Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

HTW Berlin

Fachbereich 4 - Informatik, Kommunikation und Wirtschaft

BA - Angewandte Informatik

Semesterbegleitende Arbeit - KilterVote

B35 - Mobile Betriebssysteme und Netzwerke

WiSe - 24/25

Prof. Dr. Thomas Schwotzer

Bearbeitende:	Yannik Schüler
	Simon Cornelius
Matrikelnummern:	583880
	577602

Inhaltsverzeichnis

1 Projektidee

1.1 Beschreibung

Die Applikation soll zum Wählen von Routen am Kilter Board in einer Boulderhalle helfen, denn oftmals ist es schwer eine unparteiische Wahl zwischen verschiedenen Schwierigkeitsgeraden der Routen zu gewährleisten - gerade wenn eine weite Spanne von Erfahrungsstufen unter den Boulderern vertreten ist. Jeder Boulderer kann Routen zu einem Auswahl-Pool hinzufügen und diese mit einem bestimmten Schwierigkeitsgerad versehen, dann kann jeder Boulderer für eine Route stimmen und ein Wahl-Algorithmus, basierend auf den Erfahrungsstufen der Teilnehmer, wählt dann eine Route aus. Während des Kletterns tragen die Boulderer ihre Versuche auf jeder Route in die App ein, um am Ende eine Auswertung zu erhalten. Auf Wunsch können die persönlichen Statistiken auch gespeichert werden, sowie Statistiken zwischen bestimmten Boulderern (Rivalitäten).

1.1.1 Use Cases

1. Session erstellen oder einem Session beitreten durch Angabe eines Session Codes (möglicherweise auch via NFC)
2. Angabe eines Namens und dem eigenen Erfahrungslevel im Bouldern
3. Hinzufügen k/einer oder mehrerer Routen zum Route Pool
4. Ansehen der verschiedenen Routen im Pool
5. Wählen für eine Route
6. Loggen der Versuche mit Fortschritt (in Prozenten), sowie dem Erreichen eines Checkpoints
7. Ändern des eigenen Erfahrungslevels während einer laufenden Session
8. Anzeigen von Statistiken der aktuellen Session
9. Verlassen der laufenden Session

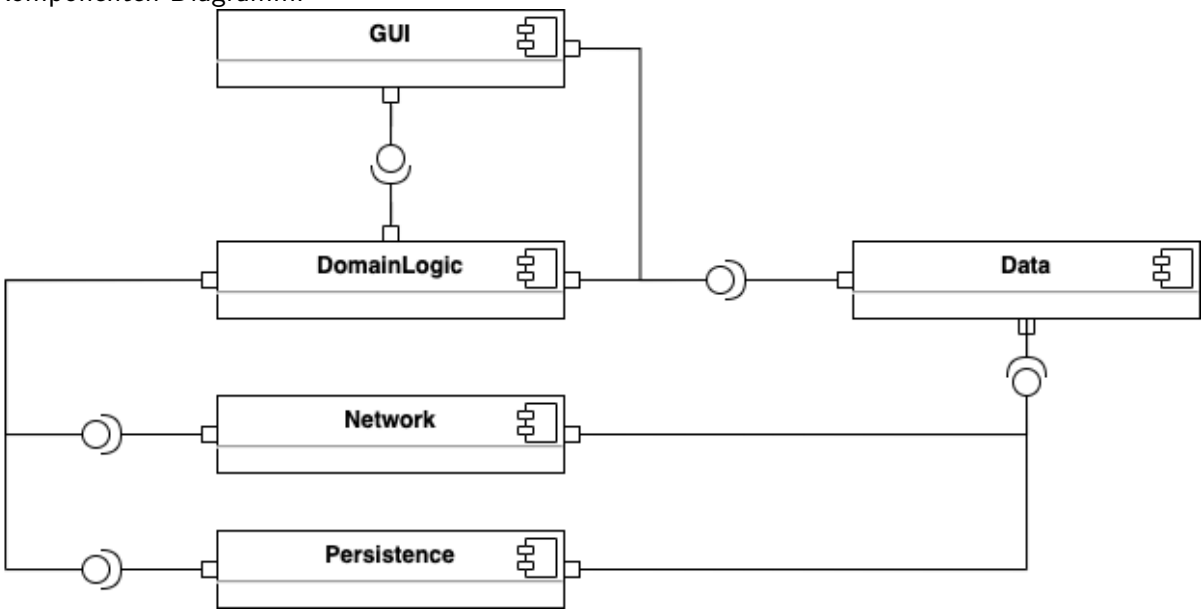
1.2 Wireframes

1.3 Einordnung in die Anforderungen

Die Applikation basiert auf einem Ad-hoc Netzwerk welches zwischen den Geräten der Boulderer aufgebaut wird. Dafür ziehen wir Bluetooth, sowie WiFi-Direct in Betracht, im Optimalfall wird die Möglichkeit für beides geboten. Die Statistiken der Boulderer und die Versuche während der Sessions werden auf dem persönlichen Gerät gespeichert und dann im Ad-hoc Netzwerk gepublished, somit liegen die Daten immer verteilt auf mehreren Geräten. Lässt es der zeitliche Rahmen zu besteht auch die Möglichkeit NFC für das Beitreten/Erstellen einer Session zwischen Boulderern zu nutzen.

