htblakaindorf
Leistung mit Menschlichkeit

informatik

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

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

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)


Flutter

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 duis 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 duis 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 duis 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 duis 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 duis 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 duis 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 to plan for future "Sternsinger" events, ensuring that every address that needs to be visited is covered and that all addresses are efficiently distributed among participating groups. Additionally, it enables the assignment of areas containing addresses a group needs to visit. This zoning feature ensures that each group is responsible for visiting only the addresses within their assigned area.

With the 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, if a new street is added to the neighborhood, the administrator can update the system to include this street and assign it to the appropriate area. Similarly, if a group drops out, the administrator can quickly reassign the addresses that have not yet been visited 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 and describe the different components, functionalities, and widgets of this tool. Additionally, it will provide information on how to use it.

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 shows 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 pages with titles. 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.

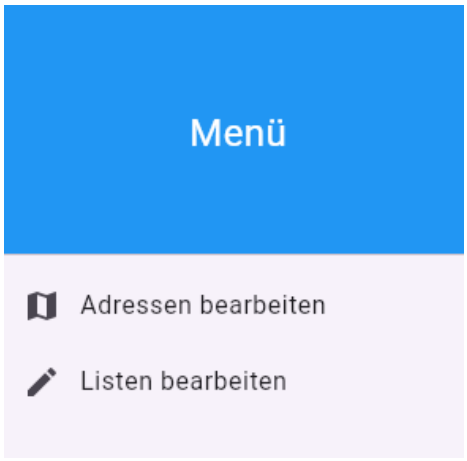

Abb. 1: Navigation in Admin-Panel

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, to give the administrator the choice of how they want to view the addresses.
- On the left side of the page are `InputFields` (18.6.1), which are used to enter new information about a new address or edit an existing one.

Overlaying the `AdminMapComponent`, there are:

- A field to filter the addresses 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.

