# Projekt: BierIdee

# Projektplan

Danilo Bargen, Christian Fässler, Jonas Furrer

# Änderungsgeschichte

| Datum | Version | Änderung | Autor |
|---|---|---|---|
| **28.02.2012** | 0.1 | Erstellung des Dokumentes | Bargen, Fässler, Furrer |
| **01.03.2012** | 0.2 | Überarbeitung für erste Abgabe | Bargen, Fässler, Furrer |
| **05.03.2012** | 0.3 | Ergänzung der fehlenden Informationen | Bargen, Fässler, Furrer |
| **06.03.2012** | 0.4 | Layouting, Ergänzung mit fehlenden Informationen | Bargen, Fässler, Furrer |
| **31.03.2012** | 0.5 | MS Termine nachgetragen | Fässler |

# Inhalt

# 1. Einführung

## 1.1 Zweck

Dies Dokument beinhaltet die Projektplanung für das Projekt BierIdee, welches im Rahmen des Modules SE2 durchgeführt wird.

## 1.2 Gültigkeitsbereich

Die Gültigkeit des Dokumentes beschränkt sich auf die Dauer des SE2 Modules FS2012.

# 2. Projekt Übersicht

Unser Ziel ist eine mobile App für Android zu entwickeln, die sich mit Bier und Biersorten beschäftigt. Man soll Biersorten bewerten, beschreiben und taggen können, weiterhin soll man Informationen zu den Sorten erhalten und auch der soziale Faktor soll zentral vertreten sein. Zudem sollen auch automatische Empfehlungen ähnlich wie bei last.fm oder Amazon möglich sein: Vorlieben der Person A sind potentielle Vorlieben der Person B, wenn sich ihr Biergeschmack stark ähnelt. Auch Location - Based Features können eingebaut werden, wie z.B. eine Karte mit Bars oder anderen Lokalen in der Umgebung (möglicherweise mit Daten aus OpenStreetMap) oder Preise in der Umgebung. Die Aktivitäten der Nutzer sollen ähnlich wie auf Twitter in einem Stream dargestellt werden.

## 2.1 Zweck und Ziel

Eines der Hauptziele dieses Projektes liegt darin, die in den Fächern SE1 und SE2 erworbenen Kenntnisse in Software-Design und Projektmanagement an einem Projekt anzuwenden. Weiterhin wollen wir den Fokus auf Teamarbeit und Dokumentation legen.
Des Weiteren werden wir uns in die Android-App Entwicklung einarbeiten und Kenntnisse in den entsprechenden Werkzeugen aneignen.

## 2.2 Primäre Features

- Verzeichnis mit Biersorten und Brauereien
- Benutzerkonten/Profile
- Erfassen/Bearbeiten von Bieren durch Benutzer
- Taggen von Bieren mit Attributen wie bspw. Sorte
- Bewerten von Bieren durch Benutzer
- Individuelle Bier-Empfehlungen
- Erfassung von Benutzer-Aktivitäten (z. B Bierkonsum)
- Timeline mit Aktivitäten
- Internationalisierung

## 2.3 Erweiterungsmöglichkeiten

- Location-Based Services
- Achievements/Badges für bestimmte Benutzeraktivitäten
- Biere mit Barcode erfassen
- Fotos von Bieren hochladen
- Offizielle Brauereiprofile
- Barprofile
- Checkin in Bars
- Bierquiz
- Statistiken (z. B. Top rated, most consumed etc.)
- Integration in bestehende Social-Networks

## 2.4 Lieferumfang

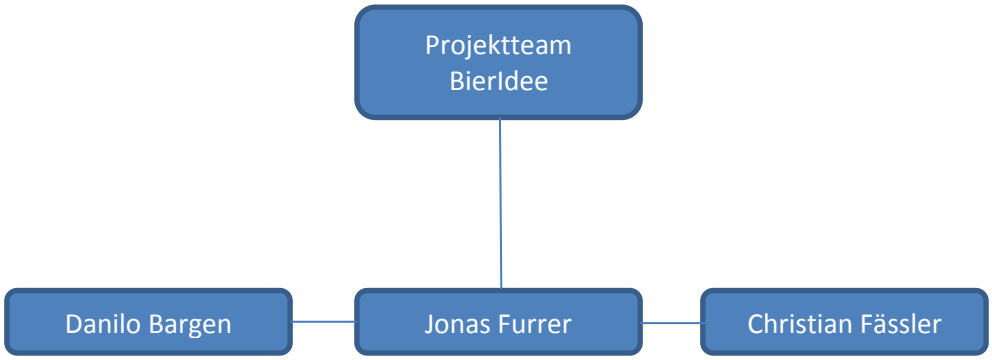
- Serverkomponente (DB / API)
- Android Client
- Projektdokumentation

