Technische Universität Berlin

Development of an interactive digital campus map with navigation functionality for mobile devices

by

Patrick Christoph Zdanowski

Matriculation Number 0410378

A thesis submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

Bachelor's Thesis

January 10, 2024

Supervised by:
Prof. Dr. Axel Küpper

Assistant supervisor:
Dr. Dr. Chuck Norris

SNET
SERVICE-CENTRIC NETWORKING

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Berlin, January 10, 2024 Patrick Christoph Zdanowski

Abstract

In this thesis, we show that lorem ipsum dolor sit amet.

Zusammenfassung

Hier kommt das deutsche Abstract hin. Wie das geht, kann man wie immer auf Wikipedia nachlesen `http://de.wikipedia.org/wiki/Abstract`...

Contents

1 Introduction

1.1 Context and background

With over 60 different buildings, more than 35000 students and an area of over 600,000 square meters, TU Berlin's campus in Charlottenburg is one of the biggest continuous campuses in Europe. While studying is the main reason for most people to be on campus, learning and visiting courses are not the only tasks that take place on it. Campus Charlottenburg gives people the opportunity to learn and research, but also to connect to other like-minded people, eat meals in one of the cafeterias or bakeries, as well as to participate in one of the many social events that take place on it. Campus Charlottenburg is also the working place of over 7000 persons, including professors, students and researchers.

With its enormous amount of important facilities for students, families and other members of the university, campus Charlottenburg plays a central role in the lives of many people.

1.2 Motivation

TU Berlin's enormous size and complexity come with several difficulties for people working and studying on campus Charlottenburg, including an overwhelming amount of information about and around the campus as well as problems with navigation across it. New members of TU Berlin have problems localizing certain buildings and are overwhelmed by the amount of information the campus provides.

A working, manageable and modern digital information culture is one of the most significant key factors for overcoming those difficulties and for successful study, work and life on campus Charlottenburg. Such a system should offer an easy and intuitive way of accessing and managing information about the university to all of TU Berlin's members.

Some online resources and systems attempt to enhance the digital information culture including websites provided by TU Berlin and Studierendenwerk Berlin as well as a campus map and digital navigation systems for mobile devices. Nevertheless, there are several problems with the current landscape of web resources and platforms provided by TU Berlin and Studierendenwerk Berlin:

One of them is the fact that there is currently no fitting digital solution for navigation on the campus. Students currently have two options when choosing a tool to navigate across it: On the one hand, TU Berlin provides a downloadable image of campus Charlottenburg, which gives a basic aerial overview of the whole campus. Although this solution is straightforward and understandable, it only provides the possibility to navigate manually over the campus. It also does not take full advantage of the digital tools mobile devices and personal computers offer us.

On the other hand, there are several external digital navigation system providers such as Google or Apple Maps. Those systems take advantage of the digital tooling we have and provide a modern and mobile way of navigation. The main problem with these systems is the fact that they do not contain a detailed mapping of campus Charlottenburg. Searching for several buildings is difficult/impossible and since these systems are not profound for most of the paths on campus, proposed routes are often based on public routes around the campus rather than the faster ones inside of it.

Another problem is the fact that the amount and size of provided platforms is, in itself, overwhelming and complicated. The information about campus Charlottenburg is spread across the internet, instead of bundled. There is no central entity, that contains all information. The fact that most of the information is provided in the form of websites has another consequence for members of TU Berlin: Looking up information fast and intuitive on mobile devices, one of the most used digital tools in the present time, is often slow, unintuitive and unnecessarily complicated.

1.3 Proposed solution

This bachelor's thesis tries to investigate the current digital information culture at TU Berlin and wants to solve information retrieval and navigation difficulties by providing an implementation of a digital information and navigation system for mobile devices.

Both of these features use an underlying offline map of campus Charlottenburg, which provides a base layer for the main screen of the app. All other features and interface elements are therefore stacked in separate layers on top of it.

1.3.1 Campus map

A custom map of campus Charlottenburg represents the main element of the app. It comes completely bundled with the application for offline usage and displays the most relevant information about the structure of campus Charlottenburg, e.g., buildings, cafeterias, green areas and pathways across the campus.

1.3.2 Navigation system

A fast, reliable and offline-usable navigation system overlays the campus map and helps users find their way across the campus. It contains all the geodata and points of interest (POI) of the campus and can reliably calculate the fastest route between them. It also integrates the GPS functionality of the mobile device, to consider the current user location while navigating.

1.3.3 Information layer

The second layer on the map displays information about the campus. It fetches and parses them automatically from publicly available web resources and maps the information onto their respective POI.

