HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Bachelorarbeit

Architekturkonzepte moderner Web-Applikationen

Hochschule für Technik Rapperswil
Frühjahressemester 2013

Erstellt: 13. März 2013, 09:12

Autoren Manuel Alabor
Alexandre Joly
Michael Weibel

Betreuer Prof. Hans Rudin
Experte Daniel Hiltebrand
Gegenleser tbd.

Our fancy abstract.. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Erklärung der Eigenständigkeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Manuel Alabor *Alexandre Joly* *Michael Weibel*

Danksagungen

Inhaltsverzeichnis

Kapitel 1 **Management Summary**

1.1. Ausgangslage

1.2. Vorgehen

1.3. Ergebnisse

1.4. Ausblick

Kapitel 2 Einleitung

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2.1. Involvierte Personen

2.1.1. Team

Manuel Alabor

Lorem Ipsum

Alexandre Joly

Lorem Ipsum

Michael Weibel

Lorem Ipsum

2.1.2. Betreuung & Bewertung

Prof. Hans Rudin

Lorem Ipsum

Kevin Gaunt

Lorem Ipsum

Daniel Hiltebrand

Lorem Ipsum

Kapitel 3 Analyse der Aufgabenstellung

Die Aufgabenstellung (siehe Anhang G) verzichtet bewusst auf funktionale Anforderungen an die zu erstellende Applikation. Weiter werden auch keine spezifischen Technologien zur Umsetzung vorgegeben.

Diese sehr offene Ausgangssituation wird lediglich durch die folgenden Ansprüche eingegrenzt:

1. Das Produkt soll unter Verwendung einer oder mehreren Internettechnologien konzipiert und umgesetzt werden.
2. Der zu erstellende Quellcode soll *State Of The Art* Architekturprinzipien ([ROC] & [Til]) exemplarisch darstellen und der interessierten Fachperson als auch Studenten der Vorlesung *Internettechnologien* als Anschauungsmaterial dienen können.

Von diesen zwei Leitsätzen ausgehend kann angenommen werden, dass die Demonstration von Architekturprinzipien klar im Vordergrund stehen soll. Da diese Prinzipien aber auch einem lernenden Publikum (Punkt 2) so interessant wie möglich präsentiert werden sollen, ist eine attraktive Verpackung ebenfalls nicht zu vernachlässigen.

Dieses Kapitel beantwortet im weiteren Verlauf dementsprechend folgende Fragen genauer:

1. Welche Schritte wurden im Bereich der Produktentwicklung durchlaufen? Auf welche Aspekte wurde besonders eingegangen?
2. Wie wurde die Technologie zur Umsetzung der generierten Produktidee ausgewählt und welche Kriterien waren dabei ausschlaggebend?
3. Wie werden die ROCA-Architekturprinzipien und Tilkovs “10 Empfehlungen” am optimalsten auf die Beispielapplikation abgebildet?

3.1. Produktentwicklung

Die Findung einer passenden Produktidee gestaltete sich unter den im vorherigen Abschnitt erwähnten Bedingungen nicht unbedingt als einfach:

Zwar soll ein Gros des Arbeitsaufwandes in das Entwickeln einer beispielhaften Architektur fliessen, diese soll aber in einem für Studierende möglichst attraktiven Gewand präsentiert werden.

3.1.1. Workshop

Um die zweitrangige Prozedur der Ideenfindung pragmatisch abhandeln zu können wurde ein Workshop mit Brainstorming und anschliessender Diskussionsrunde durchgeführt. Folgende Tabelle zeigt die Favoriten aus einem Pool generierter Ideen.

Die Spalte *Potential* bewertet jede Idee nach subjektiver Einschätzung des Projektteams unter Berücksichtigung folgender Faktoren:

- Funktionsumfang
- Konzeptionelle und technische Herausforderung
- Attraktivität (Für Projektteam)
- Attraktivität (Für Studierende des Moduls Internettechnologien)

Idee	***Pro***	***Contra***	***Potential***
WG-Aufgabenverwaltung	Viele Studierende identifizieren sich tendenziell damit, da sie selber in einer WG wohnen, Faktor *Gamification* sehr interessant		hoch
Instant Messenger	Attraktive Features wären realisierbar (Realtime, Websockets etc.)	Funktionelle Anforderungen könnten Rahmen sprengen	mittel
Aufgabenverwaltung	Evtl. gute Verwendung des Produkts	"*Gibt's wie Sand am Meer*", viele bestehende Beispielapplikationen [OS]	mittel
Chat		Bereits oft verwendet in bestehender Vorlesung, abgenutzte Thematik	tief
Forum		"*Gibt's wie Sand am Meer*"	tief

Tabelle 3.1.: Produktideenpool

Am verheissungsvollsten wurde die Idee des WG Aufgabenverwaltungstool eingeschätzt und gefiel dem gesamten Team von Beginn an ziemlich gut. Die Thematik Gamification in einem konkreten Produkt umsetzen zu können eliminierte schlussendlich die letzten Zweifel.

In einem nächsten Schritt wurde die rohe Produktidee mit einem Mindmap (siehe Anhang D) weiter ausgebaut und die ersten funktionalen Anforderungen wurden entwickelt. Daneben konnte eine konkrete Kurzbeschreibung sowie das erste Branding für das geplante Produkt formuliert resp. entworfen werden.

3.1.2. Die finale Produktidee: Roomies

Roomies soll einer WG ermöglichen, anfallende Aufgaben leicht unter den verschiedenen Bewohnern zu organisieren. Damit auch langweilige Ämtchen endlich erledigt werden, schafft Roomies durch ein Ranglisten- und Badgesystem (Gamification) einen Anreiz, um seine Mitbewohner übertrumpfen zu wollen.

Durch das Aufgreifen einer Thematik aus dem Studentenalltag soll Roomies für lernende aus dem Modul *Internettechnologien* einen leichten Einstieg in die tendenziell trockene Materie der Softwarearchitektur bieten.

3.1.3. Branding

Der namensgebende Ausdruck *Roomie* stammt aus dem US-amerikanischen und bedeutet soviel wie *Mitbewohner* oder *Zimmernachbar* [Dic]. Passend dazu soll neben dem Namen auch das restliche Produktbranding an die US-amerikanische College-Welt angelehnt werden.

Vom Logo über die Farbwahl bis zum späteren User Interface Design sollen folgende Stilelemente als roter Faden verwendet werden:

1. *Gedimmte* Farben, keine grellen Akzente
2. Simple, aber eingängige und klar definierte Formensprache
3. Serifen-betonte Schriftart als Stilmittel

Abbildung 3.1.: Roomies Logo im College Stil

Abbildung 3.2.: Roomies Logo in verschiedenen Grössen & Varianten

3.2. Technologieevaluation

Das Thema "Architekturkonzepte moderner Web-Applikationen" legt den Schluss nahe, nicht nur aktuellste Architekturprinzipien bei der Umsetzung der Beispielapplikation zu verwenden, sondern auch im Bereich der Technologiewahl auf etablierte Platzhirsche wie Java oder C# (in Verbindung mit deren Web-Frameworks) zu verzichten.

Unter Berücksichtigung der persönlichen Erfahrungen und Einschätzungen aller Projektteilnehmer wurde in Vereinbarung mit dem Betreuer im Zuge einer Evaluation eine Shortlist mit folgenden Technologiekandidaten zusammengestellt:

Kandidat	***Warum?***	***Scope***
Java	Trotz des einführenden Statements, Platzhirsche von einer engeren Auswahl auszuschliessen, war das Projektteam aufgrund der vorangegangenen Studienarbeit davon überzeugt, dass Java, insbesondere als Backendtechnologie, mit den "jungen Wilden" problemlos mithalten kann.	Web-Frameworks
JavaScript	Während den letzten zwei Jahren erlebte JavaScript eine Renaissance: Mit node.js schaffte es den Sprung vom Frontend-Layer ins Backend und erfreut sich in der OpenSource als auch der Industrie-Community grösster Beliebtheit.	node.js
Ruby	Ruby hat sich in der näheren Vergangenheit zusammen mit Ruby On Rails im Markt etablieren können. Als relativ junge Technologie durfte es aus diesem Grund bei einer Evaluation nicht ignoriert werden.	Ruby On Rails

Tabelle 3.2.: Shortlist Technologieevaluation

Die drei Kandidaten wurden innerhalb des Teams aufgeteilt und anschliessend einer tieferen Analyse unterzogen.

Folgende Abschnitte enthalten die Ergebnisse. Im Anhang E sind zusätzlich während der Analyse entstandene Materialien zu den jeweiligen Technologien verfügbar.

3.2.1. Java

3.2.2. JavaScript

3.2.3. Ruby

3.3. Architekturprinzipien

Kapitel 4 **Anforderungsanalyse**

4.1. Funktionale Anforderungen

ID	*Name*	*Beschreibung*	*Priorität*
F1	WG erstellen	Die Applikation erlaubt es eine WG zu erstellen.	hoch
F2	Einladung	Die Applikation erlaubt es, einen Benutzer in eine WG einzuladen.	hoch
F3	Aufgabe erstellen	Die Applikation erlaubt es, eine Aufgabe zu erstellen.	hoch
F4	Aufgabe erledigen	Die Applikation erlaubt es, eine Aufgabe zu erledigen.	hoch
F5	Verlassen	Die Applikation erlaubt es, eine WG zu verlassen.	mittel
F6	Aufgabe bearbeiten	Die Applikation erlaubt es, eine Aufgabe zu bearbeiten.	mittel
F7	Rangliste anzeigen	Die Applikation erlaubt es, eine Rangliste für die Bewohner einer WG anzuzeigen.	mittel
F8	Erfolge vergeben	Die Applikation erlaubt es, Erfolge aufgrund von Regeln zu vergeben.	mittel
F9	WG auflösen	Die Applikation erlaubt es, eine WG aufzulösen.	niedrig
F10	Bewohnerverwaltung	Die Applikation erlaubt es, die Bewohner einer WG zu verwalten.	niedrig
F11	Inhalte teilen	Die Applikation erlaubt es, Inhalte auf Social Media Kanälen zu teilen.	niedrig

Tabelle 4.1.: Funktionale Anforderungen

4.2. Nichtfunktionale Anforderungen

ID	*Name*	*Beschreibung*
NF1	Antwortzeit	Die Applikation antwortet bei normalen Anfragen innerhalb von 0.2s.
NF2	Desktop Browserkompatibilität	Die Applikation unterstützt Internet Explorer 8 und höher, Chrome 25 und höher, Firefox 19 und höher sowie Safari 6 und höher.
NF3	Mobile Browserkompatibilität	Die Applikation unterstützt Safari 6.0 und Android Browser 4.0.
NF4	Sicherheit	Die Applikation kontrolliert den Zugriff auf geschützte Ressourcen.
NF5	ROCA Prinzipien	Die Applikation entspricht den ROCA [ROC] Prinzipien.

