Modul : MIB 14 -Softwareentwicklungsprojekt
Softwarearchitektur

Software-Projekt
“MovieRent”

Laura Salazar López 5279483
Samira Joulaei Vakili 5301788
Fabius Pudlich 5319932
Timo Trauschold 5299641
Dahun Lee 5277155
Philipp Luis Elgert 5219650

Wintersemester 2021/2022

1 Einführung und Ziele

Aufgabenstellung

Das Team hat sich entschieden eine übersichtlich, schnelle und kostenlose Webapplikation zum Verleih von Filmen zu erstellen, um die regelmäßig wechselnden Filmverleihe in einer Website zu vereinen.

Funktion	Beschreibung	Priorität
/ LF 10 /	Filmsuche	Hoch
/ LF 20 /	Kunden-Konten erstellen	Hoch
/ LF 30 /	Kunden-Konten bearbeiten/löschen	Mittel
/ LF 40 /	Kunden-Konten History einsehbar	Hoch
/ LF 50 /	Preis berechnen (Bußgeld bei überziehen)	Hoch
/ LF 60 /	Verleih aufträge erstellen (wer/welcher Film, bis wann)	Hoch
/ LF 70 /	Kundensuche	Hoch
/LF 80/	Warenkorb	Hoch

Qualitätsziele

Qualitätsziel	Szenarien	Priorität
Simplizität	Anwendungsszenarien sollen einfach ohne viel Aufwand erfüllbar sein.	high priority (1)
Übersichtlichkeit	Funktionen sollen einfach auffindbar sein und gut verständlich.	medium priority (3)
Schnelligkeit	Funktionen sollen schnell durchlaufen werden.	high priority (2)
Stabilität	Anwendungsszenarien sollen stabil und konsistent erfüllt werden	high priority

Stakeholder

Personenname	Rollenname	Erwartungshaltung
Endanwender	Benutzer	- die Dokumentation der Architektur für ihre eigene Arbeit benötigen
Geschäftsführer	Admin	- die Dokumentation der Architektur für ihre eigene Arbeit benötigen - von der Architektur überzeugt werden müssen - mit der Architektur oder dem Code arbeiten
Kunde	indirekter Nutzer	????

2 Randbedingungen

Motivation.

Für eine tragfähige Architektur sollten Sie genau wissen, wo Ihre Freiheitsgrade bezüglich der Entwurfsentscheidungen liegen und wo Sie Randbedingungen beachten

müssen. Sie können Randbedingungen vielleicht noch verhandeln, zunächst sind sie aber da.

Form

Einfache Tabellen der Randbedingungen mit Erläuterungen. Bei Bedarf unterscheiden Sie technische, organisatorische und politische Randbedingungen oder übergreifende Konventionen (beispielsweise Programmier- oder Versionierungsrichtlinien, Dokumentations- oder Namenskonvention).

Randbedingung	**Erklärung**	**Art**
genutzte Technologien	Die Entwickler kannten sich hauptsächlich mit Skripting Sprachen und Webapplikation aus	organisatorisch
Namenskonvention	Es wurde sich auf Englische Namenskonventionen geeinigt, um einheitlichen Code zu erhalten	technisch
Fälligkeit	Das Projekt muss bis zum 17.04.2022 fertiggestellt sein.	organisatorisch

