|  |
| --- |
| **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.02.01

Разработка и администрирование баз данных .

Специальность 09.02.07

Информационные системы и программирование .

Студент Константинов Даниил Юрьевич.

*(фамилия, имя, отчество)*

Группа П50-4-20

Руководитель по практической подготовке от техникума

Серяк Даниил Владимирович.

*(фамилия, имя, отчество)*

«\_\_» \_\_\_\_\_\_ 2023 года

Оглавление

[Тема индивидуального проекта 3](#_Toc150978263)

[Описание предметной области 4](#_Toc150978264)

[Диаграммы БД 5](#_Toc150978265)

[Словарь данных 6](#_Toc150978266)

[Скрипт базы данных 8](#_Toc150978267)

[Код программы 16](#_Toc150978268)

[Вывод 55](#_Toc150978269)

+

# Тема индивидуального проекта

В качестве темы для индивидуального проекта был выбран сервис по оказанию услуг.

# Описание предметной области

Предметная область сервиса по оказанию услуг включает в себя широкий спектр деятельности, направленной на удовлетворение потребностей клиентов. Это может охватывать различные отрасли и виды услуг. В данной ситуации сервис направлен на оказание услуг в строительной сфере.

# Диаграммы БД

На рисунке 1 представлена инфологическая схема БД.

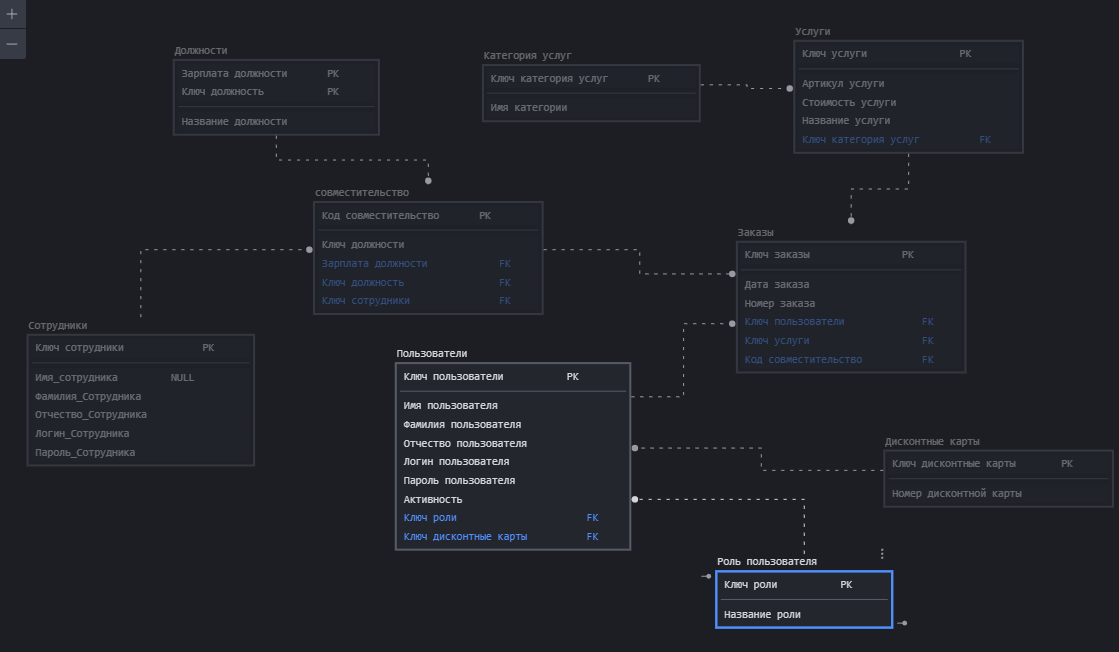


Рисунок 1 - Инфологическая схема

На рисунке 2 представлена даталогическая схема БД.

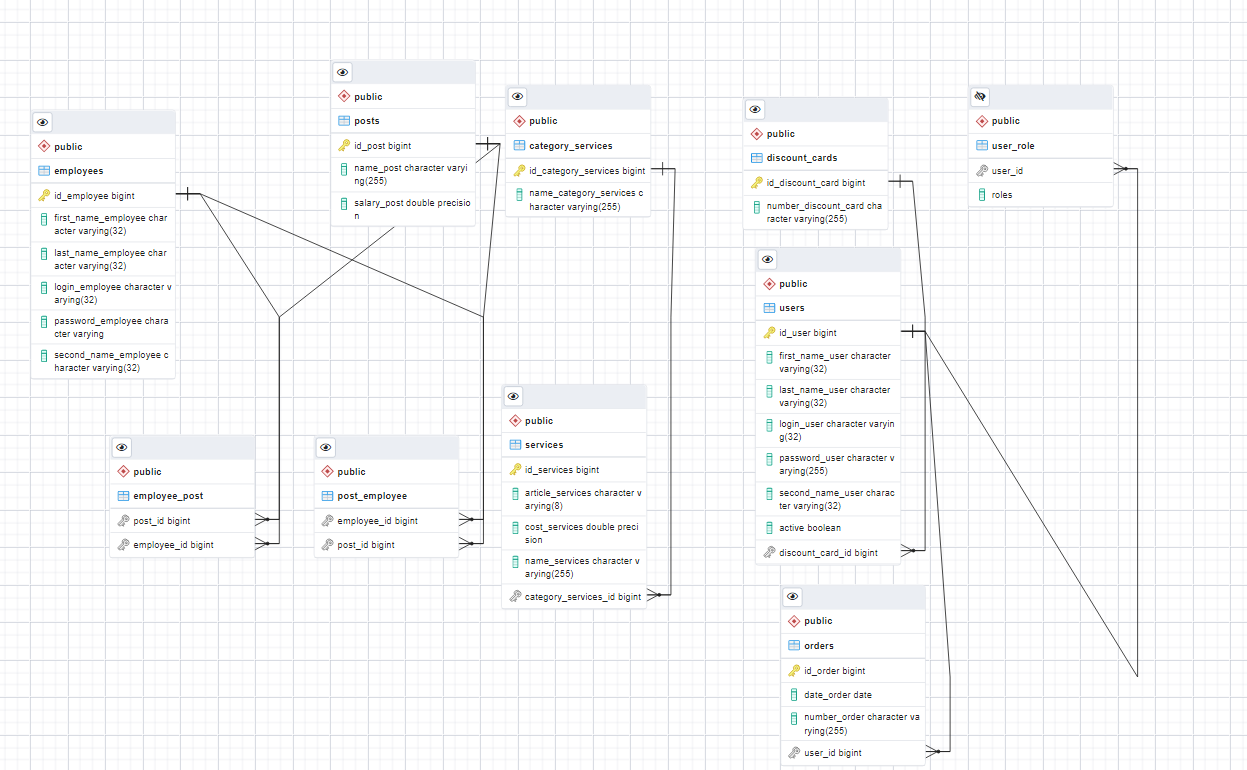


Рисунок 2 - Даталогическая схема

# Словарь данных

Таблица 1 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип данных | Ограничение | Описание |
| 1 | 2 | 3 | 4 | 5 |
| Discount\_Cards | | | | |
| 1 | Id\_Discount\_Cards | bigint | Not null | Первичный ключ таблицы «Дисконтные карты» |
| 2 | Number\_Discount\_Cards | Character varying(255) | Not null | Номер дисконтной карты |
| Posts | | | | |
| 1 | Id\_Posts | bigint | Not null | Первичный ключ таблицы «Должности» |
| 2 | Name\_Posts | Character varying(255) | Not null | Имя должности |
| 2 | Salary\_Post | Double precision | Not null | Зарплата у должности |
| Category\_Services | | | | |
| 1 | Id\_Category\_Services | bigint | Not null | Первичный ключ таблицы «Категории услуг» |
| 2 | Name\_ Category\_Services | Character varying(255) | Not null | Название категории услуг |
| Services | | | | |
| 1 | Id \_Services | bigint | Not null | Первичный ключ таблицы «Услуги» |
| 2 | Name\_Services | Character varying(255) | Not null | Название услуги |
| 3 | Cost\_Services | Double precision | Not null | Стоимость услуги |
| 4 | Article\_Services | Character varying(8) | Not null, =8 | Артикул услуги |
| Users | | | | |
| 1 | Id \_Users | bigint | Not null | Первичный ключ таблицы «пользователи» |
| 2 | First\_Name\_User | Character varying(32) | Not null, =32 | Имя пользователя |
| 3 | Second\_Name\_User | Character varying(32) | Not null, =32 | Фамилия пользователя |
| 4 | Middle\_Name\_User | Character varying(32) | Not null, =32 | Отчество пользователя |
| 5 | Password\_User | Character varying(255) | Not null | Пароль пользователя |
| 6 | Login\_User | Character varying(32) | Not null, =32 | Логин пользователя |
| 7 | Active | boolean | Not null | Активность |
| 8 | Discount\_Card\_Id | bigint | Not null | Внешний ключ на таблицу «Дисконтные карты» |
| Orders | | | | |
| 1 | Id \_Orders | bigint | Not null | Первичный ключ таблицы «Заказы» |
| 2 | Number\_Order | Character varying(255) | Not null | Номер заказа |
| 3 | Date\_Order | date | Not null | Дата заказа |
| 4 | user\_Id | bigint | Not null | Внешний ключ на таблицу «Пользователи» |
| Employee\_post | | | | |
| 1 | Post\_ID | bigint | Not null | Внешний ключ таблицы «Должности» |
| 2 | Employee\_ID | bigint | Not null | Внешний ключ таблицы «Сотрудники» |
| Post\_Employee | | | | |
| 1 | Employee\_ID | bigint | Not null | Внешний ключ таблицы «Сотрудники» |
| 2 | Post\_ID | bigint | Not null | Внешний ключ таблицы «Должности» |
| Employees | | | | |
| 1 | Id \_Employee | bigint | Not null | Первичный ключ таблицы «Сотрудники» |
| 2 | First\_Name\_ Employee | Character varying(32) | Not null, =32 | Имя сотрудника |
| 3 | Second\_Name\_ Employee | Character varying(32) | Not null, =32 | Фамилия сотрудника |
| 4 | Middle\_Name\_ Employee | Character varying(32) | Not null, =32 | Отчество сотрудника |
| 5 | Password\_ Employee | Character varying(255) | Not null | Пароль сотрудника |
| 6 | Login\_ Employee | Character varying(32) | Not null, =32 | Логин сотрудника |

# Скрипт базы данных

-- Table: public.category\_services

-- DROP TABLE IF EXISTS public.category\_services;

CREATE TABLE IF NOT EXISTS public.category\_services

(

id\_category\_services bigint NOT NULL,

name\_category\_services character varying(255) COLLATE pg\_catalog."default",

CONSTRAINT category\_services\_pkey PRIMARY KEY (id\_category\_services)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.category\_services

OWNER to postgres;

-- Table: public.discount\_cards

-- DROP TABLE IF EXISTS public.discount\_cards;

CREATE TABLE IF NOT EXISTS public.discount\_cards

(

id\_discount\_card bigint NOT NULL,

number\_discount\_card character varying(255) COLLATE pg\_catalog."default",

CONSTRAINT discount\_cards\_pkey PRIMARY KEY (id\_discount\_card)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.discount\_cards

OWNER to postgres;

-- Table: public.employee\_post

-- DROP TABLE IF EXISTS public.employee\_post;

CREATE TABLE IF NOT EXISTS public.employee\_post

(

post\_id bigint NOT NULL,

employee\_id bigint NOT NULL,

CONSTRAINT fk86mb2vk1aj5w4kwhc63otqsv4 FOREIGN KEY (employee\_id)

REFERENCES public.employees (id\_employee) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fkpjcs31jl9fo3tlcwcmsx3ifmf FOREIGN KEY (post\_id)

REFERENCES public.posts (id\_post) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.employee\_post

OWNER to postgres;

-- Table: public.employees

-- DROP TABLE IF EXISTS public.employees;

CREATE TABLE IF NOT EXISTS public.employees

(

id\_employee bigint NOT NULL,

first\_name\_employee character varying(32) COLLATE pg\_catalog."default",

last\_name\_employee character varying(32) COLLATE pg\_catalog."default",

login\_employee character varying(32) COLLATE pg\_catalog."default",

password\_employee character varying COLLATE pg\_catalog."default",

second\_name\_employee character varying(32) COLLATE pg\_catalog."default",

CONSTRAINT employees\_pkey PRIMARY KEY (id\_employee)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.employees

OWNER to postgres;

-- Table: public.executors

-- DROP TABLE IF EXISTS public.executors;

CREATE TABLE IF NOT EXISTS public.executors

(

id\_executor bigint NOT NULL,

inn\_executor character varying(32) COLLATE pg\_catalog."default",

name\_executor character varying(32) COLLATE pg\_catalog."default",

CONSTRAINT executors\_pkey PRIMARY KEY (id\_executor)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.executors

OWNER to postgres;

-- Table: public.orders

-- DROP TABLE IF EXISTS public.orders;

CREATE TABLE IF NOT EXISTS public.orders

(

id\_order bigint NOT NULL,

date\_order date,

number\_order character varying(255) COLLATE pg\_catalog."default",

user\_id bigint,

CONSTRAINT orders\_pkey PRIMARY KEY (id\_order),

CONSTRAINT fk32ql8ubntj5uh44ph9659tiih FOREIGN KEY (user\_id)

REFERENCES public.users (id\_user) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.orders

OWNER to postgres;

-- Table: public.post\_employee

-- DROP TABLE IF EXISTS public.post\_employee;

CREATE TABLE IF NOT EXISTS public.post\_employee

(

employee\_id bigint NOT NULL,

post\_id bigint NOT NULL,

CONSTRAINT fkd3o74cqf7fjkfoqi3yk7k30g0 FOREIGN KEY (employee\_id)

REFERENCES public.employees (id\_employee) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fkltwgf8d8iqffor1r4bvi78qo0 FOREIGN KEY (post\_id)

REFERENCES public.posts (id\_post) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.post\_employee

OWNER to postgres;

-- Table: public.posts

-- DROP TABLE IF EXISTS public.posts;

CREATE TABLE IF NOT EXISTS public.posts

(

id\_post bigint NOT NULL,

name\_post character varying(255) COLLATE pg\_catalog."default",

salary\_post double precision NOT NULL,

CONSTRAINT posts\_pkey PRIMARY KEY (id\_post)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.posts

OWNER to postgres;

-- Table: public.services

-- DROP TABLE IF EXISTS public.services;

CREATE TABLE IF NOT EXISTS public.services

(

id\_services bigint NOT NULL,

article\_services character varying(8) COLLATE pg\_catalog."default",

cost\_services double precision NOT NULL,

name\_services character varying(255) COLLATE pg\_catalog."default",

category\_services\_id bigint,

CONSTRAINT services\_pkey PRIMARY KEY (id\_services),

CONSTRAINT fk4iofxnqnme6uo5qqa67cwn3sq FOREIGN KEY (category\_services\_id)

REFERENCES public.category\_services (id\_category\_services) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.services

OWNER to postgres;

-- Table: public.user\_role

-- DROP TABLE IF EXISTS public.user\_role;

CREATE TABLE IF NOT EXISTS public.user\_role

(

user\_id bigint NOT NULL,

roles character varying(255) COLLATE pg\_catalog."default",

CONSTRAINT fkj345gk1bovqvfame88rcx7yyx FOREIGN KEY (user\_id)

REFERENCES public.users (id\_user) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.user\_role

OWNER to postgres;

-- Table: public.users

-- DROP TABLE IF EXISTS public.users;

CREATE TABLE IF NOT EXISTS public.users

(

id\_user bigint NOT NULL,

first\_name\_user character varying(32) COLLATE pg\_catalog."default",

last\_name\_user character varying(32) COLLATE pg\_catalog."default",

login\_user character varying(32) COLLATE pg\_catalog."default",

password\_user character varying(255) COLLATE pg\_catalog."default",

second\_name\_user character varying(32) COLLATE pg\_catalog."default",

active boolean NOT NULL,

discount\_card\_id bigint,

CONSTRAINT users\_pkey PRIMARY KEY (id\_user),

CONSTRAINT fkol1y22q7cp8rm23xr2rdnudgv FOREIGN KEY (discount\_card\_id)

REFERENCES public.discount\_cards (id\_discount\_card) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.users

OWNER to postgres;

# Код программы

Код MvcConfig.java:

package com.example.prakt3.config;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

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

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

@Configuration

@ComponentScan(basePackages = {"com.example.prakt3"})

public class MvcConfig implements WebMvcConfigurer {

public void addViewControllers(ViewControllerRegistry registry) {

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

}

}

Код WebSecurityConfig. java:

package com.example.prakt3.config;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

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

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

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

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

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

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

import javax.sql.DataSource;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private DataSource dataSource;

@Autowired

private PasswordEncoder passwordEncoder;

@Bean

public PasswordEncoder getPasswordEncoder(){

return new BCryptPasswordEncoder(8);

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

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

.usersByUsernameQuery("select Login\_User, Password\_User, active from users where Login\_User =?")

.authoritiesByUsernameQuery("select u.Login\_User, ur.roles from users u inner join user\_role ur on u.id\_user = ur.user\_id where u.Login\_User=?");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/login","/regis").permitAll()

.anyRequest().authenticated().and().formLogin().loginPage("/login")

.defaultSuccessUrl("/test").permitAll().and().logout().permitAll()

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

}

}

Код CategoryServicesController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.categoryServices;

import com.example.prakt3.models.executors;

import com.example.prakt3.repositories.categoryServicesRepository;

import com.example.prakt3.repositories.executorRepository;

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 javax.validation.Valid;

@Controller

@RequestMapping("/admin/employee")

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

public class CategoryServicesController {

private final categoryServicesRepository repositoryCategoryServices;

public CategoryServicesController(categoryServicesRepository repositoryCategoryServices) {

this.repositoryCategoryServices = repositoryCategoryServices;

}

@GetMapping("/categoryServices")

public String CategoryServices(@ModelAttribute("categoryServices") categoryServices categoryServices, Model model){

Iterable<categoryServices> categoryServices1 = repositoryCategoryServices.findAll();

model.addAttribute("categoryServices2", categoryServices1);

return "categoryServices";

}

@PostMapping(value = "/createNewCategoryServices")

public String createCategoryServices(@Valid categoryServices categoryServices, BindingResult result, Model model){

if(result.hasErrors()){

Iterable<categoryServices> categoryServices1 = repositoryCategoryServices.findAll();

model.addAttribute("categoryServices2", categoryServices1);

return "redirect:/admin/employee/categoryServices";

}

repositoryCategoryServices.save(categoryServices);

model.addAttribute("categoryServices", repositoryCategoryServices.findAll());

return "redirect:/admin/employee/categoryServices";

}

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

public String editCategoryServices(Model model, @PathVariable("id") long id) {

categoryServices categoryServices = repositoryCategoryServices.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid category ID:" + id));

model.addAttribute("categoryServices", categoryServices);

return "editCategoryServices";

}

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

public String updateCategoryServices(@PathVariable("id") long id,@ModelAttribute("categoryServices") @Valid categoryServices categoryServices, BindingResult result, Model model){

if (result.hasErrors()) {

return "editCategoryServices";

}

categoryServices.setID\_categoryServices(id);

repositoryCategoryServices.save(categoryServices);

model.addAttribute("categoryServices2", repositoryCategoryServices.findAll());

return "redirect:/admin/employee/categoryServices";

}

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

public String deleteCategoryServices(@PathVariable("id") long id, @ModelAttribute("categoryServices")categoryServices categoryServices, Model model){

categoryServices delCategoryServices = repositoryCategoryServices.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid category services ID: "+ id));

repositoryCategoryServices.delete(delCategoryServices);

model.addAttribute("categoryServices2", repositoryCategoryServices.findAll());

return "categoryServices";

}

}

Код DiscountCardController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.discountCards;

import com.example.prakt3.repositories.discountCardRepository;

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

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 javax.validation.Valid;

@Controller

@RequestMapping("/user/admin")

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

public class DiscountCardController {

private final discountCardRepository repositoryDiscountCardRepository;

public DiscountCardController(discountCardRepository repositoryDiscountCardRepository) {

this.repositoryDiscountCardRepository = repositoryDiscountCardRepository;

}

@GetMapping("/discountCards")

public String discountCards(@ModelAttribute("discountCard") discountCards discountCard, Model model){

Iterable<discountCards> discountCards = repositoryDiscountCardRepository.findAll();

model.addAttribute("discountCards", discountCards);

return "discountCards";

}

@PostMapping(value = "/createNewDiscountCard")

public String createDiscountCard(@Valid discountCards discountCard, BindingResult result, Model model){

if(result.hasErrors()){

Iterable<discountCards> discountCards = repositoryDiscountCardRepository.findAll();

model.addAttribute("discountCards", discountCards);

return "redirect:/user/admin/discountCards";

}

repositoryDiscountCardRepository.save(discountCard);

model.addAttribute("discountCards", repositoryDiscountCardRepository.findAll());

return "redirect:/user/admin/discountCards";

}

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

public String editDiscountCard(Model model, @PathVariable("id") long id) {

discountCards discountCard = repositoryDiscountCardRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid card ID:" + id));

model.addAttribute("discountCard", discountCard);

return "editDiscountCard";

}

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

public String updateDiscountCard(@PathVariable("id") long id,@ModelAttribute("discountCard") @Valid discountCards discountCard, BindingResult result, Model model){

if (result.hasErrors()) {

return "redirect:/user/admin/discountCards";

}

discountCard.setID\_discountCard(id);

repositoryDiscountCardRepository.save(discountCard);

model.addAttribute("discountCards", repositoryDiscountCardRepository.findAll());

return "redirect:/user/admin/discountCards";

}

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

public String deleteDiscountCard(@PathVariable("id") long id, @ModelAttribute("discountCard")discountCards discountCard, Model model){

discountCards delDiscountCard = repositoryDiscountCardRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid discount card ID: "+ id));

repositoryDiscountCardRepository.delete(delDiscountCard);

model.addAttribute("discountCards", repositoryDiscountCardRepository.findAll());

return "discountCards";

}

}

Код EmployeePostController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.employees;

import com.example.prakt3.repositories.employeesRepository;

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

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

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.\*;

import javax.validation.Valid;

@Controller

@RequestMapping("/admin/employee")

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

public class EmployeeController {

private final employeesRepository repository;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

public EmployeeController(employeesRepository repository){

this.repository = repository;

}

@GetMapping("/employees")

public String employees(@ModelAttribute("employee") employees employee, Model model){

model.addAttribute("employees", repository.findAll());

return "employees";

}

@PostMapping(value = "/createNewEmployees")

public String createEmployee(@Valid employees employee, BindingResult result, Model model){

if(result.hasErrors()){

return "redirect:/admin/employee/employees";

}

employee.setPassword\_Employee(passwordEncoder.encode(employee.getPassword\_Employee()));

repository.save(employee);

model.addAttribute("employees", repository.findAll());

return "redirect:/admin/employee/employees";

}

@GetMapping("/admin/employees/{id}")

public String editEmployee(Model model, @PathVariable("id") long id) {

employees employee = repository.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid employee ID:" + id));

model.addAttribute("employee", employee);

return "editEmployees";

}

@PostMapping("/admin/employees/{id}")

public String updateEmployee(@PathVariable("id") long id,@ModelAttribute("employee") @Valid employees employee, BindingResult result, Model model){

if (result.hasErrors()) {

return "redirect:/admin/employee/employees";

}

employee.setID\_Employee(id);

employee.setPassword\_Employee(passwordEncoder.encode(employee.getPassword\_Employee()));

repository.save(employee);

model.addAttribute("employees", repository.findAll());

return "redirect:/admin/employee/employees";

}

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

public String deleteEmployee(@PathVariable("id") long id, @ModelAttribute("employee")employees employee, Model model){

employees newemployee = repository.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid employee ID: "+ id));

repository.delete(newemployee);

model.addAttribute("employees", repository.findAll());

return "redirect:/admin/employee/employees";

}

}

Код ExecutorController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.categoryServices;

import com.example.prakt3.models.executors;

import com.example.prakt3.models.services;

import com.example.prakt3.repositories.categoryServicesRepository;

import com.example.prakt3.repositories.executorRepository;

import com.example.prakt3.repositories.servicesRepository;

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 javax.validation.Valid;

@Controller

@RequestMapping("/admin/employee")

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

public class ExecutorController {

private final executorRepository repositoryExecutorRepository;

public ExecutorController(executorRepository repositoryExecutorRepository) {

this.repositoryExecutorRepository = repositoryExecutorRepository;

}

@GetMapping("/executors")

public String executors(@ModelAttribute("executor") executors executor, Model model){

model.addAttribute("executors", repositoryExecutorRepository.findAll());

return "executors";

}

@PostMapping(value = "/createNewExecutor")

public String createExecutor(@Valid executors executor, BindingResult result, Model model){

if(result.hasErrors()){

return "redirect:/admin/employee/executors";

}

repositoryExecutorRepository.save(executor);

model.addAttribute("executors", repositoryExecutorRepository.findAll());

return "redirect:/admin/employee/executors";

}

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

public String editExecutor(Model model, @PathVariable("id") long id) {

executors executor = repositoryExecutorRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid executor ID:" + id));

model.addAttribute("executor", executor);

return "editExecutors";

}

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

public String updateExecutor(@PathVariable("id") long id, @ModelAttribute("executor") @Valid executors executor, BindingResult result, Model model){

if (result.hasErrors()) {

return "redirect:/admin/employee/executors";

}

executor.setID\_Executor(id);

repositoryExecutorRepository.save(executor);

model.addAttribute("executors", repositoryExecutorRepository.findAll());

return "redirect:/admin/employee/executors";

}

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

public String deleteExecutor(@PathVariable("id") long id,@ModelAttribute("executor")executors executor, Model model){

executors newexecutor = repositoryExecutorRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid executor ID: "+ id));

repositoryExecutorRepository.delete(newexecutor);

model.addAttribute("executors", repositoryExecutorRepository.findAll());

return "executors";

}

}

Код OrderController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.orders;

import com.example.prakt3.models.users;

import com.example.prakt3.repositories.ordersRepository;

import com.example.prakt3.repositories.usersRepository;

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

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 javax.validation.Valid;

@Controller

@RequestMapping("/user/employee/admin")

@PreAuthorize("hasAnyAuthority('USER', 'EMPLOYEE','ADMIN')")

public class OrderController {

private final ordersRepository repositoryOrders;

private final usersRepository repositoryUsers;

@Autowired

public OrderController(ordersRepository repositoryOrders, usersRepository repositoryUsers){

this.repositoryOrders = repositoryOrders;

this.repositoryUsers = repositoryUsers;

}

@GetMapping("/orders")

public String orders(@ModelAttribute("order") orders order, Model model){

Iterable<users> users = repositoryUsers.findAll();

model.addAttribute("users", users);

model.addAttribute("orders", repositoryOrders.findAll());

return "orders";

}

@PostMapping(value = "/createNewOrder")

public String createOrder(@Valid orders order, BindingResult result, @RequestParam String login, Model model){

if(result.hasErrors()){

Iterable<users> users = repositoryUsers.findAll();

model.addAttribute("users", users);

return "redirect:/user/employee/admin/orders";

}

users getUser = new users();

for(users element: repositoryUsers.findAll()){

if(element.getLogin\_User().equals(login)){

getUser = element;

}

}

order.setUser(getUser);

repositoryOrders.save(order);

model.addAttribute("orders", repositoryOrders.findAll());

return "redirect:/user/employee/admin/orders";

}

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

public String editOrder(Model model, @PathVariable("id") long id) {

orders order = repositoryOrders.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid order ID:" + id));

model.addAttribute("order", order);

Iterable<users> users = repositoryUsers.findAll();

model.addAttribute("users", users);

return "editOrders";

}

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

public String updateOrder(@PathVariable("id") long id,@ModelAttribute("order") @Valid orders order, @RequestParam String login, BindingResult result, Model model){

if (result.hasErrors()) {

Iterable<users> users = repositoryUsers.findAll();

model.addAttribute("users", users);

return "redirect:/user/employee/admin/orders";

}

users getUser = new users();

for(users element: repositoryUsers.findAll()){

if(element.getLogin\_User().equals(login)){

getUser = element;

}

}

order.setID\_Order(id);

order.setUser(getUser);

repositoryOrders.save(order);

model.addAttribute("orders", repositoryOrders.findAll());

return "redirect:/user/employee/admin/orders";

}

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

public String deleteOrder(@PathVariable("id") long id,@ModelAttribute("order")orders order, Model model){

orders neworder = repositoryOrders.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid order ID: "+ id));

repositoryOrders.delete(neworder);

model.addAttribute("orders", repositoryOrders.findAll());

return "orders";

}

}

Код MainController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.\*;

import com.example.prakt3.repositories.\*;

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

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.\*;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.Collections;

@Controller()

public class MainController {

private final usersRepository repositoryUsers;

private final discountCardRepository repositoryDiscount;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

public MainController(usersRepository usersRepository, discountCardRepository repositoryDiscount){

this.repositoryUsers = usersRepository;

this.repositoryDiscount = repositoryDiscount;

}

@GetMapping("/regis")

private String RegView()

{

return "regis";

}

@PostMapping("/regis")

private String Reg(users user, Model model)

{

discountCards discountCards = new discountCards();

discountCards.setOwner\_DiscountCard(user);

discountCards.setNumber\_discountCard(String.format("%08d", user.getID\_User()));

repositoryDiscount.save(discountCards);

user.setActive(true);

user.setRoles(Collections.singleton(roleEnum.USER));

user.setPassword\_User(passwordEncoder.encode(user.getPassword\_User()));

user.setDiscountCard(discountCards);

repositoryUsers.save(user);

return "redirect:/login";

}

@GetMapping("/test")

public String test(){return "test";

}

Код PostController. java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.posts;

import com.example.prakt3.repositories.postsRepository;

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

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 javax.validation.Valid;

@Controller

@RequestMapping("/admin/employee")

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

public class PostController {

private final postsRepository repositoryPosts;

@Autowired

public PostController(postsRepository repositoryPosts){

this.repositoryPosts = repositoryPosts;

}

@GetMapping("/posts")

public String posts(@ModelAttribute("post") posts post, Model model){

model.addAttribute("posts", repositoryPosts.findAll());

return "posts";

}

@PostMapping(value = "/createNewPost")

public String createPost(@Valid posts post, BindingResult result, Model model){

if(result.hasErrors()){

return "redirect:/posts";

}

repositoryPosts.save(post);

model.addAttribute("posts", repositoryPosts.findAll());

return "redirect:/admin/employee/posts";

}

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

public String editPost(Model model, @PathVariable("id") long id) {

posts post = repositoryPosts.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid post ID:" + id));

model.addAttribute("post", post);

return "editPosts";

}

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

public String updatePost(@PathVariable("id") long id,@ModelAttribute("post") @Valid posts post, BindingResult result, Model model){

if (result.hasErrors()) {

return "redirect:/admin/employee/posts";

}

post.setID\_Post(id);

repositoryPosts.save(post);

model.addAttribute("posts", repositoryPosts.findAll());

return "redirect:/admin/employee/posts";

}

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

public String deletePost(@PathVariable("id") long id,@ModelAttribute("post")posts post, Model model){

posts newpost = repositoryPosts.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid post ID: "+ id));

repositoryPosts.delete(newpost);

model.addAttribute("posts", repositoryPosts.findAll());

return "redirect:/admin/employee/posts";

}

}

Код RoleChangeController.java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.roleEnum;

import com.example.prakt3.models.users;

import com.example.prakt3.repositories.usersRepository;

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

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

@Controller

@RequestMapping("/admin/employee")

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

public class RoleChangeController {

private final usersRepository userRepository;

@Autowired

public RoleChangeController(com.example.prakt3.repositories.usersRepository userRepository) {

this.userRepository = userRepository;

}

@GetMapping("/updaterole")

public String updView(Model model)

{

model.addAttribute("users",userRepository.findAll());

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

return "roleChange";

}

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

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

{

model.addAttribute("user", userRepository.findById(id).orElseThrow());

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

return "updateRole";

}

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

public String update\_user(@RequestParam(name="roles[]", required = false) String[] roles,

@PathVariable Long id)

{

users user = userRepository.findById(id).orElseThrow();

user.getRoles().clear();

if(roles != null)

{

for(String role: roles)

{

user.getRoles().add(roleEnum.valueOf(role));

}

}

userRepository.save(user);

return "redirect:/admin/employee/updaterole";

}

}

Код ServicesController.java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.categoryServices;

import com.example.prakt3.models.orders;

import com.example.prakt3.models.services;

import com.example.prakt3.models.users;

import com.example.prakt3.repositories.categoryServicesRepository;

import com.example.prakt3.repositories.servicesRepository;

import org.hibernate.validator.internal.util.logging.Log;

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

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 javax.validation.Valid;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.Arrays;

import java.util.List;

@Controller

@RequestMapping("/admin/employee")

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

public class ServicesController {

private final servicesRepository repositoryServices;

private final categoryServicesRepository repositoryCategoryServices;

public ServicesController(servicesRepository repositoryServices, categoryServicesRepository repositoryCategoryServices) {

this.repositoryServices = repositoryServices;

this.repositoryCategoryServices = repositoryCategoryServices;

}

/\* @GetMapping("/services")

public String services(@RequestParam(name= "nameServices", defaultValue = "") String nameServices, @ModelAttribute("service") services services, Model model){

services getService = new services();

for(services item: repositoryServices.findAll()){

if(item.getName\_Services().equals(nameServices)){

getService = item;

}

}

Iterable<categoryServices> categoryServices = repositoryCategoryServices.findAll();

model.addAttribute("getService", getService);

model.addAttribute("categoryServices", categoryServices);

model.addAttribute("nameServices", nameServices);

model.addAttribute("services", repositoryServices.findAll());

return "services";

}\*/

@GetMapping("/services")

public String services(@ModelAttribute("service") services service, Model model){

Iterable<services> services = repositoryServices.findAll();

model.addAttribute("services", services);

model.addAttribute("categoryServices", repositoryCategoryServices.findAll());

return "services";

}

@PostMapping(value = "/createNewService")

public String createServices(@Valid services service, BindingResult result, @RequestParam String number, Model model){

if(result.hasErrors()){

Iterable<categoryServices> categoryServices = repositoryCategoryServices.findAll();

model.addAttribute("categoryServices", categoryServices);

return "redirect:/admin/employee/services";

}

categoryServices getCategoryService = new categoryServices();

for(categoryServices item: repositoryCategoryServices.findAll()){

if(item.getName\_categoryServices().equals(number)){

getCategoryService = item;

}

}

service.setCategoryServices(getCategoryService);

String text = "getCategoryService: "+ getCategoryService.getName\_categoryServices() + "\nnumber:" + number;

repositoryServices.save(service);

model.addAttribute("services", repositoryServices.findAll());

return "redirect:/admin/employee/services";

}

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

public String editServices(Model model, @PathVariable("id") long id) {

services service = repositoryServices.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid service ID:" + id));

Iterable<categoryServices> categoryServices = repositoryCategoryServices.findAll();

model.addAttribute("categoryServices", categoryServices);

model.addAttribute("service", service);

return "editServices";

}

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

public String updateServices(@PathVariable("id") long id, @ModelAttribute("service") @Valid services service, @RequestParam String number, BindingResult result, Model model){

if (result.hasErrors()) {

Iterable<categoryServices> categoryServices = repositoryCategoryServices.findAll();

model.addAttribute("categoryServices", categoryServices);

return "editServices";

}

service.setID\_Services(id);

categoryServices getCategoryService = new categoryServices();

for(categoryServices element: repositoryCategoryServices.findAll()){

if(element.getName\_categoryServices().equals(number)){

getCategoryService = element;

}

}

service.setCategoryServices(getCategoryService);

repositoryServices.save(service);

model.addAttribute("services", repositoryServices.findAll());

return "redirect:/admin/employee/services";

}

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

public String deleteServices(@PathVariable("id") long id,@RequestParam(name= "nameServices", defaultValue = "") String nameServices, @ModelAttribute("service")services service, Model model){

services delservice = repositoryServices.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid service ID: "+ id));

repositoryServices.delete(delservice);

services getService = new services();

model.addAttribute("services", repositoryServices.findAll());

model.addAttribute("getService", getService);

model.addAttribute("nameServices", nameServices);

return "services";

}

}

Код UserController.java:

package com.example.prakt3.controllers;

import com.example.prakt3.models.\*;

import com.example.prakt3.repositories.discountCardRepository;

import com.example.prakt3.repositories.usersRepository;

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

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

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.\*;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.validation.Valid;

import java.util.Collections;

@Controller

@RequestMapping("/admin/user")

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

public class UserController {

private final usersRepository repositoryUsers;

private final discountCardRepository discountCardRepository;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

public UserController(usersRepository repositoryUsers, discountCardRepository discountCardRepository){

this.repositoryUsers = repositoryUsers;

this.discountCardRepository = discountCardRepository;

}

@GetMapping("/users")

public String users(@RequestParam(name= "login", defaultValue = "") String login, @ModelAttribute("user") users user, Model model){

users getUser = new users();

for(users item: repositoryUsers.findAll()){

if(item.getLogin\_User().equals(login)){

getUser = item;

}

}

Iterable<discountCards> discountCards = discountCardRepository.findAll();

model.addAttribute("discountCards", discountCards);

model.addAttribute("getUser", getUser);

model.addAttribute("login", login);

model.addAttribute("users", repositoryUsers.findAll());

model.addAttribute("newUser", new users());

return "index";

}

@PostMapping(value = "/createNewUser")

public String createEmployee(@Valid users user, BindingResult result, Model model){

if(result.hasErrors()){

return "redirect:/index";

}

user.setPassword\_User(passwordEncoder.encode(user.getPassword\_User()));

repositoryUsers.save(user);

model.addAttribute("users", repositoryUsers.findAll());

model.addAttribute("users", repositoryUsers.findAll());

return "redirect:/index";

}

@PostMapping("/searchUser")

public String searchUser(@RequestParam(name = "login") String login, RedirectAttributes redirectAttributes) {

redirectAttributes.addAttribute("login", login);

return "redirect:/index";

}

@PostMapping(value = "/users")

public String createUser(@ModelAttribute("newUser")@Valid users newUser, BindingResult result, Model model){

if(result.hasErrors()){

return "redirect:/index";

}

newUser.setRoles(Collections.singleton(roleEnum.USER));

newUser.setPassword\_User(passwordEncoder.encode(newUser.getPassword\_User()));

repositoryUsers.save(newUser);

model.addAttribute("users", repositoryUsers.findAll());

return "redirect:/index";

}

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

public String editUser(Model model, @PathVariable("id") long id) {

users user = repositoryUsers.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid employee ID:" + id));

Iterable<discountCards> discountCards = discountCardRepository.findAll();

model.addAttribute("discountCards", discountCards);

model.addAttribute("user", user);

return "editUsers";

}

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

public String updateUser(@PathVariable("id") long id,@ModelAttribute("user") @Valid users user,BindingResult result, Model model){

if (result.hasErrors()) {

return "editUsers";

}

user.setID\_User(id);

user.setPassword\_User(passwordEncoder.encode(user.getPassword\_User()));

repositoryUsers.save(user);

model.addAttribute("users", repositoryUsers.findAll());

return "redirect:/index";

}

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

public String deleteUser(@PathVariable("id") long id,@RequestParam(name= "login", defaultValue = "") String login, @ModelAttribute("user")users user, Model model){

users delUser = repositoryUsers.findById(id).orElseThrow(() -> new IllegalArgumentException("Invalid employee ID: "+ id));

repositoryUsers.delete(delUser);

users getUser = new users();

model.addAttribute("users", repositoryUsers.findAll());

model.addAttribute("getUser", getUser);

model.addAttribute("login", login);

return "index";

}

}

Код categoryServices.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import java.util.List;

@Entity

@Table(name = "categoryServices")

public class categoryServices {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_categoryServices;

@NotBlank(message = "Name category services is required")

private String Name\_categoryServices;

@OneToOne(optional = true, mappedBy = "categoryServices")

private services Name\_Service;

public categoryServices(){}

public categoryServices(long ID\_categoryServices, String name\_categoryServices, services name\_Service) {

this.ID\_categoryServices = ID\_categoryServices;

Name\_categoryServices = name\_categoryServices;

Name\_Service = name\_Service;

}

public long getID\_categoryServices() {

return ID\_categoryServices;

}

public void setID\_categoryServices(long ID\_categoryServices) {

this.ID\_categoryServices = ID\_categoryServices;

}

public String getName\_categoryServices() {

return Name\_categoryServices;

}

public void setName\_categoryServices(String name\_categoryServices) {

Name\_categoryServices = name\_categoryServices;

}

public services getName\_Service() {

return Name\_Service;

}

public void setName\_Service(services name\_Service) {

Name\_Service = name\_Service;

}

}

Код discountCards.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

@Entity

@Table(name="discountCards")

public class discountCards {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_discountCard;

@NotBlank(message = "Number is required")

private String Number\_discountCard;

@OneToOne(optional = true, mappedBy = "discountCard")

private users Owner\_DiscountCard;

public discountCards(){}

public discountCards( String number\_discountCard) {

Number\_discountCard = number\_discountCard;

}

public long getID\_discountCard() {

return ID\_discountCard;

}

public void setID\_discountCard(long ID\_discountCard) {

this.ID\_discountCard = ID\_discountCard;

}

public String getNumber\_discountCard() {

return Number\_discountCard;

}

public void setNumber\_discountCard(String number\_discountCard) {

Number\_discountCard = number\_discountCard;

}

public users getOwner\_DiscountCard() {

return Owner\_DiscountCard;

}

public void setOwner\_DiscountCard(users owner\_DiscountCard) {

Owner\_DiscountCard = Owner\_DiscountCard;

}

}

Код employees.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import java.util.List;

@Entity

@Table(name = "employees")

public class employees {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_Employee;

@NotBlank(message = "First name is required")

@Size(max = 32, message = "Max Length 32")

private String FirstName\_Employee;

@NotBlank(message = "Second name is required")

@Size(max = 32, message = "Max Length 32")

private String SecondName\_Employee;

@Size(max = 32, message = "Max Length 32")

private String LastName\_Employee;

@NotBlank(message = "Login is required")

@Size(max = 32, message = "Max Length 32")

private String Login\_Employee;

@NotBlank(message = "Password is required")

private String Password\_Employee;

@ManyToMany

@JoinTable(name="post\_employee", joinColumns = @JoinColumn(name="employee\_id"),

inverseJoinColumns = @JoinColumn(name="post\_id"))

private List<posts> posts;

public employees(){}

public employees(String FirstName\_Employee, String SecondName\_Employee, String LastName\_Employee, String Login\_Employee, String Password\_Employee){

this.FirstName\_Employee = FirstName\_Employee;

this.SecondName\_Employee = SecondName\_Employee;

this.LastName\_Employee = LastName\_Employee;

this.Login\_Employee = Login\_Employee;

this.Password\_Employee = Password\_Employee;

}

public long getID\_Employee() {

return ID\_Employee;

}

public void setID\_Employee(long ID\_Employee) {

this.ID\_Employee = ID\_Employee;

}

public String getFirstName\_Employee() {

return FirstName\_Employee;

}

public void setFirstName\_Employee(String firstName\_Employee) {

FirstName\_Employee = firstName\_Employee;

}

public String getSecondName\_Employee() {

return SecondName\_Employee;

}

public void setSecondName\_Employee(String secondName\_Employee) {

SecondName\_Employee = secondName\_Employee;

}

public String getLastName\_Employee() {

return LastName\_Employee;

}

public void setLastName\_Employee(String lastName\_Employee) {

LastName\_Employee = lastName\_Employee;

}

public String getLogin\_Employee() {

return Login\_Employee;

}

public void setLogin\_Employee(String login\_Employee) {

Login\_Employee = login\_Employee;

}

public String getPassword\_Employee() {

return Password\_Employee;

}

public void setPassword\_Employee(String password\_Employee) {

Password\_Employee = password\_Employee;

}

public List<com.example.prakt3.models.posts> getPosts() {

return posts;

}

public void setPosts(List<com.example.prakt3.models.posts> posts) {

this.posts = posts;

}

}

Код executors.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

@Entity

@Table(name = "executors")

public class executors {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_Executor;

@NotBlank(message = "Name is required")

@Size(max = 32, message = "Max Length 32")

private String Name\_Executor;

@NotBlank(message = "INN is required")

@Size(max = 32, message = "Max Length 32")

private String INN\_Executor;

public executors(){}

public executors(String Name\_Executor, String INN\_Executor) {

this.Name\_Executor = Name\_Executor;

this.INN\_Executor = INN\_Executor;

}

public long getID\_Executor() {

return ID\_Executor;

}

public void setID\_Executor(long ID\_Executor) {

this.ID\_Executor = ID\_Executor;

}

public String getName\_Executor() {

return Name\_Executor;

}

public void setName\_Executor(String Name\_Executor) {

this.Name\_Executor = Name\_Executor;

}

public String getINN\_Executor() {

return INN\_Executor;

}

public void setINN\_Executor(String INN\_Executor) {

this.INN\_Executor = INN\_Executor;

}

}

Код orders.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import java.sql.Date;

import java.sql.Time;

@Entity

@Table(name = "orders")

public class orders {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_Order;

@NotBlank(message = "Number order is ruquired")

private String Number\_Order;

private Date Date\_Order;

@ManyToOne(fetch = FetchType.EAGER, targetEntity = users.class)

@JoinColumn(name = "user\_id")

private users user;

public orders(){}

public orders(String Number\_Order, Date Date\_Order, Time Time\_Order){

this.Number\_Order = Number\_Order;

this.Date\_Order = Date\_Order;

}

public long getID\_Order() {

return ID\_Order;

}

public void setID\_Order(long ID\_Order) {

this.ID\_Order = ID\_Order;

}

public String getNumber\_Order() {

return Number\_Order;

}

public void setNumber\_Order(String number\_Order) {

Number\_Order = number\_Order;

}

public Date getDate\_Order() {

return Date\_Order;

}

public void setDate\_Order(Date date\_Order) {

Date\_Order = date\_Order;

}

public users getUser() {

return user;

}

public void setUser(users user) {

this.user = user;

}

}

Код posts.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import java.util.List;

@Entity

@Table(name = "posts")

public class posts {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_Post;

@NotBlank(message = "Name post is required")

private String Name\_Post;

@NotNull(message = "Salary is required")

private double Salary\_Post;

@ManyToMany

@JoinTable(name="employee\_post", joinColumns = @JoinColumn(name="post\_id"),

inverseJoinColumns = @JoinColumn(name="employee\_id"))

private List<employees> employees;

public posts(){}

public posts(String Name\_Post, double Salary\_Post){

this.Name\_Post = Name\_Post;

this.Salary\_Post = Salary\_Post;

}

public long getID\_Post() {

return ID\_Post;

}

public void setID\_Post(long ID\_Post) {

this.ID\_Post = ID\_Post;

}

public String getName\_Post() {

return Name\_Post;

}

public void setName\_Post(String name\_Post) {

Name\_Post = name\_Post;

}

public double getSalary\_Post() {

return Salary\_Post;

}

public void setSalary\_Post(double salary\_Post) {

Salary\_Post = salary\_Post;

}

public List<com.example.prakt3.models.employees> getEmployees() {

return employees;

}

public void setEmployees(List<com.example.prakt3.models.employees> employees) {

this.employees = employees;

}

}

Код roleEnum.java:

package com.example.prakt3.models;

import org.springframework.security.core.GrantedAuthority;

public enum roleEnum implements GrantedAuthority {

USER, EMPLOYEE, ADMIN;

@Override

public String getAuthority()

{

return name();

}

}

Код services.java:

package com.example.prakt3.models;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

@Entity

@Table(name = "services")

public class services {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long ID\_Services;

@NotBlank(message = "Product name is required")

private String Name\_Services;

@NotNull(message = "Cost product is required")

private double Cost\_Services;

@NotBlank(message = "Article product is required")

@Size(min = 8, max = 8, message = "Min length 8")

private String Article\_Services;

@OneToOne(optional = true, cascade = CascadeType.ALL)

@JoinColumn(name = "categoryServices\_id")

private categoryServices categoryServices;

public services(){}

public services(String name\_Services, double cost\_Services, String article\_Services, categoryServices categoryServices) {

this.Name\_Services = name\_Services;

this.Cost\_Services = cost\_Services;

this.Article\_Services = article\_Services;

this.categoryServices = categoryServices;

}

public long getID\_Services() {

return ID\_Services;

}

public void setID\_Services(long ID\_Services) {

this.ID\_Services = ID\_Services;

}

public String getName\_Services() {

return Name\_Services;

}

public void setName\_Services(String name\_Services) {

Name\_Services = name\_Services;

}

public double getCost\_Services() {

return Cost\_Services;

}

public void setCost\_Services(double cost\_Services) {

Cost\_Services = cost\_Services;

}

public String getArticle\_Services() {

return Article\_Services;

}

public void setArticle\_Services(String article\_Services) {

Article\_Services = article\_Services;

}

public categoryServices getCategoryServices() {

return categoryServices;

}

public void setCategoryServices(categoryServices categoryServices) {

this.categoryServices = categoryServices;

}

}

Код editOrders.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Edit orders</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

<div th:insert="blocks/header" ></div>

</head>

<body>

<br/>

<hr/>

<form method="post" th:object="${order}">

<label for="Number\_Order">Enter number order: </label>

<input type="text" maxlength="8" minlength="8" th:field="\*{Number\_Order}" id="Number\_Order"/>

<span th:if="${#fields.hasErrors('Number\_Order')}" th:errors="\*{Number\_Order}" ></span>

<label for="Date\_Order">Enter date order: </label>

<input type="date" th:field="\*{Date\_Order}" id="Date\_Order"/>

<span th:if="${#fields.hasErrors('Date\_Order')}" th:errors="\*{Date\_Order}" ></span>

<div class="col">

<label>Select user:</label>

</div>

<div class="col">

<select name="login">

<div th:each="element : ${users}">

<option th:text="${element.Login\_User}"></option>

</div>

</select>

</div>

<br/>

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

</form>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>

</body>

</html>

Код editUsers.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Edit users</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

<div th:insert="blocks/header" ></div>

</head>

<body>

<br/>

<hr/>

<form method="post" th:object="${user}">

<label for="FirstName\_User">Enter first name: </label>

<input type="text" th:field="\*{FirstName\_User}" id="FirstName\_User"/>

<span th:if="${#fields.hasErrors('FirstName\_User')}" th:errors="\*{FirstName\_User}" ></span>

<label for="SecondName\_User">Enter second name: </label>

<input type="text" th:field="\*{SecondName\_User}" id="SecondName\_User"/>

<span th:if="${#fields.hasErrors('SecondName\_User')}" th:errors="\*{SecondName\_User}" ></span>

<label for="LastName\_User">Enter last name: </label>

<input type="text" th:field="\*{LastName\_User}" id="LastName\_User"/>

<span th:if="${#fields.hasErrors('LastName\_User')}" th:errors="\*{LastName\_User}" ></span>

<label for="Login\_User">Enter login: </label>

<input type="text" th:field="\*{Login\_User}" id="Login\_User"/>

<span th:if="${#fields.hasErrors('Login\_User')}" th:errors="\*{Login\_User}" ></span>

<label for="Password\_User">Enter password: </label>

<input type="text" th:field="\*{Password\_User}" id="Password\_User"/>

<span th:if="${#fields.hasErrors('Password\_User')}" th:errors="\*{Password\_User}" ></span>

<div class="col">

<label>Select discount card::</label>

</div>

<div class="col">

<select name="number">

<div th:each="element : ${discountCards}">

<option th:text="${element.Number\_discountCard}"></option>

</div>

</select>

</div>

<br/>

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

</form>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>

</body>

</html>

Код employees.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Employees</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

<div th:insert="blocks/header" ></div>

</head>

<body>

<br/>

<hr/>

<div class="container">

<div class="row">

<form th:method="POST" th:action="@{/admin/employee/createNewEmployees}" th:object="${employee}">

<div class="col">

<label for="FirstName\_Employee">Enter surname: </label>

</div>

<div class="col">

<input type="text" th:field="\*{FirstName\_Employee}" id="FirstName\_Employee">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('FirstName\_Employee')}" th:errors="\*{FirstName\_Employee}" ></span>

</div>

<div class="col">

<label for="SecondName\_Employee">Enter name: </label>

</div>

<div class="col">

<input type="text" th:field="\*{SecondName\_Employee}" id="SecondName\_Employee">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('SecondName\_Employee')}" th:errors="\*{SecondName\_Employee}" ></span>

</div>

<div class="col">

<label for="LastName\_Employee">Enter lastname: </label>

</div>

<div class="col">

<input type="text" th:field="\*{LastName\_Employee}" id="LastName\_Employee">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('LastName\_Employee')}" th:errors="\*{LastName\_Employee}" ></span>

</div>

<div class="col">

<label for="Login\_Employee">Enter login: </label>

</div>

<div class="col">

<input type="text" th:field="\*{Login\_Employee}" id="Login\_Employee">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Login\_Employee')}" th:errors="\*{Login\_Employee}" ></span>

</div>

<div class="col">

<label for="Password\_Employee">Enter password: </label>

</div>

<div class="col">

<input type="password" th:field="\*{Password\_Employee}" id="Password\_Employee">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Password\_Employee')}" th:errors="\*{Password\_Employee}" ></span>

</div>

<div class="col">

<input type="submit" value="Create!">

</div>

</form>

</div>

</div>

<br/>

<hr/>

<div class="container">

<div th:each="employee : ${employees}">

<div class="row">

<div class="row">

<div class="col">

<a th:href="@{/admin/employee/admin/employees/{id}(id=${employee.getID\_Employee()})}" th:text="${employee.getFirstName\_Employee()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/admin/employees/{id}(id=${employee.getID\_Employee()})}" th:text="${employee.getSecondName\_Employee()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/admin/employees/{id}(id=${employee.getID\_Employee()})}" th:text="${employee.getLastName\_Employee()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/admin/employees/{id}(id=${employee.getID\_Employee()})}" th:text="${employee.getLogin\_Employee()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/admin/employees/{id}(id=${employee.getID\_Employee()})}" th:text="${employee.getPassword\_Employee()}">user</a>

</div>

<div class="col">

<form th:each="post : ${employee.getPosts()}">

<div>

<a th:text="${post.getName\_Post()}"/>

<a th:href="'/admin/employee/'+ ${employee.ID\_Employee} + '/'+ ${post.ID\_Post} + '/remove'">Удалить должность</a>

</div>

</form>

</div>

<div class="col">

<a th:href="@{/admin/employee/delEmployee/{id}(id=${employee.getID\_Employee()})}">DELETE</a>

</div>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>

</body>

</html>

Код orders.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Orders</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

<div th:insert="blocks/header" ></div>

</head>

<body>

<br/>

<hr/>

<div class="container">

<div class="row">

<form th:method="POST" th:action="@{/user/employee/admin/createNewOrder}" th:object="${order}">

<div class="col">

<label for="Number\_Order">Enter number order: </label>

</div>

<div class="col">

<input type="text" maxlength="8" minlength="8" th:field="\*{Number\_Order}" id="Number\_Order">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Number\_Order')}" th:errors="\*{Number\_Order}" ></span>

</div>

<div class="col">

<label for="Date\_Order">Enter date order: </label>

</div>

<div class="col">

<input type="date" th:field="\*{Date\_Order}" id="Date\_Order">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Date\_Order')}" th:errors="\*{Date\_Order}" ></span>

</div>

<div class="col">

<label>Select user:</label>

</div>

<div class="col">

<select name="login">

<div th:each="element : ${users}">

<option th:text="${element.Login\_User}"></option>

</div>

</select>

</div>

<div class="col">

<input type="submit" value="Create!">

</div>

</form>

</div>

</div>

<br/>

<hr/>

<div class="container">

<div th:each="order : ${orders}">

<div class="row">

<div class="col">

<a th:href="@{/user/employee/admin/orders/{id}(id=${order.getID\_Order()})}" th:text="${order.getNumber\_Order()}">user</a>

</div>

<div class="col">

<a th:href="@{/user/employee/admin/orders/{id}(id=${order.getID\_Order()})}" th:text="${order.getDate\_Order()}">user</a>

</div>

<div class="col">

<a th:href="@{/user/employee/admin/orders/{id}(id=${order.getID\_Order()})}" th:text="${order.getUser().getLogin\_User()}">user</a>

</div>

<div class="col">

<a th:href="@{/user/employee/admin/delOrder/{id}(id=${order.getID\_Order()})}">DELETE</a>

</div>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>

</body>

</html>

Код posts.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Posts</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

<div th:insert="blocks/header" ></div>

</head>

<body>

<br/>

<hr/>

<div class="container">

<div class="row">

<form th:method="POST" th:action="@{/admin/employee/createNewPost}" th:object="${post}">

<div class="col">

<label for="Name\_Post">Enter name post: </label>

</div>

<div class="col">

<input type="text" th:field="\*{Name\_Post}" id="Name\_Post">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Name\_Post')}" th:errors="\*{Name\_Post}" ></span>

</div>

<div class="col">

<label for="Salary\_Post">Enter salary: </label>

</div>

<div class="col">

<input type="number" th:field="\*{Salary\_Post}" id="Salary\_Post">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Salary\_Post')}" th:errors="\*{Salary\_Post}" ></span>

</div>

<div class="col">

<input type="submit" value="Create!">

</div>

</form>

</div>

</div>

<br/>

<hr/>

<div class="container">

<div th:each="post : ${posts}">

<div class="row">

<div class="col">

<a th:href="@{/admin/employee/posts/{id}(id=${post.getID\_Post()})}" th:text="${post.getName\_Post()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/{id}/editPosts(id=${post.getID\_Post()})}" th:text="${post.getSalary\_Post()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/delPost/{id}(id=${post.getID\_Post()})}">DELETE</a>

</div>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>

</body>

</html>

Код regis.html:

<!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="@{/regis}" method="post">

<div><label> Фамилия : <input type="text" name="FirstName\_User"/> </label></div>

<div><label> Имя : <input type="text" name="SecondName\_User"/> </label></div>

<div><label> Отчество : <input type="text" name="LastName\_User"/> </label></div>

<div><label> Логин : <input type="text" name="Login\_User"/> </label></div>

<div><label> Пароль : <input type="password" name="Password\_User"/> </label></div>

<div><input type="submit" value="Регистрация"/></div>

</form>

</body>

</html>

Код roleChange.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Change Role User</title>

<div th:insert="blocks/header" ></div>

</head>

<body>

<div class="container">

<div th:each="user : ${users}">

<div class="row">

<div class="col">

<a th:text="${user.getFirstName\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getSecondName\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getLastName\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getLogin\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getPassword\_User()}">user</a>

</div>

<div th:each="role : ${roles}">

<input type="checkbox" th:id="${role}"

th:name="${'roles[]'}" th:value="${role}"

th:checked="${#lists.contains(user.roles, role)}" disabled>

<label th:text="${role}"></label>

</div>

<div class="col">

<a class="btn btn-success" th:href="@{/admin/employee/updaterole/{id}(id=${user.getID\_User()})}">Изменить роль пользователя</a>

</div>

</div>

</div>

</div>

</body>

</html>

Код services.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Services</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

<div th:insert="blocks/header" ></div>

</head>

<body>

<br/>

<hr/>

<div class="container">

<div class="row">

<form th:method="POST" th:action="@{/admin/employee/createNewService}" th:object="${service}">

<div class="col">

<label for="Name\_Service">Enter name service: </label>

</div>

<div class="col">

<input type="text" th:field="\*{Name\_Services}" id="Name\_Service">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Name\_Services')}" th:errors="\*{Name\_Services}" ></span>

</div>

<div class="col">

<label for="Cost\_Services">Enter cost service: </label>

</div>

<div class="col">

<input type="number" th:field="\*{Cost\_Services}" id="Cost\_Services">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Cost\_Services')}" th:errors="\*{Cost\_Services}" ></span>

</div>

<div class="col">

<label for="Article\_Services">Enter article service: </label>

</div>

<div class="col">

<input type="number" maxlength="8" th:field="\*{Article\_Services}" id="Article\_Services">

</div>

<div class="col">

<span th:if="${#fields.hasErrors('Article\_Services')}" th:errors="\*{Article\_Services}" ></span>

</div>

<div class="col">

<label>Select category:</label>

</div>

<div class="col">

<select name="number">

<div th:each="element : ${categoryServices}">

<option th:text="${element.Name\_categoryServices}"></option>

</div>

</select>

</div>

<div class="col">

<input type="submit" value="Create!">

</div>

</form>

</div>

</div>

<br/>

<hr/>

<div class="container">

<div th:each="service1 : ${services}">

<div class="row">

<div class="col">

<a th:href="@{/admin/employee/services/{id}(id=${service1.getID\_Services()})}" th:text="${service1.getName\_Services()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/services/{id}(id=${service1.getID\_Services()})}" th:text="${service1.getCost\_Services()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/services/{id}(id=${service1.getID\_Services()})}" th:text="${service1.getArticle\_Services()}">user</a>

</div>

<div class="col">

<a th:href="@{/admin/employee/delServices/{id}(id=${service1.getID\_Services()})}">DELETE</a>

</div>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>

</body>

</html>

Код updateRole.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>UpdateRole</title>

<div th:insert="blocks/header" ></div>

</head>

<body>

<form method="post" th:object="${user}">

<div class="container">

<div class="row">

<div class="col">

<a th:text="${user.getFirstName\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getSecondName\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getLastName\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getLogin\_User()}">user</a>

</div>

<div class="col">

<a th:text="${user.getPassword\_User()}">user</a>

</div>

<div th:each="role : ${roles}">

<input type="checkbox" th:id="${role}"

th:name="${'roles[]'}" th:value="${role}"

th:checked="${#lists.contains(user.roles, role)}">

<label th:text="${role}"></label>

</div>

<div class="col">

<button class="btn btn-success" type="submit">Изменить пользователя</button>

</div>

</div>

</div>

</form>

</body>

</html>

Продемонстрируем работу программы:

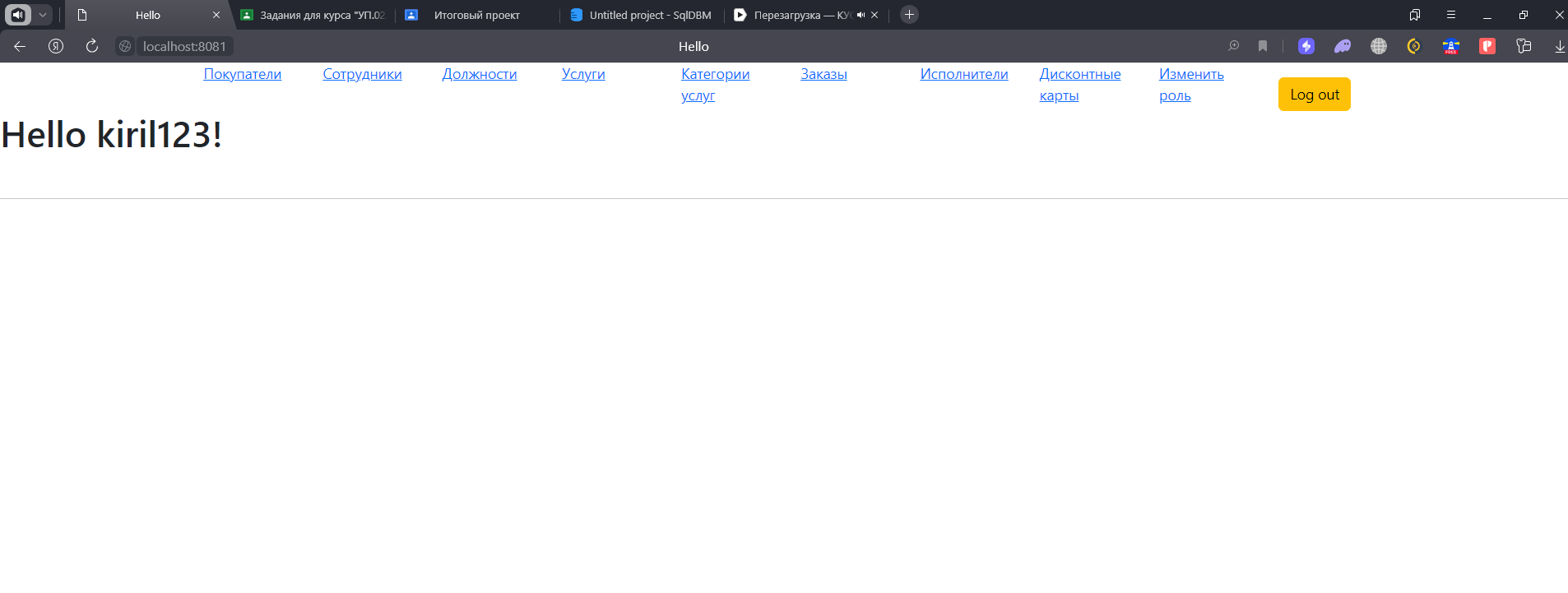


Рисунок 3 - Главная страница

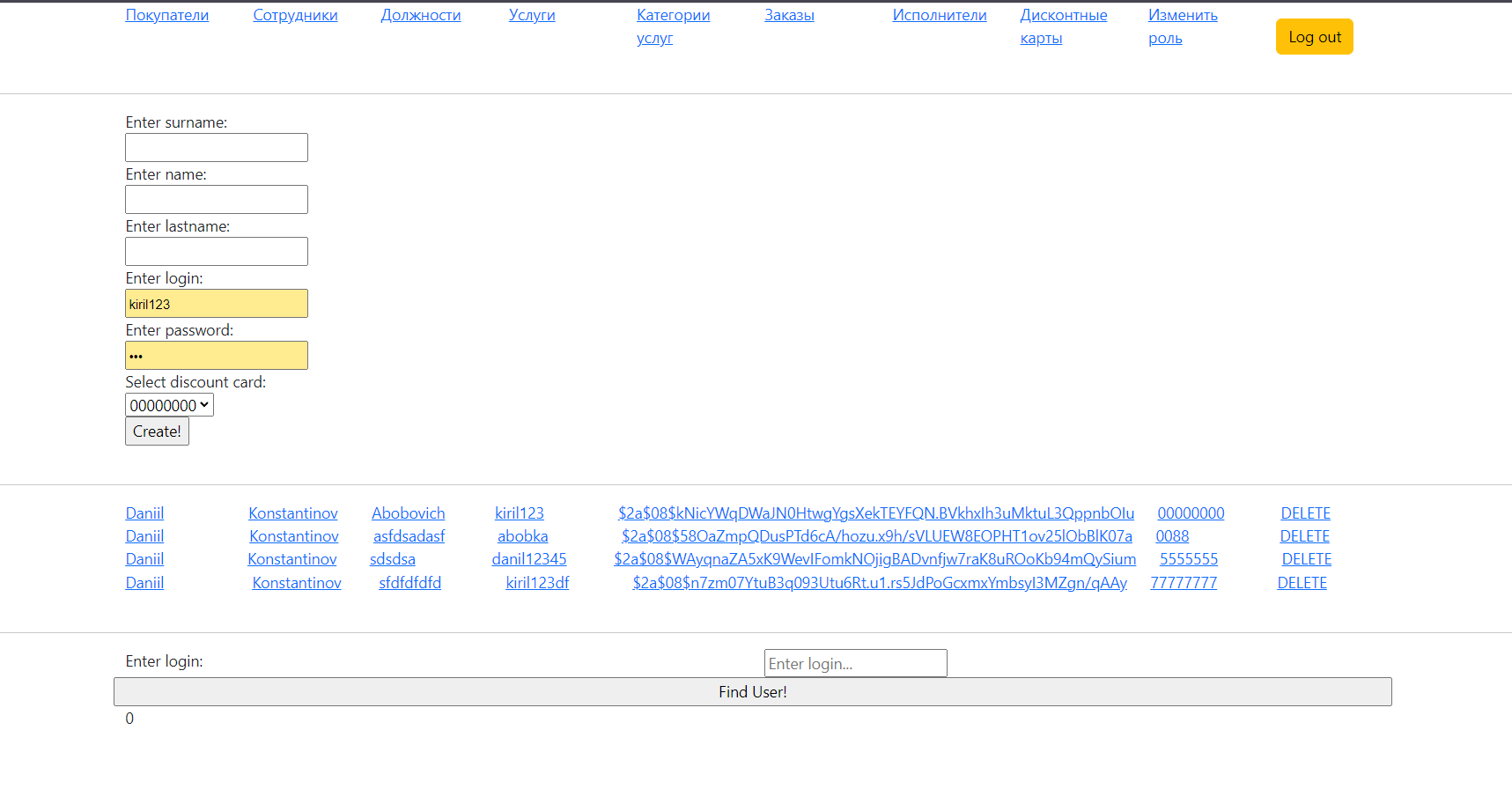


Рисунок 4 – Покупатели

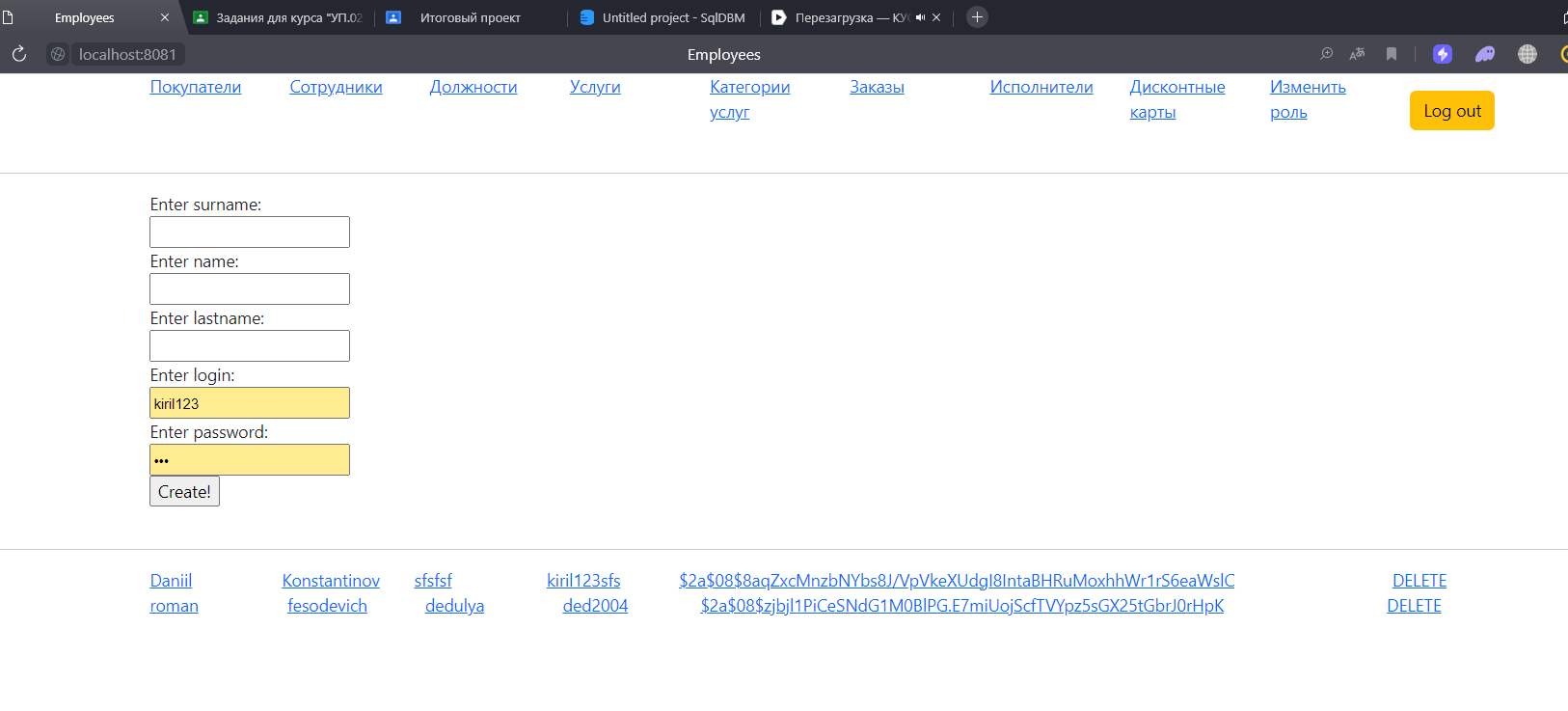


Рисунок 5 – Сотрудники

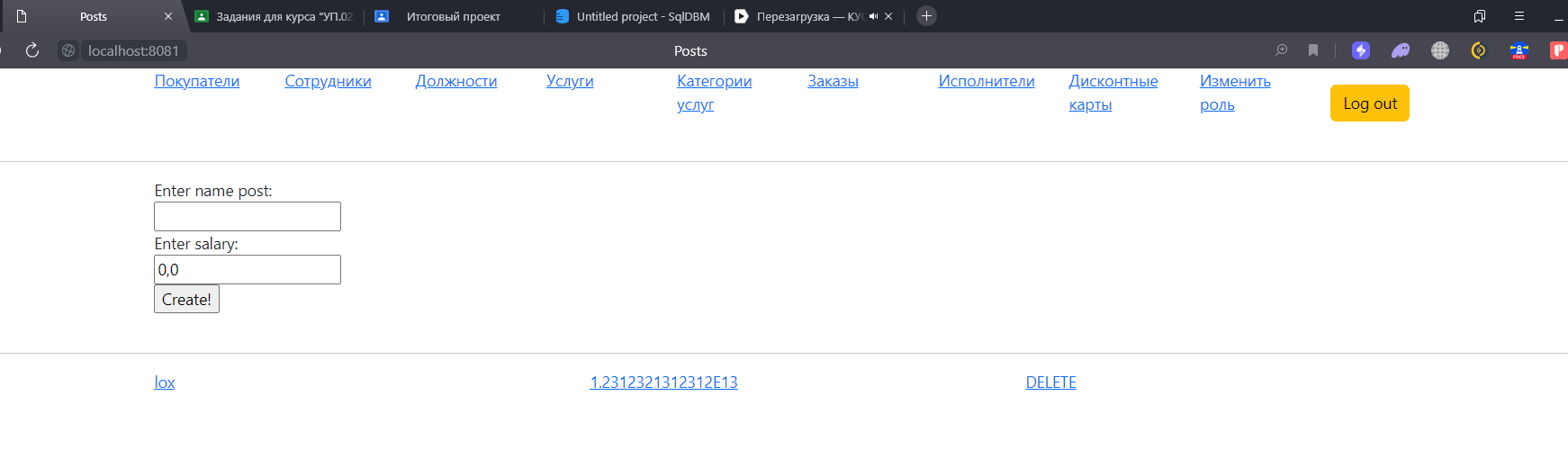


Рисунок 6 – Должности

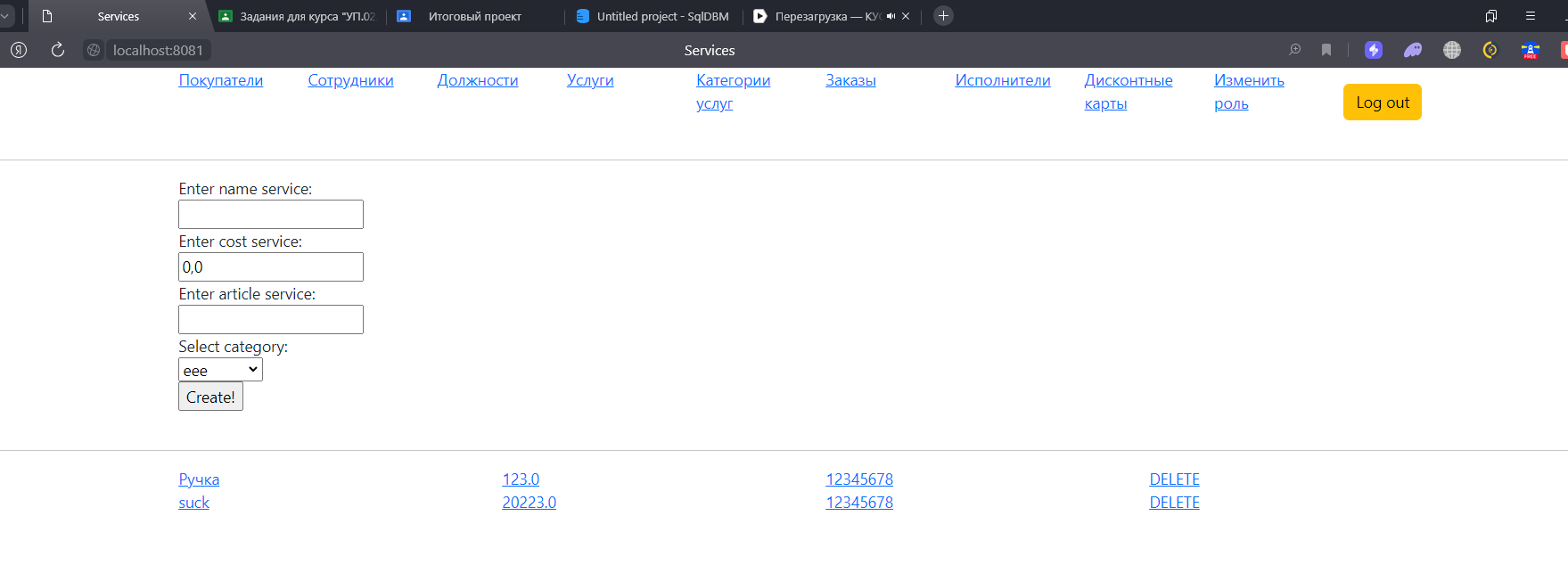


Рисунок 7 – Должности

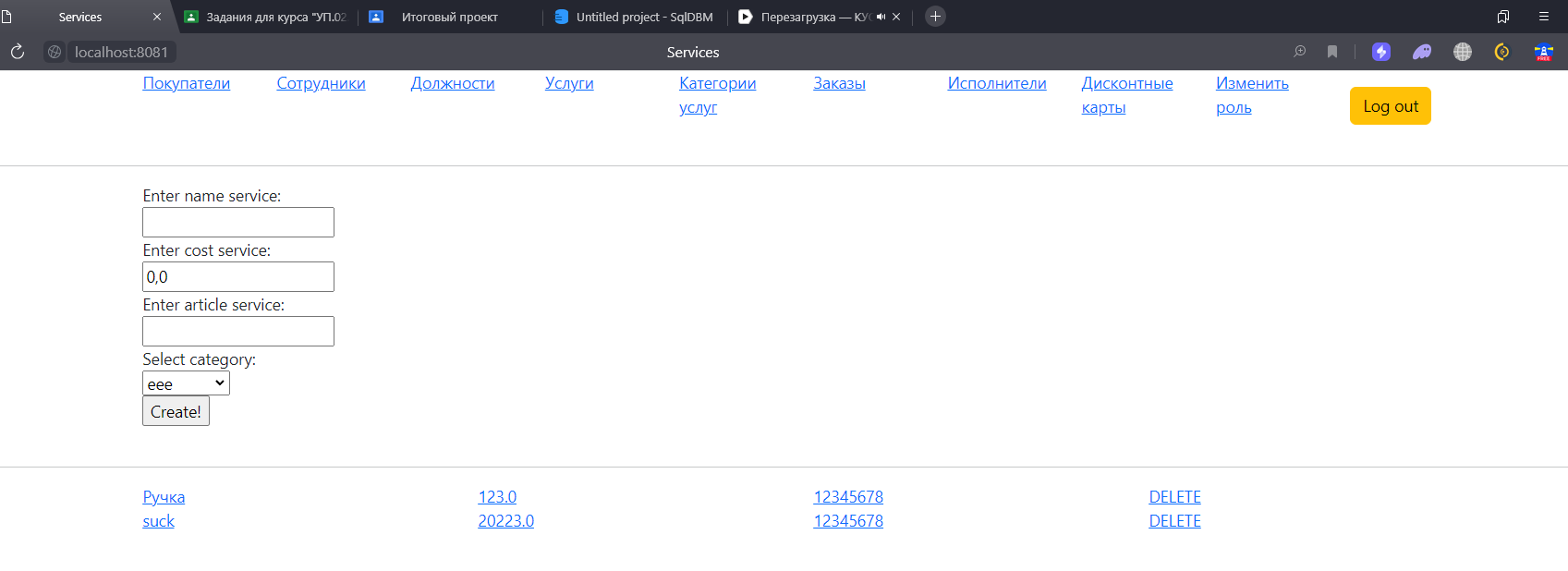


Рисунок 8 – Услуги

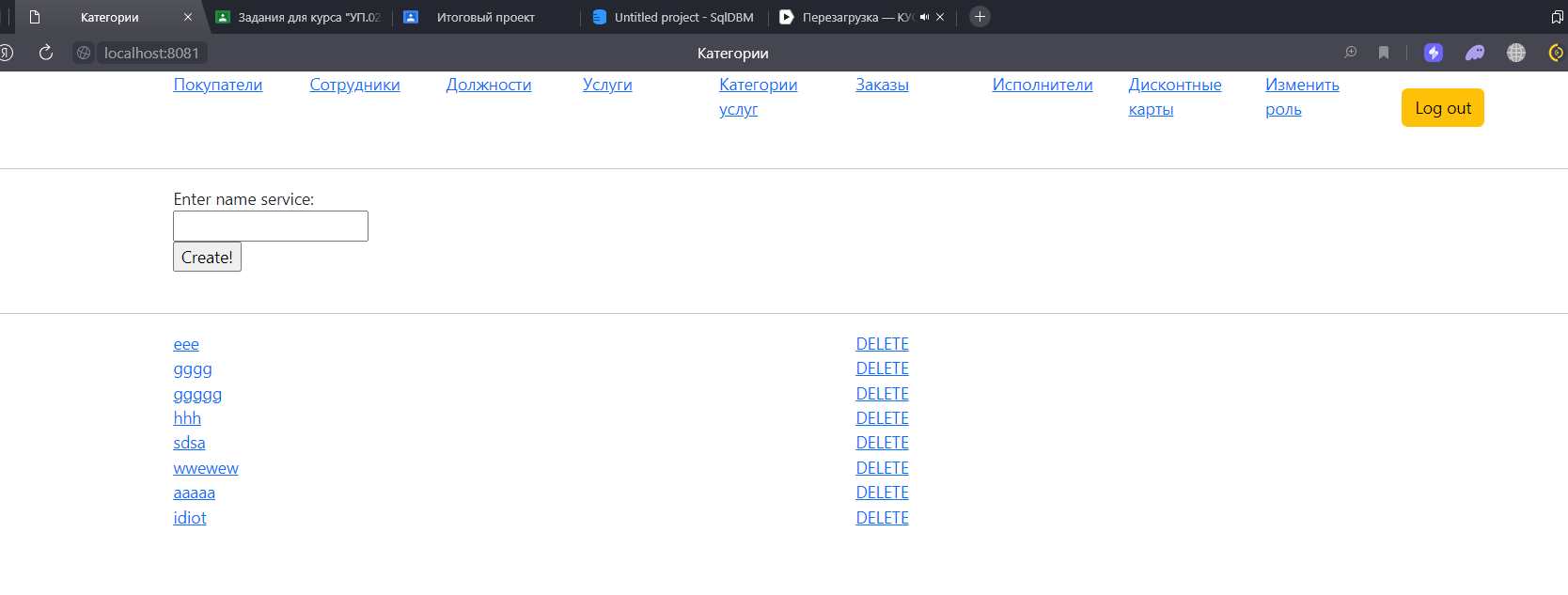


Рисунок 9 - Категории услуг

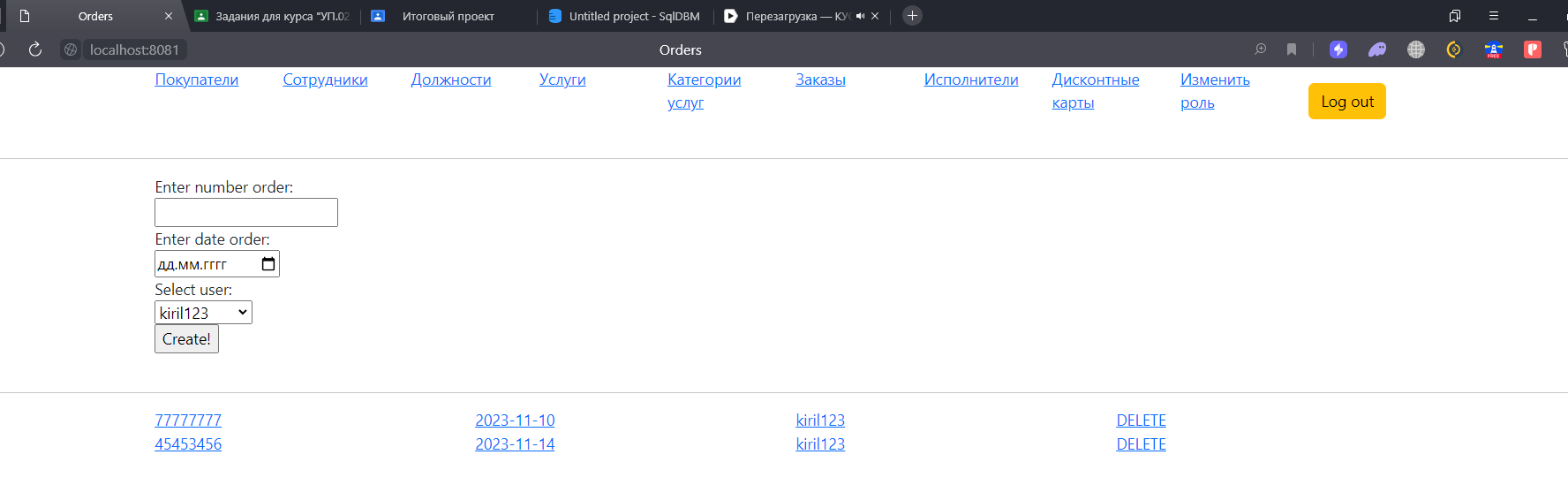


Рисунок 10 – заказы

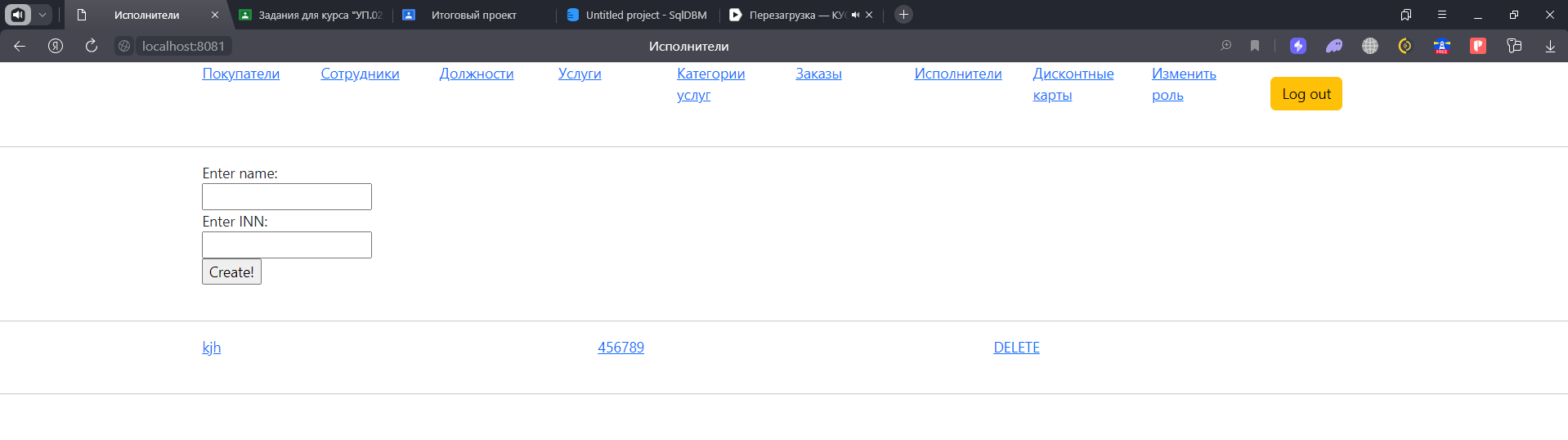


Рисунок 11 – Исполнители

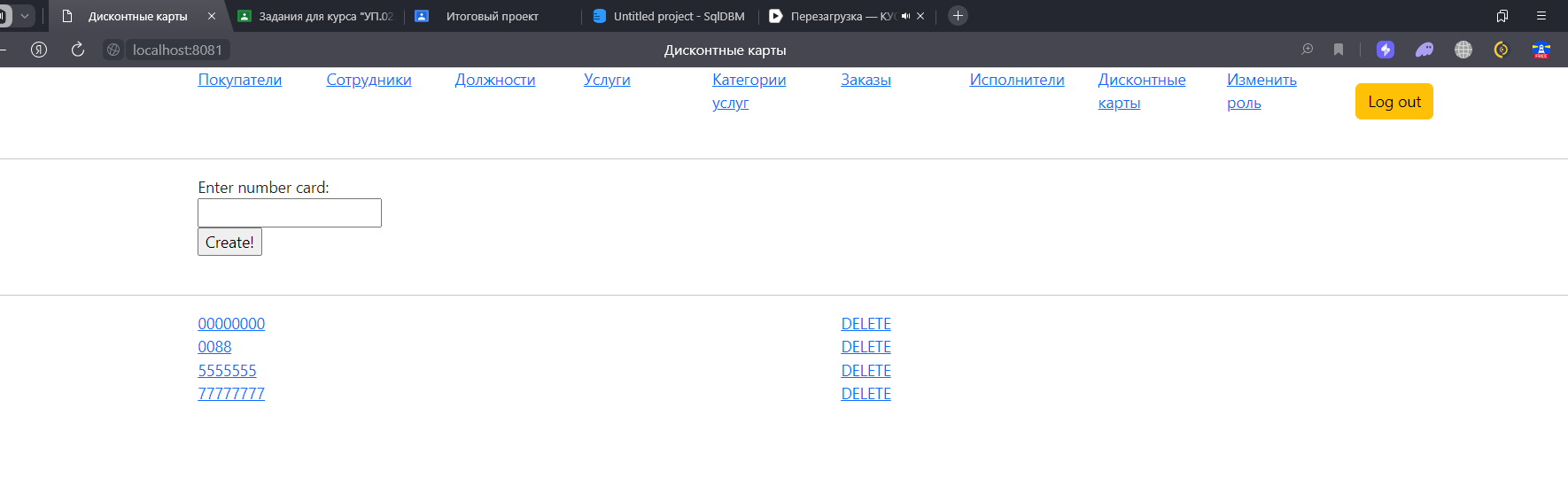


Рисунок 12 - Дисконтные карты

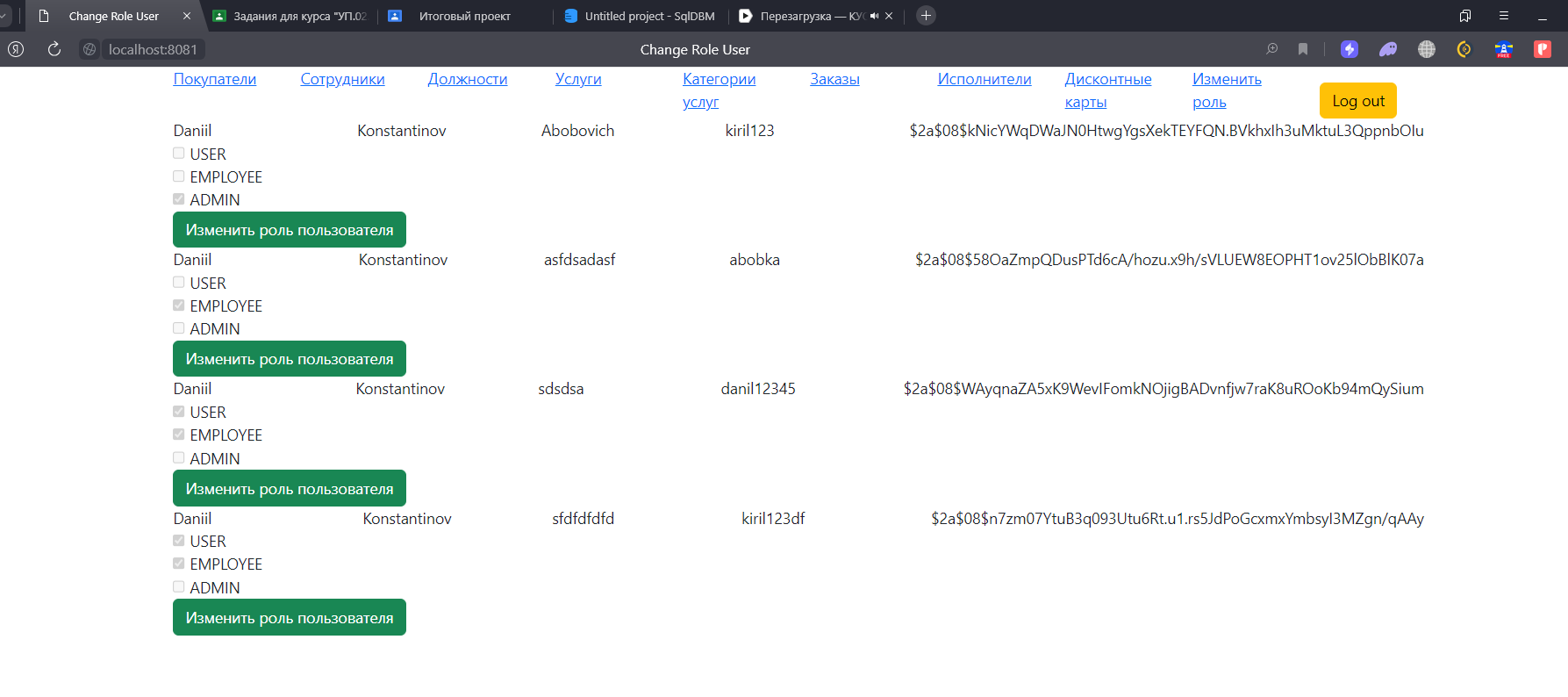


Рисунок 13 - Изменение ролей

Продемонстрируем часть работы программы:

Добавим услугу, для этого добавим категорию услуг «Стройка»

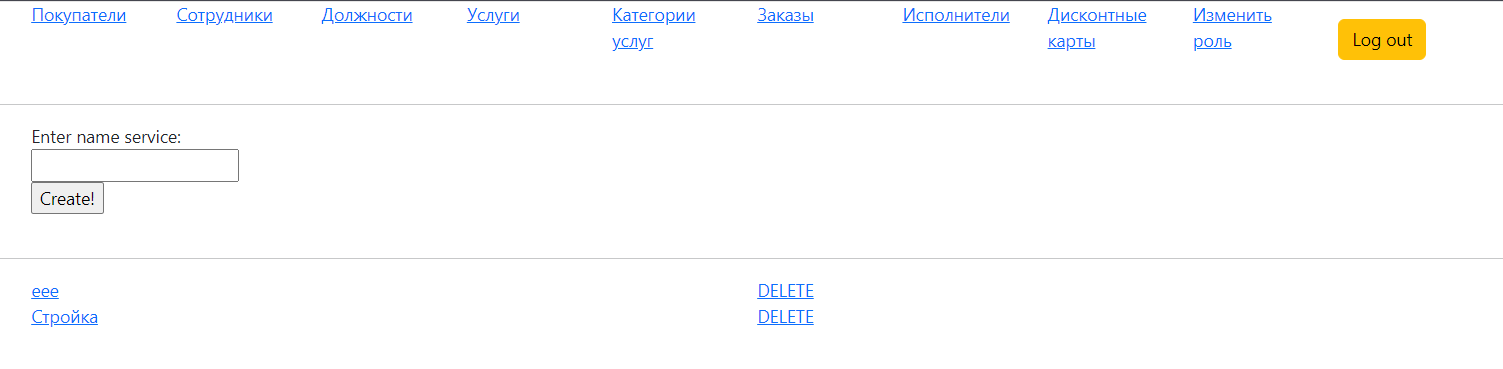


Рисунок 14 - Добавление категории

Теперь добавим услугу с этой категорией

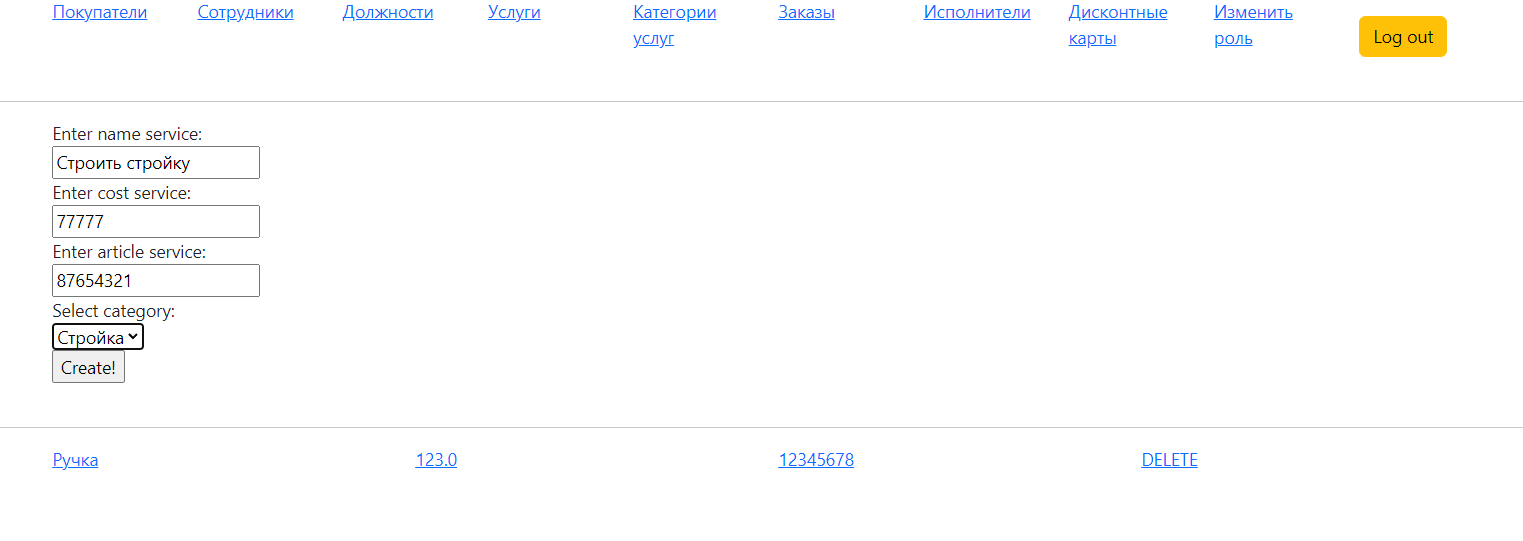


Рисунок 15 - Заполнение полей



Рисунок 16 – Добавлено

Удалим услугу:

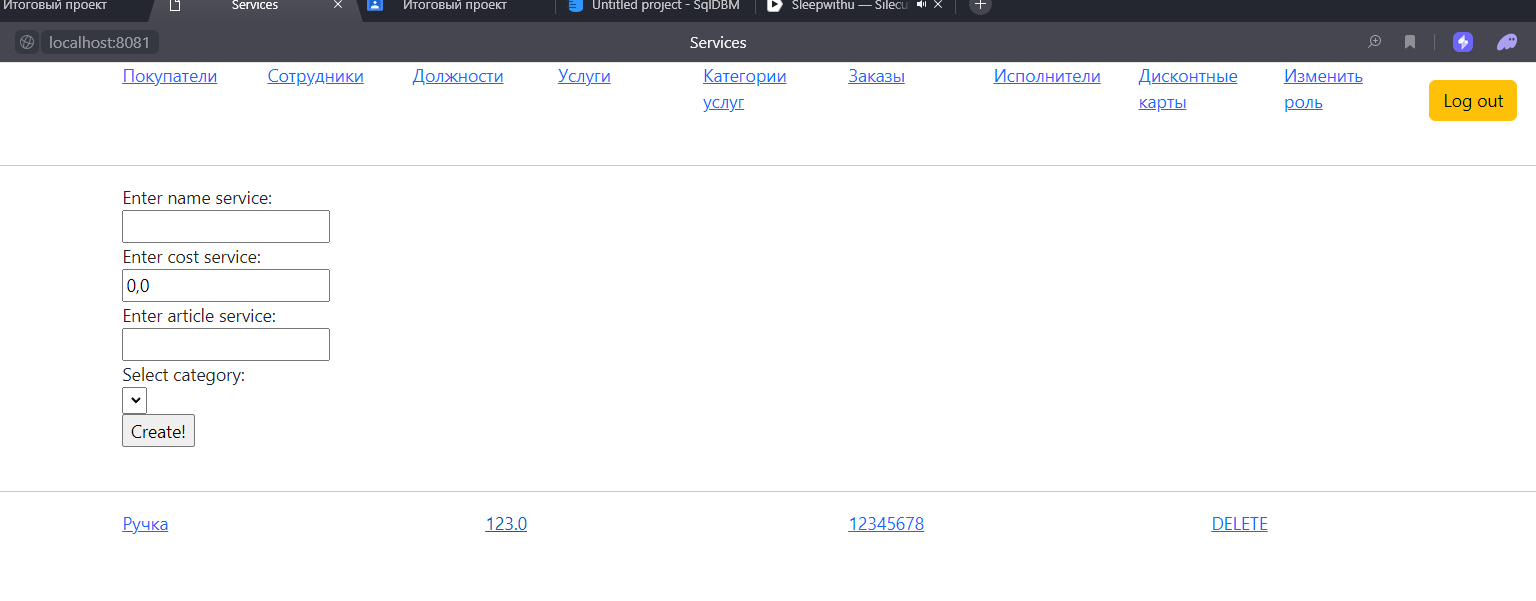


Рисунок 17 - Удаление

# вывод

В ходе прохождения производственной практики был разработан сайт на тему «Сервис услуг».