НеКОММЕРЧЕСКОЕ ЧАСТНОЕ учреждение ПРОФЕССИОНАЛЬНОГО образования

«Колледж управления и экономики»

|  |
| --- |
|  |

ОТЧЕТ

по учебной практике УП.02

по профессиональному модулю

ПМ 0.2 Осуществление интеграции программных модулей

реализуемому

в программе подготовки специалистов среднего звена

по специальности *09.02.07 Информационные системы и программирование*

Студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

учебной группы\_\_\_\_\_\_\_\_\_\_ (ФИО студента)

Тольятти

2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc161783020)

[Практическая работа №1. Разработка и оформление технического задания 4](#_Toc161783021)

[Практическая работа №2. Разработка программного продукта 8](#_Toc161783022)

[*База данных* 8](#_Toc161783023)

[*Рабочая среда программирования* 10](#_Toc161783024)

[*Разработка сайта «Музеи городов России»* 11](#_Toc161783025)

[Практическая работа №3. Тестирование и отладка программного продукта 20](#_Toc161783026)

[Практическая работа №4. Разработка руководства пользователя 23](#_Toc161783027)

[Список использованных источников 24](#_Toc161783028)

# **ВВЕДЕНИЕ**

В современном мире цифровых технологий важное место занимает разработка интерактивных веб-сайтов, предоставляющих пользователям уникальный и захватывающий опыт взаимодействия онлайн. Один из таких увлекательных проектов – интерактивный веб-сайт, посвященный музеям городов России. Виртуальные музейные пространства позволяют не только познакомиться с историей и культурным наследием городов России, но и участвовать в увлекательном путешествии по их экспозициям не выходя из дома.

Россия – удивительная страна с богатой историей, уникальной культурой и множеством интересных исторических мест и музеев. Этот проект предоставляет возможность не только углубить свои знания о культурном наследии городов России, но и почувствовать себя участником увлекательного и познавательного путешествия.

Основной целью данного интерактивного веб-сайта является создание уникального и интересного опыта для пользователей, позволяющего им узнать больше о культурном наследии городов России, их музеях и выставках. Виртуальное посещение музеев городов России позволит расширить кругозор, обогатить свои знания и погрузиться в историю и культуру нашей страны. Этот проект стремится стать ценным ресурсом для всех ценителей истории, искусства и культуры, приглашая каждого совершить увлекательное путешествие через виртуальные коридоры музеев городов России.

Разработанный сайт должен предоставить пользователю базовый интерфейс, который показывает список музеев и информацию о каждом музее.

Функции, реализуемые в проекте

На странице сайта реализованы следующие функции:

- блок с информацией о музеях

- блок с информацией о выбранном музее

# **Практическая работа №1. Разработка и оформление технического задания**

Для разработки сайта выбрал фреймворк Bootstrap.

Bootstrap - это свободный и открытый фреймворк для разработки веб-сайтов с использованием HTML, CSS и JavaScript. Он позволяет создавать адаптивные и красиво оформленные веб-сайты с удобным и интуитивно понятным дизайном.

Чтобы начать разработку сайта с использованием Bootstrap, необходимо подключить библиотеку Bootstrap к проекту. Сначала нужно загрузить и подключить необходимые файлы Bootstrap к проекту. Bootstrap предоставляет готовые компоненты и стили, которые упрощают создание адаптивных и современных веб-сайтов. Используйте его функционал для создания красивого и удобного веб-приложения.

Второй компонент, который понадобился при разработке сайта, это Thymeleaf.

Thymeleaf является популярным шаблонизатором для языка программирования Java, который используется для создания динамических веб-приложений. Он предоставляет возможность встраивать HTML шаблоны в код Java, что делает генерацию контента более удобной и эффективной. Thymeleaf поддерживает множество функций, включая рендеринг шаблонов, работу с формами, условия, циклы и многое другое. Он интегрируется хорошо с фреймворками Java, такими как Spring Framework, что делает его популярным выбором для разработчиков, создающих веб-приложения на Java. Основные преимущества Thymeleaf:

1. Простота использования: Thymeleaf предлагает интуитивный синтаксис, который легко читать и понимать, что облегчает разработку и поддержку веб-приложений.

2. Возможность встраивания: Thymeleaf позволяет встраивать динамические данные в HTML шаблоны, делая процесс рендеринга страниц более гибким и удобным.

3. Интеграция с Java-фреймворками: Thymeleaf хорошо интегрируется с различными Java-фреймворками, что облегчает использование его в проектах на Java.

4. Поддержка международизации: Thymeleaf поддерживает интернационализацию и локализацию, что позволяет легко создавать мультиязычные веб-приложения. Thymeleaf является мощным инструментом для разработки динамических веб-приложений на Java, обеспечивая удобство использования, гибкость и хорошую интеграцию с другими технологиями.

Сайт был создан для предоставления информации о музеях и их дополнительной информации.

Сайт состоит из главной страницы со списком музеев, так же есть кнопка «Выбрать» для просмотра информации о данном музее. На странице просмотра есть кнопка «Назад», чтобы откатиться назад на главную страницу.

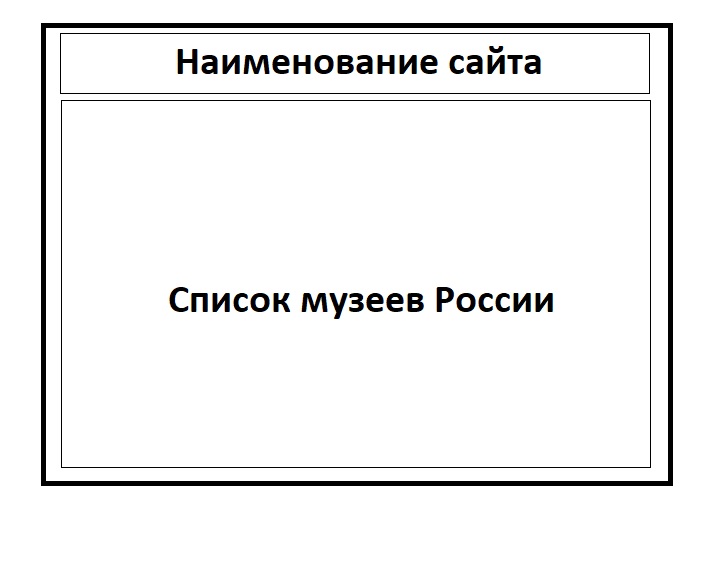


Рисунок 1 – Главная страница сайта

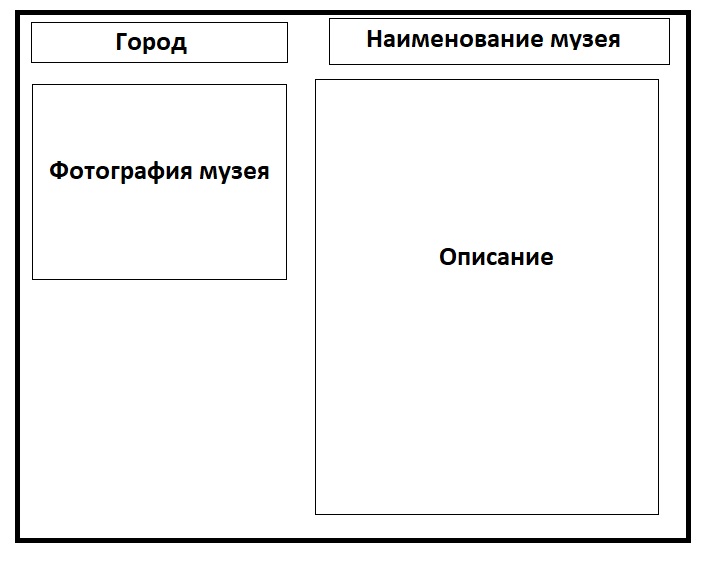


Рисунок 2 – Страница о музее

ЗАКЛЮЧЕНИЕ

В практической работе разработан план сайт «Музеи городов России», страница которого содержат в себе следующие функции:

- страница со списком музеев;

- блоки с информацией о выбранном музее;

# **Практическая работа №2. Разработка программного продукта**

## ***База данных***

В качестве базы данных выбрал MySQL. Ранее созданный бэкап базы данных, которого создали для этого проекта, который, находится в архиве «museums\_ruDB.zip» нужно распаковать.

1. Добавление базы данных в проект. Установка MySQL:

* Скачать базу данных по ссылке <https://dev.mysql.com/downloads/installer/>
* установить базу данных. При первом запуске выберите Developer;
* главное запомните логин и пароль когда будете их создавать;
* если собираетесь установить БД на отдельном компьютере (не на том, где запускаете проект), то логин которого вы создали, должен иметь права для работы с сетью.

1. Создание базы данных и присоединение бэкапа базы к существующей:

- открываем cmd от имени администратора;

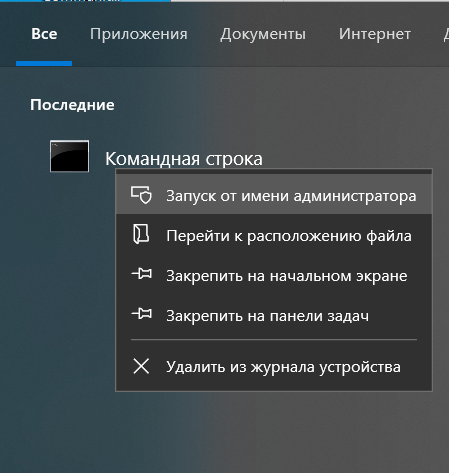


Рисунок 3 – Командная строка

- с помощью команды cd открываем каталог bin сервера MySQL на сервере. По умолчанию он находится C:\Program Files\MySQL\MySQL Server 8.0\bin;

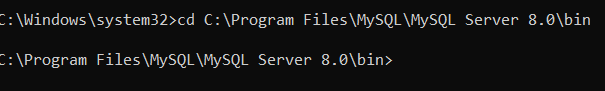


Рисунок 4 – Команда cd

* подключаемся к серверу mysql и создаем нашу базу, в которую потом будем вливать наш бэкап бд. Например «test\_info\_1», но в нашем случае «museums\_ru»

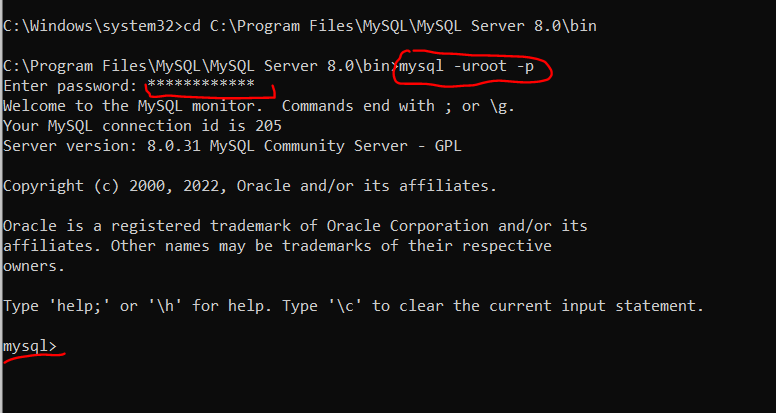


Рисунок 5 – Подключение к базе данных

- пишем команду mysql –uroot –p и вводим пароль. Логин (например root) и пароль вы пишете тот который создали при установке сервера mysql.

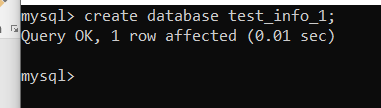


Рисунок 6 – Создание базы данных

Создали базу под именем «test\_info\_1», а в нашем случае «museums\_ru». То есть наша команда будет: “create database museums\_ru;”

- командой exit выходим из режима. Заливаем бэкап в нашу вновь созданную базу. Например бэкап лежит на диске C: поэтому команда будет следующая:

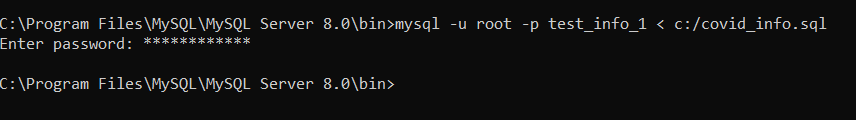


Рисунок 7 – Восстановление БД из бэкапа

Mysql-u root –p “имя базы которую создали” < расположение резервной копии

Далее вводим пароль. Внимание, вместо covid\_info.sql у нас должен быть museums\_ruDB.sql и он должен лежать в корневой диска С.

Код должен выглядеть вот так - “mysql -u root -p museums\_ru < c:/ museums\_ruDB.sql”

1. Заходим в workbench и убеждаемся, что база создана.

## ***Рабочая среда программирования***

В состав рабочей среды входят следующие компоненты:

* Spring Web для создания web-приложений, в том числе RESTful, с использованием Spring MVC (Model-View-Controller). Использует сервер Apache Tomcat по умолчанию.
* Apache Maven – основной сборщик проекта
* Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (ORM)

К этому моменту, у нас на рабочем компьютере должна быть установлена:

- Intellij Idea – среда разработки ([Install IntelliJ IDEA | IntelliJ IDEA Documentation (jetbrains.com)](https://www.jetbrains.com/help/idea/installation-guide.html?ysclid=lq6ohty2cg190345137));

- Java JDK и JRE 17 версии или выше ([JDK Installation Guide (oracle.com)](https://docs.oracle.com/en/java/javase/17/install/installation-guide.pdf));

- Maven – средство автоматической сборки проектов ([Maven – Installing Apache Maven](https://maven.apache.org/install.html));

- База данных MySQL ([Как установить MySQL на Windows: пошаговая инструкция | Timeweb Cloud](https://timeweb.cloud/tutorials/mysql/kak-ustanovit-mysql-na-windows?ysclid=lq6olc3ue2298782460))

## ***Разработка сайта «Музеи городов России»***

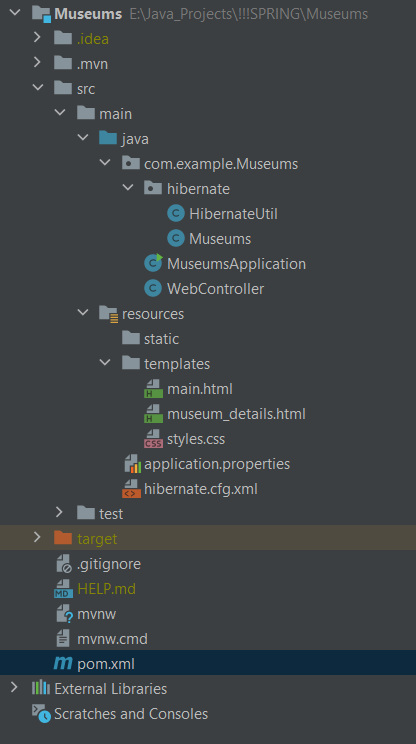


Рисунок 8 – Иерархия классов и файлов веб – приложения в Intellij Idea

Разберем детально файлы, то что есть на рисунке 8.

**pom.xml** – в файле прописаны настройки зависимостей, то есть библиотеки для разработки приложения подтягиваются автоматически с помощью maven

<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.example</groupId>  
 <artifactId>Museums</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Museums</name>  
 <description>Project for student</description>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.32</version>  
 </dependency>  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>6.1.7.Final</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
</project>

**hibernate.cfg.xml –** в файле прописываем настройки для подключения к базе данных. Указываем IP – адрес хоста, на котором установлен MySQL, и соответственно логин и пароль подключения к нему.

<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
 <property name="hibernate.connection.url">jdbc:mysql://192.168.1.38:3306/museums\_ru</property>  
 <property name="hibernate.connection.username">rewoly</property>  
 <property name="hibernate.connection.password">Marshmellow89</property>  
 <property name="hibernate.show\_sql">true</property>  
 <property name="hibernate.hbm2ddl.auto">update</property>  
 <mapping class="com.example.Museums.hibernate.Museums"/>  
 </session-factory>  
</hibernate-configuration>

**MuseumsApplication** – Класс для запуска web – приложения

package com.example.Museums;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class MuseumsApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(MuseumsApplication.class, args);  
 }  
  
}

**HibernateUtil** – Класс адаптер для работы с базой данных

package com.example.Museums.hibernate;  
  
import org.hibernate.SessionFactory;  
import org.hibernate.boot.Metadata;  
import org.hibernate.boot.MetadataSources;  
import org.hibernate.boot.registry.StandardServiceRegistry;  
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;  
  
*/\*\*  
 \* Класс для сессий с БД  
 \*/*public class HibernateUtil {  
 private static StandardServiceRegistry *registry*;  
 private static SessionFactory *sessionFactory*;  
  
 */\*\*  
 \* Метод предназначен для работы с базой данных  
 \** ***@return*** *\*/* public static SessionFactory getSessionFactory() {  
 if (*sessionFactory* == null) {  
 try {  
 // Create registry  
 *registry* = new StandardServiceRegistryBuilder().configure().build();  
  
 // Create MetadataSources  
 MetadataSources sources = new MetadataSources(*registry*);  
  
 // Create Metadata  
 Metadata metadata = sources.getMetadataBuilder().build();  
  
 // Create SessionFactory  
 *sessionFactory* = metadata.getSessionFactoryBuilder().build();  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 if (*registry* != null) {  
 StandardServiceRegistryBuilder.*destroy*(*registry*);  
 }  
 }  
 }  
 return *sessionFactory*;  
 }  
  
 public static void shutdown() {  
 if (*registry* != null) {  
 StandardServiceRegistryBuilder.*destroy*(*registry*);  
 }  
 }  
}

**Museums –** Класс сущности для управления таблицей в БД

package com.example.Museums.hibernate;  
  
import jakarta.persistence.\*;  
  
*/\*\*  
 \* Класс сущности, нужен для соединения с базой данных  
 \*/*@Entity  
@Table(name = "museums")  
public class Museums {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id")  
 private int id;  
  
 @Column(name = "name")  
 private String name;  
  
 @Column(name = "description")  
 private String description;  
  
 @Lob  
 @Column(name = "photo")  
 private byte[] photo;  
  
 @Column(name = "city")  
 private String city;  
  
 public Museums() {}  
 public Museums(String name, String description, byte[] photo, String city) {  
 this.name = name;  
 this.description = description;  
 this.photo = photo;  
 this.city = city;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public byte[] getPhoto() {  
 return photo;  
 }  
  
 public void setPhoto(byte[] photo) {  
 this.photo = photo;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
}

**WebController –** Класс контроллер для обработки запросов от клиентов.

package com.example.Museums;  
  
import com.example.Museums.hibernate.HibernateUtil;  
import com.example.Museums.hibernate.Museums;  
import org.hibernate.Session;  
import org.hibernate.Transaction;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
  
import java.util.ArrayList;  
import java.util.Base64;  
import java.util.List;  
  
*/\*\*  
 \* Класс контроллер отвечает на запросы через веб - браузеры  
 \*/*@Controller  
public class WebController {  
  
 @GetMapping("/")  
 private String getMainPage(Model model) {  
 List<Museums> allMuseums = getMuseums();  
 model.addAttribute("museums", allMuseums);  
 return "main";  
 }  
  
 @GetMapping("/museum/{id}")  
 private String getMuseumDetails(Model model, @PathVariable("id") int id) {  
 Museums museum = getMuseums().get(id - 1);  
 model.addAttribute("selectedMuseum", museum);  
 byte[] imageData = museum.getPhoto();  
  
 String base64Image = Base64.*getEncoder*().encodeToString(imageData);  
 model.addAttribute("imageData", base64Image);  
 model.addAttribute("imageWidth", "300px");  
 model.addAttribute("imageHeight", "300px");  
  
 return "museum\_details";  
 }  
  
 */\*\*  
 \* Метод возвращает список музеев из БД  
 \*/* public synchronized List<Museums> getMuseums() {  
 Transaction transaction = null;  
 List<Museums> museums = new ArrayList<>();  
 try (Session session = HibernateUtil.*getSessionFactory*().openSession()) {  
 museums = session.createQuery("from Museums", Museums.class).list();  
 } catch (Exception e) {  
 if (transaction != null) {  
 transaction.rollback();  
 }  
 e.printStackTrace();  
 }  
 return museums;  
 }  
  
  
}

**main.html** – Главная страница веб - приложения

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="utf-8"/>  
 <title>Application</title>  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>  
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>  
</head>  
<body>  
<div class="container">  
 <h1>Музеи городов России</h1>  
 <span th:if="${museums.size() == 0}">Список музеев пустой</span>  
 <table th:if="${museums.size() > 0}" class="table table-hover">  
 <thead class="thead-dark">  
 <tr>  
 <td>Id</td>  
 <td>Наименование</td>  
 <td>Город</td>  
 <td></td>  
 <td></td>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="museum : ${museums}">  
 <td th:text="${museum.id}"/>  
 <td th:text="${museum.name}"/>  
 <td th:text="${museum.city}"/>  
 <td><a class="btn btn-secondary" th:href="@{'/museum/' + ${museum.id}}">Выбрать</a></td>  
 </tr>  
 </tbody>  
 </table>  
</div>  
</body>  
</html>

**Museum\_details.html** – блок выбранного музея

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Музей</title>  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">  
</head>  
<body>  
<div class="container">  
 <div class="row">  
 <div class="col-md-4">  
 <h2 th:text="${selectedMuseum.city}"> </h2>  
 <img th:src="'data:image/jpeg;base64,' + ${imageData}" th:style="'width: ' + ${imageWidth} + '; height: ' + ${imageHeight}" alt="Изображение">  
 </div>  
 <div class="col-md-8">  
 <h2 th:text="${selectedMuseum.name}"></h2>  
 <p th:text="${selectedMuseum.description}"></p>  
 <td><a class="btn btn-secondary" th:href="@{'/'}">Назад</a></td>  
 </div>  
 </div>  
</div>  
  
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"></script>  
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>  
</body>  
</html>

ЗАКЛЮЧЕНИЕ

В данной практической работе был разработан сайт «Музеи городов России» К достоинствам разработанного сайта можно отнести:

* простой и понятный интерфейс;
* вся главная информация находится на виду у пользователя;
* все необходимые для использования функции;
* простой код, который можно легко расширить и изменить;
* доступность на слабых компьютерах;
* совместимость со всеми браузерами.

# **Практическая работа №3. Тестирование и отладка программного продукта**

Перед тестированием запускаем сервер.



Рисунок 9 – Запуск сервера

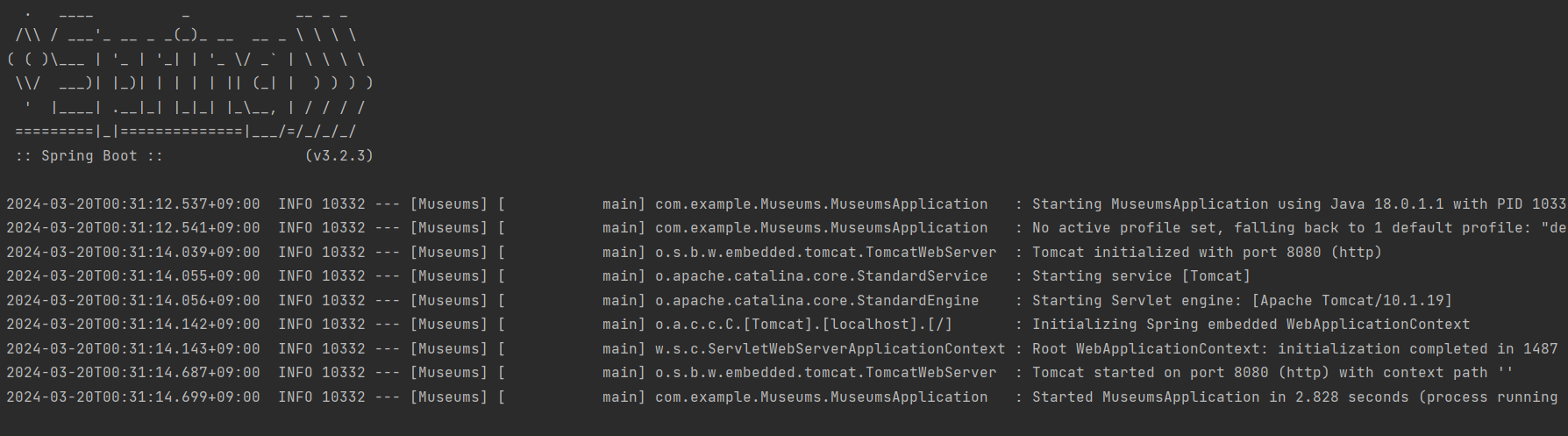


Рисунок 10 – Успешно запущенный сервер

При проведении тестирования сайта особенно важно уделить внимание каждой странице, чтобы гарантировать полное и безошибочное функционирование всех элементов. Важно убедиться, что все ссылки, кнопки, формы, изображения и другие элементы работают правильно и отображаются корректно как на десктопных, так и мобильных устройствах.

1. Главная страница: при тестировании главной страницы необходимо убедиться, что:

- Шапка и футер сайта отображаются корректно на всех устройствах. - Главное меню работает без ошибок и все ссылки ведут на соответствующие страницы.

- Карусели и изображения загружаются быстро и отображаются правильно.

- Формы подписки, поиска или обратной связи функционируют корректно.

- Кнопки действий (заказать звонок, оставить отзыв и т. д.) переходят на соответствующие страницы.

2. Страница с музеями: при тестировании страницы с музеями следует проверить:

- Отображение списка музеев с названием, описанием, фотографиями и контактной информацией.

- Фильтр по городу, типу музея или другим критериям работает корректно.

- Проверить работу карты с расположением музеев.

- Функционал добавления отзывов и рейтингов для каждого музея. - Возможность сортировки и поиска музеев по различным параметрам.

ЗАКЛЮЧЕНИЕ

В итоге, тестирование сайта на различных страницах поможет убедиться в качестве работы всех функций и обеспечить пользователей приятным и безошибочным опытом использования сайта "Музеи городов России".

# **Практическая работа №4. Разработка руководства пользователя**

# **Список использованных источников**