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.

[]


### 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.

[25] [Dag19]


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

This chapter concentrates on exploring the Spring ecosystem and covers core components like Spring Framework, Spring Boot and Spring Data JPA. It highlights the advantages of this ecosystem in simplifying Java development and improving the efficiency.

## 4.1 Spring Boot

Spring Boot is a tool which makes developing web applications and microservices with the Java Spring Framework faster and easier. As powerful as the Spring Framework on its own is, it still requires much time and knowledge to configure, set-up and deploy Spring apps. Spring Boot tries to mitigate this effort with three features

**Autoconfigure:**

- Autoconfigure initializes Spring apps with a preset of dependencies so that the developer does not have to configure those manually. Spring Boot comes with this feature to automatically configure the Spring Framework and third-party packages based on the project requirements.
- Even though the developer can override the default configuration after the initialization, the initial setup makes the development process faster and more efficient.
- Meanwhile the autoconfiguration also reduces the possibility of many human errors which can occur during the configuration of an Java application.

**Opinionated Approach:**

- When adding and configuring startup dependencies, Spring Boot takes a subjective approach, customizing them to the requirements of the project. The right packages and default values are chosen automatically by Spring Boot, eliminating the need for human setup and decision-making for each configuration.

- During initialization, project requirements can be specified, allowing selection from a vast collection of starter dependencies, known as "Spring Starters," which cover most common use cases. For example, the "Spring Web" dependency simplifies the development of Spring-based web applications by including all necessary dependencies with minimal configuration. Likewise, "Spring Security" provides built-in authentication and access control features.
- More than 50 official Spring Starters are offered by Spring Boot, and there are numerous other third-party starters accessible.

**Stand-Alone Application:**

- With the help of Spring Boot, applications can be developed that function without the need for an external web server. This is accomplished by immediately integrating a web server, such as Tomcat, with the application as it is initializing. As a result, the application can be launched with the appropriate command on any platform.

**Setting up a Spring Boot app:**

A Spring Boot project can be initialized quickly using the **Spring Initializr**. For a new Spring Boot-based project, the **Spring Initializr** can be opened, where the project details are filled in and a packaged project is downloaded as a.zip file. While initializing, a number of factors have to be chosen that determine the project structure, such as the programming language, the version of Spring Boot, and any other dependencies required to support development.

Depending on the IDE where the development is being performed, a Spring Boot project can be directly developed inside the IDE itself. Even this method supports the same amount of configuration to simplify the process of development. Not all IDEs are capable of creating a Spring Boot project out of the box, and sometimes a separate plugin needs to be installed.

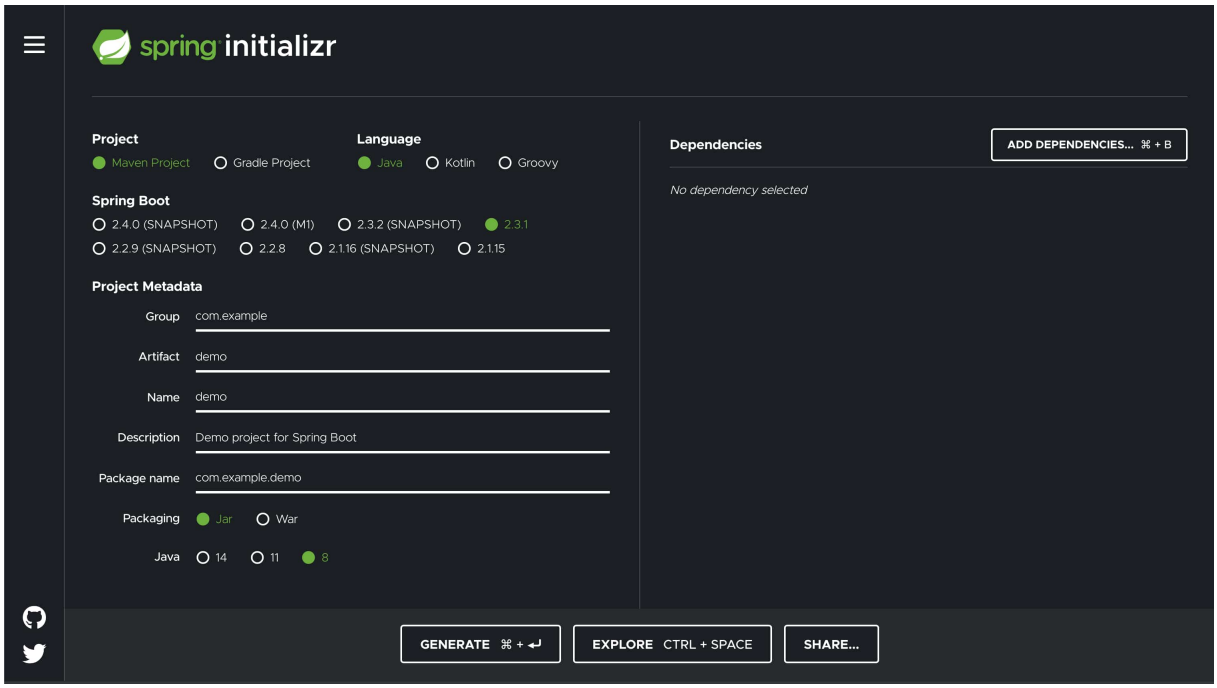


Abb. 1: GUI of the Spring Initializr

**Difference between Spring Boot and Spring Framework:**
Spring Boot's greatest strengths over the standard Spring Framework are its simplicity and quicker development. Theoretically, this is at the expense of the higher degree of flexibility that direct Spring Framework usage would provide. Practically, the compromis is more than worth it, as the Spring Framework's annotation system can still be utilized to inject other dependencies effectively. Besides, all Spring Framework functionalities like easy event handling and native security continue to be accessible.

## 4.2 Spring Data JPA

Spring Data JPA stands for Java Persistence API and provides a specification for persisting, reading and managing data from the object in the program, which are called entities, to project's tables in the database. JPA specifies a set of rules for developing interfaces that follow specific standards. As a result, JPA is just some guidelines to implement ORM. ORM is the process of persisting an object in java directly into a database table.
The goal of Spring Data JPA is to create classes called "Repositories" which significantly reduce the amount of unnecessary code to access and manipulate data from the database of a project.

**Entities:**
As already mentioned, the object from which Spring Data JPA manages the data are called entities. Every table of a database is an entity with each attribute being a column in the respective table. The `@Entity` annotation can be used to define the entity, guaranteeing that the class is understood as a component of the database structure and handled appropriately. However, this annotation is not the only one which is supported by JPA. By using annotations such as `@Id` or `@GeneratedValue`, different custom features of the entity, and therefore the database table, can be defined. Those annotations are a way to make the development of the project much faster and even more efficient.

**Data access object layer:**
Although there are many annotations that are usable for an entity, there is one annotation which is a core feature in JPA. The `@Repository` annotation is a marker for a class that fulfills the role of a repository which is also know as a Data Access Object. However, something else need to be done when using the `@Repository` annotation. When this annotation is used, the repository class must extend the `JpaRepository` class, which provides many built-in methods for managing, manipulating, sorting, and filtering data in the respective database table. If a required operation is not available by default, custom queries can also be defined within the repository class.

**Service Layer:**
Classes that use the `@Service` annotation are used with classes that provide some data manipulation methods such as a repository-class. By implementing such a class, the project structure is better understandable and more clear to other developers which may work on this project later on.

**Controller Layer:**

This Layer provides the applications with the routes with which the data can get manipulated through a GUI. By using the `@RestController` annotation, an actual controller class can be defined. The controller uses the service classes from the service layer to get the data or manipulate it in any way. It also specifies the routes which the code can then access in any form of user interface so that the user can actually use or see the data.

**Mappings:**

There are numerous annotations that define routes along with all their features and how they can be accessed. The `@GetMapping` gives the user data in any form. This can be all the data from a table or a specific entry in a table based on any filter. The `@PostMapping` is used if a user would want to create a new entry in a specific table.

For example, in a company's management system, the `@PostMapping` annotation can be used to build a route for adding a new employee to the database. If the employee's address or surname were to change, the `@PutMapping` annotation would be used to update the stored data for that employee. On the other side, if the employee were to resign, the `@DeleteMapping` annotation would be used to enable the manager to remove the employee from the company's database.

These routes can then be accessed through a GUI, allowing the user to manage, update, or delete data as needed.

## 4.3 Lombok

In the modern days of Java development, one big challenge developers face is writing boilerplate code which are code segments that repeat itself over and over again and get used often in a project. This is especially the case in frameworks like Spring Boot where we use classes like the service layer that involve a significant amount of repetitive code. "Project Lombok" is a Java library that has the aim to reduce this boilerplate code by automatically generating the code for commonly used patterns. By integrating Lombok in your Spring Boot project, you can not only simplify your code but make it easier to read maintain and write. Lombok works great with the whole Spring framework. If you would want to use a project with Spring Data JPA, you would not get that far without using any annotations provided by Project Lombok. To flag an entity class for Spring Data JPA, you need to use the `@Entity` annotation. This annotation comes from the Lombok library and adds a couple of features to this class so that JPA recognizes it as an entity for the database.

