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

import com.example.springsecurityapplication.models.Image;
import com.example.springsecurityapplication.models.Product;
import
com.example.springsecurityapplication.repositories.CategoryRepository;
import com.example.springsecurityapplication.security.PersonDetails;
import com.example.springsecurityapplication.services.ProductService;
import com.example.springsecurityapplication.util.ProductValidator;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
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.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import javax.validation.Valid;
import java.io.File;
import java.io.IOException;
import java.util.UUID;

@Controller
@RequestMapping("/admin")
//@PreAuthorize("hasAnyAuthority('ROLE_ADMIN')")
public class AdminController {

    @Value("${upload.path}")
    private String uploadPath;

    private final ProductValidator productValidator;
    private final ProductService productService;

    private final CategoryRepository categoryRepository;

    @Autowired
    public AdminController(ProductValidator productValidator,
ProductService productService, CategoryRepository categoryRepository) {
        this.productValidator = productValidator;
        this.productService = productService;
        this.categoryRepository = categoryRepository;
    }

    //    @PreAuthorize("hasRole('ROLE_ADMIN') and hasRole('')")
//@PreAuthorize("hasRole('ROLE_ADMIN') or hasRole('')")

    // Метод по отображению главной страницы администратора с выводом
товаров
    @GetMapping()
    public String admin(Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        PersonDetails personDetails = (PersonDetails)
authentication.getPrincipal();

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

        if(role.equals("ROLE_USER")){
            return "redirect:/index";
```

```
        }
        model.addAttribute("products", productService.getAllProduct());
        return "admin/admin";
    }

    // Метод по отображению формы добавление
    @GetMapping("/product/add")
    public String addProduct(Model model){
        model.addAttribute("product", new Product());
        model.addAttribute("category", categoryRepository.findAll());
//        System.out.println(categoryRepository.findAll().size());
        return "product/addProduct";
    }

    // Метод по добавлению объекта с формы в таблицу product
    @PostMapping("/product/add")
    public String addProduct(@ModelAttribute("product") @Valid Product
product, BindingResult bindingResult, @RequestParam("file_one")
MultipartFile file_one, @RequestParam("file_two") MultipartFile file_two,
@RequestParam("file_three") MultipartFile file_three,
@RequestParam("file_four") MultipartFile file_four,
@RequestParam("file_five") MultipartFile file_five) throws IOException {

        productValidator.validate(product, bindingResult);
        if(bindingResult.hasErrors()){
            return "product/addProduct";
        }
        // Проверка на пустоту файла
        if(!file_one.getOriginalFilename().isEmpty()){
            // Дирректория по сохранению файла
            File uploadDir = new File(uploadPath);
            // Если данной дирректории по пути не сущетсвует
            if(!uploadDir.exists()){
                // Создаем данную дирректорию
                uploadDir.mkdir();
            }
            // Создаем уникальное имя файла
            // UUID представляет неищменный универсальный уникальный
идентификатор
            String uuidFile = UUID.randomUUID().toString();
            // file_one.getOriginalFilename() - наименование файла с
формы
            String resultFileName = uuidFile + "." +
file_one.getOriginalFilename();
            // Загружаем файл по указаннопу пути
            file_one.transferTo(new File(uploadPath + "/" +
resultFileName));
            Image image = new Image();
            image.setProduct(product);
            image.setFileName(resultFileName);
            product.addImageProduct(image);
        }

        // Проверка на пустоту файла
        if(!file_two.getOriginalFilename().isEmpty()){
            // Дирректория по сохранению файла
            File uploadDir = new File(uploadPath);
            // Если данной дирректории по пути не сущетсвует
            if(!uploadDir.exists()){
                // Создаем данную дирректорию
                uploadDir.mkdir();
```

