FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Concept and Design of an Interoperable Mobile Ticketing-System for Software Engineering Education

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Andreas Lindner
Matrikelnummer 0525142

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Thomas Grechenig
Mitwirkung: Thomas Artner

Wien, 26. August 2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Concept and Design of an Interoperable Mobile Ticketing-System for Software Engineering Education

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Andreas Lindner
Registration Number 0525142

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Thomas Grechenig
Assistance: Thomas Artner

Vienna, 26. August 2014

(Signature of Author) (Signature of Advisor)

Concept and Design of an Interoperable Mobile Ticketing-System for Software Engineering Education

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Andreas Lindner
Registration Number 0525142

elaborate at the
Institut of Computer Aided Automation
Reseach Group for Industrial Software
to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Thomas Grechenig
Assistance: Thomas Artner

Vienna, 26. August 2014

Erklärung zur Verfassung der Arbeit

Andreas Lindner
Lehnergasse 3/5, 1230 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

Mein Dank gilt den folgenden Personen, durch welche diese Arbeit erst ermöglicht wurde:

- Meiner Partnerin Denise Haslwanter, für Ihre ständige Unterstützung, Ihren Rückhalt und Ihre Ermutigung.
- Meiner Familie, insbesondere meinen Eltern Elfriede Lindner und Peter Lindner, meinem Stiefvater Wolfgang Murth und meinen Großeltern Hildegard Lindner und Helmut Szymonik für zahlreiche gute Ratschläge, motivierende Gespräche und Hilfestellungen.
- Thomas Artner und Wolfgang Gruber für die tolle Unterstützung während der Arbeit.
- Professor Thomas Grechenig für die Möglichkeit diese Arbeit durchzuführen.
- Meinen Studienkollegen für die gegenseitige Motivation während dem Studium.
- Allen anderen Freunden und Kollegen für schöne Zeiten, Diskussionen und Denkanstöße.

Mein besonderer Dank gilt meiner Großmutter Elfriede Kritsch, ihr möchte ich diese Arbeit widmen.

Dankeschön!

Abstract

Information and Communication Technology (ICT) is subject to fast progression where technologies and methods are permanently evolving and changing. This situation demands special requirements of education and training in this field to ensure the availability of highly educated specialists.
This master thesis presents a technical foundation for software engineering education to support teachers in the creation of industrial-like assignments. A main feature is the customization to specific needs of various academic courses with different main topics. Possible fields of application of this framework are lectures regarding software engineering, software testing, software quality assurance or security.
To ensure this feature, the basis framework was designed in a way that components and technologies can be exchanged. The foundation consists of loosely coupled components and utilizes state of the art technologies. It also provides extension points for future developments. It is based in the domain mobile ticketing due to it's many cross cutting concerns, omnipresence and support from modern technologies.
Research was performed to elaborate the common knowledge a well-trained software engineer must possess, as well as the current state of teaching software engineering in an academic setting. In addition, the current level of development of mobile ticketing along with supporting technologies were surveyed. The results of the research are presented in this thesis.
One possible customization of the foundation is demonstrated by the concrete example of a software engineering course, where the existing technical base was replaced. The course assignment consists of an industrial-like project setting for groups of four to six students. Based on the course assignment and the replaced foundation, each group had to develop a ticketing application.
An evaluation was performed to assess the quality of the application of the base framework in context of this course. A questionnaire was designed and given to tutors, who supervised students during the course. The results of the questionnaire is stated and interpreted. The evaluation results show that the learning objectives were well imparted. Students were highly motivated and most of the projects were successfully finished. Some minor issues arose, which will be addressed in future courses.

Kurzfassung

Informations- und Kommunikationstechnologie (IKT) ist ein Fachgebiet, das einer ständigen Weiterentwicklung unterliegt. Dies stellt besondere Ansprüche an die Ausbildung in diesem Fachgebiet, um exzellent geschulte Fachkräfte sicherzustellen. Die vorliegende Diplomarbeit präsentiert ein technisches Basisframework um Lehrende in dem Bereich Software Engineering in der Erstellung von industrienahen Aufgaben zu unterstützen. Ein Hauptmerkmal ist die gezielte Anpassung des Basisframeworks auf spezifische Lehrveranstaltungen mit unterschiedlichen Schwerpunkten wie etwa Software Engineering, Software Testen, Software Qualitätssicherung und Security.

Die Anpassbarkeit ermöglicht es, dass Komponenten und Technologien des Basisframeworks austauschbar sind. Das System besteht aus lose gekoppelten Komponenten und verwendet Technologien, die am aktuellen Stand der Technik sind. Um zukünftige Entwicklungen zu integrieren, ist die Möglichkeit zur Erweiterung gegeben. Das Themenumfeld der technischen Basis ist mobiles Ticketing, da dieser Anwendungsbereich verschiedene Querschnittsthemen besitzt, allgegenwärtig ist und durch moderne Technologien unterstützt wird.

Eine Recherche wurde über das Wissen, welches ausgebildete Software Entwickler besitzen sollten, durchgeführt. Weiters wurden die aktuellen Methoden der Ausbildung im Bereich des Software Engineering erhoben. Betreffend mobilen Ticketing wurde der Stand der Technik und die unterstützenden Technologien ermittelt. Die Ergebnisse der Analysen werden in der Arbeit präsentiert.

Eine exemplarische Anpassung wird an dem konkreten Beispiel eines Software Engineering Kurses gezeigt, bei dem das technische Basisframework ausgewechselt wurde. Die Aufgabenstellung war die Umsetzung einer industrienahen Ticketing-Applikation durch die Studierenden in Gruppen von vier bis sechs Personen.

Der Einsatz des Basisframeworks im Kontext dieses Kurses wurde mittels eines Fragebogens evaluiert. Ziel war es festzustellen, ob der Kurs der Planung entsprechend durchgeführt werden konnte. Die Fragebögen wurden von Tutorinnen und Tutoren beantwortet, da sie durch die wöchentlichen Treffen den besten Einblick in die Gruppenarbeiten hatten. Die Ergebnisse des Fragebogens werden angeführt und ausgewertet. Die Evaluierung zeigt, dass durch den Einsatz des Basisframeworks die Lehrziele erreicht wurden. Die Studierenden waren hoch motiviert, und der Großteil der Teilnehmer konnte das Projekt erfolgreich abschließen. Die Evaluierung zeigte auch Verbesserungspotential auf, das in zukünftigen Kursen umgesetzt wird.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement and Motivation

Information and communication technology (ICT) is subject to fast progression where methods and technologies are permanent evolving. Due to this fast progression, the instruction of this discipline must also be subject to permanent evolution. Apprenticeship must be designed for change to keep pace with the technological developments to ensure high quality in education and training. New methods and technologies must be validated, important ones selected and included in the curriculum.
Good education in software engineering includes solving tasks that contain various cross-cutting problems. The construction of exercises can be supported by utilizing a methodical and technical foundation that offers a wide range of possible problem statements. The goal of this thesis is the development of such a technical foundation, which is capable of teaching the current state of the art in software engineering and is flexible enough for ongoing developments.
This foundation should be designed in an open fashion, e.g. it should use open standards and interact with existing technologies, systems and platforms. The concept should consist of loosely coupled components to support the changeability of individual components. These features make adjustments for objectives of specific courses possible. Therefore a system architecture must be developed that achieves this requirement.
The domain chosen for this thesis is mobile ticketing. It includes industrial characteristics, is supported by a wide range of state of the art technologies and is generally well-known by students due to its omnipresence in every day activities.
Internet-enabled, personal, trusted devices and mobiles have undergone rapid and wide spread propagation during the last years. New technologies that match perfectly with mobile ticketing have arisen. Nowadays, people are increasingly relying on smart phones to help them with their daily tasks. This makes mobile ticketing a highly interesting playground to employ these new developments.
The main feature of mobile ticketing is that there is no need to print a ticket on paper. The core functionality for a user is to obtain a ticket remotely on their mobile. For this purpose, the ticket

must be transferred to and stored on the user's device. For proper redemption of a ticket, it must be checked for validity. In order for the ticket to be validated, it has to be read from the mobile. After redemption, it must be assured that the ticket is devaluated. Scanning a ticket must be easy and reliable.
Industrial characteristics of mobile ticketing that can certainly be imparted to students in an educational context are as follows:

- Security: Like classic tickets, mobile tickets must be secure against forgery. Tickets must only be issued by authorized entities. Without prior approval, a ticket should not be altered. Alteration and devaluation must therefore be traceable.
- Availability: Validation and devaluation must be possible at any time. Hereby the possibility of failure or unavailability of system components must be taken into account. Therefore the system must be fault tolerant and the validity of tickets should be determinable offline without a connection to a central server.
- Performance and Scalability: It should be possible to validate a large amount of tickets in a short time frame to minimize waiting time. This is achievable by automating the validation process. The system should be scalable to feature the same performance at high load.
- Openness: Open standards and interfaces should be used to assure interoperability with existing platforms and independence of proprietary solutions.
- Usability: Persons who are not familiar with technology should be able to use the system. Therefore, operation of the system must be kept simple and the technical details transparent to the user.

1.2 Aim of the Work

The aim of this master thesis is the development of a technical and methodical foundation for education, which includes different important cross-cutting concerns of software engineering. The foundation should consist of a concept and a system design for mobile ticketing.
The concept and system design should reflect requirements of industrial solutions. The current technological and systematical state of the art of software engineering has to be taken into account. The foundation should feature a realistic setting with different addressable problem statements that include cross-cutting concerns of the engineering discipline of informatics. Based on this setting, assignments which cover different topics of software engineering shall be derivable. Possible topics are software engineering, software testing, software quality assurance, security and usability engineering.
A mobile ticketing workflow consisting of creation, issue to the user, redemption and devaluation has to be developed. Each of these events should be traceable. The system should utilize digital signatures to prevent ticket forgery and to support offline validation. Signing and validating a ticket requires a key pair, consisting of a private and a public key. The ticket should be signed with the private key before it is issued. At redemption the authenticity of the ticket is

validated with the tickets signature and the public key. To provide offline validation the public key and an optional list of invalid ticket identifiers must be transfered to the device before the validation.
The design should consist of loosely coupled components. This can be accomplished by utilizing representational state transfer (REST) [23], a suitable architectural style that achieves a loose coupling based on the hyper text transfer protocol (HTTP) [24].
The system should consist of the following parts:

- Ticket Server: The ticket server is the central part of the system. It should include the business logic and provide REST interfaces for accomplishing the various use cases.
- Ticket Server Management Application: The management application utilizes the ticket server's REST interfaces for management tasks.
- Merchant Application: The merchant application utilizes the ticket server's REST interfaces for ticket handling.
- Mobile Client Application: The mobile client holds tickets and transfers them to the mobile validation application for validation and devaluation.
- Mobile Validation Application: The mobile validation application validates tickets.

In summary the following problem statements will be covered in this master thesis

- What is the knowledge that a trained software engineer should possess? Which approaches and methods are utilized in software engineering education and training?
- Which requirements are needed for a technological foundation in the domain of mobile ticketing to provide students a realistic and technical state of the art learning environment? How can this foundation be utilized in education?
- Which design questions must be addressed to create a technological state of the art foundation for a modern mobile ticketing system including appropriate security concepts?
- What are the workflows in a modern mobile ticketing system?

1.3 Methodological Approach

At the beginning of this thesis a literature inquiry about the following topics is carried out:

- Important knowledge of software engineering: This inquiry aims to evaluate the knowledge that a trained software engineer should possess to solve industrial problem statements.
- Teaching and knowledge transfer in software engineering: The aim of this inquiry is to investigate teaching and knowledge transfer methods to prepare students optimally for industrial projects.

- Mobile ticketing: The aim of this inquiry is to establish the state of the art regarding mobile ticketing. Existing systems with their methodology and implementation are reviewed.
- State of the art technologies: This inquiry aims to determine industrially used state of the art technologies for the possible implementation of a mobile ticketing system.

Based on the literature inquiry the state of the art and objectives of software engineering apprenticeship are documented. Different methods of instruction are described.
Afterwards an introduction to mobile ticketing along with its development is given. Existing systems are described and related services for mobile ticketing are presented. State of the art technologies that are used in industrial projects and are suitable for a mobile ticketing system implementation are described.
Based on the state of the art of software engineering apprenticeship and mobile ticketing, a mobile ticketing system concept is created. During the design process the application in academics and compliance to industrial standards are taken into account. The concept is developed for general application in software engineering education with support for customization to specific courses.
Thereafter, the system design along with the used technology stack is created and documented. It consists of the domain model, the software architecture and the definition of security aspects. The system design describes the ticket server, the management application, the merchant application, the mobile client application and the mobile validation application.
After the design of the system, the application in and tailoring to a software engineering course is presented. The course itself already exists but the current technical foundation shall be replaced with state of the art technologies and a modern software architecture. The course is described along with the problem statements and the presentation of the tailored foundation. It is also shown which knowledge areas of the software engineering body of knowledge (SWEBOK) [10] are addressed in the application of the foundation in the course.
After that, the system design and application in the presented course is evaluated. Hereby a questionnaire is created and given to tutors, who guide students in the context of the presented course. The analysis of the questionnaire is given and findings are presented.
At the end of the work the results are summarized and possible extensions for future research are stated.

1.4 Structure of the Work

This work is separated in seven main chapters and three appendices. The following describes each chapter and appendix briefly.

The current chapter (chapter 1) introduces this master thesis.

Chapter 2 describes the basics and the state of the art of education in software engineering. The objectives and methods of software engineering education are stated.

Mobile ticketing is introduced in chapter 3. The history and development of mobile ticketing is presented and related services are described. Related work and existing systems concerning mobile ticketing are documented. Technologies that are suitable for a mobile ticketing system are mentioned.

A mobile ticketing system foundation for education is presented in chapter 4. This chapter identifies the possible stakeholders of the system. Ticket properties and the validation concept are presented. The system's overview is illustrated. The concept and design of each component is described.

Chapter 5 describes the customization of the mobile ticketing system foundation for application in the context of a software engineering course.

The evaluation of the presented customization from chapter 5 is presented in chapter 6. The evaluation design and the results are given.

Chapter 7 summarizes the outcome of the master thesis and states future work.

The appendices A, B and C list proposed requirements for the mobile ticketing system foundation, user stories for the software engineering course, the survey in german and uninterpreted results of the evaluation survey.

CHAPTER 2

Basics of Software Engineering Education and Training

This chapter gives an introduction to the basics of software engineering education and training. The knowledge that forms the engineering discipline software engineering, according to the guide to the software engineering body of knowledge (SWEBOK) [10], is presented. State of the art education and training methods of software engineering are stated afterwards.

2.1 Software Engineering Body of Knowledge

The third version (V3.0) of the guide to the software engineering body of knowledge [10] was released at the end of 2013. The purpose of the SWEBOK guide is the presentation of the commonly accepted part of the software engineering body of knowledge. One of the main objectives of the SWEBOK guide is to provide a basis for the creation of curricular.
The SWEBOK guide organizes the discipline software engineering in 15 knowledge areas (KA), which are:

- Software Requirements
- Software Design
- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration Management
- Software Engineering Management

- Software Engineering Process
- Software Engineering Models and Methods
- Software Quality
- Software Engineering Professional Practice
- Software Engineering Economics
- Computing Foundations
- Mathematical Foundations
- Engineering Foundations

The knowledge areas are described in the following as they are presented by the SWEBOK guide [10]. Software engineering is an interdisciplinary field which requires knowledge of other disciplines. The SWEBOK guide identifies seven related disciplines, which are:

- Computer Engineering
- Computer Science
- General Management
- Mathematics
- Project Management
- Quality Management
- Systems Engineering

SWEBOK KA Software Requirements The knowledge area software requirements addresses elicitation, analysis, specification and validation of software requirements and their management through the software life cycle. A well done requirement management process is important since the software requirements state the desired properties of a software product.

SWEBOK KA Software Design According to the SWEBOK guide and ISO/IEC/IEEE [44], software design addresses:

- The process which defines the design of the system or component, consisting of the interfaces, the components, the architecture and all other characteristics.
- The artifacts, which are created through this process.

The software design is built up by various models, which form a blueprint of the solution. These models may be evaluated for their applicability for building the solution and alternatives may be identified. The resulting design can be used to plan development tasks. The addressed knowledges are fundamentals, key issues, methods and strategies of software design.

SWEBOK KA Software Construction This knowledge area addresses all necessary tasks of the creation of working software. Involved tasks are coding, verification, debugging, unit testing and integration testing. The fundamentals, management and technologies of software construction are described.

SWEBOK KA Software Testing The knowledge area software testing is about the verification of a software. Because the execution domain of a program is usually infinite, a representative set of test cases must be selected for software testing. From these test cases the expected behavior is validated. The fundamentals and methods of software testing are stated.

SWEBOK KA Software Maintenance Software maintenance denotes all activities which are required to provide a cost-effective support for software. After the delivery the need for changes may arise, which demands alteration of the software product. That is addressed by this knowledge area, which contains the subjects fundamentals, key issues, processes and techniques of software maintenance.

SWEBOK KA Software Configuration Management Configuration management is a supporting software lifecycle process, which identifies system configurations at different points of time. This information ensures that changes in the configuration are controllable and the their integrity and traceability is maintainable throughout the life cycle of the system. This knowledge area describes the management, identification and control of software configurations.

SWEBOK KA Software Engineering Management Software engineering management is defined by the SWEBOK guide as "the application of management activities [...] to ensure that software products and software engineering services are delivered efficiently, effectively, and to the benefit of stakeholders" [10]. This knowledge area includes scope definition, project planing, software review, evaluation and measurement techniques.

SWEBOK KA Software Engineering Process The knowledge area software engineering processes includes concerned tasks and activities to develop, maintain and operate software. Included topics are software processes, software life cycles, process assessment, improvement and measurement.

SWEBOK KA Software Engineering Models and Methods Software engineering models and methods give structure to the software engineering process. With these structures, software engineering shall be more success-oriented, systematic and repeatable. This knowledge area includes subjects about modeling, methods and analyses of software engineering.

SWEBOK KA Software Quality The SWEBOK guide states that many different definitions of software quality exist. According to the SWEBOK guide, software quality is the conformance of software to all defined requirements. The knowledge area contains topics about the fundamentals, management processes and practical considerations of software quality.

SWEBOK KA Software Engineering Professional Practice Software engineering professional practice is referenced by the SWEBOK guide as the "knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner" [10]. This knowledge area describes how software engineers act in a professional manner, deal with group dynamics and communicate efficiently.

SWEBOK KA Software Engineering Economics This knowledge area is about economic and business decisions in the discipline software engineering. It describes the fundamentals and analysis of economics and economics in the context of life cycles and risks.

SWEBOK KA Computing Foundations This knowledge area includes knowledge about the development and the operational environment in which software is executed and evolving. It includes the basics of problem solving, abstraction, programming, debugging, data structures, algorithms, complexity, systems, compilers, databases, networks, parallel and distributed computing, human factors and secure development.

SWEBOK KA Mathematical Foundations Software Engineering requires mathematical knowledge since the basis of programs is mathematic. This mathematical knowledge is addressed in the mathematical foundation knowledge area which includes fundamentals about sets, relations, functions, logic, proof techniques, counting, graphs and trees, discrete probability, state machines, grammars, numerical precision, algebraic structures, accuracy, errors and number theory.

SWEBOK KA Engineering Foundations Software engineering is based on skills and knowledge, which are in common to all engineering fields. These skills and knowledges are addressed by the knowledge area engineering foundations. It consists of knowledge about empirical methods, experimental techniques, statistical analysis, measurement, modeling, simulation and prototyping.

2.2 Related Work to Software Engineering Education

A literature research was performed to determine the organization and methods of courses related to software engineering. The most relevant conference series, which deal with this topic, are Frontiers in Education (FIE) and Conference on Software Engineering Education and Training (CSEE&T). In the following, related work for this master thesis from these conferences is presented.

2.2.1 Improving Software Engineering Education through Enhanced Practical Experiences

Tian, Cooper, and Zhang present in their article [68] a problem based learning approach to software engineering. The presented teaching method was employed in a course of the computer science department at the University of Texas, Dallas. The goal of the course was to strengthen

student's software engineering skills through practical project experience.
The teacher brought in different real-world project proposals. The projects had different topics and required different skills like systems programming, web application development or mobile application development. The students organized themselves into teams. Each team chose a project proposal for development.
The course was designed so that each project ran through the software engineering phases requirement analysis, project planning, architectural design, detailed design, implementation and testing and product presentation. Each team presented their project work and progress regularly after every finished phase. Subsequently to the presentation, the whole class discussed their work, the problems that occurred and lessons learned. The discussions were moderated by the teacher and their assistants. The goal of these discussions were to maximize each of the student's learning opportunities.
The students had to participate in the project, to coordinate with the client, the teacher and the assistant, to present the project status, to participate in the discussions, and to evaluate the other teams work. The required deliverables were the project documentation, the program code and the software.
Tian et al. discovered that this teaching method was generally well received by the students. Student participation was higher compared to other software engineering courses. They also stated that this method of teaching gave the students the opportunity to gain real-world project experience which in turn gave the course participants an advantage in the IT industry compared to other graduates.

2.2.2 Using Community-based Projects in Software Engineering Education

Roshandel, Gilles, and LeBlanc present in their article [61] an education program, where students work on projects for local non-profit organizations. Thereby students gain real-world project experience and contribute to the local community.
Roshandel et al. state that their department partners every year with a local non-profit organization, which is in need of a software solution. This software solution is considered as a project, which is used as a case study by the department. The case study is used through different courses with different objectives of their master program. The courses are set in sequence, starting in fall, continuing in winter and finishing in spring. Created deliverables are used on succeeding courses. In fall, students elaborate the requirements and the software architecture. During winter, the system design is created and in the spring term a prototype is developed.
Students form teams of size of three to five persons and work together on assignments related to the project. They meet regularly with representatives of the non-profit organization to review the work and to receive input. After each course, the results are given to the non-profit organization. At the end of the course sequence, the final outcome of the case study is presented to the non-profit organization and the campus community.
Roshandel et al. state in their article that students are motivated by the real-world project experience and the possibility to contribute to non-profit organizations. The non-profit organizations benefit from the cooperation through reflection of their business rules and goals. They also get in touch with software developing processes which helps in future interactions with software developers. Depending on the outcome, the developed artifacts may be considered as an in-depth

analysis of their problem, which can be used for further development, or as a production ready software.

2.2.3 Teaching Web Engineering using a Project Component

Krutz and Meneely describe in their report [50] a method of teaching web engineering by using a project component. Students formed teams of size of four to six members at the beginning of the course. Each team had to develop a web application by building custom components and utilizing existing components through web services and application programming interfaces (APIs).
The teacher had two roles, the teaching role and the customer role. In the teaching role he gave students advice. In the customer role he mimicked a real world customer.
The project utilized the Facebook APIs for authentication and data retrieval. Beside them other data services and feeds were used to retrieve information like weather forecasts or stock exchanges. Students had to aggregate information from the external systems with internal data. Also they had to implement a chat feature with HTML5.
Different virtual machines were provided to the teams to simulate different environments for their project, like development, staging or production. So students encountered a realistic, industrial like project setting.
The course took ten weeks. In the first weeks, students should get an understanding about web engineering and team formation. Each team had to determine a team, development and testing coordinator. In the third week they had to finish the requirements document and in the fourth week the design document. Both documents should evolve further during the project.
During the project, students were given the opportunity to ask the teacher in his role as the customer questions. Hereby students were also encouraged to present artifacts like screenshots or prototypes. With these meetings the interaction with a real world customer should be simulated.
In the sixth week students had to deliver a fully functional version of their application along with updated documents, test plans and implemented tests. The appearance of the application was neglected in this release. The teams gave a presentation about their work and self reflected on it.
In the seventh week the teams released their work to students of a concurrent security course where their applications were fuzzy security tested. The results were given back to the teams.
In the tenth week, students had to deliver the final, fully functional version. In difference to the first release, the application had to have a proper appearance and had to be mobile device friendly. Again, students presented and self reflected on their work.
Krutz and Meneely state that students were motivated by the real world aspects and the utilization of proper technologies, tools and techniques. Students feedback was positive regarding the course and the project. Students stated that they benefited from the application of different APIs and web services.

2.2.4 Teaching Object-Oriented Software Design within the Context of Software Frameworks

Ali, Bolinger, Herold, Lynch, Ramanathan, and Ramnath point out in their article [1] that novice software developers abandon good design practices when they build a software using a framework.
Therefore they present an approach to overcome that problem in the context of the Android framework. They state that a good design in the context of a framework can be achieved by explaining the framework in terms of design patterns.
Their methodology to teach good design practices in the context of software frameworks consist of three steps. In the first step, the application is designed object oriented. In the second step design patterns, which are available in the framework, are applied to the object oriented design from the first step. In the third step the design from step two is changed so that the applied patterns fit within the context of the framework.
Ali et al. conclude that a good design can be achieved within a framework by following their presented method.

2.2.5 Teaching Web Services and Service-Oriented Architecture using Mobile Platforms

Sherriff describes in his article [66] a course which teaches the concepts of web services and service oriented architecture. In the course students form teams of size of three. Each team had to develop a number of web services and a mobile application, which utilizes these services. The assignments were planned so that they build upon each other. The goal of the course was to develop a global positioning system (GPS) supported mobile campus tour application. The utilized technologies were the operation system Windows Mobile 6.1 and the programming language C# for the mobile application and the programming language Java along with the web service framework Apache Axis2 [1] for the server.
The requirements of the mobile application were split in core and optional requirements. Each team had to complete all core requirements and three optional requirements. Students could also propose own optional requirements of similar difficulty.
The mobile application retrieved data from web services, which were also developed by the teams. Each week, the teams presented their work to the teachers. At the end of the semester, students had an oral exam about web services and mobile development.
Sherriff states that mobile devices are a perfect platform for teaching web services and cloud computing concepts since they are pervasive and offer many possibilities. Students were excited to use state of the art technologies and platforms, and developing skills which they can use immediately. Other motivational factors for students were realizing their own ideas and developing an application which can be used by themselves and their friends.

[1] `https://axis.apache.org/axis2/java/core/`, accessed 25/07/2014

2.2.6 How to Involve Students in FOSS Projects

Ellis, Hislop, Chua, and Dziallas describe in their work [22] how to involve students in free and open source (FOSS) projects. They state that students may gain real-world experience in a professional environment. Their work is publicly available after the course and may serve as a portfolio and reference for future employers. Students also learn to work in a large and distributed community.
Ellis et al. identified five phases to involve students in FOSS projects, which are described in the following.
In the first phase teachers should understand the commonalities and differences between the academic and the FOSS project's culture. Commonalities are that in both cultures participants are encouraged to learn, work should be released early and often and multiple people should review it. Reusing existing work is also a commonality but while in the FOSS culture existing code is reused, in an academic setting scientific work is usually only cited. Differences are that in FOSS projects everything is publicly available while in an academic setting, students are not allowed to share their work. In FOSS projects, multiple reviews are given by different people. In an academic setting the teacher usually reviews only the final submission. Students start assignments from scratch in contrast to FOSS projects, where already implemented solutions are reused. FOSS projects are more flexible in their schedule than academic courses, which are planned in advance.
In the second phase teachers should identify capable FOSS projects and mentors. Criteria for the selection of a project are beginner friendliness, possibility to integrate students work, a responsive community and commitment from project's key mentors.
The third phase is about synchronization between the FOSS project and the academic setting. Hereby, the schedule and objectives of the course are communicated to the project mentors. The ways of conversation channels used between the project members and the students are defined. The dependencies and interactions between project and course milestones are identified. If possible, local project members are invited to the campus to act as an on-site project contact. The project development setup instructions are reviewed and the commit process is specified. Potential student tasks are identified.
In the fourth phase students participate in the project. They should get into the project and setup the tools and development environment. Students and project members should get into contact as soon as possible. Students should commit, document and reflect their work publicly.
In the fifth phase at the end of the term, the students should finish their work, document it and be contactable for further questions. If a student wants to leave the project he or she should do it gracefully and not leave important work or tasks undone.
Ellis et al. summarize that involving students in FOSS projects offers many learning opportunities but teachers have to do some work in advance. The FOSS community benefits from the students work and students may be encouraged to participate beyond the course.

