

Projektgruppe Poker

Sebastian Bäcker
Sebastian Schüler
Silas Schwabe
Nicolai Schulz

Inhaltsverzeichnis

1. Beschreibung	2
2. Datenmodell	3
3. Anwendungsfälle	4
3.1 Login - Page	5
3.2 Profile - Tab	6
3.3 Lobby - Tab	7
3.4 Friends - Tab	8
4. Systemarchitektur	10
4.1 verwendete Systeme	12
4.2 Kommunikation der Systeme	13
5. Technische Umsetzung	14
5.1 Benötigte Werkzeuge zur Entwicklung	14
5.2 Android SDK einrichten	15
5.3 Einrichtung der Webstorm IDE	17
5.4 verwendete bibliotheken	18
5.5 Projektstandard	19

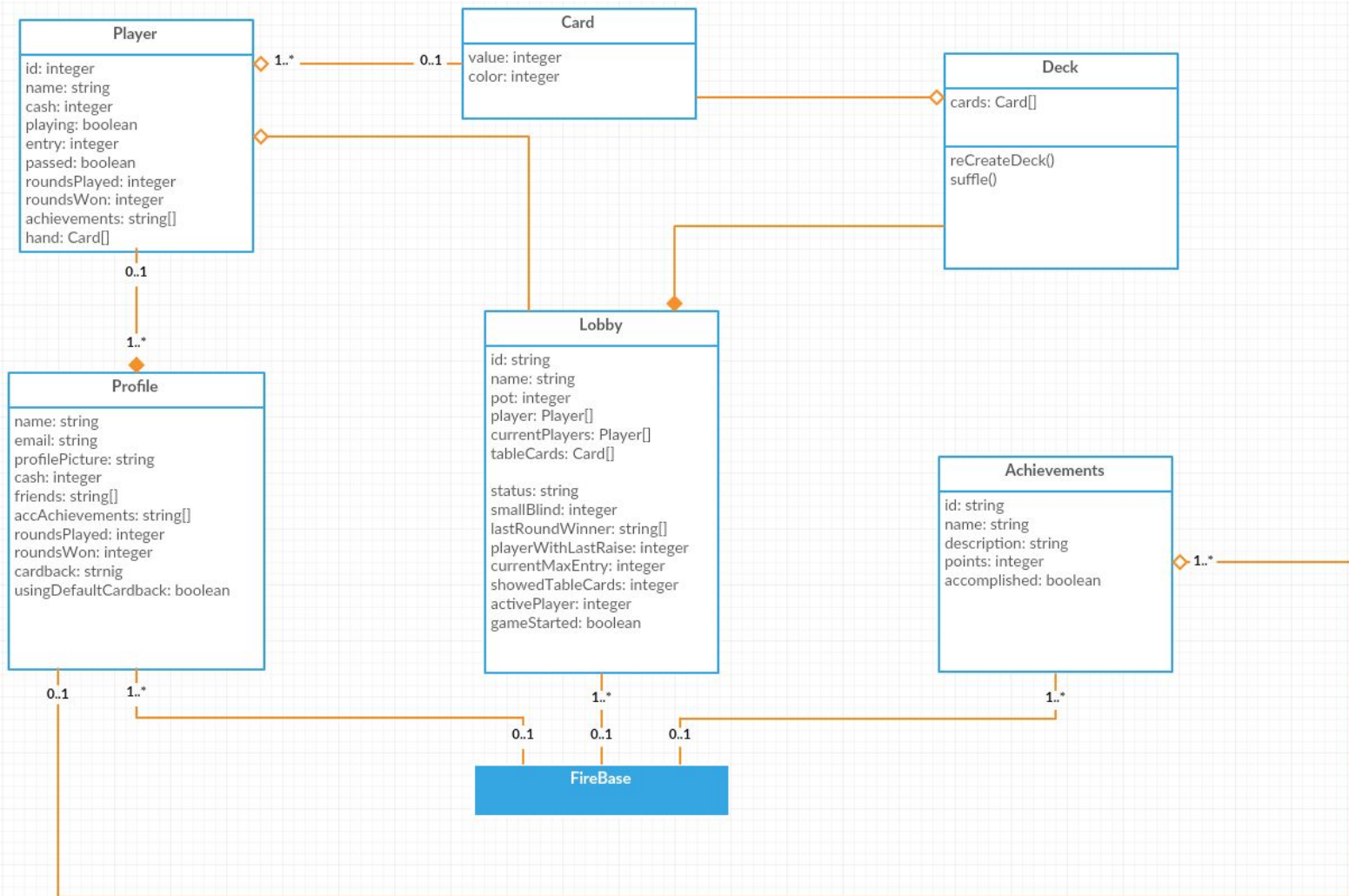
1. Beschreibung

Mit der in diesem Artikel beschriebenen App ist es möglich, ein Poker-Spiel nach den Regeln des Texas hold'em zu spielen. Um einen Einblick in die App zu erhalten, ist es notwendig, einen Account zu erstellen und sich einzuloggen. Dazu wird benötigt: eine E-Mail Adresse, ein Passwort mit mindestens 6 Zeichen, sowie ein Account-Name. Weiterhin ist es notwendig, eine permanente Internetverbindung zu besitzen, während die App verwendet wird. Die Verbindung wird benötigt, um sich erfolgreich einzuloggen, sowie um ein Spiel starten zu können. Der Nutzer der App kann dabei die Rolle eines "Leaders" einnehmen, indem er eine Lobby zum Spielen erstellt.

Nachfolgend sind die weiteren wichtigsten Funktionen beschrieben:

- Hat sich ein Spieler schon mal bei der App eingeloggt, dann wird er beim nächsten Start automatisch eingeloggt.
- Ist man eingeloggt, hat man die Wahl zwischen 4 Tabs:
 - Profilseite
 - Profilbild kann geändert werden.
 - Anzahl der Coins und erreichten Achievements wird angezeigt.
 - Liste der erreichten und noch offenen Achievements kann angezeigt werden.
 - Lobbies
 - Es kann eine Lobby erstellt werden
 - Es kann einer bestehenden Spielelobby beigetreten werden
 - In einer Lobby kann man mit den einem zur Verfügung stehenden Coins an einem Spiel mit bis zu 4 weiteren Spielern teilnehmen
 - Es kann auch einer schon bestehenden Lobby (und einem schon laufenden Spiel) beigetreten werden, sofern die maximale Spieleranzahl nicht überschritten wird
 - Freundesliste
 - Freunde können über ihre E-Mail hinzugefügt werden.
 - Profilseiten von Freunden können angesehen werden.
 - Freunde können aus der Freundesliste gelöscht werden.
 - Freunde können durch Rankings sortiert werden (wer hat am meisten Coins, Punkte durch Achievements?)
 - Einstellungen
 - Impressum
 - Eigene Kartenrücken können hochgeladen werden.
 - Darstellung der Kartenrücken anderer Spieler kann ein-/ausgeschaltet werden.
 - Die Farbe des App-Layouts kann verändert werden.
 - Man kann sich ausloggen.

