login without sharing their username and password to the service provider, neither they have to create another account and remember an additional username and password. On the other hand, providers do not need to store and validate user credentials, resulting in less resources required for verification purposes.

OAuth was created in 2006 by Blaine Cook.

Chapter 3

Implementation

This thesis main objective is a practical use of pervasive services for flexible spaces working on new reservation system for meeting rooms based on website for accessibility and secure authentication procedures.

To this end, a new service named "Book Room & Unlock Door" was launched at Aalto Computer Science Library, which has shared spaces in the form of a flexible environment. The project is provided by the cloud and third party servers, allowing interactions between a smartphone, an electronic lock and a digital sign.

A pervasive service project incorporates smart technologies (e.g., devices and appliances, sensors, wireless networking), and services always available anytime anywhere to support human behavior improving people's daily individual and group activities. According to Cook and Das [46] this pervasive or ubiquitous computing is an exciting area of computing in recent times, depending not just on the development of mobile devices, but also on the development of automated interactions, whereby devices interact with their peers and the networking infrastructure without any administrator. The process to provide transparent interfaces between disparate entities in the environment, enhancing invisibility, is an integral part of pervasive environments [61].

Therefore, the logic of our approach for pervasive service is a cloud infrastructure, fully automated system based on an electronic lock, and focused on users devices (smartphones) and modern technologies. Our work is to represent and interpret the users needs providing a service for meeting rooms.

Currently, the existing reservation systems for shared rooms face significant limitations. Specifically, access control is not enforced, thus actors can not effectively measure the space utilization, and it is possible for unauthorized people to occupy rooms (room squatting). Even though the Aalto CS Library is a smart environment, the meeting room reservation system is still

paper-based with all the above-mentioned problems. Thus, our challenge is to offer a suitable new service for flexible spaces.

To deploy new pervasive services there is need to enable easy and secure technical configuration replacing paper and ink in most applications [77].

With these requirements in mind, we will begin to outline the reference scenario in this chapter, followed by the system architecture and implementation, by detailing the core and service functionalities and their interactions.

3.1 Reference Scenario

In the reference scenario we will design the booking flow, list the equipment – the user and software prerequisites – and the reference use case.

3.1.1 Booking Flow

The process that replaces the 'old' with the 'new' service is illustrated in Figure 3.1. The previous service at Aalto University library was configured manually in a paper form and the door was always unlocked ('old' in Figure 3.1), which in our sophisticated everyday environment it was becoming quite displeasing. The next step (e.g. 'new' in the figure) is towards the creation of an automatic service. To implement it in a pervasive scenario we need to rethink the service architecture, and following Quitadamo et al. [74] the services must be implemented on-demand and capable of automaticity (self-management), self-reconfiguration and situation-awareness.

This service integrates an user-assisted booking flow shown as the 'new' part in Figure 3.1 with the following step-by-step procedure:

1. First, to start the booking flow the 'User' needs to have a device with an Internet connection and a browser;

2. Second, the user types `http://fss.cs.hut.fi:8888/` to access the reservation 'Web Page';

3. Third, the user in the web page chooses one of the three options: 'Facebook', 'Twitter' or 'Google' to 'Sign in'. This page links to the log in account in the third-party server. If user has not yet been granted access, 'Authentication' and 'Authorization' are required, username and password are prompted as well as the authorization dialog. Otherwise, if user has already been granted access to the service, the user's browser receives an access token and he or she will skip the fourth step;

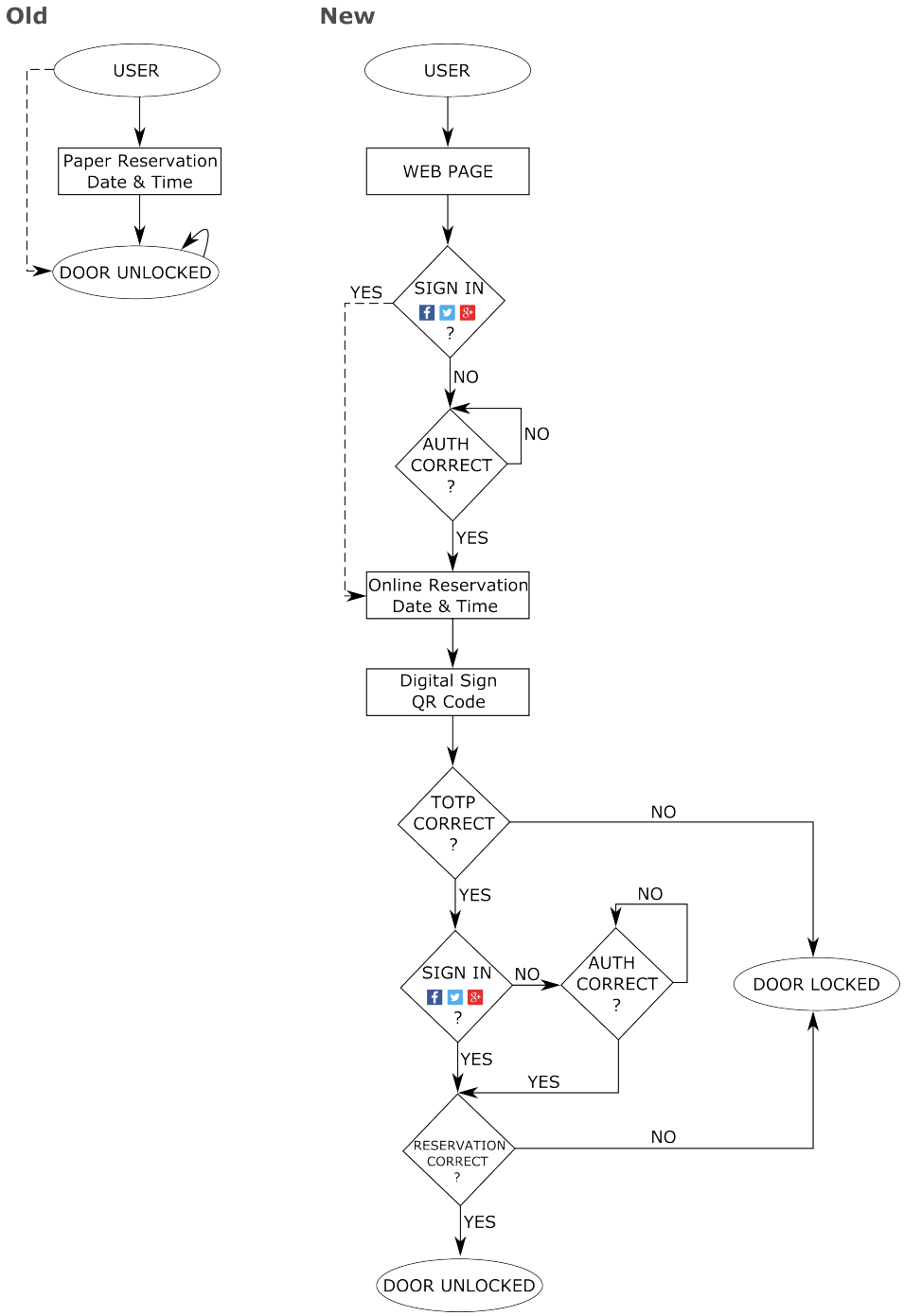


Figure 3.1: Booking Flow – Replacing the 'old' with the 'new' service

4. Fourth, if the Authentication is Correct with username and password and the application authorization was allowed, user's browser will receive access token and continues on to the next step. If not, the authentication/authorization procedure has to be repeated;

5. Fifth, upon successful authentication/authorization, the service allows the user to start the 'Online Reservation' system, where he can book a 'Date & Time' for a given room;

6. Sixth, the user scans the 'Digital Sign'. That is, the user with a mobile phone camera scans the QR code advertised in the tablet, which is integrated in the door of the given room;

7. Seventh, the server upon receiving 'TOTP' key from the digital sign scan, starts the verification. If 'Correct', the server initiates next verification. If wrong,'Door Locked';

8. Eighth, the user's browser links to 'Sign in' page, he or she must select the same account (Facebook/Twitter/Google) used for the reservation in order to continue. If the user has not yet been granted access, the fourth step is repeated. If the user has already been granted access he or she proceeds on to the next verification;

9. Ninth, if 'Reservation Date & Time Correct' the service continues. If wrong, the 'Door Locked' will remain;

10. Tenth, the booking flow terminates with 'Door Unlocked'.

This context-adaptive workflow could be represented by a user' s experience in two stages (shown in Figure 3.2):

1. At home or any location: The user with a portable computer, tablet or smartphone accesses the "Book Room & Unlock Door" webpage and logs through Facebook, Twitter or Google. He or she books the slot to the space room online. If the user uses the same account and device for the service, he or she only needs to log in once.

2. At the library: the tablet on the door always advertises the URL through the QR Code. The user smartphone scans it and the door will open. From the user's point of view, the only requirement is the knowledge of the user reservation account. From the server side, it will check the following:

    i. The validity of the TOTP key;

ii. The access token from the authentication;

iii. The validity of the reservation.

If all the conditions above are met then the server sends a command requiring to open door, otherwise the door remains closed and the appropriate error will appear on the user's screen.

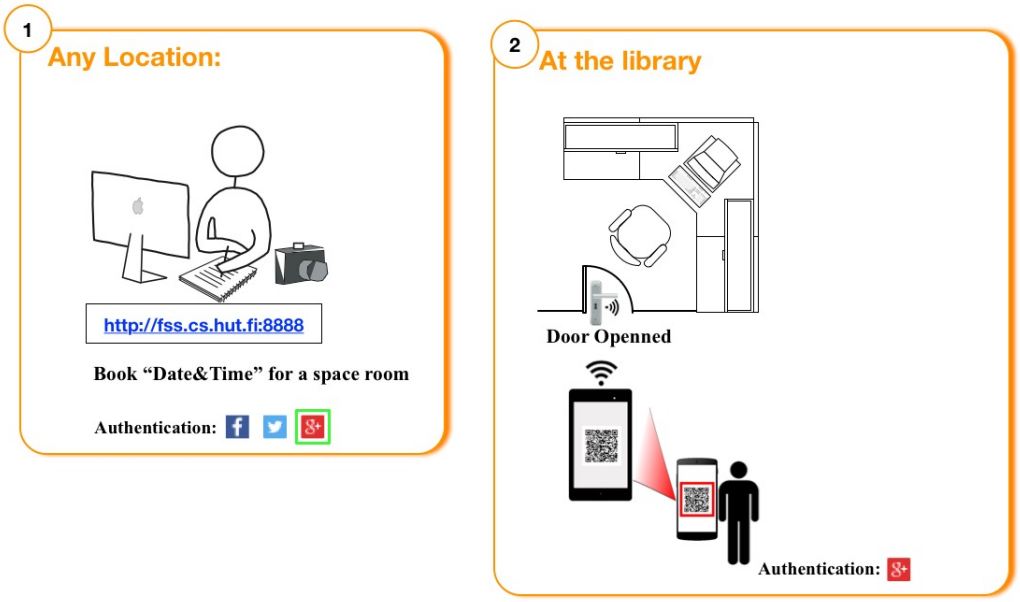


Figure 3.2: User context-adaptive workflow

3.1.2 Equipment and Service Requirements

In this subsection, the equipment needed for the project and the user prerequisites is mentioned, as well as the software requirements:

1. Equipment used for the project:
   (a) One Mohinet Gearlock [18].
   (b) Two testing phones (LG Google Nexus 5 [21] and Samsung Galaxy S5 [9], Android version KitKat 4.4) to simulate the user and a tablet Asus Google Nexus 7 [22] (Android version Jelly Bean 4.3) to advertise the QR Code and communicate with the electronic lock.

2. Equipment required by the user:
   (a) One smartphone that has:
      i. Internet connection;
      ii. Browser;
      iii. QR Code reader application.
3. Software required and installed for the project:
   (a) The project is stored on one micro instance created on Pouta, a cloud computing infrastructure provided by the Finnish IT center for Science (CSC) for research purposes, through the Infrastructure-as-a-Service (IaaS) service model.
   (b) The "Book Room & Unlock Door" machine is accessible through the public IP 86.50.168.113.
   (c) The running operating system is Ubuntu version 13.10 (Saucy Salamander).
   (d) The machine (Table 3.1) has two web servers running on port 8888 and 9999 namely website server and door server, respectively. For simplicity the service diagrams depict only one server namely Reservation server.
   (e) Both servers have Node.js installed and Hapi.js running on top.

All the information and code related to the project as well the prototypes are available on a private repository in Github.

Machine	**IP**	**Domain**
"Book Room & Unlock Door"	86.50.168.113	fss.cs.hut.fi
Server		**Port**
Reservation	Website	8888
	Door	9999

Table 3.1: "Book Room & Unlock Door" IP, Domain and Ports

3.1.3 Practical Case

In this section we describe an example in development in a flexible space service to demonstrate the challenges of the dynamic nature of a smart environment. Assume that a user requests a service to make a reservation for

rooms in the Computer Science Library at Aalto Campus.

Rita is an Aalto student agrees meeting with other colleagues at the CS library rooms to prepare a presentation for a course. Rita picks up her smartphone with Internet access and types `http://fss.css.hut.fi:8888` *and logs in through her Facebook account. By visiting this web page she books the Tomaatti meeting room for the date and time wanted.*

On the meeting day, in front of the room, Rita and colleagues can automatically open the door. First, Rita with her phone scans through the camera phone the 2D barcode (QR code) displayed by the tablet next to the meeting door. Then, she logs in with the same user account she used to make the reservation. Upon successful authentication she pulls down the door handle and accesses the room.

To develop this pervasive service (reservation and access system) for flexible spaces (meeting rooms at Aalto Campus) the following questions must be based on user contextual information gathered in the smart space (location-awareness) meeting:

Where is the library room available?

When is the library room free?

Is the room door open or closed?

Is the room occupancy adequate?

3.2 Architecture

This section details a practical architecture that allows mobile users to access a service provided by a pervasive environment. Research in pervasive computing has the contribution of many projects [82] allowing different architectural supports thus, it is difficult to provide a generic architecture. For programming pervasive spaces we need to create multiple middlewares [57] to hide the underlying system's complexity, consequently in the user's perspective it is a hard-wired process binding a friendly interface, besides, the system architecture is quite extent and complex. For instance, Oat et al. [69] proposed a system architecture for public displays which adopts web-based technologies that supports multiple modes of interaction based on physical proximity for easy configuration.

The architecture diagram of the project is illustrated in Figure 3.3 which comprises a cloud computing infrastructure where all the backend functionality is processed and the website is hosted; an electronic lock binds with

a digital sign tablet (through the local network where the user accesses the room); and a user smartphone. The electronic lock and digital sign tablet are installed in a public (or private) space.

The web-centric architecture built in the service seems to be the most versatile and functional choice by easily combining smart devices, cloud services and lock hardware from different enterprises. It also eliminates the cumbersome process of installing an application. Moreover, it does not overload the phone's memory with an heavy application.