3 Kontextabgrenzung

3.1 Fachlicher Kontext

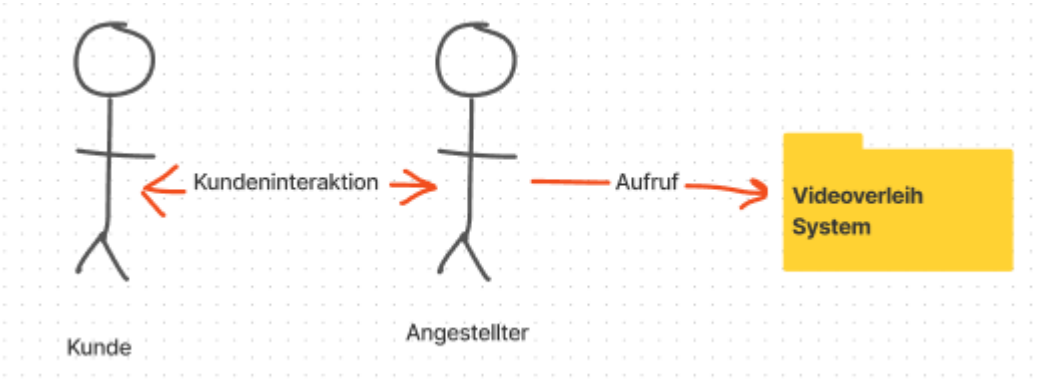

Abb. 1

3.2 Technischer Kontext

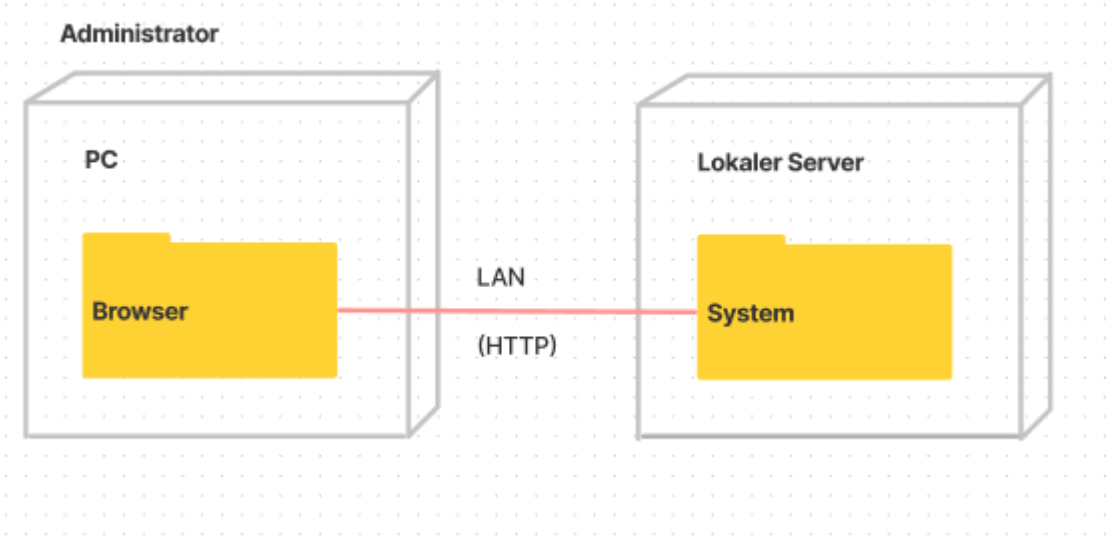

Abb. 2

fachliche Schnittstelle	technischer Kanal
Aufruf	LAN (HTTP)
URL-Aufruf	LAN (HTTP)
Kundendaten	LAN (HTTP)

4 Lösungsstrategie

Die Technologien wurden aufgrund des Wissenstands der Entwickler gewählt und wegen dem Wechsel von festen Programmen zu Webbasierten Anwendungen, vor allem in Streaming/Video Bereich.
Die Suche der Filme in der Datenbank ist besonders wichtig und soll stabil laufen. Auch das Anpassen der Kundendaten und Rückgabe dieser soll schnell und konsistent sein. Das System soll englischen Namenskonventionen folgen, da die Anwendung dadurch globaler erweiterbar ist. Server Abfragen sollen schnell und konsistent sein mit so wenigen Einbrüchen wie möglich. Die graphische Oberfläche sollte simpel und übersichtlich bleiben mit so wenig clutter wie möglich. Außerdem muss die graphische Oberfläche intuitiv bleiben, sodass keine/wenig Einführungen in das System nötig sind. Auch die interne Codierung sollte übersichtlich und verständlich sein, sodass Funktionsweisen auch mit niedriger-mittlerer Technikaffinität verständlich/nachvollziehbar sind. Anpassungen der Datenbank sollten über die Webseite möglich sein, dennoch sorgt eine übersichtliche Struktur für einfachere Änderungen direkt am Server. Insgesamt sind die Qualitätsziele streng zu beachten, da diese leitend für den Systemaufbau sind.

