České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jan L i n k a

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Android aplikace pro plánování cyklistických tras a cyklonavigaci

Pokyny pro vypracování:

1. Prozkoumejte existující cykloplánovače a navigační aplikace.
2. Ve spolupráci s experty na cyklistiku ze sdružení AUTO*MAT vytvořte „user stories“, „use cases“ a požadavky. Hlavní požadavky jsou plánování cyklotrasy, navigace a sledování trasy.
3. Navrhněte aplikaci a prototyp otestujte s cyklisty.
4. Implementujte cyklonavigační aplikaci pro systém Android.
5. Vyzkoušejte cyklonavigaci s reálnými uživateli.

Seznam odborné literatury:

[1] Václav Legát: Mobile journey planner and navigation for cyclist, 2014.
[2] Jan Hrncir and Qing Song and Pavol Zilecky and Marcel Nemet and Michal Jakob: Bicycle Route Planning with Route Choice Preferences. In Prestigious Applications of Artificial Intelligence (PAIS). 2014.
[3] C. M. Barnum: Usability Testing Essentials. Ready, Set … Test!, Elsevier - Morgan Kaufmann, 2011.
[4] J. Broach, J. Dill, and J. Gliebe, ‘Where do cyclists ride? A route choice model developed with revealed preference GPS data’, Transportation Research Part A: Policy and Practice, 46(10), 1730 – 1740, (2012).

Vedoucí bakalářské práce: Mgr. Jan Hrnčíř

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Jan L i n k a
Study programme:	Open Informatics
Specialisation:	Computer and Information Science
Title of Bachelor Project:	Android App for Bicycle Route Planning and Navigation

Guidelines:

1. Survey existing bicycle routing and navigation apps.
2. In collaboration with experts on cycling from AUTO*MAT, specify user stories, use cases, and requirements. The main requirements are bicycle route planning, navigation, and route tracking.
3. Design the app and create a prototype that will be tested with cyclists.
4. Implement the bicycle navigation app for the Android platform.
5. Evaluate the bicycle navigation with real users.

Bibliography/Sources:

[1] Václav Legát: Mobile journey planner and navigation for cyclist, 2014.
[2] Jan Hrncir and Qing Song and Pavol Zilecky and Marcel Nemet and Michal Jakob: Bicycle Route Planning with Route Choice Preferences. In Prestigious Applications of Artificial Intelligence (PAIS). 2014.
[3] C. M. Barnum: Usability Testing Essentials. Ready, Set … Test!, Elsevier - Morgan Kaufmann, 2011.
[4] J. Broach, J. Dill, and J. Gliebe, 'Where do cyclists ride? A route choice model developed with revealed preference GPS data', Transportation Research Part A: Policy and Practice, 46(10), 1730 – 1740, (2012).

Bachelor Project Supervisor: Mgr. Jan Hrnčíř

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 14, 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Bachelor's Project

Android App for Bicycle Route Planning and Navigation

Jan Linka

Supervisor: Mgr. Jan Hrnčíř

Study Programme: Open Informatics

Field of Study: Computer and Information Science

May 22, 2015

Acknowledgements

I would like to thank my supervisor Mgr. Jan Hrnčíř for helpful guidance, valuable comments, and patience, Bc. Pavol Žilecký for cooperation during his development of the backend, and finally, everyone involved in the testing of the app for their time and interest.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 22. května 2015 . . . .

Podpis autora práce

Abstract

The project deals with an implementation of an Android app for urban cyclists. The final app available under the name "Cykloplánovač" on Google Play offers the route planning with respect to cyclists' preferences and navigation during the ride. Furthermore, it can track taken journeys and use the track in "Do práce na kole" competition or to voluntarily share it to improve the route planning and to allow planning of future bicycle infrastructure. The text of the project formalizes requirements, describes the app architecture, the important parts of the implementation and the process of development and testing. The usage data from over 1000 installations is evaluated with a preview of over 1500 tracked journeys.

Abstrakt

Práce se věnuje implementaci Android aplikace pro cyklisty ve městech. Výsledná aplikace dostupná pod názvem "Cykloplánovač" na Google Play nabízí uživateli naplánování trasy s ohledem na požadavky cyklistů a navigaci během jízdy. Dále umožňuje zaznamenávání projetých tras a jejich použití v rámci soutěže "Do práce na kole" nebo dobrovolné sdílení za účelem vylepšení plánování a možné budoucí úpravy cyklistické infrastruktury. Text práce se věnuje formalizaci požadavků, popsání návrhu aplikace, důležitých prvků implementace a postupu vývoje a testování. Zároveň ukazuje data o užívání z 1000 instalací a náhled do 1500 zaznamenaných tras.

Keywords: urban cycling, Android app, bicycle planning, bicycle navigation, bicycle use tracking

Klíčová slova: městská cyklistika, Android aplikace, cykloplánování, cyklonavigace, zaznamenávání cyklojízd

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Urban cycling is a type of transportation that is affordable and considered as “green” to the environment. However, there are obstacles for the increase in the number of cyclists. Many aspects decrease their comfort such as elevation gain, road quality, and traffic volumes.

Urban cyclists can be divided into two categories. Experienced ones, who use bicycle on their commute often, and casual ones. Cyclists share some views on obstacles on their commute, however there are differences, e.g., how they view the danger of traffic or how uncomfortable are they with the elevation gain [6].

Creating a mobile application can help both categories of cyclists. One of the most perceived problems is a traffic. Dedicated planner for cyclists can find route with minimal traffic, good road quality, and lowest elevation gain optimised for user’s experience level. The mobile nature of the application also allows to lead the user through complex city areas by providing a map and instructions such as when to turn at the right moment.

A mobile application could also track journeys. This record can show the user the benefits of cycling such as lowered environmental damage and workout for his or her health. Furthermore, the tracks gathered from larger number of cyclists could be used for cyclist infrastructure planning by a municipality, or for the improvement of the cycle route planning.

1.2 Aim of the Project

The aim of the project is to build a mobile application (app) for Android smartphones capable of bicycle route planning, navigation, and route tracking.

To create an app that will be used by real users, requirements will be created in collaboration with experts on cycling from AUTO*MAT. Furthermore, existing bicycle routing and navigation apps will be surveyed for existing functionality and user feedback about desired functionality not available in current apps.

Gathered requirements will be then formalised in a form of “user stories” describing context for real life use, use cases describing interaction between the app and the user, and functional and non-functional requirements.

The process of implementation will start with a digital prototype testing graphical user interface ideas. It will also be used for testing their usability by cyclists. The implementation of the app for the Android platform will use a planning backend developed by Pavol Žilecký in his master's thesis [12] for route planning and gathering the user feedback including anonymised tracked journeys. The app will be released publicly on Google Play Shop to be available especially to the participants of "Do práce na kole" competition in May 2015.

Finally, the number of the real users and the numbers of the requested plans will be evaluated. The voluntarily gathered tracked journeys will also be previewed.

1.3 Structure of the Project

The project is structured in 8 chapters. In Chapter 2 the requirements for the app are stated and then existing apps are surveyed in Chapter 3. The components of the app are described in Chapter 4 and important parts of implementation are in Chapter 5. The process of the development and the testing is described in Chapter 6. The usage numbers by real users and preview of gathered data are in Chapter 7. Finally, the released app as a result is discussed in Chapter 8 with possible future improvements.

Chapter 2

Analysis

Requirements for the app were gathered from three main sources. Firstly, basic requirements for bicycle route planing and navigation are formulated in Václav Legát's Bachelor's project [10]. Secondly, tracking, recording, and reporting user's route are required for "Do práce na kole" (DPNK) competition and for planning of the sfuture improvements for bicycle infrastructure by AUTO*MAT NGO[1]. Finally, sending feedback including planned and taken route is necessary for further improvement of server backend.

With all previous requirements in mind, following user stories, use cases, functional and non-function requirements were created.

2.1 User Stories

Following stories describe how the app might be used by real users.

2.1.1 Visiting a Friend

Karel wants to visit his friend, who recently moved to new place. Because the weather is nice today, he decides to use his bicycle. He looks up the address using the app and gets a route from his current location. As he is in no rush, he picks bike friendly route and after reviewing the distance he decides to set out. The route is new for him, so he fastens his phone to the phone holder and rides using turn-by-turn navigation.

2.1.2 Running Errands

Tomáš needs to visit the library. He knows where it is located so he picks its location from map and gets a recommended route. As he wants to finish this quickly, he views all recommended routes and finds a shortcut. He knows this part of the city well, so he just memorizes one new turn and saves the route in case he gets lost.

[1] `http://www.auto-mat.cz/`

2.1.3 Commuting and Competing

Jana used to drive her car to work every day but not too long ago her work group decided to start commuting using bicycles to help environment and get exercise. They also entered “Do práce na kole” competition to get everyone motivated. Now she picks her saved workplace as destination and lets the app tracks her journey. When she arrives she can see how long did it take and her average speed and add the travelled distance to her total in competition. Later she can review how much she helped to reduce air pollution on the competition web page.

2.2 Use Cases

Following use cases describe interaction with the app in formal matter. Each of five diagrams describes a goal of the user.

