GRUNDLAGEN MOBILE COMPUTING

GRUPPE D

Krypto-Portfoliomanager

Studiengang

Informationstechnologien und Wirtschaftsinformatik

SS 2020

Fürst-List Daniela

Hirsch Armin

Schweiger Daniel

Varcar Elvis

INHALTSVERZEICHNIS

1 EINLEITUNG

1.1 Systemumfang

Im Dokument wird ein Portfoliomanager für Kryptowährungen beschrieben, welcher das Portfolio des Nutzers anhand von Tortendiagrammen darstellt. Es soll auch möglich sein, das Portfolio öffentlich zu machen und dieses von der Community bewerten zu lassen. Die Teilung findet über QR Codes statt. Gleichzeitig soll es möglich sein bereits veröffentlichte Portfolios einfach in das eigene einzubinden.

2 SYSTEMÜBERBLICK

2.1 Zusammenhang mit anderen Produkten/ Systemen

2.2 Technologie Stack

Datenbank und Backend: Firebase
Build Management für Android: Gradle
Infrastruktur: Git
Software Development Kit: Flutter
Programmiersprache: Dart
IDE: IntelliJ
Geplante Plattformen: Android,Web

2.3 Die wichtigsten Funktionen im Überblick

Der Kryptoportfoliomanager

- Ermöglicht das Erstellen individueller Portfolios von Kryptowährungen
- Bietet eine Sharefunktion zum Teilen der Portfolios
- Bietet ein übersichtliches Managementdashboard
- Ermöglicht es, Portfolios über einen QR-Code zu scannen, um sie sich am Tablet/Mobiltelefon anzeigen zu lassen
- Bietet eine Bewertungsfunktion
- Ermöglicht kollaboratives managen eines Portfolios inklusive User Zugriffsmanagement

2.4 Benutzerprofile

Bei den Systemnutzern werden zwei Profile mit unterschiedlichen Rollen und Verantwortlichkeiten unterschieden:

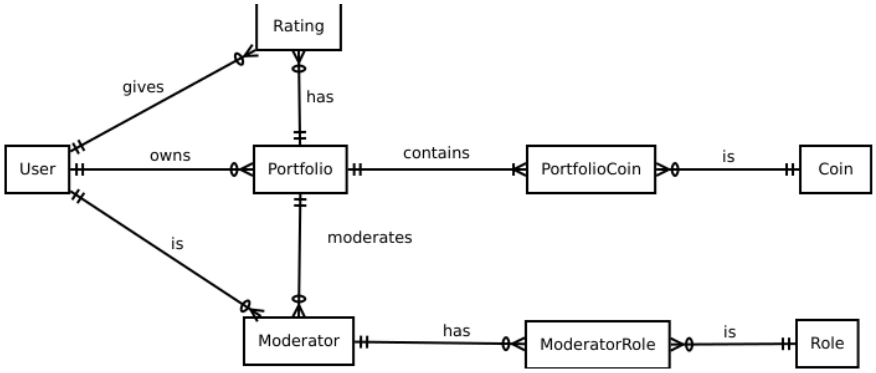
Benutzer	Funktionen und Verantwortungsbereich
User	Interessiert sich für Kryptowährungen und stellt sich Portfolios aus unterschiedlichen Währungen zusammen

2.5 Eingrenzung

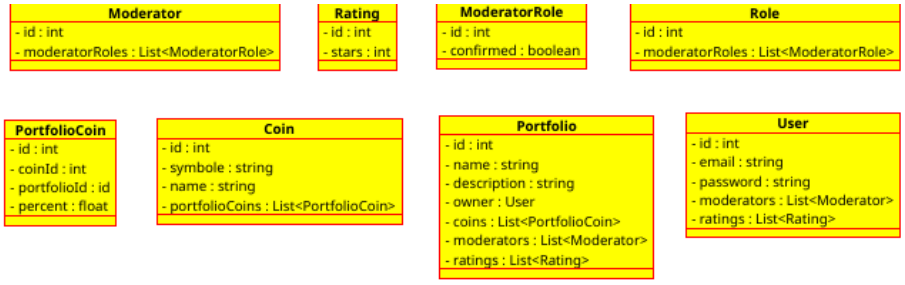
Das Dokument repräsentiert ein Studienprojekt und bildet die minimalen Funktionen des Systems, die zum Erreichen des Business Value notwendig sind ab. Zum Zweck der Realisierbarkeit dieses Projektes wird auf die Spezifikation und Umsetzung der Bereiche, die über die Kernfunktionen hinausgehen, verzichtet.

