Pflichtenheft

Gruppe 02

13. März 2015

Inhaltsverzeichnis

1 Einführung

1.1 Überblick

Dieses Dokument stellt den Systementwurf des EnergyChallenge-Systems dar. Im Folgenden werden diese Themen behandelt:

- Kurze Einleitung in die verwendeten Technologien
- Beschreibung der beteiligten Komponenten und deren Verteilung auf Hardware
- Strukturierung des Systems in einzelne Pakete
- Aufbau und Beschreibung der in den Paketen enthaltenen Klassen
- Beschreibung des Verhaltens von einzelnen Anwendungsfällen mit Hilfe von Sequenzdiagrammen

1.2 Technologien

Das in diesem Dokument beschriebene System verwendet die folgenden Technologien: Android Android ist ein auf mobilen Geräten lauffähiges Betriebssystem sowie eine Software-Plattform für z.B. Smartphones und Tablets , welches auf einer stark abgeänderten Version von Linux basiert (es wird ein Linux-Kernel verwendet). Apps, also Android Applikationen, laufen in der Android Runtime, einer Android-eigenen Laufzeitumgebung (ab Android 5.0).

- http://www.android.com

Java Java ist die Programmiersprache, in der dieses Projekt größtenteils geschrieben ist. Dabei kommt Java meistens nicht in seiner „natürlichen“ Form vor, sondern wird z.B. durch Android Klassen ergänzt.

- http://java.oracle.com

GRAILS GRAILS ist ein auf der Programmiersprache Groovy aufbauendes Web Application Framework, welches mit relativ geringem Aufwand die Möglichkeit bietet, Webseiten mit Datenbank Anbindung zu erstellen. Mit GRAILS werden auch sogenannte Groovy Server Pages (GSP) erzeugt, welche Inhalte in sonst statische Webseiten einbinden können.

- http://grails.org

GORM GORM oder auch GRAILS Object-Relational Mapping ist GRAILS' integriertes ORM-Tool welches Hibernate 4 zum persistieren von Daten in einer Datenbank verwendet.

- http://grails.org/doc/2.3.x/guide/GORM.html
- http://www.hibernate.org

Apache Shiro Apache Shiro ist ein Java Security Framework, welches zur Autorisierung und Authentifizierung von Benutzern auf Webseiten verwendet wird. Es wird das Apache Shiro Plugin für Grails verwendet, welches viele Vorgänge automatisiert und vereinfacht.

- http://shiro.apache.org
- http://grails.org/plugin/shiro

2 System Übersicht

2.1 Komponentendiagramm

Server Diese Komponente ist das Hauptprogramm, das auf dem physikalischen Server läuft. Hier findet die Datenhaltung, -bereitstellung und die Generierung der Website statt.

Model Die Modellierung sämtlicher Daten. Dazu gehört das Speichern und der Zugriff auf diese.

Profile Management Die Modellierung der Profildaten, also der Daten für Benutzer und Teams. Dazu gehört das Speichern und der Zugriff auf diese.

User Management Die Modellierung der Benutzerdaten. Dazu gehört das Speichern und der Zugriff auf diese.

Team Management Die Modellierung der Teamdaten. Dazu gehört das Speichern und der Zugriff auf diese.

Activity Management Die Modellierung der Daten der Aktivitäten. Dazu gehört das Speichern und der Zugriff auf diese.

Proposal Management Die Modellierung der Energiesparvorschläge, der Aktivitäten. Dazu gehört das Speichern und der Zugriff auf diese.

Stats Die Modellierung der Statistiken bezüglich der Aktivitäten und Besucherzahlen. Dazu gehört das Speichern und der Zugriff auf diese.

Website Präsentiert die Daten des Models und erlaubt Interaktionen.

Controller (Website) Reagiert auf Benutzereingaben beziehungsweise -anfragen und kümmert sich darum, die benötigten Daten vom Model zu erhalten.

View (Website) Präsentiert die Daten des Models.

App Controller Zuständig für die Interaktion der App mit dem Server. Auf eine Anfrage der App werden Daten aus dem Model mithilfe von JSON für die App bereitgestellt.

Android App Die native Android App, die in erster Linie zur Präsentation von Daten, die auf dem Server liegen, dient.

View (Android App) Präsentiert die Daten innerhalb der App.

Controller (Android App) Reagiert auf Benutzereingaben beziehungsweise -anfragen und kümmert sich darum die geforderte Aktion an den Server oder das App-Model weiterzuleiten.

