Noël Joost & Elena Gerber

Swiss Emergency Map

Eine Feuerwehrkarten-Webapplikation

Semesterarbeit

OST – Ostschweizer Fachhochschule
Campus St. Gallen

Betreuung

Stefan F. keller

20. Dezember 2024

Zusammenfassung

Ziel dieser Arbeit ist die Entwicklung einer Webapplikation mit Übersichtskarte und Lageplanverwaltung für die Feuerwehr. Als Basis dient die Notfallkarte der Feuerwehr Gossau ZH, die Verbesserungspotential aufweist. Zurzeit werden vertrauliche Dokumente beispielsweise über einen Dropbox-Link verwaltet, was den direkten Zugriff darauf erschwert. Zudem bietet die App keine Möglichkeit, Lagepläne einfach und sicher abzulegen und auf der Karte zu visualisieren. Nun soll eine App entwickelt werden, die Points-of-Interest (POI) und die genannten Dokumente auf einer interaktiven Karte visualisiert. Dadurch sollen Feuerwehrleute schnell auf die benötigten Informationen zugreifen können.

Nach der Einarbeitung in die bestehende Webapplikation wurden die wichtigsten Software-Technologien evaluiert. Für das Frontend wurden React mit TypeScript, für das Backend Django und Python gewählt. Die Software ermöglicht das Hochladen von Lageplänen und deren Verknüpfung mit POIs. Zusätzlich zu den Lageplänen können Nutzer weitere Dokumente verwalten, die ausschliesslich lokal gespeichert werden. Ausserdem ist die App mandantenfähig, so dass sie von mehreren unabhängigen Benutzergruppen (d.h. Feuerwehren) gleichzeitig genutzt werden kann. Im Mittelpunkt steht eine Hintergrundkarte (Satellitenbild oder Standard-Karte), die auf Basis von OpenStreetMap-Daten POI anzeigt, wie Hydranten, Schlüsselrohre, Defibrillatoren, Notfalltreffpunkte, Solaranlagen, Brandmeldezentralen und Feuerwachen. Die OSM-Daten werden in einer eigenen Datenbank gecached, um Ausfällen und Vandalismus vorzubeugen.

Das Ergebnis der Arbeit ist eine dockerisierte Webapplikation namens Swiss Emergency Map (Open Source) mit einer intuitiv bedienbaren grafischen Benutzeroberfläche, die auch unter schwierigen Einsatzbedingungen einen schnellen Zugriff auf alle wesentlichen Funktionen gewährleistet. Die konfigurierbare App kann sowohl von der Feuerwehr Gossau ZH als auch von anderen Feuerwehren genutzt werden. Damit wird die Einsatzvorbereitung und -durchführung der Feuerwehren verbessert und es werden Zeit und Ressourcen optimal genutzt.

Schlagwörter: Software, Verschiedenes, Application Design.

Management Summary

Ausgangslage und Vorgehen

Im Einsatzfall ist für Feuerwehren ein schneller und strukturierter Zugriff auf wichtige Informationen wie Lagepläne und Points-of-Interest (POI) unverzichtbar. Bestehende Lösungen, wie das Teilen von Dokumenten über Dropbox-Links oder die Nutzung der Notfallkarte Gossau, sind jedoch unzureichend strukturiert und bieten nicht die notwendigen Funktionalitäten.

Die aktuelle Notfallkarte der Feuerwehr Gossau ZH ist auf die Anzeige von POI aus OpenStreetMap (OSM) beschränkt. Vertrauliche Informationen, wie sicherheitsrelevante POI oder Lagepläne, können aus Datenschutz- und Sicherheitsgründen nicht in OSM integriert werden. Stattdessen werden solche Daten häufig über Dropbox geteilt. Dies erschwert den Zugriff, da es aufgrund der Vielzahl an Dateien zeitaufwendig sein kann, die benötigten Informationen zu finden.

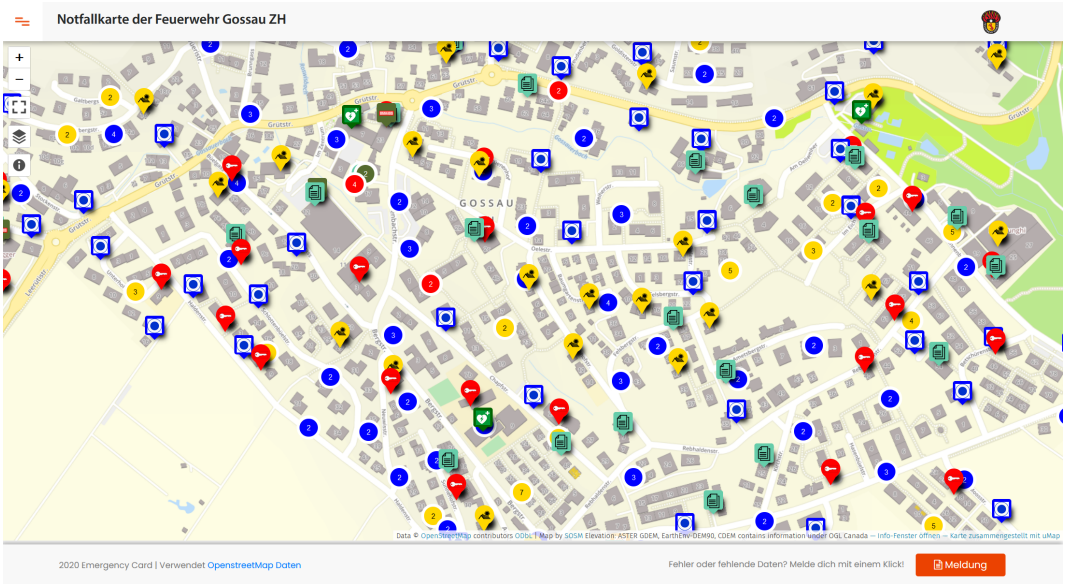

Abbildung 1: Die aktuelle Notfallkarte der Feuerwehr Gossau ZH mit verschiedenen POI.

Ziel dieses Projekts war, eine interaktive Webapplikation zu entwickeln, die es ermöglicht, POI, Lagepläne und Dokumente auf einer Übersichtskarte sicher zu visualisieren und zu verwalten. Diese Lösung soll

Feuerwehrleuten einen schnellen und sicheren Zugriff auf relevante Informationen bieten und so die Einsatzvorbereitung sowie -durchführung effizienter und sicherer gestalten.

Im ersten Schritt wurden die bestehenden Anforderungen der Notfallkarte analysiert und relevante Softwaretechnologien evaluiert. Für die Implementierung wurde React mit TypeScript für das Frontend sowie Django mit Python für das Backend ausgewählt. Diese Kombination ermöglicht eine modulare und wartbare Struktur der Anwendung, die sowohl eine ansprechende Benutzeroberfläche als auch leistungsstarke Backend-Funktionalitäten bietet.

Ein zentraler Bestandteil der Anwendung ist die Verwendung der Open-Source-Library Maplibre. Diese ermöglicht eine flexible und leistungsstarke Kartenvisualisierung, die auf OpenStreetMap-Daten basiert. Maplibre wurde gewählt, da sie die Erstellung von interaktiven Karten unterstützt und zukunftssichere Erweiterungsmöglichkeiten bietet. Obwohl derzeit keine mobile Version der Anwendung entwickelt wurde, sorgt Maplibre dafür, dass eine spätere Implementierung einfach realisierbar ist.

Result

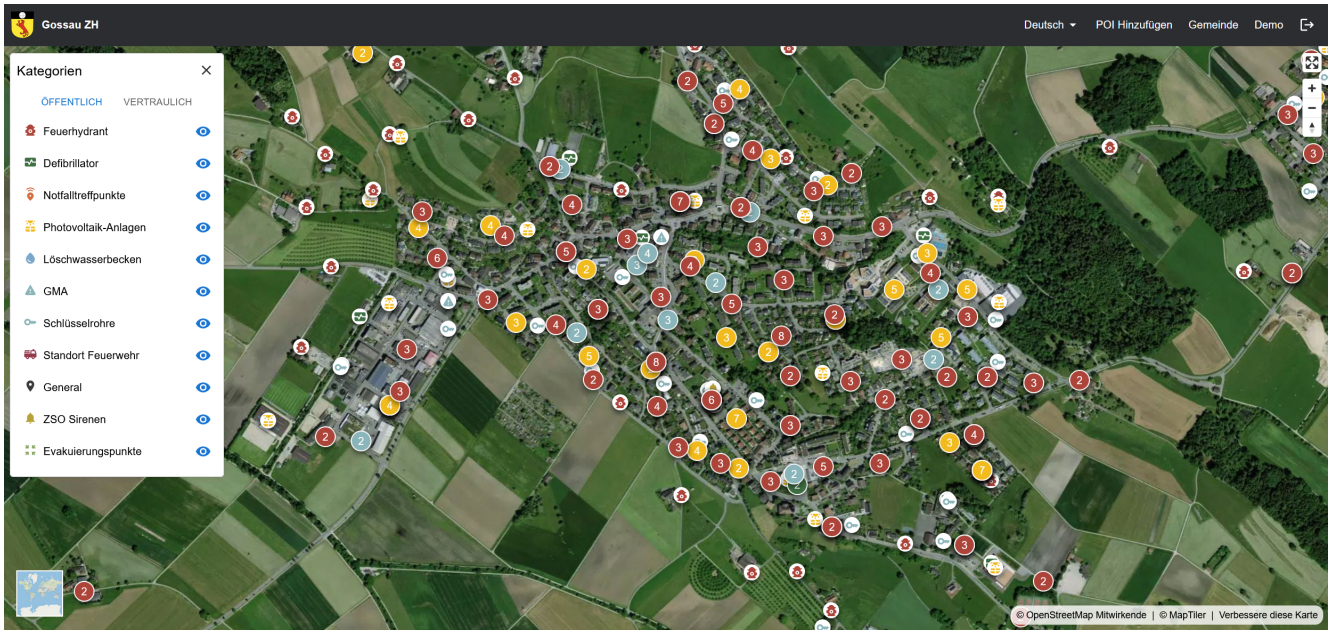

Abbildung 2: Die Swiss Emergency Map mit einer Übersichtskarte (Satellitenbild) und mit Points-of-Interest (POI) wie z.B. Hydranten.

Das Ergebnis der Arbeit ist eine dockerisierte Webapplikation namens Swiss Emergency Map (Open Source) mit einer intuitiv bedienbaren grafischen Benutzeroberfläche, die einen schnellen Zugriff auf alle wesentlichen Funktionen gewährleistet. Ein zentrales Merkmal der Anwendung ist die Integration einer interaktiven Übersichtskarte, die es ermöglicht, Points-of-Interest (POIs) zu visualisieren. Die Anwendung erlaubt es den Nutzern, POIs zu erstellen, die lokal gespeichert werden und nicht in OpenStreetMap (OSM) integriert sind. Diese POIs können mit vertraulichen Dokumenten, wie Lageplänen, verknüpft werden, die

direkt in der App hochgeladen werden können. Dadurch wird sichergestellt, dass sensible Informationen sicher verwaltet und abgerufen werden können. Zusätzlich bietet die App eine Standortsuche mit

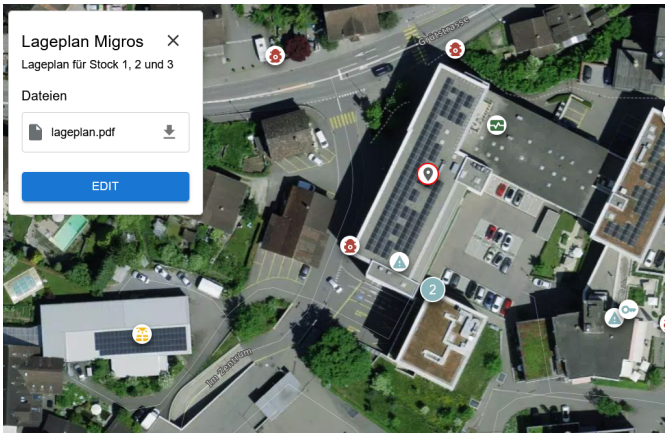

Abbildung 3: Download des Lageplans des Migros-Gebäudes in Gossau ZH aus der Karte heraus.

Geocoding-Funktionalität, die es den Nutzern ermöglicht, spezifische Standorte zu finden und diese in der Übersichtskarte anzuzeigen. Dies verbessert die Benutzererfahrung und erleichtert den schnellen Zugriff auf relevante Informationen in Notfallsituationen. Um die Verfügbarkeit und Integrität der Informationen zu gewährleisten, werden die OSM-POIs in einer lokalen Datenbank gecached. Dies schützt die Anwendung vor möglichen Ausfällen und Vandalismus, die bei der Nutzung externer Datenquellen auftreten können.

Ausblick

Die Swiss Emergency Map hat noch Potenziale für zukünftige Erweiterungen. Obwohl die Anwendung bereits für mobile Geräte optimiert ist, könnte die Entwicklung einer nativen App den Zugriff auf wichtige Informationen während Einsätzen weiter erleichtern und die Flexibilität der Feuerwehrleute erhöhen. Derzeit ist das Projekt von mehreren APIs abhängig, darunter Overpass für OSM-POI-Anfragen, Nominatim für Geocoding und Maptiler für Kartenvisualisierung. Diese Abhängigkeit kann die Stabilität der Anwendung beeinflussen. Eine eigene Lösung für Geocoding und Kartenvisualisierung würde mehr Kontrolle und Unabhängigkeit gewährleisten. Zukünftige Funktionen könnten die Anpassung von Icons und Farben für POIs, einen Passwort-Reset-Service sowie die Möglichkeit zur Verwaltung und Löschung von Kategorien umfassen. Insgesamt gibt es viele Möglichkeiten zur Weiterentwicklung der Swiss Emergency Map, um die Effizienz und Benutzerfreundlichkeit weiter zu steigern.

Inhaltsverzeichnis

Teil I

Technical Report

Kapitel 1

Einführung

1.1 Problemstellung

Ziel ist es, eine Feuerwehrkarten-App mit Objektklassen Hydranten, Feuerwehrhäusern etc. für Gemeinden zu realisieren, wobei einerseits eine Gemeinde als Beispiel dient und andererseits der Code in einem öffentlichen Repository abgelegt und gut dokumentiert ist, so dass Webentwickler diesen als Vorlage verwenden können.

1.2 Vision

Die Vision ist es, eine benutzerfreundliche und interaktive Karte zu entwickeln, die Point-of-Interests (POIs) übersichtlich darstellt und mit relevanten Informationen anreichert. Neben der reinen Visualisierung der POIs soll die Karte die Möglichkeit bieten, Dokumente hochzuladen und diese gezielt mit einzelnen POIs zu verknüpfen.

1.3 Ziele

- Evaluieren der Kartentechnologien, beginnend mit uMap [1], dann Leaflet [2] oder MapLibre JS/Native [3] (GeoAdmin JS/OpenLayers, OpenLayers, Mapbox GL JS [4], Maptiler SDK JS [5] [6], Google Maps)
- Eine Lösung ohne Back-End à la verbesserte Notfallkarte als Template für Gemeinden auf Basis "Notfallkarte Gossau".
- Eine Lösung mit Back-End
  - Mit internem Bereich mit Login inkl. Benutzerverwaltung.
  - Reine OSM-Daten: Verwalten vom Cache und Synchronisierung von OpenStreetMap-Daten

  - Kombination von OSM-Daten mit lokalen Daten, z.B. interne Bemerkungen sowie Fotos von Schlüsselrohren

- Clevere Anzeige von teilweise vielen POIs auf dem Plan/Karte.
- Mandantenfähigkeit (z.B. mit Django [7] + RLS)

1.4 Rahmenbedingungen

Dieses Projekt ist eine Semesterarbeit, die einen Zeitaufwand von 480 Stunden umfasst. Die Zeit wird auf zwei Teammitglieder aufgeteilt. Das entspricht 8 ECTS-Punkte pro Person.

Kapitel 2

Stand der Technik

Dieses Kapitel zeigt kurz auf, was die momentanen Lösungen auf dem Markt sind.

2.1 Existierende Lösungen

2.1.1 Notfallkarte Gossau ZH

Die Notfallkarte Gossau zeichnet sich durch ihre Übersichtlichkeit und schnelle Ladezeiten aus und dient als hervorragendes Beispiel für die effektive Nutzung interaktiver Karten zur Visualisierung von Point-of-Interests.

2.1.2 UTM Einsatzkarte

Die App macht deine Einsatzführung einfach und effizient: Man kann Standorte teilen, UTM-Koordinaten nutzen, Einsätze organisieren und Suchaktionen koordinieren. Erstelle Einsätze oder Übungen im Handumdrehen und lade weitere Einsatzkräfte dazu ein – so bleibt jeder immer up-to-date. Egal ob Hochwasser, Waldbrände oder grosse Suchaktionen: Mit taktischen Zeichen für Wasserentnahmestellen, Einsatzleitstellen, Gefahrenzonen oder Schlauchleitungen behältst du den Überblick und koordinierst Feuerwehr, Rettung, Polizei, Hubschrauber oder Suchhunde mühelos. [8]

2.2 Nachteile

Ein zentraler Nachteil der oben genannten Applikationen ist, dass die Übersichtlichkeit bei einer hohen Dichte von POIs eingeschränkt werden kann. Zudem bietet die Karte keine Möglichkeit, Pläne oder andere Dokumente direkt hochzuladen oder auf der Karte darzustellen. Dies limitiert die Einsatzmöglichkeiten, insbesondere in Szenarien, in denen zusätzliche visuelle Informationen oder Dateien benötigt werden.

2.3 Vorteile von Swiss Emergency Map

Die Swiss Emergency Map (SEM) vereint die Vorteile verschiedenen Ansätzen und bietet eine innovative Lösung für Notfalleinsätze. Sie zeichnet sich durch eine übersichtliche und intuitive Benutzeroberfläche aus, die eine einfache Bedienung ermöglicht und so den Fokus auf das Wesentliche legt. Als Open-Source-Plattform ist sie flexibel anpassbar und fördert die gemeinschaftliche Weiterentwicklung. Gleichzeitig gewährleistet das System höchste Sicherheitsstandards beim Hochladen und Verwalten von Daten. Ein besonderes Highlight ist die Möglichkeit, hochgeladene Informationen mit Points-of-Interest (POIs) zu verknüpfen, was eine präzise und effiziente Einsatzplanung unterstützt.

Kapitel 3

Implementierungs-Konzept

In diesem Kapitel wird beschrieben, wie zur Implementierung gelangt wurde und welche Schritte dabei durchlaufen wurden.

3.1 Wissenssammlung

Zu Beginn des Projekts war es wichtig, eine Wissensbasis aufzubauen. Dabei wurde analysiert, welche bestehenden Lösungen vorhanden sind und welche Technologien in Frage kommen. Ein zentraler Bestandteil dieser Phase war die Untersuchung von Kartenbibliotheken, um die passende Grundlage für die Anwendung zu finden.

