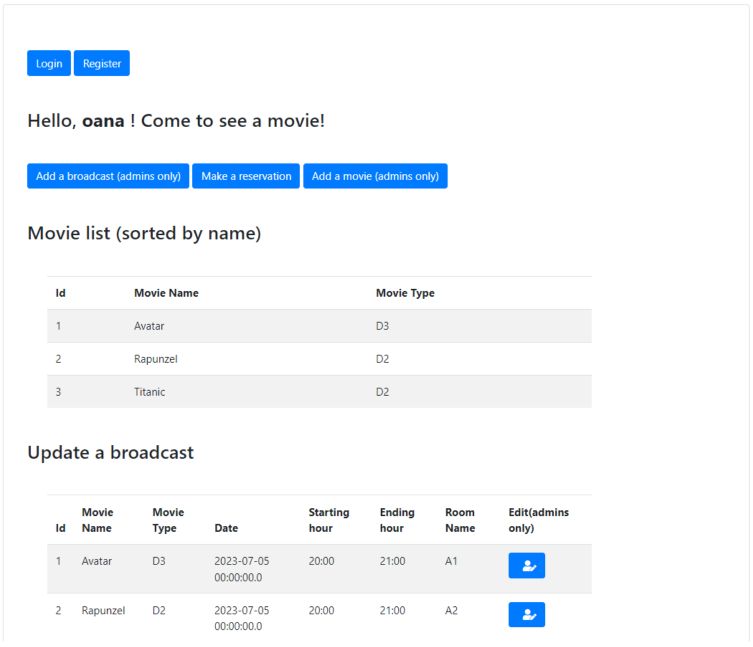
🎥 Platforma pentru management-ul rezervarilor la cinema



Hello, **oana** ! Come to see a movie!

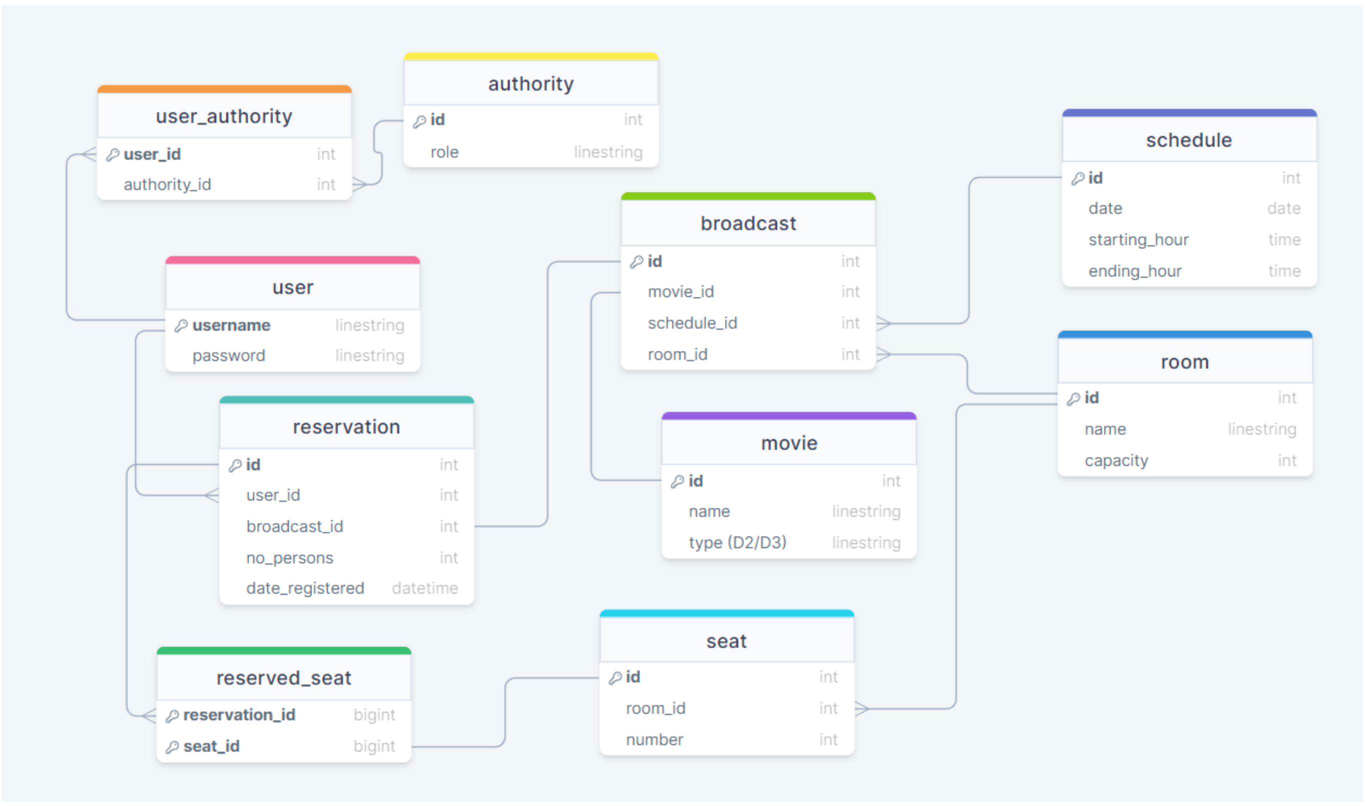
Movie list (sorted by name)

Id	Movie Name	Movie Type
1	Avatar	D3
2	Rapunzel	D2
3	Titanic	D2

Update a broadcast

Id	Movie Name	Movie Type	Date	Starting hour	Ending hour	Room Name	Edit(admins only)
1	Avatar	D3	2023-07-05 00:00:00.0	20:00	21:00	A1	
2	Rapunzel	D2	2023-07-05 00:00:00.0	20:00	21:00	A2	

📋 Baza de date - entitati



Aplicația va îndeplini cerințele următoare.

✅ Vor fi create relații între entități de toate tipurile: @OneToOne, @OneToMany, @ManyToOne, @ManyToMany.

@OneToOne

- Reservation - Broadcast

@OneToMany

- Movie - Broadcast
- Room - Seat
- Room - Broadcast
- Schedule - Broadcast

@ManyToOne

- Broadcast - Room
- Broadcast - Schedule
- Broadcast - Movie
- Reservation - User
- Seat - Room

@ManyToMany

- Reservation - Seat (reserved_seat)
- User - Authority (user_authority)

✅ Vor fi implementate toate tipurile de operații CRUD.

CREATE

```
- (Admins only)Adaugare film la lista de filme
- (Admins only) Adaugare programare film(broadcast) la o anumita data, ora, intr-o anumita sala
- Adugare user (inregistrare cu username si parola)
- Adaugare rezervare facuta de un user la un anumit film programat(broadcast)
```

UPDATE

```
- (Admins only) Modificarea unei programari a unui film (schimbare film, program, sala)
```

GET

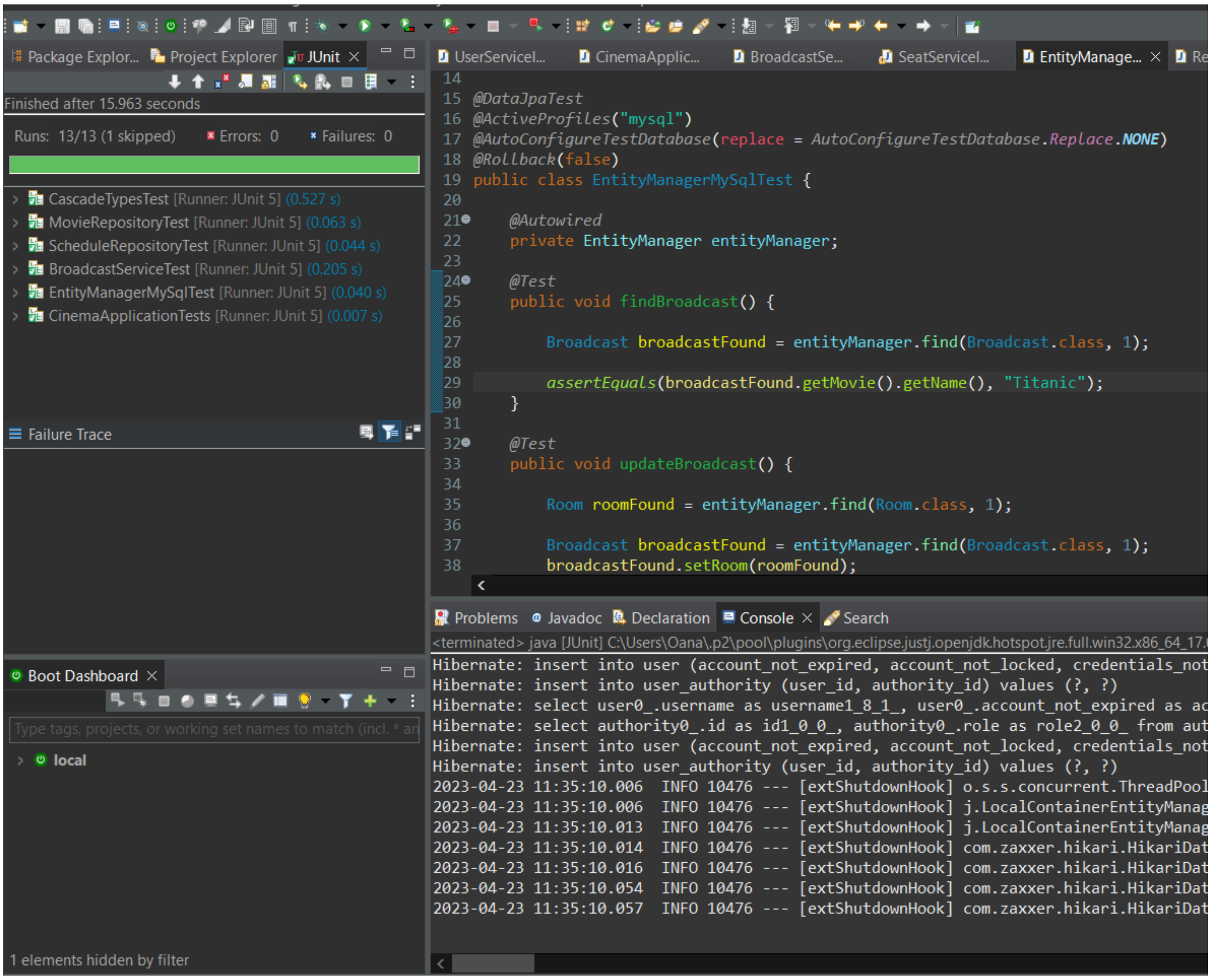
```
- Afisare lista filme pagina principala
- Afisare lista programari cu toate informatiile pagina principala
- Afisare lisat rezervari user curent pagina principala
- Alte informatii extra pe paginile secundare
```

DELETE

```
- Anularea unei rezervari
```

☑️ Se va testa aplicația folosindu-se profiluri și două baze de date diferite, una dintre ele pentru etapa de testare. Se poate utiliza și o bază de date in-memory (H2).

✅ Utilizare unit-tests/integration tests.



```
Finished after 15.963 seconds
Runs:  13/13 (1 skipped)     Errors:  0     Failures:  0

> CascadeTypesTest [Runner: JUnit 5] (0.527 s)
> MovieRepositoryTest [Runner: JUnit 5] (0.063 s)
> ScheduleRepositoryTest [Runner: JUnit 5] (0.044 s)
> BroadcastServiceTest [Runner: JUnit 5] (0.205 s)
> EntityManagerMySqlTest [Runner: JUnit 5] (0.040 s)
> CinemaApplicationTests [Runner: JUnit 5] (0.007 s)

1 elements hidden by filter
```

```java

@DataJpaTest
@ActiveProfiles("mysql")
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
@Rollback(false)
public class EntityManagerMySqlTest {

    @Autowired
    private EntityManager entityManager;

    @Test
    public void findBroadcast() {

        Broadcast broadcastFound = entityManager.find(Broadcast.class, 1);

        assertEquals(broadcastFound.getMovie().getName(), "Titanic");
    }

    @Test
    public void updateBroadcast() {

        Room roomFound = entityManager.find(Room.class, 1);

        Broadcast broadcastFound = entityManager.find(Broadcast.class, 1);
        broadcastFound.setRoom(roomFound);
```

```
<terminated> java [JUnit] C:\Users\Oana\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.
Hibernate: insert into user (account_not_expired, account_not_locked, credentials_not
Hibernate: insert into user_authority (user_id, authority_id) values (?, ?)
Hibernate: select user0_.username as username1_8_1_, user0_.account_not_expired as ac
Hibernate: select authority0_.id as id1_0_0_, authority0_.role as role2_0_0_ from aut
Hibernate: insert into user (account_not_expired, account_not_locked, credentials_not
Hibernate: insert into user_authority (user_id, authority_id) values (?, ?)
2023-04-23 11:35:10.006  INFO 10476 --- [extShutdownHook] o.s.s.concurrent.ThreadPool
2023-04-23 11:35:10.006  INFO 10476 --- [extShutdownHook] j.LocalContainerEntityManag
2023-04-23 11:35:10.013  INFO 10476 --- [extShutdownHook] j.LocalContainerEntityManag
2023-04-23 11:35:10.014  INFO 10476 --- [extShutdownHook] com.zaxxer.hikari.HikariDat
2023-04-23 11:35:10.016  INFO 10476 --- [extShutdownHook] com.zaxxer.hikari.HikariDat
2023-04-23 11:35:10.054  INFO 10476 --- [extShutdownHook] com.zaxxer.hikari.HikariDat
2023-04-23 11:35:10.057  INFO 10476 --- [extShutdownHook] com.zaxxer.hikari.HikariDat
```

✅ Se vor valida datele din formulare, se vor trata excepțiile.

- Validare date formular - la adaugarea unui film, validarea numelui

```java
public class Movie {

  ....

  @Size(min = 2, message = "*The name is too short!")
  @Size(max = 10, message = "*The name is too long (<=10 characters)!")
  private String name;
  ...

}
```

- Tratarea exceptiilor - daca vreau sa editez o programare (broadcast) care nu exista redirectionare catre notFoundException.html (mesajul: 404 not found)

```
public class BroadcastController {

    ....

    @GetMapping("/edit/{id}")
    public String showUpdateForm(@PathVariable("id") int id, Model model) {
        ...

        Optional<Broadcast> broadcast = broadcastRepository.findById(id);
        if (!broadcast.isPresent()) {

            throw new ResourceNotFoundException("Broadcast " + id + " not found");
        }

        ...
    }

    @ResponseStatus(HttpStatus.NOT_FOUND)
    @ExceptionHandler(ResourceNotFoundException.class)
    public ModelAndView handlerNotFoundException(Exception exception) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.getModel().put("exception", exception);
        modelAndView.setViewName("notFoundException");
        return modelAndView;
    }
    ...

}
```

✅ Se vor utiliza log-uri. Opțional aspecte.

```
2023-04-22 23:14:40.790  INFO 9848 --- [nio-8080-exec-6]
c.b.cinema.configuration.LoggingAspect   : aspect log after List
com.backend.cinema.services.ReservationServiceImpl.getAllReservationsByUsername(St
ring)
2023-04-22 23:14:40.790  INFO 9848 --- [nio-8080-exec-6]
c.b.cinema.configuration.LoggingAspect   : aspect log after ModelAndView
com.backend.cinema.controllers.MainController.getHome(Model)
```

```
@Component
@Aspect
@Slf4j
public class LoggingAspect {
    ...
    @After("logAnnotation()")
    public void logMethodCallAdvice(JoinPoint joinPoint) {
        log.info("aspect log after " + joinPoint.getSignature());
    }

    @Before("logPackage()")
    public void logPackageAdvice(JoinPoint joinPoint) {
        log.info("aspect log before " + joinPoint.getSignature());
    }

    @After("logMethod()")
    public void logSetterAdvice(JoinPoint joinPoint) {
        log.info("aspect log after " + joinPoint.getSignature());
        log.info("aspect log param " + joinPoint.getArgs()[0]);
    }
    ...
}
```

✅ Vor fi utilizate opțiuni de paginarea și sortarea a datelor.

- Afisarea filmelor in pagina principala ordonate alfabetic dupa nume

```
@Repository
public interface MovieRepository extends PagingAndSortingRepository <Movie,
Integer>{

}
```

```
@Controller
public class MainController {
    ...

    @Log
    @GetMapping("/main")
    public String showMainPage(Model model) {
        ...

        List<Movie> movies = new ArrayList<>();

movieRepository.findAll(Sort.by("name")).iterator().forEachRemaining(movies::add);
        model.addAttribute("movies", movies);

        ...
```

```
            return "main";
        }

        ...
    }
```

✅ Se va include Spring Security (cerința minima autentificare jdbc).

- Sistem de login si inregistrare

```
 <body>
    <div class="container" style="margin-top: 20px">
       <div class="col-md-6 col-md-offset-3">
          <form class="form-signin" method="post" th:action="perform_login">
             <h2 class="form-signin-heading">Sign in ....</h2>
             <p>
             <p th:if="${param.error}" class="text-danger">Invalid user or
password</p>
             <label for="username" class="sr-only">Username</label>
             <input type="text" id="username" name="username" class="form-
control" placeholder="Username" required autofocus>
             </p>
             <p>
                <label for="password" class="sr-only">Password</label>
                <input type="password" id="password" name="password"
class="form-control" placeholder="Password" required>
             </p>
             <button class="btn btn-lg btn-primary btn-block"
type="submit">Sign in</button>

             </br>

             <p> Don't have an account?
                  <a class="btn btn-primary" href="/users/signup"
>Register</a>
             </p>
          </form>
       </div>
    </div>
 </body>
```

- Permisiuni in functie de rolurile asociate
- Parola salvata criptata in baza de date

```
public class SecurityJpaConfig {

   ....
```

```
  @Bean
  PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }

  @Bean
  SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
    return http.authorizeRequests(auth ->
auth.antMatchers("/broadcasts").permitAll()

.antMatchers("/auction").hasAnyRole().antMatchers("/broadcasts/**").hasRole(
"ADMIN")

.antMatchers("/auction").hasAnyRole().antMatchers("/movies/**").hasRole("ADM
IN")

.antMatchers("/movies/add").hasRole("ADMIN").antMatchers("/login").permitAll
()

.antMatchers("/broadcasts/add").hasRole("ADMIN").antMatchers("/login").permi
tAll()
    // .anyRequest().authenticated()

).userDetailsService(userDetailsService).formLogin().loginPage("/login").log
inProcessingUrl("/perform_login")

.and().exceptionHandling().accessDeniedPage("/access_denied").and().httpBasi
c(withDefaults()).build();
  }

}
```