Tabelle 4.2.: Nichtfunktionale Anforderungen

4.3. Use Cases

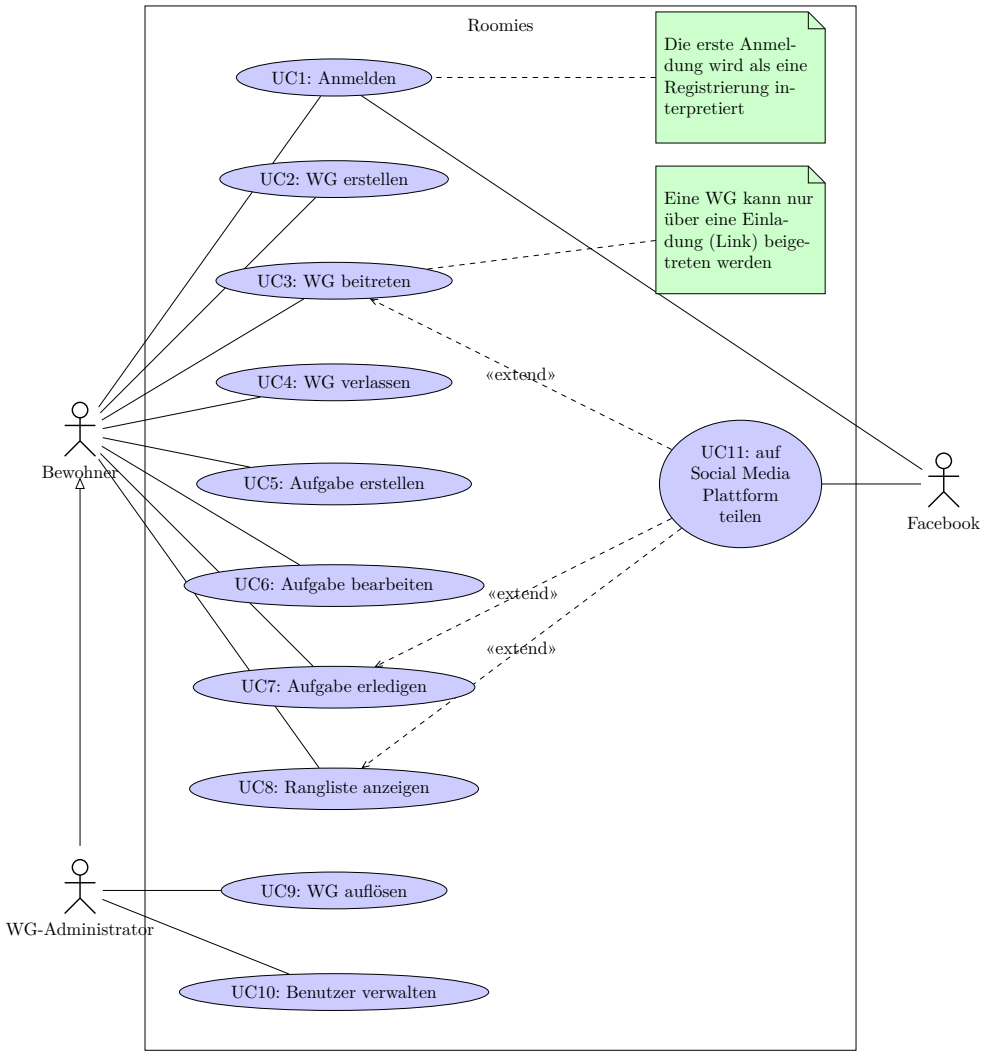


Abbildung 4.1.: Use Case Diagramm

4.3.1. Aktoren

Name	*Beschreibung*
Bewohner	Als Bewohner wird ein Anwender der Roomies Anwendung bezeichnet, der zu einer WG gehört. Dieser besitzt die Rechte Aufgaben seiner zugehörigen WG zu verwalten.
WG-Administrator	Der WG-Administrator ist eine Erweiterung des Aktors Bewohner. Der Ersteller einer WG wird automatisch als WG-Administrator ernannt. Diese Role kann an Bewohner weitergegeben werden.

Tabelle 4.3.: Aktoren

4.3.2. UC1: Anmelden

Use Case Name	UC1: Anmelden
Scope	Roomies
Primary Actor	Bewohner
Secondary Actor	Facebook
Story	Der Bewohner startet Roomies. Hier zeigt ihm das System das Anmeldeformular. Der Benutzer meldet sich mittels seines Facebook-Logins an. Das System überprüft die Daten. Sind sie gültig wird der Benutzer auf die Aufgabenseite weitergeleitet.

Tabelle 4.4.: UC1: Anmelden

4.3.3. UC2: WG erstellen

Use Case Name	UC2: WG erstellen
Scope	Roomies
Primary Actor	Bewohner
Story	Ein Bewohner hat die Möglichkeit eine WG zu erstellen, falls er noch zu Keiner angehört. Hierfür wählt der Bewohner die Option 'WG erstellen'. Das System leitet ihn auf das entsprechende Formular und fordert den Bewohner die WG-Daten einzugeben. Nachdem der Bewohner diese eingegeben hat, überprüft das System die Gültigkeit der Daten und leitet der Bewohner auf die Aufgabenseite der WG weiter. Ebenfalls die Rolle WG-Administrator werden dem Bewohner automatisch zugeteilt.

Tabelle 4.5.: UC2: WG erstellen

4.3.4. UC3: WG beitreten

Use Case Name	UC3: WG beitreten
Scope	Roomies
Primary Actor	Bewohner
Story	Um in einer WG beizutreten, muss der Link zur Einladung dem 'zukünftigen' Bewohner bekannt sein. Ein solcher Link wird vom System erzeugt. Die Verbreitung jedoch, ist völlig dem WG-Administrator überlassen und ist nicht Teil der Anwendung. Hat ein Bewohner den Link geöffnet, wird vom System eine Bestätigung gefordert. Bestätigt der Bewohner diese, wird er als Bewohner dessen WG 'registriert' und zur Aufgabenseite der WG weitergeleitet.

Tabelle 4.6.: UC3: WG beitreten

4.3.5. UC4: WG verlassen

Use Case Name	UC4: WG verlassen
Scope	Roomies
Primary Actor	Bewohner
Story	Wählt ein Bewohner die Option 'WG verlassen', wird er vom System gefordert dies zu Bestätigen. Nach der Bestätigung setzt das System der Bewohner als inaktiv und leitet der 'Ex'-Bewohner auf eine 'Aufwiedersehen-Seite' weiter. Hat der Bewohner, als einziger, die Rolle 'WG-Administrator', so muss er, vor dem inaktiv Setzen, seine Rolle einem anderen Bewohner übetragen. //TODO: neuer UC???

Tabelle 4.7.: UC4: WG verlassen

4.3.6. UC5: Aufgabe erstellen

Use Case Name	UC5: Aufgabe erstellen
Scope	Roomies
Primary Actor	Bewohner
Story	TODO

Tabelle 4.8.: UC5: Aufgabe erstellen

4.3.7. UC6: Aufgabe bearbeiten

Use Case Name	UC6: Aufgabe bearbeiten
Scope	Roomies
Primary Actor	Bewohner
Story	TODO

Tabelle 4.9.: UC6: Aufgabe bearbeiten

4.3.8. UC7: Aufgabe erledigen

Use Case Name	UC7: Aufgabe erledigen
Scope	Roomies
Primary Actor	Bewohner
Story	TODO

Tabelle 4.10.: UC7: Aufgabe erledigen

4.3.9. UC8: Rangliste anzeigen

Use Case Name	UC8: Rangliste anzeigen
Scope	Roomies
Primary Actor	Bewohner
Story	TODO

Tabelle 4.11.: UC8: Rangliste anzeigen

4.3.10. UC9: WG auflösen

Use Case Name	UC9: WG auflösen
Scope	Roomies
Primary Actor	WG-Administrator
Story	TODO

Tabelle 4.12.: UC9: WG auflösen

4.3.11. UC10: Benutzer verwalten

Use Case Name	UC10: Benutzer verwalten
Scope	Roomies
Primary Actor	WG-Administrator
Story	TODO

Tabelle 4.13.: UC10: Benutzer verwalten

4.3.12. UC11: auf Social Media Plattform teilen

//TODO: ist erweiteter UC

Kapitel 5 Technische Architektur

5.1. Einleitung

TODO: Einleitungstext

5.2. Domainmodel

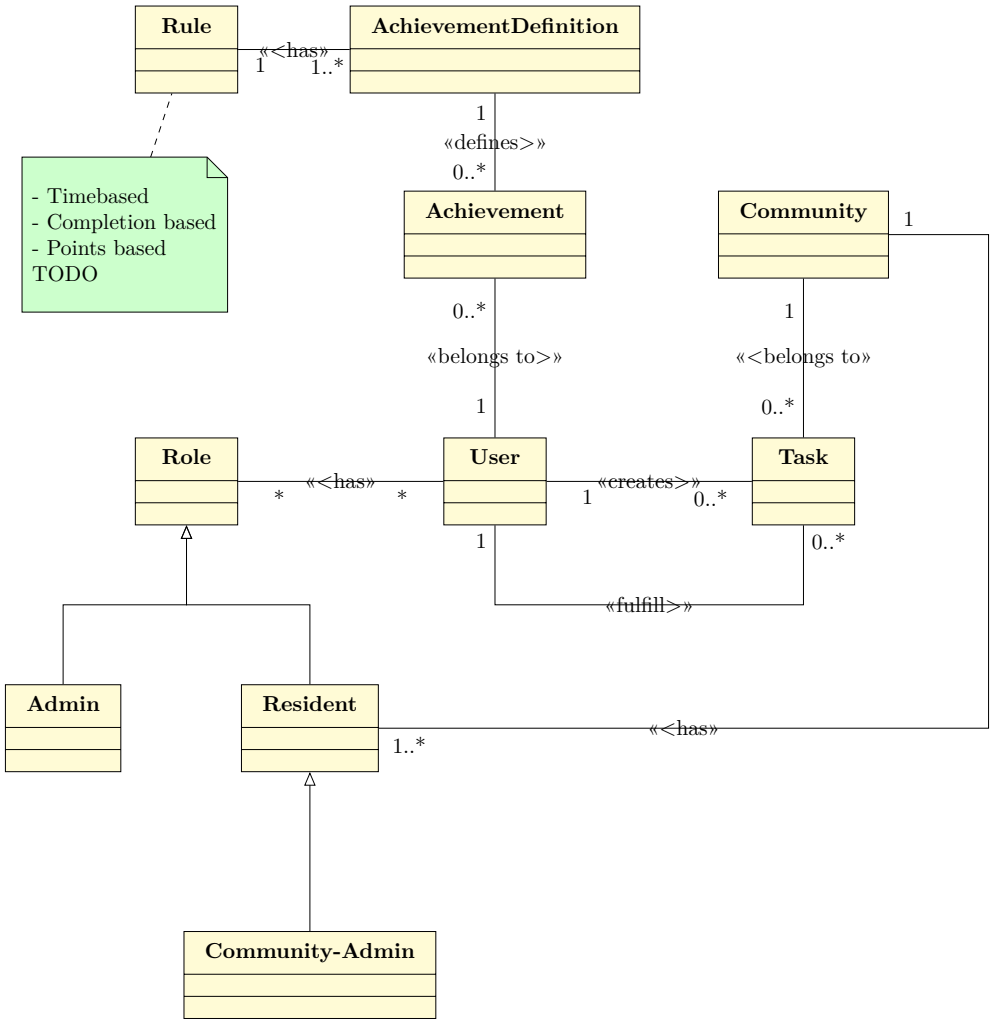


Abbildung 5.1.: Domainmodel

Achievements

Achievements (Erfolge) werden an User vergeben, wenn Sie bestimmte Regeln (Rules) erreicht haben. Regeln können Zeit-basiert (z.B. ist kürzlich einer WG beigetreten), Punkte-basiert (z.B. hat 10 Aufgaben erledigt) oder auch bei Komplettierung (z.B. hat sich registriert) vergeben werden. Jedes Achievement hat eine Definition, im Domainmodel durch die «AchievementDefinition» erwähnt, welche das Aussehen und die Regeln bestimmen.

Roles

Um die Rechte von Benutzern des Systems voneinander zu unterscheiden, wird Role Based Access Control [Sch+06, Kapitel 8.2] verwendet. Sobald ein Benutzer einer WG beitritt, bekommt er automatisch die Rolle «Resident» und wird dadurch zum Bewohner. Falls ein Benutzer eine neue WG erstellt, wird er automatisch zum «Community-Admin». Applikationsweite Administratoren haben die Rolle «Admin».