3.2 Spezifikation der Anforderungen

Bevor mit der Auswertung des Tools begonnen werden konnte, mussten die funktionalen und nicht-funktionalen Anforderungen definiert werden. Die funktionalen Anforderungen legen fest, welche konkreten Funktionen die Anwendung erfüllen muss, während die nicht-funktionalen Anforderungen Aspekte wie Performance und Benutzerfreundlichkeit umfassen.

3.3 Auswertung

Basierend auf den definierten Anforderungen wurden verschiedene Tools und Technologien evaluiert. Dabei war es entscheidend, dass die ausgewählten Werkzeuge den Anforderungen entsprechen und sowohl für die aktuelle Implementierung als auch für zukünftige Erweiterungen geeignet sind.

3.4 Architektur

Vor der eigentlichen Implementierung wurde eine Architektur entworfen, die als Grundlage für die Entwicklung dient. Diese Architektur wurde so gestaltet, dass sie flexibel genug ist, um neue Features problemlos zu integrieren, gleichzeitig aber einfach bleibt, um die Implementierung und Wartung nicht zu erschweren.

3.5 Implementierung

Nach der Auswertung und der Festlegung der Architektur begann die eigentliche Implementierung. Dank der vorangegangenen Schritte konnte die Implementierung zielgerichtet und effizient durchgeführt werden. Der strukturierte Prozess stellte sicher, dass alle Anforderungen erfüllt und mögliche Erweiterungen von Anfang an berücksichtigt wurden.

Kapitel 4

Technologie-Evaluation

In diesem Kapitel wird beschrieben, wie die verschiedenen Technologien und Tools ausgewählt wurden. Dabei wurden die Anforderungen geprüft und passende Lösungen identifiziert. Zusätzlich wurde ein Prototyp entwickelt, um die Machbarkeit der gewählten Ansätze zu testen und erste praktische Erkenntnisse zu gewinnen.

4.1 Akzeptanzkriterien

Die Akzeptanzkriterien für dieses Projekt umfassen mehrere entscheidende Aspekte. Eine einfache Umsetzung ist von zentraler Bedeutung, um den Entwicklungsprozess effizient zu gestalten. Ebenso wichtig ist eine gute Dokumentation, die sowohl die interne Zusammenarbeit als auch die spätere Nutzung und Weiterentwicklung erleichtert. Die Wahl einer Open-Source-Lösung ist ein weiterer essenzieller Faktor, da sie nicht nur Flexibilität bietet, sondern auch die gemeinschaftliche Weiterentwicklung und Anpassung ermöglicht. Die vorhandene Erfahrung im Team spielt eine entscheidende Rolle, um einen reibungslosen Start und eine erfolgreiche Umsetzung des Projekts sicherzustellen.

4.2 Frontend-Framework

4.2.1 Angular

Vorteile:

- Umfassendes Framework mit vielen integrierten Funktionen (z. B. Routing, State-Management).
- Grosse Community und umfangreiche Dokumentation.
- Unterstützt die Entwicklung skalierbarer und strukturierter Anwendungen.

Nachteile:

- Höhere Einstiegshürde aufgrund der Komplexität.
- Teilweise Overhead für kleinere Projekte.
- Stark vorgegebene Struktur, weniger flexibel als andere Frameworks.

Erfahrung: Elena hat nicht viel Erfahrung mit Angular.

4.2.2 React

Vorteile:

- Flexible und modulare Architektur, gut geeignet für individuelle Anforderungen.
- Grosse Community und vielseitiges Ökosystem mit vielen Erweiterungen.
- Einfach zu erlernen und zu verwenden, besonders für kleinere bis mittelgrosse Projekte.

Nachteile:

- Viele Drittanbieter-Bibliotheken sind notwendig, um ein vollständiges System aufzubauen.
- Schnell wechselnde Versionen und Abhängigkeiten können zu Wartungsaufwand führen.
- Keine vorgegebene Struktur, was zu Inkonsistenz führen kann.

Erfahrung: Beide haben Erfahrung mit React. [9]

4.2.3 Vue.js

Vorteile:

- Einfache Lernkurve und sehr gut geeignet für kleinere bis mittelgrosse Projekte.
- Klare und intuitive Struktur, ideal für Einsteiger.
- Flexibel und leicht erweiterbar durch das Plugin-System.

Nachteile:

- Kleinere Community und weniger umfangreiche Ressourcen im Vergleich zu Angular und React.
- Begrenzte Unterstützung für sehr grosse und komplexe Anwendungen.
- Abhängigkeit von einer kleineren Entwicklergruppe im Vergleich zu React und Angular.

Erfahrung: Keiner hat Erfahrung mit Vue.js. [10]

4.2.4 Entscheidung

Nach einer sorgfältigen Abwägung der Vor- und Nachteile wurde entschieden, dass React für dieses Projekt verwendet wird. Die Entscheidung basiert auf der Flexibilität und Modularität des Frameworks sowie der vorhandenen Erfahrung im Team.

4.3 Backend-Framework

4.3.1 Django

Vorteile:

- Umfassendes Framework mit vielen integrierten Funktionen wie Authentifizierung, Datenbankanbindung und Admin-Interface.
- Gute Dokumentation und grosse Community.
- Schnell einsetzbar für prototypische und produktive Entwicklungen.

Nachteile:

- Weniger flexibel bei der Kombination mit anderen Frontend-Frameworks.
- Kann für kleinere Projekte zu umfangreich sein.

Erfahrung: Beide Teammitglieder haben grundlegende Erfahrungen mit Django.

4.3.2 Flask

Vorteile:

- Minimalistisches Framework, das maximale Flexibilität bietet.
- Gut geeignet für kleine und schlanke Anwendungen.
- Einfache und schnelle Einrichtung.

Nachteile:

- Viele Erweiterungen notwendig, um eine umfassende Funktionalität zu erreichen.
- Weniger geeignet für grosse und komplexe Projekte.

Erfahrung: Keiner hat Erfahrung mit Flask [11].

4.3.3 Spring Boot

Vorteile:

- Leistungsstarkes Framework für komplexe, skalierbare Anwendungen.

- Gute Unterstützung für Microservices-Architekturen.
- Umfangreiche Community und Dokumentation.

Nachteile:

- Relativ hohe Einstiegshürde.
- Kann für kleinere Projekte zu umfangreich sein.

Erfahrung: Noël hat ein wenig Erfahrung mit Spring Boot [12].

4.3.4 Entscheidung

Django wurde aufgrund seiner umfassenden Funktionen, schnellen Einsatzmöglichkeiten und der vorhandenen Erfahrung im Team ausgewählt. Es bietet die notwendige Stabilität und Effizienz, um die Anforderungen des Projekts zu erfüllen.

4.4 Test-Framework

Nach der Auswahl der Frameworks wurde im nächsten Schritt die passenden Test-Frameworks evaluiert.

4.4.1 React Test

Jest

- **Vorteile:** Umfassende Testumgebung speziell für React-Anwendungen, einfache Integration, schnelle Testläufe.
- **Nachteile:** Fokus liegt auf Unit-Tests, weniger geeignet für End-to-End-Tests.

[13]

Cypress

- **Vorteile:** Ideal für End-to-End-Tests, intuitive Benutzeroberfläche, einfache Einrichtung.
- **Nachteile:** Kann langsamer sein bei umfangreichen Tests, weniger geeignet für Unit-Tests.

Erfahrung:

Beide sind mit Cypress vertraut. [14]

4.4.2 Python Test

Pytest

- **Vorteile:** Flexible und mächtige Testumgebung, unterstützt Plugins, einfache Syntax.
- **Nachteile:** Weniger geeignet für Anfänger aufgrund der mächtigen Features.

[15]

Unittest

- **Vorteile:** Eingebaute Bibliothek in Python, einfache Einrichtung, gut dokumentiert.
- **Nachteile:** Weniger flexibel im Vergleich zu Pytest, erfordert mehr Boilerplate-Code.

[16]

Erfahrung:

Alle sind mit Pytest und Unittest vertraut.

4.4.3 Entscheidung

- **Frontend:** Das Team hat sich für Cypress entschieden, da es eine benutzerfreundliche Lösung für End-to-End-Tests bietet und es bereits vertraut mit Cypress ist.
- **Backend:** Für Python-Tests fiel die Wahl auf Unittest, da es eine flexible und effiziente Lösung für umfangreiche Testanforderungen bietet, es gut dokumentierte Erweiterungen für Django existieren und es die Empfehlung von Django ist[17].

4.5 Weitere Bibliotheken und Technologien

4.5.1 Kartenbibliotheken

Für die Implementierung der Kartenfunktionalität wurden verschiedene Bibliotheken evaluiert und dabei deren Vor- und Nachteile abgewogen.

MapLibre

- **Vorteile:** Open-Source-Lösung ohne Lizenzkosten, unterstützt Vektor- und Rasterkarten, gut dokumentierte Library, hohe Flexibilität für individuelle Anpassungen. Zudem wurde MapLibre als Empfehlung von Stefan F. Keller hervorgehoben.
- **Nachteile:** Höhere Einstiegshürde für Anfänger, weniger Community-Support im Vergleich zu etablierten Bibliotheken wie Leaflet.

Leaflet

- **Vorteile:** Sehr einfach zu verwenden, grosse Community, umfangreiche Plugins verfügbar. Erfahrungen mit Leaflet wurde bereits durch ein früheres Softwareentwicklungsprojekt gesammelt.
- **Nachteile:** Eingeschränkte Unterstützung für Vektorkarten, weniger flexibel für komplexe Anpassungen.

Mapbox

- **Vorteile:** Professionelle und moderne Kartendarstellung, bietet umfangreiche Funktionen und eine breite Unterstützung für verschiedene Datenformate.
- **Nachteile:** Lizenzkosten bei kommerzieller Nutzung, Abhängigkeit von einem proprietären Anbieter.

Entscheidung

Das Team hat sich für **MapLibre** entschieden, da es eine gut dokumentierte Open-Source-Bibliothek ist, die Vektor- und Rasterkarten unterstützt und eine hohe Flexibilität bietet. Die Empfehlung von Stefan F. Keller hat die Wahl zusätzlich bestärkt.

4.5.2 Supercluster

Für das Clustering der Punkte wurde die Bibliothek SSupercluster"verwendet. Mit dieser Lösung konnte das Team bereits in der Vergangenheit positive Erfahrungen sammeln. Supercluster ermöglicht es, eine grosse Anzahl von Punkten effizient zu gruppieren, sodass diese als Cluster dargestellt werden können. Dadurch wird die Übersichtlichkeit auf der Karte verbessert, insbesondere bei hoher Punktdichte.

4.5.3 GitLab

Für die Entwicklung und Bereitstellung der Anwendung wurde GitLab verwendet, das eine einfache und effektive CI/CD-Pipeline bietet. Die Automatisierung ermöglichte es, kontinuierlich Änderungen zu integrieren und zu überprüfen.

Frontend: Im Frontend wurden die folgenden Schritte in der CI/CD-Pipeline umgesetzt:

- **Build:** Automatisches Erstellen der Anwendung, um sicherzustellen, dass der Code fehlerfrei kompiliert wird.
- **Tests:** Ausführen von Unit- und Integrationstests, um die Funktionalität der Anwendung zu überprüfen.
- **Linter:** Überprüfung des Codes auf Einhaltung von Style-Guidelines, um die Codequalität zu sichern.

Backend: Im Backend umfasst die CI/CD-Pipeline:

- **Build:** Sicherstellen, dass das Backend korrekt kompiliert und einsatzbereit ist.
- **Tests:** Automatisches Ausführen von Tests, um die Stabilität und Funktionalität des Backends zu gewährleisten.
- **PyLint:** Analysieren des Python-Codes, um mögliche Fehler und Style-Verstösse zu identifizieren.

4.5.4 Docker

Sowohl das Frontend als auch das Backend wurden in Docker-Containern umgesetzt, um die Vorteile von Docker zu nutzen:

- **Plattformunabhängigkeit:** Docker-Container laufen unabhängig von der zugrunde liegenden Infrastruktur, was die Portabilität der Anwendung erleichtert.
- **Einfache Einrichtung:** Durch die Verwendung von Dockerfiles können Entwicklungs- und Produktionsumgebungen schnell und konsistent bereitgestellt werden.
- **Isolierte Umgebungen:** Frontend und Backend laufen in separaten Containern, was Konflikte zwischen Abhängigkeiten vermeidet und die Stabilität erhöht.

4.5.5 OSM-POIs

Zunächst wurde https://terminal.osmdatapipeline.geoh.infs.ch/ verwendet, um die OSM-Daten abzurufen, da das Team die Arbeit mit SQL schätzte. Um jedoch die Abhängigkeit von einer spezifischen Plattform zu reduzieren, wurde später auf Overpass API [18] umgestiegen.

4.6 Prototyp

Es wurde mit uMap einen Prototypen erstellt, der sich durch seine einfache, schnelle und übersichtliche Handhabung auszeichnet. Der Prototyp wurde später genutzt, um die korrekte Positionierung der POIs zu überprüfen. Die Punkte wurden über die Overpass-API von OSM abgerufen. Diese Abfragen wurden im richtigen Projekt wiederverwendet.

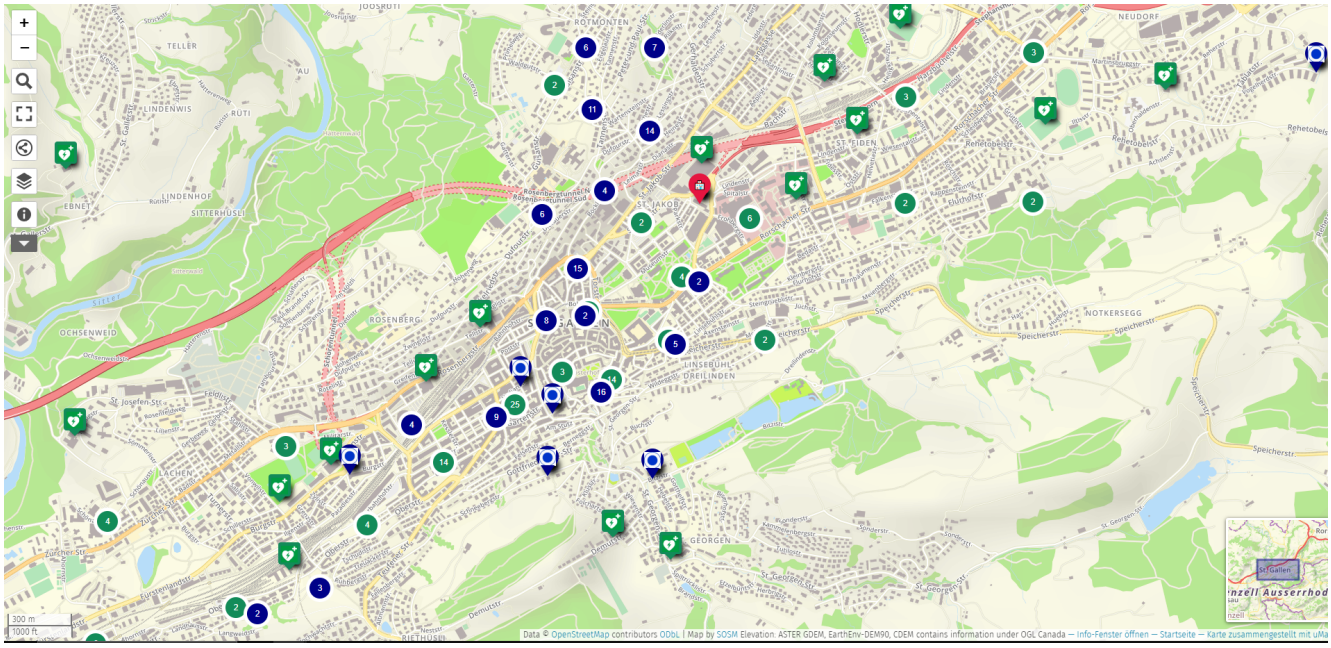

Abbildung 4.1: uMap

Kapitel 5

Resultate

In diesem Kapitel werden die Ziele präsentiert, die im Verlauf des Projekts erreicht wurden, und es wird ein Überblick darüber gegeben, welche Aufgaben noch ausstehen.

5.1 Zielerreichung

Die Erreichung des Projektziels wurde anhand der umgesetzten User-Stories gemessen. Alle der definierten User-Stories konnten erfolgreich abgeschlossen werden, was zeigt, dass die Hauptziele des Projekts erreicht wurden.

Allerdings gibt es noch eine Liste offener Bugs, die nach Abschluss der Studienarbeit behoben werden müssen. Diese Fehler sind dokumentiert und priorisiert, sodass sie in zukünftigen Arbeiten systematisch angegangen werden können.

Die spezifischen User-Stories sowie die funktionalen (FR) und nicht-funktionalen Anforderungen (NFR), die im Rahmen des Projekts behandelt wurden, sind in den Kapiteln 8.2.2 und 8.3 detailliert beschrieben.

5.2 Vergleich

In diesem Abschnitt wird die entwickelte Lösung (SEM) mit der "Notfallkarte-Gossau.ch", die derzeit in Gossau ZH verwendet wird, verglichen.

Die Notfallkarte von Gossau ZH basiert auf einer statischen Lösung, die weniger interaktiv ist und nur eingeschränkte Anpassungsmöglichkeiten bietet. Sie erfüllt zwar grundlegende Anforderungen, ist jedoch in ihrer Funktionalität begrenzt, insbesondere wenn es um das Verwalten von Dateien geht.

Im Gegensatz dazu bietet SEM eine interaktive Benutzeroberfläche. Zudem ermöglicht SEM eine sehr einfache Verbindung von Dateien mit Points-of-Interest. Ein weiterer Unterschied liegt in der Usability: SEM wurde auf Basis von User-Stories und spezifischen Anforderungen entwickelt, die eng mit den Bedürfnis-

sen der Nutzer abgestimmt wurden. Dies gewährleistet eine höhere Benutzerfreundlichkeit im Vergleich zur bestehenden Notfallkarte.

Zusammenfassend zeigt der Vergleich, dass diese Lösung eine moderne und zukunftssichere Alternative darstellt, die den aktuellen Anforderungen besser gerecht wird und Raum für Erweiterungen bietet.

Kapitel 6

Zukunft von ESM

Auf GitLab ist eine Liste mit bekannten Bugs vorhanden, die in naher Zukunft behoben werden sollen. Unmittelbar vor Abschluss des Projekts wurden zusätzliche Features identifiziert, die implementiert werden könnten. Kapitel 12 enthält eine detaillierte Beschreibung der geplanten zukünftigen Features.

Teil II

Project Documentation

Kapitel 7

Vision

Die Vision ist im Kapitel 1.2 beschrieben.

Kapitel 8

Anforderungs-Spezifikationen

8.1 Use-Case-Diagramm

Es wurde ein Use-Case-Diagramm erstellt, das als Grundlage für die User-Stories dient.

