ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Černý**	Jméno: **Pavel**	Osobní číslo: **378369**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Studijní obor:	**Softwarové inženýrství**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Lokalizace zrakově postižených chodců pomocí dialogového systému

Název diplomové práce anglicky:

Localization of visually impaired pedestrians by means of a dialog system

Pokyny pro vypracování:

Design and implement a prototype of mobile navigation application based on a dialog system for visually impaired users. The mobile navigation application will use existing dialog system and GIS. Focus on efficient communication of the mobile navigation application with the user by means of a dialog system for precise localization. Conduct a user study with the user group to evaluate usability of the prototype.

Seznam doporučené literatury:

Bui, Trung H., et al. "A POMDP approach to Affective Dialogue Modeling." NATO SECURITY THROUGH SCIENCE SERIES E HUMAN AND SOCIETAL DYNAMICS 18 (2007): 349.
May, Andrew J., et al. "Pedestrian navigation aids: information requirements and design implications." Personal and Ubiquitous Computing 7.6 (2003): 331-338.
Ungar, Simon. "13 Cognitive mapping without." Cognitive mapping: past, present, and future 4 (2000): 221.
Balata, Jan, Zdenek Mikovec, and Ivo Maly. "Navigation Problems in Blind-to-Blind Pedestrians Tele-assistance Navigation." Human-Computer Interaction&#8211;INTERACT 2015. Springer International Publishing, 2015. 89-109.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Balata, Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **13.01.2016** Termín odevzdání diplomové práce: **26.05.2017**

Platnost zadání diplomové práce:
do konce zimního semestru 2018/2019

[illegible] [illegible] [illegible]

Podpis vedoucí(ho) práce Podpis vedoucí(ho) ústavu/katedry Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání Podpis studenta

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

Master's Thesis

Localization of visually impaired pedestrians by means of a dialog system

Bc. Pavel Cerny

Supervisor: Ing. Jan Balata

Study Programme: Open Informatics, Master

Field of Study: Software Engineering

January 9, 2018

Aknowledgements

Thanks to my flat-mates for their support especially in the final hours. Thanks to my family for their support during the whole project. And thanks to my supervisor especially for kickstarting my thinking in the beginning.

Declaration

Abstract

This diploma thesis deals with the localization of visually impaired pedestrians. The thesis analyses the GPS signal in the city, available GIS and briefly analyses the spoken dialogue interface on mobile devices. After 3 iterations of prototypes, this thesis finds no usable strategy for a localization based on the natural description of the user's surroundings. Instead, it proposes and implements navigation based on the continuous collecting of GPS, or navigation using IBM Watson Conversation to acquire the information about the POI in the user's surroundings. This thesis implements and tests these methods. The thesis demonstrates, the implemented methods, can't be based on standard Geolocation APIs and open data.

Abstrakt

Tato diplomová práce se zabývá lokalizací nevidomých chodců. Práce analyzuje chování GPS signálu ve městě, dostupné podklady GIS a stručně analyzuje mluvené dialogové rozhraní pro slepé na mobilním telefonu. Po 3 iterací prototypů, práce nenachází žádnou použitelnou strategii pro lokalizaci založenou na přirozeném popisu prostředí uživatele a místo toho navrhuje a implementuje lokalizaci založenou na průběžném sběru GPS souřadnic, nebo zadání POI. Tyto metody lokalizace, dále testuje a implementuje. Práce ukazuje, že implementované metody nemůžou používat běžně dostupná API a otevřená data.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The blind and other visually impaired people want to be self-sufficient. However, traveling from one point in the city to another is very challenging for them. They have to study the path in advance and memorize it.

Nowadays they can rely on: asking pedestrians on the street, navigations for the sighted and navigations for the blind. None of these options is perfect. Asking people on the street is often uncomfortable and the responses lacks the quality[1]. Navigations for sighted pedestrians, such as Google Maps[2] and Apple Maps[3], do not offer sidewalk-based navigation, which blind users need. Instead, they guide users only through the middle of the street.

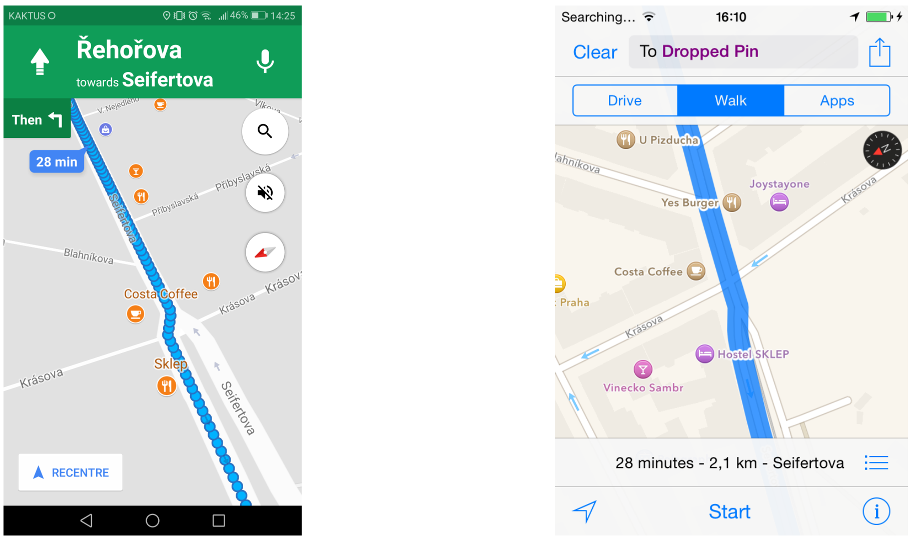


Figure 1.1: Google Maps and Apple Maps show the user just the street he should take. They provide no info about the sidewalks

Tools for blind people such as Blindsquare[4] and NotNav[5] tell the user only the direction and what points of interest are around. Users still have to somehow find, on their own, all the pedestrian crossings and which sidewalks to use. Another tool for the blind, Naviterier[6], can provide sidewalk-based navigation. However, Naviterier cannot automatically locate the user and requires them to enter an address as a starting point.

A sidewalk-based navigation would need a very exact initial location. The present GPS sensors cannot achieve sidewalk precision in urban city areas. With their current accuracy, it is never clear which side of the street or which corner of a crossing the user is standing on. Naviterier therefore requires the user to enter an address as a starting point. However, entering an address can be difficult. For example, when you have just arrived at a tram stop or when you are in a place you don't know well.

On the other hand, we know that blind people can efficiently describe their position over the phone to another blind person. So well in fact, that the second blind person, familiar with the area, can determine their precise position and point it out on a tactile map[1]. With the rise of chatbots like Apple's Siri, Amazon's Alexa or Microsoft's Cortana[7, 8, 9] in recent years it became clear that simple spoken interface could offer an elegent solution for this problem.

In this thesis, I will design and implement a mobile navigation application prototype for the blind. It will correctly localize the person and provide a sidewalk-based navigation to the target. The initial localization process will take advantage of the ability of blind people to describe their position over a phone, using a scripted chatbot to determine a precise position on a sidewalk and to continue sidewalk-based navigation to the target.

In practice this system would grant the blind and other visually impaired more self-sufficiency, shorten their preparation time for traveling and make it more comfortable.

1.2 Goals of the thesis

1. to enhance the accuracy of localization by a dialogue interface to achieve sidewalk-level of precision.
2. to design a navigation app with a dialogue interface.
3. to evaluate usability of the app with blind users

Chapter 2

Analysis

2.1 GPS localization

Almost every smartphone features the satellite navigation. But the precision of the phones' navigation is not sufficient to estimate the user's sidewalk and the user's position on this sidewalk.

A large experiment[10] demonstrated the smartphones have a mean accuracy 4.9 meters. And 4.9 is a mean value from that experiment. In the urban areas, the inaccuracy definitely exceeds 4.9m.

2.1.1 GPS in the city

The accuracy worsens when close to the buildings. The narrow streets and high buildings of the city create so-called *urban canyons*. The walls of these canyons reflect the signal and limit the number of visible satellites.

In general, the satellite navigation systems as GPS[11] and GLONASS[12] needs direct visibility to the satellite. The satellite sends a signal to the phone and measures the time how long the signal travels[13]. The device collects the signal from 4 or more satellites and computes the exact position.

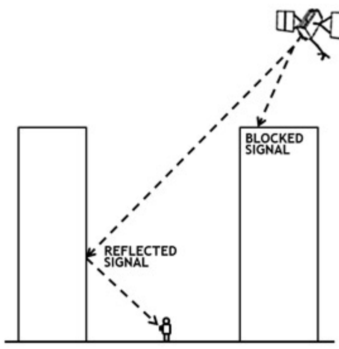


Figure 2.1: GPS signal in an urban area. Source: GPS.gov[14]

As you can see in the figure 2.1 the reflected signal travels longer then a blocked signal should be traveling. This behavior causes the inaccuracy in the calculating of the user's position. Unfortunately, this behavior can't be detected, and the only possibility to reduce the error would be using the signal from more satellites. However, the walls of the *urban canyon* reduce the number of visible satellites to a minimum.

2.1.2 GPS one-time localization

I was curious about the accuracy in the city. Therefore I decided to run a benchmark. I walked 1.1km long path around the Charles Square in Prague and nearby surroundings. Two phones[1] logged the GPS position[2] and estimated the accuracy every minute of the walk. This way I collected 23 recordings and compared them with the real position I walked through. Then I computed the difference between the estimated accuracy and the real accuracy.

	HTC desire X accuracy			BlackBerry Q10 accuracy		
#	estimated (m)	real (m)	diff (m)	estimated (m)	real (m)	diff (m)
1	48	21	-27	32	91	59
2	37	9	-28	76	46	-30
3	49	170	121	10	10	0
4	45	45	0	56	20	-36
5	51	51	0	58	20	-38
6	25	25	0	10	34	24
7	36	36	0	71	40	-31
8	46	46	0	103	62	-41
9	47	46	-1	62	62	0
10	8	8	0	62	62	0
11	35	8	-27	62	30	-32
12	-	-	-	50	15	-35
avg	38.8	42.3		54.3	41.0	
max	51	170		103	91	

This experiment demonstrated

- real average accuracy 42.3m and 41.0m in the city. (cold start)
- It demonstrated we could rely on the estimated accuracy as an upper bound (10 out of 11 cases) and (10 out of 12).

2.1.3 GPS continuous localization

You are not able to determine user's sidewalk from one GPS coordinate. But when the user starts moving and walks you can use the advantage of the knowledge that all

[1] HTC Desire X - Android, BlackBerry Q10 - BB10
[2] I used GPS Logger[15, 16] app for Android and BlackBerry 10

coordinates belong to one sidewalk. I wanted to see how the recorded data looks for a continuous movement.

I did an experiment. I walked in 6 locations around the Charles Square in Prague and nearby surroundings. One the phone[3] was running Firefox browser and inside a web application. The application collects the coordinates and draws it on top of the surrounding sidewalks[4]. I always walked around 30m and then went around the corner and another 30m.

I visually noticed, there is a strong correlation between the walked sidewalk and the shape & position of the recorded path. See figures 2.2, 2.3 and 2.4. The green dots are the recorded GPS coordinates. The red strokes are the sidewalks I used.

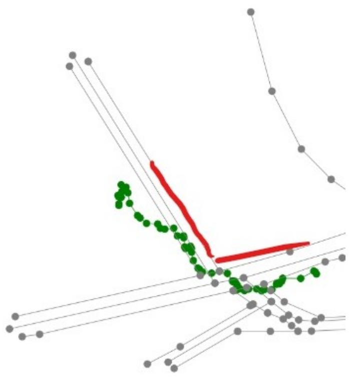


Figure 2.2: a

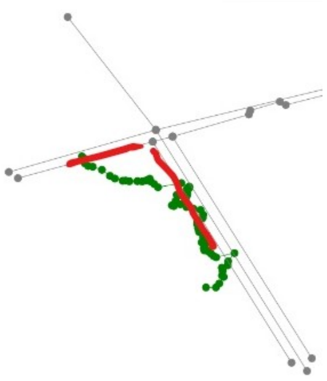


Figure 2.3: b

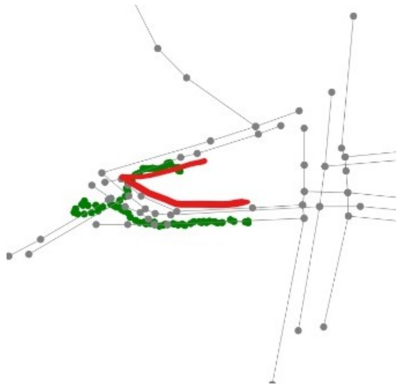


Figure 2.4: c

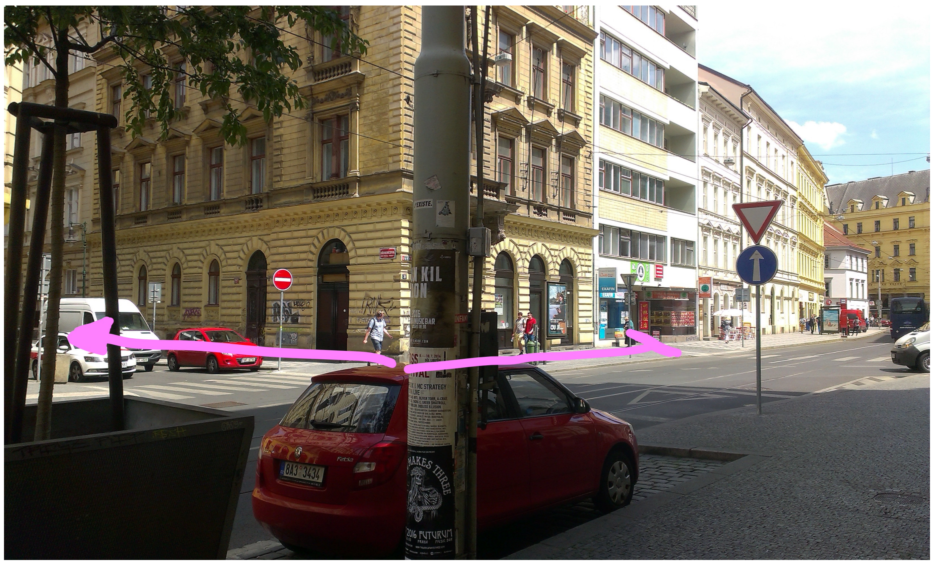


Figure 2.5: E.g., I walked the pink path and got the Figure 2.2

It seems, when the user walks around the corner, you can estimate his sidewalk.

[3]HTC Desire 500 - Android

[4]The sidewalks are provided by the FindSourceDataAPI[17]

2.2 Interface

We have chosen the conversational interface. Speaking to a machine and machine speaking back.

A graphical user interface with an accessibility mode on has a lot of overhead. e.g., “Send; button double tap to activate” or “Address; text field, double tap to edit.” Thus it is not very convenient to use. As well a graphical interface needs to be read aloud by the TTS accessibility tool.

A headphone based interface such as auditory display [18], providing the 3D position of the sound needs using of headphones. The blind people don’t like using the headphones in the city. It limits their hearing skills, e.g., incoming car, or tram and limits the auditory exploration of surroundings.

