TQS: Product specification report

Rafael Kauati [105925], Rafael Vilaça [107476], Miguel Cruzeiro [107660], Diogo Silva [107647]

v2024-06-02

1.1 Overview of the project 1
1.2 Limitations 2
2.1 Vision statement 2
2.2 Personas and scenarios 3
2.2.1 Personas 3
2.2.2 Major scenarios 4
Scenario 1: Movie Night Out 4
Scenario 2 : Keeping the Schedule Up-to-Date 5
Scenario 3 : Receptioning the viewers 5
2.3 Project epics and priorities 6
4.1 Key requirements and constraints 9
4.2 Architecture view 11
4.3 Deployment architecture 12

1 Introduction

1.1 Overview of the project

Cinemax is a cinema management system designed to enhance the movie-going experience for customers and streamline operations for cinema staff. It comprises three core components: Client, Staff, and Digital Signage.

Client: This user-friendly portal empowers customers to effortlessly search and book movie tickets. Key features include intuitive seat selection and seamless, paperless entry using QR codes, ensuring a convenient and modern ticketing experience.

Staff: This comprehensive control panel is tailored for cinema staff, enabling efficient management of movie sessions, seat assignments, and ticket sales.

Digital Signage: These dynamic digital displays are strategically placed throughout the cinema to showcase upcoming movie sessions and real-time updates on their availability, enhancing the overall ambiance and keeping users informed.

This project aims to develop an MVP that adheres to the specified usage scenarios while incorporating enterprise architecture patterns. The solution will be built using collaborative agile practices and a comprehensive Software Quality Assurance (SQA) strategy. To enhance efficiency and quality, we will implement software engineering practices such as continuous testing, continuous integration, and continuous delivery. These practices will minimize manual intervention, save valuable time, foster collaboration and trust within the development team, and streamline the development lifecycle for the rapid delivery of high-quality software releases.

1.2 Limitations

One notable limitation of the Cinemax system is that administrators are required to log in each time they need to validate a ticket. This means they cannot remain logged in to validate multiple tickets consecutively, which can slow down the ticket validation process and reduce overall efficiency.

2 Product concept and requirements

2.1 Vision statement

Purpose of the System: Cinemax is designed to streamline and enhance the experience of going to the movies for both customers and cinema staff. It provides a comprehensive solution for movie selection, ticket booking, seat management, and promotional display.

Business aspects of cinemax system:

1. **Customer Experience Enhancemen**t:
    a. **Convenient Movie Selection and Booking**: Customers can easily search for movies, check showtimes, select seats, and book tickets online, reducing the time and effort required to purchase tickets at the cinema.
    b. **Paperless Entry**: The system provides QR codes for ticket retrieval and cinema entry, minimizing the need for physical tickets and speeding up the entry process.
2. **Operational Efficiency**:
    a. **Streamlined Staff Operations**: Cinema staff can efficiently manage movie schedules, seat arrangements, and ticket sales through a user-friendly interface, reducing administrative overhead and minimizing errors.
    b. **Real-Time Availability and Updates**: The system provides real-time updates on seat availability and occupancy levels, allowing staff to make informed decisions quickly and improving overall resource management.
3. **Promotional and Informational Displays**:
    a. **Dynamic Content Scheduling**: Digital signage in the cinema displays dynamic content such as movie trailers, promotional offers, and showtimes, enhancing marketing efforts and improving customer engagement.
    b. **Real-Time Information**: Integration with the customer portal, and to the digital signage, ensures that displays show up-to-date session capacity and showtimes, providing accurate information to moviegoers.

2.2 Personas and scenarios

2.2.1 Personas

Client

Name: Sarah

Age: 23

Profession: Unemployed

Sarah has recently graduated from college but is currently unemployed. Despite her current situation, she is an enthusiastic cinephile with a deep passion for cinema. Movies serve as a means of escape for her, allowing her to immerse herself in different worlds and stories. Sarah's love for cinema is not limited to any specific genre; she enjoys exploring a wide range of films, from classic masterpieces to the latest blockbusters. As a regular cinema-goer, Sarah sees movies not just as entertainment but as a form of art that sparks creativity and ignites her imagination. She values the communal experience of watching films in theaters, relishing the shared reactions and discussions with fellow audience members.

She wants to see what movie sessions are available and the seats that are already taken.

She wants to be able to buy a ticket to the next big movie premier.

Cinema Employee

Name: Max

Age:35

Profession: Cinema Employee

Max is the dedicated employee of a local cinema. His journey into the cinema industry began as a means to support himself financially, after finishing his Basic

Education degree but, over the years, he has become an integral part of the theater's operations.Max oversees every aspect of the cinema's day-to-day functioning. He needs to schedule screenings of movies in all of the rooms available and manage ticket sales.

Max aims to streamline the management of the cinema and ensure seamless ticket availability for patrons. He is able to add new movies and sessions.

2.2.2 Major scenarios

Scenario 1: Movie Night Out

Actors:

- Client (Moviegoer)

Steps:

1. **Browse Sessions:** The client visualizes at the website a list of movie sessions that are going to be presented at the movie theater in the next few days. The client can see the details like date and time for each session.
2. **Choose a Session:** The client finds a convenient session based on date and time. They tap on the session to proceed.
3. **Select Seats:** The website displays the room layout for the chosen session. The client can see available seats and their corresponding prices. They select the seats they want (e.g., two seats in the middle).
4. **Purchase Ticket:** The client confirms their seat selection and proceeds to checkout. They enter their payment information and complete the ticket purchase.
5. **Ticket Confirmation:** The website displays a confirmation screen showing the movie title, session details, chosen seats, and total price. The client receives a digital ticket via email or within the app for future reference.

“This Friday night hits, and I'm itching to escape the apartment for a movie. With popcorn and adventure in mind, I grab my phone and open my trusty movie ticketing website. Tonight, I have a craving for action-packed thrills. Navigating to the homepage, I spot "The New Blockbuster," the latest flick everyone's been buzzing about.“

“The website showcases a selection of current movies, and I quickly find the poster for "The New Blockbuster." Clicking on it takes me to the movie's dedicated page with session times and details. But hold on... not all of them might be ideal. I click the "Filters" button and narrow things down.

"Action" goes in the genre box, and with a mischievous grin, I set the minimum rating filter to a solid 4 stars. Let's see what survives this cut!“

“Aha! The page refreshes and displays a couple of sessions for "The New Blockbuster" that fit the bill. One's tomorrow night at a theater a bit far out, but the other... that's showing later tonight at a place just around the corner! Perfect. I click on the session for a closer look.“

“The screen shows the seating chart. Scanning the rows, I spot two seats right in the middle, prime position for optimal popcorn distribution (and dodging flying popcorn kernels!). Clicking on the seats secures them as mine, and with a triumphant click, I head to checkout.“

“A quick breeze through the payment process, and that's it! The website displays a confirmation screen with all the details - movie title, date, time, my snagged seats - and the grand total. But most importantly, nestled right below, is my digital ticket, ready to be scanned on the big night. Now, all that's left is to grab some snacks, and "The New Blockbuster," here I come!“

Scenario 2 : Keeping the Schedule Up-to-Date

Actor: Theater Employee (Max)

Max, a friendly employee at the local Cinemax, arrives for his shift. He checks the movie schedule for the upcoming week and notices a glaring omission - "Space Adventure 3" isn't listed yet, despite its release being tomorrow! Knowing many fans are eagerly awaiting this sci-fi epic, Max gets to work.

Adding a New Session:

1. **Access System:** Max logs in to the employee portal of the theater's ticketing system.
2. **New Session:** He navigates to the section for managing movie sessions and clicks on "Add New Session."
3. **Movie Selection:** A list of movies pops up. Max quickly finds "Space Adventure 3" and selects it.
4. **Session Details:** A form appears for entering session details. Max fills in the information:
    - **Room:** He selects the most appropriate theater room based on the movie's expected popularity.
    - **Schedule:** He enters the date and showtime for the new session, ensuring it aligns with the movie's release date.
5. **Confirmation & Save:** After double-checking all the details, Max confirms the new session. The system adds it to the schedule, and it's now visible to moviegoers searching for "Space Adventure 3."

“Max sighs in relief. "Space Adventure 3" fans can now book their tickets, and the horror movie enthusiasts will find their spooky experience intact, just in a different theater room. He knows a smooth-running movie schedule is key to a happy audience, and he's glad he could play his part.“

Scenario 3 : Receptioning the viewers

Actor: Theater Employee (Sarah)

The long line of moviegoers inches forward as Sarah, a ticket validator at the entrance, greets each person with a smile. Tonight's a blockbuster premiere, and the excitement is palpable.

Validating Tickets:

1. **Ticket Validation**: Sarah has one way to validate tickets: by scanning the QR code.
2. **Scanning Ticket**: The first customer presents their phone, displaying the digital ticket with a QR code. Sarah aims her handheld scanner at the QR code and waits for the confirmation.
3. **System Check**: The validation device connects to the central ticketing system. A quick beep and a green light confirm the ticket's authenticity. Sarah gives a thumbs up, and the customer proceeds with a big grin.
4. **Error Handling**: Occasionally, a red light and a beep signal an invalid ticket. Sarah politely informs the customer and directs them to the customer service desk for assistance.

“ As the night progresses, Sarah efficiently validates tickets using this method, ensuring a smooth entry for moviegoers. The validation process helps prevent ticket fraud and keeps track of attendance for each session. By the end of the night, Sarah knows she's played a part in creating a memorable movie experience for everyone.“

2.3 Project epics and priorities

Use case (episode of use related to some actor's objective. → defined by the Analist)	**Epic** (bodies of work that can be broken down into a number of smaller tasks (called stories)
Choose the seats and purchase the ticket	**Client should be able to choose available seats and purchase the ticket** As a **Client,** after choosing a session, i should be able to choose the seat(s), available ones, to book in the session and purchase the ticket for the session with the seats i selected.
Retrieve all tickets bought	**Client should be able to see all the tickets bought** As a **Client,** i should be able to check all the tickets i previously bought, and their details as well
Add new movie session	**Employee should be able to add a new session for a movie** As a **Employee** of the company, i should be able to add new a movie session, for the theater he works on, the new session will associate the room number, number of seats available, price per ticket, date and hour of the session, and other info
Validate client ticket	**Employee should be able to validate a client`s ticket** As a **Employee** of the company, i should be able to to validate a given client`s ticket scanning it (e.g. QR code)
Check for coming movies	**Client can search for session for a movie** As a **Client,** i should be able to visualize the next n movies sessions to be started in the movie theater
Add movie	**Employee should be able to add a new movie.** As an **Employee** of the company, I should be able to add a new movie that will be shown in the cinema.
Delete Movie	**Employee should be able to delete a movie.** As an **Employee** of the company, I should be able to delete an existing movie.
Delete movie session	**Employee should be able to delete a session** As an **Employee** of the company, I should be able to delete an existing movie session.
See the upcoming movie sessions for the current day	**Client should be able to see all the upcoming sessions for the day.** As a **Client**, I should be able to see the upcoming sessions through the digital signage page, and they're respective up-to-date info (e.g session's seat availability).

3 Domain model

User: This is the basic model. It specifies common properties such as username, email, and password that apply to all users in the system. Clients and Employees are subclasses of Users. They inherit User traits but may also have role-specific attributes. For example, Employee may have an "employeeId" property.

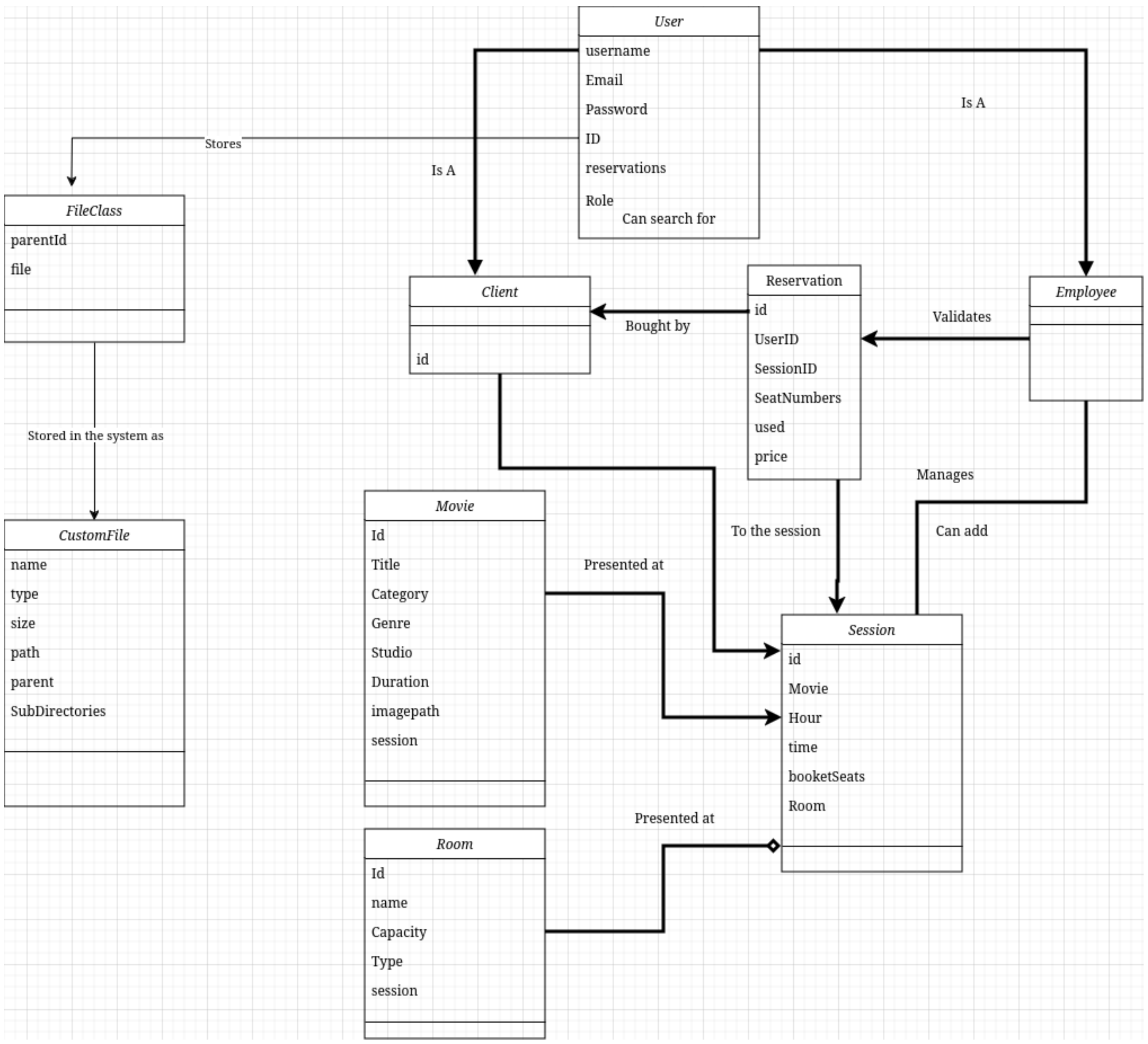
FileClass and CustomFile: CustomFile is the sql entity model and FileClass is the class used to receive multipart files from files upload in the respective endpoint, this would be used to create a filesystem in the application but since we didn't had enough time the frontend implementation didn't happen.

Movie:This model defines basic properties for the movies, worth noting that MovieClass class was created, this class works like the files class and allows the movie image to be uploaded (optionally) being its path stored in the movie model.

Session: This model has a connection with movies. A Session denotes a specific screening of a movie. It would not exist without a movie. The class is associated with Room. A MovieSession refers to the room where the movie will be screened. They both exist independently, but a MovieSession defines which room to utilize. This relationship might be accomplished using foreign keys in both classes that reference each other's IDs.

Reservation: This model has a partial connection with Client. A reservation is made by a client and is considered part of the client's records. If a client is eliminated, their bookings may also be removed.

Room: This model represents the rooms in which movies are screened. Each room can have attributes such as room name, seating capacity, and type. Rooms are used in the Session model to specify where a movie will be shown.



4 Architecture notebook

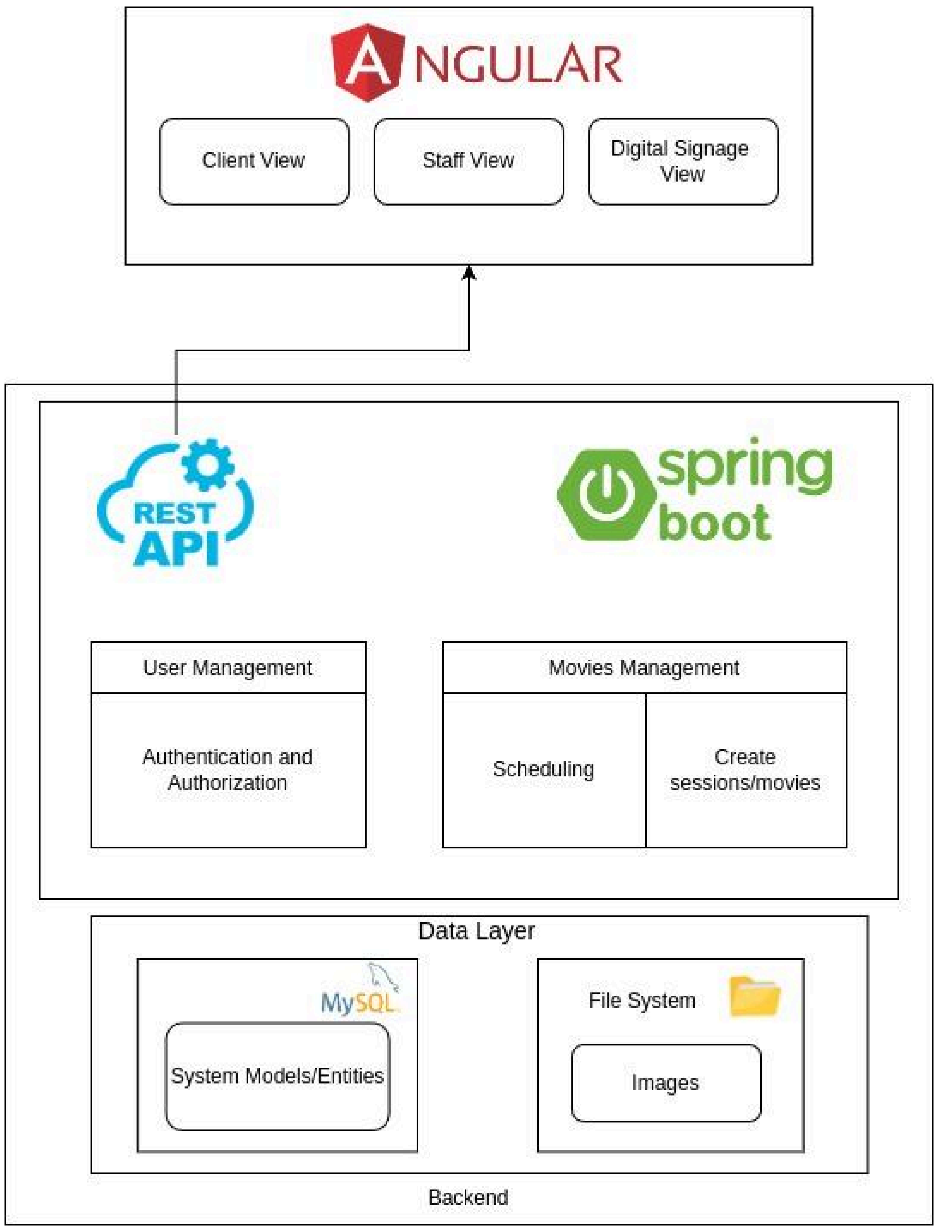
4.1 Key requirements and constraints

- The system in its entirety must have a clearly role-separated domain, so that the client and staff limits of data access and functionality are correctly defined, created, tested, and separately, but connected and synchronized, operated from each other.
- When it comes to server-side services, the system must be able to constantly receive and make accessible data linked to movie sessions in order to increase data consumption. In the context of the project, we're primarily concerned with digital signage, which means it must manage numerous incoming client actions, such as ticket purchases, and make session information available in real time to any client. Staff members' domain updates

must be made available in real time as quickly as feasible, as this configures the project's fundamental client-server behavior.

- Regarding the client-use interfaces, particularly the digital signage, it's vital to note that this module must, operationally speaking, make continual and periodic calls to the server in order to always show up-to-date information about movie sessions.
- In terms of customer reception, the user interface must also be capable of performing paperless and efficient ticket validation via QR code verification.
- The user interface should also be responsive, because they should be shown in a mobile (cell phone) format, given that clients will present their tickets via cell phones.

4.2 Architecture view



We have 3 interfaces one for the client, another for the admin, and the digital signage one, all of them represented by the angular block.

They rely on business services and validation from the backend server and API, developed with Spring Boot framework and dependency control maven.

The backend service manages the storing, querying, and updating of data tables in the MySQL database. To organize everything up, each component is designed to run virtually in docker containers with a docker-compose configuration, making it easier to deploy the entire application.

4.3 Deployment architecture

The deployment architecture of the Cinemax application involves a multi-tier setup utilizing Docker containers to streamline development, testing, and production environments. This architecture ensures that each component of the application is isolated, easily scalable, and manageable. The architecture consists of three primary services: a **MySQL database**, a **Spring Boot backend**, and an **Angular frontend**. These services are connected through a Docker bridge network, allowing for seamless communication and interoperability.

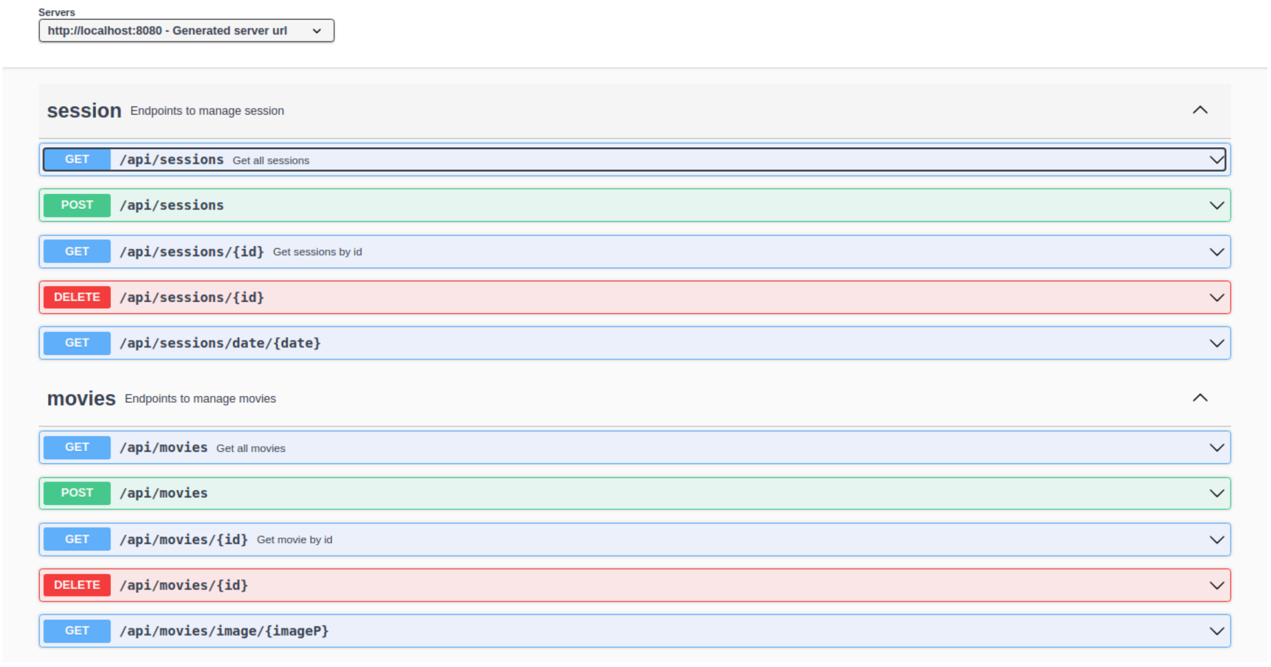
Each component is exposing specific ports to enable communication and access:

1. **MySQL Service**: Exposes port **3306** for database access.
2. **Spring Boot Service**: Exposes ports **8080** for the application, **35729** for LiveReload (useful during development), and **5005** for debugging.
3. **Angular Service**: Exposes port **4200** for accessing the frontend application.

These exposed ports allow users and other services to interact with the database, backend, and frontend of the Cinemax application seamlessly.

5 API for developers

The API documentation was hosted via Swagger.



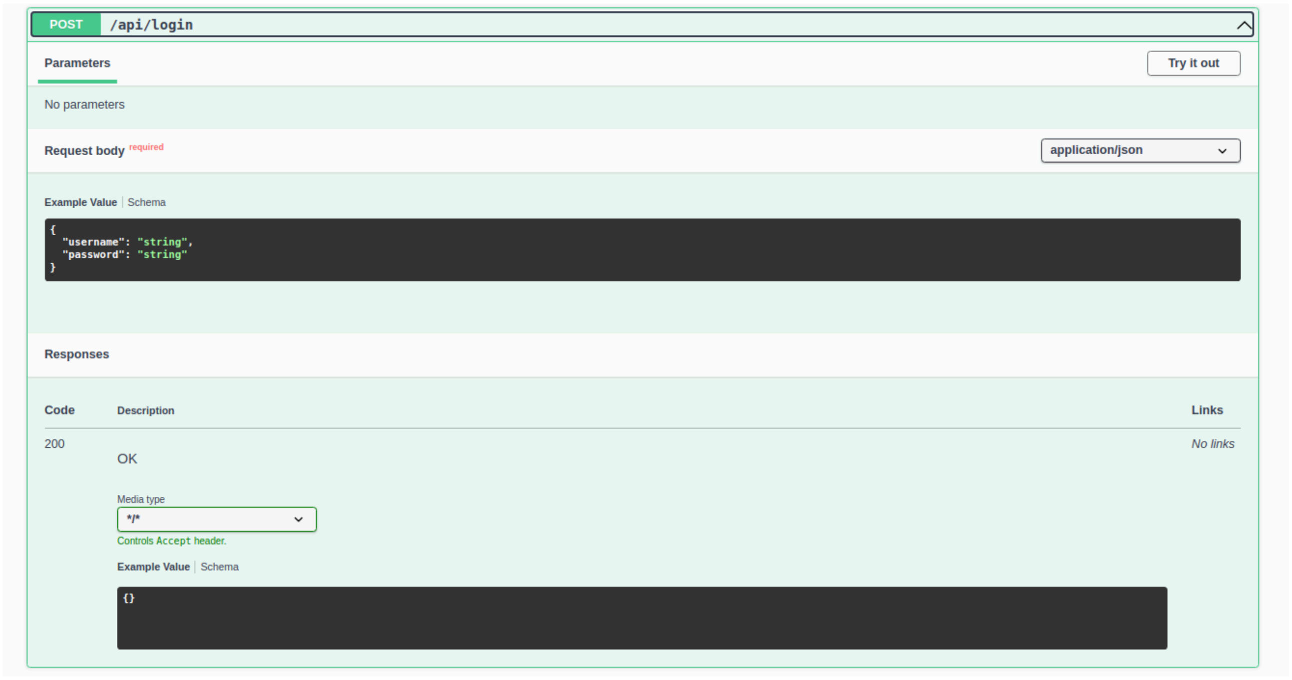
Servers
http://localhost:8080 - Generated server url

session Endpoints to manage session

GET /api/sessions Get all sessions
POST /api/sessions
GET /api/sessions/{id} Get sessions by id
DELETE /api/sessions/{id}
GET /api/sessions/date/{date}

movies Endpoints to manage movies

GET /api/movies Get all movies
POST /api/movies
GET /api/movies/{id} Get movie by id
DELETE /api/movies/{id}
GET /api/movies/image/{imageP}



POST /api/login

Parameters | Try it out

No parameters

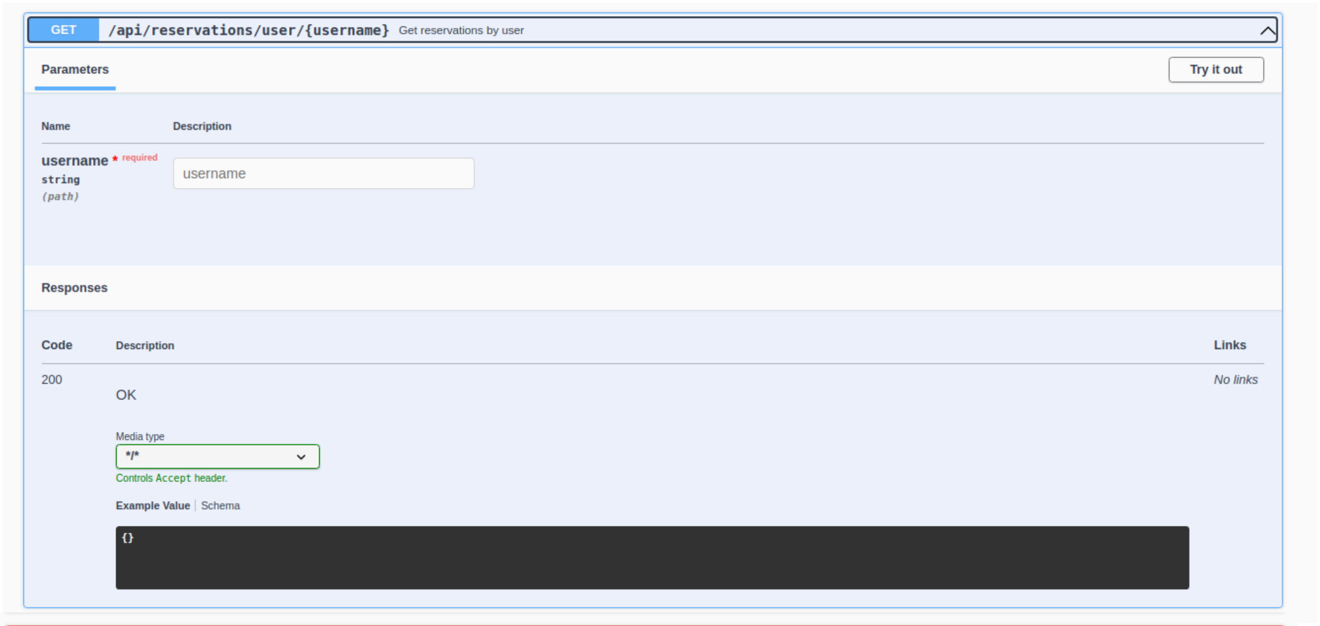
Request body required | application/json

Example Value | Schema

```
{
  "username": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	OK Media type: */* Controls Accept header. Example Value | Schema `{}`	*No links*



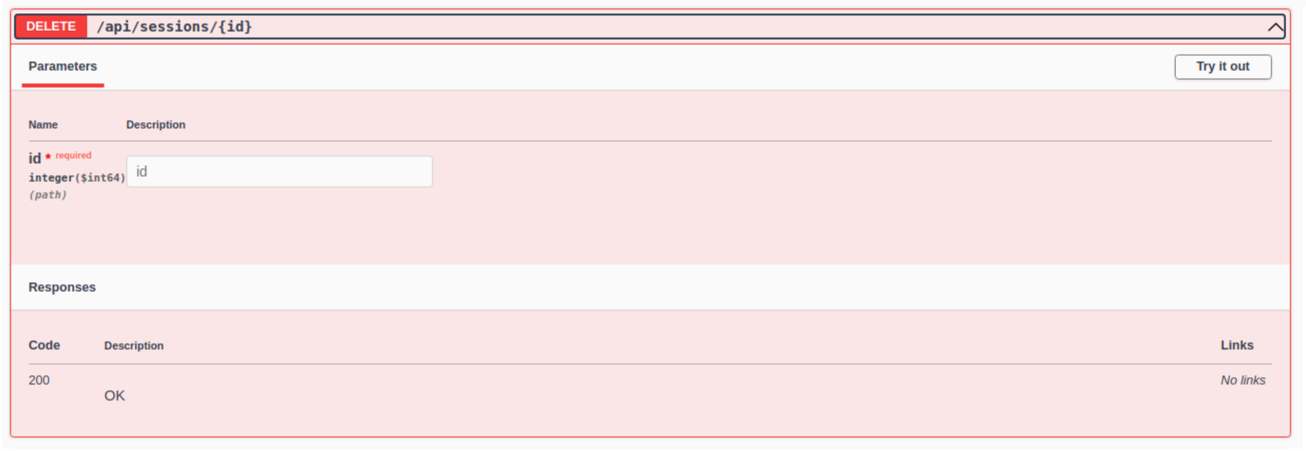
GET /api/reservations/user/{username} Get reservations by user

Parameters | Try it out

Name	Description
username * required string *(path)*	username

Responses

Code	Description	Links
200	OK Media type: */* Controls Accept header. Example Value | Schema `{}`	*No links*



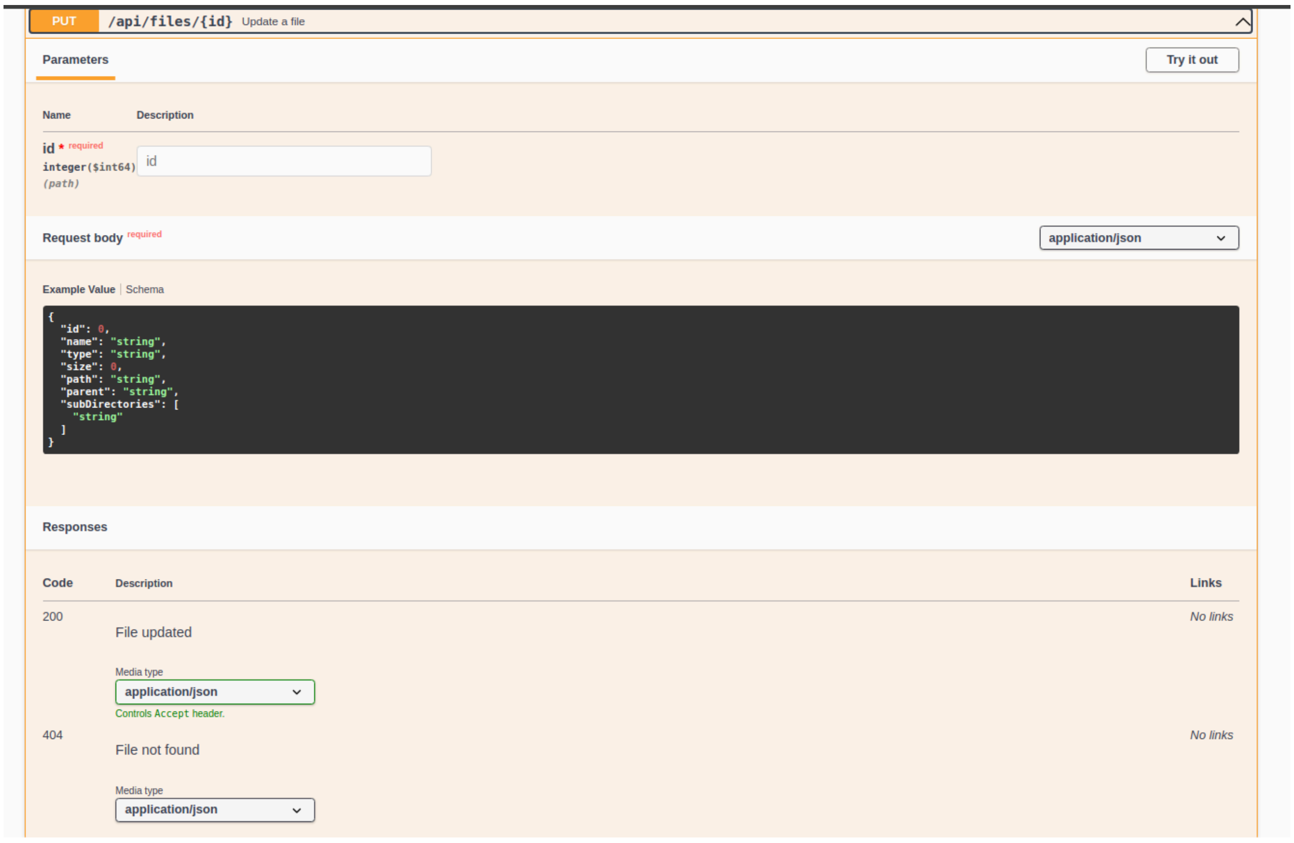
DELETE /api/sessions/{id}

Parameters | Try it out

Name	Description
id * required integer($int64) *(path)*	id

Responses

Code	Description	Links
200	OK	*No links*



PUT `/api/files/{id}` Update a file

Parameters

Try it out

Name	Description
id * required `integer($int64)` *(path)*	id

Request body required

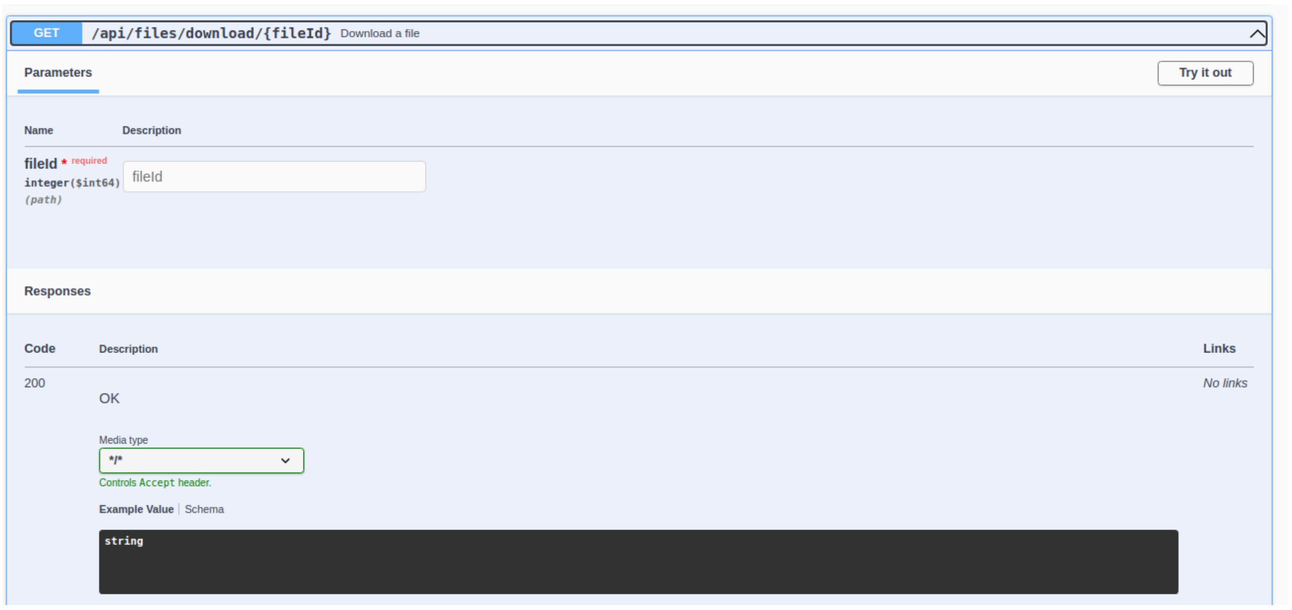
application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "type": "string",
  "size": 0,
  "path": "string",
  "parent": "string",
  "subDirectories": [
    "string"
  ]
}
```

Responses

Code	Description	Links
200	File updated Media type application/json Controls `Accept` header.	*No links*
404	File not found Media type application/json	*No links*



GET `/api/files/download/{fileId}` Download a file

Parameters

Try it out

Name	Description
fileId * required `integer($int64)` *(path)*	fileId

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. Example Value | Schema `string`	*No links*

6 References and resources

Project resources

Resource:	**URL/location:**
Git repository	https://github.com/DiogoSilva1904/CineMax
Video demo	https://www.youtube.com/watch?v=2gMAwDJ-4PQ
QA dashboard (online)	https://sonarcloud.io/summary/new_code?id=DiogoSilva1904_CineMax
CI/CD pipeline	https://github.com/DiogoSilva1904/CineMax/tree/main/.github/workflows
Deployment ready to use	http://deti-tqs-10.ua.pt:4200/