2. Datenmodell







Primär sind auf Firebase (dem Server) die Lobby, die Achievements und die Profile inklusive ihren Anmeldedaten gespeichert. Über Subscriptions lassen sich die Daten auf den Devices aktuell halten.

3. Anwendungsfälle

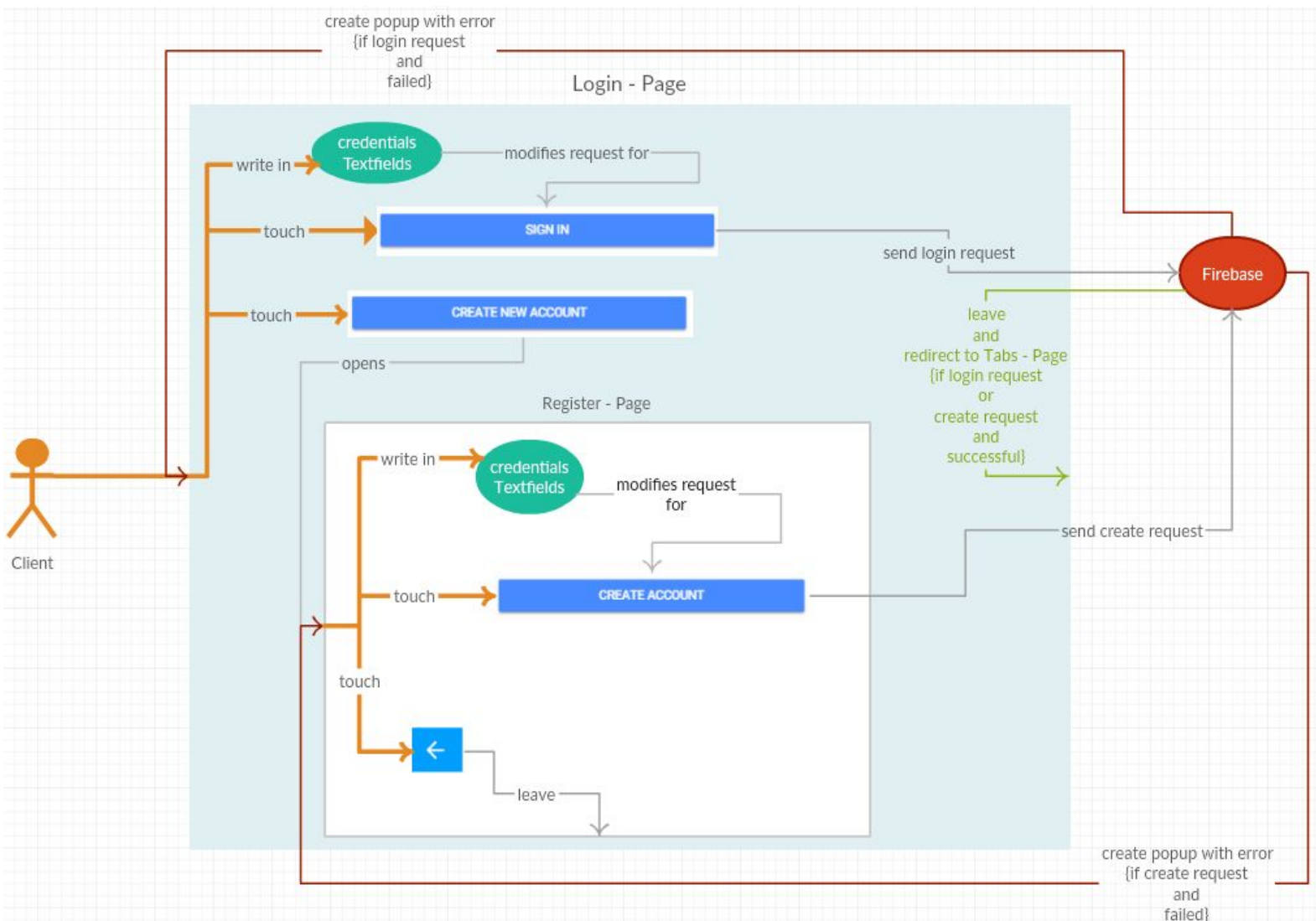
In diesem Kapitel wird beschrieben, welche Aktionen auf jeder Seite vom Benutzer ausgelöst werden können.

Dazu wurden Diagramme angefertigt, die die dynamischen Vorgänge etwas visueller darstellen sollen. Dabei haben folgende Farben eine zugeordnete Bedeutung:

-  orange sind Aktionen, die vom Benutzer ausgelöst werden können.
-  grau sind Aktionen, die von der App (im Hintergrund) ausgeführt werden. Deshalb folgt einem orangenen Pfeil oft ein grauer Pfeil.
-  grün sind Aktionen, die als Folge einer wahren booleschen Auswertung passieren sollen.
-  rot sind Aktionen, die als Folge einer nicht wahren booleschen Auswertung passieren sollen.

Andere Farben dienen lediglich der Übersichtlichkeit.

