
TODO: Headhight noch zu 12pt zurückstellen

Höhere Technische Bundeslehranstalt Kaindorf an der Sulm

Abteilung Informatik

Diplomarbeit

im Rahmen der Reife- und Diplomprüfung

Königskarte



informatik

Leon Edlinger
Paul Gigler
Andreas Weissl

5BHIF
2024/2025

Betreuer: Prof. DI Johannes Loibner, BSc
Projektpartner: Prof. DI Robert Müllerferli
Datum: MISSING DATE

All rights reserved. No part of the work may be reproduced in any form (printing, photocopying, microfilm or any other process) without the written permission of all authors or processed, duplicated or distributed using electronic systems. The authors assume no liability for the functions of individual programs or parts thereof. In particular, they assume no liability for any consequential damages resulting from the use.

The reproduction of utility names, trade names, product descriptions, etc. in this work, even without special marking, does not justify the assumption that such names are to be regarded as free within the meaning of trademark and trademark protection legislation and may therefore be used by everyone.

Statutory declaration

I declare under oath that I have written the present diploma thesis independently and without outside help, have not used sources and aids other than those indicated and have identified the passages taken from the sources used literally and in terms of content as such.

Ort, Datum

Leon Edlinger

Ort, Datum

Paul Gigler

Ort, Datum

Andreas Weissl

Abstract

Abstract in English

Kurzfassung

Kurzfassung in Deutsch

Thanks

It would not have been possible to carry out this thesis to this extent without the active support of a number of people. We would therefore like to thank everyone who supported us in the implementation of this thesis.

...

Table of Contents

1	Introduction	1
1.1	Team	3
2	Technologies	4
2.1	Frontend	4
2.1.1	Dart	4
2.1.2	Flutter	4
2.2	Backend	5
2.2.1	Java Spring	5
2.2.2	PostgreSQL	5
2.3	Version Control	6
2.3.1	Git	6
2.3.2	GitHub	6
2.4	Map Data	7
2.4.1	OpenStreetMap	7
2.4.2	GraphHopper	7
2.5	Development Tools	8
2.5.1	VS Code	8
2.5.2	IntelliJ	8
2.5.3	Android Studio	8
2.5.4	Postman	9
2.6	Deployment	10
2.6.1	Docker	10
3	Research Questions	11
3.1	Leon Edlinger	11
3.2	Paul Gigler	11
3.3	Andreas Weissl	11
4	Spring Framework	11
4.1	Spring Boot	11
4.2	Spring Data JPA	11
4.3	Lombok	11
4.4	Advantages	11
5	Area Borders	11
5.1	Purpose of Area Borders in the App	11
5.2	Overview of the Convex Hull Algorithm	12
5.3	Use Cases of the Convex Hull in Industry	12
5.4	Alternate Methods for Area Border Calculation	12
5.5	Rationale for Choosing the Convex Hull Method	12
5.6	Integration of the Algorithm into the Backend	12
5.7	Challenges and Adjustments	12
6	Structure of the Backend	12
6.1	Controller Layer	12
6.2	Service Layer	12
6.3	Repository Layer	13
6.4	Persistence Layer (Entity Classes)	13
6.5	Applied Design Principles (DTOs)	13
7	Defining usability	14
7.1	Why it is important	14
7.2	Components of Usability	14
7.3	Fundamental concepts	15
7.4	Challenges in designing for a broad user spectrum	19

8 Usability in context of maps	20
8.1 Basic Analysis of the Google Maps Interface	20
8.2 Identifying Flaws in Google's Design	21
8.3 How Could Specific User Groups Struggle with This Design	21
9 Adaptive algorithms and real-time data integration	22
9.1 Theoretical Framework	22
9.1.1 Traditional Methods for Address Database Management	22
9.1.2 Adaptive Algorithms: Concepts and Applications	22
9.1.3 Real-Time Data Integration Frameworks	22
9.2 Technical Framework	22
9.2.1 Data Sources	22
9.2.2 Adaptive Algorithms	22
9.2.3 Evaluation Metrics	22
10 Traditional Methods for Address Database Management	22
11 Adaptive Algorithms: Concepts and Applications	22
12 Real-Time Data Integration Frameworks	22
13 Implementation of the Backend	22
13.1 Config of Spring Boot (application.properties)	23
13.2 Entity Classes (Structure/Purpose)	23
13.3 JPA-Repositories (DB Access and CRUD Operations)	23
13.4 Service Classes	23
13.5 Rest Controller (API Endpoints and their Functions)	23
14 GraphHopper Setup	23
14.1 Why use GraphHopper?	23
14.2 Configuration	23
14.3 Local hosting	23
15 Working out the Wireframes	23
15.1 Map View	23
15.2 List View	23
15.3 Possible improvements for future versions	23
16 Functional implementation behind the application	24
16.1 Address-Provider	24
16.2 HTTP-Requests	24
16.3 Implementation of the Flutter Map Component	24
17 The app in use	25
17.1 Introducing new users	25
17.2 The app in operation	25
17.3 User Feedback	25
18 Final Thoughts	26
18.1 Leon Edlinger	26
18.2 Paul Gigler	26
18.3 Andreas Weissl	26
19 Meetings	27
20 Working Hours	28
21 Source code directory	29
22 List of figures	30

23 List of tables	31
24 Bibliography	32
25 Abbreviation	34

1 Introduction

Mobile apps are utilized for virtually all aspects of daily life in the modern world. So after we noticed that there is no application that allows the efficient planning of campaigns like the "Sternsinger-Aktion" we asked ourselves why, and furthermore, how hard it is to create an App with intuitive usability with the main purpose of simplifying the process of managing such a campaign and gaining a general overview of the progress made by the groups.

