What is SpringShop?

SpringShop is an online store which offers the purchase of electronic devices. By accessing the website, the user has options of registering or logging into an existing account. After that, the user selects a product from the existing list and proceeds to finalize the order.

Applied technologies

- Java (version 11)

- Maven

- Spring Boot (version 2.6.6)

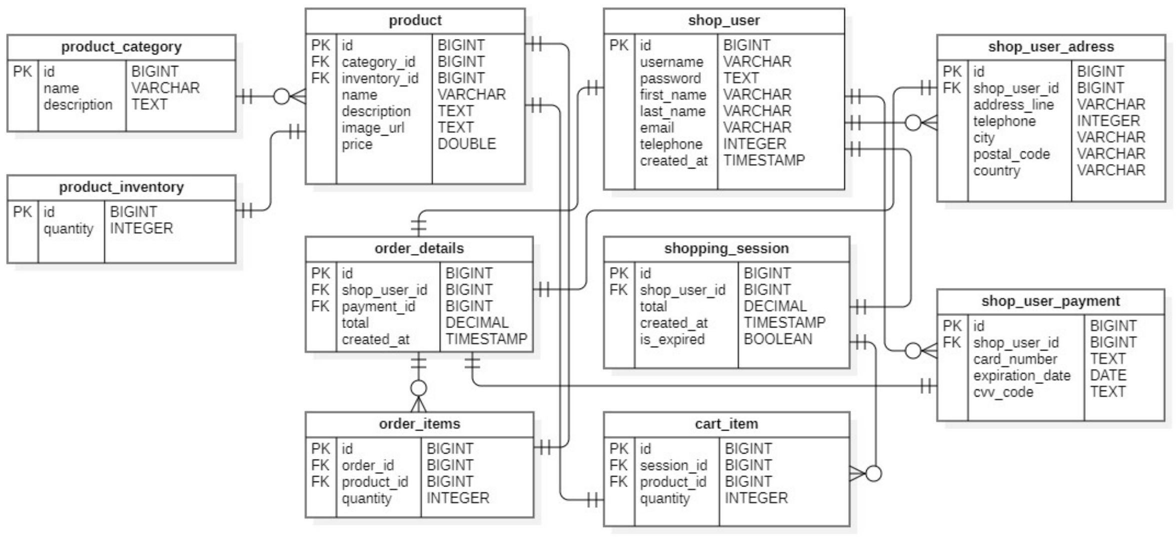
- Lombok – library for automation of creating classes

- Spring Security – a tool to ensure authentication, authorization, other security features

- PostgreSQL

- Spring Data JPA – a tool for using the database

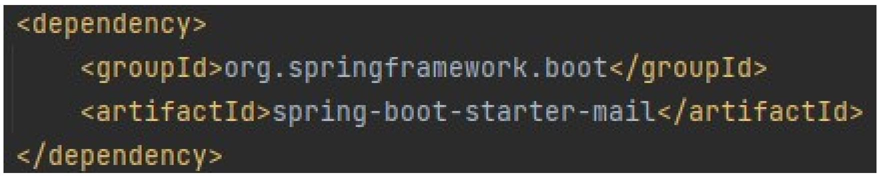
Database UML diagram



Email notifications

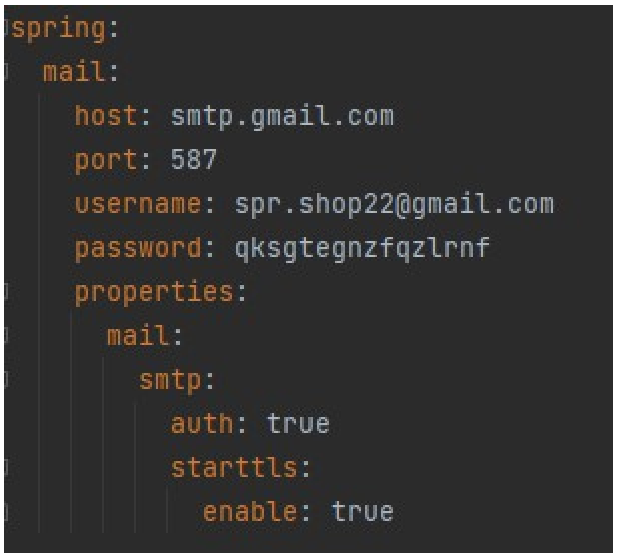
JavaMail API was used to send messages to customers. It is a library that provides solutions for sending emails.