3.1 Login - Page

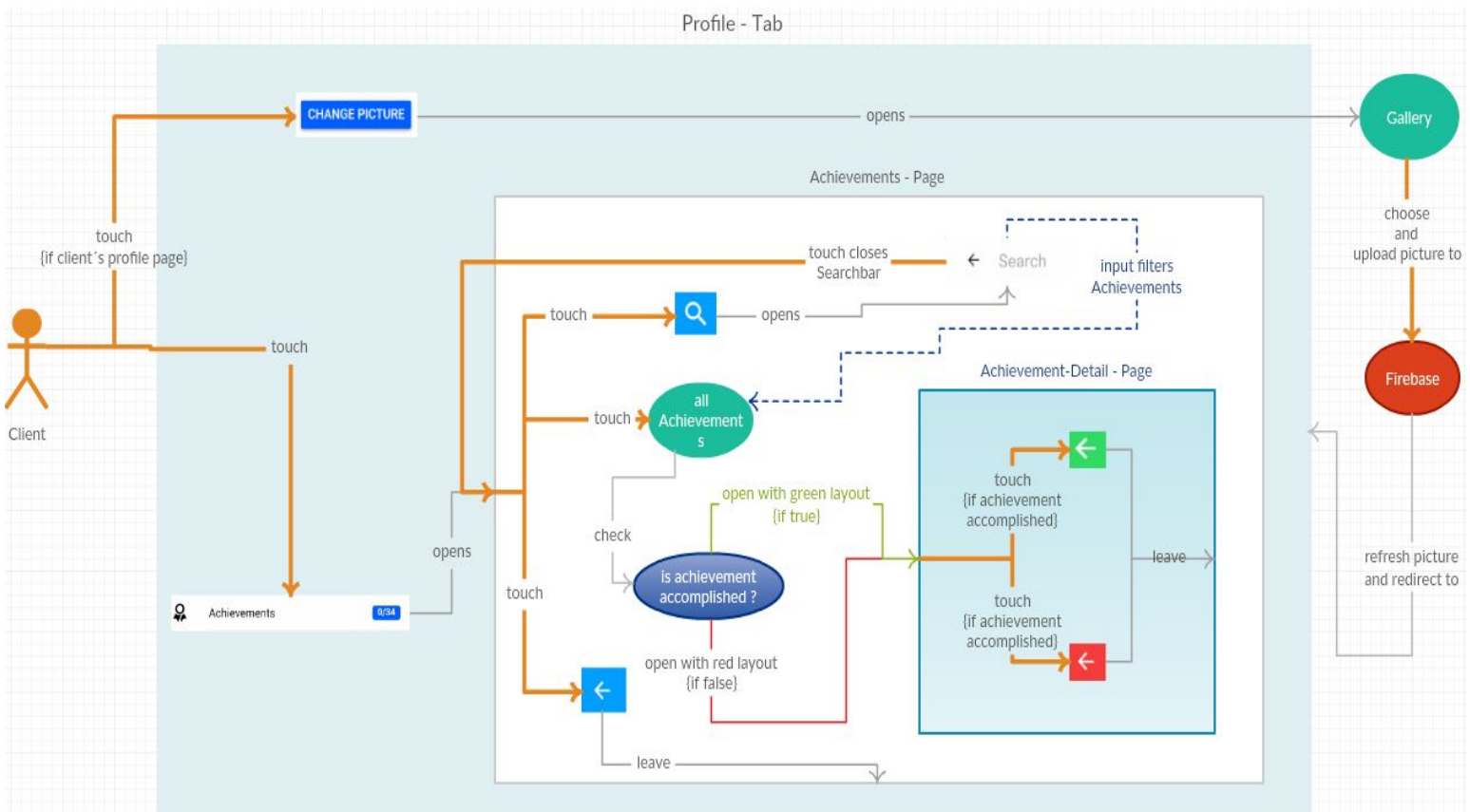


Wenn der Benutzer die App das erste Mal startet, dann gelangt er in die Login - Page. Dort kann er sich entweder einloggen (sign in) oder einen Account erstellen (create new account).

Anmerkungen:

- "credentials Textfields" auf der Login Page sind {E-Mail, Passwort}
- "credentials Textfields" auf der Register Pager sind {Displayname, E-Mail, Passwort, Confirm Password}

3.2 Profile - Tab



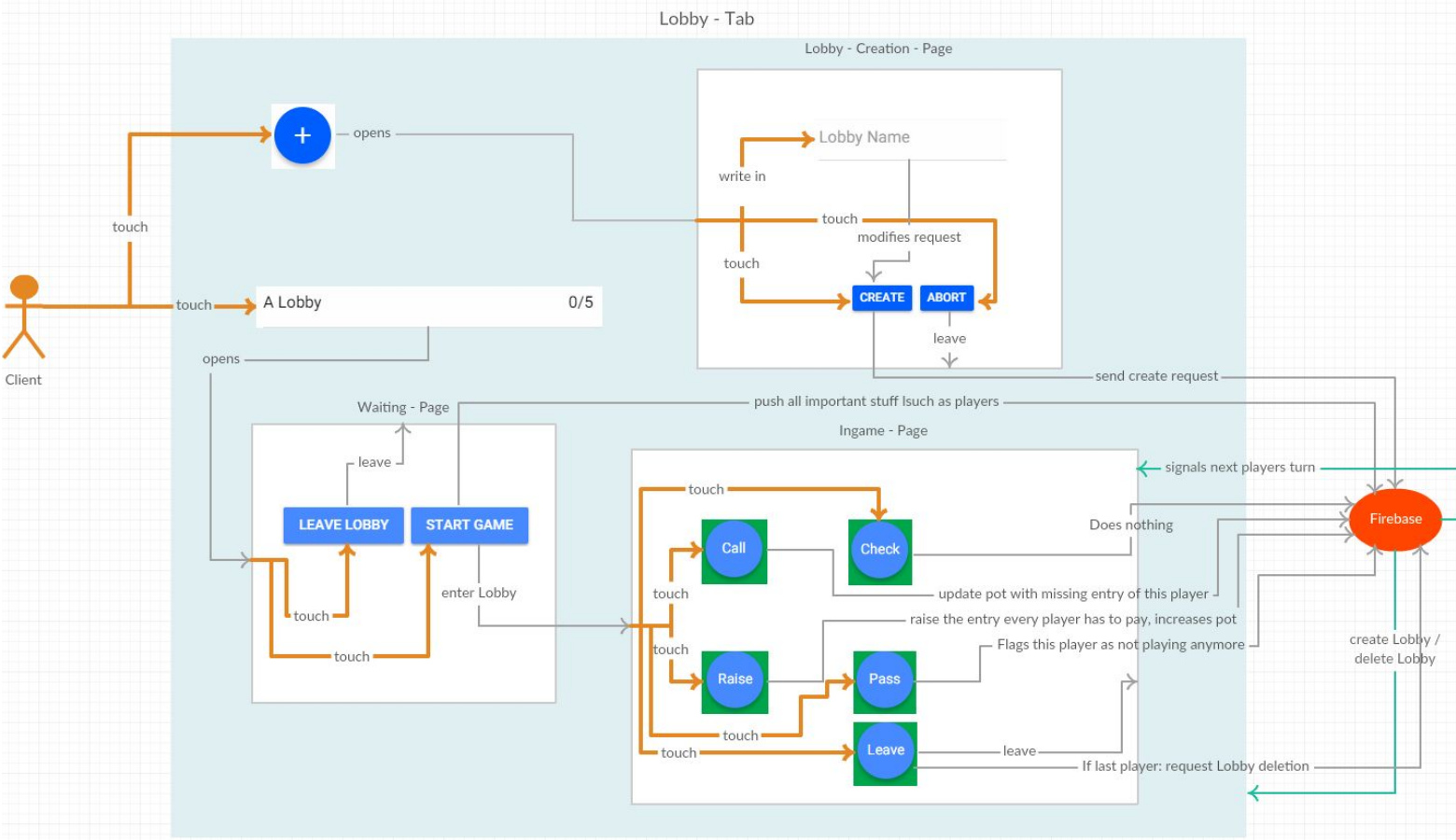
Die Profilseite ist der erste Tab der Tabspage.