Use Case 1: Planning a route

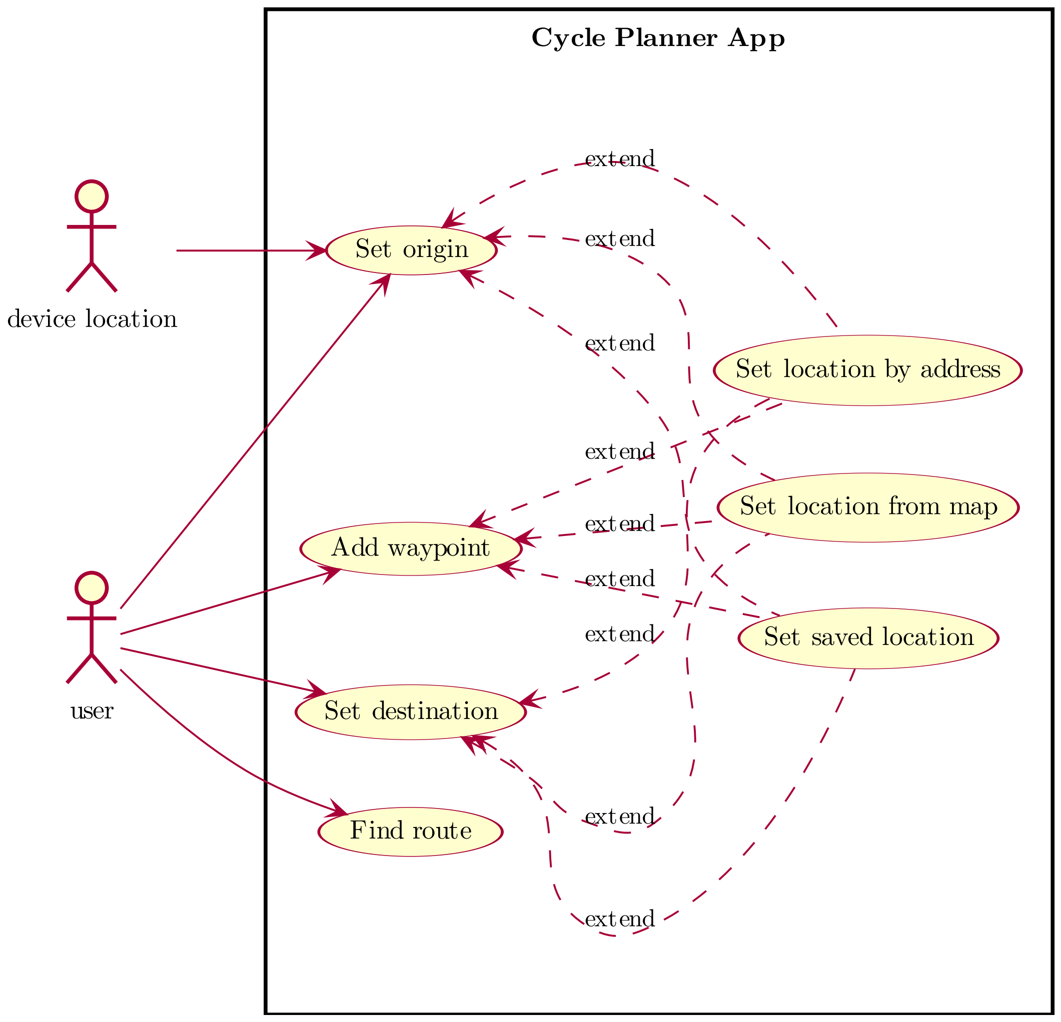


Figure 2.1: Use Case 1: Planning a route

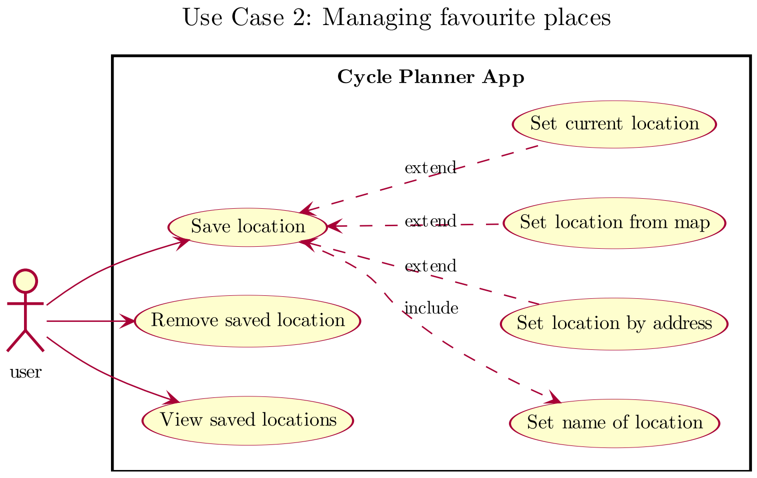


Figure 2.2: Use Case 2: Managing favourite places

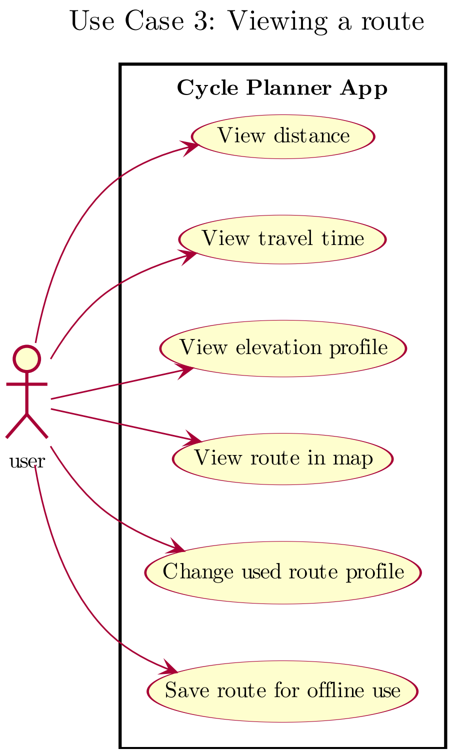


Figure 2.3: Use Case 3: Viewing a route

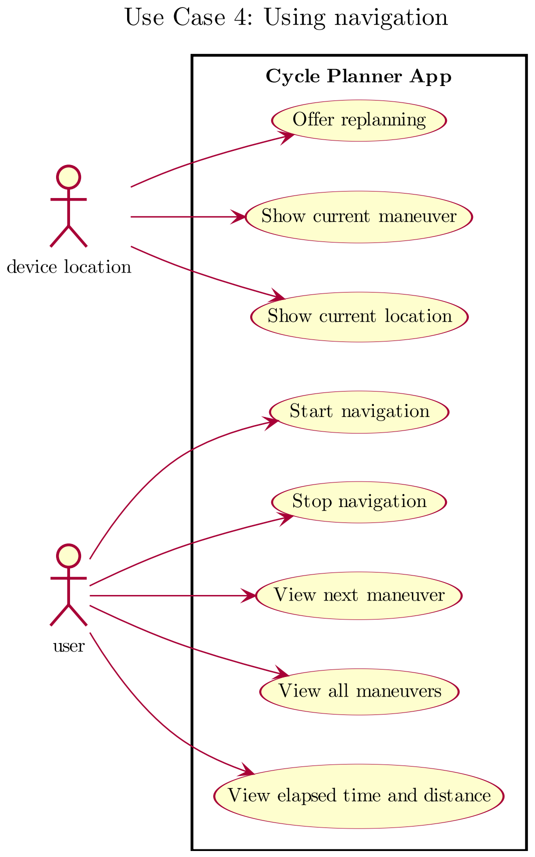


Figure 2.4: Use Case 4: Using navigation

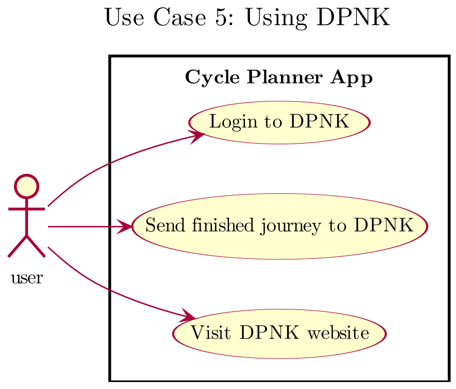


Figure 2.5: Use Case 5: Using DPNK

2.3 Functional Requirements

Following features were expanded, modified, and prioritized using e-mail conference with experts from AUTO*MAT NGO. Each functional requirement (FRQ) has an assigned priority: (M) – must, (S) – should, (C) – could, (W) – would (a space for future improvements)

Requirements are further annotated by their status in relation to existing code base from Václav Legát's Bachelor's project. (N) denotes new requirement, (E) expanded one or requiring large GUI improvements, (U) denotes requirements that can be fulfilled using existing code. Further remarks on state of code base are in following section.

FRQ01 The application will allow to set origin and destination from map, address search, device location provider or saved location. (M, E)

FRQ02 The application will find route between an origin and a destination. (M, E)

FRQ03 The application will offer turn-by-turn instructions with map for a found route. (M, E)

FRQ04 The application will be able to save favourite locations. (M, U)

FRQ05 The application will offer to send user feedback to backend. (M, E)

FRQ06 The application will track user's journey. (M, E)

FRQ07 The application will save history of request & response & tracked journey. (M, N)

FRQ08 The application will send anonymised data about tracked journey to backend. (M, N)

FRQ09 The application will offer sending tracked journey to DPNK system. (S, N)

FRQ10 The application will show links for opening DPNK web page. (S, N)

FRQ11 The application will allow saving route & map for offline use. (S, N)

FRQ12 The application will show statistics about tracked journey. (C, N)

FRQ13 The application will show summarized statistics about tracked journeys. (C, N)

FRQ14 The application will offer calculating of prevented emissions (compared to car journey). (C, N)

FRQ15 The application will allow sharing of planned route on social sites. (W, N)

FRQ16 The application will allow sharing of tracked route on social sites. (W, N)

FRQ17 The application will find 3 closest repair shops for bikes. (W, N)

2.4 Non-functional Requirements

NRQ01 The application will run on Android 4.0.4+ devices in at least 2011 year class [11]

NRQ02 The application will be usable on 4" to 5.5" screens.

Chapter 3

Cycling Apps Overview

There are many apps for cyclists with slightly different goals. Few of them offer bicycle route planning and navigation, many more are focused on tracking and sharing trips. Other apps are focused only on drivers and walking but show ideas for navigation screen. As examples of different types, following apps are described: CycleStreets, OsmAnd, BikeCityGuide, MapQuest, Google Maps, and BikeMap.

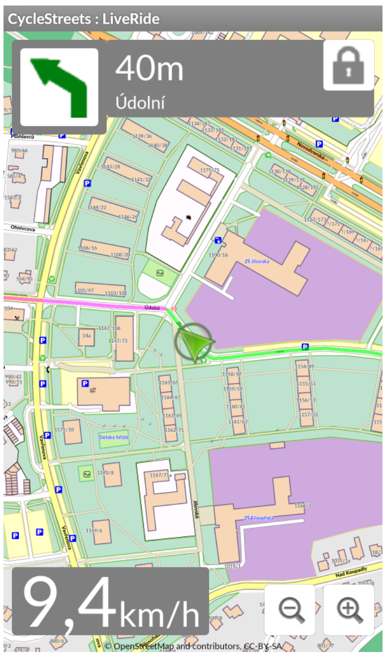


Figure 3.1: CycleStreets[a] is a cycle journey planner focused on the UK, however it can be used in the Czech Republic. It's server back-end can create plans using four different modes and takes hills into account. Main problems of the app is being designed for pre-2011 devices running Android 2.3 and older. This includes old graphical design and triggering legacy menu on newer devices. Furthermore, the installation of thhe offline map for Prague is not intuitive.

[a]`https://play.google.com/store/apps/details?id=net.cyclestreets`

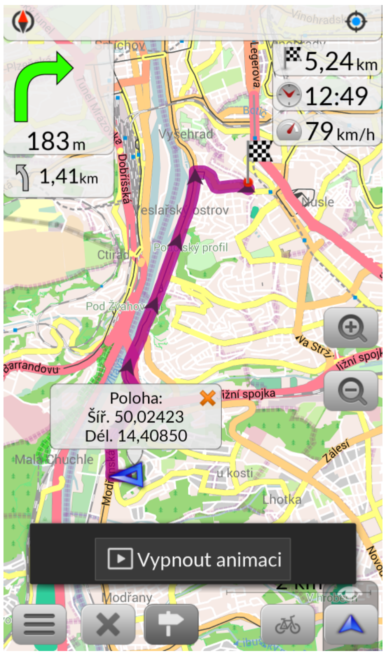


Figure 3.2: OsmAnd[a] is a map and navigation app for cars, bicycles and pedestrians with offline mode. Custom GUI and wide range of features makes the app confusing for cyclists.

