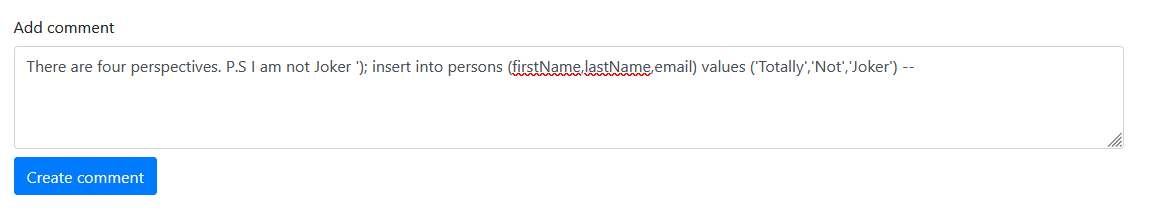
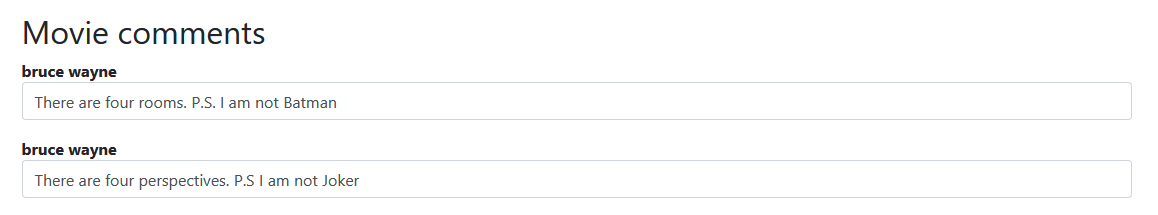
***SQL injection i Cross-site scripting***

*SQL injection -* Napad

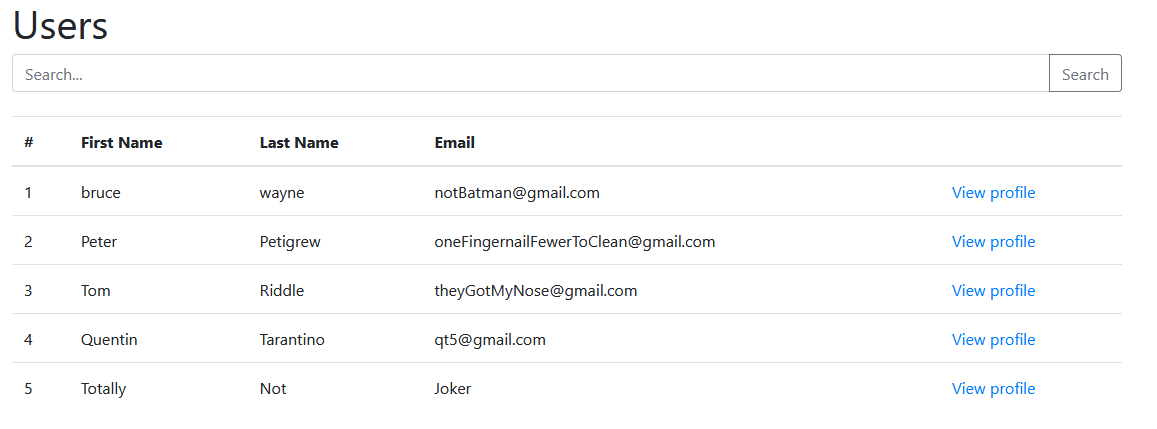
Testiram da li je polje za dodavanje komentara ranjivo na SQL Injection napad:



Komentar je uspešno dodat:



Kao i SQL Injectovani novi korisnik:

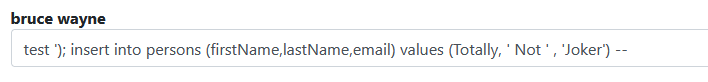


*SQL injection -* Odbrana

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

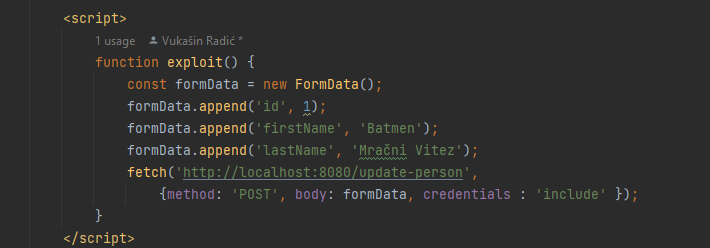
test '); insert into persons (firstName,lastName,email) values (Totally, ' Not ' , 'Joker') --

se parsira kao pravi komentar.

## **Cross-site request forgery**

*Cross-site request forgery -* Napad

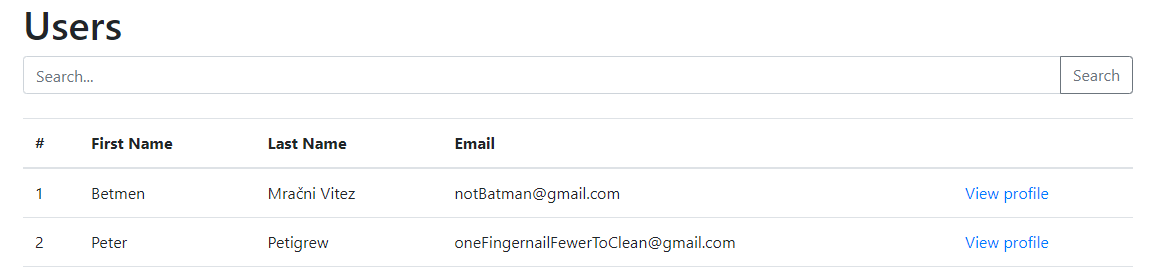
Potrebno je napisati skriptu kojom želimo prevariti korisnika i napraviti poziv ka endpointu koji menja podatke u bazi.