2 Related Work

2.1 GPS

The global positioning system (GPS) as described in [1] and [2] is the modern standard for determining the position of a user on Earth. It is widely used across several different domains including navigation, tracking functionalities, military usage and other location-based applications. At least since the rise of mobile devices equipped with GPS capabilities the majority of people regularly use GPS within several mobile apps and services.

The basis of GPS technology consists of a synchronized satellite network, which constantly broadcasts status information about the respective position, orbit and time of its members. GPS devices calculate their latitude, longitude and altitude by receiving data from at least four satellites. After data is transmitted, the signal propagation time is used for lateriation, from which the position of the device can be determined. [3] presents the mathematical background for such position determination: Based on the signal runtime, the distance R_i between each satellite "i" and the receiver is calculated. With the unknown receiver position (x, y, z) and the known satellite position (x_i, y_i, z_i) the corresponding distance can be expressed as follows: $(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = R_i$. This set of (at least 3) equations can be then solved for the receiver position (x, y, z).

Since a correct measurement of signal runtime is essential (errors of 1 ms result in uncertainties of around 300 km [2]) and consumer-friendly GPS devices usually only contain a simple quartz-based clock, error estimation and correction systems are needed to estimate the exact time.

Depending on the environment and scenario in which GPS is used, several optimizations and improvements can be made. Since this thesis implements a navigation system for mobile devices, smartphone-focused implementation and usage techniques are particularly interesting.

One important example of such an improvement is the Google fused location provider API. This API is specifically designed for usage on mobile devices and improves the efficiency and accuracy of location retrieval by combining several internal smartphone sensors such as GPS and WiFi [4].

2.2 OSM mapping service

OpenStreetMap (OSM) is a community-driven open-source mapping service that provides annotated digital maps for most of the countries in the world. It was initiated in 2004 and consists of a web application [5] that hosts a static map as well as the Overpass Turbo API [6] for data retrieval.

One problem arising from the fact that OSM is community-based is the fact that it lacks quality assurance [7]. OSM quality varies depending on the country and region [7]: While rural areas often lack information, urban locations are often precisely represented and contain information comparable to data provided by the respective government of the selected area or commercial companies.

[7], with its implementation of the web app 'Is OSM up-to-date?´ provides a systematic approach for measuring the OSM quality of an area. It presents how recently certain areas have been edited by the community and infers from that the need for revision on the map. In the case of TU Berlin's campus in Charlottenburg, the available data is well-maintained [8] and can be used as a starting point for this thesis.

2.3 Techniques used in navigation systems for mobile devices

Navigation systems for mobile devices are nowadays a crucial part of the landscape of available navigation solutions. Popular mobile apps, such as Google [9] or Apple Maps [10], often provide users the ability to navigate, locate and explore the world around them. The core features of those systems and important aspects for this thesis are:

2.3.1 Routing

Routing through a network of streets and places is an important key functionality of every digital navigation system. The most popular techniques for routing all use an underlying graph representation of the street network [11]. With this technique, streets are represented as weighted nodes, which model the underlying cost of moving between different locations.

One of the most popular algorithms to determine the fastest way between two nodes is Dijkstra's Algorithm. It greedily computes the shortest paths in the whole network and always returns the optimal solution (the fastest path). Despite its popularity and effectiveness, the algorithm's runtime scales poorly for huge graphs, e.g., the underlying data of Google Maps [11]. This is the reason why most of the algorithms used for routing in a navigation environment use different heuristics to speed up calculation. These algorithms usually trade off optimal route calculation for increased execution speed [12]. The following list presents an overview of the basic Dijkstra implementation and other commonly used routing algorithms [12].

Dijkstra: This algorithm starts at the start node and calculates the movement cost to all directly reachable nodes. It enqueues all visited nodes and repeats this process for the first node in the queue. Processed nodes get dequeued and costs for movement are updated when a shorter path is found. By calculating all possible paths in the graph, the algorithm is guaranteed to find the optimal solution. The main disadvantage of this approach is its runtime for huge graphs. Use cases are small graphs, several internet routing protocols and educational purposes.

A*: A* is a generalized version of the Dijkstra algorithm. The difference can be found in the cost function for movement between two nodes: Instead of only using the edge weight, A* applies a heuristic to the visited nodes. This heuristic usually consists of the airline between the

current node and the destination and improves calculation speed by providing the algorithm an indication of nodes that can be prioritized in the search. A* also always finds the optimal solution, while being faster than Dijkstra in most cases. Use cases are routing in bigger networks and educational purposes.

Genetic algorithms: Genetic algorithms describe a class of techniques for routing based on evolutionary processes. These algorithms start by choosing a random path and treating it like a gene. Principles like genetic crossover and random mutations are applied to it and the paths with the least movement cost are used for creating new populations of genes. This evolutionary process often converges to an acceptable solution in an adequate time. Its biggest disadvantage is the fact that it does not necessarily provide an optimal solution. Use cases are large graphs and scenarios that do not need an optimal solution.