```
            }
            // Создаем уникальное имя файла
            // UUID представляет неищменный универсальный уникальный
идентификатор
            String uuidFile = UUID.randomUUID().toString();
            // file_one.getOriginalFilename() - наименование файла с
формы
            String resultFileName = uuidFile + "." +
file_two.getOriginalFilename();
            // Загружаем файл по указаннопу пути
            file_two.transferTo(new File(uploadPath + "/" +
resultFileName));
            Image image = new Image();
            image.setProduct(product);
            image.setFileName(resultFileName);
            product.addImageProduct(image);
        }

        // Проверка на пустоту файла
        if(!file_three.getOriginalFilename().isEmpty()){
            // Дирректория по сохранению файла
            File uploadDir = new File(uploadPath);
            // Если данной дирректории по пути не сущетсвует
            if(!uploadDir.exists()){
                // Создаем данную дирректорию
                uploadDir.mkdir();
            }
            // Создаем уникальное имя файла
            // UUID представляет неищменный универсальный уникальный
идентификатор
            String uuidFile = UUID.randomUUID().toString();
            // file_one.getOriginalFilename() - наименование файла с
формы
            String resultFileName = uuidFile + "." +
file_three.getOriginalFilename();
            // Загружаем файл по указаннопу пути
            file_three.transferTo(new File(uploadPath + "/" +
resultFileName));
            Image image = new Image();
            image.setProduct(product);
            image.setFileName(resultFileName);
            product.addImageProduct(image);
        }

        // Проверка на пустоту файла
        if(!file_four.getOriginalFilename().isEmpty()){
            // Дирректория по сохранению файла
            File uploadDir = new File(uploadPath);
            // Если данной дирректории по пути не сущетсвует
            if(!uploadDir.exists()){
                // Создаем данную дирректорию
                uploadDir.mkdir();
            }
            // Создаем уникальное имя файла
            // UUID представляет неищменный универсальный уникальный
идентификатор
            String uuidFile = UUID.randomUUID().toString();
            // file_one.getOriginalFilename() - наименование файла с
формы
            String resultFileName = uuidFile + "." +
file_four.getOriginalFilename();
```

```
            // Загружаем файл по указаннопу пути
            file_four.transferTo(new File(uploadPath + "/" +
resultFileName));
            Image image = new Image();
            image.setProduct(product);
            image.setFileName(resultFileName);
            product.addImageProduct(image);
        }

        // Проверка на пустоту файла
        if(!file_five.getOriginalFilename().isEmpty()){
            // Дирректория по сохранению файла
            File uploadDir = new File(uploadPath);
            // Если данной дирректории по пути не сущетсвует
            if(!uploadDir.exists()){
                // Создаем данную дирректорию
                uploadDir.mkdir();
            }
            // Создаем уникальное имя файла
            // UUID представляет неищменный универсальный уникальный
идентификатор
            String uuidFile = UUID.randomUUID().toString();
            // file_one.getOriginalFilename() - наименование файла с
формы
            String resultFileName = uuidFile + "." +
file_five.getOriginalFilename();
            // Загружаем файл по указаннопу пути
            file_five.transferTo(new File(uploadPath + "/" +
resultFileName));
            Image image = new Image();
            image.setProduct(product);
            image.setFileName(resultFileName);
            product.addImageProduct(image);
        }

        productService.saveProduct(product);
        return "redirect:/admin";
    }

    // Метод по удалению товара по id
    @GetMapping("/product/delete/{id}")
    public String deleteProduct(@PathVariable("id") int id){
        productService.deleteProduct(id);
        return "redirect:/admin";
    }

    // Метод по получению товара по id и отображение шаблона
редактирования
    @GetMapping("/product/edit/{id}")
    public String editProduct(@PathVariable("id") int id, Model model){
        model.addAttribute("editProduct",
productService.getProductId(id));
        model.addAttribute("category", categoryRepository.findAll());
        return "product/editProduct";
    }

    @PostMapping("/product/edit/{id}")
    public String editProduct(@ModelAttribute("editProduct") Product
product, @PathVariable("id") int id){
        productService.updateProduct(id, product);
        return "redirect:/admin";
```

```
        }


}
```