

Relaxoon

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für IT-Medientechnik

Eingereicht von:

Abdulrahman Al Sabagh
Moritz Eder

Betreuer:

Thomas Stütz

Projektpartner:

solvistas GmbH, Macolution GmbH

Leonding, April 2023

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

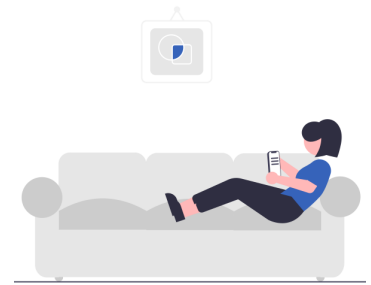
Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2023

Abdulrahman Al Sabagh & Moritz Eder

Abstract

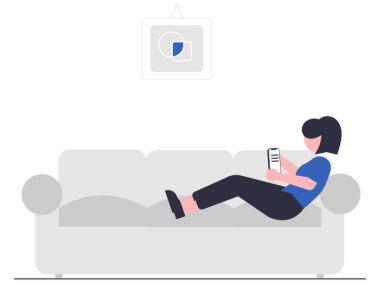
<Coming soon>



Zusammenfassung

Relaxoon ist ein Wortspiel aus den englischen Wörtern 'relax' und 'soon'.

Relaxoon bietet Entspannungsübungen in Form von Videos, Texten, Artikeln und noch mehr. Dabei ist ein sehr userfreundliches beziehungsweise simples User-Interface eins der größten Besonderheiten von Relaxoon.



Inhaltsverzeichnis

| | | |
|----------|---------------------------------------------|-----------|
| 1 | Ausgangssituation | 1 |
| 2 | Problemstellung | 2 |
| 3 | Ziele | 3 |
| 3.1 | Projektziele | 4 |
| 4 | Aufgabenstellung | 6 |
| 5 | Marktanalyse | 7 |
| 6 | Technologien | 8 |
| 6.1 | Strapi | 8 |
| 6.2 | Firebase App Distribution | 8 |
| 7 | Systemarchitektur | 9 |
| 7.1 | Komponentendiagramm | 9 |
| 8 | Entwurfsentscheidungen | 11 |
| 8.1 | React Native | 11 |
| 8.2 | Strapi | 13 |
| 8.3 | Component Library | 15 |
| 9 | Implementierung | 16 |
| 9.1 | Entity Relationship Diagram (ERD) | 16 |
| 9.2 | Medias und Articles | 16 |
| 9.3 | TutorialPage | 17 |
| 9.4 | Suche und Filterungen | 17 |
| 9.5 | State Management | 19 |
| 9.6 | Dark/Light Mode | 19 |
| 9.7 | Help und Info Screen | 20 |

| | |
|------------------------------------------------------------------------|-------------|
| 9.8 Authentifizierung | 20 |
| 9.9 Validierung | 22 |
| 9.10 Deployment | 22 |
| 9.11 Hochladen einer Mobileapp auf Firebase App Distribution | 28 |
| 10 Ausgewählte Aspekte und Probleme | 32 |
| 10.1 Probleme mit localhost und https | 32 |
| 10.2 Inkompatible Libraries beim Build-Prozess | 32 |
| 10.3 Probleme mit Thumbnails | 33 |
| 10.4 Dauer von Videos | 34 |
| 10.5 Probleme mit useEffect und React-navigation Library | 34 |
| 11 Resümee | V |
| Literaturverzeichnis | VI |
| Abbildungsverzeichnis | VIII |
| Tabellenverzeichnis | IX |
| Quellcodeverzeichnis | X |
| Anhang | XI |

1 Ausgangssituation

Viele Menschen empfinden es als angenehm oder entspannen Musik zu hören. Bei dem heutzutage immer größer werdenden Stress verliert man manchmal den Überblick über seine Aufgaben. Deshalb greifen viele Menschen auf die Musik zurück, wodurch sie sich beim Arbeiten entweder besser konzentrieren oder kurz eine Pause einlegen können.

Mit Stress und Leistungsdruck umgehen zu können, ist nicht nur für Erwachsene kein leichtes Unterfangen, sondern auch speziell für Schülerinnen und Schüler oft schwierig. Es ist wichtig, damit umgehen zu lernen, um nicht in ein Burn-Out zu fallen. Ein Mensch braucht Ruhe und Entspannung um auch auf lange Zeit gut funktionieren und produktiv lernen oder arbeiten zu können.

Zur Stressbewältigung wurde von der MACOLUTION GmbH und der solvistas Group eine Idee zur Lösung dieses Problems entwickelt. Die MACOLUTION GmbH ist ein Unternehmen, welches es sich zur Aufgabe gemacht hat, Management- und Coaching-Solutions mit modernsten Technologien und bewährten Methoden zu entwickeln. [1] Mit Relaxoon soll diese Idee Wirklichkeit werden.



Abbildung 1: Logo MACOLUTION



Abbildung 2: Logo solvistas

2 Problemstellung

Die heutige Welt und der Alltag werden immer schneller und komplexer, was für viele Menschen einen Stress verursacht.

- Gesundheit ist ein wichtiger Faktor.
- Oft fehlt die Möglichkeit sich zu entspannen.
- Viele Menschen leiden darunter.
- Die Betroffenen verlieren den Mut sich Hilfe zu holen.

Stress bewirkt bei Menschen oft nicht nur enormen Leistungsdruck, sondern er kann auch zu gesundheitlichen Folgen führen. Mögliche Folgen von hohem Stress oder hohem Leistungsdruck könnten sein:

- Zeichen von Nervosität
- Verspannungen, die zu Kopf-, Genick- und Rückenschmerzen führen können
- Vergesslichkeit
- Depression
- psychische Störungen

Anhaltender Stress kann sogar Herz/Kreislauf- und Nierenerkrankungen, Stoffwechselstörungen, Allergien oder Entzündungskrankheiten hervorrufen. [2] Daher benötigen gestresste Menschen eine Möglichkeit sich wieder entspannen zu können.

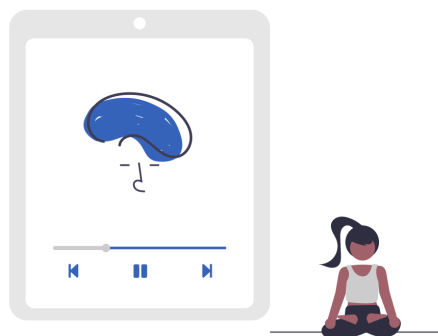


Abbildung 3: Musik dient zur Entspannung

3 Ziele

Es gibt einen klaren Unterschied zwischen dem Setzen von Zielen und dem tatsächlichen Erreichen von Zielen. Das einfache Festlegen von Zielen allein genügt nicht. Um Erfolg in einem bestimmten Vorhaben zu haben, ist es notwendig, klare Ziele zu definieren. Ein Ziel repräsentiert einen Zustand, ein Ergebnis oder einen bestimmten Ort. Der Grad und die Art der Zielerreichung sind entscheidend für die Definition von Erfolg. Daher ist das Festlegen eines Ziels lediglich der Ausgangspunkt. Dafür müssen Fortschritte oder Meilensteine erreicht und der vorgegebene Zeitrahmen eingehalten werden – und dies alles unter Verwendung legaler und legitimer Mittel und Methoden. Oftmals scheitert man nicht an mangelnder Disziplin oder Motivation, sondern an falsch formulierten oder unpassenden Zielen.

Zielen charakterisieren sich durch einfache Aufgaben, die nach der Reihe erledigt werden – sie repräsentieren feste Absichten. Hinter jedem Ziel steht immer ein konkretes Bestreben. Ziele sind nicht nur das Ergebnis rationaler Überlegungen, sondern vor allem eine Angelegenheit des Herzens und der Motivation.[3]



Abbildung 4

Ziele zu setzen ist wichtig, deswegen müssen sie von Anfang an **richtig** gesetzt werden. Beim Setzen der Ziele kann man sich an folgenden Punkten orientieren: [3]

- Klare Konkretisierung ist entscheidend für Ziele! Je präziser die Beschreibung ist, desto einfacher ist es, darauf hinzuarbeiten.
- Ziele müssen realistisch sein! Zu ehrgeizige Ziele können Frustration und Selbstzweifel auslösen. Sowohl das angestrebte Ergebnis als auch die dafür vorgesehene Zeit sollten der Realität entsprechen.
- Wenn Ziele öffentlich gemacht werden, erhöht das auch die Erfolgswahrscheinlichkeit. Der Austausch über Ziele mit Familie, Freunden oder Kollegen steigert die Motivation und erzeugt Verbindlichkeit, da man es anderen beweisen möchte.
- Immer positiv bleiben! Positive Formulierungen der Ziele motivieren langfristig mehr.
- Ein eigener Antrieb ist notwendig, um die Zielabsicht vor Augen zu haben und um dranbleiben zu können.
- Ziele erfordern Flexibilität. Der Weg zum Ziel kann sich ändern, ebenso die Zeit für das Erreichen eines Meilensteins oder sogar das Ziel selbst. Ziele sind dynamisch und nie statisch oder unveränderlich.
- Zur Visualisierung der Ziele sollten sie stets in der Gegenwart formuliert werden, damit man sich gleich Bilder im Kopf vorstellen kann. Das steigert ebenso die Motivation.

3.1 Projektziele

Das Projekt soll zeigen, dass unter Verwendung des Open-Source headless CMS 'Strapi' und des mobilen React-Native Frontends einfach eine App zur Wiedergabe von Mediendateien für verschiedene mobile Betriebssysteme erstellt werden kann. Wichtig ist dabei:

- Die Feinabstimmung der Qualität der Inhalte
- Die optimale Bereitstellung der sorgfältig ausgewählten Inhalte
- Ein ansprechendes und benutzerfreundliches Design für den passenden Look and Feel der App.
- Stressreduzierung, Schlafverbesserung und eine generelle Förderung der mentalen Gesundheit bei Personen, die gestresst sind oder unter Leistungsdruck stehen.

Um das formulierte Ziel in gewünschter Qualität und fristgerecht zu erreichen, mussten neben Schul- bzw. Lernanforderungen die Ausdauer und Priorität auf das Projekt gesetzt werden.

4 Aufgabenstellung

Personen, die oft gestresst vom Alltag sind, sollen durch Relaxoon wieder runterkommen können und sich entspannen können, damit man sich wieder mit mehr Energie und einem klaren Kopf den nächsten Aufgaben stellen kann.

5 Marktanalyse

<Coming soon>

6 Technologien

6.1 Strapi

Strapi ist ein Headless Content Managment System (CMS), welches eine vorgefertigte Benutzeroberfläche für die Content-Creators und auch für die Entwickler bereitstellt.

