|  |  |
| --- | --- |
| C:\Users\markf\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\8C1AD468.tmp | Министерство образования и науки Российской Федерации  Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина» (УрФУ)  ИРИТ-РТФ  Базовая кафедра «Аналитика больших данных и методы видео анализа» |

Оценка по проекту \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель проекта \_\_Медведев M.A.\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет по проекту на тему

«Твой ЕКБ»

Студент:  \_\_\_\_\_Потапова М.В.\_\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО) (Подпись)

Студент:  \_\_\_\_\_Латников В.Н.\_\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО) (Подпись)

Группа #: \_\_\_\_\_РИ-311055\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Екатеринбург

2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc168936389)

[АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc168936390)

[ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 6](#_Toc168936391)

[ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ РАЗРАБОТКИ 7](#_Toc168936392)

[РАЗРАБОТКА ПРОТОТИПОВ И ДИЗАЙН-ШАБЛОНОВ 9](#_Toc168936393)

[РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ 13](#_Toc168936394)

[РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ 30](#_Toc168936395)

[ЗАКЛЮЧЕНИЕ 34](#_Toc168936396)

# ВВЕДЕНИЕ

Актуальностью данной работы являются современные технологии, которые позволяют создавать приложения для облегчения поиска и обмена информацией о различных местах и событиях в городе. В Екатеринбурге, как в одном из крупнейших городов России, существует множество интересных мест, о которых полезно знать как жителям, так и туристам. Веб-приложение «Твой ЕКБ» позволит пользователям быстро находить информацию и отзывы о популярных местах города.

Целью данной работы является разработка веб-приложения, которое позволит пользователям просматривать информацию и оставлять отзывы о различных интересных местах в Екатеринбурге.

Задачи данной работы:

* Провести анализ предметной области и определить требования к системе;
* Спроектировать архитектуру приложения;
* Выбрать инструменты и технологии для разработки;
* Разработать прототипы и дизайн-шаблоны;
* Реализовать клиентскую часть приложения;
* Реализовать серверную часть приложения;
* Провести тестирование и отладку;
* Подготовить отчет и презентацию проекта.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Проект представляет собой веб-приложение, предназначенное для просмотра и оставления отзывов о различных интересных местах в Екатеринбурге, таких как торговые центры, парки и другие достопримечательности. Основная цель приложения - предоставить полезную информацию как жителям города, так и туристам, помогая им выбрать лучшее место для посещения.

Приложение ориентировано на сферу досуга и туризма, предоставляя пользователям возможность обмениваться опытом и мнениями о посещенных местах.

Основными предметами, с которыми работает приложение, являются интересные места Екатеринбурга. Это могут быть достопримечательности, парки, торговые центры и другие популярные локации города.

В системе выделяются три главных объекта:

* Пользователи: Люди, которые используют приложение для поиска информации о местах и оставления отзывов.
* Места: Локации в Екатеринбурге, о которых можно найти информацию и оставить отзывы в приложении.
* Отзывы: Комментарии, которые пользователи оставляют о посещенных местах.

Основные процессы, реализованные в приложении, включают:

* Добавление информации о местах: Администраторы могут добавлять новые места в файл приложения (App.js), указав их название, описание, адрес и изображение.
* Просмотр информации о местах: Пользователи могут просматривать информацию о местах, представленную в виде карточек на главной странице.
* Добавление отзывов: Пользователи могут оставлять отзывы о местах, оценивая их и делясь своими впечатлениями, а Администраторы, в свою очередь, могут добавлять отзывы в базу данных, которая автоматически обновляет данные на сайте.
* Просмотр отзывов: Пользователи могут просматривать отзывы других пользователей о местах, что помогает им составить мнение о том или ином месте.

Взаимодействия:

* Пользователи взаимодействуют с системой через веб-интерфейс, отправляя запросы на просмотр информации о местах и оставление отзывов, также могут обратиться к форме обратной связи.
* Серверная часть взаимодействует с базой данных для получения данных и возвращает результаты БД.
* База данных хранит всю информацию о отзывах, обеспечивая быстрый и надежный доступ к этим данным.

# ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

Блок схема архитектуры проекта представлена на Рисунке 1.

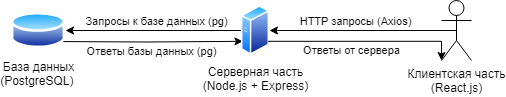


Рисунок 1 − Блок схема архитектуры проекта

Для пояснения схемы выше распишем каждый элемент:

1. Клиентская часть:

* Отправляет HTTP запросы на серверную часть с использованием библиотеки Axios.
* Получает ответы от сервера и отображает данные пользователю.

