



School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Projektarbeit (Informatik)

VanFan – Die App für Vanlifers

Autoren

Ramon Imper
Roger Wetter

Hauptbetreuung

Andreas Meier

Datum

23.12.2022

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Flawil, 23.12.2022

Unterschriften:

R. Limpert
R. Wett

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Abstract

Im Modul Software-Projekts 4 erarbeitete ein Team aus 7 Studierenden der ZHAW in Zusammenarbeit mit Mirja Qachar und Robin Bär das Grundgerüst der App VanFan. VanFan ist eine Kontaktplattform für Van-Enthusiasten. Dazu wurde zuerst eine Projektskizze erstellt, um ein grobes Konzept der gewünschten Applikation zu erhalten. Danach wurden in fünf Sprints à zwei Wochen diverse Features implementiert. Der Entwicklungsstand dieser Features wurde evaluiert. Ebenfalls wurden Erfolge und Rückschläge, die während der Entwicklung entstanden, angesprochen. Abschliessend wurde geprüft, ob sich eine Weiterentwicklung lohnen könnte und was für eine Veröffentlichung des Produkts noch getätigigt werden müsste. Im Rahmen der Projektarbeit entwickelten zwei dieser Studierenden die App weiter, auch dies weiterhin in Zusammenarbeit mit Mirja Qachar und Robin Bär.

Vorwort

Wir haben uns dazu entschieden, die VanVan App zu zweit weiterzuentwickeln, weil wir beide bereits grosse Freude am PM4 Projekt hatten. Da wir dort bereits die Grundlagen erarbeiten konnten, ging es jetzt darum, die noch bestehenden Fehler zu beheben und die fehlenden Features hinzuzufügen. Wir beide haben grosses Interesse am Erstellen von Handy-Apps und konnten uns mit diesem Projekt noch tiefer in die Materie stürzen. Aufgrund der Tatsache, dass wir nun nur noch zu zweit weiterentwickelten, haben wir nicht mehr ein Front- und ein Back-end Team erstellt, sondern haben beide an beiden Seiten gearbeitet. Dadurch konnten wir das Implementieren von Features oder das Beheben von Bugs von Anfang bis Schluss selber gestalten und so den Arbeitsprozess vereinfachen und mehr lernen. Das Weiterführen der VanFan App war für uns beide eine persönliche Bereicherung und deshalb danken wir Mirja Qachar, Robin Bär und Andreas Meier für die tolle Möglichkeit.

Inhaltsverzeichnis

1. Einleitung	5
1.1. Stand vor der PA	5
1.2. Aufgaben	5
1.3. Stand nach der PA	5
1.3.1. Feed	6
1.3.2. Map	8
1.3.3. Post	9
1.3.4. Chat	10
1.3.5. Profil	12
2. Projekt Architektur	17
2.1. Front-End	17
2.1.1. React Native	17
2.1.2. Aufbau	17
2.2. Back-End	18
2.2.1. Aufbau	18
3. Deployment	19
3.1. Expo	19
3.1.1. Expo Go	19
3.1.2. Übersicht	20
3.1.3. Build	21
3.2. Continuous Integration/Delivery mit Github Actions	22
3.2.1. Frontend	23
3.2.2. Backend	23
3.3. App Stores	24
3.3.1. Apple App Store	24
3.3.2. Google Play Store	25
4. Weiteres Vorgehen	27
4.1. Weiterentwicklung	27
5. Verzeichnisse	28
A. Anhang	32
A.1. Anhang A	32

1. Einleitung

In Zusammenarbeit mit Robin Bär und Mirja Qachar wurde die App VanFan weiterentwickelt. Die nachfolgenden Kapitel dienen als kurzen Überblick, wie der Stand der App vor dem Semester war und wie er heute ist.

1.1. Stand vor der PA

Bei Beginn der PA war die App mit den meisten grundlegenden Features ausgerüstet. Diese umfassen:

- Die gesamte App, die sowohl Frontend als auch Backend beinhaltet, wurde mittels CI/CD umgesetzt.
- Die zwei wichtigsten Features wurden implementiert.
- Es gibt eine Alpha-Version, welche bereits den ersten Testern/Testerinnen zur Verfügung gestellt wurde. Dies ist insbesondere für die Product Owner von Interesse, da sie nun direktes Feedback von potenziellen Anwender/-innen der App erhalten können.

Es wurde erfolgreich eine Rest-API entwickelt, welche bereits in kleinen Teilen von der App verwendet wird. Die DevOps-Umgebung mit Kubernetes und Appveyor sowie das Frontend mit React Native ist aufgesetzt und viel notwendiges Wissen für die Entwicklung der App wurde bereits erlangt. Für ein weitergehendes Bild des Standes vor der PA, referenzieren wir auf das Kapitel 2.2 im Fachtext, welcher sich im Abschnitt A.1 befindet.

1.2. Aufgaben

Durch die Anwendertests, welche im Sommer 2022 von Mirja und Robin vollzogen wurden, haben sich noch einige Bugs und designtechnische Änderungen ergeben, die es anzupassen galt. Es mussten ebenfalls die restlichen Hauptfeatures erstellt werden, wie die Erstellung eines Posts. Ebenfalls mussten bestehende Features noch vollständig über die Rest-API an den Server angebunden werden. Ebenfalls fehlt aktuell noch die korrekte Swagger-Dokumentation für die Rest-API, welche für bestehende und zukünftige Servercalls erstellt werden muss. Der Server muss neu aufgesetzt werden, da wir die Server von der ZHAW nicht weiter zur Verfügung haben. So muss eine neue Serverumgebung gefunden werden, welche auch in Zukunft verwendet werden kann.

1.3. Stand nach der PA

Für die bereits erwähnten Anwendertests wurde die komplette App vom Server getrennt. Dies wurde gemacht, weil wir den Server über die ZHAW gehosted haben und dieser Service während der Semesterpause eingestellt wurde. Um die App wieder an den Server

1. Einleitung

anzuschliessen, mussten wir wie in Abschnitt 2.2 erwähnt, auf einen anderen Service umsteigen.

Nachfolgend werden die fünf Hauptscreens beschrieben, was bei ihnen fertig implementiert wurde und was noch gemacht werden muss.

1.3.1. Feed

Der Feed ist fast komplett implementiert. Man sieht alle Posts von seinen Freunden, was in Abbildung 1 gut ersichtlich ist. Bei einem Post sieht man die Anzahl `Light ups` und die Anzahl `Comments` des Posts sowie die letzten beiden Kommentare, falls Kommentare vorhanden sind. Durch das Drücken auf das `Comment`-Icon oder den `All comments`-Button gelangt man in den `CommentScreen` (Abbildung 2). Des Weiteren kann man auf das Profilbild eines Users drücken und gelangt daraufhin auf dessen Profil (darauf wird in Unterabschnitt 1.3.5 eingegangen). Im `CommentScreen` kann man neue Kommentare schreiben.

1. Einleitung

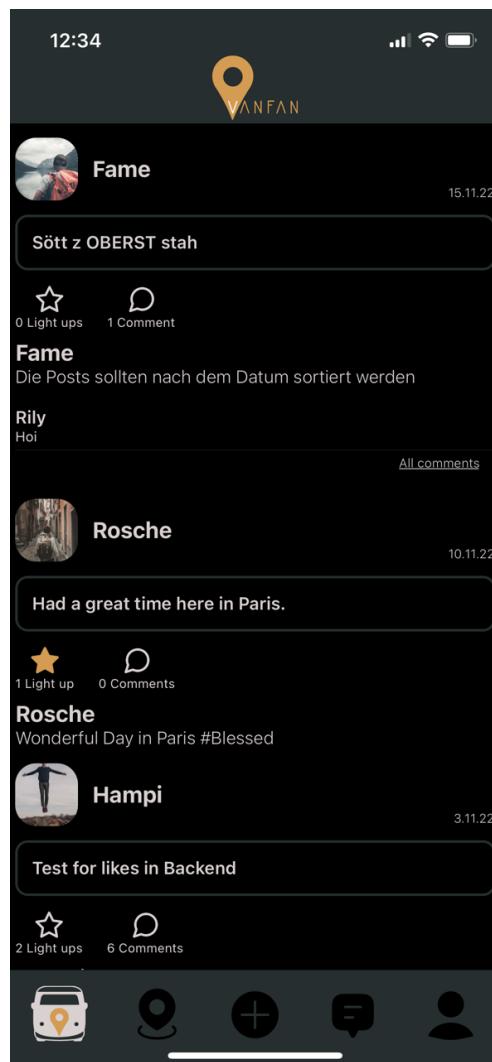


Abbildung 1.: VanFan-App: Ausschnitt des Feeds mit verschiedenen Beiträgen und den dazugehörigen Kommentaren

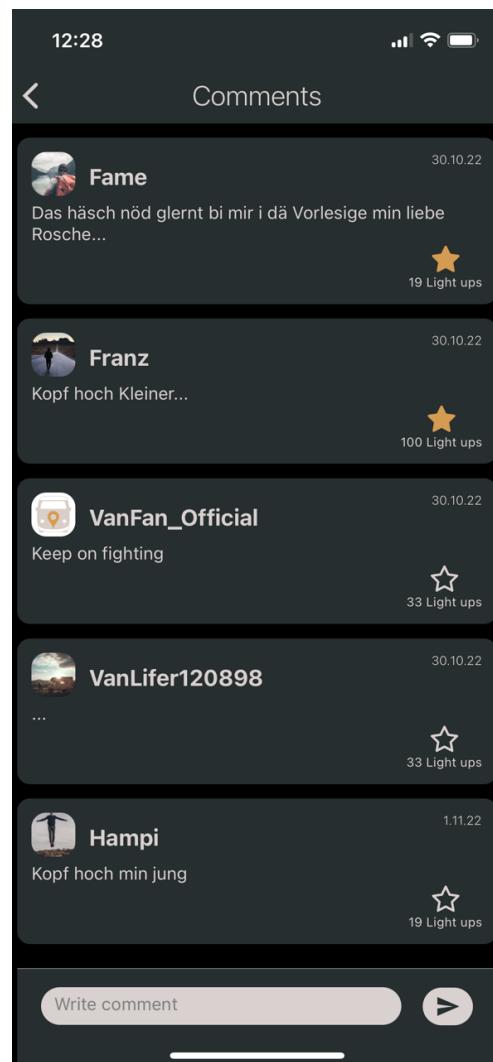


Abbildung 2.: VanFan-App: Ausschnitt des Comment-Screens mit den Kommentaren zu einem Beitrag, die nach Datum geordnet sind

Nicht implementiert

Im FeedScreen wurde Folgendes noch nicht an den Server angeschlossen:

- Die Profilbilder
- Ob ein Post geliked wurde oder nicht

Die Profilbilder werden von unserem **ServerMock** geholt, da wir noch keine Bilder auf der Datenbank speichern können. Dabei gibt es jedoch nur Profilbilder für diese User, welche wir zu Testzwecken erstellt haben. Für alle User, die anderweitig erstellt wurden, wird ein **default**-Bild angezeigt.

Ob ein Post geliked wurde, wird momentan nur anhand einer Random-Funktion 'ermittelt'.

1.3.2. Map

Die Map ist komplett implementiert. Wenn keine anderen User in der Nähe sind, sieht man auf der Map und auch auf dem Modal keine anderen User. Zudem wird auf dem Modal (kann mit dem Button oben rechts geöffnet werden) eine Nachricht angezeigt, welche sagt, dass keine anderen User momentan in der Nähe sind. Dies ist alles gut in Abbildung 3 ersichtlich. Zudem sieht man den eigenen Standort auch auf der Karte (blauer Punkt). Andere User kann man jedoch nur sehen, wenn man seinen eigenen Standort auch mit anderen Usern teilt. Teilt man seinen eigenen Standort nicht, wird wie in Abbildung 4 gut ersichtlich, der User benachrichtigt, dass er die anderen User nicht sehen kann. Vergleicht man Abbildung 3 mit Abbildung 4 ist gut erkennbar, das im ersten Bild der eigene Standort mit den anderen Usern geteilt wird.