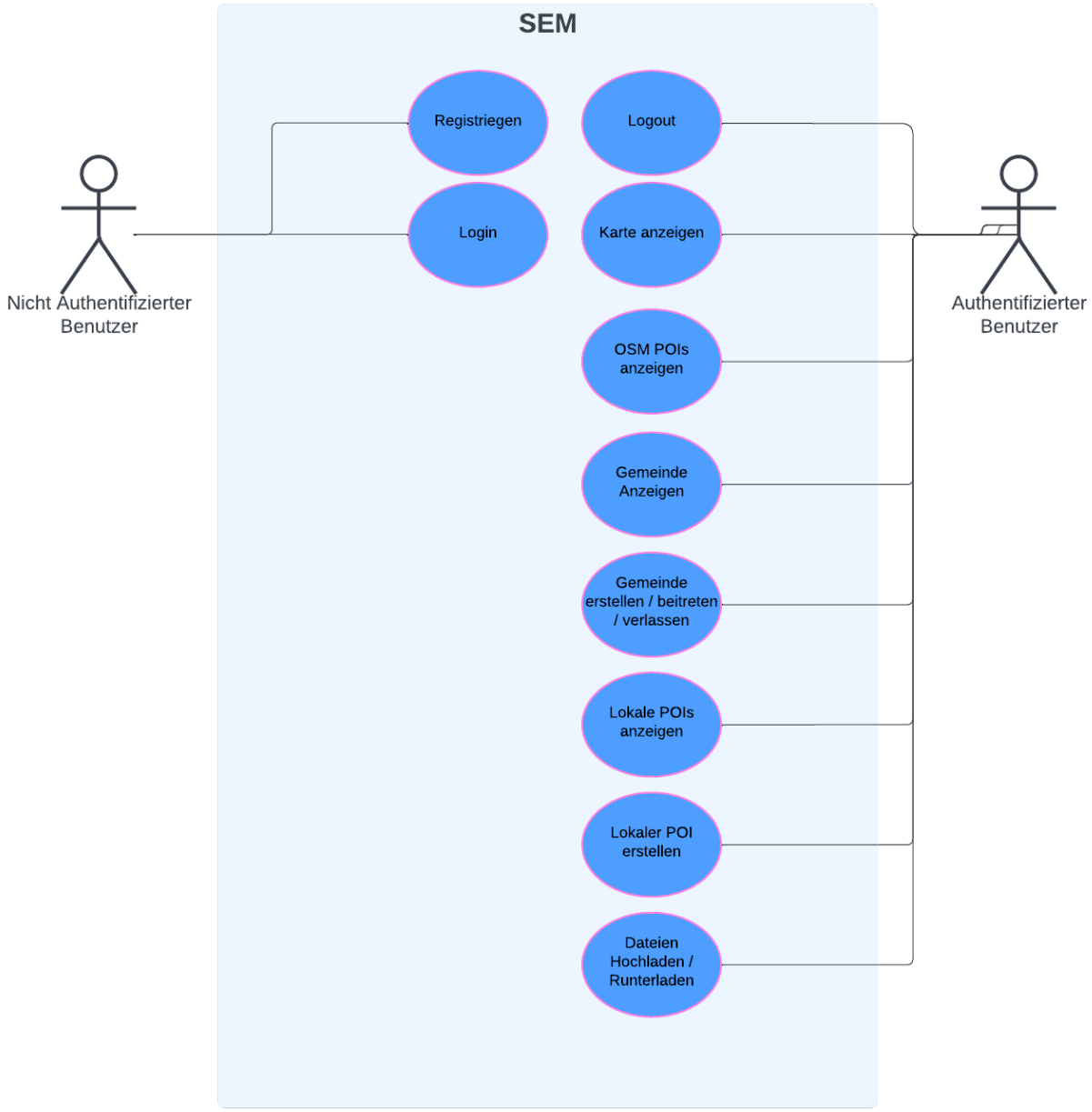

Abbildung 8.1: Use-Case-Diagramm für authentifizierte und nicht authentifizierte Benutzer

8.2 Funktionale Anforderungen

In diesem Kapitel werden kurz die Epics und User-Stories erläutert, diese dienen als funktionale Anforderungen.

8.2.1 Epics

In diesem Kapitel werden die Epics beschrieben.

Authentifizierung	
Schlüssel	FWK-1
Beschreibung	Das Epic zur Authentifizierung umfasst die Prozesse Login, Logout und Registrierung. Benutzer können sich sicher registrieren und authentifizieren, indem sie ein vereinfachtes, auf E-Mails basierendes Login-System verwenden. Nach erfolgreicher Authentifizierung verwaltet das System ihre Sitzungen während der gesamten Interaktion mit der Plattform sicher.
Status	Erledigt

Karte und Points of Interest	
Schlüssel	FWK-12
Beschreibung	Dieses Epic konzentriert sich auf die Verwaltung und Darstellung von Points of Interest (POIs).
Status	Erledigt

Gruppe	
Schlüssel	FWK-63
Beschreibung	Dieses Epic konzentriert sich auf die Verwaltung von Gruppen.
Status	Erledigt

Dateiverwaltung	
Schlüssel	FWK-63
Beschreibung	Dieses Epic konzentriert sich auf die Verwaltung von Dateien.
Status	Erledigt

8.2.2 User-Stories

Hier werden die User-Stories beschrieben.

Login	
Schlüssel	FWK-3
Beschreibung	Als registrierter Benutzer möchte ich mich sicher mit meinen Anmeldedaten einloggen, um auf mein Konto zugreifen zu können.
Status	Erledigt

Registrierung	
Schlüssel	FWK-6
Beschreibung	Als Benutzer möchte ich mich sicher auf der Plattform registrieren können, um deren Funktionen nutzen zu können.
Status	Erledigt

Logout	
Schlüssel	FWK-9
Beschreibung	Als Benutzer möchte ich mich sicher ausloggen können, um meine Sitzung zu beenden und meine Daten zu schützen.
Status	Erledigt

POIs anzeigen	
Schlüssel	FWK-13
Beschreibung	Als Benutzer möchte ich, dass POIs mit klaren Details auf der Karte angezeigt werden, um relevante Informationen leicht identifizieren zu können.
Status	Erledigt

Position suchen	
Schlüssel	FWK-21
Beschreibung	Als Benutzer möchte ich nach bestimmten Positionen suchen können, um die gewünschten Orte schnell zu finden.
Status	Erledigt

POIs CRUD	
Schlüssel	FWK-24
Beschreibung	Als Benutzer möchte ich POIs erstellen, lesen, aktualisieren und löschen können, um relevante Informationen auf der Karte zu verwalten.
Status	Erledigt

POIs aus Cache laden	
Schlüssel	FWK-29
Beschreibung	Als Benutzer möchte ich POIs von Open-Street-Map laden, um vorhandene Daten zu importieren.
Status	Erledigt

POIs cachen	
Schlüssel	FWK-29
Beschreibung	Als Benutzer möchte ich POIs von Open-Street-Map lokal cachen, um die Anwendungsleistung zu verbessern und Ladezeiten zu verkürzen.
Status	Erledigt

Kartenstil anpassen	
Schlüssel	FWK-34
Beschreibung	Als Benutzer möchte ich das visuelle Erscheinungsbild der Karte anpassen, um eine bessere Klarheit und Benutzerfreundlichkeit zu gewährleisten.
Status	Erledigt

POIs gruppieren	
Schlüssel	FWK-42
Beschreibung	Als Benutzer möchte ich POIs nach Kategorien oder Regionen gruppieren können, um die Organisation und Visualisierung zu verbessern.
Status	Erledigt

Gruppe erstellen	
Schlüssel	FWK-65
Beschreibung	Als Benutzer möchte ich eine Gruppe erstellen, um Benutzer in einem bestimmten Team oder einer Kategorie zu organisieren.
Status	Erledigt

Benutzer zu Gruppe einladen	
Schlüssel	FWK-66
Beschreibung	Als Benutzer möchte ich andere Benutzer zu einer Gruppe einladen können, damit sie teilnehmen und innerhalb der Gruppe zusammenarbeiten können.
Status	Erledigt

Gruppe verlassen	
Schlüssel	FWK-67
Beschreibung	Als Benutzer möchte ich eine Gruppe verlassen können, um nicht mehr Teil der Gruppe zu sein, wenn es erforderlich ist.
Status	Erledigt

Datei hochladen	
Schlüssel	FWK-69
Beschreibung	Als Benutzer möchte ich eine Datei hochladen und mit einer Gruppe verknüpfen, damit sie mit anderen Gruppenmitgliedern geteilt werden kann.
Status	Erledigt

Datei löschen	
Schlüssel	FWK-70
Beschreibung	Als Benutzer möchte ich Dateien löschen können, um gemeinsam genutzte Ressourcen zu verwalten und veraltete Informationen zu entfernen.
Status	Erledigt

Dateisichtbarkeit	
Schlüssel	FWK-67
Beschreibung	Als Benutzer möchte ich sicherstellen, dass nur Benutzer in meiner Gruppe bestimmte Dateien sehen können, damit sensible Informationen privat bleiben.
Status	Erledigt

8.3 Nicht-funktionale Anforderungen

In diesem Kapitel werden die nicht-funktionalen Anforderungen definiert.

Nr.	Anforderung	Priorität
	Allgemein	
NFR-01	Nur die Anwendung hat direkten Zugriff auf das Dateisystem und die Datenbank.	Muss
NFR-02	Die Anwendung verwendet Docker-Container für eine vereinfachte Bereitstellung.	Muss
	Leistung	
NFR-03	POI-Daten werden innerhalb von maximal 3 Sekunden geladen.	Muss
NFR-04	Die Grösse der Website darf 500 MB nicht überschreiten.	Soll
	Sicherheit	
NFR-05	Sensible Daten (z. B. Passwörter, API-Tokens) werden verschlüsselt gespeichert.	Muss
NFR-06	Nur autorisierte Benutzer dürfen POI- und Kartendaten ansehen oder ändern.	Muss
	Zuverlässigkeit	
NFR-07	Die Anwendung darf bei einem Fehler nicht abstürzen, sondern muss stattdessen einen Fehlerstatus zurückgeben.	Muss
NFR-08	Alle Benutzereingaben müssen validiert werden, um die Datenintegrität sicherzustellen.	Muss
	Benutzerfreundlichkeit	
NFR-09	Die Benutzeroberfläche ist intuitiv.	Muss
NFR-10	Die Benutzeroberfläche ist für Smartphones optimiert.	Muss

Nr.	Anforderung	Priorität
	Wartbarkeit	
NFR-11	Die Anwendung enthält automatisierte Tests für alle kritischen Funktionen.	Muss

Kapitel 9

Analyse und Evaluation

9.1 Domain Analyse

Hier ist die Domain Analyse.

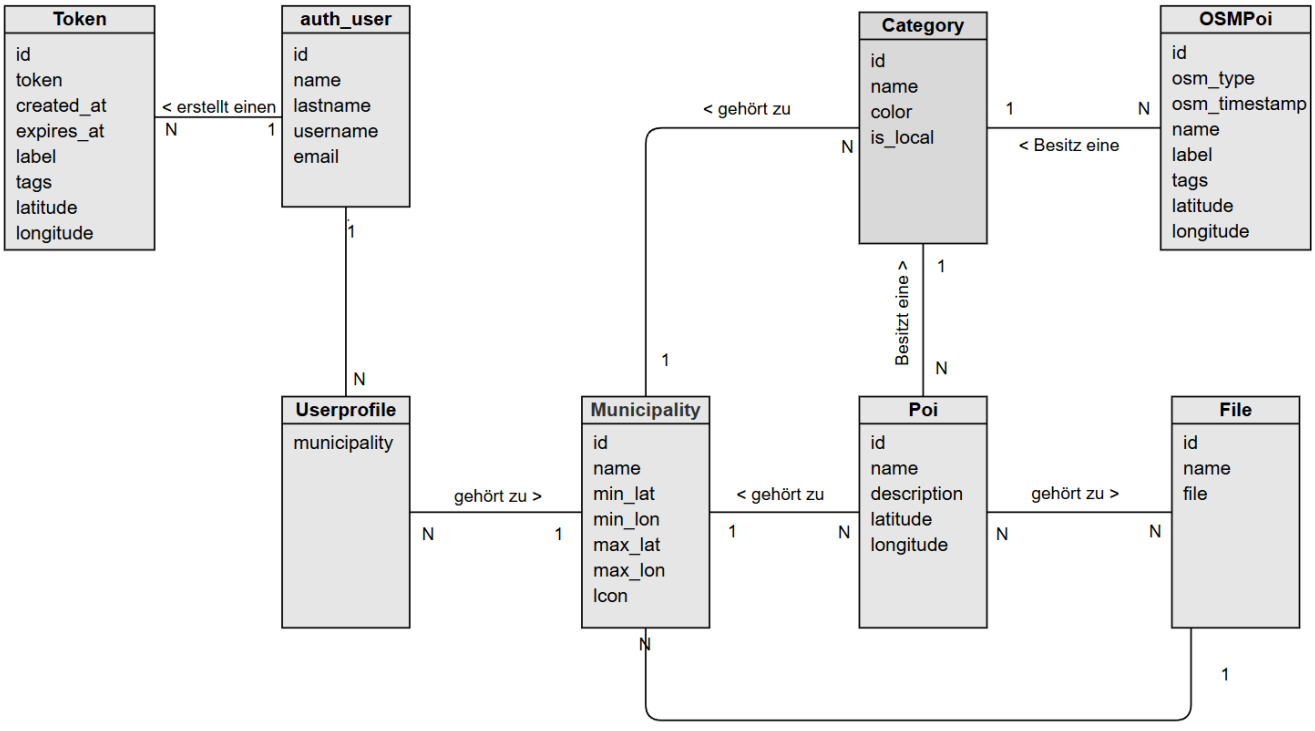

Abbildung 9.1: Domain Analyse

9.2 Objekt Katalog

9.2.1 Auth User

Die Klasse `Auth User` ist die Standard-Benutzerklasse von Django und wird für die Authentifizierung verwendet.

9.2.2 UserProfile

Die Klasse `UserProfile` besitzt eine Beziehung zur `Auth User`-Klasse. Diese Erweiterung ermöglicht es, zusätzliche Informationen wie die Gruppenzugehörigkeit zu verwalten.

9.2.3 Municipality

Die Klasse `Municipality` repräsentiert eine Gemeinde und enthält folgende Attribute:

- `min_lat`, `max_lat`, `min_long`, `max_long` – zur Definition der Bounding Box.
- `name` – der Name der Gemeinde.
- `logo` – das Logo der Feuerwehr.

9.2.4 POI (Point of Interest)

Ein `POI` gehört zu einer bestimmten Gruppe, um sicherzustellen, dass er nur von berechtigten Personen eingesehen werden kann. Ein `POI` hat folgende Eigenschaften:

- Kann eine oder mehrere Dateien enthalten.
- Ist immer einer Kategorie zugeordnet.

9.2.5 OSMPoi

`OSMPoi`-Objekte werden über die Overpass-API importiert. Sie verfügen über ähnliche Attribute wie die `POI`-Klasse:

- `lat` und `lon` – für die geografische Position.
- Mehrere Tags zur Beschreibung zusätzlicher Eigenschaften.

9.2.6 Category

Eine `Category` besitzt folgende Attribute:

- `color` – eine Farbe zur visuellen Unterscheidung.

- Ein Attribut zur Unterscheidung, ob die Kategorie lokal oder aus OSM (OpenStreetMap) hinzugefügt wurde.

Kapitel 10

Design

10.1 Architektur

Das Projekt verwendet eine Drei-Schichten-Architektur, um eine klare Trennung der Verantwortlichkeiten zu gewährleisten und die modulare Entwicklung zu fördern. Die **Präsentationsschicht**, entwickelt mit React, verwaltet die Benutzeroberfläche und Interaktionen, stellt Informationen dar und erfasst Benutzereingaben. Die **Applikationsschicht**, umgesetzt mit Django, fungiert als zentrale Verarbeitungsschicht, die die Logik bearbeitet und die Kommunikation zwischen der Präsentations- und der Datenschicht koordiniert. Die **Datenschicht**, betrieben mit PostgreSQL, verwaltet die Datenspeicherung und den Datenabruf effizient.

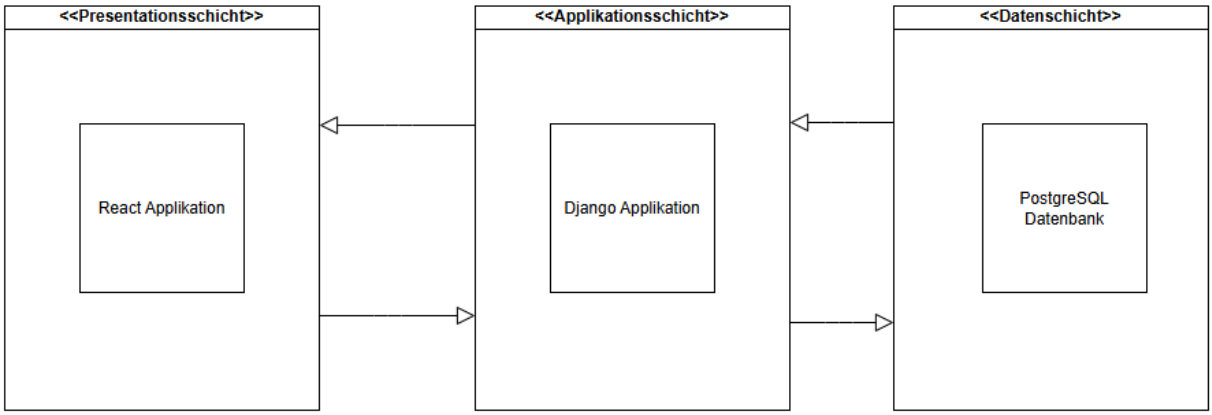

Abbildung 10.1: 3-Schichten Architektur

Dieser architektonische Ansatz, die Aufteilung der Anwendung in Präsentations-, Applikations- und Datenschicht ermöglicht die unabhängige Entwicklung spezifischer Funktionalitäten, verbessert die Wartbarkeit und vereinfacht Aktualisierungen. Änderungen in einer Schicht sind isoliert, wodurch das Risiko

von Störungen in anderen Schichten minimiert wird, was Stabilität und Skalierbarkeit sicherstellt. Der Einsatz weit verbreiteter Technologien wie React und Django erhöht die Entwicklungseffizienz, da sie von einer umfassenden Dokumentation und einer aktiven Community unterstützt werden. PostgreSQL stärkt das System zusätzlich mit seiner bewährten Zuverlässigkeit und seinen robusten Datenmanagement-Funktionen.