2.3.2 Location detection

The main system used for outdoor localization in Google Maps and other similar services is GPS [11]. During usage of the navigation app, the user gets the opportunity to activate it and give permission for its usage. While Google's fused location API is the default location provider for Android devices [4], IOS devices use Apple's Core Location framework [13].

Although GPS is the standard system for localization, several other techniques are often used to improve pedestrian location detection on mobile devices. One relevant for this thesis is pedestrian dead reckoning.

Pedestrian dead reckoning localizes the user by tracking its steps and the movement direction. In a mobile scenario, this can be accomplished with the built-in device sensors. Steps can be detected with a distinct pattern in the retrieved acceleration, while the heading angle is read from the magnetometer [14]. Additionally, the system needs a reference point for localization: Since only the distance of movement in a certain direction is tracked, the starting point has to be known. Another crucial factor for correct PDR is a precise estimation of step length [14]. The following formula presents a standard way to calculate the current location in a 2-dimensional environment with step-count i, position (x, y), estimated step length L and heading ϕ [15].

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} + L * \begin{pmatrix} sin(\phi_{i-1}) \\ cos(\phi_{i-1}) \end{pmatrix} \tag{2.1}$$

One of the biggest disadvantages of PDR is the fact that errors accumulate during usage. Imprecisions in magnetometer, acceleration and step length values can result in incorrect PDR localization during long distance usage.

[16] describes an approach for merging data of the smartphone's built-in sensors for PDR with GPS functionality using a Kalman filter. It tries to use pedestrian dead reckoning as a base for localization and corrects its error accumulation with additional position lookup through GPS. The work states that such a technique can improve localization accuracy as well as energy-saving in a pedestrian position detection scenario. This improvement is accomplished by using classical pedestrian dead reckoning techniques (PDR) and GPS capabilities for correction.

2.3.3 Estimated time of arrival (ETA)

Providing an estimation of the time it takes to overcome the distance from start to destination is an important task for modern navigation systems. Google Maps calculates the estimated time of arrival (ETA) based on several different factors [11], including:

- Distance from start point to destination
- Average speed of the user and its selected form of transportation
- Current traffic situation on the chosen route
- Official and recommended speed limits
- Road types
- Historical average speed data
- Collected data from users who traveled similar routes

By collecting and merging this information, Google Maps manages it to provide its users with a precise ETA [11].

2.3.4 Visual design

The next step after retrieving the user's location and calculating the best route as well as its ETA is the presentation of that information within the app. [17] provides an overview of different solutions for information presentation within a mobile navigation system. It compares auditory instructions with route presentation of a map, simple route presentation on a map, display of current location and direction as well as a list of textual descriptions.

After evaluating all four different methods, [17] suggests the following aspects for a successful presentation of navigation data:

- A map oriented in the current walking direction should show the user's current location as well as the taken route and information about the map (landmarks, street names)
- The map should provide a zoom function
- Textual instructions are the most inefficient way of communicating navigation information
- Audio instruction can improve the map functionality, although GPS accuracy has to be taken into account when providing information about distance

2.4 Location-based services

Location-based services are mobile applications/services, which utilize the ability to localize the user to provide customized information [18]. By offering users information about nearby restaurants, supermarkets, shops, hotels and other POI, mobile navigation systems such as Google [9] or Apple Maps [10] also fall into the category of such services. Other popular apps using location-based systems are emergency services, tour guides, delivery tracking systems, social-media platforms with location-sharing functionalities, weather apps and mobility or taxi apps [19] [20].

Location-based mobile apps nowadays usually use the in-build GPS capabilities of mobile devices to provide their services in outdoor environments. Other techniques used for indoor environments are WiFI, RFID, ZigBee and Bluetooth [20].

Since the core functionality of the software developed in this thesis is comparable to mobile navigation systems such as Google or Apple Maps, the goal of this thesis itself can be described as a location-based service.

2.4.1 Geofencing

Geofencing describes a class of techniques used to automatically send notifications based on entering or leaving a specified geographic location [21]. These geographic areas are usually specified geometrically, by placing circles, polygons or polylines into specific locations, or symbolically by specifying a location by name such as a state, a country or an address [21].

There are several use cases for geofencing including marketing systems that send promotional notifications when users enter defined areas, logistic systems, that track and detect when certain deliveries and vehicles enter destination locations, mobile apps for navigation and social networking, as well as safety applications for tracking and detecting dangerous areas. One of the most prominent examples of geofence usage in a mobile application for marketing purposes is Burger King's "Whopper detour" campaign launched in 2019 [22]. The company sold a Whopper for one cent to every user who placed an order via the Burger King app within a McDonald's restaurant.

