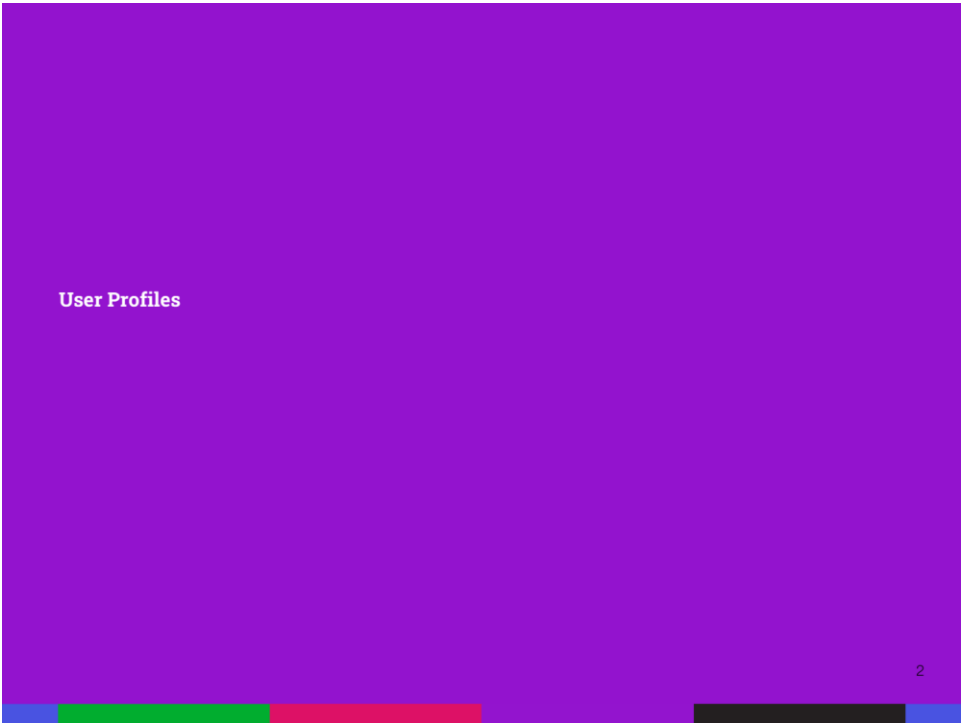


Entwicklungsprojekt 2023/24 - Audit 2

Eine Spielersuche für ein Brettspieltreffen

Frederik Hausen, Chenghua He und Esra Sancak






User Profiles

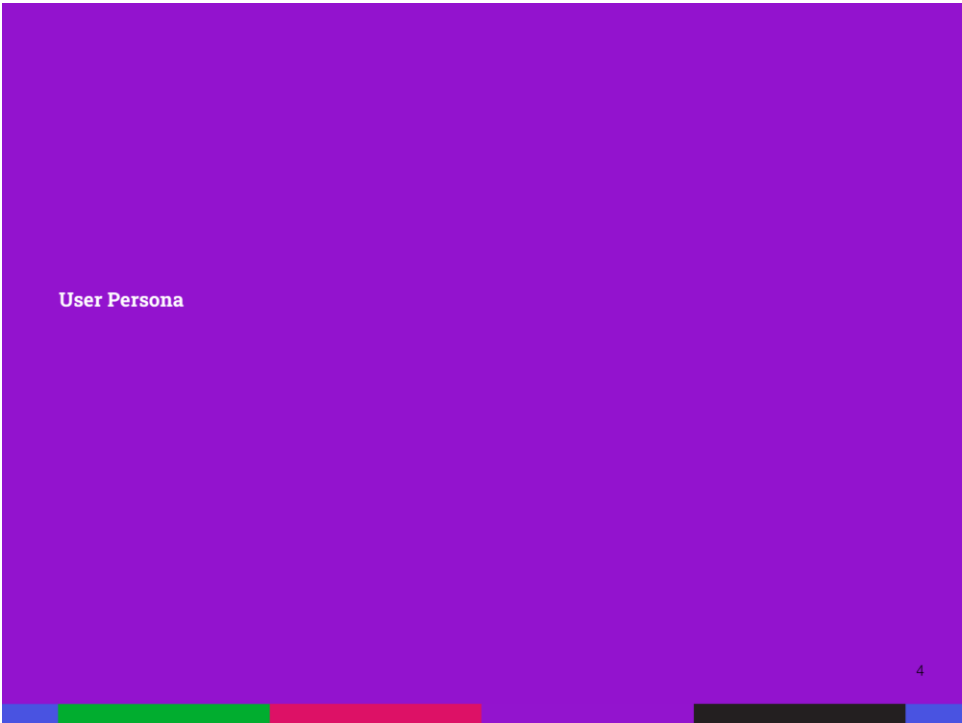
Kategorie Spielersuche: Brettspiele

Spieltyp	Lieblingsspiel	Spielstil	Erfahrung
1. Der Strategiemeister	Schach, Risiko, Siedler von Catan	Analytisch, strategisch denkend	Erfahren, sucht nach Herausforderungen
2. Der Gesellige Spieler	Codenames, Dixit, Tabu	Kommunikativ, teamorientiert	Mittel bis fortgeschritten, genießt soziale Interaktion
3. Der Themenspieler	Arkham Horror, Game of Thrones, Pandemic	Vertieft sich in die Spielwelten, liebt Geschichten	Fortgeschritten, mag komplexe Spiele mit tiefer Handlung
4. Der Familienspieler	Uno, Monopoly, Memory	Familiär, kinderfreundlich	Gelegenheitsspieler, sucht nach einfachen und lustigen Spielen
5. Der Taktiker	Carcassonne, Ticket to Ride, Stratego	Taktisch, liebt Knobelaufgaben	Mittel, mag Spiele, die schnelles Denken erfordern
6. Der Entdecker	Terraforming Mars, Gloomhaven, Scythe	Entdeckungsfreudig, offen für neue Spielmechaniken	Fortgeschritten bis Experte, liebt komplexe Spiele