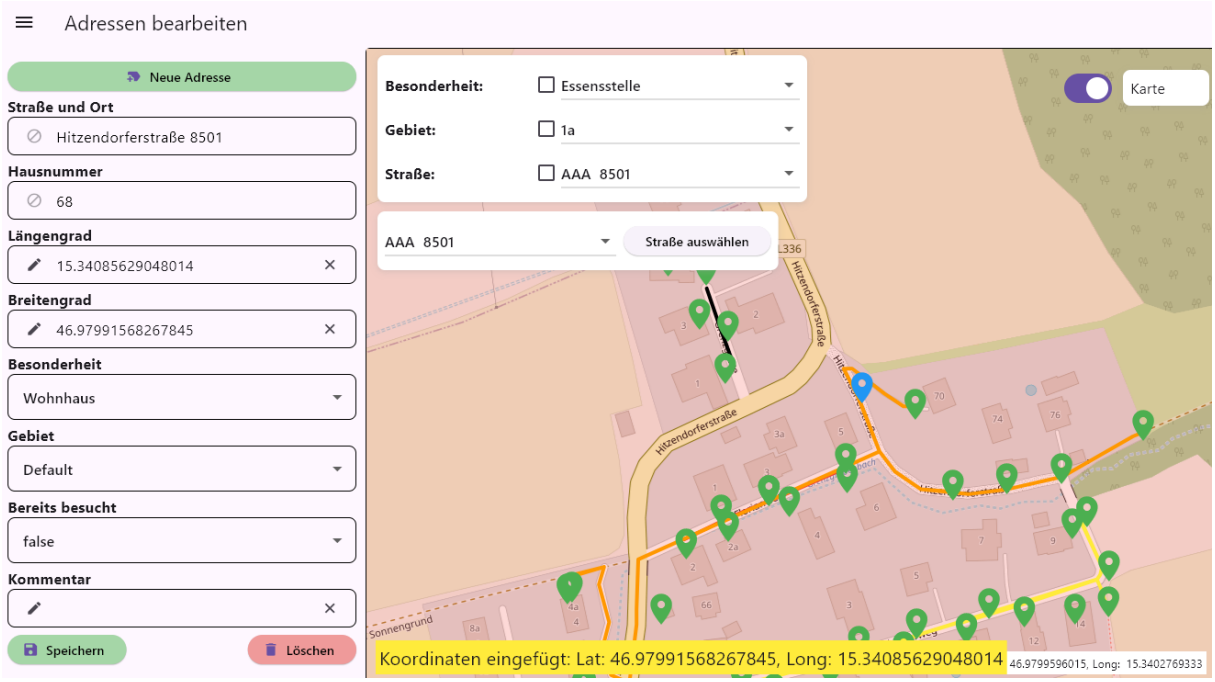

Abb. 2: AddressPage

18.2.1 Add Address

To add a new address, the button "Neue Adresse" is pressed. A single click on the map does the same thing, however it automatically fills in the coordinates of the location where the mouse was clicked. 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.

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 a `AlertDialog` to confirm the action and prevent accidental deletions.

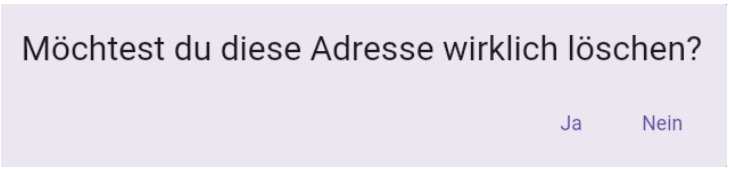

Abb. 3: Dialog to confirm deletion

18.2.4 Validation

Validation is the process of checking that data meets specific criteria before it is accepted and added. It is crucial to ensure that the data entered is correct, consistent, and meets the required standards to maintain data integrity and reliability. (Contributors to Wikimedia projects, 2025)

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 InputField filled Validation

To make sure that all `InputFields` are filled, the `validateAddressFields` is called in the `saveAddress` (18.2.1) 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 Coordinates 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 `FilteringTextInputFormatter` allows only numbers and dots with a regular expression.
- The `TextInputFormatter` checks if the input contains more than one dot and checks 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) {
            String text = newValue.text;
            if (text.split('.').length - 1 > 1 || (text.isNotEmpty &&
                text.split('.')[0].length > 3)) {
                return oldValue;
            }
            return newValue;
        }),
      ]
    : null,
```

Quellcode 3: InputFormatter in Inputfield

18.2.5 Additional Functionalities

This section highlights various functionalities implemented to improve the overall usability of the application. These additions are designed to support administrative tasks and ensure a seamless user experience.

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, overlaying the `AdminMapComponent`. This notification appears when:

- An address is added, edited, or deleted.
- Validation fails.
- Coordinates are selected on the `AdminMapComponent` (18.4.1.3).

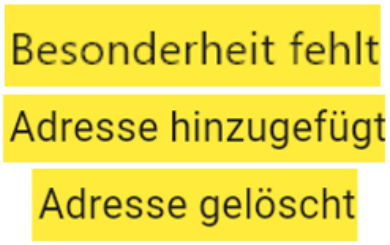

Abb. 4: Notification examples

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: 5), () {
    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

18.2.6 Edit multiple addresses

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

To listen to the CTRL-Button, the predefined `RawKeyboardListener` is used. This widget sets the `isCtrlPressed` variable to `true` when the CTRL-Button is pressed and to `false` when it is released. to make sure that repeadetly pressing the CTRL-Button, which happens if the button is pressed and held, does not interfere with the `isCtrlPressed` variable, the condition `event.repeat == false` is used.