One key enabler for geofencing is an extensive background tracking functionality [19]. In the case of modern mobile devices, smartphone apps with a geofencing functionality need the ability to constantly query the user's location, even if the app is in the background.

3 Concept and Design

The goal of this bachelor's thesis is the development of an app for mobile devices, which provides students at TU Berlin the possibility to navigate and inform themselves about their campus in Charlottenburg. The main concept of the mobile app is inspired by popular smartphone navigation systems such as Google or Apple Maps and focuses on mapping TU Berlin's main campus and the most important web resources connected to it onto a single digital map.

3.1 Key features and technologies

The following sections provide a detailed overview of the app's key features and the underlying technologies used in this thesis.

3.1.1 Digital map of campus Charlottenburg

The main element of the mobile app consists of a locally implemented map of TU Berlin's central campus in Charlottenburg. It provides the user a manageable overview of TU Berlin's buildings, pathways, green areas as well as its surrounding environment. The following list displays possible map features with a description of their relevance for the mobile app:

All buildings of TU Berlin: To provide the user the ability to easily locate the buildings of TU Berlin, all facilities connected to the university need to be specially highlighted on the map. The buildings of TU Berlin are therefore the most important map feature.

All pathways on campus: Footwalks and cycleways that lay on the campus are important for the navigation system of the mobile app. To prevent the map from being cluttered, a hierarchy must be established between important pathways and smaller routes.

All green areas on campus: Parks, trees and other green areas of TU Berlin need to be specially marked on the map. Combined with the buildings and pathways of TU Berlin, they provide the user reference points for manual localization.

External buildings: Buildings not connected to the TU Berlin do not contribute to the localization and navigation on campus. They can be nevertheless used as weakly informative reference points. A toned-down and subliminal representation on the map can be used in this case.

External pathways: All footwalks and cycleways that are outside of the campus do not provide any relevant information for the mobile app. They furthermore make the map appear more cluttered and are therefore not present on it.

External green areas: Green areas provide a source of orientation and are an important part

of the map.

Main roads: Main roads surrounding TU Berlin's campus (e.g., Straße des 17. Juli) simplify the exploration and search process while interacting with the map. They provide an important source of guidance and must be prominently presented on the map.

Small roads: Small roads also support an organized map concept. Since they are less important than main roads, a more restrained manner of display is appropriate.

External POI: The POI surrounding the campus (such as Ernst-Reuter-Platz) are heavily recognizable landmarks and support the user's orientation and localization on the map. They are therefore completely displayed on it.

All relevant features are retrieved via the Overpass-Turbo API from publicly available OpenStreetMap data. The data is then fed into the geographic information system QGIS, which is used for the creation and export of the campus map as well as manual annotation and correction of the downloaded data. Finally, the exported data is converted into a 3d model of the campus which is displayed in a respective rendering environment inside the app.

3.1.2 Navigation across the campus

The mobile app provides the user the ability to easily navigate across TU Berlin's main campus. The most important technological aspects to successfully achieve this task are routing, localization, geocoding, visual presentation of the current navigation state and calculation of estimated time of arrival (ETA).

The underlying data structure used for the whole navigation process is a weighted graph. Its vertices represent collected geodata points and the respective weighted edges are the distances between the vertices. To account for the fact that, in some cases, the fastest route between two points leads through other buildings, the entrances of all facilities are included in the graph. An edge connecting every pair of different entrance nodes of the same building is further included. The following section provides an overview of the procedure for the complete navigation process:

1. Geocoding is used to decode the user's human-readable destination input, e.g., "MAR-Gebäude" into its respective geo-coordinates and node in the graph.
2. To determine the starting point for the navigation, either another manually inserted location is geocoded or the current location of the user is determined using the built-in GPS module of the mobile phone. If the user selects the latter, the current location and heading are further tracked during the whole navigation process.
3. Considering the limited size of the graph, Dijkstra's algorithm is chosen to calculate the fastest route between the start and destination points. If the fastest route leads through other buildings, a live indoor navigation approach cannot be achieved due to imprecise GPS results. In this case, geofences are placed on the entrance and exit of the respective building. They detect when the user enters and leaves it and update the state of the navigation system accordingly.
4. The ETA is calculated based on the weights/distance of the fastest route and other factors (e.g., time of day, stoplights on the route, ...).

5. The navigation is presented visually on the campus map by drawing a polyline of the calculated route. If the user selects its location as a starting point, the current location as well as the heading retrieved from the device's magnetometer is displayed. When the user is located within a certain range of the polyline, its position and heading are mapped onto it.

