Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное   
образовательное учреждение высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №1

Дисциплина: «Технологии сетевого программирования»

Тема «Разработка базы данных»

Выполнили: Новикова В.В., Мартынов О.И.

Группа: 6302-010302D

Самара, 2025

**1. Настройка базы данных.**

1. **Установка Docker:**

* Скачали и установили Docker на компьютер.
* Авторизовались в Docker для дальнейшей работы.

1. **Создание Dockerfile:**

* Создали файл Dockerfile для настройки контейнера с PostgreSQL.

**Содержимое файла Dockerfile:**

# Используем официальный образ PostgreSQL

FROM postgres:latest

# Устанавливаем переменные окружения для базы данных

ENV POSTGRES\_DB=mydatabase

ENV POSTGRES\_USER=myuser

ENV POSTGRES\_PASSWORD=mypassword

# Открываем порт 5432 для подключения к базе данных

EXPOSE 5432

1. **Создание docker-compose.yml:**

Создали файл docker-compose.yml для упрощения управления контейнером.

Содержимое файла:

services:

postgres:

build: .

container\_name: beauty\_salon\_db

environment:

POSTGRES\_DB: mydatabase

POSTGRES\_USER: myuser

POSTGRES\_PASSWORD: mypassword

ports:

- "5432:5432"

volumes:

- ./data:/var/lib/postgresql/data

1. **Запуск контейнера:**

* Запустили контейнер с помощью команды:

docker-compose up –d

1. **Подключение к базе данных:**

* Установил DBeaver для работы с базой данных.
* Подключился к базе данных, используя следующие параметры:

Host: localhost

Port: 5432

Database: mydatabase

User: myuser

Password: mypassword

**2. Описание базы данных.**

База данных состоит из пяти сущностей, которые представляют собой основные объекты предметной области — запись на услугу в салоне красоты. Ниже приведено описание каждой сущности и связей между ними.

**Сущности:**

**1. Appointment (Запись на услугу):**

* Описывает запись клиента на конкретную услугу к определённому мастеру.
* Поля:
* id (Long) — уникальный идентификатор записи.
* client (Client) — клиент, который записался на услугу (связь Many-to-One с сущностью Client).
* master (Master) — мастер, который будет выполнять услугу (связь Many-to-One с сущностью Master).
* beautyService (BeautyService) — услуга, на которую записан клиент (связь Many-to-One с сущностью BeautyService).
* date (LocalDate) — дата записи.
* time (LocalTime) — время записи.
* status (AppointmentStatus) — статус записи (например, "подтверждено", "отменено" и т.д.).

**2. BeautyService (Услуга):**

* Описывает услугу, предоставляемую салоном красоты.
* Поля:
* id (Long) — уникальный идентификатор услуги.
* name (String) — название услуги (уникальное).
* price (Double) — стоимость услуги.
* masters (List<Master>) — какие мастера оказывают услугу

**3. Client (Клиент):**

* Описывает клиента салона красоты.
* Поля:
* id (Long) — уникальный идентификатор клиента.
* name (String) — имя клиента.
* phone (String) — телефон клиента (уникальный).
* login (String) — логин для входа в систему (уникальный).
* password (String) — пароль для входа в систему.

**4. Master (Мастер):**

* Описывает мастера салона красоты.
* Поля:
* id (Long) — уникальный идентификатор мастера.
* name (String) — имя мастера.
* phone (String) — телефон мастера (уникальный).
* login (String) — логин для входа в систему (уникальный).
* password (String) — пароль для входа в систему.
* beautyServices (List<BeautyService>) — услуги, которые оказывает мастер

**Связи между сущностями:**

**1. Appointment → Client:**

* Один клиент может иметь несколько записей на услуги (One-to-Many).
* В сущности Appointment это реализовано через поле client с аннотацией @ManyToOne.

**2. Appointment → Master:**

* Один мастер может выполнять несколько записей на услуги (One-to-Many).
* В сущности Appointment это реализовано через поле master с аннотацией @ManyToOne.

**3. Appointment → BeautyService:**

* Одна услуга может быть связана с несколькими записями (One-to-Many).
* В сущности Appointment это реализовано через поле beautyService с аннотацией @ManyToOne.

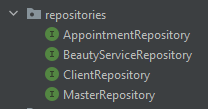
**4. Master BeautyService:**

* Один мастер может выполнять несколько услуг.
* Каждую услугу могут оказывать разные мастера.
* Связь @ManyToMany

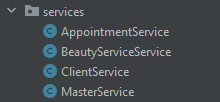
**3. Скрипты для демонстрации работы с данными.**

Для работы с данными в приложении используется слоистая архитектура, которая включает:

1. **Repositories** — для взаимодействия с базой данных. Они отвечают за выполнение операций с базой данных. Наследуют класс JPARepository



1. **Services** — для реализации логики. Используют репозитории для работы с данными.



**Настройка конфигурационного файла Application.properties:**

spring.application.name=beauty-salon-booking

# Подключение к PosgreSQL

spring.datasource.url=jdbc:postgresql://localhost:5432/mydatabase

spring.datasource.username=myuser

spring.datasource.password=mypassword

spring.datasource.driver-class-name=org.postgresql.Driver

# Настройка Hibernate

spring.jpa.database=postgresql

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

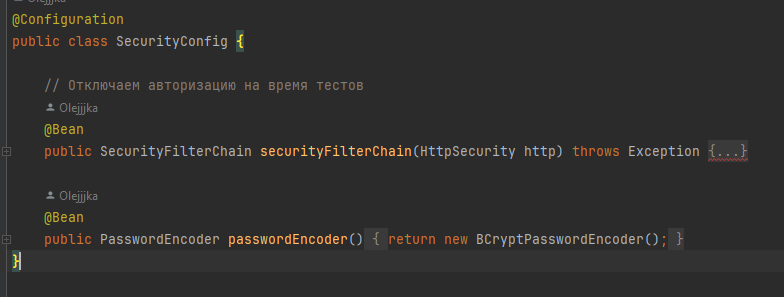
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

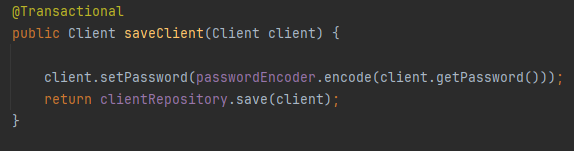
Выполнение файла data.sql

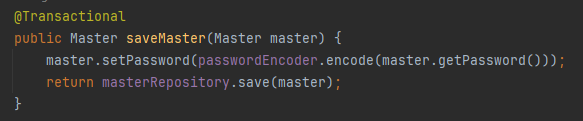
spring.sql.init.mode=always

После запуска BeautySalonBookingApplication.java наше приложение успешно подключилось к БД, были созданы необходимые таблицы на основе сущностей, их связи и добавлены тестовые данные.

**4. Реализация хеширования паролей.**

****

****

****

**5. Тесты.**

