Xideral Java Academy

TRADE CARD SHOP

CRUD implementation with Spring and Spring Security for Java application Including Junit and Mockito

By Daniel Ivan Anaya Alvarez

Objectives of the presentation

- Introduce the Business Context
- Code Implementation Overview
- Testing with JUnit and Mockito
- Future Enhancements

Business Model

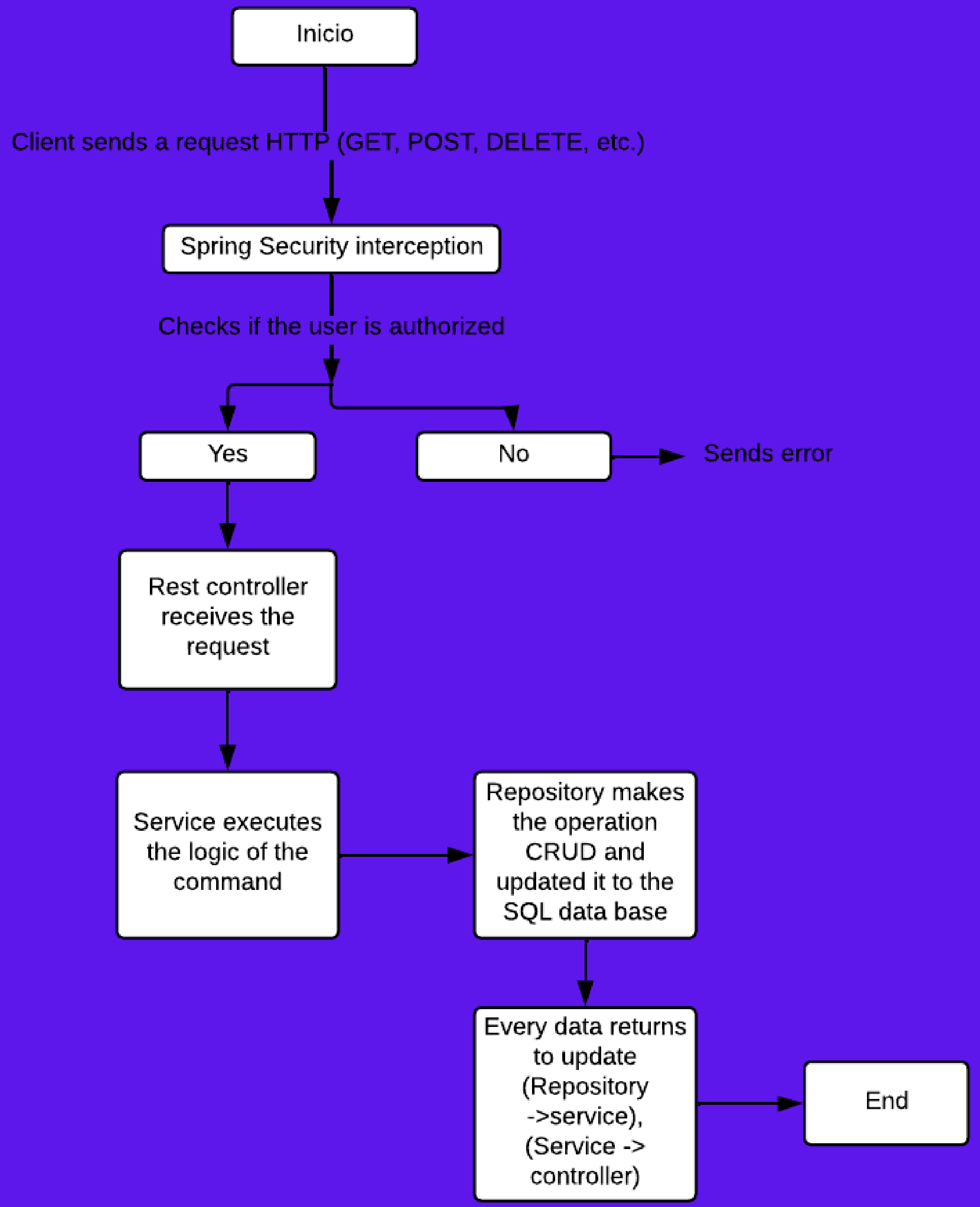
Online platform where users can register