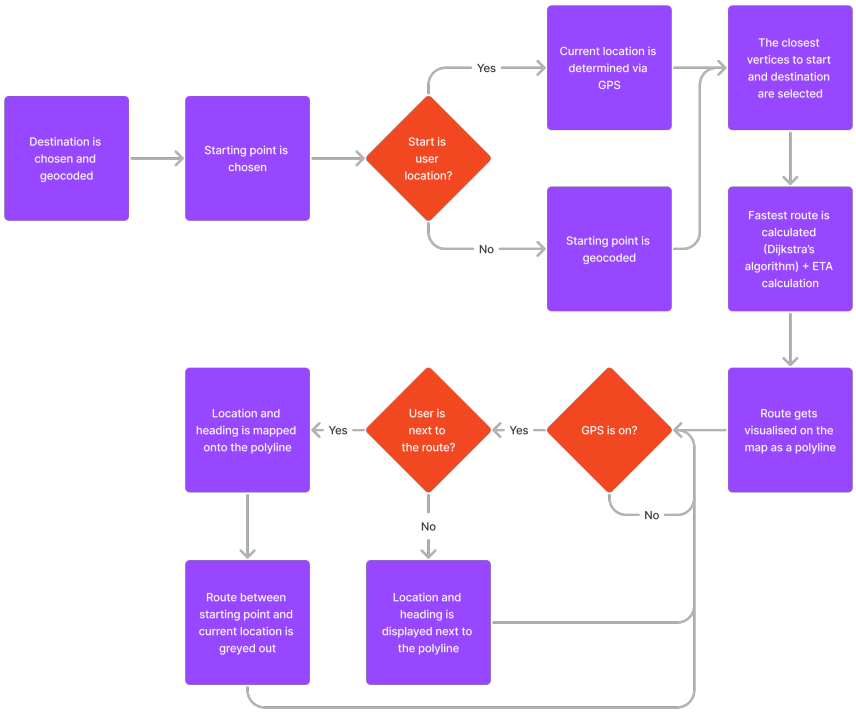

Figure 3.1: Complete navigation system procedure

3.1.3 Information layer

An additional layer on top of the campus map consisting of location-based information is a natural addition to the feature set of the mobile app. It enhances the user experience, the relevance of the whole product and the overall digital information culture of TU Berlin by providing overlays, markers and labels filled with data about the campus.

This section provides a brief overview of the underlying data and technology used to provide the information on the layer.

There are three main sources, that are publicly available and relevant to the information layer. On the one hand, there is the official website of TU Berlin [23], with sections for current events, the latest news, deadlines, etc. On the other hand, there is the MOSES system [24] containing data about all the different courses, rooms and studies. A third relevant source of information

is the website of Studierendenwerk Berlin [25] with meal plans for its different canteens and a timetable for events.

Information is collected by downloading and parsing the content of respective websites. The data is further categorized and -if possible- mapped onto different POIs on the campus map (e.g., a canteen gets its meal plan assigned). This helps to provide an intuitive and organized presentation and access to the user.

The collected data can be categorized according to its timeliness and the intervals in which it has to be updated: General information about different fields of study, courses and rooms only changes by semester. This particular data can be retrieved once at the start of every semester and does not need daily live updates. It is also possible to supply it via app updates, instead of providing a direct in-app functionality for data retrieval. Data that gets updated regularly on the other hand, e.g., meal plans, events, news, etc. has to be always retrievable from the app.

The proposed solution for data collection and provision runs on a web server and consists of a web crawler for information retrieval, parsing and POI mapping, a CSV generator to convert the crawled information into a standardized and simple-to-use format and a REST API, that provides the client/mobile device the ability to retrieve the CSV files. The web crawler and CSV generator are both triggered periodically by several CRON jobs, whose timings are dependent on the timeliness of the crawled data. Due to its simplicity, the circumstance that there are no complex relations in the data and the fact that all room data from TU Berlin is provided in it, the CSV format is chosen as an alternative to a fully-fledged database system for data storage.

By splitting the logic for crawling and provision, the workload that arises on TU Berlin's and Studierendenwerk's websites from retrieving data can be limited to a minimum: The server only crawls data when an update is needed (e.g., weekly for meal plans), instead of loading and parsing the web resources on client request. Further advantages are the fact that loading times for requests from clients are independent of TU Berlin's and Studierendenwerk's infrastructure, that the language for web crawling can be chosen independently from the programming framework (in this case Python with its Selenium [26] and BeautifulSoup [27] libraries are used) and that the web crawling logic can be changed without updating the app.