Hier kann der Benutzer die erreichten Achievements näher betrachten (Achievements) und, wenn es sein eigenes Profil ist, das Profilbild ändern (change picture).

Anmerkungen:

- Will der Benutzer erstmalig sein Profilbild wechseln, dann wird er nach den Berechtigungen für die native Gallerie-Applikation gefragt. Das wurde aus Gründen der Übersichtlichkeit nicht eingezeichnet.

3.3 Lobby - Tab

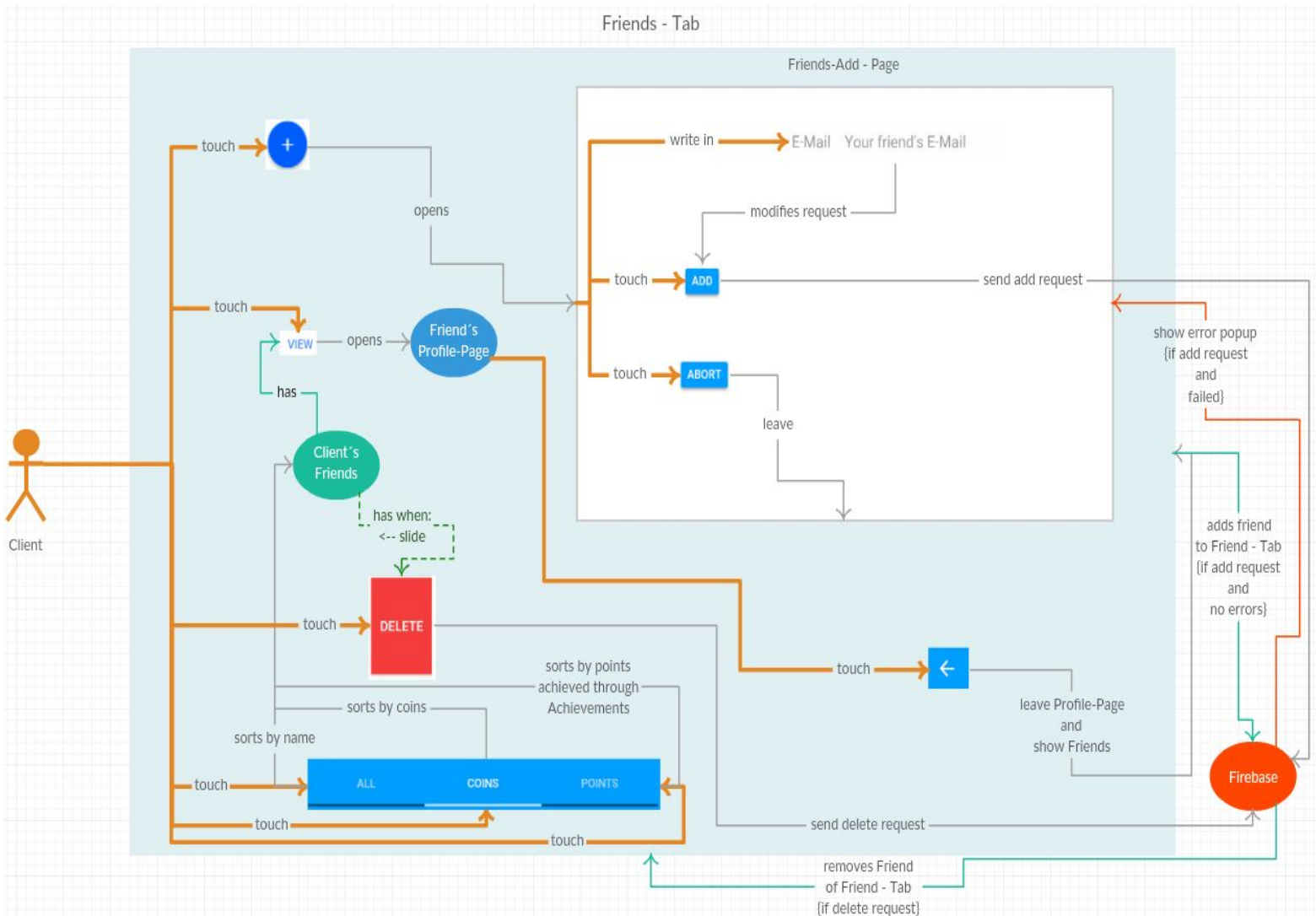


Die Lobbyseite ist der zweite Tab der Tabpage.

Hier kann der Benutzer neue Lobbys erstellen und diese betreten.

Wenn eine Lobby betreten wird kann man, sofern man mindestens zu zweit ist, anfangen zu spielen. wenn jemand diese Lobby betritt, so wird er nach dem Ende der gerade laufenden Runde in das Spiel gelassen (Er muss vorher in der Waiting-Page warten). Wenn man die Lobby verlässt und keine Spieler mehr vorhanden sind so wird die Lobby gelöscht.

