

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

Waldmeister - Outdoors

Ein Werkzeug zur Erfassung und Publikation von Waldstandorten

HSR Hochschule für Technik Rapperswil

Fall 2017

Author: Daniel Schmider

Supervisor: Prof. Stefan Keller

Kapitel 1

Abstract

"Waldmeister - Outdoors" ist eine Applikation, welche es ermöglicht, Waldstandorte in der Schweiz zu erforschen und zu erfassen. Die Applikation beschleunigt die digitale Erfassung und Publikation von Waldstandorten und erleichtert Experten die Arbeit im Feld. Über die Webapp können sich Benutzer registrieren, um öffentliche, bzw. private Benutzerflächen von mehreren Geräten aus zu erstellen, speichern und zu teilen. Diese Benutzerflächen können Waldstandorte, aber auch zusätzliche, relevante Informationen zu einem Standort beinhalten. Mithilfe der Geolocation wird die eigene Position bestimmt. Umliegende, bereits erfasste Waldstandorte und Benutzerflächen werden auf verschiedenen Ebenen auf einer Leaflet-basierten Karte angezeigt.

Keywords: Vue.JS, Django, Rest Framework, Leaflet, Leaflet editable, Geolocation, Progressive Webapp, GIS

Inhaltsverzeichnis

1	Abstract	1
2	Management summary	5
1	Problemstellung	5
2	Ziel der Arbeit	5
3	Ergebnisse	5
4	Ausblick	6
3	Teil 1 - Technischer Bericht	7
3.1	Einführung	8
3.1.1	Problemstellung, Vision	8
3.1.2	Ziele und Unterziele	8
3.1.3	Rahmenbedingungen	8
3.1.4	Vorgehen, Aufbau der Arbeit	8
3.2	Stand der Technik	8
3.2.1	GIS-Browser	8
3.2.2	ArcGIS online	8
3.2.3	Defizite	8
3.3	Bewertung	8
3.3.1	Kriterien	8
3.3.2	Schlussfolgerungen	8
3.4	Umsetzungskonzept	8
3.4.1	Lösungsansätze	8
3.5	Resultate	8
3.5.1	Zielerreichung	8
3.5.2	Ausblick und Weiterentwicklung	8
3.5.3	Persönliche Berichte	8
3.5.4	Danksagungen	8
4	Teil 2 - SW-Projektdokumentation	9
4.1	Vision	10
4.2	Anforderungsspezifikationen	10
4.2.1	Must-Haves	10

4.2.2	Optional	10
4.2.3	Use-Cases	10
4.2.4	Funktionale Anforderungen	10
4.2.5	Nicht-funktionale Anforderungen	10
4.2.6	Weitere Funktionen und Anforderungen	10
4.2.7	Detailspezifikationen	10
4.3	Analyse	10
4.3.1	Klassendiagramm	10
4.3.2	Domain Modell	10
4.3.3	Objektkatalog	10
4.4	Technologien	10
4.4.1	Django	10
4.4.2	VueJS	10
4.5	Design	10
4.5.1	Architektur	10
4.5.2	Objektkatalog	10
4.5.3	Package Struktur	10
4.5.4	Sequenz-Diagramm	10
4.5.5	UI Design	10
4.6	Implementation und Test	10
4.6.1	Implementation	10
4.7	Automatische Testverfahren	10
4.7.1	Jasmine	10
4.7.2	Karma	10
4.7.3	Travis	10
4.8	Manuelle Tests	10
4.8.1	Testfälle	10
4.8.2	Ziel	10
4.8.3	Reproduktion	10
4.9	Resultate und Weiterentwicklung	10
4.9.1	Resultate	10
4.9.2	Möglichkeiten der Weiterentwicklung	10
4.9.3	Vorgehen	10
4.10	Projektmanagement	10
4.10.1	Allgemeines	10
4.10.2	Meilensteinplanung	10
4.10.3	Releases	10
4.10.4	Issues	10
4.10.5	Prototypen	10
4.10.6	Prozessmodell	10
4.10.7	Aufwandschätzung	10
4.10.8	Risiken	10
4.11	Projektmonitoring	10

4.11.1 Soll-Ist-Zeitvergleich	10
4.11.2 Code Statistics	10
4.12 Softwaredokumentation	10
4.12.1 Installation	10
4.12.2 Tutorial, Handbuch	10
4.12.3 Referenzhandbuch	10

Kapitel 2

Management summary

1 Problemstellung

Je nach Untergrund, Bodeneigenschaften, Gelände sowie Klima gedeihen in der Schweiz unterschiedliche Typen von Wäldern. Seit einigen Jahrzehnten werden diese Typen von Experten erhoben und kartiert. Es wurden dabei verschiedene typisierte Waldstandorte festgelegt. Aktuell werden Karten, die im Auftrag der Kantone von Experten angefertigt wurden, nur in grossen Intervallen revidiert, wobei sie oft auch nicht flächendeckend vorhanden sind (z.B. in den Kantonen GR, VS, BE). Einer der Gründe dafür sind u.a. die hohen Kosten, die eine Analyse im Feld mit sich bringt. Zudem ist die Erfassung und Nachführung der Karten 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 oder von staatlichen Institutionen digitalisiert werden, bevor es an den Arbeitgeber geschickt werden und später auf kantonal isolierten Plattformen publiziert werden kann.

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 ermöglichen sowie öffentliche und private Informationen in Form von Flächen und Punkten. Auf einer Basis - Karte wird mittels GPS die eigene Position angezeigt. Darüber werden umliegende, bereits erfasste Waldstandorte, öffentliche Flächen anderer sowie die eigenen, privaten Flächen dargestellt. Diese Flächen können Waldstandorte beschreiben oder aber zusätzliche Informationen über den Standort beinhalten, z.B. eine speziell gekennzeichnete Beobachtungsfläche.

3 Ergebnisse

Nach einer Evaluation eines Prototyps, erstellt mithilfe eines kommerziellen Produkts, und der Erstellung von Mockups, wurde ein eigenes Webapp 'Waldmeister Outdoors' realisiert. Durch diese App 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. Die Webapp wurde für mobile Geräte optimiert und die gewünschten Funktionen wurden umgesetzt. Registrierte Benutzer können Benutzerflächen in Form von Polygonen direkt auf der angezeigten Map erstellen, mit zusätzlichen Informationen versehen und auf einem Server speichern.

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.

"Waldmeister - Outdoors" hat das Potential in der Schweiz ein verbreitetes Tool zur Kartierung und Beobachtung von Waldflächen zu werden und stellt eine bereits gefragte Erweiterung der beliebten "Waldmeister" App für mobile Geräte dar.

Kapitel 3

Teil 1 - Technischer Bericht

3.1 Einführung

3.1.1 Problemstellung, Vision

3.1.2 Ziele und Unterziele

3.1.3 Rahmenbedingungen

3.1.4 Vorgehen, Aufbau der Arbeit

3.2 Stand der Technik

3.2.1 GIS-Browser

3.2.2 ArcGIS online

Technologien von ESRI (Environmental Systems Research Institute) und insbesondere ArcGIS (GIS; Geografisches Informationssystem) online wurden recherchiert, um einen funktionierenden Prototypen mit offline-caching zu erstellen. Hintergrundkarten (in Form eines Tile-Layers) können auf dem Gerät zwischengespeichert werden. Die Erstellung von editierbaren Vektorlayern funktioniert auch beim offline Betrieb und können später, sobald wieder eine stabile Internetverbindung besteht, synchronisiert werden. Dieses Verhalten kann bei einer PWA durch Service-Worker rekreiert werden.

Ein GeoJSON (Geo JavaScript Object Notation) file kann ebenfalls auf der online Plattform von ArcGIS hochgeladen werden und danach auf der Map dargestellt werden. Das Layer Styling kann so konfiguriert werden, dass die Farbe eines Polygons dem Typ des jeweiligen Waldstandorts entspricht.

3.2.3 Defizite

3.3 Bewertung

3.3.1 Kriterien

3.3.2 Schlussfolgerungen

3.4 Umsetzungskonzept

3.4.1 Lösungsansätze

Backend

Frontend

Datenbank

JavaScript Libraries

Python Pakete

Werkzeuge und Tools

Testing

3.5 Resultate

3.5.1 Zielerreichung

3.5.2 Ausblick und Weiterentwicklung

3.5.3 Persönliche Berichte

3.5.4 Danksagungen

Kapitel 4

Teil 2 - SW-Projektdokumentation

4.1 Vision

4.2 Anforderungsspezifikationen

4.2.1 Must-Haves

4.2.2 Optional

4.2.3 Use-Cases

4.2.4 Funktionale Anforderungen

4.2.5 Nicht-funktionale Anforderungen

4.2.6 Weitere Funktionen und Anforderungen

4.2.7 Detailspezifikationen

4.3 Analyse

4.3.1 Klassendiagramm

4.3.2 Domain Modell

4.3.3 Objektkatalog

4.4 Technologien

4.4.1 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 eine eigene Namensgebung verwendet. [AH07] In diesem Projekt verwendet Django eine PostgreSQL Datenbank um Daten persistent zu machen. Django wird ebenfalls dazu verwendet um User einen Account zu geben, damit nur sie selbst Zugriff auf Ihre privaten Benutzerflächen haben, oder um öffentlich erstellte Benutzerflächen mit anderen zu teilen.

Django Rest Framework

Eine SPA kommuniziert hauptsächlich über API Schnittstellen mit dem Server. Hier kommt auf dem Server das Django Rest-Framework (DRF) zum Einsatz.

Es bietet ein sehr flexibles System zur Erstellung von RESTful Web-APIs. Das DRF bietet die Möglichkeit CRUD Operationen auf eine Ressource auszuführen. "Waldmeister - Outdoors" verwendet die REST Api beispielsweise um usergenerierte Flächen, Pfade oder Punkte in der 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.

4.4.2 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. [Har01] [JF08]

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. Anhand des Django Models werden Geodaten in einem gewählten Format gespeichert. Standardmässig wird von Django die Spatial Reference Identifier (SRID) Nummer 4326 (World Geodetic System, WGS84) verwendet. Dieses System benutzt Zahlen von -180, -90 bis 180, 90 um Positionen auf der Erde (in Latitude und Longitude) zu beschreiben.

Postgis wird ebenfalls dazu verwendet, dass die Daten, welche per REST-Schnittstelle an den Client geschickt werden, in richtigem Format (Multipolygone in GeoJSON) übermittelt werden.

4.4.3 VueJS

Vue.JS ist ein JavaScript Framework, welches sich zum Erstellen von Single-Page-Webapplikationen in Form einer PWA eignet. Es wurde im Jahr 2013 erstmals veröffentlicht und wurde am 19. Dezember 2017 auf die aktuellste Version 2.5.13 gepatcht. Vue.JS folgt einer Variation des Model-View-Controller-Entwurfsmusters, 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 oder React. Vue.JS kann von Entwicklern eingesetzt werden welche HTML und JavaScript beherrschen und erfordert keine weiteren Webtechnologien. Vue.JS setzt eine Website 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 (Cascading Style Sheets) 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 aus 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, der 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>
```

Separation of Concern

Was ist gemeint mit Separation of Concern (SoC) und bricht der Aufbau von Single-File-Components nicht dieses Pattern? Eine bekannte Vorgehensweise bei Softwareengineering ist es, ein Computerprogramm in logische Abschnitte einzuteilen und zu separieren. Diese Teile sollten sich um einen Zweck oder Belang (Concern) kümmern. Dies heisst jedoch nicht, dass die verschiedenen Dateitypen unbedingt in separate Dateien aufgeteilt werden. 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. [Vue]

Auf diesem Weg sind Komponenten (Template, die Logik und das Styling) 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 bzw. Verhalten (JavaScript).

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 macht.

In der HTML Definition der Hauptkomponente 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 angezeigt welche die Basis der Webseite darstellt. Methoden von Komponenten können bewirken, dass sich der Inhalt verändert, welcher an der Stelle des Router-Views angezeigt wird. Menüpunkte 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 zu empfehlen, die offizielle Vue-Router Library zu verwenden.

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. 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 ihren 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 = ''
  }
}
```

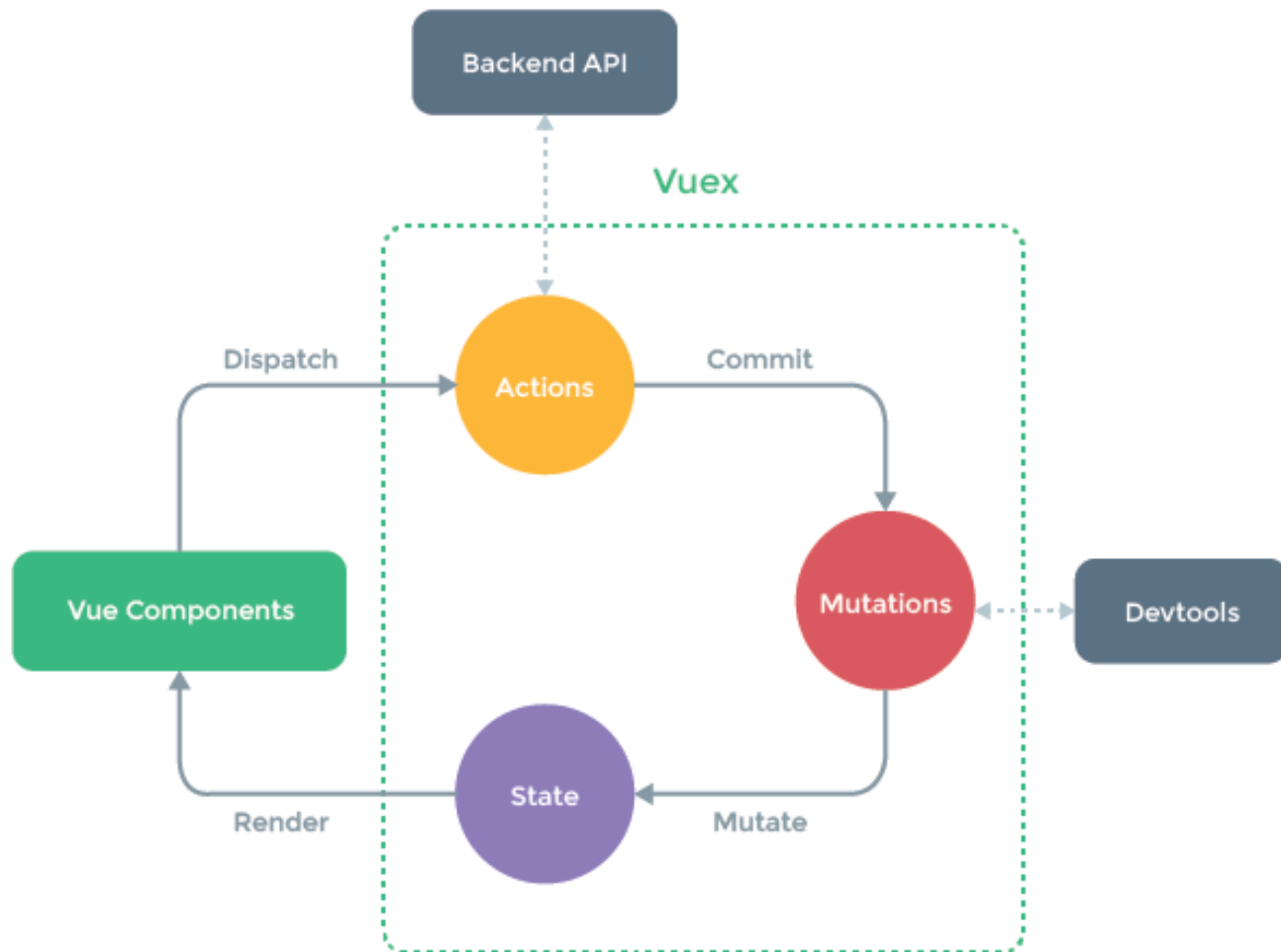


Abbildung 4.1: Vuex Action-Mutations-State Diagram

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

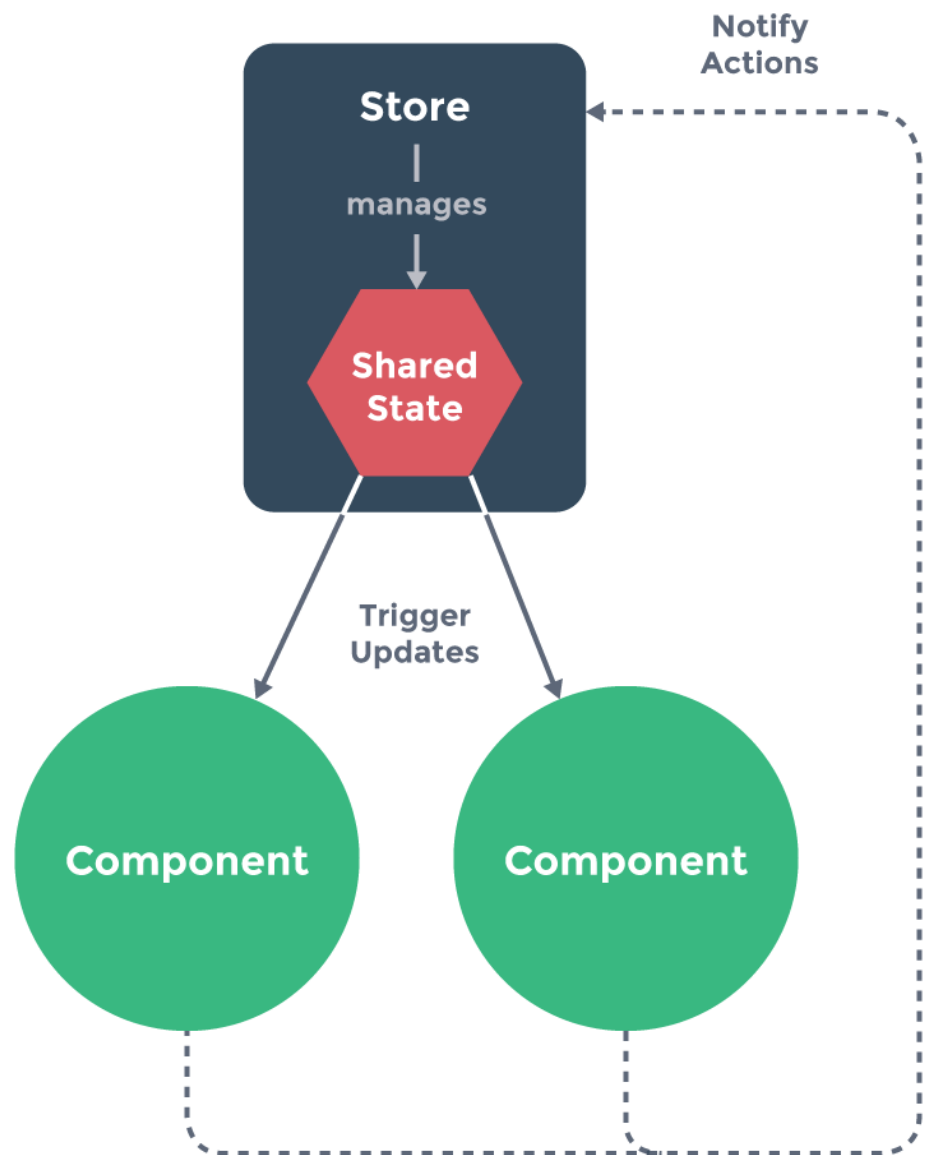


Abbildung 4.2: Vuex private

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

4.5 Design

4.5.1 Architektur