Different types or "rarities," such as Normal, Rare, and Super Rare

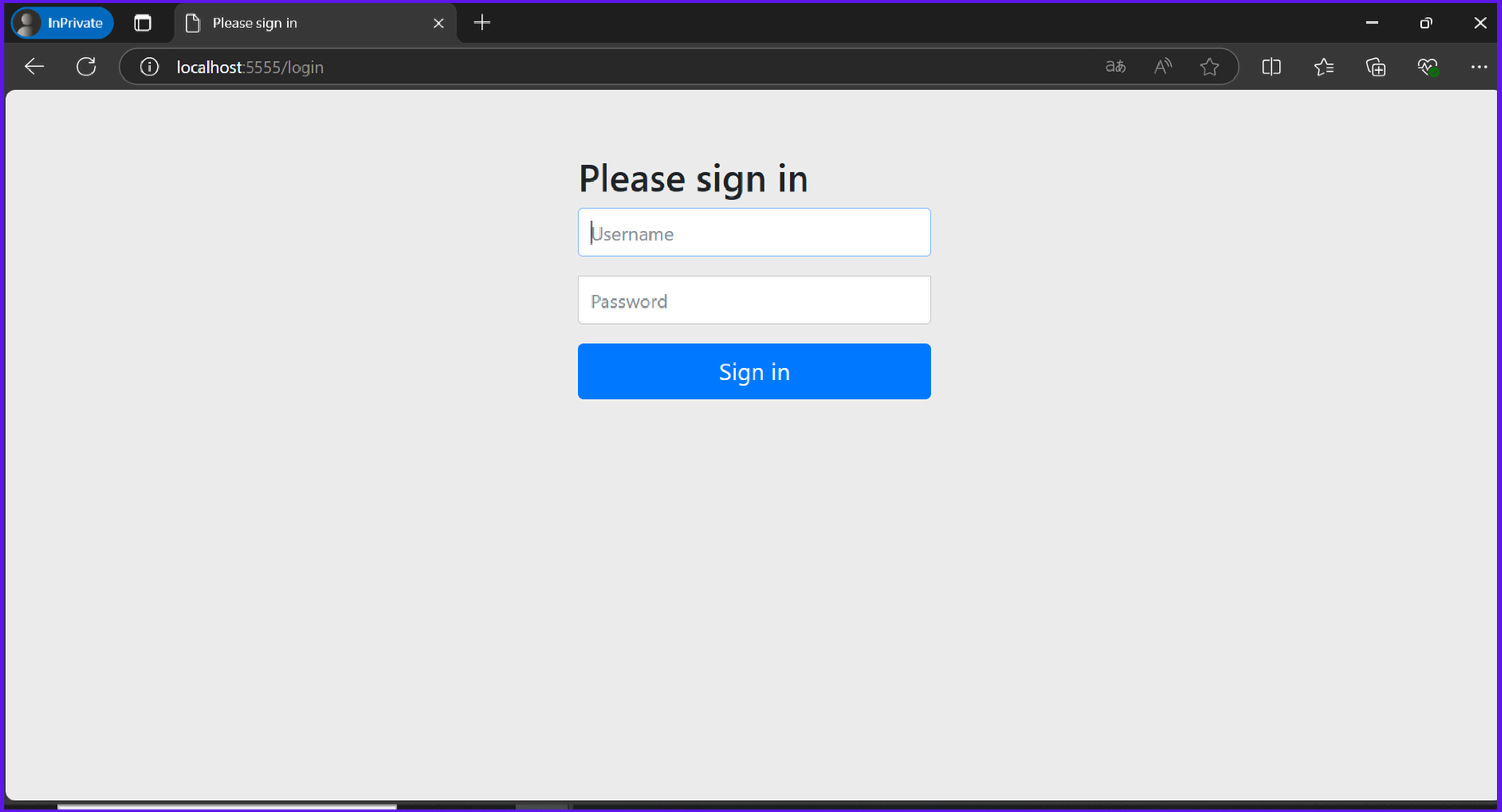
Each card type has its own unique features and value

Players can collect, trade, and use these cards to compete in the game, with rarer cards generally providing better benefits or more powerful abilities

