

**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

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Team . . . . .	2
<b>2</b>	<b>Technologies</b>	<b>3</b>
2.1	LaTeX . . . . .	3
2.2	Frontend . . . . .	3
2.2.1	Dart . . . . .	3
2.2.2	Flutter . . . . .	3
2.3	Backend . . . . .	5
2.3.1	Java Spring . . . . .	5
2.3.2	PostgreSQL . . . . .	5
2.4	Version Control . . . . .	6
2.4.1	Git . . . . .	6
2.4.2	GitHub . . . . .	6
2.5	Map Data . . . . .	7
2.5.1	OpenStreetMap . . . . .	7
2.5.2	Graphhopper . . . . .	7
2.6	Development Tools . . . . .	8
2.6.1	VS Code . . . . .	8
2.6.2	IntelliJ . . . . .	8
2.6.3	Android Studio . . . . .	8
2.6.4	Postman . . . . .	8
2.6.5	Figma . . . . .	8
2.7	Deployment . . . . .	9
2.7.1	Docker . . . . .	9
2.7.2	Uberspace . . . . .	9
2.7.3	Webmin . . . . .	9
<b>3</b>	<b>Research Questions</b>	<b>9</b>
3.1	Leon Edlinger . . . . .	9
3.2	Paul Gigler . . . . .	9
3.3	Andreas Weissl . . . . .	9
<b>4</b>	<b>Spring Framework</b>	<b>9</b>
4.1	Spring Boot . . . . .	9
4.2	Spring Data JPA . . . . .	9
4.3	Lombok . . . . .	9
4.4	Advantages . . . . .	9
<b>5</b>	<b>Area Borders</b>	<b>9</b>
5.1	Purpose of Area Borders in the App . . . . .	10
5.2	Overview of the Convex Hull Algorithm . . . . .	10
5.3	Use Cases of the Convex Hull in Industry . . . . .	10
5.4	Alternate Methods for Area Border Calculation . . . . .	10
5.5	Rationale for Choosing the Convex Hull Method . . . . .	10
5.6	Integration of the Algorithm into the Backend . . . . .	10
5.7	Challenges and Adjustments . . . . .	10
<b>6</b>	<b>Structure of the Backend</b>	<b>10</b>
6.1	Controller Layer . . . . .	10
6.2	Service Layer . . . . .	11
6.3	Repository Layer . . . . .	11
6.4	Persistence Layer (Entity Classes) . . . . .	11
6.5	Applied Design Principles (DTOs) . . . . .	11

<b>7</b>	<b>Defining usability</b>	<b>11</b>
7.1	Why it is important . . . . .	11
7.2	Fundamental concepts of usability . . . . .	11
7.3	Challenges in designing for a broad user spectrum . . . . .	11
<b>8</b>	<b>Usability in context of maps</b>	<b>12</b>
8.1	Basic Analysis of the Google Maps interface . . . . .	12
8.2	Identifying Flaws in Googles Design . . . . .	12
8.3	How could specific user groups struggle with this design . . . . .	12
<b>9</b>	<b>Adaptive algorithms and real-time data integration</b>	<b>14</b>
9.1	Theoretical Framework . . . . .	14
9.1.1	Traditional Methods for Address Database Management . . . . .	14
9.1.2	Adaptive Algorithms: Concepts and Applications . . . . .	14
9.1.3	Real-Time Data Integration Frameworks . . . . .	14
9.2	Technical Framework . . . . .	14
9.2.1	Data Sources . . . . .	14
9.2.2	Adaptive Algorithms . . . . .	14
9.2.3	Evaluation Metrics . . . . .	14
<b>10</b>	<b>Traditional Methods for Address Database Management</b>	<b>14</b>
<b>11</b>	<b>Adaptive Algorithms: Concepts and Applications</b>	<b>14</b>
<b>12</b>	<b>Real-Time Data Integration Frameworks</b>	<b>14</b>
<b>13</b>	<b>Implementation of the Backend</b>	<b>14</b>
13.1	Config of Spring Boot (application.properties) . . . . .	15
13.2	Entity Classes (Structure/Purpose) . . . . .	15
13.3	JPA-Repositories (DB Access and CRUD Operations) . . . . .	15
13.4	Service Classes . . . . .	15
13.5	Rest Controller (API Endpoints and their Functions) . . . . .	15
<b>14</b>	<b>GraphHopper Setup</b>	<b>15</b>
14.1	Why use GraphHopper? . . . . .	15
14.2	Configuration . . . . .	15
14.3	Local hosting . . . . .	15
<b>15</b>	<b>Working out the Wireframes</b>	<b>15</b>
15.1	Map View . . . . .	15
15.2	List View . . . . .	15
15.3	Possible improvements for future versions . . . . .	15
<b>16</b>	<b>Functional implementation behind the application</b>	<b>16</b>
16.1	Address-Provider . . . . .	16
16.2	HTTP-Requests . . . . .	16
16.3	Implementation of the Flutter Map Component . . . . .	16
<b>17</b>	<b>The app in use</b>	<b>17</b>
17.1	Introducing new users . . . . .	17
17.2	The app in operation . . . . .	17
17.3	User Feedback . . . . .	17
<b>18</b>	<b>Implementation Admin Panel</b>	<b>18</b>
18.1	Navigation . . . . .	18
18.2	AddressPage . . . . .	18
18.2.1	Add Address . . . . .	19
18.2.2	Edit Address . . . . .	19
18.2.3	Delete Address . . . . .	19
18.2.4	Validation . . . . .	20

