

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	2
2	Technologies	3
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
3	Research Questions	9
3.1	Leon Edlinger	9
3.2	Paul Gigler	9
3.3	Andreas Weissl	9
4	Spring Framework	9
4.1	Spring Boot	9
4.2	Spring Data JPA	9
4.3	Lombok	9
4.4	Advantages	9
5	Area Borders	9
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
6	Structure of the Backend	10
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

7	Defining usability	11
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
8	Usability in context of maps	12
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
9	Adaptive algorithms and real-time data integration	14
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
10	Traditional Methods for Address Database Management	14
11	Adaptive Algorithms: Concepts and Applications	14
12	Real-Time Data Integration Frameworks	14
13	Implementation of the Backend	14
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
14	GraphHopper Setup	15
14.1	Why use GraphHopper?	15
14.2	Configuration	15
14.3	Local hosting	15
15	Working out the Wireframes	15
15.1	Map View	15
15.2	List View	15
15.3	Possible improvements for future versions	15
16	Functional implementation behind the application	16
16.1	Address-Provider	16
16.2	HTTP-Requests	16
16.3	Implementation of the Flutter Map Component	16
17	The app in use	17
17.1	Introducing new users	17
17.2	The app in operation	17
17.3	User Feedback	17
18	Implementation Admin Panel	18
18.1	Navigation	18
18.2	AddressPage	19
18.2.1	Add Address	20
18.2.2	Edit Address	20
18.2.3	Delete Address	20
18.2.4	Validation	21

18.2.5	Additional Functionalities	23
18.2.6	Edit multiple addresses	24
18.2.7	Edit all Addresses from a Street	25
18.2.8	Edit Odd / Even Streets	26
18.2.9	Filter	27
18.3	ListEditPage	28
18.3.1	QR-Code Visualization for Areas	28
18.3.2	QR-Code-PDF Download	29
18.4	Components	30
18.4.1	AdminMapComponent	30
18.4.2	DatabaseViewComponent	31
18.4.3	PDFSaver	31
18.4.4	AdminAddressProvider	32
18.5	Models	33
18.5.1	AreaWithBorder	33
18.5.2	ScreenItem	34
18.6	Widgets	34
18.6.1	InputField	34
18.6.2	FilterRow	35
19	Final Thoughts	36
19.1	Leon Edlinger	36
19.2	Paul Gigler	36
19.3	Andreas Weissl	36
20	Meetings	37
21	Working Hours	38
22	Source code directory	39
23	List of figures	40
24	List of tables	41
25	Bibliography	42
26	Abbreviation	43

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

The Admin Panel is a centralized administrative dashboard to efficiently manage all addresses within a single environment. It is used for planning upcoming 'Sternsinger' events, ensuring that all required addresses are covered and effectively distributed among participating groups. Additionally, it provides a way to designate areas containing addresses that a group needs to visit. This zoning feature ensures that each group is responsible for only the addresses within their designated area.

With this tool, administrators can perform CRUD (Create, Read, Update, Delete) operations on addresses, streets, and areas. These features make it easy to quickly address issues and make changes to the areas that participants need to visit. For example, when a new street is added to the neighborhood, the administrator can update the system to include it and assign it to the appropriate area. Similarly, if a group withdraws from the event, the administrator can reassign any unvisited addresses to other groups. This ensures that the data remains up-to-date and allows for quick reactions to special cases, helping with the planning and execution of 'Sternsinger' events.

This chapter will outline the implementation of the Admin Panel, detailing its components, functionalities, and widgets, and providing guidance on how to use them.

18.1 Navigation

To navigate between pages, a sidebar on the left is used, which can be toggled with a button in the top-left corner of the screen. It displays a list of all pages, allowing users to switch between them with a click.

The navigation is implemented in the file `AdminNavigation`. It contains a list of `ScreenItems` (see 18.5.2). These titles are displayed at the top of the screen above the corresponding page. To keep track of the currently selected page, an internal state (`indexState`) is used. Whenever a page is selected in the sidebar, the `indexState` is updated, and the corresponding page is displayed. This widget is the main component. It makes sure that all pages are properly displayed.

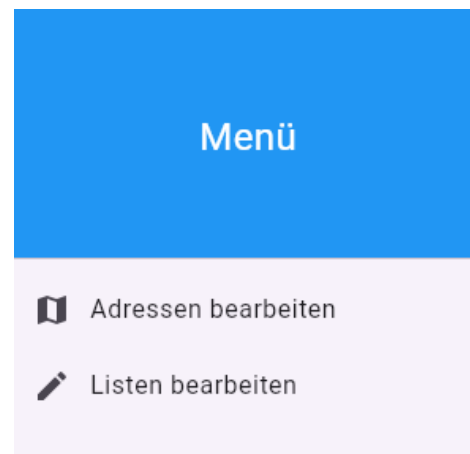


Abb. 1: Navigation in Admin-Panel

18.2 AddressPage

The first page, the `AddressPage`, displays an interface to create, update, and delete address data. It uses a form with various input fields (e.g., street, house number, coordinates, special features) and a map or database view to visualize and select addresses. The page is divided into two parts:

- On the right side, all addresses are shown either in the `AdminMapComponent` (18.4.1) or the `DatabaseViewComponent` (18.4.2). These two components display the same addresses but in different ways, giving the administrator the choice of how they want to view the addresses.
- On the left side of the page, there are `InputFields` (18.6.1), which are used to enter new information about a new address or edit an existing one.

Overlaying the `AdminMapComponent`, the following elements are displayed:

- A field to filter the addresses that are displayed.
- A button with a dropdown menu to select and edit a street.
- A switch to toggle between the `AdminMapComponent` and the `DatabaseViewComponent`.
- An information box in the bottom left corner to display `Notifications` (18.2.5.1) about the completed operations.
- A field in the bottom right corner to display the coordinates of the mouse pointer on the map.

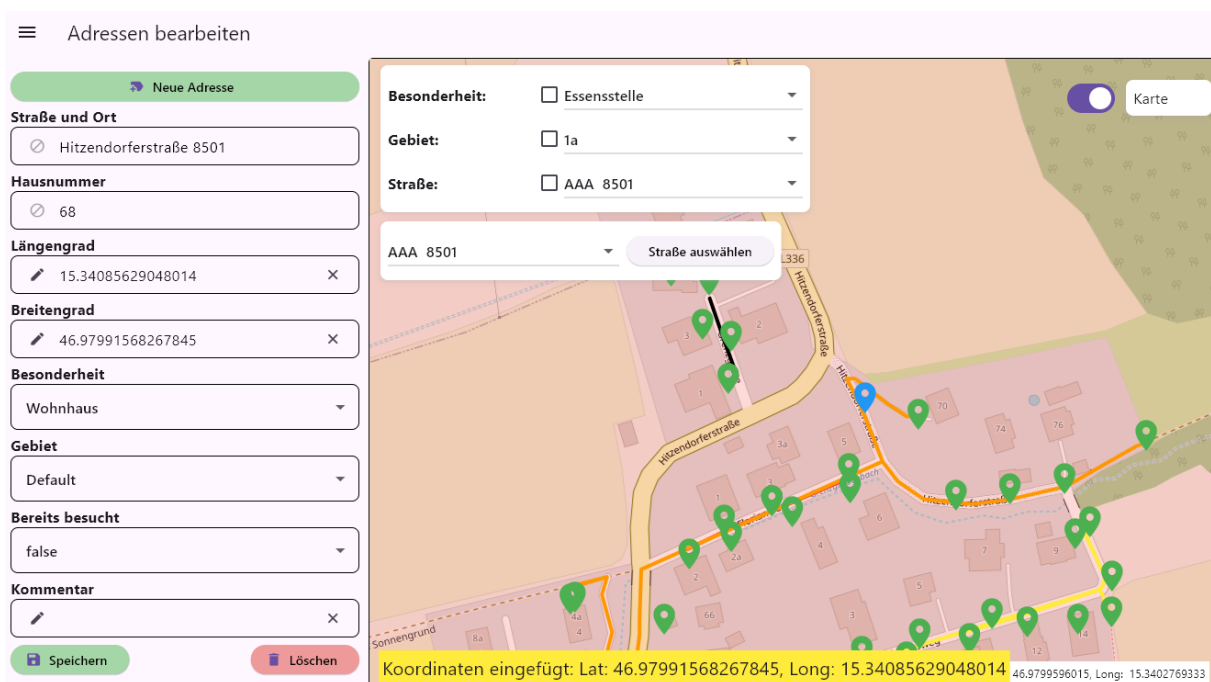


Abb. 2: AddressPage

18.2.1 Add Address

To add a new address, the "Neue Adresse" button is pressed. This triggers the `onClickNewAddress` method, which performs several key operations:

- All `InputFields` are cleared.
- The boolean variable `isNewAddress` is set to `true`, indicating that a new address is being created.