2.3 Conclusion Basics of Software Engineering Education and Training

This chapter has introduced the basic knowledge that a well educated software engineer should possess according to the SWEBOK. The introduced knowledge areas were taken into account in the design of the mobile ticketing application. The important ones in context of this thesis are software design, software testing, software quality, software engineering process, software engineering models and methods, and software engineering professional practice.
Related work of software engineering education was researched and is presented in this chapter. The gained knowledge from this research was taken into account in the design and application of the mobile ticketing foundation. The key points are that students should present their work regular to the teacher. An assignments should feature industrial needs and utilize state of the art technologies to provide excellent education and to keep the student motivation high. Students should instantly benefit from the gained knowledge and should be able to apply it.

CHAPTER 3

Introduction and Related Work regarding Mobile Ticketing

This chapter illustrates an overview to mobile ticketing. Mobile ticketing along with its development is introduced. Related work of mobile ticketing is stated and capable technologies for building a mobile ticketing system are presented afterwards.

3.1 Introduction to Mobile Ticketing

This section introduces mobile ticketing. First the history and development of electronic commerce (e-commerce), mobile commerce (m-commerce) and mobile ticketing are described. Then mobile ticketing and the advantages compared to traditional paper based ticketing are explained. Mobile tickets can either be validated online or offline. These two different approaches are discussed. Usability aspects of and business models for mobile ticketing are stated. At the end of the section, services related to mobile ticketing are documented. These are location based services and mobile payment services.

3.1.1 Definition and Development of E- and M-Commerce

Mobile ticketing is an emerging type of commerce. It belongs to m-commerce, which is a subdivision of e-commerce. This section introduces and describes e-commerce and m-commerce.

3.1.1.1 Definition and Development of E-Commerce

Kalakota and Robinson state in the encyclopedia of computer science [48] that “Electronic commerce (or e-commerce) enables the execution of transactions between two or more parties using interconnected networks”. Hereby, a transaction is defined as an exchange between two entities, where one entity sells the other entity a product or a service. Kalakota and Robinson state that e-commerce is more cost efficient, allows faster transactions and has a better performance than

traditional commerce. In the beginning, e-commerce was only done by electronic interaction, e.g. by electronic mail or online conversations. In the 1990s, easy-to-use technological solutions opened e-commerce to the public and gave it the essential push.
Kalakota and Robinson describe in their book [47] that the development of e-commerce is mainly pushed by economic forces, customer interaction forces, and technology-driven convergence. The main economic force behind e-commerce is the reduction of costs for doing business. The possibility to achieve a greater market presence and to provide additional market channels are the most important customer interaction forces. Also the possibility to target small micro segments is important in this context.
The term product isn't bound to physical things. A product can also stand for digital content, like information, music, films or electronic books. Kalakota and Robinson divide the technology-driven convergence in convergence of content and convergence of transmission. Convergence of content allows to format digital content in an appropriate manner to use it. Convergence of transmission is about delivering digital content over a network to the customer. Both have lead to a greater amount of offered digital content, whose sale has increased dramatically. For example, Amazon Inc. stated in their press release from May 19, 2011 [2] that they are already selling more electronic books than printed books.
Song and Dong describe in their article [67] that the influence factors of e-commerce were social development and commercial theory, since e-commerce depends on the social environment and how business is done in it. Other influencing factors are system science and information technology, because they define the technical capabilities of e-commerce. Song and Dong also point out that e-commerce will increase its field of application and will focus more on people in the future.
Kalakota and Robinson divide e-commerce into three classes [48]:

- Inter-Organizational: E-Commerce between two different companies (business-to-business).
- Intra-Organizational: E-Commerce inside of a company (within a business).
- Customer-to-Business: E-Commerce between a company and an end customer, which is the usual context for mobile ticketing.

3.1.1.2 Definition and Development of M-Commerce

Huang, Liu, and Wang state that the rapid diffusion of mobile internet-enabled devices has led to a new wave of e-commerce, the m-commerce [32]. Barnes pointed out that m-commerce is any transaction with a monetary value, either directly or indirectly, which is conducted over a wireless network [8]. Therefore m-commerce is location independent and can be performed anytime.
According to Huang et al., m-commerce applications have the two major attributes mobility and reachability [32]. Furthermore, Huang et al. state that according to these attributes the following services are important for m-commerce [32].

- Mobile Communication Services: These services make it possible to communicate anytime, anywhere trough e.g. e-mail, SMS or voice.
- Mobile Information Services: these services are about provision of personalized information to a specific person in a desired format on a mobile device. This provision may be supported by additional information like the geographical position of the requesting person. Details about location based services are given in section 3.1.3.1.
- Mobile Transaction Services: Mobile transaction services offer the opportunity of mobile payment. Huang et al. believe that these services are the most crucial services of m-commerce which may replace banks, ATMs and credit cards by mobile money. Details about mobile payment are given in section 3.1.3.2.
- Mobile Interaction Services: Mobile interaction services are about entertainment. With these services the customer can enjoy mobile games, music or videos on the go.

3.1.2 Definition of Mobile Ticketing

Mobile ticketing is a subdivision of m-commerce. Since the ticket itself isn't a physical object, it is perfectly qualified to be distributed via m-commerce. This section gives an introduction to mobile ticketing and its related challenges.

3.1.2.1 Ticket Definition

Generally, a ticket, a voucher or a coupon are placeholders for a service or an item. These are special tokens, which have a similar intention. This thesis denotes such tokens generally as tickets. At a certain time in the future, the ticket is exchanged for the item or service, which it stands for. Depending on the ticket's purpose and the related domain, it may have different types and different modes of validation and devaluation.
The following list states and describes different types of tickets:

- Single Use Ticket: The most common ticket is the single use ticket. This type is issued once and can be used only once. After using it, it becomes devaluated. Common fields of applications of this type are tickets for events or public transport single tickets.
- Multi Use Ticket: The multi use ticket is issued once and can be used a predefined number of times. After its last use it becomes completely devaluated. This type is also common in public transport and for services, which may be used multiple times.
- Time Dependent Ticket: The time dependent ticket becomes valid at a certain point in time and is valid for a predefined time span. During this time span it may be used arbitrarily often. After the time span it becomes devaluated. This type is used for season tickets and also for transit passes in public transport.
- Custom Ticket: Langer and Roland state in their book [51] that a custom ticket is another possible type. Hereby, the custom ticket is generated by a check-in and check-out

procedure. They state that this principle was applied for the public transport company Rhein-Main-Verkehrsverbund (RMV). A passenger touches a terminal with his near field communication (NFC) enabled phone when boarding to check-in. At deboarding, he touches the terminal again. Depending on the check-in and check-out location the best price is automatically calculated and passed to the customer. At the end of the month, the customer is billed for the taken rides. This method has the advantage that the customer does not have to take care about buying the correct ticket. Instead a valid best-price ticket is automatically created.

- Prepaid Ticket: This ticket holds a specific amount of money, which has been prepaid when the ticket was bought. The ticket can be used multiple times where each time its value is reduced by the redeemed amount.

A ticket may also hold additional information which can be intended for humans without any semantic meaning (e.g. seat, plane departure time, ...) or for artificial agents or machines with a semantic meaning (e.g. a signature for validation or an ID).

3.1.2.2 Advantages of Mobile Tickets compared to Paper Based Tickets

Mobile ticketing is a new business model for selling tickets to customers. For successful user acceptance, users and merchants must be made aware of and must have a benefit from mobile ticketing.
Traditional ticket systems are usually paper based. Hereby, the ticket is a piece of paper holding information. It is bought either at a ticket vending machine or from a shop assistant either at a counter or over the phone. The ticket is redeemed either by a person or a machine, which scans the code on the ticket.
In comparison to traditional ticketing, mobile ticketing offers many advantages. The ticket is no longer bound to a physical sheet of paper. Instead, it is represented as data, which can be stored on a mobile. A ticket can't be lost or forgotten as long as the user has his mobile with him. Mobile ticketing makes it possible to buy a ticket anytime, anywhere on the mobile. So there is no need to line up in a queue or for using a ticket vending machine. This is also an important aspect for merchants, since they may save money by providing a mobile ticketing system and therefore reduce the amount of cost intensive counters or ticket vending machines.
Neefs, Schrooyen, Doggen, and Renckens state in their article [58] that event organizers can benefit from logging the purchase behavior of their customers. The acquired information may be useful in planning future events.
Neefs et al. also present in their article [58] the results of the comparison of paper ticketing and electronic ticketing based on the offline system Tapango. They state that user acceptance is the most important key factor for a successful electronic ticketing system. For keeping user acceptance high, they defined the three main goals compactness, speed and cross-event usability for the Tapango system. The Tapango system itself is introduced in section 3.2.4.
One result of their research is that the paper based system is slightly faster than Tapango. But the time difference was hardly perceived by the users and had no negative impact on the acceptance. They state that the round-trip-time was independent of the amount of ordered tickets when using

Tapango. On the contrary the round-trip-time depended on the amount of ordered tickets when using the paper based system. Also, they state that average users are likely to exchange multiple tickets at the same time, which is handled faster by an electronic system.
Neefs et al. describe that over 80% of the test users were fully convinced by Tapango and thought that such a system has the potential to replace traditional systems.

3.1.2.3 Use-Case Scenarios

Widmann, Grünberger, Stadlmann, and Langer enhanced the existing core application of the Association of German Transport Companies (VDV) to support NFC ticketing [70]. Thereby they identified the following use cases for NFC ticketing. The extension is described in detail in section 3.2.2.

- Application Distribution: The user installs an application on his NFC enabled mobile to purchase, store and use acquired tickets. This use case is about the easy distribution of the application to the user.
- Ticket Distribution: The user utilizes his phone with the installed application to buy tickets. This use case ensures that the user can acquire and receive tickets on his or her mobile.
- Displaying Tickets: The application stores the acquired tickets. It shows an overview of the stored tickets and also displays detailed information of a ticket.
- Inspection: When the user wants to redeem a ticket from his device, the ticket must be transferable and ready for validation.
- Blacklist Management: In certain circumstances the usage of already distributed tickets should be prevented. To ensure this requirement, such tickets become blacklisted in a blacklist. This list must be generated, updated and distributed.
- Block Requests: A prior issued ticket can be put on the blacklist by a block request. A block request can be initiated either by a customer (e.g. when his device holding the tickets was lost or stolen) or by a merchant (e.g. the customer hasn't paid for the ticket). When the block request is accepted, the tickets will be blacklisted by a block command.
- Block Commands: If the prior initiated block request was accepted, the block command blacklists the ticket by adding it to the blacklist. From this point the ticket will not be accepted when someone tries to redeem it.
- Block Removal Requests: With the block removal request the blacklisting of a prior blacklisted ticket can be undone. When the block removal request is accepted, the ticket will be removed from the blacklist by a block removal command.
- Block Removal Commands: If the prior initiated block removal request was accepted, the block removal command removes a blacklisted ticket from the blacklist. From this point the ticket will be accepted again when someone tries to redeem it.

Chaumette, Dubernet, Ouoba, Siira, and Tuikka compare in their article [14] two different architectural styles for a mobile ticketing system. They based their comparison on six use cases, which describe an event ticketing scenario. The following enumeration lists five of these six use cases, which are directly related to the mobile ticketing process. The comparison is described in section 3.1.2.4.

- Selection of Events: Available events are presented to the user. He can browse through them to find one that he is interested in.
- Event Visualization: When the user has found an event he is interested in he can view detailed information about it. For this purpose the user receives a detailed presentation about the event.
- Tickets Issuance: The user can buy one or several tickets for an event he is interested in. The tickets are issued and sent to his mobile.
- Ticket Presentation: The ticket is stored on his mobile and is presented or transfered in a way which makes it possible to validate it at the event site.
- Ticket Exchange: A user may transfer tickets he bought to an other users mobile. This use-case ensures that a user can buy tickets for friends or can sell tickets.

3.1.2.4 Offline versus Online Ticketing Systems

Chaumette et al. describe and compare in their article [14] two different architectural styles for an NFC based ticketing system, one with an online, the other one with an offline approach. These approaches are compared in terms of user experience, security, economical aspects, reliability and speed of use.
The difference between these two approaches is that in the offline version the ticket is verified without any other infrastructure and communication with other systems. So, the ticket must contain all information needed for validation and may also be signed. In this case the validator has to check the signature and to avoid the reuse of the ticket he must either keep track of already used tickets or has to devaluate the ticket after checking.
In the online version, the ticket is stored in the issuer's backend system and references the static identifier of the phone's secure element[1]. With this static identifier the ticket can be allocated to the user's phone. The ticket verifier reads the static identifier and checks the validity of the referenced ticket through the ticket issuer's back-end system.
Chaumette et al. state that these two approaches are equivalent in security, economical aspects and reliability. The difference between these approaches lays in the user experience and the speed of use. The offline version is faster than the online version, because there is no need for requesting any remote ticket data from an other system. But the user has to choose the ticket explicit prior the validation. Chaumette et al. conclude that the online version has a better user experience, since there is no need to pick a ticket explicit for redemption. They also state that the

[1]The secure element is a module, which can host data and applications secure by means of tamper and forgery resistant. In a phone the secure element is typically the SIM (or UICC) card

online version may be more error prone, since tickets can only be checked if a data connection is available.

3.1.2.5 Usability Aspects of a Mobile Ticketing System

Ghíron, Sposato, Medaglia, and Moroni developed in their work [26] a prototype of a NFC based ticketing system and evaluated the usability of it. The main result of their research was that the four properties efficiency, feeling of safety, ease of use and automation are crucial for a good usability. They describe efficiency as the "rate between the achieved results and the effort spent to reach them" [26]. They state that for a sense of safety, the user must have a feeling of control over the interface. This can be achieved by keeping the interface familiar, simple and consistent. To keep the system easy to use, the actions must be simple and direct. Also, the feedback from the system must be understandable and should be presented in a visual way. To decrease the users cognitive effort when interacting with the system, actions should be performed automatically as far as possible.

3.1.2.6 Business Model for a NFC based Mobile Ticketing Service

Juntunen, Luukkainen, and Tuunainen utilized in their work [46] the generic STOF (service, technology, organization and finance) model of Bouwman, De Vos, and Haaker [11] in the context of mobile ticketing with NFC.
For the service domain, Juntunen et al. describe that mobile ticketing with NFC offers many advantages for users. Usually, a user always has their mobile with them and therefore also the stored tickets on the mobile. It is unlikely that users forget their mobile, because they are used to carrying it with them, certainly more than a ticket which they only need for special occasions. Tickets can be bought cashless, anytime and anywhere on the mobile, which makes the purchase independent of a ticket shop or vending machine. The user is not subject to waiting time in a queue, which makes the purchase faster. Service providers benefit from mobile tickets, since they can offer value added services along with their products.
For the technology domain, Juntunen et al. state that a big issue is the availability of NFC on mobiles.
In the organization domain Juntunen et al. state that a successful NFC based ticketing system requires many cooperating participants. These participants are e.g. mobile network operators, service providers and trusted service managers. They state that for many previous systems the mature drawback was that only one of each organization participated. So, previous systems weren't utilized for many domains.
The mobile network operator is responsible for data transfer in his network. Also, he may issue and manage the SIM (or UICC) card. Space on the SIM card may be rented either by the network operator or by a commissioned trusted service manager to service providers. Service providers use the rented space for their applications. Public transport operators and other service offering organizations are crucial participants. Benefits of mobile ticketing for them are reduced cash handling and a reduced effort for ticket sale and distribution, since this is done cashless on the user's mobile. Financial institutions may also profit as service providers, since they can handle the cashless payment and offer additional services like mobile banking.

For the last domain, the finance domain, Juntunen et al. describe that network operators may have a possible revenue stream by renting the space on the SIM card to service providers.
Juntunen et al. conclude that mobile ticketing is a promising usage scenario of NFC. It has many advantages for all participants compared to traditional ticketing. Especially mobile ticketing in public transport is considered a main driving factor for the spread of NFC.

3.1.3 Services related to Mobile Ticketing

Mobile ticketing is a cross cutting domain, where other services and technologies may be utilized. This section introduces location based services and mobile payment services, which enhance mobile ticketing functionality.

3.1.3.1 Location Based Services

Location based services can be used by mobile ticketing since the user's location may be utilized to trigger events in the ticketing system. So, the location information can be used to determine which ticket a user likely wants to redeem, e.g. when the user is at the airport he may use his previously bought flight ticket. Another possible field of application is offering the user tickets which may be redeemed next to him, e.g. the user is at the entrance to a zoo, the zoo tickets can be offered to him.
Hirsch, Kemp, and Ilkka define location based services as services, which depend on and are enhanced by the user's location [30]. The user's position is identified by the mobile. Dhar and Varshney state in their article [18] that different technologies like triangulation, global positioning system (GPS) and cell-ID are utilized for this purpose. For identifying outdoor positions, usually satellite based systems are utilized. Deng, Zou, Huang, Chen, and pei Yu describe in their report [17] the currently existing or planned satellite based systems GPS (U.S.A.), Galileo (Europe), Glonass (Russia) and Compass (China). Mautz gives in [55] an overview of other positioning systems, which can be used if there is no satellite connection available. The location information is derived by using different technologies, like sensors, radio-frequency identification (RFID), Bluetooth or wireless local area networks (WLANs).

3.1.3.2 Mobile Payment

Mobile ticketing involves billing of the acquired mobile tickets. For the billing process, mobile payment services can be utilized. Mobile payment services enable billing and transaction processing on the mobile. The following section introduces mobile payment services.
Mallat, Rossi, and Tuunainen state in their article [53] that mobile payments are commonly categorized into micro-payments and macro-payments. Micro-payments involve small payments, usually up to 10 or 15 USD or EUR, depending on the payment provider. Macro-payments are payments, which exceed the value of micro-payments. Moreover, Mallat et al. describe that micro- and macro-payments are furthermore divided into remote and proximity payments. Remote payments are done remotely over a network and proximity payments are done at the point of sale. Proximity payments at the point of sale can be either manned or unmanned.
Mallat et al. bring out that ticketing is already a successful application of mobile payment and

can be either a micro- or macro-payment and either proximity or remote, depending on the ticket. As Mallat et al. state, the billing in a mobile payment system can be done in different ways and present the following ones:

- Monthly Mobile Phone Bill: The payments are added to the network operators bill.
- Credit Card Bill: Payments are billed through the customers credit card.
- Direct Debit: Payments are debited directly from the customer's bank account.
- Separate Account: Payments are billed via a customer's separate account.

Customer acceptance of mobile payment is crucial for mobile ticketing. Schierz, Schilke, and Wirtz analyzed in their work [63] consumer acceptance of mobile payment services. They state that the drivers for mobile payment are perceived compatibility, individual mobility and subjective norm:

- For perceived compatibility, users must find the process of mobile payment reconcilable with their experiences, behavioral patterns and values. According to their work, perceived compatibility has the greatest impact factor on mobile payment.
- Individual mobility describes the grade of mobile lifestyle a user lives.
- Subjective norm is the desirability degree of mobile payment perceived by the social environment the user is in.

Mobile payment is an important driving factor for m-commerce, which requires secure and reliable systems.
Hu, Sueng, Liao, and Ho proposed in their report [31] a mobile payment solution based on Android using a three factor authentication. They state that mobile payment with a three factor authentication is effective and necessary to minimize the risk of unauthorized transactions. The three authentication factors consist of something the payer knows (e.g. a PIN or password), something the payer has (e.g. a SIM card) and something the payer characterizes (e.g. a biometric feature). The payment itself is done via an ad-hoc WLAN, secured with a secure sockets layer (SSL)-based hyper text transfer protocol secure (HTTPS) protocol.
Manvi, Bhajantri, and Vijayakumar developed in [54] a secure mobile payment system for a wireless environment. This system consists of a Java Enterprise Edition (JavaEE) based payment server and Java Micro Edition (JavaME) based payment clients. The payment clients connect via Bluetooth to the payment server for authorizing a transaction. The transaction itself is authorized by an account number and an account password.
Gao, Kulkarni, Ranavat, Chang, and Mei built in [25] a mobile payment system based on two dimensional barcodes. Hereby, each user owns a digital wallet, which is based on a mobile payment account. The payment accounts are located on a payment server.
Gao et al. state that this system has the advantage that goods are identified by two dimensional barcodes. So, the required user interaction is reduced, since the user gets additional information about the product and the possibility to purchase it by scanning the two dimensional barcode. The system also supports product and customer verification after the purchase.

3.2 Related Work of Mobile Ticketing

This section introduces existing mobile ticketing systems. Mobile ticketing is a young and emerging way of ticketing that is backed by new technologies. Yet, there are only a few existing systems that are predominantly developed specifically for a single company and for a specific domain. Systems, which are well documented in scientific studies have de facto hardly any relevance in practice and those who have aren't covered in publicly available documents. Also the implementations behind these systems aren't available to the public.

3.2.1 Wingbonus - NFC based mobile coupon system

Sánchez-Silos, Velasco-Arjona, Ruiz, and Gómez-Nieto present in their work [62] the NFC based mobile coupon system "Wingbonus". Wingbonus offers the possibility to acquire the ID of a coupon through several ways, e.g. from a Bluetooth server, from a NFC tag or from other users. Figure 3.1 shows the architecture of Wingbonus as presented in [62].

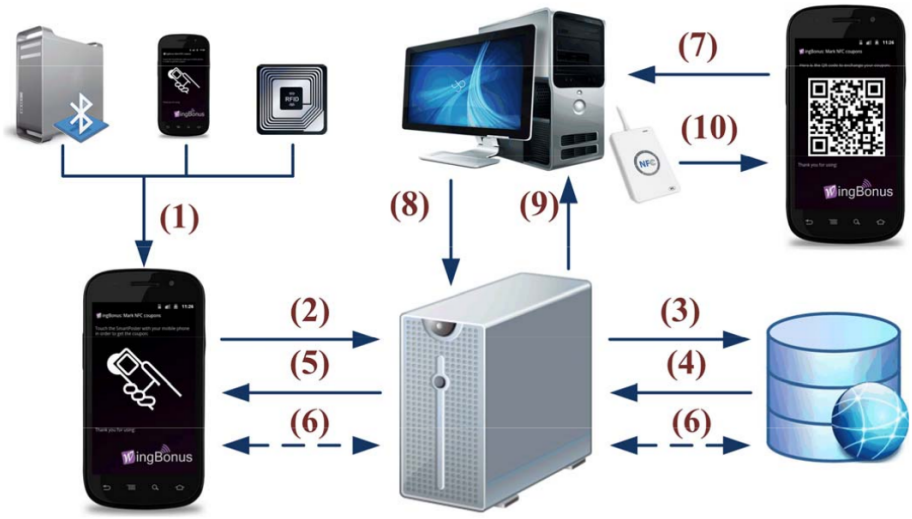

Figure 3.1: Wingbonus Architecture [62]

After the acquisition (1), the ID is sent to the Wingbonus server (2). The server validates the given ID, retrieves the related coupon (3, 4) and issues it to the user (5). Alternatively, the user may also acquire a coupon directly from the Wingbonus service application, where he chooses the desired coupon (6).
The coupon is redeemed by sending it to the merchant's PC using NFC (7), which validates the coupon against the Wingbonus server (8, 9). If the coupon is valid, the merchant's PC sends a success message back to the users smartphone using NFC (10).
Sánchez-Silos et al. have developed a custom protocol for the synchronization process between the database server and the mobile, instead of using an existing standardized communication

protocol. Standardized communication protocols are designed for general purpose what implies a general communication overhead and usually less possibilities of protocol customization. Sánchez-Silos et al. state that the custom protocol allows communication with a smaller overhead and gives more control over security and data integrity.
For security reasons, e.g. when the mobile is lost or stolen, Wingbonus encrypts the coupons when they are saved on the mobile using the Blowfish algorithm [64]. So the coupons are protected against theft since they can only be decrypted with the according key.
Wingbonus offers value added services for merchants and users. Users may collect loyalty points when using coupons and change them e.g. for discounts. Merchants can gather information about their customers for market research.

3.2.2 VDV Core Application - Integration of NFC

Widmann et al. describe in [70] the integration of NFC ticketing in the existing core application of the Association of German Transport Companies (VDV). Widmann et al. selected a subset of use cases from the VDV core application for implementation, which are described in section 3.1.2.3. The implementation on the server side was developed with C# with ASP.NET. The Microsoft SQL server was used as database. For the user side, a smart card application was implemented as Java Card Applet [40]. Figure 3.2 shows the architecture of the implementation as presented in [70].
In the system, different functional entities interact with each other. The user is represented by an user medium (e.g. the mobile phone), which hosts the smart card application. The product retailer sells tickets to the user. The product owner (who is a travel authority) defines the offered tickets. Their services are provided by the service operators. They accept and check prior issued tickets. The application owner manages the system and administrates the blacklists.
A user may buy a ticket from the product retailer's system by declaring his desired route. The product retailer's system knows the available routes from the product owner's system and creates a unique ticket for the customer. The ticket is issued to the customer's mobile and is stored in the secure element. The product owner is informed that a ticket was issued.
The service provider reads all available tickets from the user and checks if a valid ticket exists. The user's application ID and the ticket IDs are also matched against the blacklist, which allows refusing blacklisted tickets. Users and product owners may add or remove tickets to or from the blacklist. The components of the system communicate with each other via web services.
Widmann et al. state that there is a shortage of user devices supporting the NFC card emulation mode[2]. Regardless to this fact and other challenges, Widmann et al. successfully integrated NFC ticketing in the existing VDV core application.

3.2.3 Mobile Ticketing System Based on Personal Trusted Devices

Chen, Chen, and Jan describe in their work [15] a scheme for mobile ticketing, which is based on personal trusted devices. The following participants take part in their ticketing scheme.

[2] In the card emulation mode a NFC devices acts as an passive communication partner.

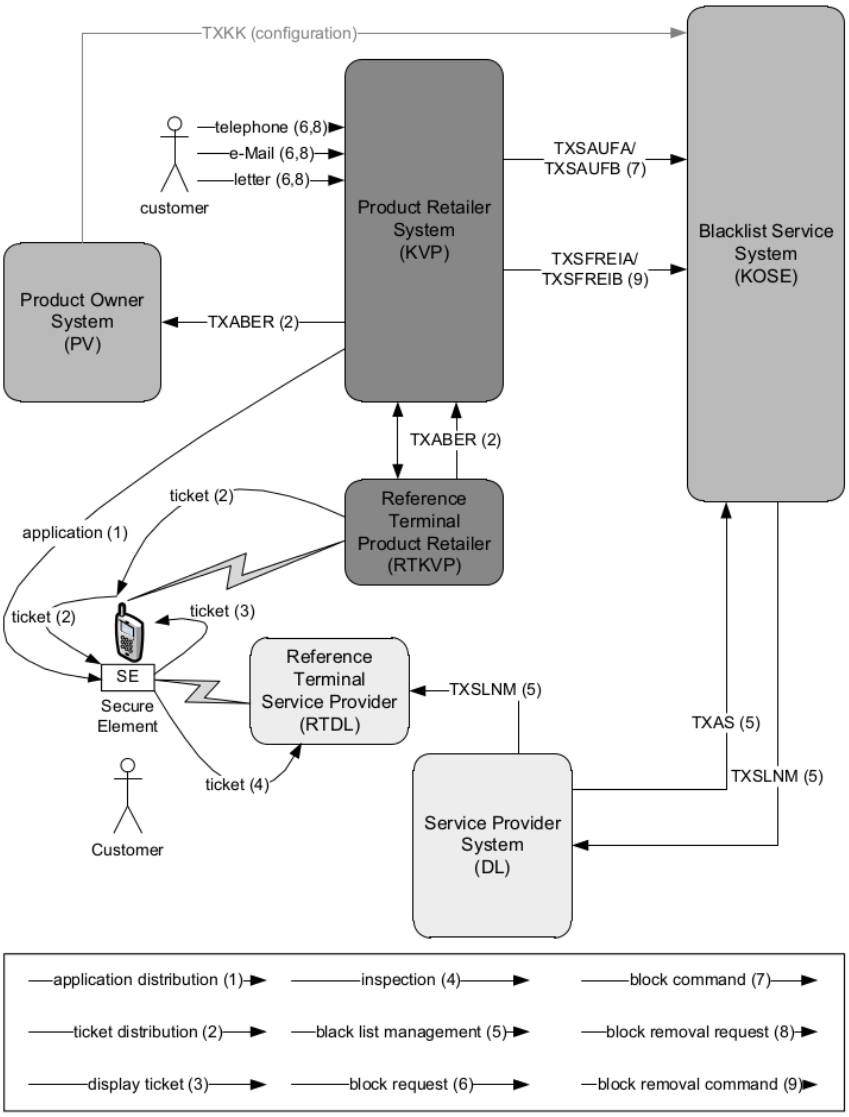

Figure 3.2: VDV NFC Integration Architecture [70]

- Subscriber: A person, who buys tickets and has a personal trusted device.
- Observer: An agent for the subscriber.
- Mobile Network Service Provider (MNSP): The provider for the data connection and billing manager.

