

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 2024

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.

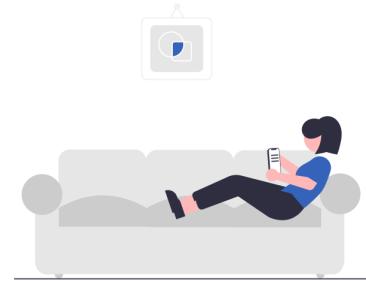
Leonding, April 2024

Abdulrahman Al Sabagh & Moritz Eder

Abstract

In order to give stressed people a simple way to relax, a stress management app has been developed.

Relaxoon is a play on the words 'relax' and 'soon'. Relaxoon offers relaxation exercises in the form of videos, texts, articles and more, which are presented in a simple and clear way.



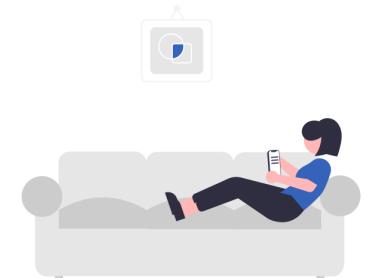
In addition to implementing the front end in React Native and the back end with the content management system Strapi, the deployment using Kubernetes is described in detail.

Zusammenfassung

Damit gestresste Personen eine simple Möglichkeit haben, sich zu entspannen, wurde eine App zur Stressbewältigung entwickelt.

Relaxoon ist ein Wortspiel aus den englischen Wörtern 'relax' und 'soon'. Relaxoon bietet Entspannungsübungen in Form von Videos, Texten, Artikeln und mehr, die in einfacher und übersichtlicher Weise dargestellt sind.

Neben der Implementierung des Frontends in React Native und des Backends mit dem Content-Management-System Strapi wird auf das Deployment mittels Kubernetes im Detail eingegangen.



Inhaltsverzeichnis

1 Ausgangssituation	1
2 Problemstellung	2
3 Ziele	3
3.1 Projektziele	4
4 Aufgabenstellung	6
4.1 Use-Case-Diagram (UCD)	7
5 Marktanalyse	10
5.1 Vorgehensweise	10
5.2 Analyse	11
6 Projektanforderungen	16
6.1 Funktionale Anforderungen:	16
6.2 Nicht funktionale Anforderungen:	16
7 Technologien	18
7.1 Strapi	18
7.2 Firebase App Distribution	24
8 Systemarchitektur	25
8.1 Komponentendiagramm	25
9 Entwurfsentscheidungen	27
9.1 React Native	27
9.2 Strapi	29
9.3 Headless-CMS vs. selbst geschriebener Server	33
9.4 Component Library	34
9.5 Typescript	34

9.6 Warum Expo und nicht React Native CLI	35
10 Design	36
10.1 Mockups	36
10.2 Tutorial	37
10.3 Login- und Registrierungsformular	39
10.4 Home-Screen	42
10.5 Kategorien	43
10.6 Media Player	45
10.7 Suchleiste	46
10.8 Favoriten	47
10.9 Einstellungen	49
10.10 Light/Dark Mode	50
10.11 Logo	52
11 Implementierung	53
11.1 Entity Relationship Diagram (ERD)	53
11.2 Wie werden Files gespeichert?	53
11.3 Medias und Articles	54
11.4 Tutorial-Page	54
11.5 Suche und Filterungen	55
11.6 State Management	57
11.7 Dark/Light Mode	57
11.8 Help und Info Screen	58
11.9 Authentifizierung	58
11.10 Validierung	59
11.11 Deployment	60
11.12 Hochladen einer Mobileapp auf Firebase App Distribution	66
12 Ausgewählte Aspekte und Probleme	69
12.1 Probleme mit localhost und https	69
12.2 Inkompatible Libraries beim Build-Prozess	71
12.3 Probleme mit Thumbnails	72
12.4 Dauer von Videos	73
12.5 Probleme mit useEffect und React-navigation Library	73
12.6 Probleme mit Dependency Updates	74

12.7 Probleme mit dem Datenmodell	75
12.8 Relaxoon Plakat	76
13 Resümee	79
14 Anhang	80
14.1 Prompt-Verlauf	80
14.2 Schriftliche Arbeitsaufteilung	VI
Literaturverzeichnis	VII
Abbildungsverzeichnis	XI
Tabellenverzeichnis	XIII
Quellcodeverzeichnis	XIV

1 Ausgangssituation

Heutzutage gibt es Stresssituationen, die dazu führen können, dass man manchmal den Überblick über die eigenen Aufgaben verliert. Um auch auf lange Zeit und produktiv Aufgaben erfüllen zu können, ist es sehr wichtig, dass man Ruhe und Entspannungsphasen in den Alltag einbaut, sei es in der Schule, der Universität oder in der Arbeit.

Langes und hartes Arbeiten kann zu zahlreichen gesundheitlichen Folgen führen, wobei Stress und Leistungsdruck das noch zusätzlich verschlimmern können. Eine häufige Konsequenz ist zum Beispiel ein Burnout. Gemäß einer repräsentativen Studie [1] aus dem Jahr 2016/17 fühlen sich nur 52 Prozent der Österreicher:innen tatsächlich gesund. 19 Prozent der Proband:innen befinden sich bereits in einem Burnout-Frühstadium, 17 Prozent sogar in einem Übergangsstadium. Schließlich ergab diese Studie, dass 8 Prozent der Österreicher:innen unter dem Burnout-Syndrom leiden.

Deshalb greifen viele Menschen auf die Musik zurück, wodurch sie sich beim Arbeiten entweder besser konzentrieren oder kurz eine Pause einlegen können.

Zur Stressbewältigung wurde von der MACOLUTION GmbH und dem Data Science Unternehmen solvistas GmbH eine Lösung für dieses Problem entwickelt. Die MACOLUTION GmbH ist ein Unternehmen, welches sich zur Aufgabe gemacht hat, Management- und Coaching-Solutions mit modernsten Technologien und bewährten Methoden zu entwickeln. [2] Mit Relaxoon soll diese Idee Wirklichkeit werden.



Abbildung 1: Logo MACOLUTION



Abbildung 2: Logo solvistas

2 Problemstellung

In der heutigen Welt entstehen immer mehr Termine für Besprechungen oder Meetings. Dadurch wird der generelle Alltag hektischer und schneller, was sowohl oft im Arbeitsleben als auch im privaten Bereich zutrifft. Daher empfinden viele Menschen den Ablauf des Tages persönlich als stressig.

Es fehlt außerdem die Möglichkeit sich zu entspannen und die Betroffenen verlieren den Mut sich Hilfe zu holen.

Stress oder Leistungsdruck kann neben einem Burnout auch noch weitere gesundheitliche Folgen hervorrufen. Mögliche Folgen von hohem Stress oder hohem Leistungsdruck können sein:

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

Anhaltender Stress kann auch zu dauerhaften Herz/Kreislauf- und Nierenerkrankungen, Stoffwechselstörungen, Allergien oder Entzündungskrankheiten führen. [3] Gesundheit ist ein sehr wichtiger Faktor, deswegen benötigen gestresste Menschen eine Möglichkeit sich wieder entspannen zu können.

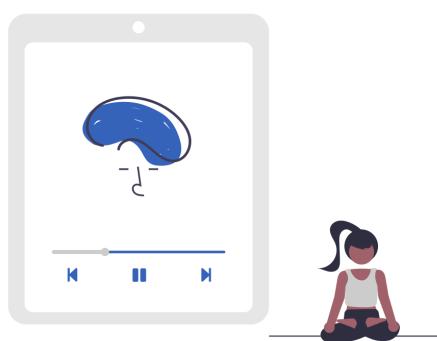


Abbildung 3: Musik dient zur Entspannung

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.

Ziele 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.[4]



Abbildung 4

Ziele zu setzen ist wichtig, deswegen müssen sie von Anfang an **richtig** gesetzt werden. Beim Setzen von Zielen kann man sich an folgenden Punkten orientieren: [4]

- 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, Freund:innen oder Kolleg:innen steigert die Motivation und erzeugt Verbindlichkeit, da man es anderen beweisen möchte.
- Immer positiv bleiben! Positive Formulierungen der Ziele motivieren langfristig mehr.
- 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 von Zielen sollten sie stets in der Gegenwart formuliert werden, damit man sich gleich Bilder im Kopf vorstellen kann. Das steigert außerdem die Motivation.

3.1 Projektziele

Bei den Projektzielen ist es wichtig sich nicht zu große Ziele vorzunehmen und nicht nur das eine Ziel im Auge zu haben. Es ist wichtig ein großes Ziel aufzuteilen, sodass viele kurzfristige Ziele zu einem langfristigen Ziel führen können. Langfristige Ziele bilden dabei auch die Leistungswirkung.

Relaxoon wurde in jeweils zweiwöchigen Sprints entwickelt. Es wurden am Anfang jedes Sprints einige kleine/kurzfristige Ziele definiert, die in den darauffolgenden Wochen erreicht werden sollten. Am Ende des Sprints wurde dann der damalige Stand des Projektes mit den Zielen verglichen, um zu wissen, ob alle Ziele erreicht wurden und welche Ziele für den nächsten Sprint vorgesehen waren.

Übersicht der wichtigsten kurzfristigen Ziele pro Sprint:

- Sprint 0: Wahl des CMS zwischen Strapi und Wordpress
- Sprint 1: Strapi- sowie App-Setup fertigstellen, Beginn UI-Design der App
- Sprint 2: Zugriff von Frontend auf Backend möglich
- Sprint 3: Beginn Implementierung des Category-Screens
- Sprint 4: Datenmodell von Relaxoon mit Strapi abgleichen
- Sprint 5: Implementierung der Datenstruktur in Strapi
- Sprint 6: UI-Design in der App umsetzen, Entwicklung Suchleiste
- Sprint 7: Echte Daten des Backends im Frontend anzeigen
- Sprint 8: Implementierung des Registrierungsformulars und das Setzen von Favoriten
- Sprint 9: Anmeldung in der App ermöglichen, Umsetzung Light und Dark Mode
- Sprint 10: Auto-Login, Tags für Medien hinzufügen

Faktoren wie Zeit, Kosten, Nutzen und Qualität sollten ebenfalls bei jedem Ziel berücksichtigt werden. Man sollte darauf achten, dass sich diese Faktoren gut miteinander vereinbaren lassen:

- Zeit: Kann das Ziel in der geplanten Zeit erreicht werden?
- Kosten: Welche Kosten werden bei der Umsetzung anfallen?
- Nutzen: Welchen Nutzen hat dieses Ziel? Für welchen Zweck wird es gebraucht?
- Qualität: In welcher Qualität soll das Ziel umgesetzt werden?

3.1.1 Leistungswirkungen

Leistungswirkungen eines Projektes beschreiben nicht das Ziel an sich, sondern was aus den Zielen folgt. Die Entspannungsapp Relaxoon soll bei den User:innen der App für geringen Stress und eine hohe Stressreduktion sorgen, was die Personen produktiver und glücklicher macht. Eine langfristige Erhaltung oder Verbesserung der Gesundheit wäre eine ideale Leistungswirkung.

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

4 Aufgabenstellung

Die Firma MACOLUTION hat Kontakte zu Streaming- und Gesundheitsexperten. Diese Fachleute erstellen viele Videos, Übungen und Artikel zu diesen Themen. Der Auftraggeber will eine Applikation für diese Kunden entwickeln, die folgende Kriterien erfüllt:

- User:innen müssen sich in der App nicht um die eigene Erstellung von Entspannungsmedien wie Texten oder Videos kümmern, weil diese Medien bereitgestellt werden, welche im Backend eingepflegt sind.
- Medien sollen individuell favorisierbar und mehreren Kategorien zugewiesen sein, die per Suchleiste in der App einfach zu finden sind.

Das Minimum Viable Product (MVP) von Relaxoon soll realisiert werden, damit möglichst früh Feedback von Nutzer:innen eingeholt werden kann, welches für die Weiterentwicklung und für die Verbesserung der App essenziell ist.

Dies bestätigt der deutsche Content-Autor Philipp Steubel, der sich im amerikanischen Softwareunternehmen Asana auf die Bereiche Marketing und Projektmanagement spezialisiert hat: „Ein MVP ist sicherlich ein gutes Hilfsmittel, wenn es darum geht, den Entwicklungsprozess an den Kundenbedürfnissen zu orientieren. Immerhin bekommen Sie so schon während der Entwicklung des Produktes wertvolles Kundenfeedback. Somit erfahren Sie aber auch mehr über die Zielgruppe selbst, welche Personengruppen von dem Produkt bzw. dem Service überzeugt sind und wie Sie die Marketingkampagnen aufsetzen können.“ [5]

Wichtig dabei ist:

- 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.

4.1 Use-Case-Diagram (UCD)

Für Relaxoon gibt es im Wesentlichen zwei Hauptakteure: der/die Content-Manager:in und die Benutzer:innen der App. Diese Rollen interagieren mit verschiedenen Funktionen innerhalb der Anwendung, wie folgendermaßen dargestellt. Um die Übersichtlichkeit zu verbessern, wurde das UCD in zwei Teile aufgeteilt.

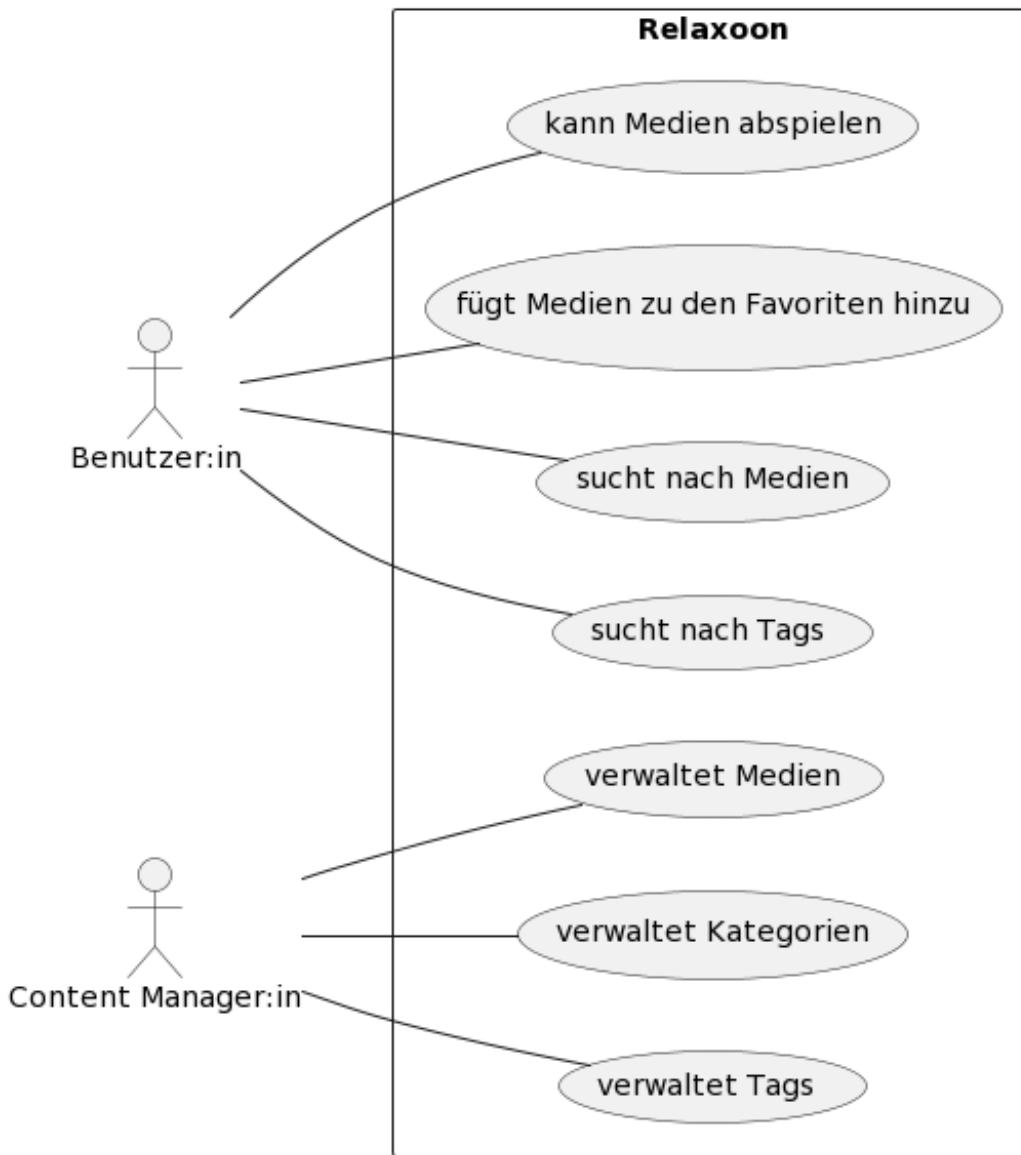


Abbildung 5: Use-Case-Diagram 1

Benutzer:in

- kann Medien abspielen: kann vorhandene Medien in der Anwendung anzeigen, um sie anzusehen.
- fügt Medien zu den Favoriten hinzu: hat die Möglichkeit, bestimmte Medien als Favoriten zu markieren, um schnell auf sie zugreifen zu können.

- sucht nach Medien: kann in der Anwendung nach bestimmten Medien suchen, indem Suchkriterien eingegeben werden.
- sucht nach Tags: kann nach Medien suchen, indem nach bestimmten Tags oder Kategorien gefiltert wird, um gezielt Inhalte zu finden.

Content Manager:in

- verwaltet Medien: kann Medien verwalten
- verwaltet Kategorien: kann Kategorien verwalten
- verwaltet Tags: kann Tags verwalten

Die Verwaltung der einzelnen Elemente wird im nächsten Diagramm genauer beschrieben.

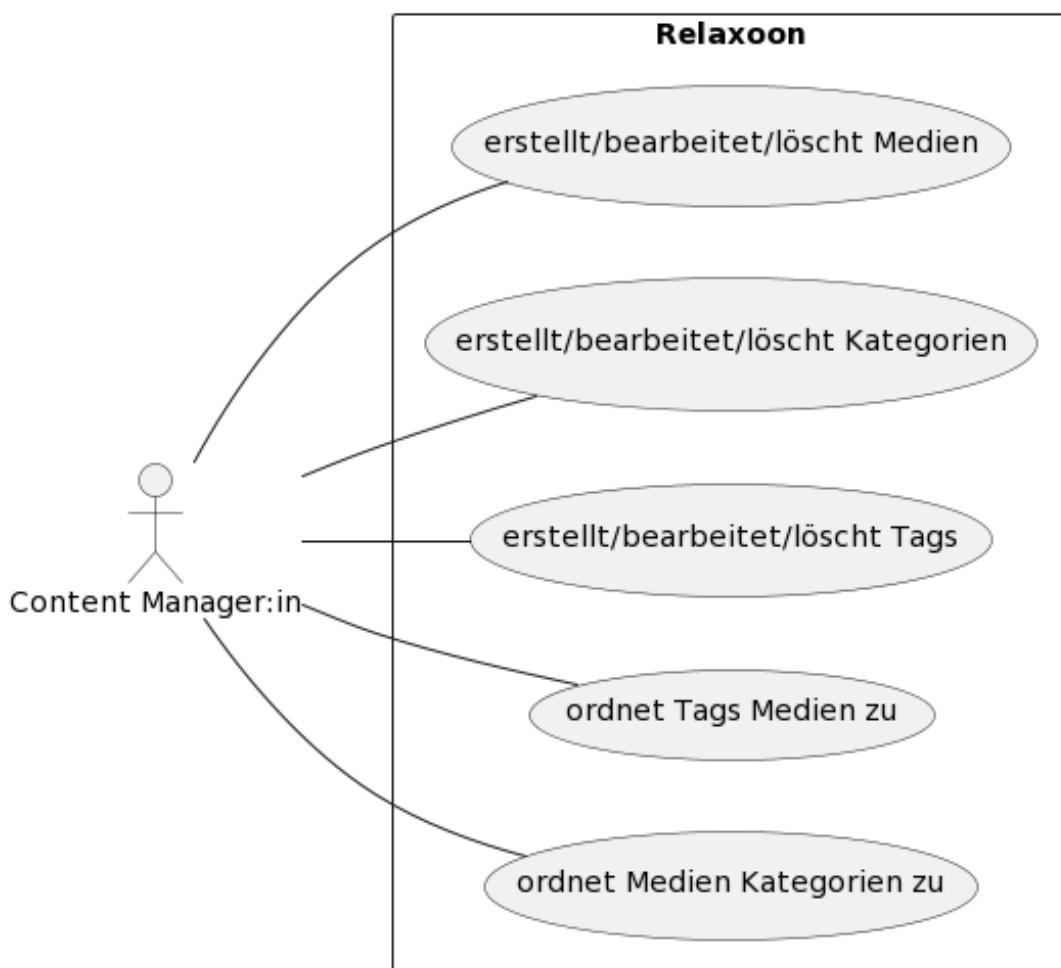


Abbildung 6: Use-Case-Diagram 2

Content Manager:in

- erstellt/bearbeitet/löscht Medien: kann neue Medien in die Anwendung hochladen, vorhandene Medien bearbeiten und nicht mehr benötigte Medien entfernen.
- erstellt/bearbeitet/löscht Kategorien: kann neue Kategorien erstellen, bestehende Kategorien bearbeiten und überflüssige Kategorien löschen.
- erstellt/bearbeitet/löscht Tags: kann neue Tags erstellen, vorhandene Tags bearbeiten und ungenutzte Tags entfernen.
- ordnet Tags Medien zu: kann Tags bestimmten Medien zuweisen, um ihre Ver-
schlagwortung und Klassifizierung zu optimieren.
- ordnet Medien Kategorien zu: kann Medien bestimmten Kategorien zuordnen,
um ihre Einordnung und Auffindbarkeit in der Anwendung zu verbessern.

5 Marktanalyse

„Marktanalysen sind die Grundlage für wichtige strategische Geschäftsentscheidungen. Nicht nur Standortentscheidungen und die Wahl von Marketingmaßnahmen hängen davon ab, sie können – je nach Schwerpunkt – auch ausschlaggebende Informationen zur Preispolitik, Expansionsplänen und Produktentwicklung beisteuern. So helfen sie etwa dabei, dass ein Markteintritt oder die Neueinführung eines Produktes gelingt oder ganz neue Märkte erschlossen werden können.“ [6]

5.1 Vorgehensweise

In einem ersten Schritt wird der Markt und die Zielgruppe definiert. Ist die App global verfügbar oder nur in bestimmten Regionen? Wer sind die potenziellen Benutzer:innen? Wie viel Interesse besteht an der App von den Kund:innen?

In einem nächsten Schritt muss der Wettbewerb untersucht werden. Welche Entspannungsapps sind auf dem Markt, welche davon sind am erfolgreichsten und warum? Dazu kommt noch eine Bewertung der aktuellen Trends im Bereich Gesundheit und Wohlbefinden. Gibt es aktuelle oder neue Technologien oder Forschungsergebnisse, die in die App integriert werden könnten?

Danach werden verschiedene Geschäftsmodelle in Erwägung gezogen. Soll die App kostenlos sein und durch Anzeigen oder In-App-Käufe finanziert werden? Oder wird es eine kostenpflichtige Premium-Version geben, mit der man mehr Funktionen innerhalb der App freischaltet?

Schließlich muss eine Marketingstrategie entwickelt werden, um die Entspannungsapp bekannt zu machen und in Umlauf zu bringen. Welche Kanäle sind am besten dazu geeignet, die gewünschten Zielgruppen zu erreichen?