```
    RawKeyboardListener(
        autofocus: true,
        focusNode: FocusNode(),
          onKey: (event) {
          if ((event.isKeyPressed(LogicalKeyboardKey.controlLeft) ||
               event.isKeyPressed(LogicalKeyboardKey.control) ||
               event.isKeyPressed(LogicalKeyboardKey.controlRight)) &&
                   event.repeat == false) {
              setState(() {
                isCtrlPressed = true;
```

```
            });
        } else if (event is RawKeyUpEvent && event.logicalKey ==
            LogicalKeyboardKey.controlLeft) {
            setState(() {
              isCtrlPressed = false;
            });
        }
      },
    ),
```

Quellcode 6: RawKeyboardListener in AddressPage

The `markerSelected` method also updates the `InputField` for the house number by listing the house numbers of the selected addresses in the controller.

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

Quellcode 7: Listed house numbers for multiple selected addresses

The `InputField` looks like this:

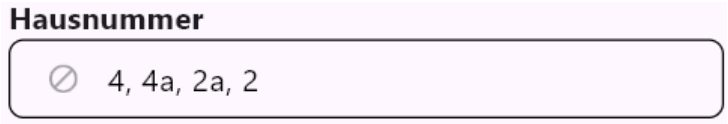

Abb. 5: House numbers of multiple selected addresses

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 can be achieved by setting the `editable` parameter of these `InputFields` to `selectedAddresses.length <= 1`, so 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 is almost the same as editing multiple addresses (18.2.6), but instead of selecting the addresses by clicking on them, a street is selected. When a street is selected, all its addresses are selected. A street can be selected in two ways:

18.2.7.1 Select Street via AdminMapComponent

Every click on the `AdminMapComponent` is checked if it is near a street. Because there is no predefined method to check if a point is on a street, the `isPointNearPolyline` method was implemented. If the

point is near a street, the method returns `true` otherwise `false`. More information about this method can be found in the `AdminMapComponent` section 18.4.1.

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 when the Admin Panel was deployed. This problem only occured on some devices. To ensure that the Admin Panel is usable on all devices, a dropdown button to choose a street and a button to select it were implemented.

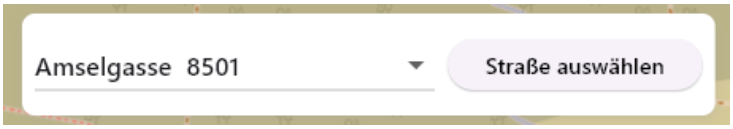

Abb. 6: Button to select a street

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

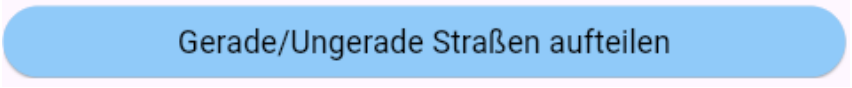

Abb. 7: Button to split street

