1 Einleitung

1.1 Motivation

Um die Besonderheiten bei der Entwicklung einer Applikation für eine mobile Plattform kennenzulernen, sollte im Rahmen des Mobile Application Labs ein Quarett Spiel für das Android System entwickelt werden. Da ein Spiel viele verschiedene Aspekte einer Plattform benötigt und keine triviale Applikation darstellt, erweist sich diese Problemstellung als geeignet für den Einstieg in die Android Programmierung.

Außerdem konnten nur wenige Quartett-Apps gefunden werden, die nach gängigen Kritieren als spielbar empfunden werden. So sind die meisten dieser Apps weder optisch ansprechend noch performant oder intuitiv bedienbar. Kreative oder innovative Ideen sind nur bei sehr wenigen Apps vorhanden. Auch die Erweiterbarkeit und der generelle Umfang, also die Anzahl an spielbaren Karten, ist meist sehr beschränkt. Zudem werden oft unötigerweise für unterschiedliche Thematiken der Quartett Karten zahlreiche Apps produziert die sich lediglich in den vewendeten Graphiken unterscheiden. All diese Probleme motivierten somit die Entwicklung einer echten Alternative für Quartett-Apps.

1.2 Ziel

Das hiermit erklärte Ziel dieses Projekts ist die Entwicklung einer Quarett-App, welche die meisten der oben aufgelisteten Probleme der momementan verfügbaren Apps löst oder diese zumindest entscheidend verbessert. Außerdem sollen effektive Techniken für die Programmierung von Android Apps erabeitet werden. Auch das Android API sollte nach Beendigung des Projekts im wesentlichen bekannt sein. Besonderheiten im Umgang mit den begrenzten Ressourcen von mobilen Plattformen sollten ebenfalls erlernt werden.

Die App selbst sollte dabei einen klassischen Einzelspieler Modus gegen einen Computergegner und mindestens ein innovatives Feature aufweisen, um sich von der Konkurrenz abzusetzen.

1.3 Aufbau

In der nun folgenden Dokumentation werden zuerst die Grundlagen des Quartettspiels, der Android Plattform und die verwendeten Frameworks erklärt. Danach folgt eine Anforderungsanalyse, welche sowohl funktionale als auch nicht funktionale Anforderungen umfasst. Im nächsten Abschnitt werden einige wichtige Details der Implementierung sowie die übergreifende Architektur und diverse Besonderheiten der App beschreiben. Zuletzt werden die gestellten Anforderungen mit der tatsächlichen Implementierung der App verglichen und es wird festgestellt welche Anforderungen bis zu welchem Grad erfüllt wurden.

2 Grundlagen

2.1 Quartettspiel

2.2 Mobile Plattform

2.3 Frameworks

3 Anforderungsanalyse

3.1 Funktionale Anforderungen

Im folgenden werden alle funktionalen und nicht funktionalen Anforderungen aufgelistet, die vor oder während der Entwicklung der Applikation gestellt wurden.

FA1 Hauptmenü
Der Benutzer kann von einem Hauptmenü, welches nach Start der App angzeigt wird, schnell auf die wichtigsten Funktionen der App zugreifen.

FA2 Spielmodi
Der Benutzer kann vor jedem Spiel aus 4 verschiedenen Spielmodi wählen: Zeitspiel (Spielende nach Ablauf eines Zeitlimits), Punktspiel (Spielende bei bestimmter Punktezahl), Rundenspiel (Spielende nach bestimmer Anzahl Runden), Insane (Vergleiche umgekehrt, Spielende nach bestimmter Anzahl Runden).

FA3 Computergegner

Im Einzelspieler Modus kann der Benutzer gegen einen simulierten Gegner antreten. Dieser sollte sich wie ein menschlicher Spieler verhalten und nicht auf Informationen zurückgreifen können die einem menschlichen Gegner normalerweise nicht zur Verfügung stehen. So sollte z.B. die Werte der Karte des Benutzers dem Computergegner nicht für die Planung des Spielzugs zur Verfügung stehen. So soll ein „unfaires“ Verhalten und damit ein frustrierendes Spielerlebnis verhindert werden.

