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

Inhaltsverzeichnis

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 der Basler Verkehrs-Betriebe 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, Departement 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 Web-Anwendung ist leichter zugänglich und verteilbar, als eine native Mobile-App. Anschließend stellte sich die Frage nach der Realisierung der Web-Anwendung. 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 Teilnehmer ein Grundwissen bezüglich der Programmiersprache haben.

Die benötigten Schnittstellen der Applikation ließen sich schnell finden. So muss eine Schnittstelle zur Fahrplanauskunft der Verkehrsbetriebe in Augsburg und Basel implementiert werden, und eine Karte angezeigt werden, auf welcher der Fußweg dargestellt 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 über Google Maps umgesetzt. Da diese Anwendung jedoch ab 1000 Anfragen pro Tag eine Gebühr verlangt, wir auf Open Source großen Wert legen und die Karte noch nicht mit Funktionalitäten befüllt war, haben wir uns in der Mitte des Projekts zu OpenStreetMaps um entschieden.

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

Bei der Umsetzung von Software-Anforderungen entstehen insbesondere Herausforderungen im Hinblick auf die Kommunikation zwischen den Nutzern und den Software-Entwicklern. Und die Auftraggeber oder die späteren Nutzer der Software müssen mit den Personen kommunizieren, die die Software realisieren. Ein wichtiger Grundsatz bei User Stories ist, dass sie aus der Sicht eines Anwenders geschrieben werden. Daher werden sie auch frei von technischen Fachbegriffen gehalten. Jeder, der am Projekt beteiligt ist, also auch jeder Anwender, sollte sie optimalerweise verstehen können. Nachdem wir unsere Stories auf Papier gebracht haben, haben wir Diese anhand ihrer Wichtigkeit und ihrer Auswirkung nach MoSCoW-Priorisierung (Must, Should, Could, Wont) realisiert. Damit haben wir die wichtigsten Ziele zusammengefasst. Diese gelten sowohl für Augsburg als auch für Basel.

Essentielle Projektanforderungen

1. Anzeige der nächsten Haltestelle

Als User möchte ich die nächste Haltestelle finden, damit ich zu Fuß laufen kann um in den nächsten Bus einsteigen zu können.

2. Auswahl von Start und Ziel

Als User möchte ich Start und Ziel angeben können, damit ich eine Route bekomme und weiß wie ich laufen soll.

3. Ansicht von Abfahrtszeiten

Als User möchte ich Abfahrzeiten angezeigt bekommen, um zu wissen, wann die nächste Straßenbahn oder der nächste Bus kommt.

4. Anzeige der Karte mit eingezeichnetem Weg

Als User möchte ich die Karte und die Wegstrecke angezeigt bekommen, um zu wissen, wie ich laufen soll.

5. Zeitangabe der Weglänge

Als Benutzer möchte ich die Gehzeit bis zur Einstiegshaltestelle in Minuten angezeigt bekommen, um zu wissen wieviel Zeit ich brauche um die nächste Haltestelle zu erreichen und wieviel Zeit mir noch bleibt um mein Ziel erreichen zu können.

Mögliche Projektanforderungen

1. Benutzerverwaltung

Als Administrator möchte ich Benutzerkonten sperren & aktivieren, Benutzerprofile aktualisieren, und die Möglichkeit haben, die Registrierungsmaske aktivieren und deaktivieren zu können.

2. Login-Funktion

Als Benutzer will ich mich einloggen können um meine gespeicherten Einstellungen für die aktuelle Nutzersession zu verwenden.

3. Profiländerung

Als Benutzer möchte ich meine Kontoinformationen (E-Mail, Passwort) ändern können.

