Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 6

Архитектура REST
тема

Преподаватель _____ А.С. Кузнецов
подпись, дата инициалы, фамилия

Студент КИ20-16/1б, 031945012 _____ Ф.В. Пироженко
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2022

СОДЕРЖАНИЕ

1 Цель

Ознакомится с механизмами поддержки архитектуры REST в Spring.

2 Задание

Изменить приложение из практического задания №5 (или №4, на усмотрение студента, т.к. при работе с защищенным приложением могут возникнуть трудности) и добавить следующий функционал:

1) Преобразовать веб-приложение таким образом, чтобы оно поддерживало архитектуру *REST*. Должны поддерживаться следующие типы запросов: *GET* (показ *html* и извлечение *json* всех/одной записей/сущностей), *POST* (добавление), *PUT* (редактирование), *DELETE* (удаление).

2) Разработать *REST*-клиент для приложения, который с использованием *RestTemplate* позволяет выполнять базовые операции по извлечению (*GET*), добавлению (*POST*), редактированию (*PUT*), удалению (*DELETE*) ресурсов. *REST*-клиент не обязан иметь пользовательский интерфейс, необходим тестовый пример, который можно запускать из консоли.

3) **Обязательным условием** является сохранение всего предшествующего функционала приложения. Для удовлетворения всем характеристикам *REST*- архитектуры приложение может быть реорганизовано (убраны *GET*-запросы с параметрами) или добавлен новый функционал.

4) *PUT*- и *DELETE*-запросы не обязательно делать из браузера. Достаточно реализации для клиентов-приложений.

Результат работы - *tar.gz*-архив или текстовый файл со ссылкой на репозиторий (код проекта должен содержать документирующие комментарии) с инструкциями по развертыванию приложения и его зависимостей. Работа должна быть защищена преподавателю. Защита состоит в пояснении хода действий для решения поставленной задачи и ответов на теоретические вопросы или вопросы, возникающие у преподавателя в ходе проверки работы.

Вариант 20. Настольная игра

3 Ход выполнения

3.1 Реализация программы

Решение задания, представлено в листингах 1-19.

Листинг 1 – конфигурационный файл pom (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://ma-
ven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>ru.sfu</groupId>
  <artifactId>Lab6</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>Lab6 Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
    <spring.version>5.3.10</spring.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
      <groupId>org.springframework</groupId>
```

Продолжение листинга 1

```
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -
->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc --
>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf-spring5 -->
    <dependency>
      <groupId>org.thymeleaf</groupId>
      <artifactId>thymeleaf-spring5</artifactId>
      <version>3.0.11.RELEASE</version>
    </dependency>
```

Продолжение листинга 1

```
    <!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-
validator -->
    <dependency>
      <groupId>org.hibernate.validator</groupId>
      <artifactId>hibernate-validator</artifactId>
      <version>6.1.7.Final</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entityman-
ager -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-entitymanager</artifactId>
      <version>5.5.7.Final</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>5.2.8.RELEASE</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.data</groupId>
      <artifactId>spring-data-jpa</artifactId>
      <version>2.3.3.RELEASE</version>
    </dependency>

    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.2.15</version>
```

Продолжение листинга 1

```
    </dependency>

    <dependency>
      <groupId>org.jetbrains</groupId>
      <artifactId>annotations</artifactId>
      <version>RELEASE</version>
      <scope>compile</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context-support</artifactId>
      <version>5.3.10</version>
    </dependency>

    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.11.3</version>
    </dependency>
  </dependencies>


  <build>
    <finalName>Lab6</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven de-
faults (may be moved to parent pom) -->
      <plugins>
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
        <!-- see http://maven.apache.org/ref/current/maven-core/default-bind-
ings.html#Plugin_bindings_for_war_packaging -->
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
```

Окончание листинга 1

```
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.2.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
      </plugins>
    </pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>8</source>
          <target>8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Листинг 2 – Файл index (index.jsp)

```
<html>
<body>
<h2>Hello!</h2>
<a href="/table_games">Begin to work</a>
</body>
</html>
```

Листинг 3 – Файл web (web.xml)

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
</web-app>
```

Листинг 4 – Файл add (add.html)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Add New Game</title>
</head>
<body>
<a href="/table_games">Back</a>
<form th:method="POST" th:action="@{/table_games}" th:object="${game}">
    <label for="gameName">Enter name: </label>
    <input type="text" th:field="*{gameName}" id="gameName"/>
    <div th:if="${#fields.hasErrors('gameName')}" th:errors="*{gameName}">ER-
ROR</div>
    <br/>

    <label for="price">Price </label>
    <input name="price" type="number" id="price"/>
    <div th:if="${#fields.hasErrors('price')}" th:errors="*{price}">ERROR</div>
    <br/>

    <label for="genre">Genre </label>
    <input type="text" th:field="*{genre}" id="genre"/>
    <div th:if="${#fields.hasErrors('genre')}" th:errors="*{genre}">ERROR</div>
    <br/>

    <label for="playerAmount">Player amount </label>
    <input name="playerAmount" type="number" id="playerAmount"/>
    <div th:if="${#fields.hasErrors('playerAmount')}" th:errors="*{playerA-
mount}">ERROR</div>
    <br/>
```

Окончание листинга 4

```
    <input type="submit" value="Add game"/>
</form>
</body>
</html>
```

Листинг 5 – Файл edit (edit.html)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Edit Game</title>
</head>
<body>
<a href="/table_games">Back</a>
<form th:method="POST" th:action="@{/table_games/edit/{id}(id=${game.getId()})}"
th:object="${game}">
    <label for="idx">ID</label>
    <input th:field="*{id}" type="number" id="idx" disabled/>
    <br/>

    <label for="gameName">Enter name: </label>
    <input type="text" th:field="*{gameName}" id="gameName"/>
    <div th:if="${#fields.hasErrors('gameName')}" th:errors="*{gameName}">ER-
ROR</div>
    <br/>

    <label for="price">Price </label>
    <input th:field="*{price}" type="number" id="price"/>
    <div th:if="${#fields.hasErrors('price')}" th:errors="*{price}">ERROR</div>
    <br/>