Mit der Verwendung davon sind Inhalte und dazu gebrauchte technische Funktionalitäten, wie zum Beispiel

- REpresentational State Transfer (REST)- bzw. Graph Query Language (Graphql)-Schnittstellen
- Logik für die Authentifizierung
- Create, Read, Update und Delete (CRUD) Funktionalitäten jeder Business Entität

sehr einfach erstellbar. [4]

6.2 Firebase App Distribution

“ Firebase App Distribution macht die Verteilung Ihrer Apps an vertrauenswürdige Tester problemlos. Indem Sie Ihre Apps schnell auf die Geräte der Tester übertragen, können Sie frühzeitig und häufig Feedback einholen. Und wenn Sie Crashlytics in Ihren Apps verwenden, erhalten Sie automatisch Stabilitätsmetriken für alle Ihre Builds, sodass Sie wissen, wann Sie zur Auslieferung bereit sind. ”[5]

Der Vorteil von Firebase App Distribution ist, dass man die Applikation auf dem eigenen Mobilegerät ausprobieren kann, ohne die App auf dem Play Store bzw. App Store hochzuladen.

7 Systemarchitektur

7.1 Komponentendiagramm

Für eine klare funktionale Übersicht auf der vorliegenden Diplomarbeit wurde ein sogenanntes Komponentendiagramm für das technische System von Relaxoon erstellt. Da Relaxoon eine Applikation ist, die auf Mobilgeräten laufen soll, wurde für die Entwicklung das cross-plattform JavaScript (js) Framework "React Native" eingesetzt. Damit der Kunde Inhalte in die App hinzufügen kann, wurde ein "Node js" basiertes Headless CMS namens "Strapi" verwendet.

Für die Kommunikation zwischen Frontend und Backend wird REST verwendet. Für die Persistierungsebene hat sich das Team für "PostgreSQL" entschieden.

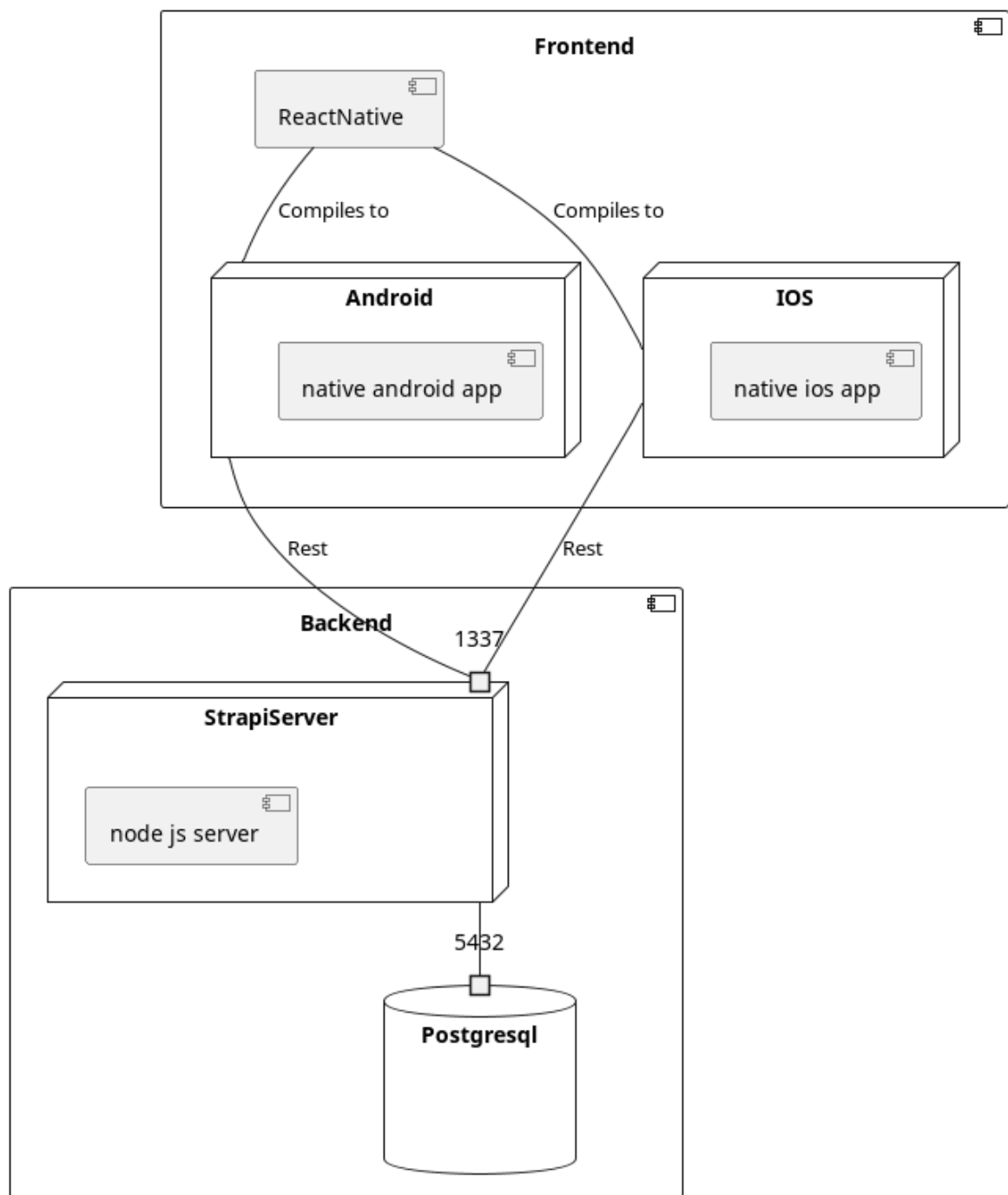


Abbildung 5: Systemarchitektur

8 Entwurfsentscheidungen

8.1 React Native

Grundsätzlich wurde React Native aus mehreren Gründen verwendet:

- React Native ist das Standardframework für cross-plattform-Lösungen bei der Firma Solvistas.
- Vorhandene Kenntnisse in den Sprachen Javascript bzw. Typescript
- Auftraggeber forderte eine cross-plattform-Lösung für das Frontend, um IOS-Android- Betriebssysteme abzudecken
- Endergebnis von React Native ist eine native App

8.1.1 Was ist eine cross-plattform-Lösung

“ Eine Cross-Platform App besteht aus einem einzigen Code, der jeweils in die native Systemsprache von Apple, Android & Co. kompiliert wird. Dadurch erhält man eine App, die mit wenig Entwicklungsaufwand auf mehreren Betriebssystemen zur Verfügung steht, sich aber dennoch wie eine native App anfühlt.” [6]

8.1.2 Warum wurde eine cross-plattform-Lösung angefordert

Der Client von Relaxoon besteht aus einer einfachen Applikation, die keine Business Logik und auch keine Performance kritische Funktionalitäten hat. Daher ist die Verwendung einer cross-plattform-Lösung sehr vorteilhaft, da alle Plattformen von einem Codebase gepflegt werden können.

8.1.3 Warum ist Relaxoon native und nicht hybrid

hybride Applikationen:

“ Eine hybride App kombiniert die besten Elemente von [N]ativen und Web-Apps. Sie werden wie eine native App installiert, aber es ist eigentlich eine Web-App innerhalb des Endgeräts. Hybride Apps werden in den gängigsten Sprachen für die Web-App Entwicklung, wie z.B. HTML und CSS, programmiert. Dies bedeutet, dass sie auf verschiedenen Plattformen verwendet werden können. Obwohl sie in der Sprache der Webanwendung entwickelt wurden, haben sie die gleiche Fähigkeit wie native Apps, sich an verschiedene Geräte, wie ein Tablet, Smartphone usw. anzupassen. ” [7]

native Applikationen:

“ Eine native App ist eine Anwendung, die entwickelt wurde, um auf einer bestimmten Plattform oder einem bestimmten Endgerät zu arbeiten. Aus diesem Grund können native Apps mit den auf der jeweiligen Plattform installierten Betriebssystem[s]funktionen interagieren und diese nutzen. ”
[7]

Grundsätzlich sind native Applikationen viel schneller und barrierefreier als hybride Applikationen [7]

8.1.4 Alternativen für React Native

Kotlin Multiplatform

Kotlin MutliPlatform (KMP) bietet bessere Performance als React Native und hat auch eine modulare Integration.

modulare Integration:

“ Probably the biggest benefit in favor of Kotlin Multiplatform is that it's an SDK and not a framework. This means that teams with existing apps can simply add a module or migrate a small part to assess its viability without a huge commitment. This really helps Kotlin address the biggest deterrent when moving to a new codebase. ” [8]

Allerdings ist das Problem davon, dass KMP noch immer im Beta ist und daher ist diese nicht stabil. Außerdem ist die Community von dieser Alternative nicht so groß wie die Community von React Native. Gute Kenntnisse in Swift **U**ser **I**nterface (UI) und Android **S**oftware **D**evelopment **K**it (SDK) werden für die Verwendung von Kotlin Multiplatform auch verlangt. Grund dafür ist, dass KMP keine Bilbiotheken für UI-Elementen wie React Native oder Flutter bereitstellt, sondern die Syntax der OS-Technologie verwendet. [8]

Flutter

Flutter an sich ist viel schneller als React Native. Außerdem unterstützt Flutter die Betriebssysteme Windows, Linux und MacOS zusätzlich zu Web, Android und IOS. [9] Das Problem bei Flutter ist, dass man gute Kenntnisse in der Programmiersprache "Dart" haben muss. [9] Für die Entwicklung von React Native sind gute Kenntnisse in den Programmiersprachen Javascript und Typescript, die wir bereits gelernt haben, gebraucht. Außerdem ist die Unterstützung von MacOS, Linux und Windows für uns gar nicht relevant.

Xamarin

Xamarin ist eine C# Framework, die für die Entwicklung von nativen cross-plattform-Lösungen zuständig ist. Die Performance von Xamarin ist viel besser als die Performance von React Native. [10]

Die Entwickler haben sich aus den Gründen, die bei Flutter bereits erwähnt wurden, für React Native entschieden. Außerdem gibt es beim Xamarin kein "hot reloading" und daher muss das Program bei jeder kleinen Änderung neugestartet werden.[10]

8.1.5 Nutzwertanalyse für das Frontend-Framework

| Kriterien | React Native | Xamarin | Flutter | KMP |
|--------------------------------|--------------|---------|---------|-----|
| Vorhandene Kenntnisse max. 50% | 50% | 0% | 0% | 10% |
| Barrierefreiheit max. 30% | 15% | 5% | 30% | 20% |
| Performance max. 20% | 5% | 20% | 15% | 10% |
| Summe | 70% | 25% | 45% | 40% |

8.2 Strapi

8.2.1 Was ist ein CMS

“Ein Content-Management-System (CMS) ist eine Softwareanwendung, die es Benutzern ermöglicht, digitale Inhalte zu erstellen, zu bearbeiten, gemeinsam zu editieren, zu veröffentlichen und zu speichern. Content-Management-Systeme werden typischerweise für Enterprise Content Management (ECM) und Web Content Management (WCM) eingesetzt.” [11]

8.2.2 Warum wurde ein CMS verwendet

Der Auftraggeber wollte die Applikation so schnell wie möglich veröffentlichen, da die Anzahl der Apps, welche die gleichen Anwendungsfälle wie Relaxoon haben, nicht so groß ist. Daher ist die Verwendung eines fertigen CMSes viel schneller als die Implementierung eines Backends.