Dialogue interface. Balata et al.[19] propose using a chatbot. They propose using a POMDP based dialogue system. Such systems provide benefits of dealing with errors in speech recognition and adapting to users emotional responses[20]. On the other hand, POMDP system is difficult to train[21]:

- It needs a lot of training dialogues - 10^5
- Giving immediate reward while online training
- Computationally expensive training - too many states in the real world.
- Difficult to build for non-experts

In my thesis, I can’t get 10^5 dialogues. I can’t build a user simulator, according to paper[21] it’s very difficult to build one without biases to certain behavior. In my thesis, I just need to validate if the chatbot solution is a way to go. I need a quick engineer friendly solution to validate:

1. we can locate a user this way
2. the prototype is usable for the user

2.2.1 IBM Watson Conversation

I decided to use IBM Watson Conversation[22]. IBM supported using the Watson Conversation by offering a free student license. The IBM Watson Conversation offers simple API, which allows me to develop the prototype faster.

2.3 Interface of the frontend

2.3.1 Blind texting

Blind texting depends on person to person. I met a person who was typing faster than me with a braille keyboard. And I met a person writing very clumsy and having troubles to orient on a QWERTY keyboard. Both persons were using their own iPhone.

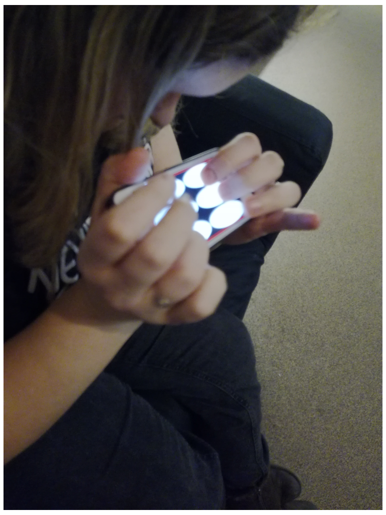


Figure 2.6: The first participant is typing on a braile keyboard

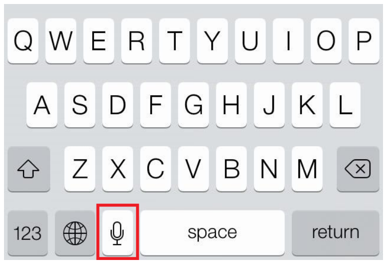


Figure 2.7: The second participant struggled to find the dictate button on a QWERTY keyboard

The first one was very comfortable with texting, wrote even using text smileys, e.g. ":)". But she needed both hands to type. See the picture 2.6.

The second person tried to use the keyboard, didn't felt comfortable with the writing and then tried to locate the button dictate on the keyboard. He had to search for a while until he found the button. See the picture 2.7

Conclusion: The users need very smooth access to dictate, and be allowed to choose between typing and dictating.

2.3.2 Up-to-date chat platforms

It's always good when you don't have to learn something new. Therefore I considered implementing the navigation as a chatbot to a WhatsApp[23], or Messenger[24].

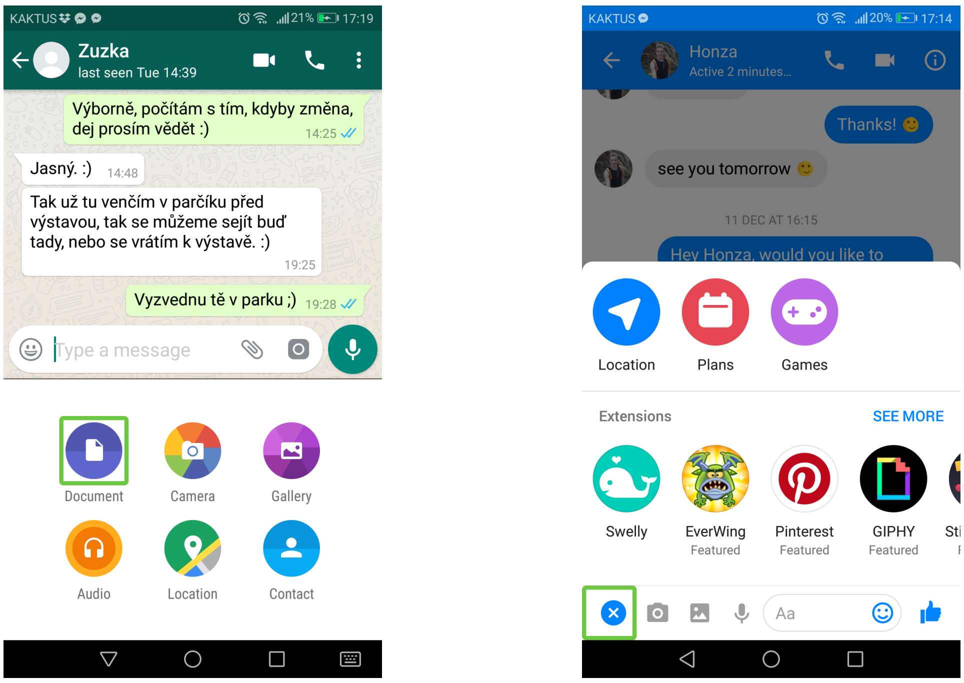
I asked 2 blind persons to try to text with me through WhatsApp and Messenger. Participant 1 (P1) was very fluent in What's App. P2 had several troubles on Messenger. He wanted to type, but he opened smileys unintentionally but didn't know about it. He tried to type, but the smileys lacked accessibility, and the screen reader didn't tell anything to the user.

The blind people use WhatsApp and Messenger, but neither of them has good accessibility for them. The screen is filled with a lot of clutter.

WhatsApp: There are 10 interactive elements on a default chat screen.

Messenger: There are 12 interactive elements on the default screen; plus each message in the chat can be interacted, thus adding even more interactive elements.

The number of elements even increase, when the user unintentionally opens some submenu, e.g., smileys, pictures, and videos. Some of those sections lack accessibility.


Zuzka
last seen Tue 14:39
Výborně, počítám s tím, kdyby změna, dej prosím vědět :) 14:25
Jasný. :) 14:48
Tak už tu venčím v parčíku před výstavou, tak se můžeme sejít buď tady, nebo se vrátím k výstavě. :) 19:25
Vyzvednu tě v parku ;) 19:28

Honza
Active 2 minutes...
Thanks!
see you tomorrow
11 DEC AT 16:15
Hey Honza, would you like to

Figure 2.8: Example of WhatsApp's and Messenger's screens cluttered with the interactive elements, which the blind users don't need.

Conclusion: I propose simple interface. A stand-alone app:

- without smileys
- with simple, accessible Voice recognition

2.4 Platform

I will design a hi-fi prototype as a web app running in the web browser. The web browser will guarantee cross-platform and cross-device compatibility.

I will develop and test on an Android phone. The iPhones, Windows mobile, and BlackBerry10 phones offer nowadays similar functionality. Therefore it would be just a matter of a bit of additional time effort to ensure it will work on them.

Nokia Symbian phones do not support Geolocation in the browser[25]. If the final prototype is based on GPS localization, they will not be supported. I didn't explore the possibilities of BlackBerry OS 6.0.

2.5 Current navigation solutions

2.5.1 How it works for sighted pedestrians

Navigation apps for sighted pedestrians use pedestrian sight to analysis the surroundings. They show the user which streets he should walk through. The user has to look around and compare what he sees on the screen with what he sees in his surroundings. Then he can decide which path to choose.

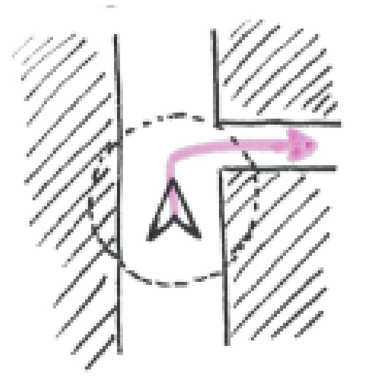


Figure 2.9: The navigation shows this screen.

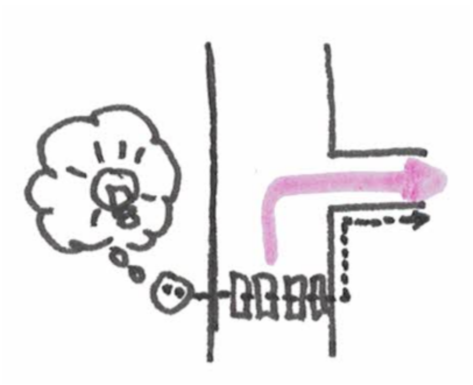


Figure 2.10: ...and the user finds how to do it.

2.5.2 Why it doesn't work for blind pedestrians

The blind pedestrians can't look around and see a sidewalk. They can explore and describe only a tiny space around them. They need to have much more detailed navigation. They need to be guided on which pedestrian crossing to cross and what line they should follow (e.g., follow the house on your right hand).

The well-known ones are Google Maps[26], Apple Maps[3] and others based on the Open Street Maps[27]. These well-known ones use improper routing strategies and lack the detail. First, the routing strategies guide the person through the middle of the street, instead of through the sidewalks. Second, they can't deal with the blind's "inability to cross open spaces (e.g., large squares)"[19] Third, Balata et al.[19] say their map sources as well lack the sufficient detail or the quality:

> The available description may be imprecise (e.g. missing sidewalks or missing handrails), or may be ambiguous (e.g., an inadequate description of pedestrian crossing, meaning that it cannot be localized and identified without visual feedback) or it may ignore specific navigation cues (e.g., the surface structure of the sidewalk, acoustic landmarks such as the specific sound of a passage, the traffic noise of a busy street, or other sensory landmarks, such as the smell of a bakery).

Therefore we can't rely on those maps.

2.5.3 Naviterier

Naviterier[6] according to poslepu.cz[28]:

> Naviterier is a navigation system independent of the GPS. It works with an accurate network of sidewalks, pedestrian crossings, shapes of corners of the houses and directions of the noise of noisy streets. Naviterier is based on special map data prepared by company CEDA, a.s.

I should, therefore, be able to build my localization method based on:

- shapes of the corners
- direction of the noise of the noisy streets
- sidewalks
- pedestrian crossings

2.5.4 Naviterier API

The data and navigation of Naviterier[6], can be accesed through the following APIs:

- FindRoutes[29]
- FindSourceData[17]
- GetAddresses[30]
- GetPois[31]

2.5.4.1 FindRoutes

This API FindRoutes[29] can find a route between two address points. It provides an itinerary of that route.

Conclusion: This API will launch, once the app localizes the pedestrian's position on the sidewalk. It brings one drawback. The app needs to estimate the nearest address to the user's estimated position. This way it decreases the accuracy of my localization. But I guess in further versions of the API it would be possible to navigate from a position on a specified sidewalk.

2.5.4.2 FindSourceData

This API FindSourceData[17] provides a data for given radius e.g. $radius = 50m$. It provides a list of Addresses, GPS coordinates of their entrances, crossways, all paths (sidewalks, underground paths, roads, stairs, walkways, etc.), points of interest and GPS coordinates of those.

This API FindSourceData doesn't provide the promised shapes of the corners. Neither I was able to de-code which streets are considered noisy and the tilt of the sidewalks.

And last, from previous experiments[1] we know, a blind pedestrian can describe the environment to another blind over a phone call. And the localizator can locate him when he knows approximately where he can be. But the blind pedestrians in that experiment used the material of the sidewalk (e.g., cobbles), the material of the doors, parked cars, columns of traffic signs. Such data are not available in this API. And probably never will, because they would be difficult to collect and maintain.

Conclusion: I am going to add the types of the corners on my own, I am going to declare the noisy streets on my own. I probably can't use the parked cars, materials of the sidewalk, doors and neither columns of the traffic signs.

2.5.4.3 GetPois

This API GetPois[31] returns the list of all points of interest in the database. But there is only some of the POIs listed.

Conclusion: I can't even use the restaurants and POIs. There is only some of them in the database.

2.6 Blind pedestrians

A couple of notes discovered in the previous researches and I need to pay attention to:

- **They answer non-logical**
  When lost blind people can get under stress. In research[32]:

  > Q: "Could you provide me with the description of your current position?" and A: "I would rather go to the start of the track and describe the track from the beginning."

- **There is no standardized terminology**
  They call some objects with multiple names or with a metaphoric description [32].

- **They may not find a point**
  Vystrcil at al.[32]:

  > The blind person might not find a particular point, but it does not mean that the navigation point is not there.

- **They can be somewhere else than they say**[32]
  If they are on street A, but they think they are in street B and both places have the same description, they can't detect it. Then they may keep saying "I am on street B" to the system.

- **They might find a different point**[32]
  Same as previous: e.g., If they find a door, the system should verify it's really the one. e.g., ask about the material.

- **They tend to linearize curved paths**[33]
  I confirm from my experiments: If the corner of the street had a very obtuse angle, they might not notice and think the street is straight.

- **They code objects relative to their body**[33]
  Compared to that Ungar[33] says, sighted people code relative to a visually derived mental map of a room, city.

2.7 Navigation principles

May at al.[34] states it's important to use landmarks for navigations. "Continue to the traffic lights" works better than "Go 40m". Because of that, I should use in my system objects, which are easily recognized by blind pedestrians, as landmarks.

When the system localized the pedestrian, the system should provide some information about the surroundings to the pedestrian. This way the pedestrian will have more confidence and trust in the system. This is similar to providing some clues along the way while navigating, as May at al.[34] recommends.

2.8 Proposed process

2.8.1 Use-cases

User got lost: User is going to the job, the time pushes him, and he suddenly realizes he is somewhere else, then he thought he is.

User goals: Move as fast as possible towards the job and get oriented on the way.

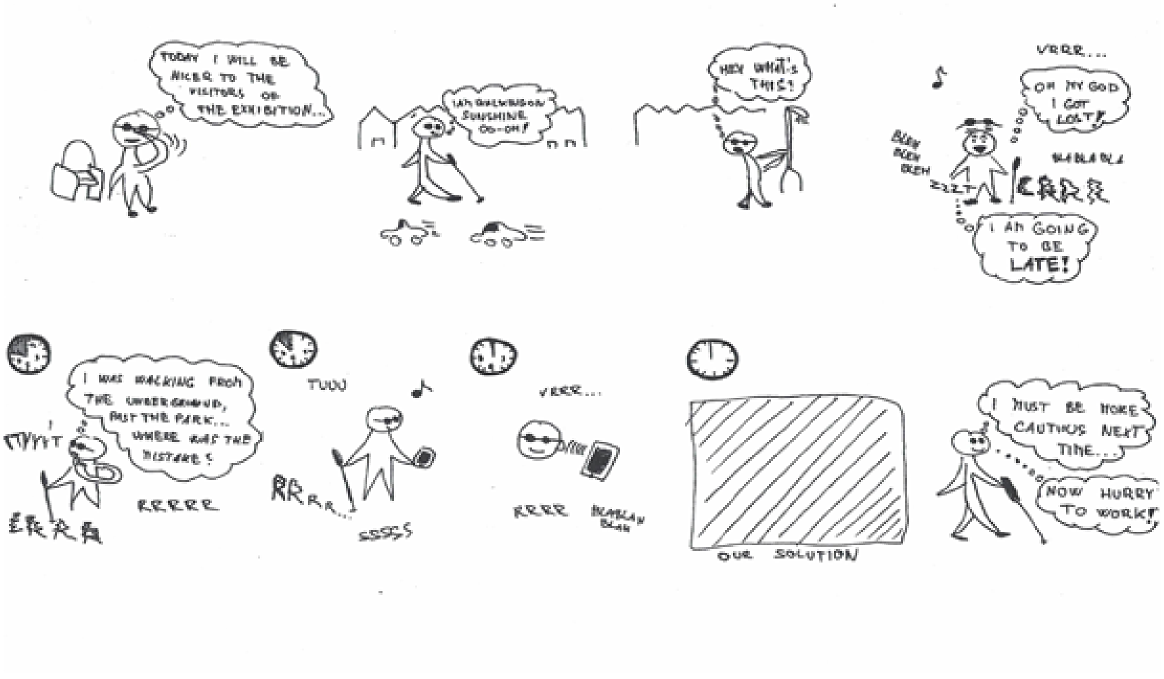

TODAY I WILL BE NICER TO THE VISITORS OF THE EXHIBITION...
SUNSHINE OO-OH!
HEY WHAT'S THIS?
VRRR...
OH MY GOD I GOT LOST!
I AM GOING TO BE LATE!
CRRR
I WAS WALKING FROM THE UNDERGROUND, PAST THE PARK... WHERE WAS THE MISTAKE?
RRRRR
TUUU
RRRR...
SSSSS
VRRR...
RRRR
BLABLA BLAH
OUR SOLUTION
I MUST BE MORE CAUTIOUS NEXT TIME...
NOW HURRY TO WORK!

