HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Namics.

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Namics.

Schulterklopfen mal virtuell

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrsemester 2015

Autor: Luca Tännler

Betreuer: Hans Rudin

Projektpartner: Beat Helfenberger, Namics AG, St. Gallen

Contents

Abstract

Ein Unternehmen – egal in welcher Branche – funktioniert besser mit motivierten Mitarbeitern. Motivation kommt nicht aus dem Nichts, also muss sich das Unternehmen damit befassen.

Die Namics will ein Pilotprojekt starten, welches zur Mitarbeitermotivation beitragen soll.
Die Idee ist es, dass sich Mitarbeiter gegenseitig motivieren können indem sie Ihren Arbeitskollegen virtuell auf die Schultern klopfen können.

In dieser Studienarbeit wurde eine Web-APP entwickelt auf welcher sich die Mitarbeiter mit ihrem Google Account einloggen können und Ihren Arbeitskollegen auf die Schultern klopfen können und auch Schulterklopfer anderer nachverfolgen können. Zusätzlich haben die Mitarbeiter die Möglichkeit, einen kurzen Text zu Ihrem Schulterklopfer zu verfassen in welchem sie ihr Lob aussprechen können.

Die Benutzung dieser Web Applikation sollte möglichst einfach gestaltet sein und einfach erreichbar – sprich über Smartphones, Tablets und PCs. Um dies zu erreichen wurde Bootstrap eingesetzt welches ermöglicht, einfach verschiedene Darstellungsoptionen einzustellen je nach Grösse des Clientgerätes.

Desweitern gibt es auch ein Email Notifikationssystem, damit die Mitarbeiter auf dem Laufenden gehalten werden von den Aktivitäten auf der Web-APP, ohne dass sie andauernd nach Neuigkeiten prüfen müssen.

Zur Implementation wurde das Spring Framework mit folgenden wesentlichen Modulen eingesetzt:

- Spring Boot
- Spring Data JPA
- Spring MVC (Im Core enthalten)
- Spring Security (+ OAuth2)

Einerseits weil es in der Namics häufig verwendet wird und weil es viele Funktionalitäten besitzt um einfach eine Web Applikation mit Java zu implementieren. All diese Module bieten viele vorimplementierte Funktionalitäten, welche den Aufwand der Umsetzung der Webapplikation verringerten.

Neben der Einfachheitshaltung der Erreichbarkeit und Nutzung der Web-APP, lässt sich die Web Applikation auch einfach installieren und online schalten. Dazu wird das Projekt mit MAVEN verwaltet, was unter anderem einen integrierten Tomcat Server beinhaltet – so dass die Applikation einfach gestartet werden kann. Des Weiteren wurde darauf geachtet, dass möglichst wenig konfiguriert werden muss um die Applikation starten zu können (z.B. Email Server, Datenbank Anbindung).

Erklärung der Eigenständigkeit

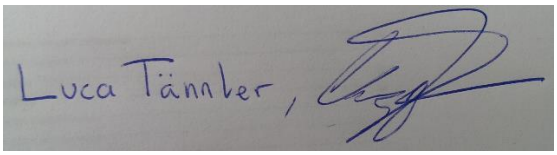
Ich erkläre hiermit,

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

Ort, Datum:

29.05.2015

Name, Unterschrift:



Aufgabenstellung

Auftraggeber und Betreuer

Diese Studienarbeit wird als „Industriearbeit“ mit der Firma *Namics AG, Teufenerstrasse 19, 9000 St. Gallen* als externem Auftraggeber durchgeführt.

- Ansprechpartner Auftraggeber: Beat Helfenberger, beat.helfenberger@namics.com
- Betreuer HSR: Prof. Hans Rudin, hrudin@hsr.ch

Studierende

Diese Arbeit wird als Studienarbeit an der Abteilung Informatik durchgeführt von

- Luca Tännler, ltaennle@hsr.ch

Einführung

Unternehmen haben das Ziel motivierte Mitarbeiter zu beschäftigen. Sie müssen sich daher mit der Motivation ihrer Mitarbeiter auseinandersetzen. Motivatoren beeinflussen die Leistung des Mitarbeiters. Herzberg beschreibt den Faktor Anerkennung als den prozentual am zweit häufigsten Faktor, welcher zu extremer Zufriedenheit führt.

Namics möchte nun in einem Pilotversuch, anhand einer mobilen Webapplikation, eine neuartige Möglichkeit für die Anerkennung bieten. Dabei geht es darum, dass Mitarbeitende eine Art Schulterklopfen virtuell an einen Kollegen durchführen können. Beispielsweise wollen Mitarbeitende auf dem nach Hause Weg im Zug oder einem Kollegen in einer anderen Niederlassung für seine Unterstützung danke sagen.

Aufgabenstellung

Es soll eine Web-Applikation für „Virtuelles Schulterklopfen“ entwickelt werden.

Namics erwartet am Ende der Arbeit eine funktionsfähige Webapplikation. Die Entwicklung findet und deren Umfang definiert sich in einem agilen Softwareentwicklungsvorgehen.

Damit Namics, nach Abschluss der Abschlussarbeit, die Applikation weiterentwickeln kann, gelten folgende Rahmenbedingungen:
- Die Webapplikation soll mit dem Bootstrap Framework Responsive umgesetzt werden. - Zu unterstützende Browser sind die Standard-Browser der Betriebssysteme iOS und Android. - Die Backendentwicklung basiert auf Basis Java mit einem Tomcat Application Server. – Das Betriebssystem für das Backend ist Linux. – Eine benötigte Datenhaltung soll in MySQL umgesetzt werden. - Das Java Spring-Framework, Apache CXF sollen in erster Linie bei der Wahl von Frameworks eingesetzt werden.

Ein git-Sourcenverwaltung wird von Namics zur Verfügung gestellt – Zielsystem für das Backend wird von Namics installiert und ist unter schulterklopfen.namics.com erreichbar.

Zur Durchführung

Mit dem Betreuer finden wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch den Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber sind vom Studierenden nach Bedarf zu initialisieren.

Alle Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten, die Besprechung ist durch den Studierenden zu leiten und die Ergebnisse sind in einem Protokoll festzuhalten, das dem Betreuer 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 Resultate.

Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente bzw. Berichtsteile 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 nur auf Wunsch in gedruckter Form abzugeben.

Termine

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

Montag, den 23.2.2015	**Beginn der Studienarbeit Ausgabe der Aufgabenstellung durch den Betreuer**
Freitag, 27.2.2015	Kick-off Besprechung mit Auftraggeber, Namics St. Gallen
26.05.2015	Kurzfassung und A0-Poster per E-Mail zur Überprüfung an Betreuer (Vorlage stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.)
29.05.2015	Der Studierende sendet die vom Betreuer abgenommene und freigegebene Kurzfassung als Word-Dokument an das Abteilungssekretariat cfurrer@hsr.ch
5.6.2015, 17:00	Abgabe der Arbeit an den Betreuer.

Beurteilung

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von 30 Stunden budgetiert (Siehe auch Modulbeschreibung der Studienarbeit.)

Für die Beurteilung ist der HSR-Betreuer verantwortlich.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/5
3. Inhalt*)	3/5

*) 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 Studienarbeiten.

Rapperswil, den 22. Februar 2015

Prof. Hans Rudin
Institut für Software
Hochschule für Technik Rapperswil

Management Summary

Ausgangslage

Mitarbeitermotivation ist ein Wichtiger Aspekt in einer Firma. Es gibt verschiedene Methoden die Mitarbeitermotivation zu steigern. Namics möchte deshalb mit einem Pilotprojekt es den Mitarbeitern ermöglichen, sich gegenseitig virtuell auf die Schulter zu klopfen und sich somit gegenseitig zu motivieren.

Damit solch ein Tool auch verwendet wird, muss es einfach erreichbar und bedienbar sein. Um dies zu erreichen, soll die Web App von alle gängigen Endgeräten wie Android- iOS-Smartphones wie auch dem PC abrufbar sein.

Damit sich die Mitarbeiter der Namics AG nicht ein weiteres Login merken müssen, sollen sich die Mitarbeiter mit Ihrem Google Account einloggen können, welchen Sie auch für andere Namics interne Dienste verwenden.

Das Projekt soll mit einer agilen Softwareentwicklungsmethode erarbeitet werden, so dass der Arbeitgeber nach jedem Sprint die Arbeitspakete priorisieren kann und definieren kann, welche Teilbereiche der Applikation als nächstes umgesetzt werden sollen.

Vorgehen, Technologien

Für das Projekt wurden folgende Voraussetzungen für Technologien gesetzt:

- Java **Spring** Framework
- **MAVEN** – Projektverwaltung
- Frontend Design mit **Bootstrap** und **JQuery**
- Integrierter Tomcat Server (In MAVEN)
- Google Login mit OAuth2

Das Projekt wurde aus einer Mischung zwischen der RUP- und der SCRUM-Methode umgesetzt. In einem ersten Schritt gab es eine Inception & Elaboration Phase aus RUP in welcher unter anderem Analysen, ein Architekturprototyp und Anforderungen definiert wurden. Nach dieser Phase wurde die WebAPP anhand von SPRINTS nach der SCRUM Methodik entwickelt.

Als erster Schritt galt es, die Anforderungen zusammen mit dem Arbeitgeber zu definieren und den Ablauf des Projekts sowie Meilensteine zu planen.

Darauffolgend musste ich mich sowohl mit dem Java Spring Framework, MAVEN, OAuth2 und Bootstrap vertraut machen und das Grundkonzept dieser Technologien verstehen.

Nachdem das Projekt und die Web Applikation geplant wurden und ich mir Grundkenntnisse der verwendeten Technologien aneignete, kam es zur Planung des ersten Sprints.

Es folgten weitere Sprints, welche jeweils zusammen mit dem Arbeitgeber und Betreuer geplant wurden.

Zum Schluss ging es darum, jegliche Dokumentationen fertigzustellen und für die Abgabe vorzubereiten.

Ergebnisse

Die Ergebnisse können grob in 2 Teile aufgeteilt werden:

- 2 Analysen:
  - Analyse Spring OAuth2 für Google Login
  - Analyse Bootstrap vs. Polymer
- Die Web Applikation

Folgend ein Screenshot eines Teils der fertigen Web Applikation:

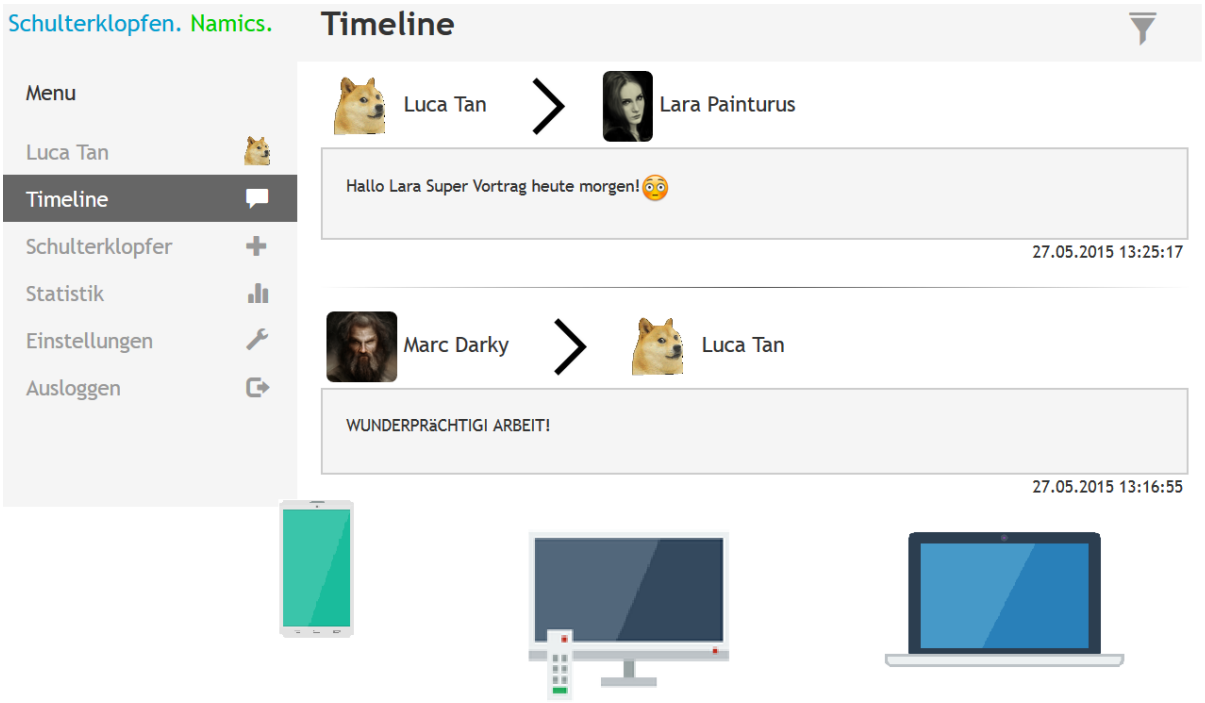


Abbildung 1 WebAPP Timeline

Ausblick

Die Applikation ist in einem Zustand welcher Reif für eine Veröffentlichung ist.

Je nach internem Erfolg, kann die Web Applikation noch erweitert werden und evtl. andern Firmen zur Verfügung gestellt werden und als Produkt verkauft werden.

Für die Weiterentwicklung und die Nutzung dieser Web Applikation ist die Namics AG zuständig, welche auch entscheidet wie diese Applikation weitergeführt wird.

Anforderungsspezifikation

Einleitung

Allgemeine Projektbeschreibung

Die Schulterklopfen WebAPP soll es Mitarbeitern möglich machen, anderen Mitarbeitern einen Schulterklopfer zu geben. Dies ist ein Pilotprojekt und dient zur Motivation von Mitarbeitern.

Die WebAPP soll von unterwegs erreichbar sein und auf den gängigen Browsern von iOS und Android laufen.

Use Case Diagramm

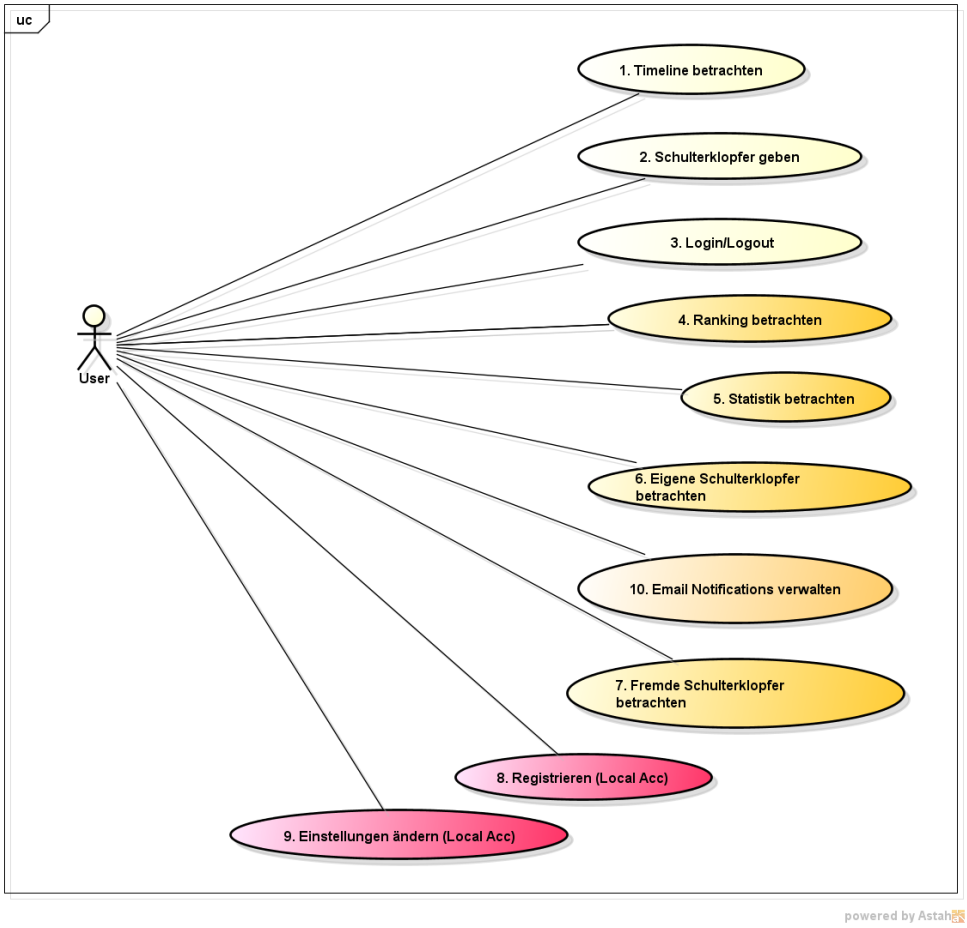


Abbildung 2: Use Case Diagramm V3

INFO: Diejenigen Use Cases mit **orangem Hintergrund** haben eine mittlere Priorität. Use Cases mit **rotem Hintergrund** haben eine tiefe Priorität. Nachdem alle UC's mit hoher Priorität umgesetzt wurden, wird entschieden, ob und welche UC's mit mittlerer und tiefer Priorität noch umgesetzt werden.

Die einzelnen Use Cases werden im Kapitel „**Funktionale Anforderungen**“ beschrieben.

Akteure

In diesem UseCase Diagramm gibt es einen Akteur: Der **User**.

Er ist der Benutzer der WebApp.

