Academia Xideral Java Monterrey

Daniel Ivan Anaya Alvarez

Spring CRUD Data JPA

Introduction

For this project is designed to implement a CRUD application using Spring Data JPA. The application simulates a game card store where users can manage their collections of various types of game cards, including common, rare, and super rare cards. This project demonstrates how to effectively use Spring Boot, Spring Data JPA, and a relational database to manage data with minimal code and effort. For this project I based on the previous Spring CRUD JPA

1. Entity (Userscards):
   - Represents the database table for users and their card collections.
   - Mapped with JPA annotations (@Entity, @Table, etc.).
2. Repository (UserscardsRepository) (**MAIN DIFFERENCE WITH NORMAL JPA)**
   - Extends JpaRepository to provide built-in CRUD functionality.
   - Defines custom query methods like findByNormalcards(int normalCards).
3. Service Layer (UsercardsService):
   - Contains business logic and calls repository methods.
   - Ensures operations are transactional with @Transactional.
4. REST Controller (UsercardsRestController):
   - Handles HTTP requests and uses the service layer for CRUD operations.
   - Create endpoints:
     - GET /usercards/all – List all users.
     - GET /usercards/{id} – Find user by ID.
     - POST /usercards/save – Add or update a user.
     - DELETE /usercards/delete/{id} – Delete a user by ID.

Configuration

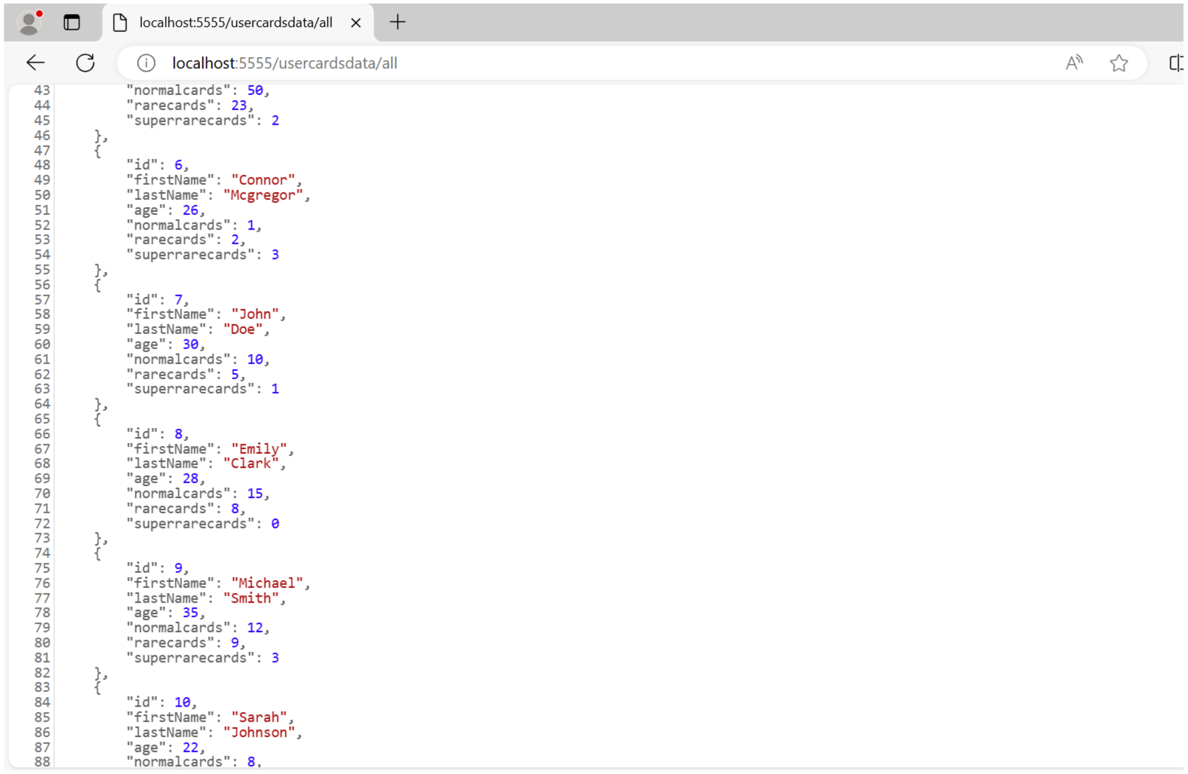
- Configured in application.properties to connect to a MySQL database.