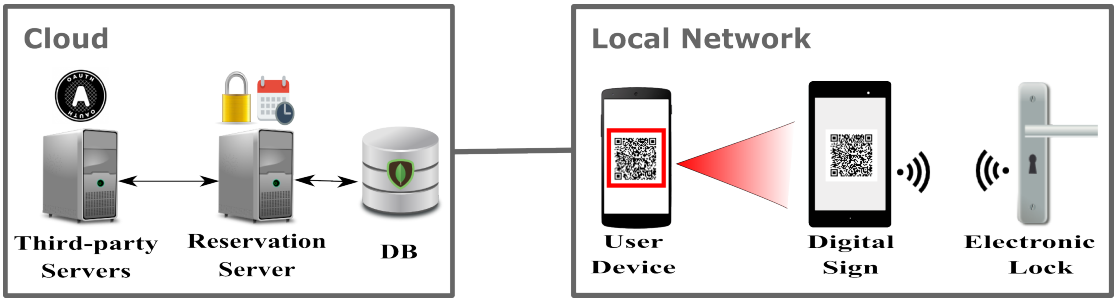


Figure 3.3: "Book Room & Unlock Door" system architecture

On the one hand, there is an *electronic lock* easy to install and manufactured by a Finnish company called Mohinet. The lock module has a Bluetooth chip to handle the door access control by receiving input commands from the Digital Sign. Another lock characteristic is that it is battery powered and just activates when necessary allowing more than 20 thousand openings or lasts five years [60]. The careful choice on the lock resulted in not requiring a new door. Also, the lock works as a pass-through, thus allowing the user in case of a problem use the physical key and in case of removal does not leave a trace.

A *Digital Sign* is in the first phase is an interface for the user to start and interact with the service by advertising a QR Code URL that changes continuously preventing a person from opening the door from outside the location. In the second phase it guides the user through the process by displaying instructions and in the last phase it communicates with the electronic lock sending an 'Unlock Door' message. After ending the cycle, it will start again by advertising the QR Code.

A *device* needed by the user to perform the steps. The user's device is assumed to have Internet connection, a browser and a QR Code reader application installed. The user will utilize the device to reserve a date and time for the room and to unlock the door.

On the other hand, the *reservation server* processes, controls and verifies all the requests and operates accordingly. The reservation server is comprised by one machine with two web servers running (website and door server). The website server manages all the decisions related to the website making it possible to book a room, check the available hours and spaces, administrate the users and provide the data set to the users. The door server manages the constant QR Code updates, the space utilization by tracking the room occupancy and also validates the user to access room. Both servers, website and door, verify the user authentication through a third party server. The reservation server is in continuous communication with the database and the third-party server.

A *database* stores the data sent from the reservation server. The database keeps all the information about the users, their reservations and their logs to the room which provides the information when a user books a room and does not use it. The database helps to predict future context values and establish trends by using the historical information.

Third-party servers are external trusted servers that secure authorization and authentication between the user and service. The third-party servers are interrogated by the website and the digital sign controls the door access to the room, denying unauthorized person access to the service or room by enabling the user identification.

Our architecture satisfies the main design goals of *Pervasive Services for Flexible Spaces* by splitting functionality into components through smart devices, sensors, service deployment in the cloud, location awareness and context data.

3.3 Core Functionality

In this section, we describe all the service foundation as well as the technologies applied for a lightweight service with an unrivaled performance, illustrated in Figure 3.4.

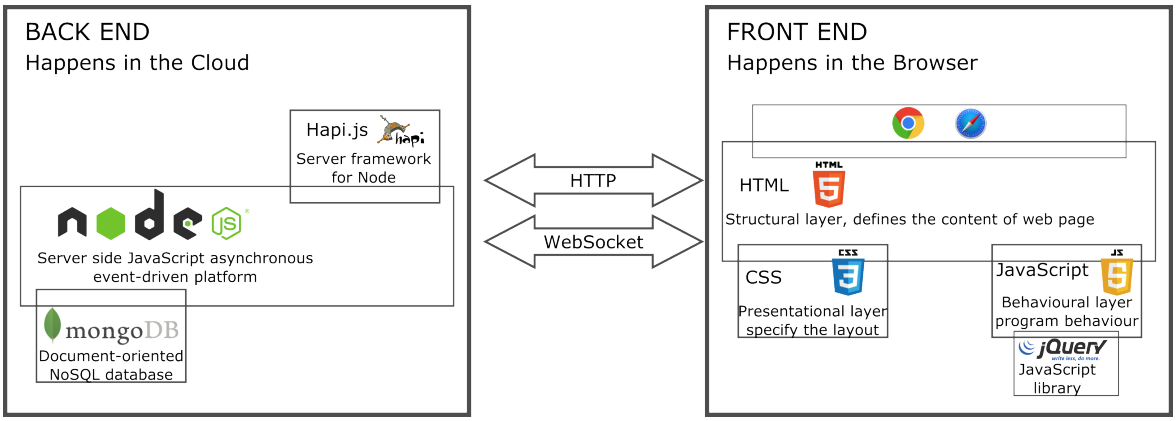


Figure 3.4: "Book Room & Unlock Door" core functionality

On the one hand, we have front end development – website – developed by standard technologies like HTML5, CSS3, JavaScript and jQuery which are supported by Google Chrome and Safari browsers as the interface between the user and the back end.

On the other hand, the back end is a cloud-based distributed system with Hapi.js server framework built on top of Node.js and MongoDB the support database.

3.3.1 Cloud

Server

The server is the engine of our proposed service in which all the steps directly or indirectly pass by the core server hence it relates to all the components of the system.

The system has two independent web servers maintained in a cloud, sharing an equal domain with different ports (Figure 3.1). The idea of having two independent servers running was if for instance the website server was down due to overload, suffered from a malicious attack or a cut on the power supply, it would not going to impact negatively the door server that is the user was still able to access the room without having knowledge the website server was dealing with difficulties. However, due to limited resources the two web servers are running in the same machine blocking the flexibility and scalability of the system.

All the server code was implemented in JavaScript language applying Hapi.js framework which runs on top of Node.js.

The service has encrypted cookies, it utilizes iron [13] to encrypt and sign the session therefore, the cookies are highly secure and it is very difficult to forge them (any modification will invalidate the cookie).

The service adopted the authentication and the basic cookies.

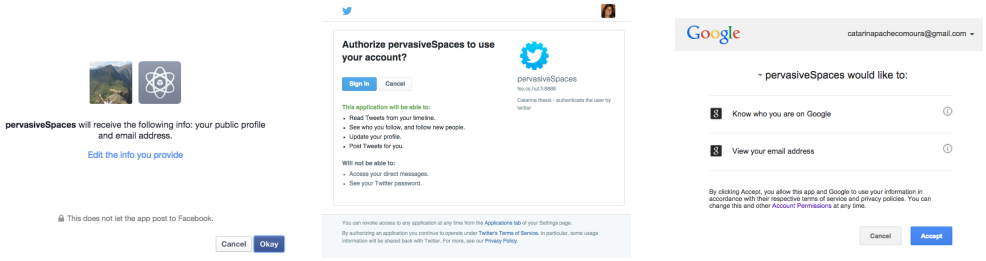
Authentication cookies, illustrated in Figure 3.1, were implemented to simplify and decrease the time of user's login. The user upon a successful authentication at the login page receive a session cookie. The development goes through assigning the 'hapi-auth-cookie' or 'cookie' scheme [2] with a 'strategy' session (lines 1 and 2). Later, with the scheme and strategy assigned, the password is set to encode the cookie (line 3), the cookie is named 'aaltoLibReservation' (line 4), unauthenticated requests to routes that apply the 'session' strategy are redirect to '/login' (line 5) and isSecure is set to false allowing the cookie be transmitted in http connections (line 6).

The following pages, reserve, my reservations, user and logout from the website server (Figures 3.7b, 3.7c, 3.7e) and the access the room page from the door server, are only possible to enter with the authentication cookie 'aaltoLibReservation' valid.

```
1 server.pack.register(require('hapi-auth-cookie'), function
      (err) {
2   server.auth.strategy('session', 'cookie', {
3     password: 'XXXXX',
4     cookie: 'aaltoLibReservation',
5     redirectTo: '/login',
6     isSecure: false
7   });
```

Listing 3.1: Example of cookie authentication

The third-party providers namely Facebook, Twitter and Google also use cookies for their authentication so when the user accesses to our service for the first time but already had logged on the third-party provider in other situation, the user does not need to type the provider credentials to authenticates in our service, however because it is the first time, the user needs to authorize our service on the provider (Figure 3.5). The user next time is not prompted with the service authorization dialog until he or she revoke the access to it.



(a) Facebook (b) Twitter (c) Google

Figure 3.5: From the third-party providers the user authorizes the "Book Room & Unlock Door" service

Furthermore, the basic cookies where implemented at the sysadmin page (Figure 3.8) with a cookie time expiration of one month or until the administrator logout or delete manually the cookie; and at a valid QR Code giving the user two minutes to successful access the room until it expires.

Database

The system uses a non-relational database management system based on MongoDB, a document oriented database. The structure of the database is shown in Figure 3.6.

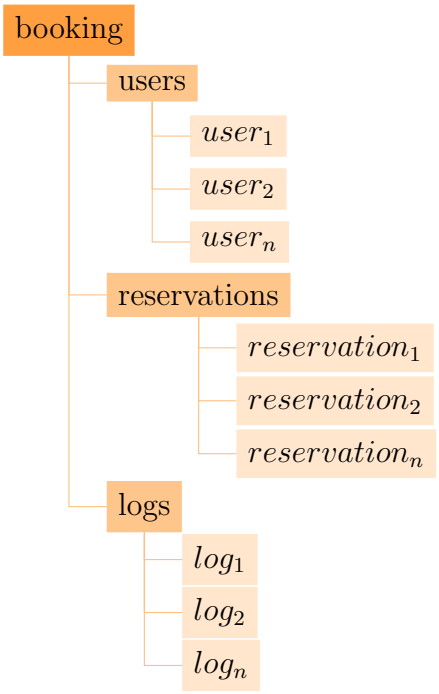


Figure 3.6: "Book Room & Unlock Door" DMS structure

The *booking* database has three collections namely: *users*, *reservations* and *logs*, which they have the role to store the users, reservations and logs information, respectively. Each collection respectively has multiple *user* documents, *reservation* documents and *log* documents.

Each document of the database is schema-free and an object in JSON format, following below an example of a document of each collection.

The user document (Listing 3.2) is an object that has the following fields:

- *_id* an unique identifier ObjectId type of the user;
- *name* a string that displays the user name;
- *providerId* an object that contains:
  - *provider* a string that presents third-party account selected by the user which can be Facebook, Twitter or Google;
  - *id* the identifier of the user from the provider account.
- *email* a string with the user email which it is only provided by Facebook and Google accounts;

- *link* a string with embed url of the user profile which it is only provided by Facebook and Google accounts;
- *picture* a string with embed url of the user picture which is only provided by Twitter and Google accounts.

```
{
        "_id": ObjectId("544884efb4062eb760b36b23"),
        "name" : "Catarina Moura",
        "providerId" : {
                "provider" : "google",
                "id" : "113201971281970431707"
        },
        "email" : "catarinapachecomoura@gmail.com",
        "link" : "https://plus.google.com/+
   CatarinaMourapt",
        "picture" : "https://
   lh4.googleusercontent.com/-agYcFszxTj0/
   AAAAAAAAAAI/AAAAAAAAANg/FpeVjpn_98U/photo.jpg"
}
```

Listing 3.2: Example *user* document

The reservation (Listing 3.3) document is an object that has the following fields:

- *_id* an unique identifier ObjectId type of the reservation;
- *date* an user reservation full date in ISODate type on MongoDB but manipulated in milliseconds at the server;
- *userid* a user identifier of the reservation ObjectId type.

```
{
        "_id" : ObjectId("5451efd714dcf02a2a42fc48")
   ,
        "date" : ISODate("2014-10-30T08:00:00Z"),
        "userId" : ObjectId("
   544884efb4062eb760b36b23")
}
```

Listing 3.3: Example *reservation* document

The log document (Listing 3.4) is an object that has the following fields:

- *_id* an unique identifier ObjectId type of the log;
- *userId* an user identifier of the reservation ObjectId type;
- *name* a string with provider account "f (Facebook) or t (Twitter) or g (Google)" and the user name of the reservation;
- *date* an ISODate with date of the reservation to enter the room;
- *timestamp* an ISODate with date that user enters the room.

```
{
        "_id" : ObjectId("5451f698f1151ec92edda4a2")
    ,
        "userId" : ObjectId("
    544884efb4062eb760b36b23"),
        "name" : "g: Catarina Moura"
        "date" : ISODate("2014-10-30T08:00:00Z"),
        "timestamp" : ISODate("2014-10-30T08:28:
    08.577Z")
}
```

Listing 3.4: Example *log* document

All the documents have an unique *_id*, generated by MongoDB. The db guarantees it being unique by creating an ObjectId with 12-byte BSON which 4-byte represents seconds since the Unix epoch, 3-byte the machine identifier, 2-byte the process id (pid) and 3-byte counter starting with a random value.

The 'Booking' database was designed and structured to achieve optimal results by providing fundamental information without unnecessary additional queries or preventing server overload through conditional expressions (if) and search algorithms in arrays. It also, provides a fast response to a query. For instance, the 'Booking' db collections are clean and coherent in which the 'reservations' and 'logs' collections have one pointer each for the user to make the queries clear and simple contrary to collections that have excess of fields or pointers.

3.3.2 Website

The website is the user first step (as mentioned in Section 3.1.1) in which accesses it through `http:fss.cs.hut.fi:8888`. As illustrated by Figure 3.4 in this section, Core Functionality, the site has been designed for the Chrome and Safari web browsers.

The website construction utilizes HTML5, CSS and JavaScript/jQuery for definition of the content (structural layer), layout web page (presentational layer) and user interactivity (behavioral layer), respectively. These web technologies are standard achieving responsiveness and simplicity by creating a friendly environment.

The website is organized (Table 3.2 and Figure 3.7) by a *home* page where the user is able to understand how the service works through a sequence of three images explaining the main steps.

A *reservation* page where it is possible to visualize the available time to book the space and to reserve.

A *my reservations* page where it is possible to see the reservations done by that user account and delete it if desired.

A *map* page where it displays the room location address and the respective map with a marker containing the specific room,address, latitude and longitude. The map was integrated via JavaScript through Google Maps API [17].

A *user* page where the user profile is presented.

A *login* page where the option to the user selects the account to sign in is given.

A *logout* option for the user to sign out being able to sign in with a different account or the same.

