StreamSwipe
Tinder für Filme

Leon Gieringer, Robin Meckler,Vincent Schreck

Studienarbeit

3. April 2021

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 1
- 1.2 Aufbau der Arbeit 1

2 Motivation **1**

3 Theoretische Grundlagen **1**
- 3.1 Framework 1
- 3.2 Language 1
  - 3.2.1 JavaScript 1
- 3.3 IDE 1
- 3.4 MongoDB 1
- 3.5 Representational State Transfer - Application Programming Interface 1
- 3.6 Firebase 1
- 3.7 Recommendationssystem 1
  - 3.7.1 Collaborative Filtering 2
  - 3.7.2 Content-based filtering 3
  - 3.7.3 Cold Start Problem 3

4 Konzept? **3**

5 Auswahl geeigneter Technologie **3**
- 5.1 Server 3
- 5.2 Datenbank 3
- 5.3 Kommunikationsschnittstelle 3

6 Backend-Implementierung **3**
- 6.1 Server 3
  - 6.1.1 Einrichtung 3
  - 6.1.2 Sicherheit 3
  - 6.1.3 Webserver 3
- 6.2 Datenbank 3
  - 6.2.1 Einrichtung 4
- 6.3 Kommunikationsschnittstelle 4
  - 6.3.1 Implementierung 4

7 Funktionen/Komponenten **4**
- 7.1 Swipe/Aussuchen/Voting 4
- 7.2 Matches/Chat 4
- 7.3 Film-/Serienvorschläge 4
- 7.4 Gruppenorgien 4
- 7.5 Gespeicherte Filme/Filmliste 4
- 7.6 Zugänglichkeit/Behindertenfreundlichkeit 4

8 Benutzeroberflächen **4**
- 8.1 Home-Screen 4
- 8.2 Gruppen 4
- 8.3 Chat 4
- 8.4 Filmliste 4

9 CodeBeispiele **4**

10 Probleme **4**

11 Fazit **4**

1 Einleitung

1.1 ..

1.2 Aufbau der Arbeit

2 Motivation

3 Theoretische Grundlagen

3.1 Framework

Hier steht mein Framework Text

3.2 Language

Hier steht mein Language Text.

3.2.1 JavaScript

In den nächsten Unterkapiteln soll ein zunächst ein historischer Überblick über die Programmiersprache JavaScript gegeben werden. Im Anschluss wird auf die Bedeutung und Nutzung von JavaScript eingegangen.

3.2.2 Historie TODO

Ihren Ursprung findet JavaScript im Jahr 1995, als Brendan Eich, ein damaliger Ingenieur des US-amerikanischen Software-Unternehmens „Netscape Communications Corporation“, innerhalb von zehn Tagen die Sprache für den Browser „Netscape Navigator“ entwickelt hat. [1] Das Ziel dabei war es, eine Skriptsprache zu entwickeln, die es Entwicklern möglich machen sollte, auf ihren Webseiten Skripte umzusetzen. Zunächst noch unter dem Namen Mocha und LiveScript änderte sich der Name aufgrund der Kooperation von Netscape und Sun, der Firma hinter der Programmiersprache Java, und der damaligen Popularität von Java zu JavaScript. [1.05] Netscape's Veröffentlichung des Netscape Navigator 2.0, der erste Browser der JavaScript unterstütze, brachte Microsoft dazu, Netscape als ernstzunehmenden Konkurrenten zu sehen. Microsoft antwortete im August 1995 mit der Veröffentlichung des ersten Internet Explorer zusammen mit der Skriptsprache JScript, die einen Dialekt der Sprache JavaScript darstellt. Dies ist ferner als Beginn der „Browserkriege“ bekannt. [1.06] Im Jahre 1997 reichte Netscape JavaScript an die European Computer Manufacturers Association, einer privaten, internationalen Normungsorganisation zur Normung von Informations- und Kommunikationssystemen und Unterhaltungselektronik (kurz ECMA[ABK]) ein. Das Ziel war es, von der ECMA einen einheitlichen Standard für die Sprache schaffen zu lassen, die fortan weiterentwickelt und von weiteren Browserherstellern genutzt werden soll. Das resultierende Standard nennt sich ECMAScript, wobei JavaScript die bisher bekannteste Implementierung dieses Standards ist. [1.07] Andere Implementierungen sind zum Beispiel ActionScript von MacroMedia, JScript von Microsoft und ExtendScript von Adobe. Jährlich wird der Standard seit Juni 2015 erweitert. ECMAScript Version 11 beziehungsweise ECMAScript 2020 bildet zum Zeitraum dieser Dokumentation [??] den aktuellen Standard. [1.08] Im Juni 2021 soll ECMAScript 2021 veröffentlicht werden. [1.09]

