Supplementary Specification
Projekt BierIdee

Danilo Bargen, Christian Fässler, Jonas Furrer

3. Mai 2012

Inhaltsverzeichnis

1 Änderungshistorie

Version	**Datum**	**Änderung**	**Person**
v1	26.03.2012	Dokument erstellt	cfaessle
v1.1	31.03.2012	Korrekturen nach Review	jfurrer

2 Einleitung

Der Sinn und Inhalt dieses Dokumentes ist es, nicht-funktionale Anforderungen, die an das Projekt gestellt sind, festzuhalten. Diese beinhalten Qualitätskritereien nach FURPS+ aber auch weitere Anforderungen an Kategorien wie Hardware, Netzwerk, Internationalisierung oder Dokumentation.

3 Usability

3.1 Efficiency

Die Benutzbarkeit der Android App ist ein wichtiger Aspekt in diesem Projekt. Da sie auf mobilen Geräten hauptsächlich durch Toucheingaben bedient wird, wird die Applikation so entworfen, dass sie sich problemlos ausschliesslich mit Toucheingaben bedienen lässt.

3.2 Learnability

Bei Applikationen für Mobiltelefone ist es Erfolgsentscheidend, dass sie intuitiv zu bedienen und zugleich lernfördernd sind. Es wird daher bewusst auf eine Hilfestellung in Form einer Bedienungsanleitung für die Android App verzichtet, jegliche Hilfestellung sollte ausschliesslich über die integrierte Oberfläche geschehen. Die Sicherstellung dieses Merkmals geschieht durch regelmässige Tests und Bewertungen durch Testpersonen (siehe User Tests).

4 Performance

Im Rahmen dieses Projektes ist das System und die serverseitige Umgebung für den Einsatz mit bis zu 50 Benutzern, rund 100 Bieren und 20 Brauereien ausgelegt. Es sollte aber skalierbar sein, so dass mit besserer Serverumgebung erheblich mehr Benutzer bedient werden könnten.

4.1 Response Time

80% der Anfragen an das System sollen innerhalb von 3 Sekunden ausgeführt werden. Der Durchschnitt soll bei maximal einer Sekunde liegen. Dieser Wert ist gemäss Jakob Nielsen tief genug, damit der Gedankenfluss des Benutzers nicht unterbrochen wird [**?**].

4.2 Computing Resource

Die Architektur wird so ausgelegt, dass so wenig Rechenleistung wie möglich auf der Clientseite benötigt wird. Das heisst sämtliche Logik und Aufbereitung der Daten wird auf der Serverseite implementiert. Da clientseitig die Unterschiede in der verwendeten Hardware sehr gross sein können (Smartphones, Tablets, Netbooks), können keine einheitlichen Anforderungen zur Rechenkapazität auf den Geräten gestellt werden. Desweiteren muss heutzutage bei der Entwicklung von Mobile Apps stark auf den Energieverbrauch geachtet werden. Mit der zentralisierten Ausführung rechenintensiver Aufgaben sind die verfügbaren Ressourcen bekannt und die Software kann entsprechend adäquat entwickelt werden.

5 Scalability

Im Rahmen dieses Projektes können die benötigten leistungstechnischen Anforderungen an das System gut eingeschätzt werden, da dieses nur von Testusern und Entwicklern benutzt wird. Falls das Projekt weitergeführt wird, muss das System jedoch skalierbar sein. Die eingesetzten zentralen Software-Komponenten wie Restlet und PostgreSQL sind gut skalierbar [**?**].

6 Reliability

6.1 Security

Das System verwaltet unter anderem Benutzerdaten wie Email-Adressen oder konsumierte Getränke. Diese Benutzerdaten sind nur nach erfolgreicher Authentisierung durch berechtige Benutzer zugänglich. Die gespeicherten Benutzerdaten sind aber nicht als “heikel” anzusehen, da durch deren unabsichtliche Veröffentlichung keine finanzielle oder massgebliche persönliche Schäden entstehen würden. Aus diesem Grund werden keine zu erfüllende Anforderungen in Punkto Sicherheit (beispielsweise zu erfüllende Standards o.ä.) spezifiziert.

6.2 Massnahmen

Passwort Policy Admin-Passwörter müssen aus mindestens 8 alphanumerischen Zeichen bestehen.

Backup Das Serversystem und die Datenbank werden täglich gesichert (File Hot-Copy). Zusätzlich gibt es Snapshots der kompletten Serverinstanz, so dass

diese bei einem kompletten Systemausfall auf einen anderen VM Host umgezogen werden kann.

6.3 Availability

Da es sich bei diesem Projekt nicht um ein kritisches Projekt handelt, sonden um ein Projekt das zur Unterhaltung dient, welches bei Nichtverfügbarkeit keine finanziellen Schäden verursacht, werden keine konkreten Anforderungen an die Systemverfügbarkeit gestellt. Grundsätzlich soll das System aber dennoch möglichst durchgehend verfübar sein um so auch kontinuierliches Testen zu ermöglichen (siehe auch Abschnitt **??** zu Benutzertests). Es werden aber keine Massnahmen wie beispielsweise der Einsatz von redundanten Komponenten vorgesehen.

6.3.1 Schwachstellen / Risiken

Datenbank Server Das System besitzt eine zentrale Datenbank. Fällt diese Komponente aus, ist die Client Applikation nicht mehr benutzbar.

REST API Ist die Serverkomponente (REST API) nicht mehr verfügbar, ist die Client Applikation nicht mehr benutzbar.

Diese beiden Punkte sind im Moment 'Single Points of Failure', beide könnten jedoch in Zukunft mit wenig Aufwand redundant ausgelegt werden.

6.4 Fehlererkennung

Das System wird mittels Integrations- und Systemtests regelmässig auf Funktionalität und möglichst grosse Fehlerfreiheit getestet. Falls doch mal Fehler im Client-System auftreten sollten, wird der Fehler/Stacktrace aufgezeichnet, so dass man das Problem in Zusammenarbeit mit dem Benutzer möglichst einfach debuggen kann.

7 Supportability

7.1 Internationalization

Das System soll verschiedene Sprachen unterstützen. Lokalisierung wird hingegen nicht unterstützt. Das heisst, es werden keine unterschiedlichen Masseinheiten, Währungen oder dergleichen ermöglicht.

8 Design Requirements

8.1 Internal Design

Der Sourcecode sollte sauber und modular aufgebaut sein. Er soll möglichst lesbar sein und wo nötig kommentiert werden.

Zur Sicherstellung der Einhaltung der definierten Code Style Guidelines wird das Eclipse Plugin Checkstyle verwendet. Grundsätzlich orientieren wir uns an den Oracle Java Code Conventions (http://www.oracle.com/technetwork/java/codeconv-138413.html).

8.1.1 Android-App

Bei der Android-Entwicklung gelten die Hinweise und Patterns aus dem Buch *Programming Android* [**?**] aus dem O'Reilly Verlag.

8.1.2 API

Die API sollte möglichst konsequent ressourcenorientiert nach den REST-Richtlinien aus der bekannten Dissertation von Roy Thomas Fielding [**?**] aufgebaut werden. Wir orientieren uns auch an den Büchern *RESTful web services* [**?**] und *REST API Design Rulebook* [**?**] aus dem O'Reilly Verlag.

8.2 External Design

Das externe Design sollte ansprechend und einfach benutzbar sein. Wir fokusieren uns im Rahmen dieses Projektes jedoch auf die technische Umsetzung, deshalb setzen wir uns keine Vorgaben betreffend externem Design.

9 Implementation Requirements

9.1 Database

Für das System wird eine Relationale Datenbank mit JDBC Kompatibilität benötigt. In diesem Projekt wird PostgreSQL eingesetzt, welches diese Anforderungen erfüllt.

9.2 Programming Language

Als Programmiersprache wird für alle Teilsysteme (REST-API, Client) Java Version 1.6 verwendet.

9.3 Frameworks

Folgende Frameworks werden verwendet:

RESTlet Version 2.0.11 (stable)

Android SDK API Level 10, Android 2.3.3

10 Interfaces

10.1 Use Interface

Siehe Abschnitt Usability

10.2 Network

Der Client kommuniziert über eine HTTP Schnittstelle mit dem Server.

10.3 Software Interface

Die Schnittstelle zwischen Client und Serverkomponenten wird via REST API erfolgen.