7. Der Wettbewerbsfreak	Poker, Werwolf, Wettbewerbsspiele	Konkurrenzorientiert, gerne unter Druck	Fortgeschritten, sucht nach Wettbewerbsatmosphäre
8. Der Kooperative Spieler	Pandemic, Forbidden Island, Robinson Crusoe	Teamorientiert, mag gemeinsame Herausforderungen	Mittel bis fortgeschritten, bevorzugt kooperative Spiele

3



Im Benutzerprofil wurden Informationen zur Spielersuche für das Brettspiel in vier separaten Abschnitten aufgegliedert. Diese umfassen den Spielertyp, Beispiele für das Lieblingsspiel, den Spielstil und die Erfahrung. Dies diente als Grundlage für die Erstellung von Personas.



User Persona

Name:	Alter:	Beruf:	Wohnort:
Anna	36	Lehrerin	München

Spiertyp/Nickname
Die Strategiemeisterin

Hobbys:
Schach, Lesen, Musikinstrument spielen

Spielerfahrung:
Spielt Schach seit der Kindheit, hat an lokalen Turnieren teilgenommen, Mitglied in einem Schachverein

Spielvorlieben:
Liebt Spiele mit strategischen Anspruch und Aufbau, schätzt anspruchsvolle Schachpartien

Spielgruppen:
Liebt klassische Schachpartien, interessiert sich für Schachstrategien und Eröffnungen, spielt auch online gegen andere Spieler