## 4.4 Advantages

The Spring Framework has a range of advantages that make it a popular choice among developers. These benefits shorten the development process and help in designing scalable and maintainable apps.

**Reduced Boilerplate Code:**

A huge factor of the Spring framework is its abilty to reduce repetitive code segments. Mainly through the Lombok library, the Spring framework gives the developer many ways to exchange repetitive code with annotations.

**Enhanced Testability:**

With Spring's huge library of dependencies you can not only get dependencies which make the coding easier but the testing too. Some dependencies gave us mock data to test the backend endpoints easier.

**Flexibility:**

The same thing that helped us testing, gave us and other projects the flexibility with which you can for example expand or change certain things in your project. This was extremely helpful for us because a certain requirement changed while we were developing the backend.

**Consistency:**

Spring provides consistent programming and configurations models across many different types of applications. Whether you would want to develop a web application or a microservice, Spring offers a unified approach which improves developer productivity.

**Improved Productivity:**

Tools like Spring Boot, which are part of the Spring ecosystem, significantly enhance developer productivity by providing better approaches to different problems and embedded servers.

# 5 Structure of the Backend

A well-structured backend is essential for building a scalable and maintainable application. This chapter explores the different layer of our backend. Each layer has its own purpose in handling client requests, processing logic in the code or managing database operations. Furthermore, this chapter provides an overview of the applied design principles we used in our backend.

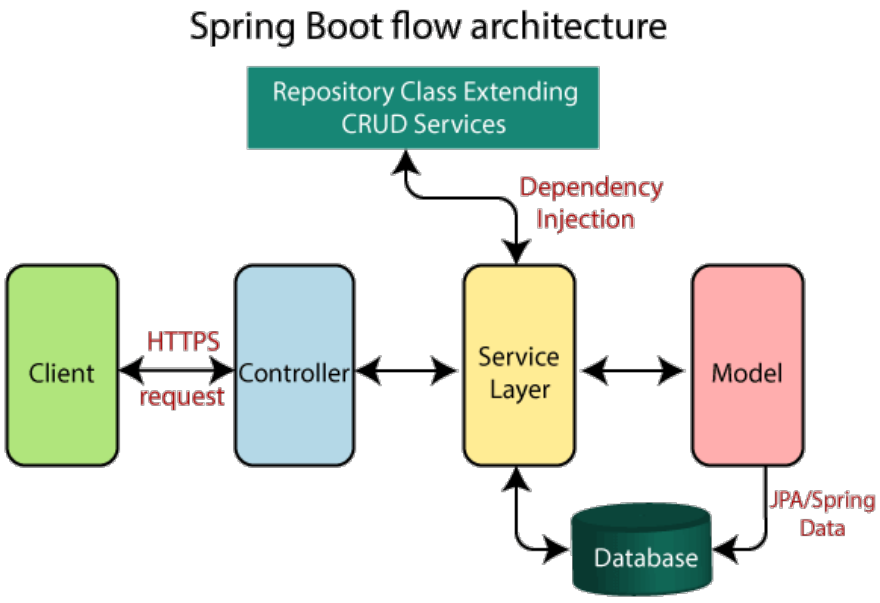


Abb. 2: Layers of Spring Boot

## 5.1 Controller Layer

The controller layer of a Spring Boot application is a significant layer that takes care of incoming HTTP requests and decides the appropriate response. It serves as the interface between the client and the backend which receives requests, hands over the tasks to the service layer and provides the responses accordingly. It ensures that the data which is used by the logic in the application is properly and effectively processed.
In a standard configuration, controllers handle a collection of HTTP methods. Amongst those HTTP methods are GET, POST, PUT and DELETE. Each methods has a certain function or action that are then performed by the application. Furthermore, every type of method corresponds to different operations on the data, such as retrieving, creating, updating or deleting information. Each controller belongs to a unique endpoint which serves as the interface for interacting with the application. The controller then uses service methods to execute the logic and return and appropriate response.
To ensure effective management of routing, Spring Boot uses annotations like `@GetMapping`, `@PostMapping`, `@PutMapping` and `@DeleteMapping`. These annotations identify the mapping between some of the HTTP methods and their respective handler methods in the controller. For example, the `@GetMapping` annotation is used for reading data, whereas the `@PostMapping` is applied for adding new data. The `@PutMapping` is utilized for updating existing data and the `@DeleteMapping` for deleting data.
One of the characteristics of the controller layer is the way it communicates with DTOs and entity classes. While the controller gets HTTP requests, it typically utilizes DTOs to enable data transfer between the client and the backend in a secure and effective manner.