10.1.1 C4-Diagramme

C4-Modellierung führt zu einer leicht verständlichen Darstellung, die dennoch alle wichtigen Informationen über eine Architektur enthält. Es wurden nicht alle möglichen C4-Diagramme abgebildet. Unten ist das Systemkontext-Diagramm und das Container-Diagramm. Weitere Informationen zur C4-Modellierung finden Sie unter https://c4model.com/.

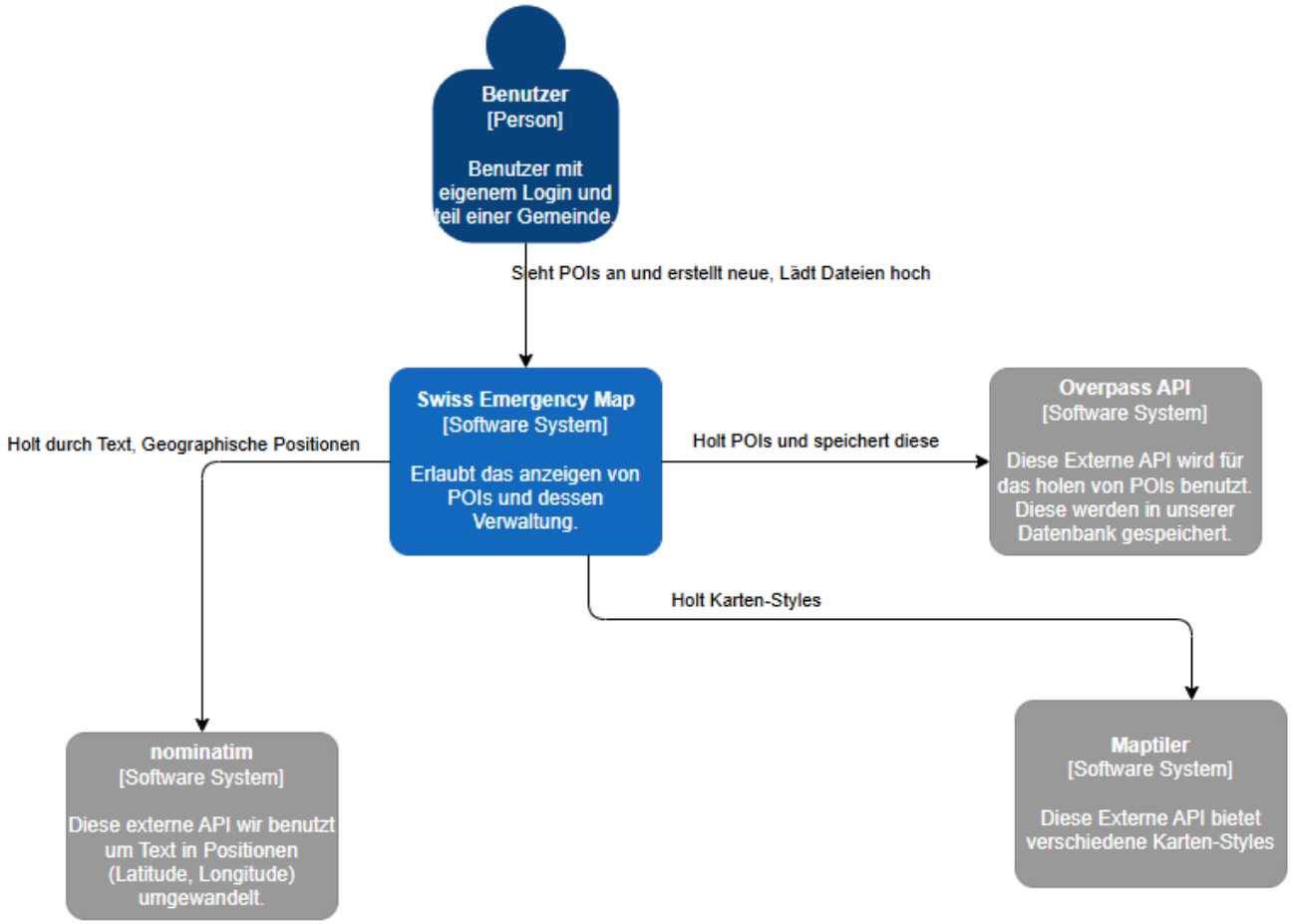

Abbildung 10.2: C4 System Context-Diagramm

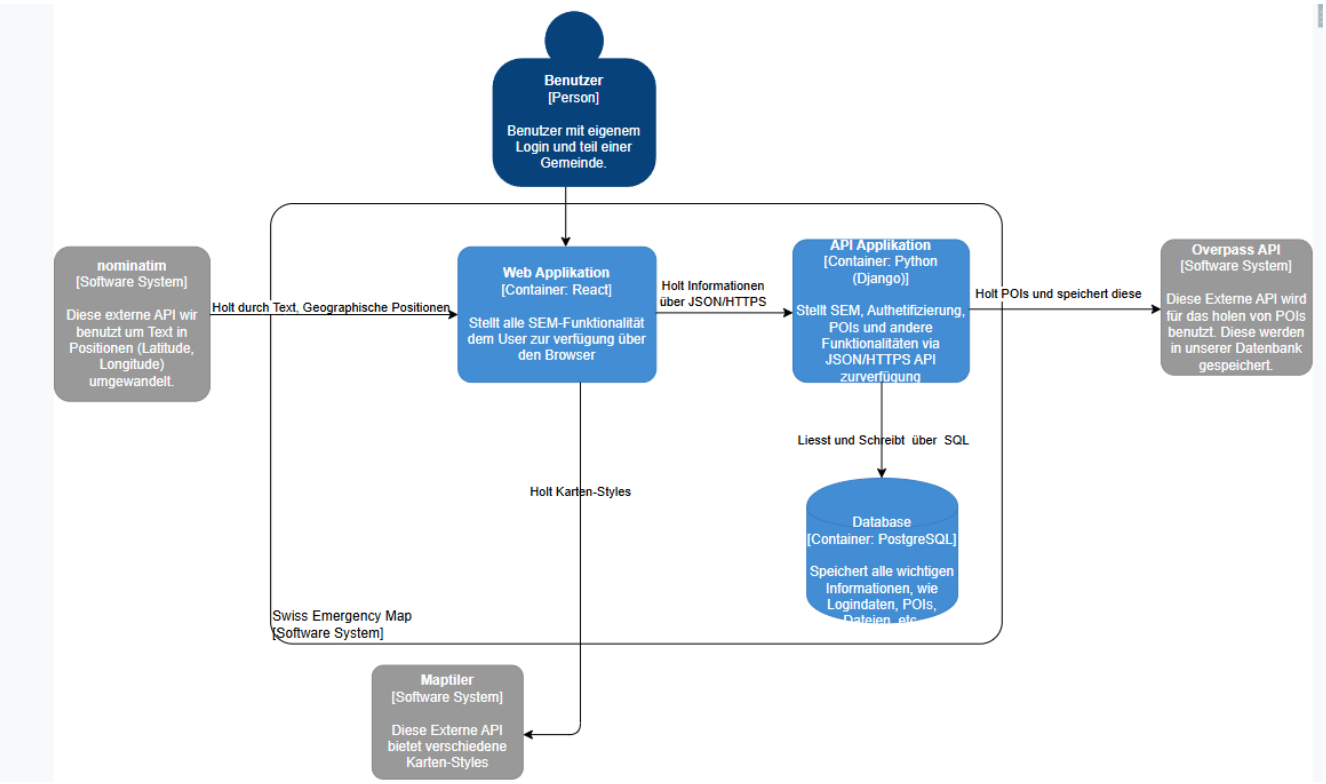

Abbildung 10.3: C4 Container-Diagramm

10.2 Wireframes

Dieser Abschnitt bietet einen umfassenden Überblick über die verfügbaren Ansichten innerhalb der Anwendung, dargestellt durch grundlegende Wireframes als visuelle Hilfsmittel. Jede Untersektion beschreibt die Funktionalität und das Layout der wichtigsten Komponenten im Detail.

10.2.1 Startseite

Die Startseite ist die erste Benutzeroberfläche, die Nutzer nach dem Einloggen sehen. Sie zeigt eine interaktive Karte mit allen Points of Interest (POIs). Nutzer können neue POIs ganz einfach über das "+Symbol in der Kartenoberfläche hinzufügen.

Die Navigationsleiste oben enthält den Benutzernamen des Nutzers sowie einen Logout-Button zur Verwaltung des Kontos. Zusätzlich steht ein Suchfeld auf der Karte zur Verfügung, das eine effiziente Suche nach bestimmten Orten ermöglicht.

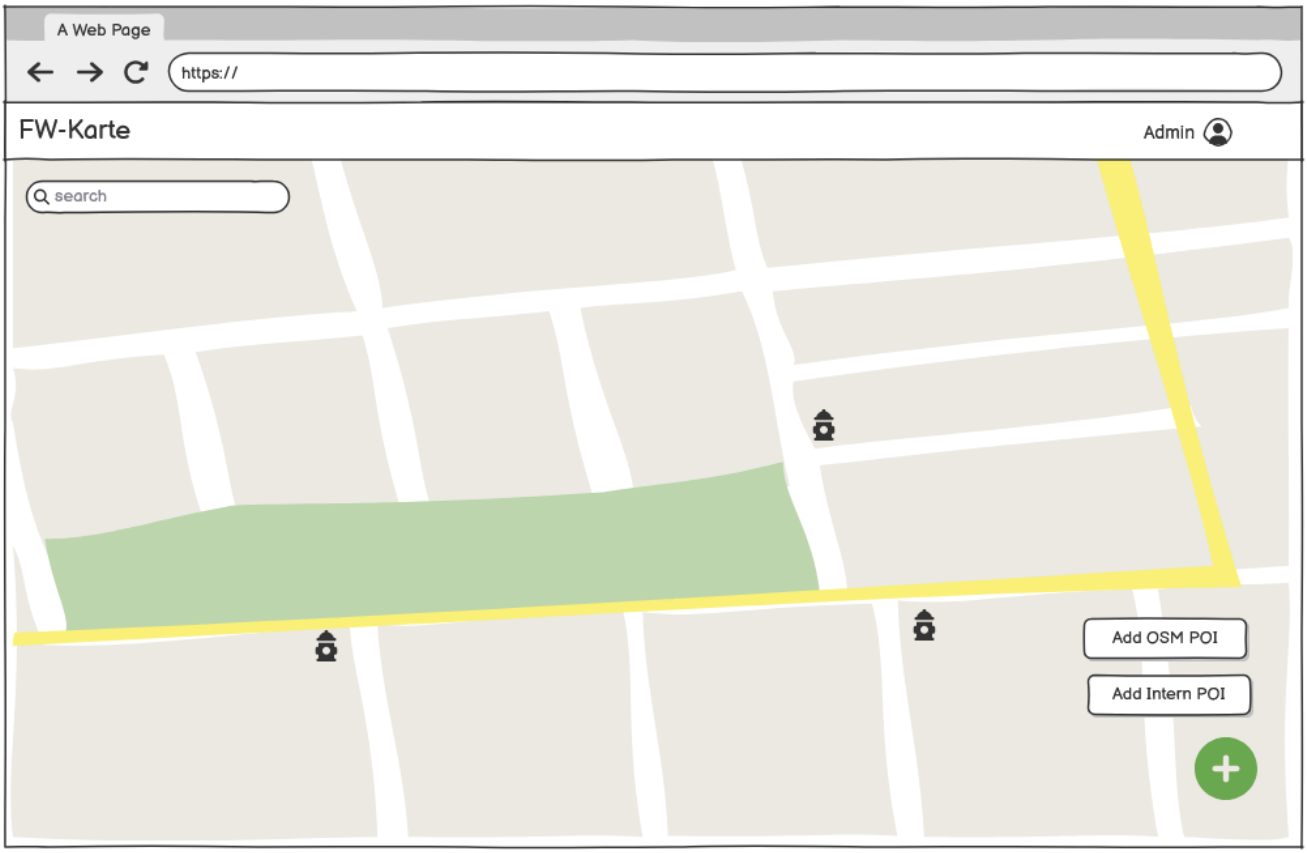

Abbildung 10.4: Wireframe: Startseite

10.2.2 POI hinzufügen (Modal)

Das Modal "POI hinzufügenërmöglicht es Nutzern, neue Points of Interest zu erstellen. Es bietet ein einfaches Formular zur Eingabe der erforderlichen Details. Falls der Inhalt den sichtbaren Bereich des Modals überschreitet, können Nutzer scrollen, um zusätzliche Felder aufzurufen.

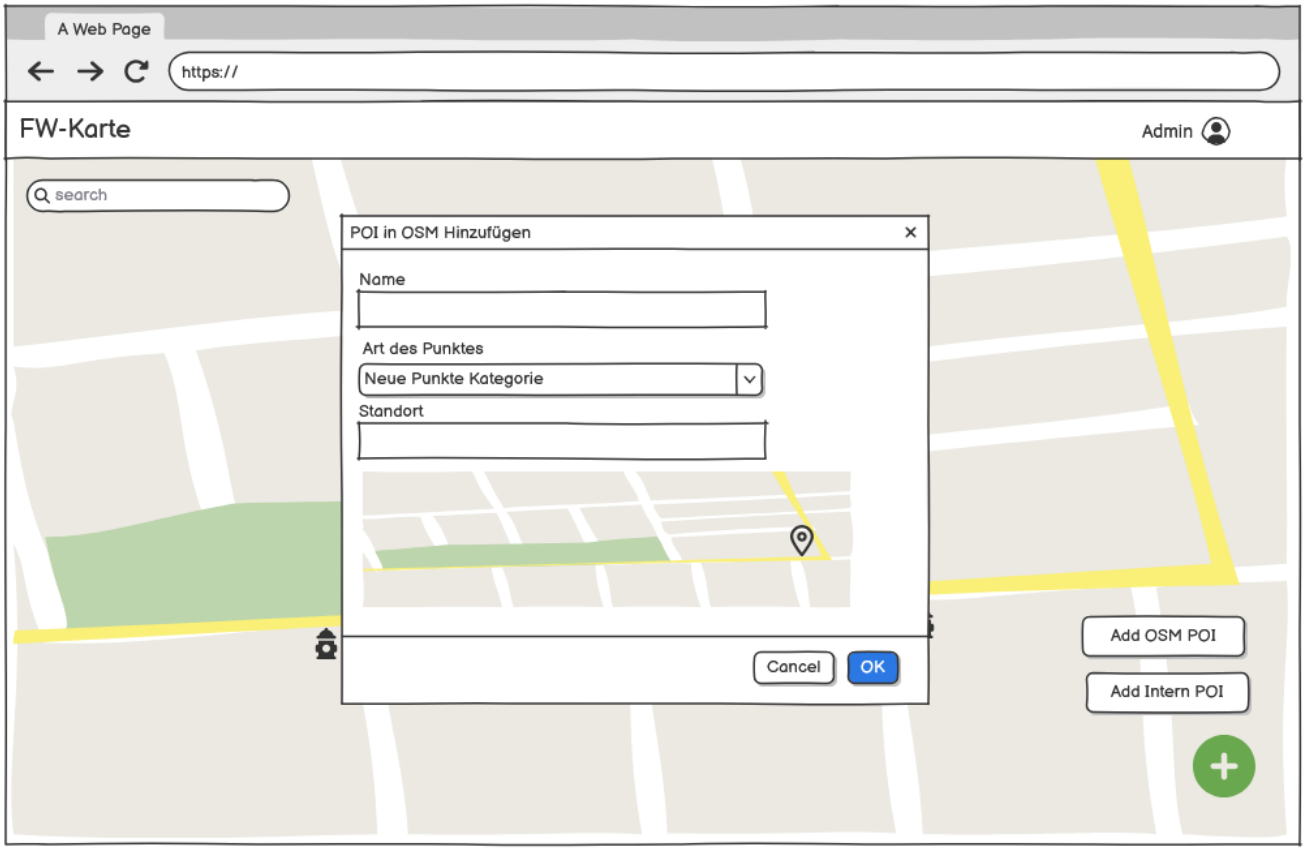

Abbildung 10.5: Wireframe: POI hinzufügen (Modal)

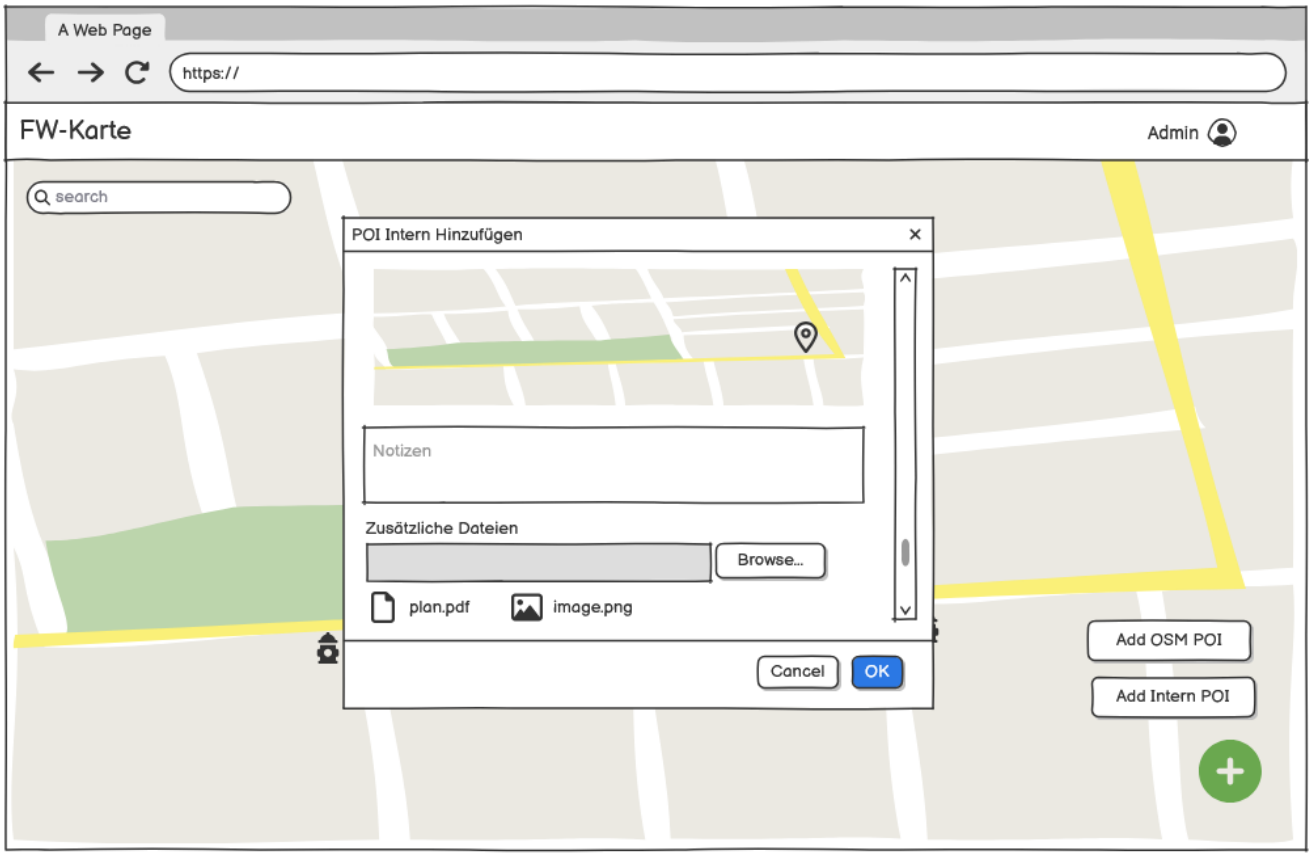

Abbildung 10.6: Wireframe: POI hinzufügen (Modal, gescrollt)

10.2.3 POI-Layer

Das "POI-LayerPanel bietet Nutzern die Möglichkeit, Points of Interest nach Kategorien zu filtern. Nach dem Öffnen zeigt es verschiedene Kategorien an, die Nutzer ein- oder ausschalten können, um die angezeigten POIs anzupassen. Nutzer können ausserdem zwischen lokalen Daten und OpenStreetMap (OSM)-Daten wählen, um maximale Flexibilität zu gewährleisten.

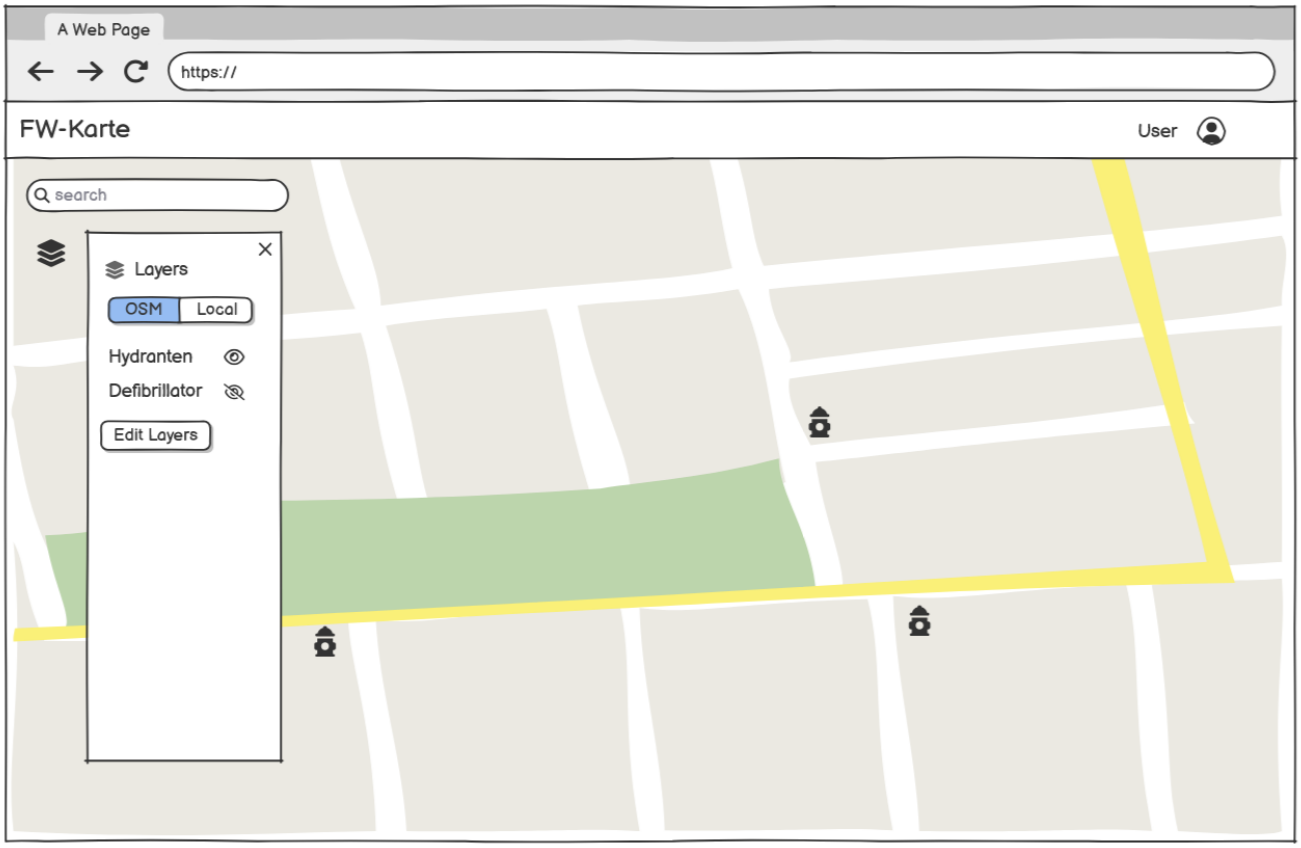

Abbildung 10.7: Wireframe: POI-Layer

10.2.4 Mobile Ansicht

Dieser Abschnitt bietet einen Überblick über die mobile Ansicht der Anwendung und zeigt auf, wie die Benutzeroberfläche und Funktionalität für kleinere Bildschirme optimiert wurden. Das Design stellt eine nahtlose Benutzerfreundlichkeit und Zugänglichkeit auf mobilen Geräten sicher, während es Konsistenz mit der Desktop-Version beibehält und sich gleichzeitig an die Einschränkungen mobiler Layouts anpasst.

Abbildung 10.8: Wireframe: Mobile Startseite

Abbildung 10.9: Wireframe: Mobile Burger-Menü

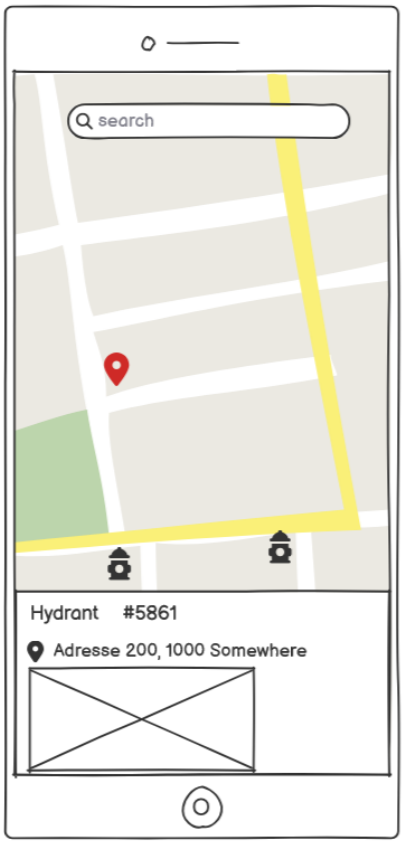

Abbildung 10.10: Wireframe: Mobile POI-Ansicht

Kapitel 11

Implementation und Testing

11.1 Implementation

11.1.1 Browser

Landing-Page

Die Landing-Page dient wie geplant als erste Ansicht nach dem Login. Ein wesentlicher Unterschied zur ursprünglichen Planung ist, dass die Funktion zum Hinzufügen von POIs nicht über das „+“-Symbol auf der Karte verfügbar ist, sondern in der oberen Navigationsleiste integriert wurde. Dort befindet sich ebenfalls der Button für die Gemeindewahl, der erst nach der Erstellung der Wireframes hinzugefügt wurde.

Ein weiteres Feature, das erst später ergänzt wurde, ist die Sprachwahl in der Navigationsleiste, die den Nutzern eine einfache Anpassung der Sprache ermöglicht. Zusätzlich bietet die untere linke Ecke der Seite die Möglichkeit, zwischen verschiedenen Kartenstilen zu wechseln, um die Darstellung der Karte individuell anzupassen.

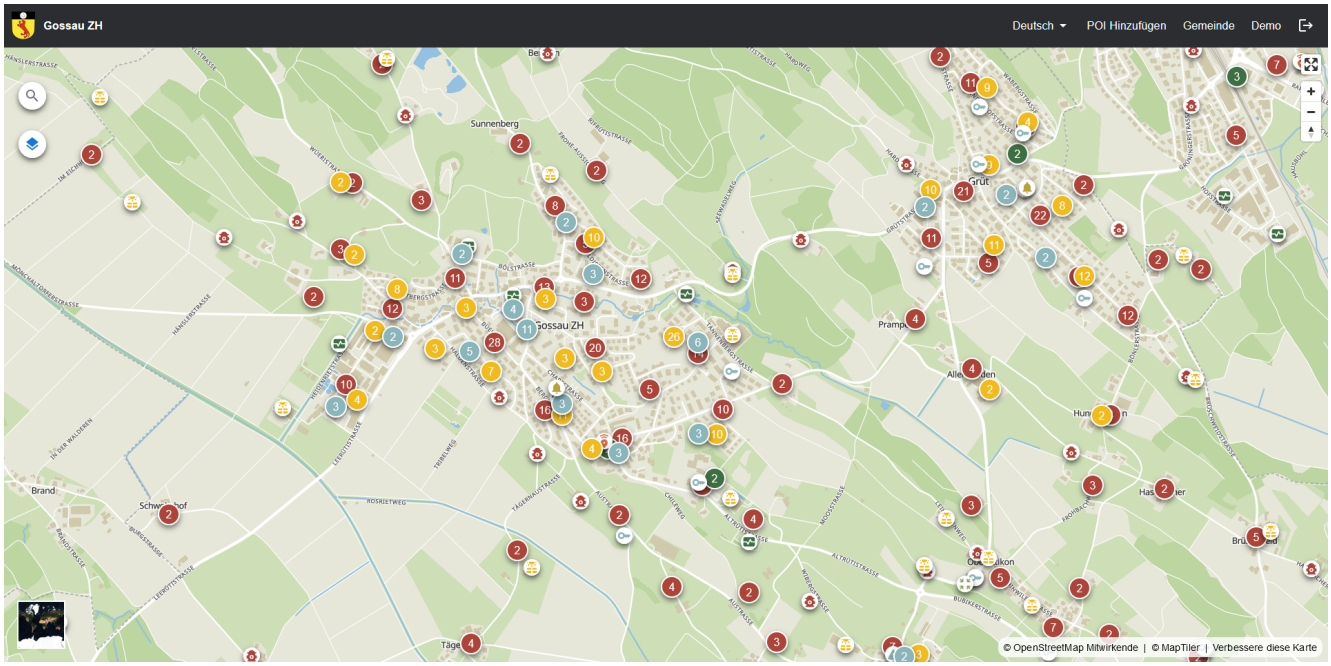

Abbildung 11.1: Implementierte Landing-Page

POI-Hinzufügen

Die Funktionalität zum Hinzufügen von POIs wurde nicht verändert, jedoch hat sich das Design grundlegend weiterentwickelt. Statt eines Modals wurde hierfür eine eigene Seite gestaltet, was insbesondere die mobile Nutzung deutlich erleichtert.

Auf dieser Seite befindet sich die Karte auf der rechten Seite, wodurch ein Scrollen nicht notwendig ist. Links sind die Eingabefelder für den Namen, die Kategorie und die Beschreibung des POIs übersichtlich angeordnet. Zudem gibt es die Möglichkeit, Dateien direkt anzuhängen. Dieses durchdachte Layout sorgt für eine intuitive und benutzerfreundliche Bedienung, sowohl auf Desktop- als auch auf mobilen Geräten.

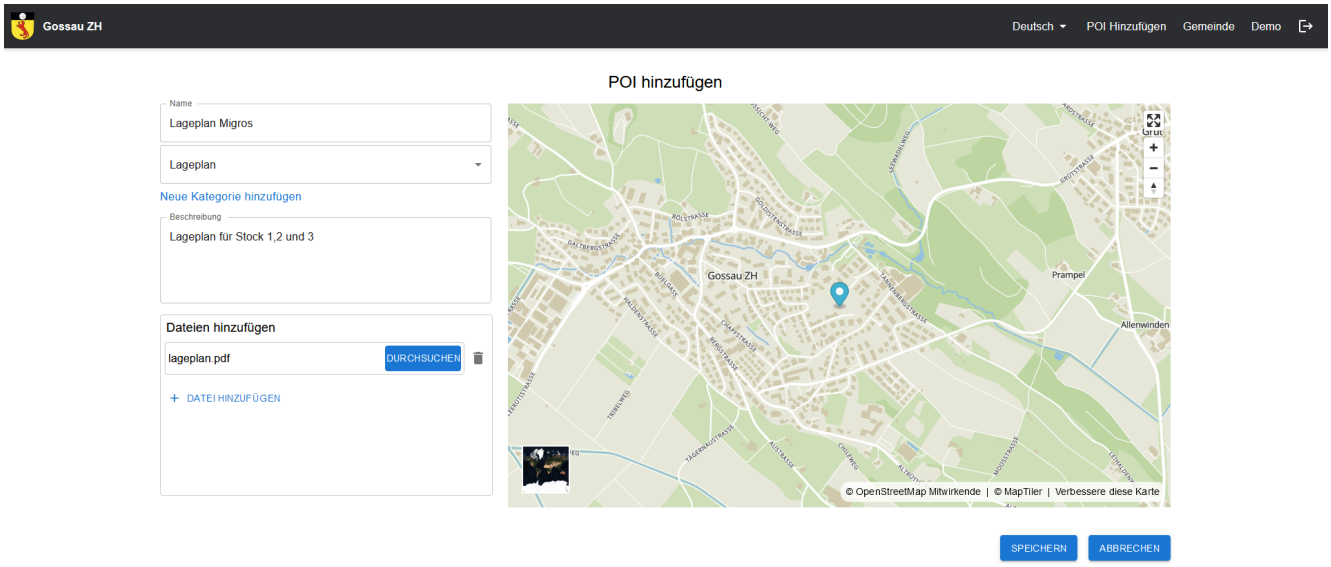

Abbildung 11.2: Implementierte POI-Hinzufügen

Gemeinde Seite

Diese Seite war ursprünglich nicht in den Wireframes vorgesehen. Das Ziel ist es, seine eigene Gruppe zu erstellen. Neben dem Namen der Gruppe kann man ein individuelles Icon hochladen, das später oben links neben dem Namen in der Navigation angezeigt wird.

Wenn man bereits Teil einer Gruppe ist, kann man Leute hinzufügen oder entfernen.

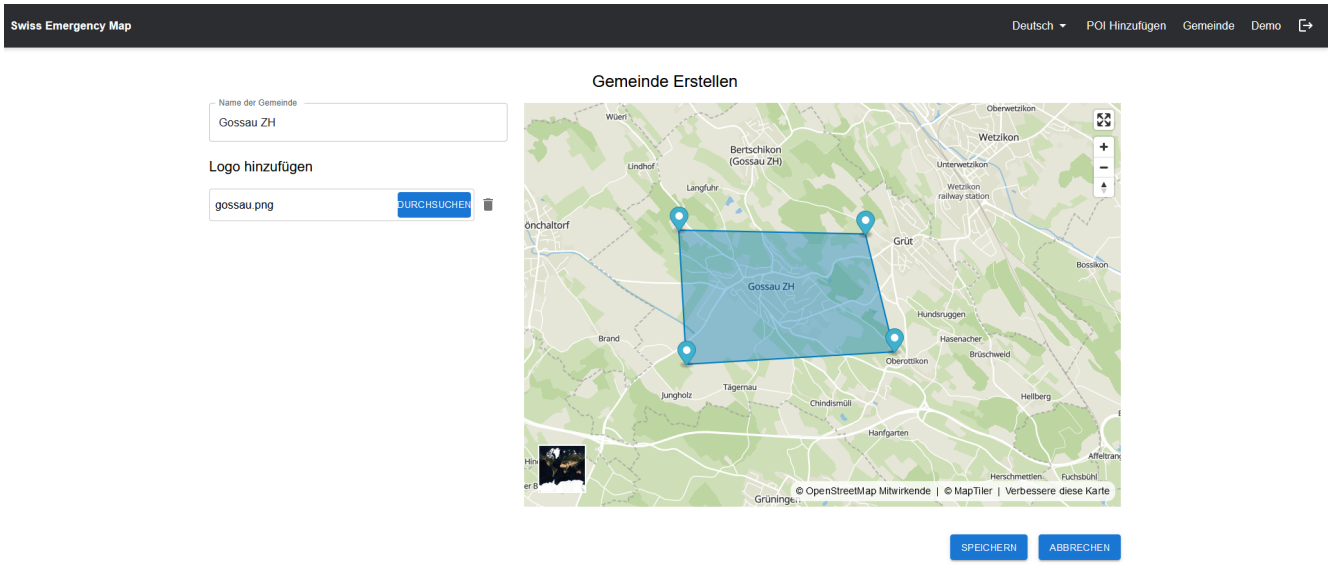

Abbildung 11.3: Implementierte Gemeinde erstellen Seite

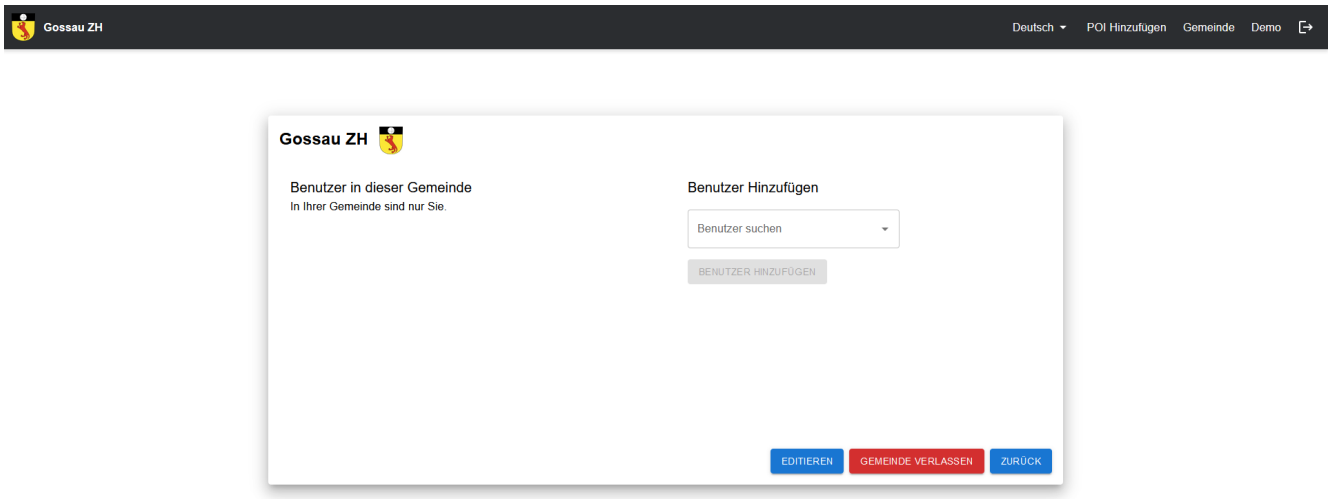

Abbildung 11.4: Implementierte Gemeinde bearbeiten Seite

11.1.2 Mobile

Ursprünglich war geplant, zusätzlich eine Compendio-App mit React Native zu entwickeln. Aus Zeitgründen ist diese jedoch nicht zustande gekommen. Stattdessen wurde die bestehende Web-App für die mobile Nutzung optimiert. In diesem Kapitel werden die implementierten Seiten präsentiert.

Landing-Page

Der grösste Unterschied zum Wireframe besteht darin, dass die Suchleiste nicht zuoberst platziert ist, sondern bei Bedarf aufgeklappt werden kann. Zusätzlich wurde die Funktion zum Hinzufügen von POIs ins Burger-Menü integriert. Ebenfalls neu ist die Möglichkeit, die Sprache direkt auszuwählen.

Abbildung 11.5: Mobile Landing-Page

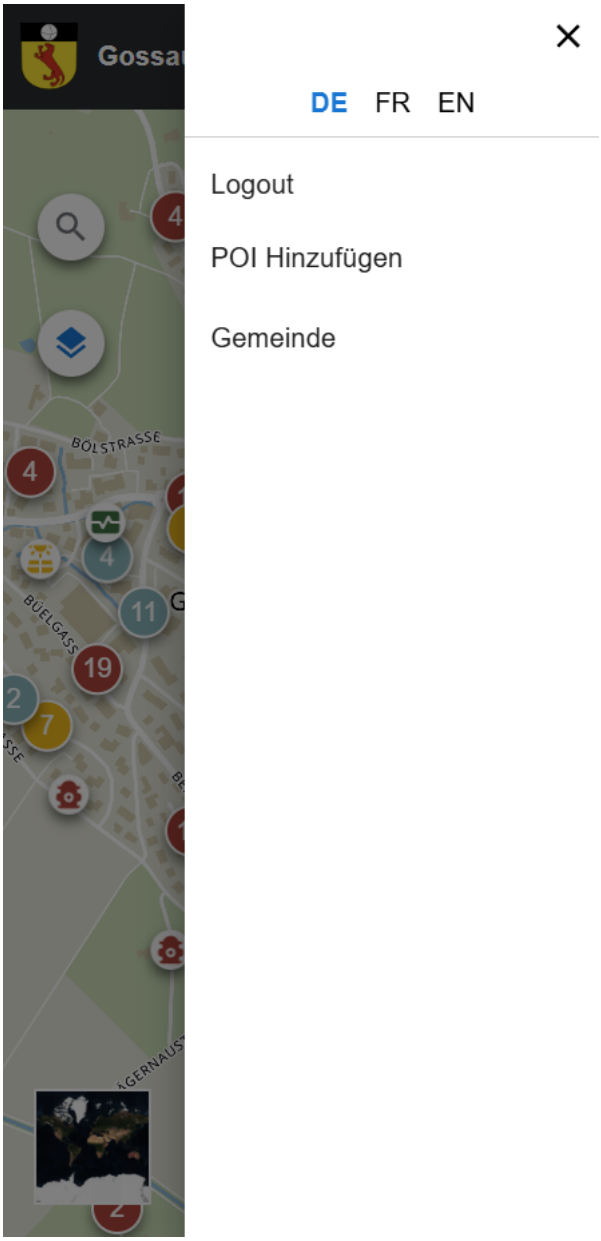

Abbildung 11.6: Mobile-Burger

Die Detailansicht, wenn man auf einen POI drückt, ist oben links und nicht unten.

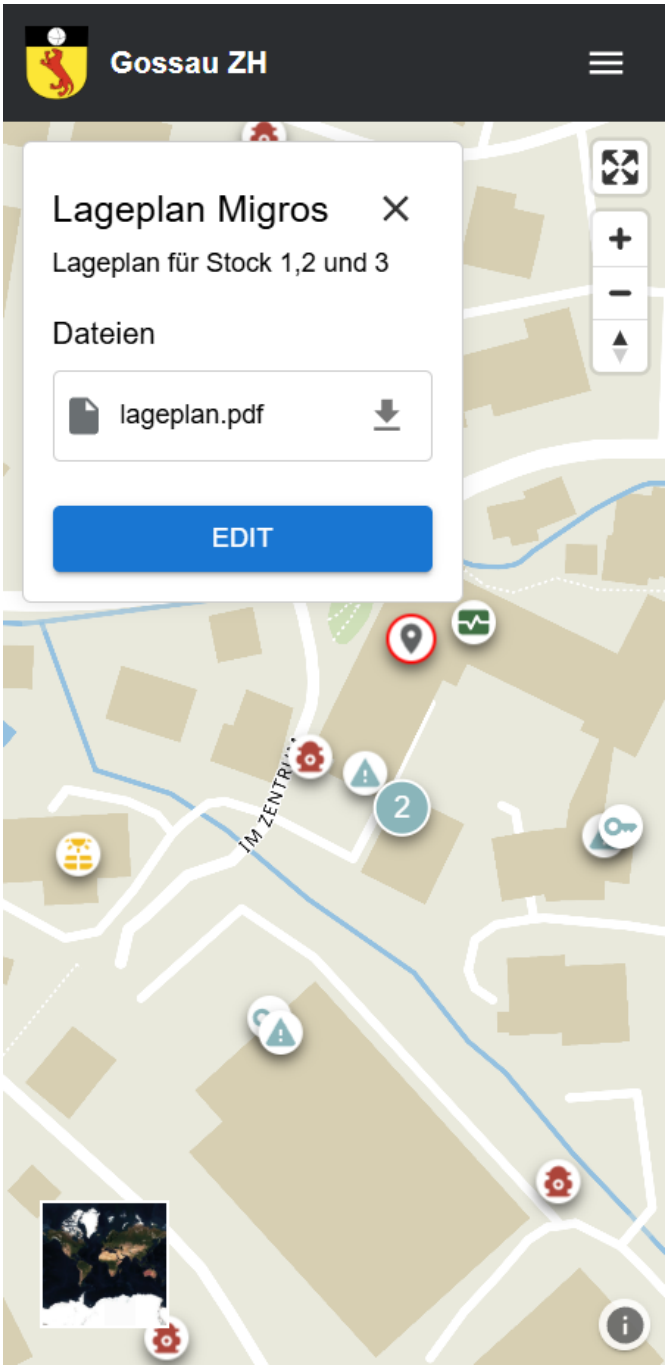

Abbildung 11.7: Mobile-Detail-Ansicht

POI-Hinzufügen

Diese Funktionalität war ursprünglich nicht für die mobile Version vorgesehen. Aufgrund des wertvollen Feedbacks von Christian Nüssli wurde sie dennoch für den mobilen Gebrauch optimiert.

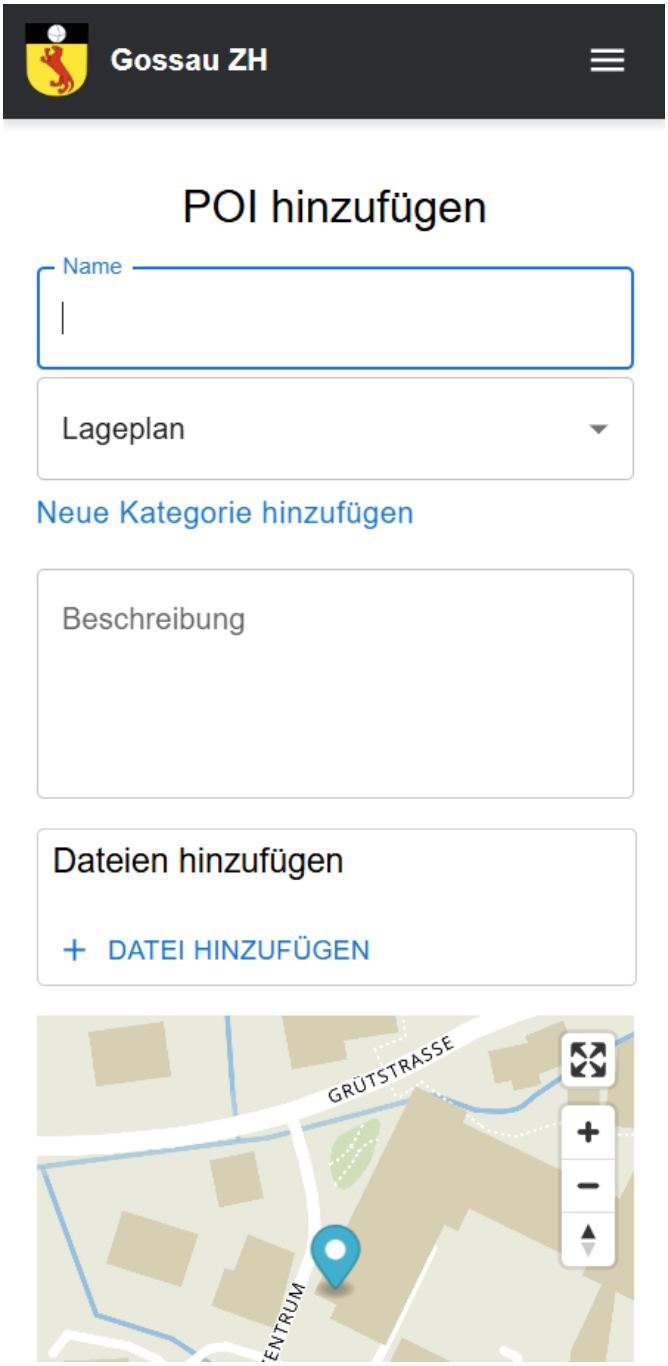

Abbildung 11.8: Mobile-Poi-Hinzufügen

Gemeinde Seite

Hier wird die Gemeinde-Seite dargestellt: zunächst beim Erstellen einer neuen Gemeinde und danach beim Bearbeiten einer bestehenden.

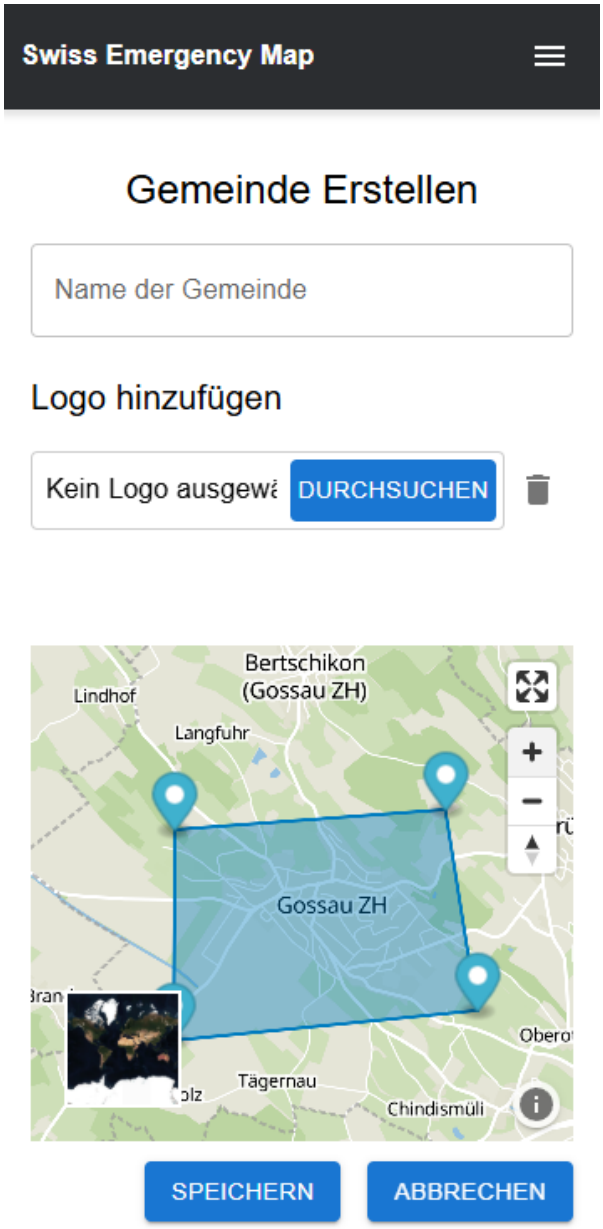

Abbildung 11.9: Mobile-Gemeinde-Hinzufügen

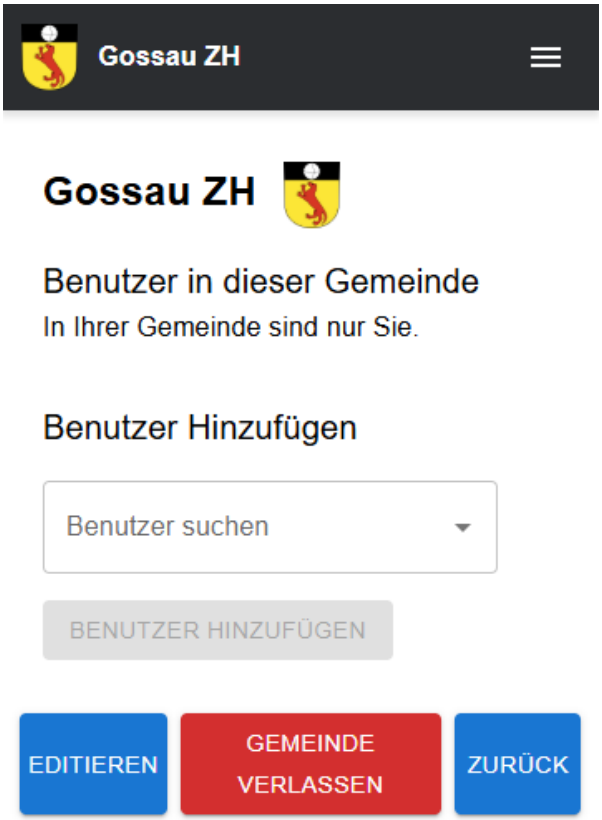

Abbildung 11.10: Mobile-Gemeinde-Bearbeiten

11.1.3 Clustering

Wie im Kapitel 4.5.2 beschrieben, wurde für das Clustering der Punkte die Bibliothek Supercluster verwendet. Eine zentrale Herausforderung bestand darin, die optimalen Breakpoints festzulegen, bei denen die Punkte zu Clustern zusammengefasst werden. Dazu wurden verschiedene Werte manuell getestet, auf dieser Basis wurden die finalen Breakpoints definiert.

Ein weiteres Problem trat beim Ein- und Ausblenden einzelner Kategorien auf: Die Supercluster wurden in einigen Fällen nicht korrekt aktualisiert und blieben teilweise sichtbar. Dieses Verhalten erforderte zusätzliche Anpassungen, um eine zuverlässige Darstellung sicherzustellen.

11.2 Automatisierte und Manuele Tests

Informationen zu den automatisierten und manuellen Test kann man im Kapitel 13.4 lesen

11.3 Bewertung der nicht-funktionalen Anforderungen

NFR-01-Nur die Anwendung hat direkten Zugriff auf das Dateisystem und die Datenbank

Die Anwendung läuft in einem Docker-Container, der den Zugriff auf das Dateisystem und die Datenbank ausschliesslich auf die Anwendung beschränkt. Der Port für Django ist geöffnet, um die notwendige Kommunikation zu ermöglichen, während der Zugriff auf andere Dienste im Container eingeschränkt ist.

Diese Anforderung ist teilweise erfüllt.

NFR-02-Die Anwendung verwendet Docker-Container für eine vereinfachte Bereitstellung

Die Anwendung läuft in Docker-Containern, was eine vereinfachte Bereitstellung und Verwaltung der Softwareumgebung ermöglicht.

Diese Anforderung ist teilweise erfüllt.

NFR-03-POI-Daten werden innerhalb von maximal 3 Sekunden geladen

Diese Anforderung wurde mithilfe des Google Chrome Performance Dev-Tools analysiert. Die Analyse ergab, dass die NFR bei einer Anzahl von bis zu ungefähr 3000 POIs erfüllt ist. Wenn mehr als 3000 POIs geladen werden, überschreitet die Ladezeit die festgelegten 3 Sekunden. Da in der Praxis normalerweise nicht so viele POIs gleichzeitig geladen werden müssen, wird diese Anforderung als teilweise erfüllt gewertet.

Diese Anforderung ist teilweise erfüllt.

NFR-04-Die Grösse der Website darf 500 MB nicht überschreiten

Die Grösse der Website wurde mithilfe der Entwicklertools in Firefox analysiert. Auch bei der Verwendung einer hohen Anzahl von POIs überschreitet die Gesamtgrösse der Website 200 MB nicht.

Diese Anforderung ist teilweise erfüllt.

NFR-05-Sensible Daten (z. B. Passwörter, API-Tokens) werden verschlüsselt gespeichert

Die Sicherheit sensibler Daten, wie Passwörter und API-Tokens, wird in dieser Anwendung durch die integrierten Funktionen von Django gewährleistet. Django verwendet standardmässig sichere Hashing-Algorithmen, um Passwörter zu verschlüsseln, bevor sie in der Datenbank gespeichert werden.

Diese Anforderung ist erfüllt.

NFR-06-Nur autorisierte Benutzer dürfen POI- und Kartendaten ansehen oder ändern

Die Sicherheit der Anwendung wird durch eine Kombination aus React Auth im Frontend und den integrierten Authentifizierungs- und Autorisierungsfunktionen von Django im Backend gewährleistet.

Diese Anforderung ist erfüllt.

NFR-07-Die Anwendung darf bei einem Fehler nicht abstürzen, sondern muss stattdessen einen Fehlerstatus zurückgeben

Diese Anforderung wurde durch umfassendes Error-Handling im Backend und Frontend erfüllt. Im Backend werden alle Ausnahmen abgefangen, und es werden entsprechende HTTP-Statuscodes zurückgegeben. Im Frontend werden Fehlerstatus durch benutzerfreundliche Fehlermeldungen ergänzt, um Abstürze zu vermeiden.

Diese Anforderung ist erfüllt.

NFR-08-Alle Benutzereingaben müssen validiert werden, um die Datenintegrität sicherzustellen

Die Validierung von Benutzereingaben erfolgt sowohl im Backend mit Django als auch im Frontend mit React. Django überprüft, ob die Eingaben den definierten Modellen entsprechen und wirft Fehler bei ungültigen Daten, um sicherzustellen, dass nur korrekte Informationen in die Datenbank gelangen. Im Frontend wird ebenfalls eine Validierung durchgeführt, um Benutzer vor der Eingabe fehlerhafter Daten zu warnen und potenzielle JavaScript-Injection-Angriffe zu verhindern. Diese duale Validierung gewährleistet die Datenintegrität und schützt die Anwendung vor schädlichen Eingaben.

Diese Anforderung ist erfüllt.

NFR-09-Die Benutzeroberfläche ist intuitiv

Die Benutzeroberfläche wurde mit React und Material-UI (MUI) [19] erstellt, wodurch ein modernes und benutzerfreundliches Design gewährleistet wird.

Diese Anforderung ist erfüllt.