Page	**URI**	**Figure**
Home	*/* or */home*	3.7a
Reserve	*/website/reservations*	3.7b
My Reservations	*/myreservations*	3.7c
Map	*/map*	3.7d
User	*/user*	3.7e
Login	*/login*	3.7f
Logout	*/logout*	–

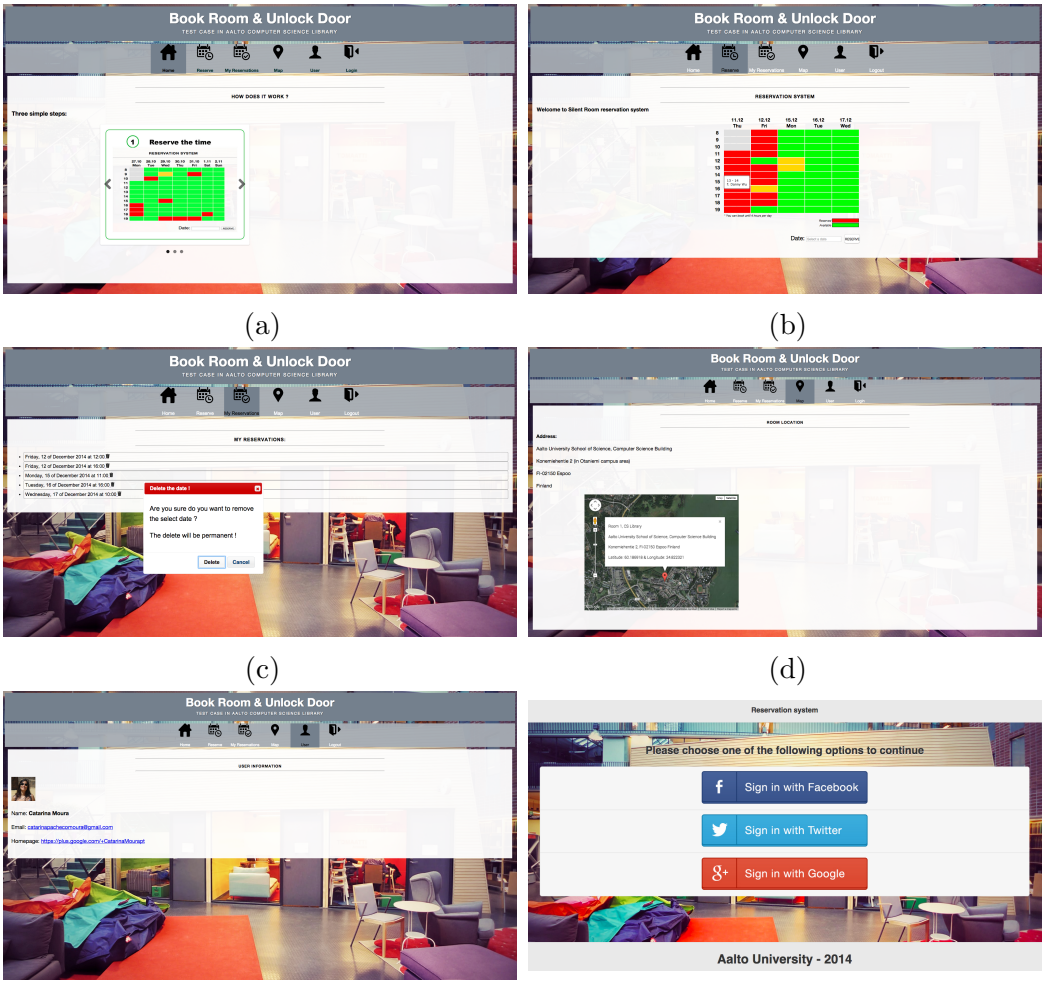
Table 3.2: "Book Room & Unlock Door" website information

The CSS is visible in the webpage layout where title box, navigation box and content box with CS library picture in the background were created. The navigation box has in each tab the picture and the name, it was designed to change to green and white letters when the mouse is on the tab and to a darker grey when the tab is selected. In addition CSS is also responsible for the website adaption in different screens width, this feature was only

introduced on the navigation bar. Screen width bigger than 600 px – *@media screen and (min-width: 600px)* { ... } – the icon and text appears in the tab and smaller than 600 px – *@media screen and (max-width: 600px)* { ... } – just the text (no icon) appears in the tab.

JavaScript/jQuery also produced some interesting details such as on the 'Reserve' tab in the timetable the green square symbolizes available, when the user selects it modifies to yellow and when the user clicks on the 'Reserve' button it changes into red. When the time is already past it turns into grey blocking and not allowing the user to select it, instead an information box warning the 'Hour expired!' appears. Furthermore, other fine points were implemented in the reservation page like disables the 'Reserve' button when any 'date & time' slot is not selected preventing the user from reserving in blank. In addition, when the mouse is on a reserved square a box who reserved and from which account is displayed, disappearing immediately when the mouse is over it. Finally, besides providing reserve for the actual week, the user also has the option to select the day from the calendar enabling schedule a time distant from now like two weeks, one month, one year later.

The information box and the calendar are jQuery plugins, Tooltip [32] and Datepicker [6] respectively. More jQuery plugins are possible to find in our built web site namely flexslider [8] in the 'Home' tab or dialog [7] (by clicking in the trash icon) in 'My Reservations' tab.



(a) (b) (c) (d) (e) (f)

Figure 3.7: "Book Room & Unlock Door" website pages

Sysadmin page

Besides the common access to the website, an additional page is provided to system administrators to track the room access logs, that is displays the information which and when the user accessed the room (Figure 3.8).

The administrator obtain the information at `http://fss.cs.hut.fi:9999/sysadmin` where needs to type an username and the respective password once (Figure 3.8a). If the credentials are correct the administrator is forwarded to the logs page (Figure 3.8b at `http://fss.cs.hut.fi:9999/logs`) which has access to the user identification database (User Id), the displayed name and the provider account used for the reservation (Name), the date and time the

reservation was made (Date), and the precise time the user enter the room (Timestamp). Through this data it is possible to evaluate the utilization rate of a room and analyze which are the users that book a room and later do not occupied it.

(a) System Administrator page

Logout

Door Access Logs

Show 10 entries Search:

User Id	Name	Date	Timestamp
545b873f169306753d7b3168	g:	Fri Dec 12 2014 10:00:00 GMT+0200 (EET)	Fri Dec 12 2014 10:10:33 GMT+0200 (EET)
548aae634bba708273ff82ba	g:	Fri Dec 12 2014 11:00:00 GMT+0200 (EET)	Fri Dec 12 2014 11:00:41 GMT+0200 (EET)
548aae634bba708273ff82ba	g:	Fri Dec 12 2014 12:00:00 GMT+0200 (EET)	Fri Dec 12 2014 12:35:23 GMT+0200 (EET)
54858e32ec4407c567b04b53	g:	Fri Dec 12 2014 13:00:00 GMT+0200 (EET)	Fri Dec 12 2014 13:09:52 GMT+0200 (EET)
54889dce3fdcbd883566efa2	g:	Fri Dec 12 2014 18:00:00 GMT+0200 (EET)	Fri Dec 12 2014 18:05:04 GMT+0200 (EET)
544f4eb70863fa8944b585bf	g:	Fri Dec 12 2014 19:00:00 GMT+0200 (EET)	Fri Dec 12 2014 19:19:37 GMT+0200 (EET)
544f4eb70863fa8944b585bf	g:	Thu Dec 18 2014 08:00:00 GMT+0200 (EET)	Thu Dec 18 2014 08:34:12 GMT+0200 (EET)
5492b5d8c44038a31c556aa0	f:	Thu Dec 18 2014 13:00:00 GMT+0200 (EET)	Thu Dec 18 2014 13:09:29 GMT+0200 (EET)
5492b5d8c44038a31c556aa0	f:	Thu Dec 18 2014 13:00:00 GMT+0200 (EET)	Thu Dec 18 2014 13:10:50 GMT+0200 (EET)
544884fab4062eb760b36b25	f:	Thu Dec 18 2014 19:00:00 GMT+0200 (EET)	Thu Dec 18 2014 19:43:13 GMT+0200 (EET)

Showing 471 to 480 of 490 entries First Previous 1 … 45 46 47 48 49 Next Last

(b) Door Access Logs page

Figure 3.8: Tracking users accesses to the room

3.3.3 Communication between the components

The service relates to technologies and protocols for the communication between the components depicted in Figure 3.9.

In the service the user accesses the reservation website through the Internet. The interactions between the user and the website are coordinated by REST API through HTTP protocol and it complements socket.io [30]. To connect and access the database the plugin hapi-mongodb [11] was installed and the data is sent and receive in documents in JSON format. In the local network the constant changing of the QR Code is provided by the web server and acquired by the tablet through socket.io communication. The QR Code

technology is scanned by the user which enables the Hapi.js server to check user veracity and provide instructions on the user mobile, also send to the tablet through socket.io the message: authentication is valid. Subsequently the tablet addresses the lock via Bluetooth.

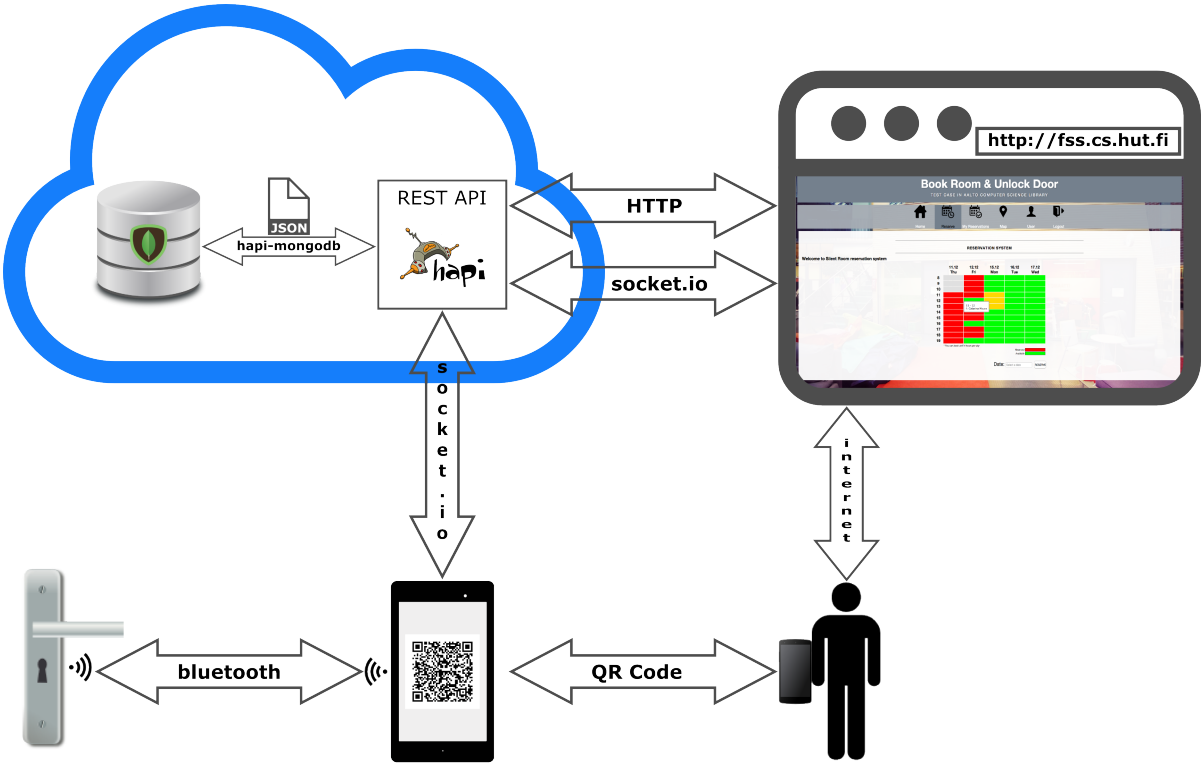


Figure 3.9: Communication between the components

The underlying communication between the mobile client and the web server and between web server and tablet is socket.io. Here is an example how websocket was implemented in our service. In this particular case it is showed the server receiving an user information request from the client, subsequently the server sending the request data.

- On the web server (Listing 3.5): the socket.io module was imported (line 1) and the socket was initialized by passing the HTTP server object (line 2). The socket will listen to the 'connection' event for incoming sockets (line 3), if the incoming socket is emitting the 'getuser' event (line 4), the web server will emit event named 'userinfo' to the connected socket (line 5) with the user information data (url

picture,name,email and homepage link).

```
1 var socketIO = require('socket.io');
2 var io = socketIO.listen(server.listener);
3 io.on('connection', function (socket) {
4   socket.on('getuser', function () {
5     socket.emit('userinfo', user);
6   });
7 });
```

Listing 3.5: Example of socket.io communication on web server side

- On the browser side (Listing 3.6): it loads the client library (line 6) and listens to the 'connection' event for incoming sockets (line 11) since the default from the client side to connect is the host that serves it is not necessary to add host URL. Whereas the webpage was requiring promptly the user info so emits a 'getuser' event (line 12) to the web server and listens to the 'userinfo' event for the requested data (line 13).

```
1  <!DOCTYPE html>
2  <html>
3  <head lang="en">
4  <meta charset="UTF-8">
5  <!-- socket.io-->
6  <script src=/socket.io/socket.io.js></script>
7  </head>
8  <body>
9  <script>
10 $(document).ready(function () {
11   var socket = io.connect();
12   socket.emit('getuser', 'User Info');
13   socket.on('userinfo', function (user) {
14         (...)
15     });
16  });
17 </script>
18 </body>
19 </html>
```

Listing 3.6: Example of socket.io communication on client side

- The output communication between web server (Listing 3.5) and the web client (Listing 3.6) results in displayed user information illustrated in Figure 3.10.

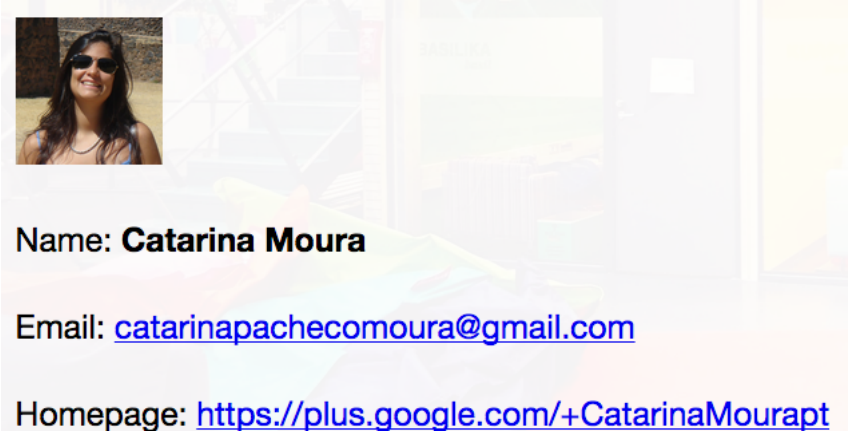


Figure 3.10: The output as the result of the communication between the web server and the web client with socket.io

Building a REST API with Hapi.js is simple and intuitive, therefore the communication between the webserver and webclient is also done with HTTP protocol and not only by websocket.

