SCORE 2011

QR Marks the Spot

Codename: “QueueR”

Author:
Janosch Henze
Stefan Lindel
Sascha Müller
Ingo Witzky

Table of Contents:

0.0 Contacts

0.1 Stakeholder:

 Matteo Rossi of Politecnico di Milano http://home.dei.polimi.it/rossi/ (rossi@elet.polimi.it)

 Michal Young of University of Oregon http://ix.cs.uoregon.edu/~michal/ (michal@cs.uoregon.edu)

The contact persons

 Igor Cavrak (igor.cavrak@fer.hr)

 Marin Orlic (marin.orlic@fer.hr)

 Ivica Crnkovic Mälardalen University, Sweden http://www.mrtc.mdh.se/~icc/ (ivica.crnkovic@mdh.se)

0.2 Development Team:

Scrum Master: [AS] Andreas Scharf - andreas.scharf@cs.uni-kassel.de
Product Owner: [IW] Ingo Witzky - ingo.witzky@cs.uni-kassel.de

Team:
[JH] Janosch Henze - janosch.henze@googlemail.com
[SL] Stefan Lindel - stefan.lindel@cs.uni-kassel.de
[SM] Sascha Müller - sascha.mueller@cs.uni-kassel.de

1.0 Introduction

The Score contest will be held annually as part of the International Conference on Software Engineering. There are lots of projects offered, were teams of 3 to 7 students can take part to show and improve their skills on software engineering. Besides QR Marks the Spot there are a lot of other interesting topics to work on (e.g. a Multi Platform Kanban Taskboard, which should aid developers in using the agile development method Kanban). Schematizing maps is another very interesting project, where students should develop a software to better create tube maps like the one of the London underground rail system.

The project that was most interesting to us was “QR Marks the Spot”, because we immediately saw the ability to create something special - something big. We had a whole lot of ideas in a mere seconds about cool games to create and play and felt like sharing our ideas with you and the world. The games we wanted to create should be different from almost all other online (browser based) games. Because we have the ability to mix different activities in these games (online and offline, outdoor and indoor, action and strategy, round based and realtime and even educational ones) this project catched our attention. Classic online games are mainly designed to be played from everywhere on the world, but all these games are online only, often there is not even a real “offline world” considered. On the other hand there exist many online platforms which give the user the possibility to share location based offline games like the well known Geocaching[1]. But they often don’t offer the ability to take part in running games, are “offline” only (the site is just used to get and add data) or offer just one game mode. The goal of QR Marks the Spot project is to create an online gaming platform, which should enable the user to create and share location based games easily. Because of these fact our team was very interested in this Project, which tries to combine those two aspects.

Our team consists of four students and a PhD. student from the Software Engineering Research Group at Kassel University. The tasks the students worked on have been chosen according to their skill levels on the specific topic, thus we ensured maximum efficiency and performance in every part of the project. The task of the PhD. student was to administer the students during the development process.

This report will give an overview on the topics we have worked on. In the following chapters 2.0 Requirement Analysis, 3.0 Management Plan and 4.0 Architectural Design we explain the background of our project, like stating of requirements, how we managed our team and the requirements and an architectural overview of our project. The final chapters Implementation, Testing and Validation and the Summary will finish the report with details on implementation, like used libraries and tools, a brief overview of methods we used to test and validate our product and finally a summary on what we have learned during the project an how we could improve ourselves.

[1] http://www.geocaching.com

2.0 Requirement analysis

Requirement analysis is a mechanism to determine the needs and conditions to meet for a new product considering conflicting requirements and beneficiaries for the users and stakeholders. In the following sub-chapter we first will give a problem statement, where we explain the problem, which had to be solved. After the problem statement, we will give an overview of our product backlog, which holds a more specific overview of the requirements we have extracted.

2.1 Problem statement

Many of the social networks (e.g. Facebook[2], Twitter[3], LinkedIn[4] etc.) have a lot of so called social games (FarmVille[5] for example is the #1 game on Facebook and played by almost 50 million people), which many users can play together.
Many of those games lack of a social character, because the games are played online and people playing do not interact that much together. Not like in normal board games where people have conversation with each other. Other social games like GeoCaching do have a social character but often lack of social networks to distribute knowledge online. When you start a game of Geocaching you mainly meet with others and go out for a walk to your next geolocation. Such location based games started to gain more participants. So variations of the Geocaching game have been developed, like QR Marks the Spot. QR Marks the Spot is a game where positions are marked with QR Codes, which can hold game relevant data. The goal of QR Marks the Spot is to create such an online platform. The platform should be able to let the user connect to different social platforms, like Facebook or Twitter to share achievements and tell friends to play along ("John just played 'CityTour Kassel' on www.queuer.org"). Also the players should be able to share their experiences via blogs or forums. As far as the gaming part is concerned the players should be able to create games around their location, such games should be based on templates, which should define the standard behaviour of the game. These templates for example contain the games that are being played using this template. Additionally there a so called game instances which indicate the game in particular.

2.2 Requirement specification

With the help of the problem statement and the project description of QR Marks the Spot we created our base requirements. They have been split in a backend, which contains the requirements of the server side of the platform, a frontend, which contains the requirements of the user interface, and an application for mobile devices (e.g. Android powered smartphones).

---

[2] http://www.facebook.com/
[3] http://www.twitter.com
[4] http://www.linkedin.com
[5] http://www.facebook.com/FarmVille

2.2.1 Product backlog:

2.2.1.1 Backend

- **Game management**
  The system should be able to
  - create games from templates.
  - edit current games.
  - manage several templates.
  - delete games.
  - let users play games.
  - let users leave a game.
  - keep track of current games.
  - update a game's status according to a given time.
- **User management**
  The system should be able to
  - add new users.
  - delete an existing user.
  - assign a standard role to a user.
  - edit existing user.
  - connect an user to social networks (like Twitter, Facebook, OpenID).
  - login an existing user.
- **Role management (e.g. administrator, user, moderator etc.)**
  The system should be able to
  - let users act according to their role.
  - assign a role to an user.
  - deny acts taken by a user according to his role.
- **QR code management**
  The system should be able to
  - keep track of QR-Codes, which exists in the database.
  - create QR-Codes.
  - identify QR-Codes.
  - assign QR-Codes to games.
  - assign games to QR-Codes.
- **Persistence of games and users (database)**
  The system should be able to
  - store game relevant data in a database.
  - store user relevant data in a database.
  - update data, which already exist in a database.
  - delete data, which already exist in a database.
- **Export game data for mobile devices**
  The system should be able to
  - export game relevant data to a mobile device (e.g. Android phone).

