|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **федеральное государственное бюджетное образовательное учреждение высшего образования «Российский экономический университет имени Г.В. Плеханова»**  **Московский приборостроительный техникум** |

ОТЧЕТ

по учебной практике

УП.04.01 Внедрение и поддержка программного обеспечения\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Профессионального модуля ПМ.04 Сопровождение и обслуживание программного обеспечения компьютерных систем .

Специальность 09.02.07 Информационные системы и программирование \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Студент Панфило Ярослав Валерьевич.

*(фамилия, имя, отчество)*

Группа Т50-1-20

Руководитель по практической подготовке от техникума

Пахомов Даниил Александрович.

*(фамилия, имя, отчество)*

«\_\_» \_\_\_\_\_\_ 2023 года

**СОДЕРЖАНИЕ**

[ИТОГОВЫЙ ПРОЕКТ 3](#_Toc152807349)

[Тема индивидуального проекта 3](#_Toc152807350)

[Описание предметной области 3](#_Toc152807351)

[Диаграммы БД 4](#_Toc152807352)

[Словарь данных 5](#_Toc152807353)

[Скрипт БД 7](#_Toc152807354)

[Код программы 13](#_Toc152807355)

[Работа программы 112](#_Toc152807356)

ИТОГОВЫЙ ПРОЕКТ

Тема индивидуального проекта

Тема индивидуального проекта: интернет магазин электронной техники.

Описание предметной области

Предметная область интернет-магазина электронной техники. Система состоит из нескольких подсистем, каждая из которых обладает определенными функциональными возможностями.

Подсистема "Товар" отвечает за просмотром каталога. В ней доступны основные операции, такие как добавление товара в заказ, их удаление. Эта подсистема доступна обычным пользователям, что позволяет им осуществлять поиск, просматривать информацию о товаре магазина и пользоваться поиском.

Другая подсистема, "Панель модератора", предоставляет возможность модераторам управлять списком. Она позволяет добавлять, удалять, просматривать и изменять данные.

Подсистема "Админ" предоставляет функционал для админов магазина, который позволяет им управлять ролями пользователей. Она включает возможность добавлять, просматривать и изменять роли в системе.

Система также включает подсистемы "Авторизация" и "Регистрация", которые позволяют пользователям создавать учетные записи и аутентифицироваться в системе. Авторизация обеспечивает безопасный доступ для пользователей, а регистрация позволяет им получить доступ к функционалу интернет-магазина.

Диаграммы БД



Рисунок Инфологическая модель данных

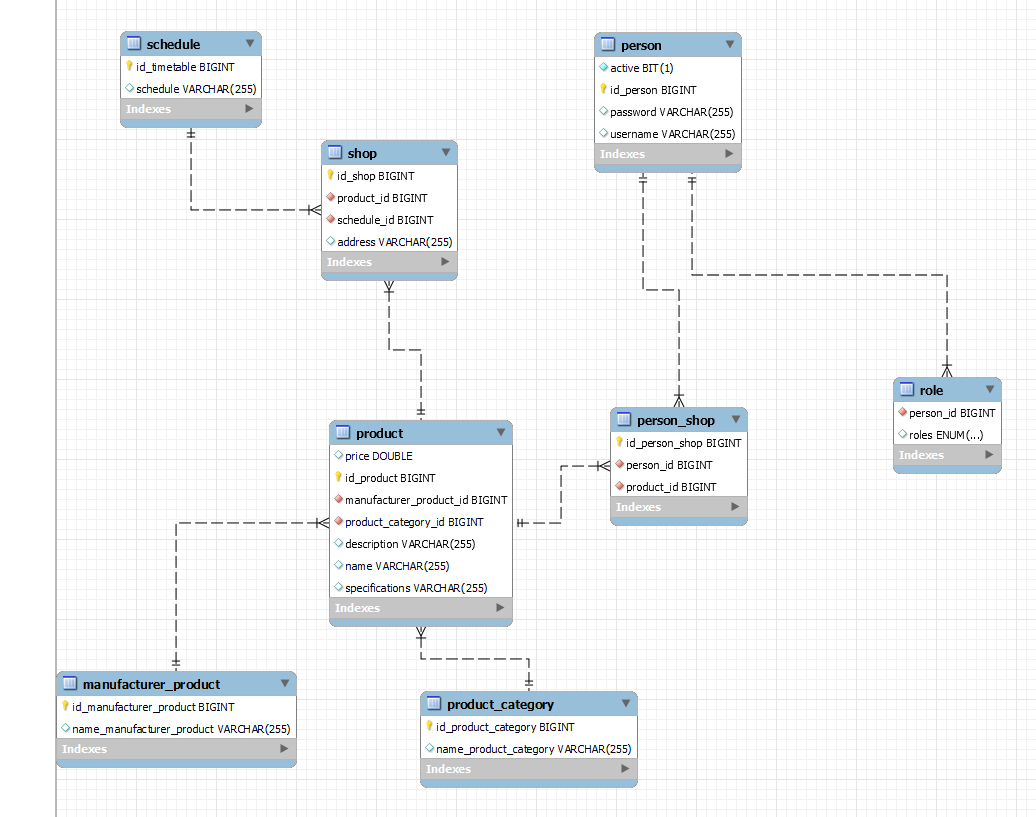


Рисунок 2 Даталогическая модель данных

Словарь данных

Таблица 1 Словарь данных

Таблица 1 – Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Тип данных | Описание |
| manufacturer\_product | | | |
| 1 | id\_manufacturer\_product | bigint | Первичный ключ |
| 2 | name\_manufacturer\_product | varchar(255) | Название производителя |
| person | | | |
| 1 | id\_person | bigint | Первичный ключ |
| 2 | username | varchar(255) | Имя пользователя |
| 3 | password | varchar(255) | Хэшированный пароль пользователя |
| 4 | active | bit(1) | Статус пользователя |
| person\_shop | | | |
| 1 | id\_person\_shop | bigint | Первичный ключ |
| 2 | person\_id | bigint | Внешний ключ таблицы «person» |
| 3 | product\_id | bigint | Внешний ключ таблицы «product» |
| product | | | |
| 1 | id\_product | bigint | Первичный ключ |
| 2 | name | varchar(255) | Название продукта |
| 3 | description | varchar(255) | Описание продукта |
| 4 | specifications | varchar(255) | Характеристика продукта |
| 5 | price | double | Цена продукта |
| 6 | manufacturer\_product\_id | bigint | Внешний ключ таблицы «manufacturer\_product» |
| 7 | product\_category\_id | bigint | Внешний ключ таблицы «product\_category» |
| product\_category | | | |
| 1 | id\_product\_category | bigint AI PK | Первичный ключ |
| 2 | name\_product\_category | varchar(255) | Название категории продукта |
| role | | | |
| 1 | person\_id |  | Внешний ключ таблицы «person» |
| 2 | roles | enum('ADMIN','EMPLOYEE','USER') | Перечисление ролей |
| schedule | | | |
| 1 | id\_timetable | bigint AI PK | Первичный ключ |
| 2 | schedule | varchar(255) | График работы магазина |
| shop | | | |
| 1 | id\_shop | bigint | Первичный ключ |
| 2 | product\_id | bigint | Внешний ключ таблицы «product» |
| 3 | schedule\_id | bigint | Внешний ключ таблицы « schedule » |
| 4 | address | varchar(255) | Адресс магазина |

Скрипт БД

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE SCHEMA IF NOT EXISTS `yp\_0401` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;

USE `yp\_0401` ;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`manufacturer\_product` (

`id\_manufacturer\_product` BIGINT NOT NULL AUTO\_INCREMENT,

`name\_manufacturer\_product` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id\_manufacturer\_product`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`person` (

`active` BIT(1) NOT NULL,

`id\_person` BIGINT NOT NULL AUTO\_INCREMENT,

`password` VARCHAR(255) NULL DEFAULT NULL,

`username` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id\_person`))

ENGINE = InnoDB

AUTO\_INCREMENT = 3

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`product\_category` (

`id\_product\_category` BIGINT NOT NULL AUTO\_INCREMENT,

`name\_product\_category` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id\_product\_category`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`product` (

`price` DOUBLE NULL DEFAULT NULL,

`id\_product` BIGINT NOT NULL AUTO\_INCREMENT,

`manufacturer\_product\_id` BIGINT NOT NULL,

`product\_category\_id` BIGINT NOT NULL,

`description` VARCHAR(255) NULL DEFAULT NULL,

`name` VARCHAR(255) NULL DEFAULT NULL,

`specifications` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id\_product`),

INDEX `FK7jr6u2ktm1qbbhnufufpw0cu9` (`manufacturer\_product\_id` ASC) VISIBLE,

INDEX `FKcwclrqu392y86y0pmyrsi649r` (`product\_category\_id` ASC) VISIBLE,

CONSTRAINT `FK7jr6u2ktm1qbbhnufufpw0cu9`

FOREIGN KEY (`manufacturer\_product\_id`)

REFERENCES `yp\_0401`.`manufacturer\_product` (`id\_manufacturer\_product`),

CONSTRAINT `FKcwclrqu392y86y0pmyrsi649r`

FOREIGN KEY (`product\_category\_id`)

REFERENCES `yp\_0401`.`product\_category` (`id\_product\_category`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`person\_shop` (

`id\_person\_shop` BIGINT NOT NULL AUTO\_INCREMENT,

`person\_id` BIGINT NOT NULL,

`product\_id` BIGINT NOT NULL,

PRIMARY KEY (`id\_person\_shop`),

INDEX `FK90viqdhyiowggaciyrqg25pgx` (`person\_id` ASC) VISIBLE,

INDEX `FK5p117jivicrufh5tiap0nlck8` (`product\_id` ASC) VISIBLE,

CONSTRAINT `FK5p117jivicrufh5tiap0nlck8`

FOREIGN KEY (`product\_id`)

REFERENCES `yp\_0401`.`product` (`id\_product`),

CONSTRAINT `FK90viqdhyiowggaciyrqg25pgx`

FOREIGN KEY (`person\_id`)

REFERENCES `yp\_0401`.`person` (`id\_person`))

ENGINE = InnoDB

AUTO\_INCREMENT = 4

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`role` (

`person\_id` BIGINT NOT NULL,

`roles` ENUM('ADMIN', 'EMPLOYEE', 'USER') NULL DEFAULT NULL,

INDEX `FKp74rfg21c55d8eebfl00w5451` (`person\_id` ASC) VISIBLE,

CONSTRAINT `FKp74rfg21c55d8eebfl00w5451`

FOREIGN KEY (`person\_id`)

REFERENCES `yp\_0401`.`person` (`id\_person`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`schedule` (

`id\_timetable` BIGINT NOT NULL AUTO\_INCREMENT,

`schedule` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id\_timetable`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `yp\_0401`.`shop` (

`id\_shop` BIGINT NOT NULL AUTO\_INCREMENT,

`product\_id` BIGINT NOT NULL,

`schedule\_id` BIGINT NOT NULL,

`address` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id\_shop`),

INDEX `FKj9eee0t9rx8u74cwuv17677xc` (`product\_id` ASC) VISIBLE,

INDEX `FKgsknmlvdvs204voeudc3nlt3x` (`schedule\_id` ASC) VISIBLE,

CONSTRAINT `FKgsknmlvdvs204voeudc3nlt3x`

FOREIGN KEY (`schedule\_id`)

REFERENCES `yp\_0401`.`schedule` (`id\_timetable`),

CONSTRAINT `FKj9eee0t9rx8u74cwuv17677xc`

FOREIGN KEY (`product\_id`)

REFERENCES `yp\_0401`.`product` (`id\_product`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

Код программы

API\_YP

ManufacturerProduct.java:

package com.example.api\_yp.Models;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Size;

import java.util.List;

@Entity

public class ManufacturerProduct {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idManufacturerProduct;

@Size(min = 3, message = "The nameManufacturerProduct cannot be less than 3 characters")

@NotBlank(message = "NameManufacturerProduct is required")

private String nameManufacturerProduct;

public ManufacturerProduct() {

}

public ManufacturerProduct(Long idManufacturerProduct, String nameManufacturerProduct) {

this.idManufacturerProduct = idManufacturerProduct;

this.nameManufacturerProduct = nameManufacturerProduct;

}

public Long getIdManufacturerProduct() {

return idManufacturerProduct;

}

public void setIdManufacturerProduct(Long idManufacturerProduct) {

this.idManufacturerProduct = idManufacturerProduct;

}

public String getNameManufacturerProduct() {

return nameManufacturerProduct;

}

public void setNameManufacturerProduct(String nameManufacturerProduct) {

this.nameManufacturerProduct = nameManufacturerProduct;

}

}

Person.java:

package com.example.api\_yp.Models;

import jakarta.persistence.\*;

import jakarta.validation.constraints.\*;

import java.util.List;

import java.util.Set;

@Entity

public class Person {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idPerson;

@Size(min = 3, max = 25, message = "The login cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Login is required")

private String username;

@Size(min = 8, message = "The password cannot be less than 8 characters")

@Pattern(regexp = "^.\*(?=.\*[a-zA-Z])(?=.\*\\d)(?=.\*[!#$%&? \"]).\*$" , message = "The password must contain uppercase and lowercase letters, numbers and special characters")

@NotBlank(message = "Password is required")

private String password;

private boolean active;

@ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)

@CollectionTable(name = "Role", joinColumns = @JoinColumn(name = "personId"))

@Enumerated(EnumType.STRING)

private Set<Role> roles;

public Person() {

}

public Person(Long idPerson, String username, String password, boolean active, Set<Role> roles) {

this.idPerson = idPerson;

this.username = username;

this.password = password;

this.active = active;

this.roles = roles;

}

public Long getIdPerson() {

return idPerson;

}

public void setIdPerson(Long idPerson) {

this.idPerson = idPerson;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

public Set<Role> getRoles() {

return roles;

}

public void setRoles(Set<Role> roles) {

this.roles = roles;

}

}

PersonShop.java:

package com.example.api\_yp.Models;

import jakarta.persistence.\*;

@Entity

public class PersonShop {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idPersonShop;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name="personId", nullable=false)

private Person person;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name="productId", nullable=false)

private Product product;

public PersonShop() {

}

public PersonShop(Long idPersonShop, Person person, Product product) {

this.idPersonShop = idPersonShop;

this.person = person;

this.product = product;

}

public Long getIdPersonShop() {

return idPersonShop;

}

public void setIdPersonShop(Long idPersonShop) {

this.idPersonShop = idPersonShop;

}

public Person getPerson() {

return person;

}

public void setPerson(Person person) {

this.person = person;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

}

Product.java:

package com.example.api\_yp.Models;

import com.fasterxml.jackson.annotation.JsonBackReference;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Pattern;

import jakarta.validation.constraints.Size;

import java.util.List;

@Entity

public class Product {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idProduct;

@Size(min = 3, message = "The name cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Name is required")

private String name;

@Size(min = 10, message = "The specifications cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Specifications is required")

private String specifications;

@Size(min = 10, message = "The description cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Description is required")

private String description;

@NotNull

private Double price;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name="manufacturerProductId", nullable=false)

private ManufacturerProduct manufacturerProduct;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name="productCategoryId", nullable=false)

private ProductCategory productCategory;

public Product() {

}

public Product(Long idProduct, String name, String specifications, String description, Double price, ManufacturerProduct manufacturerProduct, ProductCategory productCategory) {

this.idProduct = idProduct;

this.name = name;

this.specifications = specifications;

this.description = description;

this.price = price;

this.manufacturerProduct = manufacturerProduct;

this.productCategory = productCategory;

}

public Long getIdProduct() {

return idProduct;

}

public void setIdProduct(Long idProduct) {

this.idProduct = idProduct;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSpecifications() {

return specifications;

}

public void setSpecifications(String specifications) {

this.specifications = specifications;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public ManufacturerProduct getManufacturerProduct() {

return manufacturerProduct;

}

public void setManufacturerProduct(ManufacturerProduct manufacturerProduct) {

this.manufacturerProduct = manufacturerProduct;

}

public ProductCategory getProductCategory() {

return productCategory;

}

public void setProductCategory(ProductCategory productCategory) {

this.productCategory = productCategory;

}

}

ProductCategory.java:

package com.example.api\_yp.Models;

import com.fasterxml.jackson.annotation.JsonBackReference;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Size;

import java.util.List;

@Entity

public class ProductCategory {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idProductCategory;

@Size(min = 3, message = "The nameProductCategory cannot be less than 3 characters")

@NotBlank(message = "NameProductCategory is required")

private String nameProductCategory;

public ProductCategory() {

}

public ProductCategory(Long idProductCategory, String nameProductCategory) {

this.idProductCategory = idProductCategory;

this.nameProductCategory = nameProductCategory;

}

public Long getIdProductCategory() {

return idProductCategory;

}

public void setIdProductCategory(Long idProductCategory) {

this.idProductCategory = idProductCategory;

}

public String getNameProductCategory() {

return nameProductCategory;

}

public void setNameProductCategory(String nameProductCategory) {

this.nameProductCategory = nameProductCategory;

}

}

Role.java:

package com.example.api\_yp.Models;

public enum Role {

USER,

ADMIN,

EMPLOYEE;

}

Schedule.java:

package com.example.api\_yp.Models;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

@Entity

public class Schedule {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idTimetable;

@NotNull

@NotBlank(message = "Schedule is required")

private String schedule;

public Schedule() {

}

public Schedule(Long idTimetable, String schedule) {

this.idTimetable = idTimetable;

this.schedule = schedule;

}

public Long getIdTimetable() {

return idTimetable;

}

public void setIdTimetable(Long idTimetable) {

this.idTimetable = idTimetable;

}

public String getSchedule() {

return schedule;

}

public void setSchedule(String schedule) {

this.schedule = schedule;

}

}

Shop.java:

package com.example.api\_yp.Models;

import com.fasterxml.jackson.annotation.JsonBackReference;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Size;

import java.util.List;

@Entity

public class Shop {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Long idShop;

@Size(min = 5, message = "The address cannot be less than 5 characters")

@NotBlank(message = "Address is required")

private String address;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name="productId", nullable=false)

private Product product;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name="scheduleId", nullable=false)

private Schedule schedule;

public Shop() {

}

public Shop(Long idShop, String address, Product product, Schedule schedule) {

this.idShop = idShop;

this.address = address;

this.product = product;

this.schedule = schedule;

}

public Long getIdShop() {

return idShop;

}

public void setIdShop(Long idShop) {

this.idShop = idShop;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public Schedule getSchedule() {

return schedule;

}

public void setSchedule(Schedule schedule) {

this.schedule = schedule;

}

}

ManufacturerProductRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.ManufacturerProduct;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ManufacturerProductRepository extends JpaRepository<ManufacturerProduct,Long> {

}

PersonRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.Person;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.repository.query.Param;

public interface PersonRepository extends JpaRepository<Person,Long> {

Person findUserByUsername (String username);

}

PersonShopRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.Person;

import com.example.api\_yp.Models.PersonShop;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface PersonShopRepository extends JpaRepository<PersonShop,Long> {

@Query(value="SELECT \* FROM person\_shop where person\_id = :c", nativeQuery=true)

List<PersonShop> findPersonShopByPerson (@Param("c")Long idPerson);

}

ProductCategoryRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.ProductCategory;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductCategoryRepository extends JpaRepository<ProductCategory,Long> {

}

ProductRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.Product;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface ProductRepository extends JpaRepository<Product,Long> {

@Query(value = "SELECT \* FROM product inner join shop on product\_id = id\_product where address =:address and name=:name", nativeQuery = true)

List<Product> findProductByNameAndAddress (@Param("address")String address, @Param("name")String name);

}

ScheduleRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.Schedule;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ScheduleRepository extends JpaRepository<Schedule,Long> {

}

ShopRepository.java:

package com.example.api\_yp.Repositories;

import com.example.api\_yp.Models.Shop;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ShopRepository extends JpaRepository<Shop,Long> {

}

ManufacturerProductController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.ManufacturerProduct;

import com.example.api\_yp.Repositories.ManufacturerProductRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

public class ManufacturerProductController {

@Autowired

private ManufacturerProductRepository manufacturerProductRepository;

@GetMapping("/manufacturerProduct")

public ResponseEntity<List<ManufacturerProduct>> getManufacturerProduct() {

List<ManufacturerProduct> manufacturerProducts = manufacturerProductRepository.findAll();

return new ResponseEntity<>(manufacturerProducts, HttpStatus.OK);

}

@GetMapping("/manufacturerProduct/{idManufacturerProduct}")

public ResponseEntity<ManufacturerProduct> oneManufacturerProduct(@PathVariable Long idManufacturerProduct) {

Optional<ManufacturerProduct> optionalManufacturerProduct = manufacturerProductRepository.findById(idManufacturerProduct);

if (optionalManufacturerProduct.isPresent()) {

return new ResponseEntity<>(optionalManufacturerProduct.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("/manufacturerProduct")

public ResponseEntity<ManufacturerProduct> createManufacturerProduct(@Valid @RequestBody ManufacturerProduct manufacturerProductRequest) {

ManufacturerProduct manufacturerProduct = manufacturerProductRepository.save(manufacturerProductRequest);

return new ResponseEntity<>(manufacturerProduct, HttpStatus.CREATED);

}

@PutMapping("/manufacturerProduct/{idManufacturerProduct}")

public ResponseEntity<ManufacturerProduct> updateManufacturerProduct(@PathVariable Long idManufacturerProduct,