5.3. Entity-Relationship Diagramm

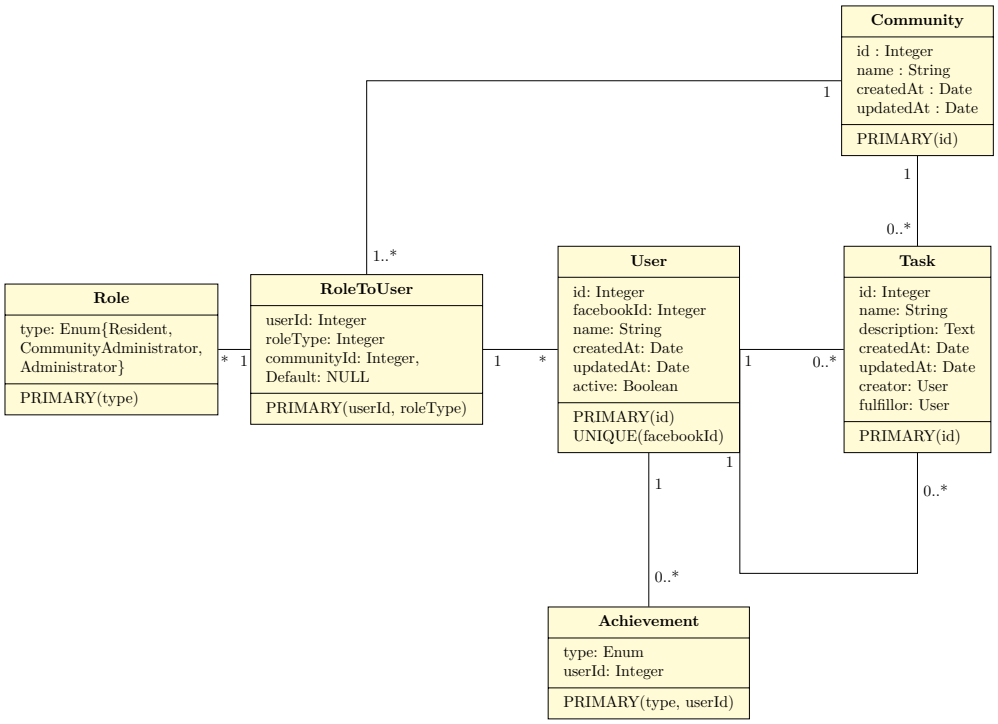


Abbildung 5.2.: Entity-Relationship Diagramm

5.4. Software Layers

Technologie-Unabhängig

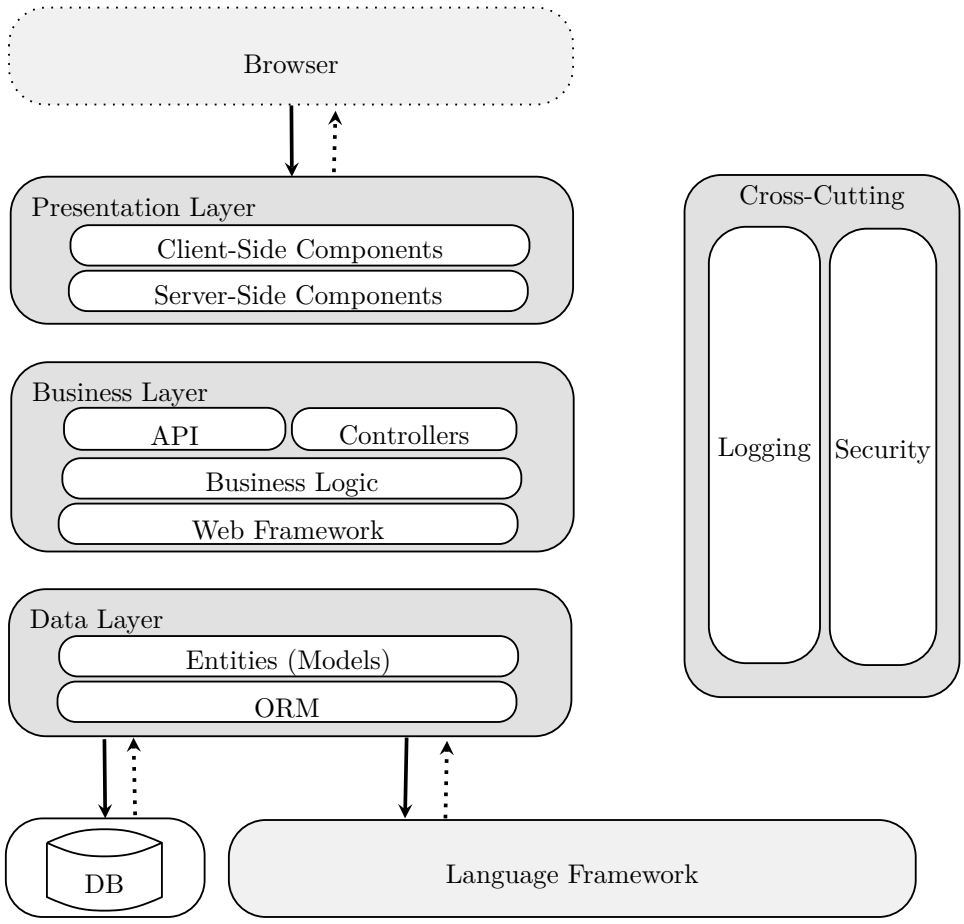


Abbildung 5.3.: Software Layers

Implementation

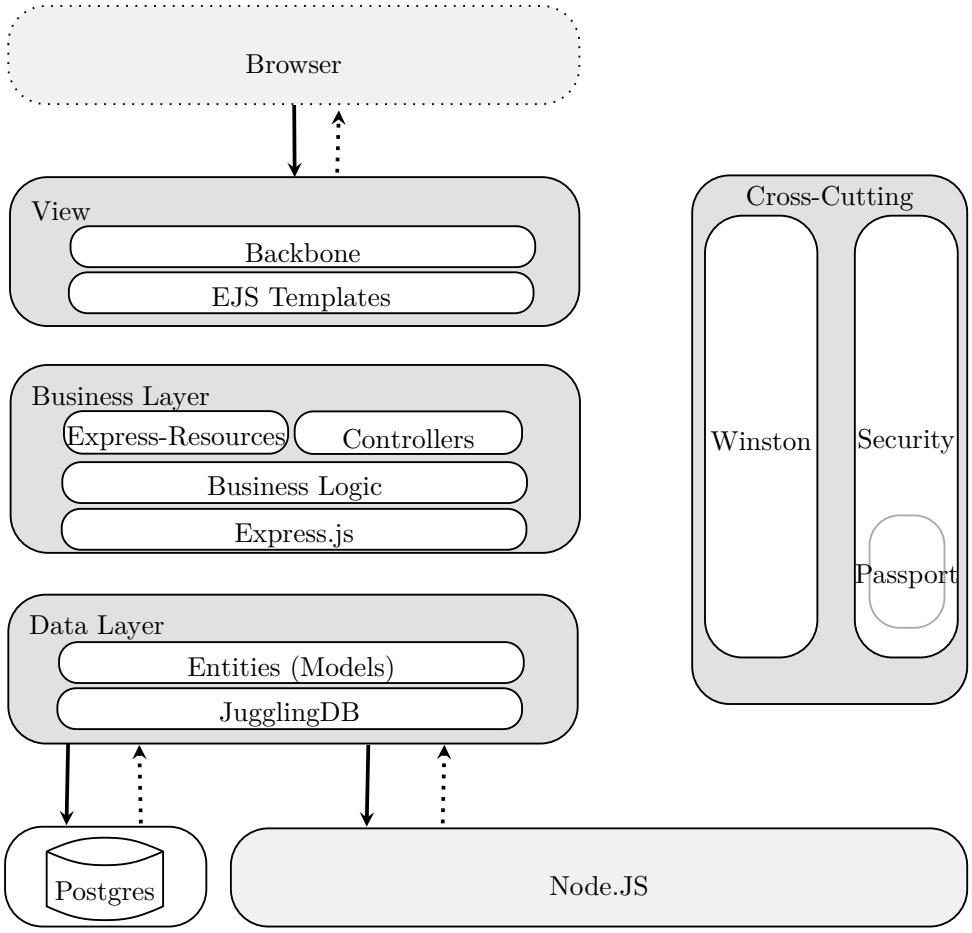


Abbildung 5.4.: Software Layers

5.5. Design View

5.6. Deployment View

TODO

Kapitel 6 **Projektplanung**

6.1. Infrastruktur

Ressource	*URL*
Projektverwaltung	`http://redmine.alabor.me/projects/ba2013`
Code: Git Repository	`https://github.com/mweibel/ba`
Code: CI	`https://travis-ci.org/mweibel/BA`
Thesis: Git Repository	`https://github.com/mweibel/BA-Dokumentation`
Thesis: PDF	`http://mweibel.github.com/BA-Dokumentation/thesis.pdf`
Thesis: CI	`https://travis-ci.org/mweibel/BA-Dokumentation`
Meeting Protokollierung	`https://github.com/mweibel/BA-Dokumentation/wiki/Meetings`

Tabelle 6.1.: Projektrelevante URL's

6.1.1. Projektverwaltung

Für die komplette Projektplanung, die Zeitrapportierung sowie das Issue-Management wird Redmine eingesetzt.

6.1.2. Entwicklungsumgebung

Zur Entwicklung von Quellcode-Artefakten steht eine mit Vagrant [Has] paketierte Virtual Machine bereit. Sie enthält alle notwendigen Abhängigkeiten und Einstellungen:

- node.js 0.10.0
- PostgreSQL 9.1
- Ruby 2.0.0 (installiert via rvm)
- ZSH (inkl. oh-my-zsh)

Das Code Repository enthält ein *Vagrantfile* welches durch den Befehl *vagrant up* in der Kommandozeile automatisch das Image der vorbereiteten VM lokal verfügbar macht und startet.

6.1.3. Git Repositories

Sowohl Quellcodeartefakte als auch die in LaTeX formulierte Thesis (dieses Dokument) wird in auf GitHub abgelegten Git Repositories versioniert bzw. zentral gespeichert.

6.1.4. Continuous Integration

Dieses Projekt verwendet Travis CI als Continuous Integration Lösung.

Beide Git Repositories (Code & Thesis) verfügen über einen Push-Hook welcher automatisch einen Build im CI-System auslöst.

6.2. Meetings

6.2.1. Regelmässiges Statusmeeting

Während der gesamten Projektdauer findet jeweils am Mittwoch um 10 Uhr ein wöchentliches Statusmeeting statt. Die Sitzung wird abwechslungsweise jeweils von einer Person aus dem Projektteam geführt sowie von einer anderen protokolliert.

Das Projektteam stellt die Agenda der aktuellen Sitzung bis spätestens am vorangehenden Dienstag Abend bereit.

6.3. Phasenplanung

Die Phasenplanung orientiert sich grob am RUP und ist unterteilt in eine *Inception-*, *Elaboration-*, fünf *Construction-* sowie jeweils eine *Transition-* und *Abschlussphase*.

Phase	***Dauer***	***Beschreibung***
Inception	3 Wochen	Projektsetup, genauere Definition der Aufgabe, Vorbereitungen & Planungen
Elaboration	3 Wochen	Anforderungsanalysen, Entwicklung eines Architekturprototypen und genauere technische Evaluationen. Guidelines für Quellcode und Testing werden erstellt.
Construction 1	2 Wochen	Umsetzung des Applikationsfundaments, UI Design, Umsetzung erster als *Hoch* priorisierter Use Cases. Qualitätssicherung in Form von Reviews & Unit Testing.
Construction 2	2 Wochen	Fertigstellung der restlichen als *Hoch* priorisierten Use Cases. Qualitätssicherung in Form von Reviews & Unit Testing.
Construction 3	2 Wochen	Umsetzung des Gamification-Teils der Applikation. Qualitätssicherung in Form von Reviews & Unit Testing.
Construction 4	2 Wochen	Implementation der restlichen als *Mittel* priorisierten Use Cases. Qualitätssicherung in Form von Reviews & Unit Testing.
Construction 5	2 Wochen	Umsetzung aller restlichen als *Tief* priorisierten Use Cases sowie erstes Bugfixing gem. geführter Issueliste. Qualitätssicherung in Form von Reviews & Unit Testing.
Transition	1 Wochen	Abschliessende Bugfixing-Arbeiten. Code-Freeze und Erstellung von Deployment-Pakete.
Abschluss	2 Wochen	Finalisierung der Dokumentation sowie Erstellung der HSR Artefakte *A100* sowie *A101*.

Tabelle 6.2.: Projektphasenbeschreibung

Jede einzelne Phase wird jeweils von einem Meilenstein abgeschlossen, was in folgendem Gantt-Diagramm ersichtlich ist.

