Projektarbeit

Im Studiengang Informatik

Sommersemester 2016



Beweg' Dich! -

Kann ich noch zur nächsten Haltestelle laufen, bevor der Bus kommt ?

Projekt-Team:

Ego, Ninos

Gubo, Christian

Kampfer, Felix

Koch, Nadja

Krempel, Matthias

Ruttmann, Katrin

Werlitz, Viktor

betreut von:

Prof. Dr.-Ing. Alexandra Teynor

Einleitung

Aufgabenstellung

Analyse

Geplanter Funktionsumfang

Projektorganisation und Management

Infrastruktur

Irgendwas 2.1.1

Frameworks und Bibliotheken

Architekturbeschreibung

Programmablauf

Benutzertest

Verbesserungsmöglichkeiten und Ausblick

**Einleitung**

In einem Zeitalter von steigender Bequemlichkeit, verknüpft mit einem europaweiten Ausbau an öffentlichen Verkehrsmitteln, kommt man als Durchschnittsbürger zwar immer schneller durch Städte, dabei wird die sportliche Betätigung jedoch immer geringer. Für viele bietet sich nicht die Möglichkeit an, “mal eben so” Sport zu betreiben. Motiviert durch die unzähligen Vorteile von regelmäßigem Sport, wurde zusammen mit dem Departement für Sport, Bewegung und Gesundheit der Universität Basel und der Fakultät für Informatik der Hochschule Augsburg eine Projektarbeit in die Wege geleitet, welche Benutzern von öffentlichen Verkehrsmitteln die Möglichkeit bietet, ungenutzte Wartezeiten zu nutzen, um dort mehr Bewegung in den Alltag zu integrieren.

**Aufgabenstellung**

Im Zeitraum von 17.03.2016 - 29.06.2016 sollte eine Anwendung entwickelt werden, welche für den User berechnet, ob er die nächste Haltestelle vor dem öffentlichen Nahverkehr bequem zu Fuß erreichen kann. Dazu soll die Anwendung, eine Fahrplanauskunft geben, die Position des Nutzers bestimmen, dessen Wegzeiten zur nächsten Haltestelle berechnen und den Weg des Nutzers zur nächsten Haltestelle auf der Karte aufzeigen. Diese Applikation soll die Fahrplaninformation der Stadtwerke Augsburg und des BVB Basel beinhalten.

Das Hauptziel des Projektes ist es, Menschen dazu zu bringen sich mehr zu bewegen statt untätig auf den Bus zu warten, denn Bewegung ist gesund. Dieses Projekt wird zusätzlich zur Hochschule Augburg, auch von der Universität Basel, Department für Sport, Bewegung und Gesundheit (<https://dsbg.unibas.ch/>) betreut. Die entstehende App kann Grundlage für Forschungsarbeiten an der Universität Basel sein. Um dieses Ziel in den Gang zu setzen, sollte die App das Interesse der User wecken, deswegen sollte die App gut bedienbar sein und auch Wert auf das GUI-Design gelegt werden. Außerdem spielt bei der Entwicklung der App auch die Qualität des Codes eine wichtige Rolle, denn nur so kann sie fehlerfrei funktionieren und bei Bedarf schnell verbessert und erweitert werden. Ein weiteres Ziel dieser Projektarbeit ist das Lernen der Zusammenarbeit im Team, gemeinsame Fehlerbehebung, das Kennenlernen typischer Probleme in Projektgruppen und die Vorbereitung der Teilnehmer auf spätere Gruppenprojekte in der freien Wirtschaft. Doch auch die Verbesserungen der Fähigkeiten im Bereich App-Entwicklung und der Python-Programmierung sind nicht zu vergessen.

**Analyse**

Die Applikation soll im Alltag, beim Warten auf eine Straßenbahn oder einen Bus verwendet werden, dadurch ist der User an ein mobiles Endgerät gebunden. Da es in der heutigen Zeit viele verschiedene Anbieter gibt und es daher viele Entwicklungsmöglichkeiten gibt, haben wir uns entschieden einen gemeinsamen Standard zu wählen, welcher auf (fast) allen Geräten ausgeführt werden kann. Eine Webanwendung ist leichter zugänglich und verteilbar, als eine Native Mobile App. Anschließend stellte sich die Frage nach der Realisierung der Webanwendung. Hier haben wir uns aufgrund unseres Studiums für das Python-basierte Web-Framework Django entschieden. Da es für uns weniger Einarbeitungszeit benötigt und so alle ein Grundwissen bezüglich der Programmiersprache haben.

Die benötigten Schnittstellen der Applikation ließen sich schnell finden, so muss eine Schnittstelle zur Fahrplanauskunft der beiden Verkehrsbetriebe Augsburg und Basel implementiert werden, und eine Karte angezeigt werden, auf welcher der Fußweg angezeigt werden kann.

Da vom Augsburger Verkehrsverbund, trotz mehrmaliger Nachfrage keine Hilfe bereitgestellt wurde, lösten wir die Schnittstellen über die EFA XML-Schnittstelle, welche viele der deutschsprachigen Verkehrsbetriebe bereitstellen, so auch AVV und BVB.

Die Karte wurde zunächst durch Google Maps umgesetzt. Da diese Anwendung jedoch ab 1000 Anfragen pro Tag eine Gebühr verlangt, haben wir uns in der Mitte des Projekts zu OpenStreetMaps umentschieden.

Auf dem Markt existieren bisher keine Programme wie unsere geplante Applikation. Jedoch stellen die Verkehrsbetriebe in Apps und Webseiten Fahrplaninformationen bereit, und auch andere Unternehmen wie die Deutsche Bahn stellen solche Apps zur Verfügung. Insbesondere die App des AVV wurde genauestens betrachtet und auf Funktionsweisen überprüft, so kann diese auch Wegzeiten angeben, jedoch nur an Start und Zielpunkten.

**Geplanter Funktionsumfang**

* Als User nächste Haltestellen finden
* Start und Ziel angeben
* Abfahrtszeiten sehen
* Karte anzeigen + Wegstrecke
* „wie lange laufe ich zur nächsten haltestelle“
* für Augsburg und Basel

**Projektorganisation und Management**

**Infrastruktur**

* GitLab (Versionskontrolle)

Ohne Versionsverwaltung kommt im Prinzip kein Software-Projekt aus. Wir haben uns für GitLab entschieden, ein relativ junges aber bereits sehr stabiles und fortgeschrittenes Git-Frontend mit eingebauter Projektverwaltung. GitLab verwaltet dabei eine beliebige Anzahl von Projekten oder genauer gesagt Git-Repositories. Die Projektverwaltung besteht aus Milestones und Issues mit Labeling-/Tagging-Funktion aber das reicht für die meisten Projekte aus. Das Ganze wird abgerundet durch ein Wiki für jedes Projekt. Die Vorteile an GitLab sind, dass die Bedienung auch für neue Benutzter einfach zu lernen ist. Neben dieser bequemen Verwaltung der Git-Repositories bietet Gitlab weitere Vorteile beim Arbeiten mit den Repositories. Durch die Übersicht auf den Projekt-Seiten können sich die am Projekt Beteiligten leicht einen Überblick über den aktuellen Entwicklungsstand verschaffen.

* Jenkins (CI)

Jenkins ist ein Continuous Integration Server. CI hat das Ziel, die Qualität der Software über permanente Integration ihrer einzelnen Bestandteile zu steigern. Statt die Software nur in sehr großen Zeitabständen kurz vor der Auslieferung zu erstellen, wird sie in kleinen Zyklen immer wieder erstellt und getestet. So können Integrationsprobleme oder fehlerhafte Tests frühzeitig und nicht erst am Tag des Release erkannt werden. Durch die kleinen Deltas zwischen den einzelnen Builds sind Fehler wesentlich leichter zu finden und zu beheben. Den Projektleitern und Kunden stehen, je nach Bedarf, stets aktuelle Informationen über den Entwicklungsstand oder ein aktuelles Testsystem zur Verfügung.

* Pycharm (priorisierte Entwicklungsumgebung)

Eine sehr wichtige Frage war, für welche Programmiersprache wir uns entscheiden. Nach der groben Orientierung, was wir alles für das Projekt brauchen könnten und nach langer Diskussion, haben wir uns anschließend für Python entschieden. Beim zugehörigen Framework, welches für die Entwicklung von Webanwendungen benötigt wird, haben wir uns für Django entschieden, welches speziell für Webanwendungen entwickelt wurde. Bei der Entwicklungsumgebung wurde im Team kein Standard festgelegt, welcher verwendet werden muss, jedoch wurde Pycharm von unserem Team priorisiert.

Die Bibliothek von Python erlaubt eine Abstraktion auf einem weitaus höheren Level als in anderen traditionellen Programmiersprachen. So kann beispielsweise ein Webserver in wenigen Zeilen geschrieben werden.