Domain Model

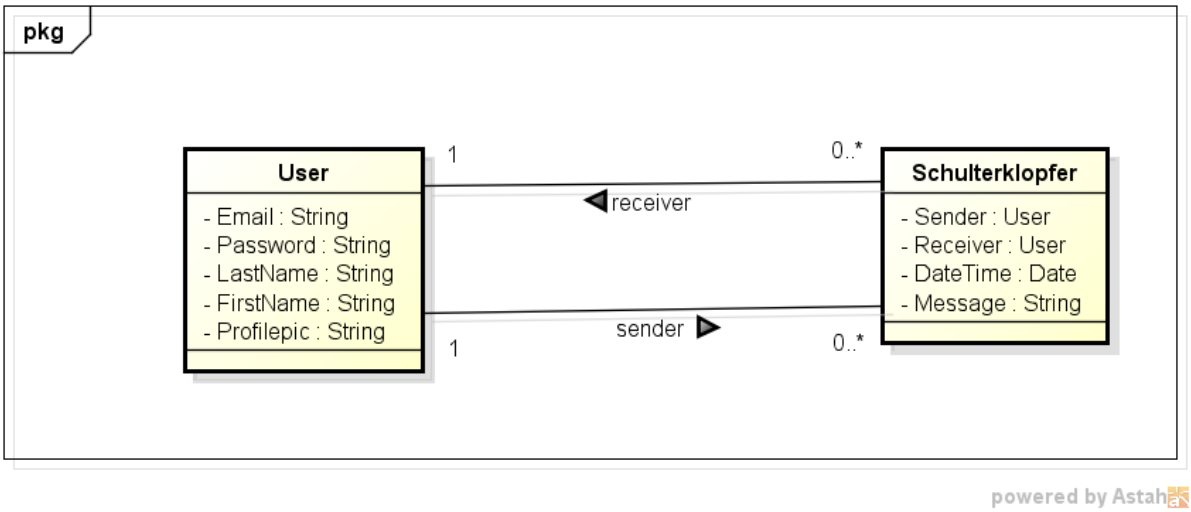


Abbildung 3 Domainmodel

Funktionale Anforderungen (Use Cases)

1. Timeline betrachten

Main Actor: User
Priorität: MUST

Der User möchte alle Schulterklopfer von allen Personen in einer nach Datum sortierten Liste ansehen. Nachdem er sich eingeloggt hat, wird ihm diese Liste angezeigt.

2. Schulterklopfer geben

Main Actor: User
Priorität: MUST

Ein Mitarbeiter hat heute eine super Leistung hingebracht und sich für sein Team eingesetzt.

Der User möchte jetzt dieser Person einen virtuellen Schulterklopfer geben.

Also klickt er auf den Button „Schulterklopfer geben“ und gelangt auf eine Seite auf welcher er nach einer Person suchen kann. Nachdem er die Person gefunden hat, klickt er auf dessen Eintrag und gelingt auf eine Seite auf welcher er einen Text für den Schulterklopfer eingeben kann. Anschliessend klickt er auf absenden und gelangt wieder auf die Timeline.

3. Login/Logout

Main Actor: User
Priorität: MUST

Der User möchte sich in die WebApp einloggen. Er gibt seine Google Login Daten ein und wird automatisch eingeloggt. Falls der User sich das erste Mal einloggt, wird ihm automatisch ein Account erstellt.

Nach dem er fertig ist mit der Benutzung der APP möchte sich der User wieder ausloggen. Dies macht er indem er in der Navigation auf den Logout-Link klickt. Nachdem Logout kommt der User wieder auf die Login Seite.

4. Ranking betrachten

Main Actor: User
Priorität: MEDIUM

Der User möchte die TOP Schulterklopfer-Erhalter (Anzahl konfigurierbar) des aktuellen Monats sehen. Auf der Ranking Seite werden ihm die konfigurierte Anzahl an Usern angezeigt mit den meist erhaltenen Schulterklopfern dieses Monats.

5. Statistik betrachten

Main Actor: User
Priorität: MEDIUM

Der User möchte seine persönliche Statistik betrachten welche ihm z.B. darüber Auskunft gibt, wie viel Schulterklopfer er insgesamt, diesen Monat oder diese Woche erhalten hat.

6. Eigene Schulterklopfer betrachten

Main Actor: User
Priorität: MEDIUM

Der User möchte seine erhaltenen Schulterklopfer betrachten und Lesen. Über die Navigation gelingt er auf seine persönliche Schulterklopfer Seite auf welcher alle empfangenen Schulterklopfer nach Datum sortiert aufgelistet sind.

Jetzt möchte er nur Schulterklopfer von einer bestimmten Person betrachten. Über die Filteroption kann er den Namen der Person angeben. Somit werden nur Schulterklopfer dieser Person aufgelistet.

7. Fremde Schulterklopfer betrachten

Main Actor: User
Priorität: MEDIUM

Der User möchte die Schulterklopfer eines Kollegen betrachten.

Er kann in einem Suchfeld den Namen des Kollegen eingeben worauf dann alle passenden Personen aufgelistet werden. Er klickt auf den Eintrag seines Kollegen und wird auf die Schulterklopfer Seite der Person weitergeleitet auf der alle Schulterklopfer dieser Person aufgelistet werden, nach Datum sortiert.

Die Einträge kann er auch nach Namen und Datum filtern.

8. Registrieren

Main Actor: User
Priorität: LOW

Der User möchte die WebApp verwenden und möchte sich deshalb bei der WebApp registrieren.

Falls er im Besitz eines Google Accounts ist, kann er sich direkt mit seinem Google Account Anmelden. Auf diese Art und Weise wird ihm automatisch ein Account erstellt.

Andernfalls kann er sich einen internen Account in der WebApp erstellen. Der User gibt seine Email-Adresse, Passwort und seinen Vor- und Nachnamen an.

9. Einstellungen ändern

Main Actor: User
Priorität: LOW

Der User möchte die Einstellungen seines Accounts ändern. Über die Navigation erreicht der User die Einstellungsseite und kann die gewünschten Änderungen vornehmen wie z.B. einen Passwortwechsel.

10. Email notifications verwalten

Main Actor: User
Priorität: MEDIUM

Der User möchte eine E-Mail erhalten, wenn er selbst und Person X und Y einen Schulterklopfer erhalten.

Über die Navigation gelingt der User zur Einstellungsseite für Email Notifications. Da definiert er, wann er Email Notifications erhalten möchte.

Nicht-Funktionale Anforderungen

1. Zuverlässigkeit

1.1 Datenverlust

Wenn die DB beschädigt oder fehlerhaft ist und Daten verloren gehen, kann man einfach ein DB-Backup einspielen ohne weitere notwendige Schritte.

2. Benutzbarkeit

2.1 Verständlichkeit

Das Programm soll ohne Vorkenntnisse benutzbar sein.

2.2 Selbsterklärendes UI

Das Userinterface ist selbsterklärend und übersichtlich gestaltet.

2.3 Mobile Browsersupport

Das UI wird in den gängigen Browsern von iOS und Android übersichtlich und korrekt dargestellt.

3. Leistung und Effizienz

Die WebAPP hat keine speziellen Anforderungen an die Leistung und Effizienz.

4. Portierbarkeit und Übertragbarkeit

4.1 Installation

Die WebAPP Server Software soll einfach zu installieren sein mit einem möglichst geringen Aufwand.

4.2 Konfiguration

Die Konfiguration soll einfach geändert werden können. (Zum Beispiel die Datenbank-Verbindung)

5. Korrektheit

An die Korrektheit des Programms werden keine speziellen Anforderungen gestellt.

Dies auch mit dem Grund, da keine heikle Daten und Ergebnisse erwartet werden.

6. Skalierbarkeit

An die Skalierbarkeit werden keine speziellen Anforderungen gestellt.

7. Wartbarkeit

7.1 Datenbank

Die Datenbank soll nicht gewartet werden müssen und selbständig laufen über einen langen Zeitraum.

8. Änderbarkeit / Erweiterbarkeit

8.1 Frameworks

Die Software soll weitestgehend mit dem Java Spring Framework umgesetzt werden um Weiterentwicklungen zu vereinfachen.

8.2 UI

Das UI soll einfach erweitert und geändert werden können.

8.3 Server-Side WebApp

Die Serverseitige Software der WebAPP soll einfach erweitert und geändert werden können ohne grossen Aufwand.

➔ Es können einfach neue Funktionalitäten hinzugefügt werden können.

Spring OAuth2 Analyse

Abstract

Das Spring Framework enthält diverse Integrationen von Sicherheits-Mechanismen/-Tools. Diese sind im Spring Security Plug-In enthalten (je nach dem in Erweiterungen).

Eines dieser Sicherheits-Mechanismen ist **OAuth**. Es ist ein offenes Protokoll mit einer standardisierten, sicheren API-Autorisierung. Mithilfe des OAuth-Protokolls kann ein User einer Anwendung den Zugriff auf seine Daten erlauben welche von einer anderen Anwendung verwaltet werden ohne dabei Details zu seiner Zugangsberechtigung (PW, Username) preiszugeben.

Spring besitzt bereits eine Integration von OAuth – Spring OAuth, eine Erweiterung des Spring Security Plug-Ins.

Es stellt sich folgende Frage: Gibt es bereits eine Google Login Integration in Spring OAuth? Wie gross ist der Aufwand? Gibt es andere Optionen?

Einleitung

Diese Analyse dient dazu, Auskunft über eine Integration von Google Login mit Spring OAuth zu geben:

- Gibt es bereits eine fertige Integration?
- Wie gross ist der Aufwand?
- Was gibt es für Optionen?

OAuth 2.0

In diesem Kapitel wird ausschliesslich OAuth 2.0 behandelt und nicht die Vorgängerversion.

OAuth 2.0 ist ein offenes Protokoll mit einer standardisierten, sicheren API-Autorisierung. Mithilfe des OAuth-Protokolls kann ein User einer Anwendung den Zugriff auf seine Daten erlauben welche von einer anderen Anwendung verwaltet werden ohne dabei Details zu seiner Zugangsberechtigung (PW, Username) preiszugeben.

OAuth 2.0 fokussiert sich auf die Vereinfachung von Client-Entwicklungen. Es liefert spezifische Authorization flows für Web-, Desktop- und Mobileapplikationen. OAuth 2.0 wird von vielen grossen Serviceleistern angeboten wie z.B. Facebook, Google, Microsoft und vielen mehr.

Mithilfe von OAuth 2.0 kann z.B. ein Google-Benutzer einer Applikation Zugriffsrechte auf bestimmte Daten geben die er bei Google hat (Persönliche Infos, Daten, …) ohne dabei seine Google Account Autorisierungsdaten bekannt geben zu müssen. Dies birgt viele Vorteile wie z.B. kann er sich mit seinen Google Login Daten bei anderen Applikationen Einloggen und muss sich nicht mehrere Benutzernamen und Passwörter merken.

Spring OAuth

Spring OAuth ist eine Erweiterung von Spring Security und liefert vorimplementierte Lösungen für die Verwendung von OAuth.

Features

- Unterstützung für OAuth Anbieter und Benutzer
- OAuth 1a
- OAuth 2

Google Login

Spring OAuth besitzt keine vorimplementierte Integration für ein Login (z.B. via Google oder Facebook). Eine eigene Implementation kann daher sehr aufwändig sein.

Für ein Login-Mechanismus zusammen mit Spring Security über z.B. Google Accounts müsste man unter anderem einen eigenen **AuthenticationProvider** und **UserDetailService** implementieren. Dies ist nötig, da die Standard Implementation von Spring einen Username & Passwort voraussetzt für eine Authentifizieren – für das Google Login via OAuth brauchen wir jedoch andere Authentifizierungs-Daten wie z.B. die Google-Account ID und einen validen Token für den Datenzugriff.

Es gibt ein Spring Plug-In welches von der Community entwickelt wurde welches eine einfache Integration von Social Logins wie z.B. von **Google+** und vielen mehr ermöglicht. Das Plug-In wird im Kapitel „Spring Social“ genauer eingeführt.

Möchte man jedoch einen Zugriff auf allgemeine Google Accounts müsste man bei Spring Social zusätzliche Features Implementieren. Es gibt noch ein weiteres Plug-In für Spring Security, welches von der Community entwickelt wurde. Auf dies wird im Kapitel „Spring Security OAuth2 Google“ eingegangen.

Spring Social

Spring Social ist ein Plug-In für das Spring Framework. Es ermöglicht es einem seine Applikation mit einem Software-as-a-Service (Saas) API Provider zu verbinden.

http://projects.spring.io/spring-social/

Spring Social selber wird auch weiter unterteilt in **Spring Social Core** und die verschiedenen Saas-Provider Libraries wie z.B. **Spring Social Google** oder **Spring Social Facebook**.

In diesem Kapitel wird ausschliesslich auf das **Spring Social Google** eingegangen.

Spring Social Google

Spring Social Google ist eine Erweiterung des Spring Social Plug-Ins und wurde von der Community entwickelt.

https://github.com/GabiAxel/spring-social-google

Eine Übersicht und Einführung in diese Erweiterung ist unter folgendem Link zu finden:

http://gabiaxel.github.io/spring-social-google-reference/overview.html

Spring Social Google ermöglicht einem die Integration von Google Services.
Die Integration findet statt mit der GoogleConnectionFactory – eine Connection Factory welche in das Spring Social Plug-In eingebunden werden kann und eine API mit Bindung an die Google REST APIs beinhaltet.

Dieses Plug-In ist jedoch begrenz und es setzt z.B. einen Google+-Account voraus.

Integration

Um diese Erweiterung zu verwenden, muss folgende Maven Dependency eingetragen werden:

```
<dependency>
  <groupId>org.springframework.social</groupId>
  <artifactId>spring-social-google</artifactId>
  <version>${org.springframework.social-google-version}</version>
</dependency>
```

Quelle: http://gabiaxel.github.io/spring-social-google-reference/overview.html

Konfiguration

Das Spring Social Plug-In kann mehrere Social Anbier verwalten. Damit also der **ConnectController** von Spring Social die **GoogleConnectionFactory** finden kann, muss diese registriert werden in der Spring Social Konfiguration:

```
 @Configuration
public class SocialConfig {

    @Inject
    private Environment environment;

    @Bean
    public ConnectionFactoryLocator connectionFactoryLocator() {
        ConnectionFactoryRegistry registry = new
ConnectionFactoryRegistry();
        registry.addConnectionFactory(new GoogleConnectionFactory(
            environment.getProperty("google.consumerKey"),
            environment.getProperty("google.consumerSecret")));
        return registry;
    }

}
```

Quelle: http://gabiaxel.github.io/spring-social-google-reference/connecting.html

Die **ConnectionFactoryRegistry** und **GoogleConnectionFactory** kann auch wie folgt in **XML** definiert werden:

```
<bean id="connectionFactoryLocator"
class="org.springframework.social.connect.support.ConnectionFactoryRegistry
">
    <property name="connectionFactories">
        <list>
            <bean
class="org.springframework.social.google.connect.GoogleConnectionFactory">
                <constructor-arg value="${google.consumerKey}" />
                <constructor-arg value="${google.consumerSecret}"
/>
            </bean>
        </list>
    </property>
</bean>
```

http://gabiaxel.github.io/spring-social-google-reference/connecting.html

Weitere Informationen zur Konfiguration findet man unter folgender URL (Dokumentation):

http://docs.spring.io/spring-social/docs/1.0.x/reference/html/connecting.html

Google API Binding

Das *Google* Interface und ihre Implementation *GoogleTemplate* liefern alle Operationen die gebraucht werden um mit Google zu interagieren.

Instanz von *GoogleTemplate* generieren:

```
String accessToken = "f8FX29g..."; // access token received from Google
after OAuth2 authorization
Google google = new GoogleTemplate(accessToken);
```

INFO: Das *GoogleTemplate* besitzt auch einen Default Konstruktor ohne Parameter um eine Instanz ohne OAuth2-Infos zu erstellen. Dies erlaubt einem eine beschränkte Anzahl an Google+-Operationen welche keine Autorisation benötigen zu benutzen.

Instanz von *Google* erhalten (via Spring Social):

```
Connection<Google> connection =
connectionRepository.findPrimaryConnection(Google.class);
Google google = connection != null ? connection.getApi() : new
GoogleTemplate();
```

Spring Social Google Sub-APIS

Das Spring Social Google API ist unterteilt in 4 weitere Sub-APIs:

- PlusOperations: Lesen von Google+ Aktivitäten & Profilen, App Aktivitäten erstellen
- TaskOperations: Lesen & verwalten von Tasks des Benutzers
- DriveOperations: Lesen & verwalten von Daten des Benutzers in Google Drive
- getAccessToken()-Methode

```
Das Google Interface
public interface Google {

    PlusOperations plusOperations();

    TaskOperations taskOperations();

    DriveOperations driveOperations();

    String getAccessToken();

}
```

Genauere Informationen zu den Sub-APIs gibt es unter folgender URL:

http://gabiaxel.github.io/spring-social-google-reference/apis.html

Spring Security OAuth2 Google

Spring Security OAuth2 Google ist kein offizielles Spring Plug-In. Es wurde von einem Entwickler implementiert, welcher selbst Spring verwendet für Applikationsentwicklungen.

Die offizielle Seite dieses Plug-Ins ist auf GitHUB unter folgender URL zu finden:

https://github.com/skate056/spring-security-oauth2-google

Die Verwendung ist ziemlich einfach und kann in wenigen Schritten in das Projekt integriert werden:

- Ein paar Klassen in das Projekt kopieren

- Einige Konfigurationen vornehmen (Hauptsächlich copy&paste)
- Google API Zugangsdaten festlegen

Integration

Folgend wird beschrieben, wie Spring Security OAuth2 Google in das Projekt integriert werden kann.

Benötigte Klassen / Konfigurationen

- AuthorityGranter.java
- DefaultUserAuthenticationConverter.java
- GoogleAccessTokenConverter.java
- GoogleTokenServices
- **OAuth2SecurityConfiguration.java**
- **Spring Security Configuration** (in diesem Beispiel in XML umgesetzt)

Google API Zugangsdaten

Um die Google OAuth2 API benutzen zu können, muss man API Zugangsdaten einholen. Dies kann man auf folgender Webseite:

https://console.developers.google.com

Man muss ein neues Projekt erstellen und API Zugangsdaten für die Google OAuth API anfordern.

