

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
3.1.1	Node.JS	2
3.1.2	Architektur TODO Aufzählung	2
3.2	Language	4
3.2.1	JavaScript	5
3.2.2	Historie TODO Aufzählung!	5
3.2.3	Wesentliche Programmiereigenschaften TODO	5
3.2.4	Anwendungsgebiete TODO	6
3.3	IDE	7
3.4	MongoDB	7
3.5	Representational State Transfer - Application Programming Interface	7
3.6	Firebase	7
3.7	Recommendationssystem	7
3.7.1	Collaborative Filtering	7
3.7.2	Content-based filtering	8
3.7.3	Cold Start Problem	8
4	Konzept?	8
5	Auswahl geeigneter Technologie	8
5.1	Server	8
5.2	Datenbank	9
5.3	Kommunikationsschnittstelle	9
6	Backend-Implementierung	9
6.1	Server	9
6.1.1	Einrichtung	9
6.1.2	Sicherheit	9
6.1.3	Webserver	9
6.2	Datenbank	9
6.2.1	Einrichtung	9
6.3	Kommunikationsschnittstelle	9
6.3.1	Implementierung	10
7	Funktionen/Komponenten	10
7.1	Swipe/Aussuchen/Voting	10
7.2	Matches/Chat	10
7.3	Film-/Serienvorschläge	10
7.4	Gruppenorgien	10
7.5	Gespeicherte Filme/Filmliste	10
7.6	Zugänglichkeit/Behindertenfreundlichkeit	10

8 Benutzeroberflächen	10
8.1 Home-Screen	10
8.2 Gruppen	10
8.3 Chat	10
8.4 Filmliste	10
9 CodeBeispiele	10
10 Probleme	10
11 Fazit	10

1 Einleitung

1.1 ..

1.2 Aufbau der Arbeit

2 Motivation

3 Theoretische Grundlagen

3.1 Framework

Hier steht mein Framework Text

3.1.1 Node.JS

Im Jahr 2009 veröffentlichte Ryan Dahl das Framework Node.js, das auf Googles V8-Engine, welche auch als JavaScript-Engine in Googles Browser Chrome zum Einsatz kommt, basiert und sich hervorragend für hochperformante, skalierbare und schnelle Webanwendungen eignet. Zudem ermöglicht es Webentwicklern die Entwicklung von serverseitigem JavaScript-Code. - [NodeJS 1.0]

3.1.2 Architektur TODO Aufzählung

Eine wesentliche Eigenschaft von Node.js ist die hohe Performance. Im Folgenden soll der Unterschied der Node.js-Architektur zu traditionellen Webservern und der damit verbundenen höheren Performance dargestellt werden.

Herkömmliche Webserver erstellen zunächst für jede ankommende Anfrage einen neuen Thread. Dieses Vorgehen ist eng mit steigendem Speicher- und Rechenaufwand verbunden. Um sich Rechenzeit, die durch die Erstellung und Zerstörung von Threads entstanden, zu sparen, wurden Threadpools eingerichtet. Dieser Threadpool enthält mehrere Threads, denen Aufgaben zugewiesen werden können. Nach erfolgreicher Abarbeitung einer Operation kann einem Thread eine weitere Aufgabe zugeordnet werden.

Es bleibt aber ein weiteres Problem: Bei der Anfragenabarbeitung kann es zu einer Form von blockierender Ein- und Ausgabe (Blocking Input/Output kurz Blocking I/O) kommen: zum Beispiel beim Suchen in einer Datenbank oder dem Laden einer Datei im Dateisystem. Während der Abarbeitung wartet der Thread solange, bis die Operation ein Ergebnis zurückwirft und belegt dabei weiterhin Speicherplatz. Bei hohem Anfragen-aufkommen kommt es dadurch zu einer hohen Speicherauslastung des Servers. Zudem kosten die Kontextwechsel zwischen den Threads im Betriebssystem weitere Rechenzeit. [Node.js 1.05] Man spricht bei diesem Architekturkonzept auch vom Multi-Threaded Server.

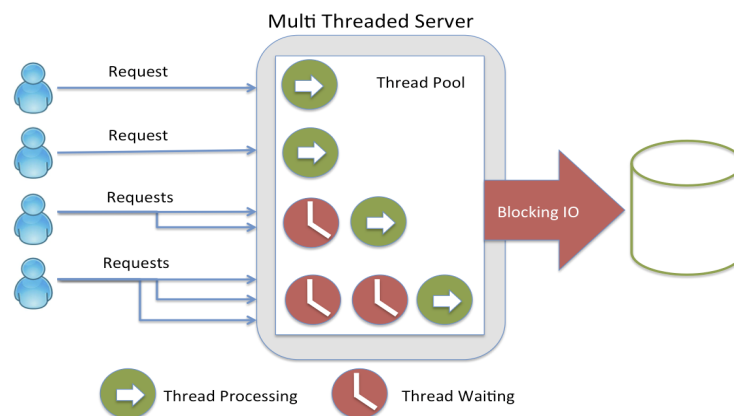


Abbildung 1: Multithreaded / Blocking I/O [Nodejs 1.1]

Node.js verfolgt einen anderen Ansatz: Anfragen werden nur in einem einzigen Thread, dem Hauptthread, abgearbeitet und in einer Warteschlange verwaltet. Dadurch bleiben Kontextwechsel zwischen Threads erspart. Hierbei handelt es sich also um einen Single-Threaded Server. Der Hauptthread verwaltet eine Schleife, die sogenannte Event Loop, die permanent Anfragen aus der Event-Warteschlange überprüft und Ereignisse, die von Ein- und Ausgangsoperationen ausgerufen werden, verarbeitet.

Bei Ankommen einer Nutzeranfrage an einen Node.js Server wird zunächst in der Event Loop geprüft, ob diese Anfrage Blocking I/O benötigt. Falls nicht, kann die Anfrage direkt bearbeitet werden und die Antwort an den Nutzer zurückgesendet werden.

Im anderen Fall wird einer von Node.js interner Workern, welche prinzipiell auch Threads sind, aufgerufen, um die jeweilige Operation auszuführen. Dabei wird eine Callback-Funktion mitgegeben, die vom Worker aufgerufen wird, sobald die Operation ausgeführt wurde. Diese Callback-Funktion kann anschließend als Ereignis von der Event Loop registriert werden. Man spricht hierbei auch von ereignisgesteuerter Architektur. [1.4]

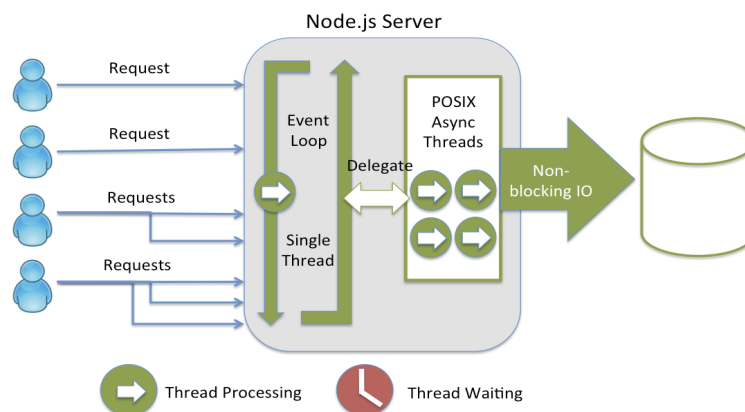


Abbildung 2: Single Threaded / Non Blocking I/O [Nodejs 1.1]

Der große Vorteil hierbei ist, dass der Hauptthread trotz der blockierenden Ein- und Ausgabeoperationen nicht anhält, und weitere Anfragen bearbeiten kann. (Non Blocking I/O - Prinzip)

3.2 Language

Hier steht mein Language Text.

3.2.1 JavaScript

In den nächsten Unterkapiteln soll 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 Aufzählung!

Ihren Ursprung findet die Programmiersprache JavaScript im Jahr 1995, als Brendan Eich, ein damaliger Ingenieur des US-amerikanischen Software-Unternehmens „Netscape Communications Corporation“, innerhalb von zehn Tagen diese 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 zu JavaScript aufgrund der Kooperation von Netscape und Sun, der Firma hinter der Programmiersprache Java, und Marketinggründen. Netscape wollte von der damaligen Popularität von Java profitieren. [1.05]

Netscape's Veröffentlichung des Netscape Navigator 2.0, der erste Browser der JavaScript unterstützte, 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 der Beginn der „Browserkriege“ bekannt. [1.06]

Im Jahre 1997 reichte Netscape JavaScript an die European Computer Manufacturers Association (kurz ECMA[ABK]), einer privaten, internationalen Normungsorganisation zur Normung von Informations- und Kommunikationssystemen und Unterhaltungselektronik ein. Das Ziel war es, von der ECMA einen einheitlichen Standard für die Sprache schaffen zu lassen, die fortan weiterentwickelt werden 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 dieser 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 die neueste Version ECMAScript 2021 veröffentlicht werden. [1.09]

3.2.3 Wesentliche Programmieigenschaften 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 Umsetzungsmöglichkeiten. Interpretation ist in diesem Zusammenhang so zu verstehen, dass der Quellcode zur Laufzeit eines Programms gelesen, übersetzt und ausgeführt wird. Syntaktisch ähnelt JavaScript kompilierten Programmiersprachen wie C, C++ und Java durch gleiche Umsetzung der Kontrollstrukturen 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 Datentyp 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 (dem Key) zu seiner zugehörigen Eigenschaft (dem Value) stattfindet.



Abbildung 3: JavaScript Objekt [1.29]

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 Webbrowser 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. [1.4]

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.

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 personalisiertes 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 *sparsity* 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:

		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

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 Bewertungsichte 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