2.6 Annahmen und Abhängigkeiten

2.6.1 Datenbank



Wie aus dem ER-Modell ersichtlich gibt es im Datenbankmodell mehrere Entitäten, welche wie nachfolgend beschrieben in Beziehung zu einander stehen. Der *User* kann *Ratings* abgeben, *Portfolios* besitzen und *Moderator* sein. Jedes Portfolio kann keine, eine oder mehrere Bewertungen haben und enthält eine oder mehrere *PortfolioCoins*. Jede PortfolioCoin ist die Ausprägung einer *Coin*. Jeder Moderator moderiert ein Portfolio und hat eine *ModeratorRole*. Jeder ModeratorRole ist eine *Role* zugewiesen.



2.7 Beschreibung der Anforderungen

2.7.1 User Stories

Die Anforderungen werden zunächst aus der Sicht der Anwender in Form von User Stories dokumentiert. Mit den User Stories werden die wesentlichen Funktionen des Systems erfasst. Jede User Story beschreibt einen Nutzen aus der Sicht der Anwender. Die Summe der User Stories erfasst den Geschäftswert, des Systems aus Anwendersicht.

Die Stories werden aus der Sicht des Benutzerprofils geschrieben, welche durch die folgende Persona repräsentiert wird:

Jon

Jon repräsentiert den Nutzer. Er ist 28 Jahre alt und handelt regelmäßig mit Kryptowährungen. Er möchte den Krypto-Portfoliomanager nutzen, um ein Portfolio aus Kryptowährungen zu erstellen, sein Portfolio immer Überblick zu haben, es mit anderen zu teilen und die Portfolios anderer Nutzer zu sehen. Jon kann unterschiedliche Rollen übernehmen.

2.7.2 Use Cases

Die Use Cases dienen der detaillierten Beschreibung der funktionalen Anforderungen in Hinblick auf das Verhalten des Systems und die Interaktion zwischen den Systemnutzern und dem System. Das Use Case Model zerlegt die Anforderungen aus den User Stories in funktionale Bereiche. Jeder Use Case beschreibt einen notwendigen Ablauf zur Erfüllung der Anforderungen an das System. Die Use Cases dienen weiters zu Spezifizierung der Abnahmekriterien der User Stories und der Ableitung der Testfälle.

3 DETAILLIERTE ANFORDERUNGEN

3.1 Grundsätzliche Regeln und Bestimmungen

3 Rollen: Portfolio Owner, Portfolio Moderator, Portfolio Viewer (aktuell nicht implementiert, wird erst in Phase 2 umgesetzt)

Owner = kann Portfolios erstellen/bearbeiten und Rechte zu seinen Portfolios managen

Moderator = kann ihm geteilte Portfolios von Ownern managen (Coin Zusammenstellung) und bewerten, kann geteilte Portfolios nicht teilen, löschen und keine Userberechtigungen verwalten

Viewer = kann ihm geteiltes Portfolio betrachten und bewerten

3.2 Funktionale Anforderungen

3.2.1 Epic 1: Als Owner möchte ich Portfolios erstellen und löschen können

3.2.1.1 User Story 1.1: Portfolio anlegen

Als Owner möchte ich beliebig viele Portfolios, die aus einer oder mehr Kryptowährungen (Coins) bestehen erstellen, um diese mit anderen Usern zu teilen können und um die Portfolios bewerten zu lassen.