---

18.2.5	Additional Functionalities . . . . .	22
18.2.6	Filter . . . . .	24
18.3	ListEditPage . . . . .	25
18.4	Components . . . . .	26
18.4.1	MapComponent . . . . .	26
18.4.2	DatabaseViewComponent . . . . .	26
18.4.3	PDFSaver . . . . .	26
18.4.4	AdminAddressProvider . . . . .	26
18.4.5	CustomHttpClient . . . . .	26
18.5	Models . . . . .	26
18.5.1	AreaWithBorder . . . . .	26
18.5.2	ScreenItem . . . . .	26
18.6	Widgets . . . . .	26
18.6.1	InputField . . . . .	26
18.6.2	FilterRow . . . . .	26
<b>19</b>	<b>Final Thoughts</b>	<b>26</b>
19.1	Leon Edlinger . . . . .	26
19.2	Paul Gigler . . . . .	26
19.3	Andreas Weissl . . . . .	26
<b>20</b>	<b>Meetings</b>	<b>27</b>
<b>21</b>	<b>Working Hours</b>	<b>28</b>
<b>22</b>	<b>Source code directory</b>	<b>29</b>
<b>23</b>	<b>List of figures</b>	<b>30</b>
<b>24</b>	<b>List of tables</b>	<b>31</b>
<b>25</b>	<b>Bibliography</b>	<b>32</b>
<b>26</b>	<b>Abbreviation</b>	<b>33</b>



# 1 Introduction

TODO: Is halt die frage ob ma den anfang einfach so schreiben, war ja eigentlich net ganz so xD

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 an "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.

TODO: vielleicht noch was rein bezüglich der borders und dann unten nurmehr drauf referenzieren?

The research part of this thesis will be dedicated to how components should act and look, so that new users can use this tool without requiring a long "onboarding" phase. It should feel familiar to interact with elements and the borders of what users can and can not 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, without the need of much manual intervention. After defining the project requirements, we noticed that we need to calculate which addresses are border addresses. So we decided to take a look into different algorithms for this task and compare them concerning their efficiency, decide on one of them and implement it.

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

## 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 LaTeX

Hier kommt eine Beschreibung zu Latex hin

### 2.2 Frontend

#### 2.2.1 Dart

Dart is a programming language initially designed for web development, with the goal, of replacing JavaScript, in mind. Today it gets used in a variety of software products, mainly because of the flutter framework. It can be compiled for many platforms and architectures (ARM, x64, RISC-V, JavaScript or WebAssembly) and is loved for its combination of High-Level Features, with practical language features like Garbage collection and optional Type annotation. It was developed by Google and is now an open-source project.

(Flutter for Beginners, n.d.)



#### 2.2.2 Flutter

Flutter is an Open-Source software development framework. It allows programmers to compile their application for different platforms including Web, macOS, IOS as well as Windows and any type of Linux-based systems, all from one code-base, 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 however needed which makes them an applicable choice.

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.

("flutter/README.md at master · flutter/flutter", 2025) (Dagne, 2019)



## 2.3 Backend

### 2.3.1 Java Spring

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet,

### 2.3.2 PostgreSQL

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet,

## 2.4 Version Control

### 2.4.1 Git

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet,

### 2.4.2 GitHub

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet,

## 2.5 Map Data

### 2.5.1 OpenStreetMap

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet,

### 2.5.2 Graphhopper

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet,

## 2.6 Development Tools

### 2.6.1 VS Code

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

### 2.6.2 IntelliJ

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

### 2.6.3 Android Studio

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

### 2.6.4 Postman

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

### 2.6.5 Figma

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.