[a]`https://play.google.com/store/apps/details?id=net.osmand`

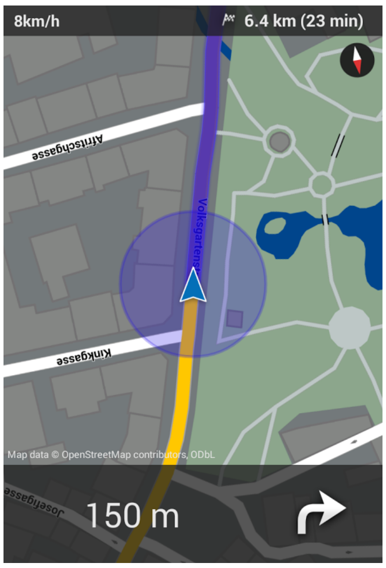


Figure 3.3: BikeCityGuide[a] is a modern app designed for cyclists in cities. The app offers functionality of bike computer and bike tours, but every city is sold as separate add-on, with no Czech city available.

[a]`https://play.google.com/store/apps/details?id=org.bikecityguide`

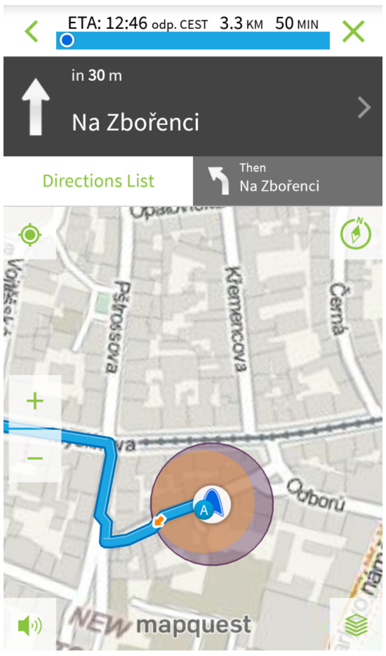


Figure 3.4: MapQuest[a] is a map and navigation app for driving and walking. It has modern, yet different design than apps from Google.

[a]`https://play.google.com/store/apps/details?id=com.mapquest.android.ace`

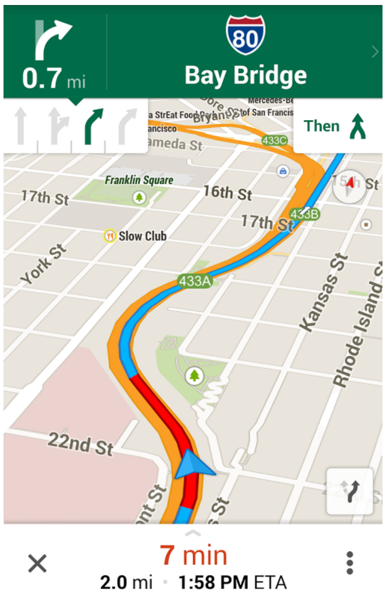


Figure 3.5: Google Maps[a] is a map and navigation app preinstalled on most Android devices. It is created in Google's *material design* and includes cycling navigation that is not available in Czech Republic.

[a]`https://play.google.com/store/apps/details?id=com.google.android.apps.maps`

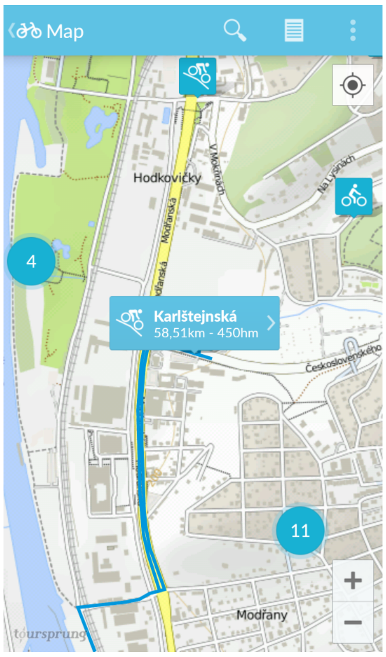


Figure 3.6: BikeMap[a] is an app for recording and sharing bike rides.

[a]`https://play.google.com/store/apps/details?id=com.toursprung.bikemap`

Chapter 4

Architecture

A mobile device with Android OS provides performance comparable to a personal computer in many aspects. Nevertheless, there are many reasons to use components showing response from the powerful backend part of the system via the internet connection, such as that demanding computation would result in long wait for the user and drain of battery power that is one of the limiting aspects of a mobile device.

The app uses four different components connected via interfaces. The most important component is the planning backend that also provides interface for feedback about the plans and the app itself. The backend is developed by Pavol Žilecký [12]. Another component is "Do práce na kole" API able to receive competition results. Finally, Nominatim API provides geocoding and reverse geocoding and Nutiteq provides vector map source. The components and interfaces are shown in Figure 4.1.

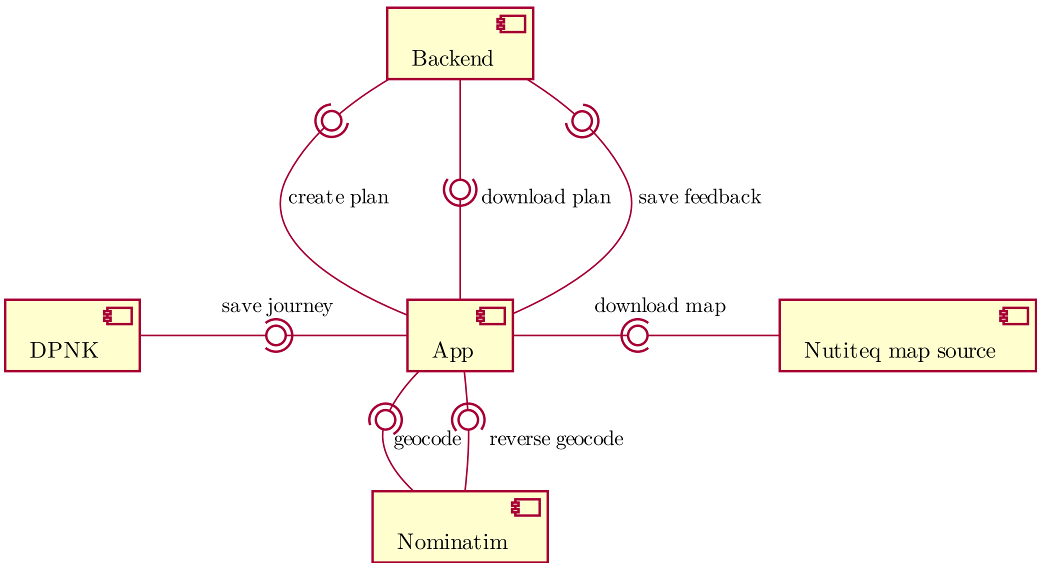


Figure 4.1: Used components

4.1 Backend

The route planner backend used by the app is a specialised planner server developed by Pavol Žilecký in his master's thesis [12]. It utilizes OpenStreetMap[1] (OSM) data and can currently find plans in 26 Czech towns and cities[2]. The criteria used for finding the best route are travel time, comfort, quietness, and elevation gain. Travel time includes changes introduced by riding downhill, uphill, or by other obstacles such as cobblestones. Comfort criterion penalizes bad road surface and difficult turns such as turning left from main road to side road. Quietness criterion prefers low traffic and bicycle infrastructure. The last criterion prefers low elevation gain.

To let the user choose the best type of a plan profile, four plans are provided: commuting, bike friendly, flat, and fast. The commuting profile is designed for people commuting to work or school. It prefers comfort and speed while not ignoring quietness and low elevation gain. Bike friendly profile is aimed at non-commuting people who do not tolerate sharing a road with motor vehicles. The flat profile tries to avoid going uphill to large extent. Finally, the fast profile uses just the lowest travel time.

The backend also receives feedback provided by the users that may include a tracked journey. Final functionality provided by the backend is gathering of anonymised voluntarily sent tracked journeys.

4.2 Do Práce Na Kole

"Do práce na kole" is an annual competition aimed at popularization of urban cycling. Current fifth year is taking place during May 2015 in 26 towns and cities[3] all over the Czech Republic. The participants are logging their commutes to and from work if they ride a bicycle. In 2014, organizers AUTO*MAT NGO and other local NGOs reported over 5700 participants.

The user can use the app to track his or her commute and to log in a send this tracked commute to the competition as journey to or from work. Other data such as date are filled in automatically by the app.

4.3 Nominatim

Nominatim[4] is an open source software that provides geocoding and reverse geocoding. Geocoding in context of the app means providing geographic coordinate for a name of a location (usually a street address). Reverse geocoding is an opposite process of providing

[1] `https://www.openstreetmap.org`

[2] Břeclav, Brno, České Budějovice, Hradec Králové, Hranice, Jablonec nad Nisou, Jihlava, Jindřichův Hradec, Karviná, Kopřivnice, Liberec, Nové Město nad Metují, Nový Jičín, Olomouc, Ostrava, Otrokovice, Pardubice, Plzeň, Praha, Přerov, Říčany u Prahy, Uherské Hradiště, Ústí nad Labem, Zlín, Znojmo

[3] See footnote 2.

[4] `http://wiki.openstreetmap.org/wiki/Nominatim`

nearest known address for a given geographic coordinate. Nominatim functions are accessible through web API which is hosted by OpenStreetMap Foundation or available as software for self-hosting.

The app uses geocoding to allow the user to find destination or other point of plan using the address. This is required because the planner API exclusively uses geographic coordinates. The reverse geocoding is used to provide the user with meaningful names to places picked from map or during tracking of route. Nominatim uses OSM data that are also used by other components of the app.

4.4 Nutiteq Online Vector Maps

Company Nutiteq[5] provides vector maps based on OSM data for users of their Nutiteq SDK maps library. The online distribution combined with on device caching enables the user to use device disk space only for maps of regions he or she really visits. The vector format for maps also provides lower data transmission and better quality on high resolution screens than bitmap format.

Map SDK and map data are important part of the app because they allow user to pick places from the map and provide a way to show planned route in context of surroundings.

[5] `https://www.nutiteq.com/`

Chapter 5

Implementation

This chapter describes the app from a developer's perspective. It covers the graphical user interface, the details of the component interfaces, location acquisition, the generation of the instructions, the navigation, the local database, and used libraries including a maps SDK.

The Android app is written using Java programming language and runs in sandbox environment of Android OS. The app has to deal with new versions of Android OS that add new APIs described by API level but keeping in mind that not every device receives Android OS updates. Distribution of Android OS versions at the time of writing the project is shown in Table 5.1. This division creates problems especially for creating graphical user interface (GUI).

Version	Codename	API level	Distribution
2.2	Froyo	8	0.3%
2.3.3–2.3.7	Gingerbread	10	5.7%
4.0.3–4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x	Jelly Bean	17	18.1%
4.3	Jelly Bean	18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1	Lollipop	22	0.7%

Table 5.1: Android platform versions including corresponding API level [3]. Data about the relative number of devices collected by Google Play Store during a 7-day period ending on May 4, 2015. Any versions with less than 0.1% distribution are not shown.

The app is divided between the GUI, access to other components using internet, route tracking, navigation, and local database for storage of tracked routes, saved places, and cached plans. Interaction with components and the local database is asynchronous to remove workload from the GUI thread and the results are delivered using the publish/subscribe event bus.

The implementation reuses and updates some small parts of the code from Václav Legát bachelor's project [10], i.e., geocoding, saved places GUI, some string resources, and three icons.

5.1 Graphical and Interaction Design Context

Android, as many other operating systems, has its Human interface guidelines to unify user experience. They are currently covered by Material Design document [8], which defines motivation and design patterns in cross platform fashion. This document is however an abstract design and a real world implementation is constrained by hardware and software limitations. This requires specialised guidelines for Android [4]. Moreover, most of Material Design components are available only from API level 21 up. However, subset of functionality is backported to older versions of Android by Google in AppCompat library [1] and by other developers in libraries.

5.2 Component Interfaces

In this section, the interfaces of the main components are described. Both the backend API and “Do práce na kole” API are designed using *Representational State Transfer (REST)* software architecture style. The communication over the internet uses *Hypertext Transfer Protocol (HTTP)* methods POST and GET.

5.2.1 Do Práce Na Kole

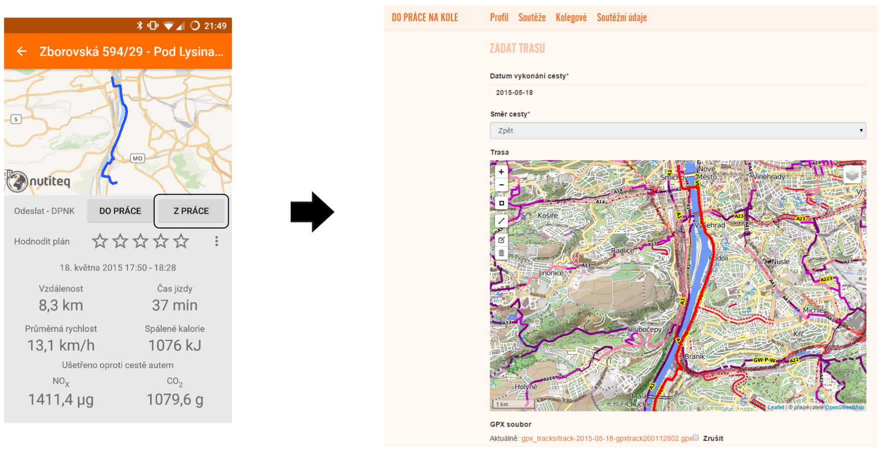
“Do práce na kole” API is a REST API receiving GPX files secured by OAuth 2.0 using “password” grant. The endpoint for OAuth 2.0 `POST /oauth2/token/` returns new temporary *access token* and permanent *refresh token* if called with proper *client id* and *client secret* for the application and either username and password provided by a user or the refresh token from the previous authorization. The endpoint `POST /rest/gpx/` receives GPX file, trip date and direction to work or from work.

This API is used through Retrofit 1.9.0 library[1]. The user can log in through a dedicated screen and the access token from the response is then stored. If the access token expires, new one is requested silently using the refresh token while sending the recorded commute. The example of use in the app is shown in Figure 5.1.

5.2.2 Backend

The backend API is a REST API communicating using HTTP request bodies in JSON format compressed using gzip compression to reduce required data transfer on slow and metered mobile connection. The app uses two endpoints for acquiring plan and one for sending feedback.

[1]`http://square.github.io/retrofit/`



18. května 2015 17:50 - 18:28

Vzdálenost 8,3 km
Čas jízdy 37 min
Průměrná rychlost 13,1 km/h
Spálené kalorie 1076 kJ
Ušetřeno oproti cestě autem
NO_X 1411,4 µg
CO_2 1079,6 g

Datum vykonání cesty* 2015-05-18
Směr cesty* Zpět
GPX soubor
Aktuálně: gpx_tracks/track-2015-05-18-gpxtrack260112802.gpx

Figure 5.1: Example of tracked journey and the same journey uploaded to "Do práce na kole" 2015 web through the app.

`POST api/v2/journey` request is used to require a plan using geographical coordinate of origin, destination, variable amount of via points, and average speed. The server returns response HTTP 201 with *journeyId* under which is the planner result accessible. The full available JSON format is shown in Figure B.1.

`GET api/v2/journeys/journeyId` request is used to download previously requested plan that includes four different plans each including plan steps with information required to visualize them and use them for navigation. The plans are transparently cached in the app to allow limited offline functionality. The full JSON format is shown in Figure B.2.

`POST api/v2/feedback` request is used to send various information from the user to the backend. The full JSON request is used when the user provides his or her feedback about journey that was tracked while using navigation. In that case, the tracked journey, the identification of the plan used for the navigation and feedback from the user are sent. The same request is however used even for automatic sending of anonymised tracked journeys from users that decided to do so. Finally, the general textual feedback about whole app is also sent through this type of request. The JSON format is shown in Figure B.3.

This API is used through OkHttp 2.3.0 HTTP client library[2]. The downloaded plans are cached in app's database to improve functionality in areas with spotty internet connection. The usage in the app is shown in Figure 5.2 and Figure 5.3.

5.3 Location Providers

Typical Android smartphone offers various sensors for determining location and device orientation. The most important sources for location are GPS and the availability of cellular towers and WiFi access points. While GPS offers the best accuracy, it requires clean line

[2]`http://square.github.io/okhttp/`

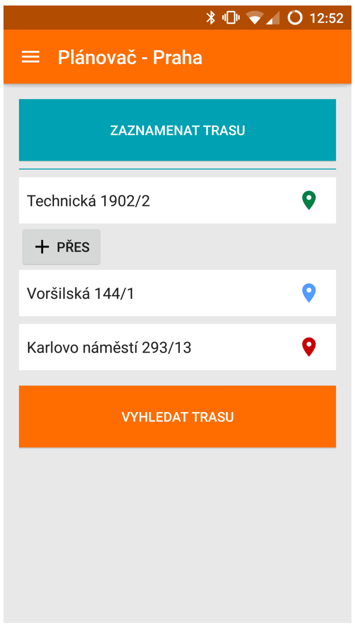


Figure 5.2: Planner with filled in input

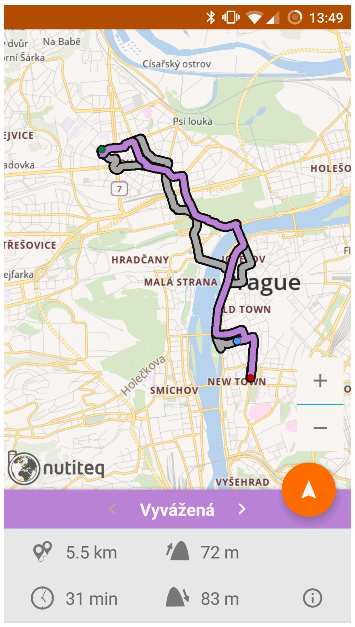


Figure 5.3: Received plans

of sight to satellites that might be obscured by tall buildings in the urban environment. It also has the highest battery consumption. In urban environment, the location can be also determined through the measurement of available cellular towers (Cell ID) and WiFi access points. The drawback is the requirement of the lookup using the internet connection.

Three libraries providing the location were tested for the app: android.location API, KalmanLocationManager library and Google Location Services API.

android.location API is an original API that is a part of Android API from level 1. It provides location from single source whenever GPS or *Network provider* (lookup using cellular towers and WiFi APs) is updated. Google however does not recommend to use this API anymore [2].

KalmanLocationManager library combines GPS and *Network provider* and their reported accuracy to estimate the location using Kalman filtering. This library provided more accurate results than android.location API, reported accuracy estimates were however unrealistic.

Google Location Services API is a part of Google Services package. It provides a "fused" location that is determined using GPS, *Network provider* and other sensors such as an accelerometer. This provides better accuracy and lower battery consumption than android.location API. Coarse location estimate can also be usually obtained immediately. Downside of this API is its dependency on Google Play Services application that is closed source software distributed only to devices that are certified for and logged in Google Play store. The API also requires asynchronous initialization for each Activity.

During testing with multiple devices, Google Location Services API provided the most accurate results. For this reason, the final version of the app uses this API.

5.4 Navigation

The app shows the recommended route in various ways. The map shows the recommended route as a strong line with a plan profile colour and real tracked path as thinner line. The user can switch between tilted view that simulates real view from bicycle and map view that offers better overview of surrounding streets as shown in Figure 5.4 and Figure 5.5.

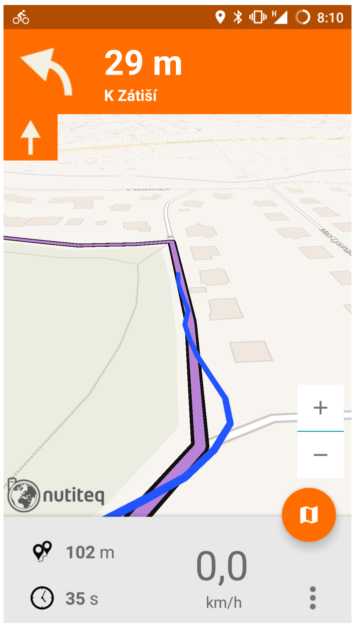


Figure 5.4: Navigation – the tilted view

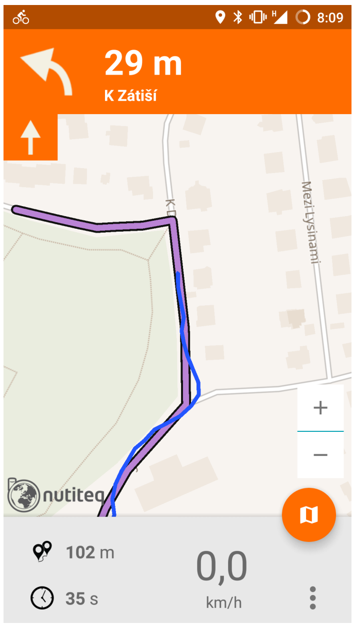


Figure 5.5: Navigation – the map view

The instructions inform a user about upcoming turn or straight manoeuvres. The instructions are created from plan steps where every step represents a point on Earth and distance and travel time to the next step. It also contains street name and surface type. The most important parameter is the angle between the last step and the current step.

To give the user always the appropriate instruction, the plan steps are divided to segments of a maximum length of 20 meters by computing location on the path using spherical Earth model. The instructions readable by the user are created from this internal route representation. Every angle is converted to manoeuvre – continue, turn left or right, slight turn left or right, or U-turn. The slight turn usually represents change between lanes – for the cyclist this usually means to switch between the road and the pavement. The angle conversion is shown in Figure 5.6. After the manoeuvres are determined, the instructions are created by merging consecutive steps with the same manoeuvre and summing the distance and travel time. The final plan step is then used as the geographical coordinate for further computations with the instruction. Finally, each plan step has assigned an instruction to display. The simplified process is shown in Figure 5.7 where the plan steps with the same colour are merged to create an instruction. The instruction is then assigned back to the plan steps with the exception of turn manoeuvre being displayed ahead.

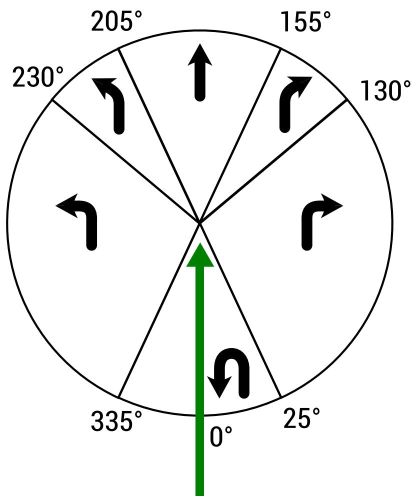


Figure 5.6: Determining manoeuvre from an angle between plan steps.