3.2.1.1.1 Akzeptanzkriterium 1.1.1

Portfolio muss eine Bewertungsfunktion haben

3.2.1.1.2 Akzeptanzkriterium 1.1.2

Portfolio muss mit anderen Usern geteilt werden können.

3.2.1.2 User Story 1.2: Portfolio löschen

Als Owner möchte ich meine Portfolios löschen können und diese somit von allen Usern mit denen ich es geteilt habe entfernen.

3.2.1.2.1 Akzeptanzkriterium 1.2.1

Portfolios dürfen nicht mehr angezeigt werden.

3.2.2 Epic 2: Als Owner und Moderator möchte ich die Zusammenstellung der Coins in Portfolios verwalten können.

3.2.2.1 User Story 2.1: Coin hinzufügen

Als Owner will ich meine Portfolios um Coins ergänzen können um das Portfolio immer am aktuellsten Stand zu haben.

3.2.2.1.1 Akzeptanzkriterium 2.1.1

Es müssen beliebig viele Coins genutzt werden können.

3.2.2.1.2 Akzeptanzkriterium 2.1.1

Hinzugefügte Coins müssen mit Usern des Portfolios geteilt werden.

3.2.2.2 User Story 2.2: Coin löschen

Wenn ich einen Coin nicht mehr besitze möchte ich ihn aus dem Portfolio entfernen.

3.2.2.2.1 Akzeptanzkriterium 2.2.1

Es müssen beliebig viele Coins gelöscht werden können.

3.2.2.2.2 Akzeptanzkriterium 2.2.1

Gelöschte Coins müssen mit Usern des Portfolios geteilt werden.

3.2.2.3 User Story 2.3: Coinverhältnis anpassen

Da ich regelmäßig mit Coins handle muss ich das Verhältnis der Coins in meinem Portfolio regelmäßig anpassen können.

3.2.2.3.1 Akzeptanzkriterium 2.3.1

Angepasste Verhältnisse müssen mit Usern des Portfolios geteilt werden.

3.2.3 Epic 3: Als Owner und Viewer, möchte ich ein Portfolio teilen bzw. geteilt bekommen können

3.2.3.1 User Story 3.1: Portfolio teilen

Um anderen Usern zu helfen bzw. meine Meinung zum Besten Portfolio zu zeigen möchte ich mein Portfolio teilen.

3.2.3.1.1 Akzeptanzkriterium 3.1.1

Portfolio muss mittels QR-Codes geteilt werden können.

3.2.3.2 User Story 3.2: Geteiltes Portfolio mittels QR Code hinzufügen

Ich will Portfolios von anderen Nutzern meinem Account hinzufügen, um diese bewerten und vergleichen zu können.

3.2.3.2.1 Akzeptanzkriterium 3.2.1

Ich will beliebig viele Portfolios mittels QR-Code hinzufügen können.

3.2.3.3 User Story 3.3: Fremde Portfolios aus meiner Portfolio Übersicht entfernen

Ich will Portfolios von anderen Nutzern aus meiner Übersicht entfernen können, wenn ich sie nicht mehr benötige. (Funktionalität fehlt im Mockup)

3.2.3.3.1 Akzeptanzkriterium 3.3.1

Portfolio muss aus der Übersicht entfernt sein, aber nicht vollständig gelöscht.

3.2.4 Epic 4: Als Moderator und Viewer, möchte ich ein Portfolio bewerten können

3.2.4.1 User Story 4.1: Erstmalige Bewertung

Als Moderator oder Viewer eines Portfolios, will ich meine persönliche Bewertung dafür abgeben können.

3.2.4.1.1 Akzeptanzkriterium 4.1.1

Bewertung muss mittels grafischer Oberfläche erstellt werden können (Sterne, Smileys, etc…).

3.2.4.2 User Story 4.2: Bewertung Ändern

Wenn ich meine Meinung zu einem Portfolio ändere möchte ich auch meine Bewertung dafür anpassen können.