## 2.5 Annahmen und Einschränkungen

In Rahmen dieses Projektes wird die App nicht offiziell Released (d.h nicht im Android-Market veröffentlicht).
Die Ressourcen der Infrastruktur lassen keinen Betrieb mit einer realistischen Anzahl Benutzer zu, ebenso können keine Tests mit entsprechend grosser Anzahl Clients durchgeführt werden.

# 3. Projektorganisation

## 3.1 Organisationsstruktur




## 3.2 Team

### 3.2.1 Danilo Bargen

| | |
|---|---|
| **Kenntnisse** | Java, HTML/CSS, REST |
| **Skype** | danilobargen |
| **Telefonnummer** | 079 728 93 96 |
| **Email** | dbargen@hsr.ch |

### 3.2.2 Jonas Furrer

| | |
|---|---|
| **Kenntnisse** | Java, HTML/CSS, Java Web |
| **Skype** | jonas-fu |
| **Telefonnummer** | 079 598 28 90 |
| **Email** | jfurrer@hsr.ch |

### 3.2.3 Christian Fässler

| | |
|---|---|
| **Kenntnisse** | JAVA, HTML/CSS, Apache |
| **Skype** | chrigi.faessler |
| **Telefonnummer** | 076 524 73 83 |
| **Email** | cfaessle@hsr.ch |

## 3.3 Verantwortlichkeiten

Das Projektteam besteht aus drei gleichgestellten Entwicklern. Unser Ziel ist es, dass jeder Entwickler in jedem Teilbereich des Projektes involviert ist, darum werden Zuständigkeiten nicht explizit zugeteilt. Dieses Vorgehen erlaubt es, Know-How optimal zu teilen und ist bei der gegebenen Projektgrösse gut realisierbar. Verantwortlichkeiten werden pro Task in der Iterationsplanung zugeteilt. Im Verlaufe des Projektes werden sich Vorlieben und Spezialkenntnisse herauskristallisieren.
Die einzige fixierte Verantwortlichkeit ist die Rolle der Ansprechperson.

| Wer | Verantwortlichkeiten |
|---|---|
| Jonas Furrer | Entwicklung, Infrastruktur, Planung |
| Danilo Bargen | Entwicklung, Infrastruktur, Planung |
| Christian Fässler | Entwicklung, Infrastruktur, Planung<br>Ansprechperson |

## 3.4 Externe Schnittstellen

**Betreuer:** Hans Rudin
**Ansprechperson / Projektleiter:** Christian Fässler

Es werden regelmässig Reviews der Work Products und des Projektstatus mit Herrn Rudin durchgeführt. Diese Reviews werden protokolliert und an alle Teilnehmenden gesendet. Die Protokolle werden dann beim nächsten Treffen bestätigt oder abgelehnt (nur durch Projektmitglieder, Hr Rudin berücksichtigt diese nur bei der Schlussbewertung).

# 4. Management Abläufe

## 4.1 Kostenvoranschlag

Voraussichtlich lassen sich die Zeitkosten des Projektes wie folgt planen:

- Voraussetzung: drei Personen in 14 Wochen.
- 8h pro Person pro Woche, ergibt insgesamt **336h**.
- Projektstart ist Montag 27. Februar 2012
- Abgabetermin ist spätestens der 1. Juni 2012 17:00

## 4.2 Zeitliche Planung

### 4.2.1 Phasen / Iterationen

Während dem gesamten Projekt wird parallel an Front- und Backend gearbeitet. Die App evolviert dadurch laufend. Die Aufgaben werden zu Beginn jeder Iteration priorisiert und entsprechend implementiert. Begonnen wird mit den Core-Features (Kommunikation Server-Client, Benutzersystem, Bierprofile) und erst dann werden Features mit tieferer Priorität entwickelt.