Eine Ausnahme bildet der Meilenstein *M9: Abgabe HSR Artefakte*. Zu diesem Zeitpunkt müssen die entsprechenden Artefakte *A100* sowie *A101* im HSR abgabebereit sein.

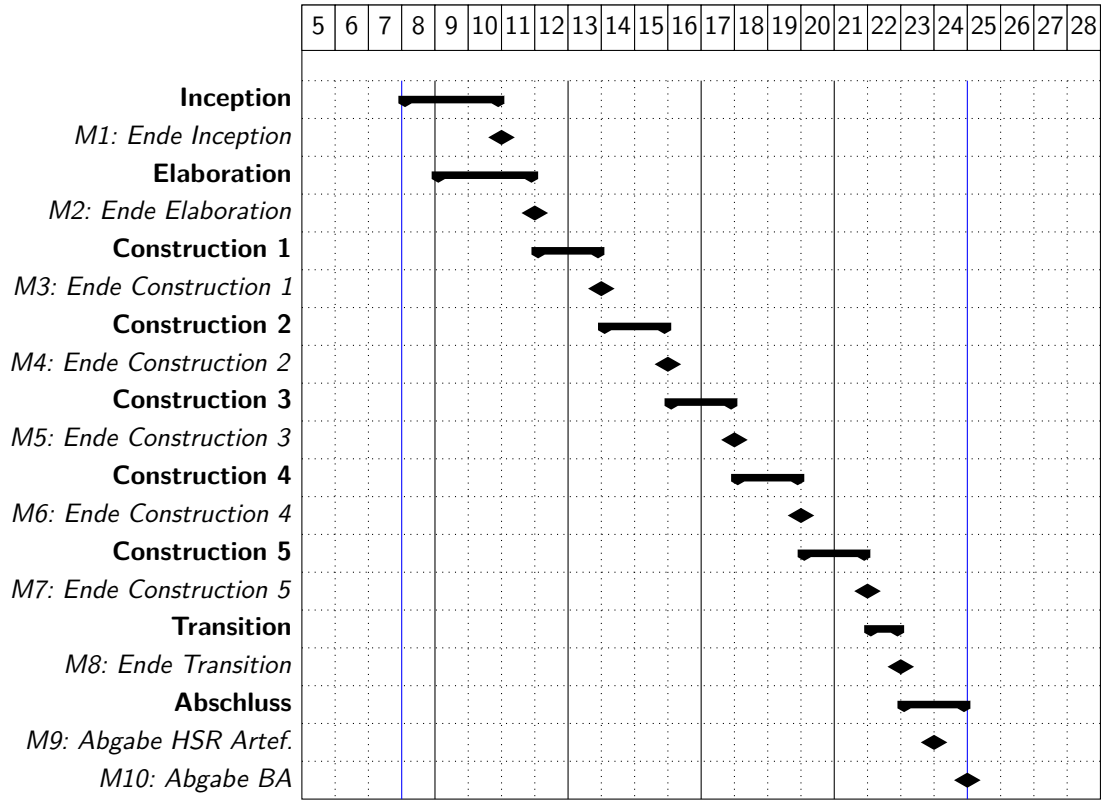


Abbildung 6.1.: Phasenübersicht mit Meilensteinen, Kalenderwochen Februar bis Juli 2013

6.4. Meilensteine

ID	*Meilenstein*	*Termin*	*Beschreibung*
M1	Ende Inception	10.03.2013	Die Aufgabenstellung wurde gem. Auftrag klar definiert und die Projektinfrastruktur ist aufgesetzt. Eine initiale Projektplanung besteht.
M2	Ende Elaboration	17.03.2013	Konkrete Technologie und Guidelines sind definiert. Anforderungsdokumente sind erstellt und abgenommen. Initiale SAD und Architekturprototyp bereit.
M3	Ende Construction 1	31.03.2013	Das Fundament der Applikation wurde implementiert. Weiter wurden die ersten Use Cases der Priorität *Hoch* umgesetzt.
M4	Ende Construction 2	14.04.2013	Alle Use Cases der Priorität *Hoch* sind umgesetzt.
M5	Ende Construction 3	28.04.2013	
M6	Ende Construction 4	12.05.2013	
M7	Ende Construction 5	26.05.2013	
M8	Ende Transition	02.06.2013	Deployment-Pakete und zugehörige Anleitungen sind bereit. Bugfixing abgeschlossen resp. ausstehende Bugs dokumentiert.
M9	Abgabe HSR Artefakte	07.06.2013	Das A0-Poster sowie die Kurzfassung der Bachelorarbeit sind dem Betreuer zugestellt.
M10	Abgabe Bachelorarbeit	14.06.2013	Alle abzugebenden Artefakte sind dem Betreuer zugestellt worden.

Tabelle 6.3.: Meilensteine

6.5. Artefakte

Dieser Abschnitt beschreibt alle Arbeitsprodukte (Artefakte), welche zwingend erstellt und abgegeben werden müssen.

Falls nicht anders vermerkt sind alle Artefakte Teil der Dokumentation.

ID	*Meilenstein*	*Artefakt*	*Beschreibung*
A20	M2	Projektplanung	Projektablauf, Infrastrukturbeschreibung & Phasenplanung
A21	M2	Analyse der Aufgabenstellung	Produktentwicklung, Technologieevaluation & Analyse Architekturprinzipien
A22	M2	Guidelines	Quellcode- und Testing-Guidelines
A23	M2	Anforderungsanalyse	Funktionale & nichtfunktionale Anforderungen, Use Cases
A24	M2	Domainmodel	Analyse der Problemdomäne
A25	M2	SAD	Beschreibung der angestrebten Architektur für die Beispielapplikation.
A26	M2	Architekturprototyp	Exemplarische Implementierung der angestrebten Technologie/Architektur *Typ: Quellcode/Applikation*
A80	M8	Quellcode Paket	Quellcode der Beispielapplikation zum eigenen, spezifischen Deployment. Bereits zur Weiterentwicklung. *Typ: Quellcode*
A81	M8	Vagrant Paket	VM-Image mit lauffähiger Version der Beispielapplikation. *Typ: Vagrant Image*
A82	M8	Heroku Paket	Beispielapplikation ist so vorbereitet, dass ein Deployment auf Heroku problemlos möglich ist. *Typ: Quellcode*
A83	M8	Installationsanleitung	Anleitung wie die verschiedenen Deployment-Pakete (Artefakte *A80-82*) eingesetzt/installiert werden können.
A100	M10	A0-Poster	Gem. HSR Vorgaben zu erstellendes Poster mit Übersucht zu dieser Bachelorarbeit.
A101	M10	Kurzfassung	Gem. HSR Vorgaben zu erstellende Kurzfassung dieser Bachelorarbeit.
A102	M10	Dokumentation	Alle bisherigen Dokumentationsartefakte zusammengefasst in einem Bericht. Wo nötig, sind entsprechende Kapitel dem Projektablauf entsprechend nachgeführt (bspw. *A25 SAD* etc.)

Tabelle 6.4.: Abzugebende Artefakte

Anhang A **Abbildungen, Tabellen & Quellcodes**

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang B Literatur

[Aira] Airbnb. *Airbnb JavaScript Style Guide*. URL: `https://github.com/airbnb/javascript` (besucht am 10. 03. 2013).

[Airb] Airbnb. *Airbnb JavaScript Style Guide (Updated)*. URL: `https://github.com/mekanics/javascript-style-guide` (besucht am 10. 03. 2013).

[Det+] Sebastian Deterding u. a. *Gamification: Toward a Definition*. URL: `http://hci.usask.ca/uploads/219-02-Deterding,-Khaled,-Nacke,-Dixon.pdf` (besucht am 11. 03. 2013).

[Dic] Urban Dictionary. *Urban Dictionary: roomie*. URL: `http://roomie.urbanup.com/1433981` (besucht am 11. 03. 2013).

[Has] HashiCorp. *Vagrant*. URL: `http://www.vagrantup.com/` (besucht am 06. 03. 2013).

[OS] Addy Osmani und Sindre Sorhus. *TodoMVC*. URL: `http://addyosmani.github.com/todomvc/` (besucht am 11. 03. 2013).

[ROC] ROCA. *Resource-oriented Client Architecture*. URL: `http://roca-style.org` (besucht am 06. 03. 2013).

[Sch+06] Markus Schumacher u. a. "Role Based Access Control". In: *Security Patterns - Integrating Security and Systems Engineering*. 1. Aufl. John Wiley & Sons, Ltd, 2006. Kap. 8.2, S. 249–252. ISBN: 978-0-470-85884-4.

[Til] Stefan Tilkov. *Building large web-based systems: 10 Recommendations*. URL: `http://www.innoq.com/blog/st/presentations/2013/2013-01-22-WebArchitectureRecommendations.pdf` (besucht am 12. 03. 2013).

Anhang C Glossar

Benutzer

Ein Benutzer .... TOOODOOO. 14

Bewohner

Ein Bewohner .... TOOODOOO. 14, 23

CI

Continuous Integration. 27

Gamification

Als Gamification oder Gamifizierung (seltener auch Spielifizierung) bezeichnet man die Anwendung spieltypischer Elemente und Prozesse in spielfremdem Kontext [Det+]. 11, 12, 29

Push-Hook

In *git* bezeichnet ein Push-Hook ein Script, welches nach jedem Commit ausgeführt wird.. 28