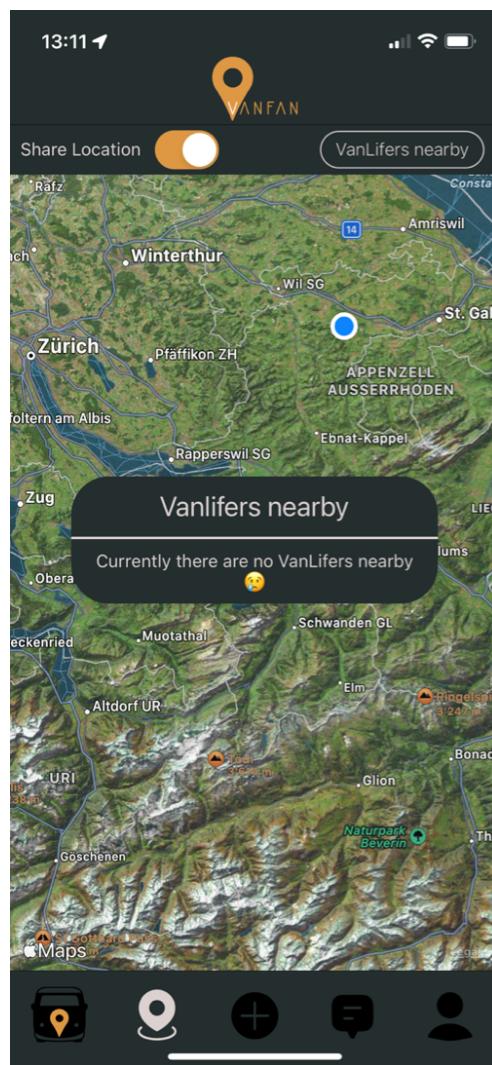


Abbildung 3.: VanFan-App: Die Karte mit dem eigenen Standort. Es sind keine Users um den eigenen Standort zu sehen und die Nachricht sagt dem User zudem, dass sich keine User in der Nähe befinden.

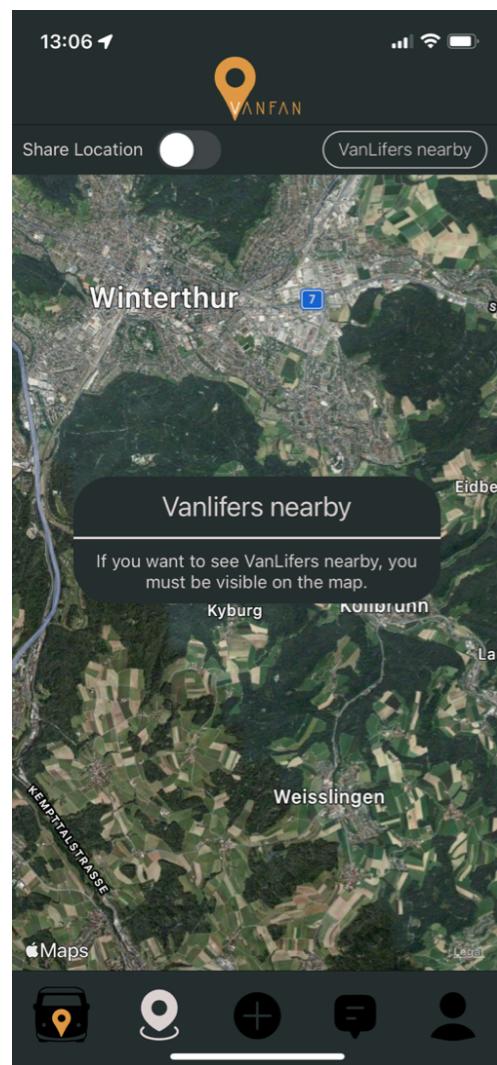


Abbildung 4.: VanFan-App: Der eigene Standort wird nicht geteilt, somit kann man auch andere User in der Nähe nicht sehen. Die Nachricht auf dem Modal sagt genau dies auch dem User.

1. Einleitung

Teilt man seinen eigenen Standort und es befinden sich auch andere User in der Nähe, sieht man diese auf der Map (Abbildung 5) sowie im Modal (Abbildung 6). Zur Sicherheit der einzelnen User wird **nicht** der korrekte Standort angezeigt.

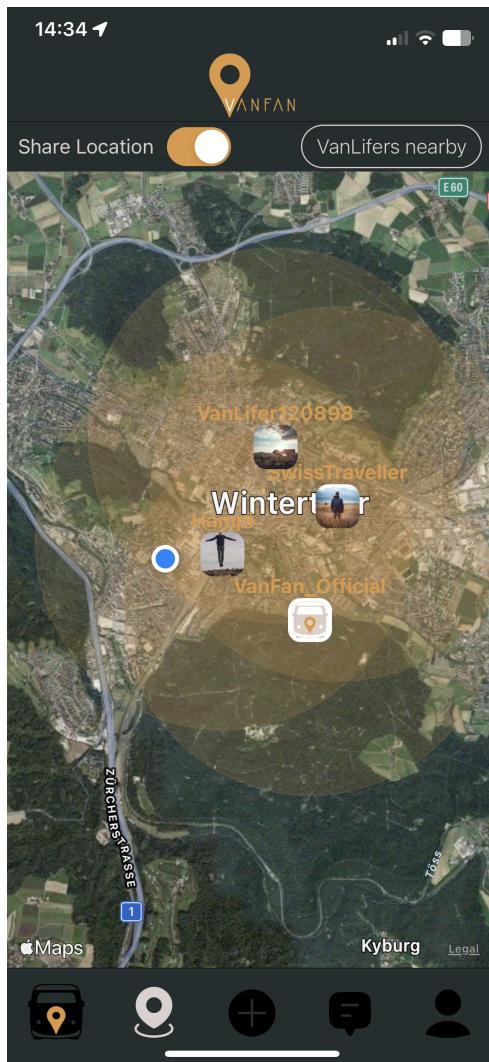


Abbildung 5.: VanFan-App: Alle User die sich gerade in der Nähe befinden, sind auf der Karte ersichtlich.

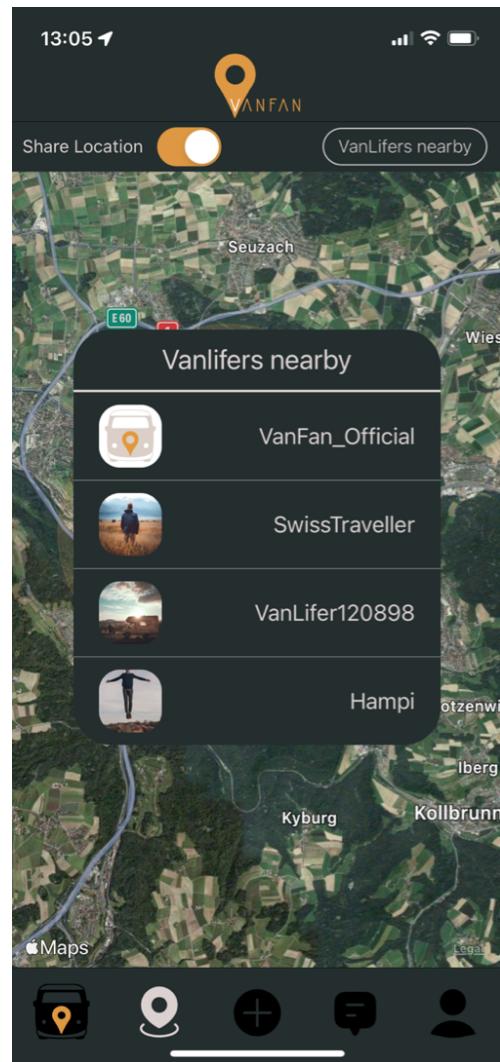


Abbildung 6.: VanFan-App: Alle User die sich gerade in der Nähe befinden, sind im Modal ersichtlich.

1.3.3. Post

Der Screen zur Erstellung von Posts funktioniert auch komplett über den Server. Wie in Abbildung 7 ersichtlich ist, kann man einen Textbeitrag erstellen. Dafür gibt es ein Textfeld, um den Haupttext einzufügen, sowie ein zweites Textfeld, um eine kurze Beschreibung hinzuzufügen. Diese Beschreibung ist jedoch hauptsächlich für Posts mit Bildern oder Videos gedacht.

1. Einleitung

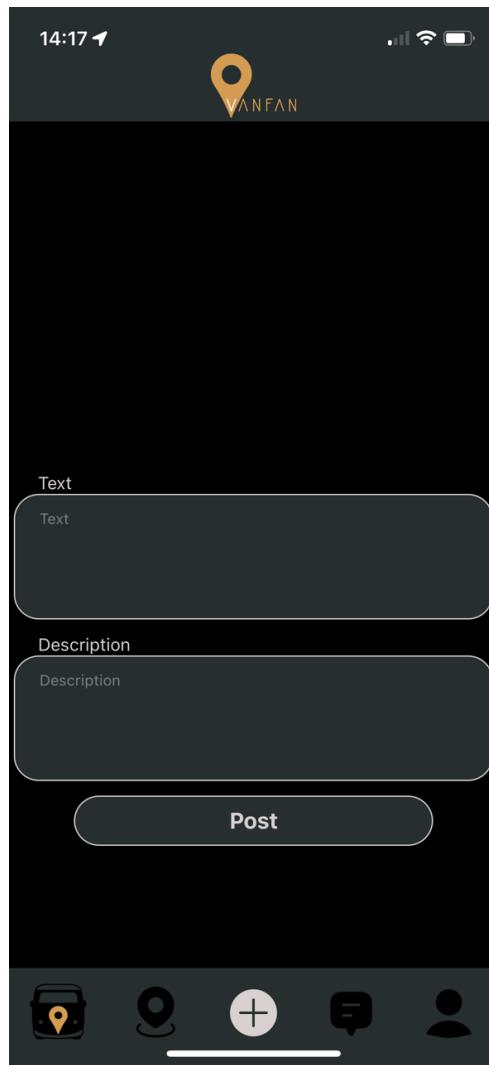


Abbildung 7.: VanFan-App: Der PostScreen
um einen Textbeitrag zu erstellen.

Nicht implementiert

Im PostScreen wurde Folgendes noch nicht Implementiert:

- Bildbeiträge können nicht erstellt werden.
- Videobeiträge können nicht erstellt werden.

Da wir, wie oben bereits erwähnt, keine Bilder und Videos auf der Datenbank speichern können, ist bis jetzt nur das Erstellen von Textbeiträgen möglich. Wenn auf Backendseite das Einbinden von Bildern und Videos auf der Datenbank implementiert wurde, kann man jedoch mit geringem Aufwand Bilder und Videos auf Frontendseite hinzufügen.

1.3.4. Chat

Abbildung 8 zeigt den ChatScreen, hier sieht man die Übersicht aller Chats die man mit anderen Usern hat. Rechts unten sieht man zudem einen Button, mit welchem man

1. Einleitung

einen neuen Chat starten kann. Drückt man auf diesen Button, kommt man in den `CreateNewChatScreen` (Abbildung 9). In diesem Screen werden einem alle user angezeigt, denen man folgt. Mit diesen Usern kann man direkt einen Chat starten. Möchte man jedoch einen User suchen, kann man diesen oben ins Suchfeld eingeben und man bekommt eine Auswahl den Usern mit der gegebenen Suchanfrage. Wenn man dann einen Chat erstellt hat, oder auf einen bestehenden Chat gedrückt hat, kommt man in den `ChatConversationScreen`, welcher in Abbildung 10 zu sehen ist.