The cleared fields can then be filled with the new information. When the "Speichern" button is pressed, the `saveAddress` method is called. This method performs several validation checks which can be seen in 18.2.4. If the address is valid, the `AdminAddressProvider` is called to add it to the database. If the operation was successful, a `Notification` is displayed and the newly added address appears on the map.

The `onClickNewAddress` method is also called within the `AdminMapComponent` when the map is clicked, to easily add a new address by clicking on the map. This is done by passing the coordinates of the click to the `onClickNewAddress` method.

18.2.2 Edit Address

Existing addresses can be edited by selecting an address in either the `AdminMapComponent` or the `DatabaseViewComponent`. The selection fills the `InputFields` with the information of the selected address, and in this case, the boolean variable `isNewAddress` is set to `false` to indicate that an existing address is 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, and the `Notification` is shown.

18.2.3 Delete Address

After selecting an address, the admin can press the "Löschen" button to delete it. This triggers the `deleteAddress` method, which calls the `AdminAddressProvider` to delete the selected addresses and displays the `Notification`.

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

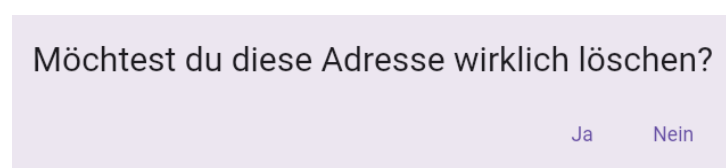


Abb. 3: Dialog to confirm deletion

18.2.4 Validation

Validation is the process of checking that data meets specific criteria before it is accepted and added to a system or database. This ensures the data is accurate, consistent, and conforms to required standards. Validation prevents errors and inconsistencies that could arise from invalid or incomplete data, maintaining the integrity and reliability of the system. It helps catch mistakes early, such as missing fields or incorrect data inputs. By ensuring only valid data is processed, validation plays a crucial role in preserving data quality. (Contributors to Wikimedia projects, 2025)

18.2.4.1 isDuplicateAddress

To determine whether an edited or newly added address already exists in the system, a comparison is made between the new address and all the existing addresses. This process ensures that any potential duplication is detected before the address is saved. The comparison takes place within the method `saveAddress`, which is responsible for handling the saving operation. If an address is found to be identical to an already existing one, the method returns `true`, indicating that the address is a duplicate. If no match is found, it returns `false`, confirming that the address is unique and can be safely added to the database.

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 InputField filled Validation

To ensure that all the required `InputFields` are properly filled, the `validateAddressFields` method is called within the `saveAddress` (18.2.1) method. The `validateAddressFields` method requires an `Address` object as its parameter.

To start the process, a new `Address` object is created, which is then passed to the `validateAddressFields` method. The method checks each field within the `Address` to ensure that all required information is present and valid. If a field is missing or incomplete, a notification is triggered to inform the user of the missing data, such as "Strasse fehlt" or "Koordinaten fehlen." If all fields are filled out correctly, the method returns `true`, signaling that the address can be saved. However, if any field is incomplete, the method returns `false`, preventing the address from being saved until all necessary information is provided.

18.2.4.3 InputField Coordinates Validation

The `InputField` validates **latitude** and **longitude** inputs to ensure their correctness. Validation is applied only when the `isCoordinateInput` parameter is set to `true`. If that is the case, 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`:

- A `FilteringTextInputFormatter` with a regular expression is used to restrict the input to digits, decimal points, and an optional minus sign at the beginning.

The image shows a red regular expression string: `r'^-?[0-9.]*`. It is centered and appears to be a code snippet.

Abb. 4: Regular expression for input validation

- A `TextInputFormatter` prevents multiple decimal points in a single number. If a user attempts to insert a second decimal point, the input is automatically rejected.
- The last `TextInputFormatter` ensures that the number of digits before the decimal point is limited to a maximum of 3. Additionally, the value before the decimal point must not exceed 180.

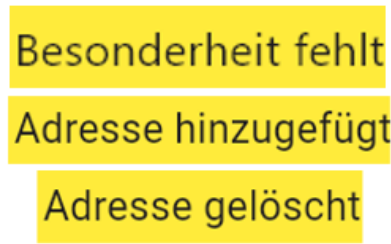
18.2.5 Additional Functionalities

This section outlines several functionalities that have been integrated to improve the overall usability and efficiency of the application. These enhancements are specifically designed to simplify tasks while ensuring a smooth and intuitive user experience.