Testing the Application

- Test with tools like Postman:
    - POST /usercards/save to add/update a user.
    - GET /usercards/all to retrieve all users.

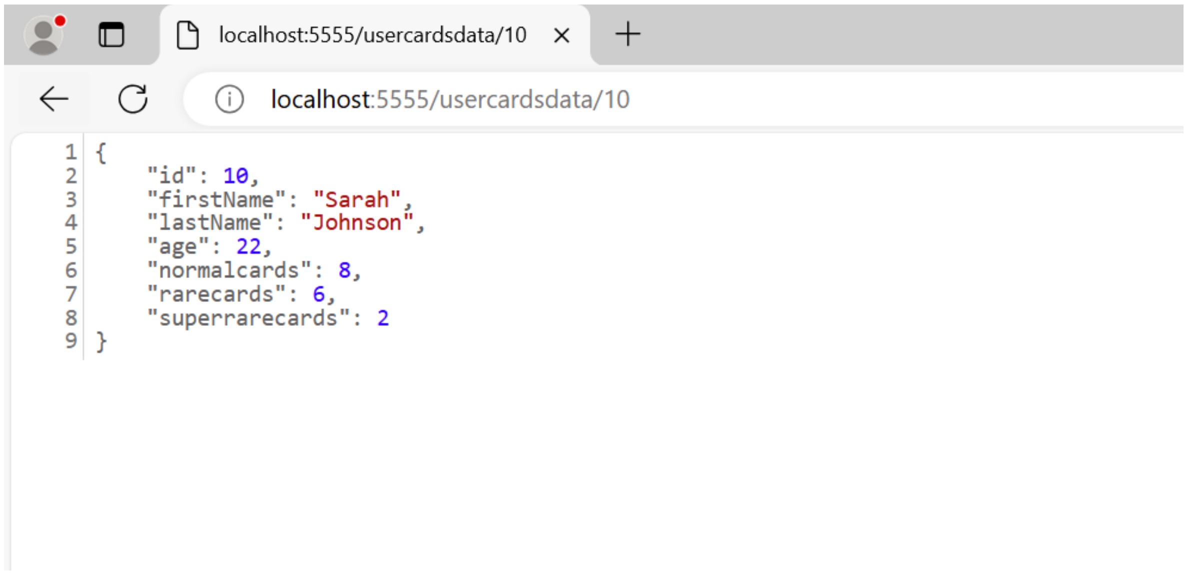
Prints

localhost:8080/usercardsdata/all (Everyone on the database)