2.2.1.2 Frontend

- **The frontend should be able to communicate with the backend**
- **User management: create, edit and delete user accounts**

The frontend should be able to
    - let the user add accounts.
    - let the user delete his account.
    - let the user edit his account.
    - allow user with specific roles to edit other accounts.
    - allow user with specific roles to delete other accounts
- **User login (normal login, Twitter, Facebook, OpenID)**
  The frontend should be able to
    - let the user login with his account.
    - let the user login with an Twitter account.
    - let the user login with an Faccebook account.
    - let the user login with an OpenID account.
- **Create games from templates**
  The frontend should be able to
    - allow users to create games from templates.
    - allow users to download QR-Codes.
    - allow users to print QR-Codes.
    - allow users upload existing QR-Codes for reuse.
- **Search and filter games**
  The frontend should be able to
    - filter games according to defined rules.
    - allow users to search for games.
- **Download game for mobile devices**
  The frontend should be able to
    - allow the user to download game content to a mobile device.
- **Track and update game progress**
  The frontend should be able to
    - keep track of a current game.
    - auto update the content, when something on a game changes.
- **Fancy Web 2.0 stuff like Facebook "I like" buttons and Twitter connection (social networking)**
  The frontend should be able to
    - have Facebook “I like” buttons.
    - allow users to post their current game updates via Twitter.
    - allow users to post their current game updates via Facebook.

2.2.1.3 Application for mobile devices

For special devices e.g. Android phones or the iPhone, their should be a special app, which let the user take advantage of hardware existing in his phone. The mobile device should be able to do GPS tracking.
The application should

- be able to search for nearby games.
- be able to control the game flow.
- be able to track the game flow.
- have an easy to control user interface.
- be able to have special user interfaces for different games.
- be able to find nearby gamers.

- be able to activate the camera when a QR-Code is near.
- be able to play a notifcation sound when a QR-Code is near.
- be able to create a new QR-Code from GPS data.
- be able to store created QR-Codes in the database.

2.2.1.4 Games

During the development process we wanted to create several games and after a brainstorm session we had developed various ideas which we will explain here. Because of the big amount we decided to only implement 2 games for a final delivery. The other games are still in the requirements but marked as optional.

- **CityTour**
  Needed:
  - mobile devices with QR code reader
  - QR codes with text in it (maybe direction for next code...)
  - (optional) GPS for next code

  The game shall be able to
  - guide the player on a predefined way.
  - describe the building/places near the QR code.
  - give hints to the next QR code location.
  - decipher a specific code gained after visiting each QR code.
- **RPG**
  Needed:
  - mobile devices with QR code reader
  - a lot of QR codes

  The game shall be able to
  - update the character according to his stats.
  - let the player interact with NPCs (QR codes).
  - let the player fight monsters (QR codes).
  - consume items (QR codes).
  - collect items (QR codes).
  - fight other players (QR codes).
  - display the stats of the player.
  - display the items owned by the player.
  - display achievements.
  - manage several classes like mage, archer, swordsmen and monk.
- **Drugwars (optional)**
  Needed:
  - mobile devices with QR code reader
  - QR codes
  - extended location based service, e.g. you need to know nearest police stations or nearby stations

  The game shall be able to
  - track buildings like police stations or high crime areas.
  - let the user buy drugs from dealers (QR codes).
  - let the user sell drugs to NPCs (QR codes).
  - punish users for selling drugs near a police station.

    - give advantage to users buying or selling drugs in high crime areas.
- **Race (optional)**
  Needed:
    - mobile devices with QR code reader
    - QR codes with "keys" (keys may be hashes, numbers or even words...)
    - (optional) Internet access

  The game shall be able to
    - track the number of QR codes entered by a user.
    - end the game after one user successfully entered all QR codes.
    - end the game after a given time.
- **Territory wars (optional)**
  Needed:
    - mobile devices with QR code reader
    - GPS coordinates
    - Internet access

  The game shall be able to
    - track territories (QR codes) of each player.
    - display territories on a map with a specified radius.
    - let the user reclaim stolen territories.
    - prevent cheating e.g. via mobile device and GPS coordinates.
- **Treasure hunt (optional)**
  Needed:
    - mobile devices with QR code reader
    - (optional) GPS
    - Internet access

  The game shall be able to
    - lead the user down a track.
    - let the user solve puzzles online.
    - let the user post a “treasure” after finding the last QR code
    - let the creator of the game define the “treasure”

3.0 Management plan

During software development often common planning and development processes are used. We have chosen the process, which best suited our team and describe the results in the following subchapters.

3.1 Team Introduction

Our team consist of four students from University of Kassel, Germany. We’re all students of computer science with some programming skills.

Team members are as follows:

- [IW] Ingo Witzky - ingo.witzky@cs.uni-kassel.de
- [JH] Janosch Henze - janosch.henze@googlemail.com
- [SL] Stefan Lindel - stefan.lindel@cs.uni-kassel.de

- [SM] Sascha Müller - sascha.mueller@cs.uni-kassel.de

3.2 Team expertise

From the total of four members in our team, three already had some knowledge of web application development. One member already had some experience in Android application development. Nobody of our team had experience in database development, we only had theoretical knowledge of database development.

3.3 Team/Project Coordination

For team/self management we used Scrum[6]. Scrum is an often used methodology for agile software development. It gave us the ability to develop software in a - for us - natural and easy way without too much 'external[7]' management involved.

The roles were chosen according to the knowledge of the member. The job of the product owner was given to Ingo, because he already had experience doing this and knows the tools we use to track our time. Our Scrum Master is Andreas, he was chosen because he was already our advisor for this project and was not intended to code.
To track our scrum status we used Agilo[8] for trac[9]. Trac is a well known project tracking and management tool that can be extended with many plugins, e.g. Agilo. Agilo itself extends trac with a lot of Scrum-tools like 'burndown charts', a project roadmap, time management functionality and a more general project timeline.

