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 вычислительной̆ техники и автоматизированных систем

Лабораторная работа №5  
по дисциплине: Технологии Web-программирования

Технологии web-программирования.

**«REST API»**

Выполнил: ст. гр. ПВ-41

Мишина Ю.А.

Проверил: Картамышев С. В.

Белгород 2020г.

**Цель**: изучить основы разработки API для web-приложений. Разработать REST API для своего проекта.

**Задание к лабораторной работе:**

1. Изучить структуру формата представления данных JSON.
2. Изучить типы запросов к API: HEAD, GET, POST, PUT, DELETE.
3. Спроектировать и реализовать собственное REST API (Получение, создание, изменение и удаление каких-либо объектов).
4. В отчёт необходимо предоставить документацию к использованию методов. (Либо словесным описание, либо через Swagger)

**Выполнение**

В данной лабораторной работе мы будем реализовывать REST API на строне бека, основываясь на базе данных из прошлого

Язык программирования: Java 8

Фреймворк: Spring, Spring Boot, JPA, Hibernate

Система сборки: Maven

База данных: PostgeSQL

В прошлой лабораторной работе у нас бы были готовы репозитории для работы с нашими сущностями. Теперь нужно написать бизнес-логику и контроллеры для работы с нашим беком.

**Разработка DTO**

Приходящий у нам JSON нужно мапить в какие-то сущности, для большей простоты работы с данными и конвертации их потом в сущности в базе данных.

Они выглядят например так:

@Data  
@Builder  
public class GameDto {  
  
 Long id;  
  
 String nameGame;  
  
 String platform;  
  
 String genre;  
  
 ImageDto cover;  
  
 String dateReleased;  
}

**Разработка контроллеров**

Разберём разработку контроллера на примере

@RestController  
@RequestMapping(ApiUtil.*INFO*)  
@RequiredArgsConstructor  
public class InfoController {  
  
 private final InfoService infoService;  
  
 @GetMapping("/getInfo")  
 public InfoDto getInfo() {  
 return infoService.getInfo();  
 }  
}

@RestController – аннотация, которая говорит, что наш контроллер будет получать и передавать данные в формате REST API.

@RequestMapping(ApiUtil.INFO) – так как у нас будет много контроллеров, мы создаём иерархию endpoint-ов.

После мы описываем метод.

@GetMapping("/getInfo ") – аннотация, которая указывает тип запроса и по какому url нужно обратиться к этому методу.

**Разработка мапперов**

С фронтом мы общаемся через DTO, а с базой данных мы общаемся через сущности. Чтобы преобразовывать одно в другое, мы будем применять мапера, а конкретно библиотеку mupstruct. Мы задаём интерфейс для мапера и объясняем, как поступать в некоторых ситуациях, реализацию библиотека сгенерирует сама.

@Mapper(uses = ImageMapper.class)  
public interface GameMapper {  
  
 GameDto toDto(Game game);  
  
 Game toEntity(GameDto dto);  
}

**Разработка сервисов**

Слой сервисов – это слой, который реализует бизнес-логику нашего приложения.

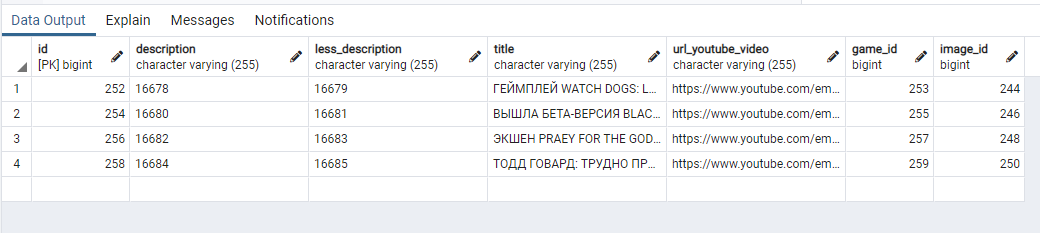
Пример:

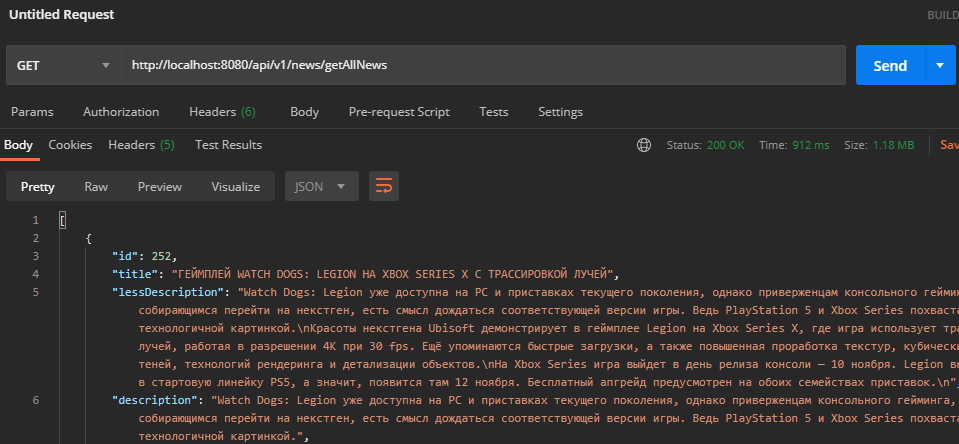
@Service  
@RequiredArgsConstructor  
public class NewsServiceImpl implements NewsService {  
  
 private final NewsRepository newsRepository;  
  
 private final NewsMapper newsMapper;  
  
 @Override  
 public List<NewsDto> getAllNews() {  
 return newsRepository.findAll()  
 .stream().map(newsMapper::toDto)  
 .collect(Collectors.*toList*());  
 }  
  
 @Override  
 public NewsDto getNewsById(Long id) {  
 News res = newsRepository.getOne(id);  
  
 return newsMapper.toDto(res);  
 }  
}

Сервис решает, что во что мапить, что сохранять в базу и так далее.

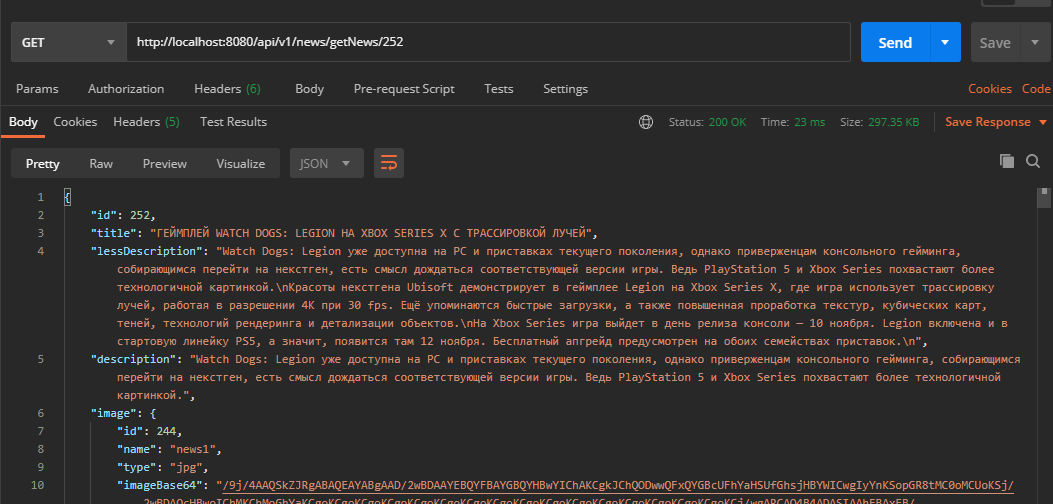
**Пример работы с REST API**

Продемонстрируем запрос на получение всех всех новостей.

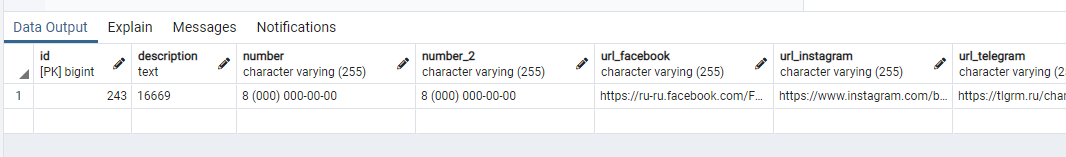


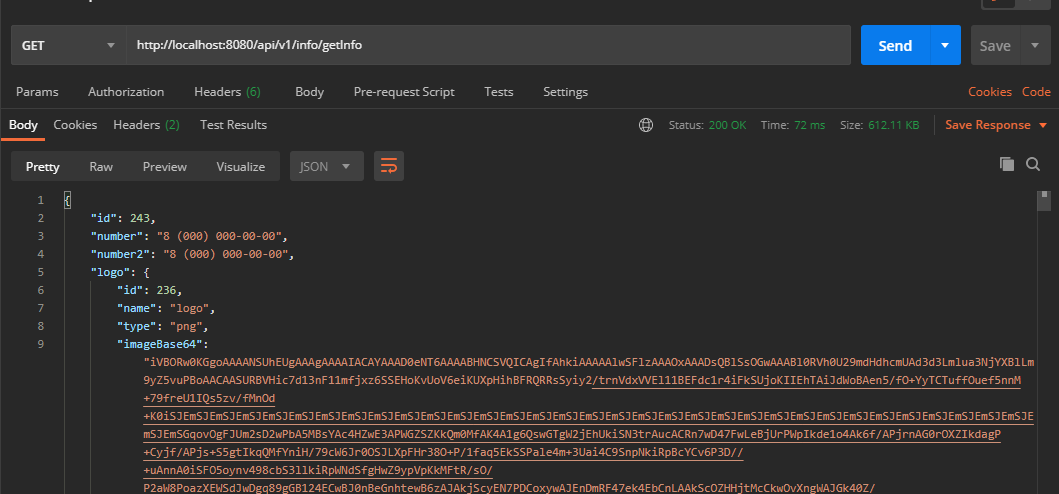


Продемонстрируем запрос на получение новостей по id.



Продемонстрируем запрос на получение информации о сайте.





Состояние других таблиц