@Valid @RequestBody ManufacturerProduct manufacturerProductRequest) {

Optional<ManufacturerProduct> optionalManufacturerProduct = manufacturerProductRepository.findById(idManufacturerProduct);

if (optionalManufacturerProduct.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

ManufacturerProduct manufacturerProduct = optionalManufacturerProduct.get();

manufacturerProduct.setIdManufacturerProduct(manufacturerProductRequest.getIdManufacturerProduct());

manufacturerProduct.setNameManufacturerProduct(manufacturerProductRequest.getNameManufacturerProduct());

ManufacturerProduct manufacturerProductUpdate = manufacturerProductRepository.save(manufacturerProduct);

return new ResponseEntity<>(manufacturerProductUpdate, HttpStatus.OK);

}

@DeleteMapping("/manufacturerProduct/{idManufacturerProduct}")

public ResponseEntity<?> deleteManufacturerProduct(@PathVariable Long idManufacturerProduct) {

Optional<ManufacturerProduct> optionalManufacturerProduct = manufacturerProductRepository.findById(idManufacturerProduct);

if (optionalManufacturerProduct.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

ManufacturerProduct manufacturerProduct = optionalManufacturerProduct.get();

manufacturerProductRepository.delete(manufacturerProduct);

return new ResponseEntity<>(HttpStatus.OK);

}

}

PersonController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.Person;

import com.example.api\_yp.Models.Role;

import com.example.api\_yp.Repositories.PersonRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.web.bind.annotation.\*;

import java.util.Collections;

import java.util.List;

import java.util.Optional;

@RestController

public class PersonController {

@Autowired

private PersonRepository personRepository;

public PasswordEncoder getPasswordEncoder(){

return new BCryptPasswordEncoder(8);

}

@PostMapping("/signUp")

public ResponseEntity<Person> signUp(@Valid @RequestBody Person person) {

Person personFromDB = personRepository.findUserByUsername(person.getUsername());

if (personFromDB != null)

{

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

person.setRoles(Collections.singleton(Role.USER));

person.setPassword(person.getPassword());

person.setActive(true);

personRepository.save(person);

return new ResponseEntity<>(person, HttpStatus.CREATED);

}

@PostMapping("/searchPerson")

public ResponseEntity<Person> signIn(@Valid @RequestBody Person person) {

Person personFromDB = personRepository.findUserByUsername(person.getUsername());

if (personFromDB == null)

{

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

return new ResponseEntity<>(personFromDB, HttpStatus.OK);

}

@GetMapping("/person")

public ResponseEntity<List<Person>> getPerson() {

List<Person> persons = personRepository.findAll();

return new ResponseEntity<>(persons, HttpStatus.OK);

}

@GetMapping("/person/{idPerson}")

public ResponseEntity<Person> onePerson(@PathVariable Long idPerson) {

Optional<Person> optionalPerson = personRepository.findById(idPerson);

if (optionalPerson.isPresent()) {

return new ResponseEntity<>(optionalPerson.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("/roles")

public ResponseEntity<Role[]> getRole() {

return new ResponseEntity<>(Role.values(), HttpStatus.OK);

}

@PutMapping("/person/{idPerson}")

public ResponseEntity<Person> updatePerson(@PathVariable Long idPerson,

@RequestBody String[] roles) {

Optional<Person> personOptional = personRepository.findById(idPerson);

if (personOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Person person = personOptional.get();

person.getRoles().clear();

if(roles != null)

{

for(String role: roles)

{

person.getRoles().add(Role.valueOf(role));

}

}

Person personUpdate = personRepository.save(person);

return new ResponseEntity<>(personUpdate, HttpStatus.OK);

}

}

PesonShopController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.Person;

import com.example.api\_yp.Models.PersonShop;

import com.example.api\_yp.Models.Shop;

import com.example.api\_yp.Repositories.PersonRepository;

import com.example.api\_yp.Repositories.PersonShopRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@Controller

public class PesonShopController {

@Autowired

private PersonShopRepository personShopRepository;

@Autowired

private PersonRepository personRepository;

@PostMapping("/searchOrder")

public ResponseEntity<List<PersonShop>> searchOrder(@Valid @RequestBody Person person) {

Person personFromDB = personRepository.findUserByUsername(person.getUsername());

if (personFromDB == null)

{

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

List<PersonShop> personShops = personShopRepository.findPersonShopByPerson(personFromDB.getIdPerson());

return new ResponseEntity<>(personShops, HttpStatus.OK);

}

@GetMapping("/personShop/{idPersonShop}")

public ResponseEntity<PersonShop> onePersonShop(@PathVariable Long idPersonShop) {

Optional<PersonShop> optionalPersonShop = personShopRepository.findById(idPersonShop);

if (optionalPersonShop.isPresent()) {

return new ResponseEntity<>(optionalPersonShop.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("/personShop")

public ResponseEntity<PersonShop> createPersonShop(@Valid @RequestBody PersonShop personShopRequest) {

PersonShop personShop = personShopRepository.save(personShopRequest);

return new ResponseEntity<>(personShop, HttpStatus.CREATED);

}

@PutMapping("/personShop/{idPersonShop}")

public ResponseEntity<PersonShop> updatePersonShop(@PathVariable Long idPersonShop,

@Valid @RequestBody PersonShop personShopRequest) {

Optional<PersonShop> optionalPersonShop = personShopRepository.findById(idPersonShop);

if (optionalPersonShop.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

PersonShop personShop = optionalPersonShop.get();

personShop.setIdPersonShop(personShopRequest.getIdPersonShop());

personShop.setProduct(personShopRequest.getProduct());

personShop.setPerson(personShopRequest.getPerson());

PersonShop personShopUpdate = personShopRepository.save(personShop);

return new ResponseEntity<>(personShopUpdate, HttpStatus.OK);

}

@DeleteMapping("/personShop/{idPersonShop}")

public ResponseEntity<?> deletePersonShop(@PathVariable Long idPersonShop) {

Optional<PersonShop> optionalPersonShop = personShopRepository.findById(idPersonShop);

if (optionalPersonShop.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

PersonShop personShop = optionalPersonShop.get();

personShopRepository.delete(personShop);

return new ResponseEntity<>(HttpStatus.OK);

}

}

ProductCategoryController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.ProductCategory;

import com.example.api\_yp.Repositories.ProductCategoryRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

public class ProductCategoryController {

@Autowired

private ProductCategoryRepository productCategoryRepository;

@GetMapping("/productCategory")

public ResponseEntity<List<ProductCategory>> getProductCategory() {

List<ProductCategory> productCategories = productCategoryRepository.findAll();

return new ResponseEntity<>(productCategories, HttpStatus.OK);

}

@GetMapping("/productCategory/{idProductCategory}")

public ResponseEntity<ProductCategory> oneProductCategory(@PathVariable Long idProductCategory) {

Optional<ProductCategory> optionalProductCategory = productCategoryRepository.findById(idProductCategory);

if (optionalProductCategory.isPresent()) {

return new ResponseEntity<>(optionalProductCategory.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("/productCategory")

public ResponseEntity<ProductCategory> createProductCategory(@Valid @RequestBody ProductCategory productCategoryRequest) {

ProductCategory productCategory = productCategoryRepository.save(productCategoryRequest);

return new ResponseEntity<>(productCategory, HttpStatus.CREATED);

}

@PutMapping("/productCategory/{idProductCategory}")

public ResponseEntity<ProductCategory> updateProductCategory(@PathVariable Long idProductCategory,

@Valid @RequestBody ProductCategory productCategoryRequest) {

Optional<ProductCategory> productCategoryOptional = productCategoryRepository.findById(idProductCategory);

if (productCategoryOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

ProductCategory productCategory = productCategoryOptional.get();

productCategory.setIdProductCategory(productCategoryRequest.getIdProductCategory());

productCategory.setNameProductCategory(productCategoryRequest.getNameProductCategory());

ProductCategory productCategoryUpdate = productCategoryRepository.save(productCategory);

return new ResponseEntity<>(productCategoryUpdate, HttpStatus.OK);

}

@DeleteMapping("/productCategory/{idProductCategory}")

public ResponseEntity<?> deleteProductCategory(@PathVariable Long idProductCategory) {

Optional<ProductCategory> productCategoryOptional = productCategoryRepository.findById(idProductCategory);

if (productCategoryOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

ProductCategory productCategory = productCategoryOptional.get();

productCategoryRepository.delete(productCategory);

return new ResponseEntity<>(HttpStatus.OK);

}

}

ProductController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.Product;

import com.example.api\_yp.Repositories.ProductRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

public class ProductController {

@Autowired

private ProductRepository productRepository;

@GetMapping("/product")

public ResponseEntity<List<Product>> getProduct() {

List<Product> products = productRepository.findAll();

return new ResponseEntity<>(products, HttpStatus.OK);

}

@GetMapping("/product/{idProduct}")

public ResponseEntity<Product> oneProduct(@PathVariable Long idProduct) {

Optional<Product> optionalProduct = productRepository.findById(idProduct);

if (optionalProduct.isPresent()) {

return new ResponseEntity<>(optionalProduct.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("/product")

public ResponseEntity<Product> createProduct(@Valid @RequestBody Product productRequest) {

Product product = productRepository.save(productRequest);

return new ResponseEntity<>(product, HttpStatus.CREATED);

}

@PutMapping("/product/{idProduct}")

public ResponseEntity<Product> updateProduct(@PathVariable Long idProduct,

@Valid @RequestBody Product productRequest) {

Optional<Product> productOptional = productRepository.findById(idProduct);

if (productOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Product product = productOptional.get();

product.setIdProduct(productRequest.getIdProduct());

product.setName(productRequest.getName());

product.setDescription(productRequest.getDescription());

product.setSpecifications(productRequest.getSpecifications());

product.setManufacturerProduct(productRequest.getManufacturerProduct());

product.setProductCategory(productRequest.getProductCategory());

Product productUpdate = productRepository.save(product);

return new ResponseEntity<>(productUpdate, HttpStatus.OK);

}

@DeleteMapping("/product/{idProduct}")

public ResponseEntity<?> deleteProduct(@PathVariable Long idProduct) {

Optional<Product> productOptional = productRepository.findById(idProduct);

if (productOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Product product = productOptional.get();

productRepository.delete(product);

return new ResponseEntity<>(HttpStatus.OK);

}

@GetMapping("/product/{address}/{name}")

public ResponseEntity<List<Product>> searchProduct(@PathVariable String address,@PathVariable String name) {

List<Product> products = productRepository.findProductByNameAndAddress(address,name);

return new ResponseEntity<>(products, HttpStatus.OK);

}

}

ScheduleController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.Schedule;

import com.example.api\_yp.Repositories.ScheduleRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

public class ScheduleController {

@Autowired

private ScheduleRepository scheduleRepository;

@GetMapping("/schedule")

public ResponseEntity<List<Schedule>> getSchedule() {

List<Schedule> schedules = scheduleRepository.findAll();

return new ResponseEntity<>(schedules, HttpStatus.OK);

}

@GetMapping("/schedule/{idSchedule}")

public ResponseEntity<Schedule> oneSchedule(@PathVariable Long idSchedule) {

Optional<Schedule> optionalSchedule = scheduleRepository.findById(idSchedule);

if (optionalSchedule.isPresent()) {

return new ResponseEntity<>(optionalSchedule.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("/schedule")

public ResponseEntity<Schedule> createSchedule(@Valid @RequestBody Schedule scheduleRequest) {

Schedule schedule = scheduleRepository.save(scheduleRequest);

return new ResponseEntity<>(schedule, HttpStatus.CREATED);

}

@PutMapping("/schedule/{idSchedule}")

public ResponseEntity<Schedule> updateSchedule(@PathVariable Long idSchedule,

@Valid @RequestBody Schedule scheduleRequest) {

Optional<Schedule> optionalSchedule = scheduleRepository.findById(idSchedule);

if (optionalSchedule.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Schedule schedule = optionalSchedule.get();

schedule.setIdTimetable(scheduleRequest.getIdTimetable());

schedule.setSchedule(scheduleRequest.getSchedule());

Schedule scheduleUpdate = scheduleRepository.save(schedule);

return new ResponseEntity<>(scheduleUpdate, HttpStatus.OK);

}

@DeleteMapping("/schedule/{idSchedule}")

public ResponseEntity<?> deleteSchedule(@PathVariable Long idSchedule) {

Optional<Schedule> optionalSchedule = scheduleRepository.findById(idSchedule);

if (optionalSchedule.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Schedule schedule = optionalSchedule.get();

scheduleRepository.delete(schedule);

return new ResponseEntity<>(HttpStatus.OK);

}

}

ShopController.java:

package com.example.api\_yp.Controllers;

import com.example.api\_yp.Models.Shop;

import com.example.api\_yp.Repositories.ShopRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

public class ShopController {

@Autowired

private ShopRepository shopRepository;

@GetMapping("/shop")

public ResponseEntity<List<Shop>> getShop() {

List<Shop> shops = shopRepository.findAll();

return new ResponseEntity<>(shops, HttpStatus.OK);

}

@GetMapping("/shop/{idShop}")

public ResponseEntity<Shop> oneShop(@PathVariable Long idShop) {

Optional<Shop> optionalShop = shopRepository.findById(idShop);

if (optionalShop.isPresent()) {

return new ResponseEntity<>(optionalShop.get(), HttpStatus.OK);

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("/shop")

public ResponseEntity<Shop> createShop(@Valid @RequestBody Shop shopRequest) {

Shop shop = shopRepository.save(shopRequest);

return new ResponseEntity<>(shop, HttpStatus.CREATED);

}

@PutMapping("/shop/{idShop}")

public ResponseEntity<Shop> updateShop(@PathVariable Long idShop,

@Valid @RequestBody Shop shopRequest) {

Optional<Shop> shopOptional = shopRepository.findById(idShop);

if (shopOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Shop shop = shopOptional.get();

shop.setIdShop(shopRequest.getIdShop());

shop.setAddress(shopRequest.getAddress());

shop.setProduct(shopRequest.getProduct());

shop.setSchedule(shopRequest.getSchedule());

Shop shopUpdate = shopRepository.save(shop);

return new ResponseEntity<>(shopUpdate, HttpStatus.OK);

}

@DeleteMapping("/shop/{idShop}")

public ResponseEntity<?> deleteshop(@PathVariable Long idShop) {

Optional<Shop> shopOptional = shopRepository.findById(idShop);

if (shopOptional.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Shop shop = shopOptional.get();

shopRepository.delete(shop);

return new ResponseEntity<>(HttpStatus.OK);

}

}

application.properties:

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://${MYSQL\_HOST:localhost}:3306/YP\_0401

spring.datasource.username=root

spring.datasource.password=1111

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.mvc.hiddenmethod.filter.enabled=true

* dns

MVC.java:

package com.example.dns.Config;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class MVC implements WebMvcConfigurer {

public void addViewControllers(ViewControllerRegistry registry) {

registry.addViewController("/Authorization").setViewName("Authorization");

}

}

WC.java:

package com.example.dns.Config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.web.client.RestTemplate;

import javax.sql.DataSource;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class WC extends WebSecurityConfigurerAdapter {

@Autowired

private DataSource dataSource;

@Bean

public PasswordEncoder getPasswordEncoder() {

return new BCryptPasswordEncoder(8);

}

@Bean

public RestTemplate restTemplate() {

return new RestTemplate();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication().dataSource(dataSource).passwordEncoder(getPasswordEncoder())

.usersByUsernameQuery("select username, password, active from person where username =?")

.authoritiesByUsernameQuery("select username, roles from person inner join role on id\_person = person\_id where username=?");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests()

.antMatchers("/Authorization","/Registration").permitAll()

.anyRequest()

.authenticated()

.and()

.formLogin()

.loginPage("/Authorization")

.defaultSuccessUrl("/User/Index")

.permitAll()

.and()

.logout()

.permitAll()

.and().csrf().disable().cors().disable();

}

}

AdminController.java:

package com.example.dns.Controllers;

import com.example.dns.Models.Person;

import com.example.dns.Models.Role;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.web.client.RestTemplateBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.security.access.prepost.PreAuthorize;

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.client.RestTemplate;

@PreAuthorize("hasAnyAuthority('ADMIN')")

@RequestMapping("/Admin")

@Controller

public class AdminController {

public String baseUrl = "http://localhost:8080/";

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

@GetMapping("/Index")

public String AdminIndex(Model model) {

Person[] requestGet = getRestTemplate().getForObject(baseUrl+"person", Person[].class);

model.addAttribute("persons",requestGet);

return "/Admin/Index";

}

@GetMapping("/Update/{id}")

public String AdminUpdate(@PathVariable Long id, Model model)

{

Person person = getRestTemplate().getForObject(baseUrl+"person/"+id, Person.class);

model.addAttribute("user",person);

model.addAttribute("roles", Role.values());

return "/Admin/Update";

}

@PostMapping("/Update/{id}")

public String AdminUpdate(@RequestParam(name="roles[]", required = false) String[] roles, @PathVariable Long id)

{

if(roles == null)

{

return "redirect:/Admin/Update/"+id;

}

getRestTemplate().put(baseUrl+"person/"+id, roles, Void.class);

return "redirect:/Admin/Index";

}

}

AuthController.java:

package com.example.dns.Controllers;

import com.example.dns.Models.Person;

import javax.validation.Valid;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.client.RestTemplate;

@Controller

public class AuthController {

public String baseUrl = "http://localhost:8080/";

public PasswordEncoder getPasswordEncoder(){

return new BCryptPasswordEncoder(8);

}

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

@GetMapping("/Registration")

public String Registration(Model model) {

return "Registration";

}

@PostMapping("/Registration")

private String Registration(@Valid Person person, BindingResult bindingResult, Model model)

{

if (bindingResult.hasErrors())

{

return "/Registration";

}

person.setPassword(getPasswordEncoder().encode(person.getPassword()));

Person result = getRestTemplate().postForObject(baseUrl+"signUp",person, Person.class);

return "redirect:/Authorization";

}

}

ModeratorController.java:

package com.example.dns.Controllers;

import com.example.dns.Models.\*;

import javax.validation.Valid;

import org.springframework.security.access.prepost.PreAuthorize;

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.client.RestTemplate;

@PreAuthorize("hasAnyAuthority('EMPLOYEE')")

@RequestMapping("/Moderator")

@Controller

public class ModeratorController {

public String baseUrl = "http://localhost:8080/";

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

@GetMapping("/Index")

public String Index(Model model) {

return "/Moderator/Index";

}

//ManufacturerProduct

@GetMapping("/ManufacturerProduct/Index")

public String ManufacturerProductIndex(Model model) {

ManufacturerProduct[] requestGet = getRestTemplate().getForObject(baseUrl+"manufacturerProduct", ManufacturerProduct[].class);

model.addAttribute("objects",requestGet);

return "/Moderator/ManufacturerProduct/Index";

}

@GetMapping("/ManufacturerProduct/Add")

public String ManufacturerProductAdd(@ModelAttribute("manufacturerProduct") ManufacturerProduct manufacturerProduct, Model model) {

return "/Moderator/ManufacturerProduct/Add";

}

@PostMapping("/ManufacturerProduct/Add")

public String ManufacturerProductPost(@Valid @ModelAttribute("manufacturerProduct") ManufacturerProduct manufacturerProduct, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

return "/Moderator/ManufacturerProduct/Add";

}

ManufacturerProduct requestPost = getRestTemplate().postForObject(baseUrl+"manufacturerProduct",manufacturerProduct, ManufacturerProduct.class);

return "redirect:/Moderator/ManufacturerProduct/Index";

}

@GetMapping("/ManufacturerProduct/Index/{id}")

public String ManufacturerDelete(@PathVariable(value = "id") Long id, Model model) {

getRestTemplate().delete(baseUrl+"manufacturerProduct/"+id);

return "redirect:/Moderator/ManufacturerProduct/Index";

}

@GetMapping("/ManufacturerProduct/Update/{id}")

public String ManufacturerUpdate(@PathVariable Long id, Model model) {

ManufacturerProduct requestGet = getRestTemplate().getForObject(baseUrl+"manufacturerProduct/"+id, ManufacturerProduct.class);

model.addAttribute("object",requestGet);

return "/Moderator/ManufacturerProduct/Update";

}

@PostMapping("/ManufacturerProduct/Update/{id}")

public String ManufacturerUpdate(@PathVariable Long id, @Valid ManufacturerProduct manufacturerProduct, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

return "redirect:/Moderator/ManufacturerProduct/Update/"+id;

}

manufacturerProduct.setIdManufacturerProduct(id);

getRestTemplate().put(baseUrl+"manufacturerProduct/"+id, manufacturerProduct, Void.class);

return "redirect:/Moderator/ManufacturerProduct/Index";

}

//Product

@GetMapping("/Product/Index")

public String ProductIndex(Model model) {

Product[] requestGet = getRestTemplate().getForObject(baseUrl+"product", Product[].class);

model.addAttribute("objects",requestGet);

return "/Moderator/Product/Index";

}

@GetMapping("/Product/Add")

public String ProductAdd(@ModelAttribute("product") Product product, Model model) {

ProductCategory[] requestGetCategory = getRestTemplate().getForObject(baseUrl+"productCategory", ProductCategory[].class);

