

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Graphics and Interaction

BACHELOR PROJECT ASSIGNMENT

Student: **Václav Legát**

Study programme: Software Engineering and Management
Specialisation: Web and multimedia

Title of Bachelor Project: **Mobile journey planner and navigation for cyclist**

Guidelines:


Design and develop a mobile application for Android smartphones capable of journey planning and navigation with focus on cyclists. During the design and analysis, take into account two target user groups (beginners and experts) and their specific requirements. Together with the supervisor, select and use suitable GIS system for route calculation and navigation. During the design and implementation process, follow User Centered Design guidelines and perform continuous formative user testing and evaluation.

Bibliography/Sources:

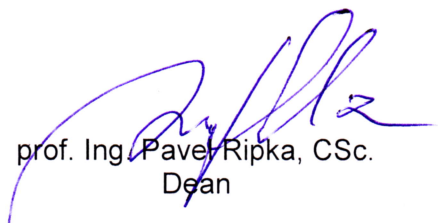
M. Jones, G. Marsden: Mobile Interaction Design, John Wiley & Sons, 2006
C. M. Barnum: Usability Testing Essentials. Ready, Set ... Test!, Elsevier - Morgan Kaufmann, 2011
G. Nudelman: Android Design Patterns: Interaction Design Solutions for Developers, Wiley, 2013

Bachelor Project Supervisor: Ing. Ivo Malý, Ph.D.

Valid until the end of the summer semester of academic year 2014/2015


prof. Ing. Jiří Žára, CSc.
Head of Department




prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 24, 2014

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction



Bachelor's Thesis

Mobile journey planner and navigation for cyclists

Václav Legát

Supervisor: Ing. Ivo Malý, Ph.D.

Study Programme: Software Technologies and Management

Field of Study: Web and Multimedia

Acknowledgement

I would like to thank Ing. Ivo Malý, Ph.D., for his helpful comments and guidance during the supervision of this thesis.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

I have no objection to usage of this work in compliance with the act § 60 Zákon č. 121/2000 Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Kladno, May 21, 2014

.....

Abstract

This Bachelor's thesis is concerned with design and implementation of a mobile application that can plan cycle journeys in Prague. The application is developed for mobile devices with Android operating system. The application uses Open Bicycle Trip Planner web planner service which searches routes according to start point, finish point and defined route profile. This web service also provides a list of instructions that are used during the navigation process. During the design process specific requirements of target users are taken into consideration. The finished application can search routes, it provides users with information about elevation profile and it also informs users about possible obstacles on the route. The application is available in Google Play store under the name City Bike Planner.

Key words

mobile application, android, cycle planner, city cycling, navigation

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací mobilní aplikace pro plánování tras určených pro cyklisty v Praze. Aplikace je vyvinuta pro mobilní zařízení s operačním systémem Android. Aplikace využívá služeb webového plánovače Open Bicycle Trip Planner, který vyhledává trasy na základě startovního a cílového bodu a definovaného profilu trasy. Tato služba navíc poskytuje sadu instrukcí, které jsou využity k navigaci při jízdě. Při návrhu aplikace jsou zohledněny požadavky cílových uživatelů. Výsledná aplikace dokáže vyhledávat trasy, poskytuje uživatelům informace o převýšení trasy a také informuje o překážkách, které se mohou na trase objevit. Aplikace je volně dostupná v obchodě Google Play pod názvem City Bike Planner.

Klíčová slova

mobilní aplikace, android, cyklistický plánovač, městská cyklistika, navigace

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	2
1.3	Structure	2
2	Analysis	3
2.1	Analysis of target groups of users	3
2.1.1	Beginner cyclists	3
2.1.2	Expert cyclists	4
2.2	Overview of required application's features	4
2.3	Review of Map SDKs	5
2.3.1	Google Maps Android API V2	5
2.3.2	MapQuest	6
2.3.3	Mapsforge	7
2.3.4	Nutiteq	7
2.3.5	OsmDroid + OSMBonusPack	7
2.3.6	Evaluation of Map SDKs	8
2.4	Review of available applications	9
2.4.1	Cyclestreets	9
2.4.2	OsmAnd	9
2.4.3	BikeCityGuide	10
2.4.4	Comparison of reviewed applications	11
2.5	Open Bicycle Trip Planner	11
2.6	City Bike Planner requirements	12
2.6.1	Functional requirements	12
2.6.2	Non-functional requirements	12
3	Design	13
3.1	Navigation between screens	13
3.2	Use cases	14
3.2.1	Route planning	14
3.2.2	Managing places	15
3.2.3	Managing routes	15
3.2.4	Managing navigation	16
3.3	GUI Design	17
3.3.1	Planner UI design	17
3.3.2	Map UI design	18
3.3.3	Route UI Design	19
3.3.4	Places and routes UI design	22
3.3.5	Navigation UI Design	23

4	Implementation	25
4.1	Places data preparation	25
4.2	Databases	25
4.3	Implementation structure	27
4.3.1	Activities and fragments	27
4.3.2	Adapters	28
4.3.3	Entities	28
4.3.4	Helpers	29
4.3.5	Location	29
4.3.6	Mock	29
4.3.7	Providers	29
4.3.8	Routing	30
4.3.9	Utils	30
4.4	Implementation details	30
4.4.1	Geocoding	30
4.4.2	Routing	31
4.4.3	Navigation	33
4.5	Graphics	33
4.6	Limits of the application	34
5	Evaluation	35
5.1	Prototype testing	35
5.2	Application testing	36
5.2.1	Results of the testing	39
6	Conclusion	41
	References	43
	Appendix A Contents of the CD	45
	Appendix B Prototype questionnaire	47
	Appendix C Testing tasks	49

List of Figures

2.1	Displaying marker and info window with Google Maps Android API V2 . . .	6
2.2	Example of data returned by Guidance API	6
2.3	Route with directions from MapQuest in Nutiteq sample application	7
2.4	Example of OSMBonusPack features	8
2.5	Cyclestreets user interface	9
2.6	OsmAnd user interface	10
2.7	BikeCityGuide user interface	10
3.1	Scheme of navigation between screens	13
3.2	Route planning use cases diagram	14
3.3	Managing places use cases diagram	15
3.4	Managing routes use cases diagram	16
3.5	Managing navigation use cases diagram	16
3.6	Planner screen interface design	18
3.7	Map screen interface design	19
3.8	Prototype route and itinerary screen design	20
3.9	Route screen interface design	21
3.10	Places and routes interface design	22
3.11	Navigation screen interface design	23
4.1	Drawer activity with navigation drawer	27
4.2	Open Bicycle Trip Planner web interface	31
4.3	Examples of used icons	33
4.4	Example of map data without cycleway	34

List of Tables

2.1	Comparison of Map SDKs for Android	8
2.2	Comparison of reviewed applications	11

List of Abbreviations

API	Application Programming Interface
CRUD	Create Read Update Delete
FRQ	Functional Requirements
JSON	JavaScript Object Notation
NRQ	Non-Functional Requirements
OBTP	Open Bicycle Trip Planner
OSM	Open Street Map
POI	Point Of Interest
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
UC	Use Case

Chapter 1

Introduction

1.1 Motivation

Today city cycling in Prague is becoming a better alternative to using a car or public transport. Compared to these motorised means of transport cycling brings more benefits but some disadvantages may be also found.

On the one hand cyclists are not limited by waiting in traffic jams which is a very common phenomenon in cities nowadays. Cyclists do not have to wait at stops for a bus or tram to arrive and they do not have to change lines. Purchasing a bicycle is a very cheap option in comparison with buying a car and it applies to its operational expenses as well. Cyclists do not have to search for an empty spot on parking lots which primarily the centre of the city has lack of. Besides that cycling is very friendly to the environment; it produces no greenhouse gas. It also helps to lower noise pollution in the city. Lastly cycling is good for people's well being. It helps to relax and lower the stress level. In addition, people might feel better if they know they are doing something for their body.

On the other hand cycling has some disadvantages too. Cyclists are endangered by cars while riding on busy city streets. Cycling is a physical activity so if people go to work on a bicycle they probably get sweaty. That is not pleasant if they have to work for another eight hours.

The number of people using bicycle for transport around the city is constantly growing. According to a survey¹ conducted in September 2012, the number of cyclists in Prague has doubled in comparison with 2010. So there are 120,000 cyclists riding a bike on regular basis in Prague.

It is important to provide an easy transition between using motorised means of transport and bicycle for new cyclists. Beginners need to find a way how to move around the city, which includes finding the right route for their journey. It is also important that they use routes that are not difficult, routes that are not dangerous and routes that lead them to their final destination with the least effort. For skilled cyclists the city should be presented from another point of view – showing them various options of routes that they may otherwise overlook and that would be more favourable for them.

¹Auto*Mat. Průzkum cyklistických preferencí ze září 2012. 2013

1.2. OBJECTIVES

1.2 Objectives

The objective of this project is to design and implement an application that can plan routes for bicycle journeys and then navigate users on these journeys. I chose “City Bike Planner” as the name of the application. The application will be developed for Android platform and will be available for download in Google Play store².

Routing service and navigation instructions will be provided by Open Bicycle Trip Planner³ (OBTP). The application will allow users to create and save places which will speed up the process of defining “from” and “to” parameters. Navigation will also be a very important part of the application and will help users during their journey and notify them of any possible difficulties on the route.

1.3 Structure

The bachelor’s thesis is comprised of six chapters. In chapter 2 I analyse cyclists in Prague and their requirements, features that are required by the application and three applications for mobile phones that try to deal with problematics of route planning for cyclists. In chapter 3 I describe use cases of the application and design of the application’s user interface. In chapter 4 I describe the implementation process and structure of the application. In chapter 5 the results from user testing are provided. In chapter 6 I discuss the results of the project and future improvements.

²<https://play.google.com/store>

³<https://bitbucket.org/mnemet/open-bicycle-trip-planner/wiki/Home>

Chapter 2

Analysis

In this chapter I will analyse target user groups and their requirements first. Then I will describe general requirements of map and location aware applications along with five SDKs for developing map applications on Android. I will provide description of three mobile applications available for cyclists. At the end of the chapter I will describe OBTP, which will be used as the routing service and functional and non-functional requirements of City Bike Planner.

2.1 Analysis of target groups of users

According to the survey conducted in 2012¹ over 190,000 cyclists ride at least once a week. Daily it is around 24,000 cyclists. Major group of 134,000 cyclists is formed by recreational cyclists, the second major group is formed by traffic cyclists. Over 50 % of cyclists are 15–19 years old. A little over 25 % are cyclists over 40. Women form only one fifth of cyclists.

Target users of the application can be divided into two groups according to their experience. The first group is formed by beginners and less experienced cyclists. The second group is formed by more experienced cyclists and expert cyclists. In the following section I will describe these two groups and their requirements. The following information is taken from the survey of cyclists preferences² conducted in 2010.

2.1.1 Beginner cyclists

For beginner cyclists the main priority is safety and comfort of their route therefore it is important to them to use routes where there are no obstacles or difficulties. The distance of route they are willing to commute is around 8 kilometres with maximum elevation around 77 metres. The survey further says that beginner cyclists would be riding more frequently if obstacles on their route were removed. The interesting fact is that both groups of cyclists do not mind time delay in comparison with other means of transport. Beginner cyclists prefer to use cycleways and cycle lanes to main streets. As the most difficult obstacles they state riding in traffic jam, turning left, riding through difficult crossroads, steep ascending and cobblestone terrain.

¹Auto*Mat. Průzkum cyklistických preferencí ze září 2012. 2013

²Filler, V. Dotazovací průzkum cyklistických preferencí – analýza výsledků. 2010.

2.2. OVERVIEW OF REQUIRED APPLICATION'S FEATURES

The application could provide an easy way of finding the right route. Cyclist can find comfortable and safe routes that would help them in their first steps and encourage them to become regular cyclists.

2.1.2 Expert cyclists

Same as for beginners the main priority for expert cyclists is safety of the route but opposite to beginners they prefer quickness of the route to its comfort. The distance they are willing to commute is around 12 kilometres with maximum elevation around 100 metres. Experts prefer cycleways, cycle lanes and they do not mind main streets but they do not prefer cycleways as much as beginners. Expert cyclists state almost the same difficult obstacles as beginners. The difference is that they do not mind turning left at the crossroads and would overcome steep ascending up to 10 %.

Expert cyclists know the city well so it is possible that they do not have to use a special application to show them how to get from place A to place B. Or they use some online routing system where they find a route and later they navigate from memory. Nevertheless the application could offer them better routes, because it takes into consideration their preferences.