localhost:5555/usercardsdata/all
localhost:5555/usercardsdata/all

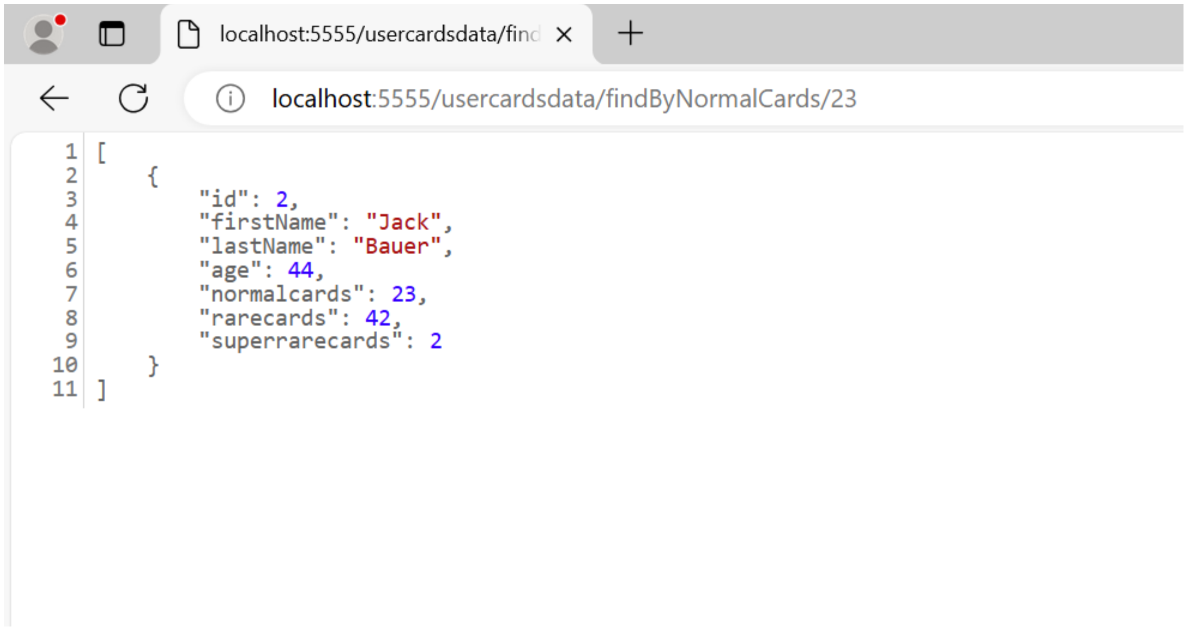
```
        "normalcards": 50,
        "rarecards": 23,
        "superrarecards": 2
    },
    {
        "id": 6,
        "firstName": "Connor",
        "lastName": "Mcgregor",
        "age": 26,
        "normalcards": 1,
        "rarecards": 2,
        "superrarecards": 3
    },
    {
        "id": 7,
        "firstName": "John",
        "lastName": "Doe",
        "age": 30,
        "normalcards": 10,
        "rarecards": 5,
        "superrarecards": 1
    },
    {
        "id": 8,
        "firstName": "Emily",
        "lastName": "Clark",
        "age": 28,
        "normalcards": 15,
        "rarecards": 8,
        "superrarecards": 0
    },
    {
        "id": 9,
        "firstName": "Michael",
        "lastName": "Smith",
        "age": 35,
        "normalcards": 12,
        "rarecards": 9,
        "superrarecards": 3
    },
    {
        "id": 10,
        "firstName": "Sarah",
        "lastName": "Johnson",
        "age": 22,
        "normalcards": 8,
```

llocalhost:5555/usercardsdata/10 (Search by id)


localhost:5555/usercardsdata/10

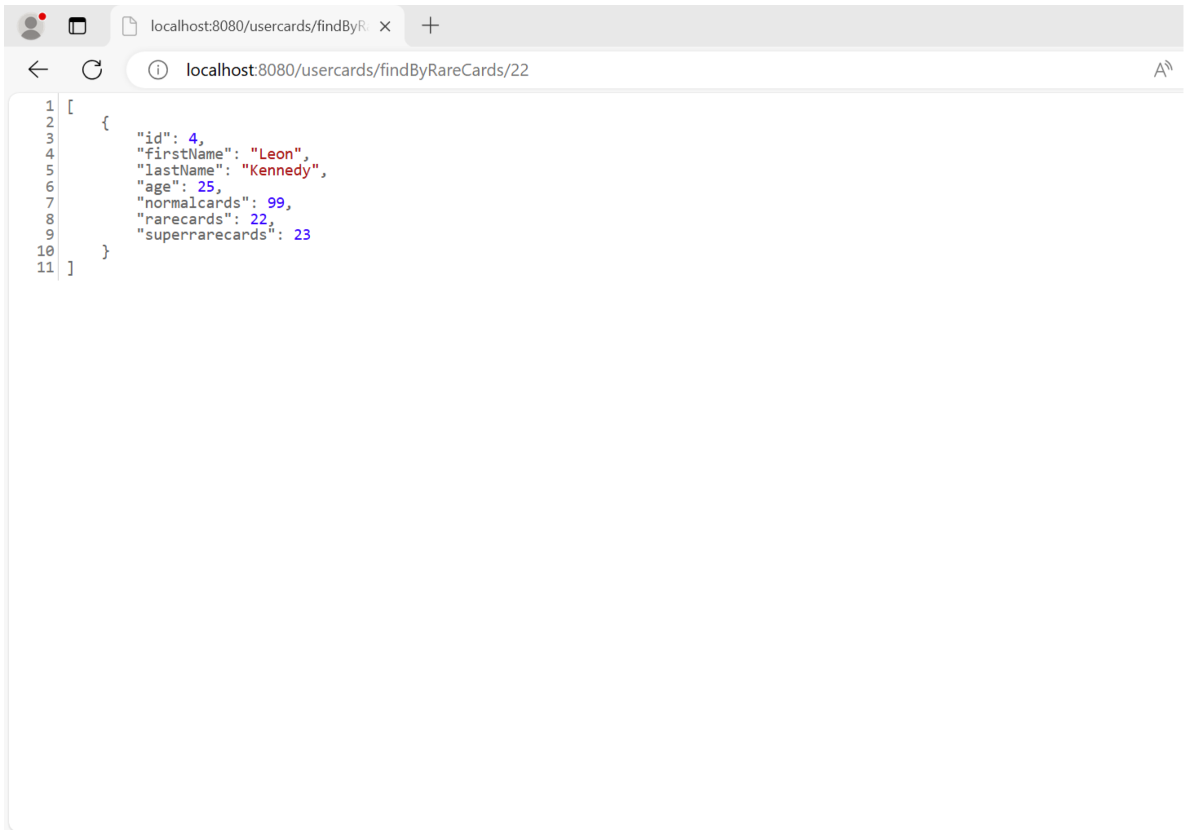
```
{
    "id": 10,
    "firstName": "Sarah",
    "lastName": "Johnson",
    "age": 22,
    "normalcards": 8,
    "rarecards": 6,
    "superrarecards": 2
}
```

