22.06.2017

Projekt „Trawell“

Projektdokumentation

Projekt Geoinformatik SS2017

Prof. Dr. Thomas Brinkhoff

Lennard Gabriel, Lisa Haltermann,

Maximilian Herbers, Nils Kirsch, David Post

Inhalt

[1. Aufgabenstellung und Projektorganisation 2](#_Toc485942388)

[2. Studie und Anforderungsanalyse 2](#_Toc485942389)

[3. Systementwurf 2](#_Toc485942390)

[4. Implementierung 3](#_Toc485942391)

[5. Systemintegration 4](#_Toc485942392)

[6. Ausblick und Zusammenfassung 4](#_Toc485942393)

[ANHANG 7](#_Toc485942394)

# Aufgabenstellung

Im Rahmen der Forschungsinitiative mFund des Bundesministeriums für Verkehr und digitale Infrastruktur werden Forschungs- und Entwicklungsprojekte rund um digitale datenbasierte Anwendungen für die Mobilität 4.0 gefördert (BMVI). Hierdurch sollen Geschäftsideen, die auf Mobilitäts-, Geo- und Wetterdaten basieren unterstützt werden. Das mFund diente als Anregung für das Projekt Geoinformatik im Sommersemester 2017.

Im Zuge des Projektes sollte eine Android-Applikation im Zusammenhang mit Bahnreisen innerhalb Europas entstehen. Ziel war es, dass die entwickelte App Anwendern ermöglicht eine Reise durch europäische Städte auf Grundlage des internationalen Bahn-Tickets Interrail-Passes zu planen und einen hilfreichen digitalen Reisebegleiter darstellt.

Zur Versionierungsverwaltung und als Filehoster wurde der webbasierte Online-Dienst GitHub mit den entsprechenden Desktop-Userinterfaces, u.a. GitDesktop, verwendet. Hierdurch konnte eine für alle Parteien nachvollziehbare Aufgabenverteilung und der zeitliche Rahmen des Projektes, u.a. durch das Anlegen von Meilensteinen, abgesteckt werden.

# Studie und Anforderungsanalyse

Die zu entwickelnde Anwendung soll besonders von Reiseinteressierten und Besitzern eines Interrailpasses genutzt werden. Unter diesem Gesichtspunkt ist vor allem auf eine intuitive benutzerfreundliche Bedienung zu achten. Der Nutzer sollte in wenigen einfachen Schritten die von ihm gewünschten Aktionen durchführen können.

Zu den von der App bereitgestellten Funktionen zählen die Auswahl der zu bereisenden Städte in einer Karte und die Auswahl einer Unterkunft in diesen Städten. Des Weiteren soll der Abruf einer „Alles-auf-einen-Blick“-Anzeige mit Informationen zum aktuellen Ziel der Reise, wie beispielsweise die aktuelle Wetterlage oder der Zeitpunkt der nächsten Abfahrt, möglich sein.

Die Route und dementsprechend die Zugverbindungen sollen automatisiert über einen Algorithmus bestimmt werden. Abschließend soll eine fertig geplante Reise über soziale Medien geteilt werden können.

Die Zielbestimmungen und Funktionen wurden in einer frühen Fassung in einem Lastenheft festgehalten, welches der Dokumentation als Anhang beiliegt (ANHANG I).

# Systemarchitektur

Die Allgemeine Systemarchitektur besteht aus zwei Teilen. Aus dem Frontend und dem Backend. Im Frontend entstand der View der Android-Applikation. In diesem wird jede Seite generiert und implementiert, sprich mit den jeweiligen Methoden hinterlegt (Anhang II).

Das Backend beschreibt die Logik und die einzelnen Klassen der App. Hier wird die Datenspeicherung und die Datenbeschaffung implementiert (Anhang III).