RUP

Rational Unified Process; Iteratives Projektvorgehen. 29

SAD

Software Architektur Dokument. 31, 32

VM

Virtual Machine. 32

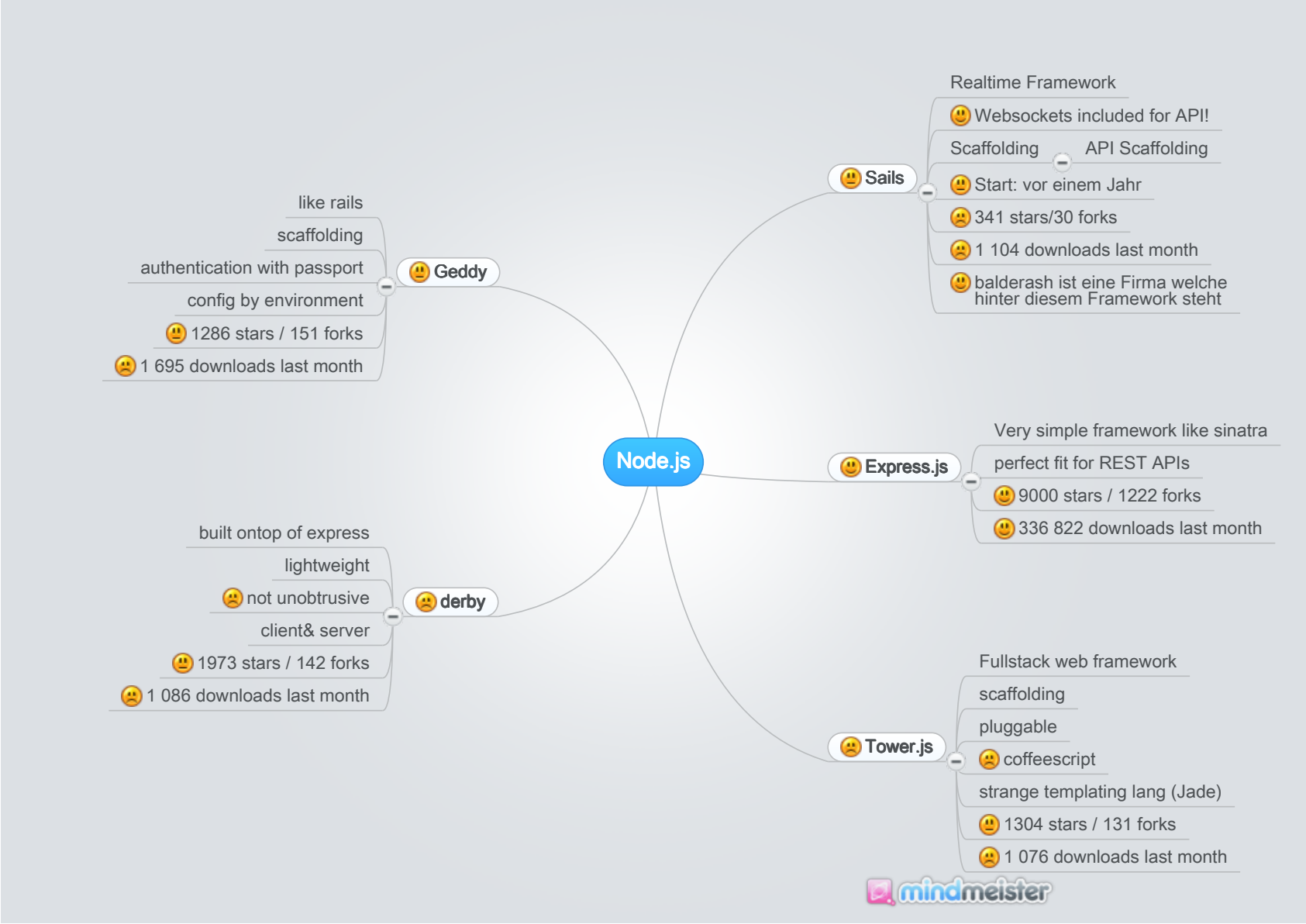
WG

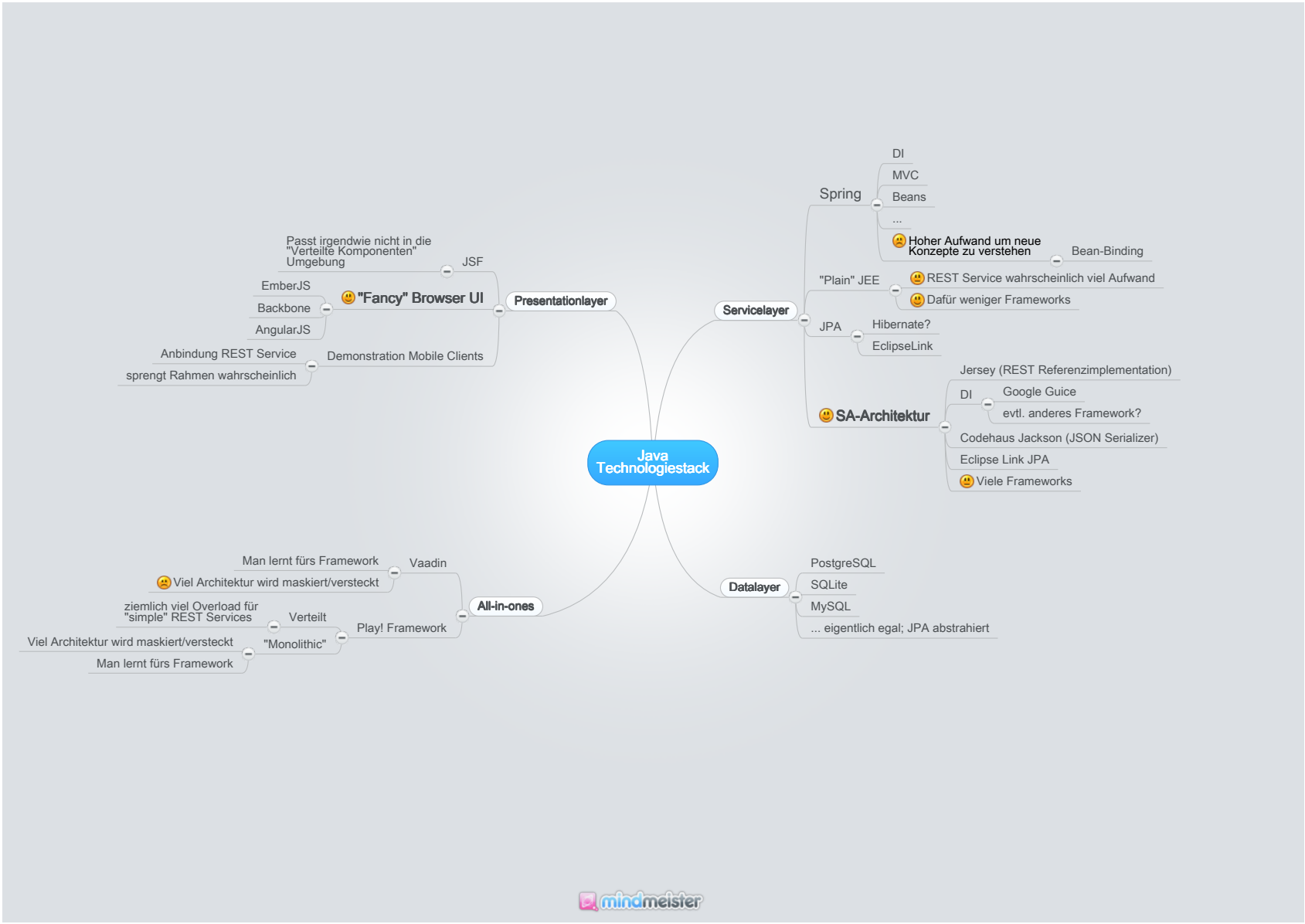
Eine WG .... TOOODOOO. 11, 12, 14, 23

Anhang D **Produktentwicklung**


Beispielapplikation "Roomies"
ROOMIES
Namensgebung
Mitbewohner
Roomie
Specials
Grundidee
Aufgabenverwaltung für WG's
Anreiz schaffen durch Gameification
Integration Social Media (Sharing etc.)
Hoffentlich eine motivierende Problem Domain
ROCA
Use Cases
Login
Facebook?
Twitter?
Registrieren
Facebook?
Twitter?
Invitationlink für bestehende WG
Aufgabe erstellen
Titel
Wiederkehrende Aufgaben (Wöchentlich etc.)
Beschreibung
Belohnung[1]
How many points does the finishing person get?
Erinnerung
E-Mail
Popup (falls Benutzer aktiv)
API
Private?[2]
Aufgaben Zuweiseung
Aufgabe zuweisen
Aufgabe reservieren[3]
Aufgabe beenden
Aufgabe bearbeiten
Aufgaben anzeigen
Eigene
Alle
Nicht reservierte/zugewiesene
WG erstellen
WG beitreten
Scoreboard anzeigen
Rangliste einer WG
Monatliche/wöchentliche Wettbewerbe?
Hall of Fame?
Diagramme zur Visualisierung?
Abwesenheitsfunktion[4]
Ferien
mindmeister

Anhang E **Technologieevaluation**


Node.js
Sails
Realtime Framework
Websockets included for API!
Scaffolding
API Scaffolding
Start: vor einem Jahr
341 stars/30 forks
1 104 downloads last month
balderash ist eine Firma welche hinter diesem Framework steht
Geddy
like rails
scaffolding
authentication with passport
config by environment
1286 stars / 151 forks
1 695 downloads last month
Express.js
Very simple framework like sinatra
perfect fit for REST APIs
9000 stars / 1222 forks
336 822 downloads last month
derby
built ontop of express
lightweight
not unobtrusive
client& server
1973 stars / 142 forks
1 086 downloads last month
Tower.js
Fullstack web framework
scaffolding
pluggable
coffeescript
strange templating lang (Jade)
1304 stars / 131 forks
1 076 downloads last month
mindmeister


Java Technologiestack
Presentationlayer
"Fancy" Browser UI
EmberJS
Backbone
AngularJS
JSF
Passt irgendwie nicht in die "Verteilte Komponenten" Umgebung
Demonstration Mobile Clients
Anbindung REST Service
sprengt Rahmen wahrscheinlich
Servicelayer
Spring
DI
MVC
Beans
...
Hoher Aufwand um neue Konzepte zu verstehen
Bean-Binding
"Plain" JEE
REST Service wahrscheinlich viel Aufwand
Dafür weniger Frameworks
JPA
Hibernate?
EclipseLink
SA-Architektur
Jersey (REST Referenzimplementation)
DI
Google Guice
evtl. anderes Framework?
Codehaus Jackson (JSON Serializer)
Eclipse Link JPA
Viele Frameworks
Datalayer
PostgreSQL
SQLite
MySQL
... eigentlich egal; JPA abstrahiert
All-in-ones
Vaadin
Man lernt fürs Framework
Viel Architektur wird maskiert/versteckt
Play! Framework
Verteilt
ziemlich viel Overload für "simple" REST Services
"Monolithic"
Viel Architektur wird maskiert/versteckt
Man lernt fürs Framework
mindmeister

Anhang F Coding Guideline

Dieses Kapitel enthält die JavaScript Coding Guideline, welche zur Erstellung von entsprechendem Quellcode im Rahmen dieser Bachelorarbeit benutzt wird.

Die Guideline basiert auf einem von Airbnb [Aira] veröffentlichten Dokument. Sie wurde zusätzlich gemäss eigenen Präferenzen [Airb] des Projektteams erweitert und optimiert.

Original Repository: airbnb/javascript

Airbnb JavaScript Style Guide() {

Ein vernünftiger Ansatz für einen JavaScript-Style-Guide

Inhaltsverzeichnis

Datentypen

- **Primitive Typen**: Bei primitiven Datentypen wird immer direkt auf deren Wert zugegriffen.
  - `string`
  - `number`
  - `boolean`
  - `null`
  - `undefined`

```
var foo = 1
    ,bar = foo;

bar = 9;

console.log(foo, bar); // => 1, 9
```

- **Komplexe Typen**: Bei komplexen Datentypen wird immer auf eine Referenz zugegriffen.
  - `object`
  - `array`

- `function`

```
var foo = [1, 2]
    ,bar = foo;

bar[0] = 9;

console.log(foo[0], bar[0]); // => 9, 9
```

[⬆]

Objekte

- Benutze die `literal syntax`, um Objekte zu erzeugen.

```
// schlecht
var item = new Object();

// gut
var item = {};
```

- Benutze keine reservierten Wörter für Attribute.

```
// schlecht
var superman = {
  class: 'superhero'
  ,default: { clark: 'kent' }
  ,private: true
};

// gut
var superman = {
  klass: 'superhero'
  ,defaults: { clark: 'kent' }
  ,hidden: true
};
```

[⬆]

Arrays

- Benutze die `literal syntax`, um Arrays zu erzeugen.

```
// schlecht
var items = new Array();

// gut
var items = [];
```

- Wenn du die Array-Länge nicht kennst, benutze `Array#push`.

```
var someStack = [];

// schlecht
someStack[someStack.length] = 'abracadabra';

// gut
someStack.push('abracadabra');
```

- Wenn du ein Array kopieren möchtest, benutze `Array#slice`. jsPerf

```
var len = items.length
    ,itemsCopy = []
    ,i;
```

```
// schlecht
for (i = 0; i < len; i++) {
  itemsCopy[i] = items[i];
}

// gut
itemsCopy = Array.prototype.slice.call(items);
```

[⬆]

Zeichenketten

- Benutze einfache Anführungszeichen `''` für Zeichenketten

```
// schlecht
var name = "Bob Parr";

// gut
var name = 'Bob Parr';

// schlecht
var fullName = "Bob " + this.lastName;

// gut
var fullName = 'Bob ' + this.lastName;
```

- Zeichenketten die länger als 80 Zeichen lang sind, sollten mit Hilfe von `string concatenation` auf mehrere Zeilen aufgeteilt werden.
- Beachte: Benutzt man `string concatenation` zu oft kann dies die performance beeinträchtigen. jsPerf & Discussion

```
// schlecht
var errorMessage = 'This is a super long error that was thrown because of Batman. When you stop to think about hou

// schlecht
var errorMessage = 'This is a super long error that \
was thrown because of Batman. \
When you stop to think about \
how Batman had anything to do \
with this, you would get nowhere \
fast.';

// gut
var errorMessage = 'This is a super long error that ' +
  'was thrown because of Batman.' +
  'When you stop to think about ' +
  'how Batman had anything to do ' +
  'with this, you would get nowhere ' +
  'fast.';
```

- Wenn man im Programmverlauf eine Zeichenkette dynamisch zusammensetzen muss, sollte man `Array#join` einer `string concatenation` vorziehen. Vorallem für den IE. jsPerf.

```
var items
    ,messages
    ,length, i;

messages = [{
    state: 'success'
    ,message: 'This one worked.'
},{
    state: 'success'
    ,message: 'This one worked as well.'
},{
    state: 'error'
    ,message: 'This one did not work.'
}];
```

```
length = messages.length;

// schlecht
function inbox(messages) {
  items = '<ul>';

  for (i = 0; i < length; i++) {
    items += '<li>' + messages[i].message + '</li>';
  }

  return items + '</ul>';
}

// gut
function inbox(messages) {
  items = [];

  for (i = 0; i < length; i++) {
    items[i] = messages[i].message;
  }

  return '<ul><li>' + items.join('</li><li>') + '</li></ul>';
}
```

[⬆]

Funktionen

- Funktionsausdrücke:

```
// anonyme Fuktionsausdrücke
var anonymous = function() {
  return true;
};

// benannte Funktionsausdrücke
var named = function named() {
  return true;
};

// direkt ausgeführte Funktionsausdrücke (IIFE)
(function() {
  console.log('Welcome to the Internet. Please follow me.');
})();
```

- Vermeide Funktionen in `non-function blocks` zu deklarieren. Anstelle sollte die Funktion einer Variablen zugewiesen werden. Dies hat den Grund, dass die verschiedenen Browser dies unterschiedlich interpretieren.