ManufacturerProduct[] requestGetManufacturer = getRestTemplate().getForObject(baseUrl+"manufacturerProduct", ManufacturerProduct[].class);

model.addAttribute("objectsCategory",requestGetCategory);

model.addAttribute("objectsManufacturer",requestGetManufacturer);

return "/Moderator/Product/Add";

}

@PostMapping("/Product/Add")

public String ProductPost(@Valid @ModelAttribute("product") Product product,

BindingResult bindingResult,

@RequestParam(name="idManufacturerProduct") Long idManufacturerProduct,

@RequestParam(name="idProductCategory") Long idProductCategory,

Model model) {

if (bindingResult.hasErrors())

{

ProductCategory[] requestGetCategory = getRestTemplate().getForObject(baseUrl+"productCategory", ProductCategory[].class);

ManufacturerProduct[] requestGetManufacturer = getRestTemplate().getForObject(baseUrl+"manufacturerProduct", ManufacturerProduct[].class);

model.addAttribute("objectsCategory",requestGetCategory);

model.addAttribute("objectsManufacturer",requestGetManufacturer);

return "/Moderator/Product/Add";

}

ProductCategory requestGetProductCategoryId = getRestTemplate().getForObject(baseUrl+"productCategory/"+idProductCategory, ProductCategory.class);

ManufacturerProduct requestGetManufacturerProductId = getRestTemplate().getForObject(baseUrl+"manufacturerProduct/"+idManufacturerProduct, ManufacturerProduct.class);

product.setProductCategory(requestGetProductCategoryId);

product.setManufacturerProduct(requestGetManufacturerProductId);

Product requestPost = getRestTemplate().postForObject(baseUrl+"product",product, Product.class);

return "redirect:/Moderator/Product/Index";

}

@GetMapping("/Product/Index/{id}")

public String ProductDelete(@PathVariable(value = "id") Long id, Model model) {

getRestTemplate().delete(baseUrl+"product/"+id);

return "redirect:/Moderator/Product/Index";

}

@GetMapping("/Product/Update/{id}")

public String ProductUpdate(@PathVariable Long id, Model model) {

Product requestGet = getRestTemplate().getForObject(baseUrl+"product/"+id, Product.class);

model.addAttribute("object",requestGet);

ProductCategory[] requestGetCategory = getRestTemplate().getForObject(baseUrl+"productCategory", ProductCategory[].class);

ManufacturerProduct[] requestGetManufacturer = getRestTemplate().getForObject(baseUrl+"manufacturerProduct", ManufacturerProduct[].class);

model.addAttribute("objectsCategory",requestGetCategory);

model.addAttribute("objectsManufacturer",requestGetManufacturer);

return "/Moderator/Product/Update";

}

@PostMapping("/Product/Update/{id}")

public String ProductUpdate(@PathVariable Long id,

@RequestParam(name="idManufacturerProduct") Long idManufacturerProduct,

@RequestParam(name="idProductCategory") Long idProductCategory,

@Valid Product product, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

ProductCategory[] requestGetCategory = getRestTemplate().getForObject(baseUrl+"productCategory", ProductCategory[].class);

ManufacturerProduct[] requestGetManufacturer = getRestTemplate().getForObject(baseUrl+"manufacturerProduct", ManufacturerProduct[].class);

model.addAttribute("objectsCategory",requestGetCategory);

model.addAttribute("objectsManufacturer",requestGetManufacturer);

return "redirect:/Moderator/Product/Update/"+id;

}

product.setIdProduct(id);

ProductCategory requestGetProductCategoryId = getRestTemplate().getForObject(baseUrl+"productCategory/"+idProductCategory, ProductCategory.class);

ManufacturerProduct requestGetManufacturerProductId = getRestTemplate().getForObject(baseUrl+"manufacturerProduct/"+idManufacturerProduct, ManufacturerProduct.class);

product.setProductCategory(requestGetProductCategoryId);

product.setManufacturerProduct(requestGetManufacturerProductId);

getRestTemplate().put(baseUrl+"product/"+id, product, Void.class);

return "redirect:/Moderator/Product/Index";

}

//ProductCategory

@GetMapping("/ProductCategory/Index")

public String ProductCategoryIndex(Model model) {

ProductCategory[] requestGet = getRestTemplate().getForObject(baseUrl+"productCategory", ProductCategory[].class);

model.addAttribute("objects",requestGet);

return "/Moderator/ProductCategory/Index";

}

@GetMapping("/ProductCategory/Add")

public String ProductCategoryAdd(@ModelAttribute("productCategory") ProductCategory productCategory, Model model) {

return "/Moderator/ProductCategory/Add";

}

@PostMapping("/ProductCategory/Add")

public String ProductCategoryPost(@Valid @ModelAttribute("productCategory") ProductCategory productCategory, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

return "/Moderator/ProductCategory/Add";

}

ProductCategory requestPost = getRestTemplate().postForObject(baseUrl+"productCategory",productCategory, ProductCategory.class);

return "redirect:/Moderator/ProductCategory/Index";

}

@GetMapping("/ProductCategory/Index/{id}")

public String ProductCategoryDelete(@PathVariable(value = "id") Long id, Model model) {

getRestTemplate().delete(baseUrl+"productCategory/"+id);

return "redirect:/Moderator/ProductCategory/Index";

}

@GetMapping("/ProductCategory/Update/{id}")

public String ProductCategoryUpdate(@PathVariable Long id, Model model) {

ProductCategory requestGet = getRestTemplate().getForObject(baseUrl+"productCategory/"+id, ProductCategory.class);

model.addAttribute("object",requestGet);

return "/Moderator/ProductCategory/Update";

}

@PostMapping("/ProductCategory/Update/{id}")

public String ProductCategoryUpdate(@PathVariable Long id, @Valid ProductCategory productCategory,BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

return "redirect:/Moderator/ProductCategory/Update/"+id;

}

productCategory.setIdProductCategory(id);

getRestTemplate().put(baseUrl+"productCategory/"+id, productCategory, Void.class);

return "redirect:/Moderator/ProductCategory/Index";

}

//Shop

@GetMapping("/Shop/Index")

public String ShopIndex(Model model) {

Shop[] requestGet = getRestTemplate().getForObject(baseUrl+"shop", Shop[].class);

model.addAttribute("objects",requestGet);

return "/Moderator/Shop/Index";

}

@GetMapping("/Shop/Add")

public String ShopAdd(@ModelAttribute("shop") Shop shop, Model model) {

Product[] requestGetProduct = getRestTemplate().getForObject(baseUrl+"product", Product[].class);

Schedule[] requestGetSchedule = getRestTemplate().getForObject(baseUrl+"schedule", Schedule[].class);

model.addAttribute("objectsProduct",requestGetProduct);

model.addAttribute("objectsSchedule",requestGetSchedule);

return "/Moderator/Shop/Add";

}

@PostMapping("/Shop/Add")

public String ShopPost(@Valid @ModelAttribute("shop") Shop shop,

BindingResult bindingResult,

@RequestParam(name="idProduct") Long idProduct,

@RequestParam(name="idSchedule") Long idSchedule,

Model model) {

if (bindingResult.hasErrors())

{

Product[] requestGetProduct = getRestTemplate().getForObject(baseUrl+"product", Product[].class);

Schedule[] requestGetSchedule = getRestTemplate().getForObject(baseUrl+"schedule", Schedule[].class);

model.addAttribute("objectsProduct",requestGetProduct);

model.addAttribute("objectsSchedule",requestGetSchedule);

return "/Moderator/Shop/Add";

}

Product requestGetProductId = getRestTemplate().getForObject(baseUrl+"product/"+idProduct, Product.class);

Schedule requestGetScheduleId = getRestTemplate().getForObject(baseUrl+"schedule/"+idSchedule, Schedule.class);

shop.setProduct(requestGetProductId);

shop.setSchedule(requestGetScheduleId);

Shop requestPost = getRestTemplate().postForObject(baseUrl+"shop",shop, Shop.class);

return "redirect:/Moderator/Shop/Index";

}

@GetMapping("/Shop/Index/{id}")

public String ShopDelete(@PathVariable(value = "id") Long id, Model model) {

getRestTemplate().delete(baseUrl+"shop/"+id);

return "redirect:/Moderator/Shop/Index";

}

@GetMapping("/Shop/Update/{id}")

public String ShopUpdate(@PathVariable Long id, Model model) {

Shop requestGet = getRestTemplate().getForObject(baseUrl+"shop/"+id, Shop.class);

model.addAttribute("object",requestGet);

Product[] requestGetProduct = getRestTemplate().getForObject(baseUrl+"product", Product[].class);

Schedule[] requestGetSchedule = getRestTemplate().getForObject(baseUrl+"schedule", Schedule[].class);

model.addAttribute("objectsProduct",requestGetProduct);

model.addAttribute("objectsSchedule",requestGetSchedule);

return "/Moderator/Shop/Update";

}

@PostMapping("/Shop/Update/{id}")

public String ShopUpdate(@Valid Shop shop,

BindingResult bindingResult,

@PathVariable Long id,

@RequestParam(name="idProduct") Long idProduct,

@RequestParam(name="idSchedule") Long idSchedule,

Model model) {

if (bindingResult.hasErrors())

{

Product[] requestGetProduct = getRestTemplate().getForObject(baseUrl+"product", Product[].class);

Schedule[] requestGetSchedule = getRestTemplate().getForObject(baseUrl+"schedule", Schedule[].class);

model.addAttribute("objectsProduct",requestGetProduct);

model.addAttribute("objectsSchedule",requestGetSchedule);

return "redirect:/Moderator/Shop/Update/"+id;

}

shop.setIdShop(id);

Product requestGetProductId = getRestTemplate().getForObject(baseUrl+"product/"+idProduct, Product.class);

Schedule requestGetScheduleId = getRestTemplate().getForObject(baseUrl+"schedule/"+idSchedule, Schedule.class);

shop.setProduct(requestGetProductId);

shop.setSchedule(requestGetScheduleId);

getRestTemplate().put(baseUrl+"shop/"+id, shop, Void.class);

return "redirect:/Moderator/Shop/Index";

}

//Schedule

@GetMapping("/Schedule/Index")

public String TimetableIndex(Model model) {

Schedule[] requestGet = getRestTemplate().getForObject(baseUrl+"schedule", Schedule[].class);

model.addAttribute("objects",requestGet);

return "/Moderator/Schedule/Index";

}

@GetMapping("/Schedule/Add")

public String TimetableAdd(@ModelAttribute("schedule") Schedule schedule, Model model) {

return "/Moderator/Schedule/Add";

}

@PostMapping("/Schedule/Add")

public String TimetablePost(@Valid @ModelAttribute("schedule") Schedule schedule, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

return "/Moderator/Schedule/Add";

}

Schedule requestPost = getRestTemplate().postForObject(baseUrl+"schedule",schedule, Schedule.class);

return "redirect:/Moderator/Schedule/Index";

}

@GetMapping("/Schedule/Index/{id}")

public String TimetableDelete(@PathVariable(value = "id") Long id, Model model) {

getRestTemplate().delete(baseUrl+"schedule/"+id);

return "redirect:/Moderator/Schedule/Index";

}

@GetMapping("/Schedule/Update/{id}")

public String TimetableUpdate(@PathVariable Long id, Model model) {

Schedule requestGet = getRestTemplate().getForObject(baseUrl+"schedule/"+id, Schedule.class);

model.addAttribute("object",requestGet);

return "/Moderator/Schedule/Update";

}

@PostMapping("/Schedule/Update/{id}")

public String TimetableUpdate(@PathVariable Long id, @Valid Schedule schedule, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors())

{

return "redirect:/Moderator/Schedule/Update/"+id;

}

schedule.setIdTimetable(id);

getRestTemplate().put(baseUrl+"schedule/"+id, schedule, Void.class);

return "redirect:/Moderator/Schedule/Index";

}

}

UserController.java:

package com.example.dns.Controllers;

import com.example.dns.Models.\*;

import org.springframework.security.access.prepost.PreAuthorize;

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.\*;

import org.springframework.web.client.RestTemplate;

@RequestMapping("/User")

@Controller

public class UserController {

public String baseUrl = "http://localhost:8080/";

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

@GetMapping("/Index")

public String UserIndex(Model model) {

Product[] requestGetProduct = getRestTemplate().getForObject(baseUrl+"product", Product[].class);

Shop[] requestGetShop = getRestTemplate().getForObject(baseUrl+"shop", Shop[].class);

model.addAttribute("objectsProduct",requestGetProduct);

model.addAttribute("objectsShop",requestGetShop);

return "/User/Index";

}

@PostMapping("/Index")

public String UserIndex(@RequestParam(name = "address",required = false) String address,

@RequestParam(name = "name", required = false) String name,

Model model) {

Product[] requestGetProduct = getRestTemplate().getForObject(baseUrl+"product/"+address+"/"+name, Product[].class);

Shop[] requestGetShop = getRestTemplate().getForObject(baseUrl+"shop", Shop[].class);

model.addAttribute("objectsShop",requestGetShop);

model.addAttribute("objectsProduct",requestGetProduct);

return "/User/Index";

}

@GetMapping("/Product/{id}")

public String ProductGet(@PathVariable(value = "id") Long id, Model model) {

Product requestGetProduct = getRestTemplate().getForObject(baseUrl+"product/"+id, Product.class);

Person[] requestGetPerson = getRestTemplate().getForObject(baseUrl+"person", Person[].class);

model.addAttribute("objectProduct",requestGetProduct);

model.addAttribute("objectPerson", requestGetPerson);

return "/User/Product";

}

public String getCurrentUsername() {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

return auth.getName();

}

@PostMapping("/Product/{id}")

public String ProductAddPerson(@PathVariable(value = "id") Long id,

Person person, PersonShop personShop)

{

person.setUsername(getCurrentUsername());

Person requestGetPerson = getRestTemplate().postForObject(baseUrl+"searchPerson",person ,Person.class);

Product requestGetProduct = getRestTemplate().getForObject(baseUrl+"product/"+id, Product.class);

personShop.setProduct(requestGetProduct);

personShop.setPerson(requestGetPerson);

PersonShop postPersonShop = getRestTemplate().postForObject(baseUrl+"personShop",personShop,PersonShop.class);

return "/User/Index";

}

@GetMapping("/Order")

public String UserOrder(Person person, Model model) {

person.setUsername(getCurrentUsername());

PersonShop[] requestGetShop = getRestTemplate().postForObject(baseUrl+"searchOrder",person ,PersonShop[].class);

model.addAttribute("objectsPersonShop",requestGetShop);

return "/User/Order";

}

@GetMapping("/Order/{id}")

public String ProductDelete(@PathVariable(value = "id") Long id, Model model) {

getRestTemplate().delete(baseUrl+"personShop/"+id);

return "redirect:/User/Order";

}

}

ManufacturerProduct.java:

package com.example.dns.Models;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import java.util.List;

public class ManufacturerProduct {

private Long idManufacturerProduct;

@Size(min = 3, message = "The nameManufacturerProduct cannot be less than 3 characters")

@NotBlank(message = "NameManufacturerProduct is required")

private String nameManufacturerProduct;

private List<Product> products;

public ManufacturerProduct() {

}

public ManufacturerProduct(Long idManufacturerProduct, String nameManufacturerProduct) {

this.idManufacturerProduct = idManufacturerProduct;

this.nameManufacturerProduct = nameManufacturerProduct;

}

public Long getIdManufacturerProduct() {

return idManufacturerProduct;

}

public void setIdManufacturerProduct(Long idManufacturerProduct) {

this.idManufacturerProduct = idManufacturerProduct;

}

public String getNameManufacturerProduct() {

return nameManufacturerProduct;

}

public void setNameManufacturerProduct(String nameManufacturerProduct) {

this.nameManufacturerProduct = nameManufacturerProduct;

}

public List<Product> getProducts() {

return products;

}

public void setProducts(List<Product> products) {

this.products = products;

}

}

Person.java:

package com.example.dns.Models;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Pattern;

import javax.validation.constraints.Size;

import java.util.List;

import java.util.Set;

public class Person {

private Long idPerson;

@Size(min = 3, max = 25, message = "The login cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Login is required")

private String username;

@Size(min = 8, message = "The password cannot be less than 8 characters")

@Pattern(regexp = "^.\*(?=.\*[a-zA-Z])(?=.\*\\d)(?=.\*[!#$%&? \"]).\*$" , message = "The password must contain uppercase and lowercase letters, numbers and special characters")

@NotBlank(message = "Password is required")

private String password;

private boolean active;

private Set<Role> roles;

public Person() {

}

public Person(Long idPerson, String username, String password, boolean active, Set<Role> roles) {

this.idPerson = idPerson;

this.username = username;

this.password = password;

this.active = active;

this.roles = roles;

}

public Long getIdPerson() {

return idPerson;

}

public void setIdPerson(Long idPerson) {

this.idPerson = idPerson;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

public Set<Role> getRoles() {

return roles;

}

public void setRoles(Set<Role> roles) {

this.roles = roles;

}

}

PersonShop.java:

package com.example.dns.Models;

public class PersonShop {

private Long idPersonShop;

private Person person;

private Product product;

public PersonShop() {

}

public PersonShop(Long idPersonShop, Person person, Product product) {

this.idPersonShop = idPersonShop;

this.person = person;

this.product = product;

}

public Long getIdPersonShop() {

return idPersonShop;

}

public void setIdPersonShop(Long idPersonShop) {

this.idPersonShop = idPersonShop;

}

public Person getPerson() {

return person;

}

public void setPerson(Person person) {

this.person = person;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

}

Product.java:

package com.example.dns.Models;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import java.util.List;

public class Product {

private Long idProduct;

@Size(min = 3, message = "The name cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Name is required")

private String name;

@Size(min = 10, message = "The specifications cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Specifications is required")

private String specifications;

@Size(min = 10, message = "The description cannot be less than 8 characters and more than 20 characters")

@NotBlank(message = "Description is required")

private String description;

@NotNull

private Double price;

private ManufacturerProduct manufacturerProduct;

private ProductCategory productCategory;

private List<Shop> shops;

public Product() {

}

public Product(Long idProduct, String name, String specifications, String description, Double price, ManufacturerProduct manufacturerProduct, ProductCategory productCategory) {

this.idProduct = idProduct;

this.name = name;

this.specifications = specifications;

this.description = description;

this.price = price;

this.manufacturerProduct = manufacturerProduct;

this.productCategory = productCategory;

}

public Long getIdProduct() {

return idProduct;

}

public void setIdProduct(Long idProduct) {

this.idProduct = idProduct;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSpecifications() {

return specifications;

}

public void setSpecifications(String specifications) {

this.specifications = specifications;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public ManufacturerProduct getManufacturerProduct() {

return manufacturerProduct;

}

public void setManufacturerProduct(ManufacturerProduct manufacturerProduct) {

this.manufacturerProduct = manufacturerProduct;

}

public ProductCategory getProductCategory() {

return productCategory;

}

public void setProductCategory(ProductCategory productCategory) {

this.productCategory = productCategory;

}

public List<Shop> getShops() {

return shops;

}

public void setShops(List<Shop> shops) {

this.shops = shops;

}

}

ProductCategory.java:

package com.example.dns.Models;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import java.util.List;

public class ProductCategory {

private Long idProductCategory;

@Size(min = 3, message = "The nameProductCategory cannot be less than 3 characters")

@NotBlank(message = "NameProductCategory is required")

private String nameProductCategory;

private List<Product> products;

public ProductCategory() {

}

public ProductCategory(Long idProductCategory, String nameProductCategory) {

this.idProductCategory = idProductCategory;

this.nameProductCategory = nameProductCategory;

}

public Long getIdProductCategory() {

return idProductCategory;

}

public void setIdProductCategory(Long idProductCategory) {

this.idProductCategory = idProductCategory;

}

public String getNameProductCategory() {

return nameProductCategory;

}

public void setNameProductCategory(String nameProductCategory) {

this.nameProductCategory = nameProductCategory;

}

public List<Product> getProducts() {

return products;

}

public void setProducts(List<Product> products) {

this.products = products;

}

}

Role.java:

package com.example.dns.Models;

public enum Role {

USER,

ADMIN,

EMPLOYEE;

}

Schedule.java:

package com.example.dns.Models;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

public class Schedule {

private Long idTimetable;

@NotNull

@NotBlank(message = "Schedule is required")

private String schedule;

public Schedule() {

}

public Schedule(Long idTimetable, String schedule) {

this.idTimetable = idTimetable;

this.schedule = schedule;

}

public Long getIdTimetable() {

return idTimetable;

}

public void setIdTimetable(Long idTimetable) {

this.idTimetable = idTimetable;

}

public String getSchedule() {

return schedule;

}

public void setSchedule(String schedule) {

this.schedule = schedule;

}

}

Shop.java:

package com.example.dns.Models;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import java.util.List;

public class Shop {

private Long idShop;

@Size(min = 5, message = "The address cannot be less than 5 characters")

@NotBlank(message = "Address is required")

private String address;

private Product product;

private Schedule schedule;

public Shop() {

}

public Shop(Long idShop, String address, Product product, Schedule schedule) {

this.idShop = idShop;

this.address = address;

this.product = product;

this.schedule = schedule;

}

public Long getIdShop() {

return idShop;

}

public void setIdShop(Long idShop) {

this.idShop = idShop;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public Schedule getSchedule() {

return schedule;

}

public void setSchedule(Schedule schedule) {

this.schedule = schedule;