8.2.3 Was ist ein Headless-CMS

“ Ein Headless CMS ist sowohl eine Weiterentwicklung als auch eine Verknappung eines klassischen CMS. Dem System werden integrale Bestandteile genommen, um es für unterschiedlichste Ausgaben kompatibel zu machen. Das gelingt dadurch, dass Frontend und Backend in einem Headless CMS nicht mehr monolithisch miteinander verknüpft sind. Das fehlende Frontend ist auch der Grund, wieso derartige CMS-Systeme als „kopflös“ (englisch: „headless“) bezeichnet werden. ” [12]

8.2.4 Warum wurde ein Headless-CMS verwendet

- Vorgabe von dem Auftraggeber
- Andere Arten von CMSes generieren statische HTML Seiten, welche für Relaxoon gar nicht gebraucht werden, da Relaxoon eine Mobileapp ist
- die Firma Solvistas beschäftigt sich intensiv mit Data Science. Da man auf dem CMS mittels REST zugreifen kann, will Solvistas ein Datenzentrum aus diesem in der Zukunft erstellen, zusätzlich zur Verwendung von CMS für Data Science Zwecke.

8.2.5 Warum Strapi und nicht Wordpress APplication Interface (API)

Vorteile von Wordpress API

- Es gibt sehr viele Einstellungen und Features
- Unterstützt Search Engine Optimization (SEO)
- guter Community Support
- Einfach zu verwenden

[4]

Nachteile von Wordpress API

- Limitierte Flexibilität, da viele Plugins und Addons kostenpflichtig sind. die kostenlose Verwendung von einigen Addons und Plugins ist limitiert.
- Ist nicht geeignet für eine Software mit großer Skalierung
- Wordpress wird von vielen Menschen verwendet und deshalb ist es für die Hacker nicht irrelevant
- Die Benutzeroberfläche davon ist nicht gut designt und deshalb ist die Verwendung davon meistens sehr verwirrend und unangenehm zu bedienen

[4]

Vorteile von Strapi

Im Gegensatz zu Wordpress, kann man mit Strapi beliebig viele Plugins und Addons verwenden. Außerdem gibt es eine eingebaute Authentifizierung bzw. Autorisierungsfunktion zusätzlich zur Unterstützung von 20 unterschiedlichen Sprachen und REST bzw. GraphQL APIS. [4]

Nachteile von Strapi

Ein Grundwissen in der Programmierung ist erforderlich, wenn man sich für Strapi entscheidet. Außerdem ist es nicht so weit verbreitet wie Wordpress und somit ist die Anzahl der 3rd-party-Libraries, die man in Strapi einbetten kann, nicht so groß. [4]

Ergebnis

Alle erwähnten Pros und Contras zeigen, dass die User EXperience (UX) und die Skalierbarkeit von Strapi viel besser als Wordpress ist. Außerdem ist Strapi viel schneller als Wordpress, da Strapi in "Node js" geschrieben ist und Wordpress in "PHP"[4]

8.2.6 Nutzwertanalyse für das Headless-CMS

| Kriterien | Strapi | Wordpress API |
|---------------------------------|--------|---------------|
| Security max.30% | 30% | 15% |
| Anpassbar max. 30% | 30% | 15% |
| Community Sup- port max. 30% | 10% | 25% |
| Performance max. 10% | 10% | 5% |
| Summe | 80% | 65% |

8.3 Component Library

Der Standardweg, um UI-Elementen in React Native zu stylen, ist JSS. Diese ist eine Technologie, wo man **C**ascading **S**tyle **S**heet (CSS) in Form von Javascript Objekten schreibt. Das Problem bei dieser Technologie liegt dabei, dass viele CSS Attribute und Funktionen nicht dabei inkludiert sind. Es gibt auch außerdem spezifische OS-Einstellungen für Barrierfreiheit wie zum Beispiel die Einstellungen für Screenreaders, die man bei der Entwicklung immer wieder vergessen kann. Aus diesen Gründen hat sich das Team für die Verwendung von 'Native Base' entschieden.

9 Implementierung

9.1 Entity Relationship Diagram (ERD)

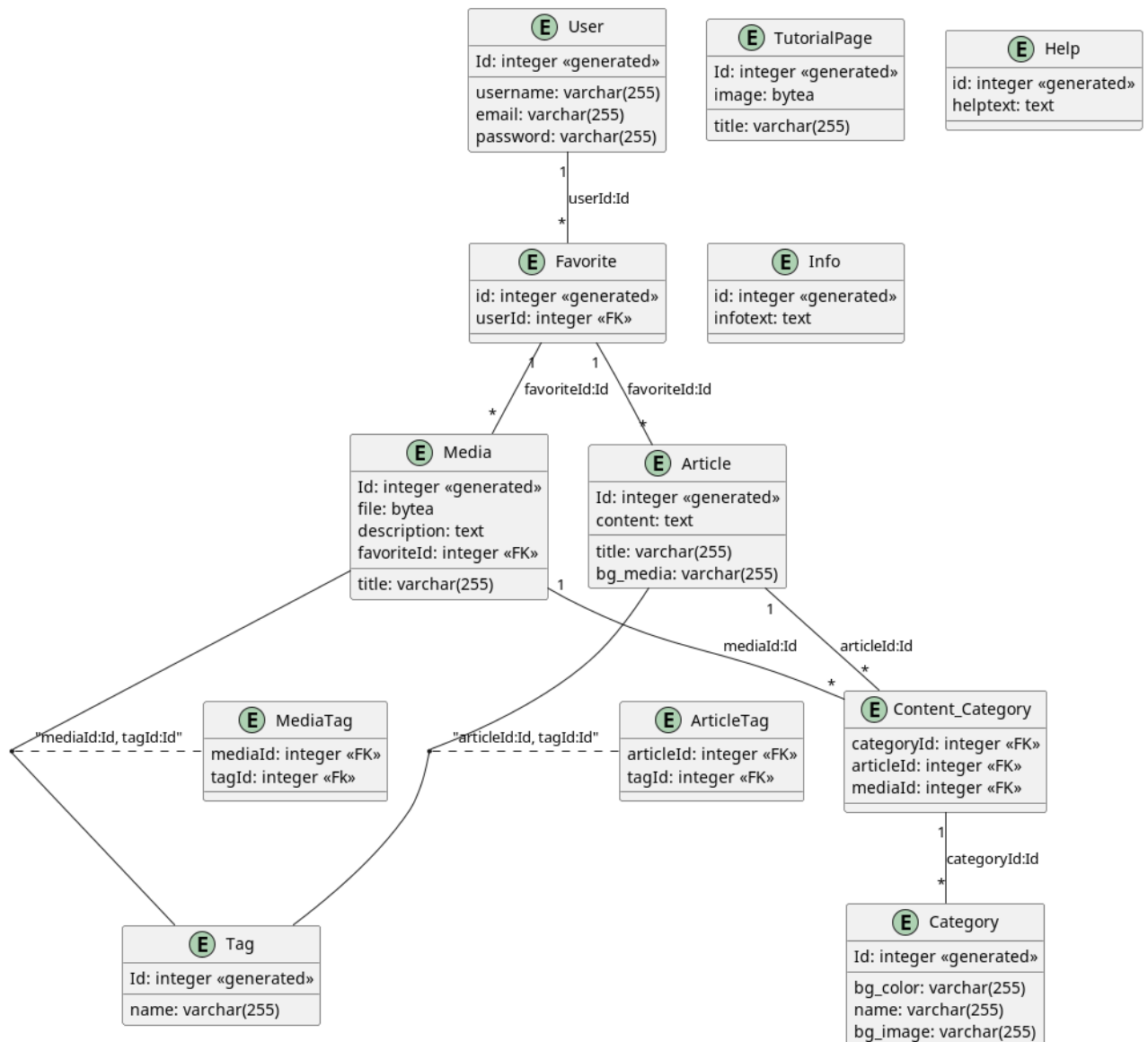


Abbildung 6: ERD

9.2 Medias und Articles

Grundsätzlich haben wir als Team entschieden, die Artikeln und die Medien in zwei speraten Tabellen zu speichern, da Artikeln mehrere Bilder, Videos und Audios Inhalte

enthalten können. Diesen können dann mittels ein "Rich Text Editor" hinzugefügt werden. Bei "Media" handelt es sich nur um ein Medienelement, nämlich ein Bild, Video oder Audio File. Es ist aber wichtig anzumerken, dass keine Benutzeroberfläche für die Artikeln implementiert wurde, da der Kunde diese nicht in dem Minimal Viable Product (MVP) haben wollte. Für die Relaisierung des kompletten Datenmodells war aber das Einfügen der Artikel erforderlich.

9.3 TutorialPage

Die Entität "TutorialPage" hat keine Beziehungen zu anderen Entitäten, da sie nur für den Inhalt des Tutorial-Slideshows, die nach der erfolgreichen Installation der App angezeigt wird, gedacht ist.

