**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

**Кафедра информатики и защиты информации**

СМК 08/03-28

Срок хранения 5 лет

ВЛГУ.100301.14.05.00 ПЗ

УТВЕРЖДАЮ

Заведующий кафедрой ИЗИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Ю. Монахов

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_20\_\_

**КУРСОВАЯ РАБОТА**

Разработка автоматизированной информационной системы

записей в автосервис

Владимир 2024

**СПИСОК ИСПОЛНИТЕЛЕЙ**

|  |  |  |
| --- | --- | --- |
| Руководитель работы |  | доц. каф. ИЗИ Д.А. Полянский |
|  | подпись, дата |  |
| Студент группы ИБ-122 |  | Д.С. Чугров |
|  | подпись, дата |  |

# АННОТАЦИЯ

Ключевые слова: АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА, БАЗА ДАННЫХ, ЗАПИСИ В АВТОСЕРВИС.

Целью работы является разработка автоматизированная информационная система записей в автосервис для повышения эффективности обработки заявок клиентов, составления расписания и анализа работы автосервиса (автоматизация составления отчетов).

Назначение разрабатываемой системы: автоматизация процесса обработки записи клиента в автосервис, обработки данных клиента и составления отчетов работы сервиса.

ОГЛАВЛЕНИЕ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | Стр. |
| Принятые сокращения | | | | | | | 6 |
| Введение | | | | | | | 7 |
| 1. | Основная часть | | | | | | 8 |
|  | 1.1. | Функциональная структура системы | | | | | 8 |
|  | 1.2. | Состав лингвистического, программного и технического обеспечения | | | | | 8 |
|  | 1.3. | Подсистема хранения данных | | | | | 9 |
|  |  | 1.3.1. | Таблица пользователей | | | |  |
|  |  |  | 1.3.1.1 | Описание полей таблицы пользователей | | | 10 |
|  |  | 1.3.2. | Таблица автомобилей | | | | 10 |
|  |  |  | 1.3.2.1 | Описание полей таблицы автомобилей | | | 10 |
|  |  | 1.3.3 | Таблица заказов | | | | 11 |
|  |  |  | 1.3.3.1. | Описание полей таблицы заказов | | | 11 |
|  |  | 1.3.4. | Таблица слотов записи | | | | 11 |
|  |  |  | 1.3.4.1. | Описание полей таблицы слотов записи | | | 11 |
|  | 1.4. | Подсистема визуализации данный | | | | | 12 |
|  |  | 1.4.1. | Таблица данных персонала | | | | 12 |
|  |  | 1.4.2. | Таблица данных клиентов | | | | 13 |
|  |  | 1.4.3. | Таблица автомобилей клиентов | | | | 14 |
|  |  | 1.4.4. | График количества оказанных услуг за месяц и процент загруженности в конкретный день | | | | 16 |
|  | 1.5 | Подсистема поиска данных | | | | | 17 |
|  |  | 1.5.1. | Поиск клиента | | | | 18 |
|  |  | 1.5.2. | Поиск персонала | | | | 18 |
|  |  | 1.5.3. | Поиск автомобиля клиента | | | | 19 |
|  |  | 1.5.4. | Поиск заказа | | | | 19 |
|  | 1.6. | Подсистема управления данными | | | | | 20 |
|  |  | 1.6.1. | Управление клиентами | | | | 20 |
|  |  |  | 1.6.1.1. | Регистрация клиента | | | 20 |
|  |  |  | 1.6.1.2. | Редактирование данных | | | 21 |
|  |  | 1.6.2. | Управление персоналом | | | | 22 |
|  |  | 1.6.3. | Управление заказами | | | | 23 |
|  |  |  | 1.6.3.1. | Создание заказа клиентом | | | 23 |
|  |  |  | 1.6.3.2. | Изменение и присвоение статуса заказа менеджером | | | 24 |
|  | 1.7. | Подсистема формирования отчетов и заказ-нарядов | | | | | 26 |
|  |  | 1.7.1. | Формирование отчетов | | | | 26 |
|  |  | 1.7.2. | Формирование заказ-нарядов | | | | 26 |
|  | 1.8. | Взаимодействие между подсистемами | | | | | 30 |
|  | 1.9. | Функционирование системы в различных режимах | | | | | 30 |
|  | 1.10 | Реализация требований защиты информации | | | | | 31 |
| Заключение | | | | | | | 32 |
| Список использованных источников | | | | | | | 33 |
| Приложение А | | | | | | | 34 |

# ПРИНЯТЫЕ СОКРАЩЕНИЯ

БД – база данных,

ИС – информационная система,

АИСЗА - Автоматизированная информационная система записей в автосервис,

СУБД - Система управления базами данных.

# ВВЕДЕНИЕ

Целью работыявляется разработка автоматизированная информационная система записей в автосервис. Для выполнения данной работы были поставлены следующие задачи:

* создание БД;
* стартовое (первоначальное) наполнение базы данных;
* разработка правил реализации функционала системы;
* отладка реализованных функций
* демонстрация реализованных функций в условиях полных и неполных данных.

Актуальность данной работы заключается в необходимости повышения эффективности управления записями клиентов и их обслуживания в автосервисах за счет автоматизации процессов, что позволяет минимизировать ошибки, улучшить качество обслуживания и оптимизировать использование ресурсов.

Основанием для разработки автоматизированная информационная система записей в автосервис (АИСЗА) является техническое задание на курсовую работу по дисциплине «Базы данных и экспертные системы» от 02.09.2024.

# ОСНОВНАЯ ЧАСТЬ

## Функциональная структура системы

АИСЗА состоит из следующих подсистем:

* подсистема хранения данных;
* подсистема авторизации;
* подсистема записи в автосервис;
* подсистема поиска клиента в базе данных;
* подсистема формирования отчетов;
* подсистема формирования заказ-нарядов;
* подсистема безопасности и контроля доступа.

## Состав лингвистического, программного и технического обеспечения

При реализации системы должны применяться следующие языки высокого уровня: Java. Выбор языка Java обусловлен его высокой популярностью, кроссплатформенностью, обширной экосистемой библиотек и фреймворков, а также возможностью создавать масштабируемые и надежные приложения, что идеально подходит для разработки информационных систем.

При реализации системы должны применяться следующие языки и стандарты взаимодействия АИСЗА со смежными системами и пользователей с АИСЗА: должны использоваться встроенные средства диалогового взаимодействия приложения: Tomcat и Java Servlet API.

Реализуемая информационная система рассчитана на функционирование в следующей программной среде: IntelliJ IDEA.

Система должна быть реализована с использованием серверов:

Сервер базы данных должен быть развернут на оборудовании:

* Процессор: не менее 1 ядра
* Оперативная память: не менее 1 ГБ
* Жесткий диск: не менее 15 ГБ

Сервер сбора, обработки и загрузки данных должен быть развернут на оборудовании:

* Процессор: не менее 1 ядра
* Оперативная память: не менее 1 ГБ
* Жесткий диск: не менее 15 ГБ

Сервер приложений должен быть развернут на оборудовании:

* Процессор: не менее 1 ядра
* Оперативная память: не менее 1 ГБ
* Жесткий диск: не менее 15 ГБ

Минимальный объем свободного пространства для хранения данных на дисковом массиве должен составлять 10 Гб.

## Подсистема хранения данных

Подсистема хранения данных АИСЗА реализована с помощью промышленной СУБД MySQL 5.7.40. Взаимосвязь таблиц представлена на рисунке ниже.

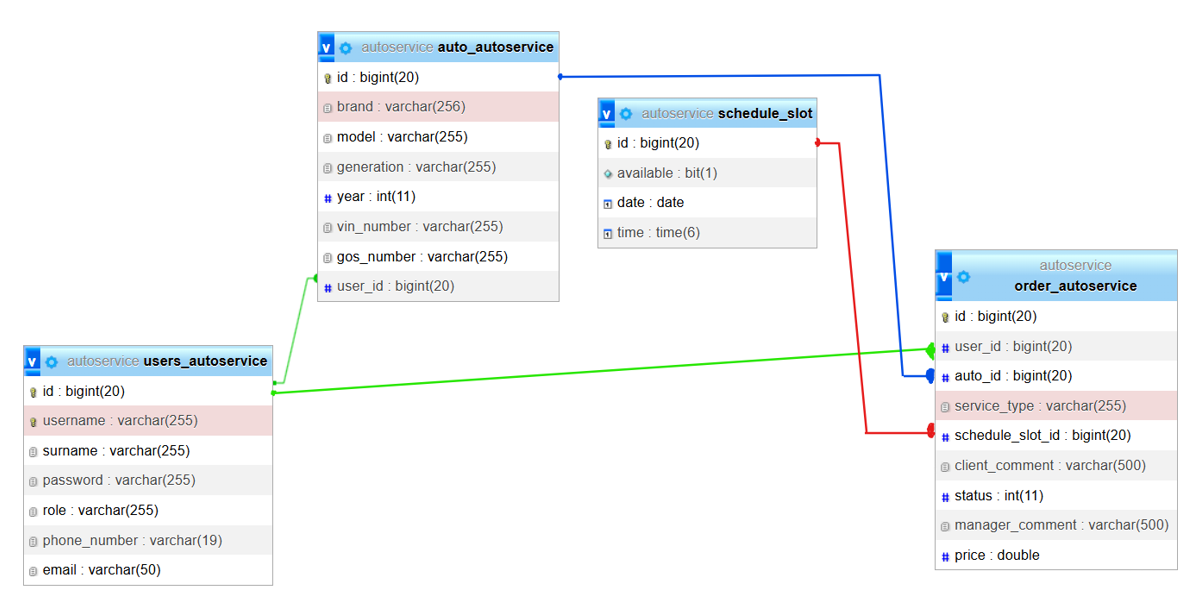


Рис. 1. Взаимосвязь таблиц базы данных для АИСЗА

1.3.1. Таблица пользователей

Система хранения пользователей представлена таблицей users\_autoservice. Она хранит основные данные клиентов, менеджеров и операторов БД.

1.3.1.1. Описание полей таблицы пользователей

1. id - индивидуальный идентификатор пользователя. Тип данных -BIGINT(20);

2. username - имя пользователя. Тип данных - VARCHAR(255);

3. surname - фамилия пользователя. Тип данных - VARCHAR(255);

4. password - пароль пользователя. Тип данных - VARCHAR(255);

5. role - роль пользователя. Тип данных - VARCHAR(255);

6. phone\_number - номер телефона пользователя. Тип данных - VARCHAR(19);

7. email - электронная почта пользователя. Тип данных - VARCHAR(50).

1.3.2. Таблица автомобилей

Система хранения автомобилей представлена таблицей auto\_autoservice. Она хранит основные данные об автомобилях клиентов.

1.3.2.1. Описание полей таблицы автомобилей

1. id - индивидуальный идентификатор автомобиля. Тип данных -BIGINT(20);

2. brand - марка автомобиля. Тип данных - VARCHAR(255);

3. model - модель автомобиля. Тип данных - VARCHAR(255);

4. generation - поколение автомобиля. Тип данных - VARCHAR(255);

5. year - год автомобиля. Тип данных - INT(4);

6. vin\_number - VIN номер автомобиля. Тип данных - VARCHAR(17);

7. gos\_number - государственный регистрационный номер. Тип данных - VARCHAR(9);

8. user\_id - индивидуальный идентификатор пользователя. Тип данных -BIGINT(20).

1.3.3. Таблица заказов

Система хранения заказов клиентов представлена таблицей order\_autoservice. Она хранит основные данные о заказах клиентов.

1.3.3.1. Описание полей таблицы заказов

1. id - индивидуальный идентификатор заказа. Тип данных -BIGINT(20);

2. user\_id - индивидуальный идентификатор пользователя. Тип данных -BIGINT(20);

3. auto\_id - индивидуальный идентификатор автомобиля. Тип данных -BIGINT(20);

4. service\_type - тип услуги. Тип данных - VARCHAR(255);

5. schedule\_slot\_id - индивидуальный идентификатор слота записи. Тип данных - BIGINT(20);

6. client\_comment - комментарий пользователя. Тип данных - VARCHAR(500);

7. status - статус заказа. Тип данных - INT(11);

8. manager\_comment - комментарий менеджера. Тип данных - VARCHAR(500);

9. price - цена заказа. Тип данных - DOUBLE.

1.3.4. Таблица слотов записи

Система хранения слотов записи на услуги представлена таблицей schedule\_slot. Она хранит доступные для записи слоты.

1.3.4.1. Описание полей таблицы слотов записи

1. id - индивидуальный идентификатор слота записи. Тип данных -BIGINT(20);

2. available - индикатор доступности слота для записи. Тип данных - BIT(1).

3. date - дата записи. Тип данных - DATE.

4. time - время записи. Тип данных - TIME(6).

## Подсистема визуализации данных

АИСЗА обеспечивает визуализацию следующей информации:

* Таблица данных персонала;
* Таблица данных клиентов;
* Таблица автомобилей клиентов;
* График количества оказанных услуг за месяц;
* Процент загруженности в конкретный день.

1.4.1. Таблица данных персонала

Визуализация таблицы данных персонала реализована для пользователя с ролью «Администратор БД» в виде страницы «/operatorBD/personnel».

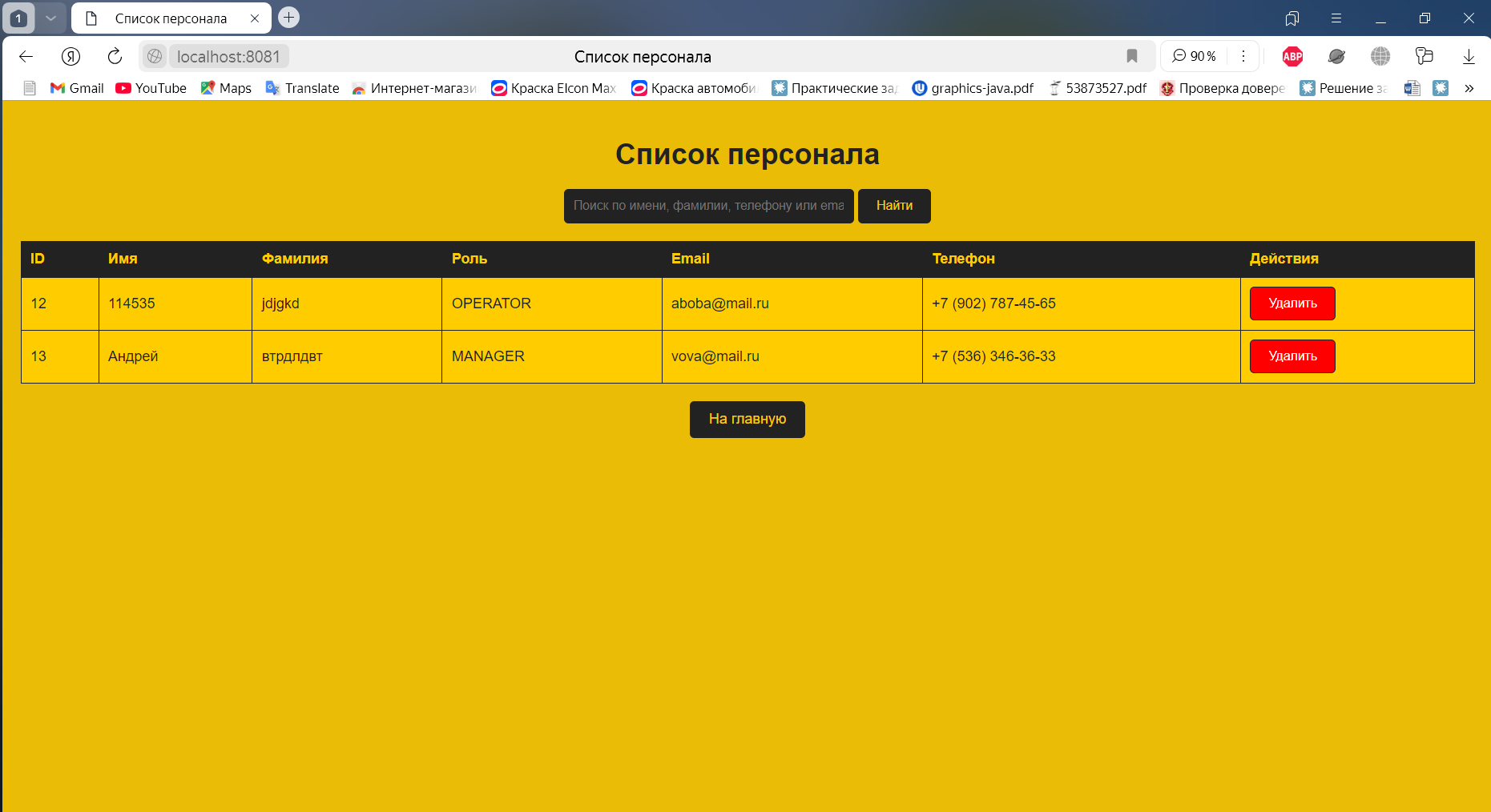


Рис. 2. Страница «Список персонала»

*// Метод вывода страницы "Список персонала"* @GetMapping("/personnel")  
 public String showPersonnel(  
 @RequestParam(value = "search", required = false) String searchQuery,  
 Model model) {  
 List<Users> personnel;  
 if (searchQuery != null && !searchQuery.isEmpty()) {  
 personnel = usersService.searchPersonnel(searchQuery);  
 } else {  
 personnel = usersService.getAllPersonnel();  
 }  
 model.addAttribute("clients", personnel);  
 model.addAttribute("searchQuery", searchQuery);  
 return "personnelList";  
 }

Листинг 1. Код показа страницы «/operatorBD/personnel»

1.4.2. Таблица данных клиентов

Визуализация таблицы данных клиентов реализована для пользователя с ролью «Менеджер автосервиса» в виде страницы «/manager/clients».

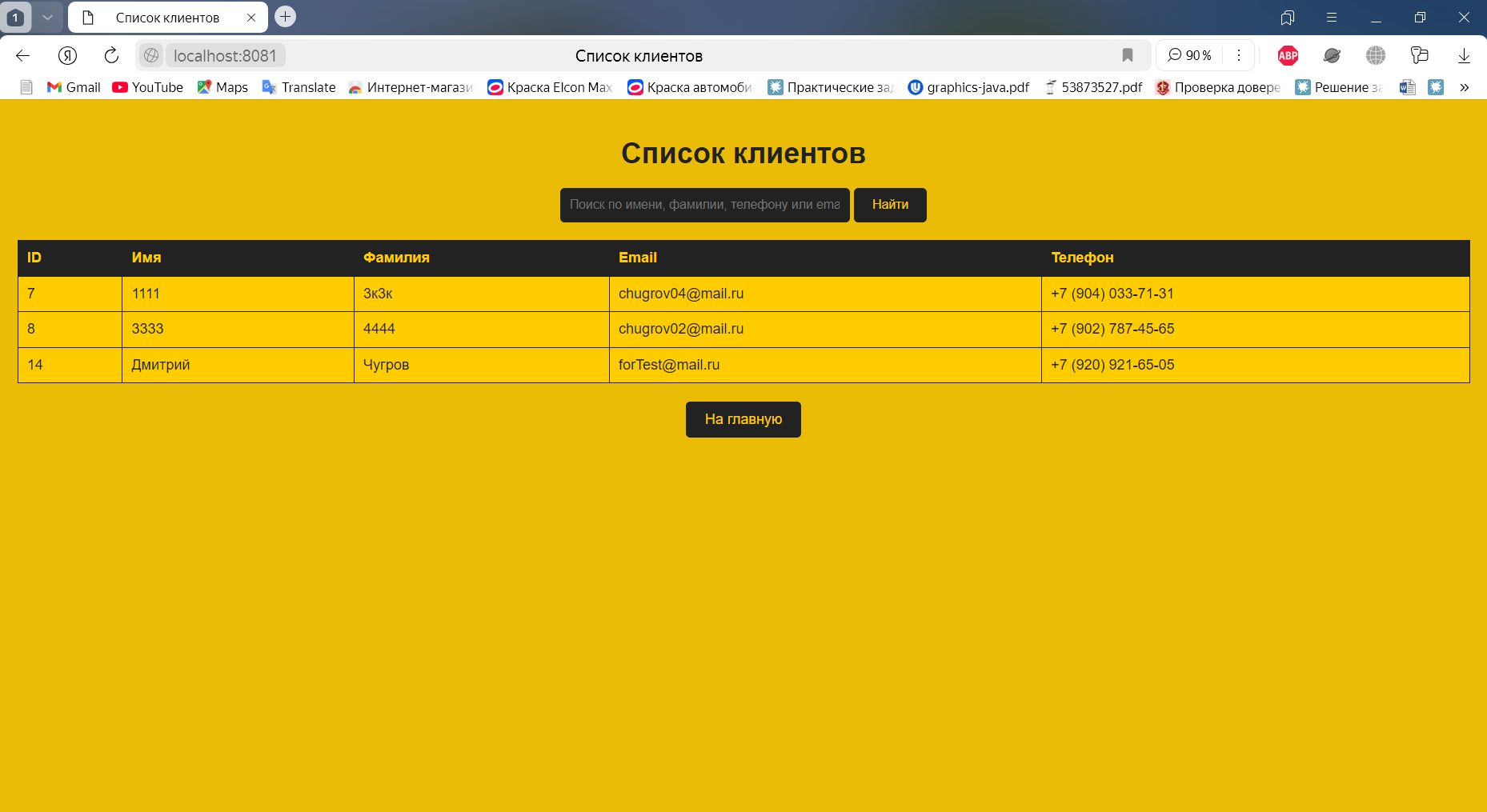


Рис. 3. Страница «Список клиентов»

*// Метод вывода страницы "Список клиентов"* @GetMapping("/manager/clients")  
 public String viewClients(@RequestParam(value = "search", required = false) String search, Model model) {  
 List<Users> clients;  
  
 if (search != null && !search.isEmpty()) {  
 clients = usersService.searchClients(search);  
 } else {  
 clients = usersService.findByRole(userRole.*CLIENT*.getRoleName());   
 }  
  
 model.addAttribute("clients", clients);  
 model.addAttribute("searchQuery", search);   
 return "managerClients";   
 }

Листинг 2. Код показа страницы «/manager/clients»

1.4.3. Таблица автомобилей клиентов

Визуализация таблицы автомобилей клиентов реализована для пользователя с ролью «Менеджер автосервиса» в виде страницы «/manager/allAutos» и для пользователя с ролью «Клиент» для вывода автомобилей только для конкретного клиента в виде страницы «/autos».

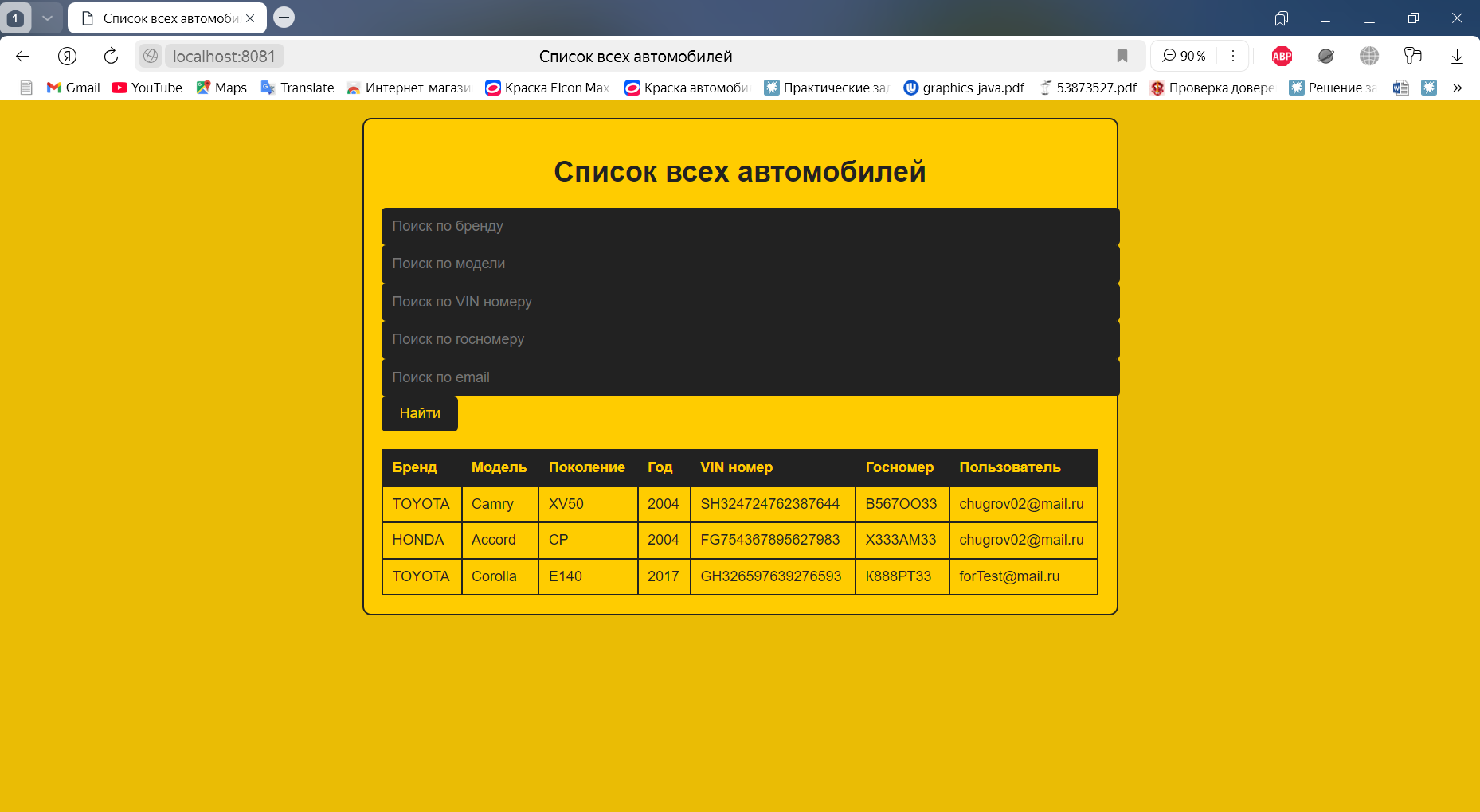


Рис. 4. Страница «Список всех автомобилей»

*// Метод вывода страницы "Список всех автомобилей"* @GetMapping("/manager/allAutos")  
 public String viewAllAutos(  
 @RequestParam(value = "brand", required = false) String brand,  
 @RequestParam(value = "model", required = false) String model,  
 @RequestParam(value = "vinNumber", required = false) String vinNumber,  
 @RequestParam(value = "gosNumber", required = false) String gosNumber,  
 @RequestParam(value = "email", required = false) String email,  
 Model modelAttribute) {  
 List<Auto> autos;  
 if (brand == null && model == null && vinNumber == null && gosNumber == null && email == null) {  
 autos = autoService.getAllAutos();  
 } else {  
 autos = autoService.searchAutos(brand, model, vinNumber, gosNumber, email);  
 }  
 modelAttribute.addAttribute("autos", autos);  
 return "allAutos";  
 }

Листинг 3. Код показа страницы «/manager/allAutos»

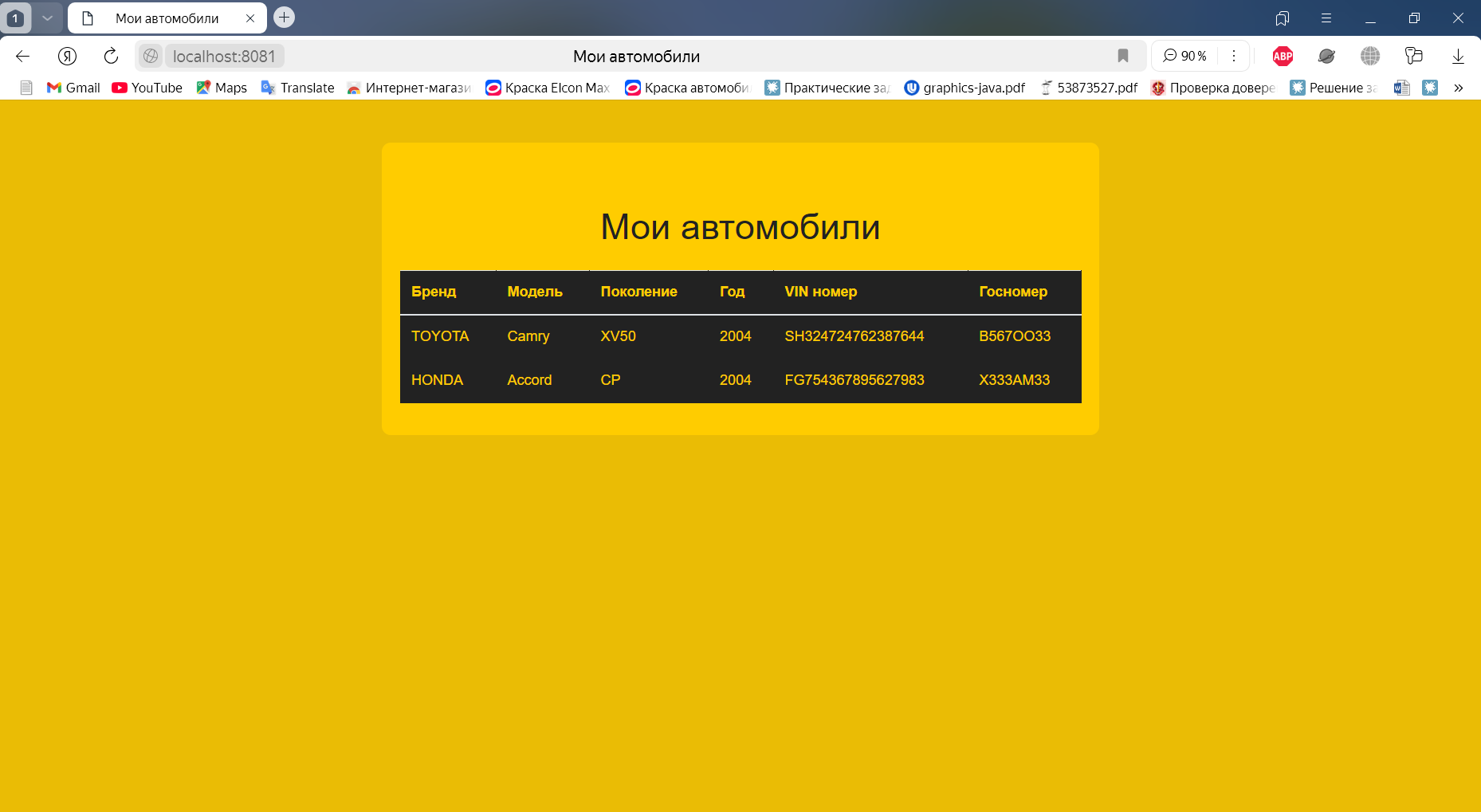


Рис. 5. Страница «Мои автомобили» для клиента

*// Метод вывода страницы "Мои автомобили"* @GetMapping("/autos")  
 public String listUserAutos(Model model, Principal principal) {  
 Users currentUser = usersService.findByEmail(principal.getName());  
  
 if (currentUser != null) {  
 List<Auto> userAutos = autoService.findByUser(currentUser);  
 model.addAttribute("autos", userAutos);  
 } else {  
 model.addAttribute("error", "Пользователь не найден.");  
 }  
  
 return "userAutos";  
 }

Листинг 4. Код показа страницы «/autos»

1.4.4. График количества оказанных услуг за месяц и процент загруженности в конкретный день

Визуализация графика количества оказанных услуг за месяц и процент загруженности в конкретный день реализована для пользователя с ролью «Менеджер автосервиса» в виде страницы «/manager/statistics». На ней менеджер вводит нужный ему месяц и его перенаправляет на страницу со статистикой.

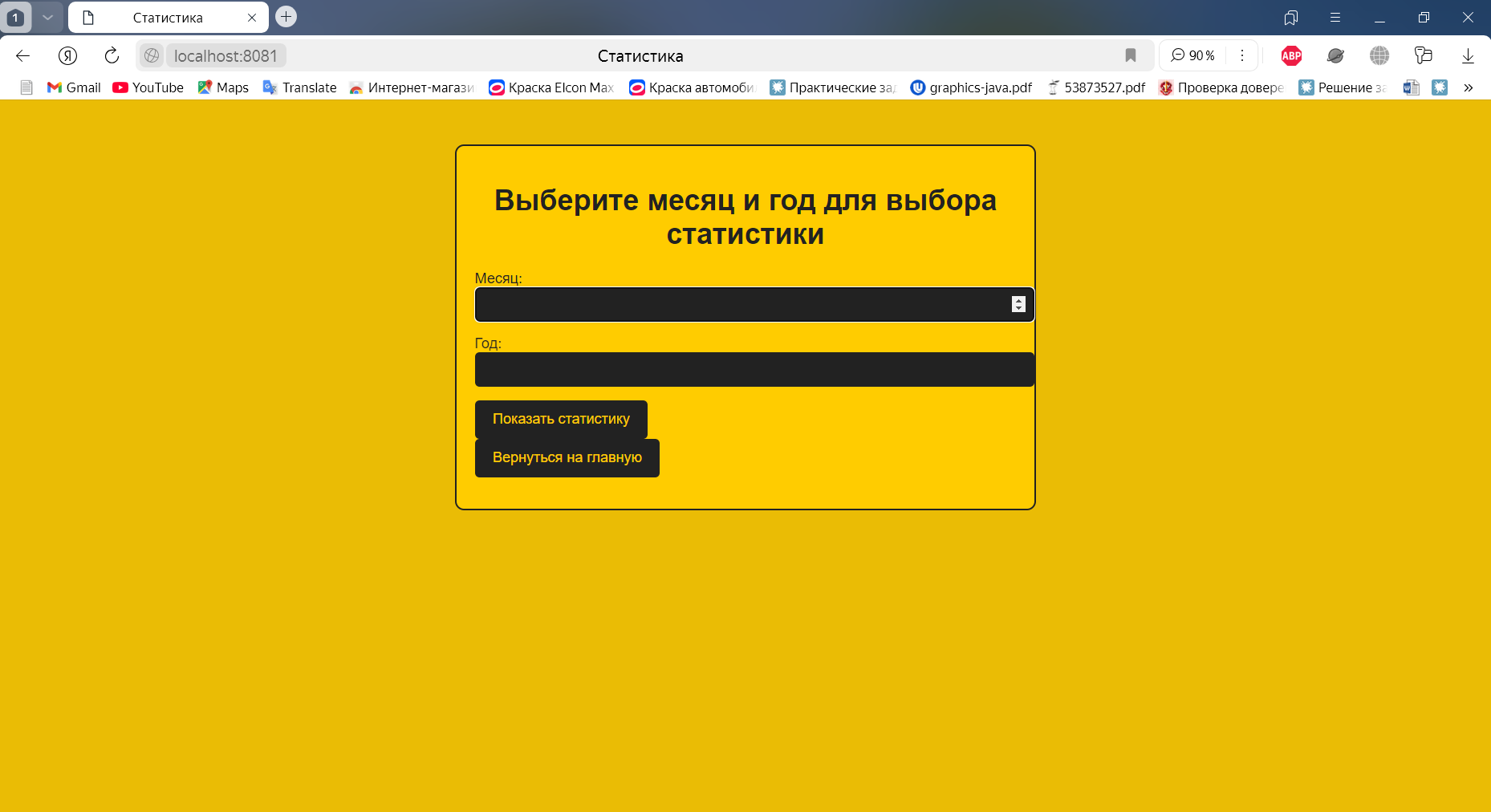


Рис. 6. Страница для выбора месяца и года для вывода статистики

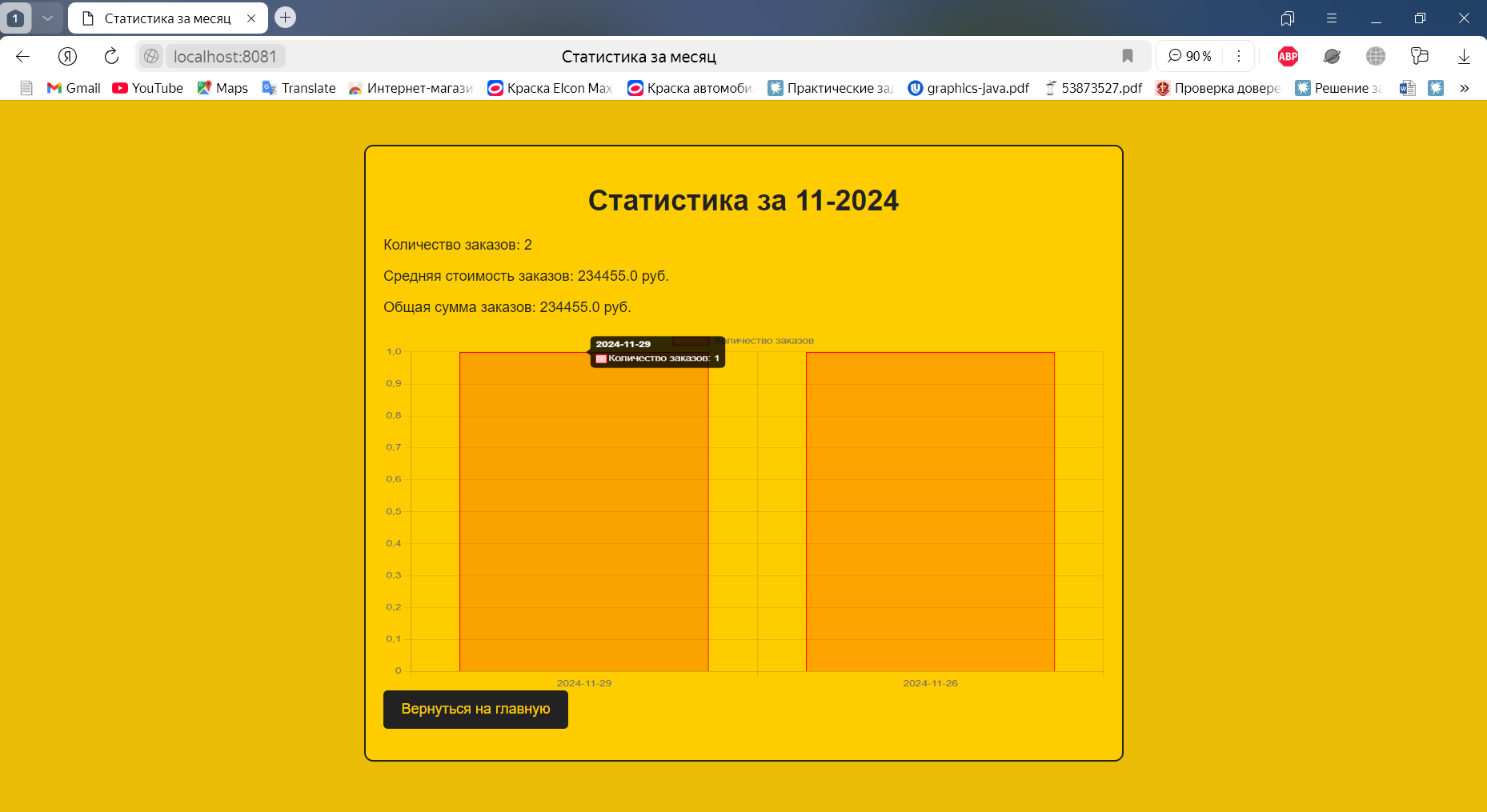


Рис. 7. Страница со статистикой

*// Методы вывода статистики* @GetMapping("/manager/statistics")  
 public String showStatisticsForm() {  
 return "managerStatsForm";  
 }  
  
 @GetMapping("/manager/statistics/results")  
 public String getMonthlyStatistics(  
 @RequestParam("month") int month,  
 @RequestParam("year") int year,  
 Model model  
 ) {  
 Map<String, Object> statistics = statisticsService.calculateMonthlyStatistics(month, year);  
 model.addAttribute("statistics", statistics);  
 model.addAttribute("month", month);  
 model.addAttribute("year", year);  
 return "managerStatistics";  
 }  
}

Листинг 5. Код показа статистики

## Подсистема поиска данных

АИСЗА обеспечивает поиск следующей информации:

* Поиск клиента (ФИ, телефон, e-mail)
* Поиск персонала (ФИ, телефон, e-mail)
* Поиск автомобиля клиента (марка, модель, год, госномер, VIN-номер, ID)
* Поиск заказа (дата, ID, статус, клиент, вид услуги)

1.5.1. Поиск клиента

Поиск клиента реализован для пользователя с ролью «Менеджер» на странице «/manager/clients» с помощью специального поля. Поиск клиента осуществляется по имени, фамилии, номеру телефона и e-mail.

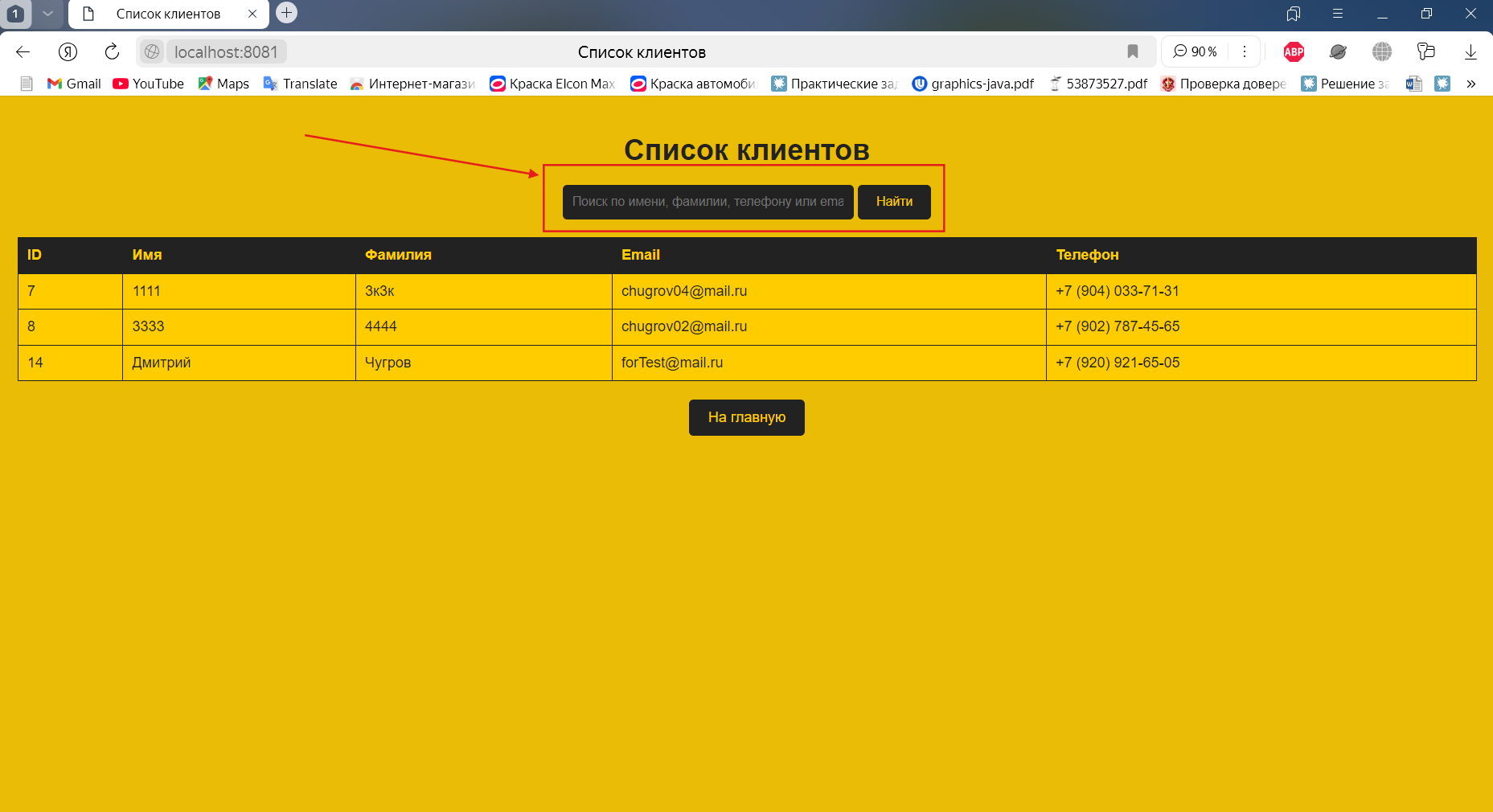


Рис. 8. Поиск клиента

1.5.2. Поиск персонала

Поиск клиента реализован для пользователя с ролью «Администратор БД» на странице «/operatorBD/personnel» с помощью специального поля. Поиск клиента осуществляется по имени, фамилии, номеру телефона и e-mail.

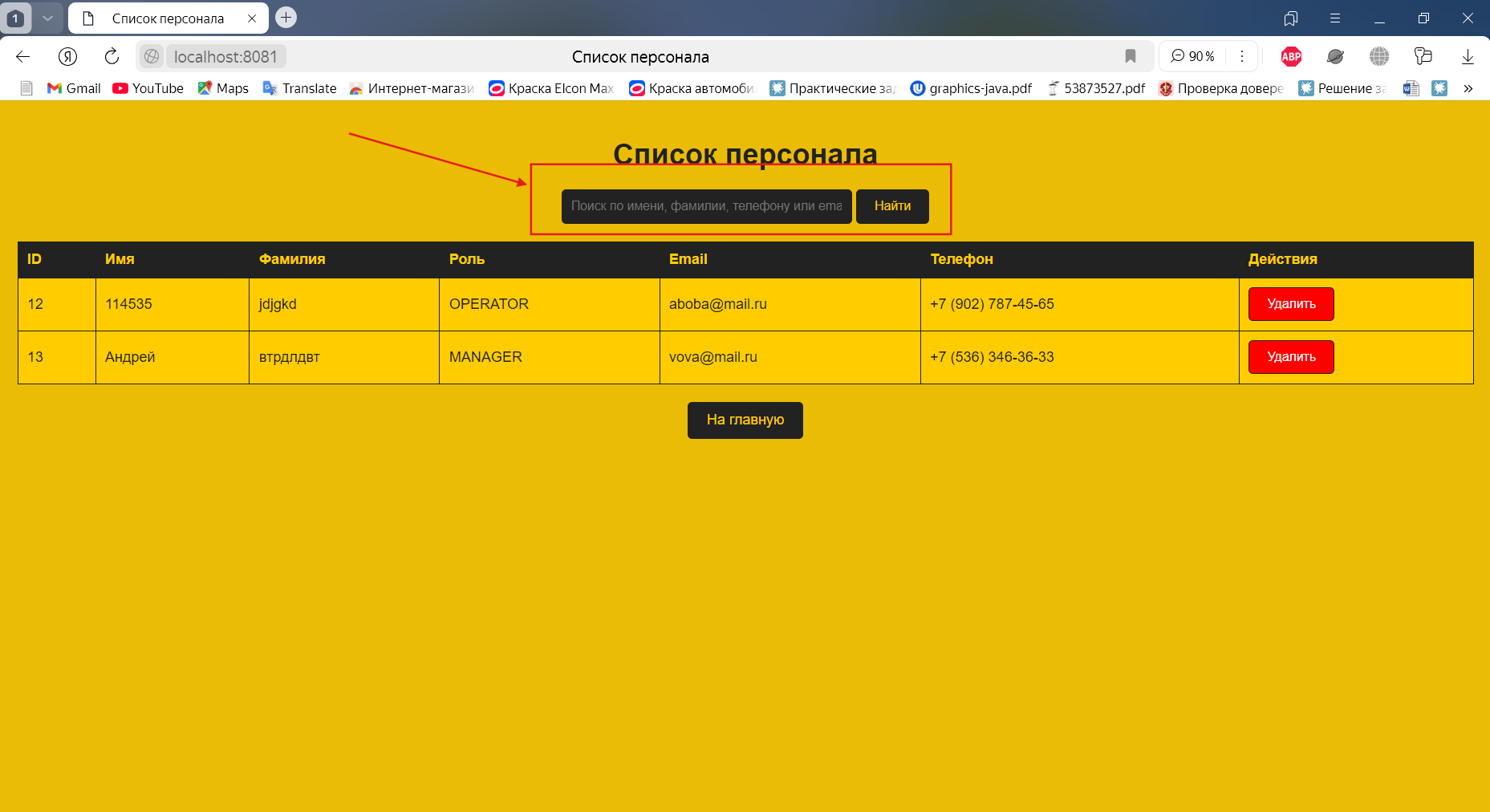


Рис. 9. Поиск персонала

1.5.3. Поиск автомобиля клиента

Поиск автомобиля клиента реализован для пользователя с ролью «Менеджер» на странице «/manager/allAutos» с помощью специального поля. Поиск автомобиля клиента осуществляется по бренду, модели, VIN номеру, госномеру автомобиля и e-mail клиента.

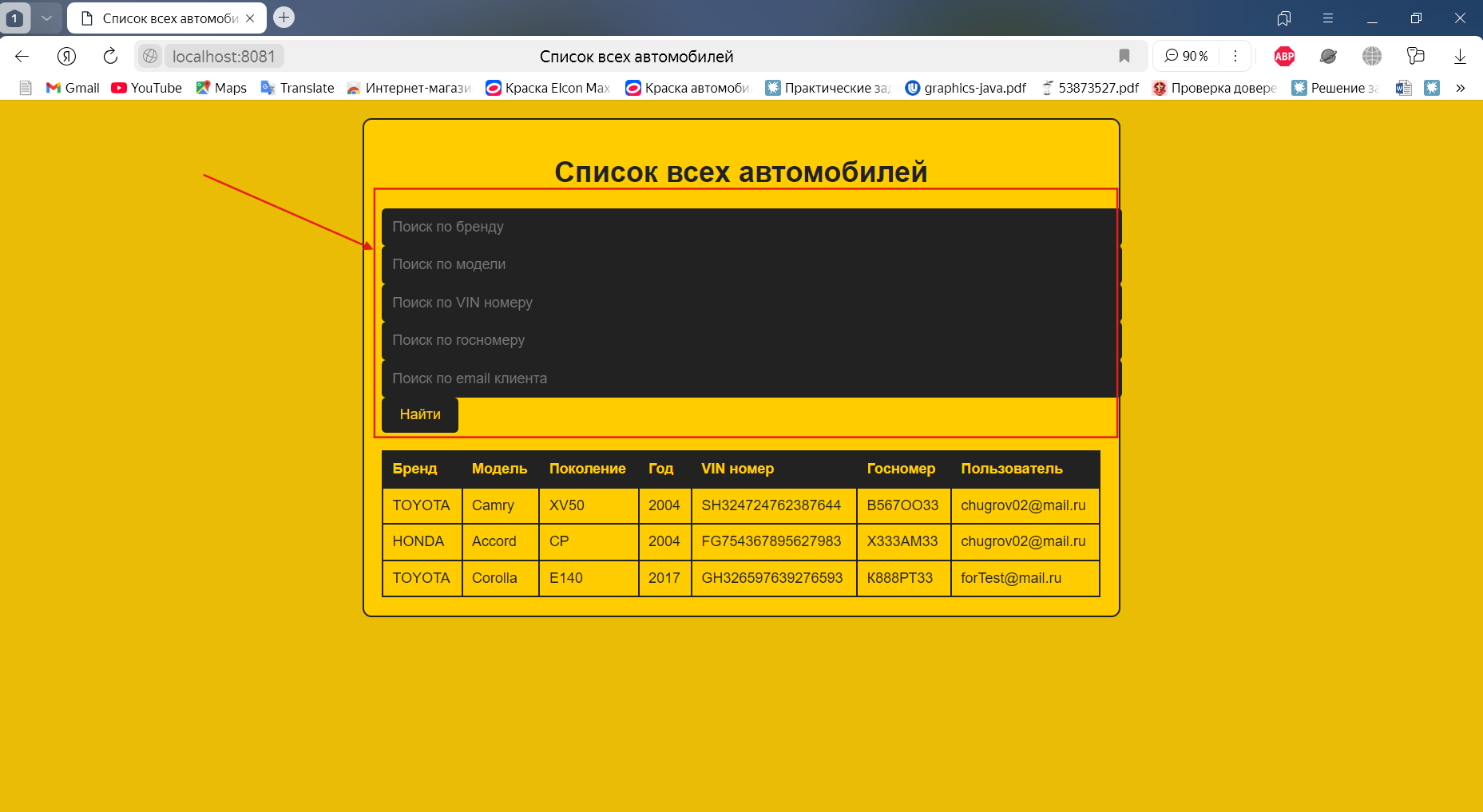


Рис. 10. Поиск клиента

1.5.4. Поиск заказа

Поиск заказа реализован для пользователя с ролью «Менеджер» на странице «/manager/orders» с помощью специального поля. Поиск заказа осуществляется по ID заказа, e-mail клиента, бренду, модели, году, VIN номеру, госномеру, статусу, виду услуги и дате заказа.

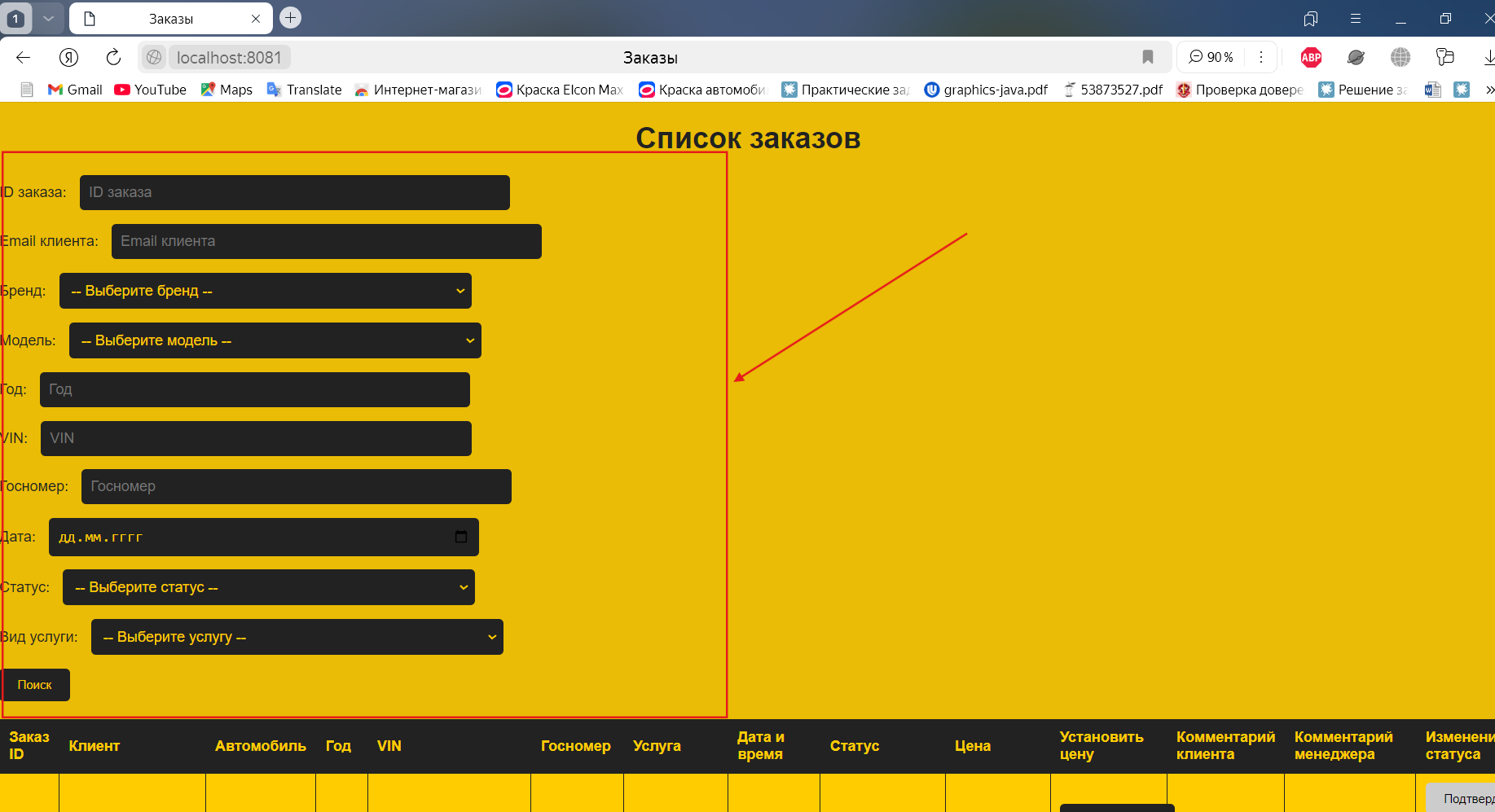


Рис. 11. Поиск заказа

## Подсистема управления данными

АИСЗА реализует управление объектами / данными:

* Управление клиентами (регистрация, редактирование данных клиента);
* Управление персоналом (регистрация и удаление персонала);
* Управление заказами (создание, изменение, присвоение статуса заказу).

1.6.1. Управление клиентами

Управление клиентами представлено в виде регистрации клиента, редактирование и удаление его данных.

1.6.1.1. Регистрация клиента

Регистрация клиент возможна с помощью страницы «/register». Все введенные данные проверяются на корректность используя паттерны.

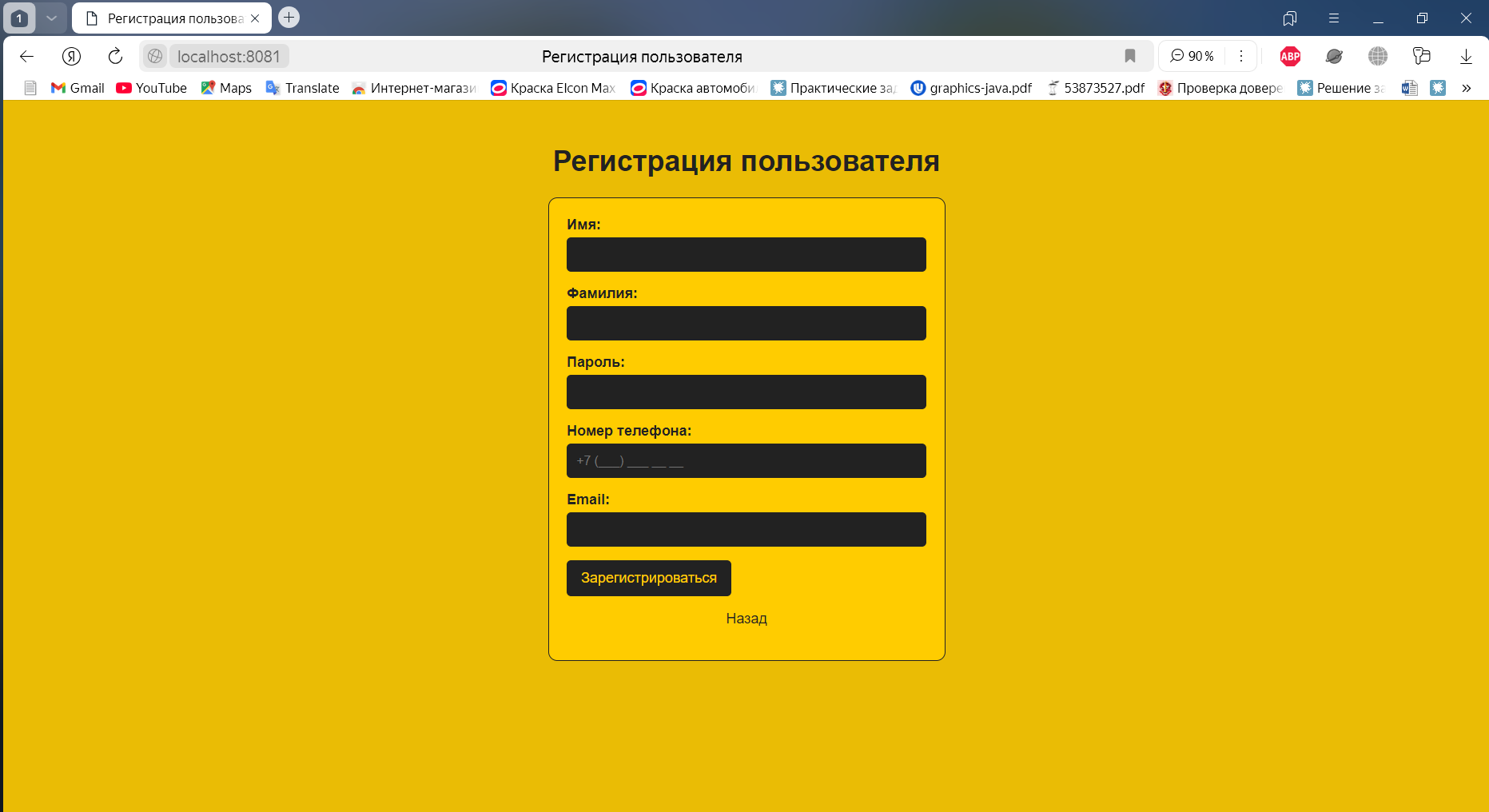


Рис. 12. Регистрация клиента

*// Методы для регистрации пользователя* @GetMapping("/register")  
 public String register(Model model) {  
 model.addAttribute("Users", new Users());  
 return "register";  
 }  
 @PostMapping("/register")  
 public String registerUser(@Valid @ModelAttribute("Users") Users users,  
 BindingResult result, Model model) {  
 if (result.hasErrors()) {  
 return "register";  
 }  
 if (!usersService.registerUser(users)) {  
 model.addAttribute("error", "Пользователь с таким email уже существует");  
 return "register";  
 }  
 return "confirmation";  
 }

Листинг 6. Код для регистрации пользователя

1.6.1.2. Редактирование данных

Редактирование данных клиент возможно с помощью кнопки в главном меню. При ее нажатии происходит перенаправление на страницу редактирования данных «/user/edit». Для удобства, текущие данные уже заполнены в поля ввода.

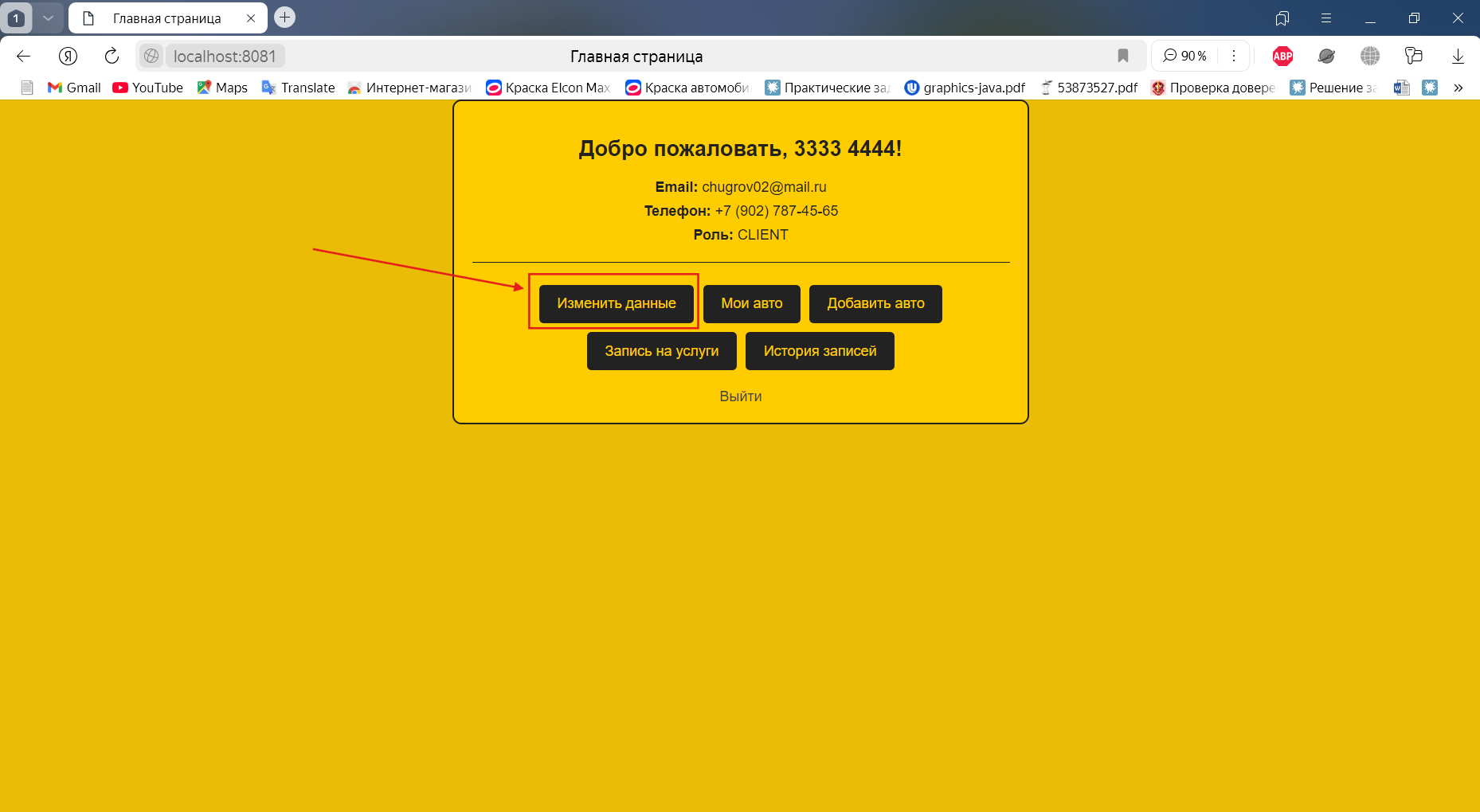


Рис. 13. Кнопка «Изменить данные» в главном меню

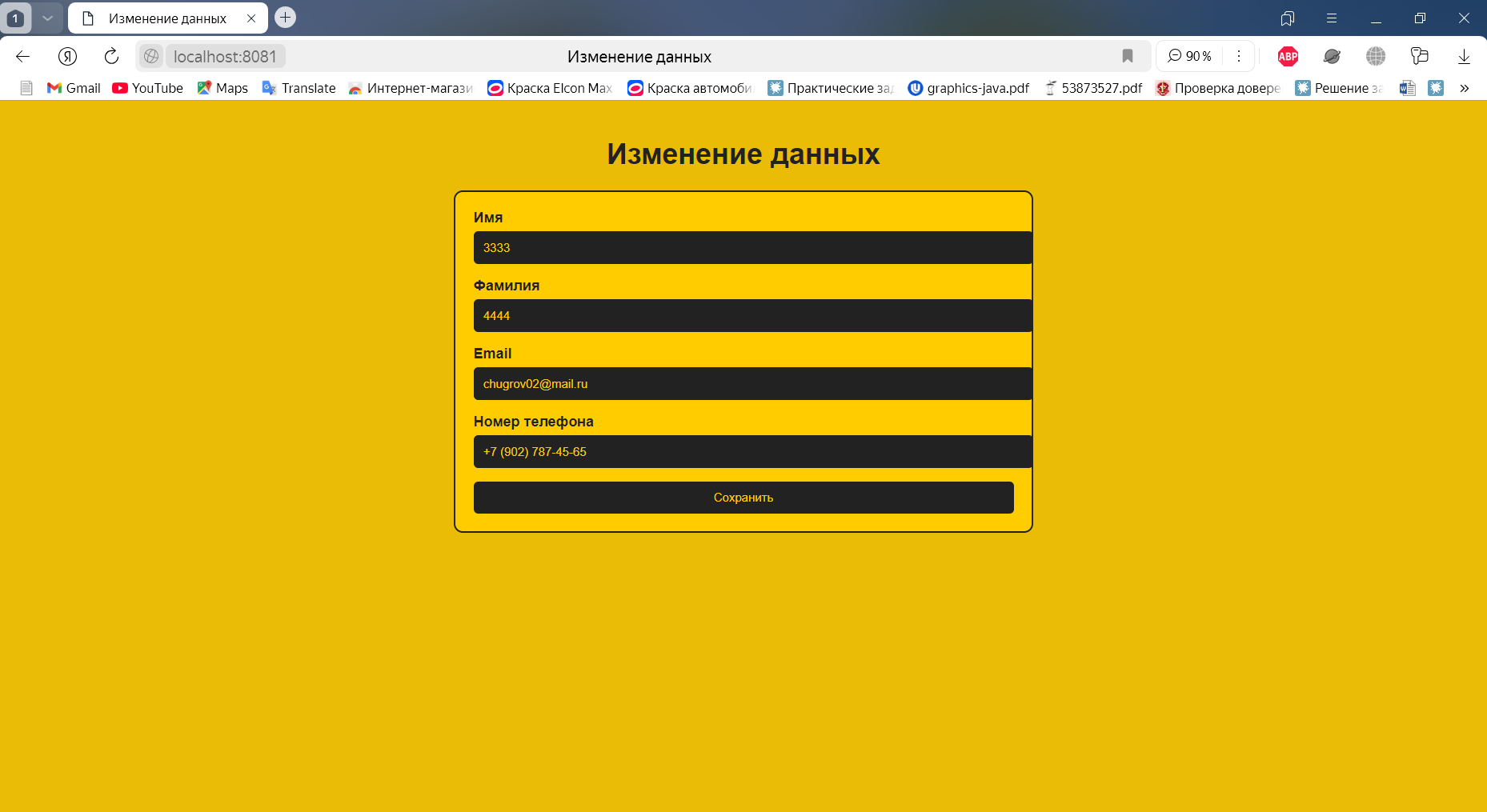


Рис. 14. Страница изменения данных

*// Методы для редактирования данных клиента* @GetMapping("/edit")  
 public String editUserForm(Principal principal, Model model) {  
 Users user = userService.findByEmail(principal.getName());  
 model.addAttribute("user", user);  
 return "edit\_user";  
 }  
  
 @PostMapping("/edit")  
 public String updateUser(  
 @ModelAttribute("user") @Valid Users user,  
 BindingResult bindingResult,  
 Principal principal,  
 RedirectAttributes redirectAttributes) {  
 if (bindingResult.hasErrors()) {  
 return "edit\_user";  
 }  
 userService.updateUser(principal.getName(), user);  
 redirectAttributes.addFlashAttribute("successMessage", "Данные успешно обновлены!");  
 return "redirect:/home";  
 }

Листинг 7. Код для редактирования данных клиента

1.6.2. Управление персоналом

Регистрация менеджера в системе и удаления возможна только «Администратором БД» в соответствии разграничением доступа с помощью страниц «» и «/operatorBD/personnel» соответственно.

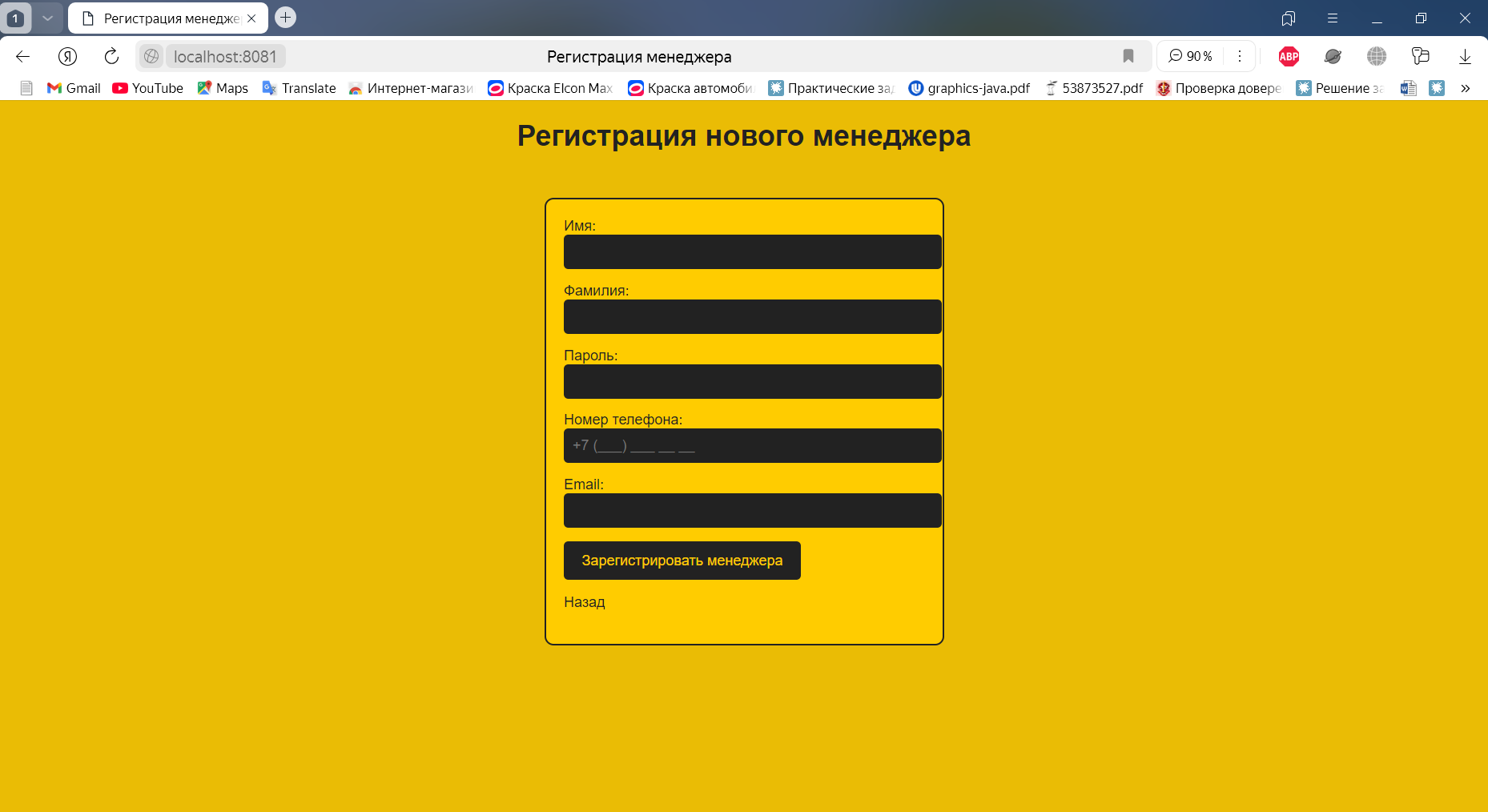


Рис. 15. Страница регистрации менеджера в системе

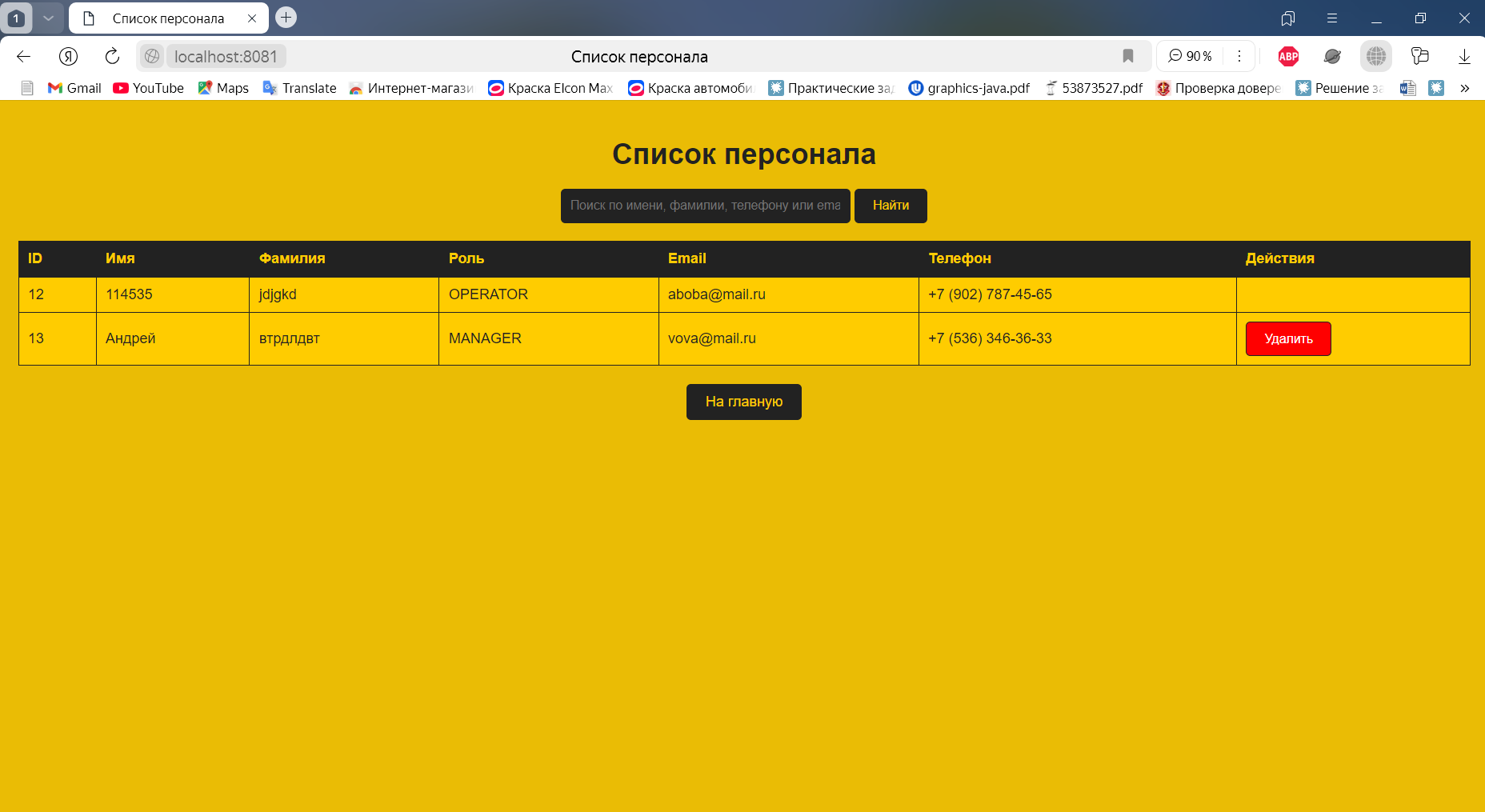


Рис. 16. Страница удаления менеджера из системы

*// Метод логики удаления персонала* @PostMapping("/deleteClient")  
 public String deleteClient(@RequestParam("id") Long id, RedirectAttributes redirectAttributes) {  
 try {  
 usersService.deleteUserById(id);  
 redirectAttributes.addFlashAttribute("successMessage", "Пользователь успешно удален.");  
 } catch (Exception e) {  
 redirectAttributes.addFlashAttribute("errorMessage", "Ошибка при удалении пользователя.");  
 }  
 return "redirect:/operatorBD/personnel";  
 }

Листинг 8. Код для редактирования данных клиента

1.6.3. Управление заказами

Управление заказами представлено следующими функциями: создание заказа клиентом, изменение и присвоение статуса заказу менеджером.

1.6.3.1. Создание заказа клиентом

Для создания заказа клиентом на главной странице существует кнопка «Запись на услуги» (рис. 17). При нажатии происходит перенаправление на страницу выбора авто клиента и вид услуги (рис. 18). Далее пользователя перекидывает на страницу выбора доступной даты и времени (рис. 19). После клиента просят подтвердить данные (рис. 20).

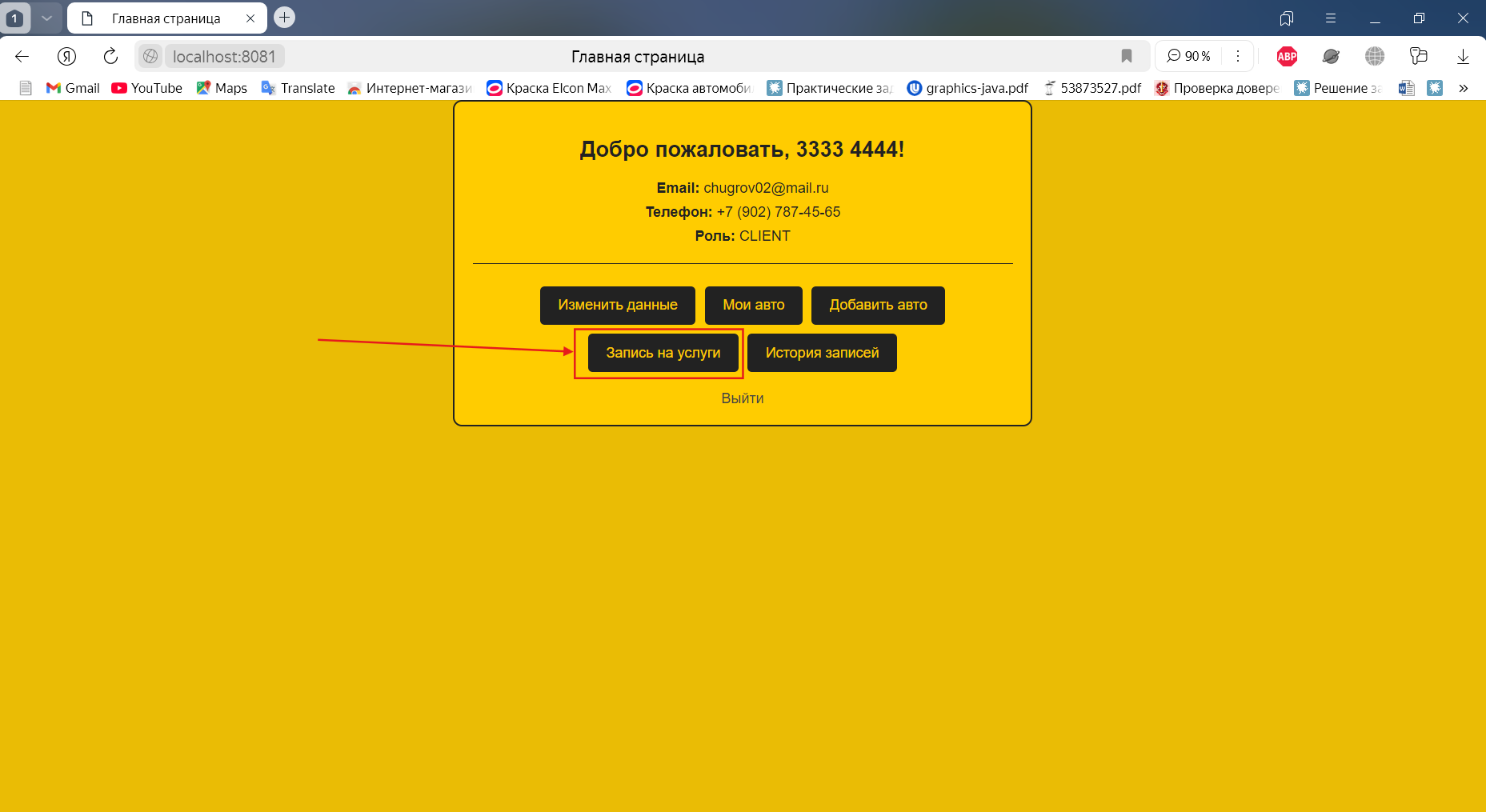


Рис. 17. Кнопка «Запись на услуги» на главном меню

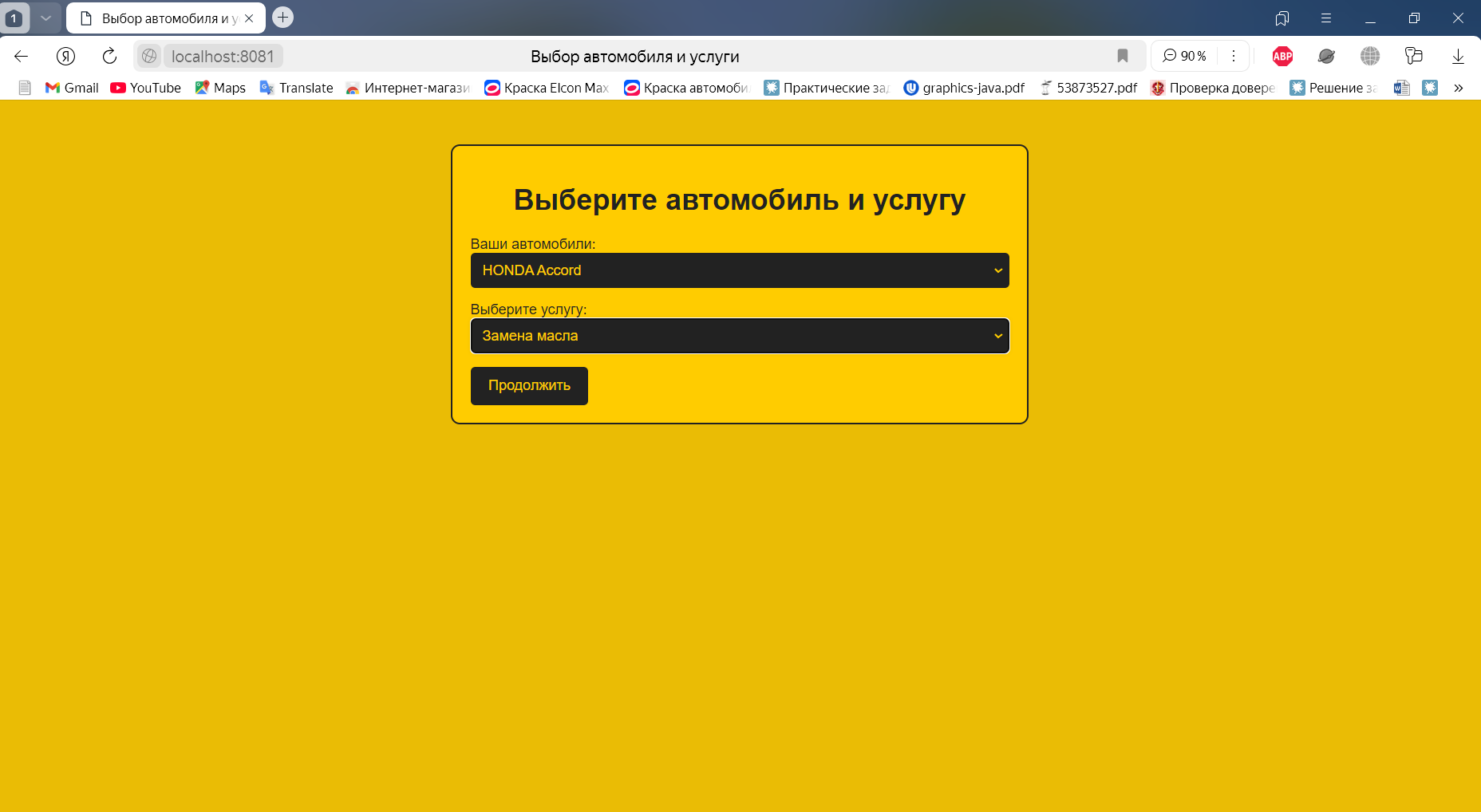


Рис. 18. Страница выбора автомобиля клиента и вида услуги

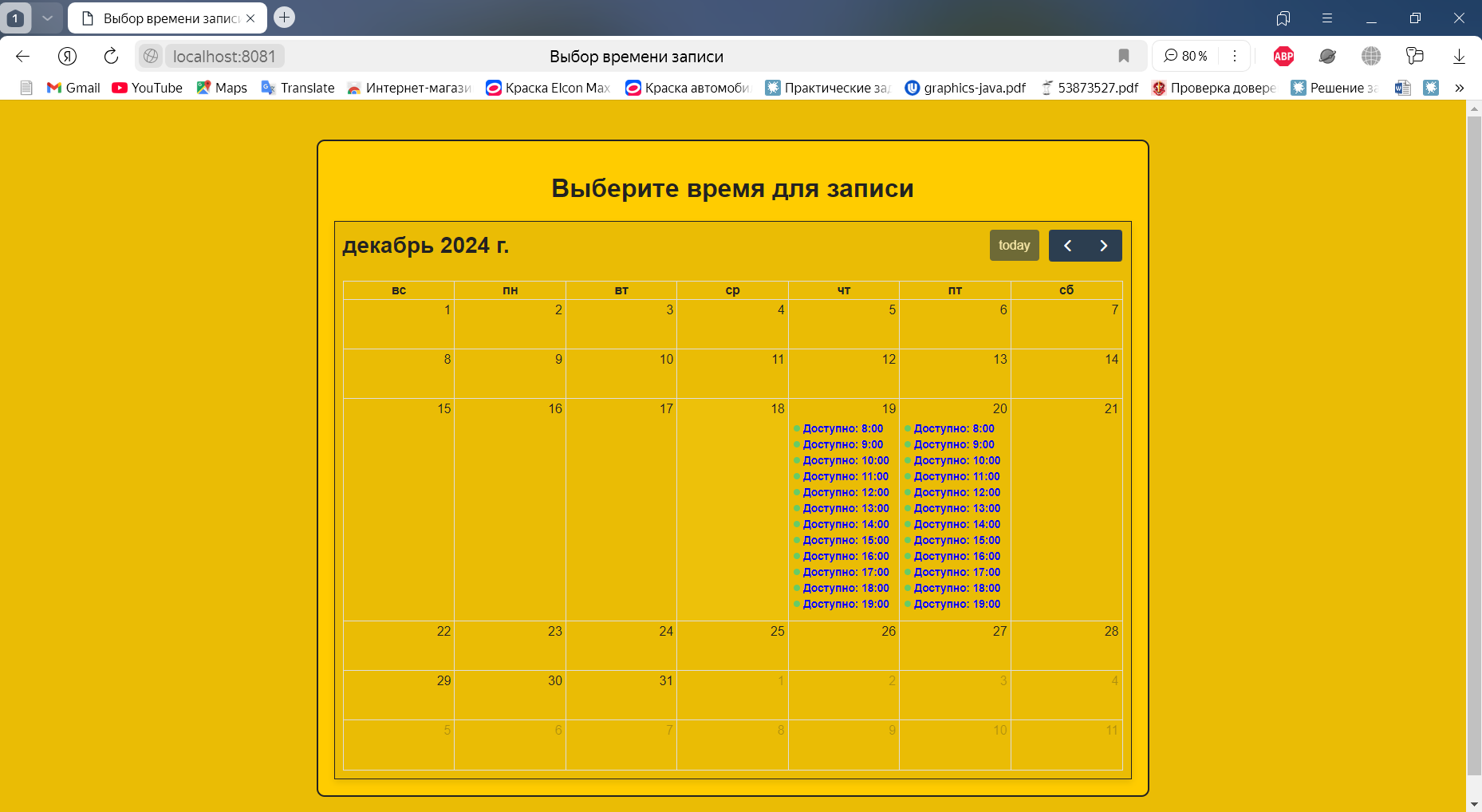


Рис. 19. Страница с выбором даты и времени записи

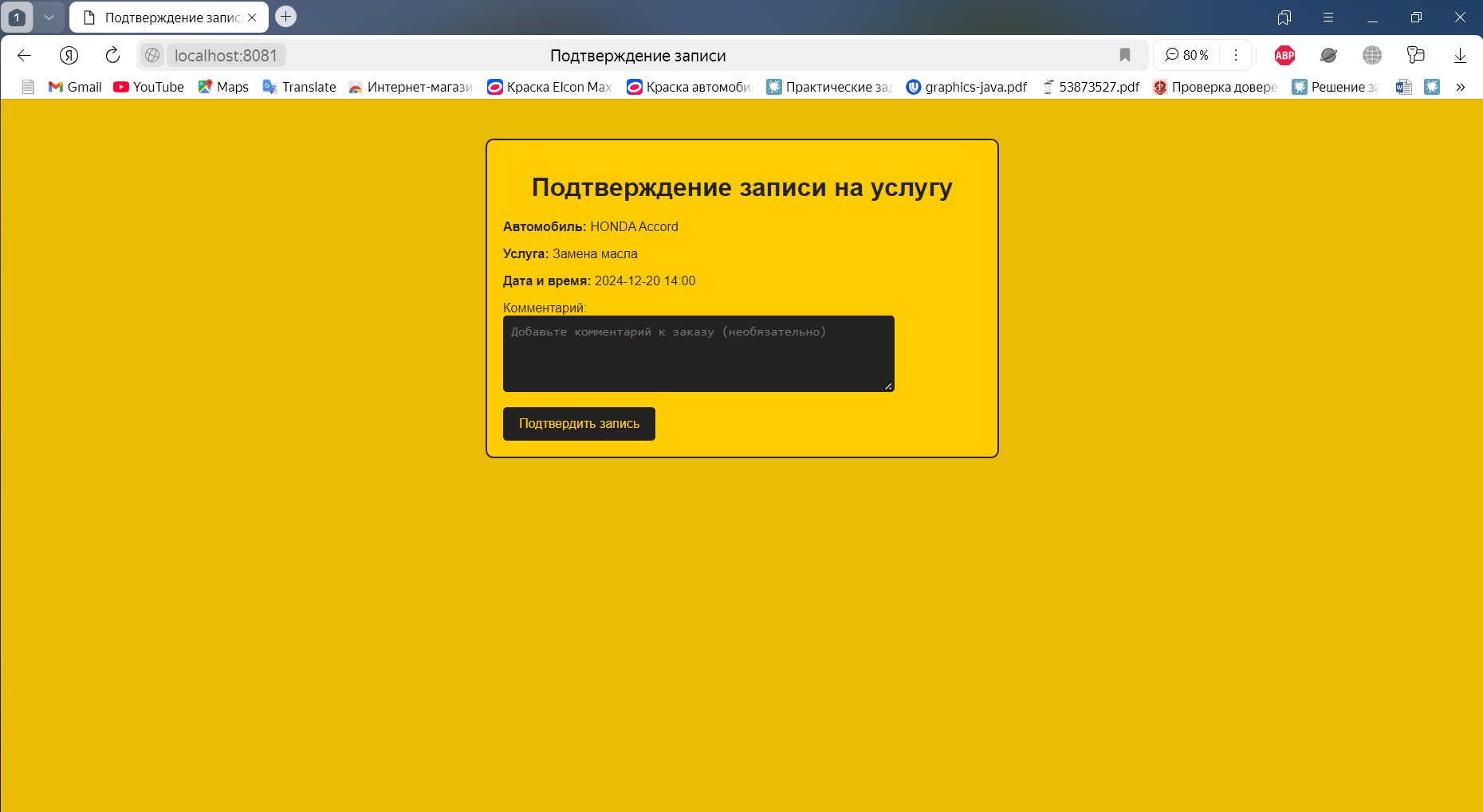


Рис. 20. Страница подтверждения данных

1.6.3.2. Изменение и присвоение статуса заказу менеджером

Изменение и присвоение статуса заказа менеджером происходит на странице «/manager/orders». Менеджер может подтвердить заказ, задать статус «На исполнении» и «Исполнен».

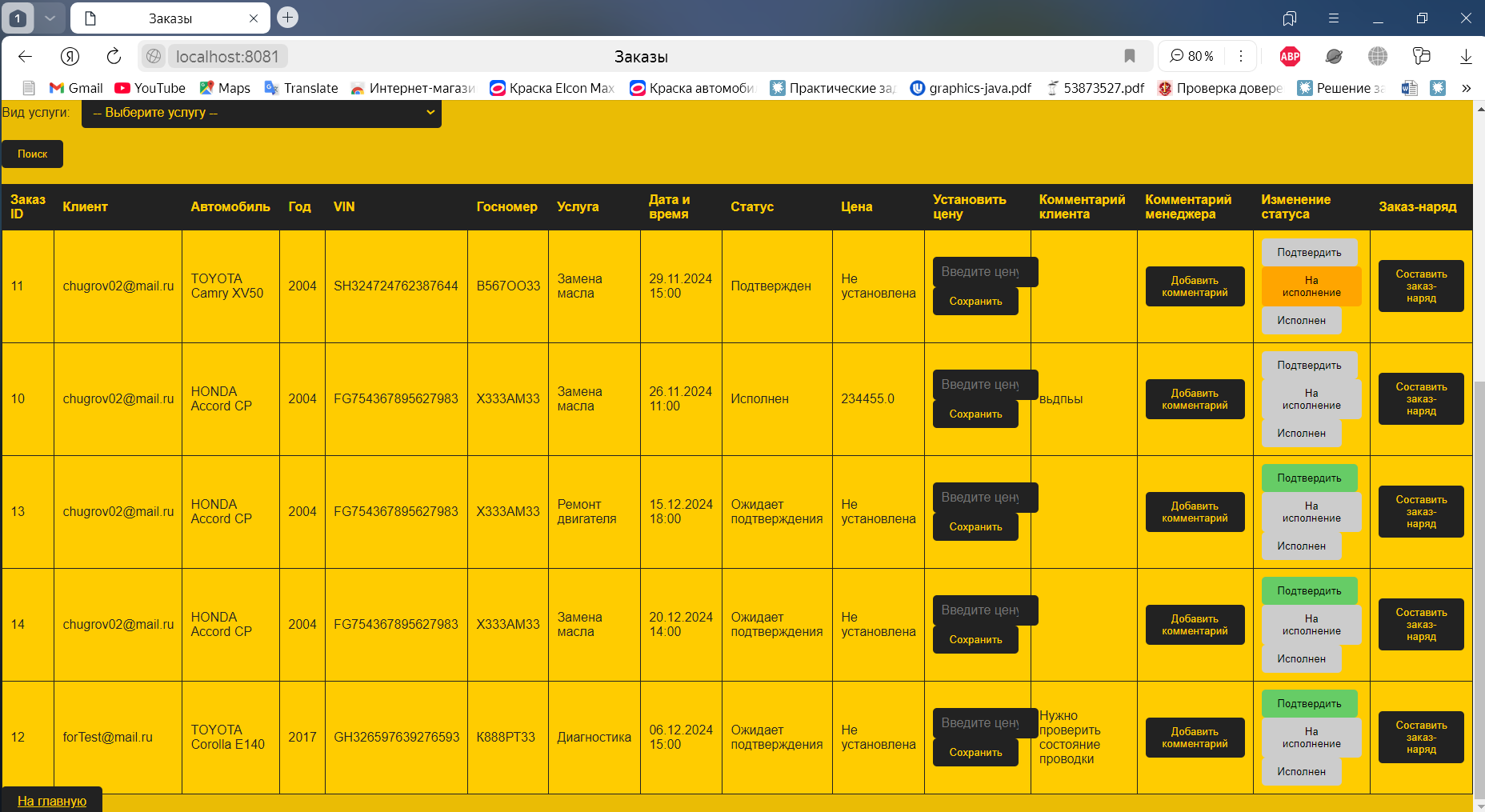


Рис. 21. Страница изменения и присвоения статуса заказа

## Подсистема формирования отчетов и заказ-нарядов

Формирование отчетов и заказ-нарядов доступно только пользователям с ролью «Менеджер».

1.7.1. Формирование отчетов

Эта функция реализована в виде графика количества оказанных услуг за месяц и процент загруженности в конкретный день реализована для пользователя с ролью «Менеджер автосервиса» в виде страницы «/manager/statistics». На ней менеджер вводит нужный ему месяц и его перенаправляет на страницу со статистикой.

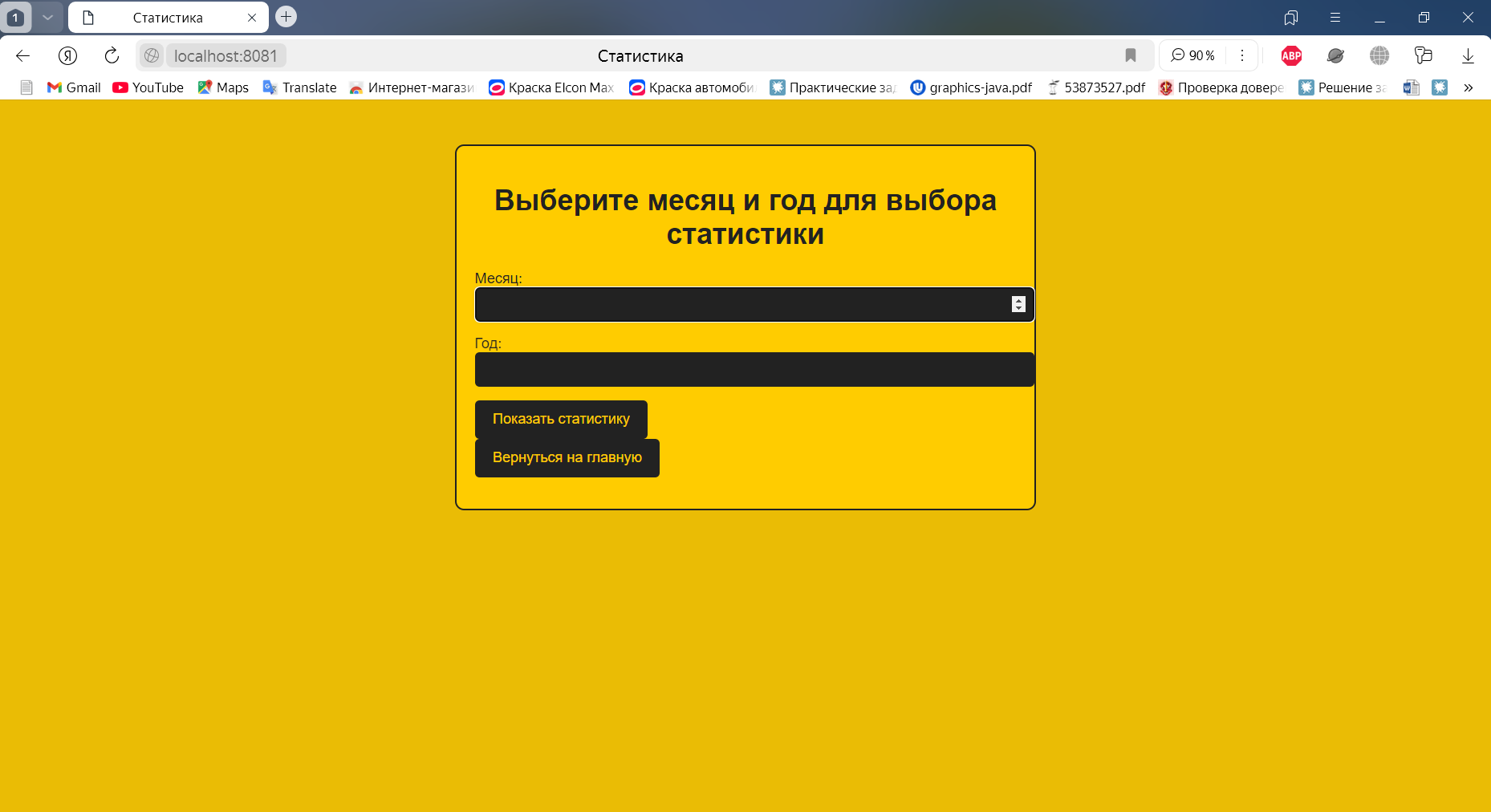


Рис. 22. Страница для выбора месяца и года для вывода статистики

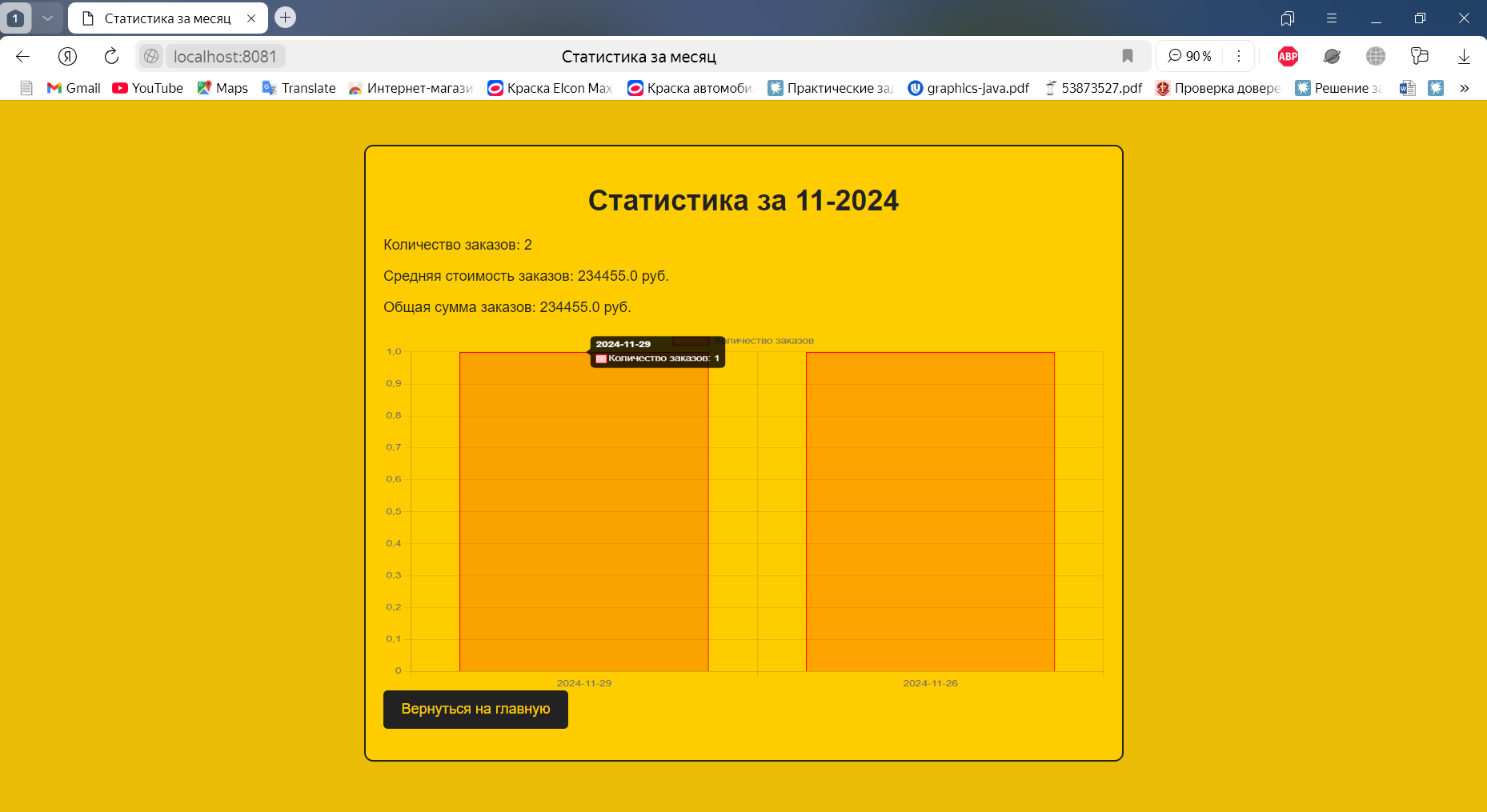


Рис. 23. Страница со статистикой

1.7.2. Формирование заказ-нарядов

Формирование заказ-нарядов реализовано на странице «» с помощью кнопки «Составить заказ-наряд». В результате будет сформирован файл .docx с основной информацией о заказе.

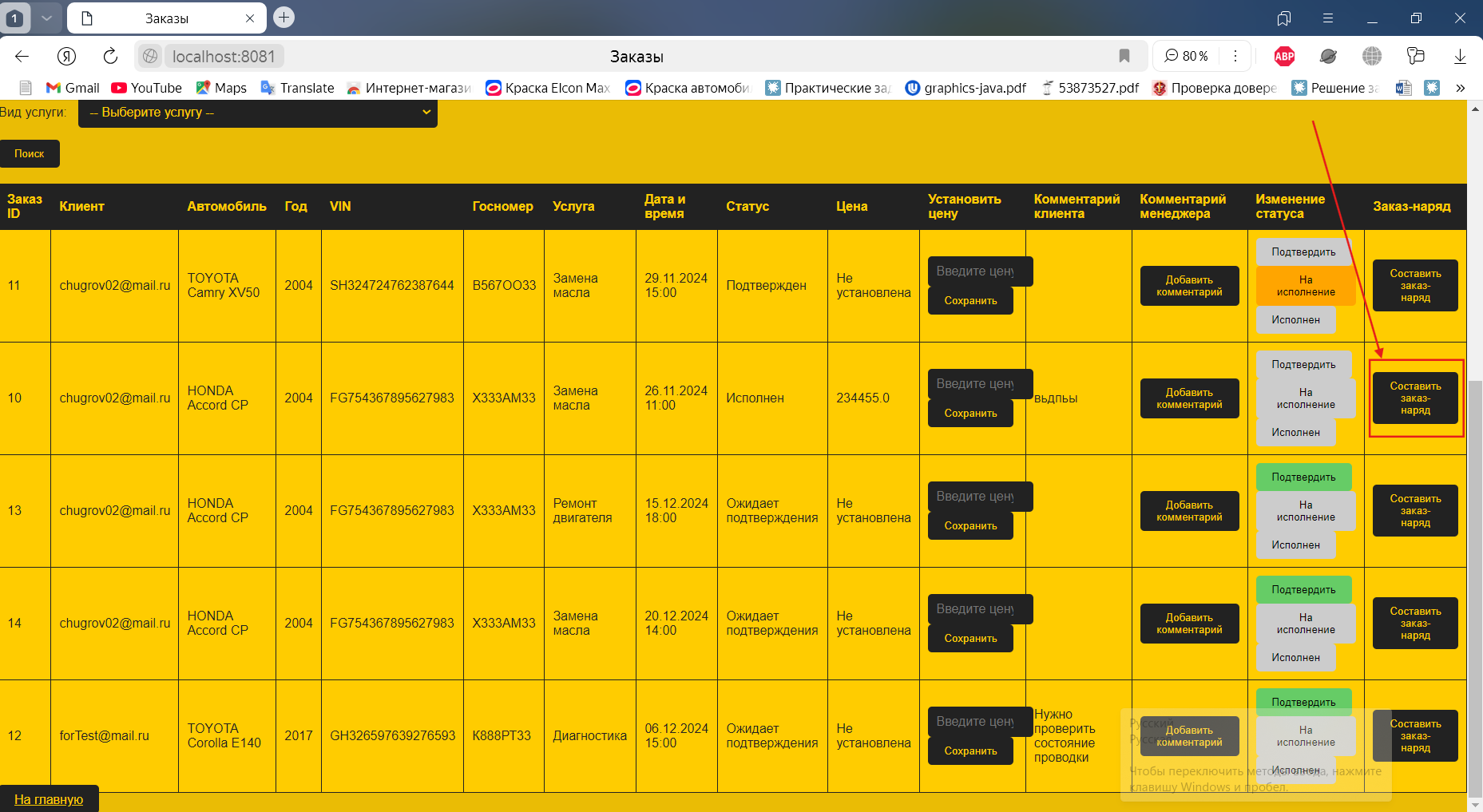


Рис. 23. Кнопка «Сформировать заказ-наряд»

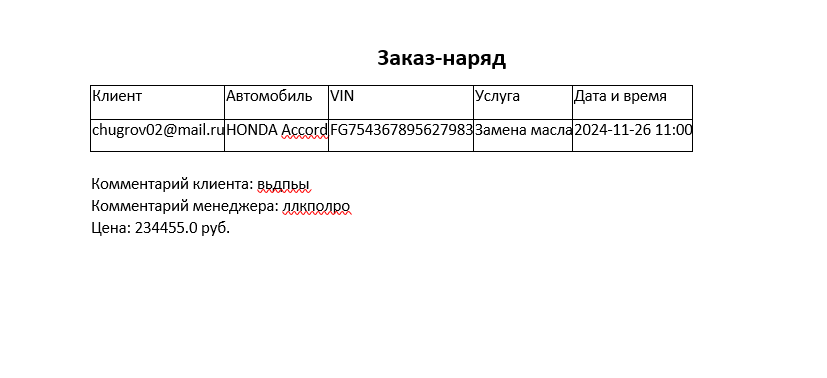


Рис. 24. Пример заказ-наряда

*// Метод логики формирования заказ-наряда* @PostMapping("/manager/orders/exportDocx")  
 public ResponseEntity<?> generateOrderDocx(@RequestParam Long orderId) {  
 try {  
 Order order = orderService.getOrderById(orderId);  
 if (order == null) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(null);  
 }  
  
 ByteArrayOutputStream out = new ByteArrayOutputStream();  
  
 try (XWPFDocument document = new XWPFDocument()) {  
 XWPFParagraph title = document.createParagraph();  
 title.setAlignment(ParagraphAlignment.*CENTER*);  
 XWPFRun titleRun = title.createRun();  
 titleRun.setText("Заказ-наряд");  
 titleRun.setBold(true);  
 titleRun.setFontSize(16);  
  
 XWPFTable table = document.createTable();  
  
 XWPFTableRow headerRow = table.getRow(0);  
 headerRow.getCell(0).setText("Клиент");  
 headerRow.addNewTableCell().setText("Автомобиль");  
 headerRow.addNewTableCell().setText("VIN");  
 headerRow.addNewTableCell().setText("Услуга");  
 headerRow.addNewTableCell().setText("Дата и время");  
  
 XWPFTableRow valueRow = table.createRow();  
 valueRow.getCell(0).setText(order.getClient().getEmail());  
 valueRow.getCell(1).setText(order.getAuto().getBrand() + " " + order.getAuto().getModel());  
 valueRow.getCell(2).setText(order.getAuto().getVinNumber());  
 valueRow.getCell(3).setText(order.getServiceType());  
 valueRow.getCell(4).setText(order.getSlot().getDate() + " " + order.getSlot().getTime());  
  
 XWPFParagraph commentsParagraph = document.createParagraph();  
 XWPFRun commentsRun = commentsParagraph.createRun();  
 commentsRun.addBreak();  
 commentsRun.setText("Комментарий клиента: " + (order.getClientComment() != null ? order.getClientComment() : "Нет"));  
 commentsRun.addBreak();  
 commentsRun.setText("Комментарий менеджера: " + (order.getManagerComment() != null ? order.getManagerComment() : "Нет"));  
 commentsRun.addBreak();  
 commentsRun.setText("Цена: " + (order.getPrice() != null ? order.getPrice() : "0") + " руб." );  
 document.write(out);  
 }  
  
 ByteArrayInputStream in = new ByteArrayInputStream(out.toByteArray());  
 HttpHeaders headers = new HttpHeaders();  
 headers.add("Content-Disposition", "attachment; filename=order\_" + orderId + ".docx");  
  
 return ResponseEntity  
 .*ok*()  
 .headers(headers)  
 .contentType(MediaType.*APPLICATION\_OCTET\_STREAM*)  
 .body(new InputStreamResource(in));  
 } catch (IOException e) {  
 e.printStackTrace();  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(null);  
 }  
 }

Листинг 9. Код для формирования заказ-наряда

## Взаимодействие между подсистемами

Взаимодействие подсистем реализовано через Dependency Injection (DI) с помощью Spring Framework. Благодаря этому подсистемы взаимодействуют через сервисы, не завися напрямую от их реализации. Соответственно контроллеры выступают как интерфейс для взаимодействия с подсистемами, позволяя вызывать методы через HTTP-запросы.

## Функционирование системы в различных режимах

В основном режиме система осуществляет свой функционал беспрепятственно для пользователя.

Особый режим применяется при отказе работы части системы или для модификации (отладки) программного кода АИСЗА. В этом случае пользователь перенаправляется на страницу «/error».

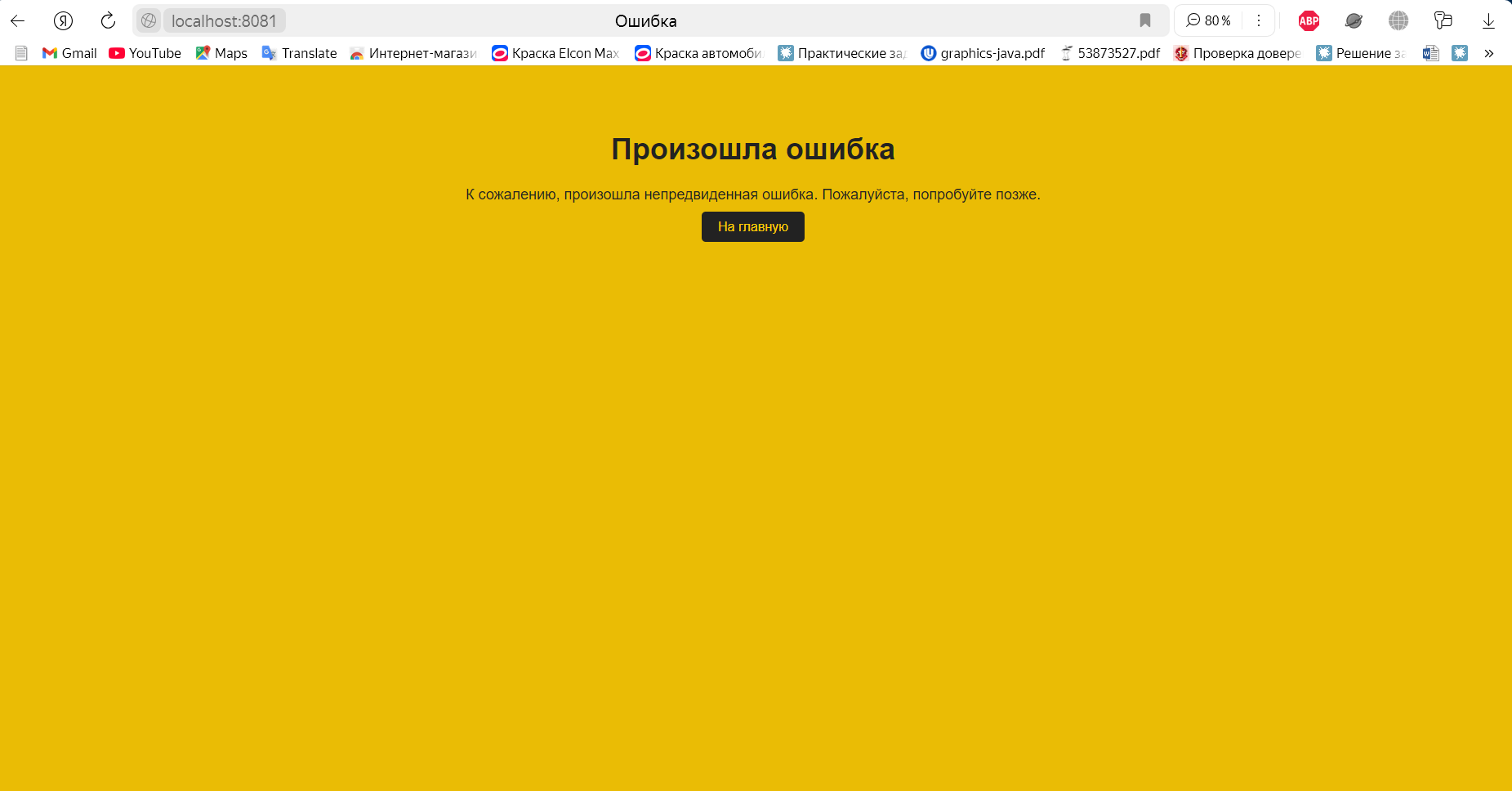


Рис. 25. Вид страницы /error

server.error.whitelabel.enabled=false  
server.error.path=/error

Листинг 10. Настройка показа страницы «/error»

## Реализация требований защиты информации

В рамках разработки АИСЗА предусмотрено хранение конфиденциальной и коммерческой информации.

Компоненты подсистемы защиты информации обеспечивают:

* идентификацию и авторизацию пользователя;
* шифрование данных пользователя;
* шифрование данных персонала;
* шифрование коммерческой информации автосервиса (данные о прибыли, количество клиентов в месяц и т.п.)

Для разграничения функционала по ролям при авторизации используется соответствующий код:

*// Метод для разграничения функционала при авторизации* @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/login", "/register").permitAll()  
 .requestMatchers("/client/\*\*").hasRole("CLIENT")  
 .requestMatchers("/operatorBD/\*\*").hasRole("OPERATOR")  
 .requestMatchers("/manager/\*\*").hasRole("MANAGER")  
 .anyRequest().authenticated()  
 )  
 .formLogin(form -> form  
 .loginPage("/login")  
 .defaultSuccessUrl("/home", true)  
 .permitAll()  
 )  
 .logout(logout -> logout  
 .permitAll())  
 .csrf(csrf -> csrf.disable());  
  
 return http.build();  
 }

Листинг 11. Код проверки пользователя на соответствующую роль при входе в систему

# ЗАКЛЮЧЕНИЕ

В результате курсовой работы получен опыт проектирования и реализации информационных систем в сфере записей в автосервис. Итогом стала разработанная автоматизированная информационная система записей в автосервис, отвечающая требованиям соответствия функционала, надежности, доступности и безопасности.

Разработанная система имеет следующие особенности:

- авторизация пользователей и разделение функционала в соответствии с ролью;

- визуализация графика статистики в реальном времени;

- составление заказ-нарядов в формате документов .docx.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Роберт Седжвик. Алгоритмы на Java [Текст] / Роберт Седжвик, Кевин Уэйн — 4-е издание. — Москва: Издательский дом "Вильямс", 2013 — 848 c.

2. Гетц Брайан. Java Concurrency на практике [Текст] / Гетц Брайан, Пайерлс Тим, Блох Джошуа, Боубер Джозеф, Холмс Дэвид, Ли Даг — Санкт-Петербург: ООО Издательство "Питер", 2020 — 464 c.

3. Брюс Шнайер Прикладная криптография: Протоколы, алгоритмы, исходные тексты на языке Си [Текст] / Брюс Шнайер — 2-е издание. — Москва: ООО Издательство "Триумф", 2002 — 1048 c.

4. Хеклер Марк Spring Boot по-быстрому [Текст] / Хеклер Марк. — Санкт-Петербург: ООО Издательство "Питер", 2022 — 352 c.

5. Гутьеррес Фелипе Spring Boot 2: лучшие практики для профессионалов [Текст] / Гутьеррес Фелипе. — Санкт-Петербург: ООО Издательство "Питер", 2020 — 462 c.

6. Мэтт Вайсфельд Объектно-ориентированное мышление [Текст] / Мэтт Вайсфельд. — Санкт-Петербург: ООО Издательство "Питер", 2014 — 304 c.

# ПРИЛОЖЕНИЕ А

SecurityConfig.java:

*// Класс конфигурации авторизации пользователя*@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
*// Метод для разграничения функционала при авторизации* @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/login", "/register").permitAll()  
 .requestMatchers("/client/\*\*").hasRole("CLIENT") *// Проверка, есть ли у пользователя роль "Клиент"* .requestMatchers("/operatorBD/\*\*").hasRole("OPERATOR") *// Проверка, есть ли у пользователя роль "Администратор БД"* .requestMatchers("/manager/\*\*").hasRole("MANAGER") *// Проверка, есть ли у пользователя роль "Менеджер"* .anyRequest().authenticated()  
 )  
 .formLogin(form -> form  
 .loginPage("/login")  
 .defaultSuccessUrl("/home", true)  
 .permitAll()  
 )  
 .logout(logout -> logout  
 .permitAll())  
 .csrf(csrf -> csrf.disable());  
  
 return http.build();  
 }  
*// Метод для авторизации пользователя* @Bean  
 public UserDetailsService userDetailsService(UsersService usersService) {  
 return email -> {  
 Users users = usersService.findByEmail(email);  
 if (users == null) {  
 throw new UsernameNotFoundException("User not found");  
 }  
 return org.springframework.security.core.userdetails.User.*builder*()  
 .username(users.getEmail())  
 .password(users.getPassword())  
 .roles(users.getRole())  
 .build();  
 };  
 }  
}

AuthController.java:

*// Класс - контроллер для авторизации и регистрации пользователя*@Controller  
public class AuthController {  
 private final UsersService usersService;  
  
 public AuthController(UsersService userService) {  
 this.usersService = userService;  
 }  
*// Метод для авторизации* @GetMapping("/login")  
 public String loginPage() {  
 return "login";  
 }  
*// Методы для регистрации пользователя* @GetMapping("/register")  
 public String register(Model model) {  
 model.addAttribute("Users", new Users());  
 return "register";  
 }  
 @PostMapping("/register")  
 public String registerUser(@Valid @ModelAttribute("Users") Users users,  
 BindingResult result, Model model) {  
 if (result.hasErrors()) {  
 return "register";  
 }  
 if (!usersService.registerUser(users)) {  
 model.addAttribute("error", "Пользователь с таким email уже существует");  
 return "register";  
 }  
 return "confirmation";  
 }  
}

ClientController.java:

*// Класс - контроллер для пользователя с ролью "Клиент"*@Controller  
public class ClientController {  
 @Autowired  
 private UsersService usersService;  
 @Autowired  
 private AutoService autoService;  
 @Autowired  
 private ScheduleSlotRepository scheduleSlotRepository;  
  
 @Autowired  
 private OrderRepository orderRepository;  
 @Autowired  
 private AutoRepository autoRepository;  
 private OrderStatus orderStatus;  
 private UserRole userRole;  
  
*// Метод для показа главной страницы* @GetMapping("/home")  
 public String homePage(Model model, Principal principal) {  
 String email = principal.getName();  
  
 Users user = usersService.findByEmail(email);  
  
 model.addAttribute("user", user);  
  
 if (user != null) {  
 if (userRole.*OPERATOR*.getRoleName().equals(user.getRole())) {  
 model.addAttribute("isOperator", true);  
 } else if (userRole.*MANAGER*.getRoleName().equals(user.getRole())) {  
 model.addAttribute("isManager", true);  
 } else if (userRole.*CLIENT*.getRoleName().equals(user.getRole())) {  
 model.addAttribute("isClient", true);  
 }  
 } else {  
 model.addAttribute("error", "Пользователь не найден.");  
 }  
  
 return "home";  
 }  
  
*// Метод для показа страницы "Добавить автомобиля"* @GetMapping("/addAuto")  
 public String showAddAutoForm(Model model) {  
 model.addAttribute("auto", new Auto());  
 List<Brand> brands = Arrays.*asList*(Brand.*values*());  
 model.addAttribute("brands", brands);  
 return "addAuto";  
 }  
*// Вспомогательный метод для вывода моделей автомобилей* @ResponseBody  
 @GetMapping("/models")  
 public Brand.Model[] getModelsByBrand(@RequestParam String brand) {  
 return Brand.*valueOf*(brand).getModels();  
 }  
 *// Вспомогательный метод для вывода моделей автомобилей* @ResponseBody  
 @GetMapping("/generations")  
 public String[] getGenerationsByModel(@RequestParam String brand, @RequestParam String model) {  
 return Arrays.*stream*(Brand.*valueOf*(brand).getModels())  
 .filter(m -> m.getName().equals(model))  
 .findFirst()  
 .map(Brand.Model::getGenerations)  
 .orElse(new String[]{});  
 }  
 *// Метод для принятия введенных данных пользователя об автомобиле* @PostMapping("/addAuto")  
 public String addAuto(@Valid @ModelAttribute("auto") Auto auto,  
 BindingResult result, Model model,  
 @AuthenticationPrincipal UserDetails userDetails, Principal principal) {  
  
 if (result.hasErrors()) {  
 return "addAuto";  
 }  
  
 String vinNumberPattern = "^[0-9A-HJ-NPR-Z]{17}$";  
 if (!auto.getVinNumber().matches(vinNumberPattern)) {  
 result.rejectValue("vinNumber", "error.auto", "VIN должен содержать только допустимые символы и быть длиной 17 символов");  
 }  
  
 String gosNumberPattern = "^[АВЕКМНОРСТУХ]{1}[0-9]{3}[АВЕКМНОРСТУХ]{2}[0-9]{2,3}$";  
 if (!auto.getGOS\_number().matches(gosNumberPattern)) {  
 result.rejectValue("GOS\_number", "error.auto", "Госномер должен быть в формате: одна буква, три цифры, две буквы, и номер региона (2 или 3 цифры)");  
 }  
  
 if (result.hasErrors()) {  
 return "addAuto";  
 }  
  
 if (autoService.isVinExists(auto.getVinNumber())) {  
 model.addAttribute("vinError", "Автомобиль с таким VIN уже существует.");  
 return "addAuto";  
 }  
  
 Users currentUser = usersService.findByEmail(principal.getName());  
 if (currentUser != null) {  
 auto.setUser(currentUser);  
 autoService.saveAuto(auto);  
 model.addAttribute("message", "Автомобиль успешно добавлен!");  
 } else {  
 model.addAttribute("error", "Пользователь не найден.");  
 return "addAuto";  
 }  
  
 return "redirect:/home";  
 }  
  
*// Метод вывода страницы "Мои автомобили"* @GetMapping("/autos")  
 public String listUserAutos(Model model, Principal principal) {  
 Users currentUser = usersService.findByEmail(principal.getName());  
  
 if (currentUser != null) {  
 List<Auto> userAutos = autoService.findByUser(currentUser);  
 model.addAttribute("autos", userAutos);  
 } else {  
 model.addAttribute("error", "Пользователь не найден.");  
 }  
  
 return "userAutos";  
 }  
 *// Методы для записи пользователя на услуги в автосервис* @GetMapping("/selectAutoAndService")  
 public String selectAutoAndService(Model model, Principal principal) {  
 Users currentUser = usersService.findByEmail(principal.getName());  
 if (currentUser != null) {  
 List<Auto> userAutos = autoService.findByUser(currentUser);  
 model.addAttribute("userAutos", userAutos);  
 }  
 return "selectAutoAndService";  
 }  
  
 @PostMapping("/selectAutoAndService")  
 public String submitAutoAndServiceSelection(@RequestParam("carId") Long carId,  
 @RequestParam("serviceType") String serviceType,  
 RedirectAttributes redirectAttributes) {  
 redirectAttributes.addAttribute("carId", carId);  
 redirectAttributes.addAttribute("serviceType", serviceType);  
 return "redirect:/selectSlot";  
 }  
 @GetMapping("/selectSlot")  
 public String showAvailableSlots(@RequestParam("carId") Long carId,  
 @RequestParam("serviceType") String serviceType,  
 Model model) {  
 LocalDate today = LocalDate.*now*();  
 List<ScheduleSlot> availableSlots = scheduleSlotRepository.findByAvailableTrue()  
 .stream()  
 .filter(slot -> !slot.getDate().isBefore(today))  
 .toList();  
  
 model.addAttribute("slots", availableSlots);  
 model.addAttribute("carId", carId);  
 model.addAttribute("serviceType", serviceType);  
 return "selectSlot";  
 }  
  
 @GetMapping("/confirmOrder")  
 public String confirmOrder(@RequestParam("slotId") Long slotId,  
 @RequestParam("carId") Long carId,  
 @RequestParam("serviceType") String serviceType,  
 Model model,  
 Principal principal) {  
  
 Users currentUser = usersService.findByEmail(principal.getName());  
  
 if (currentUser != null) {  
 List<Auto> userAutos = autoService.findByUser(currentUser);  
 model.addAttribute("userAutos", userAutos);  
 }  
 ScheduleSlot slot = scheduleSlotRepository.findById(slotId).orElseThrow();  
 LocalDateTime dateTime = LocalDateTime.*of*(slot.getDate(), slot.getTime());  
  
 String formattedDateTime = dateTime.format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm"));  
 model.addAttribute("formattedDateTime", formattedDateTime);  
  
 model.addAttribute("slot", slot);  
 model.addAttribute("serviceType", serviceType);  
  
 Auto auto = autoRepository.findById(carId).orElseThrow();  
 model.addAttribute("auto", auto);  
  
 return "confirmOrder";  
 }  
  
 @PostMapping("/confirmOrder")  
 public String bookOrder(@RequestParam("slotId") Long slotId,  
 @RequestParam("carId") Long carId,  
 @RequestParam("serviceType") String serviceType,  
 @RequestParam(value = "comment", required = false) String comment,  
 Principal principal) {  
  
 ScheduleSlot slot = scheduleSlotRepository.findById(slotId).orElseThrow();  
 slot.setAvailable(false);  
  
 Order order = new Order();  
 order.setSlot(slot);  
  
 Users currentUser = usersService.findByEmail(principal.getName());  
 if (currentUser != null) {  
 order.setClient(currentUser);  
 } else {  
 throw new IllegalArgumentException("Текущий пользователь не найден");  
 }  
  
 Auto auto = autoRepository.findById(carId).orElseThrow();  
 if (auto.getUser().equals(currentUser)) {  
 order.setAuto(auto);  
 } else {  
 throw new IllegalArgumentException("Выбранный автомобиль не принадлежит текущему пользователю");  
 }  
  
 order.setServiceType(serviceType);  
 order.setClientComment(comment);  
 order.setStatus(orderStatus.*AWAITING\_CONFIRMATION*.getValue());  
  
 orderRepository.save(order);  
 scheduleSlotRepository.save(slot);  
  
 return "okOrder";  
 }  
*// Метод для показа страницы "История обращений"* @GetMapping("/orders")  
 public String getClientOrders(Model model, Principal principal) {  
 Users currentUser = usersService.findByEmail(principal.getName());  
 if (currentUser != null) {  
 List<Order> orders = orderRepository.findByClient(currentUser);  
 model.addAttribute("orders", orders);  
 }  
 return "clientOrders";  
 }  
}

ManagerController.java:

*// Класс - контроллер для пользователя с ролью "Менеджер"*@Controller  
public class ManagerController {  
 @Autowired  
 private ScheduleSlotRepository scheduleSlotRepository;  
 @Autowired  
 private UsersService usersService;  
 @Autowired  
 private AutoService autoService;  
 @Autowired  
 private OrderService orderService;  
 private final OrderRepository orderRepository;  
 private final StatisticsService statisticsService;  
 private UserRole userRole;  
  
 public ManagerController(OrderRepository orderRepository, StatisticsService statisticsService) {  
 this.orderRepository = orderRepository;  
 this.statisticsService = statisticsService;  
 }  
  
*// Метод вывода страницы "Список всех автомобилей"* @GetMapping("/manager/allAutos")  
 public String viewAllAutos(  
 @RequestParam(value = "brand", required = false) String brand,  
 @RequestParam(value = "model", required = false) String model,  
 @RequestParam(value = "vinNumber", required = false) String vinNumber,  
 @RequestParam(value = "gosNumber", required = false) String gosNumber,  
 @RequestParam(value = "email", required = false) String email,  
 Model modelAttribute) {  
  
 List<Auto> autos;  
  
 if (brand == null && model == null && vinNumber == null && gosNumber == null && email == null) {  
 autos = autoService.getAllAutos();  
 } else {  
 autos = autoService.searchAutos(brand, model, vinNumber, gosNumber, email);  
 }  
  
 modelAttribute.addAttribute("autos", autos);  
 return "allAutos";  
 }  
 *// Методы для добавления менеджером слотов для записи на услуги в автосервис* @GetMapping("/manager/addScheduleSlot")  
 public String showAddSlotPage(Model model) {  
 List<ScheduleSlot> slots = scheduleSlotRepository.findByAvailableTrue();  
 model.addAttribute("slots", slots);  
 return "addScheduleSlot";  
 }  
 @PostMapping("/manager/addScheduleSlot")  
 public String addScheduleSlot(@RequestParam("date") String date,  
 @RequestParam("time") String time) {  
 ScheduleSlot slot = new ScheduleSlot();  
 slot.setDate(LocalDate.*parse*(date));  
 slot.setTime(LocalTime.*parse*(time));  
 slot.setAvailable(true);  
 scheduleSlotRepository.save(slot);  
 return "redirect:/manager/viewSlots";  
 }  
 @PostMapping("/manager/addFullDaySlots")  
 public ResponseEntity<String> addFullDaySlots(@RequestParam("date") String dateStr) {  
 try {  
 LocalDate date = LocalDate.*parse*(dateStr);  
 List<ScheduleSlot> slots = new ArrayList<>();  
 System.*out*.println("Добавление слотов для даты: " + date);  
  
 for (int hour = 8; hour <= 19; hour++) {  
 LocalTime time = LocalTime.*of*(hour, 0);  
 boolean slotExists = scheduleSlotRepository.existsByDateAndTime(date, time);  
  
 if (!slotExists) {  
 ScheduleSlot slot = new ScheduleSlot();  
 slot.setDate(date);  
 slot.setTime(time);  
 slot.setAvailable(true);  
 slots.add(slot);  
 System.*out*.println("Добавлен слот для времени: " + time);  
 } else {  
 System.*out*.println("Слот для времени " + time + " уже существует, пропускаем.");  
 }  
 }  
  
 if (!slots.isEmpty()) {  
 scheduleSlotRepository.saveAll(slots);  
 System.*out*.println("Все слоты на день сохранены успешно.");  
 return ResponseEntity.*ok*("Слоты на весь день добавлены успешно.");  
 } else {  
 return ResponseEntity.*ok*("Слоты уже существуют на весь день.");  
 }  
  
 } catch (DateTimeParseException e) {  
 System.*err*.println("Ошибка формата даты: " + e.getMessage());  
 return ResponseEntity.*status*(HttpStatus.*BAD\_REQUEST*).body("Неверный формат даты. Ожидаемый формат: yyyy-MM-dd.");  
 } catch (Exception e) {  
 System.*err*.println("Ошибка при добавлении слотов: " + e.getMessage());  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("Ошибка при добавлении слотов: " + e.getMessage());  
 }  
 }  
*// Метод для показа страницы "Доступные слоты для записи"* @GetMapping("/manager/viewSlots")  
 public String viewAvailableSlots(Model model) {  
 List<ScheduleSlot> slots = scheduleSlotRepository.findByAvailableTrue();  
 model.addAttribute("slots", slots);  
 return "viewSlots";  
 }  
  
*// Метод вывода страницы "Список клиентов"* @GetMapping("/manager/clients")  
 public String viewClients(@RequestParam(value = "search", required = false) String search, Model model) {  
 List<Users> clients;  
  
 if (search != null && !search.isEmpty()) {  
 clients = usersService.searchClients(search);  
 } else {  
 clients = usersService.findByRole(userRole.*CLIENT*.getRoleName());   
 }  
  
 model.addAttribute("clients", clients);  
 model.addAttribute("searchQuery", search);   
 return "managerClients";   
 }  
 @PostMapping("/manager/orders/updateStatus")  
 public String updateOrderStatus(@RequestParam Long orderId, @RequestParam int status) {  
 Order order = orderService.findById(orderId);  
 order.setStatus(status);  
 orderService.save(order);  
 return "redirect:/manager/orders";  
 }  
 @GetMapping("/manager/orders/addComment")  
 public String showAddCommentPage(@RequestParam Long orderId, Model model) {  
 Order order = orderService.findById(orderId);  
 model.addAttribute("order", order);  
 return "addComment";  
 }  
 @PostMapping("/manager/orders/addComment")  
 public String saveManagerComment(@RequestParam Long orderId, @RequestParam String managerComment) {  
 Order order = orderService.findById(orderId);  
 order.setManagerComment(managerComment);  
 orderService.save(order);  
 return "redirect:/manager/orders";  
 }  
  
*// Метод вывода страницы "Заказы"* @GetMapping("/manager/orders")  
 public String searchOrders(  
 @RequestParam(required = false) Long orderId,  
 @RequestParam(required = false) String brand,  
 @RequestParam(required = false) String model,  
 @RequestParam(required = false) Integer year,  
 @RequestParam(required = false) String vinNumber,  
 @RequestParam(required = false) String GOS\_number,  
 @RequestParam(required = false) String date,  
 @RequestParam(required = false) String clientEmail,  
 @RequestParam(required = false) Integer status,  
 @RequestParam(required = false) String serviceType,  
 Model modelAttribute) {  
  
 Brand parsedBrand = *parseBrand*(brand);  
  
 LocalDate parsedDate = (date != null && !date.isEmpty()) ? LocalDate.*parse*(date) : null;  
 System.*out*.println(parsedDate);  
 List<Order> orders = orderRepository.findOrdersByFilter(  
 orderId, parsedBrand, model, year, vinNumber, GOS\_number, parsedDate, clientEmail, status, serviceType);  
  
 modelAttribute.addAttribute("orders", orders);  
 return "managerOrders";  
 }  
*// Метод логики формирования заказ-наряда* @PostMapping("/manager/orders/exportDocx")  
 public ResponseEntity<?> generateOrderDocx(@RequestParam Long orderId) {  
 try {  
 Order order = orderService.getOrderById(orderId);  
 if (order == null) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(null);  
 }  
 ByteArrayOutputStream out = new ByteArrayOutputStream();  
  
 try (XWPFDocument document = new XWPFDocument()) {  
 XWPFParagraph title = document.createParagraph();  
 title.setAlignment(ParagraphAlignment.*CENTER*);  
 XWPFRun titleRun = title.createRun();  
 titleRun.setText("Заказ-наряд");  
 titleRun.setBold(true);  
 titleRun.setFontSize(16);  
 XWPFTable table = document.createTable();  
 XWPFTableRow headerRow = table.getRow(0);  
 headerRow.getCell(0).setText("Клиент");  
 headerRow.addNewTableCell().setText("Автомобиль");  
 headerRow.addNewTableCell().setText("VIN");  
 headerRow.addNewTableCell().setText("Услуга");  
 headerRow.addNewTableCell().setText("Дата и время");  
 XWPFTableRow valueRow = table.createRow();  
 valueRow.getCell(0).setText(order.getClient().getEmail());  
 valueRow.getCell(1).setText(order.getAuto().getBrand() + " " + order.getAuto().getModel());  
 valueRow.getCell(2).setText(order.getAuto().getVinNumber());  
 valueRow.getCell(3).setText(order.getServiceType());  
 valueRow.getCell(4).setText(order.getSlot().getDate() + " " + order.getSlot().getTime());  
 XWPFParagraph commentsParagraph = document.createParagraph();  
 XWPFRun commentsRun = commentsParagraph.createRun();  
 commentsRun.addBreak();  
 commentsRun.setText("Комментарий клиента: " + (order.getClientComment() != null ? order.getClientComment() : "Нет"));  
 commentsRun.addBreak();  
 commentsRun.setText("Комментарий менеджера: " + (order.getManagerComment() != null ? order.getManagerComment() : "Нет"));  
 commentsRun.addBreak();  
 commentsRun.setText("Цена: " + (order.getPrice() != null ? order.getPrice() : "0") + " руб." );  
 document.write(out);  
 }  
 ByteArrayInputStream in = new ByteArrayInputStream(out.toByteArray());  
 HttpHeaders headers = new HttpHeaders();  
 headers.add("Content-Disposition", "attachment; filename=order\_" + orderId + ".docx");  
 return ResponseEntity  
 .*ok*()  
 .headers(headers)  
 .contentType(MediaType.*APPLICATION\_OCTET\_STREAM*)  
 .body(new InputStreamResource(in));  
 } catch (IOException e) {  
 e.printStackTrace();  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(null);  
 }  
 }  
 private void addTableRow(XWPFTable table, String header, String value) {  
 XWPFTableRow row = table.createRow();  
 row.getCell(0).setText(header);  
 row.addNewTableCell().setText(value);  
 }  
 @PostMapping("/manager/orders/setPrice")  
 public String setOrderPrice(@RequestParam("orderId") Long orderId,  
 @RequestParam("price") Double price,  
 RedirectAttributes redirectAttributes) {  
 try {  
 orderService.updateOrderPrice(orderId, price);  
 redirectAttributes.addFlashAttribute("successMessage", "Цена успешно обновлена!");  
 } catch (Exception e) {  
 redirectAttributes.addFlashAttribute("errorMessage", "Ошибка при обновлении цены: " + e.getMessage());  
 }  
 return "redirect:/manager/orders";  
 }  
  
*// Методы вывода статистики* @GetMapping("/manager/statistics")  
 public String showStatisticsForm() {  
 return "managerStatsForm";  
 }  
  
 @GetMapping("/manager/statistics/results")  
 public String getMonthlyStatistics(  
 @RequestParam("month") int month,  
 @RequestParam("year") int year,  
 Model model  
 ) {  
 Map<String, Object> statistics = statisticsService.calculateMonthlyStatistics(month, year);  
 model.addAttribute("statistics", statistics);  
 model.addAttribute("month", month);  
 model.addAttribute("year", year);  
 return "managerStatistics";  
 }  
  
}

OperatorBDController.java:

*// Класс - контроллер для пользователя с ролью "Администратор БД"*@Controller  
@RequestMapping("/operatorBD")  
public class OperatorBDController {  
  
 @Autowired  
 private DatabaseService databaseService;  
  
 @Autowired  
 private UsersService usersService;  
 *// Метод для вывода списка существующих таблиц в БД* @GetMapping("/tables")  
 public String showTables(Model model) {  
 List<String> tables = databaseService.getAllTables();  
 model.addAttribute("tables", tables);  
 return "operatorTablesList";  
 }  
 *// Метод для вывода конкректной существующей таблицы в БД* @GetMapping("/table/{tableName}")  
 public String showTable(@PathVariable String tableName, Model model) {  
 List<List<Object>> tableData = databaseService.getTableData(tableName);  
 model.addAttribute("tableName", tableName);  
 model.addAttribute("tableData", tableData);  
 return "operatorTableData";  
 }  
 *// Методы для регистрации менеджера в системе* @GetMapping("/registerManagerForm")  
 public String showRegisterManagerForm(Model model) {  
 model.addAttribute("Manager", new Users());  
 return "registerManager";  
 }  
 @PostMapping("/registerManager")  
 public String registerManager(@Valid @ModelAttribute("Manager") Users manager,  
 BindingResult result, Model model) {  
 if (result.hasErrors()) {  
 return "registerManager";  
 }  
 if (!usersService.registerManager(manager)){  
 model.addAttribute("message", "Ошибка: менеджер с таким e-mail уже существует.");  
 }  
 return "confirmationManage";  
 }  
  
*// Метод вывода страницы "Список персонала"* @GetMapping("/personnel")  
 public String showPersonnel(  
 @RequestParam(value = "search", required = false) String searchQuery,  
 Model model) {  
 List<Users> personnel;  
 if (searchQuery != null && !searchQuery.isEmpty()) {  
 personnel = usersService.searchPersonnel(searchQuery);  
 } else {  
 personnel = usersService.getAllPersonnel();  
 }  
 model.addAttribute("clients", personnel);  
 model.addAttribute("searchQuery", searchQuery);  
 return "personnelList";  
 }  
  
*// Метод логики удаления персонала* @PostMapping("/deleteClient")  
 public String deleteClient(@RequestParam("id") Long id, RedirectAttributes redirectAttributes) {  
 try {  
 usersService.deleteUserById(id);  
 redirectAttributes.addFlashAttribute("successMessage", "Пользователь успешно удален.");  
 } catch (Exception e) {  
 redirectAttributes.addFlashAttribute("errorMessage", "Ошибка при удалении пользователя.");  
 }  
 return "redirect:/operatorBD/personnel";  
 }  
}

UserController.java:

*// Вспомогательный класс - контроллер для изменения данных пользователя*@Controller  
@RequestMapping("/user")  
public class UserController {  
  
 @Autowired  
 private UsersService userService;  
  
*// Методы для редактирования данных клиента* @GetMapping("/edit")  
 public String editUserForm(Principal principal, Model model) {  
 Users user = userService.findByEmail(principal.getName());  
 model.addAttribute("user", user);  
 return "edit\_user";  
 }  
 @PostMapping("/edit")  
 public String updateUser(  
 @ModelAttribute("user") @Valid Users user,  
 BindingResult bindingResult,  
 Principal principal,  
 RedirectAttributes redirectAttributes) {  
 if (bindingResult.hasErrors()) {  
 return "edit\_user";  
 }  
 userService.updateUser(principal.getName(), user);  
 redirectAttributes.addFlashAttribute("successMessage", "Данные успешно обновлены!");  
 return "redirect:/home";  
 }  
}

Brand.java:

*// Вспомогательный класс - enum для вывода соответсвующих к моделям автомобилям их поколений*public enum Brand {  
 *// Список автомобилей (можно дополнить)  
 TOYOTA*(new Model[]{  
 new Model("Camry", new String[]{"XV50", "XV70"}),  
 new Model("Corolla", new String[]{"E140", "E170"})  
 }),  
 *HONDA*(new Model[]{  
 new Model("Civic", new String[]{"FC", "FK"}),  
 new Model("Accord", new String[]{"CR", "CP"})  
 });  
  
 private final Model[] models;  
  
 Brand(Model[] models) {  
 this.models = models;  
 }  
  
 public Model[] getModels() {  
 return models;  
 }  
  
 public static class Model {  
 private final String name;  
 private final String[] generations;  
  
 public Model(String name, String[] generations) {  
 this.name = name;  
 this.generations = generations;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String[] getGenerations() {  
 return generations;  
 }  
 }  
 public static Brand parseBrand(String brandName) {  
 if (brandName == null || brandName.isEmpty()) {  
 System.*out*.println("null");  
 return null;  
 }  
 try {  
 System.*out*.println(Brand.*valueOf*(brandName.toUpperCase()));  
 return Brand.*valueOf*(brandName.toUpperCase());  
 } catch (IllegalArgumentException e) {  
 throw new IllegalArgumentException("Invalid brand name: " + brandName);  
 }  
 }  
 }

OrderStatus.java:

*// Вспомогательный класс - enum для статуса заказа*public enum OrderStatus {  
 *AWAITING\_CONFIRMATION*(1),  
 *CONFIRMED*(2),  
 *IN\_PROGRESS*(3),  
 *COMPLETED*(4);  
  
 private final int value;  
  
 OrderStatus(int value) {  
 this.value = value;  
 }  
  
 public int getValue() {  
 return value;  
 }  
  
 public static OrderStatus fromValue(int value) {  
 for (OrderStatus status : *values*()) {  
 if (status.value == value) {  
 return status;  
 }  
 }  
 throw new IllegalArgumentException("Unknown status value: " + value);  
 }  
}

OrderStatistics.java:

*// Вспомогательный класс для формирования статистики*public class OrderStatistics {  
 private int totalOrders;  
 private double averagePrice;  
 private Map<Integer, Long> dailyCounts;  
  
 public OrderStatistics(int totalOrders, double averagePrice, Map<Integer, Long> dailyCounts) {  
 this.totalOrders = totalOrders;  
 this.averagePrice = averagePrice;  
 this.dailyCounts = dailyCounts;  
 }  
  
 public int getTotalOrders() {  
 return totalOrders;  
 }  
  
 public double getAveragePrice() {  
 return averagePrice;  
 }  
  
 public Map<Integer, Long> getDailyCounts() {  
 return dailyCounts;  
 }  
}

UserRole.java:

*// Вспомогательный класс - enum для назначения ролей пользователей*public enum UserRole {  
 *CLIENT*("CLIENT"),  
 *OPERATOR*("OPERATOR"),  
 *MANAGER*("MANAGER");  
  
 private final String roleName;  
  
 UserRole(String roleName) {  
 this.roleName = roleName;  
 }  
  
 public String getRoleName() {  
 return roleName;  
 }  
}

Auto.java:

*// Класс - модель для таблицы хранения данных об автомобилях клиентов*@Entity  
@Table(name = "auto\_autoservice")  
public class Auto {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Enumerated(EnumType.*STRING*)  
 @NotBlank(message = "Поле не должно быть пустым")  
 private Brand brand;  
 @NotBlank(message = "Поле не должно быть пустым")  
 private String model;  
 @NotBlank(message = "Поле не должно быть пустым")  
 private String generation;  
 @NotBlank(message = "Поле не должно быть пустым")  
 @Min(value = 1949, message = "Год должен быть не меньше 1900")  
 @Max(value = 2099, message = "Год должен быть не больше 2099")  
 private int year;  
 @NotBlank(message = "Поле не должно быть пустым")  
 @Max(value = 17, message = "VIN должен содержать 17 символов")  
 @Min(value = 17, message = "VIN должен содержать 17 символов")  
 @Size(min = 17, max = 17, message = "VIN должен содержать 17 символов")  
 @Pattern(regexp = "^[0-9A-HJ-NPR-Z]{17}$", message = "VIN должен содержать только допустимые символы и быть длиной 17 символов")  
 private String vinNumber;  
 @NotBlank(message = "Поле не должно быть пустым")  
 @Max(value = 9, message = "Госномер должен содержать 8 или 9 символов")  
 @Min(value = 8, message = "Госномер должен содержать 8 или 9 символов")  
 @Size(min = 8, max = 9, message = "Госномер должен содержать от 8 до 9 символов")  
 @Pattern(regexp = "[АВЕКМНОРСТУХ]{1}[0-9]{3}[АВЕКМНОРСТУХ]{2}[0-9]{2,3}",  
 message = "Госномер должен быть в формате: одна буква, три цифры, две буквы, и номер региона (2 или 3 цифры)")  
 private String GOS\_number;  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "user\_id")  
 private Users user;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public Brand getBrand() {  
 return brand;  
 }  
  
 public void setBrand(Brand brand) {  
 this.brand = brand;  
 }  
  
 public String getModel() {  
 return model;  
 }  
  
 public void setModel(String model) {  
 this.model = model;  
 }  
  
 public String getGeneration() {  
 return generation;  
 }  
  
 public void setGeneration(String generation) {  
 this.generation = generation;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public void setYear(int year) {  
 this.year = year;  
 }  
  
 public String getVinNumber() {  
 return vinNumber;  
 }  
  
 public void setVinNumber(String vinNumber) {  
 this.vinNumber = vinNumber;  
 }  
  
 public String getGOS\_number() {  
 return GOS\_number;  
 }  
  
 public void setGOS\_number(String GOS\_number) {  
 this.GOS\_number = GOS\_number;  
 }  
  
 public Users getUser() {  
 return user;  
 }  
  
 public void setUser(Users user) {  
 this.user = user;  
 }  
}

Order.java:

*// Класс - модель таблицы хранения данных о заказах клиентов*@Entity  
@Table(name = "order\_autoservice")  
public class Order {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "user\_id", nullable = false)  
 private Users client;  
  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "auto\_id", nullable = false)  
 private Auto auto;  
  
 @Column(nullable = false)  
 private String serviceType;  
 @Column(length = 500)  
 private String clientComment;  
 @Column(length = 500)  
 private String managerComment;  
 private Double price;  
  
  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "schedule\_slot\_id", nullable = false)  
 private ScheduleSlot slot;  
  
 @Column(nullable = false)  
 private int status;  
  
 public int getStatus() {  
 return status;  
 }  
  
 public void setStatus(int status) {  
 this.status = status;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getManagerComment() {  
 return managerComment;  
 }  
  
 public void setManagerComment(String managerComment) {  
 this.managerComment = managerComment;  
 }  
  
 public Users getClient() {  
 return client;  
 }  
  
 public void setClient(Users client) {  
 this.client = client;  
 }  
 public String getClientComment() {  
 return clientComment;  
 }  
  
 public void setClientComment(String clientComment) {  
 this.clientComment = clientComment;  
 }  
 public Auto getAuto() {  
 return auto;  
 }  
  
 public void setAuto(Auto auto) {  
 this.auto = auto;  
 }  
  
 public String getServiceType() {  
 return serviceType;  
 }  
  
 public void setServiceType(String serviceType) {  
 this.serviceType = serviceType;  
 }  
  
 public ScheduleSlot getSlot() {  
 return slot;  
 }  
  
 public void setSlot(ScheduleSlot slot) {  
 this.slot = slot;  
 }  
  
 public Double getPrice() {  
 return price;  
 }  
  
 public void setPrice(Double price) {  
 this.price = price;  
 }  
}

ScheduleSlot.java:

*// Класс - модель вспомогательной таблицы хранения данных о доступных слотах для записи*@Entity  
public class ScheduleSlot {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private LocalDate date;  
 private LocalTime time;  
 private boolean available = true;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public LocalDate getDate() {  
 return date;  
 }  
  
 public void setDate(LocalDate date) {  
 this.date = date;  
 }  
  
 public LocalTime getTime() {  
 return time;  
 }  
  
 public void setTime(LocalTime time) {  
 this.time = time;  
 }  
  
 public boolean isAvailable() {  
 return available;  
 }  
  
 public void setAvailable(boolean available) {  
 this.available = available;  
 }  
}

Users.java:

*// Класс - модель таблицы хранения данных пользователей*@Entity  
@Table(name = "users\_autoservice")  
public class Users {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @NotBlank(message = "Имя пользователя не должно быть пустым")  
 private String username;  
 @NotBlank(message = "Фамилия пользователя не должно быть пустым")  
 private String surname;  
 @NotBlank(message = "Пароль не должен быть пустым")  
 @Size(min = 6, message = "Пароль должен содержать не менее 6 символов")  
 private String password;  
 private String role;  
 @Column(name = "phone\_number", length = 19)  
 @Pattern(regexp = "^\\+7 \\([0-9]{3}\\) [0-9]{3}-[0-9]{2}-[0-9]{2}$", message = "Номер телефона должен быть в формате +7 (XXX) XXX-XX-XX")  
 private String phone\_number;  
  
 @NotBlank(message = "Email обязателен")  
 @Email(message = "Некорректный формат email")  
 @Column(name = "email", length = 50)  
 private String email;  
  
 public List<Auto> getAutos() {  
 return autos;  
 }  
  
 public void setAutos(List<Auto> autos) {  
 this.autos = autos;  
 }  
  
 @OneToMany(mappedBy = "user", cascade = CascadeType.*ALL*, orphanRemoval = true)  
 private List<Auto> autos;  
  
  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getRole() {  
 return role;  
 }  
  
 public void setRole(String role) {  
 this.role = role;  
 }  
  
 public String getPhone\_number() {  
 return phone\_number;  
 }  
  
 public void setPhone\_number(String phone\_number) {  
 this.phone\_number = phone\_number;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
}

AutoRepository.java:

*// Интерфейс для обращения к таблице хранения данных об автомобилях клиентов*public interface AutoRepository extends JpaRepository<Auto, Long> {  
 Auto findByVinNumber(String vinNumber);  
 List<Auto> findByUser(Users user);  
 @Query("SELECT a FROM Auto a WHERE " +  
 "(LOWER(a.brand) LIKE LOWER(CONCAT('%', :brand, '%')) OR :brand IS NULL) AND " +  
 "(LOWER(a.model) LIKE LOWER(CONCAT('%', :model, '%')) OR :model IS NULL) AND " +  
 "(LOWER(a.vinNumber) LIKE LOWER(CONCAT('%', :vinNumber, '%')) OR :vinNumber IS NULL) AND " +  
 "(LOWER(a.GOS\_number) LIKE LOWER(CONCAT('%', :gosNumber, '%')) OR :gosNumber IS NULL) AND " +  
 "(LOWER(a.user.email) LIKE LOWER(CONCAT('%', :email, '%')) OR :email IS NULL)")  
 List<Auto> findBySearchQuery(  
 @Param("brand") String brand,  
 @Param("model") String model,  
 @Param("vinNumber") String vinNumber,  
 @Param("gosNumber") String gosNumber,  
 @Param("email") String email  
 );  
}

OrderRepository.java:

*// Интерфейс для обращения к таблице хранения данных о заказах клиентов*public interface OrderRepository extends JpaRepository<Order, Long> {  
 List<Order> findByClient(Users client);  
 @Query("SELECT o FROM Order o " +  
 "JOIN o.auto a " +  
 "WHERE (:orderId IS NULL OR o.id = :orderId) AND " +  
 "(:brand IS NULL OR a.brand = :brand) AND " +  
 "(:model IS NULL OR a.model LIKE %:model%) AND " +  
 "(:year IS NULL OR a.year = :year) AND " +  
 "(:vin IS NULL OR a.vinNumber LIKE %:vin%) AND " +  
 "(:gosNumber IS NULL OR a.GOS\_number LIKE %:gosNumber%) AND " +  
 "(:date IS NULL OR o.slot.date = :date) AND " +  
 "(:clientEmail IS NULL OR o.client.email LIKE %:clientEmail%) AND " +  
 "(:status IS NULL OR o.status = :status) AND " +  
 "(:serviceType IS NULL OR o.serviceType LIKE %:serviceType%)")  
 List<Order> findOrdersByFilter(  
 @Param("orderId") Long orderId, *// ID заказа* @Param("brand") Brand brand,  
 @Param("model") String model,  
 @Param("year") Integer year,  
 @Param("vin") String vin,  
 @Param("gosNumber") String gosNumber,  
 @Param("date") LocalDate date,  
 @Param("clientEmail") String clientEmail,  
 @Param("status") Integer status,  
 @Param("serviceType") String serviceType);  
 @Query("SELECT o FROM Order o WHERE FUNCTION('MONTH', o.slot.date) = :month AND FUNCTION('YEAR', o.slot.date) = :year")  
 List<Order> findByMonthAndYear(@Param("month") int month, @Param("year") int year);  
}

ScheduleSlotRepository.java:

*// Интерфейс для обращения к таблице хранения данных о доступных слотах для записи*public interface ScheduleSlotRepository extends JpaRepository<ScheduleSlot, Long> {  
 List<ScheduleSlot> findByAvailableTrue();  
 List<ScheduleSlot> findByDateAndAvailableTrue(LocalDate date);  
 boolean existsByDateAndTime(LocalDate date, LocalTime time);  
}

UsersRepository.java:

*// Интерфейс для обращения к таблице хранения данных пользователей*public interface UsersRepository extends JpaRepository<Users, Long> {  
 Users findByUsername(String username);  
 Users findByEmail(String email);  
 @Query("SELECT u FROM Users u WHERE u.role = 'CLIENT' AND " +  
 "(LOWER(u.username) LIKE LOWER(CONCAT('%', :query, '%')) " +  
 "OR LOWER(u.surname) LIKE LOWER(CONCAT('%', :query, '%')) " +  
 "OR LOWER(u.email) LIKE LOWER(CONCAT('%', :query, '%')) " +  
 "OR u.phone\_number LIKE CONCAT('%', :query, '%'))")  
 List<Users> findClientsBySearchQuery(@Param("query") String query);  
  
 @Query("SELECT u FROM Users u WHERE u.role = :role")  
 List<Users> findByRole(@Param("role") String role);  
 @Query("SELECT u FROM Users u WHERE u.role IN :roles")  
 List<Users> findPersonnelByRoles(@Param("roles") List<String> roles);  
  
 @Query("SELECT u FROM Users u WHERE u.role IN :roles AND " +  
 "(LOWER(u.username) LIKE LOWER(CONCAT('%', :query, '%')) " +  
 "OR LOWER(u.surname) LIKE LOWER(CONCAT('%', :query, '%')) " +  
 "OR LOWER(u.email) LIKE LOWER(CONCAT('%', :query, '%')) " +  
 "OR u.phone\_number LIKE CONCAT('%', :query, '%'))")  
 List<Users> searchPersonnelByQuery(@Param("query") String query, @Param("roles") List<String> roles);  
}

AutoService.java:

*// Класс - сервис для реализации функций, связанных с обращениями к таблице с данными об автомобилях клиентов*@Service  
public class AutoService {  
  
 @Autowired  
 private AutoRepository autoRepository;  
  
 public void saveAuto(Auto auto) {  
 autoRepository.save(auto);  
 }  
  
 public boolean isVinExists(String vinNumber) {  
 return autoRepository.findByVinNumber(vinNumber) != null;  
 }  
 public List<Auto> findByUser(Users user) {  
 return autoRepository.findByUser(user);  
 }  
 public List<Auto> searchAutos(String brand, String model, String vinNumber, String gosNumber, String email) {  
 return autoRepository.findBySearchQuery(brand, model, vinNumber, gosNumber, email);  
 }  
 public List<Auto> getAllAutos() {  
 return autoRepository.findAll();  
 }  
}

DatabaseService.java:

*// Класс - сервис для реализации функций обращений с данными о таблицах и их содержании в БД*@Service  
public class DatabaseService {  
  
 private final JdbcTemplate jdbcTemplate;  
  
 public DatabaseService(JdbcTemplate jdbcTemplate) {  
 this.jdbcTemplate = jdbcTemplate;  
 }  
  
 public List<String> getAllTables() {  
 return jdbcTemplate.queryForList("SHOW TABLES", String.class);  
 }  
 public List<List<Object>> getTableData(String tableName) {  
 String query = "SELECT \* FROM " + tableName;  
 return jdbcTemplate.query(query, (rs, rowNum) -> {  
 int columnCount = rs.getMetaData().getColumnCount();  
 List<Object> row = new ArrayList<>();  
 for (int i = 1; i <= columnCount; i++) {  
 row.add(rs.getObject(i));  
 }  
 return row;  
 });  
 }  
}

OrderService.java:

*// Класс - сервис для реализации функций, связанных с обращениями к таблице с данными о заказах клиентов*@Service  
public class OrderService {  
  
 @Autowired  
 private OrderRepository orderRepository;  
  
 public Order findById(Long id) {  
 Optional<Order> order = orderRepository.findById(id);  
 return order.orElseThrow(() -> new RuntimeException("Order not found"));  
 }  
 public Order getOrderById(Long id) {  
 return orderRepository.findById(id)  
 .orElse(null);  
 }  
  
 public void save(Order order) {  
 orderRepository.save(order);  
 }  
  
 public void updateOrderPrice(Long orderId, Double price) {  
 Order order = orderRepository.findById(orderId)  
 .orElseThrow(() -> new IllegalArgumentException("Заказ с ID " + orderId + " не найден"));  
 order.setPrice(price);  
 orderRepository.save(order);  
}  
 public OrderStatistics getMonthlyStatistics(int month, int year) {  
 List<Order> orders = orderRepository.findByMonthAndYear(month, year);  
  
 int totalOrders = orders.size();  
 double averagePrice = orders.stream()  
 .filter(order -> order.getPrice() != null)  
 .mapToDouble(Order::getPrice)  
 .average()  
 .orElse(0);  
  
 Map<Integer, Long> dailyCounts = orders.stream()  
 .collect(Collectors.*groupingBy*(  
 order -> order.getSlot().getDate().getDayOfMonth(),  
 Collectors.*counting*()  
 ));  
  
 return new OrderStatistics(totalOrders, averagePrice, dailyCounts);  
 }  
}

StatisticsService.java:

*// Класс - сервис для реализации функций для статистики*@Service  
public class StatisticsService {  
  
 private final OrderRepository orderRepository;  
  
 public StatisticsService(OrderRepository orderRepository) {  
 this.orderRepository = orderRepository;  
 }  
  
 public Map<String, Object> calculateMonthlyStatistics(int month, int year) {  
 List<Order> orders = orderRepository.findByMonthAndYear(month, year);  
  
 int totalOrders = orders.size();  
  
 double averagePrice = orders.stream()  
 .filter(order -> order.getPrice() != null)  
 .mapToDouble(Order::getPrice)  
 .average()  
 .orElse(0.0);  
 double totalPrice = orders.stream()  
 .filter(order -> order.getPrice() != null)  
 .mapToDouble(Order::getPrice)  
 .sum();  
  
 Map<LocalDate, Long> ordersPerDay = orders.stream()  
 .collect(Collectors.*groupingBy*(  
 order -> order.getSlot().getDate(),  
 Collectors.*counting*()  
 ));  
  
 Map<String, Object> statistics = new HashMap<>();  
 statistics.put("totalOrders", totalOrders);  
 statistics.put("averagePrice", averagePrice);  
 statistics.put("totalPrice", totalPrice);  
 statistics.put("ordersPerDay", ordersPerDay);  
  
 return statistics;  
 }  
}

UsersService.java:

*// Класс - сервис для реализации функций, связанных с обращениями к таблице с данными о пользователях*@Service  
public class UsersService {  
 private final UsersRepository usersRepository;  
 private final BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();  
 private static final String *PHONE\_PATTERN* = "^\\+7 \\([0-9]{3}\\) [0-9]{3}-[0-9]{2}-[0-9]{2}$";  
 private static final Pattern *pattern* = Pattern.*compile*(*PHONE\_PATTERN*);  
 private UserRole userRole;  
 public UsersService(UsersRepository userRepository) {  
 this.usersRepository = userRepository;  
  
 }  
  
 public boolean authenticate(String email, String password) {  
 Users user = usersRepository.findByEmail(email);  
  
 if (user != null && passwordEncoder.matches(password, user.getPassword())) {  
 return true;  
 }  
  
 return false;  
 }  
  
 public boolean isValidPhoneNumber(String phoneNumber) {  
 return *pattern*.matcher(phoneNumber).matches();  
 }  
 public boolean registerUser(Users newUser) {  
 if (usersRepository.findByEmail(newUser.getEmail()) != null ) {  
 return false;  
 }  
 if (!isValidPhoneNumber(newUser.getPhone\_number())) {  
 System.*out*.println(newUser.getPhone\_number());  
 return false;  
 }  
 newUser.setUsername(newUser.getUsername());  
 newUser.setSurname(newUser.getSurname());  
 newUser.setPassword(passwordEncoder.encode(newUser.getPassword()));  
 newUser.setPhone\_number(newUser.getPhone\_number());  
 newUser.setEmail(newUser.getEmail());  
 newUser.setRole(userRole.*CLIENT*.getRoleName());  
 usersRepository.save(newUser);  
 return true;  
 }  
  
 public boolean registerManager(Users newManager) {  
 if (usersRepository.findByEmail(newManager.getEmail()) != null ) {  
 return false;  
 }  
 if (!isValidPhoneNumber(newManager.getPhone\_number())) {  
 System.*out*.println(newManager.getPhone\_number());  
 return false;  
 }  
 newManager.setUsername(newManager.getUsername());  
 newManager.setSurname(newManager.getSurname());  
 newManager.setPassword(passwordEncoder.encode(newManager.getPassword()));  
 newManager.setPhone\_number(newManager.getPhone\_number());  
 newManager.setEmail(newManager.getEmail());  
 newManager.setRole(userRole.*MANAGER*.getRoleName());  
 usersRepository.save(newManager);  
 return true;  
 }  
 public Users findByUsername(String username) {  
 return usersRepository.findByUsername(username);  
 }  
 public Users findByEmail(String email){  
 return usersRepository.findByEmail(email);  
 }  
 public List<Users> searchClients(String query) {  
 return usersRepository.findClientsBySearchQuery(query);  
 }  
 public List<Users> findByRole(String role) {  
 return usersRepository.findByRole(role);  
 }  
 public List<Users> getAllPersonnel() {  
 return usersRepository.findPersonnelByRoles(List.*of*(userRole.*OPERATOR*.getRoleName(), userRole.*MANAGER*.getRoleName()));  
 }  
  
 public List<Users> searchPersonnel(String query) {  
 return usersRepository.searchPersonnelByQuery(query, List.*of*(userRole.*OPERATOR*.getRoleName(), userRole.*MANAGER*.getRoleName()));  
 }  
 public void updateUser(String email, Users updatedUser) {  
 Users user = findByEmail(email);  
 user.setUsername(updatedUser.getUsername());  
 user.setSurname(updatedUser.getSurname());  
 user.setEmail(updatedUser.getEmail());  
 user.setPhone\_number(updatedUser.getPhone\_number());  
 usersRepository.save(user);  
 }  
 public void deleteUserById(Long id) {  
 usersRepository.deleteById(id);  
 }  
}

AutoServiceApplication.java:

*// Класс для запуска автоматизированной информационной системы записей в автосервис*@SpringBootApplication  
public class AutoServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(AutoServiceApplication.class, args);  
 }  
  
}

addAuto.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Добавить автомобиль</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 600px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h1>Добавление автомобиля</h1>  
 <form th:action="@{/addAuto}" th:object="${auto}" method="post">  
 <div class="form-group">  
 <label for="brand">Бренд:</label>  
 <select id="brand" name="brand" class="form-control" onchange="loadModels()">  
 <option value="">Выберите бренд</option>  
 <option th:each="brand : ${brands}" th:value="${brand.name()}" th:text="${brand.name()}"></option>  
 </select>  
 <div th:if="${#fields.hasErrors('brand')}" th:errors="\*{brand}" class="text-danger"></div>  
 </div>  
  
 *<!-- Модель -->* <div class="form-group">  
 <label for="model">Модель:</label>  
 <select id="model" name="model" class="form-control" onchange="loadGenerations()">  
 <option value="">Выберите модель</option>  
 </select>  
 <div th:if="${#fields.hasErrors('model')}" th:errors="\*{model}" class="text-danger"></div>  
 </div>  
  
 *<!-- Поколение -->* <div class="form-group">  
 <label for="generation">Поколение:</label>  
 <select id="generation" name="generation" class="form-control">  
 <option value="">Выберите поколение</option>  
 </select>  
 <div th:if="${#fields.hasErrors('generation')}" th:errors="\*{generation}" class="text-danger"></div>  
 </div>  
  
 *<!-- Год выпуска -->* <div class="form-group">  
 <label for="year">Год:</label>  
 <input type="number" id="year" name="year" th:field="\*{year}" class="form-control" min="1900" max="2099" required>  
 <div th:if="${#fields.hasErrors('year')}" th:errors="\*{year}" class="text-danger"></div>  
 </div>  
  
 *<!-- VIN номер -->* <div class="form-group">  
 <label for="vinNumber">VIN номер:</label>  
 <input type="text" id="vinNumber" name="vinNumber" th:field="\*{vinNumber}" class="form-control" required minlength="17" maxlength="17">  
 <div th:if="${#fields.hasErrors('vinNumber')}" th:errors="\*{vinNumber}" class="text-danger"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="GOS\_number">Госномер:</label>  
 <input type="text" id="GOS\_number" name="GOS\_number"  
 th:field="\*{GOS\_number}" class="form-control"  
 placeholder="Например: А333КВ33" required>  
 <div th:if="${#fields.hasErrors('GOS\_number')}" th:errors="\*{GOS\_number}" class="text-danger"></div>  
 </div>  
  
 <button type="submit" class="btn">Добавить автомобиль</button>  
 </form>  
</div>  
  
<script>  
 function loadModels() {  
 const brand = document.getElementById('brand').value;  
 fetch(/models?brand=${brand})  
 .then(response => response.json())  
 .then(models => {  
 const modelSelect = document.getElementById('model');  
 modelSelect.innerHTML = '<option value="">Выберите модель</option>';  
 models.forEach(model => {  
 const option = document.createElement('option');  
 option.value = model.name;  
 option.text = model.name;  
 modelSelect.appendChild(option);  
 });  
 document.getElementById('generation').innerHTML = '<option value="">Выберите поколение</option>';  
 });  
 }  
  
 function loadGenerations() {  
 const brand = document.getElementById('brand').value;  
 const model = document.getElementById('model').value;  
 fetch(/generations?brand=${brand}&model=${model})  
 .then(response => response.json())  
 .then(generations => {  
 const generationSelect = document.getElementById('generation');  
 generationSelect.innerHTML = '<option value="">Выберите поколение</option>';  
 generations.forEach(generation => {  
 const option = document.createElement('option');  
 option.value = generation;  
 option.text = generation;  
 generationSelect.appendChild(option);  
 });  
 });  
 }  
</script>  
  
</body>  
</html>

addComment.html:

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <title>Добавить комментарий</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
  
 form {  
 width: 900px; /\* Статическая ширина формы \*/  
 margin: 50px auto;  
 padding: 30px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
  
 label {  
 display: block;  
 margin-bottom: 12px; /\* Увеличиваем отступ между метками и полями \*/  
 font-weight: bold;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
  
 input[type="text"], textarea {  
 width: 100%; /\* Устанавливаем ширину на 100%, чтобы поле растягивалось по всей ширине родительского блока \*/  
 padding: 12px;  
 margin-bottom: 20px;  
 border: 2px solid #222222;  
 border-radius: 4px;  
 background-color: #222222;  
 color: #ffcc00;  
 }  
  
 textarea {  
 height: 200px; /\* Статическая высота для текстового поля \*/  
 resize: none; /\* Отключаем возможность изменения размера \*/  
 }  
  
 input[type="text"]:focus, textarea:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
  
 button {  
 padding: 15px 25px;  
 background-color: #222222; /\* Черный фон \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 transition: background-color 0.3s ease;  
 }  
  
 button:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 </style>  
</head>  
<body>  
  
<h1>Добавить комментарий</h1>  
  
<form th:action="@{/manager/orders/addComment}" method="post">  
 <input type="hidden" name="orderId" th:value="${order.id}">  
 <label for="managerComment">Комментарий менеджера:</label>  
 <textarea id="managerComment" name="managerComment" rows="6" placeholder="Введите комментарий"></textarea>  
 <button type="submit">Сохранить</button>  
</form>  
  
</body>  
</html>

addScheduleSlot.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Добавить слот для записи</title>  
 <link href="https://cdn.jsdelivr.net/npm/fullcalendar@5.11.3/main.min.css" rel="stylesheet" />  
 <script src="https://cdn.jsdelivr.net/npm/fullcalendar@5.11.3/main.min.js"></script>  
 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 1000px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
  
 /\* Календарь \*/  
 #calendar {  
 width: 1000px;  
 height: 700px;  
 margin: 0 auto;  
 border: 1px solid #222222;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 padding: 10px;  
 background-color: #eabc05;  
 }  
  
 .selected-date {  
 background-color: #c99f06;  
 color: black;  
 }  
  
 .modal-content {  
 padding: 20px;  
 border: 1px solid #222222;  
 background-color: #eabc05; /\* Желтый фон для модального окна \*/  
 color: #222222;  
 }  
  
 #openFullDaySlots {  
 display: block;  
 margin: 20px auto;  
 padding: 10px 20px;  
 background-color: #222222;  
 color: #ffcc00;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 transition: background-color 0.3s ease;  
 }  
  
 #openFullDaySlots:hover {  
 background-color: #444444;  
 }  
 </style>  
</head>  
<body>  
<div class="container mt-4">  
 <h1>Добавить доступное время для записи</h1>  
 <div id="calendar"></div>  
 <div id="timeModal" class="modal" style="display: none;">  
 <div class="modal-content">  
 <h3>Выберите время</h3>  
 <form th:action="@{/manager/addScheduleSlot}" method="post">  
 <input type="hidden" id="selectedDate" name="date" />  
 <label for="time">Время:</label>  
 <input type="time" id="time" name="time" class="form-control" required>  
 <button type="submit" class="btn">Добавить слот</button>  
 </form>  
 </div>  
 </div>  
 <button id="openFullDaySlots" class="btn" style="margin-top: 10px;">Открыть записи на весь день</button>  
</div>  
  
<script th:inline="javascript">  
 // Подгружаем доступные слоты из модели  
 let slots = /\*[[${slots}]]\*/ [];  
 let selectedDate; // Для хранения выбранной даты  
  
 document.addEventListener('DOMContentLoaded', function() {  
 let calendarEl = document.getElementById('calendar');  
 let calendar = new FullCalendar.Calendar(calendarEl, {  
 initialView: 'dayGridMonth',  
 locale: 'ru',  
 displayEventTime: false, // Отключаем вывод времени в отдельной строке  
 events: slots.map(slot => {  
 let timeParts = slot.time.split(":");  
 let formattedTime = timeParts[0].replace(/^0/, '') + ":" + timeParts[1];  
  
 return {  
 title: 'Доступно: ' + formattedTime,  
 start: slot.date + 'T' + slot.time,  
 color: '#ff6666' // Красный цвет для уже заполненных слотов  
 };  
 }),  
 dateClick: function(info) {  
 let allDays = document.querySelectorAll('.fc-day');  
 allDays.forEach(day => {  
 day.classList.remove('selected-date');  
 });  
  
 info.dayEl.classList.add('selected-date');  
 selectedDate = info.dateStr;  
 document.getElementById('selectedDate').value = selectedDate;  
 document.getElementById('timeModal').style.display = 'block';  
 }  
 });  
 calendar.render();  
  
 document.getElementById('openFullDaySlots').addEventListener('click', function() {  
 if (!selectedDate) {  
 alert('Пожалуйста, выберите дату в календаре.');  
 return;  
 }  
  
 $.ajax({  
 url: '/manager/addFullDaySlots',  
 type: 'POST',  
 data: { date: selectedDate },  
 success: function(response) {  
 alert(response);  
 calendar.refetchEvents();  
 },  
 error: function(xhr) {  
 alert("Ошибка при добавлении слотов: " + xhr.responseText);  
 }  
 });  
 });  
 });  
</script>  
</body>  
</html>

allAutos.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Список всех автомобилей</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 margin-top: 20px;  
 }  
 .container {  
 max-width: 800px;  
 margin: 20px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .search-container {  
 margin-bottom: 20px;  
 display: flex;  
 flex-direction: column;  
 gap: 10px;  
 }  
 .search-container input[type="text"] {  
 width: 100%;  
 padding: 10px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 .search-container button {  
 padding: 10px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .search-container button:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
 table th, table td {  
 padding: 10px;  
 text-align: left;  
 border: 2px solid #222222; /\* Черные границы ячеек \*/  
 }  
 table th {  
 background-color: #222222; /\* Черный фон заголовков таблицы \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 table.table-striped tbody tr:nth-child(even) {  
 background-color: #ffcc00; /\* Желтый фон \*/  
 color: #222222; /\* Черный текст \*/  
}  
  
table.table-striped tbody tr:nth-child(odd) {  
 background-color: #eabc05; /\* Другой оттенок желтого для нечетных строк \*/  
 color: #222222; /\* Черный текст \*/  
}  
  
 </style>  
</head>  
<body>  
<div class="container">  
 <h1>Список всех автомобилей</h1>  
  
 <div class="search-container">  
 <form th:action="@{/manager/allAutos}" method="get">  
 <input type="text" name="brand" th:value="${brand}" placeholder="Поиск по бренду">  
 <input type="text" name="model" th:value="${model}" placeholder="Поиск по модели">  
 <input type="text" name="vinNumber" th:value="${vinNumber}" placeholder="Поиск по VIN номеру">  
 <input type="text" name="gosNumber" th:value="${gosNumber}" placeholder="Поиск по госномеру">  
 <input type="text" name="email" th:value="${email}" placeholder="Поиск по email клиента">  
 <button type="submit">Найти</button>  
 </form>  
 </div>  
  
 <table>  
 <thead>  
 <tr>  
 <th>Бренд</th>  
 <th>Модель</th>  
 <th>Поколение</th>  
 <th>Год</th>  
 <th>VIN номер</th>  
 <th>Госномер</th>  
 <th>Пользователь</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="auto : ${autos}">  
 <td th:text="${auto.brand}">Бренд</td>  
 <td th:text="${auto.model}">Модель</td>  
 <td th:text="${auto.generation}">Поколение</td>  
 <td th:text="${auto.year}">Год</td>  
 <td th:text="${auto.vinNumber}">VIN номер</td>  
 <td th:text="${auto.GOS\_number}">Госномер</td>  
 <td th:text="${auto.user.email}">Email пользователя</td>  
 </tr>  
 </tbody>  
 </table>  
</div>  
</body>  
</html>

clientOrders.html:

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <title>Обращения</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin: 20px 0;  
 background-color: #ffcc00; /\* Желтый фон таблицы \*/  
 border: 2px solid #222222; /\* Черные границы \*/  
 }  
 th, td {  
 padding: 8px;  
 text-align: left;  
 border: 1px solid #222222; /\* Черные границы ячеек \*/  
 }  
 th {  
 background-color: #222222; /\* Черный фон заголовков таблицы \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 td {  
 background-color: #222222; /\* Черный фон ячеек \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 text-decoration: none;  
 display: inline-block;  
 margin: 10px 0;  
 text-align: center;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .footer {  
 text-align: center;  
 margin: 20px 0;  
 }  
 td span.status {  
 display: inline-block;  
 padding: 4px;  
 border-radius: 4px;  
 color: #000000;  
 }  
 td span.status-4 {  
 background-color: #00ff00; /\* Зеленый для "Исполнен" \*/  
 }  
 td span.status-2 {  
 background-color: #ff00ff; /\* Желтый для "Подтвержден менеджером" \*/  
 }  
 td span.status-3 {  
 background-color: #ff6600; /\* Оранжевый для "На исполнении" \*/  
 }  
 td span.status-1 {  
 background-color: #FF0000; /\* Голубой для "Ожидает подтверждения менеджера" \*/  
 }  
 td .price {  
 background-color: transparent;  
 color: inherit;  
 }  
 a {  
 color: #222222; /\* Черный цвет текста ссылок \*/  
 text-decoration: none; /\* Убрать подчеркивание \*/  
 }  
 a:hover {  
 text-decoration: underline; /\* Подчеркивание при наведении \*/  
 }  
 </style>  
</head>  
<body>  
<h1>Ваши обращения</h1>  
  
<table>  
 <thead>  
 <tr>  
 <th>Автомобиль</th>  
 <th>Услуга</th>  
 <th>Дата и время</th>  
 <th>Статус</th>  
 <th>Цена</th>  
 <th>Ваш комментарий</th>  
 <th>Комментарий менеджера</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="order : ${orders}">  
 <td th:text="${order.auto.brand + ' ' + order.auto.model}">Автомобиль</td>  
 <td th:text="${order.serviceType}">Услуга</td>  
 <td th:text="${order.slot.date + ' ' + order.slot.time}">Дата и время</td>  
 <td>  
 <span th:switch="${order.status}">  
 <span th:case="1" class="status status-1">Ожидает подтверждения менеджера</span>  
 <span th:case="2" class="status status-2">Подтвержден менеджером</span>  
 <span th:case="3" class="status status-3">На исполнении</span>  
 <span th:case="4" class="status status-4">Исполнен</span>  
 </span>  
 </td>  
 <td>  
 <span th:text="${order.price != null ? order.price : 'Не установлена'}" class="price"></span>  
 </td>  
 <td th:text="${order.clientComment != null ? order.clientComment : 'Нет комментария'}">Комментарий</td>  
 <td th:text="${order.managerComment != null ? order.managerComment : 'Нет комментария'}">Комментарий менеджера</td>  
 </tr>  
 </tbody>  
</table>  
<div class="footer">  
 <a th:href="@{/home}" class="btn">На главную</a>  
</div>  
</body>  
</html>

confirmation.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Confirmation</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h2, h3 {  
 text-align: center;  
 color: #222222; /\* Черный цвет для заголовков \*/  
 }  
 p {  
 font-size: 16px;  
 margin: 8px 0;  
 }  
 .container {  
 max-width: 600px;  
 margin: 0 auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 text-align: center;  
 }  
 .button-group {  
 display: flex;  
 justify-content: center;  
 gap: 10px;  
 margin-top: 15px;  
 flex-wrap: wrap;  
 }  
 .button-group a {  
 text-decoration: none;  
 }  
 button {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 button:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h2>Добро пожаловать в систему записей в автосервис!</h2>  
 <p>Вы успешно зарегистрировались! Теперь вы можете прройти процедуру авторизации.</p>  
  
 <div class="button-group">  
 <a th:href="@{/login}">  
 <button>Перейти к авторизации</button>  
 </a>  
 </div>  
</div>  
  
</body>  
</html>

confirmationManage.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Менеджер зарегистрирован</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1, h2 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 600px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 display: block;  
 margin: 20px auto;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h2>Менеджер зарегистрирован!</h2>  
 <p>Поздравляем, менеджер успешно зарегистрирован в системе.</p>  
 <button class="btn" onclick="window.location.href='/home'">Перейти на главную страницу</button>  
</div>  
  
</body>  
</html>

confirmOrder.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Подтверждение записи</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 600px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 textarea {  
 background-color: #222222; /\* Черный фон для textarea \*/  
 color: #ffcc00; /\* Желтый текст в textarea \*/  
 border: 2px solid #222222;  
 border-radius: 5px;  
 padding: 8px;  
 font-size: 16px;  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h1>Подтверждение записи на услугу</h1>  
  
 <p><strong>Автомобиль:</strong> <span th:text="${auto.brand + ' ' + auto.model}"></span></p>  
 <p><strong>Услуга:</strong> <span th:text="${serviceType}"></span></p>  
  
 <p><strong>Дата и время:</strong> <span th:text="${formattedDateTime}"></span></p>  
  
 <form action="/confirmOrder" method="post">  
 <input type="hidden" name="slotId" th:value="${slot.id}" />  
 <input type="hidden" name="carId" th:value="${auto.id}" />  
 <input type="hidden" name="serviceType" th:value="${serviceType}" />  
  
 <div class="form-group">  
 <label for="comment">Комментарий:</label>  
 <textarea id="comment" name="comment" rows="4" cols="50" placeholder="Добавьте комментарий к заказу (необязательно)"></textarea>  
 </div>  
  
 <button type="submit" class="btn">Подтвердить запись</button>  
 </form>  
</div>  
  
</body>  
</html>

edit\_user.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Изменение данных</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 20px;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный заголовок \*/  
 }  
 form {  
 max-width: 600px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 }  
 label {  
 display: block;  
 font-weight: bold;  
 margin-bottom: 5px;  
 }  
 input[type="text"], input[type="email"] {  
 width: 100%;  
 padding: 10px;  
 margin-bottom: 15px;  
 border: 1px solid #222222; /\* Черная граница \*/  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон ввода \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 input[type="text"]:focus, input[type="email"]:focus {  
 border-color: #ffcc00; /\* Желтая граница при фокусе \*/  
 }  
 button[type="submit"] {  
 padding: 10px 20px;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст кнопки \*/  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 width: 100%;  
 }  
 button[type="submit"]:hover {  
 background-color: #444444; /\* Темно-серый на hover \*/  
 }  
 .error {  
 color: red;  
 font-size: 14px;  
 }  
 </style>  
</head>  
<body>  
<h1>Изменение данных</h1>  
<form th:action="@{/user/edit}" th:object="${user}" method="post">  
 <label for="username">Имя</label>  
 <input type="text" id="username" name="username" th:value="\*{username}" required>  
 <div th:if="${#fields.hasErrors('username')}" class="error" th:errors="\*{username}">Ошибка</div>  
  
 <label for="surname">Фамилия</label>  
 <input type="text" id="surname" name="surname" th:value="\*{surname}" required>  
 <div th:if="${#fields.hasErrors('surname')}" class="error" th:errors="\*{surname}">Ошибка</div>  
  
 <label for="email">Email</label>  
 <input type="email" id="email" name="email" th:value="\*{email}" required>  
 <div th:if="${#fields.hasErrors('email')}" class="error" th:errors="\*{email}">Ошибка</div>  
  
 <label for="phone\_number">Номер телефона</label>  
 <input type="text" id="phone\_number" name="phone\_number" th:value="\*{phone\_number}" required>  
 <div th:if="${#fields.hasErrors('phone\_number')}" class="error" th:errors="\*{phone\_number}">Ошибка</div>  
  
 <button type="submit">Сохранить</button>  
</form>  
</body>  
</html>

error.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Ошибка</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 text-align: center;  
 padding: 50px;  
 }  
 h1 {  
 font-size: 36px;  
 color: #222222; /\* Черный текст \*/  
 }  
 p {  
 font-size: 18px;  
 margin: 20px 0;  
 }  
 .btn {  
 padding: 10px 20px;  
 background-color: #222222; /\* Черный фон кнопки \*/  
 color: #ffcc00; /\* Желтый текст кнопки \*/  
 text-decoration: none;  
 border-radius: 5px;  
 transition: background-color 0.3s;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темно-серый фон при наведении \*/  
 }  
 </style>  
</head>  
<body>  
<h1>Произошла ошибка</h1>  
<p>К сожалению, произошла непредвиденная ошибка. Пожалуйста, попробуйте позже.</p>  
<a th:href="@{/}" class="btn">На главную</a>  
</body>  
</html>

home.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Главная страница</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h2, h3 {  
 text-align: center;  
 color: #222222; /\* Черный цвет для заголовков \*/  
 }  
 p {  
 font-size: 16px;  
 margin: 8px 0;  
 }  
 .container {  
 max-width: 600px;  
 margin: 0 auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 text-align: center;  
 }  
 .button-group {  
 display: flex;  
 justify-content: center;  
 gap: 10px;  
 margin-top: 15px;  
 flex-wrap: wrap;  
 }  
 .button-group a {  
 text-decoration: none;  
 }  
 button {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 button:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .section {  
 margin-top: 20px;  
 padding-top: 10px;  
 border-top: 1px solid #222222;  
 }  
 .logout {  
 margin-top: 20px;  
 display: block;  
 font-size: 16px;  
 color: #444444;  
 text-decoration: none;  
 }  
 .logout:hover {  
 text-decoration: underline;  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h2>Добро пожаловать, <span th:text="${user.username} + ' ' + ${user.surname}">[Имя Фамилия]</span>!</h2>  
 <p><strong>Email:</strong> <span th:text="${user.email}">[email]</span></p>  
 <p><strong>Телефон:</strong> <span th:text="${user.phone\_number}">[номер телефона]</span></p>  
 <p><strong>Роль:</strong> <span th:text="${user.role}">[роль]</span></p>  
 <div th:if="${isClient}" class="section">  
 <div class="button-group">  
 <a th:href="@{/user/edit}">  
 <button>Изменить данные</button>  
 </a>  
 <a th:href="@{/autos}">  
 <button>Мои авто</button>  
 </a>  
 <a th:href="@{/addAuto}">  
 <button>Добавить авто</button>  
 </a>  
 <a th:href="@{/selectAutoAndService}">  
 <button>Запись на услуги</button>  
 </a>  
 <a th:href="@{/orders}">  
 <button>История записей</button>  
 </a>  
 </div>  
 </div>  
  
 <div th:if="${isOperator}" class="section">  
 <h3>Доступные действия для оператора:</h3>  
 <div class="button-group">  
 <a th:href="@{/operatorBD/tables}">  
 <button>Открыть список таблиц</button>  
 </a>  
 <a th:href="@{/operatorBD/personnel}">  
 <button>Cписок персонала</button>  
 </a>  
 <a th:href="@{/operatorBD/registerManagerForm}">  
 <button>Зарегистрировать менеджера</button>  
 </a>  
 </div>  
 </div>  
  
 <div th:if="${isManager}" class="section">  
 <h3>Доступные действия для менеджера:</h3>  
 <div class="button-group">  
 <a th:href="@{/manager/allAutos}">  
 <button>Открыть список всех авто</button>  
 </a>  
 <a th:href="@{/manager/clients}">  
 <button>Список клиентов</button>  
 </a>  
 <a th:href="@{/manager/addScheduleSlot}">  
 <button>Добавить слот для записи</button>  
 </a>  
 <a th:href="@{/manager/viewSlots}">  
 <button>Посмотреть слоты для записи</button>  
 </a>  
 <a th:href="@{/manager/orders}">  
 <button>Заказы</button>  
 </a>  
 <a th:href="@{/manager/statistics}">  
 <button>Статистика</button>  
 </a>  
 </div>  
 </div>  
  
 <a th:href="@{/login}" class="logout">Выйти</a>  
</div>  
  
</body>  
</html>

login.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Авторизация в системе записей в автосервис</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h2 {  
 color: #222222; /\* Черный цвет для заголовка \*/  
 text-align: center;  
 margin-top: 50px;  
 }  
 .form-container {  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 border: 2px solid #222222; /\* Черная рамка \*/  
 max-width: 400px;  
 margin: 0 auto;  
 padding: 20px;  
 border-radius: 10px;  
 text-align: center;  
 }  
 .form-group {  
 margin-bottom: 15px;  
 display: flex;  
 flex-direction: column;  
 text-align: left;  
 }  
 label {  
 margin-bottom: 5px;  
 font-weight: bold;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 input[type="text"], input[type="password"] {  
 padding: 10px;  
 font-size: 14px;  
 border: 1px solid #444444;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон для полей ввода \*/  
 color: #ffcc00; /\* Желтые буквы \*/  
 }  
 input[type="text"]:focus, input[type="password"]:focus {  
 border-color: #ffcc00; /\* Желтая рамка при фокусе \*/  
 outline: none;  
 }  
 button {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222;  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 button:hover {  
 background-color: #444444; /\* Тёмный черный на hover \*/  
 }  
 .error {  
 color: red;  
 margin-top: 10px;  
 }  
 a {  
 color: #222222;  
 text-decoration: none;  
 }  
 a:hover {  
 text-decoration: underline;  
 }  
 </style>  
</head>  
<body>  
  
<h2>Вход в систему записей в автосервис</h2>  
  
<div class="form-container">  
 <form th:action="@{/login}" method="post">  
 <div class="form-group">  
 <label for="email">Email:</label>  
 <input type="text" id="email" name="username" required>  
 </div>  
 <div class="form-group">  
 <label for="password">Пароль:</label>  
 <input type="password" id="password" name="password" required>  
 </div>  
 <button type="submit">Войти</button>  
 <p class="error" th:if="${param.error}" th:text="'Неправильный email или пароль'"></p>  
 </form>  
 <p>Нет аккаунта? <a th:href="@{/register}">Зарегистрироваться</a></p>  
</div>  
  
</body>  
</html>

managerClients.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Список клиентов</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 20px;  
 }  
 h1 {  
 color: #222222; /\* Черный цвет для заголовка \*/  
 text-align: center;  
 margin-bottom: 20px;  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 background-color: #ffcc00; /\* Желтый фон таблицы \*/  
 }  
 th, td {  
 padding: 10px;  
 border: 1px solid #222222; /\* Черная граница \*/  
 text-align: left;  
 }  
 th {  
 background-color: #222222; /\* Черный фон заголовков таблицы \*/  
 color: #ffcc00; /\* Желтый текст в заголовках \*/  
 }  
 td {  
 color: #222222; /\* Черный текст в ячейках \*/  
 }  
 .btn {  
 padding: 10px 20px;  
 background-color: #222222; /\* Черный фон кнопки \*/  
 border: 1px solid #222222;  
 color: #ffcc00; /\* Желтый текст кнопки \*/  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 text-align: center;  
 border-radius: 5px;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темно-серый фон при наведении \*/  
 }  
 .search-container {  
 margin-bottom: 20px;  
 display: flex;  
 justify-content: center;  
 gap: 10px;  
 }  
 input[type="text"] {  
 padding: 10px;  
 font-size: 14px;  
 width: 300px;  
 background-color: #222222; /\* Черный фон для ввода \*/  
 color: #ffcc00; /\* Желтый текст ввода \*/  
 border: 1px solid #222222; /\* Черная граница \*/  
 border-radius: 5px;  
 }  
 input[type="text"]:focus {  
 border-color: #ffcc00; /\* Желтая граница при фокусе \*/  
 outline: none;  
 }  
 button[type="submit"] {  
 padding: 10px 20px;  
 font-size: 14px;  
 background-color: #222222; /\* Черный фон кнопки \*/  
 color: #ffcc00; /\* Желтый текст кнопки \*/  
 border: 1px solid #222222;  
 border-radius: 5px;  
 cursor: pointer;  
 }  
 button[type="submit"]:hover {  
 background-color: #444444; /\* Темно-серый фон при наведении \*/  
 }  
 .footer {  
 margin-top: 20px;  
 text-align: center;  
 }  
 </style>  
</head>  
<body>  
<h1>Список клиентов</h1>  
  
<div class="search-container">  
 <form th:action="@{/manager/clients}" method="get">  
 <input type="text" name="search" th:value="${searchQuery}" placeholder="Поиск по имени, фамилии, телефону или email">  
 <button type="submit" class="btn">Найти</button>  
 </form>  
</div>  
  
<table>  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Имя</th>  
 <th>Фамилия</th>  
 <th>Email</th>  
 <th>Телефон</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="client : ${clients}">  
 <td th:text="${client.id}"></td>  
 <td th:text="${client.username}"></td>  
 <td th:text="${client.surname}"></td>  
 <td th:text="${client.email}"></td>  
 <td th:text="${client.phone\_number}"></td>  
 </tr>  
 </tbody>  
</table>  
  
<div class="footer">  
 <a th:href="@{/home}" class="btn">На главную</a>  
</div>  
</body>  
</html>

managerOrders.html:

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <title>Заказы</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный текст заголовка \*/  
 }  
 .container {  
 max-width: 1200px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222;  
 margin-right: 10px;  
 width: 150px;  
 }  
 .form-control {  
 width: 100%;  
 max-width: 450px;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 .form-control:focus {  
 border-color: #444444;  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 10px 20px;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444;  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет \*/  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
 th, td {  
 border: 1px solid #222222;  
 padding: 10px;  
 text-align: left;  
 }  
 th {  
 background-color: #222222; /\* Черный фон \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 }  
 td {  
 background-color: #ffcc00; /\* Желтый фон \*/  
 color: #222222; /\* Черный текст \*/  
 }  
 select {  
 background-color: #222222;  
 color: #ffcc00;  
 padding: 8px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 }  
 button:disabled {  
 background-color: #ccc;  
 cursor: not-allowed;  
 }  
 .btn-confirm {  
 background-color: #66cc66;  
 color: black;  
 }  
 .btn-progress {  
 background-color: #ffa500;  
 color: black;  
 }  
 .btn-complete {  
 background-color: #007bff;  
 color: black;  
 }  
 form button {  
 background-color: #222222; /\* Черный фон \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 padding: 10px 20px;  
 cursor: pointer;  
 transition: background-color 0.3s ease;  
}  
  
form button:hover {  
 background-color: #444444;  
}  
 </style>  
</head>  
<body>  
<h1>Список заказов</h1>  
  
*<!-- Форма для поиска заказов -->*<form action="/manager/orders" method="get">  
 <div class="form-group">  
 <label for="orderId">ID заказа:</label>  
 <input type="number" name="orderId" id="orderId" class="form-control" placeholder="ID заказа">  
 </div>  
 <div class="form-group">  
 <label for="clientEmail">Email клиента:</label>  
 <input type="text" name="clientEmail" id="clientEmail" class="form-control" placeholder="Email клиента">  
 </div>  
 <div class="form-group">  
 <label for="brand">Бренд:</label>  
 <select name="brand" id="brand" class="form-control">  
 <option value="">-- Выберите бренд --</option>  
 <option value="TOYOTA">TOYOTA</option>  
 <option value="HONDA">HONDA</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label for="model">Модель:</label>  
 <select name="model" id="model" class="form-control">  
 <option value="">-- Выберите модель --</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label for="year">Год:</label>  
 <input type="number" name="year" id="year" class="form-control" placeholder="Год">  
 </div>  
 <div class="form-group">  
 <label for="vinNumber">VIN:</label>  
 <input type="text" name="vinNumber" id="vinNumber" class="form-control" placeholder="VIN">  
 </div>  
 <div class="form-group">  
 <label for="GOS\_number">Госномер:</label>  
 <input type="text" name="GOS\_number" id="GOS\_number" class="form-control" placeholder="Госномер">  
 </div>  
 <div class="form-group">  
 <label for="date">Дата:</label>  
 <input type="date" name="date" id="date" class="form-control" placeholder="Дата заказа">  
 </div>  
 <div class="form-group">  
 <label for="status">Статус:</label>  
 <select name="status" id="status" class="form-control">  
 <option value="">-- Выберите статус --</option>  
 <option value="1">Ожидает подтверждения</option>  
 <option value="2">Подтвержден</option>  
 <option value="3">На исполнении</option>  
 <option value="4">Исполнен</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label for="serviceType">Вид услуги:</label>  
 <select name="serviceType" id="serviceType" class="form-control">  
 <option value="">-- Выберите услугу --</option>  
 <option value="Замена масла">Замена масла</option>  
 <option value="Диагностика">Диагностика</option>  
 <option value="Ремонт двигателя">Ремонт двигателя</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <button type="submit" class="btn">Поиск</button>  
 </div>  
</form>  
  
*<!-- Таблица с заказами -->*<table>  
 <thead>  
 <tr>  
 <th>Заказ ID</th>  
 <th>Клиент</th>  
 <th>Автомобиль</th>  
 <th>Год</th>  
 <th>VIN</th>  
 <th>Госномер</th>  
 <th>Услуга</th>  
 <th>Дата и время</th>  
 <th>Статус</th>  
 <th>Цена</th>  
 <th>Установить цену</th>  
 <th>Комментарий клиента</th>  
 <th>Комментарий менеджера</th>  
 <th>Изменение статуса</th>  
 <th>Заказ-наряд</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="order : ${orders}">  
 <td th:text="${order.id}"></td>  
 <td th:text="${order.client.email}"></td>  
 <td th:text="${order.auto.brand + ' ' + order.auto.model + ' ' + order.auto.generation}"></td>  
 <td th:text="${order.auto.year}"></td>  
 <td th:text="${order.auto.vinNumber}"></td>  
 <td th:text="${order.auto.GOS\_number}"></td>  
 <td th:text="${order.serviceType}"></td>  
 <td th:text="${order.slot.date + ' ' + order.slot.time}"></td>  
 <td th:switch="${order.status}">  
 <span th:case="1">Ожидает подтверждения</span>  
 <span th:case="2">Подтвержден</span>  
 <span th:case="3">На исполнении</span>  
 <span th:case="4">Исполнен</span>  
 </td>  
 <td>  
 <span th:text="${order.price != null ? order.price : 'Не установлена'}"></span>  
 </td>  
 <td>  
 <form th:action="@{/manager/orders/setPrice}" method="post">  
 <input type="hidden" name="orderId" th:value="${order.id}">  
 <input type="number" name="price" class="form-control" placeholder="Введите цену" step="0.01" min="0" required>  
 <button type="submit">Сохранить</button>  
 </form>  
 </td>  
  
 <td th:text="${order.clientComment != null ? order.clientComment : 'Нет комментария'}"></td>  
 <td>  
 <form th:action="@{/manager/orders/addComment}" method="get" style="display:inline;">  
 <input type="hidden" name="orderId" th:value="${order.id}">  
 <button type="submit">Добавить комментарий</button>  
 </form>  
 </td>  
 <td>  
 <form th:action="@{/manager/orders/updateStatus}" method="post">  
 <input type="hidden" name="orderId" th:value="${order.id}">  
 <button type="submit" class="btn-confirm" th:disabled="${order.status != 1}" name="status" value="2">Подтвердить</button>  
 <button type="submit" class="btn-progress" th:disabled="${order.status != 2}" name="status" value="3">На исполнение</button>  
 <button type="submit" class="btn-complete" th:disabled="${order.status != 3}" name="status" value="4">Исполнен</button>  
 </form>  
 </td>  
 <td>  
 <form th:action="@{/manager/orders/exportDocx}" method="post">  
 <input type="hidden" name="orderId" th:value="${order.id}">  
 <button type="submit">Составить заказ-наряд</button>  
 </form>  
 </td>  
 </tr>  
 </tbody>  
</table>  
  
<div class="footer">  
 <a th:href="@{/home}" class="btn">На главную</a>  
</div>  
  
<script>  
 // Добавление динамической загрузки моделей в зависимости от выбранного бренда  
 document.getElementById("brand").addEventListener("change", function() {  
 const brand = this.value;  
 const modelSelect = document.getElementById("model");  
 modelSelect.innerHTML = "<option value=''>-- Выберите модель --</option>";  
  
 let models = [];  
  
 if (brand === "TOYOTA") {  
 models = ["Camry", "Corolla"];  
 } else if (brand === "HONDA") {  
 models = ["Civic", "Accord"];  
 }  
  
 models.forEach(function(model) {  
 const option = document.createElement("option");  
 option.value = model;  
 option.textContent = model;  
 modelSelect.appendChild(option);  
 });  
 });  
</script>  
  
</body>  
</html>

managerStatistics.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Статистика за месяц</title>  
 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 800px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
 canvas {  
 display: block;  
 width: 800px !important;  
 height: 400px !important;  
 }  
 </style>  
</head>  
<body>  
<div class="container">  
 <h1>Статистика за <span th:text="${month} + '-' + ${year}"></span></h1>  
  
 <p>Количество заказов: <span th:text="${statistics['totalOrders']}"></span></p>  
 <p>Средняя стоимость заказов: <span th:text="${statistics['averagePrice']}"></span> руб.</p>  
 <p>Общая сумма заказов: <span th:text="${statistics['totalPrice']}"></span> руб.</p>  
  
 <canvas id="ordersChart" width="600" height="400"></canvas>  
  
 <form class="form-group">  
 <button type="button" class="btn" onclick="window.location.href='/home'">Вернуться на главную</button>  
 </form>  
</div>  
  
<script th:inline="javascript">  
 let ordersData = /\*[[${statistics['ordersPerDay']}]]\*/ {};  
 let labels = Object.keys(ordersData);  
 let data = Object.values(ordersData);  
  
 const ctx = document.getElementById('ordersChart').getContext('2d');  
 new Chart(ctx, {  
 type: 'bar',  
 data: {  
 labels: labels,  
 datasets: [{  
 label: 'Количество заказов',  
 data: data,  
 backgroundColor: 'rgba(255, 0, 0, 0.2)',  
 borderColor: 'rgba(255, 0, 0, 1)',  
 borderWidth: 1  
 }]  
 },  
 options: {  
 scales: {  
 y: {  
 beginAtZero: true  
 }  
 }  
 }  
 });  
</script>  
  
</body>  
</html>

managerStatsForm.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Статистика</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 600px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h1>Выберите месяц и год для выбора статистики</h1>  
 <form action="/manager/statistics/results" method="get">  
 <div class="form-group">  
 <label for="month">Месяц:</label>  
 <input type="number" id="month" name="month" class="form-control" min="1" max="12" required>  
 </div>  
  
 <div class="form-group">  
 <label for="year">Год:</label>  
 <input type="number" id="year" name="year" class="form-control" min="2000" max="2100" required>  
 </div>  
  
 <button type="submit" class="btn">Показать статистику</button>  
 </form>  
 <form class="form-group">  
 <button type="button" class="btn" onclick="window.location.href='/home'">Вернуться на главную</button>  
 </form>  
</div>  
  
</body>  
</html>

okOrder.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Запись зарегистрирована</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 text-align: center;  
 }  
  
 h1 {  
 color: #222222; /\* Черный цвет заголовка \*/  
 margin-top: 50px;  
 }  
 .button {  
 display: inline-block;  
 padding: 12px 20px;  
 font-size: 16px;  
 color: #ffcc00; /\* Желтый текст \*/  
 background-color: #222222; /\* Черная кнопка \*/  
 border: none;  
 border-radius: 5px;  
 text-decoration: none;  
 cursor: pointer;  
 transition: background-color 0.3s ease;  
 }  
 .button:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .container {  
 margin-top: 50px;  
 }  
 </style>  
</head>  
<body>  
<h1>Ваша запись на услуги зарегистрирована</h1>  
<a th:href="@{/home}" class="button">На главную страницу</a>  
</body>  
</html>

operatorTableData.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Данные таблицы</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 1000px;  
 margin: 50px auto;  
 padding: 20px;  
  
  
 background-color: #eabc05; /\* Желтый фон формы \*/  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
 th, td {  
 padding: 12px;  
 text-align: left;  
 border: 1px solid #222222;  
 }  
 th {  
 background-color: #222222; /\* Черный фон для заголовков \*/  
 color: #ffcc00; /\* Желтый текст для заголовков \*/  
 }  
 tr:nth-child(even) {  
 background-color: #f2f2f2;  
 }  
 tr:hover {  
 background-color: #ddd; /\* Легкое выделение строки при наведении \*/  
 }  
 a {  
 color: #222222;  
 text-decoration: none;  
 font-size: 16px;  
 display: inline-block;  
 margin-top: 20px;  
 }  
 a:hover {  
 text-decoration: underline; /\* Подчеркивание при наведении \*/  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h1>Данные таблицы [[${tableName}]]</h1>  
 <table>  
 <tr th:each="row : ${tableData}">  
 <td th:each="cell : ${row}">[[${cell}]]</td>  
 </tr>  
 </table>  
 <a th:href="@{'/operatorBD/tables'}">Назад к таблицам</a>  
</div>  
  
</body>  
</html>

operatorTablesList.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Таблицы базы данных</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 600px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
 ul {  
 list-style-type: none;  
 padding: 0;  
 }  
 li {  
 margin: 10px 0;  
 }  
 a {  
 color: #222222; /\* Черный текст для ссылок \*/  
 text-decoration: none;  
 font-size: 18px;  
 }  
 a:hover {  
 text-decoration: underline;  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h1>Список таблиц</h1>  
 <ul>  
 <li th:each="table : ${tables}">  
 <a th:href="@{'/operatorBD/table/' + ${table}}">[[${table}]]</a>  
 </li>  
 </ul>  
 <a th:href="@{/login}" class="btn">Выйти</a>  
</div>  
  
</body>  
</html>

personnelList.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Список персонала</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 20px;  
 }  
 h1 {  
 color: #222222; /\* Черный цвет для заголовка \*/  
 text-align: center;  
 margin-bottom: 20px;  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 background-color: #ffcc00; /\* Желтый фон таблицы \*/  
 }  
 th, td {  
 padding: 10px;  
 border: 1px solid #222222; /\* Черная граница \*/  
 text-align: left;  
 }  
 th {  
 background-color: #222222; /\* Черный фон заголовков таблицы \*/  
 color: #ffcc00; /\* Желтый текст в заголовках \*/  
 }  
 td {  
 color: #222222; /\* Черный текст в ячейках \*/  
 }  
 .btn {  
 padding: 10px 20px;  
 background-color: #222222; /\* Черный фон кнопки \*/  
 border: 1px solid #222222;  
 color: #ffcc00; /\* Желтый текст кнопки \*/  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 text-align: center;  
 border-radius: 5px;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темно-серый фон при наведении \*/  
 }  
 .delete-btn {  
 background-color: #ff0000 !important;  
 color: #ffffff !important;  
 }  
 .delete-btn:hover {  
 background-color: #cc0000 !important;  
 }  
 .search-container {  
 margin-bottom: 20px;  
 display: flex;  
 justify-content: center;  
 gap: 10px;  
 }  
 input[type="text"] {  
 padding: 10px;  
 font-size: 14px;  
 width: 300px;  
 background-color: #222222; /\* Черный фон для ввода \*/  
 color: #ffcc00; /\* Желтый текст ввода \*/  
 border: 1px solid #222222; /\* Черная граница \*/  
 border-radius: 5px;  
 }  
 input[type="text"]:focus {  
 border-color: #ffcc00; /\* Желтая граница при фокусе \*/  
 outline: none;  
 }  
 button[type="submit"] {  
 padding: 10px 20px;  
 font-size: 14px;  
 background-color: #222222; /\* Черный фон кнопки \*/  
 color: #ffcc00; /\* Желтый текст кнопки \*/  
 border: 1px solid #222222;  
 border-radius: 5px;  
 cursor: pointer;  
 }  
 button[type="submit"]:hover {  
 background-color: #444444; /\* Темно-серый фон при наведении \*/  
 }  
 .footer {  
 margin-top: 20px;  
 text-align: center;  
 }  
 </style>  
</head>  
<body>  
<h1>Список персонала</h1>  
  
<div class="search-container">  
 <form th:action="@{/manager/clients}" method="get">  
 <input type="text" name="search" th:value="${searchQuery}" placeholder="Поиск по имени, фамилии, телефону или email">  
 <button type="submit" class="btn">Найти</button>  
 </form>  
</div>  
  
<table>  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Имя</th>  
 <th>Фамилия</th>  
 <th>Роль</th>  
 <th>Email</th>  
 <th>Телефон</th>  
 <th>Действия</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="client : ${clients}">  
 <td th:text="${client.id}"></td>  
 <td th:text="${client.username}"></td>  
 <td th:text="${client.surname}"></td>  
 <td th:text="${client.role}"></td>  
 <td th:text="${client.email}"></td>  
 <td th:text="${client.phone\_number}"></td>  
 <td>  
 <form th:if="${client.role == 'MANAGER'}" th:action="@{/operatorBD/deleteClient}" method="post" th:object="${client}">  
 <input type="hidden" name="id" th:value="${client.id}"/>  
 <button type="submit" class="btn delete-btn">Удалить</button>  
 </form>  
 </td>  
  
 </tr>  
 </tbody>  
</table>  
  
<div class="footer">  
 <a th:href="@{/home}" class="btn">На главную</a>  
</div>  
</body>  
</html>

register.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Регистрация пользователя</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 color: #222222; /\* Черный цвет для заголовка \*/  
 text-align: center;  
 margin-top: 50px;  
 }  
 .form-container {  
 max-width: 400px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #ffcc00; /\* Желтый фон контейнера \*/  
 border-radius: 10px;  
 border: 1px solid #222222; /\* Черные границы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 display: flex;  
 flex-direction: column;  
 }  
 label {  
 margin-bottom: 5px;  
 font-weight: bold;  
 color: #222222; /\* Черный текст для меток \*/  
 }  
 input[type="text"], input[type="password"], input[type="email"] {  
 padding: 10px;  
 font-size: 14px;  
 background-color: #222222; /\* Черный фон для полей ввода \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: 1px solid #222222; /\* Черные границы \*/  
 border-radius: 5px;  
 }  
 input[type="text"]:focus, input[type="password"]:focus, input[type="email"]:focus {  
 border-color: #ffcc00; /\* Желтая граница при фокусе \*/  
 outline: none;  
 }  
 button {  
 padding: 10px 15px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черный фон для кнопки \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: 1px solid #222222; /\* Черная граница \*/  
 border-radius: 5px;  
 }  
 button:hover {  
 background-color: #444444; /\* Темный фон при наведении \*/  
 }  
 .error {  
 color: red;  
 font-size: 12px;  
 margin-top: 5px;  
 }  
 p {  
 text-align: center;  
 }  
 a {  
 color: #222222;  
 text-decoration: none;  
 }  
 a:hover {  
 text-decoration: underline;  
 }  
 </style>  
</head>  
<body>  
<h1>Регистрация пользователя</h1>  
  
<div class="form-container">  
 <form th:action="@{/register}" th:object="${Users}" method="post">  
  
 <div th:if="${error}" class="error" style="text-align: center;">  
 <p th:text="${error}"></p>  
 </div>  
  
 <div class="form-group">  
 <label for="username">Имя:</label>  
 <input type="text" id="username" th:field="\*{username}" required>  
 <div th:if="${#fields.hasErrors('username')}" th:errors="\*{username}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="surname">Фамилия:</label>  
 <input type="text" id="surname" th:field="\*{surname}" required>  
 <div th:if="${#fields.hasErrors('surname')}" th:errors="\*{surname}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="password">Пароль:</label>  
 <input type="password" id="password" th:field="\*{password}" required>  
 <div th:if="${#fields.hasErrors('password')}" th:errors="\*{password}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="phone\_number">Номер телефона:</label>  
 <input type="text" id="phone\_number" th:field="\*{phone\_number}" required placeholder="+7 (\_\_\_) \_\_\_ \_\_ \_\_">  
 <div th:if="${#fields.hasErrors('phone\_number')}" th:errors="\*{phone\_number}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="email">Email:</label>  
 <input type="email" id="email" th:field="\*{email}" required>  
 <div th:if="${#fields.hasErrors('email')}" th:errors="\*{email}" class="error"></div>  
 </div>  
  
 <button type="submit">Зарегистрироваться</button>  
 </form>  
 <p><a th:href="@{/login}">Назад</a></p>  
</div>  
  
<script>  
 document.getElementById('phone\_number').addEventListener('input', function (e) {  
 const x = e.target.value.replace(/\D/g, '').match(/(\d{1})(\d{0,3})(\d{0,3})(\d{0,2})(\d{0,2})/);  
 e.target.value = x ? `+${x[1]} (${x[2]}) ${x[3]}-${x[4]}-${x[5]}` : '';  
 });  
</script>  
</body>  
</html>

registerManager.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Регистрация менеджера</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .form-container {  
 max-width: 400px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 input[type="text"], input[type="password"], input[type="email"] {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 input[type="text"]:focus, input[type="password"]:focus, input[type="email"]:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 button {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 button:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .error {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 font-size: 12px;  
 }  
 a {  
 color: #222222;  
 text-decoration: none;  
 transition: color 0.3s;  
 }  
 a:hover {  
 color: #444444;  
 }  
 </style>  
</head>  
<body>  
<h1 style="text-align: center;">Регистрация нового менеджера</h1>  
  
<div class="form-container">  
 <form th:action="@{/operatorBD/registerManager}" th:object="${Manager}" method="post">  
  
 <div th:if="${message}" class="error" style="text-align: center;">  
 <p th:text="${message}"></p>  
 </div>  
  
 <div class="form-group">  
 <label for="username">Имя:</label>  
 <input type="text" id="username" th:field="\*{username}" required>  
 <div th:if="${#fields.hasErrors('username')}" th:errors="\*{username}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="surname">Фамилия:</label>  
 <input type="text" id="surname" th:field="\*{surname}" required>  
 <div th:if="${#fields.hasErrors('surname')}" th:errors="\*{surname}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="password">Пароль:</label>  
 <input type="password" id="password" th:field="\*{password}" required>  
 <div th:if="${#fields.hasErrors('password')}" th:errors="\*{password}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="phone\_number">Номер телефона:</label>  
 <input type="text" id="phone\_number" th:field="\*{phone\_number}" required placeholder="+7 (\_\_\_) \_\_\_ \_\_ \_\_">  
 <div th:if="${#fields.hasErrors('phone\_number')}" th:errors="\*{phone\_number}" class="error"></div>  
 </div>  
  
 <div class="form-group">  
 <label for="email">Email:</label>  
 <input type="email" id="email" th:field="\*{email}" required>  
 <div th:if="${#fields.hasErrors('email')}" th:errors="\*{email}" class="error"></div>  
 </div>  
  
 <button type="submit">Зарегистрировать менеджера</button>  
 </form>  
 <p><a th:href="@{/home}">Назад</a></p>  
</div>  
  
<script>  
 document.getElementById('phone\_number').addEventListener('input', function (e) {  
 const x = e.target.value.replace(/\D/g, '').match(/(\d{1})(\d{0,3})(\d{0,3})(\d{0,2})(\d{0,2})/);  
 e.target.value = x ? `+${x[1]} (${x[2]}) ${x[3]}-${x[4]}-${x[5]}` : '';  
 });  
</script>  
</body>  
</html>

selectAutoAndService.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Выбор автомобиля и услуги</title>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 600px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-group label {  
 font-size: 16px;  
 color: #222222; /\* Черный цвет для меток \*/  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px;  
 font-size: 16px;  
 border: 2px solid #222222;  
 border-radius: 5px;  
 background-color: #222222; /\* Черный фон поля \*/  
 color: #ffcc00; /\* Желтый цвет текста \*/  
 }  
 .form-control:focus {  
 border-color: #444444; /\* Темный цвет при фокусе \*/  
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
 </style>  
</head>  
<body>  
  
<div class="container">  
 <h1>Выберите автомобиль и услугу</h1>  
  
 <form action="/selectAutoAndService" method="post">  
 <div class="form-group">  
 <label for="carId">Ваши автомобили:</label>  
 <select name="carId" id="carId" class="form-control" required>  
 <option value="" disabled selected>Выберите автомобиль</option>  
 <th:block th:each="auto : ${userAutos}">  
 <option th:value="${auto.id}" th:text="${auto.brand + ' ' + auto.model}"></option>  
 </th:block>  
 </select>  
 </div>  
  
 <div class="form-group">  
 <label for="serviceType">Выберите услугу:</label>  
 <select name="serviceType" id="serviceType" class="form-control" required>  
 <option value="" disabled selected>Выберите услугу</option>  
 <option value="Замена масла">Замена масла</option>  
 <option value="Диагностика">Диагностика</option>  
 <option value="Ремонт двигателя">Ремонт двигателя</option>  
 *<!-- Добавить другие услуги !!!-->* </select>  
 </div>  
  
 <button type="submit" class="btn">Продолжить</button>  
 </form>  
</div>  
  
</body>  
</html>

selectSlot.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Выбор времени записи</title>  
 <link href="https://cdn.jsdelivr.net/npm/fullcalendar@5.11.3/main.min.css" rel="stylesheet" />  
 <script src="https://cdn.jsdelivr.net/npm/fullcalendar@5.11.3/main.min.js"></script>  
 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 1000px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
  
 /\* Календарь \*/  
 #calendar {  
 width: 1000px;  
 height: 700px;  
 margin: 0 auto;  
 border: 1px solid #222222;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 padding: 10px;  
 background-color: #eabc05;  
 }  
  
 .selected-date {  
 background-color: #c99f06;  
 color: black;  
 }  
  
 .modal-content {  
 padding: 20px;  
 border: 1px solid #222222;  
 background-color: #eabc05; /\* Желтый фон для модального окна \*/  
 color: #222222;  
 }  
  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
  
 .text-danger {  
 color: #ff0000; /\* Красный цвет для ошибок \*/  
 }  
 </style>  
</head>  
<body>  
<div class="container mt-4">  
 <h1>Выберите время для записи</h1>  
 <div id="calendar"></div>  
</div>  
  
<script th:inline="javascript">  
 let slots = /\*[[${slots}]]\*/ [];  
 let carId = /\*[[${carId}]]\*/;  
 let serviceType = /\*[[${serviceType}]]\*/;  
  
 document.addEventListener('DOMContentLoaded', function() {  
 let calendarEl = document.getElementById('calendar');  
 let today = new Date(); // Текущая дата  
  
 let calendar = new FullCalendar.Calendar(calendarEl, {  
 initialView: 'dayGridMonth',  
 locale: 'ru',  
 displayEventTime: false,  
 events: slots  
 .filter(slot => {  
 // Преобразуем дату слота в объект Date  
 let slotDate = new Date(slot.date);  
 return slotDate >= today; // Исключаем прошлые даты  
 })  
 .map(slot => {  
 let timeParts = slot.time.split(":");  
 let formattedTime = timeParts[0].replace(/^0/, '') + ":" + timeParts[1];  
  
 return {  
 title: 'Доступно: ' + formattedTime,  
 start: slot.date + 'T' + slot.time,  
 color: '#66cc66', // Зеленый для доступных слотов  
 textColor: '#222222',  
 url: '/confirmOrder?slotId=' + slot.id + '&carId=' + carId + '&serviceType=' + serviceType  
 };  
 }),  
 eventClick: function(info) {  
 info.jsEvent.preventDefault();  
  
 if (info.event.url) {  
 window.location.href = info.event.url;  
 }  
 }  
 });  
  
 calendar.render();  
 });  
</script>  
</body>  
</html>

userAutos.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Мои автомобили</title>  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 color: #222222; /\* Черный цвет для заголовка \*/  
 text-align: center;  
 margin-top: 50px;  
 }  
 .container {  
 max-width: 800px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #ffcc00; /\* Желтый фон контейнера \*/  
 border-radius: 10px;  
 }  
 .alert {  
 background-color: #f8d7da; /\* Красный фон для ошибок \*/  
 color: #721c24; /\* Красный текст \*/  
 border: 1px solid #f5c6cb;  
 padding: 10px;  
 border-radius: 5px;  
 }  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
 table th, table td {  
 padding: 12px;  
 text-align: left;  
 border: 1px solid #222222; /\* Черные линии между строками \*/  
 }  
 table th {  
 background-color: #222222; /\* Черный фон для заголовков таблицы \*/  
 color: #ffcc00; /\* Желтые буквы в заголовках \*/  
 }  
 table td {  
 background-color: #222222; /\* Черный фон для ячеек \*/  
 color: #ffcc00; /\* Желтые буквы в ячейках \*/  
 }  
 table tr:hover {  
 background-color: #444444; /\* Темный фон при наведении \*/  
 }  
 </style>  
</head>  
<body>  
<div class="container mt-5">  
 <h1>Мои автомобили</h1>  
  
 <div th:if="${error}" class="alert alert-danger" th:text="${error}"></div>  
  
 <table class="table table-striped mt-4" th:if="${autos != null}">  
 <thead>  
 <tr>  
 <th>Бренд</th>  
 <th>Модель</th>  
 <th>Поколение</th>  
 <th>Год</th>  
 <th>VIN номер</th>  
 <th>Госномер</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="auto : ${autos}">  
 <td th:text="${auto.brand}">Бренд</td>  
 <td th:text="${auto.model}">Модель</td>  
 <td th:text="${auto.generation}">Поколение</td>  
 <td th:text="${auto.year}">Год</td>  
 <td th:text="${auto.vinNumber}">VIN номер</td>  
 <td th:text="${auto.GOS\_number}">Госномер</td>  
 </tr>  
 </tbody>  
 </table>  
</div>  
</body>  
</html>

viewSlots.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Доступные слоты для записи</title>  
 <link href="https://cdn.jsdelivr.net/npm/fullcalendar@5.11.3/main.min.css" rel="stylesheet" />  
 <script src="https://cdn.jsdelivr.net/npm/fullcalendar@5.11.3/main.min.js"></script>  
 <style>  
 body {  
 background-color: #eabc05; /\* Желтый фон страницы \*/  
 color: #222222; /\* Черный текст \*/  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 }  
 h1 {  
 text-align: center;  
 color: #222222; /\* Черный цвет заголовка \*/  
 }  
 .container {  
 max-width: 1000px;  
 margin: 50px auto;  
 padding: 20px;  
 border: 2px solid #222222; /\* Черная рамка \*/  
 border-radius: 10px;  
 background-color: #ffcc00; /\* Желтый фон формы \*/  
 }  
 .btn {  
 padding: 12px 20px;  
 font-size: 16px;  
 cursor: pointer;  
 background-color: #222222; /\* Черная кнопка \*/  
 color: #ffcc00; /\* Желтый текст \*/  
 border: none;  
 border-radius: 5px;  
 transition: background-color 0.3s ease;  
 }  
 .btn:hover {  
 background-color: #444444; /\* Темный черный на hover \*/  
 }  
 select {  
 background-color: #222222; /\* Черный фон для select \*/  
 color: #ffcc00; /\* Желтый текст в select \*/  
 }  
 select option {  
 background-color: #222222; /\* Черный фон для option \*/  
 color: #ffcc00; /\* Желтый текст для option \*/  
 }  
  
 /\* Календарь \*/  
 #calendar {  
 width: 1000px;  
 height: 700px;  
 margin: 0 auto;  
 border: 1px solid #222222;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 padding: 10px;  
 background-color: #eabc05;  
 }  
  
 .selected-date {  
 background-color: #c99f06;  
 color: black;  
 }  
  
 .modal-content {  
 padding: 20px;  
 border: 1px solid #222222;  
 background-color: #eabc05; /\* Желтый фон для модального окна \*/  
 color: #222222;  
 }  
 </style>  
</head>  
<body>  
<div class="container mt-4">  
 <h1>Доступные слоты для записи</h1>  
 <div id="calendar"></div>  
</div>  
<script th:inline="javascript">  
 // Подгружаем доступные слоты из модели  
 let slots = /\*[[${slots}]]\*/ [];  
 document.addEventListener('DOMContentLoaded', function() {  
 let calendarEl = document.getElementById('calendar');  
 let calendar = new FullCalendar.Calendar(calendarEl, {  
 initialView: 'dayGridMonth',  
 locale: 'ru',  
 displayEventTime: false, // Отключаем вывод времени в отдельной строке  
 events: slots.map(slot => {  
 // Форматирование времени  
 let timeParts = slot.time.split(":");  
 let formattedTime = timeParts[0].replace(/^0/, '') + ":" + timeParts[1];  
  
 return {  
 title: 'Доступно: ' + formattedTime, // Устанавливаем нужный формат заголовка  
 start: slot.date + 'T' + slot.time,  
 };  
 })  
 });  
 calendar.render();  
 });  
</script>  
</body>  
</html>

Application.properties:

server.port=\*\*\*\*\* #Указать собственный порт сервера  
spring.application.name=AutoService  
spring.datasource.url=jdbc:mysql://localhost:\*\*\*\*/autoservice? #Указать собственный порт сервера БД  
spring.datasource.username=\*\*\*\* #Указать собственное имя профиля БД  
spring.datasource.password=\*\*\*\*\*\*\*\* #Указать собственный пароль профиля БД  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
spring.datasource.initialization-mode=always  
server.error.whitelabel.enabled=false  
server.error.path=/error