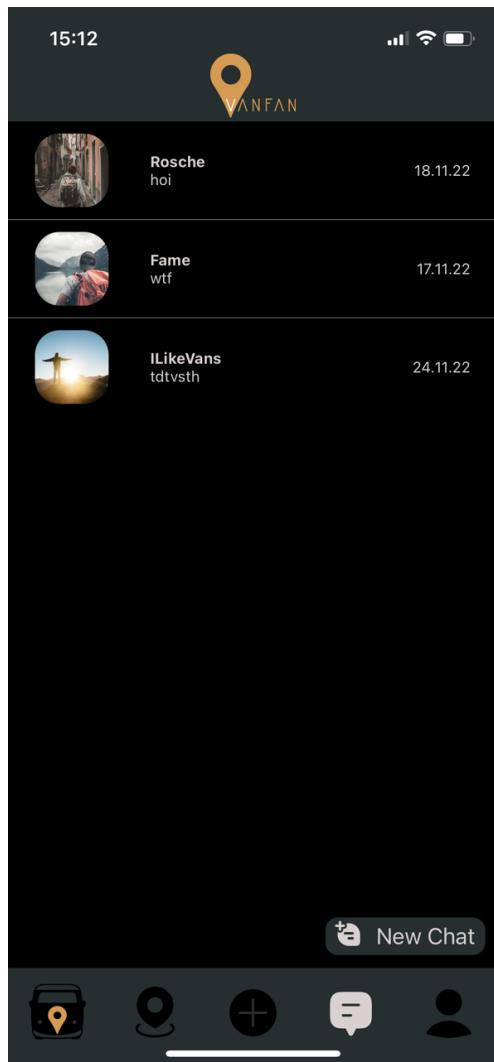


Abbildung 8.: VanFan-App: Eine Übersicht aller Chats die man bisher hat.

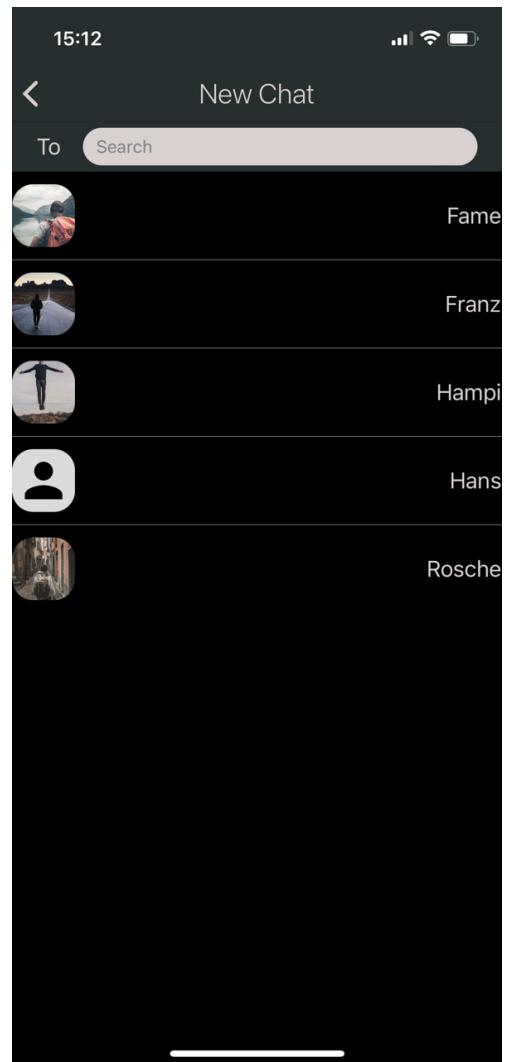


Abbildung 9.: VanFan-App: Die Auswahl an Usern für einen neuen Chat, wenn man keine Sucheingabe gemacht hat..



Abbildung 10.: VanFan-App: Ein Chat mit einem anderen User.

Nicht implementiert

Im **ChatScreen** konnte Folgendes aus Zeitgründen nicht implementiert werden:

- Ein User wird benachrichtigt, wenn eine Nachricht empfangen wurde.
- Die Übersicht der Chats ist nicht in der korrekten Reihenfolge.

1.3.5. Profil

Das eigene Profil und das Profil von anderen Usern unterscheidet sich nur minim. Bei einem fremden Profil wird noch ein Button angezeigt, mit welchem man einem User folgen oder entfolgen kann. Bei einem fremden Profil hat man auch nicht die Option, um in die Einstellungen zu gelangen (rechts oben). Man kann dafür einige Aktionen öffnen, die man bei diesem User ausführen kann. Dies ist in Abbildung 11 gut ersichtlich.

1. Einleitung

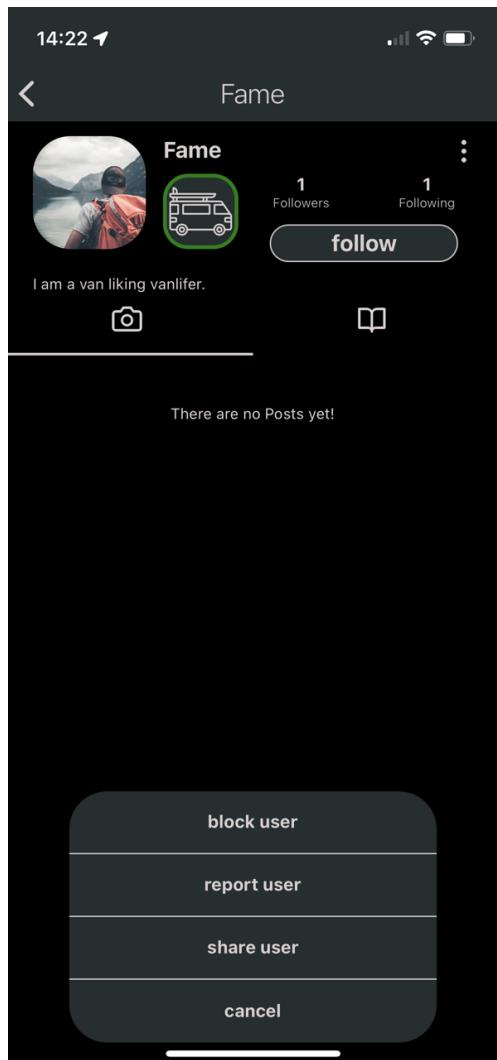


Abbildung 11.: VanFan-App: Ein fremdes Profil mit dem offenen Tab für die Bild- und Videobeiträge sowie dem offenen Modal für mehrere Aktionen.

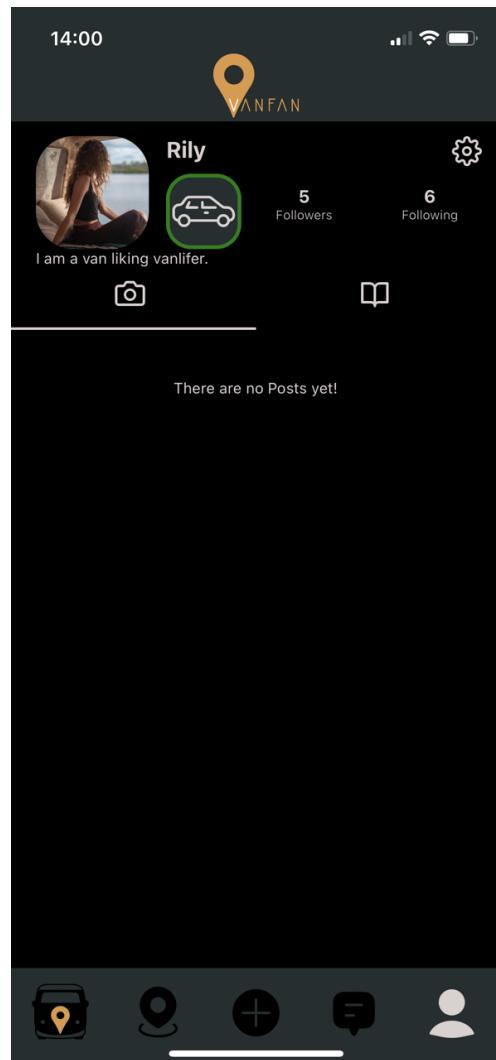


Abbildung 12.: VanFan-App: Das eigene Profil mit dem offenen Tab für die Bild- und Videobeiträge.

In den Profilen kann man auf das Buch-Icon drücken und gelangt daraufhin in die Übersicht der Textbeiträge. Dies ist in Abbildung 13 gut zu sehen. Weiter kann man von jedem User die Follower und die User sehen, denen man folgt (Abbildung 14). Auch hier gelangt man durch das Drücken auf einen User auf dessen Profil.

1. Einleitung

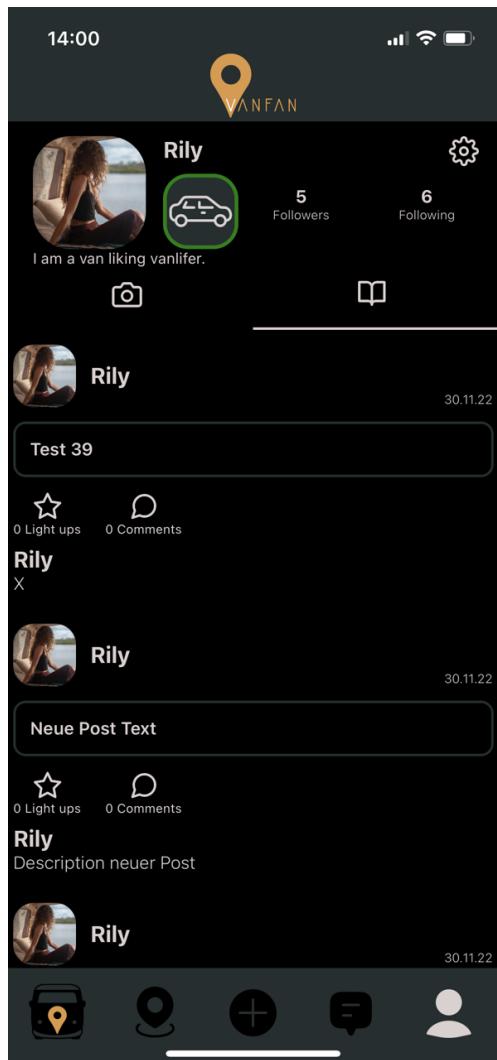


Abbildung 13.: VanFan-App: Das eigene Profil mit dem offenen Tab für die Textbeiträge.

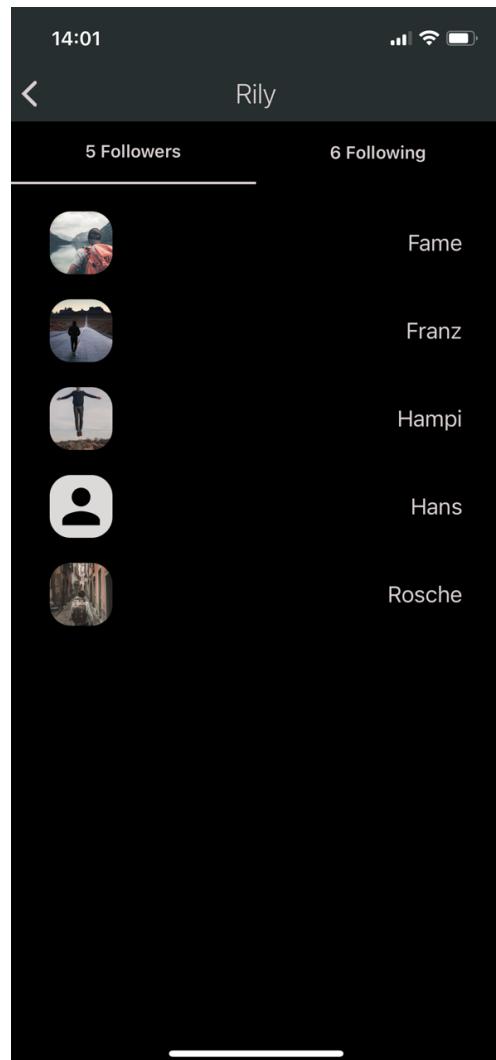


Abbildung 14.: VanFan-App: Die Übersicht aller Follower eines Profils. Drückt man rechts auf Following, sieht man zudem noch alle Users, denen diese Person folgt.

Drückt man beim eigenen Profil (Abbildung 12) auf die Einstellungen rechts oben, gelangt man in die Einstellungen (Abbildung 15 und Abbildung 16). Hier kann man diverse Einstellungen für das eigene Profil vornehmen und mit dem Save-Button speichern. Wenn man oben auf das Zahnradchen drückt, kommt man in die lokalen Einstellungen, welche in Abbildung 17 gezeigt werden.