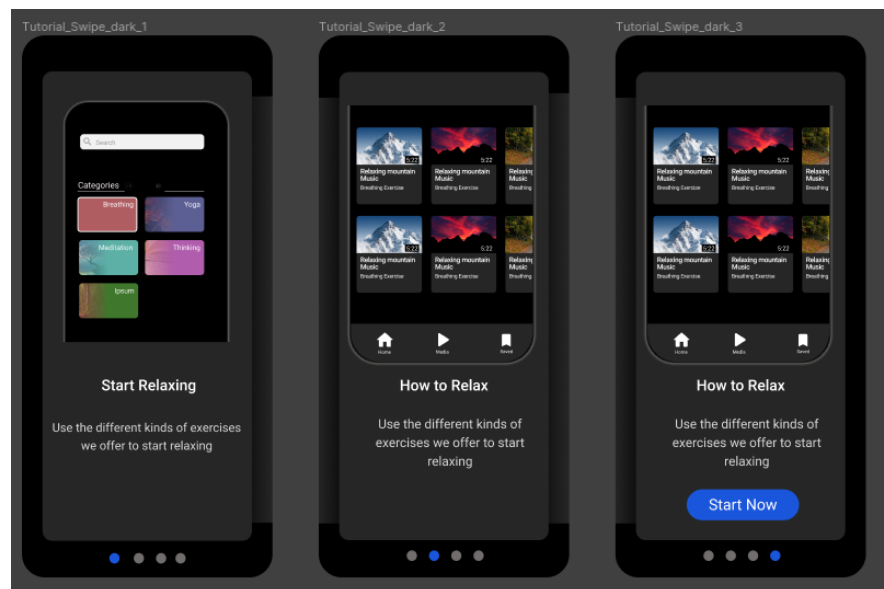


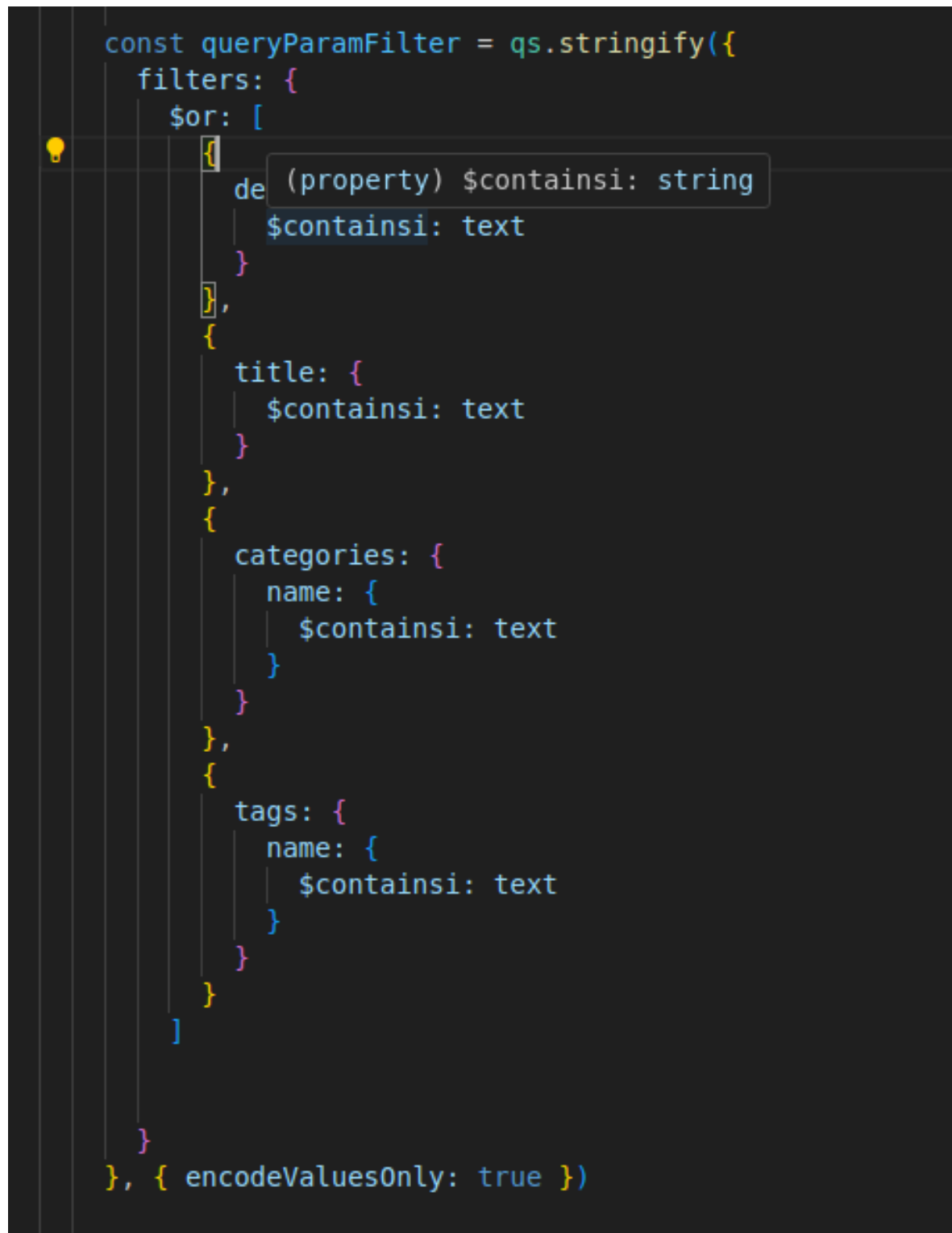
Abbildung 7: Screenshot aus dem UI Prototyp von Relaxoon

Mit der Persestierung der Inhalte von der Slideshow könnte die Erstellung von neuen Builds bei jeder Änderung vermieden werden.

9.4 Suche und Filterungen

Bei den Filterungen wurde der "Interactive Query Builder" verwendet, um die Filterparameter der Abfragen zu generieren und diese dann bei den REST Abfragen anzuhängen. Es gibt auch eine "Query Engine", welche die gleiche Funktion wie diese "Interactive Query Builder" hat. Diese kann in dem **Object Relational Mapper (ORM)** angewendet

werden. Mit der Verwendung dieser "Interactive Query Builder" kann man die Daten, die man von den REST-Ressourcen bekommt, anpassen. Es gibt auch die Möglichkeit, die REST-Ressource im Backend mittels dieser "Query Engine" anzupassen. Allerdings wurde diese "Query Engine" nicht verwendet, da die Dokumentation bzw. Intellisense von dem ORM sehr ungenau war.



```
const queryParamsFilter = qs.stringify({
  filters: {
    $or: [
      {
        (property) $containsi: string
        $containsi: text
      },
      {
        title: {
          $containsi: text
        }
      },
      {
        categories: {
          name: {
            $containsi: text
          }
        }
      },
      {
        tags: {
          name: {
            $containsi: text
          }
        }
      }
    ]
  }
}, { encodeValuesOnly: true })
```

Abbildung 8: Beispiel für den Interactive Query Builder

API-Route mit den angehängten generierten Abfrageparameter:

`/medias?&populate[tags]=true &populate[file]=true &populate[favorite][populate]=users_permissions_user`

9.5 State Management

“ Der Begriff State ist in React-Applikationen überall präsent. Generell bezeichnet State den Zustand einer Komponente, also die dynamischen Daten, die eine Komponente den Benutzer:innen anzeigt. Eine Änderung am State führt dazu, dass die Komponente neu gerendert wird und so die Datenänderung für die Benutzer:in sichtbar wird. Im einfachsten Fall verwaltet jede Komponente ihren eigenen State. Es gibt Fälle, in denen es jedoch erforderlich wird, den State zwischen mehreren Komponenten zu teilen. ” [13]

Um die Zustände der Applikation zu behandeln wurde meistens mit dem ‘useState Hook‘ von React gearbeitet. Der Einsatz einer State Management Library wie zum Beispiel RXJS oder Redux war gar nicht nötig, da die Anzahl der verwendeten UI-Komponenten nicht so groß ist.

9.5.1 useState Hook

“useState is a React Hook that lets you add a state variable to your component.” [14]

9.6 Dark/Light Mode

Da das Team sich für die Verwendung von der UILibrary ‘Native Base‘ entschieden hat, war die Realisierung von diesem Feature sehr einfach. Die Komponenten von ‘Native Base‘ haben die Properties `_light` und `_dark`, wo man die Farbe einer Komponente je nach Modus definieren kann. Native Base verwendet im Hintergrund einen useContext Hook, wo die Modi der Applikation gespeichert werden.

9.6.1 useContext Hook

“ useContext is a React hook that provides a way to share data (context) across multiple components without explicitly passing it through props. It is part of the React Context API, which is built into the React library. ” [15]

9.7 Help und Info Screen

Mit der Speicherung der Inhalten von den Ansichten "Help" und "Info" ist neues Build für das Frontend nicht mehr nötig. Somit kann der Content-Manager mehrere Freiheiten haben. Diese Inhalte wurden mit sogenannten "Single Types" persestiert. Ein "Single Type" ist nichts anders als eine Tabelle, die nur eine Zeile enthält. Bei einer Änderung des Werts von diesem "Single Type" wird diese eine Zeile aktualisiert.

9.8 Authentifizierung

Für die Authentifizierung wurde mit Json Web Token (JWT) gearbeitet. Die Authentifizierungslogik im Server ist im Strapi eingebaut. Für den Client wurde die Logik so umgesetzt, dass der User die App erst verwenden kann, wenn er bereits authentifiziert ist. Die Ansicht fürs Login soll aber nicht angezeigt werden, wenn der User bereits authentifiziert ist. Um zu überprüfen, ob der Nutzer noch eingeloggt ist oder nicht wurde das Token im AsyncStorage gespeichert. Dieser ist ähnlich zu dem localStorage in der Webentwicklung. Somit wurde bei der Öffnung der App geprüft, ob das Token valid und nicht abgelaufen ist. Damit der User auf die anderen Screens nicht zugreifen kann, wenn er nicht eingeloggt ist, wurde eine boolische Zustandsvariable verwendet, welche das LoginScreen als die zweite Ebene unserer Stack-Navigation rendert, wenn das Token invalid oder abgelaufen ist. Die erste Ebene ist für den TutorialScreen reserviert. Im folgenden Codestück ist die ganze Logik der Authentifizierung zu finden:

Listing 1: protected screens

```

1  import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
2  import { createStackNavigator } from '@react-navigation/stack';
3  import { useColorMode } from 'native-base';
4  import * as React from 'react';
5  import { useEffect, useState } from 'react';
6  import { jwtDecode } from 'jwt-decode';
7  import CategoryScreen from '../screens/CategoryScreen';
8  import HomeScreen from '../screens/HomeScreen';
9  import MediaScreen from '../screens/MediaScreen';
10 import SavedScreen from '../screens/SavedScreen';
11 import SettingsScreen from '../screens/SettingsScreen';

```

```

12 import VideoScreen from './screens/VideoScreen';
13 import WaitScreen from './screens/WaitScreen';
14 import HelpScreen from './screens/setting tabs/HelpScreen';
15 import InfoScreen from './screens/setting tabs/InfoScreen';
16 import { getToken } from './util/getUser';
17 import TabBar from './components/TabBar';
18 import TutorialScreen from './screens/intro/TutorialScreen';
19 import LoginScreen from './screens/registration/LoginScreen';
20 import RegistrationForm from './screens/registration/RegistrationForm';
21 import { colors } from './styles/ThemeColors';
22 import AsyncStorage from '@react-native-async-storage/async-storage';
23 // Screen names
24 type PageSettings = Record<string, { iconName: string; screen:
    React.ComponentType<any> }>;
25 const settingsForScreenName: PageSettings = {
26   Home: { iconName: 'home', screen: HomeScreen },
27   Media: { iconName: 'play', screen: CategoryScreen },
28   Saved: { iconName: 'list', screen: SavedScreen }
29 };
30
31 const Tab = createBottomTabNavigator();
32 const Stack = createStackNavigator();
33
34 function MainContainer(): JSX.Element {
35   const { colorMode } = useColorMode();
36   const [tokenExists, setTokenExists] = useState<boolean>(false);
37   useEffect(() => {
38     getToken()
39       .then((data) => {
40         console.info(typeof data);
41         if (!data) throw new Error('no token');
42         const decodedToken: { iat: number; exp: number; id: number } =
43           jwtDecode(data);
44         const eighthoursAfterNow = new Date(new Date().getHours() +
45           8).getMilliseconds();
46         if (decodedToken.exp < eighthoursAfterNow) {
47           setTokenExists(false);
48           AsyncStorage.clear();
49           return;
50         }
51         setTokenExists(true);
52       })
53       .catch((e) => {
54         console.info('no token', e);
55         setTokenExists(false);
56       });
57     console.info(tokenExists);
58   }, []);
59
60   return (
61     <Stack.Navigator
62       screenOptions={{
63         headerShown: false
64       }}
65     >
66     /*
67     <Stack.Screen name="WaitScreen" component={WaitScreen}/>
68     */
69     /*
70     <Stack.Screen name="Tutorial"
71       component={TutorialScreen}/>
72     */
73     <Stack.Screen
74       name="Main"
75       options={{
76         headerShown: false
77       }}
78     >
79     {() => (
80       <Tab.Navigator
81         screenOptions={{
82           headerShown: false,
83           headerTitleStyle: {
84             fontSize: 16,
85             color: colorMode === 'dark' ? colors.fontDark : colors.fontLight
86           }
87         }}
88         tabBar={((props) => <TabBar color={colorMode} {...props} />)}
89       </Tab.Navigator>
90     )}
91   )

```

```

89         {Object.entries(settingsForScreenName).map(([pageName, settings]) =>
90             (
91                 <Tab.Screen key={pageName} name={pageName}
92                     component={settings.screen} />
93             )
94         )}
95     </Tab.Navigator>
96 )}
97 </Stack.Screen>
98 <Stack.Screen name="Content" component={MediaScreen} />
99 <Stack.Screen component={VideoScreen} name="Video" />
100 <Stack.Screen component={SettingsScreen} name="Settings" />
101 <Stack.Screen name="Info" component={InfoScreen} />
102 <Stack.Screen name="Help" component={HelpScreen} />
103 </Stack.Navigator>
104 );
105 }
106
107 export default MainContainer;

```

9.9 Validierung

Strapi an sich hat eine eingebaute Validierung für Emails und Passwörter. Das Team hat aber eine client-seitige Validierung implementiert, um die Anzahl der Requests zu minimieren und eine bessere UX zu schaffen. Für die client-seitige Validierung wurde eine Library namens 'Zod' verwendet. Diese verwendet Builder Pattern um ein Validierungsschema zu bauen.