An implementation example of the RESTful web service on our service is shown in Figure 3.7 which loads the hapi module (line 1) and creates a new web server object running at 8888 port (line 2). Later this server route responds to a 'GET' request to '/' path with a redirect to '/home' path (lines 3 to 9). The route path '/home' handles request by rendering the 'home.html' page (lines 11 to 17) indicated in Figure 3.7a.

```
1  var hapi = require('hapi');
2  var server = hapi.createServer(Number(8888));
3  server.route({
4    path: '/',
5    method: 'GET',
6    handler: function (request, reply) {
7      reply.redirect('/home');
8    }
9  });
10
11 server.route({
12   path: '/home',
13   method: 'GET',
14   handler: function (request, reply) {
15     reply.file('home.html');
16   }
17 });
```

Listing 3.7: Snippet of RESTful web service implementation

3.4 Service Functionality

Security and location-awareness are essential for a Flexible Space therefore in this section we will detailed the implementation of the QR Code in the Digital Sign (Authentication Phase I) and the Third-party Servers (Authentication Phase II).

3.4.1 Authentication Phase I

Authentication Phase I is the second step in the user experience (namely, 'At the library' as mentioned in Section 3.1.1) which the user initiates by scanning an advertising the QR Code in the tablet through a smart device. The QR Code changes dynamically, thus making it secure and location-aware. These core characteristics are because the QR Code embeds an URL which contains TOTP key varying every 30 seconds. The URL:

```
http://fss.cs.hut.fi:9999/advValidation?totp= <actual_totp_key>
```

The query attribute 'totp' is the changing part and consequently the QR Code modifier.

The server, in the first phase, is responsible for securely generate a TOTP key, next embed in URL, previous displayed and later generate the QR Code with URL. At last, it emits by websocket the QR Code to the webpage displayed in the tablet (Figure 3.16a). This process is a cycle that repeats itself every 30 seconds.

Secondly, the server is also responsible for verifying the TOTP when the smart device reads the QR Code and opens the corresponding web page. The TOTP verification is done by verifying the actual TOTP key and the previous one. If the TOTP attribute in the URL is not equal to the actual or the previous TOTP key the server displays in the user phone the following message "QR Code invalid. Please scan again", otherwise it redirects the user to the login page (Authentication Phase II). Note the two authentication phases are a very quick process that lasts less than 10 seconds for the user to execute (considering that the user already did the Authentication Phase II once). The server imports a 'qrcode' module which aids in the QR Code creation and local 'totp' file that is detailed subsequently.

TOTP Algorithm

TOTP was implemented in a separate file in which it exports two core functions (*getActualKey* and *getPreviousKey*). The file imports the 'speakeasy' module [31].

The *getActualKey* and *getPreviousKey* functions obtain the actual and the previous (thirty seconds before from the actual time) TOTP key, respectively. Both functions are used by the server to verify the TOTP validity of the user URL and the *getActualKey* function is also used by the server to embed in the url and advertise it on QR Code.

These two functions employ a private function to get the key by calling it with a 'time' parameter. The private function has the following attributes (Listing 3.8 line 6 to 10):

- privateKey: a secure key generated randomly in ASCII format with twenty characters of length and enabled symbols (beyond *A-Z, a-z, 0-9, ?, _ , . , etc...*) stored in a local variable (line 2);
- timeExpiration: set to 30 seconds the recommend time-step size from RFC 6238 [68];
- time: the time parameter received from the *getActualKey* and *getPreviousKey* functions;
- length: the key length set to 8 numbers;
- encoding: the key encoding format set to ASCII.

```
1
2  var privateKey = speakeasy.generate_key({length: 20,
       symbols: true});
3
4  function totpAlgoritm(timeKey) {
5    totpKey = speakeasy.totp({
6      key: privateKey.ascii,
7      step: timeExpiration, // 30 seconds
8      time: timeKey,
9      length: 8,
10     encoding: 'ascii'
11   });
12   return totpKey;
13 };
14
15 totp.getActualKey = function () {
16   var timeKey = parseInt(Date.now() / 1000); // in seconds
17   return totpAlgoritm(timeKey);
18 }
19
20 totp.getPreviousKey = function () {
21   var timeKey = parseInt((Date.now() - ((timeExpiration) *
         1000)) / 1000); // in seconds
```

```
22    return totpAlgoritm(timeKey);
23  }
```

Listing 3.8: TOTP algorithm

3.4.2 Authentication Phase II

Authentication Phase II is used in the two stages of the user's experience (from 'Any Location' and 'At the library' as mentioned in Section 3.1.1) in order to identify and authorize the user.

Authentication Phase II was implemented by using the OAuth [26] protocol and third-party, providers namely, Facebook, Twitter and Google. The underlying model is depicted in Figure 3.11.

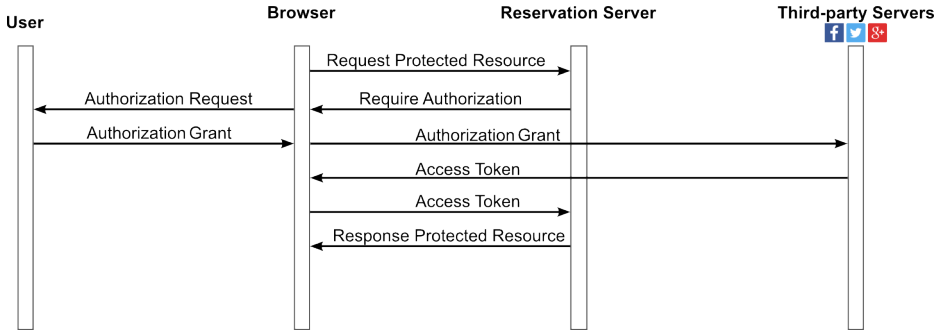


Figure 3.11: Authentication Phase II underlying model

The process begins when through the *browser* from a desktop or a smartphone the user requests a 'Protected Resource' by trying to make a reservation or to access the room. The main server, *reservation server*, answers with 'Require Authorization'. Thus, the *browser* requests an authorization, 'Authorization Request', from the *user* by showing the login page (Figure 3.7f) in the browser. The *user* accepts, 'Authorization Grant', by selecting the provider account and typing the respective username and password in a trusted server, the *third-party server*. Then, the *third-party server* (authorization server) authenticates and validates the user, and if valid, issues an 'Access Token' to the *browser*. Subsequently, the *browser* authenticates by presenting the 'Access Token' to the *reservation server*. Upon the valid access token the *reservation server* provides access to the 'Protected Resource'.

```
1
2  server.pack.register(bell, function (err) {
```

```
3
4       server.auth.strategy('facebook', 'bell', {
5         provider: 'facebook',
6         password: 'XXXXX',
7         clientId: 'XXXXX',
8         clientSecret: 'XXXXX',
9         isSecure: false
10      });
11
12      server.route({
13        method: ['GET', 'POST'],
14        path: '/loginf',
15        config: {
16          auth: {
17            strategy: 'facebook',
18            mode: 'try'
19          },
20          handler: facebook
21        }
22      });
23  }
```

Listing 3.9: Hapi.js authentication based on the bell scheme and the facebook strategy

To enable the third-party login from the providers (third-party servers) a Facebook, a Twitter and a Google application were created to set the access and callback URL; and to get the client identification (consumer key) and the client secret (consumer secret) for then configure in the reservation server.

In the reservation server the third-party login was managed by a Hapi plugin called bell [3]. Authentication in Hapi is based on schemes and strategies which in our case a bell scheme and three different strategies (facebook, twitter and google) were configured. An example using the bell scheme is depicted in Listing 3.9 where the facebook strategy was applied (line 4 to 10) by setting up: a *provider*, the third-party provider in this case facebook (line 5); a *password*, the cookie encryption password (line 6); a *clientId*, the OAuth client identifier provided by the Facebook application (line 7); a *clientSecret*, the OAuth client password provided by the Facebook application (line 8); and an *isSecure*, set to false because our service provides the http protocol and not the https (line 9).

Later, the facebook strategy is applied at the resource '/loginf' (line 14 and 17) that is the user in order to continue needs to login via the Facebook provider page.

3.5 Service Interaction

The service interaction is summarized in two distinct phases from 'Any Location' which the user reserves online the 'date & time' slot for the room and 'At the Library' which the user accesses the room, Figure 3.12 and 3.13 respectively.

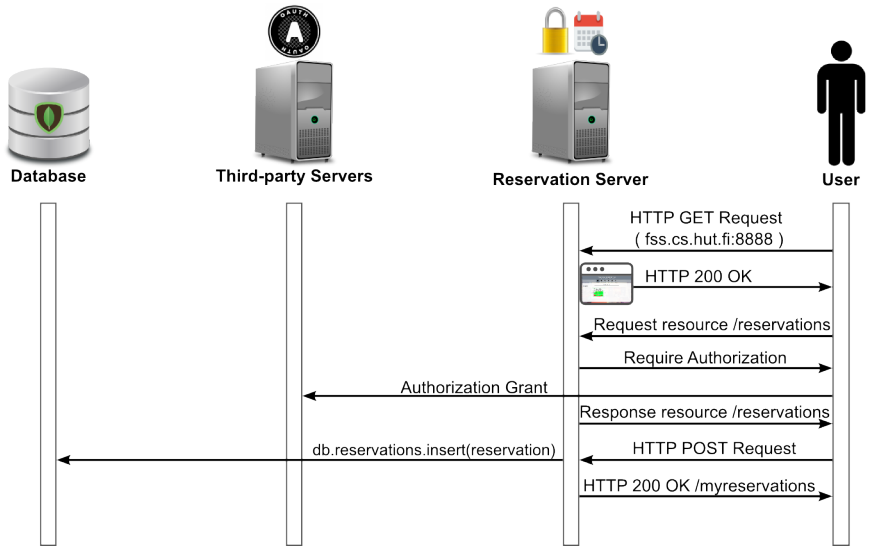


Figure 3.12: Service Interaction Phase I 'Book Room'

To successfully complete the first phase, 'Book Room' the following messages are exchanged:

1. The user browser's sends a 'HTTP GET Request' to access the website page `http://fss.cs.hut.fi:8888`.

2. The reservation server responses the request has succeed with 'HTTP 200 OK' and transmits the resource in the message body.

3. The user in order to make the reservation 'Request resource /reservations', a protected resource.

4. The reservation server responds with 'Require Authorization'.

5. The user grants the authorization, message 'Authorization Grant', to the third-party server by signing in.

6. The reservation server provide the access to the reservations page, 'Response resource /reservations'.

7. The user by accessing the reservations page, books a 'date & time' slot, 'HTTP POST Request'.

8. The reservation server, stores the reservation in the database, 'db.reservations.insert(reservation)'.

9. The reservation server also responds to the user by transmitting the reservation was successful stored by fetching the page my reservations, message 'HTTP 200 OK /myreservations'.

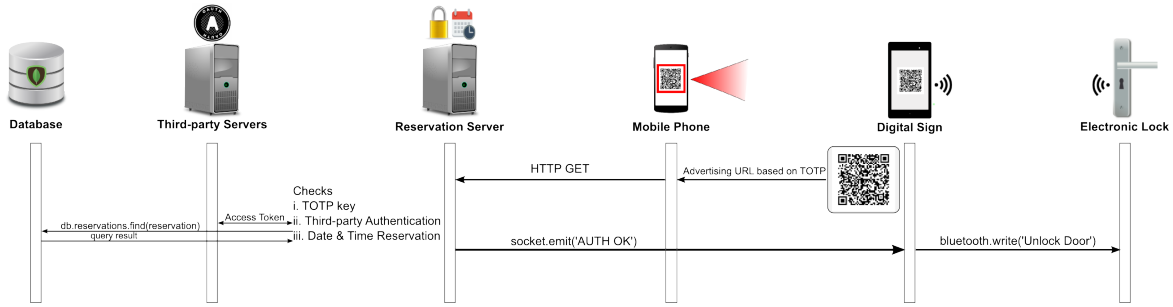


Figure 3.13: Service Interaction Phase II 'Unlock Door'

At the second phase, 'Unlock Door' the following messages are exchanged:

1. Digital sign is 'Advertising URL based on TOTP'.

2. The user mobile phone's scans the QR Code and sends an 'HTTP GET' request to the reservation server.

3. The reservation server in turn verifies the 'TOTP key' internally; the 'Third-party Authentication' by exchanging the 'Access Token' with third-party servers; and 'Date & Time Reservation' by querying the database.

4. If all the conditions are satisfied the reservation server transmits it to the digital sign by WebSocket, 'socket.emit('AUTH OK')'.

5. Finally, the electronic lock receives from the digital sign a message by Bluetooth to unlock the door, 'bluetooth.write('Unlock Door')'. Thus, the user accesses the room.

3.6 Pilot

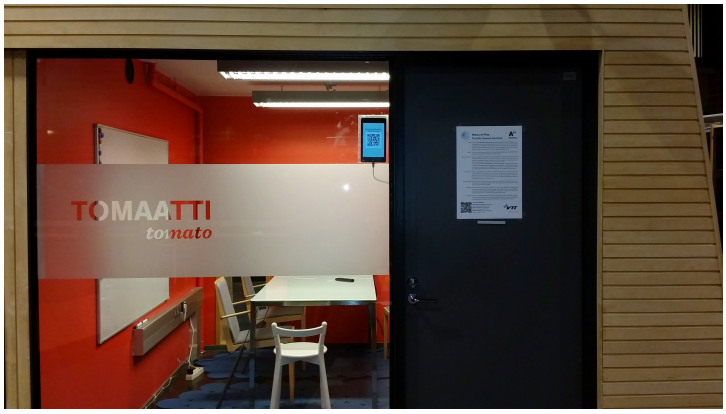
In this section, we describe the pilot [37] of our proposed service as part of the Flexible Spaces Service project. The system has been piloted at the "Tomaatti" a quiet study room in the Computer Science Library of Aalto University between November and December of 2014. The users after concluding using the pilot were proposed to fill an online questionnaire about the service for feedback whose results are not yet available.

The project stakeholders are the Aalto Campus and Facility Services, the Aalto University Properties and the Mohinet Oy enterprise. The pilot environment has an open network which enables all the users to have free access to the WiFi. The conditions are similar to those in the practical deployment, where we set it up, according to the Figure 3.14.

The pilot was deployed in "Tomaatti" room (Figure 3.14a) with an electronic lock (Figure 3.14b) installed on the door and a tablet (Figure 3.14c) mounted in the window room.

The participants in the pilot access the service "Book Room & Unlock Door" from any location (Figure 3.15a). First, the participant signs in with one of the following accounts: Facebook, Twitter and Google (Figure 3.15b). Second, the participant makes a reservation by clicking on the reservation tab and selecting an available 'date & time' slot. Finalizing it by clicking on the 'Reserve' button.

