

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

<Coming soon>

Inhaltsverzeichnis

1	Abstract	1
2	Ausgangssituation	2
3	Problemstellung	3
3.1	Coming Soon	3
4	Ziele	4
5	Aufgabenstellung	5
6	Systemarchitektur	6
6.1	Komponentendiagramm	6
7	Entwurfsentscheidungen	8
7.1	React Native	8
7.2	Strapi	10
8	Umsetzung	12
8.1	Entity Relationship Diagram (ERD)	12
8.2	Medias und Articles	12
8.3	TutorialPage	13
8.4	Suche und Filterungen	13
9	Ausgewählte Aspekte und Probleme	16
9.1	Probleme mit localhost und https	16
9.2	Inkompatible Libraries beim Build-Prozess	16
9.3	Probleme mit Thumbnails	17
9.4	Dauer von Videos	18
9.5	Probleme mit useEffect und React-navigation Library	18

10 Resümee	20
Literaturverzeichnis	V
Abbildungsverzeichnis	VII
Tabellenverzeichnis	VIII
Quellcodeverzeichnis	IX
Anhang	X

1 Abstract

<Coming soon>

2 Ausgangssituation

<Coming soon>

3 Problemstellung

3.1 Coming Soon

4 Ziele

5 Aufgabenstellung

6 Systemarchitektur

6.1 Komponentendiagramm

Für eine klare funktionale Übersicht auf unserer Diplomarbeit wurde ein sogenanntes Komponentendiagramm für das technische System von Relaxoon erstellt. Da Relaxoon eine Applikation ist, die auf Handys laufen soll, wurde für die Entwicklung die cross plattformige JavaScript (js) Framework "React Native" verwendet. Damit der Kunde Inhalte in die App hinzufügen kann, wurde ein "Node js"basiertes Headless Content Management System (CMS) namens "Strapi" verwendet. Dieses bietet eine vorgefertigte Benutzeroberfläche für die Content Creators und auch für die Entwickler, sodass man Inhalte und dazu gebrauchte technische Funktionalitäten, wie zum Beispiel **RE**presentational **S**tate **T**ransfer (REST)- bzw. **G**raph **Q**uery **L**anguage (GraphQL)-Schnittstellen für die Authentifizierung und **C**reate, **R**ead, **U**ppdate und **D**eleete (CRUD) Funktionalitäten jeder Business Entität, sehr einfach erstellen kann. Für die Kommunikation zwischen Frontend und Backend wird REST verwendet. Das Backend holt sich die Daten mit Hilfe eines postgresql Treibers aus der Datenbank.

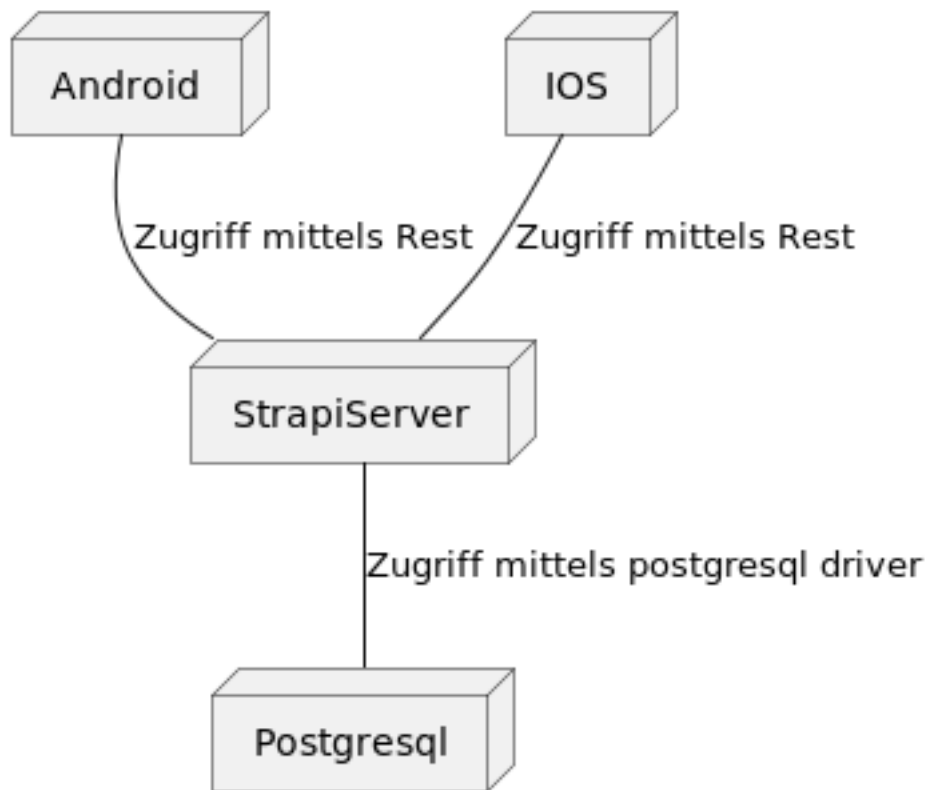


Abbildung 1: Systemarchitektur

7 Entwurfsentscheidungen

7.1 React Native

Grundsätzlich wurde React Native aus mehreren Gründen verwendet:

- Die Firma Solvistas beschäftigt sich sehr viel mit React Native Frontends und daher war die Verwendung davon verlangt.
- Vorhande Kenntnisse in den Sprachen Javascript bzw. Typescript verwenden
- Der Auftragsgeber wollte von Beginn an eine Cross-Plattformige-Lösung verwenden
- Endergebnis von React Native ist eine Native App

7.1.1 Was ist eine Crossplattformige 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.” [1]

7.1.2 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. ” [2]

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. ” [2]

Grundsätzlich sind native Applikationen viel schneller und barrierefreier als hybride Applikationen [2]

7.1.3 Warum wurde eine crossplattformige Lösung verwendet

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 crossplattformigen Lösung sehr vorteilhaft, da alle Plattformen von einem Codebase gepflegt werden können.

7.1.4 andere Alternativen für React Native

- Kotlin Mutliplattform
- Flutter
- Xamarin

7.1.5 React Native vs. Kotlin Multiplattform

Kotlin Mutliplattform 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. ” [3]

Allerdings ist das Problem davon, dass Kotlin Multiplattform noch immer im Beta ist und daher ist es 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 Multiplattform auch verlangt. [3]

7.1.6 React Native vs. 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. [4] Das Problem bei Flutter ist, dass man gute Kenntnisse in der Programmiersprache "Dart" haben muss. [4] 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.

7.1.7 React Native vs. Xamarin

Xamarin ist eine C# Framework, die für die Entwicklung von nativen crossplattformigen Lösungen zuständig ist. Die Performance von Xamarin ist viel besser als die Performance von React Native. [5]

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 Programm bei jeder kleinen Änderung neugestartet werden. [5]

7.2 Strapi

7.2.1 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.” [6]

7.2.2 Warum wurde ein CMS verwendet

Der Auftragsgeber 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.

7.2.3 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. ” [7]

7.2.4 Warum wurde ein Headless-CMS verwendet

- Vorgabe von dem Auftragsgeber
- 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.

7.2.5 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

[8]

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

[8]

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. [8]

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. [8]

Ergebnis

Alle erwähnten Pros und Contras zeigen, dass die **User EX**perience (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"[8]