The app needs to comply with specific criteria we defined in cooperation with Prof. DI Robert Müllerferli. He is the main organizer of the campaign in the parish of Lieboch and helped us to work out the key aspects our project should implement. In the finished product, every user should be able to scan a QR-Code, through which the area of this group gets assigned to the device. These areas must be dynamically adjustable, so an admin can coordinate the workload of each area more efficiently. The areas also need to be clearly visible by an outline which gets drawn through "Border" addresses. These border addresses get calculated by an algorithm implemented by us. It should be visible at a glance if there is a "specification", which can be assigned by admins, set for an address. This should be realized through the use of different icons instead of the default icon. Apart from the app itself, we also implemented a web-portal through which administrators can manage and supervise the campaign.

The investigative aspect of this thesis will focus on how components should behave and appear, so that new users can use this tool without requiring a long "onboarding" phase. Interacting with elements should feel familiar, and the limits of what users can and cannot do need to be clearly defined. Because our application also needs a reliable data source to guarantee the consistency and accuracy of marked addresses, we researched ways to keep our database up to date with minimal manual intervention. After defining the project requirements, we noticed the need to determine which addresses qualify as border addresses.

In our context, an area consists of multiple addresses, each with a defined location represented by latitude and longitude coordinates. Border addresses are the addresses that form the outer boundary of an area. For example, given five addresses with the following coordinates:

- A (0,0)
- B (2,0)
- C (0,2)
- D (-2,0)
- E (0,-2)

In this case, addresses B, C, D, and E are border addresses because they outline the area, enclosing A at the center. Thus, we explored different algorithms for this task, compared them in terms of efficiency, selected the most suitable one, and implemented it.

This thesis contains an in-depth description of our thought and development process, as well as the steps we took to achieve our goal of a functional mobile application that can be used by volunteers during the "Sternsinger-Aktion 2025," which took place in the parish of Lieboch in January 2025.

TODO: Wortwiederholung austauschen

The result of this thesis should be a mobile app that provides users with the addresses that they need to visit on this day. They then should be able to easily mark the houses they already visited. If something unusual happens at this address, the user should be able to take note of this, so the organizers have knowledge of it and can account to it in the following year.

TODO: Maybe auf verschiedene Parts aufteilen, also das man zuerst sagt Problemstellung, dann Zielsetzung damit die Introduction übersichtlicher ist

1.1 Team

This thesis was created by three Students attending the BHIF20 at the HTBLA Kaindorf Computer Science Department.

TODO: andis bild anpassen

Leon Edlinger



Database, Admin-Panel

Paul Gigler



Deployment, Mobile App

Andreas Weissl



Backend

2 Technologies

Development would not have been possible without making use of many tools, frameworks and environments. In this chapter each tool used in the creation of our software will be described briefly.

2.1 Frontend

2.1.1 Dart

Dart is a programming language initially designed for web development, with the goal of replacing JavaScript. Today, it is used in a variety of software products, mainly because of the flutter framework. Dart can be compiled for many platforms and architectures, including ARM, x64, RISC-V, JavaScript or WebAssembly and is highly popular for its combination of high-level features, combined with practical language features like garbage collection and optional type annotation. It was developed by Google and is now an open-source project. [Bie19]



Fig. 1: Dart Logo (Source: <https://dart.dev>)

2.1.2 Flutter

Flutter is an open-source software development framework. It allows programmers to compile their applications for different platforms including Web, macOS, iOS as well as Windows and any type of Linux-based systems, all from a single codebase, written in Dart. This allows for more efficient and faster cross-platform development. Another benefit of Google's toolkit are the highly customizable, predefined UI components. Developers can mix and match these components as needed which makes them an applicable choice. [25c][Dag19]

We chose flutter mainly for these reasons, but also because of our previous experience with Java to which Dart is quite similar. Through it, we were able to get started quickly, learn what we need along the way. Having a design through the components was also very helpful and saved us some time.



Fig. 2: Flutter Logo (Source: <https://flutter.dev/>)

2.2 Backend

2.2.1 Java Spring

Java on its own is an object-oriented programming language which is designed to be platform-independent. It allows programs to run on any device that has a Java Virtual Machine installed. Java is particularly useful for its strong security features and extensive community support.



Fig. 3: Java Logo (Source: <https://logodownload.org/java-logo/>)

Java Spring is an open-source framework and is based on Java. The Java Spring Framework is an in-depth programming and configuration language that simplifies development of Java applications. The most used Spring Framework is Spring Boot. Spring Boot simplifies the application process with not much need for preconfigured logic. Other Spring Frameworks which have a huge impact on Java programming are Spring Security for authentication/authorization and Spring Data for database injection. We chose Java Spring as our backend development tool, because of its simplicity and flexibility. As a framework, Spring allowed us to develop a well scalable and maintainable backend to generate APIs and connect to the database.

2.2.2 PostgreSQL

PostgreSQL is an open-source relational database management system. It is known for its reliability and scalability. It complies with SQL standard. Databases, which use PostgreSQL, are designed to handle a huge amount of data efficiently. Those databases support more advanced features such as full-text search and JSON data storage.

One of PostgreSQL's strengths is its flexibility. Developers are able to create custom data types and functions to customize the database to their specific needs. It also provides security mechanisms which include role-based access control and encryption.

We used PostgreSQL in our project as our database due to its ability to handle complex queries and its scalability which then allowed us to manage and store all the data reliably.

2.3 Version Control

2.3.1 Git

Git is a distributed VCS that was developed by Linus Torvalds in 2005. The main benefit of a VCS is to easily keep track of different versions of files. Git is the most widely used option, concerning this area of application. [25i] This is due to the fact it is open-source, therefore free, as well as reliable and easy to learn. It gets used in almost every, but not only, development project, not just for tracking the history of files, but also for developing cooperatively making use of its branching feature. This allows for development while maintaining a stable version on the main branch. [25d]



Fig. 4: Git Logo (Source: <https://git-scm.com/>)

2.3.2 GitHub

GitHub is a platform maintained by Microsoft. As the Name implies, it is based on Git and provides the opportunity to easily share your Git repositories with other users. It is free to use for non-commercial applications and a very popular option when it comes to developing in a team. [25a] Furthermore, it also implements handy extensions that integrate tightly with Git, for example GitHub-Actions, which is their solution for CI/CD Pipelines. There are many alternatives to GitHub, like, GitLab, which is open-source and can be self-hosted or BitBucket, a solution by Atlassian. [25h]



Fig. 5: GitHub Logo (Source: <https://github.com/>)

2.4 Map Data

2.4.1 OpenStreetMap

The project OpenStreetMap (OSM) is an open-source initiative that aims to make high quality map data available to everyone for free. It was created in 2004 by Steve Coast. It was later widely adopted, with the main push being the pricing Google introduced to the embedding of Google Maps, so that today it can be found in all kinds of applications. Reaching from simple embedded web-components, all the way into video games like the Microsoft Flight Simulator, which relies on OSM for its building models. The GIS behind OSM gets maintained by volunteers and is feed not only using GPS-Data but also image data and other different formats.