The participant accesses the room reserved by appearing at the door on the same 'date & time' scheduled. Second, the participant mobile scans the advertising QR Code (Figure 3.16a). The scan will open a window on a browser with the corresponding URL. The web page prompts the options of the login accounts (Figure 3.16b). The participant must choose the same login account as the reservation made. If all the conditions are met the participant receives "Access Granted" on the mobile phone and starts to follow the instruction on the tablet installed in the window (Figure 3.16c). The tablet instructs the participant to "Press down the door handle". Only then is the Bluetooth of the door activated and the communication with the tablet (i.e. the waiting process in Figure 3.16d, first image) begins. The door is unlocked and the participant accesses the room by pulling down the door handle (Figure 3.16d).

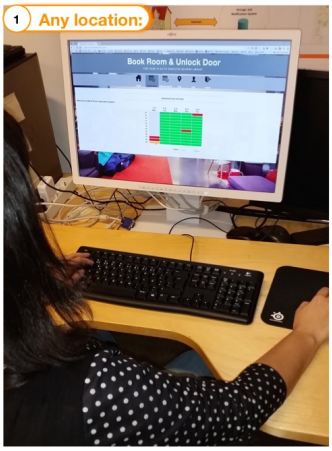


(a) "Tomaatti" room

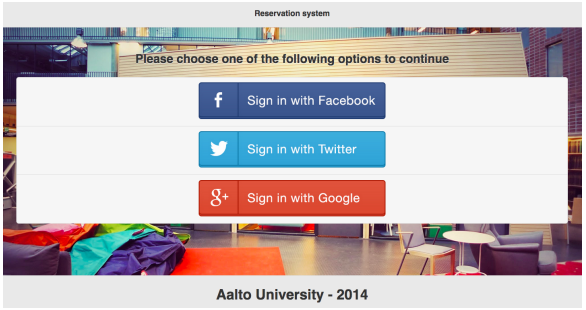
(b) Electronic lock

(c) Tablet with advertising QR Code

Figure 3.14: Local where the pilot was deployed



(a)



(b)

Figure 3.15: First phase of the user experience from 'Any Location'

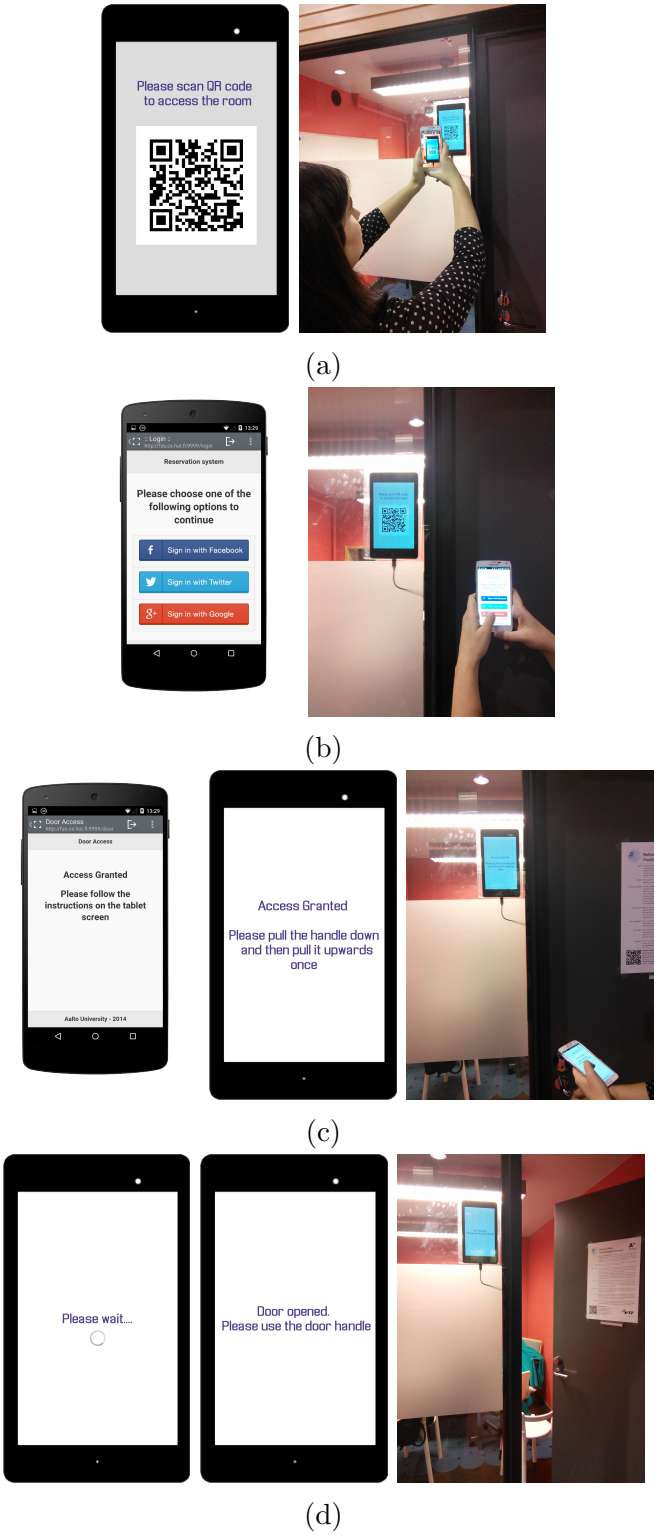


Figure 3.16: Second phase of the user experience at 'The Library'

3.7 Summary

In this chapter we have presented our implementation of a pervasive service for a flexible space, more specifically we developed the "Book Room & Unlock Door" service and installed successfully the system in the Computer Science Library at Aalto University.

The project began with the study of the previous system and the design of the new one that intended to be a functional, autonomous and simple service in a shared environment. Moreover, the chapter presented the service requirements and explained in detail the service architecture by analyzing each technology used. The novel and robust technologies used at the service contributed to be simple to expand and develop further.

Finally, the "Book Room & Unlock Door" implementation opened new ways to solve the problem that was longed faced in companies, organizations and in facilities that have meeting rooms.

Chapter 4

Business Model

A business model is an instrumental tool in both justifying a project and as measuring its success. Designing viable business cases is difficult as customer needs, organizational resources and capabilities, financial arrangements, and technological possibilities have to be taken into account. Moreover, a business model must be continuously maintained and adjusted to track the project status over time.

The term business model spread out with the advent of the Internet [87] and most of the related research focused on how to extract value from a service innovation, converting new technology to economic value (that is, the utility for customers) or capturing revenue streams for web-based firms.

Morris et al. [67] provides an approach where a business model is a unifying unit of analysis that can facilitate theory development in entrepreneurship. According to Amit and Zott [38] business model construction is to capture value creation arising from multiple sources, concluding that no single theory can fully explain the value creation potential of a venture, so there are many variations to design it.

Some technologies, such as NFC, have been considered promising, it has been having problems to succeed mostly because of the challenging business models necessary to realize meaningful services [58]. It is essential to identify the critical issues in order for a service be commercialized and succeed.

The business goal for this new project is to deliver a system for a practical use of a pervasive service for flexible spaces, which is easy-to-use, inexpensive to deploy for university-like environments, and applies a new technological design focused on authentication and a cloud paradigm.

Our model will present some non-financially features such as the SWOT analysis used to evaluate the Strengths, Weaknesses, Opportunities and Threats involved in the project. After that, we will describe some elementary aspects of the STOF method in Service, Technology, Organization and Finance do-

mains. A schematic overview of financial schemes proposing the service delivery at a reasonable price will follow, and finally concluding remarks about business model viability of this concrete case study will be given.

SWOT has been widely adopted due to its simplicity and practicality allowing agile detection of strengths, weaknesses, opportunities and threats that should be kept, eliminated, exploited and recognized respectively. Nevertheless for an ample view of the full paradigm a complementary steady framework STOF, was also analyzed.

This work was carried out in the Flexible Spaces Service (FSS) activity at Aalto University in collaboration with the VTT Technical Research Centre. The related project focused on the Computer Science Library, which is a single but spacious environment. Although registering is not compulsory, there have been an estimated 45000 students, teachers, staff and visitors entrances (the estimate may be exaggerated because the same person can be counted more than once) at the library and the potential users of the proposed service.

The University has plans to enhance the current system by offering pervasive services. Otaniemi Campus University underlying IT model has on-demand self-service available network with wireless access points installed all over in comfortable public spaces (e.g. cafeterias, libraries, hallways, class, and so forth).

The initiative to test pervasive services in flexible spaces is being held in the University because of the above-mentioned characteristics describing a smart environment and the study of student's experiences is a good role model for a pervasive service project. Our business model performs on the service named "Book Room & Unlock Door" and identifies the issues to become a successful project. The alternative to this service is the traditional paper and ink procedure.

4.1 SWOT Analysis

SWOT analysis is an analytical tool as a part of the strategic management planning process used to evaluate **s**trengths, **w**eaknesses, **o**pportunities and **t**hreats as internal and external factors affecting the project business.

On the one hand, internal aspects lie within the organization, such as personnel facilities, location, products and services, which the effects on business can be favorable identified as strengths or unfavorable identified as weaknesses.

On the other hand, external aspects are presented by the environment, outside the organization, namely, political, economic, social, technological

and competition issues. They either affect positively the organization through business opportunities, or the environment is unfavorable and becomes a threat to the business. The result of the key factors usually is present in the Internal vs. External and Positive vs. Negative quadrants.

The Table 4.1 identifies the internal and external factors for "Book Room & Unlock Door" service on SWOT analysis.

Scenario	Positive	Negative
Internal	Strengths • Availability • Simplicity • Costs • Location awareness • Multiple smart space • Novel technology	Weaknesses • Interaction • User delegation • Experience • Organizational tasks • Brand
External	Opportunities • Real-time and fast service demand • Mobile ubiquity and cloud-based • Untapped market • Growth and R&D	Threats • Target • Supplier dependence • QoS • Reliability

Table 4.1: SWOT Analysis of pervasive services for flexible spaces

A. Strengths

The strengths of this pervasive service for flexible spaces scenario are based on six internal factors:

1. Availability. A favorable internal factor gained over the traditional service is the availability for the user to access it online. The user no longer has to be in the library in order to fill in the booking paper form and carry out processes regardless the service, such as: (a) to review reservation; or (b) to book in advance, among others.

2. Simplicity. Other benefits for the end user are the simplicity and user friendliness. So, the traditional service can be replaced in a very simple and smooth way, bringing value to users, because it does not depend on anybody or anything else and it is a rich web-based interface does not require the user to download or install any application. The service simplicity allows: (a) user self-service; (b) key or card less; (c) application less.

3. Costs. The strength is the low cost, considering the following:

   (a) Initial cost per user. The cost to purchase the mobile device which is needed to access the service, but nowadays it has been reduced and it is quite affordable. In this framework we consider it negligible as users are assumed to already have a smartphone.

   (b) Capital expenditure cost. No costs related to the core network because infrastructure is already installed and available. In this case, low CAPEX is considered because it assumes what we called an "umbrella" contract concerning the network and in relation to the electronic lock installation, the lock does not require a new door or any modification thus it reduces the cost by four times.

   (c) Cost per session. It is a cost already embedded in the entire network maintenance for over-the-top (OTT) services and applications (e.g pay yearly domain name or the cost of cloud instance per hour but in this case computer science department has his own Domain Name Server (DNS) and cloud therefore no cost is add). In this framework these costs are not an issue.

   (d) Other costs. Examples of some of that trigger costs are the ones related to the technical transformation of the traditional system into the new one , and other operations costs link to the space usage and shared administrative resources.

   There are studies [33, 35] demonstrating that corporate real estate costs (fixed rent and variable maintenance) are those that counts considerable for flexible spaces. These costs we must considered low, once the traditional service already integrates spatial costs, so we do not expect significant incremental costs. However, the subject will be more detailed in the next section about financial aspects.

4. Location awareness. This internal factor is achieved by combining the reservation system with room space. So, location awareness is the important driver for the end user to accept the service, and also for space management performance. In our user case the strength is to gather information from the location and the time, extracting data from user identification to further act upon that context, considering the situations:

   (a) Deny unauthorized people;

   (b) Do not allow open the room door without an existing room booking;

(c) The user log is saved for tracking usage. For instance, if the room is booked, but at the scheduled time it is not occupied, then the reservation is automatically (in fifteen minutes) cancelled, also a penalty price could be charged.

5. Multiple smart spaces. A strength of the service is the easy adaptability to different environments. The service can be applied to our use case of meeting rooms in libraries, or other universities spaces, but also for companies and pubic working spaces, among others.

6. Novel technology. Integrating recent technologies that are robust and open source is a way to increase the value of the service. In our case we used Hapi.js, MongoDB. In the short term, technology is also a way to differentiate and potentially increase margins. In the long term, having more experience with the technology becomes a competitive advantage, both are considered strengths.

B. Weaknesses

Internal factors that are identified as weaknesses are the following:

1. Interaction. An unfavorable technical internal factor is the interaction because of the rapid change of technical requirements due to innovation or adaption to other smart spaces, and in all aspects of the service must work. Web services are scalable, however the website, for instance, could be redesigned.

2. User delegation. The service does not currently provide user delegation for multiple people to open the door which can be considered an unfavourable factor for the service.

3. Experience. One weakness comes from less experience with the new service, as it was developed by young researchers working with recent technologies with limited tutorials.

4. Organizational tasks. The service not fully supports. For instance, the rules of use are an important issue, procedures in case of breakdowns, faults, and others needs to change from traditional to new service.

5. Brand. The service is new, so until now does not have brand or market reputation.

C. Opportunities

The identification of some external factors will provide opportunities to build this new service.

1. Real-time and fast service demand. The most favorable external factor is the market, which means end users (customers if paying the service) looking for ways to utilize real-time and fast services. This new service easily creates market offers with its improved transaction speed online, its convenience will also be improved with the automation, and suitable for multi-tenancy.

2. Mobile ubiquity and cloud-based. Opportunities come from technology like a positive and external factor from the supply side of the market. Mobile devices such as smartphones, tablets, PDAs being ubiquitous (e.g smartphone is ubiquitous and it is the substitute product to the physical key) also offer considerable choices for the provisioning of cloud services among enterprises like Amazon Web Services (AWS), Google and Microsoft Azure offering public cloud services.

3. Untapped market. An opportunity comes from the lack of competition on the specific service.

4. Growth and Research & Development. On the one hand, an opportunity for service growth comes from users having become familiar with the service, so they will prefer the smartphone rather than carry a physical key or a card. Nevertheless, it will then be possible to encourage users to adopt more advanced options of the service stimulating R&D.

D. Threats

There are uncertainties that could affect negatively the service. Here, we mentioned some of these threats.

1. Target. One constraint is to target primarily end users with smartphone or tablets. This is a limitation, further developments should integrate other methods and devices with different characteristics.

2. Supplier dependence. This could be another unfavorable external factor because in this first stage of a new business model the service relies on the electronic lock and the university infrastructure. This situation could create a risk where the supplier stops granting the equipment or becomes dependent of the company service.

3. Quality of Service (QoS). Increasing the service complexity and congestion or delay could imply dissatisfaction and issues on the quality of service.

