A rule-based geospatial reasoning system for trip price calculations

Stefan Schenk

Supervisor: Willem Brouwer

Advisor: Mewis Koeman

Department of Software Engineering
Amsterdam University of Applied Sciences

This dissertation is submitted for the degree of
Bachelor Software Engineering

April 2018

Todo list

Table of contents

Chapter 1

Introduction

What was once an ordinary startup known as Uber, is now the most famous taxi dispatch company in the world [1]. In the same year that Uber was founded, a similar startup in the Netherlands called taxiID was launched; an Amsterdam based company providing end-to-end cloud solutions and mobile applications for taxi companies. Hailing a taxi has rarely been performed by sticking out ones hand, hoping to catch the attention of a bypassing taxi driver ever since. The ability to order a cab lies at everyones fingertips, literally. Recently, taxiID has started developing a new brand called YourDriverApp (YDA), a lighter and newer version of the original solution, being more focussed on smaller taxi companies. Despite the fact that YDA is new, it still depends on the price calculation functionality of the legacy system. This chapter expands on how this matter is translated into the assignment.

1.1 Context

taxiID was founded as a startup that successfully introduced smartphone taxi booking in The Netherlands, offering a wide variety of IT solutions to serve the taxi market, including a passenger app, a driver app, and administrative panels. More specifically: an app for passengers to order a taxi, an app for drivers to receive their job assignments, and services for all size businesses, offering convenient planning and dispatching without requiring local installations. Businesses that make use of taxiID's services can be found anywhere in the world. This intoduces complicated challenges while developing applications that rely on clearly defined locations and infrastructures, often vastly differing between countries, if these countries have such a system to begin with. The taxiID development team responsible for solving these problems is located in Medemblik, consisting of two mobile app developers (iOS and Android), two backend developers, a designer and two project managers.

1.2 Problem Definition

YDA depends on the price calculation module that is part of the legacy system for which it was designed and implemented. When a passenger books a ride, the departure and destination locations that have been selected are sent to the legacy system. It then proceeds and constructs a list of prices for each vehicle type that is available based on matching pricing rules that have been defined by the taxi company offering the rides. If directors of a taxi company using YDA want to modify their pricing rules, they will be obligated to use the taxiID portal, which has to store company information in a platform that is different from YDA. This makes little sense, as much as it is efficient from a technical point of view, and being easy to maintain and extend. The current price calculation module knows three types of pricing rules: fixed prices based on postal codes, tier prices based on kilometer thresholds, and dynamic calculations based on distance and duration of a ride. A company may have as many rules as required, only one rule will be used to calculate the final price, and the rules are matched in the same order respectively. The fixed rules are defined by downloading, modifying, and uploading a .csv file as presented in table 1.1, the other types of rules are simply managed through a web form.

Departure	Destination	Nr Passengers	Price	Vehicle Type
1462	1313	4	125	...
1313	1462	4	125	...
1462	1313	8	150	...
1313	1462	8	150	...
1462	1012	4	65	...
1012	1462	4	65	...
0	1462	4	65	...
1462	0	4	65	...
1462	AIR1	4	89	...
AIR1	1462	4	89	...

Table 1.1 Comma Separated File containing Fixed Prices in cents

When a passenger books a ride, the price calculation module will first compare the postal codes, amount of passengers, and vehicle types in the fixed pricing rules with the information provided by the passenger's application. The fixed price is returned as soon as a match is found. If no match is found in any of the fixed pricing rules, the system proceeds to calculate a price using a kilometer threshold based rule, given that at least one exists. This type of calculation decreases or increases the price per kilometer for every successive amount of kilometers that have surpassed a predetermined threshold. This concept will be discussed

in chapter 4. If this rule does not exist, a dynamic rule is used to calculate the price based on distance and duration of the ride. Finally, on top of the prices that have been calculated, a discount may be applied as a fixed amount, as a percentage of the price, or as a so called alternative fixed pricing table. When this last option is selected, the price will be calculated all over, using a newly referenced fixed pricing rule. This process is not just hard to understand for a user, who has to reason about the companies prices. But it is also hard to understand for programmers, who have to maintain the code that supports this functionality. A small mistake in the csv file could lead to great issues if the mistake goes through processing undetected.

1.3 Assignment

The title of this thesis reads:

"A rule-based geospatial reasoning system for trip price calculations".

A Trip Pricing System (TPS) must be designed and implemented to calculate trip prices based on user defined pricing rules. Concisely, YourDriverApp requires its own pricing calculation functionality that is similar to the existing taxiID implementation but must not be incorporated into a non-related monolithical, highly coupled system, as it is today. Also, the response body should have the exact same format, and the new system must be able to handle the exact same requests that are made to the current system. Clients must be able to set up pricing rules through the YDA portal, and potentially other portals as well. It is also important that the feature allowing clients to define locations, is usable in countries without a workable postal code system.

1.4 Research

Three main challenges that construct the assignment can be identified. Research must be conducted to attain the best possible way of mapping locations to pricing rules. What this means is that locations must be storable, comparable, and interpretable. The database must be able to store locations in an efficient manner, to which queries can be made as efficiently in order to find out whether a pricing rule applies to a given ride. For this to be the case, the stored locations must be comparable to the departure and destination location of the passenger. The user must be able to reason about his pricing rules, from which an understanding of his defined locations logically follows. But edge cases must be covered completely. For

example, a rule in the current system dictates that a user traveling to Schiphol should receive a discount. But how would the system detect that this is the case? Or what if hotel guests receive discounts, but the neighbour living next to the hotel shouldn't be allowed to benifit from these discounts unless he actually sleeps at the hotel? Secondly, a system has to be developed that encapsulates the solution that is the result of the finished research. It is helpful to extend the research of the problem beyond finding out how to incorporate the answers into a working system, where architecture has a major influence in the tools that are available. For example: if a solution to the main problem requires a database system capable of handling high quantities of geospatial queries, this requirement has to be satisfied in order to proceed in finding the final solution. Finally, a user interface has to be created that enables users to define the pricing rules. The complexity of the interface depends on how straight forward the price calculation system is constructed. The user interface should also be available in multiple portals. The best way of making the systems capabilities available to the user through the UI in the portal, must be investigated. The UI must be built keeping the user in mind, simplifying complex rule management as much as possible.

1.4.1 Questions

From the description of the problem, one main important research question can be derived:

How can a generic location-based price calculation system be implemented

that could be used in every country?

This question encapsulates the four important challenges that have to be dealt with before the project can successfully be implemented. In order to give a clear direction to the research, sub-questions are separated into four groups; location mapping, architecture, trip pricing system, and user interface.

1. In what way can locations be represented to be universally interpretable and precise?
    1.1. Which location types matter for this project?
    1.2. What are the main differences between postal systems used around the globe?
    1.3. Can postal codes be abstracted to geospatial data while retaining the same usefulness in the system?

1.4. Which Database Management Systems (DBMS)s cover the location storage use cases for this project?

2. What is most fitting solution to integrate the backend and frontend into the existing architecture?

   2.1. Which architectural patterns fit in with the exising architecture?

   2.2. Which data of adjecent systems are required to make TPS operational?

   2.3. How can authentication between services be implemented or improved?

   2.4. Which methods and technologies can be used to ensure suitability and improve maintainability and testability?

3. Which logic and information is required to calculate a trip price breakdown?

   3.1. How are rules matched?

   3.2. Which parts of the system have an impact on the calculation result?

   3.3. How should the concept of time schedules be realised?

4. How can complex pricing rules be communicated through the UI?

   4.1. Which views are essential?

   4.2. In what way can visual hierarchy guide the user through processes naturally?

   4.3. How should complex elements impacting price calculations be communicated to the user through UI components?

Answering these questions will lead to the implementation of a solid, straightforward, user-friendly system that utilizes the user interface to communicate the inner-workings of the rule-based price calculation system.

1.5 Process

A desire from within taxiID to use the SCRUM methodology to potentially improve their development process is an important factor to set up this project in a way that would introduce the team to SCRUM without forcing developers and CEO's to adopt it right away. All team members are familiarized with tools, roles, workflows, and the project artifacts somewhat indirectly. Because of the novelty of SCRUM in regard to the product owner, a pregame phase is introduced for preparation purposes, see Table 1.2. A written working method

is provided to the product owner, see Appendix A, Phase I - Pregame. This document clearly documents the interpretation of the product owners product vision and reflects from a developer viewpoint, so that miscommunications and misinterpretations can be resolved before the project is started.

Phase I - Pregame			Phase II - Game							
week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11
product definition	architectural vision	proposed solution	sprint 1	sprint 2	sprint 3	sprint 4	sprint 5	sprint 6	sprint 7	sprint 8

Table 1.2 Project roadmap

It contains an architectural vision and a proposed solution, which is agreed upon by the product owner before the backlog is created. Reading the document is recommended if more knowledge about the process and context of the assignment is desired.

References

[1] U. T. Inc. (2011) The uber story. [Online]. Available: https://www.uber.com/en-NL/our-story/

[2] J. R. Herring, "Implementation standard for geographic information - simple feature access - part 1: Common architecture," May 2011. [Online]. Available: http://portal.opengeospatial.org/files/?artifact_id=25355

[3] (2004) Geographic information – simple feature access – part 1: Common architecture. [Online]. Available: https://www.iso.org/standard/40114.html

[4] J. R. Herring, "Implementation standard for geographic information - simple feature access - part 2: Sql option," August 2018. [Online]. Available: http://portal.opengeospatial.org/files/?artifact_id=25354

[5] (2018) Postgis 2.4.5dev manual. [Online]. Available: https://postgis.net/docs/manual-2.4/using_postgis_dbmanagement.html#PostGIS_GeographyVSGeometry

[6] (2018) Mysql 5.7 reference manual - geometry class. [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/gis-class-geometry.html

[7] I. K. Center. (2018) Three-tier architectures. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/covr_3-tier.html

[8] J. Stenberg. (2014) Experiences from failing with microservices. [Online]. Available: https://www.infoq.com/news/2014/08/failing-microservices

List of figures

List of tables