FA4 Schwierigkeitsgrad
Vor jedem neuen Spiel kann der Benutzer einen von 3 verschiedene Schwierigkeitsgraden (Leicht, Mittel, Schwer) wählen. Je nach gewähltem Schwierigkeitsgrad handelt der Computergegner mehr oder weniger intelligent.

FA5 Spiel fortsetzen
Der gesamte Spielfortschritt wird kontinuierlich in der Applikation persistent gespeichert. So kann auch nach einem Neustart der App das Spiel ohne Fortschrittsverlust fortgesetzt werden.

FA6 Gallerie
Alle im Spiel vorhandenen Quartettdecks lassen sich in einer speziellen Ansicht, der Gallerie, betrachten. Dabei können alle im Deck enthaltenen Karten einzeln in einer detaillierten Ansicht betrachtet werden.

FA7 Deck Download
Die Applikation erlaubt das Herunterladen weiterer Kartendecks von einem externen Server. Nach dem Herunterladen können diese Decks genauso wie die bereits in der App vorhandenen Decks im Spiel verwendet werden. Die Decks werden persistet gespeichert und sind somit nach erfolgreichem Download auch ohne Internetverbindung dauerhaft verfügbar.

FA8 Statistiken
Die App sammelt während der Laufzeit spielbezogene Daten, um Statistiken zu ermöglichen. Diese Statistiken können vom Spieler in einer speziellen Ansicht eingesehen werden. Mögliche Statistiken sind: kill / death ratio, höchste Gewinnserie, höchste Verlustserie

FA9 Rangliste

Es gibt eine Rangliste, in welcher der Benutzer die im Spiel erhaltenen Punkte mit seinem Namen publizieren kann.

FA10 Achievements
Die App beinhaltete ein Achievementsystem. Wenn gewissen Herausforderungen erfüllt werden können Achievements freigeschaltet werden und in einer speziellen Ansicht angesehen werden.

FA11 Quartetteditor
Es wird ein Quartetteditor bereitgestellt, der es dem Benutzer ermöglich eigene Quartettdecks zu erstellen. Mit den erstellen Decks kann wie mit den bereits im Spiel integrierten Decks gespielt werden.

FA12 Levelsystem
Ein Levelsystem suggeriert permanenten Fortschritt. Nach jedem Spiel erhält der Benutzer Erfahrungspunkte, welche das Level steigen lassen. Mit diesem System soll eine Langzeitmotivation erreicht werden.

FA13 Multiplayer
Es ist über ein Netzwerk möglich gegen andere Benutzer der App anzutreten.

3.2 Nicht Funktionale Anforderungen

NFA1 Robustheit
Das Spiel soll eine gewisse Robustheit aufweisen, also auf fehlerhafte Eingaben oder unvorhergesehene Ereignisse angemessen reagieren. Im Falle eine Absturzes sollte die Applikation ohne Verlust des Spielfortschritts neu gestartet werden können.

NFA2 Erweiterbarkeit
Die Programmstruktur der Applikation sollte derart gestaltet sein, dass spätere Erweiterungen möglichst einfach vorgenommen werden können.

NFA3 Responsiveness
Die Oberfläche sollte stehts innerhalb einer sehr kurzen Zeit auf Benutzereingaben reagieren. Bei längeren Wartezeiten, etwa während eines Downloads, sollte der Benutzer