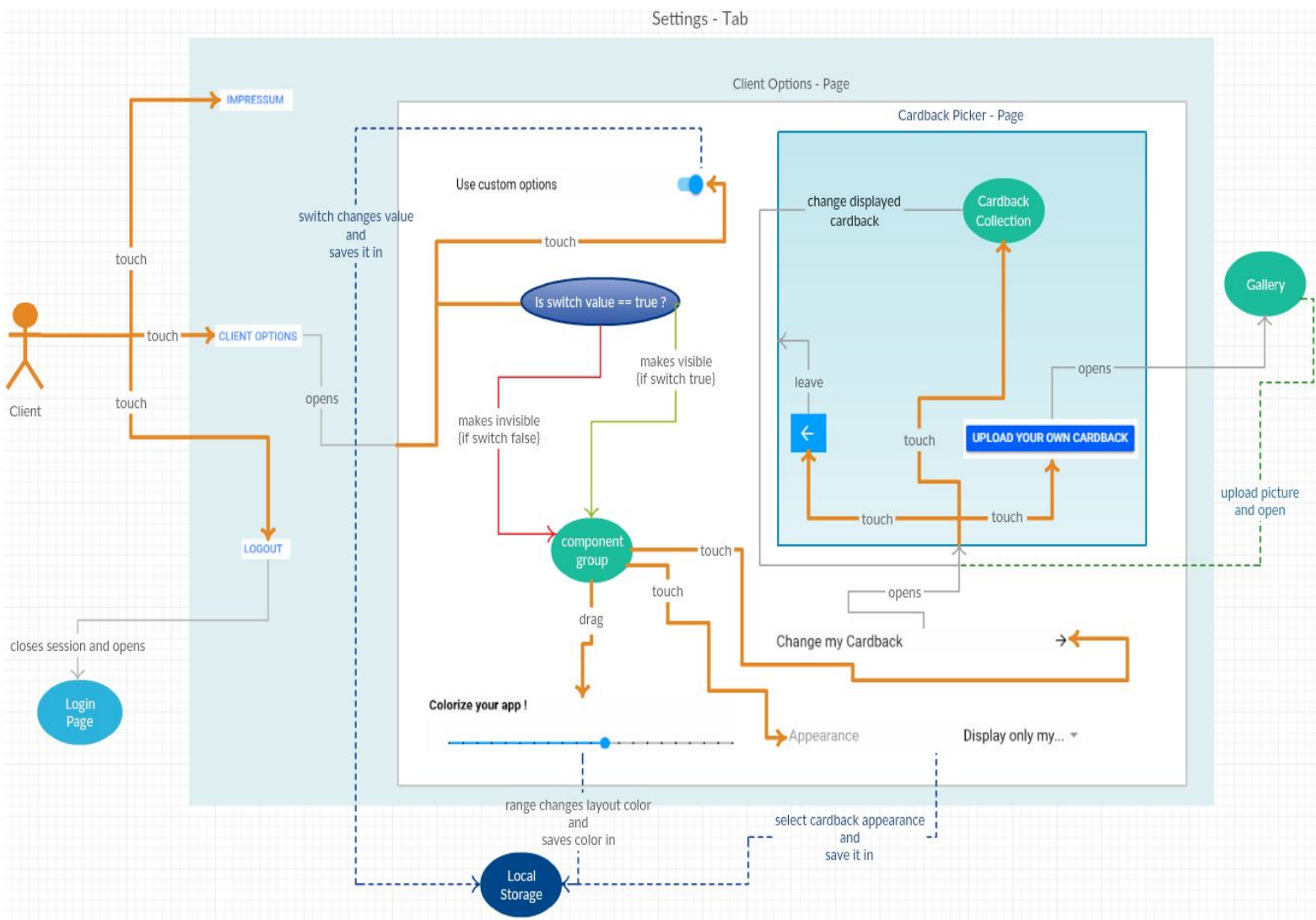
3.4 Friends - Tab



Die Freundesliste ist der dritte Tab der Tabspage.

Hier kann der Benutzer die Profile seiner Freunde ansehen (view) oder alternativ auch seine Freunde verwalten. Für die Verwaltung kann er Freunde hinzufügen (+), löschen (delete) oder nach Kriterien sortieren(all, coins, points).

3.5 Settings - Tab



Die Einstellungen sind der vierte Tab der Tabspage.

Hier kann der Benutzer das Impressum (Impressum) einsehen, die Darstellungen in seinem Client anpassen (Client Options) oder sich ausloggen (logout).

Auf der Client Options - Page kann der Benutzer die Farbe des Layouts anpassen (colorize your app!), die Darstellung der Kartenrücken anderer Spieler ändern (Appearance) oder auch einen eigenen Kartenrücken hochladen (Change my Cardback).

4. Systemarchitektur

Die Login-Page stellt den Einstiegspunkt der App dar. Beim Erstellen dieser Page wird initial ein Kommunikationskanal zu Firebase-Authentifizierung erstellt. Es wird geprüft, ob sich auf diesem Gerät bereits mit einem bestehenden Account eingeloggt wurde. Ist dies der Fall, findet ein automatischer Login mit dem verwendeten Account statt. Wurde sich noch nicht eingeloggt, bzw. existiert kein Account, so kann ein neuer Account erstellt werden. Dazu ist ein Pagewechsel auf die Registry-Page notwendig. Hier müssen alle geforderten Eingaben getätigt werden. Durch den "Create new Account" Button wird ein Request zu Firebase-Authentifizierung hergestellt. Das heißt im Grunde, werden die Angaben noch einmal überprüft, wie bspw. die Stärke des Passwortes. Konnte erfolgreich ein Account erstellt werden, wird man in die App eingeloggt. Ist man eingeloggt, unabhängig davon, ob ein neuer Account erstellt wurde, oder ob man automatisch eingeloggt wurde, besteht über die gesamte Verwendungsdauer der App eine Subscription ([RxJs-Observable](#)) auf den Zustand des Users. Das setzt voraus, dass der User bei der Verwendung der App stets eine Internetverbindung hat. Direkt nach dem Login gelangt man auf die Lobby-Page. Beim Erstellen der Page, wird ein Kommunikationskanal zur Firebase Realtime Database hergestellt. Diese Komponente dient bei der gesamten Anwendung als Backend und wird für die Datenpersistenz eingesetzt. Jegliche Änderungen an den Daten werden sofort allen Beobachtern dieser Daten mitgeteilt (zu vergleichen mit dem [Observer-Pattern](#)).

Wenn man auf der Lobby-Page nun eine neue Lobby erstellt, so wird diese in der Firebase-Database eingetragen und eine Subscription auf diese erstellt. Man kann diese Lobby nun betreten. In diesem Fall wird eine Subscription des eigenen Profils angefordert. Man kann nun während dem Spiel auf die Daten des eigenen Profils zugreifen um bspw. die Achievements oder das Geld des Spielers im Profil aktualisieren zu können.

Die meisten Berechnungen bezüglich der Logik werden von einem Spieler, dem Host, durchgeführt. Dieser aktualisiert regelmäßig (nach jeder Berechnung) die Database, so dass die anderen Spieler entsprechend immer auf dem aktuellsten Stand sind. Verlässt der Host die Lobby, so rücken alle Spieler nach und ein neuer Host führt alle Berechnungen durch. Beim Verlassen der Lobby wird die Subscription des Profils und der Lobby wieder gekündigt. Wenn alle Spieler die Lobby verlassen haben wird die Lobby gelöscht.