## 2.7 Deployment

### 2.7.1 Docker

### 2.7.2 Uberspace

### 2.7.3 Webmin

## 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**

## **7.1 Why it is important**

## **7.2 Fundamental concepts of usability**

## **7.3 Challenges in designing for a broad user spectrum**

## **8 Usability in context of maps**

### **8.1 Basic Analysis of the Google Maps interface**

### **8.2 Identifying Flaws in Googles Design**

### **8.3 How could specific user groups struggle with this design**



## **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 Implementation Admin Panel

**TODO: Bilder bearbeiten, weil gerade keine Daten angezeigt werden beim deployten Admin-Panel**

The Admin Panel is a Flutter-based administrative dashboard that allows the administrator to efficiently manage all addresses in the application, to plan future "Sternsinger" events and assign designated zones to groups. This zoning ensures organized distribution of participants. It enables CRUD (Create, Read, Update, Delete) operations on addresses, streets and zones. These features make it easy for the administrator to quickly address issues and make changes to areas that participants need to visit.

### 18.1 Navigation

The file `AdminNavigation` is used to navigate between the different pages of the Admin Panel. The navigation can be done via a sidebar on the left which can be set visible via a button on the top left of the screen. The widget contains a list of pages and maintains an internal state ( `indexState` ) to keep track of the currently selected page. Whenever a page is selected in the sidebar, the `indexState` is updated and the corresponding page is displayed.

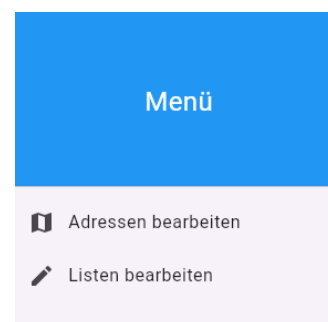


Abb. 1: Navigation in Admin-Panel

### 18.2 AddressPage

The `AddressPage` displays all addresses in the database. With it, the administrator can add, edit and delete addresses into the database. All addresses are shown either in the `MapComponent` or the `DatabaseViewComponent` on the right. On the left of the page are `InputFields` (18.6.1), which are used to enter new information about a new address, or edit an existing one. Overlapping the `MapComponent` there are:

- a field to filter the addresses displayed
- a button with a dropdown menu to select an edit a street
- a switch to toggle between the `MapComponent` and the `DatabaseViewComponent`
- a information box on the bottom left corner to display the selected coordinates, which is only visible when the admin presses the map

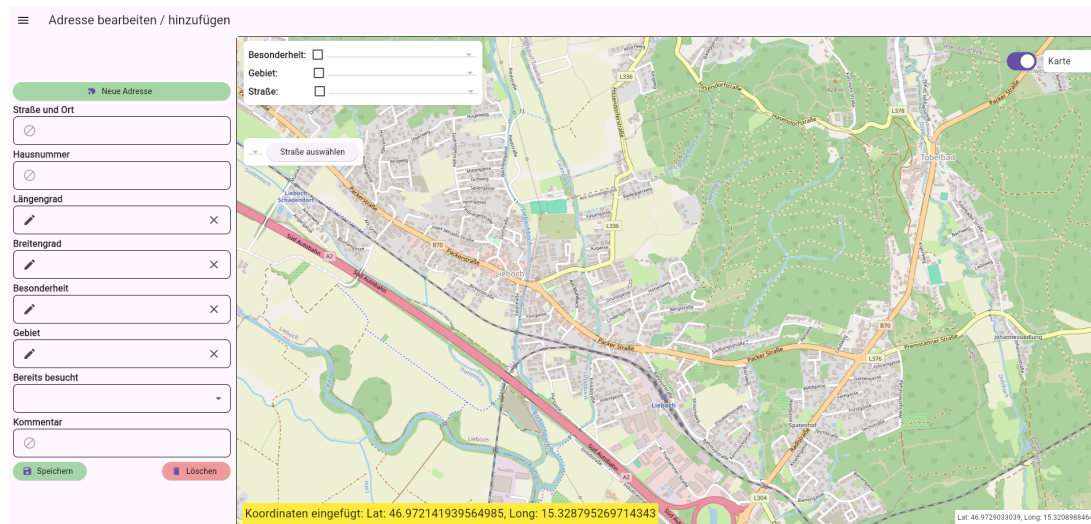


Abb. 2: AddressPage

### 18.2.1 Add Address

To add a new address, the button "Neue Adresse" is pressed. This triggers the `onClickNewAddress` method, which clears all `InputFields` and sets the boolean variable `isNewAddress` to `true`. This variable indicates that a new address is being saved. The cleared fields can then be filled with the new address information. When the "Speichern" button is pressed, the `saveAddress` method is called. This method performs validation checks and verifies whether the address already exists in the database. If the address is valid and not a duplicate, the `AdminAddressProvider` is called to add the new address to the database.

### 18.2.2 Edit Address

Existing addresses can be edited, by selecting an address in the `MapComponent` or the `DatabaseViewComponent`. The `InputFields` are then filled with the information of the selected address and however, in this case, the boolean variable `isNewAddress` is set to `false`, to indicate that an existing address or addresses are being updated. The admin can then edit the information and press the "Speichern" button. This triggers the `saveAddress` method, which performs the same validation checks as when adding a new address. The `AdminAddressProvider` updates the selected addresses in the database.

### 18.2.3 Delete Address

After selecting addresses in the `MapComponent` or the `DatabaseViewComponent`, the admin can press the "Löschen" button to delete them. This triggers the `deleteAddress` method, which calls the `AdminAddressProvider` to delete the selected addresses.

The method also triggers the `showDeleteDialog` method, which displays a `AlertDialog` to confirm the action to prevent accidental deletions.

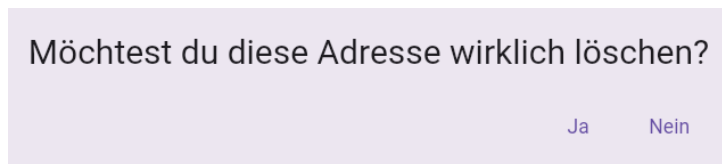


Abb. 3: Dialog to confirm deletion

### 18.2.4 Validation

Validation is the process of checking that data meets specific criteria before it is accepted and added. It is crucial to ensure that the data entered is correct.

#### 18.2.4.1 isDuplicateAddress

To determine whether an edited or newly added address already exists, All addresses are compared with the new address. It is called in the `saveAddress` method. If it already exists, the method returns `true`, otherwise `false`. A duplicate address is identified by the following criteria:

- street name
- postal code
- house number

```
bool isDuplicateAddress(List<Address> existingAddresses, Address newAddress) {  
    return existingAddresses.any((existing) =>  
        existing.street.name == newAddress.street.name &&  
        existing.street.postalCode == newAddress.street.postalCode &&  
        existing.houseNumber == newAddress.houseNumber  
    );  
}
```

Quellcode 1: isDuplicateAddress method

#### 18.2.4.2 validateAddressFields

To make sure that all `InputFields` are filled, the `validateAddressFields` is called in the `saveAddress` (18.1) method. This method needs an `Address`. So first an new `Address` is made and passed to `validateAddressFields`. It checks if all fields are filled and returns `true` if they are, otherwise `false`.

```
bool validateAddressFields(Address address) {  
    if (address.street.name.isEmpty) {  
        showNotification("Strasse fehlt", () => showAddAddressNotification =  
            true);  
    } else if (address.houseNumber.isEmpty) {
```

```

        showNotification("Hausnummer fehlt", () => showAddAddressNotification
            = true);
    } else if (address.specialFeature.text.isEmpty) {
        showNotification("Besonderheit fehlt", () =>
            showAddAddressNotification = true);
    } else if (address.area.desc.isEmpty) {
        showNotification("Gebiet fehlt", () => showAddAddressNotification =
            true);
    } else if (address.latitude == 0.0 || address.longitude == 0.0) {
        showNotification("Koordinaten fehlen", () =>
            showAddAddressNotification = true);
    } else {
        return true;
    }
    return false;
}

```

Quellcode 2: InputFormatter in Inputfield

### 18.2.4.3 InputField Validation

The `InputField` validates **latitude** and **longitude** inputs to ensure their correctness. Validation is applied only when the `isNumberInput` parameter is set to true. If that is the case, then the `inputFormatter` is passed to the `textfield` to validate the input. This `inputFormatter` guarantees that only valid inputs are accepted, preventing incorrect entries. These are the three validators used in the `InputField`:

- The first `FilteringTextInputFormatter` allows only numbers and dots.
- The first `TextInputFormatter` checks if the input contains more than one dot.
- The second `TextInputFormatter` guarantees that there are no more than three digits before the decimal point.

```

inputFormatters: widget.isNumberInput == true
? <TextInputFormatter>[
    FilteringTextInputFormatter.allow(RegExp('[0-9.]')),
    TextInputFormatter.withFunction(
        (oldValue, newValue) {
            if (newValue.text.split('.').length - 1 > 1) {
                return oldValue;
            }
            return newValue;
        })
    ],
    TextInputFormatter.withFunction(

```

```

        (oldValue, newValue) {
            String text = newValue.text;
            if (text.isNotEmpty) {
                final parts = text.split('.');
                if (parts[0].length > 3) {
                    return oldValue;
                }
            }
            return newValue;
        })),
    ]
    : null,

```

Quellcode 3: InputFormatter in Inputfield

## 18.2.5 Additional Functionalities

### 18.2.5.1 Notification

To inform the administrator about the success or failure of an operation, a `Notification` is displayed on the bottom left over the `MapComponent`. This notification appears when an address is added, edited, deleted, or when validation fails.

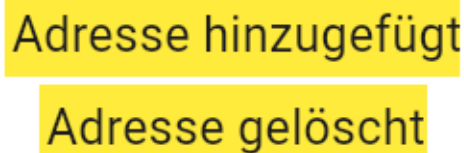


Abb. 4: Notification examples

To display this notification, the `showNotification` method is called. This method sets the `notificationVisible` variable to `true` and starts a `Timer` to turn it set it back to `false` three seconds later, so it goes away in a short time. This method accepts a message as a parameter, which is saved in the `notificationText` variable.

```

void showNotification(String message) {
    setState(() {
        notificationText = message;
        notificationVisible = true;
    });
    Timer(Duration(seconds: 3), () {
        setState(() {
            notificationVisible = false;
        });
    });
}

```

```
});
}
```

Quellcode 4: showNotification method

When the `notificationVisible` variable is set to `true`, the UI-component which shows the notification is rendered.

```
Positioned(
  left: 10,
  bottom: 10,
  child: (notificationVisible)
    ? Container(
      padding: EdgeInsets.all(4),
      color: Colors.yellow,
      child: Text(
        notificationText,
        style: TextStyle(fontSize: 20),
      ),
    )
    : Container(),
),
```

Quellcode 5: Notification in AddressPage

### 18.2.5.2 Edit Odd / Even Streets

One requirement was that addresses from one street could be automatically added to two areas based on whether the house number is even or odd. This is because it is common for all addresses with even house numbers to be on one side of the street and those with odd house numbers on the other side. This way, it is easier to assign the street sides to different areas, so that "Sternsinger" participants don't have to cross the street so often.

To make this possible, the administrator has to select a street in the `MapComponent` (18.4.1.1), then a blue button beneath the `InputFields` appears.

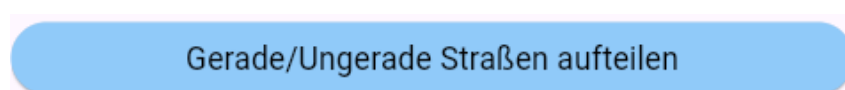


Abb. 5: Button to split street

After pressing this button, a dialog appears, where the administrator can select the areas for the addresses with even and odd house numbers.

Abb. 6: Dialog to split street

The "Speichern" button triggers the `AdminAddressProvider` and shows a `Notification` (18.2.5) if the operation was successful or not. After that, the dialog is closed.

```
int resultCode = await
  addressProvider.putAddressesOddEven(selectedAddresses.first.street.name,
    selectedAddresses.first.street.postalCode, selectedAreaEven,
    selectedAreaOdd);
if (resultCode == 200) {
  showNotification("Adressen gespeichert", () =>
    showAddressSavedNotification = true);
} else {
  showNotification("Fehler beim Speichern", () => showAddAddressNotification
    = true);
}
if (!context.mounted) return;
Navigator.of(context).pop();
```

Quellcode 6: onPressed save button in splitStreetDialog

### 18.2.6 Filter

With the Filter field, the administrator can filter the addresses displayed. It contains three dropdown menus to set the filter criteria, with one checkbox for each to toggle them. These filters can be combined as desired.