localhost:5555/usercardsdata/findByNormalCards/23 (Find people that have 23 normal cards)


localhost:5555/usercardsdata/findByNormalCards/23

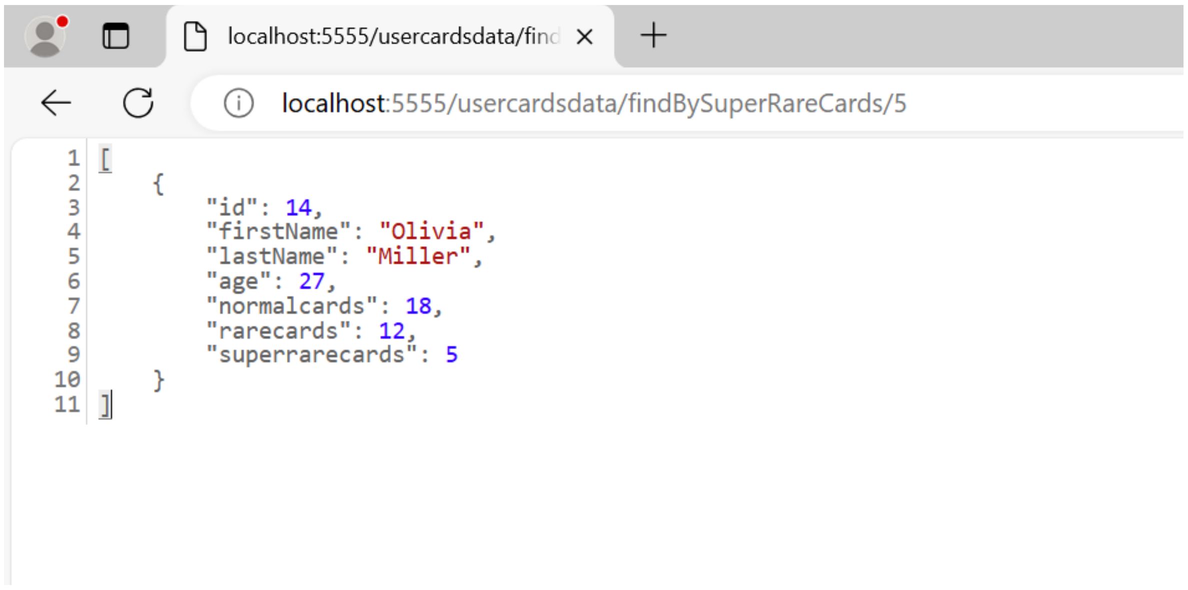
```
[
    {
        "id": 2,
        "firstName": "Jack",
        "lastName": "Bauer",
        "age": 44,
        "normalcards": 23,
        "rarecards": 42,
        "superrarecards": 2
    }
]
```

localhost:8080/usercardsdata/findByRareCards/22 (Find people with a specific number of rare cards)