2.2 Overview of required application's features

This section provides a description of features required by map and location aware applications which will be also used in the City Bike Planner application.

Map rendering There are two possible methods how to render a map base.

The first method uses prerendered map tiles that are downloaded only when they are needed. This method is limited by internet connection. Internet connection must be available and it has to provide download speed which is fast enough. Some APIs allow map tiles caching so they do not have to be downloaded every time they are needed.

The second method uses vector map. This method is limited by the size of the vector map. Mapsforge project provides vector map of the Czech Republic that has 196 MB. This method is further limited by rendering capabilities of mobile devices because the map has to be rendered every time from scratch.

Overlays Map API has to provide means to enhance map with other graphical elements known as overlays. For example marker overlays indicate individual places on map or polyline overlays that are used to display route.

Geocoding Geocoding is one of the main processes used in many location aware applications. It converts addresses like “Technická 2, Praha 6, Česká republika” into geographic coordinates which consist of latitude and longitude values. These values are used for placing markers or drawing lines and other graphical elements onto the map.

Complementary process to geocoding is reverse geocoding which converts latitude and longitude values into human readable addresses.

Routing Routing is a process of finding a path between two nodes in a graph. In most of the route finding applications routing is used to find a route between two or multiple

places. Often different modes of routing can be set to adjust users' preferences such as pedestrian, bicycle or car mode.

Navigation Navigation or more specifically turn by turn navigation is used to guide users on their route. Users are usually informed about the next change of direction so they know where they have to turn. Further they can be warned against possible dangers or obstacles on the route.

2.3 Review of Map SDKs

In the following section I will analyse available SDKs for developing Android applications with maps. I chose five SDKs for the analysis. The main SDK for working with maps is Google Maps Android API V2. Most of the other SDKs are trying to substitute older version Google Maps Android API V1.

2.3.1 Google Maps Android API V2

Google Maps Android API V2³ enables developers to add a Google map to their application. At the beginning developers have to register their project in the Google APIs Console. After the registration developers can create API key which is needed in order to use Google maps and other Google APIs. Furthermore they have to download the Google Play services which contain the Google Maps API. Google Maps API uses OpenGL for rendering the map base and only Google Maps can be used as the base. Maps can be included as a `MapView` or as a `MapFragment`. More than one map can be displayed on the screen when using `MapFragment`.

Main features:

- Drawing Markers, Polylines, Polygons, Ground and Tile Overlays on the map.
- Interacting with the map by Zooming, Panning, Rotating and Tilting.

Other APIs that can be used together with Maps API:

Location API

- Determining the device location and listening for its changes.
- Determining the type of transport such as walking, cycling, driving.
- Creating and monitoring predefined geographical regions.
- Geocoding and reverse geocoding.

Directions API

- Calculating directions between locations.
- Providing waypoints for routing (number of waypoints is limited to 8).
- Searching for directions for transit, driving, walking or cycling.

³<https://developers.google.com/maps/documentation/android/>

2.3. REVIEW OF MAP SDKS

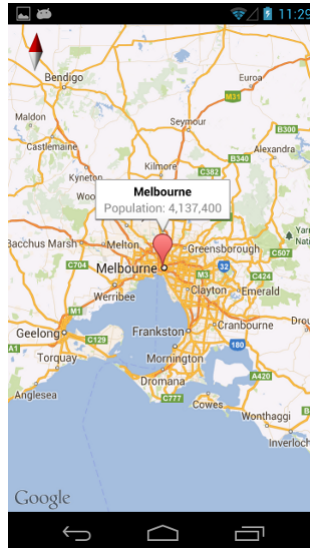


Figure 2.1: Displaying marker and info window with Google Maps Android API V2

2.3.2 MapQuest

Mapquest⁴ provides basic functions such as raster map tile rendering and overlays drawing. It also provides access to multiple MapQuest Open services:

- Guidance API Web Service allows developers to request guidance route information between two or more points. Several modes for routing can be set, such as fastest, shortest or bicycle.
- Geocoding Service allows developers to request latitude and longitude parameters for an address and vice versa.
- Nominatim Search Service allows developers to find different types of POIs in a pre-defined radius or corridor. It uses OpenStreetMap data for place searching.

Guidance Route Data		Narrative
ManeuverType: STRAIGHT	Link: Jonestown Rd	Go southwest on Jonestown Rd
ManeuverType: RIGHT	Link: Lincoln School Rd	Turn right on Lincoln School Rd
ManeuverType: RIGHT	Link: US-22	Turn right on US-22/William Penn Hwy
ManeuverType: SLIGHT_RIGHT	Link: NO ROAD NAME	Make a slight right on ramp
ManeuverType: MERGE_LEFT	Link: PA-72 N	Merge left onto PA-72 N
ManeuverType: LEFT	Link: PA-443	Turn left on PA-443/Moonshine Rd
ManeuverType: DESTINATION	Link: PA-443	Arrive at GREEN POINT, PA

Figure 2.2: Example of data returned by Guidance API

⁴<http://developer.mapquest.com/web/products/featured/android-maps-api>

2.3.3 Mapsforge

MapsForge⁵ provides a free map library. It uses vector map data so they can be accessed offline. These maps can be also styled. Vector map data for continents or individual countries can be downloaded from MapsForge server. Custom map layer can be also created⁶. Other features such as drawing markers and lines or interaction with map are very limited. Routing and geocoding are not implemented so third party libraries have to be used.

2.3.4 Nutiteq

Nutiteq⁷ is a good replacement for Google Maps API. Almost every feature in Google Maps API is also provided by Nutiteq. In addition, both raster and vector map layer can be used as the map base. MapQuest can be used for routing. Geocoding feature is not implemented but online services provided by MapQuest can be used.

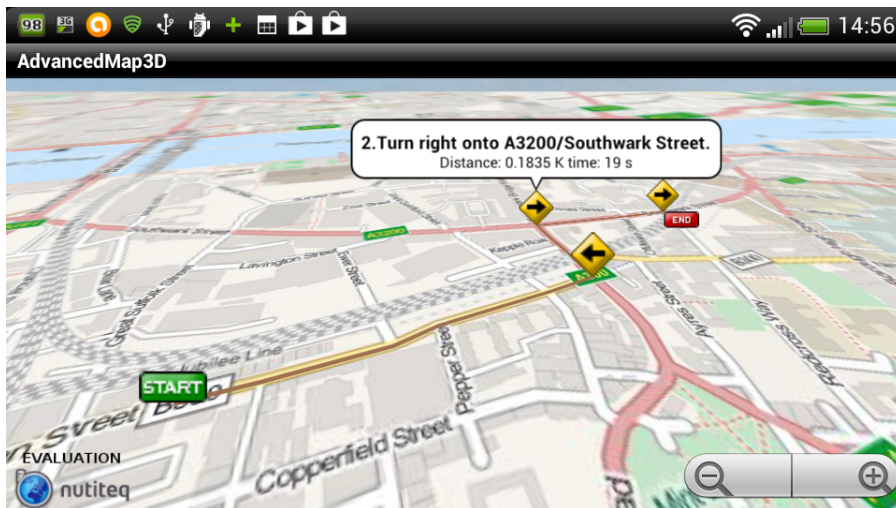


Figure 2.3: Route with directions from MapQuest in Nutiteq sample application

2.3.5 OsmDroid + OSMBonusPack

OsmDroid⁸ is a free library which represents almost full replacement of older version of Google Maps Android API V1. It works with OpenStreetMap data. Only slf4j-android logging library is required for proper working. These basic functions are extended by OSMBonusPack library⁹.

MapQuest service can be used for routing but developers need to register and obtain API key. Waypoints can be also included in route request. Route is returned in xml format and contains directions that can be used for navigation. Library contains an xml parser which transforms xml response into simple java object. Geocoding and reverse geocoding use OpenStreetMap POIs with Nominatim.

⁵<https://code.google.com/p/mapsforge/>

⁶<http://extract.bbbike.org/>

⁷<http://www.nutiteq.com/>

⁸<https://code.google.com/p/osmdroid/>

⁹<https://code.google.com/p/osmbonuspack/>

2.3. REVIEW OF MAP SDKS



Figure 2.4: Example of OSMBonusPack features

2.3.6 Evaluation of Map SDKs

In this section I provide a comparison of Android map SDKs showing the features required by location aware applications.

Table 2.1: Comparison of Map SDKs for Android

	Google Maps	MapQuest	Mapsforge	Nutiteq	OsmDroid
Map rendering	vector	raster	vector	raster, vector	raster
Overlays	yes	yes	yes	yes	yes
Geocoding	yes	yes	no*	no*	yes
Routing	yes	yes	no*	yes	yes
Directions	yes	yes	no*	yes	yes

* a third party library have to be used

I implemented the basic application with all of these Map SDKs. In all of them I came across bugs such as flickering map base or non-functional event listeners. In my opinion the best SDK is Google Maps Android API V2. Although it uses Google Maps so the internet connection have to available it provides best interface for working with map such as map rotation and map tilt important for navigation. In comparison with other map SDKs, Google Maps API contains less bugs and is the most stable SDK of those previously reviewed.

2.4 Review of available applications

In this section three journey planner applications for cyclists will be discussed. All of these applications use free OpenStreetMap¹⁰ data for routing.

2.4.1 Cyclestreets

Cyclestreets¹¹ is a mobile version of a journey planner for cyclists. It can plan routes including waypoints. Routes are obtained from the Cyclestreets website. It uses four modes for routing: quietest, balanced, fastest and shortest. It provides a turn-by-turn itinerary. It has a feature for uploading photos from routes to point out problems or good practice. These photos are displayed on a photomap which is another feature of the application. The application is intended for users in the United Kingdom but map data¹² for Prague are also available for download in Google Play.

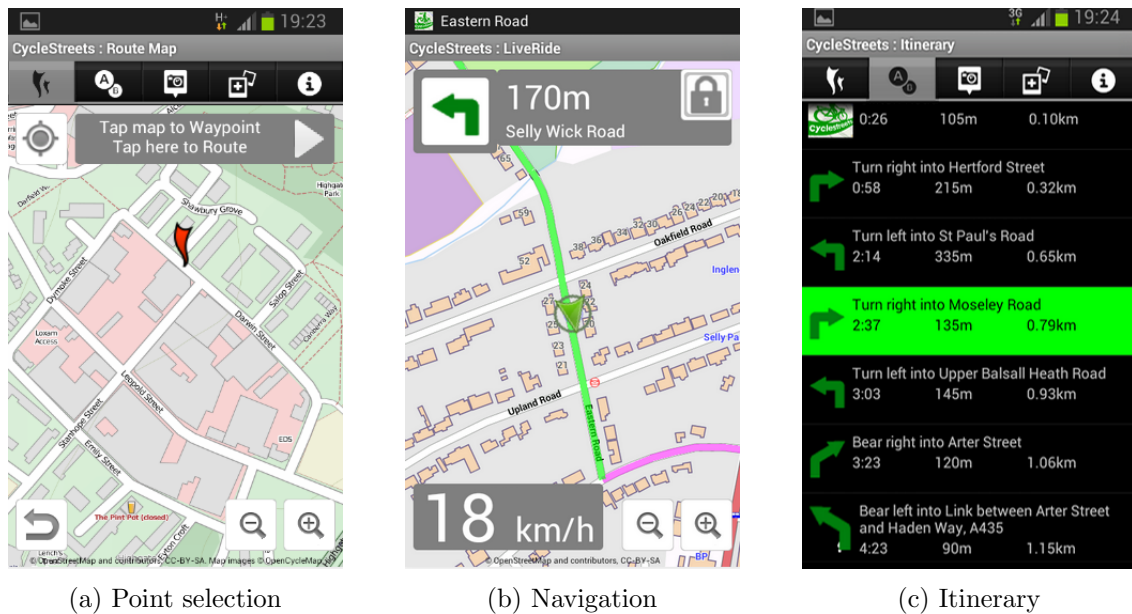


Figure 2.5: Cyclestreets user interface

2.4.2 OsmAnd

OsmAnd¹³ is an open source navigation system. The application is not designed primarily for cycle route planning. It also provides route planning and navigation for cars and pedestrians. It works both online and offline. There are two versions of the application. The free version contains free world vector maps; the number of downloadable detailed regional maps is limited to ten. The paid version offers unlimited map downloads. Moreover, it provides access to offline Wikipedia POI database.

¹⁰<http://www.openstreetmap.org/>

¹¹<https://play.google.com/store/apps/details?id=net.cyclestreets&hl=cs>

¹²<https://play.google.com/store/apps/details?id=net.cyclestreets.maps.czprague&hl=cs>

¹³<https://play.google.com/store/apps/details?id=net.osmand&hl=cs>

2.4. REVIEW OF AVAILABLE APPLICATIONS

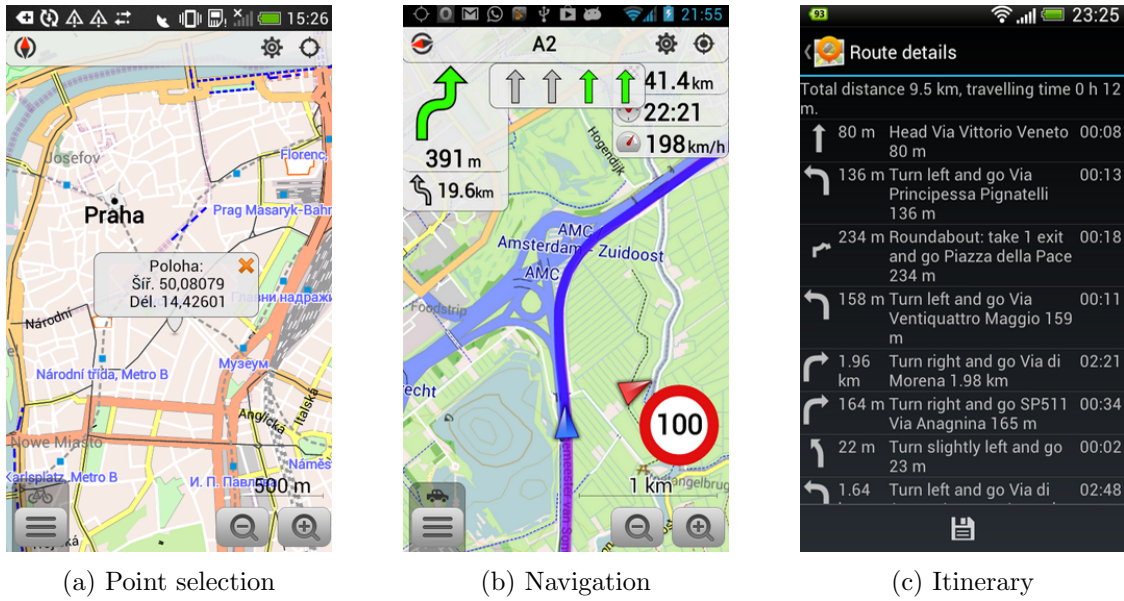


Figure 2.6: OsmAnd user interface

2.4.3 BikeCityGuide

BikeCityGuide¹⁴ is a navigation system designed specifically for city cyclists. During routing the application prefers side streets and cycleways to busy main streets. The route planning and navigation system work both online and offline. It uses vocal instructions during the turn-by-turn navigation which enables safer and comfortable cycling. A big part of the application is a tour guide. Each city has pre-assembled tours which lead around their sights and other interesting places. It works only for some European countries. The available countries are Austria, Belgium, France, Germany, the Netherlands, Spain, and Switzerland. Moreover, only few cities from each country are supported and a fee is charged for the individual city data.

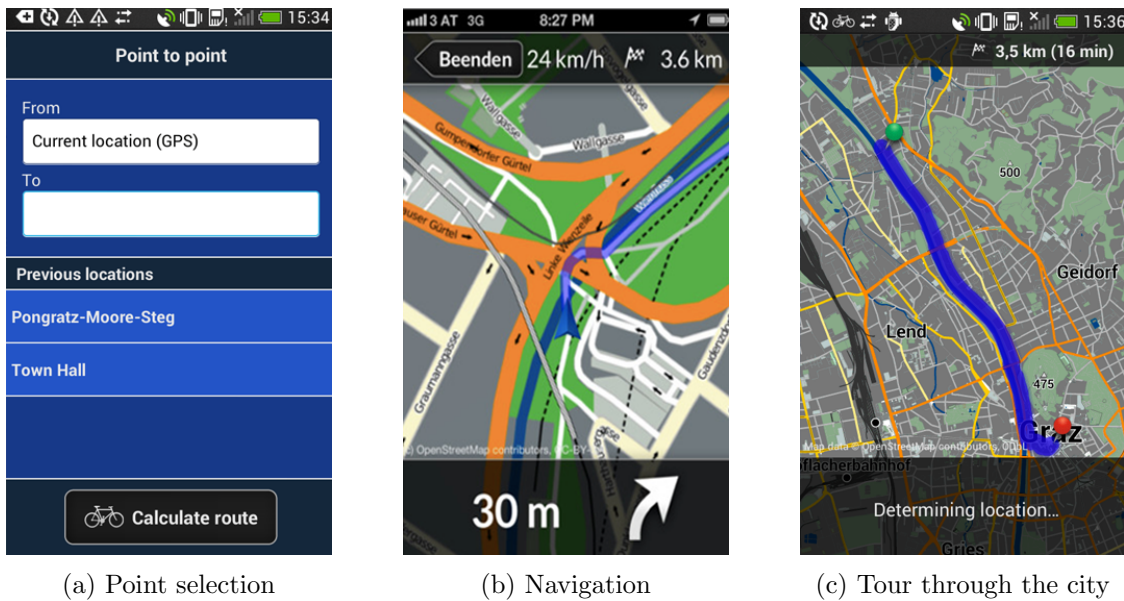


Figure 2.7: BikeCityGuide user interface

¹⁴<https://play.google.com/store/apps/details?id=org.bikecityguide&hl=cs>

2.4.4 Comparison of reviewed applications

In this section I provide a comparison table showing the features available in the reviewed applications.

Table 2.2: Comparison of reviewed applications

	CycleStreets	OsmAnd	BikeCityGuide
Primarily for cyclists	yes	no	yes
Raster map (online)	yes	yes	no
Vector map (offline)	yes (13 MB)	yes (479.9 MB)	yes
Can be used in Prague	yes	yes	no
Routing types	fastest, shortest, quietest, balanced	avoid ferries, avoid highway, avoid unpaved road	unknown

2.5 Open Bicycle Trip Planner

OBTP was originally designed and implemented by Marcel Némét as a Bachelor’s project¹⁵ at Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague. Currently it is under development by Agents4its¹⁶ group. It uses OpenStreetMap data as the base for routing. During routing it considers properties of route segments and users’ requirements.

The route is evaluated from four aspects:

Speed aspect Speed aspect prioritises cycleways and cycle lanes. It avoids routes that require cyclists to dismount their bicycle and it also avoids stairs or crossings.

Comfort aspect Comfort aspect is based on speed aspect. It avoids steep ascending. It is not strict about dismounting bicycle. In comparison with speed aspect it discourages stairs much more. It prioritises routes with lower traffic.

Length aspect Length aspect only considers length of a route and ascend. Ascend has lower impact than in speed or comfort aspect.

Quietness aspect Quietness aspect prioritises routes with low or none traffic – basically all routes intended only for cyclists.

Along with average speed the combination of these aspects are used for calculation of the route. More about the use of OBTP can be find in Chapter 4 in Routing section.

¹⁵Némét, M. Open Bicycle Trip Planner. Bachelor Thesis. Czech Technical University in Prague, 2013.

¹⁶<http://agents4its.net/>

2.6. CITY BIKE PLANNER REQUIREMENTS

2.6 City Bike Planner requirements

In this section functional and non-functional requirements of the application are listed. The functional requirements define functionality of the application's components. The non-functional requirements define constraints of the application that do not have an effect on the application's functionality.

2.6.1 Functional requirements

FRQ01 The application will allow users to set start and finish points of a route. Start and finish points serve as the main properties for route calculation. They can be set by two methods. The first method is by setting the point from the map. The second method is by typing an address into the input field.

FRQ02 Application will allow users to search for a route between two points. If all required properties are set users can start the routing process.

FRQ03 The application will allow users to save favourite places. Users can save favourite places so they do not have to search for them repeatedly if they use them frequently.

FRQ04 The application will allow users to save favourite routes. Users can save favourite routes so they do not have to search for them again.

FRQ05 The application will display route on the map. The route will be displayed on the map so that users can easily see how the route is assembled.

FRQ06 The application will display a route itinerary. The list of turn-by-turn instructions will be displayed.

FRQ07 The application will display a route profile. The route profile will be visualized as a line graph so that users can see where steep ascends or descends are on the route.

FRQ08 The application will navigate users on their route. The route will be displayed on the map along with turn-by-turn navigation instructions while following users' movement.

2.6.2 Non-functional requirements

NRQ01 The application will use OBTP as the routing service.

NRQ02 The application will run on devices with Android 4.0 and higher.

NRQ03 The application will use Google Maps as the map base.

Chapter 3

Design

In this chapter I will describe the design of the application. In the first part I will describe navigation logic between individual screens. In the second part I will focus on use cases diagrams along with description of the individual use cases. In the third part I will provide images and description of GUI design of the main screens of the application.

3.1 Navigation between screens

In this section the navigation between screens is described. The scheme of the navigation is shown in Figure 3.1. The arrows are labelled with an action which links the individual screens together. The base navigation structure is formed by the navigation drawer pattern. The navigation drawer forms the main navigation menu and contains links to the planner, places, route, routes and navigation screens.

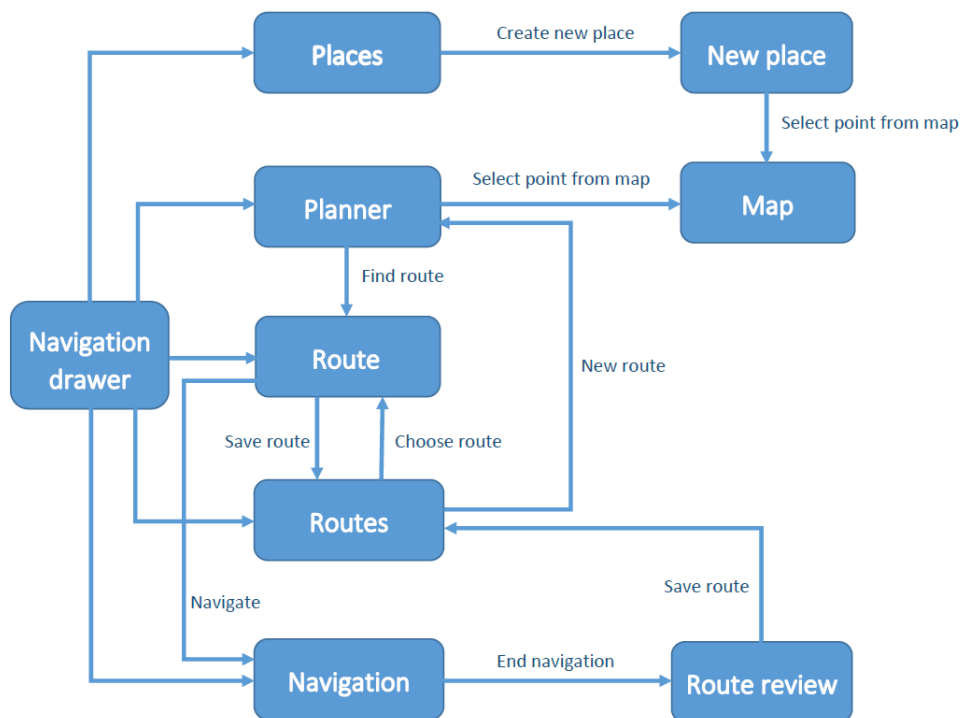


Figure 3.1: Scheme of navigation between screens

3.2. USE CASES

3.2 Use cases

In this section I will provide use cases diagrams with description of the individual use cases of the application. They are divided into four main parts according to the functionality they are connected to.

3.2.1 Route planning

In this section the use cases associated with process of route planning are described. These use cases are linked to the Planner screen.