1. Серверная часть:

* Обрабатывает HTTP запросы, полученные от клиентской части.
* Выполняет запросы к базе данных PostgreSQL с использованием библиотеки pg.
* Возвращает данные в ответ на запросы клиентской части.

1. База данных:

* PostgreSQL используется для хранения информации о местах и отзывах.
* Отвечает на запросы от серверной части и возвращает запрашиваемые данные.

Вся система взаимодействует как двусторонний поток данных:

* Клиентская часть (React.js) отправляет запросы на сервер (Node.js + Express) и получает ответы.
* Серверная часть обрабатывает запросы от клиентской части, взаимодействует с базой данных (PostgreSQL) и возвращает ответы клиентской части.
* База данных принимает запросы от серверной части и возвращает соответствующие данные.

# ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ РАЗРАБОТКИ

Для реализации данного проекта были выбраны следующие технологии:

1. Клиентская часть:
2. React.js − это библиотека JavaScript, используемая для создания пользовательских интерфейсов. Она позволяет создавать компоненты, которые можно повторно использовать в разных частях приложения. React.js позволяет быстро и эффективно разрабатывать сложные интерфейсы благодаря своей компонентной архитектуре и виртуальному DOM.
3. Axios − это библиотека для выполнения HTTP-запросов на основе промисов. Она используется для связи клиентской части с серверной. Axios предоставляет простой и мощный API для выполнения запросов к серверу, обработки ответов и обработки ошибок.
4. CSS и animate.css: CSS (Cascading Style Sheets) − это язык стилей, используемый для описания внешнего вида и форматирования документов, написанных на HTML. Animate.css − это библиотека CSS-анимаций. CSS используется для стилизации приложения, а animate.css позволяет легко добавлять анимационные эффекты для улучшения пользовательского опыта.
5. Серверная часть:
6. Node.js − это среда выполнения JavaScript на сервере, которая позволяет запускать JavaScript-код вне браузера. Node.js позволяет создавать высокопроизводительные и масштабируемые серверные приложения благодаря своей асинхронной, неблокирующей архитектуре.
7. Express.js − это минималистичный и гибкий веб-фреймворк для Node.js, который обеспечивает надежный набор функций для веб-приложений и API. Express.js упрощает создание серверных приложений и RESTful API, предлагая мощные возможности для маршрутизации и обработки запросов.
8. База данных:
9. PostgreSQL − это мощная, объектно-реляционная система управления базами данных с открытым исходным кодом, которая поддерживает расширения и стандарты SQL. PostgreSQL предлагает высокий уровень надежности, масштабируемости и производительности, а также богатый набор функций для управления данными.
10. pgAdmin − это графический инструмент для управления базами данных PostgreSQL, который предоставляет удобный интерфейс для выполнения SQL-запросов, управления таблицами и других административных задач. pgAdmin облегчает управление и мониторинг базы данных, предоставляя интуитивно понятный интерфейс для администрирования.

# РАЗРАБОТКА ПРОТОТИПОВ И ДИЗАЙН-ШАБЛОНОВ

Прототипы проекта представлены на Рисунках 2, 3.

Простая схема расположения элементов на главной странице (Рисунок 2): навигационная панель, слайдер с карточками мест, формы для добавления отзывов и обратной связи.