1. Einleitung

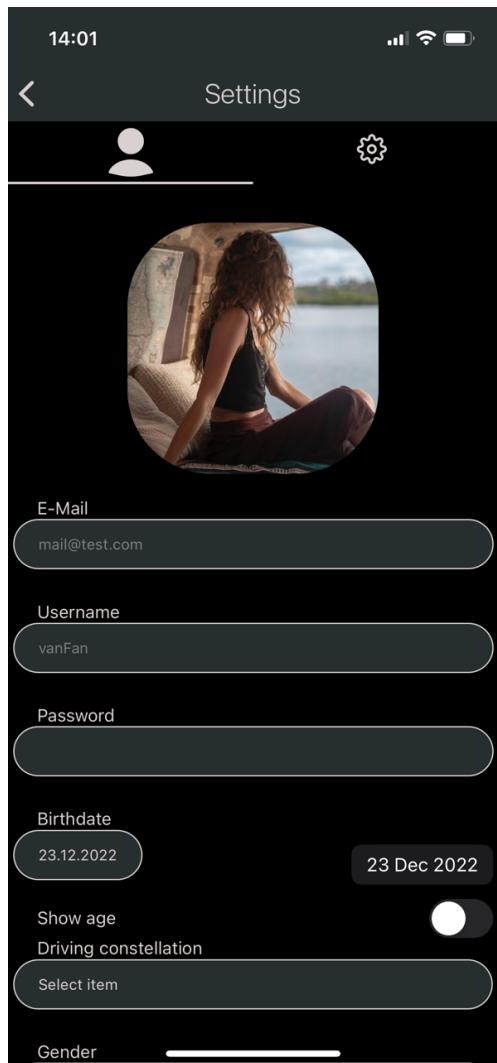


Abbildung 15.: VanFan-App: Die User Einstellungen. Hier können Profilbild, E-Mail, Username, Passwort, Geburtsdatum, die Fahrkonstellation sowie die optionale Anzeige des Alters eingestellt werden.

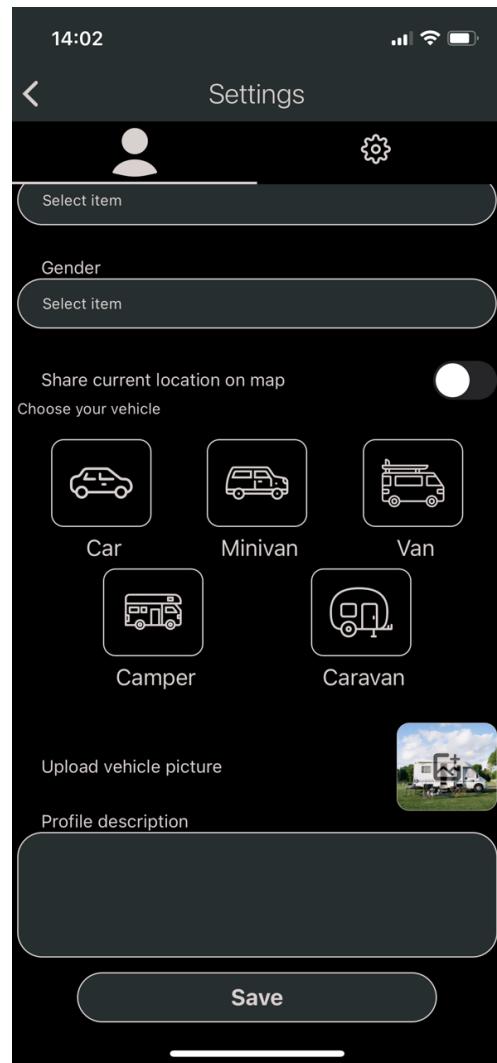


Abbildung 16.: VanFan-App: Die User Einstellungen. Hier können das Geschlecht, die Standortfreigabe, der Fahrzeugtyp, das Van-Bild und die Profilbeschreibung angepasst werden.

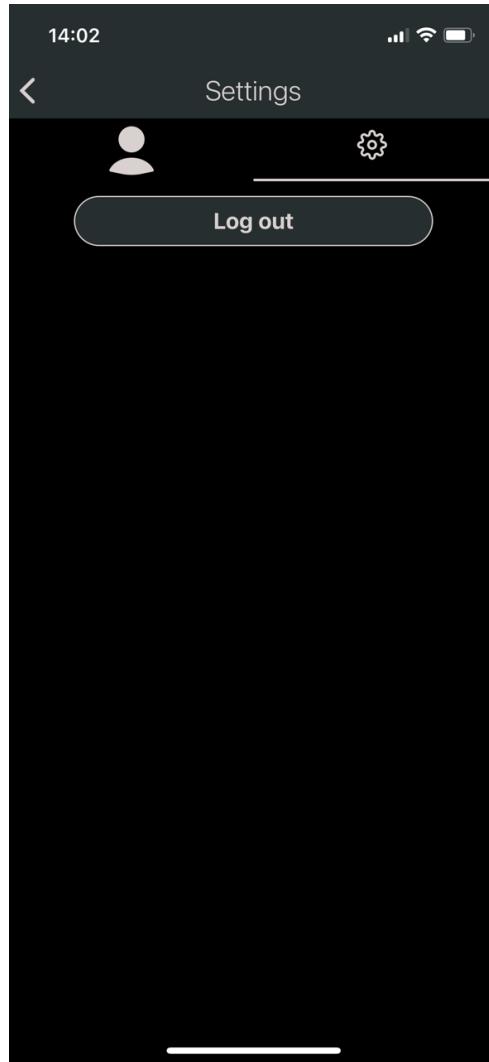


Abbildung 17.: VanFan-App: Die lokalen Einstellungen. Hier kann man sich zurzeit nur ausloggen.

Nicht implementiert

Im ProfileScreen konnte Folgendes aus Zeitgründen nicht implementiert werden:

- Das Folgen und Entfolgen eines Users wurde nicht implementiert.
- Die Aktionen bei einem fremden User wurden nicht implementiert.

2. Projekt Architektur

2.1. Front-End

2.1.1. React Native

Für das Front-End wird react native verwendet [1]. Somit können wir die App für IOS und Android anbieten. Im React Native-Rahmen ist die Klasse App die Hauptkomponente, welche die gesamte Anwendung repräsentiert. Es ist die Wurzelkomponente der App und ist für das Rendern der Benutzeroberfläche verantwortlich.

2.1.2. Aufbau

In der Klasse App geben wir einen `NavigationContainer` zurück, mit dem die verschiedenen Screens geladen werden. Der `Home-Screen` ist selbst ein `TabNavigator` und gibt die fünf Hauptscreens zurück:

- `FeedScreen`: Der Feed, auf welchem die Posts von Freunden angezeigt werden.
- `MapScreen`: Die Map, welche spannende Orte und Freunde anzeigt.
- `PostScreen`: Neuer Post kann hier erstellt werden.
- `ChatScreen`: Zeigt Chats mit Freunden an.
- `ProfileScreen`: Zeigt das eigene Profil an.

Im Ordner `src` befindet sich alles, was mit dem Erstellen der App zu tun hat während sich im Ordner `test` alles befindet, was mit dem Testen zu tun hat.

Die Unterordner des `src`-Ordners:

- `assets`: Ablage aller Bilder, welche auf der App selbst gespeichert werden.
- `colors`: Ablage für die Klassen, welche die Farben bestimmen.
- `hooks`: Ablage für die hooks.
- `language`: Alle Strings, welche auf der App angezeigt werden, werden hier in den verschiedenen Sprachen abgelegt.
- `navigation`: Ablage der Navigationsklassen.
- `screens`: Ablage der verschiedenen Screens.
- `server`: Ablage aller Files, welche mit der Verbindung zur Rest-API zu tun haben.
- `storage`: Ablage für die Verwaltung der lokal zu speichernden Daten.
- `style`: Ablage für alle Stylesheets.
- `utils`: Ablage für alle Hilfsmittel wie Konstanten.

2.2. Back-End

Als Cloud-Plattform verwenden wir Heroku [2], als Database-Programm MongoDB [3] und der Code wird in NodeJS [4], also wieder in JavaScript geschrieben.

2.2.1. Aufbau

Die Klasse `server` startet den Server und verbindet mit der Klasse `app`.

Im Ordner `src` befindet sich alles, was mit dem Erstellen des Servers zu tun hat, so auch die oben erwähnte Klasse `app`. Während sich im Ordner `test` alles befindet, was das Testen betrifft.

Die Unterordner des `src`-Ordners:

- `config`: Welche das swagger-file beinhaltet und die Konfiguration der Authentifizierung.
- `controllers`: Ablage für die controller, welche eingehende Anfragen bearbeiten und Antworten zurückgeben.
- `helper`: Ablage für die Helper.
- `middleware`
- `models`: Für die Schema der verschiedenen Objekte.
- `routes`: Für die routes der verschiedenen Objekte.
- `services`: Für die Services der verschiedenen Objekte.

3. Deployment

In diesem Kapitel wird der Ablauf beschrieben, wie die App vom Sourcecode zur herunterladbaren App auf dem Apple App Store und dem Google Play Store wird.

3.1. Expo

Wie in Unterabschnitt 2.1.1 erwähnt, ist die App auf Frontend Seite mit React Native umgesetzt worden. Mit Expo kann man React Native Anwendungen und Apps für Android und iOS erstellen. Expo stellt verschiedenste Funktionen für React Native Apps zur Verfügung [5].

Um die Funktionen von Expo zu nutzen, müssen zuerst folgende Packages heruntergeladen werden:

- EAS CLI $\geq 0.50.0$
`npm install --global eas-cli`
- Expo SDK $\geq 45.0.0$
- expo-updates $\geq 0.13.0$
`npx expo install expo-updates`

Sind die Packages heruntergeladen, kann man mit den Builds für die App Stores beginnen [6].

3.1.1. Expo Go

Während der Entwicklung mit React Native und Expo kann die App mit Expo Go direkt auf dem Handy getestet werden. In Abbildung 18 sieht man die App **Expo Go**. Darauf sind die momentan laufenden Server zu sehen sowie zuletzt geöffnete Server und auch die Projekte, die man erstellt hat. Drückt man hier auf den momentan laufenden Development Server, wird die App geladen und man kommt in einen neuen Screen, wie in Abbildung 19 zu sehen ist. Ab hier kann man die App so benutzen, wie wenn man sie vom App Store heruntergeladen hätte. Zudem können hier die Änderungen, die man im Code vollzieht, gesehen werden.

3. Deployment

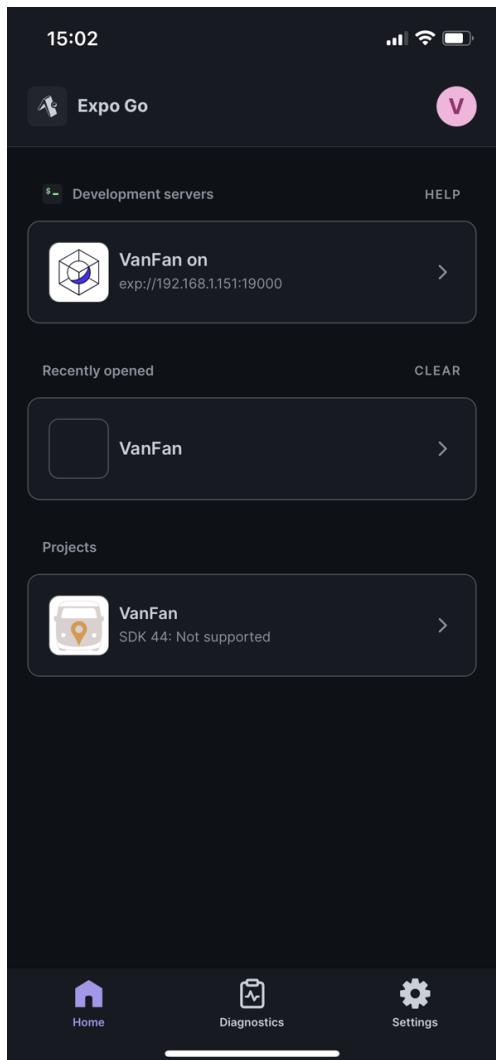


Abbildung 18.: Expo Go auf einem iPhone mit den aktuell laufenden Servern sowie den zuletzt geöffneten Projekten



Abbildung 19.: Die Login-Seite der App die auf Expo Go läuft

3.1.2. Übersicht

Auf der Developer Seite von Expo findet man die Projekte, die für dieses Profil gemacht wurden. Wie in Abbildung 20 zu sehen ist, gibt es das Projekt VanFan. Wählt man das Projekt an, kommt man zur Übersicht der App wie in Abbildung 21. In der Übersicht kann man die Builds und die Submissions, welche für das Erstellen und der Übertragung der App auf die App Stores wichtig sind, sehen.

3. Deployment

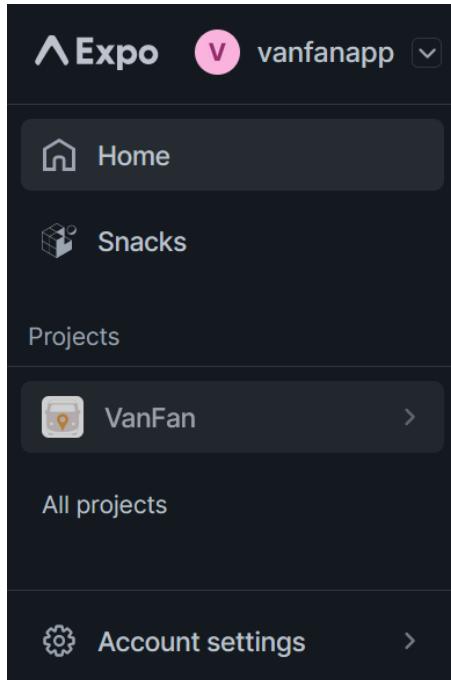


Abbildung 20.: Expo - Übersicht der Projekte des aktiven Profils

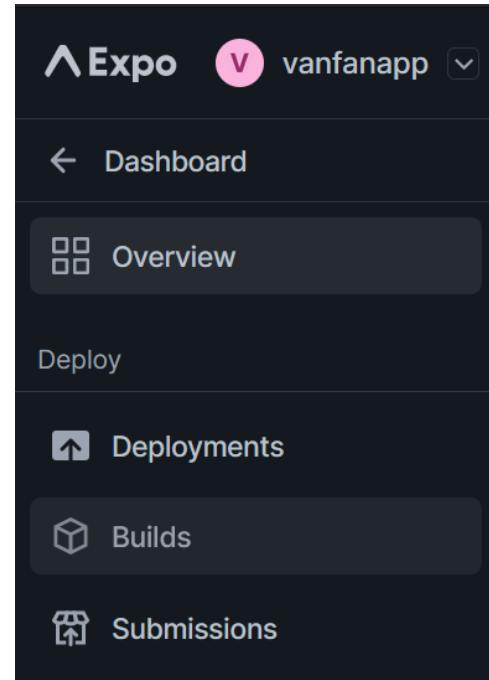


Abbildung 21.: Expo - Übersicht des Projekts mit den Builds und den Submissions

3.1.3. Build

Um einen Build manuell zu starten, muss man die Konsole im Ordner, in welchem das Projekt gespeichert ist, öffnen. Zuerst muss man sich mit dem Profil von Expo anmelden. Danach kann man mit folgendem Befehl den Build starten:

```
eas build -p <all | ios | android> --auto-submit
```

Durch das `--auto-submit` wird die App direkt nach dem Erstellen automatisch auf die Developer Seiten der App Stores geladen, ohne das man dies selber starten muss [7].

Auf der Developer Seite von Expo findet man dann diese neuen Builds direkt auf der Builds Seite, wie in Abbildung 22 zu sehen ist und die Submissions im Reiter Submissions (Abbildung 23).

3. Deployment

The screenshot shows the 'Builds' section of the Expo Build dashboard. On the left, a sidebar lists navigation options: Dashboard, Overview, Deploy, Deployments, Builds (which is selected), Submissions, Channels, Branches, Updates, Configure, Credentials, Secrets, GitHub, and Settings. The main area displays four recent build logs for the 'VanFan iOS App Store build'. Each log includes a green icon, the build name, the time it was created (9 days ago), and detailed build information:

Profile	Version	Build number	Commit	Duration
production	0.0.1	36	7fdce61*	16m 17s
production	0.0.1	35	569f26e*	10m 57s
production	1.0.0	34	7fdce61*	7m 48s
production				9 days ago

Abbildung 22.: Übersicht der Builds von der App

The screenshot shows the 'Submissions' section of the Expo Build dashboard. The sidebar on the left is identical to the one in Abbildung 22. The main area displays four recent submission logs for the 'App Store submission'. Each log includes a green icon, the submission name, the time it was created (9 days ago), and detailed submission information:

Build	Created by	Submitted
be6247af	@ vanfanapp	Dec 7, 2022 10:12 PM
		9 days ago
		9 days ago
		9 days ago
63e6c615	@ vanfanapp	Dec 7, 2022 12:25 PM
63e6c615	@ vanfanapp	Dec 7, 2022 12:25 PM

Abbildung 23.: Übersicht der Submissions der App zu den App-Stores

3.2. Continuous Integration/Delivery mit Github Actions

Das Continuous Integration und Continuous Delivery dieser App wird mittels GitHub Actions vollzogen. Mit GitHub Actions können workflows erstellt werden, wenn z. B. eine Pull-Request von einem Branch in den Master-Branch gemitget wird.

3.2.1. Frontend

Auf Frontendseite findet man im Projekt im Ordner `.github/workflows` die YAML Datei `eas.deploy.yml`.

```
# This workflow will build the app for iOS and Android
name: EAS deploy
on:
  push:
    branches: [ "main" ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Setup repo
        uses: actions/checkout@v2

      - name: Setup Node
        uses: actions/setup-node@v2
        with:
          node-version: 16.14.2
          cache: npm

      - name: Setup Expo and EAS
        uses: expo/expo-github-action@v7
        with:
          expo-version: latest
          eas-version: latest
          token: ${{ secrets.EXPO_TOKEN }}

      - name: Install dependencies
        run: npm install

      - name: Build apps and submit to stores
        run: eas build --all --auto-submit --non-interactive
```

Um das Deployment Frontend seitig weiterzuführen, müssen keine weiteren Vorkehrungen getroffen werden.

Durch diese Datei wird sichergestellt, dass bei jedem Merge in den `main`-Branch die App für Android und iOS neu erstellt und direkt übermittelt wird.

3.2.2. Backend

Das Deployment auf Backendseite wird etwas anders vollzogen als beim Frontend. Wie in Abschnitt 2.2 erwähnt, wird das Backend auf Heroku gehostet. Heroku hat ein eigenes Deployment das mit [GitHub.com](#) funktioniert. Dazu muss man sich nur mit dem GitHub Profil verbinden und dann wird bei jedem `push` oder `merge` in den `main`-Branch das Projekt neu erstellt. Wie Abbildung 24 zeigt, ist auf dem Dashboard von Heroku zu sehen, dass man mit GitHub verbunden ist.

3. Deployment

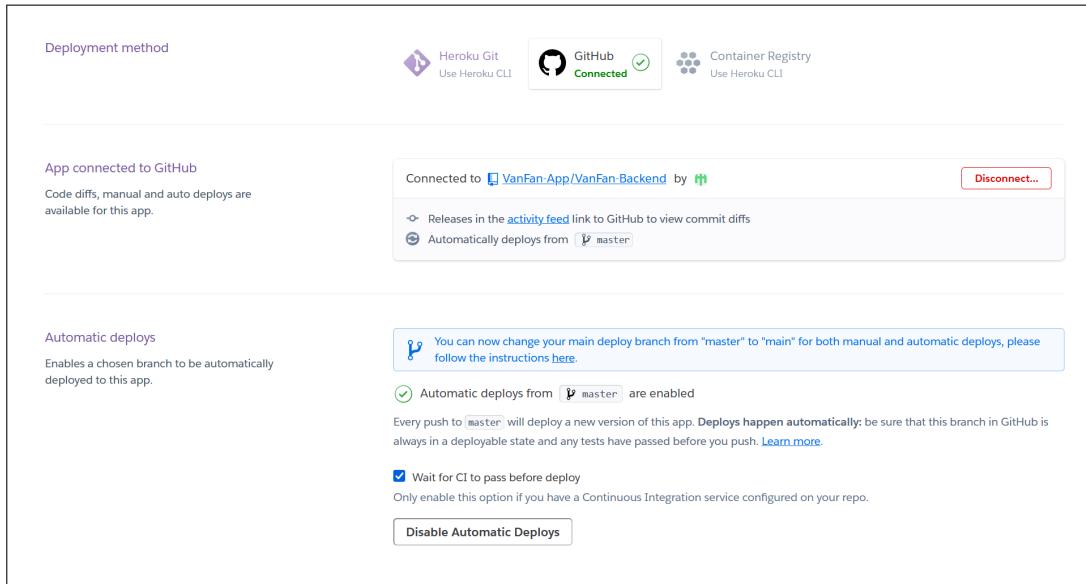


Abbildung 24.: Deployment Methode auf Heroku

Um das Deployment Backend seitig weiterzuführen, muss die App auf Heroku zuerst vom bisherigen GitHub Profil getrennt werden, um danach wieder mit einem autorisierten GitHub Profil zu verbinden, das Zugriff auf die Organisation von VanFan hat.

3.3. App Stores

Die Apps sind momentan nur für Testuser auf den Stores zur Verfügung.

3.3.1. Apple App Store

Auf der Übersichtseite von der App auf App Store Connect, findet man die Informationen zur App. Im Reiter **TestFlight** findet man die Versionen, die aktuell zur Verfügung stehen. Es muss jedoch nicht zwingend sein, dass diese Versionen auch für die Testuser veröffentlicht wurden. Es gibt zwei verschiedene Testarten, die internen und die externen Tests (Abbildung 25). Sobald eine neue Version von Expo auf App Store Connect submitted wurde, kann man diese Version genehmigen lassen. Für die interne Testgruppe ist diese Version dann direkt verfügbar. Wenn man diese Version auch für die externe Testgruppe zur Verfügung stellen möchte, muss man diese Gruppe manuell hinzufügen.

3. Deployment

The screenshot shows the 'TestFlight' tab selected in the VanFan app dashboard. On the left, there's a sidebar with 'Builds' (iOS), 'Feedback' (Crashes, Screenshots), and two main sections: 'Interne Tests' (Development) and 'Externe Tests' (Alpha Testing). The 'Externe Tests' section is expanded, showing a group named 'Alpha Testing'. It includes a link to an external URL (<https://testflight.apple.com/join/vrlzoaYB>), a button to deactivate the link, and a button to copy the link. Below this, there's a section for 'Tester-Feedback' with a note that feedback is activated and a 'Deaktivieren' button.

Abbildung 25.: Übersicht im Reiter **TestFlight** mit den Testgruppen und der Testverwaltung für die externe Testgruppe **Alpha Testing**.

Interne Tests

Um User für das interne Testen hinzuzufügen, muss man die E-Mail-Adressen der User eingeben, die man hinzufügen möchte. Die internen Testuser bekommen dann eine Einladungsmail, um die App herunterzuladen. Zudem werden die Testuser bei jeder neuen Version, die zur Verfügung gestellt wird, benachrichtigt.

Externe Tests

Das Hinzufügen von externen Testusern funktioniert gleich wie bei den internen Testusern. Zudem kann man jedoch einen öffentlichen Link aktivieren und diesen Link den Usern geben, die die App herunterladen möchten. Dies ist in Abbildung 25 gut zu sehen.

3.3.2. Google Play Store

Auf der Google Play Console findet man die Informationen zur App. Auch hier findet man eine Übersicht der Versionen, die auf den Play Store zur Verfügung gestellt werden können. Bei der Google Play Console gibt es drei verschiedene Arten für das Testing, wie in Abbildung 26 ersichtlich ist. Anders als bei TestFlight von Apple, muss bei der Google Play Console von jedem einzelnen User die E-Mail-Adresse angegeben werden, um die App herunterladen zu können. Dazu kann man eigene E-Mail-Listen erstellen, und diese den verschiedenen Testarten hinzufügen. In Abbildung 27 sieht man, dass die Testgruppe **Development** für das Internal Testing zugelassen ist.

