22.06.2017

Projekt „Trawell“

Projektdokumentation

Projekt Geoinformatik SS2017

Prof. Dr. Thomas Brinkhoff

Lennard Gabriel, Lisa Haltermann,

Maximilian Herbers, Nils Kirsch, David Post

Inhalt

[2 Aufgabenstellung 1](#_Toc487486990)

[3 Studie und Anforderungsanalyse 2](#_Toc487486991)

[4 Systemüberblick 3](#_Toc487486992)

[5 Implementierung 3](#_Toc487486993)

[5.1 Entwurf des Layouts 4](#_Toc487486994)

[5.2 Google Maps und Places API 4](#_Toc487486995)

[5.3 Weather API 4](#_Toc487486996)

[5.4 Beziehung von Zugverbindungsdaten über GTFS 4](#_Toc487486997)

[5.5 Routenfindung 4](#_Toc487486998)

[5.6 Datenmodell der Trips 4](#_Toc487486999)

[5.7 Social Media Export 4](#_Toc487487000)

[6 Ausblick und Zusammenfassung 6](#_Toc487487001)

[7 Quellen 8](#_Toc487487002)

[8 ANHANG 9](#_Toc487487003)

[Anhang I Lastenheft 9](#_Toc487487004)

[Anhang II Frontend 10](#_Toc487487005)

[Anhang III Backend 11](#_Toc487487006)

[Anhang IV API‘s 12](#_Toc487487007)

[Anhang V Dijkstra-Algorithmus 13](#_Toc487487008)

# Aufgabenstellung

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

Im Zuge des Projektes wird eine Android-Applikation im Zusammenhang mit Bahnreisen innerhalb Europas entstehen. Ziel ist es, dass die entwickelte Applikation den Anwendern ermöglicht, eine Reise durch europäische Städte auf Grundlage des intereuropäischen Bahn-Tickets, dem sogenannten Interrailpass, zu planen und einen hilfreichen digitalen Reisebegleiter zu erstellen.

Zur Quellcode-Versionierungsverwaltung wurde das Versionierungswerkzeug Git verwendet. [Was ist Git, Bekanntheit…]. Um die Entwicklung zur verinfachen über mehrere private Rechner hinweg, wurde Git in der von dem Online-Dienst GitHub verwendet. GitHub selber hostet eine Vielzahl von überwiegend Open Source Softwareprojekten. GitHub selber erweitert die ursprüngliche Git Funktionalität der Versionsverwaltung um ein agiles Projektmanagement mit Issues bzw. Aufgaben und Meilensteinen, welche je nach Themenbereich zu Personen und Meilensteinen referenziert werden können. 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, da es hier auch nicht um sehr technisch versierte Anwender handeln kann. Der Benutzer 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“ / Überblick-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 und offline 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).

Was wird verwendet, was liegt vor?

# Systemüberblick

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).

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 des Views. Zunächst wurde das Layout der Android-App implementiert. So war es möglich einen ersten Eindruck der Farbgestaltung und der Größenverhältnisse zu gewinnen. Die jetzige Farbgestaltung wurde aufgrund der Logos des „Interrail“ getroffen. So harmonieren diese.

Nachdem das Layout der Android App implementiert war, ging es darum diese mit Inhalt zu füllen. Aus diesem Grund war es notwendig, die Datenbeschaffung voran zu führen. Deshalb wurde nach den richtigen API’s gesucht. Diese mussten die unterschiedlichsten Informationen bereitstellen. Dabei mussten Nutzungsbedingungen beachtet werden. Auch musste für jede API ein Nutzungsschlüssel angefordert werden. Diese konnten dann in das vorhandene Layout eingebaut werden.

Parallel wurde an einer Lösung für die Routenfindung gearbeitet. Dessen Suche führte letztendlich zum Dijkstra-Algorithmus. Wieso der?

Anschließend konnte an den unterschiedlichsten Views gearbeitet werden, da die Daten aus den API’s schon bezogen werden konnte. So war es möglich Städte auf einer Karte auszuwählen und zu sortieren. Die Speicherung der Daten wurde in diesem Zug auch implementiert, damit die ausgewählten Daten gespeichert werden konnten.