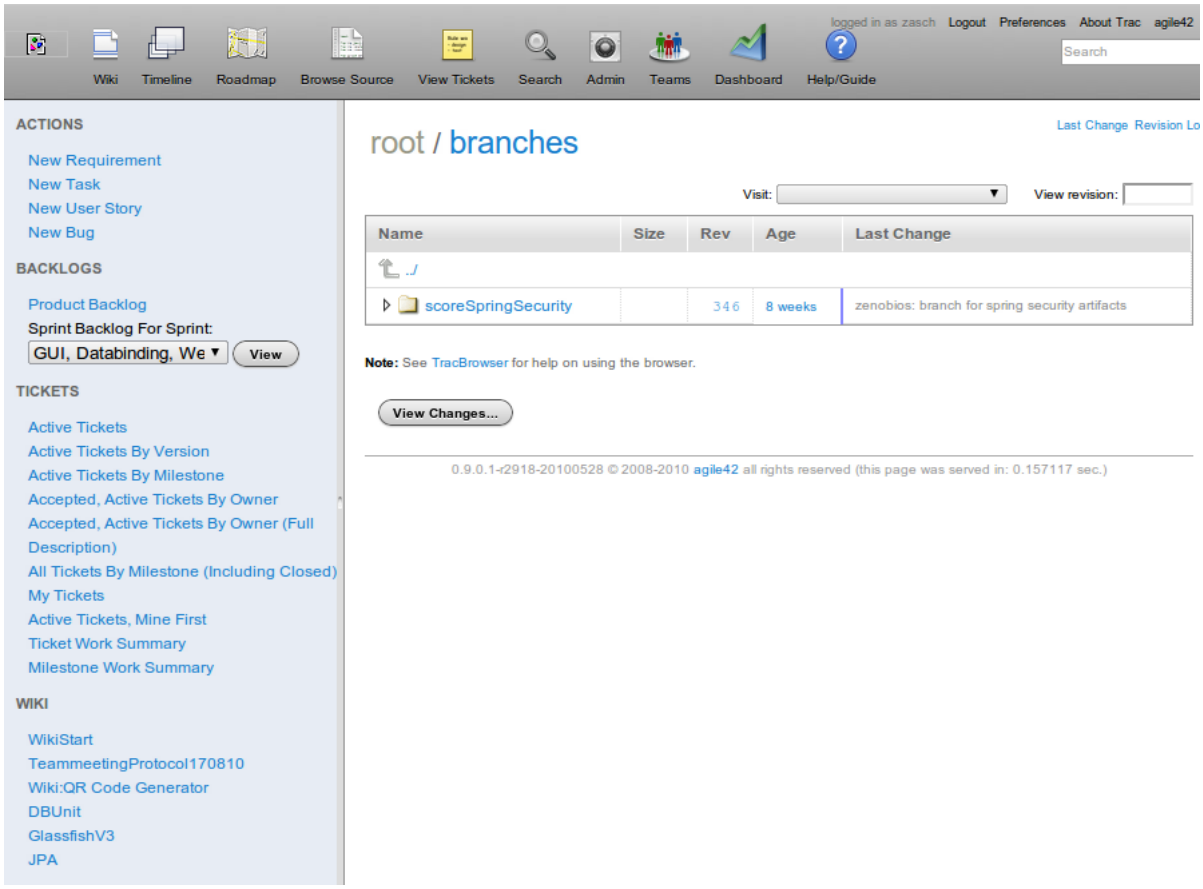
---

[6] http://en.wikipedia.org/wiki/Scrum_(development)
http://www.youtube.com/watch?v=Q5k7a9YEoUI

[7] External in this case means a person that does no developing but instead only manages the project/team.

[8] http://www.agile42.com/cms/pages/agilo/

[9] http://trac.edgewall.org/



(Screenshot of our Agilo for trac)

Especially burndown charts are very useful to the team as they indicate how much work is left till the end of the milestone (in Scrum this is called a sprint).
This document was created using Google docs[10]. This gave us an collaborative way to share and edit our document. Each team member could edit the document at the same time without messing up anything.

3.4 Source code management

To ensure that each member of our team develops on up to date source code we used SVN[11]. “SVN is an open source version control system” which helped us to easy distribute new source code to each team member. SVN also helped to reverse changes which may haved led to many errors later on.

3.5 Time management

Our team started working on the project in june 2010. We created milestones after working on requirements analysis in the first week. The milestones are used to keep an eye on the deadlines and project progress. They specify the development order because they depend on each other. Because we used Scrum as agile software development method we didn’t need a further defined timetable. Each milestone was split in smaller tasks which are

[10] http://docs.google.com
[11] http://subversion.apache.org/

processed by the team members.

We noticed that some used techniques did not work as expected after the first implementation. For fast future implementation of new game ideas it was intended to use the data model to describe many game types and use it on server and client side. Because of that the implementation of the persistence layer resulted in many problems. Lazy Loading[12] didn't work as expected because of referential integrity[13] which is used in the data model. This circumstance is responsible for our schedule delay. Therefore we decided to cancel or delay some milestones. The mobile devices website and the game template for the RPG game type were delayed to the final version.

Milstestones:
1. Scenarios and requirement analysis 06/30/10
Creating scenarios to evaluate the needed software features.
Studying the requirements to decide which technology to use.

2. Server infrastructure 07/06/10
Choosing a technology and setup a server with the needed software.

3. Initial web GUI + Android GUI 07/13/10
Web GUI: Implementation of a basic GUI with a user login and management.
Android: Implementation of a simple application which can read mobile device sensors and download a test file from the server.

4. Finish data model 07/20/10
Complete the finished design of the data model in Java.
The data model should be general as possible to describe many game ideas.

5. GUI, data binding, web service 07/27/10
Connection between GUI and data model. Integration of a web service to provide data for the web application.

6. Play the first game (CityTour) 09/06/10
Finish the implementation of creating a game, game search and starting a game. Play the game CityTour in mobile application.