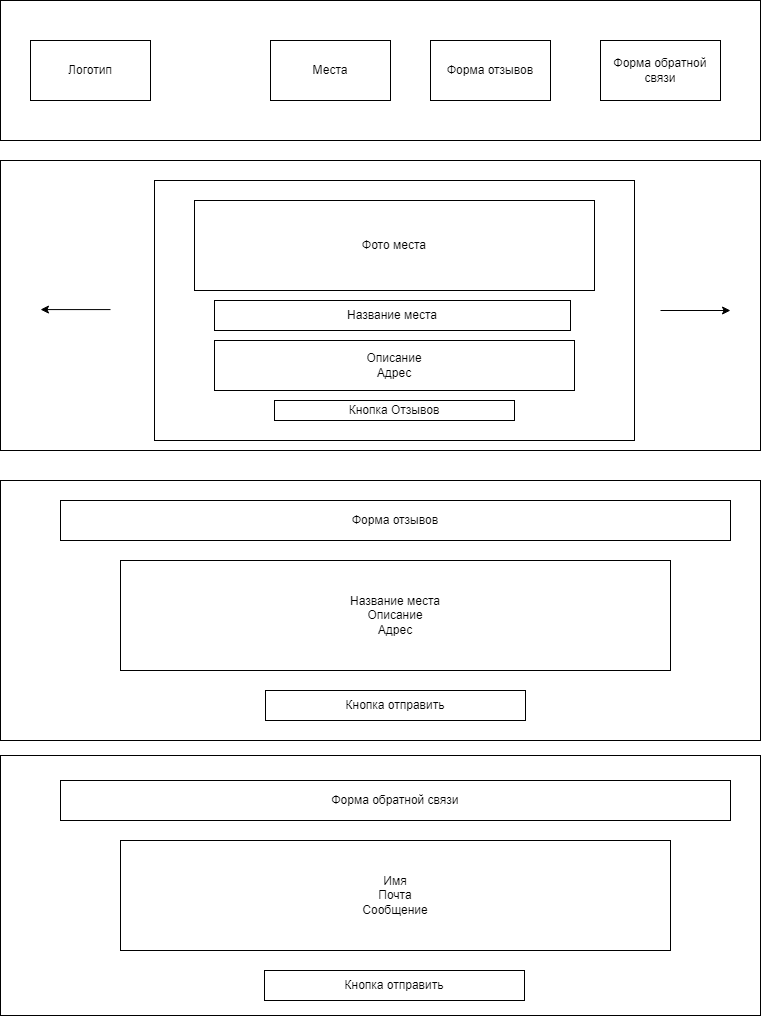


Рисунок 2 − Главная страница

Простая схема расположения элементов на странице отзывов (Рисунок 3): навигационная панель, отображение отзывов.

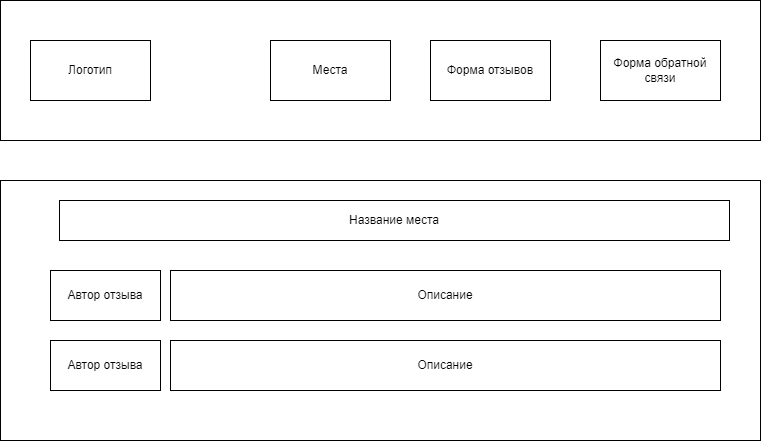


Рисунок 3 − Страница отзывов

Дизайн-шаблоны проекта представлены на

Блок навигационной панели используется как на Главной странице, так и на Странице с отзывами (Рисунок 4).

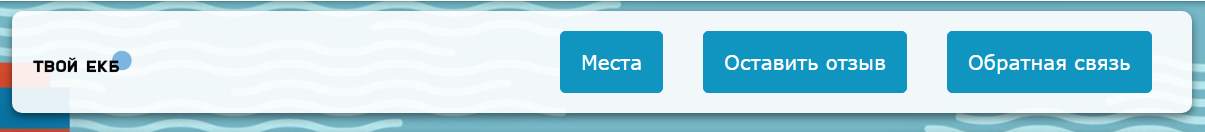


Рисунок − Блок навигационной панели

Блок футера сайта используется как на Главной странице, так и на Странице с отзывами (Рисунок 5).

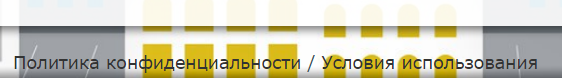


Рисунок − Блок футера сайта

На Главной странице представлены следующие блоки на Рисунках 6, 7, 8.