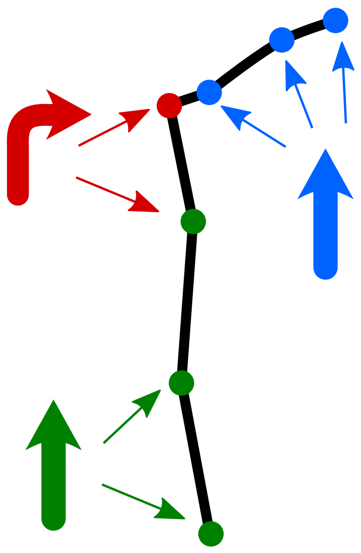


Figure 5.7: Generation of the instructions.

The turn-by-turn navigation is provided by reading an instruction of the closest plan step and adding distance from user's location to coordinates of this instruction. To provide better information to the user, if current instruction is *continue*, any upcoming instruction beside *continue* is shown to the user 160 m ahead. Furthermore, if next instruction comes sooner than 20 m after current one, the preview of the next one is also displayed.

The closest instruction is found by nearest neighbour search in a k-d tree with an average $O(\log n)$ time complexity [7].

5.5 Nutiteq as Maps SDK

The maps SDK is one of most important parts of the app. The maps with additional lines drawn on the device are necessary to show the route plan and navigate the user. They are also used to visualize tracked journeys and to allow the user to choose the origin or the destination without knowing the exact street address.

There are many map solutions available for Android, however there are two requirements that substantially limit the choice. Firstly, the maps have to be based on OpenStreetMap[3] data that are used for generating plans by the backend. This is necessary because some minor paths are missing or shown differently in other map sources. Secondly, the SDK needed to offer tilted view for the map to simulate cyclist's view during navigation.

With these requirements in mind, Nutiteq Maps SDK v3[4] was selected.

[3]https://www.openstreetmap.org
[4]https://www.nutiteq.com/nutiteq-sdk/

5.6 Database

The app uses SQLite[5] database for storing tracked journeys, user saved places, search history, cached plans, and information about cities where the backend can create plans. The database diagram is shown in Figure 5.8.

The database is encapsulated as an Android *content provider* with create, read, update and delete operations. All operations are executed asynchronously from GUI to prevent its unresponsiveness.

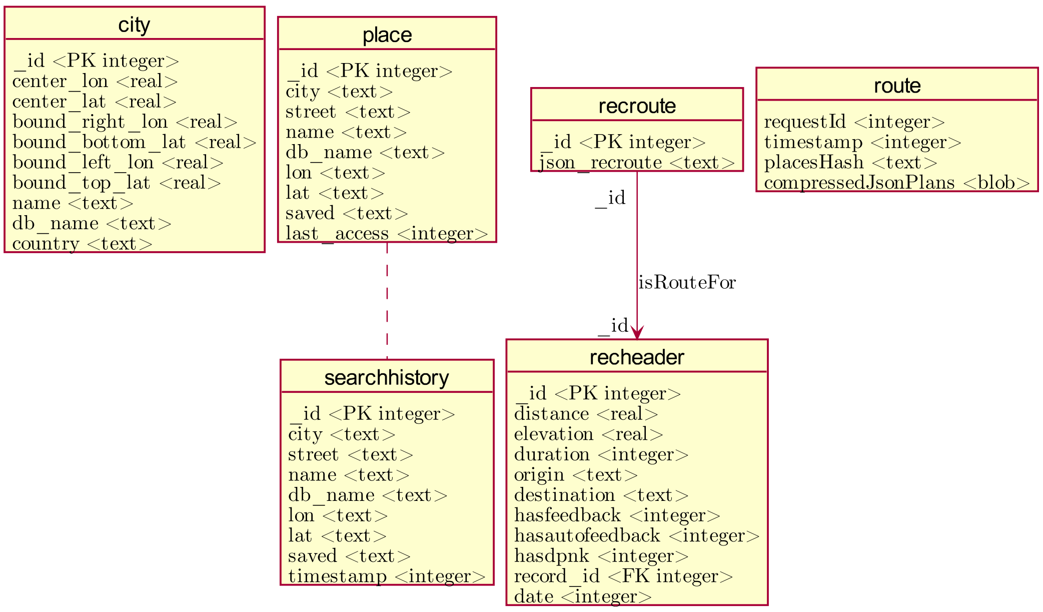


Figure 5.8: Internal storage database diagram

5.7 Libraries

In this section, libraries with roles not related to functional requirements are described.

EventBus[6] by Greenrobot is a library implementing publish/subscribe event bus. Event bus allows easy communication between GUI components and tasks running in background such as database queries, internet communication, and route recording that can run even when the user interacts with a different app.

ACRA[7] is a library enabling the app to report crashes and other unexpected behaviour to the developer with information about the phone model and Android OS version. The library can report by user's email or silently in background. The email is pre-filled with report and the user can add an additional comment. The silent reporting is sent to a backend set up

[5] `https://www.sqlite.org/`
[6] `https://github.com/greenrobot/EventBus`
[7] `https://github.com/ACRA/acra`

by the app's developer. The public version of the app is using silent reporting mode with an open source backend Acralyzer[8] set up on Cloudant[9] data layer.

User interface libraries The AppCompat library by Google brings many of new features, e.g., colour themes to devices with older level APIs such as API level 15 which is a minimum supported by the app. Other user interface elements such as floating action buttons, pop-up layout, circular progress bar, or even element shadows are implemented by libraries by other developers.

5.8 Graphical User Interface

The important GUI elements not related to other implementation concepts are *navigation drawer* and origin, destination, or via point picker screen.

The *navigation drawer* is a pattern [8] used for switching between the top level activities in the app. A drawer that can be opened in many screens in the app allows to switch between the start of the planning or the tracking, saved places management, viewing the tracked journey history, the app settings, information about the app, and sending a feedback about the app. The drawer opened from the planner screen is shown in Figure 5.9.

The place picker screen can be used for simple selection of the origin, the destination, or the via point. As shown in Figure 5.10, it contains a search field for the street address lookup, a button for selection of a location from the map, and up to six last used saved places and places from the search history. The full search history and the full saved places list can be also accessed.

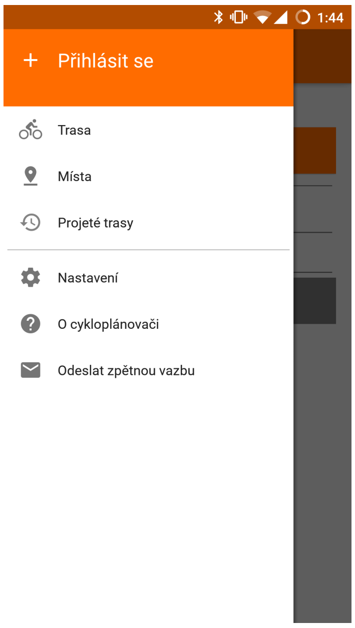


Figure 5.9: Drawer navigation.

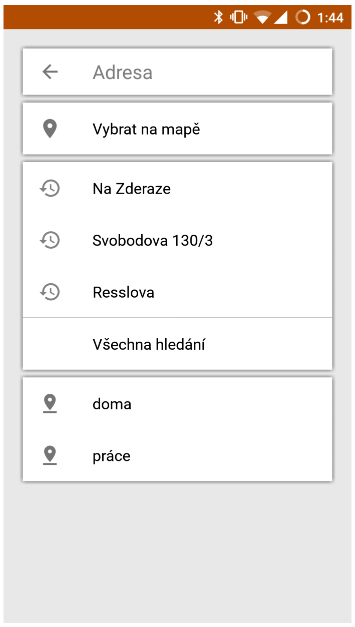


Figure 5.10: Place picker screen.

[8]https://github.com/ACRA/acralyzer
[9]http://www.cloudant.com/

Chapter 6

Iterative Development and Testing

This chapter describes the testing in general and the process of the development of the app from the prototype to the public release.

6.1 Usability Testing

There are many types of usability testing, but C. Barnum [5] focuses mainly on usability testing that observes real users – participants. She divides this testing into two categories – *formative testing* – testing done during development to fix problems, and *summative testing* – done at the end of development to confirm that the product meets requirements. These categories can be further divided by techniques with different equipment and time requirements.

- Lab testing is testing in lab, which has best observation conditions but it is time consuming and its setup is artificial for the user.
- Field testing is done on site, with worse observation conditions but it is closer to real product usage.
- Remote testing is done remotely from participant, as moderated with moderator watching in real time, or as unmoderated by the application. This type of testing is less time consuming for participant, so it can reach highly specialized participants or large number of participants.

Closely related to usability testing is concept of participatory design – potential users are asked to review a product in development.

6.2 Development Iterations and Testing

In this sections process of app development and testing is described. The iterations started with a prototype, then features were gradually added and tested by invited users in private alpha. Finally the app was released to the public.

6.2.1 Prototype

For early formative testing, a higher-fidelity horizontal prototype was created [9]. There are many tools helping the designers to create these prototypes as interactive, e.g., UXPin[1], Fluid UI[2] or Invision[3]. Most tools create the prototype with drag and drop editor from included GUI components of target system. At time of creation, most tools didn't offer components from Android 5.0. This made the Invision web application the best tool available, because it provides a strong commenting feature and allows prototype creation by linking raster images of each screen.

The interactive prototype can by found at address <`http://invis.io/5Y1S7H3QW`>. It can be viewed in a web browser on an Android device to simulate the use of the app on the target device. When viewed in a desktop browser, it allows commenting from users participating in the design. Prototype screens are also included as Figure C.1 and Figure C.2.

The comments by experts from AUTO*MAT gathered from the prototype revealed some problems with interface such as a confusing start of navigation or incorrectly placed sending form. This feedback was used during further development of the app.

6.2.2 Internal Testing

Initial versions of the app were first iterations that could be reasonably tested while riding bicycle. During these iterations the functions were gradually added and the location providers were tested and compared. The app was also tested by first volunteers to cover main Android OS versions and different settings by different users.

6.2.3 Private Alpha

Preview version of the app was tested using invite only alpha release using Google Play store[4]. Responses from this iteration were gathered through in-app feedback form and through pre-filled crash reporting emails offering user to write additional information. This approach was chosen to let presumably more experienced users in private alpha to create more specific report. This version was also used for informal field testing where three users were observed doing basic tasks.

This iteration included about 50 users and found variety of usability problems. The original maps were illegible for some users and some features such as city selection or plan profile selection were confusing. These problems were fixed by automatic selection, moving some GUI elements where users were looking for them during field testing and adding hints for other functionality.

[1] `http://www.uxpin.com/`
[2] `https://www.fluidui.com/`
[3] `http://www.invisionapp.com/`
[4] `https://play.google.com/apps`

6.2.4 Public Release

For public release, the email feedback was replaced by silent reporting using ACRA and displaying dialog allowing user to write additional description through Google Play store Crashes & ANRs functionality. The silent reporting proved to be a very successful feature, while only one user reported crash trough Google Play store, the automatic reports showed many problems that were device specific so these reports allowed to fix them even with limited variety of devices available to the developer.