7. Website for mobile devices (Groovy) 10/20/10
Implementation of a web service for mobile devices.

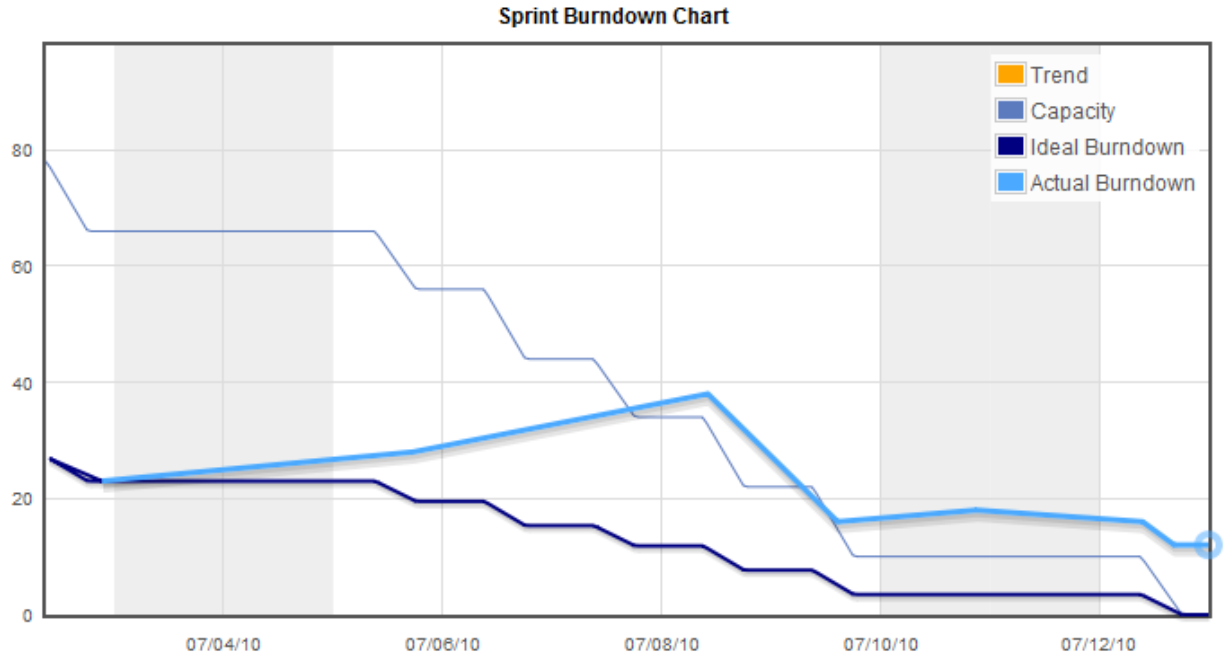
8. Play the first Version of RPG game12/24/10
Designing and implementing the RPG.

9. Final Release 01/30/10
Finish all open requirements.

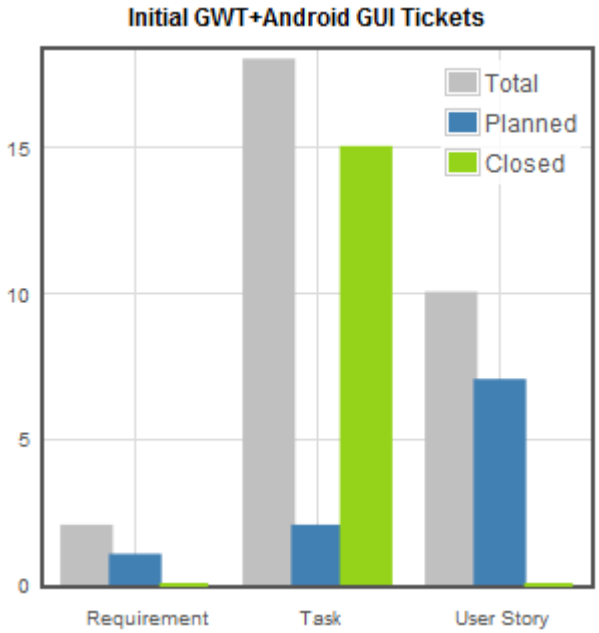
---

[12] http://en.wikipedia.org/wiki/Lazy_loading
http://wiki.asp.net/page.aspx/408/lazy-loading/
[13] http://de.wikipedia.org/wiki/Referentielle_Integrit%C3%A4t

10. Bug fixing 02/28/11
What the wording means. KILL THE BUGZ!



(Screenshot of our burndown chart extracted from Agilo)