3.2.4.2.1 Akzeptanzkriterium 4.2.1

Es muss allen Nutzern/Viewern die aktuellste Bewertung angezeigt werden.

3.2.5 Epic 5: Als Owner, möchte ich die Zugriffsrechte auf meine Portfolios verwalten können

3.2.5.1 User Story 5.1: Viewer zum Moderator machen

Als Owner eines Portfolios möchte ich beliebig viele User zu Moderatoren meines Portfolios machen.

3.2.5.1.1 Akzeptanzkriterium 5.1.1

Nur ich als Owner darf Viewer zu Moderatoren machen können.

3.2.5.2 User Story 5.2: Moderator zum Viewer machen

Als Owner will ich aus verschiedensten Gründen Moderatoren wieder zu Viewern meines Portfolios machen, um Bearbeitungsfunktionen zu entziehen.

3.2.5.2.1 Akzeptanzkriterium 5.2.1

Moderatoren dürfen, keine anderen Viewer zu Moderatoren machen.

3.2.5.3 User Story 5.3: User Black listen

Als Owner von Portfolios möchte ich Viewer für meine Portfolios Black listen, damit sie keine Einsicht mehr auf meine Portfolios haben.

3.2.5.3.1 Akzeptanzkriterium 5.3.1

Viewer dürfen sich nicht selbst von den Blacklists entfernen können.

3.2.6 Epic 6: Als Nutzer, möchte ich einen Überblick über alle und einzelne Portfolios haben

3.2.6.1 User Story 6.1: Dashboard mit Zusammenfassung aller Portfolios

Als Nutzer will ich eine Zusammenfassung alle meiner Portfolios in Form eines Dashboards sehen um mir schnell einen Überblick verschaffen können.

3.2.6.1.1 Akzeptanzkriterium 6.1.1

Das Dashboard muss gut lesbar sein.

3.2.6.2 User Story 6.2: Details einzelner Portfolios betrachten

Vom Dashboard aus will ich als User auf eine Detailansicht meiner Portfolios gelangen, um mir deren genaue Zusammensetzung ansehen zu können.

3.2.6.2.1 Akzeptanzkriterium 6.2.1

Detailansicht muss als Form eines Dashboards vorhanden sein.

3.3 Use Cases

3.3.1 Use Case 1.1.1: Portfolio anlegen

Ziel:	Portfolio anlegen
Akteure:	Owner
Auslösendes Ereignis:	Portfolio anlegen Button
Vorbedingung:	Keine

Nachbedingung:	Portfolio wurde angelegt
Beschreibung:	
User tippt auf den Portfolio anlegen Button (Plus Symbol). UI zum Anlegen des Portfolios öffnet sich. Bezeichnung des Portfolios wird eingetragen. Die Bezeichnung des Portfolios wird eingetragen. Coins können mittels der Suche hinzugefügt werden. Mittels Schieberegler kann die Aufteilung der Coins im Portfolio geregelt werden. Zusätzlich können beim Erstellen des Portfolios Moderatoren bestimmt werden. Mit einem Tipp auf den Save Button wird das Portfolio erzeugt und gespeichert. Mit einem Tipp auf den Delete Button wird das Portfolio wieder verworfen.	

3.3.2 Use Case 1.2.1: Portfolio bearbeiten

Ziel:	Portfolio bearbeiten
Akteure:	Owner, Moderator
Auslösendes Ereignis:	Portfolio bearbeiten Button
Vorbedingung:	Portfolio angelegt
Nachbedingung:	Änderung des Portfolios gespeichert
Beschreibung:	
Owner/Moderator tippt auf die Portfolioübersicht Portfolioübersicht öffnet sich. Owner/Moderator wählt das zu bearbeitende Portfolio. Portfolio Detailansicht öffnet sich. Durch Tipp auf den Bearbeitungsbutton öffnet sich die Bearbeitungsansicht (gleiche UI wie bei Portfolio Erstellung). Änderungen können in gleicher Form wie bei der Erstellung des Portfolios getätigt werden. Mit einem Tipp auf den Save Button werden die Änderungen gespeichert. Der Delete Button um das Portfolio zu löschen funktioniert nur für den Owner NICHT für Moderatoren.	