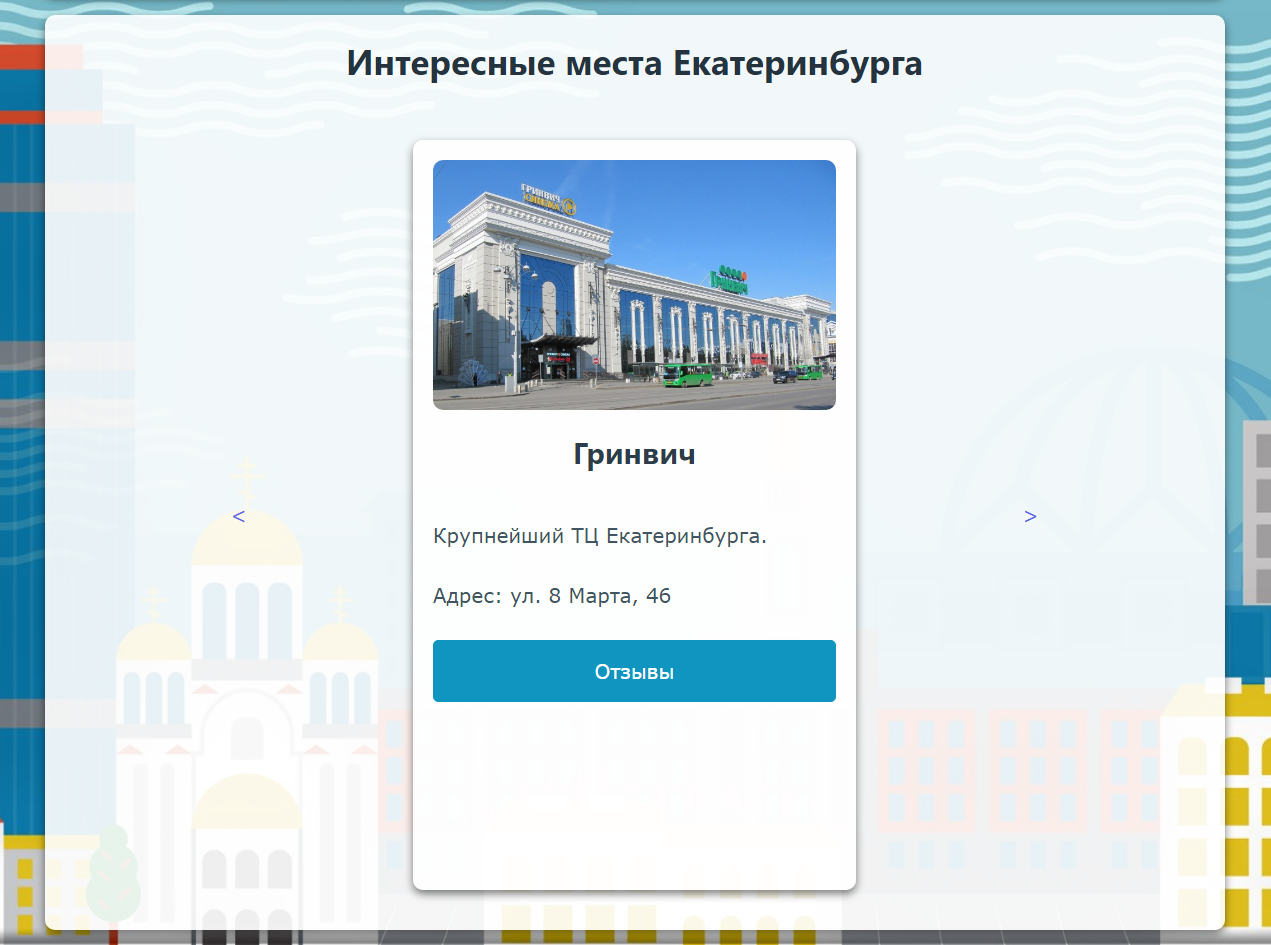


Рисунок − Блок слайдера с карточками мест

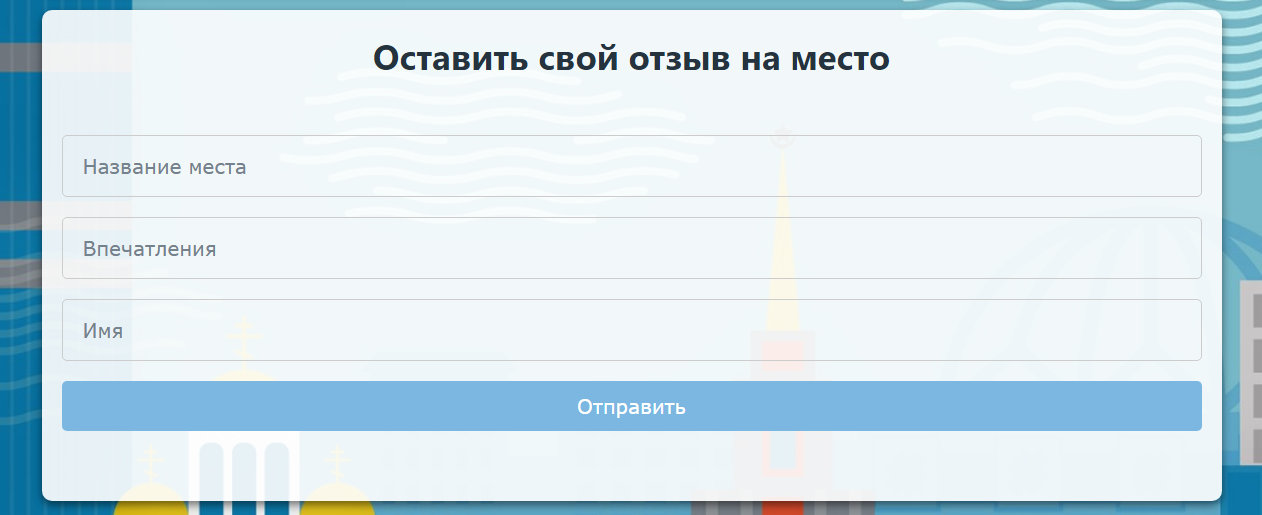


Рисунок − Блок формы отправки отзывов

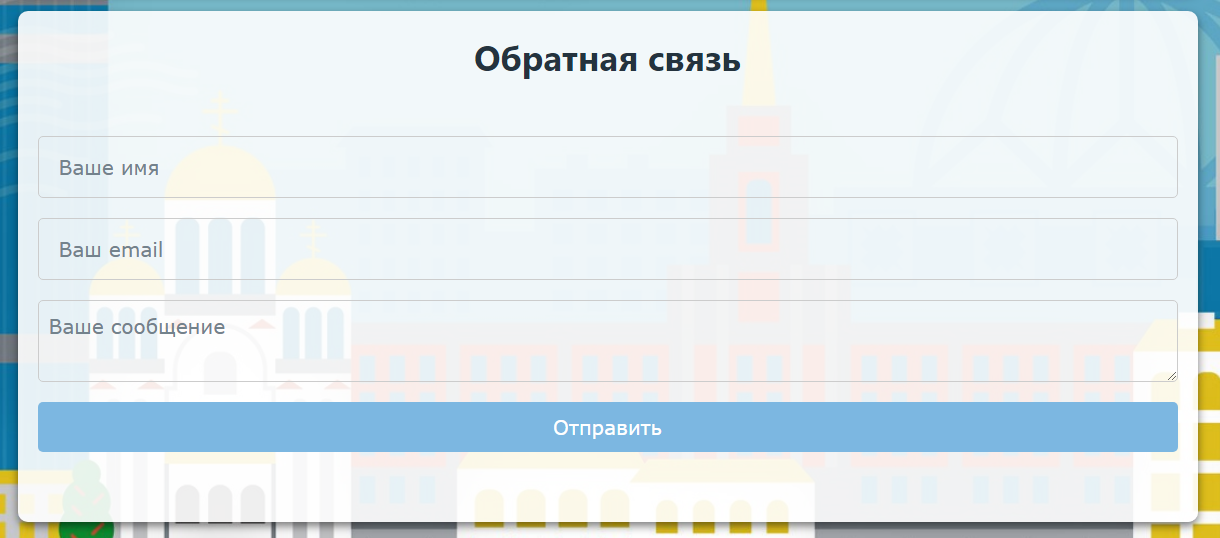


Рисунок − Блок формы обратной связи

На Странице с отзывами представлен следующий блок, кроме навигационной панели и футера сайта, представленных выше, на Рисунке 9.