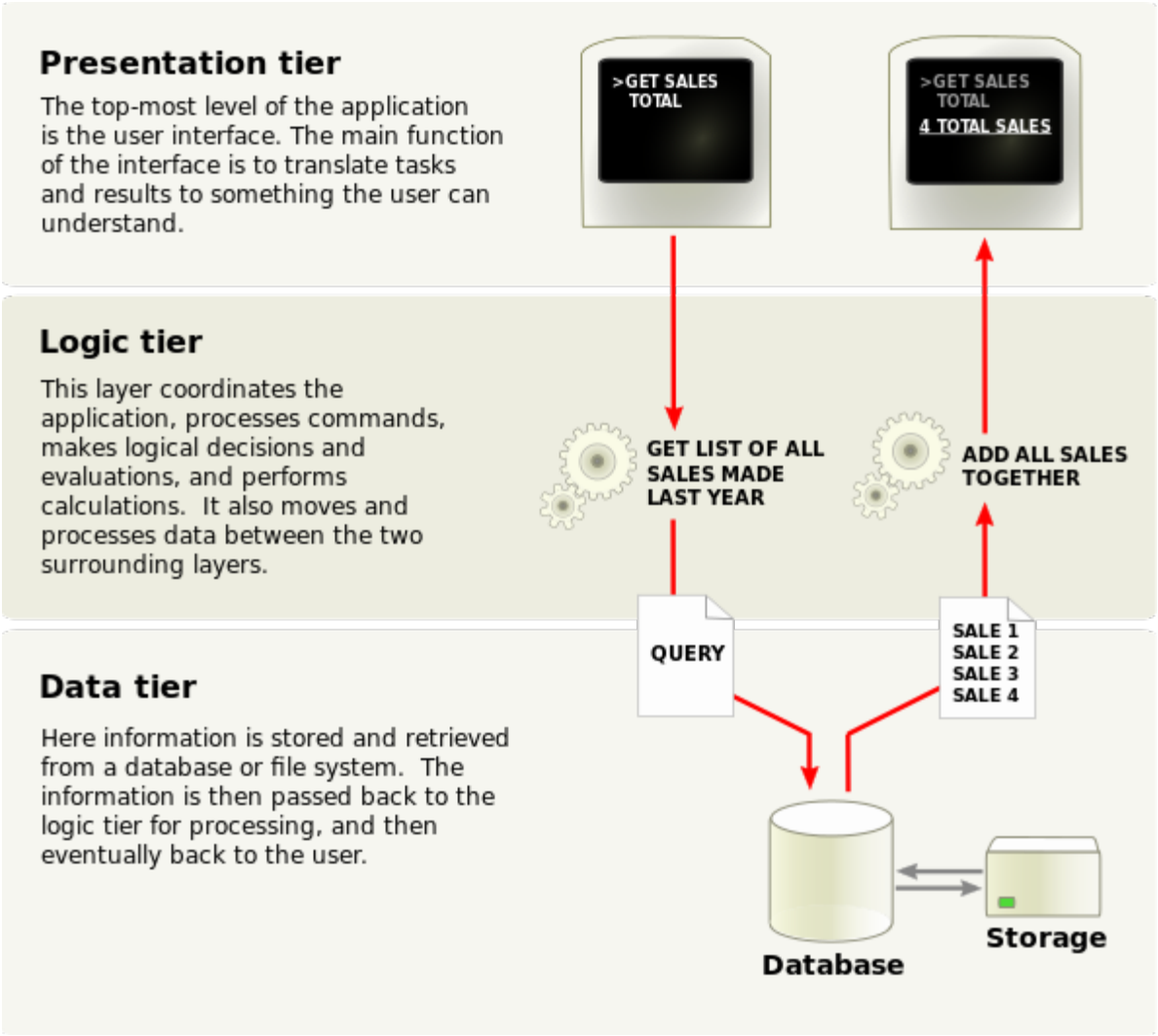


(Screenshot of our ticket overview extracted from Agilo)

4.0 Architectural design

Our architectural design depends on a classical 3-tier-architecture: The software is split up in the layers persistence, logic and presentation. The logic layer offers a defined interface for the presentation layer and the persistence layer has a defined interface for the logic layer.

So changes in the backend don't require frontend changes and vice versa. This architecture creates a clear distinction of responsibility. Every tier has its own area of responsibility and can only interact with the direct neighbour. That makes maintenance of the software easy and helps the developer quickly understand what each seperate part does.



(Source: http://en.wikipedia.org/wiki/3-tier_architecture)

4.1 Persistence layer

The persistence layer is responsible to save the data of the system. It holds relevant data for games like game instances, quests, users, game progress of each user and QR codes as well as all information required by the QueueR web application.
The Interface provided by the persistence layer is only used by the logic layer. So this layer can be on the same machine like the logic layer, but can also be on another one. This way load balancing[14] is easy and makes the application much more scalable.

4.2 Logic layer

The logic layer is the core of the QueueR service and communicates with the persistence

[14] http://en.wikipedia.org/wiki/Load_balancing_(computing)

layer over a standardized interface. This configuration makes it possible to exchange the persistence layer without change of the logic layer. Other services, like the presentation layer, can use the core to interact with the QueueR service through a given interface.

4.3 Presentation layer

This layer is the interface to the QueueR user. Everybody who wants to use our service interacts directly with the presentation layer. It is divided into several parts for usage with a desktop computer and mobile devices for the best viewing experience on each device. Each part has it's own service on the server side, but all of it use the same logical layer. This layer is used to display the relevant data to the user. It is one of the most - maybe *the* most - important part of the system.

4.3.1 Web application

The web application is part of the presentation layer and the main interaction method with our service for the user. The web application can be directly accessed with a web browser and an existing internet connection. A good layout of the application buttons and components allows the user to easy navigate in our application.
The web application allows the user to register himself and let him choose which services (e.g. Twitter, Facebook, OpenID) to connect with his account. After creating an account the user will be able to search for games near his location he'd like to play. To locate the user we use the W3C Geolocation API[15], which can be used in every browser currently on the market. If there aren't any interesting games near the users location, the user can create a game on his own. The application also allows the user to keep his friends updated with the help of Twitter or Facebook.

4.3.2 Mobile application

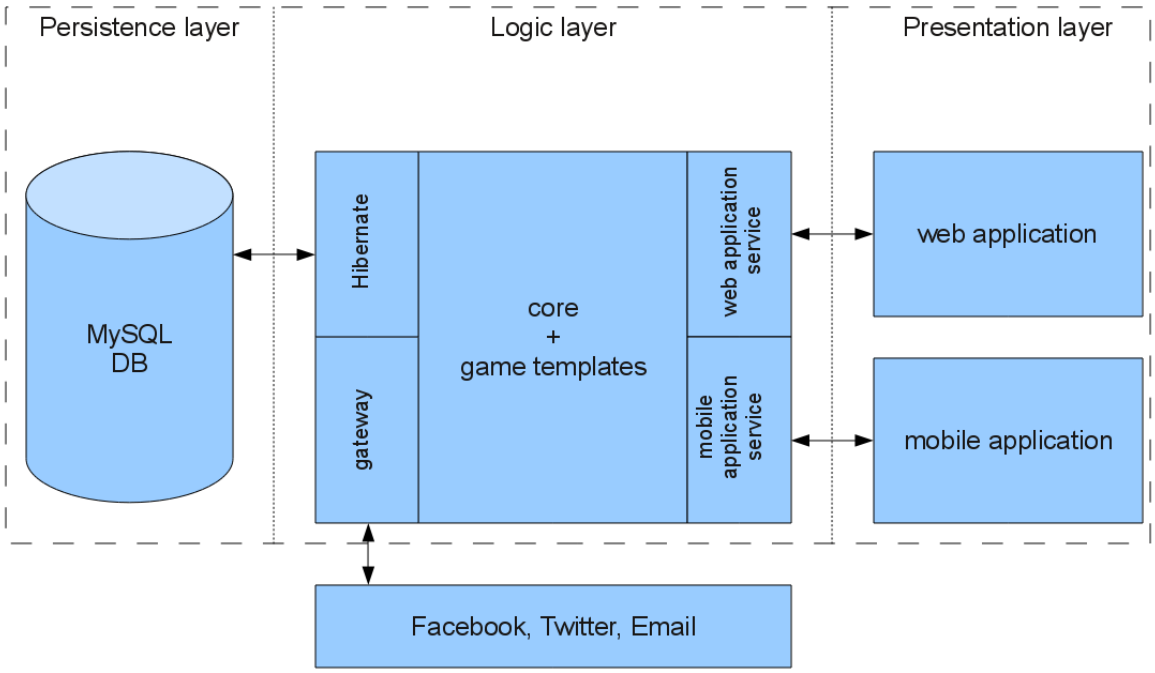
A mobile application specially designed for smartphones saves the user a lot of trouble, because web pages are often designed for standard web browser, so the smartphones often have problems displaying such sites. Therefore we decided to make such an application, so users of smartphones can easily interact with our services.
The app allows the user to download or copy game data in XML format with geo information of codes. These geo information makes it possible to play the games without a continuous connection to the internet. After a QR code has been found the built in QR code scanner allows the user to decipher the QR code and read the relevant informations.
Afterwards another mobile application allows the user to easily share his findings, e.g. at Twitter or Facebook.