| Iteration | Beschreibung | Ende | Dauer in Wochen |
|---|---|---|---|
| Inception 1 | Projektantrag einreichen | SW01 | 1 |
| Elaboration 1 | Projektplan, Domain Model, Requirements (Brief Use Cases), Configuration Management (Redmine, Jenkins, Git, lokale Entwicklungsumgebung) | SW03 | 2 |
| Elaboration 2 | Externes Design und Mockups, Datenbankdesign, Fully dressed Use Cases, Evaluation Software | SW05 | 2 |
| Elaboration 3 | Architektur Prototyp | SW06 | 1 |
| Construction 1 | Backend Benutzersystem, erste sehr einfache Android-Version, Implementierung gemäss Priorisierung und Zeitschätzung in Redmine. Erste Testversion für Alphatester. | SW08 | 2 |
| Construction 2 | Hauptfeatures (zB Bierprofile, Aktivitäten erfassen) implementieren gemäss Priorisierung und Zeitschätzung in Redmine. Zweite Testversion für Alphatester. Bugs aus Test-Feedback fixen. | SW10 | 2 |
| Construction 3 | Hauptfeatures fertigstellen, erweiterte Features gemäss Priorisierung und Zeitschätzung in Redmine. Dritte und vierte Testversion für Alphatester. Bugs aus Test-Feedback fixen. | SW12 | 2 |
| Transition | Vorbereitung zur Schlusspräsentation und Abgabe. | SW14 | 2 |

### 4.2.2 Meilensteine

#### 4.2.2.1 MS1 Projektantrag einreichen

| | |
|---|---|
| **Termin** | Ende Inception 24.2.2012 |
| **Beschreibung** | Einreichung des Projektantrages zur Genehmigung. |
| **Work Products** | Projektantrag |

#### 4.2.2.2 MS2 Projektplan

| | |
|---|---|
| **Termin** | 8.3.2012 |
| **Beschreibung** | Review Projektplan mit Zeitplan und aktuellen Iterationsplänen |
| **Work Products** | Projektplan<br>Tasks in Redmine |

#### 4.2.2.3 MS3 Anforderungen und Analyse

| | |
|---|---|
| **Termin** | 16.3.2012 (SW04 Nach Ende Elaboration 1) |
| **Beschreibung** | Review der Anforderungsspezifikation und der Domainanalyse |
| **Work Products** | Domain Model<br>Tasks in Redmine<br>Requirements Specification (nicht funktional)<br>Use Cases im Brief-Format |

#### 4.2.2.4 MS4 Ende Elaboration / Architekturprototyp

| | |
|---|---|
| **Termin** | 3.4.2012 |
| **Beschreibung** | Zwischenpräsentation mit Demo eines Architekturprototypen, Review |
| **Work Products** | Main Use Cases im Fully-Dressed Format<br>Evaluationsdokumente<br>Architektur Prototyp |

#### 4.2.2.5 MS5 Architektur/Design

| | |
|---|---|
| **Termin** | 8.5.2012 |
| **Beschreibung** | Review von Architektur/Design und Architekturdokumentation |
| **Work Products** | Beta Release für grössere Tests mit Usern |

#### 4.2.2.6 MS6 Ende Construction

| | |
|---|---|
| **Termin** | 18.5.2012 |
| **Beschreibung** | Implementierung fertig |
| **Work Products** | Release Candidate 1 |

#### 4.2.2.7 MS7 Schlusspräsentation und Abgabe

| | |
|---|---|
| **Termin** | 29.5.2012 |
| **Beschreibung** | Präsentation und Demo der Software |
| **Work Products** | Produkt (Package)<br>Projekt-Dokumentation |

## 4.3 Besprechungen und Iterationsplanung

Eine Besprechung ist jeweils an Montagnachmittagen um 13:00 geplant. Zu diesem Zeitpunkt sind alle Projektmitarbeitenden an der HSR anwesend. Ziel ist es, gegenseitig über den aktuellen Stand seiner Tasks zu informieren und allfällige Probleme kurz im Plenum anzusprechen und entsprechend der agilen Entwicklung darauf zu reagieren. Mögliche Reaktionen sind Neupriorisierung von Tickets und Korrekturen von Aufwandschätzungen und gegenseitiges Review bez. Unterstützung. Fragen die sich durch Code Reviews ergeben werden gemeinsam beantwortet – jeder weiss so über jeden Teilbereich und den Stand des Projektes Bescheid.
Am jeweils ersten Montag einer Iteration findet zudem die ausführliche Iterationsplanung statt.
In dieser Planungssitzung wird folgendes jeweils in einem Protokoll festgehalten:

- Aufnahme IST Zustand der letzten Iteration. Was wurde Erreicht. (Screenshot aus Redmine.)
- Aufnahme aufgetretene Probleme Abweichungen in der abgeschlossenen Iteration.
- Planung SOLL für die nächste Iteration. Priorisierung Tickets. Zuteilung der Tickets. (Screenshot aus Redmine.)

# 5. Risikomanagement

## 5.1 Risiken

Die technischen Risiken werden im Dokument *TechnischeRisiken.xlsx* beschrieben und gewichtet.

## 5.2 Umgang mit Risiken

Die Risiken werden während der Elaboration Phase laufend neu bewertet. In der Construction Phase werden die Risiken jeweils in die Iterationsbesprechungen einfliessen und die Massnahmen werden neu bewertetet.

# 6. Arbeitspakete

Die Arbeitspakete werden in Redmine angelegt und gepflegt.
Die Redmine Instanz für das Projekt findet sich unter http://redmine.nusszipfel.com. Der Zugriff ist anonym (nur lesend) und authentifiziert (schreibend und lesend) möglich.
Wir nutzen die gegebenen Werkzeuge von Redmine wie Duplikate, Abhängigkeiten, Ticket-Blockaden etc.
Die Arbeitspakete werden zudem einer Iteration zugewiesen.
Die Eigenschaften der Tickets können laufend angepasst werden. Der Stand wird am Ende der Iterationen jeweils festgehalten (Screenshot).
Arbeitspakete sollten nicht zu gross sein, damit man den Aufwand seriös schätzen kann.

# 7. Infrastruktur

## 7.1 Hardware

- Private Notebooks aller Projektmitarbeiter als Entwicklungsgeräte.
- Private Android Geräte aller Projektmitarbeiter für das Testing der App.
- Private Linux VM auf Hosted-Rootserver als Build-, Tracking-, und Testing-Server.
- Private Linux VM auf Hosted-Rootserver als Live-Server.
- Netzwerk und Infrastruktur der HSR.

## 7.2 Software/Tools

- IDE: Eclipse mit Adroid SDK, diverse Plugins zur Sicherstellung der Code-Qualität
- Sourcecode Verwaltung: GIT (Github)
- Build-Server: Jenkins
- Tracking: Redmine
- RDBMS: PostgreSQL

## 7.3 Räumlichkeiten

Die Projektentwicklung, die Projektplanung sowie die Sitzungen werden grundsätzlich in den Räumlichkeiten der HSR durchgeführt. Voraussichtlich wird ein Teil der Produktentwicklung in privaten Räumlichkeiten durchgeführt werden.

## 7.4 Kommunikationsmittel

Bevorzugt wird die direkte Kommunikation. Weitere verwendete Kommunikationsmittel sind:

- Email
- Skype
- Redmine (Wiki und Kommentare)
- http://minutes.io (Protokolle)

# 8. Qualitätsmassnahmen

## 8.1 Dokumentation

Die Dokumentation wird laufend aktualisiert und bei jeder Iterationsbesprechung auf Vollständigkeit geprüft. Die Dokumentation wird im Git-Repository abgelegt.
Sie wird wie der Rest des Projektes mit Hilfe von Redmine Tickets und Aktivitäten verwaltet und tracked.
Die Dokumentation wird mit Latex erstellt und durch den Buildservers regelmässig kompiliert, um immer eine aktuelle Version verfügbar zu haben.

Für die Sourcecode-Dokumentation wird JavaDoc verwendet. Das Ergebnis wird schlussendlich als Teil der API-Dokumentation ausgeliefert.

## 8.2 Projektmanagement

Als Projektmanagement  Tool kommt Redmine zum Einsatz. Sämtliche Ressourcen werden damit tracked und gemanaged. Die Arbeitspakete werden in Redmine erstellt, priorisiert, den Teammitgliedern zugeteilt und dann implementiert.
Die verwendete Redmine-Instanz ist öffentlich unter http://redmine.nusszipfel.com erreichbar.