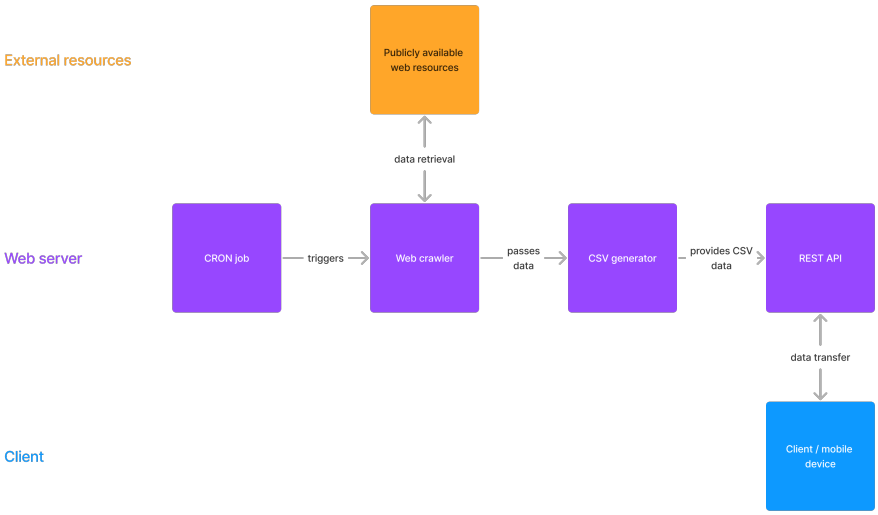

Figure 3.2: Data retrieval process for the information layer

3.1.4 Offline-first design

One important aspect of the app concept is an extensive focus on offline usability. The whole system is designed to allow the user (as far as possible) a network-connection-free app usage. This applies in particular to the navigation system, the campus map and the information layer of the app:

Campus map: The campus map is locally implemented on the device. All necessary information can be therefore loaded and used without a network connection.

Navigation system: Due to the comparatively small scope of the navigation system, the weighted graph for navigation as well as the system for routing can both be implemented and executed on-device. Only the navigation feature working with live user location updates relies on stable GPS functionality.

Information layer: Loading and presenting information on TU Berlin's main campus requires access to the respected online resources and cannot be realized without a network connection. The need for network connectivity can be nevertheless reduced: Since no data requires real-time updates (the most critical data being meal plans that require weekly updates), the data for the information layer can be stored locally after download and loaded from memory for the remaining week. This approach reduces the need for internet connectivity to a minimum.

3.2 App development framework

The choice of app development framework is an important step while conceptualizing and designing the product. It determines the programming language of the project, the built-in

device- and system-specific capabilities that can be accessed during development as well as the performance of the app. It has to be selected in consideration of the project's requirements. The following enumeration presents the most important demands for this thesis:

- Used technologies: Working with low-level hardware and operating system APIs is an important part of development. Technologies and systems used in the key features of the product (e.g., GPS capabilities) have to be accessible or implemented in the selected framework.
- Cross-platform capabilities: Native app development is complex and time-consuming. Learning and especially working with two different codebases (Java / Kotlin for Android, Swift / Obj. C for IOS) is difficult and not possible within the scope of this thesis. Cross-platform frameworks solve this problem by providing the ability to work with a single codebase for different operating systems.
- Focus on high-quality user interfaces: One of the most important requirements of the app development framework is the availability to easily build high-quality and modern user interfaces with it.
- Execution speed: Regarding the fact that the whole navigation functionality has to take place locally on the phone, the speed with which the code of the mobile app gets executed is crucial to a responsive and fast user experience. Particularly the algorithms used for routing have to be reliably computable within seconds.
- 3d rendering capabilities: The framework needs to provide a way to render and interact with the created 3d model of campus Charlottenburg. Further built-in systems such as an extensive linear algebra toolkit, a lighting engine and other real-time rendering capabilities also need to be taken into account.

Based on the previously defined requirements, the Flutter framework [28] is chosen for this work. In addition to its cross-platform capabilities, it also offers the ability to compile source code into platform-specific machine code for near-native execution performance. It further comes with access to thousands of packages through the dart package manager pub [29], an extensive set of pre-built components for user interfaces and the ability to write platform-specific code for low-level API access. Since Flutter lacks 3d rendering capabilities, the Unity engine [30] is additionally used to create and display the campus map. It gets included in the Flutter build and is responsible for rendering and interacting with the 3d campus map.

3.3 User experience design

The following sections provide an overview of the whole user experience (UX) design for the campus app. In the scope of this thesis, the complete process is broken down into UX research and wireframe design.

UX research is a process where the main goals and tasks associated with the app are identified and broken down into its most relevant UX aspects. This helps to find out the key usability requirements for the visual user interface design, consisting of wireframes and high-fidelity mockups (user interface design).