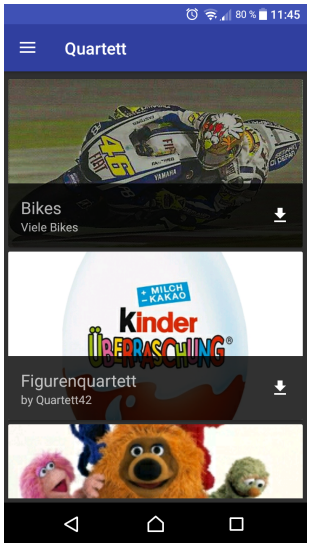

Abbildung 1: Deckansicht

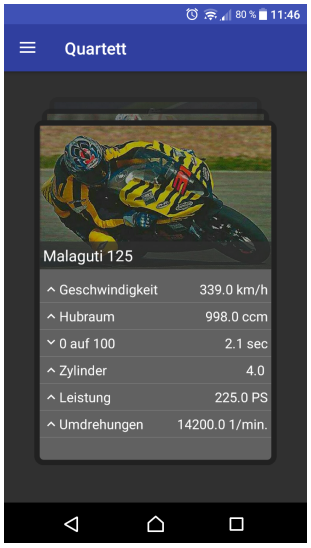

Abbildung 2: Kartenansicht

permanent, durch den Einsatz von entsprechenden GUI-Elementen, über den Fortschritt der Operation informiert werden.

NFA4 Usability
Der Benutzer sollte die App, nach einer kurzen Einführung, duch eine intuitive und benutzerfreundliche Oberfläche ohne weitere Anleitung bedienen können.

4 Konzept und Entwurf

4.1 Mockups

5 Implementierung

5.1 Ausgewählte Implementierungsdetails

5.1.1 Gallerie

In der Gallerie werden alle Kartendecks aufgelistet, die auf dem Smartphone vorhanden sind oder vom Server heruntergeladen werden können. Heruntergeladenen Decks könne durch Antippen geöffnet werden. In der geöffneten Ansicht kann der Benutzer durch vertikale Wischgesten durch die einzelnen Karten des virtuellen Kartenstapels

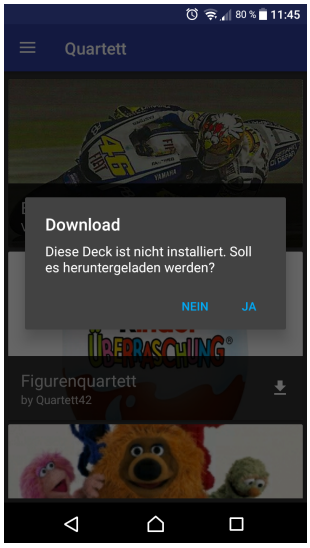

Abbildung 3: Deckansicht

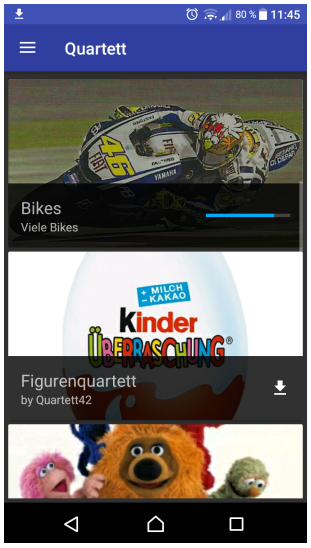

Abbildung 4: Kartenansicht

blättern. Tippt der Benutzer ein Deck an, welches nicht heruntergeladen ist, wird ein Dialog geöffnet in welchem das Herunterladen des Decks bestätigt oder abgelehnt werden kann. Das Deck wird nicht direkt geladen, da sich der Benutzer eventuell in einer Netzwerkumgebung befindet in welcher durch Downloads Kosten entstehen können. Durch den Bestätigungsdialog wird dem Benutzer somit eine Möglichkeit gegeben den Download zu einem späteren Zeitpunkt mit günstigeren Netzwerkbedingungen zu starten. Wird der Dialog bestätigt startet der Download des Decks. Im Listenelement des Decks wird ein Ladebalken angezeigt und im Notification Drawer wird eine Notification erstellt die ebenfalls den Fortschritt des Downloads anzeigt. Nach erfolgreichem Download wird die Notification geschlossen und der Ladebalken verschwindet wieder. Das Deck ist dann persistent auf dem Gerät gespeichert und kann nun auch ohne Internetverbinung angezeigt werden. Zudem können nun auch die Karten des Decks, wie oben beschreiben angezeigt werden. Auch kann das Deck nun im Einzelspieler Modus verwendet werden.