Chapter 7

Practical Evaluation

In this chapter the data about usage by real users are shown and the raw data about tracked journeys are previewed.

The data for the evaluation were gathered in period from May 1, 2015 to May 21, 2015. The app was released to public on May 4 concurrently with beginning of "Do práce na kole" competition. It was then featured on the competition website on the evening of May 4 and in a email newsletter on May 6. The data are sourced from the backend log of requests, from the feedback database, and from the Developer Console on Google Play.

During this period 1118 users downloaded the app, requested 1949 routes and tracked 1557 journeys. The requests by day are shown in Figure 7.1. From the number of the requests almost 22% had either the origin or the destination outside of the urban and suburban areas where routes can be planned by the backend. This might be explained by users looking for recreational routes, even through the app is clearly aimed at the utility cycling and covers the metropolitan areas around the cities and towns. Figure 7.2 shows the number of successfully planned routes in each area. The most routes were planned in Prague where the competition originated and that is the most populated of the areas.

The tracked journeys were gathered from the users that send them with the feedback feature and from the users that opted for automatic sending of anonymised routes during the first start of the app. From the total of 1557 journeys 172 were sent manually and 1385 automatically. In Figure 7.3 the significant drop is visible during the weekends and the state holidays. This can be attributed to fact that most users are also participating in "Do práce na kole" competition that tracks the commutes to and from work. The comparison between this drop and much smaller drop in number of planned routes is shown in Figure 7.4.

The navigation was used during 128 of tracked journeys. This is amounts to about 9% of successfully planned routes. This low rate can be attributed to these factors – the users just trying out the planner, the users that do not own phone holder for their bike and are content with just the route without navigation, and the fact that while the backend logs all the requests, the tracked journeys is gathered only from the volunteers.

The tracked journey data are represented by 678,123 geographic coordinates with timestamps grouped by the corresponding journey. This raw data should be cleaned and represented in better form such as density values on a road network graph. Such representation could be used in the improvement of the planner. One of the reasons why the clean-up is required is visible in Figure 7.5 – a journey from Prague to South Bohemia by a user in a car. The representation of the data is clearly visible in Figure 7.7. Although not optimal for computer processing, this simple visualization still shows the most popular streets and the preference regarding the bridges. Finally, the journeys in Prague are shown in Figure 7.6.

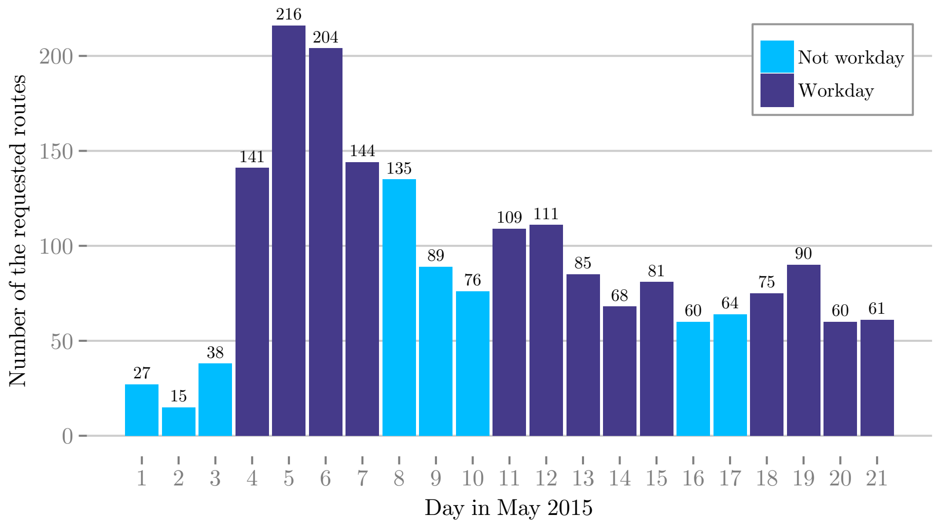


Figure 7.1: Requested routes in May 2015

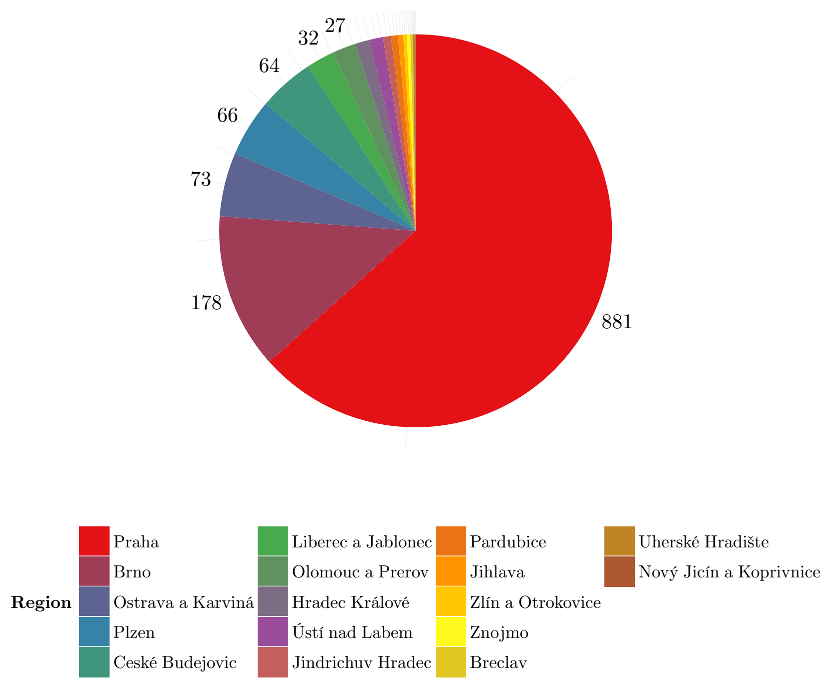


Figure 7.2: Regions where routes were requested in May 2015

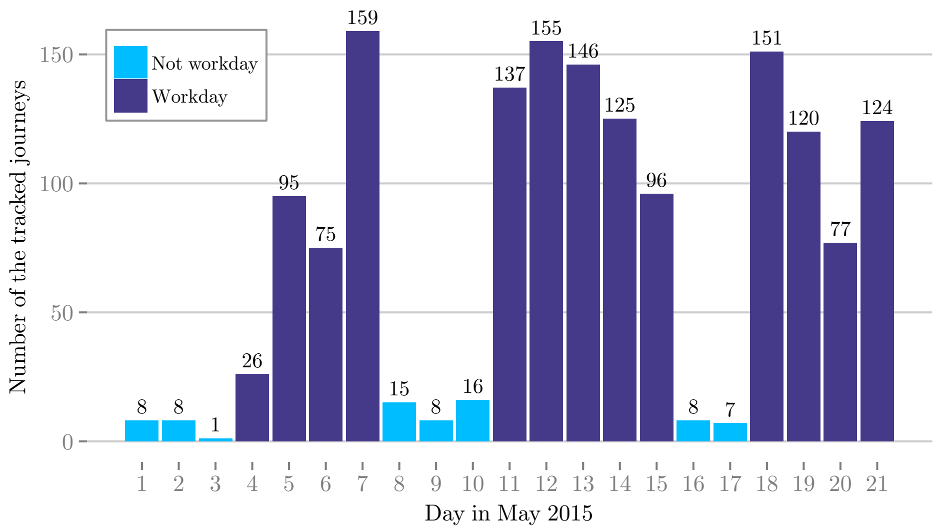


Figure 7.3: Tracked journeys in May 2015

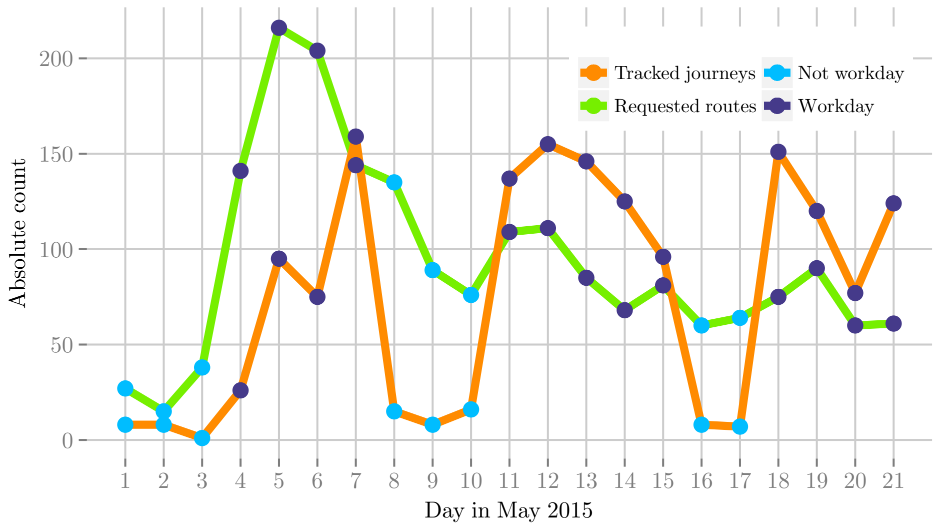


Figure 7.4: Comparison between the number of requested routes and tracked journeys in May 2015

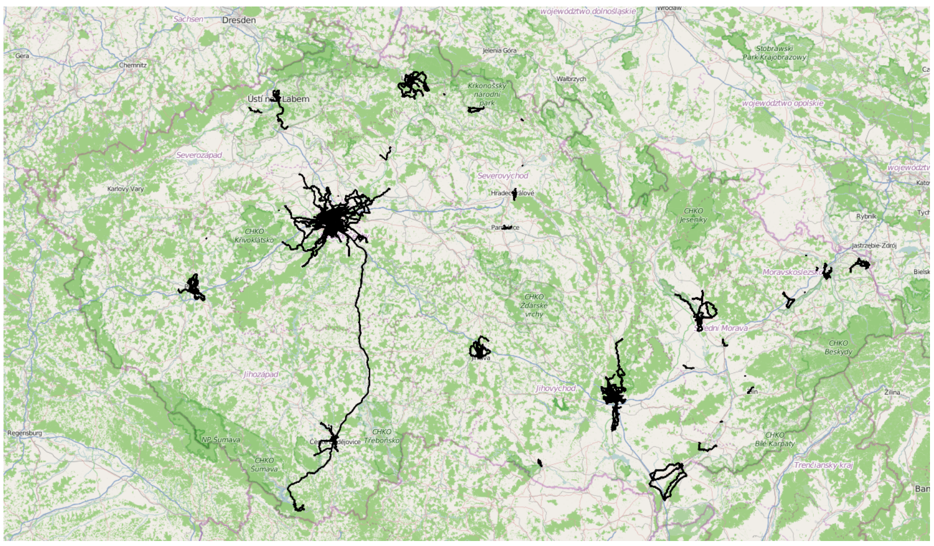


Figure 7.5: Preview of the tracked journeys – Czech Republic

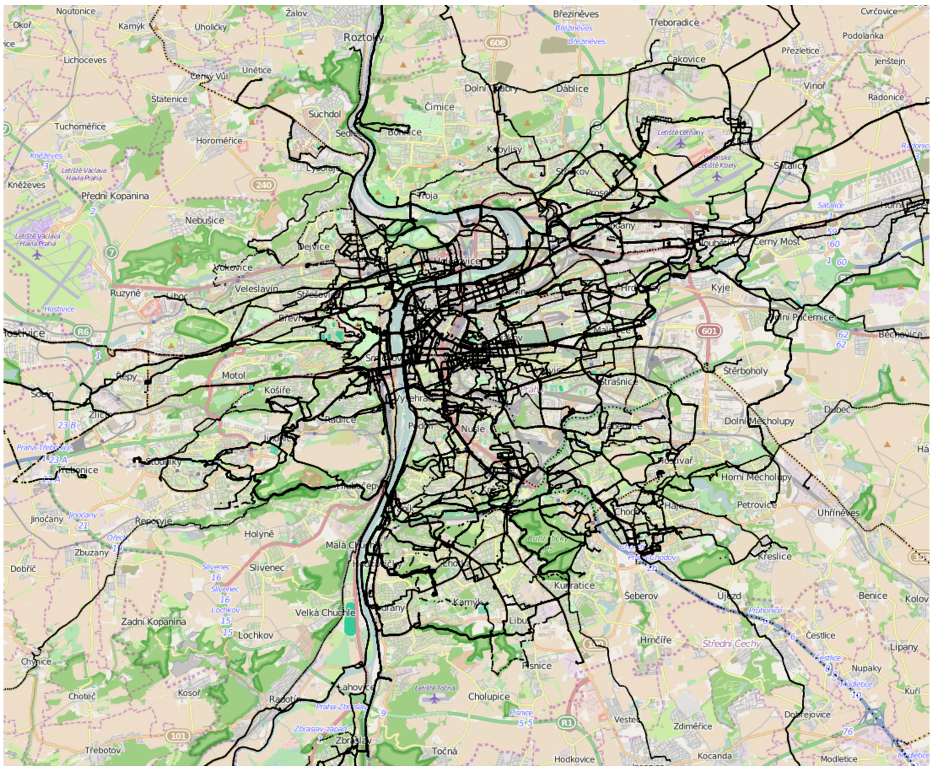


Figure 7.6: Preview of the tracked journeys – Prague

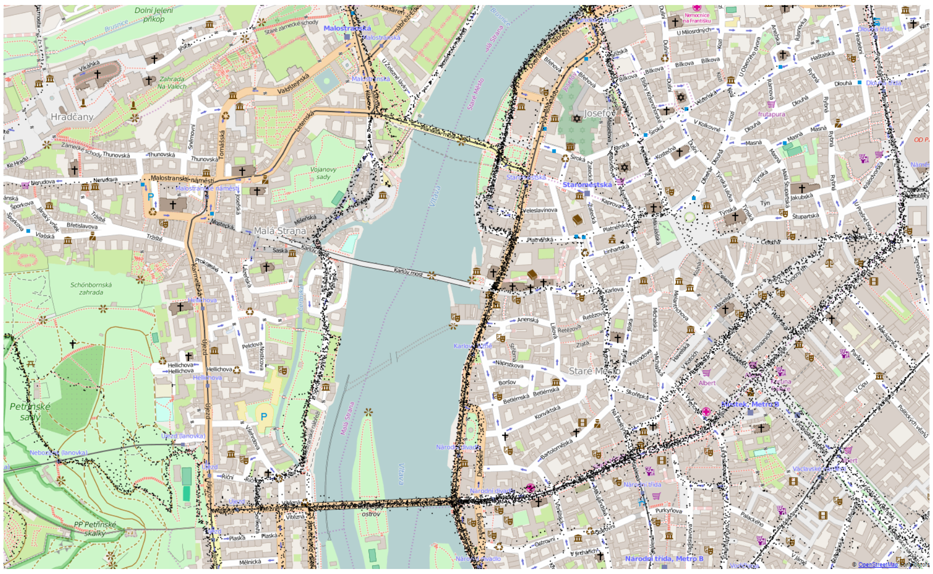


Figure 7.7: Preview of the tracked journeys – a detail from Prague

Chapter 8

Conclusion

The primary aim of the project was to build an Android app that offers bicycle route planning, navigation, and route tracking.

The final app offers four different routes based on user-submitted origin, destination, and also via points. The routes reflect different user preferences with focus on commuting, bike friendliness, flatness, or just speed. The app works in 26 Czech cities and towns. Origin, destination, and via points can be set from map, by street address search, from saved places, or from search history.

The chosen plan is then used for navigation. The app navigates the user by showing the map and the instructions. The map also displays the plan and the currently tracked journey and can be switched between tilted and top view. The user is also instructed to continue or to turn before each manoeuvre. Additional values displayed are current speed and distance and ride time left to destination. The app can also simply track a journey and display map and current distance and ride time from the origin.

Journeys from both tracking and navigation modes are saved and can be displayed with statistics about them. Each journey can be uploaded to "Do práce na kole" 2015 competition and voluntarily sent in an anonymised form to the backend to improve future plans and infrastructure for cyclists. The app was tested in development iterations from prototype, through closed betas, and then publicly released. It was featured on "Do práce na kole" 2015 competition website and newsletter. Thanks to this, the app was downloaded over 1000 times in first two weeks of the competition. Through the app, we managed to gather over 1500 cycle journeys.

The future improvements of the app can include active replanning during navigation and further improvement of tracked data. The replanning requires determining if the user left the current route. This can be difficult task because GPS accuracy does not allow to differentiate between for example road or adjacent pavement. Too aggressive full replanning would burden both battery life of the device and the backend. Additional interesting data to track could be values from device's accelerometer. These values might provide insight into riding patterns and might show other obstacles such as cobblestones that are not correctly labelled in OSM data used by the backend. Further smaller improvements such as a route sharing and summarised journey statistics are also possible to add.

Bibliography

[1] Android.com. *Android Design* [online]. 2014. [cit. 30. 11. 2014]. Accessible at: <`https://developer.android.com/training/material/compatibility.html#SupportLib`>.

[2] Android.com. *Android Developers - Reference* [online]. 2015. [cit. 11. 5. 2015]. Accessible at: <`http://developer.android.com/reference/android/location/package-summary.html`>.

[3] Android.com. *Android Platform Versions* [online]. 2015. [cit. 5. 5. 2015]. Accessible at: <`https://developer.android.com/about/dashboards/index.html#Platform`>.

[4] Android.com. *Android Design* [online]. 2014. [cit. 30. 11. 2014]. Accessible at: <`https://developer.android.com/design/material/index.html`>.

[5] BARNUM, C. *Usability Testing Essentials: Ready, Set... Test!* Chichester : Elsevier Science, 2010. ISBN 9780123785534.

[6] FILLER, V. *Dotazovací průzkum cyklistických preferencí - analýza výsledků* [online]. 2010. [cit. 7. 2. 2015]. Accessible at: <`http://prahounakole.cz/wp-content/pnk/uploads/2010/10/cyklopruzkum_5_2010_1.1.pdf`>.

[7] FRIEDMAN, J. H. – BENTLEY, J. L. – FINKEL, R. A. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.* September 1977, 3, 3, s. 209–226. ISSN 0098-3500. doi: 10.1145/355744.355745. Accessible at: <`http://doi.acm.org/10.1145/355744.355745`>.

[8] Google. *Google Guidelines* [online]. 2014. [cit. 30. 11. 2014]. Accessible at: <`http://www.google.com/design/spec/material-design/introduction.html`>.

[9] JONES, M. – MARSDEN, G. *Mobile Interaction Design.* Burlington : Wiley, 2006. ISBN 9780470090893.

[10] LEGAT, V. Mobile journey planner and navigation for cyclists. Bachelor's project, Czech Technical University in Prague, 2014.

[11] MARRA, C. – WEAVER, D. *Year class: A classification system for Android* [online]. 2014. [cit. 23. 11. 2014]. Accessible at: <`https://code.facebook.com/posts/307478339448736/year-class-a-classification-system-for-android/`>.

[12] ZILECKY, P. Junction-aware Multicriteria Bicycle Route Planning. Master's thesis, Czech Technical University in Prague, 2015 (to appear).

Appendices

Appendix A

Download / CD Content

A.1 Download Code

Figure A.1: QR code for the download of the most recent version of the app using Google Play store <`https://play.google.com/store/apps/details?id=cz.agents.cycleplanner`>.

A.2 CD Content

- `binary` – the signed APK for installation of the app.
- `data` – the data files used in evaluation in csv format
- `source` – the source code of the app, see README.txt for instructions for the build.
- `text` – the text of this bachelor's project in pdf format.

Appendix B

JSON Schema

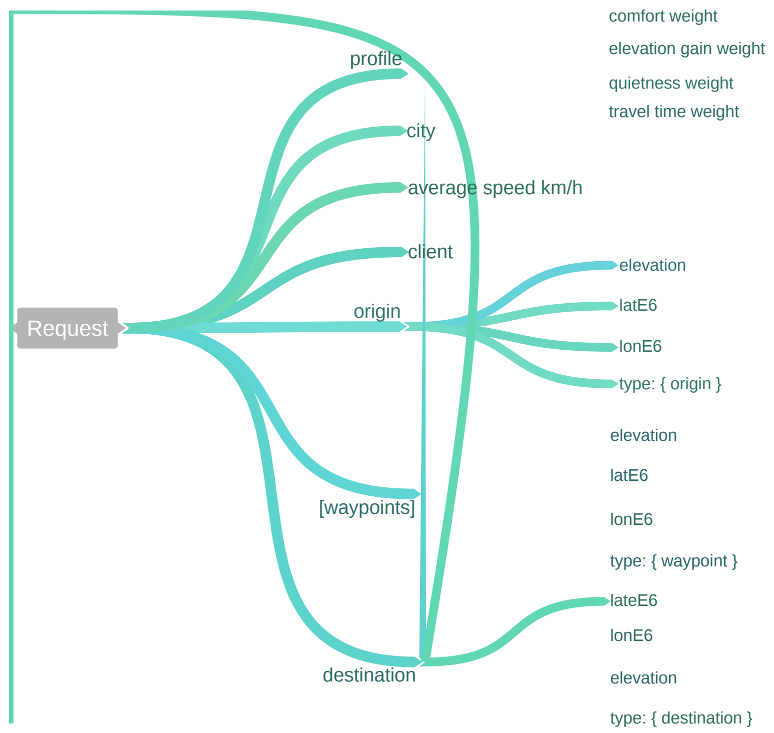


Figure B.1: JSON schema of the backend API request, source [12]