Wenn dies getan ist erhält man folgende Daten:

- Client-ID
- E-Mail-Adresse
- Clientschlüssel
- Weiterleitungs-URIs (Muss man selber definieren)
- JavaScript-Quellen (Muss man selber definieren, optional)

Konfiguration: Google API Benutzung

Im **application.properties** File (Globales Konfigurationsfile in Spring) müssen folgende Parameter definiert werden:

`google.client.id`	Client-ID für den API-Zugriff
`google.client.secret`	Secret (Clientschlüssel) für den API-Zugriff
`google.auth.scope`	Hier definiert man, auf welche Daten zugegriffen werden will (es können mehrere definiert werden – Kommagetrennt). **Beispiel:** `https://www.googleapis.com/auth/userinfo.email`, `https://www.googleapis.com/auth/userinfo.profile`
`google.accessTokenUri`	URI für Access-Token: `https://accounts.google.com/o/oauth2/token`
`google.userAuthorizationUri`	URI für die Benutzer Authorization: `https://accounts.google.com/o/oauth2/auth`
`google.authorization.code`	`authorization_code`
`google.preestablished.redirect.url`	Redirection URL: Die URL von welcher zu Google weitergeleitet wird. Z.B.: `http://yoursite.com/login/googleLogin`

Spring Security Beispielskonfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<b:beans xmlns:sec="http://www.springframework.org/schema/security"
         xmlns:b="http://www.springframework.org/schema/beans"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">
    <sec:http auto-config="true" use-expressions="true" entry-point-
ref="clientAuthenticationEntryPoint">
        <sec:http-basic/>
        <sec:logout logout-success-url="/" logout-url="/logout" delete-cookies="JSESSIONID"/>
        <sec:anonymous enabled="false"/>
        <!-- This is the crucial part and the wiring is very important -->
        <!--
            The order in which these filters execute are very important. oauth2ClientContextFilter must
be invoked before
            oAuth2AuthenticationProcessingFilter, that's because when a redirect to Google is required,
oAuth2AuthenticationProcessingFilter
            throws a UserRedirectException which the oauth2ClientContextFilter handles and generates a
redirect request to Google.
            Subsequently the response from Google is handled by the
oAuth2AuthenticationProcessingFilter to populate the
            Authentication object and stored in the SecurityContext
        -->
        <sec:custom-filter ref="oauth2ClientContextFilter" after="EXCEPTION_TRANSLATION_FILTER"/>
        <sec:custom-filter ref="oAuth2AuthenticationProcessingFilter"
before="FILTER_SECURITY_INTERCEPTOR"/>
    </sec:http>

    <b:bean id="oAuth2AuthenticationProcessingFilter"
class="org.springframework.security.oauth2.client.filter.OAuth2ClientAuthenticationProcessingFilter">
        <b:constructor-arg name="defaultFilterProcessesUrl" value="/login/googleLogin"/>
        <b:property name="restTemplate" ref="googleRestTemplate"/>
        <b:property name="tokenServices" ref="tokenServices"/>
    </b:bean>

    <!--
        These token classes are mostly a clone of the Spring classes but have the structure modified so
that the response
        from Google can be handled.
    -->
    <b:bean id="tokenServices" class="com.namics.schulterklopfer.googleOAuth2.GoogleTokenServices">
        <b:property name="checkTokenEndpointUrl"
value="https://www.googleapis.com/oauth2/v1/tokeninfo"/>
        <b:property name="clientId" value="${google.client.id}"/>
        <b:property name="clientSecret" value="${google.client.secret}"/>
        <b:property name="accessTokenConverter">
            <b:bean class="com.namics.schulterklopfer.googleOAuth2.GoogleAccessTokenConverter">
                <b:property name="userTokenConverter">
                    <b:bean
class="com.namics.schulterklopfer.googleOAuth2.DefaultUserAuthenticationConverter"/>
                </b:property>
            </b:bean>
        </b:property>
    </b:bean>

<!--
This authentication entry point is used for all the unauthenticated or unauthorised sessions to be
directed to the /googleLogin URL which is then intercepted by the oAuth2AuthenticationProcessingFilter
to trigger authentication from Google.
-->
    <b:bean id="clientAuthenticationEntryPoint"
class="org.springframework.security.web.authentication.LoginUrlAuthenticationEntryPoint">
        <b:property name="loginFormUrl" value="/login/googleLogin"/>
```

```
    </b:bean>

    <sec:authentication-manager alias="alternateAuthenticationManager">
        <sec:authentication-provider>
            <sec:user-service>
                <sec:user name="user" password="password" authorities="DOMAIN_USER"/>
            </sec:user-service>
        </sec:authentication-provider>
    </sec:authentication-manager>
</b:beans>
```

Die im obigen Config Beispiel rot- & fett-markierten stellen müssen angepasst werden. Dies ist die URL auf welcher der User dann zu Google für das Login weitergeleitet wird.

Fazit

Da für die Schulterklopfen WebAPP ein Login mit Google Accounts gebraucht wird (Nicht Google+), eignet sich das Spring Social Plug-In nicht. Also muss entweder etwas Eigenes Implementiert werden oder das vorgestellte **Spring Security OAuth2 Google** Plug-In verwendet wird.

Die Verwendung von **Spring Security OAuth2 Google** ist wohl die beste Variante, da nur wenige Konfigurationen vorgenommen werden müssen damit sich User mit ihrem Google Account einloggen können

Bootstrap vs. Polymer Analyse

Abstract

In der Entwicklung von Web-Applikationen und Websites werden immer häufiger Frameworks eingesetzt um Zeit zu sparen und Boiler Plate Code zu vermeiden. Des Weiteren ist es ein immer grösser werdender Trend responsive UIs zu gestalten – UIs die sich je nach Bildschirmgrösse/Gerät anpassen um das User Experience zu erhöhen.

In dieser Analyse werden die zwei Frameworks Bootstrap (von Twitter) und Polymer (von Google) vorgestellt und verglichen.

Bootstrap enthält verschiedene vordefinierte Web-Komponenten für Webapplikationen wie z.B. Formulare, Buttons, Navigation und vieles mehr. Des Weiteren unterstützt das Framework auch *responsive web design* um die Webapplikation je nach Gerät/Bildschirmgrösse optimal darzustellen.

Polymer beinhaltet wie Bootstrap auch eigene vordefinierte Web-Komponenten. Jedoch ist Polymer nicht ganz dasselbe. Polymer zielt mehr darauf hin, dass Entwickler ihre Webapplikationen gänzlich auf Web Component Technologien aufbauen. Des Weiteren bietet Polymer diverse APIs um den aktuellsten Web-standards gerecht zu werden. Dieses Framework ermöglicht es einem auch, eigene HTML Tags zu erstellen. Polymer beinhaltet auch vorgefertigte Funktionen um responsive UI's zu gestallten.

Einleitung

Diese Analyse dient dazu, die beiden Java Script Frameworks Bootstrap und Polymer vorzustellen und Polymer zu vergleichen. Der Leser soll zum Schluss entscheiden können, welches der beiden Java Script Frameworks am besten für sein Web-Projekt geeignet ist.

Bootstrap

Bootstrap ist ein gratis und open source Java Script Framework welches von Twitter ins Leben gerufen wurde.

http://getbootstrap.com/

Features

Komponenten

Bootstrap hat eine sehr breite Palette an HTML- und CSS-basierten Templates für Schriften, Formulare, Buttons, Navigationen und viele weitere Komponenten.

Dank diesen vordefinierten Komponenten kann man sich viel Zeit und Aufwand einsparen. Das Aussehen der Komponenten kann einfach mit CSS abgeändert werden und für das eigene Design angepasst werden.

Desweitern verhindern diese vordefinierten Komponenten, dass man Boiler Plate Code schreibt da man die Komponenten beliebig oft verwenden kann ohne sie neu zu schreiben.

Java Script Erweiterungen

Bootstrap verwendet jQuery für ihre Animationen und für viele Funktionalitäten.
Deshalb gibt es auch viele JQuery Plug-Ins die zusammen mit Bootstrap verwendet werden können.

Einige Plug-Ins werden für bestimmte Bootstrap Komponenten vorausgesetzt wie etwa für Tooltips, Modals und Dropdowns.

Responsive Design

Ein sehr wichtiges Feature von Bootstrap welches seit Version 2.0 unterstützt wird ist das responsive design.

Responsive Design bewirkt, dass sich das Layout der Website an die Eigenschaften des Endgeräts bzw. der Screen Grösse anpasst. Somit kann man das User Experience massiv verbessern da z.B. auf einem Smartphone das Layout der Website benutzerfreundlicher dargestellt wird.

Browser Support

Bootstrap wird auf den meisten gängigen Desktop- und Mobile-Browsern unterstützt. In älteren Browsern mag die Seite eventuell ein wenig anders aussehen, aber ist trotzdem voll funktional.

Folgende Tabelle zeigt die Browser-Unterstützung auf. Bei den jeweiligen Browsern ist die Rede von den neusten Versionen.

	Chrome	Firefox	IE	Opera	Safari
Android	✓ Supported	✓ Supported	N/A	✗ No Support	N/A
iOS	✓ Supported	N/A	N/A	✗ No Support	✓ Supported
Max OS X	✓ Supported	✓ Supported	N/A	✓ Supported	✓ Supported
Windows	✓ Supported	✓ Supported	✓ Supported	✓ Supported	✗ No Support

Datenquelle: http://getbootstrap.com/getting-started/#support

Beispiel

Alle folgenden Beispiele wurden vollständig mit Bootstrap gemacht und zeigen verschiedene Möglichkeiten und Funktionen von Bootstrap.

Bei den Beispielen handelt es sich um Screenshots von Beispielen auf der Bootstrap Website:
http://getbootstrap.com/getting-started/#examples

Responsive Design

Bootstrap verändert standardmässig das Aussehen der Website je nach Grösse des Screens bzw. je nach Gerät. In Folgenden Screenshots sieht man einerseits die Desktop-Version (Links) und die Mobile-Version (Rechts). Im Grossen und Ganzen sehen die Seiten gleich aus, jedoch beispielweise die Navigation hat ihr Aussehen verändert:

Desktop Version **Mobile Version**

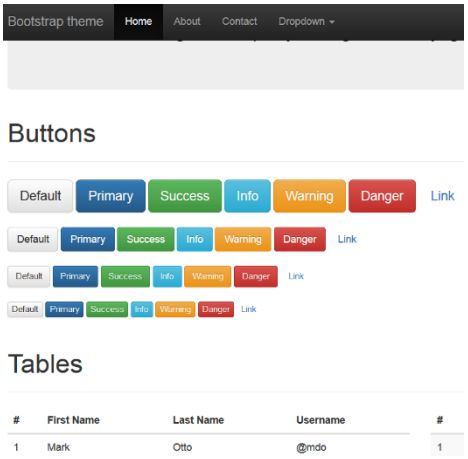


Abbildung 4 Responsive Design Vergleich 1

Abbildung 5 Responsive Design Vergleich 1

Dashboard

Bootstrap liefert templates für viele verschiedenen Dinge welche man oft in der Web Entwicklung braucht wie z.B. ein Dashboard. Der Code existiert bereits, es muss nur noch mit Daten gefüttert werden und je nach Bedarf kleinere Anpassungen & Einstellungen vorgenommen werden.

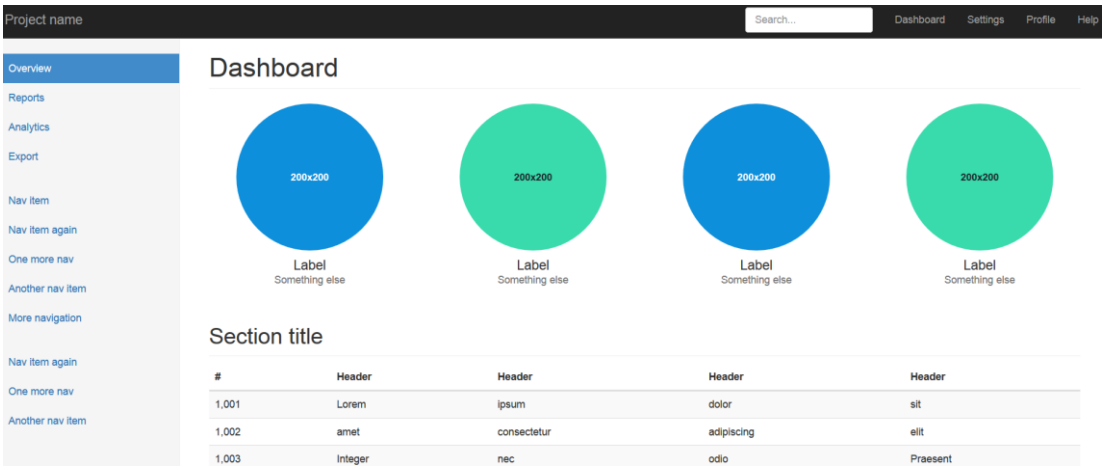


Abbildung 6 Bootstrap Dashboard

Carousel

Auch ein Template für ein „Carousel“. Dies wird sehr oft verwendet, speziell um die neusten News vorzustellen o.Ä.:

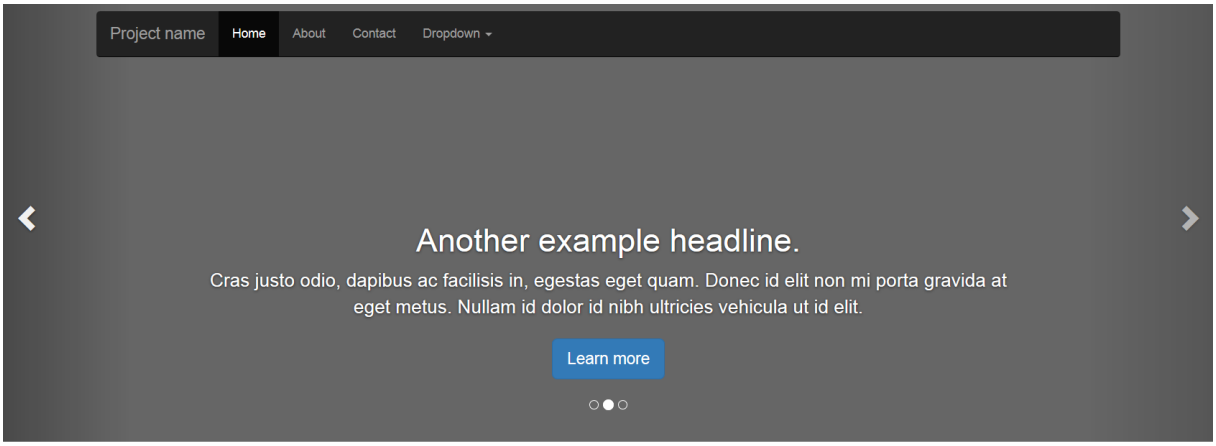


Abbildung 7 Bootstrap: Carousel

Weitere Beispiele

Unter folgendem Link gibt es eine sehr breite Palette an Beispielen der verschiedenen Komponenten, Responsive Designs, Templates etc.:

http://getbootstrap.com/getting-started/#examples

Tools

Bootlint

Bootlint ist ein offizielles Bootstrap HTML linter Tool. Es dient zur Überprüfung, ob das HTML korrekt ist.

Bootstrap Editors

Es gibt diverse Tools – sowohl als WebApp als auch Software – mit welchen man einfach Bootstrap Websites zusammen klicken kann.

Unter folgendem Link werden einige vorgestellt:

http://bootstrapbay.com/blog/bootstrap-editors/

SWOT Analyse

	Positiv (Hilfreich)	Negativ (Schädlich)
Stand: Heute	**Stärken** - Grosse Component Library - Unterstützung von Responsive Design - Mächtige Jquery Integration - Grosse Community - Gratis / Open Source	**Schwächen** - Komplettes Bootstrap ist über 150KB gross - Benötigt zwingend JS - Kann zu Problemen führen wenn gleichzeitig noch andere Frameworks eingesetzt werden
Stand: Zukunft (Vorhersage)	**Chancen**	**Gefahren**

	- Immer grösseres Angebot an Components - Evtl. kommen weitere mächtige Funktionalitäten hinzu	- Bootstrap „geht unter“ und wird von einem anderen Anbieter übertrumpft - Bootstraps Angebot wird von anderen Technologien eingeholt

Polymer

Polymer ist ein JS Framework/Library mit welchem man Web Komponenten erstellen kann – in anderen Worten: Man kann eigene HTML Elemente definieren.

Mit Hilfe von Polyfills & Sugar kann es diese Elemente erstellen und Web Component Support in Browser bringen welche dies noch nicht nativ unterstützen.

Beispiel eines Eigenen HTML Elements:

<google-map lat="37.790" long="-122.390"></google-map>

Features

Elements Library

Polymer kommt bereits mit einer reichen Sammlung an vordefinierten Elementen welche nur noch eingebunden werden müssen und verwendet werden können.

Es gibt unzählige Element/Komponenten wie z.B. Slider, Navigationen, Buttons, Listen und vielen mehr.

Eigene Elemente

Mit Polymer kann man seine eigenen HTML Tags erstellen und so Boyler Plate Code verhindern und viel Zeit und Aufwand einsparen.

Bei der Definition eines eigenen Elements erstellt man ein Template in welchem normales HTML verwendet wird. In einem weiteren Schritt kann man Funktionalitäten mit JavaScript und dem Polymer Framework implementieren.

Folgend ein Beispiel von der Polymer Webseite:

```
<!-- Define element -->
<polymer-element name="my-counter" attributes="counter">
  <template>
    <style> /*...*/ </style>
    <div id="label"><content></content></div>
    Value: <span id="counterVal">{{counter}}</span><br>
    <button on-tap="{{increment}}">Increment</button>
  </template>
  <script>
    Polymer({
      counter: 0, // Default value
      counterChanged: function() {
        this.$.counterVal.classList.add('highlight');
```

```
      },
      increment: function() {
        this.counter++;
      }
    });
  </script>
</polymer-element>

