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

Тема: «Разработка базы данных»

Дисциплина: «Технологии сетевого программирования»

Выполнили: Зайнуллова А.М.

Тарасов П.Д.

Группа: 6304-010302D

Самара, 2025

**Этапы разработки:**

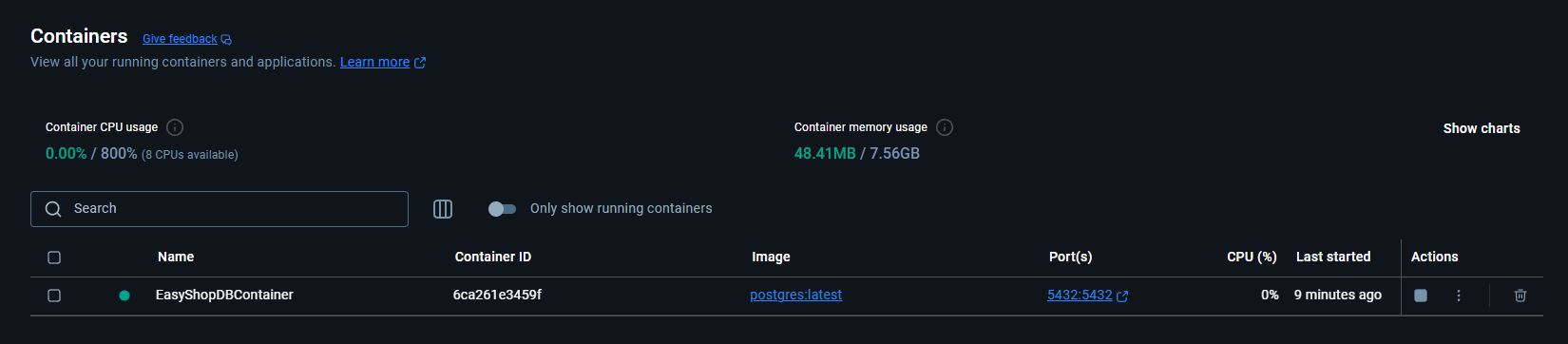
1. **Настройка базы данных**

* Развертывание базы данных **PostgreSQL** в контейнере **Docker**

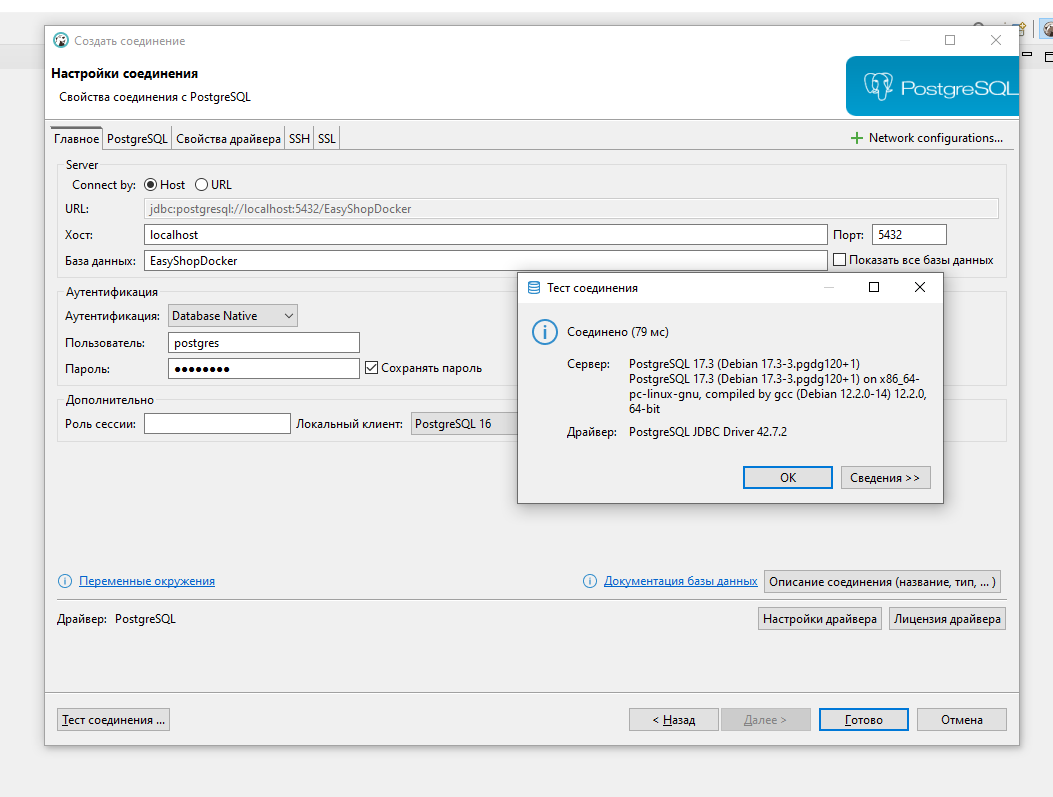
Развертывание производилось с помощью выполнения команды в командной строке операционной системы:

**docker run --name EasyShopDBContainer -e POSTGRES\_DB=EasyShopDocker -e POSTGRES\_USER=postgres -e POSTGRES\_PASSWORD=pa33word -p 5432:5432 -d postgres:latest**

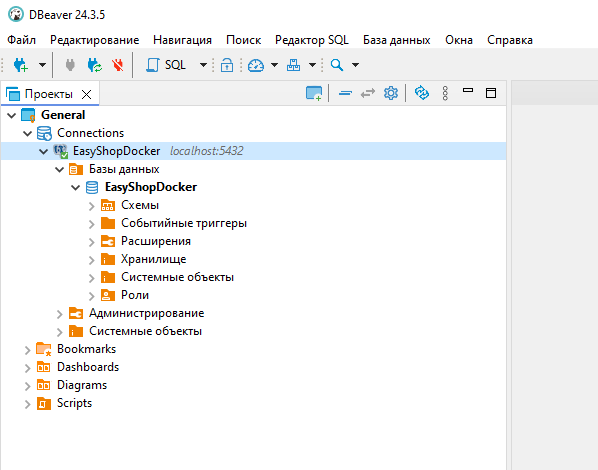
Запущенный контейнер можно наблюдать в приложении **Docker Desktop**

****

* Тестируем подключение к БД с помощью приложения **DBeaver**

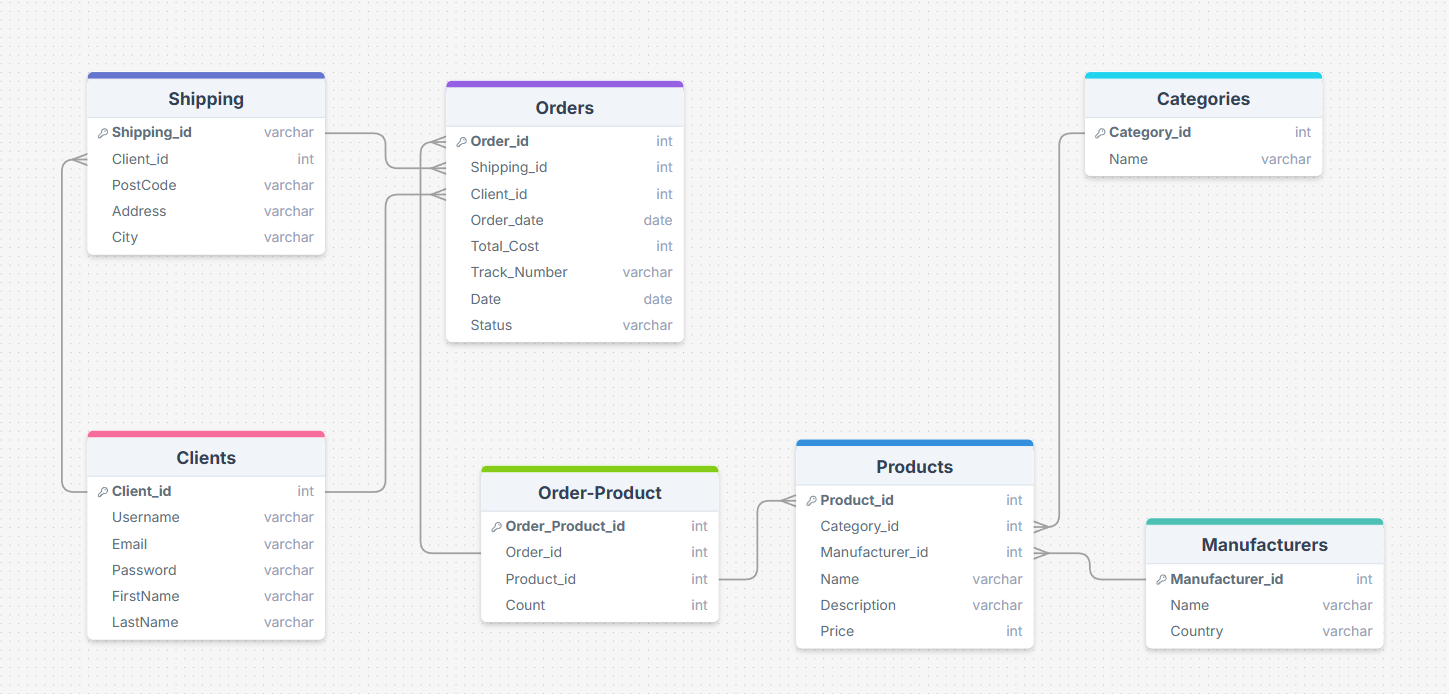
****

* После успешного тестирования можем сохранить подключение в **DBeaver.**

****

1. **Разработка ORM-модели**

* Структура сущностей и связей с ними имеет вид

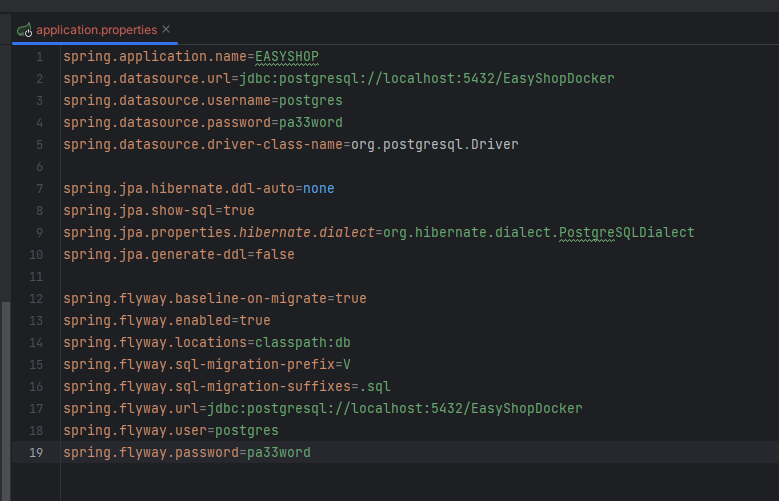