Model (Android App) Kümmert sich um die Datenhaltung innerhalb der App. Das ist zuerst nur die Information darüber, welcher Benutzer angemeldet ist, kann jedoch später noch um Cacheing-Funktionen erweitert werden, um nicht immer alle Daten vom Server holen zu müssen.

2.2 Verteilungsdiagramm

2.2.1 Android-Gerät

Das Android-Gerät ist das Arbeitswerkzeug für die Benutzer. Auf dem Gerät ist eine native Android-Applikation installiert, mit der Daten eingegeben und abgerufen werden können. Der Benutzer muss bereits registriert sein, um die Applikation nutzen zu können.

2.2.2 Computer

Benutzer können mit Hilfe eines Computers über einen Webbrowser auf eine Webseite zugreifen, welche, nach erfolgreichem einloggen, verschiedene Funktionen zur Aktivitäts- und Profilverwaltung anbietet.Des Weiteren können hier auch anonymisierte Statistiken angezeigt und diese statistischen Daten exportiert werden.

2.2.3 Server

Der Server stellt die eben erwähnte Webseite bereit. Hier werden gegebenenfalls Datenanfragen von den Android-Geräten verarbeitet.

2.3 Paketdiagramm

3 Ordner Struktur

4 Klassen

4.1 Webseite

4.2 Android App

Die Klassen in *Activities*, *Adapters* sowie die Klasse *AccessServerTask* erweitern vordefinierte Android-Klassen. Hierdurch werden viele Methoden definiert, die die vordefinierten Methoden überschreiben, wie zum Beispiel *onCreate* oder *onDestroyView*. Diese werden im Klassendiagramm und der folgenden Beschreibung nicht explizit aufgeführt.

4.2.1 Activities

MainActivity Die *Activity*, die für die Darstellung der Hauptfunktionen zuständig ist. Das bedeutet in ihr wird die Navigation aufgerufen und die meisten Funktionen, die als *Fragment* dargestellt werden.

LoginActivity Die *Activity* über die, die Anmeldung erfolgt.

SearchActivity Dient zur Ausführung einer Suche und der Anzeige der Ergebnisse.

ProposalActivity Dient zur Anzeige eines Energiesparvorschlags inklusive seiner Bewertungen und Kommentare. Ermöglicht außerdem das Bewerten und Kommentieren von Energiesparvorschlags.

TeamProfilActivity Zeigt das Profil eines Benutzers an.

UserProfilActivity Zeigt das Profil eines Teams an.

MainFragment Darstellung des Haupt-*Fragments*.

MyProfilFragment Zeigt das eigene Profil an.

RankinglistFragment Zeigt das *Fragment* für die Benutzer- bzw. Teamrangliste an.

TeamRankingsTabFragment Zeigt die aktuelle Team-Rangliste an.

UserRankingsTabFragment Zeigt die aktuelle Benutzer-Rangliste an.

ProposalFragment Zeigt alle Energiesparvorschläge an.

NavigationFragment Die Navigation innerhalb der Haupt-*Activity*.

4.2.2 Adapters

FavoriteActivitiesAdapter Zuständig für die Anzeige der favorisierten Aktivitäten. Von hier aus sollen auch Aktivitäten ausgeführt werden können, wenn sie angeklickt werden.

ActivitiesAdapter Zuständig für die Anzeige aller Aktivitäten. Bei einem Klick auf eine Aktivität wird diese ausgeführt.

UserRankingsAdapter Zuständig für die Anzeige der Benutzer-Rangliste. Bei einem Klick auf einen Benutzer soll eine *UserProfilActivity* erstellt werden, in der der angeklickte Benutzer ausgeben wird.

TeamRankingsAdapter Zuständig für die Anzeige der Team-Rangliste. Bei einem Klick auf ein Team soll eine *TeamProfilActivity* erstellt werden, in der das angeklickte Team ausgeben wird.

ProposalsAdapter Zuständig für die Anzeige der Energiesparvorschläge. Bei einem Klick auf einen Vorschlag soll eine *ProposalActivity* erstellt werden, in der das angeklickte Team ausgeben wird.

4.2.3 Tasks

AccessServerTask Eine Klasse, die eine abstrakter Serveranfrage darstellt. Alle Serveranfragen-Klassen erben von dieser Klasse und implementieren die abstrakten Methoden *createServerRequest()* und *handleServerResponse()* in welchen eine Server-Anfrage gestellt wird und eine Server-Rückgabe verarbeitet wird. Die Klasse kümmert sich um den Verbindungsaufbau, die Antwort und das Parsen der Daten, so dass die abgeleiteten Methoden sich nicht mehr darum kümmern müssen.