<!-- Use element -->
<my-counter counter="10">Points</my-counter>
```

Quelle: https://www.polymer-project.org/0.5/

Browser Support

Polymer wird von den meisten gängigen Browsern unterstützt, jedoch meist nur in den allerneusten Versionen. Polymer wird nicht in so vielen Browser(-Versionen) unterstützt wie viele andere Frameworks – Beispielsweise wird im Internet Explorer Polymer nur ab Version 10 unterstützt.

Folgend eine Tabelle der Browserunterstützung:

	Chrome	Firefox	IE 10+	Chrome(Andr.)	Safari(IOS8.1)	Safari 8+
Template	✓	✓	✗	✓	✓	✓
HTML imports	✓	Dev flag*	✗	✓	✗	✗
Custom Elements	✓	Dev flag*	✗	✓	✗	✗
Shadow DOM	✓	Dev flag*	✗	✓	✗	✗

**Mozilla announced they will not ship an implementation of HTML Imports*
Quelle: https://www.polymer-project.org/0.5/resources/compatibility.html

Ein Ziel von Polymer ist es, dass die Funktionalität nativ in jedem Browser unterstützt wird. Chrom 36 ist der erste Browser welcher die Polymer Funktionalität nativ unterstützt.

Tools

Polymer Designer

Polymer bietet einen eigenen webbasierten drag&drop Designer für die Erstellung von App Prototypen.

https://polymer-designer.appspot.com/

SWOT Analyse

	Positiv (Hilfreich)	Negativ (Schädlich)
Stand: Heute	**Stärken** - Grosse Component Library - Eigene HTML Elemente - Prototyp Designer - Bringt viele JS Funktionalitäten mit sich / APIs	**Schwächen** - Beschränkter Browser Support - Phase „Developer Preview“

Stand: Zukunft (Vorhersage)	**Chancen** - Native Integration in Browser - Starker Support	**Gefahren** - Browser bieten alle Funktionalitäten von Polymer auch & macht das Framework überflüssig - Wird abgesetzt z.B. wegen AngularJS

Vergleich

Anwendung und Funktionalität

Bootstrap unterscheidet sich von Polymer und strebt nicht dieselben Ziele an.

Bootstrap ist mehr eine Bibliothek mit vordefinierten Komponenten und Responsive Design Support während Polymer zwar auch eine Bibliothek mit vordefinierten Komponenten besitzt, man aber auch eigene HTML Elemente erstellen kann.

	Bootstrap	**Polymer**
Komponenten Bibliothek	X	X
Custom HTML Elemente	-	X
Responsive Design Support	X	X

Browser Support

Vom Browser Support her ist Bootstrap ganz sicher vorne. Polymer wird nur in den neusten Versionen der gängigen Browser unterstützt (oder gar nur teilweise unterstützt) und befindet sich noch in der „developer preview“ Phase.

Fazit

Sowohl Bootstrap als auch Polymer sind beide sehr umfangreichen und hilfreichen Frameworks. Beide Frameworks erleichtern einem die Entwicklung einer WebApp und sparen einem viel Zeit ein.

Ein klarer Nachteil von Polymer ist der Browser Support. Viele Personen verwenden noch heute veraltete Browser Versionen und deshalb kann es schnell passieren dass eine WebApp welche mit Polymer gebaut wurde nicht ganz funktioniert bzw. nicht so aussieht wie sie sollte.

Für die Studienarbeit „Schulterklopfen WebApp“ empfehle ich ganz klar Bootstrap, da sie alle Funktionalitäten enthält welche gebraucht werden und es einen grösseren Browser Support mitbringt. Speziell würde die mangelnde Unterstützung in Safari ein Problem darstellen da (wahrscheinlich) einige Mitarbeiter ein IPhone besitzen.

Wireframes

Folgend werden alle bisher erstellten Wireframes dargestellt und erläutert.

Login

Abbildung 8 Wireframes: Login

Anmeldung beim Schulterklopfen webAPP mit dem Username & Password.

➔ Bei Möglichkeit mit Google+ Login

Zum momentanen Zeitpunkt ist noch nicht klar wie das Login- bzw. Accountsystem umgesetzt wird. In diesem Wireframe war die Überlegung wie folgt:

User können entweder einen „internen“ Account mit Username und Password haben oder sich über ihren Google Account einloggen.

User Home / User Profil

Abbildung 9 Wireframes: User Home / User Profile

Dies ist die „Home“ Seite des angemeldeten Users. Zuerst gibt es eine kleine Statistik der erhaltenen Schulterklopfer. Unmittelbar danach kommen alle erhaltenen Schulterklopfer, sortiert nach Datum.

#Neue/Ungelesene Schulterklopfer erhalten z.B. eine andere Hintergrundfarbe.

#Oben links auf dem Navigations Icon wird angezeigt, wie viele neue bzw. ungelesen Schulterklopfer vorhanden sind.

User Home Filter

Abbildung 10 Wireframes: User Home Filter

Die angezeigten Schulterklopfer können z.B. nach Datum oder Sender gefiltert werden.

Nach dem Klick auf „Anwenden“ werden die angezeigten Schulterklopfe auf der User Home Seite nach den angegebenen Kriterien gefiltert.

#Der Filter Bereich fährt von rechts nach links aus, die restliche Seitenfläche wird ausgegraut.

Menü / Navigation

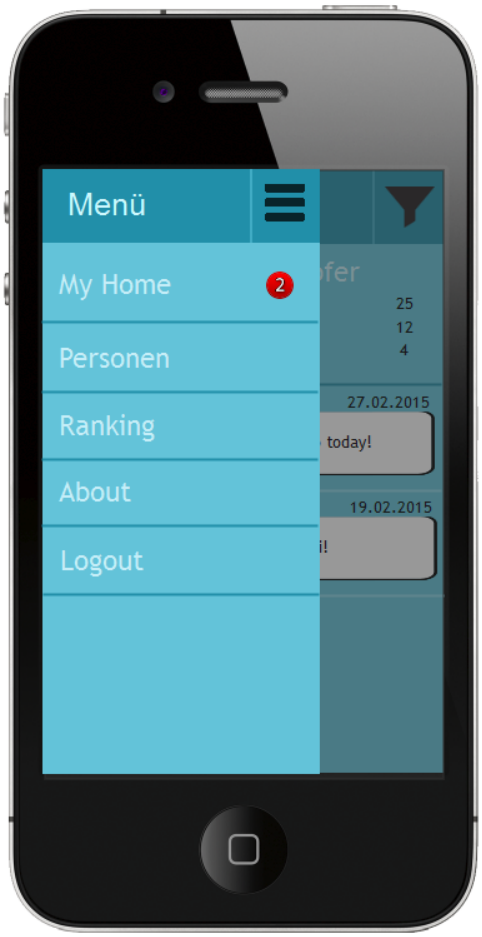


Abbildung 11 WIreframes: Navigation

Die Navigation slided von links nach rechts raus. Die Restliche Fläche der WebAPP wird ausgegraut.

Im Navigationspunkt „My Home“ wird die Anzahl an neuen/ungelesenen Schulterklopfer angezeigt.

Personen / Suche

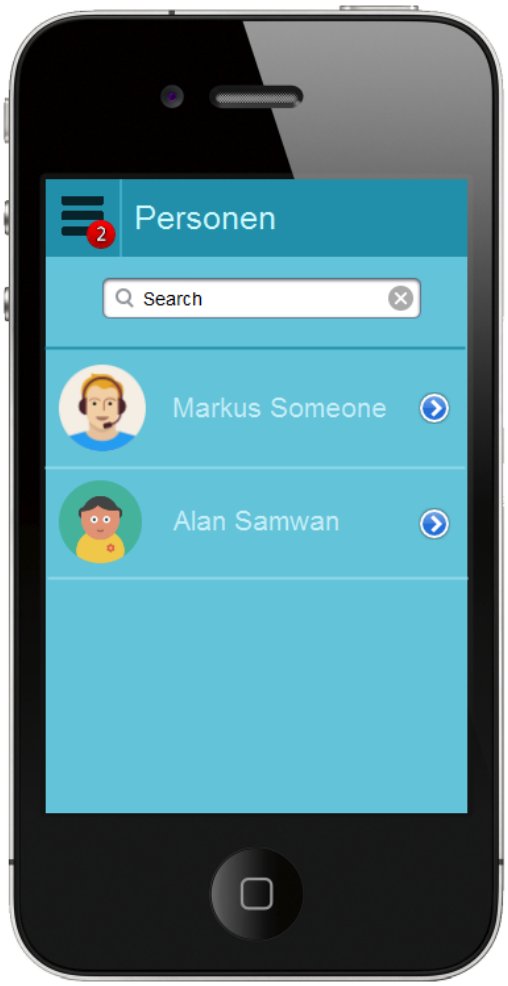


Abbildung 12 Wireframes: Personen / Suche

Auf dieser Seite kann nach Personen gesucht werden. Mit einem Klick auf eine Person gelangt man auf Ihr Profil (Siehe Wireframe 6).

#Die Resultate werden während der Eingabe neu geladen.

#Man könnte noch ein Shortcut-Button einfügen für einen neuen Schulterklopfer.

Person Profil

Abbildung 13 Wireframes: Personen Profil

Die Profil Seite einer anderen Person sieht ziemlich gleich aus wie die „2. My Home“ Seite. Hier werden alle Schulterklopfer des ausgewählten Users angezeigt.

Zusätzlich gibt es hier einen Button „Auf die Schultern klopfen“. Diesen kann man betätigen um dieser Person einen Schulterklopfer zu geben (Siehe Wireframes 7/8).

#Die angezeigten Schulterklopfer kann man auch hier nach Datum um Sender filtern.

#Die Schulterklopfer sind immer nach Datum abwärts sortiert.

„Neuer Schulterklopfer“

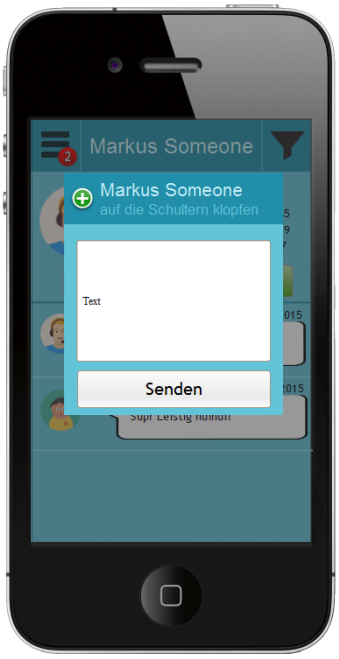


Abbildung 14 Wireframes: Neuer Schulterklopfer 1

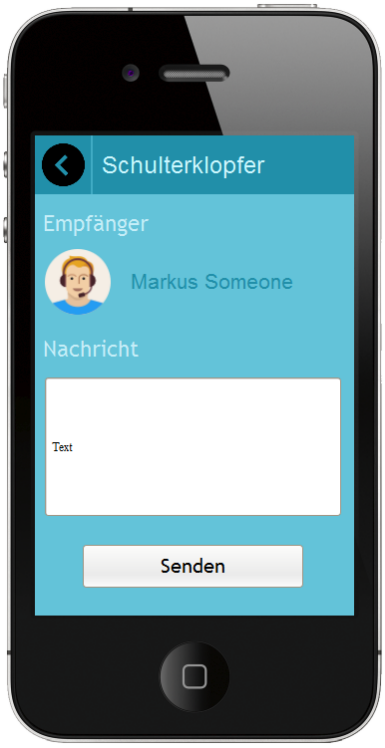


Abbildung 15 Wireframes: Neuer Schulterklopfer 2

Für die Darstellung der Seite auf welcher man einer Person einen Schulterklopfer geben kann gibt es die zwei oben dargestellten Varianten.

Diese Seite erreich man, indem man in Wireframe 6 auf den Button „Auf die Schulter klopfen“ klickt.

Variante 1: „PopUp“

Eine Variante wäre, dass auf der User Profile Seite ein PopUp erscheint, in welchem man dann den Text für den Schulterklopfer eingeben kann.

Variante 2: Eigene Seite

Eine andere Variante wäre, dass ein Schulterklopfer auf einer eigenen Seite erfasst & verschickt wird.

Registration

Abbildung 16 WIreframes: Registration

Auf diese Seite gelangt man über die Login-Seite (Startseite). Hier können sich neue User für einen „internen“ Account registrieren.

Architektur

Einführung

Übersicht

Zuerst wird eine grobe Übersicht gegeben, welche zur Orientierung dienen soll.
Danach werden auf Architektonische Ziele und deren Einschränkungen eingegangen.
Die Logische Architektur wird pro Package genau dargestellt, also deren Klassenstruktur sowie Schnittstellen und wichtige interne Abläufe.
Danach werden Informationen zum Deployment, der Datenhaltung und Kenngrössen der Leistung gegeben.

Systemübersicht

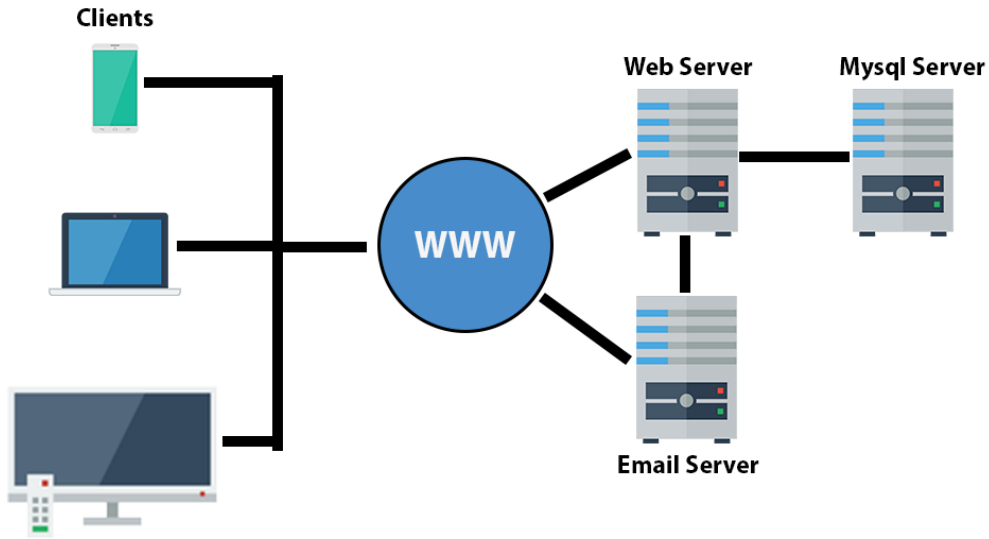


Abbildung 17 Systemübersicht

Verschiedene Clients, unteranderem Smartphones und PC's, haben via Web Browser Zugriff auf die Web APP. Die Web APP befindet sich auf dem Webserver, die Daten hingegen werden extern auf in einer Datenbank auf einem Mysql Server abgelegt.

In dieser Web APP wird es den Benutzern möglich sein, Email Notifications zu erhalten. Bei solch einer Notification, verschickt der Web Server eine Email über einen konfigurierten Email Server.

Architektonische Ziele & Einschränkungen

Softwareanforderungen

Die Software muss auf einem Tomcat Application Server laufen.

Die Datenhaltung erfolgt in einer Mysql Datenbank und wird mit JPA verwaltet.

Das UI der WebApp soll auf den gängigen iOS und Android Browsern übersichtlich dargestellt werden (Responisve Design).

Security

An die WebApp werden keine speziellen Anforderungen an die Sicherheit gestellt da es keine Sensiblen Daten enthält.

MVC

Die Software wird nach dem MVC (web) Prinzip von Spring aufgebaut.
Weitere Informationen zum Spring MVC Prinzip sind im Kapitel „Developer View“ vorhanden.

Logische Architektur

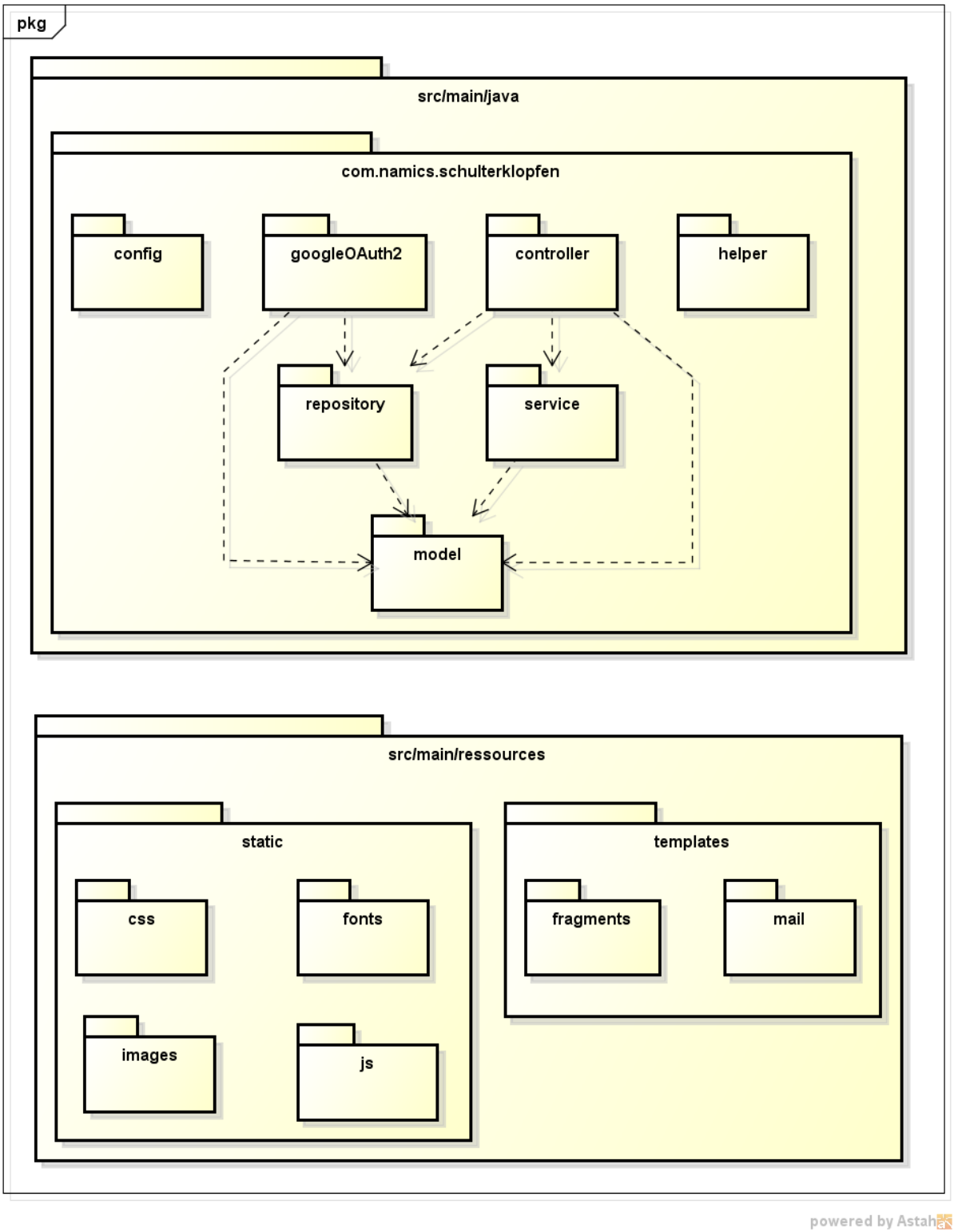


Abbildung 18 Package Diagramm

STAN Struktur

Folgend wird die Package Struktur, deren Abhängigkeiten sowie auch die Richtung der Abhängigkeiten dargestellt. Die Packages beziehen sich auf das Hauptpackage „**com.namics.schulterklopfer**“.

Anhand der Pfeile im Diagramm erkennt man wer auf wen zugreift. Die Zahl beim Pfeil sagt aus, wie viele Abhängigkeiten auf den Inhalt des jeweiligen Packages existieren.

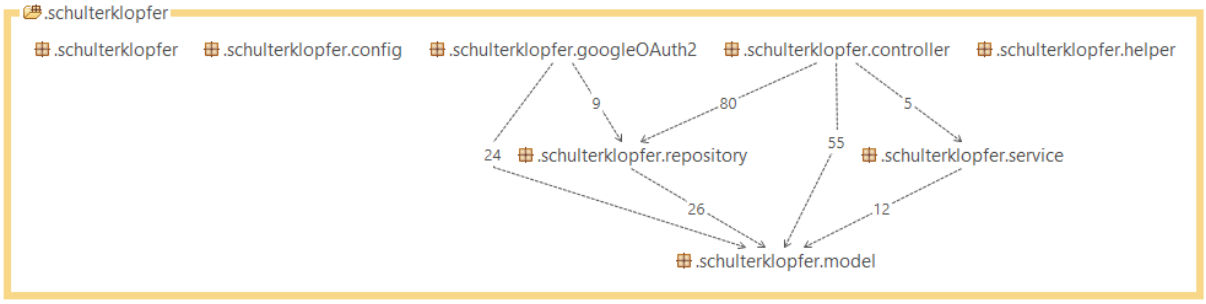


Abbildung 19 Package-Abhängigkeiten com.namics.schulterklopfer

Com.namics.schulterklopfer

Klassen im base-Package haben die Aufgabe das Programm zu initialisieren & zu starten.

Klassenstruktur

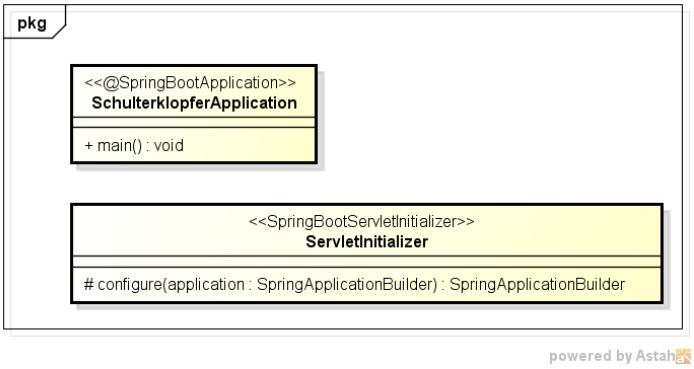


Abbildung 20 Klassendiagramm com.namics.schulterklopfer

Klasse	Funktion
SchulterklopferApplication	Enthält **main()**-Methode. Startet Spring Boot.
ServletInitializer	Ruft Spring-Boot auf, das Servlet zu initialisieren.

Com.namics.schulterklopfer.config

Dieses Package enthält die Hauptkonfigurationen für die Spring Applikation.

Klassenstruktur

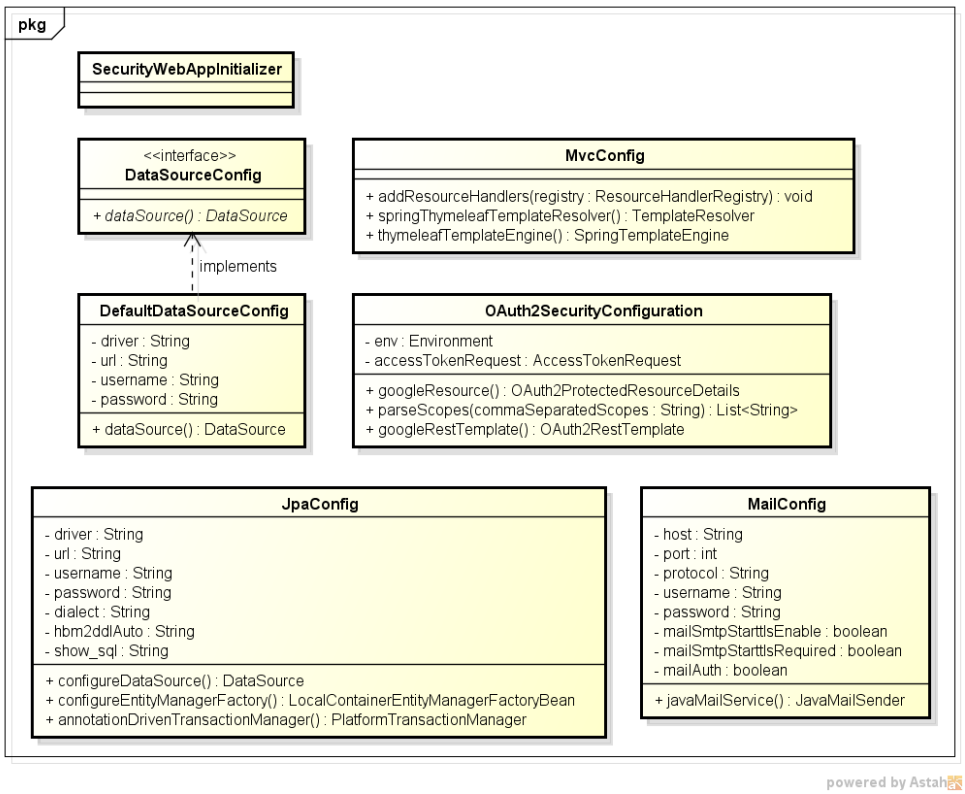


Abbildung 21 Klassendiagramm com.namics.schulterklopfer.config

Klasse	Funktion
DataSourceConfig	Interface für das **DefaultDataSourceConfig**
DefaultDataSourceConfig	DataSource für die Verbindung zur Datenbank bereitstellen.
JpaConfig	Allgemeine Konfigurationen für die Verwendung von JPA.
MailConfig	Konfiguration für die Verwendung von Mail-Versand
MvcConfig	Konfiguration für Spring MVC (z.B. TemplateResolver, Resourcehandler, …)
OAuth2SecurityConfiguration	Konfiguration für OAuth2 Verwendung.
SecurityWebAppInitializer	Allgemeine Spring Security Konfiguration (z.B. public/private Pages)

Com.namics.schulterklopfer.controller

Dieses Package enthält alle Controller. Die Controller definieren welche View verwendet werden soll und liefern die benötigten Models. Controller Klassen können für mehrere Seiten- & Request-Arten zuständig sein.

Klassenstruktur

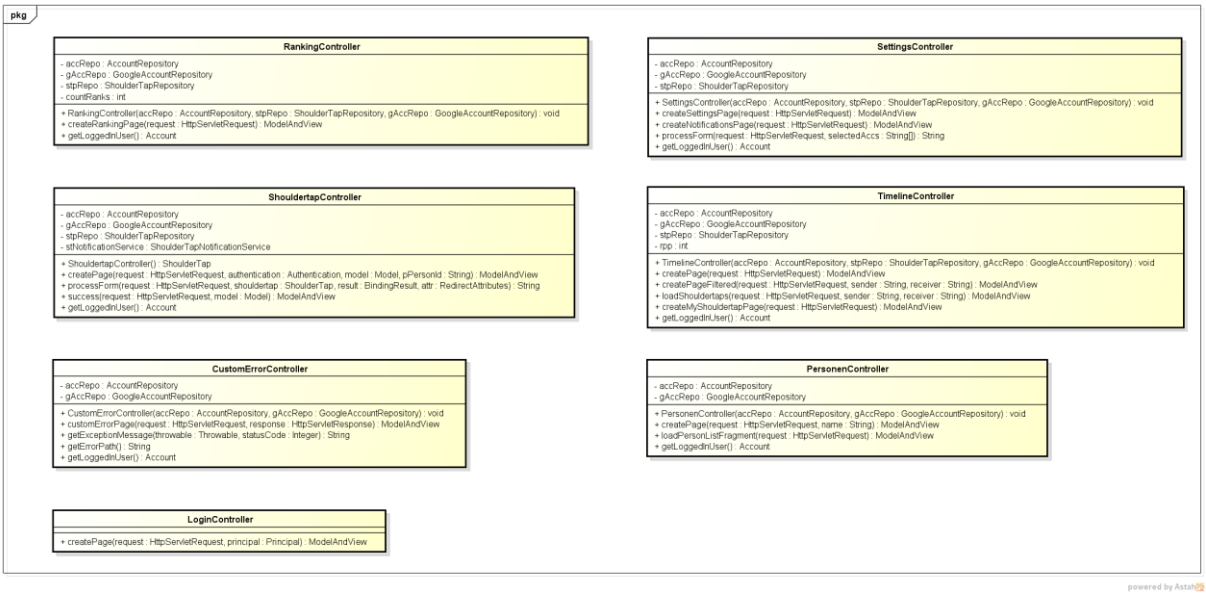


Abbildung 22 Klassendiagramm com.namics.schulterklopfer.controller

Klasse	Funktion
CustomErrorController	Spezifiziert eine custom Fehlerseite (z.B. für 404)
LoginController	
PersonenController	Zuständig z.B. für die Personensuche.
RankingController	
ShouldertapController	Zuständig für das Erstellen eines neuen Schulterklopfers.
TimelineController	Zuständig für die Anzeige-Seiten von Schulterklopfern.
