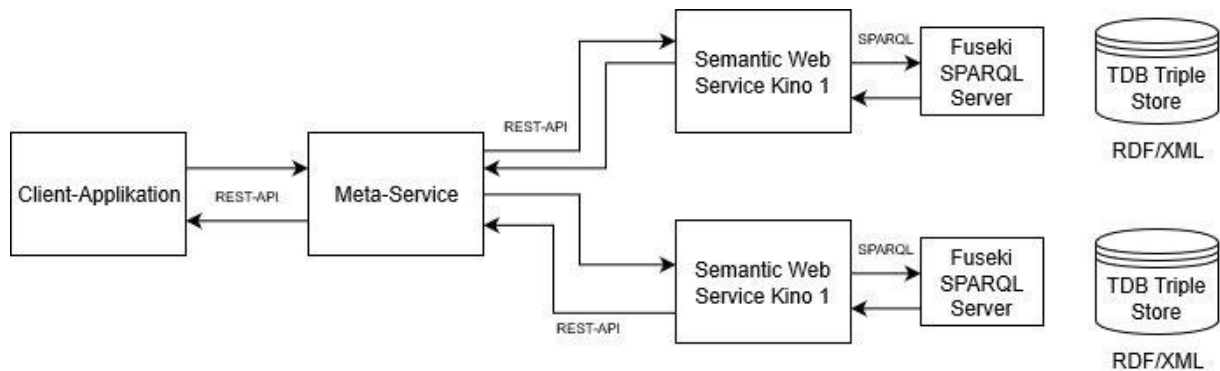


Lösungsarchitektur



Die ausführliche Beschreibung zur Lösungsarchitektur befindet sich im File PRDKE_Endpräsentation.

Installationsanleitung

Fuseki Server

Diese Installationsanleitung geht von einer Verwendung auf einem Rechner mit dem Betriebssystem Windows aus. Java 11 muss am Rechner installiert sein und eine IntelliJ Entwicklungsumgebung wird benötigt.

Um unsere Applikation starten zu können bedarf es eines Apache Jena Fuseki Servers. Dieser kann auf der Seite <https://jena.apache.org/download/> downgeloaded werden.

Apache Jena Fuseki

The binary distribution of Fuseki2 (this includes both the standalone and WAR file packaging) may be downloaded at:

[apache-jena-fuseki-3.14.0.tar.gz \(SHA512, PGP\)](#)

[apache-jena-fuseki-3.14.0.zip \(SHA512, PGP\)](#)

Individual Modules

Apache Jena publishes a range of modules beyond those included in the binary distributions (code for all modules may be found in the source distribution).

Individual modules may be obtained using a dependency manager which can talk to Maven repositories, some modules are only available via Maven.

Maven

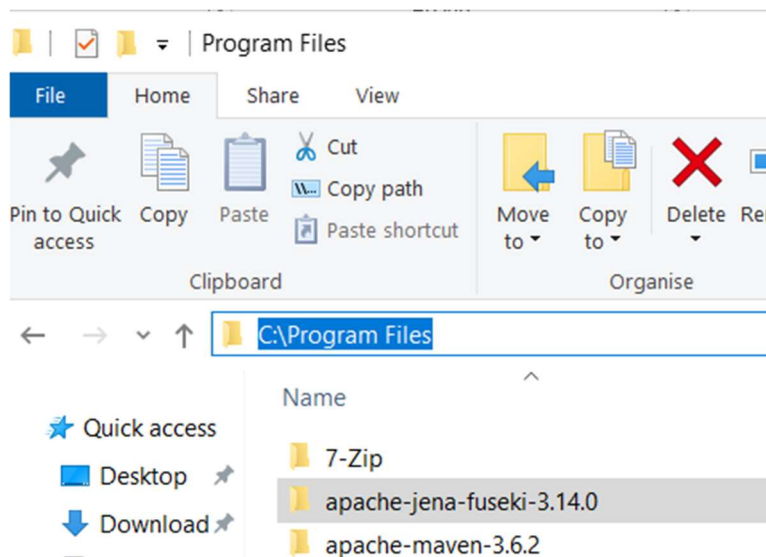
See "Using Jena with Apache Maven" for full details.

```
<dependency>
  <groupId>org.apache.jena</groupId>
  <artifactId>apache-jena-libs</artifactId>
  <type>pom</type>
  <version>X.Y.Z</version>
</dependency>
```

Source code

The development codebase is available from git.

