Prüfungsleistung im Modul:
DLMCSPSE01_D - Projekt: Software Engineering

Internationale Hochschule Fernstudium

Studiengang: M. Sc. Informatik

Projektdokumentation

Fabian Lorenz

Betreuungsperson: Prof. Dr. Markus Kleffmann

Abgabedatum: 27. Dezember 2024

1 Projektübersicht

„FairMoney“ ist eine Webanwendung, die es Gruppen ermöglicht, ihre Ausgaben und Abrechnungen effizient und transparent zu verwalten. Sie richtet sich an Gruppen ab zwei Personen, die gemeinsam Aktivitäten unternehmen und dabei Zahlungen für die Gruppe tätigen, die anschließend ausgeglichen werden sollen. Die Anwendung eignet sich für eine Vielzahl von Gruppen, wie Freundeskreise, Kolleg*innen oder Sportvereine.

Die Hauptfunktionen von FairMoney umfassen die Möglichkeit, Gruppen zu gründen, Ausgaben zu erfassen und Abrechnungen durchzuführen. Nach der Abrechnung werden die Zahlungen zwischen den Mitgliedern so verteilt, dass möglichst wenige Transaktionen notwendig sind. Alle Ausgaben und Abrechnungen sind in einer chronologischen Historie einsehbar. Durch Funktionen wie permanente Gruppen und die Möglichkeit, PayPal.me-Links zu hinterlegen, wird die Benutzerfreundlichkeit der Anwendung weiter verbessert.

Technisch basiert die Anwendung auf einer 3-Schichten Architektur, wobei ein Web-Frontend mit einem Backend kommuniziert. Die Daten werden in einer Datenbank gespeichert, auf die das Backend zugreift. Die Anwendung wird „Mobile-First“ optimiert, um eine benutzerfreundliche Nutzung auf Smartphones zu gewährleisten.

2 Risikomanagement

In diesem Kapitel werden technische und nicht-technische Projektrisiken identifiziert, bewertet und entsprechende Gegenmaßnahmen festgelegt.

Zur Identifizierung der Risiken wird auf Erfahrungen aus früheren Projekten sowie auf den Input der Stakeholder*innen zurückgegriffen, um mögliche Risiken zu analysieren. Die Bewertung erfolgt anschließend nach einer in der Industrie weit verbreiteten Methode, die die Parameter Eintrittswahrscheinlichkeit und Auswirkungsschwere einbezieht (Pathrosel, 2024, S. 273).

Der Risikofaktor (RF) wird dabei auf Basis der Eintrittswahrscheinlichkeit (W) und der Schwere der Auswirkungen (S) durch folgende Formel berechnet (Pathrosel, 2024, S. 273):

$$RF = W + S - W * S$$

Sowohl die Eintrittswahrscheinlichkeit als auch die Schwere der Auswirkungen werden dabei auf einer Skala von 0 bis 1 bewertet (Pathrosel, 2024, S. 273). Die folgende Tabelle zeigt die Bedeutung des Risikofaktors im jeweiligen Kontext:

Tab. 1: Risikofaktoren

Risikofaktor	**Risikoeinstufung**
0-0,3	Geringes Risko
0,4-0,6	Mittleres Risiko
0,7-1	Hohes Risiko

Quelle: Eigene Darstellung auf Basis von Pathrosel, 2024, S. 273.

Für die folgende Bewertung sollen die folgenden Bewertungsstufen genutzt werden:

- Nicht vorhanden = 0
- Gering = 0,2
- Mittel = 0,5
- Hoch = 0,7
- Sehr hoch = 1

Die identifizierten und bewerteten Risiken werden je nach Schweregrad durch geeignete Maßnahmen adressiert. Dabei stehen die folgenden Optionen zur Verfügung:

- Vermeidung des Risikos,
- Minderung des Risikos,
- Kontrolle des Risikos,
- Akzeptanz des Risikos,
- Übertragung des Risikos auf Dritte (Pathrosel, 2024, S. 275).

Technische Risiken

Risiko	Fehlerhafte Optimierung der Webanwendung für verschiedene Zugriffe (Desktop-/Tablet-/Smartphone-Browser).	
Eintrittswahrscheinlichkeit Gering	Schwere der Auswirkungen Gering	Risikofaktor Geringes Risko
Gegenmaßnahme	Akzeptanz des Risikos Die Anwendung wird für Smartphones optimiert, eventuell nicht optimale Optimierungen für weitere Bildschirmgrößen werden akzeptiert.	

Risiko	Mögliche Performance-Einbußen bei sehr komplexen Berechnungen.	
Eintrittswahrscheinlichkeit Mittel	Schwere der Auswirkungen Gering	Risikofaktor Mittleres Risiko
Gegenmaßnahme	Kontrolle des Risikos Die maximale Gruppengröße wird auf 20 Mitglieder*innen festgelegt, um die maximale Komplexität der Berechnungen zu kontrollieren.	

Risiko	Die Kommunikation zwischen den Gruppenmitglieder*innen für die Abrechnung ist unzuverlässig.	
Eintrittswahrscheinlichkeit Gering	Schwere der Auswirkungen Sehr hoch	Risikofaktor Hohes Risiko
Gegenmaßnahme	Übertragung des Risikos auf Dritte Anwender*innen sind selbst für die Verteilung der Informationen über einen Messenger verantwortlich, die Anwendung generiert nur einen vorbereiteten Text für die Versendung.	

Zeitliche Risiken

Risiko	Mangel an Zeitressourcen aufgrund der Umsetzung des Projekts im Zuge eines Teilzeitstudiums.	
Eintrittswahrscheinlichkeit Mittel	Schwere der Auswirkungen Mittel	Risikofaktor Hohes Risiko
Gegenmaßnahme	Minderung des Risikos Bei der Planung des Projekts werden bereits die eingeschränkten Zeitressourcen berücksichtigt, um eine unerwartete Verzögerung zu vermeiden.	

Risiko	Unerwartete technische Probleme könnten zu Verzögerungen im Projektzeitplan führen.	
Eintrittswahrscheinlichkeit Sehr hoch	Schwere der Auswirkungen Mittel	Risikofaktor Hohes Risiko
Gegenmaßnahme	Minderung des Risikos Bei der Planung des Projekts werden bereits Pufferzeiträume berücksichtigt, um eine unerwartete Verzögerung zu vermeiden.	

Sicherheitsrisiken

Risiko	Offenlegung von Daten kann sensible Nutzerdaten offenbaren.	
Eintrittswahrscheinlichkeit Mittel	Schwere der Auswirkungen Sehr hoch	Risikofaktor Hohes Risiko
Gegenmaßnahme	Vermeidung des Risikos In der Anwendung werden keine sensiblen Nutzerdaten gespeichert. Der Nutzer wird außerdem darauf hingewiesen, dass für die Benutzer keinen personenbezogenen Daten gepflegt werden dürfen.	

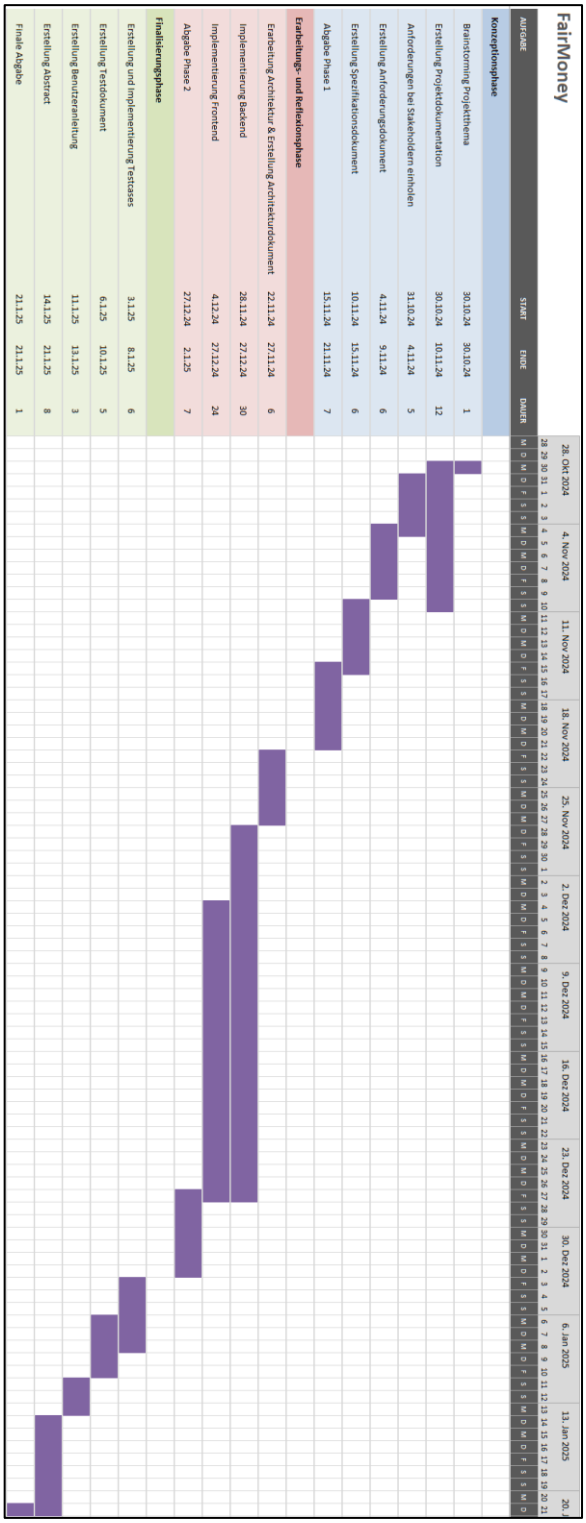
Benutzerakzeptanz

Risiko	Die Anwendung könnte von den Nutzer*innen nicht wie erwartet angenommen werden, was zu einer geringen Nutzung führen könnte.	
Eintrittswahrscheinlichkeit Mittel	Schwere der Auswirkungen Hoch	Risikofaktor Hohes Risiko
Gegenmaßnahme	Minderung des Risikos Es wird bereits in frühen Phasen der Softwareentwicklung Benutzerfeedback eingeholt, um sicherzustellen, dass die Anwendung den Bedürfnissen der Nutzer entspricht.	

3 Zeitplanung

Im Folgenden wird der Zeitplan für das Projekt vorgestellt. Der Projektstart ist am 30.10.2024 und die finale Abgabe ist für den 21.01.2025 geplant. Das Projekt gliedert sich in die drei Phasen „Konzeption“, „Erarbeitung und Reflexion“ sowie „Finalisierung“. Der Abschluss jeder Phase stellt einen wichtigen Meilenstein dar. Für die Rückmeldungen des Tutors nach den Phasen 1 und 2 wurde jeweils ein Puffer von einer Woche eingeplant, um den Feedbackprozess optimal zu berücksichtigen.

Abb. 1: Projektplan



Quelle: Eigene Darstellung

4 GitHub Link

Der entwickelte Code sowie alle erstellten Artefakte können in GitHub in folgendem Repository eingesehen werden:

https://github.com/1Joghurt/FairMoney.

Literaturverzeichnis

Pathrose, P. (2024). *ADAS and Automated Driving - Systems Engineering.* SAE International.

Prüfungsleistung im Modul:
DLMCSPSE01_D - Projekt: Software Engineering

Internationale Hochschule Fernstudium

Studiengang: M. Sc. Informatik

Anforderungsdokument

Fabian Lorenz

Betreuungsperson: Prof. Dr. Markus Kleffmann

Abgabedatum: 27. Dezember 2024

1 Stakeholder

Die Zielgruppe der Anwendung „FairMoney“ umfasst Gruppen ab zwei Personen, die gemeinsame Aktivitäten durchführen, bei denen einzelne Mitglieder Zahlungen für die gesamte Gruppe übernehmen, die im Nachgang ausgeglichen werden sollen.

Darüber hinaus richtet sich die Anwendung an Gruppen, die nicht nur einmalige Aktivitäten organisieren, sondern auch regelmäßige, gemeinschaftliche Ausgaben haben und dabei eine fortlaufende Übersicht über den finanziellen Ausgleich behalten möchten.

Die Anwendung ist nicht auf eine bestimmte Art von Gruppen beschränkt und kann vielseitig eingesetzt werden. Die konkreten Zielgruppen sind somit beispielsweise Freundeskreise, Gruppen von Arbeitskolleg*innen oder Sportvereine. Ein entscheidender Aspekt für die Zielgruppen der Anwendung ist jedoch das gegenseitige Vertrauen innerhalb der Gruppe, da alle Mitglieder eigenständig Ausgaben verwalten und Abrechnungen durchführen können.

2 Funktionale Anforderungen

Im Folgenden werden die funktionalen Anforderungen an die Anwendung definiert und mithilfe von User Stories und einem UML Use-Case Diagramm dokumentiert.

Es werden vier Rollen von Nutzer unterschieden, wobei eine natürliche Person je nach Tätigkeit mehrere Rollen einnehmen kann:

- **Gruppenersteller*in:**
  Personen, die die Anwendung nutzen, um eine Gruppe zu gründen. Sie informieren die anderen Mitglieder über die neu erstellte Gruppe.
- **Regelnutzer*in:**
  Personen, die die Anwendung regelmäßig nutzen, um Zahlungen der Gruppe einzutragen.
- **Abrechnende Nutzer*in**
  Personen, die in der Anwendung eine Abrechnung durchführen, um die Ausgaben der Gruppe ausgleichen. Sie veranlassen die Verrechnung der Zahlungen und informieren die anderen Mitglieder der Gruppe über die Abrechnung.
- **Zahlende Nutzer*in**
  Personen, die nach einer Abrechnung eine Zahlung an ein anderes Mitglied der Gruppe leisten müssen. Sie sind dafür verantwortlich, ihren Anteil an den Gesamtausgaben zu begleichen.

Gruppenerstellung

- Als Gruppenersteller*in möchte ich eine Gruppe erstellen können, um Aktivitäten und Ausgaben gemeinsam zu verwalten.
- Als Gruppenersteller *in möchte ich für jede Person in der Gruppe einen Namen pflegen, um alle Teilnehmenden eindeutig identifizieren zu können.
- Als Gruppenersteller *in möchte ich optional für jede Person einen PayPal.me-Link hinterlegen, damit Abrechnungen einfacher und schneller abgewickelt werden können.
- Als Gruppenersteller *in möchte ich nach dem Erstellen einer Gruppe eine vorgenerierte Nachricht bekommen, die ich für die Einladung der anderen Personen in einem externen Messenger nutzen kann.

Ausgaben hinzufügen

- Als Regelnutzer*in möchte ich Ausgaben in der Gruppe anlegen und dabei angeben können, wer die Zahlung übernommen hat und wer an der Abrechnung beteiligt sein soll, damit die Kosten gerecht verteilt werden.

Abrechnung durchführen

- Als Abrechnende Nutzer*in möchte ich jederzeit eine Abrechnung durchführen können, bei der alle Ausgaben so verrechnet werden, dass möglichst wenige Transaktionen zwischen den Mitgliedern erforderlich sind.
- Als Abrechnende Nutzer*in möchte ich nach dem Erstellen einer Abrechnung eine vorgenerierte Nachricht bekommen, die alle wichtigen Informationen zur Abrechnung enthält, damit ich sie in einem externen Messenger an die anderen Gruppenmitglieder versenden kann.
- Als Zahlende Nutzer*in möchte ich jederzeit bestehende Abrechnungen einsehen können.

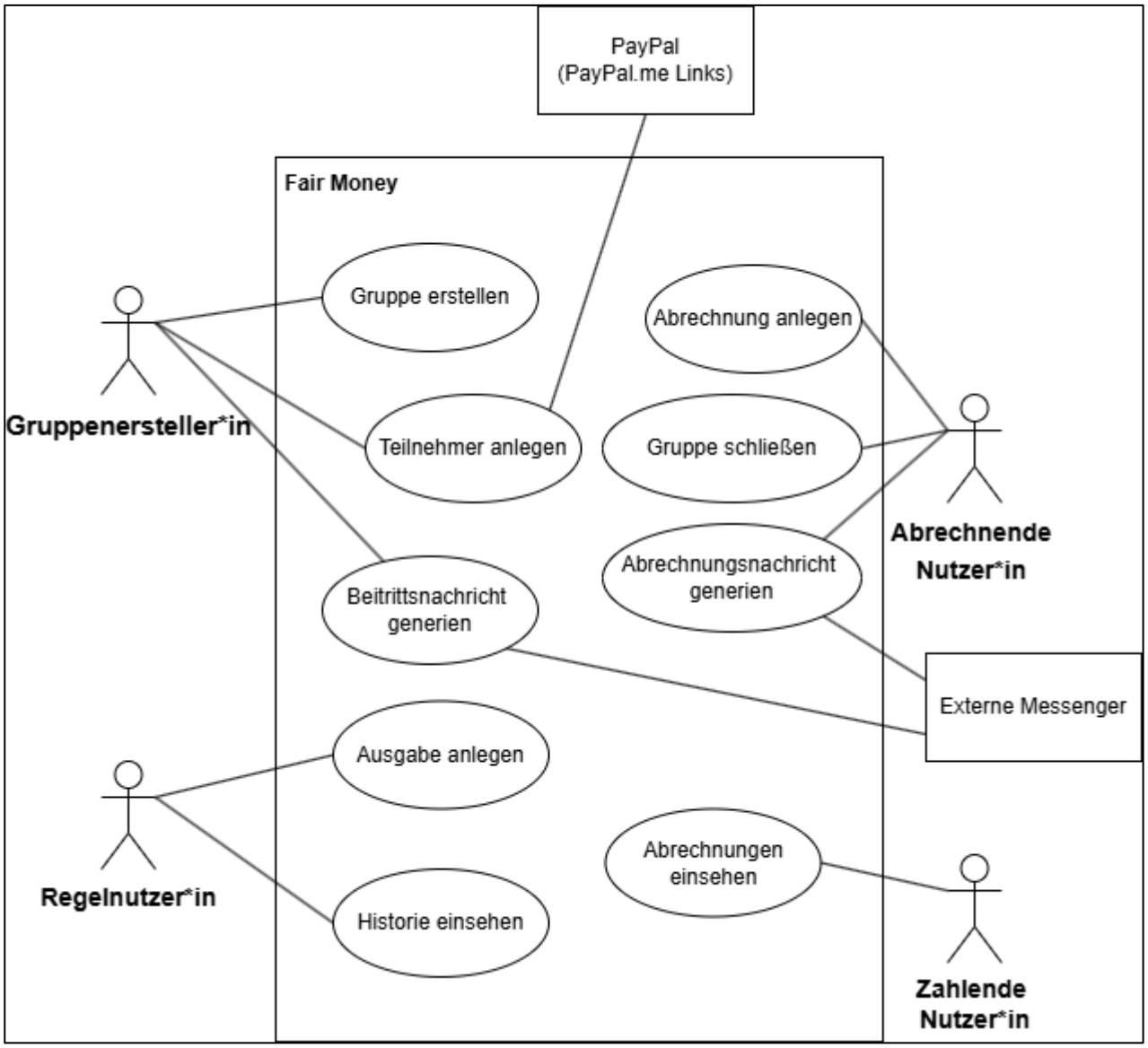
Gruppenfortführung oder -schließung

- Als Abrechnende Nutzer*in möchte ich bei jeder Abrechnung entscheiden können, ob die Gruppe weiterbesteht oder geschlossen wird.

Chronologische Historie

- Als Regelnutzer*in möchte ich alle Ausgaben und Abrechnungen in einer chronologischen Historie sehen, um jederzeit den Überblick über die getätigten Transaktionen und Abrechnungen zu behalten.

Abb. 1: UML-Use-Case-Diagramm



Quelle: Eigene Darstellung

3 Nichtfunktionale Anforderungen

Im Folgenden werden die wichtigsten nichtfunktionalen Anforderungen an das System definiert. Die Anforderungen sind nach den Kategorien Performance, Sicherheit, Kompatibilität und Benutzerfreundlichkeit gegliedert.

Performance

- Die Anwendung soll innerhalb von 2 Sekunden auf Benutzereingaben zur Erstellung von Gruppen oder Ausgaben oder zur Anzeige der Historie reagieren.
- Die Abrechnungsvorgänge sollen unabhängig von der Gruppengröße und Transaktionsanzahl innerhalb von maximal 5 Sekunden abgeschlossen sein.

Sicherheit

- Der Zugriff auf einzelne Gruppen soll ausschließlich über eine eindeutig generierte, nicht erratbare ID erfolgen, um unbefugten Zugang zu verhindern.
- Eine Registrierung oder Anmeldung ist für die Anwendung nicht erforderlich. Da die Anwendung für Gruppen mit gegenseitigem Vertrauen konzipiert ist, können alle Mitglieder ohne individuelle Authentifizierung Ausgaben hinzufügen und verwalten.

Kompatibilität

- Da die Anwendung hauptsächlich mobil während Aktivitäten auf Smartphones genutzt wird, soll sie nach dem „Mobile-First“-Prinzip primär für aktuelle Smartphones optimiert sein.

Benutzerfreundlichkeit

- Die Anwendung soll selbsterklärend sein und eine intuitive Benutzeroberfläche bieten, sodass sie ohne zusätzliche Anleitung problemlos bedienbar ist.

4 Nicht-unterstützte Funktionalitäten

In diesem Abschnitt wird kurz zusammengefasst, welche funktionalen und nicht-funktionalen Anforderungen explizit nicht mit dem Umfang dieses Projektes abgedeckt sind:

- Die Funktionalität Gruppen, Ausgaben und Abrechnungen nach ihrer Erstellung erneut zu bearbeiten wird nicht als Teil dieser Arbeit implementiert.
- Die Anwendung ist nicht an gängige Messenger- oder E-Mail-Systeme angebunden; generierte Nachrichten müssen von Nutzer*innen manuell versendet werden.
- Es wird keine technische Authentifizierungsschranke (wie z.B. Login mit Benutzername und Passwort) implementiert. Jede Person mit Kenntnis der eindeutigen URL kann sich mit der Gruppe verbinden und eine Identität auswählen.
- Die Anwendung wird nicht für generische Bildschirmgrößen optimiert; der Fokus liegt auf der Nutzung auf Smartphone-Bildschirmen.

5 Glossar

Im Folgenden sollen die wichtigsten projektspezifischen Begriffe und Abkürzungen erläutert werden.

Gruppe

Repräsentiert eine Abrechnungsgemeinschaft, in der Nutzer*innen gemeinsam Ausgaben verwalten.

Mitglieder/Teilnehmer*innen/Nutzer*innen

Personen, die die Anwendung nutzen und dafür in einer Gruppe angelegt wurden. Für diese Personen wurden Name und optional PayPal.me-Links gepflegt.

Abrechnung

Vorgang, bei dem alle bisherigen Ausgaben verrechnet und Ausgleichstranskationen erstellt werden.

PayPal.me

PayPal.me ist ein Service von PayPal, der es ermöglicht, Zahlungen über einen personalisierten Link zu empfangen, ohne dass der Zahler die E-Mail-Adresse der Empfängerin oder des Empfängers kennen muss (PayPal, 2024).

Mobile-First

Design-Ansatz, bei dem die Benutzererfahrung und das Layout einer Website oder Anwendung zunächst für mobile Geräte optimiert werden, bevor die Anpassung an größere Bildschirmgrößen erfolgt.

Literaturverzeichnis

PayPal. (2024). *Link teilen und noch einfacher Geld empfangen*. https://www.paypal.com/de/digital-wallet/send-receive-money/paypal-me

INTERNATIONALE HOCHSCHULE

Prüfungsleistung im Modul:
DLMCSPSE01_D - Projekt: Software Engineering

Internationale Hochschule Fernstudium

Studiengang: M. Sc. Informatik

Spezifikationsdokument

Fabian Lorenz

Betreuungsperson: Prof. Dr. Markus Kleffmann

Abgabedatum: 27. Dezember 2024

1 Datenmodell

Das Datenmodell von „FairMoney“ basiert auf dem Geschäftsobjekt „Group“, welches eine Abrechnungsgruppe repräsentiert. Jede Gruppe besitzt einen Titel und speichert den Status, ob sie abgeschlossen ist.

Eine „Group“ kann eine unbegrenzte Anzahl an „Payment“-Objekten umfassen, welche die einzelnen Zahlungen darstellen, die von den Gruppenmitgliedern getätigt wurden. Für jede Zahlung wird die Höhe der Ausgabe, die zahlende Person sowie die beteiligten Benutzer*innen erfasst, die die Ausgabe anteilig begleichen. Zudem wird vermerkt, ob und durch welche Abrechnung die Ausgabe bereits ausgeglichen wurde.

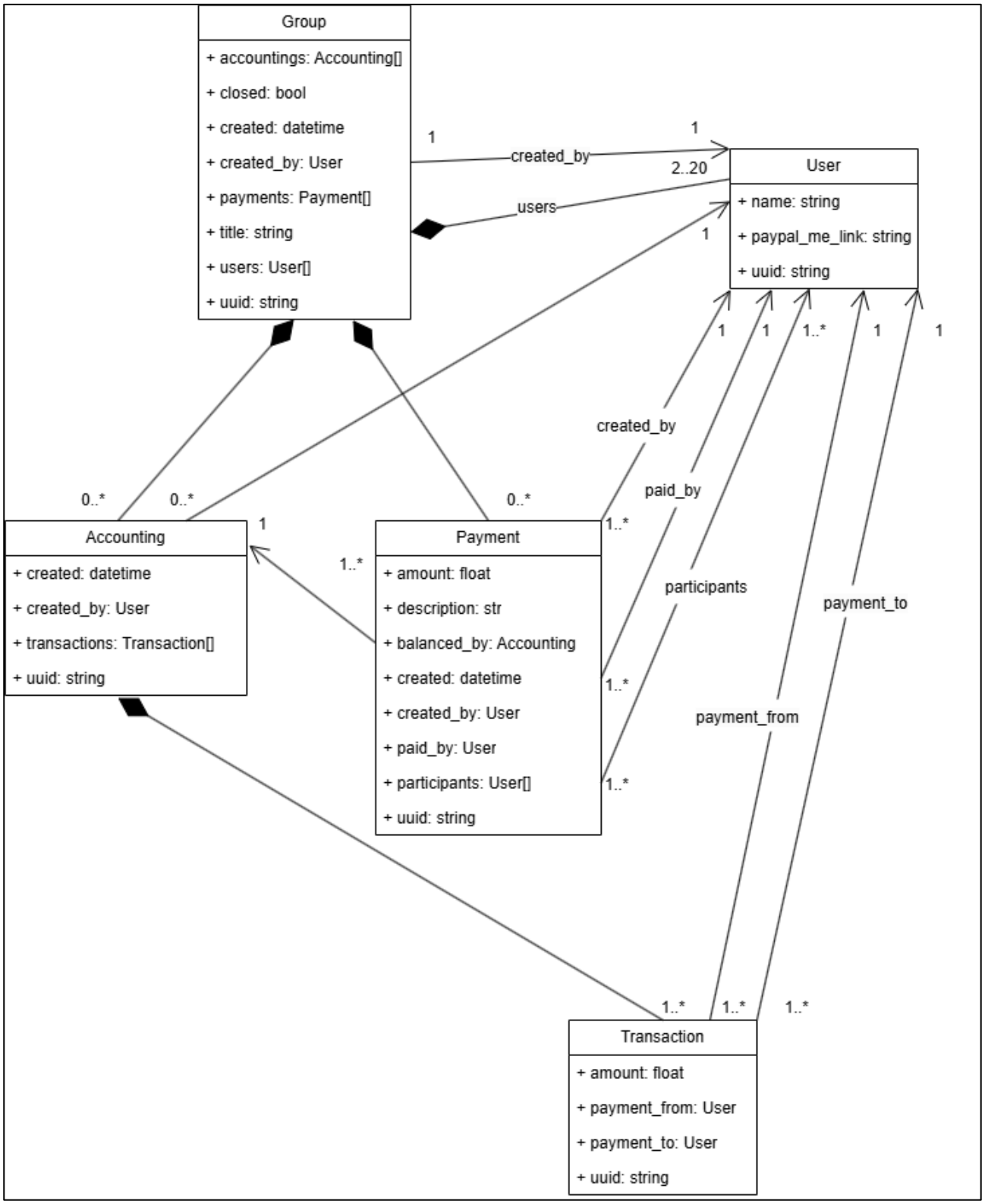
Die erwähnten Abrechnungen werden durch ein Geschäftsobjekt „Accounting“ dargestellt, die eine Menge verknüpfter „Transaction“-Objekte enthält; diese repräsentieren die einzelnen Transaktionen, die für die Abrechnung notwendig war.

Das Geschäftsobjekt „User“ repräsentiert die Benutzer*innen und speichert neben dem Namen gegebenenfalls den PayPal.me-Link, sofern dieser hinterlegt wurde.

Aus Gründen der Nachvollziehbarkeit soll bei Gruppen, Zahlungen und Abrechnungen festgehalten werden, wann und durch wen sie erstellt wurden.

Im Folgenden werden das Datenmodell und die Beziehungen in einem UML-Klassendiagramm dargestellt.

Abb. 1: UML-Klassendiagramm



Quelle: Eigene Darstellung

2 Geschäftsprozesse

Im Folgenden werden die wichtigsten Geschäftsprozesse vorgestellt. Jeder Prozess wird zunächst mit einer kurzen Beschreibung eingeführt, anschließend wird der Ablauf Schritt für Schritt erläutert. Abschließend werden die zentralen Geschäftsregeln für jeden Prozess dargestellt. Für den Prozess „Erstellen von Abrechnungen“ wird der Prozess darüber hinaus mithilfe eines UML-Aktivitätendiagramms visualisiert.

Geschäftsprozess: Erstellen einer Gruppe

Das Erstellen einer Gruppe ist der erste Schritt für Nutzer*innen, um die Anwendung „FairMoney“ für gemeinsame Abrechnungen zu nutzen. In diesem Prozess gibt eine Nutzerin oder ein Nutzer die notwendigen Informationen an und informiert die anderen Nutzer*innen über die neue Gruppe.

Ablauf:

1. Die Nutzer*in öffnet die Startseite der Anwendung und bestätigt, dass eine Gruppe erstellt werden soll.
2. Es wird ein Gruppenname festgelegt, der die Gruppe eindeutig kennzeichnet (z. B. „WG-Kasse“ oder „Urlaub Mallorca“).
3. Nun muss die Ersteller*in ihren eigenen Namen und optional einen PayPal.Me-Link hinterlegen.
4. Anschließend fügt die Ersteller*in 1 bis 19 weitere Teilnehmende zur Gruppe hinzu und pflegt dafür einen Namen. Optional kann pro User ein PayPal.Me-Link hinterlegt werden.
5. Nachdem die Gruppe erstellt wurde, wird ein eindeutiger Link für die neue Gruppe generiert und dem Nutzer bereitgestellt. Zusätzlich wird eine Nachricht generiert, die den Link enthält und vom Benutzer über Messenger oder E-Mail verteilt werden kann.
6. Anschließend ist die Gruppe aktiv und für die Erfassung von Ausgaben und Abrechnungen bereit.

Geschäftsregeln:

- Eine Gruppe muss aus mindestens zwei und maximal 20 Mitgliedern bestehen.
- Jede Gruppe muss einen Namen haben, damit die Mitglieder die Gruppe identifizieren können. Theoretisch sind mehrere Gruppen mit dem gleichen Namen erlaubt, da potenziell mehrere unabhängige Personen Gruppen für gleiche Aktivitäten erstellen (wie im Beispiel „Urlaub Mallorca“).
- Die Namen der Teilnehmer*innen innerhalb einer Gruppe müssen eindeutig sein, damit die einzelnen Namen eindeutig zu den Personen zuordenbar sind.

Geschäftsprozess: Hinzufügen von Ausgaben

Mitglieder einer Gruppe können Ausgaben erfassen, die sie für die gesamte Gruppe getätigt haben. Dieser Prozess sorgt dafür, dass alle Kosten innerhalb der Gruppe transparent und nachvollziehbar festgehalten werden. Jede Ausgabe wird dokumentiert und ist in der Gruppenhistorie einsehbar.

Ablauf:

1. Ein Mitglied öffnet die Gruppe mithilfe des gruppenspezifischen Links
2. Anschließend wählt die Benutzer*in aus, welche Identität sie annehmen möchte.
3. Das Mitglied drückt den Button zur Eingabe einer neuen Ausgabe.
4. Die Nutzer*in gibt den Betrag, den Grund der Ausgabe (z. B. „Lebensmittel“, „Miete“, „Eintrittstickets“) und den Zahlenden an.
5. Es wird festgelegt, für welche Mitglieder die Ausgabe getätigt worden ist (z. B. für alle oder nur für eine Teilgruppe).
6. Die Ausgabe wird gespeichert und für alle Mitglieder sichtbar in der Gruppenhistorie eingetragen.

Geschäftsregeln:

- Jede Ausgabe muss mit einem gültigen Betrag (numerisch und größer als 0) und einem spezifischen Zweck erfasst werden.
- Jede Ausgabe sollte automatisch mit einem Zeitstempel versehen werden, um die Reihenfolge der Einträge zu dokumentieren.
- Für jede Ausgabe wird festgehalten, wer die Ausgabe eingetragen hat (dies kann abweichen von der Person, die die Ausgabe tatsächlich getätigt hat).
- Grundsätzlich soll im Feld für den Zahlenden der aktuell angemeldete Nutzer vorausgewählt sein.

Geschäftsprozess: Erstellen von Abrechnungen

Der Abrechnungsprozess dient dazu, die finanziellen Ausgaben innerhalb der Gruppe auszugleichen. Die Anwendung errechnet auf Basis der erfassten Ausgaben die optimalen Transaktionen zwischen den Mitgliedern, um die Ausgaben mit minimalem Aufwand auszugleichen. Der Abrechnende bekommt eine vorgenerierte Nachricht zur Verteilung in externen Messengern.

Ablauf:

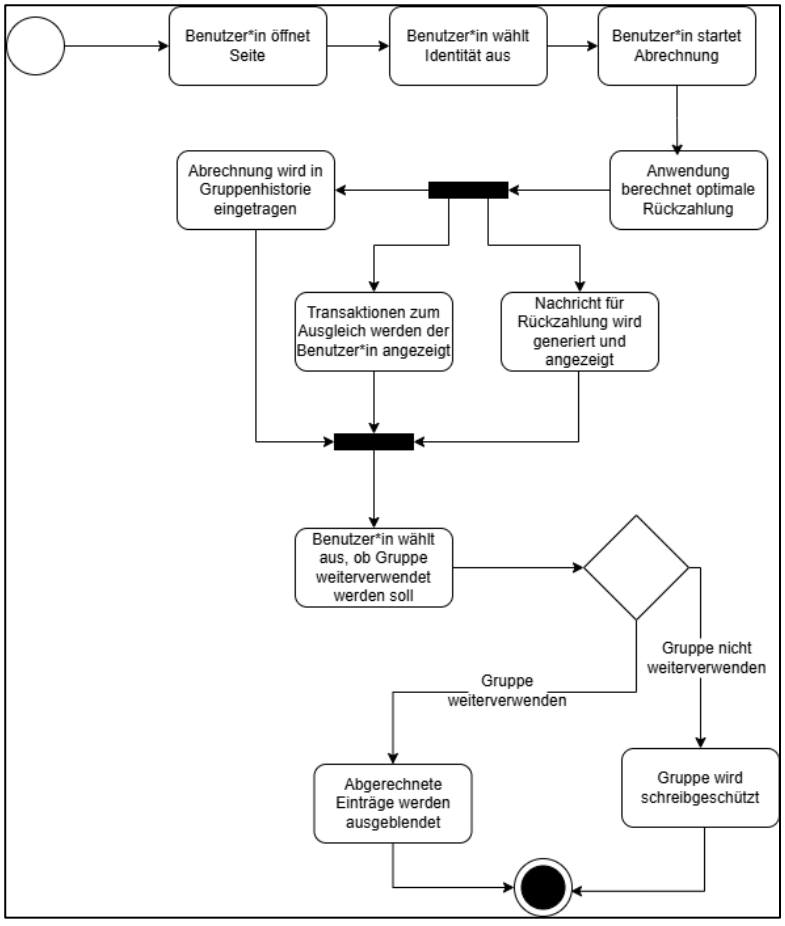
1. Ein Mitglied öffnet die Gruppe mithilfe des gruppenspezifischen Links
2. Anschließend wählt die Benutzer*in aus, welche Identität sie annehmen möchte.
3. Das Mitglied drückt den Button zur Erstellung einer neuen Abrechnung.
4. Die Anwendung berechnet den finanziellen Saldo für jedes Mitglied, basierend auf den eingetragenen Ausgaben und der jeweiligen Beteiligung.

5. Anschließend erstellt die Anwendung eine Liste von Ausgleichszahlungen, um alle Ausgaben gleichmäßig auszugleichen und zeigt sie dem Benutzer an. Zusätzlich wird eine Nachricht generiert, die die Ausgleichzahlungen auflistet und falls vorhanden, die PayPal.me-Links mit vorkonfiguriertem Link enthält.
6. Die Abrechnung wird als neuer Eintrag in der Gruppenhistorie gespeichert.
7. Die Nutzer*in wählt aus, ob die Gruppe nach der Abrechnung weiterhin genutzt wird.
8. Wurde ausgewählt, dass die Gruppe nicht weiterhin verwendet werden soll, ist die Gruppe nur mehr lesend zugreifbar. Falls die Gruppe weiterhin verwendet werden soll, ist sie weiterhin schreibbar, allerdings sind die abgerechneten Ausgaben nicht mehr sichtbar.

Geschäftsregeln:

- Die Abrechnung erfolgt auf Basis der zuletzt erfassten und bestätigten Ausgaben; Änderungen an den Ausgaben nach Beginn des Abrechnungsvorgangs werden für die Abrechnung nicht nachträglich berücksichtigt.
- Die Anwendung minimiert die Anzahl der Transaktionen, um den Prozess möglichst effizient zu gestalten.
- Die Ausgleichzahlungen werden auf ganze Cent-Beträge aufgerundet.

Abb. 2: UML-Aktivitätsdiagramm



Quelle: Eigene Darstellung

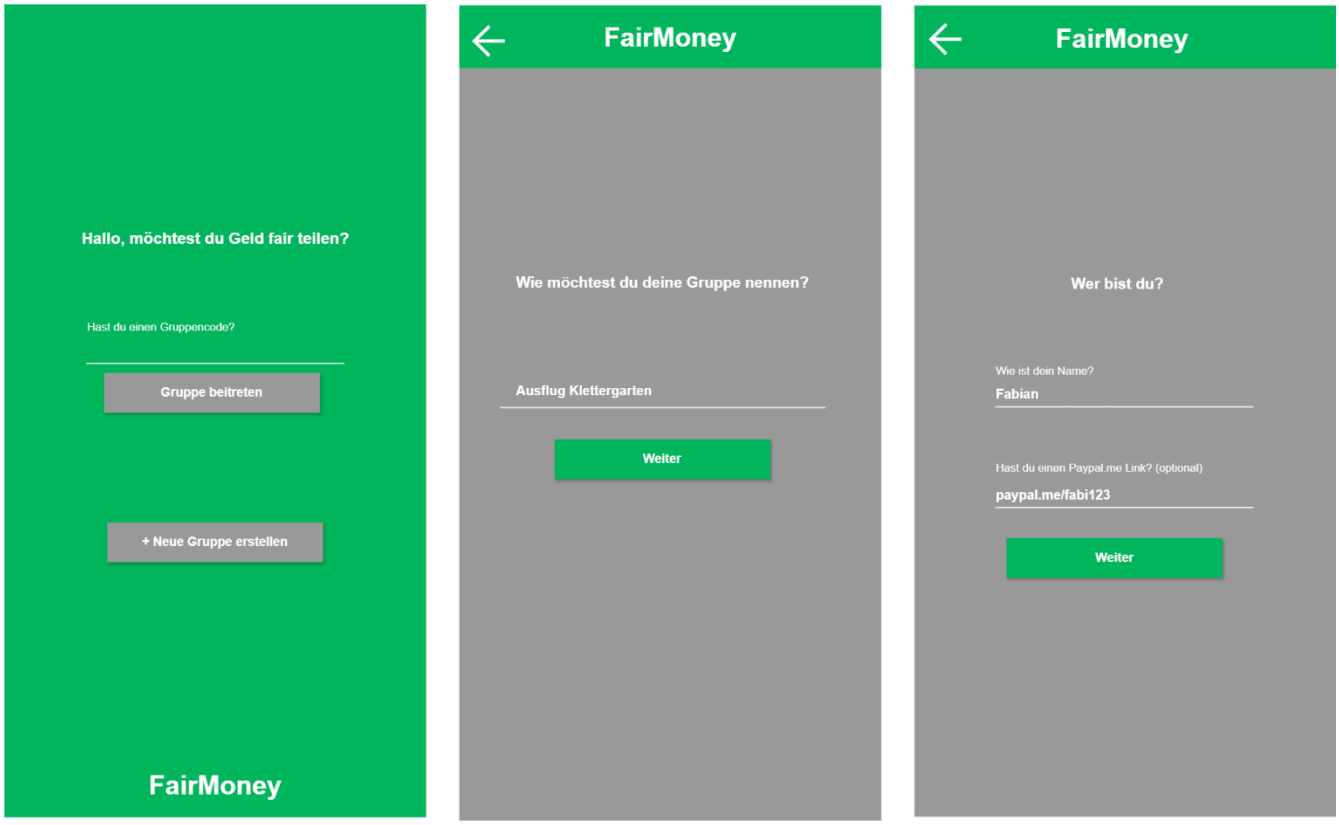
3 Systemschnittstellen

Schnittstellen zu externen Systemen sind in diesem Projekt nicht erforderlich. Die einzige relevante Verbindung nach außen betrifft den Versand von Einladungs- oder Abrechnungsnachrichten per Messenger oder E-Mail. Hierfür wird lediglich eine Vorlage bereitgestellt, die die Benutzer*innen bei Bedarf manuell in einem Messenger ihrer Wahl übertragen können. Es ist keine technische Umsetzung notwendig.

4 Benutzerschnittstellen

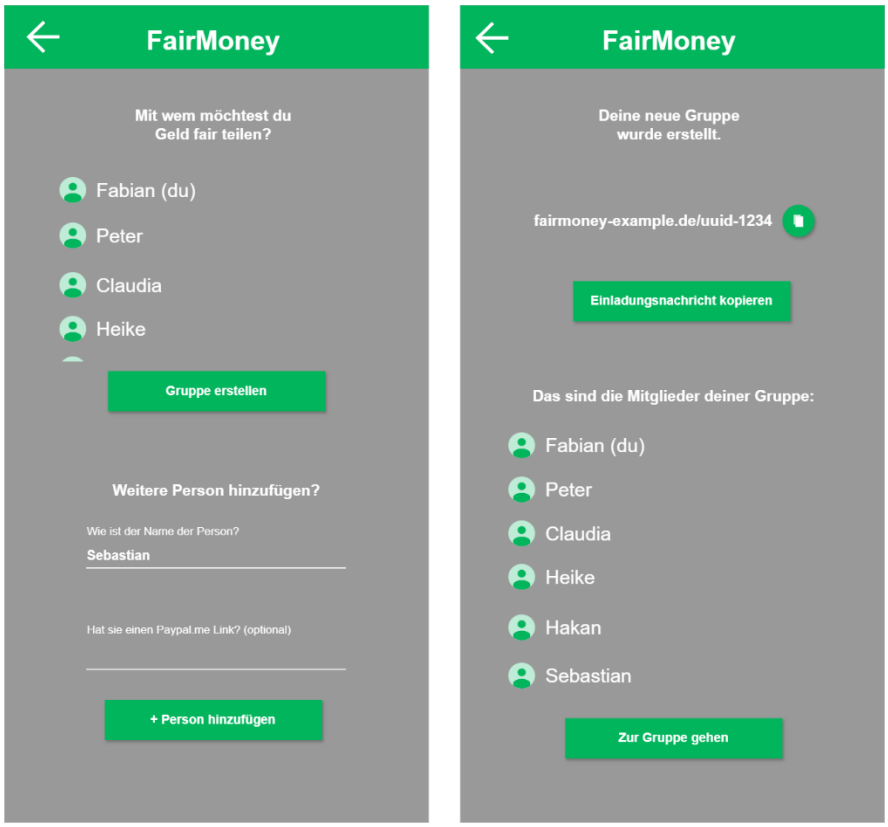
In diesem Abschnitt wird die grafische Benutzerschnittstelle, mit der der Benutzer interagiert vorgestellt. Um die Benutzerführung zu visualisieren, wurden verschiedene Bildschirmdesigns entwickelt. Die folgenden Erläuterungen umfassen die einzelnen Dialoge, den Dialogfluss sowie wichtige anwendungsspezifischen Eingaberegeln, wie etwa die Eingabevalidierungen.

Abb. 3: Bildschirmdesign Gruppenerstellung Teil 1



Quelle: Eigene Darstellung

Abb. 4: Bildschirmdesign Gruppenerstellung Teil 2



Quelle: Eigene Darstellung

Die Abbildungen 3 und 4 illustrieren den Ablauf für die Erstellung einer neuen Gruppe. Der erste Bildschirm, auch als „Landing Page“ bezeichnet, ist der Ausgangspunkt für alle Nutzenden, die keinen spezifischen Einladungslink zu einer Gruppe nutzen. Hier stehen den Nutzenden zwei Optionen zur Verfügung: entweder kann ein vorhandener Gruppencode eingegeben werden, um einer bestehenden Gruppe beizutreten, oder eine neue Gruppe kann erstellt werden. Falls ein Gruppencode eingegeben wird, wird dieser validiert, um sicherzustellen, dass er dem später im Entwicklungsprozess festgelegten Format entspricht.

Entscheiden sich Nutzende für die Erstellung einer neuen Gruppe, werden sie in einem weiteren Bildschirm zur Eingabe eines Gruppennamens aufgefordert, wobei diese Eingabe zwischen 3 und 30 Zeichen haben darf. Auf dieser Seite ist ebenfalls ein Zurückbutton in der linken oberen Ecke verfügbar, der wie auf allen folgenden Bildschirmen für den Rückschritt zum letzten Bildschirm genutzt werden kann.

Nach der Eingabe des Gruppennamens erfolgt ein Schritt, in dem die Nutzenden ihren eigenen Namen eingeben (1 bis 30 Zeichen) und optional einen PayPal.me-Link angeben können, der dem Format „paypal.me/<Name>“ folgen muss.

Auf dem vierten Bildschirm können weitere Mitglieder zur Gruppe hinzufügt werden. Die Eingabefelder und Validierungen für die neuen Mitglieder entsprechen denen des Gruppen-Erstellenden. In dieser Phase werden die bisherige Gruppenmitglieder in einer scrollbaren Liste

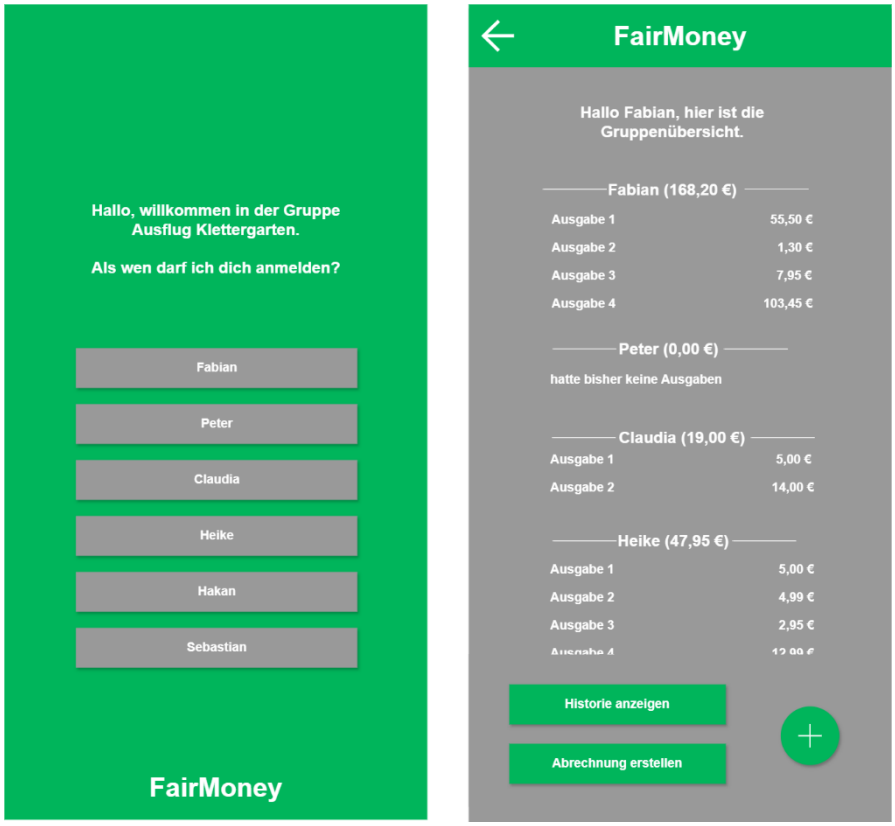
angezeigt. Sobald mindestens zwei Mitglieder in der Gruppe vorhanden sind, wird der Button „Gruppe erstellen“ aktiviert. Durch das Anklicken dieses Buttons wird die Gruppe final erstellt, und der Prozess führt zum abschließenden Bildschirm der Gruppenerstellung.

Der letzte Bildschirm der Gruppenerstellung zeigt den neu generierten Gruppenlink, der durch einen Kopier-Button direkt in die Zwischenablage übernommen werden kann. Zusätzlich wird ein Button bereitgestellt, um eine vorformulierte Einladungsnachricht zu kopieren, die den Link für den Beitritt zur Gruppe enthält. Diese Nachricht ist im folgenden Stil formuliert:

„Hallo ich habe die Gruppe „Ausflug Klettergarten“ bei FairMoney erstellt und möchte mit dir fair Geld teilen. Mach mit und trete mit dem Link *fairmoney-example.de/uuid-1234* bei. Bleib fair!“

Im unteren Bereich dieses Abschlussbildschirms wird eine Liste der hinzugefügten Mitglieder angezeigt, wodurch die Gruppenzusammensetzung nochmals sichtbar wird. Ergänzend steht hier ein Button zur Verfügung, der die Nutzenden direkt zur neu erstellten Gruppe weiterleitet. Damit endet der Prozess der Gruppenerstellung.

Abb. 5: Bildschirmdesign Gruppenübersicht



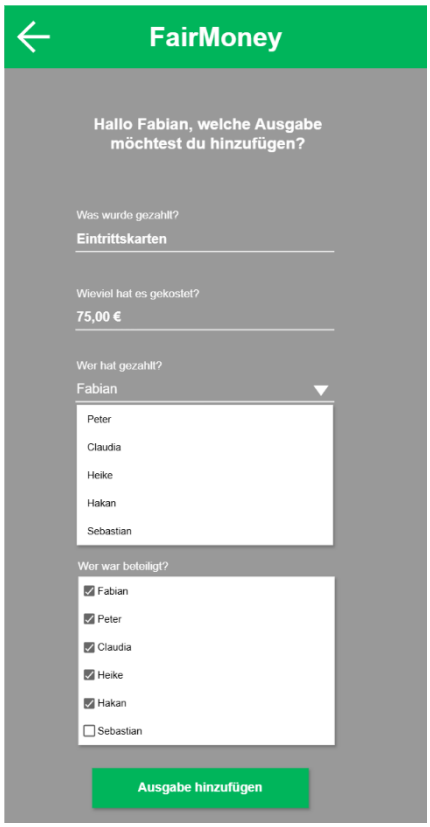
Quelle: Eigene Darstellung

Abbildung 5 zeigt die Startseite einer spezifischen Gruppe. Hier werden alle Mitglieder der Gruppe in einer Liste dargestellt, und die Nutzenden werden aufgefordert, sich selbst auszuwählen, um sich anzumelden.

Im nächsten Bildschirm befindet sich die zentrale Gruppenübersicht. Diese Übersicht präsentiert alle Gruppenausgaben in einer scrollbaren Liste, wodurch die Übersichtlichkeit auch bei einer hohen

Anzahl an Einträgen erhalten bleibt. Jede Person wird mit ihren individuellen Ausgaben und der Gesamtsumme dieser Ausgaben dargestellt.
Von der Gruppenübersicht aus haben die Nutzenden mehrere Navigationsoptionen: Sie können zur Historie wechseln, um vergangene Transaktionen oder Änderungen einzusehen, eine neue Abrechnung erstellen oder über das „+“-Symbol eine neue Ausgabe hinzufügen.

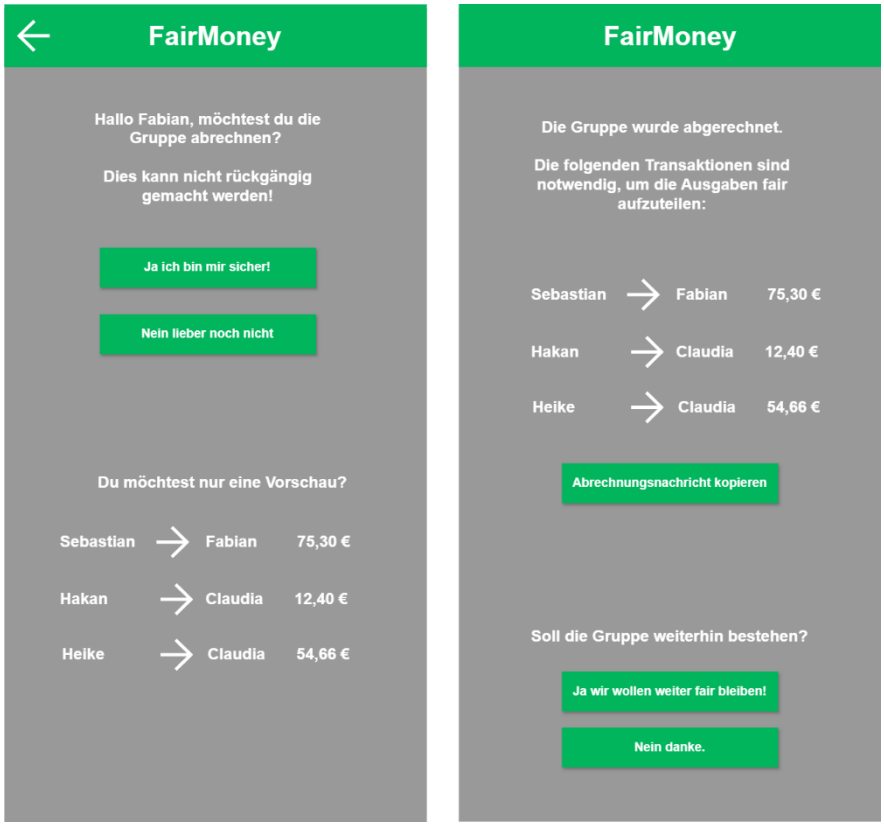
Abb. 6: Bildschirmdesign Ausgabe hinzufügen



Quelle: Eigene Darstellung

Die Abbildung 6 zeigt den Bildschirm zur Erstellung einer neuen Ausgabe, der nach einem Klick auf das „+“-Symbol im Hauptbildschirm erscheint. Auf diesem Bildschirm werden die Details zur Ausgabe eingetragen: Ein Titel, der zwischen 3 und 30 Zeichen umfassen und ein Betrag, der als positive Zahl eingegeben werden muss.
Zusätzlich muss festgelegt werden, wer die Zahlung getätigt hat. Hierfür steht ein Dropdown-Menü zur Verfügung, in dem die angemeldete Person standardmäßig vorausgewählt ist. Darüber hinaus wird in einem Multiselect-Dropdown bestimmt, auf welche Mitglieder die Ausgabe aufgeteilt werden soll, wobei standardmäßig alle Mitglieder ausgewählt sind.
Sobald alle erforderlichen Angaben gemacht wurden, wird der Button „Ausgabe hinzufügen“ aktiv. Bei Klick auf diesen, wird die Ausgabe hinzugefügt und der Nutzer zurück auf den Hauptbildschirm geleitet.

Abb. 7: Bildschirmdesign Gruppenabrechnung



Quelle: Eigene Darstellung

Abbildung 7 zeigt den Dialogablauf, der ausgelöst wird, wenn Nutzende im Hauptbildschirm auf den Button „Abrechnung erstellen“ klicken. Zunächst erscheint eine Bestätigungsabfrage, die die Nutzenden darauf hinweist, dass die Erstellung der Abrechnung nicht rückgängig gemacht werden kann. Zur Unterstützung bei dieser Entscheidung wird im unteren Bereich des Bildschirms eine Vorschau der aktuellen Abrechnung angezeigt, aus der hervorgeht, welcher Betrag an wen geschuldet wird. Entscheiden sich die Nutzenden gegen die Abrechnung, gelangen sie zurück zum Hauptbildschirm; wird die Abrechnung bestätigt, erfolgt der Wechsel zum nächsten Bildschirm, auf dem die Abrechnung durchgeführt wird.

Der folgende Bildschirm zeigt die erstellte Abrechnung mit den einzelnen erforderlichen Transaktionen, die in einer scrollbaren Liste dargestellt sind. Ein Button ermöglicht das Kopieren einer vorformulierten Nachricht, die die relevanten Transaktionen beschreibt.

Zur Veranschaulichung folgt eine Beispielnachricht, die auf den Daten aus dem Screenshot basiert. Hierbei wird angenommen, dass Fabian einen PayPal.me-Link eingerichtet hat, während Claudia dies nicht getan hat:

„Hallo ich habe die Gruppe „Ausflug Klettergarten“ bei FairMoney abgerechnet. Es ist an der Zeit die Ausgaben fair aufzuteilen. Dafür benötigen wir die folgenden Transaktionen:

Sebastian schickt Fabian 75,30 €. Dafür kannst du gerne den folgenden Link nutzen: *paypal.me/fabi123/75.30*

Hakan schickt Claudia 12,40 €.

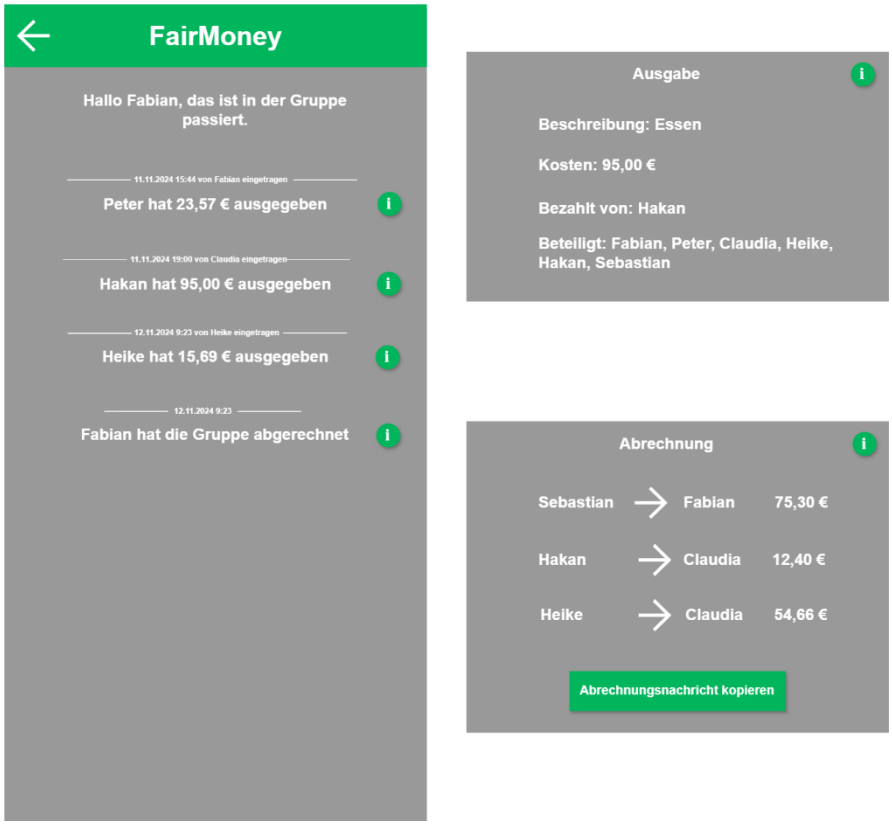
Heike schickt Claudia 54,66 €.

Falls du noch etwas nachschauen willst, findest du die Gruppe unter *fairmoney-example.de/uuid-1234*.

Danke dir fürs fair sein!“

Im unteren Teil des Bildschirms wird abgefragt, ob die Gruppe weiter bestehen bleiben soll. Bei Bestätigung dieser Option werden die Nutzenden zurück auf den Hauptbildschirm geleitet, wobei die bisherigen Ausgaben zurückgesetzt wurden. Entscheiden die Nutzenden jedoch, dass die Gruppe nicht weiter bestehen soll, kehren sie ebenfalls zum Hauptbildschirm zurück, jedoch ohne die Möglichkeit, neue Ausgaben oder Abrechnungen hinzuzufügen – die entsprechenden Buttons werden ausgeblendet.
Falls die Seite ohne ausdrückliche Bestätigung geschlossen wird, wird die Option „Gruppe bestehen lassen“ automatisch ausgewählt und die Gruppe bleibt weiterhin aktiv.

Abb. 8: Bildschirmdesign Historie



Quelle: Eigene Darstellung

Abbildung 8 zeigt den letzten Bildschirm der Anwendung, der die Historie der Gruppe darstellt. Auf diesem Bildschirm werden alle Ausgaben und Abrechnungen in einer scrollbaren Übersicht aufgelistet

Zu jedem Eintrag ist ein „i“-Button verfügbar, der bei Bedarf weitere Details in einem Popup-Fenster anzeigt.

Bei Ausgaben enthält das Popup eine umfassende Übersicht mit allen Informationen, einschließlich der Beschreibung, der Kosten, wer die Zahlung geleistet hat und welche Mitglieder an der Ausgabe beteiligt waren.

Bei Abrechnungen zeigt das Popup die notwendigen Transaktionen an. Zudem wird ein Button bereitgestellt, mit dem die Abrechnungsnachricht erneut kopiert werden kann.

Prüfungsleistung im Modul:
DLMCSPSE01_D - Projekt: Software Engineering

Internationale Hochschule Fernstudium

Studiengang: M. Sc. Informatik

Architekturdokument

Fabian Lorenz

Betreuungsperson: Prof. Dr. Markus Kleffmann

Abgabedatum: 27. Dezember 2024

1 Technologieübersicht

In diesem Projekt wurden verschiedene Technologien und Werkzeuge ausgewählt, um die Anforderungen effizient und nachhaltig umzusetzen. Die Wahl der Technologien wurde durch persönliche Erfahrung, Verbreitung, Performance und Kompatibilität geleitet. Im Folgenden werden die verwendeten Technologien für das Backend, das Frontend und das Deployment beschrieben und ihre Auswahl kurz begründet.

Backend

Das Backend des Projekts wird in **Python**[1] entwickelt, da mit dieser Programmiersprache bereits umfangreiche berufliche Erfahrungen vorliegen. Darüber hinaus ist Python laut einer Statista-Studie die weltweit am häufigsten genutzte Programmiersprache (Statista, 2024). Die Sprache zeichnet sich durch eine große Entwickler*innen-Community sowie zahlreiche Bibliotheken und Erweiterungsmöglichkeiten aus, die bei Bedarf flexibel in das Projekt integriert werden können (Python Software Foundation, 2024).

Als Web-Framework wurde **FastAPI**[2] ausgewählt. Dieses Framework unterstützt eine asynchrone Verarbeitung und bietet eine automatische API-Dokumentation mittels OpenAPI (Tiangolo, 2024). Diese Eigenschaften ermöglichen eine effiziente Entwicklung moderner Backends, die Daten per JSON für eine Frontend-Applikation bereitstellen. Darüber hinaus gehört FastAPI zu den performantesten Python-Frameworks, was insbesondere für Anwendungen von Vorteil ist, bei denen Benutzer*innen in Echtzeit auf Antworten des Backends warten, wie dies bei FairMoney der Fall ist (Tiangolo, 2024). Als Webserver wurde **Uvicorn**[3] eingesetzt, das als Quasi-Standard für FastAPI gilt und auch in der offiziellen Dokumentation von FastAPI für sämtliche Beispiele verwendet wird (Tiangolo, 2024).

Für die Datenhaltung wurde **SQLite3**[4] gewählt, eine leichtgewichtige, file-basierte Datenbank, welche keine zusätzliche Datenbankinstanz erfordert und somit den Verwaltungsaufwand minimiert (SQLite Consortium, 2024). SQLite3 eignet sich besonders für Projekte mit geringen Anforderungen an Skalierbarkeit, weshalb es für die erste Version von FairMoney ideal ist (SQLite Consortium, 2024). Um dennoch eine zukunftssichere Implementierung sicherzustellen, wird mit **SQLAlchemy**[5] ein Abstraktionslevel zur Datenbank eingeführt. Dieses Werkzeug ermöglicht bei Bedarf einen Wechsel zu anderen Datenbanksystemen wie PostgreSQL oder MySQL, ohne den Anwendungsquellcode anpassen zu müssen – lediglich der entsprechende Treiber ist auszutauschen (SQLAlchemy, 2024). Darüber hinaus bietet SQLAlchemy Unterstützung für ORM (Object-Relational

[1] https://www.python.org/about/
[2] https://fastapi.tiangolo.com/
[3] https://www.uvicorn.org/
[4] https://www.sqlite.org/
[5] https://www.sqlalchemy.org/

Mapping) und asynchrone Datenbankoperationen, was den Quellcode robuster und flexibler gestaltet (SQLAlchemy 2024).

Frontend

Im Frontend wurde **JavaScript**[6] als Programmiersprache gewählt, da sie die Standardtechnologie für Browser-Frontends darstellt und gemäß der Umfrage von Stack Overflow die drittbeliebteste Programmiersprache weltweit ist (Statista, 2024).

Als Frontend-Framework bzw. Bibliothek kommt **React**[7] zum Einsatz. React überzeugt durch eine aktive Community, der Wiederverwendbarkeit von Komponenten und einer hohen Performance (Meta Platforms Inc, 2024). Für das clientseitige Routing zwischen verschiedenen Unterseiten wird **React Router**[8] integriert. Dieses Werkzeug ist in der React-Welt etabliert und ermöglicht eine nahtlose Integration in React-Anwendungen (React Router, 2024). Zur Implementierung von Popups wird **Reactjs-popup**[9] verwendet. Insbesondere bei der Darstellung von Popups sind viele Randfälle und potenzielle Seiteneffekte zu berücksichtigen, weshalb eine vorgefertigte und von der Community getesteten Bibliothek einer Eigenimplementierung vorzuziehen ist. Mit etwa 130.000 Downloads pro Woche über npm zeigt sich die hohe Akzeptanz und Qualität dieser Bibliothek in der Entwickler-Community (NPM, 2024).

Für den Build-Prozess des Frontends wird **Vite**[10] eingesetzt. Vite ist eine moderne Alternative zu anderen Buildsystemen wie Webpack und überzeugt durch eine hohe Performance sowie eine einfache Konfiguration.

Das Styling der Webseite basiert auf mehreren Technologien. **SASS**[11] wird als CSS-Erweiterung genutzt, um zusätzliche Funktionen wie die Verwendung von Variablen in CSS zu ermöglichen. Für das Design wurde **Material Design 3**[12], ein von Google entwickeltes Designsystem, ausgewählt, um eine konsistente und benutzerfreundliche Oberfläche zu schaffen. Material Design hat eine hohe Verbreitung im Internet und wird zum Beispiel als Standard Design-System für Android genutzt (Google, 2024). Dieses System gewährleistet konsistente und benutzerfreundliche Oberflächen und findet im Internet eine hohe Verbreitung. Zur Umsetzung von Material Design 3 wird die React-Komponentenbibliothek **MUI**[13] verwendet. Diese Bibliothek stützt sich auf einer großen Community und bietet fertig designte und getestete Komponenten im Stil des Designsystem, wodurch die Entwicklungszeit signifikant verkürzt werden kann (MUI, 2024).

[6] https://ecma-international.org/publications-and-standards/standards/
[7] https://react.dev/
[8] https://reactrouter.com/
[9] https://react-popup.elazizi.com/
[10] https://vite.dev/
[11] https://sass-lang.com/
[12] https://m3.material.io/
[13] https://mui.com/

Deployment

Für die Bereitstellung wurde **Docker**[14] ausgewählt, die aktuell am häufigsten genutzte Virtualisierungslösungen im Bereich der selbst entwickelten Software (Jetbrains, 2024). Docker ermöglicht es, die Anwendung unabhängig von den lokalen Installationen auf jedem System mit Docker-Installation auszuführen. Dies gewährleistet eine hohe Portabilität und vereinfacht die Bereitstellung der Anwendung erheblich.

Zur Optimierung des Deployment-Prozesses wird **Docker Compose**[15] verwendet. Docker Compose erlaubt die Konfiguration und Verwaltung mehrerer Container in einer einzigen Datei. Dadurch wird der Aufbau und die Ausführung der gesamten Anwendung effizient und benutzerfreundlich gestaltet. Für FairMoney, welches aus mehreren Diensten besteht, wie dem Backend, dem Frontend und einem Dateiverzeichnis für die Datenbank, bietet Docker Compose eine klare Struktur und Automatisierungsmöglichkeiten.

Der Aufbau mit Docker gewährleistet eine Zukunftssicherheit, da ein späterer Umstieg von lokalen Docker-Compose-Instanzen auf performantere Orchestrierungslösungen wie Kubernetes einfach möglich ist, welche sowohl on-premise als auch in der Cloud betrieben werden können. Dies eröffnet langfristig eine hohe Skalierbarkeit und Flexibilität für FairMoney.

2 Architekturübersicht

Die Architektur von **FairMoney** basiert auf einer **3-Schichten-Architektur**, die sich aus einem Web-Frontend, einem Backend und einer Datenbank zusammensetzt. Diese Struktur wurde gewählt, da sie eine klare Trennung der Verantwortlichkeiten gewährleistet, wodurch die Entwicklung erleichtert wird und eine solide Grundlage für zukünftige Wartung sowie Erweiterungen geschaffen wird.

Client-Server-Architektur

Die Anwendung folgt dem **Client-Server-Paradigma**, bei dem das Frontend als Client agiert und über eine REST-API mit dem Backend als Server kommuniziert. Diese Trennung ermöglicht es, die beiden Komponenten unabhängig voneinander zu entwickeln und zu skalieren. Darüber hinaus bietet die REST-API die Grundlage für eine potenzielle Integration weiterer Clients, wie etwa einer nativen mobilen App, ohne dass größere Anpassungen am Backend erforderlich sind.

[14] https://www.docker.com/
[15] https://docs.docker.com/compose/

Frontend (Präsentationsschicht)

Das Frontend wurde als **Single-Page-Application (SPA)** konzipiert, um eine reaktionsschnelle Benutzeroberfläche zu gewährleisten. Beim erstmaligen Aufruf der Anwendung wird der gesamte Programmcode des Frontends geladen, und die Seite wird vollständig clientseitig gerendert und verarbeitet. Notwendige Daten werden anschließend per HTTP-Request vom Backend abgerufen.

Auch beim Wechseln der Seite durch das Navigieren auf andere (Sub-)URLs wird dies von der Single-Page-Application behandelt und man bleibt technisch weiterhin in derselben Anwendung, bekommt aber je nach Konfiguration, andere Ansichten angezeigt. Diese Technik soll bei FairMoney dafür genutzt werden, um den Zugang zu spezifischen Gruppen über den Identifier in der URL zu spezifizieren.

Dieses Konzept bietet ein flüssiges Benutzererlebnis und ermöglicht insbesondere auf mobilen Geräten eine Bedienung, die mit der einer nativen App vergleichbar ist.

Backend (Logikschicht)

Das Backend bildet die zentrale Logikschicht der Anwendung und verarbeitet sämtliche Anfragen des Frontends. Es implementiert die Geschäftslogik, darunter die Verwaltung von Gruppen, das Speichern und Berechnen von Ausgaben sowie die Optimierung der Abrechnungstransaktionen.

Die REST-API des Backends folgt dem **Level 2 des Richardson Maturity Models**, bei dem Endpunkte als Ressourcen betrachtet werden und unterschiedliche Operationen durch den HTTP-Request-Typ (GET, POST, PUT, DELETE) definiert sind (Fowler, 2010). Eine Implementierung von HATEOAS (Hypertext As The Engine Of Application State) wird als nicht erforderlich erachtet, da die API nicht öffentlich zugänglich ist und das Deployment der konsumierenden (Frontend) und bereitstellenden (Backend) Schicht zusammen erfolgt. Dadurch können Änderungen an URLs oder API-Ressourcen jederzeit zwischen Frontend und Backend abgestimmt werden (Fowler, 2010).

Datenbank (Datenhaltungsschicht)

Die Datenbank dient der persistenten Speicherung aller relevanten Informationen, wie Gruppen, Ausgaben und Abrechnungen. Sie wird vom Backend angesprochen, wodurch die Konsistenz und Sicherheit der Daten gewährleistet wird. Die Verwendung einer relationalen Datenbank (vorerst SQLite3) ermöglicht eine einfache Modellierung der Daten und bietet außerdem die Möglichkeit, bei Bedarf auf leistungsfähigere Datenbanksysteme wie z. B. PostgreSQL zu migrieren.

Vorteile der gewählten Architektur

Die gewählte 3-Schichten-Architektur bietet mehrere zentrale Vorteile für FairMoney:

1. **Skalierbarkeit**: Jede Schicht kann unabhängig voneinander skaliert werden, um den Anforderungen wachsender Benutzerzahlen gerecht zu werden.
2. **Flexibilität**: Die klare Trennung zwischen Frontend und Backend erleichtert die Weiterentwicklung und den Austausch einzelner Komponenten.
3. **Wartbarkeit**: Änderungen an einer Schicht, wie der Austausch der Datenbank oder die Erweiterung der API, können vorgenommen werden, ohne andere Schichten zu beeinflussen.
4. **Zukunftssicherheit**: Die REST-API ermöglicht die Integration neuer Clients sowie die Erweiterung der Anwendung um zusätzliche Funktionen mit minimalem Entwicklungsaufwand.

Diese Architektur stellt sicher, dass FairMoney effizient, erweiterbar und benutzerfreundlich umgesetzt werden kann, während gleichzeitig die Wartung und zukünftige Weiterentwicklung erleichtert wird.

3 Struktur

In diesem Kapitel werden die Strukturen des Backends und Frontends detailliert beschrieben. Außerdem soll mit der Berechnung optimaler Transaktionen der zentrale Algorithmus von FairMoney vorgestellt werden. Abschließend wird eine Visualisierung des Gesamtsystems inklusive der Beziehungen zwischen Frontend und Backend präsentiert.

Backend und API

Das Backend basiert auf einer mehrschichtigen Architektur, die klar zwischen den folgenden Ebenen unterscheidet:

1. Controller

Die Controller-Ebene definiert die HTTP-Endpunkte der Anwendung und ist für die Verarbeitung eingehender Anfragen sowie die Erstellung entsprechender Antworten zuständig. In jedem Controller werden die Endpunkte mit ihren Pfaden, Anfrage- und Antwort-Bodies, sowie notwendigen HTTP-Header konfiguriert. Jeder Endpunkt ruft spezifische Methoden im **Request Handler** auf, um die eigentliche Geschäftslogik auszuführen.

2. Request Handler

Die Request Handler implementieren die Geschäftslogik, die den jeweiligen Endpunkten zugrunde liegt. Sie sind verantwortlich für Datenbankabfragen, Berechnungen und andere geschäftsrelevante Operationen. Die Trennung zwischen Controllern und Request Handlern sorgt für eine saubere Abs-

traktion zwischen HTTP-spezifischer Verarbeitung und der zugrundeliegenden Geschäftslogik, wodurch die Wartbarkeit der Anwendung erhöht wird.

3. Datenzugriffsschicht

Die Datenzugriffsschicht stellt Methoden bereit, um asynchron auf die Datenbank zuzugreifen, Daten zu lesen oder zu schreiben. Diese Abstraktionsschicht ist unabhängig von den HTTP-Endpunkten und fokussiert sich ausschließlich auf die Datenoperationen. Dadurch können Funktionen wie das Lesen von Gruppendaten generisch verwendet werden, unabhängig davon, in welchem Kontext sie benötigt werden.

Schreiboperationen auf der Datenbank werden nicht direkt durch die Datenzugriffsschicht abgeschlossen (committed). Stattdessen wird die Verantwortung für den Commit von Transaktionen an die Request-Handler delegiert. Dies ermöglicht es, in einem Request mehrere Datenbankoperationen innerhalb einer einzigen Transaktion zusammenzufassen und konsistent auszuführen.

Die asynchrone Datenbankverbindung wird als Singleton gespeichert, um Wiederverwendbarkeit und Effizienz zu gewährleisten.

Design-Pattern und Abhängigkeitsmanagement

Die Struktur des Backends lehnt sich lose an den Prinzipien des **Domain-driven Designs** an. Die Controller und Request Handler sind dabei thematisch nach den Domänen **Gruppe**, **Abrechnung** und **Zahlungen** getrennt.

Die Abhängigkeiten zwischen Controllern, Request Handlern und der Datenbankverbindung werden durch **Dependency Injection** gelöst. Die notwendigen Instanzen werden dabei, bei der Initialisierung der FastAPI-Anwendung bzw. bei der Initialisierung der jeweiligen Objekte bereitgestellt.

Backend-Endpunkte

Im Folgenden werden die vom Backend bereitgestellten und vom Frontend konsumierten Endpunkte mit ihrer Konfiguration beschrieben. Eine ausführlichere Beschreibung der zugrunde liegenden Modelle wird am Ende anhand von UML-Diagrammen präsentiert. Die implementierten Endpunkte sind wie folgt:

1. Gruppe

- GET /group/{group_id}: Abruf einer Gruppe basierend auf der ID.
    - Modell für Anfrage: -
    - Modell für Antwort: *GroupResponseModel*

    - Benutzerdefinierter HTTP-Anfrage-Header: -
- POST /group: Erstellung einer neuen Gruppe mit den im Request-Body übergebenen Daten.
    - Modell für Anfrage: *GroupCreateRequestModel*
    - Modell für Antwort: *GroupResponseModel*
    - Benutzerdefinierter HTTP-Anfrage-Header: -
- PUT /group/{group_id}/close: Schließen einer Gruppe anhand der ID.
    - Modell für Anfrage: -
    - Modell für Antwort: -
    - Benutzerdefinierter HTTP-Anfrage-Header: *x_user_name* (Eingeloggte Benutzer*in)
- GET /group/{group_id}/history: Abruf der Historie einer Gruppe basierend auf der ID.
    - Modell für Anfrage: -
    - Modell für Antwort: Liste von *GroupHistoryResponseModel*
    - Benutzerdefinierter HTTP-Anfrage-Header: -

2. Zahlungen

- POST /group/{group_id}/payment: Erstellung einer Zahlung für eine Gruppe anhand der übergebenen Daten.
    - Modell für Anfrage: *PaymentRequestModel*
    - Modell für Antwort: -
    - Benutzerdefinierter HTTP-Anfrage-Header: *x_user_name* (Eingeloggte Benutzer*in)

3. Abrechnungen

- GET /group/{group_id}/accounting/preview: Vorschau einer Abrechnung für die angegebene Gruppe ohne Persistierung.
    - Modell für Anfrage: -
    - Modell für Antwort: Liste von *TransactionResponseModel*
    - Benutzerdefinierter HTTP-Anfrage-Header: -
- POST /group/{group_id}/accounting: Erstellung und Persistierung einer Abrechnung für die Gruppe.
    - Modell für Anfrage: -
    - Modell für Antwort: Liste von *TransactionResponseModel*
    - Benutzerdefinierter HTTP-Anfrage-Header: *x_user_name* (Eingeloggte Benutzer*in)

Algorithmus: Berechnung optimaler Transaktionen

Die zentrale Geschäftslogik des Backends ist die Berechnung optimaler Transaktionen innerhalb einer Abrechnung, welcher bei den Endpunkten „GET /group/{group_id}/accounting/preview“ und „POST /group/{group_id}/accounting“ genutzt wird, um die Antworten zu generieren. Hierfür wird ein **Greedy-Algorithmus** eingesetzt, welcher in jedem Schritt versucht, möglichst viel Geld auszugleichen. Aufgrund seiner Natur führt der Algorithmus in den meisten Fällen, jedoch nicht immer, zu einer optimalen Lösung.

Die Entscheidung zugunsten dieses Ansatzes erfolgte aufgrund der hohen Performance, da nicht alle möglichen Lösungen durchprobiert werden müssen. Trotz umfangreicher Tests konnte keine einfache Konstellation gefunden werden, bei der der Algorithmus eine Lösung mit mehr Transaktionen als notwendig generiert hätte. Der Ablauf des Algorithmus gliedert sich wie folgt:

1. Aufteilung der Ausgaben: Zunächst werden die individuellen Ausgaben in der Gruppe auf die Mitglieder aufgeteilt. Dabei werden eigene Ausgaben als positives Guthaben angerechnet, während Ausgaben anderer Personen, an denen man beteiligt war, das Guthaben entsprechend verringern.

2. Gruppierung in Gläubiger*innen und Schuldner*innen: Anschließend werden die Mitglieder in zwei Gruppen unterteilt:

- Gläubiger*innen: Personen mit einem positiven Guthaben.
- Schuldner*innen: Personen mit einem negativen Guthaben. Beide Gruppen werden sortiert, sodass die höchsten Beträge (Forderungen bzw. Schulden) jeweils an den Anfang der Listen stehen.

3. Iterativer Prozess zur Schuldentilgung: Ein iterativer Prozess beginnt, der wiederholt wird, bis alle Schulden und Forderungen vollständig ausgeglichen sind:

- Die Person mit der höchsten Forderung (aus der Gläubiger*innenliste) und die Person mit den höchsten Schulden (aus der Schuldner*innenliste) werden ausgewählt.
- Es wird eine Transaktion erstellt, bei der der/die Schuldner*in dem/der Gläubiger*in einen Betrag zahlt. Die Höhe der Zahlung entspricht entweder der Schuld oder der Forderung, je nachdem, welcher Betrag geringer ist.
- Wenn die Schulden höher als die Forderung waren, wird der/die Schuldner*in mit der verbleibenden Restschuld wieder in die Schuldner*innenliste eingefügt. War die Forderung höher, wird der/die Gläubiger*in mit der verbleibenden Restforderung wieder in die Gläubiger*innenliste aufgenommen.
- Die jeweilige Liste wird anschließend neu sortiert, sodass die höchsten Schulden oder Forderungen wieder an den Anfang rücken.

4. Abschluss: Sobald alle Schulden und Forderungen aufgelöst wurden, enthält die erstellte Liste der Transaktionen eine faire Aufteilung der Zahlungen innerhalb der Gruppe.

Frontend

Das Frontend der Anwendung ist als **Single Page Application (SPA)** umgesetzt und nutzt React für die Entwicklung. Dabei werden drei zentrale Hauptseiten implementiert, welche abhängig von der URL durch den React Router gesteuert werden. Innerhalb der Seiten können, basierend auf dem Status der Nutzerinteraktionen, unterschiedliche Unterseiten angezeigt werden.

Zur besseren Wartbarkeit und Wiederverwendbarkeit des Codes werden verschiedene Elemente der Benutzeroberfläche in eigenständige React-Komponenten ausgelagert. Diese können nach Bedarf in den Views eingebunden werden.

Übergreifende Funktionen wie Backend-Interaktionen, Datenformatierung und Validierung werden in separaten JavaScript-Dateien zusammengefasst.

Hauptseiten und Routing

Der React Router zeigt abhängig von der URL eine der drei Hauptseiten an:

1. Gruppe erstellen

- URL: /
- Beschreibung: Diese Seite führt durch die Schritte zur Erstellung einer Gruppe.
- Unterseiten:
    - Landing Page
    - Eingabemaske für den Gruppentitel
    - Eingabemaske für den/die Ersteller*in der Gruppe
    - Eingabemaske für Mitglieder der Gruppe
    - Übersicht nach erfolgreicher Gruppenerstellung

2. Gruppe nutzen

- URL: /{group_id}
- Beschreibung: Seite zur Verwaltung einer Gruppe. Nutzer*innen können Ausgaben eintragen und Abrechnungen erstellen.
- Unterseiten:
    - Seite zur Auswahl der Identität
    - Hauptseite mit Anzeige der Gruppendetails
    - Eingabemaske für neue Ausgaben
    - Anzeige der Gruppenhistorie
    - Seite zur Erstellung einer Abrechnung

- Übersichtsseite der Abrechnung

3. Fehlerseite

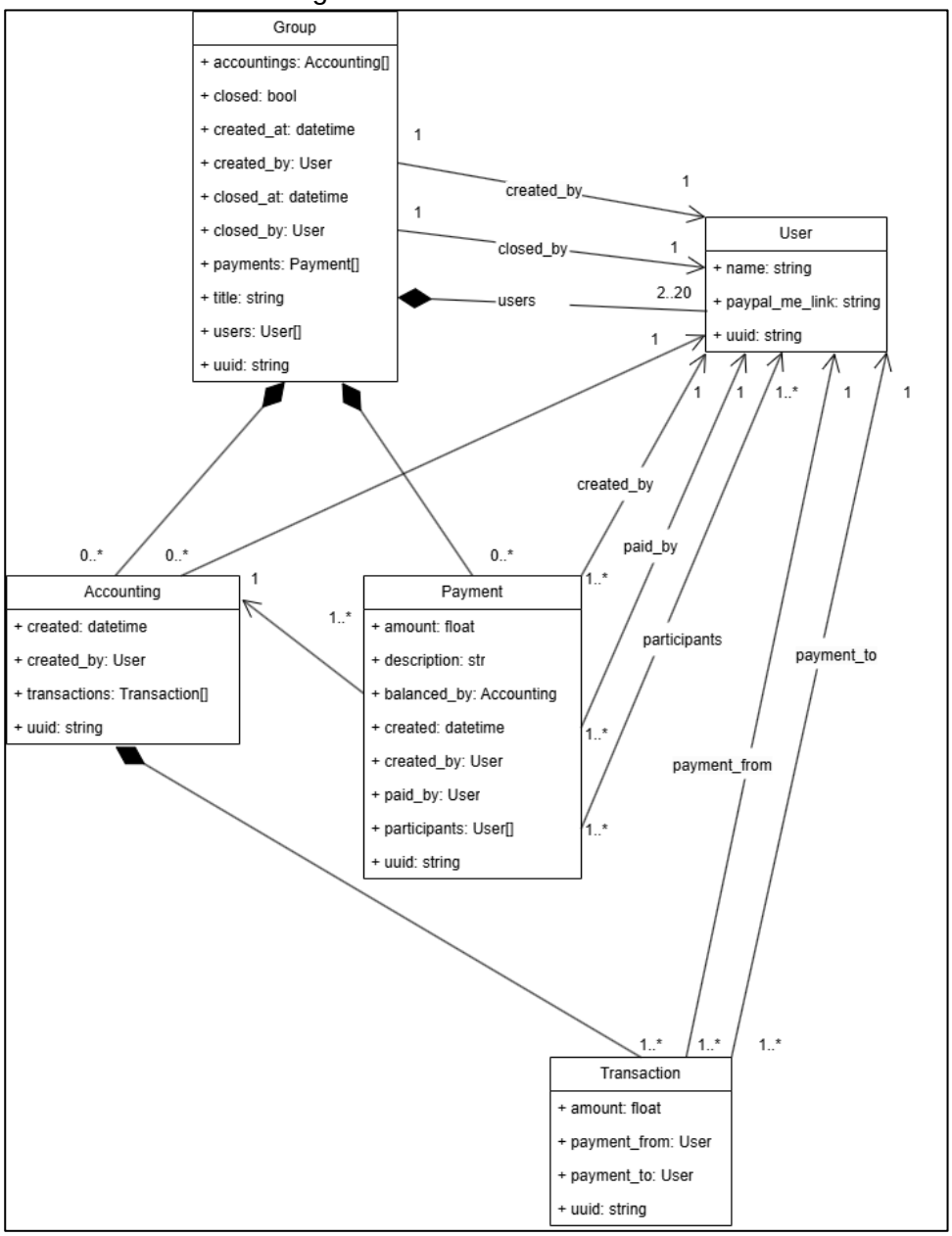
- URL: /error?errormsg={error_message}
- Beschreibung: Diese Seite wird bei Fehlern angezeigt und zeigt die entsprechende Fehlermeldung.
- Unterseiten: Keine

Visualisierung

Da weder das Frontend noch das Backend strikt objektorientiert implementiert sind, kann die vollständige Struktur nicht durch ein einzelnes UML-Klassendiagramm dargestellt werden. Stattdessen soll die Struktur im Folgenden mithilfe der folgenden Diagramme aufgezeigt werden:

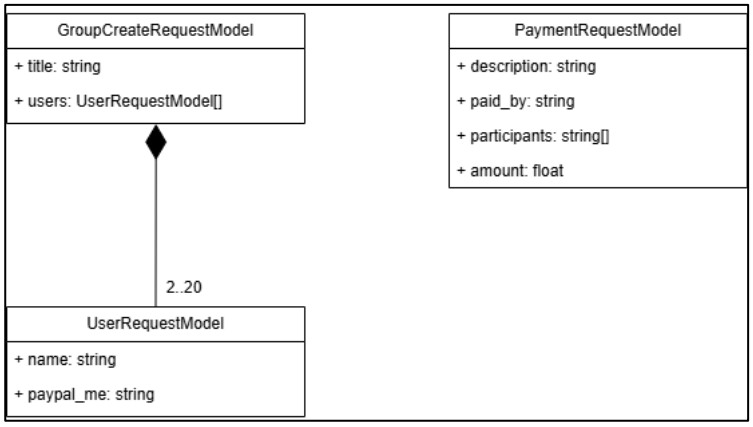
- UML-Klassendiagramms der Datenstrukturen innerhalb des Backends,
- UML-Klassendiagramm der JSON-Modelle für die Anfrage- und Antwort-Bodies der API,
- Entity-Relationship-Diagramm (Krähenfußnotation) der Datenbank, inklusive der notwendigen Beziehungstabelle zur Auflösung der n:m Beziehung zwischen Zahlungen und Benutzern,
- Visualisierung der wichtigsten Beziehungen des Gesamtsystems.

Abb. 1: UML-Klassendiagramm Backend



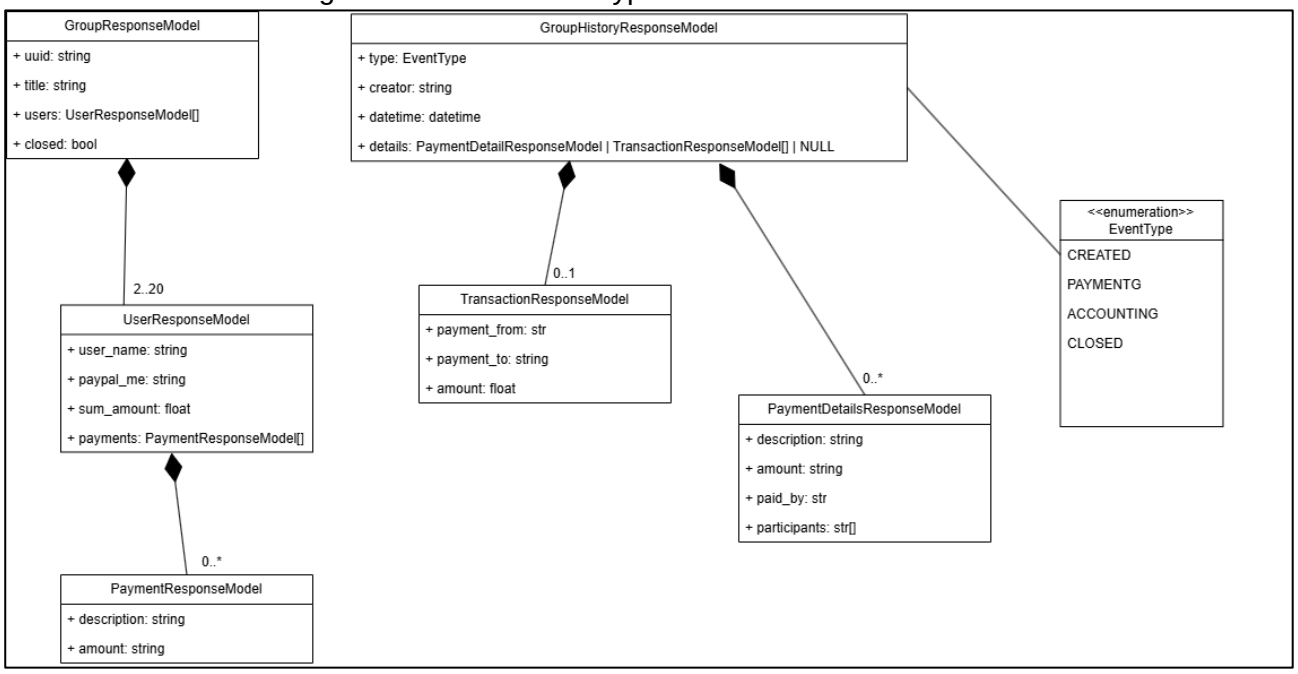
Quelle: Eigene Darstellung

Abb. 2: UML-Klassendiagramm HTTP-Anfragetypen



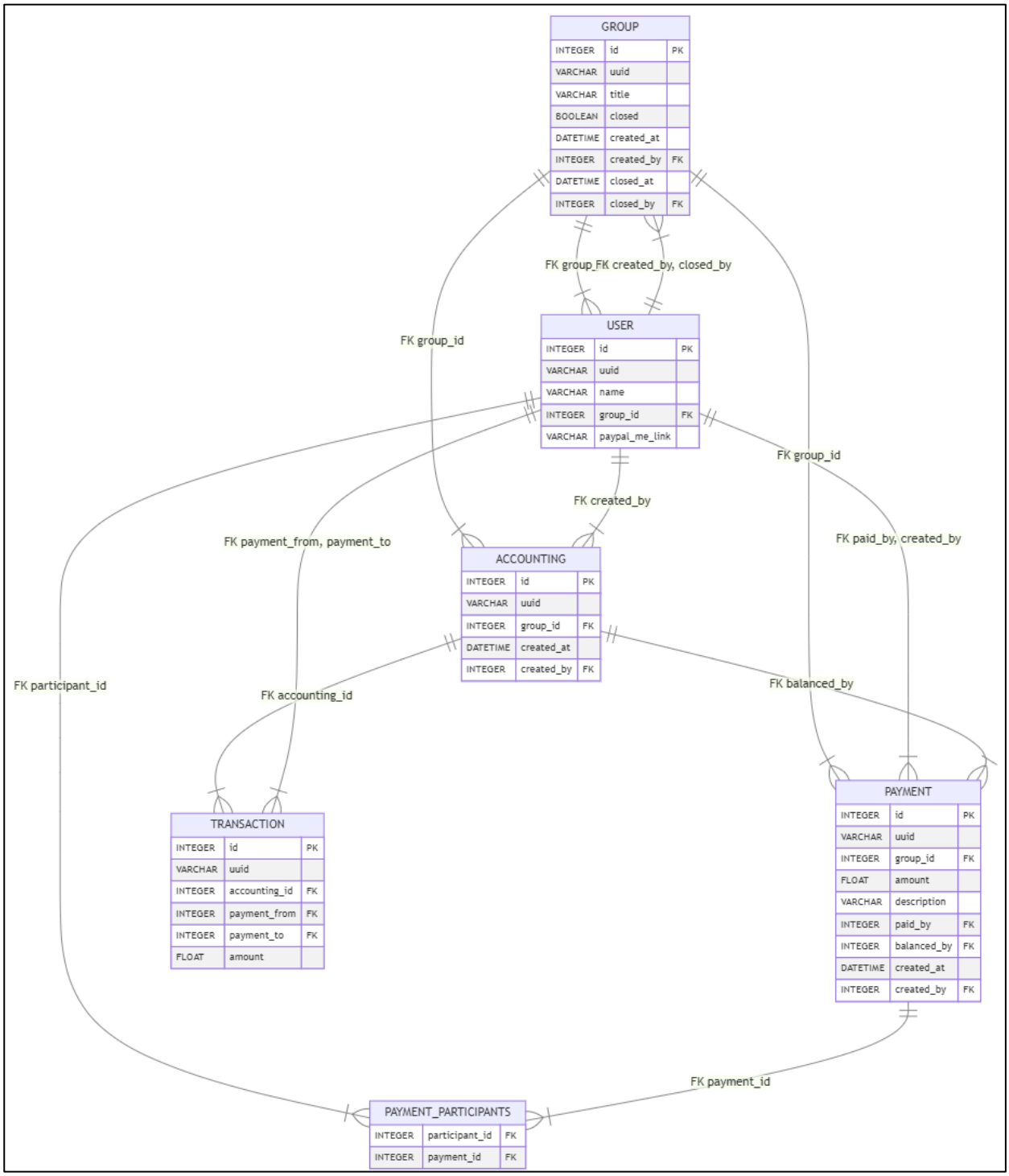
Quelle: Eigene Darstellung

Abb. 3: UML-Klassendiagramm HTTP-Antworttypen



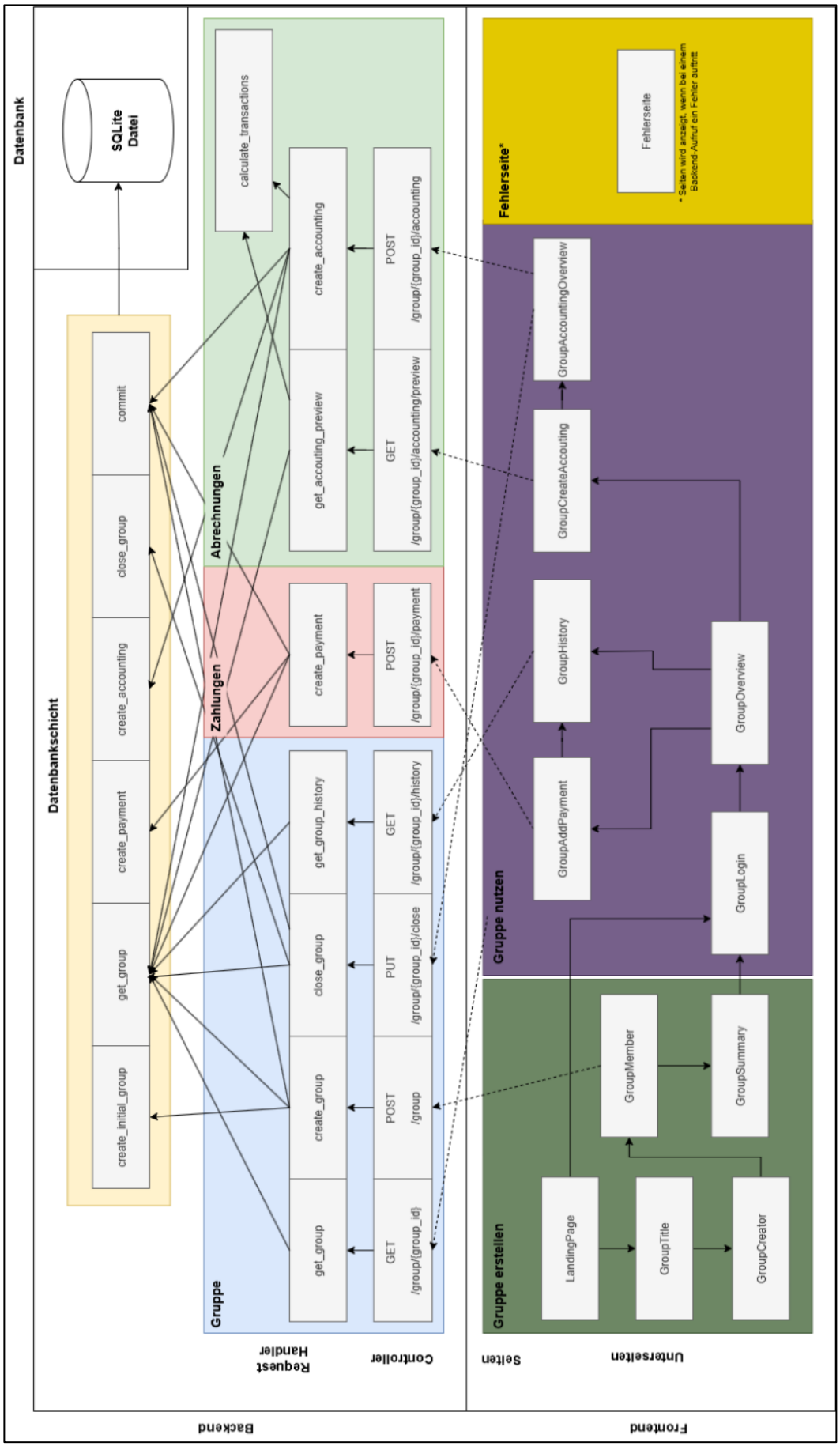
Quelle: Eigene Darstellung

Abb. 4: Entity-Relationship-Diagramm



Quelle: Eigene Darstellung

Abb. 5: Visualisierung Gesamtsystem

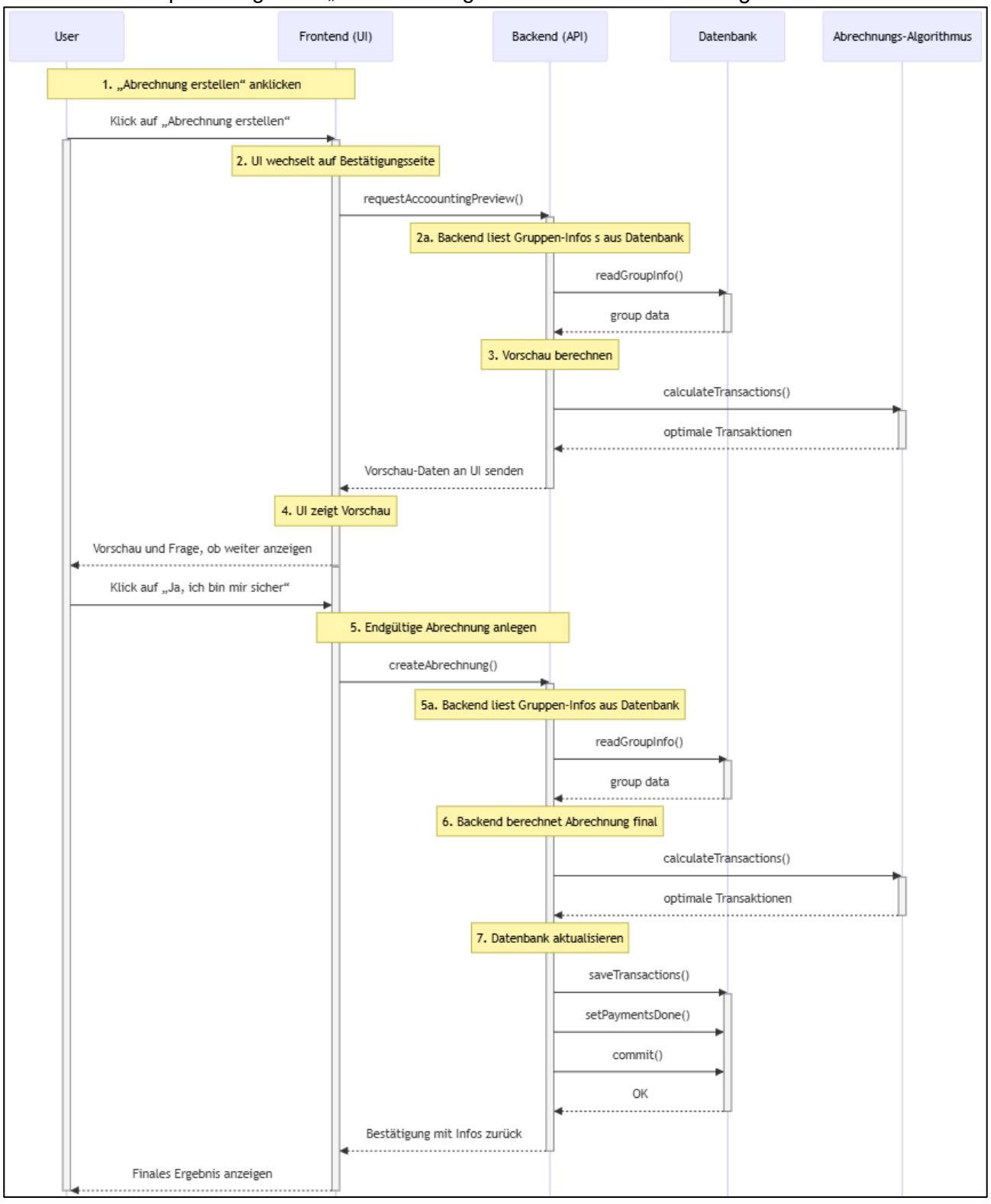


Quelle: Eigene Darstellung

4 Verhalten

Im Folgenden wird anhand der „Erstellen einer Abrechnung von der Gruppenübersicht bis zur Abrechnungsübersicht“ ein zentraler Systemablauf mithilfe eines UML-Sequenzdiagramms visualisiert.

Abb. 6: UML-Sequenzdiagramm „Visualisierung Erstellen einer Abrechnung“



Quelle: Eigene Darstellung

Literaturverzeichnis

Fowler. M. (2010). *Richardson Maturity Model: steps toward the glory of REST.* https://martinfowler.com/articles/richardsonMaturityModel.html

Google. (2024). *Material Design*. https://m3.material.io/

Jetbrains. (2023). *Developer Ecosystem: DevOps and Cloud*. https://www.jetbrains.com/lp/devecosystem-2023/devops/

Meta Platforms Inc. (2024). *React: The library for web and native user interfaces*. https://react.dev/

MUI. (2024). *Material UI - Overview*. https://mui.com/material-ui/getting-started/

NPM. (2024). *Reactjs-popup*. https://www.npmjs.com/package/reactjs-popup

Python Software Foundation. (2024). *Python: Python is powerful... and fast; plays well with others; runs everywhere; is friendly & easy to learn; is Open*. https://www.python.org/about/

React Router. (2024). *Feature Overview: Client Side Routing*. https://beta.reactrouter.com/6.28.0/start/overview

SQLAlchemy. (2024). *SQLAlchemy 2.0 Documentation*. https://docs.sqlalchemy.org/en/20/

SQLite Consortium. (2024). *About SQLite*. https://www.sqlite.org/

Statista. (2024). *Die beliebtesten Programmiersprachen weltweit laut PYPL-Index im Dezember 2024*. https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/

Tiangolo. (2024). *FastAPI: FastAPI framework, high performance, easy to learn, fast to code, ready for production*. https://fastapi.tiangolo.com/