NFR-10-Die Benutzeroberfläche ist für Smartphones optimiert

Die Optimierung für Smartphones wurde durch den Einsatz von responsivem Design mit CSS-Media-Queries und den flexiblen Komponenten von Material-UI erreicht. Die Darstellung und Navigation der Benutzeroberfläche wurden mithilfe der mobilen Visualisierungstools von Google Chrome analysiert.

Diese Anforderung ist erfüllt.

NFR-11-Die Anwendung enthält automatisierte Tests für alle kritischen Funktionen

Automatisierte Tests wurden mit Tools wie Pytest für das Backend und Cypress für das Frontend erstellt. Die Testergebnisse zeigen, dass alle kritischen Funktionen fehlerfrei funktionieren.

Diese Anforderung ist erfüllt.

Kapitel 12

Resultat und Zukünftige Entwicklung

12.1 Resultate

12.1.1 Login

Das Login wurde mithilfe von Django und JWT-Tokens umgesetzt. Diese Lösung bietet zahlreiche Vorteile. Einer der wichtigsten Aspekte ist die einfache Integration, da JWT-Tokens leicht in bestehende Systeme eingebunden werden können. Darüber hinaus ermöglicht die Nutzung von JWT eine sichere Authentifizierung, da die Tokens verschlüsselt sind und Informationen zu Benutzersitzungen enthalten. Ein weiterer Vorteil ist die Skalierbarkeit: JWT-Tokens können serverunabhängig überprüft werden, wodurch die Last auf dem Server reduziert und die Performance verbessert wird. Zudem sind JWT-Tokens flexibel einsetzbar, da sie nicht nur für Authentifizierung, sondern auch für Autorisierungsprozesse und die Übertragung sicherer Daten verwendet werden können.

12.1.2 Kernfunktionen der Anwendung

- **Hinzufügen von eigenen POIs:** Die wichtigste Funktion ist das Erstellen eigener POIs, welche zusätzlich mit Dateien versehen werden können. Dies ermöglicht eine flexible und individuelle Nutzung.
- **Schnelle Ladezeiten:** Dank optimierter Datenabfragen und effizienter Verarbeitung bleiben die Ladezeiten der Anwendung minimal.
- **Datenimport von OpenStreetMap (OSM):** Die OSM-Daten werden über einen Migrationsbefehl in Django integriert, der täglich auf dem Testserver ausgeführt wird. Dabei erfolgt ein API-Aufruf über die Overpass API, um aktuelle Geodaten abzurufen.

12.1.3 Gemeinde-Funktionalität

Eigene Gemeinden: Nutzer können nur die POIs und Dateien sehen, die ihrer jeweiligen Gemeinde zugeordnet sind. Der Gemeindename wird in der Navigationsleiste angezeigt, ergänzt durch ein individuell festlegbares Icon.

12.1.4 Clustering

Zur Verbesserung der Übersichtlichkeit werden die POIs mithilfe von Supercluster gruppiert. Dies sorgt für eine klarere Darstellung, insbesondere bei vielen Punkten auf der Karte.

12.1.5 Design und Benutzerfreundlichkeit

Die Icons stammen aus der Material-UI-Bibliothek (MUI). Für eigens erstellte Kategorien werden einzigartige Icons verwendet, um eine einfache Unterscheidung zu ermöglichen. Die Kombination dieser Funktionen und Technologien gewährleistet eine benutzerfreundliche, performante und individuell anpassbare Anwendung.

12.2 Zukünftige Entwicklung

In diesem Kapitel werden Funktionen und Optimierungen, die in der Zukunft umgesetzt oder verbessert werden könnten, um die Anwendung noch benutzerfreundlicher und leistungsfähiger zu gestalten.

12.2.1 Passwort-Reset per E-Mail

Eine Funktion zum Zurücksetzen des Passworts über die Eingabe der registrierten E-Mail-Adresse wäre eine sinnvolle Ergänzung. Dies würde die Benutzerfreundlichkeit und Sicherheit erhöhen, insbesondere für den Fall, dass ein Nutzer sein Passwort vergisst.

12.2.2 Kategorien mit eigenen Icons

Eine Erweiterung der Kategorien mit individuell gestalteten Icons würde die visuelle Unterscheidbarkeit der POIs auf der Karte weiter verbessern. Dies könnte insbesondere bei komplexeren Anwendungsfällen mit vielen Kategorien von grossem Vorteil sein.

12.2.3 Unterstützung für verschiedene Kartenstile

Durch die Integration von unterschiedlichen Kartenstilen könnten Nutzer die Darstellung der Karte individuell anpassen, zum Beispiel zwischen einer klassischen Ansicht, einer Satellitenansicht oder einem minimalistischen Stil wechseln.

12.2.4 Mehrere Kategorien pro POI

Eine weitere Funktion, die geplant ist, ist die Möglichkeit, einem POI mehrere Kategorien zuzuordnen. Dies würde die Flexibilität und Genauigkeit der POI-Daten erhöhen, indem sie mehreren thematischen Gruppierungen gleichzeitig zugeordnet werden können.

12.2.5 Adressen mithilfe von Nominatim hinzufügen

Die Integration der Nominatim API würde es ermöglichen, Adressen direkt bei der Erstellung eines POIs hinzuzufügen. Dadurch könnte der Standort eines POIs einfacher und präziser eingegeben werden, ohne die Karte manuell durchsuchen zu müssen.

12.2.6 POIs aus Planet.osm abrufen

Die Nutzung der Planet.osm-Datenbank könnte die Basis der POI-Daten erheblich erweitern. So könnten bereits existierende Daten aus OpenStreetMap automatisch integriert werden, um die manuelle Eingabe von Standardinformationen zu reduzieren.

12.2.7 POIs im Local Storage

Ein geplantes Feature ist die Möglichkeit, vertrauliche POIs im Local Storage des Browsers zu speichern. Diese Funktion dient dazu, sensible Daten vorübergehend lokal auf dem Gerät des Nutzers abzulegen, ohne sie auf den Server hochzuladen. Dies ist besonders nützlich, wenn Nutzer offline arbeiten oder ihre Daten vorübergehend speichern möchten, bevor sie eine stabile Verbindung haben, um diese später zu synchronisieren. Dabei wird sichergestellt, dass die Daten im Local Storage verschlüsselt und vor unbefugtem Zugriff geschützt sind.

12.2.8 Dateien Liste

Eine weitere Verbesserung betrifft die Darstellung hochgeladener Dateien. Neben der bisherigen Ansicht soll eine Listenansicht implementiert werden, um die Dateien übersichtlicher anzuzeigen. In dieser Ansicht werden zusätzliche Informationen wie der Dateiname, das Hochladedatum und die Dateigrösse bereitgestellt. Diese Alternative zur bestehenden Darstellung ist darauf ausgelegt, die Verwaltung hochgeladener Dateien zu erleichtern und insbesondere bei einer grossen Anzahl von Dateien für mehr Übersichtlichkeit zu sorgen. Die Listenansicht könnte zudem mit Funktionen wie Sortieren und Filtern ausgestattet werden, um die Benutzerfreundlichkeit weiter zu erhöhen.

12.2.9 Einladungslink zur Gruppe

Ein weiteres geplantes Feature ist die Möglichkeit, Einladungslinks für Gruppen zu generieren. Diese Funktion soll es den Nutzern erleichtern, neue Mitglieder zu einer Gruppe einzuladen, ohne manuell Benutzer hinzufügen zu müssen.

Mit dieser Funktionalität könnte ein Gruppenadministrator einen individuellen Link erstellen, der an potenzielle Gruppenmitglieder weitergegeben werden kann. Der Einladungslink enthält alle notwendigen Informationen, um den Nutzer direkt der entsprechenden Gruppe zuzuordnen. Nach dem Öffnen des Links wird der eingeladene Nutzer entweder aufgefordert, sich anzumelden oder ein Konto zu erstellen, sofern er noch keines besitzt. Anschliessend wird der Nutzer automatisch der gewünschten Gruppe hinzugefügt.

12.2.10 Fehlerbehebung und Optimierung

Eine wichtige Aufgabe für die Zukunft besteht darin, die noch offenen Bugs im GitLab-Projekt zu beheben. Die kontinuierliche Abarbeitung der gemeldeten Fehler wird dazu beitragen, die Stabilität und Qualität der Anwendung zu verbessern.

Kapitel 13

Qualitätsicherung

13.1 Qualitätssicherungs tools

13.1.1 Linter

ESLint
Für den TypeScript-Code wird ESLint verwendet. Damit ESLint in WebStorm funktioniert, muss ESLint zunächst installiert und in den Einstellungen aktiviert werden. Die Installation erfolgt über npm.
Pylint
Für den Python-Code wird Pylint genutzt. Um Pylint in WebStorm zu verwenden, muss es installiert und in den Einstellungen konfiguriert werden. Die Installation erfolgt über pip.

13.1.2 Guidelines

Definition of Done

- Der gesamte Code zur User Story wurde implementiert und entspricht den Coding-Standards.
- Der Code wurde vom anderen Teammitglied peer-reviewed und genehmigt.
- Relevante Unit-Tests wurden geschrieben und erfolgreich bestanden.
- Die Build-Pipeline wurde erfolgreich durchlaufen
- Die Funktionalität entspricht den funktionalen und nicht-funktionalen Anforderungen
- Alle mit der Aufgabe verknüpften Unteraufgaben wurden gelöst.
- Alle notwendigen Dokumentationen wurden erstellt und aktualisiert.

13.1.3 Git-Branching Merges

Um Fehler zu vermeiden, ist es nicht erlaubt, direkt auf den Main-Branch zu pushen. Nachdem ein Teammitglied die Funktion implementiert hat, kann ein Merge-Request auf GitLab erstellt und einem Reviewer zugewiesen werden. Der Reviewer überprüft die Änderungen und stellt sicher, dass sie der Definition of Done entsprechen. Wenn alles in Ordnung ist, wird der Merge-Request vom Reviewer genehmigt.

13.2 Environment

13.3 CI/CD

13.3.1 Workflow

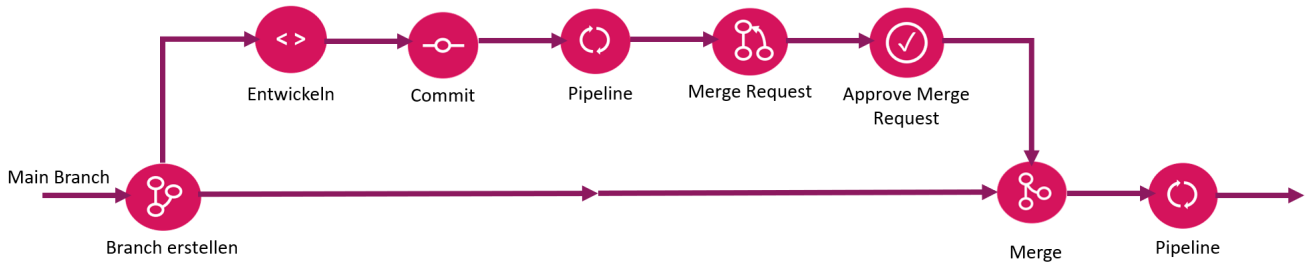

Abbildung 13.1: Workflow

13.3.2 Gitlab Pipeline

Frontend Pipeline

Die Frontend-Pipeline besteht aus vier Hauptphasen: Install, Build, Test und Lint. In der Install-Phase werden die Abhängigkeiten mit npm install installiert. Anschliessend wird in der Build-Phase das Projekt mit npm run build erstellt. In der Test-Phase werden mit einem spezifischen Cypress-Image die Tests ausgeführt. Zunächst installiert die Pipeline die Abhängigkeiten mit npm ci, startet den Server im Hintergrund und wartet, bis dieser bereit ist. Dann werden die Cypress-Tests mit dem Firefox-Browser ausgeführt. Abschliessend überprüft die Lint-Phase den Code auf Stil- und Syntaxfehler mithilfe von ESLint, um die Codequalität sicherzustellen.

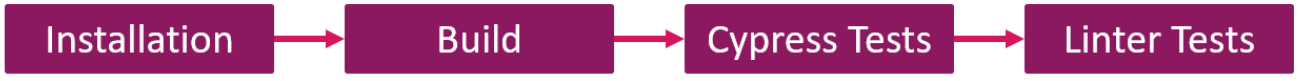

Abbildung 13.2: Frontend Pipeline

Backend Pipeline

Die Backend-Pipeline besteht aus drei Hauptphasen: Build, Test und Lint. In der Build-Phase wird mit

python manage.py collectstatic –noinput das statische Asset-Management durchgeführt, und die statischen Dateien werden als Artefakte gespeichert. In der Test-Phase werden die Datenbankmigrationen mit python manage.py migrate angewendet und die Tests mit python manage.py test ausgeführt, um die Funktionalität der Anwendung zu überprüfen

Abbildung 13.3: Backend Pipeline

13.4 Teststrategie

In diesem Projekt wird ein feature-driven Entwicklungsansatz verwendet, der die Entwicklung von Features vor dem Testen betont. Diese Methodik ermöglicht es, sich auf die Bereitstellung spezifischer Funktionalitäten zu konzentrieren, die den Anforderungen der Benutzer entsprechen.

13.4.1 Cypress

Zur Gewährleistung der Qualität der Front-End-Anwendungen wird Cypress für End-to-End-Tests eingesetzt. Cypress ist ein leistungsfähiges Test-Framework, das die Erstellung und Ausführung von Tests direkt im Browser ermöglicht und Echtzeit-Feedback liefert. Mit Cypress können Benutzerinteraktionen simuliert und überprüft werden, ob sich die Anwendung in verschiedenen Szenarien wie erwartet verhält.

13.4.2 Python-Tests

Zur Sicherstellung der Qualität des Backends kommen Python-Tests zum Einsatz, insbesondere das unittest-Modul zur Erstellung von Unit-Tests. unittest ist ein integriertes Test-Framework, das die systematische Strukturierung und Durchführung von Tests ermöglicht. Mit diesen Tests kann die Funktionalität einzelner Module und Funktionen des Backends überprüft werden, um sicherzustellen, dass sie wie erwartet arbeiten.

13.5 Kommunikations Tools

Discord
Primär wird Discord als Kommunikationsmittel genutzt. Es ermöglicht die Kommunikation über Sprach- und Textchat und wird ausserdem verwendet, um Bildschirme zu teilen und Probleme zu besprechen.
Teams
Teams wird verwendet, um Informationen mit dem Betreuer auszutauschen und wichtige Meetings mit dem Betreuer sowie einem externen Partner abzuhalten.

Kapitel 14

Projekt Management

14.1 Ressourcen

14.1.1 Team

- Elena Gerber
- Noël Joost

Das Team besteht aus zwei Vollzeitstudierenden im fünften Semester an der OST - Ostschweizer Fachhochschule.

14.1.2 Zeit

Das Projekt begann mit dem Kickoff-Meeting am 13. September 2024 und endet mit der finalen Projektabgabe am 20. Dezember 2024. Es ist vorgesehen, etwa 17 Stunden pro Woche an diesem Projekt zu arbeiten, was insgesamt 34 Stunden pro Woche und 480 Stunden für das gesamte Projekt ergibt. Die Arbeitszeit wird in einem Excel-Dokument erfasst.

14.1.3 Tooling

- Frontend Code editor: Webstorm
- Backend Code editor: Pycharm
- Issue-Tracking: Jira
- Dokumentation: LaTeX
- Dateiaustausch: Teams und Discord
- Kommunikation: Teams und Discord

14.1.4 Hosting

Während der Studienarbeit lief das Backend der Anwendung auf einem Server, der von der OST bereitgestellt wurde. Der bereitgestellte Server war eine virtuelle Maschine, die Linux betrieben wurde. Es waren 2 vCPUs zugewiesen, und der Server verfügte über 4 GB RAM und 50 GB Speicherplatz.

14.2 Prozesse und Meeting

Es wurde eine Kombination aus RUP und Scrum gewählt, die an der OST als Scrum+ bekannt ist.

14.2.1 Sprints

Sprint Planning
Zu Beginn jedes Sprints wurde ein Sprint-Planning-Meeting abgehalten. Der Sprint begann jeweils am Montag nach dem Treffen mit dem Betreuer, wodurch eine Planung basierend auf dem Feedback aus dem letzten Meeting ermöglicht wurde. Das Planning-Meeting fand am ersten Tag des Sprints statt und hatte eine maximale Dauer von 30 Minuten
Daily Scrum
Obwohl kein formelles Daily Scrum stattfand, wurden während des Sprints regelmässig informelle Gespräche geführt, um den Fortschritt zu verfolgen und offene Aufgaben zu klären.
Sprint Review
Am Ende jedes Sprints, vor dem Beginn eines neuen Sprints, fand ein Sprint Review statt. Dieses Meeting wurde online um 09:00 oder 10:00 Uhr abgehalten und umfasste die Teilnahme des Teams, des Betreuers und gelegentlich des externen Partners. Es dauerte etwa eine Stunde, in der die erreichten Ergebnisse präsentiert, Probleme und Verbesserungspotenziale diskutiert sowie die nächsten Schritte für den kommenden Sprint besprochen wurden.
Sprint Retrospective
Nach dem Sprint-Review-Meeting wurde eine Sprint-Retrospektive ohne den Betreuer und den externen Partner durchgeführt. In dieser Retrospektive wurde erörtert, was während des Sprints gut und schlecht gelaufen ist.

14.3 Risiko Management

		Impact				
		Trivial	Minor	Moderate	Major	Extreme
Likelihood	Very Likely	Low 1	Medium 1	High 1	Extreme 1	Extreme 2
	Likely	Low 2	Medium 2	High 2	High 3	Extreme 3
	Moderate	Low 3	Medium 3	Medium 4	High 4	High 5
	Unlikely	Low 4	Low 6	Medium 5	Medium 6	Medium 7
	Rare	Low 5	Low 7	Low 8	Low 9	Low 10

Risk ID:	1
Risk:	Zeitmanagement-Herausforderungen
Description:	Schlechte Planung kann zu Verzögerungen führen und die Einhaltung der Fristen gefährden.
Risk Value:	High 3 Likely / Major
Mitigation:	Einen klaren Projektplan mit wichtigen Terminen erstellen. Den Fortschritt regelmässig überprüfen und offen kommunizieren, wenn es Probleme gibt.

Risk ID:	2
Risk:	Abwesenheiten und Krankheiten
Description:	In einem Team von zwei kann die Abwesenheit eines Mitglieds das Projekt stark belasten, besonders im Winter, wenn mehr Krankheiten auftreten.
Risk Value:	High 2 Likely / Moderate
Mitigation:	So schnell wie möglich kommunizieren, wenn jemand abwesend ist, und sicherstellen, dass beide Teammitglieder alle Aspekte der Anwendung gut verstehen.

Risk ID:	3
Risk:	Herausforderungen beim Deployment
Description:	Beide Teammitglieder haben wenig Erfahrung mit Deployment-Prozessen, was zu Problemen führen kann.
Risk Value:	High 3 Likely / Major
Mitigation:	Zeit einplanen, um sich in Deployment-Praktiken und -Tools einzuarbeiten.

Risk ID:	4
Risk:	Leistungsprobleme
Description:	Die Anwendung könnte bei grossen Datensätzen oder hohen Rendering-Anforderungen langsamer werden.
Risk Value:	High 4 Moderate / Major
Mitigation:	Den Code optimieren und die Anwendung unter verschiedenen Bedingungen testen, um die Leistung zu verbessern.

Risk ID:	5
Risk:	Datenqualität und -genauigkeit
Description:	Ungenaue oder veraltete OSM-Daten könnten zu schlechten Benutzererfahrungen oder falschen Informationen führen.
Risk Value:	Medium 4 Moderate/ Moderate
Mitigation:	Daten cachen und den Benutzern die Möglichkeit geben, Ungenauigkeiten zu melden.

Risk ID:	6
Risk:	Teamdynamik und Kommunikation
Description:	Auch wenn die Teammitglieder bereits an anderen Projekten zusammengearbeitet haben, können Missverständnisse auftreten.
Risk Value:	Low 6 Unlikely / Minor
Mitigation:	Regelmässig kommunizieren und gegenseitigen Respekt zeigen.

14.4 Projektplan

14.4.1 Milestones

Name	Geplantes Datum	Ziel
Projektplan	16.09.2024	Erstellen eines vollständigen Projektplans.
UMap	23.09.2024	Analyse und Test des UMap-Ansatzes als mögliche Lösung.
Prototyp	14.10.2024	Entwicklung eines funktionalen Prototyps.
Authentifizierung-Epic	28.10.2024	Implementierung der Benutzeranmeldung und -Registrierung.
Karte-Epic	11.11.2024	Fertigstellung der Kartendarstellung und -interaktion.
Group-Epic	25.11.2024	Implementierung der POI-Clustering-Funktionalität und des Dateimanagements.
Abstract Abgabe	16.12.2024	Einreichung des Abstracts für die Broschüre.
Projekt Abgabe	20.12.2024	Finalisierung und Übergabe des gesamten Projekts.

Tabelle 14.1: Projektmeilensteine und Ziele

Dieses Projekt basiert auf SCRUM+, das eine Kombination zwischen SCRUM und RUP ist. Dieses Projekt ist in vier Phasen unterteilt.

Phase 1: Konzeption (Inception)
Die erste Phase des Projekts konzentriert sich auf die Schaffung eines stabilen Fundaments. Dazu gehört die Evaluierung geeigneter Tools und Technologien, die für die Umsetzung notwendig sind. Ebenso werden potenzielle Risiken identifiziert und bewertet, um frühzeitig Gegenmassnahmen zu planen. Diese Phase legt den Grundstein für alle weiteren Schritte im Projektverlauf.
Phase 2: Entwurf (Elaboration)
Die Entwurfsphase dient dazu, die Funktionalen-Anforderungen (FRs) und Nicht-Funktionalen-Anforderungen (NFRs) klar zu definieren. In dieser Phase wird ein Prototyp erstellt, der die grundlegenden Funktionalitäten und das Design der Karte demonstriert. Ausserdem wird eine Continuous Integration/Continuous Deployment (CI/CD) Pipeline eingerichtet, um eine effiziente Entwicklung und Bereitstellung des Projekts sicherzustellen.
Phase 3: Konstruktion (Construction)
Diese Phase konzentriert sich auf die eigentliche Entwicklung der Anwendung. Dabei sollen alle definierten Spezifikationen und Anforderungen umgesetzt werden. Die Programmierung umfasst sowohl Frontend- als auch Backend-Komponenten, um die volle Funktionalität der Plattform zu gewährleisten.
Phase 4: Übergabe (Transition)
Die Übergabephase legt den Fokus auf die abschliessende Qualitätssicherung. Hier werden alle identifizierten Fehler behoben, umfassende Tests durchgeführt und die technische sowie benutzerorientierte Dokumentation erstellt. Ziel ist es, eine stabile und gut dokumentierte Anwendung bereitzustellen, die problemlos in Betrieb genommen werden kann.

Unten ist der detaillierte Plan:

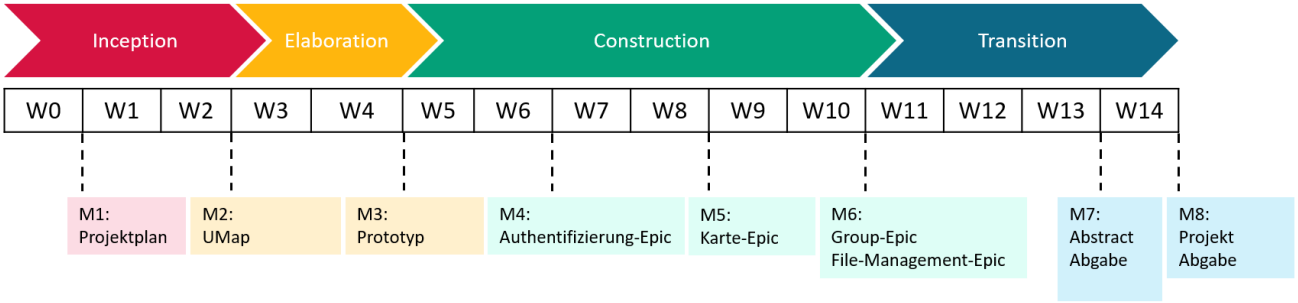

Abbildung 14.1: Projektplan

Kapitel 15

Projekt-Monitoring

Dieses Kapitel enthält ein Diagramm zur Zeiterfassung im Verlauf des Projekts.

15.1 Time-Tracking

Gemäss der offiziellen ECTS-Formel, bei der 1 ECTS = 30 Stunden entspricht, beträgt die insgesamt erforderliche Arbeitszeit pro Studierenden 240 Stunden.
Unten befindet sich eine Übersicht über die insgesamt aufgewendete Zeit pro Studierendem.

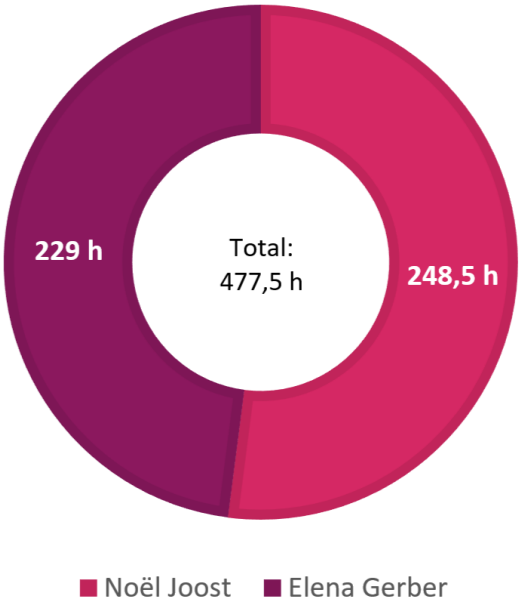

Abbildung 15.1: Zeiterfassung im Projektverlauf

15.2 Code Statistiken

Dieses Kapitel beschreibt die Code-Analyse, die als Indikator für die Codequalität dient. Für das Backend kommt PyLint und für das Frontend ESLint zum Einsatz.

15.2.1 PyLint

Pylint wurde als Qualitätsmassnahme eingesetzt, um den Code zu analysieren und einen Bericht zu erstellen. Dabei bewertet Pylint den Code und vergibt eine Punktzahl auf einer Skala von 10, die die Anzahl und den Schweregrad der festgestellten Probleme widerspiegelt. Für den vorliegenden Python-Code wurde eine Bewertung von 10/10 erzielt.

Es ist zu beachten, dass die Linter-Regeln angepasst wurden. Beispielsweise wurde die maximale Zeilenlänge erhöht, und Migrationsdateien wurden ausgeschlossen, da diese automatisch generiert werden und daher nicht bearbeitet werden sollten.

15.2.2 ESLint

ESLint wurde so konfiguriert, dass es sich an die empfohlenen Regeln für TypeScript und React. Die Codebasis für das Frontend enthält keine Linting-Probleme.

Kapitel 16

Softwaredokumentation

16.1 Technology-Stack

Dieses Kapitel fasst relevante Technologien und Werkzeuge zusammen, die für die Entwicklung der Anwendung verwendet wurden. Ausserdem werden einige Voraussetzungen beschrieben, die erfüllt sein müssen, um die Anwendung ausführen und nutzen zu können.

Technologie	Version
React	v18.3.1
React Router DOM	v6.28.0
React Auth Kit	v3.1.3
Material-UI (MUI)	v6.2.0
Redux Toolkit	v2.5.0
React-Redux	v9.2.0
i18next	v22.5.1
React-i18next	v12.3.1
Mapbox GL	v3.8.0
MapLibre GL	v4.7.1
React Map GL	v7.1.7
Supercluster	v8.0.1
TypeScript	v4.9.5
ESLint	v9.16.0
Prettier	v3.4.2
React-scripts	v5.0.1
Cypress	v13.16.1

Tabelle 16.1: Frontend Technologien

Technologie	Version
Django	v4.2.16
Django Cors Headers	v4.4.0
Django Filter	v24.3
Djangorestframework	v3.15.2
Django Environ	v0.11.2
Djangorestframework Simplejwt	v5.3.1
Pylint	v3.3.1
Pytest	v7.4.3
Requests	~2.32.3
Shapely	~2.0.6
Psycopg2	v2.9.10

Tabelle 16.2: Backend Technologien

16.2 Tool-Stack

Tool	Verwendung
WebStorm	IDE für Frontend-Entwicklung
PyCharm	IDE für Backend-Entwicklung
PgAdmin	Verwaltung und Abfragen der PostgreSQL-Datenbank
ChatGPT	Texterstellung, Textoptimierung, Rechtschreibe- und Grammatikprüfung, Übersetzung
DeepL	Übersetzung
GitHub Copilot	Unterstützung bei der Codeerstellung

Tabelle 16.3: Tool Stack

16.3 Installation

Die Dokumentation zur Installation von Swiss Emergency Map ist Teil des Repositories, das den Quellcode enthält. [20]

Anhang A

Testabdeckung

A.1 Backend-Testabdeckung

Hier ist noch die Testabdeckung von unserem Backend. Diese wurde mit Hilfe von Coverage.py erstellt. Den Report kann man im GitLab in der Pipeline finden.

Name	Stmts	Miss	Cover (%)
__init__.py	0	0	100
settings.py	29	0	100
urls.py	28	0	100
admin.py	0	0	100
apps.py	4	0	100
migrations/0001_initial.py	7	0	100
migrations/0002_osmcentroid_category.py	5	0	100
migrations/0003_remove_file_description.py	4	0	100
migrations/0004_category_color.py	4	0	100
migrations/0005_category_local.py	4	0	100
migrations/0006_municipality.py	4	0	100
migrations/0007_category_municipality_file_municipality_and_more.py	5	0	100
migrations/0008_userprofile.py	6	0	100
migrations/0009_remove_category_municipality_and_more.py	5	0	100
migrations/0010_poi_municipality.py	5	0	100
migrations/0011_municipality_logo.py	5	0	100
migrations/0012_category_municipality_file_municipality_and_more.py	5	0	100
migrations/0013_userprofile.py	6	0	100
migrations/0014_remove_municipality_users_and_more.py	4	0	100
migrations/0015_alter_poi_poi_files.py	4	0	100

Name	Stmts	Miss	Cover (%)
migrations/__init__.py	0	0	100
models/__init__.py	5	0	100
models/category.py	8	0	100
models/file.py	7	1	86
models/municipality.py	9	0	100
models/osm.py	55	0	100
models/poi.py	13	1	92
models/user.py	5	0	100
scripts/__init__.py	0	0	100
scripts/get_category.py	45	12	73
scripts/get_osm_pois.py	12	0	100
serializers/__init__.py	0	0	100
serializers/category_serializer.py	13	3	77
serializers/file_serializer.py	11	1	91
serializers/municipality_add_serializer.py	7	0	100
serializers/municipality_serializer.py	6	0	100
serializers/open_user_profile_serializer.py	8	0	100
serializers/open_user_serializer.py	6	0	100
serializers/osm_centroid_serializer.py	8	0	100
serializers/poi_serializer.py	57	6	89
serializers/user_profile_serializer.py	38	19	50
serializers/user_serializer.py	11	0	100
tests/__init__.py	0	0	100
tests/test_add_to_municipality.py	95	0	100
tests/test_authentication.py	60	0	100
tests/test_categories.py	49	0	100
tests/test_file.py	61	0	100
tests/test_osm_poi.py	51	0	100
tests/test_poi.py	63	0	100
views/__init__.py	0	0	100
views/add_user_to_municipality.py	30	5	83
views/category_view_set.py	39	4	90
views/file_view_set.py	37	4	89
views/municipality_user_view_set.py	12	4	67
views/municipality_view_set.py	25	14	44
views/open_user_view.py	11	3	73
views/osm_poi_view_set.py	17	0	100

Name	Stmts	Miss	Cover (%)
views/poi_view_set.py	61	7	89
views/remove_user_from_municipality.py	22	2	91
views/update_osm_points.py	18	10	44
views/user_profile_view_set.py	11	0	100
views/user_view_set.py	11	0	100
manage.py	11	2	82
TOTAL	**1142**	**98**	**91%**

Anhang B

Abgabe

B.1 Dokumentation

Die digitale Version der Dokumentation wird auf das AVT-Tool der OST hochgeladen.

B.2 Broschüren-Abstract

Das Broschüren-Abstract ist ein zusätzlicher Abstract, der auf https://abstract.rj.ost.ch/ eingegeben werden muss. Dieser Abstract wird vom OST gefordert und wurde vom Team rechtzeitig eingereicht.

B.3 Unterzeichnete Dokumenten

OST verlangt eine Unabhängigkeitserklärung, eine Einverständniserklärung zur Veröffentlichung auf E-Prints und eine Vereinbarung über Urheber- und Nutzungsrechte für jede Arbeit. Diese unterzeichneten Dokumente wurden fristgerecht eingereicht.

B.4 Quellcode-Repositorien

- Frontend: https://gitlab.ost.ch/feuerwehrkarte-sa/fwk-frontend
- Backend: https://gitlab.ost.ch/feuerwehrkarte-sa/fwk-backend
- Dokumentation: https://gitlab.ost.ch/feuerwehrkarte-sa/fwk-doku

Anhang C

Glossar

Begriff	Beschreibung
OSM	OpenStreetMap ist eine kollaborative Kartenanwendung, die es Benutzern ermöglicht, geografische Daten zu erstellen und zu bearbeiten.
POI	Points of Interest sind bestimmte geografische Standorte, die für Benutzer von Interesse sein können, z.B. Restaurants, Schulen, Sehenswürdigkeiten, Feuerhydranten oder Defibrillatoren.
Dropbox	Dropbox ist ein cloudbasierter Speicher- und Synchronisationsdienst, der es Nutzern ermöglicht, Dateien online zu speichern, zu teilen und darauf zuzugreifen.
Geocoding	Geocoding ist der Prozess, bei dem ein physischer Ort in geografische Koordinaten (Breiten- und Längengrad) umgewandelt wird.
MUI	Material-UI ist eine beliebte Open-Source-Bibliothek für React, die Googles Material Design umsetzt.
API	Application Programming Interface ist ein Satz von Routinen, Protokollen und Werkzeugen, die es ermöglichen, dass Softwareanwendungen miteinander kommunizieren.
FR	Functional Requirement (funktionale Anforderung) beschreibt, was ein System tun soll, um die Bedürfnisse der Benutzer zu erfüllen.
NFR	Non-Functional Requirement (nicht-funktionale Anforderung) beschreibt, wie ein System die funktionalen Anforderungen erfüllen soll, z.B. Leistung, Sicherheit und Benutzerfreundlichkeit.
LaTeX	Ein Textsatzsystem, das häufig für wissenschaftliche Arbeiten verwendet wird.
Scrum	Scrum ist ein agiles Projektmanagement-Framework, das Teams dabei unterstützt, komplexe Projekte in iterativen und inkrementellen Schritten zu entwickeln.
RUP	Rational Unified Process ist ein iterativer Softwareentwicklungsprozess, der die Entwicklung in Phasen gliedert und Best Practices für die Softwareentwicklung definiert.

Begriff	Beschreibung
RLS	Row Level Security ist eine Datenbankfunktion, die den Zugriff auf Zeilen in einer Tabelle basierend auf Benutzerrollen oder Kontextbedingungen einschränkt, um sicherzustellen, dass Benutzer nur die für sie bestimmten Daten sehen oder bearbeiten können.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] uMap, last accessed 19. Dezember 2024. [Online]. Available: https://umap.osm.ch/de/about/

[2] Leaflet, last accessed 19. Dezember 2024. [Online]. Available: https://leafletjs.com/

[3] Maplibre, last accessed 19. Dezember 2024. [Online]. Available: https://maplibre.org/

[4] Mapbox, last accessed 19. Dezember 2024. [Online]. Available: https://www.mapbox.com/

[5] openmaptiles, last accessed 19. Dezember 2024. [Online]. Available: https://openmaptiles.org/

[6] MapTiler, last accessed 19. Dezember 2024. [Online]. Available: https://www.maptiler.com/

[7] Django, last accessed 19. Dezember 2024. [Online]. Available: https://www.djangoproject.com/

[8] feuerwehr innovativ, “Utm einsatzkarte,” last accessed 14 Dezember 2024. [Online]. Available: https://feuerwehr-innovativ.at/innovationen/utm-einsatzkarte/

[9] React, last accessed 19. Dezember 2024. [Online]. Available: https://react.dev/

[10] Vue.js, last accessed 19. Dezember 2024. [Online]. Available: https://vuejs.org/

[11] Flask, last accessed 19. Dezember 2024. [Online]. Available: https://flask.palletsprojects.com/en/stable/

[12] Spring-Boot, last accessed 19. Dezember 2024. [Online]. Available: https://spring.io/projects/spring-boot

[13] Jest, last accessed 19. Dezember 2024. [Online]. Available: https://jestjs.io/

[14] Cypress, last accessed 19. Dezember 2024. [Online]. Available: https://www.cypress.io/

[15] pytest, last accessed 19. Dezember 2024. [Online]. Available: https://docs.pytest.org/

[16] python, last accessed 19. Dezember 2024. [Online]. Available: https://docs.python.org/3/library/unittest.html

[17] Django, last accessed 14. Dezember 2024. [Online]. Available: https://docs.djangoproject.com/en/5.1/topics/testing/

[18] Overpass, last accessed 19. Dezember 2024. [Online]. Available: https://wiki.openstreetmap.org/wiki/Overpass_API

[19] MUI, last accessed 19. Dezember 2024. [Online]. Available: https://mui.com/

[20] E. G. Noël Joost, last accessed 19. Dezember 2024. [Online]. Available: https://gitlab.ost.ch/feuerwehrkarte-sa