    <label for="genre">Genre </label>
    <input type="text" th:field="*{genre}" id="genre"/>
    <div th:if="${#fields.hasErrors('genre')}" th:errors="*{genre}">ERROR</div>
    <br/>

    <label for="playerAmount">Player amount </label>
    <input th:field="*{playerAmount}" type="number" id="playerAmount"/>
    <div th:if="${#fields.hasErrors('playerAmount')}" th:errors="*{playerA-
mount}">ERROR</div>
    <br/>
```

Окончание листинга 5

```
    <input type="submit" value="Edit game"/>
</form>

<form th:method="POST" th:action="@{/table_games/de-
lete/{id}(id=${game.getId()})}" th:object="${game}">
    <input type="submit" value="Delete Game"/>
</form>
</body>
</html>
```

Листинг 6 – Файл find (find.html)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Find</title>
</head>
<body>
<a href="/table_games">Back</a>
<form th:method="GET" th:action="@{/table_games/findGame}">
    <label for="price">Max price</label>
    <input type="number" id="price" name="price"/>
    <br/>
    <input type="submit" value="Find"/>
</form>
</body>
</html>
```

Листинг 7 – Файл menu (menu.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Menu</title>
</head>
<body>
<a href="/table_games/add">Add game</a><br/>
<a href="/table_games/show">Show all games</a><br/>
<a href="/table_games/find">Find games</a><br/>
</body>
</html>
```

Листинг 8 – Файл show (show.html)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Show</title>
</head>
<body>
    <a href="/table_games">Back</a>
    <div th:each="game : ${games}">
        <a th:href="@{{id}/edit(id=${game.getId()})}"
th:text="${game.toString()}">game</a>
    </div>
</body>
</html>
```

Листинг 9 – Файл showGame (showGame.html)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>th:text="${game.getGameName()}"</title>
</head>
<body>
<a href="/table_games">Back</a>
<p th:text="${game.toString()} ">game</p>
</body>
</html>
```

Листинг 10 – Файл application (application.properties)

```
#--- Postgres ---
dataSource.driverClassName =org.postgresql.Driver
jpa.database=POSTGRESQL
dataSource.url=jdbc:postgresql://localhost:5432/lab3_db
dataSource.username=postgres
dataSource.password=qwerty
```

Листинг 11 – Класс DispServletInit(DispServInit.java)

```
package ru.sfu.config;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatch-
erServletInitializer;
```

Окончание листинга 11

```
public class DispServletInit extends AbstractAnnotationConfigDispatcherServlet-
Initializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] {SpringConfig.class};
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] {"/"};
    }
}
```

Листинг 12 – Класс SpringConfig (SpringConfig.java)

```
package ru.sfu.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

import javax.sql.DataSource;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.thymeleaf.spring5.SpringTemplateEngine;
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;
import org.thymeleaf.spring5.view.ThymeleafViewResolver;
import ru.sfu.model.TableGame;
import ru.sfu.dao.TableGameDAO;
```

```
@Configuration
@ComponentScan("ru.sfu.*")
@PropertySource("classpath:application.properties")
@EnableWebMvc
public class SpringConfig implements WebMvcConfigurer
{
    @Autowired
    private Environment env;

    @Bean
    DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();

        dataSource.setDriverClassName(env.getProperty("dataSource.driverClass-
Name"));
        dataSource.setUrl(env.getProperty("dataSource.url"));
        dataSource.setUsername(env.getProperty("dataSource.username"));
        dataSource.setPassword(env.getProperty("dataSource.password"));
        return dataSource;
    }

    @Bean
    TableGameDAO tableGameDAO() {
        TableGameDAO tableGameDAO =  new TableGameDAO();
        tableGameDAO.setDateSource(dataSource());
        return  tableGameDAO;
    }

    private final ApplicationContext applicationContext;

    @Autowired
    public SpringConfig(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver templateResolver = new SpringResourceTem-
plateResolver();
        templateResolver.setApplicationContext(applicationContext);
        templateResolver.setPrefix("/WEB-INF/views/");
```

Окончание листинга 12

```
        templateResolver.setSuffix(".html");
        return templateResolver;
    }

    @Bean
    public SpringTemplateEngine templateEngine() {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver());
        templateEngine.setEnableSpringELCompiler(true);
        return templateEngine;
    }

    @Override
    public void configureViewResolvers(ViewResolverRegistry registry) {
        ThymeleafViewResolver resolver = new ThymeleafViewResolver();
        resolver.setTemplateEngine(templateEngine());
        registry.viewResolver(resolver);
    }

}
```

Листинг 13 – Класс DataConfig (DataConfig.java)

```
package ru.sfu.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.EnableTransactionManagement;
import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
@Configuration
@EnableTransactionManagement
@ComponentScan("ru.sfu.repository")
```

Окончание листинга 13

```
@EnableJpaRepositories("ru.sfu.repository")
@PropertySource("classpath:application.properties")
public class DataConfig {

    private final Environment env;

    public DataConfig(Environment env) {
        this.env = env;
    }
    @Bean
    DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();

        dataSource.setDriverClassName(env.getProperty("dataSource.driverClass-
Name"));
        dataSource.setUrl(env.getProperty("dataSource.url"));
        dataSource.setUsername(env.getProperty("dataSource.username"));
        dataSource.setPassword(env.getProperty("dataSource.password"));
        return dataSource;
    }
    @Bean
    public EntityManagerFactory entityManagerFactory() {
        HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVen-
dorAdapter();
        vendorAdapter.setGenerateDdl(true);
        LocalContainerEntityManagerFactoryBean factory = new
                LocalContainerEntityManagerFactoryBean();
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("ru.sfu");
        factory.setDataSource(dataSource());
        factory.afterPropertiesSet();
        return factory.getObject();
    }
    @Bean
    public PlatformTransactionManager transactionManager() {
        JpaTransactionManager txManager = new JpaTransactionManager();
        txManager.setEntityManagerFactory(entityManagerFactory());
        return txManager;
    }
}
```

Листинг 14 – Класс TableGamesController (TableGamesController.java)

```
package ru.sfu.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import ru.sfu.model.TableGame;
import ru.sfu.repository.TableGameRepository;

import javax.validation.Valid;
import java.util.Optional;