Die Profile-Page erstellt 2 Subscriptions, zum einen die zentrale Beobachtung des Profils von dem angemeldeten Benutzer und als nächstes eine für alle erreichbaren Achievements. Die Profilseite wurde so gestaltet, dass diese auch das Profil eines Freundes in der Anwendung darstellen kann. Dem Benutzer wird die Möglichkeit geboten, sein Profilbild zu ändern. Dazu werden Native-Funktionen benötigt, die es

ermöglichen Dateien auf dem Endgerät anzuzeigen und zu verarbeiten. Mit dem Plugin ImagePicker wird in einer Art Webview die Gallery des Smartphones dargestellt. In dieser Gallery kann der Benutzer ein Bild auswählen und als neues Profilbild hochladen. Das Bild wird im Firebase-Storage gespeichert. Das Bild wird in einen Base64 String kodiert, dazu wird das Native-Plugin File von Cordova verwendet. Das kodierte Bild wird anschließend im Verzeichnis des Benutzers abgelegt. Weiterhin ist es möglich sich auf seiner Profilseite, über einen Pagewechsel, alle Achievements anzeigen zu lassen. Die vom Benutzer bereits erreichten werden farblich hervorgehoben. Die Friend-Page erstellt initial bei ihrer Erstellung 2 weitere Subscriptions, eine für Achievements die zur Überprüfung der erreichten Achievements des Benutzers verwendet werden, eine weitere für das Profil des Benutzers. Die Toolbar mit den 3 Reitern ermöglicht eine Sortierung der vorhandenen Freunde. Mit einem Pagewechsel durch einen Klick auf den Add-Button kann ein Freund hinzugefügt werden. Auf der Settings-Page kann der Benutzer über Client-Options personalisierte Einstellungen für seine App vornehmen. Für den Kartenrücken wird zur Persistenz der Firebase-Storage verwendet. Alle weiteren Optionen, wie bspw. die Einstellung der Farbe wird im localStorage der Webapplikation gespeichert.

4.1 verwendete Systeme

Angular:

Ist ein JavaScript-Webframework zur Erstellung von Single-Page-Applikationen unter Verwendung des Model-View-ViewModel-Pattern.

<https://angular.io/>

PhoneGap/Cordova:

Hierbei handelt es sich um ein Framework zur Erstellung von hybriden Applikationen für mobile Geräte. Die Applikation wird in JavaScript,HTML und CSS entwickelt, anstatt spezifischer Programmiersprachen für das jeweilige Endgerät.

<https://cordova.apache.org/>

Ionic:

Das Ionic-Framework dient zur Erstellung hybrider Apps. Dabei basiert es auf Angular, welches die Struktur der App erstellt und Cordova, welches den nativen Container bietet. Ionic realisiert hauptsächlich die Darstellung der Benutzeroberfläche.

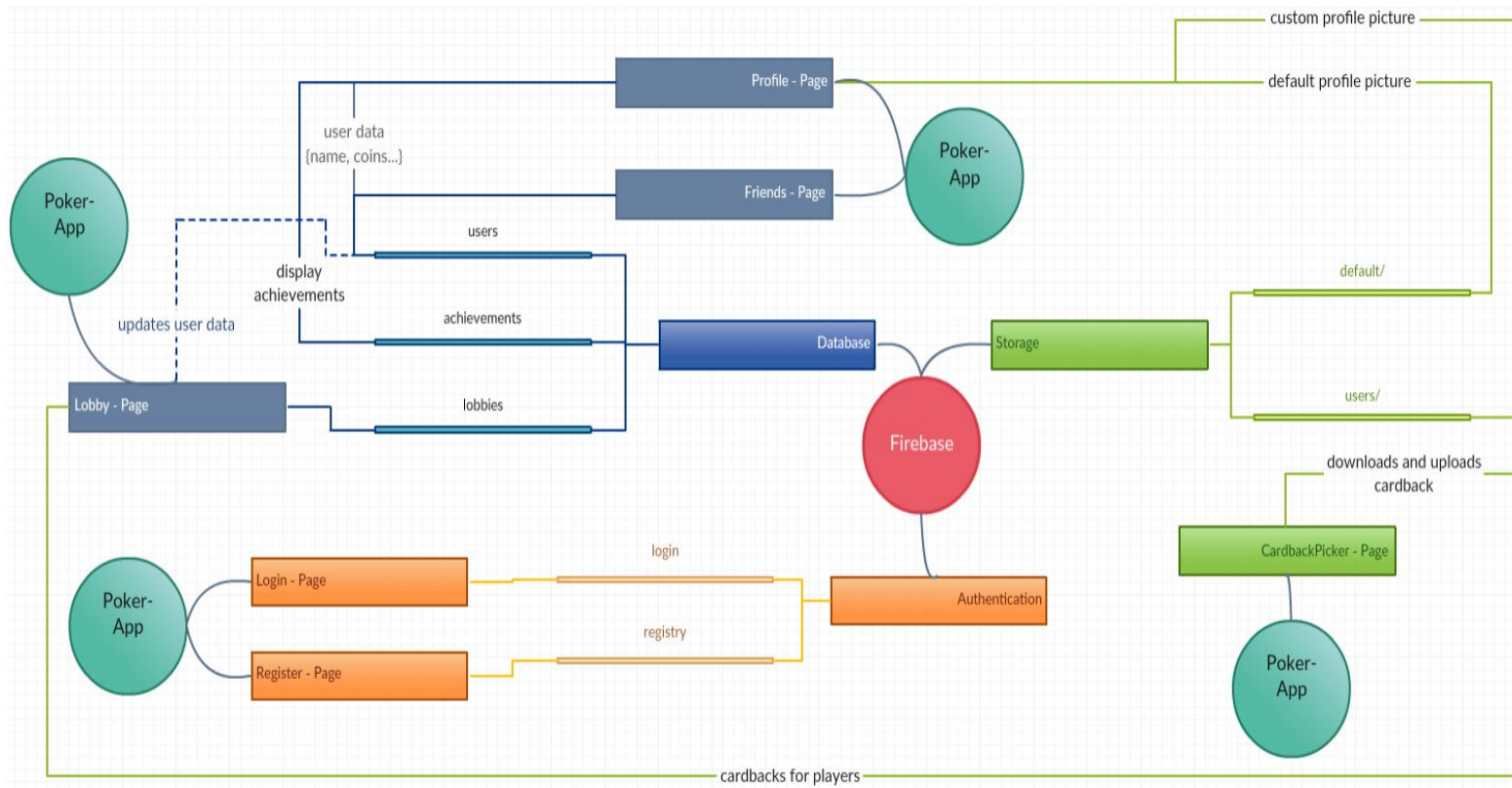
<https://ionicframework.com/>

Firebase:

Das Framework kapselt mehrere Funktionalitäten, wie Realtime-Database, Authentifizierungsservice, Storage, Functions... Dabei wird seitens der Entwickler keine weitere Rechenkapazität benötigt. Alle Berechnungen und Speicherungen finden auf der Infrastruktur von Google statt.

<https://firebase.google.com/>

4.2 Kommunikation der Systeme



5. Technische Umsetzung

5.1 Benötigte Werkzeuge zur Entwicklung

- Java Development Kit (JDK) <8.141>
 - <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- Node.js <7.10.1>
 - <https://nodejs.org/download/release/v7.1.0/>
- Node Package Manager (npm) <4.2.0>
 - wird automatisch mit node.js installiert
- Git <2.7.4>
 - <https://git-scm.com/downloads>
- Apache Cordova <7.0.1>
 - wird global über npm installiert: npm install -g cordova (Linux und OS benötigen root, bzw sudo Rechte)
- Ionic Framework <3.5.0>
 - wird global über npm installiert: npm install -g ionic (Linux und OS benötigen root, bzw sudo Rechte)
- Android Studio <2.3.3>
 - <https://developer.android.com/studio/index.html>
- Webstorm IDE <2017.2>
 - <https://www.jetbrains.com/webstorm/download/>
- Google Chrome
 - <http://www.google.de/intl/de/chrome/browser>

5.2 Android SDK einrichten

Das Android Software Development Kit (Android SDK) wird automatisch mit der Installation von Android Studio vorgenommen. Damit dieses mit Ionic, bzw. Cordova arbeiten kann, muss die Umgebungsvariable des Betriebssystems angepasst werden. Nachfolgend werden die Schritte für Linux und Windows beschrieben, OS verhält sich etwa äquivalent wie Linux.

Linux:

Die Datei ~/.bashrc muss um folgende Anweisungen erweitert werden:

```
export
ANDROID_HOME=~/.path/to/your/sdk/choosed/while/androidstudio/installation/sdk
export PATH=$PATH:$ANDROID_HOME/platform-tools
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
```

Windows:

Erstellen der Umgebungsvariable:

ANDROID_HOME

mit dem Wert:

path/to/your/sdk/choosed/while/androidstudio/installation/sdk

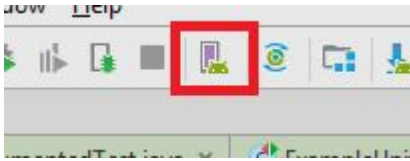
Im Anschluss muss die Path Variable wie folgt erweitert werden:

```
; %ANDROID_HOME%
; %ANDROID_HOME%/platform-tools
; %ANDROID_HOME%/tools
; %ANDROID_HOME%/tools/bin
```

Optional:

Einrichtung eines Android Virtual Device (AVD). Dies wird benötigt, wenn dem Entwickler kein Android Gerät vorliegt.

Dazu muss Android Studio gestartet werden. Anschließend auf den Button AVD Manager klicken :



Im nachfolgend geöffneten Fenster auf “Create Virtual Device” klicken. Anschließend öffnet sich ein Wizard, welches einem beim Anlegen des AVD unterstützt. Die Installation des AVD nach belieben abschließen und beenden.

Testen der Einrichtung:

Durch den Befehl: “**adb version**” kann die erfolgreiche Einrichtung des Android SDK samt den Tools, getestet werden. Die Ausgabe auf der Konsole, sollte etwa wie folgt aussehen:

```
sebastian@sebastian-Lenovo-Yoga-500-14IBD:~$ adb version
Android Debug Bridge version 1.0.39
Revision 3db08f2c6889-android
Installed as /home/sebastian/Android/Sdk/platform-tools/adb
```

Zusätzlich kann die Einrichtung des AVD getestet werden, dazu muss der Befehl “**avdmanager list avds**” eingegeben werden. Die Ausgabe sollte etwa wie folgt aussehen:

```
sebastian@sebastian-Lenovo-Yoga-500-14IBD:~$ avdmanager list avds
Available Android Virtual Devices:
  Name: Nexus_5X_API_24
  Device: Nexus 5X (Google)
  Path: /home/sebastian/.android/avd/Nexus_5X_API_24.avd
  Target: Google Play (Google Inc.)
         Based on: Android 7.0 (Nougat) Tag/ABI: google_apis_playstore/x86
  Skin: nexus_5x
  Sdcard: 100M
```

Mit dem Befehl “**git --version**” kann überprüft werden, ob die Installation von Git erfolgreich durchgeführt wurde.

5.3 Einrichtung der Webstorm IDE

Webstorm dient als primäre Entwicklungsumgebung (IDE). Die Projektdateien werden über Git bereitgestellt. Um sich die entsprechenden Dateien clonen zu können, wird ein Zugriff auf <https://git.thm.de/> benötigt.

Anschließend erstellt man einen beliebigen Ordner im Dateisystem seines Betriebssystems. In diesem Ordner öffnet man eine Konsole und clont das Git-Repo mit folgendem Befehl:

```
"git clone git@git.thm.de:ema-ss17/poker.git"
```

Jetzt kann Webstorm gestartet werden. Im geöffneten Fenster wählt man "Open" und wählt den zuvor geklonten Ordner aus.

In Webstorm im Terminal ist es erforderlich den Befehl:

```
"npm install"
```

auszuführen.

Mit diesem Befehl werden sämtliche Packages installiert, die von diesem Projekt verwendet werden.

Ein weiteres Package, welches von dem Projekt verwendet wird, muss manuell installiert werden, der ImagePicker.

Dazu muss der folgende Befehl ausgeführt werden:

```
"cordova plugin add https://github.com/dhavalsoni2001/ImagePicker.git"
```

(Hierbei kann Fehler 4 auftreten)

Die App kann im Browser betrachtet werden, hier stehen allerdings sämtliche Funktionen, die sich auf ein Smartphone, bzw. Tablet beschränken nicht zur Verfügung. Darunter fallen bspw. die Verwendung der Kamera, die Galerie des Gerätes etc.

Um die App im Browser zu starten muss der nachfolgende Befehl im Terminal im Projektordner ausgeführt werden:

```
"ionic serve"
```

Soll die App auf einem AVD:

```
"ionic cordova emulate android"
```

(Hierbei kann Fehler 5 auftreten)

oder auf einem Gerät:

```
"ionic cordova run android"
```

ausgeführt werden.