* В работе с ORM-моделью мы будем использовать **Hibernate**, **Spring Data JPA**, для этого используя **Maven**, добавляем зависимости необходимых фреймворков.

****

Добавление отдельной зависимости **Hibernate** не требуется, так как она содержится в зависимости **spring-boot-starter-data-jpa**.

* Настройку **Hibernate** и **Spring Data** произведём с помощью файла конфигурации **Application.properties**.



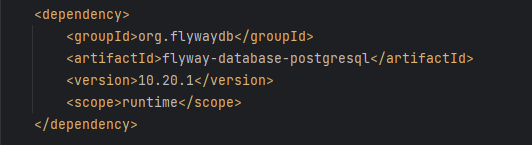
* Разработку сущностей выполняем с помощью **Java** классов и **Hibernate** аннотаций.

Пример класса сущности Категорий

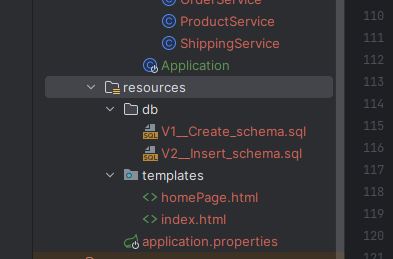


Помимо объявленных полей, каждый класс имеет методы GET и SET для каждого из объявленных полей и пустой конструктор.