@Controller
@RequestMapping("/table_games")
public class TableGameController {
    private final TableGameRepository repo;

    public TableGameController(TableGameRepository repo) {
        this.repo = repo;
    }

    @GetMapping("/menu")
    public String menu(){
        return "table_games/menu";
    }

    @GetMapping(value = "/{id}")
    public String showGame(@PathVariable("id") int id, Model model){
        Optional<TableGame> tg = repo.findById(id);
        if (tg.isPresent()) {
            model.addAttribute("game", tg.get());
            return "table_games/showGame";
        }

        return "redirect:/table_games";
    }


    @GetMapping()
```

Продолжение листинга 14

```
public String showGames(Model model){
    model.addAttribute("games", repo.findAll());
    return "table_games/show";
}

@GetMapping("/new")
public String newGame(Model model){
    model.addAttribute("game", new TableGame());
    return "table_games/add";
}

@PostMapping()
public String addGame(@ModelAttribute("game") @Valid TableGame game,
                      BindingResult bindingResult){
    if (bindingResult.hasErrors()){
        return "table_games/add";
    }

    repo.save(game);
    return "redirect:/table_games";
}



@GetMapping("/{id}/edit")
public String editGame(@PathVariable("id") int id, Model model){
    Optional<TableGame> tg = repo.findById(id);
    if (tg.isPresent()){
        model.addAttribute("game", tg.get());
        return "table_games/edit";
    }


    return "redirect:/table_games";
}

@PatchMapping("/{id}")
public String updateGame(@PathVariable("id") int id,
                         @ModelAttribute("game") @Valid TableGame game,
                         BindingResult bindingResult){
```

Окончание листинга 14

```
        if (bindingResult.hasErrors()){
            return "table_games/edit";
        }
        repo.save(game);
        return "redirect:/table_games";
    }

    @DeleteMapping("/{id}")
    public String deleteGame(@PathVariable("id") int id){
        if (repo.existsById(id)) {
            repo.deleteById(id);
        }
        return "redirect:/table_games";
    }

    @GetMapping("maxPrice")
    public String findGame(@RequestParam("price") int maxPrice, Model model)
    {
        model.addAttribute("games", repo.findAllByPriceIsLessThanEqual(max-
Price));

        return "/table_games/show";
    }

    @GetMapping("search")
    public String showFoundedGames(){
        return "table_games/find";
    }
}
```

Листинг 15 – Класс TableGame (TableGame.java)

```
package ru.sfu.model;

import javax.persistence.*;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Size;


@Entity
```

Продолжение листинга 15

```
@Table(name = "table_games")
public class TableGame {

    @Id
    @GeneratedValue
    int id;

    @Column(name = "gamename")
   @NotEmpty(message = "Name should not be empty")
   @Size(min = 3, max = 30, message="Name should  be between 3 and 30")
    String gameName;

    @Column(name = "price")
    @Min(value = 1, message = "price should be greater than 0")
    @Max(value = 9000, message = "price should be less than 9000")
    int price;

    @Column (name = "playeramount")
    @Min(value = 1, message = "player amount should be greater than 0")
    @Max(value = 20, message = "player amount should be less than 20")
    int playerAmount;

    @Column (name = "genre")
    @NotEmpty(message = "Genre should not be empty")
    @Size(min = 3, max = 30, message="Genre should  be between 3 and 30")
    String genre;

    public TableGame(){}

    public TableGame(int id, String gamename, int price, int playerAmount,
String genre) {
        this.id = id;
        this.gameName = gamename;
        this.price = price;
        this.playerAmount = playerAmount;
        this.genre = genre;
    }

    public int getId() {
        return id;
```

Окончание листинга 15

```
    }

    public void setId(int id) {
        this.id = id;
    }
    public String getGameName() {
        return gameName;
    }
    public void setGameName(String gameName) {
        this.gameName = gameName;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }

    public int getPlayerAmount() {
        return playerAmount;
    }

    public void setPlayerAmount(int playerAmount) {
        this.playerAmount = playerAmount;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    @Override
    public String toString(){
        return "id: " + id + ", name: " + gameName + ", price: " + price + ",
playersAmount: " + playerAmount + ", genre: " + genre;
    }
}
```

Листинг 16 – Интерфейс TableGameRepository (TableGameRepository.java)

```
package ru.sfu.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import ru.sfu.model.TableGame;

import java.util.List;
import java.util.Optional;

@Repository
public interface TableGameRepository extends CrudRepository<TableGame, Integer>
{
    List<TableGame> findAllByPriceIsLessThanEqual(int value);

}
```

Листинг 17 – Класс TableGameRest (TableGameRest.java)

```
package ru.sfu.rest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import ru.sfu.model.TableGame;
import ru.sfu.service.TableGameService;

import java.util.List;

@RestController
@RequestMapping("/rest")
public class TeblaGameRest {

    @Autowired
    TableGameService service;

    @PostMapping()
    public ResponseEntity<?> addGame(@RequestBody TableGame tg){
        service.create(tg);
        return new ResponseEntity<>(HttpStatus.CREATED);
    }
```

Окончание листинга 17

```
    @GetMapping
    public ResponseEntity<List<TableGame>> showGames(){
        List<TableGame> tableGames = service.readAll();

        return tableGames != null &&  !tableGames.isEmpty()
                ? new ResponseEntity<>(tableGames, HttpStatus.OK)
                : new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @GetMapping(value = "/{id}")
    public ResponseEntity<TableGame> read(@PathVariable(name = "id") int id) {
        final TableGame tg = service.read(id);

        return tg != null
                ? new ResponseEntity<>(tg, HttpStatus.OK)
                : new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PatchMapping("/{id}")
    public ResponseEntity<?> updateGame(@PathVariable("id") int id,
                            @RequestBody TableGame tg){
        tg.setId(id);
        final boolean updated = service.update(tg, id);
        return  updated
                ? new ResponseEntity<>(HttpStatus.OK)
                : new ResponseEntity<>(HttpStatus.NOT_MODIFIED);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?>  deleteGame(@PathVariable("id") int id){
        final boolean deleted = service.delete(id);

        return deleted
                ? new ResponseEntity<>(HttpStatus.OK)
                : new ResponseEntity<>(HttpStatus.NOT_MODIFIED);
    }
}
```

Листинг 18 – Класс TableGameServiceImp (TableGameServiceImp.java)

