



HSR

HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Projektarbeit 2, Master of Science in Engineering,
Major Software and Systems

Waldmeister - Outdoors

HSR Hochschule für Technik Rapperswil

Fall 2017

Author: Daniel Schmider

Supervisor: Prof. Stefan Keller

Kapitel 1

Abstract

Keywords:

Inhaltsverzeichnis

1 Abstract	1
2 Management summary	4
2.1 Ausgangslage	4
2.2 Ziel der Arbeit	4
2.3 Ergebnisse	5
2.4 Ausblick	5
3 Motivation und Ausgangslage	6
3.1 Ein mobiles App für Feldforschung im Wald	6
3.2 Use Cases	7
3.2.1 Zugriff auf Vegetationskundliche Karten	7
3.2.2 Bearbeitung der Vegetationskundlichen Karten	7
3.2.3 Unterstützung während der Untersuchung	7
3.2.4 Automatische Bestimmung einer Waldfläche	8
3.3 Mobile Limits	8
4 Technologies	9
4.1 Progressive Webapp	9
4.2 ESRI / ArcGIS online	9
4.3 Vue.JS	10
4.3.1 Separation of Concern	11
4.3.2 Vue-Router	12
4.3.3 VueX	12
4.3.4 Vuetify	14
4.4 Axios	14
4.5 Django	14
4.5.1 Django Rest Framework	14
4.6 PostgreSQL	15
4.6.1 PostGIS	15
4.7 Leaflet	15

4.7.1	TileLayer	15
4.7.2	Leaflet editable	15
5	Implementation	18
5.1	Mockup	18
5.2	UMLs	18
5.2.1	Use Case - Diagramm	19
5.2.2	Klassendiagramm, Datenbankdiagramm	19
5.2.3	Sequenzdiagramm	19
6	Results	33
6.1	Discussion / Screenshots	33
7	Links	35
8	API Documentation	36

Kapitel 2

Management summary

2.1 Ausgangslage

Je nach Untergrund, Bodeneigenschaften, Gelände sowie Klima gedeihen unterschiedliche Typen von Wäldern. Diese Typen werden Waldstandorte genannt. Aktuell werden Karten, die im Auftrag des Kantons von Experten angefertigt wurden, nur in grossen Intervallen revidiert oder sind nicht öffentlich zugänglich. Einer der Gründe dafür sind u.a. die hohen Kosten, die eine Analyse im Feld mit sich bringt. Zudem ist die Erfassung und Nachführung geprägt von analogen Vorgängen, da die vorhandenen technischen Geräte und Programme für den Einsatz im Feld ungeeignet sind. Daher muss von Hand Niedergeschriebenes im Büro digitalisiert werden, bevor es an den Arbeitgeber geschickt werden und später auf kantonal isolierten Plattformen publiziert werden kann.

2.2 Ziel der Arbeit

Die Erfassung und Publikation von Waldstandorten sollte vereinfacht und beschleunigt werden. Dabei sollen digitale Technologien eingesetzt werden wie Smartphone, GPS und Internet. Diese neuen Instrumente sollen entsprechend geschulten Nutzern die Erfassung von Waldstandorten sowie öffentliche und private Informationen in Form von Flächen und Punkten ermöglichen. Auf einer Karte wird mittels GPS die eigene Position ermittelt. Umliegende, bereits erfasste Waldstandorte, öffentliche Flächen anderer User und die eigenen privaten Flächen werden angezeigt. Diese Flächen können ebenfalls Waldstandorte beschreiben oder zusätzliche Informationen über den Standort beinhalten, z.B. eine speziell gekennzeichnete Beobachtungsfläche.

2.3 Ergebnisse

Nach einer Evaluation eines Prototyps, erstellt mithilfe eines kommerziellen Produkts (vgl. Abbildung 2), und der Erstellung von Mockups, wurde ein eigenes Webapp aldmeister Outdoors Prototyp realisiert. Durch die Webapplikation kann die Arbeit der Experten erleichtert werden. Da die Waldstandort-Karte gleichzeitig im Web synchronisiert ist, wird darüber hinaus der Informationsaustausch unter allen Beteiligten erleichtert.

2.4 Ausblick

Grosse Teile der Schweiz sind noch unkartiert, und viele Waldstandorte könnten sich unter dem Einfluss der Klimaerwärmung verändern. Die kontinuierliche Beobachtung solcher Standorte ist Forschungsgegenstand und die Arbeit im Feld ist unerlässlich. "Waldmeister Outdoors" kann im Berufsalltag sowie bei der Kommunikation mit Institutionen den Arbeitsfluss beschleunigen. Weitere Features wie die Verwendung von Plus Codes und offline-Fähigkeiten welche bei Verbindungsproblemen zum Einsatz kommen, bzw Benutzerflächen automatisch synchronisieren sobald eine Verbindung besteht. Des weiteren bietet es sich an, dass sich User in Gruppen einklinken können um unter sich Benutzerflächen zu teilen und zu besprechen, bevor sie veröffentlicht werden. Ebenfalls sollten erstellte Flächen von registrierten Benutzern und deren Gruppen verändert und gelöscht werden können, nachdem sie Erstellt wurden.

Kapitel 3

Motivation und Ausgangslage

3.1 Ein mobiles App für Feldforschung im Wald

Aufbauend auf der existierenden "Waldmeister" App, welches von Studenten und Professoren als Nachschlagewerk für Waldstandortbestimmungen in der Praxis verwendet wird, existieren viele, teils digitalisierten Karten, welche den Stand der Forschung in sogenannten Vegetationskundlichen Karten beschreiben. Waldstandortbestimmung ist ein sich ständig im Wechsel befindendes Thema und auch in der Schweiz können sich Standorte oder deren Befund über Jahre ändern. Vegetationskundliche Karten, welche von Experten im Auftrag des Kantons angefertigt werden, befinden sich in statischen oder ungewarteten Zuständen in Archiven des Kantons, oder werden in grossen Intervallen (5-10 Jahre) revidiert. Teilweise sind diese Daten daher gar nicht, oder nur oder in sehr veralteten Zustand zugänglich. Dies liegt hauptsächlich am grossen Kostenaufwand welche eine Analyse im Feld durch Experten mit sich bringt. Der Datenfluss ist geprägt von analogen Vorgängen, vor allem da viele technische Geräte nicht für den Einsatz im Feld geeignet sind und kantonale Organisationen als Mittelsmänner etabliert sind, welche die Daten von Experten archivieren und ggf. in digitaler Form veröffentlichen. "Waldmeister - Outdoors" zielt darauf hin den Arbeitszyklus der Analyse und Publikation von Waldstandorten zu vereinfachen und zu beschleunigen. Dies soll erreicht werden durch neue technische Möglichkeiten und Medien wie dem Smartphone, GPS und Mobiles Internet. Das Instrument "Waldmeister - Outdoors" soll dazu verwendet werden die Erfassung von Geoinformationsdaten bezüglich der Waldstandortbestimmung zu standardisieren und deren übermittlung an die zuständige Behörde, Forschern und anderen Experten zu beschleunigen. Anstelle eines Jahrrelangen Projekts, einen bestimmten Standort in Waldstandorte zu kategorisieren, soll ein System entwickelt werden, welches eine inkrementelle digitale Erfassung und Publikation ermöglicht, den Arbeitsaufwand der Experten erleichtert und den

Informationsaustausch, - und Abgleich beschleunigt.

3.2 Use Cases

3.2.1 Zugriff auf Vegetationskundliche Karten

Während der Feldforschung kann es sehr hilfreich sein auf bereits kategorisierte Waldflächen Zugriff zu haben um sich am Standort zu orientieren. Dieses Material ist oft in einem Kantonalen Portal z.B. <https://maps.zh.ch> erhältlich, sind jedoch nur selten kompatibel mit mobilen Geräten, interagieren nicht mit dem GPS des Smartphones oder die Webseiten sind schwer zu navigieren. "Waldmeister-Outdoors" soll einem gezielten Zweck dienen und nicht mit Kartenmaterial überladen werden um den Zugriff auf die Vegetationskundliche Karten zu vereinfachen und die Navigation zu beschleunigen.

3.2.2 Bearbeitung der Vegetationskundlichen Karten

Kantonale Vegetationskundliche Karten sind statisch (d.h. Read-Only) und können von Usern nicht bearbeitet oder erweitert werden. Dies führt dazu dass Kartenmaterial in veraltetem Zustand vorliegt und Fehler oder Veränderungen nur mit grossem Aufwand upgedatet werden können. Nicht nur führt dies zu Problemen bei der Kommunikation mit anderen Experten, Forschern und Studenten, es behindert auch den Arbeitsfluss der Person, welche sich im Feld befindet, da andere Mittel zur temporären Festhaltung der Ergebnisse verwendet werden müssen, um an einem späteren Zeitpunkt wieder darauf zugreifen zu können (z.B. eine halbjährliche Untersuchung eines Standorts, ohne die Zwischenergebnisse zu publizieren). Dies geschieht oft analog, auf Papier im Feld und kann zu einem späteren Zeitpunkt zum persönlichen Gebrauch digitalisiert werden, was zu einem erhöhten Arbeitsaufwand führt. Durch die Verwendung von "Waldmeister - Outdoors" kann das gleiche Werkzeug benutzt werden um eine Fläche während der Untersuchung sowohl im Feld als auch im Büro zu beschreiben, sowie die finalen Befunde am Ende der Untersuchung zu publizieren und zu Teilen.

3.2.3 Unterstützung während der Untersuchung

Feldforschung hat oft mit der Orientierung und der Manövrierung des Standorts an sich zu tun und auch hier kann "Waldmeister - Outdoors" den Arbeitsaufwand simplifizieren und reduzieren. Durch die direkte digitale Erfassung von beliebigen Notizen bezüglich dem Standort müssen diese nicht mehr analog erfasst werden

und können direkt der Position in der realen Welt zugeordnet werden. Ist Beispielsweise ein Gebiet schwer Befahrbar oder zugänglich kann dies direkt auf der digitalen Karte erfasst und für persönliche Zwecke gespeichert werden, können aber ebenfalls publiziert werden falls sie für Andere von Interesse sind. Es kann sich hierbei auch um Pfade oder einzelne Standorte von Indikatoren handeln, bzw. eine genaue Lage der Observationsfläche welche untersucht werden soll.

3.2.4 Automatische Bestimmung einer Waldfläche

Durch den Zugriff auf eine Datenbank, in welcher jede Waldstandort einem Typ zugeordnet ist, kann mithilfe des Mobilen Geräts der Typ des Waldstandorts in welcher sich ein User gerade befindet automatisch bestimmt werden. Dies kann mithilfe des GPS Sensors des Mobilen Geräts und einer Datenbankabfrage zu jedem Zeitpunkt geschehen.