Projektorganisation und Management

Vorgehensweise

Das Software Engineering umfasst die Themen der Projektplanung, der Projektorganisation und der Projektdurchführung, also das Management und die Organisation von Softwareprojekten. Die Projektorganisation und das Management nehmen in Projekten eine Vielzeit von Aufgaben wahr. Diese Aufgaben wurden in der Gruppe besprochen, einer oder mehreren Person(en) zugewiesen und in GitLab als “Issue” mit Verantworlichem formuliert und dementsprechend erledigt. Gegenseitige Unterstützung, mehrmaliges Treffen in einer Woche und ständige Online-Kommunikation haben einen guten Überblick über den aktuellen Stand der App-Entwicklung verschafft. Die delegierten Aufgaben haben wir getrennt oder in kleingebildeten Gruppen etwa in der folgenden Reihenfolge erledigt.

1. Feststellung und Analyse der Ziele

Die Ziele wurden klar definiert und priorisiert, weil man in einem befristeten Zeitraum nicht alle Anforderungen erfüllen kann.

2. Einarbeitungsphase

Nachdem die priorisierten Anforderungen und Mittel gefunden waren, wurden Diese dementsprechend studiert.

3. Beschreibung der Systemarchitektur

In diesem Punkt wurden die Zusammenhänge zwischen den Anforderungen und dem zu konstruierenden System beschrieben und dazu eine grobes Design entworfen.

4. Feindesign, Zustandsdiagramm

Um eine bessere Übersicht über den Projektplan zu schaffen, wurden ein Zustandsdiagramm erstellt. An diesem Diagramm sind die Beziehungen zwischen den verschieden Anforderung gut darstellbar.

5. Lauffähiger Programmcode

In diesem Punkt wurden die Anforderungen in Code umgewandelt. Hier zeigte sich die Qualität der hervorgegangenen Planung, durch welche die Entwicklung erleichtert wurde. Jedoch stellt man sich auch in dieser Phase Herausforderungen, da vorhergehend nicht alles beachtet werden konnte, was sich in dieser Phase als wichtig herausstellt.

6. Retrospektive

Erwähnungwert ist der Punkt Retrospektive, denn das gemeinsame Zurückschauen und Bewerten der Zusammenarbeit, hilft rechtzeitig, Lösungen und Verbesserungen zu finden und durchzuführen.

7. Tests

Wie schon erwähnt, ist die fehlerfreie Software ein Hauptziel unseres Projekts, deswegen spielen die Tests auch eine wichtige Rolle. Deshalb haben wir und potenzielle Benutzer die Software auf Erfüllung ihres Einsatzes geprüft. Spätestens hier sollten möglichst alle Fehler gefunden und beseitigt werden.

8. Dokumentation

Die Dokumentation fasst die wichtigen Informationen über das Projekt zusammen und soll diese zur weiteren Verwendung darstellen.

9. Organisation des Projekttages

An unserem Projektstand wurden Interessierte durch Plakate mit den wichtigsten Informationen versorgt und konnten die App live testen. Ebenso standen wir bereit Erläuterungen zu geben und Fragen zu Beantworten. Der Vortrag am Ende des Tages lieferte nocheinmal genaueste Informationen zum Projekt und dessen Implementierung.

10. Projekttag

Das Interesse war immer da, und wir haben mit großer Freude und großem Stolz unsere App vorgestellt.

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, dies 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.

Ehemaliges Taiga

Im Rahmen der Produktentwicklung einer Web-App haben wir uns vorab mit einigen Tools für Projektmanagement auseinander gesetzt, um anschließend das aus unseren Sicht passendste Tool für unsere Zwecke einzusetzen. Diese Zwecke sollten uns erlauben einen grundsätzlichen Plan für die Produktentwicklung zu erstellen, Aufgaben und Teilschritte einzuteilen und Ressourcen zu planen und zu verwalten. Ein Grund, wieso wir uns auch schlussendlich für Taiga entschieden haben, war der offene Quellcode der es

jedem erlaubt die Anwendung bei sich selbst zu hosten, diese zu erweitern oder zu adaptieren. Und da wir Flexibilität suchten, war die Entscheidung nach einigen Tests des Tools schnell gefallen. Durch Fehler und Probleme, wie zum Beispiel, bei der Einleitung von User Stories und davon abgeleiteten Tasks und falsch abgespeicherten E-Mails haben wir uns gemeinsam zum Umstieg auf GitLab entschieden.