## 8.3 Entwicklung

Der Sourcecode wird mit Hilfe von Git verwaltet und versioniert. Als Hoster für das zentrale Repository wird GitHub verwendet. Die Codequalität wird mit Hilfe von Code-Style-Guidelines und entsprechenden Plugins in der IDE sowie Reviews (siehe Punkt Review) und automatisierten Tests (siehe Punkt Unit Testing) sichergestellt.

### 8.3.1 Vorgehen

Jedes Teammitglied hat ein lokales Repository, welches mittels eines zentralen Servers (Github) periodisch mit den anderen lokalen Repositories zusammengeführt wird. Commits sollten sich grundsätzlich auf Redmine-Tasks beziehen. Die Commit-Messages werden sorgfältig gewählt um möglichst aussagekräftig zu sein.
Die Feature-Entwicklung wird hauptsächlich in Feature Branches erledigt. Git ist bekannt für *cheap branching*, daher ist das im Gegensatz zu beispielsweise SVN kein Problem. Die Feature Branches werden dann auf Github gepushed. Wenn ein Feature fertig entwickelt ist, wird ein GIthub Pull Request vom Feature Branch auf den Master Branch erstellt. Dieser Pull Request ist eigentlich nichts anderes als ein Ticket, an welches Code angefügt ist. Der Code kann dann von anderen reviewed und kommentiert werden. Wenn man sich einig ist, dass der Commit gut ist, kann er in den *Master* Branch gemerged werden. Dieser wird dann auch von Jenkins gebuildet. Das erfolgreiche Bestehen von Tests wird vorher mit der lokalen Entwicklungsumgebung sichergestellt.
Ein gutes Beispiel dieses Workflows findet sich unter https://github.com/django/django/pull/58.

### 8.3.2 Code Reviews

Code Reviews werden regelmässig und in kurzen Abständen durchgeführt. Grundsätzlich soll jedes Teammitglied Reviews durchführen, die Reviews werden gezielt zwischen den Teammitgliedern abgewechselt. Kein genügend komplexer Commit wird ohne Review in den Master-Branch gemerged. Die Reviews werden mittels Pull Requests durchgeführt (siehe auch Punkt 8.3.1 ).

### 8.3.3 Code Style Guidelines

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).

## 8.4 Build Prozess

Der Master-Branch des Repositorys wird bei jedem Commit gebuildet. Somit stellen wir die Continuous Integration sicher und sehen jederzeit das Resultat der Tests sowie die Testabdeckung. Für einen fehlerhaften Commit definieren wir entsprechende (milde) Strafen, zB Gipfeli kaufen.

## 8.5 Testen

### 8.5.1 Unit Testing

Unit-Tests werden laufend während der Entwicklung geschrieben. Das Bereitstellen eines Unit-Tests ist integraler Bestandteil eines Arbeitspaketes. Die Tests werden, wie auch der Produkt-Sourcecode, mit Hilfe von Git verwaltet und befinden sich im gleichen Repository. Die Test-Abdeckung werden wir einerseits in der IDE mit Hilfe von *EclEmma* und auf dem Build-Server mit Hilfe eines geeigneten Plugins (wird noch evaluiert) sichergestellt. Die Tests werden bei jedem Commit in den Master Branch ausgeführt.

### 8.5.2 Integration Tests

Auf dem Build-Server werden mithilfe der Android Testing Tools regelmässig Integrationstests durchgeführt. Die Serverkomponente wird mittels JUNIT getestet.

### 8.5.3 System- und Usabilitytests

Die Applikation wird regelmässig durch freiwillige Tester getestet. Diese Tests beinhalten einerseits das Bewerten der Benutzbarkeit und Andererseits das Erfüllen der Anforderungen im Sinne von Systemtests. Das Feedback erhalten wird direkt mittels Formular oder Tickets zum Redmine Ticketsystem. Das genaue Vorgehen wird noch bestimmt.
Solche Tests werden voraussichtlich nach jeder Iteration der Construction Phase durchgeführt. Möglicherweise treffen wir uns dazu mit den Testern in gemütlicher Atmosphäre, um ein „APPero“ durchzuführen. Dies fördert die Motivation und bringt uns direktes Feedback.