3.3 Mobile Limits

In vielen Fällen ist eine Standortbestimmung durch GPS im Wald sehr ungenau und die Internetverbindung kann instabil sein. Im Idealfall überträgt das Werkzeug so wenig Daten wie möglich, speichert diese auf dem Gerät und wartet auf eine stabile Verbindung um Daten auf den Server zu übertragen.

Kartenmaterial sollte wenn möglich permanent auf das Mobile Gerät geladen werden können, damit Datenvolumen bei der Verwendung im Feld nicht strapaziert werden.

Kapitel 4

Technologies

4.1 Progressive Webapp

Um das Werkzeug "Waldmeister - Outdoors" nicht auf eine Platform von Mobilen Geräten zu beschränken (iOS oder Android), setzt es auf die Prinzipien der Progressive Webapps (PWA). Diese beschreiben eine Webseite welche viele Merkmale besitzt, welche bisher den nativen Apps vorbehalten waren. Eine PWA ist gewissermassen eine responsive Website, welche auch offline verwendet werden kann und schliesst dadurch eine zusätzliche Entwicklung einer nativen App, parallel zur Webseite überflüssig. Eine PWA erreicht diese offline Fähigkeiten durch den Einsatz von Service Workern; ein JavaScript welches von Web-Browsern im Hintergrund ausgeführt wird. Einmal online aufgerufen, können die Inhalte beim nächsten Besuch der Seite auch dann angezeigt werden, wenn eine schlechte oder sogar gar keine Internetverbindung besteht (Offline-Betrieb). Auch die von nativen Apps bekannten Push-Benachrichtigungen sind mit Service Workern möglich.
[?]

Grundlegende Charakteristiken der PWA sind Offline Funktionalität, Push Notifications, Add-To-Homescreen, jedoch ist keine Installation notwendig (beispielsweise über den Apple Appstore oder Google Playstore). Verbreitete Mobile Browser wie Firefox und Google Chrome haben bereits eine vollständige Unterstützung von PWAs, Safari sollte dies bis Ende Februar 2018 ebenfalls implementiert haben und kann daher auch auf iOS verwendet werden. PWAs können auf den Home-Screen des mobilen Geräts hinzugefügt werden.

4.2 ESRI / ArcGIS online

Technologien von ESRI und insbesondere ArcGIS online wurden recherchiert um einen funktionierenden Prototypen mit offline-caching zu erstellen. Hintergrund-

karten (in Form einen Tile-Layers) können auf dem Gerät zwischengespeichert werden, und die Erstellung und Synchronisation von editierbaren Vektorlayern funktioniert auch beim offline Betrieb. Vektorlayer können Punkte, Pfade oder Flächen beschreiben und können offline erstellt werden und werden bei verfügbaren Internetverbindung mit einem Server synchronisiert. Dieses Verhalten kann bei einer PWA durch Service-Worker rekriert werden.

4.3 Vue.JS

Vue.JS ist ein JavaScript framework welches sich zum Erstellen von Single-Page Webapplikationen eignet. Es wurde im Jahr 2013 erstmals veröffentlicht und wurde am 19. Dezember 2017 auf die aktuellsten Version 2.5.13 gepatcht. Vue.JS folgt einer Variation des Model-View-Controller Entwurfsmusters genannt dem Model - View - ViewModel Muster. Wie auch das MVC folgt MVVM dient es der Trennung von Darstellung und der Logik der Benutzerschnittstelle. Dies erlaubt dem nutzenden Entwickler, die Struktur der Anwendung nach eigenen Ansprüchen zu richten. Entwickler beschreiben es daher als "less opinionated" im Vergleich zu anderen populären JavaScript Webframeworks wie Angular.JS und React. Vue.JS kann von Entwicklern eingesetzt werden welche HTML und JavaScript beherrschen und erfordert keine weiteren Webtechnologien. Vue.JS setzt eine Webseite aus Instanzen und Komponenten, bzw Single File Components zusammen. Single File Components sind bei VueJS welche Architekturprobleme von mittel bis grossen Webapps, welche vollständig von JavaScript getrieben werden, zu verbessern versucht. Folgende Probleme tauchen dabei auf:

1. Global definitions
Global definitions force unique names for every component
2. String templates
String templates lack syntax highlighting and require ugly slashes for multiline HTML
3. No CSS support
No CSS support means that while HTML and JavaScript are modularized into components, CSS is conspicuously left out
4. No build step
No build step restricts us to HTML and ES5 JavaScript, rather than preprocessors like Pug (formerly Jade) and Babel

VueJS besagt, dass all diese Probleme von Single File Components (mit .vue extension) dank Werkzeugen wie Webpack und Browserify gelöst werden. Eine solche Komponente besteht auf HTML Template, JavaScript und CSS in einer eigenen, abgekapselten Datei. Durch das erzielt VueJS

1. Complete syntax highlighting
2. CommonJS modules
3. und Component-scoped CSS

Wem diese Idee Abkapselung nicht gefällt kann weiterhin ein CSS auslagern und in eine Komponente (innerhalb des HTML Templates) importieren:

```
<!-- my-component.vue -->
<template>
  <div>This will be pre-compiled </div>
</template>
<script src="./my-component.js"></script>
<style src=".my-component.css"></style>
```

4.3.1 Separation of Concern

Was ist mit Separation of Concern (SoC)? Eine logische Vorgehensweise bei Softwareengineering ist es, ein Computerprogramm in logische Abschnitte einzuteilen und zu separieren. Diese Teile sollten sich um einen Zweck oder Belangen (Concern) kümmern. Dies heißt jedoch nicht, dass die verschiedenen Dateitypen unbedingt in separaten Dateien aufgeteilt werden müssen. In der modernen User Interface Entwicklung und den Entwicklern von VueJS ist es oft einfacher gefallen verschiedene Komponenten, welche lose gekoppelt sind, zu komponieren, statt sie auf drei riesigen Layern (HTML, JS und CSS) getrennt zu halten, sie aber in den Komponenten zu verflechten. [?]

Auf diesem Weg sind Komponenten (Template, die Logik und das Styling) sind zusammenhängender und auch einfacher zu warten, obwohl dies nicht den Prinzipien von SoC folgt. Traditionelles SoC unterteilt dies in die Gruppen der Zwecke Organisation (HTML), Präsentation (CSS) und Interaktion und Verhalten (JavaScript)

4.3.2 Vue-Router

Der Vue-Router ist das Herzstück einer Single-Page-Applikation (SPA). Der offizielle Vue-Router ist ein Client-seitiger Router welcher mithilfe der HTML5 History API voll funktionsfähiges Client-side routing.

In der HTML Definition der Haupkomponente kann <router-view> als Platzhalter verwendet werden um die Komponenten anzuzeigen, welche abhängig von der momentanen Route an dieser Stelle angezeigt werden sollen. Ein Wechsel zwischen diesen Routen bewirkt kein Page-Reload, da dies von Vue.JS lediglich innerhalb derselben Page änderungen bewirkt und keine tatsächlichen URL Aufrufe ausführt.

Der Vue Router wird innerhalb einer Hauptseite, welcher die Grundlage der Webseite darstellt, angezeigt. Methoden von Komponenten können bewirken dass sich der Inhalt, welcher an der Stelle des Router-Views angezeigt wird, verändert. Menupunkte im Header (z.B Register, Login, About, Map), bewirken mit .push() dass der Vue-Router mithilfe des index.js files die korrekten Komponenten an dieser Stelle anzeigt. Dies kann auch direkt über eine URL Eingabe /register oder /login erfolgen, ohne dass der Aufruf über eine interne Komponente ausgeführt werden muss. Die Datei index.js beinhaltet daher alle möglichen Pfade welche von der Webapp aufgelöst werden und bestimmt die angezeigten Komponenten welche mit dieser Route verknüpft sind.

Alternativ könnten die ähnlichen Lösungen von Page.js oder Director als Third-Party Produkte an dieser Stelle integriert werden, es ist jedoch empfohlen die offizielle Vue-Router Library zu verwenden.

4.3.3 VueX

VueX ist eine offizielle Erweiterung von Vue.JS und fungiert als Statusmanager. VueX arbeitet mit einem Store welcher die Zustände aller Komponenten in einer Vue Applikation über Regeln definiert. VueX besteht aus Actions, Mutations und States, und Aktionen können in dieser Reihenfolge eine Auswirkung auf die Vue Komponenten haben.

VueX hat Vorteile bei mittel bis grossen Projekten welche auf dem Single-Page-Application Prinzip basieren. VueX bietet auch die Möglichkeit einen zentralen Store in kleinere Module aufzuteilen, jedes mit ihrer eigenen State, Mutations, Actions Werten.

Da Komponenten in Vue abgekapselt sind, können sie standardmäßig nicht auf Daten zugreifen welche in anderen Komponenten definiert werden. Solche Daten müssen per Store verfügbar gemacht werden damit sie über mehrere Instanzen geteilt werden können.

```

const sourceOfTruth = {}

const vmA = new Vue({
  data: sourceOfTruth
})

const vmB = new Vue({
  data: sourceOfTruth
})

```

Wird nun sourceOfTruth verändert, wird sie in allen Komponenten, in welcher sie verwendet wird automatisch auf den neuen Stand gebracht. Dies kann in grösseren Applikationen schnell unübersichtlich werden, da jede Komponente diese sourceOfTruth verändern kann, ohne eine nachvollziehbare Spur zu hinterlassen. Es wird daher empfohlen das Store Pattern von Vuex zu implementieren, welche Veränderungen am Store nur über Mutationen zulässt. Somit wird es klarer zu welcher Zeit welche Mutationen aufgerufen werden können und wie sie durchgeführt wurden:

```

var store = {
  debug: true,
  state: {
    message: 'Hello!'
  },
  setMessageAction (newValue) {
    if (this.debug) console.log('setMessageAction_triggered_with
      ↪ ', newValue)
    this.state.message = newValue
  },
  clearMessageAction () {
    if (this.debug) console.log('clearMessageAction_triggered')
    this.state.message = ''
  }
}

```

Komponenten können auch private Zustände haben, dies wird mit "privateState" erreicht. In diesem Fall muss der sharedState ebenfalls definiert werden.

Dies bewirkt dass eine Komponente nicht direkt einen Wert oder Zustand im Store verändern kann, sondern dies über einen Event ausführt, welcher den Store selbst informiert welchen State es über eine Mutation zu verändern gilt.

4.3.4 Vuetify

Vuetify ist eine Vue.JS UI Framework welches das Frontend aus vielen, Material-Design basierten UI Bausteinen zusammenbaut.

4.4 Axios

Axios ist eine JavaScript library zur Erstellung von Promise-Based HTTP Requests, welche "Waldmeister-Outdoors" dazu verwendet mit dem Server zu kommunizieren. Axios ermöglicht es asynchrone HTTP requests zu REST Endpunkten abzusetzen oder Requests wie Create, Read, Update, Delete (CRUD) Operationen auszuführen. Axios kann in puren JavaScript Projekten verwendet werden oder auch in Projekten welche auf Vue.JS basieren. Ein Promise-Objekt repräsentiert die in der Zukunft geschehende Komplettierung einer asynchronen Operation (oder deren Abbruch durch einen Fehler).

4.5 Django

Als Server zur Verwaltung der User und der Daten welche die User generieren und benötigen kommt Django zum Einsatz. Es ist ein Open-Source Webframework welches das Python Gegenstück zu Ruby-On-Rails darstellt. Im Kern folgt es dem Model-View-Controller Prinzip, obwohl es eigene Namensgebung für diese Verwendet. Django verwendet eine PostgreSQL Datenbank um Daten persistent zu machen. Django wird ebenfalls dazu verwendet um User einen Account zu geben, damit nur sie Zugriff auf Ihre privaten Flächen haben, bevor sie vom User veröffentlicht werden.

4.5.1 Django Rest Framework

Da eine SPA hauptsächlich über API Schnittstellen mit dem Server kommuniziert, wird auf dem Server das Django Rest-Framework (DRF) eingesetzt. Es bietet ein sehr flexibles System zur Erstellung von RESTful Web-APIs. Das DRF bietet die Möglichkeiten GET, POST, PUT und DELETE auf eine Resource auszuführen. "Waldmeister-Outdoors" verwendet die REST Api beispielsweise um usergenerierte Flächen, Pfade oder Punkte in die Datenbank zu speichern oder diese zur Darstellung in der Map aus der Datenbank zu laden. Ebenfalls werden Benutzer welche sich Registrieren mit Username, Password und ggf. Emailadresse in der Datenbank eingetragen.

4.6 PostgreSQL

Postgres ist das Datenbank Management System welches mit Django zusammen die Daten persistent macht, welche die User per API in der PWA generieren. Hierzu wird das Django Packet Psycopg2 verwendet. Django kann durch Models ein Datenbankschema beschreiben, welches von PostgreSQL generiert und in einer lokalen PostgreSQL Instanz gespeichert wird.

4.6.1 PostGIS

Um Geoinformationsdaten wie z.B. Polygone und Pfade korrekt zu speichern wird auf der Datenbank das Plugin PostGIS installiert. Dadurch kann Django die benötigten Datenbankmodelle erstellen und per REST Schnittstelle speichern.

4.7 Leaflet

Leaflet ist eine JavaScript Library welche es ermöglicht Map auf dem Client darzustellen. Es fokussiert sich auf Simplizität, Performanz und ist sehr schlank, was einer PWA sehr entgegen kommt. Die Library ist nur 38 KB gross und ermöglicht es viele Features welche bei der Darstellung einer Map benötigt wird zu verwenden.

4.7.1 TileLayer

Der TileLayer fungiert als Hintergrundkarte welche dynamisch geladen wird. Je nach benötigtem Kartenausschnitt werden die Tiles als .pngs geladen und auf der Karte dargestellt. Dies führt dazu dass je nach Grösse und Zoomstufe des Kartenausschnitts nur minimalen Datenaufwand betätigt wird. Als Hintergrundkarte werden die "Terrain" Tiles von "stamen-tiles" verwendet.

4.7.2 Leaflet editable

Damit die User neue Polygone, Punkte und Pfade erfassen können benötigt Leaflet das Plugin "Leaflet editable", welches es ermöglicht neue Objekte direkt auf der Map zu zeichnen, oder bestehende Objekte zu editieren. Sobald ein Objekt abgeschlossen ist, wird es per REST Schnittstelle an die Datenbank übertragen. Der User hat die Möglichkeit die Objekt einen Namen als Label zuzuweisen, und es zwischen privat oder public zu wechseln. Jeder User hat nur Zugriff auf seine eigenen privaten Flächen, bzw. Pfade und Punkte, ausser er wählt es diese zu Veröffentlichen, bzw "public" zu machen, damit sie alle User auf der Map sehen.

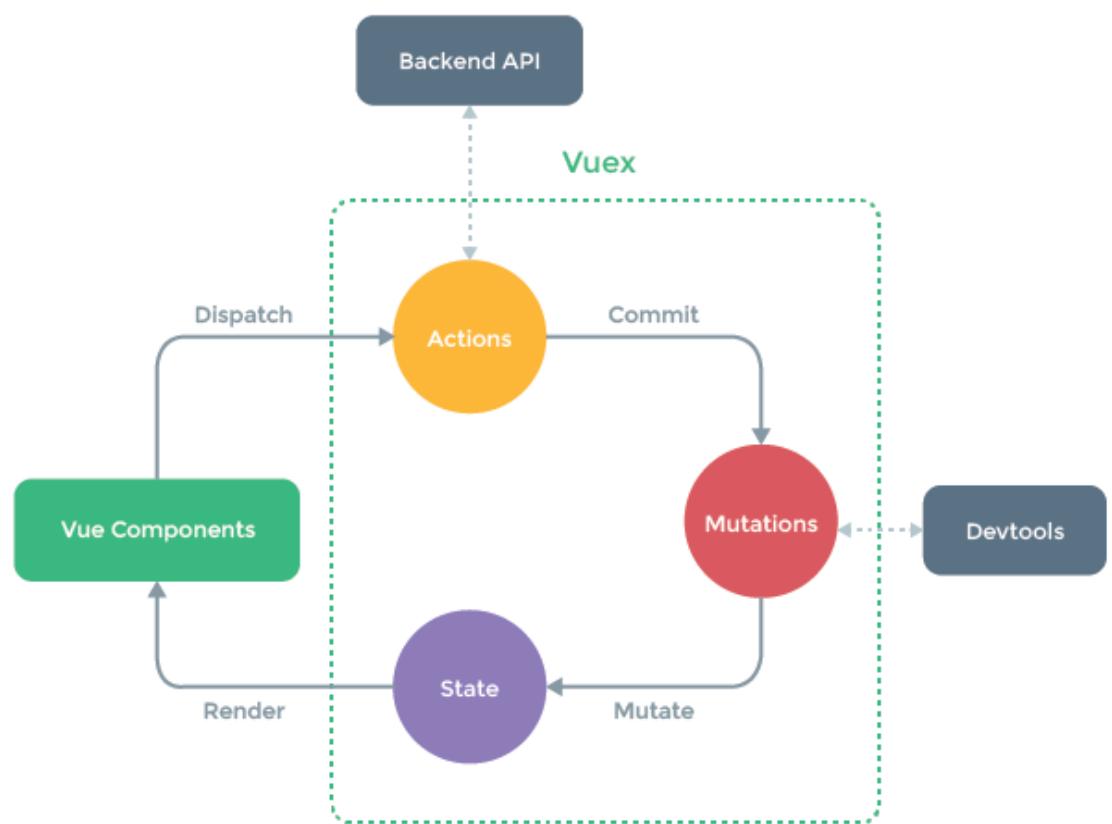


Abbildung 4.1: Vuex Action-Mutations-State Diagram

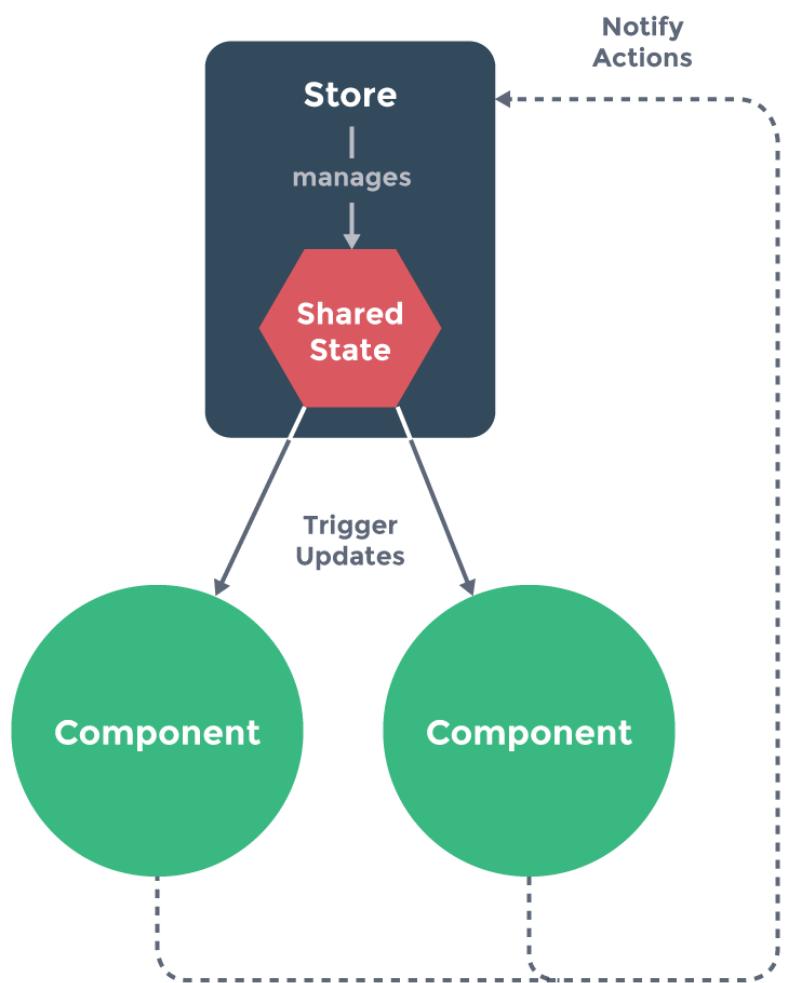


Abbildung 4.2: Vuex private

Kapitel 5

Implementation

5.1 Mockup

Die Mockups wurden vor der Implementation erstellt um das Screendesign und Layout klarer zu definieren bevor es um die technische Implementation von "Waldmeister - Outdoors" ging. In den Abbildungen 1 bis 9 kann man den Arbeitsschritt Einloggen und Erstellen einer neuen Fläche und eines Points of Interests (POI) sehen. Zusätzlich sieht der User seine eigene Location auf der Map eingetragen und hat über das Menu "My Places" Zugriff auf eine Liste seiner erstellten Flächen. Ein Kontextmenu gibt bei der Anzeige eines bestimmten Objekts zusätzliche Informationen über dieses.

5.2 UMLs

UML Diagramme geben Auskunft über die Architektur und Abläufe des Systems. Es ist eine grafische Repräsentation in Form von Diagrammen welche in der Sprache nach den Definitionen der "Unified Modeling Language" produziert sind. Sie bestimmt bei der Modellierung Beziehungen und Abhängigkeiten zwischen Begriffen und Notationen für diese Begriffe und statischen Strukturen und dynamischen Abläufen. Sie finden Verwendung und ist die dominierende Sprache für die Softwaresystem-Modellierung und dient zur Kommunikation von Ideen zwischen Projektauftraggeber, Softwareentwickler und Systemingenieuren. In den 90er Jahren wurde die erste Spezifikation unter dem Namen UML 1.x entwickelt und wurde im Jahr 2000 revidiert und wurde in UML2 umbenannt. Seit Februar 2008 liegt Version 2.2 in der finalen Version vor. Seit Juni 2015 wird die aktuellste Version 2.5 verwendet.

Hauptschlich definiert die Sprache Aktionen, Aktivitäten, Allgemeines Verhalten, Anwendungsfile, Informationsflüsse, Interaktionen, Klassen, Komponenten, Kom-

positionsstrukturen, Modelle, Profile, Schablonen, Verteilungen und Zustandsautomaten. Diese kommen Strukturdiagrammen und Verhaltensdiagramme zum Einsatz, wie u.a. dem Klassen und Komponentendiagramm, oder Aktivitätsdiagramm, Sequenzdiagramm, Kommunikationsdiagramm, etc... Die Grenzen zwischen den über vierzehn Diagrammtypen verlaufen weniger scharf. UML2 verbietet nicht, dass ein Diagramm grafische Elemente enthält, die eigentlich zu unterschiedlichen Diagrammtypen gehören. Elemente aus einem Strukturdiagramm und aus einem Verhaltensdiagramm können nach Anwendungsfall auf dem gleichen Diagramm dargestellt werden, falls damit eine besonders treffende Aussage zu einem Modell gemacht werden kann.

5.2.1 Use Case - Diagramm

Das Diagramm 5.10 zeigt auf welche Möglichkeiten ein User hat mit dem Werkzeug zu interagieren. Ein User welcher sich nicht registriert kann weder Flächen generieren oder editieren und kann keine privaten Flächen sehen. Er kann jedoch die Vegetationskundliche Karte und öffentliche Flächen aller anderen User sehen. Nachdem er sich registriert und eingeloggt hat, kann er Flächen erstellen und editieren. Diese teilen sich in öffentliche und private Benutzerflächen auf. Diese Option kann er während der Erfassung der Geometrie der Fläche frei wählen. Standardmäßig ist eine solche Fläche privat und wird nur durch Wunsch des Users öffentlich gemacht. Eine bereits erstellte Fläche kann aber im Nachhinein auf öffentlich umgeschaltet werden, nachdem sie z.B. die Geometrie und Position vom User finalisiert wurde. Dies ist vor allem nach Absprache zwischen anderen Experten denkbar, welche über Gruppen auch auf private, noch nicht öffentliche Benutzerflächen Zugriff haben um die Eigenschaften und Lage einer Fläche zu analysieren und über diese zu diskutieren.

5.2.2 Klassendiagramm, Datenbankdiagramm

Das Diagramm 5.11 schildert die Relation und Ausbau der wichtigsten Klassen des Systems. Dies zeigt, dass Benutzerflächen nur von registrierten Users erstellt werden können und immer einem solchen zugewiesen sind. Wird ein Benutzer aus dem System gelöscht, werden alle Flächen welche diesem Account zugeordnet sind aus dem System gelöscht.

5.2.3 Sequenzdiagramm

Die folgenden Sequenzdiagramme geben detaillierten Einblick in den Ablauf des Registrierung-, - und Loginvorgangs sowie das Laden der Public und Private Areas

und deren Darstellung im Client.

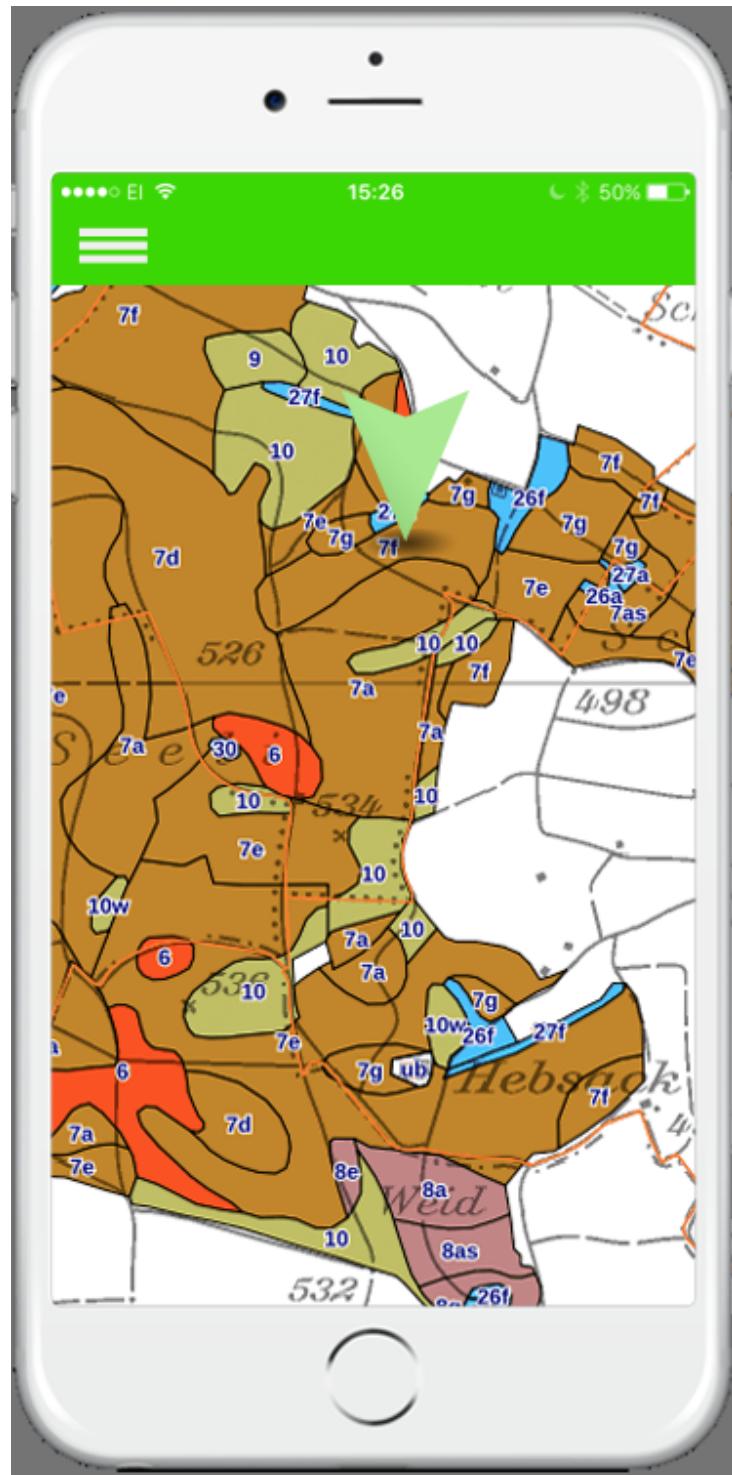


Abbildung 5.1: Mockup Screen 1

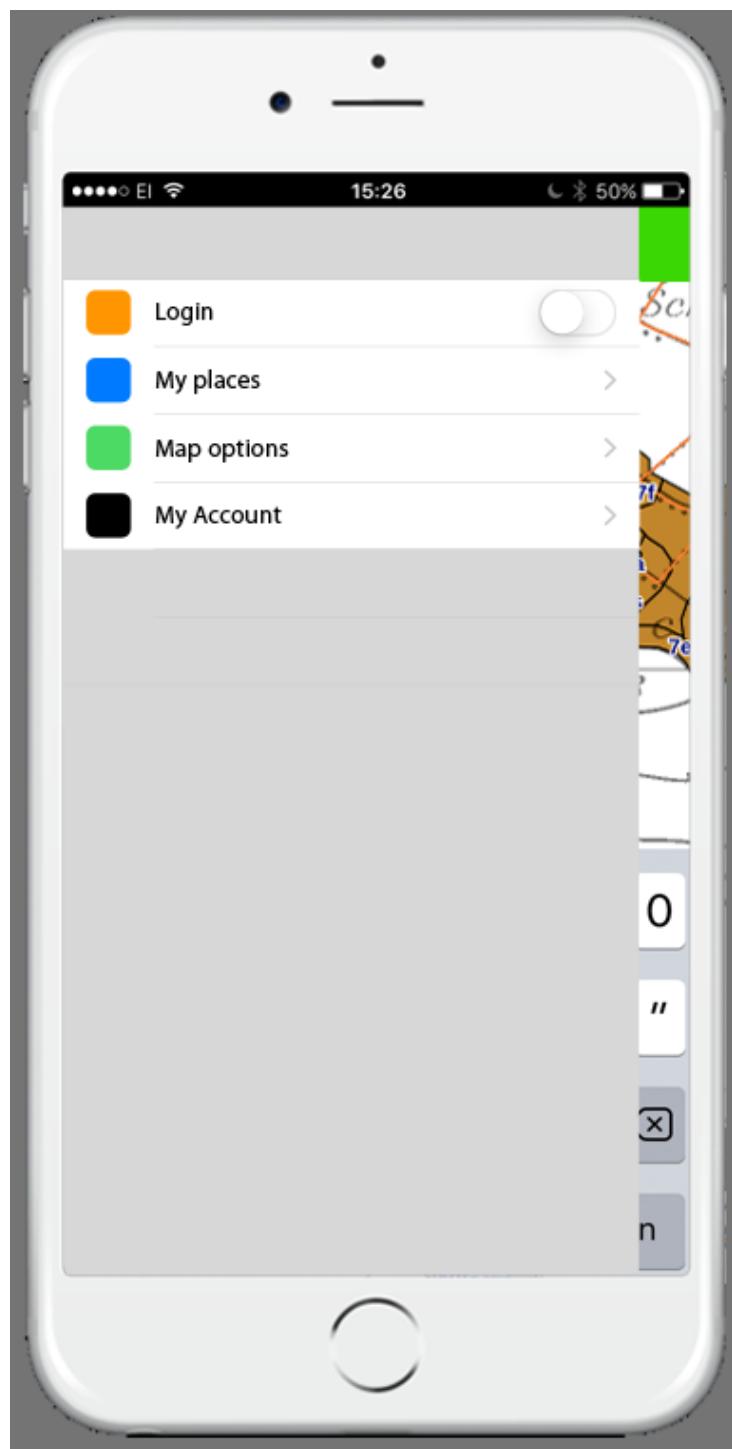


Abbildung 5.2: Mockup Screen 2

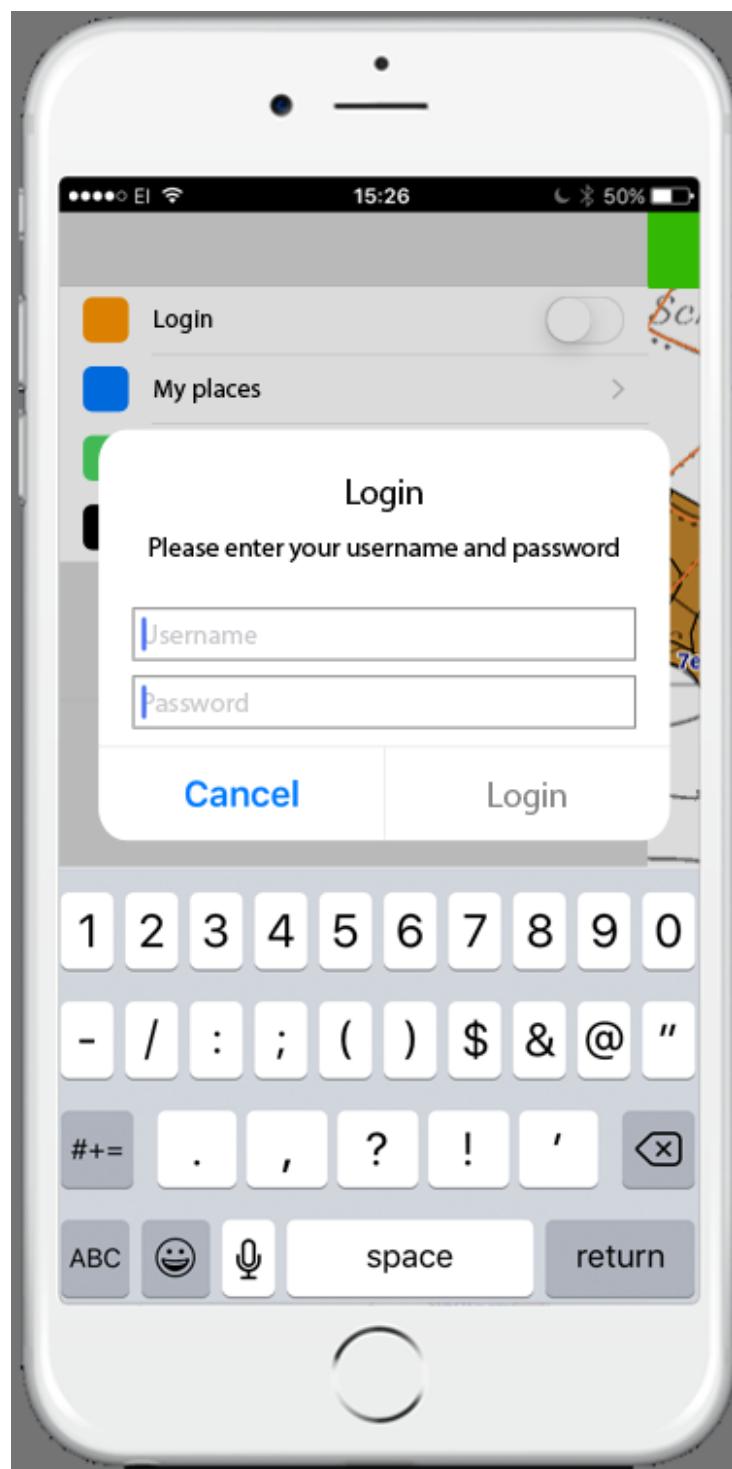


Abbildung 5.3: Mockup Screen 3

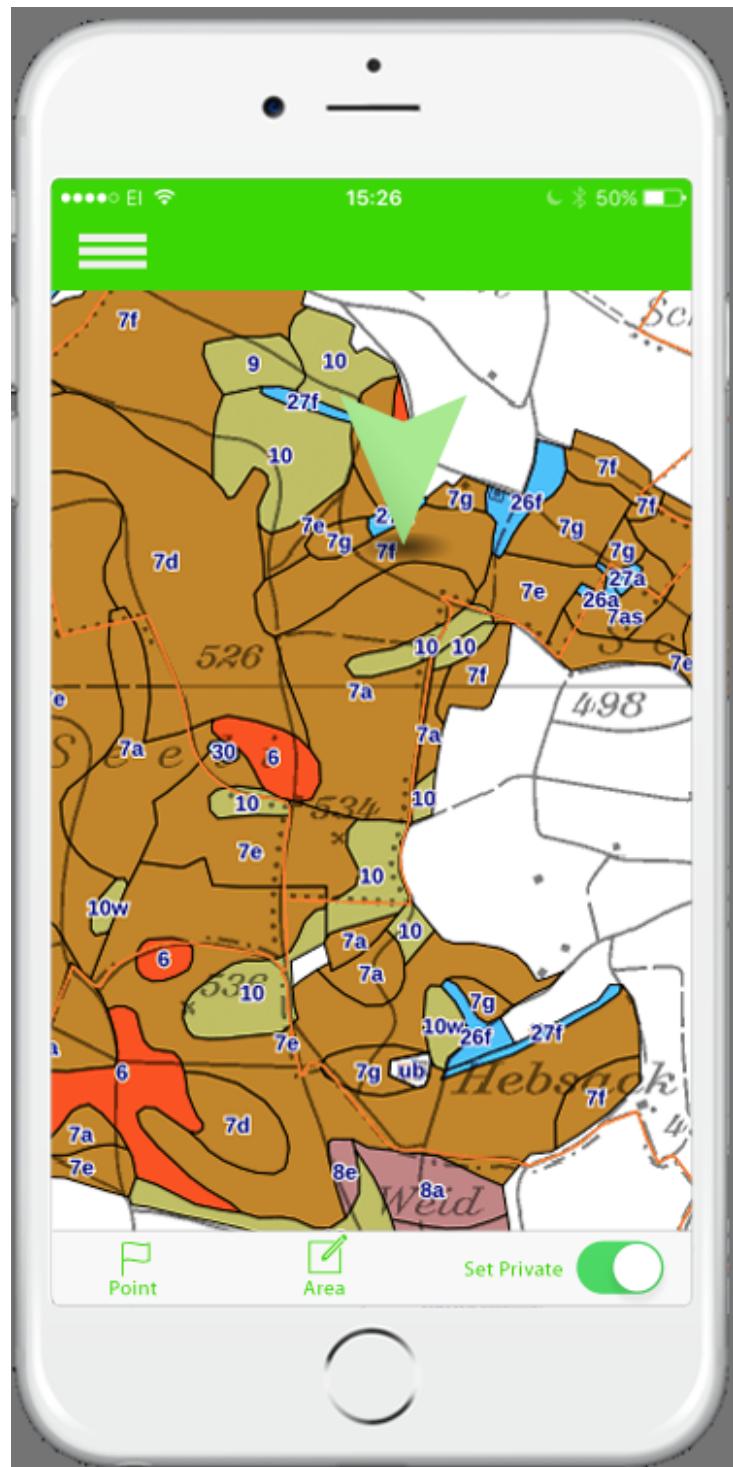


Abbildung 5.4: Mockup Screen 4

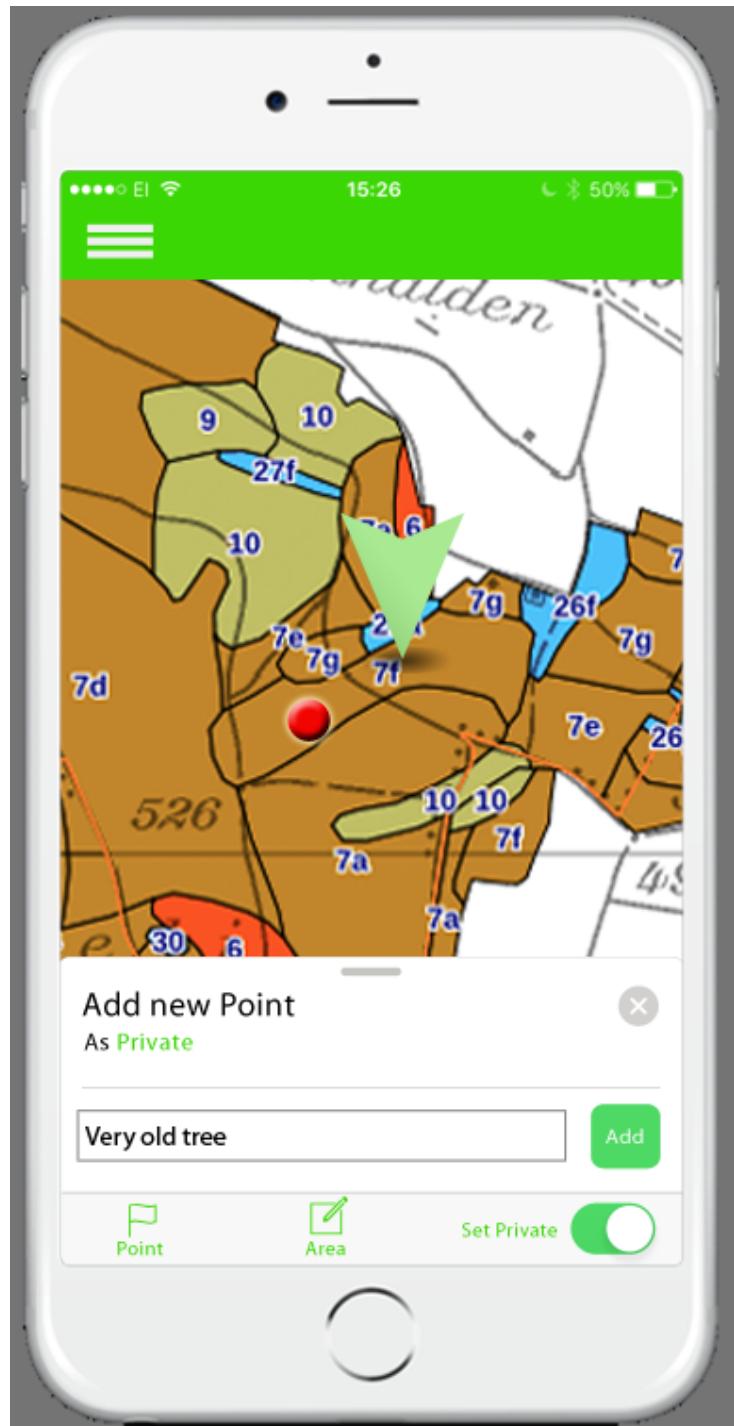


Abbildung 5.5: Mockup Screen 5

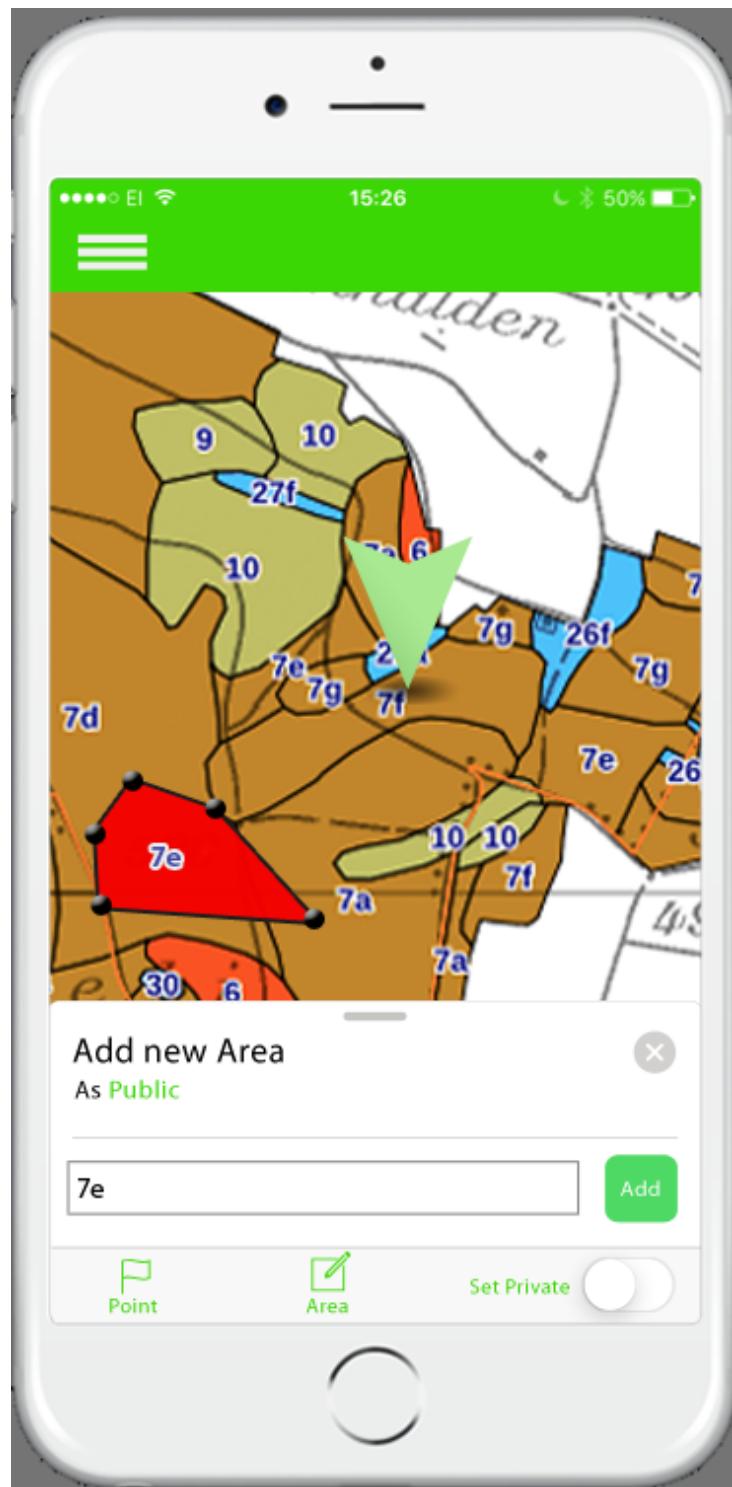


Abbildung 5.6: Mockup Screen 6

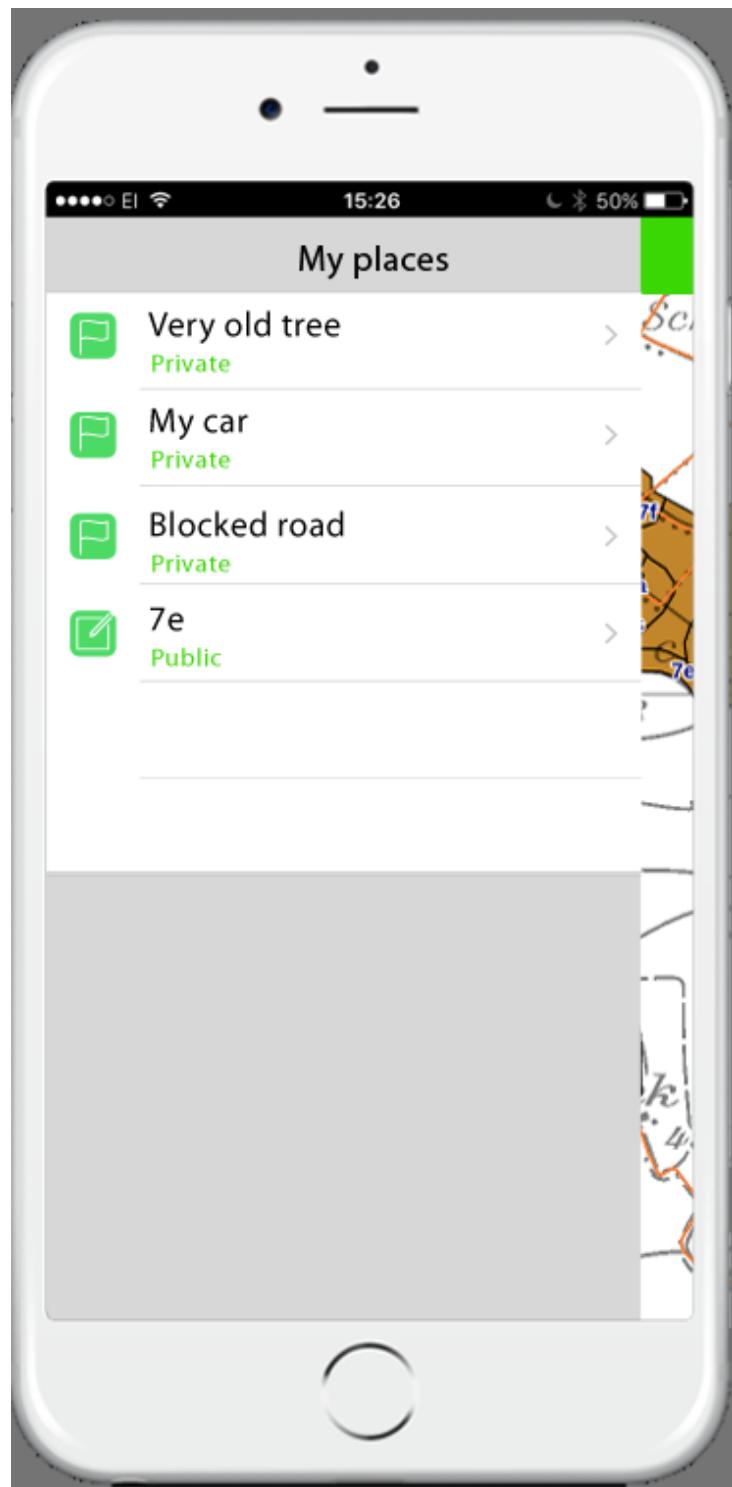


Abbildung 5.7: Mockup Screen 7

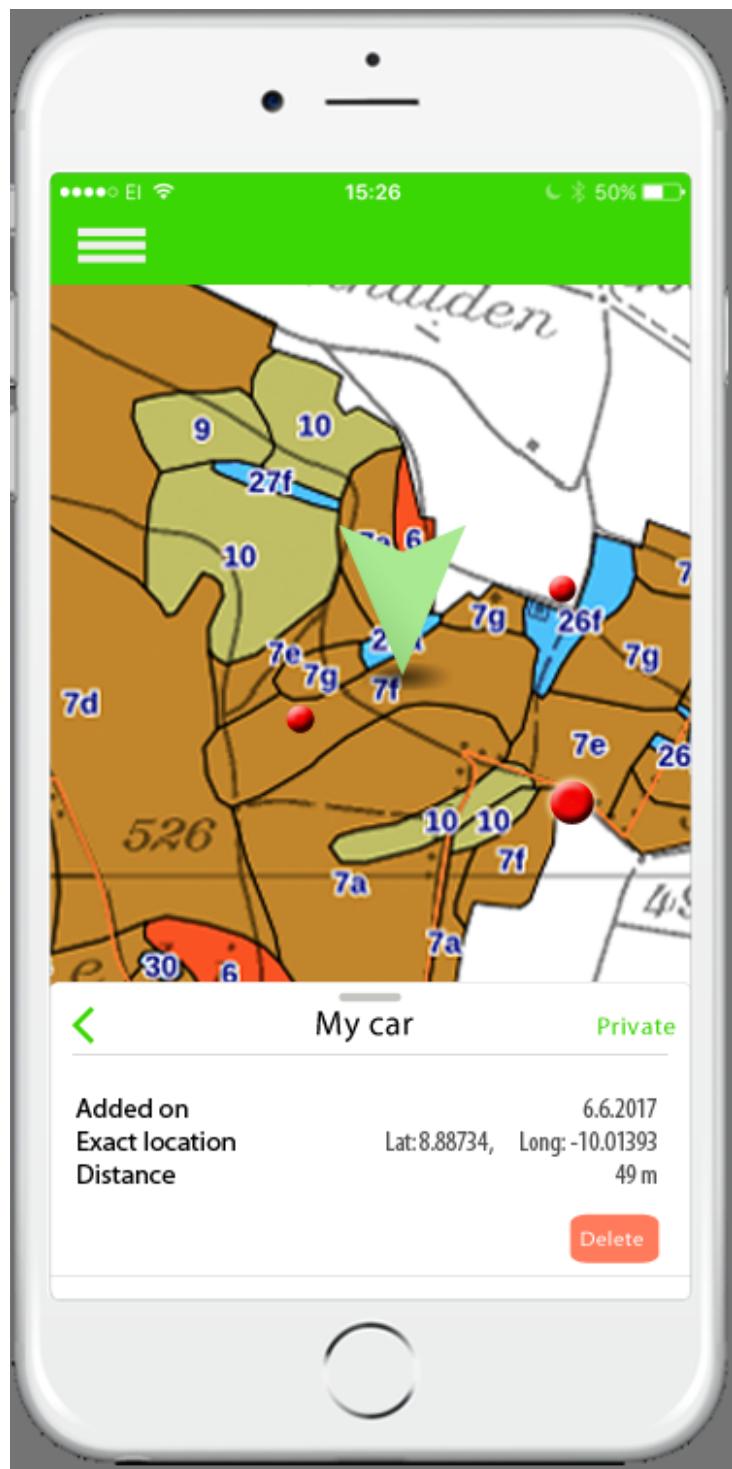


Abbildung 5.8: Mockup Screen 8

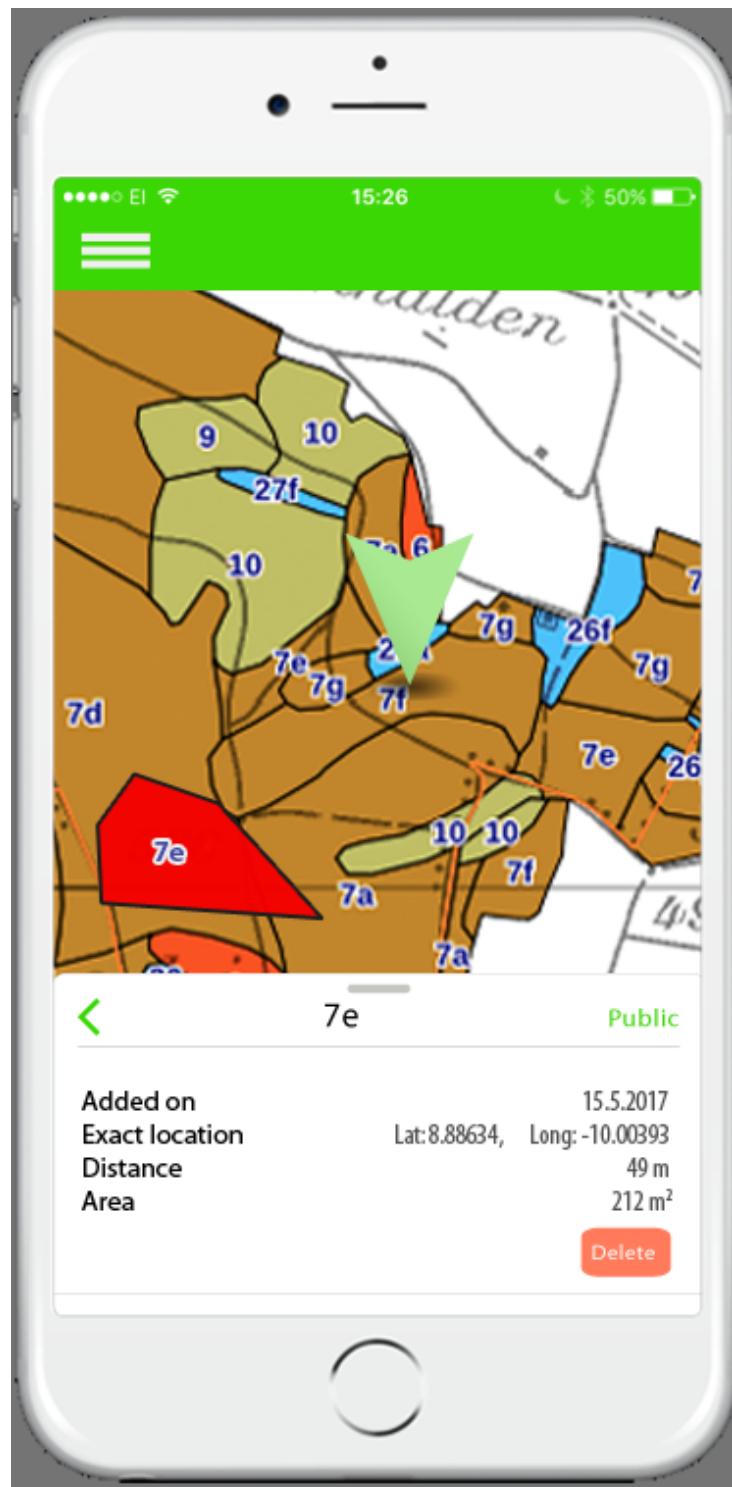


Abbildung 5.9: Mockup Screen 9

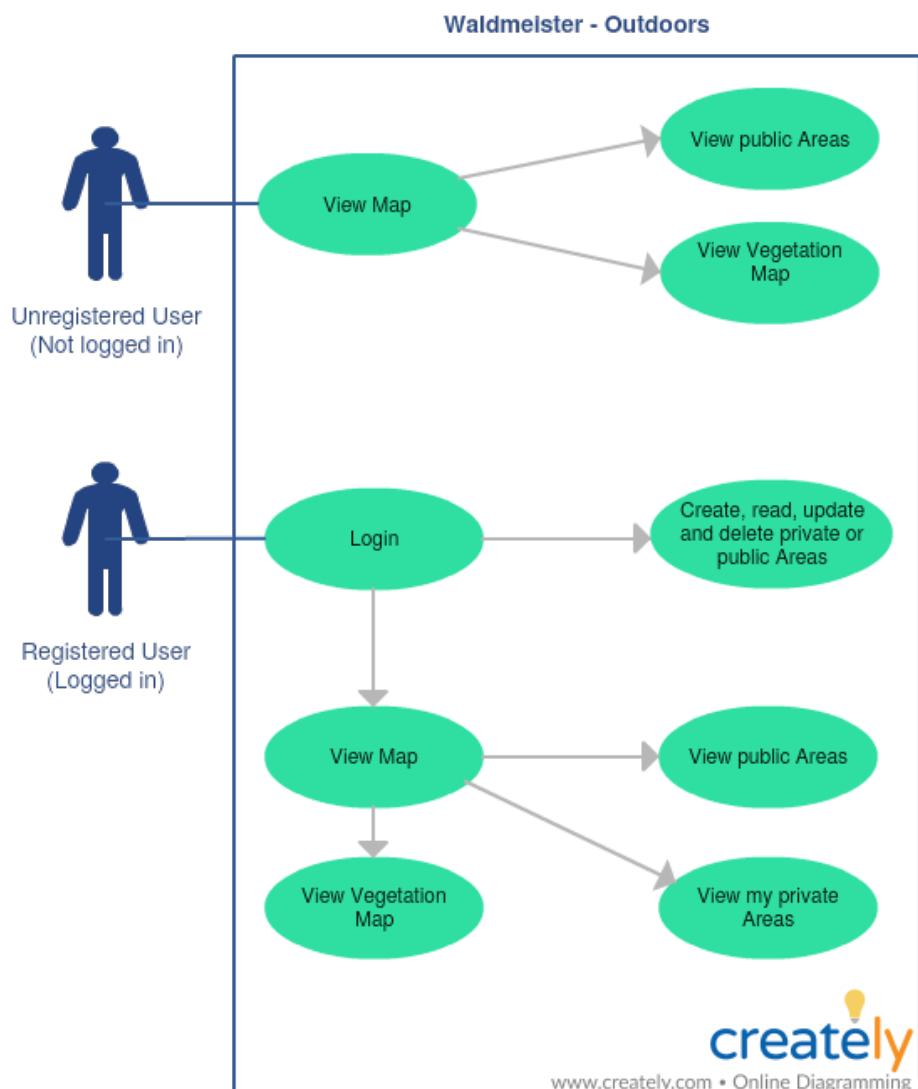


Abbildung 5.10: Use Case Diagram

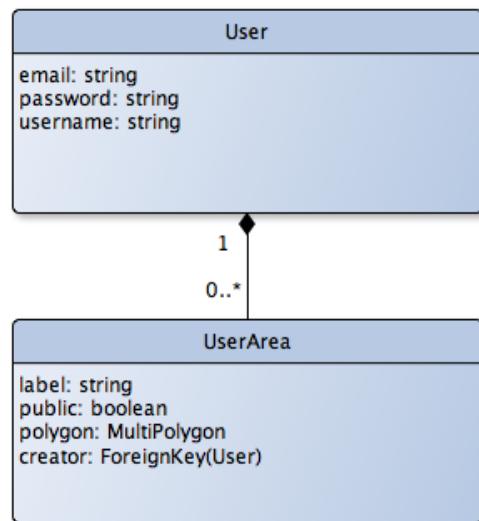


Abbildung 5.11: Klassendiagramm

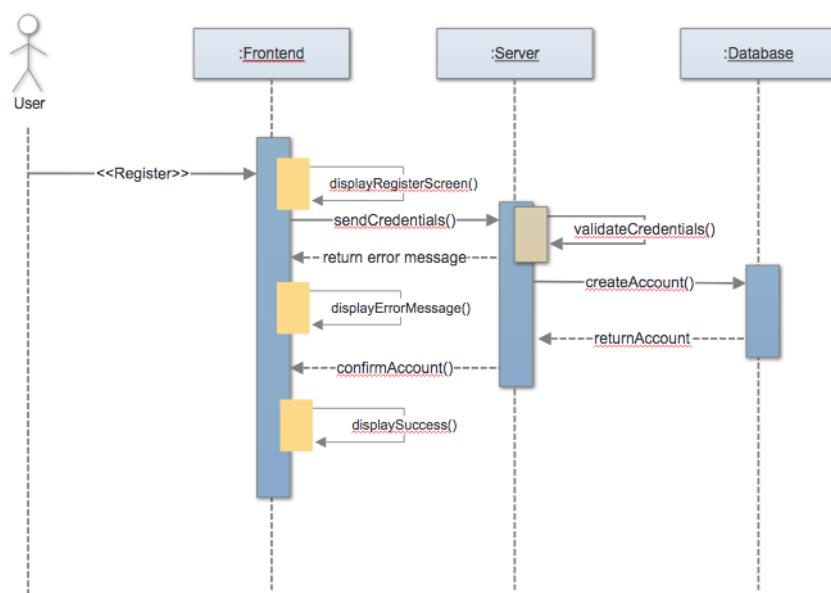


Abbildung 5.12: Sequenzdiagramm, Register

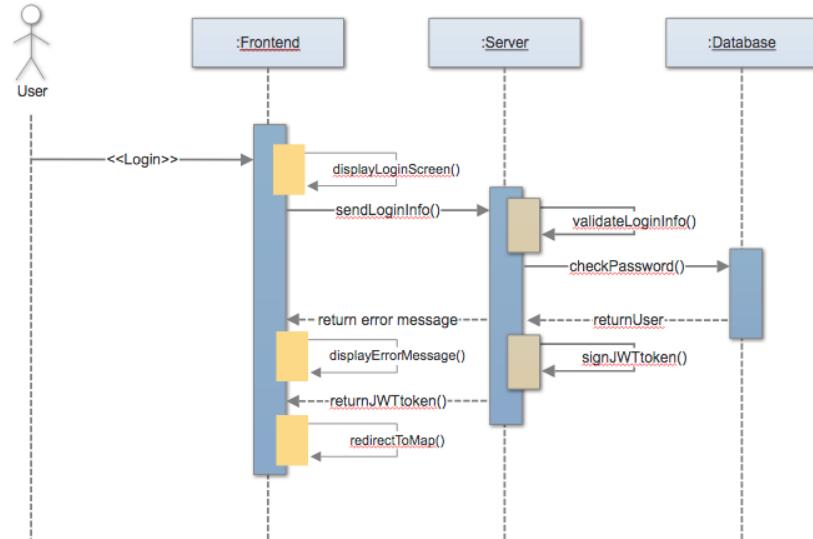


Abbildung 5.13: Sequenzdiagramm, Login

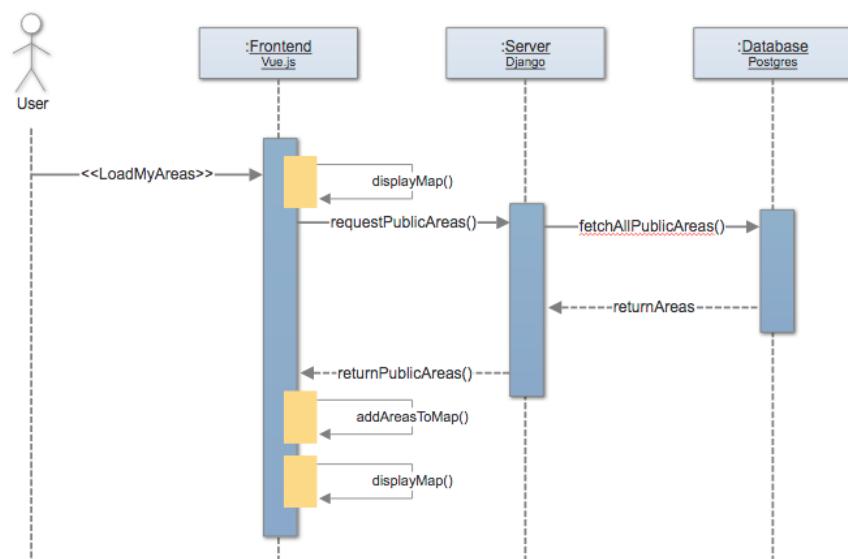


Abbildung 5.14: Sequenzdiagramm, Public Areas

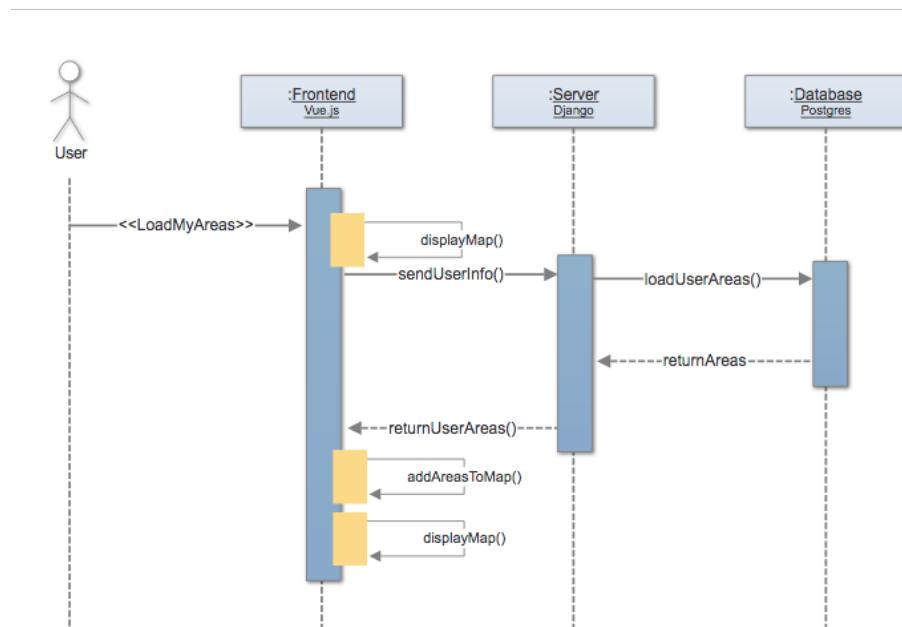


Abbildung 5.15: Sequenzdiagramm, My Areas

Kapitel 6

Results

Das Projekt wurde mit genannten Technologien umgesetzt und wird auf Github gehostet:

<https://github.com/dschmide/Waldmeister>

6.1 Discussion / Screenshots

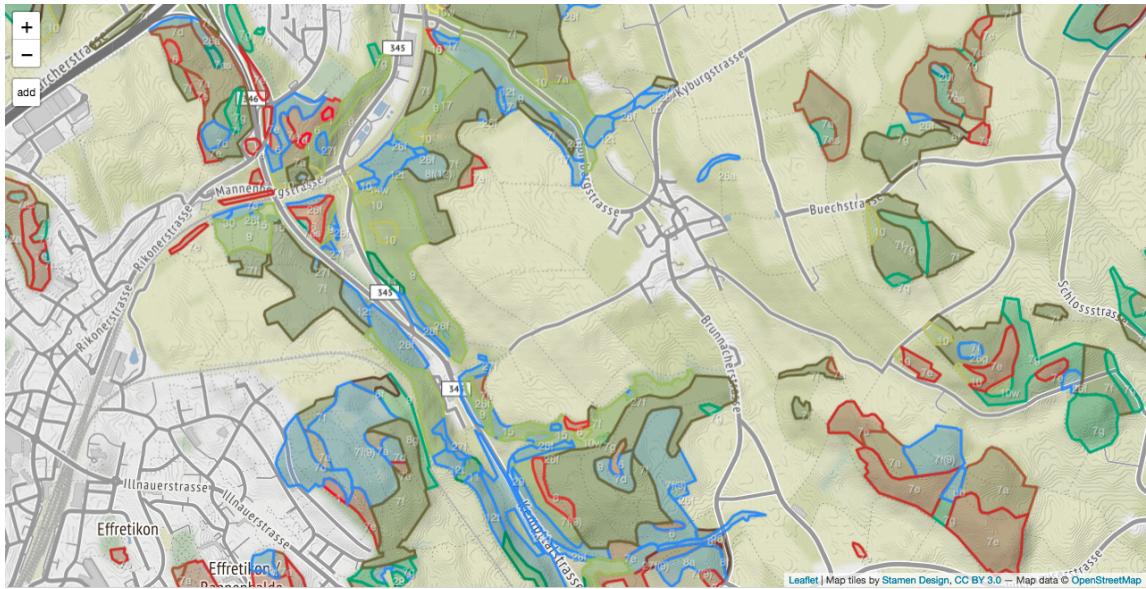


Abbildung 6.1: Screenshot Desktop, Vegetationskundliche Karte

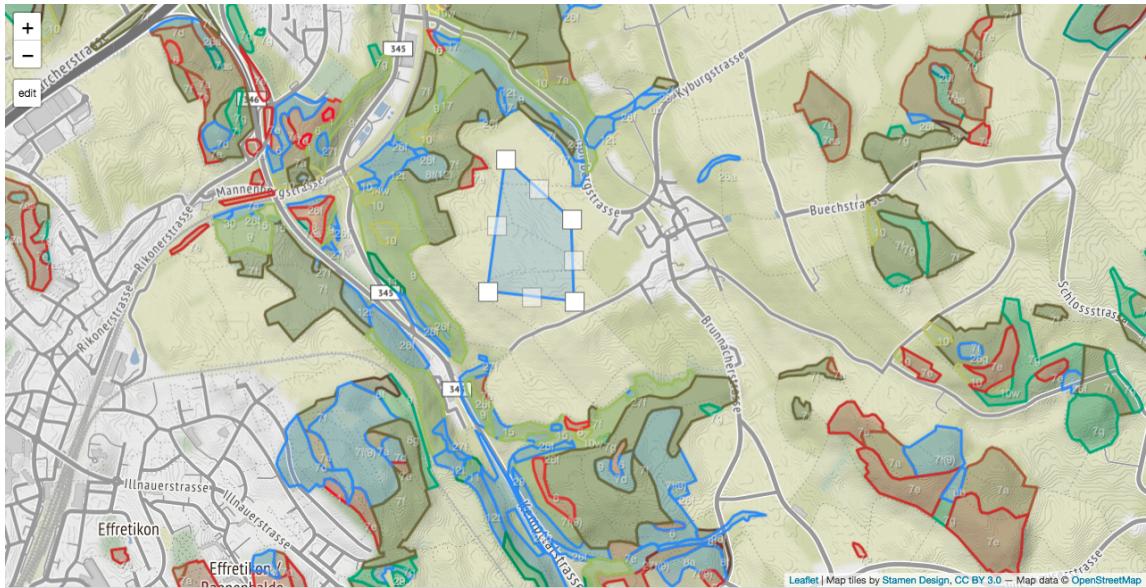


Abbildung 6.2: Screenshot Desktop, EditArea

Kapitel 7

Links

<https://github.com/dschmide/Waldmeister>

<https://maps.zh.ch?topic=WaldVKZHscale=18634x=2706590.13y=1251180.39srid=2056>

Kapitel 8

API Documentation

Abbildungsverzeichnis

4.1	Vuex Action-Mutations-State Diagram	16
4.2	Vuex private	17
5.1	Mockup Screen 1	20
5.2	Mockup Screen 2	21
5.3	Mockup Screen 3	22
5.4	Mockup Screen 4	23
5.5	Mockup Screen 5	24
5.6	Mockup Screen 6	25
5.7	Mockup Screen 7	26
5.8	Mockup Screen 8	27
5.9	Mockup Screen 9	28
5.10	Use Case Diagram	29
5.11	Klassendiagramm	30
5.12	Sequenzdiagramm, Register	30
5.13	Sequenzdiagramm, Login	31
5.14	Sequenzdiagramm, Public Areas	31
5.15	Sequenzdiagramm, My Areas	32
6.1	Screenshot Desktop, Vegetationskundliche Karte	34
6.2	Screenshot Desktop, EditArea	34

Literaturverzeichnis

- [BL] Tim Berners-Lee. Practical rdf. *Scientific American*.
- [BS08] Christian Bizer and Andreas Schultz. The berlin sparql benchmark. Technical report, Freie Universität Berlin, Web-based Systems Group, 2008.
- [Cha09] Raymond F Chamberlin, Donald D; Boyce. Sequel: A structured english query language. *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control*, 2007-06-09.
- [fS] International Organization for Standardization. Iso/iec 9075-1:2008.
- [JU10] Sören Auer Jörg Unbehauen, Claus Stadler. *Optimizing SPARQL-to-SQL Rewriting*. PhD thesis, University of Leipzig, 2010.
- [neo17] https://neo4j.com/developer/graph-db-vs-rdbms/#_from_relational_to_graph_databases, 6 2017.
- [rdb] <https://www.w3.org/2005/incubator/rdb2rdf/>.
- [Ste12] Pavel Stehule. *Historie projektu PostgreSQL*. 9 June 2012.
- [tbl] http://www-sop.inria.fr/acacia/fabien/lecture/licence_travaux_etude2002/thesemanticweb/.
- [use] <http://www.w3.org/2001/sw/sweo/public/usecases/>.