8 Umsetzung

8.1 Entity Relationship Diagram (ERD)

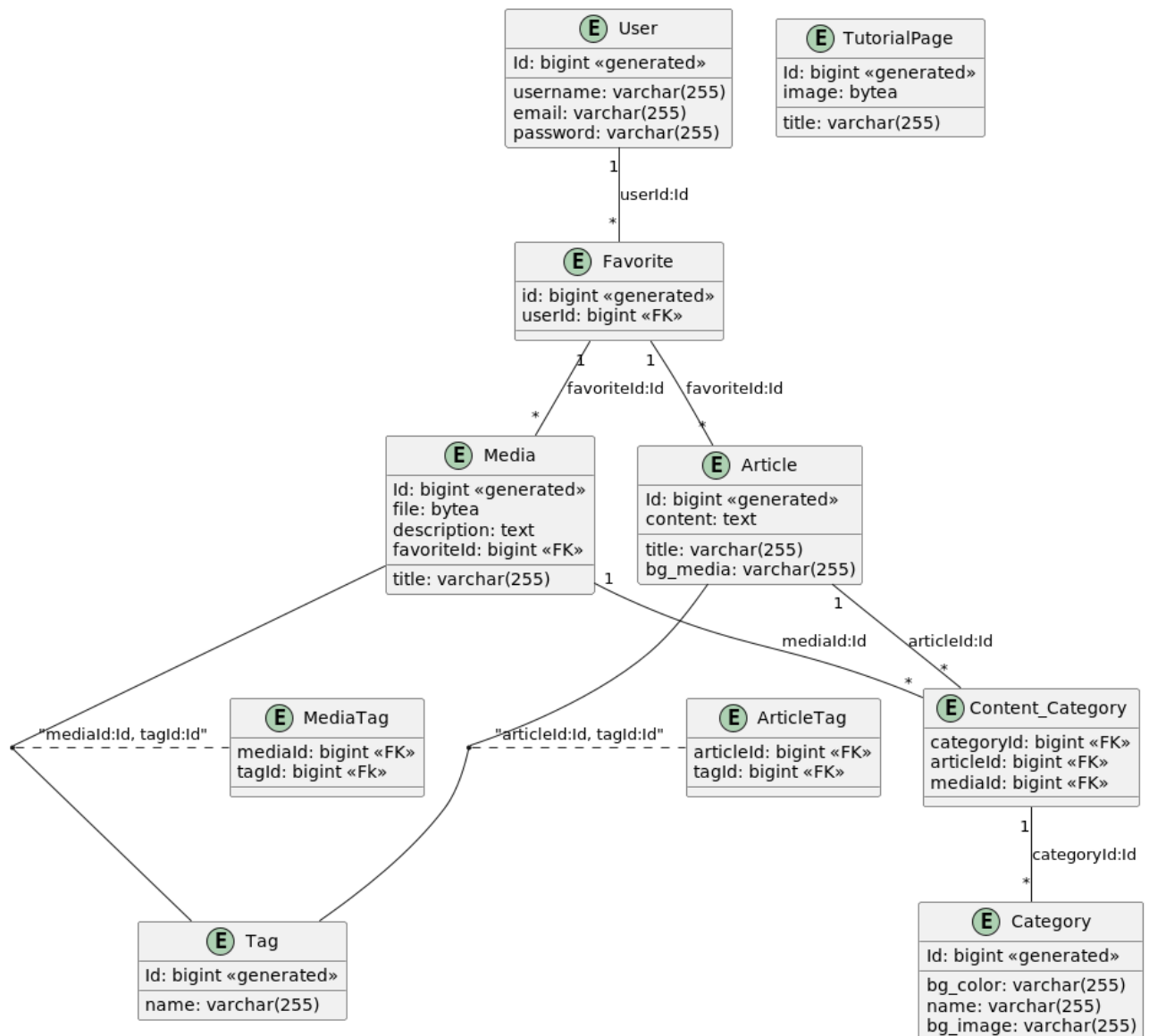


Abbildung 2: ERD

8.2 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 Datenmodels war aber das Einfügen der Artikel erforderlich.

8.3 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.

