

## Inhaltsverzeichnis

<b>1.</b>	<b>Allgemein .....</b>	<b>2</b>
1.1.	<i>Technologien .....</i>	2
1.2.	<i>Backend.....</i>	2
1.3.	<i>Installation und Inbetriebnahme.....</i>	3
<b>2.</b>	<b>Architektur .....</b>	<b>4</b>
2.1.	<i>Routen .....</i>	5
2.2.	<i>Kommunikation mit API.....</i>	5
2.3.	<i>Snackbar – Messages .....</i>	6
2.4.	<i>Keycloak .....</i>	7
<b>3.</b>	<b>Benutzeroberfläche .....</b>	<b>8</b>
3.1.	<i>Startseite .....</i>	8
3.2.	<i>Movie Details.....</i>	9
3.3.	<i>Admin – Menü.....</i>	10
3.4.	<i>Admin – Dashboard.....</i>	11
<b>4.</b>	<b>Besondere Komponenten .....</b>	<b>22</b>
4.1.	<i>Diagramme.....</i>	22
4.2.	<i>Corona Settings .....</i>	22
4.3.	<i>Id Validierung .....</i>	23
4.4.	<i>Messages – Snackbar .....</i>	23
4.5.	<i>Bild Upload.....</i>	24
4.6.	<i>Pipes .....</i>	24

## 1. Allgemein

### 1.1. Technologien

Dieses Projekt wurde mit den Technologien Angular, TypeScript, HTML und CSS umgesetzt. Eine ansprechende Benutzeroberfläche wurde mit Angular-Material erzielt. Zusätzlich zu dieser UI-Bibliothek wurden drei weitere Komponenten (mit „npm“) installiert. Dabei handelt es sich um File-Upload-, Zeitauswahl- sowie eine Diagramm-Komponenten.

### 1.2. Backend

Das Backend (SWK-Teil) wurde in .Net implementiert und stellt die folgenden Endpoints zur Verfügung (die folgende API-Dokumentation wurde mit „Swagger“ generiert):

The screenshot displays the Swagger API documentation for the Apollo Web application, organized into four main sections:

- Category**:
  - GET /apollo/api/category
  - POST /apollo/api/category
  - PUT /apollo/api/category
  - GET /apollo/api/category/{categoryId}
- CinemaHall**:
  - GET /apollo/api/cinemahall
  - POST /apollo/api/cinemahall
  - PUT /apollo/api/cinemahall
  - GET /apollo/api/cinemahall/{cinemaHallId}
  - GET /apollo/api/cinemahall/version/{cinemaHallVersionId}
  - GET /apollo/api/cinemahall/version/{cinemaHallVersionId}/seats
- Genre**:
  - GET /apollo/api/genre
  - POST /apollo/api/genre
  - PUT /apollo/api/genre
  - GET /apollo/api/genre/{genreId}
- Movie**:
  - GET /apollo/api/movie/{movieId}/cover
  - GET /apollo/api/movie/{movieId}/cover/hd
  - GET /apollo/api/movie/{movieId}
  - DELETE /apollo/api/movie/{movieId}
  - GET /apollo/api/movie
  - POST /apollo/api/movie

**PaymentStatistic**

- GET /apollo/api/paymentstatistic/genre
- GET /apollo/api/paymentstatistic/month/{year}
- GET /apollo/api/paymentstatistic/year
- GET /apollo/api/paymentstatistic/weekday

**Reservation**

- GET /apollo/api/reservation
- GET /apollo/api/reservation/paid/{reservationId}
- GET /apollo/api/reservation/{reservationId}

**Schedule**

- GET /apollo/api/schedule
- POST /apollo/api/schedule
- PUT /apollo/api/schedule
- DELETE /apollo/api/schedule
- GET /apollo/api/schedule/{scheduleId}
- GET /apollo/api/schedule/current
- GET /apollo/api/schedule/filtered

### 1.3. Installation und Inbetriebnahme

- Klonen der Projekte (SWK5 und WEA5)
- Starten der Datenbank im Docker (SWK – Skript ist vorhanden)
- Befüllen der Datenbank mit Testdaten (SWK - Skript ist vorhanden)
- Starten des .NET API-Projekts
- Installation der Abhängigkeiten der Angular-Applikation mit „npm install“
- Konfiguration der Konstanten (Port,...) in „environments/environment.ts“
- Auswahl der Authentifizierungsmethode:
  - o auth.config.ts KeyCloak oder IdentityServer -> beide Konfigurationen stehen zur Verfügung.
- Starten der Angular-Applikation mit „ng serve“. Optional kann der Port durch die Option „-- Port“ explizit angegeben werden.

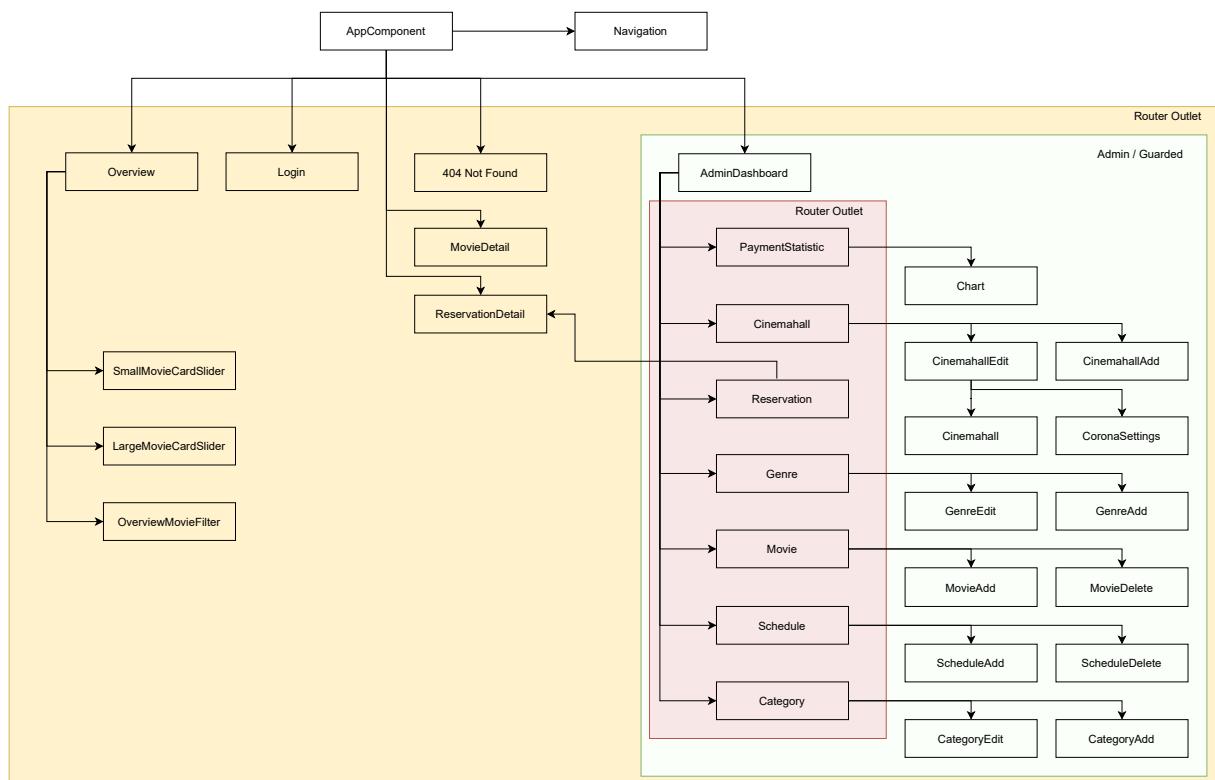
#### Hinweise:

- In der API-Implementierung wurden explizit Anfragen des Default-Port 4200 erlaubt. Sollte ein anderer Port verwendet werden, so muss der Browser ohne CORS – Sicherheitsregeln gestartet werden.
- Wird KeyCloak als Authentifizierungsmethode verwendet, so muss der Browser ohne CORS – Regeln gestartet werden. (KeyCloak wird über Port 8080 angesprochen)

## 2. Architektur

Wie bei Angular-Applikationen üblich, wurde die Trennung zwischen API-Schnittstelle (Service) und Code-Behind / UI strikt getrennt. Darüber hinaus wurde die Applikation auf mehrere Komponenten (> 40) aufgeteilt. Admin-Seiten werden durch einen Guard geschützt. Sollte der Benutzer nicht angemeldet sein, so wird dieser auf eine Login-Seite weitergeleitet.

Das nachstehende Diagramm zeigt den Aufbau der Angular-Applikation. Startpunkt der Applikation ist die Komponente „AppComponent“. Es wurden zwei Router-Outlets und ein Guard verwendet. Aus Gründen der Übersichtlichkeit wurde auf die Darstellung aller Komponenten verzichtet.



Alle Bearbeitungsformulare wurden in Form eines Dialogs (PopUp) implementiert. Die Daten werden über den Konstruktor weitergereicht bzw. beim Schließen des Dialogs wieder zurückgegeben.

## 2.1. Routen

Das folgende Listing zeigt die verwendeten Routen:

```
App-routing.module.ts
const routes: Routes = [
  { path: '', redirectTo: 'overview', pathMatch:'full' },
  {
    path: 'index.html',
    redirectTo: 'overview',
    pathMatch: 'full'
  },
  { path: 'overview', component: MovieOverviewComponent },
  { path: 'movies/:id', component: MovieDetailsComponent },
  { path: 'reservations/:id', component: ReservationDetailComponent },
  {
    path: 'admin',
    component: AdminDashboradComponent,
    canActivate: [NavigateToAdminGuard],
    children : [
      { path: '', redirectTo: 'statistic', pathMatch:'full' },
      { path: 'statistic', component: AdminPaymentStatisticsComponent },
      { path: 'cinemahall', component: AdminCinemaHallComponent },
      { path: 'reservation', component: AdminReservationComponent },
      { path: 'genre', component: AdminGenreComponent },
      { path: 'movie', component: AdminMovieComponent },
      { path: 'schedule', component: AdminScheduleComponent },
      { path: 'category', component: AdminCategoryComponent }
    ]
  },
  { path: 'login', component: AdminLoginComponent },
  { path: '**', component: PageNotFoundComponent }
];
```

## 2.2. Kommunikation mit API

In der Angular-Applikation stehen insgesamt 7 Services zur Verfügung. Die Funktionalität wurde wie im Backend nach Anwendungsbereichen sortiert.

- CategoryService
- CinemahallService
- GenreService
- MovieService
- PaymentStatisticService
- ReservationService
- ScheduleService

Für eine generische „Live“-ID-Validierung wurde zusätzlich ein Interface „BasisService“ implementiert. Dadurch kann ein beliebiger Service der Validierungsfunktion übergeben werden. Dieses Interface wird von den Services: CategoryService, CinemaHallService, MovieService, GenreService implementiert (da bei den zugehörigen Datensätzen der Benutzer die ID vergibt -> zb Kinosaalbezeichnung: „IMAX“). Das folgende Listing zeigt die Implementierung der Live-Async-ID-Validierung. Bei dieser Implementierung wird eine Debounce-Time von 500ms verwendet.

```
IdValidator.ts
export class IdExistsValidator {
  static createValidator(service: BasisService, time: number = 500): AsyncValidatorFn {
    return (control: AbstractControl): Observable<ValidationErrors> => {
      return timer(time).pipe(
        switchMap(() => service.getById_(control.value)),
        map(isTaken => (isTaken ? { idExists: true } : null)),
        catchError(() => of(null))
      );
    }
}
```

```
    };
}
```

### 2.3. Snackbar – Messages

Tritt ein Fehler (HTTP Status-Code) bei der Kommunikation mit dem Backend auf, so werden diese für den Benutzer sichtbar in einer Snackbar am unteren Bildschirmrand angezeigt. Die Snackbar wurde als „Injectable“ implementiert und bei Bedarf aufgerufen. Auch „Success“-Nachrichten werden mit Hilfe der Snackbar eingeblendet.

Für die Fehler- und Erfolgsbehandlung wurden ebenfalls generische Funktionen implementiert, die von allen Service-Methoden verwendet werden. Durch dieses Konzept erhält der Benutzer nach Aktionen wie „Create Genre“ Feedback. Die folgenden Listings zeigen einen Ausschnitt der Implementierungen:

#### Genre.service.ts (Ausschnitt)

```
create(genre: Genre): Observable<any> {
  return this.http.post<Genre>(`${environment.server}/genre`, genre)
    .pipe(
      tap(() => successHandler("genre created", this.snackBarMessage)),
      catchError((err => errorHandler(err, "genre", this.snackBarMessage)))
    );
}
```

#### Success.ts

```
import { SnackbarMessageService } from "../snackbar-message/snackbar-message.service";

export const successHandler = (message: string, snackBarMessage: SnackbarMessageService) =>
{
  snackBarMessage.openSnackBar(message, "snackbar-message-success");
}
```

#### Error.ts

```
import { Observable, of } from "rxjs";
import { SnackbarMessageService } from "../snackbar-message/snackbar-message.service";

export const errorHandler = (error: Error | any, message: string, snackBarMessage: SnackbarMessageService): Observable<any> => {
  if (error.status == 409)
    snackBarMessage.openSnackBar(`"${message} id conflict occurred`, "snackbar-message-error");
  else if (error.status == 404)
    snackBarMessage.openSnackBar(`"${message} not found`, "snackbar-message-error");
  else
    snackBarMessage.openSnackBar(`something went wrong :(`, "snackbar-message-error");
  console.log(error);
  return of(null);
}
```

#### Snackbar-message.service.ts (Ausschnitt)

```
openSnackBar(message: string, styleClass: string) {
  this.snackBar.open(message, null, {
    duration: 5000,
    panelClass: [styleClass]
  });
}
```

## 2.4. KeyCloak

Als Identity Server wurde, zusätzlich zu jenem aus der Übung, KeyCloak über Docker aufgesetzt. Der Docker-Container kann über den Port 8080 angesprochen bzw. konfiguriert werden. Die Konfiguration wurde mit Hilfe der offiziellen Dokumentation durchgeführt.

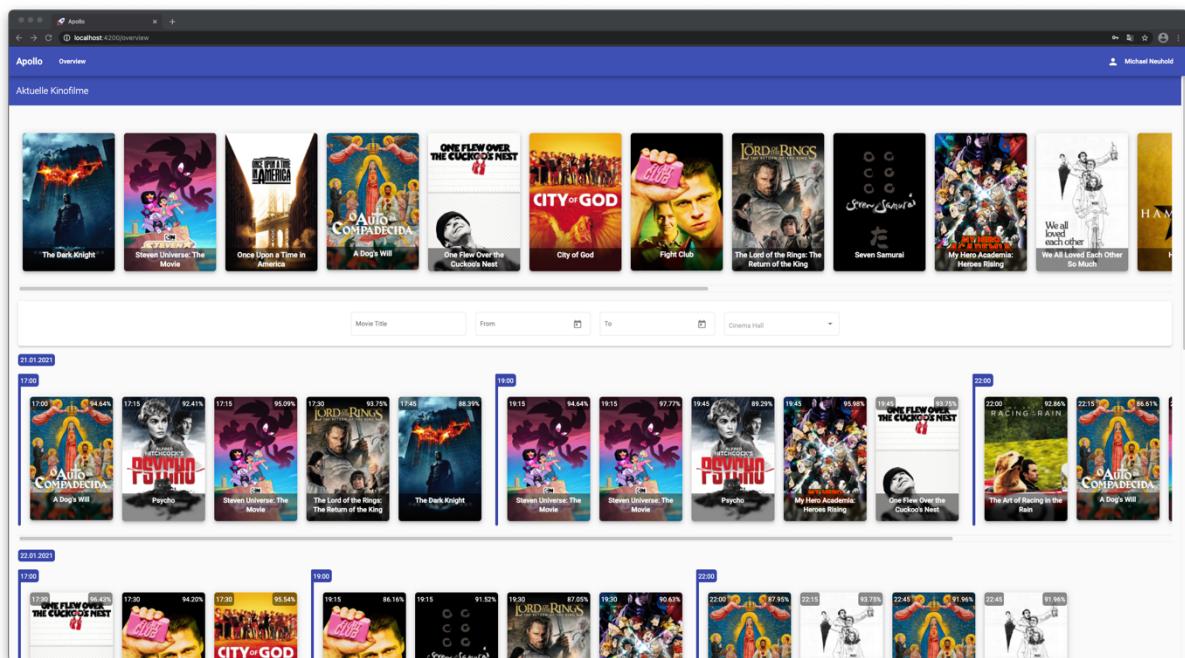
Es wurde ein Client „apollo-web“ und ein Benutzer „Michael Neuhold“ angelegt. Zusätzlich wurde noch für den Client der Implizite Flow erlaubt. Die entsprechenden Endpoints für den Login wurden in der Angular-Applikation im Authentifizierungs-Konfigurationsfile hinterlegt. Zusätzlich wurde der „SilentRefresh“ aktiviert, um den id\_token zu aktualisieren. Die Session endet nach 1h Inaktivität.

The screenshot shows the Keycloak Admin UI for the 'Apollo' realm. The left sidebar is dark-themed and includes sections for 'Configure' (Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication) and 'Manage' (Groups, Users, Sessions, Events, Import, Export). The 'Clients' section is currently selected. The main right panel shows the 'Clients > apollo-web' view. The 'Settings' tab is active, displaying configuration for the 'Apollo-web' client. The client ID is set to 'apollo-web'. The 'Name' field is empty. The 'Description' field is also empty. The 'Enabled' switch is set to 'ON'. The 'Always Display in Console' switch is set to 'OFF'. The 'Consent Required' switch is set to 'OFF'. The 'Login Theme' field is empty. The 'Client Protocol' is set to 'openid-connect'. The 'Access Type' is set to 'public'. The 'Standard Flow Enabled' switch is set to 'ON'. The 'Implicit Flow Enabled' switch is set to 'ON'. The 'Direct Access Grants Enabled' switch is set to 'OFF'.

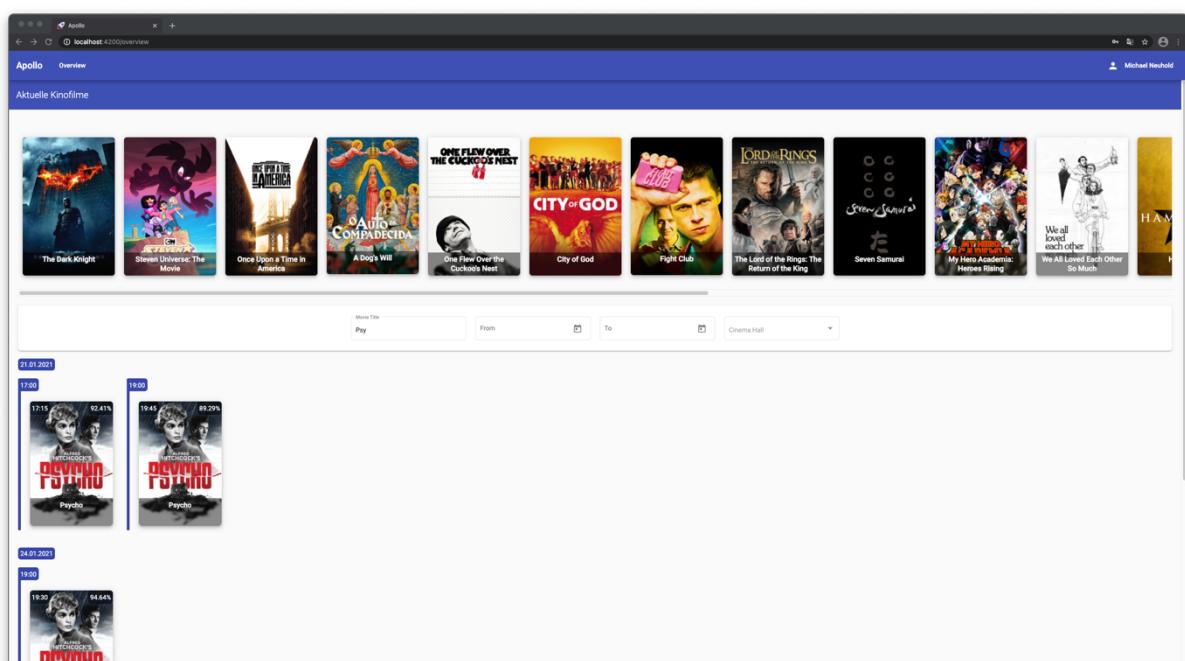
### 3. Benutzeroberfläche

#### 3.1. Startseite

Die nachstehenden Screenshots zeigen die Filmübersichtsseite mit Suche nach Film-Titel, Datum der Aufführung und Kinosaal. Bei diesen Screenshots sieht man zu dem, dass ein Admin (rechts oben) angemeldet ist. In der ersten Reihe werden die aktuellsten Filme (nicht redundant) angezeigt. Alle Film-Reihen unterhalb der Filtereingabe zeigen eine gruppierte Auflistung nach Aufführungszeit.



Der folgende Screenshot zeigt die Übersicht mit angewandter Filmsuche nach „Psy“.



Der nachstehende Screenshot zeigt die Übersicht mit angewandter Datumssuche (auf einen Tag beschränkt).

The screenshot shows the Apollo web application's movie overview page. At the top, there is a search bar with fields for 'Movie Title', 'From' (set to 1/22/2021), 'To' (set to 1/23/2021), and 'Cinema Hall'. Below the search bar is a grid of movie posters for various films like 'The Dark Knight', 'Steven Universe: The Movie', 'Once Upon a Time in America', etc. The main content area displays a grid of movie showtimes for the selected date range. Each row represents a specific movie, and each column represents a different screening time. The grid includes movie titles like 'One Flew Over the Cuckoo's Nest', 'City of God', 'Fight Club', 'Seven Samurai', 'The Lord of the Rings: The Return of the King', 'My Hero Academic: Heroes Rising', and 'A Dog's Will'. Each entry in the grid also includes a small thumbnail of the movie poster.

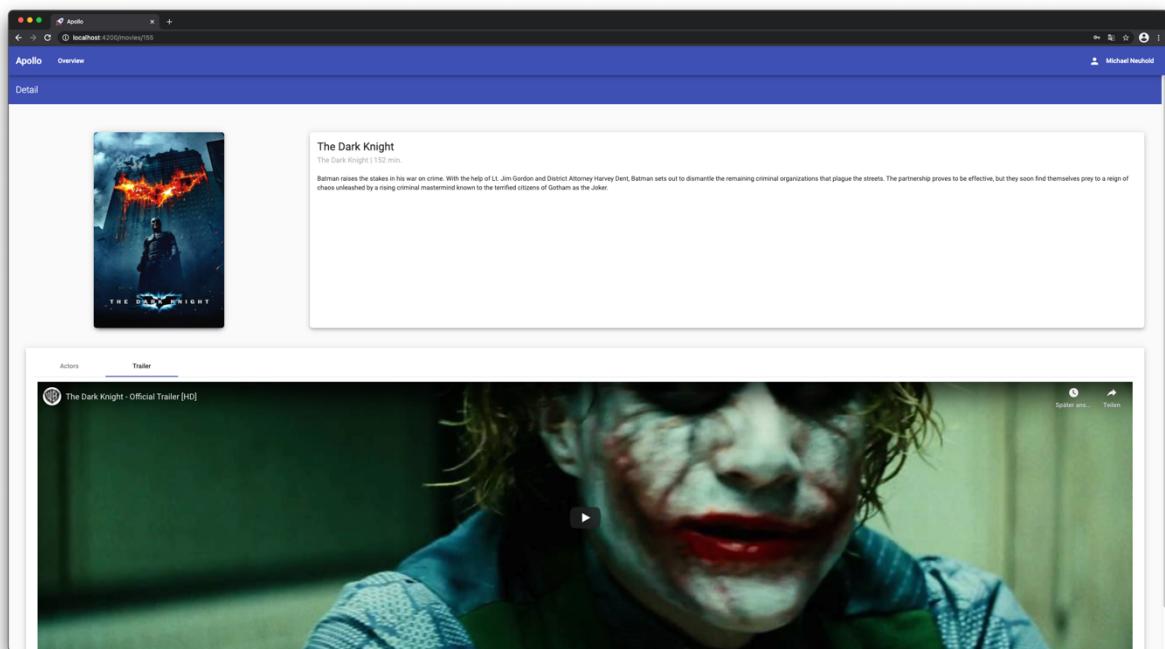
### 3.2. Movie Details

The screenshot shows the Apollo web application's movie detail page for 'The Dark Knight'. The top section features the movie's poster and the title 'The Dark Knight | 132 min.'. A brief plot summary follows: 'Batman raises the stakes in his war on crime. With the help of Lt. Jim Gordon and District Attorney Harvey Dent, Batman sets out to dismantle the remaining criminal organizations that plague the streets. The partnership proves to be effective, but they soon find themselves prey to a reign of chaos unleashed by a rising criminal mastermind known to the terrified citizens of Gotham as the Joker.' Below the plot summary, there are tabs for 'Actors' and 'Trailer'. The 'Actors' tab lists numerous cast members and their roles, including Heath Ledger (Joker), Michael Caine (Alfred Pennyworth), Gary Oldman (Commissioner James Gordon), Aaron Eckhart (Harvey Dent / Two-Face), Maggie Gyllenhaal (Rachel Dawes), Morgan Freeman (Lucius Fox), Nestor Carbonell (Major Anthony Garcia), Monique Gabriela Curnen (Arona Ramirez), and many others. The 'Trailer' tab is currently inactive.

#### Anmerkung:

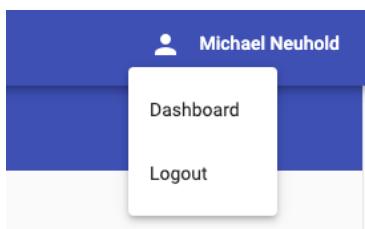
Für performantes Laden der Seiten wurden zwei verschiedene Bilder bzw. Bild-Endpoints verwendet. Bei der Filmübersichtsseite werden Bilder mit geringerer Auflösung angezeigt und auf der Movie Detail Seite werden Bilder mit HD-Auflösung angezeigt.

Es kann zwischen den Anzeigen „Schauspieler“ und „Trailer“ gewechselt werden. Der Trailer wurde durch ein „IFrame“ eingebunden.



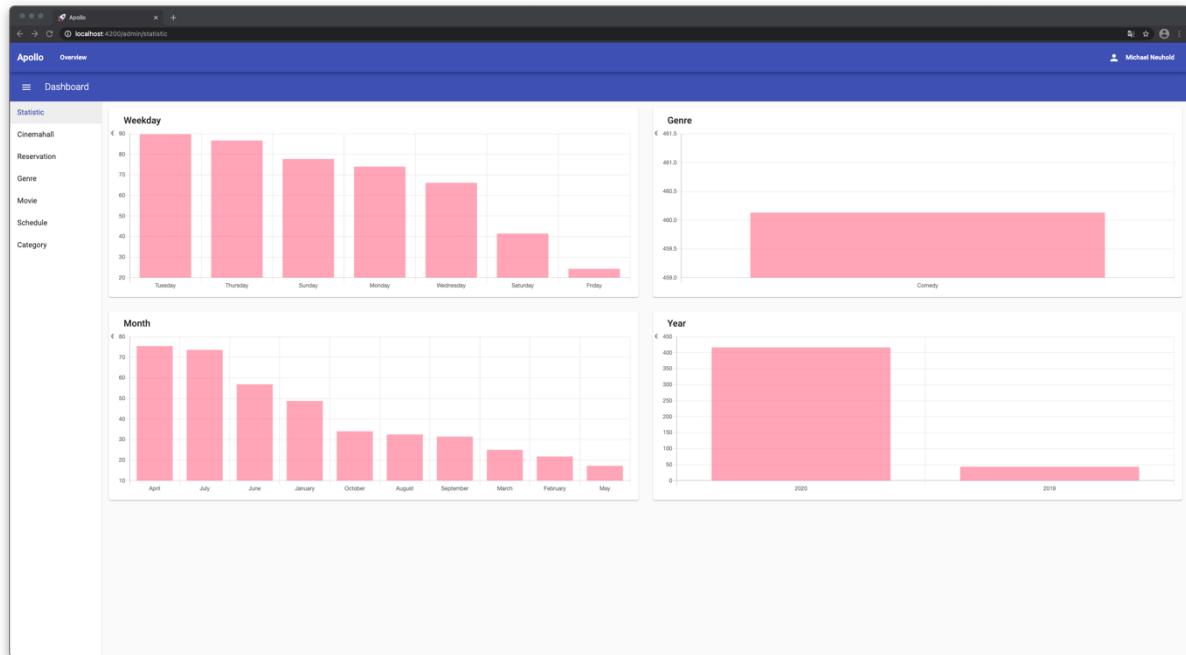
### 3.3. Admin – Menü

Ist man angemeldet, so kann man durch Klick auf den Benutzernamen auf das Admin-Dashboard wechseln oder ein „Logout“ durchführen. Der Benutzername wird von den „IdentityClaims“ ausgelesen und angezeigt.

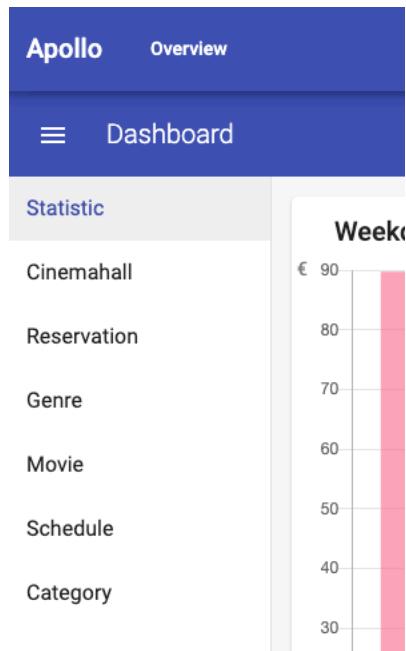


### 3.4. Admin – Dashboard

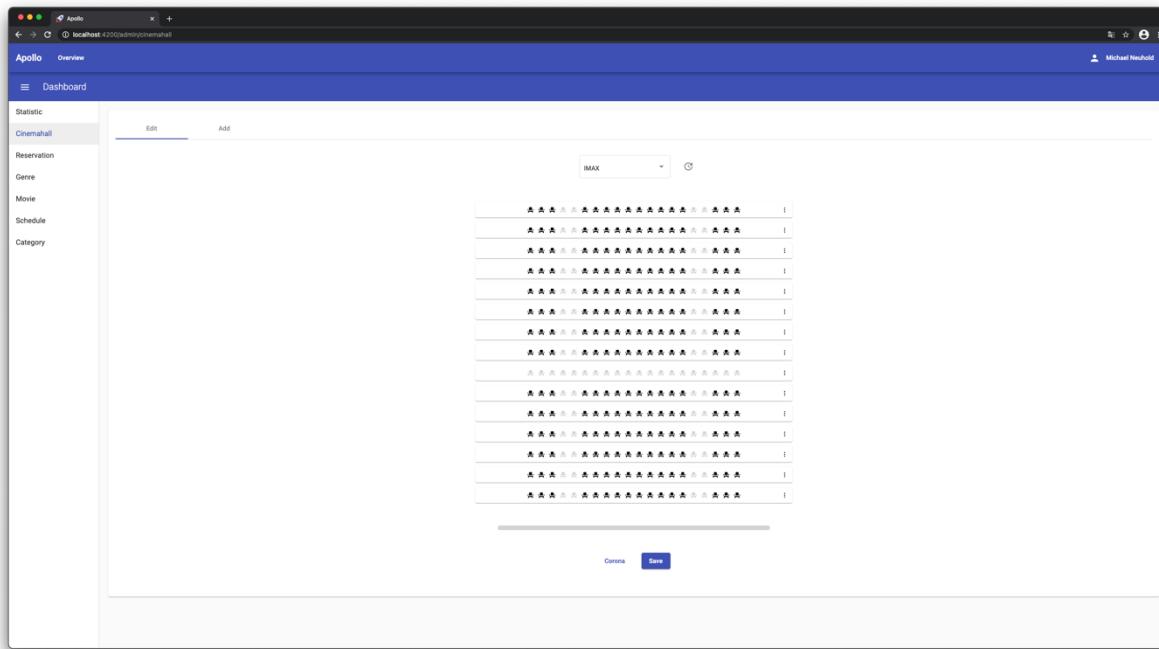
Der nachstehende Screenshot zeigt einen Überblick über das Admin-Dashboard. Das Backend unterstützt Auswertungen über Bezahlte Reservierungen. Aus diesem Grund wurden Diagramme zur Visualisierung eingebunden.



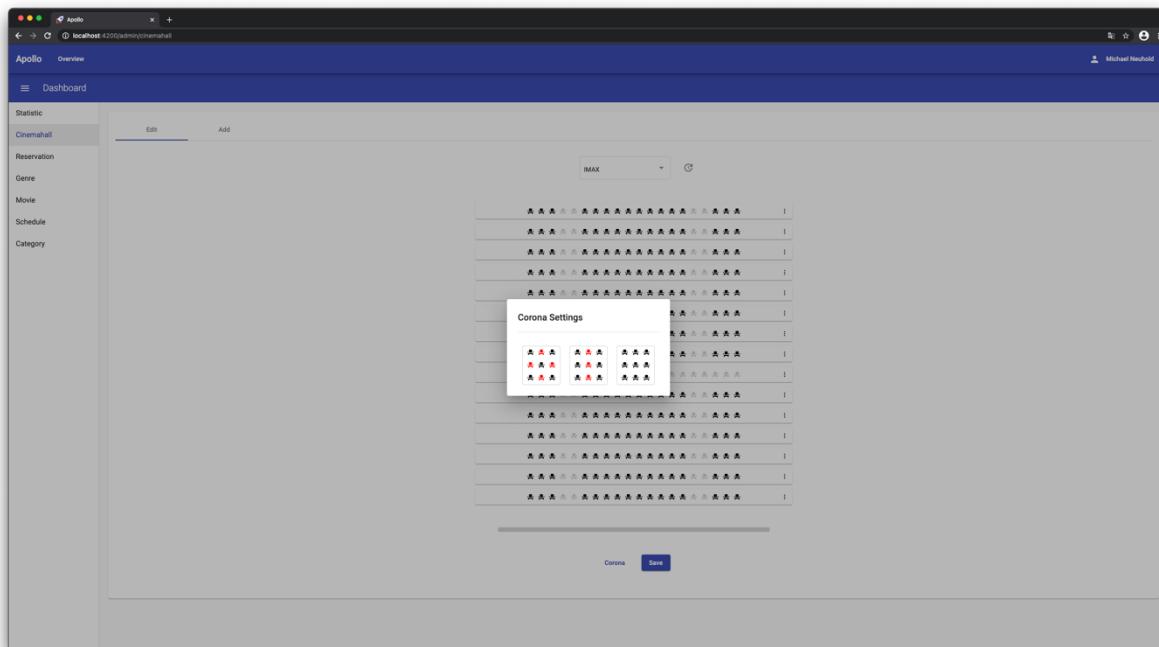
Konkret können folgenden Punkte über das Admin-Dashboard gepflegt werden:



Die nachstehenden Screenshots zeigen die Funktionalitäten für „CinemaHall“.



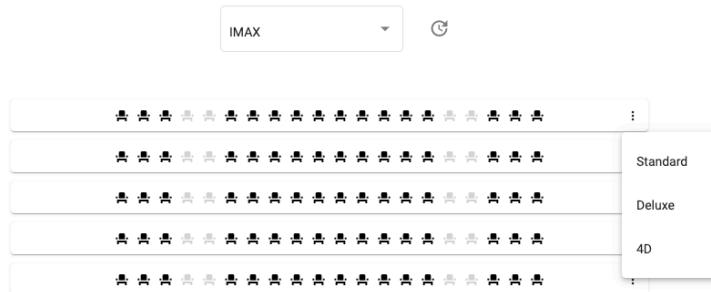
Der nachstehende Screenshot zeigt die Corona-Einstellungen für das Speeren von Sitzplätzen nach einem bestimmten Muster. Bereits Reservierte Sitze sind nicht betroffen, da die Cinemahalls in der Datenbank versioniert werden.



Sitze können auch manuell gesperrt (corona) oder als Gang (not available / grau) markiert werden.



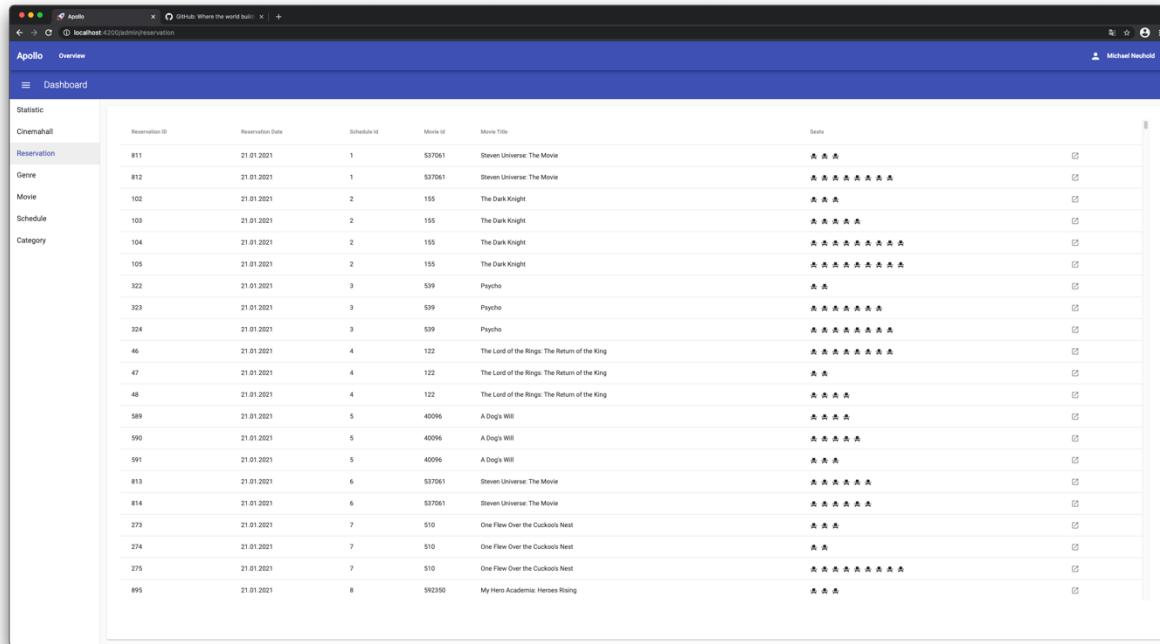
Wird der Button auf für eine Sitzreihe betätigt, so kann die Kategorie der Sitzreihe geändert werden.



Wird auf den Tab „Add“ gewechselt, so kann ein neuer Kinosaal hinzugefügt werden. Dabei kann die „Standardgröße (15x20)“ oder eine neue Größe ausgewählt werden. Hier wird der Name des Kinosaals als Schlüssel verwendet. Aus diesem Grund wird während der Eingabe mit dem zuvor angesprochenen Async-Validator überprüft, ob die ID (Saal 10) noch verfügbar ist.

Name	Name Saal 10 <small>valid</small>	Name IMAX <small>cinemahall name already exists</small>
Rows	Rows 15	Rows <small>row number is required</small>
Seats	Seats 20	Seats <small>seats number is required</small>
<input type="checkbox"/> use default size	<input checked="" type="checkbox"/> use default size	<input type="checkbox"/> use default size
<button>Save</button>	<button>Save</button>	<button>Save</button>

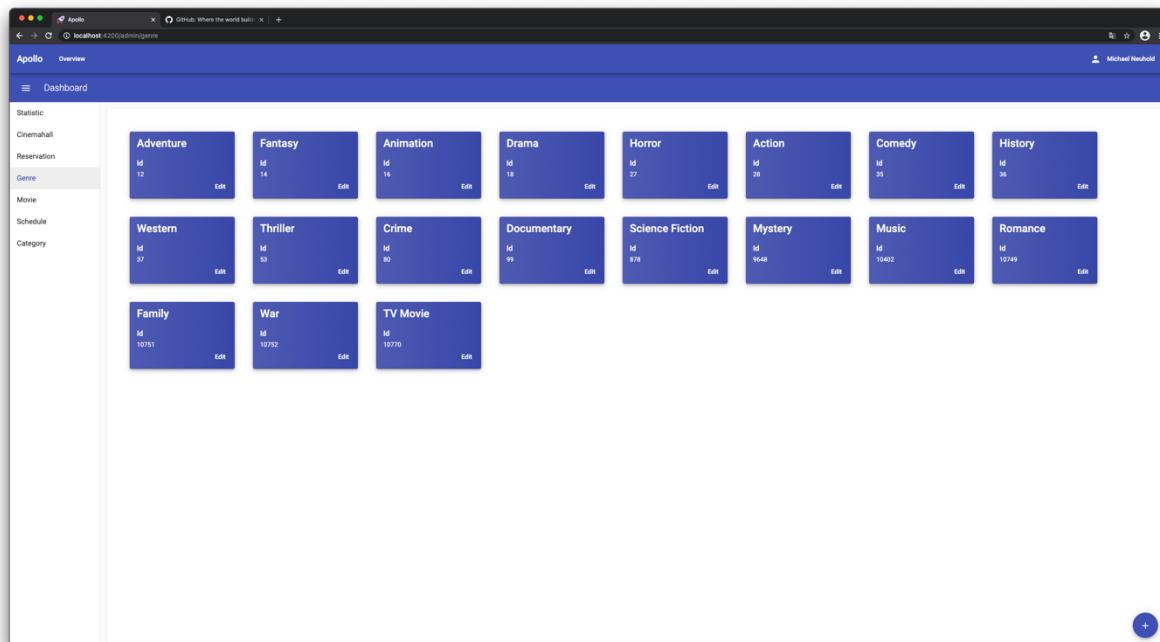
Der nachstehende Screenshot zeigt eine tabellarische Auflistung der bereits getätigten Reservierungen.



The screenshot shows a table titled "Reservations" with the following columns: Reservation ID, Reservation Date, Schedule ID, Movie ID, Movie Title, and Seats. The data includes various movie titles like "Steven Universe: The Movie", "The Dark Knight", and "My Hero Academia: Heroes Rising". Each row has an edit icon on the right.

	Reservation ID	Reservation Date	Schedule ID	Movie ID	Movie Title	Seats	
Cinemall	811	21.01.2021	1	537061	Steven Universe: The Movie	▲ ▲ ▲	
Reservation	812	21.01.2021	1	537061	Steven Universe: The Movie	▲ ▲ ▲ ▲ ▲ ▲ ▲	
Genre	812	21.01.2021	2	155	The Dark Knight	▲ ▲ ▲	
Movie	102	21.01.2021	2	155	The Dark Knight	▲ ▲ ▲	
Schedule	103	21.01.2021	2	155	The Dark Knight	▲ ▲ ▲	
Category	104	21.01.2021	2	155	The Dark Knight	▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲	
	105	21.01.2021	2	155	The Dark Knight	▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲	
	322	21.01.2021	3	539	Psycho	▲ ▲	
	923	21.01.2021	3	539	Psycho	▲ ▲ ▲ ▲ ▲ ▲	
	324	21.01.2021	3	539	Psycho	▲ ▲ ▲ ▲ ▲ ▲ ▲	
	46	21.01.2021	4	122	The Lord of the Rings: The Return of the King	▲ ▲ ▲ ▲ ▲ ▲ ▲	
	47	21.01.2021	4	122	The Lord of the Rings: The Return of the King	▲ ▲	
	48	21.01.2021	4	122	The Lord of the Rings: The Return of the King	▲ ▲ ▲ ▲	
	589	21.01.2021	5	40096	A Dog's Will	▲ ▲ ▲	
	590	21.01.2021	5	40096	A Dog's Will	▲ ▲ ▲ ▲	
	591	21.01.2021	5	40096	A Dog's Will	▲ ▲ ▲	
	813	21.01.2021	6	537061	Steven Universe: The Movie	▲ ▲ ▲ ▲ ▲ ▲	
	814	21.01.2021	6	537061	Steven Universe: The Movie	▲ ▲ ▲ ▲ ▲ ▲	
	273	21.01.2021	7	510	One Flew Over the Cuckoo's Nest	▲ ▲	
	274	21.01.2021	7	510	One Flew Over the Cuckoo's Nest	▲ ▲	
	275	21.01.2021	7	510	One Flew Over the Cuckoo's Nest	▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲	
	895	21.01.2021	8	592350	My Hero Academia: Heroes Rising	▲ ▲ ▲	

Der nachstehende Screenshot zeigt alle verfügbaren Genre. Mit Klick auf Edit kann die Bezeichnung geändert werden. Mit Klick auf den „Hinzufügen“-Button (rechts unten) können neue Genre hinzugefügt werden.



The screenshot shows a grid of genre names, each in a blue box with an "Edit" button. The genres listed are Adventure, Fantasy, Animation, Drama, Horror, Action, Comedy, History, Western, Thriller, Crime, Documentary, Science Fiction, Mystery, Music, and Romance. A plus sign button is located in the bottom right corner of the grid.

Adventure	Fantasy	Animation	Drama	Horror	Action	Comedy	History
Western	Thriller	Crime	Documentary	Science Fiction	Mystery	Music	Romance
Family	War	TV Movie					

The screenshot shows the Apollo Web application's 'Genre' management interface. On the left, a sidebar menu includes 'Statistic', 'Cinemahall', 'Reservation', 'Genre' (which is selected), 'Movie', 'Schedule', and 'Category'. The main area displays a grid of genre cards. A card for 'Family' (Id: 10751) has an 'Edit' button highlighted. A modal dialog titled 'Edit genre' is open over the grid, containing a single input field with the value 'Family' and a 'Save' button.

This screenshot shows the same Apollo Web application interface, but the 'Genre' card for 'Family' (Id: 10751) now has an 'Add' button highlighted. A modal dialog titled 'Add genre' is open, containing two input fields: one with the value '24224' and another with the placeholder 'Documentation WEA5'. A 'Add' button is at the bottom of this dialog.

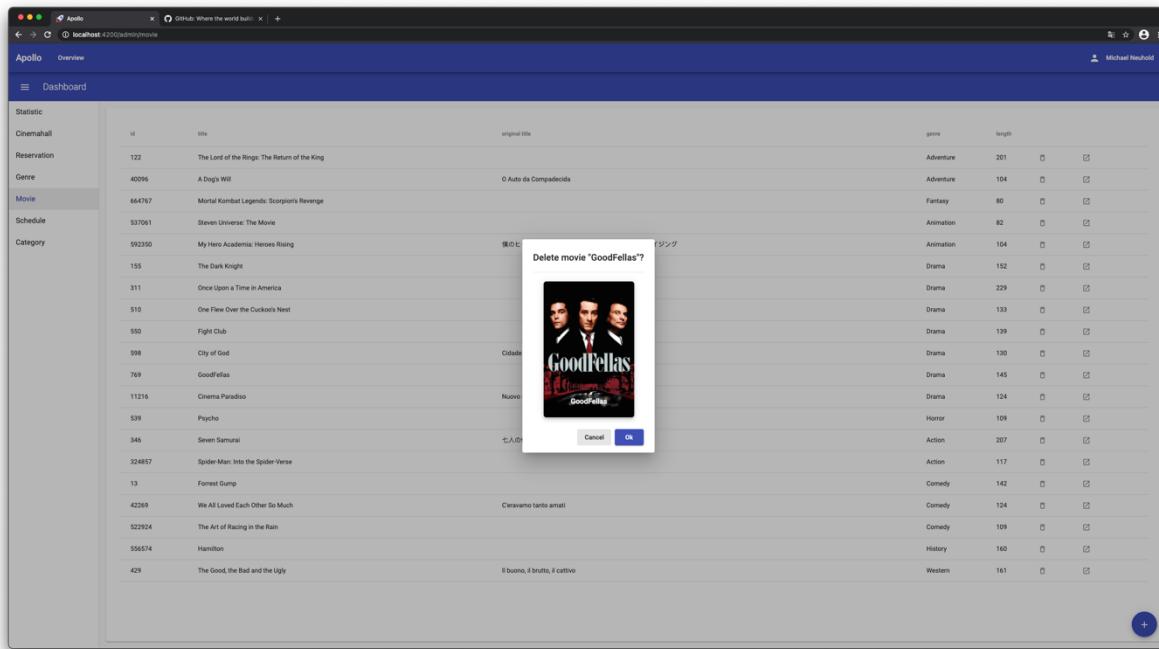
Der folgende Screenshot zeigt eine Auflistung aller Verfügbaren Filme. Filme können hinzugefügt und gelöscht werden. Durch eine Verlinkung gelangt man auf die Movie – Detail Seite.

The screenshot shows the Apollo Admin interface with the title "Apollo" at the top. On the left, there is a sidebar with navigation links: Statistic, Cinemall, Reservation, Genre, Movie (which is highlighted in blue), Schedule, and Category. The main area displays a table of movies with columns: id, title, original title, genre, and length. The table contains numerous movie entries, such as "The Lord of the Rings: The Return of the King", "A Dog's Way Home", "Mortal Kombat Legends: Scorpion's Revenge", etc. A blue "+" button is located in the bottom right corner of the table area.

Der nachstehende Screenshot zeigt das Formular zum Anlegen eines neuen Films. Es können zwei Bilder durch einen Fileupload hinzugefügt werden. Diese werden „base64“-formatiert an das Backend übertragen und in der Datenbank gespeichert.

This screenshot shows the "Add movie" form overlaid on the movie list. The form fields include: movie id\* (12424), movie title\* (Titel des neuen Films), movie original title\* (Titel des neuen Films), and a description field containing placeholder text "Hier könnte eine Beschreibung stehen". Below these fields is a "cast" section with a dropdown menu showing actors: Michael, Julian, Max, and an option to "enter new actor...". There is also a "trailer (optional)" input field. At the bottom of the form, there are "Cancel" and "Add" buttons. In the background, the movie list is visible with the same entries as the first screenshot.

Der nachstehende Screenshot zeigt den Dialog zum Löschen eines Films.

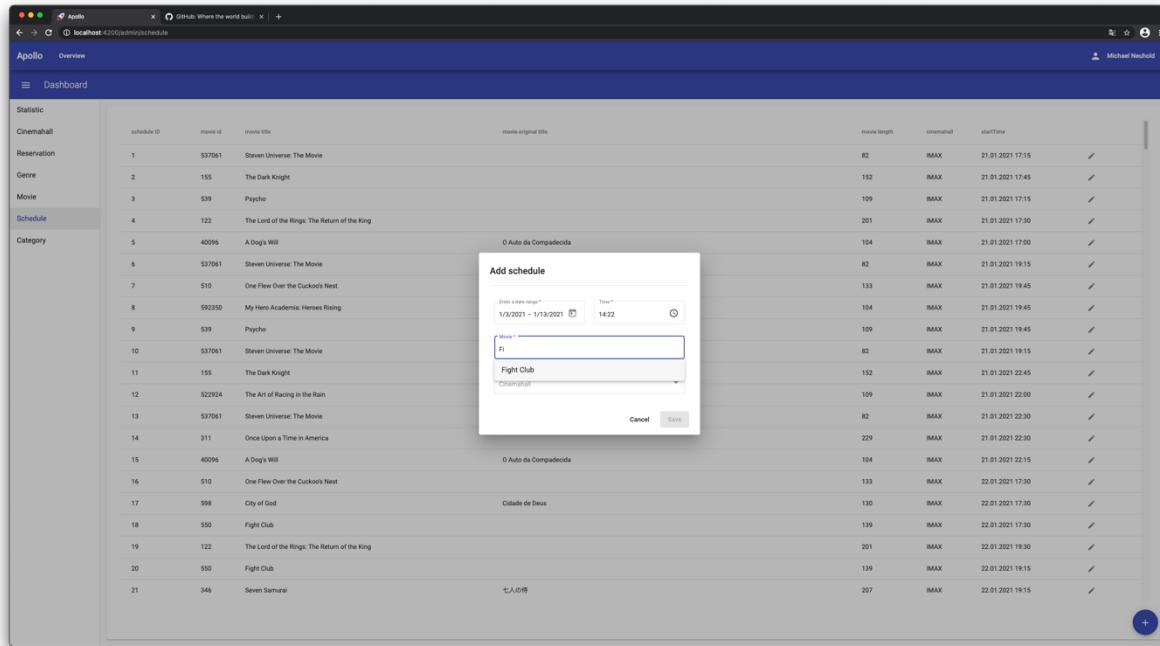


Der nachstehende Screenshot zeigt die Übersicht über alle angelegten Filmvorstellungen. Filmvorstellungen können bearbeitet und angelegt werden.

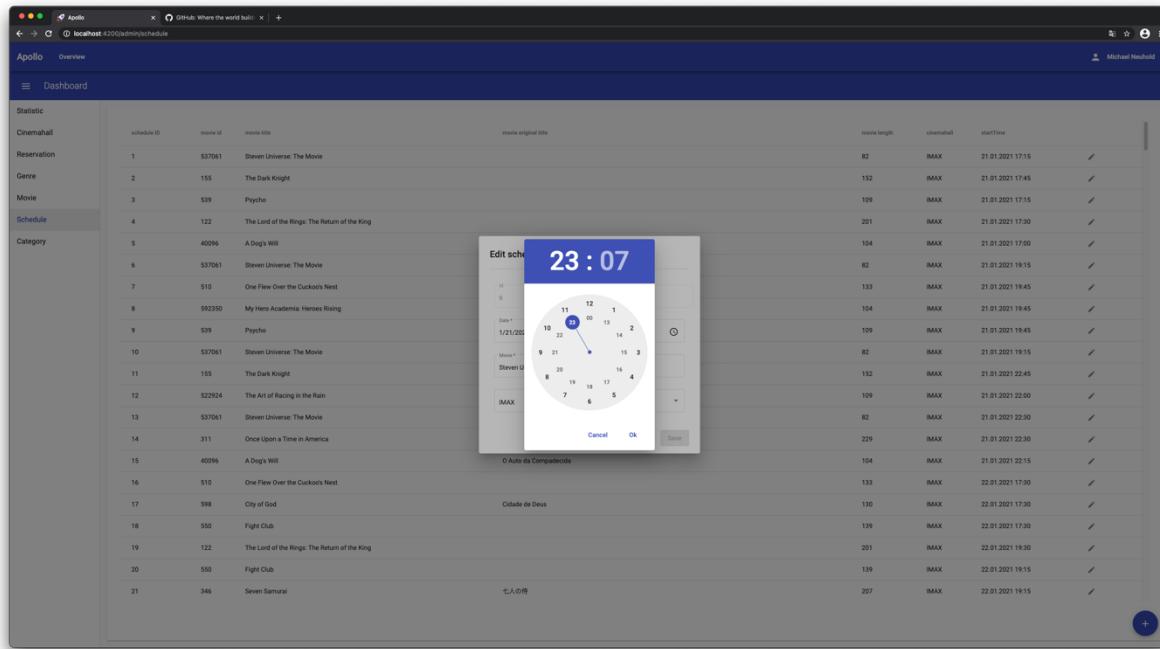
The screenshot shows a table of scheduled movie showings on the left. The table has columns for schedule ID, movie ID, movie title, movie original title, movie length, cinemahall, and startTime. Each row also has an edit icon. The movie 'GoodFellas' is listed multiple times across different dates and times.

	schedule ID	movie ID	movie title	movie original title	movie length	cinemahall	startTime
Cinemahall	1	537061	Steven Universe: The Movie		82	IMAX	21.01.2021 17:15
Reservation	2	155	The Dark Knight		152	IMAX	21.01.2021 17:45
Genre	3	539	Psycho		109	IMAX	21.01.2021 17:15
Movie	4	122	The Lord of the Rings: The Return of the King		201	IMAX	21.01.2021 17:30
Schedule	5	40996	A Dog's Will	O Auto da Compadecida	104	IMAX	21.01.2021 17:00
Category	6	537061	Steven Universe: The Movie		82	IMAX	21.01.2021 17:15
	7	510	One Flew Over the Cuckoo's Nest		133	IMAX	21.01.2021 19:45
	8	592350	My Hero Academia: Heroes Rising	僕のヒーローアカデミア THE MOVIE ヒーローズ・ライジング	104	IMAX	21.01.2021 19:45
	9	539	Psycho		109	IMAX	21.01.2021 19:45
	10	537061	Steven Universe: The Movie		82	IMAX	21.01.2021 19:15
	11	155	The Dark Knight		152	IMAX	21.01.2021 22:45
	12	522924	The Art of Racing in the Rain		109	IMAX	21.01.2021 22:00
	13	537061	Steven Universe: The Movie		82	IMAX	21.01.2021 22:30
	14	311	Once Upon a Time in America		229	IMAX	21.01.2021 22:30
	15	40996	A Dog's Will	O Auto da Compadecida	104	IMAX	21.01.2021 22:15
	16	510	One Flew Over the Cuckoo's Nest		133	IMAX	22.01.2021 17:30
	17	598	City of God	Cidade de Deus	130	IMAX	22.01.2021 17:30
	18	550	Fight Club		139	IMAX	22.01.2021 17:30
	19	122	The Lord of the Rings: The Return of the King		201	IMAX	22.01.2021 19:30
	20	550	Fight Club		139	IMAX	22.01.2021 19:15
	21	346	Seven Samurai	七人の侍	207	IMAX	22.01.2021 19:15

Der folgende Screenshot zeigt das Anlegen einer neuen Vorstellung. Es kann eine Datumsspanne, eine Uhrzeit, ein Film sowie ein Kinosaal gewählt werden. Bei der Auswahl des Films unterstützt die Applikation durch eine Autovervollständigung.



Der nachstehende Screenshot zeigt das Ändern der Uhrzeit einer Vorstellung. Hier wird die zusätzlich installierte Time-Picker Komponente verwendet.



Der nachstehende Screenshot zeigt die Übersicht der Sitzkategorien.

The screenshot shows the Apollo Web application's interface. On the left, there is a sidebar with a navigation menu containing links: Statistic, Cinemahall, Reservation, Genre, Movie, Schedule, and Category. The 'Category' link is highlighted. The main content area displays three categories of seats in blue boxes:

- Standard**  
Description: Standard Sitz mit wenig Beinfreiheit  
Price: 10 €  
Edit
- Deluxe**  
Description: Deluxe Sitz mit viel Beinfreiheit und Tisch  
Price: 12 €  
Edit
- 4D**  
Description: Ganz neues Kinoerlebnis...  
Price: 14 €  
Edit

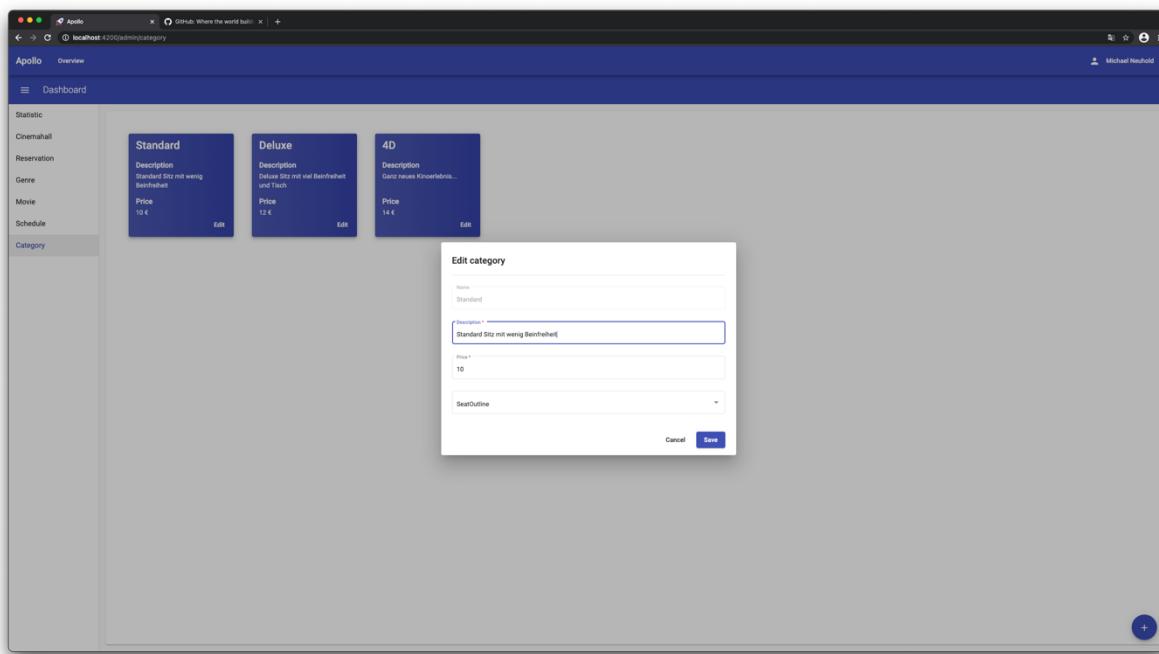
A large blue circular button with a '+' sign is located in the bottom right corner of the main content area.

Die folgenden Screenshots zeigen das Hinzufügen und Ändern der Kategorien.

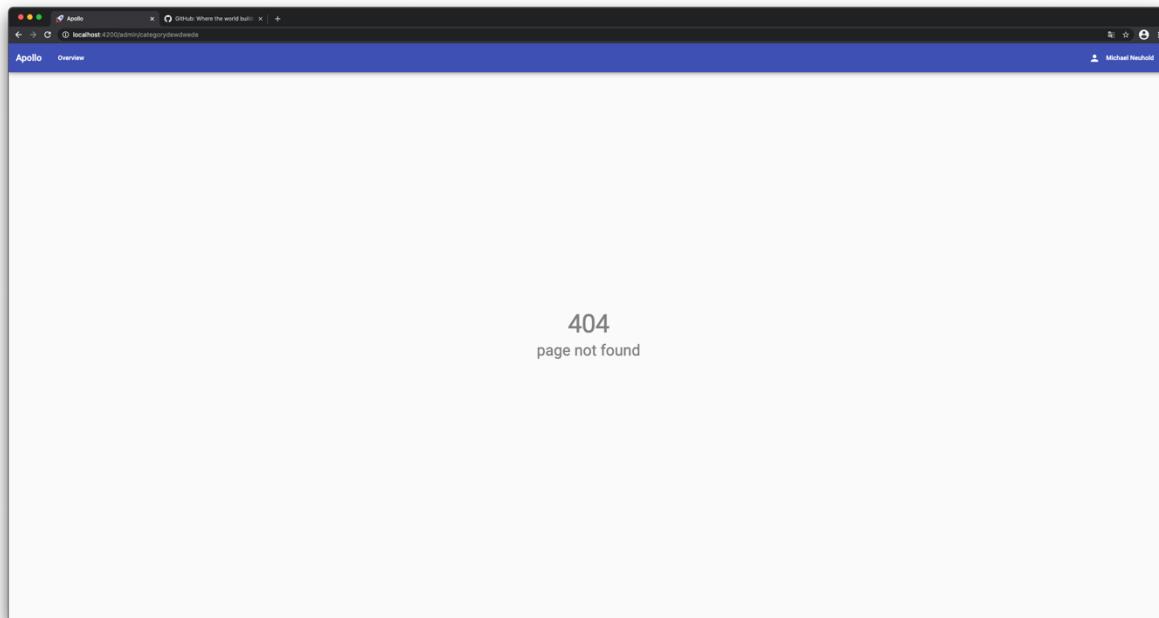
This screenshot shows the same Apollo Web application interface as the previous one, but with a modal dialog box overlaid on the screen. The dialog is titled 'Add category' and contains the following fields:

- Name\*:
- Bezeichnung\*:
- Preis\*:
- Icon:

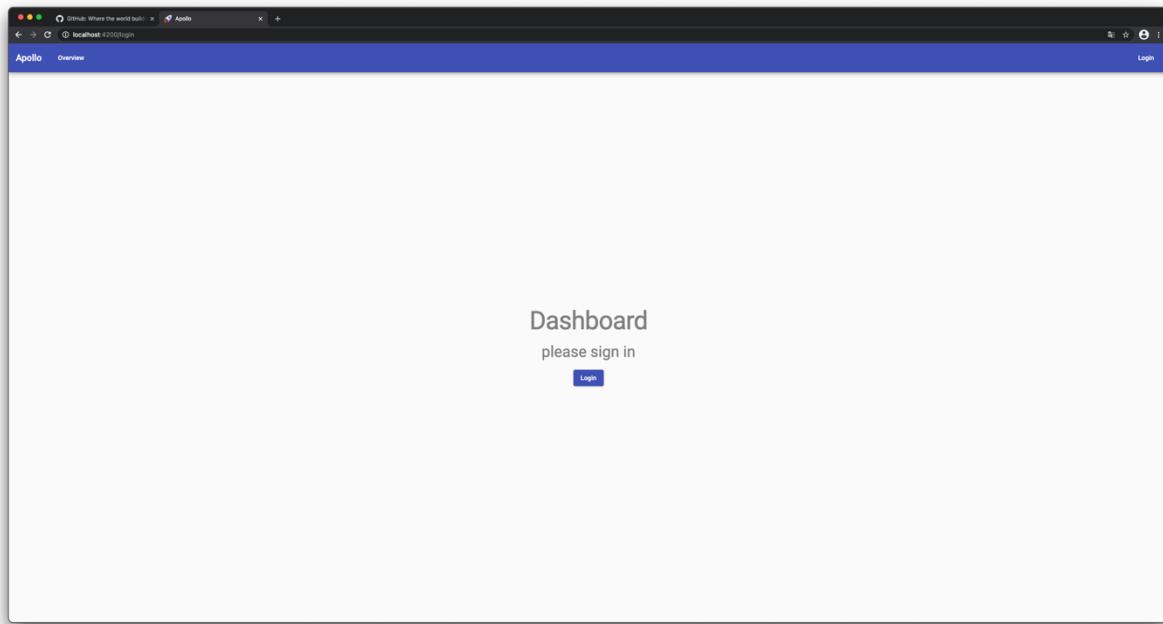
At the bottom right of the dialog box are two buttons: 'Cancel' and 'Add'. The background of the application is dimmed to indicate that the dialog is active.



Wird eine unbekannte URL eingegeben, so wird eine 404 – Error Message / Komponente angezeigt.



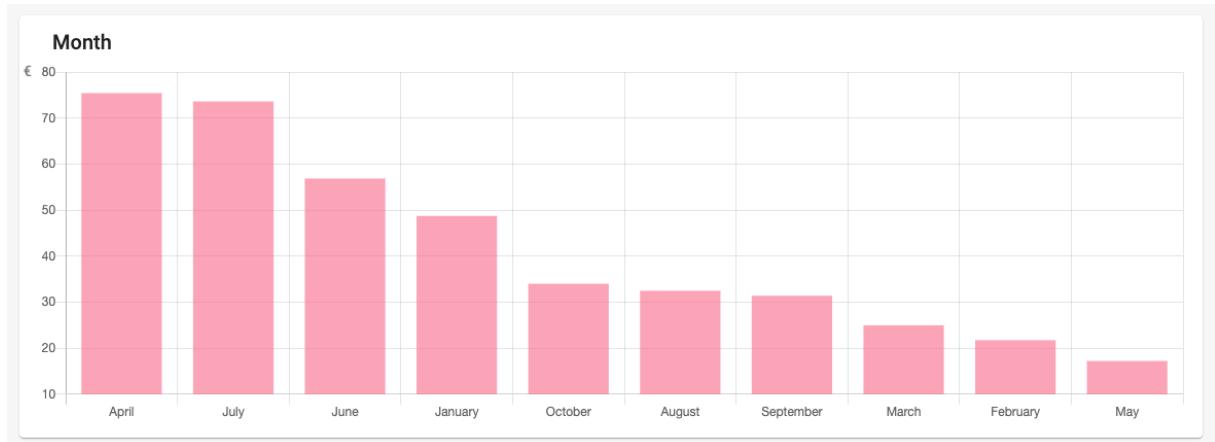
Wird zur Admin-URL navigiert, ohne Anmeldung, so wird die folgende Komponente angezeigt.



## 4. Besondere Komponenten

### 4.1. Diagramme

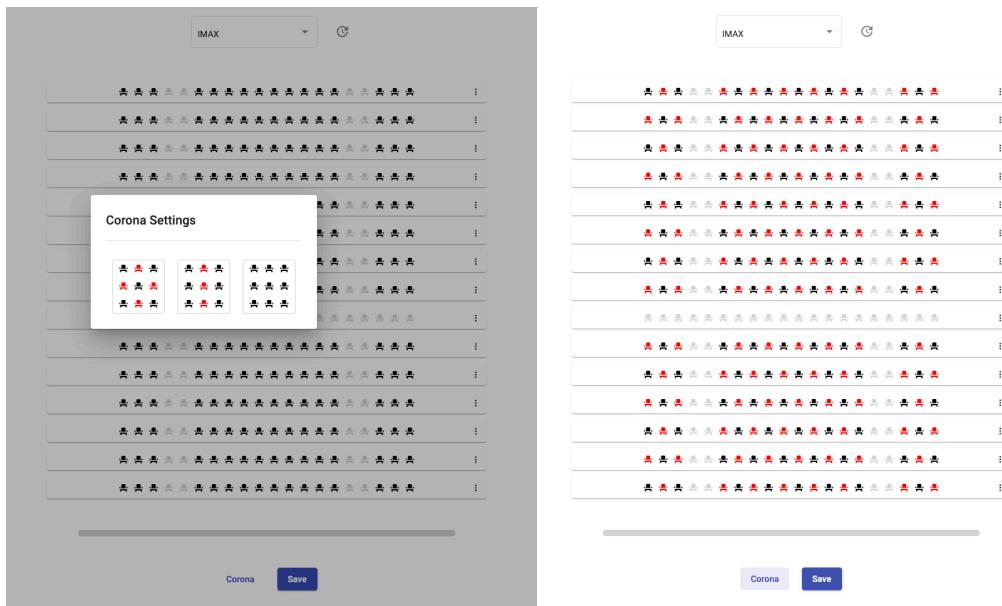
Die Diagramme wurden mit Hilfe der Bibliothek „ng2-charts“ umgesetzt. Die Implementierung wurde durch die API bereits vorgesehen. Es wurde eine Chart-Komponente entwickelt, die mehrmals mit unterschiedlichen Daten angezeigt wird.



### 4.2. Corona Settings

Es wurden drei Corona-Modi vorgesehen, die über das Admin-Dashboard auf einen Kinosaal angewandt werden können. Es können die folgenden Modi gewählt werden:

- Chessboard (siehe Bild)
- Alternately
- No-Restrictions



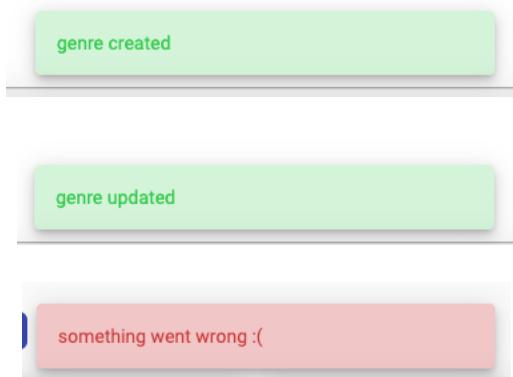
#### 4.3. Id Validierung

Durch die ID-Validierung kann bereits vor dem Absenden des Formulars die Verfügbarkeit einer ID überprüft werden. Wie bereits zuvor erläutert wurde dafür ein Interface implementiert, wodurch eine generische Weitergabe der Services an den Validator ermöglicht wurde.

The screenshot shows a mobile application interface with a header bar displaying '99' and '878'. Below it is a modal dialog titled 'Add genre'. Inside the dialog, there are two input fields. The first field, labeled 'Id \*', contains the value '12' and has a red border, indicating an error. A red error message 'genre id already exists' is displayed below this field. The second field, labeled 'Name \*', is empty and has a blue border. At the bottom of the dialog are two buttons: 'Cancel' and 'Add'.

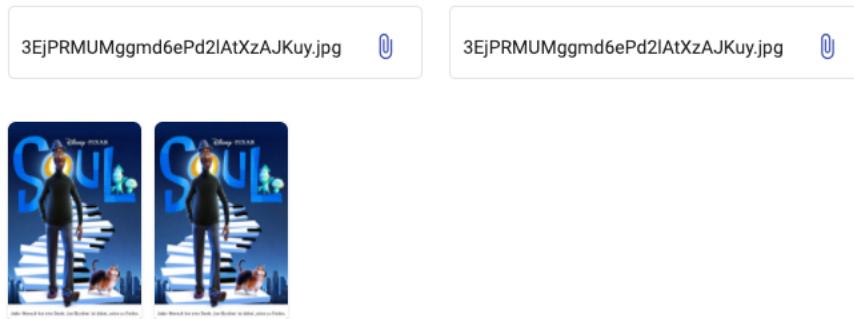
#### 4.4. Messages – Snackbar

Alle Service Methoden verwenden die zuvor gezeigten Fehler- bzw. Erfolgsfunktionen. Durch diese Funktionen werden die jeweiligen Snackbar-Nachrichten ausgegeben. Die nachstehenden Screenshots zeigen einige Beispiele:



#### 4.5. Bild Upload

Beim Einfügen von neuen Filmen wird die Möglichkeit geboten, Bilder zu laden. Nach Auswahl durch die File-Upload-Komponente werden die Bilder angezeigt. Durch das Absenden des Formulars werden die Bilder als „base64“ formatierter String an das Backend übertragen und dort gespeichert.



#### 4.6. Pipes

Es wurden einige Pipes für diverse Umwandlungen implementiert. Die folgenden zwei Listings zeigen Beispiele.

##### image-endpoint.pipe.ts

```
@Pipe({
  name: 'imageEndpoint'
})
export class ImageEndpointPipe implements PipeTransform {

  transform(endpoint: string): string {
    return `${BASE}:${PORT}/${endpoint}`;
  }
}
```

##### Image-base-64.pipe.ts

```
@Pipe({
  name: 'imageBase64'
})
export class ImageBase64Pipe implements PipeTransform {

  transform(value: string): string {
    return 'data:image/jpeg;base64,' + value;
  }
}
```