Flowchart how does the code works

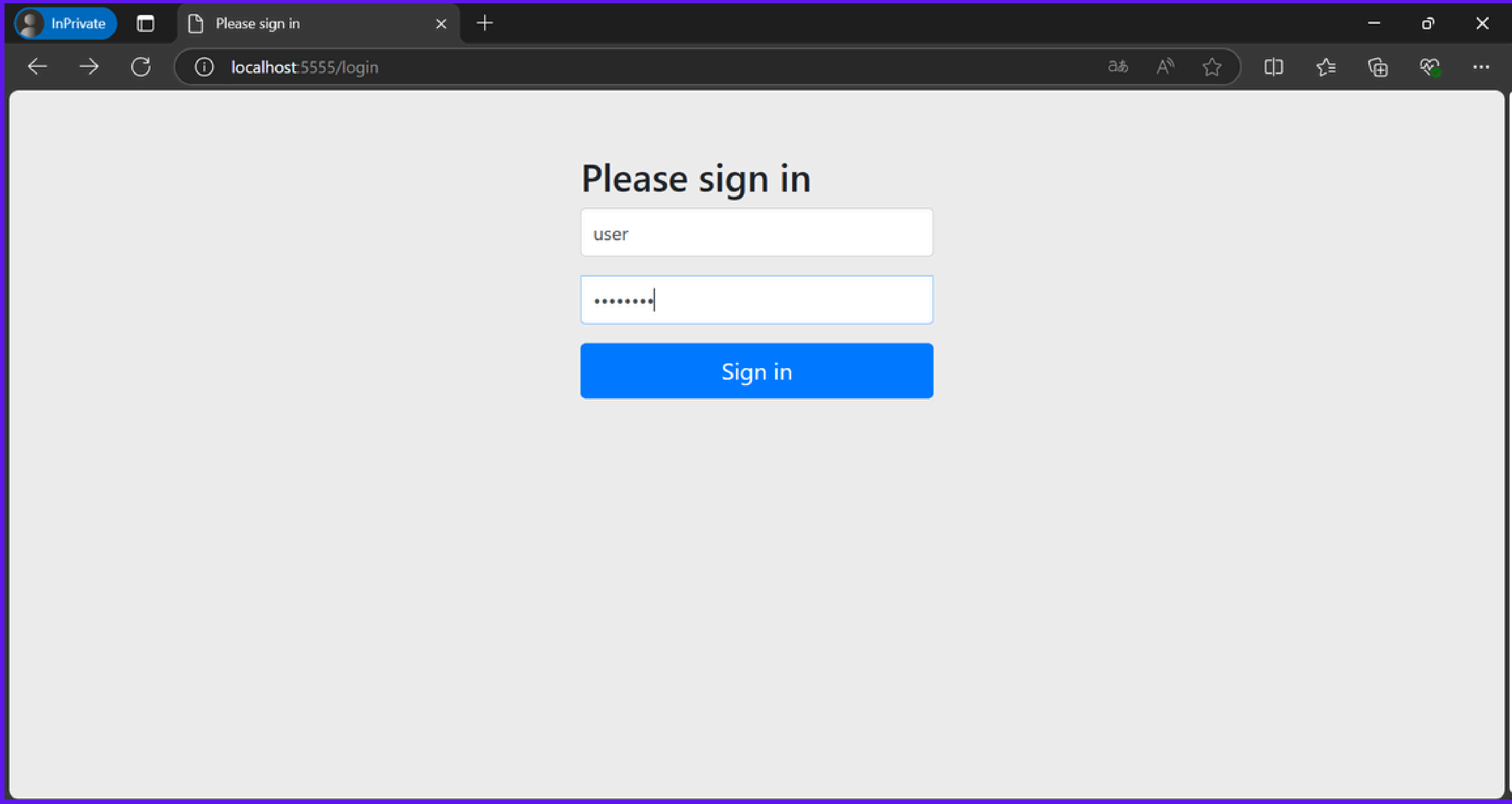

Inicio
Client sends a request HTTP (GET, POST, DELETE, etc.)
Spring Security interception
Checks if the user is authorized
Yes
No
Sends error
Rest controller receives the request
Service executes the logic of the command
Repository makes the operation CRUD and updated it to the SQL data base
Every data returns to update (Repository ->service), (Service -> controller)
End