Das downgeloadete Verzeichnis an einer beliebigen Stelle am Rechner entpacken und speichern. Der Fuseki Server kann anschließend via Commandline im Admin Modus gestartet werden. Dazu muss in der Commandline in das eben entpackte Verzeichnis navigiert werden und folgender Befehl abgesetzt werden: `java -jar fuseki-server.jar`



Nach Aufruf des oben angeführten Befehls sollte der Fuseki Server starten und keine Fehler ausgeben.

```
Administrator: Command Prompt - java -jar fuseki-server.jar
C:\Program Files>cd apache-jena-fuseki-3.14.0
C:\Program Files\apache-jena-fuseki-3.14.0>java -jar fuseki-server.jar
[2020-01-23 11:56:22] Server INFO Apache Jena Fuseki 3.14.0
[2020-01-23 11:56:23] Config INFO FUSEKI_HOME=C:\Program Files\apache-jena-fuseki-3.14.0\
[2020-01-23 11:56:23] Config INFO FUSEKI_BASE=C:\Program Files\apache-jena-fuseki-3.14.0\run
[2020-01-23 11:56:23] Config INFO Shiro file: file:///C:/Program Files/apache-jena-fuseki-3.14.0/run\shiro.ini
[2020-01-23 11:56:23] Config INFO Configuration file: C:\Program Files\apache-jena-fuseki-3.14.0\run\config.ttl
[2020-01-23 11:56:24] riot WARN [line: 5, col: 9 ] Bad IRI: <C:\Program Files\apache-jena-fuseki-3.14.0\run\config.ttl#> Code: 4/UNWISE_CHARACTER in PATH: The character matches no grammar rules of URIs/IRIs. These characters are permitted in RDF URI References, XML system identifiers, and XML Schema anyURIs.
[2020-01-23 11:56:24] riot WARN [line: 5, col: 9 ] Bad IRI: <C:\Program Files\apache-jena-fuseki-3.14.0\run\config.ttl#> Code: 17/WHITESPACE in PATH: A single whitespace character. These match no grammar rules of URIs/IRIs. These characters are permitted in RDF URI References, XML system identifiers, and XML Schema anyURIs.
[2020-01-23 11:56:24] Config INFO Load configuration: file:///C:/Program%20Files/apache-jena-fuseki-3.14.0/run/configuration/cinema1.ttl
[2020-01-23 11:56:25] Config INFO Load configuration: file:///C:/Program%20Files/apache-jena-fuseki-3.14.0/run/configuration/cinema2.ttl
[2020-01-23 11:56:26] Config INFO Register: /cinema2
[2020-01-23 11:56:26] Config INFO Register: /cinema1
[2020-01-23 11:56:26] Server INFO Started 2020/01/23 11:56:26 CET on port 3030
```

Anschließend kann der Fuseki Server im Browser über localhost und dem angegebenen Port gestartet werden. (In diesem Fall 3030) Der Befehl <http://localhost:3030/> muss in die Adresszeile im Browser eingegeben werden.

Die Benutzeroberfläche des Fuseki Servers sollte sich öffnen. Bei erstmaligem Aufruf existieren noch keine Datasets. Wenn bereits welche angelegt wurden, werden diese hier angezeigt.

The screenshot shows the Apache Jena Fuseki main dashboard. At the top, there is a navigation bar with the Apache Jena Fuseki logo, a home icon, and links for 'dataset', 'manage datasets', and 'help'. The server status is indicated as 'Server status: [green dot]'. The main heading is 'Apache Jena Fuseki' with the version 'Version 3.14.0' and uptime 'Uptime: 2m 04s'. Below this, it says 'Datasets on this server' and 'There are no datasets on this server yet. [Add one.](#)'. A light blue box contains instructions: 'Use the following pages to perform actions or tasks on this server:' followed by links for 'Dataset' (Run queries and modify datasets), 'Manage datasets' (Administer the datasets), and 'Help' (Summary of commands).

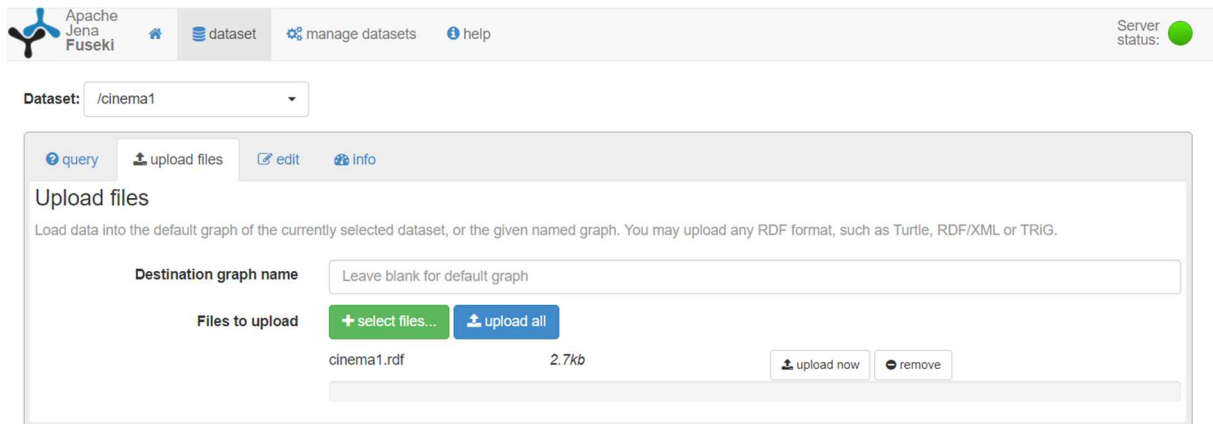
Um Datasets anzulegen, muss zu “manage datasets” navigiert werden. Unter “add new dataset” kann ein neues dataset angelegt werden. Dazu einmal ein dataset mit dem Namen “cinema1” und eines mit dem Namen “cinema2” (Kleinschreibung beachten) anlegen und unbedingt bei Dataset type “Persistent (TDB2)” auswählen. Dies gewährleistet, dass unsere Änderungen persistent sind.

This screenshot shows the 'Manage datasets' page. The 'add new dataset' tab is active. The 'Dataset name' field contains 'cinema1'. Under 'Dataset type', three options are listed: 'In-memory' (dataset will be recreated when Fuseki restarts, but contents will be lost), 'Persistent' (dataset will persist across Fuseki restarts), and 'Persistent (TDB2)' (dataset will persist across Fuseki restarts). The 'Persistent (TDB2)' option is selected. A 'create dataset' button is at the bottom right.

Sind beide Datasets angelegt, können unter “manage datasets” via “upload data” für das jeweilige Kino rdf Daten hochgeladen werden. Diese rdf Daten können entweder selbst angelegt werden, oder bereits bestehende werden benutzt.

This screenshot shows the 'Manage datasets' page with the 'existing datasets' tab selected. It lists two datasets: '/cinema1' and '/cinema2'. For each dataset, there are three buttons: 'remove', 'backup', and 'upload data'.

Dazu muss unter “select files” das gewünschte *.rdf File gesucht und ausgewählt werden. Via “upload all” oder “upload now” können die Daten hochgeladen und gespeichert werden.



Apache Jena Fuseki

dataset manage datasets help

Server status: ●

Dataset: /cinema1

query upload files edit info

Upload files

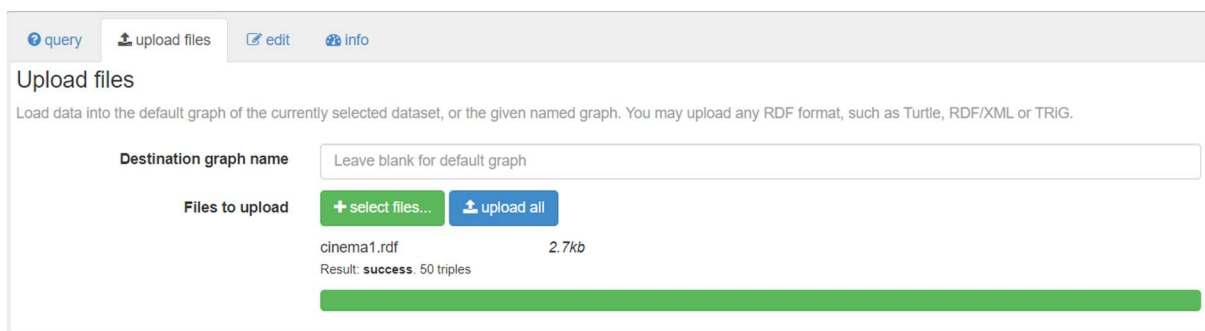
Load data into the default graph of the currently selected dataset, or the given named graph. You may upload any RDF format, such as Turtle, RDF/XML or TRIG.

Destination graph name:

Files to upload: + select files... upload all

cinema1.rdf 2.7kb upload now remove

Der Benutzer wird anschließend über einen erfolgreichen oder fehlgeschlagenen Upload informiert.



query upload files edit info

Upload files

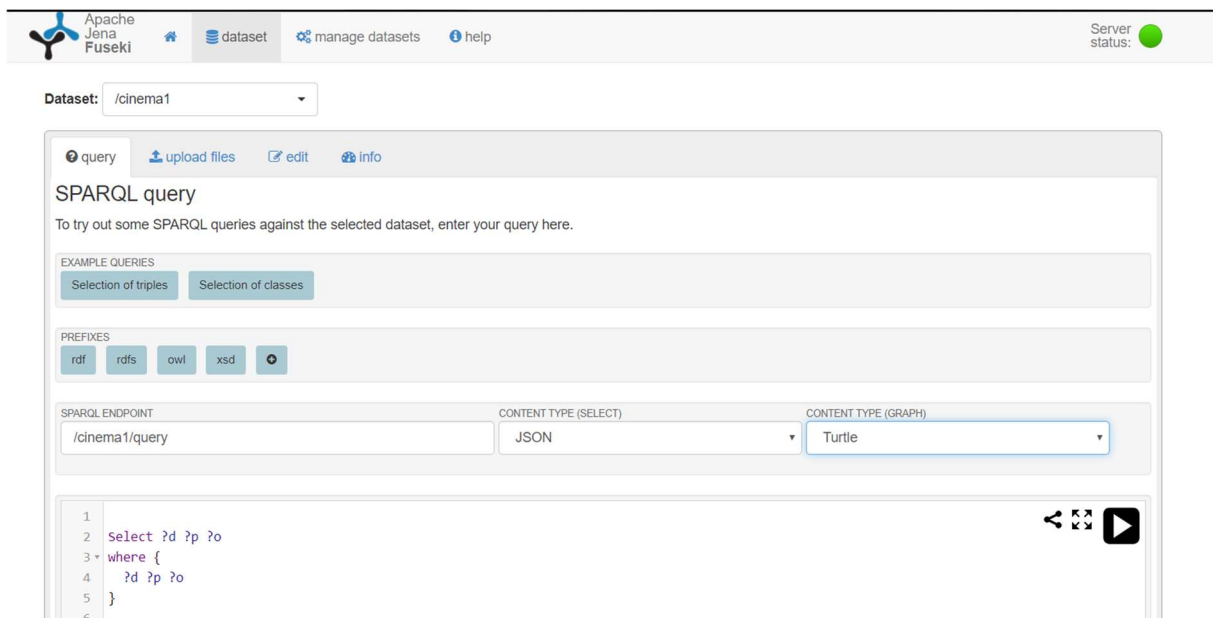
Load data into the default graph of the currently selected dataset, or the given named graph. You may upload any RDF format, such as Turtle, RDF/XML or TRIG.

Destination graph name:

Files to upload: + select files... upload all

cinema1.rdf 2.7kb
Result: **success**: 50 triples

Wurden beide Cinema Daten hochgeladen, können via “Dataset” Section Sparql queries auf die jeweiligen Cinemas abgesetzt werden. (Für den Ablauf der Applikation ist dies nicht notwendig)



Apache Jena Fuseki

dataset manage datasets help

Server status: ●

Dataset: /cinema1

query upload files edit info

SPARQL query

To try out some SPARQL queries against the selected dataset, enter your query here.

EXAMPLE QUERIES: Selection of triples Selection of classes

PREFIXES: rdf rdfs owl xsd +

SPARQL ENDPOINT:

CONTENT TYPE (SELECT): JSON

CONTENT TYPE (GRAPH): Turtle

```
1
2 select ?d ?p ?o
3 where {
4   ?d ?p ?o
5 }
6
```

↶ ↷ ▶

QUERY RESULTS

Table Raw Response

Showing 1 to 100 of 100 entries

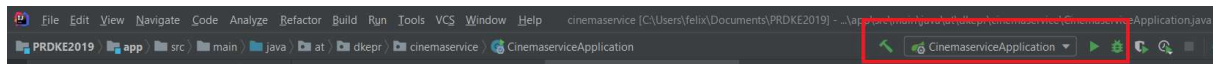
Search: Show **All** entries

	d	p	o
1	<http://www.cinemas.fake/megaplex/movies/interstellar>	<http://www.cinemas.fake/megaplex/movies/info#title>	"Interstellar"
2	<http://www.cinemas.fake/megaplex/movies/interstellar>	<http://www.cinemas.fake/megaplex/movies/info#genre>	"Science Fiction"
3	<http://www.cinemas.fake/megaplex/movies/interstellar>	<http://www.cinemas.fake/megaplex/movies/info#year>	"2014"
4	<http://www.cinemas.fake/megaplex/movies/interstellar>	<http://www.cinemas.fake/megaplex/movies/info#language>	"Englisch"
5	<http://www.cinemas.fake/megaplex/movies/interstellar>	<http://www.cinemas.fake/megaplex/movies/info#duration>	"169"
6	<http://www.cinemas.fake/megaplex/movies/interstellar>	<http://www.cinemas.fake/megaplex/movies/info#FSK>	"12"

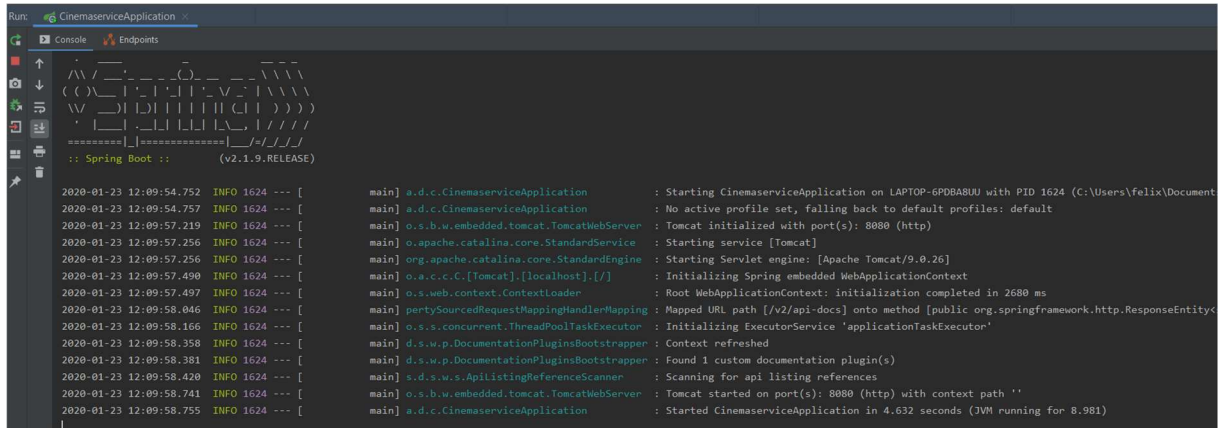
Nun sind unsere Datensätze konfiguriert.

Backend

Um unsere Applikation zu starten, muss unser Projekt in IntelliJ geöffnet werden. Beim Starten des (Springboot basierten) Backend ist zu beachten, dass die CinemaserviceApplication gestartet wird.



Über die Console kann verfolgt werden, ob der Start erfolgreich war.



Anschließend können via localhost und dem angegebenen port (in diesem Fall 8080) Daten im Triplestore abgefragt werden. (<http://localhost:8080/Cinema2/Movies/Fr>)

Hier werden Rest Calls abgesetzt und die Responses anschließend im Browser angezeigt. Die Rest Calls werden dabei auf die implementierten Funktionen in Cinema1 und Cinema2 (RestController) via RequestMapping Annotation gemapped. Die hier aufgerufenen und gemappten Funktionen rufen dann in einem weiteren Schritt anschließend die Sparql Verarbeitung in einer weiteren Klasse auf.

Die unterstützten Abfragen hierbei sind:

Kino	Gemappte Funktion	Syntax	Beispielhafter Aufruf
Cinema1	getMoviesByDay(String day)	/Cinema1/Movies/{day}	http://localhost:8080/Cinema1/Movies/Fr
Cinema1	getAllReservations()	/Cinema1/Reservations/All	http://localhost:8080/Cinema1/Reservations/All
Cinema1	getReservationsByMovie(String movie)	/Cinema1/Reservations/{movie}	http://localhost:8080/Cinema1/Reservations/Joker
Cinema1	removeReservation(String movie, String reservation)	/Cinema1/Remove/Reservation/{movie}	Aufruf erfolgt eigentlich nur via Frontend
Cinema1	addReservationToMovie(String movie, String reservation)	/Cinema1/Create/Reservation/{movie}	Aufruf erfolgt eigentlich nur via Frontend
Cinema2	getMoviesByDay(String day)	/Cinema2/Movies/{day}	http://localhost:8080/Cinema2/Movies/Fr
Cinema2	getAllMovies()	/Cinema2/Movies/All	http://localhost:8080/Cinema2/Movies/All
Cinema2	getReservationName(String name)	/Cinema2/Movies/Reservations/{name}	http://localhost:8080/Cinema2/Movies/Reservations/Felix

Cinema2	getMoviesPerGenre(String genre)	/Cinema2/Movies/Genre/{genre}	http://localhost:8080/Cinema2/Movies/Genre/Horror
Cinema2	getMenus()	/Cinema2/Movies/Menus	http://localhost:8080/Cinema2/Movies/Menus
Cinema2	addReservationToMovie(String movie, String reservation)	/Cinema2/Create/Reservation/{movie}	Aufruf erfolgt eigentlich nur via Frontend
Cinema2	addMenuToMovie(String Menu, String movie)	/Cinema2/Create/Menu/{movie}	Aufruf erfolgt eigentlich nur via Frontend
Cinema2	removeMenu(String movie, String menu)	/Cinema2/Remove/Menu/{movie}	Aufruf erfolgt eigentlich nur via Frontend
Cinema2	removeReservation(String movie, String reservation)	/Cinema2/Remove/Reservation/{movie}	Aufruf erfolgt eigentlich nur via Frontend

Implementiert sind diese Mappings in

\PRDKE2019\cinema1\src\main\java\at\dkepr\cinemaservice\cinema1\api\Cinema1API.java

und

\PRDKE2019\cinema2\src\main\java\at\dkepr\cinemaservice\cinema2\api\Cinema2API.java

Dort ist auch der Aufruf der Sparql Verarbeitung ersichtlich.

Auf diese Funktionen wird via MetaServie zugegriffen. Auf das MetaService wird via Frontend zugegriffen.

Konfiguration MetaService

Das Metaservice ist konfigurierbar. Dies betrifft beispielsweise unter anderem die Funktionalität Menü's zu buchen, Name des Kinos oder die URL. Das Konfig File ist im Projektverzeichnis abrufbar unter

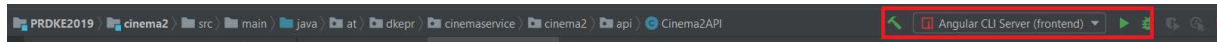
\PRDKE2019\app\src\main\resources\application.properties

```
starmovie.name= Starmovie
starmovie.url= http://localhost:8080/Cinema1
starmovie.menu=false
megaplex.name= Megaplex
megaplex.url= http://localhost:8080/Cinema2
megaplex.menu=true
```

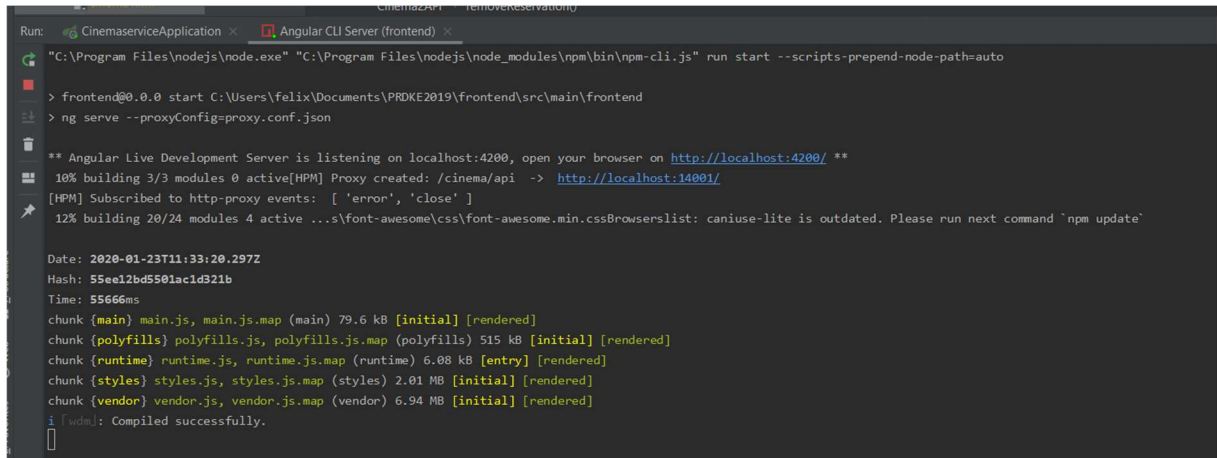
Frontend

Um dieses auf Angular basierte Frontend zu starten, wird im IntelliJ (während das Backend weiterhin läuft) in der Menüleiste der "Angular CLI Server (frontend)" gestartet.

Felix Baumgärtner, Andreas Eigner, Denny Graf, Mathias Hausleithner



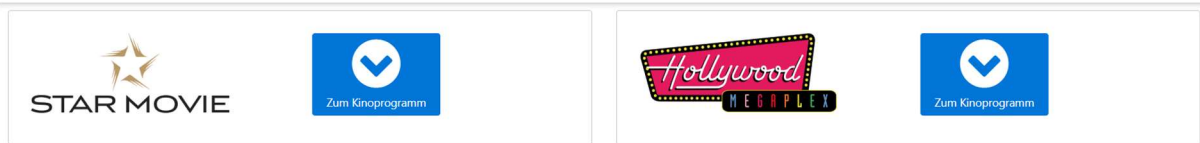
Via Terminal wird der Benutzer hingewiesen, wenn die Frontend Applikation erfolgreich gestartet ist.



Anschließend ist es via localhost und dem angegebenen Port (in diesem Fall 4200) aufrufbar.

<http://localhost:4200/cinemas>

Kinoservice



Das Cinemaservice wurde nun erfolgreich eingerichtet. Die Anwendung der Applikation ist im Dokument Screenshots_Frontend beschrieben.