OSM is the backbone of this thesis and without it, it wouldn't have been possible to develop this tool. [25g] [Wik24]



Fig. 6: OSM Logo (Source: <https://wiki.openstreetmap.org/>)

2.4.2 GraphHopper

GraphHopper (GH) in its core is an open-source routing engine written in Java. It can be deployed as a web server and is used to calculate the optimal route, distance or time between multiple different points. It supports multiple modes of transportation as well as "snap to road" technology. [25e] GH also provides a ready to go API that you can access for free, but we choose to deploy our own instance since the requests we needed to make exceed the limits of the free version. [24]



Fig. 7: GraphHopper Logo (Source: <https://brandfetch.com/graphhopper.com>)

2.5 Development Tools

2.5.1 VS Code

Visual Studio Code (VS Code) is a version of the Code-OSS project with Microsoft-specific customizations, making it a closed-source product licensed by Microsoft. VS Code is highly customizable, with support for community-made plugins and many options for the user to define how things should act and look. [25k] There are many handy features implemented out of the box, such as native VCS support. Since VS Code is an electron based application, it can not only be used on a desktop computer, but also directly in the browser of any device. [25l] [25j]

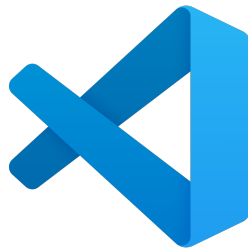


Fig. 8: VS Code Logo (Source: <https://code.visualstudio.com/brand>)

2.5.2 IntelliJ

IntelliJ IDEA is an IDE primarily designed for Java and Kotlin developers, published by JetBrains. In comparison to VS Code, it is more resource-intensive due to included tools like the direct integration of a database connection via the embedded DataGrip version. IntelliJ is the base framework used for other, more specific IDEs by JetBrains, like Android Studio, WebStorm or DataGrip. Its functionality can also be extended through plugins. [25f] [Con25]



Fig. 9: IntelliJ IDEA Logo (Source: <https://www.jetbrains.com/company/brand/>)

2.5.3 Android Studio

Android Studio is based on the IntelliJ framework, but, as the name suggests, with the specification to developing Android apps. Aside from Java and Kotlin based pure Android apps, it can also be used for developing cross-platform applications using the flutter framework. The IDE introduced by Google also features an Emulator for Pixel devices which simplifies testing. Lastly, it provides an intuitive way to compile the app and flash it to a physical Android device using ADB and USB-debugging.

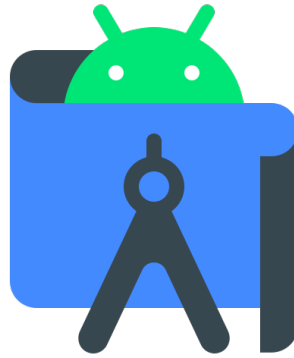


Fig. 10: Android Studio Logo (Source: <https://img.icons8.com/?size=100&id=040Frkjznvcd&format=png&color=000000>)

2.5.4 Postman

Postman is an API platform, primarily used to test APIs. It features collaboration and an automatic code-generation, allowing users to design requests directly in postman and export them into any of the supported languages. We used it to develop and test our API, without a frontend application to minimize potential points of failure.



Fig. 11: Postman Logo (Source: <https://www.postman.com/legal/logo-usage/>)

2.6 Deployment

2.6.1 Docker

Docker is a tool for containerization and one of the most popular options in this field. Through containerization, it is possible to isolate a process from the host machine, in a virtual environment. Furthermore, docker is portable which makes it a great choice for software projects that many developers with different environments work on. It is used to create a controlled infrastructure which guarantees the correct configuration for the production server. We used Docker to host our PostgreSQL database across multiple different systems throughout the development process.



Fig. 12: Docker Logo (Source: <https://www.docker.com/>)

3 Research Questions

3.1 Leon Edlinger

3.2 Paul Gigler

3.3 Andreas Weissl

4 Spring Framework

The backend leverages the **Spring Framework**, a comprehensive framework for enterprise Java development. This section explores its key components and advantages.

4.1 Spring Boot

Spring Boot simplifies configuration and deployment with embedded servers and opinionated setups. This reduces boilerplate code and accelerates development.

4.2 Spring Data JPA

Spring Data JPA provides abstractions for database interactions, streamlining CRUD operations and custom query creation.

4.3 Lombok

Lombok reduces boilerplate code by generating getters, setters, and other methods at compile time, improving code readability and maintainability.

4.4 Advantages

Using Spring enhances productivity, reduces setup complexity, and ensures scalability, making it ideal for this project.

5 Area Borders

The area borders feature addresses the research question by implementing computational geometry algorithms for precise geographical boundary calculations.

5.1 Purpose of Area Borders in the App

Accurate area borders are essential for defining regions based on user input, supporting the app's mapping functionality.

5.2 Overview of the Convex Hull Algorithm

The convex hull algorithm identifies the smallest convex polygon enclosing a set of points, making it a suitable choice for this project.

5.3 Use Cases of the Convex Hull in Industry

Applications of convex hulls in mapping, computer graphics, and robotics highlight their importance in solving real-world problems.