SettingsController	Zuständig für die Anzeige von Einstellungs Seiten (z.B. Email Notifications)

Com.namics.schulterklopfer.googleOAuth2

Dieses Package enthält Klassen für die Funktionalität des Google Logins via OAuth 2.
Alle in diesem Package enthaltenen Klassen stammen aus folgender Quelle und wurden übererarbeitet:

https://github.com/skate056/spring-security-oauth2-google

Klassenstruktur

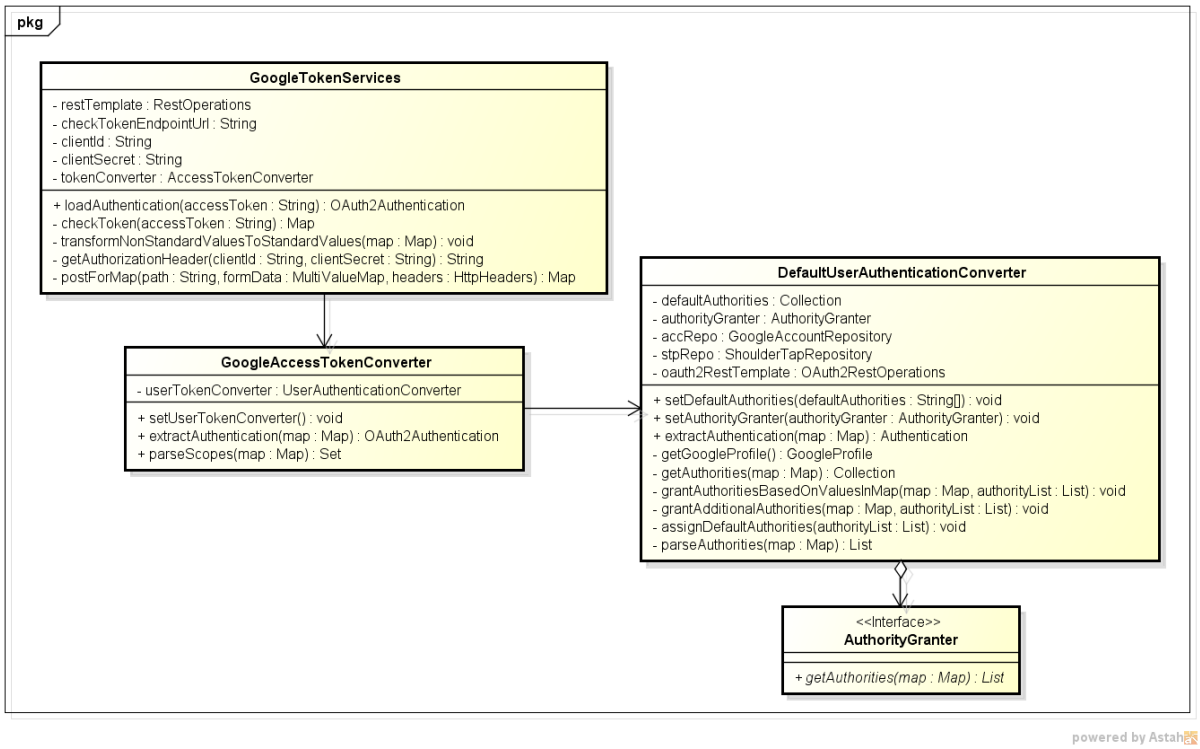


Abbildung 23 Klassendiagramm com.namics.schulterklopfer.googleOAuth2

Klasse	Funktion
AuthorityGranter	Interface
DefaultUserAuthenticationConverter	Loggt unter anderem User ein mit Google OAuth2 in Spring Security.
GoogleAccessTokenConverter	Schnittstelle für Google Access Tokens und OAuth2 Tokens in Spring.
GoogleTokenServices	Parametrierung und Services für Google OAuth2.

Schnittstellen

Der **GoogleAccessTokenConverter** dient als Schnittstelle zwischen Google OAuth2 Tokens und der OAuth2 Token Implementierung von Spring Security OAuth2.

Wichtige interne Abläufe

In der Klasse **DefaultUserAuthenticationConverter** gibt es eine Methode „**extractAuthentication(…)**“. Diese Methode liefert ein Authentication-Objekt zurück. In dieser Methode wird festgelegt, ob es sich um eine Gültige Anmeldung handelt oder nicht.

In dieser Methode wird auch ein neuer Account erstellt vom Typ GoogleAccount falls sich der User zum ersten Mal einloggt.

Falls zukünftig weitere Authentifizierungen nötig sind (z.B. Überprüfung der E-Mail des Google Accounts), muss dies in dieser Methode definiert werden.

Com.namics.schulterklopfer.helper

Dieses Package enthält allgemeine Hilfsklassen für die Applikation.

Klassenstruktur

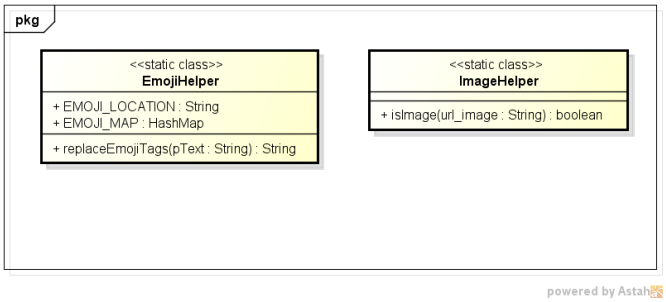


Abbildung 24 Klassendiagramm com.namics.schulterklopfer.helper

Klasse	Funktion
EmojiHelper	Hilfsklasse für die Benutzung von Emojis.
ImageHelper	Hilfsklasse für den Umgang mit Bildern.

Com.namics.schulterklopfer.model

Dieses Package enthält die Model-Klassen der Applikation.
Model-Klassen dessen Objekte in der Datenbank persistiert werden müssen, sind mit JPA-Annotations bestückt und gelten als „**Entitätsklassen**“.

Klassenstruktur

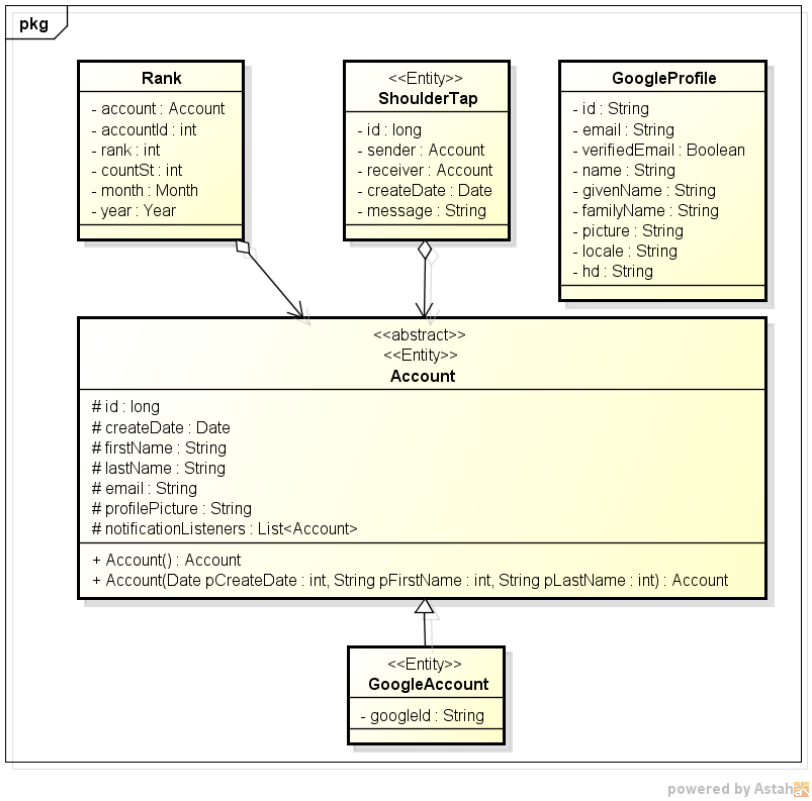


Abbildung 25 Klassendiagramm com.namics.schulterklopfer.model

Klasse	Funktion
Account (Entity)	Basis Typ eines Benutzeraccounts
GoogleAccount (Entity)	Subtyp eines Accounts
ShoulderTap (Entity)	

GoogleProfile	Hilfsobjekt für eingeloggten Google-User.
Rank	Hilfsobjekt für das Ranking.

Com.namics.schulterklopfer.repository

Dieses Package enthält die Repositories der Model-Entities. Sie verwalten die Model-Entities und bieten diverse Funktionen z.B. für die Persistierung eines Objekts, Suchabfragen etc.

Die Repositories sind mit Spring Data JPA-Annotations versehen.

Klassenstruktur

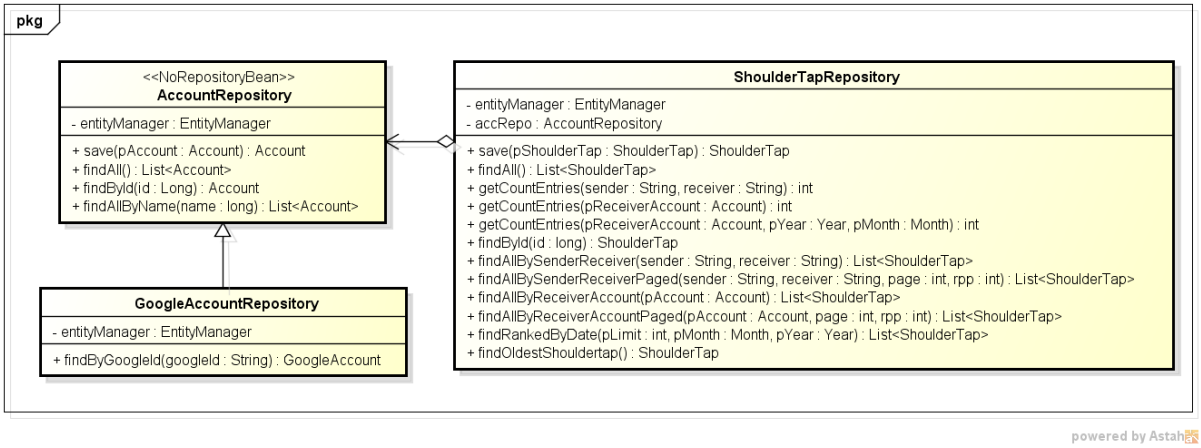


Abbildung 26 Klassendiagramm com.namics.schulterklopfer.repository

Klasse	Funktion
AccountRepository	Funktionen für die Speicherung und das Auslesen von Accounts
GoogleAccountRepository	Spezifische Funktionen für GoogleAccount-Entities
ShoulderTapRepository	Funktionen für die Speicherung und das Auslesen von ShoulderTaps

Com.namics.schulterklopfer.service

In diesem Package befinden sich Service-Klassen (Business Logic) welche bestimmte Aufgaben bearbeiten.

Klassenstruktur

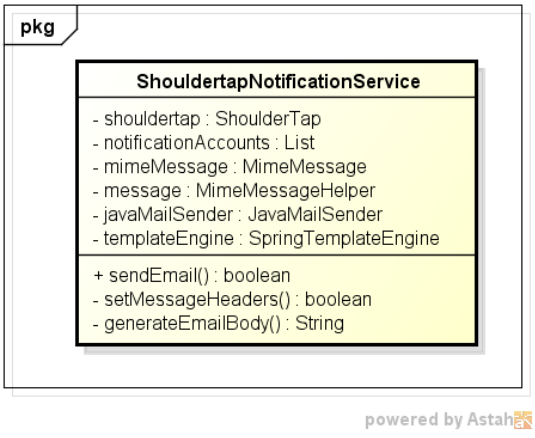


Abbildung 27 Klassendiagramm com.namics.schulterklopfer.service

Klasse	Funktion

ShouldertapNotificationService	Erstellung und Versand von Email Notifications von neuen ShoulderTaps.

Wichtige interne Abläufe

Klasse: ShouldertapNotificationService

Um ein Mail zu verschicken muss der entsprechende ShoulderTap und alle Empfänger (min. 1) dem Objekt des Services übergeben werden.

Mit Aufruf der Methode **sendEmail()** wird anhand eines Templates ein HTML-Email generiert und an alle Empfänger verschickt.

Wichtig: Bei mehreren Empfängern werden nicht mehrere Mails verschickt sondern diese werden als **BC(Blind Copy)**-Empfänger hinzugefügt.

Spezifische Abläufe und Umsetzungsmethoden

Spring MVC Web

Das Spring MVC Web vereinfacht die Entwicklung von Web Applikationen nach dem MVC Prinzip.

Im Spring MVC Modell sieht dies wie folgt aus:

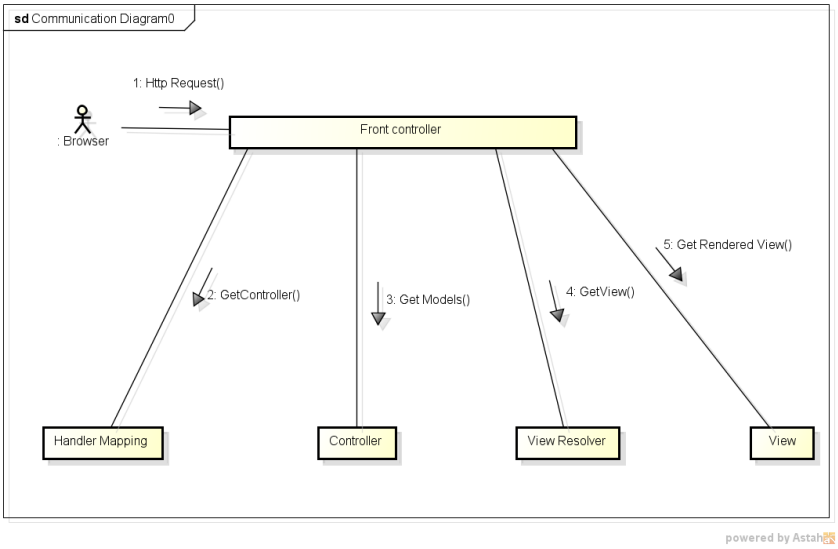


Abbildung 28 Spring MVC

1. Ein Client schickt einen Seitenrequest an den Web Server (Dieser wird an einen Front Controller weitergeleitet)
2. Anhand des Requests ermittelt das Handler Mapping welcher Controller aufgerufen werden muss.
3. Der Controller wird aufgerufen. Dieser liefert die Models (& evtl. weitere Daten) zurück welche für die Seite gebraucht werden.
4. Der View Resolver entscheidet welche view (in der Regel ein HTML-Template) verwendet werden soll. Die Definition der View kann auch schon direkt im Controller definiert werden.

Zum Schluss wird die View anhand der vom Controller bereitgestellten Daten gerendert und eine HTML-Seite an den Front Controller zurückgegeben. Diese Seite wird dann dem Client zurückgeschickt.

Google Login mit OAuth2

Spring Security OAuth2 Google ist kein offizielles Spring Plug-In. Es wurde von einem Entwickler implementiert, welcher selbst Spring verwendet für Applikationsentwicklungen.

Die offizielle Seite dieses Plug-Ins ist auf GitHUB unter folgender URL zu finden:

https://github.com/skate056/spring-security-oauth2-google

Integration

Folgend wird beschrieben, wie Spring Security OAuth2 Google in das Projekt integriert wurde.

Benötigte Klassen / Konfigurationen

- AuthorityGranter.java
- DefaultUserAuthenticationConverter.java
- GoogleAccessTokenConverter.java
- GoogleTokenServices
- **OAuth2SecurityConfiguration.java**
- **Spring Security Configuration** (in diesem Beispiel in XML umgesetzt)

Google API Zugangsdaten

Um die Google OAuth2 API benutzen zu können, muss man API Zugangsdaten einholen. Dies kann man auf folgender Webseite:

https://console.developers.google.com

Man muss ein neues Projekt erstellen und API Zugangsdaten für die Google OAuth API anfordern.

Wenn dies getan ist erhält man folgende Daten:

- Client-ID
- E-Mail-Adresse
- Clientschlüssel
- Weiterleitungs-URIs (Muss man selber definieren)
- JavaScript-Quellen (Muss man selber definieren, optional)

Konfiguration: Google API Benutzung

Im **application.properties** File (Globales Konfigurationsfile in Spring) müssen folgende Parameter definiert werden:

`google.client.id`	**Client-ID für den API-Zugriff**
`google.client.secret`	Secret (Clientschlüssel) für den API-Zugriff

`google.auth.scope`	Hier definiert man, auf welche Daten zugegriffen werden will (es können mehrere definiert werden – Kommagetrennt).
	Beispiel:
	`https://www.googleapis.com/auth/userinfo.email,`
	`https://www.googleapis.com/auth/userinfo.profile`

`google.accessTokenUri`	URI für Access-Token: `https://accounts.google.com/o/oauth2/token`
`google.userAuthorizationUri`	URI für die Benutzer Authorization: `https://accounts.google.com/o/oauth2/auth`

google.authorization.code	authorization_code
google.preestablished.redirect.url	Redirection URL: Die URL von welcher zu Google weitergeleitet wird. Z.B.: http://yoursite.com/login/googleLogin

Spring Security Beispielskonfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<b:beans xmlns:sec="http://www.springframework.org/schema/security"
         xmlns:b="http://www.springframework.org/schema/beans"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">
    <sec:http auto-config="true" use-expressions="true" entry-point-
ref="clientAuthenticationEntryPoint">
        <sec:http-basic/>
        <sec:logout logout-success-url="/" logout-url="/logout" delete-cookies="JSESSIONID"/>
        <sec:anonymous enabled="false"/>
        <!-- This is the crucial part and the wiring is very important -->
        <!--
            The order in which these filters execute are very important. oauth2ClientContextFilter must
be invoked before
            oAuth2AuthenticationProcessingFilter, that's because when a redirect to Google is required,
oAuth2AuthenticationProcessingFilter
            throws a UserRedirectException which the oauth2ClientContextFilter handles and generates a
redirect request to Google.
            Subsequently the response from Google is handled by the
oAuth2AuthenticationProcessingFilter to populate the
            Authentication object and stored in the SecurityContext
        -->
        <sec:custom-filter ref="oauth2ClientContextFilter" after="EXCEPTION_TRANSLATION_FILTER"/>
        <sec:custom-filter ref="oAuth2AuthenticationProcessingFilter"
before="FILTER_SECURITY_INTERCEPTOR"/>
    </sec:http>

    <b:bean id="oAuth2AuthenticationProcessingFilter"
class="org.springframework.security.oauth2.client.filter.OAuth2ClientAuthenticationProcessingFilter">
        <b:constructor-arg name="defaultFilterProcessesUrl" value="/login/googleLogin"/>
        <b:property name="restTemplate" ref="googleRestTemplate"/>
        <b:property name="tokenServices" ref="tokenServices"/>
    </b:bean>

    <!--
        These token classes are mostly a clone of the Spring classes but have the structure modified so
that the response
        from Google can be handled.
    -->
    <b:bean id="tokenServices" class="com.namics.schulterklopfer.googleOAuth2.GoogleTokenServices">
        <b:property name="checkTokenEndpointUrl"
value="https://www.googleapis.com/oauth2/v1/tokeninfo"/>
        <b:property name="clientId" value="${google.client.id}"/>
        <b:property name="clientSecret" value="${google.client.secret}"/>
        <b:property name="accessTokenConverter">
            <b:bean class="com.namics.schulterklopfer.googleOAuth2.GoogleAccessTokenConverter">
                <b:property name="userTokenConverter">
                    <b:bean
class="com.namics.schulterklopfer.googleOAuth2.DefaultUserAuthenticationConverter"/>
                </b:property>
            </b:bean>
        </b:property>
    </b:bean>

<!--
```

```
This authentication entry point is used for all the unauthenticated or unauthorised sessions to be
directed to the /googleLogin URL which is then intercepted by the oAuth2AuthenticationProcessingFilter
to trigger authentication from Google.
-->
    <b:bean id="clientAuthenticationEntryPoint"
class="org.springframework.security.web.authentication.LoginUrlAuthenticationEntryPoint">
        <b:property name="loginFormUrl" value="/login/googleLogin"/>
    </b:bean>

    <sec:authentication-manager alias="alternateAuthenticationManager">
        <sec:authentication-provider>
            <sec:user-service>
                <sec:user name="user" password="password" authorities="DOMAIN_USER"/>
            </sec:user-service>
        </sec:authentication-provider>
    </sec:authentication-manager>
</b:beans>
```