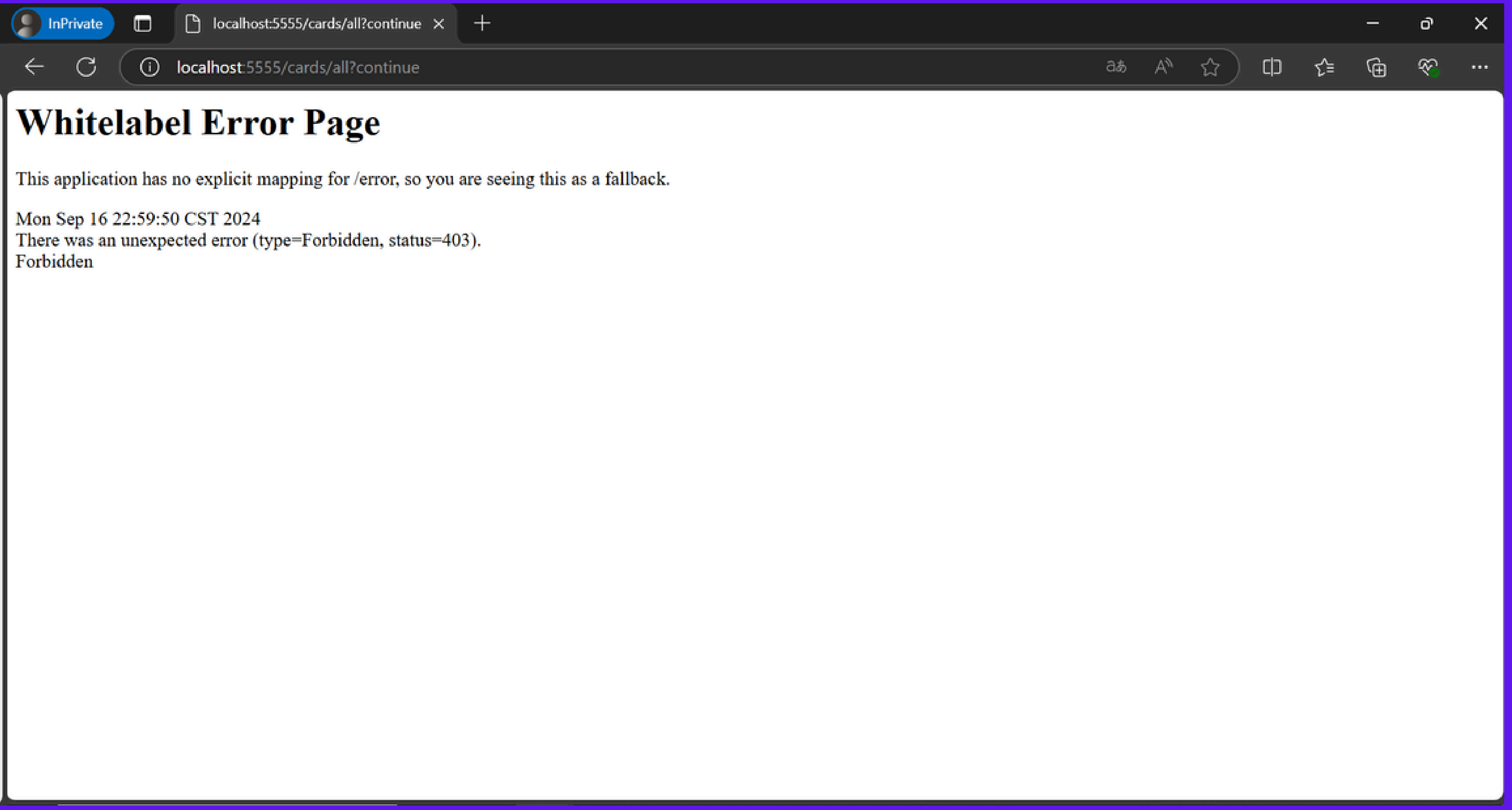
Spring Security



Spring Security

Not authorized user response



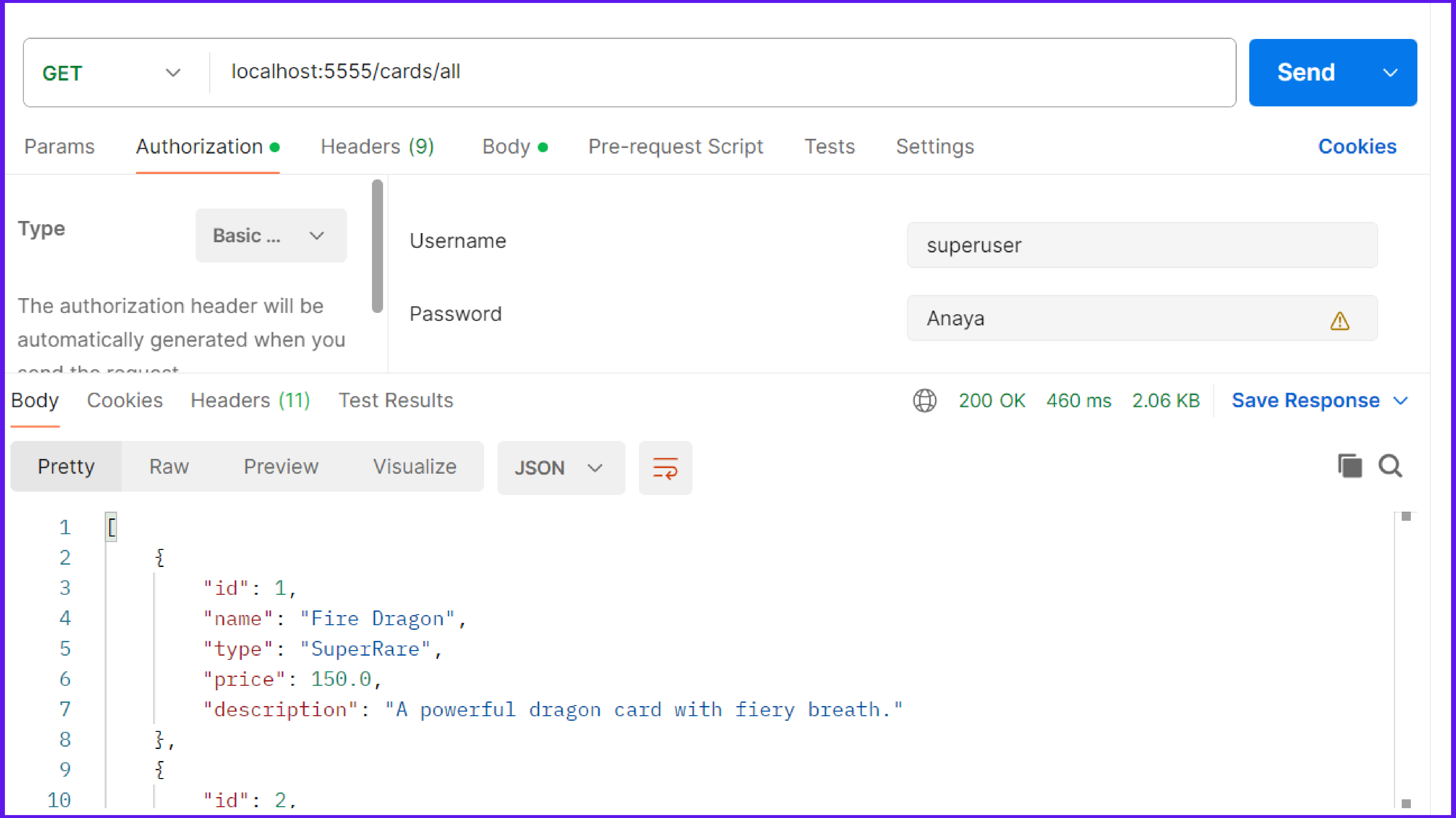

InPrivate
localhost:5555/cards/all?continue
localhost:5555/cards/all?continue

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Mon Sep 16 22:59:50 CST 2024
There was an unexpected error (type=Forbidden, status=403).
Forbidden

Authorized user response



localhost:5555/cards/all

Type: Basic ...

The authorization header will be automatically generated when you send the request