4. Reliability. The system can have outages and it may not be able to solve them immediately, for instance: it does not provide a redundant server; the electronic lock may deplete the battery fail; the tablet may not be replaceable.

The SWOT analysis assessment for "Book Room & Unlock Door" business case shows the service internal strengths which are very important to realize the value proposition for the service, especially the new way of working. In particular, it allows a cost-effective service and suitable for multitenancy, and also the availability and simplicity of the service for the user. These can be combined with technological capabilities to location awareness to prevent unauthorized utilization of resources, and could also be applied to other smart and flexible spaces. However, some weaknesses, such as the user delegation, the developers knowledge, organizational and marketing issues, as well as technical requirements should be very well specified and, if necessary, some of the basic issues should be reconsidered. On the other hand, the model should demonstrate the capacity to adapt to external opportunities in R&D and market approach in terms of real-time and fast service demand, as well as the mobile devices ubiquity and cloud-based service supply are important factors for the business. However, there is the impact of the unfavorable external business environment in relation to consumers target, proprietary solutions and quality of service definitions.

The SWOT analysis can be subjective, so it is advisable to do it in group for a clear classification trying to avoid misleading ideas. The SWOT does not prioritize issues, does not provide solutions and it is a qualitative approach so far. Yet, the SWOT helps to understand the business and develop goals and strategies for it, taking advantage of the strengths, addressing weaknesses to develop into strengths, capitalizing opportunities and preventing threats to turn into opportunities. It can be used to make a quantitative analysis where these factors should be detailed in measurable variables for financial metrics to support the business decision. In Section 4.3 related to financial aspects, we provide some issues in order to propose a quantitative approach to our business case.

4.2 STOF Method

The concept of business models is very useful because it considers value creation for customers/users and value capturing from service development in relation to technology, this is what Bouwman et al. [42] considered when they develop the STOF model to design viable business models primarily for mobile services, that can also be used for digital services.

There are many different classifications of components for business models. In the context of the STOF model, the importance is given to services, innovation, and electronic services in the general context where business models play a central role in generating service innovation. This is a very good approach to our specific service "Book Room & Unlock Door" which is a system designed for a pervasive service to be delivered in a flexible space.

The STOF model uses the STOF method as a practical tool. In this section, we will describe the qualitative approach of STOF however, it also can be used to make a quantitative analysis. In the next section we will present some quantitative financial aspects for the practical case.

STOF Domains

STOF is based on four core domains: Service, Technology, Organization and Finance. It focuses on the service evaluation at an early stage, and each domain is interconnected (see Figure 4.1).

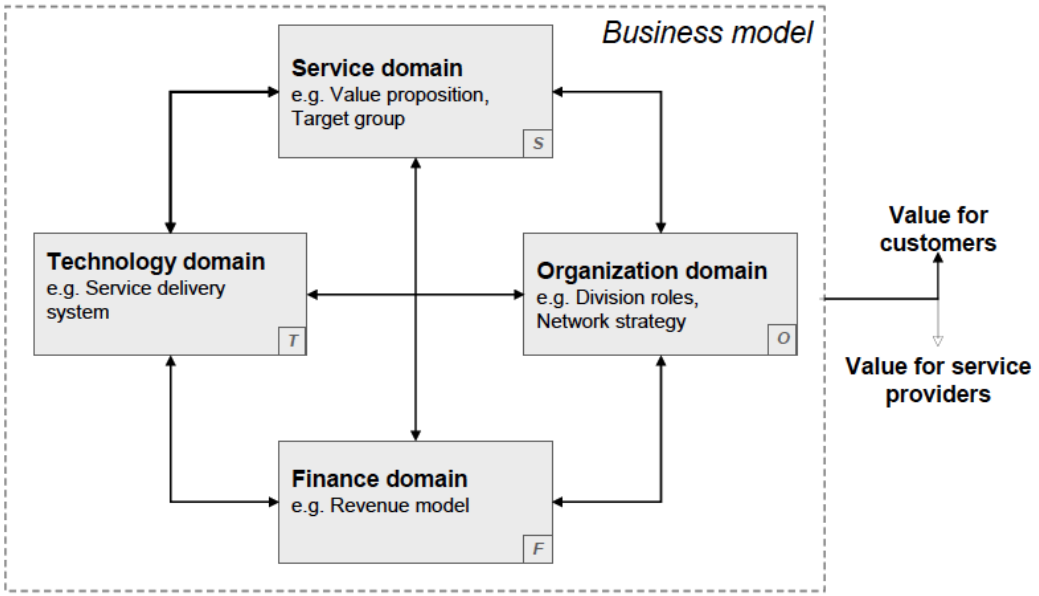


Figure 4.1: STOF business model domains [42]

According to Bouwman et al. [42] the importance is given to the Service domain; it sets the requirements on the technologies used in the Technology domain, it identifies the issues about value network in the Organization domain while affecting the revenue sources of the Finance domain.

Similarly, the Technology domain impacts on the delivered value in the Service domain, and the costs in the Finance domain. The Organization domain has direct impact on the other domains. The Finance domain determines the pricing in the Service domain [58].

STOF Method

The STOF model uses the STOF method in four steps: the Quick Scan; the Critical Success Factors (CSFs); the Critical Design Issues (CDIs) and the Internal and External Issues.

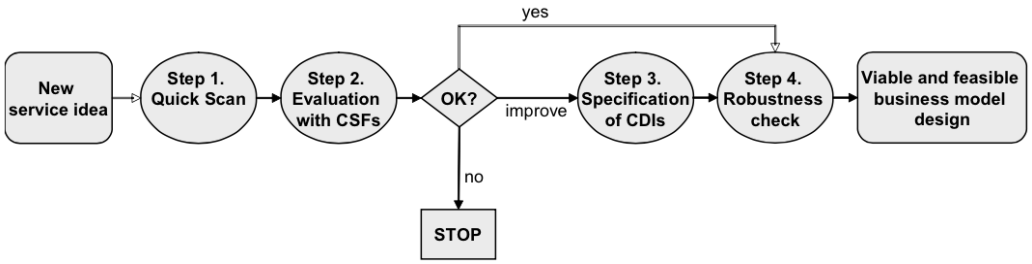


Figure 4.2: Design steps in the STOF method [42]

In the diagram in Figure 4.2 shows that the starting point for a new service idea is the quick scan focus on the four domains: Service, Technology, Organization and Finance.

Practical Case

Our goal is to create a business model outline for the Computer Science Library case study. In our case we have defined the first step, which includes aspects from the four domains, consisting in Service domain focus on value proposition and target; Technology domain is a driver for new innovative services in the delivery and the system; Organization domain for actor roles and network strategy, that are the resources; and Finance domain for pricing and investments (CAPEX & OPEX), that is the revenue model. The list of factors for each domain is summarized on Table 4.2 for "Book Room & Unlock Door" scenario.

STOF domain	Scenario at CS Library
Service	• Value proposition • Target group
Technology	• Mobile device • Tablet • Electronic lock • Cloud-base • Wireless LAN
Organization	• Project funded by EIT ICT Labs, coordinated by VTT and service delivered by Aalto University
Finance	• Cost reduction • Pricing scheme • Profit

Table 4.2: STOF domains of pervasive services for flexible spaces

The first step in the STOF method starts with a checklist of the relevant factors in the Service domain, succeeded by the other three domains in the following order: Technology, Organization and Finance.

A. Service Domain The service domain focuses on the service value by comparing the new to existing similar (traditional) services.

The core value proposition with this new service is to prevent unauthorized people from unlocking the door, thus addressing security and privacy concerns.

Another core value is includes usability because users carry their smartphone all the time and use it almost everywhere, so they are more likely to take it with them and less likely to leave it at home than a physical key or card. In this new service we take into account the user's availability when users no longer have to be at the library to use the service. At the same time, users benefit from saving time when they no longer need to wait for the CS building opening hours to book a room because the new service is independent of the time or the day.

In addition, it includes user efficient decision making, such as: the user can be offered a possibility to review the status of the reservation, to delete and create another appointment; the user can book in advance – 15 days, 1 month, 1 year earlier – unlike the old system where the user could only book during the current week. Moreover, the design system is based on allowing the user to reserve in advance and the core action (open door) is activated when needed.

On the other hand, the value proposition on the management side offers managers efficient decision making for dynamic space utilization and recording usage, such as: the room is booked but it is not occupied then the reservation is automatically (in fifteen minutes) available again.

Another feature in the Service domain is the target group. This is interrelated with a clearly defined target group (customer segments) dealing with consumer or business market and between niche or mass market. In this case, people using smartphones and using flexible spaces is a market niche. In addition, despite the fact that at first the service could seem not so interesting – as it happens with all new technological devices – based on Mark Weiser's vision [84] we can say that our service has become familiar and transparent, it will attract and select people from all segments with the end user value proposition associated.

B. Technology Domain Pervasive services are delivered via technology and should have an acceptable quality level. Bouwman et al. [42] in their

business models formulation suggested a technical architecture with the following components: middleware (mobile device); web services as the core functionality; and the network and infrastructure characteristics at the base.

The technology architecture for our new service is detailed in Chapter 3 and it is aligned with the author's suggestion. Firstly, the middleware component comprises the user's mobile device, digital signage (QR Code) in the tablet, and electronic lock sensor. Secondly, the web service layer includes cloud based divided into the web server, database, website and third party authentication. Lastly the base layer consists of the network and the infrastructure of the wireless LAN. The technology components are listed in Table 4.2.

In addition, critical functionalities that are needed to develop any service that runs on the network and is offered to a client [42] are authentication and security. Our service provides a secure identification of the user through the third party authentication in a way that the user is identifiable to the service and at the same time excludes the possible risks of identity theft.

C. Organization Domain The organization domain defines the value network that consists in business partners who have resources and capabilities, which interplay in order to create service value.

This service requires collaboration among the following players: one Venture Capitalist; one Projector Coordinator; one Space Owner; one Service Leader; one Service Developer; one Pilot Holder and End Users.

As the *Venture Capitalist*, EIT ICT LABS provides the support for the project.

The *Project Coordinator*, Ingrid Schembri, supervises the Flexible Spaces Project, monitoring its services development, namely the "Book Room & Unlock Door".

Aalto University is the *Space Owner* and the infrastructure and premises proprietor. It is also responsible for the internal rental system management, named "Asio".

The *Service Leader* is Aalto University: Distributed and Pervasive System Research Group, Mario Di Francesco. He is responsible for the service's design and implementation. In addition, he is the element in charge of addressing VTT and acquiring a Pilot Holder.

The *Service Developer*, Catarina Moura, is the programmer and responsible for choosing the appropriate tools and framework for the service. Additionally, she guarantees the proper functioning of it.

CS Library is the *Pilot Holder*, who manages the pilot space and regulates its price or penalties.

Finally, the *End Users* adopt the service and accept the rules, as well as the pricing plan.

D. Finance Domain The finance domain defines the financial arrangements in the value chain which must result in acceptable outcomes from the participating partners. In other words, the finance domain goal is to create a viable business model by generating financial benefits for the actors involved.

Finance variables are detailed in Section 4.3, namely investment, costs, revenues, but also risk (sensitivity analysis) and performance indicators (market adoption, return on investment) have an impact on the business model, however, these last metrics are not analysed in this thesis. The bottom line of the business model is addressed in the finance domain, which is shown on Table 4.2.

With regard to this finance domain, the important figures for this domain analysis are the cost-reduction achieved by synergy in network, the pricing scheme considering the results from the balance between service direct costs and revenues and also the dynamic pricing strategy in which the price adapts to the observed congestion on the reservation system. Finally, the financial statement resulting in variable profit is not an issue and a break even is the goal.

Critical Success Factors (CSFs)

After the first phase which consist of the fours domain's characteristics, this second step regards the project evaluation with Critical Success Factors (CSFs).

CSFs specify "the limited number of areas in which satisfactory results will ensure that the business model creates value for the customer and for the business network" [42]. In other words, the underlying logic is that a negative estimation will imply an impediment in the business model's viability, and therefore a system will suffer a re-evaluation and redesign. The selection of CSFs in "Book Room & Unlock Door" case is summarized on Table 4.3, which are related but in a different way with the factors already focus on the SWOT analysis.

Critical Success Factors	Scenario at CS Library
1.Clearly Defined Target Group	*Positive.* The service targets all the people with smartphone.
2.Quality of the Value Proposition	*Positive.* The service supports the features: • Key and card less. • Deny unauthorized person. • Online centralized system. • Access control suitable for multi-tenancy. • Review status reservation and advance booking. • Save the log user to identify who open the door.
3.Quality of the Delivery System	*Negative.* The system can has a breakdown, for example in: • Server. The service does not provide server redundancy therefore the user will not be able to book or unlock door. • Battery electronic lock. The lock does not react to the input commands therefore the door remains locked. • Tablet. The service does not provide equipment substitution at the time, then the QR Code is not transmitted. *Positive.* Help-desk offer (the building janitor open door manually).
4.Acceptable Division of Roles	*Positive.* The organization is clear and concise.
5.Acceptable Profitability	*Positive.* Profit is not an issue, although breaking even is desirable.
6.Unobtrusive Customer Retention	*Not Applicable.* The service not yet in the market.
7.Acceptable Risks	*Not Applicable.* Sensitivity analysis in financial aspects is out of the thesis scope.

Table 4.3: CSFs of pervasive services for flexible spaces

Another critical issue in the technology domain is related to supplier

dependency. Accordingly to Bouwman et al. [42] web services can be provided by third parties and should not be dependent on a specific company for IT resources and infrastructure. In the foundation of our service there is dependence between the service and the Aalto IT resources characterized as a threat in the SWOT model.

Finally, the STOF critical success factors assessment is positive for "Book Room & Unlock Door" business case. It demonstrates through defined target group, the quality of the value proposition, division of roles and acceptable profitability. However, a few critical issues from the technology domain namely, quality of deliver system and supplier dependency are unsatisfactory, hence, it should be reconsidered and redesigned on CDIs.

The subsequent phases, CDIs and a robustness check is for future work because it will be more appropriate to be done in the first roll-out of the service.

Other New Service Benefits

There are also other potential benefits from the new service, one is the dynamic space utilization, for example the meeting room at CS Library, allows the smartphone to be paired with a display and set up automatically the presentation on the big screen [69].

A second benefit would be combining the smartphone and the light sensor for controlling on/off and the intensity of the room lighting. Additionally, the service could store the information preference of the user and the next time adjusts correctly the parameters.

A third benefit would be the service sensing the user while he is approaching the door and in parallel verify all the user conditions and opening the door without any user interaction.

A fourth benefit would be combining the online booking and a third-party application, e.g., Google Calendar or iCal, in a way that the user is notified for the scheduled meetings. However, to incorporate the mentioned benefits in the service more finance and R&D would be needed.

4.3 Financial

A viable business model must ensure that costs, investments and risk sources are divided in such a way is acceptable by all parties involved. The service also needs to generate enough revenues. Finally, the project partners can share benefits (revenue share).