First add dependencies to the project:



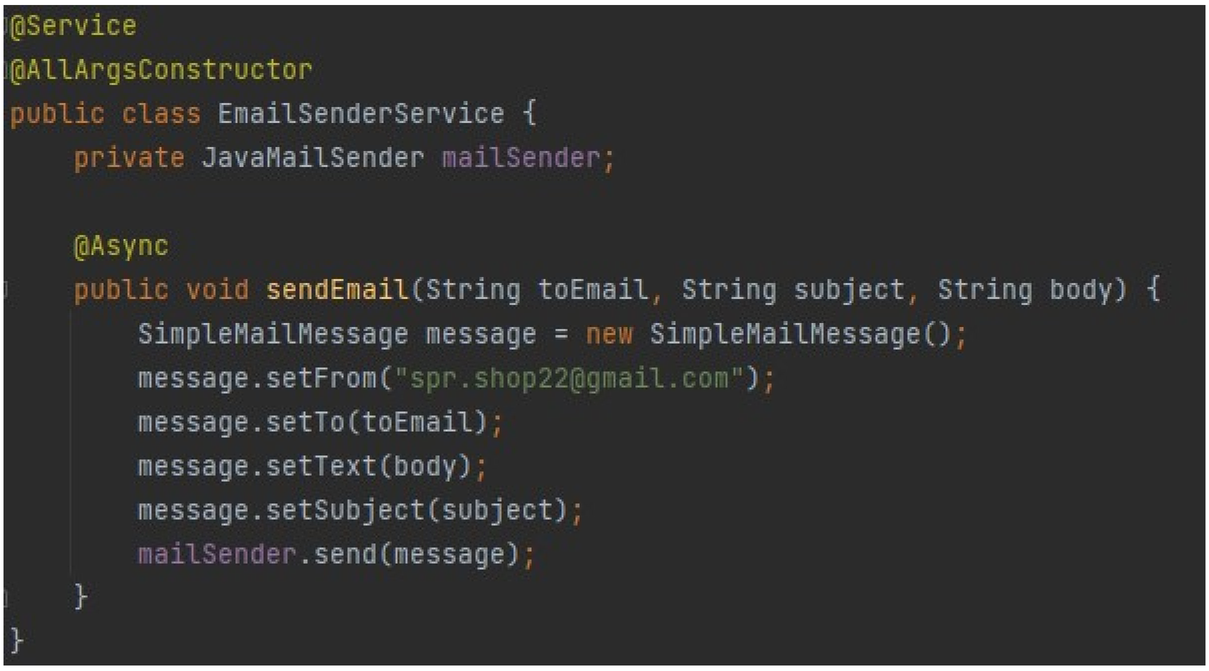
```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

Then in the applications.yaml file we need to add the configuration:



```yaml
spring:
  mail:
    host: smtp.gmail.com
    port: 587
    username: spr.shop22@gmail.com
    password: qksgtegnzfqzlrnf
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
```

Implementation:

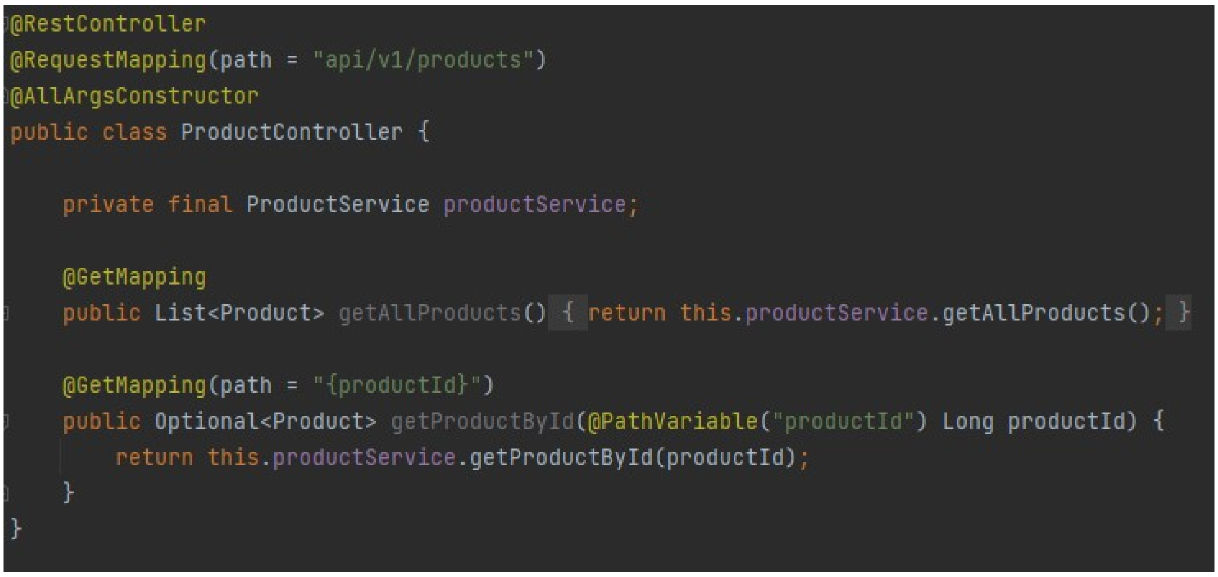