Username: superuser

Password: Anaya

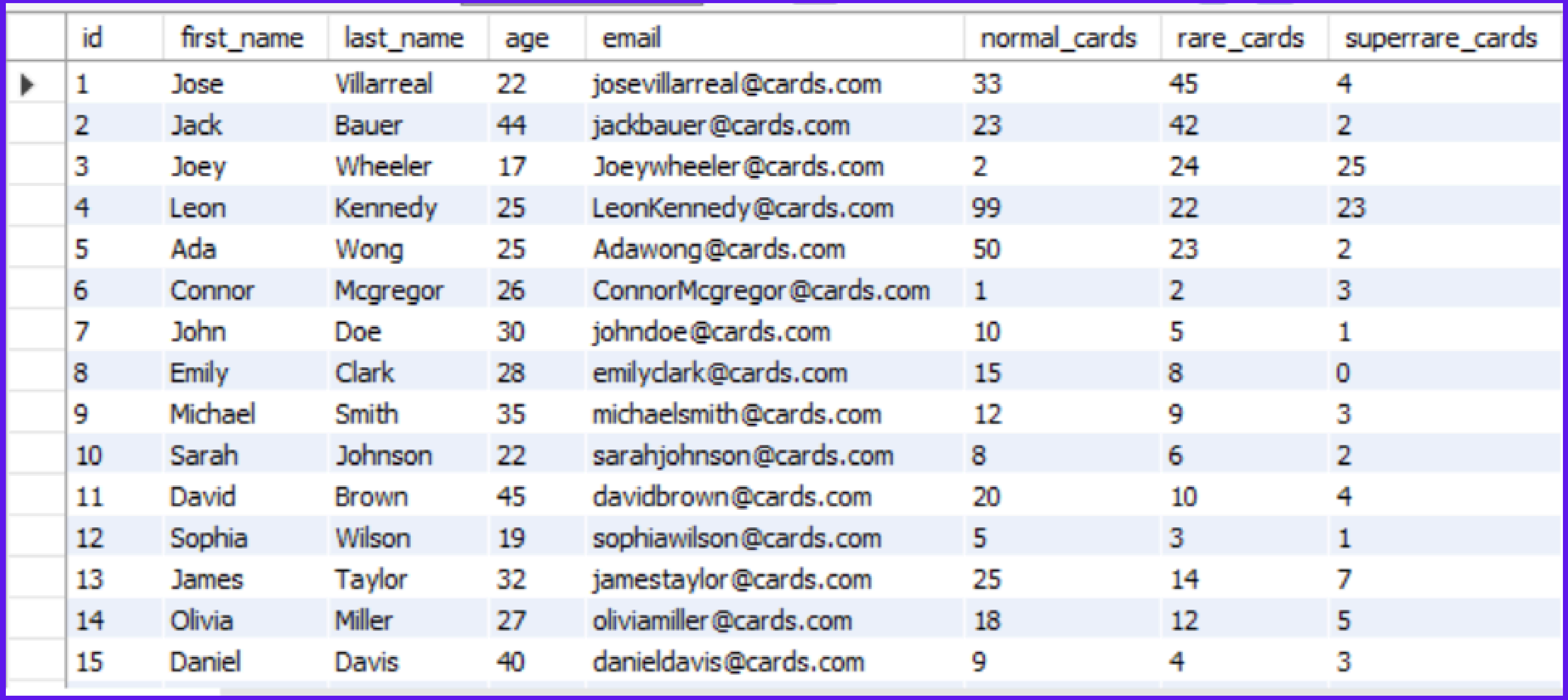
200 OK 460 ms 2.06 KB

```
[
    {
        "id": 1,
        "name": "Fire Dragon",
        "type": "SuperRare",
        "price": 150.0,
        "description": "A powerful dragon card with fiery breath."
    },
    {
        "id": 2,
```