- Ticket Service Provider: Issues mobile tickets and cooperates with the mobile network service provider to offer the ticketing service.

- Verifier: A service agent for the ticket service provider, who verifies the subscriber's ticket. Additionally, the verifier offers the services to subscribers.

The proposed ticketing scheme by Chen et al. uses a public key infrastructure. The scheme satisfies the following requirements:

- Fairness: A participant can't get an advantage over another correctly behaving participant.

- Non-repudiation: A participant can't deny a step or action, which has taken place.

- Anonymity: The subscriber is anonymous to the ticket issuer and ticket verifier.

- No forging: A mobile ticket can't be forged.

- Efficiency Ticket Verification: Ticket verification is done fast.

- Simplicity: All operations are designed as simple as possible.

- Practicability: The mobile ticketing scheme is applicable.

- Obviate the Embezzlement: A mobile ticket can't be used illegally.

Figure 3.3 shows the architecture of the system as presented in [15]. The system consists of three phases, the ticket request phase (phase 1), the ticket issue phase (phase 2) and the ticket verification phase (phase 3).
In phase 1, the subscriber requests a ticket from the MNSP (1). The MNSP generates a unique transaction ID and adds it to the request. Then he signs the request and forwards it to the ticket service provider (2).
In phase 2, the ticket service provider verifies the signature of the MNSP, generates a mobile ticket and sends it to the observer (3). The observer verifies the MNSP's and ticket service provider's signatures, generates a response signature and sends it back to the ticket service provider (4). Then, the observer generates a hidden mobile ticket for the subscriber and forwards it along with the response signature to the MNSP (5). The MNSP checks the response signature and forwards the hidden mobile ticket to the subscriber's mobile device (6).
In Phase 3, the subscriber's ticket is checked by the verifier (7). Used tickets are sent to the ticket service provider (8).
This system offers the possibility of offline validation, since the tickets are signed and can therefore be validated by a pre-issued public key.

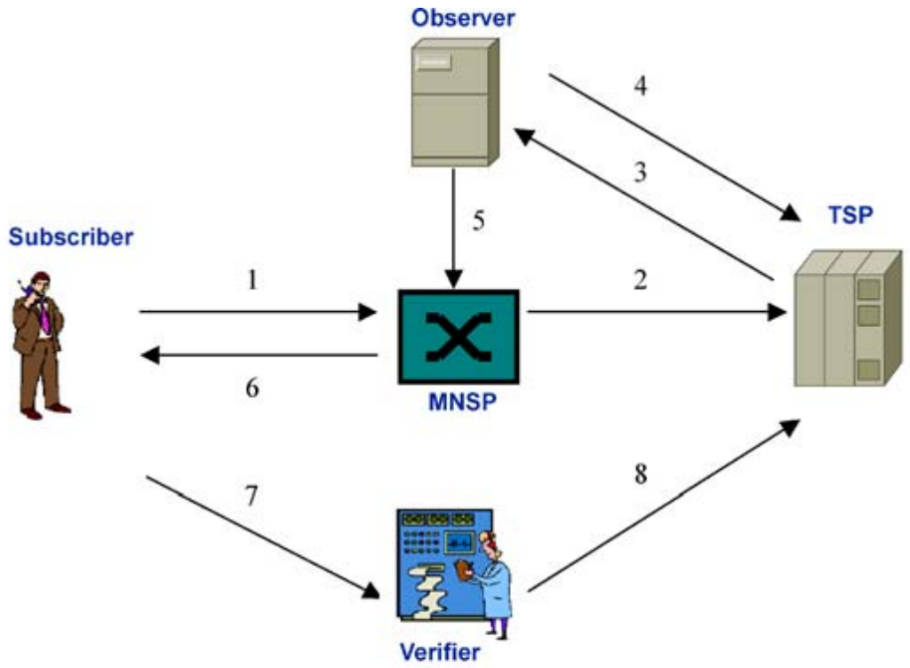

Figure 3.3: Mobile Ticketing System Based on Personal Trusted Devices architecture [15]

3.2.4 Tapango

Neefs et al. present in their work [58] the offline ticketing system Tapango. For Tapango, every user has an electronic wallet (e-wallet), which holds his event tickets. This e-wallet can be hosted on the users mobile and is linked to the users corresponding account.
Figure 3.4 shows the architecture of Tapango as presented in [58]. The architecture consists of the following parts, which communicate with each other via simple object access protocol (SOAP) webservices.

- Web Interface: Users can pre-purchase tickets and vouchers from the server via the web interface.
- Terminals: Tickets and vouchers are redeemed via terminals. The terminal reads the ticket from the e-wallet and sends it back to the server after use.
- Syncpoint: The syncpoint is used to receive pre-purchased tickets and vouchers from the server and putting them into the e-wallet. For this purpose, the unique e-wallet ID is read by the syncpoint and the according tickets and vouchers are retrieved from the server and transfered via NFC to the e-wallet.

- Cashier: Users can purchase tickets and vouchers from a cashier. These bought items are transfered to the e-wallet.
- Server: The server stores the pre-purchased tickets and vouchers until they are obtained by the user.

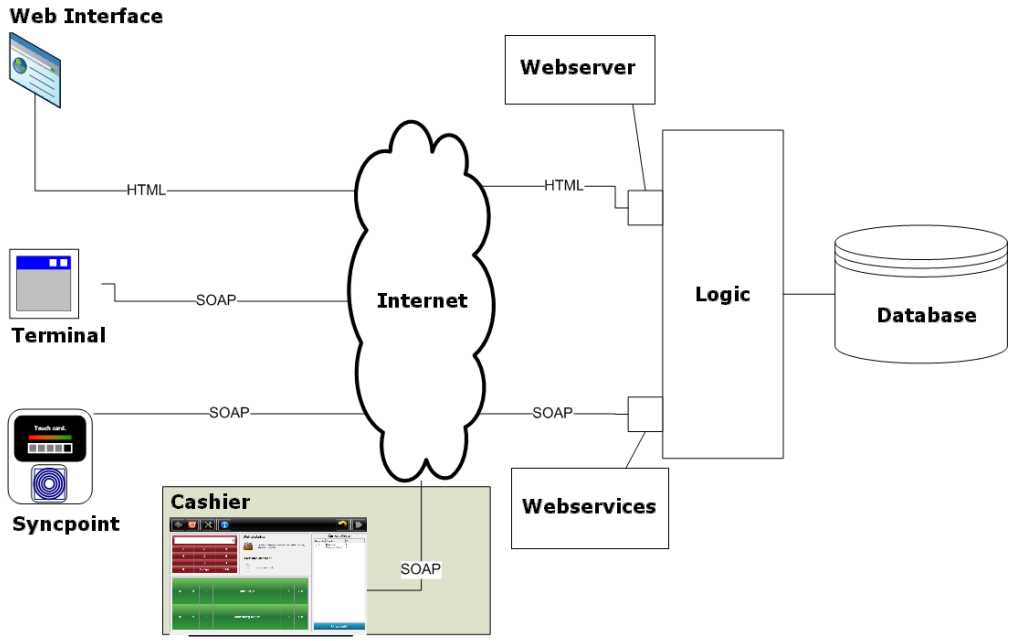

Figure 3.4: Tapango Architecture [58]

Tapango separates transactions in two groups, those who happen before the event and those who happen at the event.
Before the event, a user can register himself and buy tickets and vouchers via the web interface. If the user owns a NFC card reader, he can transfer the tickets and vouchers from the server to his e-wallet. If not he can obtain them via a syncpoint.
At the event site, the user can obtain an e-wallet, which is issued and linked to his account by an employee. He can receive his tickets via syncpoints. It is also possible to purchase tickets and vouchers at the event site from a cashier. To redeem a ticket, the user holds his mobile to a terminal, which checks if an appropriate ticket is present.

3.2.5 Offline and Online Architectures

Chaumette et al. presented in their work [14] an online and an offline approach for a mobile ticketing system. The result of the comparison between them is given in section 3.1.2.4. The following illustrates the architecture of these two systems.
Figure 3.5 shows the architecture of the offline system approach as presented in [14].

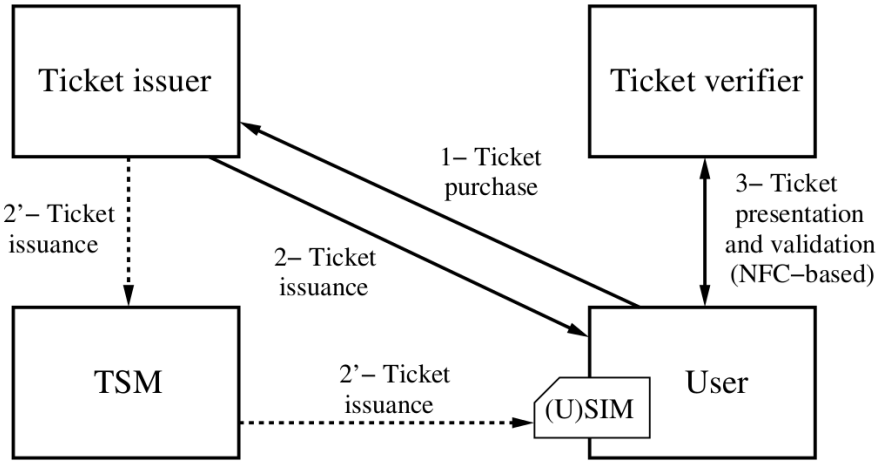

Figure 3.5: Offline System Architecture [14]

In this scenario the user purchases a ticket from the ticket issuer (1). The ticket issuer issues the ticket via the trusted service manager (TSM) to the secure element in the users mobile (2, 2'). The TSM is the only entity that is able to write data on the secure element. To redeem the ticket, it is read from the secure element by the ticket verifier via NFC. The ticket is validated without usage of any external structure.
Figure 3.6 shows the architecture of the online system approach as presented in [14].
In contrast to the offline approach, the user only possesses his phone which contains a secure element holding a static identifier. This secure element can be either the SIM card or any other element that has a unique and forgery safe identifier. The user purchases a ticket from the ticket issuer with his phone's secure element's ID. The ticket is created in the ticket issuers backend system (1). The ticket verifier reads the secure element's ID via NFC (2) and checks if a valid ticket with that ID exists in the ticket issuer's backend system (3). The received acknowledgment from the ticket issuer's backend system is sent to the user's phone via NFC (4).

3.2.6 Secure eTickets Based on QR-codes with User-Encrypted Content

Conde-Lagoa, Costa-Montenegro, González-Castaño, and Gil-Castiñeira present in their article [16] a secure electronic ticketing system based on quick response codes (QR-codes) with user encrypted content. For security reasons, the QR-code content is encrypted to avert ticket thievery. Figure 3.7 shows the design of this ticketing system.
In step 1, the user buys a ticket via a TLS encrypted connection. The payment is done via a mobile payment system. The issued ticket is a unique ID, which the user receives in step 2. This ID grants the user access to the event or service.
Immediately after the ticket purchase but before the ticket is stored, the user choses a cipher key

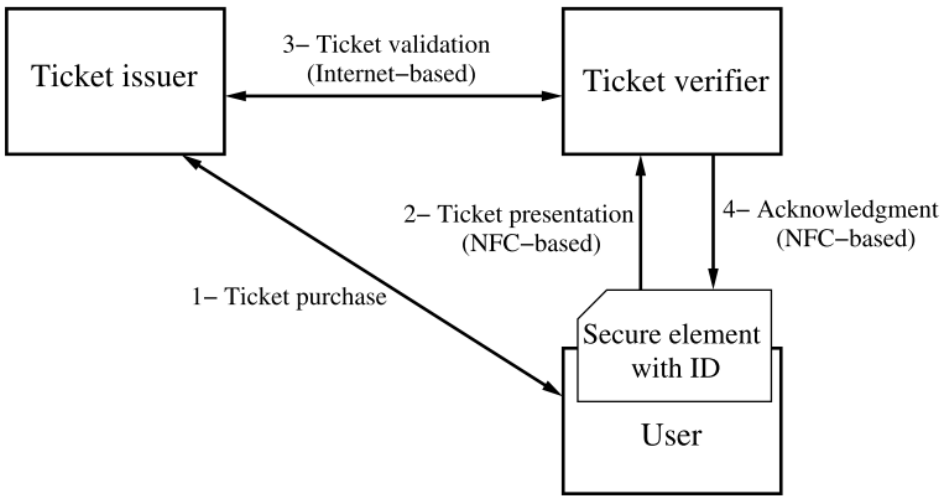

Figure 3.6: Online System Architecture [14]

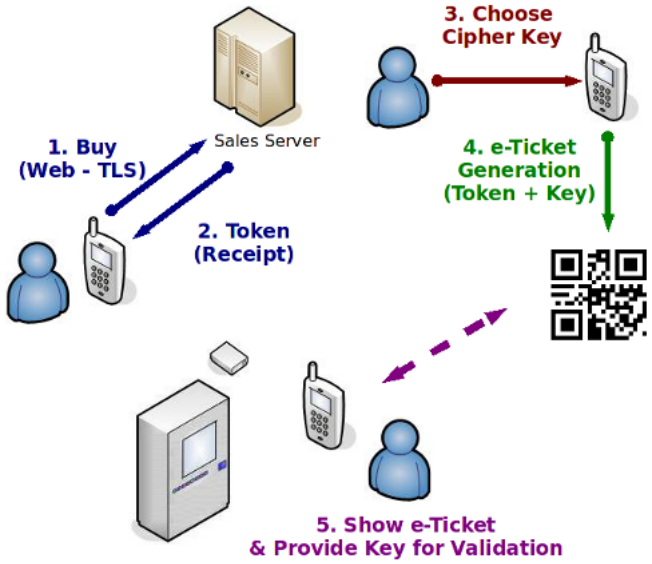

Figure 3.7: eTicket System based on QR-codes with User Encrypted Content [16]

in step 3. In step 4 the ticket becomes encrypted with the chosen cipher. So, the ticket becomes useless without the cipher, since it can't be decrypted without it.

In step 5 the ticket is validated by showing the QR-code and entering the cipher key to decrypt the ticket. Only if the ticket is successfully decrypted with the given cipher and only if it is valid, the user has access to the event or service.

3.2.7 Apple Passbook

Apple Passbook is an application for iOS for keeping all different kinds of coupons and tickets, which are called passes [7].
Passbook itself is not a ticketing system. The pass validation is done by the merchant's ticketing system. A pass contains a two dimensional barcode (2D barcode), which is scanned for redemption. Supported are the 2D barcodes QR, Aztec and PDF417. Passes can be added through Passbook-enabled apps, mails, messages or the web browser. A pass can be selected by either picking it by hand from the Passbook application, or automatically by utilizing the current time and/or the users current location. For this purpose the pass must have information about the location and/or time of redemption.
A pass contains additional information like merchant's contact information, discount information, a gate and departure time of a flight, a gift card balance and others. This information can be updated through Apple's push notification service. So the user is kept up to date with information concerning his tickets. The user can be informed when e.g. his gift card balance was altered or the departure gate of his flight has changed.
Passes can be stored in iCloud. Passes are synchronized across different devices and can be recovered if the user's mobile is lost or damaged.
A pass consists of different files, which are arranged in a package, called the pass package. The central file which defines the pass is pass.json. It is a java script object notation (JSON) [19] file which identifies the pass and contains relevant information (location and time of redemption, merchant contact info, information about the pass, ...). The package may also contain images, which are shown on the pass, and localization data.
Before distribution, the pass is cryptographically signed and compressed which allows to verify the issuer.

3.3 Capable Technologies for Building a Mobile Ticketing Systems

The previous section gives an introduction to existing mobile ticketing system. It can be seen that many different technologies exist for building a state of the art mobile ticketing system. This section introduces technologies and platforms that may be utilized for building a state of the art mobile ticketing system. These technologies can be categorized in data transfer related technologies, crypto related technologies and widely distributed mobile operating systems.

3.3.1 Data Transfer related Technologies

This section describes technologies, which are related to data transport and transfer.

3.3.1.1 Quick Response Code

The quick response code (QR-code) is a two dimensional barcode, which encodes information in a two dimensional picture. It is similar to the bar code, but uses two dimensions for encoding information. The QR-code was developed in 1994 by the Japanese automotive industry Denso Wave, which is a subsidiary of Toyota. The code was designed for automatic identification of components by cameras in a short time. Due to the high practicability of this code it emerged to other usage scenarios and became standardized by AIM, JIS and ISO/IEC. The ISO/IEC standard that defines the QR-code, is ISO/IEC 18004:2006 [41]. Other standards of two dimensional barcodes exist, like the Aztec Code, the Data Matrix and others.
QR-codes support numeric data, alphanumeric data, byte data and Kanji[3] characters. A QR-code may have 21x21 modules (version 1) and up to 177x177 modules (version 40), with an increase of four modules per side per version. The maximum data, which can be contained are either 7089 characters when using numeric data, 4296 characters when using alphanumeric characters, 2953 characters when using byte data and 1817 characters when using Kanji data. The standard also defines four levels of error correction, L with 7%, M with 15%, Q with 25% and H with 30%.
The QR-code is a quadratic picture on a white background with black squares. Figure 3.8 shows the basic scheme of a QR-code in version 7 [41]. The scheme gives information about the used QR-code version and orientation points for normalizing the picture after scan. Figure 3.9 shows a complete example of a QR-code.
The QR-code can be read orientation independent, since it is normalized by the included scheme.
Data may be encoded in up to 16 QR-codes, which can be read in an arbitrary order. After reading the last one, the data is reconstructed.
The QR-code is a very important technology for mobile ticketing, since a ticket ID or a small ticket can be quickly transferred with it. The QR-code bridges the gap between an analog and a digital ticket without the need for any user input.
This method of transference has no overhead nor does it need a special configuration. A wide range of devices exist (including legacy ones) that are capable of showing a QR-code. Many existing ticketing systems utilize QR-codes.

3.3.1.2 Near Field Communication

NFC is a technology for transferring data wireless between two opposite devices. NFC is defined in the standards ISO/IEC 18092:2013 [43] / ECMA-340 [20] and ISO/IEC 21481:2012 [42] / ECMA-352 [21]. A main driver behind the standardization of NFC is the NFC Forum Inc. The NFC Forum Inc. was formed in 2004 and has over 170 members.
According to the standards, NFC operates at 13.56MHz and supports a transfer rate up to

[3] Kanji characters are Chinese characters

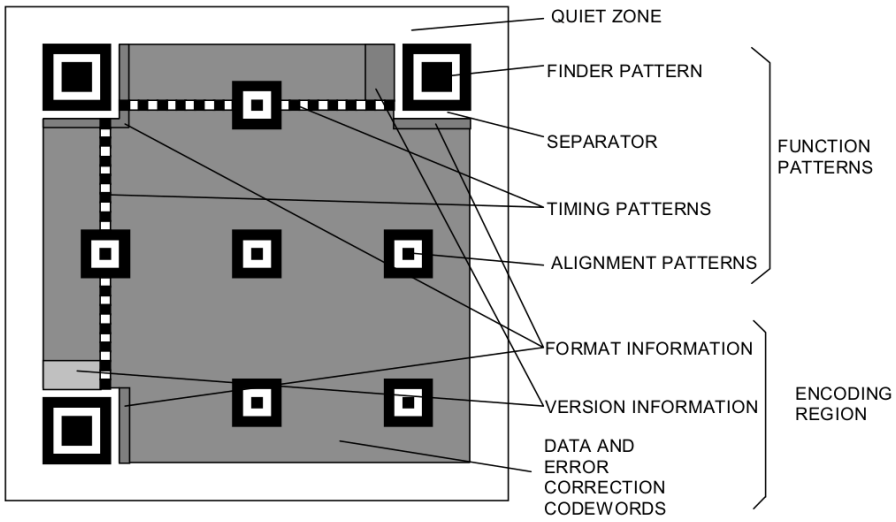

Figure 3.8: Structure of a QR-code 2005 Version 7 Symbol [41]

Figure 3.9: Example of a QR-code

424kbps. The defined maximum distance for communication is 10cm. NFC can either operate between two powered devices or between a powered device and an unpowered NFC tag. Between two powered devices a two way communication is possible. Between a powered device and an unpowered tag, the powered device induces electromagnetic energy in the unpowered tag and reads the contained date from the unpowered tag.
NFC does not have a pairing procedure as Bluetooth or WiFi have. So, NFC has no overhead to initiate a communication since it is started by bringing the communication partners close to-

gether. But through the omitting of the pairing procedure, the possible fields of application are lesser then the ones of Bluetooth or WiFi.
Want describes in his article [69] that fields of application of NFC are connection handover, mobile ticketing and mobile payment. For mobile connection handover NFC is used due to its small overhead to initiate other types of connections. An example for this purpose is Android Beam, which utilizes NFC to initiate a Bluetooth connection between two devices [29].
The NFC Forum Inc. defined the NFC data exchange format (NDEF) standard [34]. The standard defines how data is exchanged between communication partners using NFC. The communication is message based were each message is composed of an arbitrary number of records. The NFC forum defines the records in the record type definition standard (RTD) [35].
The white paper Essentials for Successful NFC Mobile Ecosystems of the NFC Forum Inc. [36] presents ways how to build successful NFC mobile services. The NFC Forum Inc. states that NFC has a wide field of application. Examples are mobile payment, data exchange, authentication at access controls, mobile ticketing and many others. The NFC Forum Inc. believes that the key factors for NFC services being a success are:

- Support for a large variety of existing and future business models in the NFC ecosystem.
- That the responsibility of each player in the NFC ecosystem is clearly defined in a specific model.

NFC is an emerging technology with a large economic potential. Yet it lacks a high market diffusion, but there are already some mobiles which support NFC. Want believes that support for NFC in Apple's iPhone would be the kickstart for this technology [69]. At the time of writing most available NFC services are isolated applications what should be subject of change in the future.

3.3.1.3 Bluetooth

Bluetooth is a technology for transferring data wireless over short to medium distances. It is an industry standard [33], which is developed by the Bluetooth Special Interest Group Inc. (Bluetooth SIG). The current Bluetooth version is 4.1, which was released in 2013. The main use cases for Bluetooth is data communication between devices over a short to medium distance. Examples are data transfer between mobile phones or transfer of input data from a keyboard to a computer.
According to the standard, Bluetooth operates in the frequency band between 2400MHz and 2483.5MHz. It supports data rates of 1 Mbps for Basic Rate and up to 3 Mbps for Enhanced Data Rate.
The Bluetooth specification defines three different classes of radios with different power consumption and range. Class 1 uses 100mW of power and has a range of 100m. Class 2 uses 2.5mW of power and has a range of 10m. Class 3 uses 1mW of power and has a range of 1m.
Bluetooth defines different Bluetooth profiles for communication. Such a profile defines a concrete application of Bluetooth and defines how the communication is performed. To use a specific Bluetooth profile for communication, all interacting devices must support it. Bluetooth

profiles are e.g. OBEX for object exchange between devices, and PAN for personal area networking.
In the Bluetooth specification version 4, the Bluetooth low energy technology was introduced which enables Bluetooth devices to operate with a very low consumption of power. Such devices can be built very small and can run for months or even years with a single power source. Apple Inc. utilizes Bluetooth low energy for low-cost and low-powered transmitters called iBeacons [6]. A smart phone may perform a defined action, when it comes into proximity of such a beacon. Other promising fields of application among others of Bluetooth low energy are healthcare, indoor positioning and mobile payment.

3.3.1.4 JavaScript Object Notation (JSON)

The JavaScript Object Notation (JSON) is a format for data interchange. JSON is described and defined in the standards RFC 7159 [12] and ECMA-404 [19]. According to the ECMA-404 [19] JSON is lightweight, text-based and language independent.

3.3.2 Crypto Technologies for Mobile Tickets

This section introduces the for mobile ticketing important crypto technology OpenPGP along with the crypto library Bouncy Castle [52].
OpenPGP defines a message format by utilizing public-key and symmetric cryptography and is described in the RFC 4880 [13]. OpenPGP was derived from PGP, created by Phil Zimmermann. An implementation of OpenPGP is GnuPG.
According to RFC 4880 [13], the core functions of OpenPGP are

- Data encryption
- Data decryption
- Data signing
- Key management

In cryptography, unencrypted text is denominated as plaintext and encrypted text as cyphertext. Symmetric cryptography algorithms are using the same key for encrypting plaintext to cyphertext and for decrypting cyphertext to plaintext. Symmetric cryptography requires that the symmetric key is only shared between authorized entities, since anyone could decrypt cyphertext with it.
Public-key cryptography algorithms utilize a key pair, which consists of a public and a private key. The public key is made public and anyone may use it either to verify a signature or to encrypt plaintext. The private key is used to generate a signature and to decrypt cypher text. Public-key cryptography requires that the private key is kept private and is shared with nobody else.
The OpenPGP standard can be utilized for mobile ticketing to sign and encrypt tickets. With a signature, the authenticity of the ticket can be verified. With encryption, the ticket or only parts of it can be kept confidential.

- Data Encryption and Decryption: With data encryption and decryption, confidentiality is achieved. According to the OpenPGP standard as described in RFC 4880 [13], the plaintext is encrypted by utilizing symmetric and public-key cryptography. For each encryption (called session), an unique symmetric key (called session key) is generated. The plaintext is encrypted using the session key.
  Each cyphertext receiver must have a key pair, consisting of a public and a private key. The symmetric session key is encrypted for each receiver with the receivers public key. The encrypted symmetric key is attached to the cypher text.
  Each receiver first decrypts the encrypted symmetric key using his private key. With the decrypted symmetric key, the cyphertext is decrypted in plaintext.

- Data Signation: With data signation, authentication, integrity and non-reputation is achieved. According to the OpenPGP standard as described in RFC 4880 [13], a hash code of the data to be signed is generated. The signature is created with the generated hash code and the private key. The signature is attached to the data.
  A receiver generates a new hash code of the received data. The newly generated hash code is verified by using the attached signature. If the verification is successful, the data wasn't altered and was sent by the sender who owns the according private key.

Bouncy Castle is the de facto standard crypto library for Java and C# [52], which offers encryption APIs. As stated on their website, the Bouncy Castle APIs consist among others of:

- Lightweight cryptographic API for Java.
- A provider for the Java Cryptography Extension and the Java Cryptography Architecture.
- Generators/Processors for OpenPGP (RFC 4880 [13]).

Bouncy Castle is developed and maintained by the Australian charity Legion of the Bouncy Castle Inc.

3.3.3 Mobile Operating Systems

This section gives an overview for mobile operating systems, which are capable for mobile ticketing. This overview considers only the most important platforms and is not exhaustive. As described in the sections 3.3.1.1 and 3.3.1.2, the QR-code and NFC are key technologies for mobile ticketing.

3.3.3.1 Google Android

Android [27] is a mobile operating system, which is based on the Linux kernel. It is open source but contains proprietary components like device drivers. Android is developed and released by the Open Handset Alliance, which is at the time of of writing an alliance of 87 companies with the goal to develop open standards for mobile devices. The alliance was formed by Google Inc., which is the main member of it.
There exist more then 900 million of active Android powered devices presently at the time of

writing [27]. Google Inc. has an application store for Android, called Google Play, which contains more then 975.000 applications at the time of writing.
At the time of writing, the current Android version is 4.4, named KitKat and has the application level interface (API) 19 [28]. Google Inc. offers a software development kit (SDK) for Android for developing applications, which is based on Java. Android also supports using C and C++ for native development.

3.3.3.2 Apple iOS

Apple iOS is a mobile operating system, which is based on Apple's Mac OS X. It is closed source and therefore can't be customized. iOS is developed by Apple Inc. Apple iOS is used on all Apple's mobile devices, which are iPhone, iPad, iPod touch and Apple TV.
According to Apple Inc. they sold over 700 million of iOS devices until September 2013 [3]. Apple Inc. has an application store for iOS, which contains more then 900.000 applications [4]. The current iOS version is 7 [5]. Apple Inc. offers a SDK for developing applications, which is based on Objective-C along with the Xcode IDE.

3.3.3.3 Windows Phone

Windows Phone is a mobile operating system, which is developed by the Microsoft Corporation [56]. It is closed source and therefore can't be customized. The current version of Windows phone is 8.
The Microsoft Corp. has an application store for Windows Phone [56]. They offer a SDK for developing applications, which can be developed with Visual C#, Visual Basic and Visual C++. Visual Studio .NET is the official supported development IDE.