## 5.2 Service Layer

The service layer plays a significant role in the backend organization to process the logic and serve as a buffer layer between the repository and controller layers. It makes the application modular, maintainable and scalable since it decouples request handling and data access. This seperation is necessary to provide flexibility in changes and additions to the application without impacting other parts of the project.
One of the primary advantages of having a dedicated service layer is that it can be reused. Rather than duplicating the logic all over the system, controllers and other code sinppets can use service methods when they need them. Not only does thi save code duplication, but it also simplifies future changes, since logic changes can be implemented in a single location without having an impact on the application overall.
Another significant aspect of this layer is transaction management. When an operation has more than one step and consists of a sequence of action, a guarantee that either all or none of those steps are executed is an absolute necessity. By defining such transactionals boundaries at the service level, the application ensures data consistency and prohibits incomplete operations from triggering undesired actions.
Wrapping complex logic within the service layer keeps the backend organized and clean. Rather than adding several conditions and operations directly into controllers or repositories, the service layer offers and organized area for processing the data.

## 5.3 Repository Layer

The repository layer is in charge of managing interactions with the database by exposing efficient methods for storing, reading, updating and deleting data. By using this layer, the backend is kept modular which results in the application being more maintainable and scalable.
Spring Data JPA comes with numerous in-built methods supporting most of the common database operations. By utilizing the repository layer, operations like saving new data, getting all existing data sets, updating data and deleting data are automatically available. This cust down a lot of unnecessary code required for database operations. Furthermore, it also accelerates the development process by providing many of methods.
Although built-in repository methods cover most scenarios, there are time when more complicated queries are required. To handle such cases, Spring Data JPA supports defining custom queries through annotations. This allows developers to get certain data in an efficient way without performing unnecessary database operations.

## 5.4 Persistence Layer (Entity Classes)

The Spring Boot application's persistence layer handles database communication, entity classes and how different entities are related along with ensuring data consistency. With JPA in Spring Boot, developers can create a well-structured and optimized data model which enhances the performance and scalability of the application.
Entity classes are used to represent database tables and need to be defined with an annotation to be scanned by Spring Boot JPA. If the table name is different from the class name, it can be annotated explicitly. In addition, unique identifiers need to be defiend to ensure that every entity is unique.
Cascading operations automatically persistence and deletion but must be dealt with cautiously in order to not lose data. Validation maintains the integrity of data by placing contraints like mandatory fields and length limitations.
Auditing enables tracking when and by whom entities are modified which provides more accountability. Restricting field values to predefined choices guarantees information consistency and using DTOs optimize queries by retrieving only neccessary fields.

## 5.5 Applied Design Principles (DTOs)

During the development of a Spring Boot application, having an efficient data communication between different layers is crucial. And exactly this is where DTOs are needed.
DTOs are objects that carry data between layers in an application. They are used to encapsulate and transport data mostly between a server and a client.

**Why Use DTOs?**

DTOs are crucial for a backend architecture due to their abilities to enhance the security, maintainability, as well as the performance of the data. Their ability to hide data is among their major strengths because they support exposing only the necessary information to clients wihtout showing sensitive data.
From a performance-perspective, DTOs maximize network efficiency by only sending the data that is required for the current action performed by the backend. This advantage is particularly valuable in systems where bandwidth and response time are essential.

Lastly, DTOs are adaptable which allows developers to customize responses to specific use cases. They can support data combination from different entities or the exclusion of unnecessary fields which then results in more efficient responses and easier processing.
By utilizing DTOs as they are intended, applications can achieve better modulatity, security and overall performance.

**Example for Using DTOs**

A common scenatio where DTOs are useful is when managing user information in an application.
Imagine a system that stores user details such as first name, last name, email and password. If the application exposes the user data to external clients, it could lead to security risks, as sensitive information like passwords should never be shared. Instead of exposing the entire user object, a DTO can be used to filter out unnecessary or sensitive data and only transfer the relevant information.
Therefore, when a user requests a list of other users in an application, the system does not need to send private details like passwords. As an alternative, the application can create an DTO that only contains essential information, such as first name, last name and email. This would ensure that the system remains secure while still providing necessary data for the application to function properly.
DTOs also help optimizing performance by limiting the amount of data transferred. In this case, the application would send the passwords between a client and a server every time the backend needs to send data. However, by using DTOs, the application is limited by sending the first name, last name and email but not the password.
To ensure DTOs are as efficient as possible within a backend application, certain best practices should be followed.

**Keeping DTOs Simple:**

- DTOs should only contain the necessary fields required for the specific use case that its needed for. Having too much data makes them harder to maintain and can lead to unnecessary complexity. By keeping DTOs organized, they remain efficient and can be used as they are intended.

**Using Validation:**

- Data validation in DTOs is required to maintain data integrity and prevent processing invalid or incomplete data. By applying validation rules, such as not allowing a field to be empty or limiting the length of different fields, errors are caught early. This prevents incorrect data from being passed on to the logic or inserted to the database.

**Using Automation Tools:**

- Manually constructing an DTO based off of an entity is known to create errors and to be time consuming. To simplify this, there are tools that can be used to map objects automatically. They reduce the usage of unnecessary code and speed up the development process by getting DTOs properly filled without writing manual code.

**Documenting the DTOs:**

- Proper documentation of every DTO created in the application is crucial. Frameworks like Swagger or SpringFox provide the possibility of auto-generated API documentation, which is easily understandable

by other developers concerning the structure and expected data of each DTO. This makes collatboration easier and consistent within the project.

While DTOs provide numerous benefits in organizing data transfer between the layers of an application, there are some challenges that come with them. As a project grows in complexity, it may take more effort to keep the DTO approach organized. It is wise to know the possible disadvantages so that DTOs can be utilized effectively without introducing unnescessary overhead.

**Maintenance Overhead:**

- As an application evolves, the number of DTOs can grow extensively. Monitoring these DTOs, modifying them as new requirements emerge and maintaining them synchronized with the logic can prove to be an overwhelming task. Without proper structuring and an explicit DTO strategie, their maintenance can introduce excessive complexity.

**Performance Impact:**

- While DTOs are used to reduce the amount of data transferred over the network, mapping entities to DTOs and vice versa is extra processing. This process will be a bottleneck in performance in high-traffic applications. Using optimized mapping strategies and automated tools for this purpose can help this problem. This still needs to be taken into consideration when implementing DTOs.

**Consistency:**

- Keeping DTOs and entity models in sync over time is challenging. When there are any changes in the underlying data structure, it is essential that the impacted DTOs are also adjusted accordingly. If these changes are not carefully handled, inconsistencies may arise which leads to potential data mismatches and errors in API responses.

**Complexity:**

- In use cases involving simple data interactions, DTOs may not be required. Adding DTOs to every operation will probably add a layer of abstraction that will not be as valuable. In this scenario, working with entities directly could be a simpler and cleaner approach.

# 6 Area Borders

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

## 6.1 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.

### 6.2 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.

### 6.3 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.

### 6.4 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.

### 6.5 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.

### 6.6 Integration of the Algorithm into the Backend

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

### 6.7 Challenges and Adjustments

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

## 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 Final Thoughts

## 18.1 Leon Edlinger

## 18.2 Paul Gigler

## 18.3 Andreas Weissl

# 19 Meetings

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

# 20 Working Hours

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

Table 1: Arbeitszeitnachweis

# 21 Source code directory

Source Code directory, kein plan was des is

# 22 List of figures

# 23 List of tables

# 24 Bibliography

[] *Flutter for Beginners*. URL: 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.

[25] *flutter/README.md at master · flutter/flutter*. [Online; accessed 23. Jan. 2025]. Jan. 2025. URL: https://github.com/flutter/flutter/blob/master/README.md.

[Dag19] Lukas Dagne. “Flutter for cross-platform App and SDK development”. In: (2019).

# 25 Abbreviation

| | |
|---|---|
| ADC | Analog Digital Converter |
| API | Application Programming Interface |
| BLE | Bluetooth Low Energy |
| CPU | Central Processing Unit |
| DAC | Digital Analog Converter |
| DAVE | Digital Application Virtual Engineer |
| DSP | Digital Signal Processor |
| FPU | Floating Point Unit |
| FPV | First Person View, First Pilot View |
| GPIO | General Purpose Input/Output |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| HDMI | High Definition Multimedia Interface |
| $I^2C$ | Inter-Integrated Circuit |
| IDE | Integrated Development Environment |
| IP | Internet Protocol |
| RPI | Raspberry Pi |
| SD | Secure Digital |
| SPI | Serial Peripheral 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 |