Figure 2.11: Storyboard of the use-case User got lost

User exited a tram: User is going to buy a bottle of honey in a store recommended by his friend. He arrives with a tram, steps out, stands on the platform and needs to know where to go to reach the store.

User goals: Move as fast as possible towards the shop and get oriented on the way.

2.8.2 Strategy

Balata at al.[1], observed the following strategy when recovering from getting lost: 1) The user describes the previous path, 2) describes where he is now, 3) together with the helping person they find an unambiguous point and then 4) the helping person navigates the user to the target.

I am going to use the same strategy just starting directly from the point 2). I won't use the description of the previous track because the user can get lost earlier than he thinks[32]. And the description of the previous path may be therefore misleading.

2.8.3 Goals for the design

- Localize the pedestrian and guide him where he wants
  1. Determine precise sidewalks
  2. Launch pedestrian navigation
- runnable on as many phones as possible
  1. not GPS based if possible
  2. without compass if possible
- dialogue interface

2.8.3.1 Target group

age:	no-restrictions
gender:	men, women
knowledge:	can move independently with a cane
others:	smartphone user

Chapter 3

Design

3.1 Additional Researches

I performed some researches during the design process. In this section, I describe mainly thou outputs of those researches.

3.1.1 How to enhance the Naviterier's API

3.1.1.1 Noise

I wasn't able to decode what routes are considered as noisy in the output from the FindSourceData API[17]. Therefore I had to decide on my own. I checked the map (Fig. 3.1) at Mapy.cz[35], and I decided to use the yellow streets as noisy.

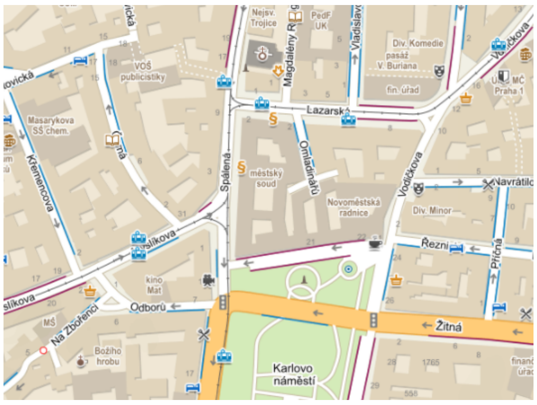


Figure 3.1: A map of the city

At the Charles Square it is:

- Žitná
- Charles square

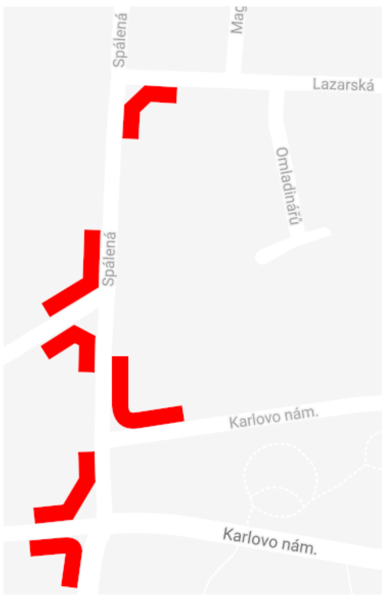


Figure 3.2: On the picture you can see the beveled corners, rounded corners, and regular corners stored in the underlying map data.

3.1.1.2 Corners

The FindSourceData API[17] doesn't provide any info about the corners of the streets. Therefore I used reverse engineering. I used the Naviterier Developer interface[36]. I always asked to find a route from some address before the corner to another address just behind the corner. The route description always mentions the type of the corner. This way I was able to discover the map data which will probably be available in the future version of the FindSourceData API.

3.1.2 Blind People's Orientation Points in the City

I conducted a walk through the surroundings of Charles square with 3 visually impaired people. They described for me, what they could recognize on the street.

The research proved: I can't rely on the reported type of the street's corner. There is a difference if you will ask "What type is this corner?" "Sharp and there is a hole inside" and if you ask for just confirmation. "Can this be a beveled corner?" "I think I can agree." (both about the same corner; see the Fig. 3.3.)

Further, the research demonstrated; **they can report a corner as a different type**. The participants mostly didn't touch the corner during walking, even with the cane. They just walked close to the wall. And therefore they estimated the corner type based on their trajectory. See the figure 3.4.

They switched the slightly-rounded corner with a beveled corner(P1); the rounded corner with a 90° corner (P1,2); the beveled corner with a slightly rounded (P2); and the beveled corner with two 45° corners (P1,3).

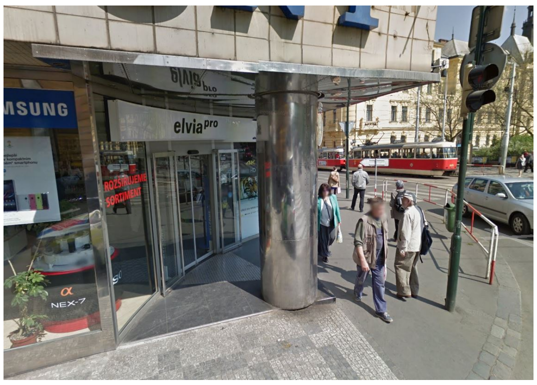


Figure 3.3: The sharp corner with a hole inside can be as well a beveled corner. Depends how you ask.

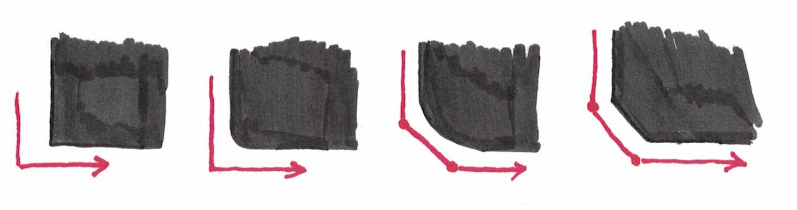


Figure 3.4: The red arrow is the trajectory, when some blind persons walks around the corner. If they don't touch the corner, and usually they don't do it, they would report it as the type of the corer based on the shape of their path.

I can't rely on detecting mild hill. As a sighted person, I analyze visually a long part of the sidewalk, and therefore I notice even if the mildest hills. The blind people analyze only a short segment with their foots and cane. Therefore I noticed of a long very mild inclination and they didn't (P1,2). On the other hand, they noticed of the transverse tilt of the sidewalk, which I didn't (P1,2).

I can't rely on the functional markings of the pedestrian crossings, P2 and P3 had several times troubles to find them.

3.1.3 Blind People and the Street Names

I took 1 blind person, walked him through the streets close to his job. He works at Charles Square. I always asked them about the street names.

He was able to describe 4 out of 8 streets. And when he had to ask someone on the street, it seemed to me the asked person had no troubles to respond. This observing leads me on the prototype *Corner of Two Streets*.

The participant pointed out several tram stops when we went around them and pointed out some buildings "a court," "a town hall." This observing led me to try entering a *POI* (point of interest) as a localization method.

The participant said about one street: "I am *not sure* about that street" - This observing led me to an idea: to provide a list of the nearby streets, and the user would be able to rehearse them, even when you do not remember exactly. I validated this idea in the prototype *POI with the Hints*.

3.2 Prototypes

I made 5 iterations of the prototypes.

iteration 1 Describe the surroundings

iteration 2 Describe the surroundings v2

iteration 3 Describe the surroundings v3

iteration 4 POI, POI with the Hints, GPS, GPS without the Compass, Corner of Two Streets

iteration 5 POI v2, GPS v2, Reverse GC

I implemented and tested them the following way:

iteration 1 dialog with user, a map of the city, 5 sighted users

iteration 2 Wizzard of Oz, a map of the city, 4 sighted users

iteration 3 Wizzard of Oz, in the city, 3 sighted users

iteration 4 Wizzard of Oz, in the city, 5 blind users

iteration 5 implemented Web app, in the city, 5 blind users

3.3 Prototypes

3.3.1 Describe the Surroundings

3.3.1.1 Describe the Surroundings v1

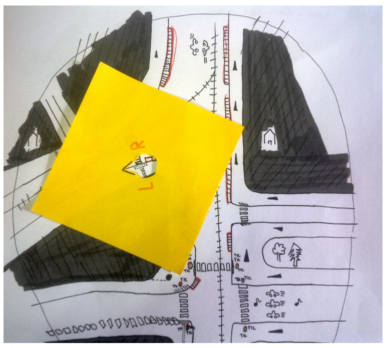


Figure 3.5: The map of the user

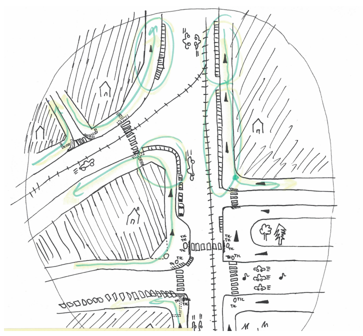


Figure 3.6: The map of the localizator

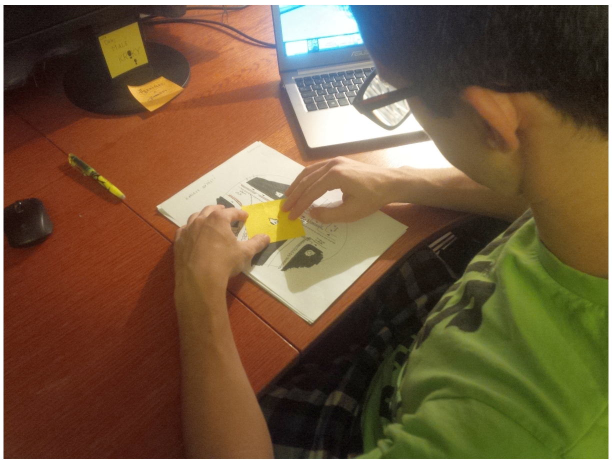


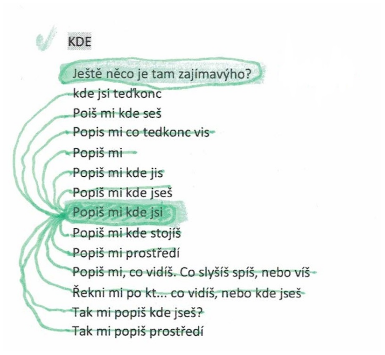
Figure 3.7: A photo from the user testing

Prototype For this prototype; I proposed a game for two players. One player was simulating a *blind* user walking in the city and the second player was playing the *system*, asking questions and trying to localize him.

Goal Observe what strategies will a human use to localize the blind user. I wanted to discover patterns and strategies, which I could later automatize.

Description The user had a task *select a place on the map*. He was given a paper with a cut-out hole representing the narrow area which a blind person can explore with his cane. The system asks, e.g., "Do you hear the trams, and where?". The user answers. When he walks, he draws on an extra piece of paper his direction and the length of the movement (simulates collecting of GPS). 5 people participated in this study. 2 men, 3 women, age 21-53. Each participant went over 4 maps with increased difficulty.

Results I was able to find in: 18 caeses out of 22. I failed in 4 cases. In 2 cases the experimenter insufficiently explained the map. In another 2 cases, the experimenter didn't notice of another possible place on the map, which as well matched the description and announced the result as the wrong one of the two possible. Participants liked the testing and did a lot of roleplaying: P4 "My cane is knocking into a grass" or P5 "wait, I have to walk a little." I collected all the system's phrases (the questions and the orders I said during the game). I clustered the phrases with the same meaning and always choose the simplest to represent the cluster. Based on this questions and orders I made a diagram of the following prototype.



KDE
Ještě něco je tam zajímavýho?
kde jsi teďkonc
Poiš mi kde seš
Popis mi co tedkonc vis
Popiš mi
Popiš mi kde jis
Popiš mi kde jseš
Popiš mi kde jsi
Popiš mi kde stojíš
Popiš mi prostředí
Popiš mi, co vidíš. Co slyšíš spíš, nebo víš
Řekni mi po kt... co vidíš, nebo kde jseš
Tak mi popiš kde jseš?
Tak mi popiš prostředí

Figure 3.8: Example of a cluster of the questions, and two questions I chosen as the representative for this cluster

3.3.1.2 Describe the Surroundings v2

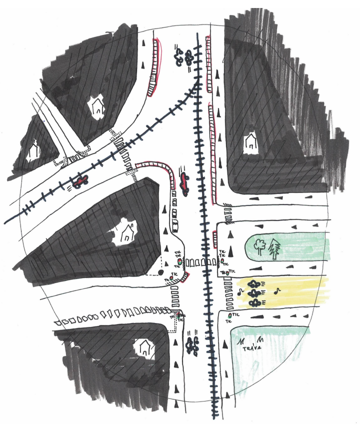


Figure 3.9: The improved map of the user

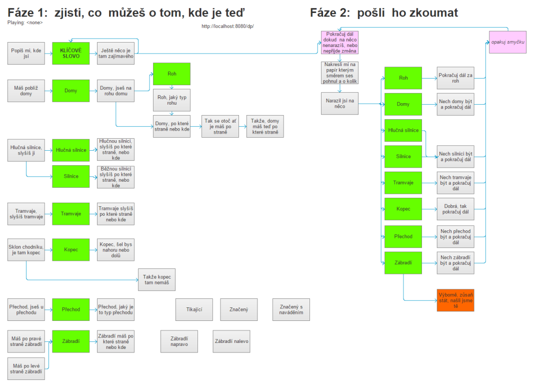

Fáze 1: zjisti, co můžeš o tom, kde je teď
Playing: <none>
http://localhost:8080/dp/
Fáze 2: pošli ho zkoumat
Popiš mi, kde jsi
KLÍČOVÉ SLOVO
Ještě něco je tam zajímavého
Máš poblíž domy
Domy
Domy, jseš na rohu domu
Roh
Roh, jaký typ rohu
Domy, po které straně nebo kde
Tak se otoč ať je máš po straně
Takže, domy máš teď po které straně
Hlučná silnice, slyšíš ji
Hlučná silnice
Hlučnou silnici, slyšíš po které straně, nebo kde
Silnice
Běžnou silnici slyšíš po které straně, nebo kde
Tramvaje, slyšíš tramvaje
Tramvaje
Tramvaje slyšíš po které straně nebo kde
Sklon chodníku, je tam kopec
Kopec
Kopec, šel bys nahoru nebo dolů
Takže kopec tam nemáš
Přechod, jseš u přechodu
Přechod
Přechod, jaký je to typ přechodu
Tikající
Značený
Značený s naváděním
Máš po pravé straně zábradlí
Zábradlí
Zábradlí máš po které straně nebo kde
Zábradlí napravo
Zábradlí nalevo
Máš po levé straně zábradlí
Pokračuj dál dokud na něco nenarazíš, nebo nepřijde změna
opakuj smyčku
Nakresli mi na papír kterým směrem ses pohnul a o kolik
Narazil jsi na něco
Roh
Pokračuj dál za roh
Domy
Nech domy být a pokračuj dál
Hlučná silnice
Silnice
Nech silnici být a pokračuj dál
Tramvaje
Nech tramvaje být a pokračuj dál
Kopec
Dobrá, tak pokračuj dál
Přechod
Nech přechod být a pokračuj dál
Zábradlí
Nech zábradlí být a pokračuj dál
Výborně, zůsaň stát, našli jsme tě

Figure 3.10: The dialogue diagram of the localizator

Prototype I built a low fidelity prototype of a dialogue system. The prototype was a flow diagram and set of pre-recorded texts. When I clicked a diagram field, the computer read it aloud. I used Axure RP[37] to create the diagram and Acapela TTS(text-to-speech)[38] to read the text aloud.