```
package ru.sfu.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.sfu.model.TableGame;
import ru.sfu.repository.TableGameRepository;

import java.util.List;
import java.util.Optional;

@Service
public class TableGameServiceImp implements TableGameService {

    @Autowired
    TableGameRepository repo;

    @Override
    public void create(TableGame tg) {
        repo.save(tg);
    }

    @Override
    public List<TableGame> readAll() {
        return (List<TableGame>) repo.findAll();
    }

    @Override
    public TableGame read(Integer id) {
        Optional<TableGame> tg = repo.findById(id);
        if (tg.isPresent())
            return tg.get();
        return  null;
    }

    @Override
    public boolean update(TableGame tg, Integer id) {
        if (repo.existsById(id)){
            repo.save(tg);
            return true;
        }
```

Окончание листинга 18

```
        return false;
    }

    @Override
    public boolean delete(Integer id) {
        if (repo.existsById(id)) {
            repo.deleteById(id);
            return true;
        }
        return false;
    }
}
```

Листинг 19 – ИнтерфейсTableGameService (TableGameService.java)

```
package ru.sfu.service;

import ru.sfu.model.TableGame;

import java.util.List;

public interface TableGameService {
    void create(TableGame tg);
    List<TableGame> readAll();
    TableGame read(Integer id);
    boolean update(TableGame tg, Integer id);
    boolean delete(Integer id);
}
```

3.2 Демонстрация работы программы

Демонстрация проделанной работы представлена на рисунках 1-7.

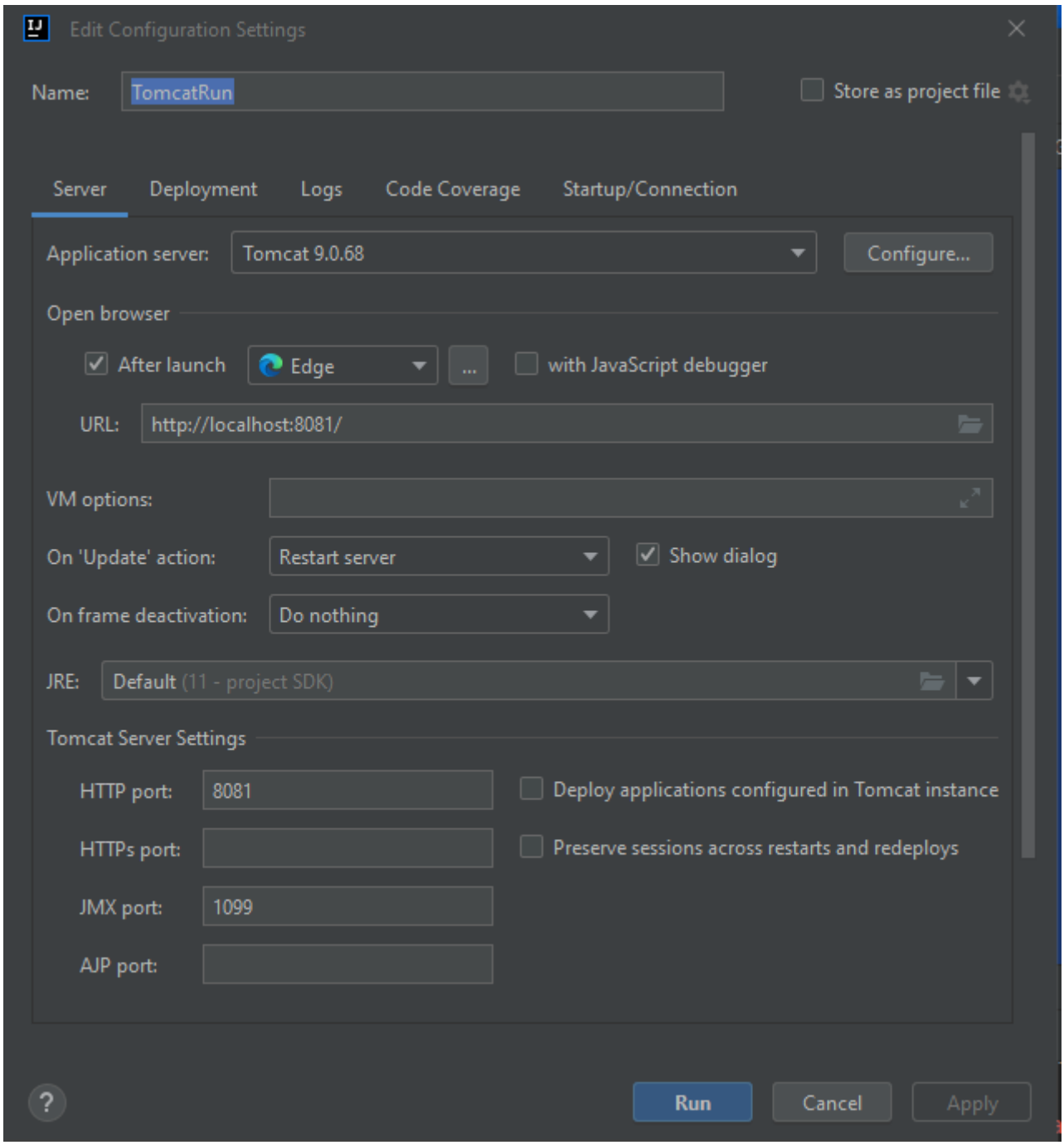

Рисунок 1 – Настройка Tomcat

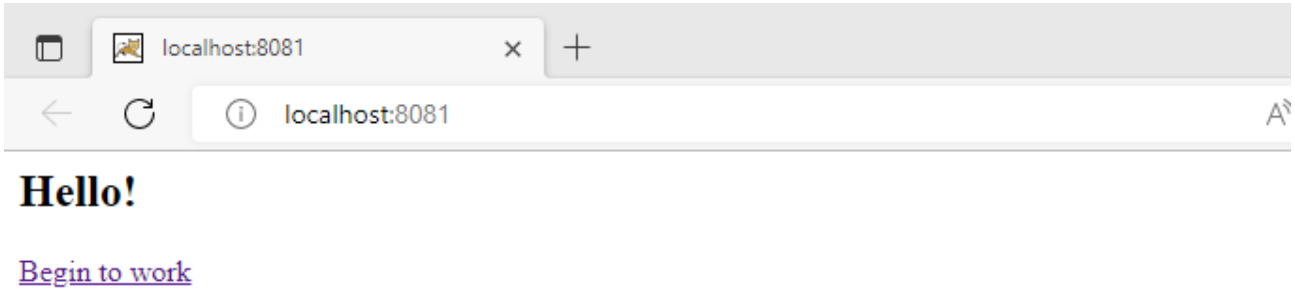

