Dokumentation

Vorwort  
- Einleitung  
- Ziel

Konzeption  
- Thematik

* Funktionalitäten
* Frameworks
* Packages/Plugins

Appentwicklung  
Vorgehensweise  
Planung / welche Architektur / Meilensteine und Features festlegen   
- Setup – expo, emulator, react native

- Navigation  
- Eintragssystem  
- To Do Listen  
-Modals / Forms  
- Standortabfrage – google places autocomplete, maps  
- Validation  
- Bilder Zugriff auf die Galerie / Validierung der Inputs  
- Speichersystem  
- Settings – Profilbild   
-DatePicker DateTimePicker Kalender

- Bewertungssystem  
- Statistik  
-Introduction Modal Walkthrough  
- Grafikimplementation – global Stylesheet, Fonts, Accordion Collapse

Hintergründe  
- Build APK -dependencies depcheck – lokal oder mit eas  
- User Testing /Funktionales Testing/Performance Testing/UX Testing   
- Fazit  
Abbildungsverzeichnis

Design  
Seiten  
Funktionalitäten

Vorwort  
Mein Bereich umfasst das Entwerfen, Entwickeln und Implementieren der mobilen App-Anwendung. Da ich mich das erste Mal in Folge dieser Arbeit mit mobiler App Entwicklung auseinandergesetzt habe, wurde ich mit vielem Neuen konfrontiert und habe stehts neues dazu gelernt, was meine Arbeit sehr interessant für mich gemacht hat. Durch Änderungen des UX- oder UI-Designs, das Implementieren neuer Funktionen oder das Ausbessern von Fehlern wurden über die Zeit im gesamten Arbeitsprozess immer wieder Komponenten bearbeitet oder verworfen, deswegen sind die nächsten Kapitel in einer annähernden Abfolge aufgelistet.

Ziel  
Konzeption und Implementierung einer App für ein Android, welche den/die User\*in eigene Reisen wie in einem Reisetagebuch dokumentieren und kleine Planungen durchführen lässt. Durch eine leichte Bedienung kann der/die Nutzer\*in die Reisen schön und übersichtlich jederzeit wieder aufrufen und Revue passieren lassen.

Motivation

Konzeption  
 Vorgehensweise  
 Bestriebssystem  
- Android  
Frameworks

Zusammenarbeit  
Um einen effizienten miteinander Arbeiten zu ermöglichen, wurden Inhalte, Bemerkungen und weiteres über Milanote geteilt. Zudem konnte sich beide Teammitglieder so selbst durch Backlogs und To-To Listen organisieren. Um das eigene Projekt abzusichern wurde dieses auf Github als Repository angelegt.

Setup

Um eine React Native App zu erstellen, ist es nötig, einige Werkzeuge zu installieren. Dazu gehören Node.js, die Node Package Manager (NPM), React Native Expo oder CLI und ein IDE wie Visual Studio Code oder Atom. Darüber hinaus ist es je nach Zielplattform nötig, die jeweiligen SDKs und das Android oder iOS Emulator zu installieren.

Als Code-Editor wurde mit Visual Studio Code gearbeitet, welcher aufgrund von Benutzerfreundlichkeit und kostenfreier Nutzung den Standard darstellt.

Einrichten der Entwicklungsumgebung

Expo GO  
Es gibt zwei Wege mit React Native zu starten. Erstens mit Expo Go und zweitens mit React Native CLI. Da Expo Go einen schnellen Start in die Entwicklung, besonders für Personen die frisch mit mobiler App-Entwicklung starten verspricht und eine große Auswahl an Tools und Services, die Entwicklern helfen, ihre Apps zu optimieren, zu verwalten und zu skalieren bietet, habe ich mich für diese Option entschieden. Das Einzige worauf hierbei geachtet werden muss ist die Verwendung der richtigen Node Version.

Android Studio   
Damit man sein Projekt nicht nur ausschließlich auf der App Expo Go, die man sich auf sein Android Mobiltelefon installiert, ausführen kann, sondern einen Emulator benutzen will, ist die Verwendung von Android Studio optimal. Dieser stellt Android Mobiltelefon Emulatoren in allen verschiedenen Ausführungen, Formaten und Versionen bereit. Perfekt zum testen von verschiedenen Größen in Bezug auf die Anordnungen der Inhalte. Die Installation von Android Studio ist anfangs fehlgeschlagen und hat einige Probleme verursacht. Nach gründlicher Fehlersuche wurde das Problem bei der Windowsversion 11 identifiziert. Benötigt wurde….

Extensions  
Prettier-Code formatter  
Simple React Snippets  
React Native Tools

Vorgehensweise  
Im August 2022 wurde die Entwicklungsumgebung eingerichtet und ein Zeitplan erstellt, der grob einschätzte, wie viel Zeit für die verschiedenen Funktionalitäten und Phasen der Entwicklung benötigt würde. Man strebte danach, sich an diesen und an die Milestones zu halten, um den Überblick zu behalten und ausreichend Zeit für Tests und Anpassungen zu haben. Da manche Features mehr Zeit in Anspruch nahmen als geplant, mussten Prioritäten gesetzt und alternative Lösungswege gefunden werden. Daher war es für meine Teamkollegin und mich wichtig, dass wir uns an den Zeitplan hielten und unsere Arbeit zügig erledigten. Sie kümmerte sich um das Corporate Design und Layout, welches ich anschließend implementierte. Dadurch konnten wir sicherstellen, dass wir rechtzeitig mit der Entwicklung fertig wurden und alle Funktionen wie geplant getestet und optimiert werden konnten.

App-Entwicklung   
Im Folgendem Abschnitt werden alle Schritte beschrieben, die es benötigt hat, die Applikation zu realisieren. Begonnen beim Aufbau beziehungsweise der Ordnung des Systems zu den verschiedenen Funktionalitäten bis zur Implementierung des Designs.

Aufbau  
Hauptseiten  
Grundsätzlich besteht die Applikation aus 4 Hauptseiten, diese stellen das Gerüst für die gesamte App dar und werden im Navigator abgebildet.   
Home-Screen: Hier wird die wichtigste Information gesammelt, heißt alle erstellten Tagebücher sind hier aufgelistet, der Benutzername und die Statistiken schmücken diese Seite. Außerdem können über einen Button auch neue Tagebücher hinzugefügt werden.   
Reisen-Screen: Diese Seite sammelt ausschließlich alle erstellten Tagebücher und enthält die Funktion neue Reisen hinzuzufügen.  
Listen-Screen: Damit der Funktionsumfang der Applikation ausgeweitet wird gibt es zudem die Funktion To-Do-Listen zu erstellen. Diese werden in dieser Route gespeichert, angezeigt und erstellt.   
Profil-Screen: Hier besteht die Möglichkeit ein Profilbild hochzuladen, einen Nutzernamen zu definieren, sowie einen Hintergrund festzulegen oder weitere Informationen über Datenschutz oder ähnliches auszulesen.

Formulare  
Um ein neues Tagebuch oder einen Eintrag zu erstellen muss ein Formular ausgefüllt werden. Diese werden als Komponente erstellt und können so an verschiedenen Stellen zum Einsatz kommen, ohne den Code wiederholen zu müssen. Dasselbe System wurde bei den meisten Funktionalitäten angewendet.  
Navigation  
Eines der wichtigsten Features einer funktionalen Applikation ist die Navigation. Eine Unterteilung in die wichtigsten Seiten, die Übersicht gewährleistet ist deshalb essenziell. React Native bietet 3 eingebaute Navigatoren an.

Stack Navigator  
Der Stack Navigator bietet die Möglichkeit, zwischen Bildschirmen zu wechseln, wobei jeder neue Bildschirm wie auf einem Stapel ganz oben platziert wird. Um eine einfache Implementierung zu ermöglichen, wurde eine neuer File erstellt, welches alle Seiten auf, die gewechselt werden soll, importiert und jedem einzelnen einen Namen für den Header zuweist. Hier kann jeder Header individuell gestylt werden. Des Weiteren wird hier der Stack-Navigator mit einem einheitlichen selbst designten Header erstellt. Auf dieses kann man nun von jedem anderen Ort zugreifen und verwenden.

Komplikationen  
Obwohl sich der Navigator selbst schnell aufbauen lässt, kam es bei diesem Abschnitt zu einigen Problemen, die viel Zeit in Anspruch genommen haben. Anfangs sollte die Applikation mit einem Bottom-Tab-Navigator für die Hauptseiten und mit einem Stack-Navigator für alle weiteren Bildschirmwechsel in Einsatz kommen. Da die Navigatoren aber im App.js in die Return-Methode als Komponente eingebaut werden muss, hatte das Implementieren von diesen zwei verschiedenen Komponenten zur Folge, dass jedes Mal nur einer dieser funktionsfähig eingebaut werden konnte. Die Lösung dafür ist es Navigatoren zu verschachteln, das bedeutet, dass ein Navigator innerhalb eines Bildschirms eines anderen Navigators dargestellt wird. Dieser Lösungsansatz wurde nach vielen Versuchen jedoch verworfen und darauffolgend durch eine einheitliche Lösung ausgetauscht. Das System wurde auf ausschließlich Stack-Navigatoren umgebaut und der Footer wurde selbst erstellt.

Erstellen eines Eintrag-Systems  
Die Anwendung soll in der Lage sein, personalisierte Tagebucheinträge zu erstellen, die individuell mit Einträgen zu jedem Tag versehen werden können. Zu diesem Zweck wurden mehrere Komponenten konzipiert, die die Tagebücher repräsentieren und durch eine Schleife eingebaut werden können.  
  
Modal  
Das Modal - eine Komponente - ist eine einfache Möglichkeit, Inhalte über einer Ansicht zu präsentieren. Man kann diesem Modal verschiedene Animationen zuweisen und individuell stylen. Damit ein Tagebuch erstellt werden kann müssen vorerst benutzerdefinierte Eingaben getätigt werden. Diese Abfrage wird in einem von React Native bereitgestelltem Modal implementiert. Der/die Nutzer\*in öffnet dieses automatisch nach dem Drücken des Plus-Buttons im Haupt- oder Reisen-Screen. Insgesamt wurden 3 verschiedene Formulare erstellt, die innerhalb eines Modals sichtbar werden.

Das erste zum Erstellen des Tagebuches. Hierfür wurden 6 verschiedene Inputs erstellt: Name, Reiseziel, Reisezeitraum, Titelbild und Reisebeschreibung. Nach dem bestätigen der ausgefüllten Inhalte wird das Tagebuch synchron im Home-Screen und im Reise-Screen erstellt. Die Eingaben werden gespeichert und mitgegeben. Durch die Eingabe von Start- und End-Datum können die Tage berechnet und demzufolge dargestellt werden.   
Das zweite zum Erstellen des Tagebucheintrags. Mit 3 verschiedenen Inputs: Titel, Beschreibung und Fotos hinzufügen.

Das dritte Formular ist dafür zuständig eine Liste mit einem benutzerdefinierten Titel zu erstellen. Hierbei kann lediglich über einen Textinput ein Name für die Liste erstellt werden.

Komplikationen   
Die Formulare wurden zu Beginn mit Formik erstellt, dieses Formula-Bibliothek hilft dabei Formulare zu erstellen, organisieren und validieren. Jedoch hat es beim Versuch states in dieser Komponente zu verwenden Probleme gegeben weswegen dieses Konstrukt wieder verworfen wurde. Das System wurde danach auf die traditionelle Lösung mit Textinputs ohne Formik umgebaut. Deshalb musste demzufolge eine andere Lösung für die Validierung der Eingaben gefunden werden.

Validierung  
Für die Validierung dieser Werte wurde zu Beginn Yup verwendet, ein einfacher Schema-Builder für das Validieren und Parsen von Daten in Echtzeit. Yup ermöglicht die Eingabe einzugrenzen auf Länge, Zahlen, Wörter und weiteres. Da Yup Hand in Hand mit Formik geht musste auch dieser Ansatz der Validierung an zwei Enden wieder verändert werden. Die Listen-Erstellung wurde weiter mit diesem Tool validiert.

Standortabfrage   
Ein weiteres Feature der App stellt die automatische Vervollständigung beziehungsweise Suchfunktion des Reiseortes dar. Im Formular wird die Adresse als Drop-Down Menü dargestellt und kann von dem Nutzenden über einen einfachen Klick ausgewählt werden. Ursprünglich war geplant hierfür Google-Places-Autocomplete zu verwenden im Arbeitsverlauf hat sich dieser Dienst jedoch als kostenpflichtig herausgestellt. Nach langer Suche wurde dann eine Alternative gefunden. Eine nicht allzu bekannte Mapbox-Such-Komponente ausschließlich für React Native. Das Implementieren dieser stellte sich als schwieriger dar als anfangs gedacht. Einen API-Zugangs-Token zu generieren an sich hat keine Probleme gemacht, jedoch das richtige einsetzten dieses und einen gratis Anbieter zu finden ebenso nicht. Das Lösen dieses Problems hat sich über einen längeren Zeitraum gezogen und viele verschiedene Komplikationen mit sich gebracht. Das Stylen des Dropdown-Menüs ist aufgrund Einschränkungen der Komponente leider nicht möglich gewesen. Dennoch kann diese Funktionalität als ein kleiner, aber wichtiger Schritt zu einem benutzerfreundlicheren Erlebnis angesehen werden. Zudem stellt diese Komponente verschiedene Daten zur Verfügung, wie z.B. die Postleitzahl, die Kontinent-ID, usw. auf dem die Berechnung der Statistik aufbaut.

Zugriff auf die Galerie und Kamera  
Ein Bild sagt mehr als tausend Worte. An sie erinnert sich der Mensch leichter als an Texte. Damit der/die Nutzer\*in eine Reise in ihrer ganzen Pracht dokumentieren kann ist es eine Grundvoraussetzung seine/ihre Gedanken mithilfe von Bilder zu unterstützen. Um Zugriff auf die Galerie und die Kamera des Nutzenden zu bekommen wurde ein React Native-Modul verwendet namens React-Native-Image-Picker. Mit diesem Modul wurde Qualität, Format, Medientyp und Bearbeitung über den/die Nutzer\*in bestimmt. Der Medientyp wurde hierbei auf Bilder eingeschränkt. Diese können von allen Speicherorten des Smartphones gewählt werden. Zum Einsatz kommt dieses Modul an drei Stellen. Als erstes beim Upload des Profilbildes, hierbei wird in Echtzeit überprüft ob schon ein Bild festgelegt wurde, wenn dem so ist kann man dieses Bearbeiten beziehungsweise austauschen. An zweiter Stelle wurde der Upload von Bildern beim Erstellen des Tagebuches implementiert. Beim Ausfüllen des Generierungs-Formulars wird die Möglichkeit geboten ein Thumbnail zu erstellen. Der Pfad des gewählten Bildes wird dabei gespeichert und kann so in anderen Komponenten ausgelesen werden und wie hier als Image ein Hintergrundbild darstellen.   
Zuletzt, als wichtigster Anwendungsfall, müssen sich Bilder bei den generierten Einträgen einfügen lassen. Hier wurde ebenso im Formular zum Erstellen des Eintrags gearbeitet. Über einen Button wird der Code getriggered und es lassen sich mehrere Bilder aus der Galerie wählen und ebenfalls wieder löschen. Gleich wie beim Thumbnail wird der Pfad gespeichert und an seiner Einsatzstelle wieder ausgelesen. Mit der Umsetzung dieser Funktionalität wurde sich bereits früh in der Entwicklungsphase der App auseinandergesetzt, da dieses eines der wichtigsten Bestandteilen eines Eintrages wiederspeigelt. Der Research zu den möglichen Optionen hat sich zwar etwas in die Länge gezogen und einige gescheiterte Versuche hinter sich, aber nach der Findung einer funktionalen Lösung war die Implementation kein Problem mehr. Mit dem Speichern der Bildpfade wurde sich im letzten Viertel der Arbeitsphase auseinandergesetzt.

Datum- und Uhrzeit-Auswahlmenü  
Um die Benutzerfreundlichkeit zu erhöhen ist es besonders hilfreich in Apps, in denen Benutzer Termine, Ereignisse oder ähnliche Datumsangaben angeben müssen einen, ein Datums- und Uhrzeit-Auswahlmenü in einer React Native-App zu integrieren. Der react-native-community/datetimepicker ist nach gründlichen Nachforschungen die beste Option für die Einbindung dafür, da er einfach zu bedienen ist und eine Reihe von benutzerdefinierten Optionen bietet. Er ermöglicht es, das Aussehen und Verhalten des Date-Time-Pickers anzupassen, um eine bessere Benutzerfreundlichkeit zu ermöglichen. Darüber hinaus konnte die Auswahl an Datum und Uhrzeit beschränkt werden, um sicherzustellen, dass Daten korrekt eingegeben werden.

Um die Ermittlung der Reisetage so einfach wie möglich zu gestalten, wurden zwei Input-Felder implementiert, eines für das Start- und eines für das Enddatum. Durch die Subtraktion dieser beiden Werte lässt sich die Anzahl der Reisetage ermitteln, nach denen das weitere Schema aufbaut. Um eine korrekte Eingabe zu gewährleisten, wurde eine Validierung implementiert, die verhindert, dass eine negative Anzahl an Reisetagen berechnet wird. Wird dieser Fall erkannt, wird das Formular nicht abgeschickt und es wird eine optische Warnung angezeigt, um auf die fehlerhafte Eingabe hinzuweisen. Durch diese Maßnahme ist es möglich, einen korrekten Ablauf der Reiseplanung zu gewährleisten. Es wird eine schnelle und effiziente Ermittlung der Reisetage ermöglicht, ohne dass der Nutzer viel Zeit und Mühe in die Eingabe investieren muss.

Auch kann so einfach die Uhrzeit ausgelesen werden. Angewandt wird das beim Erstellen des Eintrages. Automatisch wird die Zeit beim Einreichen des Formulars ausgelesen, gespeichert und im erstellten Eintrag angezeigt, ähnlich wie bei einem Tagebucheintrag, der per Hand geschrieben wird.

Bei der Umsetzung dieses Features gab es keine großen Komplikationen, jedoch wurde hier stets an Verbesserungen gearbeitet. In einem frühen Stadium mussten die Reisetage manuell eingegeben werden, was für die Nutzer\*innen nicht optimal war. Wir entdeckten schnell, dass es für die Umsetzung dieses Features notwendig war, den Nutzer\*innen eine visuelle Unterstützung bei der Eingabe der Reisetage zu bieten. Deshalb entschied sich das Team für die Implementierung eines Kalenders, um eine einzigartige Visualisierung der Reisetage zu ermöglichen. Nachdem mehrere Lösungsvorschläge getestet und verworfen wurden, konnte schließlich eine benutzerfreundliche Lösung gefunden und der ursprüngliche Code entsprechend angepasst werden. Dieses Feature ermöglicht es den Nutzer\*innen, mit wenigen Klicks ihre Reise zu planen und wir sind überzeugt, dass die Einführung dieses Features den Anwender\*innen einen einfacheren und benutzerfreundlicheren Workflow ermöglicht.

Bewertungssystem   
Indem man dem User die Möglichkeit bietet seine erstellten Reisen zu bewerten, kann man als Entwickler einer Reisetagebuch-App die User Experience verbessern. Der User erhält ein Feedback über seine eigenen Reiseerlebnisse und kann diese für künftige Reiseplanungen berücksichtigen. Dadurch kann die App ihre Relevanz erhöhen und die Userzufriedenheit steigern. Um die Bewertung jedes einzelnen selbst erstellten Eintrags zu ermöglichen, wurde ein 5-Sternebewertungs-System integriert.

Dafür wurde eine Sternenbewertungskomponente erstellt, die es den Benutzer:innen ermöglicht, eine Bewertung für ein bestimmtes Element abzugeben. Zudem wurde die "Animated"-API verwendet, um eine Animation hinzuzufügen, wenn eine Sternbewertung ausgewählt wird. Der geschriebene Code ermöglicht es dem Benutzer, eine Bewertung von 1 bis 5 Sternen abzugeben, indem einfach auf die Sterne geklickt wird. Wenn der Benutzer einen Stern auswählt, wird ein "spring" -Effekt erzeugt, wobei der Stern größer wird und dann wieder in seine ursprüngliche Größe zurückkehrt. Der Sternbewertungsstatus wird in einem Zustand gespeichert, sodass die ausgewählte Bewertung im Async-Storage gespeichert werden kann.

Listen erstellen  
Packlisten und To-Do-Listen sind nützliche Werkzeuge, um Menschen dabei zu helfen, sich an alles zu erinnern, was sie für ihre Reise benötigen. Durch die Erstellung einer Packliste kann sichergestellt werden, dass alles, was gebraucht wird, nicht vergessen wird, und es kann sich so auf die Reise vorbereitet werden. Zudem bieten sie eine klare Struktur und ermöglichen Aufgaben zu priorisieren und zu planen. Daher ist es dem Team ein Anliegen gewesen, Funktionalitäten zum Planen vor dem Antreten einer Reise einzubauen, um die Anwendung nicht nur auf die Dokumentation während oder nach der Reise zu beschränken. Um dem/der Anwender\*in die Planung einer Reise zu erleichtern, wurden verschiedene Funktionen in die Applikation integriert, die dem Benutzer helfen, sich einen Überblick über die zu erledigenden Aufgaben zu verschaffen und eine gute und effiziente Planung zu ermöglichen. Durch die Umsetzung dieser Funktionalitäten konnten wertvolle Erfahrungen bezüglich der Erstellung und Planung einer Packliste gesammelt werden.

Um das zu ermöglichen wurde ein Listenbildschirm erstellt. Dieser enthält ein Modal, das angezeigt wird, wenn der Benutzer auf den "Neue Liste hinzufügen" -Button drückt, darüber hinaus enthält er eine Flat-List, die die vorhandenen Listen anzeigt. Im Code wurden die Funktionen zum Hinzufügen und Löschen von Listen erstellt, die das AsyncStorage verwenden, um die Daten zu speichern. Es kann also eine Liste erstellt werden die den/die Benutzer\*in Aufgabenelemente hinzufügen lässt. Es gibt auch eine Funktion zum Abhaken von Elementen, mit dem eine Aufgabe als erledigt markiert werden kann und diese in der Hierarchie automatisch nach unten wandert.

Statistiken  
Benutzerabhängige Statistiken ermöglichen es dem User, seine Daten visuell zu verfolgen und so seine Fortschritte zu messen. Indem sichtbar gemacht wird, welche Erfolge gemacht werden, wird der Nutzer motiviert, langfristig die App zu verwenden. Dies bedeutet, dass der/die Nutzer\*in mehr Zeit damit verbringt, die App zu erkunden und mehr Funktionen auszuprobieren, was letztlich zu einer Steigerung der Nutzung führt. Zudem kann es für den/die Benutzer\*in eine zusätzliche Motivation sein, seine Fortschritte zu verfolgen und so seine Ziele zu erreichen.

In der Umsetzung wurden 2 verschiedene Statistiken konzipiert. In der ersten Statistik wird ermittelt, wie viele Länder der Welt der/die Nutzer\*in bereits besucht und dokumentiert hat. In der zweiten Statistik wird die Gesamtzahl aller Urlaubstage dargestellt. Beide Probleme wurden auf ähnliche Weise gelöst, indem die erforderlichen Werte, die beim Erstellen der Formulare ermittelt wurden, asynchron gespeichert wurden, sodass sie überall abgerufen und angezeigt werden können.

Nachdem Nachforschungen betrieben wurden, wurde ein passendes React-Native-Modul gefunden, das es ermöglicht, eine weitere Statistik als Kuchendiagramm darzustellen. Jedoch konnte die Implementierung dieses Moduls aufgrund der begrenzten vorhandenen Zeit nicht mehr in die Arbeit aufgenommen werden.

Modal Walkthrough

Beim ersten Gebrauch einer App ist eine ausführliche Einführung in die Grundlagen der Anwendung und deren Benutzung ein wesentlicher Bestandteil, um den Nutzern den Einstieg zu erleichtern. Dadurch kann das System schneller erfasst und effizient genutzt werden.

Um dieses Feature umzusetzen, wurde das React-Native-Copilot Modul eingesetzt, um eine einfache, intuitive Anleitung in Form eines Schritt-für-Schritt-Walkthroughs zu ermöglichen. Dabei stellte sich heraus, dass es zu Komplikationen kam, als verschiedene Parameter miteinander in Konflikt gerieten. Um zu einem funktionalen Ergebnis zu gelangen, musste der Code mehrmals überarbeitet werden.

Speichersystem

Ein wichtiger Teil der Applikation ist das Speichern der Daten. Dazu habe ich mich, anders als in der Vorfeldanalyse geplant, entschieden die Daten lokal auf dem Gerät der Nutzenden zu speichern. Verwendet wurde dafür das AsyncStorage-Modul. AsyncStorage ist ein einfaches Key-Value-basiertes Speichersystem, das es ermöglicht, Daten im JSON-Format zu speichern.  
  
In der App-Komponente werden vier Zustände, einer für die Tagebücher, einer für die Listen, einer für die Hintergründe und einer für Profibild und Benutzername mit Hilfe des useState-Hooks initialisiert. Diese Zustände werden mithilfe der useEffect-Hook in das AsyncStorage-System gespeichert, indem sie als JSON-Strings in den entsprechenden Schlüsseln (reisen, tasks, backgroundImageNumber und profile) gespeichert werden.

Beim erneuten Laden der App werden die Daten aus dem AsyncStorage-System abgerufen und in den Zuständen gespeichert. Wenn ein Zustand nicht im AsyncStorage-System gefunden wird, wird er auf einen leeren Array, ein leeres Objekt oder eine Standard-Nummer gesetzt, je nachdem, um welchen Zustand es sich handelt.

Für die Implementierung dieses Systems wurde eine erhebliche Menge an Zeit aufgewendet. Es war notwendig, die verschiedenen verfügbaren Möglichkeiten gründlich zu analysieren und diejenigen auszuwählen, die am besten geeignet sind, um die Anforderungen des Systems zu erfüllen. Besonders das Speichern der Sternebewertung und des Textinputs stellte in diesem Teil des Prozesses unerwartete Hürden und Schwierigkeiten dar, die es zu überwinden galt.

Darüber hinaus war es notwendig, das Schema des Systems sorgfältig zu planen und zu definieren, um sicherzustellen, dass alle erforderlichen Daten in einer effektiven und effizienten Weise gespeichert und abgerufen werden können. Dank der Zusammenarbeit mit einem erfahrenen Kollegen, der bereit war, seine Expertise und Erfahrung einzubringen, um bei der Umsetzung des Schemas zu helfen konnte dieser wichtige Part erfolgreich abgeschlossen werden.

Während des gesamten Entwicklungsprozesses mussten ständige Anpassungen und Verbesserungen vorgenommen werden. Insgesamt war die Implementierung dieses Systems ein aufwändiger und herausfordernder Prozess, der jedoch durch eine gründliche Planung und Zusammenarbeit gelungen bewältigt wurde.

Design  
Im nächsten Abschnitt wird darauf eingegangen wie die von meiner Teamkollegin ausgearbeitet Designlösungen in die Applikation implementiert wurden.

Zusammenarbeit

Wir haben unseren Zeitplan so ausgerichtet, dass meine Teamkollegin in den ersten Wochen am UX-Design arbeitet und danach auf dieser Grundlage das Design ausarbeitet. Parallel dazu habe ich begonnen, die Grundfunktionalitäten der Applikation aufzubauen. Auf diese Weise konnten wir nach der ersten Phase schon die Inhalte austauschen, auf die die App aufbaut, und gleichzeitig wichtige Aufgaben erledigen und unsere Zeit effizient nutzen. Die ausgearbeiteten Entwürfe haben wir als Adobe XD Projekt geteilt, wodurch ich direkt Größen, Farbcodes und Buttonlogik auslesen konnte.

Globales Style-Sheet

Damit nicht für jede einzelne Komponente dieselben Style-Regeln definiert werden müssen wurde ein globales StyleSheet erstellt. Dieses JavaScript-Objekt, enthält eine Sammlung von Styling-Regeln, für die Komponenten zum Beispiel verschiedene Eigenschaften wie Farben, Schriftarten, Abstände usw., die auf der gesamten Anwendung angewendet werden sollen. Auf diese Weise wurde nicht nur das Styling der Anwendung konsistent gehalten und Duplikation von Code vermieden, sondern auch die Wartbarkeit und Lesbarkeit des Codes wurde verbessert. Wenn man beispielsweise eine Änderung an einer Styling-Regel vornehmen musste, konnte man dies einfach an einer zentralen Stelle im globalen StyleSheet tun, anstatt jede einzelne Komponente durchzugehen und das Styling manuell zu ändern.

Ausklappbare Textelemente

Bei der Entwicklung einer benutzerfreundlichen Anwendung werden häufig ausklappbare Textelemente benötigt. Es sollte ermöglicht werden, dass Inhalte bei einem Klick ein- und ausgeblendet werden können, beispielsweise im Help-Center für FAQs oder bei den erstellten Tagebucheinträgen. Zur Realisierung dieses Vorhabens wurde das Accordion-Collapse Modul eingesetzt, eine Bibliothek für React Native, welche es ermöglicht, ausklappbare Komponenten schnell und einfach zu erstellen. Durch den Einsatz dieser Bibliothek wird den Benutzern ermöglicht, große Mengen an Inhalten in einer geordneten und strukturierten Weise zu präsentieren.

Die verwendete Accordion-Collapse Bibliothek bietet eine nutzerfreundliche und intuitiv bedienbare API, welche es ermöglicht, die Akkordeon-Komponenten einfach und schnell anzupassen.

Schriftarten  
Da eine IOs-Schriftart vorgesehen war musste diese richtig importiert und eingebaut werden. Um das zu ermöglichen, wurden in der App-Komponente alle Schriftarten beliebig benannt und der Pfad angegeben. Somit ist es nun in jeder Komponente mühelos möglich, auf die gewünschte Schriftart zuzugreifen.

Hintergründe  
Im Rahmen unserer Anwendung wurde die Möglichkeit implementiert, dass der Benutzer den Hintergrund der Anwendung nach seinen Wünschen anpassen kann. Hierfür hat meine Teamkollegin sechs unterschiedliche Hintergründe erstellt, die dem Benutzer zur Auswahl gestellt wurden. Um die Auswahl des Hintergrunds zu erleichtern, wurde ein eigener Screen eingerichtet, auf den der Benutzer über die Einstellungen der Anwendung gelangen konnte. Dort hatte der Benutzer die Möglichkeit, zwischen den verschiedenen Hintergründen zu wählen und somit das Erscheinungsbild der Anwendung individuell zu gestalten.  
Durch die Einbindung dieser Option konnte die Nutzerfreundlichkeit der Anwendung erheblich gesteigert werden, da der Benutzer die Möglichkeit hatte, das Erscheinungsbild der Anwendung an seine persönlichen Vorlieben anzupassen. Zudem trägt die Auswahlmöglichkeit der Hintergründe dazu bei, dass die Anwendung auch ästhetisch ansprechend gestaltet werden kann und somit einen positiven Eindruck beim Benutzer hinterlässt.

Die Implementierung dieser Funktion erforderte eine sorgfältige Planung und Umsetzung, um sicherzustellen, dass die Hintergründe korrekt dargestellt und ausgewählt werden können. Hierbei wurden bewährte Methoden und Technologien eingesetzt, um eine hohe Funktionalität und Benutzerfreundlichkeit zu gewährleisten.

Es wurde eine Ansicht konzipiert, die es ermöglicht, ein Hintergrundbild auszuwählen. Nach der Auswahl eines Bildes wird eine zugehörige Nummer gespeichert. Diese Nummer wird dann in allen Bildschirmen abgerufen, die einen Hintergrund enthalten, und das entsprechende Hintergrundbild wird zugewiesen. Um die Auswahl im lokalen Speicher des Geräts zu speichern, wurde der StoreContext genutzt.

Komplikationen

Während der Implementierung einiger Designentscheidungen traten unvorhergesehene Schwierigkeiten auf, insbesondere bei der Ausrichtung mehrerer Elemente. Trotz der Anwendung aller erforderlichen Schritte zur Zentrierung eines Objekts konnte dessen Positionierung nicht immer vollständig korrekt umgesetzt werden, was zu einer langwierigen und anspruchsvollen Baustelle führte.   
Des Weiteren ergaben sich Schwierigkeiten bei der Darstellung von Elementen in einer horizontalen Ausrichtung innerhalb einer Flatlist. Wie sich später herausstellte, gelten hierbei andere Regeln als bei der üblichen Verwendung von Flexdirection und es war notwendig, alternative Lösungen zu finden.   
Aufgrund der Komplexität des Tagebuch-Konstrukts, das Reisen und Einträge miteinander verknüpft, traten bei selbst kleinen Design-Änderungen häufig Schwierigkeiten auf. Die Lösung dieser Probleme erforderte kontinuierliche Anpassungen und Neugestaltungen. Die Darstellung der Inhalte auf dem Home-Screen und dem Reisen-Screen erforderte dazu ein kooperierendes System. Beide Bildschirme stellen zu einem Teil dieselben Inhalte dar mussten aber unterschiedlich dargestellt werden. Dies erforderte viel Zeit und es hätte wahrscheinlich eine effizientere Lösung geben können.  
  
Ein weiteres Hindernis ergab sich bei der Verwendung des von React Native bereitgestellten Headers zur Navigation innerhalb der Anwendung mittels des Stack-Navigators. Da unser Design keinen Header vorsieht, sollte dieser ausgeblendet werden. Allerdings wurde festgestellt, dass das Hintergrundbild der Anwendung trotzdem nicht hinter diesem unsichtbaren Header dargestellt wurde, wenn dieser transparent gemacht wurde. Dies führte zu Schwierigkeiten, für die im vorgesehenen Zeitrahmen keine passende Lösung gefunden wurde.

Ein paar weitere Probleme traten beim Arbeitsprozess auf, darunter die korrekte Darstellung des Datums im Hinblick auf die standardisierte amerikanische Datumsangabe. Mehrere Stunden wurden diesem Problem gewidmet, doch leider konnte keine zufriedenstellende Lösung erzielt werden. Des Weiteren machte der Hintergrund eines Formulars Probleme. Auf unerklärliche Weise ließ er sich nicht auf dieselbe Weise wie auf anderen Seiten reproduzieren.

Fazit