Entwicklung einer mobilen Applikation zur Suche und Verwaltung von Brettspielen

Software Engineering

erstellt am 9. Februar 2024

Fakultät Wirtschaft und Gesundheit

Studiengang Wirtschaftsinformatik

Kurs WWI2021F

von

Simon Burbiel

Lukas Grosserhode

Tim Keicher

Simon Spitzer

Inhaltsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Die fortschreitende Digitalisierung hat in den letzten Jahren zu einem starken Wandel in der individuellen Freizeitgestaltung geführt. So konnten sich Streaming-Dienste und Videospiele als eine beliebte Form der Freizeitbeschäftigung etablieren. Dennoch erfreuen sich auch Brett-, Karten- und Würfelspiele nach wie vor einer großen Beliebtheit. Gründe hierfür liegen vor allem darin, dass analoge Spiele soziales Miteinander fördern und eine willkommene Abwechslung zu digitalen Medien darstellen. Dabei kann es jedoch schwierig sein, ein passendes Spiel zu finden, welches den eigenen Vorlieben entspricht. Ebenso kann es bei zunehmendem Bestand an Spielen kompliziert werden, den Überblick zu behalten, welche Spiele derzeit in Besitz sind und welche für einen künftige Anschaffung in Frage kommen.

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 annhand 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.

1.3 Arbeitspakete

Um eine bessere Organisation des Entwicklungsprozesses zu gewährleisten, wird das Projekt in verschiedene Arbeitspakete unterteilt. Zu diesen zählen:

1. **UI-Design:** Hier soll das Design der App gestaltet werden. Hierunter fallen bspw. die Gestaltung der Farbgebung, Schriftarten und Icons.

2. **API-Integration:** In diesem Paket soll die Anbindung einer externen API realisiert werden, welche die Daten zu den Spielen bereitstellt. Die Formatierung der Daten soll dabei der Struktur der Datenbank entsprechen.

3. **MongoDB-Setup:** In diesem Paket soll die Datenbank für die App eingerichtet werden. Hierunter fallen primär die Erstellung der Datenbank, die Einrichtung der Collections und das Schreiben von Fuktionen, welche Abrufe (GET), Einfügungen (POST) und Löschungen (DELETE) von Daten ermöglichen.

4. **Suchfunktions-Implementierung:** Hierbei wird die Suchfunktion für die App konzipiert und implementiert. Die Suchfunktion soll dabei die durch die API bereitgestellten Informationen nach den eingegebenen Suchbegriffen durchsuchen und die passenden Ergebnisse zurückgeben. Selbiges soll auch für die Elemente in der eigenen Datenbank ermöglicht werden.

Um eine annähernd gleichmäßige Verteilung der Arbeitspakete zu gewährleisten, wird jedem Teammitglied ein Arbeitspaket zugewiesen (siehe Tab. 1). Für größere Arbeitspakete ist zudem jeweils ein weiteres Teammitglied zur Unterstützung vorgesehen. In allen anderen Fällen erfolgt die Beihilfe zu den Tätigkeiten der hauptverantwortlichen Teammitglieder abhängig von der aktuellen Auslastung des jeweiligen Unterstützers.

Arbeitspaket	**Teammitglied**	**Unterstützung**
UI-Design	Tim Keicher	auslastungsabhängig
API-Integration	Simon Spitzer	Simon Burbiel
MongoDB-Setup	Lukas Großerhode	Simon Spitzer
Suchfunktions-Implementierung	Simon Burbiel	auslastungsabhängig

Tab. 1: Zuweisung der Arbeitspakete.

2 User Interface Konzept

2.1 Allgemeines und Funktionalität

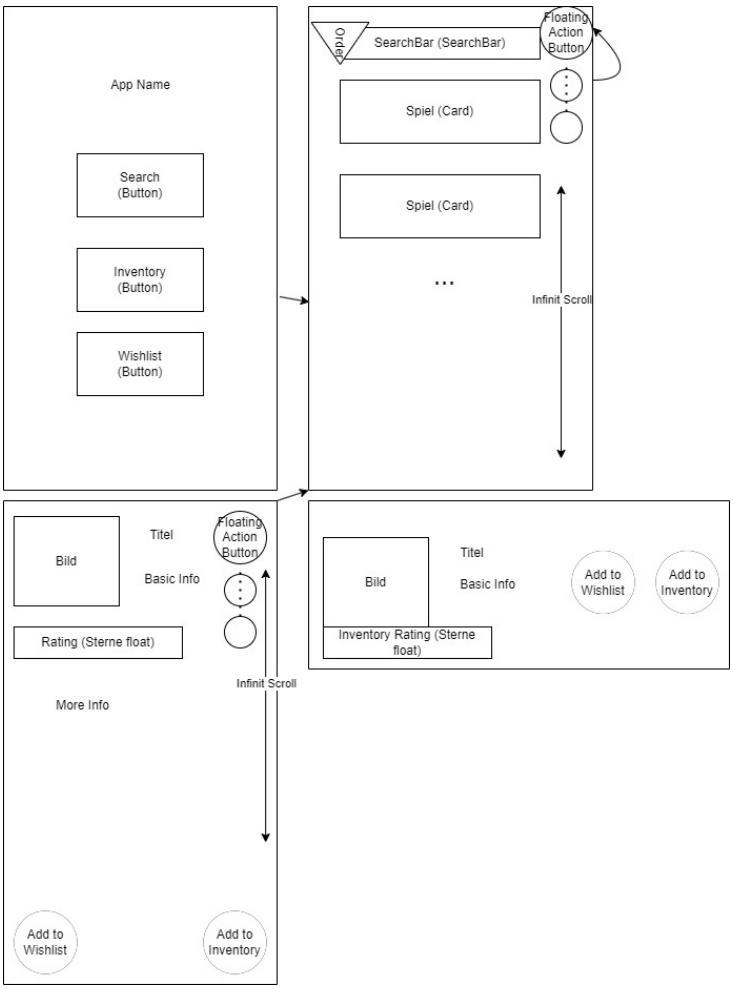

Abb. 1: Erster Entwurf eines Konzeptes für die Benutzeroberfläche.

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 Beschreibung der App

3.1.1 Funktionen der jeweiligen Seiten

Homepage

Die Homepage der App zeigt ein zufälliges und derzeit besonders beliebtes Spiel an, welches von der BoardGameGeek API bezogen wird. Die Generierung erfolgt automatisch und bei jedem Aufruf der Seite neu. Daher wird dieser Fähigkeit in der `ngOnInit()`-Funktion auf der Homepage aufgerufen. Darüber hinaus dient die Homepage primär als Ausgangspunkt für die Navigation zu den anderen Seiten (siehe Tab. 2).

Funktion	Beschreibung
`goToSearch()`	Navigiere zur Suchleiste.
`goToWishlist()`	Navigiere zur Wunschliste.
`goToInventory()`	Navigiere zum Inventar.
`openGameDetails()`	Navigiere zur Infopage des zufälligen Spiels.

Tab. 2: Funktionen der Homepage.

Searchpage

Infopage

3.1.2 Genutzte Interfaces und Modelle

In der Welt der Softwareentwicklung ist es entscheidend, dass Daten, die zwischen verschiedenen Systemen ausgetauscht werden, klar definierte Strukturen haben. TypeScript-Interfaces sind hierbei äußerst nützlich, insbesondere wenn es um die Interaktion mit APIs geht. Das Boardgame Interface beschreibt alle wesentlichen Informationen eines Brettspiels, die über die BGG-API abgerufen werden und in einer MongoDB gespeichert werden. Dazu gehören Details wie der eindeutige Identifikator des Spiels, das Veröffentlichungsjahr, die Spieleranzahl, die Spieldauer, das empfohlene Alter, eine Beschreibung und ein Vorschaubild. Dieses Interface stellt sicher, dass die von der API gelieferten Daten vollständig und in einer korrekten Form sind, die direkt in der Datenbank gespeichert werden kann (so dass auch Falls kein Bild in der Api Verfügbar ist dies abgespeichert wird). Das Game Interface hingegen ist eine reduzierte Version des Boardgame Interfaces. Es kann in Situationen eingesetzt werden, in denen nicht alle Informationen eines Brettspiels benötigt werden, wie zum Beispiel in einer Übersichtsliste von Spielen. Durch

das Weglassen von Details wie Spielzeit oder Altersangabe wird die übertragene Datenmenge verringert, was die Effizienz verbessern und die Ladezeiten verkürzen kann. Das Randomgame Interface ist spezialisiert auf die Darstellung eines zufällig ausgewählten Spiels aus den Top 50 der am besten bewerteten Brettspiele. Neben der Standardinformation wie Identifikator, Name, Veröffentlichungsjahr und einem Bild des Spiels, speichert dieses Interface zusätzlich den Rang des Spiels, also seine Position in der Rangliste. Diese Information ist besonders wertvoll, um auf einen Blick die Beliebtheit eines Spiels einschätzen zu können, diese wird basierend auf ihrer Position in der Top 50 eingelesen. Diese Interfaces sind essenzielle Bausteine für die zuverlässige Datenarchitektur. Sie sorgen nicht nur für eine korrekte Typisierung und eine klare Vertragsgestaltung zwischen dem Backend (der API) und dem Frontend, sondern tragen auch dazu bei, den Datenfluss übersichtlich und wartbar zu halten. So kann die Software sich an verändernde Anforderungen anpassen, ohne dass es zu einem Durcheinander in der Datenstruktur kommt.

Boardgame

Das `Boardgame` Interface repräsentiert die Struktur zur Speicherung von Informationen über ein Brettspiel. Die Eigenschaften sind wie folgt:

Eigenschaft	Typ	Beschreibung
`objectId`	`string`	Dient der eindeutigen Identifikation eines Spiels.
`yearPublished`	`string`	Das Jahr, in dem das Spiel veröffentlicht wurde.
`minPlayers`	`string`	Die minimal erforderliche Anzahl an Spielern.
`maxPlayers`	`string`	Die maximal mögliche Anzahl an Spielern.
`playingTime`	`string`	Die durchschnittliche Spieldauer.
`minPlayTime`	`string`	Die minimal mögliche Spieldauer.
`maxPlayTime`	`string`	Die maximal mögliche Spieldauer.
`age`	`string`	Die empfohlene Altersangabe.
`description`	`string`	Eine Beschreibung des Spiels.
`name`	`string[]`	Eine Liste von Namen, unter denen das Spiel bekannt ist.
`publisher`	`string[]`	Eine Liste von Verlagen, die das Spiel veröffentlicht haben.
`averageWeight`	`string`	Die durchschnittliche Komplexität des Spiels.
`averageRating`	`string`	Die durchschnittliche Bewertung des Spiels.
`thumbnail`	`string`	Ein Link zu einem Vorschaubild des Spiels.
`usersRated`	`string`	Die Anzahl der Benutzer, die das Spiel bewertet haben.

Tab. 3: Eigenschaften des `Boardgame` Interfaces.

Game

Das `Game` Interface repräsentiert eine reduzierte Version des `Boardgame` Interfaces. Es wird verwendet, um die Datenmenge zu verringern, wenn nicht alle Informationen eines Spiels benötigt

werden. Die Eigenschaften sind hierbei wie folgt:

Eigenschaft	Typ	Beschreibung
`objectId`	`string`	Dient der eindeutigen Identifikation eines Spiels.
`name`	`string`	Gibt den Namen des Spiels an.
`yearPublished`	`string`	Das Jahr, in dem das Spiel veröffentlicht wurde.
`thumbnail`	`string`	Ein Link zu einem Vorschaubild des Spiels.

Tab. 4: Eigenschaften des `Game` Interfaces.

Hinweis: Es gibt zwei Versionen des `Game` Interfaces, eine mit der Eigenschaft „thumbnail“ und eine ohne. Es ist wichtig sicherzustellen, dass die richtige Version verwendet wird, um der erforderlichen Datenstruktur für die Anwendung zu entsprechen.

Randomtopgame

Das `Randomtopgame` Interface repräsentiert die Struktur zur Speicherung von Informationen über ein zufällig ausgewähltes Spiel aus den Top 50 der am besten bewerteten Brettspiele. Hierfür werden folgende Eigenschaften definiert:

Eigenschaft	Typ	Beschreibung
`id`	`string`	Dient der eindeutigen Identifikation eines Spiels.
`name`	`string`	Der Name des Spiels.
`yearPublished`	`string`	Das Jahr, in dem das Spiel veröffentlicht wurde.
`thumbnail`	`string`	Ein Link zu einem Vorschaubild des Spiels.
`rank`	`string`	Die Position des Spiels in der Rangliste.

Tab. 5: Eigenschaften des `Randomtopgame` Interfaces.

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

druch 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 drei benutzten 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 Suchfunktion 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:

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

Detaillierte Spielinformationen

Der zweite 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. Die detaillierten Spielinformationen enthalten folgende Informationen:

Name	Formattyp
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 (URL to thumbnail image)
usersRated	String

Tab. 6: Die zurückgegebenen detaillierten Informationen

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 Entwicklet beteiligt sein können.

Game of the Day

Der dritte Endpunkt im Bunde ist die 'xmlapi2/hotoverall'-Funktion. Sie gibt die top 50 Spiele mit besonders guter Bewertung und Beliebtheit zurück. Dabei werden genau fünf unterschiedliche Informationen aufgeführt:

- id: String (objectId von vorher earlier)
- rank: String (Ranking Position der Top 50 Spiele)
- thumbnail: String (URL to thumbnail image)
- name: String
- yearpublished

Im Anschluss kann ebenfalls mit der ObjectId eine Anfrage für detaillierte Informationen gestartet werden. Einsatz findet dieses Verfahren wenn mit einem Klick das Game of the Day ausgewählt wird und somit die zurückgegebene ObjectId auf die Infopage weitergeleitet wird. Somit konnte eine weitere Seite wiederverwendet werden und zudem ein neues Feature für die App ermöglicht.

3.3.2 MongoDB API

In diesem Teil wird nur der externe Aufruf auf den selbsterstellten MongoDB API-Endpunkt behandelt. Der Interne Aufbau der MongoDB wird in der Beschreibung der App aufgeführt. Es gibt insgesamt drei Zugriffsmethoden für jede Collection auf die MongoDB: searchGamesWishlist(), searchGamesInventory(), addToWishlist(gameData), addToInventory(gameData), removeFromWishlist(objectId), removeFromInventory(objectId)

Die Get Methode, hier searchGames..., bezieht sich af den HTTP Endpoint "http://localhost:8999/wishlisht" oder beziehungsweise "... /inventory" und liefert eine Liste aus JSON-Elementen zurück. Für das hinzufügen von Brettspielen wird die HTTP-Client Methode POST benutzt und die gameData übergeben, wobei Gamedata ein Objekt des Modells Boardgame.ts ist. Dieses Objekt wird dann in eine JSON Form in die ausgewählte collection hinzugefügt. Bei der Entfernung eines Spiels wird als Parameter die ObjectId des Spiels übergeben. Dieser einzigartige Klassifikator sorgt dafür, dass wenn ein Spiel in der Collection enthalten ist, dass diese ObjectId als Klassifikator verwendet, wird dieses Brettspiel entfernt. Des Weiteren ist dies aus Effizienz gründen vorteilhaft, da kein komplettes Objekt übergeben wird, geparsed werden muss und schlussendlich verglichen. Es kann direkt eine Überprüfung statt finden.

```
1    this.http.delete(`${this.inventoryUrl}/${objectId}`)
```

Listing 3.1: Löschaufruf der MongoDB API

3.3.3 Verwendung der APIs im Projekt selbst

Die Definition der Methoden zum aufrufen der API-Endpunkte ist in dem File "boardgame.service.ts" des "services"-Ordner zu finden. Dazu wird das HTTPClient Modul importiert und es wird ein htttp.get Aufruf mit den jeweiligen Parametern durchgeführt. Da das Ergebnis in Form einer XML-Datei vorliegt und es vorteilhafter ist mit einem JSON-Format zu arbeiten wird zu Beginn eine Transfarmation des Formats in die JSON Form durchgeführt. Dies geschieht konkret durch das importierte Modul "xml2json".

Die daraufhin vorliegende JSON kann nun dem Modell game.ts, boardgame.ts oder randomtopgame.ts zugewiesen werden. Also eine initialisierung einer Liste von Objekte dieser Modelle basierend auf dem spezifischen API-Aufruf. Im folgenden wird dann Array als return Element der Funktion definiert. Dies ermöglicht es erstmalig ein subscribe auf die Funktion zu legen und auf der benötigten Page anzuzeigen.

Die konkrete Verwendung der APIs ist auf den folgenden Pages ist wiefolgt:

- Homepage
  - Game of the Day mit 'xmlapi2/hotoverall'-Funktion
- Searchpage

  - Suchfunktion mit '/xmlapi/search'-Funktion
- Inventory, Wishlist
  - MongoDB GET mit "http://localhost:8999/inventory/"-Funktion
- Infopage
  - Detaillierte Spielinformationen mit '/xmlapi/boardgame/<gameid>'-Funktion
  - MongoDB mit "http://localhost:8999/inventory?objectId"-DELETE Funktion
  - MongoDB mit "http://localhost:8999/inventory?gameData"-ADD Funktion
  - MongoDB mit "http://localhost:8999/wishlisht?objectId"-DELETE Funktion
  - MongoDB mit "http://localhost:8999/wishlisht?gameData"-ADD Funktion

3.4 Anleitung zum Starten der Applikation

4 Aufteilung des Projektes

5 Zusammenfassung und Ausblick

Anhang

Anhangverzeichnis

Anhang 1: Anhang 1

Anhang 2: Anhang 2

Anhang 3: Anhang 3