Technische Dokumentation

Studienarbeit

Contract Manager App

Semester: Herbstsemester 2022

Version: 01.00
Datum: 2022-12-23 14:59:34Z
Git Version: 678912b

Projekt Team: André Blöchlinger
Simon Canal
Linard Vincenz

Projekt Betreuer: Prof. Frank Koch

Industriepartner: Michael Güntensperger (AdaptIT GmbH)

OST – Ostschweizer Fachhochschule

Abstract

In der heutigen Zeit werden viele Verträge digital abgeschlossen. Zudem gibt es immer mehr Dienste, welche als Mietmodell angeboten werden und Zahlungen direkt auf der Kreditkarte oder dem Konto belasten. Da kann ein Vertrag schon mal aus dem Blickfeld geraten. Um eine bessere Übersicht über seine Verträge zu erhalten, soll ein zentraler Speicherort geschaffen werden, welcher über anstehende Kündigungstermine aufmerksam macht.

Das Frontend wurde in Form eines Mobile Apps mithilfe Googles plattformübergreifendem Software Development Kit Flutter realisiert. Dieses basiert auf der Programmiersprache Dart. Die Backend-Umgebung ist umgesetzt als REST-API, welche mit NodeJS entwickelt wurde. Dabei wurde TypeScript als Sprache verwendet, da dieses die Typensicherheit gewährleistet. Das Backend kommuniziert mit dem Frontend über einen Caddy Reverse-Proxy und der MongoDB Datenbank. Die Datenbank sowie das Backend werden auf dem Cloudhosting Provider DigitalOcean betrieben. Das Backend sowie der Reverse-Proxy und die Datenbank sind containerisiert mit Docker. Damit ein Passwort-Reset vollzogen werden kann, hat man SendGrid als externe Mailing API verwendet. Für die Push-Benachrichtigung sowie für den Google Analytics Service wurde Firebase genutzt.

Mithilfe der Contract Manager App können Verträge erfasst sowie anderen Familienmitgliedern zugewiesen werden. Die Verträge sind kategorisierbar und werden auf einem Dashboard anschaulich dargestellt. So hat man immer einen Überblick über die aktuellen monatlichen Kosten. Insbesondere hilft die App, auslaufende Verträge und mögliche Kündigungstermine im Blick zu behalten. Dies geschieht mittels Push-Benachrichtigung und ist zusätzlich in der App ersichtlich. Dadurch bietet die App eine optimale Aufbewahrung von Verträgen für Einzelpersonen und Familien an.

Inhaltsverzeichnis

1 Management Summary **1**
1.1 Ausgangslage . . . 1
1.2 Vorgehen . . . 1
1.3 Ergebnisse . . . 2
1.4 Ausblick . . . 3

2 Anforderungen **4**
2.1 Funktionale Anforderungen . . . 4
2.2 Nicht-Funktionale Anforderungen . . . 5

3 Design **10**
3.1 Domain Model . . . 10
3.2 Systemarchitektur . . . 11
3.3 Sprachen, Libraries, Frameworks . . . 16
3.4 Sequenzdiagramme . . . 18
3.5 Schnittstellen . . . 25
3.6 Deployment . . . 25
3.7 Wireframes . . . 28

4 Implementierung **33**
4.1 Projektstruktur . . . 33
4.2 Qualitätsnachweis . . . 34
4.3 Ergebnisse . . . 36
4.4 Einschränkungen . . . 51

5 Weiterentwicklung **52**
5.1 Optionale Anforderungen . . . 52

6 Projektplanung **53**
6.1 Einführung . . . 53
6.2 Projektorganisation . . . 53
6.3 Zeitmanagement . . . 54
6.4 Meilensteine . . . 55
6.5 Iterationen (Sprints) . . . 56
6.6 Besprechungen . . . 60
6.7 Risikomanagement . . . 60
6.8 Arbeitspakete . . . 62
6.9 Qualitätsmassnahmen . . . 63
6.10 Projektmonitoring . . . 64

7 Schlussbericht **69**
7.1 Erfahrungsberichte . . . 69
7.2 Danksagung . . . 70
7.3 Fazit . . . 70

A Verzeichnisse **72**
Abbildungsverzeichnis 72
Tabellenverzeichnis 73
Literatur 75

B Anhang **77**

1. Management Summary

1.1 Ausgangslage

Ob für Einzelpersonen oder Grossfamilien, es fallen viele verschiedene Verträge an. Es ist leicht, den Überblick über all diese Verträge zu verlieren. Man muss pünktlich die Rechnungen bezahlen sowie die Auslaufzeit eines Vertrages im Auge behalten. Auch Kündigungsfristen müssen eingehalten werden. In dieser Studienarbeit hat man sich mit diesen Problemen auseinandergesetzt.

Das Ziel dieser Arbeit ist eine funktionierende mobile Applikation zu erstellen, der Contract Manager. Mit der App soll man Verträge erfassen können, welche man kategorisieren und Familienmitgliedern zuweisen kann. Die App soll auch einen Kostenüberblick aufzeigen. Des Weiteren soll die App über Push-Benachrichtigungen auf auslaufende Verträge und mögliche Kündigungstermine hinweisen. Zusätzlich hat diese Arbeit zum Ziel, aus der Entstehung, der Durchführung und der Fertigstellung des Softwareprojekts reichlich Erfahrungen zu sammeln. Zugleich sollen technische Fachkenntnisse dazugewonnen werden. Der gesamte Projektverlauf wird dokumentiert und zu einer akademischen Arbeit verfasst. Dies soll für den Auftakt und den erfolgreichen Abschluss der Bachelorarbeit dienen, welche im nächsten Semester ansteht.

1.2 Vorgehen

Zu Beginn der Studienarbeit hat man die vorgegebenen Anforderungen mit dem Industriepartner und dem Betreuer besprochen und allfällige Fragen geklärt. Ausserdem wurden ein Projektplan erstellt und Meilensteine festgelegt. Hinzu kamen die Bestimmungen der Technologien, welche im Projekt verwendet werden sollten.

Nach dieser ersten Projektphase ging man ins Entwickeln einer Softwarearchitektur über. Die ersten Entwürfe für die visuelle Gestaltung der Applikation wurden erstellt und zusammen mit der Architektur fortlaufend angepasst. Der erste Prototyp wurde erstellt, um zu testen, ob alle Technologien miteinander kompatibel sind und keine unangenehmen Überraschungen während der Entwicklungsphase zum Vorschein kommen.

Im weiteren Verlauf des Projekts wurden die Anforderungen implementiert und abgearbeitet. Das Projektmanagement wurde mit der agilen Methode Scrum+ im Auge behalten. Man einigte sich auf wöchentliche Iterationen, damit man über den aktuellen Fortschritt berichten und die weitere Planung vornehmen konnte.

Quelle: Alle Icons sind Eigentum des jeweiligen Rechteinhabers

Abbildung 1.1: Verwendete Technologien

1.3 Ergebnisse

Von den vorgegebenen Anforderungen konnte ein Grossteil umgesetzt werden. Zum Teil mussten Anforderungen aufgrund von Zeitmangel in Absprache mit dem Industriepartner und dem Betreuer gestrichen oder reduziert werden. Das Projektteam hat dies bereits im Projektverlauf bemerkt und so wurde eine Neueinschätzung gemacht.

Mit der abschliessenden Version des Contract Managers ist es den Benutzenden möglich, sich zu registrieren und anzumelden. Ausserdem kann man Verträge erfassen und diese vorgängig erstellten Familienmitgliedern zuweisen. Auf dem Dashboard hat man eine nützliche Auflistung der monatlichen Kosten. Die Verträge sind nach Kategorie aufgeteilt und werden in einem Diagramm dargestellt. Damit die Benutzenden keine Kündigungstermine mehr verpassen, erhält man frühzeitig eine Push-Nachricht auf das Mobilgerät gesendet. Dies geschieht auch bei auslaufenden Verträgen. Die App wird momentan als interner Test im Google Play Store angeboten. So ist es für Android-Nutzer möglich, die App herunterzuladen. Jedoch wurde die App so konzipiert, dass sie ohne grossen Aufwand auch für Apple's iOS-Geräte wie auch im Web nutzbar gemacht werden kann.

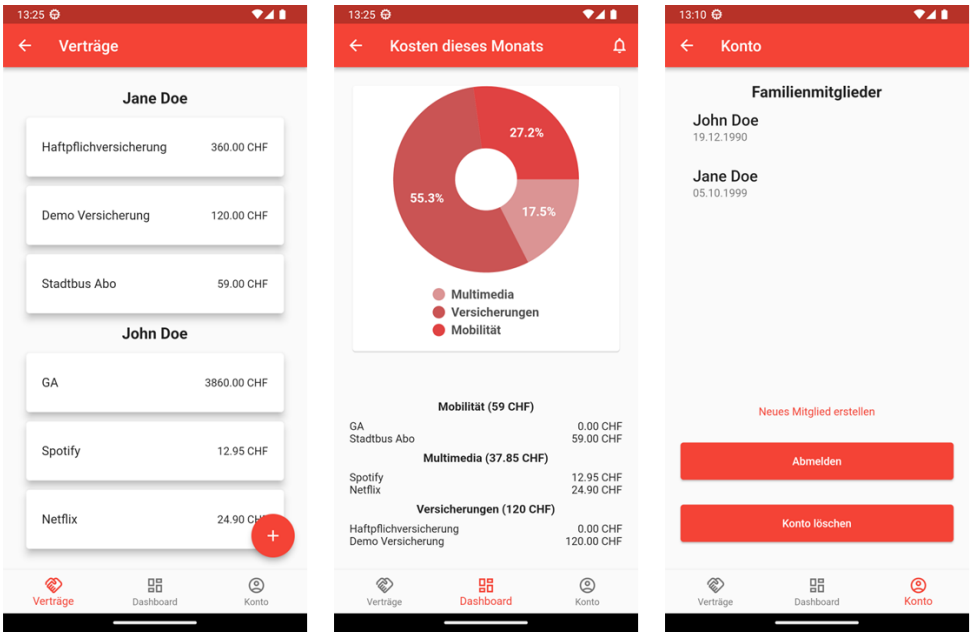

Abbildung 1.2: Screenshots der Contract Manager App

1.4 Ausblick

Die Contract Manager App hat noch viel Potenzial. Einerseits wären hierbei die gestrichenen Anforderungen zu erwähnen. Andererseits sind noch etliche optionale Anforderungen vorhanden, welche ebenfalls spannende Erweiterungen darstellen könnten. Auch im Bereich Design kann man vieles noch verbessern. Zudem könnte man eine barrierefreie Version der App entwickeln, damit möglichst viele Personen die App optimal Nutzen können.

2. Anforderungen

In diesem Kapitel wird auf die funktionalen sowie die nicht-funktionalen Anforderungen eingegangen.

2.1 Funktionale Anforderungen

Die nachfolgenden funktionalen Anforderungen (FR) stellen Produkteigenschaften oder Funktionen dar, welche implementiert werden, damit die Benutzenden ihre Aufgaben erfüllen können. Diese Anforderungen wurden vor dem Projektstart definiert.

ID	Beschreibung
FR1	Benutzende können sich registrieren. Dabei wird E-Mail-Adresse, Passwort, Vorname, Nachname und Geburtsdatum angegeben.
FR2	Benutzende können sich mit einem registrierten Account einloggen.
FR3	Benutzende können ihren Account löschen.
FR4	Benutzer können Verträge erfassen. Dabei wird Name, Beschreibung, einmalige Kosten, wiederkehrende Kosten, erstmöglicher Kündigungstermin, Vertragsbeginn, Vertragsende, Zahlungsperiode, Kündigungsperiode definiert.
FR5	Benutzende können einem Vertrag Dokumente beifügen.
FR6	Benutzende können zusätzliche Personen in ihrem Konto erfassen.
FR7	Benutzende können Verträge anderen Personen ihres Kontos zuweisen.
FR8	Benutzende können sich in einem Dashboard einen Überblick über ihre Verträge machen.
FR9	Benutzende werden via Push-Benachrichtigungen über Kündigungstermine ihrer Verträge informiert.
FR10	Benutzende können ihr Passwort zurücksetzen (Passwort vergessen Funktionalität).
FR11	Benutzende können die Applikation in Google's Play Store oder in Apple's App Store herunterladen.
FR12	Administratoren können eine Auswertung über das Nutzverhalten der Benutzer des Contract Manager Apps einsehen.

Tabelle 2.1: Funktionale Anforderungen

2.2 Nicht-Funktionale Anforderungen

Folgende nicht-funktionalen Anforderungen (NFR) wurden vor Projektbeginn definiert. Sie legen Grundsätze fest, wie das Produkt am Ende sein soll. Die Auflistung ist angelehnt an den ISO-25010-Standard [8].

2.2.1 Performance Efficiency

ID:	NFR-1
Anforderung:	Das Laden einer Seite sollte in einer angemessenen Zeitspanne vonstattengehen.
Auslöser:	Genervte Benutzer aufgrund von langen Ladezeiten.
Reaktion:	Nutzung eines zuverlässigen und schnellen Webhostings und einer optimierten Datenbank.
Messungen:	Jede Seite sollte nicht länger als 150ms für das Laden benötigen

Tabelle 2.2: Nicht-Funktionale Anforderung 1

ID:	NFR-2
Anforderung:	Das Backend soll eine angemessene Anzahl an Anfragen bewältigen können.
Auslöser:	Das System ist überladen und Anfragen werden nicht ausgeführt. Das System bricht zusammen aufgrund von zu vielen Anfragen.
Reaktion:	Begrenzen der Anfragen pro Zeitpunkt oder Erhöhung der CPU und Speicherbedarf.
Messungen:	Das Backend sollte 500 Requests pro Minute handeln können

Tabelle 2.3: Nicht-Funktionale Anforderung 2

ID:	NFR-3
Anforderung:	Die Datenbank hat genug Kapazität.
Auslöser:	Datenbank kann bei Überlauf keine neuen Daten mehr aufnehmen.
Reaktion:	Datenbank mit genügend Speicherplatz definieren.
Messungen:	Die Datenbank soll bis zu 10'000 Verträge und 500 Benutzer managen können.

Tabelle 2.4: Nicht-Funktionale Anforderung 3

2.2.2 Usability

ID:	NFR-4
Anforderung:	Features sollen gemäss Priorität abgearbeitet werden.
Auslöser:	Das Entwicklerteam implementiert Features, welche der Kunde nur bedingt möchte oder für wenig relevant hält.
Reaktion:	Absprache mit Kunden und Features priorisieren.
Messungen:	Das Entwicklerteam implementiert die Features gemäss der abgesprochenen Priorität mit dem Kunden.

Tabelle 2.5: Nicht-Funktionale Anforderung 4

ID:	NFR-5
Anforderung:	Das Erfassen eines Vertrages soll effizient sein.
Auslöser:	Zu grosse Zeitinvestition in die Vertragserstellung wirkt sich negativ auf die Usability aus.
Reaktion:	Schnelle und einfache Vertragserstellung soll möglich sein
Messungen:	Vertragserstellung soll maximal 30 Sekunden dauern.

Tabelle 2.6: Nicht-Funktionale Anforderung 5

ID:	NFR-6
Anforderung:	Die App soll benutzerfreundlich sein.
Auslöser:	Applikationen mit wenig Benutzerfreundlichkeit werden vom User ungern genutzt.
Reaktion:	Vor dem Deployment soll die App getestet werden.
Messungen:	Drei von vier Test-Usern sollten das UI (Kategorien: layout, responsiveness, colour, content) der Applikation mit iPhone / Android Handy mit einer Note von mindestens 8 von 10 bewerten, wobei 10 das Beste ist.

Tabelle 2.7: Nicht-Funktionale Anforderung 6

ID:	NFR-7
Anforderung:	Die App muss betriebsbereit sein.
Auslöser:	Eine nicht-betriebsbereite App hat wenig Nutzen.
Reaktion:	Die App für den Betrieb vorbereiten und laufen lassen.
Messungen:	Implementierte Funktionalität (Datenbank, Backend) sollen deployed und die App bei Google als interner Test und in Apple Testflight eingerichtet werden.

Tabelle 2.8: Nicht-Funktionale Anforderung 7

2.2.3 Reliability

ID:	NFR-8
Anforderung:	Errors dürfen nicht zum Absturz führen.
Auslöser:	App stürzt aufgrund eines Errors ab.
Reaktion:	Error-Handling implementieren.
Messungen:	Errors sollen keine Systemfehler erzeugen, aber eine Error Nachricht Zeigen und das System auf den vorherigen Zustand zurücksetzen.

Tabelle 2.9: Nicht-Funktionale Anforderung 8

ID:	NFR-9
Anforderung:	Errors müssen einsehbar sein.
Auslöser:	Errors treten auf.
Reaktion:	Error-logging implementieren.
Messungen:	Jeder Error soll im System geloggt werden.

Tabelle 2.10: Nicht-Funktionale Anforderung 9

2.2.4 Security

ID:	NFR-10
Anforderung:	Die App soll keine unerlaubten Eingaben erlauben.
Auslöser:	Unerlaubte Eingaben können zu Datenlecks führen.
Reaktion:	Eingaben müssen validiert werden.
Messungen:	Daten, welche in Eingabefelder abgefüllt werden, sollen zuerst validiert werden, bevor diese durch das System verarbeitet werden. SQL Injection test der Eingabefelder sollte keine Verletzlichkeiten zeigen.

Tabelle 2.11: Nicht-Funktionale Anforderung 10

ID:	NFR-11
Anforderung:	Sensitive Daten müssen in der Datenbank geschützt werden.
Auslöser:	Bei einem Angriff entweichen Daten aus der Datenbank.
Reaktion:	Passwörter als sicherer Hash nicht im Klartext in der Datenbank speichern.
Messungen:	User-Passwörter werden nicht in plain-text in der Datenbank gespeichert.

Tabelle 2.12: Nicht-Funktionale Anforderung 11

ID:	NFR-12
Anforderung:	Ein User hat nur Zugriff auf seine Daten.
Auslöser:	Ein User missbraucht Daten, für welche er keinen Zugriff haben sollte.
Reaktion:	User müssen sich autorisieren.
Messungen:	Wenn sich ein User in die App einloggt, werden ihm auch nur seine Daten / auf Daten, die er Zugriff haben soll, angezeigt.

Tabelle 2.13: Nicht-Funktionale Anforderung 12

2.2.5 Maintainability

ID:	NFR-13
Anforderung:	Die App soll einfach weiterentwickelt werden können.
Auslöser:	Die Implementationsstruktur verwirrt Entwickler und sie brauchen viel mehr Zeit für die Weiterentwicklung.
Reaktion:	ModuLarer Aufbau der Implementationsstruktur.
Messungen:	Businesslogik im Backend soll modular aufgebaut werden, sodass sie erweitert werden kann.

Tabelle 2.14: Nicht-Funktionale Anforderung 13

ID:	NFR-14
Anforderung:	Backend-Zugriffe müssen getestet werden.
Auslöser:	Gewisse Funktionen funktionieren nicht wunschgemäss.
Reaktion:	Testen der Backend-API.
Messungen:	Die Backend-API soll durch API-testing Tools getestet werden.

Tabelle 2.15: Nicht-Funktionale Anforderung 14

2.2.6 Portability

ID:	NFR-15
Anforderung:	Das App soll auf mehreren Betriebssystemen laufen.
Auslöser:	Wichtigste Betriebssysteme im mobilen Bereich.
Reaktion:	Nutzung eines cross-platform Frameworks.
Messungen:	Die App soll auf Android und iOS laufen.

Tabelle 2.16: Nicht-Funktionale Anforderung 15

3. Design

In diesem Kapitel wird auf das Design der Architektur sowie des UI eingegangen.

3.1 Domain Model

Die nachfolgende Abbildung zeigt das Domain Model der Contract Manager App.

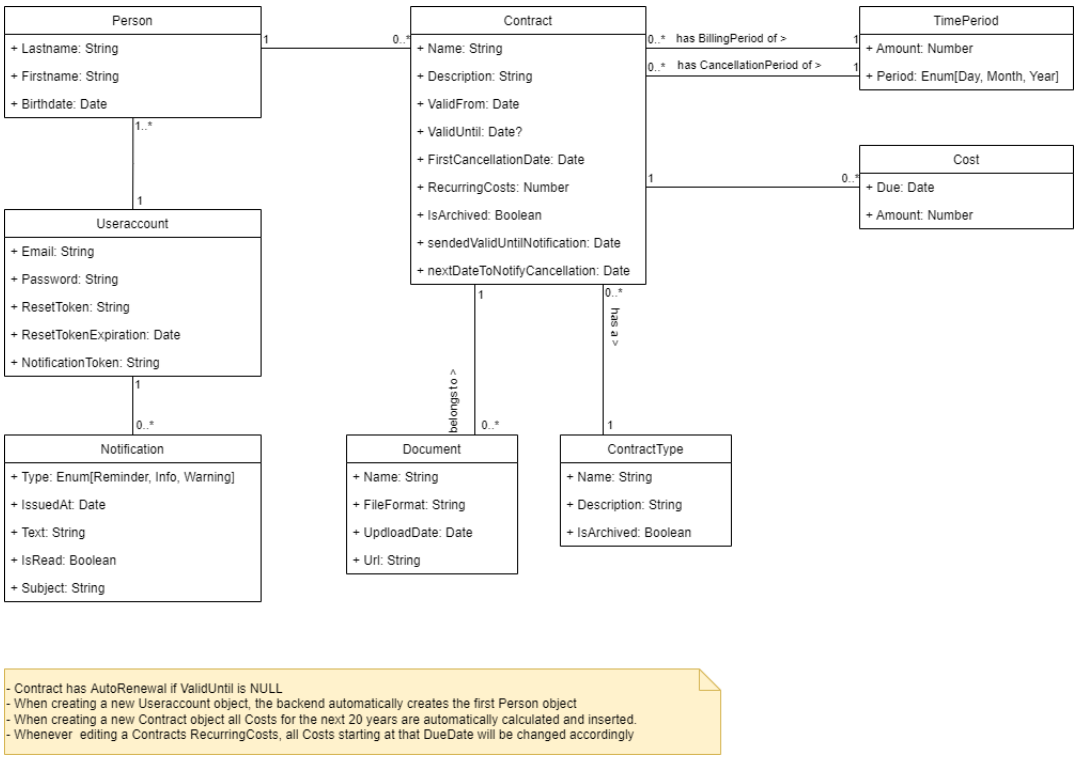

Abbildung 3.1: Domain Model

3.2 Systemarchitektur

Um die Systemarchitektur darzustellen, werden Teile des C4-Modells verwendet.

3.2.1 System-Kontext Diagramm

Der Contract Manager kommuniziert mit drei Akteuren. Nebst dem eigentlichen Benutzer der Applikation ist dies SendGrid[15] als externes E-Mail System und Google Firebase[11], welches für das Verarbeiten der Push Benachrichtigungen und die Anbindung an Google Analytics zuständig ist. Die externen Systeme werden via HTTPS Protokoll angesprochen. Die Daten werden dabei jeweils im JSON Format übertragen.

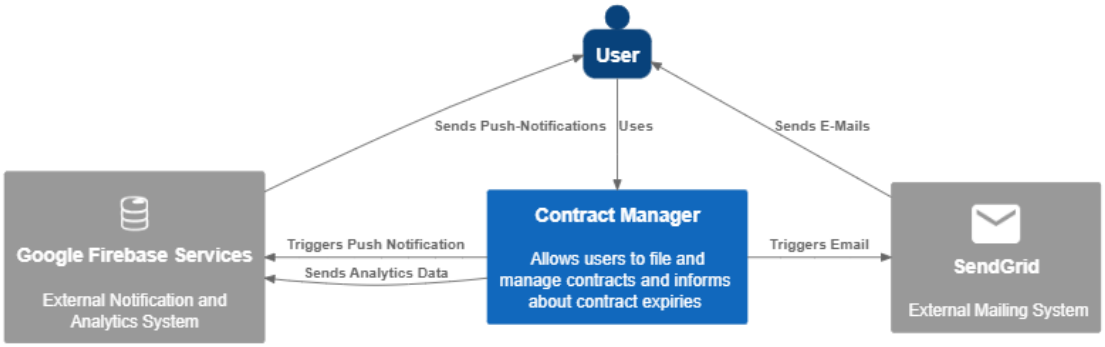

Abbildung 3.2: System Context Diagramm

3.2.2 Container Diagramm

Im Container Diagramm ist der interne Aufbau der Applikation ersichtlich. Der Benutzer interagiert über das Frontend, eine Flutter[12] Mobile App, mit dem System.
Die Flutter App kommuniziert über HTTPS mit einem Caddy[3] Reverse Proxy. Dieser leitet die Anfragen weiter an das NodeJS[2] API, welches dann die Anfragen bearbeitet. Zur Persistierung der Daten ist eine MongoDB[6] Datenbank angebunden, welche über Mongoose[14] angebunden ist.
Das Backend, bestehend aus Reverse Proxy, API und Datenbank wird auf Digitalocean[10] in einem Ubuntu Droplet mit Docker[4] betrieben. Dabei sind die Komponenten in separaten Docker Containern untergebracht.

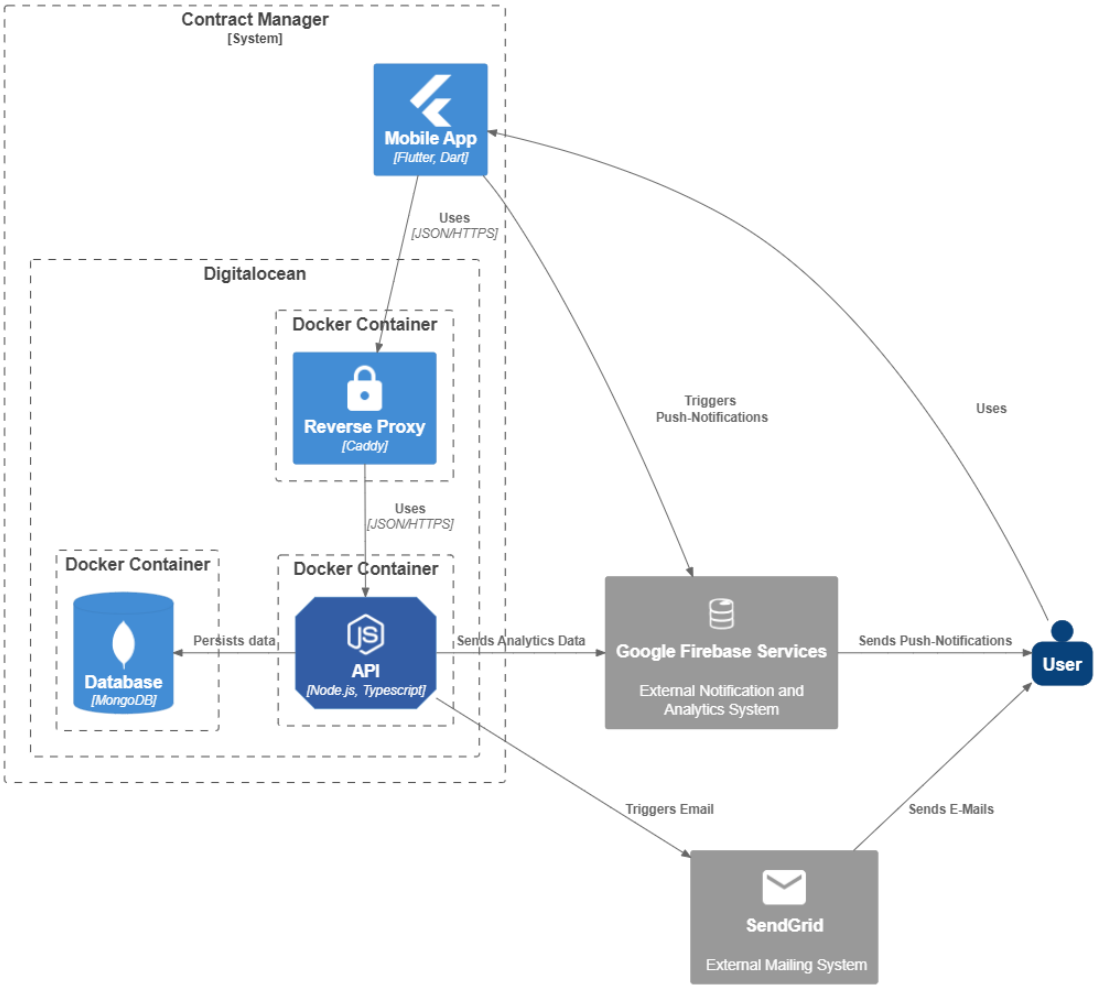

Abbildung 3.3: Container Diagramm

3.2.3 Komponenten Diagramme

Die einzelnen Komponenten des Contract Manager werden im Komponenten-Diagramm detaillierter betrachtet.

Mobile App

Das Frontend des Contract Manager wird als Cross-Plattform Mobile App mit Flutter entwickelt. Somit kann die Applikation sowohl mit Android als auch mit iOS verwendet werden. Zur sauberen Trennung von UI-Code und Businesslogik wird das MVVM Pattern angewendet.

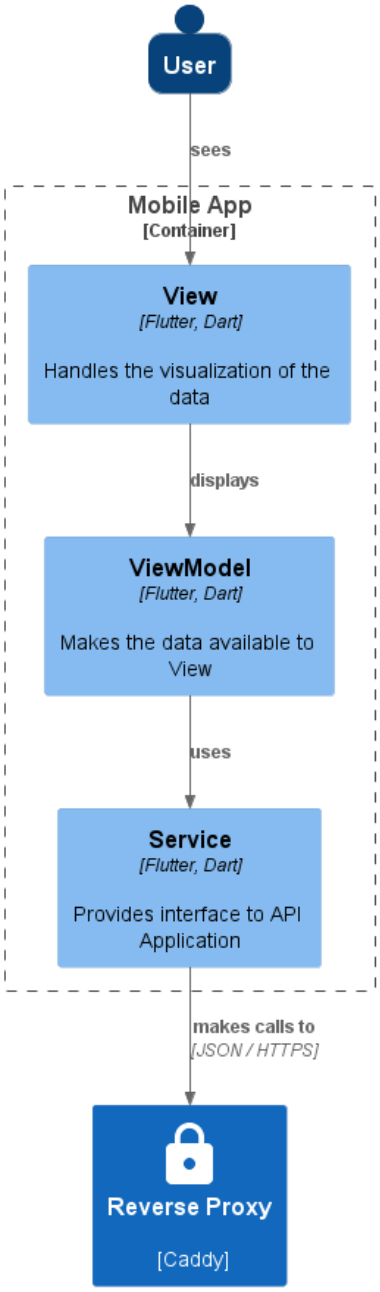

Abbildung 3.4: Komponenten Diagramm - Mobile App

API

Die Backend-Komponente wird mit NodeJS und dem Webframework Express.js realisiert. Für zusätzliche Typensicherheit wird die Programmiersprache TypeScript verwendet. Konzeptionell wird die API pro Aufgabenbereich jeweils in Controller-, Service- und Repositoryschicht aufgeteilt. Zudem sind diverse Middlewares im Einsatz, um die Authentifizierung, Autorisierung sowie Inputvalidierung vorzunehmen.
Die Controller sind für das Handling der Anfragen zuständig. In der Serviceschicht werden Daten transformiert und von Modell Klassen auf DTO Klassen und umgekehrt gemappt. Zugriffe auf die Datenbank passieren ausschliesslich in den Repositories.

Zudem werden für den Passwortreset benötigte HTML Views über NodeJS bereitgestellt.

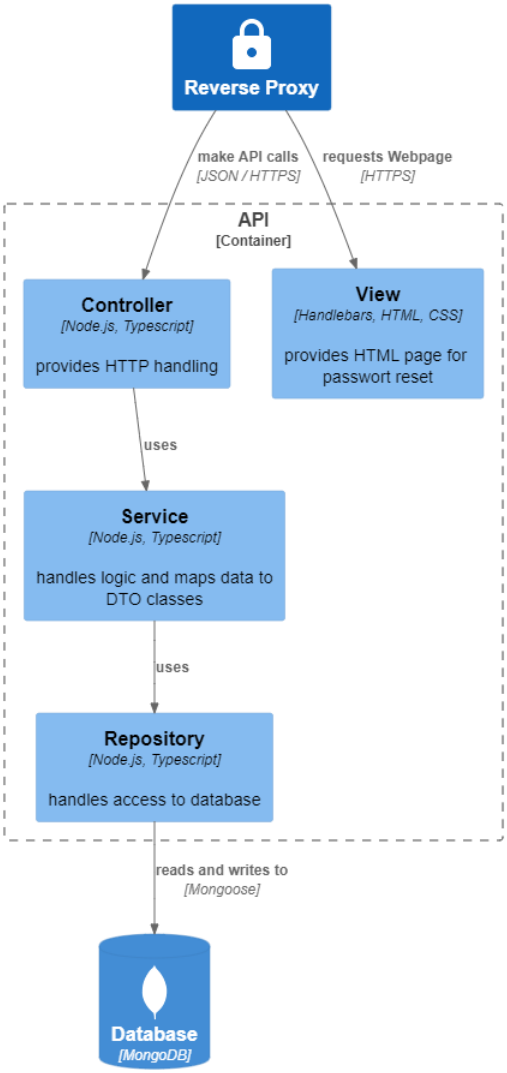

Abbildung 3.5: Komponenten Diagramm - API

3.2.4 Konzepte

Sowohl im Frontend als auch im Backend werden zum Teil Konzepte eingesetzt, welche in den Architektur-Diagrammen nicht dargestellt werden können. Auf diese Konzepte wird in diesem Kapitel näher eingegangen.

Middlewares

Im Backend werden für die Authentifizierung, Autorisierung sowie die Inputvalidierung Middlewares eingesetzt. Die Express.js Library bietet sich für die Verwendung von Middlewares unter NodeJS an und wird daher eingesetzt.

DTO Klassen

Für den Datenaustausch von Frontend und Backend werden DTO Klassen implementiert. So werden nur die Daten übermittelt, welche auch wirklich benötigt werden. Dies ist aus Security- aber vor allem auch aus Performancegründen sinnvoll. Zudem wird durch die Verwendung von DTOs eine Unabhängigkeit gegenüber den Datenbanktabellen bzw. Modell Klassen erreicht, was ein grosser Vorteil im Bereich Maintainability darstellt.

Beim Mapping zwischen den DTO- und den Modell-Klassen wird bewusst auf den Einsatz von Libraries und Packages verzichtet. Dadurch werden weitere externe Abhängigkeiten vermieden.

Docker und Dockercompose

Das Backend Komponenten bestehend aus API, Reverse Proxy und Datenbank werden als Docker Services definiert und in einer Dockercompose Konfiguration zusammengestellt. Durch den Einsatz von Dockercompose werden alle Komponenten zentral an einer Stelle konfiguriert und zusammengestellt, was die Wartung vereinfacht.
Dies ermöglicht einen einfachen Betrieb auf beliebiger Hardware. Auch der nachträgliche Umzug zu einem anderen Hostingpartner ist durch die Dockerisierung wesentlich einfacher möglich.

JWT Tokens

Um den Benutzern des Contract Manager ein optimales Benutzererlebnis zu ermöglichen und dennoch nicht auf Security zu verzichten, werden JWT Tokens zur Etablierung von Stateless Sessions eingesetzt.
Beim erfolgreichen Login, erhält der Benutzer jeweils sowohl ein Access- wie auch ein Refreshtoken. Diese Tokens werden auf dem Client gespeichert und bei den Anfragen jeweils an den Server als Authorization Header mitgesendet. Das Accesstoken wird dabei für den Zugriff benötigt und muss bei jeden Request mitgesendet werden. Deshalb hat jenes nur eine kurze Gültigkeitsdauer. Sobald es abgelaufen ist, holt sich der Client mit dem Refreshtoken, welches länger gültig ist, ein neues Accesstoken beim Backend ab.

String Ressourcen

Im Frontend werden alle UI Texte als String Ressourcen erfasst. Dies ist als Vorbereitung implementiert, sodass zu einem späteren Zeitpunkt einfacher eine Internationalisierung eingerichtet werden kann.

3.3 Sprachen, Libraries, Frameworks

Für die Entwicklung des Contract Manager werden eine Vielzahl von Sprachen, Libraries sowie Frameworks eingesetzt.

3.3.1 Vorgaben durch Aufgabenstellung

Anhand der Aufgabenstellung sind grosse Teile der technischen Umgebung vorgegeben.

- Mobile App: React Native, Flutter oder Ionic
- API: NodeJS
- Datenbank: MongoDB
- Document Storage: Digitalocean oder S3

Dadurch ist die Wahl der möglichen Technologien in vielen Bereichen eingeschränkt. Weil dem Projektteam das Know-how für viele der bereits festgelegten Technologien fehlt, werden wenn immer möglich bekannte Frameworks und Libraries verwendet.

3.3.2 Flutter

Für die Mobile App haben wir die Wahl zwischen React Native[5], Flutter und Ionic[7]. Alle diese SDKs oder Frameworks ermöglichen die Entwicklung von Cross-Platform Mobile Apps.
Da alle Teammitglieder bisher weder mit der einen noch der anderen Technologie Erfahrungen sammeln konnten, viel die Entscheidung anfangs entsprechend schwer.

Mit Ionic ist der Einsatz von Wrappers notwendig, um ein natives UI Erlebnis zu ermöglichen. Dies hat einen direkten Einfluss auf die Performance. Dieser Fakt und die Tatsache, dass Ionic in den letzten Jahren an Beliebtheit verlor[16], wurde Ionic als mögliche Technologie ausgeschlossen.
Sowohl React Native als auch Flutter bieten eine sehr gute Performance sowie eine hohe Akzeptanz in einer breiten Community. Durch die regelmässigen Beratungsgespräche wurde in Erfahrung gebracht, dass der Industriepartner bereits in der Vergangenheit auf Probleme beim Entwickeln von React Native Applikationen stiess. Im Hinblick darauf, dass der Contract Manager eventuell nach Abschluss der Arbeit durch den Industriepartner weiterentwickelt wird und das Projektteam weder React Native noch Flutter bevorzugte, entschied man sich für Letzteres.

3.3.3 Handlebars

Für den Passwortreset-Prozess müssen vom Backend einige Views bereitgestellt werden. Um dies möglichst einfach zu halten, wird mit Handlebars eine bereits bekannte Template-Engine eingesetzt. Die Views werden entsprechend simpel mit HTML und CSS zusammengestellt.

3.3.4 Caddy

Um im Backend das Routing zu den Views sowie der API zu konfigurieren, wird Caddy als Reverse Proxy eingesetzt. Die Entscheidung wird aufgrund der einfachen Konfiguration getroffen und weil bereits einige Teammitglieder Caddy verwendet haben und entsprechendes Know-how mitgebracht haben.

3.3.5 Express.js

Da Express.js die Entwicklung einer API unter NodeJS stark vereinfacht und dem gesamten Team bereits bekannt ist, werden keine Alternativen dazu evaluiert.

3.3.6 TypeScript

Um im Backend Typensicherheit zu erhalten, wird TypeScript eingesetzt.

3.3.7 Mocha

Mocha[1] ist das Testframework schlechthin für NodeJS Applikationen. Alle Teammitglieder konnten bereits Erfahrungen damit sammeln.

3.3.8 Chai

Mithilfe der Chai Assertion Library[9] können Assertions aneinandergekettet werden. Mocha und Chai sind ein Dreamteam, welche fast immer zusammen eingesetzt werden.

3.4 Sequenzdiagramme

Um die wichtigsten Prozesse der Contract Manager App genauer zu betrachten, werden diese in einem Sequenzdiagramm detailliert beschrieben.

3.4.1 Registrierung

Dieser Prozess beschreibt die Registrierung eines neuen Useraccounts über den Registrierungsscreen der Mobile App.
Beim Erstellen eines neuen Useraccounts wird jeweils automatisch in der Datenbank eine erste Person zu diesem Useraccount erfasst.

Useraccounts für eine bereits verwendete E-Mail Adresse können nicht erstellt werden. Dies ist notwendig, um den Passwortreset Prozess sicher umzusetzen.

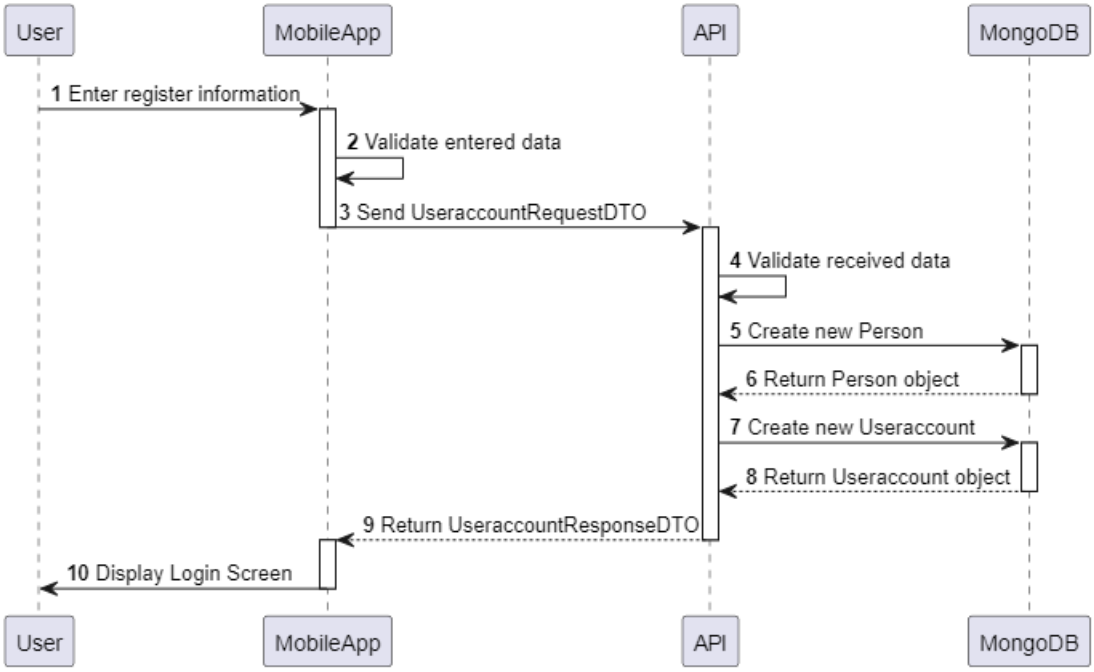

Abbildung 3.6: Sequenzdiagramm - Registrierung

3.4.2 Login

Dieser Prozess beschreibt die Anmeldung mit Username und Passwort über den Login-Screen der Mobile App.

Dieser Prozess muss nicht bei jedem Starten der Contract Manager App durchlaufen werden, sondern lediglich in folgenden Szenarien:

- Die Mobile App wird zum ersten Mal gestartet
- Es fand ein manuelles Abmelden in der Mobile App statt
- Im Secure Store des Mobile Device ist kein Refreshtoken vorhanden
- Im Secure Store des Mobile Device ist ein abgelaufenes Refreshtoken vorhanden

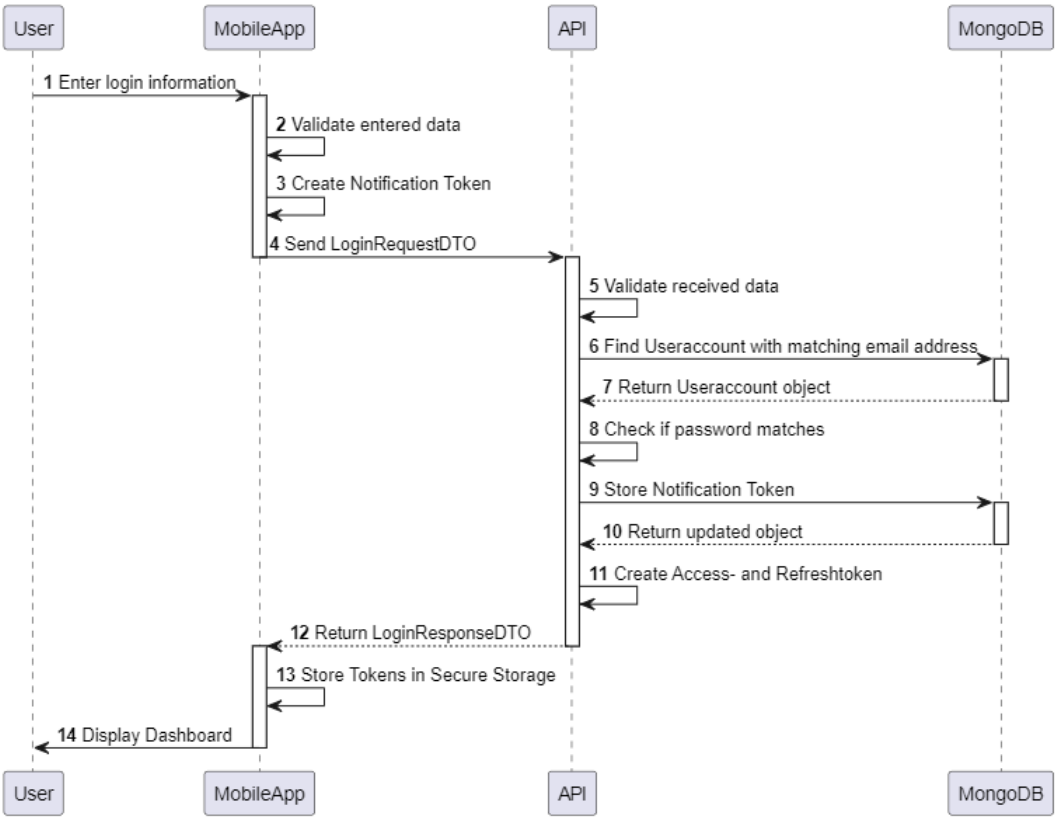

Abbildung 3.7: Sequenzdiagramm - Login

3.4.3 Authenticate

Dieser Prozess beschreibt die Authentifizierung über den Authenticate-Screen der Mobile App.
Der Authenticate-Screen wird immer beim Starten der App aufgerufen. Dort wird jeweils geprüft, ob ein gültiges Accesstoken auf dem Mobile Device vorhanden ist. Falls dies der Fall ist, wird der Benutzer direkt angemeldet und an das Dashboard weitergeleitet. Dieser Fall wird nachfolgend im Sequenzdiagramm beschrieben.

Ist das Accesstoken abgelaufen, wird anhand des Refreshtokens ein neues Accesstoken generiert und dem Mobile Device zur Speicherung zurückgesandt. Zudem erfolgt eine Weiterleitung auf das Dashboard.
Falls das Refreshtoken abgelaufen, ungültig oder nicht vorhanden ist, wird der Benutzer auf den Login-Screen weitergeleitet.

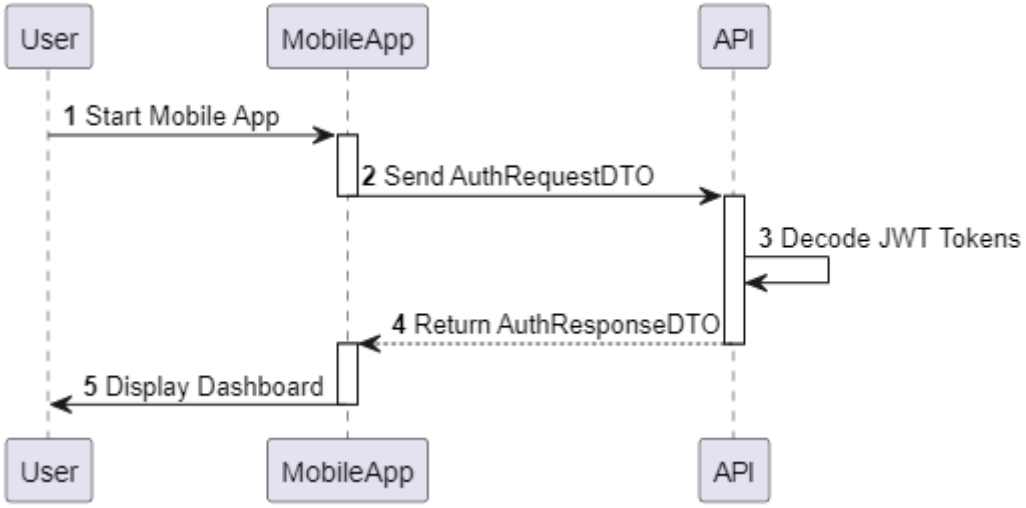

Abbildung 3.8: Sequenzdiagramm - Authenticate

3.4.4 Passwortreset

Dieser Prozess beschreibt den Passwortreset in der Contract Manager App. Für den Passwortreset wird sowohl die Mobile App als auch der Browser verwendet.

Nachdem der Benutzer die E-Mail-Adresse seines Useraccounts eingetragen hat, wird ein Token generiert. Dieses Token wir mitsamt der vordefinierten Gültigkeitsdauer für den Useraccount gespeichert. Danach wird eine E-Mail an die eingegebene Adresse versandt, welche eine URL enthält.
Wird dieser Link im Browser aufgerufen, landet der Benutzer auf einem Formular, auf welchem sowohl das Token, die E-Mail-Adresse sowie das neue Passwort eingegeben werden muss. Sind diese Eingaben korrekt, wird ein Hash des neuen Passworts in der Datenbank gespeichert. Zudem wird das Token gelöscht.

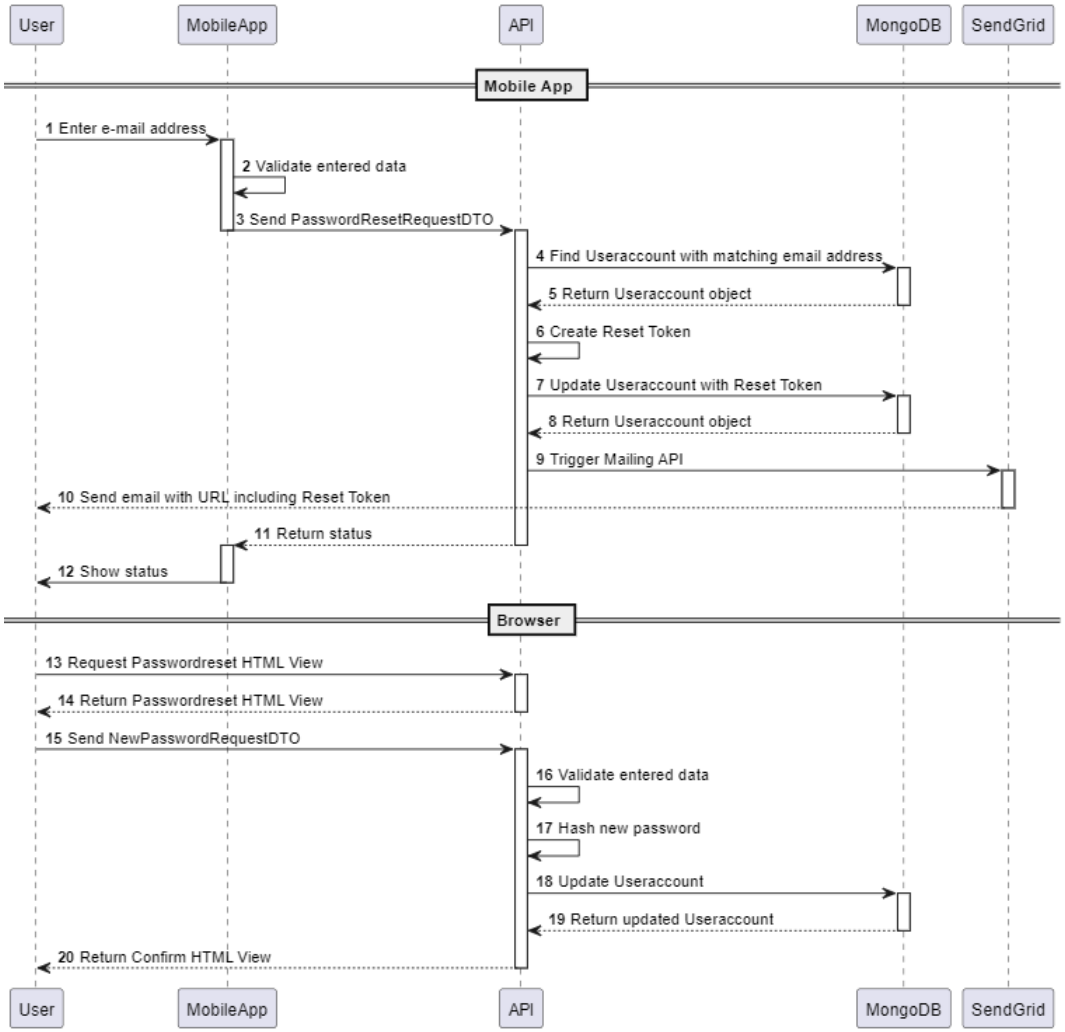

Abbildung 3.9: Sequenzdiagramm - Passwortreset

3.4.5 Dashboard

Dieser Prozess beschreibt, welche Daten in der Contract Manager App beim Anzeigen des Dashboards geladen werden.

Beim Zugriff aufs Dashboard werden zwei API Calls abgesetzt. Einerseits werden Daten zur Befüllung der Tabelle angefordert.

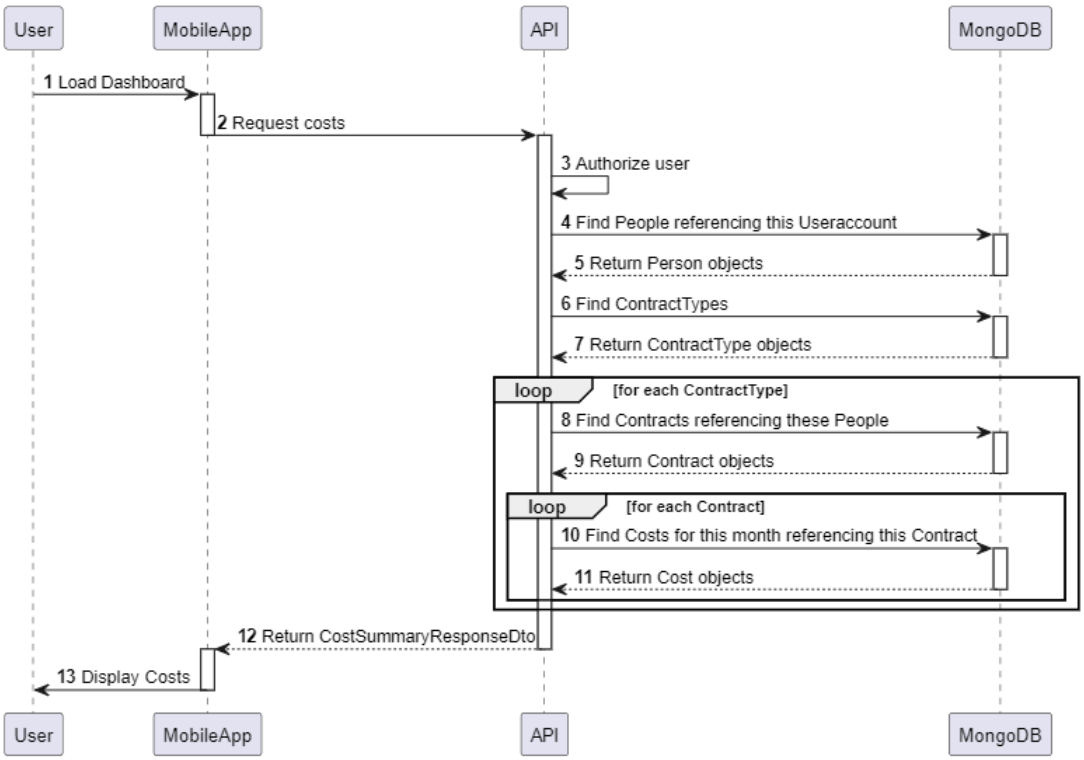

Abbildung 3.10: Sequenzdiagramm - Dashboard Kostentabelle

Zum anderen werden Daten in einem anderen Format als Datengrundlage für das Kostendiagramm benötigt. Diese Abfragen werden parallel ausgeführt. Sobald die Mobile App eine Antwort erhält, werden die Daten entsprechend auf dem Screen dargestellt.

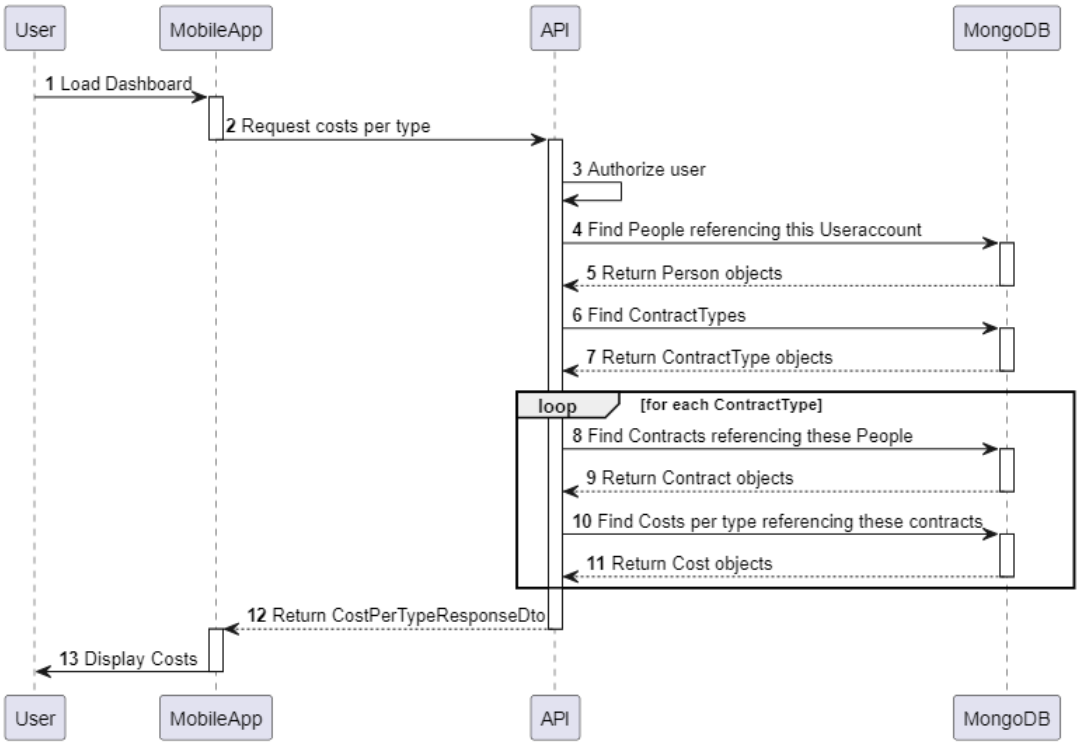

Abbildung 3.11: Sequenzdiagramm - Dashboard Kostendiagramm

3.4.6 Vertrag hinzufügen

Dieser Prozess beschreibt das Hinzufügen eines neuen Vertrags in der Contract Manager App.

Immer wenn ein neuer Vertrag hinzugefügt wird, werden sogleich die Kosten bzw. Zahlungstermine berechnet. Dies wird anhand des Vertragbeginns und der Zahlungsperiode für die nächsten 20 Jahre (oder bis zum Ablaufdatum) berechnet.

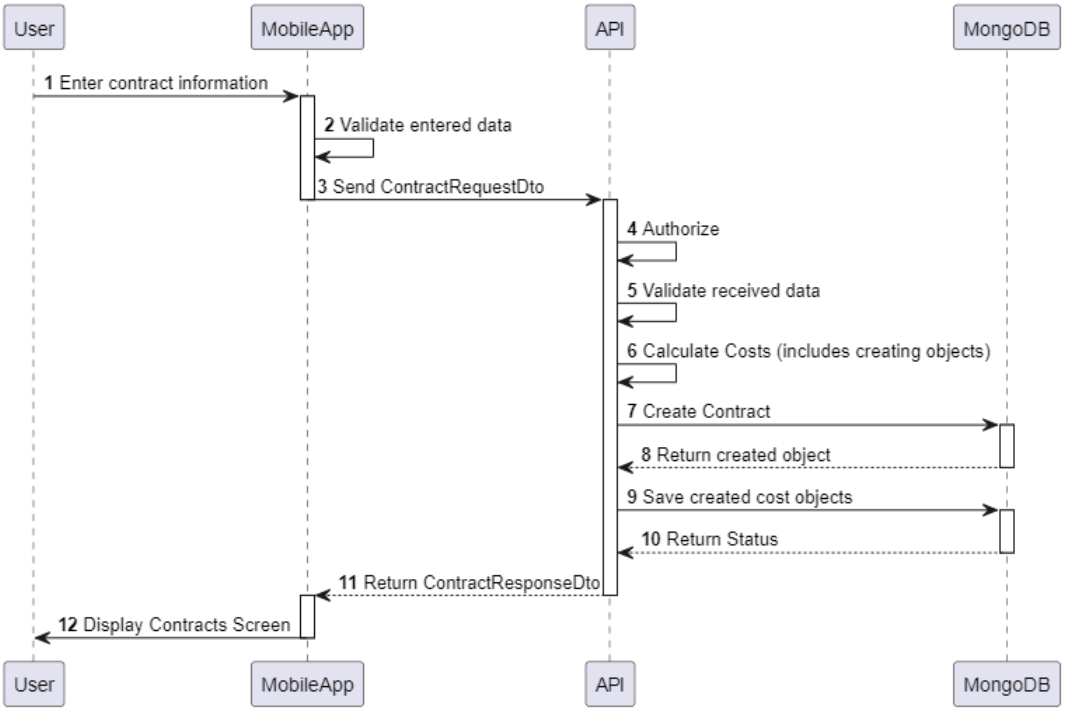

Abbildung 3.12: Sequenzdiagramm - Vertrag hinzufügen

3.5 Schnittstellen

Die Contract Manager Applikation kommuniziert mit zwei externen Systemen.

3.5.1 SendGrid

Für das Zustellen der Passwortreset Tokens im Passwortreset Prozess ist die Nutzung einer Mailing-API unabdingbar. Es wurde sowohl SendGrid, als auch Mailjet als mögliche Option evaluiert. Da SendGrid nebst klassischem Mailing auch den Einsatz von Templates ermöglicht, fiel die Wahl auf SendGrid.

3.5.2 Google Firebase

Eine Anforderung war es, Analysedaten zur Nutzung der Applikation zu sammeln. Zudem sollten Benutzer Push-Benachrichtigungen auf dem Mobile App erhalten können. Google Firebase konnte mit Firebase Cloud Messaging und Google Analytics beide Anforderungen erfüllen. Da Flutter ebenfalls aus dem Hause Google kommt, war auch die Kompatibilität gegeben.

3.6 Deployment

Alle Codedateien, Dokumentationen sowie weitere projektbezogene Dateien werden in einem Gitlab Repository verwaltet. Dabei werden sowohl alle projekt- als auch alle produktbezogenen Dateien in jeweils einem separaten Gitlab Projekt geführt.

3.6.1 Repository

Alle Komponenten von Front- und Backend sind in einem gemeinsamen Repository abgelegt.

Branches

Das Projekt verfügt über zwei beständige Branches, dem Master- und dem Develop-Branch. Entwickelt wird in weiteren Branches, welche jeweils auf einen Task oder eine Userstory aus YouTrack referenzieren. Aus diesen Branches wird nach abgeschlossener Arbeit ein Merge Request zusammengestellt und von einem anderen Teammitglied kontrolliert. Ist dieser Review erfolgreich, wird die Arbeit in den Develop-Branch gemerged. Falls Fehler, Probleme oder Unstimmigkeiten im Code festgestellt werden oder Fragen auftreten, muss der Ersteller des Merge Requests Korrekturen vornehmen.

Der Develop-Branch enthält somit stets eine lauffähige Version des Produkts. Neue Features sind dennoch schnell verfügbar und können bereits getestet und verwendet werden.

Da nach Abschluss dieses Projekts keine finalisierte Applikation vorliegt, welche gleich im produktiven Betrieb verwendet werden kann, wird bewusst auf einen Merge auf den Master-Branch verzichtet. Wenn zu einem späteren Zeitpunkt eine produktive Version vorliegt, kann diese dann auf den Master-Branch gemerged werden.

Infrastruktur

Die Contract Manager App wird in den folgenden Systemen entwickelt, getestet und betrieben.

System	Beschreibung
Localhost	Die Applikation wird lokal entwickelt und direkt aus der IDE als auch über Docker getestet.
Gitlab	Die Gitlab Instanz der OST wird für den gesamten DevOps Prozess von der Versionsverwaltung bis hin zum Deployment des fertigen Produkts verwendet. Das Projektmanagement mitsamt Dokumentation wird ebenfalls in Gitlab verwaltet.
DigitalOcean	Auf einem Droplet des Hostingproviders DigitalOcean wird das Backend in mehreren Docker-Containern betrieben.

Tabelle 3.1: Infrastruktur Systeme

Continuous Integration (CI)

Jeder Commit, der in das Gitlab Repository gepusht wird, stösst die CI Pipeline an. Dabei werden folgende Schritte für das Backend Verzeichnis durchlaufen:

- Dependencies installieren
- Projekt builden
- Tests starten

Nur wenn all diese Schritte erfolgreich sind, erhält die Pipeline den Status "Passed". Wenn die Pipeline den Status "Failed" erreicht, werden die nachfolgenden Schritte, wie bspw. das Deployment nicht mehr durchgeführt.

Continuous Deployment (CD)

Für den Develop-Branch wurde zusätzlich ein Continuous Deployment eingerichtet. Dies wird als zusätzliche Stage in der bereits vorhandenen Pipeline realisiert.
Der Develop-Branch ist vor direkten Änderungen geschützt. Entsprechend werden Änderungen nur via Merge Request eingeführt. Dabei wird dann jeweils die Deploy-Stage zusätzlich zum CI Teil der Pipeline getriggert. In diesem Teil der Pipeline wird eine SSH Verbindung zum DigitalOcean Droplet hergestellt. Vor dort aus wird die aktuelle Version des Repositories abgeholt. Danach können die Docker Contrainer erstellt und hochgefahren werden.

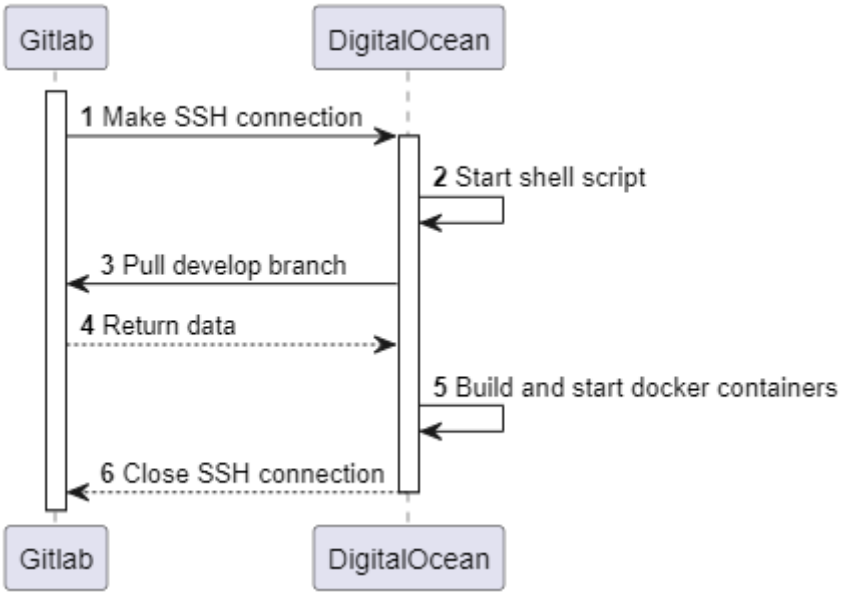

Abbildung 3.13: Sequenzdiagramm - Continuous Deployment

3.7 Wireframes

Die nachfolgenden Wireframes zeigen einen UI Entwurf, welcher die Anforderungen an das Produkt umsetzt.

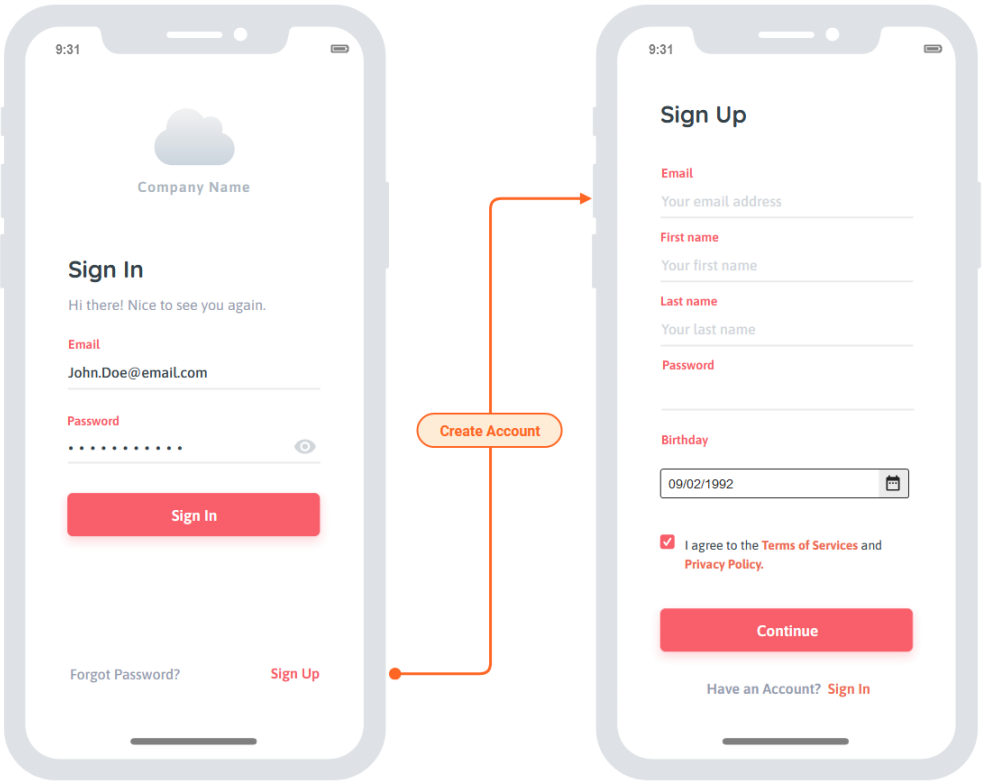

Abbildung 3.14: Wireframe - Login und Registrierung

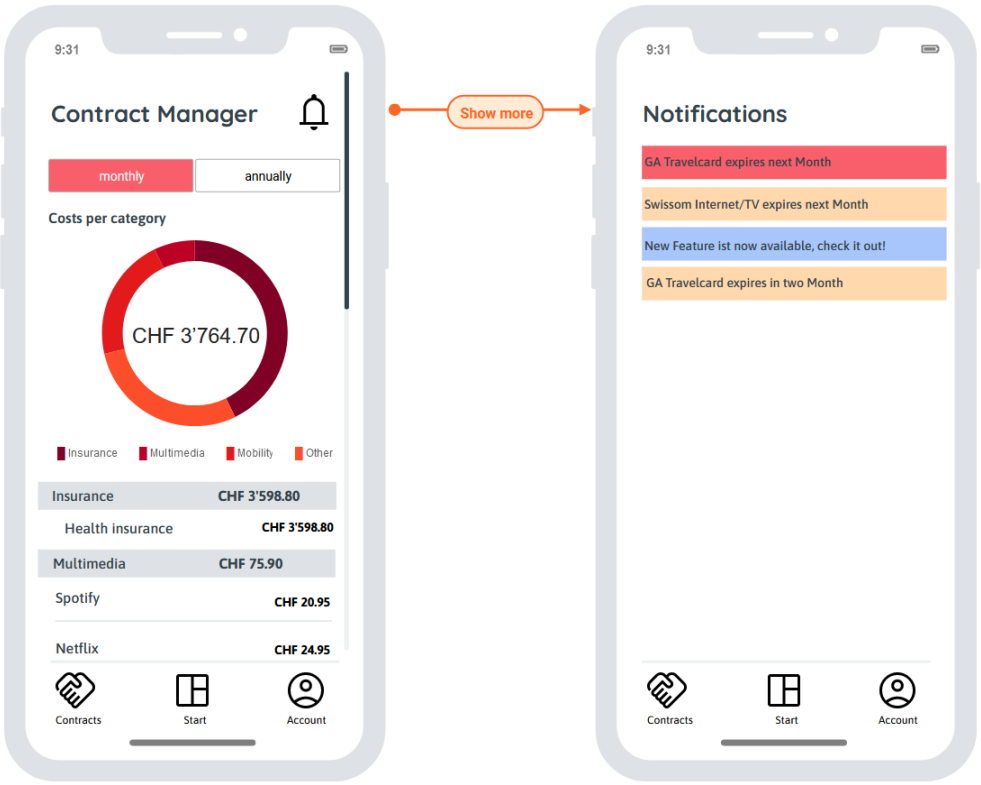

Abbildung 3.15: Wireframe - Dashboard

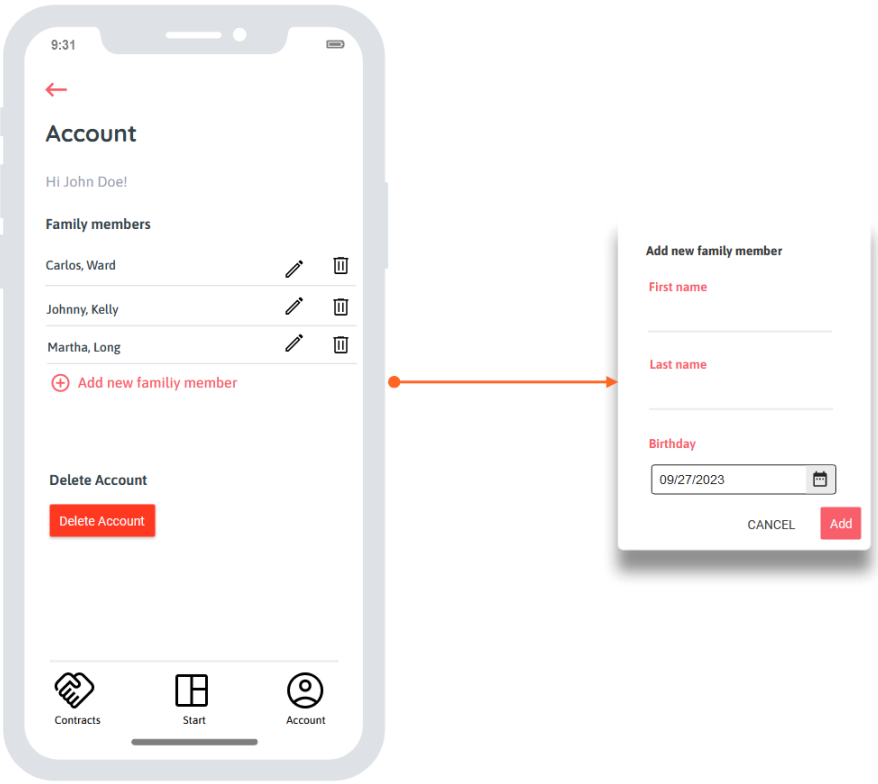

Abbildung 3.16: Wireframe - Mein Konto

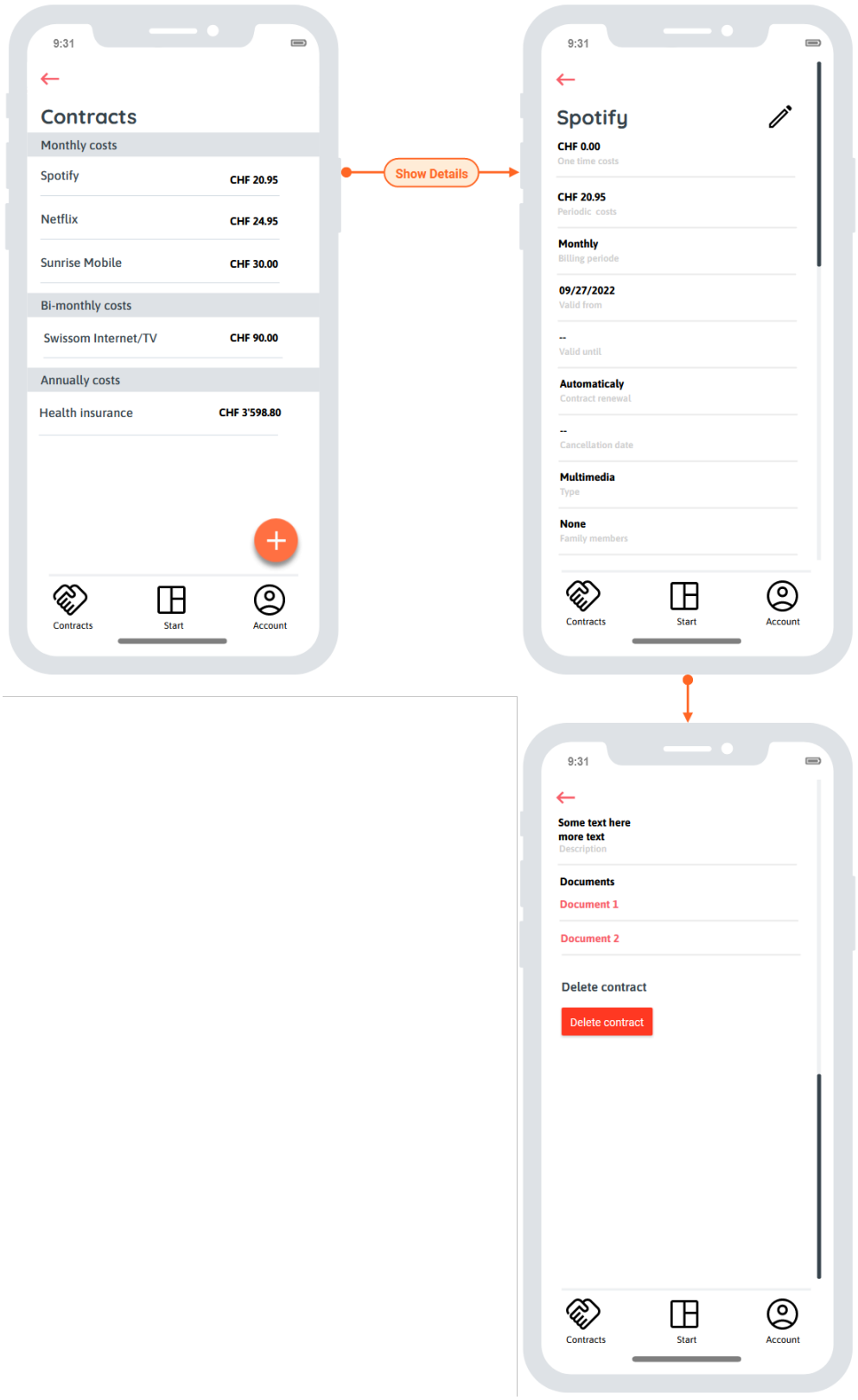

Abbildung 3.17: Wireframe - Verträge

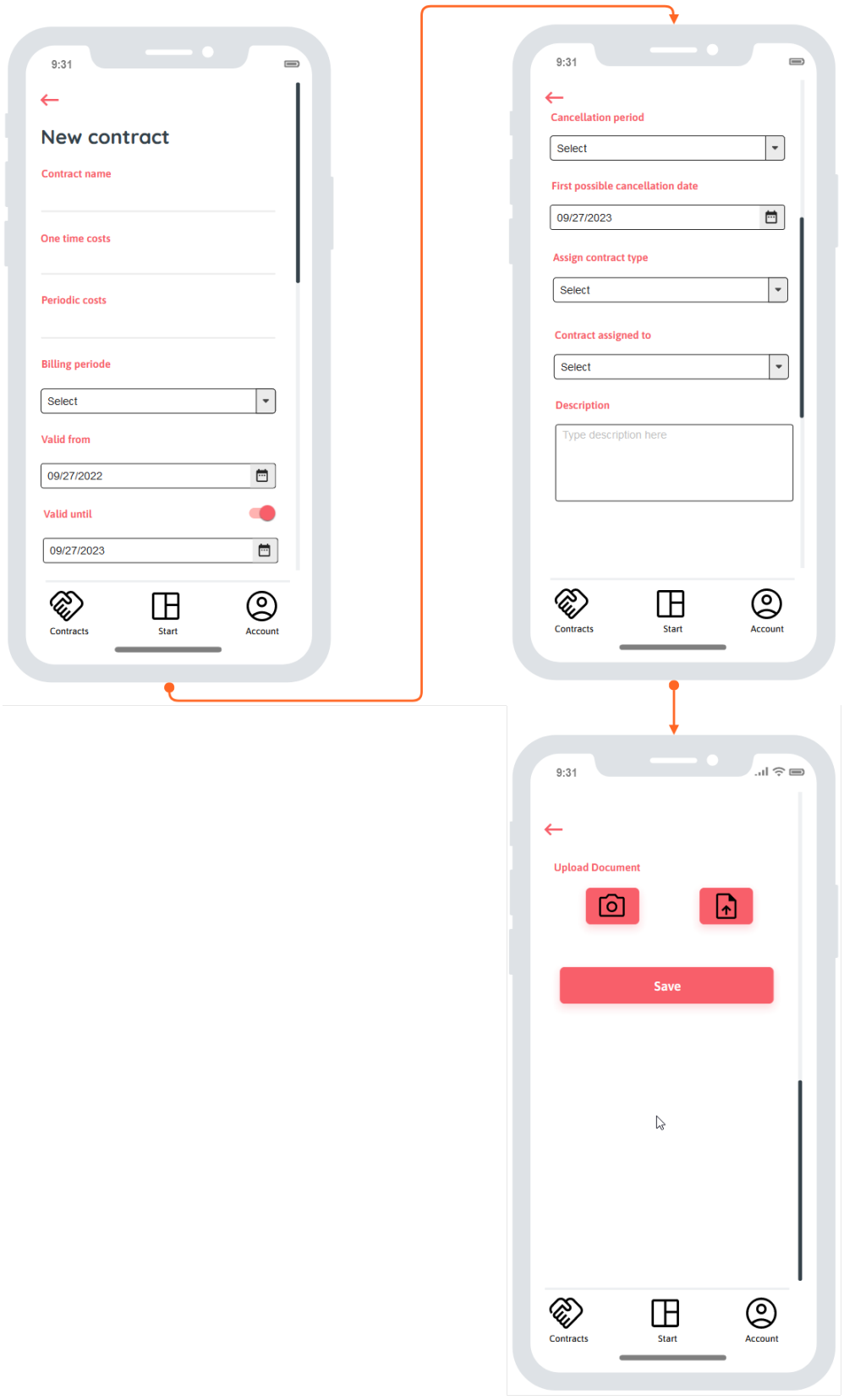

Abbildung 3.18: Wireframe - Vertrag hinzufügen

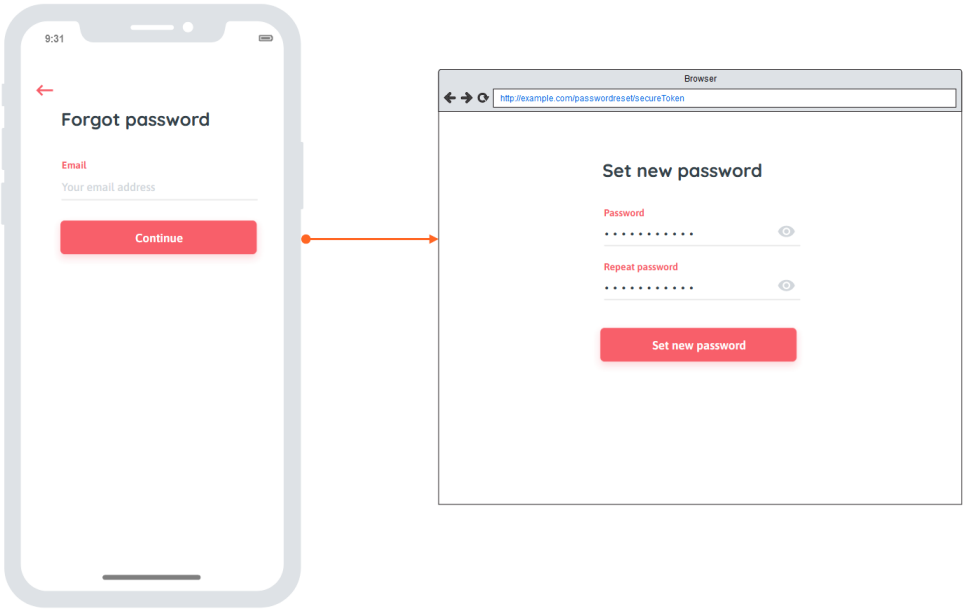

Abbildung 3.19: Wireframe - Passwortreset

4. Implementierung

In folgendem Kapitel wird die Implementierung der App beschrieben. Dabei wird auf die wichtigsten Details und die dahinterliegenden Entscheidungen eingegangen.

4.1 Projektstruktur

Das Projekt liegt auf der Gitlab Instanz der OST und ist folgendermassen unterteilt:

Verzeichnis	Beschreibung
api-tests	Im Verzeichnis api-tests ist die Postman Kollektion der API-Tests sowie ein kurzes README für die genauere Erklärung.
client	Beinhaltet den Quellcode der Flutter App (Frontend).
deployment	Im Deployment ist das Shell-Skript enthalten, das für die Bereitstellung des Backends auf DigitalOcean benötigt wird.
rest-api-server	Der Quellcode des NodeJS Servers (Backend).

Tabelle 4.1: Projekt-Datenstruktur

4.1.1 Code

Da alle Teammitglieder sowohl das Frontend als auch das Backend entwickeln, gibt es nur ein Repository für beides. Im Backend-Verzeichnis sind zudem die Docker Konfigurationsdateien abgelegt.

4.1.2 Projekt Management

Die Dokumentation ist in einem separaten Gitlab Repository gespeichert. Als Vorlage dient ein leicht angepasstes Software Engineering Projekt (SEProj) LaTex-Dokument. Dadurch ist die Versionierung der Dokumentation sichergestellt. Die Zeiterfassung sowie das Sprint-Planning wird über YouTrack erledigt. Da ein Teil des Teams bereits im SEProj mit YouTrack gearbeitet hat und sich dies dort bewährt hat, wurde auch in diesem Projekt entschieden, auf YouTrack zu setzen.

4.2 Qualitätsnachweis

In diesem Abschnitt ist die Umsetzung der Qualitätsmassnahmen dargestellt.

4.2.1 API-Testing (NFR-14)

Während der Entwicklung wird die jeweilige Backend-Route per Postman getestet, um sicherzustellen, dass das Backend den Anforderungen entsprechend funktioniert. Als API-Test Tool wurde ebenfalls Postman eingesetzt. In der nachfolgenden Abbildung ist exemplarisch ein API-Test bzw. dessen Durchführungsresultat zu sehen.

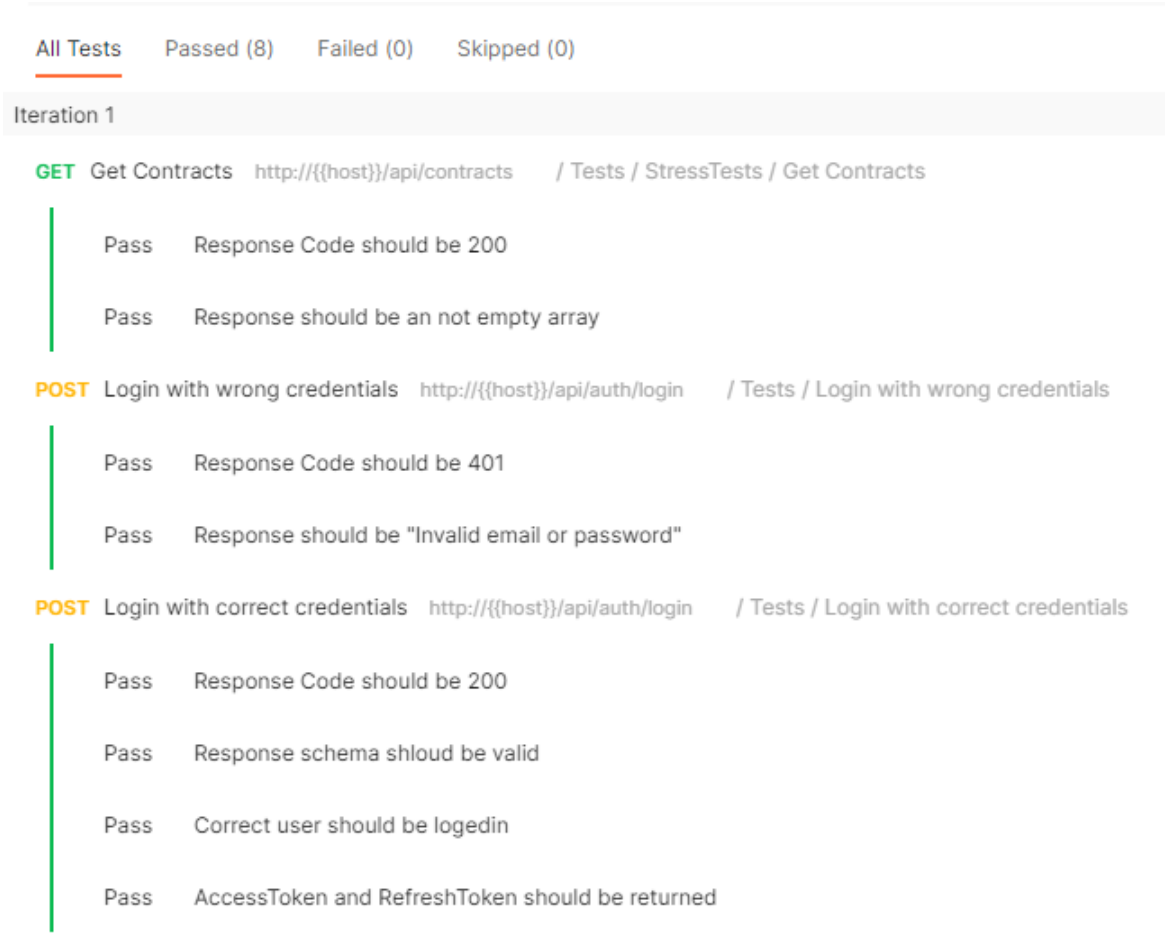

Abbildung 4.1: Postman API-Tests

4.2.2 Unit-Testing

Unit-Tests sind exemplarisch in der Gitlab Pipeline integriert, um so für eine Weiterentwicklung der App gerüstet zu sein. Für die Tests wird Mocha und Chai eingesetzt. Die Unit-Tests werden in der CI-Pipeline bei jedem Commit auf das Repository ausgeführt.

4.2.3 Stress Test (NFR-2)

Um sicherzustellen, dass das Backend genügend Anfragen pro Minute verarbeiten kann (NFR-3), wird die Rest-API per Postman einem Stresstest unterzogen. Der Test besteht darin, alle Verträge eines eingeloggten Benutzers abzufragen. Dabei wird erwartet, dass die Antwort des Servers ein HTTP Statuscode 200 (Test 1) und die Liste aller Verträge des Benutzers (Test 2) zurückgibt. Daraus resultieren 500 Requests an das Backend und 1000 erfolgreiche Tests.
Im Ganzen benötigte das Backend nicht mal eine halbe Minute, um alle Requests zu verarbeiten. Somit ist eine Verarbeitung von 500 Requests in einer Minute ohne Probleme machbar. Zusätzlich sind bei diesem Test noch etliche andere Faktoren im Spiel, die das Ergebnis negativ beeinflussen. Zum Beispiel die Tatsache, dass alle Anfragen vom gleichen Ort stammen und somit alle Anfragen nacheinander abgesetzt werden und nicht gleichzeitig von verschiedenen Orten. Aus diesem Grund haben wir auch die Ressourcen des DigitalOcean Droplets überwacht. Diese zeigen auf, dass der grösste Flaschenhals bei diesem Test nicht die Applikation selber ist, sondern der Umstand, dass alle Anfragen vom selben Client kommen. Die CPU Auslastung steigt während des Tests nicht mal auf 6%.

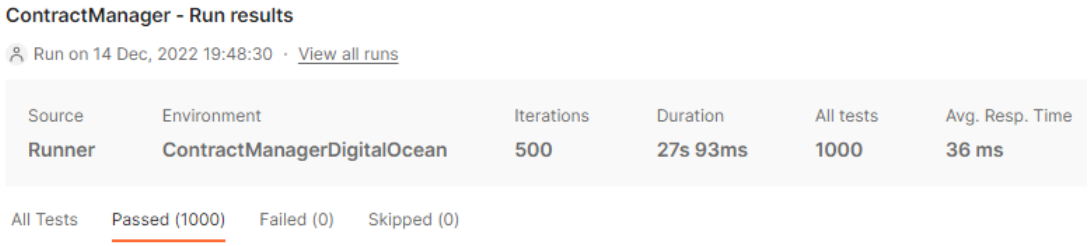

Abbildung 4.2: Postman Stress-Test

4.2.4 Usability Test

Die Usability Tests sollen aufzeigen, ob die App benutzerfreundlich und funktionsfähig ist. Die Resultate sind gesamthaft positiv ausgefallen.
Vereinzelte Mängel und Verbesserungsvorschläge sind trotzdem aufgezeigt worden. Einige dieser Mängel konnten direkt behoben werden.

Passwortreset-Token abgelaufen

Bei einem Usability Test ist aufgefallen, dass unter Umständen der Passwort-Reset-Token nicht richtig übermittelt wurde. Wenn dieser Token ein + Zeichen beinhaltet, konnte er nicht richtig ins HTML Formular übertragen werden und wurde stattdessen in einen Leerschlag umgewandelt. Somit stimmt das Token nicht mit dem gespeicherten Wert in der Datenbank überein und der Passwortreset schlägt fehl.
Neu wird der Token, bevor er per E-Mail versendet wird, so codiert, dass dieser vom Browser korrekt interpretiert werden kann. Dies wird anhand der *encodeURIComponent()* Funktion bewerkstelligt.

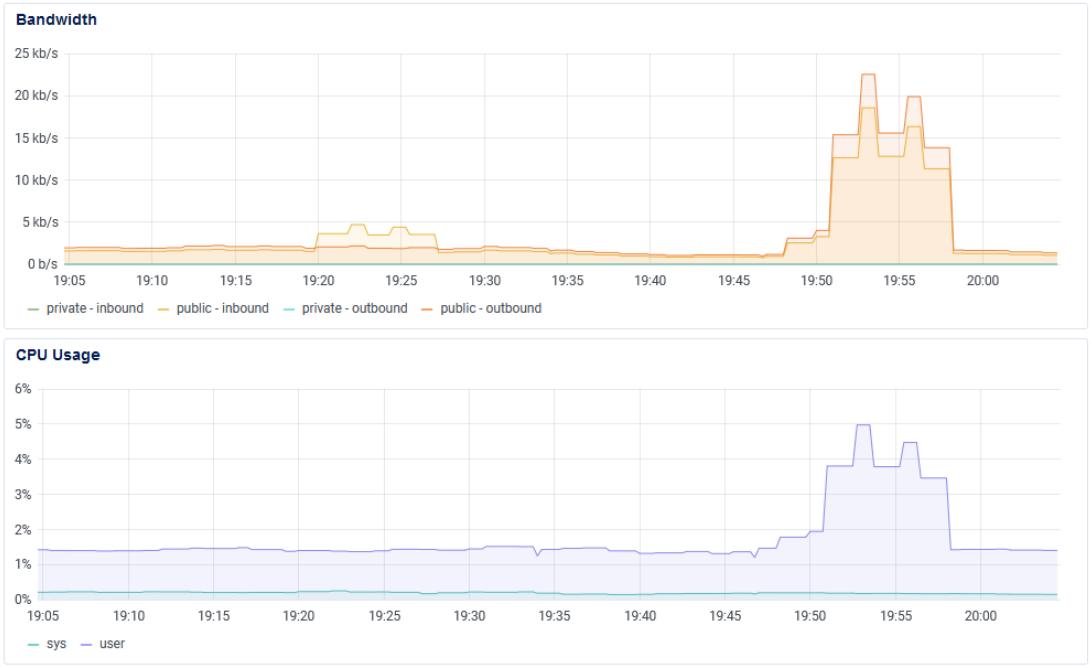

Abbildung 4.3: DigitalOcean Ressourcen Verbrauch während des Stress-Tests

Zurück Schaltfläche loggt aus

Nach dem Registrieren und anschliessendem Login, konnte man per "Zurück Schaltfläche" auf den Registrieren-Screen gelangen. Danach war es möglich noch, einen Schritt zurück zu navigieren, worauf man ausgeloggt wurde.
Dieses Fehlverhalten ist nun behoben, indem in Flutter eine andere Art der Weiterleitung bei der Navigation verwendet wird. Konkret wird beim Registrieren, Ausloggen und Konto löschen mit *pushNamedAndRemoveUntil()* zurück navigiert. Dies sorgt dafür, dass der Verlauf der besuchten Screens gelöscht wird.

4.3 Ergebnisse

Die funktionalen Anforderungen konnten nahezu alle umgesetzt werden. Benutzer der App können sich registrieren und einloggen, Verträge erfassen und erhalten eine Übersicht über die anfallenden Kosten des laufenden Monates. Dies ist auf dem Dashboard tabellarisch wie auch grafisch mittels Kuchendiagramm ersichtlich. Die Kosten sind nach Kategorie gruppiert und kategorisch aufsummiert. Jeder Benutzer der App kann zusätzliche Familienmitglieder erfassen und beim Erstellen eines Vertrags jener dem gewünschten Familienmitglied zuweisen. 14 Tage vor Ablauf eines Vertrags oder vor Ablauf eines Kündigungstermins wird der Benutzer entsprechend darüber informiert.

4.3.1 Infrastruktur

Auf DigitalOcean wird ein Ubuntu Droplet für das Backend betrieben. Auf diesem laufen die drei Docker Container, die im Container Diagramm ersichtlich sind. 3.3

Reverse Proxy

Der Caddy Reverse Proxy verarbeitet alle Anfragen, die von der App aus an das Backend gestellt werden und leitet sie an das API weiter.

Port	Zweck
80 (HTTP)	Darüber kommuniziert die App mit dem Backend (über den Reverse Proxy).
22 (SSH)	Wird für die Bereitstellung des Develop-Branches genutzt.

Tabelle 4.2: Backend - Reverse Proxy

4.3.2 Authentisierung

Die Authentisierung ist mit der Hilfe von Json-Web-Tokens (JWT) umgesetzt. Wenn sich ein User einloggt, sendet dieser E-Mail-Adresse und Passwort an das Backend. Dort wird das Passwort gehashed und mit dem Wert in der Datenbank verglichen. Zudem wird geprüft, ob dieses zur E-Mail-Adresse passt. Wenn dies der Fall ist, wird ein Access-Token mit einer Gültigkeitsdauer von 15 Minuten erstellt. Das Refresh-Token, welches ebenfalls generiert wird, hat eine Gültigkeit von einem Jahr. Die Tokens bestehen aus Folgendem:

- Das Verfahren, um das Token zu erstellen (in unserem Fall RS265-Algorithmus und Type JWT).
- E-Mail Adresse des Benutzers
- Benutzer Identifikation (id) aus der Datenbank
- Zeitpunkt der Erstellung
- Ablaufdatum

Danach wird E-Mail-Adresse, Benutzer-Id, Access- und Refresh-Token am Frontend zurückgesendet. Der Benutzer ist nun authentisiert und speichert im Secure Storage die beiden Tokens. Bei der nächsten Anfrage durch den Benutzer (z. B. Abruf der Verträge) werden die beiden Tokens mitgeliefert und können vom Backend ausgewertet werden.

- Wenn der Access-Token gültig ist, ist der Benutzer authentisiert und kann auf seine Daten zugreifen.
- Ist der Access-Token abgelaufen, wird geprüft, ob der Refresh-Token gültig ist. Ist dies der Fall, wird ein neuer Access-Token erstellt und der Benutzer kann seine Daten abrufen.
- Sind beide Tokens abgelaufen/ungültig, wird der Benutzer ausgeloggt und muss sich erneut einloggen.

4.3.3 Zusätzlich implementierte Features

Folgende zusätzliche Features oder Erweiterungen der Features sind in der App implementiert.

Notification-Screen

Zusätzlich zu den Anforderungen ist ein Notification-Screen implementiert. Dieser zeigt alle Benachrichtigungen an, die an diesem Account versendet wurden. Je nach Kategorie der Benachrichtigungen wird ein anderes Symbol dargestellt.

- Abgelaufener Vertrag
- Kündigungstermin
- Warnung

Die Kategorie Warnung ist lediglich eine Vorbereitung für eine allfällige Erweiterung und wird momentan nicht verwendet.

Text im Frontend ausgelagert

Als Vorbereitung für eine mehrsprachige App sind alle Texte, die im Frontend angezeigt werden, in einer separaten Strings-Klasse definiert.

Automatisch einloggen

Wenn man in der App einmal eingeloggt ist, wird man mit dem Access- bzw. Refresh-Token automatisch wieder eingeloggt. Dies war keine Anforderung, aber ist heute "State of the Art". Ansonsten wäre die App auch sehr mühselig zu bedienen, daher war es uns wichtig, dass wir dies umsetzen konnten.

4.3.4 Benachrichtigungen

Die Benachrichtigungen werden einmalig beim Starten des Backend-Servers und anschliessend einmal pro Tag ermittelt und versendet. Dies wird über ein CronJob im Backend abgearbeitet.

Ablaufdatum

Um über ablaufende Verträge informiert zu werden, werden alle Verträge in der Datenbank nach dem Ablaufdatum abgefragt. Wenn dieses Datum 14 (oder weniger) Tage in der Zukunft liegt, wird der Benutzer informiert.
Dies passiert, indem die Benachrichtigung in der Datenbank gespeichert und der Benutzer anhand des NotificationTokens über Firebase benachrichtigt wird. Zudem wird das Benachrichtigungsdatum (sendedValidUntilNotification) dem Vertrag hinzugefügt, um sicherzustellen, dass die Benachrichtigung nicht erneut ausgelöst wird.
Dies ist so implementiert, um gegebenenfalls einem Server-Neustart und somit mehrfachen Versenden der Benachrichtigung vorzubeugen. Ausserdem kann so auch sichergestellt werden, dass verpasste Benachrichtigungen infolge Serverausfall oder Wartung nachgeholt werden.

Kündigungstermin

Im Gegensatz zum Ablaufdatum ist es beim Kündigungstermin etwas komplizierter. Verträge, die kein Ablaufdatum haben, können je nach Kündigungsperiode monatlich oder jährlich gekündet werden. Somit muss der Benutzer immer wieder informiert werden. Das Benachrichtigungsdatum wird anhand des erstmöglichen Kündigungsdatums errechnet. Für jeden neuen Vertrag wird beim nächsten Durchlauf des CronJobs das nächste zu benachrichtigende Datum errechnet und in der Datenbank gespeichert.
Danach muss der CronJob jeweils nur noch dieses Datum prüfen. Wenn dieses Datum in der Zukunft liegt muss nichts erledigt werden. Wenn das Datum in der Vergangenheit liegt, heisst dies, dass eine Benachrichtigung bereits überfällig ist. Entsprechend wird dies nachgeholt und das Datum mit Hilfe der Kündigungsperiode aktualisiert.

4.3.5 Backend

In diesem Abschnitt wird näher auf den Aufbau der API eingegangen.

Controller

Die Controller sind die Schnittstelle im Backend zwischen der Route (die vom Frontend angesprochen werden) und der weiteren Verarbeitung.

Controller	Inhalt
auth	Stellt alle Schnittstellen, die mit Authentisierung zu tun haben, bereit. Vom Registrieren, Einloggen, Passwort zurücksetzen bis zum Konto löschen.
contract	Darüber lassen sich Verträge erstellen. Alle Verträge des jeweils eingeloggten Benutzers abfragen. Sowie alle Verträge und Vertragskategorien zurückgeben.
const	Gibt die Kosten der Verträge sowie die monatlichen Kosten per Kategorie des eingeloggten Benutzers zurück.
notification	Damit werden alle Benachrichtigungen des Benutzers erstellt und abgerufen.
person	Dieser Controller ruft alle Personen des jeweiligen Kontos ab, erstellt und löscht sie.

Tabelle 4.3: Backend - Controller

Middleware

Middlewares werden vor Aufruf eines Controllers ausgeführt. So wir z. B. durch die Middleware geprüft, ob ein Benutzer eingeloggt ist und die gewünschte Route aufrufen darf.

Middleware	Inhalt
auth	Stellt sicher, dass nur Daten vom jeweilig eingeloggten Benutzer zurückgegeben/hinzugefügt werden können.
validateResource	Ist für die Validierung der Daten, welche vom Frontend kommen, zuständig

Tabelle 4.4: Backend - Middleware

Utils

Die Utils sind eine Sammlung von verschiedenen Funktionen und Hilfsfunktionen für die Ausführung diverser Aufgaben im Backend.

Utils	Inhalt
connect	Baut die Verbindung zur Datenbank auf.
costCalculation	Berechnet beim Hinzufügen eines Vertrages die wiederkehrenden Kosten, bis der Vertrag abläuft oder für 20 Jahre im Voraus.
firebase	Initialisiert firebase-admin was für das Versenden von Push-Benachrichtigungen benötigt wird.
jwt	- Erstellt Access- und Refresh-Token beim Login. - Generiert neue Access-Token, wenn diese abgelaufen sind. - Validiert Access- und Refresh-Token.
logger	Loggt Fehler und Informationen, entweder in der Konsole (für die Entwicklung) oder in einer Datei (für die Produktion).
mailer	Sendet die E-Mail für das Zurücksetzen des Passworts.
notification.cron	Registriert den CronJob.
notification	- Berechnet das nächste Versanddatum der Benachrichtigung pro Vertrag. - Sendet dem Benutzer Push-Benachrichtigungen.

Tabelle 4.5: Backend - Utils

4.3.6 Konfiguration

Um das Back-, swowie Frontend auf dem jeweiligen System korrekt laufen zu lassen, muss Client und Server entsprechend Konfiguriert werden. Dazu wird Folgendes eingesetzt. Im Frontend wird die Backend Url über die constant.dart Datei definiert und kann beim Starten der App überschrieben werden (mehr dazu im README des Repos). Im Backend wird eine .env Datei für das Abspeichern der Konfiguration benutzt.

- In der Datei /rest-api-server/config/default.ts werden die Daten aus der .env-Date eingelesen. Falls diese leer sind, werden Standardwerte gesetzt. Folgendes ist dort definiert:
  - Die Url zur Datenbank mit Login-Daten.
  - Gültigkeitsdauer der Access- und Refreshtoken.
  - Pfad zum Firebase Private Key.
  - Wie viele Tage vor Ablauf des Vertrages /des Kündigungstermins man informiert werden will.
  - SendGrid Token.
  - Logging Einstellungen (Konsole oder Datei, wie lange die Daten behalten werden, etc.)

- Für den Betrieb mit Docker wird dieselbe Datei (/rest-api-server/config/default.ts) verwendet.
  - Name der MongoDB Datenbank.
  - Login Daten und Port der Datenbank.
  - Port für den Reverse-Proxy
  - Logging Pfad.
  - Pfad zum Firebase Private Key.
  - In welchem Intervall der CronJob für die Benachrichtigungen laufen soll.

4.3.7 Frontend

Um die mobile App möglichst einfach zu halten, ist die App in drei Haupt-Screens unterteilt.

Verträge

Unter Verträge können erfasste Verträge eingesehen und neue hinzugefügt werden. Die Verträge werden nach Familienmitgliedern gruppiert. Ein Vertrag beinhaltet folgende Informationen:

- Den Namen des Vertrages
- Eine kurze Beschreibung (optional)
- Von wann an der Vertrag gültig ist
- Bis wann der Vertrag gültig ist (falls leer gelassen = unbegrenzte Laufzeit)
- Wann der Vertrag zum ersten mal gekündigt werden kann
- Die wiederkehrenden, sowie einmaligen Kosten (z.B. Setupgebühr bei einem Internet/TV Abo)
- Das gewünschte Familienmitglied dem dieser Vertrag gehört
- Vertragstyp bzw. um welche Kategorie es sich bei diesem Vertrag handelt
- Die Zahlungsperiode und Kündigungsperiode. (z.B. Wenn ein Vertrag monatlich bezahlt und gekündet werden kann, werden beide Perioden mit 1 und Monat definiert)

Screenshots

Das Ergebnis des Frontends ist am besten anhand von Screenshots darzustellen.

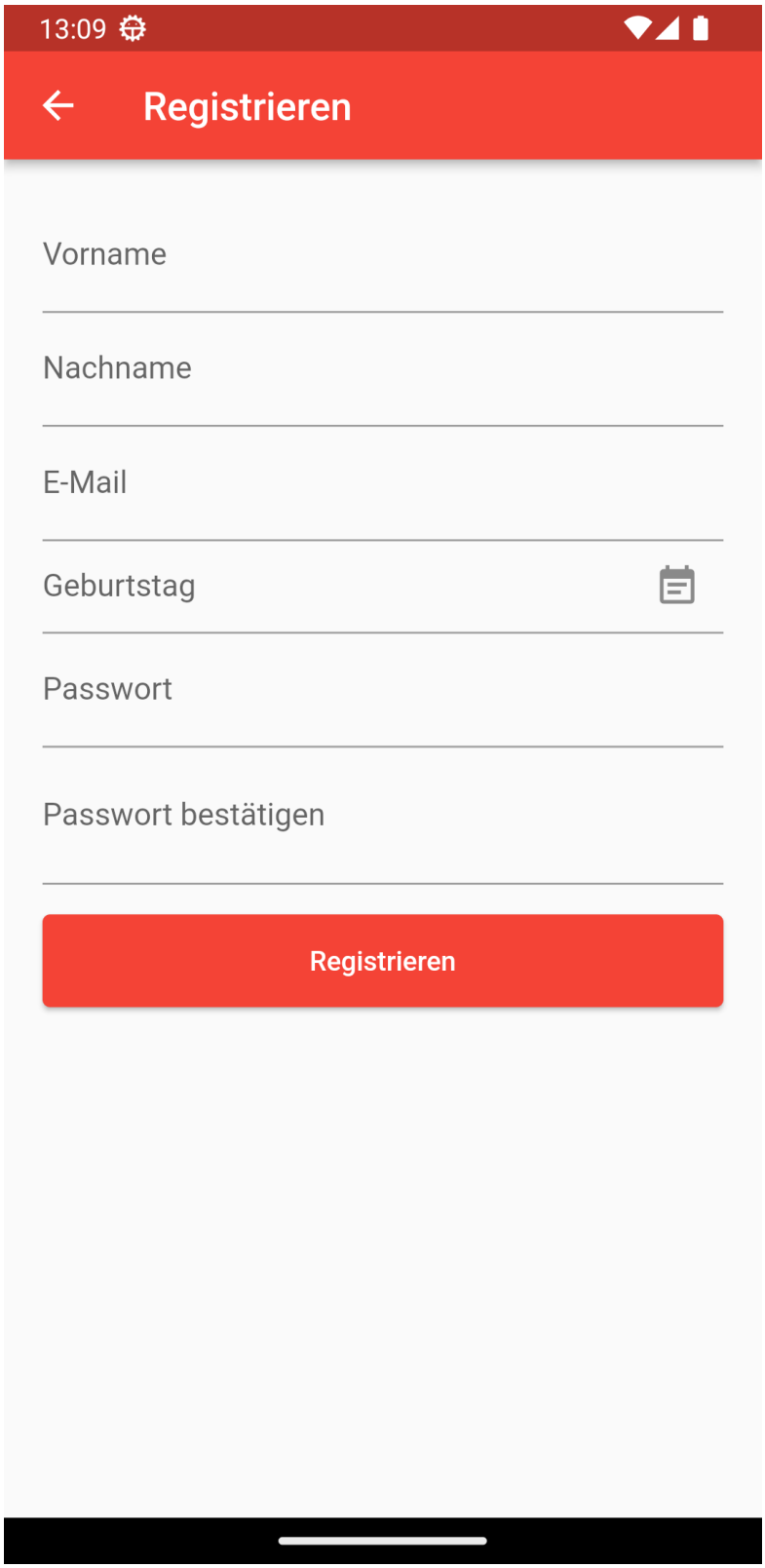

Abbildung 4.4: Registrierung

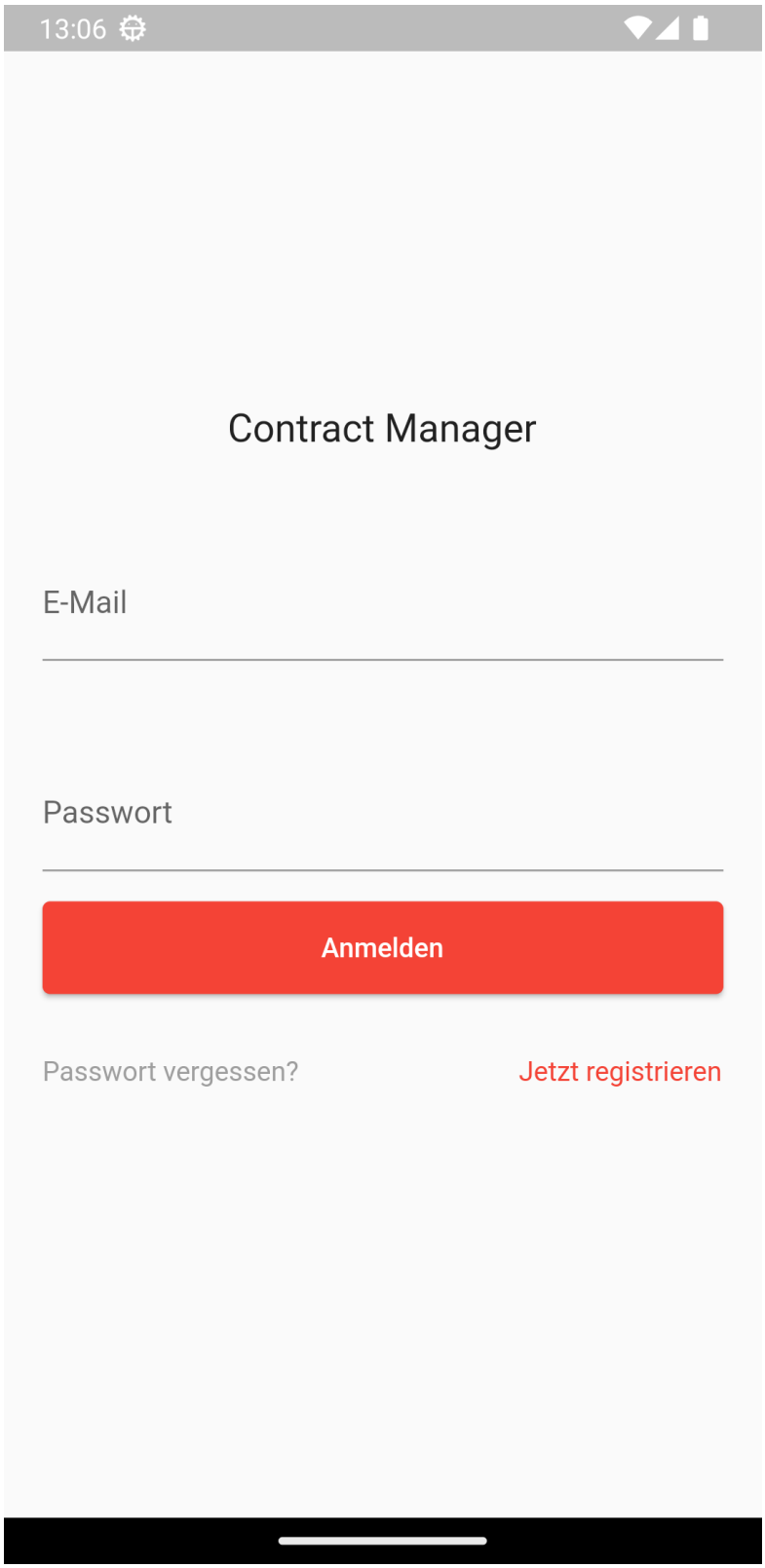

Abbildung 4.5: Login

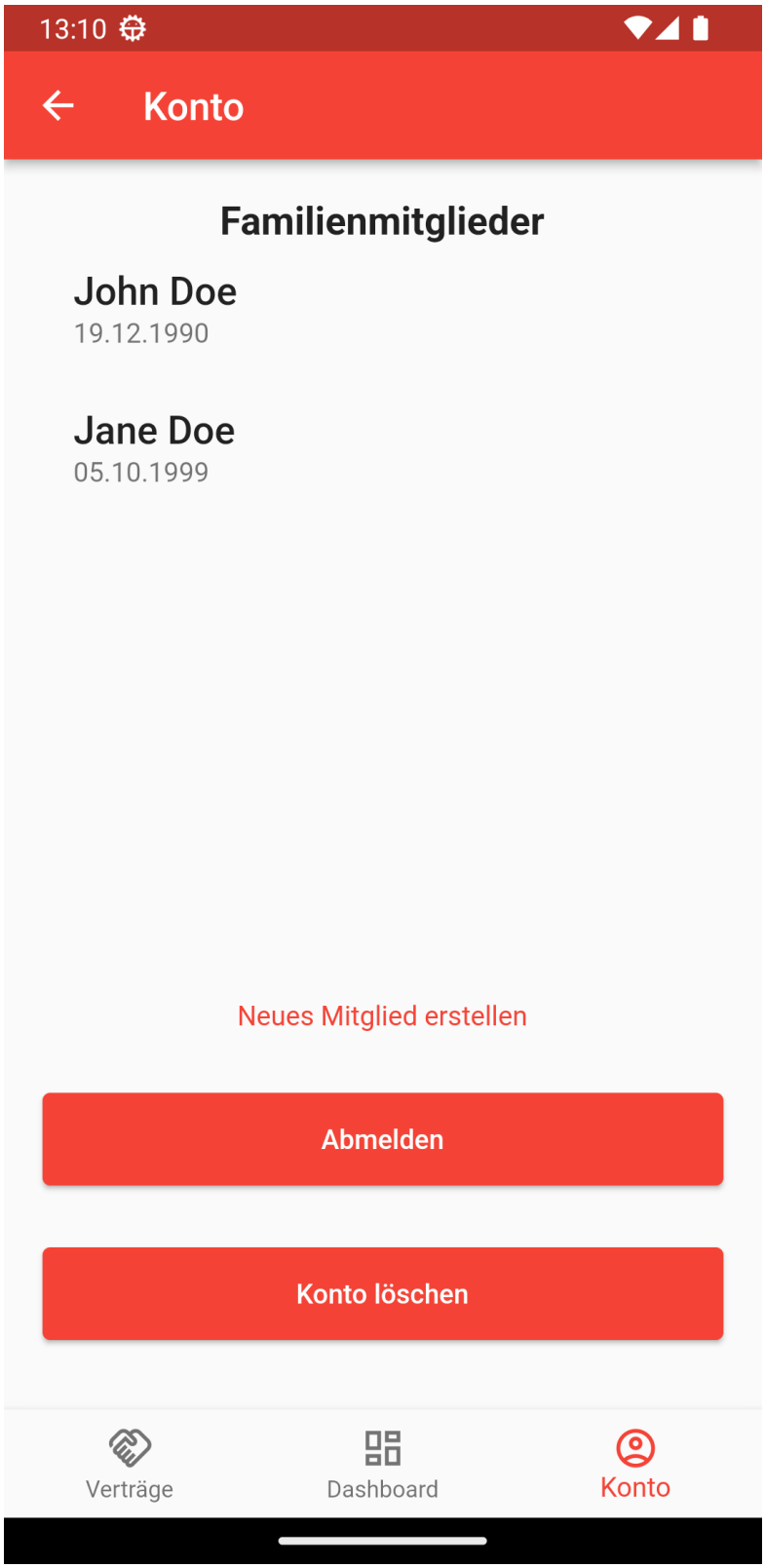

Abbildung 4.6: Account

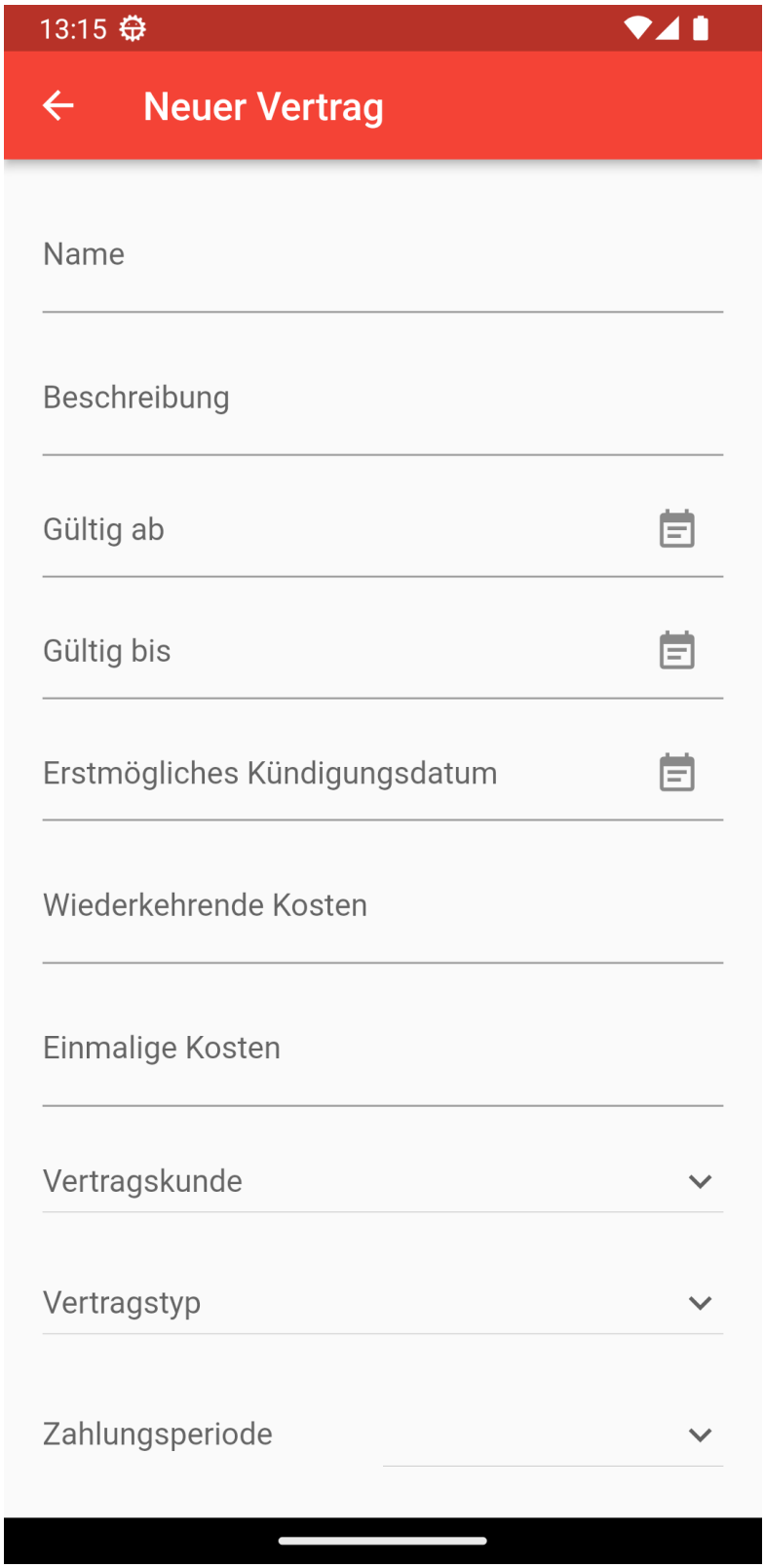

Abbildung 4.7: Vertrag hinzufügen

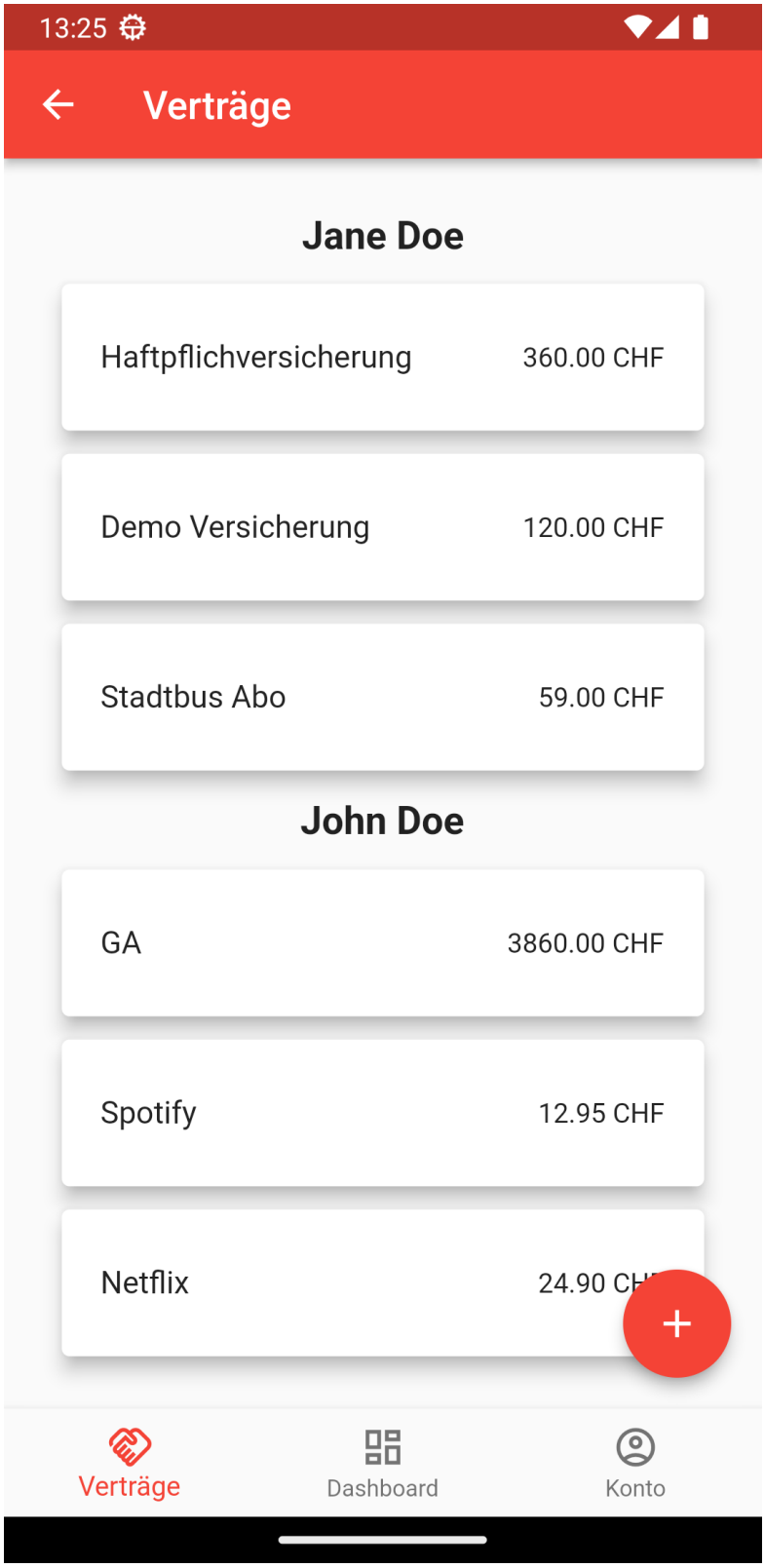

Abbildung 4.8: Übersicht der Verträge

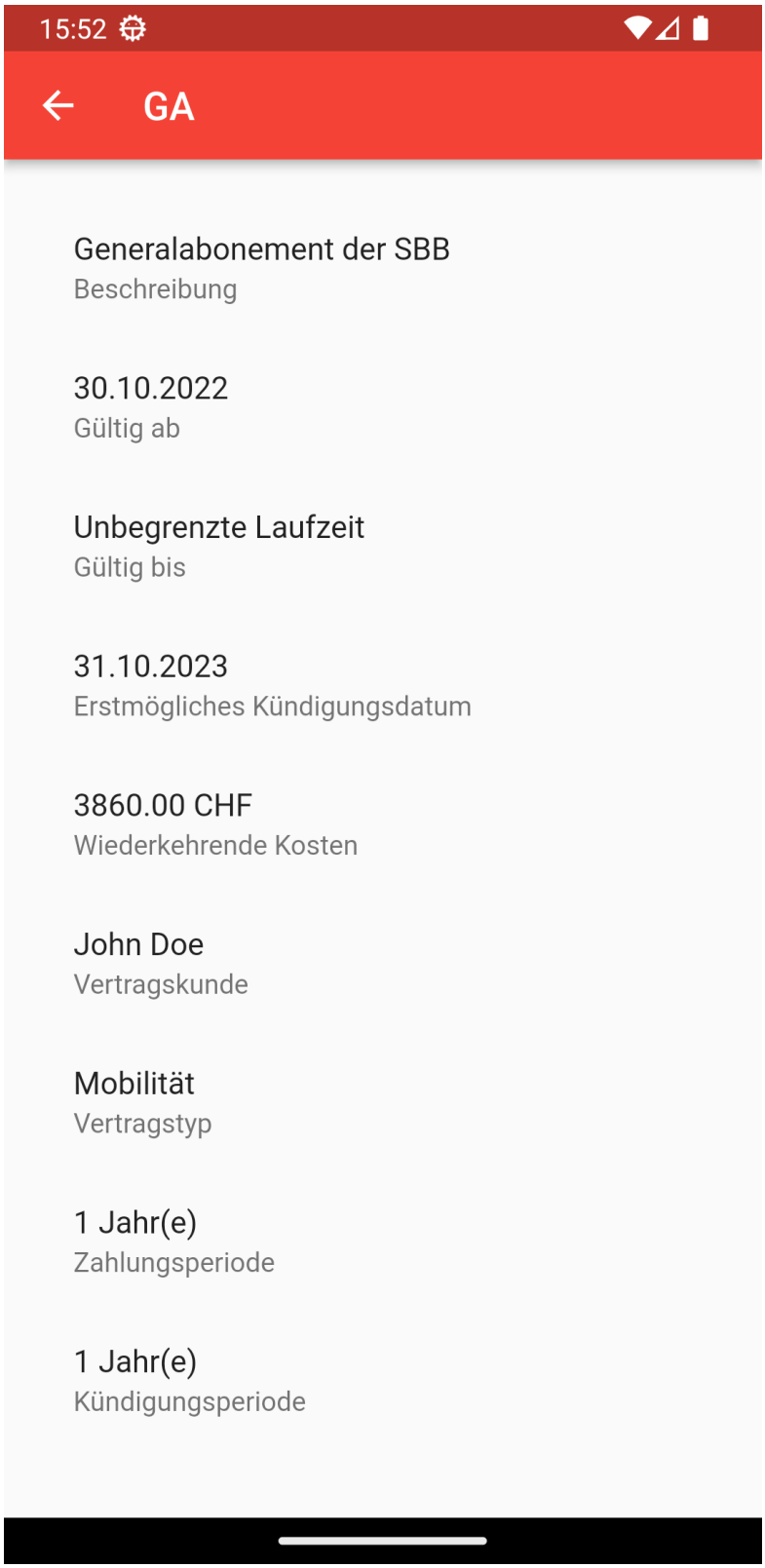

Abbildung 4.9: Details eines Vertrages

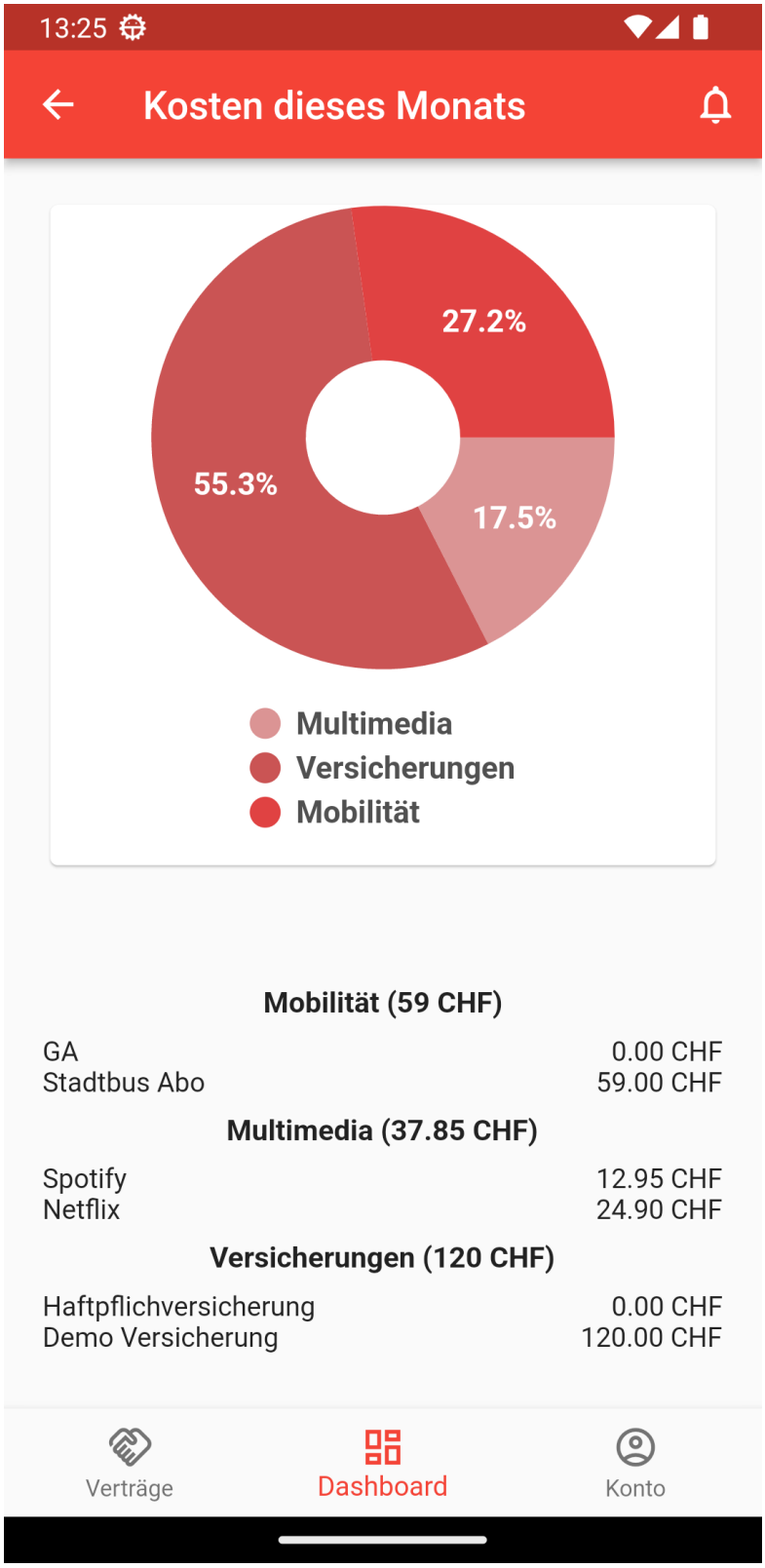

Abbildung 4.10: Dashboard - Übersicht der Kosten dieses Monates

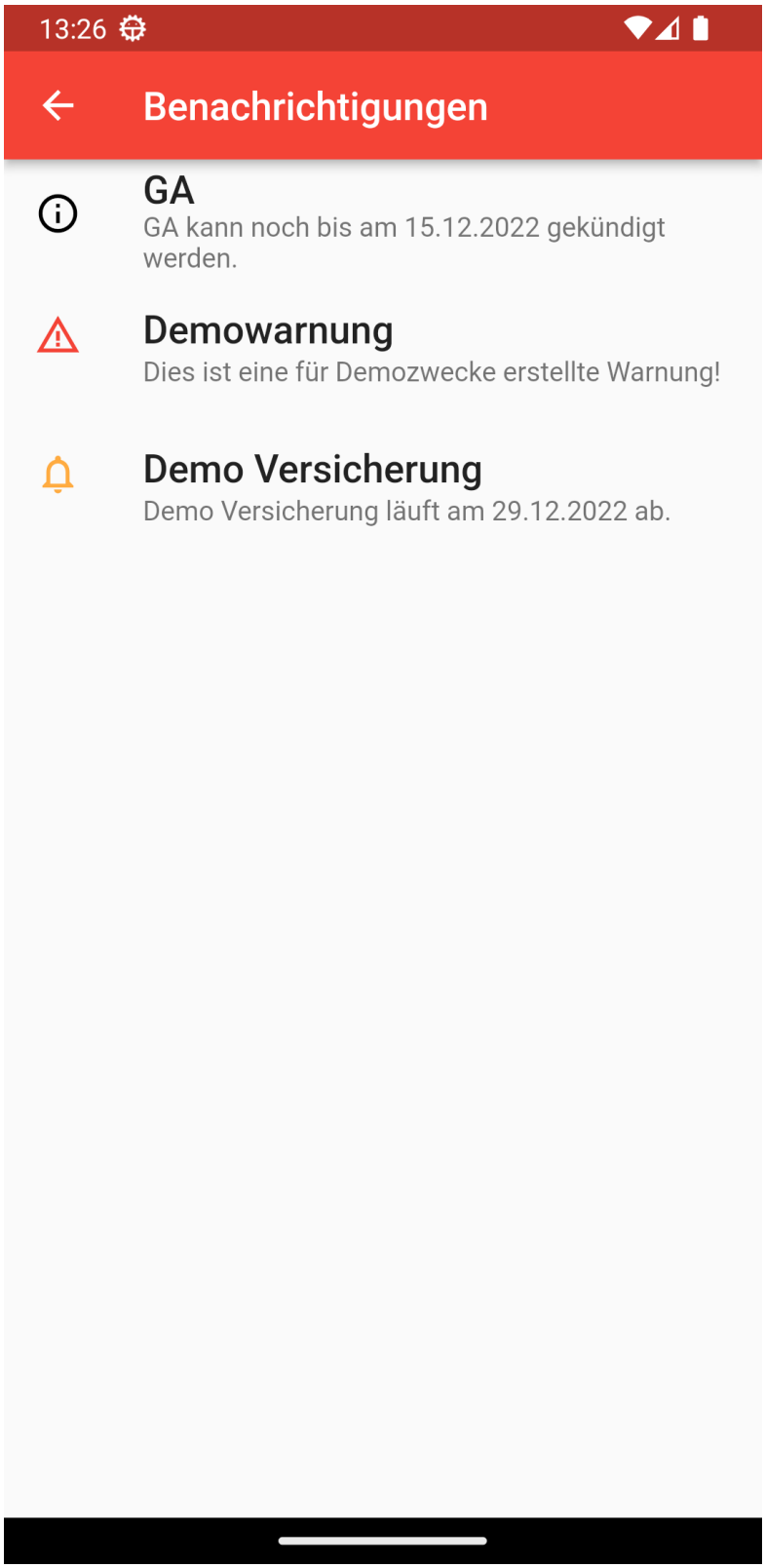

Abbildung 4.11: Benachrichtungs-Screen

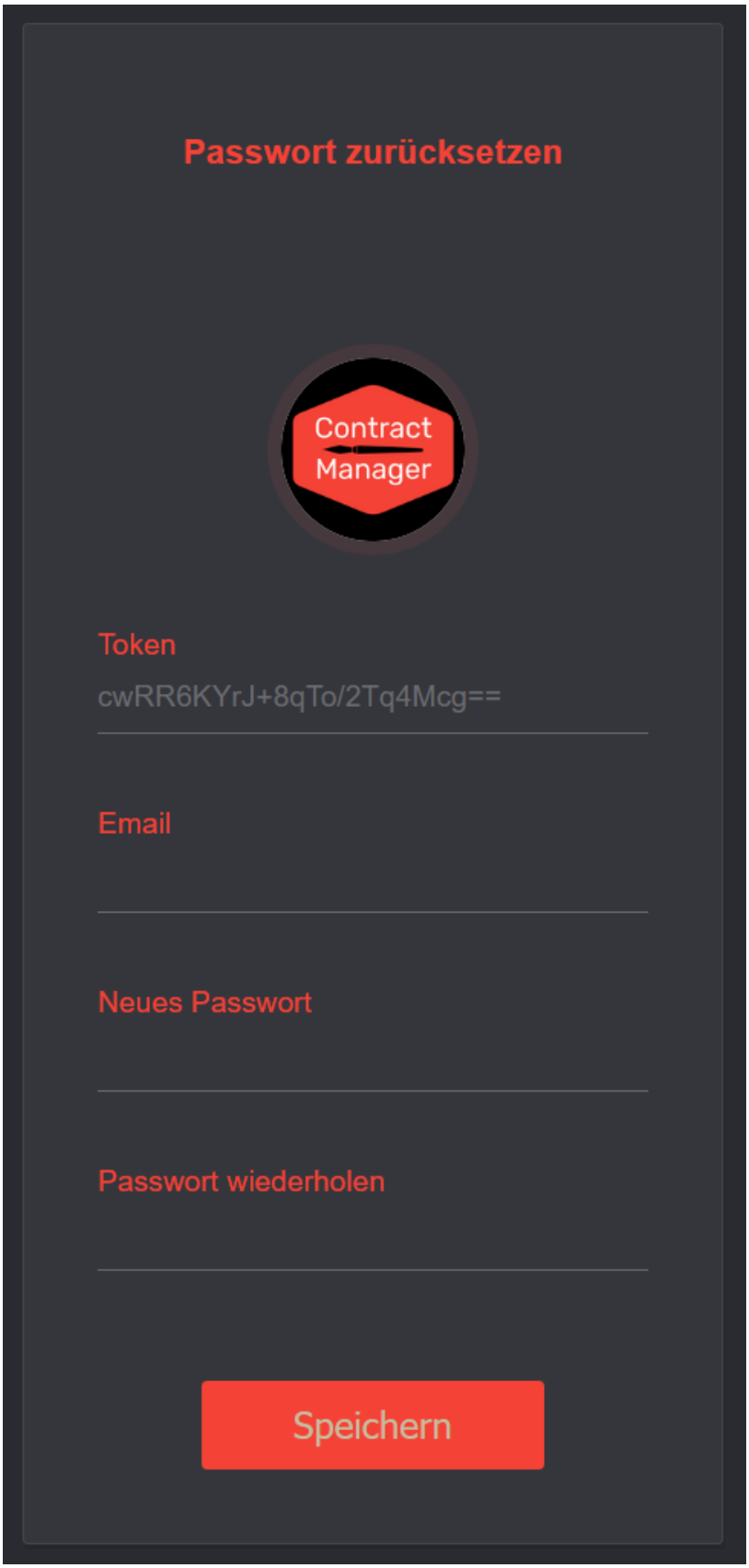

Abbildung 4.12: Passwort-Reset-Webseite

4.4 Einschränkungen

Im Ganzen sind zwei nicht-funktionale und eine funktionale Anforderung nicht umgesetzt.

4.4.1 Dokumente hochladen (FR5)

Das Hochladen von gescannten oder abfotografierten Verträge ist nicht möglich mit der App. (FR5) Das Backend läuft zwar auf DigitalOcean, aber das Hochladen von zusätzlichen Daten wird nicht unterstützt und ist eine mögliche Erweiterung.

4.4.2 Datenbank mit 10'000 Verträge und 500 Benutzern (NFR-3)

Die Anforderung die Datenbank testhalber mit Dummy Daten zu befüllen wurde nicht umgesetzt. Da MongoDB aber einfach zu skalieren ist und doch recht gross werden kann, dürften in der Datenbank problemlos 10'000 Verträge und 500 Benutzer gespeichert werden können.

4.4.3 NoSQL Injection sollen nicht möglich sein (NFR-10)

NoSQL Injection-Tests wurden keine durchgeführt. Somit ist nicht bewiesen, dass die Applikation davor geschützt ist.

5. Weiterentwicklung

Vor Beginn der Studienarbeit wurden schon optionale Features festgelegt. Während der Entwicklungszeit sind einige dazugekommen. Diese könnten für die Weiterentwicklung der Applikation implementiert werden.

5.1 Optionale Anforderungen

ID	Beschreibung
OR1	Benutzende können ihren Account teilen. Geteilte Accounts sollen nur Lese-Zugriff erhalten.
OR2	Benutzende können Abonnementvorlagen verwenden.
OR3	Benutzende können Verträge scannen und im PDF-Format ablegen.
OR4	Externe haben die Möglichkeit, Verträge direkt in das Konto eines Benutzenden hochzuladen. Dafür benötigen sie E-Mail-Adresse sowie das Geburtsdatum des Benutzenden.
OR5	Die Applikation kann mit einer Schnittstelle zu einer Bank oder einem Zahlungsanbieter Transaktionsdaten abgleichen und somit Standardabonnements wie Netflix automatisch erfassen.
OR6	Die Applikation kann auch als Website mit gleicher Funktionalität genutzt werden.
OR7	Die Vertragsdaten werden verschlüsselt auf der Datenbank abgelegt, sodass Administratoren keine Einsicht auf diese Daten haben.
OR8	Die Applikation kann an Vergleichsportale angebunden werden, damit man Vorschläge für Optimierungen bekommt. Dies können zum Beispiel bessere Konditionen einer Versicherung sein.
OR9	Benutzende können zusätzliche Dokumente zu einem Vertrag hochladen. Dies können zum Beispiel nachträgliche Vertragsanpassungen oder Schadensmeldungen einer Versicherung sein.
OR10	Benutzende können individuelle Push-Benachrichtigungen erstellen. Zum Beispiel können so Erinnerungen an bevorstehende Zahlungen erstellt werden.
OR11	Benutzende können innerhalb der Applikation neue Accounts für Familienmitglieder erstellen.
OR12	Benutzende können Verträge zukunftsbedingt anpassen, falls sich etwa Kosten ab einem bekannten Datum ändern.
OR13	Benutzende können im Dashboard aus verschiedenen Ansichten auswählen, etwa damit man nach Kosten filtern kann.

Tabelle 5.1: Optionale Anforderungen

6. Projektplanung

In diesem Kapitel wird auf die Projektplanung während der gesamten Studienarbeit eingegangen. So soll ersichtlich werden, wie die Organisation und das zeitliche Management vonstattenging sowie auf mögliche Risiken reagiert wurde. Zudem sind Informationen über das Qualitätsmanagement gegeben.

6.1 Einführung

Die Projektplanung wird zu Beginn der Studienarbeit festgelegt und stetig angepasst. Dies hat zum Ziel, einen erfolgreichen Projektabschluss zu erreichen. Die Gültigkeit umfasst die gesamte Projektdauer innerhalb des Herbstsemesters 2022.

6.1.1 Lieferumfang

- Dokumentation mit Anhängen
- Technischer Bericht
- Gitlab-Repository mit Programmcode
- Zeitauswertung

6.2 Projektorganisation

Das Projektteam trifft die allerwichtigsten Entscheidungen mit dem Betreuer und dem Industriepartner AdaptIT GmbH, wobei Letzterer die absolute Entscheidung trifft. Innerhalb des Projektteams herrscht eine flache Hierarchie, bei der alle Beteiligten gleichermassen am Projekt mitarbeiten. Die abschliessende Bewertung übernimmt der Betreuer.

6.2.1 Organsisationstruktur

Die Organisation setzt sich aus drei Parteien zusammen. Dies sind das Projektteam, der Betreuer und der Industriepartner.

Name	E-Mail	Rolle
André Blöchlinger	andre.bloechlinger@ost.ch	Teammitglied
Simon Canal	simon.canal@ost.ch	Teammitglied
Linard Vincenz	linard.vincenz@ost.ch	Teammitglied
Prof. Frank Koch	frank.koch@ost.ch	Betreuer
Michael Güntensperger	info@adaptit.ch	Industriepartner

Tabelle 6.1: Organsisationstruktur

6.2.2 Rollen Projektteam

Innerhalb des Projektteams werden zusätzliche Rollen verteilt. Die Rolle bzw. deren Zuständigkeit beschreibt nicht zwingend, wer die damit verbundenen Aufgaben durchführt. Es ist nicht so gedacht, dass nur eine Person dokumentiert. Jedoch hat eine Person ein zusätzliches Augenmerk auf die Dokumentation und somit eine grössere Verantwortung, dass die Aufgaben in diesem Bereich erledigt werden.

Rolle	**Zuständigkeit**
- Meeting protokollieren - Software Architekt	André Blöchlinger
- Kommunikation mit Betreuer - Meeting organisieren - Issue- und Timetracking	Simon Canal
- Dokumentation	Linard Vincenz

Tabelle 6.2: Rollen im Projektteam

6.3 Zeitmanagement

In folgendem Kapitel werden Informationen zu Projektdauer, voraussichtlichem Zeitaufwand und der Zeitplanung gegeben.

6.3.1 Zeitdauer

Der offizielle Beginn der Studienarbeit ist am 19. September 2022. Die Abgabe kann bis zum 23. Dezember 2022 um 17.00 Uhr erfolgen.

Projektdauer:	14 Wochen
Projektbeginn:	19. September 2022
Projektende:	23. Dezember, 17.00 Uhr

Tabelle 6.3: Zeitdauer

6.3.2 Zeitbudget

Die einzelnen Teamitglieder beteiligen sich mit insgesamt 240 Arbeitsstunden am Projekt. Dies ergibt pro Woche einen Arbeitsaufwand von ca. 17 Stunden. Somit stehen dem Projekt insgesamt 720 Stunden zur Verfügung. Dieses Zeitbudget soll nicht mit mehr als 20% überschritten bzw. unterschritten werden.

Anzahl Projektmitarbeiter:	3
Aufwand pro Woche und Person:	17 Stunden
Aufwand Total:	720 Stunden

Tabelle 6.4: Zeitbudget

6.3.3 Zeitplanung

Die Zeitplanung erfolgt mit der Projektmanagementsoftware YouTrack. So ist eine agile Arbeitsweise gewährleistet, da man die Erfassung stetig anpassen kann. Die geplanten sowie die effektiven Zeitaufwände werden auf 30 Minuten genau erfasst.
Zur Planung des Projekts wird RUP sowie Scrum (SCRUM+) verwendet. Dabei wird für die grobe Planung RUP und für die Feinplanung Scrum eingesetzt. Ein Sprint in Scrum erstreckt sich dabei jeweils über eine Woche. Vor jedem neuen Sprint wird jeweils ein Refinement Meeting und ein Sprint Planning durchgeführt, um die Tasks für den neuen Sprint zu definieren. Während eines Sprints wird pro Woche ein "Daily"-Scrum abgehalten, um den Status zu überprüfen und allfällige Fragen zu klären. Am Ende eines Sprints werden dann abschliessend jeweils sowohl ein Sprint Review als auch eine Sprint Retrospective durchgeführt, um in Erfahrung zu bringen, ob alle Tasks abgearbeitet wurden und gegebenenfalls Justierungen vorgenommen werden müssen.

Phase	Start	Ende
Inception	19.09.2022 (SW01)	02.10.2022 (SW02)
Elaboration	03.10.2022 (SW03)	23.10.2022 (SW05)
Construction	24.10.2022 (SW06)	11.12.2022 (SW12)
Transition	12.12.2022 (SW13)	23.12.2022 (SW14)

Tabelle 6.5: Zeitplanung

6.4 Meilensteine

Die Meilensteine wurden nach den wichtigsten Epics bestimmt.

6.4.1 M0: Kick off - 21.09.2022

Der Projektstart ist lanciert und die erste Sitzung mit Betreuer und Industriepartner wurde durchgeführt. Hierbei wurden die Aufgabenstellung, das weitere Vorgehen und Administratives besprochen.

6.4.2 M1: Prototyp lauffähig - 09.10.2022

Bis hierhin ist der Projektplan, das Domain Model, UI-Design und die Systemarchitektur erstellt. Man macht sich mit den unterschiedlichen Technologien vertraut und ein Prototyp wird erstellt.

6.4.3 M2: Registrierung und Login - 23.10.2022

Die Registrierung und das Login funktionieren. Das Deployment auf DigitalOcean ist eingerichtet. Sämtliche Abhängigkeiten zu externen Systemen sind implementiert. Die Einrichtung von Google Analytics ist abgeschlossen.

6.4.4 M3: CRUD Contracts - 06.11.2022

Man ist in der Lage, Verträge zu erstellen, diese zu lesen und zu bearbeiten sowie zu löschen. Die Benachrichtigungen über auslaufende Verträge und Kündigungstermine funktionieren.

6.4.5 M4: Usability Test - 27.11.2022

Die wichtigsten Funktionen der App sind erstellt. Usability Tests werden durchgeführt. Das Produkt ist in einem nahezu fertigen Zustand. Somit steht der MVP, das minimal brauchbare Endprodukt.

6.4.6 M5: Abgabe Projekt - 23.12.2022, 17.00 Uhr

Das Projekt ist abgeschlossen. Die gesamte Dokumentation und der Quellcode werden abgeben.

6.5 Iterationen (Sprints)

Nachfolgend sind die Sprints abgebildet, welche als Grundlage der Planung dienen sollten.

6.5.1 Sprint 0

Produkt	Dokumentation
• Aufgabenstellung verstehen • Einrichtung der Projektmanagement-tools	• Erstellung Projektplan • Vorbereitung Dokumentation • Entwurf Domain Model erstellen

Tabelle 6.6: Sprint 0

6.5.2 Sprint 1

Produkt	Dokumentation
• Technologiestack definieren • Entwicklungsumgebung einrichten	• Anpassungen Domain Model • Entwurf UI Design erstellen • Durchführung Risikoanalyse

Tabelle 6.7: Sprint 1

6.5.3 Sprint 2

Produkt	Dokumentation
• Einrichtung Hosting Provider • Erstellung Prototyp	• Anpassung UI Design • Entwurf Architektur erstellen • Erstellung Deploymentstrategie

Tabelle 6.8: Sprint 2

6.5.4 Sprint 3

Produkt	Dokumentation
• Implementation Registrierung und Login	• Anpassung Architektur

Tabelle 6.9: Sprint 3

6.5.5 Sprint 4

Produkt	Dokumentation
• Abschluss Registrierung und Login • Implementation Push-Benachrichtigungen	

Tabelle 6.10: Sprint 4

6.5.6 Sprint 5

Produkt	Dokumentation
• Implementation CRUD Contracts	• Aktualisierung Risikoanalyse

Tabelle 6.11: Sprint 5

6.5.7 Sprint 6

Produkt	Dokumentation
• Abschluss CRUD Contracts • Implementation Dashboard	

Tabelle 6.12: Sprint 6

6.5.8 Sprint 7

Produkt	Dokumentation
• Einrichtung Google Analytics • Backlog aufarbeiten	

Tabelle 6.13: Sprint 7

6.5.9 Sprint 8

Produkt	Dokumentation
• Unit-Tests schreiben • Code optimieren	

Tabelle 6.14: Sprint 8

6.5.10 Sprint 9

Produkt	Dokumentation
• Abschluss Dashboard • Einrichtung Google Play Store • Einrichtung Apple Store	• Durchführung Usability Tests

Tabelle 6.15: Sprint 9

6.5.11 Sprint 10

Produkt	Dokumentation
• Feedback aus Usability Test umsetzen • Implementierung optionale Features	

Tabelle 6.16: Sprint 10

6.5.12 Sprint 11

Produkt	Dokumentation
• Code optimieren, Refactoring • Bug Fixes • Implementierung optionale Features	

Tabelle 6.17: Sprint 11

6.5.13 Sprint 12

Produkt	Dokumentation
• Code optimieren, Refactoring • Bug Fixes • Reserve	• Implementierung beschreiben • Weiterentwicklung beschreiben

Tabelle 6.18: Sprint 12

6.5.14 Sprint 13

Produkt	Dokumentation
• Bug Fixes • Reserve	• Management Summary • Projektmonitoring • Fazit • Eigenständigkeitserklärung

Tabelle 6.19: Sprint 13

6.6 Besprechungen

Nachfolgend sind Informationen zu finden, welche über die unterschiedlichen Besprechungen innerhalb des Projektteams sowie mit Betreuer und Industriepartner Auskunft geben.

6.6.1 Besprechung mit Betreuer und Industriepartner

Das Projektteam trifft sich wöchentlich am Donnerstag über Microsoft Teams mit dem Betreuer und dem Industriepartner. Die Sitzung dauert maximal eine Stunde lang, in welcher der aktuelle Stand, das weitere Vorgehen und anfallende Fragen besprochen werden. Falls das Projektteam keinen Mehrwert in einer Sitzung sieht, kann man diese im Vorfeld absagen. Die Sitzungen werden protokolliert und das Besprechungsprotokoll an alle Beteiligten verschickt. Vor einer Sitzung werden Traktanden festgelegt und versendet.

6.6.2 Sprint Planning

Das Projektteam trifft sich jeden Montag am Campus Rapperswil oder per Teams, um den vergangenen Sprint zu beenden sowie den neuen zu planen.

- Sprint Review (08:00 - 08:30)
  - Vergangener Sprint überblicken
- Sprint Retrospective (08:30 - 09:30)
  - über vergangenen Sprint reflektieren
  - vergangener Sprint abschliessen
- Refinement Meeting (09:30 - 10:30)
  - Verfeinerung von Epics und User Stories in Tasks
- Sprint Planning (10:30 - 11:00)
  - Feinplanung für nächsten Sprint erstellen

Projekt Planung

Alle 2 Wochen wird die Grobplanung überarbeitet. Hierbei trifft sich das Projektteam nach der Sprint Planning Besprechung. Das Meeting soll etwa 15 Minuten dauern.

6.7 Risikomanagement

Zu diesem Kapitel erhält man Einblicke in die Risikofestlegung und wie mit Risiken umgegangen wird.

6.7.1 Erkannte Risiken

Im Verlauf eines Projekts können jederzeit Risiken eintreten. Damit man einen möglichen Schaden besser einschätzen kann, sind die erkannten Risiken mit potenzieller Eintrittswahrscheinlichkeit [%] und dadurch vorstellbar entstandene Schäden [h] erfasst. Da es sich hierbei um Schätzungen handelt, sind diese Zahlen nicht als endgültig zu betrachten.

Nr.	Beschreibung	[h]	%	Gewichtung
R1	Ungenaue Zeitschätzung, was zu Zeitmangel führt	24	60%	14.4
R2	Softwarearchitektur ist ungeeignet	20	40%	8
R3	Ausfall von Teammitgliedern	32	30%	9.6
R4	Verlust von Daten	8	10%	0.8
R5	Das Domain Model ist ungeeignet	16	20%	3.2
R6	Die ausgewählten Technologien sind inkompatibel zueinander	20	15%	3
R7	Technologie-Kenntnisse sind unzureichend	28	35%	9.8
R8	Missverständnisse aufgrund von mangelnder Kommunikation	24	50%	12
R9	Applikation ist unverständlich und unbefriedigend	20	15%	3
R10	Konflikte im Team	4	10%	0.4
	Total Schaden:	**196**		**Total Gewichtung: 64.2**

Tabelle 6.20: Risiken

6.7.2 Risikomatrix

In der Risikomatrix sind die ursprünglich erkannten und dokumentierten Risiken dargestellt.

Probabilität	Schweregrad			
	Vernachlässigbar	Marginal	Kritisch	Katastrophal
Hoch				
Wahrscheinlich				
Möglicherweise			R1, R8	
Unwahrscheinlich			R2	R3, R7
Selten	R4, R10	R5	R6, R9	

Tabelle 6.21: Risikomatrix

6.7.3 Vorgehen mit Risiken

Zur Risikominimierung sind nachfolgend Erklärungen gegeben, welche behilflich sind, um den Risikoeintritt bestmöglich zu verhindern. Zusätzlich sind Massnahmen zu entnehmen, falls ein Risiko tatsächlich eintreten sollte.

Nr.	Vorbeugung	Verhalten beim Eintreten
R1	Planung regelmässig anpassen, damit man frühzeitig erkennt, falls etwas nicht funktionieren sollte wie gewünscht	Sinnvolle Priorisierung der Aufgaben, damit trotzdem ein lauffähiges und nutzbares Produkt am Ende des Projekts zur Verfügung steht
R2	Genug Zeit für Architektur nehmen, gut planen und reviewen lassen	Alternative erarbeiten
R3	Gesunde Ernährung und Sport sowie Puffer-Slots einbauen	Sinnvolle Neupriorisierung der Aufgaben, allenfalls Anforderungen streichen
R4	Teammitglieder behalten ihre Änderungen auch nach dem Mergen auf ihren lokalen Maschinen. Regelmässig den aktuellen Stand auf Gitlab pushen	Backup aus lokalen Daten wiederherstellen
R5	Genug Zeit für Domain Model nehmen, gut planen und reviewen lassen	Anpassung mit allerhöchster Priorität vornehmen
R6	Gute und genaue Planung sowie Sondierung bei der Auswahl der Technologien	Prüfung der Problemumgehung, falls nicht möglich Technologiewechsel vornehmen
R7	Genügend Zeit für die Informationsbeschaffung, offizielle Dokumentationen hinzuziehen	Offizielle Dokumentationen hinzuziehen, Ansprechspersonen oder Community fragen
R8	Informationskanal festlegen und regelmässige Besprechungen abhalten	Ausserplanmässige Besprechungen durchführen
R9	UI Design frühzeitig besprechen und Usability Tests durchführen	Applikation gemäss Tester-Feedback anpassen
R10	Offener und direkter Austausch untereinander	Intervention durch Betreuer

Tabelle 6.22: Vorgehen mit Risiken

6.8 Arbeitspakete

Die Arbeitspakete sind aufgrund der funktionalen Anforderungen und der Meilensteine grob definiert. Diese sind als Epics im YouTrack erfasst und mit einem relativen Aufwand geschätzt. Innerhalb des Sprint Plannings werden diese Epics zu kleineren User Stories und diese wiederum zu Aufgaben spezifiziert. Die Aufgaben sind mit einer absoluten Aufwandsschätzung bestückt.

6.9 Qualitätsmassnahmen

Damit das Produkt eine hohe Qualität aufweist, werden nachfolgende Vorkehrungen getroffen.

Massnahme	Zeitraum	Ziel
Sprint Reviews	Sprintende	Es wird regelmässig über den aktuellen Stand informiert, damit man Probleme frühzeitig detektieren kann
Code Reviews	Merge Request	Einschleichung von Fehlern vermeiden und sauberen Code garantieren
Meilensteine	Fixer Tag	Erfolgreiche Beendigung des Projekts soll so gewährleistet sein.
Automatisiertes Testing	Push	Testen der Funktionalität des Codes
Definition of done	Beenden der Aufgabe	DoD ist gegeben, falls 6.9.1 eingehalten sind

Tabelle 6.23: Arbeitspakete

6.9.1 Definition of done

- CI erfolgreich, bei Status "Failed" zwingend Fehlerkorrektur durchführen
- Realisierung der Coding Guidelines erfüllt
- Code Review erfolgreich
- Bei Sprint Reviews erfolgt die Beendigung des Tasks endgültig

6.9.2 Dokumentation

Die gesamte Dokumentation ist auf dem Gitlab Server der Ostschweizer Fachhochschule gespeichert. Das Projektteam arbeitet direkt auf dem Master Branch. Bei jedem Push wird das Dokument im PDF-Format als Artifakt neu kompiliert. Die Versionisierung wird dadurch sichergestellt.

6.9.3 Projektmanagement

Um das Projektmanagement zu überwachen, wird YouTrack verwendet. Die Aufgaben werden durch eine Aufwandsschätzung erfasst. Der tatsächliche Aufwand wird nachgetragen. Unter dem Reiter Bericht sind die Gesamtzeiten der Mitglieder, Zeiten nach Aufgabe sowie nach Sprint und Epics einzusehen.

6.9.4 Code Reviews

Jedes Projektmitglied erstellt nach der Fertigstellung einer Aufgabe einen Merge Request und weist diesen einem Teammitglied zu und informiert das Team. Das zugewiesene

Teammitglied reagiert auf die Information und überprüft zeitnah den Merge Request, ob alle Qualitätsmassnahmen sauber eingehalten sind. Darauf wird der Merge Request bestätigt. Nach der Bestätigung wird wieder das ganze Projektteam informiert.

6.9.5 Code Style Guides

Im Backend wird ein empfohlener Style Guide [17] verwendet. Im Frontend wird Flutter's interner Linter[13] eingesetzt. Durch Beachten dieser Guides kann man einheitlichen Code erstellen.

6.10 Projektmonitoring

In diesem Kapitel sind Erläuterungen zum Projektverlauf zu finden.

6.10.1 Zeitauswertung

Nachfolgend sind einige Zeitauswertungen dokumentiert. Diese sind nach dem gesamten Arbeitsaufwand pro Teammitglied und nach Aufwand pro Kategorie definiert. Ausserdem gibt es eine Darstellung des geschätzten bzw. des effektiven Aufwands. Auch der gesamte Projektplan ist hier zu entnehmen.

Arbeitsverteilung

Die Grafik veranschaulicht den Zeitaufwand in Stunden für jedes Projektmitglied.

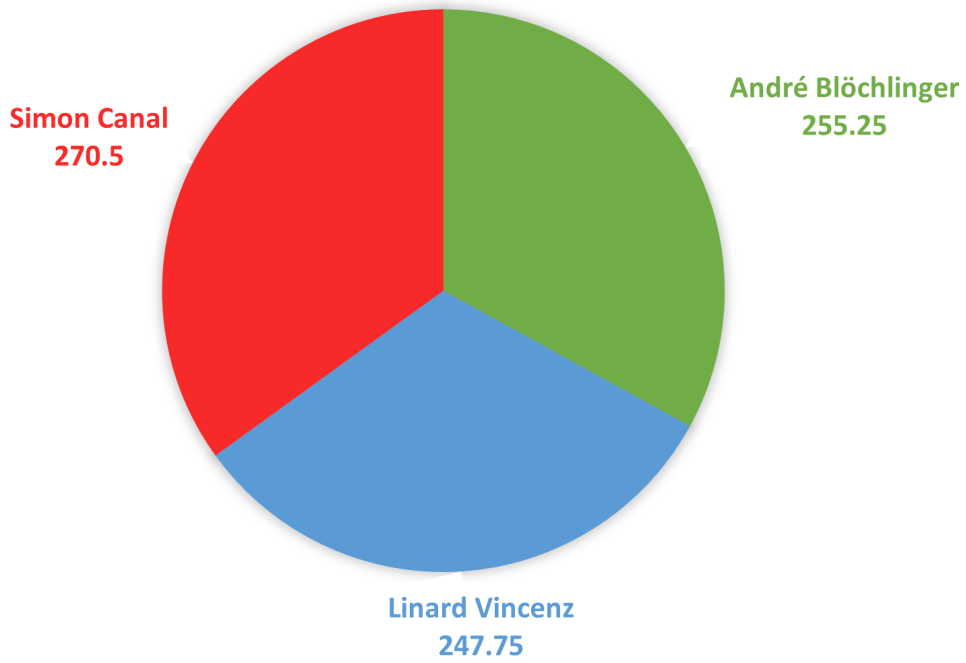

Abbildung 6.1: Zeitauswertung nach Projektmitglied

Arbeitsbereich

Alle erstellten Aufgaben sind einer Kategorie zugewiesen. Insgesamt wurden 9 Kategorien definiert. Alle Werte in der Grafik sind in Stunden angegeben.

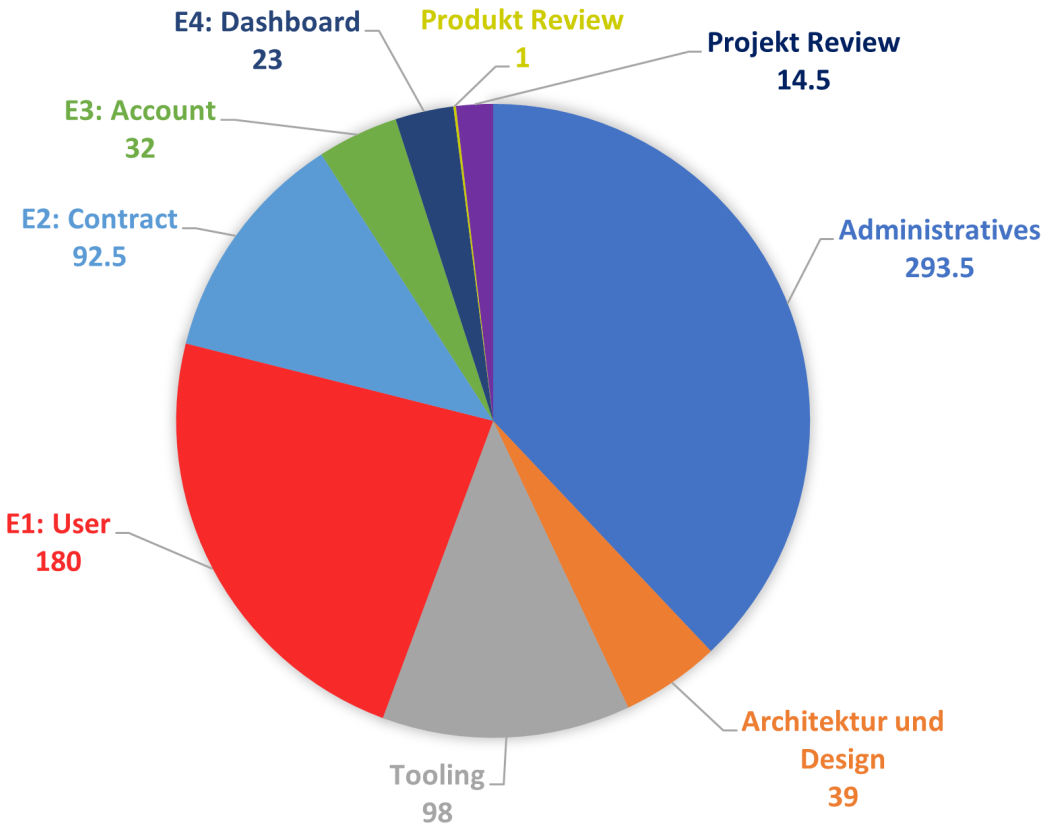

Abbildung 6.2: Zeitauswertung nach Kategorie

Soll-Ist-Vergleich

Bei Projektbeginn wurden Zeitschätzungen für die Kategorien erstellt. In dem unten stehenden Diagramm sind diese Schätzungen in Relation zu den effektiven Zeitaufwänden dargestellt. Zu sehen ist, dass im Bereich Administratives der Aufwand massiv fehlgeschätzt wurde.

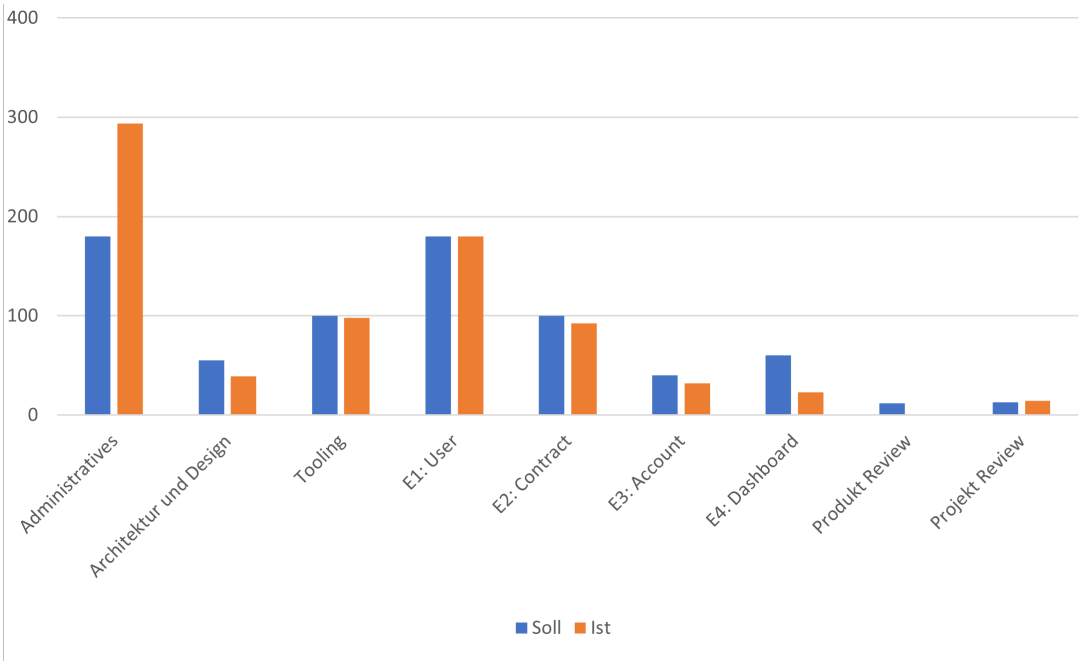

Abbildung 6.3: Soll-Ist-Vergleich

Meilenstein-Einhaltung

Von den fünf definierten Meilensteinen sind drei nicht rechtzeitig erreicht worden. Dies hat vor allem damit zu tun, dass das Projektteam zu Beginn des Projekts die Wissenssbeschaffung für Flutter unterschätzt hat. Im Projektplan 6.10.3 sind die verpassten bzw. die erreichten Meilensteine ersichtlich.

6.10.2 Implementierungsziele

Im Folgenden wird auf die Implementierungsziele eingegangen. Zu sehen sind erfüllte Anforderungen sowie nicht erfüllte Anforderungen. Während des Projekts wurde klar, dass nicht alle Anforderungen wunschgemäss erfüllt werden konnten. Das Projektteam hat deshalb in einer Besprechung mit dem Industriepartner und dem Betreuer eine Neueinschätzung der Anforderungen gemacht. Daraus konnte ein Neupriorisierung erzielt werden, welche von allen Parteien abgesegnet wurde.

Funktionale Anforderungen

Die funktionalen Anforderungen verweisen auf die Tabelle 2.1.

ID	Umsetzung
FR1	Umgesetzt
FR2	Umgesetzt
FR3	Umgesetzt
FR4	Umgesetzt
FR5	Nicht umgesetzt
FR6	Umgesetzt
FR7	Umgesetzt
FR8	Umgesetzt
FR9	Umgesetzt
FR10	Umgesetzt
FR11	Teilweise umgesetzt
FR12	Umgesetzt

Tabelle 6.24: Implementierungsziele Funktionale Anforderungen

FR5 wurde während des Projektverlaufs gestrichen aufgrund fehlender Ressourcen. Bei FR11 wurde auf den Google Play Store reduziert. Die Applikation ist dort als interner Test eingerichtet und ist zum Herunterladen bereit.

Nicht-Funktionale Anforderungen

Die nicht-funktionalen Anforderungen verweisen auf die Anforderungen in Kapitel 2.2.

ID	Umsetzung
NFR1	Umgesetzt
NFR2	Umgesetzt
NFR3	Umgesetzt
NFR4	Umgesetzt
NFR5	Nicht umgesetzt
NFR6	Teilweise umgesetzt
NFR7	Umgesetzt
NFR8	Umgesetzt
NFR9	Umgesetzt
NFR10	Teilweise umgesetzt
NFR11	Umgesetzt
NFR12	Umgesetzt
NFR13	Umgesetzt
NFR14	Umgesetzt
NFR15	Umgesetzt

Tabelle 6.25: Implementierungsziele nicht-funktionale Anforderungen

Bei NFR6 wurde aufgrund mangelnder Resourcen die Anzahl der Test-User auf 2 reduziert. Bei NFR10 wurde aus selbigem Grund der Test der SQL-Injection gestrichen.

6.10.3 Projektplan

Der folgende Projektplan zeigt den geplanten sowie den effektiven Arbeitsaufwand. Zudem sind die Meilensteine ersichtlich und ob diese eingehalten wurden.

	SOLL-Aufwand IST-Aufwand [in Stunden]	Termine	SW1 KW38	SW2 KW39	SW3 KW40	SW4 KW41	SW5 KW42	SW6 KW43	SW7 KW44	SW8 KW45	SW9 KW46	SW10 KW47	SW11 KW48	SW12 KW49	SW13 KW50	SW14 KW51
Inception																
Administratives	160															
	128															
Elaboration																
Architektur und Design	55															
	39															
Tooling	100															
	98															
Construction																
E1: User	180															
	180															
E2: Contract	100															
	92.5															
E3: Account	40															
	32															
E4: Dashboard	60															
	23															
M1: Prototyp lauffähig		09.10.2022			⋆											
		✓			⋆											
M2: Registrierung und Login		23.10.2022					⋆									
		×						⋆								
M3: CRUD Contracts		06.11.2022							⋆							
		×										⋆				
M4: Usability Test		27.11.2022										⋆				
		×													⋆	
Transition																
Produkt Review	12															
	1															
Projekt Review	13															
	14.5															
M5: Abgabe Projekt		23.12.2022														⋆
		✓														⋆
Projektende																
Gesamtstunden geplant	720															
Gesamtstunden geleistet																

Legende	
Meilenstein	⋆
Geplante Arbeit	
Effektiv daran gearbeitet	
Daran ungeplant gearbeitet	
Zeitpuffer für Risiken	
Termin eingehalten	✓

7. Schlussbericht

In diesem Kapitel sind Erfahrungsberichte, Danksagungen sowie das Fazit über das Projekt erfasst.

7.1 Erfahrungsberichte

Jedes Teammitglied berichtet über seine persönlichen Erfahrungen während des Projektverlaufs.

7.1.1 André Blöchlinger

Mit dem Sprichwort "Aller Anfang ist schwer." motivierte ich mich vor Beginn der Studienarbeit. Doch mit viel Erleichterung stellte ich fest, dass dies nicht auf diese Arbeit zutrifft. Als ich Linard und Simon bei Projektbeginn kennenlernte, konnte ich feststellen, dass wir auf einer Wellenlänge sind. Deshalb war die Zusammenarbeit mit ihnen sensationell. Zudem war die Betreuung von Frank und Michael sehr hilfreich und sie liessen uns viele Freiheiten in der Realisierung des Projekts.
Die Anfangsphase lief reibungslos ab. Spätestens bei der Implementierung kamen wir von unserem gewünschten Kurs ab, da die eingesetzten Technologien teilweise bis ganz unbekannt für uns waren. Vor allem mit Flutter habe ich mit einigen Problemen gekämpft. Dank unserer agilen Projektplanung wurde uns im Verlauf des Projekts bewusst, dass wir nicht alle Anforderungen wie gewünscht umsetzen können. So haben wir das Gespräch mit Betreuer und Industriepartner gesucht und einen gemeinsamen Nenner finden können. Nichtsdestotrotz konnten wir eine Mobile App entwickeln, welche ich für nützlich halte, jedoch noch sehr ausbaufähig ist.
Meine wichtigsten Erkenntnisse aus der Studienarbeit sind die Wichtigkeit der Zusammenarbeit sowie eine gute Ausarbeitung des Projektmanagements. Und nicht zuletzt die dazugewonnene Erfahrungen und Fachkenntnisse.

7.1.2 Linard Vincenz

Die Zusammenarbeit im Team hat sehr gut geklappt. Wir haben von Anfang an sehr gut miteinander harmoniert, was die Arbeit stark erleichtert. Simon und ich kannten uns bereits etwas besser und haben auch schon das eine oder andere Projekt zusammen gestemmt. Da ich André vorher noch nicht kannte, wusste ich nicht, wie die Zusammenarbeit klappen würde. Aber es hat wunderbar funktioniert.
In der Elaborations-Phase waren wir effizient und konnten dadurch schnell mit der Entwicklung anfangen. Da wir aber auf teils gänzlich neue bzw. für uns unbekannte Technologien setzen mussten, war der Anfang recht steinig. Deswegen mussten wir viel ausprobieren und Zeit in die Recherche stecken. Dies war zwar spannend und lehrreich, wurde aber gegen Ende des Projektes je länger, je mehr zum Zeitproblem. Aus diesem Grund mussten wir in Absprache mit Betreuer und Industriepartner einige Anforderungen streichen. Dies ist natürlich sehr schade. Aber am Ende konnte wir trotzdem noch einiges Umsetzen, was wir zuvor eigentlich gestrichen hatten. Das macht sogleich auch wieder mehr Freude, wenn sich das Endprodukt sehen lässt. Meiner Meinung nach haben wir eine gute Basis für eine Weiterentwicklung erreicht.

7.1.3 Simon Canal

Unser dreier Team wurde spontan zusammengesetzt, da Linard und ich mich für dieses Thema beworben haben und André noch ein Teammitglied suchte. Wir haben uns von Beginn an super verstanden und funktionierten als Team so, als ob wir bereits seit Jahren zusammenarbeiten würden.
Die Arbeit an der Studienarbeit war eine spannende, lehrreiche, aber auch sehr anstrengende Zeit. Viele der eingesetzten Technologien waren für uns neu. Zum Einen ist das schön, da man dadurch neue Sachen kennenlernt und sich weiterentwickeln kann. Zum Anderen mangelt es gerade während eines solchen Projekts an Zeit, sich damit auseinanderzusetzen.
Die Tatsache, dass wir vieles recherchieren, lernen und ausprobieren mussten, brachte unseren Zeitplan dermassen durcheinander, dass schlussendlich die Anforderungen angepasst werden mussten. Dank unseres agilen Projektmanagements und frühzeitigem Reagieren, konnte rechtzeitig eingegriffen werden. Somit konnten wir schlussendlich trotzdem das Projekt erfolgreich beenden und meiner Meinung nach ein gutes Endprodukt liefern, was sich in Zukunft noch weiterentwickeln lässt.

7.2 Danksagung

Während der Studienarbeit standen wir in regelmässigem Kontakt mit unserem Betreuer Frank Koch sowie Michael Güntensperger von AdaptIT als Industriepartner. Die Zusammenarbeit mit beiden war immer unkompliziert und zielorientiert und bleibt uns gut in Erinnerung.
Besonders bedanken möchten wir uns für die Unterstützung, das stets offene Ohr für Fragen sowie die Kompromisse, welche wir eingehen konnten während der Arbeit. Die Neubewertung und Priorisierung der Anforderungen mitten im Projekt konnten sehr zielführend und speditiv erarbeitet werden und man hat Rücksicht auf unsere Anliegen genommen.

7.3 Fazit

Die Zusammenarbeit sowohl im Projektteam als auch mit unserem Betreuer und Industriepartner hat durchwegs reibungslos funktioniert. Zudem hat die Arbeitsteilung innerhalb des Teams sehr gut funktioniert. Wir hatten stets ein gutes Arbeitsverhältnis zueinander und waren zu keinem Zeitpunkt mit Problemen oder Unstimmigkeiten innerhalb des Teams konfrontiert. Dies alles war der Schlüssel zu einem erfolgreichen Abschluss des Projekts.

Mit dem Technologiestack, welcher bereits vom Auftraggeber definiert und/oder eingeschränkt war, kannte sich niemand im Projektteam aus. Dieses nicht vorhandene Wissen wirkte sich sehr stark auf unser Projektfortschritt aus. Viele Technologien wurden im Projekt zum allerersten Mal verwendet, wie beispielsweise Flutter.

Die Arbeit stellte uns vor einige Schwierigkeiten und zeigte uns auf, wie wichtig es ist, vorab die Anforderungen klar abzusprechen, um die Aufwände sinnvoll schätzen zu können.
Unsere bereits vorgegebenen Anforderungen waren teils nicht ins Detail ausformuliert und liessen somit Interpretationsspielraum offen. Dadurch hatten wir am Anfang des Projekts nicht die gleiche Auffasung davon, was genau zu erledigen ist wie unser Kunde.

Die Neubewertung und Priorisierung der Anforderungen im Laufe des Projekts konnten dann diese Unklarheiten beseitigen.

Grundsätzlich denken wir, dass wir das Projekt trotz schwierigen Bedingungen erfolgreich meistern konnten. Wir konnten leider nicht alle Anforderung umsetzen, haben aber dank agiler Entwicklungsmethoden eine funktionierende App geliefert, welche vor allem im Hinblick auf die Weiterentwicklung sehr gut gerüstet ist.

Rückblickend hätten wir mehr Zeit in das saubere Ausarbeiten der Anforderungen stecken müssen. Somit hätten die Aufwände wahrscheinlich besser abgeschätzt werden können.
Zudem würden wir uns bei einem nächsten Projekt bei komplett unbekannten Technologien entweder am Anfang des Projekts mehr mit der Technologie auseinandersetzen oder besser komplett darauf verzichten. Das Problem ist ja schlussendlich in jedem Fall der zeitliche Faktor. Also auch wenn man sich in der Inception Phase mehr den unbekannten Technologien widmet, müssten dann einige Anforderungen gestrichen oder angepasst werden, um den zeitlichen Rahmen nicht zu sprengen.

A. Verzeichnisse

Glossar

Begriff	Beschreibung
CronJob	Ein CronJob ist eine Aufgabe die vom System nach einer gewissen Zeit abgearbeitet wird. Wobei diese Zeit individuell definiert werden kann.
DigitalOcean	DigitalOcean ist ein Anbieter von Cloud Diensten.
Docker	Docker ist eine freie Software zur Isolierung von Anwendungen mit Hilfe von Containervirtualisierung.
Dockercompose	Dockercompose ist ein Tool um Multi-Container Applikationen zu erstellen und betreiben.
Droplet	Die virtuellen Server auf DigitalOcean werden als Droplet bezeichnet.
DTO	Ein Data Transfer Object (DTO) wird verwendet, um auszutauschende Daten abzuspeichern.
Postman	Postman ist eine Software um APIs testen zu können.
Reverse Proxy	Ein Reverse-Proxy ist ein Proxy in einem Rechnernetz, der Ressourcen für einen externen Client von einem oder mehreren internen Servern holt.
UI	User Inferface (dt.: Grafische Benutzeroberfläche)
Wrappers	Als Wrapper wird ein Stück Software bezeichnet, welches ein anderes Stück Software umgibt.

Abbildungsverzeichnis

1.1 Verwendete Technologien 2
1.2 Screenshots der Contract Manager App 3

3.1 Domain Model 10
3.2 System Context Diagramm 11
3.3 Container Diagramm 12
3.4 Komponenten Diagramm - Mobile App 13
3.5 Komponenten Diagramm - API 14
3.6 Sequenzdiagramm - Registrierung 18
3.7 Sequenzdiagramm - Login 19
3.8 Sequenzdiagramm - Authenticate 20
3.9 Sequenzdiagramm - Passwortreset 21
3.10 Sequenzdiagramm - Dashboard Kostentabelle 22
3.11 Sequenzdiagramm - Dashboard Kostendiagramm 23
3.12 Sequenzdiagramm - Vertrag hinzufügen 24
3.13 Sequenzdiagramm - Continuous Deployment 27
3.14 Wireframe - Login und Registrierung 28
3.15 Wireframe - Dashboard 29
3.16 Wireframe - Mein Konto 29
3.17 Wireframe - Verträge 30
3.18 Wireframe - Vertrag hinzufügen 31
3.19 Wireframe - Passwortreset 32

4.1 Postman API-Tests 34
4.2 Postman Stress-Test 35
4.3 DigitalOcean Ressourcen Verbrauch während des Stress-Tests 36
4.4 Registrierung 42
4.5 Login 43
4.6 Account 44
4.7 Vertrag hinzufügen 45
4.8 Übersicht der Verträge 46
4.9 Details eines Vertrages 47
4.10 Dashboard - Übersicht der Kosten dieses Monates 48
4.11 Benachrichtungs-Screen 49
4.12 Passwort-Reset-Webseite 50

6.1 Zeitauswertung nach Projektmitglied 64
6.2 Zeitauswertung nach Kategorie 65
6.3 Soll-Ist-Vergleich 66

Tabellenverzeichnis

2.1 Funktionale Anforderungen 4
2.2 Nicht-Funktionale Anforderung 1 5
2.3 Nicht-Funktionale Anforderung 2 5
2.4 Nicht-Funktionale Anforderung 3 5
2.5 Nicht-Funktionale Anforderung 4 6
2.6 Nicht-Funktionale Anforderung 5 6
2.7 Nicht-Funktionale Anforderung 6 6
2.8 Nicht-Funktionale Anforderung 7 7
2.9 Nicht-Funktionale Anforderung 8 7
2.10 Nicht-Funktionale Anforderung 9 7
2.11 Nicht-Funktionale Anforderung 10 7
2.12 Nicht-Funktionale Anforderung 11 8
2.13 Nicht-Funktionale Anforderung 12 8
2.14 Nicht-Funktionale Anforderung 13 8
2.15 Nicht-Funktionale Anforderung 14 8
2.16 Nicht-Funktionale Anforderung 15 9

3.1 Infrastruktur Systeme 26

4.1 Projekt-Datenstruktur 33
4.2 Backend - Reverse Proxy 37
4.3 Backend - Controller 39
4.4 Backend - Middleware 39
4.5 Backend - Utils 40

5.1 Optionale Anforderungen 52

6.1 Organsisationstruktur 53
6.2 Rollen im Projektteam 54
6.3 Zeitdauer 54
6.4 Zeitbudget 54
6.5 Zeitplanung 55
6.6 Sprint 0 56
6.7 Sprint 1 56
6.8 Sprint 2 57
6.9 Sprint 3 57
6.10 Sprint 4 57
6.11 Sprint 5 57
6.12 Sprint 6 58
6.13 Sprint 7 58
6.14 Sprint 8 58
6.15 Sprint 9 58
6.16 Sprint 10 59
6.17 Sprint 11 59
6.18 Sprint 12 59
6.19 Sprint 13 59

6.20 Risiken 61
6.21 Risikomatrix 61
6.22 Vorgehen mit Risiken 62
6.23 Arbeitspakete 63
6.24 Implementierungsziele Funktionale Anforderungen 67
6.25 Implementierungsziele nicht-funktionale Anforderungen 67

B.1 Usability Test 1 vom 18.12.2022 78
B.2 Usability Test 2 vom 18.12.2022 80

Literatur

[1] OpenJS Foundation. Mocha - simple, flexible, fun. https://mochajs.org/, 2022. [Online; accessed 22.12.2022].

[2] OpenJS Foundation. NodeJS documentation. https://nodejs.org/en/docs/, 2022. [Online; accessed 22.12.2022].

[3] Stack Holdings GmbH. Caddy documentation. https://caddyserver.com/docs/, 2022. [Online; accessed 22.12.2022].

[4] Docker Inc. Develop faster. Run anywhere. https://www.docker.com/, 2022. [Online; accessed 22.12.2022].

[5] Meta Platforms Inc. React Native - Learn once, write anywhere. https://reactnative.dev/, 2022. [Online; accessed 22.12.2022].

[6] MongoDB Inc. Welcome to the MongoDB Documentation. https://www.mongodb.com/docs/, 2022. [Online; accessed 22.12.2022].

[7] Ionic. The mobile SDK fot the Web. https://ionicframework.com/, 2022. [Online; accessed 22.12.2022].

[8] ISO25000. ISO/IEC 25010. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, 2022. [Online; accessed 22.12.2022].

[9] Chai Assertion Library. Chai Assertion Library. https://www.chaijs.com/, 2022. [Online; accessed 22.12.2022].

[10] DigitalOcean LLC. The simple cloud that drives business growth. https://www.digitalocean.com/, 2022. [Online; accessed 22.12.2022].

[11] Google LLC. Firebase. https://firebase.google.com/, 2022. [Online; accessed 22.12.2022].

[12] Google LLC. Flutter documentation. https://docs.flutter.dev/, 2022. [Online; accessed 22.12.2022].

[13] Google LLC. Introducing package:flutter_lints. https://docs.flutter.dev/release/breaking-changes/flutter-lints-package/, 2022. [Online; accessed 20.12.2022].

[14] Mongoose. mongoose - elegant mongodb object modeling for node.js. https://mongoosejs.com/, 2022. [Online; accessed 22.12.2022].

[15] Twilio SendGrid. SendGrid. https://sendgrid.com/, 2022. [Online; accessed 22.12.2022].

[16] Statista. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/, 2022. [Online; accessed 22.12.2022].

[17] Basarat Ali Syed. StyleGuide. https://basarat.gitbook.io/typescript/styleguide/, 2022. [Online; accessed 20.12.2022].

B. Anhang

Usability Test

Nr.	Aufgabe	Bewertung (1-10)	Bemerkung
1.	Einen neuen Benutzer registrieren (mit gültiger E-Mail-Adresse).	9	Auswahl Geburtsdatum mühsam, da man für Monatswahl zurückblättern muss
2.	Mit erstelltem Benutzer anmelden. *Wenn dies nicht möglich war, kann mit folgendem User eingeloggt werden:* *E-Mail:* *jane.doe@vincenz.me* *Passwort: Password123*	10	hat gut geklappt
3.	Neue Person im Konto hinzufügen (z.B. ein Familienmitglied)	8	Gewichtung des Button "Neues Mitglied erstellen" erhöhen
4.	Einen Vertrag hinzufügen und einer Person zuweisen (sich selbst oder der gerade eben erstellten Person) *Hinweis: 1. Wenn "Gültig bis" leer gelassen wird, handelt es sich um einen automatisch verlängernden Vertag. (z.B. Spotify-Abo) 2. Bei der Zahlungsperiode wird eine Zahl erwartet und ob (Tag, Monat, Jahr). Z.B. Jeden zweiten Monat, wäre dann 2 und Monat(e).*	6	- Label Bei Vertragstyp und Vertragskunde fehlt + Hinweis Required fehlt - Name besser als Vertragsname benennen - Komma bei Kosten rundet auf ganze Zahlen ab, soll nicht angezeigt werden auf Tastatur
5.	Der gerade eben erstellte Vertrag unter Verträge ansehen und die Angaben prüfen.	8	Label oberhalb des Eintrages
6.	Unter "Konto" sich mit dem angemeldeten User abmelden.	10	

Nr.	Aufgabe	Bewertung (1-10)	Bemerkung
7.	Das Passwort des erstellen Users zurücksetzen und sich anschliessend wieder anmelden.	7	- Token-Feld verwirrt, am besten weglassen - Website ist nicht responsive
8.	Erneut anmelden, um sich zu vergewissern, dass das Passwort erfolgriech zurückgesetzt wurde.	10	
Allgemeines			- Bei Vertrag kommt in Header nach Vertragserstellung Pfeil-zurück, welche auf alte Version verweist 4.2.4 - Schriftgrösse etwas klein - Total pro Kategorie in Dashboard nicht in Klammer, da dies als Randinformation gilt. Auffälligere Darstellung - Gesamttotal fehlt - Konto: Gewichtung der Buttons, Mitglied erstellen ist am wichtigsten, erst dann Abmelden und Konto löschen
Total		**8.5**	

Tabelle B.1: Usability Test 1 vom 18.12.2022

Nr.	Aufgabe	Bewertung (1-10)	Bemerkung
1.	Einen neuen Benutzer registrieren (mit gültiger E-Mail-Adresse).	8	Anfangsbuchstaben nicht automatisch Grossbuchstaben. Datum nur per DateTime Picker erfassbar. Google Passwort Manager nicht nutzbar. Nach Registrierung keine Benutzerführung.
2.	Mit erstelltem Benutzer anmelden. *Wenn dies nicht möglich war, kann mit folgendem User eingeloggt werden:* *E-Mail:* *jane.doe@vincenz.me* *Passwort: Password123*	7	Nach Login wird man mit 'back' ausgeloggt. 4.2.4
3.	Neue Person im Konto hinzufügen (z.B. ein Familienmitglied)	9	Sehr versteckter Button/Link
4.	Einen Vertrag hinzufügen und einer Person zuweisen (sich selbst oder der gerade eben erstellten Person) *Hinweis: 1. Wenn "Gültig bis" leer gelassen wird, handelt es sich um einen automatisch verlängernden Vertag. (z.B. Spotify-Abo) 2. Bei der Zahlungsperiode wird eine Zahl erwartet und ob (Tag, Monat, Jahr). Z.B. Jeden zweiten Monat, wäre dann 2 und Monat(e).*	6	Gültigkeit bis automatisch ausfüllen. Select-Boxen Beschreibung ist nicht mehr ersichtlich, wenn etwas ausgewählt wird. Fehlermeldung von Felder verschwindet nicht nach Korrektur. Fehlermeldung bei Select-Boxen wird nicht angezeigt. Währung fehlt bei Betrag. Zahlungsperiode-Logik ist nicht verständlich und zu kompliziert.
5.	Der gerade eben erstellte Vertrag unter Verträge ansehen und die Angaben prüfen.	10	
6.	Unter "Konto" sich mit dem angemeldeten User abmelden.	10	

Nr.	Aufgabe	Bewertung (1-10)	Bemerkung
7.	Das Passwort des erstellen Users zurücksetzen und sich anschliessend wieder anmelden.	7	Token beim 1. Mal nicht gültig. 4.2.4 UI von Passwortreset-Website wird im Darkmode dargestellt, App nicht.
8.	Erneut anmelden, um sich zu vergewissern, dass das Passwort erfolgriech zurückgesetzt wurde.	10	
Total		**8.375**	

Tabelle B.2: Usability Test 2 vom 18.12.2022

OST
Ostschweizer
Fachhochschule

Aufgabenstellung Semesterarbeit „Contract Manager App“

Autor: **Frank Koch**

Version: **1.0**

Anpasst am: **17.08.2022**

1. Beteiligte Personen

- Studierende: André Blöchlinger, Simon Canal und Linard Vincenz
- Industriepartner: AdaptIT GmbH, Michael Güntensperger
- Betreuer: Frank Koch

2. Problembeschrieb

Immer mehr Verträge werden digital abgeschlossen und somit wird eine zentrale Verwaltung/Aufbewahrung erschwert. Gleichzeitig werden immer mehr Dienste wie z.B. Netflix und co. als Subscription angeboten und direkt von der Kreditkarte oder Konto abgebucht. Zahlungen werden dabei leicht übersehen oder Kündigungstermine verpasst.

3. Aufgabenstellung

Der Contract Manager soll ein zentraler Speicherort für Verträge und Subscriptions sein und mit Push Notifications auf anstehende Kündigungstermine aufmerksam machen. Zudem werden die monatlichen Kosten kategorisiert und auf einem Dashboard grafisch dargestellt. Verträge werden zu Beginn manuell erfasst.

Technische Umgebung

Für die Umsetzung wird mit Web-Technologien gearbeitet.

- Mobile-App (in React Native, Flutter oder ionics)
- Backend (Node.js)
- DB (MongoDB)
- Document Storage (DigitalOcean oder S3)

Funktionale Anforderungen

- Die App soll bei Google Play als interner Test und in Apple Testflight eingerichtet werden.
- Benutzer können sich registrieren, einloggen und den Account löschen
- Verträge inklusive Dokument / Fotografierter Vertrag, Ablaufdatum, Kategorie (z.B. Versicherung, Freizeit,..) und Beschreibung speichern
- Die Verträge werden in einem Dashboard dargestellt
- Benutzer wird über Ablauf von Verträgen informiert
- Verträge den Familienmitgliedern zuweisen können (haben keinen separaten Account)
- Passwort vergessen-Funktionalität
- Einfache Auswertungsübersicht z.B. Google Analytics für Admins

Optionale Anforderungen

- Accounts teilen können (z.B. Ehepaar)
- Template Abos zur Auswahl anbieten (z.B. Netflix, PrimeVideo, Spotify, …)
- Verträge aus dem App als PDF scannen
- Externe haben die Möglichkeit den Benutzern anhand deren Mail-Adresse und Geburtsdatum, Verträge hochzuladen. Z.B. kann eine Versicherung dem Kunden ein Vertrag direkt hochladen, so entfällt der Scan-Prozess
- Schnittstelle zu einer Bank / Zahlungsanbieter, um die Transaktionsdaten abzugleichen und Standard-Abos automatisch zu erfassen (z.B. Netflix).
- Erstellen einer Website mit gleicher Funktion wie in der App
- Verschlüsselte Ablage der Kundenverträge (so dass nicht mal der Admin diese einsehen kann)
- Vergleichsportal wie Comparis anbinden, um z.B. Versicherungsoptimierungen vorschlagen zu können (Problem, dass Daten mit externen geteilt werden müssen)
- Upload von zusätzlichen Dokumenten zu einem Vertrag z.B. eine Schadensfallmeldung bei einer Versicherung, oder eine nachträgliche Vertragsanpassung

Nicht-Funktionale Anforderungen

- Das Entwicklerteam implementiert die Features gemäss der abgesprochenen Priorität mit dem Kunden
- Das Erfassen eines Vertrages soll maximal 30 Sek. dauern
- Das Backend sollte 500 Requests pro Minute handeln können
- Jede Seite sollte nicht länger als 150ms für das Laden benötigen
- Das App soll auf Android und iOS laufen
- Drei von vier Test-user sollten das UI (Kategorien: layout, responsiveness, colour, content) der Applikation mit iPhone / Android Handy mit einer Note von mindestens 8 von 10 bewerten, wobei 10 das Beste ist.
- Die Datenbank soll bis zu 10'000 Verträge und 500 Benutzer managen können.
- Errors sollen keine Systemfehler erzeugen, aber eine Error Nachricht Zeigen und das System auf den vorherigen Zustand zurücksetzen.
- Jeder Error soll im System geloggt werden
- Daten welche in Eingabefelder abgefüllt werden, sollen zuerst validiert werden, bevor diese durch das System verarbeitet werden. SQL Injection test der Eingabefelder sollte keine Verletzlichkeiten zeigen.
- User-Passwörter werden nicht in plain-text in der Datenbank gespeichert.
- Wenn sich ein User in die App einloggt, werden ihm auch nur seine Daten / auf Daten die er Zugriff haben soll, angezeigt.
- Businesslogik im Backend soll modular aufgebaut werden, so dass sie erweitert werden kann.
- Die Backend-API soll durch API-testing Tools getestet werden.
- Implementierte Funktionalität (Datenbank, Backend) sollen deployed und die App bei Google als interner Test und in Apple Testflight eingerichtet werden.

4. Zur Durchführung

Mit dem Betreuer finden Besprechungen gemäss Absprache statt. Die Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben.

5. Dokumentation und Abgabe

Siehe Leitfaden Abschnitt 5.5 "Umfang und Form der Abgabe".

6. Termine

Siehe veröffentlichte «Termine SA HS22».

7. Bewertung

Siehe veröffentlichter «Leitfaden BA SA» Abschnitt 6 "Bewertung", insbesondere 6.4.

Rapperswil, den 20.09.22

Frank Koch