Two table implementation

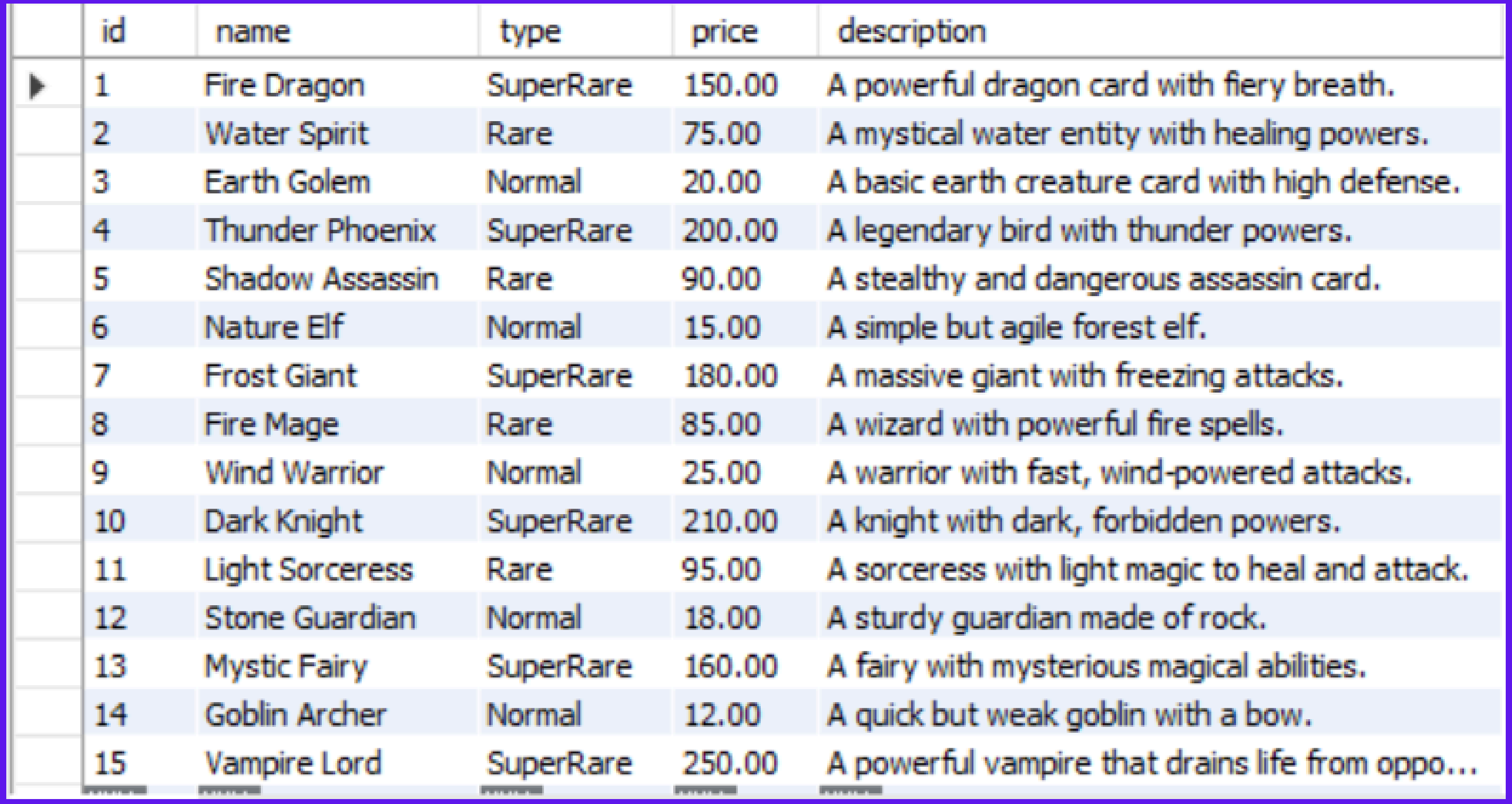
SQL Tables

user_card



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

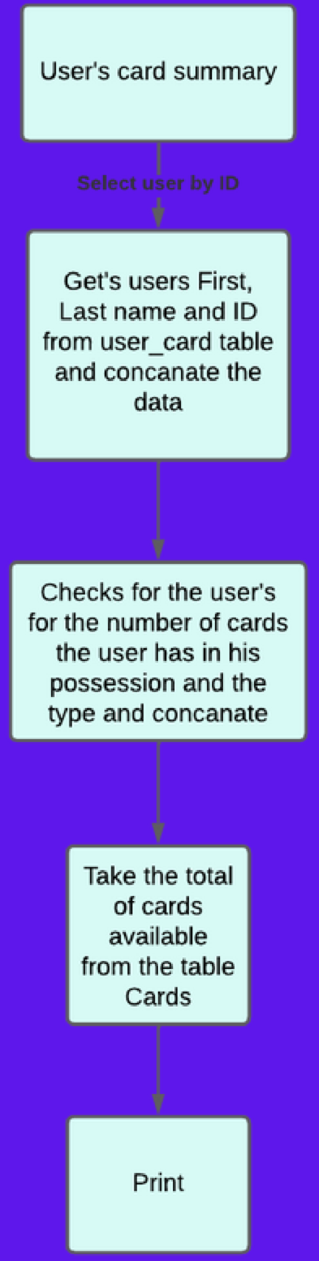
cards



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...

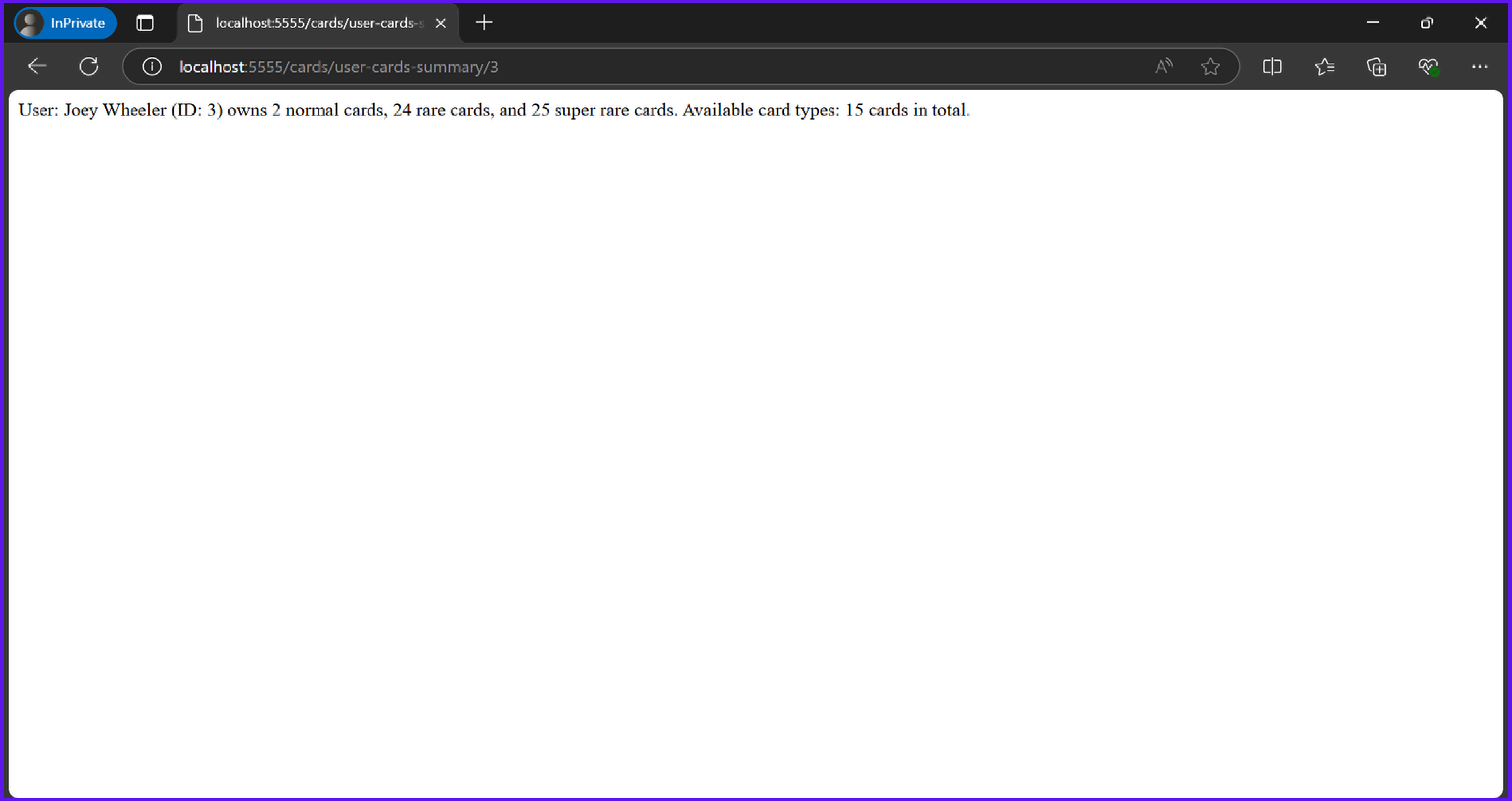
Two table implementation

How does the two table implementation works


User's card summary
Select user by ID
Get's users First, Last name and ID from user_card table and concanate the data
Checks for the user's for the number of cards the user has in his possession and the type and concanate
Take the total of cards available from the table Cards
Print