Goal I wanted to verify the questions and orders from the first experiment, if they would work even when read by a machine voice.

Description I played the game from the first experiment, but this time I was asking the questions based on the diagram logic, and the questions were read aloud by the computer. I was recording the dialogues to mp3. Then transcribed them as a walk through thorough the diagrams. 4 people participated in this study. 2 men, 2 women, age 22-25. Each participant went over 4 maps with increased difficulty.

Results I was able to locate in: 11 cases out of 16. I failed in 5 cases. 2 cases: the user wrongly understood the arrows marking a hill. 1 case: unclear what one particular order means. 1 case: the system was missing orders which would differentiate some 2 places. 1 case: troubles with the terminology of the 45° corner and the beveled corner.

Figure 3.11: A photo from the user testing iteration 2

I collected the usability mistakes and errors in the dialogue structure. And further improved the logic of the diagram.

3.3.1.3 Describe the Surroundings v3

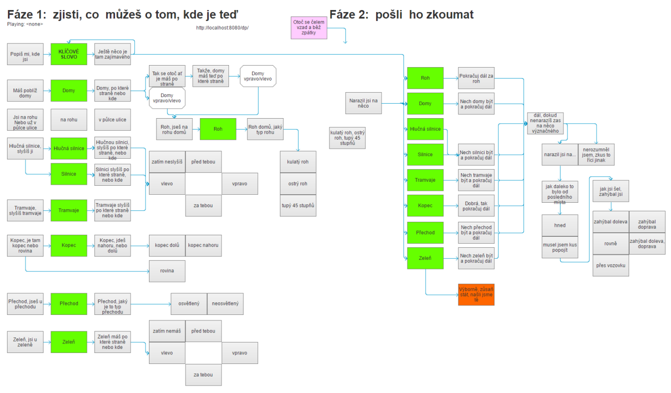


Figure 3.12: The dialogue diagram of the localizator

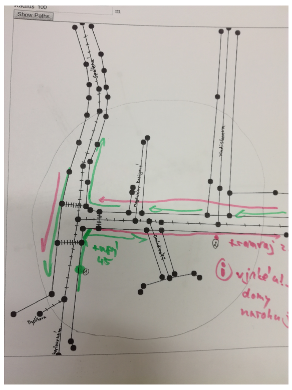


Figure 3.13: the map of the localizator. Based on the data from FindSourceData API[17]

Figure 3.14: Google Maps and Apple Maps show the user just the street he should take. They provide no info about the sidewalks

Prototype I improved the previous prototype, replaced the map for Naviterier GIS map and tested in the city. **Goal** Validate the dialogue system would work in the city with the real map from the GIS.

Description I played the game again, this time the user was in the real city. I had the map and tried to locate the person on the map. The questions were read aloud by the computer. 3 people participated in this study. 2 women, 1 men, age 22-26. Each participant went over 4 places in the city.

Results The testing proved the GIS map is insufficiently specific for the dialogue system and we need to change the strategy. I was able to locate in 8 cases from 13. I had to 4 times ask: What's the name of the 2 streets on that corner. 2 times I hit into the person replied something the dialog was not prepared for, e.g., "a hot-dog stand," "a crossing of two streets." In 3 cases I located the person on a very long segment and had no idea, where on that segment he is. I made a quick fix, which helped me to know the person is at the end of the segment. Still, I was not able to distinguish on which end of that segment.

3.3.2 Corner of Two Streets

3.3.2.1 Corner of Two Streets v1

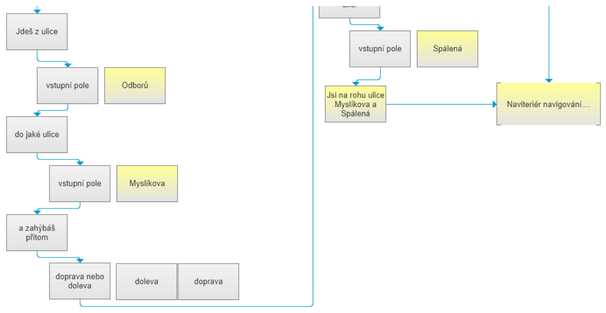


Figure 3.15: Example of a part of the dialogue diagram

Prototype The prototype asked the user to go to the corner of two street's and enter their names, which direction he would turn around the corner. This can decide the position for some crossings. If it can't be decided, he would be sent to another corner and prompted to enter the name of that street. The Prototypes were talking to blind participants through TTS in the city.

Goal Validate if this prototype is usable. Decide if I should implement it in next iteration.
Description 5 visually impaired people participated in this study. 4 men, 1 women, age 38-64. 2 born blind, 1 blind since childhood. 1 perceives dark vs. light, 1 point vision.

Results This prototype was very stressful for the users: P1: "This was, of course, the worst, I have to get too much information", P5: "I don't have time to stay and wait 4 minutes till someone helps me". Therefore I decide to skip this prototype in the next iteration

3.3.3 POI (Point of Interest)

3.3.3.1 POI v1

Prototype This prototype can locate the user at: an address, a POI or on a tram stop. The prototype was talking to the user through the TTS.

Goal Determine which prototype to implement in the next iteration.

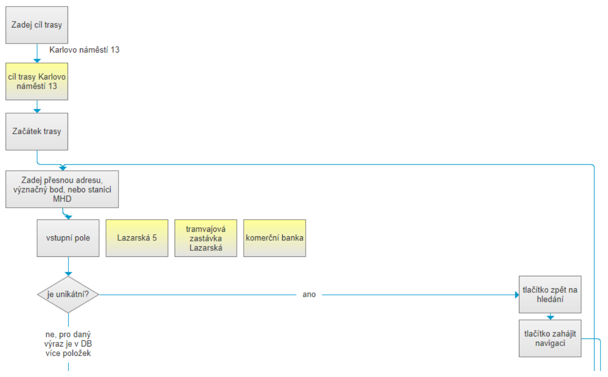
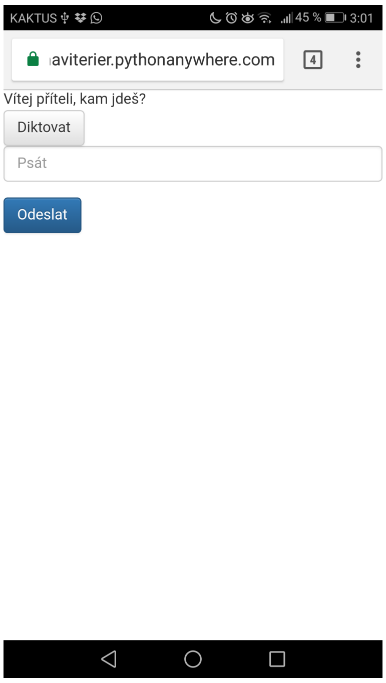

Zadej cíl trasy
Karlovo náměstí 13
cíl trasy Karlovo náměstí 13
Začátek trasy
Zadej přesnou adresu, význačný bod, nebo stanici MHD
vstupní pole
Lazarská 5
tramvajová zastávka Lazarská
komerční banka
je unikátní?
ano
tlačítko zpět na hledání
tlačítko zahájit navigaci
ne, pro daný výraz je v DB více položek

Figure 3.16: Example of a part of the dialogue diagram of POI v1

Description The user had a task "Go for the ice cream to a confectionery at the address Karlovo náměstí 13" (The first have Ke Kunratickému lesu 3). The user had a phone in his hand. He was instructed to speak on the phone, and explained how to control buttons – "when asked touch either upper part of the screen or the lover part of the screen." Experimenter observed where the user touched the screen and analyzed what user says, then he played the corresponding text of the reply. 5 visually impaired people participated in this study. 4 men, 1 women, age 38-64. 2 born blind, 1 blind since childhood. 1 perceives dark vs. light, 1 point vision.

Results The testing discovered: The users need a possibility to change the target P1. P1, P2 didn't know the terminal station of the tram. In the next version, the prototype has to allow any stop on the way in the direction of the tram. When I lead them on a tram stop, P2,3,4 and 5 had no idea what tram stop is it. P2 couldn't find even while using the pager. Make instructions & questions more user-friendly. The testing confirmed: the mental model (to estimate a tram stop: give us the tram number & direction) works

3.3.3.2 POI v2



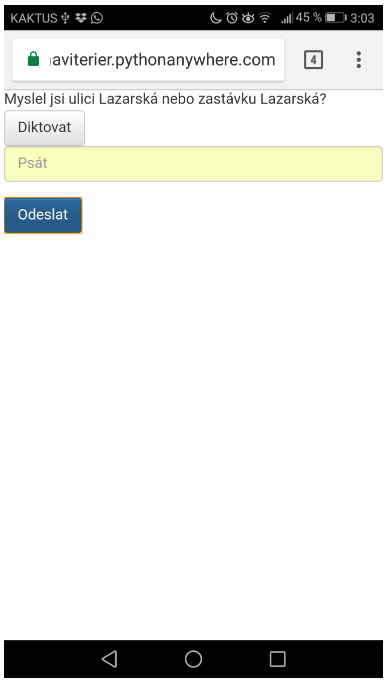

KAKTUS
45 % 3:03
ıaviterier.pythonanywhere.com
Myslel jsi ulici Lazarská nebo zastávku Lazarská?
Diktovat
Psát
Odeslat

Figure 3.17: Example screens from the POI v2

Prototype Mobile web app. The system asks user as long until it gets the street name and the land registry number, or a tram stop name, tram line and the direction of the tram (it's enough when the user mentions any stop along the next route of the tram). User's answers are sent to Conversation API[22], which understands the context and information and generates the appropriate response. An example could be:

> Wellcome my friend, where are you going? Myslíkova 22 You are going to Myslíkova 22. What is your current address or current stop of public transport? Tram Národní třída You are on stop Národní třída. Let's determine exact platform. What line and in which direction? (any stop along the route is acceptable) 9 Národní divadlo So you are now leaving from stop Národní třída, line 9 in Direction národní divadlo

And it will extract the parameters:

- *targetStreet: Myslíkova*
- *targetHouseNumber: 22*
- *sourceLine: 9*
- *sourceStopName: Národní třída*
- *sourceNextStop: Národní divadlo*

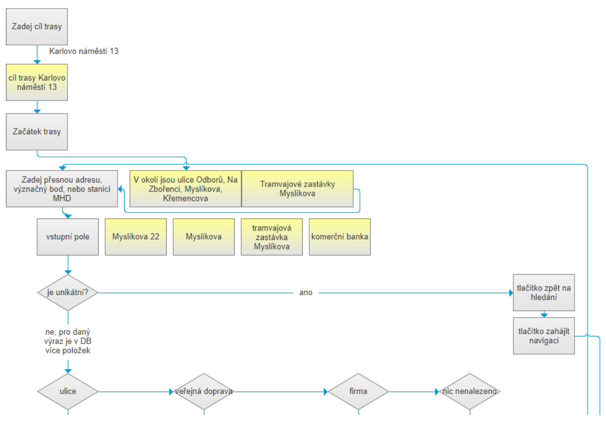

Zadej cíl trasy
Karlovo náměstí 13
cíl trasy Karlovo náměstí 13
Začátek trasy
Zadej přesnou adresu, význačný bod, nebo stanici MHD
V okolí jsou ulice Odborů, Na Zbořenci, Myslíkova, Křemencova
Tramvajové zastávky Myslíkova
vstupní pole
Myslíkova 22
Myslíkova
tramvajová zastávka Myslíkova
komerční banka
je unikátní?
ano
tlačítko zpět na hledání
tlačítko zahájit navigaci
ne, pro daný výraz je v DB více položek
ulice
veřejná doprava
firma
nic nenalezeno

Figure 3.18: Example of the dialogue diagram of the POI with the Hints v1

The triplet (*sourceLine*, *sourceStopName*, *sourceNextStop*)is then converted to GPS coordinates(My DPP API) and GPS coordinates to address (HERE API)

Goal Validate the usability of the implemented prototype. Test the voice chat interface; Test if it is usable to localize based on trams.

Description Tested with 5 visually impaired people. See the chapter Testing for more details.

Results See the chapter Testing.

3.3.3.3 POI future work

I needed to keep the complexity reduced for the prototyping. The future versions can utilize all the means of the public transport (buses, metro, trains, etc.).

3.3.4 POI with the Hints

3.3.4.1 POI with the Hints v1

Prototype A dialog system implemented in Axure + Acapela TTS. It contains the POIs from FindSourceData API[17] in the close surroundings. The prototype asks for target's address, then asks for the current position and reads aloud the list of nearby streets nearby and tram stops. The prototype allows localization of the user based on a tram stop, an address or a name of some POI, e.g., restaurant, shop, etc.

Goal Determine which prototype to implement from iteration 4. In this prototype, I wanted to learn about the usability of localization by an address or POI.

Description I tested with 5 visually impaired. The user was given the task "Go for the ice cream to a confectionery at the address Karlovo náměstí 13". The user had a phone in his hand. He was instructed to speak on the phone, and explained how to control the buttons – "when asked touch either upper part of screen or lover screen." The experimentator observed where the user touched the screen and analyzed what the user says, then he played the corresponding text of the reply. 5 visually impaired people participated in this study. 4

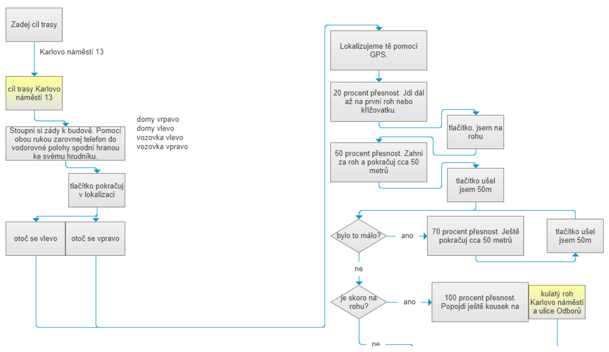

Zadej cíl trasy
Karlovo náměstí 13
cíl trasy Karlovo náměstí 13
Stoupni si zády k budově. Pomocí obou rukou zarovnej telefon do vodorovné polohy spodní hranou ke svému hrudníku.
domy vrpavo
domy vlevo
vozovka vlevo
vozovka vpravo
tlačítko pokračuj v lokalizaci
otoč se vlevo
otoč se vpravo
Lokalizujeme tě pomocí GPS.
20 procent přesnost. Jdi dál až na první roh nebo křižovatku.
tlačítko, jsem na rohu
50 procent přesnost. Zahni za roh a pokračuj cca 50 metrů
tlačítko ušel jsem 50m
bylo to málo?
ano
70 procent přesnost. Ještě pokračuj cca 50 metrů
tlačítko ušel jsem 50m
ne
je skoro na rohu?
ano
100 procent přesnost. Popojdi ještě kousek na
kulatý roh Karlovo náměstí a ulice Odborů
ne

Figure 3.19: Example of the dialogue diagram of the GPS v1

men, 1 women, age 38-64. 2 born blind, 1 blind since childhood. 1 perceives dark vs. light, 1 point vision.