5.4 Alternate Methods for Area Border Calculation

Alternative methods like Voronoi diagrams and alpha shapes were considered but found less suitable due to complexity or computational demands.

5.5 Rationale for Choosing the Convex Hull Method

The convex hull algorithm offers a balance of simplicity, efficiency, and accuracy, aligning with the project's requirements.

5.6 Integration of the Algorithm into the Backend

The algorithm is implemented in the service layer, ensuring smooth integration with other backend components.

5.7 Challenges and Adjustments

Challenges included handling edge cases like collinear points, which were resolved through specific algorithm adjustments.

6 Structure of the Backend

The backend follows a layered architecture to promote separation of concerns, scalability, and maintainability. This section outlines the roles of each layer.

6.1 Controller Layer

The controller layer acts as the interface for incoming HTTP requests, delegating them to appropriate service methods.

6.2 Service Layer

The service layer contains business logic, validating data and coordinating interactions between controllers and repositories.

6.3 Repository Layer

Repositories abstract database operations, allowing the backend to interact with the database without explicit SQL queries.

6.4 Persistence Layer (Entity Classes)

Entity classes define the data model and its mapping to the relational database, ensuring a consistent schema.

6.5 Applied Design Principles (DTOs)

Data Transfer Objects (DTOs) enhance encapsulation and optimize data transfer between layers and external clients.

7 Defining usability

Since my research question "How can user-experience principals add to an intuitive map displayment for nonprofit activities in which people of different technical know-how levels collaborate?" is all about usability, I want to introduce you to its basic concepts and challenges but also provide some examples on how usability can impact a software's revenue and perception.

Usability is a critical aspect of software and interface design, ensuring that users can efficiently and effectively interact with a product or system. Its job is to provide clear feedback and "experiences" to the user, so interactions between software and human feel smooth and straight forward. Because each human being is different in its emotional experiences, it is difficult to design a kind of "one size fits all" solution. Due to this circumstance, many studies and experiments were conducted. [Nie24]

7.1 Why it is important

Usability ensures that users can accomplish their goals with minimal frustration and maximum efficiency. With the increasing reliance on digital tools, usability plays a key role, not only, in shaping user experiences but also accessibility of software for diverse user groups. A well-designed and thought-out usability concept can go a long way from refining a once tedious and complicated to use product, to one that can be operated even by non-familiar users or disabled people. This plays a big part in the inclusion of all age and knowledge groups as well as the general market share through mass adoption because of the easiness.

7.2 Components of Usability

According to Jakob Nielsen, usability consists of five core components. To achieve the best possible usability, each of factors must be taken into account and be improved to its maximum.

- Learnability

How **easy** it is to accomplish basic tasks the first time

- Efficiency

How **quickly** task can be accomplished after an initial learning period

- Memorability

How **memorable** actions are to users so, after an extended period of not using a software

- Error handling

How **many** errors users make while using the design and how **sever** they are

- Satisfaction

How **pleasant** the overall experience of using the product is

[Nie24]

Now that we are aware of these key points, what measures can we take to reach the goal of great usability? According to Nasrullah Hamidli, human-computer-interaction relies on consistency, visibility, feedback, and simplicity. Consistency ensures users do not need to learn new interactions for each task. For example, buttons should look alike and be in a similar location. This makes for a more natural navigation across the product and an overall familiar feel. Simplicity connects directly to this. Its goal is to minimize clutter and make user interfaces easy to understand and provide one, clear way to accomplish a task, not many possible, but complicated and unintuitive ways. It also aims to reduce distractions. Visibility allows users to clearly understand their options at any given moment, this is most often achieved through visual cues, like, grayed out buttons. This goes hand in hand with the feedback aspect, which provides immediate confirmation of actions. Loading indicators, color-changes and alike get used most often.

Another important part of designing a good UI are typography and colors. These can act as parameters for the attention and emotions of users, as well as establish visual hierarchies, which intern, contribute again to a simpler to navigate interface. [Ham23]

7.3 Fundamental concepts

In this section we will look further into these basic concepts and examine what designers can concretely do to improve usability. To make more descriptive, what impacts usability, we will, along the way, modify a simple website.

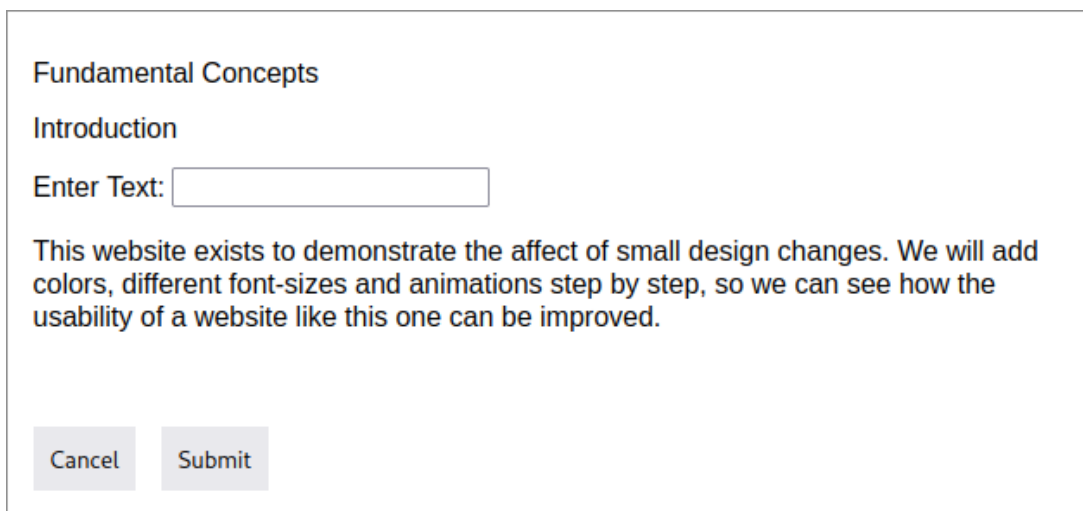
The image shows a web form with a light gray border. At the top, the title 'Fundamental Concepts' is displayed in a bold, dark blue font. Below the title, the word 'Introduction' appears in a smaller, dark blue font. Underneath, the text 'Enter Text:' is followed by a rectangular text input field. Below the input field, a paragraph of text reads: 'This website exists to demonstrate the affect of small design changes. We will add colors, different font-sizes and animations step by step, so we can see how the usability of a website like this one can be improved.' At the bottom of the form, there are two buttons: 'Cancel' and 'Submit', both with a light gray background and dark gray text.