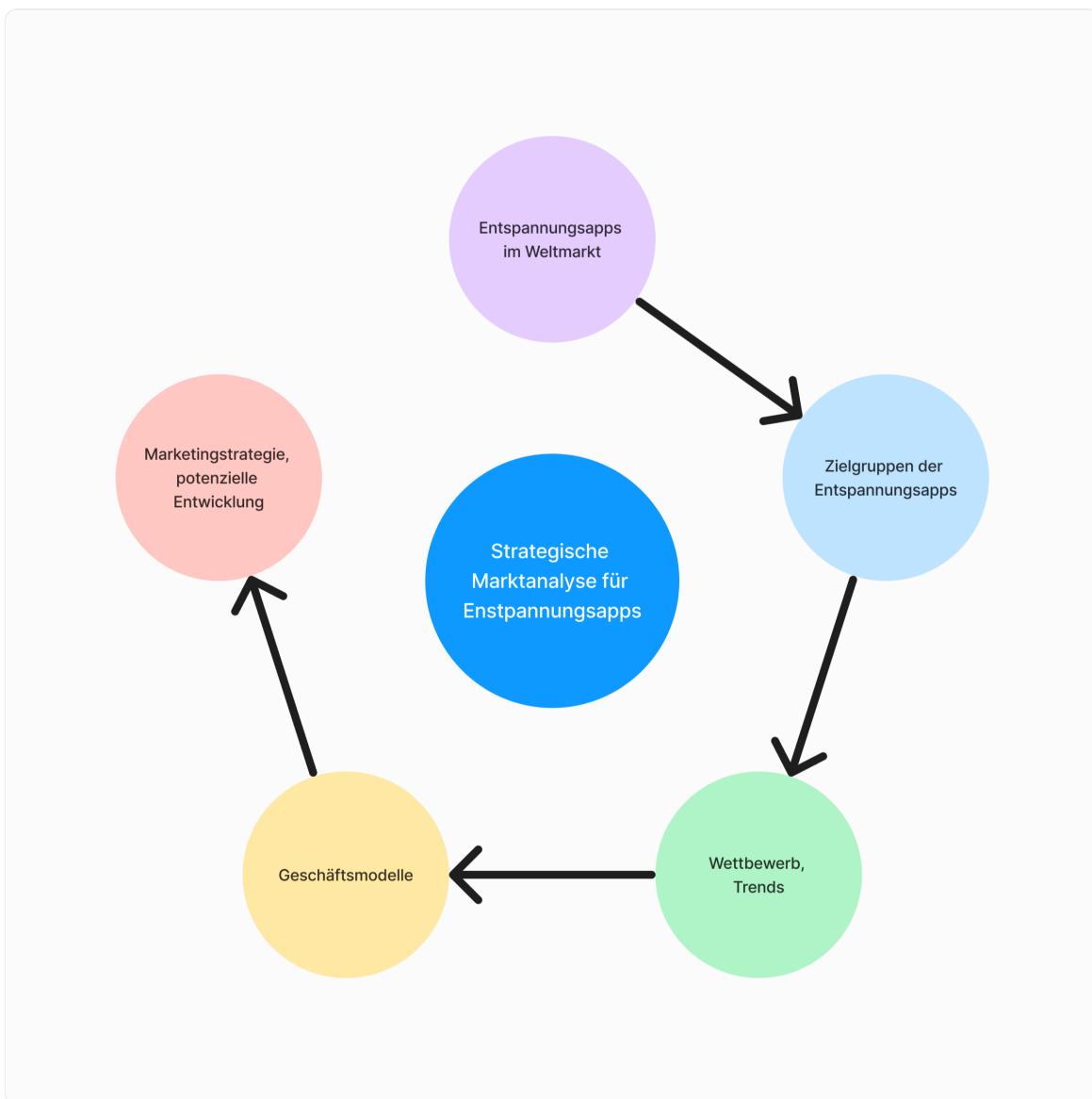


Abbildung 7: Ablauf

Durch diese Analyse erhält man wichtige Erkenntnisse über die aktuelle Situation sowie potenzielle Entwicklungen auf dem Markt für Entspannungsapps.

5.2 Analyse

Wie schon bei der Planung des Projektes formuliert wurde, soll die App Relaxoon, welche Stress bei bedürftigen Personen reduziert, im Play Store und im App Store international verfügbar sein. Der Markt bei Entspannungsapps ist sehr groß, was zu hoher Konkurrenz führt.

5.2.1 Wettbewerb und Trends

Einerseits gibt es viele Content-Creator, die ihre selbst erstellten Entspannungsmedien und -übungen, ohne die Verwendung einer konkreten Entspannungsapp, auf den bekanntesten und auch größten Plattformen wie zum Beispiel YouTube oder Spotify direkt hochladen. Das könnte bei einigen Menschen dazu führen, sich gar keine App installieren zu wollen, weil es unnötig erscheint. Andererseits bieten Apps, die extra auf Entspannung konzipiert sind, massive Vorteile.

Einige markante Features von Entspannungsapps sind:

- Kurse, die eine Reihe von Entspannungsübungen in gezielter Reihenfolge bereitstellen
- Viele verschiedene Funktionen zum Abspielen von beruhigenden Geräuschen und Musik
- Anleitungen zur Meditation, welche ebenfalls als Einschlafhilfe dienen und für einen besseren Schlaf sorgen

Bei Entspannungsapps gibt es auch verschiedene Arten. Zum einen welche, die sich ausschließlich auf Meditation spezialisiert haben, wohingegen andere nur audiovisuelle Inhalte zur Verfügung stellen. Es gibt ebenfalls Apps auf dem Markt, die ausschließlich Spiele innerhalb der App zur Entspannung implementiert haben.

Die Funktion von spielerischem Entspannen soll in Relaxoon nicht enthalten sein. Relaxoon soll die Möglichkeit bieten, alle Bedürfnisse der User:innen abzudecken, wobei man sich persönlich als User:in auch dafür entscheiden kann, nach bestimmten Vorgaben zu filtern.

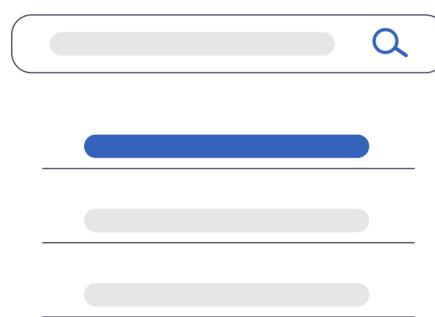


Abbildung 8

Man kann sich also dafür entscheiden, einen Text zum Entspannen zu lesen, ruhige Musik zu hören, sich ein entspannendes Video anzusehen, oder eine eigene Übung zu machen, in der mehrere Sinne angesprochen werden.

5.2.2 Geschäftsmodelle

Bei den Geschäftsmodellen gibt es, wie bereits oben erwähnt, für eine Entspannungsapp verschiedene Varianten.

Werbung

Das Prinzip von Werbung in der App wäre die erste Möglichkeit mit dem Projekt Geld zu verdienen. Es würde jedoch den Zweck, dass man sich bei der Verwendung der App entspannen soll, sehr stark negativ beeinflussen. Wenn man sich zum Beispiel vor jeder Übung ein 30-sekündiges Werbevideo anschauen muss, oder wenn Pop-Up-Fenster auftauchen würden, die der Nutzer oder die Nutzerin ständig wegtippen müssten, um fortfahren zu können, wäre das kontraproduktiv. Das würde sehr viele Leute abschrecken und darin resultieren, dass die App häufig direkt wieder deinstalliert wird.

Einzelne In-App-Käufe

Das Prinzip von In-App-Käufen ist sehr viel ansprechender. Es behindert die Nutzer:innen in keiner Weise daran, die App ohne Unterbrechungen zu verwenden. Man könnte In-App-Käufe den Kund:innen so anbieten, dass sie zum Beispiel für eine kleine Summe eine Übung freischalten können, oder auch für eine höhere Summe gleich zehn komplette Übungen auf einmal erwerben können.

Premium Version

Die wahrscheinlich geeignetste Lösung für Monetarisierung ist eine Premium Version für Relaxoon zu verkaufen, mit der man die gesamten Features der App benutzen kann. Diese könnte man einmalig bezahlbar machen, sodass die User:innen für immer Zugriff auf Premium Inhalte bekommen, oder auch monatlich verrechnen, dass jeden Monat wie bei einem Abonnement gezahlt werden muss.

Es wichtig zu erwähnen, dass man die App auch schon im Play- und im App Store käuflich machen könnte. Das würde jedoch die Verbreitung und Reichweite der App einschränken, weil sich durch den Preis, der schon vorab zu bezahlen wäre, weniger Leute die App tatsächlich kaufen und dann erst herunterladen würden. Eine von Grund auf kostenlose App passt deswegen besser.

Marketingstrategie

Im Online- und Web-Marketing steht die Steigerung der Sichtbarkeit und die Umwandlung von Besuchern in zahlende Kunden im Vordergrund. Es wird oft auf Strategien wie Suchmaschinenoptimierung (SEO), Content-Marketing und Social-Media-Werbung gesetzt wird, um eine breite Online-Sichtbarkeit zu erreichen. Im Gegensatz dazu legt das App-Marketing den Fokus auf Aspekte wie App Store-Optimierung (ASO), gezielte Werbekampagnen innerhalb von App-Plattformen und die Schaffung einer positiven Nutzererfahrung, um sowohl die Anzahl der Downloads als auch das langfristige Engagement der Nutzer zu steigern. [7]

Ein Problem, das häufig dabei entsteht ist, dass Personen eine App oft gar nicht lange installiert haben und oft diese schon nach nur einmaligem Nutzen wieder deinstallieren.

Auf der Website absatzwirtschaft.de wurde das noch genauer erläutert: [8]

„App-Marketer haben weniger als sechs Tage Zeit, um das Interesse von Nutzern erneut zu wecken: Der durchschnittliche Nutzer wartet zwischen dem letzten Öffnen der App und der Deinstallation knapp sechs Tage. Dabei variiert die Zeitspanne stark zwischen den einzelnen App-Kategorien – von weniger als drei Stunden bis zu etwa 15 Tagen. Entertainment- und Lifestyle-Apps werden besonders schnell verworfen, ihre Nutzer deinstallieren Apps im Durchschnitt nach einem halben bis ganzen Tag. E-Commerce- und Reise-Apps hingegen haben eine wesentlich längere Lebensdauer und werden in der Regel zehn bis elf Tage nach der letzten Session eines Nutzers gelöscht.“

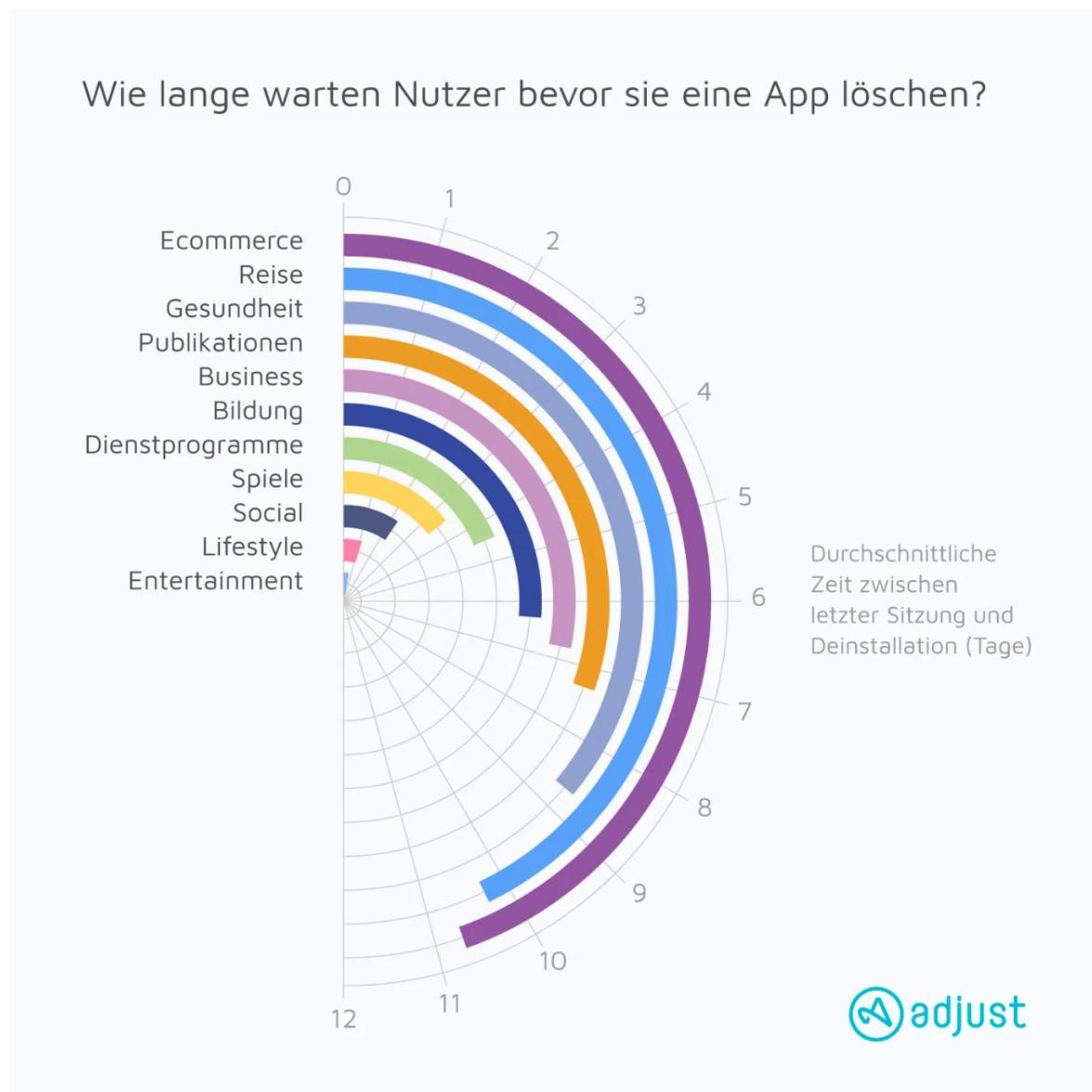


Abbildung 9: App Statistik [8]

6 Projektanforderungen

Beim Projektmanagement wird grundsätzlich zwischen zwei Hauptkategorien von Anforderungen unterschieden.

6.1 Funktionale Anforderungen:

- beschreiben, welche Funktionen das System bereitstellen soll und welche Aufgaben es ausführen muss.
- definieren die spezifischen Funktionen, Dienste oder Aufgaben, die das System erfüllen muss, um die Benutzeranforderungen zu erfüllen.
- konzentrieren sich auf die Was-Aspekte des Systems und beschreiben, welche Ergebnisse erwartet werden.
- Beispiel: „Das System muss dem Benutzer ermöglichen, sich mit einem Benutzernamen und Passwort anzumelden.“

6.2 Nicht funktionale Anforderungen:

- beschreiben Eigenschaften des Systems, die nicht unbedingt Funktionen sind, sondern Qualitätsmerkmale, die das System erfüllen muss.
- betreffen oft die Wie-Aspekte des Systems und definieren Qualitätsmerkmale wie Leistung, Sicherheit, Benutzerfreundlichkeit und Zuverlässigkeit.
- berücksichtigen Aspekte wie Skalierbarkeit, Wartbarkeit, Zuverlässigkeit, Leistung und Benutzerfreundlichkeit.
- Beispiel: „Das System muss eine Antwortzeit von weniger als 2 Sekunden für Benutzeranfragen sicherstellen.“

Die oberen zwei Abschnitte 6.1 und 6.2 wurden von einem KI-Modell generiert.[9]

Bei Relaxoon können alle Anforderungen auch in funktionale und nicht funktionale Anforderungen unterteilt werden. Eine nicht funktionale Anforderung wäre zum Beispiel die einfache Bedienung der App. Der User oder die Userin soll sich in der App problemlos zurechtfinden können, ohne dafür ein Manual zu benötigen. Dazu braucht man:

- eine Erklärung am Anfang bzw. ein Tutorial, wenn man die App das erste Mal startet, um überhaupt zu wissen, wie die App funktioniert und wie sie zu verwenden ist.
- ein schlichtes und übersichtliches Registrierungsformular mit Begründung dazu, damit der/die User:in weiß, warum er sich registrieren muss.
- eine Homepage, auf der man sich leicht zurechtfinden kann und die eine Übersicht über populäre Medien bietet, die bereits hochgeladen worden sind.
- eine Navigationsleiste, mit der man einfach zu anderen Seiten oder Features der App weitergeleitet wird.
- einen klaren Weg, um selbst personalisierte Einstellungen in der App treffen zu können.

Auf der anderen Seite wäre ein Beispiel für eine funktionale Anforderung die Möglichkeit, sich wieder einloggen zu können, wenn man sich bereits schon einmal registriert hat, damit ein zu lange dauernder Login die User:innen nicht verleitet, die App direkt wieder zu verlassen. Das kann man mit einer „Login Daten merken“-Option umsetzen oder man verwendet eine „Ich besitze bereits einen Account“-Option bei User:innen, die sich die App auf einem zusätzlichen Endgerät heruntergeladen haben. Eine weitere Projektanforderung für Relaxoon ist die **Übertragbarkeit** d.h. die App für mehrere Betriebssysteme nutzbar machen. Wenn eine App übertragbar ist, schafft das einige Vorteile:

- erhöhte Zielgruppenabdeckung
- keine Vorgabe des Mobilgerätetyps
- Wettbewerbsvorteil gegenüber anderen

7 Technologien

7.1 Strapi

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

Durch den Einsatz von Strapi sind Inhalte und dafür notwendige technische Funktionalitäten sehr einfach erstellbar. [10]

Welche eingebauten Funktionalitäten bietet Strapi für Entwickler und Content-Manager?

Für die Entwickler werden folgende Funktionalitäten zur Verfügung gestellt:

- fertige REST-Schnittstellen
- GraphQL Schnittstellen
- eine Benutzeroberfläche, wo die Struktur der Datenbank editiert werden kann
- eine eingebaute JWT Authentifizierung
- File Upload
- Query Engine API
- Entity Service API
- Plugin für Internationalisierung
- Plugin Store, mit dem sehr viele hilfreiche Plugins wie zum Beispiel OpenAPI installieren werden können
- Autorisierung und Rollenmanagement

7.1.1 REST-Schnittstellen

Für jedes Modell werden die benötigten GET, POST, PUT und DELETE REST-Schnittstellen standardmäßig erstellt. Jeder Endpoint unterstützt auch eine Menge von unterschiedlichen Parametern, welche den Aufbau des Responsebodys steuern können wie zum Beispiel.: filters, populate und fields. [11]

7.1.2 Definition von Datenstruktur und Datenmodell

Mit der Verwendung dieser Oberfläche kann man folgende Elemente erstellen:

- Collection Types
- Single Types
- Components

Collection Types

”Collection types are content-types that can manage several entries.” [12]

Mit der Verwendung von dieser Collection Types kann man die Struktur einer Tabelle bzw. einer Collection definieren. Allerdings entspricht die Struktur der definierten SQL-Tabelle oder NoSQL-Collection nicht eins zu eins dem Collection Type. Die speziellen Datentypen wie Files oder Custom Components werden nicht in der Datenbank als eine Tabellenspalte gespeichert. Stattdessen werden die speziellen Typen in einer anderen Tabelle gespeichert.

Folgende Datentypen werden von Strapi zur Verfügung gestellt:

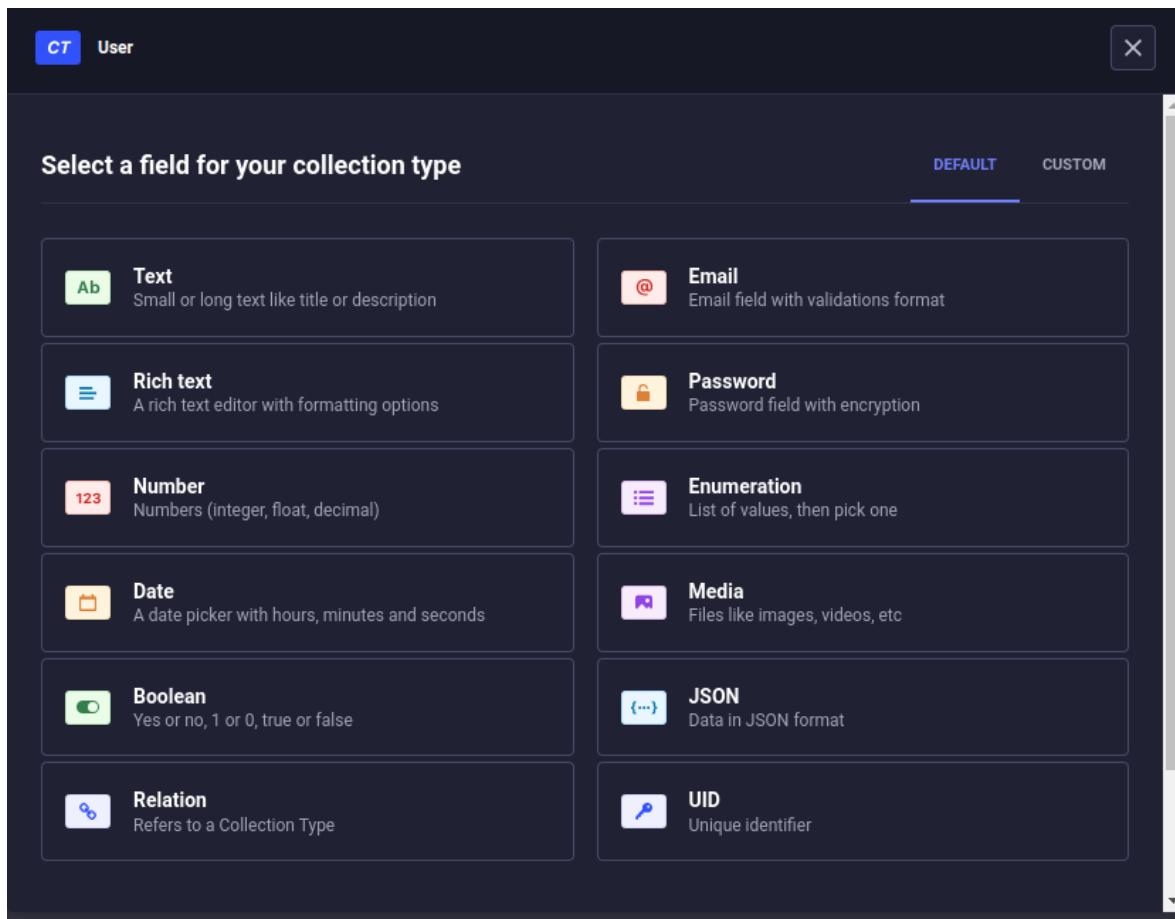


Abbildung 10: Datentypen

Die Relationen zu anderen "Collection Types" werden von Strapi als ein Datentyp interpretiert. Dieser unterstützt die Standardbeziehungen einer relationalen Datenbank zusätzlich zu bidirektionalen Beziehungen. Bei den "Many to Many" Beziehungen ist die Erstellung eines zusätzlichen Collection Types, welcher als eine Assoziationsstabelle dient, gar nicht nötig. Die Assoziationsstabellen werden von Strapi im Hintergrund generiert.



Abbildung 11: Relationen

Zusätzlich zur Erstellung von Spalten haben Collection Types auch eine eingebaute Validierung, die man aktivieren kann.

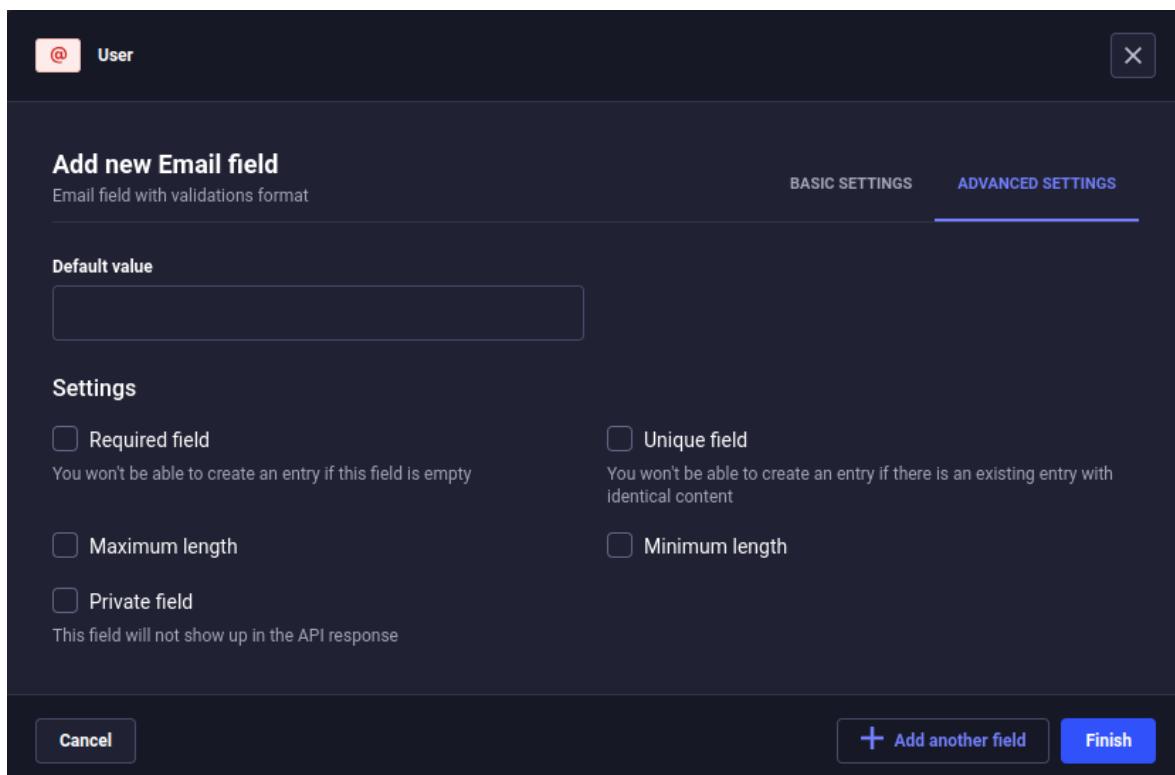


Abbildung 12: validation

Single Types

”Single types are content-types that can only manage one entry” [12]

Components

”Components are a data structure that can be used in multiple collection types and single types.” [12]

Es ist anzumerken, dass die Änderung der Struktur der Datenbank nur im ”Dev Mode” funktionieren kann. In der produktiven Umgebung ist die Änderung der Tabellenstruktur nicht möglich.

7.1.3 Query Engine API

”The Strapi backend provides a Query Engine API to interact with the database layer at a lower level. The Query Engine API should mostly be used by plugin developers and developers adding custom business logic to their applications.” [13]

Diese agiert als **Object Relational Mapper** (ORM) bzw. Query Builder, in der man SQL oder NoSQL Queries aus einem JS Code generieren kann. Die selektierte Spalten werden dann in JS-Objekten serialisiert.

7.1.4 Entity Service API

”The Strapi backend provides an Entity Service API, built on top of the Query Engine API. The Entity Service is the layer that handles Strapi’s complex data structures like components and dynamic zones, and uses the Query Engine API under the hood to execute database queries.”

Der Unterschied zwischen diesem Service und der Query Engine API liegt darin, dass die Entity Service API nicht nur die SQL oder NoSQL Struktur einer Tabelle bzw. einer Collection, sondern auch die besonderen Datentypen eines Strapi-Models wie zum Beispiel Files und Custom Components abfragen kann. [14]

7.1.5 File Upload

Dabei ist das Wichtige, dass man Strapi so konfigurieren kann, dass es die Uploads für mehrere Bildschirme anpassen kann.

Listing 1: file upload config in strapi

```
1  export default ({ env }) => ({
2    upload: {
3      config: {
4        breakpoints: {
```

```

5           xlarge: 1920,
6           large: 1000,
7           medium: 750,
8           small: 500,
9           xsmall: 64
10          },
11          },
12          },
13      );

```

Es gibt auch andere Standardeinstellungen für File Upload wie zum Beispiel die Bestimmung von der Uploadgröße. [15]

Für den Content-Manager werden folgende Funktionalitäten zur Verfügung gestellt:

- eine Oberfläche, bei der die Inhalte eingepflegt werden können
- eine Media Library

7.1.6 Eingabe von Inhalten

Strapi untersucht die Typen eines Collection Types und stellt dann dem Content-Manager die passenden Eingabefelder zur Verfügung. Der Content-Manager hat nicht nur die Möglichkeit Inhalte zu verwalten, sondern auch vorgefertigte Inhalte in das System einzugeben. Diese kann er dann später veröffentlichen. Falls ein Eintrag nicht veröffentlicht ist, wird dieser in der API nicht zurückgeschickt.

7.1.7 Media Library

Die Media Library ermöglicht die Verwaltung von Files und Assets. Es gibt dabei die Möglichkeit neue Assets hochzuladen, Unterordner zu erstellen und die Assets in Unterverzeichnissen zu speichern. Zusätzlich kann man Assets löschen, umbenennen oder zuschneiden. Man kann die Files in der Media Library auch beliebig sortieren und danach suchen. [16]

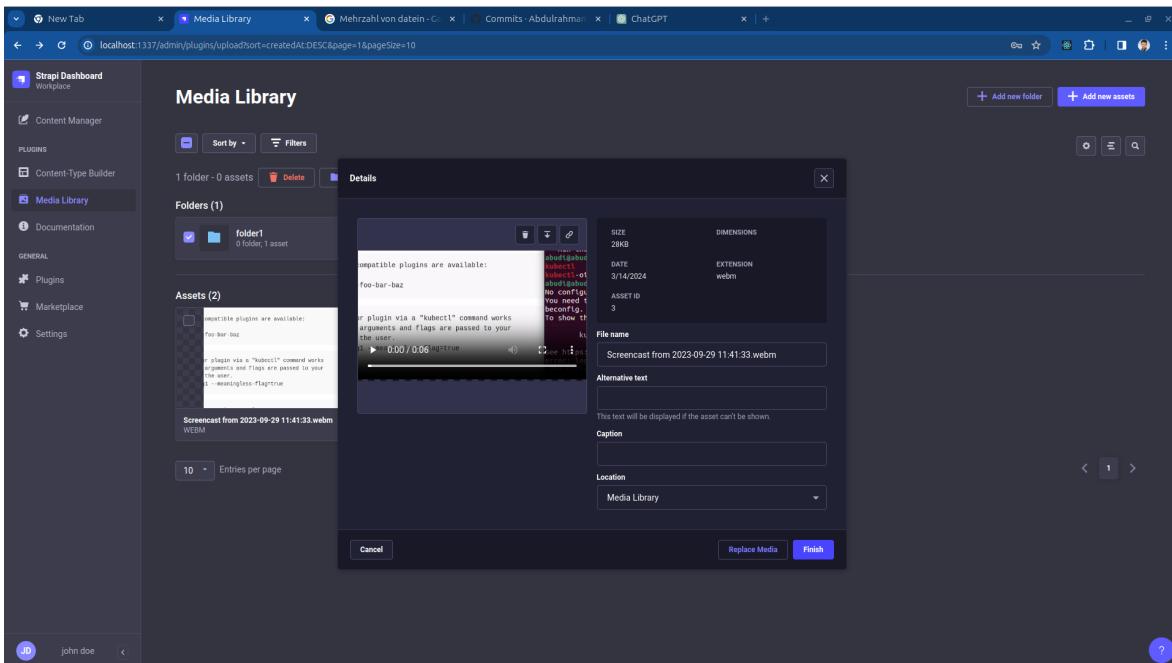


Abbildung 13: media-library

Es gibt auch die Möglichkeit, den Link einer Datei, wie zum Beispiel ein Video aus YouTube, in der Media Library einzubetten. Allerdings ist diese Funktion nur bei der aktuellsten Version leicht anwendbar. Bei älteren Versionen ist die Umkonfigurierung des Reverse Proxys des Systems notwendig. Sonstige Probleme mit lizenzzfreien Inhalten sind dann nicht vom System zu lösen, sondern vom Content-Manager. [17]

7.2 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.”[18]

Der Vorteil von Firebase App Distribution ist, dass man die Applikation auf dem eigenen Mobilgerät ausprobieren kann, ohne die App auf dem Play Store bzw. App Store hochladen zu müssen.

8 Systemarchitektur

8.1 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 läuft, 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.

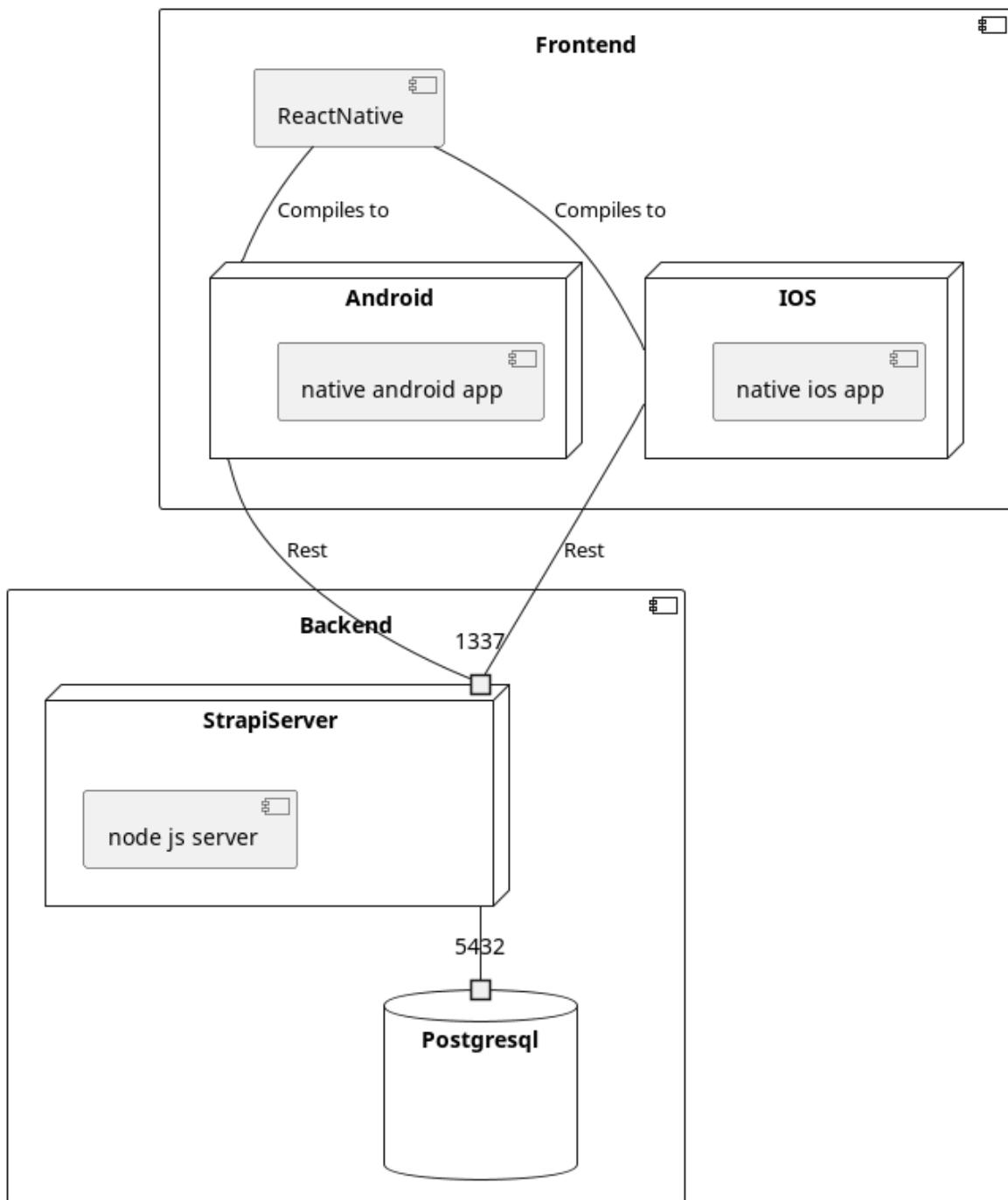


Abbildung 14: Systemarchitektur

9 Entwurfsentscheidungen

9.1 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
- Der Auftraggeber forderte eine Cross-Plattform-Lösung für das Frontend, um IOS-Android- Betriebssysteme abzudecken
- Endergebnis von React Native ist eine native App

9.1.1 Was ist eine Cross-Plattform-Lösung

”Eine Cross-Plattform 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.” [19]

9.1.2 Warum eine Cross-Plattform-Lösung angefordert wurde

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

9.1.3 Warum Relaxoon native und nicht hybrid wurde

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, [p]rogrammiert. 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.”[20]

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 Betriebssystemfunktionen interagieren und diese nutzen.”

[20]

Grundsätzlich sind native Applikationen viel schneller und barrierefreier als hybride Applikationen [20]

9.1.4 Alternativen für React Native

Kotlin Multiplatform

Kotlin MutliPlattform (KMP) bietet eine 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. ” [21] Allerdings ist das Problem dabei, dass KMP nur in der Beta-Version verfügbar ist und daher ist diese nicht stabil. Außerdem ist die Community dieser Alternative nicht so groß wie die Community von React Native. Gute Kenntnisse in Swift **User Interface (UI)** und **Android Software Development Kit (SDK)** werden für die Verwendung von Kotlin Multiplatform auch verlangt. Grund dafür ist, dass KMP keine Bibliotheken für UI-Elemente wie React Native oder Flutter bereitstellt, sondern die Syntax der OS-Technologie verwendet. [21]

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. [22] Das Problem bei Flutter ist, dass man gute Kenntnisse in der Programmiersprache

”Dart” haben muss. [22] Für die Entwicklung von React Native sind gute Kenntnisse in den Programmiersprachen Javascript und Typescript von Vorteil. Außerdem ist die Unterstützung von macOS, Linux und Windows nicht relevant für eine mobile App.

Xamarin

Xamarin ist ein C# Framework, das für die Entwicklung von nativen Cross-Plattform-Lösungen zuständig ist. Die Performance von Xamarin ist viel besser als die Performance von React Native. [23]

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.[23]

9.1.5 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%

9.2 Strapi

9.2.1 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.” [24]

9.2.2 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 CMS viel schneller als die Implementierung eines Backends.

9.2.3 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 „kopflos“ (englisch: „headless“) bezeichnet werden.“ [25]

Ein Beispiel für ein seit weitverbreitetes Headless-CMS ist Storyblok. Dieses bietet nicht nur eine REST-API, sondern auch ein Plugin, bei dem man die Ansicht des Benutzers editieren kann. [26,]

9.2.4 Warum wurde ein Headless-CMS verwendet

- Vorgabe des Auftraggebers
- Andere Arten von CMS generieren statische HTML Seiten, welche für Relaxoon gar nicht gebraucht werden, da Relaxoon eine mobile App ist.
- Die Firma solvistas beschäftigt sich intensiv mit Data Science. Da man auf das CMS mittels REST zugreifen kann, will solvistas in der Zukunft ein Datenzentrum aus diesem erstellen. Zusätzlich wird das CMS für Data Science Zwecke verwendet.

9.2.5 Warum Strapi und nicht Storyblok

Was ist Storyblok

”Storyblok ist ein Headless-CMS und die erste Wahl für ein modernes, responsives und flexibles Content-Management-System. Dieses CMS ermöglicht ein schnelles und einfaches [R]eagieren auf stetig wachsende Anforderungen. Außerdem können Inhalte leicht an unterschiedliche Endgeräte angepasst werden.“ [27]

Die Vorteile von Storyblok liegen darin, dass dieses nicht nur als REST-Server verwendet werden kann. Es gibt bei diesem die Möglichkeit, einige UI-Elemente anzupassen. Allerdings kann diese Funktion bei Relaxoon nicht angewendet werden, da Relaxoon eine Handyapplikation ist. Storyblok ist auch kostenpflichtig. Außerdem bietet Storyblok keinen interactiven Query Builder wie Strapi (Siehe 7.1.1). Bei Storyblok gibt es auch keinen ORM und daher ist das Anpassen von API-Responses kaum möglich. [26] Grundsätzlich war Strapi die richtige Entscheidung für die vorliegende Arbeit, da die API von diesem sehr flexibel ist. Außerdem ist Strapi kostenlos.

9.2.6 Warum Strapi und nicht Wordpress Application Interface (API)

Vorteile von Wordpress API

- viele Einstellungen und Features
- unterstützt Search Engine Optimization (SEO)
- guter Community Support
- einfach zu verwenden

[10]

Nachteile von Wordpress API

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

[10]

Vorteile von Strapi

Im Gegensatz zu Wordpress, kann man mit Strapi beliebig viele Plugins und Add-ons verwenden. Außerdem gibt es eine eingebaute Authentifizierung bzw. Autorisierungsfunktion zusätzlich zur Unterstützung von 20 unterschiedlichen Sprachen und REST bzw. Graphql APIs. [10]

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. [10]

Ergebnis

Alle erwähnten Pros und Kontras 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. [10]

9.2.7 Nutzwertanalyse für das Headless-CMS

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

9.3 Headless-CMS vs. selbst geschriebener Server

9.3.1 Vorteile eines Headless-CMS

Man ist mit einem Headless-CMS viel schneller als bei einem selbst geschriebenen Server, da alle **Create Read Update Delete (CRUD)** Operationen auf der Datenbank und auf der API Ebene automatisch generiert werden. Man braucht ein minimales technisches Wissen, um das Ganze zu bedienen. Der Umstieg auf eine andere Datenbank wie zum Beispiel von PostgreSQL auf Mongo oder umgekehrt oder auch auf eine andere Networking Technologie wie REST und Graphql erfolgt mit einer kleinen Änderung in den Konfigurationen des CMS. Bei einem REST-Server ist dieser Umstieg meistens kompliziert und könnte dazu führen, dass man den kompletten Codebase ändern muss, wenn man den Code nicht sauber strukturiert hat. Außerdem gibt es bei einem Headless-CMS eine fertige implementierte JWT Authentifizierung zusätzlich zu den fertigen Autorisierungsschnittstellen, File Upload und die Optimierung der hochgeladenen Bilder. Diese Art von CMS verfügt auch über eine Oberfläche, bei der die Inhalte sehr einfach eingespielt werden können.

9.3.2 Nachteile von einem Headless-CMS

Die Wartung ist sehr schwierig und speziell bei einem Update. Dabei muss man auch vieles ändern bzw. umkonfigurieren. Es ist auch meistens nicht zu leicht, das System zu erweitern und man kann nicht jede einzelne Funktionalität im System anpassen. Außerdem sind die Communities von vielen Headless-CMS sehr klein, da diese Art von CMS sehr modern ist.

9.3.3 Vorteile eines selbst geschriebenen Servers

Man kann das System beliebig erweitern und die Wartung von diesem ist viel leichter, wenn man die richtige Technologie verwendet. Ein selbst geschriebener Server kann auch sehr performant sein im Vergleich zu einem Headless-CMS, wenn man die richtige Technologie für das Entwickeln verwendet.

9.3.4 Nachteile eines selbst geschriebenen Servers

Man muss sich in mindestens einer Programmiersprache und einer Datenbank gut auskennen, um eine vernünftige API zu schreiben. Außerdem ist Erfahrung in der

Strukturierung einer API bzw. eines großen Codebases notwendig. Daher sind gute Kenntnisse bei den Themen Design Patterns, unterschiedliche Programmierparadigmen und unterschiedliche Guidelines wie z.B.: The Clean Architecture oder The Solid Principles sehr wichtig. Man muss sich auch an gewisse Scrum und Projektmanagement Regeln sowie REST-Guidelines oder Guidelines der verwendeten Programmiersprache halten.

Man kann nach diesem Vergleich sagen, dass man sich für einen REST-Server entscheiden soll, wenn es um ein wichtiges Projekt geht, das nicht ständig geändert, sondern nur erweitert werden soll. Man soll sich auch für diesen entscheiden, wenn die "Business Logic" bei einem höheren Komplexitätslevel liegt. Ein selbst geschriebenes System ist bei Usecases, wo Performance eine große Rolle spielt und auch bei Cloud basierten Anwendungen sehr empfehlenswert.

9.4 Component Library

Der Standardweg, um UI-Elemente in React Native zu stylen, ist JSS. Diese ist eine Technologie, bei der man Cascading Style Sheet (CSS) in Form von Javascript Objekten schreibt. Das Problem bei dieser Technologie liegt darin, dass viele CSS Attribute und Funktionen nicht inkludiert sind. Es gibt außerdem spezifische OS-Einstellungen für Barrierefreiheit wie zum Beispiel die Einstellungen für Screenreader, die man bei der Entwicklung immer wieder vergessen kann. Aus diesen Gründen hat sich das Team für die Verwendung von 'Native Base' entschieden.

9.5 Typescript

Im Gegensatz zu Angular ist die Verwendung von TypeScript bei React oder React Native optional. TypeScript wurde jedoch eingesetzt, um Typisierungsprobleme zu beheben. Dies ist besonders wichtig, da leichte Typisierungsfehler zu Problemen führen können, da die Betriebssysteme Android und IOS bei Typisierungsfehlern restriktiver als der Browser sind. Dies liegt daran, dass die transpilierten IOS- und Android-Applikationen auf statisch typisierten Sprachen, nämlich Kotlin und Swift, basieren. Die Verwendung von Typescript wird auch in der Dokumentation von React Native empfohlen. [28]

9.6 Warum Expo und nicht React Native CLI

9.6.1 Was ist Expo

”Simplified Development Experience Expo is a framework and platform for building React Native applications. It aims to provide a simplified development experience by abstracting away some of the complexities involved in setting up and configuring a React Native project.”[29]

Das Team hat sich in erster Linie für Expo entschieden, da diese CLI den Project Setup erleichtert. Der Grund dafür ist, dass man sich nicht über spezifische plattformabhängige Module kümmern soll. Expo bietet einige Komponenten und Libraries, welche von Mobilgeräten und manchmal von unterschiedlichen Browssern unterstützt werden. Außerdem ist es bei Expo auch möglich, dass die Applikation auf einem physikalischen Android und auch auf einem IOS Gerät gestartet werden kann. Ein Macos-Gerät ist für das Starten der Anwendung auf IOS-Geräte gar nicht nötig. [29]

10 Design

10.1 Mockups

Als Grundlage für die Entwicklung des Designs von Relaxoon standen Design-Mockups des Auftraggebers zur Verfügung.

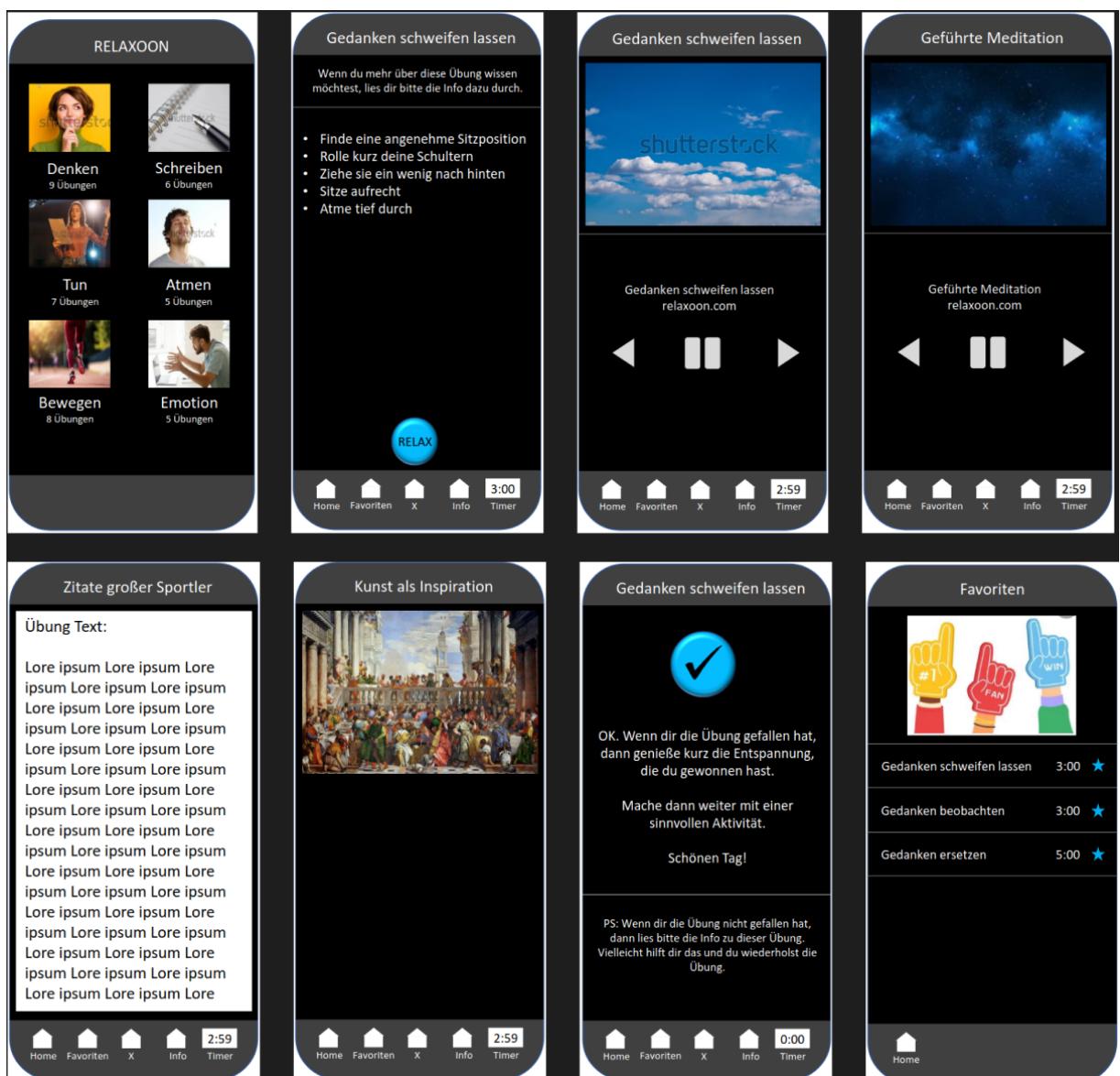


Abbildung 15: Mockups

Von oben links bis unten rechts auf dem Bild sind folgende Seiten abgebildet:

1. Home-Screen
2. Übung: Intro
3. Übung: Video
4. Übung: Ton
5. Übung: Text
6. Übung: Foto
7. Übung: Outro
8. Favoritenliste

10.2 Tutorial

Zu Beginn, wenn man die App zum ersten Mal öffnet, kommt man zu einem Tutorial. Dieses gibt den User:innen einen ersten Einblick in die App.

Im UI-Prototypen von Relaxoon wurde das Tutorial folgendermaßen realisiert:

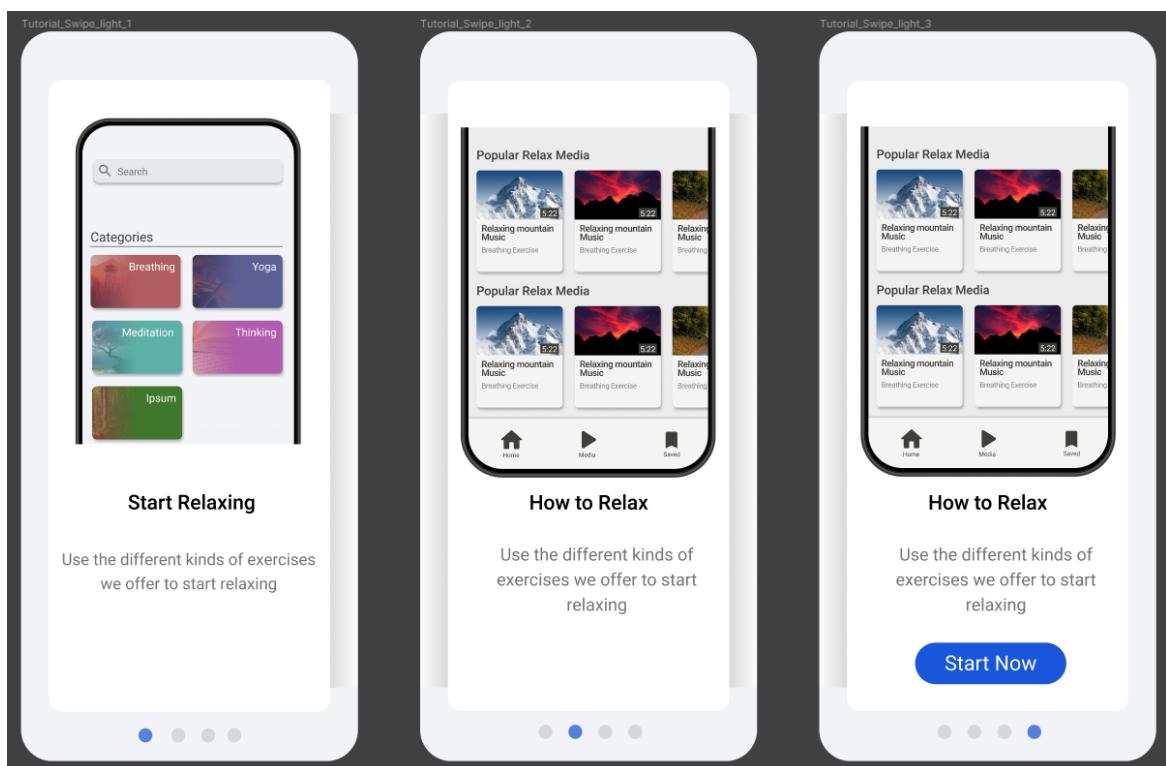
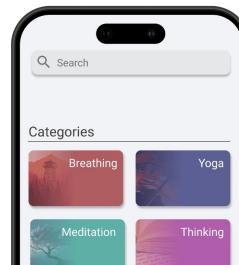


Abbildung 16: Tutorial im UI-Prototypen

Das Konzept wurde bei der Programmierung wie folgt verwirklicht:

RELAXOON



Herzlich willkommen bei Relaxoon!

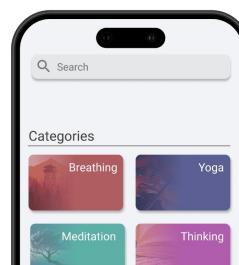
Die neue App für Entspannung und Stress-Abbau bringt dir mehr Wohlbefinden - 24x7 und wo immer du gerade bist!

Wische von rechts nach links, um fortzufahren.



Abbildung 17: Erste Tutorial-Page

RELAXOON



Entspannende Inhalte mit Qualität

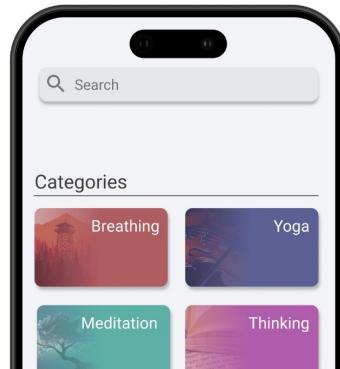
Unser Alltag fordert uns immer mehr und entzieht uns wertvolle Energie. Phasen der Entspannung sind nötig, damit wir wieder Energie aufladen.

Qualitätsgeprüfte Videos, Musik und Texte bringen Dir wertvolle Relaxoon-Minuten der Entspannung und helfen dabei den Alltag zu meistern.



Abbildung 18: Zweite Tutorial-Page

RELAXOON



Finde Ruhe mit Relaxoon-Übungen

Neben entspannenden Inhalten bietet dir Relaxoon Videos mit Übungen, mit denen du deine Ruhe wieder findest. Folge den Schritt-für-Schritt Anleitungen und du wirst schon bald entspannter sein. Viel Spaß!

Jetzt Starten



Abbildung 19: Dritte Tutorial-Page

Auf der letzten Tutorial-Page gelangt man über den "Jetzt Starten"-Button weiter zum Registrierungsformular.

10.3 Login- und Registrierungsformular

Das Registrierungsformular ist, wie unten dargestellt, nach folgendem Schema aufgebaut:

1. Username
2. E-Mail
3. Passwort

4. Passwort erneut eingeben

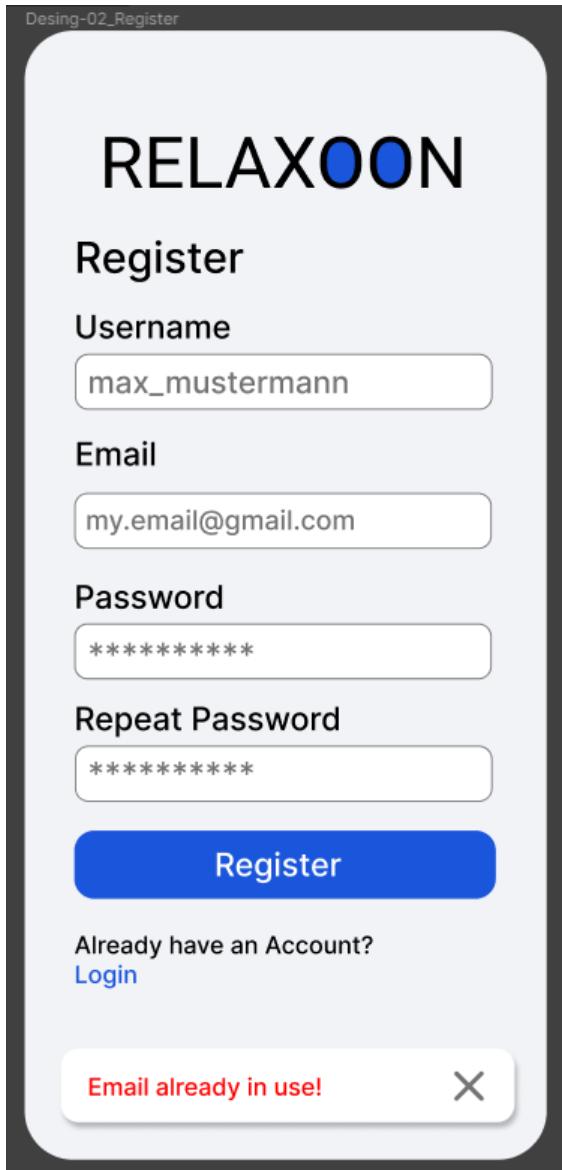


Abbildung 20: Registrierung UI-Prototyp

RELAXOON

Register

Username*

Email*

Password*

Repeat Password*

Register

Already have an account?

[Login](#)

Email already in use!

Abbildung 21: Registrierung in der App

Mit dem "Register"-Button kommt man zum Home-Screen, sofern der Username, die E-Mail und das Passwort korrekt validiert werden können.

Falls man bereits einen Account besitzt, kann man über das blaue "Login" zum Login-Screen navigieren.

Um sich einloggen zu können, muss man bei diesem Screen die bereits registrierte E-Mail und das dazugehörige Passwort eingeben und auf "Login" drücken. Damit gelangt man ebenfalls zum Home-Screen. Über das blaue "Create an Account" wird man zurück zur Registrierung geleitet, falls man sich einen neuen Account erstellen will.

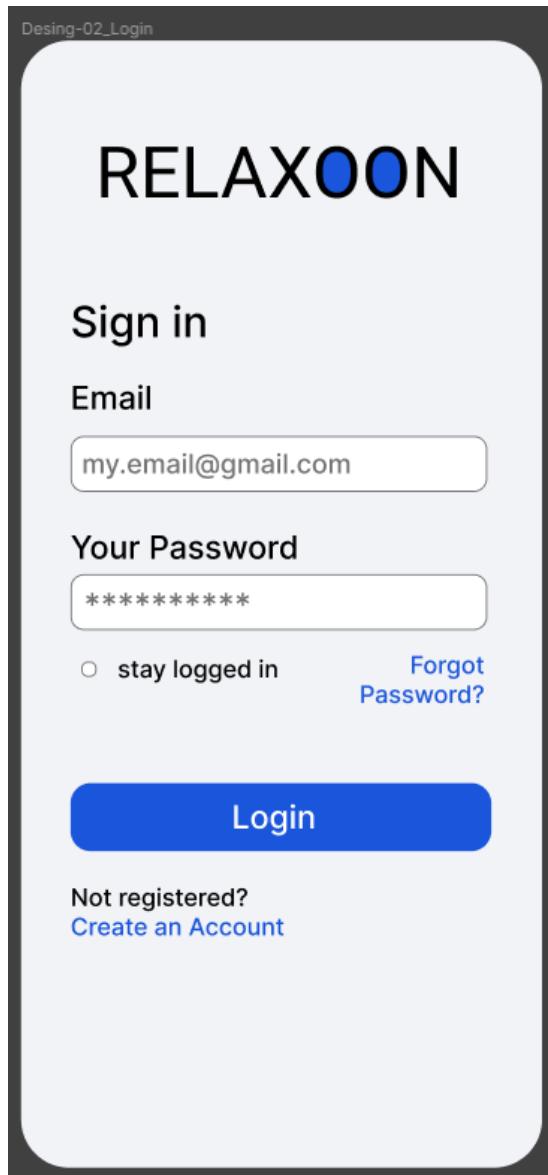


Abbildung 22: Login UI-Prototyp

RELAXOON

Sign in

Email*

Password*

Stay logged in
[Forgot Password?](#)

[Login](#)

Not registered?
[Create an Account](#)

Abbildung 23: Login in der App

Bei erfolgreichem Login wird der/die User:in darüber mit einer Nachricht informiert, die am unteren des Bildschirms aufscheint.

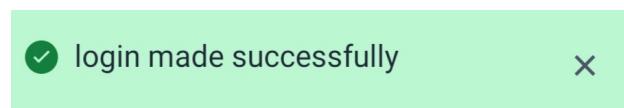


Abbildung 24: Bestätigungsbenachrichtigung

10.4 Home-Screen

Auf dem Home-Screen ist links oben ein Bereich zu sehen, in dem der zuvor eingegebene Username dargestellt wird. Durch Drücken auf das Symbol rechts oben gelangt man zu den Einstellungen. (siehe Kapitel 10.9)

Unterhalb davon werden eingepflegte Medien vom Backend mit Thumbnail, Titel und einer gekürzten Beschreibung angezeigt. Auf der Oberfläche ganz unten befindet sich die Navigationsleiste, mit der man zwischen den unterschiedlichen Screens navigieren kann.

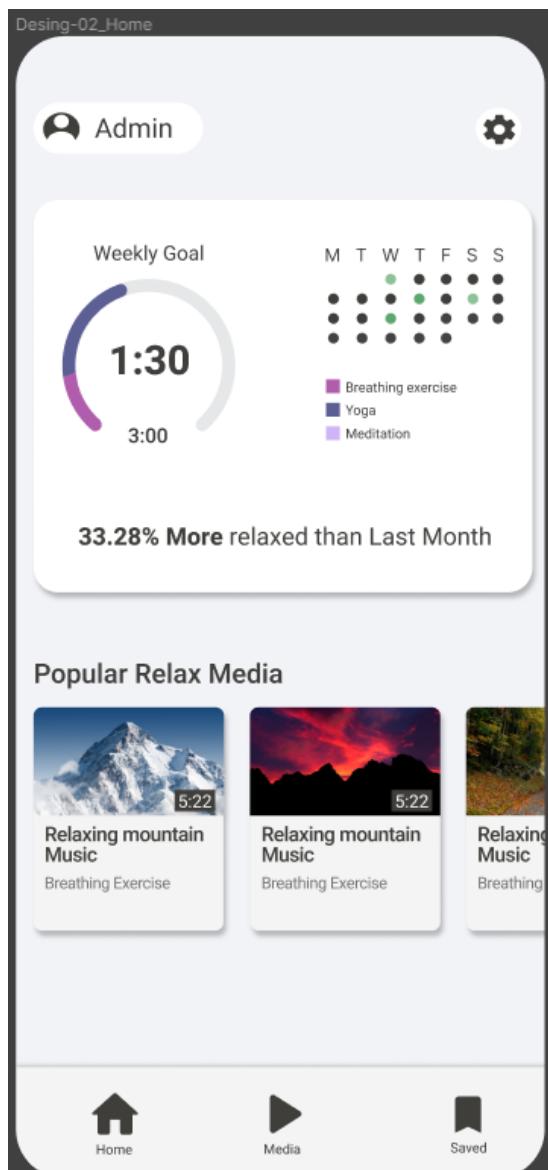


Abbildung 25: Home-Screen UI-Prototyp

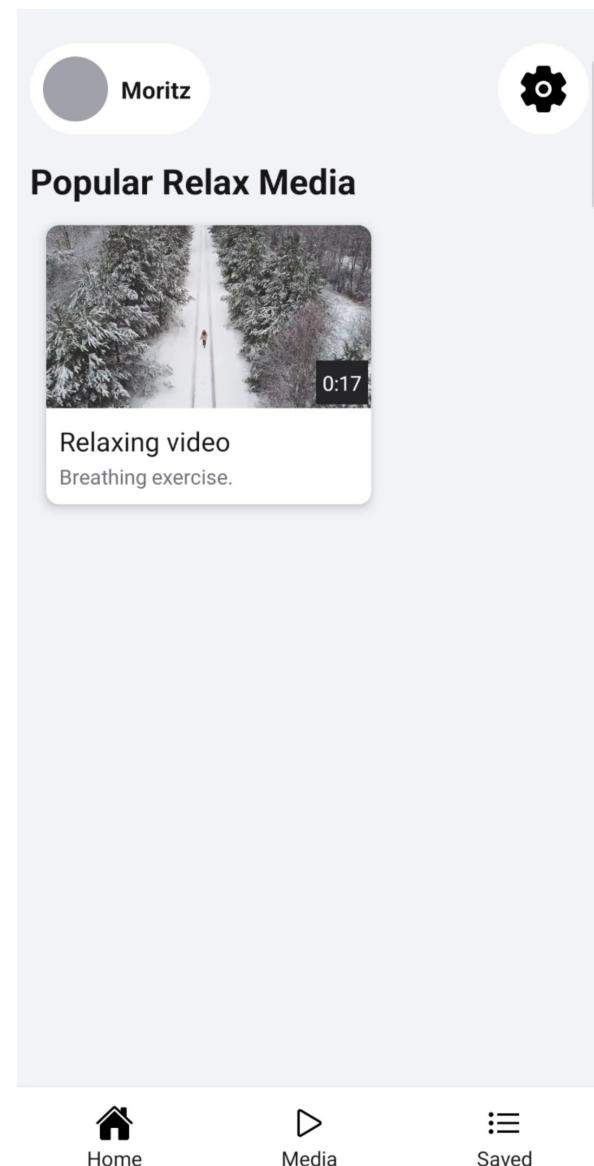


Abbildung 26: Home-Screen in der App

10.5 Kategorien

Wenn man in der Navigationsleiste unten auf "Media" tippt, wird man zu einer Übersicht von Kategorien weitergeleitet. Hier befindet sich am oberen Rand des Bildschirms die Suchleiste (siehe Kapitel 10.7) und darunter werden alle im Backend eingepflegten Kategorien angezeigt. Jede Kategorie hat eine Box, in der der Name der Kategorie und das Hintergrundbild zu sehen ist.

Durch das Auswählen einer Kategorie kommt man auf die Unterseite, in der die dazugehörigen Medien aufgelistet werden.

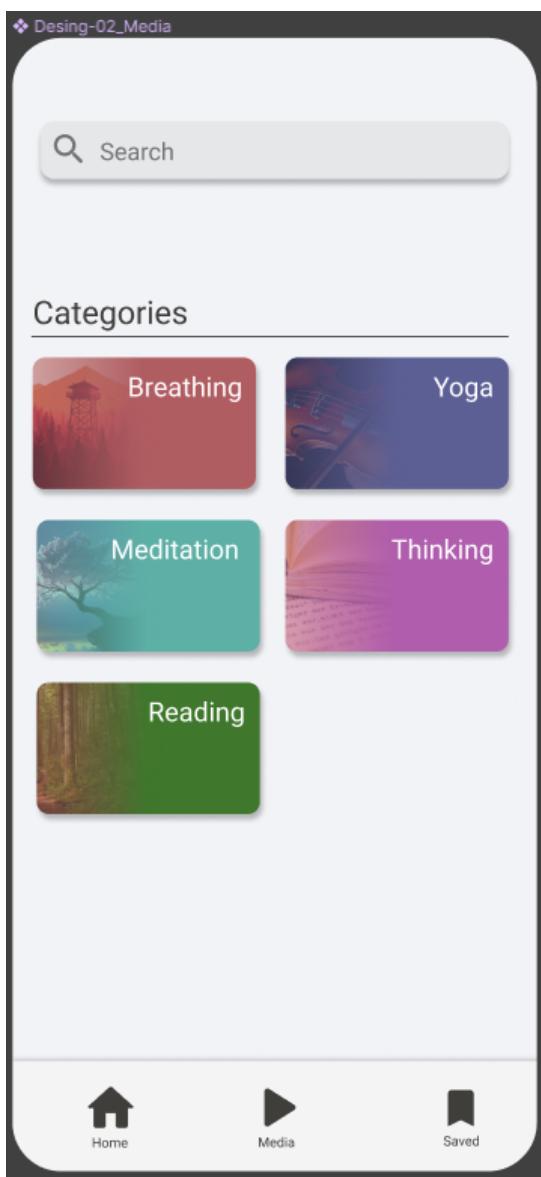


Abbildung 27: Kategorien UI-Prototyp

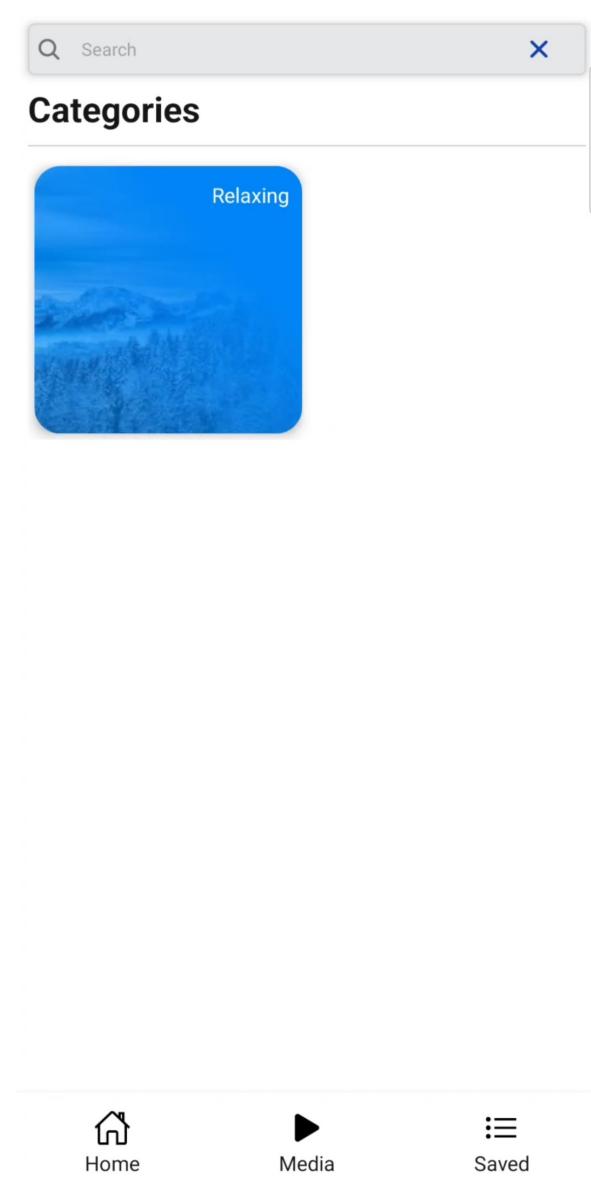


Abbildung 28: Kategorien in der App

Auf der Kategorie-Unterseite befindet sich links oben der "Zurück-Button". Dieser leitet den/die User:in wieder zurück auf den vorher dargestellten Screen.

Weiters ist der Titel der Kategorie und das Hintergrundbild zu sehen. Darunter werden, wie oben bereits erwähnt, die dazugehörigen Medien mit Thumbnail, Titel und einer gekürzten Beschreibung aufgelistet.

Rechts daneben ist noch ein Stern zu sehen, mit dem man ein Medium als Favorit markieren kann. (siehe Kapitel 10.8)

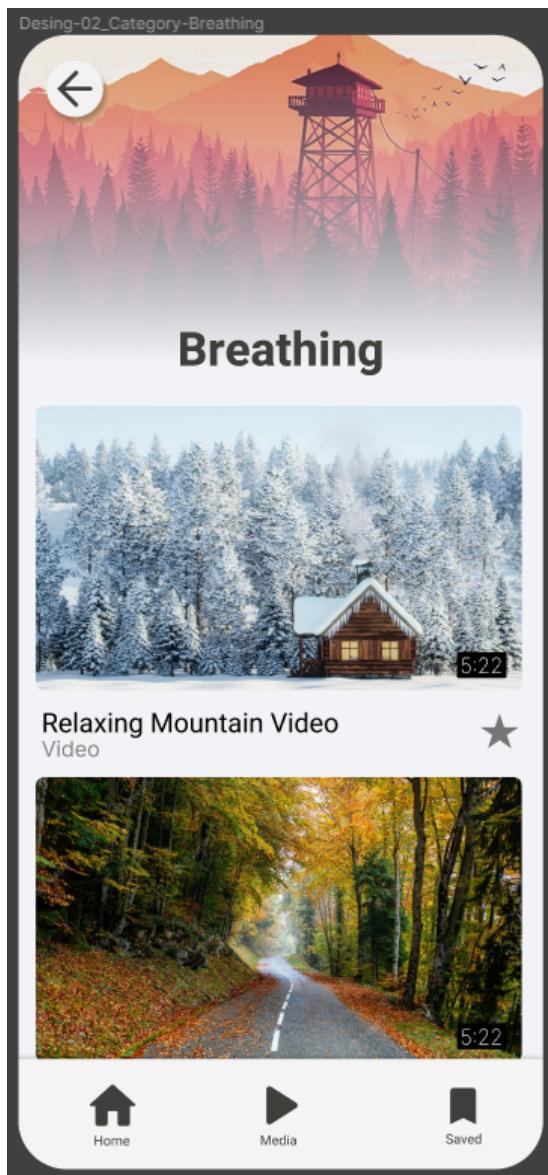


Abbildung 29: Unterseite UI-Prototyp



Abbildung 30: Unterseite in der App

10.6 Media Player

Wenn man ein Video auswählt, wird man zum Media Player weitergeleitet. Dieser bietet die Funktionen ein Video abzuspielen, zurückzuspulen und vorzuspulen. Man kann das Video auch vergrößern und sich auf dem Smartphone waagrecht anschauen.

Unter dem Media Player ist der Titel, der Stern zum Markieren eines Favoriten und die Beschreibung abgebildet. Wenn man ein Video als Favorit markiert, wird der Stern ausgefüllt.

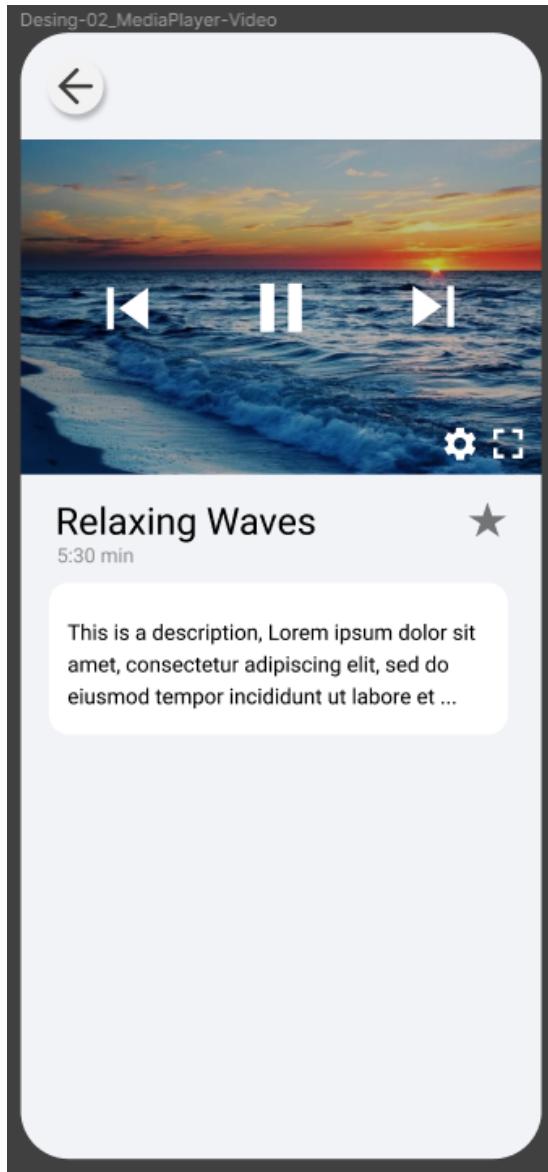


Abbildung 31: Media Player UI-Prototyp

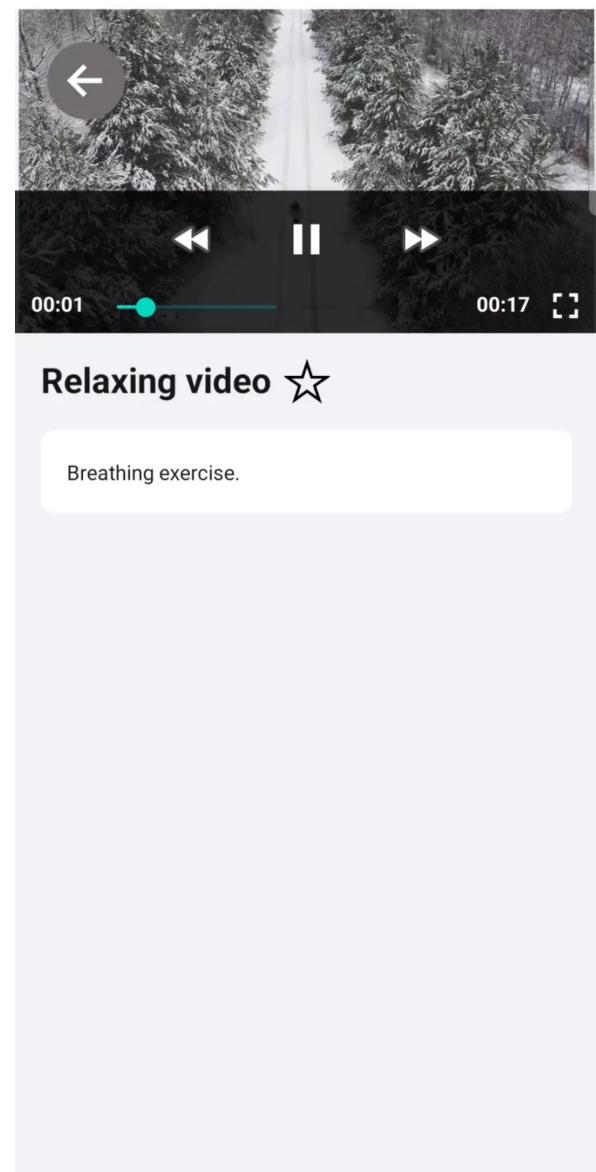


Abbildung 32: Media Player in der App

10.7 Suchleiste

Mit der Suchleiste kann man direkt nach einem Video suchen. Wenn man auf das vorgeschlagene Ergebnis tippt, wird man direkt zum Media Player weitergeleitet.

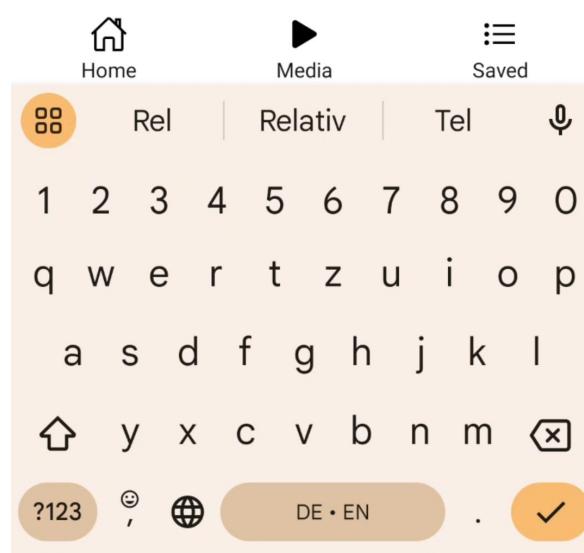
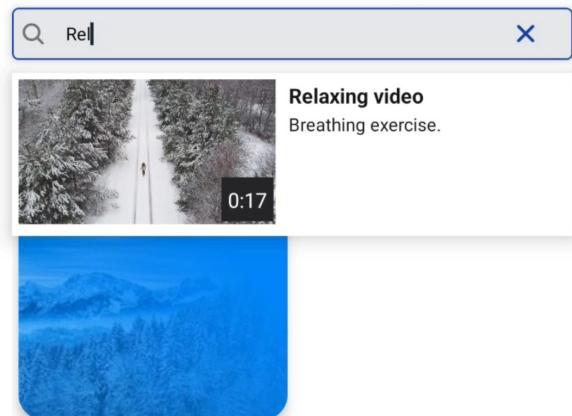


Abbildung 33: Suchleiste

10.8 Favoriten

Die favorisierten Medien werden auf der "Saved-Page" aufgelistet, die man in der Navigationsleiste findet.

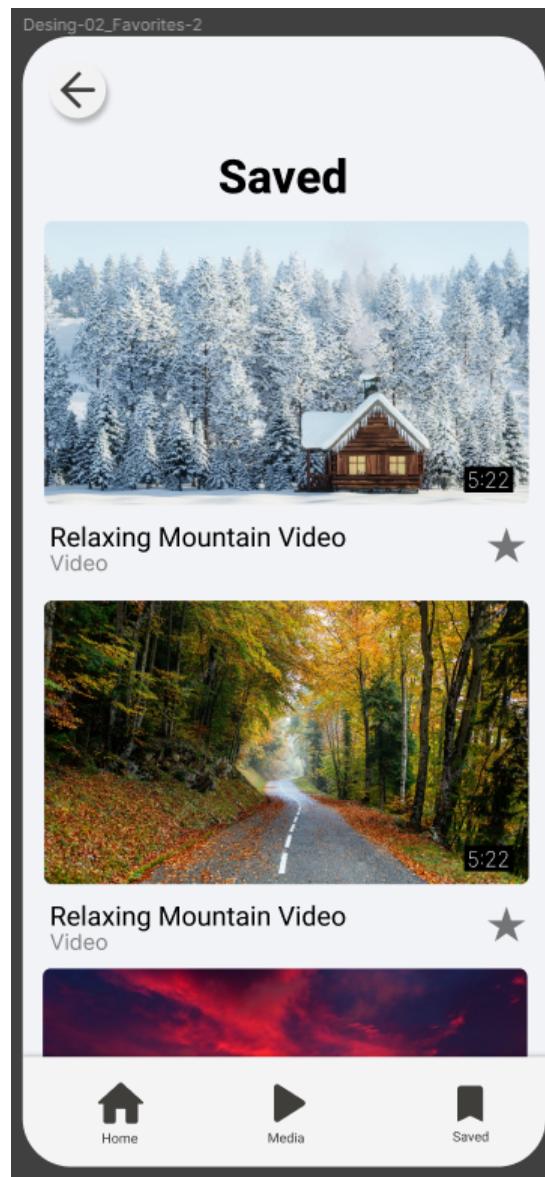
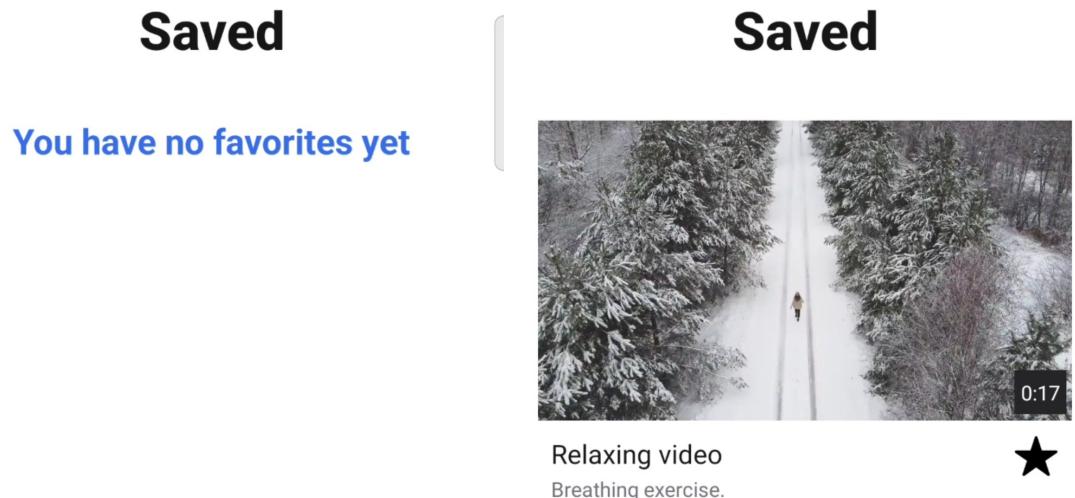


Abbildung 34: Favoriten UI-Prototyp

Wenn man als User:in noch keinen Favoriten gesetzt hat, erscheint die Nachricht: "You have no favorites yet".

Einen Favoriten kann man direkt auf der Favoritenseite durch das Antippen des Sterns wieder löschen, oder auch wenn man im Media Player auf den ausgefüllten Stern tippt.



Home



Media



Saved

Abbildung 35: Kein Favorit gesetzt



Home



Media



Saved

Abbildung 36: Ein Favorit gesetzt

10.9 Einstellungen

In den Einstellungen ist neben dem "Zurück-Button" auch ein Schalter zu finden, mit dem man das Aussehen von Relaxoon verändern kann. Es wird zwischen Light Mode und Dark Mode unterschieden. (siehe Kapitel 10.10)

Eine Info- und Helpfunktion sind auch verfügbar.

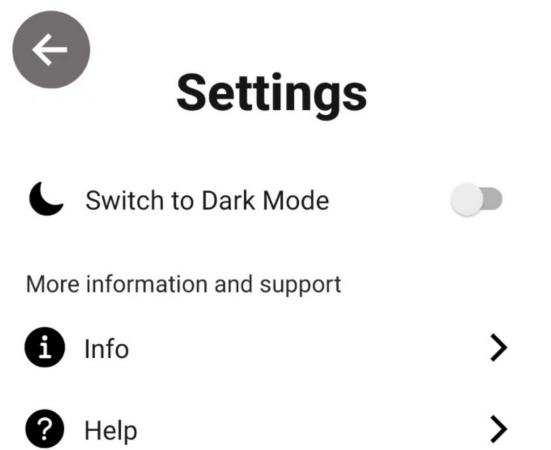


Abbildung 37: Einstellungen in der App

10.10 Light/Dark Mode

Der Light Mode und Dark Mode bieten dem Benutzer die Möglichkeit, das Erscheinungsbild der App entsprechend der eigenen Vorlieben oder den Umgebungsbedingungen anzupassen. Im Anschluss sind Screenshots von Relaxoon im Dark Mode abgebildet.

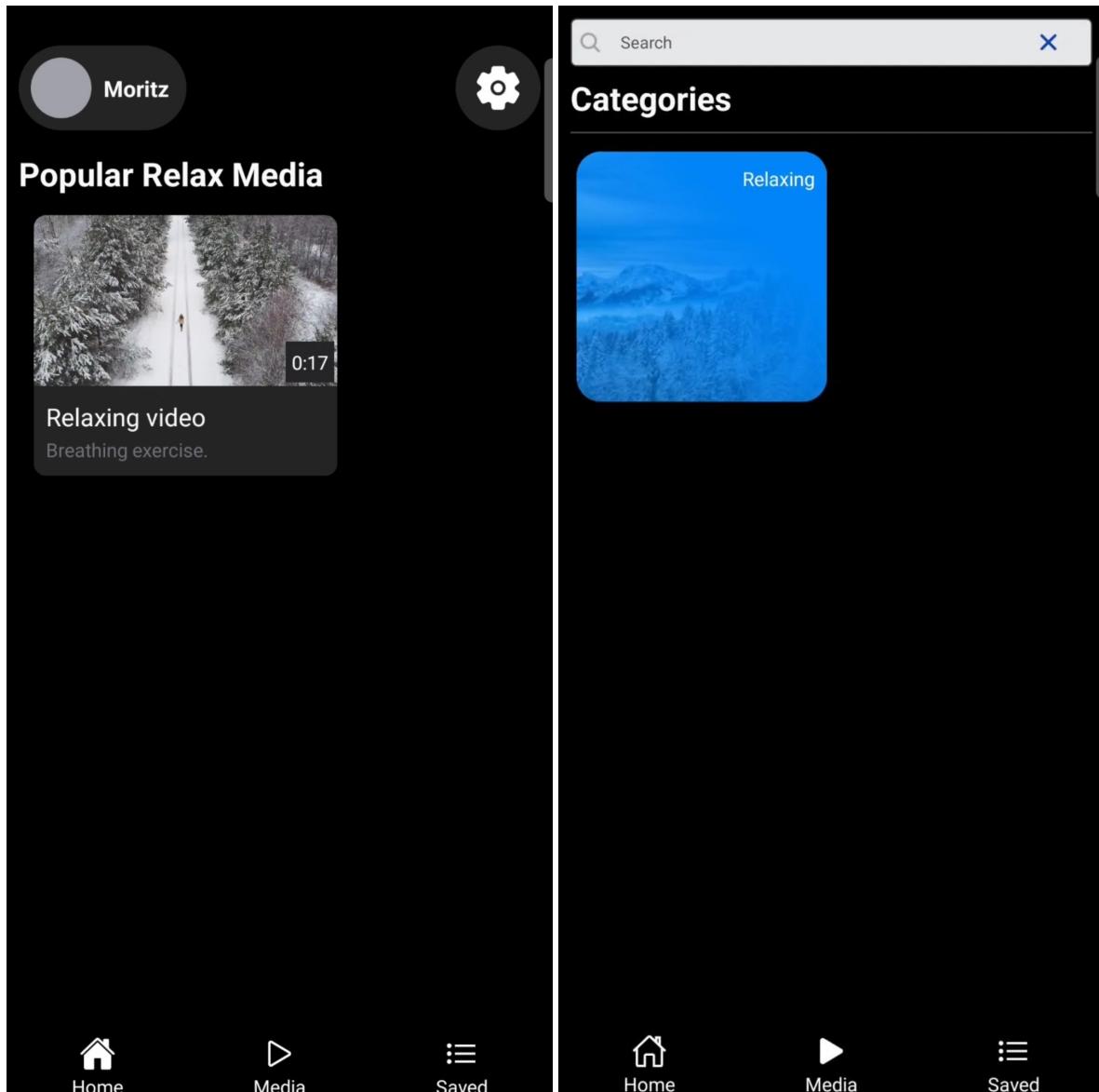


Abbildung 38: Dark Home-Screen

Abbildung 39: Dark Kategorie-Screen

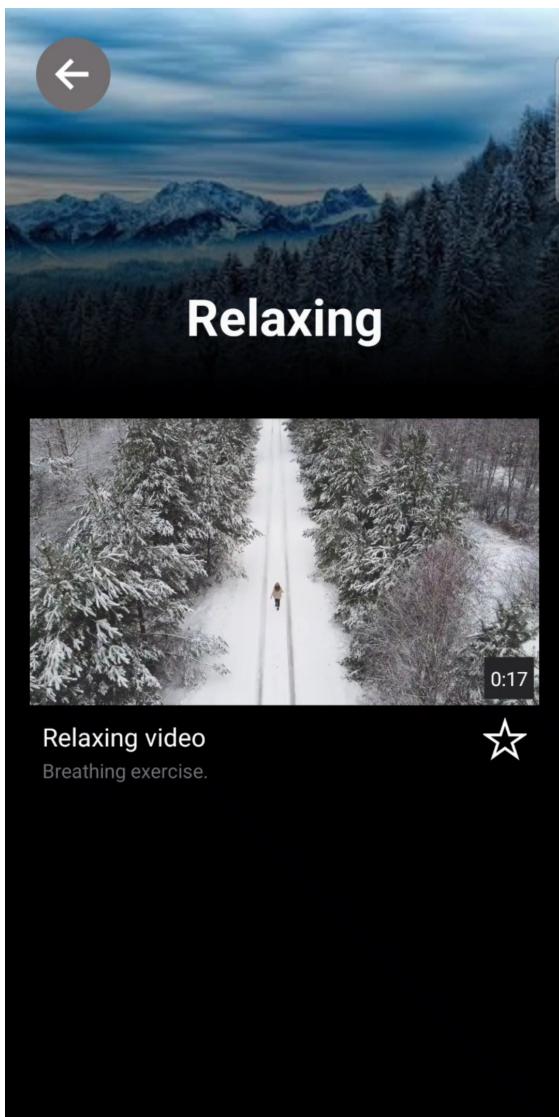


Abbildung 40: Dark Media-Screen

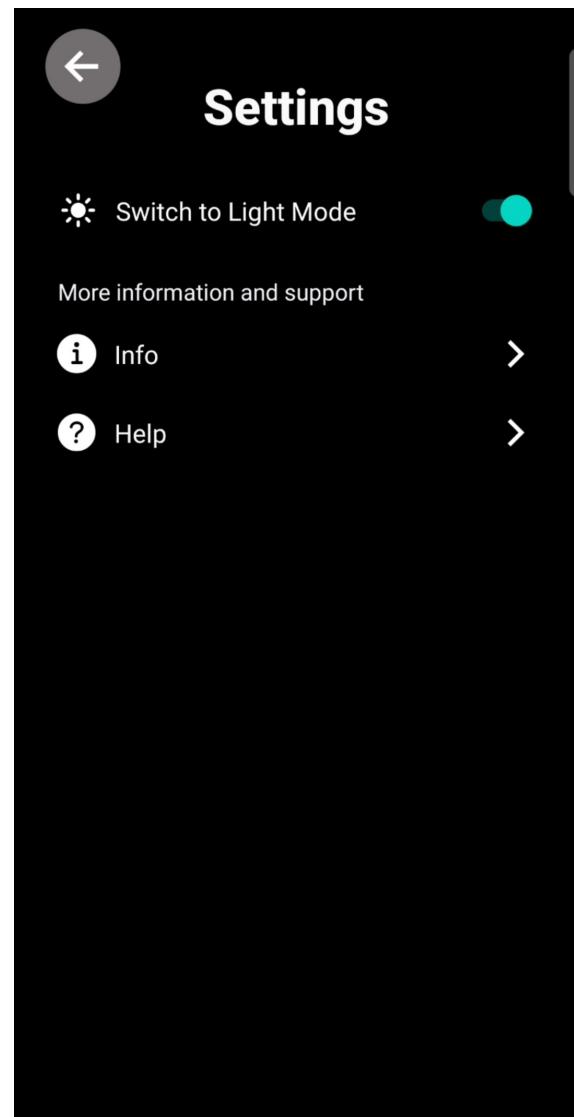


Abbildung 41: Dark Settings-Screen

10.11 Logo

Das Logo von Relaxoon besteht aus den beiden Buchstaben 'o', die stilisiert dargestellt sind. Es wird sowohl als App-Icon verwendet als auch im Google Play Store und im App Store angezeigt werden. Es gibt zwei Varianten des Logos: eine in Weiß und eine in Schwarz. Diese sind unten zu sehen.

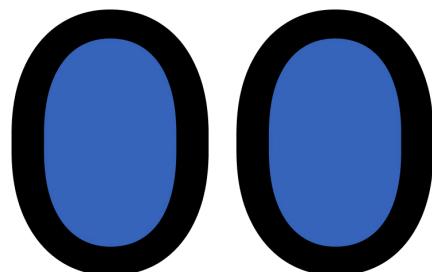


Abbildung 42: Logo in Weiß

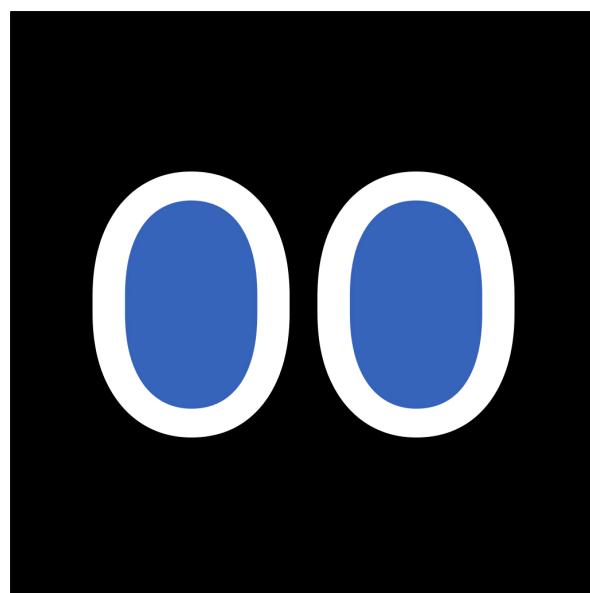


Abbildung 43: Logo in Schwarz

11 Implementierung

11.1 Entity Relationship Diagram (ERD)

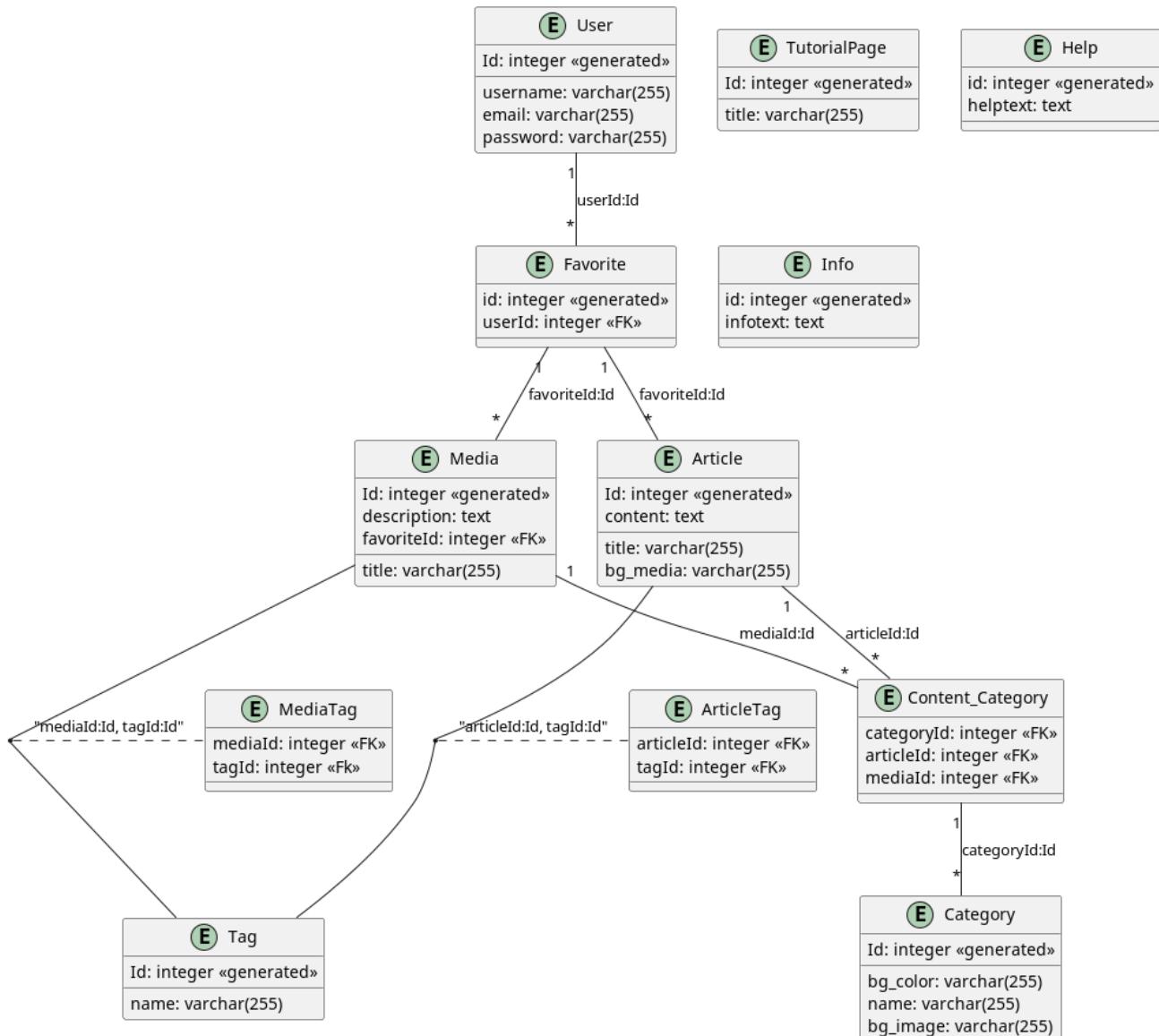


Abbildung 44: ERD

11.2 Wie werden Files gespeichert?

Im diesem Kapitel wird genau erklärt, wie Strapi mit der Verwaltung von Files umgeht. Es wurde keine Dokumentation für die folgenden Informationen gefunden. Daher

untersuchten die Entwickler das Datenmodell von Strapi. Man sieht oben in dem ERD, dass die Bilder nicht in einer Spalte gespeichert werden. Strapi speichert die Informationen der Files wie zum Beispiel Größe, Name und Auflösung in einer Tabelle namens "files". Diese ist auch mit einer anderen Tabelle namens "files_related_morphs" verbunden. In "files_related_morphs" wird der Name des Modells gespeichert, welcher von Strapi verwendet wird, um die Daten in der Datenbank zu manipulieren. Das Id des Element, welches einen Zugriff auf die Datei benötigt, wird auch in dieser Tabelle gespeichert. Der Client kann dann die Files erst erhalten, wenn er den Parameter "populate=file" bei dem REST-Request daran hängt. Strapi manipuliert dann die Ergebnisse, sodass die gefragten Daten, zum Beispiel Medias und die zugehörige Dateien eines Mediaelements im Responsebody, zurückgeschickt werden können.

11.3 Medias und Articles

Grundsätzlich wurde als Team entschieden, die Artikel und die Medien in zwei separaten Tabellen zu speichern, da Artikel mehrere Bilder, Videos und Audios enthalten. Diesen können dann mittels einem "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 Artikel implementiert wurde, da der Auftraggeber diese nicht in dem **Minimal Viable Product (MVP)** haben wollte. Für die Realisierung des kompletten Datenmodells war aber das Einfügen der Artikel erforderlich.

11.4 Tutorial-Page

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

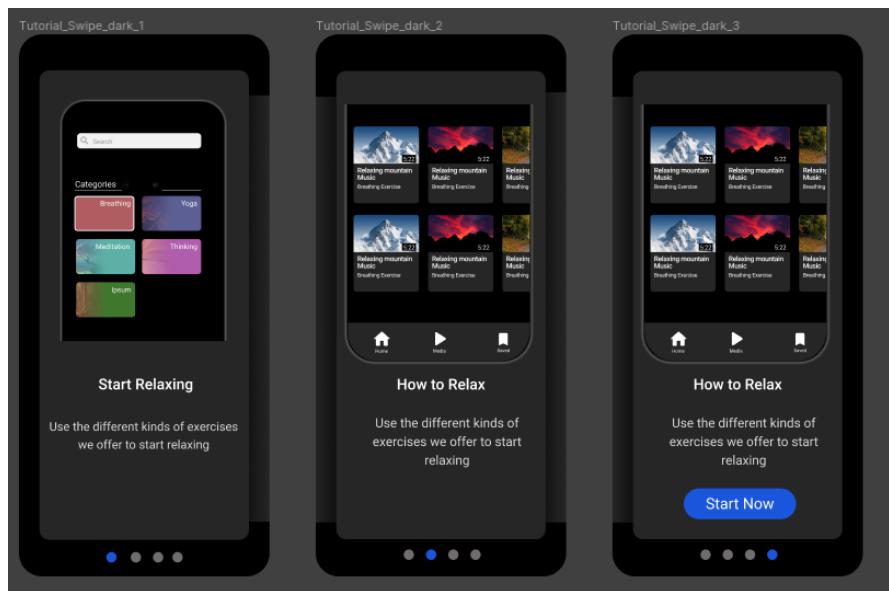


Abbildung 45: Screenshot aus dem UI Prototyp von Relaxoon

Das Team hat sich entschlossen, die Daten der Slideshow in einer Tabelle zu speichern. Eine alternative Lösungsmöglichkeit besteht darin, die Daten der Slideshow in der Applikation hart zu codieren. Das Team hat sich für die Persistierung der Daten entschieden, damit das Einpflegen der Inhalte nicht über die Applikation erfolgt. Wenn das Team sich für die alternative Lösungsvariante entschieden hätte, müsste bei jeder kleinen Änderung ein neuer Android- bzw. IOS-Build erstellt werden.

11.5 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 (Siehe 7.1.1 für mehrere Details). Es gibt auch die Möglichkeit, die REST-Ressource im Backend mittels der "Query Engine API" oder mit dem Einsatz der "Entity Service API" anzupassen. Allerdings wurden die Rückgabedaten des Endpoints gar nicht umgestellt, da die Dokumentation bzw. IntelliSense des "Query Engine API" bzw. "Entity Service API" sehr ungenau waren.

Listing 2: Code von Interactive Query Builder

```

1  const queryParamFilter = qs.stringify(
2    {
3      filters: {
4        $or: [
5          {
6            description: {
7              $containsi: text
8            }
9          },

```

```

10         {
11             title: {
12                 $containsi: text
13             }
14         },
15     {
16         categories: {
17             name: {
18                 $containsi: text
19             }
20         }
21     },
22     {
23         tags: {
24             name: {
25                 $containsi: text
26             }
27         }
28     }
29   ]
30 },
31 { encodeValuesOnly: true
32 );
33

```

In dem obigen Codesnippet befindet sich der Code für die Suchleiste. Es wird am Anfang entweder nach der Kategorie, nach dem Tag, nach der Beschreibung oder nach dem Titel eines Videos gefiltert. Wenn ein oder mehrere Treffer für diese Parameter gefunden werden, werden sie im Responsebody zurückgegeben. Die Groß- und Kleinschreibung wird dabei gar nicht beachtet.

API-Route mit den angehängten generierten Abfrageparametern:

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

Damit man sicherstellt, ob die generierten Parameter richtig sind, wurde folgende Ansicht von Postman verwendet:

The screenshot shows the Postman interface with the following details:

- Request URL:** https://relaxoon.solvistas.com/api/medias?filters[categories][id]=3&populate[tags]=true&populate[file]=true&populate[favorite][populate]=users_permissions_user
- Query Params:**
 - filters[categories][id]: 3
 - populate[tags]: true
 - populate[file]: true
 - populate[favorite][populate]: users_permissions_user
- Body:** The response body is shown in Pretty JSON format, containing:


```

1 {
2     "data": [],
3     "meta": {
4         "page": 1
5     }
6 }
```
- Status:** 200 OK, Time: 41 ms, Size: 958 B

Abbildung 46: Ansicht von Postman

11.6 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:innen 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.” [30]

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 nicht nötig, da die Anzahl der verwendeten UI-Komponenten nicht so groß war.

11.6.1 useState Hook

”useState is a React Hook that lets you add a state variable to your component.” [31]

11.7 Dark/Light Mode

Da das Team sich für die Verwendung von der UI-Library ”Native Base” entschieden hat, war die Realisierung dieses Features sehr einfach. Die Komponenten von Native Base haben die Properties `_light` und `_dark`, sodass man die Farbe einer Komponente je nach Modus definieren kann. Native Base verwendet im Hintergrund einen `useContext` Hook, in dem die Modi der Applikation gespeichert werden.

11.7.1 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.” [32]

11.8 Help und Info Screen

Für die gleichen Gründe, welche im Unterkapitel 11.4 genannt wurden, wurden die Daten dieser zwei Ansichten in einem Single Type (Siehe 7.1.2) gespeichert.

11.9 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/die User:in die App erst verwenden kann, wenn er bereits authentifiziert ist. Die Ansicht fürs Login soll aber nicht angezeigt werden, wenn der/-die User:in bereits authentifiziert ist. Um zu überprüfen, ob der/die Nutzer:in noch eingeloggt ist oder nicht, wurde der Token im AsyncStorage gespeichert. Dieser ist ähnlich zu dem localStorage in der Webentwicklung. Somit wurde bei der Öffnung der App geprüft, ob der Token valid und nicht abgelaufen ist. Damit der/die User:in auf die anderen Screens nicht zugreifen kann, wenn er nicht eingeloggt ist, wurde eine boolsche Zustandsvariable verwendet, welche das LoginScreen als die zweite Ebene unserer Stack-Navigation rendert, wenn der 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 3: protected screens

```

1   // Screen names
2   type PageSettings = Record<string, { iconName: string; screen:
3     React.ComponentType<any> }>;
4   const settingsForScreenName: PageSettings = {
5     Home: { iconName: 'home', screen: HomeScreen },
6     Media: { iconName: 'play', screen: CategoryScreen },
7     Saved: { iconName: 'list', screen: SavedScreen }
8   };
9
10  const Tab = createBottomTabNavigator();
11  const Stack = createStackNavigator();
12
13  function MainContainer(): JSX.Element {
14    const { colorMode } = useColorMode();
15    const [tokenExists, setTokenExists] = useState<boolean>(false);
16    useEffect(() => {
17      getToken()
18        .then((data) => {
19          console.info(typeof data);
20          if (!data) throw new Error('no token');
21          const decodedToken: { iat: number; exp: number; id: number } =
22            jwtDecode(data);
23          const eighthoursAfterNow = new Date(new Date().getHours() +
24            8).getMilliseconds();
25          if (decodedToken.exp < eighthoursAfterNow) {
26            setTokenExists(false);
27            AsyncStorage.clear();
28            return;
29          }
29        })
29      setTokenExists(true);
30    })
31  }

```

```

30         .catch((e) => {
31             console.info('no token', e);
32             setTokenExists(false);
33         });
34     console.info(tokenExists);
35 }, []);
36
37     return (
38         <Stack.Navigator
39             screenOptions={{{
40                 headerShown: false
41             }}}
42         >
43             /**
44             <Stack.Screen name={"WaitScreen"} component={WaitScreen}/>
45         */
46             /**
47             {!tokenExists && <Stack.Screen name="Tutorial"
48                 component={TutorialScreen}/>}
49         */
50             {!tokenExists && <Stack.Screen name="Login" component={LoginScreen} />}
51             {!tokenExists && <Stack.Screen name="Registration"
52                 component={RegistrationForm} />}
53             <Stack.Screen
54                 name="Main"
55                 options={{{
56                     headerShown: false
57                 }}}
58             >
59                 {() => (
60                     <Tab.Navigator
61                         screenOptions={{{
62                             headerShown: false,
63                             headerTitleStyle: {
64                                 fontSize: 16,
65                                 color: colorMode === 'dark' ? colors.fontDark : colors.fontLight
66                             }
67                         }}}
68                         tabBar={(props) => <TabBar color={colorMode} {...props} />}
69                     >
70                         {Object.entries(settingsForScreenName).map(([pageName, settings]) =>
71                             (
72                                 <Tab.Screen key={pageName} name={pageName}
73                                     component={settings.screen} />
74                             )));
75                     </Tab.Navigator>
76                 )}
77             </Stack.Screen>
78             <Stack.Screen name="Content" component={MediaScreen} />
79             <Stack.Screen component={VideoScreen} name="Video" />
80             <Stack.Screen component={SettingsScreen} name="Settings" />
81             <Stack.Screen name="Info" component={InfoScreen} />
82             <Stack.Screen name="Help" component={HelpScreen} />
83         </Stack.Navigator>
84     );
85 }
86
87 export default MainContainer;

```

11.10 Validierung

Strapi 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 4: 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);

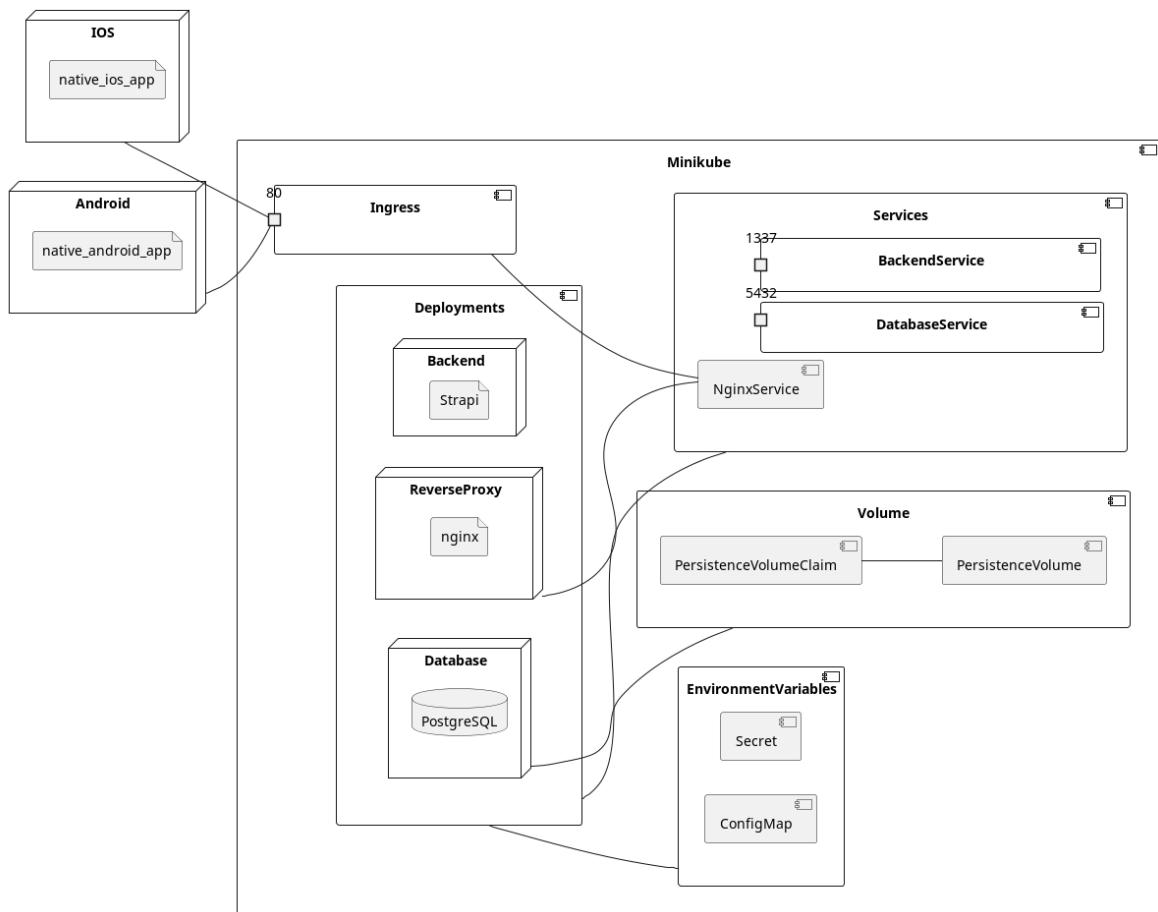
```

11.11 Deployment

11.11.1 Allgemeines

Die Veröffentlichung der Applikation im Play Store und im App Store wurde vom Team nicht verlangt. Das Hochladen der Android-Applikation auf Firebase App Distribution war jedoch eine essentielle Aufgabe. Das Backend wurde auf den Servern der Firmen solvistas und MACOLUTION deployt. Für Demonstrationszwecke wurde ein lokales Deployment auf Minikube erstellt.

11.11.2 Deployment Diagram



11.11.3 Deployment von dem Backend auf Minikube

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

- Strapi
- Datenbank

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

11.11.4 K8S-Konfigurationen für die Datenbank

Für das Verständnis dieses Kapitels ist es wichtig zu wissen, was ein Pod ist:

”Ein Pod (übersetzt Gruppe/Schote, wie z. B. eine Gruppe von Walen 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.“ [33]

Zusätzlich zu einer Deployment- und Service-Komponente ist für die Sicherung der Credentials der Datenbank eine sogenannte Secret-Komponente nötig. Für die Vermeidung von Datenverlusten in Fällen wie Neustart eines Pods oder für die Aktualisierung der Konfiguration des Pods ist das Anlegen einer PersistenceVolume-Komponente sehr notwendig. 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 und für die PersistenceVolume-Komponente bzw. die PersistenceVolumeClaim-Komponente wurden fertige Konfigurationen genommen.

Listing 5: 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 6: 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

```

11.11.5 K8S-Konfigurationen für Strapi

Für Strapi wird auch eine Secret-Komponente benötigt, da der Server geheime Umgebungsvariablen benötigt.

Damit das Deployment von Strapi erstellt werden kann, muss die Anwendung zuerst in ein Docker Image umgewandelt werden. Dies kann mit der Verwendung des folgenden Dockerfiles erfolgen.

Listing 7: 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 zlib-dev
4    libpng-dev vips-dev git > /dev/null 2>&1
5  ARG NODE_ENV=production
6  ENV NODE_ENV=${NODE_ENV}
7
8  WORKDIR /opt/
9  COPY package.json package-lock.json ./
10  RUN npm install -g node-gyp
11  RUN npm config set fetch-retry-maxtimeout 600000 -g && npm install
12    --only=production
13  ENV PATH /opt/node_modules/.bin:$PATH
14  WORKDIR /opt/app
15  COPY .
16  RUN npm run build
17  # Creating final production image
18  FROM node:18-alpine
19  RUN apk add --no-cache vips-dev
20  ARG NODE_ENV=production
21  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 WORKDIR /opt/
41 COPY --from=build /opt/node_modules ./node_modules
42 WORKDIR /opt/app
43 COPY --from=build /opt/app ./
44 ENV PATH /opt/node_modules/.bin:$PATH
45
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 in Einsatz nimmt.

11.11.6 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 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 8: 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 9: 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

```

11.11.7 Was ist eine Deployment-Komponente

Eine Deployment-Komponente dient dazu, dass ein Pod erstellt wird und die benötigten Ressourcen bzw. Volumes und Zugriffsrechte dafür definiert werden.[34]

Für das Anlegen einer Deployment-Komponente muss man zuerst die Anwendung containerisieren. Für weit verbreitete Software, wie zum Beispiel 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 10: 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

```

11.11.8 Erstellung einer Service-Komponente

Mit dem Einsatz eines Services in Kubernetes können Pods, die sich im gleichen Cluster befinden, miteinander kommunizieren. [35]

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

Listing 11: create a service component

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

11.11.9 Erstellung einer Ingress-Komponente

Was ist eine Ingress-Komponente

“An ingress controller acts as a reverse proxy and load balancer. It implements a Kubernetes Ingress. The ingress controller adds a layer of abstraction to traffic routing, accepting traffic from outside the Kubernetes platform and load balancing it to Pods running inside the platform.” [36]

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

Ein Ingress Controller ist mehr oder weniger mit einem Reverse Proxy zu vergleichen. Die Konfigurationen für den Ingress-Controller sind in dem folgenden Code-Snippet zu finden:

Listing 12: ingress-controller

```

1   # see
2       https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/
3   apiVersion: networking.k8s.io/v1
4   kind: Ingress
5   metadata:
6       name: relaxoon-ingress
7
8   spec:
9       rules:
10      - host: hello-world.info
11          http:
12              paths:
13                  - path: /relaxoon
14                      pathType: Prefix
15                      backend:
16                          service:
17                              name: relaxoon-strapi
18                              port:
19                                  number: 80

```

11.11.10 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 13: generate android and IOS

```
1 npx expo prebuild
```

11.11.11 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 folgt aus:

Listing 14: generate apk

```
1 ./gradlew assembleRelease
```

11.12 Hochladen einer Mobileapp auf Firebase App Distribution

Nach dem Anlegen eines Accounts bei Firebase sind folgende Schritte durchzuführen:

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

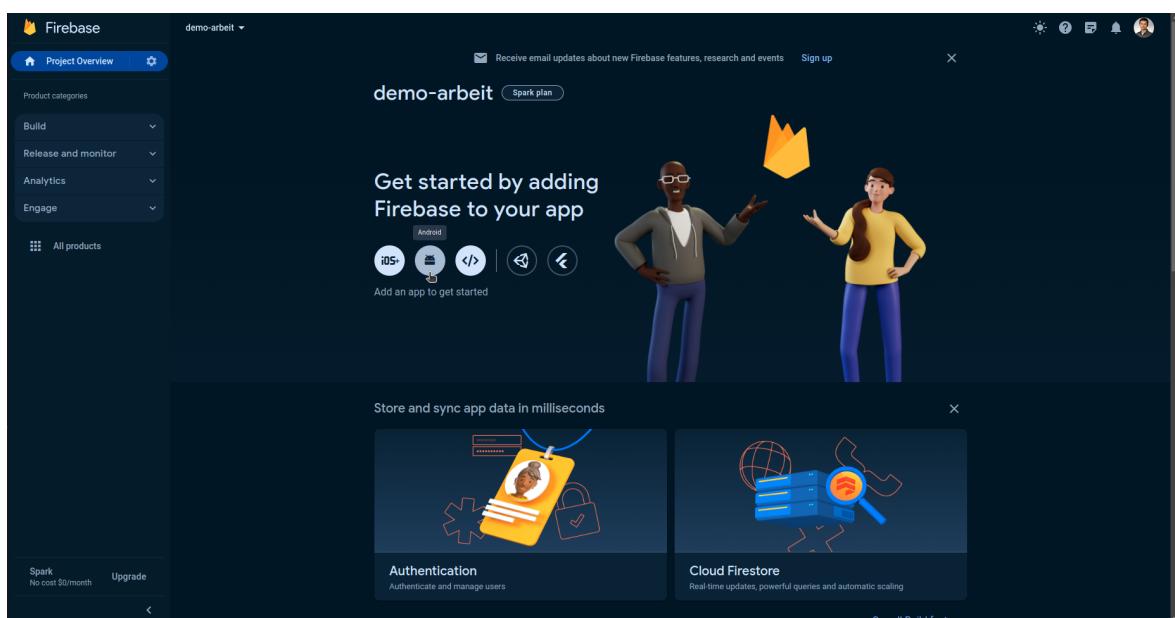


Abbildung 47: create android config

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

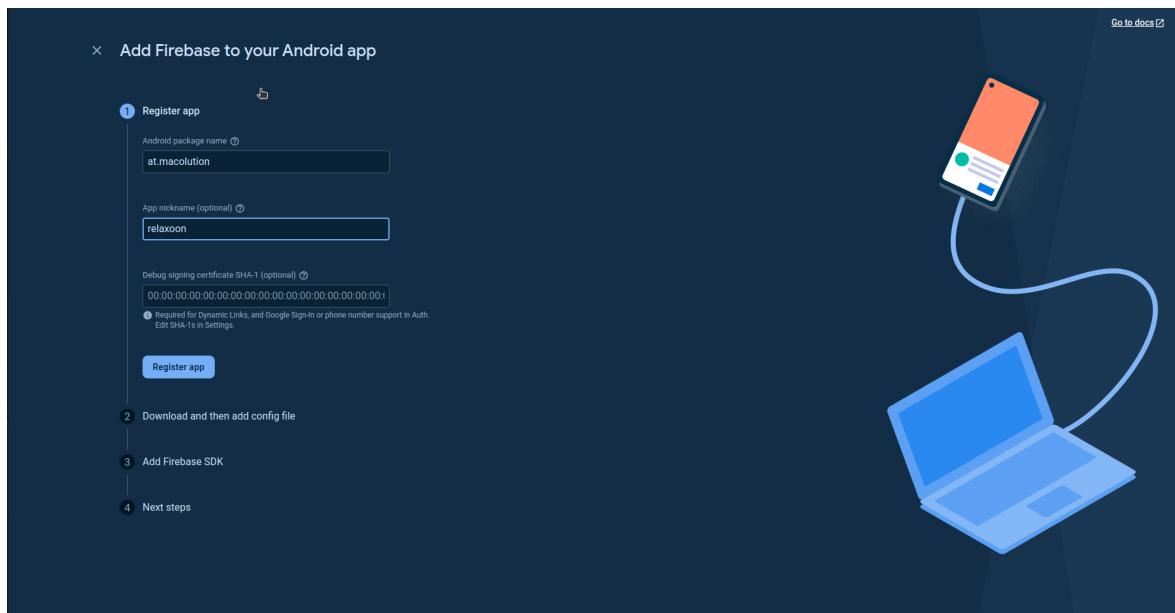


Abbildung 48: Android Config

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

Danach geht man auf die App Distribution und klickt auf "Get Started".

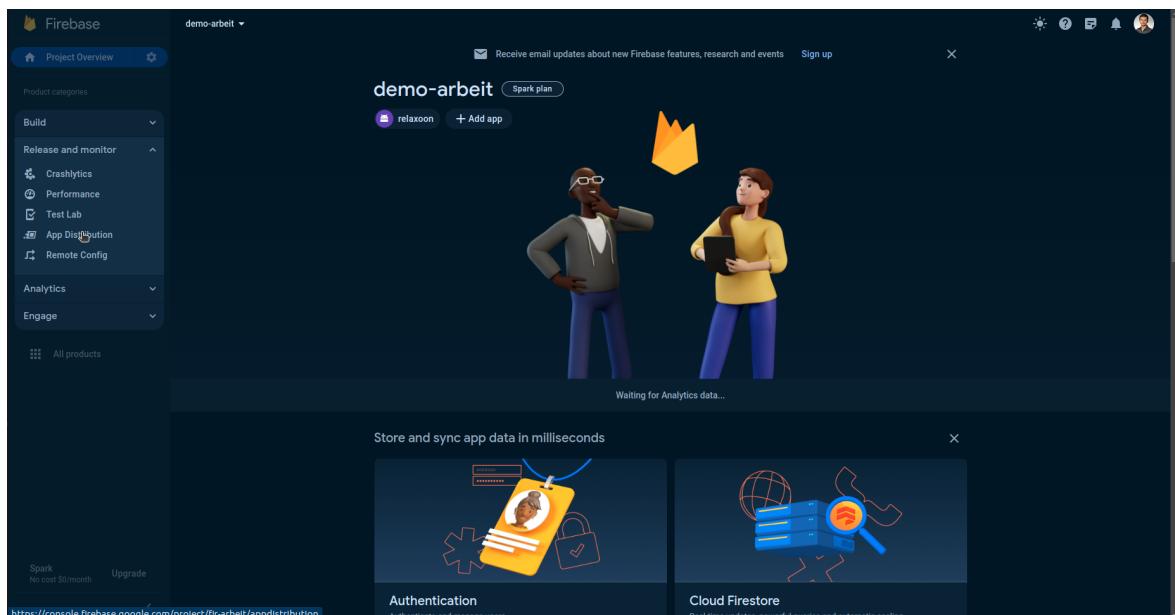


Abbildung 49: App Distribution

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

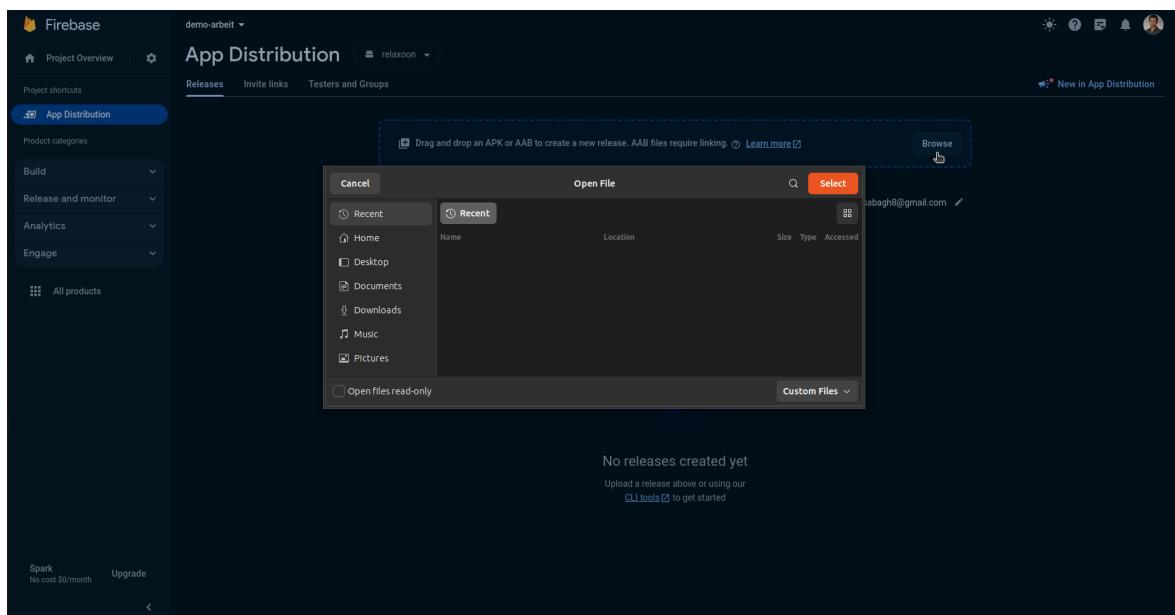


Abbildung 50: App Upload

Das Produkt kann danach an andere Testuser:innen verteilt werden, indem man die Email-Adressen dieser Tester:innen beim Release eingibt.

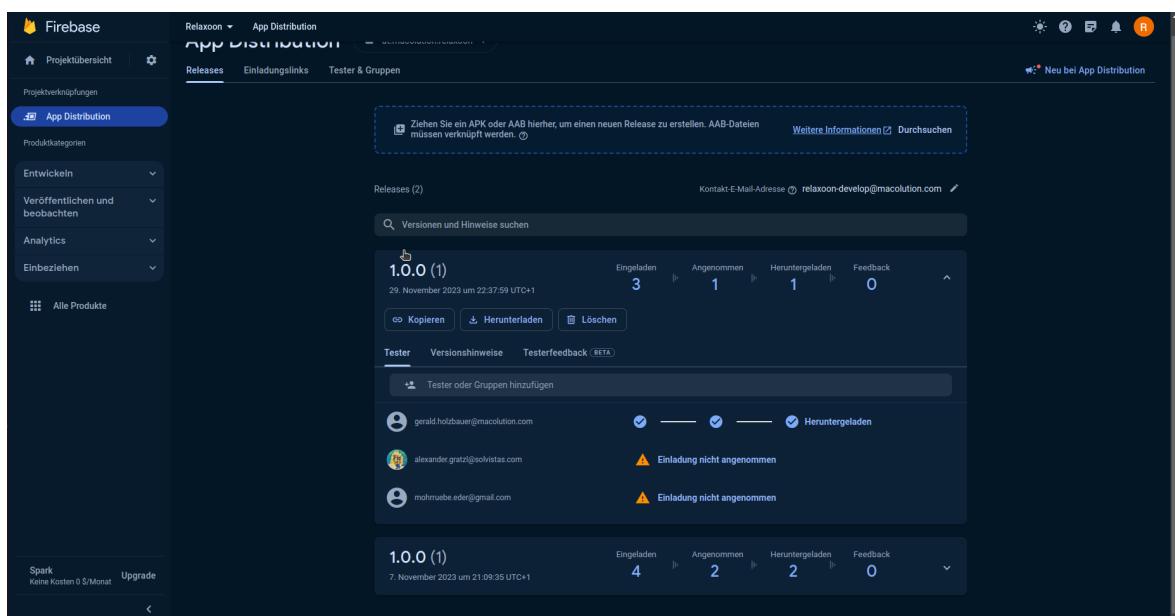


Abbildung 51: Releases

12 Ausgewählte Aspekte und Probleme

12.1 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 verwendet. Bei IOS und Android ist das Holen der Daten von nicht sicheren Quellen beziehungsweise nur von HTTP gar nicht möglich. Auf Android gibt es aber einige Ausnahmen für das Holen der Daten von einem lokalen Host. Bei Android kann man bei den generierten Build und Project Files einige Konfigurationen ändern. Das ist jedoch keine gute Lösung, da diese Files bei jedem Build geändert werden können. Außerdem können diese generierten Files nicht in das **Version Control System** (VCS) eingecheckt werden. Bei Android muss man die Adresse 10.0.2.2 statt localhost verwenden.[37] 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. Dieses Tool kann dann den lokalen Port des Servers in das Internet mittels eines Tunnels weiterleiten und eine HTTPS-Adresse für den weitergeleiteten Server generieren. Somit schaut die Verbindung zwischen Client und Server wie folgt aus:

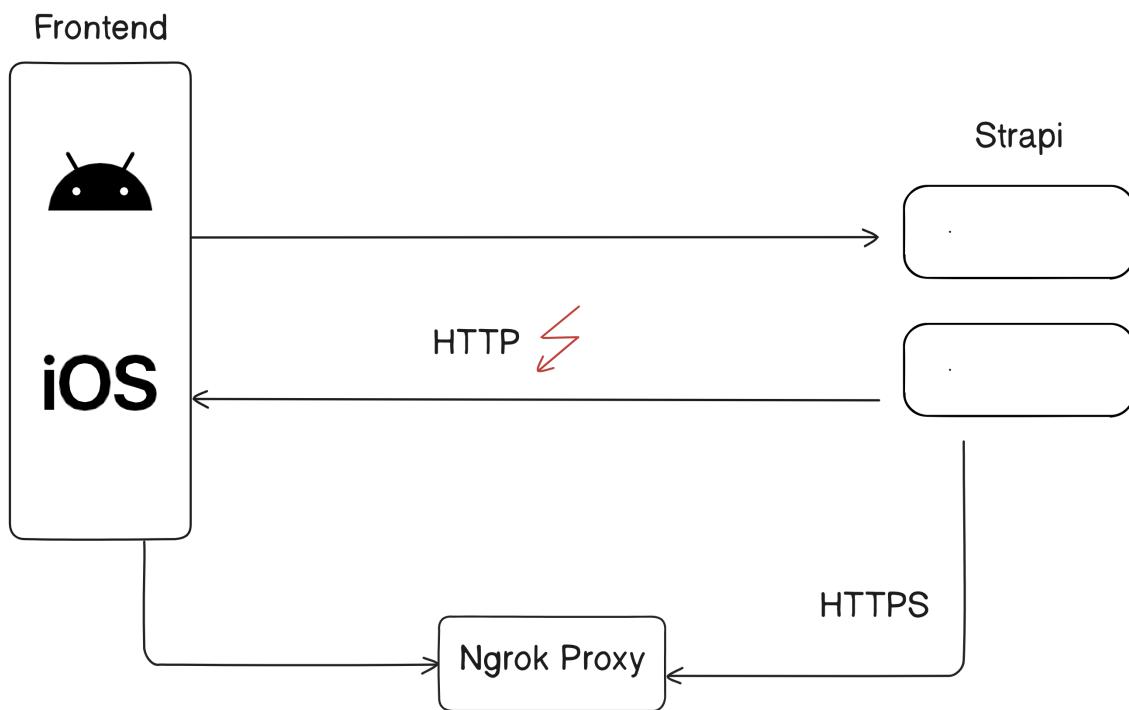


Abbildung 52: ngrok

Das Problem tauchte wiederholt in der Testphase auf. Als Lösung hat sich das Team ein Zertifikat besorgt. Nach dem Einbauen dieses Zertifikates hat sich die Verbindung zwischen Client und Server um einiges verändert. Diese schaut in dem produktiven System so aus:

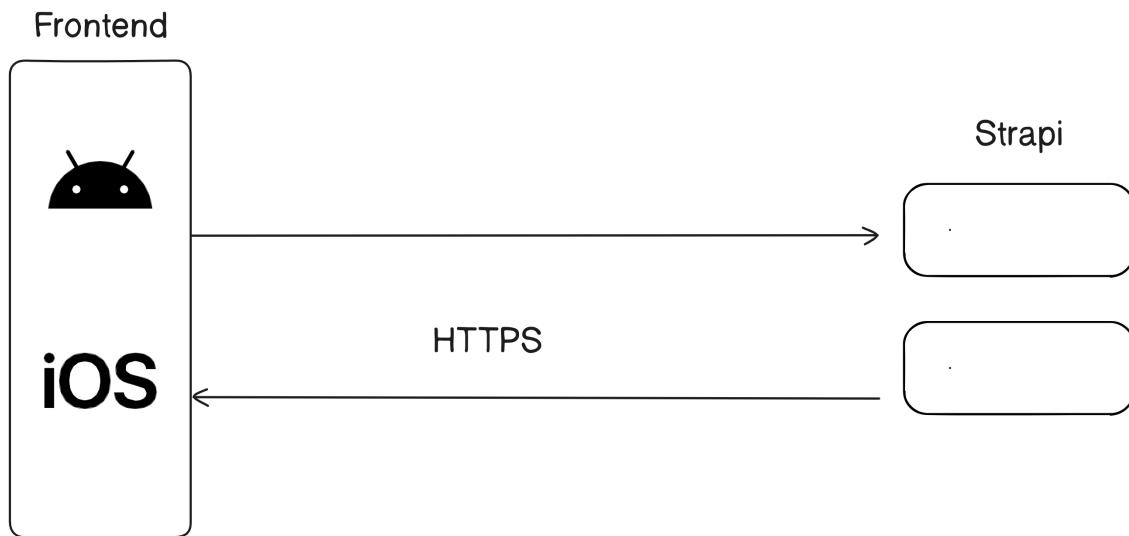


Abbildung 53: HTTPS in prod

Dieses Problem war relativ schwierig zum Debuggen, als man die Applikation mit dem Staging Server verbunden hat. Es gab keine eindeutige Fehlermeldung, in der erklärt wird, dass die Zertifikate nicht mehr gültig sind oder gar nicht existieren. Die

Applikation meldete, dass keine Verbindung zum Server aufgebaut werden kann, obwohl der Server schon lauffähig war.

12.2 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 war, 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 [38] werden unterschiedliche Gründe und mehrere Lösungsmöglichkeiten für das 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 besitzt, die nicht mehr gebraucht bzw. nicht mehr verwendet werden. Beim Löschen dieser Libraries und der Wiederinstallation von den Modulen mittels ”npm install” tauchte die ”legacy-peer-deps” Meldung wieder auf. Zur Problemlösung wurde ein Tool namens ”depcheck” installiert, welches den Codebase scannt und dem Entwickler bekannt gibt, ob eine Library im Codebase verwendet wird oder nicht. [39] Nach dem Löschen aller nicht benötigten Libraries und der Generierung der Apk bzw. AaB hat die Applikation problemlos funktioniert.

12.2.1 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.” [40]

12.2.2 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.” [40]

12.3 Probleme mit Thumbnails

Relaxoon zeigt mehrere Videos für Antistress-Meditationen an. Damit diese schön darstellbar sind, muss jedes Video unbedingt ein Thumbnail haben. Der Content-Manager könnte aber beim Erstellen des Videos vergessen, ein Thumbnail für das Video zu erstellen. Auf der Adminoberfläche können keine Thumbnails zum Video hinzugefügt werden, da diese Aktion sehr schlechte User Experience verursacht. Man kann zwar eine Funktion implementieren, die nach dem Hochladen eines Videos ein Thumbnail mittels **Fast Forward Moving Picture Experts Group** (ffmpeg) aus dem ersten Bild des Videos erstellt. Diese Möglichkeit ist 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. Es wurde eine Expo-Library gefunden, die im Frontend Thumbnails für die Videos erstellt. Die Verwendung dieser war aber nicht sehr vorteilhaft, da der Generierungsprozess sehr langsam war. Als Lösung wurde im Code ein nicht abspielbares Videoelement deklariert, dass für den Zuschauer als Thumbnail dient. Das Video ist erst abspielbar, wenn man darauf klickt. Diese Lösung wurde mit den Konzepten ”Lazy Loading” und ”Suspenses” kombiniert, damit Loading-Spinners statt leere Flächen für den User angezeigt werden, wenn das Laden des Videos etwas länger dauert.