Рисунок 2 – Главная страница

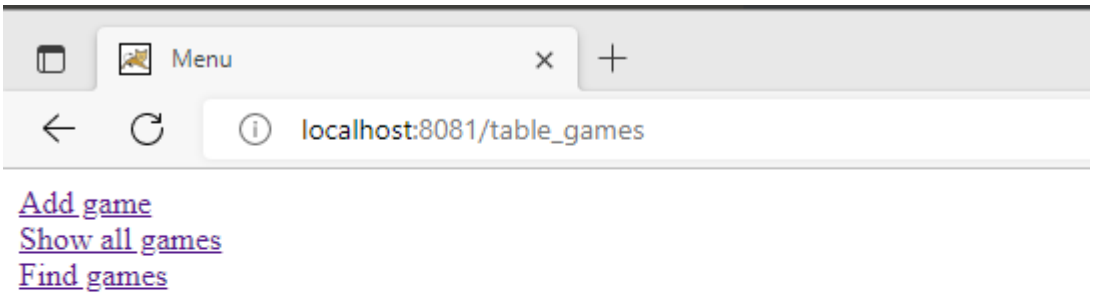

Рисунок 3 – Страница меню

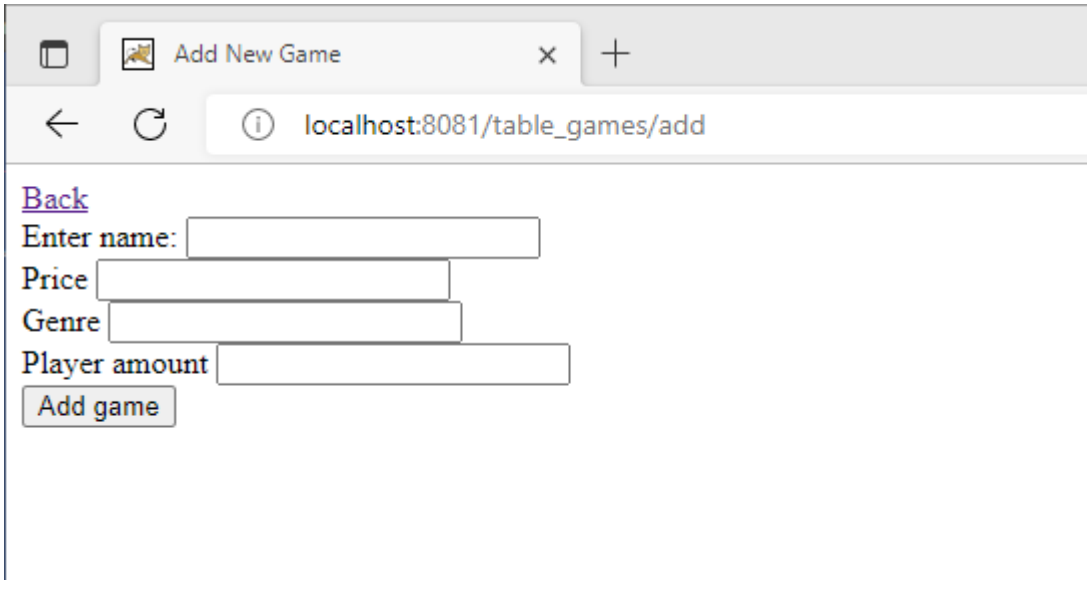

Рисунок 4 – Страница добавления новой записи

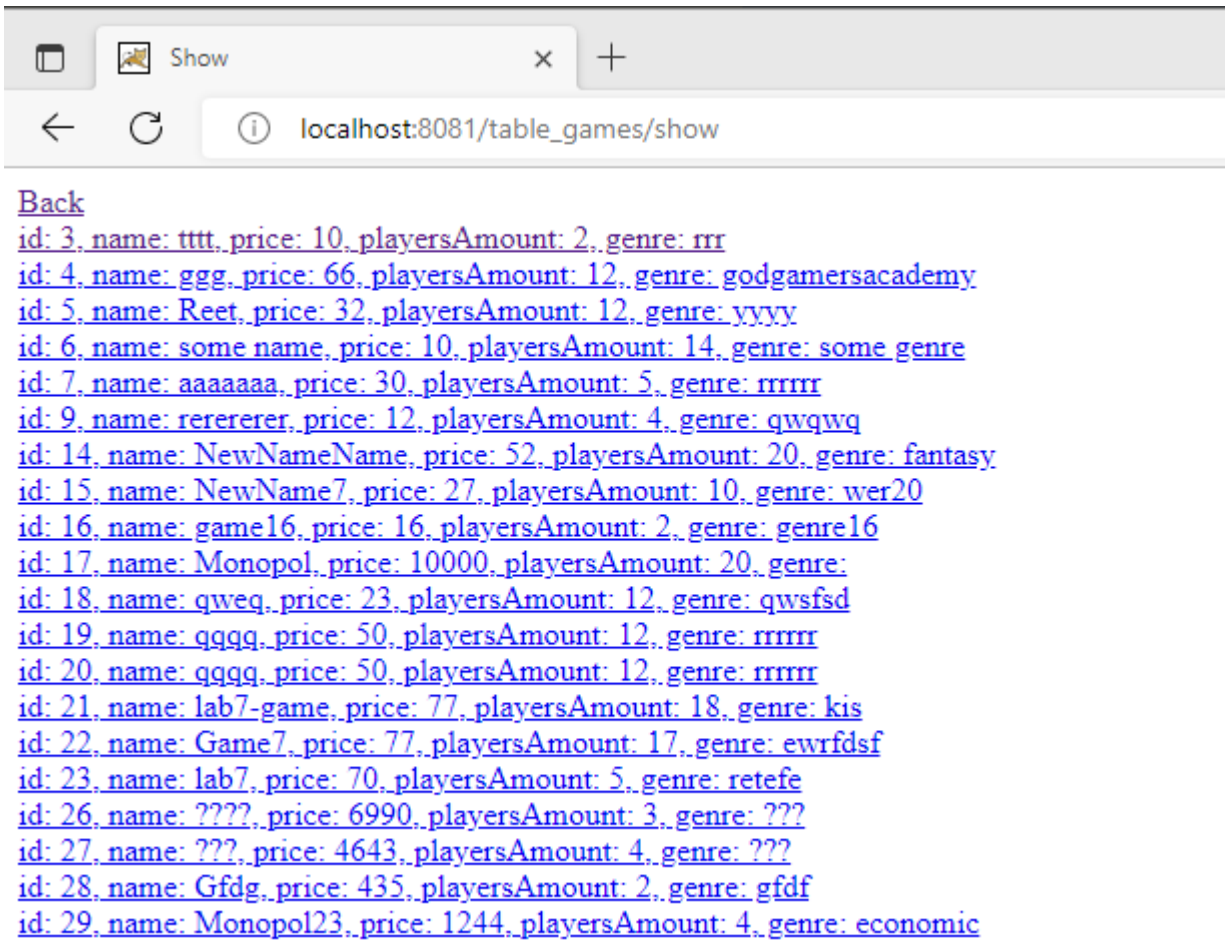

Рисунок 5 – Страница просмотра записей

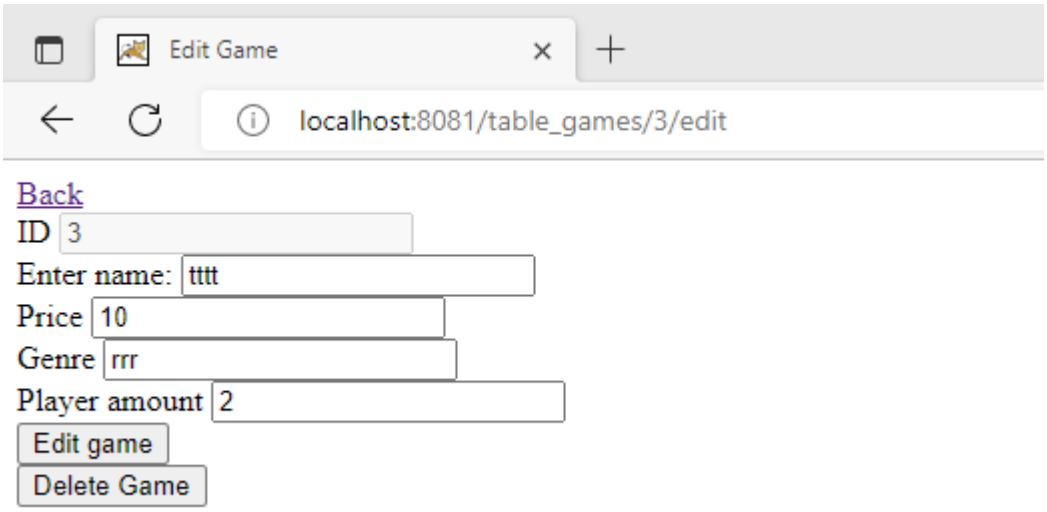

Рисунок 6 – Страница редактирования и удаления записи

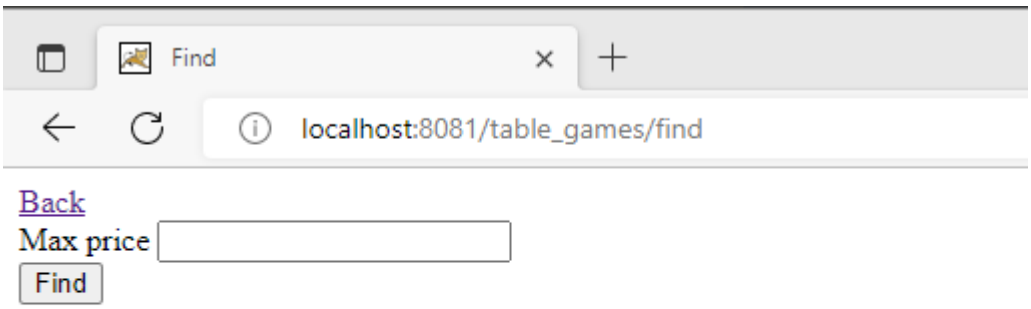

Рисунок 7 – Страница поиска записи по критерию максимальной цены

Работа с REST представлена на рисунках 8 – 13.

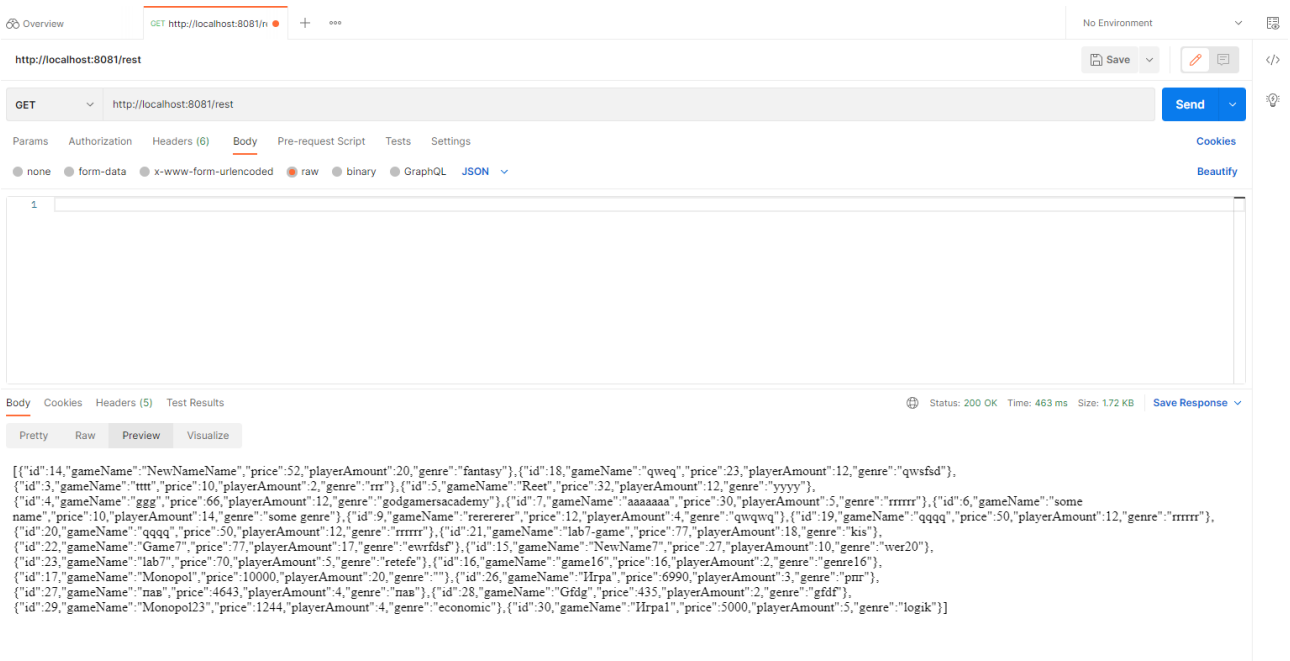

