# Cinemax

Final Project TQS

*Rafael Kauati 105925*
*Rafael Vilaça 107476*
*Miguel Cruzeiro 107770*
*Diogo Silva 107647*

# TOPIC OUTLINE

# Team and Roles

Rafael

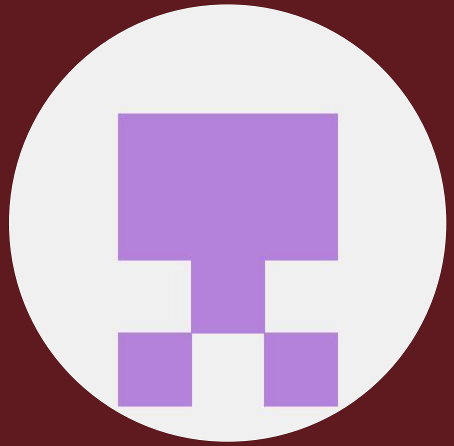
Team Leader

Miguel

Product Owner

Rafael

DevOps Master


Diogo

QA Engineer

# Cinemax

# Cinemax

- Client: movie sessions visualization and tickets booking
- Staff : Manage movies and sessions
- Digital signage : Display up-to-date info about  upcoming movie sessions availability
- Ticket Validation: Efficient paperless validation with QR code

# Personas

## Client

- Name : Sarah
- Age : 23 y.o
- Profession : unemployed
- Movie geek

## Employee

- Name : Max
- Age : 45 y.o
- Profession : Cinema Employee
- Dedicated movie theater manager

## Scenarios

“Craving an action movie on Friday night, you open your favorite ticketing website, filter by genre and high ratings, and find the perfect showing of "The New Blockbuster" at a nearby theater later that night. After selecting your ideal seats, you breeze through checkout and secure your digital ticket, ready for a popcorn-filled adventure.”

## Movie Night Out

“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.”

## Keeping the schedule up-to-date

"As the night unfolds, Sarah swiftly validates tickets using this method, guaranteeing a seamless entry for moviegoers. This validation process helps to deter ticket fraud and maintains accurate attendance records for each showtime. By the end of the evening, Sarah can take pride in knowing she's contributed to creating a fantastic movie experience for everyone."

## Receptioning viewers

## Epics

## Choose the seats and purchase the tickets

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

As a Client, i should be able to check all the tickets i previously bought, and their details as well

## Add new movie sessions

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

As a Employee of the company, i should be able to validate a given client`s ticket scanning it (e.g. QR code)

## Add/Delete movie

As an Employee, i should be able to add a movie or delete an existing moving

## Check for coming movies

As a Client, i should be able to visualize the next n movies sessions to be started in the movie theater

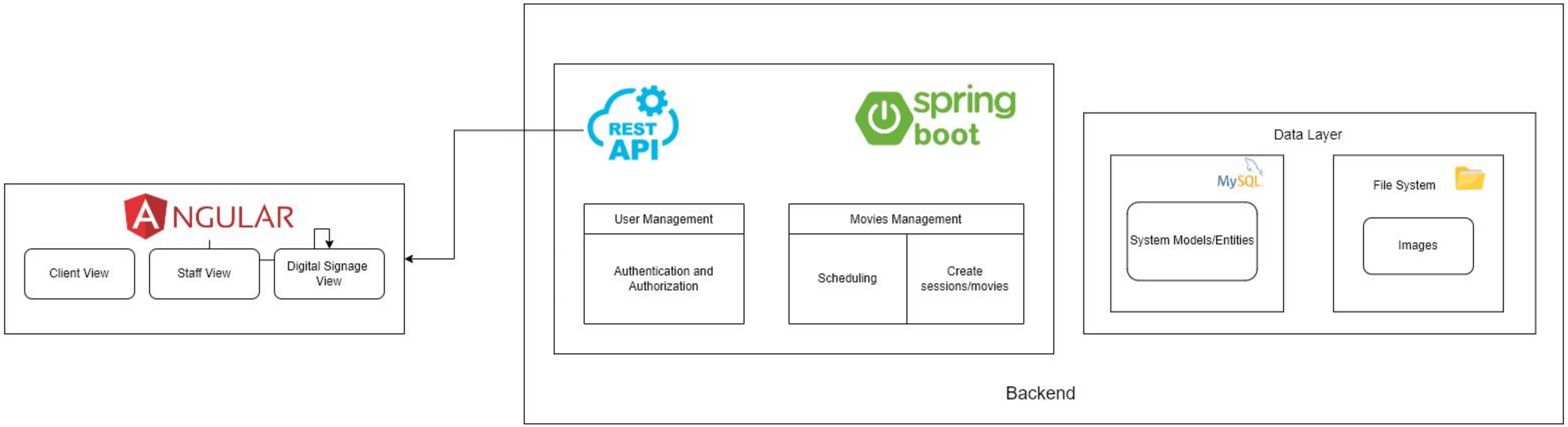
## Delete movie sessions

As a Employee of the company, i should be able to delete a existing movie session

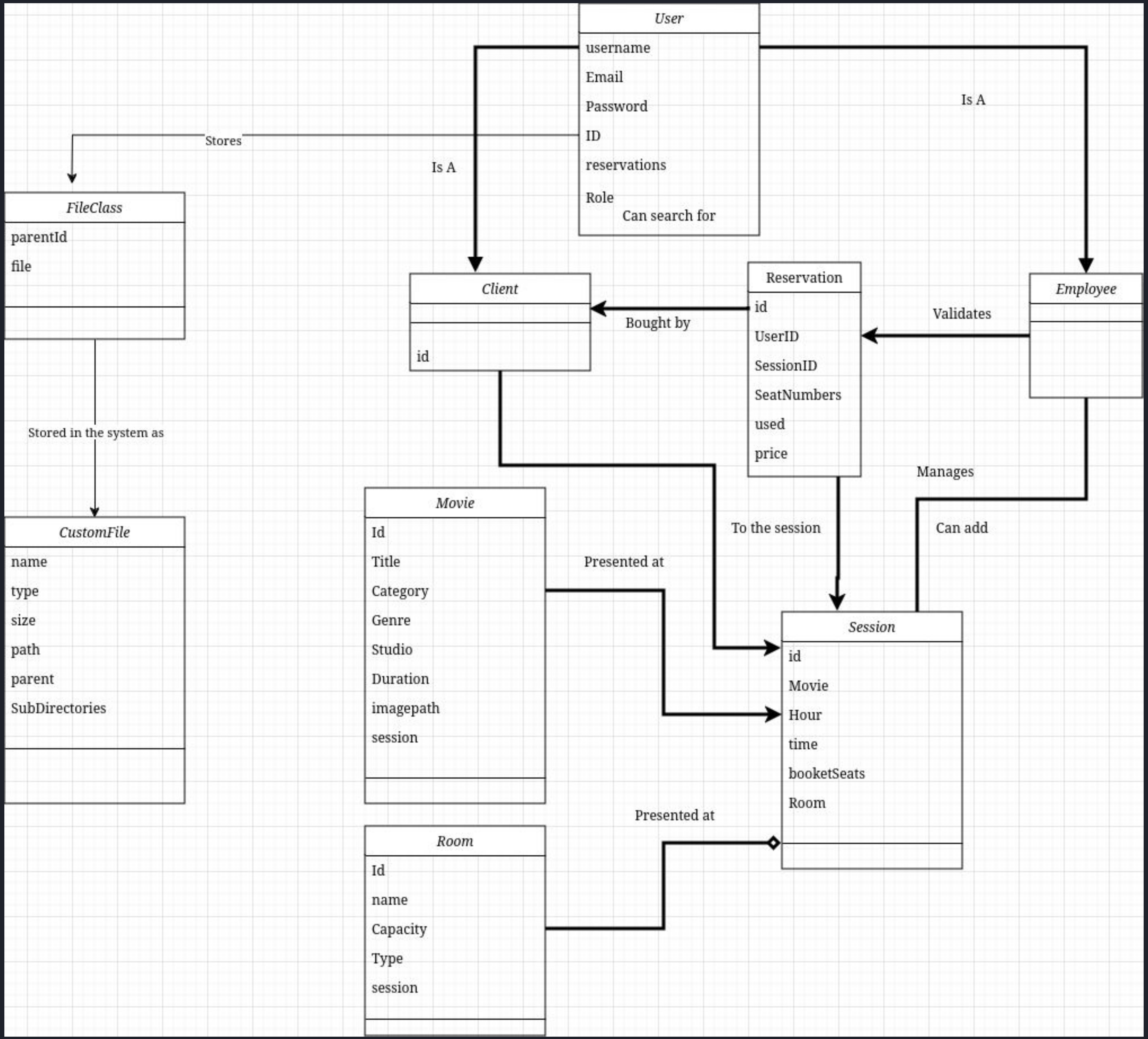
## See the upcoming movie sessions for the current day

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).

## Architecture

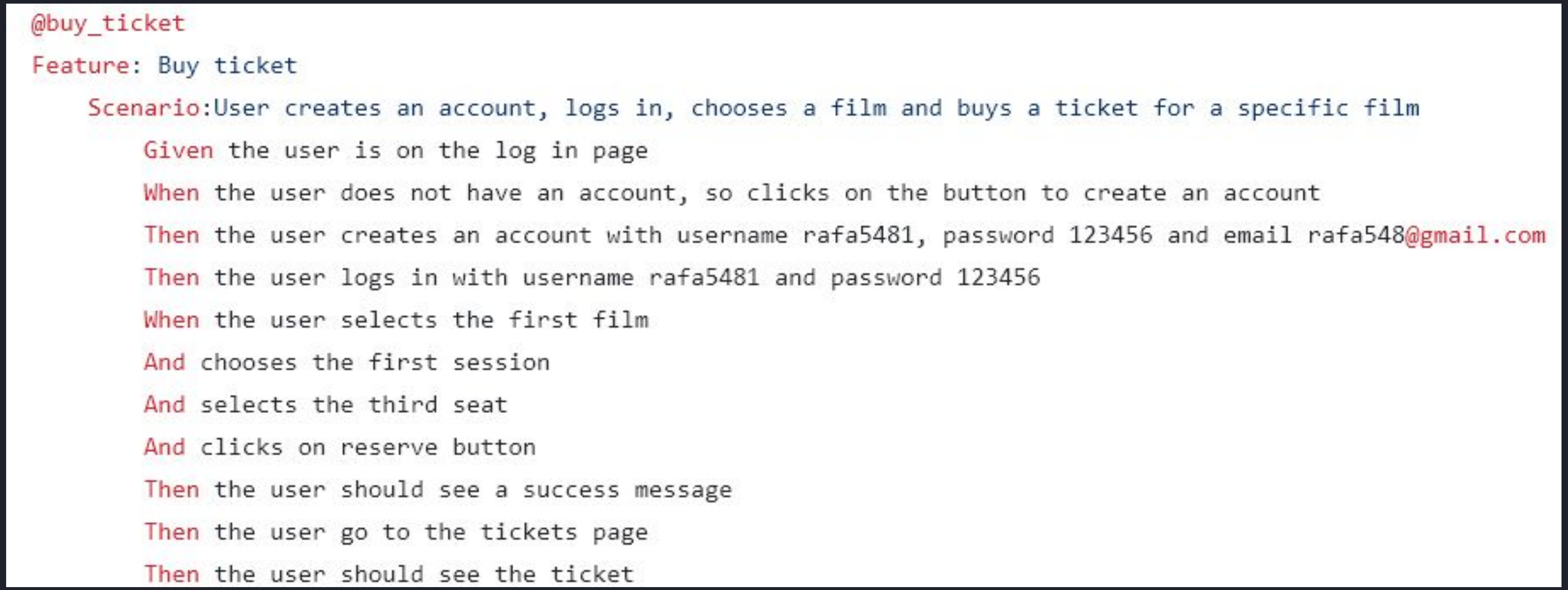

ANGULAR
Client View
Staff View
Digital Signage View
REST API
spring boot
User Management
Authentication and Authorization
Movies Management
Scheduling
Create sessions/movies
Data Layer
MySQL
System Models/Entities
File System
Images
Backend

## Domain Model

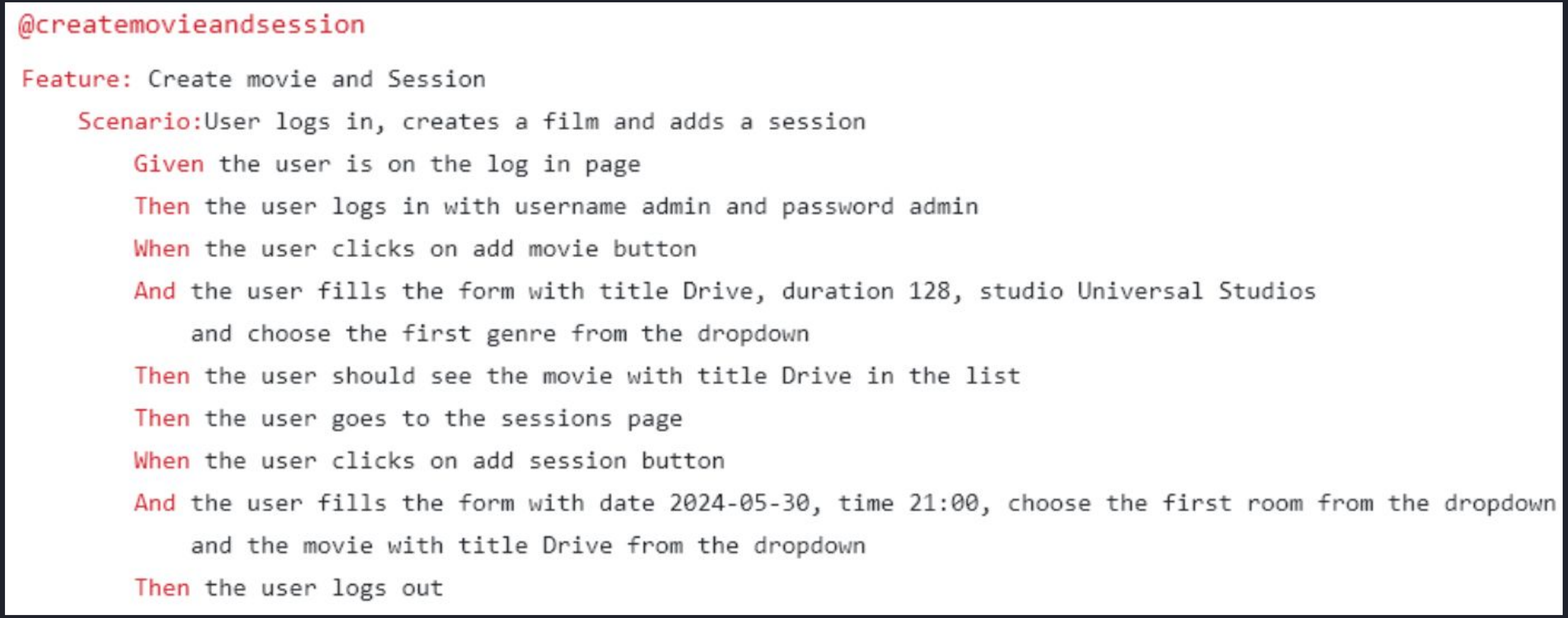

User
username
Email
Password
ID
reservations
Role
Can search for
Is A
Is A
Stores
FileClass
parentId
file
Stored in the system as
CustomFile
name
type
size
path
parent
SubDirectories
Client
id
Bought by
Reservation
id
UserID
SessionID
SeatNumbers
used
price
Validates
Employee
Manages
Can add
To the session
Movie
Id
Title
Category
Genre
Studio
Duration
imagepath
session
Presented at
Session
id
Movie
Hour
time
booketSeats
Room
Presented at
Room
Id
name
Capacity
Type
session

## Tests

# Functional Tests

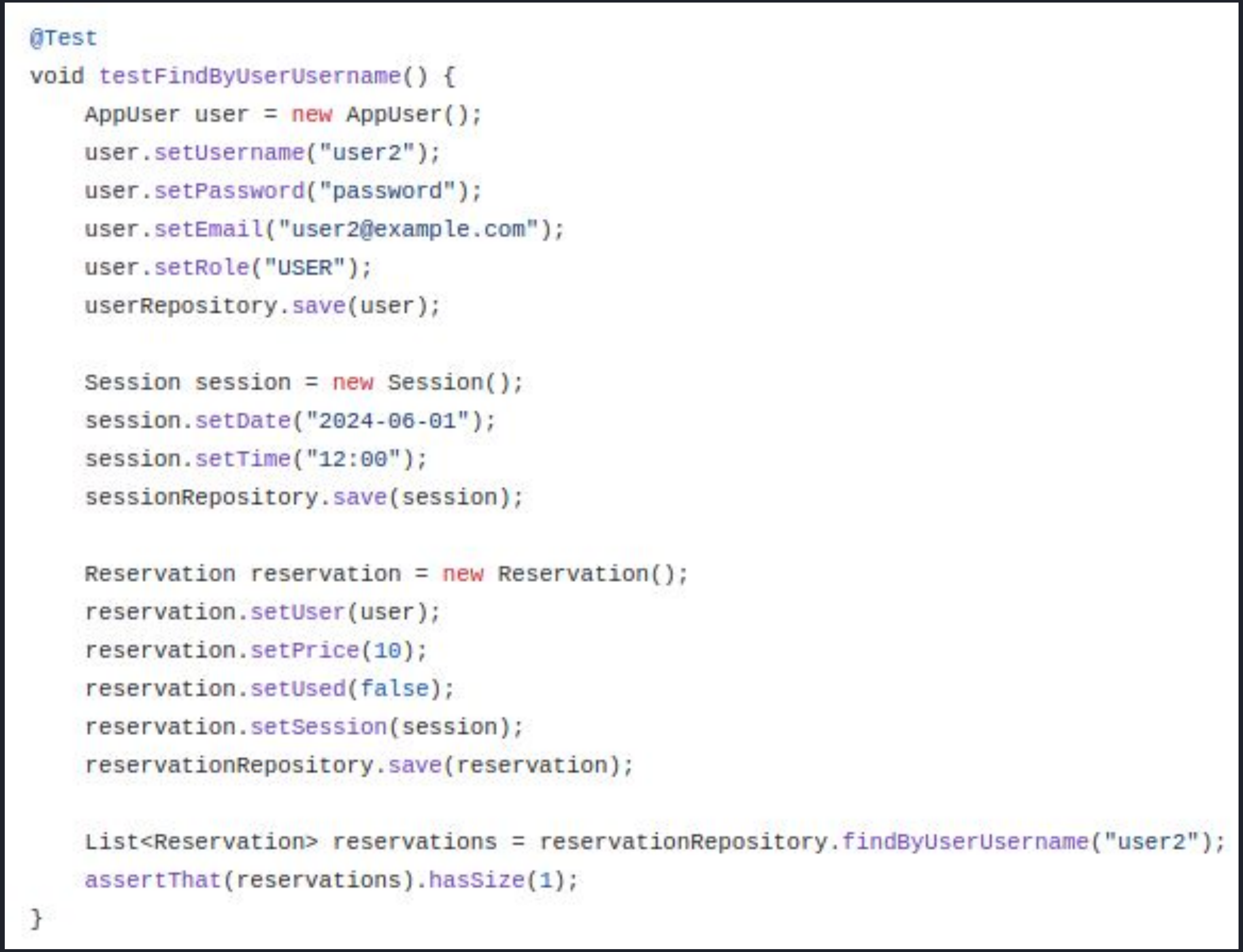


```
@buy_ticket
Feature: Buy ticket
    Scenario:User creates an account, logs in, chooses a film and buys a ticket for a specific film
        Given the user is on the log in page
        When the user does not have an account, so clicks on the button to create an account
        Then the user creates an account with username rafa5481, password 123456 and email rafa548@gmail.com
        Then the user logs in with username rafa5481 and password 123456
        When the user selects the first film
        And chooses the first session
        And selects the third seat
        And clicks on reserve button
        Then the user should see a success message
        Then the user go to the tickets page
        Then the user should see the ticket
```

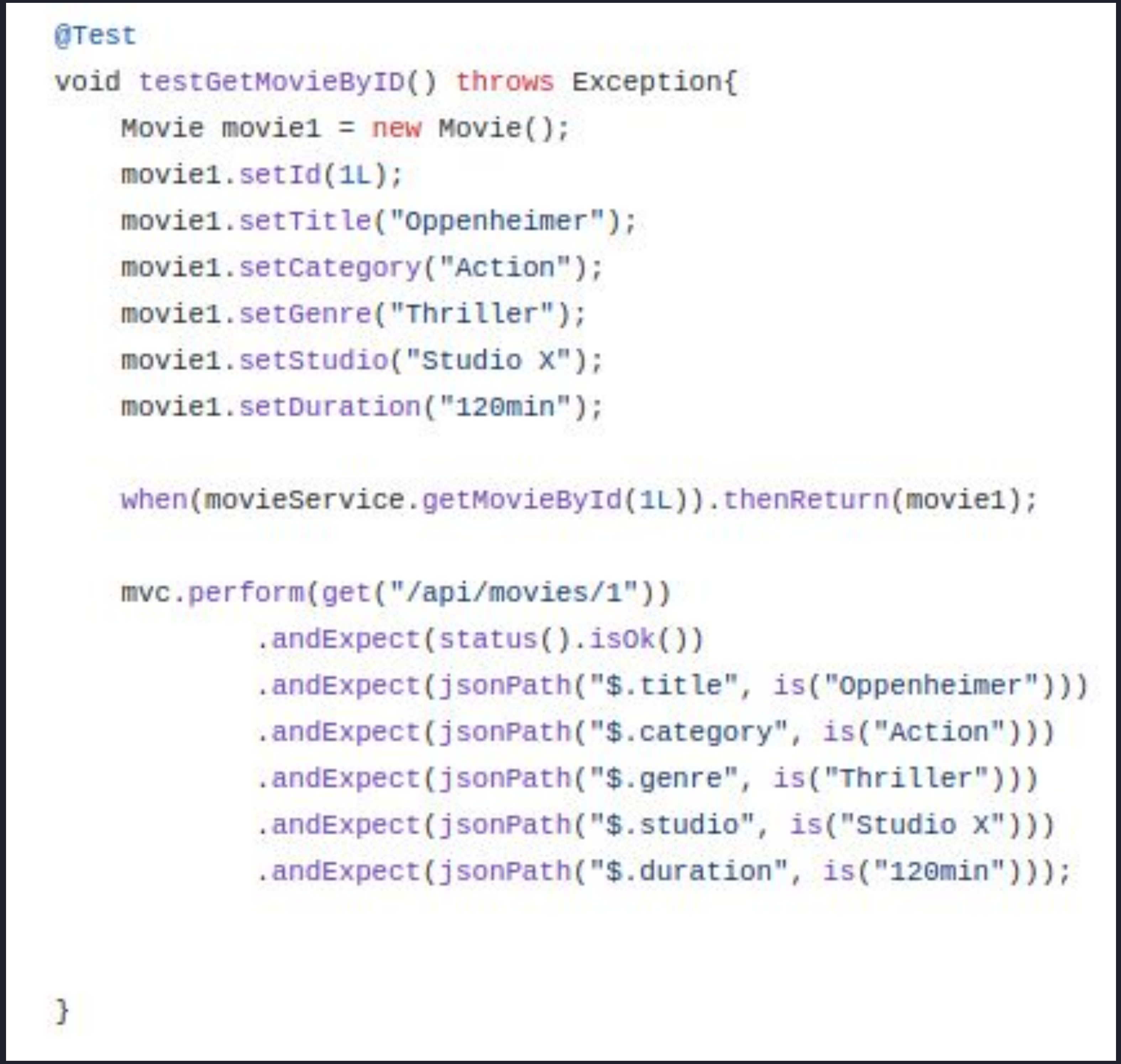


```
@createmovieandsession
Feature: Create movie and Session
    Scenario:User logs in, creates a film and adds a session
        Given the user is on the log in page
        Then the user logs in with username admin and password admin
        When the user clicks on add movie button
        And the user fills the form with title Drive, duration 128, studio Universal Studios
            and choose the first genre from the dropdown
        Then the user should see the movie with title Drive in the list
        Then the user goes to the sessions page
        When the user clicks on add session button
        And the user fills the form with date 2024-05-30, time 21:00, choose the first room from the dropdown
            and the movie with title Drive from the dropdown
        Then the user logs out
```

# Units Tests



```
@Test
void testFindByUserUsername() {
    AppUser user = new AppUser();
    user.setUsername("user2");
    user.setPassword("password");
    user.setEmail("user2@example.com");
    user.setRole("USER");
    userRepository.save(user);

    Session session = new Session();
    session.setDate("2024-06-01");
    session.setTime("12:00");
    sessionRepository.save(session);

    Reservation reservation = new Reservation();
    reservation.setUser(user);
    reservation.setPrice(10);
    reservation.setUsed(false);
    reservation.setSession(session);
    reservationRepository.save(reservation);

    List<Reservation> reservations = reservationRepository.findByUserUsername("user2");
    assertThat(reservations).hasSize(1);
}
```

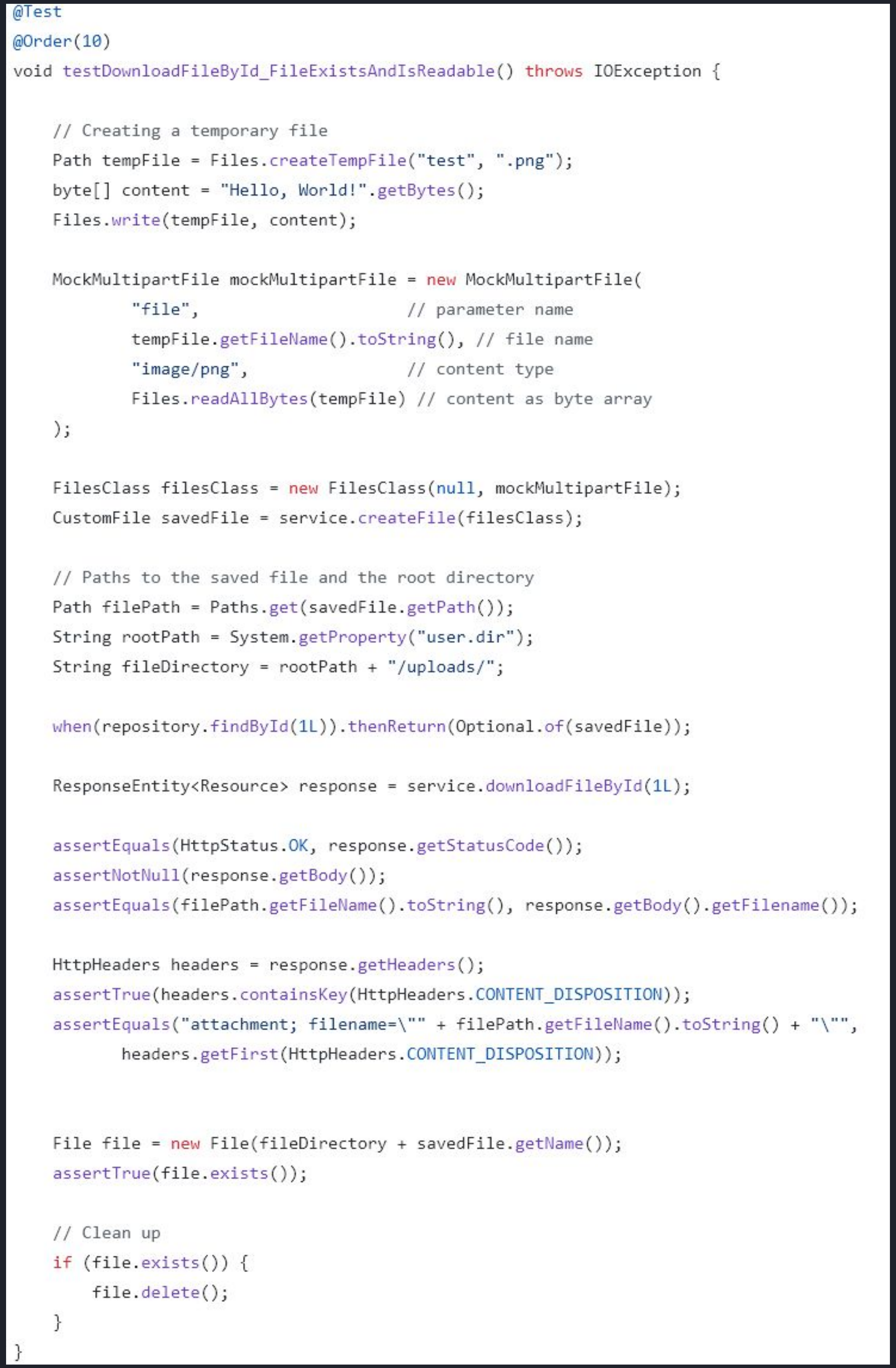


```
@Test
void testGetMovieByID() throws Exception{
    Movie movie1 = new Movie();
    movie1.setId(1L);
    movie1.setTitle("Oppenheimer");
    movie1.setCategory("Action");
    movie1.setGenre("Thriller");
    movie1.setStudio("Studio X");
    movie1.setDuration("120min");

    when(movieService.getMovieById(1L)).thenReturn(movie1);

    mvc.perform(get("/api/movies/1"))
            .andExpect(status().isOk())
            .andExpect(jsonPath("$.title", is("Oppenheimer")))
            .andExpect(jsonPath("$.category", is("Action")))
            .andExpect(jsonPath("$.genre", is("Thriller")))
            .andExpect(jsonPath("$.studio", is("Studio X")))
            .andExpect(jsonPath("$.duration", is("120min")));


}
```

Repository Layer

Controller Layer

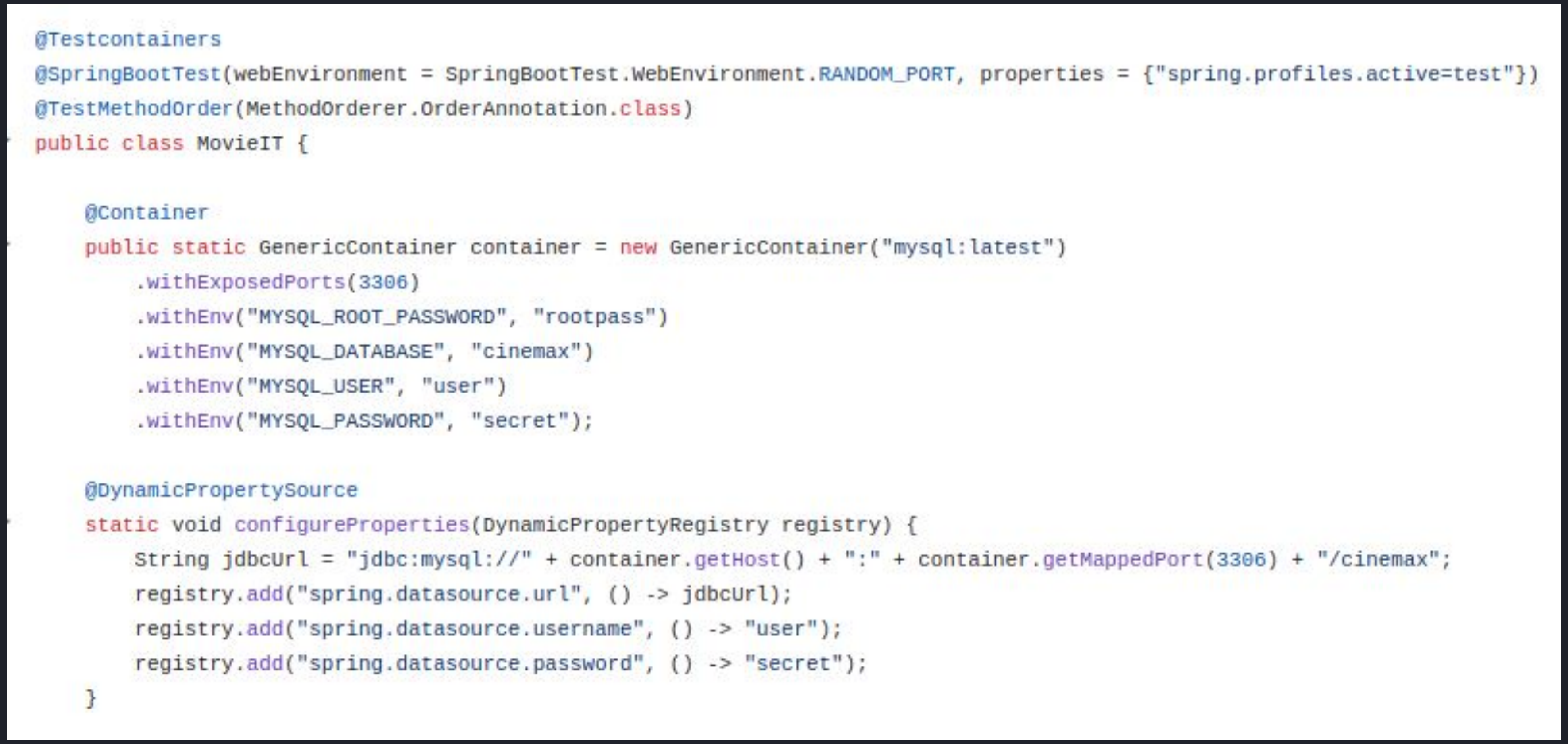
# Units Tests



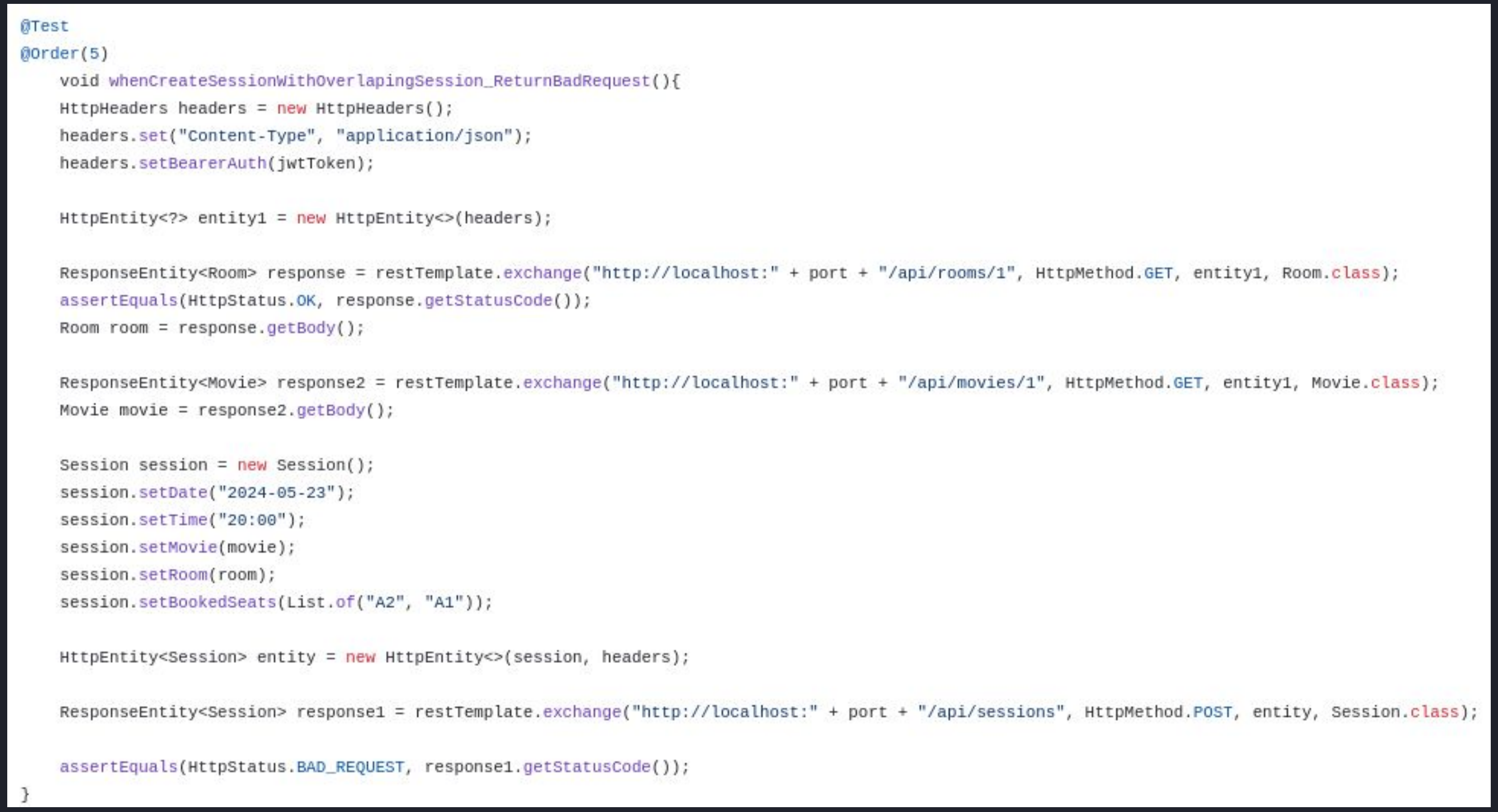
```java
@Test
@Order(10)
void testDownloadFileById_FileExistsAndIsReadable() throws IOException {

    // Creating a temporary file
    Path tempFile = Files.createTempFile("test", ".png");
    byte[] content = "Hello, World!".getBytes();
    Files.write(tempFile, content);

    MockMultipartFile mockMultipartFile = new MockMultipartFile(
            "file",                      // parameter name
            tempFile.getFileName().toString(), // file name
            "image/png",                 // content type
            Files.readAllBytes(tempFile) // content as byte array
    );

    FilesClass filesClass = new FilesClass(null, mockMultipartFile);
    CustomFile savedFile = service.createFile(filesClass);

    // Paths to the saved file and the root directory
    Path filePath = Paths.get(savedFile.getPath());
    String rootPath = System.getProperty("user.dir");
    String fileDirectory = rootPath + "/uploads/";

    when(repository.findById(1L)).thenReturn(Optional.of(savedFile));

    ResponseEntity<Resource> response = service.downloadFileById(1L);

    assertEquals(HttpStatus.OK, response.getStatusCode());
    assertNotNull(response.getBody());
    assertEquals(filePath.getFileName().toString(), response.getBody().getFilename());

    HttpHeaders headers = response.getHeaders();
    assertTrue(headers.containsKey(HttpHeaders.CONTENT_DISPOSITION));
    assertEquals("attachment; filename=\"" + filePath.getFileName().toString() + "\"",
           headers.getFirst(HttpHeaders.CONTENT_DISPOSITION));


    File file = new File(fileDirectory + savedFile.getName());
    assertTrue(file.exists());

    // Clean up
    if (file.exists()) {
        file.delete();
    }
}
```

## Service Layer

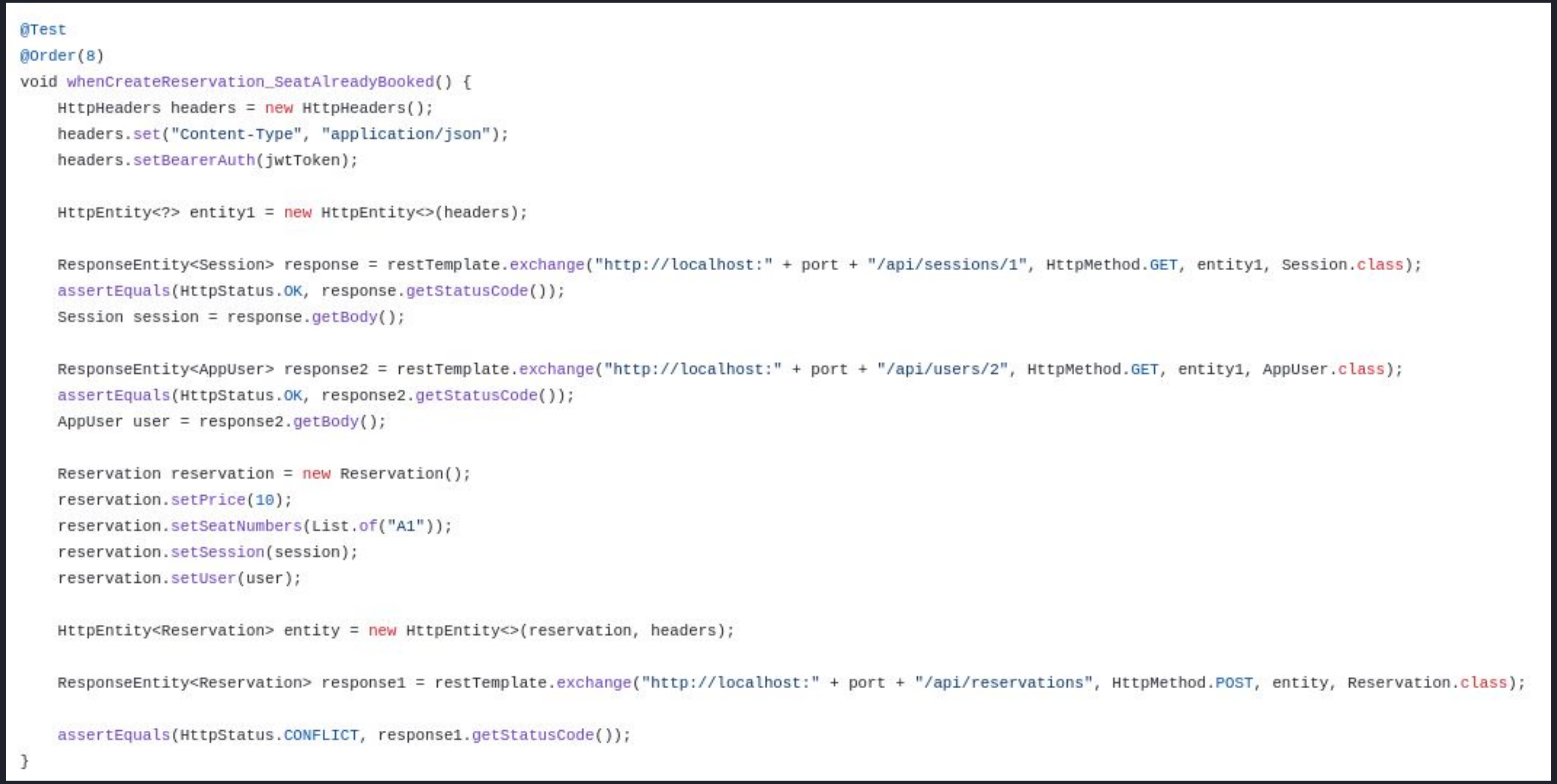
# IT Tests



```java
@Testcontainers
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT, properties = {"spring.profiles.active=test"})
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class MovieIT {

    @Container
    public static GenericContainer container = new GenericContainer("mysql:latest")
        .withExposedPorts(3306)
        .withEnv("MYSQL_ROOT_PASSWORD", "rootpass")
        .withEnv("MYSQL_DATABASE", "cinemax")
        .withEnv("MYSQL_USER", "user")
        .withEnv("MYSQL_PASSWORD", "secret");

    @DynamicPropertySource
    static void configureProperties(DynamicPropertyRegistry registry) {
        String jdbcUrl = "jdbc:mysql://" + container.getHost() + ":" + container.getMappedPort(3306) + "/cinemax";
        registry.add("spring.datasource.url", () -> jdbcUrl);
        registry.add("spring.datasource.username", () -> "user");
        registry.add("spring.datasource.password", () -> "secret");
    }
```

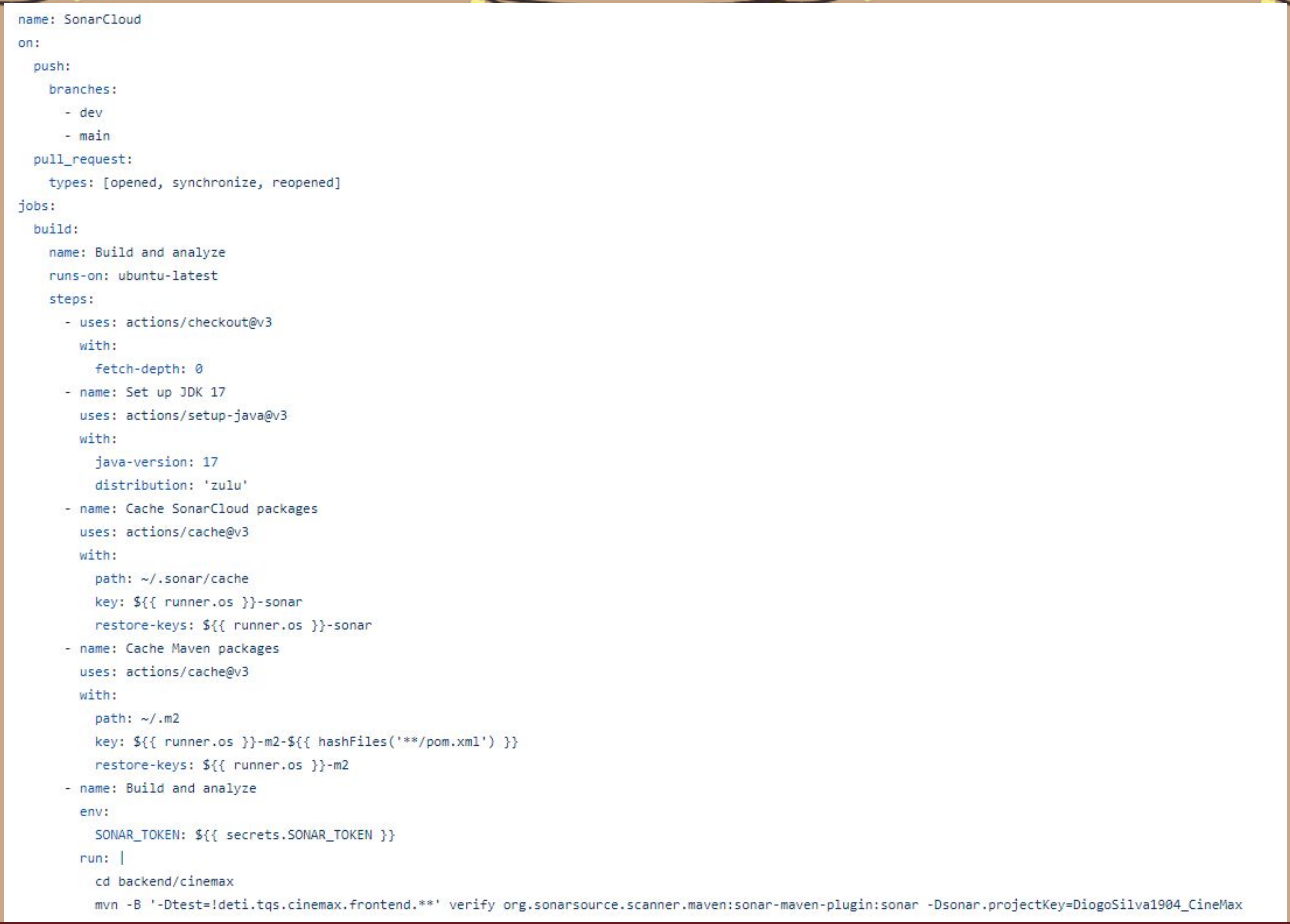


```java
@Test
@Order(5)
    void whenCreateSessionWithOverlapingSession_ReturnBadRequest(){
    HttpHeaders headers = new HttpHeaders();
    headers.set("Content-Type", "application/json");
    headers.setBearerAuth(jwtToken);

    HttpEntity<?> entity1 = new HttpEntity<>(headers);

    ResponseEntity<Room> response = restTemplate.exchange("http://localhost:" + port + "/api/rooms/1", HttpMethod.GET, entity1, Room.class);
    assertEquals(HttpStatus.OK, response.getStatusCode());
    Room room = response.getBody();

    ResponseEntity<Movie> response2 = restTemplate.exchange("http://localhost:" + port + "/api/movies/1", HttpMethod.GET, entity1, Movie.class);
    Movie movie = response2.getBody();

    Session session = new Session();
    session.setDate("2024-05-23");
    session.setTime("20:00");
    session.setMovie(movie);
    session.setRoom(room);
    session.setBookedSeats(List.of("A2", "A1"));

    HttpEntity<Session> entity = new HttpEntity<>(session, headers);

    ResponseEntity<Session> response1 = restTemplate.exchange("http://localhost:" + port + "/api/sessions", HttpMethod.POST, entity, Session.class);

    assertEquals(HttpStatus.BAD_REQUEST, response1.getStatusCode());
}
```

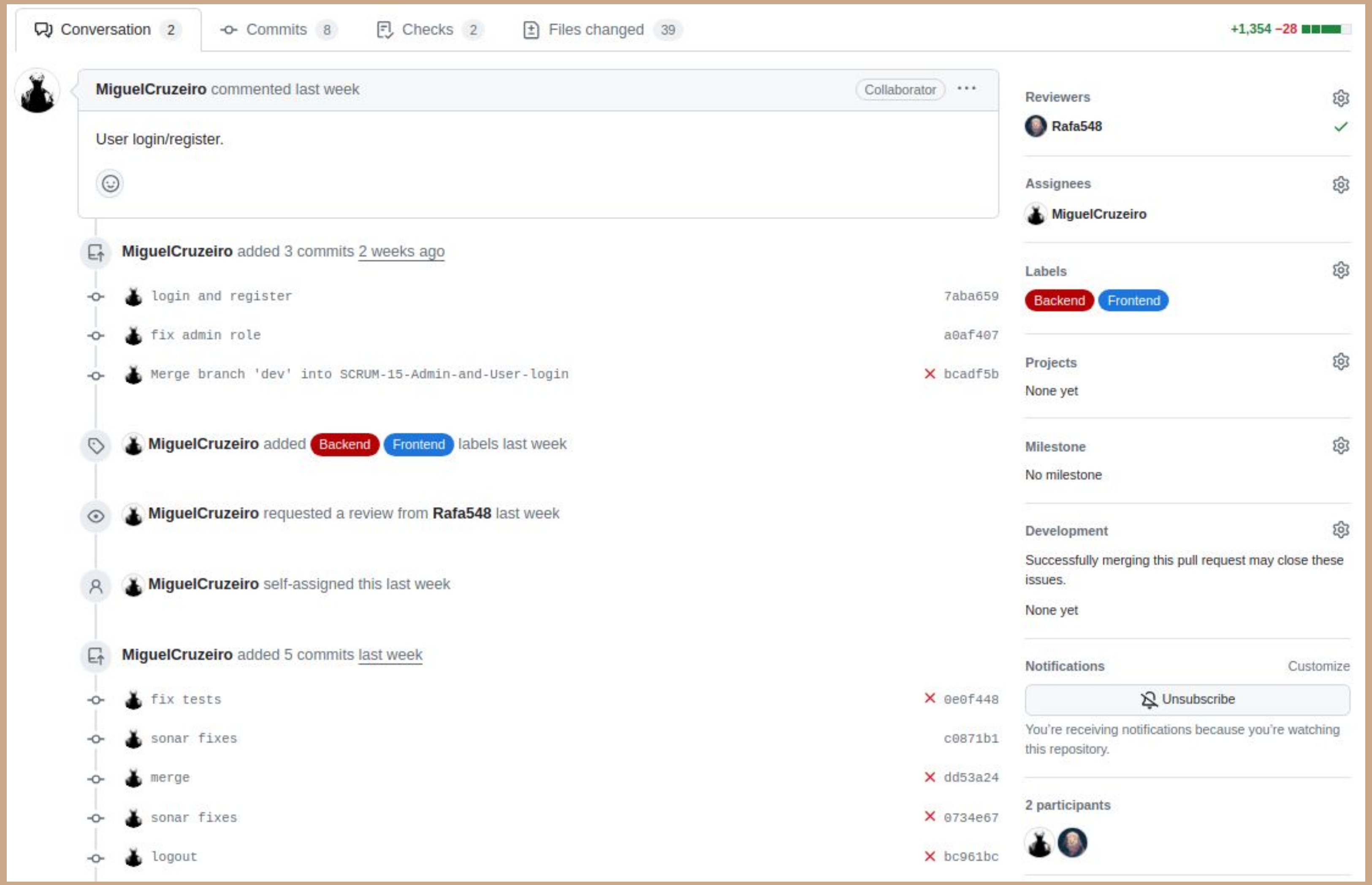


```java
@Test
@Order(8)
void whenCreateReservation_SeatAlreadyBooked() {
    HttpHeaders headers = new HttpHeaders();
    headers.set("Content-Type", "application/json");
    headers.setBearerAuth(jwtToken);

    HttpEntity<?> entity1 = new HttpEntity<>(headers);

    ResponseEntity<Session> response = restTemplate.exchange("http://localhost:" + port + "/api/sessions/1", HttpMethod.GET, entity1, Session.class);
    assertEquals(HttpStatus.OK, response.getStatusCode());
    Session session = response.getBody();

    ResponseEntity<AppUser> response2 = restTemplate.exchange("http://localhost:" + port + "/api/users/2", HttpMethod.GET, entity1, AppUser.class);
    assertEquals(HttpStatus.OK, response2.getStatusCode());
    AppUser user = response2.getBody();

    Reservation reservation = new Reservation();
    reservation.setPrice(10);
    reservation.setSeatNumbers(List.of("A1"));
    reservation.setSession(session);
    reservation.setUser(user);

    HttpEntity<Reservation> entity = new HttpEntity<>(reservation, headers);

    ResponseEntity<Reservation> response1 = restTemplate.exchange("http://localhost:" + port + "/api/reservations", HttpMethod.POST, entity, Reservation.class);

    assertEquals(HttpStatus.CONFLICT, response1.getStatusCode());
}
```

CI



```yaml
name: SonarCloud
on:
  push:
    branches:
      - dev
      - main
  pull_request:
    types: [opened, synchronize, reopened]
jobs:
  build:
    name: Build and analyze
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
        with:
          fetch-depth: 0
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: 17
          distribution: 'zulu'
      - name: Cache SonarCloud packages
        uses: actions/cache@v3
        with:
          path: ~/.sonar/cache
          key: ${{ runner.os }}-sonar
          restore-keys: ${{ runner.os }}-sonar
      - name: Cache Maven packages
        uses: actions/cache@v3
        with:
          path: ~/.m2
          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
          restore-keys: ${{ runner.os }}-m2
      - name: Build and analyze
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
        run: |
          cd backend/cinemax
          mvn -B '-Dtest=!deti.tqs.cinemax.frontend.**' verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar -Dsonar.projectKey=DiogoSilva1904_CineMax
```



Conversation 2 | Commits 8 | Checks 2 | Files changed 39 | +1,354 −28

**MiguelCruzeiro** commented last week — Collaborator

User login/register.

**MiguelCruzeiro** added 3 commits 2 weeks ago

```
login and register                                      7aba659
fix admin role                                          a0af407
Merge branch 'dev' into SCRUM-15-Admin-and-User-login   ✗ bcadf5b
```

**MiguelCruzeiro** added Backend Frontend labels last week

**MiguelCruzeiro** requested a review from **Rafa548** last week

**MiguelCruzeiro** self-assigned this last week

**MiguelCruzeiro** added 5 commits last week

```
fix tests      ✗ 0e0f448
sonar fixes      c0871b1
merge          ✗ dd53a24
sonar fixes    ✗ 0734e67
logout         ✗ bc961bc
```

Reviewers: Rafa548 ✓

Assignees: MiguelCruzeiro

Labels: Backend, Frontend

Projects: None yet

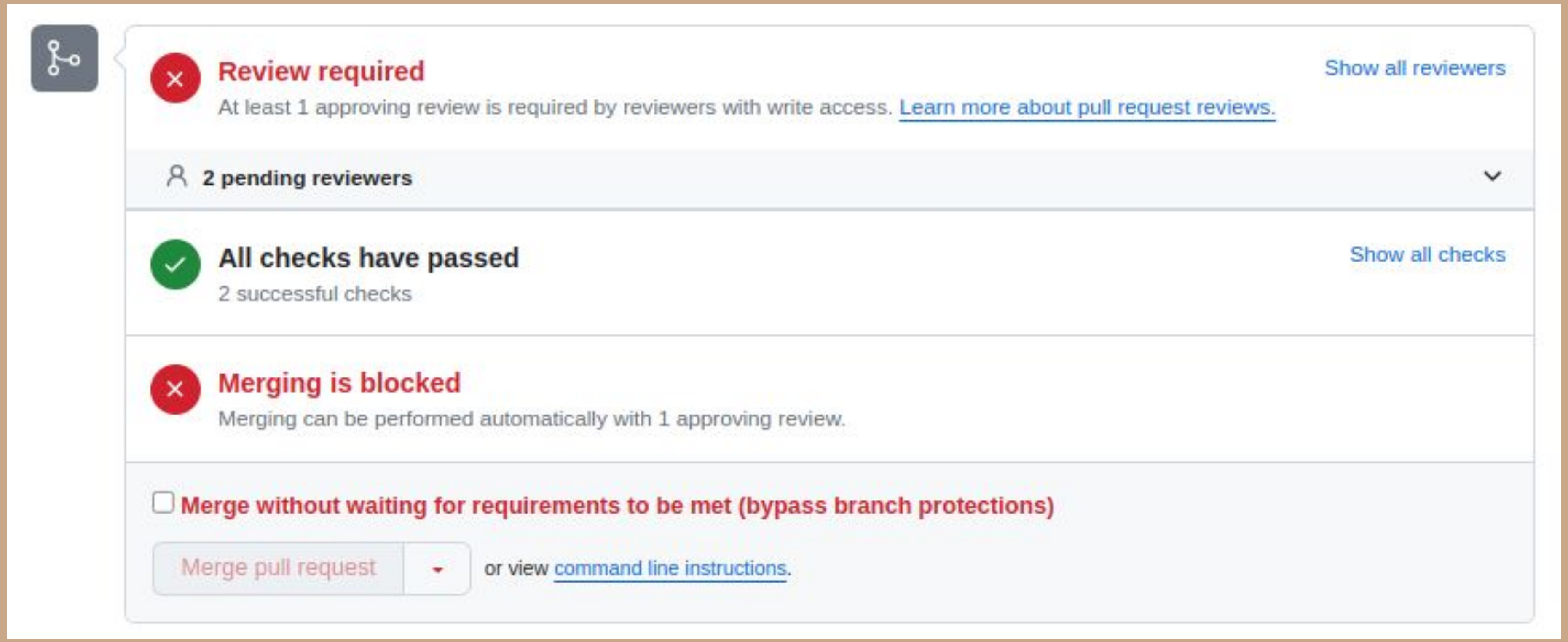
Milestone: No milestone

Development: Successfully merging this pull request may close these issues. None yet

Notifications: Customize — Unsubscribe

You're receiving notifications because you're watching this repository.

2 participants



**Review required** — Show all reviewers

At least 1 approving review is required by reviewers with write access. Learn more about pull request reviews.

**2 pending reviewers**

**All checks have passed** — Show all checks

2 successful checks

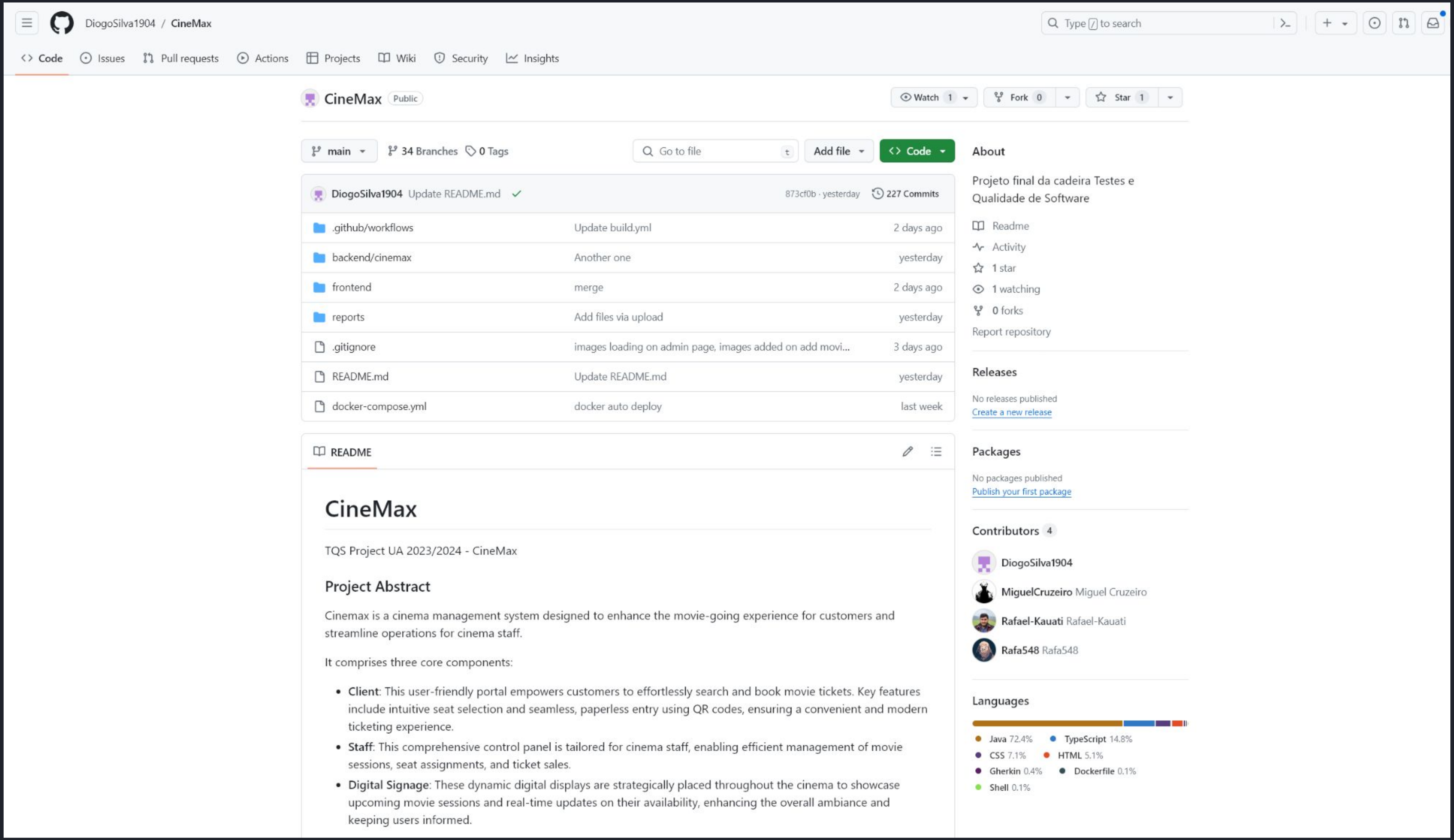
**Merging is blocked**

Merging can be performed automatically with 1 approving review.

☐ **Merge without waiting for requirements to be met (bypass branch protections)**

Merge pull request — or view command line instructions.

# Source/Version Control



DiogoSilva1904 Update README.md ✓ 873cf0b · yesterday 227 Commits

| | | |
|---|---|---|
| .github/workflows | Update build.yml | 2 days ago |
| backend/cinemax | Another one | yesterday |
| frontend | merge | 2 days ago |
| reports | Add files via upload | yesterday |
| .gitignore | images loading on admin page, images added on add movi... | 3 days ago |
| README.md | Update README.md | yesterday |
| docker-compose.yml | docker auto deploy | last week |

**About**

Projeto final da cadeira Testes e Qualidade de Software

Readme
Activity
1 star
1 watching
0 forks
Report repository

**Releases**

No releases published
Create a new release

**Packages**

No packages published
Publish your first package

**Contributors** 4

DiogoSilva1904
MiguelCruzeiro Miguel Cruzeiro
Rafael-Kauati Rafael-Kauati
Rafa548 Rafa548

**Languages**

● Java 72.4% ● TypeScript 14.8%
● CSS 7.1% ● HTML 5.1%
● Gherkin 0.4% ● Dockerfile 0.1%
● Shell 0.1%

README

CineMax

TQS Project UA 2023/2024 - CineMax

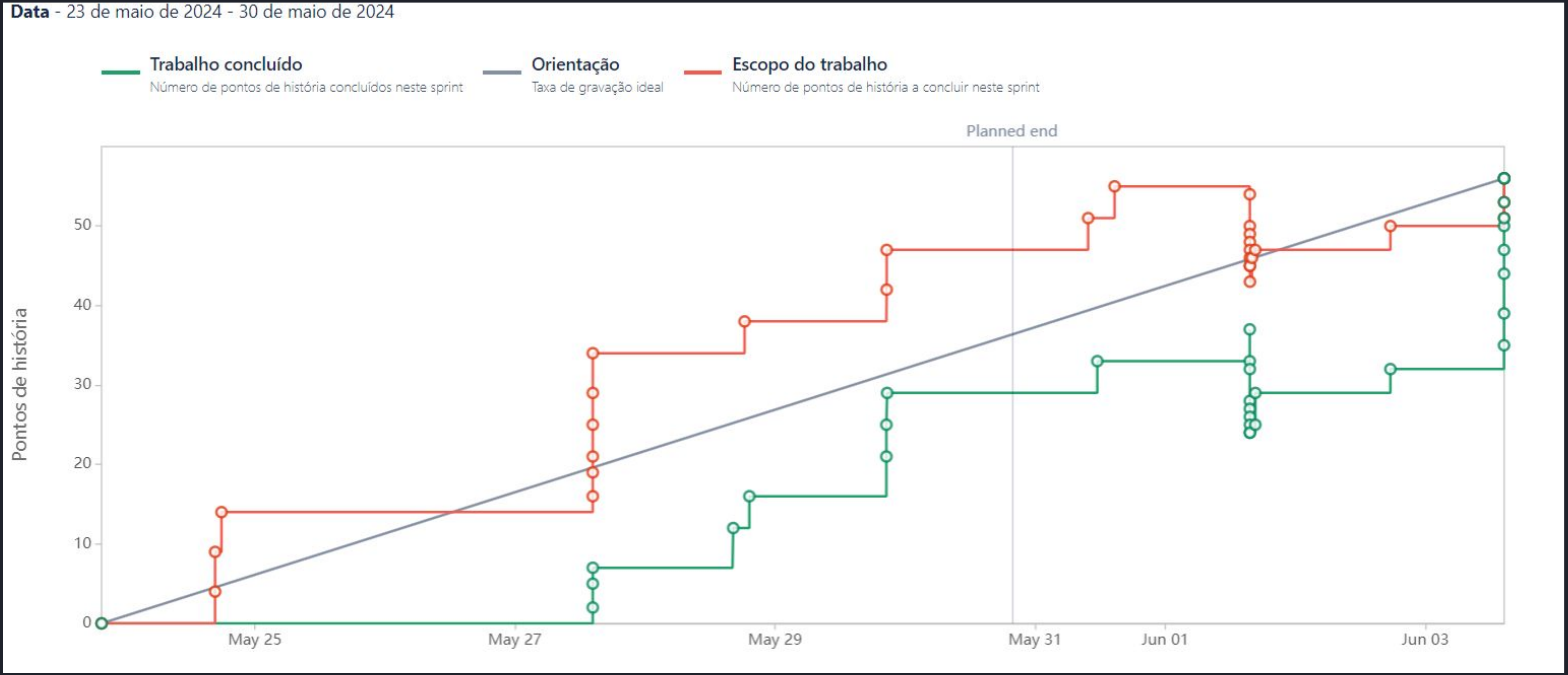
Project Abstract

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:

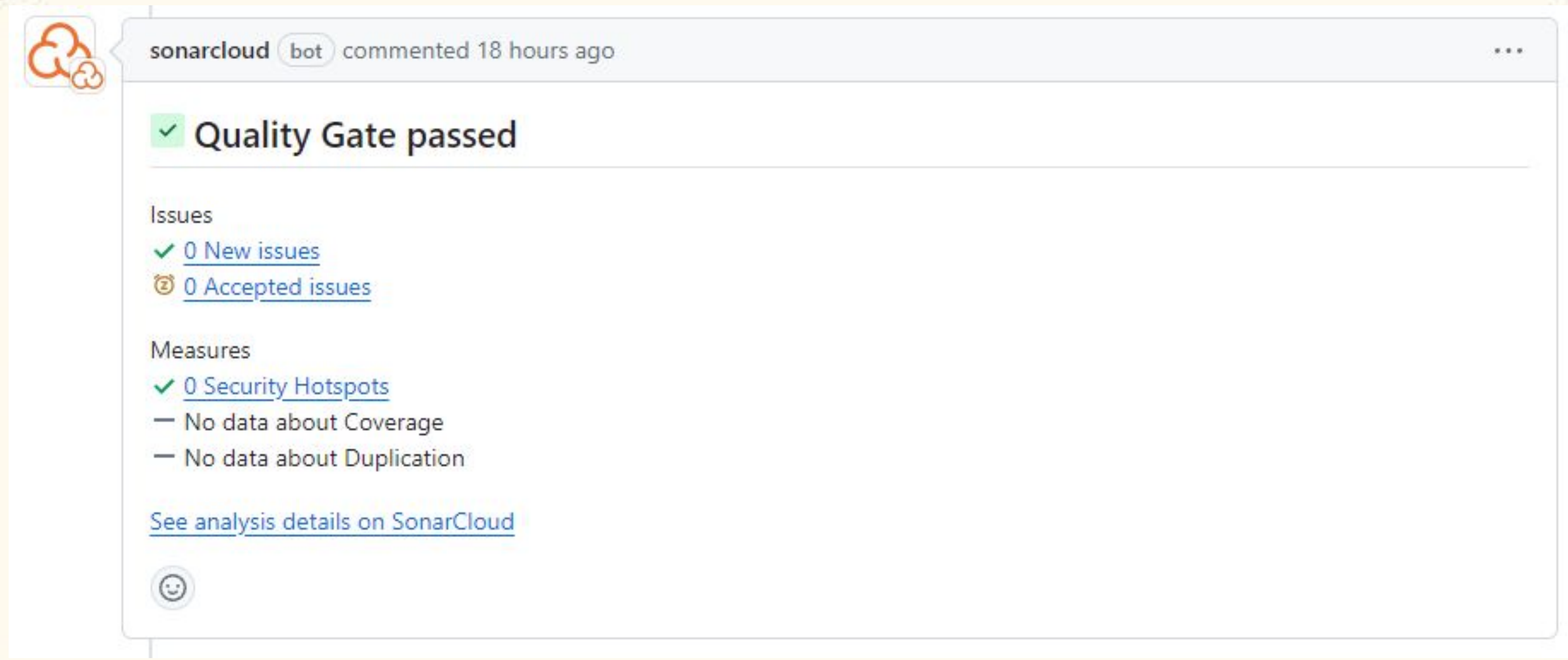
- **Client**: This user-friendly portal empowers customers to effortlessly search and book movie tickets. Key features include intuitive seat selection and seamless, paperless entry using QR codes, ensuring a convenient and modern ticketing experience.
- **Staff**: This comprehensive control panel is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

# Agile Project Management



**Data** - 23 de maio de 2024 - 30 de maio de 2024

## Code Quality



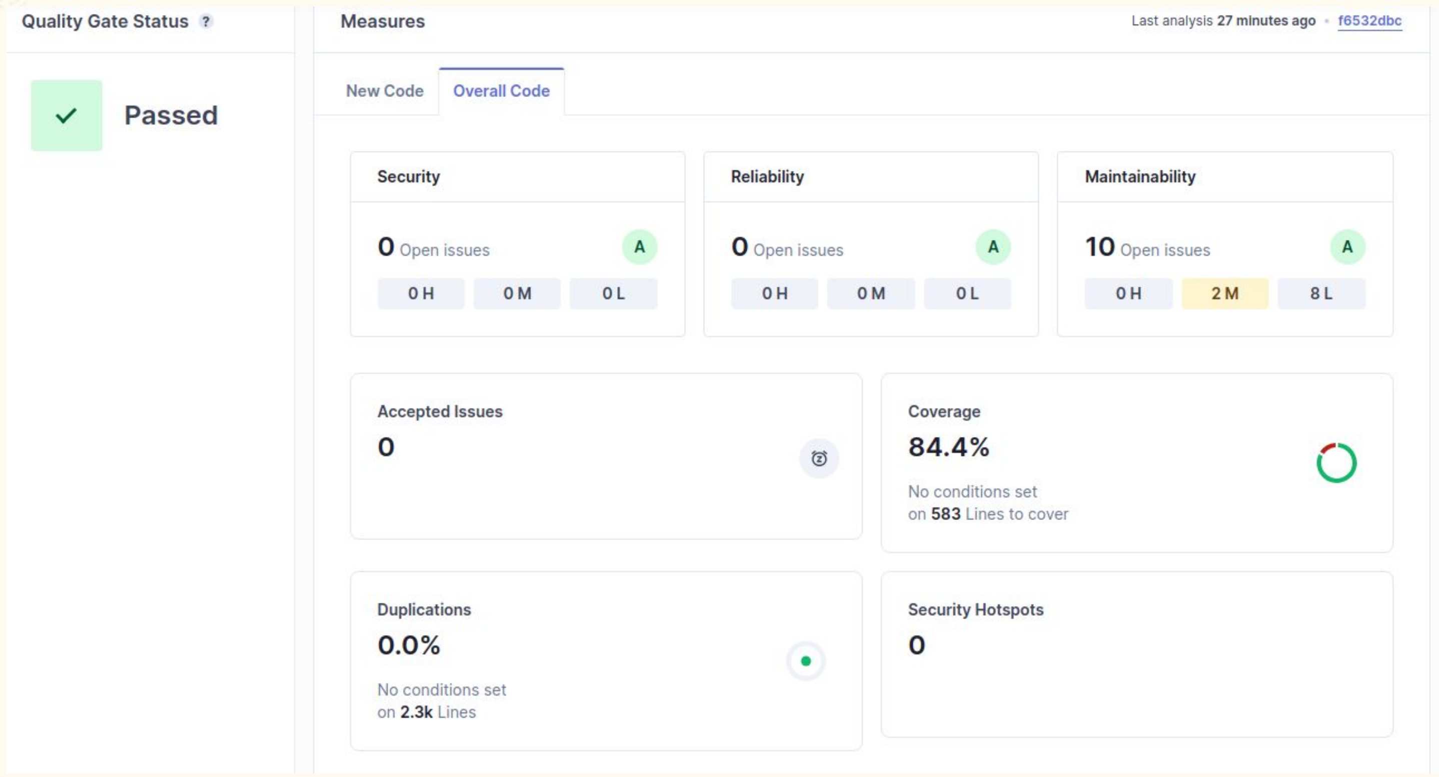
sonarcloud bot commented 18 hours ago

✓ Quality Gate passed

Issues
- ✓ 0 New issues
- 0 Accepted issues

Measures
- ✓ 0 Security Hotspots
- — No data about Coverage
- — No data about Duplication

See analysis details on SonarCloud



Quality Gate Status

✓ Passed

Measures

Last analysis **27 minutes ago** · f6532dbc

New Code | Overall Code

| Security | Reliability | Maintainability |
|---|---|---|
| 0 Open issues A | 0 Open issues A | 10 Open issues A |
| 0 H 0 M 0 L | 0 H 0 M 0 L | 0 H 2 M 8 L |

Accepted Issues
0

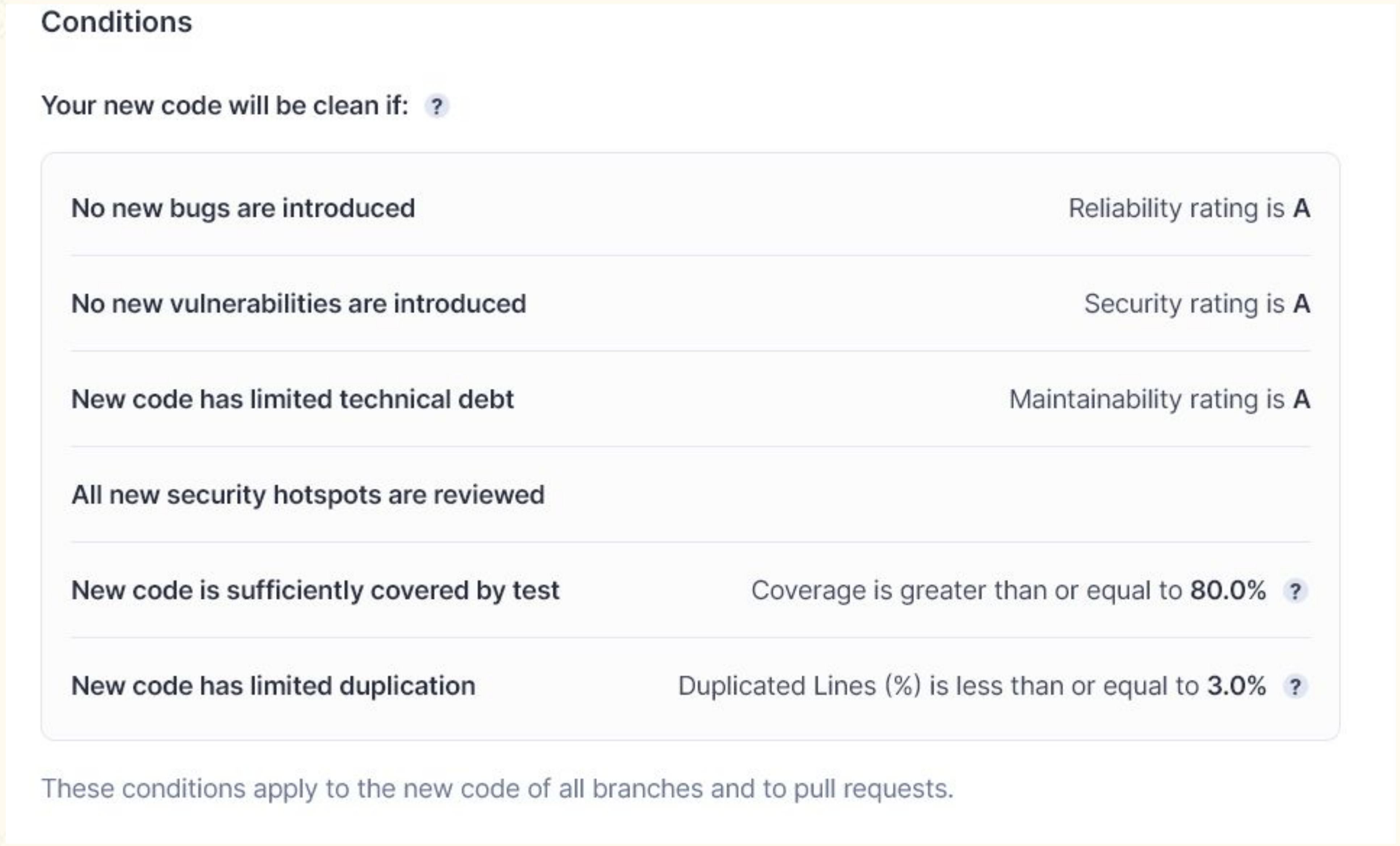
Coverage
84.4%
No conditions set
on **583** Lines to cover

Duplications
0.0%
No conditions set
on **2.3k** Lines

Security Hotspots
0

Automated evaluation

Detailed quality gate dashboard


Conditions

Your new code will be clean if: ?

| | |
|---|---|
| No new bugs are introduced | Reliability rating is **A** |
| No new vulnerabilities are introduced | Security rating is **A** |
| New code has limited technical debt | Maintainability rating is **A** |
| All new security hotspots are reviewed | |
| New code is sufficiently covered by test | Coverage is greater than or equal to **80.0%** ? |
| New code has limited duplication | Duplicated Lines (%) is less than or equal to **3.0%** ? |

These conditions apply to the new code of all branches and to pull requests.

## Quality Gate

## Demo

# Thank You