МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Шаблоны программных платформ языка Джава
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Приложение «Стоматологическая клиника»

Студент: Сулемайнов Артём Вадимович
Группа: ИКБО-30-22
Работа представлена к защите_____(дата)_____/Сулейманов А.В./
(подпись и ф.и.о. студента)

Руководитель: доцент Куликов Александр Анатольевич

Работа допущена к защите_____(дата)_____/Куликов А.А./
(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: _____
_____ / _____ /
_____ / _____ /
(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. РТУ МИРЭА. 2024

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: Шаблоны программных платформ языка Джава
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Сулейманов Артём Вадимович
Группа: ИКБО-30-22
Срок представления к защите: 10.05.2024
Руководитель: доцент Куликов Александр Анатольевич

Тема: Приложение «Стоматологическая клиника»

Исходные данные: индивидуальное задание на разработку; документация по Spring Framework и JEE, документация по языку Java (версия не ниже 8); инструменты и технологии: JDK (не ниже 8), создание Spring MVC web-приложений, RESTful web-сервисов, Spring ORM и Spring DAO, Gradle, gitHub, IntelliJIDEA. Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

Перечень вопросов, подлежащих разработке, и обязательного графического материала: 1. Провести анализ предметной области и формирование основных требований к приложению, 2. Обосновать выбор средств ведения разработки. 3. Разработать приложение с использование фреймворка Spring, выбранной технологии и инструментария. 4.Провести тестирование приложения. 5. Оформить пояснительную записку по курсовой работе 6. Провести анализ текста на антиплагиат 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: _____/Р. Г. Болбаков/, «9» февраля 2024 г.

Задание на КР выдал: _____/ А.А.Куликов/, «9» февраля 2024 г.

Задание на КР получил: _____/А.В. Сулейманов/, «9» февраля 2024 г.

РЕФЕРАТ

Отчет 22 с., 17 рис., 5 источн.

ИНТЕРНЕТ, СЕРВЕРНАЯ РАЗРАБОТКА, SPRING, SPRING BOOT, POSTGRESQL, GRADLE, СТОМАТОЛОГИЯ

Объект исследования – серверное приложение "Стоматологическая клиника".

Предмет исследования – создание серверного приложения "Стоматологическая клиника".

В рамках исследования был проведен тщательный анализ области деятельности и рассмотрены существующие веб-приложения, предоставляющие аналогичные услуги.

Проанализированы этапы разработки веб-платформы, применяемые программные решения и среда для разработки.

Результатом проекта стало серверное приложение "Стоматологическая клиника", содержащее основной сценарий использования сайта стоматологической клиники.

REPORT

Report 22 p., 17 fig., 5 sources.

INTERNET, SERVER-SIDE DEVELOPMENT, SPRING, SPRING BOOT, POSTGRESQL, GRADLE, DENTISTRY

The object of the study is the server application "Dental Clinic".

The subject of the study is the development of the server part of the Flower Shop web application.

During the research, a thorough analysis of the domain was conducted, and existing web applications providing similar services were reviewed.

The stages of web platform development, the software solutions used, and the development environment were analyzed.

The result of the project is the server application "Dental Clinic," which includes the main usage scenario of a dental clinic website.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете применяют следующие термины с соответствующими определениями:

Геттеры – методы, которые позволяют получать значения приватных переменных класса, тем самым обеспечивая капсуляцию данны

Сеттеры – методы, которые предоставляют интерфейс для изменения значений приватных переменных, повышая контроль над тем, как и когда данные могут быть изменены

Эндпоинт – конечная точка URL, к которой можно обращаться для выполнения определённых действий с ресурсами приложения

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете применяются следующие сокращения и обозначения:

CRUD – Create, Read, Update, Delete

JPA – Java Persistence API

UML – Unified Modeling Language

ВВЕДЕНИЕ

Целью курсовой работы является разработка серверного приложения «Стоматологическая клиника», а также анализ предметной области с целью определения основных требований к приложению. Основная задача заключается в разработке собственной архитектуры и функционала приложения, используя язык программирования Java [1], фреймворк Spring [2] и базу данных PostgreSQL [3]. Для тестирования функционала приложения будет использоваться инструмент Bruno [4]. Для достижения поставленной цели были сформулированы следующие задачи:

– провести анализ предметной области, связанной с разрабатываемым приложением,

– выбрать инструменты для ведения разработки,

– разработать приложение с использованием выбранных средств разработки,

– протестировать разработанное приложение.

Для решения поставленных задач используются различные аналитические методы и сравнительный анализ. Информационная база для данного исследования состоит из знаний, полученных в ходе прохождения курса "Шаблоны программных платформ на языке Java", а также из материалов, доступных в интернете.

В разделе, представляющем логическую структуру, описаны аналитические методы, использованные в ходе работы, а также структура базы данных приложения.

Раздел, посвященный обоснованию выбора технологий, включает информацию о программных инструментах, примененных в проекте.

В разделе разработки подробно излагается информация о созданном приложении, его тестировании и архитектуре.

Заключительный раздел подводит итоги выполненной работы и содержит ссылку на исходный приложения.

1 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

1.1 Анализ предметной области

Анализ предметной области проводился среди веб-сайтов стоматологических клиник. Для анализа было выбрано веб-приложение ОН КЛИНИК [5], представленное на рисунке 1.

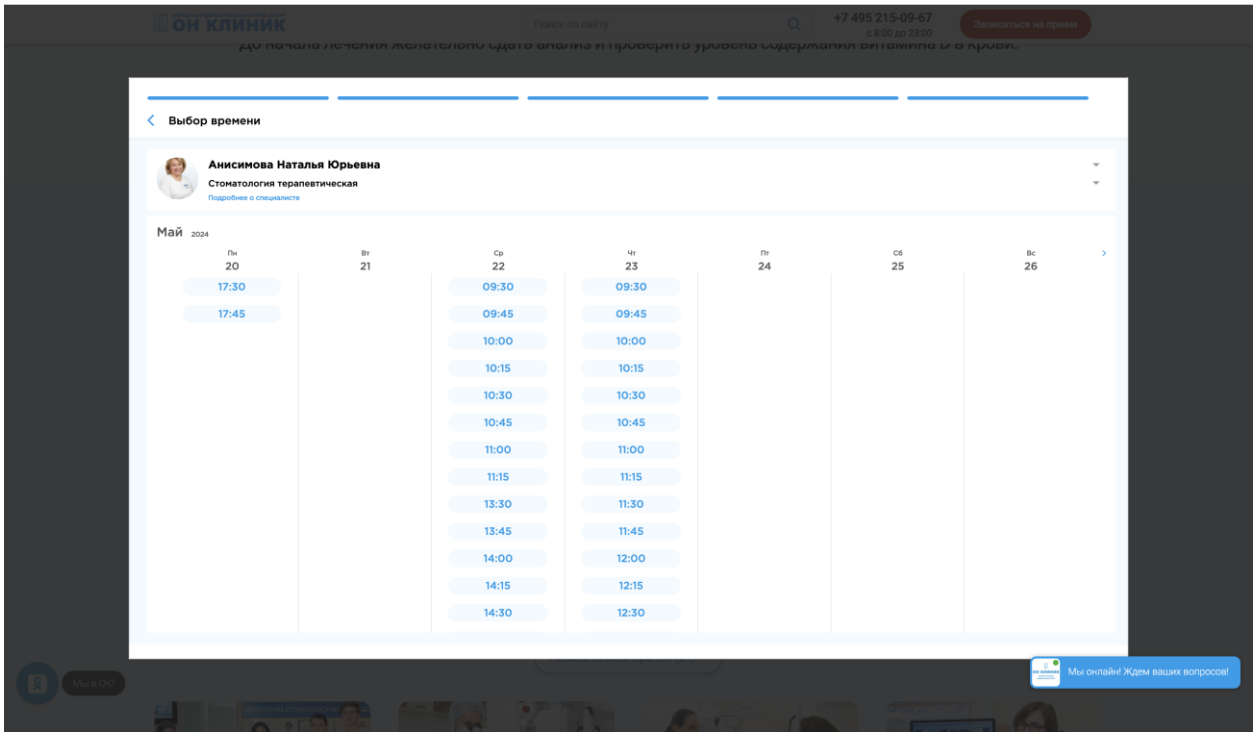

Рисунок 1 – Клиентская часть сайта ОН КЛИНИК

Данное веб-приложение содержит функционал выбора медицинского сотрудника, времени, предоставляемой услуги, а также записи личной информации клиента.

Из анализа веб-сайта следует выделить ключевые возможности, которыми должен обладать разрабатываемое серверное приложение стоматологической клиники:

– создание и редактирование данных о медицинских сотрудниках, услугах и клиентах,

– добавление и редактирование заказа, связанного с сотрудником, клиентом и предоставляемой услугой.

1.2 Структура базы данных

Для хранения пользователей, сотрудников, услуг и заказов веб-приложения, используется база данных со структурой, описанной с помощью UML-диаграмм на рисунке 2.

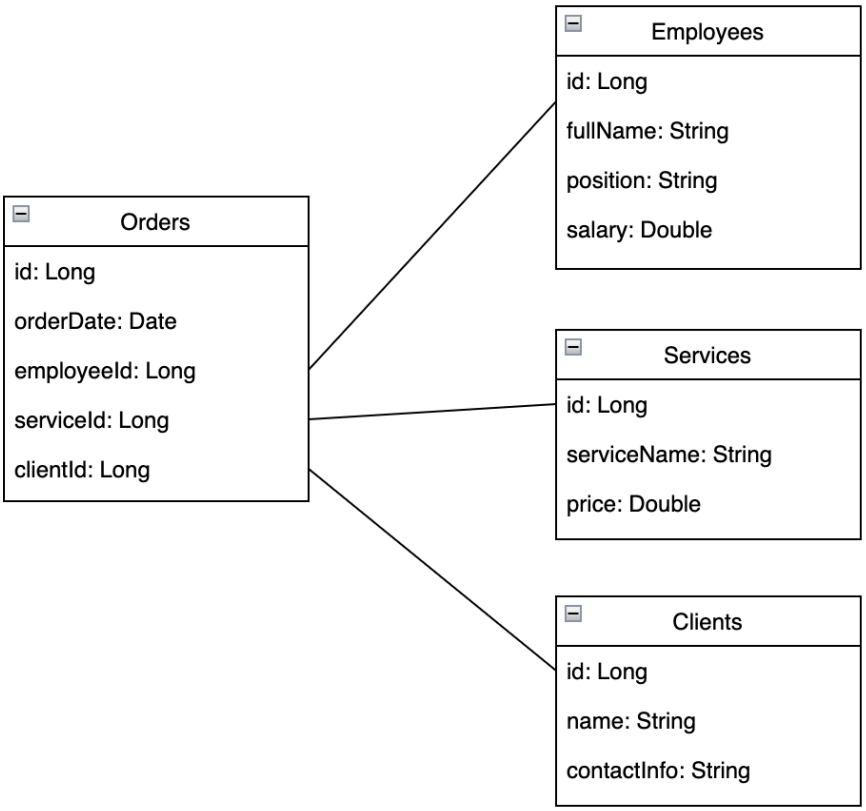

Рисунок 2 – Структура базы данных приложения

2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

2.1 Используемое прикладное программное обеспечение

Для разработки приложения использовалась среда разработки IntelliJ IDEA Community Edition.

2.2 Используемые технологии

В процессе разработки приложения "Стоматологическая клиника" были использованы следующие ключевые технологии:

Java: Язык программирования Java был выбран в качестве основного инструмента из-за его переносимости, безопасности и мощных библиотек для разработки Enterprise-приложений. Java поддерживает многопоточность, что критически важно для разработки высокопроизводительных серверных приложений, работающих с большим количеством пользовательских запросов.

Gradle: Для автоматизации процесса сборки проекта выбран инструмент Gradle. Он позволяет значительно ускорить процесс сборки и обеспечивает высокую степень конфигурируемости, что упрощает интеграцию с различными платформами и инструментами, используемыми в проекте.

Spring Framework: Использование Spring Framework дает возможность строить модульные, легко масштабируемые веб-приложения. Spring Data JPA упрощает работу с базами данных и уменьшает количество рутинного кода благодаря автоматизации многих задач, связанных с установлением соединения, транзакциями и выполнением запросов.

PostgreSQL: Выбор PostgreSQL в качестве системы управления базами данных был обусловлен её надежностью, поддержкой транзакций, масштабируемостью и широкими возможностями в области обеспечения безопасности данных. Расширенный функционал PostgreSQL позволяет эффективно обрабатывать большие объемы информации, что особенно важно для приложений, работающих с большим числом пользователей.

Bruno: Инструмент для тестирования API Bruno был выбран вместо Postman благодаря его удобству и широким возможностям автоматизации

тестирования. Возможности Bruno позволяют удобно управлять тест-кейсами, настраивать окружения для тестирования и автоматически выполнять тесты, что важно в условиях динамической работы над проектом.

Применение перечисленных технологий позволило создать надежное и эффективное приложение, способное удовлетворять высокие требования пользователей и обладающее потенциалом для дальнейшего расширения и модификации в ответ на изменяющиеся условия использования.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Структура приложения

Рабочая директория приложения содержит в себе несколько уровней. Код находиться в директории java/com/example/DentistClinic, а в директории resourses содержится файл application.properties. Директория изображена на рисунке 3.

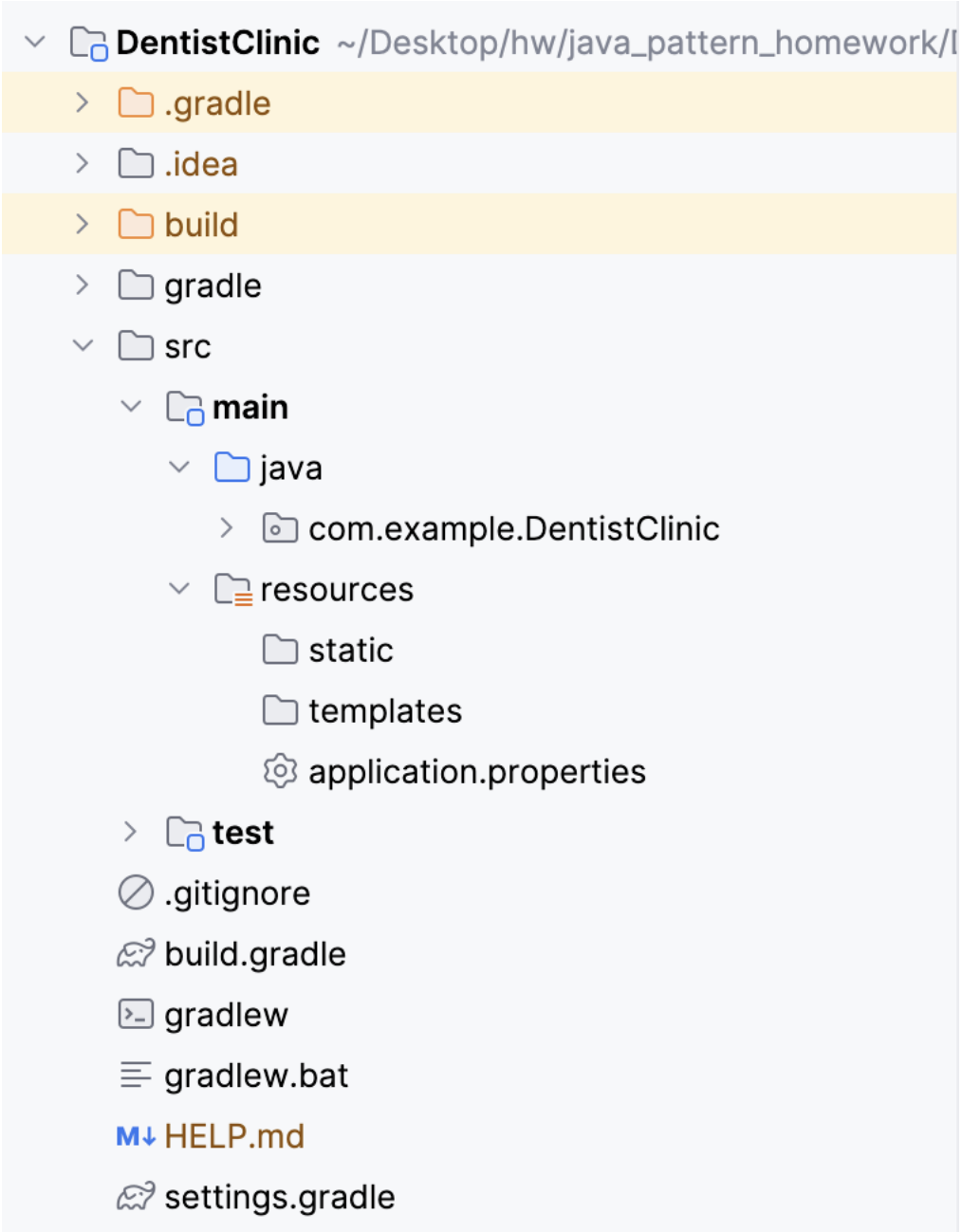

Рисунок 3 – Рабочая директория приложения

3.2 Подключение к базе данных

Для установления соединения с базой данных приложения необходимо провести конфигурацию файла application.properties, детали которой представлены на рисунке 4. В этом файле указываются строка подключения, логин и пароль пользователя, необходимые для доступа к выбранной базе данных. База данных размещена локально.

```
spring.application.name=DentistClinic
spring.datasource.url=jdbc:postgresql://localhost:5432/dentistclinic
spring.datasource.username=user123
spring.datasource.password=user123
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

Рисунок 4 – Файл application.properties

3.3 Разработка серверной части приложения

Для работы с таблицей сотрудников была написана модель, описывающая поля таблицы и содержащая геттеры и сеттеры, репозиторий, который выступает как медиатор между доменным слоем приложения (где размещаются бизнес-модели) и слоем доступа к данным, сервис, обеспечивающий логику управления данными и бизнес-процессами и контроллер, обрабатывающий CRUD запросы по пути employees. Код представлен на рисунках 5-8.

```java
@Entity  20 usages  stufade
@Table(name = "employees")
public class Employee {

    @Id  2 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)  2 usages
    private String fullName;

    @Column(nullable = false)  2 usages
    private String position;

    @Column(nullable = false)  2 usages
    private Double salary;

    // Геттеры и сеттеры
    public Long getId() {  stufade
        return id;
    }

    public void setId(Long id) {  stufade
        this.id = id;
    }

    public String getFullName() {  1 usage  stufade
        return fullName;
    }
```

Рисунок 5 – Модель сотрудника

```java
package com.example.DentistClinic.repository;

import com.example.DentistClinic.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {  2 usages  stufade
}
```

Рисунок 6 – Репозиторий сотрудника

```java
@Service  4 usages  stufade
public class EmployeeService {

    @Autowired  4 usages
    private

    EmployeeRepository employeeRepository;

    public List<Employee> findAllEmployees() { return employeeRepository.findAll(); }

    public Employee findEmployeeById(Long id) { return employeeRepository.findById(id).orElse( other: null); }

    public Employee saveEmployee(Employee employee) { return employeeRepository.save(employee); }

    public void deleteEmployee(Long id) { employeeRepository.deleteById(id); }
}
```

Рисунок 7 – Сервис сотрудника

```java
@RestController  no usages  stufade
@RequestMapping("/employees")
public class EmployeeController {

    @Autowired  6 usages
    private EmployeeService employeeService;

    @GetMapping  no usages  stufade
    public List<Employee> getAllEmployees() { return employeeService.findAllEmployees(); }

    @GetMapping("/{id}")  no usages  stufade
    public Employee getEmployeeById(@PathVariable Long id) { return employeeService.findEmployeeById(id); }

    @PostMapping  no usages  stufade
    public Employee createEmployee(@RequestBody Employee employee) { return employeeService.saveEmployee(employee); }

    @PutMapping("/{id}")  no usages  stufade
    public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails) {
        Employee employee = employeeService.findEmployeeById(id);
        employee.setFullName(employeeDetails.getFullName());
        employee.setPosition(employeeDetails.getPosition());
        employee.setSalary(employeeDetails.getSalary());
        return employeeService.saveEmployee(employee);
    }

    @DeleteMapping("/{id}")  no usages  stufade
    public void deleteEmployee(@PathVariable Long id) { employeeService.deleteEmployee(id); }
}
```

Рисунок 8 – Контроллер сотрудника

Код для клиента и услуги аналогичен.

Для работы с таблицей заказов была добавлена связь с таблицей сотрудников, услуг и сотрудников, а также логика проверки существования данных для создания заказа. Код представлен на рисунках 9-12.

```java
@Entity  16 usages  stufade
@Table(name = "orders")
public class Order {

    @Id  2 usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne  2 usages
    @JoinColumn(name = "employee_id", nullable = false)
    private Employee employee;

    @ManyToOne  2 usages
    @JoinColumn(name = "client_id", nullable = false)
    private Client client;

    @ManyToOne  2 usages
    @JoinColumn(name = "service_id", nullable = false)
    private BusinessService service;

    @Column(nullable = false)  2 usages
    private Date orderDate;

    // Геттеры и сеттеры
    public Long getId() {  stufade
        return id;
    }

    public void setId(Long id) {  stufade
        this.id = id;
    }
```

Рисунок 9 – Модель заказа

```java
package com.example.DentistClinic.repository;

import com.example.DentistClinic.model.Order;
import org.springframework.data.jpa.repository.JpaRepository;

public interface OrderRepository extends JpaRepository<Order, Long> {  2 usages  stufade
}
```

Рисунок 10 – Репозиторий заказа

```java
@Service  2 usages  stufade
public class OrderService {

    @Autowired  4 usages
    private OrderRepository orderRepository;

    public List<Order> findAllOrders() {  1 usage  stufade
        return orderRepository.findAll();
    }

    public Order findOrderById(Long id) {  2 usages  stufade
        return orderRepository.findById(id).orElse( other: null);
    }

    public Order saveOrder(Order order) {  2 usages  stufade
        return orderRepository.save(order);
    }

    public void deleteOrder(Long id) {  1 usage  stufade
        orderRepository.deleteById(id);
    }
}
```

Рисунок 11 – Сервис заказа

```java
@PostMapping  no usages  stufade
public Order createOrder(@RequestBody CreateOrderRequest request) {
    System.out.println(request.getClientId());
    Employee employee = employeeService.findEmployeeById(request.getEmployeeId());
    Client client = clientService.findClientById(request.getClientId());
    BusinessService service = businessServiceService.findServiceById(request.getServiceId());

    if (employee == null || client == null || service == null) {
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid employee, client, or service ID");
    }

    Order order = new Order();
    order.setEmployee(employee);
    order.setClient(client);
    order.setService(service);
    order.setOrderDate(request.getOrderDate());

    return orderService.saveOrder(order);
}
```

Рисунок 12 – Обработка создания заказа

3.4 Тестирование приложения

Используя программу Bruno и браузер, были протестированы несколько эндпоинтов приложения. Результат тестирования добавления и получения пользователей представлен на рисунках 13-14.

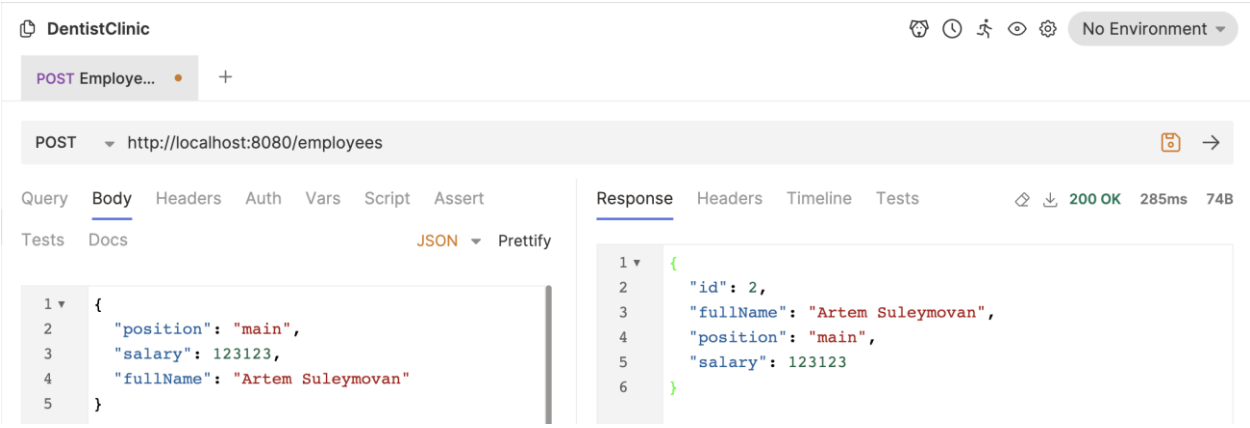

Рисунок 13 – Тестирование создания сотрудника

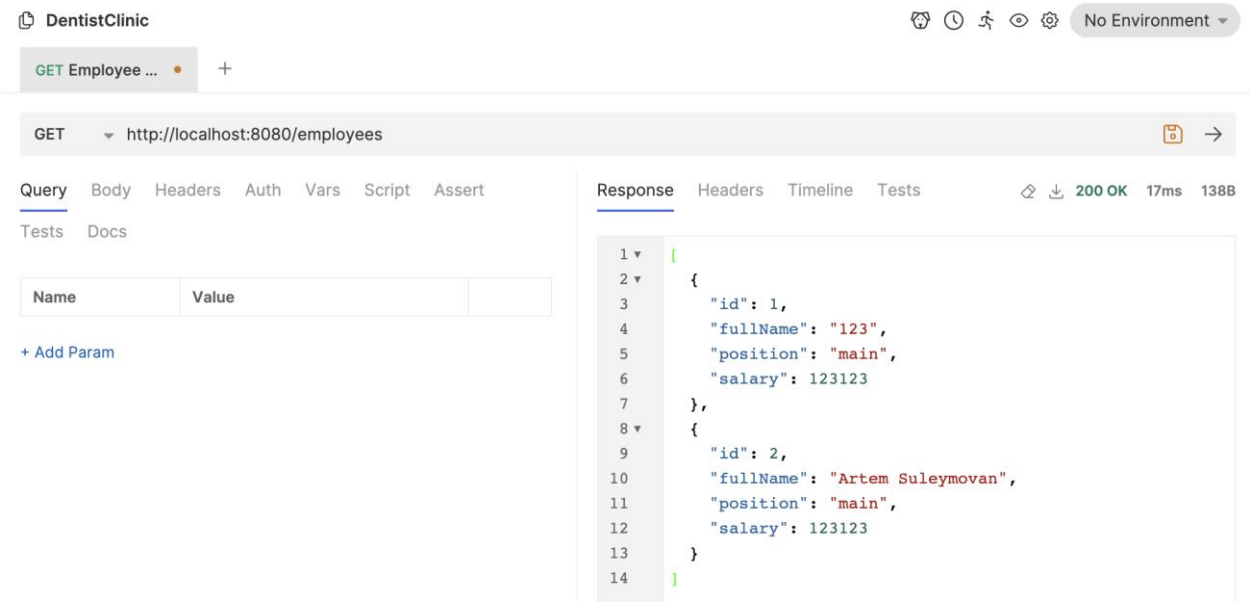

Рисунок 14 – Тестирвоание получения всех сотрудников

Далее протестируем создание и получение заказов. Результат тестирования представлен на рисунках 15-17.

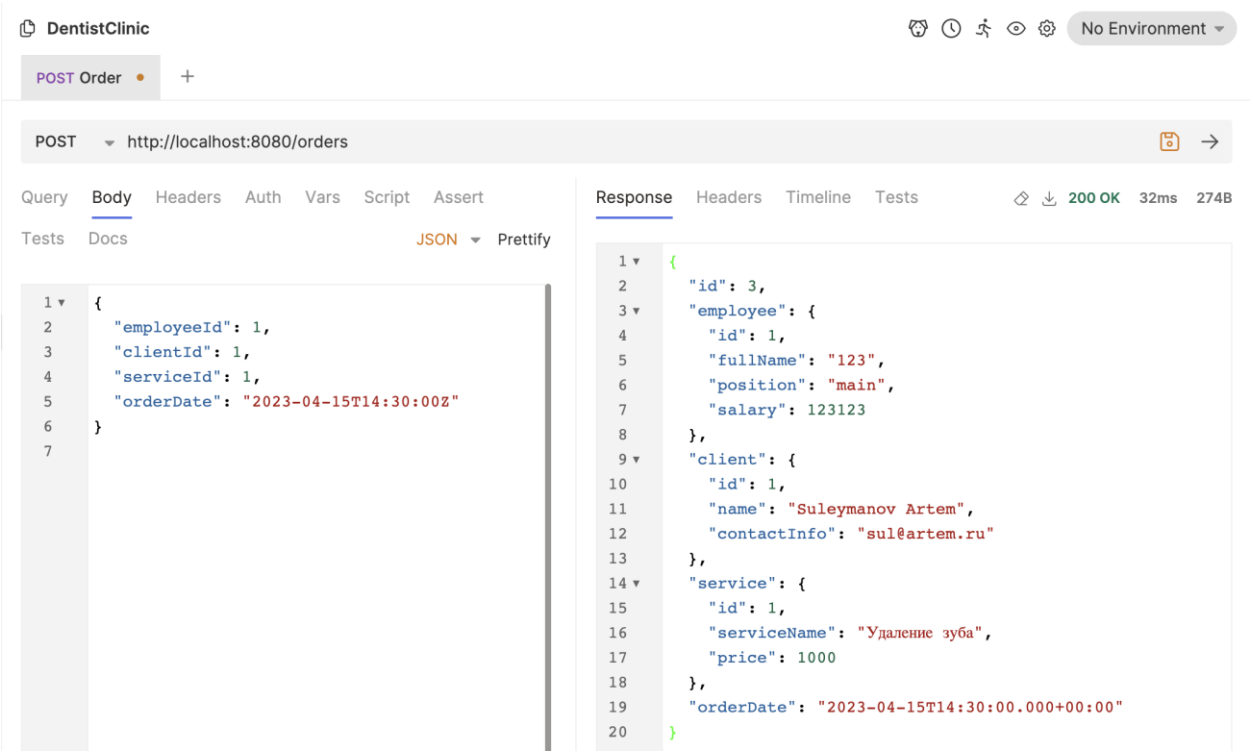

Рисунок 15 – Тестирование создания заказа

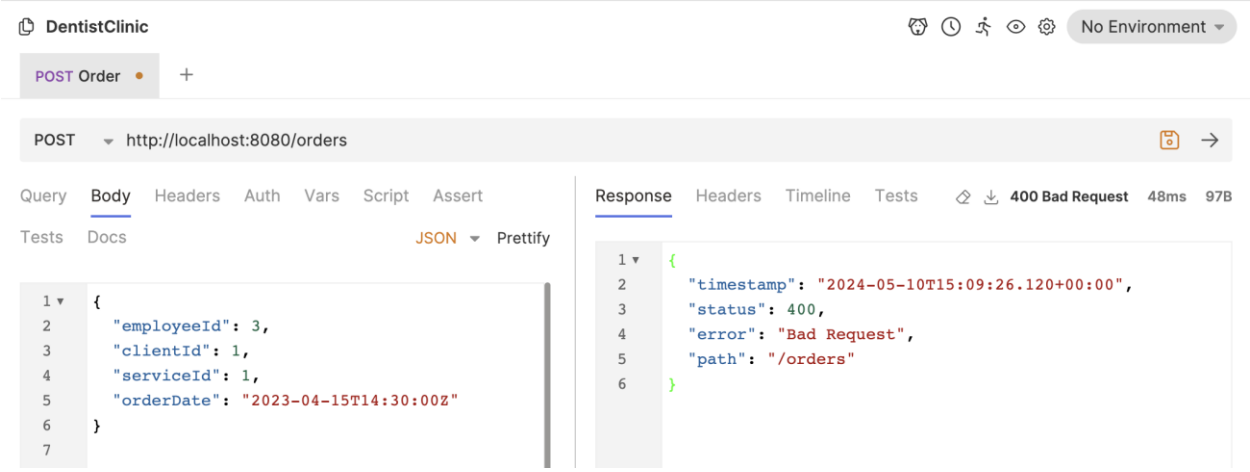

Рисунок 16 – Тестирование создания заказа с несуществующим сотрудником

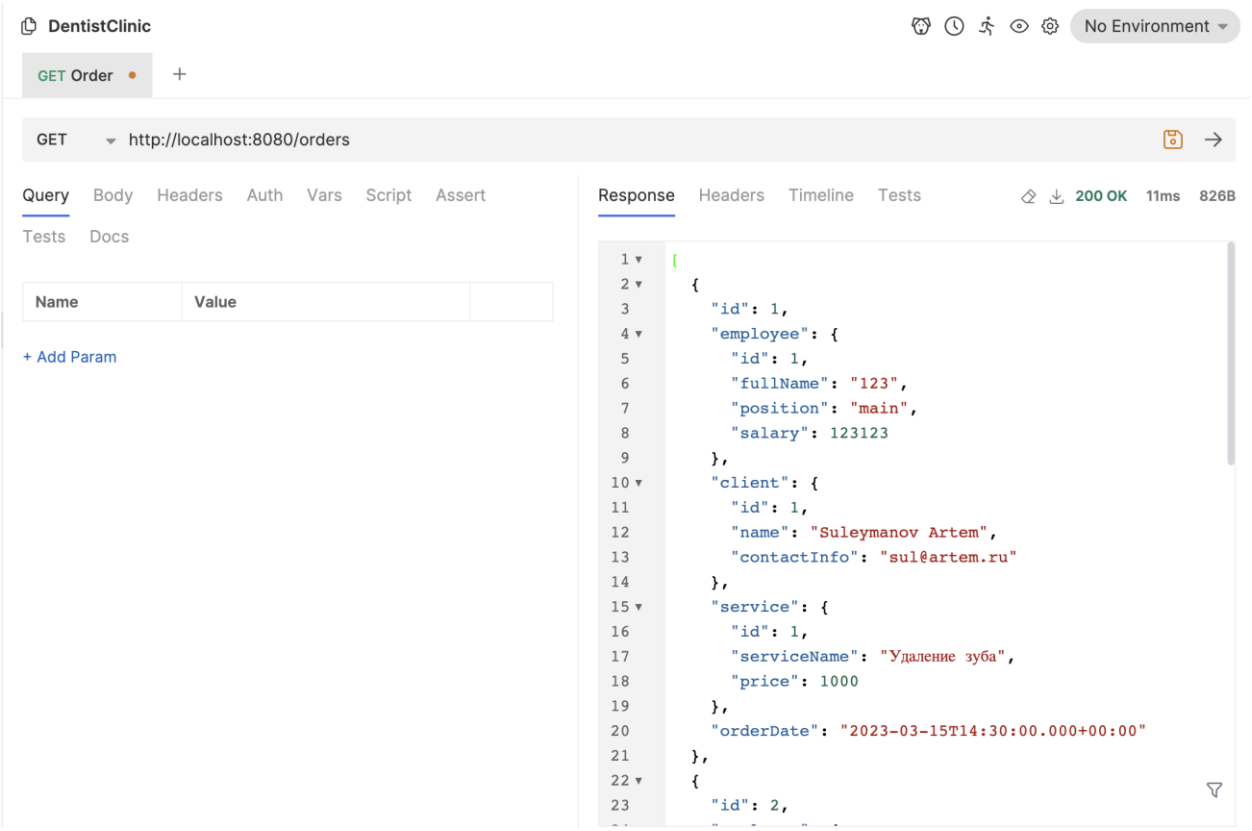

Рисунок 17 – Тестирование получения заказов

ЗАКЛЮЧЕНИЕ

