МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4  
по дисциплине: Технологии Web-программирования

Выполнил студент: Прокопенко Н.Н.

Группы: ПВ-41

Проверил: Картамышев С. В.

Белгород 2020 г.

**Цель**: изучить основы взаимодействия web-приложения с базой данных. Спроектировать базу данных для хранения информации приложения (страницы, пользователи и т.п.).

**Задание к лабораторной работе:**

1. Выбрать подходящую СУБД.
2. Изучить методы взаимодействия web-приложения с базой данных (ORM, Active Record).
3. Разработать структуру базы данных.
4. Разработать соответствующие модели в приложении.
5. В отчёт приложить схему базы данных, а также код одной из моделей (на своё усмотрение).

**Выполнение**

Язык программирования: Java 15

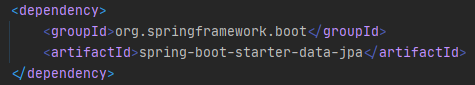
Фреймворк: Spring, Spring Boot, JPA, Hibernate

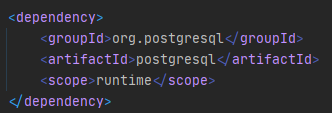
Система сборки: Maven

База данных: PostgeSQL

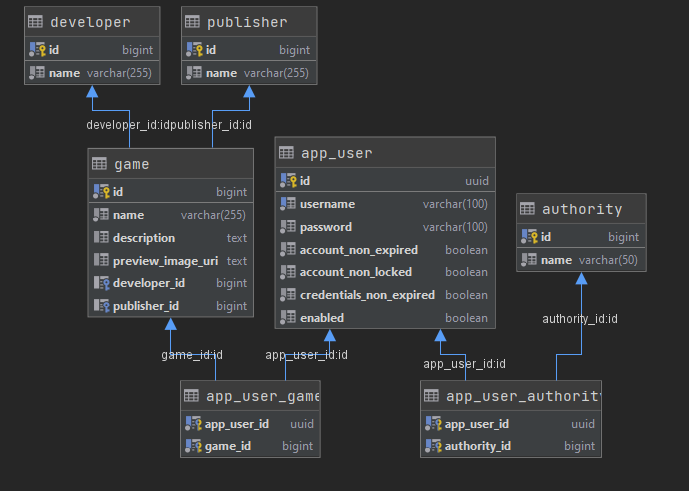
**Подключение необходимых библиотека**

Зависимости, которые нужны для работы с базой данных:





**Структура базы данных**



**Создание сущностей**

Возьмем для примера сущность Game:

package ru.brarion.steamlikeappapi.business.entity;  
  
import lombok.AccessLevel;  
import lombok.Data;  
import lombok.experimental.FieldDefaults;  
  
import javax.persistence.\*;  
import javax.validation.constraints.NotNull;  
  
@Entity**(**name = "game"**)**@Data  
@FieldDefaults**(**level = AccessLevel.***PRIVATE*)**public class Game **{** @Id  
 @GeneratedValue**(**strategy = GenerationType.***IDENTITY*)** Long **id**;  
  
 @NotNull  
 @Column  
 String **name**;  
  
 @Column  
 String **description**;  
  
 @Column  
 String **previewImageUri**;  
  
 @ManyToOne**(**fetch = FetchType.***LAZY***, cascade = CascadeType.***ALL*)** @JoinColumn**(**name = "developer\_id"**)** Developer **developer**;  
  
 @ManyToOne**(**fetch = FetchType.***LAZY***, cascade = CascadeType.***ALL*)** @JoinColumn**(**name = "publisher\_id"**)** Publisher **publisher**;  
**}**

**}**

**Создание JPA репозиториев**

Также возьмем для пример репозиторий для работы с сущностями GameRepository:

package ru.brarion.steamlikeappapi.business.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import ru.brarion.steamlikeappapi.business.entity.Game;  
  
import java.util.List;  
  
public interface GameRepository extends JpaRepository**<**Game, Long**> {** List**<**Game**>** findAllByNameStartsWith**(**String nameStart**)**;  
  
 List**<**Game**>** findAllByDeveloperNameStartsWith**(**String developerNameStart**)**;  
  
 List**<**Game**>** findAllByPublisherNameStartsWith**(**String publisherNameStart**)**;  
**}**

**Работа с базой данных**

Настройки базы данных и системы миграций Liquibase:

