

Projekt Touch Grass

Fach: Software&PlattformArchitektur

Klasse: B-TIA_TIP-23-a-

Autoren: Renisch Loganathan

Jan Zbinden

Remo Rothacher

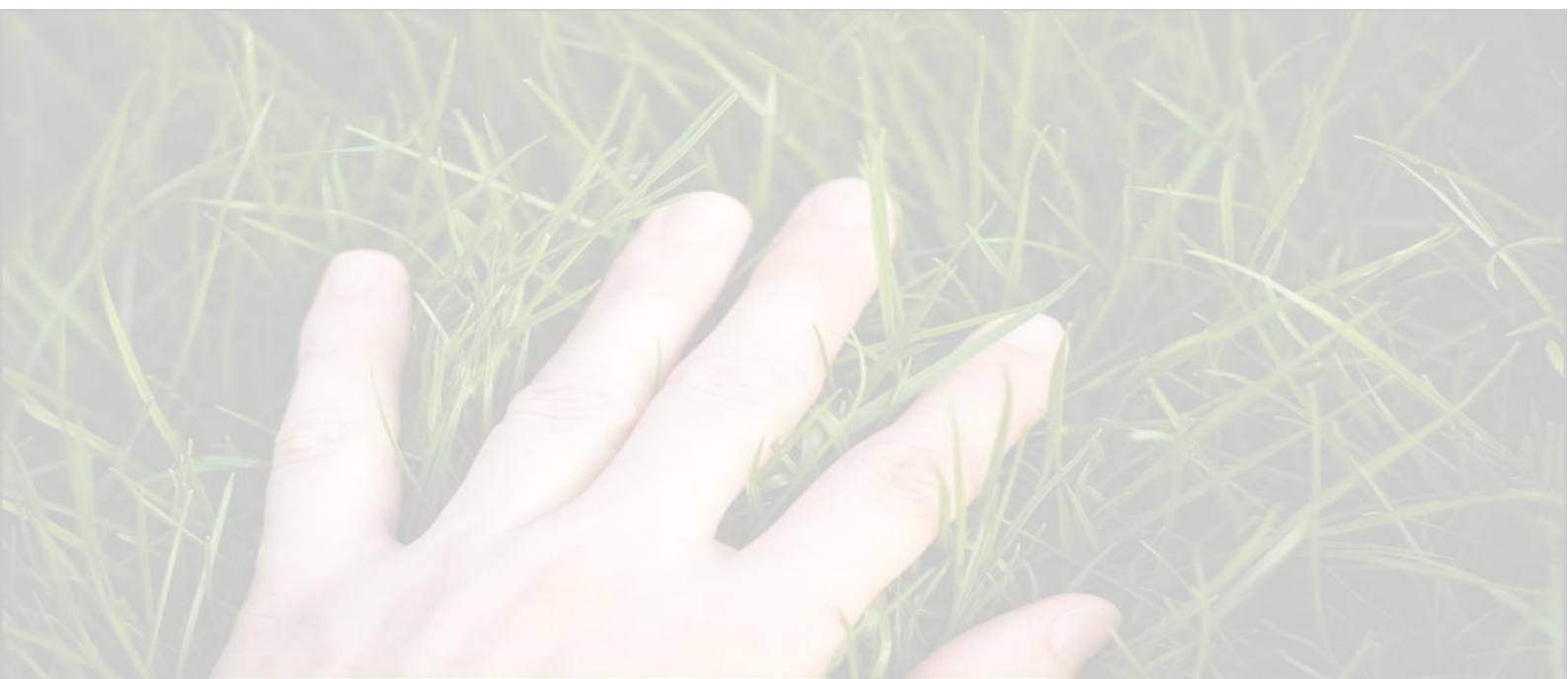
Flavio Guggisberg

Lisa Oberli

Adrian Aeschlimann

Dozent: Fabian Hirter

Abgabedatum: 09.03.2024



Inhalt

Abstract	3
1.1 Einleitung.....	4
1.2 Zeitplan.....	5
2 Zielsetzung.....	6
2.1 Integration der User-Stories in die Entwicklung.....	6
2.1.1 Kartenfunktionen	6
2.1.2 Profilverwaltung.....	7
2.1.3 Abenteuer und Quests.....	7
3 Hauptteil.....	8
3.1 Variantenentscheide	8
3.1.1 Codeverwaltung.....	8
3.1.2 Markdown Linter in Pipeline.....	9
3.1.3 Programmiersprache Frontend	10
3.1.4 Acceptance Test Tool.....	11
3.1.5 Map Tool	12
4 Ergebnisse.....	13
4.1 Hauptseite	14
4.2 Profil	15
4.3 Profil bearbeiten.....	16
4.4 Achievements	17
4.5 Quests	18
5 Diskussion	19
5.1 Arbeiten in der Gruppe	19
5.2 Konflikte und deren Lösungen.....	19
5.3 Entscheidungen verzögern.....	20
6 Ausblick	21
6.1 Frontend	21
6.2 Backend.....	21
7 Eigenständigkeitserklärung.....	22
8 Abbildungsverzeichnis.....	22
9 Quellenangabe.....	22

Abstract

Im Rahmen des Fachs *Software- und Plattformarchitektur* entstand das Projekt **Touch Grass**, eine Webanwendung, die durch Gamification Jugendliche dazu motivieren soll, sich aktiv in der Natur zu bewegen. Ziel der Plattform ist es, spielerische Anreize für Outdoor-Aktivitäten zu schaffen und so die Freude an Bewegung zu fördern. Nutzer können durch eine interaktive Karte mit freischaltbaren Bereichen und eingebundenen Wanderwegen Abenteuer erleben und durch Bewegung Erfahrungspunkte (XP) sammeln, die es ermöglichen, neue Kartenbereiche zu erkunden und den Avatar individuell auszustatten.

Das Projekt wurde von einem sechsköpfigen Team mit unterschiedlichen beruflichen Hintergründen durchgeführt. Im Entwicklungsprozess wurden Software-Tools wie GitHub zur Codeverwaltung, JavaScript für das Frontend sowie Gauge für Akzeptanztests eingesetzt. Herausforderungen in der Teamkoordination konnten durch die Einführung eines klaren Projektmanagements und verbesserte Kommunikationswege erfolgreich bewältigt werden.

Die Anwendung befindet sich aktuell in einem Prototypenstatus. Die nächsten Schritte umfassen die Implementierung eines Backends, die Synchronisierung von Profildaten sowie die Erweiterung des Quest-Systems. **Touch Grass** legt den Fokus auf gesundheitsfördernde Aktivitäten und hebt sich von bestehenden Location-Based-Games ab, indem es das Naturerlebnis und körperliche Bewegung in den Mittelpunkt stellt.

1.1 Einleitung

Im Rahmen des Fachs Software- und Plattformarchitektur erhielten wir den Auftrag, ein Projekt zu planen und durchzuführen. Zu Beginn entwickelten alle Studierenden gemeinsam mit ihrem Sitznachbarn eine Projektidee ohne strikte Vorgaben, ausser dass der Umfang bewusst gross gewählt sein sollte und eine vollständige Umsetzung in der vorgegebenen Zeit nicht erforderlich war. Anschliessend wurden diese Ideen in Form eines Elevator-Pitches der Klasse präsentiert. Nach den Vorstellungen konnte jeder Studierende drei favorisierte Ideen auswählen. Basierend auf diesen Präferenzen wurden die Teams gebildet. So entstand unsere Gruppe «Touch Grass».

Wir sind sechs Studierende mit unterschiedlichen Hintergründen. Drei von uns haben eine Lehre als Informatiker Applikationsentwicklung EFZ absolviert, einer als Informatiker Systemtechnik EFZ, einer als Kaufmann EFZ und einer als Automatiker EFZ. Diese Vielfalt an Erfahrungen bringt spannende Perspektiven in unser Team.

Unsere zentrale Idee bestand darin, eine gesundheitsfördernde App zu entwickeln, die Jugendliche durch „Gamification“ dazu motiviert, draussen sportlich aktiv zu werden. Dies soll durch wöchentliche Aktivitäten und mithilfe eines Quest-Systems hauptsächlich eine jüngere Zielgruppe ansprechen.

Dies war kein komplett neues Konzept. Schon 2016, als Pokémon GO auf den Markt erschien, war das Interesse an „Location-Based“-Spielen enorm. Durch diesen Hype folgte eine ganze Reihe dieses Genres. Da aber in fast allen diesen Spielen das Gameplay im Vordergrund steht, wollten wir einen Schritt zurückgehen und das Wandern selbst in den Fokus rücken. Die Quests sollen erfüllt werden, indem man beispielsweise auf einem Wanderweg am Ende ein bestimmtes Wahrzeichen beschreibt oder die Inschrift auf einem alten Brunnen entschlüsselt. Damit möchten wir Jugendliche vom Digitalen zurück in die Natur führen.

Daraus entstand unsere Idee „Touch Grass“, eine Webanwendung, die Nutzer dazu motiviert, sich aktiv in der Natur zu bewegen. Die Plattform bietet eine interaktive Karte mit freischaltbaren Bereichen und eingebundenen Wanderwegen. Durch Bewegung sammeln Nutzer Erfahrungspunkte (XP), mit denen sie neue Kartenbereiche und Belohnungen freischalten oder ihren Avatar individuell ausstatten können. Ein integriertes Freunde-System ermöglicht es, sich mit anderen zu vernetzen und Herausforderungen gemeinsam zu bestreiten. Ziel der Anwendung ist es, durch Gamification-Anreize einen spielerischen Zugang zu Outdoor-Aktivitäten zu schaffen und so die Freude an Bewegung in der Natur zu fördern.

1.2 Zeitplan

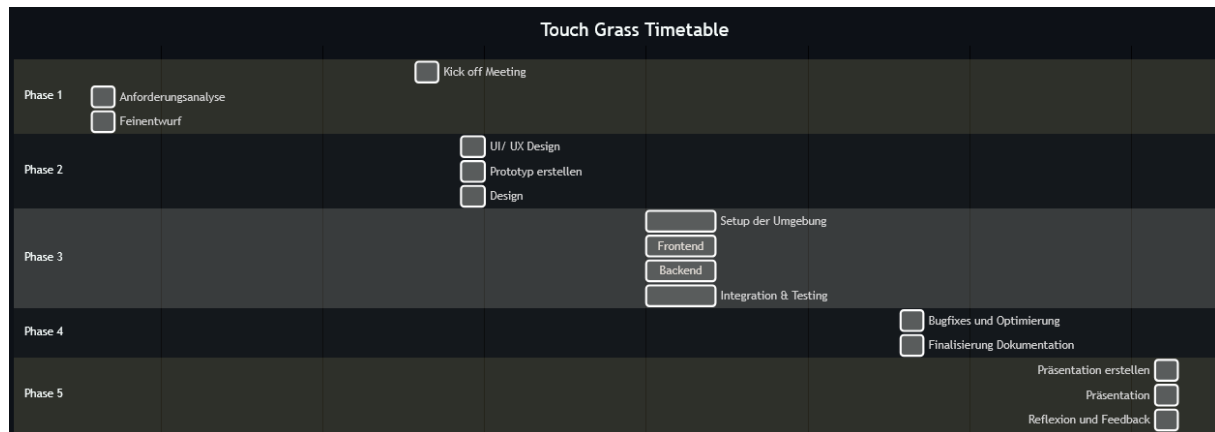


Abbildung 2: Zeitplan

Zum Start des Projektes, haben wir einen Zeitplan erarbeitet. Da wir nicht genau wussten wie viel in der verfügbaren Zeit machbar ist, haben wir alles eingeplant was nötig ist, um einen lauffähigen Prototyp zu haben. Wir haben unser Projekt in 5 Phasen unterteilt. In Phase 1 haben wir mit einem Kick-off-Meeting gestartet, gefolgt von der Anforderungsanalyse und dem Feinentwurf. Phase 2 konzentrierte sich auf das UI/UX-Design, die Erstellung eines ersten Prototyps sowie das Design der Anwendung. In Phase 3 konnten wir nicht ganz alles nach Plan umsetzen. Wir konnten die Entwicklungsumgebung einrichten, das Frontend entwickeln und führten die Integration und das Testing durch. Das Backend haben wir bewusst aussen vorgelassen, weil wir sonst zu viel Zeit damit verbracht hätten, dieses aufzusetzen und wir am Schluss wahrscheinlich nichts wirklich zeigen könnten. In Phase 4 lag der Fokus auf Bugfixes, Optimierungen und der Finalisierung der Dokumentation. Schliesslich befassten wir uns in Phase 5 mit der Erstellung und Durchführung der Präsentation sowie einer abschliessenden Reflexion und Feedback-Runde.

2 Zielsetzung

2.1 Integration der User-Stories in die Entwicklung

Um sicherzustellen, dass die Funktionen von Touch Grass den Erwartungen der Nutzer entsprechen, wurden User-Stories definiert. Diese beschreiben die Anforderungen aus Sicht der Benutzer und gibt uns eine Richtlinie für die Funktionen.

2.1.1 Kartenfunktionen

Die Navigation innerhalb der App erfolgt über eine interaktive Karte. Nutzer sollen sich darin orientieren, ihren Fortschritt verfolgen und Quest-Orte identifizieren können.

Relevante User-Story:

- Als Nutzer möchte ich meinen Fortschritt auf der Karte sehen können, damit ich nachvollziehen kann, welche Gebiete ich bereits freigeschaltet habe.
- Als Nutzer möchte ich in die Karte hinein- und herauszoomen können, um mich besser orientieren zu können.
- Als Nutzer möchte ich die Karte bewegen können, um verschiedene Bereiche zu erkunden.
- Als Nutzer möchte ich Quest-Orte auf der Karte sehen können, damit ich weiss, wo ich meine Aufgaben erledigen kann.
- Als Nutzer möchte ich meinen aktuellen Standort auf der Karte sehen, um zu wissen, wo ich mich gerade befinde.

2.1.2 Profilverwaltung

Jeder Nutzer soll die Möglichkeiten haben, ein individuelles Profil zu erstellen, den eigenen Fortschritt zu speichern und sich mit Freunden zu vernetzen.

Relevante User-Story:

- Als Nutzer möchte ich mein Profil sehen können, um meine persönlichen Daten zu überprüfen.
- Als Nutzer möchte ich mein Profil bearbeiten können, z. B. mein Profilbild wechseln, um mein Profil individuell zu gestalten.
- Als Nutzer möchte ich meinen Fortschritt im Profil sehen können, damit ich meine Erfolge und Aktivitäten nachvollziehen kann.
- Als Nutzer möchte ich Freunde hinzufügen oder entfernen können, um mit anderen Spielern in Kontakt zu bleiben.
- Als Nutzer möchte ich die Profile meiner Freunde anschauen können, um ihre Erfolge und Aktivitäten zu sehen.

2.1.3 Abenteuer und Quests

Das Quest-System ist ein zentraler Bestandteil der App und sorgt für Motivation, neue Orte zu erkunden und Aufgabe zu lösen.

Relevante User Story:

- Als Nutzer möchte ich eine Übersicht meiner Abenteuer sehen können, um zu wissen, welche Aufgaben ich erledigen muss.
- Als Nutzer möchte ich meinen Fortschritt in jedem Abenteuer sehen können, um zu wissen, wie viel ich bereits geschafft habe.
- Als Nutzer möchte ich eine Liste der offenen und abgeschlossenen Abenteuer sowie eine Übersicht meiner bereits erreichten Erfolge sehen können.

3 Hauptteil

3.1 Variantenentscheide

Im Unterricht wurde uns gelernt das es wichtig ist, bei einem Projekt eine nachvollziehbare Dokumentation zu verwalten. Angefangen hat es bei dem Entscheid was für einen Versionenverwaltung-Dienst wir verwenden. Nach unserer Recherche fanden wir folgende Dienste: GitHub, GitLab, Azure DevOps und Launchpad. Wir haben uns entschieden, GitHub als zentrale Plattform zur Codeverwaltung zu verwenden. Diese Entscheidung haben wir getroffen, weil die Mehrheit vom Team schon einmal mit GitHub gearbeitet hat. Dadurch konnten die Mitglieder, die damit vertraut sind, die Unerfahrenen am besten unterstützen. Ausserdem bietet es eine umfassende Unterstützung für die Versionskontrolle mit Git, ermöglicht paralleles Arbeiten, die Integration von Pipelines ist einfach und es ist gut dokumentiert und bietet eine übersichtliche Benutzeroberfläche. Nachdem wir unser GitHub-Repo aufgesetzt hatten, ging es darum, wie wir unsere Architektur-Entscheide protokollierten. Hier wählten wir das ADR-Tool von Nat Pryce aus da es die besten Bewertungen hatte und die Installation relativ einfach aussah.

3.1.1 Codeverwaltung

Um effizient als Team zusammenarbeiten zu können, benötigten wir eine zentrale Plattform als Codeverwaltung. Diese Plattform muss mehrere Anforderungen erfüllen:

- Versionskontrolle und Änderungen des Codes nachvollziehen
- Parallele Bearbeitung
- Integration von Tools für Code-Review

Entscheidung:

Für die zentrale Codeverwaltungsplattform haben wir uns für GitHub entschieden. Diese Entscheidung haben wir getroffen, weil die Mehrheit in unserem Team bereits Erfahrungen mit GitHub machen durfte. Es bietet eine umfassende Unterstützung für die Versionskontrolle mit Git, ermöglicht paralleles Arbeiten und die Integration von Pipelines.

3.1.2 Markdown Linter in Pipeline

In unserem Projekt verwenden wir Markdown-Dateien wie zum Beispiel README.md, um wichtige Informationen klar und verständlich darzustellen. Damit diese Dateien einheitlich formatiert sind und fehlerfrei sind, benötigen wir ein Tool, das Markdown-Dateien automatisch auf Syntaxfehler und Stilabweichungen überprüft.

Entscheidung:

Unser Team hat entschieden, die GitHub Action `DavidAnson/markdownlint-cli2-action@v17` zu verwenden, um die Markdown-Validierung in unserem Workflow zu implementieren. Diese Entscheidung haben wir aus folgenden Gründen getroffen:

- Die Action hat die meisten Bewertungen im GitHub Marketplace
- Eine benutzerdefinierte Konfigurationsdatei (`markdownlint.json`) kann erstellt und nach Wunsch konfiguriert werden
- Der Output ist gut und klar verständlich und zeigt sowohl Fehler als auch Empfehlungen an

Konsequenzen:

Durch die Integration von `DavidAnson/markdownlint-cli2-action@v17` wird sichergestellt, dass alle Markdown-Dateien im Projekt automatisch auf Syntaxfehler und Stilabweichungen geprüft werden. Dies führt zu einer einheitlichen und konsistenten Dokumentation, was die Lesbarkeit und Wartbarkeit der Dateien verbessert. Teammitglieder müssen sich mit der Konfigurationsdatei `.markdownlint.json` und den Fehlermeldungen vertraut machen, um Probleme effizient zu beheben. Der Workflow wird regelmässig ausgeführt.

3.1.3 Programmiersprache Frontend

Für unser Projekt benötigten wir eine Programmiersprache für das Frontend, die eine dynamische und interaktive Benutzererfahrung ermöglicht. Die Sprache sollte weit verbreitet gut unterstützt und flexibel genug sein, um eine Vielzahl von Anforderungen abzudecken. Gleichzeitig legen wir Wert auf Cross-Plattform-Fähigkeiten und eine starke Entwickler-Community, um Entwicklerressourcen und langfristige Wartbarkeit sicherzustellen.

Entscheidung:

Wir haben uns aus folgenden Gründen für JavaScript entschieden:

- Breite Unterstützung durch Framework
- Cross-Plattform-Fähigkeit
- Etablierte Community und Ressourcen
- Etablierte Integration in bestehende Systeme und Webbrowser
- Möglichkeit zur Erstellung dynamischer und interaktiver Benutzeroberflächen
- Flexibilität: JavaScript eignet sich sowohl für einfache Skripte als auch für komplexe Anwendungen
- Einfach zu lernen für komplette Anfänger im Team

Diese Punkte wie auch der [Blogartikel von Snipcart](#), welche die Vorteile und Bedeutung von JavaScript für die moderne Webentwicklung noch mehr im Detail hervorhebt, haben uns überzeugt.

Konsequenzen:

Durch die Verwendung von JavaScript wird sichergestellt, dass wir eine moderne, flexible und weit verbreitete Technologie nutzen, die eine dynamische Benutzererfahrung ermöglicht. Die große Entwickler-Community und die verfügbaren Tools erleichtern die Entwicklung und Wartung. Teammitglieder müssen sich jedoch mit den Best Practices und Sicherheitsaspekten von JavaScript vertraut machen, um die Vorteile der Technologie voll auszuschöpfen und potenzielle Risiken zu minimieren.

3.1.4 Acceptance Test Tool

In unserem Projekt ist es entscheidend, die Qualität der Software kontinuierlich sicherzustellen, indem die Funktionalitäten regelmäßig durch Akzeptanztests überprüft werden. Diese Tests müssen klar definiert, leicht verständlich und einfach wartbar sein, um die Zusammenarbeit zwischen technischen und nicht-technischen Teammitgliedern zu fördern. Zudem benötigen wir ein Tool, das sich gut in bestehende CI/CD-Pipelines integrieren lässt und eine klare Trennung von Testlogik und Testdaten ermöglicht.

Entscheidung:

Wir haben uns entschieden, das Tool **Gauge** für die Implementierung von Akzeptanztests zu verwenden. Diese Entscheidung basiert auf folgenden Faktoren:

- **Lesbarkeit:** Gauge verwendet eine leicht verständliche Syntax für Testszenarien, die auch für Nicht-Entwickler zugänglich ist.
- **Flexibilität:** Unterstützt mehrere Programmiersprachen und ermöglicht einfache Anpassungen.
- **Modularität:** Trennung von Spezifikationen (Testfälle) und Logik (Implementierung).
- **Integration:** Leichte Integration in CI/CD-Pipelines und mit anderen Tools wie Selenium oder REST-Assured.
- **Community-Support:** Etablierte Community und aktive Weiterentwicklung.
- **Erweiterbarkeit:** Unterstützt benutzerdefinierte Plugins für zusätzliche Funktionalität.

Konsequenzen:

Durch die Verwendung von Gauge wird sichergestellt, dass Akzeptanztests klar strukturiert und leicht wartbar sind. Die Lesbarkeit der Tests fördert die Zusammenarbeit zwischen Entwicklern, Testern und Stakeholdern. Zudem vereinfacht die Modularität die Wiederverwendbarkeit von Testbausteinen. Teammitglieder müssen sich jedoch mit der Syntax und den Best Practices von Gauge vertraut machen, um das Tool effektiv zu nutzen. Die Integration in bestehende Workflows wird zusätzliche Initialarbeit erfordern, bietet jedoch langfristig große Vorteile für die Testautomatisierung.

3.1.5 Map Tool

Wir benötigen eine Lösung, um nicht nur eine Map anzuzeigen, sondern auch einen Pfad zwischen verschiedenen Punkten auf dieser Map. Das wird benötigt, um die Wanderwege darzustellen.

Entscheidung:

Wir haben uns entschieden, das Tool Grasshopper für die Erstellung von Karten zu verwenden. Danach werden die Karten nach umap exportiert, um die Maps embedable zu machen. Diese Entscheidung basiert auf folgenden Faktoren:

- **Editor:** Grasshopper bietet einen online Editor, mit welchem einfache Karten mit einem Pfad erstellt werden können.
- **Embedable:** Die erstellten Karten können durch umap gehostet und embeded werden.
- **Kosten:** Beide Tools basieren auf OpenStreetMap und somit kostenlos.
- **Komplexität:** Die Map benötigt nur ein einzelnes Iframe und sind somit einfach zu implementieren.
- **Anpassbarkeit:** Die Map welche durch umap kann nur schlecht, während der runtime editiert werden. Sollte das benötigt werden, muss nach einer anderen Lösung gesucht werden.

Konsequenzen:

Diese Tools nehmen uns den Aufwand ab, selbst eine Lösung zu implementieren, um die Wanderwege darstellen zu können. Falls die Map während der runtime editiert werden muss, können wir mit hoher Wahrscheinlichkeit die generierten Maps von Grasshopper weiterverwenden.

4 Ergebnisse

In diesem Teil wird der Stand der Arbeit zum Zeitpunkt der Abgabe der Arbeit dargestellt.

Der aktuelle Stand des Projektes kann unter <https://touch-grass-six.vercel.app/> angesehen werden.

Der Code ist unter <https://github.com/FaKiieZ/touch-grass> zu finden.

4.1 Hauptseite

Die Hauptseite zeigt die Karte an. Ausserdem sind direkt einige unserer Features ersichtlich.

- 1) Eine Route kann ausgewählt werden. Momentan sind nur 2 Routen auswählbar welche als Platzhalter erstellt wurden. Für dieses Feature wurden Tools ausgewählt und entwickelt, um Routen als .GPX-Dateien zu erstellen und anschliessend den Benutzern zur Auswahl zu geben.
- 2) Der aktuelle Standort ist durch einen Pin ersichtlich. Die Sicht ist auf die unmittelbare Umgebung eingeschränkt. Im Aufgedeckten Bereich ist die Route ersichtlich, wobei sich die Route nicht an den aktuellen Standpunkt anpasst.
- 3) Der Endpunkt der Route wird markiert und ist auch im Fog of War ersichtlich.
- 4) Fürs Anzeigen der Karte und der Kartenelementen wird Leaflet verwendet, die Kartendaten werden dadurch von OpenStreetMap bereitgestellt.

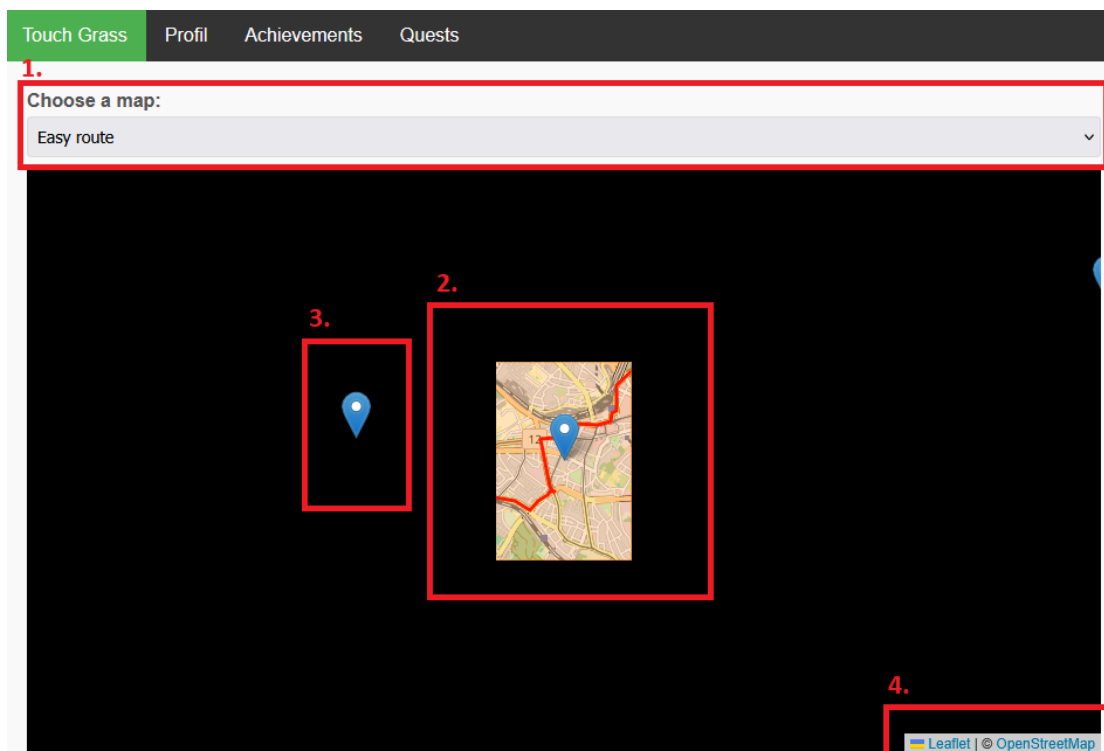


Abbildung 3: Touch Grass – Startseite

4.2 Profil

Auf dem Profil kann sich der Benutzer seine Daten ansehen. Da momentan weder Backend noch Login vorhanden ist, werden hier lokale Daten angezeigt, welche im Local Storage des Browsers gespeichert werden.

- 1) Der Benutzer kann auf ersten Blick seine Daten ansehen, das beinhaltet Namen, Alter, Ort und Berufung. Der Benutzer kann sein Profil bearbeiten.
- 2) Der aktuelle Levelfortschritt wird angezeigt
- 3) Die Freundesliste wurde mit einigen Platzhaltern befüllt. Die Profile der Freunde sind noch nicht ersichtlich.

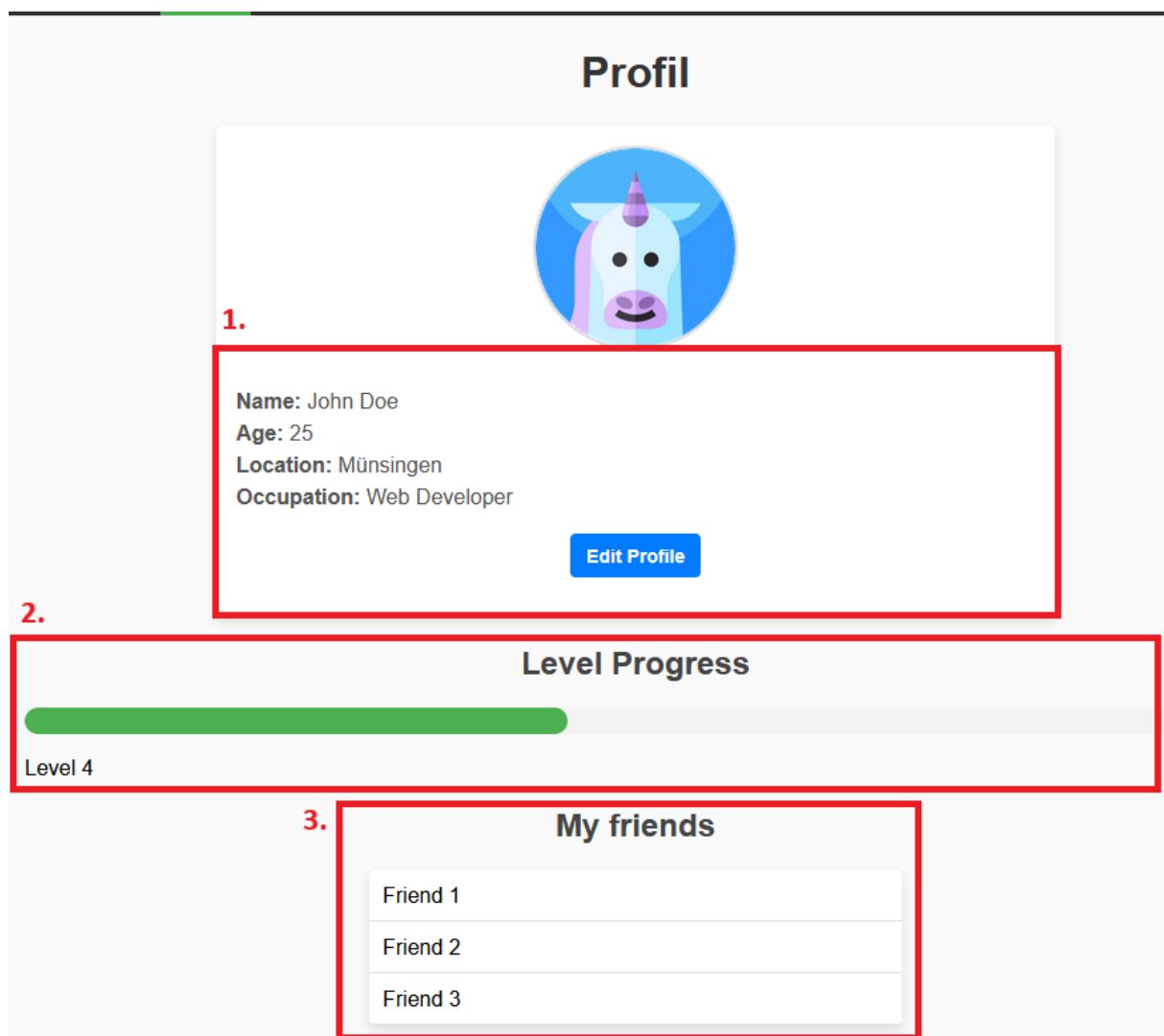



Abbildung 4: Touch Grass - Profil





4.3 Profil bearbeiten

Auf der Profil Bearbeitungsansicht kann der Benutzer seine Daten anpassen. Als Profilbild kann eines der vordefinierten Bilder ausgewählt werden.

Profil Details



Choose Profile Icon:



Name:

Age:

Location:

-- Select a canton --

Occupation:

Save

Abbildung 5: Touch Grass - Profil Detail

4.4 Achievements

Auf der Achievements Ansicht sind einige Langzeitziele ersichtlich, welche abgehakt werden können. Diese können momentan vom Benutzer direkt abgeschlossen werden und sind nicht an Events gebunden. Die abgeschlossenen Achievements werden noch nicht gespeichert.

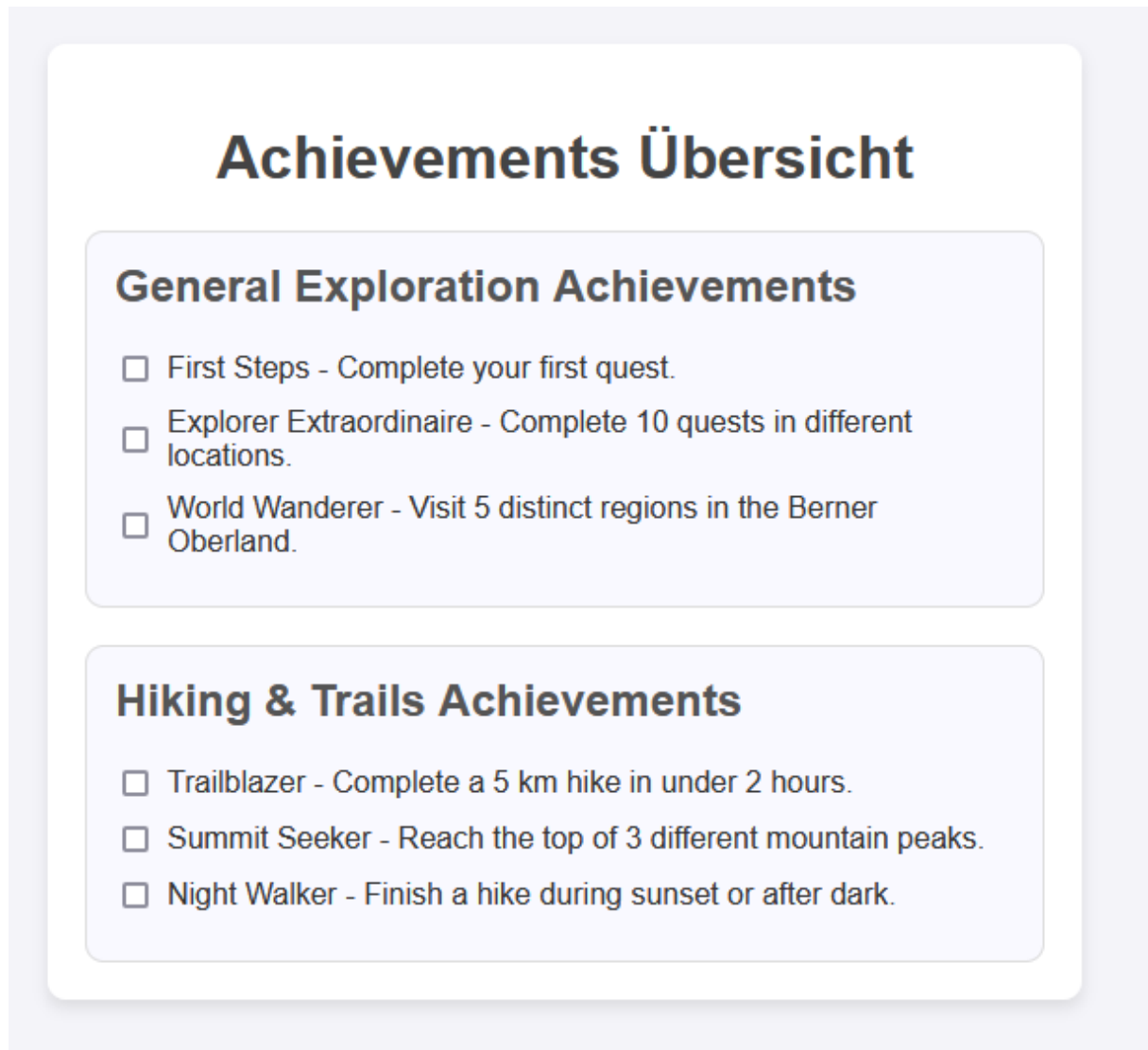


Abbildung 6: Touch Grass - Achievements

4.5 Quests

Auf der Quests Ansicht sind Ziele ersichtlich, welche sich täglich oder wöchentlich erneuern.

- 1) Die Quests können abgeschlossen werden und sind momentan nicht an Events gebunden. Der Status der Quest wird lokal gespeichert.
- 2) Der Levelfortschritt wird angezeigt und lokal gespeichert. Die abgeschlossenen Quests erstellen eine Belohnung von XP, welche abgeholt werden kann.

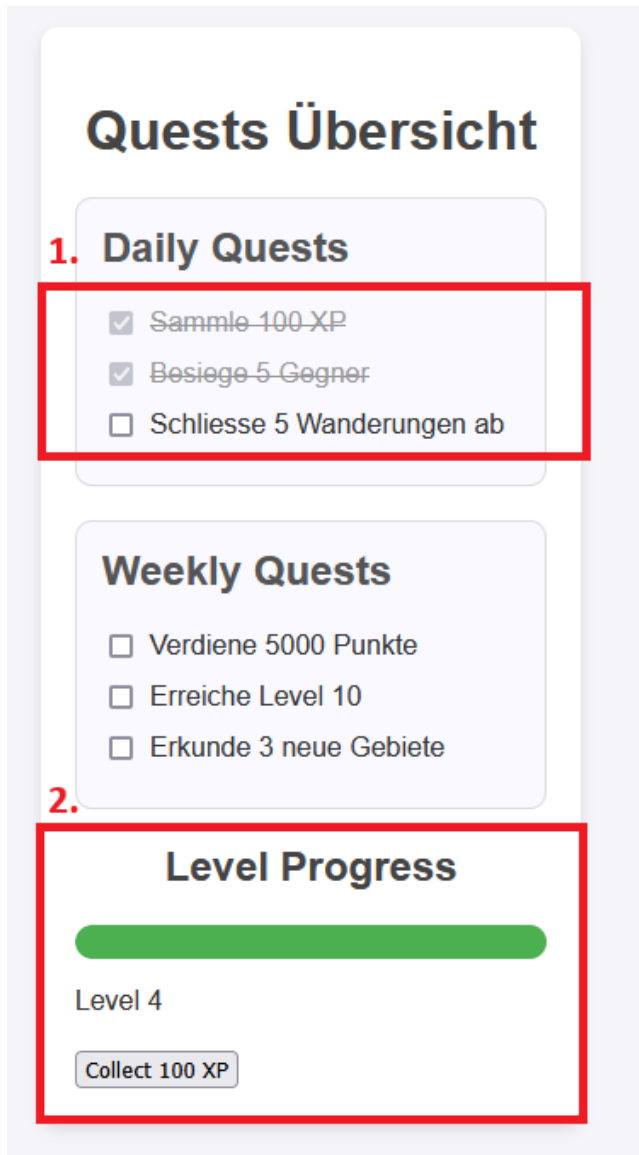


Abbildung 7: Touch Grass - Quests

5 Diskussion

5.1 Arbeiten in der Gruppe

Unsere Gruppe bestand aus sechs Mitgliedern mit unterschiedlichem Vorwissen. Diese Vielfalt brachte sowohl Vor- als auch Nachteile mit sich. Mitglieder ohne Vorkenntnisse im Programmieren hatten anfangs Schwierigkeiten, konnten jedoch durch die Unterstützung der erfahreneren Teammitglieder, selbst einfachere Aufgaben erfolgreich abschließen.

Die Zusammenarbeit erwies sich als wertvoll, da sie es ermöglichte, Wissen auszutauschen und voneinander zu lernen. Besonders die Kombination aus technischer Kompetenz und neuen Perspektiven führte zu kreativen Lösungsansätzen. Herausforderungen, die für einzelne Mitglieder unlösbar erschienen, konnten durch gemeinsame Diskussionen und Teamarbeit überwunden werden.

Insgesamt trug die Gruppenarbeit zum Erfolg des Projekts bei, da jeder sein individuelles Potenzial einbringen konnte und wir gemeinsam ein besseres Ergebnis erzielten, als es allein möglich gewesen wäre.

5.2 Konflikte und deren Lösungen

Während unserer Zusammenarbeit stiessen wir auf das Problem, dass es keinen klar definierten Projektleiter gab. Dadurch kam es zu doppelter Arbeit, da Aufgaben unkoordiniert erledigt wurden. Einige Teammitglieder arbeiteten an denselben Inhalten, während andere Unsicherheiten hatten welche Schritte als nächstes folgen sollten. Dies führte zu Konflikten wie zum Beispiel als zwei Mitglieder gleichzeitig ihre eigene Lösung, der Avatar Auswahl im Profil, in den Main Branch pushen wollten.

Um dieses Problem zu lösen, entschieden wir uns gemeinsam, eine Person als Projektleiter zu bestimmen. Diese Person übernahm die Verantwortung für die Koordination und stellte sicher, dass alle Aufgaben klar verteilt waren. Zusätzlich führten wir eine Planung ein, bei der wir zu Beginn jeder Lektion besprachen, welche Aufgaben an diesem Tag erledigt werden sollten. So arbeiteten wir in kleinem Gruppen mit gezielten Aufgaben um Doppelarbeit zu vermeiden.

Ein weiteres Problem war auch das unsere Gruppe ursprünglich in drei Reihen voneinander getrennt sassen. Dadurch war jeder nur auf seinen eigenen Laptop fokussiert und arbeitete für sich. Dies erschwerte die Kommunikation und führte zu Missverständnissen, welche den Teamfluss verhinderten. Ideen wurden nicht direkt geteilt, und es fühlte sich eher nach Einzelarbeit als nach Gruppenarbeit an.

Um dies zu verbessern, entschieden wir uns dafür, nur noch einen Laptop pro Aufgabe zu nutzen. Dadurch konnten wir uns gemeinsam auf eine Aufgabe konzentrieren und Diskussionen effektiver führen. Zudem setzten wir uns in eine gemeinsame Reihe, sodass

jeder sich aktiv einbringen konnte. Diese Veränderung förderte nicht nur den Austausch, sondern verbesserte auch unsere Zusammenarbeit und den Teamgeist erheblich.

5.3 Entscheidungen verzögern

Im Verlauf der Arbeit versuchten wir Entscheidungen so spät wie möglich zu treffen, um uns nicht an Entscheidungen binden zu müssen. Ein Resultat davon war, dass wir bis Ende der Arbeit kein Backend definierten und ausser der Kartenansicht kaum Abhängigkeiten aufbauten.

Für unser Projekt funktionierte dieser Ansatz sehr gut, da wir in der limitierten Zeit etwas erstellen konnten, was auch vorgezeigt werden kann. Während der Arbeit stiessen wir nie auf das Problem, dass eine Entscheidung nicht früher getroffen wurde.

6 Ausblick

Das Projekt wurde absichtlich zu gross angesetzt und hat somit viele Aspekte die unvollständig umgesetzt wurden.

6.1 Frontend

Momentan werden Daten im Frontend entweder verworfen oder im Local Storage des Browsers gespeichert. Diese Daten müssten in der Zukunft mit einem Backend synchronisiert werden.

Die nächsten Schritte abgesehen von Integration mit dem Backend wären:

- Karte/Route
 - Aktuelle Position und Routenziel unterschiedlich darstellen
 - Abschliessen der Route ermöglichen
 - Abgeschlossene Routen anzeigen
 - Aufdecken der Karte ermöglichen
 - UX der Karte überarbeiten. Z.B. Zoom, Navigation
- Profil
 - Profildaten Felder neu bewerten, welche benötigt sind und welche eventuell noch fehlen.
 - Freunde hinzufügen
 - Profil von Freunden ansehen

6.2 Backend

Das Backend wurde weder erstellt, noch wurden Technologien fürs Backend gewählt. Für die Weiterentwicklung des Projekts wird ein Backend benötigt, welches mindestens folgende Anforderungen erfüllt:

- Benutzersessions
- Registration und Login
- Datenbank für Profildaten inkl. Levels und Freundesliste
- Tracking von erreichten Achievements und Quests inkl. Generieren von neuen Quests
- Übertragen von .GPX-Dateien für Routen

7 Eigenständigkeitserklärung

“Wir versichere hiermit, dass wir die vorliegende Arbeit selbstständig, ohne fremde Hilfe, verfasst sowie keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Stellen, die wörtlich oder sinngemäss aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.“

8 Abbildungsverzeichnis

Abbildung 1: Titelbild	1
Abbildung 2: Zeitplan	5
Abbildung 3: Touch Grass – Startseite	14
Abbildung 4: Touch Grass - Profil	15
Abbildung 5: Touch Grass - Profil Detail	16
Abbildung 6: Touch Grass - Achievements	17
Abbildung 7: Touch Grass - Quests	18

9 Quellenangabe

Modern Softwareengineering, David Farley (2021)

<https://snipcart.com/blog/why-javascript-benefits>