Fig. 13: Starting point of usability example website

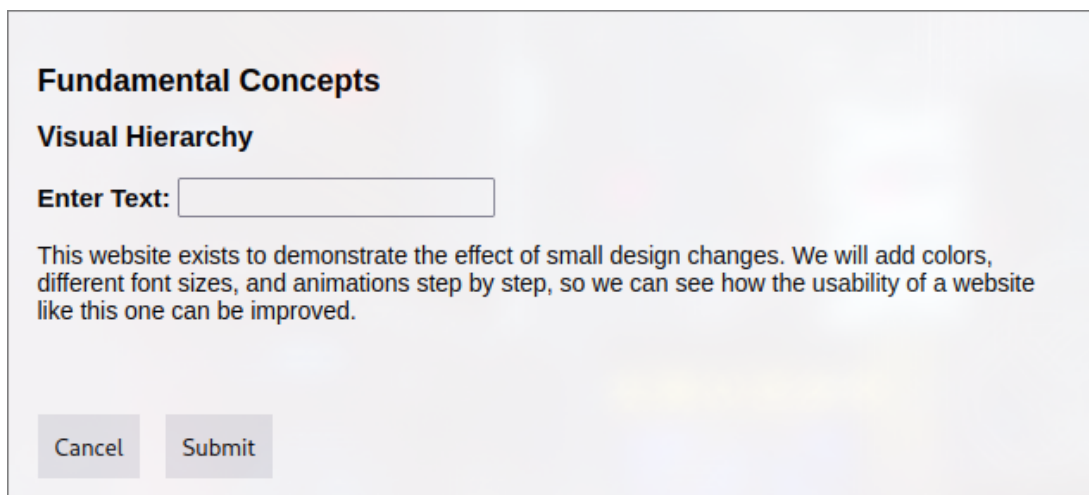
Visual Hierarchy

Starting of with Visual hierarchy. It is the most basic principle in UI/UX design and dictates, how users perceive and navigate content. A strong visual hierarchy contributes to simplicity and ensures that important elements are more visually prominent, guiding users toward essential actions and information. Factors include the text

size, weight and spacing.

In the initial version, visitors of our website are not able to clearly and promptly differentiate between headings and actual text. This is due to the use of only a single font size and weight.

If these attributes get tuned just a little, it becomes suddenly much easier to distinguish primary content from secondary information. Headings are bold and larger, while body text is appropriately sized for readability. Proper spacing and alignment create a structured and pleasant reading experience.



Fundamental Concepts

Visual Hierarchy

Enter Text:

This website exists to demonstrate the effect of small design changes. We will add colors, different font sizes, and animations step by step, so we can see how the usability of a website like this one can be improved.

Cancel Submit

Fig. 14: Example with better visual Hierarchy through different font size and weight

Color Theory & Contrast

Color selection plays a critical role in usability and aesthetics. Well-chosen colors draw the attention to important elements while poor color choices can make content inaccessible, especially for users with visual impairments. The advantages of attention through colors can also be used in reverse, for example, a button to delete something important may be colored red, to indicate a fatal action, but could also be gray, so the user must consciously decide to press it. This acts as a safety measure.

To help designer with their color choices and visualize what colors work together, the color wheel was invented. On it, relations between colors can be shown easily. It consists of three groups of colors.

- **Primary**

These are the fundamental colors that cannot be created by mixing other colors. They vary, depending on the media format they will be used on. On screens the *RGB Model* is in use, while when printing color, cyan, magenta and yellow (*CMY Model*) get used.

- **Secondary**

Secondary colors arise by mixing two primary colors to equal amounts. In designing applications, they often get used for buttons or highlighting important information.

- **Tertiary** They are often associated with a calm and refined aesthetic. They are not quite the same eye-catchers as primary colors, rather they have their value in creating appealing combinations.