Whatsapp

Natürlich kann die beliebteste Chat-App Whatsapp auf modernen Smartphones nicht die richtigen Managertools ersetzen, doch für den schnellen Informationsaustausch ist diese sehr gut geeignet. Zum Beispiel kann eine Verspätung schnell mitgeteilt werden oder wenn man bei einer Aufgabe Hilfe brauchte, besprach man diese auch per Whatsapp. Da es eine App ist und heutzutage (fast) jeder eine Internetverbindung hat, kann man problemlos einander erreichen und wichtige Sachen besprechen. Jeder weiß Bescheid. Dabei hat man auch den Nachteil, dass bei sehr vielen Nachrichten der Überblick verloren gehen kann und vielleicht die wichtigen Informationen untergehen könnten. Nichtsdestotrotz waren wir mit dieser Kommunikationsart zufrieden.

Frameworks und Bibliotheken

Django (web framework)

Beim zugehörigen Framework, haben wir uns für Django entschieden, welches speziell für Web-Anwendungen entwickelt wurde. Bei der Entwicklungsumgebung wurde im Team kein Standard festgelegt, welcher verwendet werden muss, jedoch wurde PyCharm von unserem Team priorisiert.

Python (Programmiersprache)

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. 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. Der andere Vorteil war die einfache und konsistente Syntax.

Bootstrap

Bootstrap ist ein populäres Frontend-Framework, mit welchem sich mobilfähige Webseiten mit moderner Technik einfach und zuverlässig erstellen lassen. Bootstrap wurde in unserem Projekt in Kombination mit einem LESS-Compiler angewandt.

jQuery

Das beliebte JavaScript-Framework jQuery diente als Grundlage unserer JavaScript-Bibliotheken. Mit jQuery ist es relativ einfach HTML-Elemente anzusprechen, abzufragen oder auch zu ändern.

EFA Schnittstelle

Um an die Daten der verschieden Verkehrsbetriebe (in unserem Fall der Augsburger AVV und der BVB aus Basel) zu kommen hatten wir Glück, da beide Anbieter Schnittstellen des EFA-Dienstes der Mentz Datenverarbeitung GmbH verwenden. Die Verkehrsbetriebe bieten dabei eine einfache Webadresse an, an die man Argumente per URL übergibt und anschließend Daten im XML- bzw. JSON-Format zurückbekommt. Dabei werden keine Gebühren verlangt und es erfolgt auch keine Form von Benutzerauthentifizierung. Das einzige Problem dabei war der komplette Mangel an öffentlich zugänglicher Dokumentation - weder bei den Bereitstellern der EFA-Schnittstelle, noch vom Entwickler selber gibt es eine Anleitung. Auch durch E-mail Verkehr und persönlichen Kontakt mit dem AVV-Kundendienst konnten wir keinerlei Informationen auftreiben.