Wireframes, on the other hand, form the first visual version of the app design. They determine the basic structure of the app, its layout, the placement of basic user interface elements as

well as their hierarchy. Wireframes are based on the previously defined usability requirements in the UX research process.

The next paragraphs describe a set of important usability requirements that are then further incorporated into the wireframe design.

3.3.1 Search versus exploration

One of the most important usability requirements for the app results from the clash of two important user needs, namely search and exploration. These needs result from the fact that users may use the app for different purposes and with different expectations.

Search, on the one hand, describes a scenario, in which the user tries to find or locate a specific piece of information within the app. This search for information can be expressed with concrete questions, e.g.:

- What is the fastest way between MAR and TEL buildings?
- What are the opening times of the TU library?
- What food can I get tomorrow at the main cafeteria?
- Where does course App-Entwicklung take place?
- In which building is the SNET department located?

One main requirement for the user experience of the app is therefore the possibility for fast, easy and structured access to important key information about TU Berlin's campus. All search-task-related user interface elements should be easily identifiable as such, prominently positioned and accessible to the user. Additionally, to reduce the search time for the seeking person, the design should also provide multiple ways to access the most important information. Important user interface elements in this case are search bars, descriptive icons and textual hints as well as an adequately chunked information hierarchy in the whole app.

Exploration, on the other hand, is an app use case, in which the user "just browses around" in search of nothing particular. The seeking person either wants to learn, find out or experience something new or uses the app with an open question in mind. In this scenario, the user does not seek a specific piece of information but rather expects the app to provide the possibility, to easily navigate and view through structured content without the need for a concrete search.

Examples of goals and questions for exploration use cases can be the following:

- Are there any interesting places on TU Berlin's campus that I could visit?
- Which cafeteria has the best food today?
- Are there any appealing courses, outside of the scope of my studies, that I can attend this semester?
- Search for upcoming events by TU Berlin or Studierendenwerk
- Search for learning spaces at TU Berlin

In these use cases, the outcome of the exploration action is often determined by the subjective

preferences of the users. The app therefore cannot give an optimal final solution to the search but rather present a set of information, from which the user can select or find the most suitable one.

This results in two important requirements for the app design. Firstly, visual elements for the structured presentation of information need to be used. Examples of that can be tables, scrollable lists, clearly designed information cards and different kinds of markers on the campus map. These elements help the user by giving hints about existing information in the app and by structuring related information, which then can be easily overviewed and potentially compared while exploring.

Secondly, the choice of which information is portrayed in the app is important. Since too much information provision risks a bloating of the app (and therefore a reduced user experience), the final campus data needs to be carefully selected and filtered for in-app display. The establishment of a clear hierarchy between important, frequently used data and less important information also contributes to this.

3.3.2 Designing a user-friendly campus map

3.3.3 Chunking of information within the app

3.4 User interface design

4 Implementation

4.1 Collection of geodata and POI

4.1.1 Overview of needed geodata

4.1.2 Collection data from OSM via Overpass Turbo API

4.1.3 Enhancing OSM data manually in QGIS

4.1.4 Export of geodata

4.2 Generation of digital campus map

4.2.1 Data import and conversion in Unity

4.2.2 Mesh generation for streets, green areas, water and 3d buildings

4.2.3 Map design

4.2.4 Integration into the Flutter app

4.3 Navigation system development

4.3.1 Representation of geodata for navigation

4.3.2 Routing across the campus

4.3.3 Time estimation for routes

4.3.4 Embedding the current user location via GPS

4.4 Interactive information layer development

4.4.1 Collection of campus relevant information from the web

4.4.2 Processing of information and internal representation

4.5 User interface development

4.5.1 Navigation system

4.5.2 Information layer

4.5.3 Enhancing the user experience with additional screens and features

5 Evaluation

5.1 Navigation system verification

5.2 Information layer verification

6 Conclusion

List of Tables

List of Figures

Bibliography

[1] I. Getting, "Perspective/navigation-the global positioning system," *IEEE Spectrum*, vol. 30, no. 12, pp. 36–38, 1993.

[2] S. Kumar and K. B. Moore, "The evolution of global positioning system gps technology," *Journal of Science Education and Technology*, vol. 11, pp. 59–80, 2002.

[3] S. Farahani, "Chapter 7 - location estimation methods," pp. 225–246, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780750683937000078

[4] Google, "Fused location provider api | google for developers," 2023. [Online]. Available: https://developers.google.com/location-context/fused-location-provider?hl=de

[5] "OpenStreetMap," Oct. 2023, [Online; accessed 29. Oct. 2023]. [Online]. Available: https://www.openstreetmap.org/#map=14/53.3797/9.7587

[6] "Overpass turbo – OpenStreetMap Wiki," Sep. 2023, [Online; accessed 29. Oct. 2023]. [Online]. Available: https://wiki.openstreetmap.org/wiki/Overpass_turbo

[7] M. Minghini and F. Frassinelli, "Openstreetmap history for intrinsic quality assessment: Is osm up-to-date?" *Open Geospatial Data, Software and Standards*, vol. 4, 2019. [Online]. Available: https://doi.org/10.1186/s40965-019-0067-x

[8] "Is OSM up-to-date?" May 2022, [Online; accessed 29. Oct. 2023]. [Online]. Available: https://is-osm-uptodate.frafra.eu/#17/52.51274/13.32600

[9] "Google Maps Platform," Jun. 2023, [Online; accessed 30. Oct. 2023]. [Online]. Available: https://developers.google.com/maps?hl=de

[10] "Karten," Oct. 2023, [Online; accessed 30. Oct. 2023]. [Online]. Available: https://www.apple.com/de/maps

[11] H. Mehta, P. Kanani, and P. Lande, "Google maps," *International Journal of Computer Applications*, vol. 178, pp. 41–46, 05 2019.

[12] M. Noto and H. Sato, "A method for the shortest path search by extended dijkstra algorithm," vol. 3, pp. 2316–2320 vol.3, 2000.

[13] "Core Location | Apple Developer Documentation," Oct. 2023, [Online; accessed 29. Oct. 2023]. [Online]. Available: https://developer.apple.com/documentation/corelocation

[14] W. Kang and Y. Han, "Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2906–2916, 2015.

[15] B. Wang, X. Liu, B. Yu, R. Jia, and X. Gan, "Pedestrian dead reckoning based on motion mode recognition using a smartphone," *Sensors*, vol. 18, no. 6, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/6/1811

[16] M. Basso, A. Martinelli, S. Morosi, and F. Sera, "A real-time gnss/pdr navigation

system for mobile devices," *Remote Sensing*, vol. 13, no. 8, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/8/1567

[17] A. Stark, M. Riebeck, and J. Kawalek, "How to design an advanced pedestrian navigation system: Field trial results," pp. 690–694, Sep. 2007.

[18] A. Küpper, *Location-Based Services: Fundamentals and Operation*. John Wiley and Sons Ltd, 2005.

[19] A. Küpper, U. Bareth, and B. Freese, "Geofencing and background tracking – the next features in lbss," *INFORMATIK 2011 - Informatik schafft Communities*, 2011.

[20] P. Sadhukhan, N. Mukherjee, and P. K. Das, *Location-Based Services for Smart Living in Urban Areas*. Cham: Springer International Publishing, 2021, pp. 53–69. [Online]. Available: https://doi.org/10.1007/978-3-030-71288-4_3

[21] S. Rodriguez Garzon and B. Deva, "Geofencing 2.0: Taking location-based notifications to the next level," p. 921–932, 2014. [Online]. Available: https://doi.org/10.1145/2632048.2636093

[22] "Campaign of the Year: Burger King's 'Whopper Detour'," *Marketing Dive*, Dec. 2019, [Online; accessed 28. Nov. 2023]. [Online]. Available: https://www.marketingdive.com/news/burger-king-whopper-detour-mobile-marketer-awards/566224

[23] "Technische Universität Berlin," Nov. 2023, [Online; accessed 16. Nov. 2023]. [Online]. Available: https://www.tu.berlin

[24] "Moses - TU Berlin," Nov. 2023, [Online; accessed 16. Nov. 2023]. [Online]. Available: https://moseskonto.tu-berlin.de/moses/index.html

[25] "studierendenWERK BERLIN - Startseite," Nov. 2023, [Online; accessed 16. Nov. 2023]. [Online]. Available: https://www.stw.berlin

[26] "Selenium," Nov. 2023, [Online; accessed 29. Nov. 2023]. [Online]. Available: https://www.selenium.dev

[27] "beautifulsoup4," Apr. 2023, [Online; accessed 29. Nov. 2023]. [Online]. Available: https://pypi.org/project/beautifulsoup4

[28] "Flutter - Build apps for any screen," Oct. 2023, [Online; accessed 7. Nov. 2023]. [Online]. Available: https://flutter.dev

[29] "Dart packages," Nov. 2023, [Online; accessed 7. Nov. 2023]. [Online]. Available: https://pub.dev

[30] "Echtzeit-Entwicklungsplattform von Unity | 3D, 2D, VR- und AR-Engine," Dec. 2023, [Online; accessed 19. Dec. 2023]. [Online]. Available: https://unity.com/de

Appendices

Appendix 1

```
for($i=1; $i<123; $i++)
{
    echo "work harder! ;)";
}
```