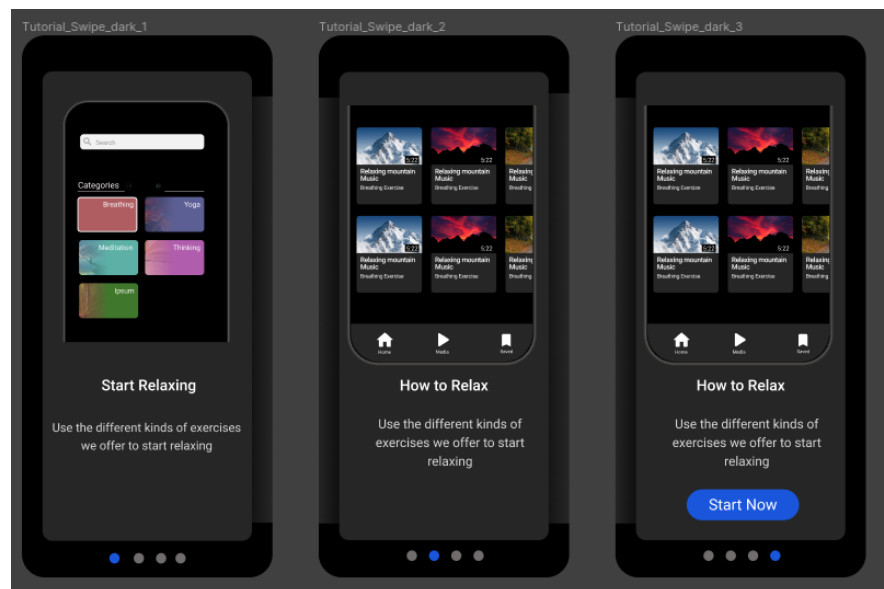


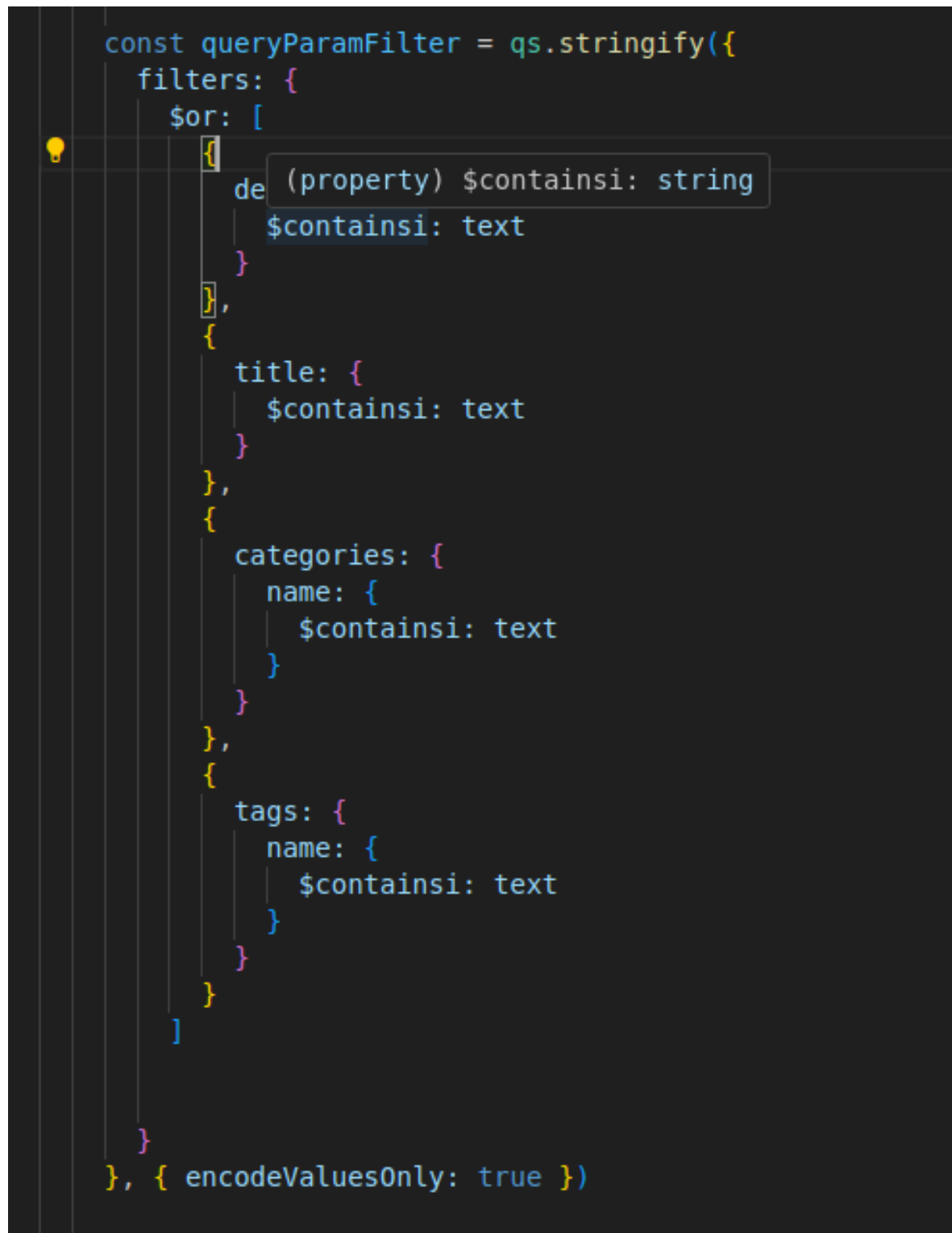
Abbildung 3: Screenshot aus dem UI Prototyp von Relaxoon