3.3.3 Use Case 1.3.1 Portfolio löschen

Ziel:	Portfolio löschen
Akteure:	Owner
Auslösendes Ereignis:	Portfolio löschen Button
Vorbedingung:	Portfolio angelegt
Nachbedingung:	Portfolios gelöscht
Beschreibung:	
Owner tippt auf die Portfolioübersicht Portfolioübersicht öffnet sich. Owner wählt das zu bearbeitende Portfolio. Portfolio Detailansicht öffnet sich. Durch Tipp auf den Bearbeitungsbutton öffnet sich die Bearbeitungsansicht (gleiche UI wie bei Portfolio Erstellung). Durch einen Klick auf den Delete Button wird das ausgewählt Portfolio gelöscht.	

3.3.4 Use Case 1.4.1 Portfolio teilen/geteilt bekommen

Ziel:	Portfolio mit anderen Teilen
Akteure:	Owner, Viewer
Auslösendes Ereignis:	Portfolio Detailansicht öffnen
Vorbedingung:	Portfolio angelegt
Nachbedingung:	Neuer User hat Portfolio in seiner Übersicht
Beschreibung:	
Owner tippt auf die Portfolioübersicht Portfolioübersicht öffnet sich. Owner wählt das zu teilende Portfolio.	

Portfolio Detailansicht öffnet sich, mit scanbarem QR-Code. Viewer scannt den gezeigten QR-Code vom Owner. Portfolio wird in der Portfolioübersicht des Viewers angezeigt.

3.3.5 Use Case 1.5.1 Portfolio bewerten

Ziel:	Portfolio bewerten
Akteure:	Moderator, Viewer
Auslösendes Ereignis:	Klick auf Bewertungsschaltfläche
Vorbedingung:	Fremdes Portfolio in Übersicht vorhanden
Nachbedingung:	Bewertung des Portfolios wird aktualisiert
Beschreibung:	
Moderator/Viewer tippt auf die Portfolioübersicht Portfolioübersicht öffnet sich. Moderator/Viewer wählt das zu bewertende Portfolio. Portfolio Detailansicht öffnet sich, mit Schaltfläche zur Bewertung. Mittels tippen auf die Schaltfläche zur Bewertung, wird dem Portfolio die entsprechende Bewertung zugeordnet. Gesamtbewertung des Portfolios verändert sich.	

3.3.6 Use Case 1.6.1 Portfolio Bearbeitungsrechte vergeben (noch vollständig nicht implementiert)

Ziel:	Berechtigungen für Portfolios vergeben
Akteure:	Owner
Auslösendes Ereignis:	Klick auf Manage Moderators Button
Vorbedingung:	Eigenes Portfolio vorhanden

Nachbedingung:	Neuer Moderator ist einem Portfolio zugeordnet
Beschreibung:	
Owner tippt auf die Portfolioübersicht Portfolioübersicht öffnet sich. Owner wählt ein eigenes Portfolio. Portfolio Detailansicht öffnet sich, mit Manage Moderators Button. Moderators Übersicht öffnet sich Mittels Suchfeldes wird nach Usern gesucht User der zum Moderator des Portfolios werden wird ausgewählt Berechtigungen werden an den entsprechenden User vergeben.	