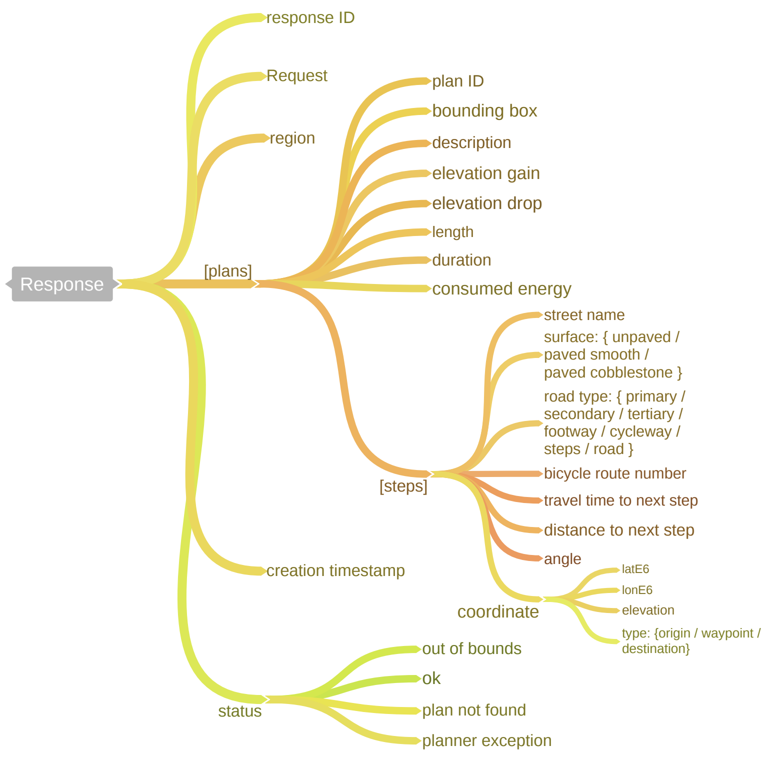


Figure B.2: JSON schema of the backend API response, source [12]

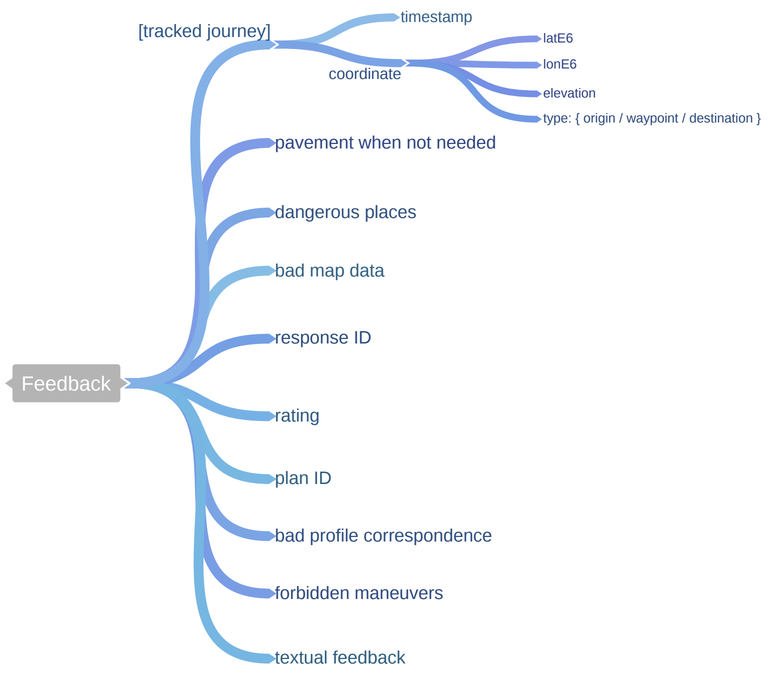


Figure B.3: JSON schema of the backend API feedback to API request, source [12]

Appendix C

The Preview of the Prototype

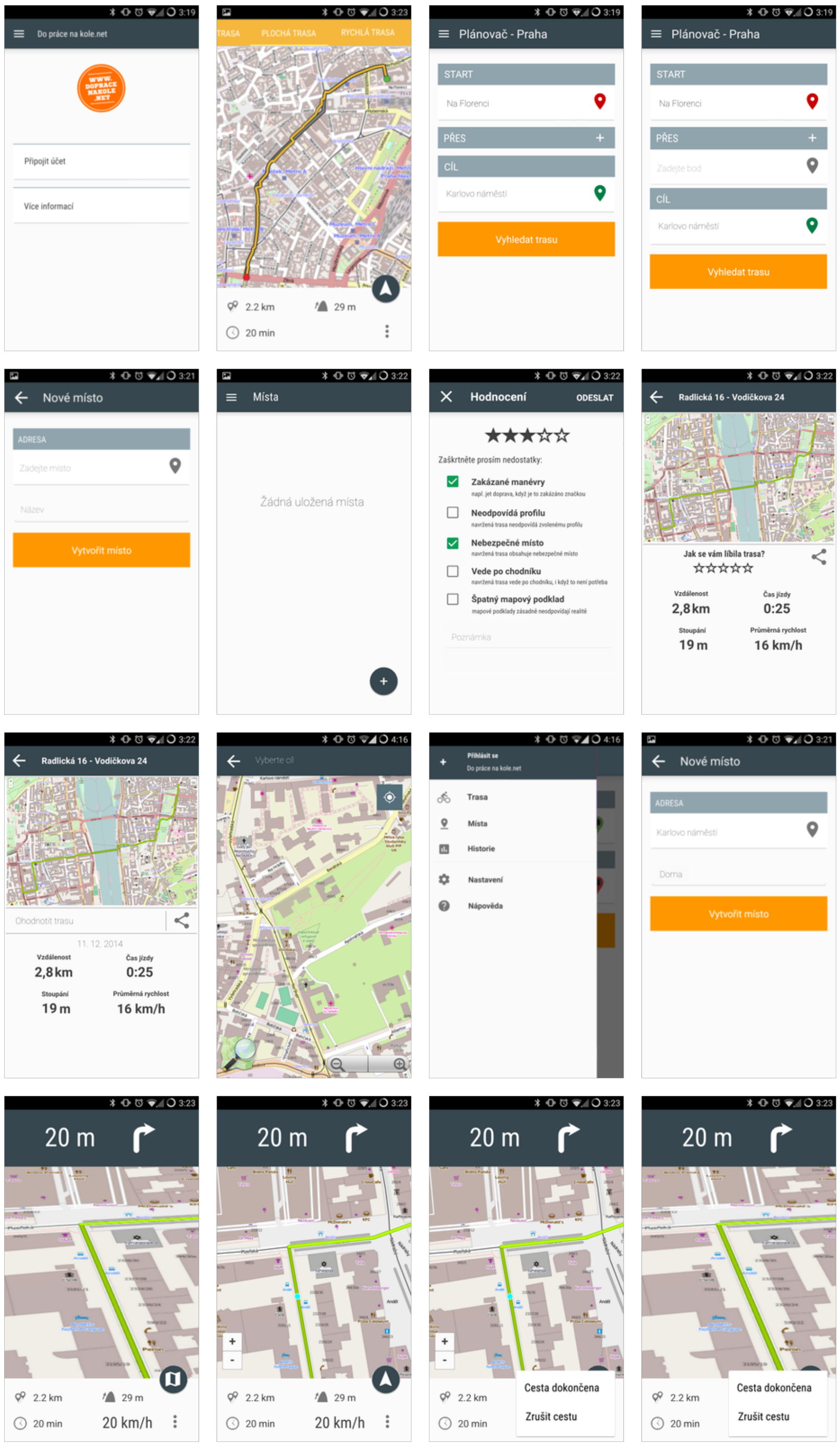


Figure C.1: Prototype screens preview 1

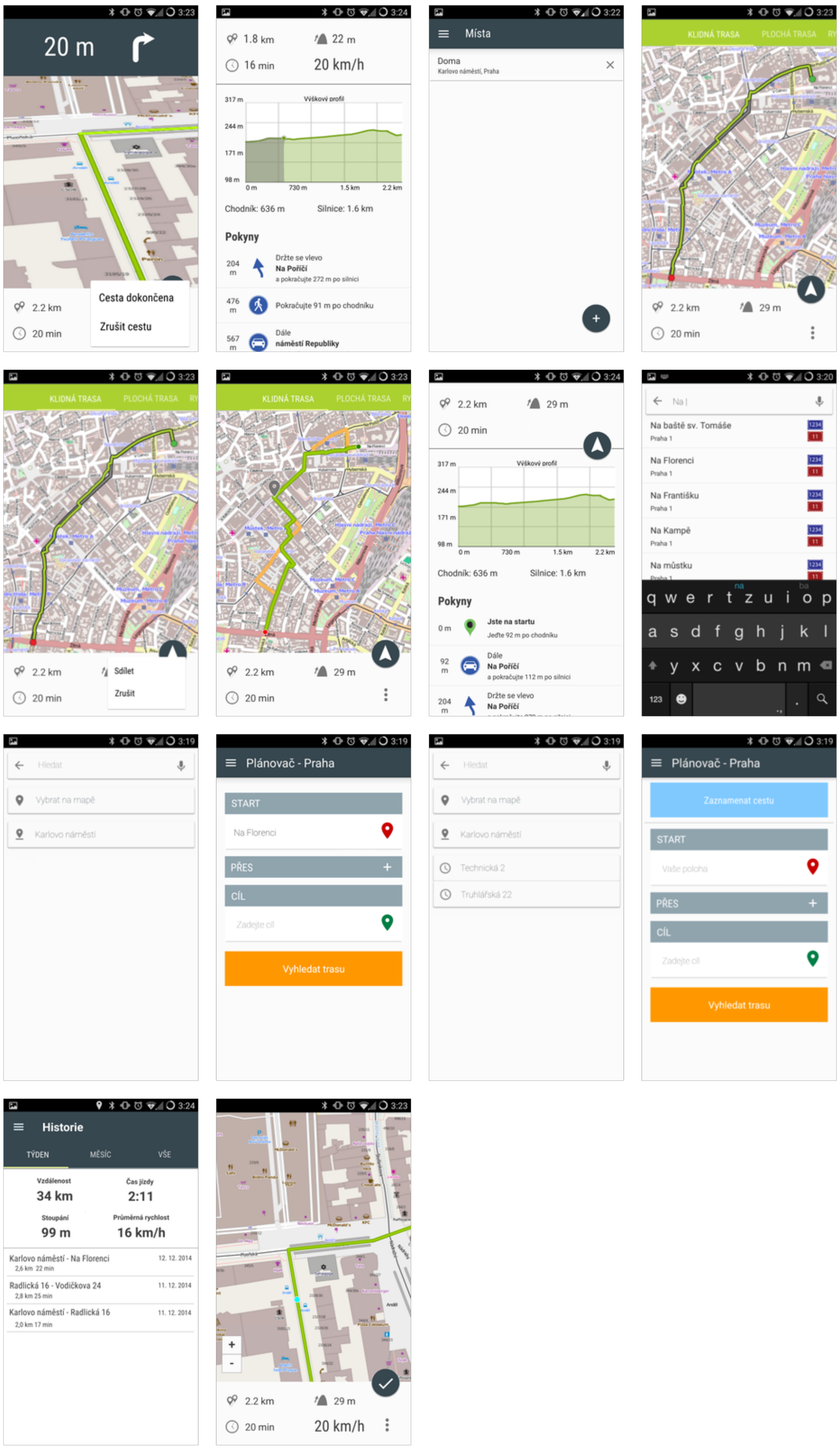


Figure C.2: Prototype screens preview 2