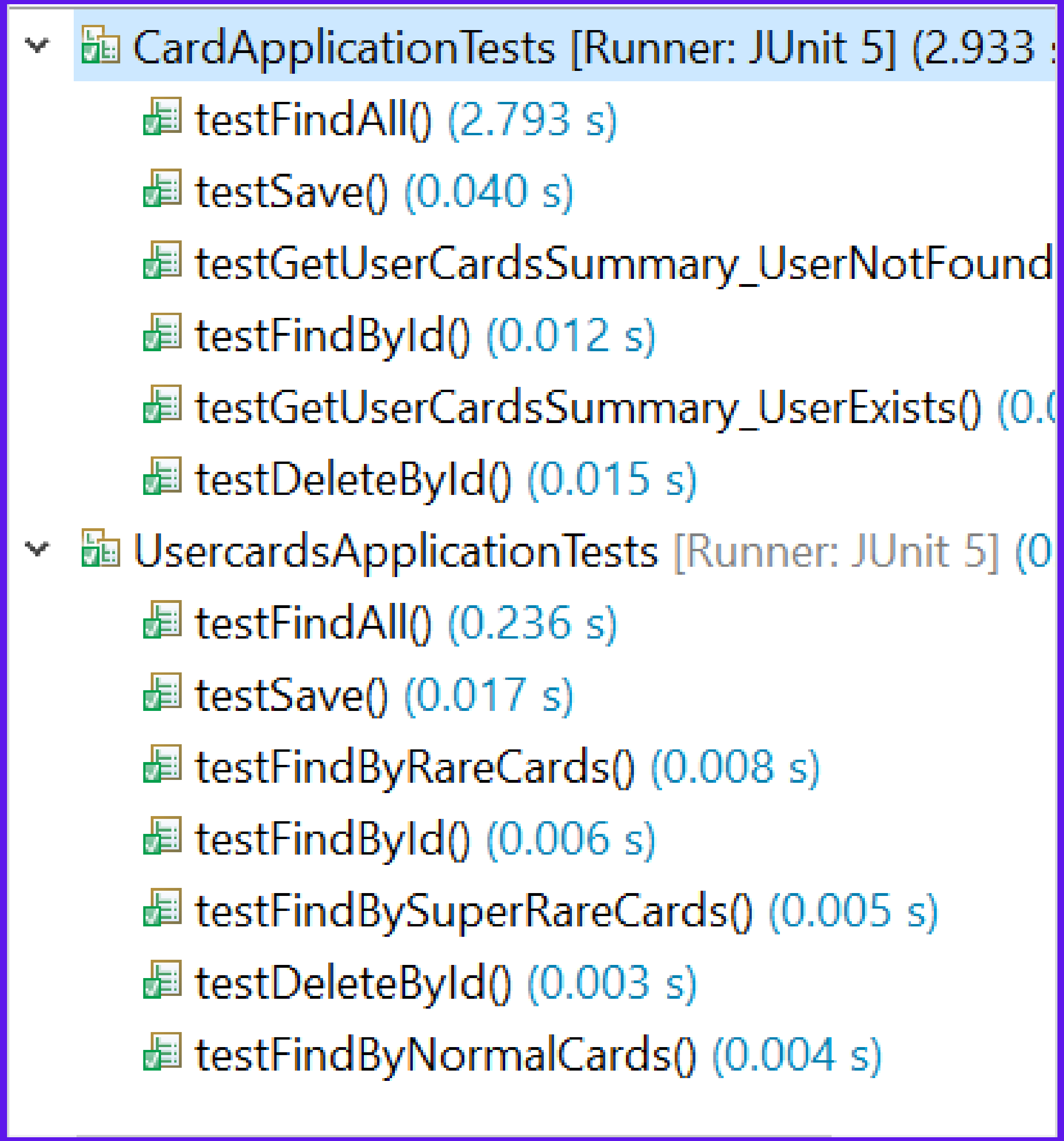
Two table implementation

Output from Localhost

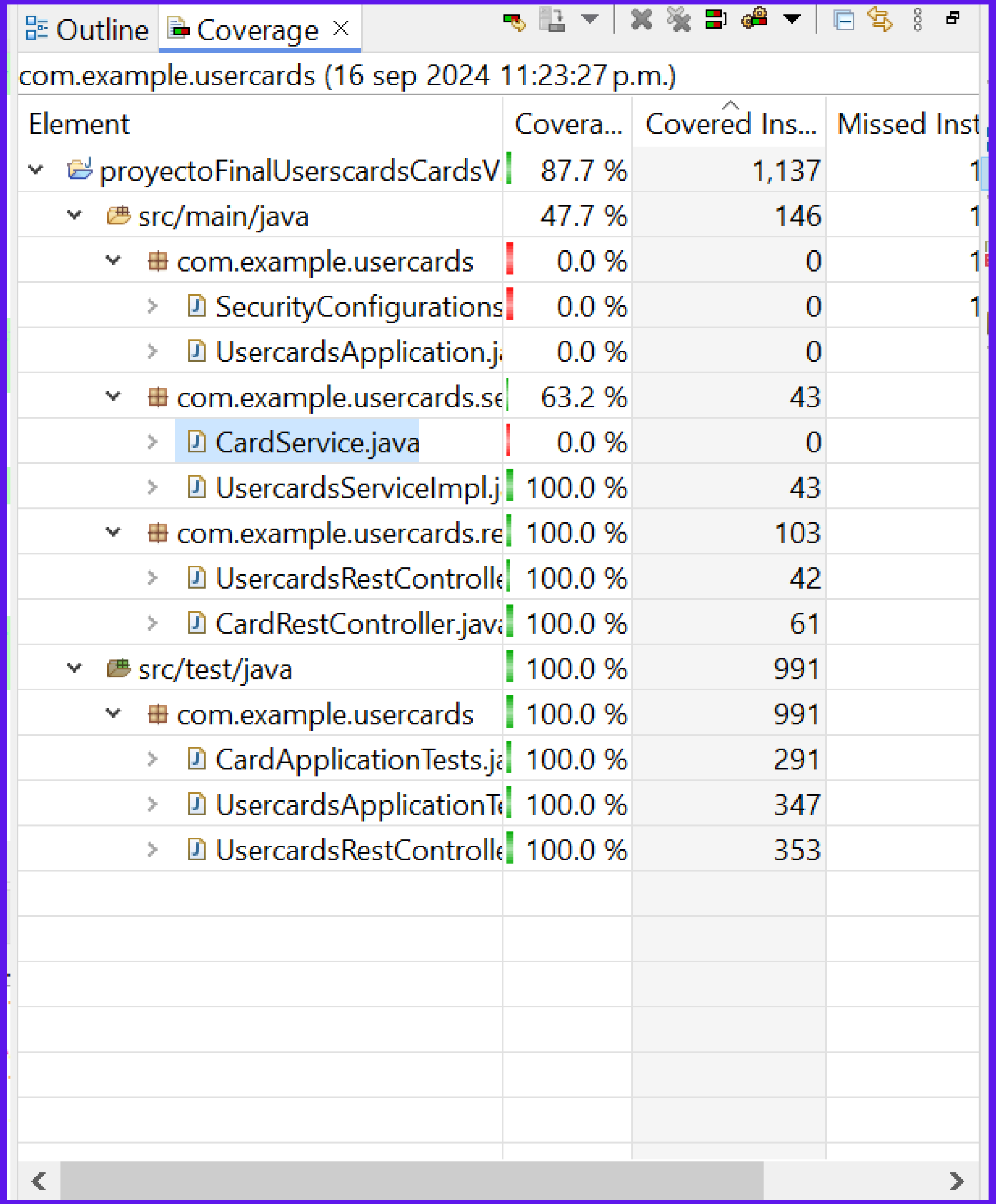

InPrivate
localhost:5555/cards/user-cards-s
localhost:5555/cards/user-cards-summary/3

```
User: Joey Wheeler (ID: 3) owns 2 normal cards, 24 rare cards, and 25 super rare cards. Available card types: 15 cards in total.
```

JUnit and Mockito

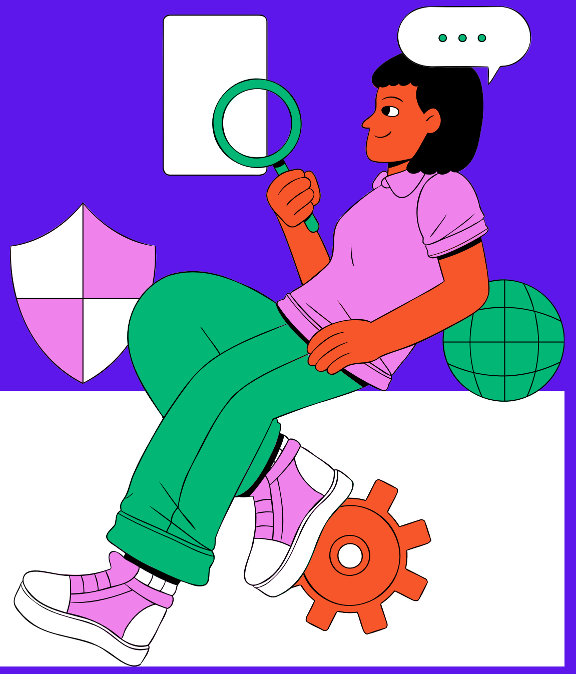


```
CardApplicationTests [Runner: JUnit 5] (2.933
    testFindAll() (2.793 s)
    testSave() (0.040 s)
    testGetUserCardsSummary_UserNotFound
    testFindById() (0.012 s)
    testGetUserCardsSummary_UserExists() (0.
    testDeleteById() (0.015 s)
UsercardsApplicationTests [Runner: JUnit 5] (0
    testFindAll() (0.236 s)
    testSave() (0.017 s)
    testFindByRareCards() (0.008 s)
    testFindById() (0.006 s)
    testFindBySuperRareCards() (0.005 s)
    testDeleteById() (0.003 s)
    testFindByNormalCards() (0.004 s)
```



com.example.usercards (16 sep 2024 11:23:27 p.m.)

Element	Covera...	Covered Ins...	Missed Inst
proyectoFinalUserscardsCardsV	87.7 %	1,137	1
src/main/java	47.7 %	146	1
com.example.usercards	0.0 %	0	1
SecurityConfigurations	0.0 %	0	1
UsercardsApplication.j	0.0 %	0	
com.example.usercards.se	63.2 %	43	
CardService.java	0.0 %	0	
UsercardsServiceImpl.j	100.0 %	43	
com.example.usercards.re	100.0 %	103	
UsercardsRestControlle	100.0 %	42	
CardRestController.java	100.0 %	61	
src/test/java	100.0 %	991	
com.example.usercards	100.0 %	991	
CardApplicationTests.ja	100.0 %	291	
UsercardsApplicationTe	100.0 %	347	
UsercardsRestControlle	100.0 %	353	



Future Enhancements

Do an attractive frotend in order to catch clients attention

Implement this Spring project to a public domain

Migrate from a local data base to a cloud base one

Optimize code and establish better error results

Conclusion

- A Spring Boot
- B Spring Security
- C Unit Testing
- D Two table implementation