Завершение разработки приложения для "Стоматологической клиники" ознаменовалось успешным исполнением всех запланированных задач и достижением желаемых результатов. Аналитическая работа по изучению специфики сферы позволила адекватно выбрать инструменты разработки, что, в свою очередь, привело к успешной реализации и последующему тщательному тестированию функций приложения. Разработанная система характеризуется надежностью, удобством в обращении и наличием всех необходимых функций для эффективного управления процессами в клинике.

Тестирование продемонстрировало отличную производительность программы по ключевым параметрам, таким как надежность работы и эффективность использования ресурсов. Благодаря высоким техническим характеристикам, программное решение имеет все шансы занять лидирующие позиции на рынке IT-решений для медицинских учреждений. В рамках курсового проекта был подготовлен полноценный отчет и презентация, а также проведена проверка данных материалов на уникальность. Проект был реализован в полном объеме, достигнув всех поставленных перед началом его выполнения задач.

Исходный код расположен на GitHub: https://github.com/stufade/java_pattern_homework/tree/main/DentistClinic.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. OpenJDK 21 Documentation [Электронный ресурс]. – URL: https://devdocs.io/openjdk~21/ (дата обращения 06.05.2024).

2. Spring Framework [Электронный ресурс]. – URL: https://spring-projects.ru/projects/spring-framework/ (дата обращения 06.05.2024).

3. PostgreSQL Documentation [Электронный ресурс]. – URL: https://www.postgresql.org/docs/ (дата обращения 06.05.2024).

4. Bruno [Электронный ресурс]. – URL: https://www.usebruno.com/ (дата обращения 06.05.2024).

5. ОН КЛИНИК [Электронный ресурс]. – URL: https://www.onclinic.ru/stomatologiya/ (дата обращения 06.05.2024).