12.3.1 Suspense

”<Suspense> lets you display a fallback until its children have finished loading.”[41]

12.3.2 Lazy Loading

”lazy lets you defer loading component’s code until it is rendered for the first time.”[42]

12.4 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 obiges Problem). Für die Berechnung der Dauer wurde im Frontend das ”onLoad” Event, welches in den Properties (props) des Videoelements ist, verwendet, um die Dauer zu lesen und diese in das Format ”mm:ss” umwandeln zu können.

12.5 Probleme mit useEffect und React-navigation

Library

Für das Holen der Daten aus dem Server wurde 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. Später wurde herausgefunden, dass die Library ”React-navigation” den ”useEffect Hook” im Hintergrund deaktiviert. Für das Holen der Daten ist die Übergabe eines sogenannten ”useCallback Hooks” als Parameter in einem custom hook namens ”useFocusEffect” von React-navigation notwendig.[43]

12.5.1 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.” [44]

12.5.2 useEffect

”useEffect is a React Hook that lets you synchronize a component with an external system.”[45]

”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. ” [46]

12.5.3 useCallback

”useCallback is a React Hook that lets you cache a function definition between re-renders.” [47]

12.5.4 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.” [48]

12.6 Probleme mit Dependency Updates

Im Laufe der Implementierungsphase wurden einige Dependencies wie React Native oder Strapi ein oder zweimal upgedatet. Bei einigen Libraries haben die Updates problemlos funktioniert. Bei anderen wie zum Beispiel ”Expo Cli” war es so, dass das Programm

nach dem Update nicht mehr lauffähig war. Daher hat man sehr viele Einstellungen in dem Programm ändern müssen, um die Kompilierfehler zu beseitigen. Bei der Library "JWT-Decod", welche für die Dekodierung von dem JWT eingesetzt wurde, hat der Code am Beginn problemlos funktioniert. Nach einem Update kam es dazu, dass die Library den Token nicht decodieren konnte. Nach einer langen Recherche kam man darauf, dass man ab der vierten Version eine Library namens "base-64" installieren muss. Danach kann diese importiert werden und als ein Singleton in dem Projekt definiert werden.

Listing 15: base-64 als Singleton

```
1 import { decode } from 'base-64';
2 global.atob = decode;
```

[49]

Ein anderes Beispiel, bei dem das gleiche Problem wieder auftauchte, war bei der Aktualisierung von der Library "Native base". Es stellte sich heraus, dass die vor dem Update verwendete Version von Native Base die letzte ist, die vom Hersteller unterstützt wird.

12.7 Probleme mit dem Datenmodell

Nach der Realisierung stellte sich heraus, dass das ERD um einiges verbessert werden kann. In dem folgenden Bild ist das neue Design des verbesserten ERDs zu finden.

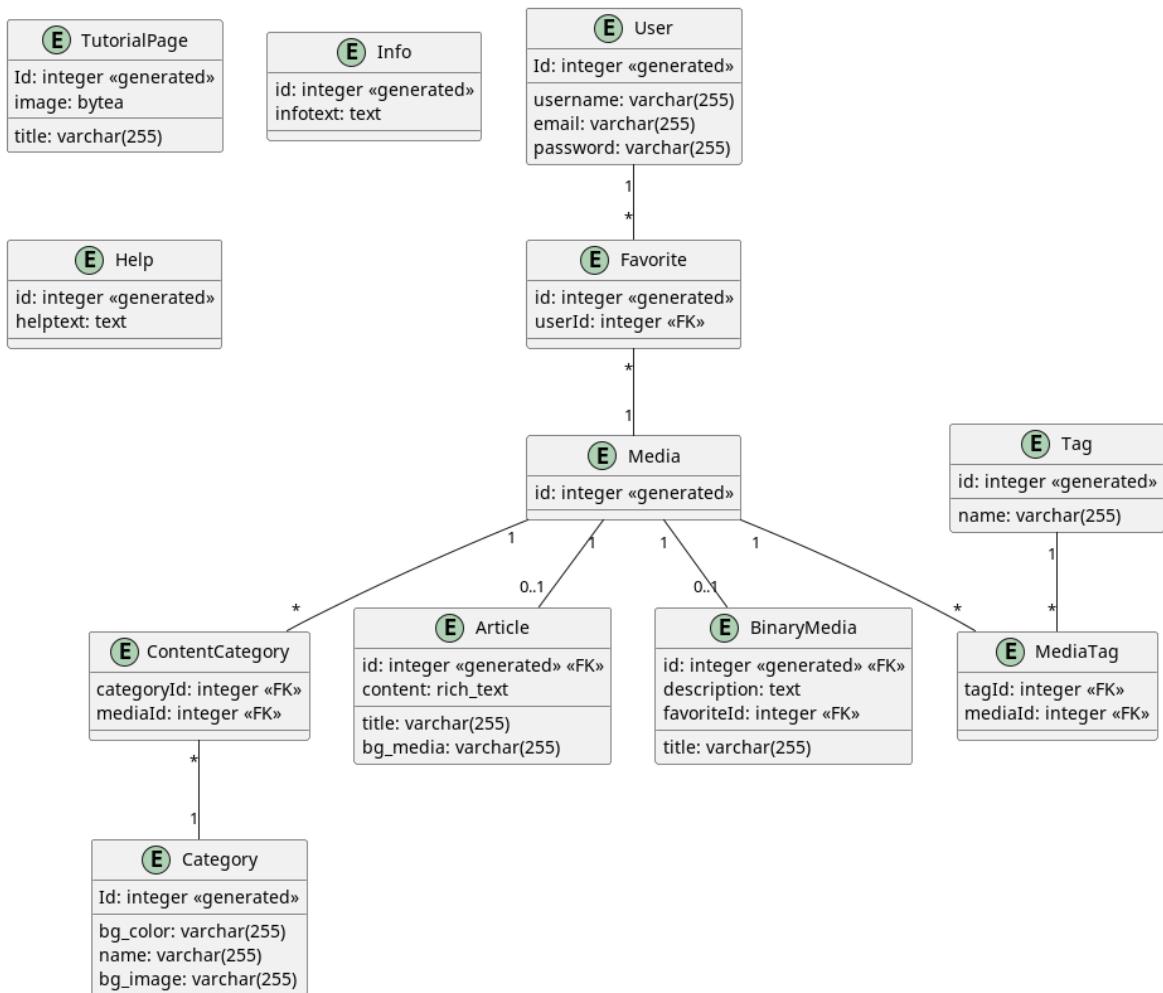


Abbildung 54: Neues ERD

Die Vorteile dieser Verbesserungen liegen darin, dass man alle Arten von Medien nur mit einer einzigen SQL-Query abfragen kann. Außerdem ist die Komplexität um einiges reduziert, da nicht jede Medienart eine Beziehung zu den Tags und Kategorien hat. Vorher waren zwei Queries nötig, um alle Arten der Medien abzufragen.

12.8 Relaxoon Plakat

Im Zuge des Projektes wurde mithilfe von Adobe Illustrator ein Plakat zur Repräsentation der mobilen App entwickelt.

12.8.1 Umsetzungsschritte

1. Ein Farbverlauf wurde bestimmt, der sich von der hellblauen Farbe `#64C5DF` (Hexadezimalschreibweise) bis zu einem dunkleren Blau `#3A62AC` streckt. Der Farbverlauf dient als Grundlage des Plakates.

2. Abgerundete Dreiecke sowie die abstrakte Form, welche am Rand des Plakats entlang geht, wurden in Kombination mit dem Farbverlauf erzeugt und platziert.
3. Die Logos von Relaxoon, der HTL Leonding, den Firmen solvistas und MACOLUTION, dem CMS Strapi und der Programmiersprache React Native wurden hinzugefügt.
4. Der Diplomarbeitsbetreuer und die Projektpartner wurden namentlich erwähnt.
5. Das Grundlayout eines Smartphones wurde erstellt und mit einem Entwurf des Tutorials der App versehen.
6. Fotos und Namen der App-Entwickler wurden eingefügt.
7. Mit einem kurzen Beschreibungstext wurde das Plakat fertiggestellt.



Abbildung 55: Plakat

13 Resümee

Nach einer intensiven und bereichernden Erfahrung im Rahmen unserer Diplomarbeit zur Entwicklung einer Entspannungs-App möchten wir einige wichtige Erkenntnisse zusammenfassen. Zentral für den Erfolg unseres Projekts war die konsequente Einhaltung des langfristig geplanten und sorgfältig koordinierten Projektablaufs. Durch die fristgerechte Umsetzung der einzelnen Arbeitsschritte und die korrekte Ausführung der zugewiesenen Aufgaben konnte ein reibungsloser Ablauf gewährleistet werden.

Ein entscheidender Faktor für unseren Erfolg war die effektive Zusammenarbeit im Team. Durch offene Kommunikation, gegenseitige Unterstützung und ein starkes Zusammengehörigkeitsgefühl konnten wir die gesteckten Ziele erreichen. Die regelmäßigen Treffen und die Reflexion der Meilensteine ermöglichten es uns, aufkommende Herausforderungen rechtzeitig zu erkennen und gemeinsam Lösungsstrategien zu entwickeln.

Besonders wertvoll war auch das Praktikum bei unserem Auftraggeber solvistas. Diese praktische Erfahrung ermöglichte es uns, unser theoretisches Wissen in der realen Welt anzuwenden und wertvolle Einblicke in die Arbeitsweise eines Unternehmens zu gewinnen. Die enge Zusammenarbeit mit den Experten von solvistas hat nicht nur unsere fachlichen Fähigkeiten erweitert, sondern auch zu einer vertieften Verständigung über die Anforderungen und Bedürfnisse der Zielgruppe geführt.

Insgesamt können wir auf eine äußerst positive und erfolgreiche Zusammenarbeit zurückblicken. Diese Erfahrung hat uns nicht nur fachlich, sondern auch persönlich weitergebracht und wird uns in unserer zukünftigen beruflichen Laufbahn von großem Nutzen sein.

14 Anhang

14.1 Prompt-Verlauf

Eingabe:

Erkläre mir den Unterschied zwischen funktionalen und nicht funktionalen Anforderungen.

KI-Modell:

Funktionale und nicht-funktionale Anforderungen sind zwei Hauptkategorien von Anforderungen in der Softwareentwicklung und anderen Bereichen des Projektmanagements. Hier sind die grundlegenden Unterschiede zwischen ihnen:

Funktionale Anforderungen: Beschreiben, welche Funktionen das System bereitstellen soll und welche Aufgaben es ausführen muss. Definieren die spezifischen Funktionen, Dienste oder Aufgaben, die das System erfüllen muss, um die Benutzeranforderungen zu erfüllen. Konzentrieren sich auf die "WasAspekte des Systems und beschreiben, welche Ergebnisse erwartet werden. Beispiel: "Das System muss dem Benutzer ermöglichen, sich mit einem Benutzernamen und Passwort anzumelden."

Nicht-funktionale Anforderungen: Beschreiben Eigenschaften des Systems, die nicht unbedingt Funktionen sind, sondern Qualitätsmerkmale, die das System erfüllen muss. Betreffen oft die "WieAspekte des Systems und definieren Qualitätsmerkmale wie Leistung, Sicherheit, Benutzerfreundlichkeit und Zuverlässigkeit. Berücksichtigen Aspekte wie Skalierbarkeit, Wartbarkeit, Zuverlässigkeit, Leistung und Benutzerfreundlichkeit. Beispiel: "Das System muss eine Antwortzeit von weniger als 2 Sekunden für Benutzeranfragen sicherstellen."

In einem Softwareprojekt sind funktionale Anforderungen eher auf die eigentlichen Aufgaben und Funktionen der Software ausgerichtet, während nicht-funktionale Anforderungen die Qualität und Leistung des Systems definieren. Beide Arten von Anforderungen sind wichtig, um ein vollständiges und erfolgreiches Produkt zu entwickeln, da sie gemeinsam die grundlegenden Spezifikationen und Erwartungen für das System festlegen.

14.2 Schriftliche Arbeitsaufteilung

1. Inhaltsverzeichniss
2. Ausgangssituation [ME]
3. Problemstellung [ME]
4. Ziele [ME]
5. Aufgabenstellung [ME]
6. Marktanalyse [ME]
7. Projektanforderungen [ME]
8. Technologien [AA]
9. Systemarchitektur [AA]
10. Entwurfsentscheidungen [AA]
11. Design [ME]
12. Implementierung [AA]
13. Ausgewählte Aspekte und Probleme [AA]
 - a) Relaxoon Plakat [ME]
14. Resümee [ME]

Literaturverzeichnis

- [1] Unbekannt, „VOLKSLEIDEN BURNOUT,” 2021, letzter Zugriff am 24.3.2024. Online verfügbar: https://www.promentesteiermark.at/blog/volksleiden-burnout-das-ausbrennen-wird-noch-immer-unterschaetzt/#:~:text=19%20Prozent%20der%20Proband*innen,Syndrom%20sollten%20keinesfalls%20ignoriert%20werden.
- [2] T. Aichinger, „MACOLUTION,” 2023. Online verfügbar: <https://www.solvistas.com/de/news/solvistas-group-waechst-weiter>
- [3] Unbekannt, „Stress.” Online verfügbar: [https://www.internisten-im-netz.de/fachgebiete/psyche-koerper/stress.html#:~:text=Zeichen%20von%20Nervosit%C3%A4t%20\(Z%C3%A4hneknirschen%20in,Stoffwechselst%C3%B6rungen%20Allergien%20und%20Entz%C3%BCndungskrankheiten%20f%C3%BCr%C3%BChren.\)](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%20Allergien%20und%20Entz%C3%BCndungskrankheiten%20f%C3%BCr%C3%BChren.))
- [4] ——, „Ziele setzen,” 2023, letzter Zugriff am 20.12.2023. Online verfügbar: <https://karrierebibel.de/ziele-setzen/>
- [5] P. Steubel, „Was ist ein MVP?” 2023, letzter Zugriff am 24.3.2024. Online verfügbar: <https://asana.com/de/resources/minimum-viable-product>
- [6] www.creditreform.de, „SO FUNKTIONIERT DIE MARKTANALYSE:METHODEN UND BEISPIELE,” 2024, letzter Zugriff am 24.3.2024. Online verfügbar: <https://www.creditreform.de/aktuelles-wissen/praxisratgeber/marktanalyse>
- [7] B. Penin, „App Marketing,” 2023, letzter Zugriff am 24.3.2024. Online verfügbar: <https://www.itportal24.de/ratgeber/app-marketing-wie-vermarkte-ich-meine-app#App-Marketing-Kanaele>
- [8] absatzwirtschaft.de, „Apps deinstallieren,” 2018, letzter Zugriff am 24.3.2024. Online verfügbar: <https://www.absatzwirtschaft.de/top-studie-top-oder-flop-app-nutzer-entscheiden-innerhalb-von-sechs-tagen-ob-sie-eine-app-deinstallieren-221132/>
- [9] OpenAI. (2024) GPT-3.5-based Chatbot. Answer provided by the GPT-3.5-based language model. Online verfügbar: <https://www.openai.com/gpt-3>
- [10] C. Wannakhaos, „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>
- [11] starpi.io, „REST API parameters,” 2024, letzter Zugriff am 14.3.2024. Online verfügbar: <https://docs.strapi.io/dev-docs/api/rest/parameters>
- [12] strapi.io, „Introduction to the Content-type Builder,” 2024, letzter Zugriff am 14.3.2024. Online verfügbar: <https://docs.strapi.io/user-docs/content-type-builder#:~:text=Collection%20types%20are%20content%2Dtypes,collection%20types%20and%20single%20types>.

- [13] ——, „Query Engine API,” 2024, letzter Zugriff am 14.3.2024. Online verfügbar: <https://docs.strapi.io/dev-docs/api/query-engine>
- [14] ——, „Entity Service API,” 2024, letzter Zugriff am 14.3.2024. Online verfügbar: <https://docs.strapi.io/dev-docs/api/entity-service>
- [15] ——, „Upload plugin,” 2024, letzter Zugriff am 14.3.2024. Online verfügbar: <https://docs.strapi.io/dev-docs/plugins/upload>
- [16] ——, „Introduction to the Media Library,” 2024, letzter Zugriff am 14.3.2024. Online verfügbar: <https://docs.strapi.io/user-docs/media-library>
- [17] S. Srivastava, „Getting “Network Error” while uploading an image to strapi from a url,” 2022, letzter Zugriff am 20.3.2024. Online verfügbar: <https://forum.strapi.io/t/getting-network-error-while-uploading-an-image-to-strapi-from-a-url/20269/5>
- [18] firebase.google.com, „Firebase-App-Verteilung,” 2023, letzter Zugriff am 20.12.2023. Online verfügbar: <https://firebase.google.com/docs/app-distribution?hl=de>
- [19] L. Hahn, „Cross-Plattform App – Plattformübergreifende Entwicklung mit Flutter, React Native & Co.” 2023. Online verfügbar: <https://www.itportal24.de/ratgeber/Cross-Plattform-app>
- [20] 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/>
- [21] N. Mansour, „React Native vs Kotlin Multiplatform: The 2023 Guide,” 2023. Online verfügbar: <https://www.instabug.com/blog/react-native-vs-kotlin-mutliplatform-guide>
- [22] L. Hahn, „Flutter vs. React Native,” 2023. Online verfügbar: <https://www.itportal24.de/ratgeber/flutter-vs-react-native#React-vs-Flutter>
- [23] L. von Arcitech, „Xamarin vs. React Native,” 2023. Online verfügbar: <https://www.linkedin.com/pulse/xamarin-vs-react-native-comprehensive-comparison-Cross-Plattform/>
- [24] F. Churchville, „Content-Management-System (CMS),” 2021. Online verfügbar: <https://www.computerweekly.com/de/definition/Content-Management-System-CMS>
- [25] ionos, „Headless CMS,” 2022. Online verfügbar: <https://www.ionos.at/digitalguide/hosting/cms/headless-cms-was-sind-die-vorteile/>
- [26] strapi.io, „Strapi vs Storyblok,” 2024, letzter Zugriff am 15.3.2024. Online verfügbar: <https://strapi.io/headless-cms-comparison/strapi-vs-storyblok>
- [27] nueva.ch, „Was ist Storyblok?” letzter Zugriff am 15.3.2024. Online verfügbar: <https://nueva.ch/news/was-ist-storyblok>
- [28] reactnative.dev, „How TypeScript and React Native works,” 2023, letzter Zugriff am 20.3.2024. Online verfügbar: <https://reactnative.dev/docs/typescript#:~:text=We%20recommend%20that%20you%20use,using%20Babel%20instead%20of%20TypeScript.>

- [29] S. S. P. Limited, „Expo vs React Native CLI,” 2023, letzter Zugriff am 20.3.2024. Online verfügbar: <https://medium.com/@softworthsolutionspvtltd/expo-vs-react-native-cli-7e47c7630039>
- [30] S. Springer, „Zentrales State Management in React ,” 2022. Online verfügbar: <https://entwickler.de/react/zentrales-state-management-react>
- [31] react.dev, „useState,” 2024, letzter Zugriff am 23.2.2024. Online verfügbar: <https://react.dev/reference/react/useState>
- [32] M. Gold, „Using useContext in React: a comprehensive guide,” 2023, letzter Zugriff am 24.3.2024. 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>.
- [33] kubernetes.io, „Pods,” 2023, letzter Zugriff am 19.12.2023. Online verfügbar: <https://kubernetes.io/de/docs/concepts/workloads/pods/>
- [34] ——, „Deployments,” 2023, letzter Zugriff am 19.12.2023. Online verfügbar: <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
- [35] ——, „Service,” 2023, letzter Zugriff am 19.12.2023. Online verfügbar: <https://kubernetes.io/docs/concepts/services-networking/service/>
- [36] traefik.io/, „What is a Kubernetes ingress controller?” letzter Zugriff am 14.3.2024. Online verfügbar: <https://traefik.io/glossary/kubernetes-ingress-and-ingress-controller-101/#:~:text=An%20ingress%20controller%20acts%20as,Pods%20running%20inside%20the%20platform>.
- [37] 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>
- [38] 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>
- [39] npmjs.com depcheck, „Wie funktioniert depcheck,” 2023. Online verfügbar: <https://www.npmjs.com/package/depcheck>
- [40] 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/>
- [41] react.dev, „Suspenses,” 2023. Online verfügbar: <https://react.dev/reference/react/Suspense>
- [42] ——, „Lazy Loading,” 2023. Online verfügbar: <https://react.dev/reference/react/lazy>
- [43] 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>
- [44] react.dev, „Hooks,” 2023. Online verfügbar: <https://react.dev/reference/react/hooks>

- [45] ——, „useEffect hook,” 2023. Online verfügbar: <https://react.dev/reference/react/useEffect>
- [46] ——, „Synchronizing with Effects,” 2023. Online verfügbar: <https://react.dev/learn/synchronizing-with-effects>
- [47] ——, „useCallback,” 2023. Online verfügbar: <https://react.dev/reference/react/useCallback>
- [48] reactnavigation.org, „useFocusEffect,” 2023. Online verfügbar: <https://reactnavigation.org/docs/use-focus-effect/>
- [49] ShepSims, „v4.0.0 React Native support - property atob doesn't exist,” 2024, letzter Zugriff am 12.3.2024. Online verfügbar: <https://github.com/auth0/jwt-decode/issues/241>

Abbildungsverzeichnis

1	Logo MACOLUTION	1
2	Logo solvistas	1
3	Musik dient zur Entspannung	2
4	3
5	Use-Case-Diagram 1	7
6	Use-Case-Diagram 2	8
7	Ablauf	11
8	12
9	App Statistik [8]	15
10	Datentypen	20
11	Relationen	21
12	validation	21
13	media-library	24
14	Systemarchitektur	26
15	Mockups	36
16	Tutorial im UI-Prototypen	37
17	Erste Tutorial-Page	38
18	Zweite Tutorial-Page	38
19	Dritte Tutorial-Page	39
20	Registrierung UI-Prototyp	40
21	Registrierung in der App	40
22	Login UI-Prototyp	41
23	Login in der App	41
24	Bestätigungs Nachricht	41
25	Home-Screen UI-Prototyp	42
26	Home-Screen in der App	42
27	Kategorien UI-Prototyp	43
28	Kategorien in der App	43
29	Unterseite UI-Prototyp	44
30	Unterseite in der App	44
31	Media Player UI-Prototyp	45
32	Media Player in der App	45
33	Suchleiste	46
34	Favoriten UI-Prototyp	47
35	Kein Favorit gesetzt	48
36	Ein Favorit gesetzt	48
37	Einstellungen in der App	49
38	Dark Home-Screen	50
39	Dark Kategorie-Screen	50
40	Dark Media-Screen	51
41	Dark Settings-Screen	51
42	Logo in Weiß	52

43	Logo in Schwarz	52
44	ERD	53
45	Screenshot aus dem UI Prototyp von Relaxoon	55
46	Ansicht von Postman	56
47	create android config	66
48	Android Config	67
49	App Distribution	67
50	App Upload	68
51	Releases	68
52	ngrok	70
53	HTTPS in prod	70
54	Neues ERD	76
55	Plakat	78

Tabellenverzeichnis

Quellcodeverzeichnis

1	file upload config in strapi	22
2	Code von Interactive Query Builder	55
3	protected screens	58
4	Validierungsschemen	59
5	K8S PVC	61
6	K8S PV	62
7	Strapi Dockerfile	62
8	Secrets für Strapi	63
9	Secrets für die Datenbank	63
10	create k8s deployments	64
11	create a service component	64
12	ingress-controller	65
13	generate android and IOS	65
14	generate apk	66
15	base-64 als Singleton	75