- **Beachte:** ECMA-262 definiert einen Block als eine Abfolge/Liste von Statements. Eine Funktion hingegen ist **kein** Statement. Read ECMA-262's note on this issue.

```
// schlecht
if (currentUser) {
  function test() {
    console.log('Nope.');
  }
}

// gut
if (currentUser) {
  var test = function test() {
    console.log('Yup.');
  };
}
```

- Benenne einen Parameter nie `arguments`, denn dies wird das `arguments`-Objekt, dass in jedem Funktionskörper zur Verfügung

steht, überschreiben.

```
// schlecht
function nope(name, options, arguments) {
  // ...stuff...
}

// gut
function yup(name, options, args) {
  // ...stuff...
}
```

[⬆]

Eigenschaften

- Benutze die Punktnotation, um auf die Eigenschaften eines Objekts zuzugreifen.

```
var luke = {
  jedi: true
  ,age: 28
};

// schlecht
var isJedi = luke['jedi'];

// gut
var isJedi = luke.jedi;
```

- Benutze die Indexnotation `[]`, um auf die Eigenschaften eines Objekts zuzugreifen, sofern der Index eine Variable ist.

```
var luke = {
  jedi: true
  ,age: 28
};

function getProp(prop) {
  return luke[prop];
}

var isJedi = getProp('jedi');
```

[⬆]

Variablen

- Benutze immer `var`, um Variablen zu deklarieren. Tut man dies nicht, werden die Variablen im globalen Namespace erzeugt – was nicht gewüscht werden sollte.

```
// schlecht
superPower = new SuperPower();

// gut
var superPower = new SuperPower();
```

- Benutze immer nur ein `var`, um mehrere aufeinanderfolgende Variablen zu deklarieren. Deklariere jede Variable auf einer eigenen Zeile.

```
// schlecht
var items = getItems();
var goSportsTeam = true;
var dragonball = 'z';

// gut
var items = getItems()
```

```
    ,goSportsTeam = true
    ,dragonball = 'z';
```

- Deklariere Variablen ohne direkte Zuweisung immer als letztes. Dies ist vorallem hilfreich, wenn man später eine Variable anhand einer zuvor deklarierten Variable initialisieren möchte.

```
// schlecht
var i, len, dragonball
    ,items = getItems()
    ,goSportsTeam = true;

// schlecht
var i, items = getItems()
    ,dragonball
    ,goSportsTeam = true
    ,len;

// gut
var items = getItems()
    ,goSportsTeam = true
    ,dragonball
    ,i
    ,length;
```

- Weise den Wert einer Variable, wenn möglich, immer am Anfang des Gültigkeitsbereichs zu. Dies hilft Problemen mit der Variablendeklaration vorzubeugen.

```
// schlecht
function() {
  test();
  console.log('doing stuff..');

  //..other stuff..

  var name = getName();

  if (name === 'test') {
    return false;
  }

  return name;
}

// gut
function() {
  var name = getName();

  test();
  console.log('doing stuff..');

  //..other stuff..

  if (name === 'test') {
    return false;
  }

  return name;
}

// schlecht
function() {
  var name = getName();

  if (!arguments.length) {
    return false;
  }

  return true;
}
```

```
// gut
function() {
  if (!arguments.length) {
    return false;
  }

  var name = getName();

  return true;
}
```

[⬆]

Hoisting

- Variablendeklarationen werden vom Interpreter an den Beginn eines Gültigkeitbereichs genommen, genannt (`hoisting`). Wohingegen die Zuweisung an der ursprünglichen Stelle bleibt.

```
// Dies wird nicht funktionen (angenommen
// notDefined ist keine globale Variable)
function example() {
  console.log(notDefined); // => throws a ReferenceError
}

// Wird eine Variable nach seiner ersten
// Referenzierung deklariert, funktioniert
// dies dank des hoistings.
// Beachte aber, dass die Zuweisung von true
// erst nach der Referenzierung stattfindet.
function example() {
  console.log(declaredButNotAssigned); // => undefined
  var declaredButNotAssigned = true;
}

// Der Interpreter nimmt die Variablendeklaration
// an den Beginn des Gültigkeitbereichs.
// So kann das Beispiel wiefolgt umgeschrieben
// werden:
function example() {
  var declaredButNotAssigned;
  console.log(declaredButNotAssigned); // => undefined
  declaredButNotAssigned = true;
}
```

- Anonyme Funktionen `hoisten` ihren Variablennamen, aber nicht die Funktionszuweisung.

```
function example() {
  console.log(anonymous); // => undefined

  anonymous(); // => TypeError anonymous is not a function

  var anonymous = function() {
    console.log('anonymous function expression');
  };
}
```

- Benannte Funktionen `hoisten` ihren Variablennamen, aber nicht der Funktionsname oder Funktionskörper.

```
function example() {
  console.log(named); // => undefined

  named(); // => TypeError named is not a function

  superPower(); // => ReferenceError superPower is not defined

  var named = function superPower() {
```

```
    console.log('Flying');
  };

  // Das gleiche gilt, wenn der Funktionsname
  // derselbe ist, wie der Variablenname
  function example() {
    console.log(named); // => undefined

    named(); // => TypeError named is not a function

    var named = function named() {
      console.log('named');
    };
  }
}
```

- Funktionsdeklarationen `hoisten` ihren Namen und ihren Funktionskörper.

```
function example() {
  superPower(); // => Flying

  function superPower() {
    console.log('Flying');
  }
}
```

- Für weitere Informationen siehe hier: JavaScript Scoping & Hoisting by Ben Cherry

[⬆]

Bedingungen und Gleichheit

- Ziehe `===` und `!==` gegenüber `==` und `!=` vor.
- Bedingungsausdrücke werden immer gezwungen der `ToBoolean` Methode ausgewertet zu werden. Diese folgt den folgenden einfachen Grundregeln:
  - **Objekte** werden als **true** gewertet
  - **Undefined** wird als **false** gewertet
  - **Null** wird als **false** gewertet
  - **Booleans** werden als **der Wert des Booleans** gewertet
  - **Zahlen** werden als **false** gewertet sofern **+0, -0, or NaN**, ansonsten als **true**
  - **Zeichenketten** werden als **false** gewertet, sofern sie leer ist `''`, ansonsten als **true**

```
if ([0]) {
  // true
  // Arrays sind Objekte und Objekte werden als true ausgewertet
}
```

- Benutze `shortcuts`

```
// schlecht
if (name !== '') {
  // ...stuff...
}

// gut
if (name) {
  // ...stuff...
}

// schlecht
if (collection.length > 0) {
  // ...stuff...
}

// gut
if (collection.length) {
```

```
  // ...stuff...
}
```

- Für weitere Informationen siehe hier: Truth Equality and JavaScript by Angus Croll

[⬆]

Blöcke

- Benutze geschweifte Klammern für alle mehrzeiligen Blöcke.

```
// schlecht
if (test)
  return false;

// gut
if (test) return false;

// gut
if (test) {
  return false;
}

// schlecht
function() { return false; }

// gut
function() {
  return false;
}
```

[⬆]

Kommentare

- Benutze `/** ... */` für mehrzeilige Kommentare. Daran kann eine Beschreibung, eine Typendefinition und Werte für alle Parameter und den Rückgabewert angegeben werden.

```
// schlecht
// make() returns a new element
// based on the passed in tag name
//
// @param <String> tag
// @return <Element> element
function make(tag) {

  // ...stuff...

  return element;
}

// gut
/**
 * make() returns a new element
 * based on the passed in tag name
 *
 * @param <String> tag
 * @return <Element> element
 */
function make(tag) {

  // ...stuff...

  return element;
}
```

- Benutze `//` für einzeilige Kommentare. Platziere den Kommentar auf einer separaten Zeile oberhalb der beschriebenen Zeile. Vor

den Kommentar kommt eine Leerzeile.

```
// schlecht
var active = true;  // is current tab

// gut
// is current tab
var active = true;

// schlecht
function getType() {
  console.log('fetching type...');
  // set the default type to 'no type'
  var type = this._type || 'no type';

  return type;
}

// gut
function getType() {
  console.log('fetching type...');

  // set the default type to 'no type'
  var type = this._type || 'no type';

  return type;
}
```

[⬆]

Whitespace

- Benutze weiche Tabulatoren (`soft tabs`) mit 2 Leerzeichen.

```
// schlecht
function() {
····var name;
}

// schlecht
function() {
·var name;
}

// gut
function() {
··var name;
}
```

- Platziere ein Leerzeichen vor einer öffnenden Klammer.

```
// schlecht
function test(){
  console.log('test');
}

// gut
function test() {
  console.log('test');
}

// schlecht
dog.set('attr',{
  age: '1 year'
  ,breed: 'Bernese Mountain Dog'
});

// gut
```

```
dog.set('attr', {
  age: '1 year'
  ,breed: 'Bernese Mountain Dog'
});
```

- Platziere eine Leerzeile an das Ende der Datei.

```
// schlecht
(function(global) {
  // ...stuff...
})(this);
```

```
// gut
(function(global) {
  // ...stuff...
})(this);
```

- Rücke bei langen Methodenverkettungen ein.

```
// schlecht
$('#items').find('.selected').highlight().end().find('.open').updateCount();

// gut
$('#items')
  .find('.selected')
    .highlight()
    .end()
  .find('.open')
    .updateCount();

// schlecht
var leds = stage.selectAll('.led').data(data).enter().append("svg:svg").class('led', true)
    .attr('width',  (radius + margin) * 2).append("svg:g")
    .attr("transform", "translate(" + (radius + margin) + "," + (radius + margin) + ")")
    .call(tron.led);

// gut
var leds = stage.selectAll('.led')
    .data(data)
  .enter().append("svg:svg")
    .class('led', true)
    .attr('width',  (radius + margin) * 2)
  .append("svg:g")
    .attr("transform", "translate(" + (radius + margin) + "," + (radius + margin) + ")")
    .call(tron.led);
```

[⬆]

Führende Kommas

- **Ja.**

```
// schlecht
var once,
    upon,
    aTime;

// gut
var once
    ,upon
    ,aTime;

// schlecht
var hero = {
  firstName: 'Bob',
  lastName: 'Parr',
```

```
  heroName: 'Mr. Incredible',
  superPower: 'strength'
};

// gut
var hero = {
    firstName: 'Bob'
    ,lastName: 'Parr'
    ,heroName: 'Mr. Incredible'
    ,superPower: 'strength'
};
```

[⬆]

Semikolons

- **Ja.**

```
// schlecht
(function() {
  var name = 'Skywalker'
  return name
})()

// gut
(function() {
  var name = 'Skywalker';
  return name;
})();
```

[⬆]

Typumwandlung

- Benutze `type coercion` am Anfang eines Statements.
- Bei Zeichenketten:

```
//  => this.reviewScore = 9;

// schlecht
var totalScore = this.reviewScore + '';

// gut
var totalScore = '' + this.reviewScore;

// schlecht
var totalScore = '' + this.reviewScore + ' total score';

// gut
var totalScore = this.reviewScore + ' total score';
```