Die im obigen Config Beispiel rot- & fett-markierten stellen müssen angepasst werden. Dies ist die URL auf welcher der User dann zu Google für das Login weitergeleitet wird.

E-Mail Notifications

Allgemein

User können Email Notifications erhalten wenn sie selber oder andere User Schulterklopfer erhalten. Der User kann selbst wählen, bei welchen Usern er per Mail informiert werden möchte.

In der Model Klasse „Account“ gibt es eine Liste „`notificationListeners`“ in welcher die jeweiligen Accounts abgelegt sind, welche notifiziert werden sollen wenn dieses Objekt einen Schulterklopfer erhält.

Mail Konfiguration

Die Einstellungen für den Mail Server befinden sich unter *src/main/resources/**application.properties***.

Es gibt folgende Einstellungen:

Code	Anmerkung
email.host	Hostadresse des Mail-Servers
email.port	
email.protocol	Protokoll z.B. **SMTP**
email.username	
email.password	
email.mail.smtp.starttls.enable	True / False
email.mail.smtp.starttls.required	True / False
email.mail.auth	True / False

Für die Konfiguration und Aktivierung des Mail-Versands gibt es im Package **com.namics.schulterklopfer.config** eine Mail Konfigurations Klasse **MailConfig.java**.

Diese Config-Klasse enthält ein Bean welches ein Objekt vom Typ JavaMailSender zurückgibt. Anhand dieser kann man dann über den konfigurierten Mail Account Emails verschicken.

Mail Versand

Für die Generierung des Email-Templates und des Versandes gibt es eine Service Klasse „**ShouldertapNotificationService“**.

Wenn ein neuer Schulterklopfer verschickt wird, wird ein neues Objekt der oben genannten Klasse erstellt. Bevor das Mail-Template generiert & verschickt werden kann, muss der zugehörige Schulterklopfer und eine Liste mit den Empfängern (Minimum 1) gesetzt werden.

Anschliessend kann die Methode **sendEmail()** aufgerufen werden, welche dann das Email Template generiert und verschickt.

Wichtig: Bei mehreren Empfängern werden **NICHT** mehrere Emails verschickt, sondern werden alle weitere Empfänger in den BC (Blind Copy) eingetragen.

Libraries / Frameworks

Spring

Die WebAPP baut auf dem Spring Framework auf. Folgende Spring Module werden eingesetzt:

- Spring Boot ➔ Projektverwaltung/Generierung
  - Siehe Kapitel *Development View*
- Spring MVC ➔ Vereinfacht die Erstellung einer MVC basierten Web APP
  - Siehe Kapitel *Development View*
- Spring Security ➔ Allgemeine Security Basics
- Spring Data JPA ➔ Daten mit DB
- Thymeleaf ➔ Template Engine

Spring Boot

Mit Spring Boot kann man stand-alone Spring-basierte Applikationen generieren – so muss man die ganzen Imports etc. nicht selber machen.

http://projects.spring.io/spring-boot/

Spring MVC (Web)

Spring MVC ist eine Extension des Spring Frameworks welches einem ermöglicht Web-Applikationen mit einer Serverseitigen MVC-Architektur zu erstellen.

Spring MVC ist bereits im Spring Framework (Core) enthalten.

http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

Spring Security

Spring Security integriert diverse Sicherheits-Mechanismen wie z.B. ein vorimplementiertes Login-System, geschützte Seiten etc.

http://projects.spring.io/spring-security/

Spring Data JPA

Integration für die Verwendung von JPA in Spring.

http://projects.spring.io/spring-data-jpa/

Thymeleaf

Template Engine mit welcher man Templates erstellen kann die automatische gerendert werden. In diesem Projekt wurde eine Thymeleaf Version verwendet welche speziell für Spring angepasst wurde.

http://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html

Bootstrap

Bootstrap wird als HTML/CSS/JS Framework für das Frontend verwendet. Dies bietet vordefinierte Web-Komponenten und Unterstützung für Responsive Design.

http://getbootstrap.com/

JQuery

JQuery ist ein Java Script Framework welches diverse vorimplementierte Funktionalitäten bietet. Dies wird unter anderem auch von Bootstrap verwendet.

https://jquery.com/

Development View

Spring Boot

Mit Spring Boot kann man stand-alone Spring-basierte Applikationen generieren – so muss man die ganzen Imports etc. nicht selber machen.

Features

- Stand-alone Applikationen erstellen
- Tomcat, Jetty oder Undertow direkt integrieren
- Spring automatisch konfigurieren wo möglich
- Production-ready features wie metrics, health checks und externalized configurations lieferen
- Keine XML Konfiguration und keine Code Generierung
- MAVEN Projekt generieren

Basis Projekt generieren

Unter der URL http://start.spring.io/ (Spring INITIALIZR) kann man sich online ein Basis Projekt generieren lassen.

Man kann folgende Konfigurationen vornehmen:

- Projekt Metadaten definieren
- Dependencies auswählen
    - Core
    - Web
    - Data
    - Template Engines
    - Database
    - Social
    - I/O
    - Ops

Prozesse und Threads

In dieser Applikation gibt es nur einen Thread welcher von Spring verwaltet wird. Spezielle Angaben dazu gibt es keine.