Listing 2: Validierungsschemen

```

1  import { z } from 'zod';
2
3  export const nameValidator = z.string().min(3);
4  export const emailValidator = z.string().email();
5  export const passwordValidator = z.string().min(6);

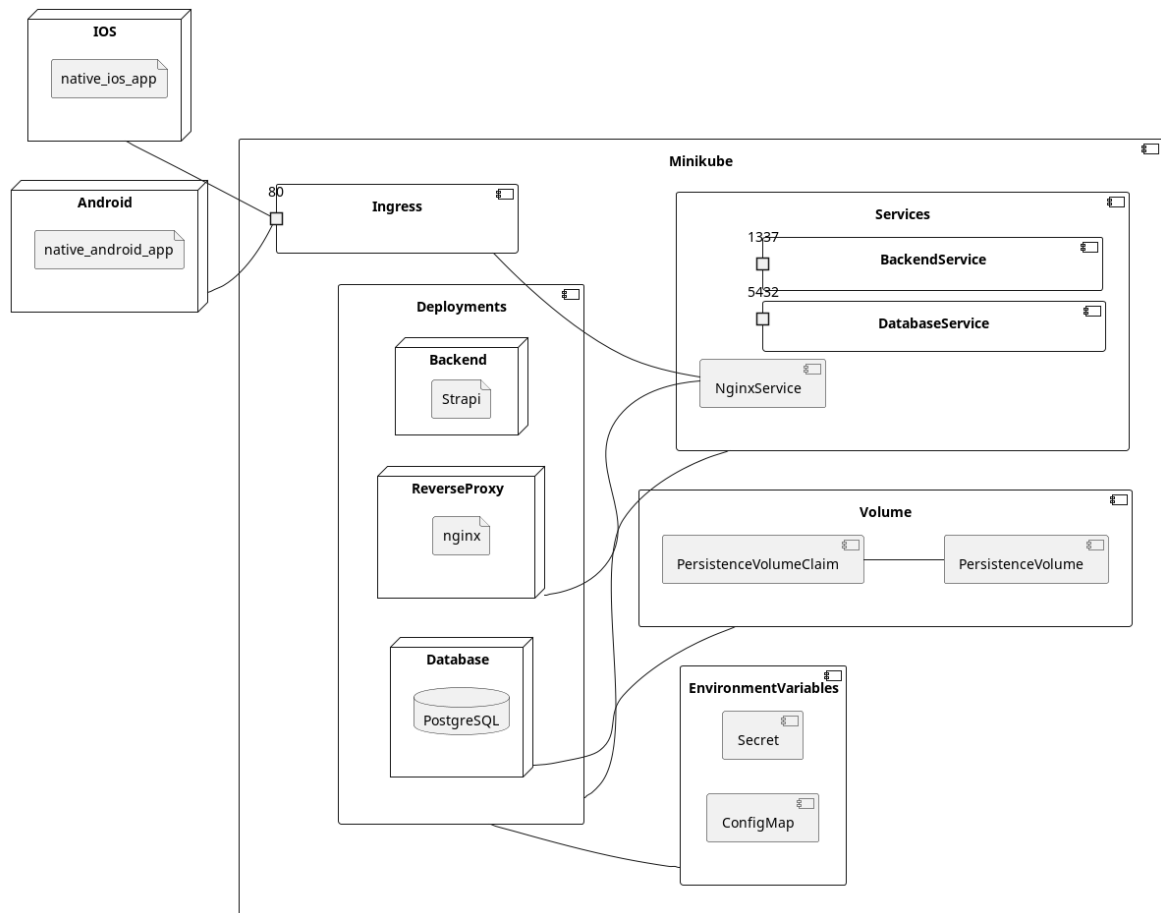
```

9.10 Deployment

9.10.1 Allgemeins

Die Veröffentlichung der Applikation auf den Play Stores wurde von dem Auftraggeber nicht in die Frage gestellt. Das Hochladen der Android-Applikation auf Firebase App Distribution wurde aber von dem Team verlangt. Das Backend wurde auf den Servers der Firmen Solvistas und Macolution deployt. Für Demonstrationszwecke wurde ein lokales Deployment auf Minikube erstellt

9.10.2 Deployment Diagram



9.10.3 Deployment von dem Backend auf Minikube

Damit ein funktionsfähiges Deployment erstellt werden kann sind solche Komponenten zu deployen:

- Strapi
- Datenbank
- Reverse Proxy (Nginx)

Für jede dieser genannten Bauteile des Backends sind K8S- (Kubernetes) Deployments und Services zu erstellen.

9.10.4 K8S-Konfigurationen für die Datenbank

Zusätzlich zu einer Deployment- und Service-Komponente ist für die Sicherung von den Credentials der Datenbank eine sogenannte Secret-Komponente gebraucht. Für die Vermeidung von Datenverlusten in Fällen wie Neustart eines Pods oder die Aktualisierung

der Konfiguration von dem Pod ist das Anlegen einer PersistenceVolume-Komponente sehr dringend. Damit das Deployment auf die PersistenceVolume-Komponente zugreifen kann, ist eine sogenannte PersistenceVolumeClaim-Komponente zu definieren.

Für die Konfiguration der Secret-, Service- und Deployment-Komponente wurde die `kubectl` Command Line Interface verwendet. Die PersistenceVolume-Komponente bzw. die PersistenceVolumeClaim-Komponente wurden fertige Konfigurationen genommen.

Listing 3: K8S PVC

```

1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    finalizers:
5      - kubernetes.io/pvc-protection
6    name: strapi-pvc
7    namespace: default
8  spec:
9    accessModes:
10     - ReadWriteMany
11    resources:
12     requests:
13       storage: 10Mi
14     storageClassName: standard

```

Listing 4: K8S PV

```

1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    finalizers:
5      - kubernetes.io/pv-protection
6    labels:
7      type: local
8    name: strapi-volume
9    resourceVersion: "33077"
10   uid: ae6d772a-0090-4074-b3ac-1edb929daf29
11  spec:
12    accessModes:
13     - ReadWriteOnce
14    capacity:
15     storage: 10Gi
16    hostPath:
17     path: /mnt/data
18     type: ""
19    persistentVolumeReclaimPolicy: Retain
20    storageClassName: manual
21    volumeMode: Filesystem
22  status:
23    phase: Available

```

9.10.5 K8S-Konfigurationen für Strapi

Für Strapi wird auch eine Secret-Komponente benötigt, da der Server geheime Umgebungsvariablen benötigt. Es gibt auch einige. Damit das Deployment von Strapi erstellt werden kann, muss die Anwendung zuerst containerisiert werden. Dies kann erfolgen mit der Verwendung des folgenden Dockerfiles.

Listing 5: Strapi Dockerfile

```

1  # Creating multi-stage build for production

```

```

2 FROM node:18-alpine as build
3 RUN apk update && apk add --no-cache build-base gcc autoconf automake zlib-dev
  libpng-dev vips-dev git > /dev/null 2>&1
4 ARG NODE_ENV=production
5 ENV NODE_ENV=${NODE_ENV}
6
7 WORKDIR /opt/
8 COPY package.json package-lock.json ./
9 RUN npm install -g node-gyp
10 RUN npm config set fetch-retry-maxtimeout 600000 -g && npm install
  --only=production
11 ENV PATH /opt/node_modules/.bin:$PATH
12 WORKDIR /opt/app
13 COPY . .
14 RUN npm run build
15 # Creating final production image
16 FROM node:18-alpine
17 RUN apk add --no-cache vips-dev
18 ARG NODE_ENV=production
19 ENV NODE_ENV=${NODE_ENV}
20 ENV HOST=0.0.0.0
21 ENV PORT=1337
22 ENV APP_KEYS=secret
23 ENV API_TOKEN_SALT=secret
24 ENV ADMIN_JWT_SECRET=secret
25 ENV TRANSFER_TOKEN_SALT=secret
26 # Database
27
28 # Database
29 ENV DATABASE_CLIENT=postgres
30 ENV DATABASE_HOST=localhost
31 ENV DATABASE_PORT=5432
32 ENV DATABASE_NAME=strapi
33 ENV DATABASE_USERNAME=strapi
34 ENV DATABASE_PASSWORD=strapi
35 ENV DATABASE_SSL=false
36 ENV JWT_SECRET=cVRog3q5woTNB8EJ+vKPFA==
37
38
39
40
41 WORKDIR /opt/
42 COPY --from=build /opt/node_modules ./node_modules
43 WORKDIR /opt/app
44 COPY --from=build /opt/app ./
45 ENV PATH /opt/node_modules/.bin:$PATH
46
47 RUN chown -R node:node /opt/app
48 USER node
49 EXPOSE 1337
50 CMD ["npm", "run", "start"]

```

Dieses Image soll dann in einem Container Registry wie zum Beispiel Dockerhub oder Github Container Registry hochgeladen werden, damit die K8S-Deployment-Komponente dieses Image im Einsatz nimmt.