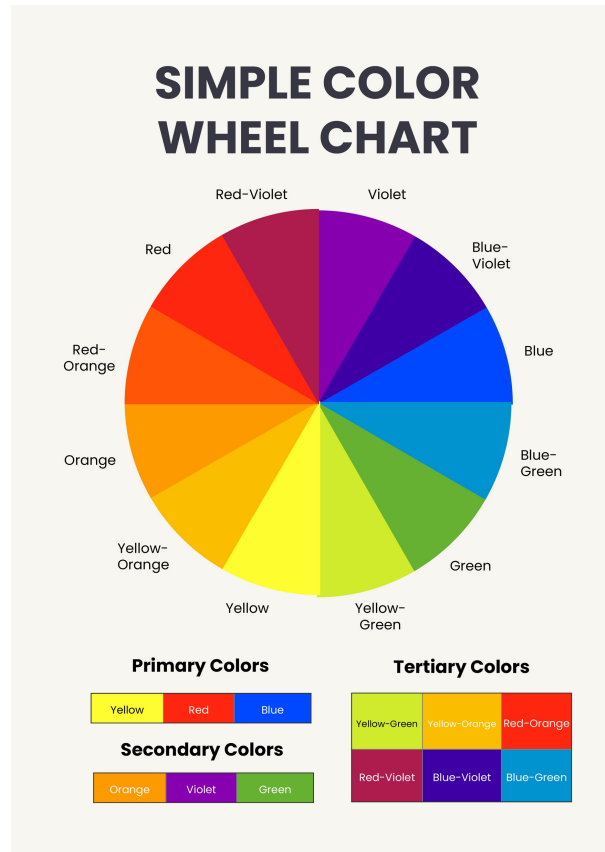
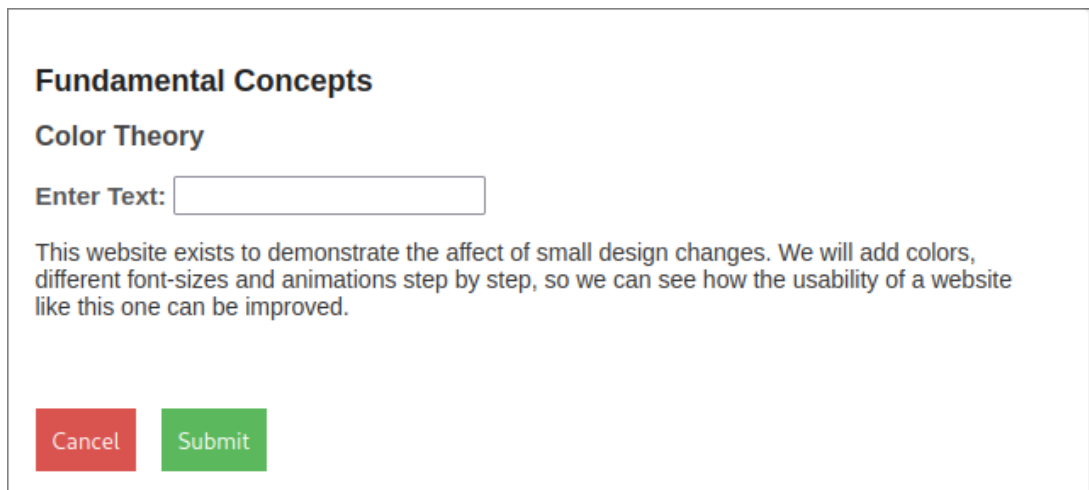


Fig. 15: Color wheel with primary, secondary and tertiary colors sorted (Source: <https://www.template.net/editable/114132/simple-color-wheel-chart>)

Relationship Type	Description	Use case
Complementary	Colors opposite each other	Creating bold contrast
Analogous	Colors next to each other	Harmonious, natural looks
Triadic	Three evenly spaced colors	Balanced, vibrant designs
Split Complementary	One color + two adjacent to its complement	Softer contrast than complementary

Table 1: Key color combinations with effects (Source: <https://colorpage.ai/blog/color-theory-for-beginners>)

The current design doesn't feature any color at all. To highlight the buttons, we can give them a color. Now we also need to change the text color, because, if left as is, the contrast would not be great, and therefore the text harder to read. [25b, vgl.]



Fundamental Concepts

Color Theory

Enter Text:

This website exists to demonstrate the affect of small design changes. We will add colors, different font-sizes and animations step by step, so we can see how the usability of a website like this one can be improved.

Cancel **Submit**

Fig. 16

Consistency

Consistency in design enhances user experience by ensuring that UI elements and interactions are predictable across different states of the application. Also, a consistent aesthetic can create familiarity, brand recognition and improves the overall visual appeal.

Button Placement & Behavior

Buttons are a critical and one of the most common interactive elements in UI design. Their placement, size and responsiveness affects usability and efficiency. Since the first graphical UIs, the actions to move forward were always on the right side. In contrast, actions to cancel are most commonly placed somewhere on the left. If designers choose to deliberately change this universal agreement, it leads to potential user errors. Also, a color-change on hover got implemented, so the user now gets clearly informed that all required fields are populated, and he can move on.

This design highlights the "Submit" Button, and places it in the correct location. Through this, the simplicity to use is improved because users do not have to search for the button that lets them continue.

TODO: Gendern?

Feedback

Providing immediate and clear feedback enhances user confidence and prevents frustration. There are many forms in which such feedback can be provided, for example, vibrations, sounds or pop-up messages. Feedback can also be supplied by something like a grayed-out button, to signal that this action is not currently available. Success messages or error messages also add to the usability of a website, as well as loading indicators and different animations.

7.4 Challenges in designing for a broad user spectrum

Designing for a diverse user base requires the addressing of varying levels of experience, prior knowledge, cognitive abilities, and accessibility needs. Failure to account for these differences can lead to usability issues, preventing certain groups from effectively using a system. Designers must implement features such as adjustable text sizes, screen reader compatibility, and intuitive navigation to ensure accessibility for all users.

Interfaces should always be tailored to the needs and expectations of the end-user. This leads to challenges when the user-group is not clearly defined or consists of people with widely different backgrounds. For example elderly people often need further guidance when interacting with digital solutions than members of younger generations. This leads back to the core components of usability-design, interfaces need to be simple and unmistakable in their functionality. Failing to provide these core concepts will sooner or later result in a frustrated and shrinking user base. [Ham23]

8 Usability in context of maps

In this section we will further inspect the usability of mapping solutions like Google Maps. We will identify some flaws of Google's design choices and how they could influence specific user groups.

8.1 Basic Analysis of the Google Maps Interface

Google Maps is one of the most widely used mapping and navigation applications globally. Its feature set includes real-time traffic updates, route planning, and location discovery. Generally speaking, it is quite difficult to design a simultaneously user-friendly and functional mapping application. Maps get overloaded and confusing quite easily. They are bloated with information like street names, house numbers, borders, and geographical features like rivers or lakes. Due to this fact, maps are not easy to design according to the principles of usability. But Google developed a very good and intuitive concept on which we now will take a look.

The interface of Google Maps consists of a map, search bar, and a menu for additional settings. The search function is prominently displayed as it is the most common tool used. This is a good use of *visual hierarchy* as the initial focus when opening the app immediately gets drawn to the bigger hint text in the search bar on top. Below it, there is a list of buttons, so users can quickly search for local places that match a specific category, like restaurants, cafés, or gas stations. Notably, there are no buttons for movement actions such as zooming or panning; all this is controlled through swipe and pinch gestures directly on the map. [BG08]

To start navigating to a specific area, you can search for the street name and house number, or the name of a company or other details. Maps gives you recommendations and tries to provide auto-suggestions for your target. When the user selects a destination, the navigation can be started through a big blue button. This is an example of applied *color theory*. The route gets marked by an again bright blue line, which creates a good contrast to the other colors used on the map and captures the attention of the user. This line also has multiple purposes other than displaying the route; for example, if a part is orange, that means the traffic at this point is beginning to jam. If then there is a full stop traffic jam, the line turns red at this section. Also, icons for speed cameras or accidents that other users reported get displayed along it.