18.2.5.1 Notification

To inform the administrator about the success or failure of an operation, a `Notification` is displayed at the bottom left, overlaying the `AdminMapComponent`. This `Notification` appears when:

- An address is added, edited, or deleted.
- Validation fails.
- Coordinates are selected on the `AdminMapComponent` (18.4.1.2).



Besonderheit fehlt
Adresse hinzugefügt
Adresse gelöscht

Abb. 5: Notification examples

To display this notification, the `showNotification` method is called. This method sets the `notificationVisible` variable to `true` and starts a `Timer` to reset it to `false` after three seconds, causing the notification to disappear shortly after. The method accepts a message as a parameter, which is stored in the `notificationText` variable.

```
void showNotification(String message) {
    setState(() {
        notificationText = message;
        notificationVisible = true;
    });
    Timer(Duration(seconds: 5), () {
        setState(() {
            notificationVisible = false;
        });
    });
}
```

Quellcode 2: 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 3: Notification in AddressPage

18.2.6 Edit multiple addresses

To make it easier to edit multiple addresses at once, the CTRL key on the keyboard is listened to. When the CTRL key is pressed, the `isCtrlPressed` variable is set to `true`. This variable is passed to the `DatabaseViewComponent` and the `AdminMapComponent` to inform them that multiple addresses are to be selected. The components can use the `markerSelected` function, passed as a parameter, to set the selected addresses in the `AddressPage`. The `saveAddress` method is then called to save the selected addresses.

To detect when the CTRL key is pressed or released, the predefined `RawKeyboardListener` widget is utilized. This widget listens for keyboard events and updates the state of the `isCtrlPressed` variable accordingly. When the CTRL key is pressed, the widget sets `isCtrlPressed` to `true`, indicating that the key is being held down. Conversely, when the CTRL key is released, `isCtrlPressed` is set to `false`, signaling that the key is no longer pressed.

To handle repeated key presses effectively, especially when the CTRL key is held down, a condition is implemented to avoid continuously setting `isCtrlPressed` to `true`. This is achieved by checking the `event.repeat` property, ensuring that the variable is only updated when `event.repeat == false`.

The `markerSelected` method also updates the `InputField` by listing the house numbers of the selected addresses in the controller. This ensures that the user is presented with a clear overview of all the selected addresses house numbers, making it easier to manage multiple selections.

```
controllers["houseNumber"]?.text = selectedAddresses
    .map((address) => address.houseNumber)
    .toList()
    .join(', ');
```

Quellcode 4: Listed house numbers for multiple selected addresses

The `InputField` then looks like this:

Hausnummer

Abb. 6: House numbers of multiple selected addresses

The selected addresses are saved in the `selectedAddresses` variable in the `AddressPage`. If multiple addresses are selected, certain `InputFields`, such as house numbers, coordinates, or comments, will be disabled because changing them for all selected addresses would not make sense. This is achieved by setting the `editable` parameter of these `InputFields` to `selectedAddresses.length <= 1`, ensuring that they are only editable when a single address is selected.

18.2.7 Edit all Addresses from a Street

All addresses from a street can be edited at once. This process is very similar to editing multiple addresses (18.2.6), but instead of individually selecting each address by clicking on it, the entire street is selected. Once a street is chosen, all addresses associated with that street are selected. This feature simplifies the process of managing multiple addresses within the same street. A street can be selected in two ways:

18.2.7.1 Select Street via AdminMapComponent

Every click on the `AdminMapComponent` is checked to see if it is near a street. Since there is no predefined method to check if a point is on a street, the `isPointNearPolyline` method was implemented. This method checks whether the click was on the street. More information about this method can be found in the `AdminMapComponent` section 18.4.1.

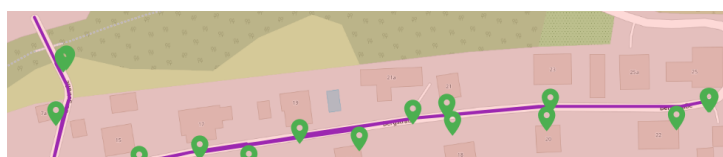


Abb. 7: Street example in AdminMapComponent

18.2.7.2 Select Street via Button

This was implemented because we encountered a problem where it was not possible to click on a street on some devices. To ensure that this feature is usable on all devices, a dropdown button to choose a street and a button to select it were implemented.



Abb. 8: Button to select a street

18.2.8 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 makes it 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 `AdminMapComponent` (18.4.1.1), then a blue button beneath the `InputFields` appears.

