

Entwicklung einer mobilen Applikation zur Suche und Verwaltung von Brettspielen

Software Engineering

vorgelegt am 8. Februar 2024

Fakultät Wirtschaft und Gesundheit

Studiengang Wirtschaftsinformatik

Kurs WWI2021F

von

SIMON BURBIEL

LUKAS GROSSERHODE

TIM KEICHER

SIMON SPITZER

Inhaltsverzeichnis

Abkürzungsverzeichnis	III
Abbildungsverzeichnis	IV
Tabellenverzeichnis	V
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	1
2 User Interface Konzept	2
2.1 Allgemeines und Funktionalität	2
2.2 Beschreibung der erstellten Mockups	3
2.2.1 Homepage	3
2.2.2 Searchpage	3
2.2.3 Infopage	3
2.2.4 Wunschliste	3
2.2.5 Inventar	3
3 Technische Umsetzung	4
3.1 Aufbau des Backends	4
3.2 Beschreibung der verwendeten Services	4
3.3 Beschreibung der verwendeten APIs	4
3.3.1 BoardGameGeek XML API	4
3.3.2 MongoDB API	5
3.3.3 Verwendung der API im Projekt selbst	6
3.4 Anleitung zum Starten der Applikation	6
4 Aufteilung des Projektes	7
5 Zusammenfassung und Ausblick	8
Anhang	9

Abkürzungsverzeichnis

API Application Programming Interface

Abbildungsverzeichnis

1	Erster Entwurf eines Konzeptes für die Benutzeroberfläche.	2
2	Ergebnis der ‘/xmlapi/search’-Funktion bei Suchterm Frika	4

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

1.2 Zielsetzung

Zur Unterstützung soll daher im Rahmen des Moduls „Software Engineering“, welche als Prüfungsleistung die Entwicklung einer mobilen Applikation vorsieht, eine Ionic-App erstellt werden, welche die Suche und Verwaltung von Brettspielen ermöglicht. Im Kern soll die App über folgende Funktionalitäten verfügen:

- **Suche nach Spielen:** Eine Suchleiste ermöglicht die Suche nach einer Vielzahl von Brett-, Karten- und Würfelspielen.
- **Anzeige detaillierter Informationen zu Spielen:** Hierzu gehören unter anderem Name, Hersteller, Erscheinungsjahr, Anzahl der Spieler sowie eine Beschreibung des Spiels. Diese sollen über ein „Application Programming Interface (API)“ eingespielt werden.
- **Verwaltung von Spielen in einer Wunschliste:** In der Wunschliste können Spiele hinterlegt werden, welche der Nutzer sich zu einem späteren Zeitpunkt kaufen möchte.
- **Verwaltung von Spielen im eigenen Inventar:** Das Inventar bietet eine Übersicht über die Spiele, welche der Nutzer bereits besitzt.
- **Vergabe von Bewertungen zu Spielen:** Dem Nutzer soll es auch ermöglicht werden, eine eigene Bewertung für jedes Spiel abgeben zu können. Diese Funktion soll anhand eines 5-Sterne-Systems realisiert werden.

Der Nutzer soll sich zwischen den Seiten der App frei bewegen können. Die App soll dabei eine übersichtliche und intuitive Benutzeroberfläche bieten, welche die Funktionalitäten der App klar darstellt und eine einfache Bedienung ermöglicht.

2 User Interface Konzept

2.1 Allgemeines und Funktionalität

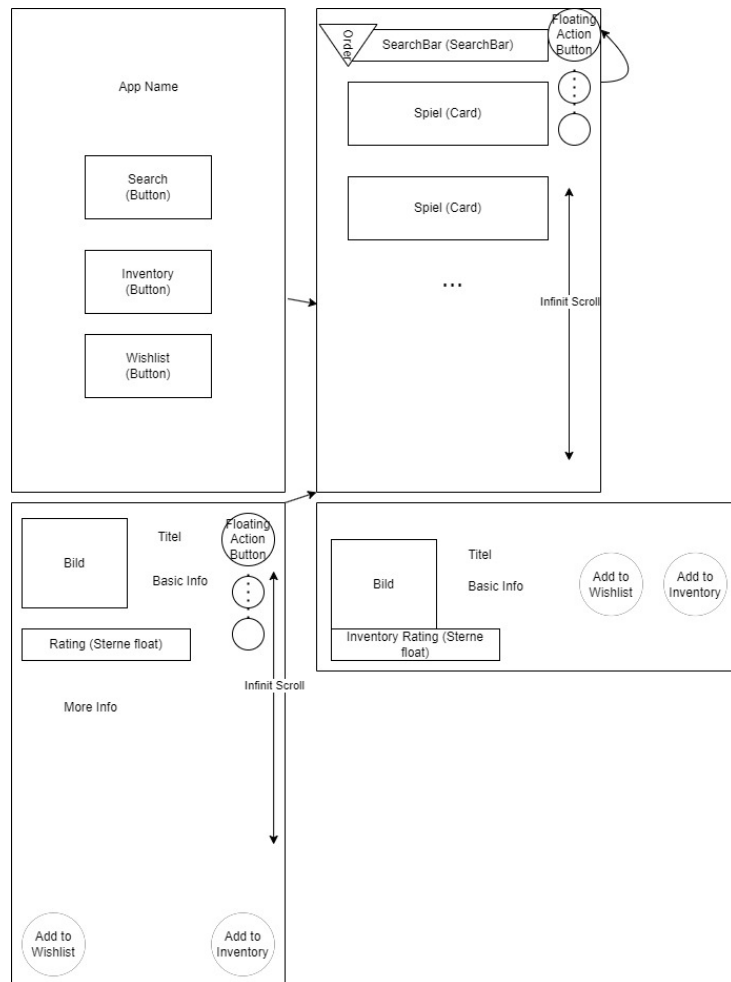


Abb. 1: Erster Entwurf eines Konzeptes für die Benutzeroberfläche.

2.2 Beschreibung der erstellten Mockups

2.2.1 Homepage

2.2.2 Searchpage

2.2.3 Infopage

2.2.4 Wunschliste

2.2.5 Inventar

3 Technische Umsetzung

3.1 Aufbau des Backends

3.2 Beschreibung der verwendeten Services

3.3 Beschreibung der verwendeten APIs

3.3.1 BoardGameGeek XML API

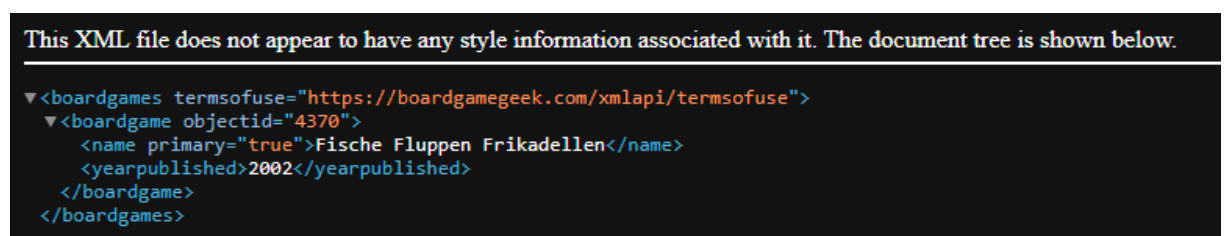
Die BoardGameGeek XML API bietet eine Schnittstelle zum Zugriff auf eine Vielzahl von Informationen rund um Brettspiele, die auf BoardGameGeek.com, einer umfangreichen Datenbank und Community für Brettspiel-Enthusiasten, verfügbar sind. Diese API ermöglicht es Entwicklern durch verschiedene Endpoints auf Spielinformationen, Benutzerkollektionen und Forendiskussionen zuzugreifen. Dabei zu beachten ist, dass die API die Antwort als XML-Format weitergibt. Da häufig mit JSON gearbeitet wird ist eine mögliche Umformung der Daten sinnvoll.

Suchfunktion

Eine der beiden Endpunkte ist die ‘/xmlapi/search’-Funktion bei der Nutzer als Input einem spezifischen Suchterm eingeben. Als Ergebnis enthält man eine Liste an Brettspielen in XML Format, deren Namen oder Alias im Suchbegriff enthalten war. Die detaillierte Antwort enthält folgende Informationen:

- objectId des Brettspiels
- Name des Brettspiels
- Erscheinungsjahr des Brettspiels

Die reale Ausgabe für den fiktiven Suchterm “Frika” würde somit folgendes zurückgeben:



The image shows a screenshot of an XML API response. At the top, a message states: "This XML file does not appear to have any style information associated with it. The document tree is shown below." Below this, the XML structure is displayed with a tree view. The root element is <boardgames termsfuse="https://boardgamegeek.com/xmlapi/termsfuse">. It contains a single child element <boardgame objectid="4370">. This element has three sub-elements: <name primary="true">Fische Fluppen Frikadellen</name>, <yearpublished>2002</yearpublished>, and </boardgame>. The root element is closed with </boardgames>.

```
<?xml version="1.0" encoding="UTF-8" ?>
<boardgames termsfuse="https://boardgamegeek.com/xmlapi/termsfuse">
  <boardgame objectid="4370">
    <name primary="true">Fische Fluppen Frikadellen</name>
    <yearpublished>2002</yearpublished>
  </boardgame>
</boardgames>
```

Abb. 2: Ergebnis der ‘/xmlapi/search’-Funktion bei Suchterm Frika

Detaillierte Spielinformationen

Der andere Endpunkt ist die `/xmlapi/boardgame/<gameid>`-Funktion und dient dazu, detaillierte Informationen zu einem Brettspiel zu erhalten. Nutzer könnten hier noch einige weitere Parameter mitgeben um spezifische Informationen zu erhalten, jedoch ist dies in unserem Projekt nicht notwendig, da die Ergebnisse schon detailliert genug sind. Die `gameId` ist hierbei der Input, ist gleichzusetzen mit der `objectId` und kann aus des oben erhaltenen Ergebnisses der Suchfunktion entnommen werden.

- `objectId`: String
- `yearPublished`: String
- `minPlayers`: String
- `maxPlayers`: String
- `playingTime`: String
- `minPlayTime`: String
- `maxPlayTime`: String
- `age`: String
- `description`: String
- `name`: Array (String)
- `publisher`: Array (String)
- `averageWeight`: String
- `averageRating`: String
- `thumbnail`: String (Link to picture source)
- `usersRated`: String

Es ist zu erkennen, dass der Name und der Publisher Arrays sind. Dies lässt sich darauf zurückführen, dass es mehrere Namen für dasselbe Spiel gibt (teilweise auch in anderen Sprachen) und mehrere Entwickler beteiligt sein können.

3.3.2 MongoDB API

LUKAS TEIL

3.3.3 Verwendung der API im Projekt selbst

3.4 Anleitung zum Starten der Applikation

4 Aufteilung des Projektes

5 Zusammenfassung und Ausblick

Anhang

Anhangverzeichnis

Anhang 1	Anhang 1	10
Anhang 2	Anhang 2	10
Anhang 3	Anhang 3	10

Anhang 1: Anhang 1

Anhang 2: Anhang 2

Anhang 3: Anhang 3