Abb. 7: Filter field in AddressPage

The filter is passed and applied to the `MapComponent` and the `DatabaseViewComponent`. The criteria and their enabled/disabled state are managed by the following variables in the `AddressPage` class.

```
bool specialFeatureFilter = false;
bool areaFilter = false;
bool streetFilter = false;
```



```
String selectedStreetFilter = "";
String selectedSpecialFeatureFilter = "";
String selectedAreaFilter = "";
```

Quellcode 7: Filter variables in AddressPage

This is an example of how a `FilterRow` is defined in the `AddressPage` class (18.6.2):

```
FilterRow(
    label: "Besonderheit:",
    tooltipMessage: "Besonderheitsfilter aktivieren/deaktivieren",
    filterValue: specialFeatureFilter,
    onFilterChanged: (bool? newValue) {
        setState(() => specialFeatureFilter = newValue ?? false);
    },
    selectedValue: selectedSpecialFeatureFilter,
    items: specialFeatureTextList,
    onDropDownChanged: (String? newValue) {
        setState(() => selectedSpecialFeatureFilter = newValue ?? "");
    },
),
```

Quellcode 8: FilterRow in AddressPage

### 18.3 ListEditPage

The `ListEditPage` is used to manage streets, special features and areas.

## **18.4 Components**

### **18.4.1 MapComponent**

#### **18.4.1.1 Edit multiple Addresses**

#### **18.4.1.2 Select Street**

#### **18.4.1.3 onClickNewAddress**

#### **18.4.1.4 Select Coordinates on Map**

### **18.4.2 DatabaseViewComponent**

### **18.4.3 PDFSaver**

### **18.4.4 AdminAddressProvider**

### **18.4.5 CustomHttpClient**

## **18.5 Models**

### **18.5.1 AreaWithBorder**

### **18.5.2 ScreenItem**

## **18.6 Widgets**

### **18.6.1 InputField**

### **18.6.2 FilterRow**

## **19 Final Thoughts**

### **19.1 Leon Edlinger**

### **19.2 Paul Gigler**

### **19.3 Andreas Weissl**

## **20 Meetings**

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

## 21 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
<b>Summe</b>		<b>187 h</b>

Table 1: Arbeitszeitznachweis

## **22 Source code directory**

Source Code directory, kein plan was des is

## 23 List of figures

1	Navigation in Admin-Panel . . . . .	18
2	AddressPage . . . . .	19
3	Dialog to confirm deletion . . . . .	20
4	Notification examples . . . . .	22
5	Button to split street . . . . .	23
6	Dialog to split street . . . . .	24
7	Filter field in AddressPage . . . . .	24

## 24 List of tables

1	Arbeitszeitznachweis . . . . .	28
---	--------------------------------	----

## 25 Bibliography

Dagne, L. (2019). Flutter for cross-platform app and sdk development.

*Flutter for Beginners*. (n.d.). [https://books.google.at/books?hl=de&lr=&id=pF6vDwAAQBAJ&oi=fnd&pg=PP1&dq=benefits+dart+language&ots=dZJWUGVs4x&sig=a196WqhXmQzuy23cmcKpEplqn\\_k&redir\\_esc=y#v=onepage&q=benefits%20dart%20language&f=false](https://books.google.at/books?hl=de&lr=&id=pF6vDwAAQBAJ&oi=fnd&pg=PP1&dq=benefits+dart+language&ots=dZJWUGVs4x&sig=a196WqhXmQzuy23cmcKpEplqn_k&redir_esc=y#v=onepage&q=benefits%20dart%20language&f=false)

flutter/README.md at master · flutter/flutter [[Online; accessed 23. Jan. 2025]]. (2025, January).

<https://github.com/flutter/flutter/blob/master/README.md>



## 26 Abbreviation

ADC	Analog Digital Converter
API	Application Programming Interface
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
DAC	Digital Analog Converter
DAVE	Digital Application Virtual Engineer
DSP	Digital Signal Processor
FPU	Floating Point Unit
FPV	First Person View, First Pilot View
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GUI	Graphical User Interface
HDMI	High Definition Multimedia Interface
I <sup>2</sup> C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IP	Internet Protocol
RPI	Raspberry Pi
SD	Secure Digital
SPI	Serial Peripheral Interface
UI	User Interface
USB	Universal Serial Bus
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver Transmitter
WLAN	Wireless Local Area Network
WPA	WiFi Protected Access
XML	Extensible Markup Language