3.3.3.4 Firefox OS

Firefox OS is a mobile operating system, which is developed by the Mozilla Foundation. Firefox OS is open source and is based on the Linux kernel. The operating system provides a HTML 5 runtime environment along with support of other open web standards [57]. The main components of Firefox OS are [57]

- Gaia: The user interface level.
- Gonk: The lower level operating system. It consists of the Linux kernel and the hardware abstraction layer.
- Gecko: The layout engine. It reads and renders web content such as HTML, CSS and JavaScript.

The Mozilla Foundation offers development tools for creating applications for the Firefox OS. Applications are developed using HTML 5 [57]. Firefox OS applications may be published self or via the Firefox OS marketplace.

3.4 Conclusion Introduction and Related Work regarding Mobile Ticketing

This chapter has introduced mobile ticketing, presented related work of mobile ticketing and stated capable technologies for building a mobile ticketing system.
The introduction to mobile ticketing presented the basics of mobile ticketing and defined the common terminology used in this thesis. Different types of tickets, like the single use ticket or the time dependent ticket, were introduced. The advantages that a user has his mobile tickets usually with him and that he can buy them anytime and anywhere are stated. The use case scenarios for mobile ticketing are presented. The related services mobile payment and location based services were described.
The related work of mobile ticketing describes the result of a research of existing systems. The system details are taken into account in the design of the presented mobile ticketing foundation in this thesis. These systems haven't got many properties in common and their ticket formats are incompatible.
Capable technologies for building a mobile ticketing system were stated. These technologies are needed for the design and implementation of a state of the art mobile ticketing system. The data transfer related technologies QR-code, NFC, Bluetooth and JSON are described. The message format OpenPGP for ticket signation and encryption was introduced. The most important mobile operation systems, which are Android, iOS, Windows Phone and Firefox OS were briefly presented.

CHAPTER 4

Conceptual Design of a Mobile Ticketing System for Education

Section 3.2 gives a brief introduction to existing ticketing systems. These systems are each built for a specific domain and therefore don't have many attributes or features in common. The tickets in these systems have incompatible formats, are handled differently and diverge in their usage.
In the following a conceptual design of a mobile ticketing system for application in education is presented which shall overcome those drawbacks. The presented concept uses open standards, consists of loosely coupled components and may be tailored to different domains and courses. The ticketing functionality is separated from the domain dependent functionality. Through this separation multiple clients (called merchants) and multitenancy are supported. According to this separation the ticketing system is divided into the following parts:

- A domain independent ticketing server (ticket server).
- A management application for the ticketing server (ticket server management application).
- A domain dependent merchant application (merchant application).
- A domain independent ticket validation application (validation application).
- A domain independent mobile client application (mobile client).

In the following the stakeholders, the ticket properties, the validation process, the system overview and a description of each system component is presented.

4.1 Stakeholders

For this mobile ticketing system the merchant, the ticket validator, the ticket server administrator and the ticket customer are identified as stakeholders.

4.1.1 Merchant

The ticket merchant is the entity, who offers services. He is interested in selling his services as easily as possible to his customers. A customer should only have access to a merchant's service, if he has paid for it. The merchant's revenue depends on that what makes it very crucial for him. The merchant is interested in having a detailed report about the degree of capacity utilization of his services. This information may be useful for him in future planning and service adoption.
The advantage for the merchant is that he can utilize a reliable and easy to handle ticketing system. He does not need to take care of the ticketing process himself. He utilizes the provided infrastructure and outsources the whole process.

4.1.2 Ticket Validator

The ticket validator is commissioned by the merchant for verifying and devaluating issued tickets. Since the mobile validation application should handle the verification and devaluation, the ticket validator does not need to know any details about the ticketing process. He only has to assure that the ticket is read by the validation application and that the user has access to the service if he possesses a valid ticket.
His advantage in this system is that he does not need to check tickets by hand and that he immediately knows if a ticket is valid. It would be even possible to run the mobile validation application fully automatically with appropriate access control mechanisms.

4.1.3 Ticket Server Administrator

The ticket server administrator registers and supports merchants in all of their concerns. He handles all management tasks related to the ticketing process. He knows the details of it, solves problems in it and clarifies support requests.
The advantage for the ticket server administrator is that he has an unified interface for handling complex ticketing processes.

4.1.4 Ticket Customer

The ticket customer acquires and uses tickets. Tickets in his possession are stored on his mobile phone. The typical ticket customer isn't very technically adapted. He wants to buy and use tickets without hassle.
His advantages in this system are that he does not have to queue for buying tickets and that he has an unified ticket handling process.

4.2 Ticket properties and validation

Section 3.1.2.1 gives a non exhaustive overview of possible ticket types. They are considered in the system design and extensibility to new types is supported. Figure 4.1 shows the domain model of this ticketing system.

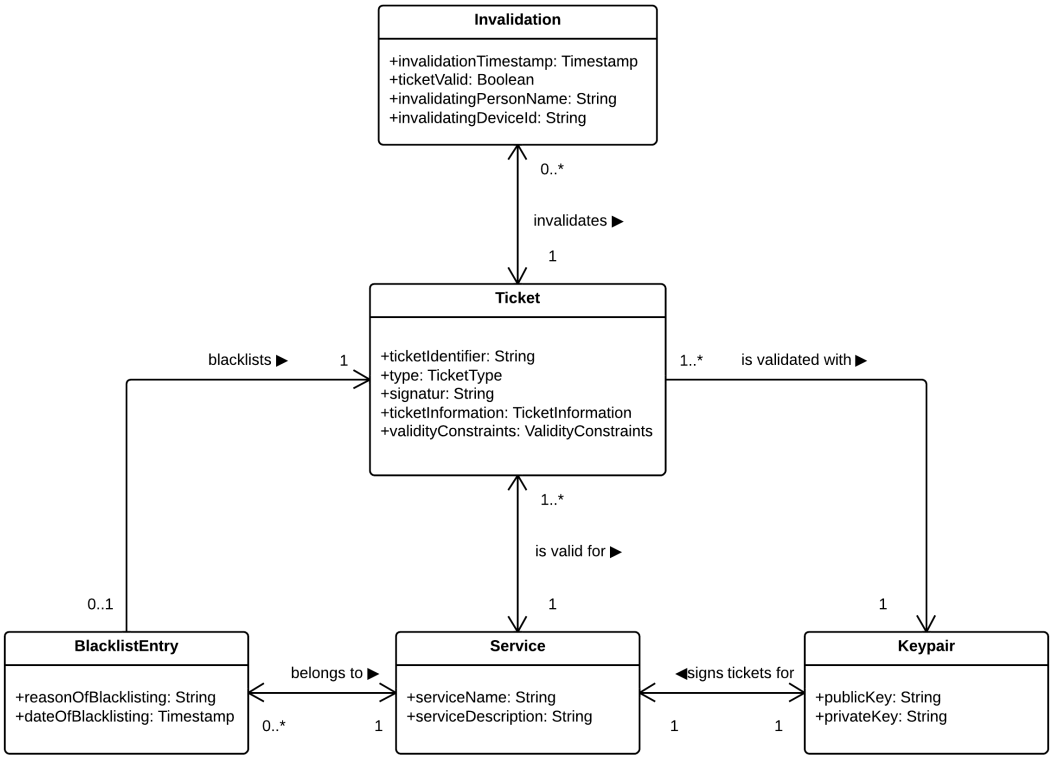

Figure 4.1: Ticket System Domain Model

A ticket consists of a unique identifier, a specific ticket type, validity constraints, domain dependent data and a signature. A ticket belongs to exactly one service, which has a key pair, consisting of a private and a public key.
The ticket identifier identifies the ticket unequivocally. The ticket type and the validity constraints state the circumstances under which the ticket is valid. The domain dependent data is arbitrary and is determined by the merchant and could be anything. The signature is created by the services private key and is generated over the ticket identifier, the ticket type and the validity constraints.
The ticket is created and signed with the private key by the ticket server. The merchant may add arbitrary domain dependent data to the ticket, which serves as an information resource for the customer. Since that data is not signed it won't be considered in the validity check. Afterwards the ticket is issued to the customer.
The validity of a ticket is determined by its type, its signature, its identifier, its validity constraints and prior devaluations. The ticket type specifies the method how the validity is determined. With the signature the authenticity of the ticket can be checked. The ticket identifier is unique and can be added to an optional blacklist, if the usage of the ticket should be prohibited. Also prior

devaluations are considered in the check of ticket validity.
The validation is done in two steps. The first step is offline where the ticket is validated by its signature, by its validity constraints and by the optional ticket identifier blacklist. The signature is validated with the services public key. With the valid signature it is ensured that the ticket wasn't altered or forged by the customer.
The blacklist check is optional and may exclude ticket identifiers, which are not valid for any reason. Then the ticket is validated by it's validity constraints.
If the offline check succeeds the ticket is checked in the second step online by utilizing remote data. Hereby the validity is checked by concerning previous devaluations. If the online check also succeeds, a new devaluation entry is created and the user may access the service afterwards. The service's public key and the ticket blacklist is transferred to the validating device prior to the validation process so that it is available offline.
This two step validation has the advantage that tickets can be checked offline under uncertainty if the remote data with previous devaluations isn't available. The downside in this case is that it would be possible that an already devaluated ticket is accepted again. This could be prevented by keeping track of already devaluated tickets offline along with an synchronization procedure between the devaluating entities.

4.3 System Overview

This section introduces the logical system overview. This system is composed of a ticket server along with its ticket server management application, many different merchant applications, many different mobile clients and many different mobile validation applications. Figure 4.2 gives a schematic overview of the components of this ticketing system with their interacting partners.
The system consists of loosely coupled components to guarantee:

- Interoperability with different applications.
- Expandability to new domains.

The communication between the system components is done via REST webservices, since they are available for many different platforms and ensure a loose coupling.
The ticket server is the central part, which handles the ticketing process. Many different merchant applications from different domains can utilize the ticket server for the ticketing process. A merchant can manage all his tickets with the ticket server.
The ticket server is managed by a ticket server administrator with the ticket server's management application. A ticket server administrator registers and manages merchants and gives them support if needed.
Services and tickets are presented to the user through the merchant's application. The available tickets and their according state are requested from the ticket server by the merchant application. If the user buys a ticket, it is issued to the user from the ticket server through the merchant application.
The mobile client holds all purchased tickets. When the user wants to redeem a ticket to use a service, he chooses the ticket on his mobile client. The client prepares the ticket for transport to

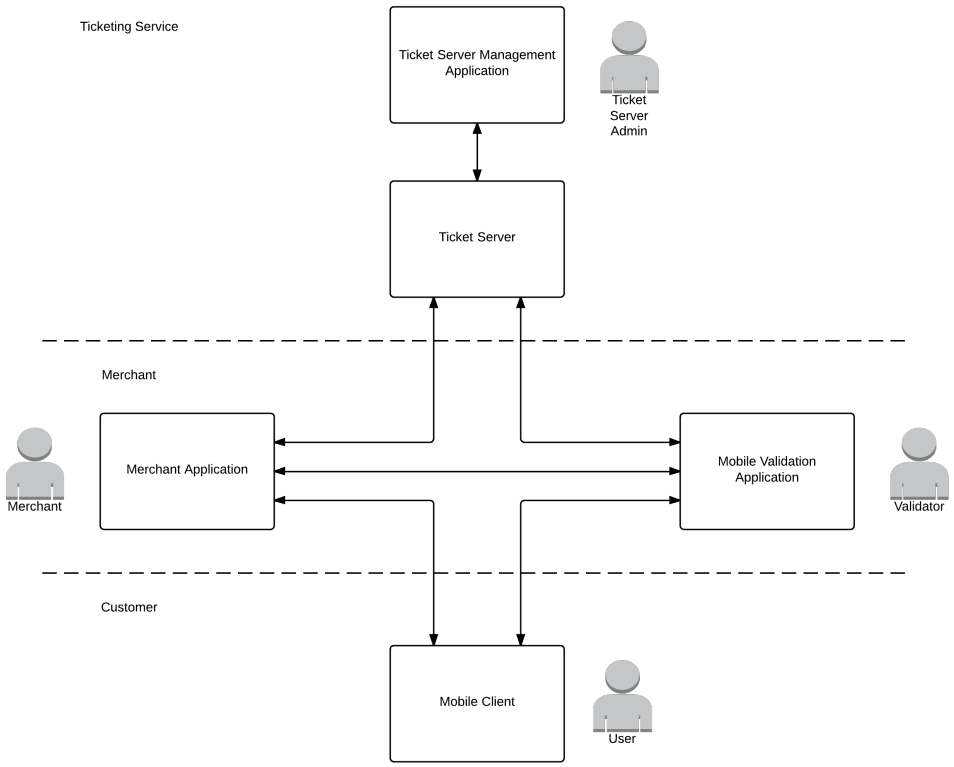

Figure 4.2: Mobile Ticketing System Design

the mobile validation application.
The mobile validation application receives the ticket from the users mobile and validates it by utilizing the ticket server. If the ticket is valid it is devaluated depending on the ticket type. The user may use the service afterwards.

4.3.1 Ticket Purchase

Figure 4.3 shows the sequence for purchasing a ticket. In step 1 the customer requests available services from the merchant application. The merchant application replies the customer with available services in step 2. In step 3 the customer requests available tickets for a desired service. Since the tickets are managed by the ticket server, the merchant application forwards the request to the ticket server in step 4. The ticket server replies the merchant application with all tickets for the requested service with their according state in step 5. The merchant application preprocesses the tickets with their states and forwards them to to the customer in step 6.
In step 7 the customer requests to buy tickets. The merchant application requests the signed tickets, which the customer wants to buy, from the ticket server in step 8. In step 9 the ticket server replies with the signed tickets to the merchant application. The merchant application

issues the valid and signed tickets to the customer in step 10. In step 11 the customer imports the issued tickets to his mobile client.

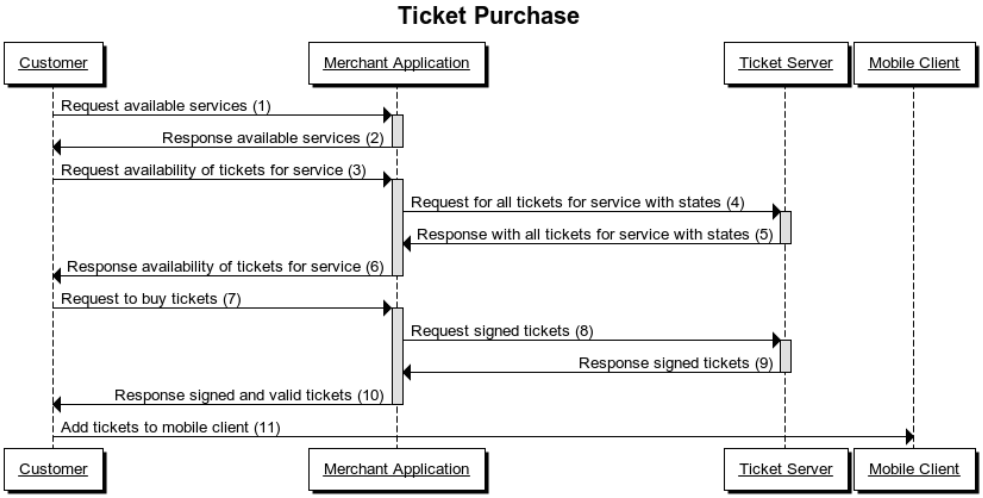

Figure 4.3: Sequence Diagram for Ticket Purchase

4.3.2 Prepare Mobile Validation Application

Figure 4.4 shows the sequence for preparing a mobile validation application prior to the ticket validation process.
In step 1 the merchant application requests a ticket validator certificate from the ticket server. In step 2 the ticket server replies with a valid ticket validator certificate. In step 3 the issued certificate is added to the mobile validation. In step 4 the mobile validation application requests the offline validation data from the ticket server. In step 5 the ticket server replies with the offline validation data to the mobile validation application.

4.3.3 Redeem Ticket

Figure 4.5 shows the single steps for redeeming a ticket.
In step 1 the mobile client transfers his ticket to the mobile validation application. The mobile validation application validates the ticket with the offline validation data in step 2. If the offline validation succeeds, the mobile validation application requests ticket validation and devaluation from the ticket server in step 3. The ticket server replies with the ticket validity and if it has been successfully devaluated in step 4.

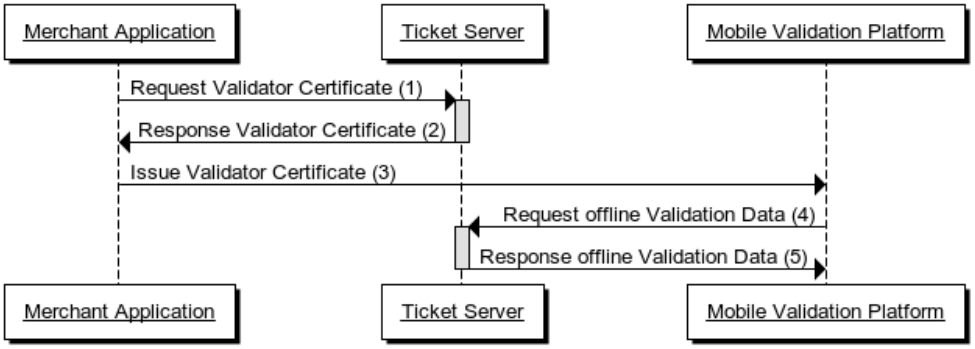

Figure 4.4: Sequence Diagram for Preparing Mobile Validation Application

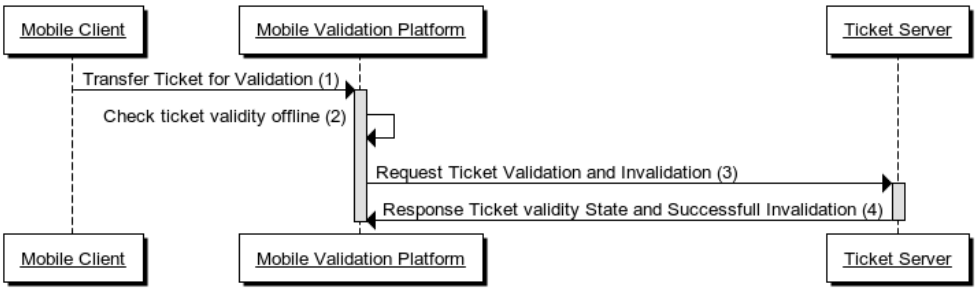

Figure 4.5: Sequence Diagram for Redeeming a Ticket

4.4 Proposed Java Technologies for a System Realization

This section presents Java technologies, which are capable for a realization of the presented mobile ticketing system. These technologies are state of the art, have a wide distribution and comply to enterprise requirements.

4.4.1 Java Technology Platform

Java is a software platform[1], which was created by Sun Microsystems Inc. Sun Microsystems Inc. was bought by the Oracle Corporation in 2010. Part of the software platform is the object oriented programming language Java. Java source code is compiled to Java byte-code and is ran by the Java Runtime Environment. The Runtime Environment compiles the byte-code to machine specific code. Java has a built in standard software library and is supported by many different frameworks. Important frameworks are presented in the following. The Java community process enhances the programming language and the standard software library.

[1] `http://www.oracle.com/technetwork/java/index.html`, accessed 25/07/2014

4.4.2 Spring Framework

Spring[2] is a Java application framework for building Java enterprise applications. The basis of Spring was introduced by Johnson in his book [45]. Spring offers a wide range of modules and services. The most important are:

- Dependency Injection: Spring has an inversion of control container for dependency injection.
- Aspect Oriented Programming: Aspects may be defined for cross cutting concerns in Spring.
- Transaction Management: Spring has a built in transaction management.
- REST Webservices: Spring has a built in framework for creating RESTful webservices.
- Data Access: Spring supports access to relational database management systems for data handling.
- Security: Spring offers many security services like authentication and authorization.
- Testing: Spring supports Unit- and Integration testing.

4.4.3 Hibernate

Hibernate[3] is an object relational mapping (ORM) library. Hibernate maps data from a relational database to Java objects in a transparent way. Different database systems are supported. Data may be accessed via the SQL-like Hibernate Query Language (HQL) or via the Hibernate Criteria-API. Hibernate implements the Java Persistence API (JPA) standard 2.1 as described in JSR-338 [37]. Hibernate is in depth described in the book [49] by King and Bauer. King is the founder of the Hibernate project.
The most important Hibernate components are:

- Hibernate ORM: The object relational mapper. It includes Hibernate Annotations for mapping, the Entity Manager for data handling and Hibernate Envers for versioning and auditing.
- Hibernate Search: Offers full text search functionality in domain objects.
- Hibernate Validation: Reference implementation of the Bean Validation standard as described in JSR-303 [39].

[2] `http://projects.spring.io/spring-framework/`, accessed 25/07/2014
[3] `http://hibernate.org/`, accessed 25/07/2014

4.4.4 JavaServer Faces

JavaServer Faces (JSF) is a specification for creating web application user interfaces. The specification is created by the Java Community Process. The current version is 2.2, which is specified in JSR-344 [38]. JSF is component based and part of the Java Enterprise Edition technology.

4.4.5 JavaFX

JavaFX[4] is a framework for building rich client frameworks. It is a client technology of Java SE and is released by the Oracle Corporation.

4.4.6 Jackson

Jackson[5] is a library for handling the JavaScript Object Notation (JSON) data format. The JSON data format may be used for RESTful web applications.

4.5 Ticket System Design

This section describes the components and technical details of the mobile ticketing system. The detailed functional requirements and usage scenarios are presented in appendix A. The presented design should be considered as a reference design with the possibility of customization for exercises or courses.

4.5.1 Ticket Server

The ticket server handles all tasks, which are directly related to ticketing. It is domain independent and can be utilized by different domain dependent merchant applications. The issued tickets are transferred through the merchant application to the mobile client.
The ticket server is the central part of this mobile ticketing system. The availability of this system must be kept high, since a downtime implies a complete stop of ticket handling for all hosted merchants. That must be avoided since selling and validating tickets are crucial business cases for merchants. Therefore, the described offline validation is used.
The response time of the validation interface must be short and it must be able to handle many parallel requests.
Along with different merchants different requirements arise for the ticketing process. The most widely used ticket types are presented in section 3.1.2.1, but the system is built with support for new type of tickets along with their respective method of validation.

4.5.1.1 Software Architecture and Technology Stack

Figure 4.6 presents the architecture of the ticket server, which consists of the following layers. Each layer is called by the above one and propagates a request to the one below.

[4] `http://docs.oracle.com/javase/8/javase-clienttechnologies.htm`, accessed 25/07/2014

[5] `http://wiki.fasterxml.com/JacksonHome`, accessed 25/07/2014

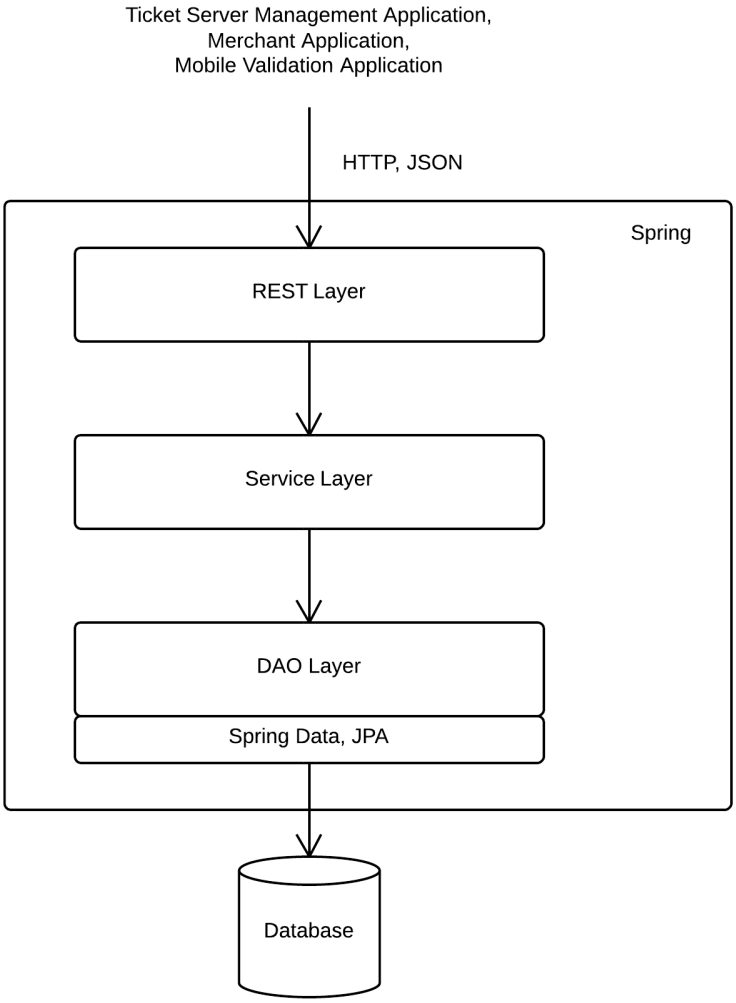

Figure 4.6: Software Architecture Ticket Server

- REST Layer: The REST layer provides the REST services for other system components and handles the incoming requests. It invokes the service layer and generates the replies. This layer utilizes the Spring REST functionality.
- Service Layer: The service layer encapsulates the business logic. It utilizes the DAO layer for data loading and saving.
- DAO Layer: The data access object (DAO) layer handles loading data from and saving data to the database. The loaded data from the database entities is mapped to data transfer objects (DTOs) to ensure that the higher layers are independent from the database. The

DAO layer utilizes Spring Data and JPA for object-relational data mapping (ORM) and querying.

Spring is used as the application framework to build the ticket server.

4.5.1.2 Functional Decomposition

Figure 4.7 shows the functional decomposition into modules of the ticket server along with their dependencies. The modules are described in the following.

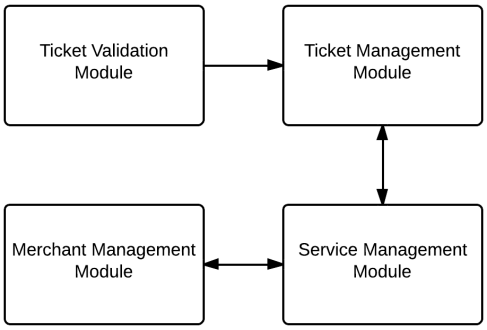

Figure 4.7: Module Structure Ticket Server

- Ticket Validation Module: The ticket validation module contains all the functionality that is related to ticket validation and devaluation. It uses the functionality of the ticket management module for ticket retrieval and update.

- Ticket Management Module: The ticket management module contains all functionality that is related to ticket handling, like generation, retrieve or update. It utilizes the service management module to retrieve the services.

- Service Management Module: The service management module contains all functionality that is related to service handling, like generation, retrieve, update or key generation. It utilizes the merchant management module for merchant retrieve and the ticket management module for ticket retrieve and update.

- Merchant Management Module: The merchant management module contains all functionality that is related to merchant handling, like generation, retrieve or update. It utilizes the service management module for service retrieve and update.

4.5.1.3 Package Structure

Figure 4.8 shows the package structure of the ticket server. The packages have the following contents:

- `rest`: Contains REST endpoint classes from the REST layer.
- `service`: Contains interfaces for the service implementations.
- `service.impl`: Contains implementations of the service interfaces.
- `dao`: Contains interfaces for the DAO implementations.
- `dao.impl`: Contains implementations of the DAO interfaces.
- `data.dto`: Contains data transfer objects, which are used in every layer.
- `data.entity`: Contains data entities. The DAOs map data entities to DTOs.
- `exception`: Contains exceptions.

4.5.1.4 Database Model

Figure 4.9 shows the data model of the ticket server.
The main entity is `Ticket` along with their subclasses `SingleTicket`, `ReusableTicket`, `TimeLimitedTicket` and `CustomTicket`. Based on the subclass, the business logic determines the validity of the ticket. The subclass gives additional information, which is needed for the validation and devaluation. Another type of ticket can be implemented by subclassing `Ticket` with according additional information for validation.
`Service` references a concrete service, for which the issued ticket may be used. Each `Service` has a `Keypair`, which is used for ticket signature generation and validation. `ServiceProvider` is the entity, which offers the service. Such a provider may have different offered services. `TicketAdministrator` is an administrator, who can manage the ticket server with the ticket server management application. `TicketAdministrator` and `ServiceProvider` are subclasses of `Account`.
`TicketIdentifier` is the concrete instance of a ticket, which is used for validation[6]. A `TicketIdentifier` belongs to exact one `Ticket`. A `Ticket` may have more than one `TicketIdentifier` but only one which is valid.
When a ticket is devaluated, a new `Voidation` entry is generated. A ticket is validated by it's signature using the public key of the `Keypair`, the validity of its `TicketIdentifier`, it's type and previous `Voidation`.
The `ValidatorCertificate` is used for authenticating a validating entity and tracking who created a `Voidation`.