3.3.7 Use Case 1.7.1 Portfolios betrachten

Ziel:	Portfolioübersicht/Portfolio Detailansicht anzeigen
Akteure:	Owner
Auslösendes Ereignis:	Klick auf Dashboard Schaltfläche
Vorbedingung:	Portfolio vorhanden
Nachbedingung:	
Beschreibung:	
Owner tippt auf die Dashboardschaltfläche Portfolioübersicht öffnet sich. Dashboard mit Daten allen vorhandenen Portfolios öffnet sich Klick auf ein Portfolio um die Detailansicht des Portfolios anzuzeigen Ein Dashboard mit der genauen Aufteilung des Portfolios wird angezeigt	

4 LESSONS LEARNED

4.1 Vorgehensweise bei der Technologieauswahl

Am Beginn der Planungsphase wurde in einem Teammeeting die Funktionen der Applikation festgelegt, anschließend fand eine lebendige Diskussion zu den verschiedenen Technologien stand. Recht schnell wurden die Möglichkeiten auf React Native und Flutter eingegrenzt und es fand eine ausführliche Diskussion zu den Vor- und Nachteilen dieser statt. Schlussendlich hat die Gruppe sich für Flutter entschieden da eine neue Programmiersprache das Interesse der Entwickler geweckt hat und während der Recherche Firebase als mögliche Technologie für das Backend angesprochen wurde. Durch die erleichterte Implementierung von Firebase in die Applikation mittels Flutter wurde auch die Planung der weiteren Vorgehensweise vereinfacht. Firebase wurde gewählt da aufgrund des kurzen Zeitrahmens der Entwicklungszeit eine vollständige Implementierung des Backend's die Entwicklungsressourcen strapazieren würde was wiederum zu einer Überziehung des Zeitrahmens führen würde. Es stellt eine Vielzahl von Funktionen zur Verfügung besonders die Real Time Database ist jedoch für die Entwicklung dieser Applikation wichtig, da sie eine Verteilung der Daten vereinfacht. Die Daten werden hierbei in einem JSON abgespeichert und auf alle Verbundenen Clients verteilt, diese Verteilung findet bei jeder Datenänderung statt. Sind Benutzer offline werden Änderungen lokal gespeichert und bei erneuter Verbindung übertragen, alle diese Funktionen findet auf Client Seite statt was zu einer guten Benutzererfahrung führt.

4.2 Entwicklung

Das Team wurde am Beginn des Projekts in 2 Gruppen geteilt ein Programmierteam und ein Team für das Dokument. Aufgrund der limitierten Personenzahl war eine Person Mitglied in beiden Teams. Das Programmierteam wurde weiter in zwei Personen Teams eingeteilt und die Use Cases wurden innerhalb dieser Teams aufgeteilt. Dies hat am Anfang recht gut funktioniert da die Programmierung modularer war und es den Teams möglich war Use Cases einzeln zu implementieren oder das alle Personen anwesend waren. Dieses unabhängige Arbeiten hat dazu geführt das besonders am Anfang der Fortschritt recht schnell war, und die zweier haben dazu geführt das Pair Programming möglich war. Leider hat diese Art der Entwicklung im späteren Stadium zu einem großen Problem geführt da 2 Personen aus unserem Team aufgrund von Negativen Versuchen aus dem Studium ausgeschlossen wurden. Da die Teams sehr separat aktiv waren und die Ausfälle ein komplettes Programmier Team betroffen haben gab es auf einmal teile der Use Cases die noch nicht programmiert worden sind und nicht innerhalb der Zeitrahmen in der Applikation implementiert werden

konnten. Somit wurde die Implementation von Moderator ausgelassen da diese Implementation innerhalb der Zeitspanne nicht möglich war. Aufgrund dieses Rückschlags werden diese Funktionen zu einem späteren Zeitpunkt implementiert. Use Case 1.6.1 funktioniert somit momentan nicht, da jeder Teilnehmer die Funktionen ausführen kann. Es wurde zwar damit begonnen aber noch nicht vollständig implementiert. Die Zusammenarbeit im Team war jedoch super und besonders die Implementierung des QR Codes ging einfacher als gedacht. Gleichzeitig war das Aufsetzten der Umgebung am Anfang zeitaufwendiger als gedacht was die Programmierzeit reduziert hat, besonders in Windows ist die Menge an benötigter Programme sehr umfangreich und besonders für kleine Festplatten problematisch. Das Aufsetzten hätte hier nicht individuell stattfinden sollen da es immer wieder zu Problemen bei der Installation gekommen ist. Sehr gut hat die Implementation mit Firebase funktioniert da nur einer unserer Mitglieder wirklich ein Entwickler ist was auch zu einer Verlangsamung der Programmierung geführt hat. Besonders beim Cachen der Daten war das bemerkbar da wir die Implementierung dieser Funktion nicht in der vorgegebenen Zeit geschafft haben.