3. Deployment

The screenshot shows the Google Play Console interface for internal testing. On the left, there's a sidebar with navigation links like Dashboard, Inbox, Statistics, Publishing overview, Release (with sub-options like Releases overview, Production, Testing, Internal testing, Pre-registration, Pre-launch report, and Reach and devices), and Testers. The main content area is titled "Internal testing" and has a sub-header "Create and manage internal testing releases to make your app available to up to 100 internal testers. [Learn more](#)". It shows a "Track summary" section with "Active" and "Draft release: 0.0.1". Below this are two tabs: "Releases" (which is selected) and "Testers". The "Releases" tab displays a list of releases, starting with "0.0.1" which is currently a draft. There are buttons for "View release details", "Discard release", and options to "Show summary" or "Promote release".

Abbildung 26.: Übersicht im Reiter **Internal testing** mit der Übersicht des Releases.

This screenshot shows the same Google Play Console interface, but the "Testers" tab is now selected. The main content area is titled "Internal testing" and has a sub-header "Create and manage internal testing releases to make your app available to up to 100 internal testers. [Learn more](#)". It shows a "Track summary" section with "Active" and "Draft release: 0.0.1". Below this is the "Testers" tab, which displays a table of test groups. The table has columns for "List name", "Users", and "Feedback URL or email address". Two groups are listed: "AlphaTesters" (10 users) and "Development" (7 users). There are also buttons for "Create email list" and "Feedback URL or email address".

Abbildung 27.: Übersicht der Testgruppen im Internal testing.

4. Weiteres Vorgehen

In Abschnitt 1.3 wurde der Fortschritt des Projekts festgehalten. Auch wenn der Grossteil der App bereits fertig implementiert ist, gibt es noch ausstehende Aufgaben. Leider können wir die App nicht weiterentwickeln und die Product Owner müssen ein neues Entwicklungsteam finden.

4.1. Weiterentwicklung

Es gibt noch einige Dinge, die getan werden müssen. Die Wichtigsten davon sind bereits als Issue im Github Repository hinterlegt. Grundsätzlich kann festgehalten werden, dass die Grundkomponenten erstellt wurden und an den Server angeschlossen sind, jedoch fehlen viele Kleinigkeiten noch, wie Optionen oder Aktionen mit Komponenten oder Mitnutzern. Es gibt einige Dinge, die die Product Owner bei der Suche nach dem neuen Team berücksichtigen sollten. Zunächst einmal muss das Team über die notwendigen Fähigkeiten und Erfahrungen verfügen, um das Produkt erfolgreich zu entwickeln. Dazu gehören Kenntnisse in der Programmierung und im Projektmanagement, aber auch in der spezifischen Technologie, die für das Produkt verwendet wird.

Zusammenfassend lässt sich sagen, dass die Entwicklung unserer App noch nicht abgeschlossen ist. Es gibt noch einige Funktionen, die hinzugefügt werden müssen und es werden sicherlich noch einige Bugs auftauchen, sobald die App häufiger in Benutzung ist. Doch sind wir zuversichtlich, dass ein gutes Team von Entwicklern diese App in kurzer Zeit auf den Markt bringen kann.

5. Verzeichnisse

Literaturverzeichnis

- [1] reactnative. (2022) *getting started*. [Online]. URL: <https://reactnative.dev/docs/getting-started>
- [2] heroku. (2022) *how heroku works*. [Online]. URL: <https://devcenter.heroku.com/articles/how-heroku-works>
- [3] mongodb. (2022) *What is MongoDB Atlas?* [Online]. URL: <https://www.mongodb.com/docs/atlas/>
- [4] nodejs. (2022) *getting started guide*. [Online]. URL: <https://nodejs.org/en/docs/guides/getting-started-guide/>
- [5] Expo. (2022) *What is Expo*. [Online]. URL: <https://docs.expo.dev/introduction/expo/>
- [6] Expo. (2022) *Getting Started*. [Online]. URL: <https://docs.expo.dev/eas-update/getting-started/>
- [7] Expo. (2022) *EAS Build*. [Online]. URL: <https://docs.expo.dev/build/introduction/>

Abbildungsverzeichnis

1.	VanFan-App: Ausschnitt des Feeds mit verschiedenen Beiträgen und den dazugehörigen Kommentaren	7
2.	VanFan-App: Ausschnitt des Comment-Screens mit den Kommentaren zu einem Beitrag, die nach Datum geordnet sind	7
3.	VanFan-App: Die Karte mit dem eigenen Standort. Es sind keine Users um den eigenen Standort zu sehen und die Nachricht sagt dem User zudem, dass sich keine User in der Nähe befinden.	8
4.	VanFan-App: Der eigene Standort wird nicht geteilt, somit kann man auch andere User in der Nähe nicht sehen. Die Nachricht auf dem Modal sagt genau dies auch dem User.	8
5.	VanFan-App: Alle User die sich gerade in der Nähe befinden, sind auf der Karte ersichtlich.	9
6.	VanFan-App: Alle User die sich gerade in der Nähe befinden, sind im Modal ersichtlich.	9
7.	VanFan-App: Der PostScreen um einen Textbeitrag zu erstellen.	10
8.	VanFan-App: Eine Übersicht aller Chats die man bisher hat.	11
9.	VanFan-App: Die Auswahl an Usern für einen neuen Chat, wenn man keine Sucheingabe gemacht hat..	11
10.	VanFan-App: Ein Chat mit einem anderen User.	12
11.	VanFan-App: Ein fremdes Profil mit dem offenen Tab für die Bild- und Videobeiträge sowie dem offenen Modal für mehrere Aktionen.	13
12.	VanFan-App: Das eigene Profil mit dem offenen Tab für die Bild- und Videobeiträge.	13
13.	VanFan-App: Das eigene Profil mit dem offenen Tab für die Textbeiträge.	14
14.	VanFan-App: Die Übersicht aller Follower eines Profils. Drückt man rechts auf Following, sieht man zudem noch alle Users, denen diese Person folgt.	14
15.	VanFan-App: Die User Einstellungen. Hier können Profilbild, E-Mail, Username, Passwort, Geburtsdatum, die Fahrkonstellation sowie die optionale Anzeige des Alters eingestellt werden.	15
16.	VanFan-App: Die User Einstellungen. Hier können das Geschlecht, die Standortfreigabe, der Fahrzeugtyp, das Van-Bild und die Profilbeschreibung angepasst werden.	15
17.	VanFan-App: Die lokalen Einstellungen. Hier kann man sich zurzeit nur ausloggen.	16
18.	Expo Go auf einem iPhone mit den aktuell laufenden Servern sowie den zuletzt geöffneten Projekten	20
19.	Die Login-Seite der App die auf Expo Go läuft	20
20.	Expo - Übersicht der Projekte des aktiven Profils	21
21.	Expo - Übersicht des Projekts mit den Builds und den Submissions	21
22.	Übersicht der Builds von der App	22
23.	Übersicht der Submissions der App zu den App-Stores	22

Abbildungsverzeichnis

24.	Deployment Methode auf Heroku	24
25.	Übersicht im Reiter TestFlight mit den Testgruppen und der Testverwaltung für die externe Testgruppe Alpha Testing	25
26.	Übersicht im Reiter Internal testing mit der Übersicht des Releases.	26
27.	Übersicht der Testgruppen im Internal testing.	26

A. Anhang

A.1. Anhang A

ZHAW
Zürcher Hochschule für Angewandte Wissenschaften
Technikumstrasse 9
8400 Winterthur

School of Engineering
Frühlingssemester 2022

Fachartikel
VanFan

Dozent: Andreas Meier
NoTecS: Rifat Gürboy
Studenten: Alain Michienzi, Dario Haas, Erni Nano, Jan Ledergerber,
Ramon Imper, Roger Wetter, Timo Nigg

Winterthur, 27.05.2022

Abstract

Im Rahmen des Software-Projekts 4 erarbeitete das Entwicklerteam bestehend aus sieben Studierenden der ZHAW in Zusammenarbeit mit Robin Bär und Mirja Qachar die App VanFan. VanFan ist eine Kontaktplattform für Van-Enthusiasten/Van-Enthusiastinnen. Dazu wurde zuerst eine Projektskizze erstellt, um ein grobes Konzept der gewünschten Applikation zu erhalten. Danach wurden in fünf Sprints à zwei Wochen diverse Features implementiert. Der Entwicklungsstand dieser Features wurde evaluiert. Ebenfalls werden Erfolge und Rückschläge, die während der Entwicklung entstanden, angesprochen. Abschliessend wird geprüft, ob sich eine Weiterentwicklung lohnen könnte und was für eine Veröffentlichung des Produkts noch getätigten werden müsste.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Ziel.....	1
1.2	Umsetzung.....	1
1.3	Risiken.....	2
1.4	Kosten	2
2	Resultate	4
2.1	Analyse Zielerreichung	4
2.2	Stand des Produkts.....	4
2.2.1	Settings.....	4
2.2.2	Feed.....	5
2.2.3	Karte	6
2.2.4	Chat	6
2.3	Zusammenarbeit & Entwicklung	6
3	Diskussion und Ausblick	7
3.1	Weiterentwicklung	7
3.2	Offene Aufgaben	7
4	Literaturverzeichnis.....	8
5	Abbildungsverzeichnis	8
6	Tabellenverzeichnis	8

1 Einleitung

In Zusammenarbeit mit Robin Bär und Mirja Qachar soll die App VanFan entwickelt werden. Sie sind die Product Owner der App. Die nachfolgenden Kapitel dienen als verkürzte Projektskizze, damit sich sowohl Entwickler als auch Product Owner auf die kommenden Arbeiten einstellen können.

1.1 Ziel

In der Schweiz sowie in Europa hat der Verkauf von Vans in den letzten Jahren und vor allem auch in der Zeit der Pandemie regelrecht geboomoht. Ferien in einem mobilen Gefährt sind nicht nur kostengünstig, sie bringen auch zusätzliche Flexibilität, da man nicht ortsgebunden ist. Bei den jüngeren Generationen liegt diese Art zu Reisen im Trend. Camper/-innen sind gesellige Menschen. Sie schätzen das Zusammensein, neue Camper/-innen kennenzulernen oder gemeinsame Abenteuer zu erleben [1].

Mit VanFan soll diese Gemeinsamkeit gestärkt werden. Eine zentrale Plattform für Vanlifer/-innen soll es ermöglichen auf einfache Art und Weise neue Kontakte zu knüpfen. In wenigen Klicken soll ersichtlich sein, wer wo reist. Wichtig dabei ist zu wissen, wer in der Nähe ist, da die Möglichkeit besteht zusammen zu reisen. Mit Hilfe von Fotos, Videos oder Texten soll es möglich sein, die eigen Reise zu dokumentieren und zu teilen zu können.

Damit die App von einer möglichst grossen Community profitieren kann, soll sie kostenlos zur Verfügung gestellt werden. Im Zentrum steht das Knüpfen von sozialen Kontakten und nicht der Profit. Es soll jedoch auch die Option eines Premium-Accounts geben, welcher den Benutzer/-innen Zugriff auf exklusive Features ermöglicht.

1.2 Umsetzung

Die Priorität der App ist das einfache Knüpfen von Kontakten. Eine integrierte Landkarte ist Dreh und Angelpunkt der App. Die Landkarte bietet den Benutzer/-innen die folgenden Optionen:

- Anzeigen des eigenen Standortes, damit andere Benutzer/-innen diesen sehen können.
- Suchfunktion/Anzeigen, wer sich in der Nähe befindet.

Eine weitere Eigenschaft der App ist, dass die Benutzer/-innen ihren Reisestatus in ihrem Profil anzeigen lassen können. Der Reisestatus gibt Auskunft darüber, wie man unterwegs ist. Zum Reisestatus gehören die folgenden Informationen:

- **Status:** Wie reise ich? – allein, als Paar, in einer Gruppe, mit der Familie
- **Konstellation:** Wie ist man unterwegs? – mit dem Auto, dem Minivan, einem Van, einem Camper oder einem Wohnwagen

Damit die verschiedenen Benutzer/-innen miteinander in Kontakt treten können, bietet VanFan eine Chat-Funktion an.

Weitere Anforderungen die von der App berücksichtigt werden:

- **Datenschutz:** Benutzer/-innen geben sensible Daten an bspw. Standort oder Reisestatus. Diese Daten sollen nicht für alle zugänglich sein (Privatisierung der Daten), oder so weit eingeschränkt werden (Verschleierung der Daten), damit sie nicht missbraucht werden können.
- **Offline-Gebrauch:** Je nach dem wo sich die Benutzer/-innen befinden, werden diese keinen Internetzugriff haben. Daher soll die App auch offline funktionieren, obwohl dann nicht alle Features zur Verfügung stehen.
- **Chat:** Um vor Missbrauch (Spam) zu schützen, sollen Benutzer/-innen in der Lage sein, unerwünschte Kontaktanfragen oder Inhalte zu melden und entsprechend den Kontakt blockieren zu können.

1.3 Risiken

Jedes neue Projekt birgt Risiken. Um den Erfolg des Projekts besser steuern zu können, sind nachfolgend die Risiken zusammengefasst:

- I. **Mobile App:** Die App wird mit React Native umgesetzt, damit sollen Android als auch iOS Benutzer/-innen von der App Gebrauch machen können. Jedoch ist React Native neu für das Team. Dies kann zu ungenauen Aufwandsabschätzungen führen und das Projekt in Verzug bringen.
- II. **DevOps:** VanFan setzt auf eine Continuous Integration / Continuous Delivery (CI/DI) Umgebung mit Kubernetes und Cloud Build Server (Appveyor). In diesem Bereich ist im Team nur wenig Wissen vorhanden, wodurch ein erhebliches Risiko besteht.
- III. **Projektzeitrahmen:** Für dieses Projekt ist der Entwicklungszeitraum beschränkt. Es ist daher wichtig, dass sich das Entwicklerteam und die externen Auftraggeber/-innen auf die wichtigsten Features fokussieren.
- IV. **Schnittstellen:** Innerhalb des Entwicklerteams wird zwischen Frontend- und Backend-Team unterschieden. Eine gute Kommunikation ist daher essentiel.

Die folgende Risikomatrix (Tabelle 1) stellt die Eintrittswahrscheinlichkeit im Verhältnis zum möglichen Schaden bezüglich des Erfolgs des Projekts dar.

Eintrittswahrscheinlichkeit	Schaden		
	Tief	Mittel	Hoch
Hoch	II		
Mittel	IV	III	I
Tief			

Tabelle 1: Risikomatrix – Eintrittswahrscheinlichkeit eines Risikos im Verhältnis zum möglichen Schaden der entstehen könnte.

1.4 Kosten

Für die Entwicklung von VanFan steht ein Entwicklerteam von sieben Personen zur Verfügung. Dieses Team arbeitet an der Basisversion, für welche ein Aufwand von insgesamt 840 Arbeitsstunden geschätzt wird. Da die Basisversion im Rahmen des Software-Projekt-Modul an der ZHAW entsteht, fallen in dieser Zeit (zwei Monate) keine Entwicklerkosten an. Ebenfalls gibt es keine Kosten für die Serverlandschaft, da die Infrastruktur der ZHAW benutzen werden darf. Danach wird das Projekt ungefähr vier Monate brauchen, bis der erste Release herausgegeben werden kann. In diesen vier Monaten wird mit ca. 1'500 Arbeitsstunden à CHF 50 gerechnet. Somit entstehen Entwicklungskosten von ca. CHF 75'000.

Für die Inbetriebnahme der App fallen die folgenden Kosten an:

- **iOS:** Jährliche Gebühr von 99 USD. [2]
- **Google-Play:** Einmalige Gebühr von 25 USD. [3]

Für das Hosting fallen monatliche Kosten von CHF 30 an.

Die Basis-App ist kostenlos verfügbar. Bei Bedarf können die Benutzer/-innen auf ein Premium-Account umsteigen. Dieser ist für CHF 20 pro Jahr erhältlich. In den nächsten fünf Jahren wird angenommen, dass die App 50'000-mal heruntergeladen wird. Es wird angenommen, dass sich 3'000 Benutzer ein Premium-Abonnement kaufen werden. Dies ergibt einen Gewinn von CHF 60'000. Durch Werbung, welche auf den kostenlosen-Abonnements laufen wird, sollen zusätzlich CHF 30'000 generiert werden.

Der Break-Even-Point wird gemäss diesen Annahmen nach 58 Monaten erreicht.

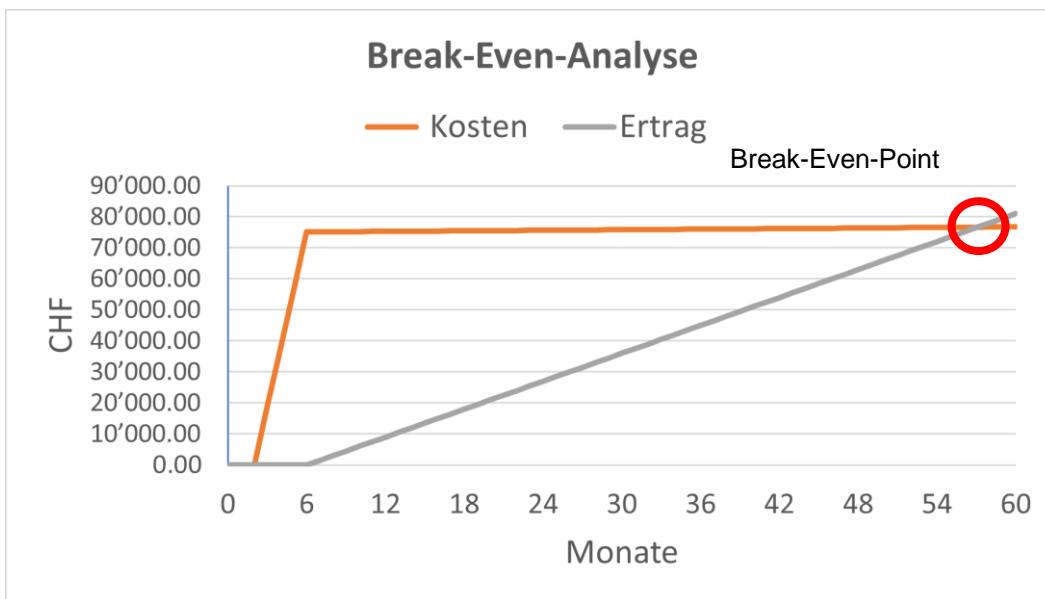


Abbildung 1: Break-Even-Analyse mit Break-Even-Point nach fünf Jahren

Abbildung 1 visualisiert die anfallenden Kosten und erwarteten Erträge. Bei weiteren Kosten (Bugfixes, neuen Features oder Wartung der App) wird sich der Break-Event-Point weiter hinauszögern.

2 Resultate

Das Entwicklerteam arbeitete während fünf Sprints à zwei Wochen an der App. Zurzeit befindet es sich im sechsten Sprint. Dies bedeutet das gewisse Features noch nicht fertig sind, da sie in diesem Moment umgesetzt werden. Nachfolgend werden die Erreichten Ziele und der Stand des Produkts präsentiert.

2.1 Analyse Zielerreichung

Ziel war es nach Abschluss des Moduls Live gehen zu können. Dies umfasst eine App welche:

- einem ersten Testpublikum präsentiert wurde und
- alle wichtigsten Features beinhaltet.

Rückblickend lässt sich sagen, dass dieses Ziel zu hoch war. Dies bedeutet jedoch nicht, dass es keine Erfolge gab.

Während der Entwicklung gab es Teilerfolge. Diese umfassen:

- Die gesamte App, dies beinhaltet Frontend als auch Backend, wurde mittels CI/CD umgesetzt.
- Die zwei wichtigsten Features wurden implementiert.
- Es gibt eine Alpha-Version, welche bereits den ersten Testern/Testerinnen zur Verfügung gestellt wurde. Dies ist insbesondere für die Product Owner von Interesse, da sie nun direktes Feedback von potenziellen Anwender/-innen der App erhalten können.

Weiter können Erfolge im Entwicklerteam festgehalten werden. Es wurde erfolgreich eine Rest API entwickelt. Viele Teammitglieder haben bereits an Rest APIs gearbeitet, jedoch noch nie eine von Grund auf neu erstellt. Die DevOps-Umgebung mit Kuberentes und Appveyor sowie das Frontend mit React Native war neu für das Team. Hier musste das benötigte Wissen zuerst erlangt und dann umgesetzt werden. Dabei gab es immer wieder kleinere Probleme, welche überwunden werden mussten. Diese Probleme wurden im Kollektiv besprochen und man suchte gemeinsam Lösungen. Dank dieser Vorgehensweise konnten diverse Hürden gemeistert werden.

2.2 Stand des Produkts

Die folgende Auflistung ist eine Zusammenfassung aller geplanter Features.

- **Settings:** Ein/-e Benutzer/-in kann sich anmelden/registrieren. Er/Sie hat Zugriff auf sein/ihr Profil und kann dieses bearbeiten.
- **Feed:** Ein/-e Benutzer/-in kann den Feed einsehen und bearbeiten. Auf dem Profil sind alle geposteten Beiträge des Benutzers ersichtlich.
- **Karte:** Ein/-e Benutzer/-in kann mit der Karte interagieren. Benutzer/-innen in der Nähe werden angezeigt und es kann Kontakt aufgenommen werden.
- **Chat:** Ein/-e Benutzer/-in kann mit anderen Benutzer/-innen in Kontakt treten.

Die nachfolgenden Kapitel werden genau auf die einzelnen Features eingehen. Besonderes Augenmerk hatte dabei die Karte, da sie das Alleinstellungsmerkmal von VanFan ist.

2.2.1 Settings

Damit mit der Implementierung der grossen Features begonnen werden konnte, war es wichtig, dass sich Benutzer/-innen auf der App anmelden können. Dies führte dazu, dass Login/Registrierung sowie das Profil als erstes umgesetzt wurde. In den Settings können Einstellungen des Benutzers vorgenommen werden. Diese umfassen:

- Personalien bearbeiten
- Profilbild
- Nickname
- Fahrzeugtyp, mit welchem man am Reisen ist
- Profilbeschreibung

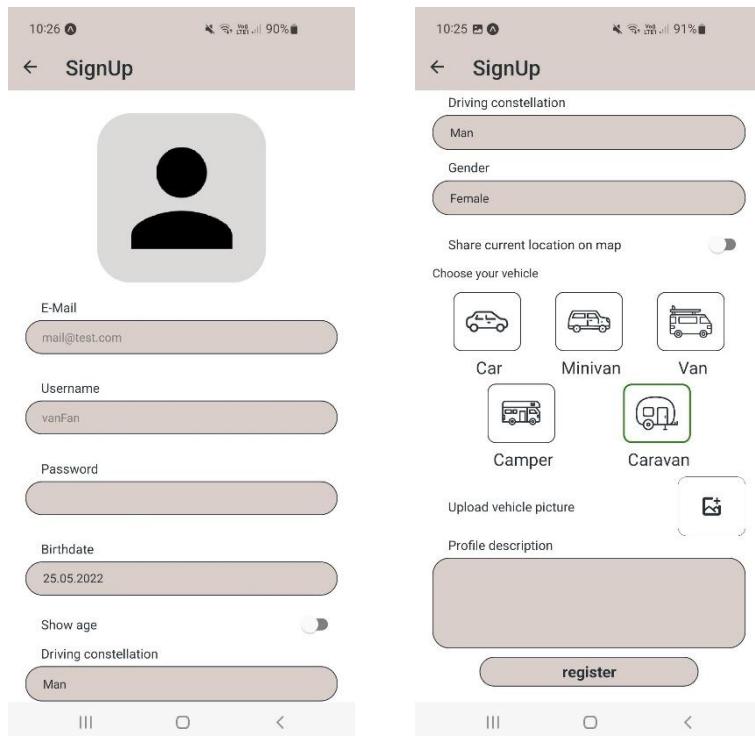


Abbildung 2: VanFan-App – Registrierungsseite mit Anmeldeinformationen

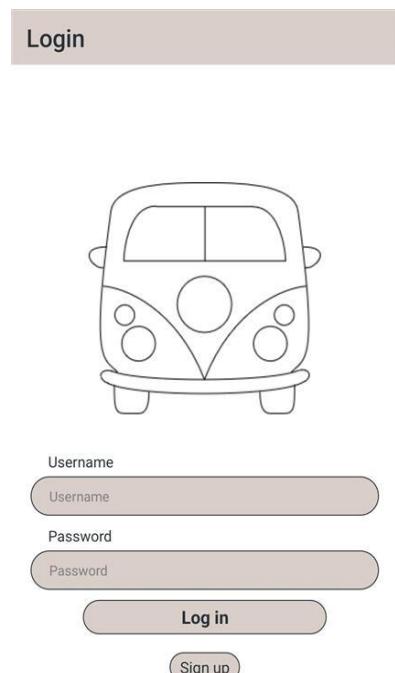


Abbildung 3: VanFan-App – Login-Seite

Abbildung 2 zeigt die Registrierungsseite. Alle Informationen, die hier angegeben werden, können in den Settings wieder angepasst werden.