One of the core strengths of Google Maps is its interactive and responsive design. Users can zoom in and out using intuitive pinch gestures on mobile devices or scroll actions on desktops. The transition between zoom levels is smooth, preserving context and avoiding disorientation through too big scaling steps. Additionally, the map changes depending on the zoom level. Street names and buildings get displayed only if the user has zoomed in enough. The same happens with markers for businesses and other map data. Through this concept, Google ensures that users do not get overwhelmed. [BG08]

8.2 Identifying Flaws in Google's Design

While Maps is a well-polished product, it is not perfect. One significant flaw is the cognitive overload caused by excessive information. The inclusion of business listings, other suggested routes, live traffic data, and user-generated content can make it difficult for users to focus on their primary navigation tasks. [BG08]

8.3 How Could Specific User Groups Struggle with This Design

Google Maps caters to a broad spectrum of users, but its design can pose difficulties for certain demographics:

Elderly Users: Many elderly individuals may find the interface overwhelming due to small text sizes, densely packed information, and complex menus. Their unfamiliarity with modern digital navigation tools may lead to confusion, especially when trying to search for locations or adjust route preferences. A lack of prominent, simplified navigation options tailored to this group amplifies the issue. [All22]

Users with Low Digital Literacy: People who are not well-versed in digital technology could struggle with Google Maps' multitude of features. They may have difficulty understanding icons, switching between different map modes, or using advanced functionalities like saved locations and street view. A more guided and *simplified* mode could enhance their experience.

Users with Disabilities: Visually impaired users may struggle with *insufficient contrast*, small icons, and the *lack of tactile feedback*. While screen readers can assist, Google Maps does not always provide clear, structured data for these tools. Additionally, users with motor impairments may find it hard to navigate menus and interact with small buttons, particularly on touch screens. [Fro+19]

By addressing these usability concerns, Google Maps could enhance its interface to be more intuitive and accessible for diverse user groups.

9 Adaptive algorithms and real-time data integration

9.1 Theoretical Framework

9.1.1 Traditional Methods for Address Database Management

9.1.2 Adaptive Algorithms: Concepts and Applications

9.1.3 Real-Time Data Integration Frameworks

9.2 Technical Framework

9.2.1 Data Sources

9.2.1.1 GPS Data

9.2.1.2 External APIs

9.2.1.3 User Inputs

9.2.2 Adaptive Algorithms

9.2.2.1 Fuzzy Matching

9.2.2.2 Machine Learning Model

9.2.2.3 Rule-Based Filters

9.2.2.4 Dynamic Duplicate Resolution

9.2.2.5 Real-Time Address Normalization

9.2.3 Evaluation Metrics

9.2.3.1 Accuracy

9.2.3.2 Latency

10 Traditional Methods for Address Database Management

11 Adaptive Algorithms: Concepts and Applications

12 Real-Time Data Integration Frameworks

13 Implementation of the Backend

The backend implementation combines theoretical concepts with practical solutions to ensure functionality and scalability.

13.1 Config of Spring Boot (application.properties)

The `application.properties` file configures essential settings, including database connections, logging, and server parameters.

13.2 Entity Classes (Structure/Purpose)

Entity classes define the application's data model, using annotations to map fields to database tables.

13.3 JPA-Repositories (DB Access and CRUD Operations)

Repositories simplify database access by providing methods for CRUD operations and enabling custom queries.

13.4 Service Classes

Service classes encapsulate business logic, coordinating data flow between controllers and repositories.

13.5 Rest Controller (API Endpoints and their Functions)

REST controllers define API endpoints, processing requests and returning responses to ensure seamless interaction with the frontend.

14 GraphHopper Setup

14.1 Why use GraphHopper?

14.2 Configuration

14.3 Local hosting

15 Working out the Wireframes

15.1 Map View

15.2 List View

15.3 Possible improvements for future versions

16 Functional implementation behind the application

16.1 Address-Provider

16.2 HTTP-Requests

16.3 Implementation of the Flutter Map Component

17 The app in use

17.1 Introducing new users

17.2 The app in operation

17.3 User Feedback

18 Final Thoughts

18.1 Leon Edlinger

18.2 Paul Gigler

18.3 Andreas Weiszl

19 Meetings

Protokolle der Meetings, vielleicht auch ein zeitplan wann immer und wie lang

20 Working Hours

Arbeitspaket-Nr.	Beschreibung	Dauer
1	Einführung und Einarbeitung	8 h
2	Grundkonzept erstellen	8 h
3	Struktur der App festlegen	6 h
5	Wifi-Socket in App implementieren	39 h
6	Write-Funktionalität in App implementieren	14 h
7	Read-Funktionalität in App implementieren	19 h
8	Trim-Funktionalität in App implementieren	10 h
9	Konfigurationsmöglichkeiten für Flug in App implementieren	16 h
10	Höhenregelung-Funktionalität in App implementieren	14 h
12	Graphische Darstellung der Flugdaten	18 h
14	App testen und debuggen	19 h
26	Gesamtkonzept testen und debuggen	16 h
Summe		187 h