```java
@Service
@AllArgsConstructor
public class EmailSenderService {
    private JavaMailSender mailSender;

    @Async
    public void sendEmail(String toEmail, String subject, String body) {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setFrom("spr.shop22@gmail.com");
        message.setTo(toEmail);
        message.setText(body);
        message.setSubject(subject);
        mailSender.send(message);
    }
}
```

Endpoints

For each of the classes in the project, we created methods which are endpoints. They can be used to display objects of a class, add, or remove them.

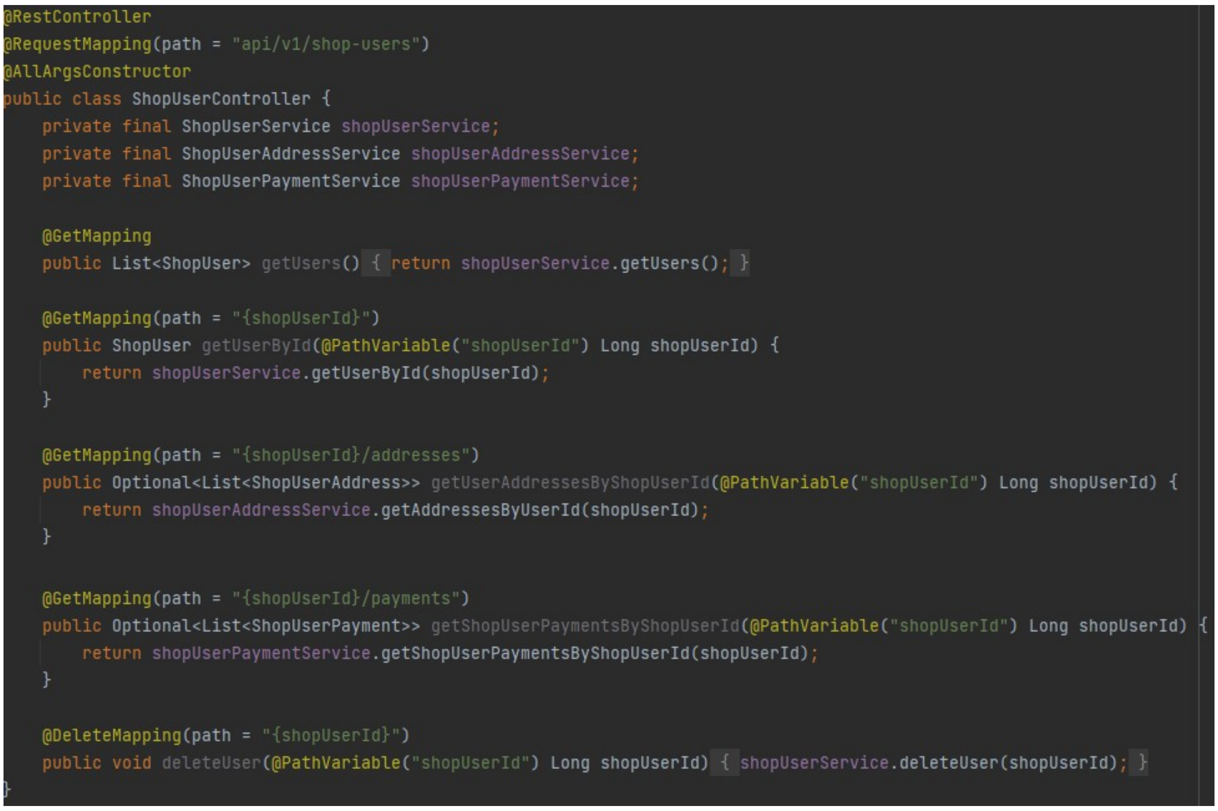
1. **Product controller**