Рисунок 8 – Вывод всех записей

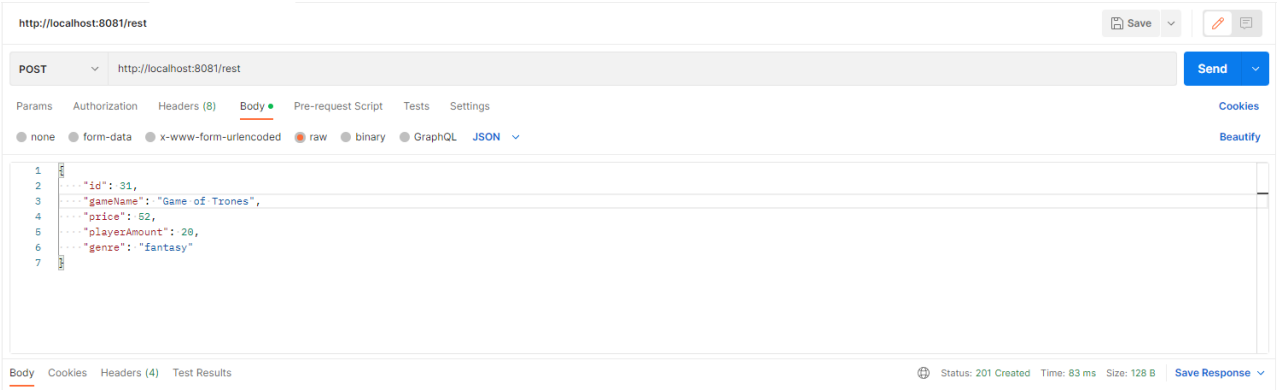

Рисунок 9 – Создание записи

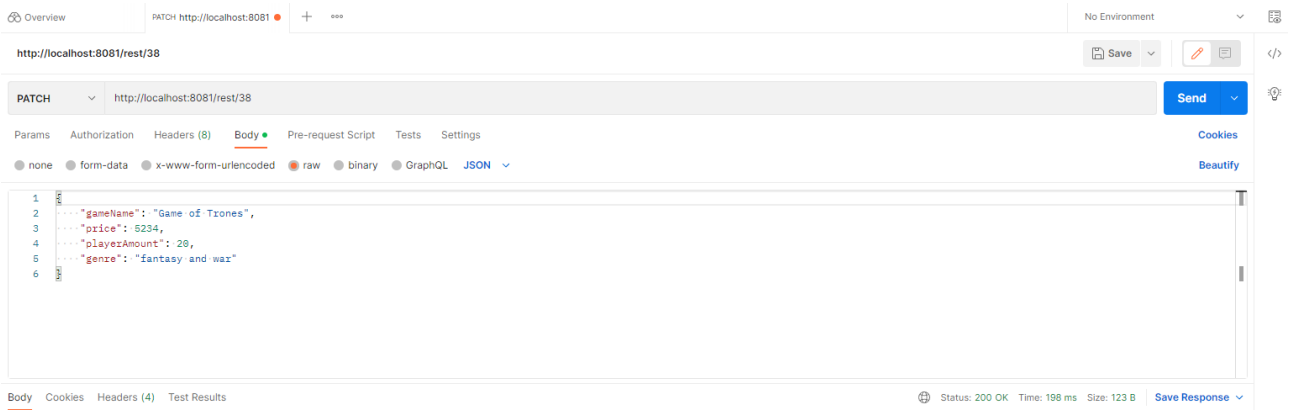

Рисунок 10 – Изменение записи

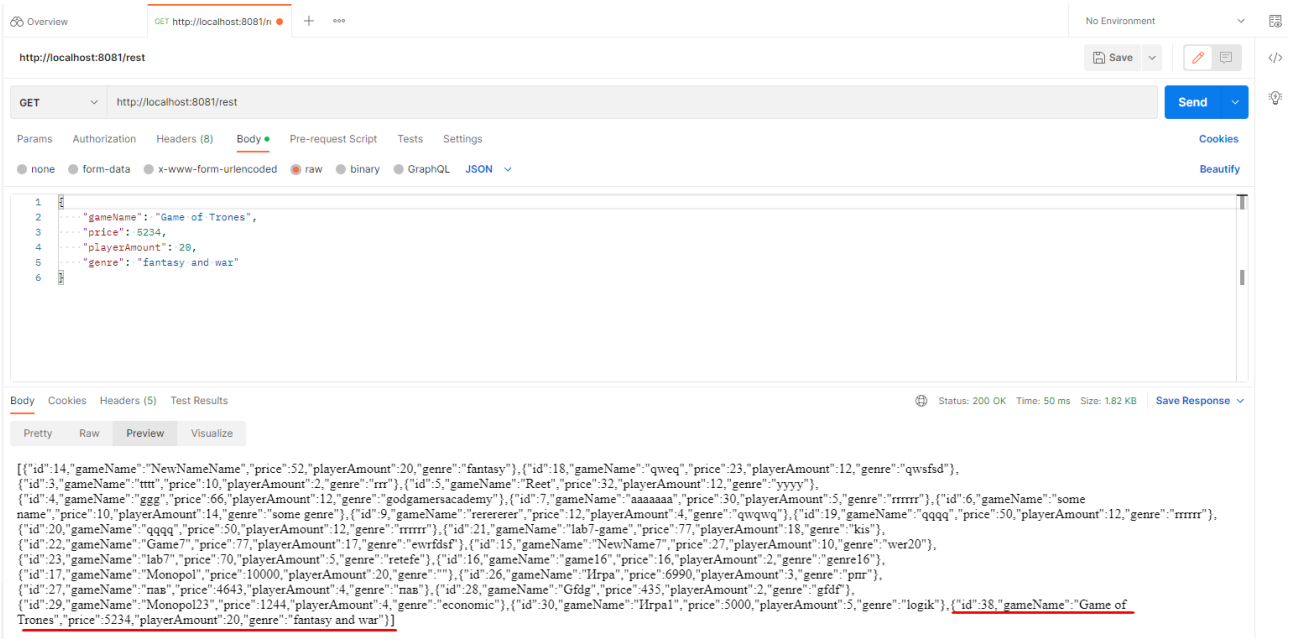

Рисунок 11 – Результат изменения записи

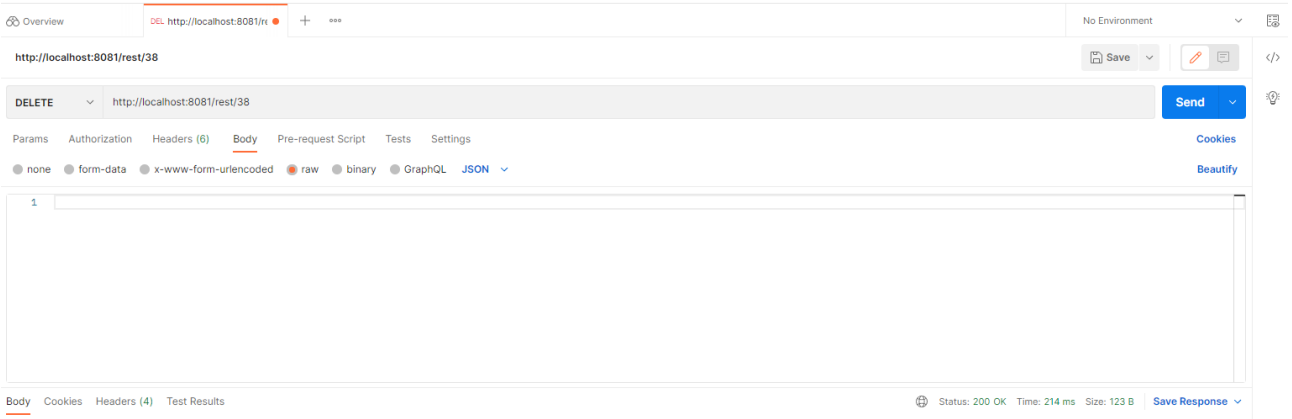

Рисунок 12 – Удаление записи

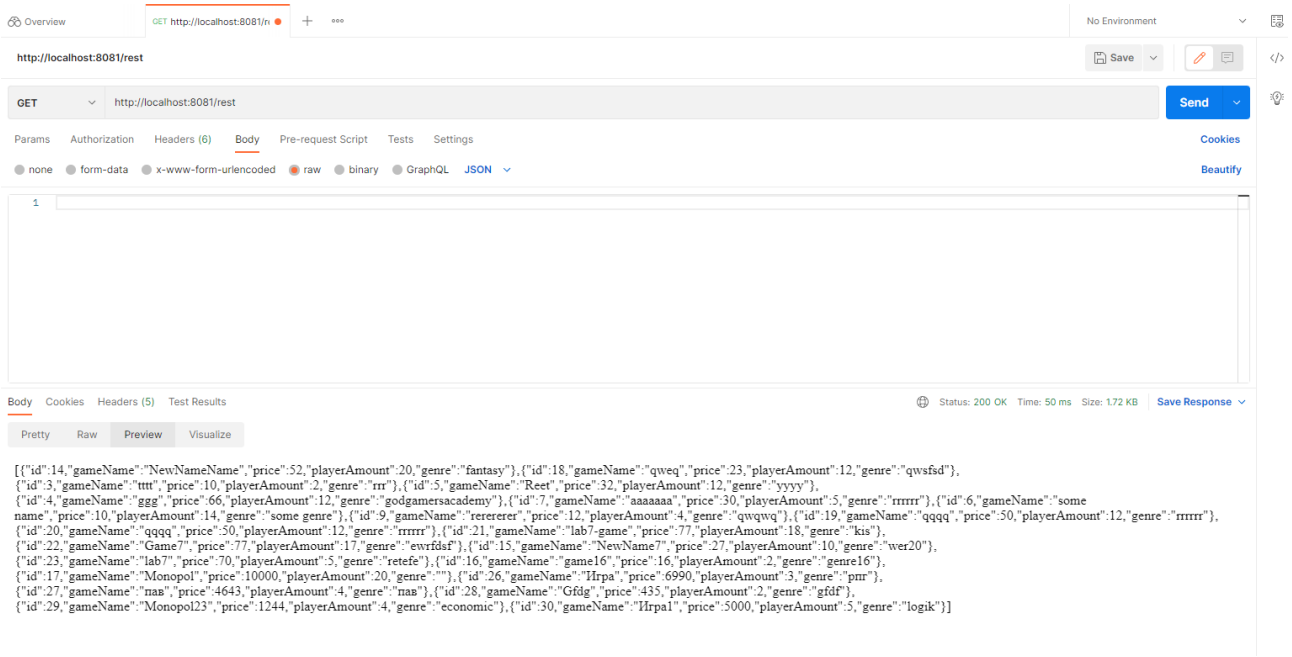

Рисунок 13 – Результат удаления записи

Вывод

В результате проделанной практической работы была выполнена следующая задача:

- были ознакомлены с механизмом поддержки архитектуры REST в Spring.