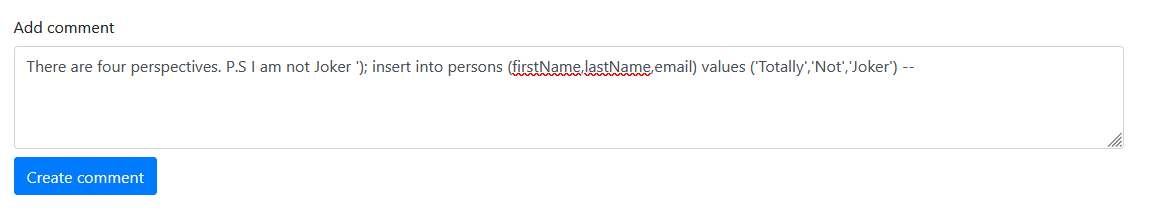
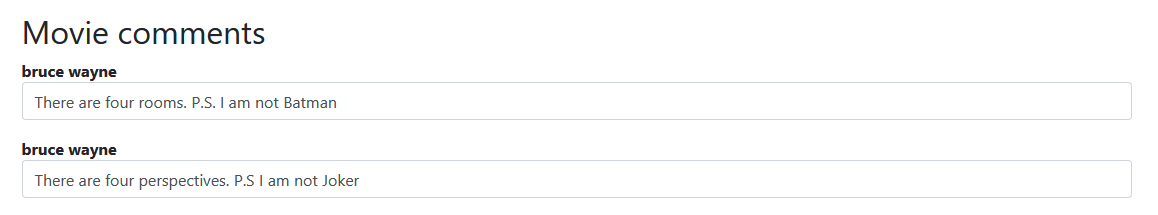
***SQL injection i Cross-site scripting***

*SQL injection -* Napad

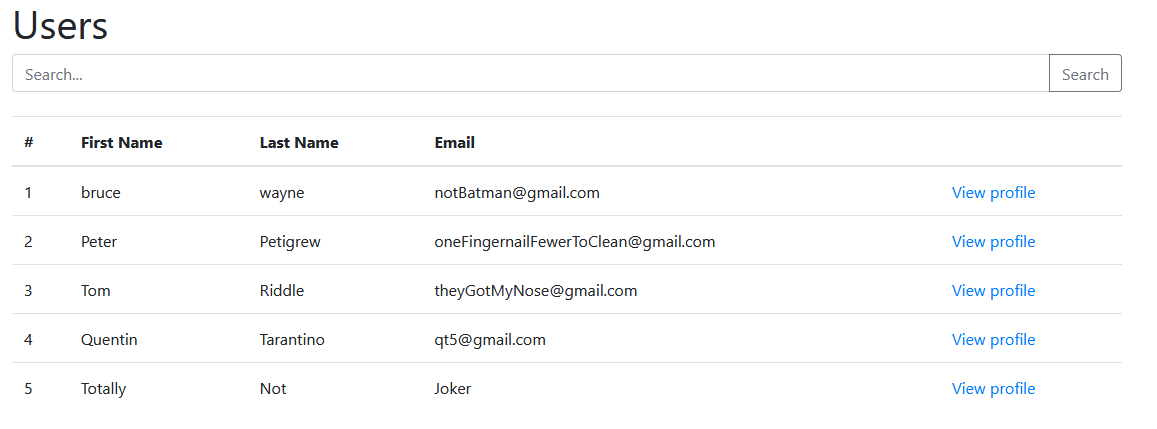
Testiram da li je polje za dodavanje komentara ranjivo na SQL Injection napad:



Komentar je uspešno dodat:



Kao i SQL Injectovani novi korisnik:

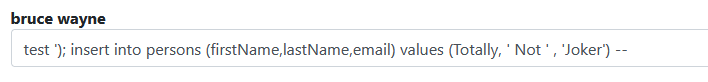


*SQL injection -* Odbrana

Dodavanjem narednog bloka koda, rešavamo problem

public void create(Comment comment) {  
 String query = "insert into comments(movieId, userId, comment) values (?,?,?)";  
  
 try (Connection connection = dataSource.getConnection();  
 PreparedStatement statement = connection.prepareStatement(query);  
 ) {  
 statement.setInt(1,comment.getMovieId());  
 statement.setInt(2, comment.getUserId());  
 statement.setString(3,comment.getComment());  
 statement.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
}

Pokušavamo opet istu akciju:



Ovog puta, linija

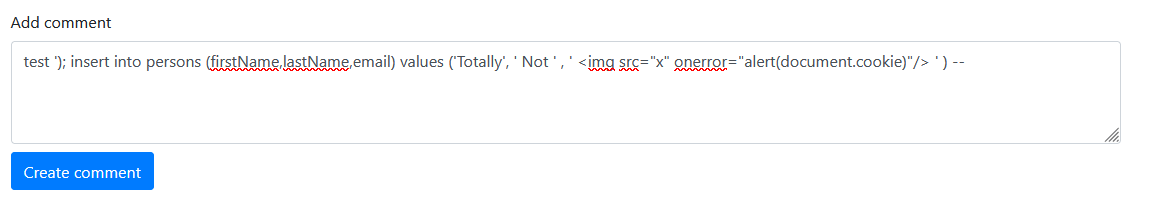
test '); insert into persons (firstName,lastName,email) values (Totally, ' Not ' , 'Joker') --

se parsira kao pravi komentar.

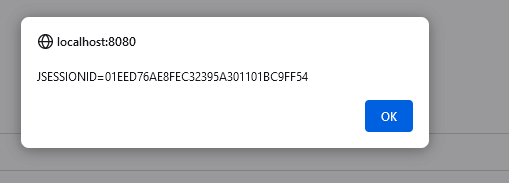
*Cross-site scripting - Napad*

Za jedan od parametara krišom dodatog korisnika, stavljamo skriptu za ispis njegovog kolačića sesije:

test '); insert into persons (firstName,lastName,email) values ('Totally', ' Not ' , ' <img src="x" onerror="alert(document.cookie)"/> ' ) --

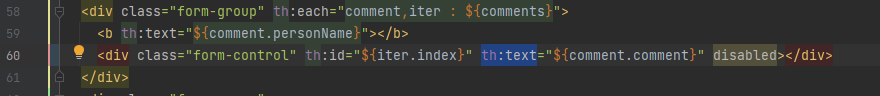


Kolačić se prikazuje:



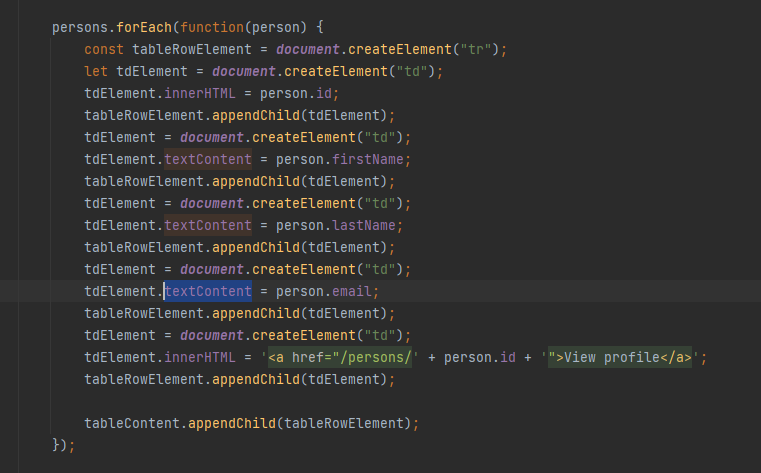
*Cross-site scripting - Odbrana*

**Za ispis pri dodavanju komentara**: (uskladišteni)



th:utext -> th:text

**Za ispis pri pretrazi**: (reflektovani)

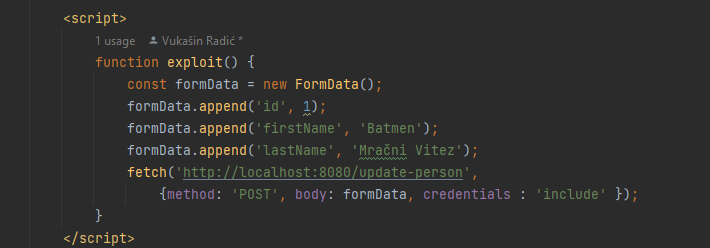


innerHTML -> textContent

## **Cross-site request forgery**

*Cross-site request forgery -* Napad

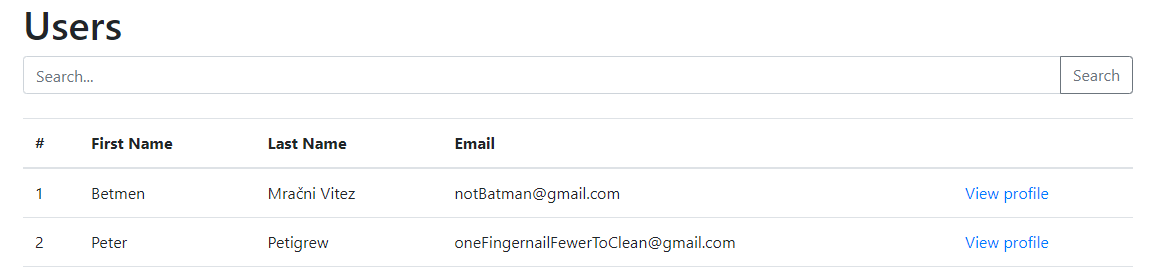
Potrebno je napisati skriptu kojom želimo prevariti korisnika i napraviti poziv ka endpointu koji menja podatke u bazi.



Pokrećemo aplikaciju na portu 3000



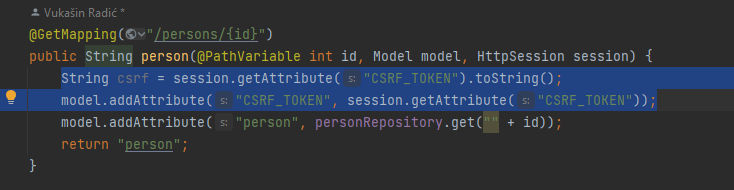
Maliciozni napad je prošao i dodat je Mračni Vitez u listu korisnika.



*Cross-site request forgery -* Odbrana

Korak 1

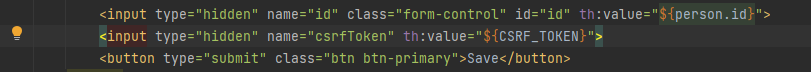
Nakon generisanja tokena I kreiranje sesije u koji se dodaje, čitamo ga I upisujemo.



Korak 2

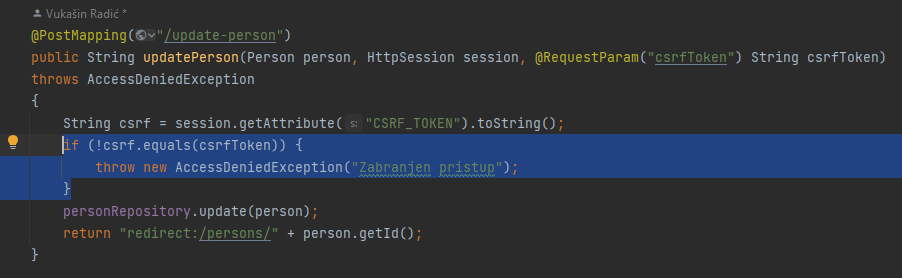
Želimo da korisnik svaki put kada šalje zahtev, on sadrži token koji ga verifikuje.

Ubacujemo ga kao skriveno polje forme:



Korak 3

Na serverskoj strani uporedjujemo token sesije sa onim u poslatoj formi:



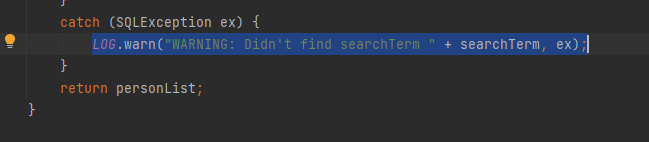
***DevOps***

### Rukovanje izuzecima i logovanje

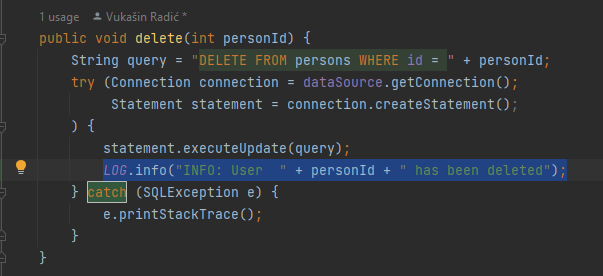
Dodajemo logove na sledećim mestima:

**Klasa PersonRepository**

Pretraga korisnika

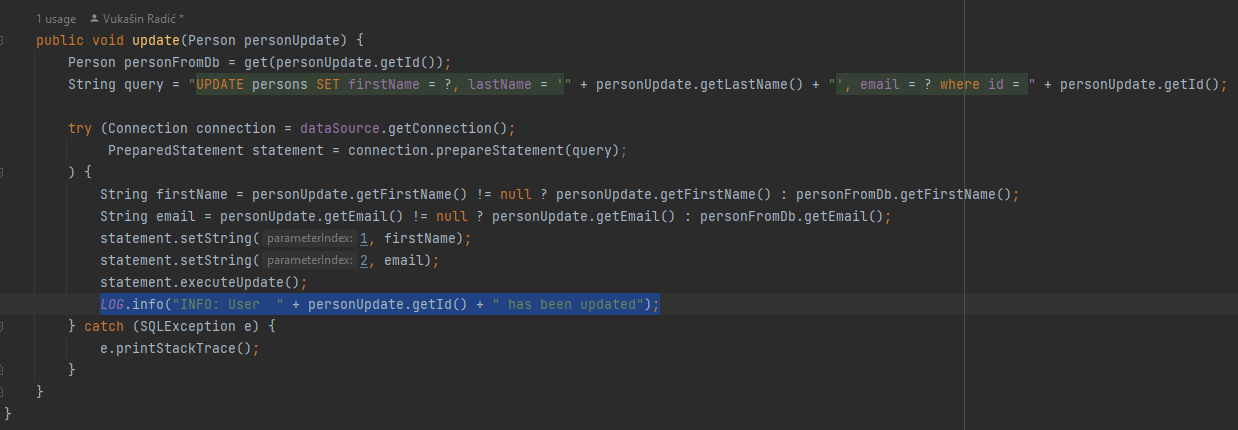
****

Brisanje korisnika





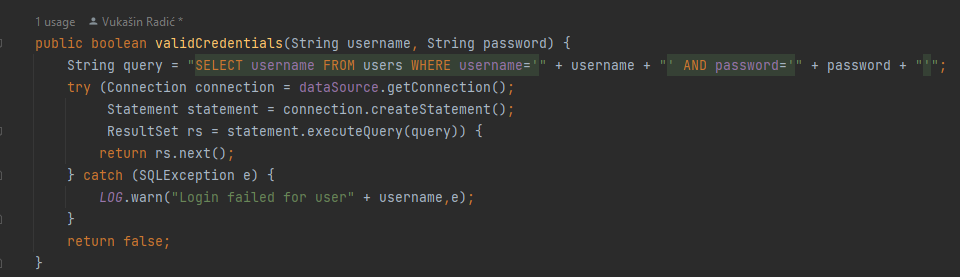
Ažuriranje korisnika



******

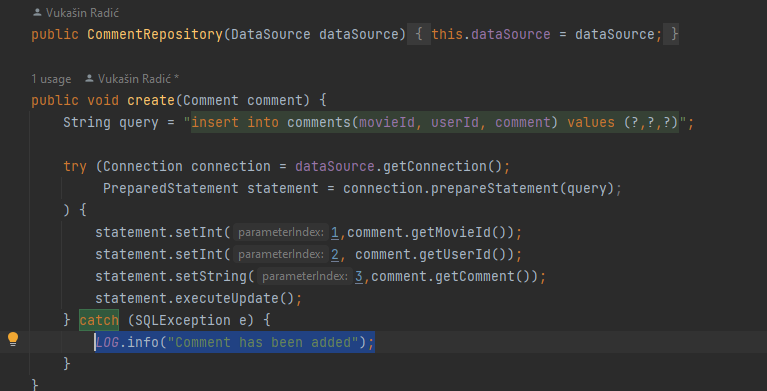
**Klasa UserRepository**

Neuspešni login



**Klasa CommentRepository**

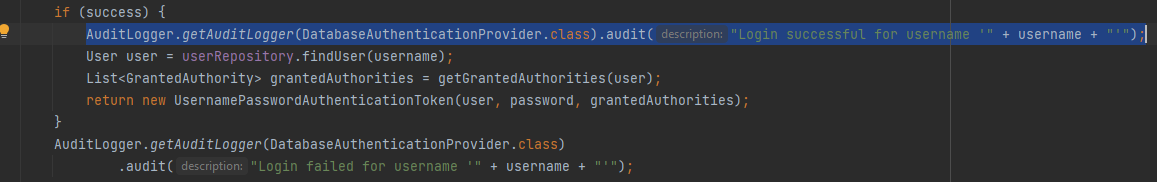
Dodavanje komentara

****

Auditing

Sigurnosno osetljive operacija koje želimo da “auditujemo” kako bi bile neporecive:

**Prijava korisnika**

****

**Brisanje korisnika**

****

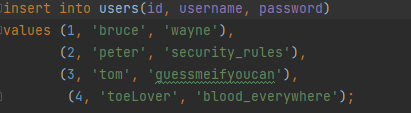
**Ažuriranje korisnika**

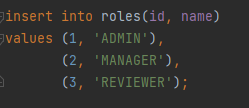
****

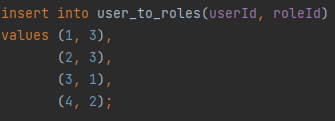
***Autorizacija***

1. **Za početak implementiramo autorizacioni model u bazi podataka:**

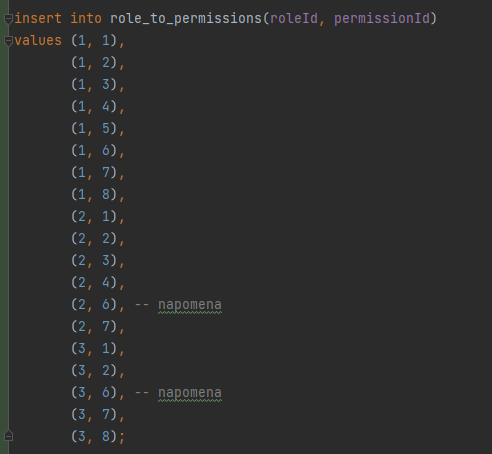
Korisnik **tom** ima rolu *ADMIN*, **toelover** rolu *MANAGER* i korisnici **bruce** i **peter** imaju rolu reviewer.

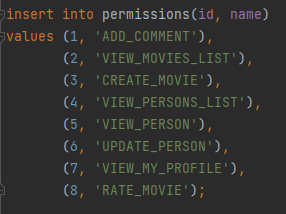






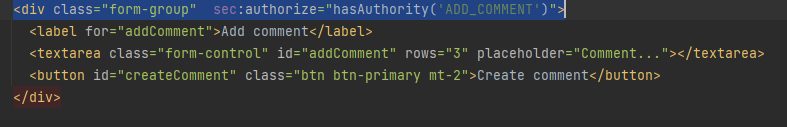
Iz matrice permisija:

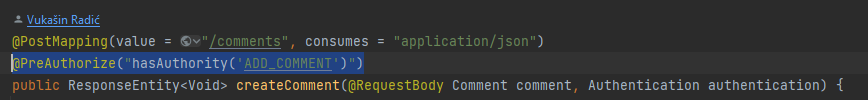




1. Permisije su u projektu već bile učitane
2. **Provera permisija na frontendu i backendu**

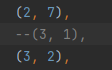
**Add comment:**

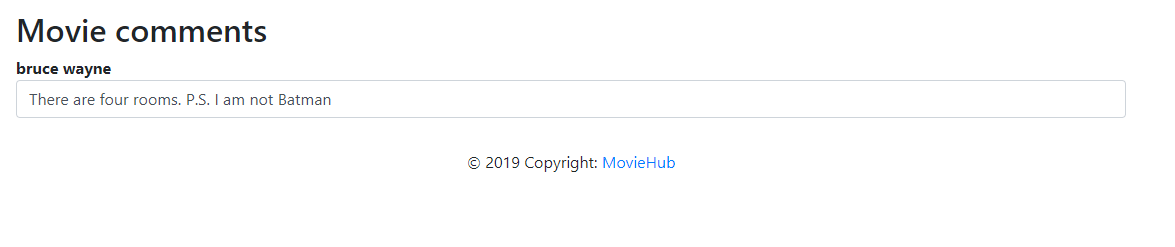
****

****

*Test:*

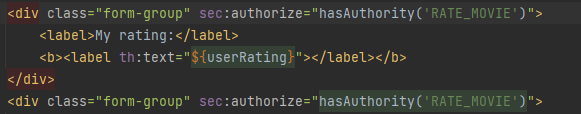
*Ukidamo permisiju da bismo videli kako bi izgledao prikaz stranice za eventualno novu rolu koju ne bi imala permisiju dodavanja komentara:*

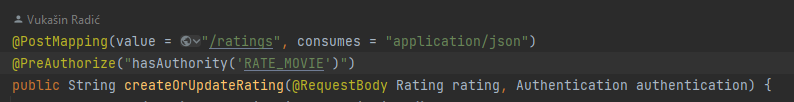
**

**

*Uspešno je implementirana permisija*

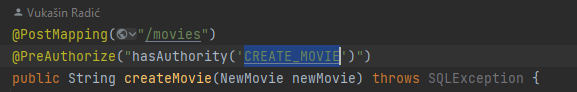
**Rate movie:**

****

****

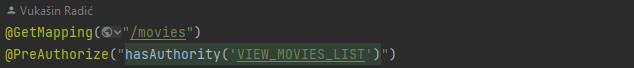
**Create movie:**

****

****

**View movies:**

****

****

Slično i za ostale...