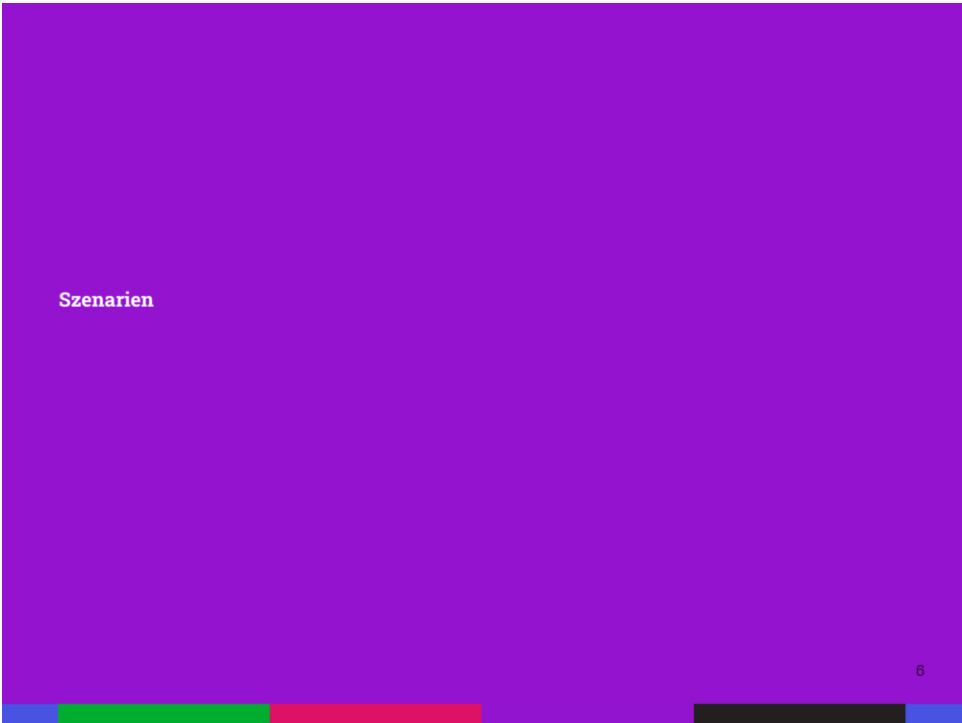
Präferierte Umgebung:
Bevorzugt eine ruhige und fokussierte Atmosphäre und konzentriert sich gerne auf das Spiel.

Anliegen:
Sucht nach herausfordernden Schachpartien, schätzt Lernmaterialien für Schachstrategien, wünscht sich eine lebendige Online-Schachcommunity

Ziel:
Strebt an, ihre Schachfähigkeiten zu verbessern, plant die Teilnahme an regionalen Schachmeisterschaften

5

Die User Personas geben einen detaillierten Einblick in die Spielercommunity, indem sie Informationen zu den individuellen Spielern präsentieren. Durch die Bereitstellung von umfassenden Angaben zu den Spielern ermöglicht sie uns, sich ein klares Bild von den Präferenzen, dem Spielverhalten und der Erfahrung der einzelnen User zu machen. Diese Darstellung bietet außerdem eine mögliche Grundlage für die letztendlichen Profile der User im System. Sie wären darüber in der Lage potenzielle Spielpartner zu finden, die ihren Vorlieben und Spielstilen entsprechen. Weitere Personas sind hier zu finden: [Personas1](#) [Personas2](#)



Szenario für Spiele-Prototypen

Lisa und Nick lieben Brettspiele und haben an ihrem selbst entwickelten Brettspiel gearbeitet. Nachdem sie den Prototypen fertiggestellt hatten, standen sie vor einem Problem: Sie brauchten dringend Tester, um das Spiel zu überprüfen und Feedback zu erhalten, bevor sie es weiter verbessern konnten. Sie haben den Prototypen zunächst in ihren Freundeskreisen vorgestellt und getestet, aber Lisa und Nick wünschten sich eine breitere Masse an Testern, die Erfahrungen mit Brettspielen haben und konstruktive Kritik geben können.


Nachdem sie verschiedene Methoden ausprobiert hatten, um Tester zu finden, stießen sie auf eine App, um Mitspieler für Brettspiele zu finden.

Lisa und Nick erstellten ein Profil auf der App, formulierten eine Spielersuche und markierten es als Test eines Prototyps. Sie beschrieben das Spiel, um anderen Spielern eine Vorstellung davon zu geben, worum es ging. Die App ermöglicht es ihnen auch, bestimmte Kriterien für Tester festzulegen, wie Erfahrung mit komplexen Spielen oder Interesse an bestimmten Spielgenres.

Daraufhin bekamen die beiden viele Nachrichten von Interessierten, die Fragen zu den Prototypen stellten und freiwillige Personen, die das Spiel testen wollten. Die App ermöglichte es ihnen, gezielt nach Spielern zu suchen, die ihren Kriterien entsprechen und bereit waren, konstruktives Feedback zu geben.