GetTeamProfileTask Die Klasse, die dafür zuständig ist, ein Teamprofil vom Server zu laden und in der *TeamProfilActivity* anzuzeigen.

GetUserProfileTask Die Klasse, die dafür zuständig ist, ein Benutzerprofil vom Server zu laden und in der *UserProfilActivity* bzw. im *MyProfilFragment* anzuzeigen.

GetTeamRakingTask Die Klasse, die dafür zuständig ist, das Team-Ranking vom Server zu holen und an den *TeamRankingsAdapter* weiterzureichen.

GetUserRakingTask Die Klasse, die dafür zuständig ist, das Benutzer-Ranking vom Server zu holen und an den *TeamRankingsAdapter* weiterzureichen.

GetActivitiesTask Die Klasse, die dafür zuständig ist, die Liste aller Aktivitäten vom Server zu holen und an den *ActivitiesAdapter* weiterzureichen.

GetFavoriteActivitiesTask Die Klasse, die dafür zuständig ist, die Liste der favorisierten Aktivitäten vom Server zu holen und im *MainFragment* anzuzeigen.

GetProposalsTask Die Klasse, die dafür zuständig ist, die Energiesparvorschläge vom Server zu holen und an den *ProposalsAdapter* weiterzureichen.

GetProposalTask Die Klasse, die dafür zuständig ist, einen konkreten Energiesparvorschlag vom Server zu laden und in der *ProposalActivity* anzuzeigen.

SearchTask Die Klasse, die dafür zuständig ist, eine Suchanfrage an den Server zu stellen und das Ergebnis in der *SearchActivity* anzuzeigen.

DoActivityTask Die Klasse, die dafür zuständig ist, dem Server mitzuteilen, dass ein Benutzer eine Aktivität ausgeführt hat.

CommentProposalTask Die Klasse, die dafür zuständig ist, dem Server mitzuteilen, dass ein Benutzer einen Energiesparvorschlag bewertet oder kommentiert hat.

4.2.4 Utils

IoX Klasse, die Methoden für generelles Input/Output bereitstellt. Das ist zum aktuellen Zeitpunkt nur die Methode *readInputStream*, die einen *InputStream* in einen String umwandelt.

ServerRequest Klasse, die eine Serveranfrage modelliert. Eine Serveranfrage besteht aus einem Adressaten auf dem Server und aus der eigentlichen Anfrage. Die eigentliche Anfrage ist ein JSON-Objekt.

5 Dynamisches Verhallten

5.1 Login/Registrierung

5.2 Aktivität erledigen

5.3 Rangliste ansehen

5.4 Team erstellen

5.5 Team blocken

5.6 Vorschlag erstellen

5.7 Vorschlag bewerten

5.8 Statistiken exportieren

5.9 Vorschlag in Aktivität umwandeln

5.10 App: Benutzerrangliste ansehen

Die Serveranfragen in der App funktionieren alle nach dem selben Muster. Benutzerrangliste ansehen steht hier exemplarisch für weitere Anwendungsfälle dieser Art. Wenn der Nutzer in der App zur Benutzerrangliste navigiert, wird diese automatisch bei der Erstellung des *Fragments* geladen. Dabei wird auf ein *GetUserRankingTask* die Methode *execute()* aufgerufen. *GetUserRankingTask* erbt von *AccessServerTask*, die wiederum von der Androidklassse *AsyncTask* erbt. In dem *AccessServerTask* wird die Methode *doInBackground()* aufgerufen, die wiederum die Methode *createServerRequest* aufruft, die im *GetUserRanking-Task* definiert ist.
Sobald die *doInBackground()*-Methode fertig ausgeführt wurde, wird (vom *Async-Task*) die Methode *onPostExecute()* aufgerufen, die *handleServerResponse()* des *GetUserRankingTask* ausführt. Hier wird dann die Darstellung im *Fragment* getätigt.
Der Vorteil dieser Art der Implementierung ist, dass nur einmal eine generelle Serverabfrage definiert werden muss, und sich keine weitere Gedanken um dessen Implementierung gemacht werden muss. Deswegen ist die Durchführung der HTTP-Abfrage hier auch nicht weiter aufgeführt.

6 Anhang

6.1 Klassen Diagramme