5 Bausteinsicht

5.1 Whitebox Gesamtsystem

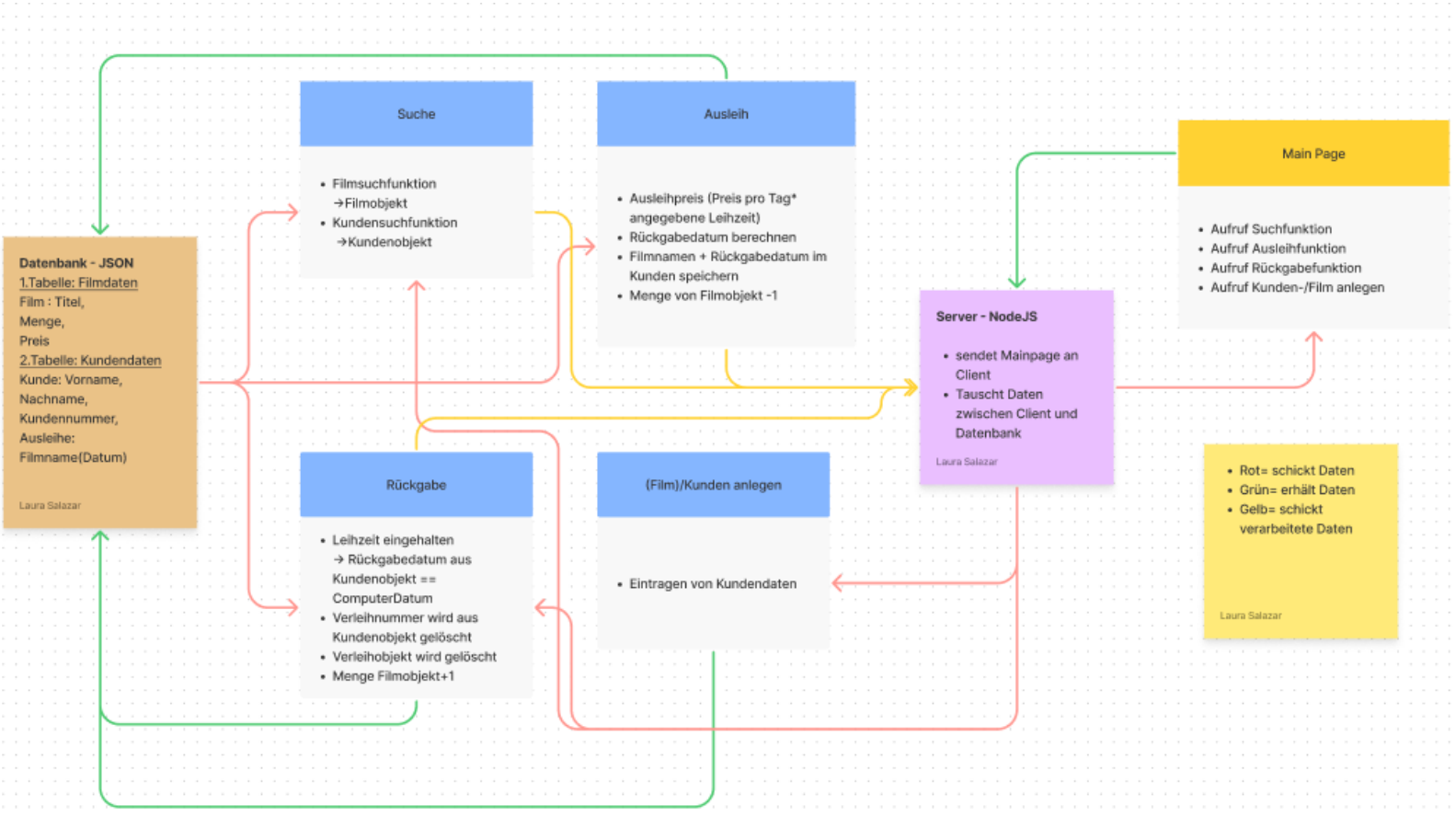

Abb.3

Begründung
Das System besteht aus verschiedenen Bausteinen, welche unabhängig voneinander agieren und Daten austauschen.

Enthaltene Bausteine

- *Datenbank*
- *Suche*
- *Ausleihe*
- *Rückgabe*
- *Mainpage*
- *Objekt anlegen*

Wichtige Schnittstellen
- *Server Schnittstelle*

Name	Verantwortung
Datenbank <database>	Datensicherung: - von Filmdaten <movies.json> - von Kundendaten <users.json>
Schnittstelle <routes>	Auslesen Datenbank: - nach Kundendaten <getUsers()> - nach Filmdaten <getMovies()>
Suche <script.js>	Abgleich Eingabe: - Kundendaten <createMovieSearch()> - Filmdaten <createUsersSearch()> - Vorschlägen <createSuggestions()>
Rückgabe <script.js>	Anzeige Kundendaten und Ausgeliehene Filme <createCustomerResults()>
Ausleihe <script.js>	Einfügen FilmID in Kundenobjekt <saveData()>
Objekt anlegen <script.js>	Anlegen von: - Kundendaten <saveCustomerData()> //- Filmdaten <saveMovieData()>//
Mainpage <index.html>	erhält Usereingaben und schickt diese an entsprechende Funktionen, GUI

5.2 Verfeinerungsebene 2

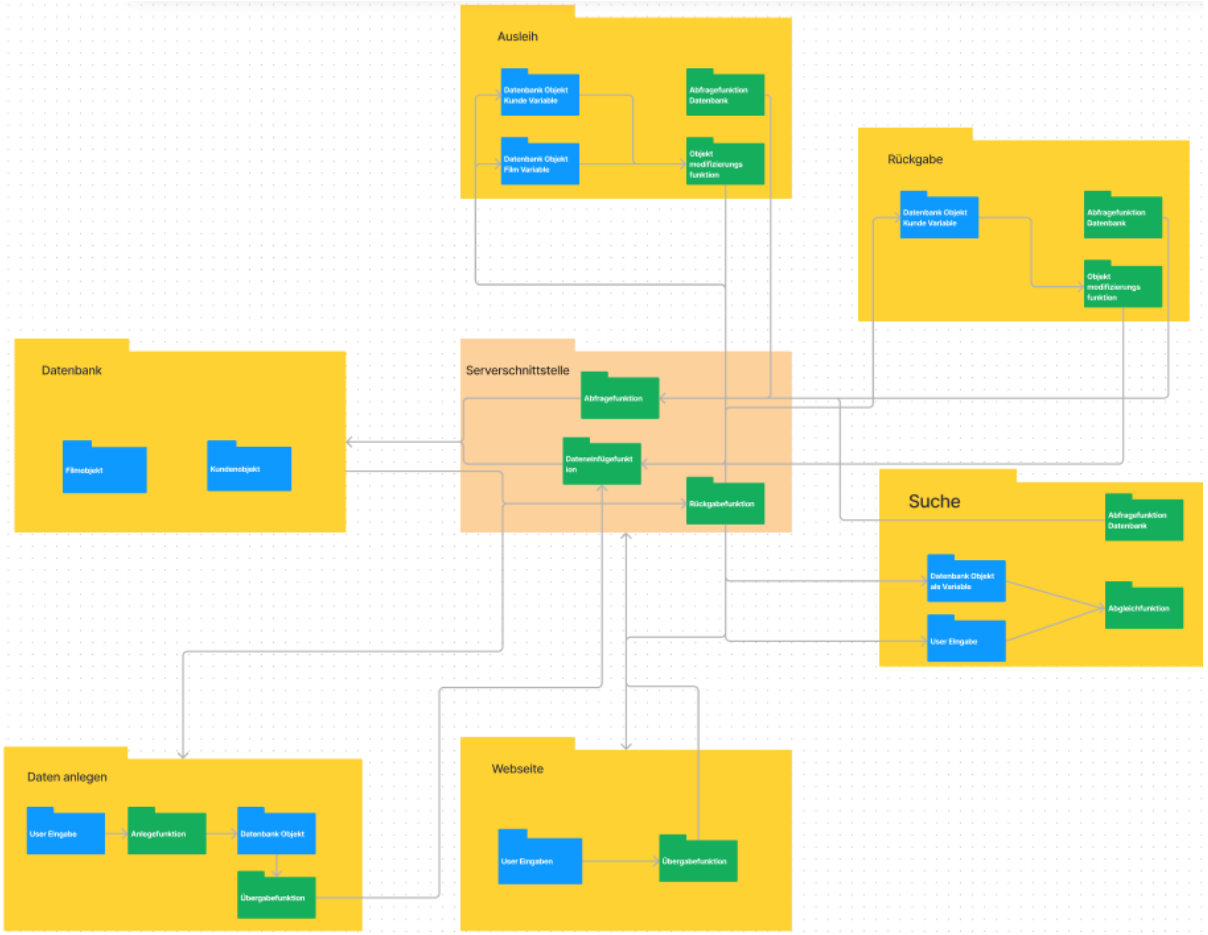

Abb. 4

5.2.1 Datenbank

Tabelle	Inhalt
Filmdaten	• Titel (Name des Films) • Menge (Wie viele Filme verfügbar sind) • Preis (Ausleihpreis)
Kundendaten	• Vorname • Nachname • Kundennummer • Titel • Ausleihdatum