Lisa und Nick organisierten Testrunden über die App, luden die ausgewählten Tester ein und führten Spieletests durch. Durch die App können sie auch die Meinungen der Spieler protokollieren, Fragen stellen und Diskussionen über verschiedene Aspekte ihres Spiels führen.

Mit Hilfe der App erhalten sie wertvolles Feedback von einer breiten Palette an Spielern mit unterschiedlichen Perspektiven und Spielstilen. Sie konnten das Spiel verbessern und anpassen, basierend auf den Rückmeldungen, die sie durch die App erhalten hatten.



Wir haben für die Veranschaulichung der Use Cases unterschiedliche Szenarien erstellt, um die Funktionen der App zu erleuchten. Es wurden Nutzer mit verschiedenen Problemen erschaffen, welche durch die App oder Teilfunktionen der App gelöst werden sollen. Im Vordergrund stand aber vor allem das Finden von anderen Mitspielern mit ähnlichen Interessen. ([alle Szenarien](#))

Projektrisiken

Projektrisiken

Qualität der Vorschläge:

Irrelevante Vorschläge: Wenn das System nicht in der Lage ist, Vorschläge von Spieltreffen, die die Benutzer interessieren, bereitzustellen, kann dies die Benutzerzufriedenheit beeinträchtigen und dazu führen, dass die App weniger genutzt wird. Hinzu kommt, dass für die Entwicklung eines solchen Algorithmus keine Vorerfahrung vorliegt.

Abhängigkeit von Drittanbietern:


Für die Funktion des Systems ist der Aufbau einer Spieledatenbank notwendig. Hierzu käme in Frage die Datenbank eines Drittanbieters zu nutzen. Hierdurch entsteht das Risiko, dass die gewünschte Funktion nicht verfügbar ist, z.B. wegen Nutzungskosten der API.

Die zu verwendende Datenbank ist Boardgamegeek.com. (BGG)

Fehlerhafte Implementierung eigener Datenbank

Für die Funktion des Systems ist trotz BGG eine eigene Datenspeicherung nötig, z.B. für das Hinzufügen von Prototypen von Spieleentwicklern. Hierzu soll eine eigene Datenbank genutzt werden.

9



Die hier beschriebenen Risiken stellen die wichtigsten technischen Herausforderungen dar. Hervorzuheben ist dabei, dass das Matching eine deutlich höhere Priorität hat als die Risiken bezüglich der Datenlage. Hier gäbe es Alternativen und Fallback Optionen, die für den Algorithmus nicht vorhanden sind. Es stellt den Kern unserer Anwendungslogik dar.



Proof of Concept für Empfehlungen

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

Für den 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. Die Ähnlichkeit wird ermittelt, indem die Metadaten verglichen werden. Für den PoC werden Objekte als ähnlich befunden, die mindestens 3 übereinstimmende Metadaten besitzen, davon mindestens eine Brettspielgenre. Darüber hinaus werden die Objekte nach den meisten übereinstimmenden Metadaten sortiert.

Die Sortierung der Empfehlungen wird bei Aufruf variiert, um nicht immer die gleichen Inhalte anzuzeigen.

Es soll für den Prototyp eine einfache Benutzeroberfläche im Android Studio erstellt werden, welche die Empfehlungen vereinfacht darstellen soll.

Exit-Kriterien:


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

Fail-Kriterien:

- es werden zum Großteil Inhalte empfohlen, die nicht zu ähnlichen Nutzer passen
- es werden zum Großteil Inhalte empfohlen, die nicht ähnlich zu den aufgerufenen Inhalten passen
- keine Inhalte werden empfohlen
- es werden keine ähnlichen Inhalte für die Empfehlung gefunden
- es werden nur die gleichen Inhalte empfohlen

Fallbacks:

- Inhalte mit den meisten Aufrufen werden empfohlen, falls sonst keine ähnlichen Inhalte gefunden werden können
- Einzelne Inhalte mit vielen Aufrufen werden unabhängig der Ähnlichkeit empfohlen, um eine Varianz zu erzeugen
- Nutzer müssen mindestens 2 präferierte Brettspielgenres angeben, um mehr mögliche Empfehlungen bekommen zu können