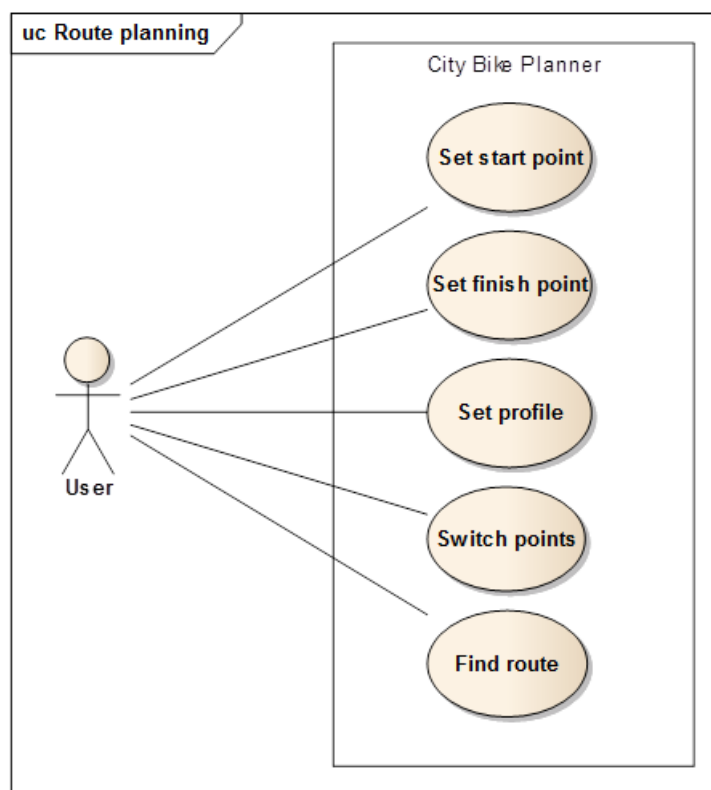


Figure 3.2: Route planning use cases diagram

UC01 Set start point – user sets a start point from the previously saved places, from the map, by searching in the street names database or by accessing the current location.

UC02 Set finish point – user sets a finish point from the previously saved places, from the map or by searching in the street names database.

UC03 Set profile – user sets a profile of the route such as Commuting, Peaceful or Extreme.

UC04 Swith points – user swiches start and finish point.

UC05 Find route – user starts the routing process.

3.2.2 Managing places

In this section the use cases associated with places management are described. These use cases are linked to the Places screen and New Place screen.

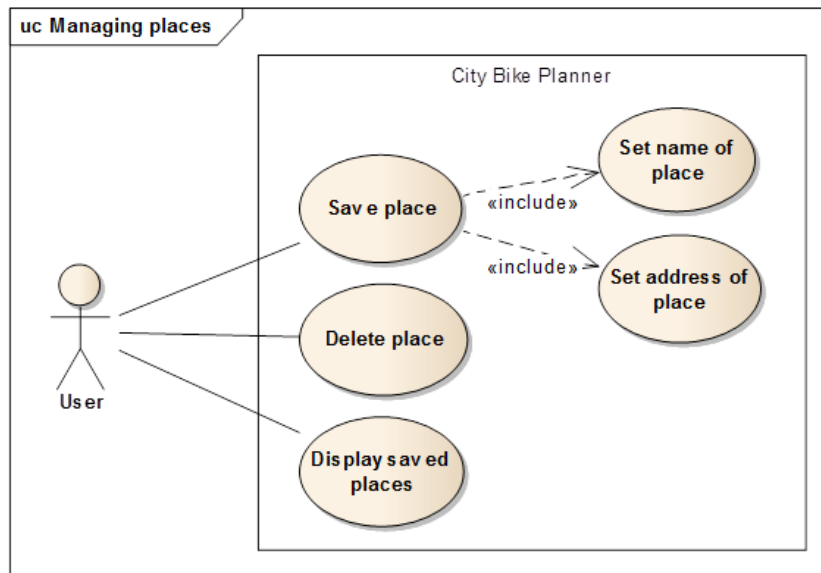


Figure 3.3: Managing places use cases diagram

UC06 Save place – user saves a new place into the database.

UC06a Set name of place – user provides a name for a new place.

UC06b Set address of place – user sets an address from the map or from the suggestions.

UC07 Delete place – user deletes a place from the database.

UC08 Display saved places – user displays a list of the saved places.

3.2.3 Managing routes

In this section the use cases associated with route management are described. These use cases are linked to the Route screen, Routes screen and RouteReview screen.

UC09 Save route – user saves a new route.

UC09a Set name of route – user provides a name for a new route.

UC10 Delete route – user deletes a route from the database.

UC11 Display saved routes – user displays a list of the saved routes.

UC12 Display current route – user displays the currently active route.

UC12a Display map – user displays a route drawn onto the map.

UC12b Display itinerary – user displays the itinerary.

UC12c Display profile – user displays the route profile.

3.2. USE CASES

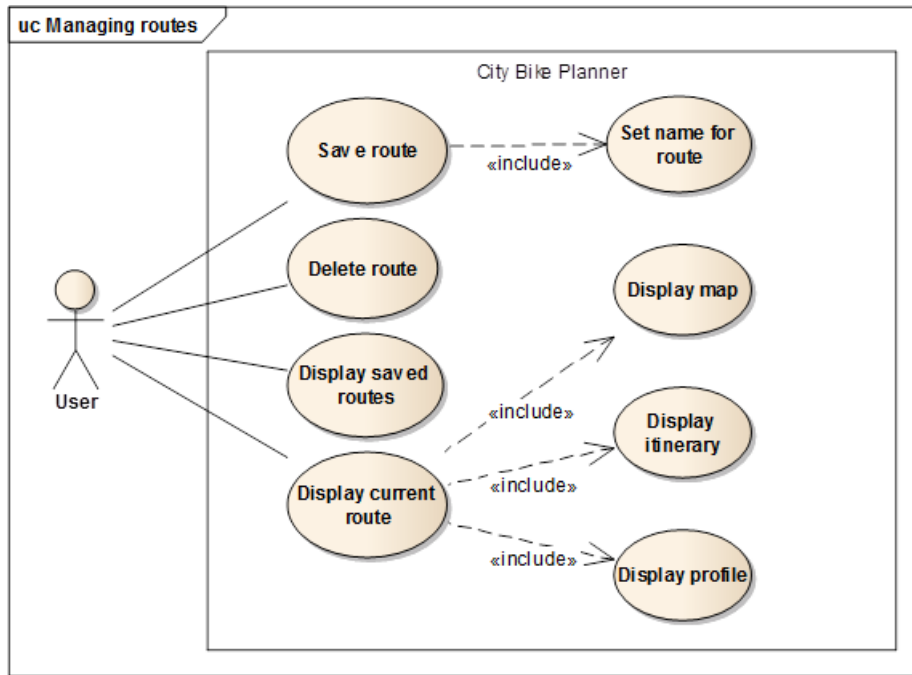


Figure 3.4: Managing routes use cases diagram

3.2.4 Managing navigation

In this section the use cases associated with navigation management are described. These use cases are linked to the Navigation screen.

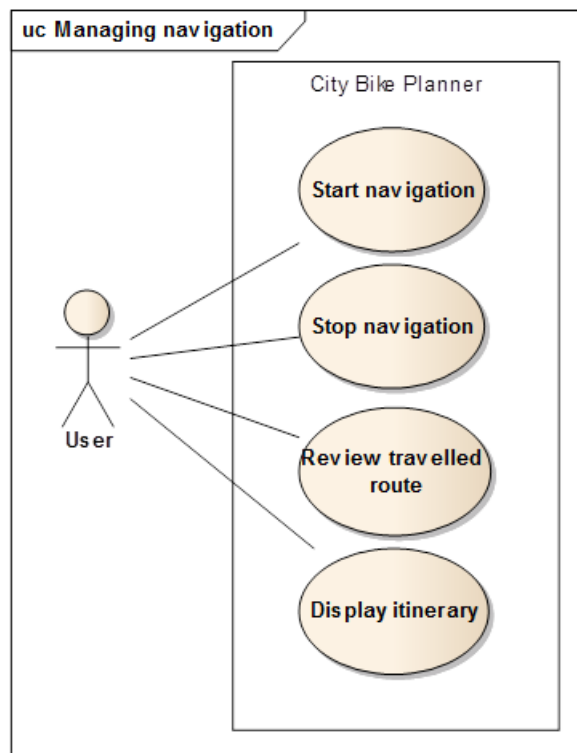


Figure 3.5: Managing navigation use cases diagram

UC13 Start navigation – user starts the navigation process.

UC14 Stop navigation – user stops the navigation process.

UC15 Review travelled route – user displays the travelled route.

UC16 Display itinerary – user displays the itinerary that starts with the current instruction.

3.3 GUI Design

In this section the user interface design for main screens will be discussed. I designed a prototype of the application as a part of my semester project. The prototype was created with Balsamiq¹ wireframing software. The prototype was tested by real users. Three people were invited to testing sessions from whom two were advanced cyclists who ride several times a week and one was a professional cyclist who rides every day. During the design process I followed Android Design guidelines².

The final UI had to be redesigned in order to meet users' preferences and to be able to work with OBTP. This section contains images from the original prototype in comparison with redesigned UI that was created in Android Studio³ layout editor.

3.3.1 Planner UI design

The planner screen (see Figure 3.6b) is the main part of the whole application. Its function is to enable users to provide parameters needed for route planning. Firstly there are parameters for definition of start and finish points of the route. Secondly there is a parameter which defines the route profile.

The route points definition group is formed by autocomplete text fields and image buttons on the right. These buttons provide access to “select from map” feature. When the buttons are clicked the map screen is opened and users can select needed locations. When the autocomplete text field is activated a suggestion box with the saved places appears. Users can choose from the saved places or type a new address; the suggestions will be filtered according to the characters typed in.

“Select from map” feature is intended for specifying precise positions of points. If the text field is filled the icon that indicates the position is shown on the map and can be easily dragged to the required position. In addition there is an option for quick access to the current position located in the action bar. After activating this option the current location is filled into the text field. An action to switch the start and finish points is also located in the action bar.

The parameter for definition of the route profile is set by a spinner. Three profiles are available: Commuting, Peaceful and Extreme.

The button that starts the routing process is located at the bottom of the screen.

¹<http://balsamiq.com/>

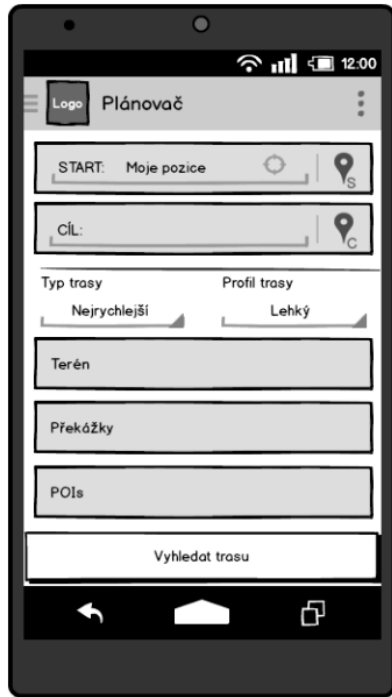
²<https://developer.android.com/design/index.html>

³<http://developer.android.com/sdk/installing/studio.html>

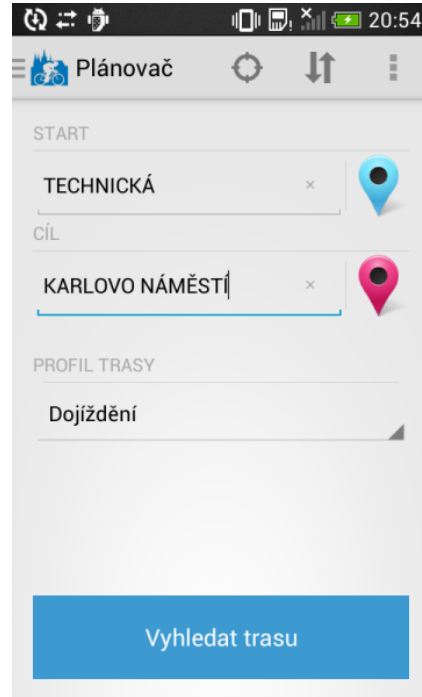
3.3. GUI DESIGN

Changes made

There were three additional buttons for definition of terrain, obstacles and POIs of the route in the prototype (see Figure 3.6a). They had to be removed because OBTP can only calculate routes according to four aspects so only the spinner for setting the route profile was kept. These profiles have predefined aspect values. By removing these buttons a new space was gained so the titles of the text fields were repositioned under the fields to meet design guidelines.



(a) Prototype planner screen



(b) Redesigned planner screen

Figure 3.6: Planner screen interface design

3.3.2 Map UI design

The map screen (see Figure 3.7b) displays map and a marker according to the currently set mode – start point mode, finish point mode and new places mode.

The point can be set by long touch gesture or by dragging the marker. Marker info window contains the title of the selected mode and address corresponding with marker location. An action to confirm selection is located in the action bar.

Users interact with the map via gestures. Swipe gesture is for map moving. Pinch open and pinch close gestures are assigned to zooming in and zooming out respectively.

Changes made

Originally the map screen (see Figure 3.7a) was designed to also show the found route so the action to switch to the navigation was located in the action bar. In the redesigned version the route is shown in separate activity and the map screen enables only selection of points.

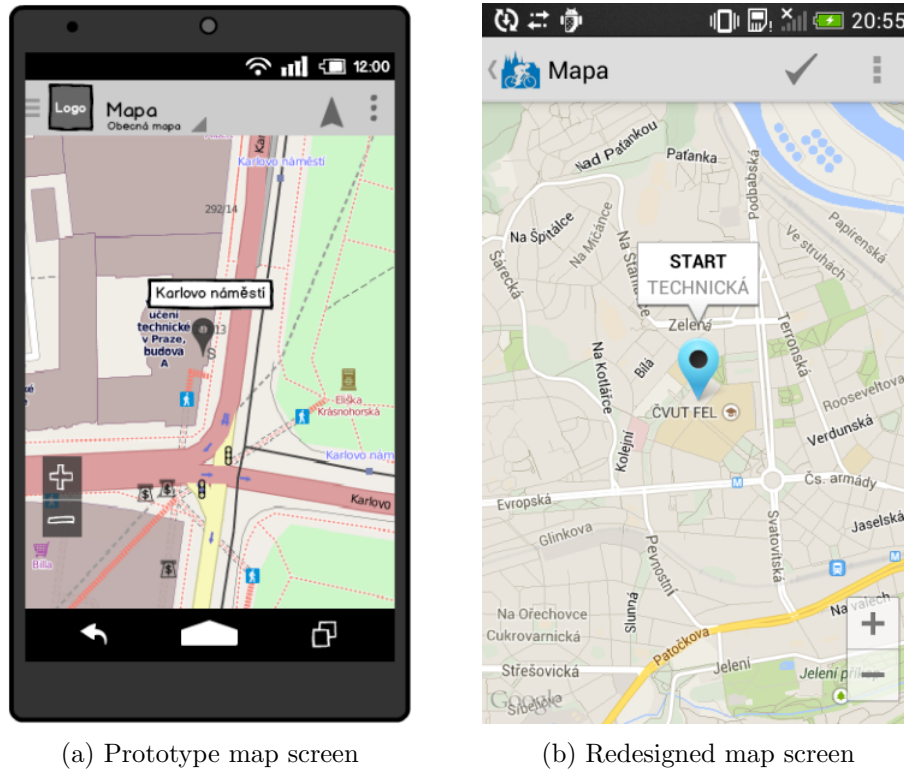


Figure 3.7: Map screen interface design

3.3.3 Route UI Design

Route screen is formed by three individual screens each of which displays different visualization of the route. These screens are accessible through the tabs navigation.

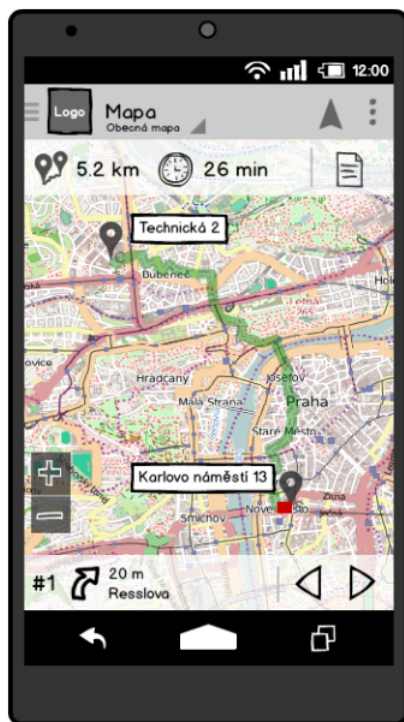
The map tab displays the route with its total length and time estimation (see Figure 3.9a). The route is drawn in different colours according to the type of the individual route segment such as cycleway or road. The icons indicating obstacles on the route are also displayed on the map to give users warning. The itinerary tab displays the itinerary of the route (see Figure 3.9b). The profile tab displays the elevation profile of the route and some additional statistics such as the total ascending or the total amount of metres travelled on cycleways (see Figure 3.9c).

The actions for saving the route and for switching to the navigation are located in the action bar.

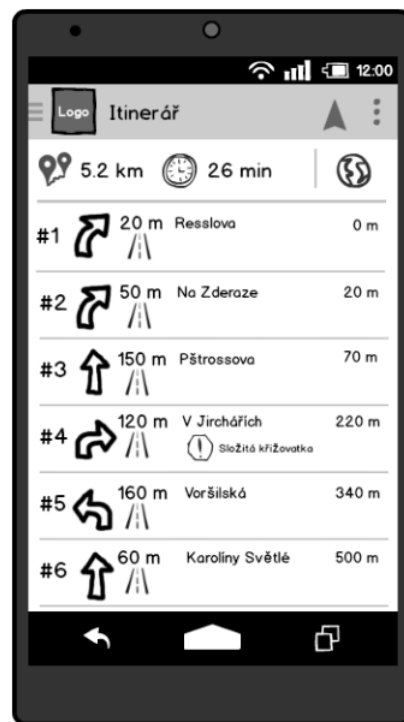
Changes made

The original route visualization was a part of the map screen (see Figure 3.8a). In the redesigned version it is a part of the new activity with the tabs navigation. Along with the route drawn onto the map there is other tab containing the route itinerary which was originally displayed on a separate screen. The profile screen was not included in the previous version and was added to satisfy requirements of users – they wanted some kind of visualization of physical requirement of the route and more information about the route.

3.3. GUI DESIGN

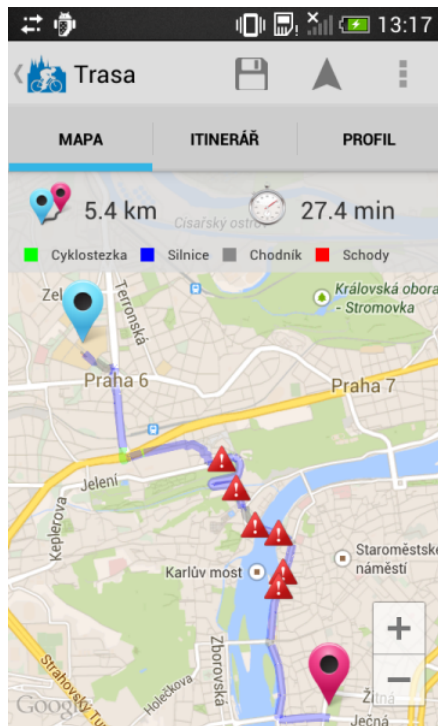


(a) Route screen

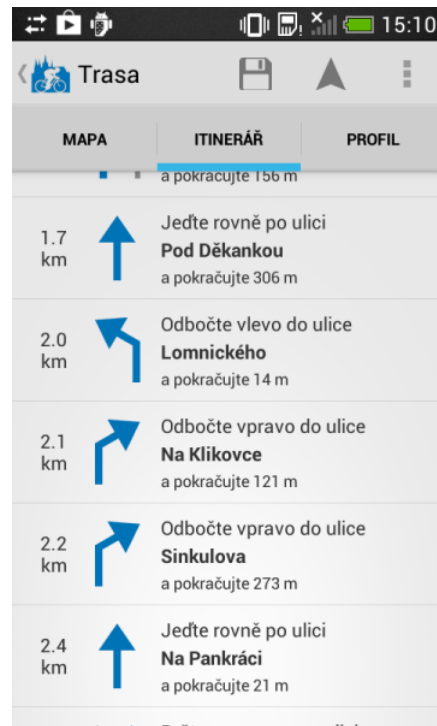


(b) Itinerary screen

Figure 3.8: Prototype route and itinerary screen design



(a) Route drawn on map



(b) Route itinerary



(c) Route profile

Figure 3.9: Route screen interface design

3.3. GUI DESIGN

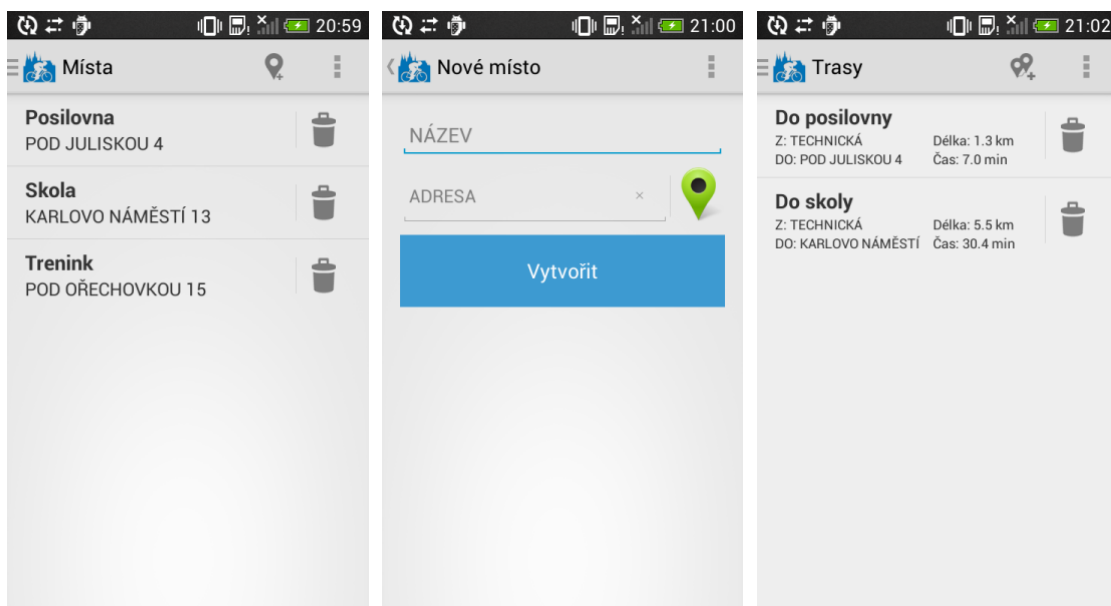
3.3.4 Places and routes UI design

The places screen contains a list of the previously saved places (see Figure 3.10a). Each list item includes the name and address of the place. There is a button for removing the place from the list on the right side of the item. The action for adding a new place to the list is located in the action bar.

If users want to add a new place a new screen is displayed (see Figure 3.10b). The screen contains two text fields. The first one is for the name of the place which is used instead of the address in suggestions list when choosing start or finish point of the route and the second one is for the address itself. There is also an image button for setting a place from the map on the right side of the address text field.

The places screen was redesigned only visually and it is not different from the prototype design so only the redesigned screens will be shown below.

The routes screen is designed similarly to the places screen. It displays the name of the route, start and finish address, route length and time estimation. An action for a new route is located in the action bar and leads to the planner screen. The routes screen was not a part of the prototype design.



(a) List of places

(b) Creation of new place

(c) Routes screen design

Figure 3.10: Places and routes interface design

3.3.5 Navigation UI Design

The navigation screen (see Figure 3.11b) provides users with turn-by-turn instructions and visualizes the nearest surroundings during the ride. The top part of the screen contains the instructions themselves. On the top left side there is an enlarged icon of the direction and distance to the direction change to them together with the address linked to this instruction. Users' position is shown at the bottom part of the screen, so that users can see a bigger part of the map that lies before them. The orientation of the map is recalculated according to the current direction of users' movement. The actions to display the itinerary and to stop the navigation are located in the action bar.

Changes made

The original version displayed icons of POIs on the map (see Figure 3.11a) and the instructions were positioned differently. In the redesigned version only users' position and basic instructions are shown so the individual content parts do not fight for users' attention. The route can have four colors – blue (road), green (cycleway), gray (pavement) and red (stairs) which show users to know where they are supposed to ride.