[6]Hereby a ticket is not used in the usual nomenclature. The `TicketIdentifier` would be the entity, which is usually referenced by the noun ticket. The nomenclature `Ticket` is used here as an abstract entity, which denotes e.g. a seat in a show.

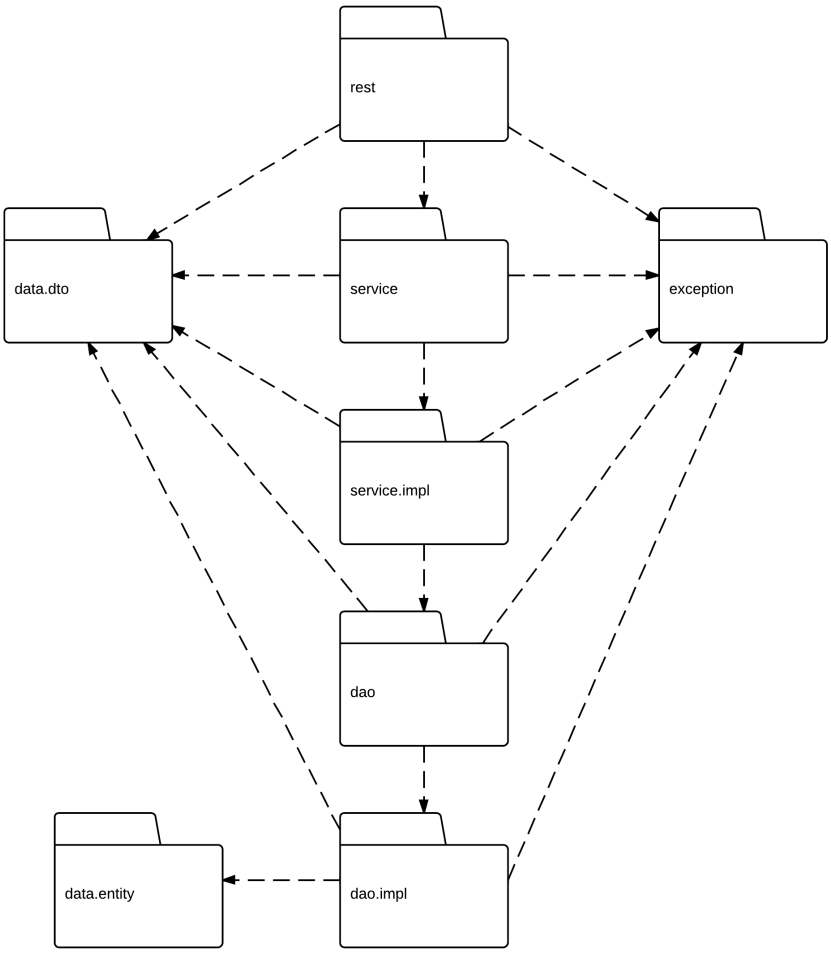

Figure 4.8: Package Structure Ticket Server

4.5.2 Ticket Server Management Application Design

The ticket server management application is used to administrate the ticket server. It is used by the ticket server administrator and utilizes the ticket server's interfaces.

4.5.2.1 Software Architecture and Technology Stack

Figure 4.10 presents the architecture of the ticket server's management application, which consists of the following layers. Each layer is called by the above one and propagates a request to the one below.

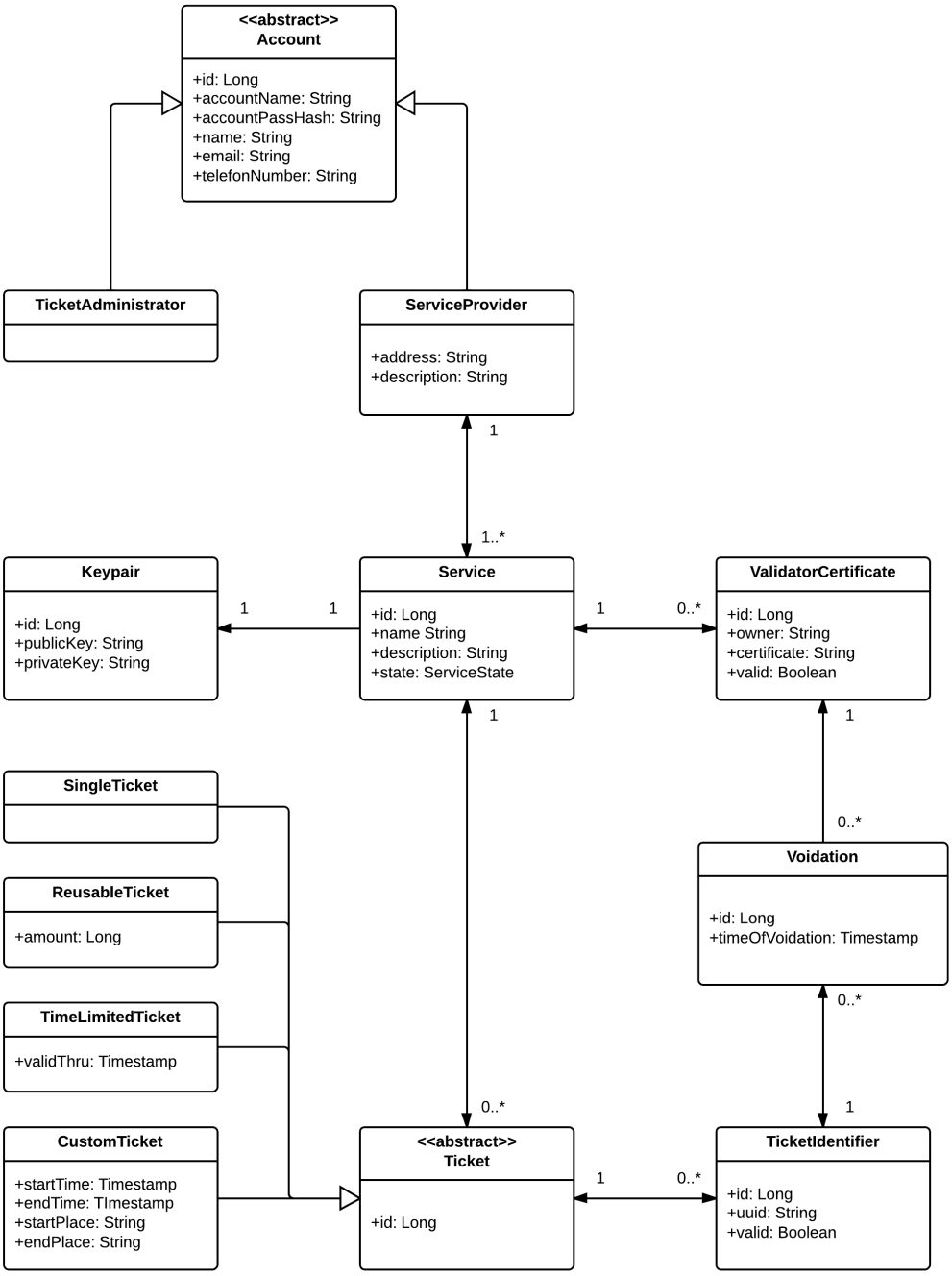

Figure 4.9: Data Model Ticketserver

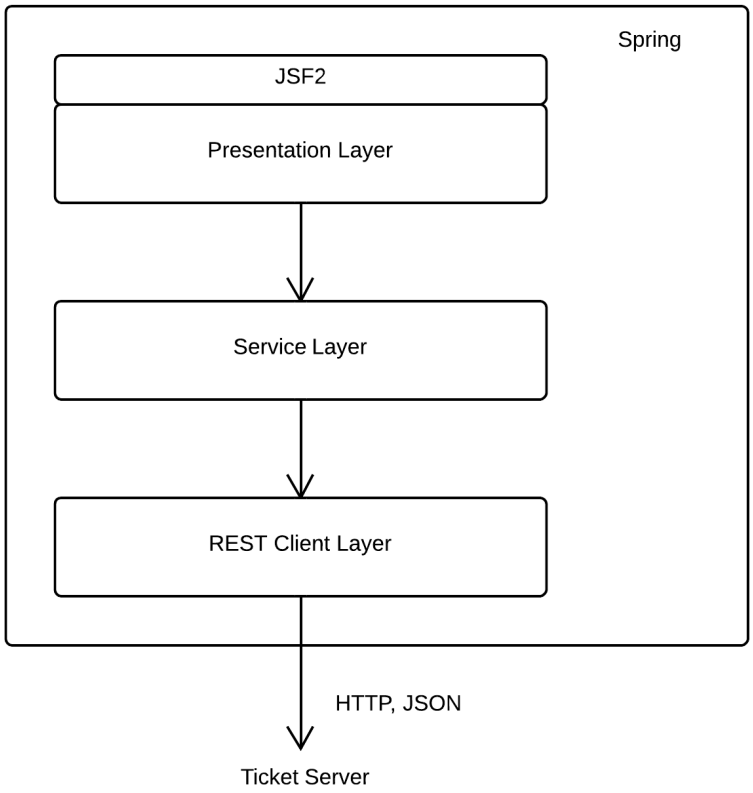

Figure 4.10: Software Architecture Ticket Server Management Application

- Presentation Layer: The presentation layer handles the view. It uses the underlying service layer. The presentation layer is created with JavaServer Faces (JSF) 2.

- Service Layer: The service layer encapsulates the business logic. It utilizes the REST client layer to utilize the functionality of the ticket server.

- REST Client Layer: The REST client layer invokes the REST interfaces of the ticket server. It is called by the above service layer and passes retrieved information from the ticket server to it.

Spring is used as the application framework to build the ticket server management application. The management application does not store ticket related data, since all relevant data is handled by the ticket server.

4.5.2.2 Functional Decomposition

Figure 4.11 shows the functional decomposition into modules of the ticket server management application along with their dependencies. The modules are described in the following.

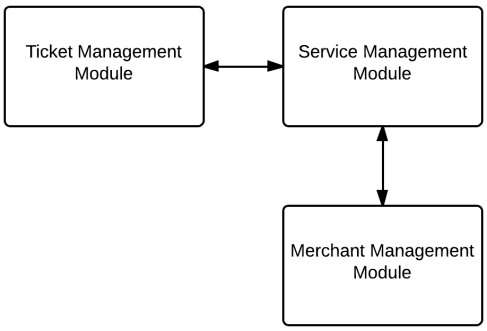

Figure 4.11: Module Structure Ticket Server Management Application

- Ticket Management Module: The ticket management module contains all functionality that is related to ticket administration, like generation, retrieve or update. It utilizes the service management module for service retrieve.
- Service Management Module: The service management module contains all functionality that is related to service administration, like generation, retrieve, update or key generation. It utilizes the merchant management module for merchant retrieve and the ticket management module for ticket retrieve and update.
- Merchant Management Module: The merchant management module contains all functionality that is related to merchant administration, like generation, retrieve or update. It utilizes the service management module for service retrieve.

4.5.2.3 Package Structure

Figure 4.12 shows the package structure of the ticket server's management application. The packages have the following contents:

- `ctrl`: Contains the JSF controller classes for the view.
- `service`: Contains interfaces for the service implementations.
- `service.impl`: Contains implementations of the service interfaces.
- `client`: Contains interfaces for the client implementations.

- `client.rest.impl`: Contains implementations of the client interfaces.
- `dao`: Contains interfaces for the DAO implementations.
- `dao.impl`: Contains implementations of the DAO interfaces.
- `data.entity`: Contains data entities. The DAOs map data entities to DTOs.
- `data.dto`: Contains data transfer objects, which are used in every layer.
- `exception`: Contains exceptions.

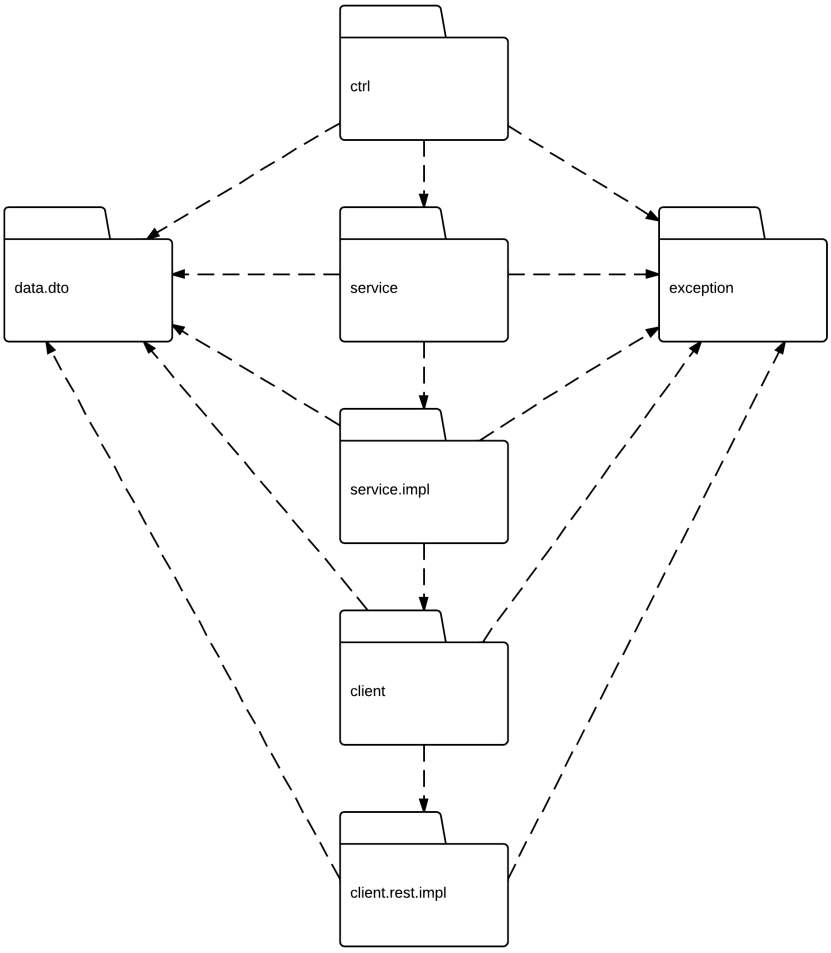

Figure 4.12: Package Structure Ticket Server Management Application

4.5.3 Merchant Application Design

The merchant application is created and operated by the merchant. It provides all domain dependent functionality along with offering services and tickets by utilizing the ticket server.
The merchant application doesn't need to be build upon a specific technology or platform. Basically every possible approach, which can utilize the ticket server's provided interfaces, is possible.
The merchant application depends on the domain and the merchant's specific requirements. The presented merchant application is an arbitrary example of a concrete merchant application.

4.5.3.1 Software Architecture and Technology Stack

Figure 4.13 presents the architecture of the merchant application, which consists of the following layers. Each layer is called by the above one and propagates a request to the one below.

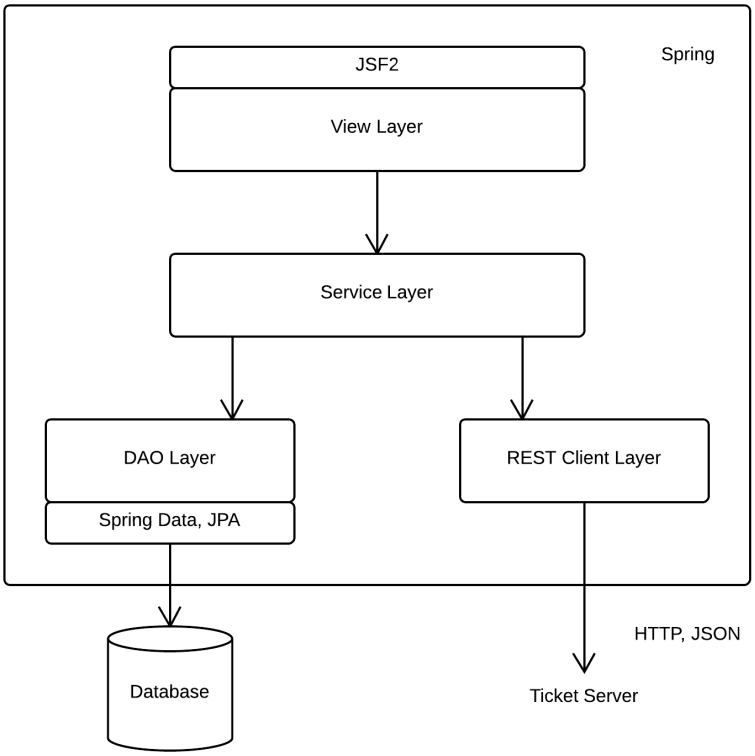

Figure 4.13: Software Architecture Merchant Application

- Presentation Layer: The presentation layer handles the view. It uses the underlying service layer. The presentation layer is created with JavaServer Faces (JSF) 2.

- Service Layer: The service layer encapsulates the business logic. It utilizes the REST client and DAO layer. It processes and aggregates data from the merchants database and the ticket server.
- REST Client Layer: The REST client layer invokes the REST interfaces of the ticket server. It is called by the above Service layer and passes retrieved information from the ticket server to it.
- DAO Layer: The data access object (DAO) layer handles loading data from and saving data to the database. The loaded data from the database entities is mapped to data transfer objects (DTOs) to ensure that the higher layers are independent from the database. The DAO layer utilizes Spring Data and JPA for object-relational data mapping (ORM) and querying.

Spring is used as the application framework to build the ticket server management application.

4.5.3.2 Functional Decomposition

Figure 4.14 shows the functional decomposition into modules of the merchant application along with their dependencies. The modules are described in the following.

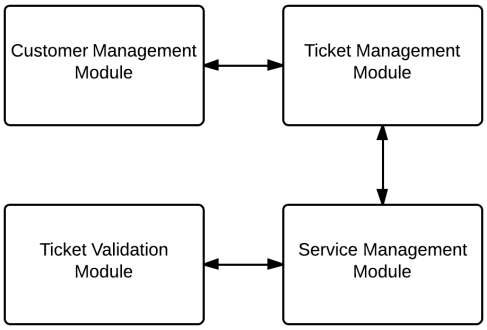

Figure 4.14: Module Structure Merchant Application

- Customer Management Module: The customer management module contains all functionality that is related to customer administration like generation, retrieve or update. It utilizes the ticket management module for ticket retrieve and update.
- Ticket Management Module: The ticket management module contains all functionality that is related to ticket administration, like generation, retrieve or update. It utilizes the service management module for service retrieve and the customer management module for customer retrieve.

- Service Management Module: The service management module contains all functionality that is related to service administration, like generation, retrieve, update or key generation. It utilizes the ticket validation module for validation information retrieve and the ticket management module for ticket retrieve and update.
- Ticket Validation Module: The ticket validation module contains all functionality that is related to ticket validation, like request ticket validator certificate or issue ticket validator certificate. It utilizes the service management module for service retrieve.

4.5.3.3 Package Structure

Figure 4.15 shows the package structure of the merchant application. The packages have the following contents:

- `ctrl`: Contains the JSF controller classes for the view.
- `service`: Contains interfaces for the service implementations.
- `service.impl`: Contains implementations of the service interfaces.
- `client`: Contains interfaces for the client implementations.
- `client.rest.impl`: Contains implementations of the client interfaces.
- `data.dto`: Contains data transfer objects, which are used in every layer.
- `exception`: Contains exceptions.

4.5.4 Mobile Client

The mobile client is installed on the users mobile and manages the obtained tickets. When the user wants to redeem a ticket, it is transferred to the mobile validation application for validation. The mobile client doesn't need to be build upon a specific technology or platform. Basically every possible technology, which can store and transfer a ticket in an appropriate way is possible. The mobile client should be easy to understand and use, so that users who are not very familiar with technical devices, can use it.

4.5.4.1 Software Architecture and Technology Stack

Figure 4.16 presents the architecture of the client application, according to the Android platform [28]. The architecture consists of the following modules.

- Activity: The activity components each represent a single user interface screen of the application. An activity utilizes the service components for long running operations and the content provider components for data handling.

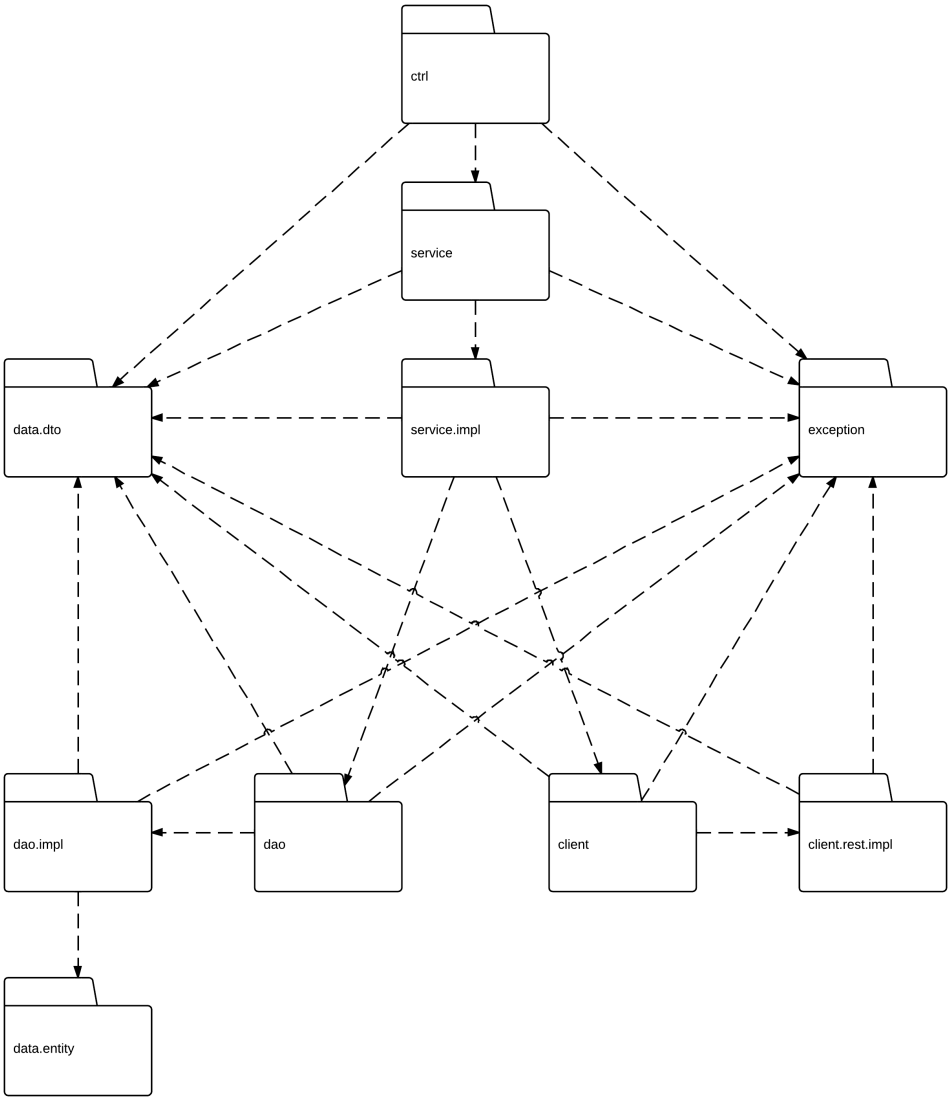

Figure 4.15: Package Structure Merchant Application

- Service: The service components are used for background tasks and long running operations. A service utilizes the content provider components for data handling.

- Content Provider: The content provider components are used for data handling. Tickets are stored on the file system and user or application data is stored in the SQLite database.

The application is executed on Android and utilizes the provided Android APIs.

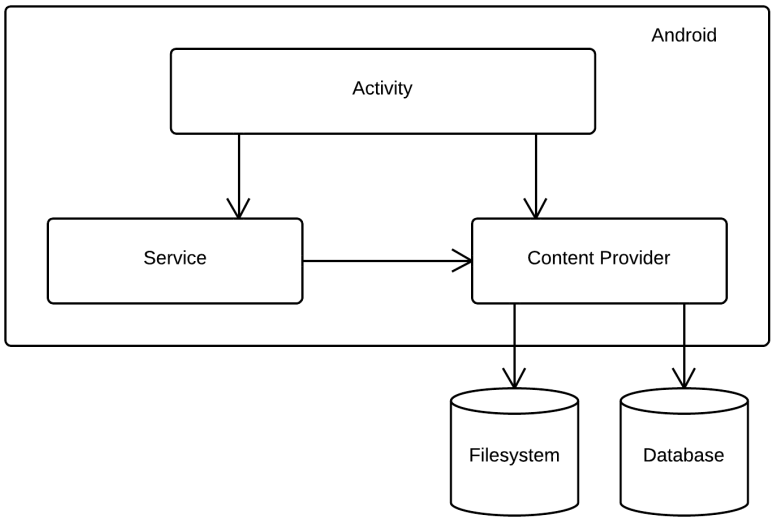

Figure 4.16: Software Architecture Client Application

4.5.4.2 Functional Decomposition

Figure 4.17 shows the functional decomposition into modules of the mobile client along with their dependencies. The modules are described in the following.

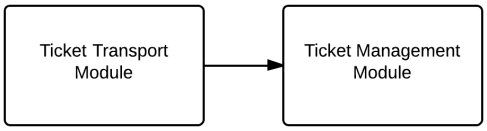

Figure 4.17: Module Structure Mobile Client

- Ticket Transport Module: The ticket transport module contains all functionality that is related to ticket transfer from the mobile client to the mobile validation application. It utilizes the ticket management module for ticket loading.

- Ticket Management Module: The ticket management module contains all functionality that is related to ticket handling, like storing, listing or showing details.

4.5.4.3 Package Structure

Figure 4.18 shows the package structure of the mobile client application. The packages have the following contents:

- `ui`: Contains the activity classes.
- `service`: Contains the service classes.
- `data`: Contains the content provider classes.
- `exception`: Contains exceptions.

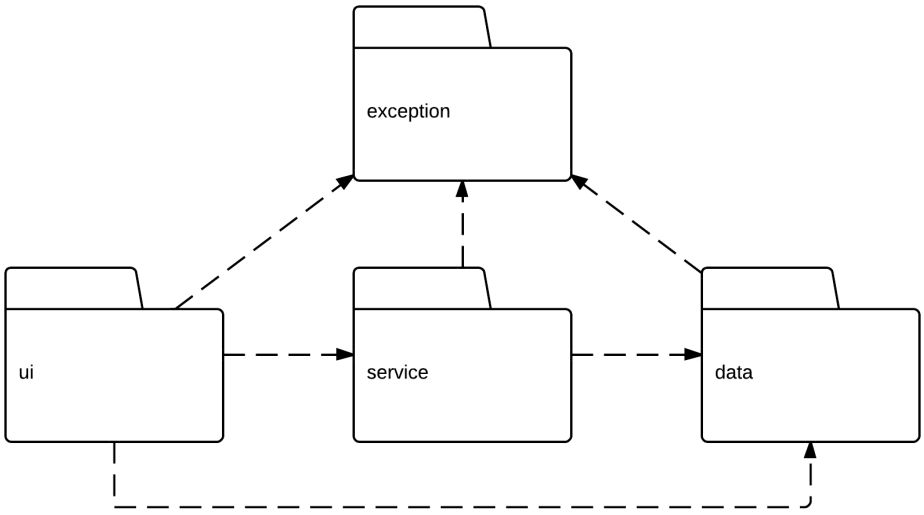

Figure 4.18: Package Structure Mobile Client Application

4.5.5 Mobile Validation Application Design

The mobile validation application does the ticket validation. The ticket is transferred from the mobile client to the mobile validation application. In case of a transfer failure, the transfer should be retried. A verified ticket should be devaluated according to its type by the validation application.
The mobile validation application doesn't need to be built upon a specific technology or application. Basically every possible technology, which can transfer a ticket in an appropriate way from the mobile client, is possible.
The mobile validation application can be either operated by a human or automatically by utilizing turnstiles or similar access controls.

4.5.5.1 Software Architecture and Technology Stack

Figure 4.19 presents the architecture of the mobile validation application, according to the Android platform [28]. The architecture consists of the following modules.

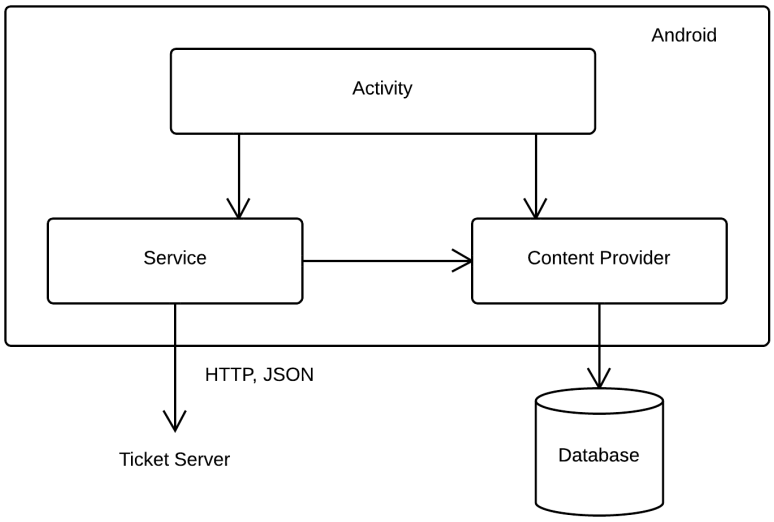

Figure 4.19: Software Architecture Mobile Validation Application

- Activity: The activity components represent each a single user interface screen of the application. An activity utilizes the service components for long running operations and the content provider components for data handling.

- Service: The service components are used for background tasks and long running operations. A service utilizes the content provider components for data handling. A service may also call the ticket servers REST interfaces.

- Content Provider: The content provider components are used for data handling. Data is stored in the SQLite database.

The application is executed on Android and utilizes the provided Android APIs.

4.5.5.2 Functional Decomposition

Figure 4.20 shows the functional decomposition into modules of the mobile validation application along with their dependencies. The modules are described in the following.

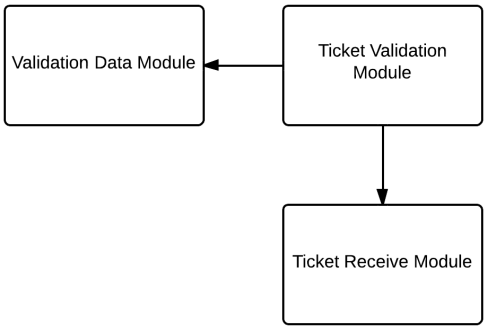

Figure 4.20: Module Structure Mobile Validation Application

- Validation Data Module: The validation data module contains all functionality that is related to administrating data needed for ticket validation, like retrieving offline validation data.
- Ticket Validation Module: The ticket validation module contains all functionality that is related to ticket validation, e.g. the functionality to validate a ticket off- or online. It utilizes the validation data module to retrieve needed validation data and the ticket receive module to receive the ticket to be validated.
- Ticket Receipt Module: The ticket receipt module contains all functionality that is related to ticket receipt from the mobile client, e.g. the functionality for reading a QR-code.

4.5.5.3 Package Structure

Figure 4.21 shows the package structure of the mobile validation application. The packages have the following contents:

- `ui`: Contains the activity classes.
- `service`: Contains the service classes.
- `data`: Contains the content provider classes.
- `exception`: Contains exceptions.

4.6 Ticket Data Structure

This section describes a possible data structure implementation of a ticket for the presented system. The proposed implementation is an archive file which consists of a JSON file, a signature and a HTML 5 document.

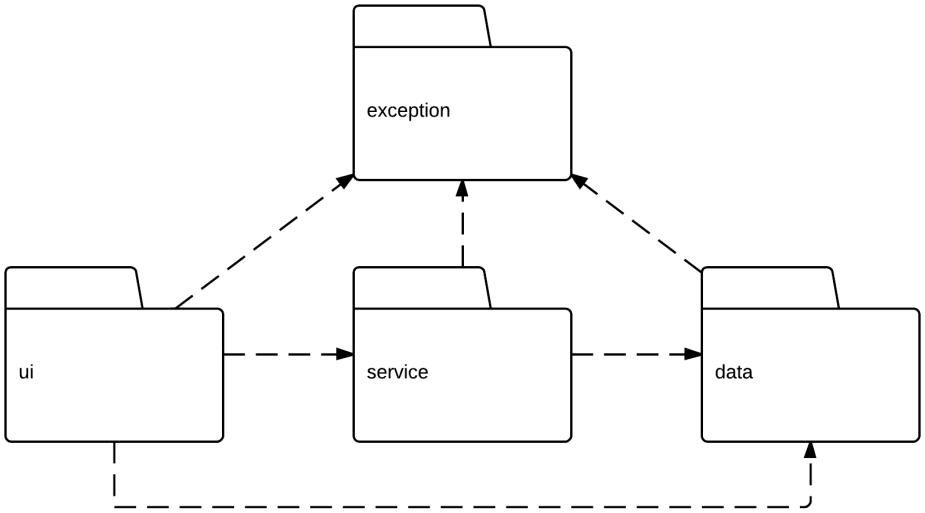

Figure 4.21: Package Structure Mobile Validation Application

4.6.1 Immanent Ticket Data - The JSON File and the Signature

The JSON file includes all immanent ticket data, which is needed and used in the validation process. The JSON file includes:

- The ticket's unique identifier, referenced in the JSON file as a string with the identifier `ticketIdentifier`.
- The service's unique identifier, referenced in the JSON file as a string with the identifier `serviceIdentifier`.
- The ticket's type, referenced in the JSON file as a string with the identifier `type`.
- The ticket's validity constraints, referenced in the JSON file as an object with the identifier `constraints`. It is optional and its content depends on the ticket type.

The following examples present two possible ticket JSON files.

```
{
  "ticketIdentifier": "FDA7B6E0-29F9-11E4-8C21-0800200C9A66"
  "serviceIdentifier": "75388"
  "type": "single"
}
```

Listing 4.1: Example of a Single Use Ticket

```
{
  "ticketIdentifier": "3D5B5D20-29FD-11E4-8C21-0800200C9A66"
  "serviceIdentifier": "75389"
  "type": "timeLimited"
  "constraints": {
    "validFrom": "2014-05-26T13:00:00+02:00"
    "validTo": "2014-05-26T18:00:00+02:00"
  }
}
```

Listing 4.2: Example of a Time Limited Ticket

The JSON file along with the signature is generated by the ticket server and is transfered to the mobile validation application.
Different cryptosystems exist which are capable of digital signature creation and verification. For the presented system the Rivest, Shamir and Adleman (RSA) cryptosystem [60] is used for signature creation and verification. RSA is commonly considered secure if implemented and used correctly.
The proposed file names for the JSON file and the signatur file are `ticket.json` and `signature.asc`.

4.6.2 Human Readable Ticket Data - The HTML 5 Document

The HTML 5 document intends to inform the user about the booked service, which the ticket stands for. It is created by the merchant application before the ticket is issued to the user. The content is arbitrary and it is used only on the user's mobile. It is not transfered to the mobile validation application and is not considered in the check of validity, since it is not signed. The mobile client may utilize an embedded web browser to present the document.
The proposed structure of the HTML5 document is a directory in the archive with the name `html`. The directory shall include a file named `index.html`, which is the index file for the HTML5 document. Except of the index file, the content and structure of this directory is arbitrary.

4.6.3 Ticket Archive Structure

The following directory tree lists the proposed structure of the ticket archive file. The ticket archive is compressed using the ZIP file format [59].

```
/
├── ticket.json ..... This JSON file contains the ticket immanent data.
├── signatur.asc. ..... This file contains the signature of ticket.json.
├── html ..... This directory contains the HTML5 document. The structure is arbitrary.
    └── index.html ..... This file is the index file for the HTML5 document.
```

CHAPTER 5

Tailoring of the Technical Foundation to Software Engineering Courses with Different Topics

This chapter introduces different software engineering courses, which may be supported with the presented foundation from chapter 4. An existing software engineering and project management course is described and the tailoring of the foundation in context of this course is presented.

5.1 Introduction of Software Engineering Courses

This section states types of software engineering courses with different topics, which address important knowledge areas of the SWEBOK. The objectives of the course types along with the possible support from and application of the presented foundation are described.

Basic Software Engineering A basic software engineering course should give students first insights in developing software as a team in a professional manner. A concrete example of such a course is described in detail with the application of the presented foundation in the following sections.

Advanced Software Engineering An advanced software engineering course should be built upon the gained knowledge of a basic software engineering course. Students should get further insights in developing software in a professional way. The presented foundation may be used untailored so that students implement a whole and complex software system.

Software Maintenance and Evolution Software ages over time and must be maintained, which is the topic of this course. A possible application of the foundation in this context is to provide an implementation of the foundation with old technologies to students. Students

have to take the provided code to the state of the art by changing outdated technologies and by implementing additional use cases.

Software Quality Assurance A Software quality assurance course should teach basic aspects of software quality. The foundation can support such a course by providing an implementation of low quality and known defects to students. They have to test the system, find bugs, create bug reports and improve the overall quality of the software.

Software Testing A course about software testing should give students deeper insights in validating software. It should teach advanced techniques of software testing. Students may learn such techniques by testing a provided foundation with special tools. That may involve mocking techniques, unit and integration testing or automated user interface testing.

Web Application Development A web application development course should teach students frameworks and technologies for implementing web applications. Parts of the presented system can be developed as web applications. A ticket server with defined REST interfaces may be provided to students. A possible assignment would be developing a merchant application by utilizing the provided ticket server.

Mobile Application Development A mobile application development course should teach students platforms and technologies for implementing mobile applications. A possible course assignment could be the implementation of the mobile client and the mobile validation application for Android or iOS. A ticket server could be provided for that assignment.

Security A course about software security should teach the fundamentals of software hardening and defensive programing. A faulty implementation of the presented foundation may has many attack vectors. Students may be assigned with a penetration test of such an implementation.

User Interface and Interaction Design A user interface and interaction design should teach students the design and development of user interfaces, which are easy to use and fast to understand. In context of the presented foundation, students may be assigned with the creation of user interfaces for the merchant application or the mobile client.

5.2 Introduction to an Existing Software Engineering and Project Management Course

This section introduces an already existing software engineering and project management course, which is addressed to students, who have basic knowledge in programming, common algorithms, design patterns, and object oriented design. This knowledge should have been taught in prior courses and are required to finish the presented course.
The current technical foundation of this course is outdated and is to be replaced with one that

has a modern architecture and uses future proof technologies. The organizational part of this course should remain unchanged.

5.2.1 Structure of the Course

In Austria, an academic course is usually held within four months, which is the time frame for the presented course. In the first month the required knowledge, which students should possess to finish this course, is evaluated. Afterwards the project phase starts, which has a time frame of three months.
Students should gain further knowledge in software engineering in a professional manner. They team up in groups of four to six people to create a theoretically shippable software from given requirements.
Each group is assigned to a tutor and a teacher. The tutor's duty is to supervise and assist the group during the project. Therefore, he meets with the group once per week to verify the project state, to give feedback and to help if problems arise. The teacher's duty is to evaluate and grade the project. He meets with the group and the tutor three times during the course. At the first meeting, the group should be familiar with the requirements, have set up the project environment, have created the software architecture and have implemented the first functionality. At the second meeting, the group should have finished two thirds of the project. At the last meeting, the group must have finished the project and present it. Based on the project outcome and the process how the project was created the teacher assesses the group's work.

5.2.2 Software Development Methodology

The project is created in an agile Scrum-like development framework. In difference to the Scrum model [65], students can't work full-time on the project what leads to the following customizations:

- A sprint has a length of one to three weeks.
- Every team member should work 12 hours per week on the project.
- The Scrum master and the product owner are not assigned.
- The daily Scrum is held only twice per week.

Each group has a provided Redmine instance and a Git repository. Redmine serves as the central project management tool, Wiki system and tracker. All documentation, tickets and time tracking must be done with it. The source code must be managed in the Git repository.

5.2.3 Scope of the Course

The assumed customer of the project is the fictitious Austrian company Ticketline, which sells tickets for different events like cinema, theater, concert, ... and merchandise articles. Ticketline provides shops, where customers can get information about events, buy tickets and merchandise articles. The Ticketline company needs an IT system to support their shop assistants. The

detailed requirements are given to students as user stories, which are presented in appendix B. The following stakeholders are assumed in this software engineering course:

- Ticketline Company Representative: The representative of the Ticketline company represents the company. He is interested in maximizing the revenue of the company and to satisfy his customers.
- Ticketline Company Shop Assistant: The shop assistant assists customers. He provides information, sells tickets, reserves tickets and sells merchandise. He is interested in an easy to use application which supports him in his tasks.
- Ticketline Customer: The Ticketline customer interacts with the shop assistant and is interested in buying tickets, reserving tickets and buying merchandise.

The deliverables, which must be created by each group, are:

- Time Tracking: Each student must track his work along with the required time.
- Meeting Protocols: Every group meeting must be documented.
- Project Risks: Project risks must be identified, rated and evaluated.
- Gantt Chart: A Gantt chart must be created for project planning.
- User Interface Prototype: Prior to the implementation, each group must design a user interface prototype.
- Software Architecture: Prior to the development the software architecture of the application must be documented.
- Design Document: The design document describes the design decisions of the application. It includes a high level class diagram of the application, used design patterns, the domain model, exception handling, interface specifications, logging policies and security aspects.
- Development Policies: Each group creates development policies which must be followed in the construction.
- Application: The current state of the application is presented in every meeting to the tutor and the teacher. Along with the application, unit tests must be created.
- Test Plan: The test plan outlines how software testing is performed. The software must be tested manually and in an automated fashion.
- Manual Test Cases: Describes manual test cases for testing the application.
- Test Reports: Each execution of the manual test cases must be documented as test reports. Bugs must be documented and tracked.

The addressed knowledge areas of this course according to the SWEBOK guide [10], are:

- Software Design: Students have to work out the detailed design of the system. They have to determine the software architecture and make design decisions regarding the system.
- Software Construction: Students have to create a working software throughout the course. At the end, the system must be finished and be theoretical shippable.
- Software Testing: The created software must be tested by the students. They have to determine test cases, execute them and create test reports.
- Software Engineering Process: The project is created in a Scrum-like development framework. Therefore, students must deal with software engineering processes and follow the Scrum model.
- Software Quality: Students have to apply knowledge about software quality to deliver a product which is in conformance to the requirements.
- Software Engineering Professional Practice: The course is designed to simulate an professional industrial project. So students must act in the project in a professional manner to deliver a good product.

5.3 Tailoring of the Technical Foundation to an Existing Software Engineering and Project Management Course

This section introduces the elevation of the existing software engineering and project management course from section 5.2 to the state of the art by using the presented technical foundation from chapter 4.

5.3.1 System Overview

Based on the conceptual design of the technical foundation from chapter 4, a tailored design for the stated software engineering and project management course is presented. The architecture of the foundation is simplified since the course is addressed to yet inexperienced students who would be overwhelmed by the full fledged application. Also the given architecture does not fit the project requirements. Therefore the foundation is tailored to a client-server architecture, which consists of an application backend and a rich client application:

- Application Backend: The backend handles all tasks and use cases of the Ticketline system. It provides REST interfaces for the rich client application.
- Rich Client Application: The rich client application is used by the shop assistant. It provides an user interface and utilizes the provided REST interfaces from the application backend.

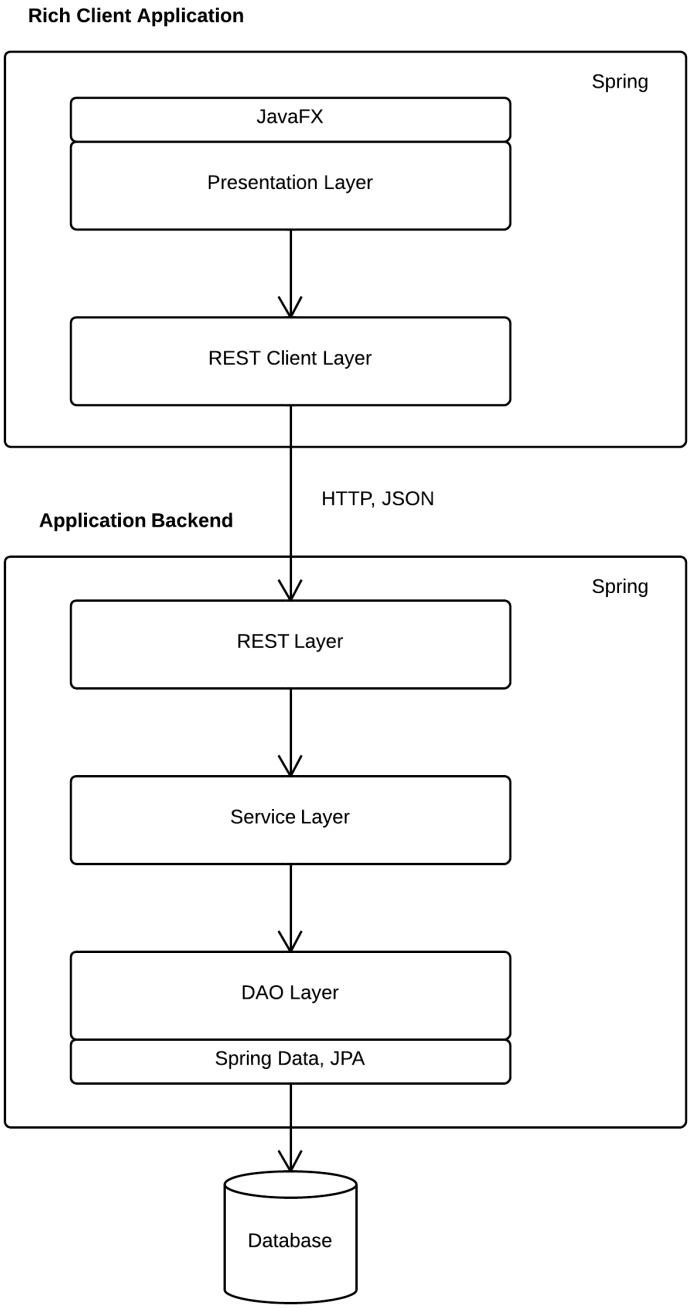

Figure 5.1: Simplified System Design

Figure 5.1 gives an schematic overview of the simplified architecture. The details of the two applications are stated in the following sections.
The tailored and reduced design is specific for the Ticketline company and does not support multiple clients. The application backend corresponds to the ticket server and the rich client application corresponds to a merchant application. The mobile client, the mobile validation application and the ticket server management application were omitted.
The core frameworks and libraries, which are used in the course to build the system, are:

- Spring: Used as the application framework to build the system.
- JavaFX: Used to build the user interface of the rich client application.
- Hibernate: Used as the ORM mapper.
- Jackson: Used for marshaling and unmarshaling JSON objects from and to Java objects.
- JUnit: Used as unit testing framework.
- Tomcat: Used as the servlet container.

5.3.2 Design of the Application Backend

The application backend contains all the logic and the data of the system. It provides REST interfaces for the rich client application. Spring is used as the application framework. The application backend runs in a Tomcat servlet container. It consists of the following layers:

- REST Layer: The REST layer provides the REST services for the rich client application. It invokes the service layer and generates the responses. This layer utilizes the Spring REST functionality.
- Service Layer: The service layer encapsulates the business logic. It utilizes the DAO layer for data loading and saving.
- DAO Layer: The data access object (DAO) layer handles loading data from and saving data to the database. The DAO layer utilizes Spring Data and JPA for object-relational data mapping (ORM) and querying.

5.3.3 Design of the Rich Client Application

The rich client application contains only the presentation logic. For handling the use cases it utilizes the provided REST interfaces of the application backend. Spring is used as the application framework. The rich client application consists of the following layers:

- Presentation Layer: The presentation layer handles the view. It uses the underlying REST layer. The presentation layer is created with JavaFX.
- REST Client Layer: The REST client layer invokes the REST interfaces of the application backend. It is called by the above presentation layer and passes retrieved information from the application backend to it.

5.4 Addressed SWEBOK Knowledge Areas with the Tailored Foundation

The tailored foundation in the context of the software engineering and project management course addresses the following SWEBOK knowledge areas:

- Software Design: Students have to create the software design in the context of the Spring Framework. They also have to follow and apply the provided state of the art client-server architecture of the system.
- Software Construction: Students have to create the system using different software frameworks. Also code conventions are defined, which must be followed.
- Software Testing: Students have to create unit tests to test parts of the system in isolation. Also manual test cases have to be determined and executed.
- Software Quality: Besides testing software, students have to apply static code analyses. The software must be in conformance to the requirements and must support the shop assistant in his tasks.
- Software Engineering Process: Students must apply the Scrum-like development framework. The software must be tested and in a theoretical shippable state at the end of each sprint.
- Software Engineering Professional Practice: Students must act at the project in a professional manner. Therefore, they must communicate in the team, work together and solve conflicts.

5.5 Provided Code Artifacts for the Software Engineering and Project Management Course

At the beginning of the course, students receive a prototype, which contains an already implemented use case and the data model. This prototype eliminates the initial effort for the project setup which can be quite high for unexperienced students and possess a high risk for early project termination. Therefore and because of the positive experience from the existing course this concept was taken over to the new system. Also, code reuse is a topic in this course were students should learn to make themselves familiar with existing code.
Students are encouraged to use, extend and change the prototype based on their needs within their own projects. The prototype is a Maven multi module project, which parts are described in the following. Figure 5.2 shows the dependencies between the modules.

- tl_client: This module contains the rich client application. It utilizes the tl_dto module.
- tl_server: This module contains the REST layer and service layer of the application backend. It utilizes the tl_dto module and the tl_db module.

- tl_dto: This module contains data transfer objects and is used by the tl_client and tl_server modules.
- tl_db: This module contains the DAO layer and the database entities of the application backend. It is used by the tl_server and the tl_data_generator modules.
- tl_data_generator: This module contains functionality to generate data in the database. It utilizes the tl_db module.

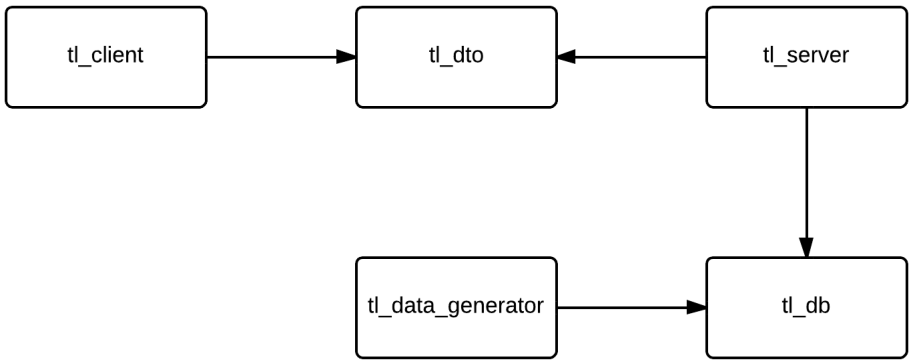

Figure 5.2: Dependencies Maven Modules

CHAPTER 6

Evaluation

This chapter evaluates the application of the mobile ticketing system design from chapter 4 in context of the presented course from chapter 5. The design of the evaluation is given and the results are discussed. The given course is also compared to the related work from section 2.2.

6.1 Evaluation Design

This section describes the design of the qualitative evaluation of the tailored foundation in context of the presented software engineering and project management course. The goal of the evaluation is the assessment of the application of the technical foundation and the discovery of possible improvements. The qualitative approach was chosen over the quantitative in favor of the additional information the material contains, according to Bortz and Döring [9].
The evaluation is done through a questionnaire, which is given to and answered by the tutors. The tutors are chosen as the interviewee since they had the most insights in the groups work due to the weekly meetings. The questions were created in accordance to the presented guidelines by Bortz and Döring [9].
The following enumeration lists the english questionnaire. The tutors received the survey in their native language german, which is given in appendix C.1.

1. User Stories and Assignment Description

    a) Did the students have difficulties to understand the provided user stories or the assignment description? If yes, which ones?

    b) Were adjustments of the user stories or the assignment made? If yes, which ones?

    c) Were questions regarding the user stories or the assignment description asked by the students? If yes, which ones?

2. Provided Code and Data Model

    a) Did the students have troubles to familiarizing themselves with the provided code? If yes, which ones?

    b) Were adjustments made to the provided data model? If yes, which ones?

    c) Were parts of the provided code seen as erroneous by a majority of the students? If yes, which ones?

    d) Were functions or whole artifacts missed by the students in the provided content? If yes which ones?

3. Provided Architecture

    a) Did the students have trouble with the provided architecture or did misunderstandings arise? If yes, which ones?

    b) Were adjustments made to the provided architecture? If yes, which ones?

    c) Were questions regarding the provided architecture asked by the students? If yes, which ones?

    d) Were violations of the provided software architecture identified? If yes, which ones?

4. Application of Frameworks and Construction

    a) Did the students understand the purpose of using software frameworks? If not, which lacks of clarity existed?

    b) Did problems with using the software frameworks arise? If yes, which ones and how were they solved?

5. Project Result

    a) Could students successfully finish their project? If not, why not?

    b) Did the finished projects satisfy the defined requirements? If not, which ones weren't satisfied?

6.2 Evaluation Discussion

This section discusses the results of the survey. The uninterpreted, filled out surveys, as they were answered by the tutors, are given in appendix C.2. The survey was given to nine tutors, who supervised the course in the last term. Among them, five returned a filled out survey.

6.2.1 Discussion regarding the User Stories and Assignment Description

The returned surveys state that the assignment was generally well understood by the students. A few questions were asked regarding details of user stories. Seen from the point of view that real world user stories also leave out some details, an adaptation to the given ones isn't necessary. But regarding the payment process and invoice generation a clarification or additional user

stories would be necessary. Another question was if user interaction concurrency must be considered. Since user interaction concurrency isn't in the scope of this course it should be clarified that it is not part of the project.
Students should be encouraged to implement own ideas and to do more than the essential assignment. As the surveys state, outstanding groups did more than the essential and implemented technical finesses in the project.

6.2.2 Discussion regarding the Provided Code and Data Model

The provided code artifact and data model should help students to make themselves familiar with the provided architecture and technologies. Students were encouraged to modify the given code and the data model, if they desire so, without violating the architecture.
The surveys show that the students generally perceived the provided artifacts as helpful. Initial problems were the usage of the involved frameworks since students may have been overwhelmed by them at the start of the project. The tutors clarified questions and described the frameworks.
The data model was slightly adapted by most groups. One group even created a completely new model.
One survey states that students didn't understand the purpose of the data generator, which generates the test data. That should be clarified in the assignment.
One group felt the need of aspect oriented programming in the provided artifacts. They altered the provided code and used aspects for cross-cutting concerns, e.g. the logging.

6.2.3 Discussion regarding the Provided Architecture

The surveys show that the groups generally followed and understood the provided architecture. Some groups enhanced the architecture by implementing aspects for cross cutting concerns or switched to an enterprise grade database.
Some groups implemented input validation only on the client. Since the server should also validate the input that should be clarified in the assignment. Also the REST concept wasn't fully understood. Some projects violated the HTTP interfaces or used the HTTP methods as function calls instead of resources. The REST concept should be taught more in depth before students start with the assignment.
Some general questions were asked regarding the difference between data transfer objects and entities, the difference between unit and integration testing or the meaning of mocking. Another question was if the client should also have a service layer. The service layer corresponds to the REST layer in the provided architecture. For these types of questions the tutor is the one who should clarify them since these are very specific and don't require a general discussion.

6.2.4 Discussion regarding the Application of Frameworks and Construction

The surveys state that the most questions and problems arose by utilizing the provided frameworks. Such problems were lazy and eager loading with Hibernate, the loop detection and resolution in Jackson or the Maven integration in Eclipse. To overcome such problems, an understanding of the used frameworks is essential. The concepts behind the frameworks must be

clear. The tutors usually have that knowledge and should clarify these questions.
Another problem was the usage of the version control system Git[1]. Most students haven't worked with Git before which led to some difficulties. Since Git is state of the art, students should make themselves familiar with it. Tutors helped them out if they faced problems.
Some outstanding groups refined the development process by utilizing additional tools like a build server. Some also switched from the given in-memory database to an enterprise grade database like PostreSQL[2]. Other refinements were polling or server push in the top ten events to get instant updates. Some groups implemented instant searches in the search UI.
Generally the purpose of frameworks was well understood by the students. The surveys state that some frameworks give unclear error messages and that the familiarization with them takes a long time. But the tutor assists students in these tasks.