Sehr vorteilhaft ist Tkinter, ein plattformübergreifendes Werkzeug zur Programmierung grafischer Benutzeroberflächen.

Generelle Eigenschaften:

* Python

Vorteile: einfache und konsistente Syntax

umfangreiche Standardbibliothek („batteries included“)

einfache Einbindung von C-Bibliotheken bzw. gute Unterstützung zum Schreiben von Modulen in C

es existieren Implementierungen speziell für die Java VM (Jython) und die CLR (IronPython)

Nachteile: als interpretierte Sprache relativ langsam, aber schnell genug für viele Anwendungen

Anwendungsgebiete: Webanwendungen, Einsatz als Skriptsprache, Prototyping, Desktopanwendungen, diverses[/list]

* Ehemals Taiga
* Whatsapp (schneller Informationsaustausch)

-

**Irgendwas 2.1.1**

**Frameworks und Bibliotheken**

* Django (web framework)
* Python (sprache)
* EFA XML-Schnittstelle
* Json

Die Soll-Konzeption dient der Entwicklung von umsetzungsfähigen Lösungsansätzen zu den in der Ist-Analyse aufgedeckten Schwachstellen und Problembereichen. Sie orientiert sich an den im Projektauftrag definierten Zielen der Organisationsuntersuchung. (Grobentwurf)

**Architekturbeschreibung**

Routenalgorithmus:



Der Algorithmus erhält als Parameter die Position des Nutzers als GPS-Koordinaten und das Ziel des Nutzers in Form einer "Stopid", welche jede Station eindeutig in der Stadt identifiziert.

Zunächst werden mit Hilfe der EFA Webschnittstelle normale Routen ermittelt, diese werden auch auf der Website der AVV ausgegeben.

Für jede Route wird nun ein Worker gestartet, dadurch können alle Routen parallel bearbeitet werden und der Overhead durch die Kommunikation über das Netzwerk wird reduziert.

In der Route wird nun jede Station geprüft, dabei wird bei der Station die dem Nutzer am nächsten liegt gestartet. Dies passiert ebenfalls parallel. Es wird ermittelt, welche Zeit der Nutzer benötigt um zu dieser spezifischen Station zu gehen. Nun wird analysiert, ob der Nutzer den gleichen Bus an dieser Station nehmen könnte. Die Abfahrtszeit des Buses an der jeweiligen Station ist bekannt.

Das passiert solange, bis eine Station nicht mehr rechtzeit erreicht werden kann oder die Endstation erreicht wird. Nun wird in jeder Route die Station gesucht, die am weitesten vom Nutzer entfernt ist, bzw. die längste Gehzeit besitzt.

Diese gefundenen Station werden nun wieder in die EFA-Webschnittstelle gegeben und zurückgegeben werden dann die optimierten Routen für diesen Nutzer/Standort. Da die EFA-Webschnittstelle bei jeder Suche mehrere Routen vorschlägt, werden gleiche Routen, sowie Routen der gleichen Linie, die später fahren gefiltert. Zuletzt werden alle verbleibenden Routen der Abfahrtszeit nach sortiert.

Anfänglich wurde Algorithmus komplett seriell erstellt um Fehler durch Parallelisierung auszuschließen. Die Messzeiten lagen bei ca. 10-15 Sekunden. Durch die Optimierung mit mehreren Workern kann die Zeit deutlich auf 2-6 Sekunden reduziert werden. Das Ansprechen der Webschnittstellen benötigt hier den größten Teil der Zeit, deshalb ist es kein Problem viele Threads zu starten, da alle lediglich auf ein Ergebnis der API warten.

Leider wurde bei der Parallelsierung zuerst die falsche Funktion verwendet. In Python gibt es verschiedene Möglichkeiten zu Parallelisierung.

**Programmablauf**

- ZUSTANDSDIAGRAMM

- aktivitätendiagramm

**Benutzertest**

- automatisierte GUI Tests

- NUTZERTEST HABEN WIR NOCH NICHT

**Verbesserungsmöglichkeiten und Ausblick**

Was ist aus diesem Projekt geworden und was wir gelernt haben.

Die Ergebnisse der Arbeit, die Antwort(en) auf die Forschungsfrage(n) sowie daraus resultierende Schlussfolgerungen sind nochmals kurz und prägnant zu formulieren und zu begründen.

Darüber-hinaus ist die Tauglichkeit des gewählten Vorgehens und der gewählten Methode zu diskutieren. Ein Fazit und ein Ausblick auf weitere Forschungsperspektiven schließen dieses Kapitel ab.