Results Too much information confused all the users, In the next iteration, I had to reduce the amount of the text. The users had troubles to distinguish between when they are asked for the target and when for the source, in the POI v2 I improved the wording of those fields. 3 users thought the app would find them.

3.3.5 GPS

3.3.5.1 GPS v1

Prototype This prototype estimates the direction to the target, and send him in that direction. The user has to go until he reaches a corner. Then it asks him to turn around the corner and walk another 30m. Based on this trajectory the prototype should be able to estimate the sidewalk. I implemented this prototype as a dialogue diagram in Axure with TTS messages from Acapela Box.

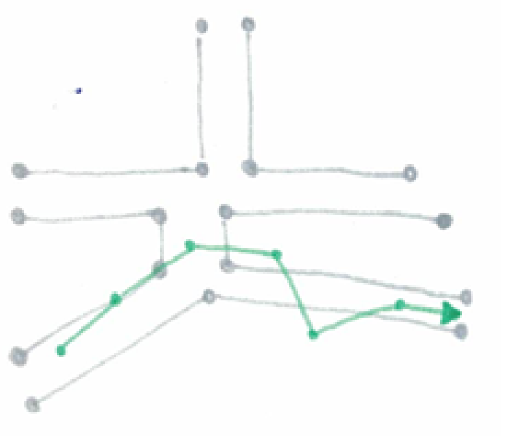


Figure 3.20: Example showing the green collected path and the gray sidewalks

Goal Determine which of the prototypes to implement from iteration 4.

Description Tested with 5 visually impaired people. The user was given a phone and instructed, when he taps the upper part of the display it means button 1, lower part button 2. In fact, the phone was turned off. The experimenter stood nearby and determined the dialogue by clicking on the diagram and the computer read the text aloud with TTS. 5 visually impaired people participated in this study. 4 men, 1 women, age 38-64. 2 born blind, 1 blind since childhood. 1 perceives dark vs. light, 1 point vision.

Results The participants liked this solution. Still, I discovered some usability problems. E.g., The device said: "Hold the phone in the horizontal position. The lower part of the phone should aim to your chest" But 2 users didn't hold the phone correctly. Compare the figures 3.21, 3.22. Therefore I skipped the instructions how to hold the phone in the next iteration.

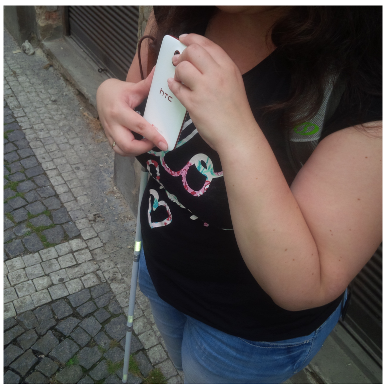


Figure 3.21: The incorrect position of the phone

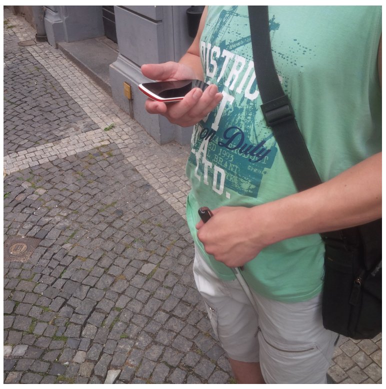


Figure 3.22: The correct postion of the phone

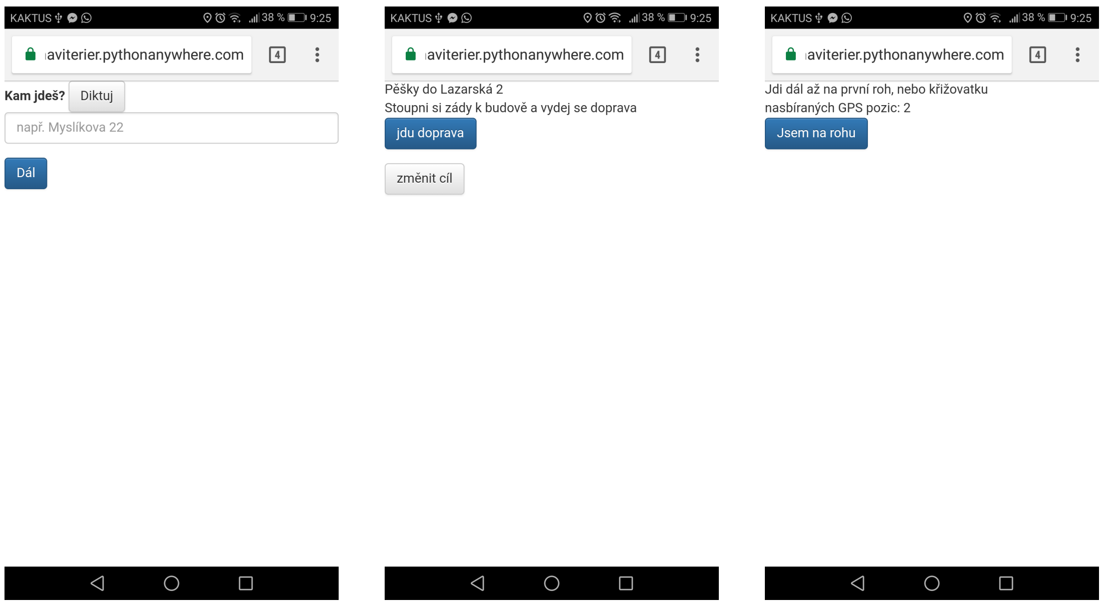
3.3.5.2 GPS v2

Prototype This prototype asks for the target address. It asks the user to stand with his backs to the wall. The compass built in the phone tells which direction is the bee-line to the target. If it' in the interval $< -90°, 90° >$ it sends the user to the right, for $(90°, 270°)$ it sends him left. Then it starts collecting the coordinates; the user walks to the corner of the street. He marks down he is on the corner. And he walks another 30 meters. Then he hits "estimate my position." His log of coordinates is then matched to the map of the sidewalks in FindSourceData API[17]. The prototype then knows the user's sidewalk and project the user's last coordinates on the sidewalk. From that position, the prototype finds the nearest address on this sidewalk and start the navigation from that address. I implemented this prototype as a Hi-Fi web app.

Goal Validate the usability of the implemented prototype.

Description Tested with 5 visually impaired people. See the chapter Testing for more details.

Results See the chapter Testing.



KAKTUS 9:25 38 %
aviterier.pythonanywhere.com
Kam jdeš? Diktuj
např. Myslíkova 22
Dál

KAKTUS 9:25 38 %
aviterier.pythonanywhere.com
Pěšky do Lazarská 2
Stoupni si zády k budově a vydej se doprava
jdu doprava
změnit cíl

KAKTUS 9:25 38 %
aviterier.pythonanywhere.com
Jdi dál až na první roh, nebo křižovatku
nasbíraných GPS pozic: 2
Jsem na rohu

Figure 3.23: Example screens from the GPS v2

3.3.5.3 GPS Future Work

In the GPS v2 prototype, to speed up the prototyping cycle, I implemented a very naive version of the algorithm. The algorithm calculates only the distance; it doesn't utilize the shape of the collected data.

3.3.6 GPS without the Compass

3.3.6.1 GPS without the Compass v1

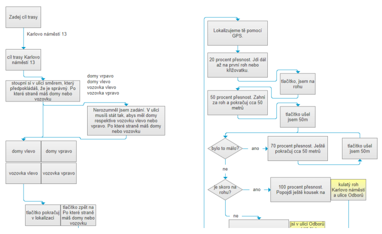


Figure 3.24: Example of the dialogue diagram of the GPS without the Compass v1

Prototype This prototype is almost the same as GPS v1, but at the beginning of the journey asks the user: "Start walking in the direction, you think goes to the target."

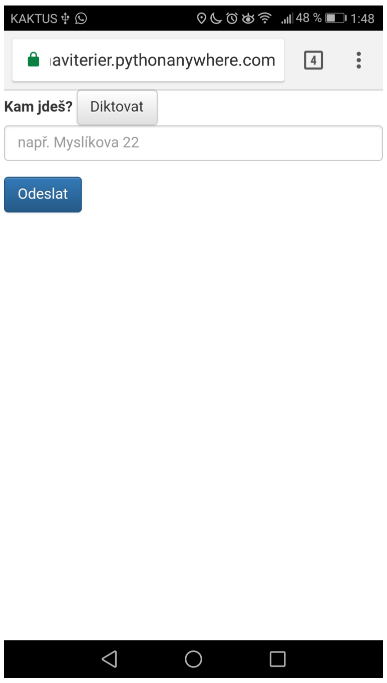
Goal Determine which of the prototypes to implement from iteration 4.

Description 5 visually impaired people participated in this study. 4 men, 1 women, age 38-64. 2 born blind, 1 blind since childhood. 1 perceives dark vs. light, 1 point vision.

Results The users had no idea which direction is the correct one in the beginning. Therefore I decided not to implement this prototype as the Hi-Fi version.

3.3.7 (Reverse Geocoding)

3.3.7.1 Revere geocoding v1



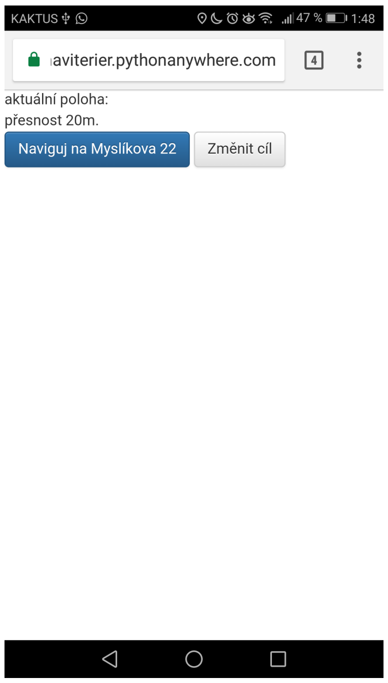


Figure 3.25: Example screens form the Reverse Geocoding v1

Prototype This prototype represents the State-of-the-Art. The prototype asks the user about the target address; then it estimates the address using the geolocation of the phone. I implemented this prototype as a Hi-Fi web app.

Goal -I expect this solution to fail because it can estimate an address on the other side of the street. But it represents the State-of-the-Art

Description Tested with 5 visually impaired people. See the chapter Testing for more details.

Results See the chapter Testing.

Chapter 4

Implementation

I implemented three prototypes as High-Fidelity prototypes. The implemented prototypes are Point of Interest (POI), GPS, and Reverse Geocoding (Reverse GC).

4.1 Technologies

4.1.1 Django

I implemented all three prototypes by use of Django framework[39]. Django is a Web framework based on language Python. Django simplifies the development of the backend, frontend, and REST API.

Originally, I wanted to write the code in Java framework Springboot 1.4.0[40] (and front end in templating system Thymeleaf[41]). However, I was very slow with the development. I was slowed down by the robustness of Java language and the Springboot framework. I always had to declare an object, whenever I wanted to share any data between frontend and backend. As well I had to implement getters and setters for all variables of that object. But I had to modify the transferred data very often.

I was learning how to use the Watson Conversation's dialogue. I was doing experiments how to control the logic of the program by the Watson Conversation, e.g., turn on the GPS, launch navigation between two points, etc. Because of those, I had to rewrite both the frontend and backend very frequently. And that was very slow and uncomfortable. Compared to that Django framework allows very quick development. The framework itself and the language Python are very brief.

Let's look at an example: You can write a simple view. The view will display just the plain text "Hello world":

```
def index(request):
    return HttpResponse("Hello, world!")
```

By changing one line, you can instead render a html template:

```
def index(request):
    return render(request, 'index.html', {})
```

Or pass some variables to the template:

```
def index(request):
    return render(request, 'index.html', {name="foo", displayGreeting=True})
```

Doing the same in Java would need much more changes in the code. What I was coding a month in Java, I built in Django in a week.

I used Django in the version 1.11.1.

4.1.2 jQuerry

The frontend communicates with the backed through jQuerry. jQuery[42] is a JavaScript library. This library simplifies the usage of AJAX and accessing of the APIs.

4.1.3 Bootstrap

I needed to display the prototypes comfortably on a mobile phone. Therefore I used the Bootstrap toolkit[43]. Bootstrap set the CSS styles and makes all the HTML elements, larger and therefore mobile friendly.

I used Bootstrap in the version 3.3.7.

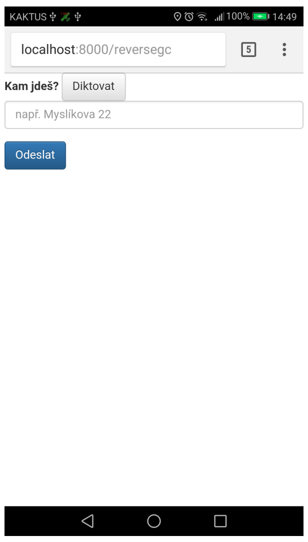


Figure 4.1: Example of a page using the Bootstrap

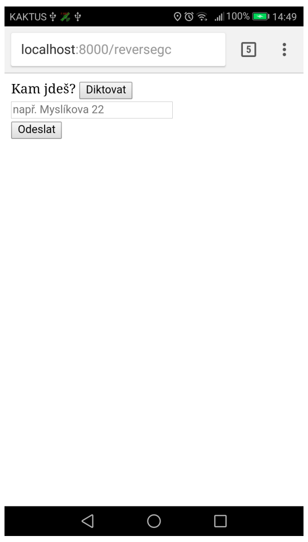


Figure 4.2: Example of a page without the Bootstrap

4.2 Prototypes implementation

I as stated at the beginning of the chapter. I implemented as hi-fidelity prototypes only the prototypes Reverse Geocoding (Reverse GC), Point of Interest (POI) and GPS.

4.2.0.1 Reverse GC

This prototype asks for the target *T*, launch phone's geolocation, estimate the user's coordinates and find the nearest address *A*. And then the prototype find a route from *A* to *T*.

The prototype is using the following parts:

- geolocation - to estimate the users coordinates
- voice input - to fill in the html input fields
- `getAddress(lat, lon, callback)` - to get an Address from the actual GPS coordinates of the phone.
- `inNaviterierDB(address, callback)` - to validate the user's estimated address & target address are in the area covered by the navigation
- `logExperiment(lat, lon, estAddress, targetAddress, experimentType, userPath, callback)` - to record the experiment for the user testing
- `redirectToNavigation(sourceAddress, targetAddress, currentUrl)` - finally navigate the user to the target

4.2.0.2 POI

This prototype has a dialogue with the user. It asks for the target *T*, then asks for the starting position *S*, get the nearest address *A* of *S*. And then the prototype find a route from *A* to *T*. The starting position *S* can be a tram stop or an address.