* Для миграции базы данных будем использовать **Flyway**, добавляем зависимость в pom файл и определяем конфигурацию **Flyway** в представленном выше файле **Application.properties**.

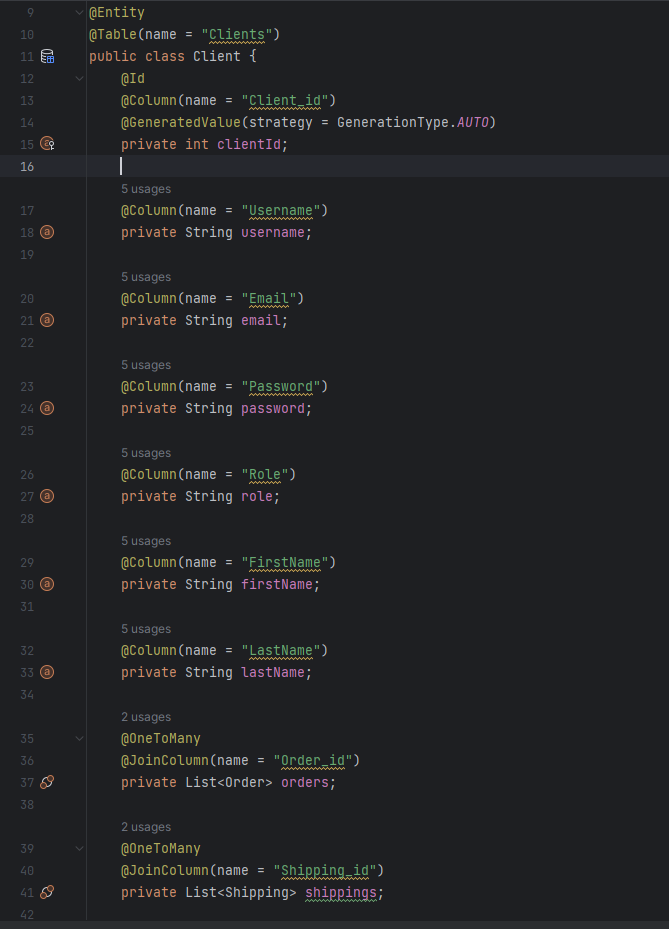


* После настройки **Flyway** добавляем файлы бэкапа базы данных в указанную при конфигурации папку.



1. **Разработка модели пользователя**

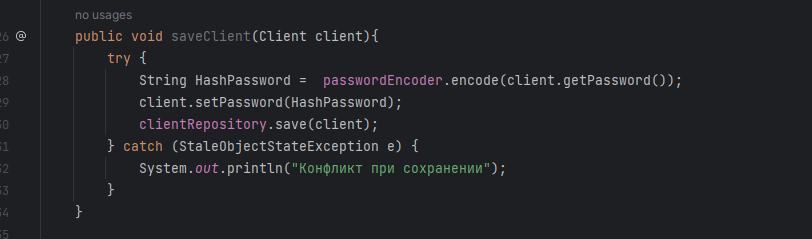
* Сущность пользователя, которая будет использоваться для аутентификации и авторизации пользователей сервиса, имеет вид



* Аутентификация, авторизация и хеширование выполнено с помощью фреймворков **Spring Security** и **Bcrypt**. Конфигурация **Spring Security** выполнена с помощью класса-контроллера вида:

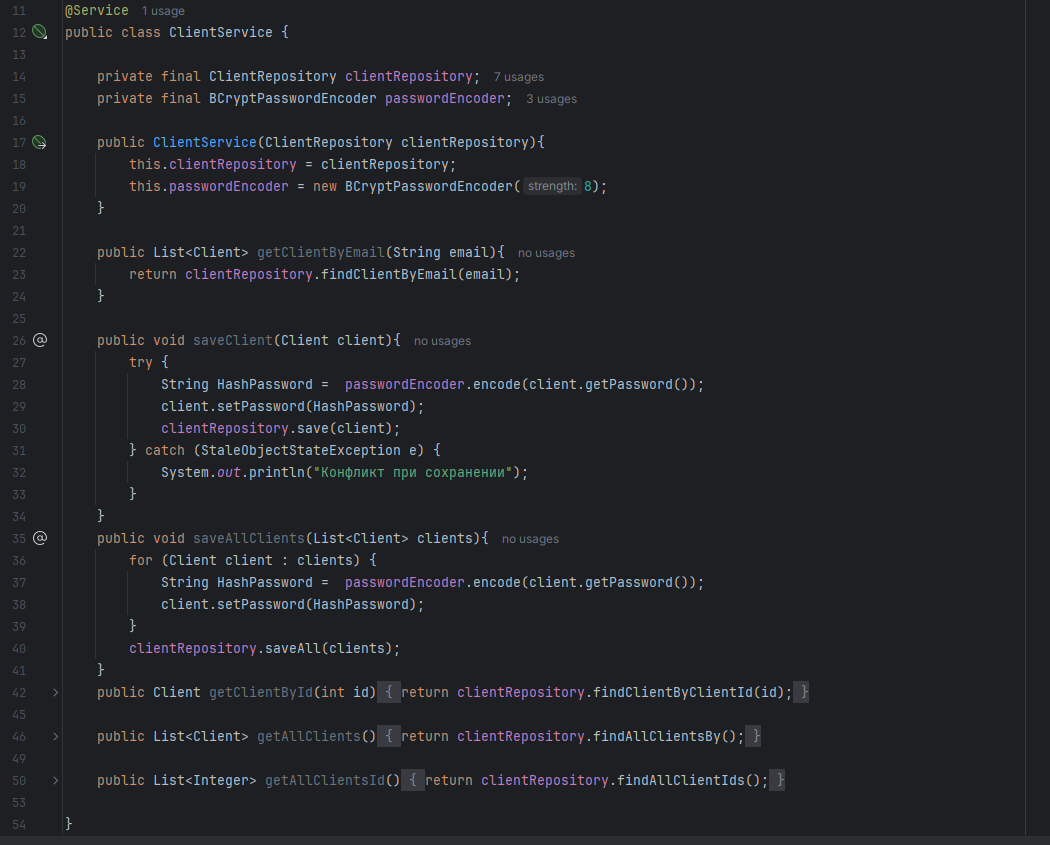


Хеширование паролей происходит в Service слое приложения при сохранении нового пользователя в базе данных.



1. **Функционал для работы с моделями**

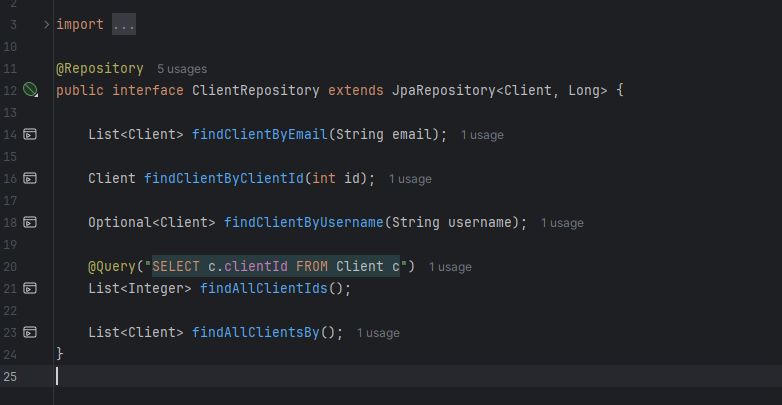
Для реализации функционала работы с моделями были выполнены Service и Repository слои приложения. Для каждой из сущностей был создан класс entityService, реализующий все необходимые методы работы с данной сущностью. Пример полной реализации методов в классе ClientService:



Класс ClientService индивидуально имеет поле типа BCryptPasswordEncoder passwordEncoder, данный объект класса используется для хеширования паролей пользователей при сохранении в базе данный, так же он участвует в авторизации пользователей при входе на вебсайт приложения.

Каждый из Service классов имеет свой экземпляр соответствующего интерфейса слоя Repository, который класс использует для взаимодействия с базой данных.

Пример реализации ClientRepository:



* Заполнение тестовыми данными выполнено через отдельный класс для генерации массивов подходящих записей.

Определённые методы класса генерации данных:

