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

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

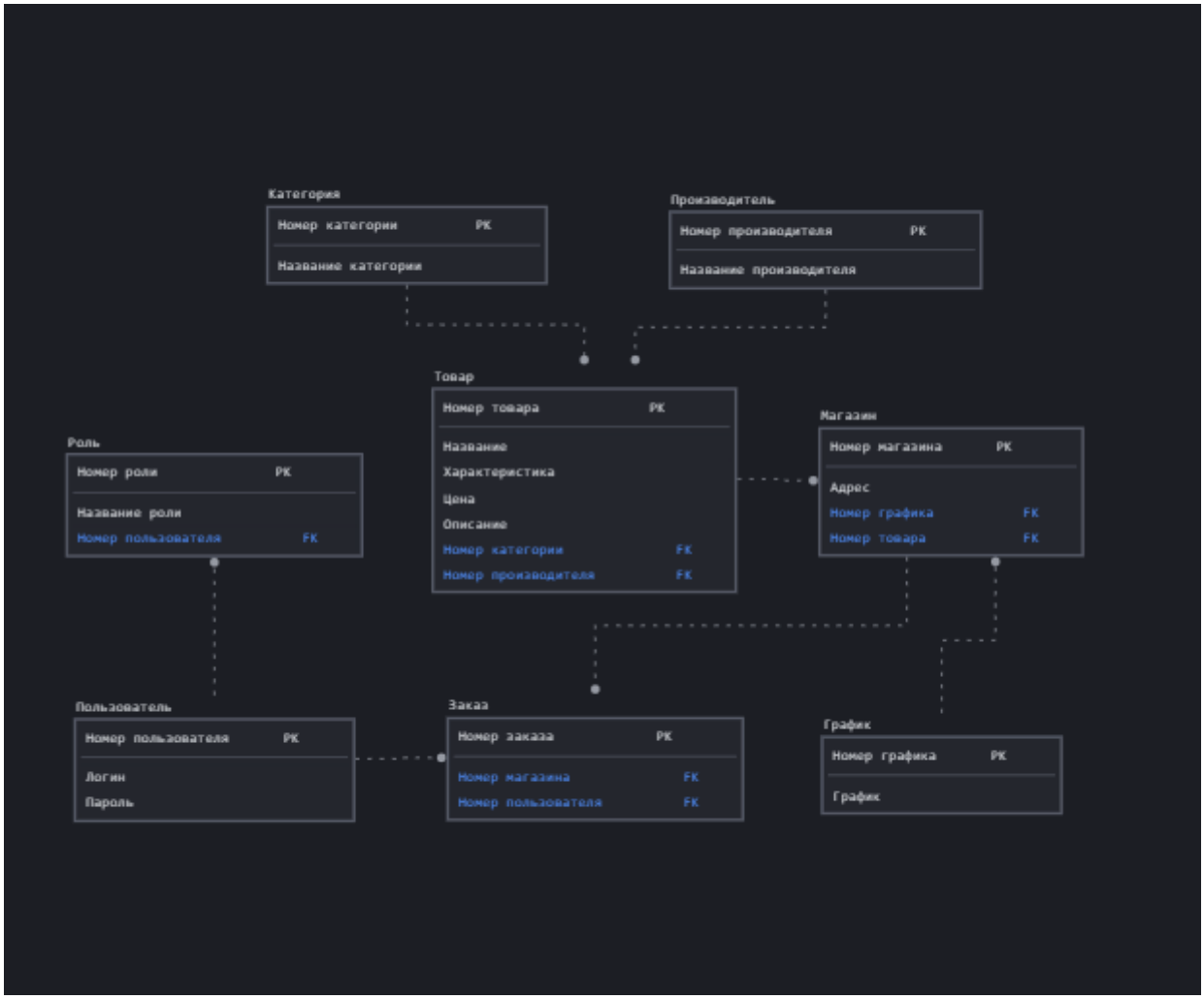


Рисунок 1 Инфологическая модель данных

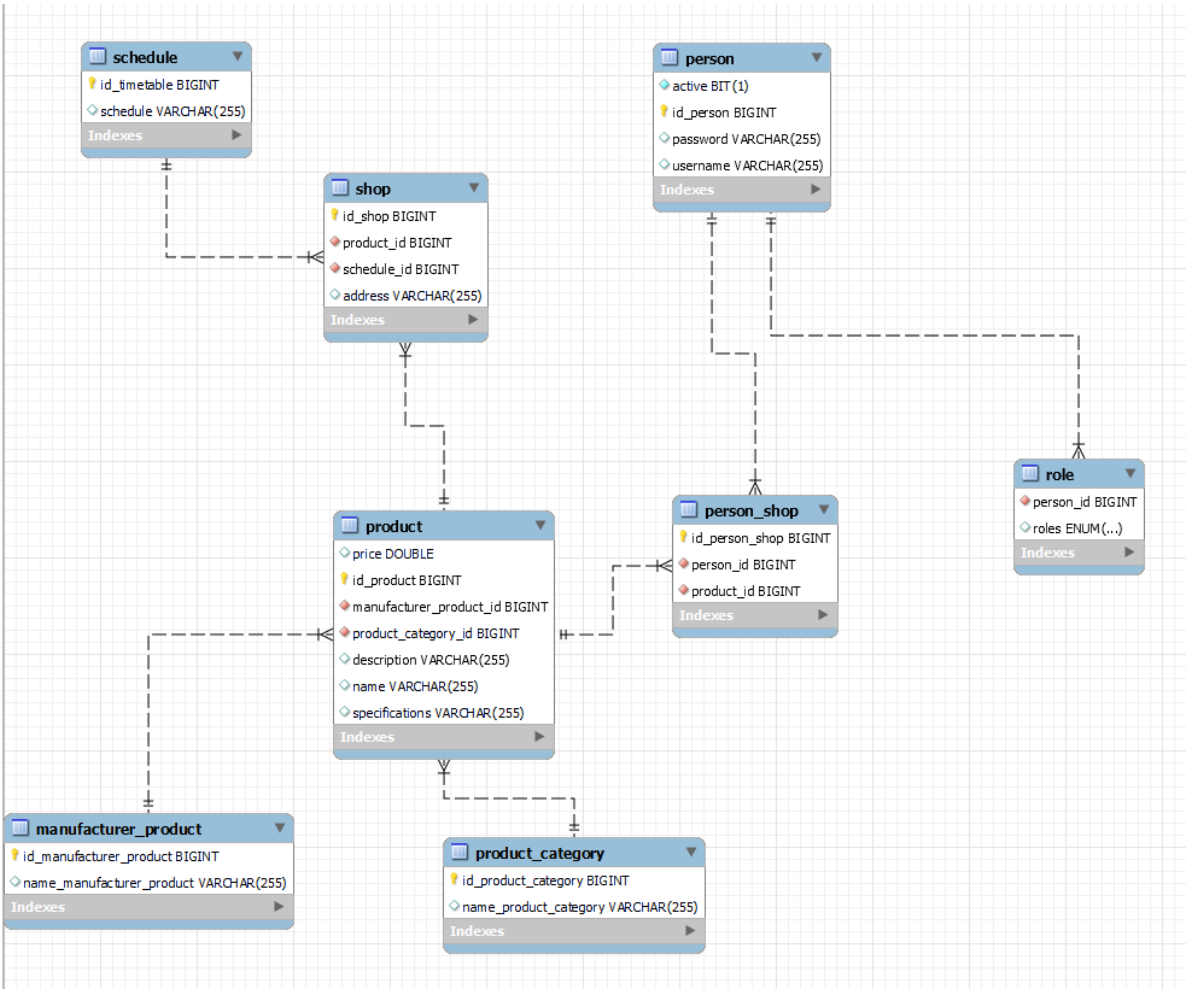


Рисунок 2 Даталогическая модель данных

Словарь данных

Таблица 1 Словарь данных

Таблица 2 – 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.getIdProductCate
gory());

productCategory.setNameProductCategory(productCategoryRequest.getNameProd
uctCategory());

            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.Authentica
tionManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlob
alMethodSecurity;
```

```
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSec
urity;



import
org.springframework.security.config.annotation.web.configuration.WebSecurityCo
nfigurerAdapter;
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(getPassword Encoder())
            .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/"+idManufacturer
Product, 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;
    }

}
```

```
server.port=8081
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/YP_0401
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.mvc.hiddenmethod.filter.enabled=true
```

Рисунок 3 Подключение к БД

Работа программы

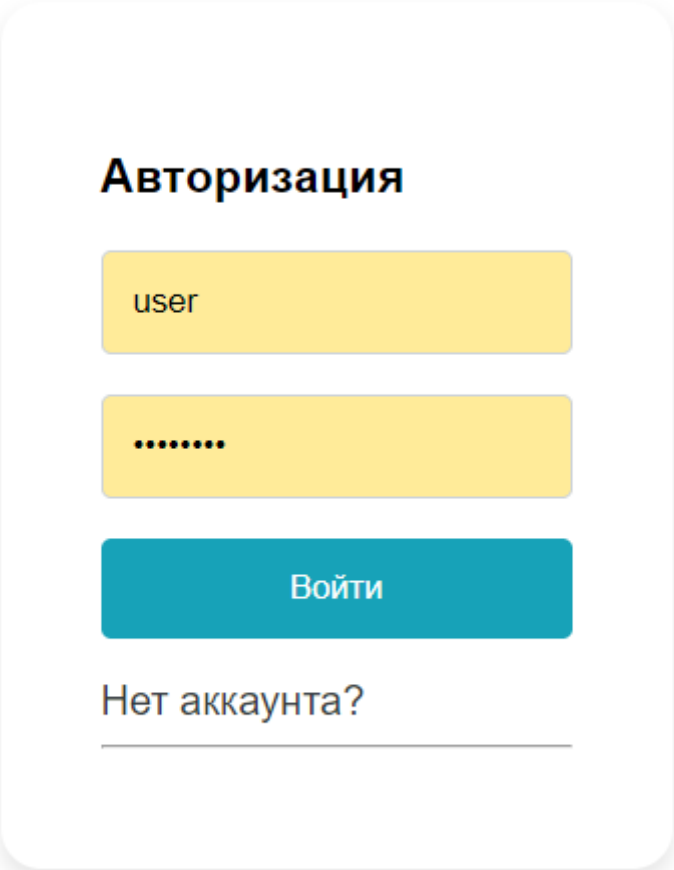


Рисунок 4 Авторизация

Регистрация

user

••••••••

Зарегистрироваться

Есть аккаунт?

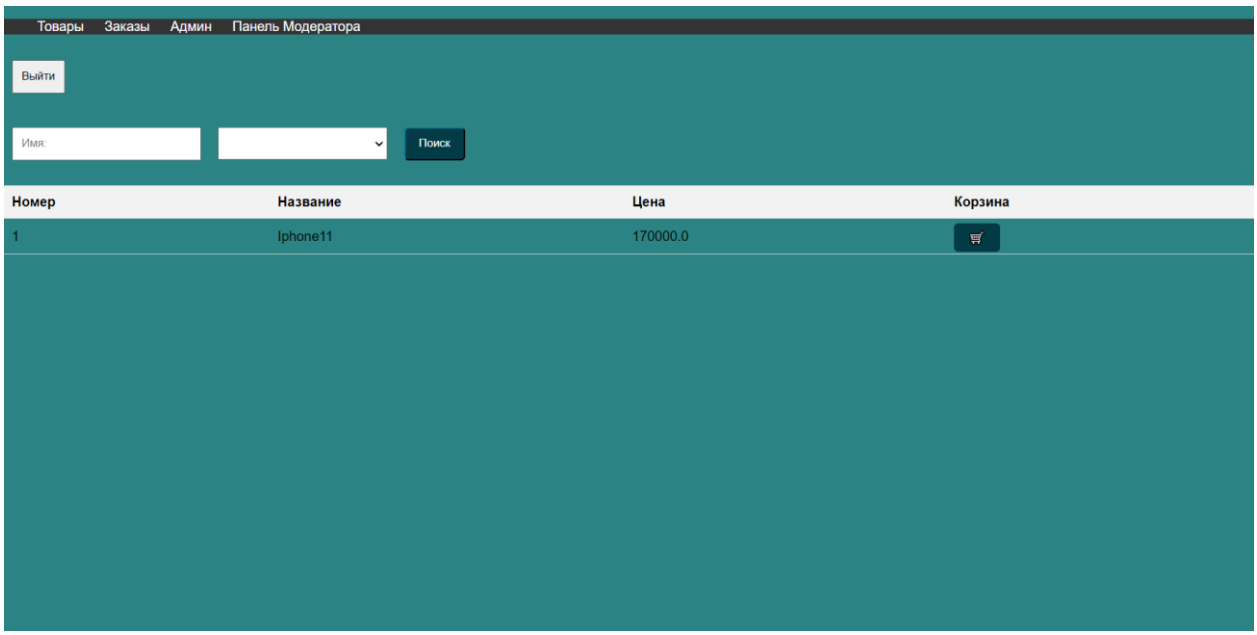


Рисунок 5 Главная страница товаров

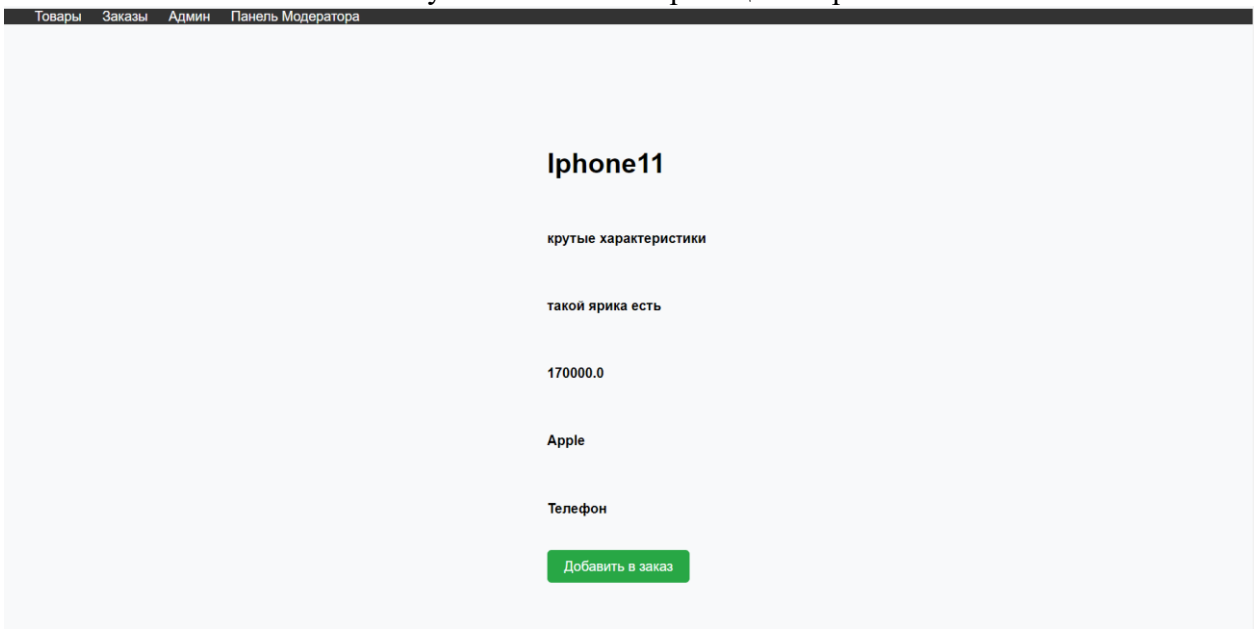


Рисунок 6 Вход в корзину

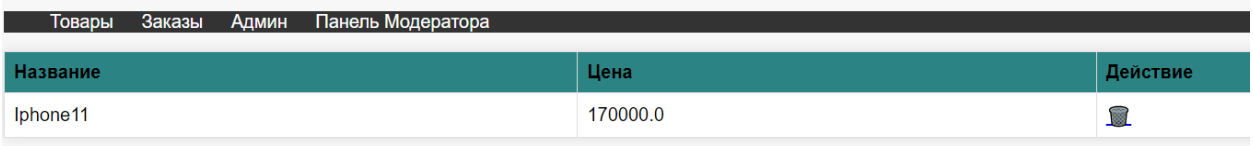


Рисунок 7 Заказы

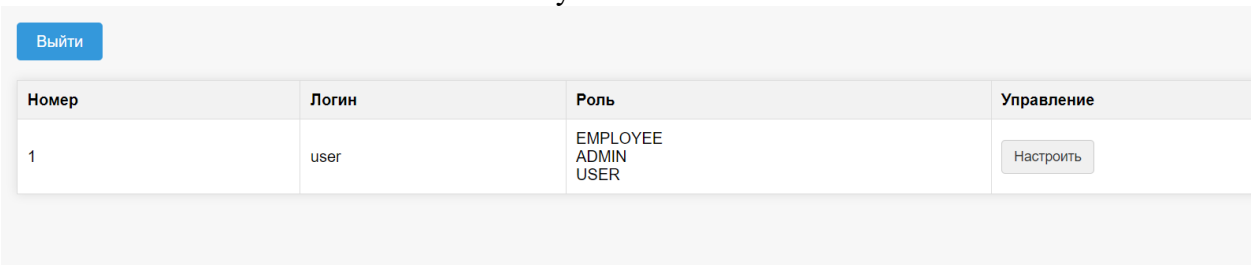


Рисунок 8 Страница Admina

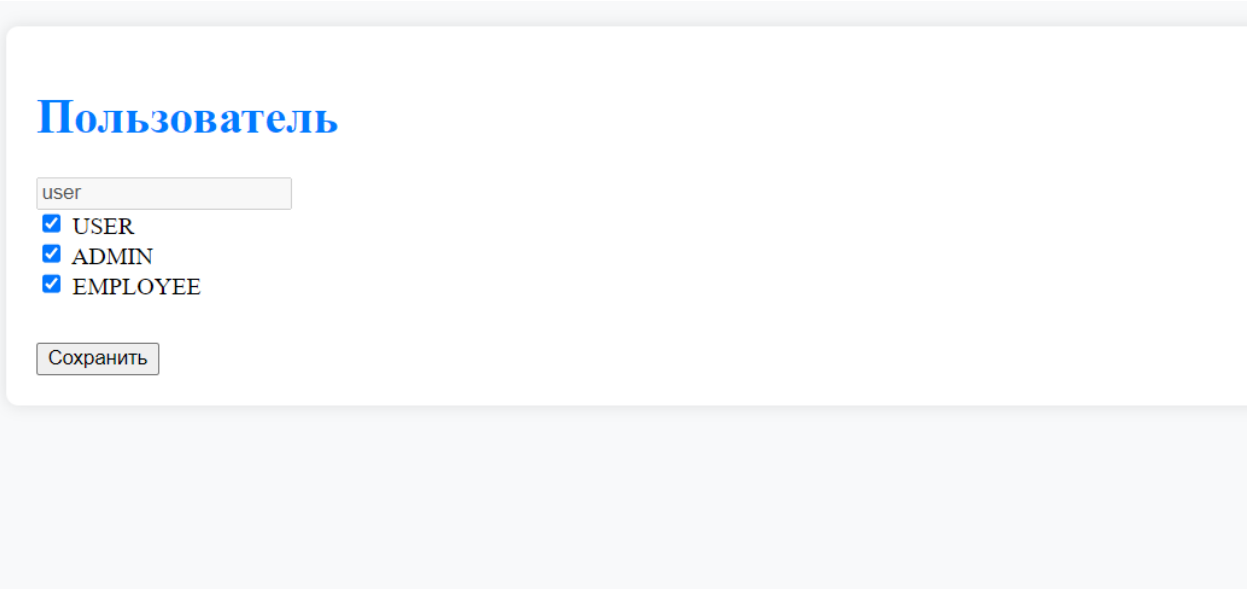


Рисунок 9 Управление настройками над пользователем

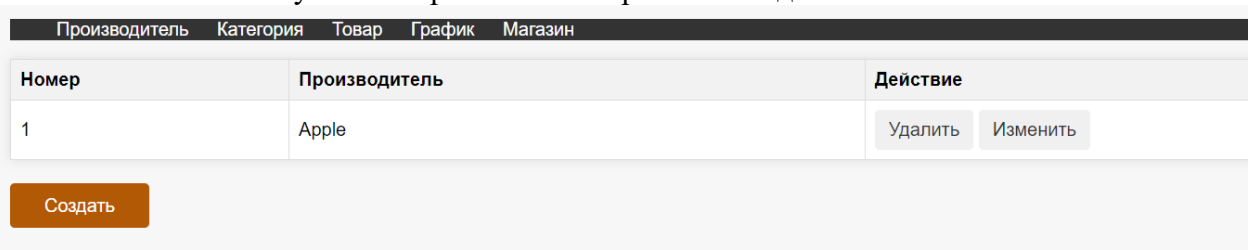


Рисунок 10 Страница Модератора таблица Производитель

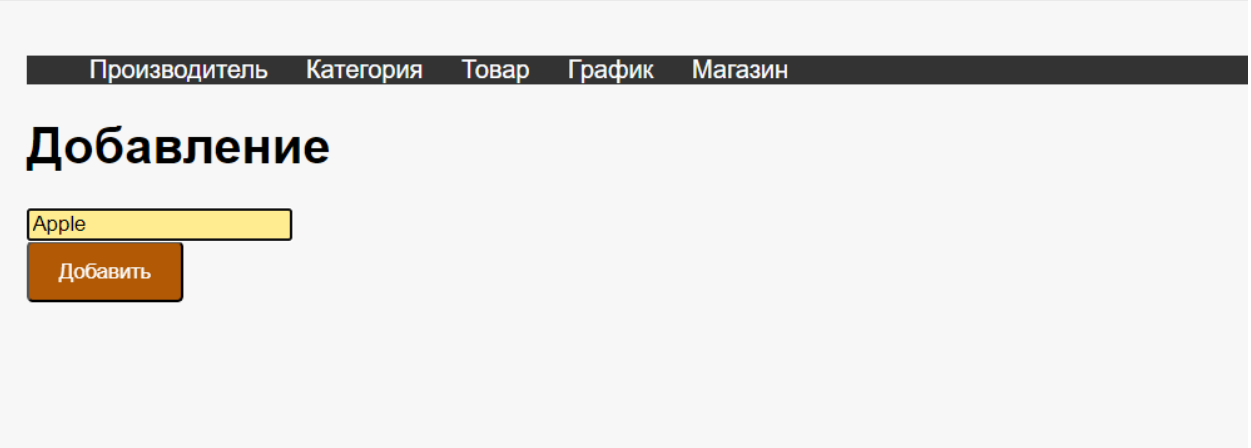


Рисунок 11 Добавление производителя

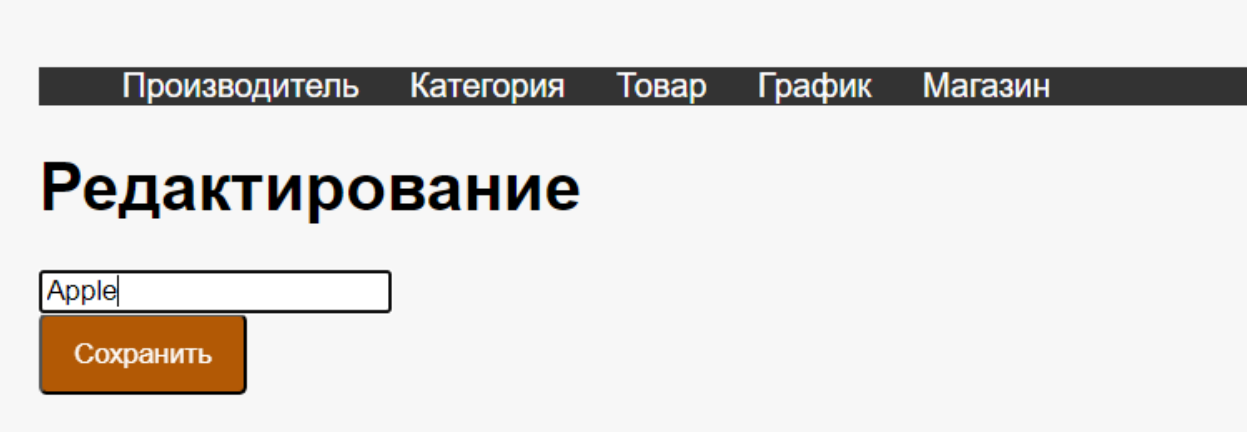


Рисунок 12 Изменение производителя

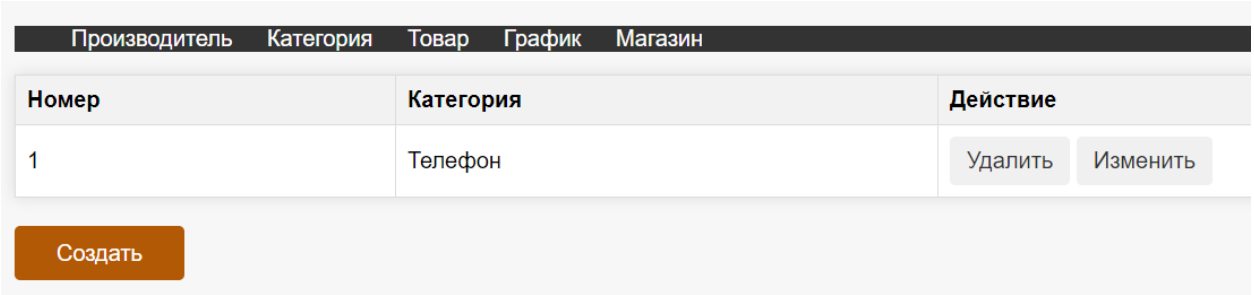


Рисунок 13 Категория

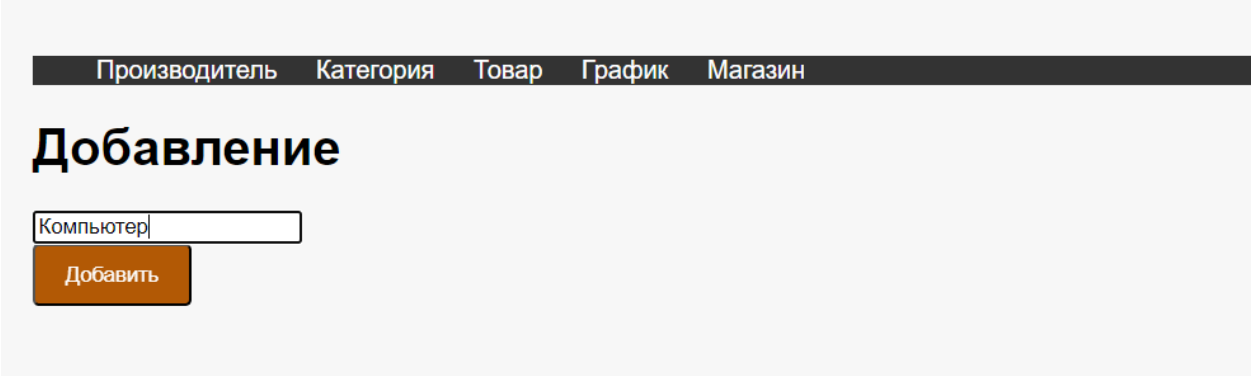


Рисунок 14 Добавление категории

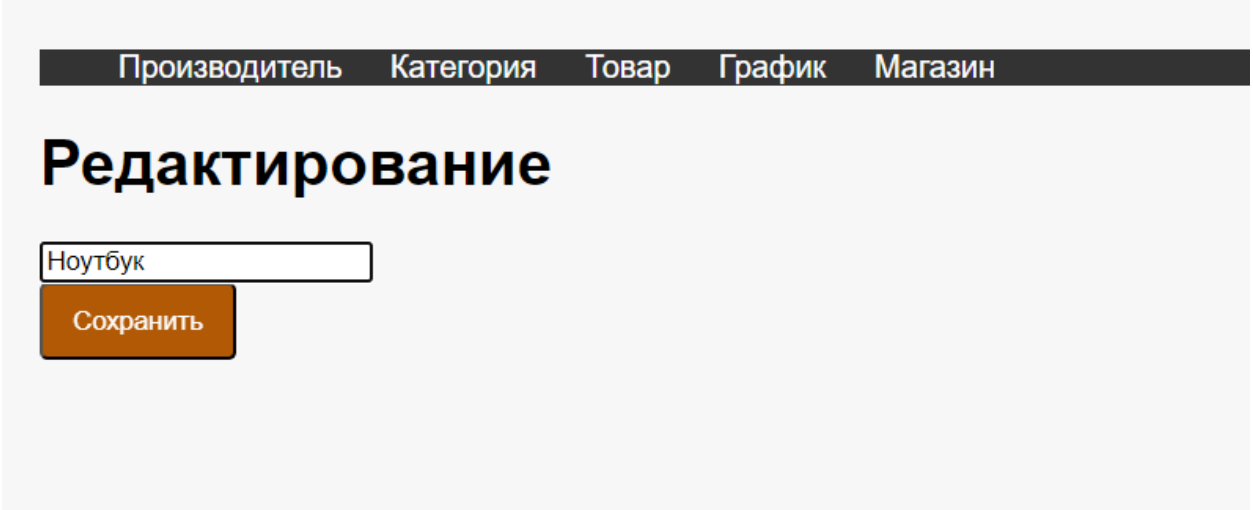


Рисунок 15 Редактирование категории

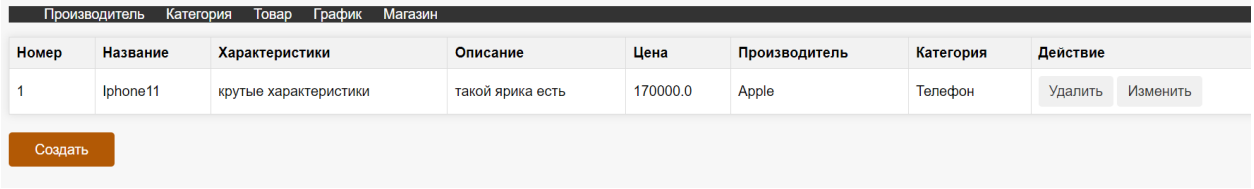


Рисунок 16 Товар

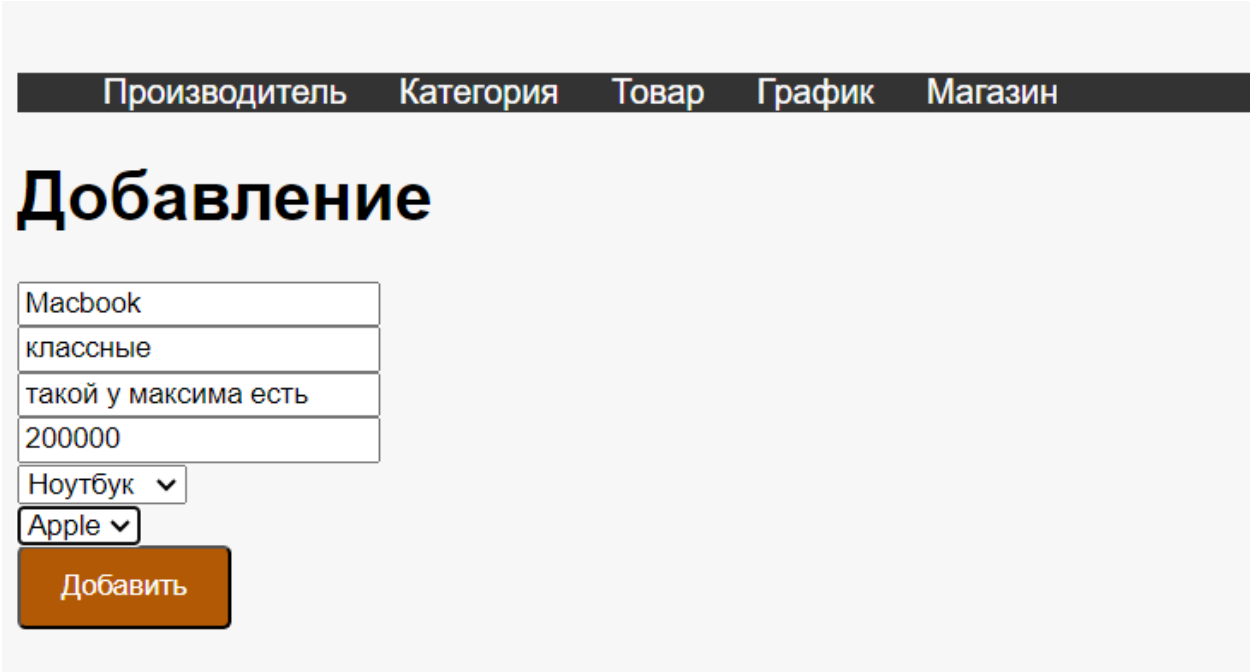


Рисунок 17 Добавление товара

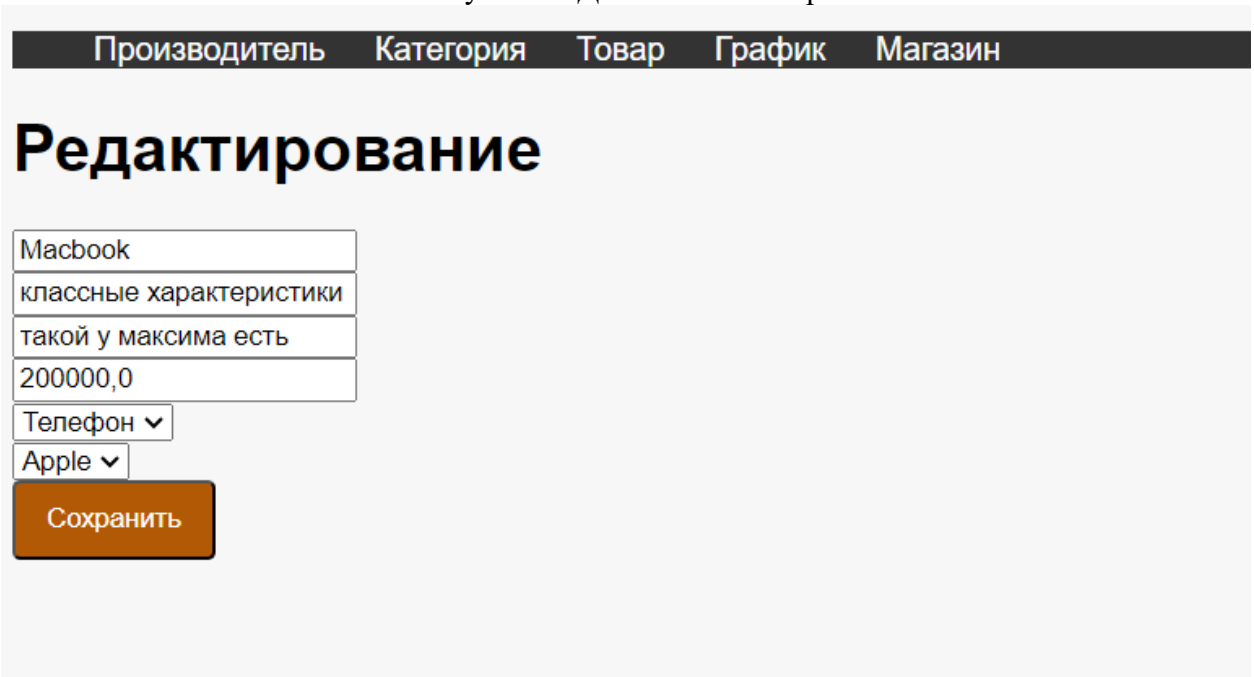


Рисунок 18 Изменение товара

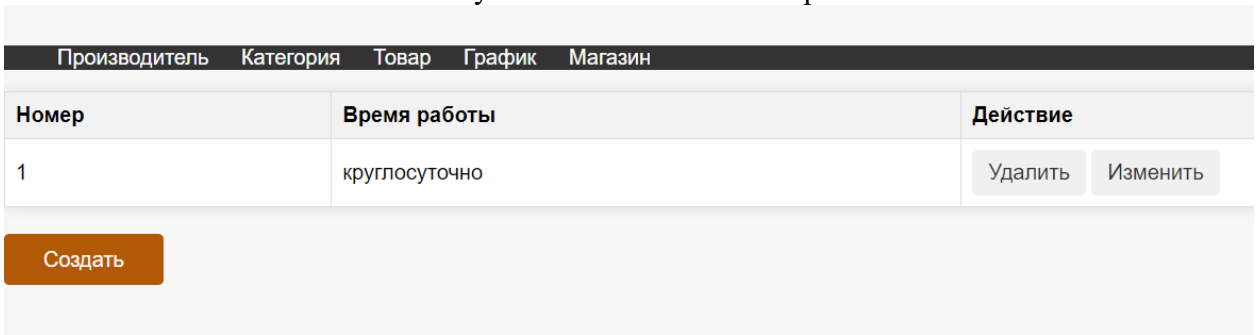


Рисунок 19 График

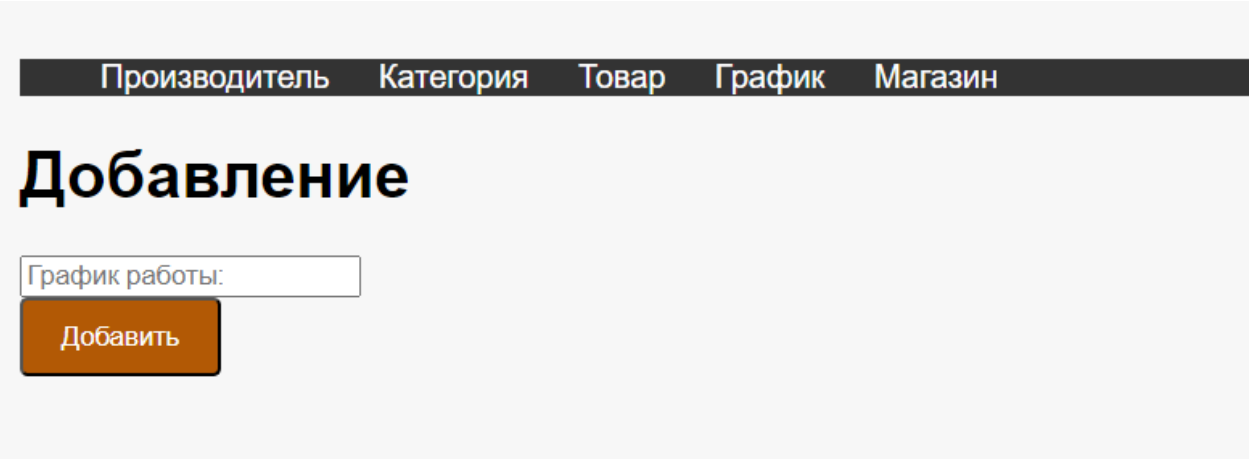


Рисунок 20 Добавление графика

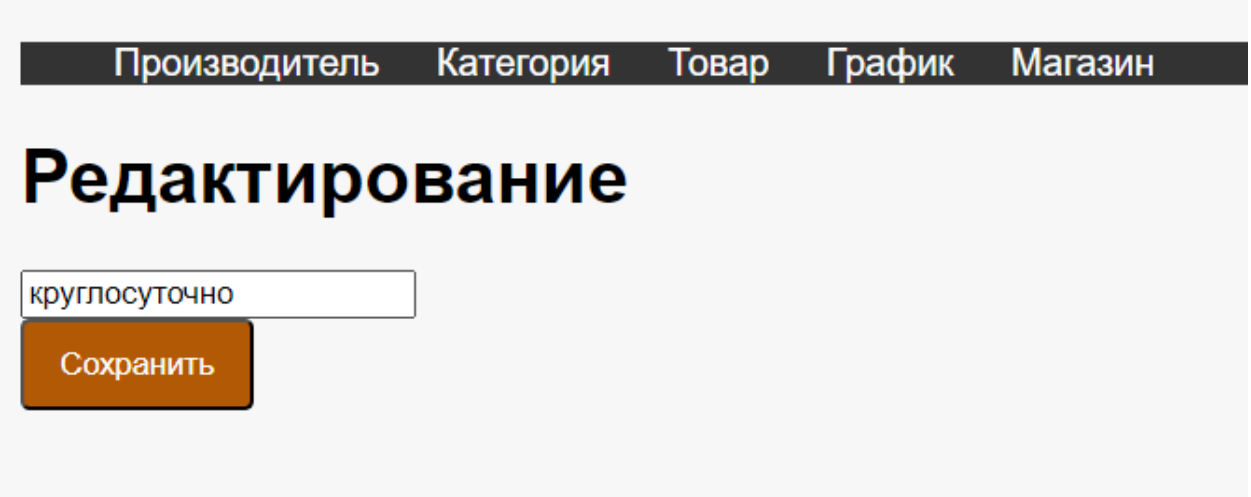


Рисунок 21 Изменение графика

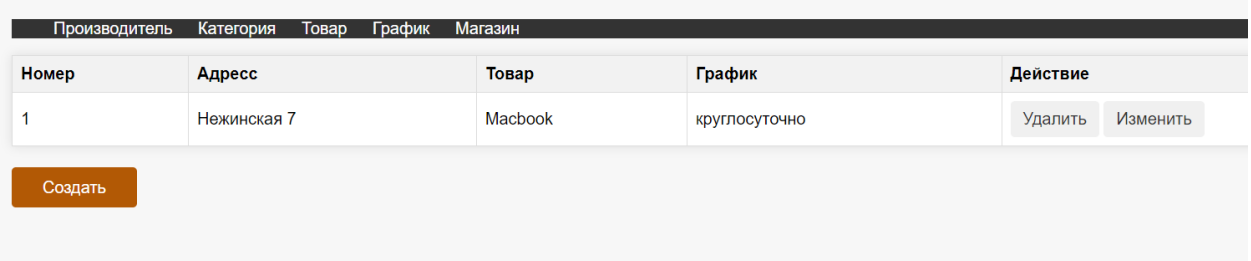


Рисунок 22 Магазин

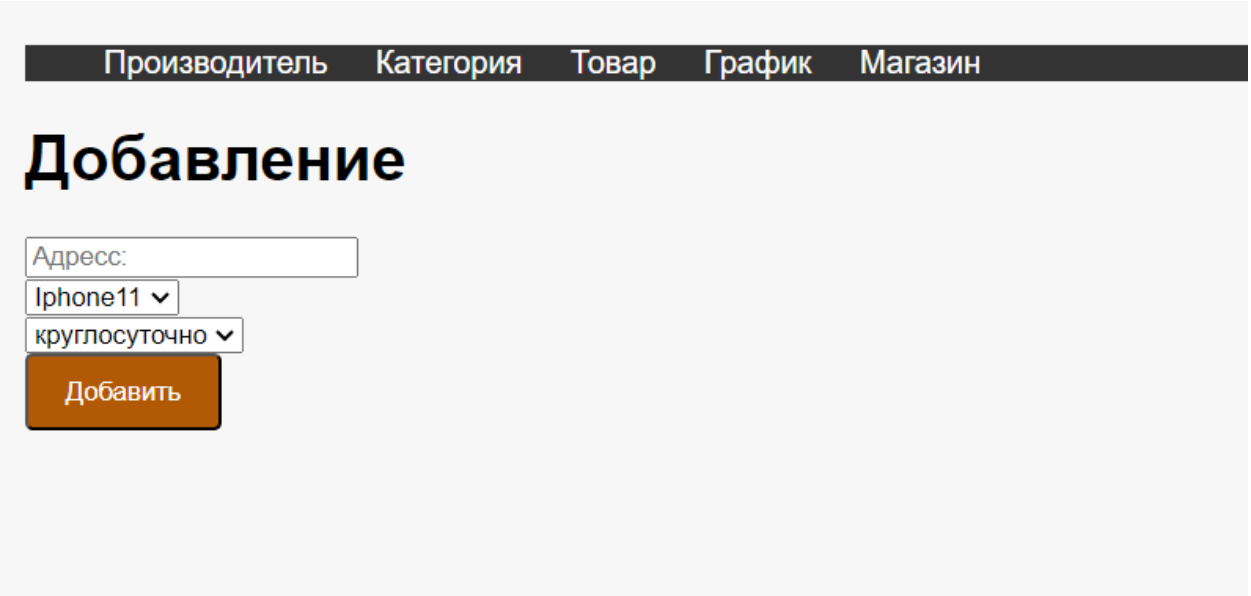


Рисунок 23 Добавление

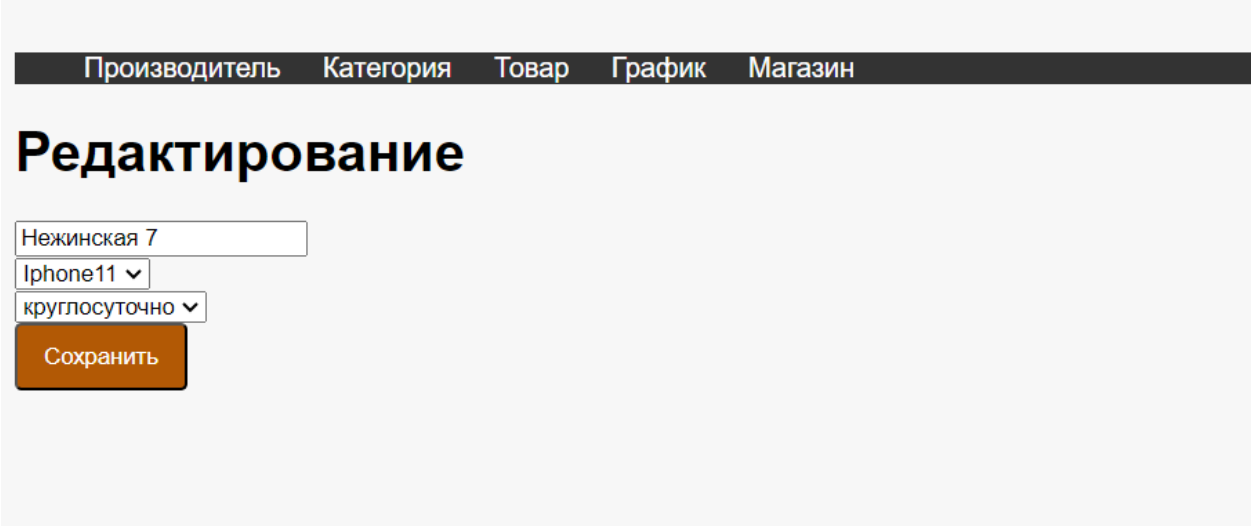


Рисунок 24 Изменение

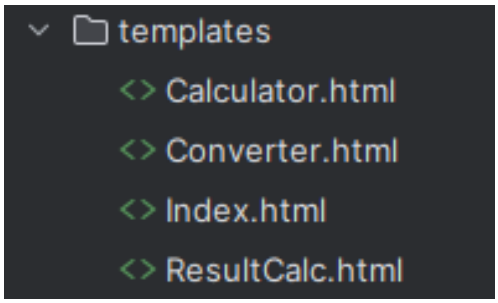
Вывод: в итоге было написано приложение интернет-магазина электронной техники.

Практическая работа №1

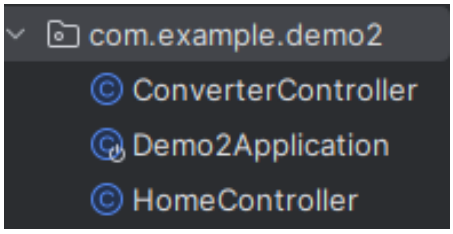
Цель работы: создать калькулятор, конвертер валют. Используя @GetMapping и @PostMapping также использовать @RequestParam

Ход работы:

1. Создаем файлы разметки html для всех необходимых по условию страниц.



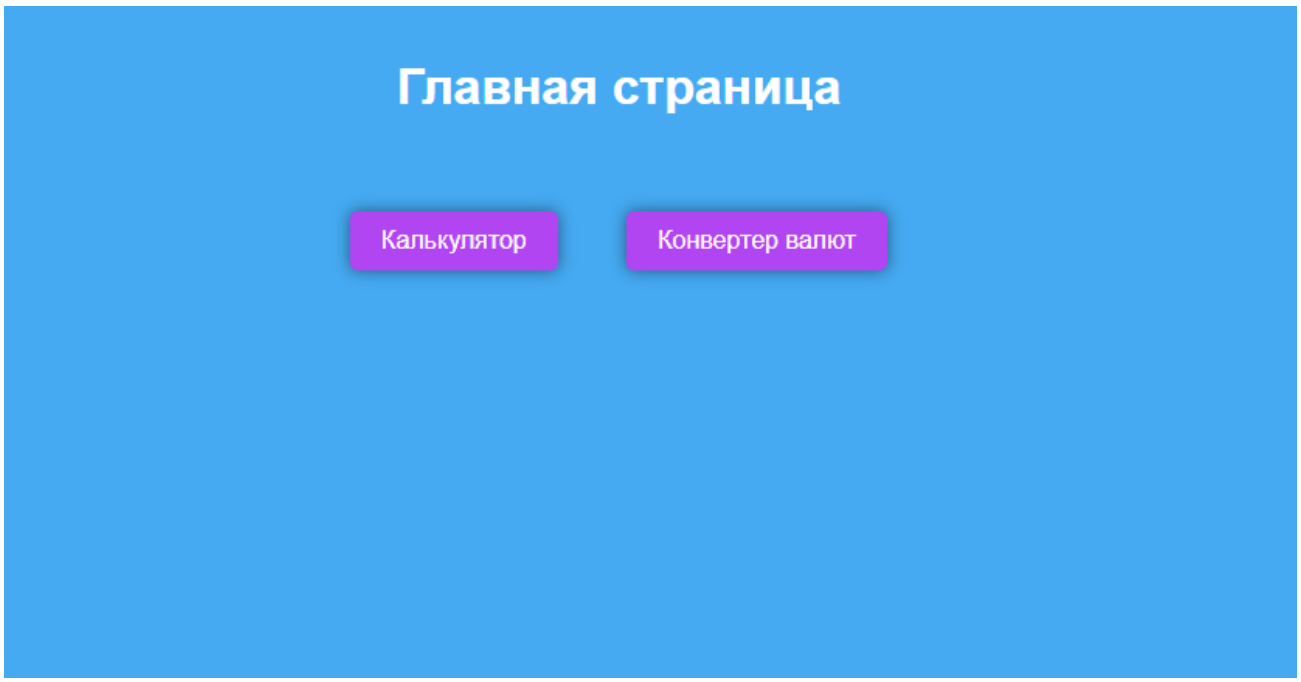
2. Создаем контроллеры для конвертера и для калькулятора



3. Разметка для главной страницы

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Главная</title>
    <style>
        body {
            background:#45aaf2;
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 20px;
            margin: 0;
            width: auto;
            height: auto;
        }

        h1 {
            color: #fff;
            padding: 20px;
            border-radius: 10px;
            margin-top: 0;
        }

        a {
            display: inline-block;
            margin: 20px;
            padding: 10px 20px;
            background:#b245f2;
            color: #fff;
            text-decoration: none;
            border-radius: 5px;
```

Итог



4. Прописываем все в контроллере для калькулятора

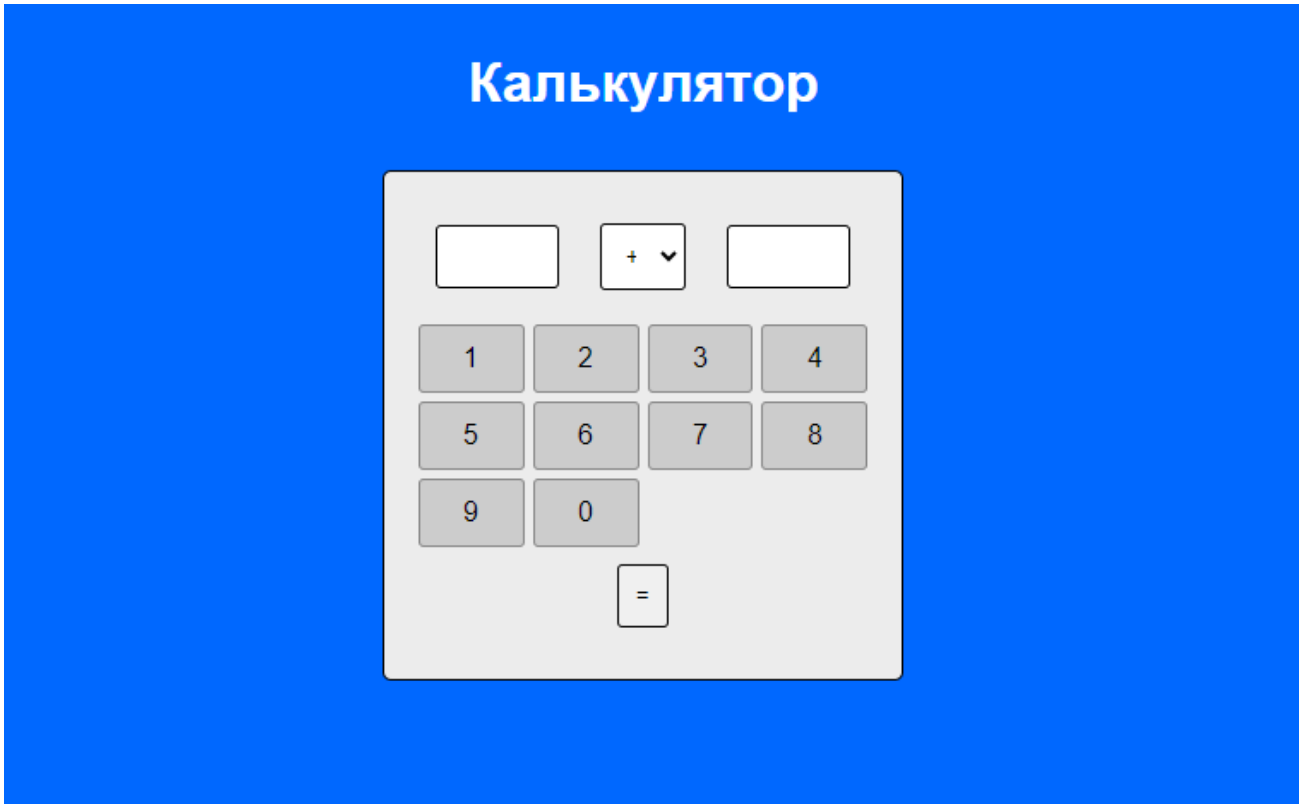
Используем switch для того, чтобы была логика в калькуляторе.

```
        @RequestParam("num1") double num1,
        @RequestParam("num2") double num2,
        @RequestParam("operation") String operation,
        Model model) {
    double result = 0;

    switch (operation) {
        case "add":
            result = num1 + num2;
            break;
        case "subtract":
            result = num1 - num2;
            break;
        case "multiply":
            result = num1 * num2;
            break;
        case "divide":
            if (num2 != 0) {
                result = num1 / num2;
            } else {
                model.addAttribute( attributeName: "error",  attributeValue: "Деление на ноль невозможно");
                return "calculator";
            }
            break;
        default:
            model.addAttribute( attributeName: "error",  attributeValue: "Неизвестная операция");
            return "calculator";
    }

    model.addAttribute( attributeName: "result", result);
    return "ResultCalc";
}
```

5. Разметка для страницы калькулятора

```html
<h1>Калькулятор</h1>
<div class="calculator">
    <form action="/ResultCalc" method="post">
        <input type="text" name="num1" placeholder="" required>
        <select name="operation" required>
            <option value="add">+</option>
            <option value="subtract">-</option>
            <option value="multiply">*</option>
            <option value="divide">/</option>
        </select>
        <input type="text" name="num2" placeholder="" required>
        <div class="buttons">
            <button onclick="addToField()">1</button>
            <button onclick="addToField()">2</button>
            <button onclick="addToField()">3</button>
            <button onclick="addToField()">4</button>
            <button onclick="addToField()">5</button>
            <button onclick="addToField()">6</button>
            <button onclick="addToField()">7</button>
            <button onclick="addToField()">8</button>
            <button onclick="addToField()">9</button>
            <button onclick="addToField()">0</button>

        </div>
        <input type="submit" value="=">
    </form>
</div>
```

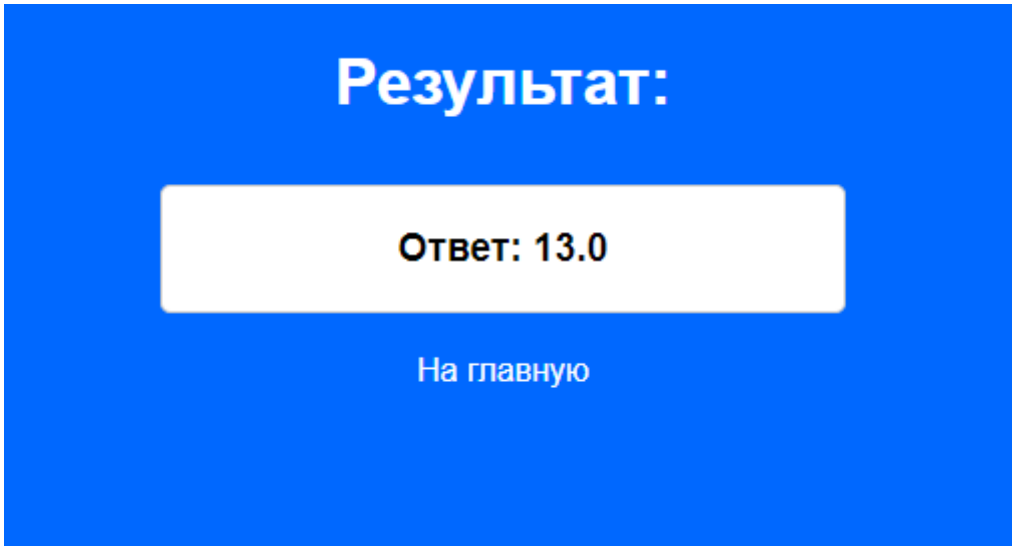
Итог



Страница результата

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Результат</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #0068ff;
            margin: 0;
            padding: 0;
            text-align: center;
        }

        h1 {

            color: #fff;
            padding: 10px;
            border-radius: 5px;
        }

        .result_text {
            background-color: #fff;
            border: 1px solid #ccc;
            padding: 20px;
            border-radius: 5px;
            margin: 20px auto;
            max-width: 300px;
        }
```

Итог

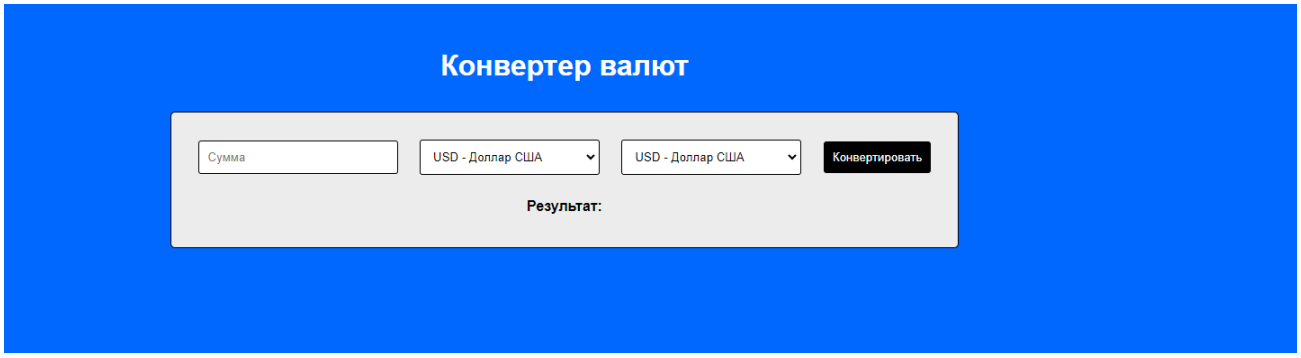


Калькулятор выполняет необходимые по условию функции также имеется перенаправление на главную.

6. Страница конвертера

```
</head>
<body>
<h1>Конвертер валют</h1>
<form action="/converter" method="post">
    <input type="text" name="amount" placeholder="Сумма" required>
    <select name="fromCurrency" required>
        <option value="USD">USD - Доллар США</option>
        <option value="EUR">EUR - Евро</option>
        <option value="TL">TL - Турецкая лира</option>
        <option value="RUB">RUB - Российский рубль</option>
    </select>
    <select name="toCurrency" required>
        <option value="USD">USD - Доллар США</option>
        <option value="EUR">EUR - Евро</option>
        <option value="TL">TL - Турецкая лира</option>
        <option value="RUB">RUB - Российский рубль</option>
    </select>
    <input type="submit" value="Конвертировать">
    <p><b>Результат:</b> <span th:text="${convertedAmount}"></span></p>
</form>
</body>
</html>
```

Итог



Пропишем теперь логику в контроллере

```
double USD = 96.41;
double EUR = 103.91;
double TL = 3.6070;
double convertedAmount = 0.0;

if ("RUB".equals(fromCurrency) && "USD".equals(toCurrency)) {
    convertedAmount = amount * USD;
} else if ("RUB".equals(fromCurrency) && "EUR".equals(toCurrency)) {
    convertedAmount = amount * EUR;
} else if ("RUB".equals(fromCurrency) && "TL".equals(toCurrency)) {
    convertedAmount = amount / TL;
} else if ("USD".equals(fromCurrency) && "EUR".equals(toCurrency)) {
    convertedAmount = amount * (USD / EUR);
} else if ("USD".equals(fromCurrency) && "TL".equals(toCurrency)) {
    convertedAmount = amount * (USD / TL);
} else if ("USD".equals(fromCurrency) && "RUB".equals(toCurrency)) {
    convertedAmount = amount * USD;
} else if ("EUR".equals(fromCurrency) && "TL".equals(toCurrency)) {
    convertedAmount = amount * (EUR / TL);
} else if ("EUR".equals(fromCurrency) && "USD".equals(toCurrency)) {
    convertedAmount = amount * (EUR / USD);
} else if ("EUR".equals(fromCurrency) && "RUB".equals(toCurrency)) {
    convertedAmount = amount * EUR;
} else if ("TL".equals(fromCurrency) && "USD".equals(toCurrency)) {
    convertedAmount = amount * (TL / USD);
} else if ("TL".equals(fromCurrency) && "EUR".equals(toCurrency)) {
    convertedAmount = amount * (TL / EUR);
} else if ("TL".equals(fromCurrency) && "RUB".equals(toCurrency)) {
    convertedAmount = amount * TL;
}
```

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.

```java
public class TicketsModel {

    3 usages
    private int _id;
    3 usages
    private String _name;
    3 usages
    private int _cost;
    3 usages
    private String _size;

    no usages
    public TicketsModel(){}

    public int getId() { return _id; }

    public void setId(int id) { this._id = id; }

    public String getName() { return _name; }

    public void setName(String name) { this._name = name; }

    2 usages
    public int getCost() { return _cost; }

    1 usage
    public void setCost(int cost) { this._cost = cost; }


    2 usages
    public String getSize() { return _size; }

    1 usage
    public void setSize(String size) { this._size = size; }

    4 usages
    public TicketsModel(int id, String name, int cost, String size){
        _id = id;
        _name = name;
        _cost = cost;
        _size = size;
    }
```

Рисунок 1. Геттеры и сеттеры.

2. Также создаем к каждой модели класс DAO.

```java
2 usages
@Component
public class TicketsDAO {

    5 usages
    private static int TICKETS_COUNT;
    8 usages
    private List<TicketsModel> tickets;
    {
        tickets = new ArrayList<>();

        tickets.add(new TicketsModel(++TICKETS_COUNT, name: "Астана", cost: 15000,  size: "L"));
        tickets.add(new TicketsModel(++TICKETS_COUNT, name: "Санкт-Петербург", cost: 2500,  size: "M"));
        tickets.add(new TicketsModel(++TICKETS_COUNT, name: "Казань", cost: 4500,  size: "M"));
        tickets.add(new TicketsModel(++TICKETS_COUNT, name: "Пермь", cost: 12000,  size: "XL"));
    }

    public List<TicketsModel> index() { return tickets; }

    public TicketsModel show(int id){
        return tickets.stream().filter(ticketsModel -> ticketsModel.getId() == id).findAny().orElse( other: null);
    }

    1 usage
    public void save(TicketsModel tickets) { tickets.setId(++TICKETS_COUNT); }

    public void update(int id, TicketsModel ticketsModel){
        TicketsModel updateTickets = show(id);
        updateTickets.setName(ticketsModel.getName());
        updateTickets.setCost(ticketsModel.getCost());
        updateTickets.setSize(ticketsModel.getSize());
    }

    public void delete(int id) { tickets.removeIf(t-> t.getId() == id); }

}
```

Рисунок 2. Класс DAO

3. Также создаем класс со всеми стилями.

```
.menu {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

.menu li {
    display: inline;
    margin-right: 10px;
}

.menu a.btn {
    display: inline-block;
    padding: 10px 20px;
    text-decoration: none;
    background-color: #007bff;
    color: #fff;
    border-radius: 4px;
    border: 1px solid #3333;
    transition: background-color 0.3s ease, transform 0.3s ease, box-shadow 0.3s ease;
}

.menu a.btn:hover {
    background-color: #0056b3;
    transform: scale(1.1);
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
}

/* Стили для мобильных устройств */
@media (max-width: 768px) {
    .menu {
        text-align: center;
    }
    .menu li {
        margin-right: 0;
        margin-bottom: 10px;
        display: block;
    }
    .menu a.btn {
        margin-right: 0;
    }
}
```

Рисунок 3. Стили.

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

```java
@Controller
@RequestMapping("/project")
public class ProjectController {

    7 usages
    private UserDAO _userDAO;
    7 usages
    private TicketsDAO _ticketsDAO;
    7 usages
    private PlanesDAO _planesDAO;
    7 usages
    private CarsDAO _carsDAO;
    7 usages
    private RobotsDAO _robotsDAO;

    @Autowired
    public ProjectController(UserDAO userDAO, TicketsDAO ticketsDAO , PlanesDAO planesDAO , CarsDAO carsDAO , RobotsDAO robotsDAO){
        _userDAO = userDAO;
        _ticketsDAO = ticketsDAO;
        _planesDAO = planesDAO;
        _carsDAO = carsDAO;
        _robotsDAO = robotsDAO;
    }

    //MAIN
    @GetMapping()
    public String Main(Model model) { return "project/main"; }
    //MAIN

    // Users begin
    @GetMapping("/users")
    public String Users(Model model){
        model.addAttribute( attributeName: "user", _userDAO.index());
        return "project/users";
    }

    @GetMapping("/{id}/user")
    public String showUser(@PathVariable("id") int id, Model model){
        // Вывод определенного пользователя
        model.addAttribute( attributeName: "user", _userDAO.show(id));
        return "project/showUser";
    }
```

Maven wrapper
File
.mvn/wrapper/maven-wrapper.properties...
Disable Maven wrapper in settings.

Рисунок 4. Главный контроллер.

```java
@GetMapping("/{id}/editUser")
public String editUser(Model model, @PathVariable("id") int id){
    model.addAttribute( attributeName: "user",_userDAO.show(id));
    return "project/editUser";
}

@PatchMapping("/{id}/updateUser")
public String updateUser(@ModelAttribute("user") UserModel userModel, @PathVariable("id") int id){
    _userDAO.update(id,userModel);
    return "redirect:/project";
}

@DeleteMapping("/{id}/deleteUser")
public String deleteUser(@PathVariable("id") int id){
    _userDAO.delete(id);
    return "redirect:/project";
}

// Users end
```

Рисунок 5. Запросы.

5. Также имеется верстка для каждой страницы.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Tickets</title>
    <link th:href="@{/index.css}" rel="stylesheet" />
</head>
<body>
<div th:each="tickets: ${tickets}">
    <a th:href="@{/project/{id}/tickets(id = ${tickets.getId()})}" th:text="${tickets.getName()}"></a>
</div>

<a href="/project/newTickets">create new tickets</a>
</body>
</html>
```

Рисунок 6. Основной элемент.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Edit Tickets</title>
    <link th:href="@{/edit.css}" rel="stylesheet" />
</head>
<body>

<form th:method="PATCH" th:action="@{/project/{id}/updateTickets(id = ${tickets.getId()})}" th:object="${tickets}">
    <div>
        <label for="name">Enter name</label>
        <input type="text" id="name" th:field="*{name}">
    </div>

    <div>
        <label for="size">Surname</label>
        <input type="text" id="size" th:field="*{size}">
    </div>

    <div>
        <label for="cost">Cost</label>
        <input type="number" id="cost" th:field="*{cost}">
    </div>

    <input type="submit" value="Update">
</form>

</body>
</html>
```

Рисунок 7. Редактирование.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>New Tickets</title>
    <link th:href="@{/new.css}" rel="stylesheet" />
</head>
<body>
<form th:method="POST" th:action="@{/project/createTickets}" th:object="${tickets}">
    <div>
        <label for="name">Enter name</label>
        <input type="text" id="name" th:field="*{name}">
    </div>

    <div>
        <label for="size">Size</label>
        <input type="text" id="size" th:field="*{size}">
    </div>

    <div>
        <label for="cost">Cost</label>
        <input type="number" id="cost" th:field="*{cost}">
    </div>

    <input type="submit" value="Create">
</form>
</body>
</html>
```

Рисунок 8. Добавление.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Tickets</title>
</head>
<body>
<link th:href="@{/show.css}" rel="stylesheet" />

<p th:text="${tickets.getId()}"></p>
<p th:text="${tickets.getName()}"></p>
<p th:text="${tickets.getSize()}"></p>
<p th:text="${tickets.getColor()}"></p>
<p th:text="${tickets.getCost().toString()}"></p>

<a th:href="@{/project/{id}/editTickets(id = ${tickets.getId()})}">Edit</a>

<form th:method="DELETE" th:action="@{/project/{id}/deleteTickets(id = ${tickets.getId()})}">
    <input type="submit" value="DELETE">
</form>

</body>
</html>
```

Рисунок 9. Отображение.

6. Главная страница и общий список всех файлов.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Главная</title>
    <link th:href="@{/my.css}" rel="stylesheet" />
</head>
<body>

<h1>Главная</h1>

<form th:method="GET" th:action="@{/project/users}">
    <input type="submit" class="ui-button" value="Пользователи">
</form>

<form th:method="GET" th:action="@{/project/tickets}">
    <input type="submit" value="Билеты">
</form>

<form th:method="GET" th:action="@{/project/planes}">
    <input type="submit" value="Самолеты">
</form>

<form th:method="GET" th:action="@{/project/cars}">
    <input type="submit" value="Машины">
</form>

<form th:method="GET" th:action="@{/project/robots}">
    <input type="submit" value="Роботы">
</form>
</body>
</html>
```

Рисунок 10. Главная.

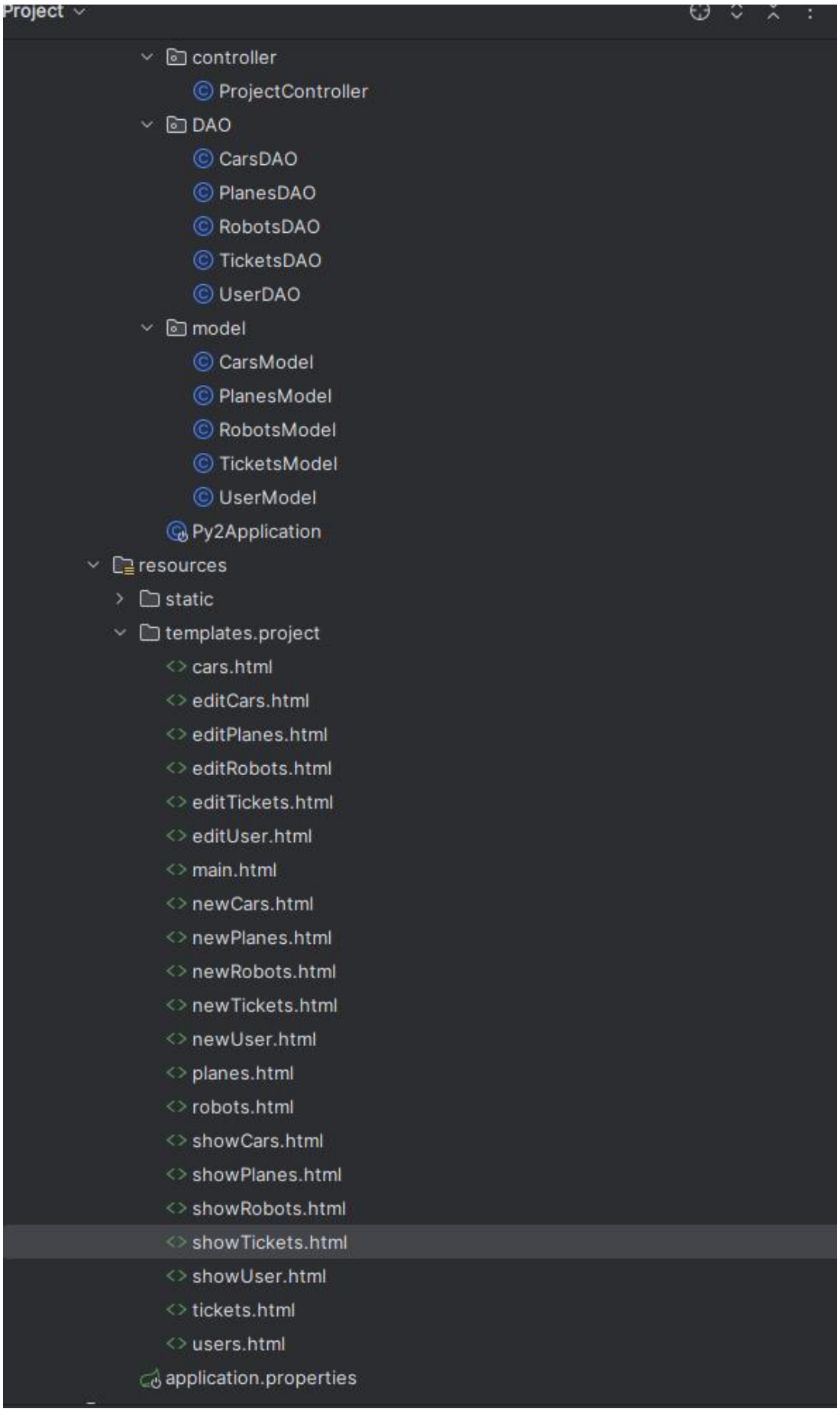
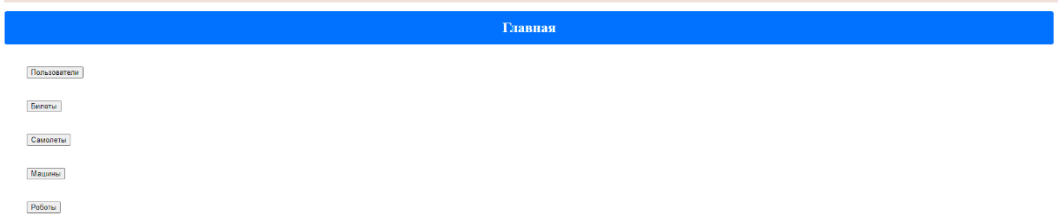
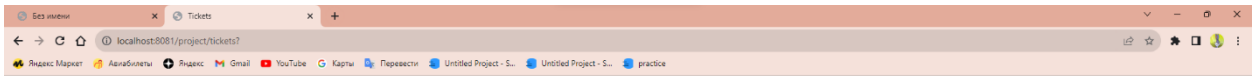


Рисунок 251. Список файлов.

Итог:



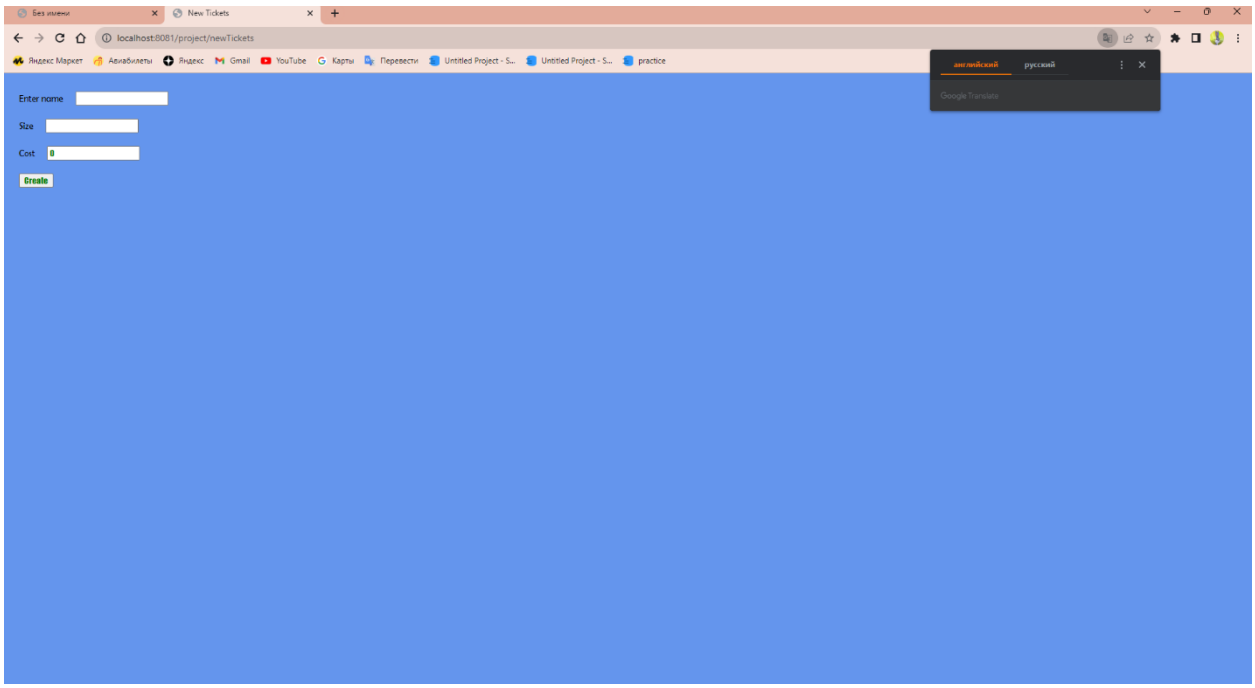


Астана

Санкт-Петербург

Казань

Пермь
новый билет



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.

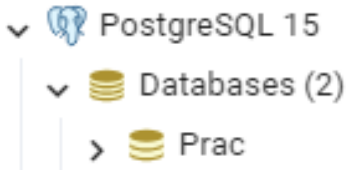


Рисунок 1. База данных.

```
users.html    application.properties ×    ProjectController.java    P
1   spring.datasource.url=jdbc:postgresql://localhost:5432/Prac
2   spring.datasource.username=postgres
3   spring.datasource.password=123
4
5   spring.jpa.hibernate.ddl-auto=update
6
7   spring.jpa.show-sql=true
8   spring.jpa.database=POSTGRESQL
9   spring.jpa.properties.hibernate.jdbc.lob.non_contextual_cred
10  server.port=8081
```

Рисунок 2. Подключение к проекту.

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

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

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>2.0.2</version>
        </dependency>
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>jakarta.xml.bind</groupId>
            <artifactId>jakarta.xml.bind-api</artifactId>
            <version>4.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.javassist</groupId>
            <artifactId>javassist</artifactId>
            <version>3.20.0-GA</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
        </dependency>

    </dependencies>
```

Рисунок 3. Дополнительные зависимости.

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

```
14 usages
@Entity
public class TicketsModel {

    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    no usages
    @NotBlank(message = "Name is required")
    @Size(max = 50)

    private String name;

    no usages
    @NotBlank(message = "Cost is required")
    @Size(max = 50)

    private int cost;

    3 usages
    @NotBlank(message = "Size is required")
    @Size(max = 50)

    private int _id;
    3 usages
    private String _name;
    3 usages
    private int _cost;
    3 usages
    private String _size;

    no usages
    public TicketsModel(){}

    public int getId() { return _id; }

    public void setId(int id) { this._id = id; }
```

Рисунок 4. Дополнительные условия.

4. Также создаются репозитории интерфейса.

```
import com.example.py2.model.CarsModel;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

no usages
public interface CarsRepos extends JpaRepository<CarsModel, Long> {
    no usages
    List<CarsModel> findByName(String name);
}
```

Рисунок 5. Интерфейс-репозиторий.

5. В контроллере все старые действия изменились.

```
@GetMapping()
public String Main(Model model) { return "project/main"; }



@GetMapping("/users")
public String Users(Model model){
    model.addAttribute("user", userRepos.findAll());
    return "project/users";
}

@GetMapping("/{id}/user")
public String showUser(@PathVariable("id") long id, Model model){

    model.addAttribute("user", userRepos.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid user Id:" + id)));
    return "project/showUser";
}


@GetMapping("/newUser")
public String newUser(@ModelAttribute("user") UserModel userModel){

    return "project/newUser";
}

@PostMapping("/createUser")
public String createUser(@Valid UserModel userModel, BindingResult result, Model model){
    if (result.hasErrors()) {
        return "project/newUser";
    }
    userRepos.save(userModel);
    model.addAttribute("user", userRepos.findAll());
    return "project/users";
}

@GetMapping("/{id}/editUser")
public String editUser(Model model, @PathVariable("id") long id){
    model.addAttribute("user", userRepos.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid user Id:" + id)));
    return "project/editUser";
ProjectController > newUser                                                                                    68:1
```

Рисунок 6. Изменённые действия.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Tickets</title>
    <link th:href="@{/index.css}" rel="stylesheet" />
</head>
<body>
<div th:each="tickets: ${tickets}">
    <a th:href="@{/project/{id}/tickets(id = ${tickets.getId()})}" th:text="${tickets.getName()}"></a>
</div>

<a href="/project/newTickets">create new tickets</a>

<form th:method="post" th:action="findTickets">
    <input type="text" name="filter">
    <button type="submit">Найти</button>
</form>

</body>
</html>
```

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.

```
3 usages
@OneToOne(optional = true, cascade = CascadeType.ALL)
@JoinColumn(name="passport_id")
private Passport passport;
```

Рисунок 1. связь со стороны User.

```
3 usages
@OneToOne(optional = true, mappedBy = "passport")
private UserModel owner;
```

Рисунок 2. связь со стороны Passport.

```java
@PostMapping("/createUser")
public String createUser(@Valid UserModel userModel, @RequestParam String number, BindingResult result, Model model){
    if (result.hasErrors()) {
        return "project/newUser";
    }
    userModel.setPassport(passportRepos.findByNumber(number));
    userRepos.save(userModel);
    model.addAttribute( attributeName: "user", userRepos.findAll());
    return "project/users";
}

@GetMapping("/{id}/editUser")
public String editUser(Model model, @PathVariable("id") long id){
    model.addAttribute( attributeName: "user", userRepos.findById(id).orElseThrow(() -> new IllegalArgumentException("Inva
    Iterable<Passport> passport = passportRepos.findAll();
    model.addAttribute( attributeName: "passport", passport);
    return "project/editUser";
}

@PatchMapping("/{id}/updateUser")
public String updateUser(@PathVariable("id") long id, @Valid UserModel userModel, @RequestParam String number, Binding
    if (result.hasErrors()) {
        userModel.setId(id);
        return "project/editUser";
    }
    userModel.setPassport(passportRepos.findByNumber(number));
    userRepos.save(userModel);
    model.addAttribute( attributeName: "user", userRepos.findAll());
    return "project/users";
}
```

Рисунок 3. Изменение контроллера под связь таблиц

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

```java
3 usages
@ManyToOne(
        optional = true,
        cascade = {CascadeType.ALL}
)
private CastleModel castleModel;
```

Рисунок 4. ManyToOne.

```java
3 usages
@OneToMany(
        mappedBy = "castleModel",
        fetch = FetchType.EAGER
)
private Collection<FeodalModel> rulers;
```

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.

```
3 usages
@ManyToMany
@JoinTable (name="book_library",
        joinColumns=@JoinColumn (name="book_id"),
        inverseJoinColumns=@JoinColumn(name="library_id"))
private List<LibraryModel> libraries;
```

Рисунок 6. Подключение со стороны таблицы книг.

```
@ManyToMany
@JoinTable(name="book_library",
        joinColumns=@JoinColumn (name="library_id"),
        inverseJoinColumns=@JoinColumn(name="book_id"))
private List<BookModel> books;
```

Рисунок 7. Подключение со стороны таблицы библиотек.

```
@GetMapping("/libraryBook")
private String LibraryBook(Model model){
    Iterable<LibraryModel> libraries = libraryRepos.findAll();
    model.addAttribute( attributeName: "libraries", libraries);
    Iterable<BookModel> books = bookRepos.findAll();
    model.addAttribute( attributeName: "books", books);
    return "project/LibraryBook";
}

@PostMapping("/libraryBook/add")
public String LibraryBookAdd(@RequestParam String library, @RequestParam String book, Model model)
{
    LibraryModel libraryModel = libraryRepos.findByName(library).get(0);
    BookModel bookModel = bookRepos.findByName(book).get(0);
    libraryModel.getBooks().add(bookModel);
    bookModel.getLibraries().add(libraryModel);
    libraryRepos.save(libraryModel);
    bookRepos.save(bookModel);
    return "project/main";
}
```

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.

```
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>5.5.2</version>
</dependency>
<dependency>
    <groupId>jakarta.validation</groupId>
    <artifactId>jakarta.validation-api</artifactId>
    <version>2.0.2</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.0-b06</version>
</dependency>
```

Рисунок 1. Зависимости для авторизации.

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

```java
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private DataSource dataSource;


    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.jdbcAuthentication().dataSource(dataSource).passwordEncoder(NoOpPasswordEncoder.getInstance()/*getPasswordEncoder()*/)
                .usersByUsernameQuery("select login, password, active from personal where login =?")
                .authoritiesByUsernameQuery("select u.login, ur.roles from personal u inner join user_role ur on u.id_personal = ur.user_id where u.login=?");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers("/project/login","/project/registration").permitAll() ExpressionInterceptUrlRegistry
                .anyRequest().authenticated().and().formLogin().loginPage("/project/login") FormLoginConfigurer<HttpSecurity>
                .defaultSuccessUrl("/project").permitAll().and().logout().permitAll() LogoutConfigurer<HttpSecurity>
                .and().csrf().disable().cors().disable();
    }
}
```

Рисунок 2. WebSecurityConfig.

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

```java
package com.example.py2.config;

import ...

@Configuration
public class MvcConfig implements WebMvcConfigurer {
    no usages
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController( urlPathOrPattern: "/project/login").setViewName("project/login");
    }
}
```

Рисунок 3. MvcConfig.

4. Делаем роли для аккаунтов.

```java
package com.example.py2.model;

import org.springframework.security.core.GrantedAuthority;
6 usages
public enum Role implements GrantedAuthority{
    1 usage
    USER, ADMIN, HELPER;
    @Override
    public String getAuthority() { return name(); }
}
```

Рисунок 4. Роль.

```java
@Entity
public class Personal {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id_personal;
    3 usages
    private String login;
    3 usages
    private String password;
    3 usages
    private boolean active;

    public Personal(){}

    3 usages
    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;
```

Рисунок 5. Personal.

5. Добавляем вызовы для проверки.

```java
@GetMapping("/registration")
private String RegView() { return "project/regis"; }
@PostMapping("/registration")
private String Reg(Personal user, Model model)
{
    Personal user_from_db = personalRepos.findByLogin(user.getLogin());
    if (user_from_db != null)
    {
        model.addAttribute( s: "message",  o: "Пользователь с таким логином уже существует");
        return "project/regis";
    }
    user.setActive(true);
    user.setRoles(Collections.singleton(Role.USER));
    personalRepos.save(user);
    return "redirect:/project/login";
}
```

Рисунок 6. Controller.

6. Создаются представления входа и добавления аккаунта.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
<head>
    <title>AUTH</title>
</head>
<body>
<div th:if="${param.error}">
    Неверное имя пользователя или пароль.
</div>
<form th:action="@{/project/login}" method="post">
    <div><label> Логин : <input type="text" name="username"/> </label></div>
    <div><label> Пароль : <input type="password" name="password"/> </label></div>
    <div><input type="submit" value="Авторизация"/></div>
</form>
<a href="/project/registration">Регистрация</a>
</body>
</html>
```

Рисунок 7. Login.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
<head>
    <title>AUTH</title>
</head>
<body>
<div th:if="${message}">
    Пользователь уже существует
</div>
<form th:action="@{/project/registration}" method="post">
    <div><label> Логин : <input type="text" name="login"/> </label></div>
    <div><label> Пароль : <input type="password" name="password"/> </label></div>
    <div><input type="submit" value="Регистрация"/></div>
</form>
</body>
</html>
```

Рисунок 8. Registration.

7. Проверка на работоспособность.

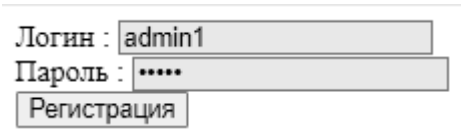


Рисунок 9. Регистрация.

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.

```
package com.example.py2.model;

import org.springframework.security.core.GrantedAuthority;
9 usages
public enum Role implements GrantedAuthority{
    1 usage
    USER, ADMIN, HELPER;
    @Override
    public String getAuthority() { return name(); }
}
```

Рисунок 1. Роль.

2. Делим контроллеры на трое.

```
package com.example.py2.controller;

import ...

@Controller
public class MainController {

    @Autowired
    private com.example.py2.repositories.PersonalRepos personalRepos;
    @Autowired
    private PasswordEncoder passwordEncoder;

    3 usages
    private final BookRepos bookRepos;
    2 usages
    private final CastleRepos castleRepos;
    2 usages
    private final DogRepos dogRepos;
    2 usages
    private final TshirtRepos tshirtRepos;
    2 usages
    private final UserRepos userRepos;
    2 usages
    private final PassportRepos passportRepos;
    2 usages
    private final FeodalRepos feodalRepos;
    3 usages
    private final LibraryRepos libraryRepos;
```

Рисунок 2. 1 Controller.

```
package com.example.py2.controller;

import ...

@Controller
@RequestMapping("/project")
@PreAuthorize("hasAnyAuthority('ADMIN')")
public class AdminController {

    /*
    private UserDAO _userDAO;
    private TshirtDAO _tshirtDAO;
    private DogDAO _dogDAO;
    private CastleDAO _castleDAO;
    private BookDAO _bookDAO;
     */

    @Autowired
    private PasswordEncoder passwordEncoder;
    1 usage
    private final BookRepos bookRepos;
    1 usage
    private final CastleRepos castleRepos;
    1 usage
    private final DogRepos dogRepos;
    1 usage
    private final TshirtRepos tshirtRepos;
    12 usages
    private final UserRepos userRepos;
```

Рисунок 3. 2 Controller.

```
package com.example.py2.controller;


import ...

@Controller
@RequestMapping("/project")
@PreAuthorize("hasAnyAuthority('ADMIN','HELPER')")
public class HelperController
{
    @Autowired
    private PasswordEncoder passwordEncoder;
    13 usages
    private final BookRepos bookRepos;
    16 usages
    private final CastleRepos castleRepos;
    11 usages
    private final DogRepos dogRepos;
    11 usages
    private final TshirtRepos tshirtRepos;
    1 usage
    private final UserRepos userRepos;
    1 usage
    private final PassportRepos passportRepos;
    12 usages
    private final FeodalRepos feodalRepos;
    13 usages
    private final LibraryRepos libraryRepos;
```

Рисунок 4. 3 Controller.

3. Создаем верстки страниц.

```
<head>
    <meta charset="UTF-8">
    <title>Personals</title>
    <link th:href="@{/index.css}" rel="stylesheet" />
</head>
<body>
<div th:each="personal: ${personal}">
    <a th:href="@{/project/{id}/personal(id = ${personal.getId()})}" th:text="${personal.getLogin()}"></a>
</div>

</body>
</html>
```

Рисунок 5. Список пользователей.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div class="container">
    <h1 th:text="'Пользователь : ' + ${user_object.login}"></h1>
    <div class="container">
        <h2 th:text="'Роли : ' + ${user_object.roles}"></h2>
        <p th:text="${user_object.active} == True ? 'Пользователь: Активный' : 'Пользователь: Неактивный'"></p>
    </div>
    <a th:href="@{/project/{id}/editPersonal(id = ${user_object.getId()})}"
       class="btn btn-warning">Редактировать</a>
    <a href="/project/personals" class="btn btn-success">Назад</a>
</div>
</body>
</html>
```

Рисунок 6. Подробная информация.

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div class="container">
    <form method="post" >
        <div>
            <input type="text" th:value="${user_object.login}" name="login">
        </div>
        <div th:each="role : ${roles}">
            <input type="checkbox" th:id="${role}"
                   th:name="${'roles[]'}" th:value="${role}"
                   th:checked="${#lists.contains(user_object.roles,role)}">

            <label th:text="${role}"></label>
        </div>
        <button class="btn btn-success" type="submit">Изменить пользователя</button>
    </form>

    <a class="btn btn-danger" th:href="'/project/' + ${user_object.id} + '/personal'">Назад</a>
</div>
</body>
</html>
```

Рисунок 7. Редактирование профиля.

5. Взаимодействие админа с версткой.

```java
    @GetMapping("/{id}/personal")
    public String detailView(@PathVariable long id, Model model)
    {
        model.addAttribute( s: "user_object",personalRepos.findById(id).orElseThrow());
        return "project/showPersonal";
    }
    @GetMapping("/{id}/editPersonal")
    public String updView(@PathVariable long id, Model model)
    {
        model.addAttribute( s: "user_object", personalRepos.findById(id).orElseThrow());
        model.addAttribute( s: "roles", Role.values());
        return "project/editPersonal";
    }
    @PostMapping("/{id}/editPersonal")
    public String update_user(@RequestParam String login,
                              @RequestParam(name="roles[]", required = false) String[] roles, @PathVariable Long id)
    {
        Personal user = personalRepos.findById(id).orElseThrow();
        user.setLogin(login);
        user.getRoles().clear();
        if(roles != null)
        {
            for(String role: roles)
            {
                user.getRoles().add(Role.valueOf(role));
            }
        }
        personalRepos.save(user);
        return "redirect:/project/personals";
    }
}
```

Рисунок 8. Контроллер Админа.

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