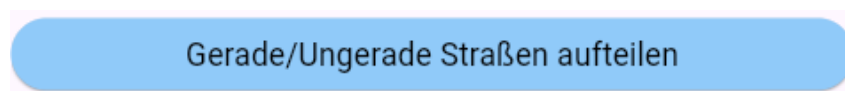


Abb. 9: Button to split street

After pressing this button, a dialog appears, where the administrator can select the areas for the addresses with even and odd house numbers. The "Speichern" button triggers the `AdminAddressProvider` and displays a `Notification` (18.2.5.1) indicating whether the operation was successful or not. Afterward, the dialog is closed.

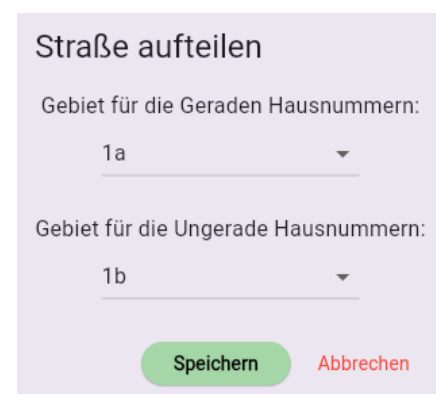


Abb. 10: Dialog to split street

18.2.9 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.

Besonderheit:	<input type="checkbox"/> Besuch nicht gewünscht	▼
Gebiet:	<input type="checkbox"/> 1a	▼
Straße:	<input checked="" type="checkbox"/> Amselgasse 8501	▼

Abb. 11: Filter in AddressPage

The filter is passed and applied to both the `AdminMapComponent` and the `DatabaseViewComponent`. This ensures that the data presented to the user is filtered according to the criteria they select. The criteria and their enabled/disabled state are managed by a series of variables within the `AddressPage` class. These variables control which filters are active and hold the selected filter values. The variables are defined as follows:

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

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

Quellcode 5: Filter variables in AddressPage

The filter is then applied to the addresses in the components. The filtering process is done step-by-step, starting with the area filter, followed by the special feature filter, and finally the street filter. If any of the filter conditions are met, the addresses are filtered accordingly. Only the `filteredAddresses` are displayed in the `AdminMapComponent` and the `DatabaseViewComponent`.

```
if (widget.isGebietFilter && currentGebietFilter != null) {
    filteredAddresses = addresses.where((address) => address.area.desc ==
        currentGebietFilter).toList();
}

if (widget.isBesonderheitFilter && currentBesonderheitFilter != null) {
    filteredAddresses = filteredAddresses.where((address) =>
        address.specialFeature.text == currentBesonderheitFilter).toList();
}

if (widget.isStreetFilter && currentStreetFilter != null) {
    String name = currentStreetFilter!.split(" ")[0];
    String number = currentStreetFilter!.split(" ")[1];
    filteredAddresses = filteredAddresses.where((address) => address.street.name
        == name && address.street.postalCode == number).toList();
}
```

Quellcode 6: Filtering addresses in AddressPage

18.3 ListEditPage

The `ListEditPage` is used to manage **streets**, **special features**, and **areas**. It allows the administrator to add, edit, and delete these entities. The page is divided into two parts. On the right side, all entities are displayed in a table and can be selected. On the left, there is a dropdown menu for selecting between the three options. The information of the selected item is shown in `InputFields`, where the information can be edited, saved, or deleted using the "Speichern" or "Löschen" button.

Abb. 12: ListEditPage

18.3.1 QR-Code Visualization for Areas

To make it easier for users of the user application to get information about their area, a QR-Code is generated for the selected area, which can be scanned to get all information. The QR-Code contains the name of the area. The currently selected area is saved in the `selectedItem` variable.

Abb. 13: QR-Code for area

To display the QR-Code, the `QrImageView` Widget is used.

```
QrImageView(
  data: selectedItem,
  size: 300,
  padding: const EdgeInsets.all(16.0),
  gapless: false,
),
```

Quellcode 7: QRCode Generation in ListEditPage

18.3.2 QR-Code-PDF Download

A PDF with QR codes for all areas can be downloaded, making it easier to distribute the information to users. The PDF is generated using the `savePDF` method of the `PDFSaver` class.