2 Komponenten

Komponenten Diagramm:



Ganz grob besteht die Applikation aus 5 Komponenten.

1. GUI
2. Data
3. Domain Logic
4. Network
5. Persistence

Die GUI Komponente greift dabei auf die Domain Logic Komponente zu, welche die nötigen Daten zur Anzeige in der GUI bereitstellt. Die Domain Logic Komponente wiederum greift auf die Network Komponente und die Persistence Komponente zu. Über diese beiden Komponenten erfragt die Domain Logic Komponente die nötigen Daten, entweder von anderen Geräten im gleichen Ad-hoc Netzwerk oder durch die auf dem Gerät gespeicherten Daten. Alle diese Komponenten greifen auf die Date Komponente zu, da in diesem die Modelle der einzelnen Datenstrukturen liegen.

2.1 GUI

Die bereits erwähnten Use Cases, sowie die Wireframes bilden bereits ein gutes Konzept für die GUI. Es wird nicht viele Activities geben, was genau auf der GUI angezeigt wird hängt hauptsächlich vom aktuellen Status der Session ab und kann somit durch verschiedene Fragments realisiert werden. Die GUI bietet keine Schnittstelle, da sie nur auf andere Komponenten zugreift. Das Testen dieser ist nur durch End-to-End Test möglich z.B. durch Bibliotheken wie Espresso.

2.1.1 Main Activity

2.1.1.1 Sessions

Ist der Boulderer noch keiner Session beigetreten, kann er dies im Hauptbildschirm tun oder eine neue erstellen. Ist die Session erstellt werden alle beigetretenen Boulderern mit Namen und Erfahrungslevel angezeigt. Hier kann nun ausgewählt werden ob die Boulderer ihre Statistiken vergleichen wollen oder eine Kilter Runde starten wollen.

2.1.1.2 Rivalitäten

Jeder Boulderer published seine eigenen Statistiken im bestehenden Ad-hoc Netzwerk, so können Statistiken angezeigt und untereinander verglichen werden.

2.1.1.3 RoutePool

Sind alle Boulderer bereit beginnen die Boulderer mit dem Befüllen des Routen Pools. Hierzu können sie entweder bereits erstellte Route auswählen, oder komplett neue Routen erstellen.

2.1.1.4 Voting

Ist dieser befüllt, können die Boulderer für eine Route stimmen. Haben alle Boulderer ihre Stimme abgegeben, wird eine Route ausgewählt und jeder Boulderer kann mit seinen Versuchen beginnen.

2.1.1.5 Attempts

Der Boulderer mit dem aktuellen Versuch bekommt die Möglichkeit seinen Versuch abzuschliessen indem er seinen Fortschritt in Prozenten angibt. Das geht so lange bis alle Boulderer die Route geschafft haben oder ihre Versuche verbraucht sind dann können die Boulderer wieder für eine Route abstimmen.

2.1.2 Route Creator

In dieser Activity wird den Boulderern die Möglichkeit gegeben eine eigene Route zu erstellen, abzuändern und zu speichern.

2.2 Data

Hier liegen wie bereits erwähnt die verschiedenen Models der einzelnen Datenstrukturen. Nach aktuellem Stand sind folgende Strukturen von Relevanz:

1. User
2. Route
3. Vote
4. Statistics

2.2.1 User

Der User besteht hauptsächlich aus:

- Name
- durchschnittliches Erfahrungslevel
- Statistics

2.2.2 Route

Eine Route ist charakterisiert durch:

- Schwierigkeitsgrad
- Darstellung der Route (wahrscheinlich als Bild, oder in Form eines Arrays)

2.2.3 Vote

Ein Vote hat folgende Bestandteile:

- gewählte Route
- Gewichtung des Votes

2.2.4 Statistics

Nach aktuellem Stand wollen wir vor allem die folgenden Daten in den Statistics unterbringen:

- höchstes erreichtes Erfahrungslevel
- Versuche pro Route inklusive prozentualem Fortschritt auf dieser Route

2.3 Domain Logic

Diese Komponente kann völlig unabhängig von Android implementiert werden und eignet sich damit für Unit Tests. Abhängigkeiten von der Network und Persistence Komponente können gemockt werden. Aufgabe dieser Komponente ist die Steuerung zwischen Nutzerinteraktionen und dem Bereitstellen der benötigten Daten für die verschiedenen Activities/Fragments. Damit werden folgende Aktionen bereitgestellt:

- Session erstellen
- Boulderer zu Session hinzufügen
- Boulderer von Session entfernen
- Routen zum Pool hinzufügen
- Routen Pool abfragen
- Abstimmen für Routen in Pool
- Meist gewählte Route abfragen

2.4 Network

Mit dieser Komponente werden die einzelnen Verbindungen zwischen den Geräten der Boulderer hergestellt. Außerdem werden die Anfragen von bestimmten Daten, sowie deren Antworten zwischen den Geräten ausgetauscht. Die Network Komponente bietet folgende Funktionen:

- Session erstellen
- Session beitreten
- Session verlassen
- Session Information aktualisieren

2.5 Persistence

Diese Komponente übernimmt das Speichern der für den Boulderer relevanten Daten. Dazu gehören:

- Boulderer Name
- Boulderer Erfahrungslevel
- Statistiken des Boulderers pro Route

3 Interfaces und Testing

3.1 Network

```
interface Network {
    fun getNearbySessions(): List<Session>
    fun joinSession(sessionId: String)
    fun leaveSession()
    fun createSession(): Session
    fun updateSessionInfo(sessionDetails: SessionDetails)
}
```

Unit Tests

- `getNearbySessions()` gibt Sessions zurück (mit mocked nearby sessions)
- `getNearbySessions()` gibt keine Sessions zurück (ohne nearby sessions)
- `joinSession()` tritt gemockter Session bei
- `leaveSession()` verlässt gemockte Session
- `createSession()` erstellt neue Session
- `updateSessionInfo()` setzt Session Details (testen via Spy)

3.2 Pool

```
interface Pool {
    fun createPool(): Pool
    fun getPoolDetails(): PoolDetails
}
```

Unit Tests

- `createPool()` erstellt eine neue Instanz von `Pool`
- `createPool()` gibt gültige `Pool`-Daten zurück (z. B. mit gemockter Implementierung prüfen)
- `createPool()` erstellt keine Duplikate, wenn ein Pool bereits existiert (ggf. mit Singleton-Mechanismus testen)

3.3 Statistics

```
interface Statistics {
    fun setNewHighestGrade(highestGrade: Grade)
    fun setCurrentRouteProgress(progress: Int)
    fun addFinishedRoute(finished: Route)
    fun addRivalryWin(rival: Rival)
}
```

Unit Tests

- `setNewHighestGrade()` setzt neues Höchstgrad (mit Mock-Datenbank prüfen)
- `setCurrentRouteProgress()` aktualisiert Fortschritt (mit Mock überprüfen)
- `addFinishedRoute()` fügt abgeschlossene Route in Datenbank ein
- `addRivalryWin()` fügt Rivalitätsgewinn in Datenbank ein

3.4 Route

```
interface Route {
    fun getRouteByName(name: String): Route?
    fun getRoutesByGrade(grade: Grade): List<Route>
}
```

Unit Tests

- `getRouteByName()` gibt existierende (mock) Route zurück
- `getRouteByName()` gibt `null` zurück, wenn Route nicht existiert
- `getRoutesByGrade()` gibt Liste der Routen für Grad zurück
- `getRoutesByGrade()` gibt leere Liste zurück, wenn keine Routen existieren

3.5 RouteManager

```
interface RouteManager {
    fun addVote(vote: Vote)
    fun getWinningRoute(): Route
}
```

Unit Tests

- `addVote()` fügt Stimme mit hinzu (Stimmengewicht ebenfalls testen)
- `getWinningRoute()` gibt Route mit den meisten Stimmen (unter Berücksichtigung der Gewichtung) zurück
- `getWinningRoute()` gibt `null`, wenn keine Stimmen vorhanden sind

3.6 Vote

```
interface Vote {
    fun getRoute(): Route
    fun getVoteWeight(): Int
}
```

Unit Tests

- `getRoute()` gibt die Route zurück, die abgestimmt wurde
- `getVoteWeight()` gibt korrektes Stimmgewicht zurück

3.7 VoteManager

```
interface VoteManager {
    fun addRoute(route: Route)
    fun removeRoute(route: Route)
    fun getRoutes(): List<Route>
}
```

Unit Tests

- `addRoute()` fügt neue Route hinzu
- `getRoutes()` gibt Liste aller hinzugefügten Routen zurück
- `getRoutes()` gibt leere Liste zurück, wenn keine Routen hinzugefügt wurden

3.8 Integration Tests

getestete Komponenten	Test Beschreibung	erwartetes Resultat
`Network` + `Pool`	Abhängigkeit zwischen Network und Pool	eine neue Session erstellt auch einen dazugehörigen Pool
`RouteManager` + `VoteManager`	einfache Interkation zwischen Route- und VoteManager	eine einzige Wahl auf eine Route führt dazu, dass diese Route die Wahl gewinnt
`Pool` + `Statistics`	Highscores und andere Statistiken werden beim Beenden einer Runde gesetzt	gegebene Statistiken sind am ende der Runde in der Datenbank