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: Мельников С.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-compose up -d –build

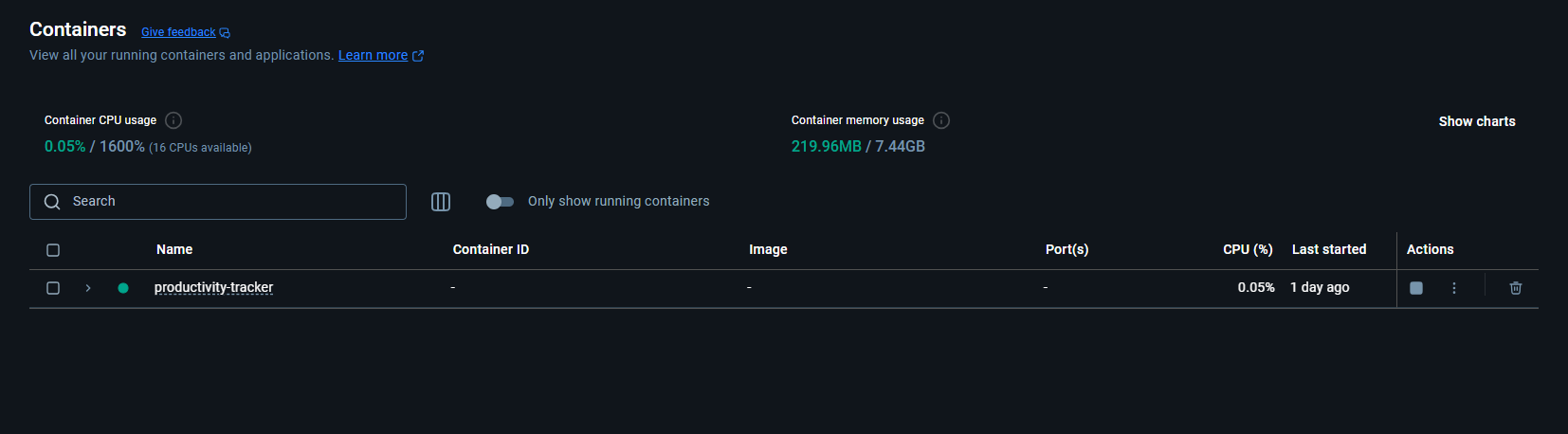
docker ps

Проверка, что контейнер запущен

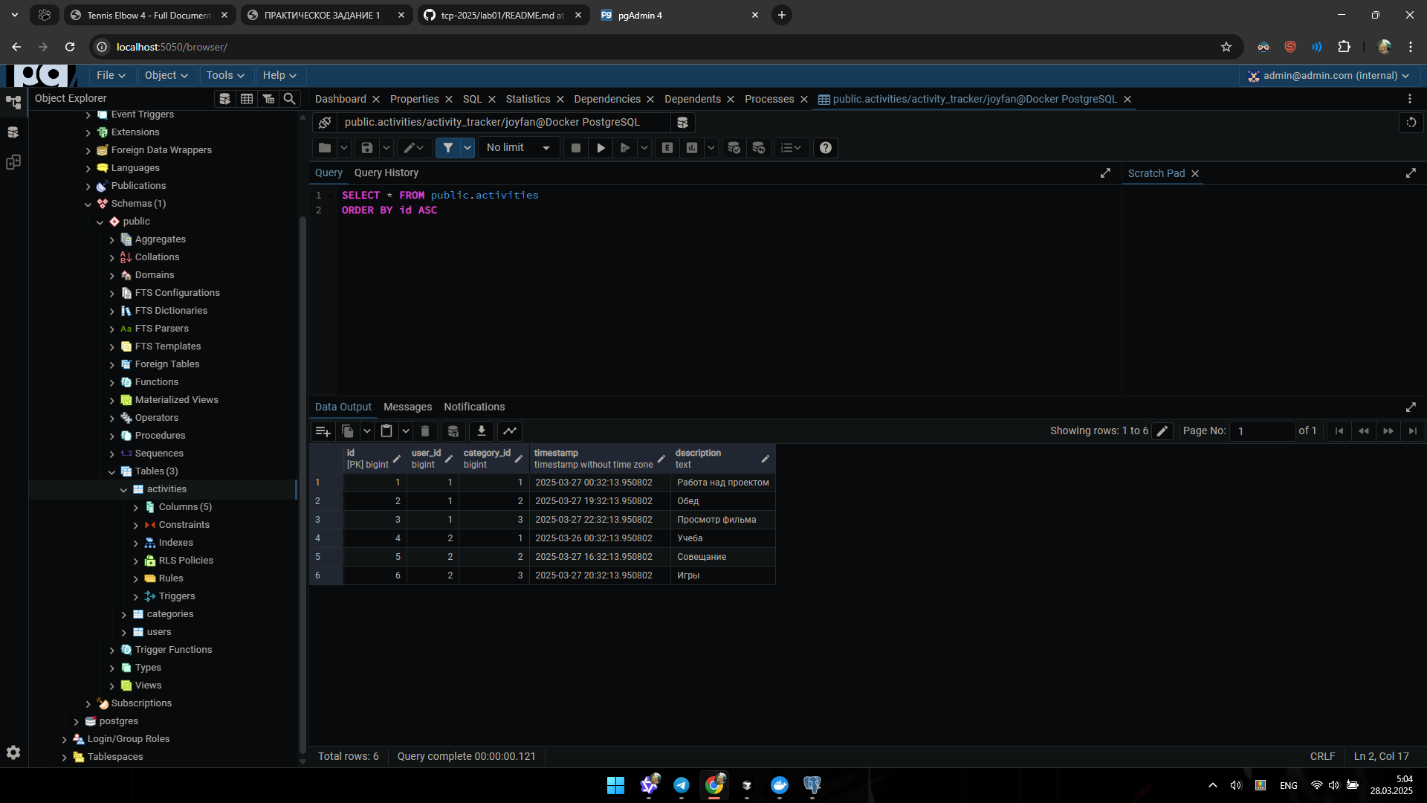
docker exec productivity-tracker-postgres-1 pg\_isready