localhost:8080/usercards/findByR
localhost:8080/usercards/findByRareCards/22

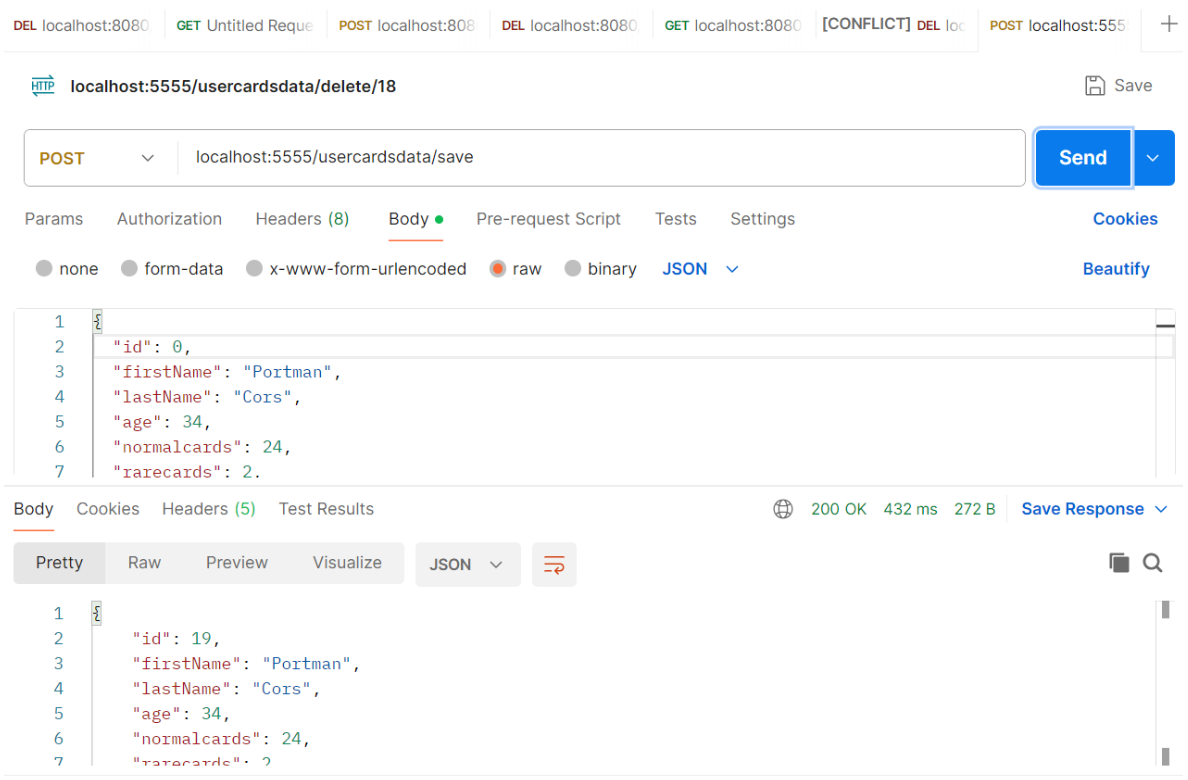
```
[
    {
        "id": 4,
        "firstName": "Leon",
        "lastName": "Kennedy",
        "age": 25,
        "normalcards": 99,
        "rarecards": 22,
        "superrarecards": 23
    }
]
```

localhost:5555/usercardsdata/findBySuperRareCards/5 (Find people with a specific amount of super rare cards)


localhost:5555/usercardsdata/finc
localhost:5555/usercardsdata/findBySuperRareCards/5

```
[
    {
        "id": 14,
        "firstName": "Olivia",
        "lastName": "Miller",
        "age": 27,
        "normalcards": 18,
        "rarecards": 12,
        "superrarecards": 5
    }
]
```

localhost:5555/usercardsdata/save (Add a user to the database with its corresponding id)