4.5.2 Objektkatalog

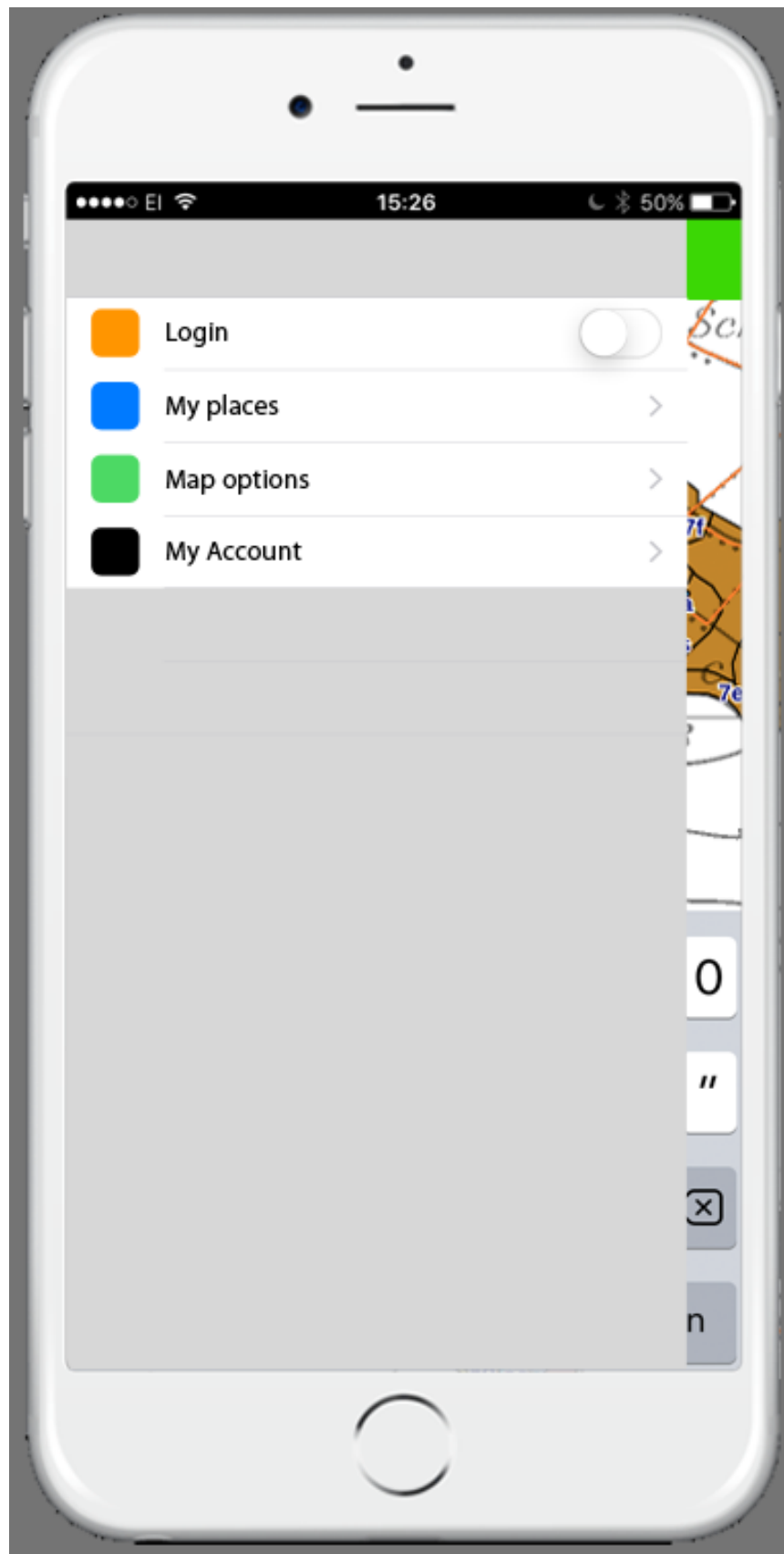
4.5.3 Package Struktur

4.5.4 Sequenz-Diagramm

4.5.5 UI Design

Mockups

Die Mockups wurden vor der Implementation erstellt, um 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 ausgewählten Objekts zusätzliche Informationen.



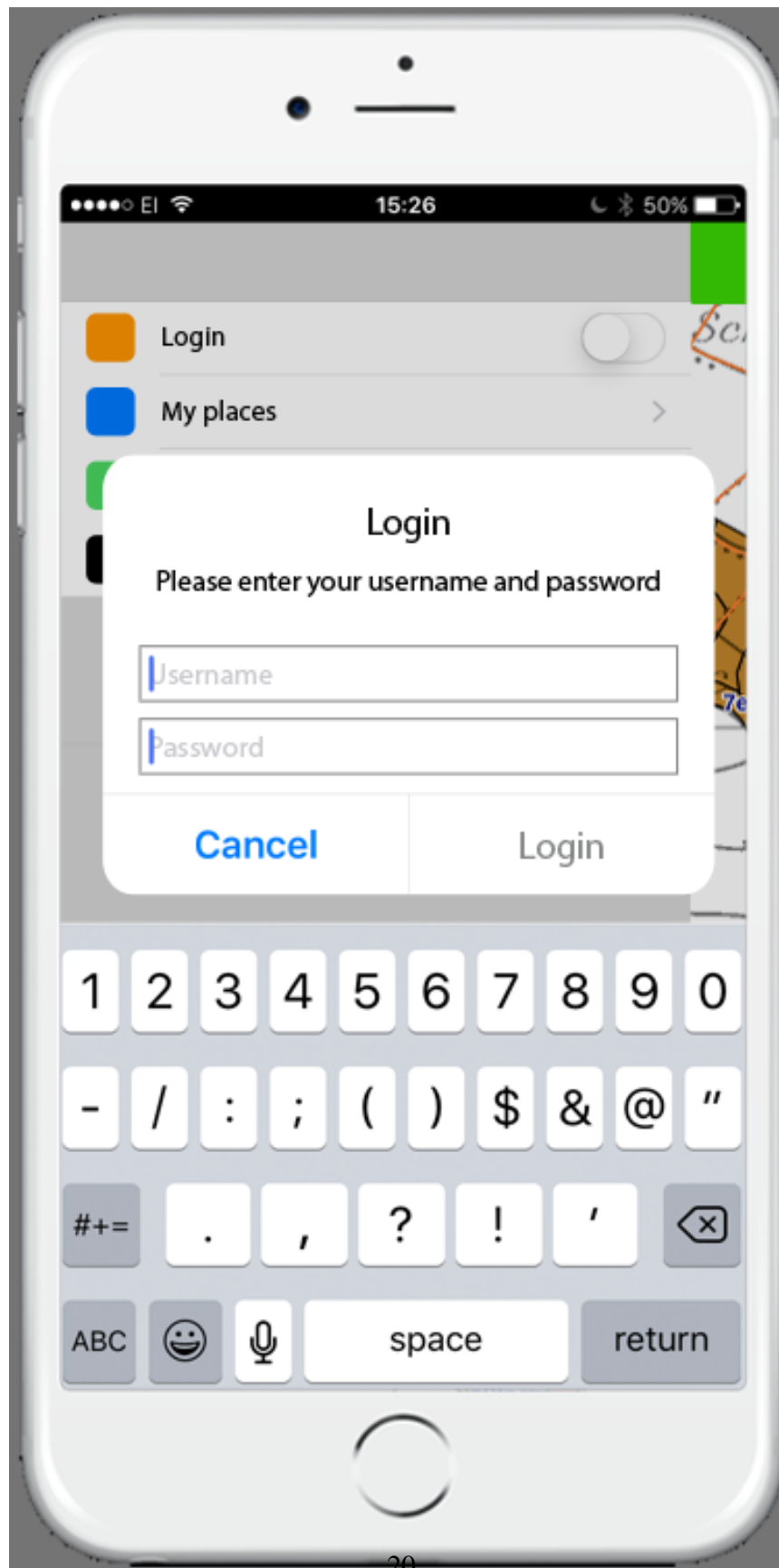


Abbildung 4.5: Mockup Screen 3

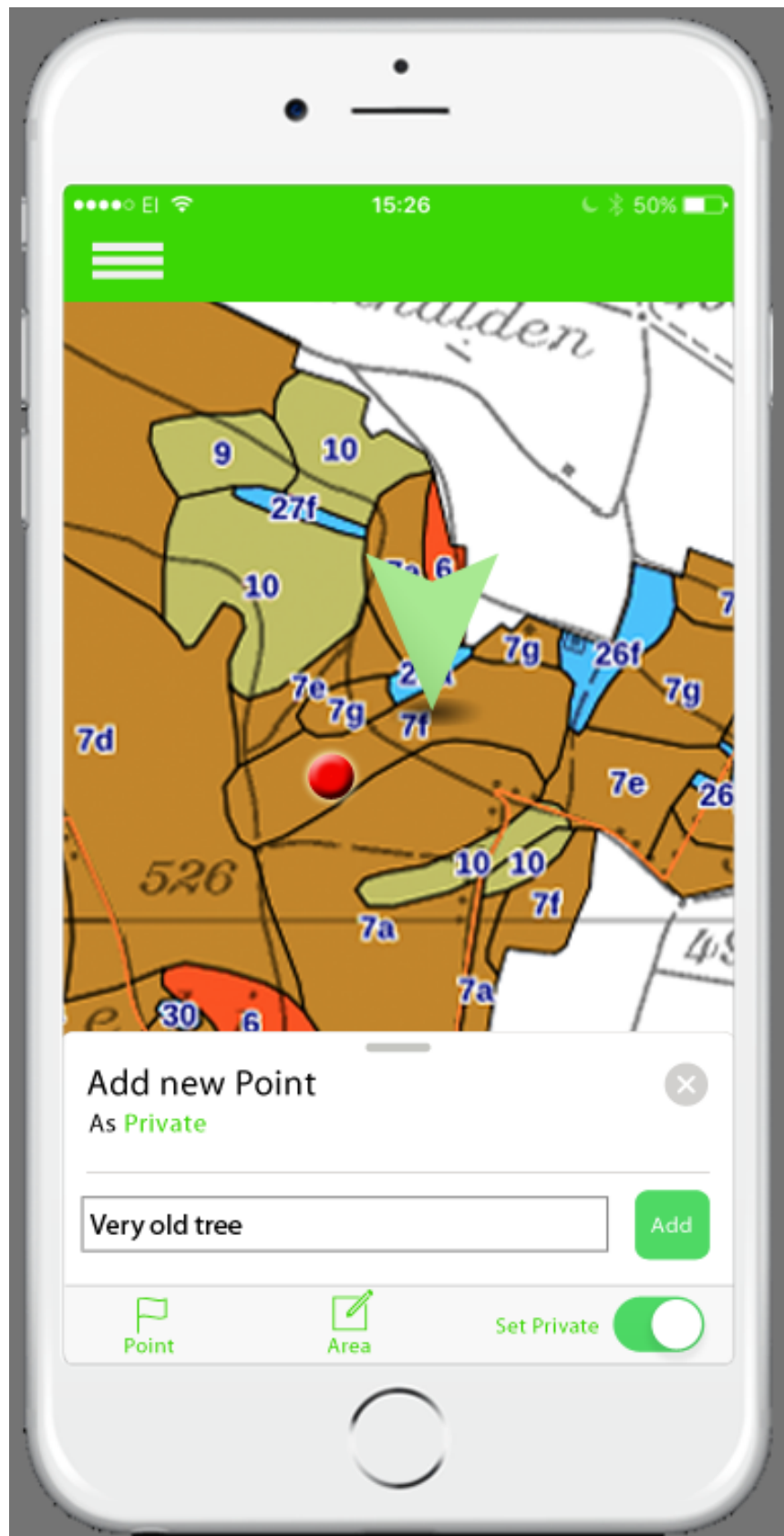


Abbildung 4.7: Mockup Screen 5

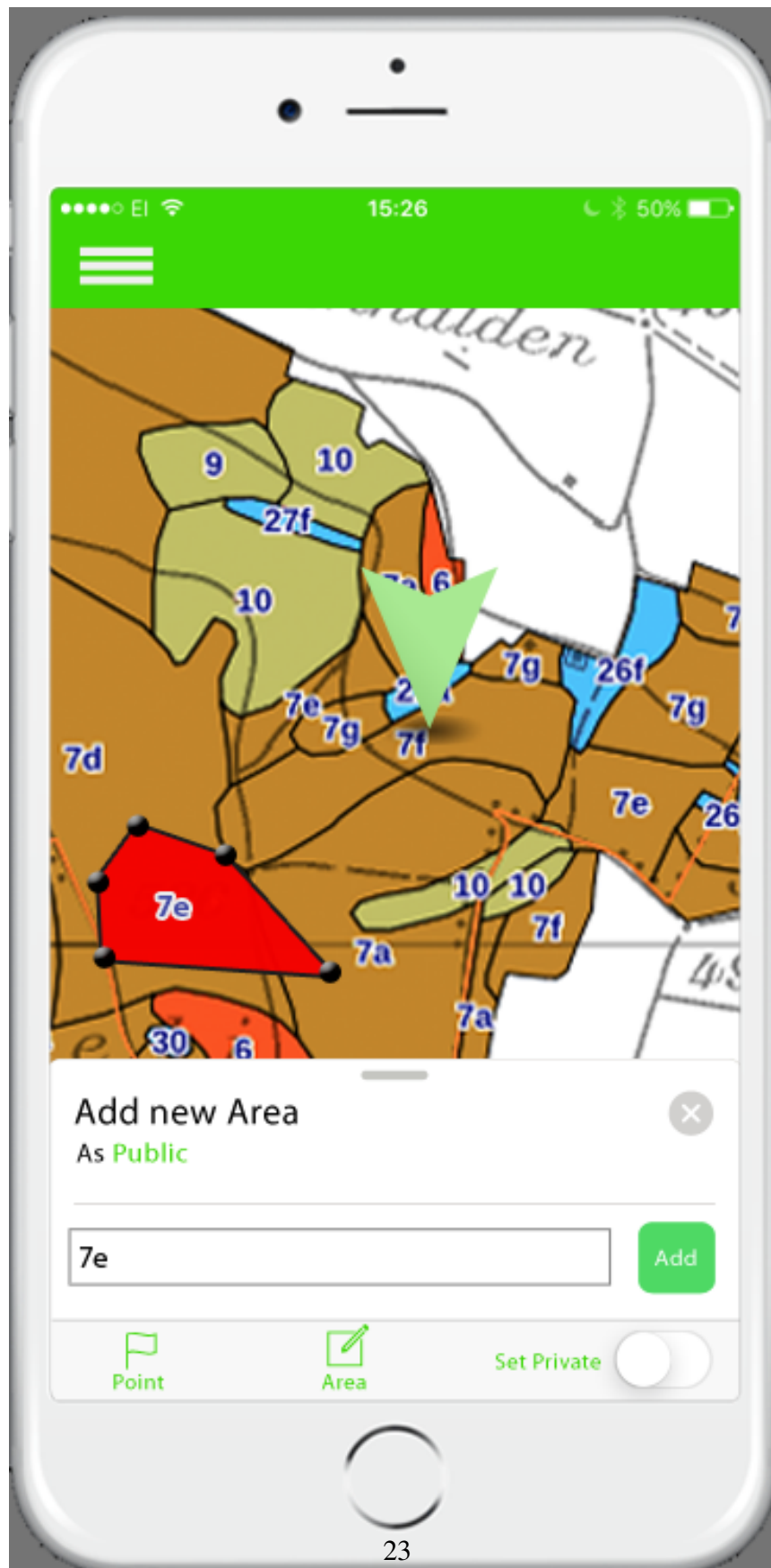


Abbildung 4.8: Mockup Screen 6

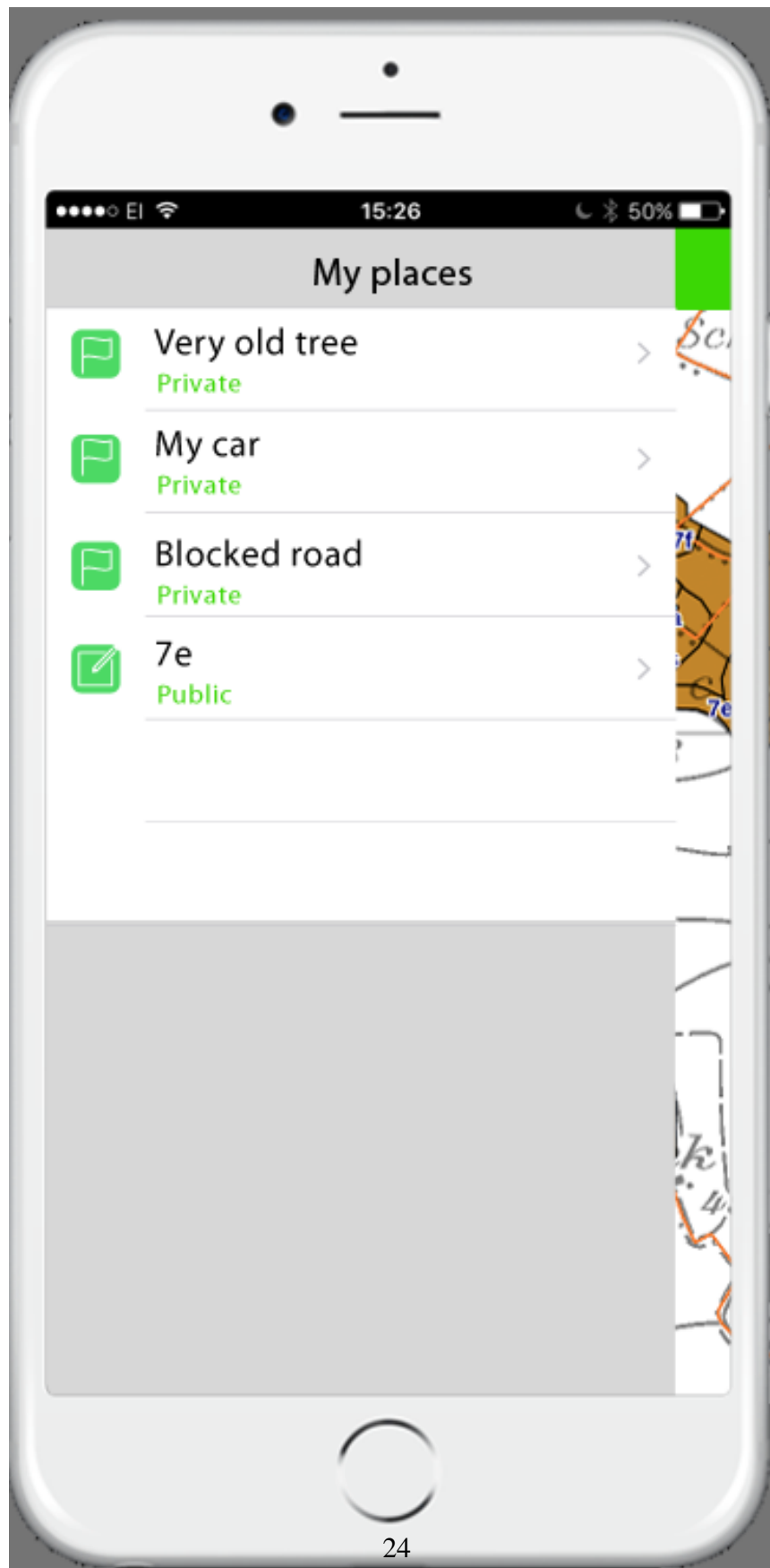


Abbildung 4.9: Mockup Screen 7

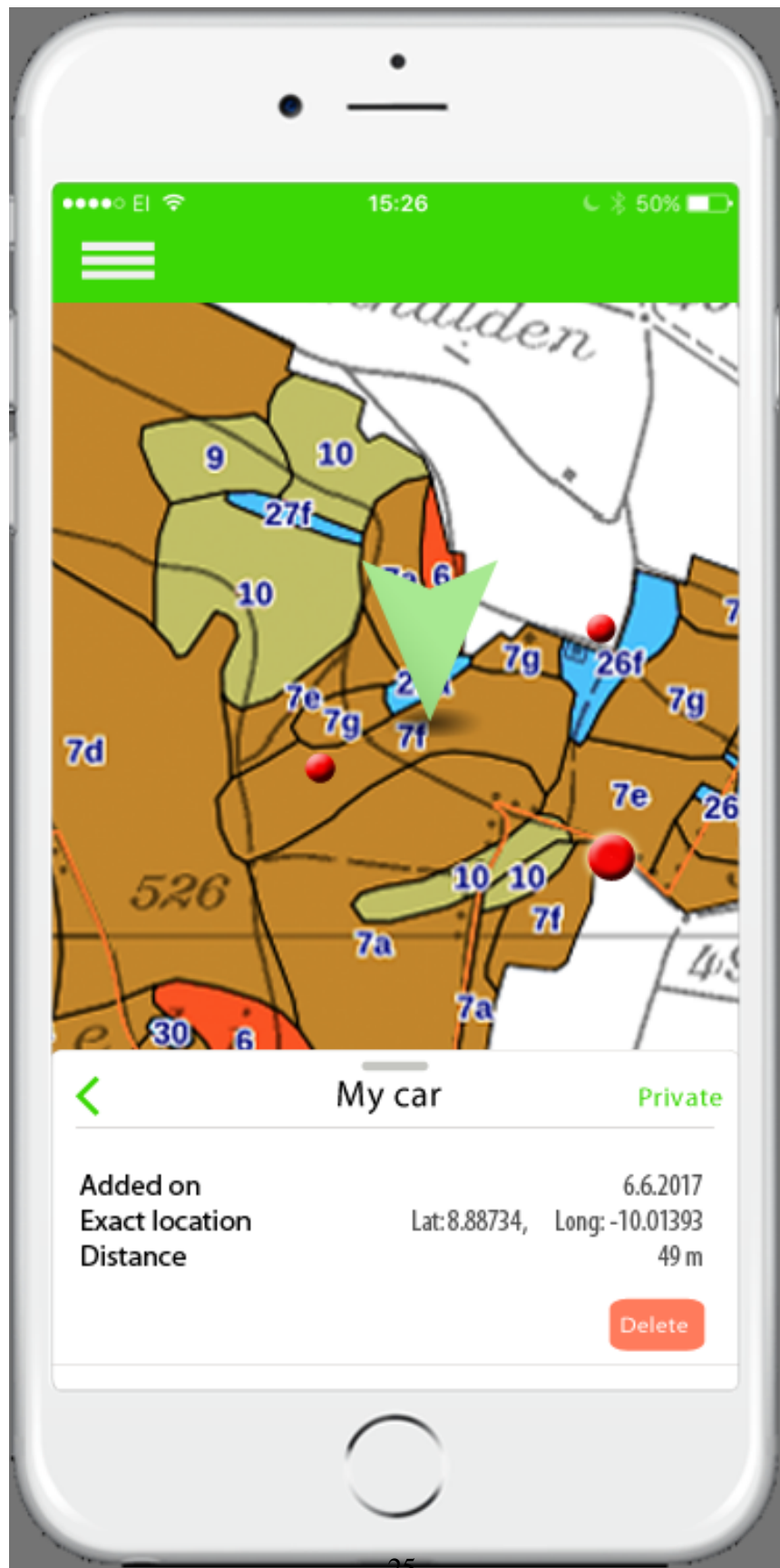


Abbildung 4.10: Mockup Screen 8

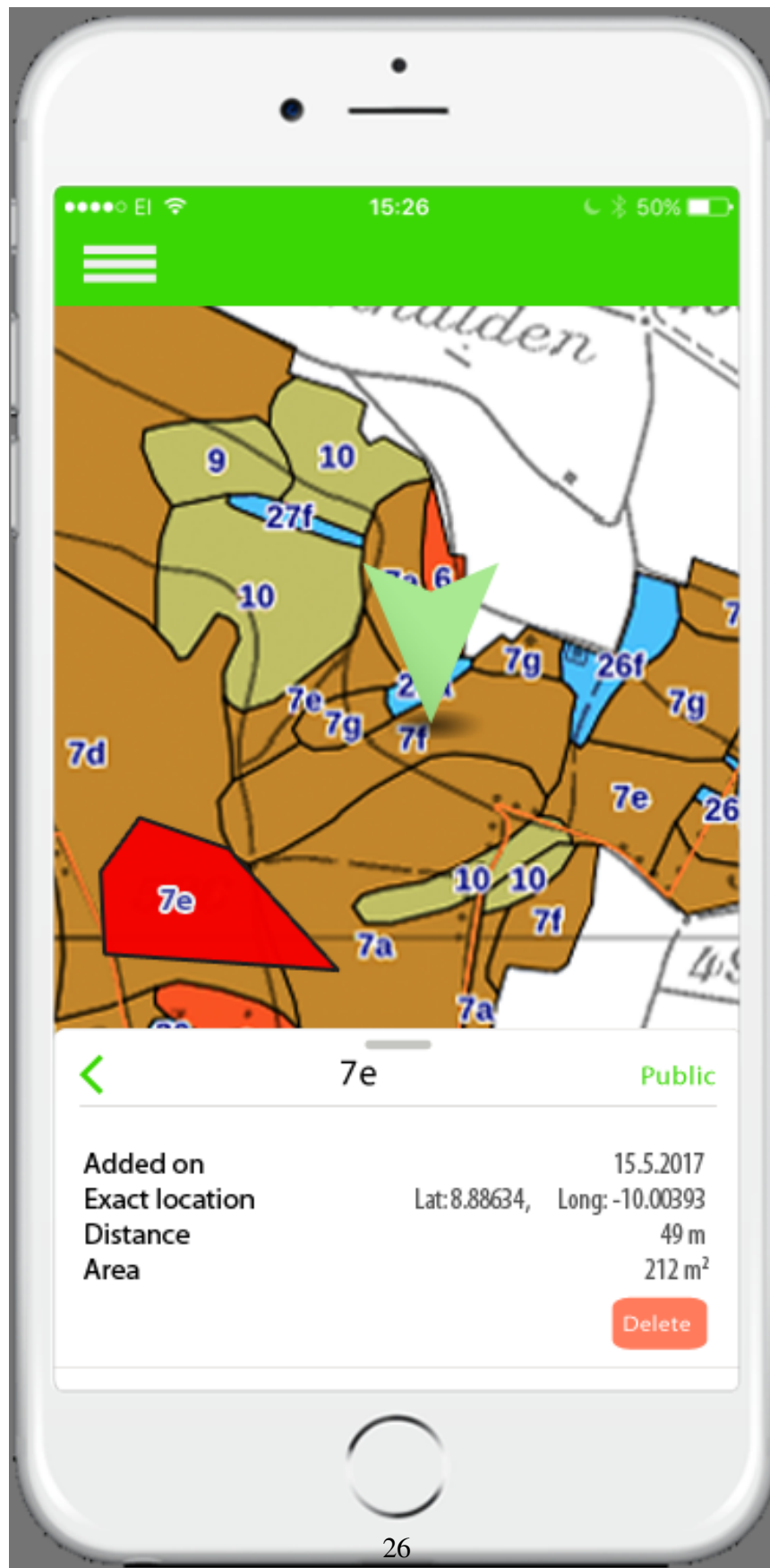


Abbildung 4.11: Mockup Screen 9

4.6 Implementation und Test

4.6.1 Implementation

Technische Implementation

API und Datenquellen

Login

Kartenmaterial

etc.

4.7 Automatische Testverfahren

Testgetriebene Entwicklung (TDD) ist eine Methode zur Softwareentwicklung, welche oft bei der agilen Entwicklung von Computerprogrammen eingesetzt wird. Bei TDD erstellt der Programmierer Software-Tests, um das korrekte Verhalten von Softwarekomponenten zu planen und zu überprüfen. Unit Tests (auch als Unittest oder als Komponententest bezeichnet), werden verwendet um die funktionalen Einzelteile eines Softwareprojekts zu prüfen.

4.7.1 Jasmine

4.7.2 Karma

Tests für Vue.JS wurden mit dem Test Runner Karma und dem Testframework Jasmine erstellt . Sie können über den Befehl "npm run test" ausgeführt werden, aus dem "client" folder des Vue-Servers.

Getestet werden Komponenten und deren Inhalt, welche von VueJS verwendet werden.

Damit Jasmin die Tests ausführen kann, muss der Code von Webpack zuerst kompiliert werden. Danach wird er in einer Browserumgebung (z.B Chrome oder Firefox) ausgeführt und das Testframework sammelt die zu testenden Werte und Verhalten. Es vergleicht diese mit gegebenen Werten und Zuständen, welche vom Programmierer erwartet werden. Stimmen sie überein, ist der Test bestanden. Stimmen sie nicht überein, ist der Test gescheitert und Jasmine meldet einen Fehler.

4.7.3 Travis

Continuous Integration

Um Continuous mit automated build und testing zu realisieren, habe ich Travis verwendet. Die Software Travis CI wurde 2011 in Berlin erstellt und 2013 veröffentlicht. Auf der Website travis-ci.org wird das Repo, welches auf Github gehostet wird, verknüpft. Jedes mal wenn auf das remote Repo gepusht wird, führt Travis die config in Form einer .yaml Datei aus, welche im root folder des Repos hinterlegt ist. Sie installiert alle Programme und Dependancies auf einer Linux Maschine und testet ob der Build in der aktuellen Version funktioniert. Es entsteht eine Version History mit dazugehöriger config. Travis

meldet, ob der Build und die Tests erfolgreich durchgeführt werden konnten. Dies geschieht auch auf separaten Branches des Projekts.

























































 ci_pgport	ci	 Daniel Schmider	 #44 failed	 999ddd4	 4 min 15 sec	
					 about an hour ago	
 ci	Continuous Integration with python and vue.js	 Daniel Schmider	 #42 passed	 18bfa3e	 2 min 35 sec	
					 about an hour ago	
 ci	ci	 Daniel Schmider	 #40 passed	 a284575	 3 min 35 sec	
					 about 2 hours ago	
 ci	ci	 Daniel Schmider	 #39 errored	 8f331cc	 12 min 31 sec	
					 about 3 hours ago	
 ci	ci	 Daniel Schmider	 #38 errored	 54ed094	 12 min 11 sec	
					 about 3 hours ago	
 ci	ci	 Daniel Schmider	 #37 errored	 418df28	 12 min 1 sec	
					 about 3 hours ago	
 ci	ci	 Daniel Schmider	 #36 errored	 45a47af	 12 min 24 sec	
					 about 3 hours ago	
 ci	ci	 Daniel Schmider	 #35 passed	 f3d06b6	 1 min 28 sec	
					 about 15 hours ago	

Abbildung 4.12: TravisCI Version History












```

{
  "language": "python",
  "python": 3.4,
  "virtualenv": {
    "system_site_packages": true
  },
  "addons": {
    "postgresql": "9.5",
    "apt": {
      "packages": [
        "postgresql-9.5-postgis-2.3"
      ]
    },
    "firefox": "latest"
  },
  "before_install": [
    "sudo apt-get -qq update",
    "sudo apt-get install binutils libproj-dev gdal-bin python-gdal"
  ],
  "install": [
    "pip install -r requirements.txt",
    "(cd client && npm install)"
  ],
  "script": [
    "python manage.py test WaldmeisterMap",
    "(cd client && npm test)"
  ],
  "global_env": "MOZ_HEADLESS=1 PGPORT=5435",
  "os": "linux",
  "group": "stable",
  "dist": "trusty"
}

```

Abbildung 4.13: Travis config .yaml, Version 45

Default Branch

 master  33 builds	# 33 failed  about 15 hours ago	 7028d0c   Daniel Schmider					
--	---	--	---	---	---	---	---

Active Branches























 ci_pgport  2 builds	# 45 failed  about an hour ago	 1afa786   Daniel Schmider					
 ci  8 builds	# 42 passed  about 2 hours ago	 18bfa3e   Daniel Schmider					

Abbildung 4.14: Travis branches

4.8 Manuelle Tests

4.8.1 Testfälle

4.8.2 Ziel

4.8.3 Reproduktion

4.9 Resultate und Weiterentwicklung

4.9.1 Resultate

4.9.2 Möglichkeiten der Weiterentwicklung

4.9.3 Vorgehen

4.10 Projektmanagement

4.10.1 Allgemeines

GitHub

4.10.2 Meilensteinplanung

4.10.3 Releases

4.10.4 Issues

4.10.5 Prototypen

4.10.6 Prozessmodell

Scrum

4.10.7 Aufwandschätzung

Zeitplan

Projektplan

4.10.8 Risiken

Deadlines

Technologien

Frameworks

Mobile Limits

4.11 Projektmonitoring

32

4.11.1 Soll-Ist-Zeitvergleich

4.11.2 Code Statistics

4.11.3 Software-Checklisten

Abbildungsverzeichnis

Glossar

Akronyme

SRID = Spatial Reference Identifier

GIS = Geografisches Informationssystem

MVC = Model - View - Controller

MVVM = Model - View - ViewModel

HTML = Hypertext Markup Language

CSS = Cascading Style Sheets

ES5 = ECMA Script 5

ECMA = European Computer Manufacturers Association

SoC = Separation of Concern

SPA = Single-Page-Applikation

EK72 = Ellenberg Klötzli

POI = Points of Interests

UML = Unified Modeling Language

JWT = (JSON Web Token)

JSON = JavaScript Object Notation

REST = Representational state transfer

CRUD = Create, Read, Update, Delete

DRF = Django Rest-Framework

CI = Continuous Integration

Literaturverzeichnis

- [AH07] Jacob Kaplan-Moss Adrian Holovaty. *The Definitive Guide to Django*. Apress, 2007.
- [djo] Djoser, rest implementation of django authentication system.
<https://github.com/sunscrapers/djoser>.
- [Geo] Geojson geometry lookup
<https://github.com/simonepri/geojson-geometries-lookup>.
- [git] Github projektboards automatisierung
<https://help.github.com/articles/about-automation-for-project-boards/>.
- [Har01] Jens Hartwig. *PostgreSQL - professionell und praxisnah*. Addison-Wesley, 2001.
- [JF08] Wesley Chun Jeff Forcier, Paul Bissex. *Python Web Development with Django*. Addison-Wesley Professional, 2008.
- [Plu] Plus codes webseite fuer developer
<https://plus.codes/developers>.
- [Ser] Introduction to service worker
<https://developers.google.com/web/fundamentals/primers/service-workers/>.
- [Vue] Vue single file components
<https://vuejs.org/v2/guide/single-file-components.html>.
- [ZHG] Geometadaten gis-zh - geolion
http://www.parcs.ch/wpz/pdf_public/2013/9601_20131015_131512_gds_110.pdf.