To generate the QR codes, the `generateQRCode` method in the `ListEditPage` class is used. This method takes a `String` as a parameter and converts it into a QR code using the predefined `QrPainter`, which assists in rendering the QR code ("QrPainter class - qr_flutter library - Dart API", 2025). It then generates an image of the QR code, converts it into PNG byte data, and transforms the byte data into a list of bytes (`Uint8List`) to be used in the PDF.

Then the `savePdf` method of the `PDFSaver` class creates a PDF document containing the QR codes for all areas. It uses the predefined `pdf` package (imported as `pw`) to manage the PDF document ("pdf | Dart package", 2025). The function starts by initializing a new PDF using the `pw.Document()` class and an empty list to store the generated QR codes. The method then iterates over the list of areas, generating a QR code for each description of all areas using `generateQRCode` and adding it to the list.

After that, a page is added to the PDF document using the `pw.MultiPage` widget. This page includes a title and the description of each area along with its corresponding QR code. After building the page content, the PDF data is saved by calling `pdf.save()`. Finally, the `PDFSaver.savePdf` method is called to save the PDF as "Gebiete.pdf".

Alle Gebiete:

1a

1b



Abb. 14: PDF with QR-Codes for all areas

18.4 Components

18.4.1 AdminMapComponent

With this component, the addresses can be displayed on a geographic map. The map used is OpenStreetMap, which is free to use for everyone. Even though the Admin Panel is not used commercially, OpenStreetMap's attribution rules are followed. The map shows the required copyright notice "© OpenStreetMap contributors" in the bottom-left corner. The word "OpenStreetMap" is clickable and links directly to <https://www.openstreetmap.org/copyright>, opening the official license page in a new tab. This meets OpenStreetMap's license requirements by clearly acknowledging the data source and linking to their terms. ("Urheberrecht und Lizenz", 2025)

© OpenStreetMap contributors

Abb. 15: OpenStreetMap attribution

18.4.1.1 Select Street

To select a street, the `isPointNearPolyline` method was implemented. This method checks if the click was on a street. It takes the coordinates of the click as a `LatLng` object and a street as a list of `LatLngs`. After each click on the map, a loop is triggered that calls the `isPointNearPolyline` method for every street, passing the click coordinates during each iteration.

The `isPointNearPolyline` method uses another loop, which iterates over all points of the street. Since the street is represented as a list of `LatLngs`, every two consecutive points are connected to form a line. During each iteration of the loop, a helper method `isNear` is called, which calculates the distance between the click and the currently iterated line segment.

```
bool isPointNearPolyline(LatLng point, List<LatLng> polylinePoints) {
    const double tolerance = 0.00005;
    for (int i = 0; i < polylinePoints.length - 1; i++) {
        if (_isNear(point, polylinePoints[i], polylinePoints[i + 1], tolerance)) {
            return true;
        }
    }
    return false;
}
```

Quellcode 8: isPointNearPolyline method

The `isNear` method takes the click point, the two points for the line and the tolerance. If the distance is smaller than the predefined tolerance, the method returns `true`, indicating that the click was on the street.

```
bool _isNear(LatLng p, LatLng a, LatLng b, double tolerance) {
    double dx = b.latitude - a.latitude;
    double dy = b.longitude - a.longitude;
```

```

double t = ((p.latitude - a.latitude) * dx + (p.longitude - a.longitude) *
    dy) / (dx * dx + dy * dy);
t = t.clamp(0, 1);
double cx = a.latitude + t * dx;
double cy = a.longitude + t * dy;
return (p.latitude - cx).abs() + (p.longitude - cy).abs() <= tolerance;
}

```

Quellcode 9: isNear method

18.4.1.2 Select Coordinates on Map

To select coordinates to the currently selected address, the method `onCoordinateSelected` is called via a callback from the `AdminMapComponent`. This method sets the coordinates of the selected address to the coordinates of the click. The `InputFields` for the coordinates are then filled with the new coordinates.

Because it wouldnt make any sense to select coordinates for multiple addresses at once, the length of the `selectedAddresses` list is checked. Only if it is smaller to 2, the coordinates can be selected.

18.4.2 DatabaseViewComponent

To display the addresses in a table, this component was implemented. It takes the `selectedAddresses`, the filter variables, and the `isCtrlPressed` variable from the `AddressPage` as parameters. The `isCtrlPressed` variable is used to track whether multiple addresses are being selected. The `selectedAddresses` are displayed in the table and can be selected by clicking on them.

Above the table, two fields are displayed: one showing the number of selected addresses and the other showing the total number of found addresses. The table is implemented using the `ListView` widget, a predefined widget in Flutter.