}

}

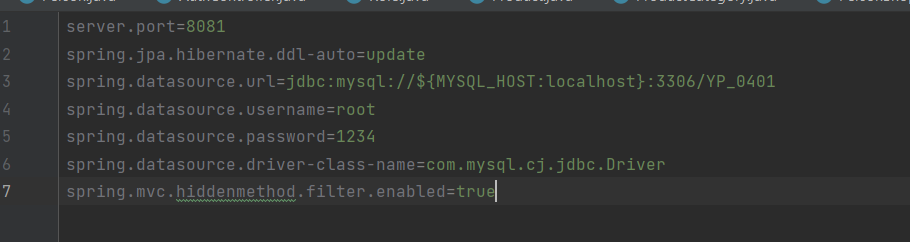


Рисунок Подключение к БД

Работа программы

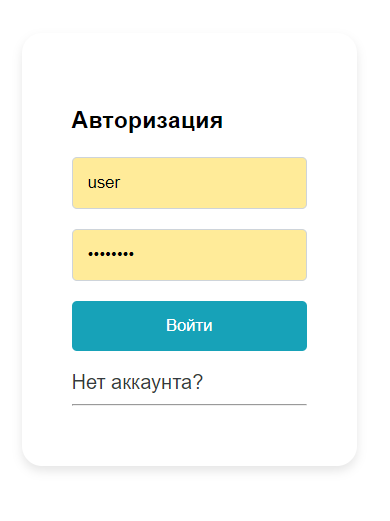
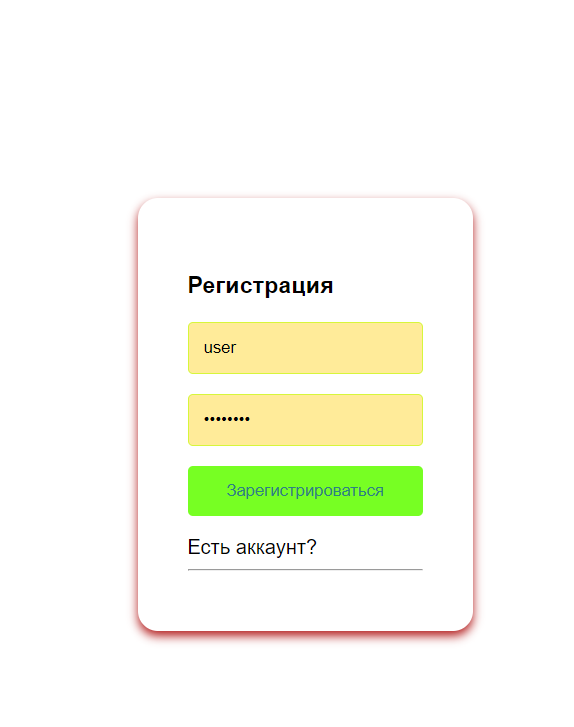


Рисунок Авторизация



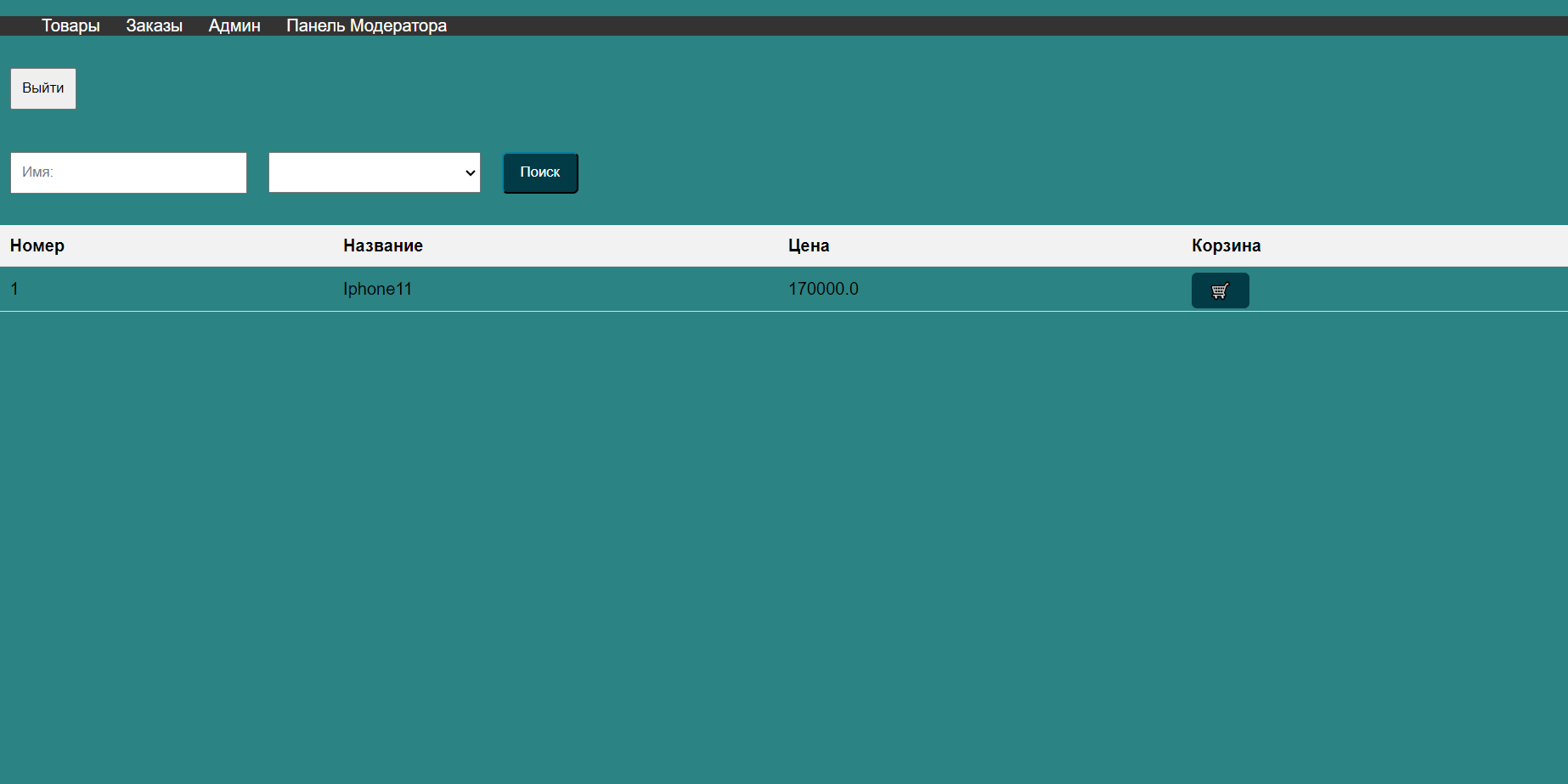


Рисунок 5 Главная страница товаров

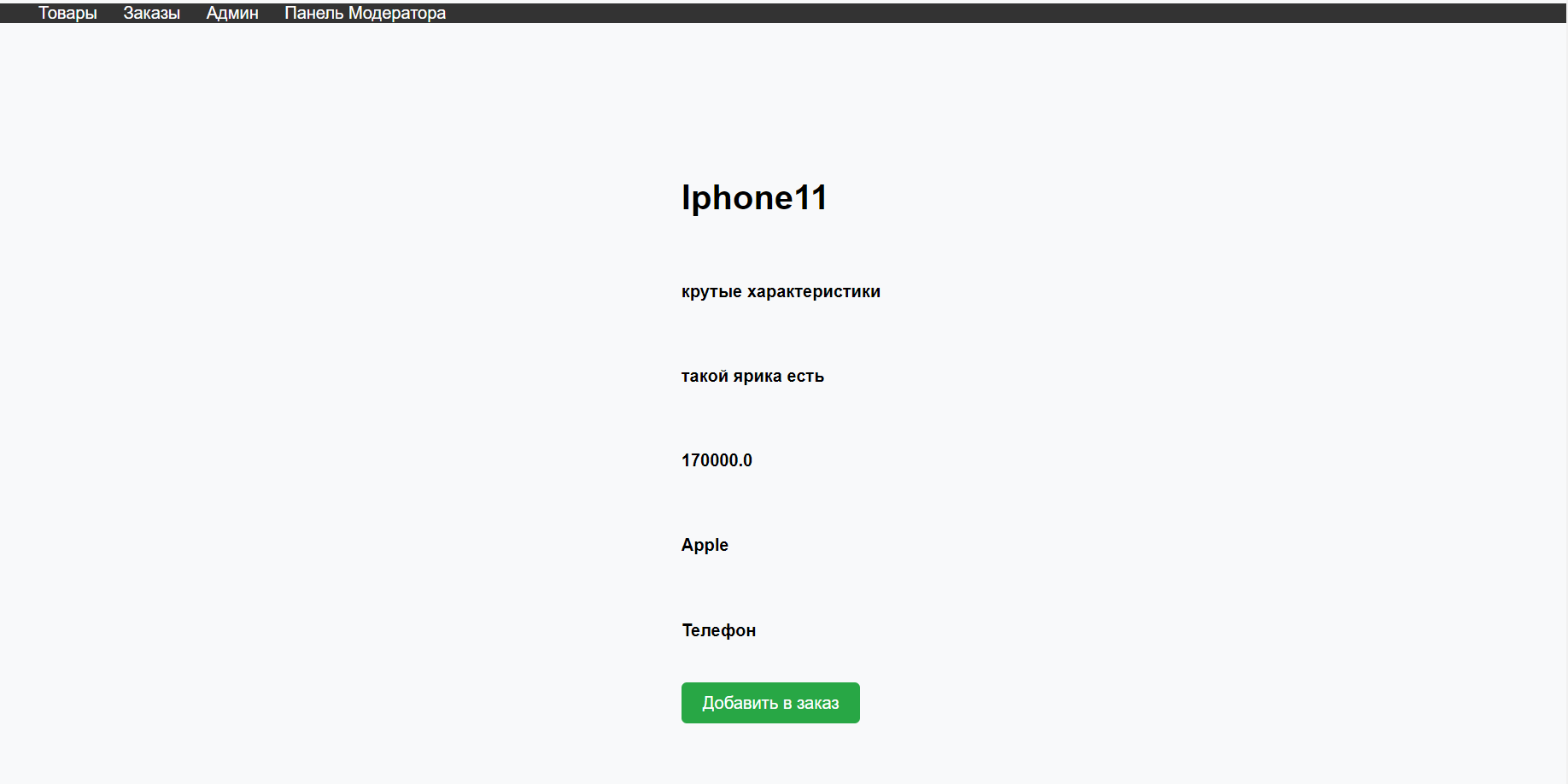


Рисунок 6 Вход в корзину

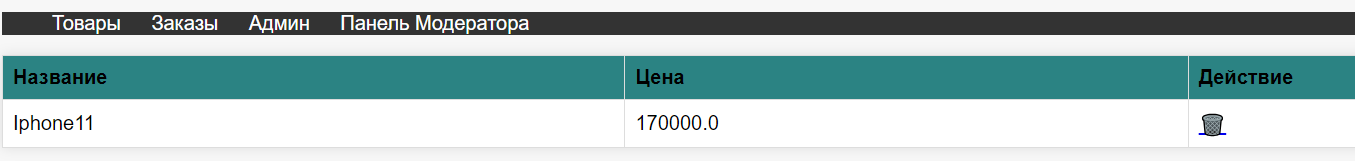


Рисунок 7 Заказы



Рисунок 8 Страница Админа

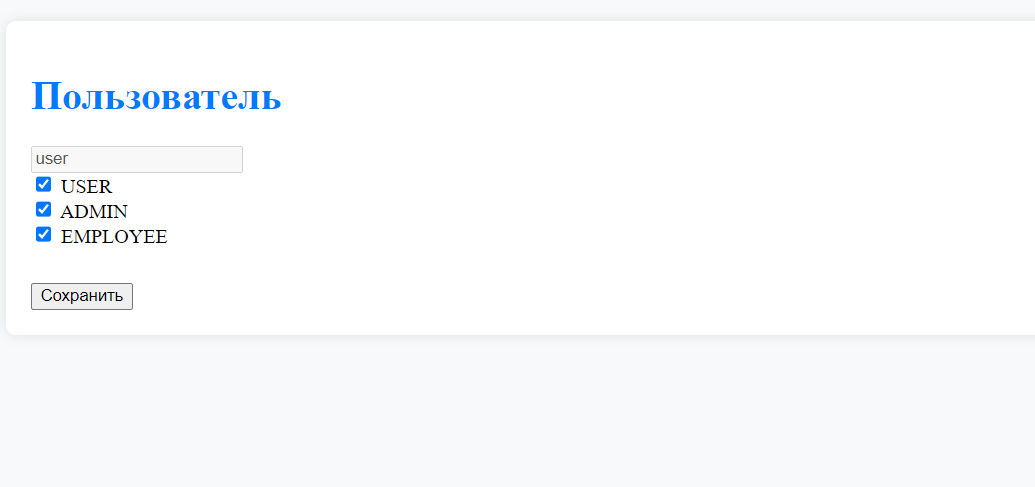


Рисунок 9 Управление настройками над пользователем

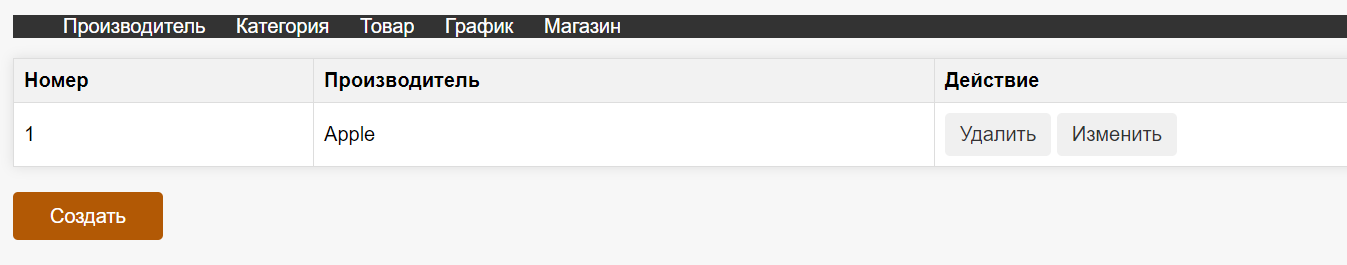


Рисунок Страница Модератора таблица Производитель

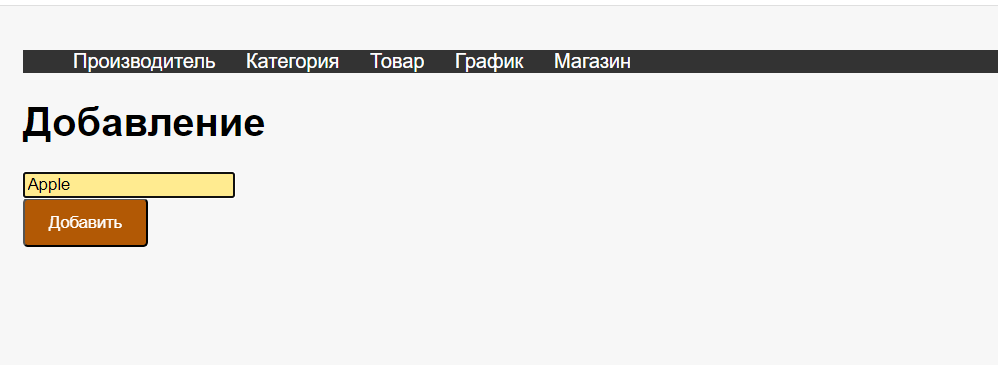


Рисунок 11 Добавление производителя

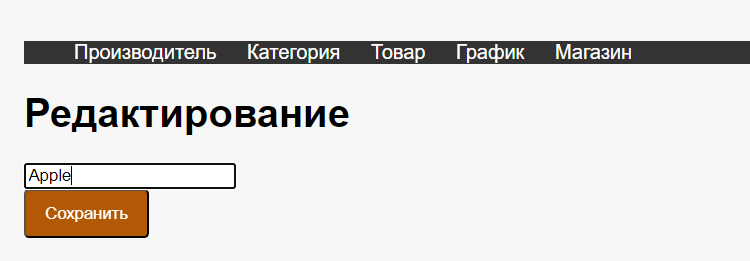


Рисунок 12 Изменение производителя

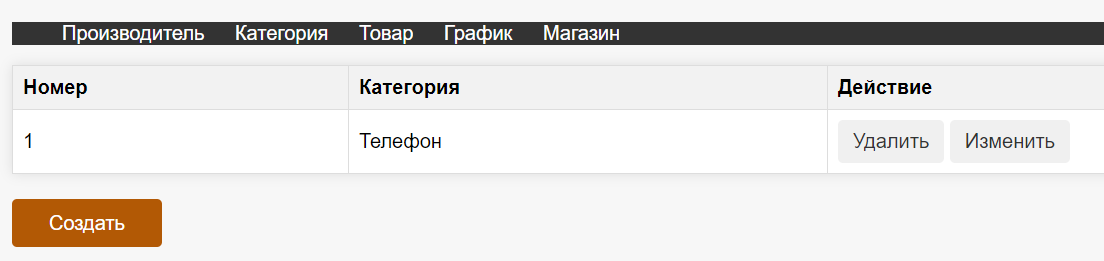


Рисунок 13 Категория

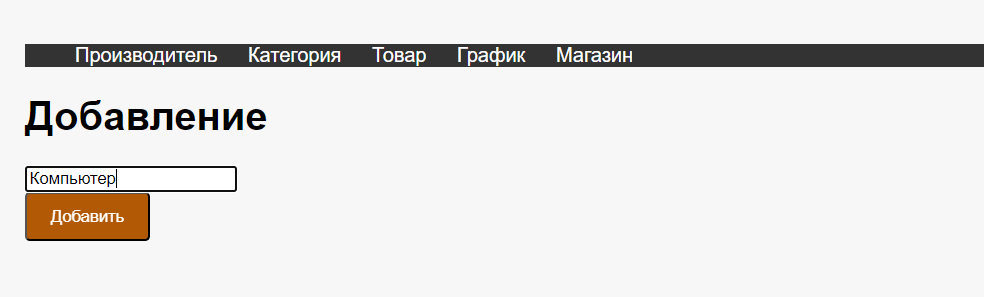


Рисунок 14 Добавление категории

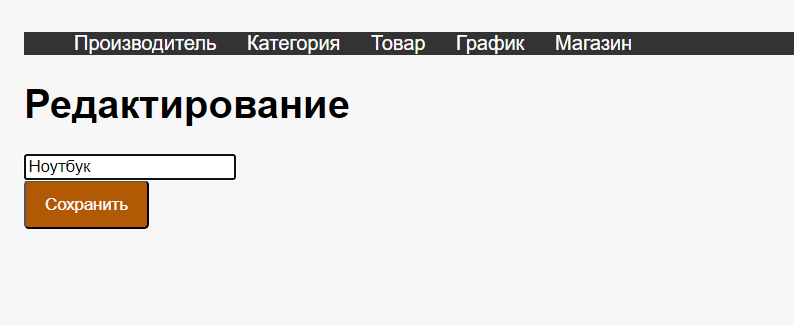


Рисунок 15 Редактирование категории

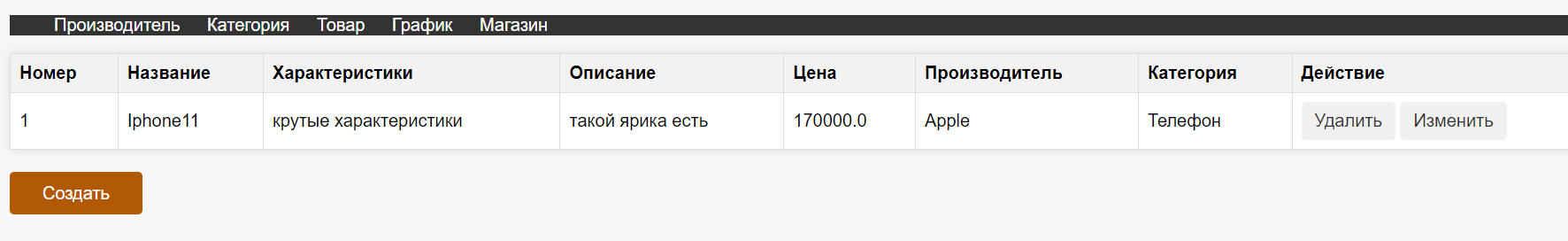


Рисунок 16 Товар

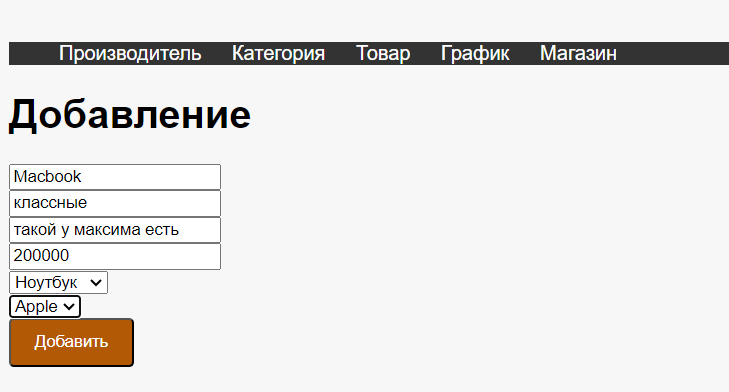


Рисунок 17 Добавление товара

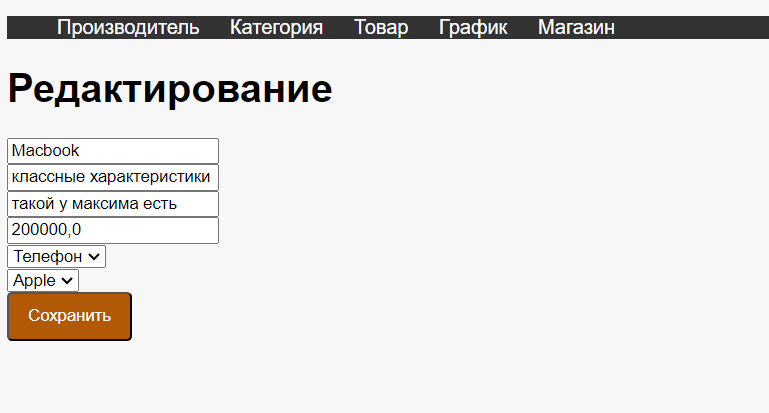


Рисунок 18 Изменение товара

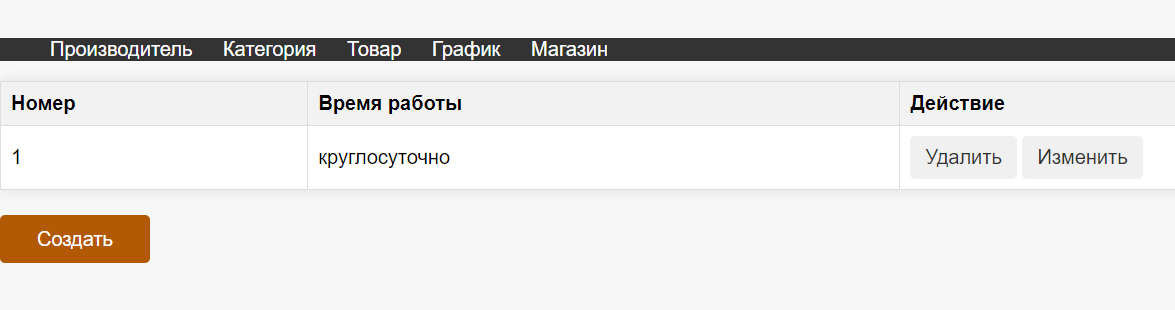


Рисунок 19 График



Рисунок 20 Добавление графика

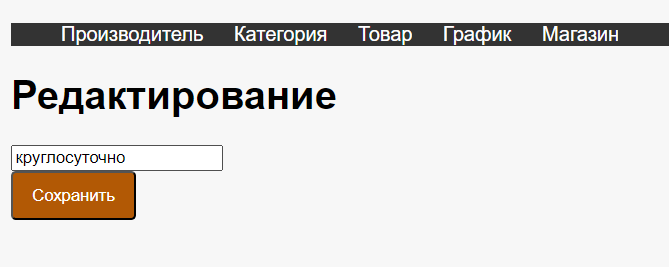


Рисунок 21 Изменение графика

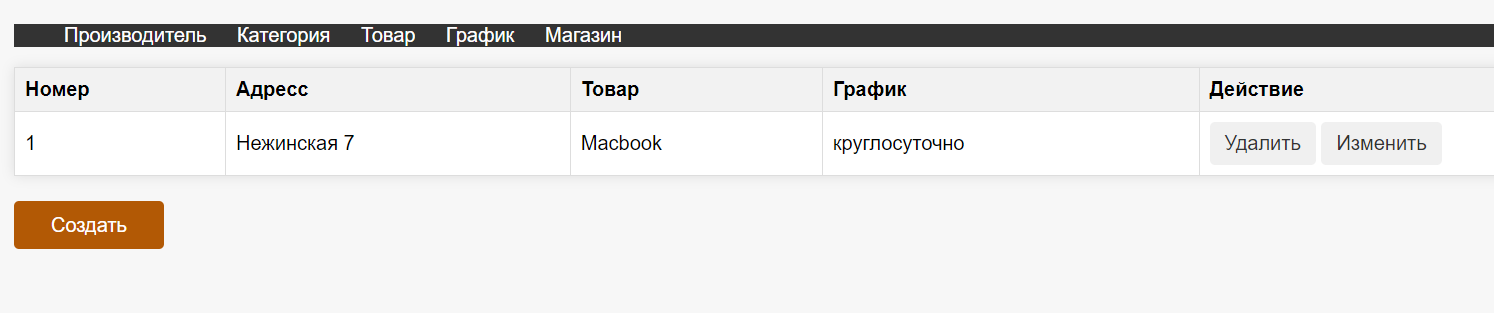


Рисунок 22 Магазин

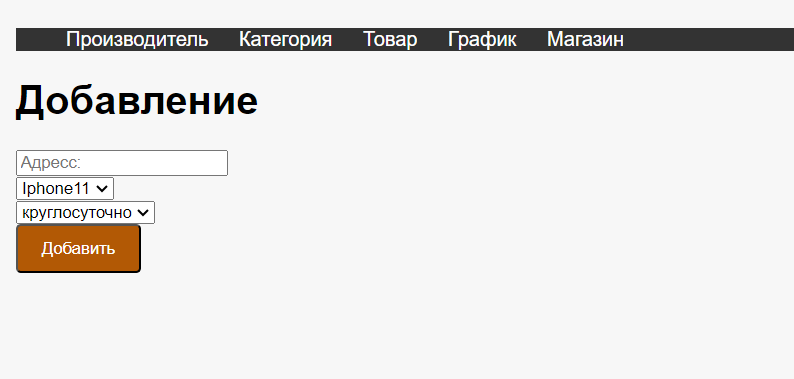


Рисунок 23 Добавление

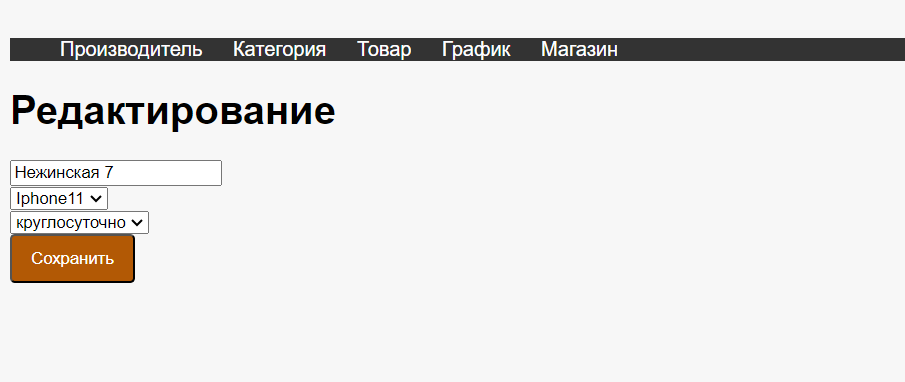
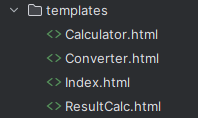
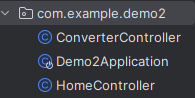


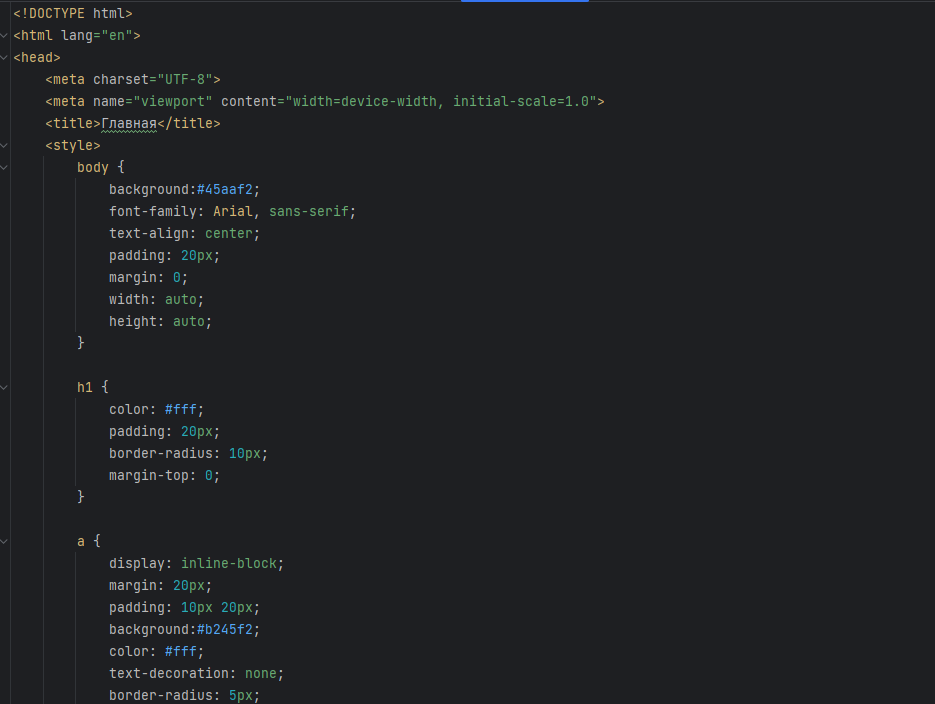
Рисунок Изменение

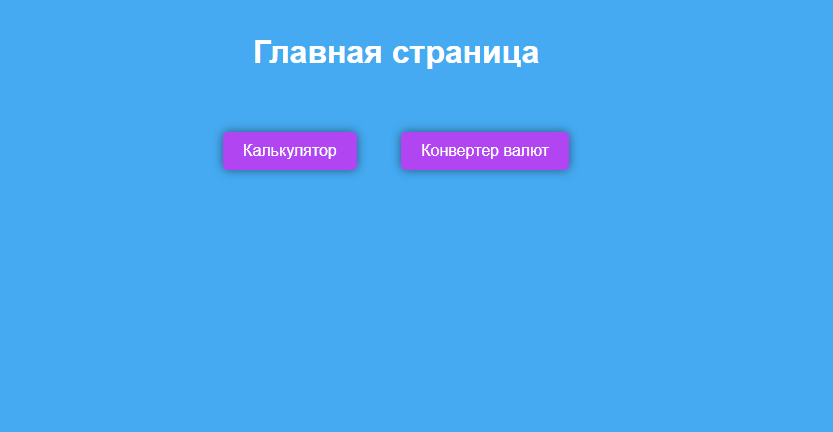
Вывод: в итоге было написано приложение интернет-магазина электронной техники.

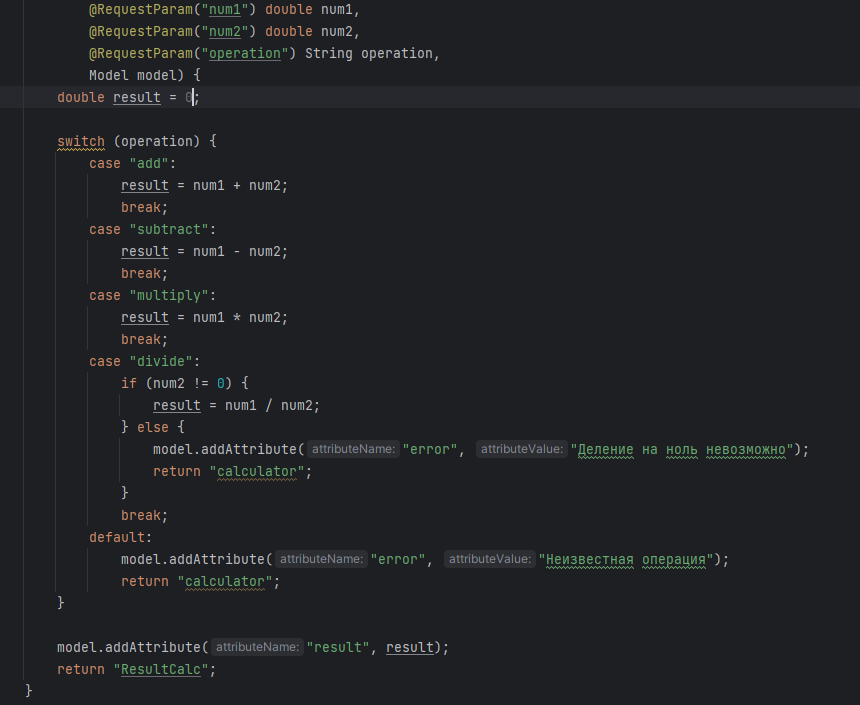
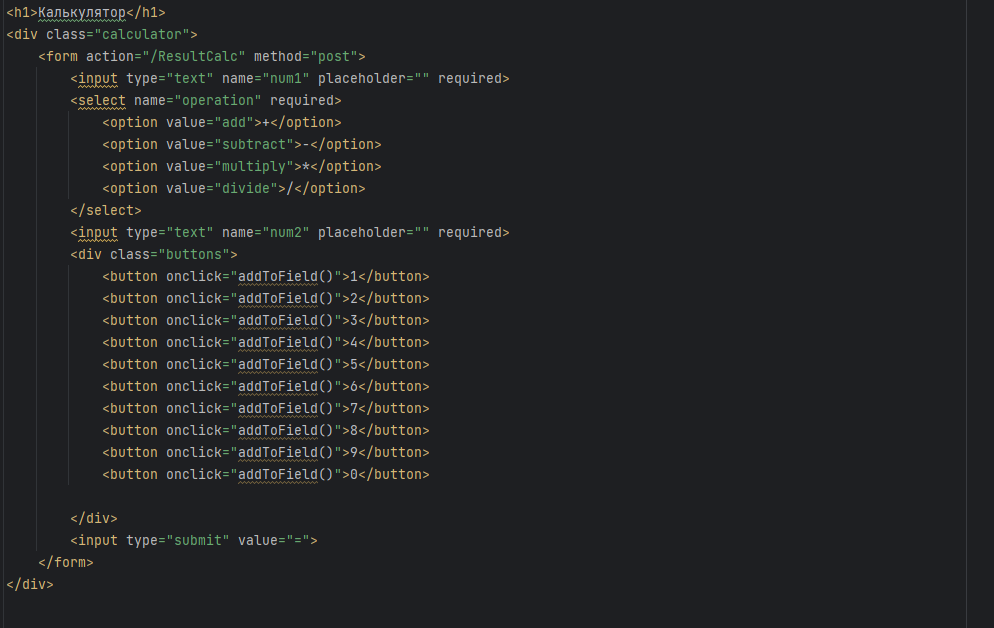
Практическая работа №1   
Цель работы: создать калькулятор, конвертер валют. Используя @GetMapping и @PostMapping также использовать @RequestParam   
  
Ход работы:  
  
1. Создаем файлы разметки html для всех необходимых по условию страниц.  
  


2. Создаем контроллеры для конвертера и для калькулятора   
  


3. Разметка для главной страницы

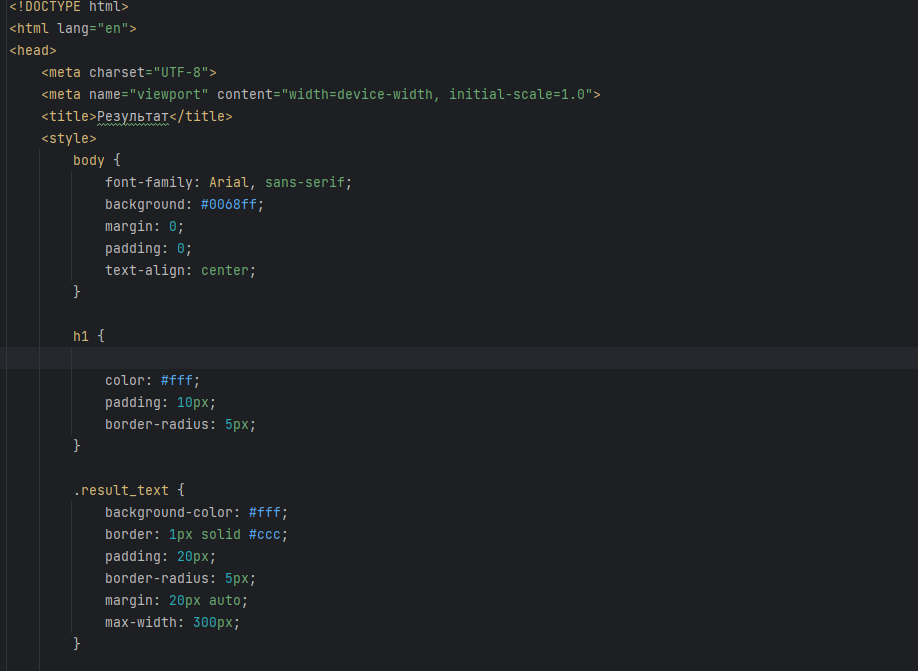


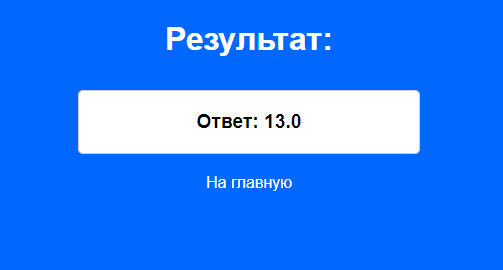
Итог  
  


4. Прописываем все в контроллере для калькулятора  
Используем switch для того, чтобы была логика в калькуляторе.  
  
  
  
5. Разметка для страницы калькулятора  
  


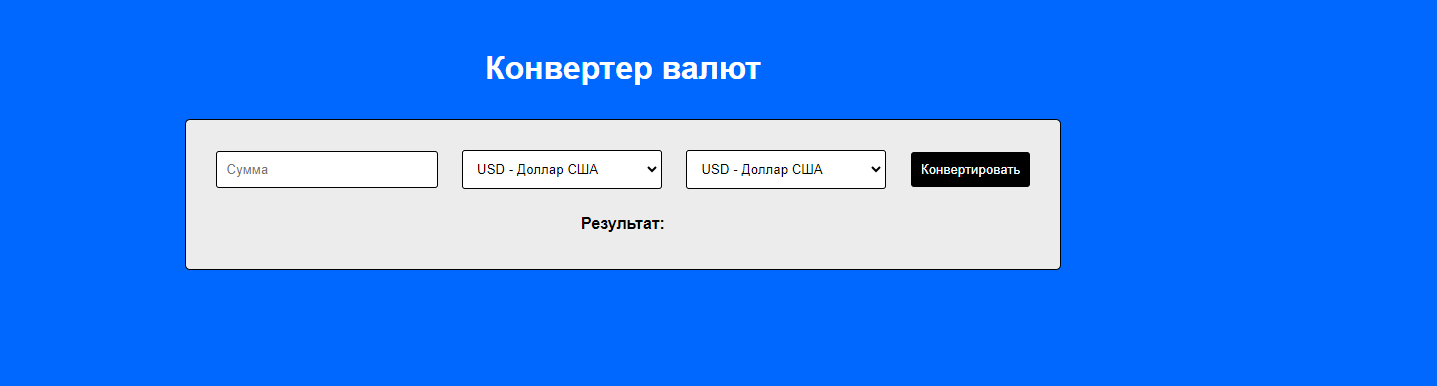
Итог  


Страница результата



Итог  


Калькулятор выполняет необходимые по условию функции также имеется перенаправление на главную.  
  
6. Страница конвертера  
  


Итог  


Пропишем теперь логику в контроллере  
  


Прописав и рассчитав все курсы валют, было подогнано необходимое значения для достоверной информации.

Вывод: Сделали калькулятор и конвертер соответствующий заданию.

**Практическая работа №2**

Цель работы:

В данной практической необходимо реализовать приложение, в котором будет реализован паттерн DAO.

Требования:

1) На 3 необходимо создать 5 моделей по 4 поля в каждой, и реализовать базовый паттерн DAO.

2) На 4 необходимо сделать навигацию по своему сайту. Так же необходимо подтянуть стили на страницы.

3) На 5 необходимо добавить универсальный класс для обработки CRUD-операций, который будет вызываться для каждой модели.

1. У нас имеется 5 моделей и к каждым из них необходимо прописать геттеры и сеттеры.

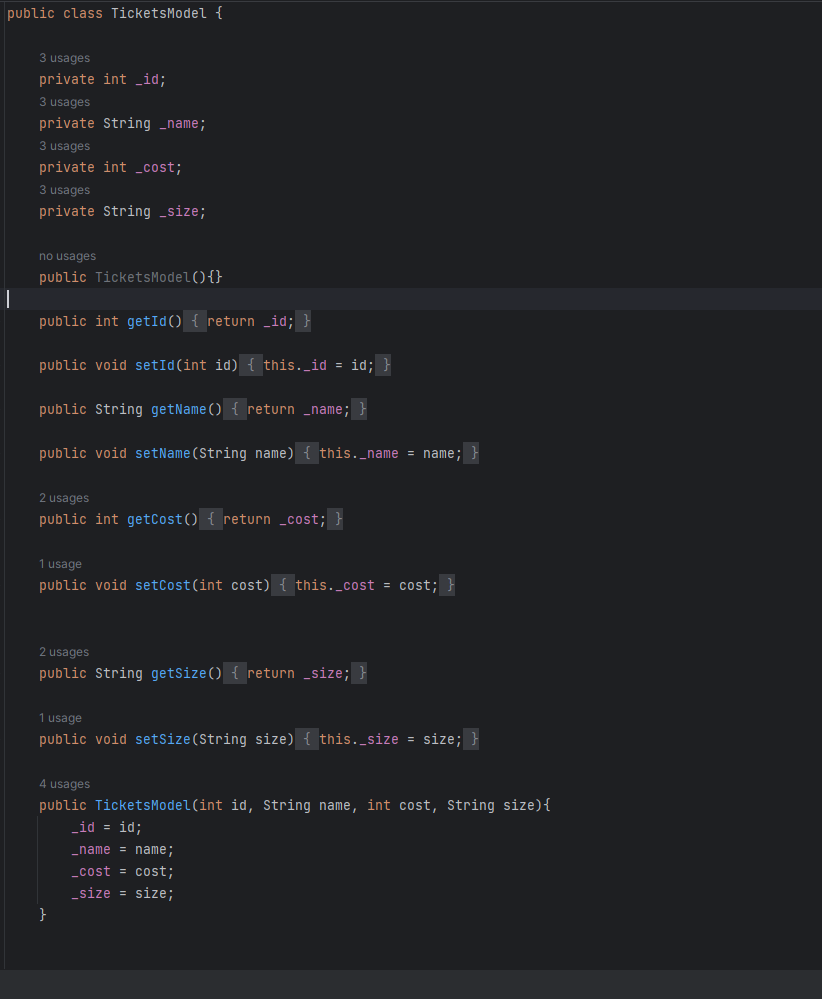


Рисунок 1. Геттеры и сеттеры.

2. Также создаем к каждой модели класс DAO.

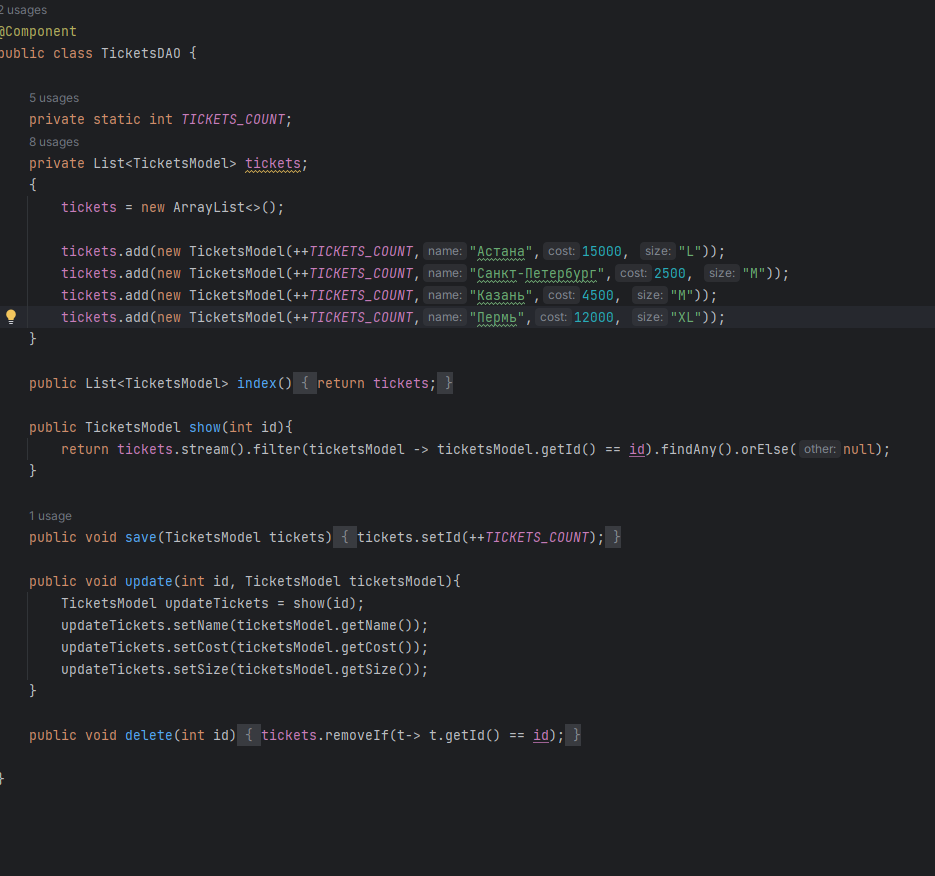


Рисунок 2. Класс DAO

3. Также создаем класс со всеми стилями.

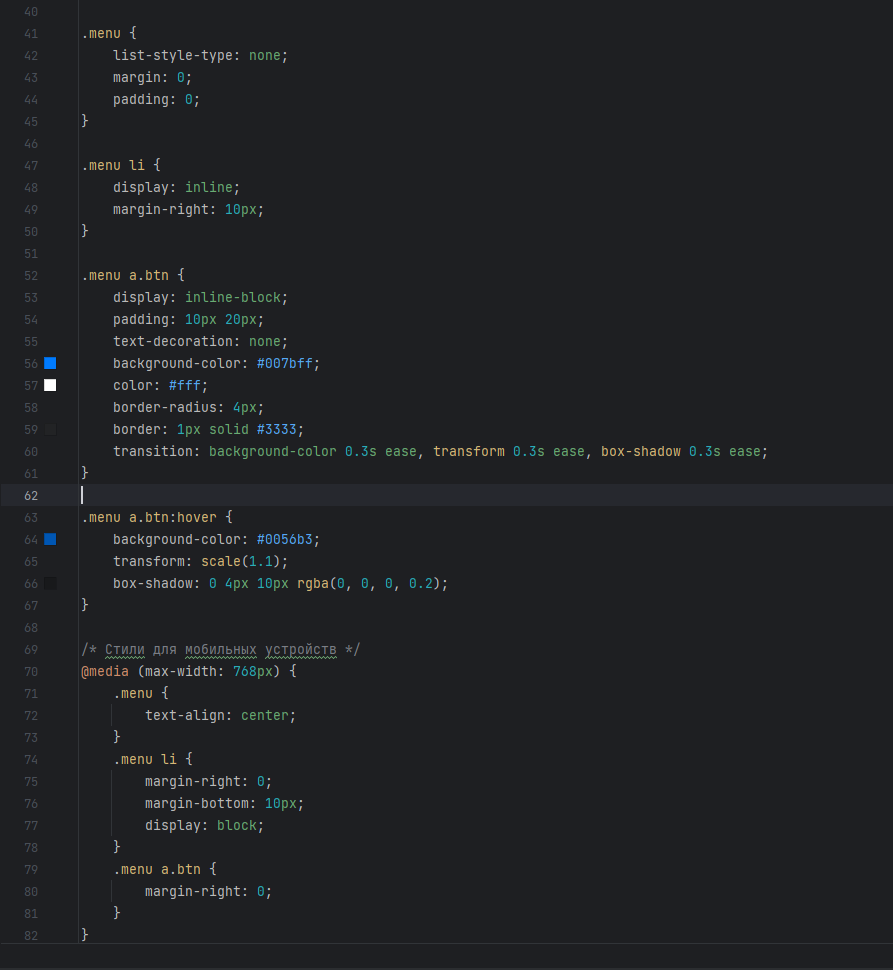


Рисунок 3. Стили.

4. Также у нас имеется общий контроллер для того, чтобы прописать все необходимое для классов и для главной странички.

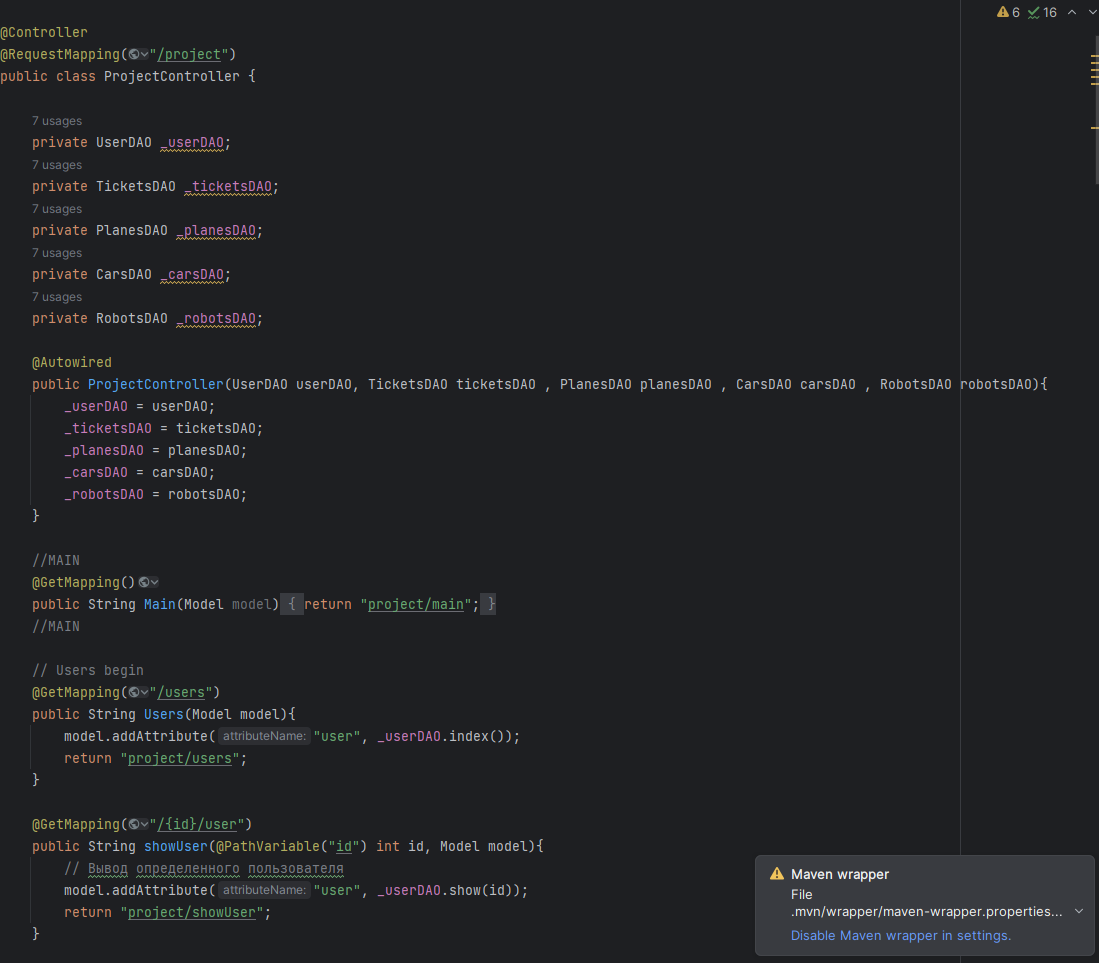


Рисунок 4. Главный контроллер.



Рисунок 5. Запросы.

5. Также имеется верстка для каждой страницы.



Рисунок 6. Основной элемент.

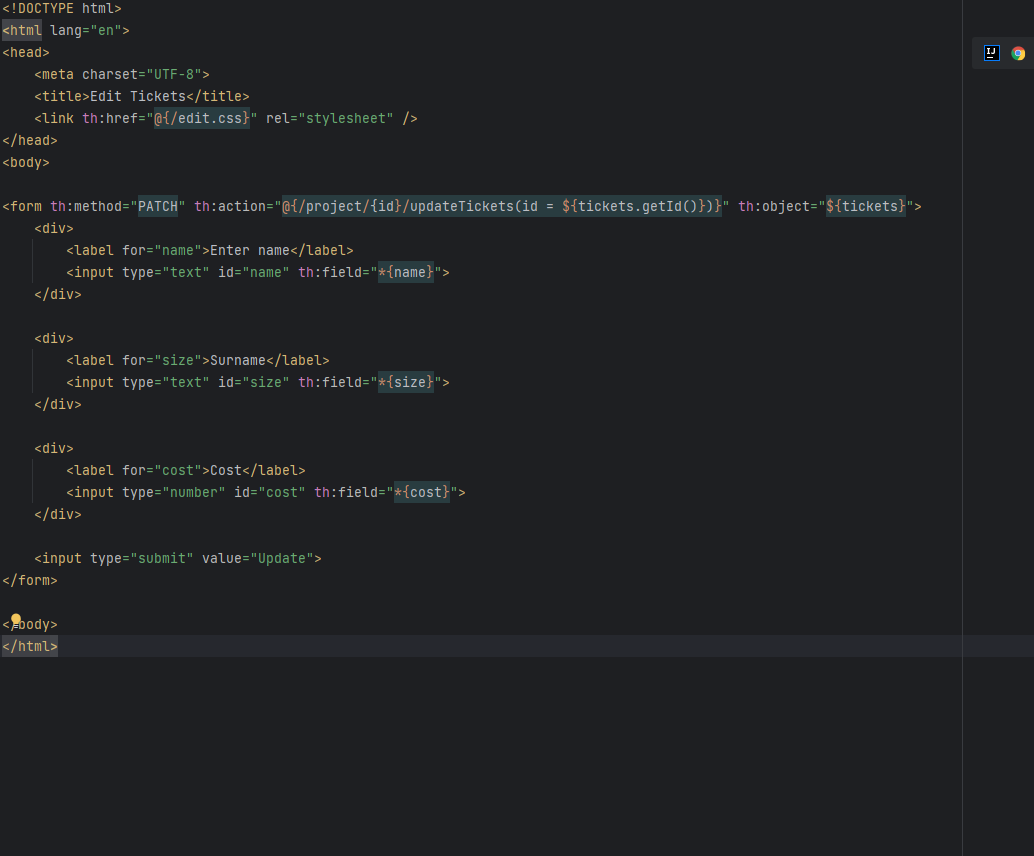


Рисунок 7. Редактирование.

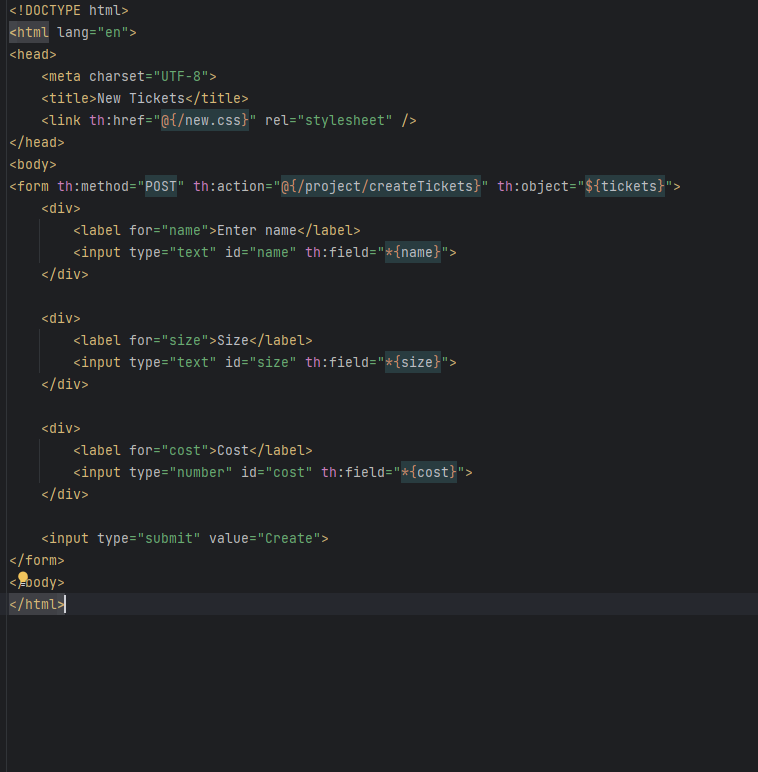


Рисунок 8. Добавление.

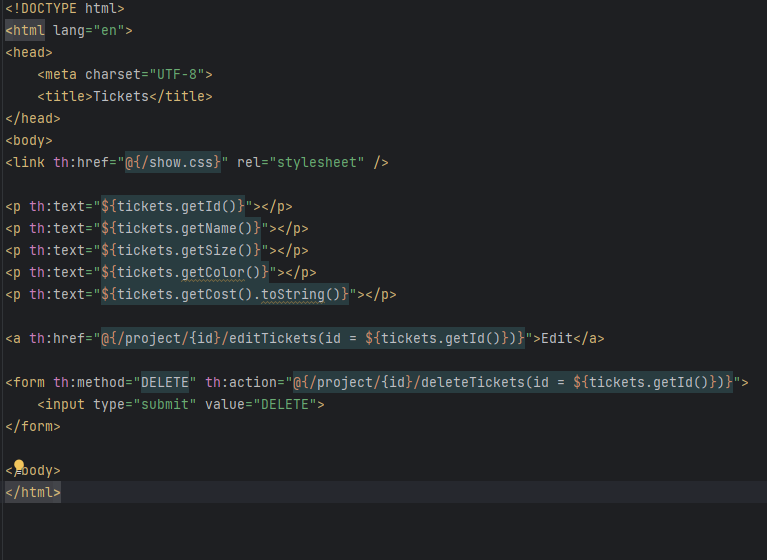


Рисунок 9. Отображение.

6. Главная страница и общий список всех файлов.

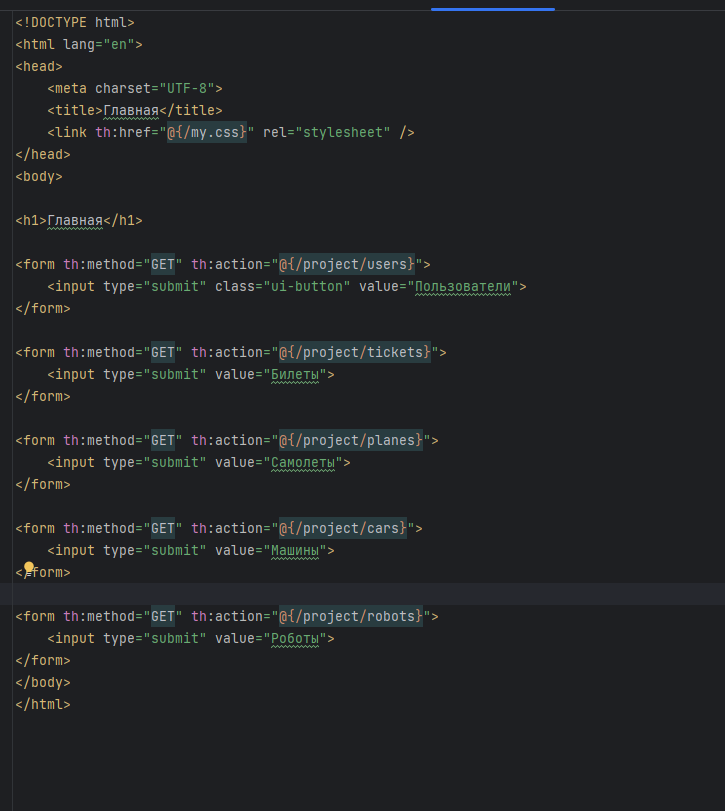


Рисунок 10. Главная.

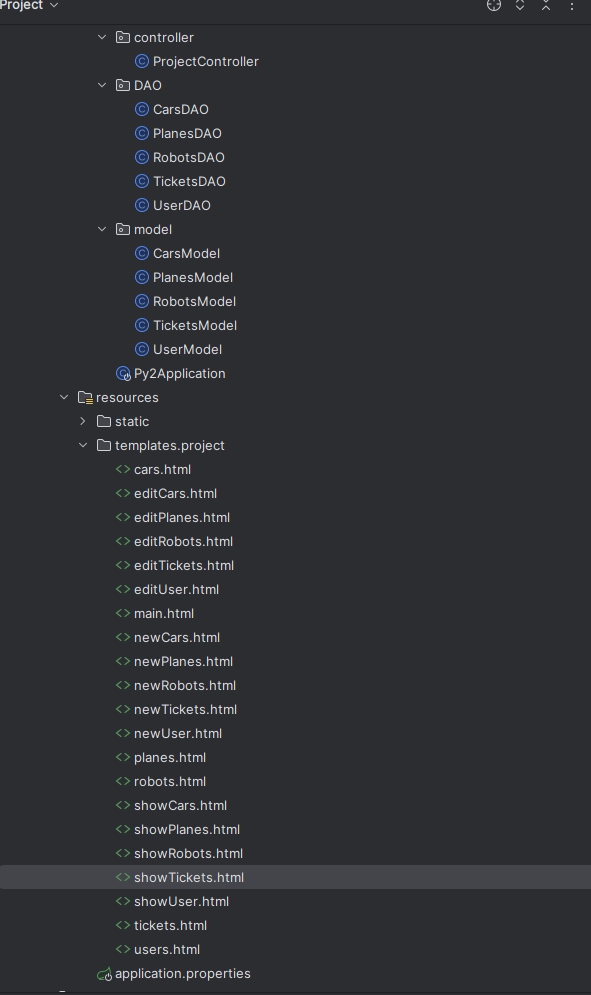
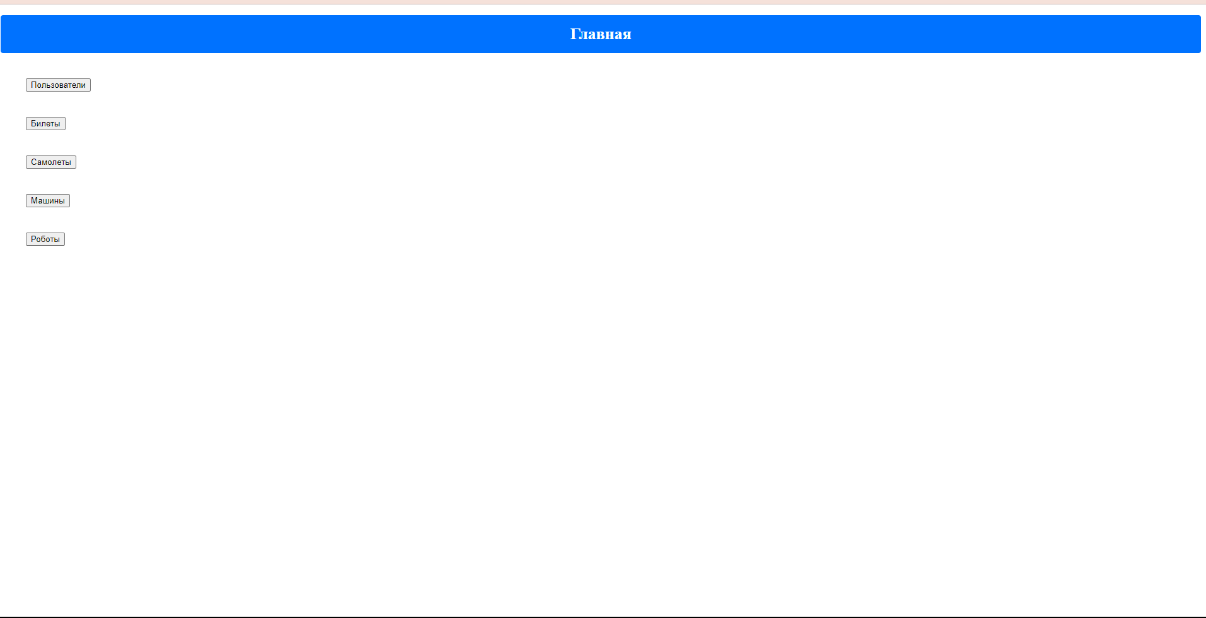
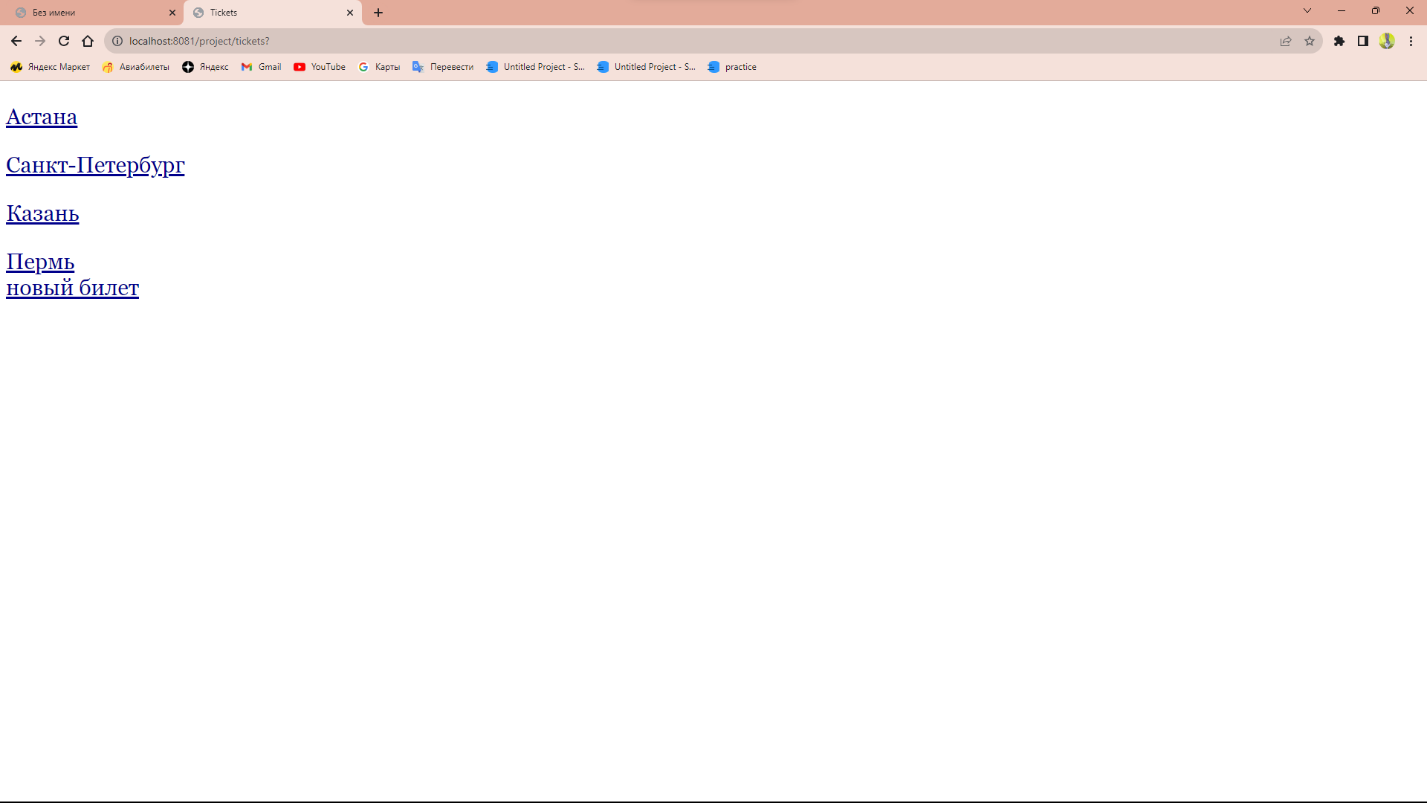
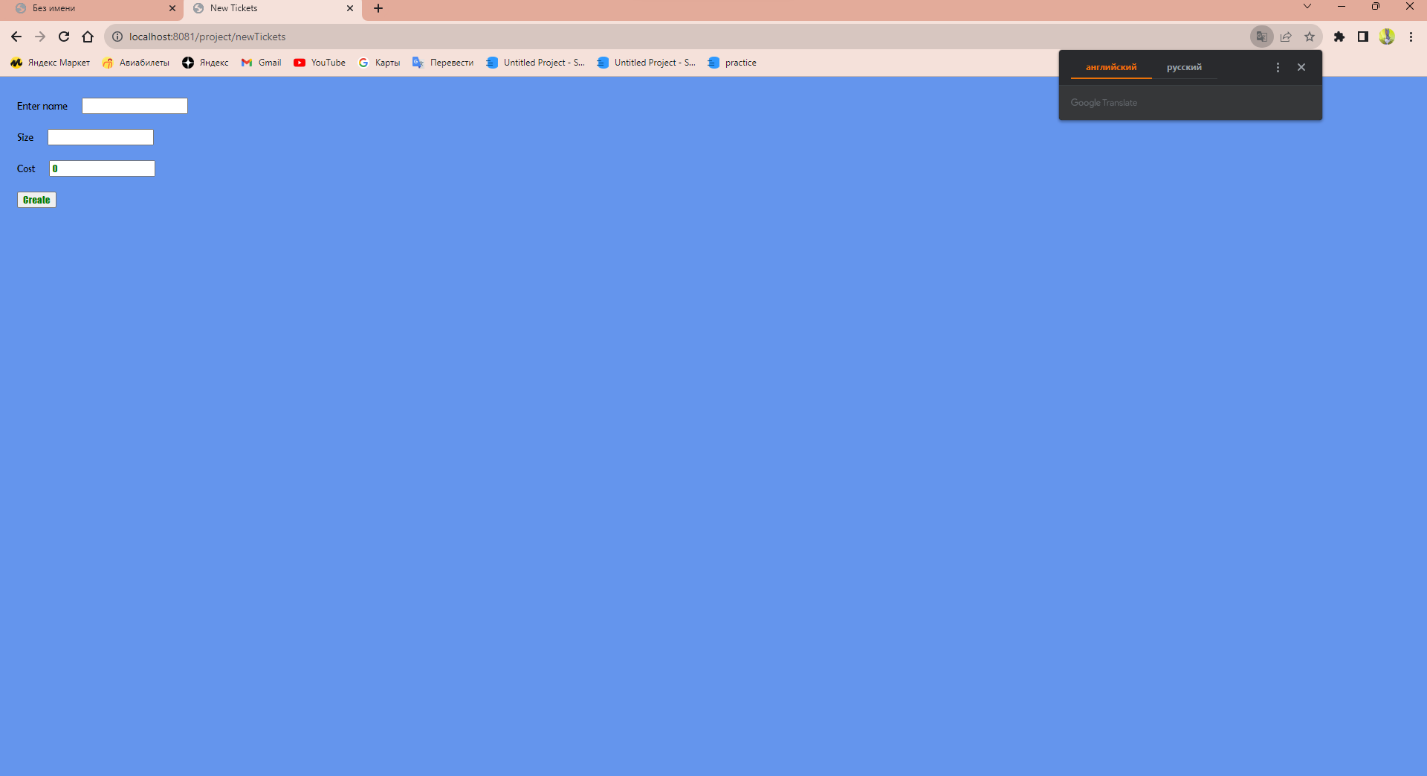


Рисунок 11. Список файлов.

Итог:







Вывод: В этой практической работе была продемонстрирована работа с паттерном DAO, также была реализована общая навигационная страница. Мы применили все полученные знания на занятии и применили их на практике.

Практическая работа №3

Цель работы: В данной практической работе необходимо подключить любую СУБД(MySQL, PostgreSQL, MS SQL) и изменить модели из предыдущей работы под новые технологии.

Требования:

1) На 3 необходимо просто переделать предыдущую работу с подключением выбранной СУБД, сделать валидацию на каждое поле с помощью аннотаций (пример @NotBlank или @Size).

2) На 4 добавить поиск определенной записи, можете выбрать любое поле, по которому будет происходить поиск.

3) На 5 написать универсальные классы для CRUD операций.

1. Создаем БД и подключаем к проекту.

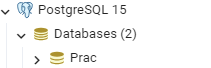


Рисунок 1. База данных.



Рисунок 2. Подключение к проекту.

2. Затем после подключения у нас подрубаются соответствующие зависимости для дальнейшей работы.

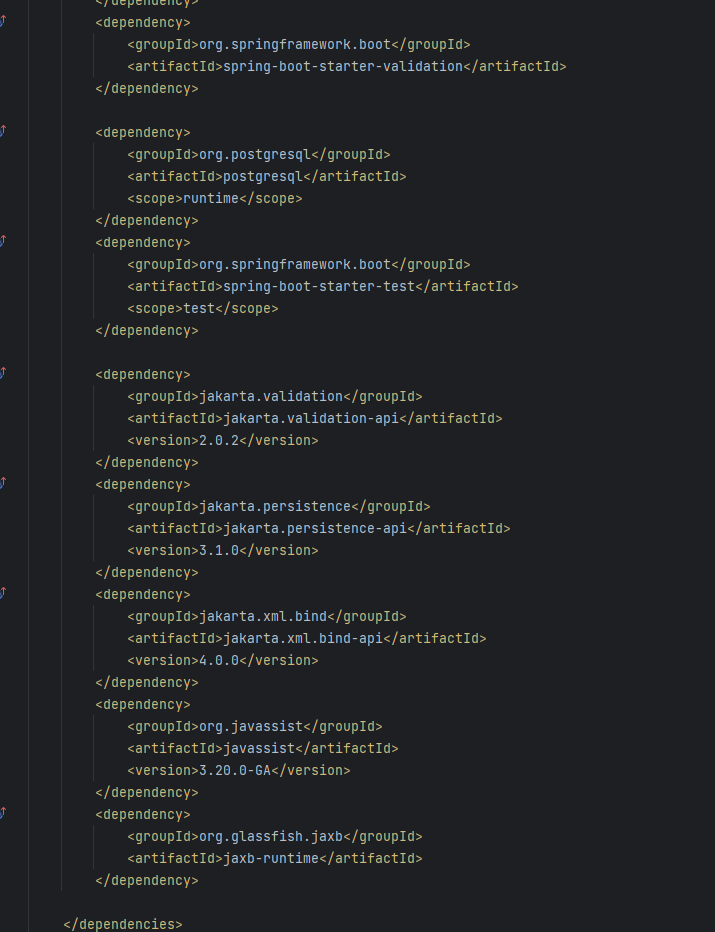


Рисунок 3. Дополнительные зависимости.

3. Далее к уже имеющейся модели мы должны добавить соответствующие условия для обработки их после внесения в базу.



Рисунок 4. Дополнительные условия.

4. Также создаются репозитории интерфейса.

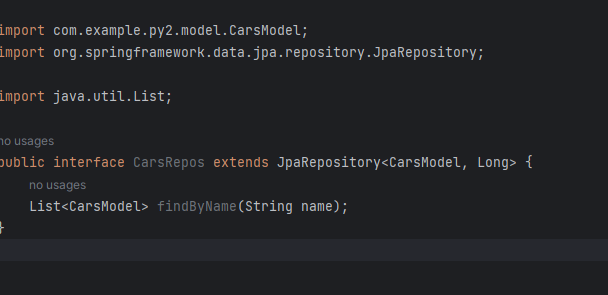


Рисунок 5. Интерфейс-репозиторий.

5. В контроллере все старые действия изменились.

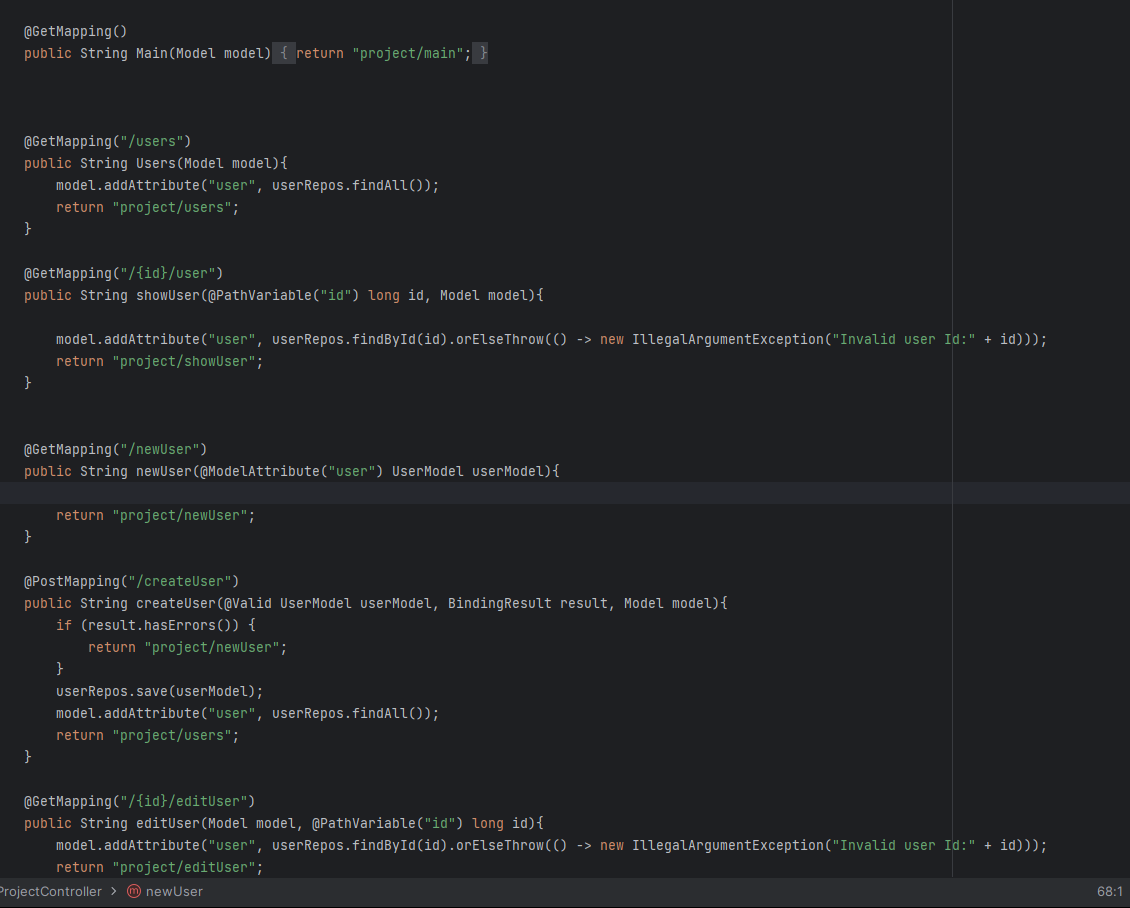


Рисунок 6. Изменённые действия.

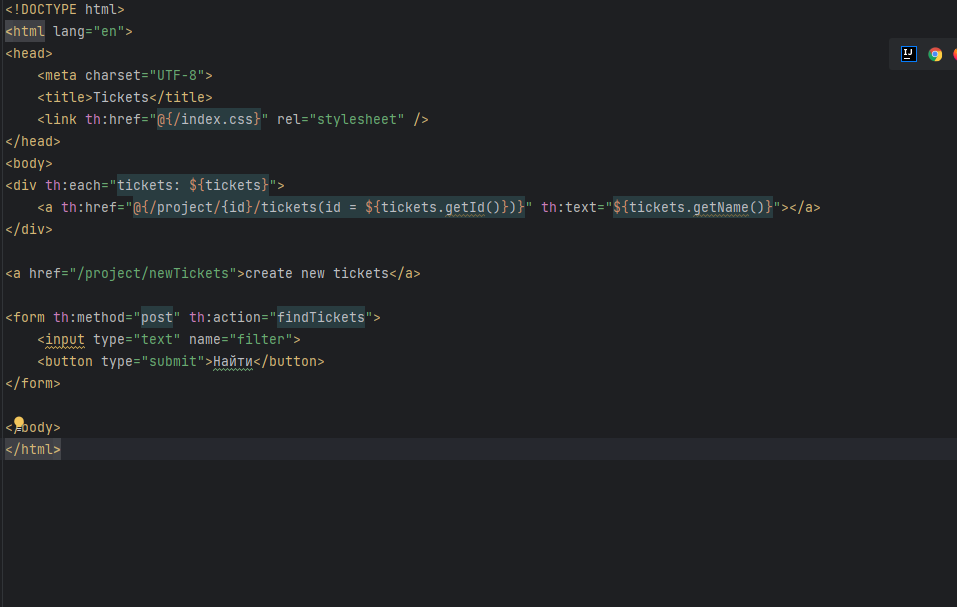


Рисунок 7. HTML.

Вывод: В данной практической работе была подключена СУБД PostgreSQL и изменены модели из предыдущей работы под новые технологии. Для каждого поля модели была создана валидация с помощью аннотаций. Также был реализован поиск объектов модели в БД с помощью параметра названия объекта.

Практическая работа №4

Цель работы: Реализовать 3 типа связей, описать их и продемонстрировать их работу, для этого следует создать дополнительные модели и связать с уже созданными таблицами или между собой.

1. Была реализована связь One to One, через одноимённую аннотацию. С помощью аннотации JoinColumn создаётся новый столбец, поддерживающий связь.

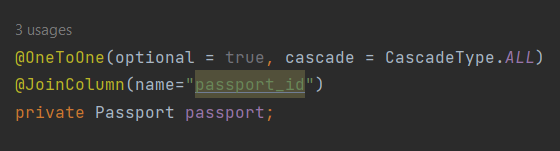


Рисунок 1. связь со стороны User.

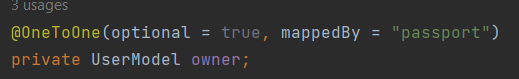


Рисунок 2. связь со стороны Passport.

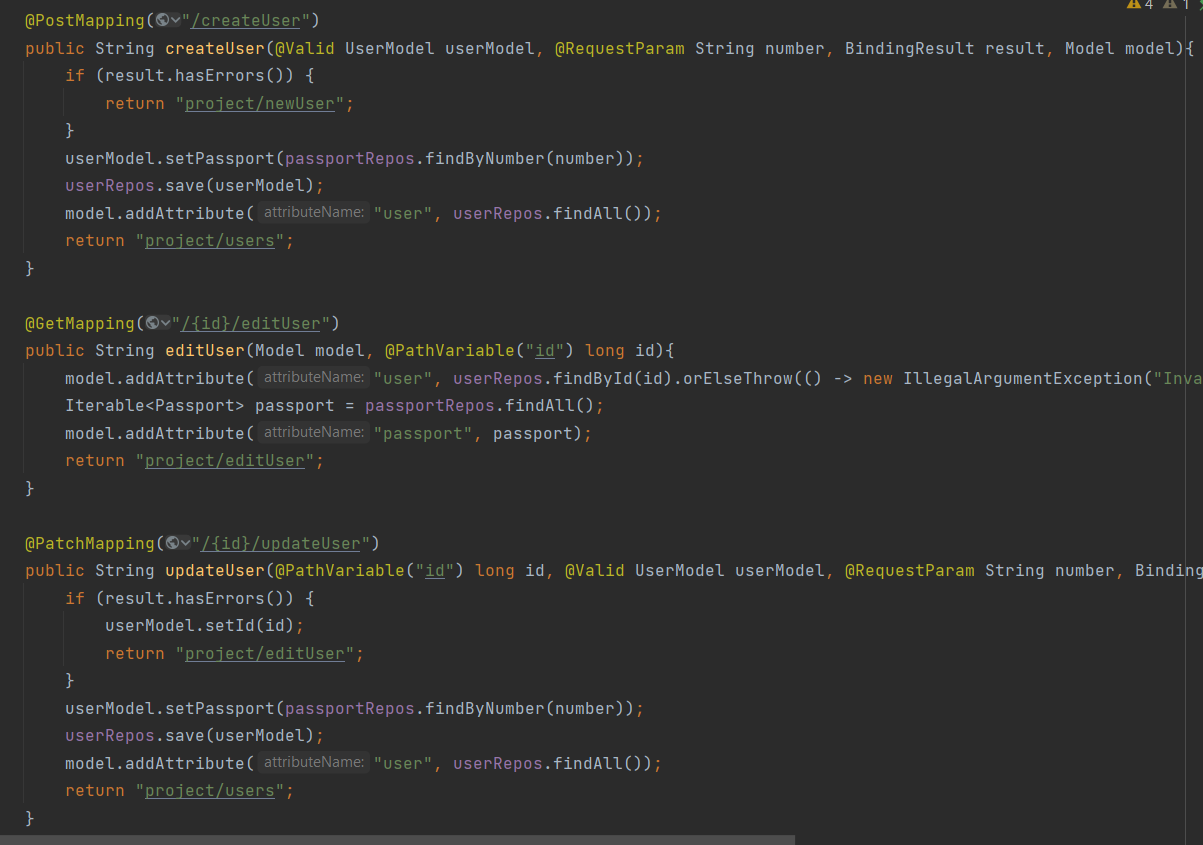


Рисунок 3. Изменение контроллера под связь таблиц

2. Для связи многих к одному используются две связанные аннотации: OneToMany и ManyToOne.

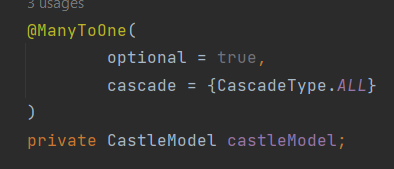


Рисунок 4. ManyToOne.

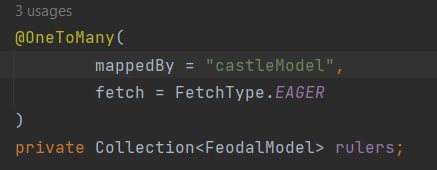


Рисунок 5. OneToMany.

3. В конце реализуется связь многие ко многим. Для этого используется аннотации ManyToMany и JoinTable, где создаются два столбца, связанные с таблицами связи MtM.

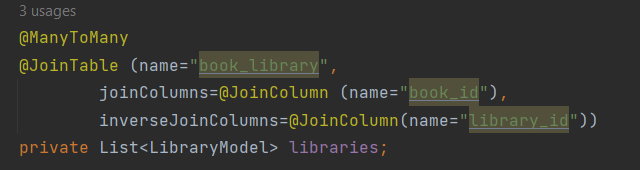


Рисунок 6. Подключение со стороны таблицы книг.

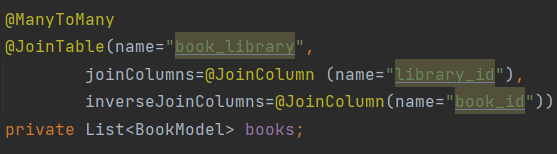


Рисунок 7. Подключение со стороны таблицы библиотек.

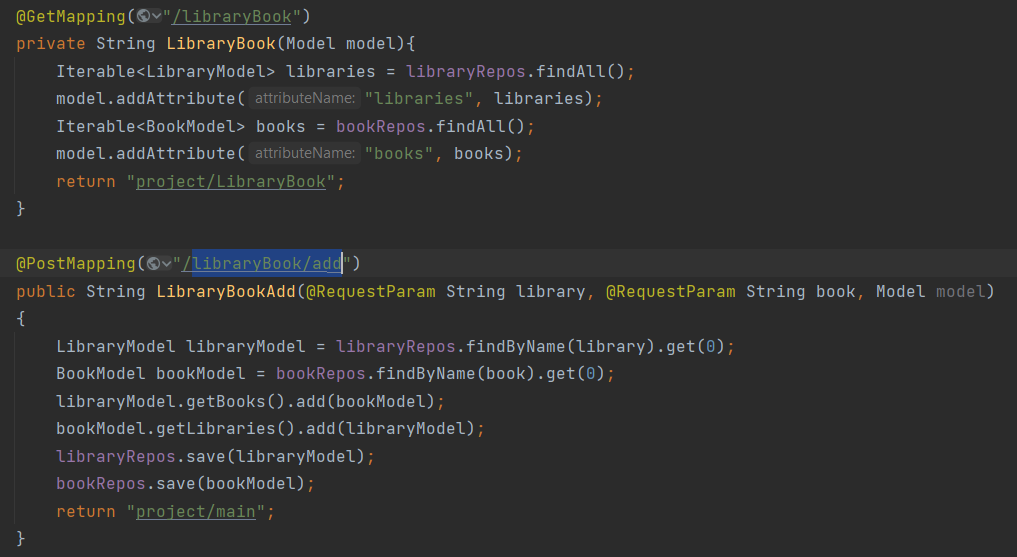


Рисунок 8. Методы промежуточной таблицы в контроллере.

Вывод: в данной работе были реализованы 3 типа связей между таблицами в БД, и реализованы дополнительные модели, которые участвовали в образовании связей с уже созданными таблицами.

Практическая работа №5

Цель: добавить регистрацию и авторизацию в свой проект.

1. Для реализации авторизации и регистрации будем использовать несколько дополнительных зависимостей.



Рисунок 1. Зависимости для авторизации.

2. Определенный класс обеспечивает взаимодействие страницы входа и базы данных.

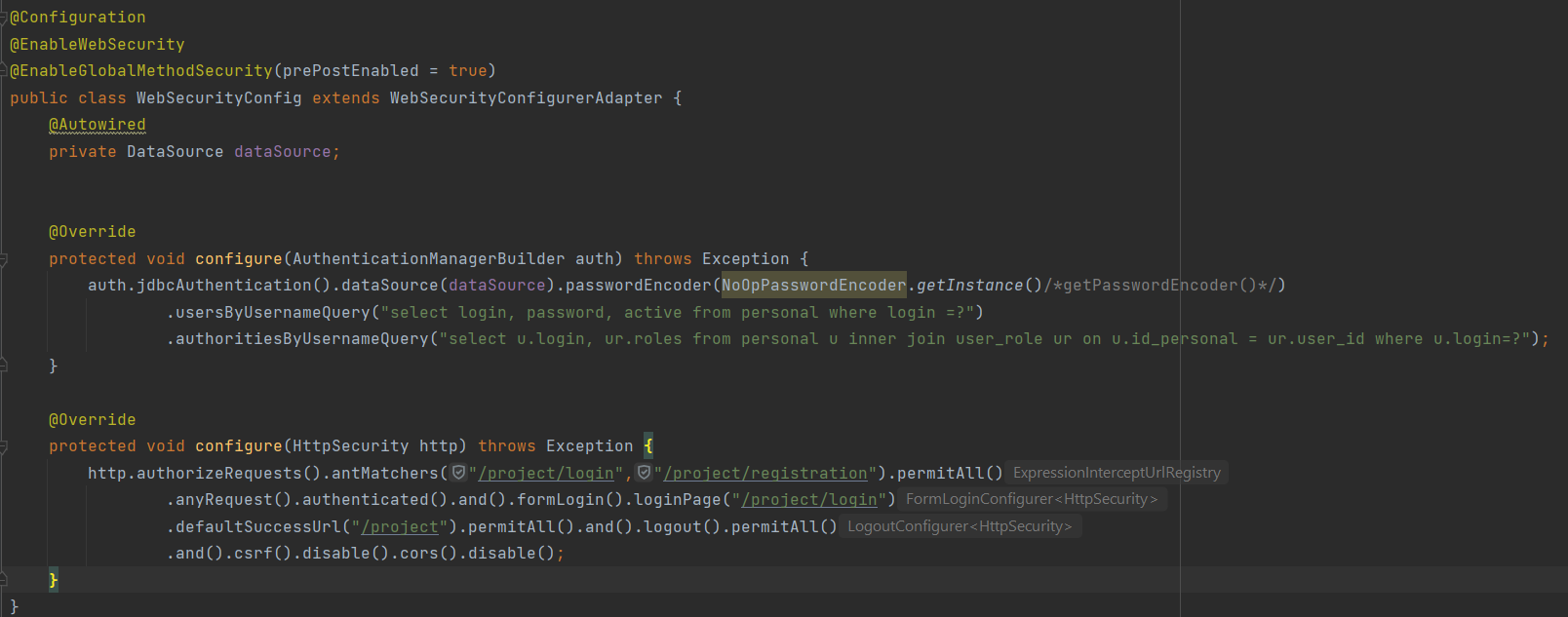


Рисунок 2. WebSecurityConfig.

3. Следующий класс добавляет контроллер для представления страницы авторизации пользователя.

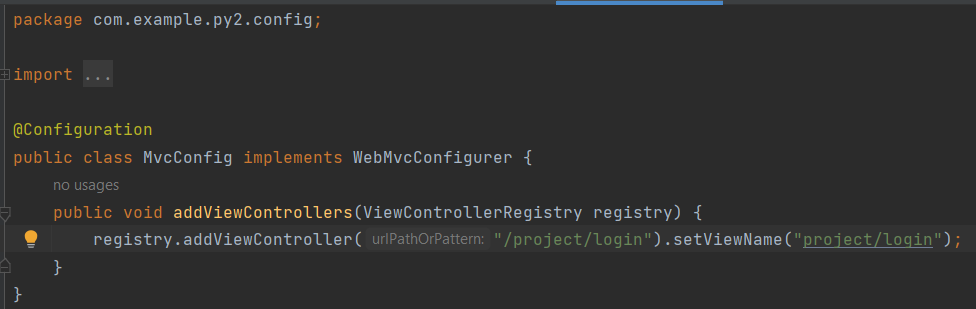


Рисунок 3. MvcConfig.

4. Делаем роли для аккаунтов.

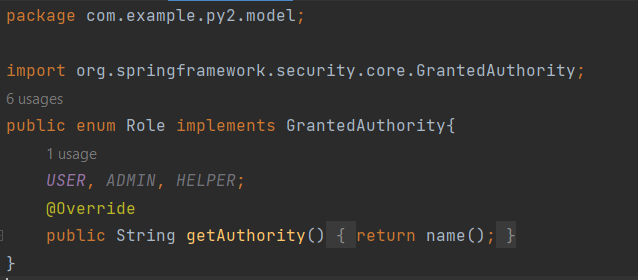


Рисунок 4. Роль.

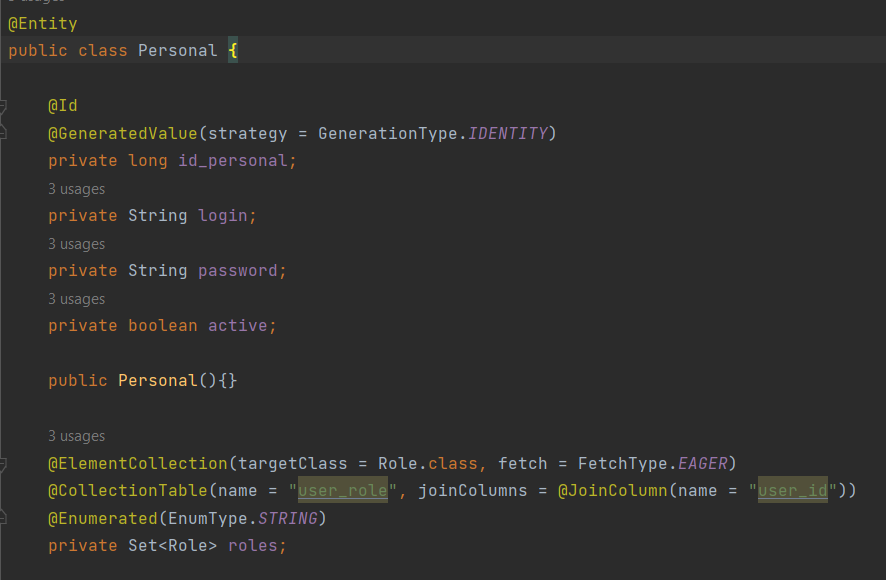


Рисунок 5. Personal.

5. Добавляем вызовы для проверки.



Рисунок 6. Controller.

6. Создаются представления входа и добавления аккаунта.



Рисунок 7. Login.

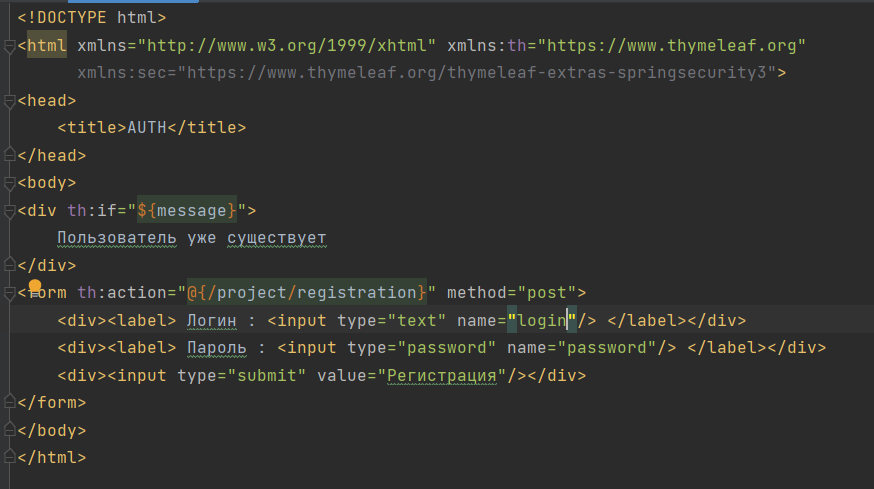


Рисунок 8. Registration.

7. Проверка на работоспособность.



Рисунок 9. Регистрация.

Вывод: В этой работе мы реализовали авторизацию и регистрацию пользователей.

Практическая работа №6

Цель: реализовать механизм шифрование пароля пользователя. Добавить разграничение прав доступа для пользователей.

1. Добавляются дополнительные роли в Role enum.



Рисунок 1. Роль.

2. Делим контроллеры на трое.

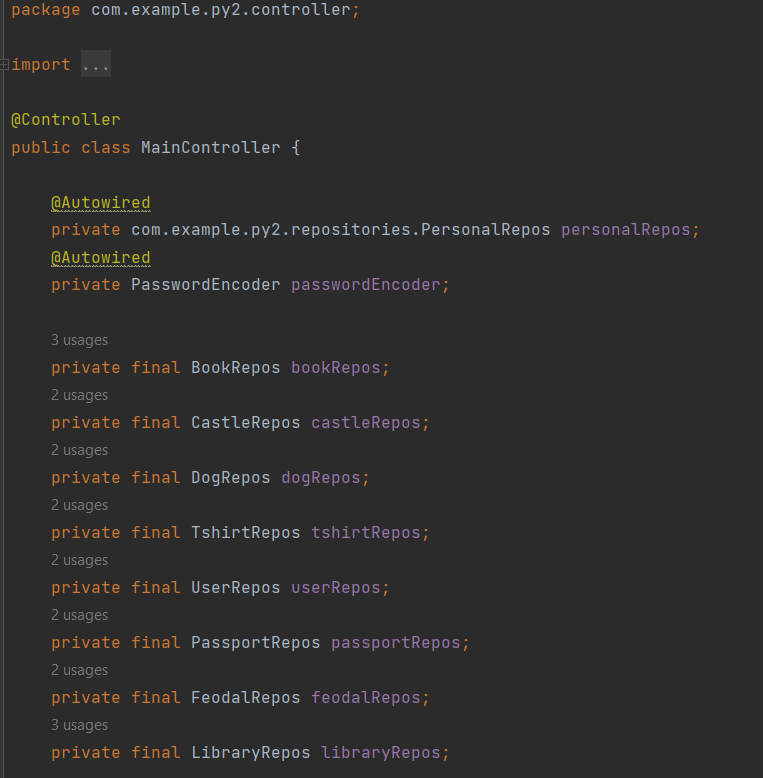


Рисунок 2. 1 Controller.



Рисунок 3. 2 Controller.

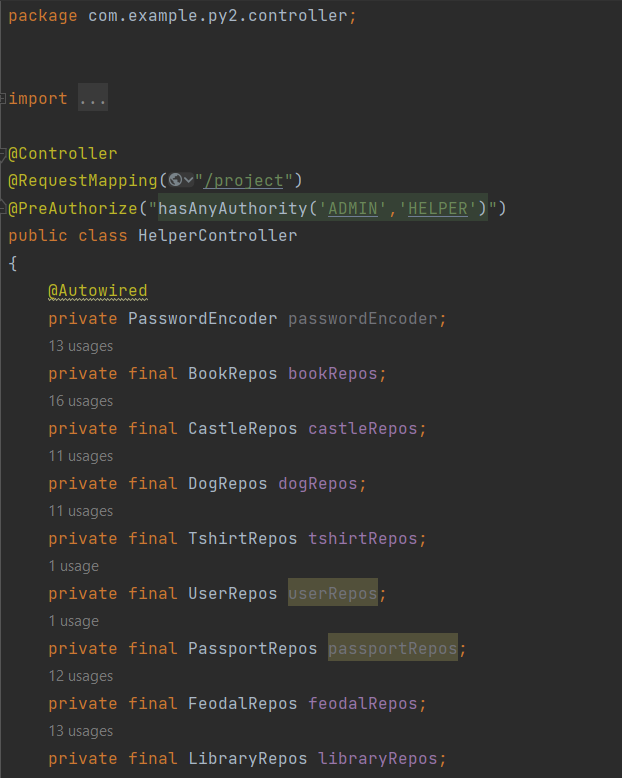


Рисунок 4. 3 Controller.

3. Создаем верстки страниц.



Рисунок 5. Список пользователей.



Рисунок 6. Подробная информация.



Рисунок 7. Редактирование профиля.

5. Взаимодействие админа с версткой.

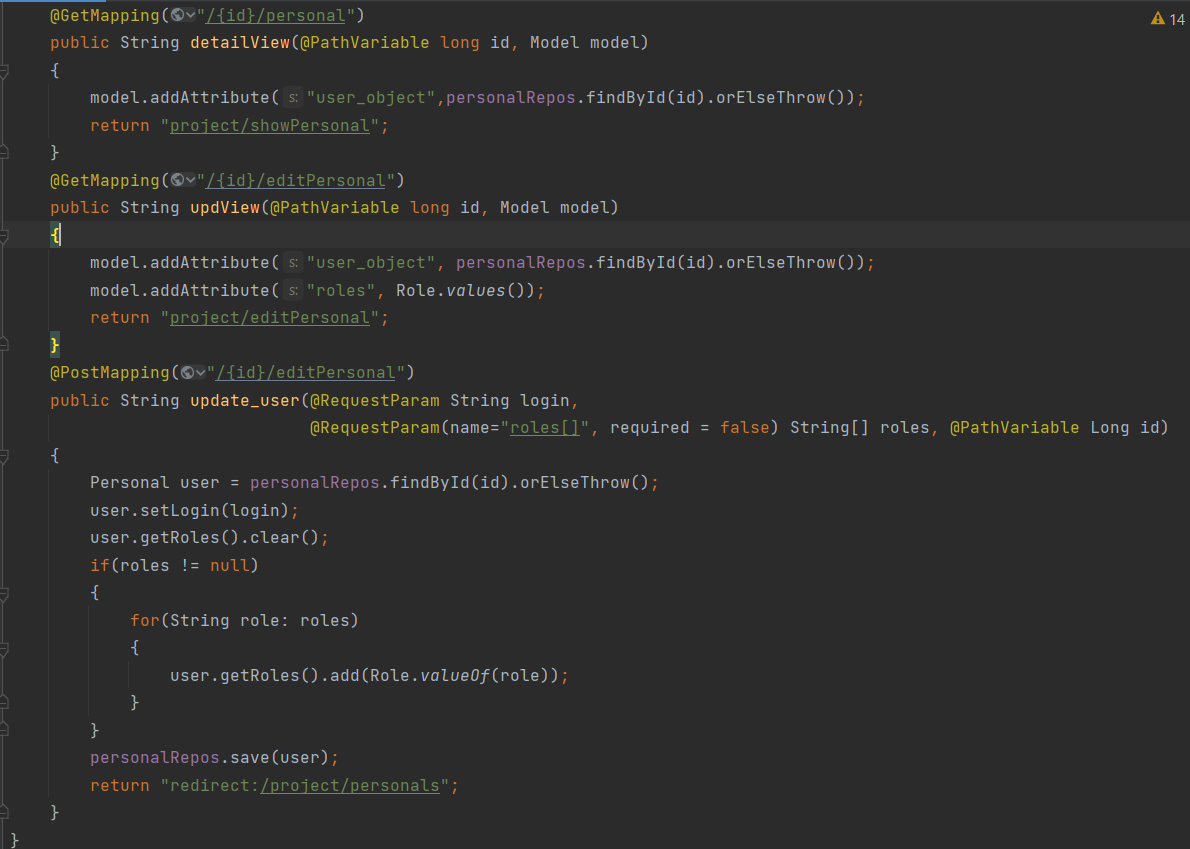


Рисунок 8. Контроллер Админа.

Вывод: В данной практической работе мы реализовали шифровку пароля пользователя и разобрались с тем, как настраивать это.