9.10.6 K8S-Secrets

Damit man Umgebungsvariablen in einem K8S-Cluster definieren kann gibt es zwei Möglichkeiten. Man kann entweder eine ConfigMap-Komponente verwenden oder eine Secret-Komponente. Es gibt auch die Möglichkeit, diese Umgebungsvariablen direkt in den Konfigurationen von der Deployment-Komponente händisch einzutragen. Für geheime bzw. sensible Daten werden K8S-Secrets verwendet. Das Anlegen der benötigten Secrets für unsere Anwendung erfolgte durch die Verwendung folgender Befehle

Listing 6: Secrets für Strapi

```

1     kubectl create secret generic strapi-server-secret \
2     --from-literal=PORT=1337 \
3     --from-literal=APP_KEYS=secret
4     --from-literal=API_TOKEN_SALT=secret \
5     --from-literal=ADMIN_JWT_SECRET=secret \
6     --from-literal=TRANSFER_TOKEN_SALT=secret \
7     --from-literal=DATABASE_CLIENT=postgres \
8     --from-literal=DATABASE_PORT=5432 \
9     --from-literal=DATABASE_NAME=secret \
10    --from-literal=DATABASE_USERNAME=secret \
11    --from-literal=DATABASE_PASSWORD=secret \
12    --from-literal=DATABASE_SSL=false \
13    --from-literal=JWT_SECRET=secret

```

Listing 7: Secrets für die Datenbank

```

1 kubectl create secret generic strapi-secret \
2 --from-literal=POSTGRES_USER=strapi \
3 --from-literal=POSTGRES_PASSWORD=strapi \
4 --from-literal=POSTGRES_DB=strapi

```

9.10.7 Was ist eine Deployment-Komponente

Eine Deployment-Komponente dient dazu, dass ein Pod erstellt wird und die benötigten Ressourcen bzw. Volumes und Zugriffsrechten dafür definiert werden.[16]

Für das Anlegen einer Deployment-Komponente muss man zuerst die Anwendung containerisieren. Für weit verbreitete Software wie z.B.: PostgreSQL gibt es bereits viele vorhandene Images auf unterschiedliche Container-Registries

Folgende Befehle wurden verwendet, um die Deployment-Komponenten für Strapi und PostgreSQL zu erstellen:

Listing 8: create k8s deployments

```

1 kubectl create deployment relaxoon-db --image=postgres:12.16-bullseye --port=5432
2 kubectl create deployment relaxoon-strapi
   --image=ghcr.io/Abdulrahman-AL-Sabagh/relaxoon-strapi:latest --port=8080

```

Was ist ein Pod

“ Ein Pod (übersetzt Gruppe/Schote, wie z. B. eine Gruppe von Wälen oder eine Erbsenschote) ist eine Gruppe von einem oder mehreren Containern mit gemeinsam genutzten Speicher- und Netzwerkressourcen und einer Spezifikation für die Ausführung der Container. Die Ressourcen eines Pods befinden sich immer auf dem gleichen (virtuellen) Server, werden gemeinsam geplant und in einem gemeinsamen Kontext ausgeführt. Ein Pod modelliert einen anwendungsspezifischen "logischen Server": Er enthält eine

oder mehrere containerisierte Anwendungen, die relativ stark voneinander abhängen. ” [17]

9.10.8 Erstellung einer Service-Komponente

Mit dem Einsatz eines Services in Kubernetes können Pods, die sich im gleichen Cluster befinden, miteinander kommunizieren. [18]

Für das Anlegen einer Service-Komponente kann man diesen Befehl nutzen:

Listing 9: create a service component

```
1 kubectl expose deployments/<Name des Pods> --port=5432
```

9.10.9 Erstellung einer Ingress-Komponente

Damit der Emulator, der sich außerhalb des lokalen Clusters von Minikube befindet, mit dem deployten Backend kommunizieren kann, ist das Anlegen einer Ingress notwendig.

//TODO Config für die Ingress-Komponente noch hingeben

//TODO Dieses Kapitel noch fertig schreiben

“ In einem Computernetzwerk befindet sich ein einfacher Reverse-Proxy zwischen einer Gruppe von Servern und den Clients, die sie verwenden wollen. Als Client gilt jede Hardware oder Software, die Anfragen an einen Server senden kann. [...]. Der Reverse-Proxy fängt alle Anfragen von den Clients an die Server ab und liefert auch alle Antworten und Dienste von den Servern wieder an die Clients zurück. Aus Sicht des Kunden sieht es so aus, als würde alles von einer einzigen Stelle ausgehen.

” [19]

9.10.10 Buildprozess für das Frontend

Damit man das Frontend deployt, muss man zuerst den nativen Android und IOS Code generieren. Danach sollen die Android bzw. IOS Anwendungen mit einer IDE oder mit der Kommando-Zeile gebaut werden. Um den nativen Code zu generieren ist folgendes einzugeben:

Listing 10: generate android and ios

```
1 npx expo prebuild
```

9.10.11 Erstellung einer .apk Datei

Für Android wurde die **.Android PacKage** (apk) mit dem Einsatz von gradle Wrapper generiert. Folgendes muss installiert und richtig konfiguriert werden, damit eine .apk Datei generierbar ist:

- Java 11
- Installation von sdkmanager
- Lizenzen von sdkmanager müssen akzeptiert sein
- Installation von einer Android SDK
- Konfigurationen für JAVA_HOME und ANDROID_HOME
- Installation einer CLI namens "ninja"

Der Generierungsbefehl für die Build-Datei schaut dann wie folgendes aus:

Listing 11: generate apk

```
1 ./gradlew assembleRelease
```

9.11 Hochladen einer Mobileapp auf Firebase App Distribution

Nachdem Anlegen eines Accounts bei Firebase sind folgende und die Erstellung eines Projektes Schritte zu machen:

Damit die benötigten Firebase Services für die Applikationen aktiviert werden, muss man zuerst die benötigten Konfigurationen eingeben.

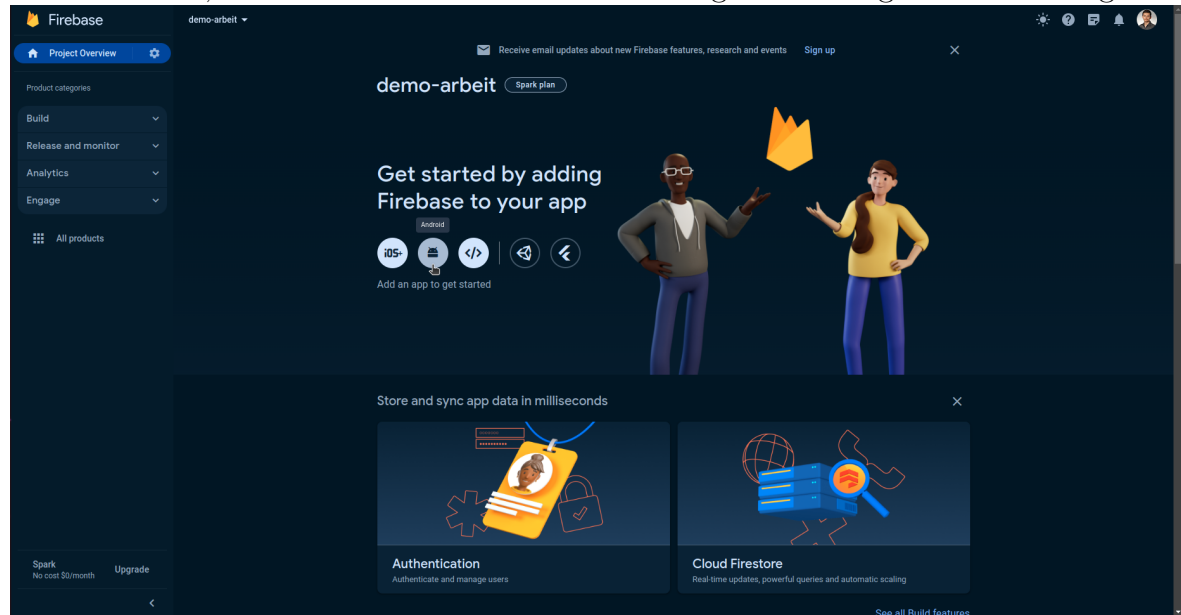


Abbildung 9: create android config

Folgende Daten müssen eingegeben werden, damit die Firebase Services für die Android Applikation eingeschaltet werden können.

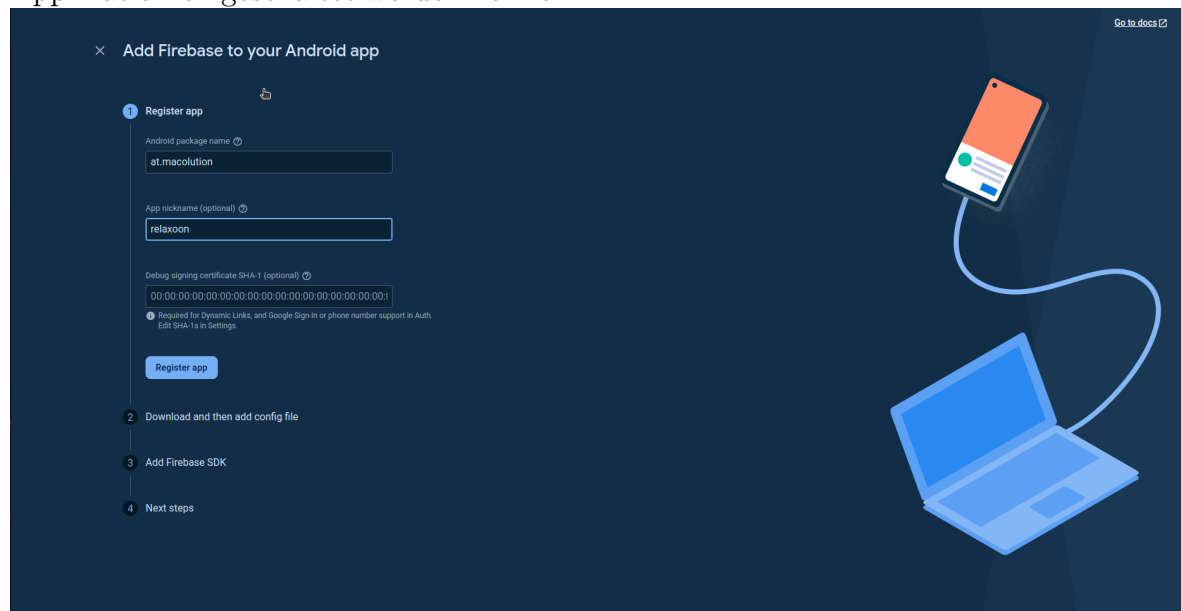


Abbildung 10: Android Config

Die restlichen Punkte sind für das Projekt Relaxoon irrelevant, da die Firebase-SDK in der vorliegende Arbeit nicht eingesetzt wird.