Deployment

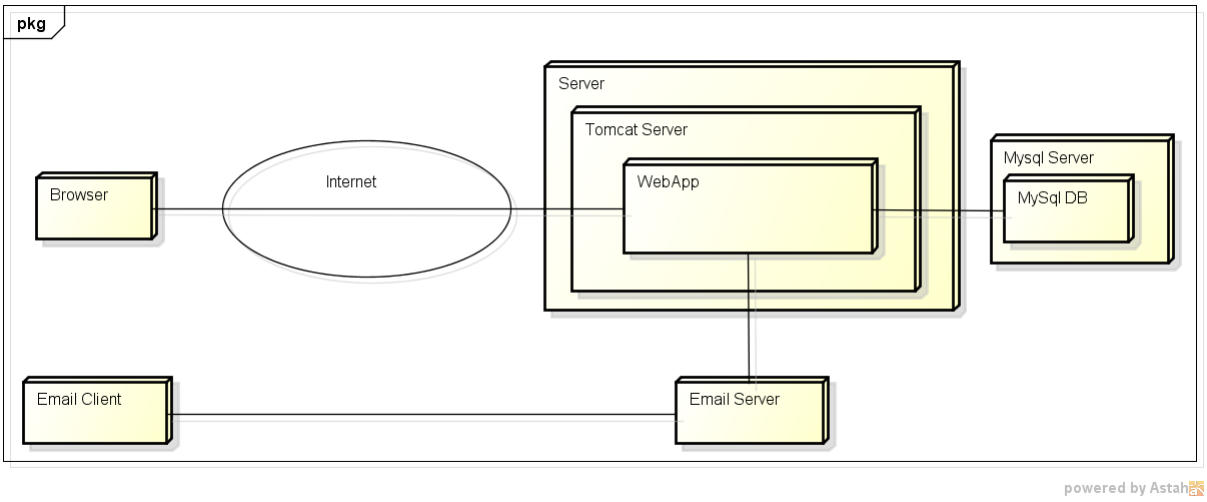


Abbildung 29 Deployment Diagramm

Die WebApp wird als WAR-File abgeliefert und wird auf einer Tomcat Server Instanz ausgeführt. Die Datenbank befindet sich voraussichtlich auf dem gleichen Server. Das Projekt wurde mit MAVEN erstellt welches einen integrierten Tomcat Server beinhaltet. Die Applikation kann also einfach gestartet werden.

Die MySQL Datenbank wird extern erstellt und verwaltet.

Benutzer greifen via Web-Browser auf die WebApp zu welche voraussichtlich unter der Domain schulterklopfen.namics.com abrufbar ist.

Datenspeicherung

Konfigurationen

Konfigurationen wie z.B. Datenbankverbindungen werden in einem externen Konfigurations-File gespeichert um Änderungen einfach zu halten. Das Konfigurations-File wird nach einem Spring System umgesetzt, so dass die Konfigurationswerte vom Programm einfach ausgelesen werden können.

Siehe folgende URL für eine genauere Beschreibung der Konfigurationsmöglichkeiten:

http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

Daten / Datenmodell

Daten von der WebApp (z.B. Benutzerprofile, Schulterklopfer etc.) werden in einer MySQL Datenbank angelegt.

In diesem Projekt wird eine JPA verwendet (Spring Data JPA) mit welcher die Daten verwaltet werden. Die Erstellung der Tabellen und Verknüpfungen wird von JPA automatisch übernommen.

JPA Entities

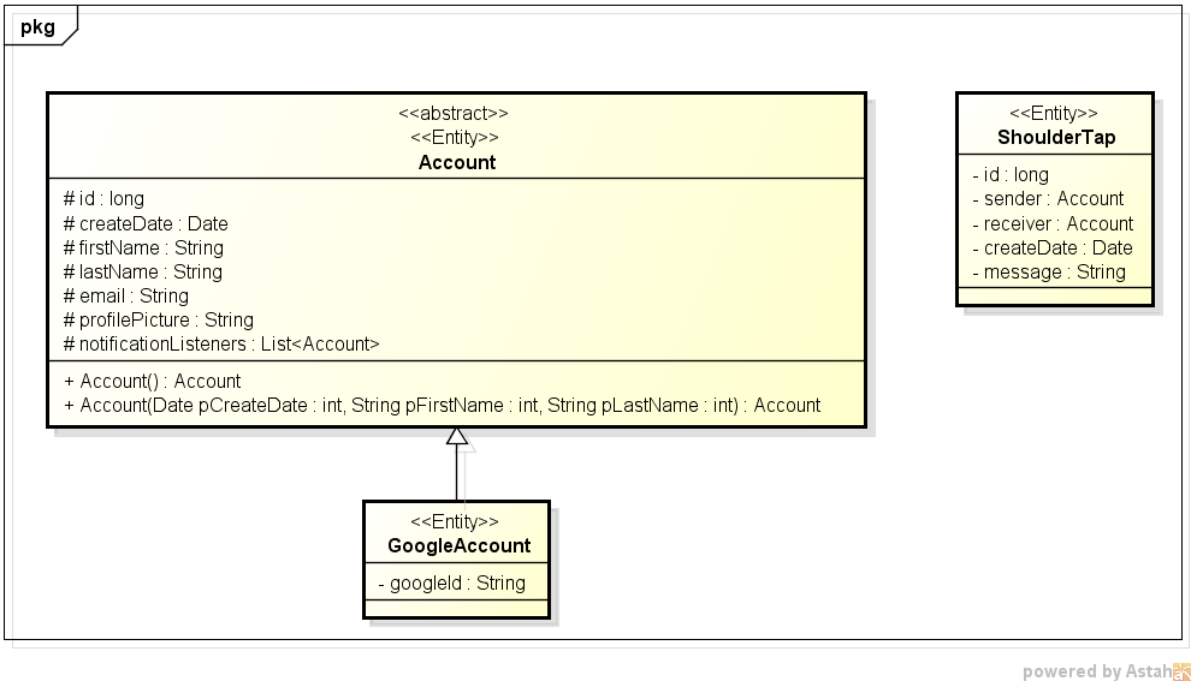


Abbildung 30 Klassendiagramm JPA Entities

Es gibt die im obigen Diagramm dargestellten Entitäten in diesem Projekt. All diese werden in der Datenbank persistiert. Bei der Vererbung von GoogleAccount zu Account gibt es eine geteilte Tabelle (In JPA wird dies *SINGLE TABLE INHERITANCE* genannt).

JPA Repositories

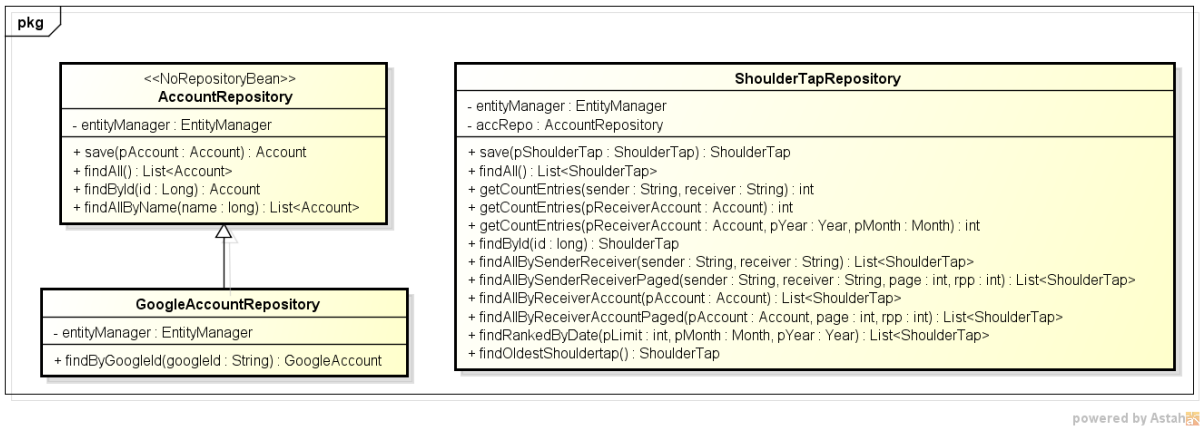


Abbildung 31 Klassendiagramm JPA Repositories

Für die Verwaltung der Entitäten gibt es jeweils ein Repository. Das Repository ist zuständig für jegliche CRUD-Funktionalitäten der Entitäten.

Um den Zugriff auf die Datenbank zentralisiert und gebündelt zu halten, sollen jegliche Zugriffe auf die jeweiligen Objekte über die Repositories geschehen.

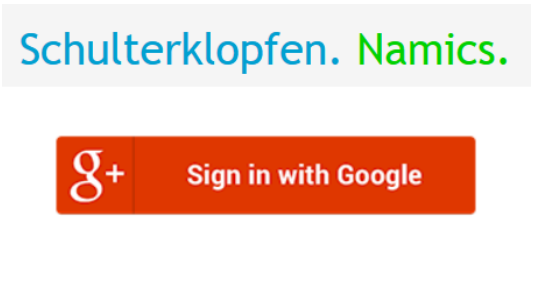
Grössen und Leistung

Bezüglich Grössen und Leistung werden keine speziellen Anforderungen an die WebAPP gestellt. Die WebApp wird voraussichtlich minimale Leistungsanforderungen haben und wird maximal 600 Benutzer haben. Diese Leistungsanforderungen erfordern keine Planung.

Externes Design

In diesem Kapitel werden die Ergebnisse der Applikation anhand von Screenshots aufgezeigt.

Login



Dies ist die Login Seite der WebApp. Dies ist die einzige öffentliche Seite der WebApp.

Mit einem Klick auf den Button „Sign in with Google“ wird man zu Google weitergeleitet (Siehe nächster Screenshot).

Abbildung 32 Externes Design: Login

Auf der Google Seite kann der Benutzer das Konto wählen mit welchem er sich einloggen möchte. Nach dem Login und Bestätigung der Rechte welche die WebApp anfordert, wird der Benutzer zur Timeline weitergeleitet.

Abbildung 33 Externes Design: Login 2

Timeline

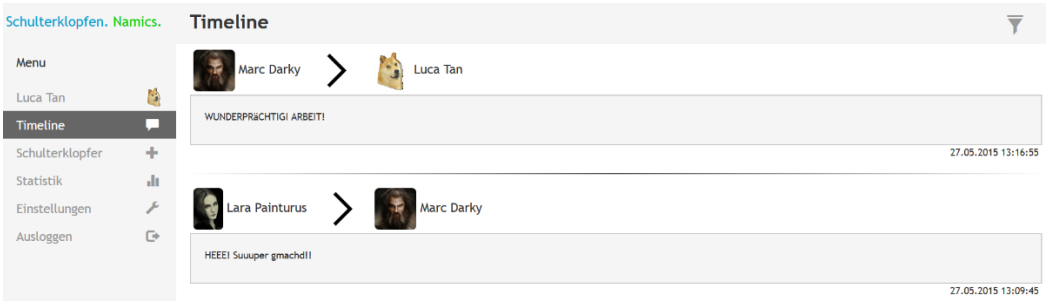


Abbildung 34 Externes Design: Timeline

In der Timeline werden alle Schulterklopfer von allen Usern, sortiert nach Datum, aufgelistet.

Mit einem Klick auf das Filter-Symbol im oberen rechten Ecken kann man die Auflistung der Schulterklopfer nach Sender und Empfänger Filtern:

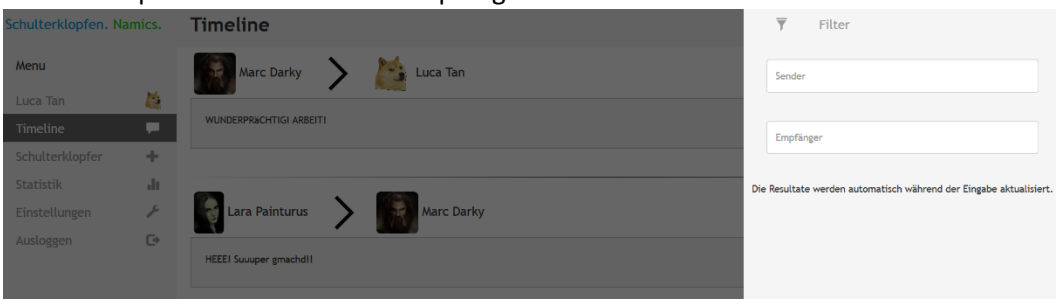


Abbildung 35 Externes Design: Timeline Filter

Meine Schulterklopfer

Abbildung 36 Externes Design: Meine Schulterklopfer

Auf der Seite „Meine Schulterklopfer“ werden nur die Schulterklopfer angezeigt, welche der User erhalten hat.

Zusätzlich beinhaltet die Seite eine kleine persönliche Statistik zu der Anzahl der erhaltenen Schulterklopfer nach Monat, Jahr und Insgesamt.

Schulterklopfer geben

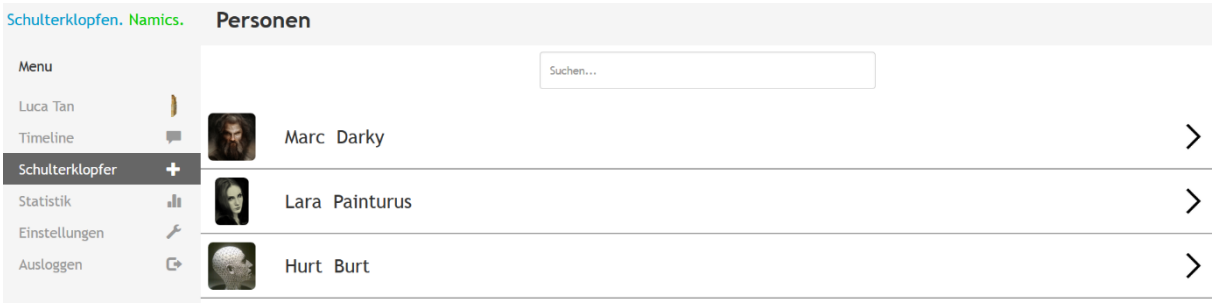


Abbildung 37 Externes Design: Schulterklopfer geben 1

Möchte ein User jemandem auf die Schultern klopfen, muss er vorerst eine Person wählen. Per Klick auf einen Personeneintrag, gelingt er zu einer Eingabemaske: Siehe nächster Screenshot.

Nachdem der User die Person ausgewählt hat welcher er auf die Schultern klopfen möchte, kann er eine kurze Nachricht schreiben und sein Lob aussprechen.

Per Klick auf den Button „Absenden“ wird der Schulterklopfer abgeschickt.

Abbildung 38 Externes Design: Schulterklopfer geben 2

Abbildung 39 Externes Design: Schulterklopfer geben 3

Statistik

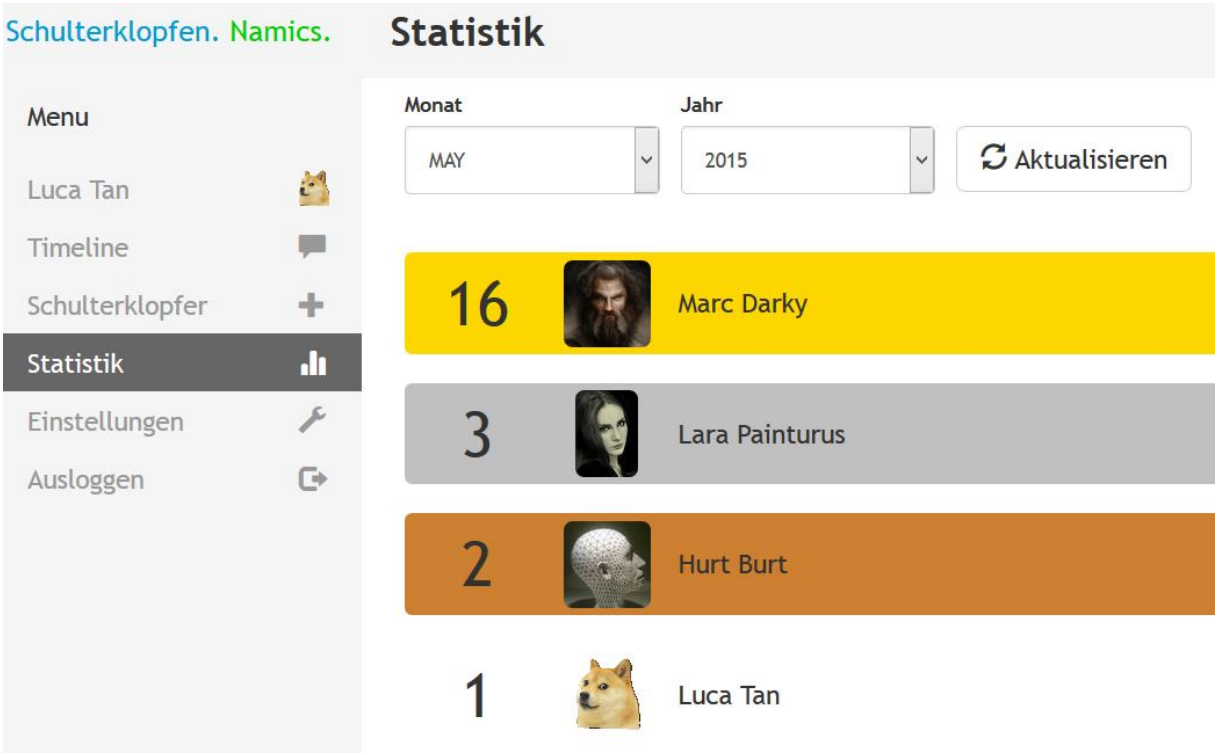


Abbildung 40 Externes Design: Statistik

Die Statistik bezieht sich jeweils auf einen Monat. Hier werden die User aufgelistet, welche am meisten Schulterklopfer erhalten haben – die Anzahl der angezeigten Personen ist konfigurierbar.

Um die Statistik von vorherigen Monaten anzuzeigen, kann in den oben stehenden Dropdowns der Monat und das Jahr ausgewählt werden. Mit Klick auf den Button „Aktualisieren“ werden die Statistikdaten des ausgewählten Monats angezeigt.