Pokrećemo aplikaciju na portu 3000



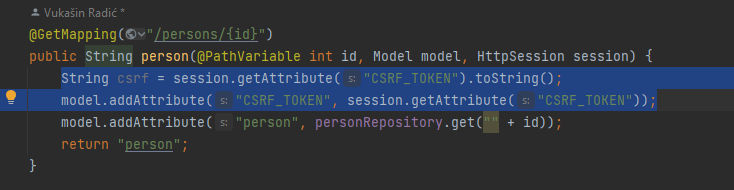
Maliciozni napad je prošao i dodat je Mračni Vitez u listu korisnika.



*Cross-site request forgery -* Odbrana

Korak 1

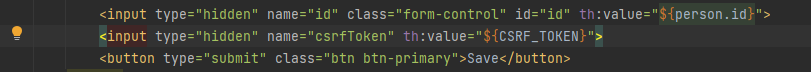
Nakon generisanja tokena I kreiranje sesije u koji se dodaje, čitamo ga I upisujemo.



Korak 2

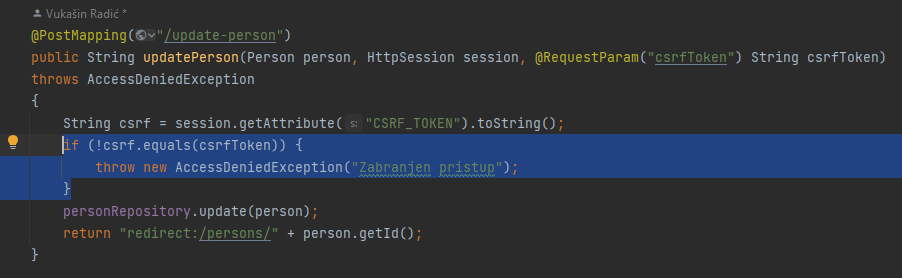
Želimo da korisnik svaki put kada šalje zahtev, on sadrži token koji ga verifikuje.

Ubacujemo ga kao skriveno polje forme:



Korak 3

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



***DevOps***

### Rukovanje izuzecima i logovanje

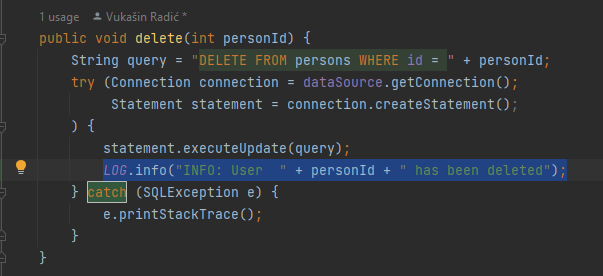
Dodajemo logove na sledećim mestima:

**Klasa PersonRepository**

Pretraga korisnika

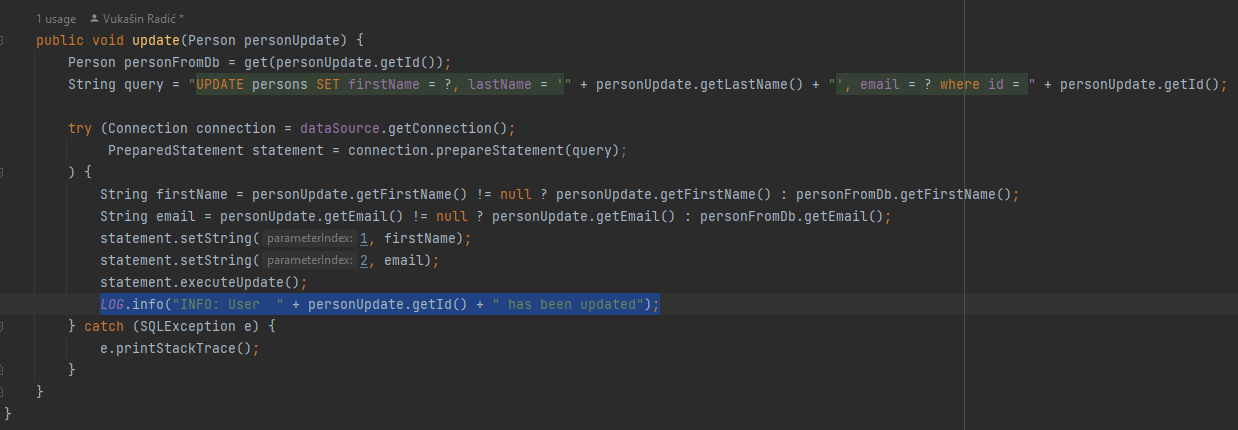
****

Brisanje korisnika





Ažuriranje kosnika



******