```java
@RestController
@RequestMapping(path = "api/v1/products")
@AllArgsConstructor
public class ProductController {

    private final ProductService productService;

    @GetMapping
    public List<Product> getAllProducts() { return this.productService.getAllProducts(); }

    @GetMapping(path = "{productId}")
    public Optional<Product> getProductById(@PathVariable("productId") Long productId) {
        return this.productService.getProductById(productId);
    }
}
```

- *getAllProducts* - returns a list of products from the database
- *getProductById* - returns a product that has the given identifier

2. **ShopUser controller**



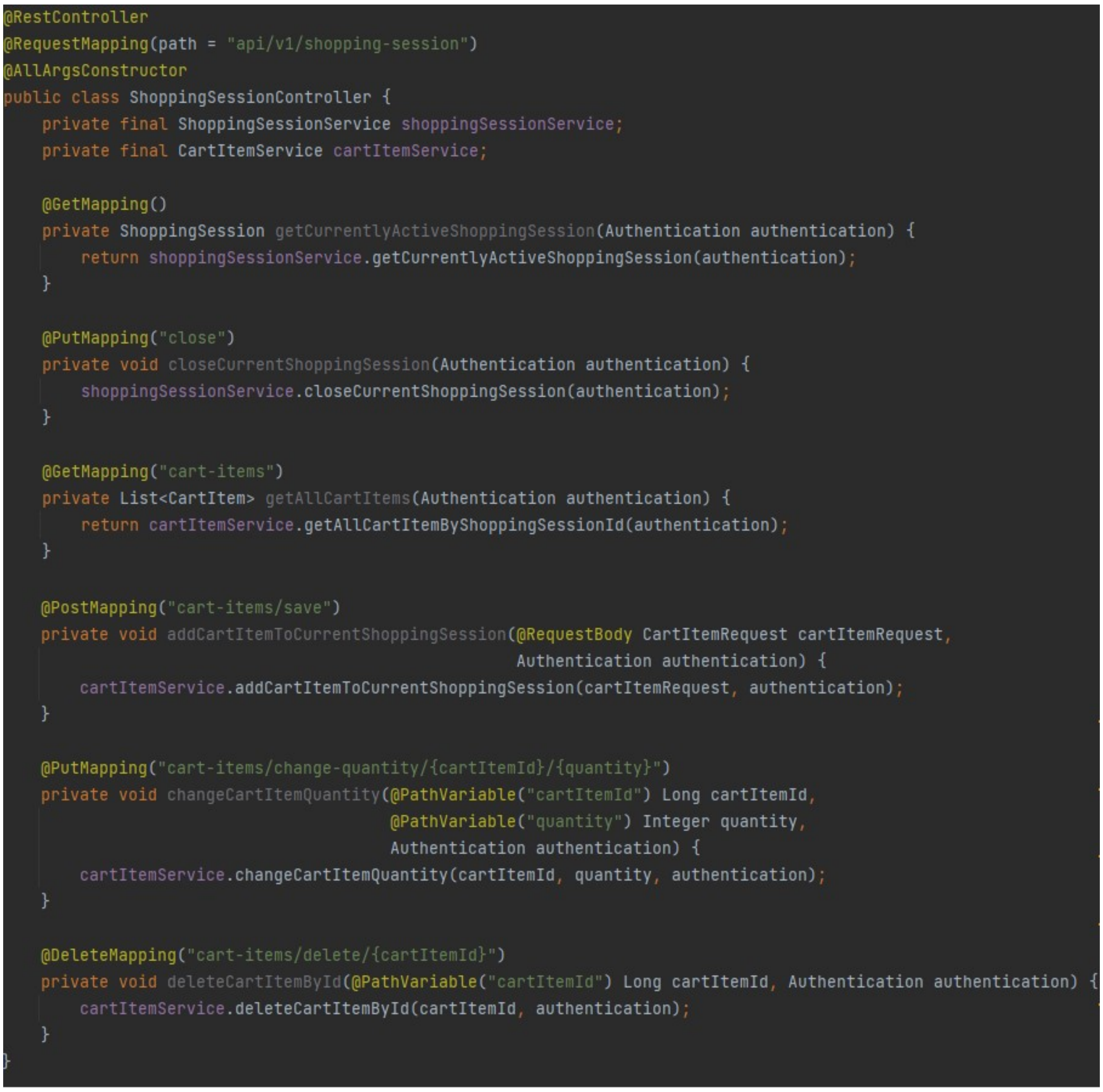
```java
@RestController
@RequestMapping(path = "api/v1/shop-users")
@AllArgsConstructor
public class ShopUserController {
    private final ShopUserService shopUserService;
    private final ShopUserAddressService shopUserAddressService;
    private final ShopUserPaymentService shopUserPaymentService;

    @GetMapping
    public List<ShopUser> getUsers() { return shopUserService.getUsers(); }

    @GetMapping(path = "{shopUserId}")
    public ShopUser getUserById(@PathVariable("shopUserId") Long shopUserId) {
        return shopUserService.getUserById(shopUserId);
    }

    @GetMapping(path = "{shopUserId}/addresses")
    public Optional<List<ShopUserAddress>> getUserAddressesByShopUserId(@PathVariable("shopUserId") Long shopUserId) {
        return shopUserAddressService.getAddressesByUserId(shopUserId);
    }

    @GetMapping(path = "{shopUserId}/payments")
    public Optional<List<ShopUserPayment>> getShopUserPaymentsByShopUserId(@PathVariable("shopUserId") Long shopUserId) {
        return shopUserPaymentService.getShopUserPaymentsByShopUserId(shopUserId);
    }

    @DeleteMapping(path = "{shopUserId}")
    public void deleteUser(@PathVariable("shopUserId") Long shopUserId) { shopUserService.deleteUser(shopUserId); }
}
```

- *getUsers* - returns a list of shop users

- *getUserById* - returns a user that has the identifier specified in the parameter

- *getUserAdressesByShopUserId* - returns a list of adresses of the user that has the identifier specified in the parameter
- *getShopUserPamentsByShopUserId* - returns a list of payments of the user that has the identifier specified in the parameter
- *deleteUser* - removes the shop user that has the identifier specified in the parameter

3. ShoppingSession controller



```java
@RestController
@RequestMapping(path = "api/v1/shopping-session")
@AllArgsConstructor
public class ShoppingSessionController {
    private final ShoppingSessionService shoppingSessionService;
    private final CartItemService cartItemService;

    @GetMapping()
    private ShoppingSession getCurrentlyActiveShoppingSession(Authentication authentication) {
        return shoppingSessionService.getCurrentlyActiveShoppingSession(authentication);
    }

    @PutMapping("close")
    private void closeCurrentShoppingSession(Authentication authentication) {
        shoppingSessionService.closeCurrentShoppingSession(authentication);
    }

    @GetMapping("cart-items")
    private List<CartItem> getAllCartItems(Authentication authentication) {
        return cartItemService.getAllCartItemByShoppingSessionId(authentication);
    }

    @PostMapping("cart-items/save")
    private void addCartItemToCurrentShoppingSession(@RequestBody CartItemRequest cartItemRequest,
                                                     Authentication authentication) {
        cartItemService.addCartItemToCurrentShoppingSession(cartItemRequest, authentication);
    }

    @PutMapping("cart-items/change-quantity/{cartItemId}/{quantity}")
    private void changeCartItemQuantity(@PathVariable("cartItemId") Long cartItemId,
                                        @PathVariable("quantity") Integer quantity,
                                        Authentication authentication) {
        cartItemService.changeCartItemQuantity(cartItemId, quantity, authentication);
    }

    @DeleteMapping("cart-items/delete/{cartItemId}")
    private void deleteCartItemById(@PathVariable("cartItemId") Long cartItemId, Authentication authentication) {
        cartItemService.deleteCartItemById(cartItemId, authentication);
    }
}
```

- *getCurrentlyActiveShoppingSession* – returns the active session of the currently logged in user

- *closeCurrentShoppingSession* – closes the active sesssion of the currently logged in user

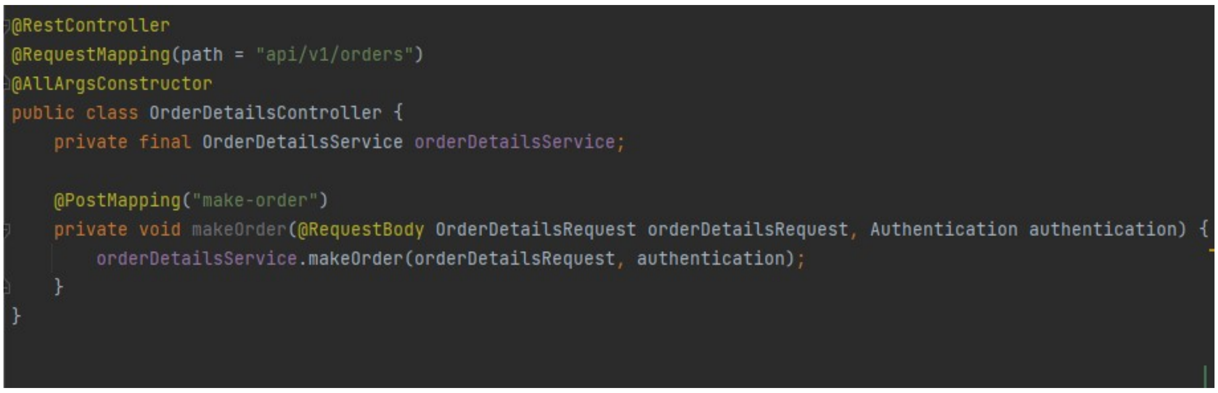
- *getAllCartItems* – returns a list of items from the cart in the current session

- *addCartItemToCurrentShoppingSession* – adds an item to the current shopping session

- *changeCartItemQuantity* – changes the quantity of the product whose identifier equals cartItemId

- *deleteCartItemById* – removes a product whose identifier equals cartItemId from the session

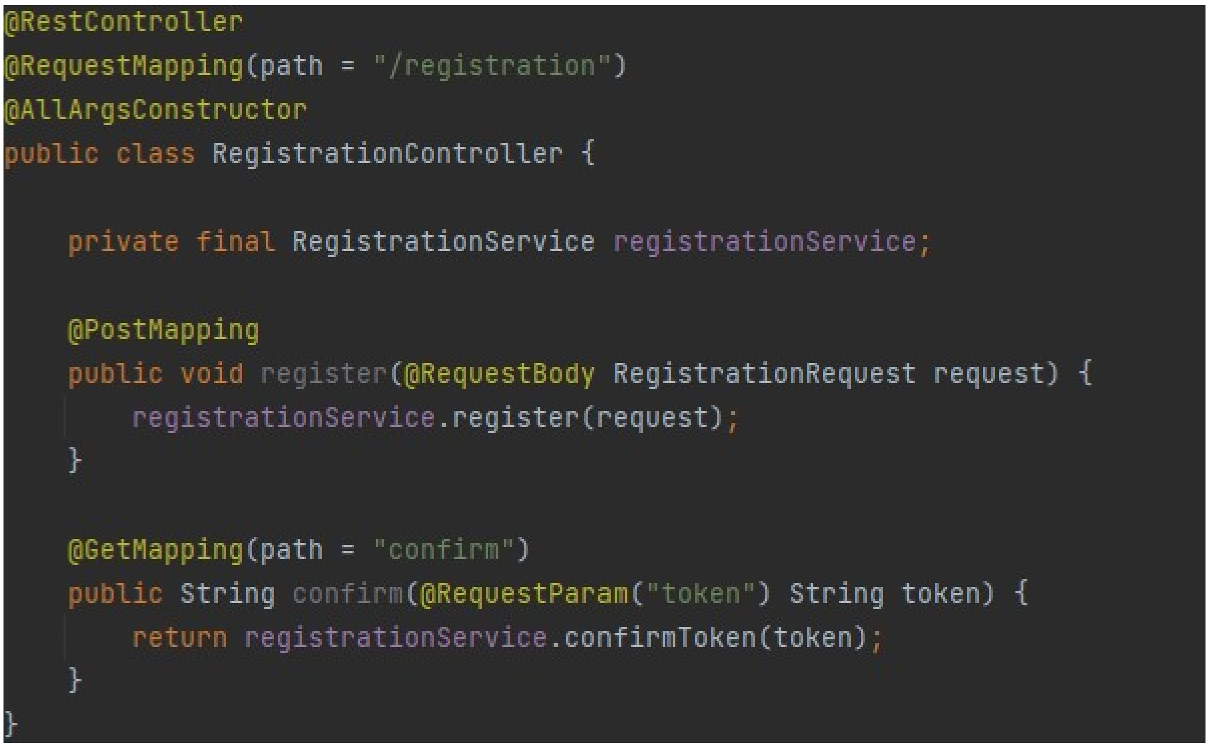
4. **OrderDetails controller**



```java
@RestController
@RequestMapping(path = "api/v1/orders")
@AllArgsConstructor
public class OrderDetailsController {
    private final OrderDetailsService orderDetailsService;

    @PostMapping("make-order")
    private void makeOrder(@RequestBody OrderDetailsRequest orderDetailsRequest, Authentication authentication) {
        orderDetailsService.makeOrder(orderDetailsRequest, authentication);
    }
}
```

- *makeOrder* – makes and order for products in the current shopping session

5. **Registration controller**

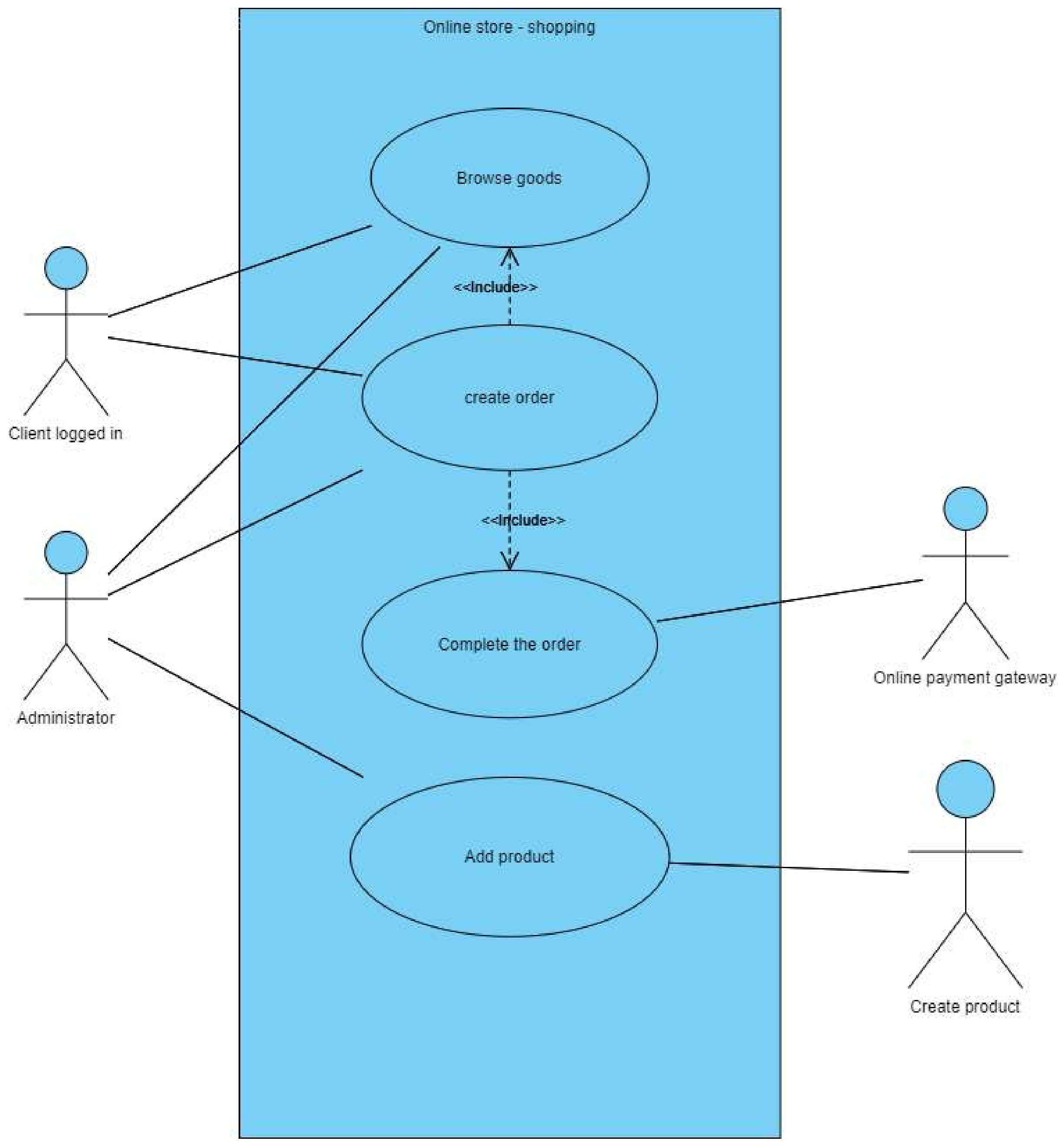


```java
@RestController
@RequestMapping(path = "/registration")
@AllArgsConstructor
public class RegistrationController {

    private final RegistrationService registrationService;

    @PostMapping
    public void register(@RequestBody RegistrationRequest request) {
        registrationService.register(request);
    }

    @GetMapping(path = "confirm")
    public String confirm(@RequestParam("token") String token) {
        return registrationService.confirmToken(token);
    }
}
```

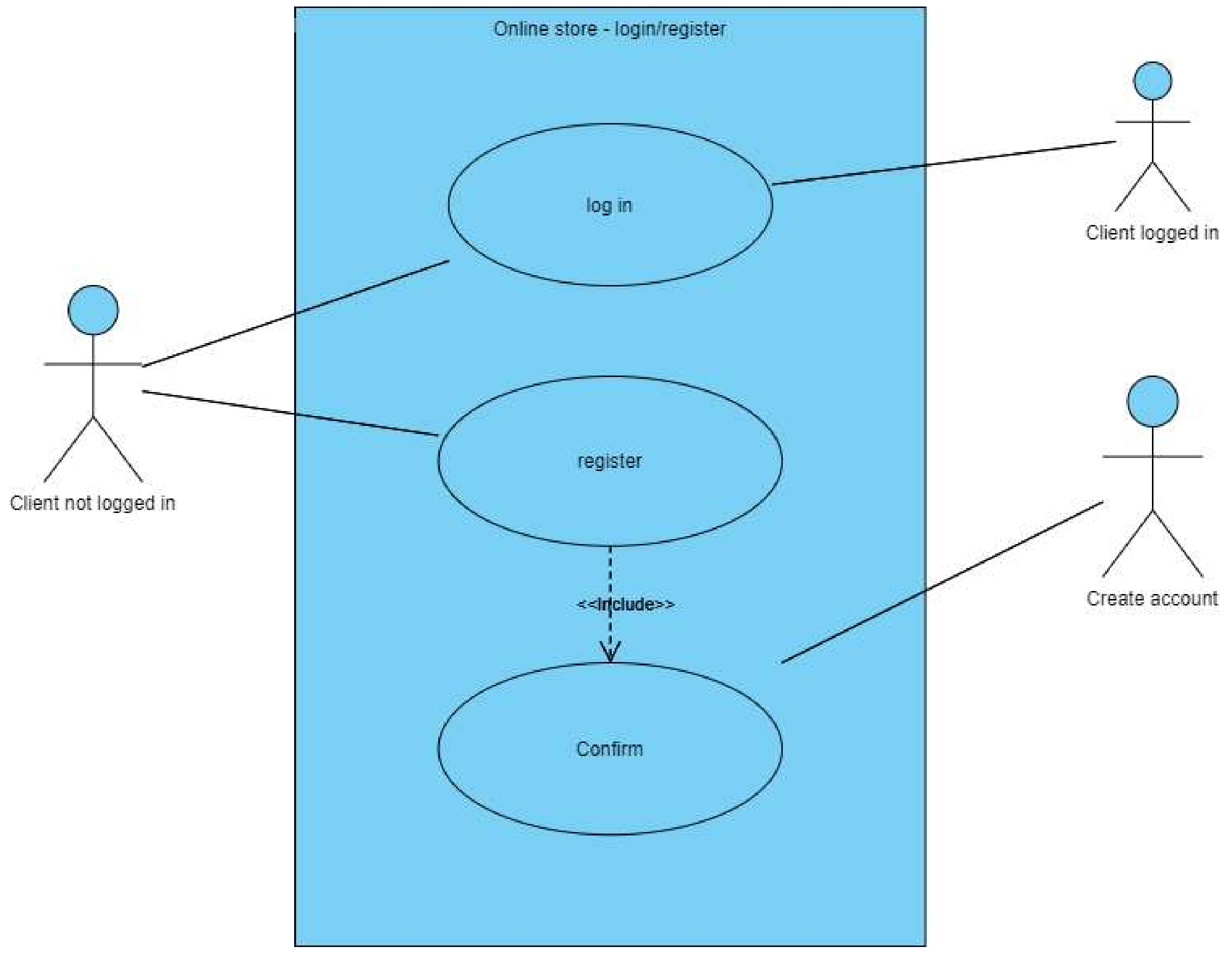
- *register* – creates a new token for the user given in JSON format, and sends an activation link to his email address
- *confirm* – confirms the token specified in the parameter and adds the user to the database

Use cases

- **shopping**



- register/login


Online store - login/register
log in
Client logged in
Client not logged in
register
<<Include>>
Create account
Confirm