Einstellungen / Email Notifications

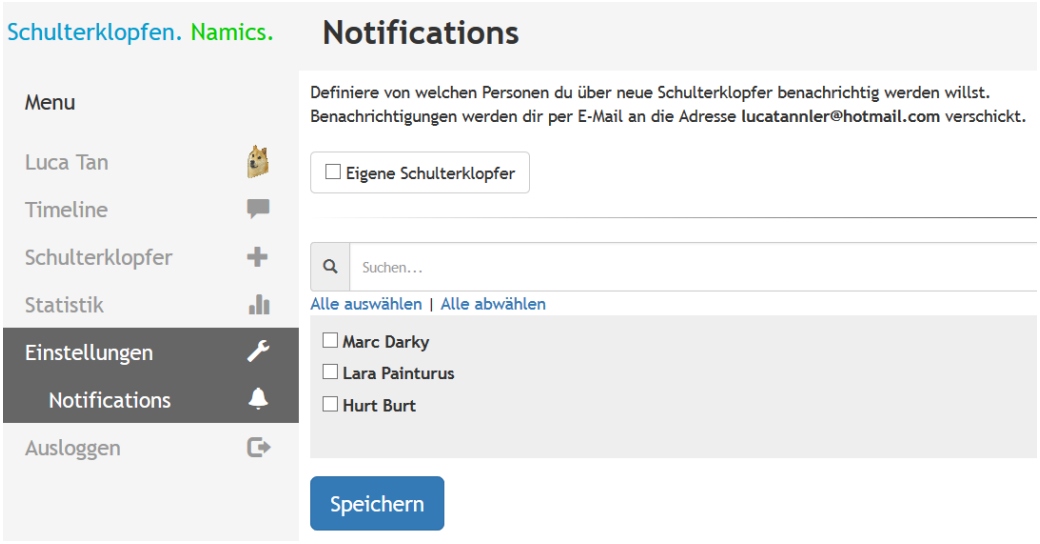


Abbildung 41 Externes Design: Einstellungen / Email Notifications

User können nach Bedarf Emailnotifications erhalten wenn jemand ein Schulterklopfer erhält. Auf obig angezeigter Seite kann der User definieren bei welchen Personen er notifiziert werden möchte.

Responsive Design

Die WebApp wurde responsive designed damit die Darstellung sowohl auf einem PC mit einem grossen Bildschirm wie auch auf einem Smartphone übersichtlich angezeigt wird.

Folgend ein Beispiel der Seite Timeline in der Ansicht auf einem grossen Bildschirm (1. Screenshot) und die Ansicht auf einem Smartphone (2. Screenshot).

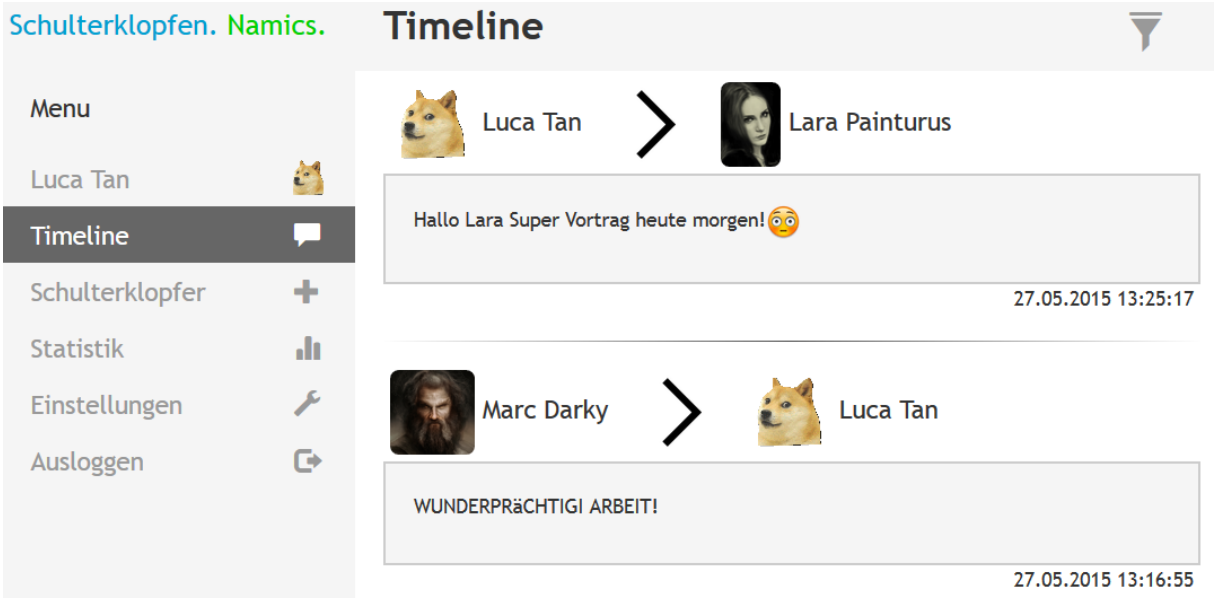


Abbildung 42 Externes Design: Responsive Design 1

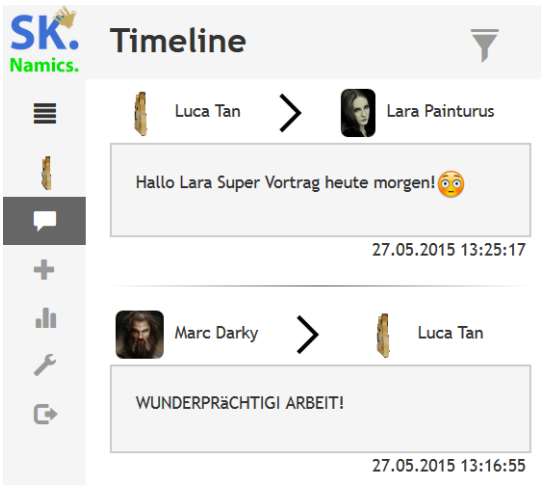


Abbildung 43 Externes Design: Responsive Design 2

Glossar

Spring	Spring ist ein Java Framework welches viele Funktionalitäten bieten um Web-Applikation (oder auch Standalone) zu erstellen.
Bootstrap	Bootstrap ist ein Frontend Framework für die Entwicklung von WebApps. Es bietet viele Komponenten für das Web wie auch einfach Variante, um eine Website Responsive umzusetzen
Responsive Design	Mit Responsive Design meint man (z.B.) Webseiten, welche Ihr Design automatisch an die Grösse des Bildschirmes anpassen.
RUP	Rational Unified Process. Dies ist ein Vorgehensmodell für die Softwareentwicklung.
JQuery	Dies ist ein JavaScript Framework welches viele Funktionalitäten bietet und einem viel Arbeit abnehmen kann.
OAuth2	OAuth2 ist ein offenes Protokoll welches eine sichere und standardisierte API-Autorisierung sowohl für Desktop-, Web- als auch Mobile-Applikationen erlaubt.
Redmine	Redmine ist ein Projektverwaltungs Tool in welchem man Arbeitspakete, Arbeitszeiten und weitere Daten erfassen kann.
UI	UI steht für User Interface ➔ Benutzeroberflächen

Literaturverzeichnis

Thema	**Beschreibung / Link**
OAuth2	Herstellerseite: http://oauth.net/2/ Allgemeiner Artikel: http://de.wikipedia.org/wiki/OAuth
Spring Framework	Einführung in Spring Framework http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html
Bootstrap	Artikel über Bootstrap http://de.wikipedia.org/wiki/Bootstrap_%28Framework%29

Quellenangaben

	Quelle
Spring Framework	http://www.spring.io
Bootstrap	http://getbootstrap.com/
JQuery	http://getbootstrap.com/
Google OAuth2 Login für Spring	https://github.com/skate056/spring-security-oauth2-google

Wichtige Informationen

Ich habe auf diversen Seiten im Internet (z.B. Foren und Blogs), hauptsächlich Stackoverflow, nach Lösungen zu Problemstellungen gesucht. Teile die 1-1 kopiert wurden, sind im Quellcode ausdrücklich mit Kommentaren versehen und einem Verweis auf die Quelle.

Die meisten verwendeten Icons im Frontend stammen aus dem Bootstrap Framework.
Alle anderen Medien welche nicht selber erstellt wurden, stammen von http://www.iconfinder.com/ dabei handelt es sich ausschliesslich um Bilder, welche verwendet werden dürfen.

Abbildungsverzeichnis

Anhänge

Projektplan

Änderungsgeschichte

Datum	Version	Änderung	Autor
02.03.2015	1.00	Initial	Luca Tännler
08.03.2015	1.01	Risk & Qualitätsmanagement aktualisiert	Luca Tännler
17.03.2015	1.02	Korrekturen & Anpassungen	Luca Tännler
18.03.2015	1.03	Korrekturen & Anpassungen & Ergänzungen	Luca Tännler

Projekt Übersicht

Zweck

Dieses Dokument beschreibt die Planung für die Vorbereitung und Entwicklung des Projekts „Schulterklopfen WebAPP“ der Firma Namics.

Gültigkeitsbereich

Studienarbeit im Frühlingssemester 2015 an der HSR.

Lieferumfang

- Projektplan
- Iterationsplanungsdokumente
- Analyse: Bootstrap vs. Polymer
- Analyse: Spring OAuth: Google Login
- Front- und Backend der WebAPP (muss nicht vollständig sein)
- Quellcode
- Zeiterfassung
- Installationsanleitung
- WAR-File mit Webanwendung
- DB-Script
- Maven Build Scripts
- Anforderungsspezifikation (+UC’s, Domainmodell)
- Architekturdokument

Projektorganisation

Organisationsstruktur

Das Projekt ist eine **Einzelarbeit** von **Luca Tännler** und übernimmt alle Aufgaben wie Projektleitung, Entwicklung etc.

Projektbetreuer: Hans Rudin
Auftraggeber: Beat Helfenberger / Namics AG

Kommunikationswege

- **Meetings**
- **Email**

- **Telefon**

Management Abläufe

Kostenvoranschlag

Die Dauer des Projekts beträgt ca. 14 Wochen mit einem Arbeitsaufwand von insgesamt mindestens 240h.

Zeitliche Planung

Die Planung und Verwaltung der Arbeit erfolgt in Redmine.
Diese wird nach jeder Iteration(Sprint) aktualisiert.

Da die Umsetzung der WebApp agil entwickelt wird, werden die Iterationen während der Entwicklung zu Beginn jeder Iteration/Sprint geplant nach Situation und Bedarf.

Phasen / Iteration

Das Projekt wird mit einer Mischung aus RUP und Scrum umgesetzt.

Zu Beginn wird es eine Inception und Elaboration Phase geben. Die Entwicklung der WebApp erfolgt dann mit Iterationen (Sprints) nach der SCRUM Methodik. Die Iterationen werden durchnummeriert und in Redmine verwaltet. Am Ende jeder Iteration wird ein Iterationsassessment abgehalten.

*Siehe Dokument **Iterationsplanung.pdf** für eine genauere Beschreibung.*

Meilensteine

Datum	Meilenstein	Beschreibung
27.02.2015	Kickoff Meeting	Kickoff Meeting in St. Gallen bei der Namics AG.
23.03.2015	Meeting Namics	Meeting bei Namics. Vorstellen der 2 geforderten Analysen und Besprechung der Projektplanung.
27.03.2015	End of Elaboration	Fertigstellung der Analysen & Projektplanung.
27.03.2015 – 26.05.2015	Iterationen (Mehrere)*	Mehrere Iterationen für die Entwicklung der WebApp. Ein Milestone pro Iteration.
26.05.2015	Kurzfassung & A0 Poster	Erarbeitung einer Kurzfassung & A0 Poster für Projektabgabe.
05.06.2015	Abgabe	Abgabe des gesamten Projekts.

*Die Iterationen werden jeweils nach Ende jeder Phase / Iteration geplant. Da nach einer agilen Entwicklungsmethode vorgegangen wird, können zum jetzigen Zeitpunkt noch keine festen Iterationen geplant werden.

Reviews / Besprechungen

Während der Projektzeit werden mehrere Reviews / Besprechungen je nach Situation und Bedarf mit dem Projektbetreuer und/oder Arbeitgeber abgehalten.

Zu jedem Review / Besprechung wird ein Protokoll geschrieben und auf Redmine als PDF abgelegt. Das Protokoll soll Auskunft über die Traktanden, besprochenes, abgemachtes sowie kommende Termine geben.

Risikomanagement

Risiken

Die Risiken sind im Dokument **TechnischeRisiken.xlsx** zu finden.

Umgang mit Risiken

Ziel ist es die Risiken möglichst gering zu halten.
Dennoch gibt es Risiken die auftreten können. Um Probleme/Verzögerungen die auftreten können zu vermeiden wird mit Sorgfalt gearbeitet und versucht im Falle eines Problems den „Schaden“ und Behebungsaufwand möglichst gering zu behalten.

Am Ende jeder Iteration/Phase wird eine Überprüfung durchgeführt, ob die Arbeit korrekt umgesetzt wurde und ob Fehler aufgetreten sind. Im Falle von Fehlern wird versucht, diese schnellst möglich zu beheben.

Werkzeuge / Infrastruktur

Zur Entwicklung, Building & Testen wird der persönliche Laptop/PC verwendet.

Übersicht der Tools

GIT

Die Entwicklung wird mit GIT versioniert. Der Arbeitgeber Namics AG stellt einen GIT Server zur Verfügung.

Redmine

Projektplanungstool. In diesem Tool wird die gesamte Arbeit geplant. Dieses Tool verfügt auch über ein internes Wiki. Neue Iterationen werden jeweils nach Iterationsabschluss geplant und erfasst.

In Redmine wird folgendes Verwaltet:

- Arbeitszeit
- Milestones
- Roadmap
- Arbeitspakete / Iterationen
- Zeitplan
- Fehler / Bugs
- Reviews / Meetings

Eclipse

Entwicklungsumgebung.

Astah

Programm für die Erstellung von UML Diagrammen.

Tomcat

Java Application Server

Apache Maven

Build-Management-Tool.

MySQL

Das zu verwendende Datenbanksystem.

Frameworks

Für die Entwicklung wird die Verwendung von folgenden Frameworks Vorausgesetzt:

- Verwendung vom Java Spring Framework
- Verwendung von Bootstrap oder Polymer Framework

Dokumentation

Dokumentationen werden hauptsächlich in Word-Dokumenten erarbeitet und final als PDF-Dokument exportiert.

Qualitätsicherung

Dokumentation

Dokumente

Die Dokumente haben alle dieselbe Struktur:

- Titelseite mit
  - Titel
  - Autor
  - Projektname
  - evtl. weitere Infos
- Kopfzeile mit Dokument Titel und Autor
- Seitenzahlen unten rechts
- Änderungsverlauf (bei grösseren Dokumenten)
- Inhaltsverzeichnis (bei grösseren Dokumenten)

Iterationen

Iterationen werden Dokumentiert. Es wird jeweils beschrieben, was in der jeweiligen Iteration gemacht wird und die Resultate (+ Iterationsasssessment).

Die Iterationen werden jeweils vor Beginn geplant und dokumentiert.

Meetings

Zu jedem Meeting wird ein Protokoll geführt in welchem ersichtlich ist

- was besprochen wird
- Resultate / Entscheidungen
- Neue Termine
- Und evtl. weiteres

Projektmanagement

Zeiterfassung

Die Arbeitszeit wird fortlaufend in Redmine erfasst und auf aktuellem Stand gehalten.
Die Zeiten werden jeweils dem entsprechenden Arbeitspaket zugewiesen.

Iterationen

Die Iterationen werden jeweils nach einer beendeten Iteration geplant. Eine Iteration wird dann in Redmine erfasst und Arbeitspakete erstellt.

Nach jeder Iteration werden Reviews gehalten (Ausgeschlossen Spezialsituationen bei denen kein Review nötig ist.)

Entwicklung

Code Reviews

Code Reviews werden jeweils am Ende jeder Iteration durchgeführt (wenn nötig).

Codestatistik

Kategorie	**Wert**
Anzahl Klassen	30
Anzahl Attribute	89
Anzahl Interfaces	2
Total Lines of Code	1997
Anzahl Methoden	153
Anzahl Packages	9

Eclipse Metrics Auswertung

Folgende Auswertung bezieht sich auf das gesamte Projekt.

Metric	Total	Mean	Std. Dev.	Maxim...
Number of Parameters (avg/max per method)		0.885	0.974	4
Number of Static Attributes (avg/max per type)	3	0.1	0.396	2
Efferent Coupling (avg/max per packageFragment)		3.111	2.378	7
Specialization Index (avg/max per type)		0.157	0.422	2
Number of Classes (avg/max per packageFragment)	30	3.333	2.055	7
Number of Attributes (avg/max per type)	89	2.967	2.702	9
Abstractness (avg/max per packageFragment)		0.066	0.097	0.25
Normalized Distance (avg/max per packageFragment)		0.256	0.256	0.7
Number of Static Methods (avg/max per type)	3	0.1	0.3	1
Number of Interfaces (avg/max per packageFragment)	2	0.222	0.416	1
Total Lines of Code	1997			
Weighted methods per Class (avg/max per type)	299	9.967	13.255	73
Number of Methods (avg/max per type)	153	5.1	5.002	20
Depth of Inheritance Tree (avg/max per type)		1.267	0.442	2
Number of Packages	9			
Instability (avg/max per packageFragment)		0.765	0.312	1
McCabe Cyclomatic Complexity (avg/max per method)		1.917	2.722	23
Nested Block Depth (avg/max per method)		1.378	0.728	4
Lack of Cohesion of Methods (avg/max per type)		0.384	0.409	1
Method Lines of Code (avg/max per method)	903	5.788	9.755	70
Number of Overridden Methods (avg/max per type)	11	0.367	1.14	6
Afferent Coupling (avg/max per packageFragment)		2.222	3.765	11
Number of Children (avg/max per type)	2	0.067	0.249	1

Abbildung 44 Eclipse Metrics Auswertung (Projekt)