5 AUSBLICK

Im nächsten Schritt, wollen wir unsere App im bei Google Play veröffentlichen. Dazu müssen wir uns bei mehreren Diensten anmelden: Neben einem Google-Konto benötigen wir noch einen kostenpflichtigen Zugang für die Plattform Google Play Developer Console.
Unsere App liegt bereits im APK-Format (Abkürzung für „Android Package“) vor, um auf einem Gerät mit Android-Betriebssystem installiert werden zu können. Die APK-Datei enthält alle Bestandteile unserer Anwendung in komprimierter Form. Darüber hinaus ist auch eine Signierung dieser Datei erforderlich – die digitale Signatur dient dem Nachweis über den Urheber einer App und ist ebenfalls für die Installation unter Android notwendig. Wir werden das mit Android Studios machen.
Es gibt verschiedene Möglichkeiten, um eine Android-Applikation zu signieren. Relativ einfach gelingt das durch die Nutzung des Android Studios. Die folgenden Zeilen beschreiben Schritt für Schritt, wie Sie Ihre App in das APK-Format inklusive einer Signatur übertragen.
Angemeldet mit unserem Google Account müssen wir uns bei Google Play Developer Console registrieren. Dafür müssen wir einmalig 25 US-Dollar bezahlen.
Da wir mit der App Geld verdienen möchten, müssen wir auch ein Google-Payments-Händlerkonto anlegen.
Nachdem wir uns bei allen erforderlichen Google-Diensten registriert haben, können wir im Google Play Store die App hochladen. Allerdings ist mit dem Upload der Applikation diese nicht direkt im Google Play Store verfügbar, denn jede eingereichte App wird vor der Publikation von Google geprüft.
Der Upload der App erfolgt über die Google Play Developer Console. Nach dem Log-in auf der Plattform klicken wir im Menü unter „Alle Apps“ den Button „Neue App hinzufügen“. Dort geben Sie dann die Sprache der Anwendung an und tragen den Namen der App ein. Durch einen Klick auf „APK hochladen“ gelangen wir in ein neu erstelltes Menü, das den Namen unserer App trägt. Dort sehen wir auf der linken Seite verschiedene Menüpunkte („APK“, „Store-Eintrag“, „Einstufung des Inhalts“ etc.), denen wir uns nach und nach widmen.
Da sich unsere App noch im Teststadium befindet, haben wir beschlossen, einen einen Alphatest mit einer kleinen geschlossenen Testgruppe durchzuführen, die wir über E-mail zum Test einladen durchführen.
Nach dem erfolgreichen Test laden wir später auch die finale Anwendung der App hoch.
Im Store-Eintrag geben wir jene Informationen an, die Nutzer später auf der App-Seite im Google Play Store finden sollen.
Hinsichtlich der Preisgestaltung haben wir uns überlegt, keinen festen App-Preis anzubieten, weil dies eine große Hürde für für die potentiellen Nutzer darstellen kann und weil Google 30% des Netto-Umsatzes je Transaktion einbehält. Deshalb bieten wir unsere App als Freemium-

App an. Die grundlegende Applikation zum Anlegen der Portfolios ist kostenlos, alle weiteren Features (z.B. Moderatorfunktion) sind nur gegen Bezahlung verfügbar.
Nach der Veröffentlichung der App werden wir diese regelmäßig warten und mit neuen Features bestücken. Als neue Erweiterungen sind geplant:

- Erweiterung des Dashboards
- Sharen des Portfolios über Soziale Medien
- Cashing