The prototype is using the following parts:

- voice input - to fill in the HTML input fields
- `getGpsFromTriple(stop, direction, line, callback)` - to get GPS coordinates of a tram stop
- `getAddress(lat, lon, callback)` - to get an address. The address is the nearest to the GPS position of the tram stop.
- `getEntryGPS(address, callback)` - (for the testing) to log, the GPS coordinates, in case the user entered an address as a starting point
- `inNaviterierDB(address, callback)` - to validate the user's estimated address & target address are in the area covered by the navigation
- `logExperiment(lat, lon, estAddress, targetAddress, experimentType, userPath, callback)` - to record the experiment for the user testing
- `redirectToNavigation(sourceAddress, targetAddress, currentUrl)` - finally navigate the user to the target
- text Watson

4.2.0.3 GPS

This prototype asks for the target *T*, launch the geolocation and compass and determine which direction the user should start (left/right). Then it sends the user to the corner of the street, on the corner the user turns around the corner and walks 30m. Then the prototype estimates the sidewalk S position on that sidewalk P. Then it finds the nearest address A on the sidewalk S. And then the prototype find a route from A to T.

The prototype is using the following parts:

- voice input - to fill in the HTML input fields
- geolocation - to estimate the users coordinates C in the beginning
- `getGPS(address, callback)` - to get the coordinates of the target, so it can determine the direction from C to T and say the user whether he should start to the left or the right.
- `inNaviterierDB(address, callback)` - to validate the user's estimated address & target address are in the area covered by the navigation
- `gpsFromUserPath(userPath, callback)` - to project the collected user's path over the sidewalks and find the sidewalk, which match this projection the best. And to estimate the user's position on that sidewalk.
- `getAddressOnSidewalk(lat, lon, callback)` - to find the nearest address, which lies on the same sidewalk as is the requested coordinates
- `logExperiment(lat, lon, estAddress, targetAddress, experimentType, userPath, callback)` - to record the experiment for the user testing
- `redirectToNavigation(sourceAddress, targetAddress, currentUrl)` - finally navigate the user to the target
- determine the direction in the beginning - to send the user in the correct direction, before it starts to collect the coordinates of his path
- logging GPS coordinates - to log the path of the user before the corner and after the corner

4.2.1 Parts of Prototypes

The prototypes have a lot of parts in common; some are unique for each prototype. However, all parts are described in this part.

4.2.1.1 Voice Input

The voice input allows the user to dictate his inputs. It spares the blind users from clumsy typing, and it allows using the phone with one hand. It transfers the spoken words to the text. I am using Web Speech API[44]. It's a standardized API for the browsers. And it allows the developers to write one code and use it in all modern browsers The language is set to the Czech language by the following piece of code `recognition.lang = "cs-CZ";`

4.2.1.2 `getAddress(lat, lon, callback)`

`getAddress` helps to transform any GPS coordinates to an address. It finds the address in pattern streetName houseNumber for any given latitude and longitude. `getAddress` is function using jQuerry. It accesses my API, which calls HERE API's reverse geocoding[45]. My API then extracts the streetName and houseNumber from the HERE API's response.

Originally I intended to use the Google reverse geocoding[46]. But Google for same cases worked fine, but in some cases, it returned multiple addresses unified together, e.g., Vodičkova 18-22, instead of expected single address, e.g., Vodičkova 18. I didn't figure out how to turn this feature off, so I switched to using the HERE API, which seems to behave more consistently.

4.2.1.3 `getGpsFromTriple(stop, direction, line, callback)`

`getGpsFromTriple` find the GPS position of the tram stop platform from 3 pieces of information: the stop name, the number of the tram line and the name of some next stop in the direction of the tram.

The algorithm works as the following:

```
1) find all the routes of the given tram LINE
2) find all possible head-signs on those routes
3) for each head-sign
  a) find all rides with given head-sign
  b) find a ride somewhere in the middle of the day
       (probably it will not be modified. E.g., the last ride of the day might go
        to a different terminal station to maintain the train during the night)
  c) find if the trip goes through the STOP
  d) find if the trip goes in the DIRECTION
  e) check if STOP proceeds DIRECTION
      i) return GPS coordinates of the STOP
```

`getGpsFromTriple` is function using jQuerry. It accesses my API, which then uses the data from the DPP Database.

4.2.1.4 `inNaviterierDB(address, callback)`

`inNaviterierDB` helps to verify if the requested address is valid. Let's have an example address, e.g., Vodičkova 15. It can find if the street name Vodičkova exists; if the combination of street name and the house number exists; if the combination of the street name and the land registry number exists. That way it can say to the user whether he has a typo in the street name or just in the number. As well as a valid address it's able to say the address is in the area covered by the navigation.

`getAddress` is function using jQuerry. It accesses my API. The address is validated against the Addresses Database described in section **??**

4.2.1.5 `logExperiment(lat, lon, estAddress, targetAddress, experimentType, userPath, callback)`

`logExperiment` serves the purpose of the user testing. It logs each experiment to the database. So I am later able to evaluate the error of each method. E.g., What's the difference between the coordinates estimated by the GPS sensor, the coordinates of the estimated address and the coordinates, where the user stood in reality during the experiment. (The experimentator has to enter the last coordinates manually after running the experiment.) `logExperiment` is function using jQuerry. It access my API, and it stores it in the database table `user_testing_experiment`

4.2.1.6 `redirectToNavigation(sourceAddress, targetAddress, currentUrl)`

`redirectToNavigation` is a standalone web page. It navigates the user between two addresses. The navigation instructions are provided by the FindRoute API[29].

When you provide the source address and the target address to the FindRoute API, it will provide you with up to multiple possible paths and their itineraries. In my API, I always take the first path. The possibilities are not important for my prototyping.

When you provide a misspelled, a non-existing address or an address which is still not covered by navigation, the API will return an empty list of paths. My API tests which address is wrong and reports it to the user (starting address/target address/both addresses).

`redirectToNavigation` is a function, which redirects to the page Navigate. The page Navigate access my API, which access the FindRoute API[29] and collect the navigating instructions from there.

4.2.1.7 logging GPS coordinates

The logging is split into two parts. The system logs the collected coordinates to an array `beforeCorner`, and when the user marks he stands on the corner, the system switches the array and logs to the `afterCorner`.

4.2.1.8 `gpsFromUserPath(userPath, callback)`

The prototype is collecting the coordinates of the user's traveled path. `gpsFromUserPath` then tries to find a match of the path to the network of sidewalks.

The sidewalks are received through my API. My API loads all map elements from Find-SourceData API[17]. Then it extracts only sidewalks. All roadways, pedestrian crossings, etc. are thrown away. Then it merges all sidewalk elements, which belong together, and yields them as the *complete sidewalks*. See figure 4.3

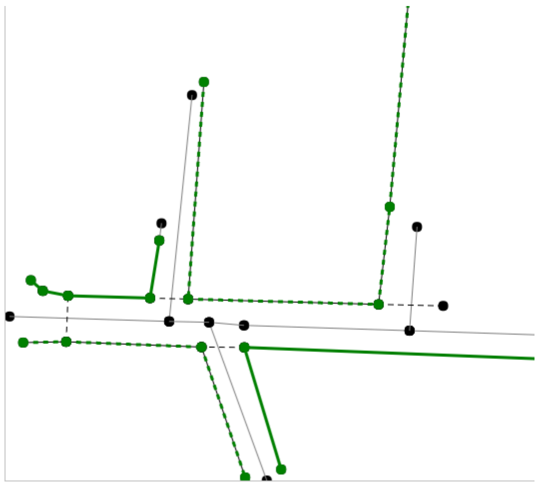


Figure 4.3: Examples of the *complete sidewalks*. Each *complete sidewalk* is either depicted with a green dashed or green solid line

Each *complete sidewalk* is then scanned to detect the corners. A corner is defined as a place on a *complete sidewalk*, which bends for an angle $\geq 140°$. The value $140°$ was determined experimentally. The *complete sidewalks* are this way split into clusters.

User's recorded path consists of two parts *before corner* and *after corner*.

The `gpsFromUserPath` then calculate the *total distance* of the *before corner* and the *after corner* from each two *adjecent clusters*. `gpsFromUserPath` then yields the two clusters with lowest the *total distance* as the sidewalk, which the user used. And project the last coordinate of User's recorded path to that sidewalk. That projection coordinates are then yielded as user's *actual GPS position*.

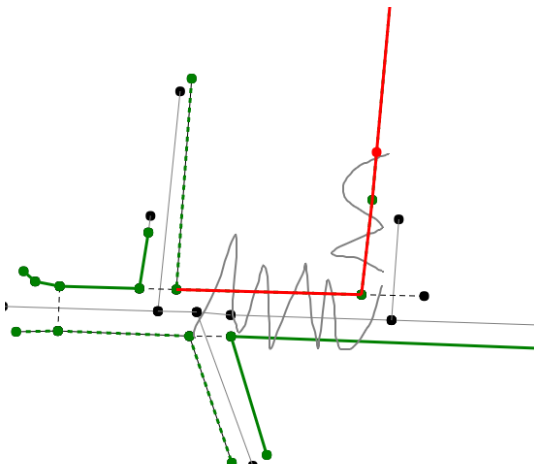


Figure 4.4: Example of a projection: the grey lines are the *before corner* and the *after corner* GPS coordinates. The red lines are two *adjacent clusters*. The red point is the estimated user's *actual GPS position*.

4.2.1.9 `getAddressOnSidewalk(lat, lon, callback)`

`getAddressOnSidewalk` finds the used sidewalk. And it finds all addresses lying on that sidewalk. And then finally it finds the address on that sidewalk, which entrance is the nearest to the requested latitude and longitude.

`getAddressOnSidewalk` is a function using jQuery, which access my API. My API then gets all the necessary information from FindSourceData API[17].

4.2.1.10 Text Watson

Text Watson is implemented in the POI prototype. It sends the user's response to the Watson Conversation's dialogue. This function as well synchronizes the dialogue with the state of the app of the POI prototype. All information is passed through object `context`. Both the dialogue and the app has access and rights to modify this object.

The dialogue sends through the `context`, the extracted start position, the target position, whether the start position is a tram stop or an address and if the dialogue collected all the required info and the navigation can be launched.

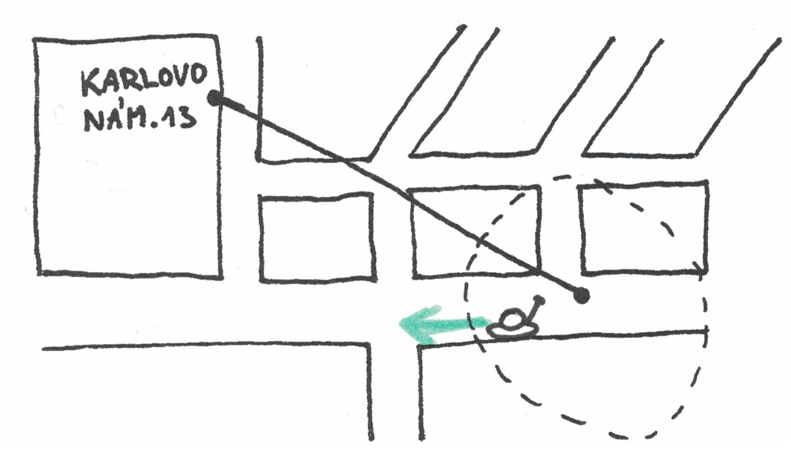


Figure 4.5: Image shows the user's real position, the area estimated by the geolocation (dashed ring), the target, e.g., Karlovo nám. 13, the azimuth to the target, and the optimal decision *start to left* (green arrow)

4.2.1.11 Determine the direction in the beginning

This serves to advise the user at the beginning whether he should start to the left or the right.

The app instructs the user to stand with his back to the building. Next, the app finds the GPS coordinates of the target and GPS coordinates of the user (from the geolocation). Then it calculates on what azimuth lies the target. And it compares the azimuth with the user's heading of the compass. Based on this comparison it determines if it's optimal for the user to start to the left, or to the right. It should be obvious from figure 4.5

4.2.1.12 Geolocation

Geolocation launches the W3C Geolocation[47] in the browser. W3C Geolocation is a standardized API available on all modern browsers. On the mobile devices, it provides the information about the current GPS coordinates, heading of the compass, and tilts of the device. In my prototypes, I am using only the GPS coordinates and heading of the compass.

4.3 Databases

My implementation uses data from two databases: DPP Database and Addresses Database. Both are described further.

4.3.1 DPP Database

I use DPP database to get GPS coordinates of any tram stop. I look for the combination of 3 elements the tram stop's name, the line of a tram and some stop on the way in the direction of the tram. These 3 elements determine the exact platform. And this database knows GPS coordinates for each platform in the city.

The DPP database contains data about the public transport in Prague. It has info about the lines of the metro, the trams, the buses, the chairlift and the ferries. Next, there is a list of all stops with their GPS coordinates, the list of all journeys of all the lines and most important, the sequence of all stops on a journey of each line.

I downloaded the data *Open Data Time Tables of Public Transport* from the Prague's Open Data Portal[48] and uploaded them to the database. The data are valid for 7 days. The data are quite big; the largest table has around 1.7 millions of entries. Therefore I imported the data only once. And I didn't implement their update on a regular basis.

4.3.2 Addresses Database

The addresses database contains a list of all addresses. And I use it to validate if the requested address exists or if it is available in the system. This prevents the user from entering, e.g., a house number which doesn't exist.

One address consists of the street name, the house number, and the land registry number.

I Every combination of the street name and house number is unique. And every combination of the street name and the land registry number is unique. Therefore it's enough when the user enters only one of the numbers. Therefore I decided to use the street name and house number only for the whole implementation.

The addresses were collected only once from the GetAddresses API[30] and stored in my database. Then, my database was enriched by a field *street_noaccents*. In this field, the street name is striped of any diacritics, spaces or special characters. e.g., "náměstí I. P. Pavlova" is stored as *namestiippavlova*. And whenever the app is searching in this database, it compares the stripped versions of the requested street name and striped version stored in this database. This technique allows the user to write without diacritics and not worry about the correct spacing.

4.4 Third-Party APIs

4.4.1 Watson Conversation

Watson Conversation[22] is an API, which allows creating a dialogue. Then this API can process the user's replies and provide him back with appropriate answers.

4.4.1.1 Intents

The documentation says, Watson Conversation can understand the *intent* of the user, e.g., User says "Karlovo náměstí 13" and Watson should understand. I am now entering a starting address. And the `address="Karlovo náměstí 13"`. It should as well extract Entities – the important facts, e.g., name of a stop. E.g. User says "Karlovo náměstí 13" `street="Karlovo náměstí""`, `houseNumber=13`.

I wanted my intents to be:

- enter the target location

- enter the actual location
- edit the previous
- enter the street, e.g., "Lazarská"
- enter the tram stop, e.g. "tram Lazarská"

In reality, the Intents don't work so well in Czech language (Watson Conversation didn't support the Czech Language at the time of the implementation). Therefore I decided to design the structure of the dialogue the way, It splits entering the target location and entering the start location in two separate steps.

The intention to edit the previous, I replaced by detecting an entity `@Back`. Whenever user mentions "back", "edit", "change", or similar. I return him to the previous question.

The intentions to enter the street or the tram stop, I replaced by the entities `@Street` and `@Stop`.

I am using another 7 entities to extract the variables.

- About the start tram stop: `@StartLineDirection`, `@StartLineNumber`, `@StartStopName`,
- About the start address: `@StartLandregistryNumber`, `@StartStreet`
- About the target address: `@TargetLandregistryNumber`, `@TargetStreet`.

4.4.1.2 Detecting the Numbers

During the implementation of my prototypes, Watson Conversation didn't support detecting the numbers. Therefore I simply prefilled the `@StartLandregistryNumber` and the`@TargetLandregistryNumber` with numbers 1-500 as the values for the entity.

4.4.1.3 Detecting the Streets

I prefilled the `@StartStreet` and the `@TargetStreet` with the list of streets. I get the list of all addresses from the GetAddresses API[30]. I filtered and uploaded only the unique names of the streets.

4.4.1.4 Detecting the Tram Stops

I prefilled the `@StartStopName` with the list of names of the stops. I filtered those manually from POIs get through the GetPois API[31].

4.4.1.5 Detecting the Tram Lines

I prefilled the `@StartLineNumber` with the list of the names of the lines. I get the list manually. I searched for all the lines going through the stops using Find a connection[49] tool at web pages dpp.cz.

4.4.1.6 Detecting the Tram Directions

I prefilled `@StartLineDirection` with a list of tram stops. For each tram line, I find the list of all its stops on the fan page TRAM-BUS.CZ[50]. Then I filtered and uploaded only the unique names.

4.4.1.7 Implementation of STEP-BACK

I hit into trouble how to implement, editing the previous response. A node of the dialogue Originally did the following: 1. it reads the reply of the user, e.g., “Lazarská” 2. it extracts the variables, e.g.`$StartStreet = "Lazarská"` 3. text the response, e.g., “You are at Lazarská. What’s the land registry number?” 4. and redirect the input to next node (or nodes).

But, when the user wants to edit the previous information and go back, he needs to hear the question from step 3) again. And he doesn’t want the system to perform the steps 1) and 2).

To achieve this behavior, I split each node of the original dialog, into two nodes:

Node 1:

1. read the reply
2. extracts variables
3. immediately jump to the Node 2
4. it’s skipped

Node 2:

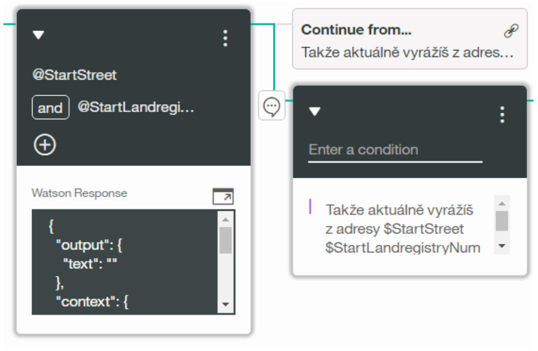
1. do nothing
2. do nothing
3. text the response
4. redirect the input

This way when going back, I can jump directly to the Node 2. This hack behaves as if I would jump on step 3) in the original node.

4.4.2 FindRoutes API

All the prototypes use the FindRoutes API[29] to get the navigation instructions. Once the starting address is estimated, this API is asked for the navigation instructions can find a route between two addresses.

This API doesn’t allow to use GPS coordinates or a sidewalk ID as a starting point. Because of that, all the tested prototypes need to determine the nearest address to the user. FindRoutes API then uses this address to navigate. Such behavior can cause errors. In the end, all three implemented prototypes rely on some kind of the reverse geocoding. And differ only in the precision of the initially estimated GPS coordinates. The reverse geocoding brings a risk of:



```
{
  "output": {
    "text": ""
  },
  "context": {
```

Takže aktuálně vyrážíš z adres...

Takže aktuálně vyrážíš z adresy $StartStreet $StartLandregistryNum

Figure 4.6: Example of one original node split into the two new nodes

- localizing an address behind the corner
- localizing an address across the street
- localizing an address on a totally different sidewalk (as the one in a parallel street)

Chapter 5

Testing

5.1 Participants Pool

I was testing all prototypes with the users. Some of the prototypes (the ones on the beginning), was tested with people with a regular sight. The prototypes in the later phases iterations 4 and 5 were tested with the blind people. All the participants received a chocolate bar of dark chocolate as compensation of their time.

5.1.1 Sighted Participants

For testing with the sighted participants, I used my friends and my family. Some other friends followed as the effect of the snowball method.

5.1.2 Blind Participants

I was collecting the contacts on the blind people just through the snowball method. As the seeds I used the Invisible exhibition[51], some personal websites I found through the Google (I am not making citations to their websites, concerning their privacy), a blind friend of my friend or I approached some of the blind people I met in the city during my errands. All these people willingly provided me with contacts on themselves and recommended their friends, who would be interested in trying this new technology.

5.2 Testing of the Prototypes

Some of the information about the testing is already in the Chapter 3 Design. In this section, I provide additional details to those testings and complete results of testing of the 5th iteration. See section 5.6.

5.2.1 1st testing - Map

participants 5 sighted people, 2 men, 3 women, age 21-53 *procedure* We showed the participants a map with the following places in the following order: Štěpánská street, Žitná street, Lazarská Street and Charles Square.

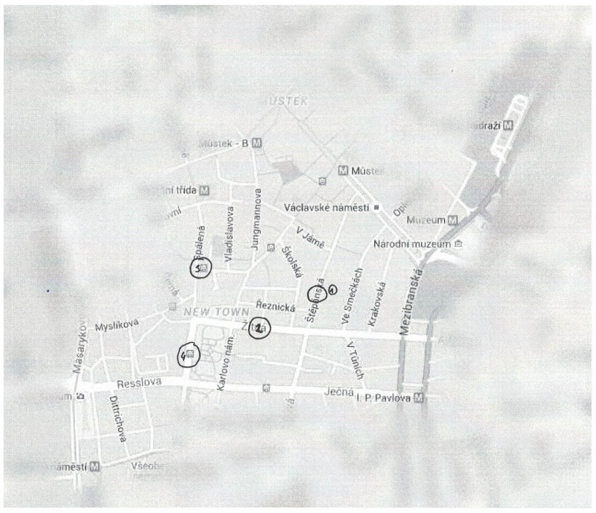


Figure 5.1: The tested locations in the city

Before the testing, I taught the participants the objects in the map. I showed the participants the Google Street View of given location when necessary, so they can better understand the map.

5.3 2nd testing - Map + Wizard of Oz

No additional info. Check the chapter 3 Design.

5.4 3rd testing - City + Wizard of Oz

No additional info. Check the chapter 3 Design.

5.5 4th testing - Blind: City + Wizard of Oz

Participants: 5 blind. 4 men, 1 women, age 38-64. 2 born blind, 1 blind since childhood. 1 perceive dark vs light, 1 point vision.

Every participant went over locations 1, 2, .., 5. Order of the methods was the following:

participant	location 1	location 2	location 3	location 4	location 5
1	POI with hints	POI	GPS w/o compass	GPS	corner of 2 streets
2	POI with hints	POI	GPS	corner of 2 streets	GPS w/o compass
3	POI with hints	POI	GPS	corner of 2 streets	GPS w/o compass
4	POI with hints	POI	corner of 2 streets	GPS w/o compass	GPS
5	POI with hints	POI	corner of 2 streets	GPS w/o compass	GPS

The position of the locations was the following:

- Location 1 Myslíkova 22

- Location 2 at Lazarská tramstop in direction Národní Třída
- Location 3 Jungmannova 5
- Location 4 V Jámně 1
- Location 5 Navrátilova 13

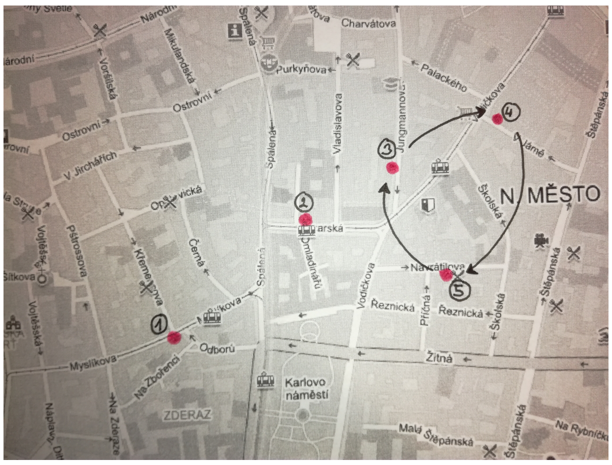


Figure 5.2: The locations and the tested prototypes on those locations

to every participant. For one participant we used his iPhone. I always led the person to the starting point and told her/him to use the app, to get to a given address. e.g., Vodičkova 28, Školská 15. The address was always different for each prototype. The participants were asked to think aloud.

apparatus All prototypes were interactive websites, running in a Chrome Browser[52] on the cellphone Honor 7 Lite, 5" screen, Android 7 Nougat. The websites were read aloud and controlled by screen reader Google TalkBack[53]. I was always standing nearby, writing notes, taking photos, taking screenshots, when something went wrong. As well, the app recorded the estimated GPS coordinates and the GPS coordinates of the estimated address (The address from which the navigation was launched)

5.6.1 Procedure order

The methods were tested in the order given by the following table:

Participant	1st	2nd	3rd
1	POI	RG	GPS
2	POI	RG	GPS
3	RG	GPS	POI
4	GPS	POI	RG
5	RG	GPS	POI

This order ensured every method was at least once tested as 1st, 2nd and 3rd method. This way I tried to eliminate the effect of the learning curve.

5.6.2 Starting places

There was no logic in selecting the starting points; they were selected randomly while walking in the city. For GPS, and reverse GC the place was chosen randomly For POI we took a tram and the exit the tram on a randomly chosen tram stop. Every session was started in a unique place.

5.6.3 Results

The method performed during the testing as follows:

- RG successfully navigated the user (0 out of 5).
- POI successfully navigated the user. (2 out of 5).
- GPS successfully navigated the user. (3 out of 6).

- The method `dpp.views.getGpsFromTriple` didn't find any tram connection for given combination (1 out of 7).
- The position of the tram stop is misplaced in the DPP Database. (1 out of 7).
- The method `getAddressFromProjection` didn't return any address (1 out of 6).

5.6.3.1 General Usability Problems

Implementation in the Browser. P2 and 3 opened the cards and had troubles to close them.

5.6.3.2 Reverse geocoding

participant	navigation	address	technically
1	failed to launch	across the street	bug Google API
2	failed to launch	across the street	bug Google API
3	wrong instructions	across the street	ok
4	wrong instructions	across the street	ok
5	failed to launch	failed to estimate	bug HERE API

Case 1, located an address across the street (Vodičkova 15, instead Vodičkova 22) plus `apis.google_api.getAddress` returned the information in an unexpected order

Case 2, located an address across the street (Vodičkova 23, instead Vodičkova 28) plus `apis.google_api.getAddress` returned the information in an unexpected order

Case 3, located an address across the street (Vladislava 6 instead Vladislava 5)

Case 4, located an address across the street (Národní 3 instead of Národní 6)

Case 5, `apis.here_api.getAddress` responded without the field 'HouseNumber'. It returned only the street name 'Masarykovo nábřeží'

The reverse geocoding failed to navigate for all 5 cases.

It suffered from technical issues. The prototype failed to estimate an address in 1 case because of unexpected behavior of HERE API[45]. In the other 4 cases, the prototype estimated the address on the side of the street. The prototype managed to launch the navigation for participants 3 and 4. But both participants suffered from getting lost during the navigation, because of the wrong initial address.

Usability mistakes: It doesn't say that the prototype starts to work without the GPS, once it switched to the navigation. P1 correctly recognized, the navigation goes from another side of the street, but when it kept saying the wrong orders, he started to believe the prototype is right and he is must be wrong.

5.6.3.3 POI

participant	navigation	address	technically
1a	failed to launch	failed to estimate	user entered false data
1b	failed to launch	failed to estimate	bug Google API
2	ok	ok	ok
3a	failed to launch	failed to estimate	bug getAddressfromTriple
3b	ok	ok	ok
4	failed to launch	failed to estimate	bug HERE API
5	wrong instructions	48m away	ok

Case 1a, user inserted a direction of the tram, which the tram never goes. Therefore it didn't located him.

Case 1b, method `apis.google_api.getAddress` returned 'NovéMěsto' instead of e.g. Vodičkova 15.

Case 3a, method `dpp.views.getGpsFromTriple` didn't find any tram connection for stop Národní Třída (line 22, direction BÍLÁ HORA)

Case 4, `apis.here_api.getAddress` responded without the field 'HouseNumber'. It returned only the street name 'Divadelní'

Case 5, the GPS position of tram stop Myslíkova (line 5, direction Lazarská) in the DPP database is misplaced.

This prototype again suffered from the technical problems and problems of the data.

In this prototype, the user P4 had troubles find how to go back P4. He was expecting a button instead texting "back". The other users didn't need to go back during the testing. Some of the nodes of the dialogue are too wordy "Your 'Narodni Trida'; I can't recognize whether you thought the stop or the street. Let's try it a different way... Did you meant the street 'Národní' or the stop 'Národní třída'?" P1 entered a combination, which is false, the system should detect before accepting a value, if the tram goes through the stop, and if it goes through the stop mentioned as the direction.

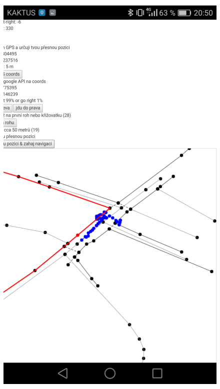
5.6.3.4 GPS

participant	navigation	address	technically
1	ok, but must return	ok	ok
2	wrong instructions	wrongly estimated	ok
3	ok	ok	ok
4a	wrong instructions	wrongly estimated	ok
4d	failed to launch	failed to estimate	bug getAddressFromProjection
5	ok, but must return	ok	ok

Case 1, the user had to return 70m after being localized

Case 2, the projection matches sidewalk on the other side of the street. See the screenshot from Václavské náměstí (Using the shape for estimation could fix it)

Case 4a, the address was estimated in the street before the corner. But the data


KAKTUS 63 % 20:50

t-right: -6
: 330

n GPS a určuji tvou přesnou pozici
304495
237516
: 5 m
S coords
google API na coords
775395
146239
t 99% or go right 1%
eva jdu do prava
ž na první roh nebo křižovatku (28)
i rohu
cca 50 metrů (19)
u přesnou pozici
u pozici & zahaj navigaci

Figure 5.3: Case 2: The projection matches to the sidewalk across the street

beforeCorner and afterCorner data were imprecisely annotated, but the fault of the interface. The user thought he is on the corner, marked it, but he was in the middle of the street, he realized latter. But he had no possibility to fix it.

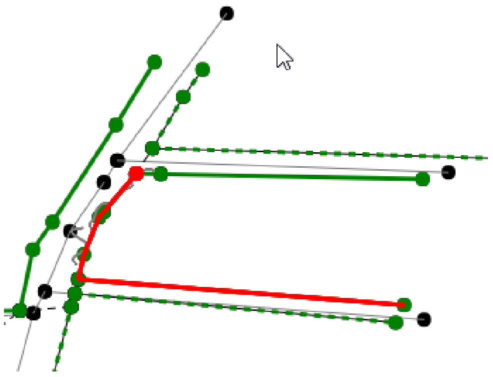


Figure 5.4: Case 4a: The estimated address is before the corner (grey, user path, red point estimated position) street

Case 4d, the GPS was correctly estimated. But the gpsLocalization.functions.getAddressFromProjection returned no address for given coordinates. There is some bug in that function.

Case 5, the user had to return 79m

The participants P2, 3 and 5 had troubles on the initialization, when they executed the order go to left, the system changed the order to go to the right. This way they started to infinitely pivot. As a solution, split the "stand with your backs to the wall and go left/right" into two separate steps. The users P1, 2, 3 and 5 were annoyed, when the system was announcing the number of collected coordinates. None of them had any idea, what those number means. Solution: turn it off. P4 reached the corner of the house, but it was not the corner of the street. He realized it later, but he had no possibility to mark the correct corner. P3 wondered about "Go to the next crossing or corner." There is a difference between those two. See the image ??

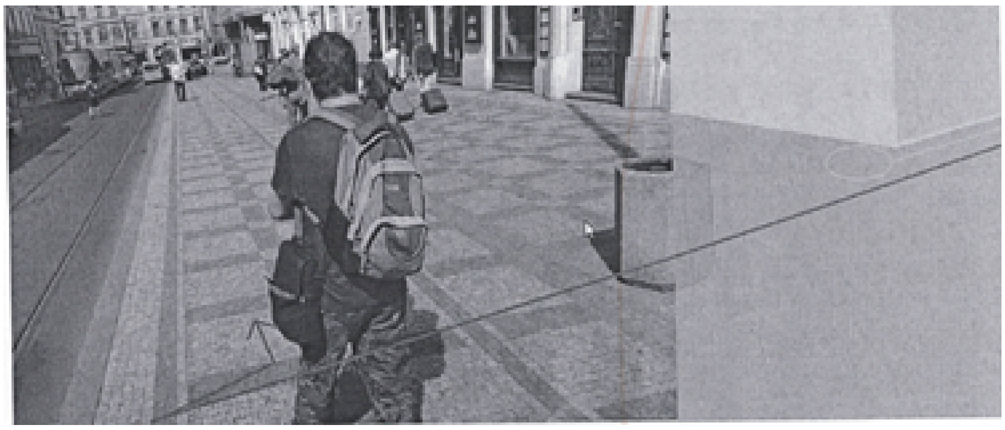


Figure 5.5: The user walked to the road instead of staying on the corner. The red arrow is the trajectory of the user. The white circle is the desired position.

5.6.4 Follow-ups

5.6.4.1 Reverse Geolocation

5.6.4.2 Google API

The Google reverse Geocoding API[46] returned the data in unexpected pattern for 3 locations (out of 6).

For most of the locations the API returns the data in the pattern HouseNumber, land registry number, street name: in the code it looks like this:

```
[
  {'long_name': '1', 'short_name': '1', 'types': ['street_number']},
  {'long_name': '1012', 'short_name': '1012', 'types': ['premise']},
  {'long_name': 'Národní', 'short_name': 'Národní', 'types': ['route']},
  ...
]
```

But for some locations it returns multiple house numbers. It shows e.g. Vodičkova 18-22 instead of just e.g. Vodičkova 18:

```
[
{'long_name': '18-22', 'short_name': '18-22', 'types': ['street_number']},
{'long_name': 'Vodičkova', 'short_name': 'Vodičkova', 'types': ['route']},
{'long_name': 'Nové Město', 'short_name': 'Nové Město', 'types': ['neighborhood', 'polit
...
]
```

Because of this unpredictable behaviour, I switched to HERE API[45]. And tested with participants 3, 4 and 5 using the HERE API.

The multiple house numbers example is from 50.0800307N, 14.4234307E (tram stop Vodičkova, tram 9 direction Václavské náměstí).

5.6.4.3 Here API

The HERE API[45] failed to return the HouseNumber for 2 locations (out of 10). It returned only the name of the street.

The influenced locations are:

50.07673N, 14.4141167E Tram stop Jiráskovo náměstí, tram 5, direction Anděl -> returns street 'Masarykovo nábřeží'

50.07673N, 14.4141167E Tram stop Jiráskovo náměstí, tram 5, direction Anděl -> returns street 'Masarykovo nábřeží'

50.081449N, 14.413722E Tram stop Národní divadlo, tram 22, direction Malostranská -> returns street 'Divadelní'

I didn't manage to find why it happens. Instead of using the estimation of an address. I propose the following:

1. implement navigation from a sidewalk. (current version of FindRouteAPI[29] can use only address)

or

2. replace reverse geocoding, with the following algorithm
   a. find the nearest sidewalk
   b. find the nearest address on that sidewalk

5.6.4.4 DPP data

The database[48] provided the wrong location of the tram stop Myslíkova (tram 5, direction Václavské náměstí). The position of the stop Myslíkova in the database is wrong. The coordinates of the stop are cca. 40m away from the real stop.

I noticed of some other miss-placed stops: Lazarská (tram 5, dir. Myslíkova), 27m away, Jiráskovo náměstí (tram 5, dir. Sídliště Barrandov), 36m, Národní divadlo (tram 22, dir. Malostranská), 23m.

I further noticed the stops are missplaced in Google Maps[26] : Myslíkova, Lazarská, Jiráskovo nám., Národní divadlo. And Lazarská is missplaced even in Mapy.cz[35].

I propose to collect the own database of the GPS coordinates of the tram stops and give it to Dopravní podnik[49]. This way the database[48] would contain actual data and our app could benefit if Dopravní Podnik, will make some changes to this database.

Chapter 6

Conclusion

The goal of this thesis was to design and implement a prototype of a mobile navigation application for the visually impaired. The prototype should achieve a sidewalk-precision level of localization. The application should be based on a dialogue using an existing dialogue system with an existing GIS. And the development should be focused on an efficient communication with usability in mind.

6.1 Summary

In this thesis, I demonstrated that we could expect the average accuracy of a GPS signal around 41m in a city. I further demonstrated that we could estimate the precise sidewalk where the user stands, if the user is instructed to walk about 30m, turn around the corner and walk ca. 30 meters more. I researched, nowadays, that we can use only Naviterier[6] GIS for navigation of visually impaired people. I analyzed what data, we can use from this GIS. I enhanced the output of Naviterier's API[17] by adding types of corners of houses, types of the traffic lights and the noise of the streets. I researched how the blind describe their surroundings in the literature, how to navigate them, and with three visually impaired I've studied what can they feel and recognize in the city.

Based on that, I did five iterations of prototyping. In the first three iterations, I tested how we can localize the user based on the natural description of his surroundings and the output of the enhanced Naviterier's API. The prototype was based on a dialogue implemented using the method Wizzard-of-Oz with increased fidelity in each iteration. I tested the prototypes with five to three sighted people (qualitative testing). Still, I didn't manage to discover any promising strategy. Therefore in the next two iterations, I introduced five new prototypes based on the three new methods (names of two crossing streets, entering a point of interest, and collecting the user's path). I implemented the five prototypes, first as low fidelity prototypes. I prepared dialogue diagrams tested by the method Wizzard-of-Oz and tested them with five visually impaired people (qualitative testing). Then I implemented the two most promising prototypes as the High-Fidelity prototypes - an accessible web application. I implemented the state-of-the-art method - reverse geocoding as well as the Hi-Fi prototype. I tested this three Hi-Fi prototypes with another five visually impaired people (qualitative testing).

I discovered in the last testing that we couldn't rely our methods on some of the external data and APIs: Some positions of the stops in the *Open Data Time Tables of Public Transport*[48] (DPP database) are misplaced. The reverse geolocation APIs as Google Geocoding[46] or HERE API[45] for some locations return the data in an unexpected format e.g. the house number is missing.

Furthermore, I discovered that some combination of the algorithms and the external data needs further inspection: The algorithm `getGpsFromTriple` localizing the tram platform doesn't work in some cases. For example, it might be wrong, or the data have an unexpected format or the routes of the tram lines are more irregular than I expected them to be. The algorithm `getAddressFromProjection`, which finds the nearest address on the given sidewalk, doesn't work in some locations.

6.2 Goals Accomplishment

I managed to implement two prototypes and the state-of-the-art Hi-Fi prototypes as a working web application. The prototypes promise to have the sidewalk-precision level. However, the technical problems mentioned above caused that I failed to validate if the prototypes indeed achieve the desired precision. One of the implemented hi-fi prototype features the dialogue interface. For the other two prototypes, I decided the benefits of the implementation of such interface wouldn't pay off at this level of fidelity. I used an existing dialogue system Watson Conversation[22] and existing GIS[6]. I conducted five iterations of user testing to ensure the perfect usability. And indeed, in the last testing, the prototypes suffered mostly from the technical problems than from the problems of the interface.

Bibliography

[1] Balata, J.; Mikovec, Z.; Novacek, J. Field Study: How Blind People Communicate While Recovering from Loss of Orientation. In *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference On*, IEEE, pp. 313–318.

[2] LLC, G. Maps, Google Android. 5. ledna 2018. Available from: <https://play.google.com/store/apps/details?id=com.google.android.apps.maps>

[3] iOS - Maps, Apple. Available from: <https://www.apple.com/ios/maps/>

[4] BlindSquare. Available from: <http://www.blindsquare.com>

[5] Smithson, L. NotNav GPS Accessibility. 1. listopadu 2017. Available from: <https://play.google.com/store/apps/details?id=com.smithson.notnav&hl=cs>

[6] Naviterier - Navigační Aplikace a Služby. Available from: <https://naviterier.cz/>

[7] Amazon Alexa. Available from: <https://developer.amazon.com/alexa>

[8] Apple Siri. Available from: <https://www.apple.com/ios/siri/>

[9] Cortana. Available from: <https://www.microsoft.com/en-us/windows/cortana>

[10] van Diggelen, F.; Enge, P. The World's First GPS MOOC and Worldwide Laboratory Using Smartphones,. In *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, pp. 361 – 369. Available from: <https://www.ion.org/publications/abstract.cfm?articleID=13079>

[11] Global Positioning System. Page Version ID: 813276537. Available from: <https://en.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=813276537>

[12] GLONASS. Page Version ID: 813158685. Available from: <https://en.wikipedia.org/w/index.php?title=GLONASS&oldid=813158685>

[13] Satellite Navigation. Page Version ID: 813288262. Available from: <https://en.wikipedia.org/w/index.php?title=Satellite_navigation&oldid=813288262>

[14] GPS.Gov: GPS Accuracy. Available from: <https://www.gps.gov/systems/gps/performance/accuracy/>

[15] GPS Logger for Android. Available from: <https://play.google.com/store/apps/details?id=com.mendhak.gpslogger>

[16] GPS Logger for BlackBerry 10. Available from: <https://play.google.com/store/apps/details?id=com.mendhak.gpslogger>

[17] FindSourceData API. Available from: <http://147.32.81.71/NaviTerier.ProcessingService/json/metadata?op=FindSourceData>

[18] Loomis, J. M.; Golledge, R. G.; Klatzky, R. L. Navigation System for the Blind: Auditory Display Modes and Guidance. volume 7, no. 2.

[19] Balata, J.; Mikovec, Z.; Maly, I. Navigation Problems in Blind-to-Blind Pedestrians Tele-Assistance Navigation. In *Human-Computer Interaction*, Springer, pp. 89–109.

[20] Bui, T. H.; Poel, M.; Nijholt, A.; et al. A Pomdp Approach to Affective Dialogue Modeling. volume 18: p. 349.

[21] Young, S.; Găsić, M.; Thomson, B.; et al. Pomdp-Based Statistical Spoken Dialog Systems: A Review. volume 101, no. 5: pp. 1160–1179.

[22] IBM Watson Conversation. Available from: <https://www.ibm.com/watson/services/conversation/>

[23] Inc, W. WhatsApp Messenger. 4. prosince 2017. Available from: <https://play.google.com/store/apps/details?id=com.whatsapp>

[24] Facebook. Messenger. 6. prosince 2017. Available from: <https://play.google.com/store/apps/details?id=com.facebook.orca>

[25] Firtman, M. *Programming the Mobile Web*. "O'Reilly Media, Inc.", ISBN 978-1-4493-9589-6.

[26] Mapy Google. Available from: <https://www.google.cz/maps/>

[27] OpenStreetMap. Available from: <https://www.openstreetmap.org/>

[28] Naviterier - Poslepu.Cz. Available from: <http://poslepu.cz/tag/naviterier/>

[29] FindRoutes API. Available from: <http://147.32.81.71/NaviTerier.ProcessingService/json/metadata?op=FindRoutes>

[30] GetAddresses API. Available from: <http://147.32.81.71/NaviTerier.ProcessingService/json/metadata?op=GetAddresses>

[31] GetPois API. Available from: <http://147.32.81.71/NaviTerier.ProcessingService/json/metadata?op=GetPois>

[32] Vystrcil, J.; Maly, I.; Balata, J.; et al. Navigation Dialog of Blind People: Recovery from Getting Lost: p. 58.

[33] Ungar, S. 13 Cognitive Mapping Without. volume 4: p. 221.

[34] May, A. J.; Ross, T.; Bayer, S. H.; et al. Pedestrian Navigation Aids: Information Requirements and Design Implications. volume 7, no. 6: pp. 331–338.

[35] Mapy.Cz. Available from: <https://mapy.cz/>

[36] Naviterier - Developer. Available from: <https://naviterier.cz/dev/index.php>

[37] Axure RP. Available from: <https://www.axure.com/>

[38] Acapela Box : Create Your Text to Speech Messages. Available from: <https://acapela-box.com/AcaBox/index.php>

[39] The Web Framework for Perfectionists with Deadlines | Django. Available from: <https://www.djangoproject.com/>

[40] Spring Boot. Available from: <https://projects.spring.io/spring-boot/>

[41] Thymeleaf. Available from: <http://www.thymeleaf.org/>

[42] jquery.org, j. F. jQuery. Available from: <https://jquery.com/>

[43] contributors, J. T.; Bootstrap, M. O. Bootstrap. Available from: <https://getbootstrap.com/>

[44] Web Speech API Specification. Available from: <https://w3c.github.io/speech-api/webspeechapi.html>

[45] Reverse Geocode Resource - Geocoder API - HERE API. Available from: <https://developer.here.com/documentation/geocoder/topics/resource-reverse-geocode.html>

[46] Getting Started | Google Maps Geocoding API. Available from: <https://developers.google.com/maps/documentation/geocoding/start>

[47] W3C Geolocation API. Page Version ID: 787950918. Available from: <https://en.wikipedia.org/w/index.php?title=W3C_Geolocation_API&oldid=787950918>

[48] Jízdní Řády - Opendata Praha. Available from: <http://opendata.praha.eu/dataset/dpp-jizdni-rady>

[49] Dopravní Podnik Hlavního Města Prahy. Available from: <http://www.dpp.cz/>

[50] Linky Tramvají - TRAM-BUS.Cz. Available from: <https://www.tram-bus.cz/mhd-praha/tramvaje/linky-tramvaji/>

[51] Neviditelná Výstava. Available from: <http://neviditelna.cz>

[52] LLC, G. Prohlížeč Chrome – Google. 15. prosince 2017. Available from: <https://play.google.com/store/apps/details?id=com.android.chrome>

[53] LLC, G. Google TalkBack. 21. června 2017. Available from: <https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback>

Appendix A

Contents of the SD card

```
│ readme.md ........................ the instructions how to launch the prototypes
├── data ........................................ the working data files directory
│   ├── dpp_data ............... info about the public transport; see Section 4.3.1
│   └── db.sqlite3 ....... Django's database prefiled with the Addresses and the DPP info
├── src ........................................ the directory of source codes
│   ├── chat_demo Deobfuscator ....... the source code of the Hi-Fi prototypes in Django
│   ├── pyhon_virtenv ................ the setuped virtual eviroment in Python 3.6
│   └── latex ................................... the source code of this thesis
└── thesis.pdf ................................... this thesis in PDF format
```