Um die Gallerie mit diesen beschriebenen Funktionen zu realisieren waren einige besondere Implementierungsmethoden notwendig. Da den meisten Decks und Karten relativ hoch auflösendede Bilder zugeordnet sind, entstehen beim Anzeigen der Decks bzw. Karten Verzögerungen, da große Datenmengen geladen werden müssen. Dabei ist außerdem zu erwähnen, dass das Laden der Bilder der Decks, die nicht auf dem Gerät gespeichert sind, über das Netzwerk erfolgt und somit zusätzliche Verzögerungen ent-

stehen. All diese Verzögerungen sind als starke Ruckler bemerkbar und beinträchtigen das Nutzererlebnis erheblich. Das Laden der Bilder wurde daher auf einen zweiten Thread ausgelagert. Somit kann der Benutzer weitere Interaktion vornehmen während im Hintergrund die Bilder nachgeladen werden. Auch der Download eines Decks verwendet einen eigenen Thread, um Verzögerungen im Hauptthread der Applikation zu vermeiden. Zusätzlich wurde hier auch das Service Modell der Android Platform verwendet. So kann garantiert werden, dass der Download abgeschlossen wird, auch wenn die Gallerie verlassen oder die App während des Downloads geschlossen wird. Um die heruntergeladenen Daten in der Datenbank zu speichern war ebenfalls eine besonderer Strategie nötig. Die Daten können nicht sofort gespeichert werden, da sonst bei Fehlern inkonsistente Zustände in der Datenbank auftreten. Daher werden geladene Daten zunächst im RAM gehalten bis der Download vollständig abgeschlossen ist und dann in einer einzigen Transaktion in die Datenbank geschrieben. Somit wird immer ein konsistenter Zustand der Datenbank erreicht und Fehler können einfacher abgefangen werden.

5.1.2 Einzelspieler

5.1.3 Multiplayer

5.2 Architektur

5.2.1 Datenmodell

Im obigen ER Diagramm ist die Struktur unseres Datenmodells dargestellt. Zur Realisierung wurde die in Android integrierte SQL Datenbank SQLite in Kombination mit Sugar ORM verwendet. So konnte die einzelnen Entitäten direkt über Klassen angesprochen werden und es mussten keine SQL Statements verwendet werden. Allerdings beherrscht Sugar ORM in der verwendeten Version keine Listen und Beziehungen zwischen den einzelnen Eintitäten können mit Sugar ORM ebenfalls nur schwierig oder gar nicht dargestellt werden. Die Daten der gespeicherten Bilder wurden nicht in die SQL Datenbank geladen sondern direkt im internen Speicher des Geräts abgelegt. Auch die „Preferences“ werden nicht mit SQL gespeichert sondern in den SharedPreferences des Android Systems abgelegt. SharedPreferences ist eine einfach Key-Value Datenbank für kleine Datenmengen, wie z.B. die Einstellungen einer App.

