1 Einleitung

1.1 Motivation

1.2 Ziel

1.3 Aufbau

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 Computer-

gegner 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 Nichtfunktionale 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 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

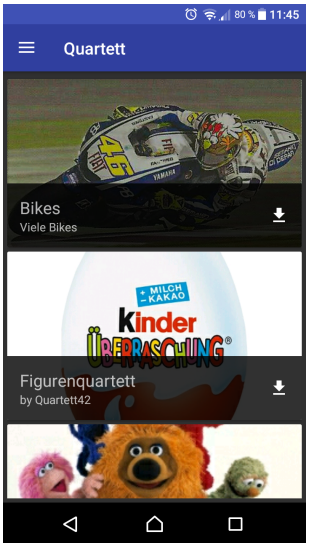

Abbildung 1: Deckansicht

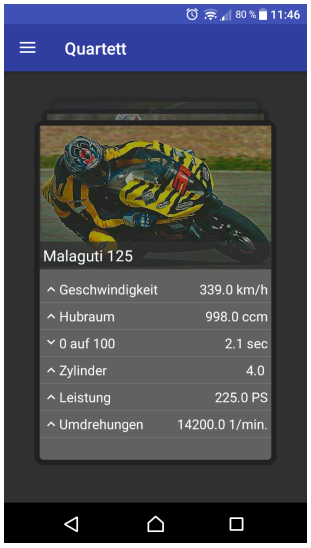

Abbildung 2: Kartenansicht

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 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

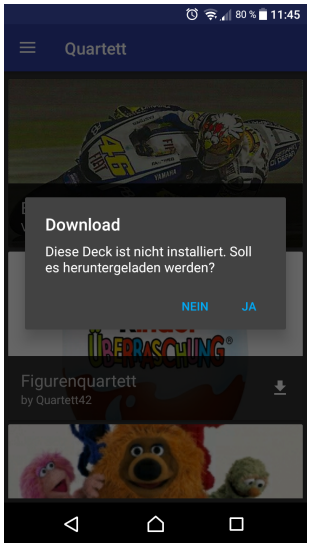

Abbildung 3: Deckansicht

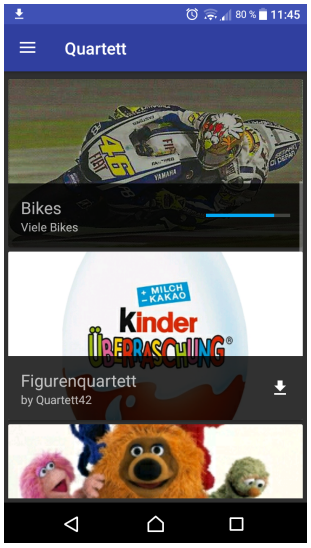

Abbildung 4: Kartenansicht

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 entstehen. 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

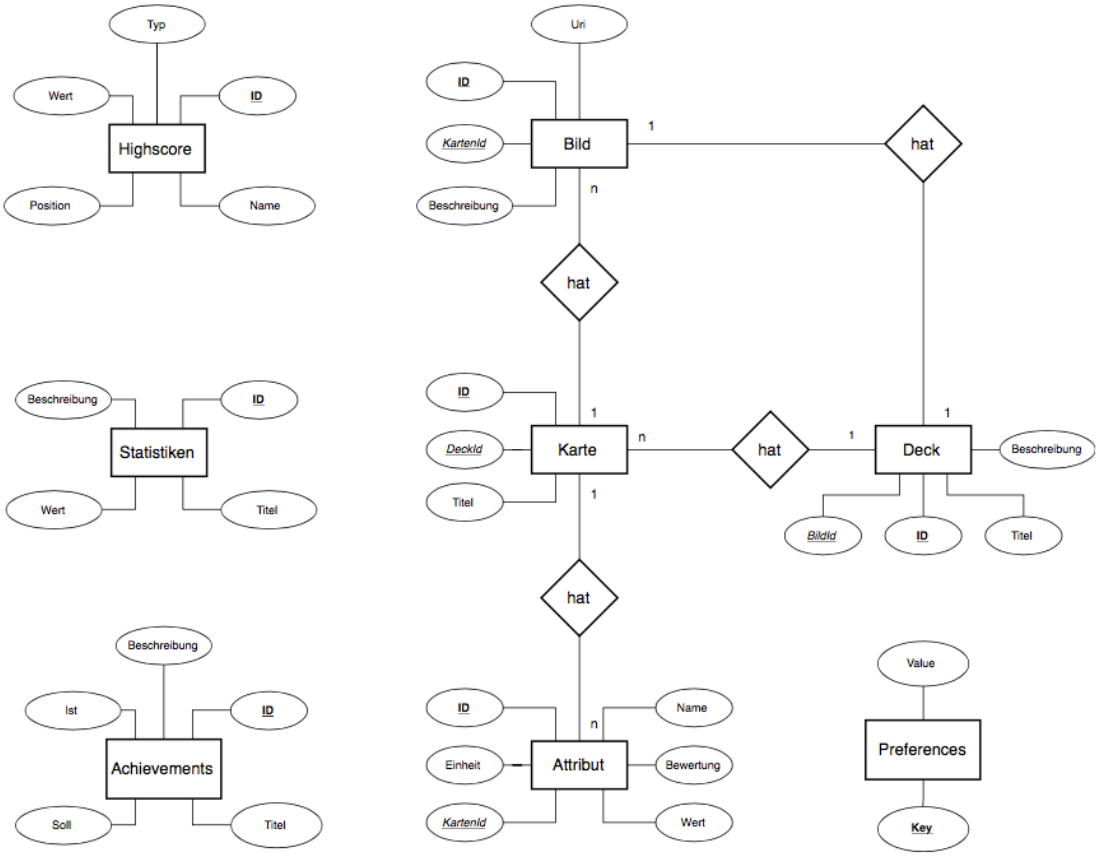

Abbildung 5: ER Diagramm des Datenmodells

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.

5.2.2 Klassenstruktur

5.3 Besonderheiten

5.3.1 Model-View-Presenter

6 Anforderungsabgleich