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

8.4 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 4: 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 Ausgewählte Aspekte und Probleme

9.1 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.[9] 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.

9.2 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 **Android Package** (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 [10] 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 der 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. [11] Nach dem Löschen aller nicht benötigten Libraries und die Generierung der apk bzw. aab hat die Applikation problemlos funktioniert.

9.2.1 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.” [12]

9.2.2 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.” [12]

9.3 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.

9.3.1 Suspense

“<Suspense> lets you display a fallback until its children have finished loading.” [13]

9.3.2 Lazy Loading

“lazy lets you defer loading component’s code until it is rendered for the first time.” [14]

9.4 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.

9.5 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. [15]

9.5.1 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.” [16]

9.5.2 useEffect

“useEffect is a React Hook that lets you synchronize a component with an external system.” [17]

“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. ” [18]

9.5.3 **useCallback**

“useCallback is a React Hook that lets you cache a function definition between re-renders.” [19]

9.5.4 **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.” [20]

10 Resümee

Coming soon

Literaturverzeichnis

- [1] L. Hahn, „Cross-Platform App – Plattformübergreifende Entwicklung mit Flutter, React Native & Co.” 2023. Online verfügbar: <https://www.itportal24.de/ratgeber/cross-platform-app>
- [2] 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/>
- [3] N. Mansour, „React Native vs Kotlin Mutliplatform: The 2023 Guide,” 2023. Online verfügbar: <https://www.instabug.com/blog/react-native-vs-kotlin-mutliplatform-guide>
- [4] L. Hahn, „Flutter vs. React Native,” 2023. Online verfügbar: <https://www.itportal24.de/ratgeber/flutter-vs-react-native#React-vs-Flutter>
- [5] L. von Arcitech, „Xamarin vs. React Native,” 2023. Online verfügbar: <https://www.linkedin.com/pulse/xamarin-vs-react-native-comprehensive-comparison-cross-platform/>
- [6] F. Churchville, „Content-Management-System (CMS),” 2021. Online verfügbar: <https://www.computerweekly.com/de/definition/Content-Management-System-CMS>
- [7] ionos, „Headless CMS,” 2022. Online verfügbar: <https://www.ionos.at/digitalguide/hosting/cms/headless-cms-was-sind-die-vorteile/>
- [8] 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>
- [9] 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>
- [10] 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>
- [11] npmjs.com depcheck, „Wie funktioniert depcheck,” 2023. Online verfügbar: <https://www.npmjs.com/package/depcheck>
- [12] 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/>
- [13] react.dev, „Suspenses,” 2023. Online verfügbar: <https://react.dev/reference/react/Suspense>

- [14] —, „Lazy Loading,” 2023. Online verfügbar: <https://react.dev/reference/react/lazy>
- [15] 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>
- [16] react.dev, „Hooks,” 2023. Online verfügbar: <https://react.dev/reference/react/hooks>
- [17] —, „useEffect hook,” 2023. Online verfügbar: <https://react.dev/reference/react/useEffect>
- [18] —, „Synchronizing with Effects,” 2023. Online verfügbar: <https://react.dev/learn/synchronizing-with-effects>
- [19] —, „useCallback,” 2023. Online verfügbar: <https://react.dev/reference/react/useCallback>
- [20] reactnavigation.org, „useFocusEffect,” 2023. Online verfügbar: <https://reactnavigation.org/docs/use-focus-effect/>

Abbildungsverzeichnis

1	Systemarchitektur	7
2	ERD	12
3	Screenshot aus dem UI Prototyp von Relaxoon	13
4	Beispiel für den Interactive Query Builder	14

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang