



TECHNISCHE HOCHSCHULE MITTELHESSEN

**THM**

**CAMPUS  
GIESSEN**

**MNI**

Mathematik, Naturwissenschaften  
und Informatik

**CS2018 Entwicklung mobiler Anwendungen WS2022/23**

# **Projektbericht**

**Gruppe 8**

Biebl, Maximilian

Wagner, Hendrik

Krs, Dennis

Nagy, Vanessa

Thomas, Sven

Dozent:

Prof. Dr. Steffen Vaupel

16. Januar 2023

Technische Hochschule Mittelhessen, Gießen

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Problem und Zielsetzung . . . . .	1
1.2	Anforderungen . . . . .	1
1.3	Ausgewählte Technologien . . . . .	1
1.4	Organisation und Vorgehensmodell . . . . .	2
<b>2</b>	<b>Anforderungen</b>	<b>3</b>
2.1	Grundlegende Funktionsbeschreibung . . . . .	3
2.2	Funktionale Anforderungen . . . . .	3
2.3	Nichtfunktionale Anforderungen . . . . .	12
2.4	(Benutzer) Schnittstellen / Ein-Ausgabeformate . . . . .	12
2.5	Fehlverhalten . . . . .	12
2.6	Abnahmekriterien . . . . .	13
<b>3</b>	<b>Entwurf</b>	<b>14</b>
3.1	Technische Funktionen (und Funktionsabhängigkeiten) . . . . .	14
3.2	Datenmodell . . . . .	14
3.3	Strukturmodell (Aufbau) . . . . .	14
3.4	Funktionsmodell (Verhalten) . . . . .	14
3.5	Testfälle . . . . .	14
<b>4</b>	<b>Umsetzung</b>	<b>15</b>
4.1	Schuldenberechnung . . . . .	15
4.2	CRUD-Handler . . . . .	16
4.3	Nichtumsetzung von Funktionen . . . . .	16
4.3.1	Google-Maps Integration . . . . .	16
4.3.2	Währungs-API . . . . .	16
4.3.3	QR-Code-Scanner . . . . .	16
4.4	Designunterschiede unter Android und iOS . . . . .	17
<b>5</b>	<b>Qualitätssicherung (Testprotokoll)</b>	<b>18</b>
<b>6</b>	<b>Evaluation</b>	<b>19</b>
6.1	Funktionale Anforderungen . . . . .	19
<b>7</b>	<b>Anwenderdokumentation</b>	<b>21</b>
7.1	Seitenerläuterung . . . . .	21
7.1.1	Anmeldeseite . . . . .	21
7.1.2	Registrierungsseite . . . . .	22
7.1.3	Home . . . . .	23
7.1.4	Reisebearbeiten/erstellen . . . . .	24
7.1.5	Detailansicht einer Reise . . . . .	25
7.1.6	Liste der beigetretenen Reisen . . . . .	27
7.1.7	Details einer Zahlung . . . . .	28
7.1.8	Details einer Zahlung beim Erstellen/Bearbeiten . . . . .	29
7.1.9	Schuldenansicht bei Guthaben . . . . .	30

7.1.10	Schuldenansicht bei Schulden . . . . .	31
7.1.11	Einladungsmodal . . . . .	32
7.1.12	Optionen . . . . .	33
7.2	Sitemap . . . . .	34

# 1 Einleitung

Auf den folgenden Seiten finden Sie unseren ausführlichen Projektbericht zu unserer App „Almanify“. In diesem Bericht werden die Anforderungen, die Entwicklungsprozesse und die Ergebnisse unserer Arbeit detailliert beschrieben. Wir werden auch auf die Herausforderungen eingehen, mit denen wir während der Entwicklung konfrontiert waren, und wie wir sie gelöst haben. Zudem werden wir die Funktionen und Möglichkeiten der App aufzeigen und eine Übersicht über die Ergebnisse der Entwicklung und der Benutzertests präsentieren.

## 1.1 Problem und Zielsetzung

Auf einer Reise mit mehreren Beteiligten kommt es oft vor, dass sich Teilnehmer gegenseitig Geld vorlegen, um eine Bezahlung zu vereinfachen. Schnell kann es dabei unübersichtlich werden, insbesondere bei einem Roadtrip mit mehreren Freunden, bei dem jeder für etwas anderes bezahlt und man nicht immer eine klare Übersicht hat. Einer kauft noch schnell ein paar Snacks für alle, der andere fährt zwischendurch das Auto tanken, und jemand hat kein Bargeld, um dem Fremdenführer Trinkgeld zu geben. Daher muss er sich das Geld leihen. Die App „Almanify“ soll dabei helfen, diese komplexe Situation zu vereinfachen, indem alle Beteiligten die Ausgaben transparent miteinander verwalten können. Am Ende sowie während der Reise soll es den Nutzern möglich sein, offenen Schulden gegenüber Mitreisenden einzusehen, um diese begleichen zu können.

## 1.2 Anforderungen

Bei Öffnung der App müssen sich Benutzer zunächst einloggen oder registrieren. Dann können Sie Reisen erstellen und andere Nutzer über einen Invitecode oder QR-Code einladen. Ausgaben im Rahmen einer Reise können hinzugefügt werden. Beim Erstellen von Ausgaben gibt vorgegebene Auswahlmöglichkeiten für Währung, Kategorie und Beteiligte. Nutzer können Einträge bearbeiten und Schulden gegenüber anderen auswerten. Die App schlägt einen effizienten finanziellen Ausgleich vor und Schulden werden in eine vom User gewählte Währung umgerechnet. Nach Abschluss einer Reise kann dies archiviert werden.

## 1.3 Ausgewählte Technologien

Die Anwendung basiert auf Angular. Als Framework für die Implementierung der mobilen Anwendung kommt Ionic zum Einsatz. Firebase dient als Backend für die Datenhaltung und Nutzerverwaltung.

Wir beschränken uns bei der Implementierung der nativen Komponenten der App auf die Android-Plattform.

## 1.4 Organisation und Vorgehensmodell

Die Gruppe wurde zunächst mit einem ausführlichen Kickoff-Meeting, welches schon erste Mockups enthielt, vom Ideengeber über das Konzept der App aufgeklärt. Im Anschluss an das Kickoff-Meeting wurden grundsätzliche Features und Designentscheidungen anhand der Mockups besprochen.

Im laufenden Projekt orientierten wir uns an Scrum. Die Sprints dauerten eine Woche. Wir trafen uns jeden Mittwoch in Discord, um die Features der vorherigen Woche zu besprechen und zu sehen, wo Verbesserungen nötig waren. Außerdem diskutierten wir, welche Features in der kommenden Woche implementiert werden sollten.

## 2 Anforderungen

### 2.1 Grundlegende Funktionsbeschreibung

Bei Öffnung der App können sich Benutzer einloggen oder registrieren. Dabei wird ein Benutzerprofil erstellt, das den Nutzernamen enthält. Reisen können innerhalb der mobilen Anwendung erstellt werden. Weitere Nutzer können über einen Invite-Code beitreten. Getätigte Ausgaben können den beigetretenen Reisen hinzugefügt werden. Einträge für Ausgaben umfassen einen Titel, die Person, die bezahlt hat, die Summe, Währung, eine Kategorie, Zahldatum, die Beteiligten, die Ausgaben verursacht haben und ein optionales Bild (z. B. vom Kassenbon). Für Währung, Kategorie und Beteiligte gibt es vorgegebene Auswahlmöglichkeiten. Einträge für Ausgaben können entsprechend der Attribute sortiert werden. Nutzer können Einträge für Ausgaben nachträglich bearbeiten. Es ist möglich, Schulden bzw. Ansprüche gegenüber anderen auszuwerten. Die App schlägt einen effizienten finanziellen Ausgleich unter den Beteiligten vor, sodass möglichst wenige Transaktionen nötig sind. Schulden des Nutzers gegenüber anderen werden in eine von ihm angegebene Währung umgerechnet. Nach Abschluss einer Reise kann dies archiviert werden.

### 2.2 Funktionale Anforderungen

Nun werden die funktionalen Anforderungen anhand von Use Cases beschrieben.

Tabelle 1: Use Case: Einen Account erstellen

Abschnitt	Inhalt
Primärer Akteur	Nutzer
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Nutzer öffnet die App und hat noch keinen Account. Er möchte jetzt einen Account erstellen.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Der Nutzer öffnet die App.</li> <li>2. Die App zeigt die Anmeldeseite an.</li> <li>3. Der Nutzer wählt auf der Anmeldeseite „Sign up“ aus.</li> <li>4. Er füllt dann seinen Namen, seine E-Mail-Adresse, und sein Passwort ein.</li> <li>5. Er bestätigt sein Passwort durch erneute Eingabe.</li> <li>6. Der Nutzer tippt auf „Sign up“, um seinen Account zu erstellen.</li> <li>7. Die App zeigt erneut die Anmeldeseite mit nun vorausgefüllten Feldern an.</li> <li>8. Es erscheint eine Meldung, dass der Account erfolgreich erstellt wurde.</li> </ol>
Alternativszenarien	(Keine)
Ausnahmeszenarien	<ol style="list-style-type: none"> <li>4a. Der Nutzer gibt keine gültige E-Mail-Adresse ein.  [4a1.] Die App zeigt eine Fehlermeldung an und fordert den Nutzer auf, eine gültige E-Mail-Adresse einzugeben.  [4a2.] Der <i>Sign up</i>-Knopf ist ausgegraut, bis der Nutzer eine gültige E-Mail-Adresse eingegeben hat. <i>Zurück zu 4.</i></li> <li>4b. Der Nutzer gibt ein Passwort ein, das nicht den Anforderungen entspricht.  [4b1.] Die App zeigt eine Fehlermeldung an und fordert den Nutzer auf, ein Passwort einzugeben, das die Anforderungen erfüllt.  [4b2.] Der <i>Sign up</i>-Knopf ist ausgegraut, bis der Nutzer ein Passwort eingegeben hat, das die Anforderungen erfüllt. <i>Zurück zu 4.</i></li> <li>5a. Der Nutzer gibt bei der Bestätigung des Passworts ein anderes Passwort ein.  [5a1.] Die App zeigt eine Fehlermeldung an und fordert den Nutzer auf, das gleiche Passwort erneut einzugeben.  [5a2.] Der <i>Sign up</i>-Knopf ist ausgegraut, bis der Nutzer das gleiche Passwort erneut eingegeben hat. <i>Zurück zu 5.</i></li> </ol>
Vor-/ Nachbedingungen	<p>Vor1. Der Nutzer ist derzeit nicht mit der <i>Remember Me</i>-Einstellung angemeldet.</p> <p>Nach1. Der Nutzer besitzt einen Account.</p>

Tabelle 2: Use Case: In Account einloggen

Abschnitt	Inhalt
Primärer Akteur	Nutzer
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Nutzer öffnet die App und hat bereits einen Account. Er möchte sich jetzt anmelden.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>Der Nutzer öffnet die App. Er ist nicht angemeldet.</li> <li>Die App zeigt die Anmeldeseite an.</li> <li>Der Nutzer gibt seine E-Mail-Adresse und sein Passwort ein.</li> <li>Er tippt auf <i>Login</i>.</li> <li>Er wird in den Account eingeloggt und auf seine aktive Reise weitergeleitet.</li> </ol>
Alternativszenarien	<ol style="list-style-type: none"> <li>Der Nutzer hat bei der letzten Anmeldung die <i>Remember Me</i>-Einstellung aktiviert. [1a1.] Die Anmeldedaten liegen gespeichert vor. <i>Weiter mit 5.</i></li> <li>Der Nutzer hat keine aktive Reise. [5a1.] Die App zeigt die Startseite an.</li> </ol>
Ausnahmeszenarien	<ol style="list-style-type: none"> <li>Der Nutzer gibt keine gültige E-Mail-Adresse ein. [3a1.] Die App zeigt eine Fehlermeldung an und fordert den Nutzer auf, eine gültige E-Mail-Adresse einzugeben. [3a2.] Der <i>Login</i>-Knopf ist ausgegraut, bis der Nutzer eine gültige E-Mail-Adresse eingegeben hat. <i>Zurück zu 3.</i></li> <li>Die Zugangsdaten sind ungültig. [4a1.] Die App zeigt eine Fehlermeldung an und fordert den Nutzer auf, die Zugangsdaten erneut einzugeben. <i>Zurück zu 3.</i></li> </ol>
Vor-/ Nachbedingungen	Vor1. Der Nutzer besitzt einen Account.



Tabelle 3: Use Case: Eine Reise erstellen

Abschnitt	Inhalt
Primärer Akteur	Reiseleiter
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Nutzer möchte eine neue Reise anlegen.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Der angemeldete Nutzer navigiert zur Startseite (<i>Home</i>).</li> <li>2. Er tippt auf <i>Create Journey</i>.</li> <li>3. Ein Dialog öffnet sich, in dem er die Reiseinformationen eingeben kann.</li> <li>4. Er gibt einen Titel, eine Standardwährung sowie ein Start- und Enddatum ein.</li> <li>5. Er tippt auf <i>Save</i>.</li> <li>6. Nach Erstellen der Reise öffnet sich das Einladungsmodal.</li> </ol>
Alternativszenarien	(Keine)
Ausnahmeszenarien	<ol style="list-style-type: none"> <li>4a. Der Nutzer gibt einen ungültigen Reisezeitraum ein. <ol style="list-style-type: none"> <li>[4a1.] Die App zeigt eine Fehlermeldung an und fordert den Nutzer auf, einen gültigen Reisezeitraum einzugeben.</li> <li>[4a2.] Der <i>Save</i>-Knopf ist ausgegraut, bis der Nutzer einen gültigen Reisezeitraum eingegeben hat. <i>Zurück zu 4.</i></li> </ol> </li> </ol>
Vor-/ Nachbedingungen	<p>Vor1. Der Nutzer besitzt einen Account und ist angemeldet.</p> <p>Nach1. Es existiert eine neue Reise.</p>

Tabelle 4: Use Case: Eine Reise beitreten

Abschnitt	Inhalt
Primärer Akteur	Nutzer
Weitere Akteure	Reiseleiter
Auslösende Ereignisse	Der Nutzer möchte einer Reise beitreten, die der Reiseleiter erstellt hat.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Der angemeldete Nutzer navigiert zur Startseite (<i>Home</i>).</li> <li>2. Er tippt auf <i>Join Journey</i>.</li> <li>3. Er gibt den Invitecode ein, den der Reiseleiter ihm mitgeteilt hat.</li> <li>4. Er tippt auf <i>Apply</i>.</li> <li>5. Der Nutzer wird zur Reise hinzugefügt. Die App leitet ihn zur Reise weiter.</li> </ol>
Alternativszenarien	<ol style="list-style-type: none"> <li>3a. Der Nutzer benutzt das QR-Code-Feature.               <ol style="list-style-type: none"> <li>[3a1.] Der Nutzer tippt auf <i>Scan QR-Code</i>.</li> <li>[3a2.] Die App öffnet die Kamera.</li> <li>[3a3.] Der Nutzer scannt den QR-Code. <i>Weiter mit 5.</i></li> </ol> </li> </ol>
Ausnahmeszenarien	<ol style="list-style-type: none"> <li>3b. Der Nutzer gibt einen ungültigen Invitecode ein.               <ol style="list-style-type: none"> <li>[3b1.] Die App meldet einen invaliden Invitecode. <i>Zurück zu 3.</i></li> </ol> </li> </ol>
Vor-/ Nachbedingungen	<p>Vor1. Der Nutzer besitzt einen Account und ist angemeldet.            Vor2. Der Nutzer ist nicht Mitglied der Reise.            Nach1. Der Nutzer ist Mitglied der Reise.</p>

Tabelle 5: Use Case: Zahlung festhalten

Abschnitt	Inhalt
Primärer Akteur	Reisemitglied
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Reisemitglied möchte eine neue Zahlung festhalten, die er oder eine andere Person im Rahmen der Reise getätigt hat.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Das Reisemitglied ist auf der Seite der Reise.</li> <li>2. Er tippt auf <i>New Payment</i>.</li> <li>3. Ein Dialog öffnet sich, in dem er die Daten der Zahlung eingibt.</li> <li>4. Er fügt einen Titel hinzu, wählt den Zahlenden, die Währung, den Betrag, das Datum und die Kategorie aus.</li> <li>5. Er wählt die Reiseteilnehmer aus, für die die Zahlung getätigt wurde.</li> <li>6. Er tippt auf <i>Save</i>. Die Zahlung wird gespeichert.</li> </ol>
Alternativszenarien	<ol style="list-style-type: none"> <li>4a. Der Nutzer möchte zusätzlich ein Bild einfügen. <ol style="list-style-type: none"> <li>[4a1.] Er tippt auf die <i>Add Image</i> Schaltfläche.</li> <li>[4a2.] Die App öffnet die Kamera.</li> <li>[4a3.] Der Nutzer macht ein Foto, welches hochgeladen wird.</li> </ol> <i>Zurück zu 4.</i> </li> </ol>
Ausnahmeszenarien	<ol style="list-style-type: none"> <li>4b. Der Nutzer gibt einen ungültigen (negativen) Betrag ein. <ol style="list-style-type: none"> <li>[4b1.] Die App meldet einen invaliden Betrag.</li> <li>[4a2.] Der <i>Save</i>-Knopf ist ausgegraut, bis der Nutzer einen gültigen Betrag eingegeben hat. <i>Zurück zu 4.</i></li> </ol> </li> </ol>
Vor-/ Nachbedingungen	Nach1. Die Zahlung ist gespeichert.

Tabelle 6: Use Case: Zahlung einsehen

Abschnitt	Inhalt
Primärer Akteur	Reisemitglied
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Reisemitglied möchte eine eingetragene Zahlung einsehen.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Das Reisemitglied ist auf der Seite der Reise.</li> <li>2. Er tippt auf eine Zahlung.</li> <li>3. Die Zahlung weitet sich aus und zeigt grundlegende Informationen wie Zahler und Zahldatum an.</li> <li>4. Der Nutzer tippt auf das Auge, um die Details der Zahlung einzusehen.</li> <li>5. Die Zahlungsdetails werden angezeigt.</li> </ol>
Alternativszenarien	(Keine)
Ausnahmeszenarien	(Keine)
Vor-/ Nachbedingungen	Vor1. Es existiert eine Zahlung.

Tabelle 7: Use Case: Schulden einsehen

Abschnitt	Inhalt
Primärer Akteur	Reisemitglied
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Reisemitglied möchte einsehen, welche Schulden er hat bzw. was ihm geschuldet wird.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Das Reisemitglied ist auf der Seite der Reise.</li> <li>2. Er tippt auf <i>Debts</i>.</li> <li>3. Der Schuldenrechner öffnet sich.</li> <li>4. Dem Nutzer wird die Gesamtsumme an Geld angezeigt, das er anderen Reisemitgliedern schuldet.</li> <li>5. Darunter werden die einzelnen Schulden aufgeführt.</li> <li>6. Der Nutzer kann die Währung ändern, indem er auf <i>Currency</i> tippt.</li> <li>7. Der Nutzer kann eine getätigte Ausgleichszahlung eintragen, indem er auf <i>Pay</i> neben einem Betrag tippt.</li> </ol>
Alternativszenarien	<ol style="list-style-type: none"> <li>4a. Der Nutzer hat keine Schulden, aber andere Reisemitglieder schulden ihm Geld. <ol style="list-style-type: none"> <li>[4a1.] Dem Nutzer wird die Gesamtsumme an Geld angezeigt, die andere Reisemitglieder ihm schulden.</li> <li>[4a2.] Darunter werden die einzelnen Schulden aufgeführt.</li> <li>[4a3.] Der Nutzer kann die Währung ändern, indem er auf <i>Currency</i> tippt.</li> <li>[4a4.] Der Nutzer kann eine getätigte Ausgleichszahlung eintragen, indem er auf <i>Mark as repaid</i> neben einem Betrag tippt.</li> </ol> </li> <li>4b. Der Nutzer hat keine Schulden, und andere Reisemitglieder schulden ihm auch nichts. <ol style="list-style-type: none"> <li>[4b1.] Dem Nutzer wird angezeigt, dass er keine Schulden hat.</li> </ol> </li> </ol>
Ausnahmeszenarien	(Keine)
Vor-/ Nachbedingungen	(Keine)

Tabelle 8: Use Case: Reise archivieren

Abschnitt	Inhalt
Primärer Akteur	Reiseleiter
Weitere Akteure	(Keine)
Auslösende Ereignisse	Ein Reiseleiter möchte eine Reise archivieren.
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Der Reiseleiter ist auf der Liste aller Reisen.</li> <li>2. Er scrollt zur seiner Reise, die er archivieren möchte.</li> <li>3. Er tippt auf das Schlosssymbol.</li> <li>4. Er bestätigt die Archivierung.</li> </ol>
Alternativszenarien	(Keine)
Ausnahmeszenarien	(Keine)
Vor-/ Nachbedingungen	<p>Vor1. Die Reise darf nicht bereits archiviert sein.</p> <p>Nach1. Die Reise ist archiviert.</p>

Tabelle 9: Use Case: Reisen einsehen

Abschnitt	Inhalt
Primärer Akteur	Reisemitglied
Weitere Akteure	(Keine)
Auslösende Ereignisse	Das Reisemitglied möchte eine Reise mit deren Zahlungen einsehen
Szenario	Beschreibung
Hauptszenario	<ol style="list-style-type: none"> <li>1. Das Reisemitglied ist auf der Liste aller Reisen.</li> <li>2. Er scrollt zur Reise, die er einsehen möchte.</li> <li>3. Er tippt auf das Bild der Reise.</li> <li>4. Er wird auf die Reise weitergeleitet.</li> </ol>
Alternativszenarien	(Keine)
Ausnahmeszenarien	(Keine)
Vor-/ Nachbedingungen	<p>Nach1. Das Reisemitglied ist auf der Seite der Reise.</p>

### 2.3 Nichtfunktionale Anforderungen

1. **Sicherheit:** Die App soll sicher sein, d. h. keine sensiblen Daten wie Passwörter oder Kreditkartendaten speichern.  
So soll die Menge an Daten, die von Nutzern in der App oder auf dem Server gespeichert werden, möglichst klein gehalten werden.
2. **Benutzerfreundlichkeit:** Die App soll einfach zu bedienen sein, damit die Nutzer schnell und einfach mit ihr arbeiten können.  
Dazu soll die App eine intuitive Benutzeroberfläche haben, die den Nutzern die Bedienung der App erleichtert.
3. **Schnelligkeit:** Die App soll schnell reagieren, damit die Nutzer nicht lange auf die Ergebnisse warten müssen.
4. **Robustheit:** Die App soll robust sein, d. h. sie soll auch bei Fehlern oder unerwarteten Eingaben nicht abstürzen.  
Dazu soll die App Fehlermeldungen ausgeben, wenn sie nicht mit den Eingaben umgehen kann, und in einen sicheren Zustand zurückkehren.
5. **Wartbarkeit:** Die App soll einfach zu warten sein, damit die Entwickler schnell und einfach Fehler beheben können.  
Dazu soll die App gut dokumentiert sein und eine gute Struktur haben. Code soll sinnvoll aufgeteilt sein und in einzelne Funktionen ausgelagert werden. Wenn möglich soll der Code wiederverwendet werden, um Redundanzen zu vermeiden.
6. **Portierbarkeit:** Die App soll auf möglichst vielen Geräten laufen, damit möglichst viele Nutzer sie verwenden können.  
Dazu soll die App auf Android und iOS laufen und das Design der App an die jeweiligen Geräte angepasst sein.
7. **Korrektheit:** Die App soll den funktionalen Anforderungen genügen.
8. **Zuverlässigkeit:** Die App soll den Vorstellungen der Nutzer entsprechen.

### 2.4 (Benutzer) Schnittstellen / Ein-Ausgabeformate

Die App soll mit dem Server kommunizieren, um die Daten zu erhalten und zu speichern. Es gibt festgelegte Datenstrukturen zu Nutzern, Reisen und Zahlungen, die der Server und die App verwenden.

- **Nutzer:** Ein Nutzer hat einen Namen und eine Standardwährung<sup>1</sup>.
- **Reise:** Eine Reise hat einen Namen, eine Beschreibung, ein Startdatum, ein Enddatum, eine Standardwährung, einen Invite-Code, einen Reiseleiter und eine Liste von Reisemitgliedern.
- **Zahlung:** Eine Zahlung hat eine zugeordnete Reise, einen Betrag, einen Absender, einen Empfänger, eine Beschreibung, ein Datum und eine Währung und ein optionales Bild.

### 2.5 Fehlverhalten

Werden invalide Daten eingegeben, soll die App eine Fehlermeldung ausgeben und in einen sicheren Zustand zurückkehren. Dies soll ebenfalls für zu große Datenmengen gelten. Um solche Fehler zu vermeiden, soll die App die Eingaben validieren.

---

<sup>1</sup>Die E-Mail-Adresse und das Passwort wird von Firebase verwaltet.

### 2.6 Abnahmekriterien

Die App soll die nicht- und funktionalen Anforderungen erfüllen. Dabei haben die funktionalen Anforderungen Priorität. Die App soll die Datenstrukturen (ggf. nach Bedarf angepasst) verwenden, um mit dem Server zu kommunizieren. Sie soll die Daten validieren und sicher Fehlermeldungen ausgeben, wenn die Eingaben nicht valide sind.



## **3 Entwurf**

### **3.1 Technische Funktionen (und Funktionsabhängigkeiten)**

### **3.2 Datenmodell**

### **3.3 Strukturmodell (Aufbau)**

### **3.4 Funktionsmodell (Verhalten)**

### **3.5 Testfälle**

## 4 Umsetzung

### 4.1 Schuldenberechnung

Für die Schuldenberechnung musste zunächst ein Algorithmus entwickelt werden, der die Schulden berechnet innerhalb einer bestimmten Reise berechnet. Es gab eine Reihe von Anforderungen an den Algorithmus, die erfüllt werden mussten:

- Der Algorithmus soll alle Zahlungen innerhalb der Reise berücksichtigen.
- Hat Person A Person B zweimal Geld *geliehen* (d. h. für diese Person eine Zahlung getätigt), sollen diese Zahlungen zusammengefasst werden.
- Zahlungen können mehrere Personen betreffen, z. B. wenn eine Person für mehrere Personen bezahlt. In diesem Fall muss die Schuld aufgeteilt werden.
- Die Anzahl an geforderten Ausgleichszahlungen soll minimiert werden.

Besonders der letzte Punkt verursachte Schwierigkeiten, da es sich hier um ein Optimierungsproblem handelt. Ein Ansatz war es, die Zahlungen als einen Graphen zu modellieren, in dem die Knoten die Personen und die Kanten die Zahlungen darstellen. Ausgehend von diesem Modell sollten dann Kanten vereinfacht werden. Dafür konnten einige Regeln aufgestellt werden:

1. Transitiv:  $x > y : A \xrightarrow{x} B \xrightarrow{y} C$  wird zu  $A \xrightarrow{x-y} B$  und  $A \xrightarrow{y} C$
2. Transitiv 2:  $x < y : A \xrightarrow{x} B \xrightarrow{y} C$  wird zu  $A \xrightarrow{x} C$  und  $A \xrightarrow{y-x} C$
3. Transitiv 3:  $x = y : A \xrightarrow{x} B \xrightarrow{y} C$  wird zu  $A \xrightarrow{x} C$
4. Reflexiv:  $A \xrightarrow{x} A$  wird eliminiert (da die Person sich selbst nichts schuldet)
5. Symmetrisch:  $A \xrightarrow{x} B \xrightarrow{y} A$  wird zu  $A \xrightarrow{x-y} B$  (bzw.  $A \xleftarrow{y-x} B$ )
6. Kreise: Bei einem Kreis wird die kleinste Kante entfernt und die restlichen Kanten um den Betrag der entfernten Kante verkleinert

Diese händische Vereinfachung stellte sich als übertrieben aufwändig heraus, als eine andere Idee aufkam: Statt dem Zahlungsgraphen wird lediglich die Liste der Zahlungen betrachtet. Für jede Person wird ein Wert verwaltet, auf welchen für jede Zahlung, in die der Person profitiert hat, der (Teil-)betrag subtrahiert wird. Für jede Zahlung, die die Person selbst getätigt hat, wird der Betrag addiert.

Mit diesen Werten wird dann wie in diesem Pseudocode beschrieben vorgegangen:

```

sei P eine gefüllte Map [Person, Wert]
sei S eine leere Liste von zu begleichenden Schulden [Schuldner,
    Glaebiger, Betrag]
sortiere P nach Werten aufsteigend
solange P nicht leer ist
    Person A = erste Person in P (Person mit kleinstem Wert, d.h. Person,
    die am meisten schuldet)
    Person B = letzte Person in P (Person mit groesstem Wert)
    Betrag = min(negiere(Wert von A), Wert von B)
    Fuege in S eine neue Schuld [A, B, Betrag] ein
    setze Wert von A auf Wert von A - Betrag
    setze Wert von B auf Wert von B + Betrag
    entferne A aus P, wenn Wert von A = 0
    entferne B aus P, wenn Wert von B = 0

```

Mit diesem Algorithmus werden die Schulden berechnet und in einer Liste gespeichert. Diese Liste wird dann verwendet, um eventuell relevante Schulden dem Benutzer anzuzeigen. Da der Algorithmus sehr simpel ist (Laufzeit  $\mathcal{O}(n^2)$ , generiert durch das Sortieren der Liste), genügt es, ihn auf Nachfrage im Client zu berechnen und die Schulden nicht im Server zu speichern.

### 4.2 CRUD-Handler

Die App kommuniziert mit mehreren Datenbanktabellen und muss daher für jede Tabelle einen eigenen Handler implementieren. Um die Implementierung zu vereinfachen, wurde ein generischer Handler entwickelt, der die CRUD-Operationen für eine beliebige Tabelle erledigt. Dieser Handler wird mit dem Namen der Tabelle initialisiert und kann dann die CRUD-Operationen für diese Tabelle ausführen.

1. **createAndGetID**: Erstellt einen neuen Eintrag in der Tabelle.
2. **readByID**: Liest einen Eintrag aus der Tabelle anhand der ID.
3. **update**: Aktualisiert einen Eintrag in der Tabelle.
4. **delete**: Löscht einen Eintrag aus der Tabelle.

### 4.3 Nichtumsetzung von Funktionen

#### 4.3.1 Google-Maps Integration

Die App sollte für eine Zahlung eine Karte anzeigen können, auf der die Position des Zahlungsortes angezeigt werden könnte. Dafür sollte die App die Google Maps API verwenden.

Die Integration der Google Maps API in die App wäre jedoch sehr aufwändig gewesen – so hätten wir neben der Darstellung auch die Möglichkeit der Suche nach Orten implementieren müssen. Zudem müssten GPS-Daten gesammelt werden, um einen Zahlungsort sinnvoll vorzuschlagen.

Diese Funktion wurde nicht umgesetzt, da das Feature nicht zwingend notwendig war und in vielen Fällen keine sinnvollen zusätzlichen Daten liefern würde – so ist z. B. der Name des Restaurants oft in der Zahlung als Titel angegeben. Auch war der generierte Nutzen gering, da Nutzer wenig Nutzen für die Information haben, wo sie eine Zahlung getätigt haben.

#### 4.3.2 Währungs-API

Die App sollte die Möglichkeit bieten, die Währung der Zahlungen zu ändern. Dies wurde umgesetzt, nicht jedoch das Abrufen der aktuellen Wechselkurse.

Dies hatte mehrere Gründe:

1. Die API, die wir für die Umrechnung der Währungen verwenden wollten, ist kostenpflichtig. Wir konnten keine kostenlose API finden, die die Wechselkurse aktuell hält.
2. Die Schuldenbegleichung könnte durch die Umrechnung der Währungen verfälscht werden, da die Wechselkurse nicht aktuell sind. Dann könnte es z. B. nach einer Woche so aussehen, als ob ein Schuldner etwas zu viel oder zu wenig bezahlt hat, wenn die Wechselkurse sich in der Zwischenzeit geändert haben.

#### 4.3.3 QR-Code-Scanner

Die App sollte die Möglichkeit bieten, einer Reise mittels QR-Code beizutreten. Die Generierung des QR-Codes wurde implementiert, jedoch nicht die Funktion, den QR-Code zu scannen. Dies wurde nicht umgesetzt, da das Plugin für den QR-Code-Scanner nicht funktioniert hat und wir das Problem trotz Zuhilfenahme von Dokumentation und weiteren Quellen nicht lösen konnten.

### 4.4 Designunterschiede unter Android und iOS

Die App wurde hauptsächlich für Web und Android entwickelt. Die iOS-Version wird beim Ionic-Framework automatisch generiert, das Styling wurde aber für Android angepasst. Es wurden nachträglich einige Änderungen vorgenommen, um das Design auf iOS zu verbessern.

## 5 Qualitätssicherung (Testprotokoll)

## 6 Evaluation

In diesem Kapitel wird evaluiert, inwiefern die Software die gestellten Anforderungen umsetzt. Dabei werden die funktionalen Anforderungen betrachtet.

### 6.1 Funktionale Anforderungen

#### Use Case 1: Einen Account erstellen

Hauptszenario: Alle Kriterien wurden erfüllt.

Ausnahmeszenarien: Das Szenario 4b wurde nicht umgesetzt, da für die Passworteingabe keine Regelvalidierung umgesetzt wurde. Dementsprechend wird keine Fehlermeldung wie z.B. 'Das Passwort muss mindestens 8 Zeichen lang sein, einen Großbuchstaben und ein Sonderzeichen enthalten.' angezeigt. Dies wäre Aufgabe des Teammitglieds gewesen, das abgesprungen ist.

#### Use Case 2: In Account einloggen

Hauptszenario: Der Nutzer wird nicht wie Punkt 5 beschreibt auf seine aktive Reise weitergeleitet. Hier haben wir uns dazu entschieden, stattdessen auf die Liste aller aktiven Reisen weiterzuleiten, weil es als verwirrend empfunden wurde nach dem Login nicht in einer Form von Übersicht zu landen. Alle anderen Kriterien wurden erfüllt.

Alternativszenarien: Alle Kriterien wurden erfüllt.

Ausnahmeszenarien: Der Login-Button ist nicht wie gefordert ausgegraut, wenn der Nutzer eine ungültige E-Mail-Adresse eingibt. Dies wäre Aufgabe des Teammitglieds gewesen, das abgesprungen ist. Der Button ist lediglich ausgegraut, solange noch keine E-Mail-Adresse bzw. Passwort eingegeben wurde. Alle anderen Kriterien wurden erfüllt.

#### Use Case 3: Eine Reise erstellen

Hauptszenario: Alle Kriterien wurden erfüllt.

Ausnahmeszenarien: Es können ungültige Reisezeiträume eingegeben werden (Enddatum liegt vor Startdatum). Dies wäre Aufgabe des Teammitglieds gewesen, das abgesprungen ist.

#### Use Case 4: Eine Reise betreten

Hauptszenario: Alle Kriterien wurden erfüllt.

Alternativszenarien: Der QR-Scanner wurde nicht implementiert. Grund dafür war, dass wir das QR-Scanner Plugin nicht lauffähig bekommen haben.

Ausnahmeszenarien: Alle Kriterien wurden erfüllt.

#### Use Case 5: Zahlung festhalten

Hauptszenario: Alle Kriterien wurden erfüllt.

Alternativszenarien: Der Nutzer kann ein Bild hochladen, allerdings nur Bilder aus dem Gerätespeicher und nicht per Kamera. Wir haben dafür die Camera API von Capacitor verwendet. Die Kamera konnte geöffnet werden und ein Foto konnte ebenfalls geschossen

werden, allerdings wurde das geschossene Foto nicht hochgeladen; diesen Fehler konnten wir nicht beheben.

Ausnahmeszenarien: Der Nutzer kann negative Beträge eingeben, ohne dass die App dies verhindert oder meldet. Dies wäre Aufgabe des Teammitglieds gewesen, das abgesprungen ist.

### **Use Case 6: Zahlung einsehen**

Hauptszenario: Alle Kriterien wurden erfüllt.

### **Use Case 7: Schulden einsehen**

Hauptszenario: Alle Kriterien wurden erfüllt.

Alternativszenarien: Alle Kriterien wurden erfüllt.

### **Use Case 8: Reise archivieren**

Hauptszenario: Alle Kriterien wurden erfüllt.

### **Use Case 9: Reisen einsehen**

Hauptszenario: Alle Kriterien wurden erfüllt.

Somit wurden bis auf die genannten Punkte alle funktionalen Anforderungen, die in Form von Use Cases formuliert wurden, umgesetzt.

## 7 Anwenderdokumentation

### 7.1 Seitenerläuterung

#### 7.1.1 Anmeldeseite

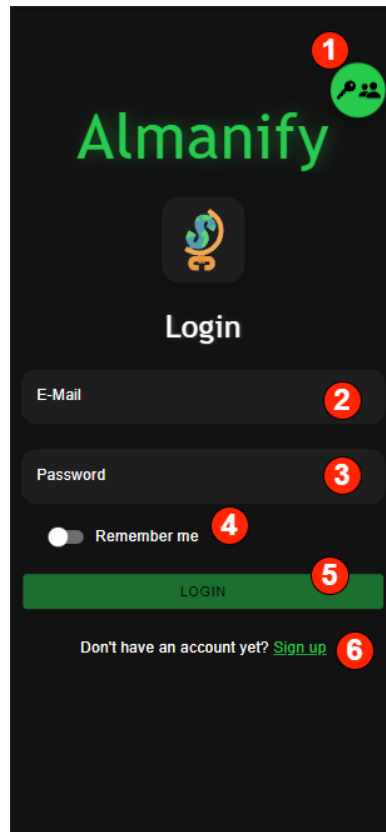


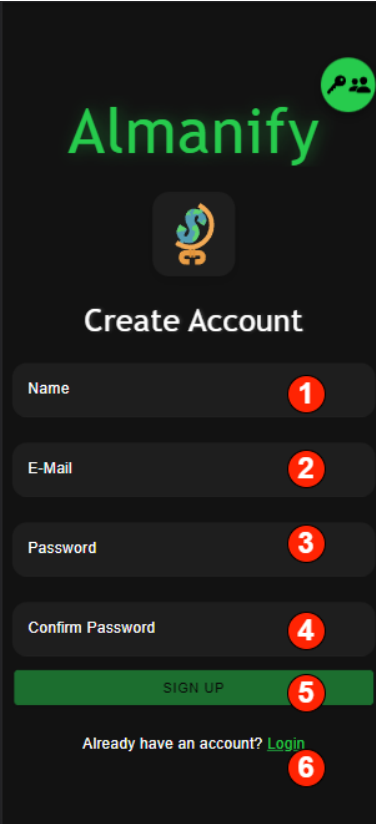
Abbildung 1: Login

Auf dieser Seite kann sich der User anmelden.

- ① Schnelllogin: ermöglicht das schnelle Anmelden mit verschiedenen Testusern. (nur im Debugmodus vorhanden)
- ② Eingabefeld für die E-Mail-Adresse des Nutzers, mit der er sich registriert hat.
- ③ Eingabefeld für das Passwort des Nutzers.
- ④ *Remember Me*: der User kann auswählen, ob er sich nach Schließen der App erneut anmelden möchte.
- ⑤ Button um sich einzuloggen.
- ⑥ Wenn der Nutzer noch keinen Account hat gelangt er hier zur Registrierungsseite (Siehe 7.1.2).



### 7.1.2 Registrierungsseite



The image shows a mobile app interface for 'Almanify' with a dark background. At the top, the 'Almanify' logo is in green, accompanied by a green circular icon with a white keyhole and two people. Below the logo is a square icon with a blue and orange stylized 'S' and 'G'. The text 'Create Account' is centered in white. Below this are four input fields: 'Name', 'E-Mail', 'Password', and 'Confirm Password', each with a red circle containing a white number (1, 2, 3, 4) to its right. A green 'SIGN UP' button with a red circle containing a white number 5 is below the fields. At the bottom, the text 'Already have an account? Login' is shown, with 'Login' in green and a red circle containing a white number 6 to its right.

Abbildung 2: Signup

Auf dieser Seite kann sich der User registrieren.

- ① Eingabefeld für den Nutzernamen, welcher den Mitreisenden später angezeigt wird.
- ② Eingabefeld für die E-Mail-Adresse des Nutzers.
- ③ Eingabefeld für das Passwort des Nutzers.
- ④ Eingabefeld um das Passwort des Nutzers zu bestätigen.
- ⑤ Button um sich zu registrieren.
- ⑥ Wenn der Nutzer einen Account hat gelangt er hier zurück zu Loginseite (Siehe 7.1.1).

### 7.1.3 Home

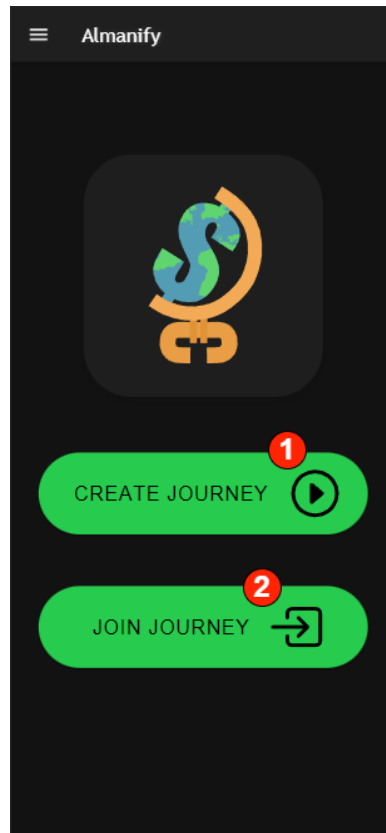


Abbildung 3: Home

Auf diese Seite gelangt der User nach Login, wenn keine aktive Reise vorhanden ist.

- ① Button zum Erstellen einer neuen Reise: der Nutzer gelangt zur Reiseerstellenseite (Siehe 7.1.4).
- ② Button zum Beitreten einer existierenden Reise: der Nutzer gelangt zur Reisebeitretenseite (Siehe [TODO]).

### 7.1.4 Reisebearbeiten/erstellen

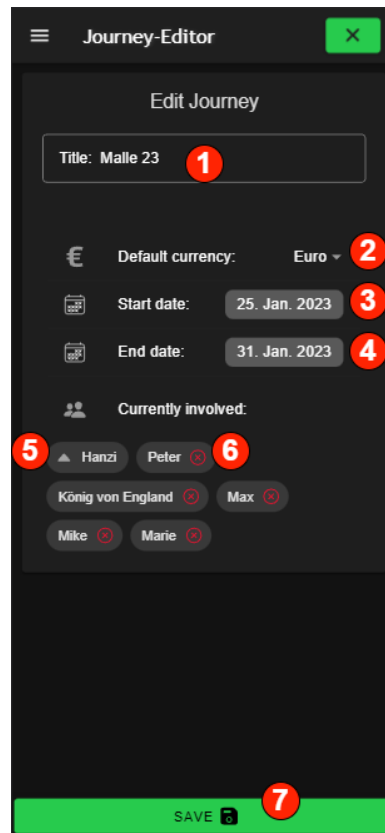


Abbildung 4: Journey-Editor

Auf dieser Seite kann der User eine Reise erstellen oder bearbeiten. Entsprechend der Situation ist der Titel „New Journey“ oder „Edit Journey“.

- ① Eingabefeld um der Reise einen Titel zugeben der später für alle Angezeigt wird.
- ② Dropdown zur Auswahl einer Standardwährung auf der Reise. Die Standardwährung ist später beim Erstellen von Zahlungen voreingestellt.
- ③ Öffnet Modal zur Wahl des Startdatums der Reise.
- ④ Öffnet Modal zur Wahl des Enddatums der Reise.
- ⑤ Ersteller einer Reise wird mit Dreieckicon markiert.
- ⑥ Nicht-Ersteller einer Reise können mit x-Button aus der Reise gekickt werde.
- ⑦ Button zum Speichern des Eintrags.

### 7.1.5 Detailansicht einer Reise

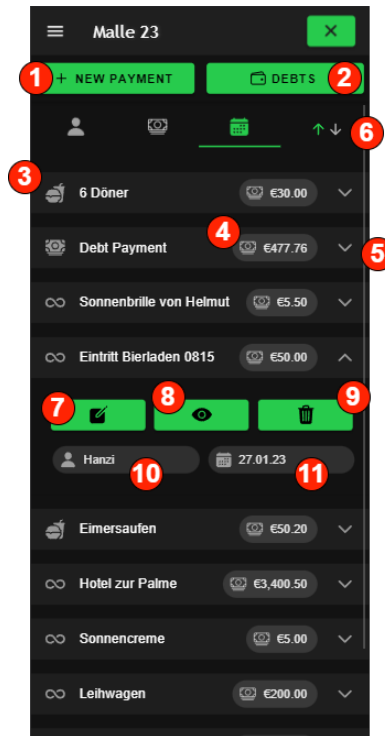








Abbildung 5: Journey-Details

Auf dieser Seite hat der Nutzer eine Übersicht aller Zahlungen einer Reise. Der Nutzer gelangt nach Login auf diese Seite, wenn es sich um die aktuellste Reise handelt.

- ① Button zum Erstellen einer neuen Zahlung: der Nutzer gelangt zur Zahlungerstellenseite (Siehe 7.1.8).
- ② Button zur Schuldenübersicht: der Nutzer gelangt zu Schuldenübersicht der Reise (Siehe 7.1.10 und 7.1.9).
- ③ Icon sagt über Kategorie der Zahlung aus (siehe Tabelle 10).
- ④ Betrag und Währung einer Zahlung.
- ⑤ Öffnen von weiteren Optionen zu einer Zahlung.
- ⑥ Zahlung können auf- und absteigend anhand des Zahlers, des Betrags und des Zahldatums sortiert werden.
- ⑦ Button zum Bearbeiten einer Zahlung: Der Nutzer gelangt zur Zahlungsbearbeitenseite (Siehe 7.1.8).
- ⑧ Button zu Details einer Ausgabe: Der Nutzer gelangt zur Detailansicht einer Zahlung (Siehe 7.1.7).
- ⑨ Button zum Löschen einer Zahlung.
- ⑩ Nutzernamen des Zahlers.
- ⑪ Zahldatum.

Tabelle 10: Zahlungskategorien

Bezeichnung	Icon	Beschreibung
Accommodation		Ausgaben für Unterkunft.
Food & Drink		Ausgaben für Essen und Trinken.
Entertainment		Ausgaben für Unterhaltung.
Transfer		Ausgaben die Transportmittel betreffen.
Repayment		Rückzahlung von Schulden an andere Mitreisende.
Other		alles was in keine andere Kategorie passt.

### 7.1.6 Liste der beigetretenen Reisen



Abbildung 6: Journey-List

Auf dieser Seite hat der Nutzer eine Übersicht aller beigetretenen Reisen.

- ① Button zum Erstellen einer neuen Reise: der Nutzer gelangt zur Reiseerstellenseite (Siehe 7.1.4).
- ② Auswahl zwischen aktiven und archivierten Reisen.
- ③ Anzeige, ob es sich um eine aktive oder archivierte Reise handelt.
- ④ Button zu Details einer Reise: der Nutzer gelangt zu einer Listenansicht aller Reisezahlungen (Siehe 7.1.5).
- ⑤ Der Nutzer bekommt das Einladungsmodal angezeigt (Siehe 7.1.11).
- ⑥ Button zum Reisebearbeiten: der Nutzer gelangt zur Reisebearbeitenseite (Siehe 7.1.4).
- ⑦ Button zum Löschen einer Reise.
- ⑧ Button zum Archivieren einer Reise.
- ⑨ Titel der Reise als Vektorbild; kann vom Nutzer durch eigenes Bild ersetzt werden.
- ⑩ Ist der Nutzer nicht Ersteller einer Reise, wird ihm nur der Button zu den Details einer Reise angezeigt.

### 7.1.7 Details einer Zahlung

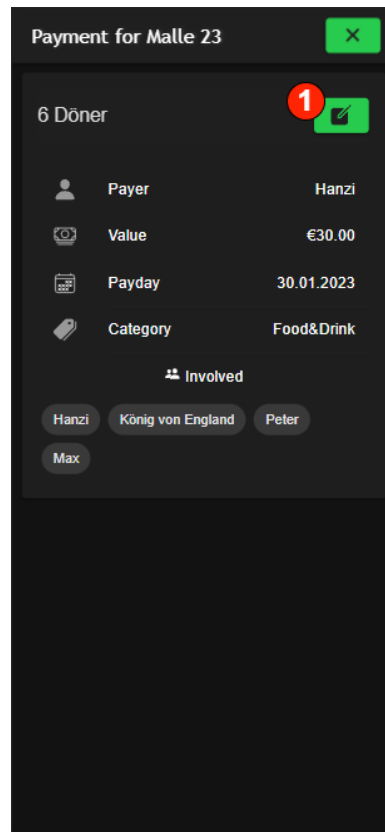


Abbildung 7: Payment-Details

Auf dieser Seite sieht der Nutzer alle Informationen zu einer Zahlung.

- ① Wechseln in den Bearbeitenmodus (Siehe 7.1.8).

### 7.1.8 Details einer Zahlung beim Erstellen/Bearbeiten

Payment for Malle 23

Title: 6 Döner

Payer: Hanzi

Currency: Euro

Value: 30

Payday: 30. Jan. 2023

Category: Food&D...

Involved

Hanzi, König von England, Peter, Max

+ (Add button)

+ (Camera icon)

SAVE

Abbildung 8: Payment-Details (Edit-Mode)

Auf dieser Seite kann der Nutzer die Informationen zu einer Zahlung erstellen oder bearbeiten.

- ① Titel der Zahlung
- ② Dropdown mit allen Reisebeteiligten zur Auswahl des Zahlers. Der Ersteller einer Zahlung ist hier vorausgewählt.
- ③ Dropdown für die Wahl der Währung. Die Standardwährung der Reise ist beim Erstellen vorausgewählt.
- ④ Input für den Betrag der Zahlung.
- ⑤ Öffnet Modal, um den Zahltag auszuwählen.
- ⑥ Dropdown zur Auswahl der Zahlungskategorie.
- ⑦ Durch den x-Button können Nutzer als Zahlungsverursacher entfernt werden.
- ⑧ Mit dem Plus-Button können einzelne Nutzer oder alle Nutzer hinzugefügt werden. Zusätzlich gibt es die Möglichkeit alle Nutzer zu entfernen.
- ⑨ Button um die Kamera zu öffnen und um ein Bild der Rechnung festzuhalten.
- ⑩ Button zum Speichern des Eintrags.



### 7.1.9 Schuldenansicht bei Guthaben

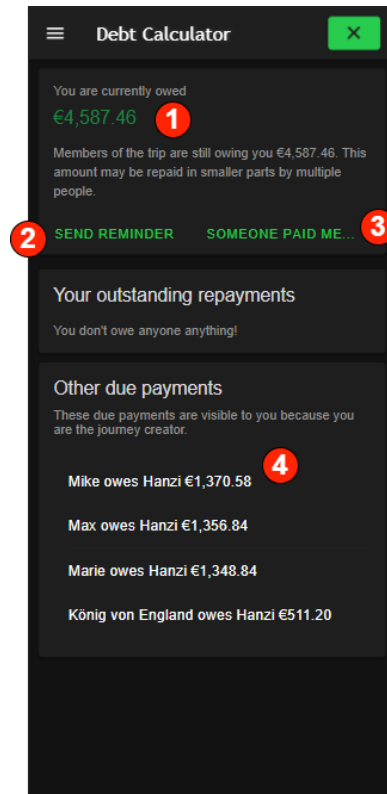


Abbildung 9: Debt-Calculator (Owed)

Auf dieser Seite sieht der Nutzer sein Guthaben bzw. seine Schulden gegenüber anderen (Siehe 7.1.10).

- ① Der Betrag, was einem die Mitreisenden noch schulden.
- ② Sendet eine Push-Nachricht als Erinnerung an alle Schuldner.
- ③ Öffnet Zahlung erstellen, um den gezahlten Betrag festzuhalten.
- ④ Als Reiseersteller hat man zusätzlich die Übersicht über alle offene Schulden einer Reise.

### 7.1.10 Schuldenansicht bei Schulden

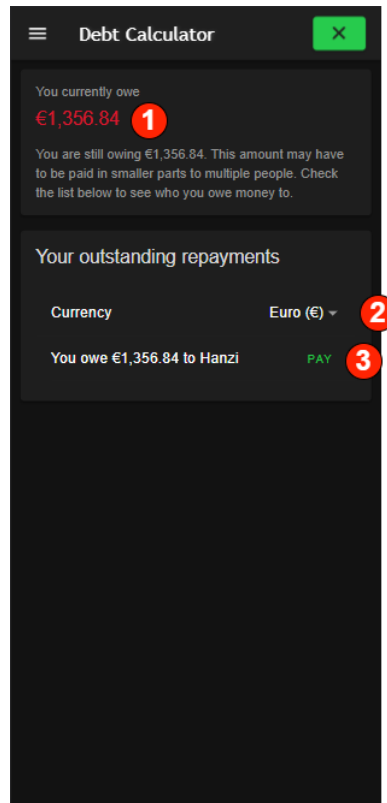


Abbildung 10: Debt-Calculator (Owe)

- ① Der Betrag, was man den Mitreisenden noch schuldet.
- ② Dropdown zur Auswahl der Schuldenwährung.
- ③ Öffnet Zahlung erstellen mit vor gefüllten Werten, um die Übermittlung einer Zahlung festzuhalten.

### 7.1.11 Einladungsmodal

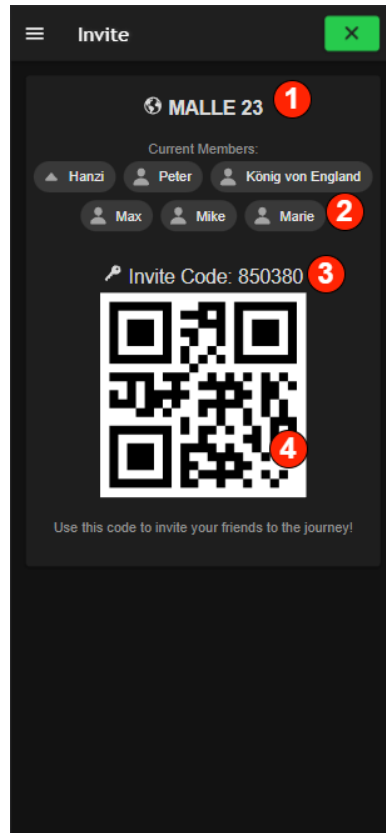


Abbildung 11: Invite-Modal

Auf dieser Seite erhält der Reiseersteller alle Informationen, um weitere Nutzer in seine Reise einzuladen.

- ① Title der Reise für die eingeladen wird.
- ② Liveansicht aller der Reise beigetretenen Nutzer.
- ③ Invitecode zur Eingabe auf der Reisebeitrittseite um der Reise beizutreten.
- ④ Der Invitecode als QR-Code zum einfachen Scannen auf der Reisebeitrittseite um der Reise beizutreten.

### 7.1.12 Optionen

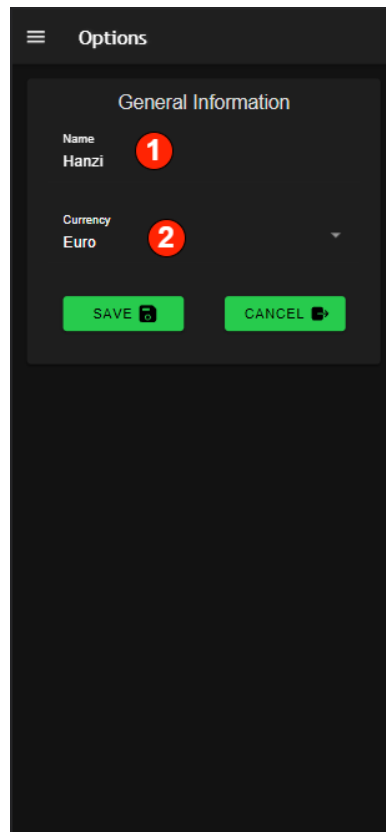


Abbildung 12: Optionen

Auf dieser Seite kann der Nutzer seine Daten anpassen.

- ① Ändern des Nutzernamens.
- ② Anpassen der Wunschwährung in der Schulden angezeigt werden sollen.