Abbildung 3 zeigt die Login-Seite, welche von Benutzer/-innen verwendet werden kann, die bereits registriert sind. Benutzer/-innen die regelmässig auf die App zugreifen, müssen sich nicht jedes Mal erneut anmelden, da die Informationen lokal gemerkt werden. Wird die App nach längerer Zeit wieder geöffnet, so ist man gezwungen sich erneut anzumelden.

Dieser Bereich ist vollständig implementiert und funktionsfähig.

2.2.2 Feed

Um eine anschauliche Startseite zu erstellen, wurde das Feed Feature entwickelt. Dadurch können Benutzer/-innen die aktuellen Posts von Freunden direkt auf der Startseite entdecken. Der Feed wurde so erstellt, dass immer nur die neusten Beiträge ersichtlich sind. Die Posts sind untereinander eingereiht um es dem/der Benutzer/-in zu ermöglichen sich durch die Posts zu scrollen.

Zusätzlich können:

- Zu den Profilen der Postverfasser/-innen navigiert werden.
- Benutzer/-innen Posts kommentieren.
- Benutzer/-innen Posts liken.

Abbildung 4 zeigt einen Ausschnitt des Feeds. Es zeigt einen Post von VanFan Official unter welchem auch bereits die ersten Kommentare zu sehen sind.



Abbildung 4: VanFan-App – Ausschnitt des Feeds mit Kommentaren zu einem Post

2.2.3 Karte

Die Karte ist das visuell wichtigste Feature mit der höchsten funktionellen Priorität.

Die Karte bietet folgende Funktionen:

- Anzeigen aller Benutzer/-innen in einem gewünschten Radius.
- Kontaktaufnahme mit einem Benutzer über den Chat.
- Teilen des Standorts kann ein-/ausgeschaltet werden.

Damit andere Benutzer/-innen nicht Zugriff auf den genauen Standort haben, werden die Standorte bei der Anzeige verschleiert.

Diese Karte ist vollständig implementiert und funktionsfähig. Die Kontaktaufnahme über den Chat ist noch nicht vollständig implementiert. Dies Feature gehört jedoch zum Chat, welcher in Kapitel 2.2.4 beschreiben wird.

Abbildung 5 zeigt die Karte mit Benutzer/-innen welche sich in der Nähe befinden. Wenn nicht explizit ein Radius angegeben wird, werden alle Benutzer/-innen, welche sich in einem Umkreis von 5km aufhalten angezeigt. Ebenfalls sieht man den eigenen Standort (blauer Punkt, gelb eingekreist) und die Standorte von anderen Benutzer/-innen (Icons mit Benutzername, rot eingekreist).

2.2.4 Chat

Damit die Benutzer/-innen miteinander in Kontakt treten können brauchen sie einen Chat. Jedem/Jeder Benutzer/-in der App ist es möglich andere Benutzer/-innen anzuschreiben und somit verschiedene Chats mit diversen Benutzern zu führen.

Obwohl dieser wichtig ist, bekam er eine niedrigere Priorität als die Karte. Dieses Feature wurde in Sprint sechs aufgenommen und befindet sich momentan in Bearbeitung.

2.3 Zusammenarbeit & Entwicklung

Nach fünf durchgeführten Sprints kann das Team auf Erfolge und Rückschläge zurückblicken. Die Erfahrung ist dabei nicht ausgeblieben. Gerade im agilen Umfeld mit CI/CD konnte das Team durch die Praxis viel Erfahrung gewinnen.

Die Zusammenarbeit mit den Product Ownern Robin Bär und Mirja Qachar lief sehr gut. Sowohl für sie als auch für das Entwicklerteam war es eine neue Erfahrung. In den jeweiligen Sprint-Planungen entstand ein konstruktiver Dialog, in welchem das Entwickelte präsentiert wurde und neue Features geplant werden konnten. Dies führte auch dazu, dass bereits eine Alpha-Version von VanFan an die ersten Benutzer/-innen geben werden konnte.

Rückschläge gab es im Entwicklungsbereich. Dabei kann auf zwei grosse Events zurückgeblickt werden.

1. **DevOps:** Das Aufsetzen der Kubernetes-Umgebung hat mehr Zeit in Anspruch genommen als ursprünglich angenommen. Dies führte dazu, dass geplante Arbeiten vom 2. in den 3. Sprint verschoben werden mussten.
2. **React Native:** Wie in der Risikomatrix bereits angesprochen, ist React Native neu für das Team. Dies wurde in den Features, welche mit React Native zu tun hatten, beachtet. Das Testen gestaltete sich jedoch schwieriger als ursprünglich angenommen.

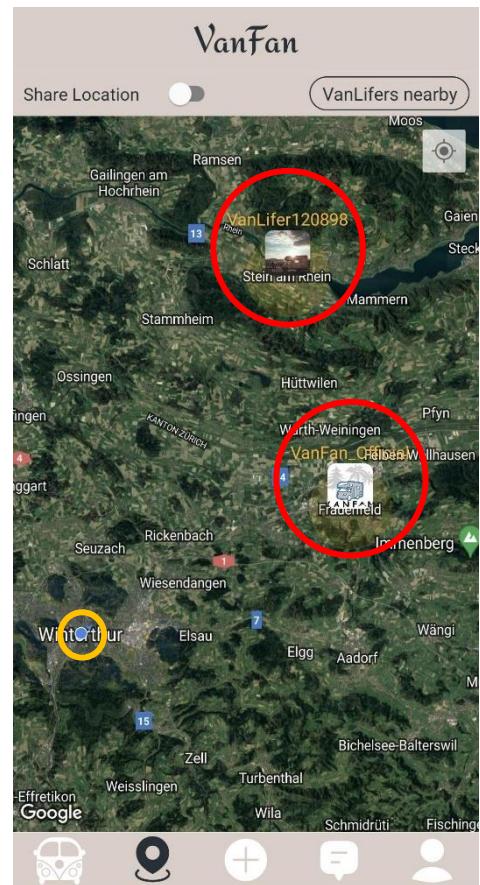


Abbildung 5: VanFan-App – Karte mit eigenem Standort und Benutzer/-innen in

3 Diskussion und Ausblick

In Kapitel 2 Resultate wurde der Fortschritt des Projekts festgehalten. Da sich die App erst in der Alpha-Version befindet gibt es noch ausstehende Aufgaben. Außerdem möchten die Product Owner, dass die App fertiggestellt wird. Die folgenden Kapitel beschäftigen sich mit der Weiterentwicklung von VanFan. Dabei werden zwei Fragen beantworten:

1. Ist die Weiterentwicklung des Produkts erfolgsversprechend?
2. Was müsste noch gemacht werden und was würde es kosten, um das Produkt auf den Markt zu bringen?

3.1 Weiterentwicklung

Zum jetzigen Zeitpunkt lässt sich noch keine genaue Aussage treffen, ob die Weiterentwicklung des Produkts erfolgsversprechend ist. Die Alpha-Version ist erst seit kurzem verfügbar. Diese wird jetzt von potenziellen Benutzer/-innen getestet. Anhand dieses Feedbacks lässt sich besser beurteilen, ob und wie erfolgsversprechend die Weiterentwicklung des geplanten Produkts ist. Gerade das Feedback kann den Product Owners bestätigen, ob ihre Idee gut ankommt oder nicht.

Zwei Dinge lassen sich jedoch bereits jetzt feststellen:

1. Die Product Owner haben das Bedürfnis eine App für die Vanlife-Community zu erstellen, da es eine App zur Knüpfung neuer Kontakte in der Vanlife-Community noch nicht gibt.
2. Die Kostenrechnung (siehe Einleitung Kapitel 1.4 Kosten) zeigt, dass es sich bei der App VanFan nicht um eine gewinnbringende App handelt. Mit den Kosten können gerade Unterhalt und Betrieb gedeckt werden.

Zusammengefasst bedeutet dies; es ist zu früh, um voreilig Schlüsse zu ziehen. Es gibt eine Community, welche nach einer einheitlichen App wünscht und die App würde genug Gewinn erbringen, um sich selbst aufrecht zu halten. Daher lohnt es sich, auf ein Feedback von den Alpha-Tester/-innen zu warten. Anhand von diesem lässt sich besser beurteilen, ob sich die App auf dem richtigen Weg befindet, oder ob es Anpassungen braucht.

3.2 Offene Aufgaben

Eine erste Version der App konnte erfolgreich veröffentlicht werden. Die von den Product Owners geforderten Funktionalitäten für einen ersten Release sind umgesetzt, oder befinden sich gerade in der Entwicklung. Das Userinterface sowie die API können noch weiter verbessert werden.

Bei der API beispielsweise wurden noch keine Tests bezüglich der Performance durchgeführt. Steigt die Anzahl Benutzer/-innen entstehen mehr Calls zur API. Die Antworten des Servers sollten bei hoher Auslastung nicht wesentlich länger dauern, wie aktuell bei geringer Auslastung. Da die Benutzer/-innen ansonsten darunter leiden. Im Verlauf der Alpha-Testphase wird sich zeigen, wie stark die bisherige Implementation des Servers ausgelastet ist und wo noch Verbesserungen vorgenommen werden müssen.

Beim Userinterface können die Ladezeiten, wenn man von einem Fenster um anderen wechselt, noch optimiert werden.

Sobald diese ersten Tests abgeschlossen werden konnten und die dadurch entdeckten Probleme behoben wurden, kann die App mit zusätzlichen Funktionalitäten erweitert werden. Da die Client- und Server-Architektur bewusst so geplant wurde, dass Erweiterungen einfach umzusetzen sind und keine Abhängigkeiten verursachen, ist der Aufwand, um eine neue Funktionalität einzubinden relativ gering und kann zeitgemäß umgesetzt werden.

Dieser Aufgabe werden sich Roger Wetter und Ramon Imper widmen, da sie im Herbstsemester 2022 an VanFan im Rahmen der Projektarbeit weiterarbeiten werden.

4 Literaturverzeichnis

- [1] M. Sterk, «www.tcs.ch,» Touring Club Schweiz, [Online]. Available: <https://www.tcs.ch/de/camping-reisen/camping-insider/ratgeber/reisevorbereitung/vanlife-im-trend.php>. [Zugriff am 27. 05. 2022].
- [2] Apple Inc., «<https://developer.apple.com/>,» Apple inc., 2022. [Online]. Available: <https://developer.apple.com/support/compare-memberships/>. [Zugriff am 27. 05. 2022].
- [3] Google LLC, «<https://support.google.com/googleplay/android-developer#topic=3450769>,» Google LLC, 2022. [Online]. Available: <https://support.google.com/googleplay/android-developer/answer/6112435?hl=de#zippy=%2Cschritt-registrieren-sie-sich-f%C3%BCr-ein-google-play-entwicklerkonto%2Cschritt-stimmen-sie-der-vertriebsvereinbarung-f%C3%BCr-entwickler-zu%2Cschritt-bezahlen-sie-die->. [Zugriff am 27. 05. 2022].

5 Abbildungsverzeichnis

Abbildung 1: Break-Even-Analyse mit Break-Even-Point nach fünf Jahren	3
Abbildung 2: VanFan-App – Registrierungsseite mit Anmeldeinformationen	5
Abbildung 3: VanFan-App – Login-Seite	5
Abbildung 4: VanFan-App – Ausschnitt des Feeds mit Kommentaren zu einem Post	5
Abbildung 5: VanFan-App – Karte mit eigenem Standort und Benutzer/-innen in der Nähe.....	6

6 Tabellenverzeichnis

Tabelle 1: Risikomatrix – Eintrittswahrscheinlichkeit eines Risikos im Verhältnis zum möglichen Schaden der entstehen könnte.	2
--	---