To date, business models to realize open source technological services, comprising pervasive services, suggest the free exploitation of the economic fact that the marginal cost of software production and delivery approaches zero. Besides this, there are subsections about pricing schemes, costs and partnership and the financial statement about the viability of the project.

4.3.1 Pricing Schemes

Pricing is a highly important critical issue for the service. Subsequently, the price scheme for pervasive services strongly influences whether or not end users are willing to start adopting the service that as customers are not used to pay.

Nowadays, interest has been renewed in pricing research; researchers need to account for technological challenges as well as socioeconomic factors that determine the eventual adoption and implementation of pricing schemes. Challenges in pricing research concerning primarily the Internet, and more recently the digital services, were reported by Sen et al. [76]. The authors in Figure 4.3 highlight the role of static and dynamic pricing summarizing that:

- Static pricing is when prices do not vary with the network congestion level. These pricing plans are: Fixed Flat-rate; Usage-based; Priority Pricing; Cumulus Pricing; Application-based or App-Based Pricing; and Time-of-Day.

- Dynamic pricing is when prices adjust in response to the network congestion level. The selected price categories are: Hourly price charges; Location and cell-load based; and Time-and usage based.

Pricing Practice		Example Pricing Plan (see the article for details)		
Type	Category	Description	Network	Country
Static	Fixed Flat Rate	Monthly fee; unlimited	Both	Vanishing worldwide
		Monthly; flat to a cap, then usage based	Wired	U.S. (AT&T, Verizon)
		Monthly; flat to a cap, then throttle	Wired/wireless	Spain (Orange) U.S. (Comcast)
		Monthly; shared	Wired/wireless	Canada (Rogers) U.S. (AT&T, Verizon)
	Usage Based	Hourly rate	Wireless	Egypt (Mobinil)
		Cap, then metered	Wired/wireless	Worldwide (e.g., U.S., U.K.)
	Priority Pricing	Priority pass (for dongle users)	Wireless	Singapore (SingTel)
	Time of Day	Daytime & nighttime rates	Wireless	India (BSNL)
		Users choose happy hours	Wireless	U.K. (Orange)
	Cumulus Pricing	Usage-based contract negotiation	Wireless	U.K. (Vodafone)
	App-Based Pricing	Free access to select apps; bundling	Wireless	U.K. (Orange) Denmark (TDC)
Dynamic	Congestion Based	Hourly price changes	Wireless (voice calls)	Uganda (MTN)
		Location and cell-load based	Wireless (voice calls)	India (Uninor)
		Time- and usage based	Wireless (data)	U.S. (pilot trial)

Figure 4.3: Example price plans [76]

These new pricing schemes require support from software platforms on user devices and on service providers. Furthermore, the determination of the optimal price for access requires to develop economic models that account for the cost structure [76].

With these in mind, we try to assess pricing for our service with the underlying basic concepts of:

- Pricing aligned between the price the service providers are ready to offer and the price that consumers are willing to accept.
- Pricing to cover setup costs.
- Dynamic pricing adapts the price in response to the market demand.

In a pricing scheme the first concept is the price aligned with the service value proposition. On the one hand, the strategy in traditional service is

zero price charged. On the other hand, the implemented service will charge a price to switch to an automated system. These users needs will appeal and persuade them to pay the service. An attractive pricing plan is structured according to: i) Flat rate; ii) Per event price; iii) Subscription, among the most known.

What would users be prepared to pay for "Book Room & Unlock Door" at CS Library? The price in this user case with the new service is zero; however, a penalty charge will be added hypothetically to match other bookings prices in campus buildings (e.g., penalties in the library; internal rents; sauna rooms).

Another concept about pricing is whether the price covers setup costs. The pricing plan must confront the user value with the costs component. On the one hand, the infrastructure costs are negligible because of the synergistic effect. On the other hand, there are costs related to capital expenditure (e.g., service development, lock equipment and tablet).

In the pricing scheme a dynamic pricing strategy can be followed due to fluctuating in the reservation system, so the price will adjust. On the one hand, the price variation allows alleviating the reservation system for better management. On the other hand, usually users dislike it because they do not have a fixed price, so problems for service adoption can occur. Therefore, the pricing plan for "Book Room & Unlock Door" could be constructed with the following assumptions:

- Consumers should pay the price or penalties.
- Penalties should be charged with a dynamic price strategy in order to incentive users to adjust their behavior through situations like:
  - Avoiding empty rooms. This situation happens when there is a waiting list for the rooms which have very low occupancy rate.
  - Avoiding inadequate room booking. This situation occurs when the space size (m^2) does not fit the number of people in the space. Although this penalty is not applicable if all suitable rooms are occupied.
  - Avoiding congestion. This situation is observed when there is a constant congestion at a certain hour or day.
- Price target to break even, make a profit is not a target for the CS Library.
- Space rent element is the core of the service business. Aalto has an internal rent-system.

- Extra income can pay for example advertisements to direct consumers to the service.

The reservation system for our service was designed to allow these and other features because we considered user context as described in Chapter 3.

4.3.2 Costs and Partnerships

As financial resources of the project also have side costs, nowadays the importance is to achieve the best cost structure, so each firm faces the question to perform allocated tasks on their own or to outsource them. Moreover, creating new business, the organizational issues counts and partnerships are a good way to value creation from innovation.

Costs

Pervasive services cloud-based are less costly because the critical costs related to Capital Expenditure (CAPEX) including the infrastructure and equipment investment is not an issue, basing on the fact that all infrastructure is already installed and available. Operational Expenditure (OPEX) with regards to network is reduced and only some overhead non-network costs are relevant, such as corporate real estate (fixed rent and variable maintenance) and other administrative costs (see Table 4.4).

Partnerships

Ultimately, for partnerships between Universities and Research Centers, the division is characterized by a fee fixed related to project development – funded by EIT ICT Labs – and royalty sharing. In this case the partnership is a joint project between Aalto and VTT sponsored by EIT ICT Labs whence the pilot will be launched at Aalto CS Library. EIT ICT Labs and VTT taking part, will probably install the service in their premises, which will help to promote the service.

4.3.3 Financial Statement

The financial statement presents first the investment cost that is capital expenditure (CAPEX), then the profit and loss account (P&L) including the operational costs (OPEX) and the income (Revenues) of the service. Finally, a line of the P&L table shows the EBITDA (**E**arnings **B**efore **I**nterest, **T**axes, **D**epreciation, and **A**mortization) calculation and the following line the depreciation of the investment. The bottom line of the P&L statement

is the service profit (result). This section presents a hypothetical financial plan to "Book Room & Unlock Door" project.

However, out of this thesis scope, there are additional metrics to be applied in financial modeling. The most well known are the following:

- NPV (Net Present Value) provides how much the project is worth, the total amount of money for the project, giving the current amount (revenues minus costs) by the sum of the future cash flows at a discount rate, called "time value of money".
- ROI (Return on Investment) is a percentage from the ratio Profit/Investment, which helps to evaluate performance of an investment and make the decision if it is worth.
- Payback is the number of years to take profit from an investment.
- Sensitivity analysis recognizes uncertainty in the project, by the variation of the assumptions to settle the basic, pessimist and optimistic scenarios for the service.

To construct a financial statement template for "Book Room & Unlock Door", see Table 4.4, we need to consider the following assumptions:

- Fixed hourly rate for internal rents, according to the Aalto Price List.
- Penalty hourly cost, according to the rules of use.
- Availability of 1 electronic lock, which will be installed in CS Library "Tomatti room" in the first year (2014):
  - Each *Gearlock Mohinet* will have an average cost of **€380** each, including Backend support for 2 years.
- The development of the service including programming and organizational tasks will cost around **€5,000**.
- Availability of 1 tablet per door:
  - Each costs **€370**.
- Rooms expected to be used:
  - 1 room (Tomatti room). The size is 8,5 m^2. Rent costs about €30-40/m^2 a month.

  - **260 days** per year (excluded weekends, holidays, vacancy).
  - open from 8 a.m. to 8 p.m.

- Overhead costs based on other business cases, to be expected around **25%**.

- Depreciation of investment is **2 years** for equipment (minimum guarantee period under European Union rules).

For the case's basic scenario the project for 1 library room in a year period the break even will happen if the following is taken in consideration:

- Scenario 1) Fixed flat **€2.00/hour with usage 8h/day**; or

- Scenario 2) Fixed flat **€1.50/hour with usage 10h/day**.

The break even calculation:

- Identifies the number of hours daily per room to be used to cover business expenses (before tax or interest costs). It is helpful when updating the Business Plan.

- Hypothetically, a penalty can be charged on top of internal rent prices, based on the pricing dynamic scheme described in Section 4.3.1.

Scenario 1) Fixed flat €2.00/hour with usage 8h/day

Investment (CAPEX)	
Deployment Lock	€380
Tablet	€370
Software Development	€5,000
Total	€5,750
Operational Costs (OPEX)	
Overhead costs	€1,040
Network infrastructure	€0
Space Rental	€0
Total	€1,040
Revenues	
1 room based on daily usage	€4,160
Total	€4,160
EBITDA	€3,120
Depreciation	€2,875
Result (profit)	**€245**

Table 4.4: P&L "Book Room & Unlock Door". Scenario 1) 1 Room/1 Lock (2014)

Calculation aid:

Overhead costs = 25% × Revenues =€4,160×0.25 =€1,040
Revenues = 8 hours×€2/hour × 260 days = €4,160
EBITDA = Revenues − OPEX = €4,160 − €1,040 = €3,120
Depreciation = $\frac{CAPEX}{numberofyears} = \frac{5,750}{2}$ =€2,875/year
Profit = EBITDA − Depreciation/year = €3,120 − €2,875 = €245

4.4 Summary

In this chapter, we have illustrated the benefit of business models to move from research user prototype towards a service proposition to deliver to customers a pervasive service in a flexible space.

Our analysis has not resulted in an exhaustive business model, although we feel that it is an important step towards "go to market". This business model associated to the service "Book Room & Unlock Door" proposed im-

portant questions such as how to create customer value and who to involve in the value chain.

This study has shown that one possible break-through is the adoption of the pervasive service for flexible spaces. This service brings greater flexibility and efficiency in work environments; it also inspires the service opportunity to grow. Furthermore, significant drivers affect positively the deployment of the service particularly being an over-the-top service, thus resulting in acceptable prices on capital and operational expenditure, a clear value proposition and potential stakeholders such as the government or commercial parties.

This theoretical analysis provides the first step for commercial exploitation for similar services focused on booking spaces and an automated door opening system. If the service is successfully adopted, it can become a mature pervasive service for smart environments in the near future.

Chapter 5

Conclusions

In this thesis we have developed a pervasive service for flexible spaces by building a cloud-based reservation and access system for shared environments. The new architecture proposed copes the existing limitations in shared rooms through, automating the reservation and the access to the room. The implemented service was developed with open and modern technologies creating a positive environment to extend the work. Furthermore, the pervasive service was successfully deployed as a pilot in Aalto Computer Science Library. This achievement is a step forward to the research in this area. Finally, to substantiate our work, we study the viability of the project by presenting a business model. It was possible to highlight the powerful characteristics of the framework and the issues to address it. This service reached a step towards to commercial exploitation through the availability and the convenience of the service; and the possibility to measure the space utilization which allows to maximize its usage.

While this service was being developed, we faced some limitations concerning: the technology implemented, for instance the user needs to have a smartphone and a QR Code reader application in order to utilize our service; it was not possible to integrate NFC technology in the service as the embedded hardware used was not reliable enough; due to physical constraints in the pilot environment, we could not use the NFC capabilities of the tablet either the integration of the Aalto login option was not realized due to timing and bureaucratic constraints.

This proposed service has potential to be extended by integrating more advanced features, such as the following :

- Implementing sensors for sensing door status and detecting human presence. With these sensors it was possible to accurately prevent room squatting and increase utilization efficiency.

- Increasing the context-aware sensing data which consequently reduces the user's tasks. For instance, user proximity to the room eliminates the necessity the user performs the second phase 'At the Library' by instantaneously acknowledging is the right user to access the room.
- Spread the service to adapt to multiple smart environments.
- Switch the service to connections over HTTPS.
- Create an option for the user who booked the room, to allow other people to open the door as well (i.e. delegation).
- Explore electronic lock alternatives.

Bibliography

[1] Aalto Department of Computer Science and Engineering. http://cse.aalto.fi/en/. Accessed: 2014-11-08.

[2] hapi-auth-cookie. https://github.com/hapijs/hapi-auth-cookie. Accessed: 2014-11-08.

[3] bell. https://github.com/hapijs/bell. Accessed: 2014-11-08.

[4] BSON. http://bsonspec.org/. Accessed: 2014-11-08.

[5] CSS. http://www.w3.org/Style/CSS/Overview.en.html. Accessed: 2014-11-08.

[6] Datepicker. http://jqueryui.com/datepicker/. Accessed: 2014-11-08.

[7] Dialog. http://jqueryui.com/dialog/. Accessed: 2014-11-08.

[8] FlexSlider. http://www.woothemes.com/flexslider/. Accessed: 2014-11-08.

[9] Samsung Galaxy S5. http://www.samsung.com/global/microsite/galaxys5/. Accessed: 2014-11-08.

[10] Hapi.js. http://hapijs.com/, . Accessed: 2014-11-08.

[11] Hapi-MongoDB. https://github.com/Marsup/hapi-mongodb, . Accessed: 2014-11-08.

[12] HTML. https://developer.mozilla.org/en-US/docs/Web/HTML. Accessed: 2014-11-08.

[13] iron. https://github.com/hueniverse/iron. Accessed: 2014-11-08.

[14] jQuery. http://jquery.com/. Accessed: 2014-11-08.

[15] JavaScript. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed: 2014-11-08.

[16] JSON. http://www.json.org/. Accessed: 2014-11-08.

[17] Google Maps JavaScript API v3. https://developers.google.com/maps/documentation/javascript/tutorial. Accessed: 2014-11-08.

[18] Mohinet. http://www.mohinet.fi/. Accessed: 2014-11-08.

[19] MongoDB. http://www.mongodb.org/, . Accessed: 2014-11-08.

[20] MONGODB - THE LEADING NOSQL DATABASE. http://www.mongodb.com/leading-nosql-database, . Accessed: 2014-11-08.

[21] LG Google Nexus 5. http://www.google.com/nexus/5/, . Accessed: 2014-11-08.

[22] Asus Google Nexus 7. http://www.asus.com/Tablets_Mobile/Nexus_7/, . Accessed: 2014-11-08.

[23] NFC Forum. http://nfc-forum.org/. Accessed: 2014-11-08.

[24] Node.js. http://nodejs.org/. Accessed: 2014-11-08.

[25] npm. https://www.npmjs.org/. Accessed: 2014-11-08.

[26] OAuth. http://oauth.net/. Accessed: 2014-11-08.

[27] NFC. http://www.nearfieldcommunication.org/. Accessed: 2014-11-08.