Danach soll man auf die App Distribution gehen und dann auf "Get Started" klicken

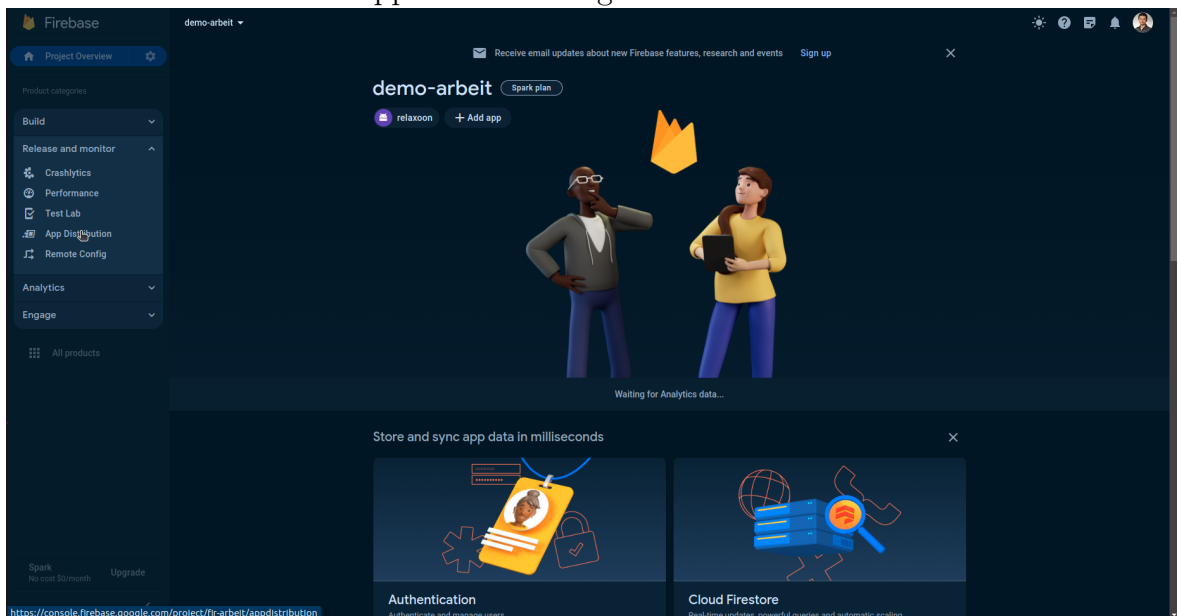


Abbildung 11: App Distribution

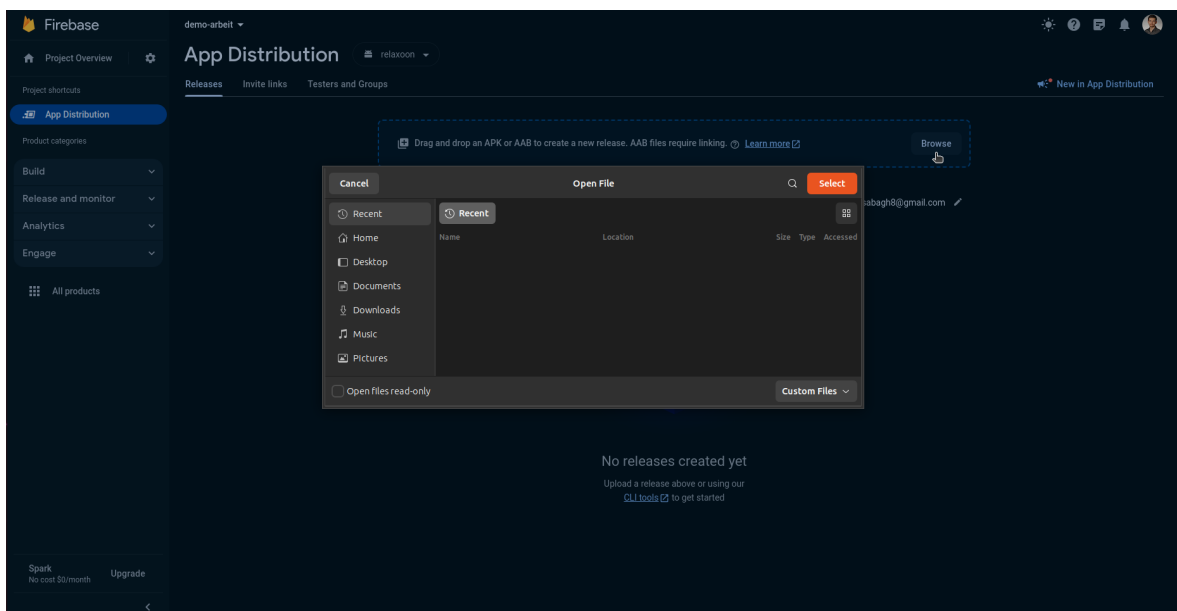


Abbildung 12: App upload

Die App kann mit diesem "Browse Fenster" oder mit "Drag and Drop" hochgeladen werden

Das Produkt kann dann an anderen TestUsers verteilt werden, indem man die Email-Adressen dieser Testers beim Release eingibt.

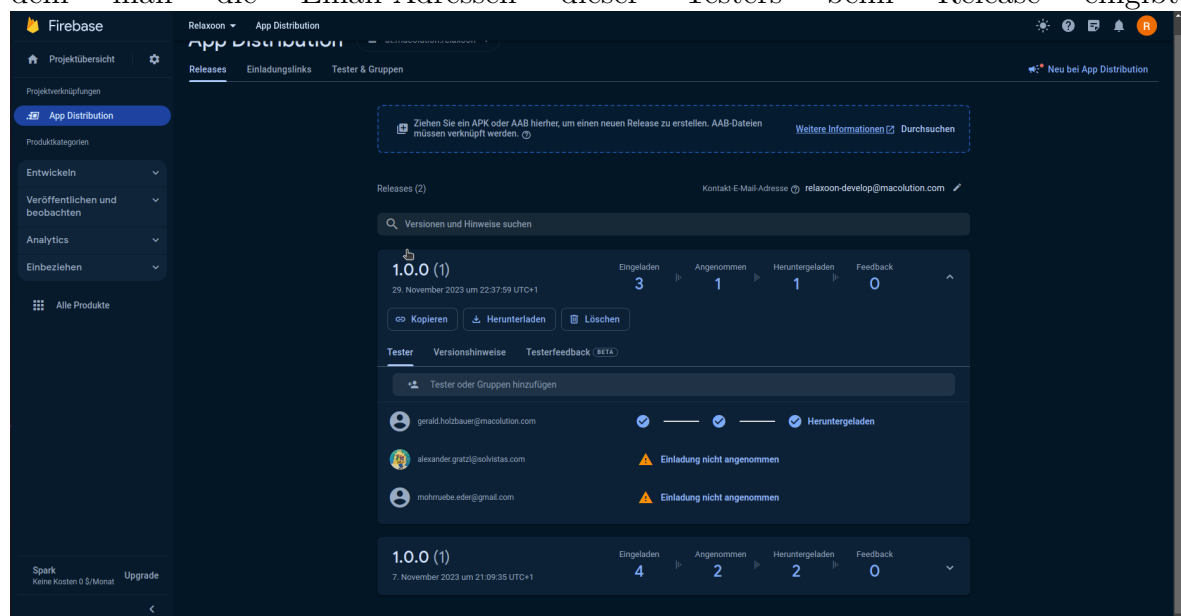


Abbildung 13: releases

10 Ausgewählte Aspekte und Probleme

10.1 Probleme mit localhost und https

Im Projekt wurde die **Uniform Resource Locator** (URL) für den (API)-Server in einem ".env"-File gespeichert. Für die lokale Entwicklung wurde Strapi auf dem lokalen Host immer verwendet. Bei IOS und Android ist das Holen der Daten von einer nicht sicheren Quellen **Also Known As** (aka) nur http Protokolle gar nicht möglich. Auf Android gibt aber einige Ausnahmen für das Holen der Daten von dem lokalen Host. Bei Android kann man bei den generierten Build und Project Files einige Konfigurationen ändern. Das ist zwar keine gute Lösung, da diese Files bei jedem Build geändert werden können. Außerdem können diese generierten Files nicht in das **Version Control System** (VCS) eingchecked werden. Bei Android muss man 10.0.2.2 statt localhost schreiben.[20] Bei IOS gibt es keine einzige Möglichkeit, um Daten aus http Quellen oder aus dem localhost zu holen. Als Lösung wurde in der lokalen Entwicklung eine **Command Line Interface** (CLI) namens "ngrok" verwendet, welche den lokalen Port des Servers ins Internet mittels eines Tunnels weiterleitet und eine https Adresse für den weitergeleiteten Server generiert.

Das Problem tauchte wiederholt in der Test-Phase auf und war für die Entwickler nicht lösbar, da der Staging Server für das Backend von Relaxoon auf den Servern der Solvistas deployt ist. Die Entwickler verfügen über keine Zugriffsrechte auf diesem Server. Das Problem wurde mithilfe von den Netzwerkadministratoren von Solvistas gelöst. Es wurde ein "Let's encrypt"-Zertifikat für den Staging Server eingebaut.

10.2 Inkompatible Libraries beim Build-Prozess

In der Entwicklungsphase wurden einige Libraries mittels **Node Package Manager** (npm) installiert. Einige Libraries waren mit der Node Version oder mit anderen Libraries nicht kompatibel. Eine Lösung wäre, das Label "-legacy-peer-deps" oder "-force" beim Befehl "npm install" anzuhängen. Während der Entwicklung wurde das Label "-legacy-peer-deps" immer beim "npm install" integriert und danach hat alles problemlos funktioniert. Später beim Deployment von Android ist das Problem bei der Generierung von Android apk bzw. **Android App Bundle** (aab) aufgetaucht. Die Apk bzw. Aab wurde erfolgreich generiert, aber die App war nicht funktionsfähig. Es wurde für das Finden des Problems auf Android ein Tool namens "logcat" verwendet, um die Fehlermeldung zu finden. Dieses Problem an sich ist in der Community sehr stark verbreitet und es gibt dafür mehrere Gründe und mehrere Lösungen. Unter diesem Githublink [21] werden unterschiedliche Gründe und mehrere Lösungsmöglichkeiten für das gleiche Problem beschrieben. Es wurde jede einzelne Lösungsmöglichkeit versucht und keine von diesen vorgeschlagenen Lösungen funktionierte. Aus Erfahrung hat man gewusst, dass die Codebase einige Libraries hat, die nicht mehr gebraucht bzw. verwendet werden. Beim Löschen dieser Libraries und die Wiederinstallation von den Modulen mittels "npm install" tauchte diese "legacy-peer-deps"-Meldung wieder auf. Man dachte, dass das Problem daran liegt

und es wurde ein Tool namens "depcheck" installiert, welches den Codebase scannt und den Entwickler bekannt gibt, ob eine Library im Codebase verwendet wird oder nicht. [22] Nach dem Löschen aller nicht benötigten Libraries und die Generierung der apk bzw. aab hat die Applikation problemlos funktioniert.

10.2.1 npm install --legacy-peer-deps

“ The ‘--legacy-peer-deps’ flag is used when you encounter compatibility issues with peer dependencies while installing packages. Peer dependencies are required by a package but aren’t automatically installed alongside it. In some cases, when a package has not been updated to support the latest version of its peer dependency, the installation may fail due to conflicting versions. Adding the ‘--legacy-peer-deps’ flag allows npm to use an older, compatible version of the peer dependency, ensuring a successful installation.” [23]

10.2.2 npm install --force

“ The ‘--force’ flag is a more drastic option and should be used with caution. It instructs npm to forcefully install packages, even if it encounters errors or conflicts. This can be useful in situations where you want to override any version or compatibility checks and forcibly install packages. However, it is important to note that using ‘--force’ may lead to unexpected issues, such as breaking dependencies or introducing incompatibilities, so it should be used sparingly and with a good understanding of its consequences.” [23]

10.3 Probleme mit Thumbnails

Unsere App soll mehrere Videos für Antistress-Meditationen anzeigen. Jedes Video muss unbedingt ein Thumbnail haben. Der Content-Manager könnte aber beim Erstellen des Videos vergessen, ein Thumbnail für das Video zu erstellen. Auf unserer Adminoberflächen können keine Thumbnails zu dem Video hinzugefügt werden, da diese Aktion sehr schlechte User Experience verursachen könnte. Man kann zwar eine Funktion implementieren, die nach dem Hochladen eines Videos ein Thumbnail mittels **F**ast **F**orward **M**oving **P**icture **E**xperts **G**roup (ffmpeg) aus dem ersten Bild des Videos erstellt. Diese Möglichkeit war aber sehr aufwendig und nicht empfehlenswert, da die damalige Dokumentation von Strapi nicht sehr genau war. Außerdem könnte diese Alternative bei den Updates von Dependencies nicht mehr funktionsfähig sein. Zum Glück haben wir eine Expo-Library gefunden, die im Frontend Thumbnails für unsere Videos erstellt. Die Verwendung davon war aber nicht sehr vorteilhaft, da der Generierungsprozess sehr langsam war. Als Lösung haben wir uns entschieden, ein nicht abspielbares Video auf den Screens, wo mehrere Videos vorgeschlagen werden, anzuzeigen. Das Video ist erst abspielbar, wenn man darauf klickt. Wir haben diese Lösung mit den Konzepten "Lazy Loading" und "SSuspenses" zusammen kombiniert, damit Loading-Spinner statt leere Flächen für den User angezeigt werden, wenn das Laden des Videos etwas länger dauert.