2072 Adressen gefunden					3 Adressen ausgewählt	
Straße	Hausnummer	Ort	Gebiet	Besonderheit	Bereits besucht	Kommentar
Ahorn gasse	4	8501	Default	Wohnhaus	false	
Ahorn gasse	5	8501	Default	Wohnhaus	false	
Ahorn gasse	6	8501	Default	Wohnhaus	false	
Ahorn gasse	8	8501	Default	Wohnhaus	false	
Ahorn gasse	10	8501	Default	Wohnhaus	false	
Ahorn gasse	12	8501	Default	Wohnhaus	false	
Ahorn gasse	14	8501	Default	Wohnhaus	false	

Abb. 16: DatabaseViewComponent

18.4.3 PDFSaver

The `PdfSaver` class provides a static method to save a PDF file from a byte array. The method `savePdf` takes a `Uint8List` of bytes and a `String` representing the file name as parameters. Depending on the

platform, it saves the PDF accordingly.

```
static Future<void> savePdf(Uint8List bytes, String fileName) async {
  if (kIsWeb) {
    final blob = html.Blob([bytes], 'application/pdf');
    final url = html.Url.createObjectUrlFromBlob(blob);
    final anchor = html.AnchorElement(href: url)
      ..target = 'blank'
      ..download = fileName
      ..click();
    html.Url.revokeObjectUrl(url);
  } else {
    await FileSaver.instance.saveFile(
      name: fileName,
      bytes: bytes,
      ext: 'pdf',
      mimeType:MimeType.pdf,
    );
  }
}
```

Quellcode 10: savePdf method in PDFSaver

18.4.4 AdminAddressProvider

This class serves as a bridge between the Admin Panel and the backend. It is responsible for all CRUD operations on addresses, streets, special features, and areas. It is used in the `AddressPage` and the `ListEditPage`. The `AdminAddressProvider` includes the `ChangeNotifier` mixin. A mixin is a way to reuse code across multiple classes, without using inheritance like in Java. (“Mixins”, 2025) The `ChangeNotifier` mixin is used to notify the UI when the data changes. (“ChangeNotifier class - foundation library - Dart API”, 2025) This is done by calling the `notifyListeners` method.

Here is a typical method in the `AdminAddressProvider` class. It contains these functionalities:

- `async` : enables non-blocking operations and ensures that the UI remains responsive while waiting for tasks like network requests to complete.
- `await http.get` : sends a GET request to the server to fetch all streets and waits for the response.
- `jsonDecode` : processes the JSON response from the server.
- `utf8.decode` : transforms the UTF-8 encoded response body into readable text.
- `map` : converts the decoded JSON to a list of `Street` objects.
- `sort` : sorts the list alphabetically.

- `notifyListeners` : notifies the listeners that the data has changed.
- `catch` : catches any errors that occur during the operation.

```
Future<void> fetchStreets() async {
    try {
        final response = await
            http.get(Uri.parse('$serverURL/$baseURL/admin/getAllStreets'));
        if (response.statusCode == 200) {
            final List<dynamic> decodedJSON =
                jsonDecode(utf8.decode(response.bodyBytes));
            streets = decodedJSON.map((json) => Street.fromJson(json)).toList();
            streets.sort((a, b) => a.name.compareTo(b.name));
            notifyListeners();
        } else {
            throw Exception('Failed to load streets');
        }
    } catch (e) {
        print('Error fetching streets: $e');
    }
}
```

Quellcode 11: typical method in AdminAddressProvider

18.5 Models

Models represent and manage data structures within the Admin Panel. They are designed to encapsulate data and provide a structured approach for interacting with it. Various models have been developed to ensure maintainability and efficiency.

18.5.1 AreaWithBorder

To associate an area with its border coordinates, the `AreaWithBorder` model was introduced. It stores all border points as a list of `LatLng` objects, which are then used in the `AdminMapComponent` to draw the boundary. This visualization helps to clearly distinguish which addresses belong to which area.

```
class AreaWithBorder {
    final Area area;
    final List<LatLng> border;

    AreaWithBorder(this.area, this.border);
}
```

Quellcode 12: AreaWithBorder class

18.5.2 ScreenItem

The `ScreenItem` was created to store data related to a menu option in the `AdminNavigation`. It is used to save the title, associated screen, and icon for each navigation item.

```
class ScreenItem {
    final String title;
    final Widget screen;
    final Icon icon;

    ScreenItem({required this.title, required this.screen, required
        this.icon});
}
```

Quellcode 13: ScreenItem class

18.6 Widgets

Widgets are used to build and reuse UI components in Flutter. They help maintain a consistent look and feel across the entire interface, ensuring a seamless user experience. Several custom widgets were implemented for the Admin Panel.

18.6.1 InputField

This widget defines a customizable input field. The `InputField` requires a label and a `TextEditingController` to manage the input. Optional parameters include a boolean `editable`, a list of `Strings` called `dropDownOptions`, and a boolean `isCoordinateInput` to specify whether the input should be numeric.

To indicate whether the `InputField` is editable, icons are displayed on the left side of the field. If the field is not editable, a blocked symbol appears. If it is editable, a pencil icon is shown. When the `InputField` is editable and not a dropdown, a cross icon appears on the right side. Pressing this icon clears the content.

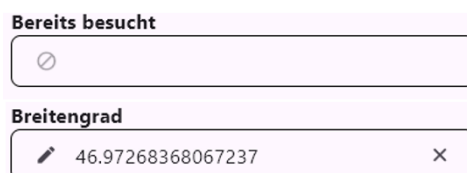


Abb. 17: InputField examples

If the `dropDownOptions` parameter is provided with a non-empty list of `Strings`, the input field is rendered as a dropdown menu. The selected value is displayed, and users can choose from the available options. The selected dropdown value is automatically synchronized with the `TextEditingController`.

To ensure valid inputs, the `InputField` uses an `inputFormatter` for validation. This formatter is applied only when the `isNumberInput` parameter is set to true. For more details about the validation, refer to 18.2.4.3.

18.6.2 FilterRow

This widget defines a reusable filter row. The `FilterRow` combines a label, a toggleable checkbox, and a dropdown menu to enable dynamic data filtering. It requires several parameters, including callbacks. A callback is a method passed as an argument to another function and executed after a specific event. These callbacks are used to handle activating and deactivating the filter, as well as changes to the filter's state and the selected dropdown value. The following parameters are used:

- `label` : Identification string for the filter.
- `tooltipMessage` : Message displayed when hovering over the checkbox.
- `filterValue` : Boolean value tracking the filter's active state.
- `items` : List of strings for the dropdown options.
- `selectedValue` : Manages the selected dropdown value.
- `onFilterChanged` : Callback for updating the filter state.
- `onDropdownChanged` : Callback for updating the selected dropdown value.

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 14: FilterRow in AddressPage

Besonderheit: ☐ Besuch nicht gewünscht ▼

Abb. 18: 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
Summe		187 h

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	Regular expression for input validation	22
5	Notification examples	23
6	House numbers of multiple selected addresses	25
7	Street example in AdminMapComponent	25
8	Button to select a street	26
9	Button to split street	26
10	Dialog to split street	26
11	Filter in AddressPage	27
12	ListEditPage	28
13	QR-Code for area	28
14	PDF with QR-Codes for all areas	29
15	OpenStreetMap attribution	30
16	DatabaseViewComponent	31
17	InputField examples	34
18	FilterRow	35

24 List of tables

1	Arbeitszeitznachweis	38
---	--------------------------------	----

25 Bibliography

- ChangeNotifier class - foundation library - Dart API [[Online; accessed 25. Feb. 2025]]. (2025, February).
<https://api.flutter.dev/flutter/foundation/ChangeNotifier-class.html>
- Contributors to Wikimedia projects. (2025, February). Data validation - Wikipedia [[Online; accessed 24. Feb. 2025]]. https://en.wikipedia.org/w/index.php?title=Data_validation&oldid=1276518123
- 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=a196WqhXmQzuy23cmckKpEplqn_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>
- Mixins [[Online; accessed 25. Feb. 2025]]. (2025, February). <https://dart.dev/language/mixins>
- pdf | Dart package [[Online; accessed 2. Mar. 2025]]. (2025, March). <https://pub.dev/packages/pdf>
- QrPainter class - qr_flutter library - Dart API [[Online; accessed 2. Mar. 2025]]. (2025, March).
https://pub.dev/documentation/qr_flutter/latest/qr_flutter/QrPainter-class.html
- Urheberrecht und Lizenz [[Online; accessed 27. Feb. 2025]]. (2025, February).
<https://www.openstreetmap.org/copyright>

26 Abbreviation

ADC	Analog Digital Converter
API	Application Programming Interface
BLE	Bluetooth Low Energy
CRUD	Create Read Update Delete
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 ² C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IP	Internet Protocol
PDF	Portable Document Format
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