Table 2: Arbeitszeitznachweis

21 Source code directory

Source Code directory, kein plan was des is

22 List of figures

1	Dart Logo (Source: https://dart.dev)	4
2	Flutter Logo (Source: https://flutter.dev/)	4
3	Java Logo (Source: https://logodownload.org/java-logo/)	5
4	Git Logo (Source: https://git-scm.com/)	6
5	GitHub Logo (Source: https://github.com/)	6
6	OSM Logo (Source: https://wiki.openstreetmap.org/)	7
7	GraphHopper Logo (Source: https://brandfetch.com/graphhopper.com)	7
8	VS Code Logo (Source: https://code.visualstudio.com/brand)	8
9	IntelliJ IDEA Logo (Source: https://www.jetbrains.com/company/brand/)	8
10	Android Studio Logo (Source: https://img.icons8.com/?size=100&id=040Frkjznvcd&format=png&color=000000)	9
11	Postman Logo (Source: https://www.postman.com/legal/logo-usage/)	9
12	Docker Logo (Source: https://www.docker.com/)	10
13	Starting point of usability example website	15
14	Example with better visual Hierarchy through different font size and weight	16
15	Color wheel with primary, secondary and tertiary colors sorted (Source: https://www.template.net/editable/114132/simple-color-wheel-chart)	17
16	18

23 List of tables

1 Key color combinations with effects(Source: <https://colorpage.ai/blog/color-theory-for-beginners>)

2 Arbeitszeitrachweis 28

24 Bibliography

- [24] *Host Your Own Worldwide Route Calculator With GraphHopper - GraphHopper Directions API*. [Online; accessed 30. Jan. 2025]. Apr. 2024. URL: <https://www.graphhopper.com/blog/2022/06/27/host-your-own-worldwide-route-calculator-with-graphhopper>.
- [25a] *About GitHub and Git - GitHub Docs*. [Online; accessed 16. Feb. 2025]. Feb. 2025. URL: <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>.
- [25b] *Color Theory for Beginners: Essential Design Tips*. [Online; accessed 4. Mar. 2025]. Mar. 2025. URL: <https://colorpage.ai/blog/color-theory-for-beginners>.
- [25c] *flutter/README.md at master · flutter/flutter*. [Online; accessed 23. Jan. 2025]. Jan. 2025. URL: <https://github.com/flutter/flutter/blob/master/README.md>.
- [25d] *Git - Branching and Merging*. [Online; accessed 16. Feb. 2025]. Feb. 2025. URL: <https://git-scm.com/about/branching-and-merging>.
- [25e] *graphhopper*. [Online; accessed 30. Jan. 2025]. Jan. 2025. URL: <https://github.com/graphhopper/graphhopper>.
- [25f] *IntelliJ IDEA – the Leading Java and Kotlin IDE*. [Online; accessed 30. Jan. 2025]. Jan. 2025. URL: <https://www.jetbrains.com/idea>.
- [25g] *OSM: The simple map that became a global movement*. [Online; accessed 28. Jan. 2025]. Jan. 2025. URL: <https://web.archive.org/web/20240420144212/https://www.directionsmag.com/article/1163>.
- [25h] *Quickstart for GitHub Actions - GitHub Docs*. [Online; accessed 16. Feb. 2025]. Feb. 2025. URL: <https://docs.github.com/en/actions/writing-workflows/quickstart>.
- [25i] *Stack Overflow Developer Survey 2022*. [Online; accessed 16. Feb. 2025]. Feb. 2025. URL: <https://survey.stackoverflow.co/2022/#version-control-version-control-system>.
- [25j] *Visual Studio Code for the Web*. [Online; accessed 30. Jan. 2025]. Jan. 2025. URL: <https://code.visualstudio.com/docs/editor/vscode-web>.
- [25k] *vscode*. [Online; accessed 30. Jan. 2025]. Jan. 2025. URL: <https://github.com/microsoft/vscode>.
- [25l] *What is the Visual Studio Code editor built on*. [Online; accessed 30. Jan. 2025]. Jan. 2025. URL: <https://stackoverflow.com/questions/29966093/what-is-the-visual-studio-code-editor-built-on>.
- [All22] Allyant. “Is Google Maps Accessible?” In: *Allyant* (2022). URL: <https://allyant.com/is-google-maps-accessible/>.
- [BG08] Sarah E Battersby and Kirk Goldsberry. “User-centered design for digital map navigation tools”. In: *Cartography and Geographic Information Science* 35.4 (2008), pp. 233–244.
- [Bie19] Alessandro Biessek. *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2*. Packt Publishing Ltd, 2019.
- [Con25] Contributors to Wikimedia projects. *IntelliJ IDEA - Wikipedia*. [Online; accessed 30. Jan. 2025]. Jan. 2025. URL: https://en.wikipedia.org/w/index.php?title=IntelliJ_IDEA&oldid=1272820016.
- [Dag19] Lukas Dagne. “Flutter for cross-platform App and SDK development”. In: (2019).
- [Fro+19] Jon E Froehlich et al. “Grand challenges in accessible maps”. In: *Interactions* 26.2 (2019), pp. 78–81.
- [Ham23] Nasrullah Hamidli. “Introduction to UI/UX design: key concepts and principles”. In: *Academia*. URL: https://www.academia.edu/98036432/Introduction_to_UI_UX_Design_Key_Concepts_and_Principles [accessed 2024-04-27] (2023).
- [Nie24] Jakob Nielsen. “Usability 101: Introduction to Usability”. In: *Nielsen Norman Group* (Jan. 2024). URL: <https://www.nngroup.com/articles/usability-101-introduction-to-usability>.

- [Wik24] OpenStreetMap Wiki. *Main Page — OpenStreetMap Wiki*. [Online; accessed 28-January-2025]. 2024.
URL: https://wiki.openstreetmap.org/w/index.php?title=Main_Page&oldid=2752444.

25 Abbreviation

VCS	Version Control System
API	Application Programming Interface
CI CD	Continuous Integration & Continuous Deployment
OSM	Open Street Map
GIS	Geo Information System
GH	GraphHopper
IDE	Integrated Development Environment
GPS	Global Positioning System
GUI	Graphical User Interface
ADB	Android Debugging Bridge
USB	Universal Serial Bus
REST	REpresentational State Transfer
JVM	Java Virtual Machine
RGB	Red Green Blue
CMY	Cyan Magenta Yellow