Bekannte Fehler:

1. Polyfills für Promise in Firebase.
 - Beim Versuch der Ausführung auf einem Gerät oder einem AVD kann es zu einem Fehler beim Build-Prozess kommen. Mit folgendem Befehl lässt sich dieser Fehler beheben:
 - “npm install promise-polyfill --save-exact”
2. Mehrfach Installation der App in verschiedenen Versionen
 - Failure [INSTALL_FAILED_UPDATE_INCOMPATIBLE]
 - Dieser Fehler lässt sich beheben, indem man die App auf dem Device manuell einmal deinstalliert und dann den ursprünglichen Befehl wiederholt.
3. Gradle Wrapper.
 - Could not find gradle wrapper within Android SDK
 - Fehler lässt sich beheben, indem man Gradle manuell installiert:
<https://docs.gradle.org/current/userguide/installation.html>
4. Error: Current working directory is not a Cordova-based project.
 - Kann durch ein ausführen von “ionic serve” behoben werden. Im Anschluss an dieses Command, einfach den ursprünglichen Command erneut ausführen.
5. Failed to install 'cordova-plugin-screen-orientation': CordovaError: Failed to fetch plugin
 - Dieser Fehler tritt meist beim ersten ausführen der App auf. Also beim Aufruf von “ionic cordova run/emulate android”. Er lässt sich beheben, indem der Befehl einfach erneut ausgeführt wird.

5.4 verwendete bibliotheken

Hier ist eine Auflistung der explizit hinzugefügten Bibliotheken des Projekts zu finden.

- **Firestore <4.1.3>**
- **Angularfire2 <4.0.0-rc.1>**
- **ImagePicker**
- **cordova-plugin-file <4.3.2>**
- **cordova-plugin-screen-orientation <2.0.1>**

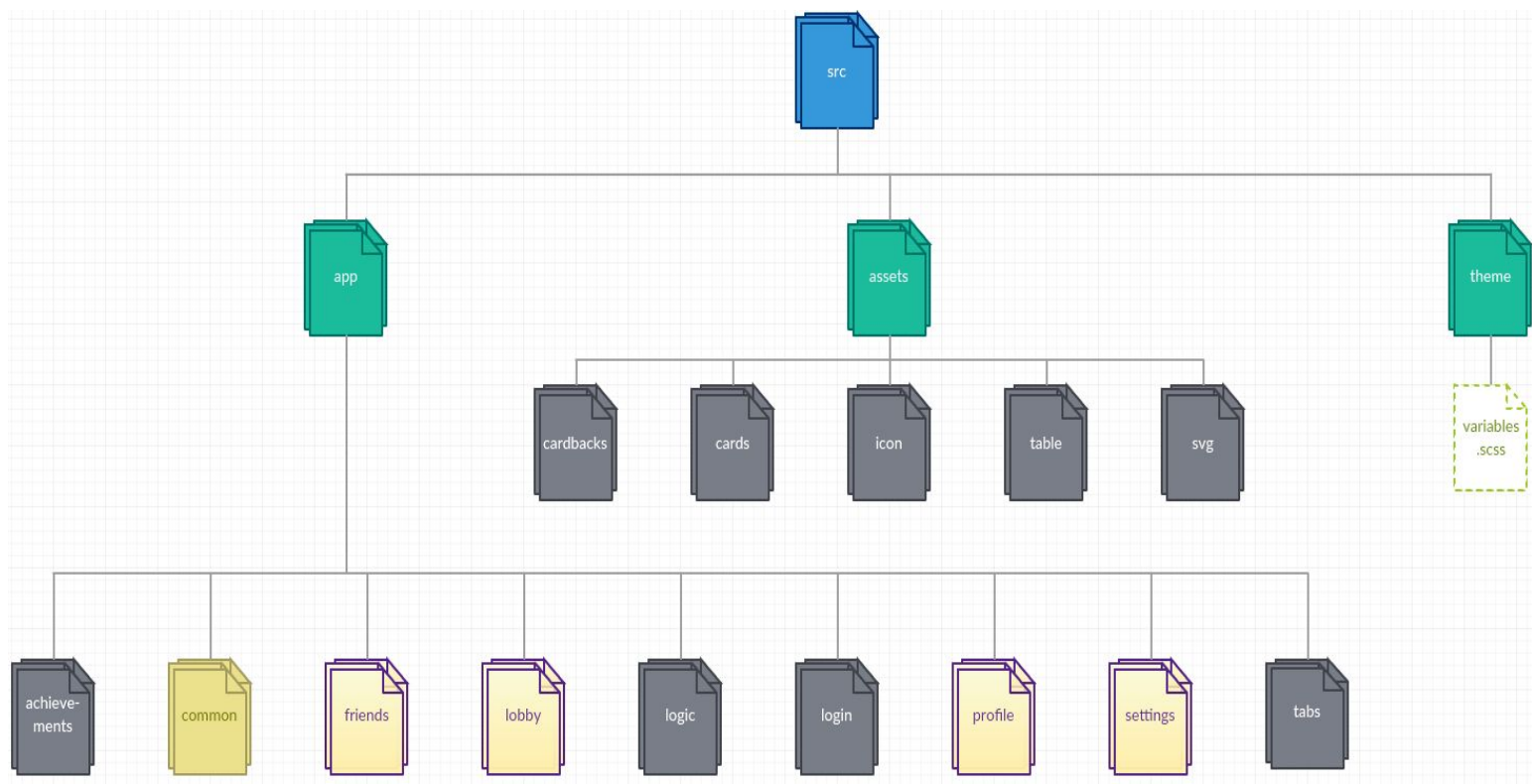
Alle weiteren verwendeten Bibliotheken finden sich in der **package.json** direkt im Projekt-Ordner.

5.5 Projektstandard

Bei der Umsetzung der App wurde sich an verschiedene Konventionen des Angular-Style-Guide gehalten.

Jede Angular-Komponente, sprich Component, Service, Module oder HTML-Template wird in eine eigene Datei geschrieben ([Single-Responsibility-Principle](#)). Wobei die Benennung der Datei und der Klassenname einer weiteren Konvention folgen. Wird eine Datei erstellt, die eine Angular-Component abbilden soll, so ist dies im Dateinamen wie folgt anzuzeigen:

“my-page.component.ts. Der Klassenname wird analog für jeden Bindestrich im Camel-Case verfasst. Der Klassenname für die zuvor genannte Datei wäre sinngemäß: “MyPageComponent” ([Symbols and Filenames](#)). Bei Angular-Apps ist es üblich, dass sich die Umsetzung der App im Ordner “app” befindet (.html, .ts, .js, .scss, .css). Die Projektverzeichnis Struktur wurde wie auf dem folgenden Bild ersichtlich umgesetzt. Dabei wurde wenn möglich ein Ordner für jedes Feature angelegt ([Folders-by-Feature](#)). Eine Ausnahme ist der Ordner “common”, hier befinden sich Dateien, die allgemein genutzt werden können.



{friends, lobby, profile, settings} sind die vier Tab-Seiten.