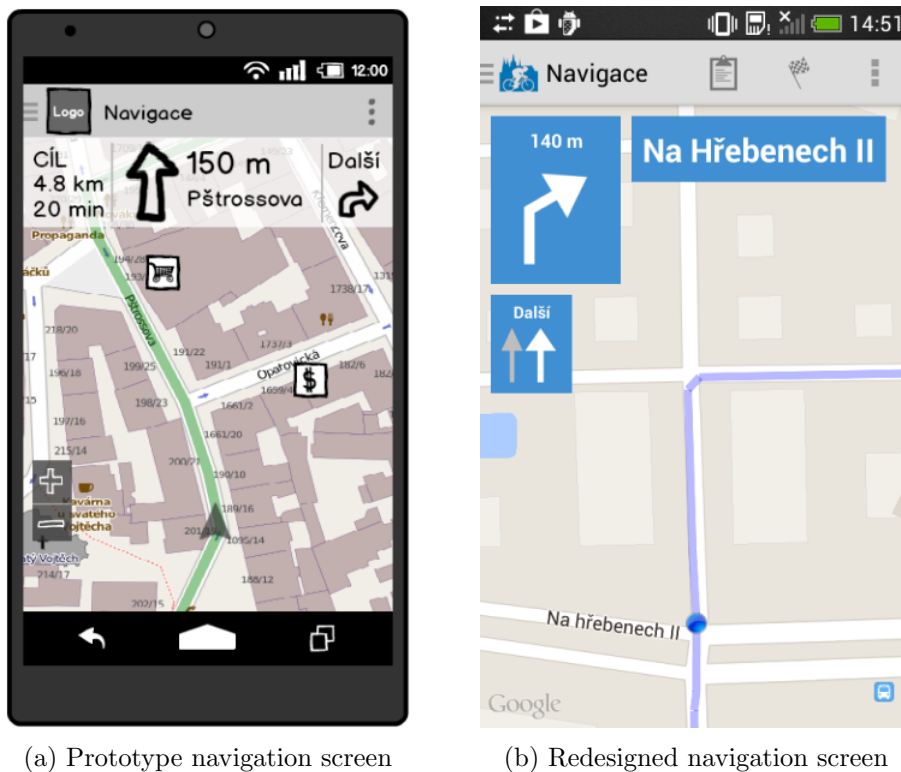


Figure 3.11: Navigation screen interface design

3.3. GUI DESIGN

Chapter 4

Implementation

The application was implemented for Android operating system. It is designed to run on mobile devices with Android 4.0 and higher. It uses OBTP routing service for route calculation and navigation instructions. It also uses Google Play services for map displaying and for geocoding and reverse geocoding features. The HTTP protocol is used for communication with the OBTP and its services are accessed via REST API. In this section I will describe the implementation process and structure of the application.

4.1 Places data preparation

I created a database that contains street names in Prague which are used for suggestions while searching for places. Users can set start or finish point of route by selecting a street from the suggestions list. I obtained initial data containing names of streets from the official database of addresses located on Ministry of the Interior website¹. The database contains over 8,000 entries. There was a lot of duplicities because some streets are located inside more than one city district so for each district there is an individual entry. Other problem was duplicity of abbreviations such as “Karlovo nám.” and “Karlovo náměstí” or “Malostranské nábř.” and “Malostranské nábřeží”. I deleted these duplicities and the final number of streets is 7,278.

As the next step I had to assign latitude and longitude values to each street. I used API Mapy.cz to access these values. Mapy.cz² provides latitude and longitude values that are positioned in the middle of the street. So for specifying precise position of wanted place the feature “Select from map” have to be used.

The `placesdatabase.db` file is located in the `assets/databases` directory and is loaded into the application using Android `SQLiteAssetHelper`³ which is a helper class that manages creation and version management using raw asset files.

4.2 Databases

The application contains two SQLite databases. The first database is used to save places and the second database is used to save routes. I created the places database externally

¹<http://aplikace.mvcr.cz/adresy/>

²<http://api.mapy.cz/>

³<https://github.com/jgilfelt/android-sqlite-asset-helper>

4.2. DATABASES

and it is loaded into the application differently than common databases. That is the reason why there are two databases instead of one. The routes database is created by the application. Each of the databases is accessed with its own content provider.

Places database

The places database contains table of places from `placesdatabase.db` file that are used for suggestions and it also contains favourite places saved by user. The table consists of the following columns:

- **_id** is the unique identifier of database record,
- **place_name** is the name of the place without diacritics according to which the suggestions are filtered,
- **name** is the name of the place with diacritics,
- **address** is the name of the street associated with the place,
- **latitude** is the latitude value of the place's location,
- **longitude** is the longitude value of the place's location,
- **my_place** is an indicator of places saved by a user.

Routes database

The routes database contains table of users' favourite routes. It also contains the currently active route. The start and finish addresses defined by users have to be saved along with their latitude and longitude values because they are not contained in server response. The table consists of the following columns:

- **_id** is the unique identifier of database record,
- **json** is the json string containing response from server,
- **name** is the name of the route,
- **start_address** is the name of the street associated with start point,
- **start_latitude** is the latitude value of start point,
- **start_longitude** is the longitude value of start point,
- **finish_address** is the name of the street associated with finish point,
- **finish_latitude** is the latitude value of finish point,
- **finish_longitude** is the longitude value of finish point,
- **length** is the length of the route,
- **time** is the time estimation of the route,
- **my_route** is an indicator of routes saved by a user.

4.3 Implementation structure

The application is divided into several packages. Their purpose and the main classes located in them will be introduced in this section.

4.3.1 Activities and fragments

The `activities` package contains classes that represent the individual application screens. These screens are formed by UI layouts and components which are located in the `res/layout` directory. For some of the activities a fragment from the `fragments` package is used as the main layout for the screen. The main activities and fragments are:

DrawerActivity The `DrawerActivity` class serves as the main activity for the application that is launched right after the application is launched. It uses the Navigation Drawer⁴ for navigation to other screens. It can have four different fragments as the main layout:

- The `PlannerFragment` class contains UI for displaying form for definition of route properties.
- The `PlacesFragment` class contains UI for displaying a list of the saved places.
- The `RoutesFragment` class contains UI for displaying a list of the saved routes.
- The `NavigationFragment` class contains UI for displaying the map and navigation instructions.

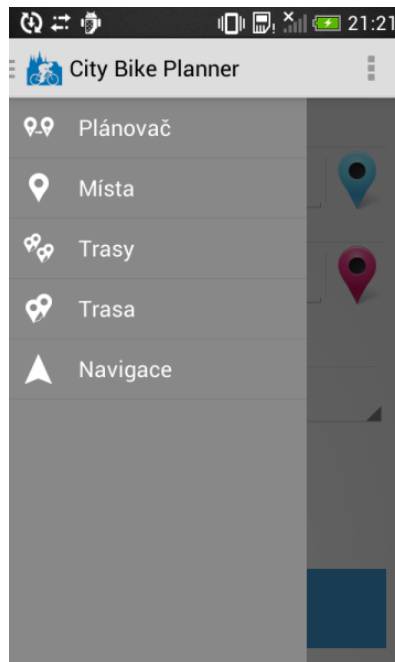


Figure 4.1: Drawer activity with navigation drawer

⁴<https://developer.android.com/design/patterns/navigation-drawer.html>

4.3. IMPLEMENTATION STRUCTURE

MapActivity The `MapActivity` class has the `MapsFragment` class as a layout. It displays map from which users can select positions for the start point, for the finish point or a new place.

RouteActivity The `RouteActivity` class contains the Fixed Tabs⁵ navigation for access to three separate fragments:

- The `RouteMapFragment` class displays the found route on the map.
- The `RouteItineraryFragment` class displays a list of turn-by-turn instructions.
- The `RouteProfileFragment` class displays route profile information and a graph of elevation for which the `GraphView`⁶ library is used.

RouteReviewActivity The `RouteReviewActivity` class displays planned route and real route that users have travelled.

4.3.2 Adapters

The `adapters` package contains classes that are used to populate different types of lists in the application.

DrawerAdapter The `DrawerAdapter` class provides data for the navigation drawer.

ItineraryAdapter The `ItineraryAdapter` class provides data for itinerary fragment in the `RouteActivity`.

PlacesAdapter The `PlacesAdapter` class provides data from the database of places for a list of the saved places in the `PlacesActivity`.

PlacesAutocompleteAdapter The `PlacesAutocompleteAdapter` class provides data from the database of places that are used for suggestions for autocomplete text fields in the `PlannerFragment` and in the `NewPlaceActivity`.

RoutesAdapter The `RoutesAdapter` class provides data from the database for a list of the saved routes in the `RoutesActivity`.

4.3.3 Entities

The `entities` package contains mainly classes that are used to parse JSON route data into java object.

Instruction The `Instruction` class represents instructions data linked to the individual route coordinates.

RouteNode The `RouteNode` class represents an individual route coordinate and contains the `Instruction` linked to them.

⁵<https://developer.android.com/design/building-blocks/tabs.html>

⁶<http://android-graphview.org/>

Route The route is formed by the `RouteNode` objects. The `Route` object is used for displaying the route on the map, for creating an itinerary and for displaying the profile graph.

4.3.4 Helpers

The `helpers` package contains classes that simplify work with graphical resources and start, finish and new place points specific variables.

DirectionImageHelper The `DirectionImageHelper` class contains methods that help to transform string representation of directions types to resources images. These methods are used for displaying directions images in itinerary and during the navigation process.

GeneralPointHelper The `GeneralPointHelper` is an interface that defines methods for the work with start, finish and new place points and markers as their representation on the map. The `StartHelper`, `FinishHelper` and `NewPlaceHelper` classes implement this interface. Each of these classes then return their specific titles, snippets, positions and icons for displaying markers on the map and for routing.

MarkerFactory The same activity is used for map displaying so the `MarkerFactory` class creates a specific marker according to the mode in which the activity is launched.

4.3.5 Location

The `location` package contain classes that work with users' location.

GeocoderManager The `GeocoderManager` class contains methods used for geocoding and reverse geocoding.

LocationReceiver The `LocationReceiver` class extends the `BroadcastReceiver`⁷ class and it listens for changes of device's location.

NavigationManager The `NavigationManager` class is used for requesting device's location that is used during the navigation process.

4.3.6 Mock

The `mock` package contains classes used for testing mock location updates from a custom test provider. The test provider was used only for testing the navigation process to ensure the right turn-by-turn instructions are displayed and that the map is correctly rotated in the direction of user's movement.

4.3.7 Providers

The `providers` package contains two providers – the `PlacesProvider` and `RoutesProvider`. These providers give access to databases that store places and routes and to CRUD operations to modify their contents.

⁷<http://developer.android.com/reference/android/content/BroadcastReceiver.html>

4.4. IMPLEMENTATION DETAILS

4.3.8 Routing

The `routing` package contains classes that are used for obtaining route from server and its additional modification.

RouteManager The `RouteManager` contains only one static method that returns the currently active route from the `RoutesProvider`.

RouteParser The `RouteParser` class contains a method that parses server response into a `Route` object. Gson⁸ open-source java library is used for parsing JSON response.

RouteRequest The `RouteRequest` class is used for creating route request that is sent to server in order to obtain calculated route.

4.3.9 Utils

The `utils` package contains additional classes.

HttpRequest The `HttpRequest` class created by Kevin Sawicki⁹ is used for communication with server.

MathUtil and SphericalUtil The `MathUtil` and `SphericalUtil` classes are part of Google Maps Android API utility library¹⁰. They are used for bearing calculation, for correct map rotating and for indentation of the current position indicator on the navigation screen.

4.4 Implementation details

In this section details of implementation are discussed. In the first part I will describe geocoding and reverse geocoding and its use. Then I will focus on the use of OBTP. In the last part I will describe the navigation functionality.

4.4.1 Geocoding

In the application both geocoding and reverse geocoding are used. The `Geocoder`¹¹ class from Google Play services is used for both types of geocoding.

The geocoding is used when users are setting a place by the text input method. As they write the `GetPointTask` is executed asynchronously in order to obtain corresponding latitude and longitude values. The task is located in the `PlannerFragment` class and uses the `getPointForAddress` method from the `GeocoderManager`. When users only select a place from suggestions latitude and longitude values from the `placesdatabase.db` are used and no task is executed.

The reverse geocoding is used when setting a point from the map. Users can set a point by long touch on the desired position on the map. A marker is created and the `GetAddressTask`

⁸<https://code.google.com/p/google-gson/>

⁹<http://kevinsawicki.github.io/http-request/>

¹⁰<https://github.com/googlemaps/android-maps-utils>

¹¹<http://developer.android.com/reference/android/location/Geocoder.html>

is executed asynchronously in order to obtain the address of the place. The task uses combination of `getStreetNameForAddress` and `getAddressForLatLng` methods from the `GeocoderManager`. The task is also executed after users have dragged an existing marker to a new position.

4.4.2 Routing

OBTP was selected as the routing service. It also returns turn-by-turn instructions. The service is accessible via REST API. In this section I will describe communication with the service.

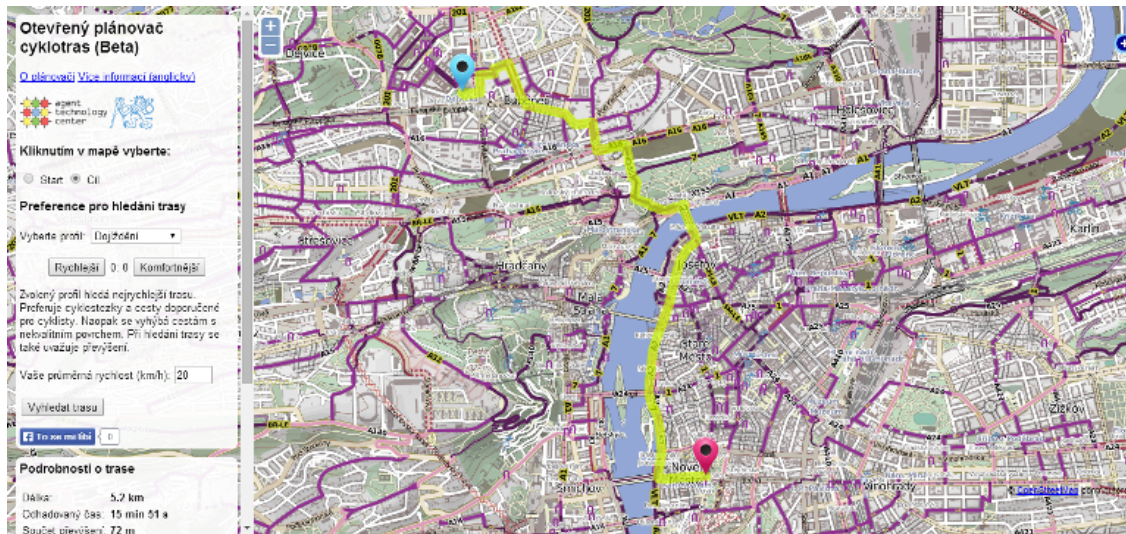


Figure 4.2: Open Bicycle Trip Planner web interface

Request structure

A GET request that is sent to server is created by the `createRequest` method located in the `RouteRequest` class. The method takes `LatLng` values for start and finish points and `Profile` of the route as the parameters from which it creates request in the following structure:

```
http://transport.felk.cvut.cz/cycle-planner-1.1.3-SNAPSHOT/bicycleJourneyPlanning/
planJourney?startLon=* & startLat=* & endLon=* & endLat=* & avgSpeed=*
& travelTimeWeight=* & comfortWeight=* & quietnessWeight=* & flatnessWeight=*
```

where:

- **startLon** and **startLat** are longitude and latitude values of the start point,
- **endLon** and **endLat** are longitude and latitude values of the finish point,
- **avgSpeed** is a cyclist's average speed in kilometres per hour,
- **travelTimeWeight** is speed aspect value,
- **comfortWeight** is comfort aspect value,
- **quietnessWeight** is quietness aspect value,

4.4. IMPLEMENTATION DETAILS

- **flatnessWeight** is shortest distance aspect value,

and * represents decimal values of the individual properties.

Profiles

The profile of the route can be set to meet users' preferences and it is represented by the **Profile** interface. Three basic profiles are implemented as they are described in Németh's Bachelor's thesis in Chapter 5. I describe the profiles along with the used values for **travelTimeWeight**, **comfortWeight**, **quietnessWeight**, **flatnessWeight** aspects respectively:

- **Commuting profile** finds a quick and comfortable route. It prioritises cycleways and avoids unsuitable surface.
Aspects values: 2, 1, 0, 0.
- **Peaceful ride profile** finds a comfortable route and avoids traffic.
Aspects values: 0.5, 3, 6, 0.
- **Extreme profile** finds the shortest route but may contain more obstacles.
Aspects values: 3, 0, 0, 1.

Sending request

When users set all parameters on the planner screen they can start the routing process. When they do a request is created and sent asynchronously by executing the **RouteTask**. The **HttpRequest** class is used for the communication with the server. A connection and read timeouts are set to five seconds so it gives plenty of time for the application to connect to the server and obtain a response. When the timeouts run out an exception is thrown and users are informed by a dialog¹² that connection to the server has failed.

Server response

A response from the server containing the route properties is sent in JSON format and is parsed by the **parseRoute** method from the **RouteParser** class.

A response contains the following properties:

- **status** – indicates if a route was found,
- **responseId** – unique identification number of the route,
- **boundingBox** – left, top, right and bottom coordinates of bounding box of the route,
- **coordinates** – array of longitude and latitude pairs which defines the route shape,
- **length** – length of a route in metres,
- **duration** – estimated time in seconds,

¹²<http://developer.android.com/guide/topics/ui/dialogs.html>

- **elevationGain** – total elevation gain in metres,
- **elevationDrop** – total elevation drop in metres,
- **elevationProfile** – array of altitude values in metres corresponding to coordinates,
- **cumulativeDistance** – distance of coordinate from the start point,
- **instructions** – array of instructions, each instruction can contain:
 - **streetName** – name of the street that starts in the corresponding coordinate,
 - **roadType** – indicates the type of the road such as footway or cycleway,
 - **manoeuvre** – turn-by-turn instruction such as turn left or turn right,
 - **surface** – indicates the type of surface such as unpaved or cobblestone.

4.4.3 Navigation

The navigation is an essential part of the application. When users start the navigation process the `startLocationUpdates` method from the `NavigationManager` class is executed so the manager begins to request location updates every second. The `LocationReceiver` obtains these updates and manages them in the `onReceive` method.

The `LocationReceiver` is used in the `NavigationFragment` where the received locations are handled. Every location update is recorded and saved in the Application context so it can be later used to display the real route that users travelled. Bearing is always calculated from the previous location and current location for the correct map rotation in the direction of users' movement.

The `computeOffsetOrigin` method is used for indentation of the current position indicator on the map so it is positioned near the bottom part of the screen.

4.5 Graphics

All graphics resources are located in `res/drawable-xxxx` directories. I designed logo of the application and directions icons that are used in the itinerary and during navigation. Other icons and especially marker icons were downloaded from Icons Land¹³ website.



(a) Application icon



(b) Start marker



(c) Turn left icon

Figure 4.3: Examples of used icons

¹³<http://www.icons-land.com/>

4.6 Limits of the application

Unfortunately the application cannot satisfy all needs and requirements of cyclists. The implementation is limited by:

Map data The Google maps are used for displaying the map in the application. These maps are not designed for cyclists so the cycleways are not highlighted on the map (see Figure 4.4) and also some other map visualizations do not have to correspond to the real live situations. That is the major problem because the route that is draw onto the map may appear to lead nowhere but in reality it does, and it is just not contained in the map base, which may confuse users.

Route customization The original design took the detailed properties of the route and users' requirements into consideration. Users could select types of surfaces that they wanted to include in route calculation or they could define obstacles that they did not want to encounter on their ride. This original design had to be abandoned due to capability of OBTP so only three profiles are available for definition of the route and the individual terrains are drawn in different colours.

Instructions OBTP uses OSM data for routing and for creation of navigation instructions. The correctness of instructions depends only on the quality of the OSM data. Some manoeuvres are not available so it may happen that there are more instructions between two manoeuvres but they are not contained in server response. That may also confuse users because there are no instructions available even if the route is obviously changing its shape on the map.

GPS data The application uses GPS data to access the current location of users. The accuracy of the data depends on the ability of the mobile device to receive this data and on the quality of GPS satellite coverage of the individual parts of the city.

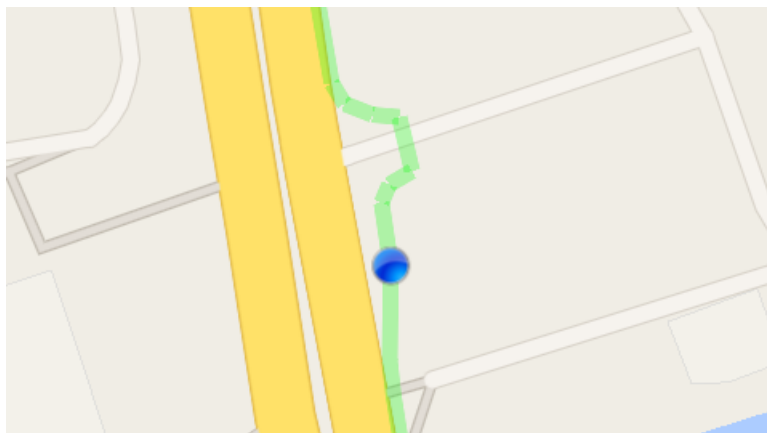


Figure 4.4: Example of map data without cycleway

Chapter 5

Evaluation

5.1 Prototype testing

I designed a low-fidelity prototype of the application. Its functionality was limited but the main features were available. The prototype was tested by three real users during the testing sessions. Two of them were cyclists who ride in the city several times a week and one was a professional cyclist who rides every day.

The sessions were divided into two parts. In the first part the users answered questions about their experience with city cycling (see Appendix B). In the second part they worked with the prototype and were asked to think aloud in order to receive feedback.

The most important findings from the sessions were:

1. The users mostly use journey planner applications available from web. They find the route online and later they navigate from the memory or from written instructions. They do not use mobile applications for navigation while they are riding.

Solution: An action for a quick access to the itinerary was added to the action bar. That could substitute the written instructions.

2. The users would prefer if the application found several routes from which they could choose the route which they like the best.

Solution: The solution would be to display several routes. In the current implementation of the application this option is not implemented. There is a problem of how to decide that two or more routes are different enough so that they can be displayed together. This is an interesting problem to solve in the future development.

3. The users do not mind obstacles on the route but they would like to be informed about them.

Solution: The warning icons of possible obstacles are displayed on the map.

4. The users state cobblestone routes as the most difficult obstacle.

Solution: A description of what an obstacle is can be displayed by clicking a warning icon. The obstacles that can be determined from the OSM data are only cobblestone and unpaved road.

5.2. APPLICATION TESTING

5. The users do not like steep ascending.

Solution: The profile of the route is displayed in the profile tab on the route screen. Users can see where they have to overcome steep ascending.

These findings were considered during the redesigning process of the user interface. The changes made in the user interface are shown in chapter 3 in GUI Design section.

5.2 Application testing

The final application was tested by eight users. The participants of the test were students between 20–25. They all have experience with cycling in the city.

The objective of the session was to test usability of the application and find possible defects in the interface design. The participants were provided with a list of tasks (see Appendix C). The tasks were designed to test almost every function in the application. I was watching the participants as they were executing the tasks and took notes.

The application City Bike Planner in version 2.4 was tested and it was installed on mobile device HTC Desire 500 with Android 4.1.2 Jelly Bean operating system.

Now I will describe the results from the testing sessions. For each task the expected execution is described along with information about how many participants executed task as expected. If there were problems with a task's execution a description of the differences from the expected executions are also included.

At the start of the session the navigation drawer was displayed to simulate first launch of the application.

Task 1 You want to plan a new route.

Expected execution: Clicking on the Planner item in the navigation drawer menu.

Correctly executed by: 8 participants

Task 2 You want to set your current location as your start point.

Expected execution: Clicking on the my location action located in the action bar.

Correctly executed by: 3 participants

Issues: Five participants did not associated the my location icon with setting the current location. Two participants started typing address into the input field, three participants tried to set the start from the map.

Task 3 You have changed your mind, you want to start in Technická street now.

Expected execution: Typing the word “Technická” into the start input field. Choosing Technická street from the suggestions list as soon as it appears.

Correctly executed by: 8 participants

Task 4 You want to set Karlovo náměstí 13 as your finish point.

Expected execution: Typing the word “Karlovo náměstí” into the finish input field. Choosing Karlovo náměstí street from the suggestions list as soon as it appears and adding the number 13 to the end.

Correctly executed by: 8 participants

Task 5 You are in no hurry and you do not like busy streets, so you set the route profile accordingly.

Expected execution: Setting the peaceful profile as the route profile. (By default the commuting profile was set.)

Correctly executed by: 8 participants

Task 6 Find the route.

Expected execution: Clicking on the find route button.

Correctly executed by: 8 participants

Task 7 Find out how long the route is and what the time estimation is.

Expected execution: Reading the information from the labels in the route screen.

Correctly executed by: 8 participants

Task 8 It is possible that you will encounter some obstacles on your route. Find out what kind of obstacles they are.

Expected execution: Clicking on warning icons and reading the descriptions of the warnings.