Der Algorithmus für Empfehlungen stellt eine essenzielle Funktion unserer App dar, weshalb wir dessen Proof of Concept als unbedingt notwendig empfinden. Für den PoC haben wir eine grobe Recherche bezüglich Algorithmen für Empfehlungen durchgeführt ([hier zu finden](#)). Im Rahmen des PoCs haben uns für den Einsatz von Collaborative Filtering und Content-Based Filtering entschieden, weil es einfacher in der Durchführung ist als KI gestützte Methoden und für unsere Zwecke vollkommen ausreicht.

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.
- Externe Mittel nutzen um Request zu senden und manuelle Eingabe der Daten

Ein wichtiger Punkt in der Auswahl der Datenbank war, dass wir die Kategorisierung, die die Datenbank verwendet, übernehmen werden, da wir ansonsten jedes Spiel selber manuell überprüfen müssten, um es nach unseren Vorstellungen einzuteilen.

In der Recherche zu möglichen Datenbanken hat sich herausgestellt, dass es kaum eine Brettspieldatenbank gibt die auch eine Schnittstelle anbietet. Abgesehen von Boardgamegeek und APIs, die selber Boardgamegeek nutzen, konnten wir keine Alternative ausfindig machen. Daraus ergab sich das Problem, dass uns zunächst nicht ersichtlich war ob BGG überhaupt die Spiele in Genres einteilt. Die einzige Einteilung, die wir sehen konnten war nach Thematik und einzelnen Mechaniken der Spiele. Am Ende sind wir doch noch darauf gestoßen, wie BGG die Genres einteilt (Community

Voting) und wo in den API-Daten wir das finden können.

Damit bleiben dann am Ende eine Reihe von Vorteilen.

Boardgamegeek ist bei weitem die vollständigste
Brettspieldatenbank, der Zugriff auf die Daten ist kostenlos möglich
und wir sind in der Lage alle für uns relevanten Daten zu
übernehmen.

Proof of Concept für Filterung der relevanten Daten

Voraussetzung: 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 Stimmen 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 und Eingabe

13

Wie zuvor erwähnt, geschieht die Einteilung von Spielen in Genres auf Boardgamegeek nach einem Community Voting System.

Es stehen 6 Genres zur Verfügung: Abstract Games, Customizable Games, Thematic Games, Family Games, Party Games, Strategy Games und Wargames. Eine Erklärung dazu ist über <https://boardgamegeek.com/wiki/page/subdomain> zu finden. Es ist vorgesehen diese zu übersetzen und im System zur Erklärung mit anzubieten.

Ein Spiel benötigt mindestens 5 Stimmen und mindestens 30% aller Stimmen, um in ein Genre eingeteilt zu werden. Damit ist es möglich in bis zu 3 Genres zu gehören. Allerdings ist es durch dieses System bei einigen Spielen der Fall, dass nicht genügend Abstimmungen vorhanden sind, um es in irgendein Genre zuzuordnen. Wir haben die Entscheidung getroffen, in

diesen Fällen das Spiel zu ignorieren, da scheinbar das Interesse an dem Spiel eh verschwindend gering ist. Es ist vorgesehen den User über dieses System zu informieren, falls ein Spiel nicht gefunden werden kann.

Proof of Concept für Speicherung in eigene DB

Für das System ist ein die Nutzung eines eigenen Datenbanksystem trotzdem notwendig. Diese Datenbank 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.



Frühe Spieleprototypen sind nicht auf BGG erfasst und somit wären diese für unser nicht System verfügbar. Wir würden also Spieleentwickler darum bitten, nach dem BGG Schema ihre Prototypen einzuordnen. Diese Information muss dann von uns gespeichert werden, damit der Matchingalgorithmus in der Lage ist, die Prototypen vorzuschlagen.

Projektplan

Meilensteine für Audit 3

22.01.14 Modellierung von Datenstruktur/Klassenmodellierung

07.01.24 Algorithmus PoC durchgeführt

14.01.24 DB PoCs durchgeführt, Spezifizierte Use Cases