5.0 Implementation

Almost the whole implementation is done in Java because it is platform independent, object orientated, modern and - last but not least - well documented. We used a lot of libraries to minimize coding work, deeply described for each tier.

---

[15] http://www.w3.org/TR/geolocation-API/



(Architectural design overview)

5.1 Persistence layer

For the persistence layer we use a MySQL RDBMS (Relational Database Management System). This RDBMS is highly scalable and easy to install, manage and backup. We used UML Lab[16] Modeling IDE for the data model. The JPA[17] (Java persistence API) Interface uses this model to create the database structure automatically. JPA is an abstraction layer and makes using database systems in Java relatively easy for developers, e.g. you don't have to write special code for, lets say, MySQL.

We use Hibernate[18] as JPA implementation. It's widely used in industrial environments and thus well tested, robust, stable and secure. We use it as an interface between our server logic and the RDBMS MySQL without the need of specially configuring our data model. This makes working with Hibernate relatively easy and straight forward.

As mentioned above, the data model was initially created using the UML Lab Modeling IDE. UML Lab is an IDE especially designed for modeling UML diagrams and based on Eclipse[19]. It also supports code generation for Java and JPA as well as reverse engineering and some other advanced features.

---

[16] http://www.uml-lab.com/en/uml-lab/
[17] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
[18] http://www.hibernate.org/
[19] http://www.eclipse.org

The data model was designed to be as generic as possible to be able to create lots of different game types without the need of changing the data model in any way. To model our data model we used “treasure hunt” and “RPG” as examples.

Please see the appendix for the complete class diagram of our model.

5.2 Server logic

The server logic exists of several self made classes. This classes exists on the server side and have several interfaces, which are used by the presentation layer. Those interfaces allow the presentation layer to easily get or store data on the server.
The server logic has interfaces as well, which allow an easy communication with external systems, like Twitter, Facebook and E-Mail.
The server logic also uses defined interfaces of the persistence layer. Those interfaces were chosen, because a change on one component should not change all the other components.

Besides that the server logic contains the game templates. Those templates are used to simulate a started game. They’re used to trigger actions, when a QR code is entered via the web page or the mobile application.

5.3 Client side

The client side of our system is the most important part for the user. Users must work and interact with our system with these part.

5.3.1 Website/Webapplication

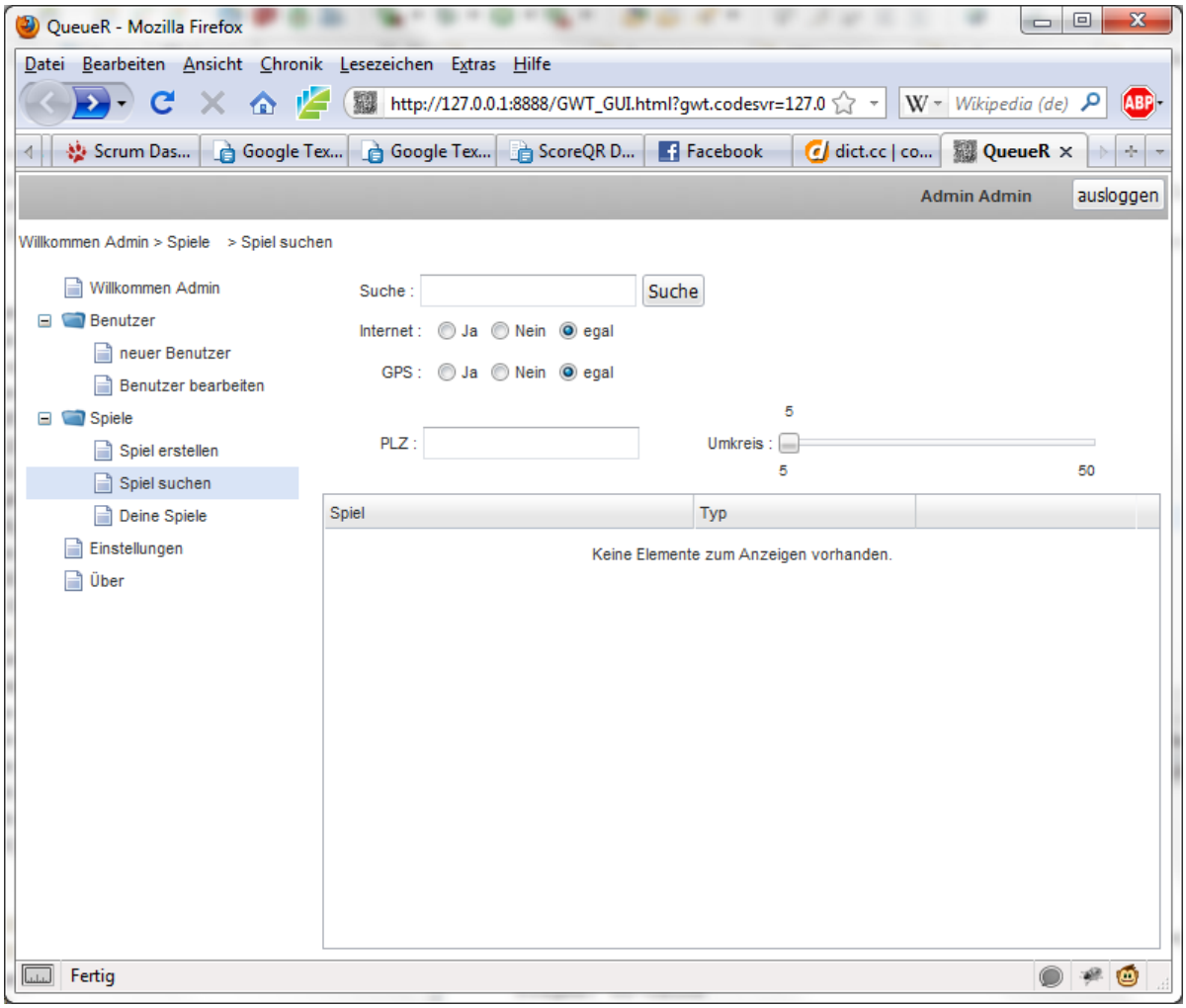
Our web application is build with the Google Web Toolkit[20] (GWT). This toolkit allows us to write the web application in Java and using dynamic techniques like AJAX[21]. GWT is a framework from Google to build dynamic web applications from Java which will be cross-compiled to Javascript[22] code and are runnable directly in a web browser. With this technique the programmer can work with his favorite programming language and doesn’t need to know about HTML and CSS. GWT also brings an easy interface (GWT RPC[23]) to communicate with the server side and with this advantage the programmer doesn’t need to know anything about Javascript but can use the advantage of AJAX anyway. With AJAX only the needed data will be transfered to the client, which saves traffic. It is also possible to compute more tasks inside the browser without the need to do server calls. With this advantage the server would be relieved.