Als letztes wurde der Dijkstra-Algorithmus implementiert. So war es möglich das nun auch die kürzeste Verbindung zwischen zwei Städten herauszufinden und diese in einem Trip abzuspeichern.

Dies führte dann zum Abschluss der Implementierung der Android-App

## Entwurf des Layouts

* Unsere Layout Entwürfe

## Google Maps und Places API

* Bibliotheken, Zugangsschlüssel, Kartenanbindung, XML Layout

## Weather API

* JSON Format, Zugriffsart, Realisiert druch …

## Beziehung von Zugverbindungsdaten über GTFS

* GTFS Feeds, Python, Problem, nicht verwertbar, Abstraktion

## Routenfindung

* Java Graph Modell, Dijkstra, interne API, UML

## Datenmodell der *Trips*

* Sugar ORM, ORM Erklären,

## Social Media Export

* Export via Android API, Als Bild mit Text, Beispiel

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**

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.

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.

Für die Wetterdaten wurde eine Openweather Api verwendet. Diese liefert die Temperatur und die Luftfeuchtigkeit einer Stadt zurück. Allerdings ist auch diese auf 60 Aufrufe pro Minute begrenzt.

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 4.2 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.

All die Daten die nachher für eine Tour gespeichert werden, werden in eine SQLite Datenbank abgelegt mit der Unterstützung einer Sugar ORM ORM Erklären, Was ist ein Object Relational Mapper und wann ist er sinnvoll, wann nicht? Bibliothek. Dabei werden folgende Informationen für eine Tour abgespeichert:

* Start- und Zielstadt
* Start- und Endzeitpunkt
* Länge der Tour
* Übrige Städte in einer separaten Tabelle mit Fremdschlüssel auf das Tour Objekt
* Unterkünfte ebenfalls in einer separaten Tabelle mit Verweis auf die jeweilige Stadt

So ist es möglich eine Tour anzulegen und eine solche auch wieder auszulesen.

Ein Schema der Api‘s ist im Anhang abgelegt (Anhang IV).

**Routenfindung**

Die Routenfindung wird durch einen Dijkstra-Algorithmus realisiert. Dieser beschreibt den kürzesten Pfad nach Edgar Dijkstra. Dies bedeutet, dass die Vertices, Edges und Trips als Attribut der Edges aus einer CSV-Datei geladen werden. Vertex sind dabei Orte die ausgewählt wurden. Edges hingegen sind Direktverbindungen zwischen solchen Vertex Städten. Kanten sind eine Liste von Trips, die mit Zugnummer, Zugmodell und Abfahrt beschrieben werden. Des Weiteren gehören zu den Kanten noch die Informationen zur Entfernung und zu den Kosten dieser Fahrt, welche durch die Fahrzeit in Minuten berechnet werden. Dabei wird die Route nun durch eine inverse Routenbildung aufgrund von Zeigern berechnet. Das Ergebnis ist eine Liste von Trips.

Ein Klassendiagramm zum Dijkstra-Algorithmus ist im Anhang abgelegt (Anhang V)

# 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.

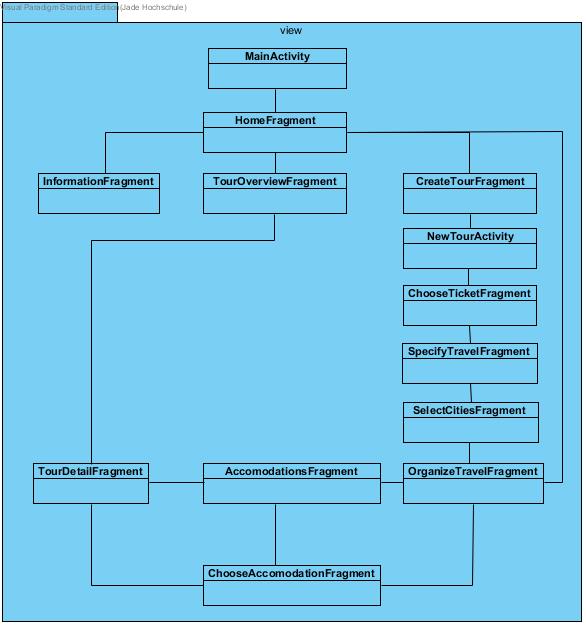
# Quellen

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 Backend

Ein 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 Dijkstra-Algorithmus