6.2.5 Discussion regarding the Project Result

According to the surveys, the students were generally able to finish the projects and were graded accordingly. The quality was partially not perfect since bugs existed or the usability wasn't good. Stated examples for this are input validation or the ticket workflow.
This course is a simulated real world project, which is the first bigger project for most students. Therefore a perfect solution can't be expected. But students should see and understand the potential of improvements so they can learn from their mistaktes for future projects.

6.2.6 Evaluation Summary

The evaluation shows that the assignment was generally well received by the students and that the learning objectives could be imparted to them. The problems which arose the most were specific and depended on previous experiences of the students. Most of these issues could be solved with help from the tutors.
Some clarification is needed regarding the implementation of the payment process, the invoice generation or the concurrency handling. Also the data generator, the validation and the REST concept should be presented in more depth.
Generally the foundation fitted the course very well. The students' motivation was high and most of the projects were successfully finished.

6.3 Comparison of the Course to Related Work

Based on the literature survey of section 2.2, the general design of the project assignment is compared to the related work in the following.
Section 2.2.1 describes the problem based learning approach for software engineering by Tian et al. [68]. In difference to the approach from Tian et al., there is only one project proposal for all groups. Therefore it is assured that the solutions are comparable and the complexity is equal for each group. Requirements engineering is omitted so that the groups can concentrate on the

[1] `http://git-scm.com/`, accessed 25/07/2014
[2] `http://www.postgresql.org/`, accessed 25/07/2014

software construction and testing. Each group presents their work regularly to the tutor and the teacher. At the end of the semester good solutions are chosen, which are presented to all students in place of a regular discussion.
Like in the work by Roshandel et al. [61] students work on a project with industrial needs. In difference, the project is not for a real customer, since students may not yet have enough experience to create industrial software.
As in the work by Krutz and Meneely [50], the assignment is built around web services. Krutz and Meneely state that students were motivated and benefited by using these technologies. The tutor and the teacher also have the two roles as the teacher and the customer. The groups likewise use their artifacts in the discussions. Different environments were omitted in favor of the other phases.
Spring is used as the general application framework. Students have to apply good design practice in context of that framework, whereby the problem statement as presented by Ali et al. [1] arise. Students are encouraged to learn the framework and to apply good design decision in it. The tutors assisted the groups and described how to apply patterns in the context of the Spring framework.
Sherriff points out in his article [66] that students were excited by using state of the art technologies and developing skills which they can use immediately. The presented assignment involves using REST web services. So students may apply the learned lessons to utilize available popular web services like the Facebook APIs.

CHAPTER 7

Summary and Future Work

This thesis introduced a state of the art mobile ticketing foundation for application in software engineering education and training. The creation of the foundation is composed of several tasks. The basics of software engineering education and training were researched and presented. The knowledge that a trained software engineering professional should posses, according to the SWEBOK, was introduced and related work of education and training stated. The state of the art of mobile ticketing was investigated and introduced. Related work and supporting technologies of mobile ticketing were described.

Based on the related work, a mobile ticketing system foundation for education was created and presented in detail. Possible stakeholders for this system were described. A system overview along with the system details and a ticket data structure were given. The ticket system satisfies general requirements of a mobile ticketing system but is flexible enough for tailoring to specific software engineering courses.

Such a tailoring was presented through the elevation of an existing software engineering and project management course to the state of the art by applying the foundation. Hereby, the existing course was described and the details of the tailoring were stated. Knowledge areas, which are addressed by utilizing the foundation in the context of the course, were described.

The use of the foundation in the presented course was evaluated by a survey, which was given to the tutors. The evaluation indicated that the course was generally well received by the students. Learning objectives were well imparted through the foundation.

Some issues arose by the application, which should be addressed. Some clarification regarding the payment process, the invoice generation and the concurrency handling should be done. The technical details of the provided code and the REST concept should be presented more in depth.

Future work could be the tailoring and application of the foundation to other university grade courses. Another starting point could be taking students proposed requirements more into account in the assignments.

Technical improvements could be an enhanced offline validation protocol through a synchronization procedure between the validating applications. Saving tickets in the cloud could also be considered. The concrete implementation of the payment process with payment mocks could

also be part of future work. Other enhancements could be report generation and performing statistical analysis on the ticket server.
In summary, this thesis indicates that the presented foundation fits software engineering courses well. Students can work on different assignments with different topics without becoming familiar with new domains each time. Teachers and tutors have a foundation, which can be easily tailored for their courses without too much hassle.

List of Figures

Bibliography

[1] Zoya Ali, Joseph Bolinger, Michael Herold, Thomas Lynch, Jay Ramanathan, and Rajiv Ramnath. Teaching object-oriented software design within the context of software frameworks. In *Frontiers in Education Conference (FIE), 2011*, 2011.

[2] Amazon Inc. Amazon Media Room: Press Releases. `http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1565581&highlight=`, 2011. Accessed: 23/07/2014.

[3] Apple Inc. Apple - Press Info - iOS 7 With Completely Redesigned User Interface & Great New Features Available September 18. `https://www.apple.com/pr/library/2013/09/10iOS-7-With-Completely-Redesigned-User-Interface-Great-New-Features-Available-September-18.html`, 2013. Accessed: 23/07/2014.

[4] Apple Inc. Apple - Press Info - Apple Unveils iOS7. `https://www.apple.com/pr/library/2013/06/10Apple-Unveils-iOS-7.html`, 2013. Accessed: 23/07/2014.

[5] Apple Inc. iOS Dev Center - Apple Developer. `https://developer.apple.com/devcenter/ios/index.action`, 2014. Accessed: 23/07/2014.

[6] Apple Inc. iBeacon for Developers - Apple Developer. `https://developer.apple.com/ibeacon/`, 2014. Accessed: 23/07/2014.

[7] Apple Inc. Passbook for Developers - Apple Developer. `https://developer.apple.com/passbook/`, 2014. Accessed: 23/07/2014.

[8] Stuart J. Barnes. The mobile commerce value chain: analysis and future developments. *International Journal of Information Management*, 22(2), 2002.

[9] Jürgen Bortz and Nicola Döring. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*. Springer Medizin Verlag Heidelberg, 2006.

[10] Pierre Bourque and Richard E. (Dick) Fairley. *Guide to the Software Engineering Body of Knowledge Version 3.0*. IEEE Computer Society, 2013.

[11] Harry Bouwman, Henny De Vos, and Timber Haaker. *Mobile Service Innovation and Business Models*. Springer Berlin Heidelberg, 2008.

[12] Tim Bray. The javascript object notation (json) data interchange format. Technical Report RFC 7159, Internet Engineering Task Force (IETF), 2014.

[13] Jon Callas, Lutz Donnerhacke, Hal Finney, D. Shaw, and R. Thayer. Openpgp message format. Technical Report RFC 4880, Internet Engineering Task Force (IETF), 2007.

[14] Serge Chaumette, Damien Dubernet, Jonathan Ouoba, Erkki Siira, and Tuomo Tuikka. Architecture and comparison of two different user-centric nfc-enabled event ticketing approaches. In *Smart Spaces and Next Generation Wired/Wireless Networking*. Springer Berlin Heidelberg, 2011.

[15] Yu-Yi Chen, Chin-Ling Chen, and Jinn-Ke Jan. A mobile ticket system based on personal trusted device. *Wireless Personal Communications*, 40(4), 2007.

[16] David Conde-Lagoa, Enrique Costa-Montenegro, Francisco J. González-Castaño, and Felipe Gil-Castiñeira. Secure etickets based on qr-codes with user-encrypted content. In *Digest of Technical Papers International Conference on Consumer Electronics (ICCE), 2010*, 2010.

[17] Zhongliang Deng, Dejun Zou, Jianming Huang, Xu Chen, and Yan pei Yu. The assisted gnss boomed up location based services. In *5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009*, 2009.

[18] Subhankar Dhar and Upkar Varshney. Challenges and business models for mobile location-based services and advertising. *Communications of the ACM*, 54(5), 2011.

[19] ECMA. The JSON Data Interchange Format. Norm ECMA-404, European Computer Manufacturers Association, 2013.

[20] ECMA. Near Field Communication Interface and Protocol (NFCIP-1). Norm ECMA-340, European Computer Manufacturers Association, 2013.

[21] ECMA. Near Field Communication Interface and Protocol -2 (NFCIP-2). Norm ECMA-352, European Computer Manufacturers Association, 2013.

[22] Heidi J. C. Ellis, Gregory W. Hislop, Mel Chua, and Sebastian Dziallas. How to involve students in foss projects. In *Frontiers in Education Conference (FIE), 2011*, 2011.

[23] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[24] Roy Thomas Fielding and Julian Reschke. Hypertext transfer protocol (http/1.1): Message syntax and routing. Technical Report RFC 7230, Internet Engineering Task Force (IETF), 2014.

[25] Jerry Gao, Vijay Kulkarni, Himanshu Ranavat, Lee Chang, and Hsing Mei. A 2d barcode-based mobile payment system. In *Third International Conference on Multimedia and Ubiquitous Engineering, 2009*, 2009.

[26] Stefano Levialdi Ghíron, Serena Sposato, Carlo Maria Medaglia, and Alice Moroni. Nfc ticketing: a prototype and usability test of an nfc-based virtual ticketing application. In *First International Workshop on Near Field Communication, 2009*, 2009.

[27] Google Inc. Android. `http://www.android.com/`, 2014. Accessed: 11/07/2014.

[28] Google Inc. Android Developers. `https://developer.android.com/index.html`, 2014. Accessed: 23/07/2014.

[29] Google Inc. NFC Basics | Android Developers. `https://developer.android.com/guide/topics/connectivity/nfc/nfc.html`, 2014. Accessed: 23/07/2014.

[30] Frederick Hirsch, John Kemp, and Jani Ilkka. *Mobile Web Services: Architecture and Implementation*. John Wiley & Sons, 2007.

[31] Jhe-Yi Hu, Chien-Cheng Sueng, Wei-Hsiang Liao, and Chian C. Ho. Android-based mobile payment service protected by 3-factor authentication and virtual private ad hoc networking. In *Computing, Communications and Applications Conference (ComComAp), 2012*, 2012.

[32] Hao Huang, Lu Liu, and JianJun Wang. Diffusion of mobile commerce application in the market. In *Second International Conference on Innovative Computing Information and Control, 2007.*, 2007.

[33] Bluetooth Special Interest Group Inc. BLUETOOTH SPECIFICATION Version 4.1. Technical report, Bluetooth Special Interest Group Inc., 2013.

[34] NFC Forum Inc. NFC Data Exchange Format (NDEF) Technical Specification. Technical report, NFC Forum Inc., 2006.

[35] NFC Forum Inc. NFC Record Type Definition (RTD) Technical Specification. Technical report, NFC Forum Inc., 2006.

[36] NFC Forum Inc. Essentials for Successful NFC Mobile Ecosystems. Technical report, NFC Forum Inc., 2008.

[37] Oracle Inc. JSR 338: Java Persistence 2.1. *The Java Community Process*, 2013.

[38] Oracle Inc. JSR 344: JavaServer Faces 2.2. *The Java Community Process*, 2013.

[39] Red Hat Inc. JSR 303: Bean Validation. *The Java Community Process*, 2009.

[40] Sun Microsystems Inc. JSR 268: Java Smart Card I/O API. *The Java Community Process*, 2006.

[41] ISO/IEC. Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification. Norm ISO/IEC 18004:2006, International Organization for Standardization, 2006.

[42] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication Interface and Protocol -2 (NFCIP-2). Norm ISO/IEC 21481:2012, International Organization for Standardization, 2012.

[43] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1). Norm ISO/IEC 18092:2013, International Organization for Standardization, 2013.

[44] ISO/IEC/IEEE. Systems and software engineering - vocabulary. *ISO/IEC/IEEE 24765:2010*, 2010.

[45] Rod Johnson. *Expert one-on-one J2EE design and development*. John Wiley & Sons, 2002.

[46] Antero Juntunen, Sakari Luukkainen, and Virpi Kristiina Tuunainen. Deploying nfc technology for mobile ticketing services - identification of critical business model issues. In *2010 Ninth International Conference on Mobile Business / 2010 Ninth Global Mobility Roundtable*, 2010.

[47] Ravi Kalakota and Marcia Robinson. *E-Business: Roadmap for Success*. Addison-Wesley, 1999.

[48] Ravi Kalakota and Marcia Robinson. Electronic commerce. In *Concise Encyclopedia of Computer Science*. John Wiley & Sons, 2004.

[49] Gavin King and Christian Bauer. *Java Persistence with Hibernate*. Manning, 2006.

[50] Daniel E. Krutz and Andrew Meneely. Teaching web engineering using a project component. In *Frontiers in Education Conference (FIE), 2013*, 2013.

[51] Josef Langer and Michael Roland. *Anwendungen und Technik von Near Field Communication (NFC)*. Springer Berlin Heidelberg, 2010.

[52] Legion of the Bouncy Castle Inc. bouncycastle.org. `https://www.bouncycastle.org/`, 2014. Accessed: 23/07/2014.

[53] Niina Mallat, Matti Rossi, and Virpi Kristiina Tuunainen. Mobile banking services. *Communications of the ACM*, 47(5), 2004.

[54] Sunil S. Manvi, Lokesh B. Bhajantri, and M. A. Vijayakumar. Secure mobile payment system in wireless environment. In *International Conference on Future Computer and Communication, 2009*, 2009.

[55] Rainer Mautz. Overview of current indoor positioning systems. *Geodezija ir Kartografija*, 35(1), 2009.

[56] Microsoft Corp. The Smartphone Reinvented Around You | Windows Phone (United States). http://www.windowsphone.com/en-us, 2014. Accessed: 23/07/2014.

[57] Mozilla Foundation. Firefox OS - Mozilla | MDN. https://developer.mozilla.org/en-US/Firefox_OS, 2014. Accessed: 23/07/2014.

[58] Jef Neefs, Frederik Schrooyen, Jeroen Doggen, and Karel Renckens. Paper ticketing vs. electronic ticketing based on off-line system 'tapango'. In *Second International Workshop on Near Field Communication (NFC), 2010*, 2010.

[59] PKWARE Inc. APPNOTE.TXT - .ZIP File Format Specification. http://www.pkware.com/documents/casestudies/APPNOTE.TXT, 2012. Accessed: 23/07/2014.

[60] Ronald Linn Rivest, Adi Shamir, and Leonard Max Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), 1978.

[61] Rosanak Roshandel, Jeff Gilles, and Richard LeBlanc. Using community-based projects in software engineering education. In *24th IEEE-CS Conference on Software Engineering Education and Training (CSEET), 2011*, 2011.

[62] Juan J. Sánchez-Silos, Francisco J. Velasco-Arjona, Irene Luque Ruiz, and Miguel Ángel Gómez-Nieto. An nfc-based solution for discount and loyalty mobile coupons. In *4th International Workshop on Near Field Communication (NFC), 2012*, 2012.

[63] Paul Gerhardt Schierz, Oliver Schilke, and Bernd W. Wirtz. Understanding consumer acceptance of mobile payment services: An empirical analysis. *Electronic Commerce Research and Applications*, 9(3), 2010.

[64] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption*. Springer Berlin Heidelberg, 1994.

[65] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2002.

[66] Mark Sherriff. Teaching web services and service-oriented architecture using mobile platforms. In *Frontiers in Education Conference (FIE), 2010*, 2010.

[67] Jian Song and Zhaoyang Dong. Influence factors and development tendency on electronic commerce. In *The 2nd IEEE International Conference on Information Management and Engineering (ICIME), 2010*, 2010.

[68] Kun Tian, Kendra Cooper, and Kang Zhang. Improving software engineering education through enhanced practical experiences. In *IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS), 2011*, 2011.

[69] Roy Want. Near field communication. *Pervasive Computing, IEEE*, 10(3), 2011.

[70] Rainer Widmann, Stefan Grünberger, Burkhard Stadlmann, and Josef Langer. System integration of nfc ticketing into an existing public transport infrastructure. In *4th International Workshop on Near Field Communication (NFC), 2012*, 2012.

APPENDIX A

Functional Requirements and Usage Scenario Descriptions

This appendix lists and describes the functional requirements and indented usage scenarios of the presented mobile ticketing system from chapter 4. The described applications are

- the ticket server and the ticket server management application,
- the merchant application,
- the mobile client application and
- the mobile validation application.

Each requirement is presented by a title, a user story and a description.

A.1 Ticket Server and Ticket Server Management Application

The functional requirements of the ticketing server are based on the identified use cases by Widmann et al. [70] and Chaumette et. al. [14], which are described in section 3.1.2.3.

- **Register Merchant:**

  As a ticket server administrator I'd like to register a new merchant on the ticket server, so that the merchant can utilize the ticket server to manage his services and tickets.

  A ticket server administrator should be able to register new merchants. He enters the merchant's details. With the registration, the username and password are created, which authenticates the created merchant. The ticket server administrator gives the credentials

to the merchant. After the merchant was registered, he is stored in the system, can authenticate himself with his username and password and can manage his services and tickets.

- **Reset Merchant's Password:**

  As a ticket server administrator I'd like to reset the merchant's password, so that the merchant can authenticate himself again.

  The ticket server administrator should be able to reset the merchant's password. So, a merchant gets a new password, if the old one was compromised or lost. The merchant can authenticate himself with the new password again.

- **Register Service:**

  As a merchant I'd like to register a service on the ticket server, so that I can offer it to my customers and let the ticket process being handled by the ticket server.

  A merchant should be able to create new services. A service is registered by entering it's description and details. After the service registration process, the service is stored on the ticket server and can be managed by the merchant through the provided interfaces. For a new generated service, the ticket issuing isn't possible until the merchant opens the service.

- **Open Service:**

  As a merchant I'd like to open a service on the ticket server, so that tickets can be bought and validated for it.

  A merchant should be able to open a new or canceled service. When the service is opened, tickets can be bought by the users. Bought tickets can be validated.

- **Cancel Service:**

  As a merchant I'd like to cancel a service, so that no more tickets can be bought or validated for it.

  A merchant should be able to cancel a service, if it isn't available any more or if it is canceled. By canceling the service, no more tickets may be bought for it. Customers, who already posses tickets, shall be able to return them. Tickets are valid till the user returns them. A canceled service can be reopened again.

- **Get all available Services for a Merchant:**

  As a merchant I'd like to retrieve all my available services, so that I can manage them.

A merchant should be able to retrieve all of his services. The merchant can view the details for each service, can open it or cancel it. He may also view all tickets, along with their state, of it.

- **Get all Tickets for a Service:**

  As a merchant I'd like to retrieve all tickets along with their state, so that I can offer them to my customers.

  A merchant should be able to retrieve all tickets with their according state for any of his services. He can offer available tickets to his customers.

- **Generate Tickets for a Service:**

  As a merchant I'd like to add tickets to my services, so that my customers can acquire them.

  A merchant shall be able to add tickets to his services. For this purpose he chooses the service and the ticket type. It should be possible to generate many tickets at once. Generated tickets must be verifiable.

- **Acquire signed Ticket:**

  As a merchant I'd like to acquire a ticket, so that I can sell it to a customer, who can use it later. The ticket should be already signed so that it's validity can be checked.

  A merchant should be able to sell his tickets. If a customer buys a ticket from the merchant, the merchant acquires the ticket from the ticket server and gives it to his customer. The ticket is signed by the ticket server and the signature is attached to the ticket. With the signature and the public key, the authenticity of the ticket can be validated. The status of the ticket is updated on the ticket server and the user can use the ticket.

- **Cancel Ticket:**

  As a merchant I'd like to cancel a previously acquired ticket, so that I can prohibit the use of it.

  A merchant should be able to prohibit the usage of a ticket. For this purpose the merchant cancels the ticket, which usage should be prohibited. A ticket becomes canceled if it is returned, stolen or lost. A reason for cancellation should be given to make this process traceable.

- **Retrieve Ticket Details:**

  As a merchant I'd like to retrieve the details of a ticket, so that I can check if it is acquired and view it's devaluation information.

The ticket merchant should be able to retrieve the details of any of his tickets. Along with the status, the devaluation information should be retrieved. Each redemption of a ticket should be traced along with the circumstances of it.

- **Create Ticket Validator Certificate**

  As a merchant I'd like to create ticket validator certificates, which I can give to my ticket validators, so that they can authenticate themselves on the ticket server to devaluate tickets.

  The ticket merchant should be able to generate and retrieve ticket validator certificates. These certificates are needed to validate and devaluate a ticket on the ticket server. The merchant gives these certificates to his ticket validators.

- **Invalidate Ticket Validator Certificate**

  As a merchant I'd like to invalidate a prior created ticket validator certificate, so that it can't be used for authentication any more.

  The ticket merchant should be able to invalidate a prior created ticket validator certificate. Invalidation is needed if a certificate is lost or compromised to prohibit authentication with it.

- **Receive Offline Validation Data:**

  As the ticket merchant's validator I'd like to retrieve offline validation data from the ticket server, so that I can validate a ticket offline.

  The ticket merchant's validator should be able to obtain offline validation data from the ticket server. With this data, he can validate a ticket in the first step offline. The offline validation data consists of the offline validation key and ticket identifiers, which are blacklisted.

- **Validate Ticket:**

  As a ticket merchant's validator I'd like to check the validity of a ticket, so that I can check if a customer is entitled to access a service.

  A ticket merchant's validator should be able to check the validity state of a ticket. The ticket's validity is checked by it's type and previous devaluation information.

- **Devaluate Ticket:**

  As a ticket merchant's validator I'd like to devaluate a ticket on the ticket server, so that a ticket can't be used beyond it's validity.

  A ticket merchant's validator should be able to devaluate a ticket. The devaluation along with it's circumstances is stored on the ticket server.

A.2 Merchant Application

This section lists generic and theater specific usage scenarios of a merchant application. For the theater specific scenarios is a single theater with different halls and performances assumed.

A.2.1 Generic Usage Scenarios for a Merchant Application

- **Create a new Service:**

  > *As a merchant I'd like to create a new service, so that I can sell tickets to my customers for it.*

  A merchant should be able to create services. The service description and details are managed in the merchant's application. The service is also created on the ticket server so that tickets, which belong to that service, can be managed.

- **Open Service:**

  > *As a merchant I'd like to open a service, so that tickets can be bought and validated for that service.*

  A merchant should be able to open a service. What happens in the merchant's application is domain dependent, but this request is also passed to the ticket server, so that tickets can be acquired and validated.

- **Cancel Service:**

  > *As a merchant I'd like to cancel a service, so that no more tickets can be bought and validated for that service.*

  A merchant should be able to cancel a service. What happens in the merchant's application is domain dependent, but this request is also passed to the ticket server, so that tickets can't be acquired and validated any more.

- **Edit Service:**

  > *As a merchant I'd like to edit a service, so that I can keep it up to date.*

  A merchant should be able to edit the details of a service. He chooses the service, updates the details, and saves it in the merchant application.

- **Add Tickets for a Service:**

  > *As a merchant I'd like to add tickets to my services, so that my customers can acquire them.*

A merchant should be able to generate tickets for a service from his merchant application. Generated tickets are saved on the ticket server and can be retrieved by the merchants application to offer them to the customers.

- **View Services:**

  As a ticket customer I'd like to view all available services of the ticket merchant, so that I know which services are offered.

  A customer should be able to view services of a merchant. For this purpose the merchant's application retrieves all offered services from the ticket server. The details of the services are stored in the merchant's application, but the information regarding the ticketing process and if the service is open comes from the ticket server. The customer can choose a service to view it's details and available tickets. From there he can choose to acquire a ticket.

- **View Tickets for a Service:**

  As a customer I'd like to view tickets for a service, along with their availability, so that I can buy tickets for a desired service.

  A customer should be able to view tickets, along with their availability for each offered service. Domain dependent data is retrieved from the merchant's application, information regarding the ticketing process is retrieved from the ticket server. With this data, the merchant application can present available tickets to the customer, who can acquire them. If a ticket is acquired by a customer, the ticket's availability is updated on the ticket server and the ticket is issued to the customer through the merchant's application.

- **Buy Ticket:**

  As a customer I'd like to buy tickets for services from a merchant, so that I can use them later.

  A customer should be able to buy tickets for a service from a merchant. If a customer acquires a ticket, the ticket is retrieved by the merchant's application and issued to the user. The availability of the ticket is updated on the ticket server.

- **Return Ticket:**

  As a customer I'd like to return a previously acquired ticket, so that I get refund.

  A ticket customer should be able to return a ticket. He gets a refund for the returned ticket. The ticket becomes invalid afterwards and can't be used anymore.

- **Mark Ticket as Invalid:**

  As a merchant I'd like to mark a previously sold ticket as invalid, so that I can prohibit the use of it.

  A merchant should be able to mark a previously sold ticket as invalid on the ticket server. The ticket is invalid afterwards and can't be used anymore.

- **View Ticket Details:**

  As a merchant I'd like to view the ticket details, so that I can check if it is acquired and view it's devaluation information.

  A merchant should be able to view the details of a ticket. That involves domain dependent data, which comes from the merchant's application and data about the ticketing process state, which comes from the ticket server.

- **Generate Ticket Validator Certificate:**

  As a merchant I'd like to give my ticket validators a certificate, so that they can authenticate themselves on the ticket server to devaluate tickets.

  A merchant should be able to generate ticket validator certificates. These certificates are generated on the ticket server and are retrieved by the merchant's application. From there, the merchant can give them to his ticket validators. The ticket validators use them to authenticate themselves on the ticket server.

A.2.2 Theater Usage Scenarios for a Merchant Application

- **Register Customer:**

  As a customer I'd like to register myself in the merchant's system, so that I can buy tickets for his performances.

  A ticket customer should be able to register himself in the merchants system. After the registration, the customer can buy tickets for performances. The merchant can keep track of the customers transactions and can use this information for business analysis and adoptions.

- **Performance Overview:**

  As a customer I'd like to have an overview of all available performances, so that I can choose one that I'd like to see.

  A customer should have an overview of all available performances. He can pick an performance to see the according details of it. In the next step he can buy tickets for the selected performance.

- **View Seating Chart:**

  As a customer I'd like to view a seating chart of a chosen performance, so that I can buy tickets for specific seats.

  A customer should be able to view a seating chart for a chosen performance. In this chart he sees, which seats are free or already sold. The customer can pick specific seats and buy tickets for them.

- **Acquire ticket:**

  As a customer I'd like to buy tickets for a performance, so that I can see it.

  A ticket customer should be able to buy tickets for a performance. For this purpose he chooses the desired seats from the seating chart and buys them. The tickets are sent via e-mail to him. He can import the tickets to his mobile client.

- **Return Ticket:**

  As a customer I'd like to return a previously acquired ticket, so that I get refund, if I can't see the show.

  A ticket customer should be able to return a ticket if he can't attend the performance. He gets a refund for the returned ticket. The ticket becomes invalid afterwards and can't be used anymore.

A.3 Mobile Client

- **Ticket Overview:**

  As a ticket user I'd like to have an overview of my obtained tickets, so that I know which tickets I possess.

  A ticket user should be able to see all of his obtained tickets. If he chooses a ticket, he should see the according details of it.

- **Ticket Detail View:**

  As a ticket user I'd like to see the details of a ticket, so that I know the details of the service which the ticket stands for.

  A ticket user should be able to see details of a ticket. These details may be the location where the service takes place, a seat number or other information.

- **Import Ticket:**

  As a ticket user I'd like to import a ticket in the mobile client, so that the ticket is managed by the client and can be used.

  A ticket user should be able to import a ticket in the mobile client, which handles it after import. If the user wants to redeem a ticket, the mobile client ensures the transfer to the mobile validation application.

- **Redeem Ticket:**

  As a ticket user I'd like to redeem a ticket, so that I can engage the service which the ticket stands for.

  A ticket user should be able to redeem a ticket. For this purpose he chooses the desired ticket. The ticket is prepared for the transfer to the mobile validation application. After the ticket was transferred and validated, the ticket is devaluated.

A.4 Mobile Validation Application

- **Add Ticket Validator Certificate:**

  As a merchant I'd like to add a ticket validator certificate to the mobile validation application, so that I can authenticate myself on the ticket server.

  The ticket merchant should be able add his certificate to the mobile validation application. With this certificate he can authenticate himself on the ticket server on every request. Each certificate is unique.