[28] Flexible Spaces Service. https://www.eitictlabs.eu/about-us/calls-tenders/call-for-smes-in-facility-services-enabling-technologies/. Accessed: 2014-11-08.

[29] QR Code. http://www.qrcode.com/en/. Accessed: 2014-11-08.

[30] Socket.IO. http://socket.io/. Accessed: 2014-11-08.

[31] speakeasy. https://github.com/markbao/speakeasy. Accessed: 2014-11-08.

[32] Tooltip. http://jqueryui.com/tooltip//. Accessed: 2014-11-08.

[33] VTT Review 2013. `http://www.vtt.fi/files/vtt/vtt_review_2013.pdf`. Accessed: 2014-11-08.

[34] Ryan Dahl: Original Node.js presentation. `https://www.youtube.com/watch?v=ztspvPYybIY`, June 2012. The first presentation on Node.js from Ryan Dahl at JSConf 2009. Accessed: 2014-11-08.

[35] Aalto University Annual Report 2013. `http://www.aalto.fi/en/about/reports_and_statistics/aalto_university_annual_report_2013.pdf`, March 2014. Accessed: 2014-11-08.

[36] Office Space Concept for Aalto Arts, April 2014. Final Report on the Working Environment.

[37] Notice of Pilot: Flexible Spaces Services. `http://lib.aalto.fi/en/current/news/fss-pilot.pdf`, November 2014. Accessed: 2014-12-08.

[38] Raphael Amit and Christoph Zott. Value creation in E-business. *Strategic Management Journal*, 22(6-7):493–520, 2001. ISSN 1097-0266. doi: 10.1002/smj.187. URL `http://dx.doi.org/10.1002/smj.187`.

[39] F. Belqasmi, R. Glitho, and Chunyan Fu. RESTful web services for service provisioning in next-generation networks: a survey. *Communications Magazine, IEEE*, 49(12):66–73, December 2011. ISSN 0163-6804. doi: 10.1109/MCOM.2011.6094008.

[40] Robin Berjon, Steve Faulkner, Travis Leithead, Silvia Pfeiffer, Edward O'Connor, and Erika Doyle Navara. HTML5. Candidate recommendation, W3C, July 2014. http://www.w3.org/TR/2014/CR-html5-20140731/.

[41] A. Bouain, A. El Fazziki, and M. Sadgal. Pervasive services vs. Web services: Survey and comparison. In *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, pages 552–557, April 2014. doi: 10.1109/ICMCS.2014.6911148.

[42] Harry Bouwman, Henny De Vos, Timber Haaker, et al. *Mobile service innovation and business models*, volume 2010. Springer, 2008.

[43] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, RFC Editor, March 2014. URL `http://www.ietf.org/rfc/rfc7159.txt`.

[44] Ethan Cerami. *Web services essentials: distributed applications with XML-RPC, SOAP, UDDI & WSDL.* " O'Reilly Media, Inc.", 2002.

[45] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty. Intelligent agents meet the semantic Web in smart spaces. *Internet Computing, IEEE*, 8(6):69–79, Nov 2004. ISSN 1089-7801. doi: 10.1109/MIC.2004.66.

[46] Diane J. Cook and Sajal K. Das. How smart are our environments? An updated look at the state of the art . *Pervasive and Mobile Computing*, 3(2):53 – 73, 2007. ISSN 1574-1192. doi: http://dx.doi.org/10.1016/j.pmcj.2006.12.001. URL `http://www.sciencedirect.com/science/article/pii/S1574119206000642`. Design and Use of Smart Environments.

[47] N. Davies, M. Langheinrich, R. Jose, and A. Schmidt. Open Display Networks: A Communications Medium for the 21st Century. *Computer*, 45(5):58–64, May 2012. ISSN 0018-9162. doi: 10.1109/MC.2012.114.

[48] ECMA International. *Standard ECMA-262 - ECMAScript Language Specification*. 5.1 edition, June 2011. URL `http://www.ecma-international.org/publications/standards/Ecma-262.htm`.

[49] EmpireJS. 10 Empire Node — Ben Acker — Hapi js. `https://www.youtube.com/watch?v=or9uV37HTwk`, October 2014. Accessed: 2014-11-08.

[50] Elisabetta Farella, Mirko Falavigna, and Bruno Riccó. Aware and smart environments: The Casattenta project. *Microelectronics Journal*, 41(11):697 – 702, 2010. ISSN 0026-2692. doi: http://dx.doi.org/10.1016/j.mejo.2010.01.008. URL `http://www.sciencedirect.com/science/article/pii/S0026269210000170`. {IEEE} International Workshop on Advances in Sensors and Interfaces 2009.

[51] I. Fette and A. Melnikov. The WebSocket Protocol. RFC 6455, RFC Editor, December 2011. URL `https://tools.ietf.org/rfc/rfc6455.txt`.

[52] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, June 1999. URL `http://tools.ietf.org/rfc/rfc2616.txt`.

[53] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[54] N. Haller, C. Metz, P. Nesser, and M. Straw. A One-Time Password System. RFC 2289, RFC Editor, February 1998. URL `http://tools.ietf.org/rfc/rfc2289.txt`.

[55] D. Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, RFC Editor, October 2012. URL `https://tools.ietf.org/rfc/rfc6749.txt`.

[56] Matthias Heinrich and Martin Gaedke. Data Binding for Standard-based Web Applications. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 652–657, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0857-1. doi: 10.1145/2245276.2245402. URL `http://doi.acm.org/10.1145/2245276.2245402`.

[57] S. Helal. Programming pervasive spaces. *Pervasive Computing, IEEE*, 4(1):84–87, Jan 2005. ISSN 1536-1268. doi: 10.1109/MPRV.2005.22.

[58] A. Juntunen, S. Luukkainen, and V.K. Tuunainen. Deploying NFC Technology for Mobile Ticketing Services 150; Identification of Critical Business Model Issues. In *Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on*, pages 82–90, June 2010. doi: 10.1109/ICMB-GMR.2010.69.

[59] Yung-Wei Kao, Guo-Heng Luo, Hsien-Tang Lin, Yu-Kai Huang, and Shyan-Ming Yuan. Physical access control based on qr code. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, pages 285–288, Oct 2011. doi: 10.1109/CyberC.2011.55.

[60] Jouni Koljonen. Mohinet. In *Smart Ageing Network Finland* , page 17. Tekes, 2013. URL `https://www.tekes.fi/globalassets/global/ohjelmat-ja-palvelut/ohjelmat/turvallisuus/safety-and-security-ohjelmaraportti/safety-and-security-explore-the-topics/healthcare-and-home-artikkelit/sanf_esite_digi_final_lr.pdf`.

[61] Mohan Kumar, Behrooz A. Shirazi, Sajal K. Das, Byung Y. Sung, David Levine, and Mukesh Singhal. Pico: A middleware framework for pervasive computing. *IEEE Pervasive Computing*, 2(3):72–79, 2003. ISSN 1536-1268. doi: http://doi.ieeecomputersociety.org/10.1109/MPRV.2003.1228529.

[62] Jari Laarni, Timo Miiluniemi, Esa Nykänen, Ingrid Schembri, and Eric Richert. Case study summary report: New ways of working. Technical report, VTT, 2014.

[63] Yishan Li and S. Manoharan. A performance comparison of sql and nosql databases. In *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*, pages 15–19, Aug 2013. doi: 10.1109/PACRIM.2013.6625441.

[64] Peter Lubbers and Frank Greco. Html5 web sockets: A quantum leap in scalability for the web. *SOA World Magazine*, 2010.

[65] Hannamaija Määttä. Supporting distributed knowledge work in a knowledge intensive organisation. 2011.

[66] A. B. M. Moniruzzaman and Syed Akhter Hossain. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *CoRR*, abs/1307.0191, 2013. URL `http://arxiv.org/abs/1307.0191`.

[67] Michael Morris, Minet Schindehutte, and Jeffrey Allen. The entrepreneur's business model: toward a unified perspective. *Journal of Business Research*, 58(6):726 – 735, 2005. ISSN 0148-2963. doi: http://dx.doi.org/10.1016/j.jbusres.2003.11.001. URL `http://www.sciencedirect.com/science/article/pii/S014829630300242X`. Special Section: The Nonprofit Marketing Landscape.

[68] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. TOTP: Time-Based One-Time Password Algorithm. RFC 6238, RFC Editor, May 2011. URL `https://tools.ietf.org/rfc/rfc6238.txt`.

[69] Elena Oat, Mario Di Francesco, and Tuomas Aura. MoCHA: Augmenting pervasive displays through mobile devices and web-based technologies. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 506–511. IEEE, 2014.

[70] Zachary Parker, Scott Poe, and Susan V. Vrbsky. Comparing nosql mongodb to an sql db. In *Proceedings of the 51st ACM Southeast Conference*, ACMSE '13, pages 5:1–5:6, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1901-0. doi: 10.1145/2498328.2500047. URL `http://doi.acm.org/10.1145/2498328.2500047`.

[71] PayPalUIE. Eran Hammer of WalMart Labs presents hapi. `https://www.youtube.com/watch?v=Recv7vR8ZlA`, March 2013. Accessed: 2014-11-08.

[72] V. Pimentel and B.G. Nickerson. Communicating and Displaying Real-Time Data with WebSocket. *Internet Computing, IEEE*, 16(4):45–53, July 2012. ISSN 1089-7801. doi: 10.1109/MIC.2012.64.

[73] Adina Ploscar. Xml-rpc vs. soap vs. rest web services in java–uniform using wswrapper. *Int. J. Comput*, 4:215–223, 2012.

[74] Raffaele Quitadamo, Franco Zambonelli, and Giacomo Cabri. The service ecosystem: Dynamic self-aggregation of pervasive communication services. In *Proceedings of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, SEPCASE '07, pages 1–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2970-4. doi: 10.1109/SEPCASE.2007.11. URL `http://dx.doi.org/10.1109/SEPCASE.2007.11`.

[75] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2): 300–304, 1960. doi: 10.1137/0108018. URL `http://dx.doi.org/10.1137/0108018`.

[76] Soumya Sen, Carlee Joe-Wong, Sangtae Ha, and Mung Chiang. A Survey of Smart Data Pricing: Past Proposals, Current Plans, and Future Trends. *ACM Comput. Surv.*, 46(2):15:1–15:37, November 2013. ISSN 0360-0300. doi: 10.1145/2543581.2543582. URL `http://doi.acm.org/10.1145/2543581.2543582`.

[77] Mohit Sethi, Elena Oat, Mario Di Francesco, and Tuomas Aura. Secure bootstrapping of cloud-managed ubiquitous displays. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 739–750. ACM, 2014.

[78] Altamira TC. Node.js explained. `https://www.youtube.com/watch?v=ztspvPYybIY`, July 2012. Jeff Kunkle presents on Node.js. Accessed: 2014-11-08.

[79] Pedro Teixeira. *Professional Node.js: Building Javascript based scalable software*. John Wiley & Sons, 2012.

[80] M. Tim Jones. Understand Representational State Transfer (REST) in Ruby. `http://www.ibm.com/developerworks/library/os-understand-rest-ruby/os-understand-rest-ruby-pdf.pdf`, August 2012. Accessed: 2014-11-08.

[81] J.S. van der Veen, B. van der Waaij, and R.J. Meijer. Sensor data storage performance: Sql or nosql, physical or virtual. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 431–438, June 2012. doi: 10.1109/CLOUD.2012.18.

[82] Xiaohang Wang, Jin Song Dong, ChungYau Chin, Sanka Ravipriya Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *Computing*, 1(2):67–74, 2002.

[83] Roy Want and Bill N. Schilit. Interactive Digital Signage. *Computer*, 45 (5):21–24, May 2012. ISSN 0018-9162. doi: 10.1109/MC.2012.169. URL `http://dx.doi.org/10.1109/MC.2012.169`.

[84] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

[85] Mark Weiser and JohnSeely Brown. The Coming Age of Calm Technology. In *Beyond Calculation*, pages 75–85. Springer New York, 1997. ISBN 978-0-387-98588-6. doi: 10.1007/978-1-4612-0685-9_6. URL `http://dx.doi.org/10.1007/978-1-4612-0685-9_6`.

[86] Jiehan Zhou, J. Riekki, and Junzhao Sun. Pervasive service computing toward accommodating service coordination and collaboration. In *Frontier of Computer Science and Technology, 2009. FCST '09. Fourth International Conference on*, pages 686–691, Dec 2009. doi: 10.1109/FCST.2009.39.

[87] Christoph Zott, Raphael Amit, and Lorenzo Massa. The business model: recent developments and future research. *Journal of management*, 37 (4):1019–1042, 2011.

Appendix A

NFC - based prototypes

Besides the actual hardware-software solution presented in Chapter 3 and deployed in the pilot, we have also investigated using Near Field Communication (NFC) as tag technology. To this end, we have performed the two prototypes detailed next.

A.1 Raspberry PI equipped with a card reader

This prototype consist of a ACR122U USB NFC reader from Advanced Card Sytems (ACS) and the Raspberry PI (Figure A.1). The ACR was acting as a tag, behaving in card emulation mode. The Raspberry PI interacted with the reader through libnfc. The issue with this solution was that the ACR was unreliable. The problem persisted even when the ACR reader was connected to an Unix Machine instead of the Raspberry PI.

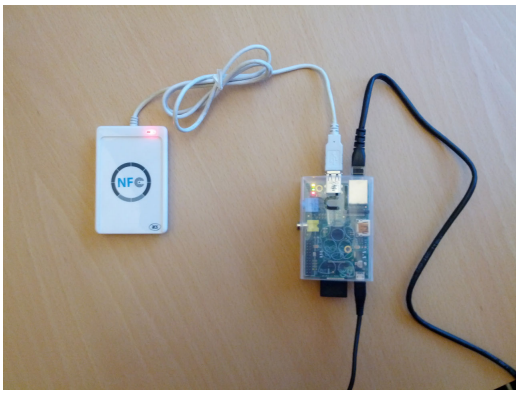


Figure A.1: The NFC reader and the Raspberry PI

A.2 Android phones equipped with NFC

After the first attempt, the next try was to use Android phones equipped with NFC acting as an NFC card. To this end the mode used to continuously rewrite the tag was the card emulation. The drawback was the phone that simulates the user required to setup some parameters to communicate with the other phone, which it was not practical.

Lastly, we moved to the P2P mode in both Androids. The solution worked smoothly, as the phone was continuously transmitting the new TOTP and the user phone was able to receive it. The only requirement needed from the user phone was to enable the NFC option.

However, a small detail prevented us to use this NFC-based solution in the pilot. For the P2P communication be established both phones require to be touching back to back and unfortunately it was not feasible in our test deployment, because the device would be attached to the glass for advertise the QR Code and give instructions to the user.

The main propose of integrating the NFC in this service was to offer a simple and fast alternative to the QR Code advertisement. Although, the NFC integration on the phones are gradually increasing, the majority of them still does not have it. For example the NFC in the iPhone has just appeared recently.