Academia Xideral Java Monterrey

Daniel Ivan Anaya Alvarez

Final Project

Introduction

For this final project, I have developed a web application for a trading card business store. The application allows the registration of users with the following details: first name, last name, age, email, number of normal cards, rare cards, and super rare cards. The main objective is to create a robust CRUD (Create, Read, Update, Delete) application using Spring Boot, JPA (Java Persistence API) Data, and Spring Security while also testing functionalities with JUnit and Mockito. The project also involves managing two related tables in a local SQL database.

Resources

- Java 17- Programming language for developing the application.
- Postman - Tool for testing and interacting with the REST API endpoints.
- Eclipse - Integrated Development Environment (IDE) used for writing and debugging code.
- SQL - Database language used to create and manage the local SQL database.

Objectives

- Utilize Spring JPA Data for data persistence and repository management.Spring Security
- Implement Spring Security to handle user authentication and authorization.
- Write and execute tests with JUnit and Mockito to ensure code reliability.
- Operate with two tables in a local SQL database to manage user and card data.

Database Structure

Table 1: user_card

This table represents the users registered in the trading card store. It stores all the personal details of the users, including their names, age, email, and the number of different types of cards they own.

id	first_name	last_name	age	email	normal_cards	rare_cards	superrare_cards
1	Jose	Villarreal	22	josevillarreal@cards.com	33	45	4
2	Jack	Bauer	44	jackbauer@cards.com	23	42	2
3	Joey	Wheeler	17	Joeywheeler@cards.com	2	24	25
4	Leon	Kennedy	25	LeonKennedy@cards.com	99	22	23
5	Ada	Wong	25	Adawong@cards.com	50	23	2
6	Connor	Mcgregor	26	ConnorMcgregor@cards.com	1	2	3
7	John	Doe	30	johndoe@cards.com	10	5	1
8	Emily	Clark	28	emilyclark@cards.com	15	8	0
9	Michael	Smith	35	michaelsmith@cards.com	12	9	3
10	Sarah	Johnson	22	sarahjohnson@cards.com	8	6	2
11	David	Brown	45	davidbrown@cards.com	20	10	4
12	Sophia	Wilson	19	sophiawilson@cards.com	5	3	1
13	James	Taylor	32	jamestaylor@cards.com	25	14	7
14	Olivia	Miller	27	oliviamiller@cards.com	18	12	5
15	Daniel	Davis	40	danieldavis@cards.com	9	4	3

Table 1 user_Card

Table 2: cards

The second table, named cards, contains the details of the different types of cards available in the store. It includes the card type, market price, and a brief description of each card.

id	name	type	price	description
1	Fire Dragon	SuperRare	150.00	A powerful dragon card with fiery breath.
2	Water Spirit	Rare	75.00	A mystical water entity with healing powers.
3	Earth Golem	Normal	20.00	A basic earth creature card with high defense.
4	Thunder Phoenix	SuperRare	200.00	A legendary bird with thunder powers.
5	Shadow Assassin	Rare	90.00	A stealthy and dangerous assassin card.
6	Nature Elf	Normal	15.00	A simple but agile forest elf.
7	Frost Giant	SuperRare	180.00	A massive giant with freezing attacks.
8	Fire Mage	Rare	85.00	A wizard with powerful fire spells.
9	Wind Warrior	Normal	25.00	A warrior with fast, wind-powered attacks.
10	Dark Knight	SuperRare	210.00	A knight with dark, forbidden powers.
11	Light Sorceress	Rare	95.00	A sorceress with light magic to heal and attack.
12	Stone Guardian	Normal	18.00	A sturdy guardian made of rock.
13	Mystic Fairy	SuperRare	160.00	A fairy with mysterious magical abilities.
14	Goblin Archer	Normal	12.00	A quick but weak goblin with a bow.
15	Vampire Lord	SuperRare	250.00	A powerful vampire that drains life from oppo...

Table 2 cards

Core Features Implemented

1. **CRUD Operations:**
   Implemented using Spring JPA Data to interact with the database tables (user_card and cards). Each table is represented by its corresponding JPA entity class.

2. **REST API Endpoints:**
   - Developed controllers (UsercardsRestController and CardRestController) to manage user and card data through HTTP methods (GET, POST, PUT, DELETE).
   - Endpoints secured with Spring Security annotations (@PreAuthorize).

3. **Security Configuration:**

- Configured Spring Security to handle user authentication and authorization.
- Implemented in-memory user details service to manage roles (USER and ADMIN).

4. **Testing:**
    - Wrote unit tests using JUnit and Mockito to ensure the reliability of the service layer.
    - Tested REST endpoints using Postman to validate API behavior and security restrictions.

Challenges Encountered

- Configuring Spring Security for proper authentication and authorization, particularly for handling roles and securing specific endpoints.
- Troubleshooting 401 Unauthorized errors when making POST and DELETE requests via Postman due to incorrect configurations or missing headers.

Conclusion

This project allowed me to gain hands-on experience in building a full-stack web application using the Spring Boot framework, particularly focusing on data persistence with Spring JPA and securing the application with Spring Security. The integration of testing frameworks such as JUnit and Mockito was instrumental in ensuring the application's reliability. Although some challenges were encountered, such as setting up authentication and authorization, these difficulties provided valuable learning opportunities and deepened my understanding of the Spring ecosystem.

Appendix (Code)

User_data.sql

```
CREATE DATABASE  IF NOT EXISTS users_cards;
USE users_cards;

Drop table if exists user_card;

CREATE TABLE user_card (
  id int NOT NULL AUTO_INCREMENT,
  first_name varchar(45) DEFAULT NULL,
  last_name varchar(45) DEFAULT NULL,
  age int(100) DEFAULT NULL,
  email varchar(45) DEFAULT NULL,
  normal_cards int(100) DEFAULT NULL,
  rare_cards int(100) DEFAULT NULL,
  superrare_cards int(100) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;

-- Data for table employee

INSERT INTO user_card VALUES
        (1,'Jose','Villarreal',22,'josevillarreal@cards.com', 33, 45, 4),
        (2,'Jack','Bauer',44,'jackbauer@cards.com',23,42,2),
```

```
    (3,'Joey','Wheeler',17,'Joeywheeler@cards.com',2,24,25),
    (4,'Leon','Kennedy',25,'LeonKennedy@cards.com', 99, 22, 23),
    (5,'Ada','Wong',25 ,'Adawong@cards.com', 50,23,2),
    (0,'Connor','Mcgregor',26 ,'ConnorMcgregor@cards.com', 1,2,3);


INSERT INTO `user_card` VALUES
  (6,'John','Doe',30,'johndoe@cards.com', 10, 5, 1),
  (7,'Emily','Clark',28,'emilyclark@cards.com', 15, 8, 0),
  (8,'Michael','Smith',35,'michaelsmith@cards.com', 12, 9, 3),
  (9,'Sarah','Johnson',22,'sarahjohnson@cards.com', 8, 6, 2),
  (10,'David','Brown',45,'davidbrown@cards.com', 20, 10, 4),
  (11,'Sophia','Wilson',19,'sophiawilson@cards.com', 5, 3, 1),
  (12,'James','Taylor',32,'jamestaylor@cards.com', 25, 14, 7),
  (13,'Olivia','Miller',27,'oliviamiller@cards.com', 18, 12, 5),
  (14,'Daniel','Davis',40,'danieldavis@cards.com', 9, 4, 3),
  (15,'Liam','Martinez',29,'liammartinez@cards.com', 7, 5, 6);
```