Die Datenbeschaffung betrifft vor allem die Api’s. Dabei wurden mehrere Api’s für unterschiedlichste Zwecke benötigt. Für die Unterkünfte also die Accommodations wurde eine GooglePlaces Api verwendet. Diese liefert alle benötigten Informationen zu einem Hotel in einer Stadt, gefiltert mit einem Radius. Diese Api ist allerdings auf 1.000 Zugriffe pro Tag beschränkt, was in diesem Fall aber kein Problem darstellt. Für die Wetterdaten wurde eine Openweather Api verwendet. Diese liefert die Temperatur und die Luftfeuchtigkeit einer Stadt zurück. Allerdings ist auch diese auf Aufrufe pro Minute begrenzt. Ein weitere Api ist die GoogleMaps Api, welche für eine Hintergrunddarstellung benutz wird. Auf dieser Grundlage ist es möglich Punkte einzuzeichnen und wieder herauszunehmen. Auch diese ist auf 1.000 Aufrufe pro Tag beschränkt. Ein anderer Bestandteil des Architekturschemas sind die GTFS Feeds. Diese wurden in einer Textdatei zusammengefasst und in eine SQL Lite Datenbake überführt. Das daraus entstehende Graph-Modell wird später noch unter dem Punkt 4.1 Datengrundlage und Datenspeicherung sowie unter dem Punkt 4.2 Routenfindung erläutert. Die gespeicherten Touren kann man über, auf dem Telefon installierte Apps, geteilt werden. Dafür muss aber eine App wie Facebook installiert werden, also eine App die diese Schnittstelle bedient. Ein Schema der Api‘s ist im Anhang abgelegt (Anhang VI).

Sind in der Datei ProjektInformatikClassDiagramm.vpp

# Implementierung

Umsetzung der Programmierung – als Unterpunkte die einzelnen Themen. Kurze Erläuterung des Vorgehens und der Gründe für die Entscheidungen die zu diesem Vorgehen geführt haben.

Die Umsetzung der geplanten Systemarchitektur startete mit der Implementierung der Zwischenschicht. Zunächst wurde die Spiel- und Anzeigelogik, wie zum Beispiel Spielfeld und Gewinnprüfung, programmiert. Zum Testen dieser war das Erstellen eines einfachen Computergegners nützlich. Es wurden Interfaces als Schnittstellen zu den Frontends geschaffen. Die Modelle für den Nutzer oder einen Spielzug konnten sowohl für die Frontends als auch für die Serverumgebung verwendet werden. Den Kern der Zwischenschicht bildet die Singletonklasse Connect4. Die Zwischenschicht wurde bis zum Abschluss des Projektes immer wieder unterschiedlich umfangreich angepasst oder erweitert.

Nachdem eine erste Version der Zwischenschicht funktionsfähig war, wurde die Android Anwendung geschrieben. Hier ging es vorrangig darum, ein reibungsloses einbinden der Interfaces der Zwischenschicht zu gewährleisten und die Darstellung des Spielfeldes zu implementieren. Beim Aufbau der Benutzeroberfläche wurde sich möglichst an den Entwurf gehalten.

In der Serverkomponente wurden zunächst die Servlets erstellt. Diese stellen die Server-Datenbank-Kommunikation dar, welche über Ebean Object-Relational-Mapping umgesetzt wurde. Hierfür wurden die Datenbanktabellen als Java-Klassen dargestellt. Dies ermöglichte einen einfachen Zugriff für alle nötigen Operationen auf diese. Diese „Datenbanktabellenklassen“ wurden auch in die Zwischenschicht eingefügt um die Antworten des Servers möglichst komfortabel verarbeiten zu können. Zusätzlich wurden Service-Klassen erstellt, die den Aufruf der Servlets von der Zwischenschicht aus steuern.

Zur Authentifizierung der Client-Anfragen wurde das Java-Web-Token genutzt. Dieses wird bei erfolgreichem Login eines Clients automatisch im Header des Server Response mitgeschickt und enthält die nötigen Informationen zur Identifizierung des anfragenden Nutzers. Ein Mitschicken der teils vertraulichen Daten über eine Query ist somit nicht nötig.

Mit der Implementierung der Sockets, wurde die Kommunikation zwischen zwei Clients über den Server umgesetzt. Eine Payload-Klasse wird mit den zu übermittelnden Daten bestückt und über den Serversocket an den Gamesocket des Clients übermittelt. Hier wird die Payload wieder entladen und die übertragene Nachricht oder der Spielzug werden dem Client übergeben. Im Frontend wird dies als SocketListener-Interface eingebunden.

Abschließend wurde die Web-Anwendung programmiert. Hier wurden die im Android-Frontend bereits umgesetzten Implementierungen größtenteils übernommen. Dies führte jedoch teilweise zu eher schwer nachvollziehbarem oder nicht funktionierendem Code. Eine ausführliche Überarbeitung war und ist noch von Nöten.

Einen besseren Eindruck zur Implementierung aller Komponenten und Module vermittelt jedoch der Quellcode selbst. Die Dokumentation des Programmcodes wurde teils während der Implementierung teils nachträglich durchgeführt.

## Datengrundlage und Datenspeicherung

## Routenfindung

## Systemintegration

Nötig?

Die Systemintegration erfolgte wie aus dem Teil Implementierung hervorgeht schrittweise. Einzelne Komponenten wurden entwickelt, ihre Funktionsfähigkeit teilweise mit JUnit-Tests geprüft und mit anderen funktionierenden Komponenten verbunden. Ein guter Indikator hierfür ist die Zwischenschicht, welche nach und nach um entsprechende Module erweitert wurde.

# Ausblick und Zusammenfassung

Aktuell nur Auswahl an europäischen Städten und den Strecken zwischen diesen

Explizite Auswahl von gewünschten Zugstrecken in Bezug auf Abfahrtszeit

Auswahl und speichern von POI`s der jeweiligen Stadt auf einer Karte

Bestellung des Interrail Tickets direkt in der App

Buchungsanfragen an Unterkunftsanbieter direkt in der App

Unsere Wünsche an die APIS

Apis allgemeiner fürs sharing ohne die anderen Apps installiert haben zu müssen

Sharing schöner und komfortabler individueller

Einiger der im Lastenheft als Wunschfunktionen vereinbarten Funktionen sind zum aktuellen Zeitpunkt noch nicht implementiert. Darunter fallen das Blockieren und Melden anderer Spieler, Unterschiedliche Nutzerkonten mit erweiterten oder eingeschränkten Rechten, das Anpassen der eigenen Repräsentation im Spiel und das flexible, serverseitige Austauschen der Werbeflächen.

Bei der Umsetzung der Web-Anwendung konnte das volle Potential nicht erreicht werden, dies wird vor allem im Vergleich mit der Android-Applikation deutlich. Trotzdem ist das Ergebnis ein solide funktionierendes System mit zwei Frontend-Anwendungen und einem Servergestützten „Vier gewinnt“-Spiel.

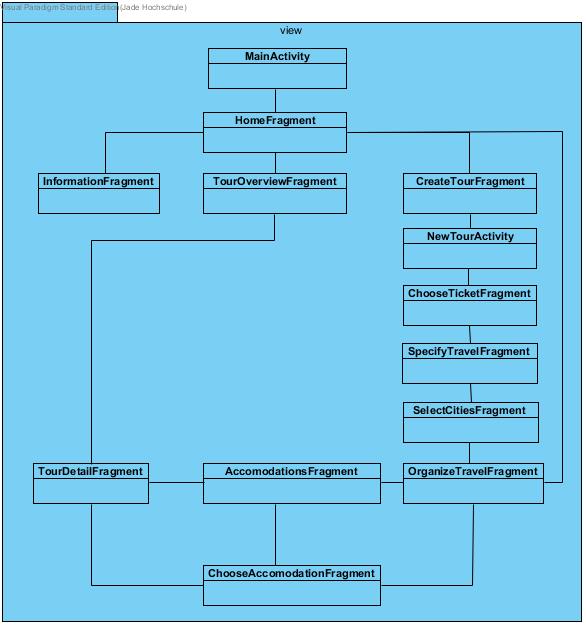
# Quellen

Bundesministerium für Verkehr und digitale Infrastruktur – Webseite: <http://www.bmvi.de/DE/Themen/Digitales/mFund/Foerderung/foerderung.html>, zuletzt abgerufen am 06.07.2017

# ANHANG

## Anhang I Lastenheft

## Anhang II Frontend



## Anhang III BackendEin Bild, das Text enthält. Mit sehr hoher Zuverlässigkeit generierte Beschreibung

## Anhang IV API‘s



Ausgewählte GTFS Dateien (\*.txt) zusammenführen und ins

relationale Modell übertragen

Wetterdaten

Dijkstra

POIs

Unterkünfte

Tour teilen

Kartendarstellung

und eigene Features anzeigen

Datenhaltung und Tourspeicherung



## Anhang V

## Anhang VI

## Anhang VII

## Anhang VIII

## Anhang IX

## Anhang X

## Anhang XI

## Anhang XII