**Proof of Concept für Empfehlungen**

Personalisierte Empfehlungen sollen anhand von Nutzerdaten in der App angezeigt werden. Die Nutzerdaten beinhalten den Standort, präferierte Brettspielgenres und -arten, Alter der Person, Erfahrungsgrad, und zuvor aufgerufene Inhalte. Für die Kategorisierung der Inhalte werden diese mit Metadaten versehen. Die Metadaten der Brettspiele beinhalten Brettspielgenres, Altersempfehlung, Komplexität, Spielerzahl, und Spieldauer. Für die Gruppensuche werden die Metadaten Standort, Brettspielgenres und -arten, Alter für die Veranstaltung und Erfahrungsgrad benötigt.

Für das PoC werden Beispieldaten der Brettspiele, Nutzer und Gruppensuche erstellt. Für den Test des Algorithmus sollen die Methoden Collaborative Filtering und Content-Based Filtering zum Einsatz kommen. Collaborative Filtering wird benutzt, um Inhalte von Nutzern mit ähnlichen Präferenzen zu empfehlen. Content-Based Filtering wird hingegen benutzt, um ähnliche Inhalte zu empfehlen.

Es soll für den Prototyp eine einfache Benutzeroberfläche im Android Studio und ein Backend erstellt werden. Die Datenbank wird mithilfe von MongoDB erstellt.

In der Datenbank sollen die Nutzerdaten und Brettspiele hinterlegt werden.

Nutzer sollen auf der Benutzeroberfläche Gruppensuche und Brettspiele aufrufen und ihre Präferenzen bearbeiten können.

Exit-Kriterien:

* Inhalte, die ähnlichen Nutzern gefielen werden empfohlen und auf der Benutzeroberfläche angezeigt
* ähnliche Brettspiele werden empfohlen und auf der Benutzeroberfläche angezeigt, wenn ein Brettspiel aufgerufen wird

Fail-Kriterien:

* Inhalte, die nicht zu den Nutzer passen werden empfohlen
* ähnliche Inhalte, die nicht zu den Inhalt passen werden empfohlen
* keine Inhalte werden empfohlen

Fallbacks:

* der Nutzer soll nicht-relevante Inhalte als solches markieren können, um den Algorithmus zu verbessern

**Proof of Concept für generellen Datenbankzugriff von BGG**

Für die Funktion des Systems ist eine Brettspieldatenbank notwendig um unsere eigene Datenlage zu speisen. Hierzu wird Boardgamegeek.com (BGG) verwendet, der Zugriff ist kostenlos unbegrenzt möglich. Dieser Zugriff soll anhand des PoCs getestet werden. Hierzu wird ein http GET request für ein konkretes Spiel an BGG gesendet. Die empfangenen Daten sowie http Statuscodes werden dann über die Konsole ausgegeben. Zum Test wird das Spiel Twilight Struggle verwendet. (id=12333)

Exit-Kriterien:

* Request wird erfolgreich versendet
* Daten zum gesuchten Spiel werden erfolgreich empfangen und ausgegeben

Fail-Kriterien:

* Request wird nicht erfolgreich versendet.
* Es werden keine Daten empfangen.
* Es werden keine Daten ausgegeben.

Fallback:

* Es Wird eine andere API verwendet.

**Proof of Concept für Filterung der relevanten Daten**

**Vorraussetzung:** BGG Datenbank wird verwendet und ist verfügbar.

Nicht alle Daten aus der BGG Datenbank sind für uns relevant. Die Datenpunkte die wir benötigen sind: Name, Spielerzahl, Komplexität, Genre(s), Altersempfehlung und Spieldauer.

Leider sind nicht alle Spiele auf BGG in ein Genre eingeteilt, da dies über Community Voting geschieht und ein Spiel eine Mindestzahl an Votes braucht, um eingeteilt zu werden. Dies ist nicht immer der Fall, in diesen Fällen wird das Spiel von uns nicht mit aufgenommen.

Um die gewünschten Informationen über ein Spiel zu erhalten ist der Request mit entsprechenden Parametern anzupassen. Aus den Daten werden die benötigten Datenpunkte entnommen und diese mitsamt den http Statuscodes über die Konsole ausgegeben. Zum Test wird wieder „Twilight Struggle“ sowie „The Cones of Dunshire“ (id=165694) verwendet.

Exit-Kriterien:

* Beide Requests mit Parametern werden erfolgreich versendet.
* Daten werden erfolgreich empfangen.
* Gefilterte Daten werden für Twilight Struggle ausgegeben.
* Daten für The Cones of Dunshire werden nicht ausgegeben, da Genre fehlt.

Fail-Kriterien:

* Requests werden nicht erfolgreich versendet.
* Daten werden nicht empfangen.
* Twilight Struggle Daten werden gar nicht oder nicht richtig gefiltert ausgegeben.
* The Cones of Dunshire Daten werden ausgegeben.

Fallbacks:

* *Manuelle Filterung? Mir fällt da kein vernünftiger Fallback zu ein, wir müssen mit den Daten umgehen können – Frederik*

**Proof of Concept für Speicherung in eigene DB**

Für das System ist ein eigenes Datenbanksystem trotzdem notwendig. Diese soll mit den Daten aus BGG gespeist und aktuell gehalten werden. Damit dient sie als Fallback-Option, falls BGG nicht erreichbar ist. Außerdem können so Entwickler ihre Prototypen mit in die Datenbank einfügen und sind somit für den Matching-Algorithmus verfügbar. Für den PoC wird eine Datenbank eingerichtet. Anhand einer Eingabe von mindestens Name, Genre und Komplexität wird ein Spielobjekt erzeugt, welches dann auf der Konsole auszugeben ist, sowie auf der Datenbank zu speichern ist.

Exit-Kriterien:

* Spielobjekt wird erfolgreich erzeugt.
* Spiel wurde erfolgreich der Datenbank hinzugefügt.

Fail-Kriterien:

* Spielobjekt wird nicht erfolgreich erzeugt.
* Spiel konnte nicht der Datenbank hinzugefügt werden.

Fallbacks:

* Alternative persistente Datenspeicherung als ein Datenbanksystem nutzen. Spielobjekte könnten einzeln als JSONs exportiert werden.