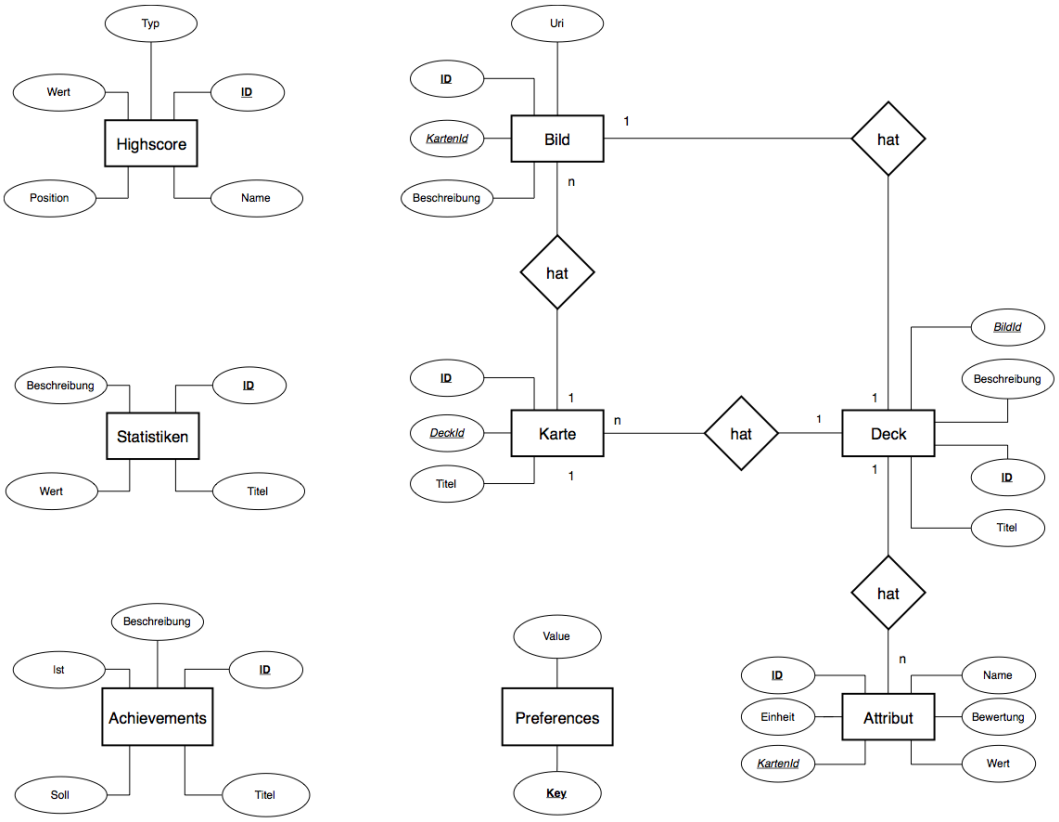

Abbildung 5: ER Diagramm des Datenmodells

5.2.2 Klassenstruktur

Um die Applikation zu strukturieren wurde hier das Model-View-Presenter Pattern angewendet. Mit diesem Pattern wird jede Activity in Model-, View- und Presenter- Komponenten zerlegt. Die Model Komponenten beinhalten die Zugriffsschicht auf die Datenbank mit allen zugehörigen Klassen. In der hier vorgestellten App wird das Model durch Sugar ORM bzw. durch dessen Entitätsklassen dargestellt. Damit existiert auch nur ein gemeinsames Model für alle Activities. Auf das Model kann nur vom Presenter aus zugegriffen werden. Dieser verwaltet die eigentlich Logik einer Activity. Dabei ist der Presenter selbst eine einfache Java Klasse ohne direktes Wissen über das Android System oder die Datenbank. Events der View werden an den Presenter weitergeleitet und dort bearbeitet. Der Presenter kann zur Bearbeitung von Events sowohl auf das Model als auch auf die View Komponente zugreifen. Diese Komponenten beinhaltet die

Anzeigeschicht und verwaltet die einzelnen GUI-Elemente. In unserer App wird die View von einer Fragment Klasse implementiert. Direkt kommuniziert die View nur mit dem Presenter und greift nicht auf das Model zu. In der Praxis gibt es dabei allerdings einige Ausnahmen.

Um z.B. das Anzeigen von Listen in Android effizient zu realisieren werden sogenannte Adapter verwendet. Diese greifen meist direkt auf die Datenbank zu und stellen die erhaltenen Daten direkt in einer ListView da. Damit können diese Klassen nicht eindeutig der Model oder View Kompomenten zugeordnet werden. Daher implementieren diese meist sowohl die Eigenschaften des Models als auch der View.

Durch die klare und immer gleichförmige Strukturierung in Model, View und Presenter, wird das Erstellen von neuen Activities vereinfacht, da somit ein klarer Arbeitsablauf gegeben ist. Auch das lesen und editieren von fremden Code wird erleichtert, da von vornherein anhand der Benennung der einzelnen Klassen klar ist welche Aufgaben diese Komponenten übernehmen.