5.2.1.1 Filmdaten

Bezeichnung	Bedeutung
movies.json	Datensicherung von Filmdaten
Übersichtsdiagramm	siehe Abb. 5
lokale Beziehungen und Abhängigkeiten	<routes/index.js(getMovies())>
Entwurfsentscheidungen	

5.2.1.2 Kundendaten

Bezeichnung	Bedeutung
users.json	Datensicherung von Kundendaten
Übersichtsdiagramm	siehe Abb. 5
lokale Beziehungen und Abhängigkeiten	<routes/index.js(getUsers())>
Entwurfsentscheidungen	

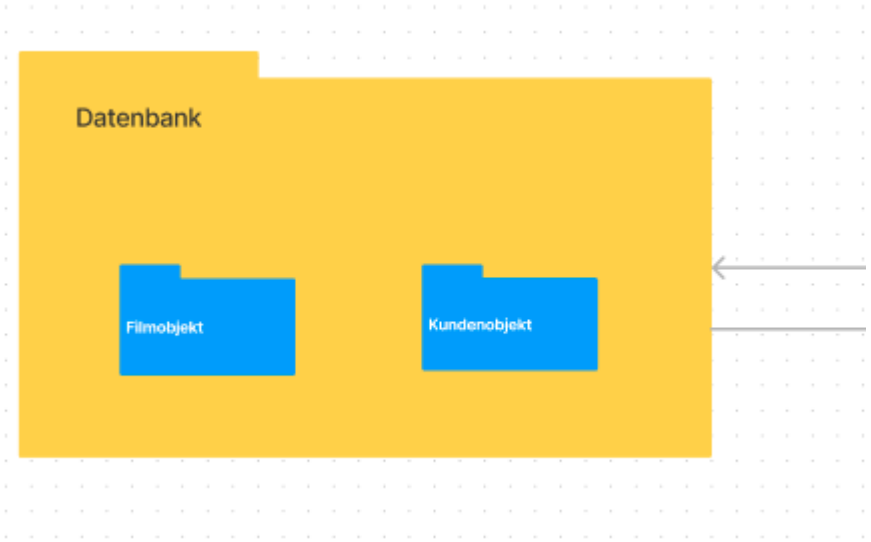

Abb. 5

5.2.2 Suche

5.2.2.1 Filmdaten Suche

Bezeichnung	Bedeutung
<createMovieSearch()>	Suche Filme:: Gleicht Filmdaten aus <routes/index.js(getMovies())> mit Eingabe aus <index.html>
Übersichtsdiagramm	Abb. 6
lokale Beziehungen und Abhängigkeiten	<routes/index.js> <public/javascripts/script.js> <public/index.html>
Entwurfsentscheidungen	

5.2.2.2 Kundendaten Suche

Bezeichnung	Bedeutung
<createUserSearch()>	Suche Kunden:: Gleicht Filmdaten aus <routes/index.js(getUsers())> mit Eingabe aus <index.html>
Übersichtsdiagramm	Abb. 6
lokale Beziehungen und Abhängigkeiten	<routes/index.js> <public/javascripts/script.js> <public/index.html>
Entwurfsentscheidungen	

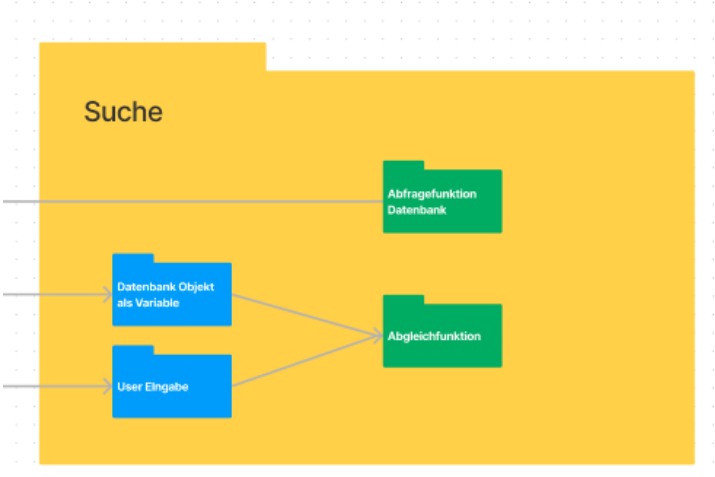

Abb. 6

5.2.3 Rückgabe

Bezeichnung	Bedeutung
<script.js>	Rückgabe :: Auswahl der Filme, die Kunde zurückgeben will <createCustomerResults()>
Übersichtsdiagramm	
lokale Beziehungen und Abhängigkeiten	<routes/index.js> <public/javascripts/script.js> <public/index.html>
Entwurfsentscheidungen	

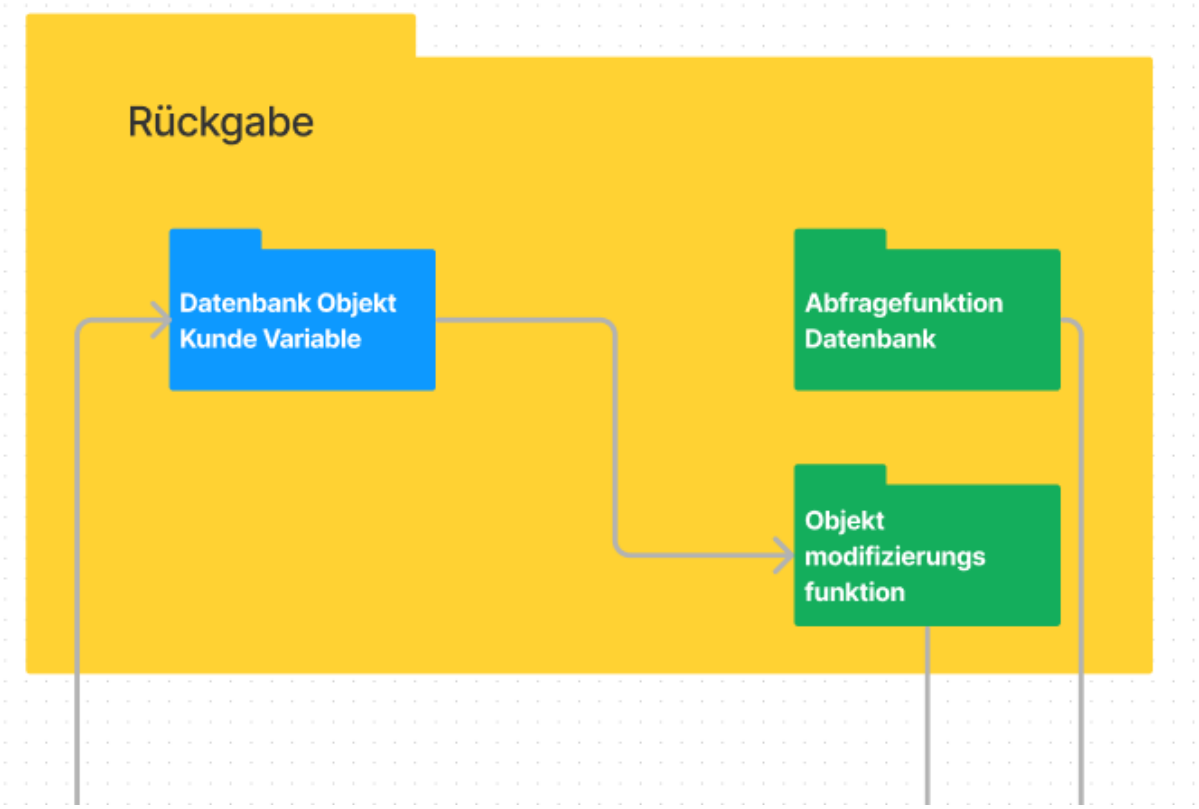

Abb. 7

5.2.4 Ausleihe

Bezeichnung	Bedeutung
<script.js>	Ausleihe:: Einfügen von FIlmID in Kundenobjekt <saveData()>
Übersichtsdiagramm	Abb.7
lokale Beziehungen und	<routes/index.js>

Abhängigkeiten	<public/javascripts/script.js> <public/index.html>
Entwurfsentscheidungen	

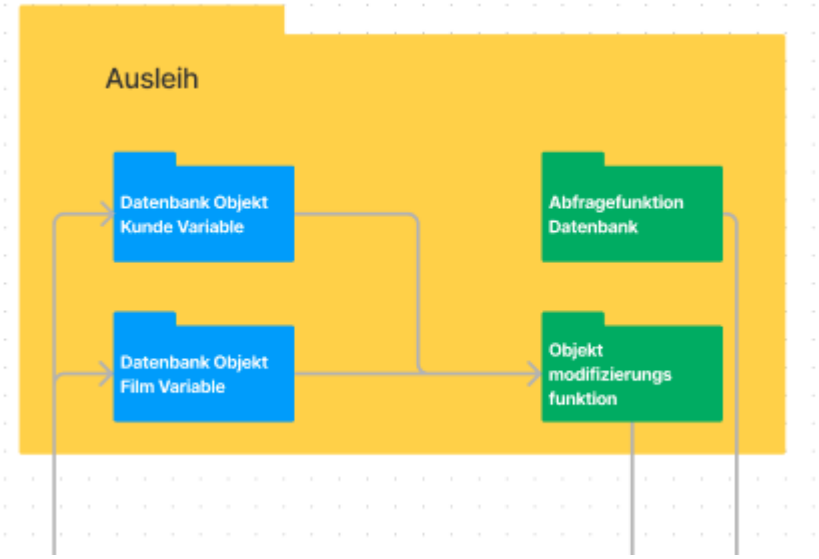

Abb. 8

5.2.5 Objekt anlegen

Bezeichnung	Bedeutung
<script.js>	Kunden anlegen:: saveCustomerData()
Übersichtsdiagramm	Abb.8
lokale Beziehungen und Abhängigkeiten	<routes/index.js> <public/javascripts/script.js> <public/index.html>
Entwurfsentscheidungen	

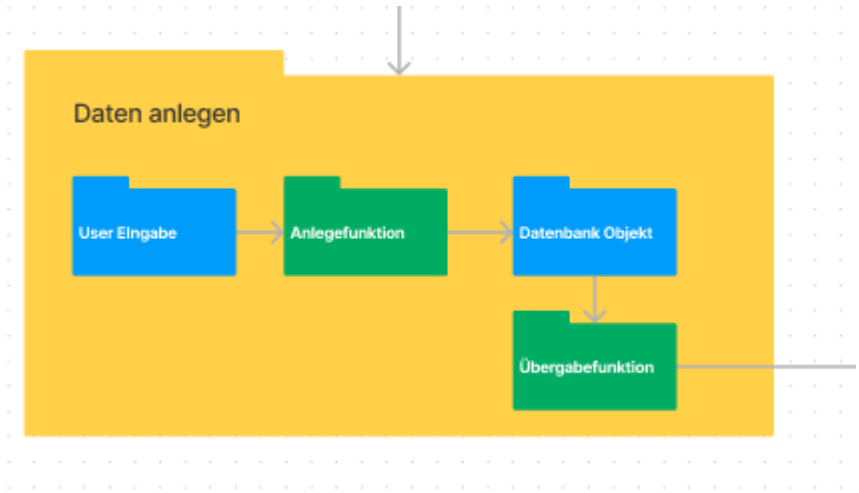

Abb. 9

5.2.6 Main Page

Bezeichnung	Bedeutung
index.html	Anzeige und Eingabe von Daten
Übersichtsdiagramm	
lokale Beziehungen und Abhängigkeiten	<public/stylesheets/style.css> <public/javascript/script.js>
Entwurfsentscheidungen	<ajax/jquery.min.js> <npm/axios/axios.min.js>

5.2.7 Serverschnittstelle

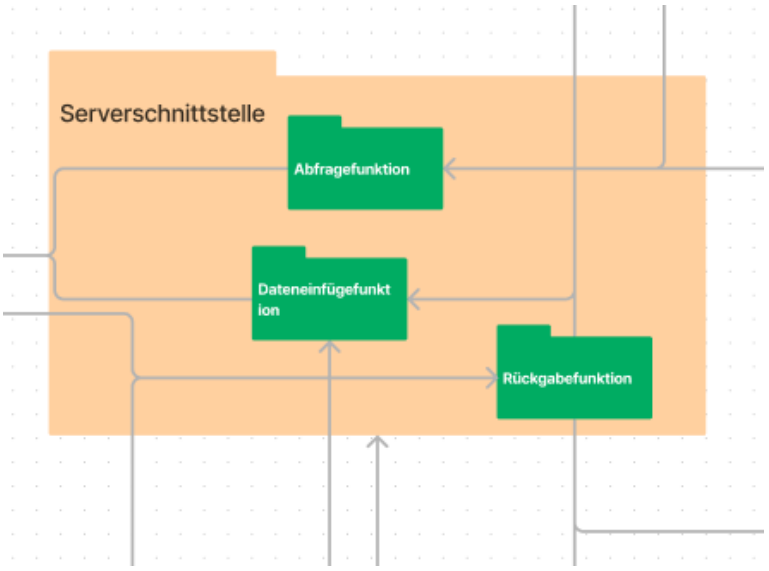

Abb. 10

Bezeichnung	Bedeutung
routes/index.js	Auslesen Daten aus Json <get_data()>
Übersichtsdiagramm	Abb. 9
lokale Beziehungen und Abhängigkeiten	<database/movies.json> <database/users.json> <public/javascript/script.js>
Entwurfsentscheidungen	

5.2.8 Sekundär Funktionen

Name	**Beschreibung**
removeMovie()	● Filme aus dem Warenkorb entfernen
kartView()	● Warenkorb ein/-ausblenden
showWindow()	● Fenster der Suchergebnisse ● einblenden
dim()	● Seite beim Klick in das Suchfeld abdunkeln
undim()	● Abdunklung aufheben
saveData()	● Beim Checkout die Daten an den Server schicken und übertragen

6 Laufzeitsicht

6.1 Request Verarbeitung

Ablauf:

1.Benutzer ruft am PC die URL auf (für Entwicklung localhost:3000

aber nach der Veröffentlichung von MovieRent ist die Seite über z.B. https://movie-rent.de aufrufbar).
2. Die Seite index.html wird vom Server an den Client geschickt und die index.html wird im Browser geöffnet. Die zugehörige Javascript und CSS Datei wird geladen, weil diese in der index.html eingebunden sind.
3. Nachdem die Dateien geladen wurden, wird das CSS angewendet und das Javascript ausgeführt. Weil wir in der Javascript Datei ein Page-Ready-Event benutzen, wird somit direkt nach dem Aufrufen der Startseite die Funktion getData() ausgeführt.
Mit der Funktion getdata() wird ein Request an den Server gesendet.
Um Daten von der Webseite aus vom Server anzufordern, benutzen wir die Javascript-Library Axios https://axios-http.com/docs/intro.
Die Route für die Filmdaten heißt /movies und ist auch in der Datei routes/index.js zu finden, weil so dort im Backend “registriert” bzw. einfach erstellt/gecoded werden muss.
Wenn diese Route von der getdata() Funktion aus dem Frontend aufgerufen wird, liefern wir mit res.json(movies) das Array von Filmen zum Client.

7 Verteilungssicht

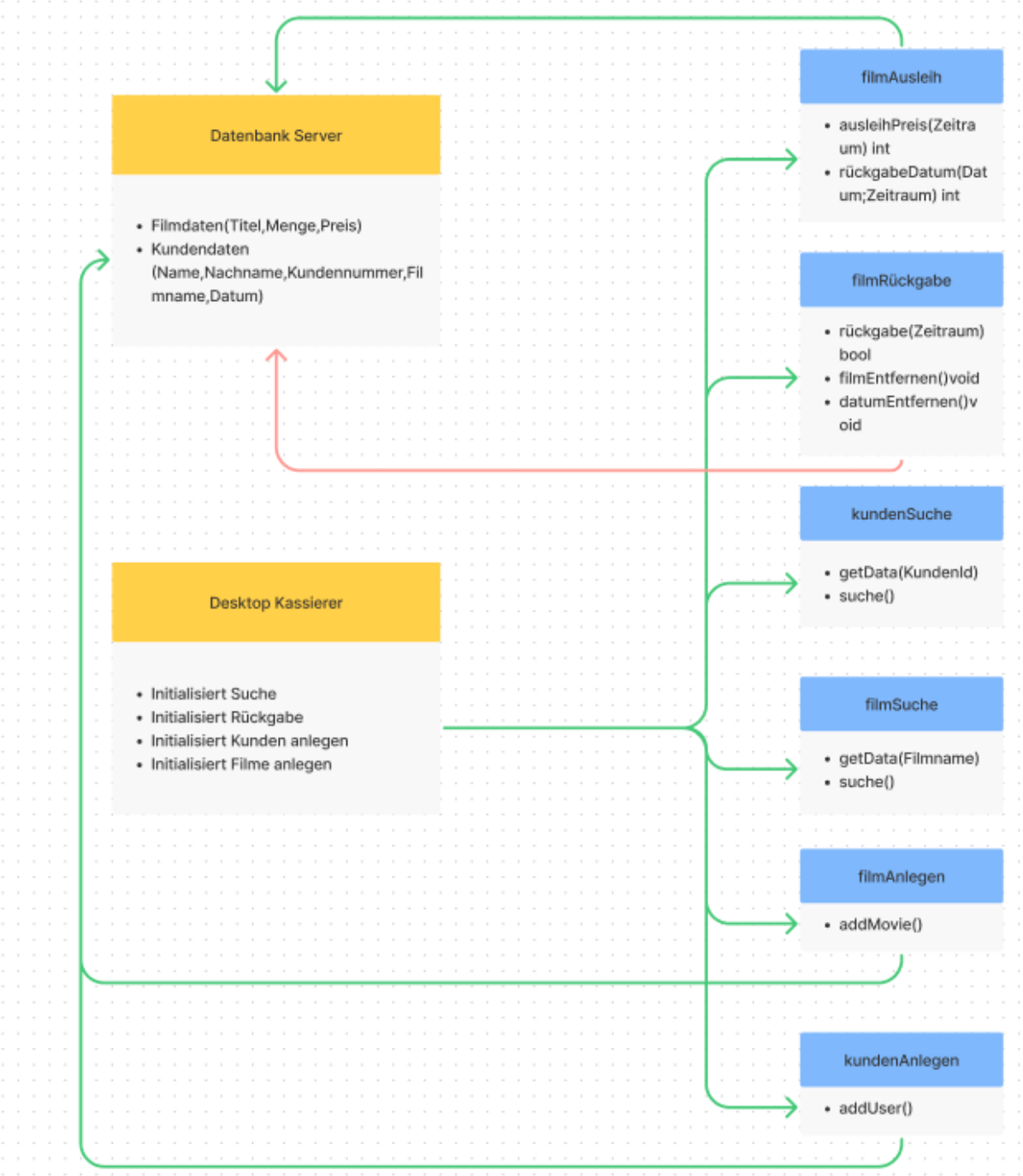

Abb. 11

Da das gesamte System spezifischen Lokalen zugehört, kann die gesamte Infrastruktur auf einem Computer laufen. Dabei relevant sind die Aspekte: Festplatte, Speicher, CPU. Der Computer sollte ebenfalls mit einem Webbrowser ausgestattet sein.

Verteilung	Begründung	Qualitäts/Leistungsmerkmale	Softwareartefakte
Festplatte	- beinhaltet Gesamt- Software und speicher diese	- großen Datenspeicher (z.B. 1TB) zur Erweiterung Filmdaten	- Server - Datenbank - Dateien
Arbeitsspeicher	- Zwischenspeicher für CPU Berechnungen	- schnell,groß, um Userinteraktionen besser auszuführen (8 GB RAM)	- Server - Datenbank - Dateien
CPU	- berechnet Interaktionen und Softwareinterne Prozesse	- schnell um Userinteraktionen und interne Prozesse gut auszuführen (2,5 gHz mit 8 Kernen)	- Funktionen - Input
Webbrowser	- visuelle Darstellung Software	- Geschwindigkeit, der anderen Bausteine einhalten	- GUI - Inputs
interne LAN Verbindung	- Verbindung zwischen Teilen	-	???

8 Querschnittliche Konzepte

8.1 Persistenz

Die Speicherung von Daten (Film+Kundendaten) findet in diesem System in zwei JSON Dateien statt. JSON (JavaScript Object Notation) ist ein Textformat zum Speichern und Transport von Daten. JSON ist relativ leicht erlernbar und benötigt wenig Speicherplatz. Außerdem ist das Format der JSON Datei ähnlich dem Aufbau eines JavaScript Objects, sodass eine Konvertierung einfach stattfinden kann. Da 2 relevante Datenmengen (Kunden und Filme) gespeichert werden müssen, wurden 2 Dateien angelegt welche jeweils eine Datensammlung enthält.

Diese Dateien sind folgendermaßen aufgebaut:

Beispiel database/movies.json:

```
[
```

```
{
“title” : ”The Avengers”,
“available” : ”5”,
“price” : “2€”
},
… {weitere Filme}
]
```

Beispiel database/users.json:

```
[
{
“forename” : “Peter”,
“surname” : “Parker”,
“cNumber” : “12345”
},
… {weitere Kunden}
]
```

Um auf die JSON Dateien auslesen zu können wird das “Express” Node Module Package (http://expressjs.com/) verwendet. welche de Zugriff auf die `getData()` *Funktion erlaubt:*

Beispiel routes/index.js:

```
getMovies(function(cb){
data.movies 0 cb.movies
res.json({data: data});
});

var getMovies = function(cb){
fs.readFile(‘database/movies.json’,’utf8’, function(err,movies){
if(err){return console.log(err)};
cb{movies: JSON.parse(movies)});
});
}
```

Diese Datenbank läuft auf einer Node.js (https://nodejs.org/en/about/) Serverumgebung. Node verwendet JavaScript serverseitig und arbeitet dabei asynchron. Node wird zur Erstellung dynamischer Webseiten verwendet und kann Dateien auf einem Server erstellen, öffnen, bearbeiten und löschen. Außerdem sammelt Node Formular Daten, welches zur Erstellung neuer Film- oder Kundendaten relevant ist. Die Modifizierung der JSON Datenbanken findet ebenfalls über Node statt.
Node verwendet verschiedene eingebaute Module, welche ähnlich der Libraries von JS funktionieren. Um den Server als HTTP Server agieren zu lassen, verwendet das System das HTTP-Modul (https://www.w3schools.com/nodejs/ref_http.asp).

Das Application Skelett verwendet folgende Struktur:

```
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug
```

Neben einem Grundgerüst für die Ordnerstruktur bietet Express HTTP Helfer und konsistente Routing Performance.

Zur Abfrage dieser Datenbank nutzen wir das Javascript Framework Axios (https://axios-http.com/docs/intro), welches ein HTTP-Client für Node.js und des Browsers ist. Axios verwendet serverseitig Node Module und clientseitig XMLHttpRequests. Zur Anfrage von Daten wird zunächst also ein XMLHttpRequest von dem Client an den Server geschickt. Darauf antwortet der Server mit den angefragten Daten innerhalb eines Arrays, welches als Variable gespeichert werden. Diese Variable existiert zur Weiterverarbeitung der Daten, wird danach jedoch gelöscht. Mit Hilfe des Arrays können abgespeicherte Daten auf der Benutzeroberfläche angezeigt und innerhalb des Codes angepasst werden, sodass ein Kundenobjekt beispielsweise den ausgeliehenen Filmnamen abspeichert. Änderungen dieser Daten können erst mit Hilfe der Variable gemacht werden ohne zu "hardcoden", also direkt in der Datenbank Daten einzufügen.

8.2 Coding Guidelines

Zur Vereinheitlichung des Programmierens wurde beschlossen Variablennamen in englischer Sprache zu halten. Außerdem sollen die Regelungen des "Clean Code" befolgt werden, das heißt klar verständliche Variablennamen, die intuitiv die Funktion eines Codeabschnitts vermitteln. Auch die Konventionen wie Camelcases und Formatierung sollten befolgt werden. Kommentare sollten so kurz wie möglich und lang wie nötig sein, hauptsächlich soll jedoch jeder Codeabschnitt auch ohne Beschreibung verständlich sein. Zweideutige oder redundante Variablennamen sollten möglichst verhindert werden.

Da die 100-prozentige Einhaltung der Clean Code Regelungen nicht möglich sind, muss sich auf die Best Practices der Entwickler verlassen werden bzw. regelmäßige Code Reviews (min. einmal der Anpassung des Codes) zur Überprüfung dieser gehalten werden.

9 Entwurfsentscheidungen

Entscheidung	Erklärung
NodeJs	hat Php Funktionalität in Js Syntax -> keine neu zu erlernende Syntax
Axios Library	Anforderung von Daten vom Server (HTTP Requests)
Url -> Webserver	Benötigt um außerhalb von Lokaler Umgebung auf Server zuzugreifen
Express.js	ermöglicht Serverseitige einrichtung
NPM Package	Zugriff auf vorgefertigte Funktionen
File System	Zugriff auf Dateisystem

10 Glossar

Begriff	Definition
Button	Eine englische Übersetzung für das deutsche Wort Knopf (oder Schaltfläche). Der Button ist ein Bedienelement in grafischen Benutzeroberflächen, das dem Benutzer ermöglicht, eine zugeordnete Funktion auszulösen
Dropdown-(Menü)	Das Dropdown-Menü ist eine spezielle Form des Auswahlmenüs. Es erscheint eine Auswahlliste auf dem Bildschirm.

Element	Grundbestandteil oder Komponenten, aus dem mit anderen zusammen etwas aufgebaut wird.
Frustrationen	Was die Arbeiter an ihrem System stört
History	Mit der History ist eine Chronik oder ein Verlauf gemeint: Eine Liste von zuletzt durchgeführten Aktionen.
Main-Page	Die Main-Page ist die Startseite des Programms.
Pop-Up (Fenster)	Pop-Up Fenster sind Bedienfelder, die sich automatisch öffnen, und der Benutzer einen Text (oder ähnliche Elemente) gezeigt bekommt.
Skript	Ein Skript ist eine kurze Abfolge von Befehlen oder Funktionen, die sich in einer Webseite integrieren lässt.
webbasiert	Aus dem World Wide Web beruhend bzw. mit ihm arbeitend
Best Practices	bestmögliche [bereits erprobte] Methode, Maßnahme o. Ä. zur Durchführung, Umsetzung von etwas
HTTP	Protokoll, welches die Datenübertragung zwischen Anwendungen regelt
Modul	austauschbares, komplexes Element innerhalb eines Gesamtsystems, eines Gerätes oder einer Maschine, das eine geschlossene Funktionseinheit bildet
URL	Adresse einer einzelnen Webseite
clutter	Unordnung
Client	Programm, welches Dienste eines Servers in Anspruch nimmt
Server	Rechner, der für andere in einem Netzwerk mit ihm verbundene Systeme bestimmte Aufgaben übernimmt und von dem diese ganz oder teilweise abhängig sind