Fachbereich: MNI – Mathematik, Naturwissenschaften und Informatik

Studiengang: Social Media Systems

Veranstaltung: CS2018 Entwicklung mobiler Applikationen (EMA)

Leitung: Sebastian Süß

Semester: Sommersemester 2022

Softwaredokumentation zum Projekt

Abgegeben von:

Name: Ngoc Minh Thu Nguyen
E-Mail: ngoc.minh.thu.nguyen@mni.thm.de
Matrikel-Nr.: 5311972

Name: Johanna Heiler
E-Mail: johanna.heiler@mni.thm.de
Matrikel-Nr.: 5305784

Name: Nico Schneider
E-Mail: nico.schneider-2@mni.thm.de
Matrikel-Nr.: 5245802

Name: Julia Leimbach
E-Mail julia.leimbach@mni.thm.de
Matrikel-Nr.: 5284973

Datum der Abgabe: 29.07.2022

Inhalt

1. Einleitung

1.1 Fachlicher Hintergrund und Zielsetzung

Ziel dieser Applikation ist das Erfassen von Ausgaben/Einnahmen von Personengruppen sowie das Verteilen dieser auf alle Gruppenmitglieder. Einsatzszenarien sind beispielsweise Urlaubs- und Festival Gruppenreisen, Kneipentouren oder WGs.

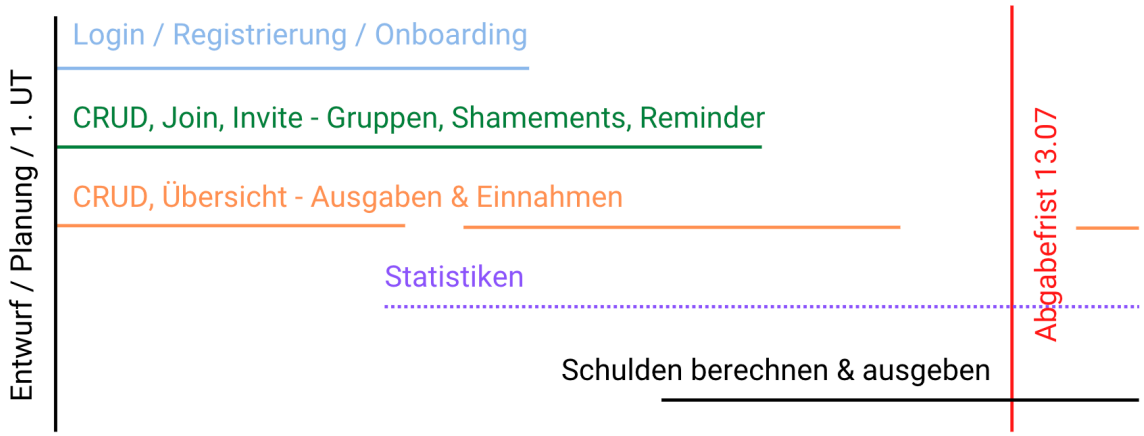
1.2 Anforderungen (zusammengefasst)

Die funktionalen Anforderungen bestehen darin eine Gruppenverwaltung nach CRUD bereitzustellen. Werden von Gruppenmitgliedern neue Ausgaben oder Einnahmen erstellt, soll es möglich sein Belege wie Rechnungen, Kassen/Pfandbons als Bild hinzuzufügen. Weiterhin können Ausgaben/Einnahmen wie Wasser-/Stromrechnungen periodisch wiederkehrend sein. Außerdem gibt es eine Übersicht von Ausgaben und Einnahmen in Form von Statistiken. Weiter werden Zahlungserinnerungen per E-Mail oder Messenger-Dienste verschickt. Shamements dienen zur Steigerung der Motivation der Benutzer*innen. Alle Daten der Benutzer*innen und Gruppen werden auf einem Backend gesichert und können wieder geladen werden.

1.3 Organisation und Vorgehensmodell

Zu Beginn der Entwicklung, wurden Wireframes und auf diesen basierende Mockups erstellt. Abschließend ist der Mockup zu einem klickbaren Prototypen in Figma modifiziert worden. Ziel war es eine Design-Vorlage für die spätere Programmierung der App zu schaffen und Standards für UI-Elemente festzulegen. Figma bietet ebenfalls die Möglichkeit, die Usability zu überprüfen. Dies wurde anhand eines ersten Usability-Test genutzt. Der Test wird an einer späteren Stelle der Dokumentation beschrieben.

Des Weiteren wurden Meilensteine definiert und der zeitliche Aufwand eingeschätzt. Dieser wurde ausschließlich anhand der Umsetzung des MVP gemessen. Vor Beginn der Programmierung sind die Aufgaben in einem Issue-Board auf YouTrack formuliert worden. Das Board dient als Übersicht des Projektfortschritts. Zur weiteren Absprache eigneten sich wöchentliche Meetings, die zum Ende der Projektphase häufiger wurden. Weiterhin diente eine Whatsapp Gruppe der Kommunikation.



2. Anforderungen

2.1 Funktionale Anforderungen

Die folgenden Anforderungen ergeben sich aus der User-Story-Map, die im Vorfeld erstellt wurde. Sie beginnt damit, dass der*die Nutzer*in sich zunächst einen Überblick über die App verschaffen möchte. Genauer wünscht er sich eine Einführung in die App, um die Funktionen kennenzulernen. Weiterhin soll die App intuitiv zu bedienen sein.

Das nächste Epic befasst sich mit dem Registrieren/ Login. Nutzer*innen möchten sich registrieren, um die App nutzen zu können. Ebenso möchten sie sich einloggen, um auf gespeicherte Daten zugreifen zu können. Ferner wünscht sich der*die Nutzer*in Social-Logins. Wesentlich ist auch die Möglichkeit seinen Account verwalten zu können.

Im nächsten Schritt landet der*die Nutzer*in auf der Gruppen Übersichtsseite. Hier besteht das Epic darin, dass der*die Nutzer*in sich wünscht die Gruppe verwalten zu können. Zunächst möchte er*sie eine Gruppe erstellen. Die nächste Anforderung besteht darin, weitere Nutzer*innen einer Gruppe hinzufügen zu können bzw. eine Einladung zu der Gruppe zu verschicken. Alle Teilnehmer in einer Gruppen sollen verwaltet werden können. Ebenso soll es möglich sein, an mehreren Gruppen teilzunehmen. Hierfür wird eine Übersichtsliste gebraucht.

Im Folgenden möchte der User Ausgaben verwalten können. Ziel ist es Ausgaben auf den neuesten Stand zu bringen. Hier ist seine Priorität darin Belege in Form von Bildern als Nachweise den Ausgaben hinzufügen zu können. Außerdem möchte er kenntlich machen, welche Ausgaben wiederkehrend sind, um seinen Aufwand zu minimieren. Ergänzend sollen Statistiken über die Ausgaben und Einnahmen einen Überblick geben. Ferner sollen die Details einer einzelnen Ausgabe oder Einnahme einsehbar sein. Die Darstellung der Aus- und Einnahmen in Form eines Diagramms ist ein Feature über den MVP hinaus.

Die Benutzer*innen sollen die Möglichkeit erhalten, auf eigenen Wunsch die Ausgaben und Einnahmen ihrer Gruppe verrechnen und aufteilen zu lassen. Die errechneten Beträge, welche das Gruppenmitglied zu bezahlen hat oder noch erhält soll anschließend pro Gruppe angezeigt werden. Dies soll einen Überblick ermöglichen. Ferner sollen ausschließlich Einnahmen und Ausgaben für Nutzer*innen sichtbar sein, welche noch nicht verrechnet wurden.

Das letzte Epic betrifft das Erinnern von ausstehenden Zahlungen. Nutzer*innen wollen Erinnerungen versenden und erhalten können, um auf ausstehende Zahlungen innerhalb der App aufmerksam gemacht zu werden. Das Erinnern an Zahlungen soll mit der Vergabe von Shamements einher gehen.

2.2 Technische Anforderungen

Entwickelt wird eine Progressive-Web-App für die Betriebssysteme IOS und Android. Weiterhin wird eine Datenbank gebraucht die eine persistente Datenspeicherung und Synchronisation bereitstellt.

2.3 Anwendungsfälle

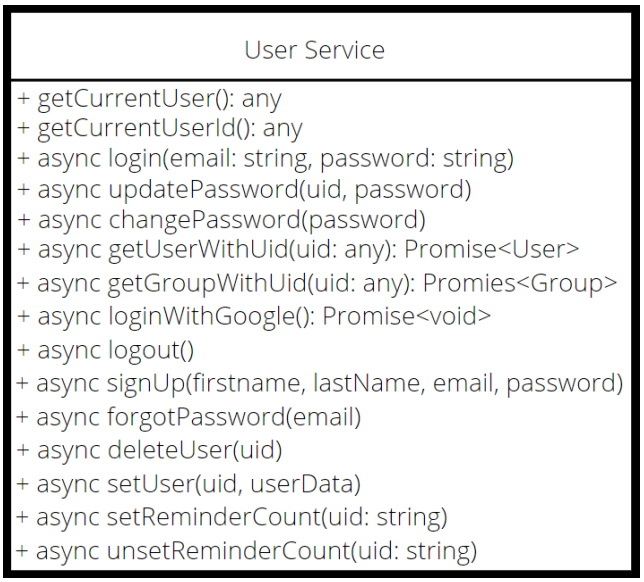
Die Anwendungsfälle ergeben sich aus den zuvor formulierten Anforderungen. Zunächst wurde beschrieben, dass der*die Nutzer*innen sich einen Überblick über die App wünscht. Das wird in Form eines Onboarding Prozesses umgesetzt. Ausgewählt wurde eine Benefit- bzw. Function-Oriented Onboarding Prozess. Dem*der Nutzer*innen werden kompakt die Features mit der höchsten Priorität vorgestellt. Dies passiert einmalig bei der ersten Nutzung der App. Danach wird der*die Nutzer*innen sich erfolgreich registrieren bzw. im späteren Verlauf einloggen. Dazu gibt es Eingabefelder für den Vor- und Nachnamen, eine E-Mail Adresse. Um die Authentifizierung des Nutzers zu gewährleisten und später zu prüfen, ist ein persönliches Passwort erforderlich. Es besteht auch die Möglichkeit einen Social-Login mit Google zu nutzen oder ein vergessenes Passwort zurücksetzen.

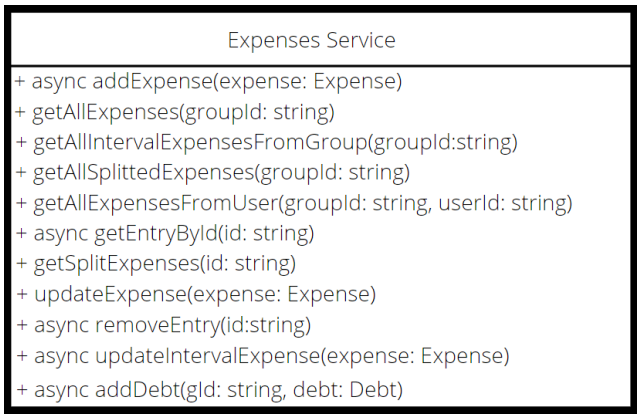
Bei Erfolg, gelangt der*die Nutzer*innen auf die Übersichtsseite der Gruppen. An dieser Stelle kann er*sie eine neue Gruppe erstellen und bekommt eine Übersicht seiner Gruppen in Form einer Liste. Über einen Button gelangt der*die Nutzer*innen auf die Seite eine neue Gruppe zu erstellen. Hier kann er einen Namen wählen. Dazu wird eine ID generiert. Mit diesen Attributen können weitere Mitglieder*innen der Gruppe werden. Alternativ ist es möglich über eine ID und Namen in eine weitere Gruppe einzutreten. Der*Die Benutzer*in kann die Gruppe verlassen oder andere Mitglieder entfernen.

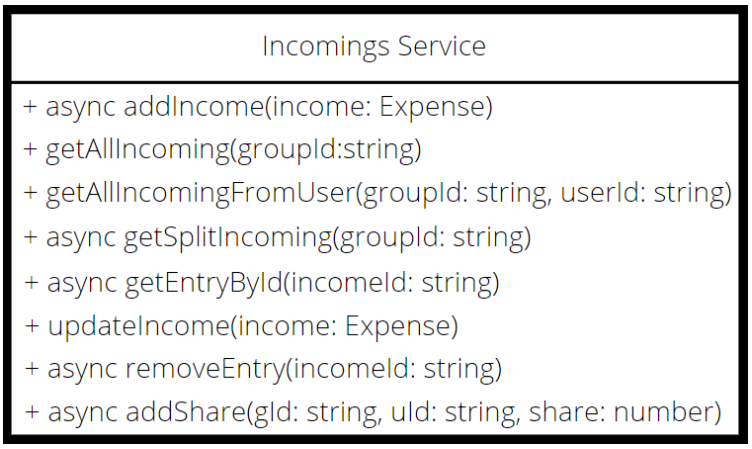
Im Anschluss gelangt der*die Nutzer*innen auf eine Detailseite einer Gruppe. Hier wird ihm die Bilanz der Gruppe angezeigt, sowie die Gruppenmitglieder. An dieser Stelle lässt sich die Gruppe verwalten, wie Mitglieder bearbeiten, die Gruppe verlassen und eine Gruppeneinladung versenden. Außerdem kann man seine Gruppenmitglieder auf ausstehende Zahlungen hinweisen. Von dort kommt der*die Nutzer*innen auf die Einnahmen und Ausgabenseite der App. Hier kann er neue Einnahmen und Ausgabe anlegen. Neben Eingabefeldern für Name, Betrag und dem Datum, kann eine Datei ausgewählt werden, um den Kassenbon zu hinterlegen. Weiterhin kann über ein Select kenntlich gemacht werden, dass die Ausgabe wiederkehrend ist. Diese werden in einer Liste angezeigt und können auch

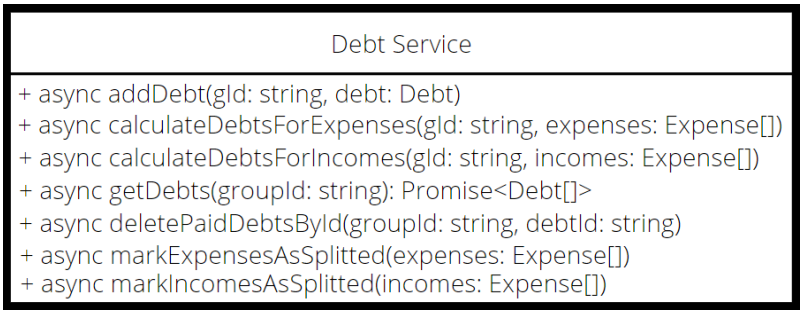
im Nachhinein bearbeitet werden. Auch kann sich der*die Nutzer*innen eine Statistik über seine Einnahmen und Ausgaben in Form eines Diagramms anzeigen lassen. Dieses kann nach den letzten Monaten gefiltert werden. Auf dieser Seite befindet sich auch die Hauptfunktion der App. Über einen Button werden die Ausgaben und Einnahmen unter den Gruppenmitgliedern aufgeteilt. Dabei werden die bisherigen Einzahlungen der Mitglieder berücksichtigt.

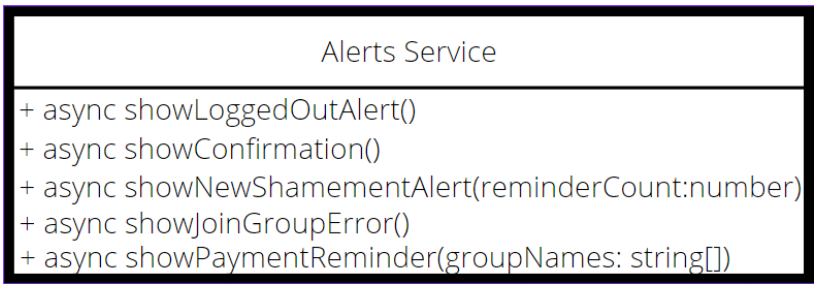
2.4 Fachliches Modell

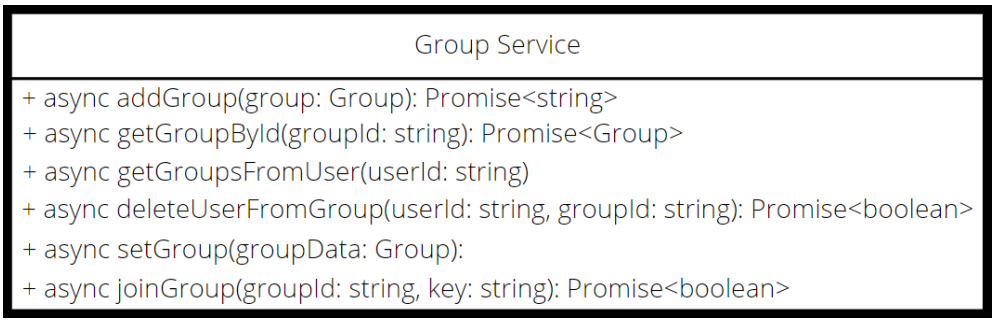


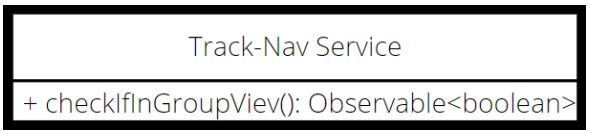





Debt Service
+ async addDebt(gId: string, debt: Debt)
+ async calculateDebtsForExpenses(gId: string, expenses: Expense[])
+ async calculateDebtsForIncomes(gId: string, incomes: Expense[])
+ async getDebts(groupId: string): Promise<Debt[]>
+ async deletePaidDebtsById(groupId: string, debtId: string)
+ async markExpensesAsSplitted(expenses: Expense[])
+ async markIncomesAsSplitted(incomes: Expense[])


Alerts Service
+ async showLoggedOutAlert()
+ async showConfirmation()
+ async showNewShamementAlert(reminderCount:number)
+ async showJoinGroupError()
+ async showPaymentReminder(groupNames: string[])


Group Service
+ async addGroup(group: Group): Promise<string>
+ async getGroupById(groupId: string): Promise<Group>
+ async getGroupsFromUser(userId: string)
+ async deleteUserFromGroup(userId: string, groupId: string): Promise<boolean>
+ async setGroup(groupData: Group):
+ async joinGroup(groupId: string, key: string): Promise<boolean>


Track-Nav Service
+ checkIfInGroupViev(): Observable<boolean>

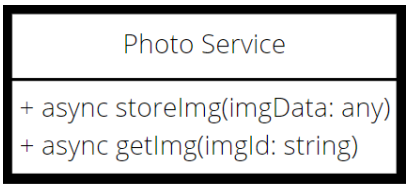
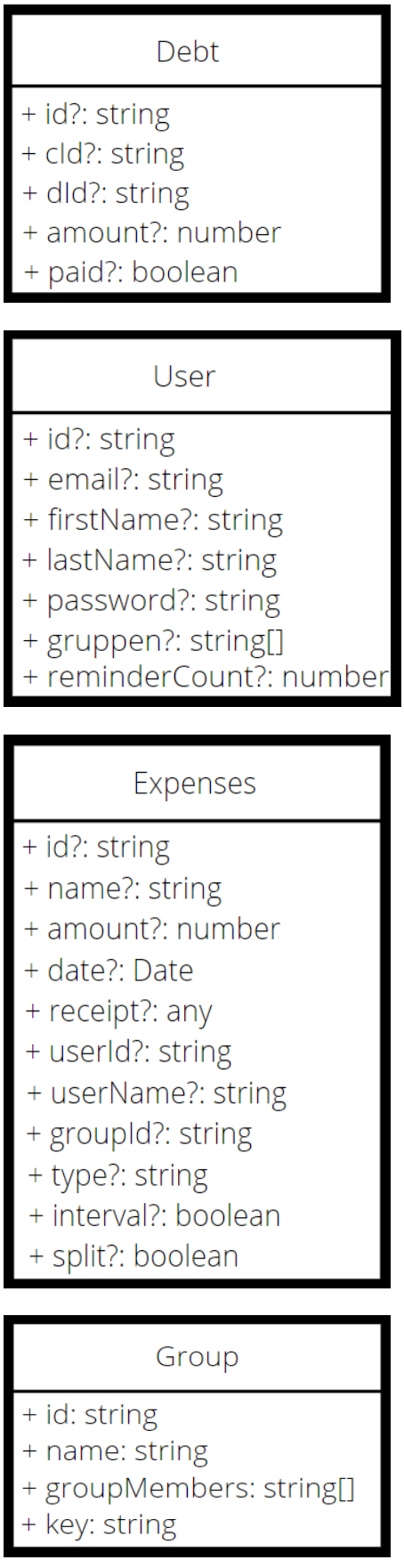
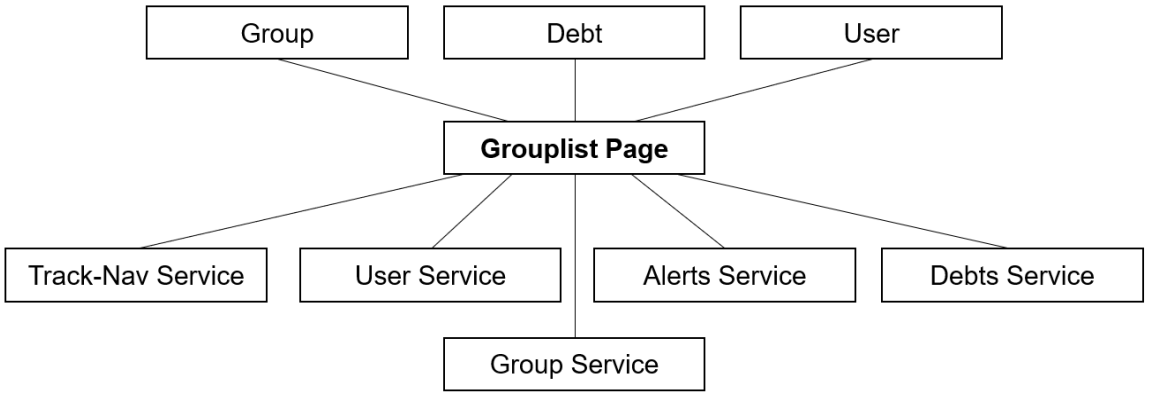
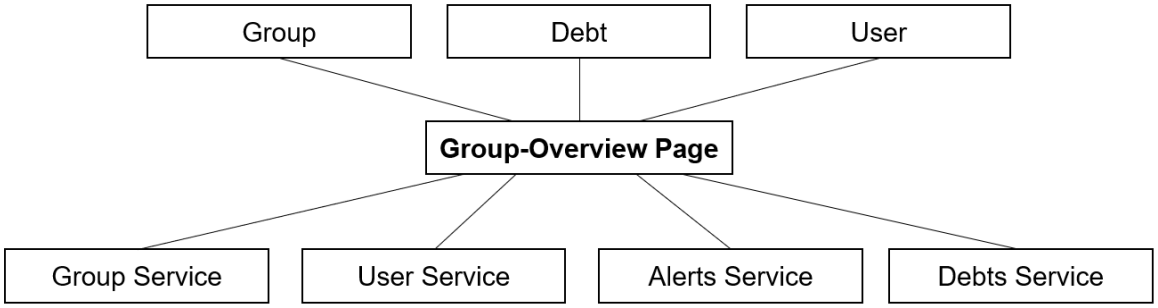
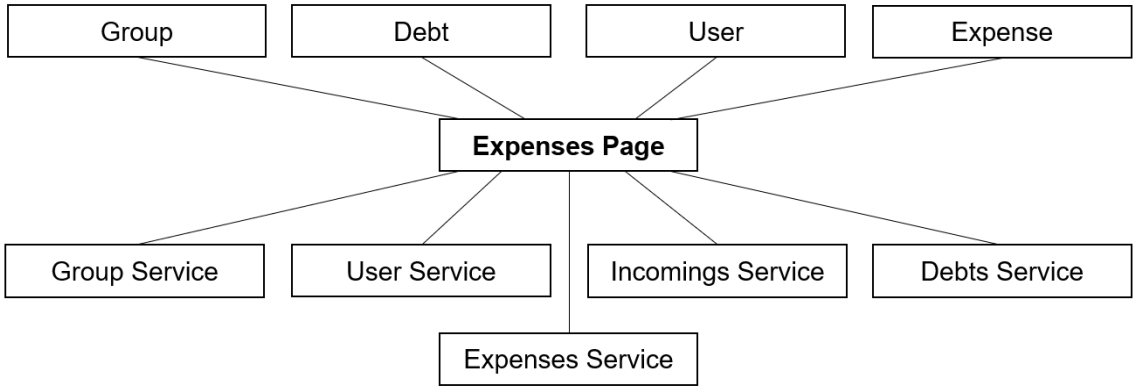


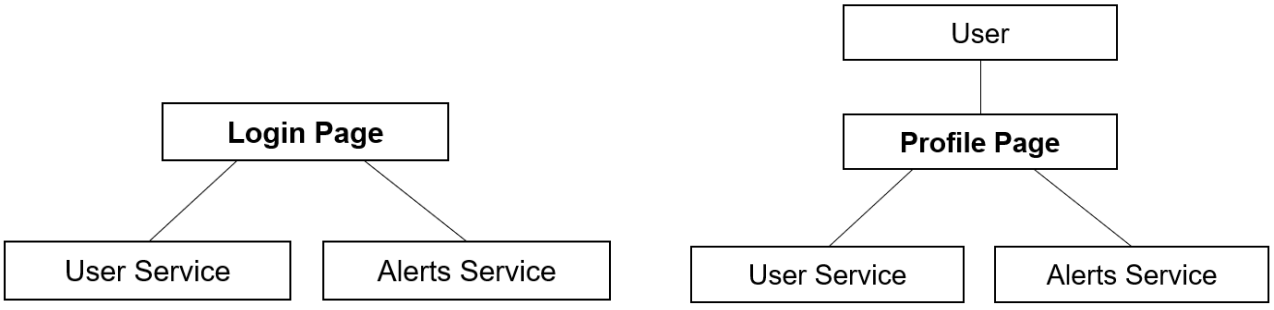
Photo Service
+ async storeImg(imgData: any)
+ async getImg(imgId: string)

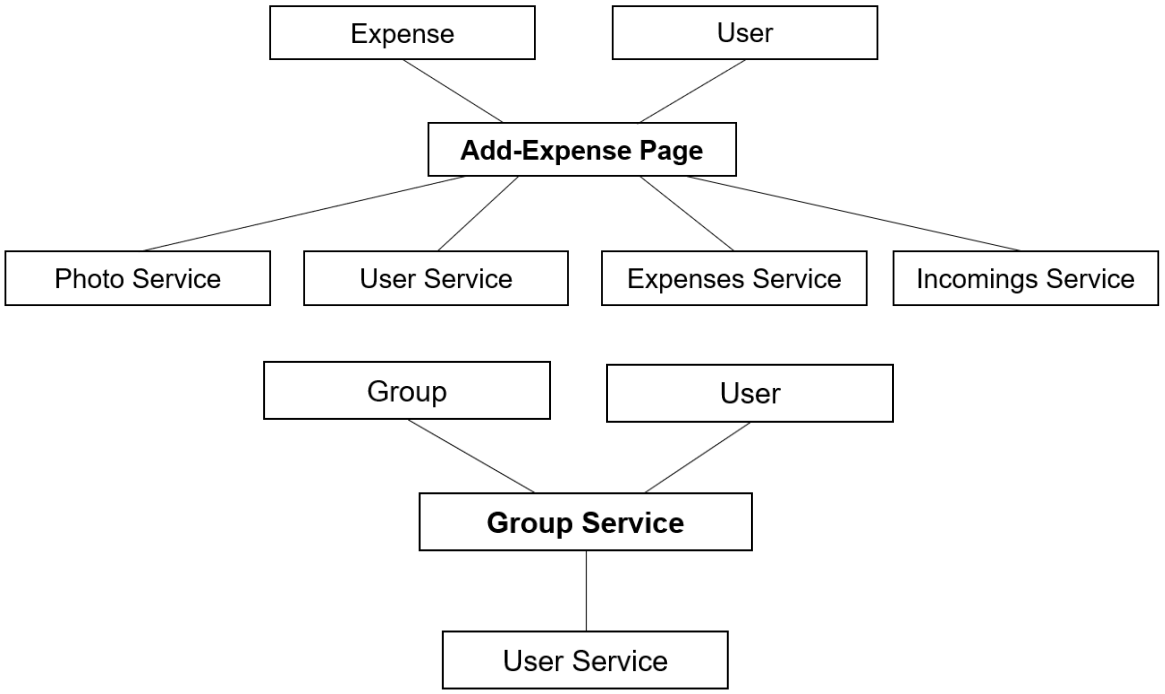

Debt
+ id?: string
+ cId?: string
+ dId?: string
+ amount?: number
+ paid?: boolean
User
+ id?: string
+ email?: string
+ firstName?: string
+ lastName?: string
+ password?: string
+ gruppen?: string[]
+ reminderCount?: number
Expenses
+ id?: string
+ name?: string
+ amount?: number
+ date?: Date
+ receipt?: any
+ userId?: string
+ userName?: string
+ groupId?: string
+ type?: string
+ interval?: boolean
+ split?: boolean
Group
+ id: string
+ name: string
+ groupMembers: string[]
+ key: string

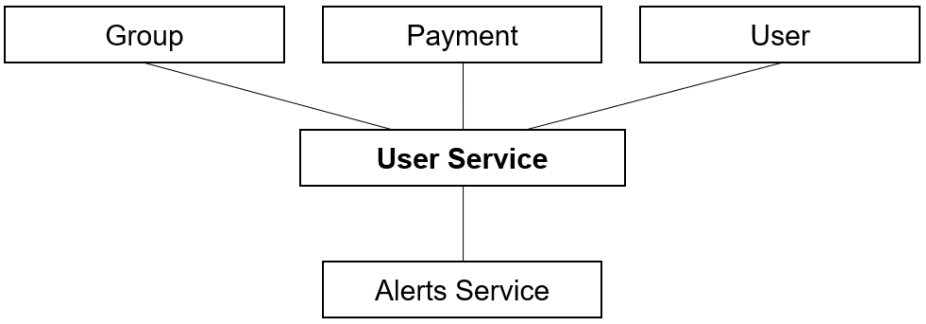

Group
Debt
User
Grouplist Page
Track-Nav Service
User Service
Alerts Service
Debts Service
Group Service

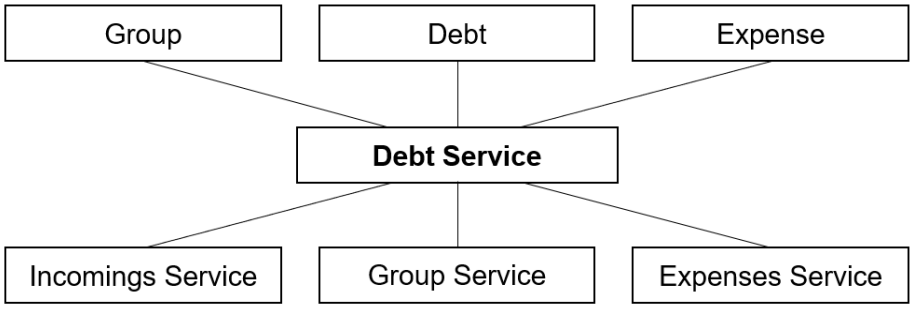

Group
Debt
User
Group-Overview Page
Group Service
User Service
Alerts Service
Debts Service

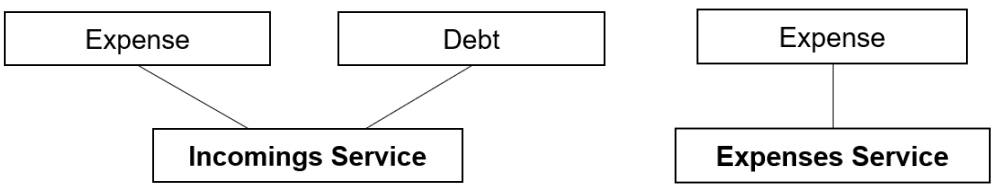

Group
Debt
User
Expense
Expenses Page
Group Service
User Service
Incomings Service
Debts Service
Expenses Service


Login Page
User Service
Alerts Service
User
Profile Page
User Service
Alerts Service


Expense
User
Add-Expense Page
Photo Service
User Service
Expenses Service
Incomings Service
Group
User
Group Service
User Service

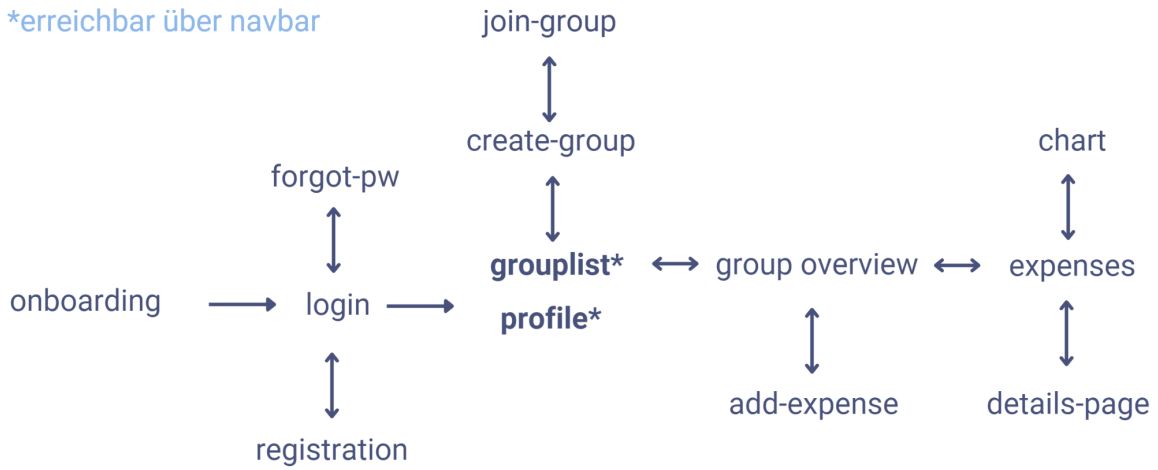

Group
Payment
User
User Service
Alerts Service


Group
Debt
Expense
Debt Service
Incomings Service
Group Service
Expenses Service


Expense
Debt
Expense
Incomings Service
Expenses Service

3. Entwurf

3.1 Views und Navigation

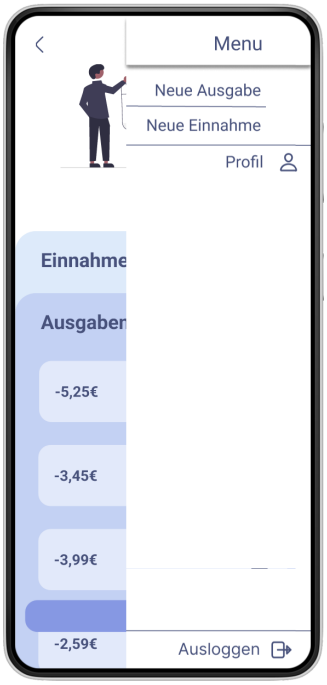


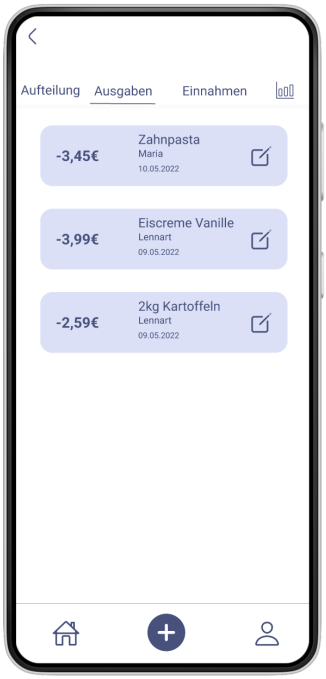
3.2 Usability Tests

Um eine intuitive Bedienung der Applikation und eine sinnvolle Usability für die Nutzer*innen zu gewährleisten, wurden während der Entwicklung zwei Usability Tests durchgeführt. Der erste Test wurde mit einem klickbaren Prototyp in Figma durchgeführt. Getestet wurden die Sinnhaftigkeit der Positionierung der ausgewählten UI-Elemente Funktionen sowie das Look-and-Feel der Applikation. Aufgrund der Ergebnisse dieses Tests wurde das Design nochmal angepasst.

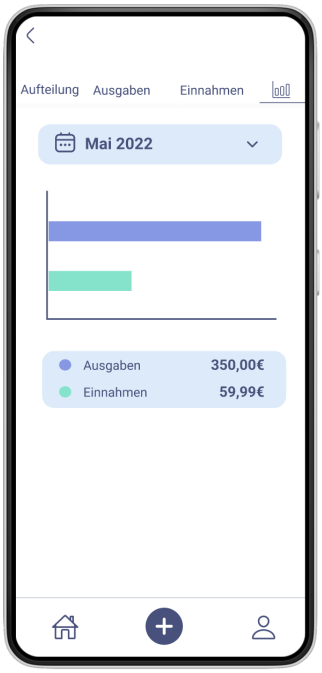
Die linke Seite zeigt das veraltete Design. Auf der rechten Seite sind die überarbeiteten Screens.

Das Menü wurde durch eine Navigationsleiste ersetzt. In diesem Menü befanden sich an verschiedenen Stellen in der App unterschiedliche Funktionen. Das widerspricht der Erwartungshaltung der Nutzer*innen, da dieses Menü im eigentlichen Sinn eine Art Anker für sie ist und grundsätzlich immer die gleichen Funktionen beinhalten sollte. Im User-Centered-Design dient dieses Menü der Navigation und hilft dem*der Benutzer*in auf Seiten innerhalb der App zurückzufinden.





Durch die eingefügte Navbar war es nicht mehr möglich die Cards für Einnahmen und Ausgaben hochziehen zu können, da der untere Teil der Seite überladen wirkte. Eingesetzt wurden Reiter für die einzelnen Pages. Dies führt zu einer besseren Übersicht der Seiten.



Außerdem wurden auf allen Seiten Back-Buttons eingesetzt, da diese den Testpersonen gefehlt haben. Außerdem wurden Illustrationen im oberen Bereich der Seite entfernt, da sie zu viel Platz auf der Seite einnahmen.

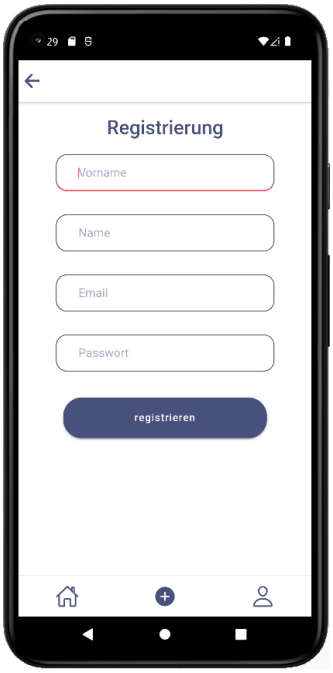
Für die Durchführung des zweiten Usability Tests wurde ein Fragenkatalog entwickelt. Dieser ist so aufgebaut, dass das Interview mit einer Vorbefragung startet. Hier wurde nach dem Allgemeinen Umgang mit Apps gefragt im Bezug auf die Thematik gefragt. Im nächsten Teil wurde nach dem Spontanen Eindruck der App gefragt. Der*die Nutzer*in sollte die Gestaltung der App beurteilen und was die App auf den ersten Blick anbietet. Im dritten Teil wurde ein Use Case getestet. Ziel ist, dass der*die Nutzer*in die gesamte Klickstrecke durchlaufen bis dahin eine erste Ausgabe anzulegen. Hierbei wird die Intuitivität und die Klickstrecke geprüft. Im letzten Teil sollte die App exploriert werden. Alle Seitenbereiche sollten getestet werden, sowie die Navigation zwischen den Seiten. Getestet wurde weiterhin, ob der*die Nutzer*in alle Funktionen fand.

Aus Zeitgründen konnten nur die wichtigsten Ergebnisse des Tests umgesetzt werden. Neben kleineren Änderungen an der Gestaltung der UI-Elemente ist der Test als positiv zu bewerten. Zum Beispiel die Einführung der Navigationsleiste war eine passende Entscheidung, denn sie wurde von allen Testpersonen korrekt verwendet. Die Funktionalitäten um den Use Case durchzuführen wurden ohne Schwierigkeiten gefunden. Funktional wurde noch ergänzt, dass auch bereits geteilte Ausgaben und Einnahmen noch in der Übersichtsleiste angezeigt werden. Trotzdem muss festgehalten werden, dass alle Testpersonen einen Hintergrund im Bereich der Informatik haben. Für noch aussagekräftigere Ergebnisse sollen weitere Personen befragt werden mit diversen demografischen Daten.

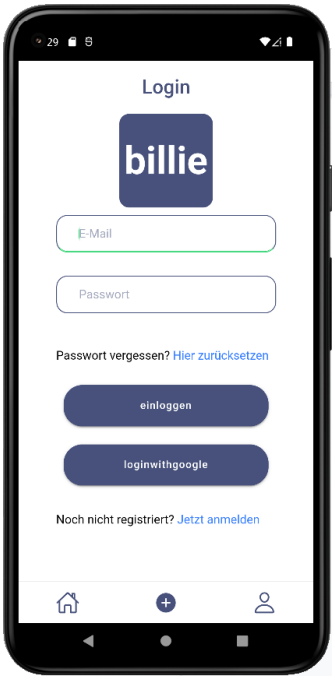
3.3 UI-Design

Die folgenden Screenshots zeigen Ausschnitte der entwickelten App. Als Basis dienten der Prototyp aus Figma und die Ergebnisse der Usability Tests.

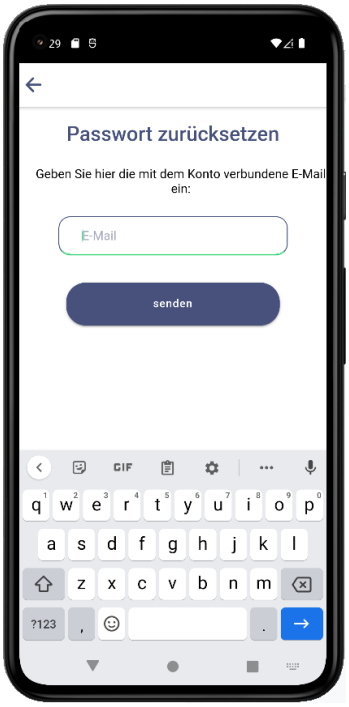
Beim ersten Öffnen der App erscheint ein Onboarding-Prozess. Dieser gibt dem*der Nutzer*in eine Einführung in die wichtigsten Funktionen der App



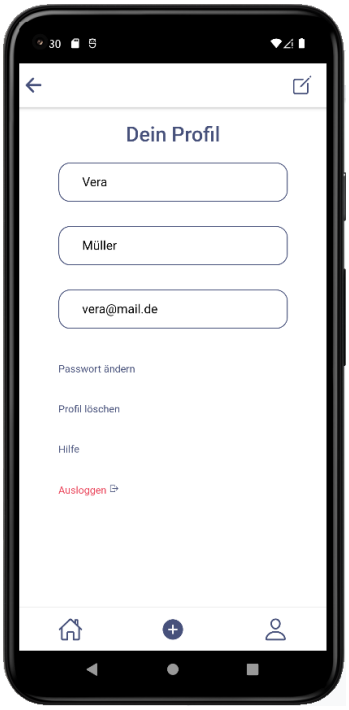
Anschließend wird der*die Nutzer*innen auf die Seite zur Registrierung/Login geführt. Bei Erfolg erhält er Zugang zu der Applikation.



Der*die Nutzer*innen kann sich mit den bei der Registrierung angegeben Daten anmelden.

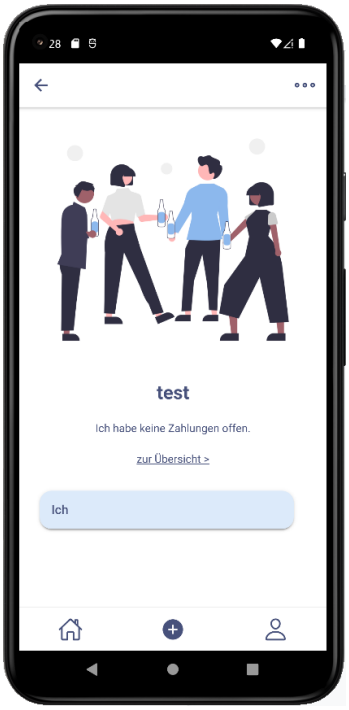


Hat der*die Nutzer*innen sein Passwort vergessen, kann er es zurücksetzen.



Die Profilseite ist nach dem Registrieren/Login über die Navbar erreichbar. Hier lassen sich die Benutzerdaten bearbeiten.

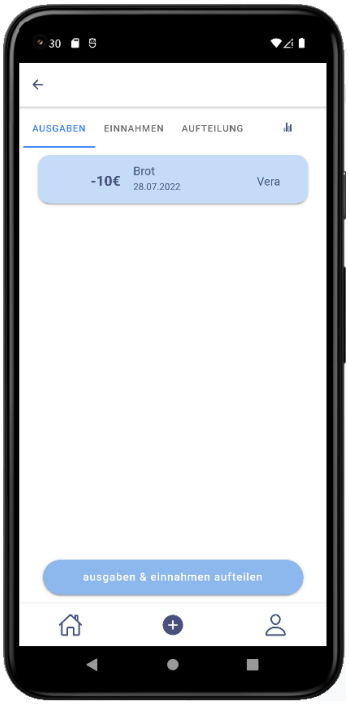
Nach erfolgreicher Anmeldung gelangt der*die Nutzer*in auf die Seite der Gruppenübersicht. Hier werden ihm eine Auflistung von Gruppen angezeigt, bei denen er*sie Mitglied ist.



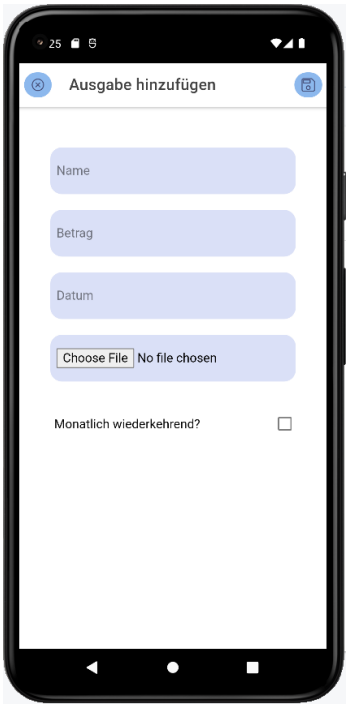
Mit Auswahl einer Gruppe kommt der*die Nutzer*in auf die Detailseite der Gruppe. Es werden weitere Details zur Gruppe angezeigt.

Der*die Nutzer*in kann einen Namen für die neue Gruppe wählen. Eine ID wird generiert.Ergänzend finden sich weiterführende Funktionen, um eine Gruppe zu erstellen oder einer bestehenden beizutreten.

Über eine Einladung kann der*die Nutzer*in einer bestehenden Gruppe beitreten.



Im nächsten Schritt gelangt der*die Anwender*in über die Details der Gruppe auf diese Seite. Hier wird ihm eine Auflistung der Einnahmen oder Ausgaben angezeigt.



Über einen Reiter werden dem*der Nutzer*in eine Auflistung der Ausgaben angezeigt. Ebenso lassen sich neue Ausgaben hinzufügen.

Hier kann sich der*die Nutzer*in eine Übersicht der Ausgaben und Einnahmen anzeigen lassen. Diese sind nach beliebig filterbar.

4. Systemarchitektur

4.1 Ausgewählte Technologien

Für die Realisierung der App wurden verschiedene Dienste von Googles Firebase verwendet. Hierbei handelt es sich um eine Anwendung, die auf einer externen Seite betrieben wird. Für diesen Anwendungsfall wurde die E-Mail- und passwortbasierte Authentifizierung und der Social-Login über Google integriert. Mit Firebase können die Benutzerdaten sicher gespeichert werden. Diese Art der Registrierung/ des Logins ist für den*die Nutzer*in komfortabel und die dazugehörigen Daten lassen sich entsprechend ablegen und auslesen. Für die Datenverwaltung wurde der Firestore verwendet. Es handelt sich um eine NoSQL-Cloud-Datenbank, bei der die Daten in Echtzeit gespeichert und synchronisiert werden.

Billie wurde mit Hilfe des Anwendungsdesign-Frameworks und Entwicklungsplattform Angular entwickelt. Angular unterstützt die Entwicklung unter Typescript, womit die App programmiert wurde.

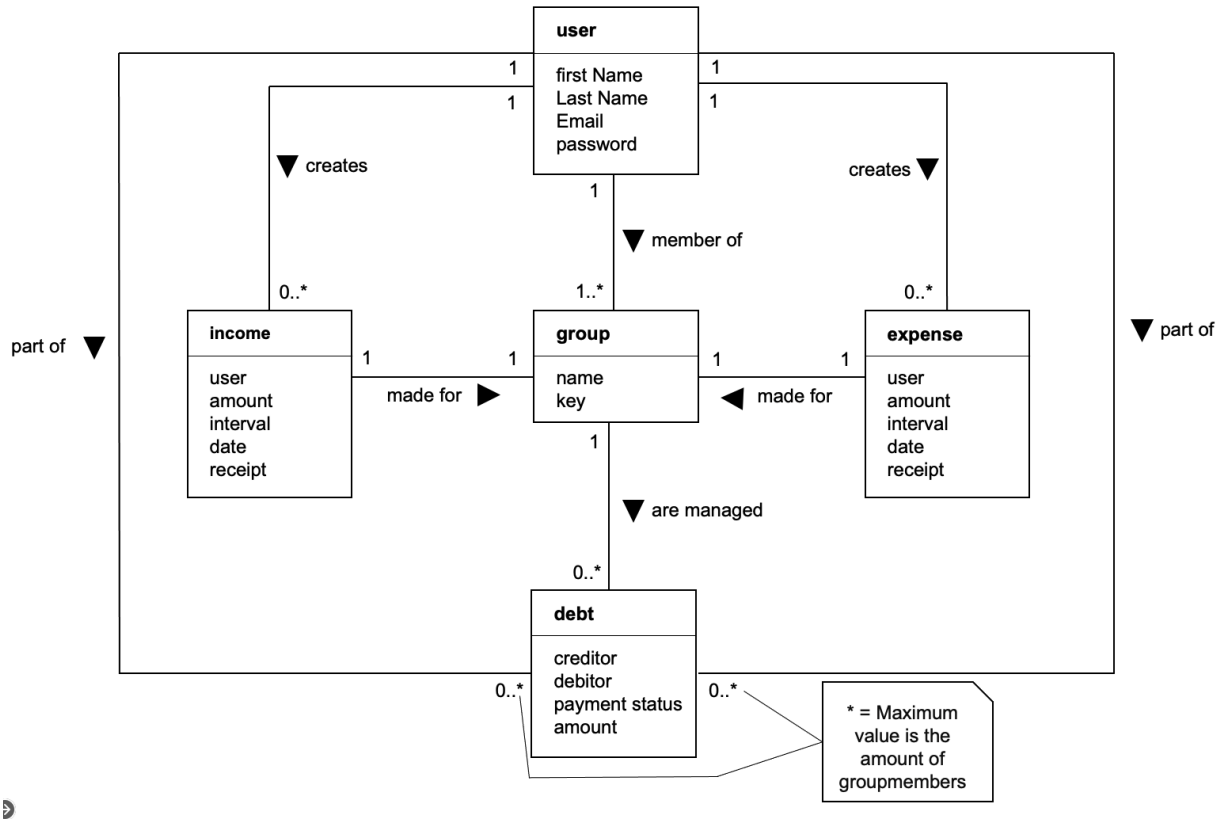
Durch das Frontend-Framework Ionic wurde die grafische Realisierung der mobilen App umgesetzt. Dabei helfen die vordefinierten UI-Komponenten bei der Entwicklung und sorgen für eine gute Usability für die Nutzer*innen. Capacitor wurde in das Projekt integriert, um billie als mobile App zu starten.

Für die Umsetzung der Statistiken wurde das Plugin chart.js implementiert. Mit der Erweiterung ng2 charts wird die Entwicklung mit Angular unterstützt. Das Plugin stellt unterschiedliche Diagramme bereit, wodurch die Statistiken als Doughnut-Chart erzeugt werden können.

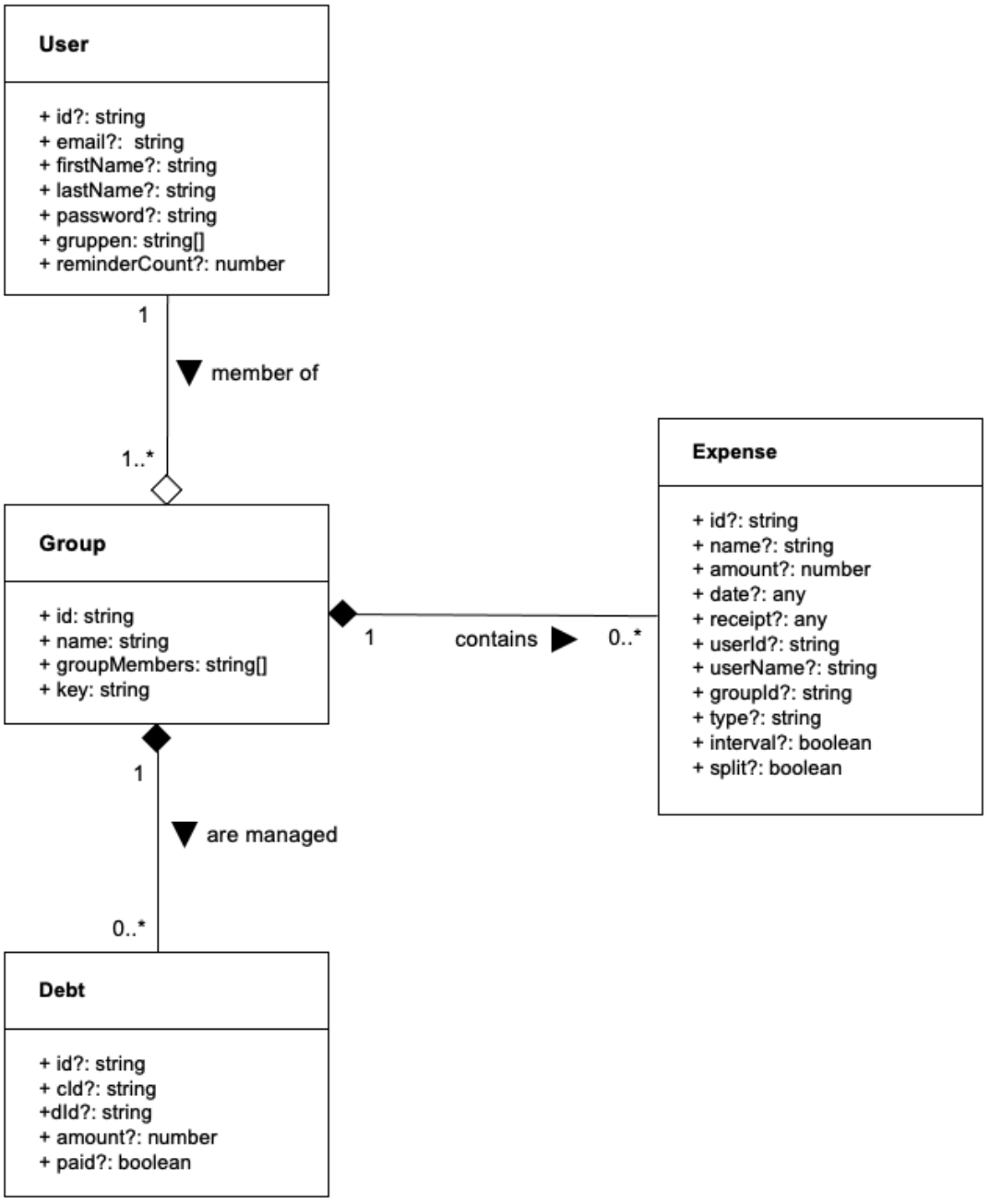
Um die Lauffähigkeit der Applikation auf Endgeräten mit einem Androidbetriebssystem zu garantieren, wurde die Entwicklungsumgebung Android Studio von JetBrains verwendet. Durch den darin enthaltenen Emulator wurde der zweite Usability Test auf einem Smartphone durchgeführt.

4.2 Standardisierte Notationen

4.2.1 Domänendiagram



4.2.1 Klassendiagramm



5. Implementierung

5.1 Technische Umsetzung

Um eine flüssige Entwicklung im Team zu gewährleisten, wurden zu Beginn verschiedene technologische Standards festgelegt. Bei der Projektstruktur wurde sich an der Vorlage orientiert, die die Angular CLI erstellt, wenn eine neue Seite erstellt wird. Außerdem gibt es beispielsweise für Komponenten und Bilddateien separate Ordner. Die genaue Projektstruktur wird im folgenden Kapitel beschrieben. Zum Vorbeugen von Mergekonflikten, wurden für die Bearbeitung eines neues Issues ein neuer Branch erstellt. Die beschriebenen Funktionen wurden mit Kommentaren versehen. So ist der Code auch für die anderen Teammitglieder verständlich. Außerdem wurden alle Kommentare in der API-Dokumentation sowie die Commit-Messages wurden in Englisch verfasst.

Dependencies:

"@angular/common": "~13.2.2",

"@angular/core": "~13.2.2",

"@angular/fire": "^7.4.1",

"@angular/forms": "~13.2.2",

"@angular/platform-browser": "~13.2.2",

"@angular/platform-browser-dynamic": "~13.2.2",

"@angular/router": "~13.2.2",

"@capacitor/android": "3.5.1",

"@capacitor/app": "1.1.1",

"@capacitor/core": "^3.6.0",

"@capacitor/filesystem": "^1.1.0",

"@capacitor/haptics": "1.1.4",

"@capacitor/keyboard": "1.2.2",

"@capacitor/share": "^1.1.2",

"@capacitor/status-bar": "1.0.8",

"@ionic/angular": "^6.0.0",

"chart.js": "^3.8.0",

"date-fns": "^2.28.0",

"ng2-charts": "^3.1.2",

"rxjs": "~6.6.0",

"tslib": "^2.2.0",

"zone.js": "~0.11.4"

Dev-Dependencies:

"@angular-devkit/build-angular": "~13.2.3",

"@angular-eslint/builder": "~13.0.1",

"@angular-eslint/eslint-plugin": "~13.0.1",

"@angular-eslint/eslint-plugin-template": "~13.0.1",

"@angular-eslint/template-parser": "~13.0.1",

"@angular/cli": "~13.2.3",

"@angular/compiler": "~13.2.2",

"@angular/compiler-cli": "~13.2.2",

"@angular/language-service": "~13.2.2",

"@capacitor/cli": "3.5.1",

"@ionic/angular-toolkit": "^6.0.0",

"@ionic/lab": "3.2.13",

"@types/jasmine": "~3.6.0",

"@types/jasminewd2": "~2.0.3",

"@types/node": "^12.11.1",

"@typescript-eslint/eslint-plugin": "5.3.0",

"@typescript-eslint/parser": "5.3.0",

"eslint": "^7.6.0",

"eslint-plugin-import": "2.22.1",

"eslint-plugin-jsdoc": "30.7.6",

"eslint-plugin-prefer-arrow": "1.2.2",

"jasmine-core": "~3.8.0",

"jasmine-spec-reporter": "~5.0.0",

"karma": "~6.3.2",

"karma-chrome-launcher": "~3.1.0",

"karma-coverage": "~2.0.3",

"karma-coverage-istanbul-reporter": "~3.0.2",

"karma-jasmine": "~4.0.0",

"karma-jasmine-html-reporter": "^1.5.0",

"protractor": "~7.0.0",

"ts-node": "~8.3.0",

"typescript": "~4.4.4"

Android Dependencies (build.gradle):

classpath 'com.android.tools.build:gradle:7.2.1'

classpath 'com.google.gms:google-services:4.3.13'

repositories:

google()

mavenCentral()

allprojects:

google()

mavenCentral()

gradle.wrapper.properties:

distributionUrl=https\://services.gradle.org/distributions/gradle-7.3.3-all.zip

variables.gradle:

minSdkVersion = 22

compileSdkVersion = 30

targetSdkVersion = 30

androidxActivityVersion = '1.2.0'

androidxAppCompatVersion = '1.2.0'

androidxCoordinatorLayoutVersion = '1.1.0'

androidxCoreVersion = '1.3.2'

androidxFragmentVersion = '1.3.0'

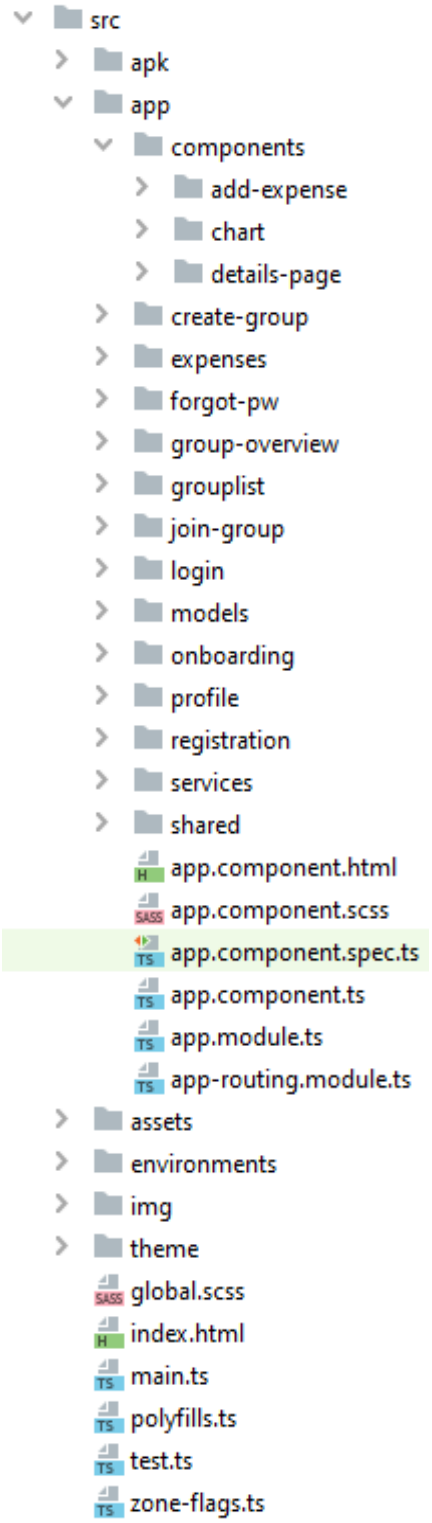
junitVersion = '4.13.1'

androidxJunitVersion = '1.1.2'

androidxEspressoCoreVersion = '3.3.0'

cordovaAndroidVersion = '10.1.1'

5.2 Projektstruktur



<- Die apk-Datei liegt im “apk” Ordner

<- Pages liegen in gleichnamigen Ordnern

<- Alle Services, liegen im “services” Ordner

<- Alle Bilder und Icon liegen im Ordner “assets” und die Firebase config im Ordner “environment”

6. Evaluation & Aussicht

Nach einigen Wochen Entwicklungsphase konnte der MVP vollständig umgesetzt werden und ebenso einige der als optional definierten Features. Die Usability im Allgemeinen kann noch weiter ausgebaut werden. So könnten die Nutzer*innen ein Profilbild hochladen oder auch das Layout könnte noch weiterentwickelt werden. Ebenso gibt es auch funktionale Weiterentwicklungsmöglichkeiten. Ein optionales Feature, das nicht bearbeitet werden konnte, ist ein Chat zwischen den Nutzern*innen. Äquivalent zu den Shamements könnten noch Achievement eingeführt werden, um die Motivation der Nutzer*innen weiter zu steigern. Ein wichtiger Punkt wäre, auswählen zu können, welche Gruppenmitglieder an Ausgaben/Einnahmen beteiligt werden sollen. Des Weiteren könnte eine Schnittstelle zu der Hardware der Smartphonekamera die Komfortabilität der Anwender steigern Bilder einer Ausgabe hinzufügen. Sodass sie nicht erst ein Bild machen müssen um es anschließend als Datei in der App hochzuladen. Weiterhin könnten Gruppen wiederverwendbar werden, indem mehr Daten veränderbar sind. Ein letzter Punkt wäre, die App zu deployen.

7. Anwenderdokumentation

Installation der App:

git clone	Um das aktuelle Repository über die Konsole der IDE zu klonen
npm i	Installiert die Abhängigkeiten
ionic serve	Startet die Anwendung

Ausführung der Seiten:

Onboarding	➔ Durchklicken
Registrierung	➔ Eingabefelder ausfüllen ➔ Auf „registrieren“ drücken
Login	➔ Eingabefelder ausfüllen ➔ Social Login via Google verwenden ➔ Auf „einloggen“ drücken ➔ Oder Passwort zurücksetzen
Passwort Vergessen	➔ Eingabefeld ausfüllen ➔ Auf „senden“ drücken
Gruppenliste	➔ Auf Gruppe drücken ➔ Auf Button drücken „neue gruppe erstellen“
Gruppe erstellen	➔ Eingabefeld ausfüllen ➔ Auf „erstellen“ drücken ➔ Auf „mit gruppen-id teilnehmen“ auf folgende Seite zu gelangen
An Gruppe teilnehmen	➔ Eingabefelder ausfüllen ➔ Auf „los geht´s“ drücken, um an bestehender Gruppe teilzunehmen
Gruppenübersicht	➔ Auf „Details anzeigen“ drücken ➔ Über das Menü Mitglieder bearbeiten, Gruppe verlassen oder Gruppeneinladung versenden

	➔ Shamements/Achievements an Gruppenmitglieder versenden über langes drücken auf den Namen ➔ Auf Back-Button drücken, um auf vorherige Seite zu gelangen
Aufteilung	➔ Auf „ausgaben & einnahmen aufteilen“ drücken, um Teilen auszulösen.
Einnahmen	➔ Auf das Plus in der Navigationsleiste drücken und “Einnahme hinzufügen” auswählen ➔ Auf „ausgaben & einnahmen aufteilen“ drücken, um Teilen auszulösen ➔ Auf Back-Button drücken, um auf vorherige Seite zu gelangen
Ausgaben	➔ Auf das Plus in der Navigationsleiste drücken und “Ausgabe hinzufügen” auswählen ➔ Auf „ausgaben & einnahmen aufteilen“ drücken, um Teilen auszulösen. ➔ Auf Back-Button drücken, um auf vorherige Seite zu gelangen
Statistik	➔ Über Selection-Dropdown nach Monat filtern ➔ Auf „ausgaben & einnahmen aufteilen“ drücken, um Teilen auszulösen. ➔ Auf Back-Button drücken, um auf vorherige Seite zu gelangen
Navigationsleiste	➔ Haus: Gruppenliste ➔ Plus: neue Gruppe/Ausgabe/Einnahme erstellen ➔ User: Profil