5.3 Besonderheiten

Eventuell können wir hier den Multiplayer einfügen. Oder wir lassen den Abschnitt einfach weg und erklären den Muliplayer in Implementierungsdetails.

6 Anforderungsabgleich

Im Folgenden werden die gegebenen Anforderungen mit der tatsächlichen Implementierung verglichen. Es wird festgestellt ob eine Anforderung ganz, teilweise oder gar nicht erfüllt wurde.

6.1 Funktionale Anforderungen

FA1 Hauptmenü - erfüllt
Das Hauptmenü wurde wie in der Anforderung beschrieben implementiert.

FA2 Spielmodi - erfüllt
Alle Spielmodi wurden wie gefordert implementiert und können vor jedem Spiel ausgewählt werden.

FA3 Computergegner - erfüllt
Für den Einzelspielermodus wurde ein Computergegner implementiert, der ein faires Spielerlebnis bietet. Diese „KI“ kann je nach Schwierigkeitsgrad unterschiedlich gut abschätzen welche Attribut einer Karte die höchsten Gewinnchancen hat, und imitiert so das Verhalten eines menschlichen Spielers.

FA4 Schwierigkeitsgrad - erfüllt
Die geforderten Schwierigkeitsgrade wurden implementiert und können vor dem Beginn eines neuen Spiels ausgewählt werden. Die Wahl des Schwierigkeitsgrads beinflusst das Verhalten der KI.

FA5 Spiel fortsetzen - erfüllt
Nach jedem Spielzug wird der komplette Zustand des Spiels in die interne Datenbank geschreiben und damit persistent gespeichert.

FA6 Gallerie - erfüllt
Die Gallerie wurde wie in „Implementierungsdetails“ beschreiben implementiert.

FA7 Deck Download - erfüllt
Der Deckdownload wurde wie in „Implementierungsdetails“ beschreiben in die Gallerie integriert.

FA8 Statistiken - erfüllt
Die Statistiken wurden implementiert und werden graphisch aufbereitet in einer speziellen Ansicht dargestellt.

FA9 Rangliste - erfüllt
Die beschreibene Ranglist wurde implementiert. Der Spieler wird mit dem Namen der zu Beginn eines Spiels eingegeben wurde, automatisch mit seiner erreichten Punktezahl in die Rangliste aufgenommen

FA10 Achievements - teilweise erfüllt
Es wurde eine Oberfläche zur Anzeige der Achievments implementiert. Ein fertiges Achievementsystem ist allerdings nicht in der App vorhanden.

FA11 Quartetteditor - nicht erfüllt
Der Quartetteditor wurde zu Gunsten des Multiplayers entfernt.

FA12 Levelsystem - nicht erfüllt
Das Levelsystem wurde zu Gunsten des Multiplayers entfernt.

FA13 Multiplayer - erfüllt
Der Multiplayer wurde mit Hilfe der Google Play Services implementiert und erlaubt es zwei Benutzer der App über das Internet gegeneinander anzutreten.

6.2 Nicht Funktionale Anforderungen

NFA1 Robustheit - teilweise
Der Multiplayer weist aufgrund der Komplexität und fehlender Entwicklungszeit noch einige Stabilitätprobleme auf. Ansonsten konnten während der Entwicklung keine Fehler festgestellt werden die Robostheit der App beinträchtigen.

NFA2 Erweiterbarkeit - erfüllt
Durch die Verwendung des Model-View-Presenter Patterns ist eine einfache Erweiterbarkeit gegeben.

NFA3 Responsiveness - erfüllt
Durch den permanenten Einsatz von asynchronen Programmiertechniken wurde sichergestellt, dass die Applikation immer entsprechend schnell reagiert.

NFA4 Usability - erfüllt
Um die Usability sicher zu stellen wurden die Google Human Interface Guidelines als Referenz für die Gestaltung der Oberflächen verwendet.