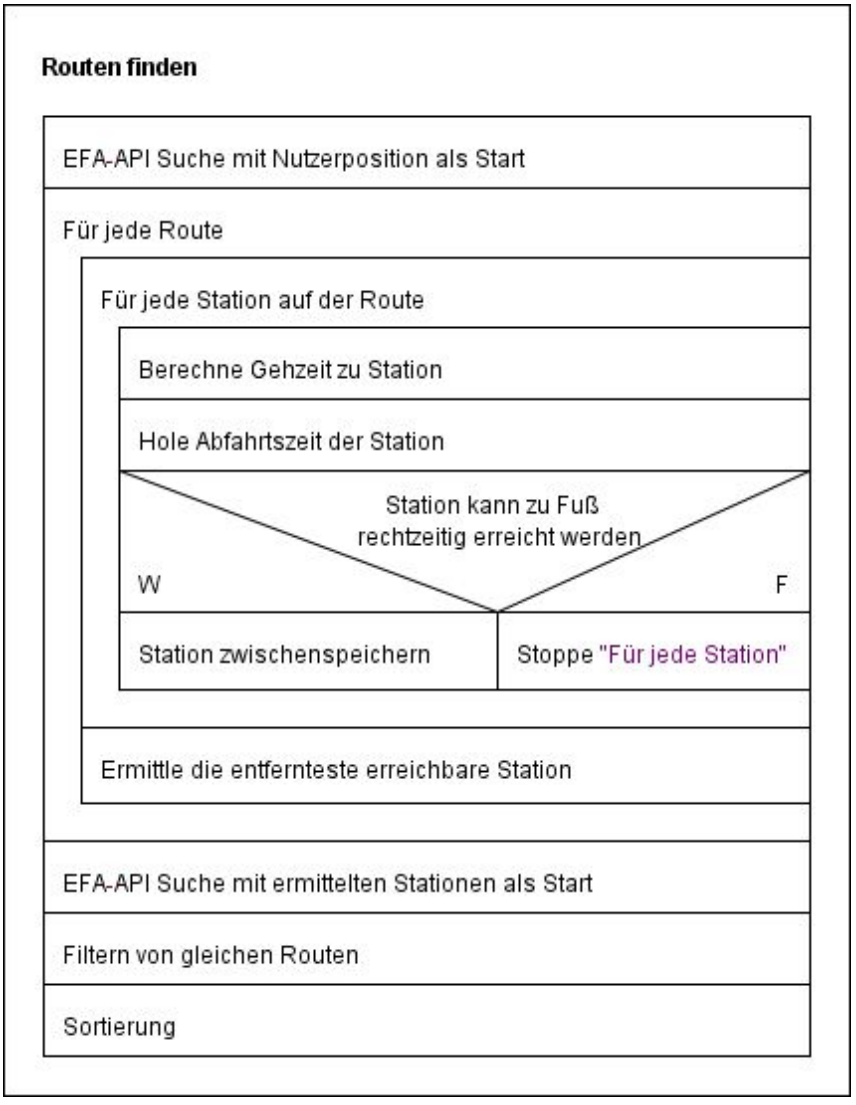
Nach langer Suche (ca. ein Monat nach Projektstart) fanden wir die veröffentlichten Folien eines Entwicklertreffens des österreichischen Verkehrsbetriebs Wiener Linien und konnten daraus unsere ersten sinnvollen EFA-Anfragen konzipieren.

Architekturbeschreibung

Die Konstrukte der "Strukturierten Programmierung" kann man auch mit Diagrammen darstellen, zum Beispiel mit einem Struktogramm, beziehungsweise nach den Erfindern

Dr. Ike Nassi und Dr. Ben Shneiderman genannten Nassi-Shneiderman-Diagramm. Mit Kontrollstrukturen wird der Ablauf eines Algorithmus gesteuert, mit dem Vorteil, dass der Programmierer zu disziplinierter Gestaltung des Programmablaufs gezwungen wird.

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 Bus 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.

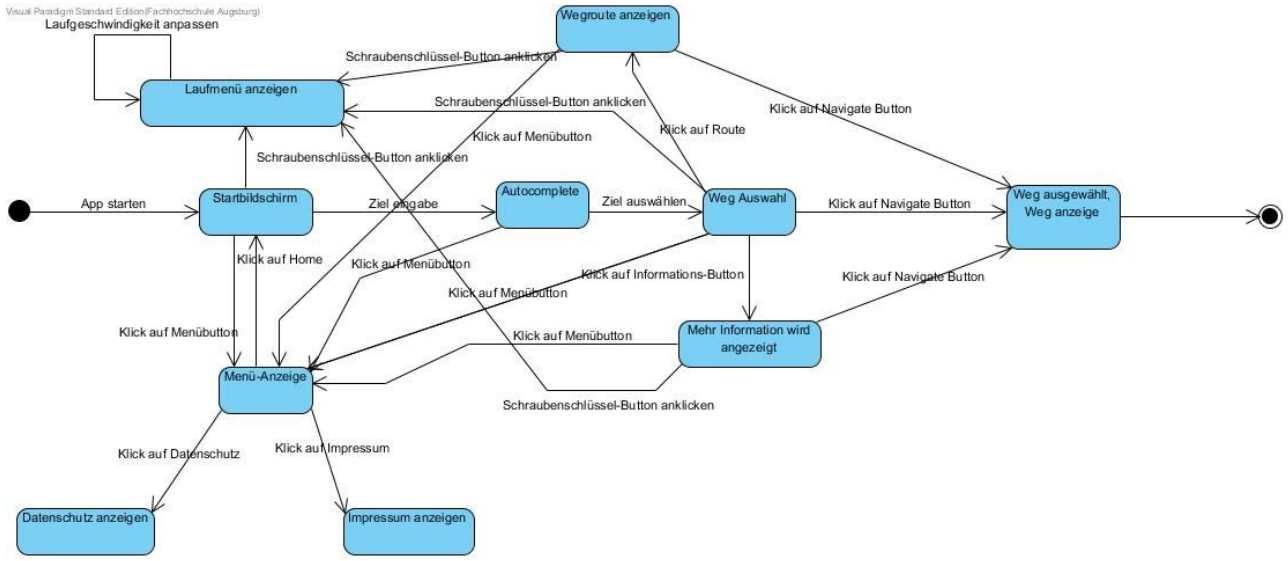
Diese gefundenen Station werden nun wieder in die EFA-Webschnittstelle gegeben und zurückgegeben werden dann die optimierten Routen für diesen Nutzer/Standort. Da die EFA-Webschnittstelle bei jeder Suche mehrere Routen vorschlägt, werden gleiche Routen, sowie Routen der gleichen Linie, die später fahren 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. Die Funktion “map” in dem Pythonmodul “Multiprocessing” stellt eine einfache Möglichkeit bereit, eine bestimmte Sache zu parallelisieren. Dabei werden mehrere Prozesse erzeugt, die jeweils einen Webrequest an die externe API schicken und die Antwort verarbeiten. Nach einigen Tests gab es jedoch keine Verbesserung der Performance. Eher das Gegenteil trat ein. Dies ist verwunderlich, da die meiste Zeit für das Warten der Antwort aus dem Netzwerk benötigt wird. Hierbei entsteht keinerlei Rechenlast und somit auch keine gegenseitige Verlangsamung der Prozesse. Da nach einigen Optimierungen und Veränderungen keine Verbesserung erfolge wurde auf eine threadbasierte Möglichkeit gesetzt. Dabei gibt es Worker, kleine eigenständige Threads, die in einer globalen Warteschlange(Queue) nach

neuen Aufgaben suchen, die abgearbeitet werden müssen. Wenn eine solche Aufgabe gefunden wurde, wird sie bearbeitet und danach in die Ergebnisliste(Resultqueue) geschrieben. Mittels des Threadings konnte nun eine deutliche Performancesteigerung festgestellt werden.

Programmablauf



Das Programm kann durch den Aufruf der URL im Browser gestartet werden. Nach dem Start befindet sich der User auf dem Startbildschirm, auf welchem eine Karte und ein Ziel-Input angezeigt wird. Er kann dort ebenfalls das Menü durch einen Button am linken Bildschirmrand aufrufen oder Anpassungen vornehmen, indem er auf einen Button am rechten Bildschirmrand klickt. Im Menü befinden sich die Unterpunkte “Home”, durch das man wieder auf die Startseite zurückgeführt wird, “Datenschutz”, durch das die Datenschutz-Vereinbarungen mit den User angezeigt werden und “Impressum”, durch welches das Impressum der Applikation angezeigt wird. Wird der Ziel-Input angeklickt, kann dort nach einer passenden Zielhaltestelle gesucht werden. Durch eine Autocomplete Funktion wird dieses Suchen vereinfacht. Hat sich der User durch Klick auf eine der Haltestellen entschieden, werden ihm mehrere Haltestellen angezeigt, von welchen er laufen könnte. An diesem Punkt kann sich der User die Zwischenhaltestellen zu der

jeweiligen Haltestelle anzeigen lassen, dazu muss der “i” Button am rechten Rand der jeweiligen Haltestellen-Anzeige ausgewählt werden. Ebenso können Weglinien angezeigt werden, indem auf den Namen der Haltestelle geklickt wird. Entscheidet sich der User für eine Haltestelle, kann er diese durch einen Klick auf den jeweiligen “Navigate” Button ins Navigationsfenster gelangen, in welchem die Route und die verbleibende Laufzeit, sowie die Abfahrtszeit des Wagens angezeigt wird.

Benutzertest

Um die Schwachstellen und Probleme bei der Interaktion von Benutzern mit der App „Beweg' Dich“ aufzudecken und damit die Effektivität und Effizienz während der Nutzung zu steigern, haben wir eine Reihe von Testmethoden eingesetzt. An Usability und User Experience kommt man heutzutage nicht mehr vorbei, sie sind ein entscheidendes Maß für Zufriedenheit oder Unzufriedenheit bei den Benutzer und sollten auf keinen Fall vernachlässigt werden.

Wir haben User Experience (UX) Testing durchgeführt, um die Nutzungserlebnis bei der Interaktion mit unserer App zu testen. Die sicherlich beste und effektivste Methode das herauszufinden, ist der Test mit realen Nutzern. Dabei wurden die Benutzer beim Testen der App beobachtet und alle Fehler, die dabei aufgetreten sind, notiert. Zum Beispiel, haben wir darauf geachtet, ob die Überschriften richtig verstanden wurden oder die Buttons richtig benutzt wurden, die bereits erstellte Information ausreichend war und ob das Design einfach, übersichtlich und verständlich war. Dabei haben wir natürlich darauf geachtet, ob es funktioniert, dass man an das Ziel kommt, ohne sich zu verlaufen und den Bus erreicht, denn eine Software, die zwar einfach zu bedienen ist, aber nicht ansprechend genug wirkt, ist nicht ausreichend. Die dabei entstandenen Fehler, haben wir sofort dementsprechend verbessern können und die App benutzerfreundlicher gestaltet. Da die App-Entwicklung sich ständig änderte, war gezieltes manuelles Testen schneller und einfacher durchzuführen als automatisierte Tests.

Verbesserungsmöglichkeiten und Ausblick

Der aktuelle Stand der App, bietet noch viele Möglichkeiten der Weiterentwicklung. Es könnte unter anderem auch die Möglichkeit überprüft werden, direkt das Ziel anzulaufen oder weiter entfernte Haltestellen. Da dies mitunter schneller gehen kann, als nur die nächste Haltestelle anzulaufen.

Außerdem könnte man noch Nutzerprofile erstellen. Dadurch könnten Laufdaten gesammelt werden, um bessere Laufstrecken für den Nutzer zu finden oder auch ein Wettbewerbssystem, um sich mit anderen Nutzern zu vergleichen, ist denkbar.

Des Weiteren müsste der Bereich Optimierung verbessert werden, da das Thema bei uns nur wenig betrachtet wurde.

Was für folgende Arbeiten besser beachtet werden sollte, ist ein konstanter Austausch. Da dies die Zusammenarbeit erheblich verbessert. Außerdem sollten die Zeit besser eingeteilt werden, um so ein optimales Ergebnis zu erzielen.

Bei unserem Vorgehen hatten wir grundsätzlich ein großes Problem, dass immer eine Person Experte für einen Bereich war und wir dadurch bei einem Ausfall dieser Person erhebliche Probleme gehabt hätten.

Trotzdem war das Projekt aus unserer Sicht ein voller Erfolg und eine lehrreiche Erfahrung für alle Teilnehmer.