There are lots of libraries with GUI elements available to extend the functionality of GWT. These libraries make it easy to build a web application. It is also possible to integrate interfaces to other APIs with such libraries into GWT. For example we use the library for the Google maps API to easily integrate maps and show the residence of a player and the

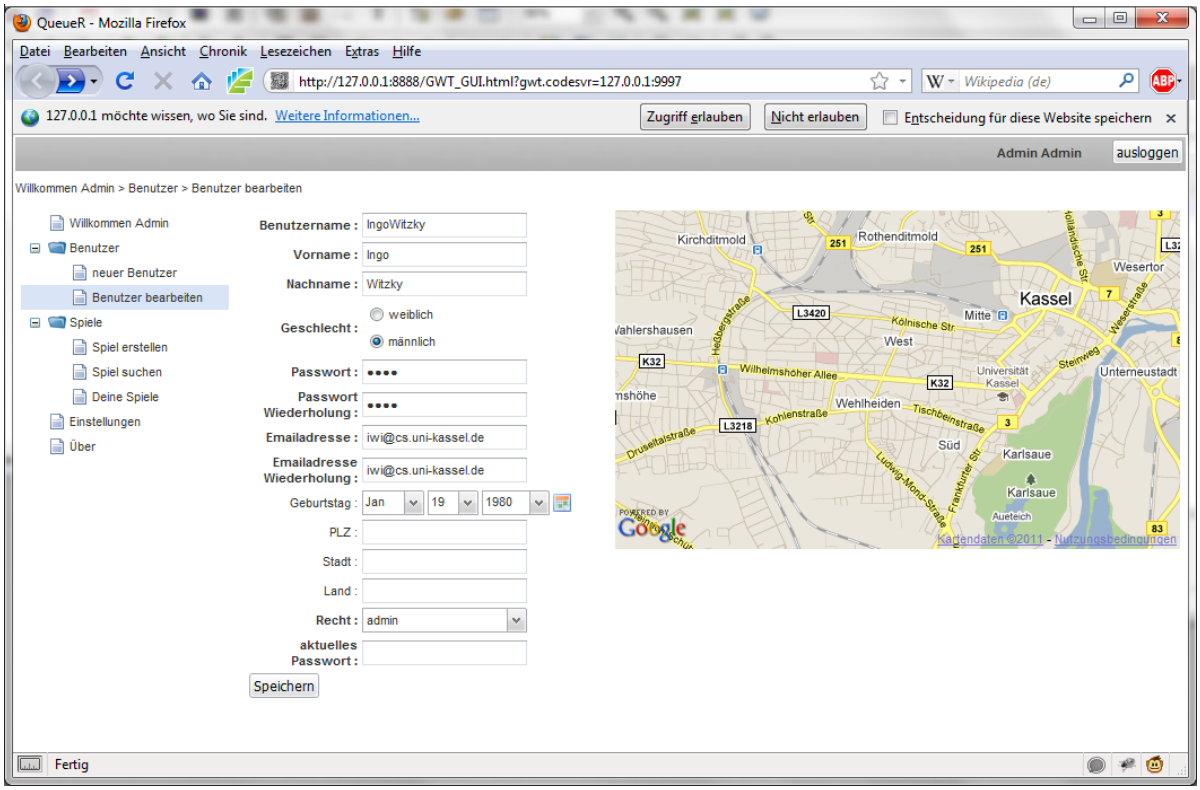
---

[20] http://code.google.com/intl/de-DE/webtoolkit/
[21] http://en.wikipedia.org/wiki/Ajax_(programming)
[22] http://en.wikipedia.org/wiki/JavaScript
[23] http://code.google.com/intl/de-DE/webtoolkit/doc/1.6/DevGuideServerCommunication.html

location of the QR codes.



(Search games using different criteria)



(Edit a user; Geolocation in action on the top)

5.3.2 Mobile application

The mobile application is also written in Java and can be used on Android powered smartphones. Through the XMLVM[24] Crosscompiler we're able to port this application to the Apple[25] iPhone without further configuration and development. RIM[26] BlackBerry phones also support Java applications, thus porting our app to this platform should be easy, too. So we could almost reach 60% (as seen below) of all mobile phone users.
The following part only contains information about Android and Android powered phones. A simple port of the application to iOS was made, but not all features were tested. A port to BlackBerry is planned but wasn't started yet.

Worldwide Smartphone Sales to End Users by Operating System in 3Q10 (Thousands of Units)

Company	3Q10 Units	3Q10 Market Share (%)	3Q09 Units	3Q09 Market Share (%)
Symbian	29,480.1	36.6	18,314.8	44.6
Android	20,500.0	25.5	1,424.5	3.5
iOS	13,484.4	16.7	7,040.4	17.1

[24] http://www.xmlvm.org/overview/
[25] http://www.apple.com
[26] http://www.rim.com/

Research In Motion (RIM)	11,908.3	14.8	8,522.7	20.7
Microsoft Windows Mobile	2,247.9	2.8	3,259.9	7.9
Linux	1,697.1	2.1	1,918.5	4.7
Other OS	1,214.8	1.5	612.5	1.5
Total	**80,532.6**	**100.0**	**41,093.3**	**100.0**

(Source: http://www.gartner.com/it/page.jsp?id=146631 (November 2010))

The application has a touchscreen optimized user interface which perfectly fits all screen resolutions. It supports the user in search of the next QR codes. It does this via the GPS module of the phone and a build in compass. With the help of Google Maps you can also navigate to the next QR code.
Scanning of the codes is relatively easy as well. The application uses the excellent ZXING[27] Barcode Scanner to extract the codes from the pictures taken with the phones camera. The Barcode Scanner as well as Google Maps are already build in. Our application uses the other programs APIsand integrates their techniques seamlessly into our application.

To avoid duplicate code the mobile application uses the same data model as the rest of the system. The app also has the ability to download and open a XML file of a game from the phones SD card. The XML file contains the positions and the identification of all QR codes for this particular game. It's also possible to 'save' a game on the phone, which means to alter the XML file and add new information (e.g. a "QR code found" flag).

[27] http://code.google.com/p/zxing/

(Screenshots of the Android client)

6.0 Testing and validation

We tried to automatically test our software with so called unit tests. Unit tests are used in the software engineering process to make sure everything works correctly and as expected. They are written before writing the productive code - this is called test-first-principle[28] - and can be run automatically, e.g. without user interaction. This reduces the amount of time needed for testing our application drastically and also ensures the quality of our product, because bugs can be spotted early in the development process. This process will also make sure that the modifications in the code will not break your system without your knowledge. For this purpose we use JUnit, a unit testing framework for Java. For further information see the JUnit homepage[29]. The following screenshot shows the coverage report for our model.

[28] http://en.wikipedia.org/wiki/Test-driven_development
[29] http://www.junit.org

EMMA Coverage Report (generated Fri Jan 14 20:50:57 CET 2011)

[all classes]

COVERAGE SUMMARY FOR PACKAGE [org.queuer.model]

name	class, %	method, %	block, %	line, %
org.queuer.model	100% (11/11)	79% (185/233)	58% (1954/3377)	59% (597,2/1020)

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
Item.java	100% (1/1)	10% (2/21)	16% (50/309)	13% (12/93)
Location.java	100% (1/1)	83% (5/6)	30% (17/56)	37% (7/19)
QRUser.java	100% (1/1)	92% (49/53)	49% (412/835)	56% (134/239)
PlayingGame.java	100% (1/1)	79% (15/19)	51% (145/282)	51% (42,9/84)
Player.java	100% (1/1)	78% (29/37)	56% (308/552)	54% (91,9/169)
QRCode.java	100% (1/1)	88% (14/16)	70% (163/234)	72% (56,9/79)
Actor.java	100% (1/1)	88% (7/8)	72% (73/101)	72% (26/36)
Quest.java	100% (1/1)	93% (13/14)	73% (146/200)	71% (46/65)
Identifier.java	100% (1/1)	78% (18/23)	74% (245/331)	74% (67/90)
GameInstance.java	100% (1/1)	89% (25/28)	81% (311/383)	76% (87,6/115)
GameEntity.java	100% (1/1)	100% (8/8)	89% (84/94)	84% (26/31)

[all classes]

EMMA 2.0.5312 EclEmma Fix 2 (C) Vladimir Roubtsov

(EclEMMA[30] coverage report of our base model classes)

Gui testing, on the other hand, isn't as easy as the above method. It can be achieved using so called mockup-techniques (for example see mockito[31] or, for SWT, SWTBot[32]), but it's easier to test the GUI by hand - a user looks at it and clicks on buttons etc. This also ensures a great amount of quality, because you don't really think (and test automatically) of everything that a user can do and maybe will do. So we let him just do it and see, if everything works as expected. Even some big companies have a fairly huge group of UI testers that do manual testing.

7.0 Summary and future work

The project was really interesting, but took a lot of effort to accomplish several tasks. Let's start at the beginning.
First thing we did was trying to team up with some guys over in Estonia at the University of Tartu. That took a while because we've never met them or heard of them before. After some conferences via Sykpe and emails they decided that they would need too much time for the project and quit. So it was just us four again. It took a lot of time and effort to evaluate the tools and libraries to use and how they work. So, after a rough start we actually started developing some stuff.
We've learned, that teaming up and management needs a lot of time. We didn't actually meet all the goals we set at the beginning, but creating a "City Tour" works, as well as all the user creation and login does. We also managed to get the database (tables, connection, relationships between tables and our object structure etc.) working and optimized our data model and general structure of our program.

[30] http://www.eclemma.org/

[31] http://www.mockito.org

[32] http://www.eclipse.org/swtbot

Some things could have been done better, especially the communication inside the team (Janosch for example hasn't been able to be at all our meetings and often was informed a little too late and so on). We also would have needed more team members to get all the work done, but maybe we just set too ambitious goals.
But, we do have a really robust data model, used some advanced tools like the UML Lab code generation and learned to work really well as a team.

But that wasn't all. Aside of the Score Contest a lot of future work is planned. QueueR will be continued at the Software Engineering Research Group at the University of Kassel and may be a good starting point for e.g. Spring Security[33], UiBinder for GWT[34] or making it more scalable via the Glassfish Application Servers advanced features like JDBC-Connection-Pooling[35]. There's a long list of other things to implement and work on, so stay tuned.

[33] http://static.springsource.org/spring-security/site/
[34] http://code.google.com/intl/de-DE/webtoolkit/doc/latest/DevGuideUiBinder.html
[35] http://en.wikipedia.org/wiki/Connection_pool