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

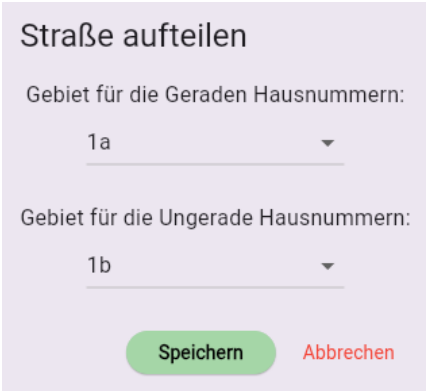

Abb. 8: Dialog to split street

The "Speichern" button triggers the `AdminAddressProvider` and shows a `Notification` (18.2.5.1) 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 8: onPressed save button in splitStreetDialog

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.

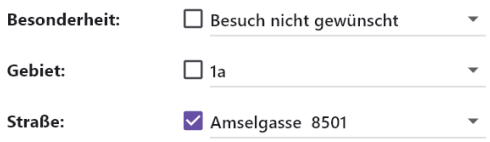

Abb. 9: Filter field in AddressPage

The filter is passed and applied to the `AdminMapComponent` 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 9: 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 10: FilterRow 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` on the this side, where the information can be edited, saved or deleted using the "Speichern" or the "Löschen" button.

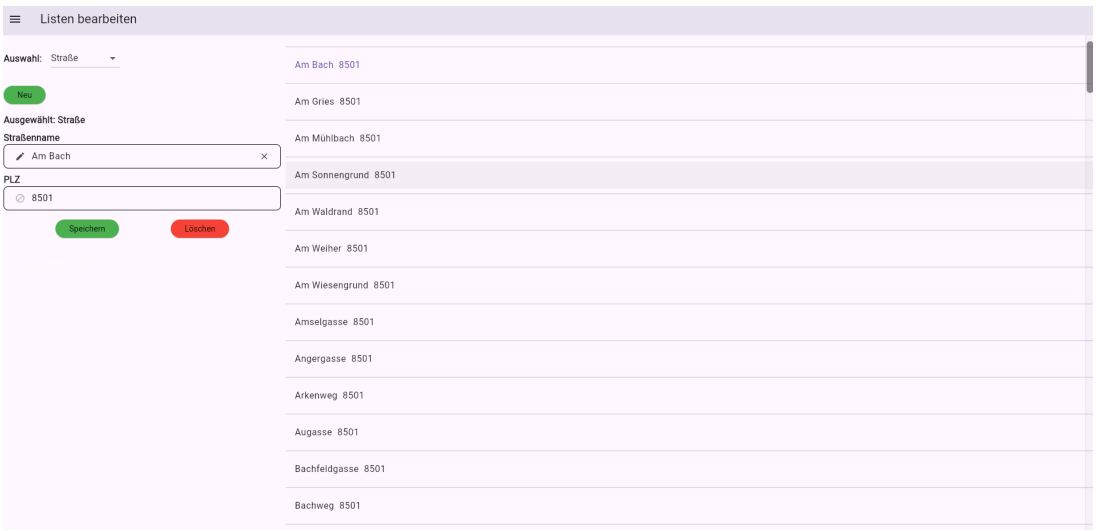

Abb. 10: ListEditPage

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 every 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.

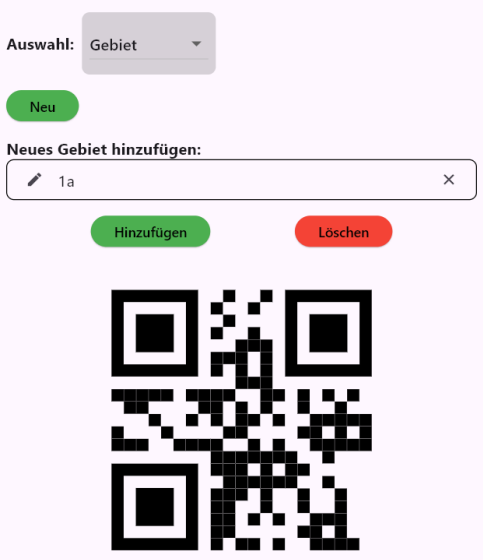

Abb. 11: QR-Code for area

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

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

Quellcode 11: QrCode Generation in ListEditPage

18.3.2 QR-Code-PDF Download

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

18.4 Components

18.4.1 AdminMapComponent

18.4.1.1 Select Street

18.4.1.2 onClickNewAddress

18.4.1.3 Select Coordinates on Map

18.4.2 DatabaseViewComponent

To display the addresses in a table, this component is used. As parameters it takes the `selectedAddresses`, the filter variables aswell as the `isCtrlPressed` variable from the `AddressPage`. The

`selectedAddresses` are displayed in the table and can be selected by clicking on them.

On the top are two fields

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 file 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 12: 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 mulitple 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 13: typical method in 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
Summe		**187 h**

Table 1: Arbeitszeitnachweis

22 Source code directory

Source Code directory, kein plan was des is

23 List of figures

24 List of tables

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=a196WqhXmQzuy23cmcKpEpIqn_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

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^2C	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