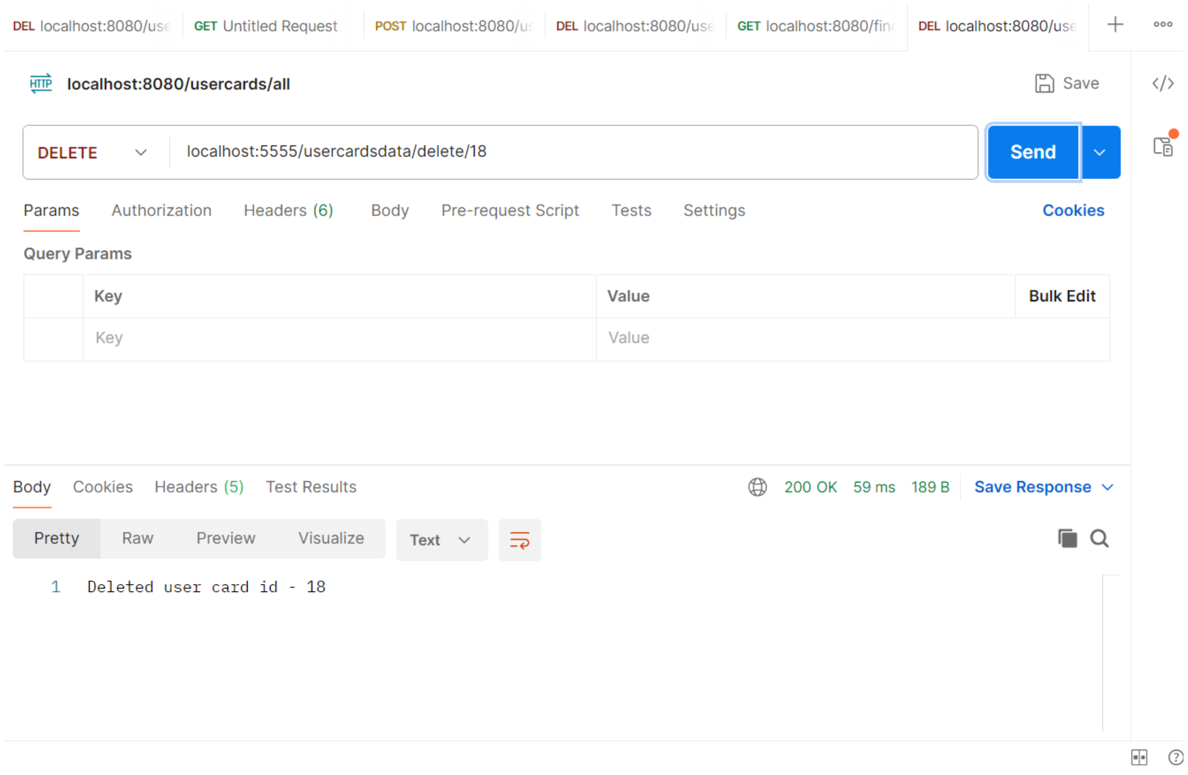
POST localhost:5555/usercardsdata/save

```
{
"id": 0,
"firstName": "Portman",
"lastName": "Cors",
"age": 34,
"normalcards": 24,
"rarecards": 2,
```

Body — 200 OK 432 ms 272 B — JSON

```
{
    "id": 19,
    "firstName": "Portman",
    "lastName": "Cors",
    "age": 34,
    "normalcards": 24,
    "rarecards": 2
```

localhost:5555/usercardsdata/delete/18 (Deletes a user by its id)



DELETE localhost:5555/usercardsdata/delete/18

Key	Value
Key	Value

Body — 200 OK 59 ms 189 B — Text

```
Deleted user card id - 18
```

Conclusion

This Spring JPA CRUD Data helps us make our methods easier than using simple JPA, its method implied using less method like using less traditional coding for SQL (Quering) and gave us more straightforward method for obtention data.

Appendix (Code)

UsercardsApplication.java

```java
package com.example.usercards;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


//Main application to run
@SpringBootApplication
public class UsercardsApplication {

	public static void main(String[] args) {
		SpringApplication.run(UsercardsApplication.class, args);
	}

}
```

UsercardsRestController.java

```java
package com.example.usercards.rest;
import com.example.usercards.entity.Userscards;
import com.example.usercards.service.UsercardsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/usercardsdata") // Prefix for route
public class UsercardsRestController {

    private final UsercardsService usercardsService;

    // Inject the service using autowired
    @Autowired
    public UsercardsRestController(UsercardsService usercardsService) {
        this.usercardsService = usercardsService;
    }
    // Endpoint to get all user
    @GetMapping("/all")
    public List<Userscards> findAll() {
        return usercardsService.findAll();
    }

    // Endpoint to get user by id
    @GetMapping("/{id}")
    public Userscards findById(@PathVariable int id) {
        return usercardsService.findById(id);
    }

    // Endpoint to get save an user or update an existing one
    @PostMapping("/save")
    public Userscards save(@RequestBody Userscards usercards) {
        return usercardsService.save(usercards);
    }

    // Endpoint to eliminate an existing user by id
    @DeleteMapping("/delete/{id}")
    public String deleteById(@PathVariable int id) {
        usercardsService.deleteById(id);
        return "Deleted user card id - " + id;
    }

    // Endpoint to search usercards by their amount of normal cards
    @GetMapping("/findByNormalCards/{normalCards}")
    public List<Userscards> findByNormalCards(@PathVariable int normalCards) {
        return usercardsService.findByNormalCards(normalCards);
    }

    // Endpoint to search usercards by their amount of rare cards
    @GetMapping("/findByRareCards/{rareCards}")
    public List<Userscards> findByRareCards(@PathVariable int rareCards) {
        return usercardsService.findByRareCards(rareCards);
    }

    // Endpoint to search usercards by their amount of rare cards
    @GetMapping("/findBySuperRareCards/{superRareCards}")
    public List<Userscards> findBySuperRareCards(@PathVariable int
superRareCards) {
        return usercardsService.findBySuperRareCards(superRareCards);
    }
}
```

UsercardsRepository.java

```
package com.example.usercards.dao;

import com.example.usercards.entity.Userscards;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
```

```
@Repository
public interface UserscardsRepository extends JpaRepository<Userscards, Integer>
{

    // Custom query methods using method naming conventions
    List<Userscards> findByNormalcards(int normalCards);
    List<Userscards> findByRarecards(int rareCards);
    List<Userscards> findBySuperrarecards(int superRareCards);
}
```

Application.properties

```
spring.application.name=usercards
#
server.port = 5555

# JDBC properties
#
spring.datasource.url=jdbc:mysql://localhost:3306/users_cards
spring.datasource.username=jdbcdaniel
spring.datasource.password=xideral
```

UsercardsService.java

```
package com.example.usercards.service;

import com.example.usercards.entity.Userscards;
import java.util.List;

public interface UsercardsService {
    List<Userscards> findAll(); // Retrieves all userscards
    Userscards findById(int id); // Finds a userscard by ID
    List<Userscards> findByNormalCards(int normalCards); // Finds userscards by
normal card count
    List<Userscards> findByRareCards(int rareCards); // Finds userscards by rare
card count
    List<Userscards> findBySuperRareCards(int superRareCards); // Finds
userscards by super rare card count
    Userscards save(Userscards usercards); // Saves or updates a userscard
    void deleteById(int id); // Deletes a userscard by ID
}
```

UsercardsServiceImpl.java

```
package com.example.usercards.service;

import com.example.usercards.dao.UserscardsRepository;
import com.example.usercards.entity.Userscards;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;

@Service
public class UsercardsServiceImpl implements UsercardsService {

    private final UserscardsRepository userscardsRepository;

    // Injects the repository using constructor injection
    @Autowired
    public UsercardsServiceImpl(UserscardsRepository userscardsRepository) {
        this.userscardsRepository = userscardsRepository;
    }

    @Override
    public List<Userscards> findAll() {
        return userscardsRepository.findAll(); // Calls the repository method to 
get all userscards
    }

    @Override
    public Userscards findById(int id) {
        Optional<Userscards> result = userscardsRepository.findById(id); // 
Calls the repository method to find by ID
        return result.orElse(null); // Returns the result or null if not found
    }

    @Override
    public List<Userscards> findByNormalCards(int normalCards) {
        return userscardsRepository.findByNormalcards(normalCards); // Calls the 
custom repository method
    }

    @Override
    public List<Userscards> findByRareCards(int rareCards) {
        return userscardsRepository.findByRarecards(rareCards); // Calls the 
custom repository method
    }

    @Override
    public List<Userscards> findBySuperRareCards(int superRareCards) {
        return userscardsRepository.findBySuperrarecards(superRareCards); // 
Calls the custom repository method
    }

    @Override
    public Userscards save(Userscards usercards) {
        return userscardsRepository.save(usercards); // Calls the repository 
method to save or update the userscard
    }

    @Override
    public void deleteById(int id) {
        userscardsRepository.deleteById(id); // Calls the repository method to 
delete by ID
```

```
    }
}
```