Correctly executed by: 8 participants

Task 9 Find out on the map on which type of route you will travel most of the time.

Expected execution: Associating the colour legend with the parts of the route drawn in different colours and stating pavement as the type of the road which forms most of the route.

Correctly executed by: 8 participants

Task 10 Find out what the distance from the start point is when you go through U Písečné brány street.

Expected execution: Switching to the itinerary tab and scrolling down the list to the item with U Písečné brány street.

Correctly executed by: 6 participants

Issues: One participant did not associated the list of instructions with the word “itinerary”. The other participant tried to find the U Písečné brány street on the map and guess the distance.

Task 11 Find out how many metres you will travel on cycleways.

Expected execution: Switching to the profile tab and reading the information.

5.2. APPLICATION TESTING

Correctly executed by: 5 participants

Issues: Three participants tried to click on the cycleway in the legend in the map tab.

Task 12 Find out what the distance from the start point is when you are at the highest altitude. What is the altitude?

Expected execution: Switching to the profile tab and reading the information from the graph.

Correctly executed by: 8 participants

Task 13 You do not want to set out just yet, but you want to save your route for later. Save your route.

Expected execution: Clicking on the save route action located in the action bar. Naming the route and saving it.

Correctly executed by: 8 participants

Task 14 Display your route again.

Expected execution: Clicking on the route item in the list of the saved routes.

Correctly executed by: 8 participants

Task 15 Switch to the navigation.

Expected execution: Clicking the navigate action located in the action bar.

Correctly executed by: 7 participants

Issues: One participant opened the navigation from the navigation drawer.

Task 16 Launch the navigation.

Expected execution: Clicking the start navigation button.

Correctly executed by: 8 participants

Task 17 You are not sure where to go next. Display the itinerary and see the instructions.

Expected execution: Clicking the show itinerary action located in the action bar.

Correctly executed by: 7 participants

Issues: One participant tried to display the itinerary from the route screen.

Task 18 End the navigation prematurely.

Expected execution: Clicking the end navigation action located in the action bar.

Correctly executed by: 8 participants

Task 19 Find out how many kilometres you have actually travelled.

Expected execution: Reading the information from the route review screen associated with red colour.

Correctly executed by: 8 participants

Task 20 You want to visit Planetárium tomorrow. You know that Planetárium is in Královská obora 233. Save this place, so you can easily use it in route search tomorrow.

Expected execution: Opening the places screen from the navigation drawer menu. Clicking on the new place action located in the action bar. Naming the place, setting the address and clicking on the save place button.

Correctly executed by: 8 participants

Task 21 You are at Planetárium and you want to plan a route back to your start point.

Expected execution: Clicking on the flip points action located in the action bar. (Technická street was set as the start point and Planetárium was set as the finish point by default.)

Correctly executed by: 5 participants

Issues: Three participants tried to set the points by typing the addresses into the input fields.

Task 22 Delete Planetárium from the list of saved places.

Expected execution: Clicking on the bin icon in the place item in the saved places list.

Correctly executed by: 8 participants

5.2.1 Results of the testing

All participants stated the application is easy to use as whole. Most of the problems were associated with actions positioned in the action bar, the participants were focused on the main content of the screen and might have overlooked these actions. They also stated that if the my location button was positioned near the start input field it would be easier for them to associate it with setting the current location. The same applies to the flip points action. The participants also liked the suggestions because they did not have to type the whole name of the street.

5.2. APPLICATION TESTING

Chapter 6

Conclusion

In this Bachelor's project I have designed a journey planner application for cyclists. The application is intended to be used only in Prague. It uses Open Bicycle Trip Planner for route planning and for navigation instructions.

The main part of the application is the planner which enables users to select start and finish point of the journey. The profile of the route can be further customized to meet users' specific requirements. The application contains a database of Prague street names that are used as suggestions to make it easier for users to set journey's start and finish points. Also the points can be set from the map so users can specify a precise location in Prague and they do not depend on predefined locations. The individual locations and users' favourite places can be saved in the application so that the process of setting the start and finish points can be speeded up.

The application visualizes the found route in three different ways. The first way displays the route on the map where different types of route are drawn in different colours so users can see where they are supposed to ride on the road or cycleway. The second way displays route itinerary which is a list of individual turn-by-turn instructions with information about where to turn, which street to turn to and how far it is to turning. The third way displays the elevation profile of the route. The profile is visualized in graph so users can see where on the route they will have to go uphill or downhill.

The last important part of the application is the navigation. Users can see their progress on their journey because the application tracks their movement. The navigation also displays instructions to help users with orientation on their ride. The itinerary is also accessible from the navigation so users can easily check several successive instructions.

During the design process the application was tested by real users and their requirements were considered in order to improve the usability of the application.

The current application implementation is in some ways limited. Users can choose only from three different profiles for routing that correspond to profiles used in OBTP. The map bases might not correspond with real situations because Google Maps are used and they are not intended for cyclists. The turn-by-turn instructions or manoeuvres may be wrong in some cases because the instructions are derived from OSM data that may not be complete or accurate.

The application can be improved further. The problem of the bad map bases could be solved by using vector map bases containing information for cyclists but currently there is no map SDK for Android that could provide all important features for the application without bugs and ensure absolute functionality without crashes. A new custom map

SDK would have to be implemented in order to provide the application with a completely functional SDK. At the same time if vector map bases would be used the connection to the Internet would not be required during the navigation. The profiles that users can choose could be more extended so users have more than three profiles to choose from. This extension would have to be a result of discussion with OBTP developers so more profiles that are different enough from each other could be provided.

References

- [1] *Android Developers*. URL: <http://developer.android.com/index.html> (visited on 04/25/2014).
- [2] *Android GraphView*. URL: <http://android-graphview.org/> (visited on 04/26/2014).
- [3] Auto*Mat. *Průzkum cyklistických preferencí ze září 2012*. 2013. URL: http://www.auto-mat.cz/wp-content/uploads/2013/04/cyklopr%C5%AFzkum_2012_re%C5%A1er%C5%A1e.pdf.
- [4] *BikeCityGuide*. URL: <http://www.bikecityguide.org/> (visited on 04/26/2014).
- [5] I.G. Clifton. *Android User Interface Design: Turning Ideas and Sketches into Beautifully Designed Apps*. Usability. Pearson Education, 2013. ISBN: 9780133154818.
- [6] *Cyclestreets*. URL: <http://www.cyclestreets.net/> (visited on 04/26/2014).
- [7] Google Developers. *Google Maps Android API v2*. URL: <https://developers.google.com/maps/documentation/android/> (visited on 04/26/2014).
- [8] V. Filler. *Dotazovací průzkum cyklistických preferencí - analýza výsledků*. 2010. URL: http://prahounakole.cz/wp-content/pnk/uploads/2010/10/cyklopruzkum_5_2010_1.1.pdf.
- [9] J. Gilfelt. *Android SQLiteAssetHelper*. URL: <https://github.com/jgiltfelt/android-sqlite-asset-helper> (visited on 04/26/2014).
- [10] E. Hellman. *Android Programming: Pushing the Limits*. Pushing the Limits. Wiley, 2013. ISBN: 9781118717356.
- [11] *MapQuest*. URL: <http://developer.mapquest.com/> (visited on 04/26/2014).
- [12] *Mapsforge*. URL: <https://code.google.com/p/mapsforge/> (visited on 04/26/2014).
- [13] R. Meier. *Professional Android 4 Application Development*. ITPro collection. Wiley, 2012. ISBN: 9781118237229.
- [14] M. Némét. “Open Bicycle Trip Planner”. Bachelor Thesis. Czech Technical University in Prague, 2013. URL: https://dip.felk.cvut.cz/browse/pdfcache/nemetma1_2013bach.pdf.
- [15] G. Nudelman. *Android Design Patterns: Interaction Design Solutions for Developers*. Wiley, 2013. ISBN: 9781118417553. URL: <http://books.google.cz/books?id=Ifg1ZpSo7cwC>.
- [16] *Nutiteq*. URL: <http://www.nutiteq.com/> (visited on 04/26/2014).
- [17] *OsmAnd*. URL: <http://osmand.net/> (visited on 04/26/2014).
- [18] *OSMBonusPack*. URL: <https://code.google.com/p/osmbonuspack/> (visited on 04/26/2014).
- [19] *Osmdroid*. URL: <https://code.google.com/p/osmdroid/> (visited on 04/26/2014).

REFERENCES

- [20] B. Phillips. *AndroidCourseResources*. URL: <https://github.com/bignerdranch/AndroidCourseResources> (visited on 04/26/2014).
- [21] J. Pucher and R. Buehler. *City Cycling*. MIT Press. ISBN: 9780262304993.
- [22] K. Sawicki. *http-request*. URL: <http://kevinsawicki.github.io/http-request/> (visited on 04/26/2014).

Appendix A

Contents of the CD

<dir> apk -- contains apk file of the application

<dir> prototype -- contains low-fidelity prototype of the application

<dir> source codes -- contains source codes of the application

<dir> thesis-latex -- contains latex version of the thesis

<dir> thesis-pdf -- contains thesis pdf file

Appendix A: Contents of the CD

Appendix B

Prototype questionnaire

Experiment – Prototyp mobilní navigace pro cyklisty

17. 2. 2014

Participant:

Demografické údaje

- věk
- pohlaví

Jízda na kole

- Jak dlouho jezdíte?
- Na čem jezdíte?
(silniční/horské/citybike/silniční/trackové kolo, elektrokolo, lehokolo/kolo/koloběžka)
- Jak často?
- Styl jízdy podle:
 - povětrnostní podmínky
 - terén
 - kvalita silnic
 - profil trati
 - hustota dopravy
 - profil křižovatky
 - stoupání
 - vzdálenosti
 - klasifikace překážek
- Kam? (Popsat detailněji trasu – vzdálenost, vlastní trasa, četnost, alternace trasy, zastávky na trase, různé trasy tam a zpět?)
- Kolik tras znáte?
 - Mate nové nebo pravidelné trasy, pro které si trasu musíte naplánovat?

Appendix B: Prototype questionnaire

- Potřebujete se podívat někdy do mapy (do instrukcí) na info o trase?

Plánování tras a navigace

- Plánujete si trasy a jak?
 - Plánujete dopředu nebo až “u kola”
 - Používáte nějaké aplikace? Popiště přesně, jak je používáte.
- Používáte něco na trase (jak se navigujete)?
 - aplikace

Co byste ocenili při plánování a při jízdě, aby to bylo:

- komfortnější
- bezpečnější
- rychlejší

Appendix C

Testing tasks

1. Chcete si naplánovat novou trasu.
2. Jako startovní pozici si chcete nastavit Vaši aktuální pozici.
3. Rozmysleli jste si startovní pozici, nyní chcete začínat v ulici Technická.
4. Jako svůj cílový bod si chcete nastavit Karlovo náměstí s číslem popisným 13.
5. Nikam nepospícháte, nemáte rádi rušné ulice, a tak si nastavíte odpovídající profil trasy.
6. Vyhledejte trasu.
7. Zjistěte, jak je trasa dlouhá a za jak dlouho byste se mohli dostat do cíle.
8. Je možné, že se na trase setkáte s nějakými obtížemi. Zjistěte, jaké obtíže to jsou.
9. Z mapy zjistěte, po jakém typu cesty pojedete nejčastěji.
10. Zjistěte, v jaké vzdálenosti od startu budete projíždět ulicí U Písecké brány.
11. Zjistěte, kolik metrů celkem pojedete po cyklostezkách.
12. Zjistěte, v jaké vzdálenosti od startu se budete nacházet v nejvyšší nadmořské výšce a jaká výška to je.
13. Nyní trasu nechcete absolvovat, ale chcete si ji uložit pro pozdější použití. Uložte si trasu.
14. Zobrazte znovu tuto trasu.
15. Přepněte se do navigace.
16. Spusťte navigaci.
17. Nejste si jisti, jak dále pokračovat, zobrazte si itinerář a prohlédněte si následující instrukce.
18. Předčasně ukončete navigaci.
19. Zjistěte, kolik kilometrů jste skutečně ujeli.

Appendix C: Testing tasks

20. Zítřa chcete navštívit Planetárium. Víte, že se Planetárium nachází v ulici Královská obora 233. Uložte si toto místo, abyste ho mohli zítřa jednoduše použít pro vyhledání trasy.
21. Nacházíte se u Planetária a chcete naplánovat trasu zpět tam odkud jste přijeli.
22. Smažte Planetárium z uložených míst.