- Benutze immer `parseInt` für Zahlen und gebe immer eine Basis für die Typumwandlung an.
- Wenn man aus Performancegründen kein `parseInt` verwenden will und ein `Bitshifting` benutzt, sollte man einen Kommentar hinterlassen, wieso dies gemacht wurde.

```
var inputValue = '4';

// schlecht
var val = new Number(inputValue);

// schlecht
var val = +inputValue;

// schlecht
var val = inputValue >> 0;
```

```
// schlecht
var val = parseInt(inputValue);

// gut
var val = Number(inputValue);

// gut
var val = parseInt(inputValue, 10);

// gut
/**
 * parseInt was the reason my code was slow.
 * Bitshifting the String to coerce it to a
 * Number made it a lot faster.
 */
var val = inputValue >> 0;
```

- Bei Booleans:

```
var age = 0;

// schlecht
var hasAge = new Boolean(age);

// gut
var hasAge = Boolean(age);

// gut
var hasAge = !!age;
```

[⬆]

Namenskonventionen

- Benutze keine `einzeichigen` Namen. Die Namen sollten beschreibend sein.

```
// schlecht
function q() {
  // ...stuff...
}

// gut
function query() {
  // ..stuff..
}
```

- Benutze `camelCase`, um Objekte, Funktionen und Instanzen zu benennen.

```
// schlecht
var OBJEcttssss = {};
var this_is_my_object = {};
var this-is-my-object = {};
function c() {};
var u = new user({
  name: 'Bob Parr'
});

// gut
var thisIsMyObject = {};
function thisIsMyFunction() {};
var user = new User({
  name: 'Bob Parr'
});
```

- Benutze `PascalCase`, um Klassen und Konstrukturen zu benennen.

```
// schlecht
function user(options) {
  this.name = options.name;
}

var bad = new user({
  name: 'nope'
});

// gut
function User(options) {
  this.name = options.name;
}

var good = new User({
  name: 'yup'
});
```

- Benutze führende Untenstriche `_`, um private Eigenschaften zu benennen.

```
// schlecht
this.__firstName__ = 'Panda';
this.firstName_ = 'Panda';

// gut
this._firstName = 'Panda';
```

- Um eine Referenz an `this` zuzuweisen, benutze `_this`.

```
// schlecht
function() {
  var self = this;
  return function() {
    console.log(self);
  };
}

// schlecht
function() {
  var that = this;
  return function() {
    console.log(that);
  };
}

// gut
function() {
  var _this = this;
  return function() {
    console.log(_this);
  };
}
```

- Gib deinen Funktionen einen Namen. Dies ist hilfreich für den `stack trace`.

```
// schlecht
var log = function(msg) {
  console.log(msg);
};

// gut
var log = function log(msg) {
  console.log(msg);
};
```

[⬆]

Zugriffsmethoden

- Zugriffsmethoden für Objekteigenschaften sind nicht von Nöten.
- Macht man dennoch Zugriffsmethoden, benutze `getVal()` und `setVal('hello')`.

```
// schlecht
dragon.age();

// gut
dragon.getAge();

// schlecht
dragon.age(25);

// gut
dragon.setAge(25);
```

- Wenn die Eigenschaft ein Boolean ist, benutze `isVal()` oder `hasVal()`.

```
// schlecht
if (!dragon.age()) {
  return false;
}

// gut
if (!dragon.hasAge()) {
  return false;
}
```

- Es ist in Ordnung `get()`- und `set()`-Methoden zu erstellen, aber sei konsistent.

```
function Jedi(options) {
  options || (options = {});
  var lightsaber = options.lightsaber || 'blue';
  this.set('lightsaber', lightsaber);
}

Jedi.prototype.set = function(key, val) {
  this[key] = val;
};

Jedi.prototype.get = function(key) {
  return this[key];
};
```

[⬆]

Konstruktoren

- Weise die Methoden dem `prototype` des Objektes zu, anstelle den `prototype` mit einem neuen Objekt zu überschreiben. Wenn man den `prototype` überschreibt wird eine Vererbung unmöglich, denn damit wird die Basis überschrieben!

```
function Jedi() {
  console.log('new jedi');
}

// schlecht
Jedi.prototype = {
  fight: function fight() {
    console.log('fighting');
  }

  ,block: function block() {
    console.log('blocking');
  }
};
```

```
// gut
Jedi.prototype.fight = function fight() {
  console.log('fighting');
};

Jedi.prototype.block = function block() {
  console.log('blocking');
};
```

- Methoden können `this` zurückgeben, um eine Methodenverkettung zu ermöglichen.

```
// schlecht
Jedi.prototype.jump = function() {
  this.jumping = true;
  return true;
};

Jedi.prototype.setHeight = function(height) {
  this.height = height;
};

var luke = new Jedi();
luke.jump(); // => true
luke.setHeight(20) // => undefined

// gut
Jedi.prototype.jump = function() {
  this.jumping = true;
  return this;
};

Jedi.prototype.setHeight = function(height) {
  this.height = height;
  return this;
};

var luke = new Jedi();

luke.jump()
  .setHeight(20);
```

- Es ist in Ordnung eine eigene `toString()`-Methode zu schreiben, aber man sollte sicherstellen, dass diese korrekt funktioniert und keine Nebeneffekte hat.

```
function Jedi(options) {
  options || (options = {});
  this.name = options.name || 'no name';
}

Jedi.prototype.getName = function getName() {
  return this.name;
};

Jedi.prototype.toString = function toString() {
  return 'Jedi - ' + this.getName();
};
```

[⬆]

Module

- Ein Modul sollte mit einem `!` beginnen. Dies stellt sicher, dass wenn in einem Modul das abschliessende Semikolon vergessen wurde, keine Fehler entstehen, wenn die Scripte zusammengeschnitten werden.
- Eine Datei sollte in `camelCase` benannt sein, in einem Ordner mit dem selben Namen liegen und dem Namen entsprechen mit dem es exportiert wird.
- Benutze eine Methode `noConflict()`, welche das exportierte Modul auf die vorhergehende Version setzt und diese zurück gibt.

- Deklariere immer `'use strict';` am Anfang des Moduls.

```
// fancyInput/fancyInput.js

!function(global) {
  'use strict';

  var previousFancyInput = global.FancyInput;

  function FancyInput(options) {
    this.options = options || {};
  }

  FancyInput.noConflict = function noConflict() {
    global.FancyInput = previousFancyInput;
    return FancyInput;
  };

  global.FancyInput = FancyInput;
}(this);
```

[⬆]

jQuery

- Stelle allen jQuery-Objektvariablen ein `$` voran.

```
// schlecht
var sidebar = $('.sidebar');

// gut
var $sidebar = $('.sidebar');
```

- Speichere `jQuery lookups`, sofern sie mehrmals gebraucht werden.

```
// schlecht
function setSidebar() {
  $('.sidebar').hide();

  // ...stuff...

  $('.sidebar').css({
    'background-color': 'pink'
  });
}

// gut
function setSidebar() {
  var $sidebar = $('.sidebar');
  $sidebar.hide();

  // ...stuff...

  $sidebar.css({
    'background-color': 'pink'
  });
}
```

- Für DOM-Abfragen benutze `Cascading` : `$('.sidebar ul')` oder parent > child `$('.sidebar > .ul')` . jsPerf

- Benutze `find` mit `scoped jQuery object queries`

```
// schlecht
$('.sidebar', 'ul').hide();

// schlecht
$('.sidebar').find('ul').hide();
```

```
// gut
$('.sidebar ul').hide();

// gut
$('.sidebar > ul').hide();

// gut (langsamer)
$sidebar.find('ul');

// gut (schneller)
$($sidebar[0]).find('ul');
```

[⬆]

ECMAScript 5 Kompatibilität

- Verweis auf Kangax's ES5 Kompatibilitätstabelle

[⬆]

Testing

- **Ja.**

```
function() {
  return true;
}
```

[⬆]

Performance

- On Layout & Web Performance
- String vs Array Concat
- Try/Catch Cost In a Loop
- Bang Function
- jQuery Find vs Context, Selector
- innerHTML vs textContent for script text
- Long String Concatenation
- Loading...

[⬆]

Ressourcen

Lese dieses

- Annotated ECMAScript 5.1

Andere Styleguides

- Google JavaScript Style Guide
- jQuery Core Style Guidelines
- Principles of Writing Consistent, Idiomatic JavaScript

Andere Styles

- Naming this in nested functions - Christian Johansen
- Conditional Callbacks

Bücher

- JavaScript: The Good Parts - Douglas Crockford

- JavaScript Patterns - Stoyan Stefanov
- Pro JavaScript Design Patterns - Ross Harmes and Dustin Diaz
- High Performance Web Sites: Essential Knowledge for Front-End Engineers - Steve Souders
- Maintainable JavaScript - Nicholas C. Zakas
- JavaScript Web Applications - Alex MacCaw
- Pro JavaScript Techniques - John Resig
- Smashing Node.js: JavaScript Everywhere - Guillermo Rauch

Blogs

- DailyJS
- JavaScript Weekly
- JavaScript, JavaScript...
- Bocoup Weblog
- Adequately Good
- NCZOnline
- Perfection Kills
- Ben Alman
- Dmitry Baranovskiy
- Dustin Diaz
- nettuts

[⬆]

In the Wild

Dies ist eine Liste von Organisationen, welche diesen Style Guide benutzen. Sende uns einen `Pull request` oder öffne einen `issue` und wir werden dich der Liste hinzufügen.

- **Airbnb**: airbnb/javascript
- **American Insitutes for Research**: AIRAST/javascript
- **ExactTarget**: ExactTarget/javascript
- **GoCardless**: gocardless/javascript
- **GoodData**: gooddata/gdc-js-style
- **How About We**: howaboutwe/javascript
- **MinnPost**: MinnPost/javascript
- **National Geographic**: natgeo/javascript
- **Razorfish**: razorfish/javascript-style-guide
- **Shutterfly**: shutterfly/javascript

[⬆]

Übersetzungen

Dieser Styleguide ist in den folgenden Sprachen erhältlich:

- :en: **Englisch**: airbnb/javascript
- **Japanisch**: mitsuruog/javacript-style-guide

[⬆]

The JavaScript Style Guide Guide

- Reference

[⬆]

Contributors

- View Contributors

[⬆]

Lizenz

[⬆]

};

Anhang G Aufgabenstellung

Die folgenden drei Seiten enthalten die offizielle Aufgabenstellung dieser Bachelorarbeit.

Aufgabenstellung Bachelorarbeit für Manuel Alabor, Alexandre Joly und Michael Weibel „Architekturkonzepte moderner Web-Applikationen“

1. Auftraggeber, Betreuer und Experte

Bei dieser Arbeit handelt es sich um eine HSR-interne Arbeit zur Unterstützung des Moduls Internettechnologien.

Auftraggeber/Betreuer:

- Prof. Hans Rudin, HSR, IFS hrudin@hsr.ch +41 55 222 49 36 (Verantw. Dozent, Betreuer)
- Kevin Gaunt, HSR,IFS kgaunt@hsr.ch +41 55 222 4662 (Betreuer)

Experte:

- Daniel Hiltebrand, Crealogix

2. Studierende

Diese Arbeit wird als Bachelorarbeit an der Abteilung Informatik durchgeführt von

- Manuel Alabor malabor@hsr.ch
- Alexandre Joly ajoly@hsr.ch
- Michael Weibel mweibel@hsr.ch

3. Ausgangslage

Das Modul Internettechnologien ist stark Technologie-zentriert. Wünschbar ist eine Weiterentwicklung (Aktualisierung, Verbesserung) mit vermehrter Beachtung von konzeptionellen und Architektur-Fragen. In letzter Zeit haben sich Prinzipien und Konzepte herauskristallisiert, nach denen Web-Applikationen am besten aufgebaut werden. Siehe zum Beispiel [1] oder [2]. Um diese Prinzipien und Konzepte anschaulich zu vermitteln, braucht es neben Erläuterungen möglichst anschauliche Beispiele und Übungsaufgaben. Ziel dieser Arbeit ist es, die Weiterentwicklung des Moduls Internettechnologien entsprechend zu unterstützen.

