```
package com.example.springsecurityapplication.controllers;

import com.example.springsecurityapplication.enumm.Status;
import com.example.springsecurityapplication.models.Cart;
import com.example.springsecurityapplication.models.Order;
import com.example.springsecurityapplication.models.Product;
import com.example.springsecurityapplication.repositories.CartRepository;
import
com.example.springsecurityapplication.repositories.OrderRepository;
import com.example.springsecurityapplication.security.PersonDetails;
import com.example.springsecurityapplication.services.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@Controller
public class UserController {

    private final OrderRepository orderRepository;

    private final CartRepository cartRepository;
    private final ProductService productService;

    @Autowired
    public UserController(OrderRepository orderRepository, CartRepository
cartRepository, ProductService productService) {
        this.orderRepository = orderRepository;
        this.cartRepository = cartRepository;
        this.productService = productService;
    }

    @GetMapping("/index")
    public String index(Model model){
        // Получае объект аутентификации - > с помощью
SecurityContextHolder обращаемся к контексту и на нем вызываем метод
аутентификации. По сути из потока для текущего пользователя мы получаем
объект, который был положен в сессию после аутентификации пользователя
//        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
//
//        // Преобразовываем объект аутентификации в специальный объект
класса по работе с пользователями
//        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();
//        System.out.println("ID пользователя: " +
personDetails.getPerson().getId());
//        System.out.println("Логин пользователя: " +
personDetails.getPerson().getLogin());
//        System.out.println("Пароль пользователя: " +
personDetails.getPerson().getPassword());

        // Получае объект аутентификации - > с помощью
SecurityContextHolder обращаемся к контексту и на нем вызываем метод
```

```
аутентификации. По сути из потока для текущего пользователя мы получаем
объект, который был положен в сессию после аутентификации пользователя
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

        // Преобразовываем объект аутентификации в специальный объект
класса по работе с пользователями
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();

        String role = personDetails.getPerson().getRole();

        if(role.equals("ROLE_ADMIN")){
            return "redirect:/admin";
        }
        model.addAttribute("products", productService.getAllProduct());
        return "user/index";
    }

    // Добавить товар в корзину
    @GetMapping("/cart/add/{id}")
    public String addProductInCart(@PathVariable("id") int id, Model
model){
        Product product = productService.getProductId(id);
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();
        int id_person = personDetails.getPerson().getId();
        Cart cart = new Cart(id_person, product.getId());
        cartRepository.save(cart);
        return "redirect:/cart";
    }

    @GetMapping("/cart")
    public String cart(Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();
        int id_person = personDetails.getPerson().getId();
        List<Cart> cartList = cartRepository.findByPersonId(id_person);
        List<Product> productList = new ArrayList<>();
        for (Cart cart: cartList) {

productList.add(productService.getProductId(cart.getProductId()));
        }

        float price = 0;
        for (Product product: productList) {
            price += product.getPrice();
        }

        model.addAttribute("price", price);
        model.addAttribute("cart_product", productList);
        return "user/cart";
    }

    @GetMapping("/info/{id}")
    public String infoProduct(@PathVariable("id") int id, Model model){
        model.addAttribute("product", productService.getProductId(id));
```

```
        return "product/infoProduct";
    }

    @GetMapping("/cart/delete/{id}")
    public String deleteProductFromCart(Model model, @PathVariable("id")
int id){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();
        int id_person = personDetails.getPerson().getId();
        cartRepository.deleteCartById(id, id_person);
        return "redirect:/cart";
    }

//    @PostMapping("/search")
//    public String productSearch(@RequestParam("search") String search,
@RequestParam("ot") String ot, @RequestParam("do") String Do,
@RequestParam(value = "price", required = false, defaultValue = "")
String price, @RequestParam(value = "category", required = false,
defaultValue = "") String category, Model model){
//        System.out.println("lf");
//        return "redirect:/product";
//    }

    @GetMapping("/order/create")
    public String createOrder(){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();
        int id_person = personDetails.getPerson().getId();
        List<Cart> cartList = cartRepository.findByPersonId(id_person);
        List<Product> productList = new ArrayList<>();
        for (Cart cart: cartList) {

productList.add(productService.getProductId(cart.getProductId()));
        }

        String uuid = UUID.randomUUID().toString();

        for (Product product: productList) {
            Order newOrder = new Order(uuid, 1, product.getPrice(),
Status.Оформлен, product, personDetails.getPerson());
            orderRepository.save(newOrder);
            cartRepository.deleteCartById(product.getId(), id_person);
        }
        return "redirect:/orders";
    }

    @GetMapping("/orders")
    public String ordersUser(Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();
        List<Order> orderList =
orderRepository.findByPerson(personDetails.getPerson());
        model.addAttribute("orders", orderList);
        return "/user/orders";
    }
```

```
}
```