Sure, I can suggest a Spring Boot project with OneToMany relationship, Authentication, Authorization, User entity and other related entities. Here's a basic outline of the project:

Project Name: Online Store

Description: An online store that allows customers to purchase items and manage their account.

Entities:

1. User - This entity will store the user information including username, password, email, role, and other details.
2. Product - This entity will store the product information including name, description, price, and other details.
3. Order - This entity will store the order information including order number, date, customer details, product details, and other details.

Relationships:

1. OneToMany relationship between User and Order entities. One user can have multiple orders.
2. OneToMany relationship between Product and Order entities. One product can be included in multiple orders.

Authentication and Authorization:

The project will use Spring Security for authentication and authorization. Spring Security will provide the following features:

1. Login page to authenticate users
2. Access control based on user role
3. User registration page to create new users
4. Password encryption and decryption

Project Setup:

1. Create a new Spring Boot project with Spring Web, Spring Data JPA, Spring Security, and Thymeleaf dependencies.
2. Create the User, Product, and Order entities with the necessary attributes.
3. Configure the OneToMany relationships between User and Order entities and Product and Order entities.
4. Implement the authentication and authorization features using Spring Security.
5. Implement the user registration page to allow new users to register.
6. Create the necessary controllers, services, and repositories for the entities.
7. Implement the views using Thymeleaf.

Conclusion:

In this project, you will learn how to implement OneToMany relationships, authentication, authorization, and user registration in a Spring Boot project. You can extend the project further by adding features like payment processing, product reviews, and order tracking.

In the Online Store project I outlined earlier, the manipulation of Product, Order, and User entities will be controlled by the Spring Security authorization feature based on the roles of the logged-in user.

Here's a breakdown of the user roles and their associated permissions:

1. ROLE\_USER - A user with this role can view products, place orders, and view their own orders.
2. ROLE\_ADMIN - A user with this role can perform all actions of ROLE\_USER, plus they can add, edit, and delete products, view all orders, and edit and delete any order or user.

The Spring Security configuration can be set up to allow or deny access to specific routes or actions based on the user's role. For example, only users with the ROLE\_ADMIN can access the product management page where they can add, edit, or delete products.

Similarly, only users with the ROLE\_ADMIN can access the order management page where they can view all orders, edit and delete any order, and view and edit any user.

And only the user who is currently logged in can access their own user profile page to view and edit their personal details.

In summary, the manipulation of Product, Order, and User entities will be controlled by the Spring Security authorization feature based on the roles of the logged-in user.