docker exec productivity-tracker-postgres-1 pg\_isready

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

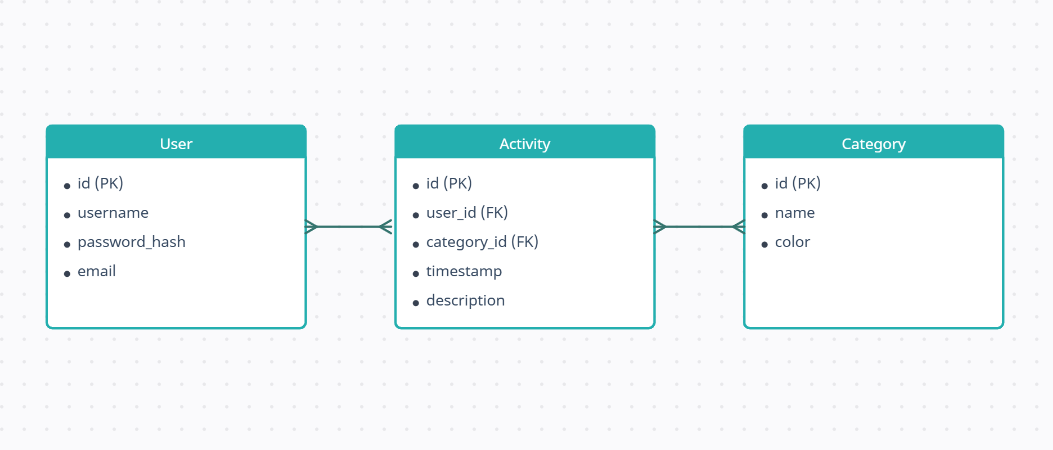
****

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

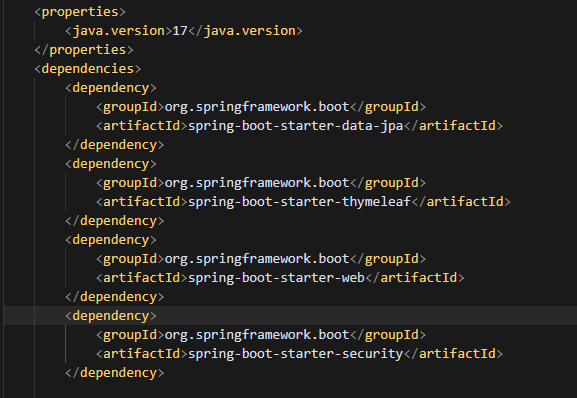
****

1. **Разработка ORM-модели**

* Структура сущностей и связей с ними имеет вид

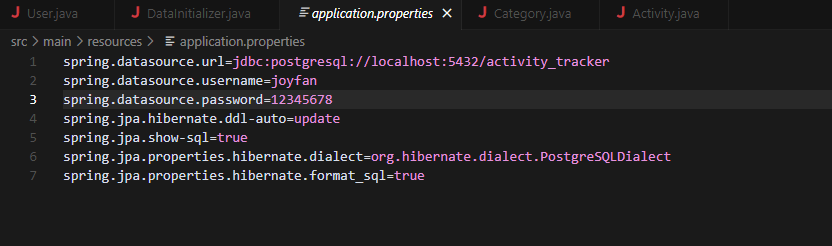


* В работе с ORM-моделью мы будем использовать **Hibernate**, **Spring Data JPA**, для этого используя **Maven**, добавляем зависимости необходимых фреймворков.

****

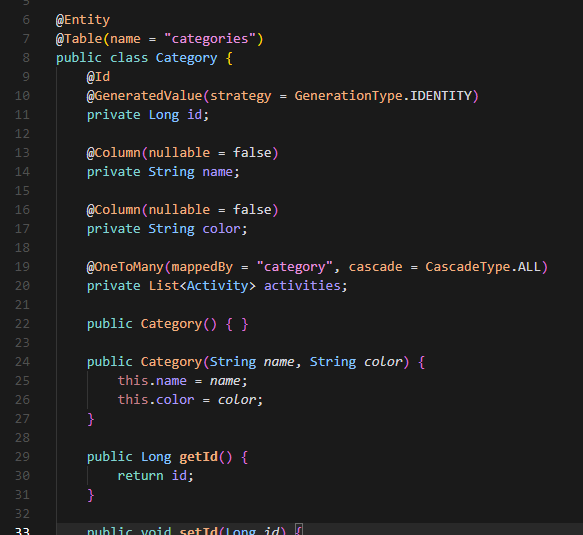
Добавление отдельной зависимости **Hibernate** не требуется, так как она содержится в зависимости **spring-boot-starter-data-jpa**.

* 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.

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:**



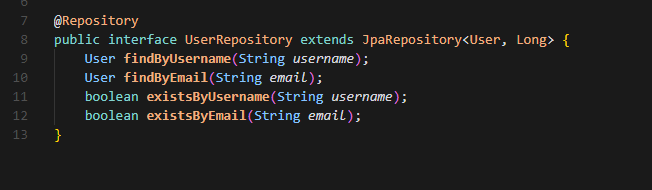
1. **Функционал для работы с моделями**

Для реализации функционала работы с моделями были выполнены Service и Repository слои приложения:



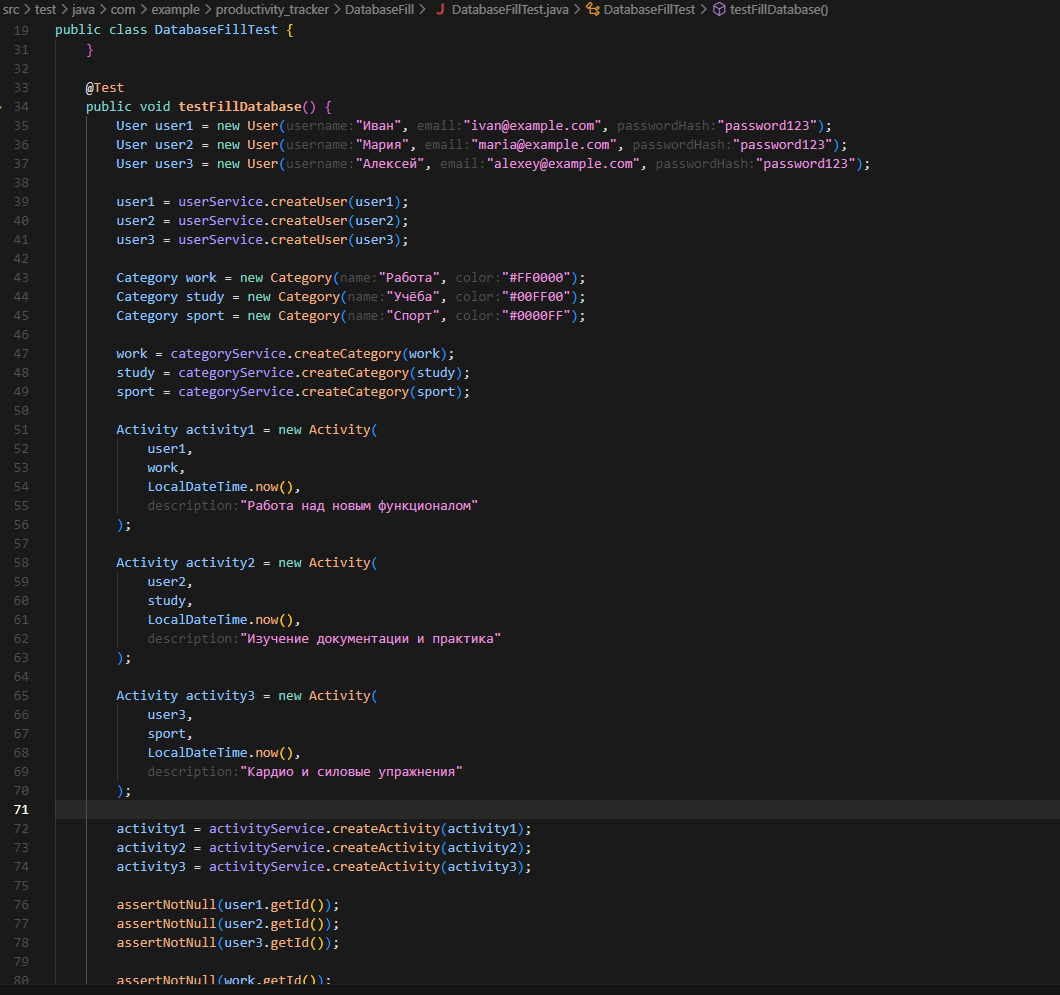
Каждый из Service классов имеет свой экземпляр соответствующего интерфейса слоя Repository, который класс использует для взаимодействия с базой данных.

Пример реализации UserRepository:



1. **Тестирование**

Заполнение тестовыми данными выполнено через отдельный класс



Пример реализации ServiceTest;