3.2.3 Wesentliche Programmiereigenschaften TODO

„JavaScript is Not Java“ [1.091 ??]. Die Programmiersprache JavaScript wird aufgrund ihrer Namensgebung oft in falsche Zusammenhänge zu Java gebracht. Das häufigste Missverständnis

sei, JavaScript wäre eine vereinfachte Version von Java. [1.091] JavaScript ist eine interpretierte Programmiersprache mit objektorientierten Umsetzungs-möglichkeiten. Interpretation ist in diesem Zusammenhang so zu verstehen, dass der Quellcode zur Laufzeit eines Programms gelesen, übersetzt und ausgeführt wird. Syntaktisch ähnelt JavaScript Programmiersprachen wie C, C++ und Java durch gleiche Umsetzung der Programmierkonstrukte wie den Bedingungen, Schleifen oder den booleschen Operatoren. [1.1] Wesentliche Unterschiede sind dagegen, dass JavaScript zum einen eine schwach-typisierte Sprache ist. Durch die schwache Typisierung haben Variablen keinen festen Dateityp und können diesen dynamisch zur Laufzeit ändern. Des Weiteren findet bei JavaScript die Objektorientierung prototypenbasiert statt. Diese Form der Programmierung wird auch klassenlose Objektorientierung bezeichnet. Anders als bei der klassenbasierten Programmierung, bei der Objekte aus vordefinierten Klassen instanziiert werden, werden hier Objekte durch Klonen bereits existierender Objekte erzeugt. Die Objekte, die geklont werden, sind dabei als Prototyp-Objekte zu verstehen. Beim Klonen werden alle Attribute und Methoden des Prototyp-Objekts in das neue Objekt übernommen und können dort überschrieben sowie erweitert werden. Objekte in JavaScript sind eher als Zuordnungslisten, ähnlich wie assoziative Arrays oder Hash-Tabellen, anzusehen, da bei der Eigenschaftszuweisung lediglich ein Mapping eines Namens zu seiner zugehörigen Eigenschaft stattfindet. Ein weiterer Unterschied zu den anderen Programmiersprachen ist, dass alle Funktionen und Variablen außer der primären Datentypen Boolean, Zahl und Zeichenfolge, als Objekte verstanden werden können.

3.2.4 Anwendungsgebiete TODO

Ursprünglich fand JavaScript seinen Einsatz hauptsächlich darin, dynamische Webseiten im Web-browser anzuzeigen. Die Verarbeitung erfolgte dabei meist clientseitig durch den Webbrowser (dem sogenannten Frontend). [1.3] Heutzutage findet sich die Sprache dagegen in wesentlich größeren Einsatzgebieten wieder. Bis vor einigen Jahren war die Serverseite anderen Programmiersprachen wie Java oder PHP vorbehalten. Die Veröffentlichung von Node.js, einer plattformübergreifenden Laufzeitumgebung, die JavaScript außerhalb eines Webbrowsers ausführen kann, führte zu einer immer größeren Verbreitung von serverseitigen Anwendungen (dem Backend), die auf JavaScript basieren. Auf Node.js wird ausführlicher im nächsten Kapitel eingegangen. Ferner findet JavaScript heutzutage aber auch seinen Einsatz in mobilen Anwendungen, Desktopanwendungen, Spielen oder 3D-Anwendungen.

3.3 IDE

Hier steht mein IDE Text.

3.4 MongoDB

Hier steht mein Database Text.

Es gibt verschiedene Datenbankmodelle.

3.5 Representational State Transfer - Application Programming Interface

Hier steht mein Database Text.

Es gibt verschiedene Datenbankmodelle.

3.6 Firebase

Hier steht mein Firebase Text.

		Items					
		1	2	...	i	...	m
Users	1	2		1			3
	2	4			5		
	...			1			4
	u		4		5		1
		2				3	
	n		4		3		

Tabelle 1: Nutzer-Item Matrix mit Bewertungen. Jede Zelle $r_{u;i}$ steht hierbei für die Bewertung des Nutzers u an der Stelle i

3.7 Recommendationssystem

Auf der Webseite Youtube allein werden minütlich mehr als 500 Stunden Videomaterial hochgeladen. (https://blog.youtube/press/, 10.02.2021) Um bei einer solch unvorstellbaren Menge an Daten (allein auf einer Webseite) den Überblick als Endnutzer behalten zu können, ist ein personalisierten Filtersystems unausweichlich.

Solche Filtersysteme, auch Recommendation System genannt, nutzt bisher gesammelte Daten um Nutzern potentiell interessante Objekte jeweils individuell vorzuschlagen. Ein sogenannter *Candidate Generator* ist hierbei ein Recommendation System, welches die Menge M als Eingabe erhält und für jeden Nutzer eine Menge N ausgibt. Hierbei umfasst M alle Objekte und gleichzeitig gilt $N \subset M$.

Die Bestimmung einer solchen Menge N beruht grundlegend auf zwei Informationsarten. Erstens die sogenannten Nutzer-Objekt Interaktionen, also beispielsweise Bewertungen oder auch Verhaltensmuster; Und zweitens die Attributwerte von jeweils Nutzer oder Item, also beispielsweise Vorlieben von Nutzern oder Eigenschaften von Items. (Charu C. Aggarwal - Recommender Systems) Systeme, welche zum Bewerten ersteres benutzen, werden *collaborative filtering* Modelle genannt. Andere, welche zweiteres verwenden, werden *content-based filtering* Modelle genannt. Wichtig hierbei ist jedoch, dass *content-based filtering* Modelle ebenfalls Nutzer-Objekt Interaktionen (v.a. Bewertungen) verwenden können, jedoch bezieht sich dieses Modell nur auf einzelne Nutzer - *collaborative filtering* basiert auf Verhaltensmustern von allen Nutzern bzw. allen Objekten.

Ein solches Recommendation System kann im einfachsten Fall wie in 3.7 als Matrix dargestellt werden.

3.7.1 Collaborative Filtering

Unter *collaborative filtering* versteht man das Betrachten von Ähnlichkeiten im Verhalten von Nutzern anhand von Bewertungen und Präferenzen, bzw. anhand der Ähnlichkeiten von Objekten.

Diese Art leidet sehr unter dem *sparcity* Problem, also dass die Nutzer zu wenige Bewertungen von Objekten ausüben. Daher sind Vorhersagen über Ähnlichkeit von Nutzern aufgrund unzureichender Datensätze nicht sinnvoll möglich(siehe 3.7.3).

Generell unterscheidet man in zwei Typen:

1. *Memory-based Methoden*: Es wird, wie oben beschrieben, aus gesammelten Daten Ähnlichkeit herausgearbeitet und Nutzer-Objekt Kombinationen durch eben diese vorhergesagt. Daher wird dieser Typ auch *neighborhood-based collaborative filtering* genannt. Man unterscheidet weiter in:

   (a) *User-based*: Ausgehend von einem Nutzer A werden Nutzer mit ähnlichen Nutzer-Objekt Kombinationen gesucht, um Vorhersagen für Bewertungen von A zu treffen. Ähnlichkeitsbeziehungen werden also über die Reihen der Bewertungsmatrix berechnet.
   (b) *Item-based*: Hierbei werden ähnliche Objekte gesucht und diese genutzt um die Bewertung eines Nutzers für ein Objekt vorherzusagen. Es werden somit Spalten für die Berechnung der Ähnlichkeitsbeziehungen verwendet.

2. *Model-based Methoden*: Machine Learning und Data Mining Methoden werden verwendet um Vorhersagen über Nutzer-Objekt Kombinationen zu treffen. Hierbei sind auch gute Vorhersagen bei niedriger Bewertungsdichte in der Matrix möglich.

(Charu C. Aggarwal - Recommender Systems)

3.7.2 Content-based filtering

3.7.3 Cold Start Problem

`https://dl.acm.org/doi/pdf/10.1145/3383313.3412488` `http://www.microlinkcolleges.net/elib/files/undergraduate/Photography/504703.pdf`
kann man das als Quelle angeben??

4 Konzept?

Test..

5 Auswahl geeigneter Technologie

Hier steht mein TechnologieAuswahl Text...

5.1 Server

Anforderungen sind ...

5.2 Datenbank

Es gibt verschiedene Datenbankmodelle. Bei grossen Datenbank spielt .. eine wichtige Rolle..

5.3 Kommunikationsschnittstelle

Verschiedene Modelle für die Kommunikation verteilter Systeme...

6 Backend-Implementierung

Unter Backend versteht man... , Sie differenziert durch... von den ... .

6.1 Server

Hier steht mein BackendServer Text.

6.1.1 Einrichtung

Zunächst...

6.1.2 Sicherheit

Den Server gilt es zu schützen.

6.1.3 Webserver

Den Server gilt es zu schützen.

6.2 Datenbank

Hier steht mein BackendDatenbank Text.

6.2.1 Einrichtung

Hier steht mein Imlementierung Text.

6.3 Kommunikationsschnittstelle

Hier steht mein BACKENDKommunikationschnittstelle Text.

6.3.1 Implementierung

7 Funktionen/Komponenten

7.1 Swipe/Aussuchen/Voting

7.2 Matches/Chat

7.3 Film-/Serienvorschläge

7.4 Gruppenorgien

7.5 Gespeicherte Filme/Filmliste

7.6 Zugänglichkeit/Behindertenfreundlichkeit

8 Benutzeroberflächen

8.1 Home-Screen

8.2 Gruppen

8.3 Chat

8.4 Filmliste

9 CodeBeispiele

10 Probleme

11 Fazit