- **Receive Offline Validation Data:**

  As the ticket merchant's validator I'd like to retrieve offline validation data, so that I can validate a ticket offline.

  The ticket merchant's validator should be able to obtain offline validation data. With this data, he can validate a ticket in the first step offline. The offline validation data consists of the offline validation key and ticket identifiers, which are blacklisted.

- **Validate Ticket:**

  As the ticket merchant's validator I'd like to validate a ticket, so that I can grant the ticket possessor the service if he possess a valid ticket.

  The ticket merchant's validator should be able to validate a ticket. The validation is done in two steps. In the first step the ticket is validated with the offline validation data. If this check succeeds, the ticket is checked online with remote data on the ticket server. If the online check also succeeds, the ticket possessor gains access to the desired service.

APPENDIX B

User Stories of the Course Assignment

This appendix lists the user stories, which are given to students as stated in section 5.2.

> *As a shop assistant I'd like to authenticate myself in the system with my username and password. When I'm logged in I'd like to be able to log out again. If I enter a wrong password five times in a row I'd like my account being locked for security reasons.*
>
> *As a shop assistant I'd like to switch the language of the user interface between German and English.*
>
> *As a shop assistant I'd like to create new customers. A new customer should be saved permanently in the system.*
>
> *As a shop assistant I'd like to change data of existing customers. The changes should be saved permanently in the system.*
>
> *As a shop assistant I'd like to see company news directly after the login. The site should only present me news, which I haven't already read.*
>
> *As a shop assistant I'd like to see and read company news, which I have previously read.*
>
> *As a shop assistant I'd like to see the top ten events of a month in terms of the amount of sold tickets. The presentation should be done with charts. From there, I'd also like to sell or reserve tickets for these events.*
>
> *As a shop assistant I'd like to search for events of an artist. I'd like to search for an artist's firstname or lastname. All artists, who match the search criteria should be presented to me as a list. If I choose an artist, I'd like to see his events. From there, I'd like to sell or reserve tickets for a show of that artist.*

As a shop assistant I'd like to search for event locations. I'd like to search for the name, street, city, country or zipcode of the location. All event locations, which match the search criteria should be presented to me as a list. If I choose an event location, I'd like to see the shows of that event location. From there, I'd like to sell or reserve tickets for the presented shows.

As a shop assistant I'd like to search for events. I'd like to search for the description, the name, the type, the duration (with an tolerance of +/- 30 minutes) or the content. All events, which match the search criteria, should be presented to me as a list. If I choose an event, I'd like to see the belonging shows. From there, I'd like to sell or reserve tickets for the presented shows.

As a shop assistant I'd like to search for shows. I'd like to search for the date and time, the price (within a meaningful tolerance), the event or the event location. All shows, which match the search criteria, should be presented to me as a list. From there, I'd like to sell or reserve tickets for the presented shows.

As a shop assistant, I'd like to do a reservation for a show for a new, registered or anonymous customer. Hereby, I'd like to pick the seats from a graphical seating chart. I'd like to be able to pick multiple seats. After the successful reservation I'd like to see the reservation number.

As a shop assistant, I'd like to sell tickets for a show to a new, registered or anonymous customer. Hereby, I'd like to pick the seats from a graphical seating chart. I'd like to be able to pick multiple seats.

As a shop assistant I'd like to cancel previously reserved tickets. Hereby, the customer tells me either the reservation number or his name along with the name of the show, for which he has reserved tickets. The canceled seats should be marked as free afterwards.

As a shop assistant I'd like to cancel previously sold tickets. Hereby, the customer tells me the name of the show, for which he has bought the tickets. The canceled seats should be marked as free afterwards.

As a shop assistant I'd like to sell previously reserved tickets. For that, the customer should either tell me the reservation number or his name along with the show, for which he has tickets reserved. The reserved seats should be marked as sold afterwards. The customer should also be able to buy additional tickets or only a part of the reserved ones. Additional sold seats should be marked as sold and not sold seats should be marked as free afterwards.

As a shop assistant I'd like to see a graphical seating chart of a show. Hereby I'd like to see which seats are free, reserved or sold. From there, I'd like to reserve or sell tickets of that show.

As a shop assistant I'd like to sell merchandise articles to a new, registered or anonymous customer. I'd like to see available merchandise articles. The customer should be able to buy arbitrary amounts of different articles and should be able to choose from different types of payment.

As a shop assistant I'd like to see available premiums.

As a shop assistant I'd like to see the available bonus points of a registered customer.

As a shop assistant I'd like to exchange bonus points of a registered customer for an available premium.

APPENDIX C

Evaluation Survey

This appendix gives the survey in German from chapter 6, as it was handed to the tutors. It also lists the uninterpreted filled out surveys as they were returned by the tutors. The results are discussed in section 6.2.

C.1 German Translation of the Survey

1. User Stories und Aufgabenstellung

    a) Hatten Studierende Schwierigkeiten mit dem Verständnis der zur Verfügung gestellten User Stories oder der Aufgabenstellung? Wenn ja, welche?

    b) Wurden Anpassungen bei den User Stories oder der Aufgabenstellung vereinbart? Wenn ja, welche?

    c) Wurden Fragen betreffend der User Stories oder der Aufgabenstellung von den Studierenden gestellt? Wenn ja, welche?

2. Zur Verfügung gestellter Code und Datenmodell

    a) Hatten Studierende gröbere Schwierigkeiten bei der Einarbeitung in den zur Verfügung gestellten Code? Wenn ja, welche?

    b) Wurden Anpassungen bei dem zur Verfügung gestellten Datenbankmodell durchgeführt? Wenn ja, welche?

    c) Wurden Codefragmente in dem zur Verfügung gestellten Code von einer Mehrheit der Studierenden als Fehlerhaft empfunden? Wenn ja, welche?

    d) Wurden Funktionen oder ganze Artefakte in den zur Verfügung gestellten Artefakten von den Studierenden vermisst? Wenn ja, welche?

3. Architekturvorgabe

    a) Hatten Studierende Schwierigkeiten mit der Architekturvorgabe oder traten Missverständnisse auf? Wenn ja, welche?

    b) Wurden Anpassungen bei der Architekturvorgabe vereinbart? Wenn ja, welche?

    c) Wurden Fragen betreffend der Architekturvorgabe von den Studierenden gestellt? Wenn ja, welche?

    d) Wurden Architekturverletzungen in der Umsetzung festgestellt? Wenn ja, welche?

4. Einsatz von Frameworks und Implementierung

    a) Verstanden Studierende den Sinn hinter dem Einsatz von Softwareframeworks? Falls nicht, worin bestanden die Unklarheiten?

    b) Wurden Probleme in dem Einsatz der Frameworks festgestellt? Wenn ja, welche und wie wurden sie gelöst?

5. Projektergebnis

    a) Konnten die Studierenden ihre Projekte erfolgreich abschließen? Falls nicht, warum nicht?

    b) Entsprachen die fertiggestellten Projekte den vorgegebenen Anforderungen? Falls nicht, welche wurden nicht erfüllt?

C.2 Filled out Uninterpreted Surveys

C.2.1 Survey 1

1. User Stories und Aufgabenstellung

   a) Hatten Studierende Schwierigkeiten mit dem Verständnis der zur Verfügung gestellten User Stories oder der Aufgabenstellung? Wenn ja, welche?
   Die User Stories wurden gut verstanden. Es gab rein vom Verständnis keine Probleme.

   b) Wurden Anpassungen bei den User Stories oder der Aufgabenstellung vereinbart? Wenn ja, welche?
   Ja, bei einer extrem guten Gruppe:

   - *Nicht implementieren:*
     - *Merchandise-Store*
     - *Prämien*
   - *Stattdessen:*
     - *CI -> Team City mit Sonar*
     - *Postgres DB für produktive App, testen mit HSQL in memory*
     - *Automatisches Refresh bei Top 10 (Polling oder Server Push)*
     - *Load Test*
     - *Instant Search wo möglich*

   Sie haben das Logging auch mit AspectJ gelöst und einen extrem mächtigen Entity to DTO Converter geschrieben und das UI super gestylt. Auch weil sie CI machen wollten, wurde Ihnen der Merchandise-Store + die Prämien-User Stories erlassen.
   Bei der anderen (eher schlechteren) Gruppe:

   - *Alles implementieren und zusätzlich folgende Änderungen:*
     - *Postgres DB für produktive App, testen mit HSQL in memory*
     - *Automatisches Refresh bei Top 10 (Polling oder Server Push)*
     - *Instant Search wo möglich*

   Die Änderungen kamen zustande weil:

   - *Postgres DB häufig verwendet wird.*
   - *Instant Search cool ist und man sich Buttons ersparen kann.*
   - *Wenn der Screen länger offen ist, sich die Top 10 sonst nicht ändern und der Verkäufer nicht die aktuellen Daten hat.*

   c) Wurden Fragen betreffend der User Stories oder der Aufgabenstellung von den Studierenden gestellt? Wenn ja, welche?
   Wie setzt man den Merchandise Store am besten um?

   - *Soll man was am Client halten oder nicht (bzgl. Warenkorb)?*
   - *Wie geht man mit parallelen Zugriffen um?*

Wie soll das Lockout genau aussehen?

- *Wie soll das Entsperren passieren?*

Dürfen wir SVN verwenden? Viele hatten Probleme mit git, vor allem beim Mergen.

2. Zur Verfügung gestellter Code und Datenmodell

   a) Hatten Studierende gröbere Schwierigkeiten bei der Einarbeitung in den zur Verfügung gestellten Code? Wenn ja, welche?
   Nur technisch interessierte Studenten konnten sich in Hibernate einarbeiten. Sie waren mit den Annotations und den Query-Möglichkeiten überfordert.

   b) Wurden Anpassungen bei dem zur Verfügung gestellten Datenbankmodell durchgeführt? Wenn ja, welche?
   Beim Sitzplan gab es dort und da Änderungen. Manche haben die Sitze in Kategorien gruppiert, andere nur in Reihen.

   c) Wurden Codefragmente in dem zur Verfügung gestellten Code von einer Mehrheit der Studierenden als Fehlerhaft empfunden? Wenn ja, welche?
   Zur UI und Service Schicht gab es kein Feedback, sprich das hat schon so gepasst. Beim Datenbankmodell und Hibernate-Code gab es oft Probleme. Gerade das Fetching hat nicht gut funktioniert, da viel zu viel nachgeladen wurde, auch wenn man es auf LAZY gestellt hat.

   d) Wurden Funktionen oder ganze Artefakte in den zur Verfügung gestellten Artefakten von den Studierenden vermisst? Wenn ja, welche?
   Die gute Gruppe wollte keinen Log-Code in den Methoden haben. Deswegen hab sie es mit AspectJ gelöst. Ich weiß nicht ob Bean Validation drinnen war, wenn nicht, dann könnte man das noch reingeben. Joda-Time wurde auch verwendet.

3. Architekturvorgabe

   a) Hatten Studierende Schwierigkeiten mit der Architekturvorgabe oder traten Missverständnisse auf? Wenn ja, welche?
   Großteils keine Probleme, da die Implementierung von UI bis DB durchgezogen war. Mit Hibernate hatte jeder Probleme.

   b) Wurden Anpassungen bei der Architekturvorgabe vereinbart? Wenn ja, welche?
   DB: Postgres anstatt HSQL. Eigenes Logging-Modul mit AspectJ.

   c) Wurden Fragen betreffend der Architekturvorgabe von den Studierenden gestellt? Wenn ja, welche?
   Dürfen wir die Architektur verändern? Was dürfen wir verändern?

   d) Wurden Architekturverletzungen in der Umsetzung festgestellt? Wenn ja, welche?
   Da die Vorgabe recht gut war, hielten sich die Studenten an den Aufbau. Schichtenverletzungen gab es keine. Auch haben alle Gruppen einen Konverter geschrieben. Validiert wurde auf jeden Fall am Server, aber auch am Client. Insgesamt war die Umsetzung ganz ok.

4. Einsatz von Frameworks und Implementierung

    a) Verstanden Studierende den Sinn hinter dem Einsatz von Softwareframeworks? Falls nicht, worin bestanden die Unklarheiten?
    Stand nie zur Diskussion.

    b) Wurden Probleme in dem Einsatz der Frameworks festgestellt? Wenn ja, welche und wie wurden sie gelöst?
    Hibernate: n+1 und Lade Probleme – Query direkt abgesetzt
    Jackson: Loop-Auflösung bei zirkularen Verbindungen – da gibt es eine Annotation

5. Projektergebnis

    a) Konnten die Studierenden ihre Projekte erfolgreich abschließen? Falls nicht, warum nicht?
    In der extrem guten Gruppe konnte eine Person nicht abschließen, weil sie persönliche Probleme + Minderwertigkeitskomplexe hatte. In der anderen Gruppe konnte jeder abschließen.

    b) Entsprachen die fertiggestellten Projekte den vorgegebenen Anforderungen? Falls nicht, welche wurden nicht erfüllt?
    Es wurde alles wie vereinbart (mit obigen Änderungen) implementiert. Keine Gruppe hatte weniger User Stories umgesetzt als vereinbart.

C.2.2 Survey 2

1. User Stories und Aufgabenstellung

    a) Hatten Studierende Schwierigkeiten mit dem Verständnis der zur Verfügung gestellten User Stories oder der Aufgabenstellung? Wenn ja, welche?
    Wirkliche Verständnisprobleme gab es nie bei meinen bisher drei Gruppen. Es kamen lediglich Fragen, wie man die User Reaktivierung (bei Sperre nach fünfmaliger Falscheingabe) umsetzen soll und wie das Bonusprogramm aussehen soll. Ich würde als Userstory dazunehmen, dass Zahlungsinformation und Rechnung behandelt werden müssen. Das geht mMn. aus den bisherigen User Stories nicht klar hervor und eine Gruppe (von drei) hat das daher nicht umgesetzt, weil sie eben der Meinung waren, dass es kein „Muss“ ist, obwohl die Entities im vorgegeben Code bereits vorhanden waren.

    b) Wurden Anpassungen bei den User Stories oder der Aufgabenstellung vereinbart? Wenn ja, welche?
    Nein. Keine.

    c) Wurden Fragen betreffend der User Stories oder der Aufgabenstellung von den Studierenden gestellt? Wenn ja, welche?
    Wie bereits beschrieben kamen Fragen zur User-Reaktivierung und dem Bonusprogramm. Bei ersterem wurde meistens die Lösung über Admin-User gewählt, bei zweiterem eben verschiedene Ansätze, wobei ich es nicht schlecht finde, dass nicht alles klar vorgegeben ist, da man so auch seine eigenen Ideen einbringen kann und bisschen zum Kreativsein aufgefordert wird.

2. Zur Verfügung gestellter Code und Datenmodell

    a) Hatten Studierende gröbere Schwierigkeiten bei der Einarbeitung in den zur Verfügung gestellten Code? Wenn ja, welche?
    Die Aufgabe und Sinnhaftigkeit des Data-Generators war unklar.

    b) Wurden Anpassungen bei dem zur Verfügung gestellten Datenbankmodell durchgeführt? Wenn ja, welche?
    Eine von drei Gruppen hat das Datenbankmodell angepasst und komplett umgeändert.
    Anfangs schaut das vielleicht ein bisschen komisch aus, weil sie den Warenkorb auch persistieren, das macht aber bei deren Umsetzung Sinn, da Reservierungen über den Warenkorb laufen, d.h. beim Kaufen einer Reservierung nennt man keine Reservierungsnummer sondern die des Warenkorbs.
    Was vielleicht noch ein bisschen zu Verwirrung geführt hat, waren die vielen Felder bei manchen Attributen, weil die Studenten tlw. dachten, dass sie alle verwenden müssen

    c) Wurden Codefragmente in dem zur Verfügung gestellten Code von einer Mehrheit der Studierenden als Fehlerhaft empfunden? Wenn ja, welche?
    Nein.

d) Wurden Funktionen oder ganze Artefakte in den zur Verfügung gestellten Artefakten von den Studierenden vermisst? Wenn ja, welche?
Nein.

3. Architekturvorgabe

a) Hatten Studierende Schwierigkeiten mit der Architekturvorgabe oder traten Missverständnisse auf? Wenn ja, welche?
Nein.

b) Wurden Anpassungen bei der Architekturvorgabe vereinbart? Wenn ja, welche?
Nein.

c) Wurden Fragen betreffend der Architekturvorgabe von den Studierenden gestellt? Wenn ja, welche?
Klar gab es hin und wieder ein paar Fragen, aber keine an die ich mich noch erinnern könnte bzw. solche die großes Unverständnis gezeigt hätten.

d) Wurden Architekturverletzungen in der Umsetzung festgestellt? Wenn ja, welche?
Inputvalidierung wurde teilweise nur am Client durchgeführt(!). Die REST-Schnittstellen wurden nicht immer den HTTP-Methoden (GET, PUT, POST, DELETE) entsprechend definiert und/oder die URIs repräsentierten Aktionen anstelle von Ressourcen. Liegt mMn. hauptsächlich an fehlender Erfahrung und vielleicht wird darauf bei der Eingangspräsentation auch zu wenig Wert gelegt.

4. Einsatz von Frameworks und Implementierung

a) Verstanden Studierende den Sinn hinter dem Einsatz von Softwareframeworks? Falls nicht, worin bestanden die Unklarheiten?
Die Sinnhaftigkeit der Frameworks wurde so nie wirklich angesprochen, ich bin mir aber sicher, dass jeder der sich mit dem Projekt programmiertechnisch beschäftigt hat, den Sinn versteht. Es gab jedenfalls nie Fragen, die mir gezeigt hätten, dass der Sinn unklar ist.

b) Wurden Probleme in dem Einsatz der Frameworks festgestellt? Wenn ja, welche und wie wurden sie gelöst?
Nein, ich kann mich an keine ernsthaften Probleme bei meinen Gruppen erinnern.

5. Projektergebnis

a) Konnten die Studierenden ihre Projekte erfolgreich abschließen? Falls nicht, warum nicht?
Ja, drei von drei Gruppen.

b) Entsprachen die fertiggestellten Projekte den vorgegebenen Anforderungen? Falls nicht, welche wurden nicht erfüllt?
Ja, einzig Rechnung und Zahlungsmethode wurde bei einer Gruppe nicht behandelt (siehe 1.a).

C.2.3 Survey 3

1. User Stories und Aufgabenstellung

    a) Hatten Studierende Schwierigkeiten mit dem Verständnis der zur Verfügung gestellten User Stories oder der Aufgabenstellung? Wenn ja, welche?
    JA. Wie bzw wonach genau gesucht werden soll war nicht klar. Bzw. wie die Filter zu implementieren sind.

    b) Wurden Anpassungen bei den User Stories oder der Aufgabenstellung vereinbart? Wenn ja, welche?
    Nein

    c) Wurden Fragen betreffend der User Stories oder der Aufgabenstellung von den Studierenden gestellt? Wenn ja, welche?
    JA. Wonach soll gesucht werden können, bzw. welche Filter sollen wie kombinierbar sein. Ob es notwendig ist mehrere Filter miteinander zu kombinieren.

2. Zur Verfügung gestellter Code und Datenmodell

    a) Hatten Studierende gröbere Schwierigkeiten bei der Einarbeitung in den zur Verfügung gestellten Code? Wenn ja, welche?
    Ja. Spring und Maven. Fehlermeldungen wahren schwer zu interpretieren.

    b) Wurden Anpassungen bei dem zur Verfügung gestellten Datenbankmodell durchgeführt? Wenn ja, welche?
    Ja. Nur für die Rechnung, hier wurde ein zusätzliches Feld hinzugefügt.

    c) Wurden Codefragmente in dem zur Verfügung gestellten Code von einer Mehrheit der Studierenden als Fehlerhaft empfunden? Wenn ja, welche?
    Nein.

    d) Wurden Funktionen oder ganze Artefakte in den zur Verfügung gestellten Artefakten von den Studierenden vermisst? Wenn ja, welche?
    Encode und Decode der URLs vom Server und Client.

3. Architekturvorgabe

    a) Hatten Studierende Schwierigkeiten mit der Architekturvorgabe oder traten Missverständnisse auf? Wenn ja, welche?
    Ja. Die Aggregation von Show und Ticket wurde als nicht richtig empfunden, sowie von Customer zu Receipt.

    b) Wurden Anpassungen bei der Architekturvorgabe vereinbart? Wenn ja, welche?
    Nein.

    c) Wurden Fragen betreffend der Architekturvorgabe von den Studierenden gestellt? Wenn ja, welche?
    Siehe Frage 3.a

d) Wurden Architekturverletzungen in der Umsetzung festgestellt? Wenn ja, welche?
Siehe Frage 3.a

4. Einsatz von Frameworks und Implementierung

a) Verstanden Studierende den Sinn hinter dem Einsatz von Softwareframeworks? Falls nicht, worin bestanden die Unklarheiten?
Ja.

b) Wurden Probleme in dem Einsatz der Frameworks festgestellt? Wenn ja, welche und wie wurden sie gelöst?
Maven. Zitat Student: "Wenn es funktioniert dann ist alles um einiges einfacher und schneller, wenn man ein kleines Problem hat, dann sucht man teilweise ewig bis man dieses gelöst hat. Auch nervt das andauernde clean package machen nach jeder noch so kleinen änderung. Teilweise hat maven aus undefinierbaren Gründen nicht funktioniert und nach einem eclipse neustart wieder funktioniert. Auch muss man bei den Spring Annotations sehr genau sein, da hier die Fehlermeldungen auch sehr nichtssagend sind und man ewig nach Fehlern sucht".

5. Projektergebnis

a) Konnten die Studierenden ihre Projekte erfolgreich abschließen? Falls nicht, warum nicht?
JA

b) Entsprachen die fertiggestellten Projekte den vorgegebenen Anforderungen? Falls nicht, welche wurden nicht erfüllt?
JA

C.2.4 Survey 4

1. User Stories und Aufgabenstellung

   a) Hatten Studierende Schwierigkeiten mit dem Verständnis der zur Verfügung gestellten User Stories oder der Aufgabenstellung? Wenn ja, welche?
   Nein.

   b) Wurden Anpassungen bei den User Stories oder der Aufgabenstellung vereinbart? Wenn ja, welche?
   Ja. Die Top10 sollten per Push-Notification vom Server aktualisiert werden, dafür musste die Merchandis User Storie nicht umgesetzt werden. Zusätzlich wurde auf eine PostgresDB umgestellt.

   c) Wurden Fragen betreffend der User Stories oder der Aufgabenstellung von den Studierenden gestellt? Wenn ja, welche?
   Der Zusammenhang zwischen Aufführungen und Veranstaltungen wird häufig nicht verstanden.

2. Zur Verfügung gestellter Code und Datenmodell

   a) Hatten Studierende gröbere Schwierigkeiten bei der Einarbeitung in den zur Verfügung gestellten Code? Wenn ja, welche?
   Der Zusammenhang zwischen Ticket und TicketIdentifier ist häufig unklar.

   b) Wurden Anpassungen bei dem zur Verfügung gestellten Datenbankmodell durchgeführt? Wenn ja, welche?
   Für das Prämiensystem, das Anzeigen der noch nicht gesehenen News und das Speeren des Users nach mehrmaligem fehlerhaften anmelden wurde die DB angepasst. Allerdings sind das ja Änderungen die gewünscht sind.

   c) Wurden Codefragmente in dem zur Verfügung gestellten Code von einer Mehrheit der Studierenden als Fehlerhaft empfunden? Wenn ja, welche?
   Nein.

   d) Wurden Funktionen oder ganze Artefakte in den zur Verfügung gestellten Artefakten von den Studierenden vermisst? Wenn ja, welche?
   Nein.

3. Architekturvorgabe

   a) Hatten Studierende Schwierigkeiten mit der Architekturvorgabe oder traten Missverständnisse auf? Wenn ja, welche?
   Häufig wird das REST-Konzept nicht richtig verstanden und daher die Schnittstellen eher Serviceorientiert definiert.

   b) Wurden Anpassungen bei der Architekturvorgabe vereinbart? Wenn ja, welche?
   Siehe 1.b.

c) Wurden Fragen betreffend der Architekturvorgabe von den Studierenden gestellt? Wenn ja, welche?
Nein.

d) Wurden Architekturverletzungen in der Umsetzung festgestellt? Wenn ja, welche?
Siehe Frage 3.a

4. Einsatz von Frameworks und Implementierung

a) Verstanden Studierende den Sinn hinter dem Einsatz von Softwareframeworks? Falls nicht, worin bestanden die Unklarheiten?
Eine Gruppe stellte den Sinn von hibernet bei dem Projekt in Frage. Sie hatten immer wieder Probleme mit detached Entities hatten. Lag aber wohl eher an der Gruppe ;)

b) Wurden Probleme in dem Einsatz der Frameworks festgestellt? Wenn ja, welche und wie wurden sie gelöst?
Siehe 4.a.

5. Projektergebnis

a) Konnten die Studierenden ihre Projekte erfolgreich abschließen? Falls nicht, warum nicht?
Es haben zumindest alle bestanden.

b) Entsprachen die fertiggestellten Projekte den vorgegebenen Anforderungen? Falls nicht, welche wurden nicht erfüllt?
Es wurde keine qualitativ gute Software erstellt. Validierungen fehlten, der Workflow schien nicht sonderlich gut durchdacht, die Änderungen an den User Sories wurden nur teilweise umgesetzt.

C.2.5 Survey 5

1. User Stories und Aufgabenstellung

   a) Hatten Studierende Schwierigkeiten mit dem Verständnis der zur Verfügung gestellten User Stories oder der Aufgabenstellung? Wenn ja, welche?
   Keine

   b) Wurden Anpassungen bei den User Stories oder der Aufgabenstellung vereinbart? Wenn ja, welche?
   Keine

   c) Wurden Fragen betreffend der User Stories oder der Aufgabenstellung von den Studierenden gestellt? Wenn ja, welche?
   Es wurde die Frage gestellt ob TL mit Concurrency (Zugriff von mehreren Clients gleichzeitig) umgehen können muss -> nein, wird im Rahmen der LVA nicht benötigt

2. Zur Verfügung gestellter Code und Datenmodell

   a) Hatten Studierende gröbere Schwierigkeiten bei der Einarbeitung in den zur Verfügung gestellten Code? Wenn ja, welche?
   Es wurden keine Probleme geäußert

   b) Wurden Anpassungen bei dem zur Verfügung gestellten Datenbankmodell durchgeführt? Wenn ja, welche?
   Keine

   c) Wurden Codefragmente in dem zur Verfügung gestellten Code von einer Mehrheit der Studierenden als Fehlerhaft empfunden? Wenn ja, welche?
   Keine

   d) Wurden Funktionen oder ganze Artefakte in den zur Verfügung gestellten Artefakten von den Studierenden vermisst? Wenn ja, welche?
   Keine

3. Architekturvorgabe

   a) Hatten Studierende Schwierigkeiten mit der Architekturvorgabe oder traten Missverständnisse auf? Wenn ja, welche?
   Der TA einer Gruppe war anfangs der Meinung, dass es auch beim Client eine Service-Schicht (zwischen UI und REST) gibt, welche jedoch im Codegerüst nicht vorhanden ist. Ihm wurde erklärt dass die REST-Schicht quasi die Service-Schicht darstellt.

   b) Wurden Anpassungen bei der Architekturvorgabe vereinbart? Wenn ja, welche?
   Keine

   c) Wurden Fragen betreffend der Architekturvorgabe von den Studierenden gestellt? Wenn ja, welche?
   Diverse Verständnisfragen (Unterscheid zwischen Entitiies und DTOs, Unit / Integrationstests / Mocking, . . .)

d) Wurden Architekturverletzungen in der Umsetzung festgestellt? Wenn ja, welche?
Keine

4. Einsatz von Frameworks und Implementierung

a) Verstanden Studierende den Sinn hinter dem Einsatz von Softwareframeworks? Falls nicht, worin bestanden die Unklarheiten?
Keine Unklarheiten

b) Wurden Probleme in dem Einsatz der Frameworks festgestellt? Wenn ja, welche und wie wurden sie gelöst?
Keine Probleme, nur lange Einarbeitungszeit

5. Projektergebnis

a) Konnten die Studierenden ihre Projekte erfolgreich abschließen? Falls nicht, warum nicht?
Alle erfolgreich

b) Entsprachen die fertiggestellten Projekte den vorgegebenen Anforderungen? Falls nicht, welche wurden nicht erfüllt?
Entsprachen bist auf kleine Mängel (Bugs, Usability) den Anforderungen