4. Aufgabenstellung

In dieser Arbeit sollen die in [1], [2] und weiteren Quellen dargestellten Prinzipien und Konzepte analysiert werden. Gemeinsam mit dem Betreuer sollen daraus in das Modul Internettechnologien zu transferierende Prinzipien und Konzepte ausgewählt werden, und es soll überlegt werden, wie diese Inhalte anschaulich für den Unterricht aufbereitet werden können. In der Folge sollten entsprechende Resultate erarbeitet werden, welche das Unterrichten der ausgewählten Inhalte möglichst gut unterstützen. Eine wichtige Rolle dürfte dabei eine anschauliche Beispielapplikation bilden.

Details werden im Verlauf der Arbeit zwischen Studierenden und Betreuer vereinbart.

5. Zur Durchführung

Mit dem Betreuer finden wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen.

Alle Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten, die Besprechung ist durch die Studierenden zu leiten und die Ergebnisse sind in einem Protokoll festzuhalten, das den Betreuern und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Alle Resultate sind vollständig auf CD/DVD in 3 Exemplaren abzugeben. Der Bericht ist ausgedruckt in doppelter Ausführung abzugeben.

7. Referenzen

[1] Stefan Tilkov
Building large web-based systems: 10 Recommendations
Präsentation an der OOP 2013, München
PDF als Beilage

[2] http://roca-style.org
ROCA Resource-oriented Client Architecture - A collection of simple recommendations for decent Web application frontends

8. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html.

Montag, den 18. Februar 2013	**Beginn der Bachelorarbeit,** **Ausgabe der Aufgabenstellung durch die Betreuer**
7. Juni 2013	Abgabe Kurzbeschreibung und A0-Poster. Vorlagen stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
14. Juni 2013, 12:00	**Abgabe** der Arbeit an den Betreuer bis 12.00 Uhr. Fertigstellung des A0-Posters bis 12.00 Uhr. Abgabe der Posters im Abteilungssekretariat 6.113.
14. Juni 2012	HSR-Forum, Vorträge und Präsentation der Bachelor- und Diplomarbeiten, 16 bis 20 Uhr
5.8. - 23.08.2013	Mündliche Prüfung zur Bachelorarbeit

9. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von 30 Stunden budgetiert (Siehe auch Modulbeschreibung der Bachelorarbeit https://unterricht.hsr.ch/staticWeb/allModules/19419_M_BAI.html.).

Für die Beurteilung ist der HSR-Betreuer verantwortlich.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/6
3. Inhalt*)	3/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit mit den Studierenden festgelegt.

Im Übrigen gelten die Bestimmungen der Abteilung Informatik für Bachelorarbeiten.

Rapperswil, den 20. Februar 2013

Prof. Hans Rudin
Institut für Software
Hochschule für Technik Rapperswil

Anhang H **Meetingprotokolle**

Bachelorarbeit Vorbesprechung 14. Februar 2013

Teilnehmer

- Hans Rudin, HRU (HSR)
- Kevin Gaunt, KGA (HSR)
- Daniel Hiltebrand, DHI (Crealogix)
- Manuel Alabor, MAL (Team)
- Alexandre Joly, AJO (Team)
- Michael Weibel, MWE (Team, Protokoll)

Traktanden

- StoryboardBuilder - Was ist der aktuelle Stand?
- Oder was habt ihr für Ideen?

Meeting

StoryboardBuilder

Einführung DHI

- Ende Januar Entscheid: Nicht weiterentwickeln
- Allerdings nicht weil Produkt/Markt nicht interessant wäre
- Ziel war: damit die Crealogix UX-Services zu unterstützen
- Crealogix wird keine UX-Services gegen aussen mehr anbieten
    - mehr interne Projekte betreuen

- Zwei Schwerpunkte: Education & Financial Services

    - StoryboardBuilder gehört nicht in einen solchen Schwerpunkt

- Dies obwohl gutes Potential gesehen wird für das Produkt
- Anforderungsspezifikation verfeinert bis Ende Januar
- Technischer Prototyp gestartet, aber wieder gestoppt aufgrund der Neuorientierung
- Idee wäre: Storyboardbuilder an externe Firma weitergeben
- Bestehende Mitbewerber bauen ihre Angebote aus

Diskussion BA

- Frage an die Runde: ist es interessant für euch, den Storyboardbuilder in der BA weiter zuentwickeln?
    - MAL: Wie würde das aussehen?
        - DHI: Beispiel aufgrund gewählter Technologie zu entwickeln
        - DHI: Auf Basis der fachlichen Spezifikation der Crealogix
        - HRU: BA sollte nicht zu einer Fleissarbeit werden
        - HRU: Was wären denn die Herausforderungen wenn das weiterentwickelt werden würde in BA?
        - HRU: Die momentane Ausgangslage ist anders, da kein wirklicher Kunde existiert
        - DHI: Es sind sicher einige Ideen da, die technisch Herausfordernd sind
        - DHI: Grafische Repräsentation auf Screen, Objektmodell umsetzen
        - DHI: Wie gesagt, hat auch nichts dagegen, wenn eine neue Arbeit gemacht werden würde
        - DHI: Laut Kenntnisstand von DHI sollte auch von seiten Industriepartner weniger Betreuung beinhalten, mehr von Team
- KGA: Was ist nun der Anspruch an das Meeting? Müssen wir am Ende des Meetings schon wissen was gemacht werden soll?
    - DHI: Ist offen, hat keinen Anspruch auf Entscheid jetzt - sollte aber bald geschehen, da Bachelorarbeit bald startet
    - DHI:

- DHI: Hat div. Alternativen die man anschauen könnte
- MAL: für ihn ist das Durchführen der BA mit dem Storyboardbuilder nicht mehr besonders interessant, aufgrund Änderung seitens Crealogix
    - DHI: Ja das stimmt, BA würde nicht mehr zu einem realen Produkt führen

- MWE: gehts ähnlich wie MAL
- DHI: Thema 1:
    - **Kino reservations system**
    - Kennt Kinobesitzer in rapperswil
    - Buchungssystem
    - Mit anbindung an Kassensystem
    - nicht nur einzelne Plätze
    - sondern auch Firmenanlässe, Frauenkino, Catering etc.
    - Konzeptionell entwerfen
    - soweit wie möglich implementieren
    - könnte sehr interessant sein, DHI's Meinung nach

- DHI: Thema 2
    - **Personal Finance Management**
    - Zusatz zu Ebanking
    - Eigene Zahlungen analysieren
    - Verschiedene Banken
    - Soviel für Versicherung, Einkauf, etc.
    - Basiert auf einer isländisch-schwedischen Firma
    - Integrationsarbeit (in ein Bankensystem einbauen)
    - MAL: Geht es darum, das zu integrieren und anschaulich darzustellen, und Data-Mining ist schon gemacht?
    - DHI: Ja
        - Schwierigkeit Sicherheit
        - Zertifizierung, Authentifizierung
        - nur Teilaspekt möglich zum lösen
        - Versch. Komponenten
        - Aspekt wichtig auf welcher sich die BA konzentrieren soll

    - DHI: Müsste das genauer anschauen wie das machbar wäre
        - Da es ein sehr grosses System wäre
        - Müsste verifizieren ob das gehen soll?

- HRU: Thema 2 wohl eher interessant (Team bejaht)
  - HRU: Wäre das so kurzfristig machbar?
    - DHI: müsste angeschaut werden
    - DHI: Verträge bestehen
    - DHI: anderes ist Ebanking in Java mit Oracle
      - DHI: Verfügbar machen möglich
  - HRU: Zwei Komponenten, was ist Webfrontend?
    - DHI: Netty server von airlock für authentifizierung
    - DHI: möglichst HTML das übers web geht
    - DHI: J2EE - JSP vorne
  - HRU: Isländische Software
    - DHI: .NET basiert
    - DHI: hat aber ein WCF/REST/AJAX schnittstelle welche relativ gut ins Frontend integrierbar wäre
    - DHI: von eienm System ins andere System transferieren (Oracle zu MSSQL DB)
    - DHI: Statistisch aufwerten, und wieder anzeigen
    - DHI: machbarkeit unklar, muss verifiziert werden
- MAL: Was hat HRU für Projekte
  - MAL: Realtime sachen wären interessant (Mobile, Chat, Messaging?)
  - HRU: hat keine Projekte
- DHI: Hat evtl. noch andere Projekte, müsste das aber noch anschauen
- MAL: Bis Testumgebung steht würden wohl wochen vergehen
  - DHI: stimmt wohl
- MWE: Persönlich intressiert vorallem WebRTC
- HRU: was ist mit web realtimecommunication (WebRTC) gemeint?
- MWE: Near-realtime communication (Daten, Video, Audio) zw. Browsern
- MWE: Was wäre denn für Crealogix interessant - zentrale Frage?
  - DHI: Crealogix muss nicht unbedingt dabei sein, wenn nicht nötig
- DHI: update maus-scanner
  - MAL: wäre denn mobile auch ein Thema?
  - DHI: Mobile ist sehr zentral

- HRU: gibt es fraktionen (web/mobile) im Team?
  - MWE: Web-Mensch, aber Mobile wäre auch sehr interessant

- DHI: Fragt bei Crealogix CEO/entwicklungsleiter nach bzgl. Mobile
- MAL: Elearning wäre auch interessant bzlg. Mobile
  - DHI: könnte nachfragen obs da auch was geben würde?

- AJO: Auch vorallem Mobile interessant
  - arbeitet auch vorallem in Mobile
  - HRU: Auch sie MAL ;)

- DHI: müsste nachfragen, aber wäre sicher interessant
- MAL: Wäre sicher interessant auf DHI's Themen zu warten, andererseits müsste auch Teamintern bzw. mit KGA/HRU geschaut werden
- DHI: wann beginnt die Arbeit? - Montag
- DHI: 3 Bereiche Education
  - Campusmanagement
  - Time to learn
    - 30'000 die mit TTL arbeiten
    - Mit Center for young professional zusammenarbeit (evtl. da was interssantes)
    - In Bubikon

- DHI: was machen wenn nichts herauskommt?
  - DHI: würden gerne mit euch zusammenarbeiten, wenn möglich

- HRU: Wäre schon der Weg zum gehen
  - parallel müssten überlegungen angestellt werden obs was anderes geben würde
  - gibt den 3 Studierenden möglichst freie Hand

- KGA: Termin bis wann die Entscheidung möglich sein
  - HRU: allerspätistens erste Woche
  - DHI: wird noch heute mit den 3 Crealogix leuten schauen
  - DHI: Bis morgen, 15.02. Antwort wenn möglich
  - MAL: Mittwoch zu spät oder zu früh?
  - HRU: Wann sind sie @HSR?
    - MAL: MO/DI/MI

    - HRU: Mittwoch wäre nicht unbedingt zu spät
    - MAL: Mittwoch wäre Zusammenkunft, um definitiv zu entscheiden. Aber mit Kommunikation bis dann
    - MAL: DHI kann sicher schnell entscheiden, je nach dem wäre pers. Anwesenheit nicht nötig
    - DHI: würde gerne persönlich dabei sein

- DHI: gibt morgen Feedback
- HRU: das ist gut - in der Runde Team/HSR diskutieren was gemacht werden kann
- KGA: Was wird mit UX passieren? (Expert Talks)
    - DHI: Prio 1 hat interne Aufträge und Prio 2 externe

Diskussion im Team

- KGA: also ist keines der beiden Thema sehr interessant für Team?
- Team: Ja, insbesondere auch zu gross für die kurze Zeit (Thema 2)
- HRU: Reservationssystem vorallem Businessanalyse
- HRU: Sicher gut zu schauen was DHI einbringt
- HRU: Aber auch schauen was wir für Ideen haben
- KGA: was wäre ursprüngliche Idee für SA gewesen
- MWE: XMPP Server in node.js modular, flexibel bzgl. Datenbankanbindung
- MAL: und auch Skalierbarkeit von node.js interessant
- MAL: interessant wäre vielleicht frontend framework
- HRU: Alle miteinander auf dem Laufenden halten bzgl. Ideen

Nächstes Meeting

- Mittwoch, 20. Februar 2013, 10:10 Uhr