10.3.1 Suspense

“<Suspense> lets you display a fallback until its children have finished loading.” [24]

10.3.2 Lazy Loading

“lazy lets you defer loading component’s code until it is rendered for the first time.” [25]

10.4 Dauer von Videos

Beim Hochladen eines Videos in Strapi, wird die Dauer des Videos nur auf der Oberfläche angezeigt. Diese Information ist aber in den REST-Schnittstellen der Meta-Daten von den Files nicht inkludiert. Bei diesem Problem ist die Verwendung von ffmpeg auch eine Lösungsmöglichkeit. Es wurde aber nicht mit ffmpeg gearbeitet (siehe das obige Problem). Für die Berechnung der Dauers wurde im Frontend das `onLoadEvent`, welches in den Properties (props) des Videoelements ist, verwendet, um die Dauer zu lesen und diese in das Format "mm:ss" umwandeln zu können.

10.5 Probleme mit useEffect und React-navigation Library

Für das Holen der Daten aus dem Server würde die Fetch API verwendet. Diese wurde auch in einem `useEffect` Hook eingegeben, damit die Daten bei jeder Aktualisierung eines spezifischen Zustandes aus dem Server geholt werden können. Beim Anklicken der unterschiedlichen Navigationsbuttons wurde bemerkt, dass die Daten sich gar nicht ändern. Am Anfang wurde vermutet, dass das Problem ein Caching Problem sein könnte. Nach einer Recherche kam man darauf, dass die Library "React-navigation" den `useEffect` Hook im Hintergrund deaktiviert. Für das Holen der Daten ist die Übergabe ein sogenanntes `useCallback` Hooks Parameter in einem custom hook namens `useFocusEffect` von React-navigation notwendig. [26]

10.5.1 Hooks

“Hooks let you use different React features from your components. You can either use the built-in Hooks or combine them to build your own. This page lists all built-in Hooks in React.” [27]

10.5.2 useEffect

“useEffect is a React Hook that lets you synchronize a component with an external system.” [28]

“Some components need to synchronize with external systems. For example, you might want to control a non-React component based on the React state, set up a server connection, or send an analytics log when a component

appears on the screen. Effects let you run some code after rendering so that you can synchronize your component with some system outside of React.[...] Effects let you specify side effects that are caused by rendering itself, rather than by a particular event. ” [29]

10.5.3 useCallback

“useCallback is a React Hook that lets you cache a function definition between re-renders.” [30]

10.5.4 useFocusEffect

“The useFocusEffect is analogous to React’s useEffect hook. The only difference is that it only runs if the screen is currently focused. The effect will run whenever the dependencies passed to React.useCallback change, i.e. it’ll run on initial render (if the screen is focused) as well as on subsequent renders if the dependencies have changed. If you don’t wrap your effect in React.useCallback, the effect will run every render if the screen is focused.” [31]

11 Resümee

Coming soon

Literaturverzeichnis

- [1] T. Aichinger, „MACOLUTION,” 2023. Online verfügbar: <https://www.solvistas.com/de/news/solvistas-group-waechst-weiter>
- [2] Unbekannt, „Stress.” Online verfügbar: [https://www.internisten-im-netz.de/fachgebiete/psyche-koerper/stress.html#:~:text=Zeichen%20von%20Nervosit%C3%A4t%20\(Z%C3%A4hneknirschen%20in,Stoffwechselst%C3%B6rungen%2C%20Allergien%20und%20Entz%C3%BCndungskrankheiten%20f%C3%BChren.\)](https://www.internisten-im-netz.de/fachgebiete/psyche-koerper/stress.html#:~:text=Zeichen%20von%20Nervosit%C3%A4t%20(Z%C3%A4hneknirschen%20in,Stoffwechselst%C3%B6rungen%2C%20Allergien%20und%20Entz%C3%BCndungskrankheiten%20f%C3%BChren.))
- [3] —, „Ziele setzen,” 2023. Online verfügbar: <https://karrierebibel.de/ziele-setzen/>
- [4] C. Wannakhao, „Strapi vs. WordPress,” 2022. Online verfügbar: <https://www.trienpont.com/strapi-vs-wordpress-which-one-should-you-use-for-your-next-cms-project/#:~:text=Strapi%20is%20again%20the%20winner,that%20need%20to%20be%20upd>
- [5] firebase.google.com, „Firebase-App-Verteilung,” 2023, letzter Zugriff am 20.12.2023. Online verfügbar: <https://firebase.google.com/docs/app-distribution?hl=de>
- [6] L. Hahn, „Cross-Platform App – Plattformübergreifende Entwicklung mit Flutter, React Native & Co.” 2023. Online verfügbar: <https://www.itportal24.de/ratgeber/cross-platform-app>
- [7] yeeply.com, „Native vs. hybrid,” 2023, letzter Zugriff am 10.11.2023. Online verfügbar: <https://www.yeeply.com/de/blog/was-sind-native-web-und-hybride-apps/>
- [8] N. Mansour, „React Native vs Kotlin Multiplatform: The 2023 Guide,” 2023. Online verfügbar: <https://www.instabug.com/blog/react-native-vs-kotlin-multiplatform-guide>
- [9] L. Hahn, „Flutter vs. React Native,” 2023. Online verfügbar: <https://www.itportal24.de/ratgeber/flutter-vs-react-native#React-vs-Flutter>
- [10] L. von Arcitech, „Xamarin vs. React Native,” 2023. Online verfügbar: <https://www.linkedin.com/pulse/xamarin-vs-react-native-comprehensive-comparison-cross-platform/>
- [11] F. Churchville, „Content-Management-System (CMS),” 2021. Online verfügbar: <https://www.computerweekly.com/de/definition/Content-Management-System-CMS>
- [12] ionos, „Headless CMS,” 2022. Online verfügbar: <https://www.ionos.at/digitalguide/hosting/cms/headless-cms-was-sind-die-vorteile/>
- [13] S. Springer, „Zentrales State Management in React ,” 2022. Online verfügbar: <https://entwickler.de/react/zentrales-state-management-react>
- [14] react.dev, „useState,” 2024, letzter Zugriff am 23.2.2024. Online verfügbar: <https://react.dev/reference/react/useState>

- [15] M. Gold, „Using useContext in React: a comprehensive guide,” 2023. Online verfügbar: <https://medium.com/@msgold/using-usecontext-in-react-a-comprehensive-guide-8a9f5271f7a8#:~:text=useContext%20is%20a%20React%20hook,built%20into%20the%20React%20library>.
- [16] kubernetes.io, „Deployments,” 2023, letzter Zugriff am 19.12.2023. Online verfügbar: <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
- [17] —, „Pods,” 2023, letzter Zugriff am 19.12.2023. Online verfügbar: <https://kubernetes.io/de/docs/concepts/workloads/pods/>
- [18] —, „Service,” 2023, letzter Zugriff am 19.12.2023. Online verfügbar: <https://kubernetes.io/docs/concepts/services-networking/service/>
- [19] C. Pinchfsky, „Reverse-Proxy-Server erläutert: Was sie sind und wie man sie benutzt,” 2020. Online verfügbar: <https://www.avast.com/>
- [20] StackOverflow, „fetching data from localhost on android,” 2016. Online verfügbar: <https://stackoverflow.com/questions/33704130/react-native-android-fetch-failing-on-connection-to-local-api>
- [21] facebook/react-native github mhrpatel12, „couldn’t find DSO to load: libjscexecutor.so caused by: dlopen failed: library "libjsc.so" not found,” 2019. Online verfügbar: <https://github.com/facebook/react-native/issues/25537>
- [22] npmjs.com depcheck, „Wie funktioniert depcheck,” 2023. Online verfügbar: <https://www.npmjs.com/package/depcheck>
- [23] S. Saleem, „npm install –legacy-peer-deps vs –force,” 2023. Online verfügbar: <https://www.linkedin.com/pulse/npm-install-legacy-peer-deps-vs-force-shaharyar-saleem/>
- [24] react.dev, „Suspenses,” 2023. Online verfügbar: <https://react.dev/reference/react/Suspense>
- [25] —, „Lazy Loading,” 2023. Online verfügbar: <https://react.dev/reference/react/lazy>
- [26] StackOverflow, „useEffect not called in React Native when back to screen,” 2020. Online verfügbar: <https://stackoverflow.com/questions/60182942/useeffect-not-called-in-react-native-when-back-to-screen>
- [27] react.dev, „Hooks,” 2023. Online verfügbar: <https://react.dev/reference/react/hooks>
- [28] —, „useEffect hook,” 2023. Online verfügbar: <https://react.dev/reference/react/useEffect>
- [29] —, „Synchronizing with Effects,” 2023. Online verfügbar: <https://react.dev/learn/synchronizing-with-effects>
- [30] —, „useCallback,” 2023. Online verfügbar: <https://react.dev/reference/react/useCallback>
- [31] reactnavigation.org, „useFocusEffect,” 2023. Online verfügbar: <https://reactnavigation.org/docs/use-focus-effect/>

Abbildungsverzeichnis

| | | |
|----|-------------------------------------------------------|----|
| 1 | Logo MACOLUTION | 1 |
| 2 | Logo solvistas | 1 |
| 3 | Musik dient zur Entspannung | 2 |
| 4 | | 3 |
| 5 | Systemarchitektur | 10 |
| 6 | ERD | 16 |
| 7 | Screenshot aus dem UI Prototyp von Relaxoon | 17 |
| 8 | Beispiel für den Interactive Query Builder | 18 |
| 9 | create android config | 29 |
| 10 | Android Config | 29 |
| 11 | App Distribution | 30 |
| 12 | App upload | 30 |
| 13 | releases | 31 |

Tabellenverzeichnis

Quellcodeverzeichnis

| | | |
|----|--------------------------------------|----|
| 1 | protected screens | 20 |
| 2 | Validierungsschemen | 22 |
| 3 | K8S PVC | 24 |
| 4 | K8S PV | 24 |
| 5 | Strapi Dockerfile | 24 |
| 6 | Secrets für Strapi | 26 |
| 7 | Secrets für die Datenbank | 26 |
| 8 | create k8s deployments | 26 |
| 9 | create a service component | 27 |
| 10 | generate android and ios | 27 |
| 11 | generate apk | 28 |

Anhang