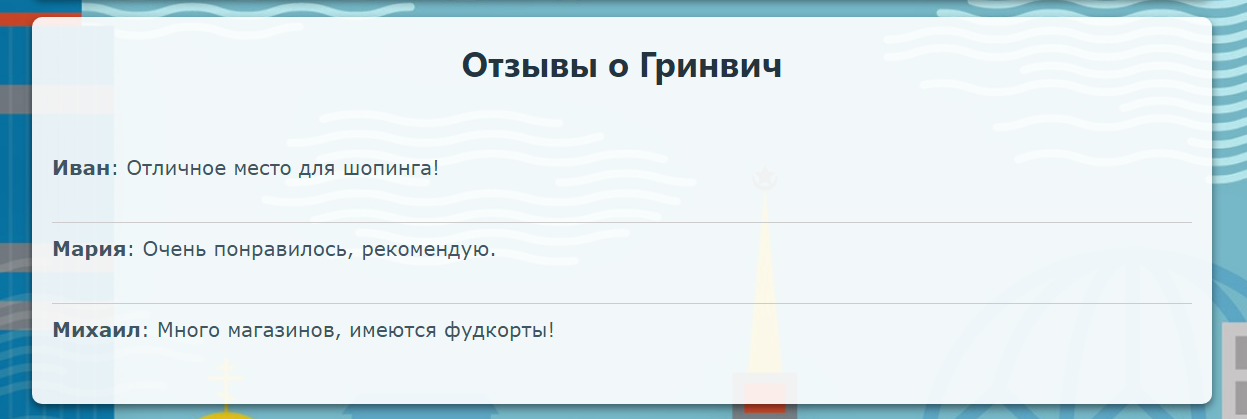


Рисунок − Блок отображения отзывов

# РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ

Описание разработки интерфейса:

* 1. Навигационная панель

**Основной код блока и его стиль**

* Header: Шапка страницы, содержащая навигационную панель и логотип. Она имеет белый полупрозрачный фон, отступы и закругленные углы для улучшения внешнего вида.
* Navbar: Панель навигации, реализованная с помощью flexbox, которая обеспечивает расположение логотипа и навигационных ссылок на одной линии.
* Nav-links: Список навигационных ссылок. Ссылки размещены горизонтально, каждая ссылка имеет внутренние и внешние отступы для создания пространства между ними.
* Logo: Логотип, который имеет фиксированную ширину в 100px и анимацию animate\_\_heartBeat при загрузке страницы.
* handleLogoClick: Обрабатывает клик по логотипу. Перенаправляет пользователя на главную страницу.

*Код:*

**App.js**

<header className="header">

<div className="navbar">

<img src="/images/logo.png" alt="Логотип" className="logo animate\_\_heartBeat" onClick={handleLogoClick} />

<ul className="nav-links">

<li><a href="#places" role="button">Места</a></li>

<li><a href="#suggest" role="button">Оставить отзыв</a></li>

<li><a href="#feedback" role="button">Обратная связь</a></li>

</ul>

</div>

</header>

const handleLogoClick = () => {

window.location.href = '/';

};

**App.css**

<!-- Используется для всего сайта, повторяться в последующих описаниях интерфейса НЕ будет -->

html {

background-image: url('../public/images/background.jpg');

background-size: cover;

background-attachment: fixed;

font-family: Verdana, Geneva, Tahoma, sans-serif;

}

<!-- Используется класс container для Главной страницы, повторяться в последующих описаниях интерфейса НЕ будет -->

.container {

max-width: 1200px;

margin: 0 auto;

padding: 10px;

}

.header{

background-color: rgba(255, 255, 255, 0.9);

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);

margin-bottom: 20px;

}

.navbar {

display: flex;

justify-content: space-between;

align-items: center;

}

.nav-links {

list-style-type: none;

margin: 0;

padding: 0;

}

.nav-links li {

display: inline;

margin-right: 30px;

margin: 20px;

}

.nav-links li a {

text-decoration: none;

color: #ffffff;

}

.logo {

width: 100px;

}

**Реализация адаптивности блока**

Для обеспечения адаптивности, используются медиа-запросы. Они позволяют изменять стили в зависимости от ширины экрана, что делает интерфейс удобным для использования на устройствах с разными размерами экрана.

1. @media screen and (max-width: 1200px):

* Контейнер .container получает максимальную ширину 90% для уменьшения пустого пространства на больших экранах.

1. @media screen and (max-width: 576px):

* Шапка .header изменяет направление расположения flex-элементов на колонку и выравнивает элементы по ширине, чтобы они располагались друг под другом.
* Навигационные ссылки .nav-links получают дополнительный верхний отступ, чтобы не смещать контент.
* Логотип .logo-container получает нижний отступ для лучшего отображения.
* Ширина элементов .nav-links и .logo-container устанавливается на 100%, чтобы они занимали всю ширину экрана.

*Код:*

**App.css**

@media screen and (max-width: 1200px) {

.container {

max-width: 90%;

}

}

@media screen and (max-width: 576px) {

.header {

flex-direction: column;

align-items: stretch;

}

.nav-links {

margin-top: 10px;

}

.logo-container {

margin-bottom: 10px;

}

.nav-links,

.logo-container {

width: 100%;

}

}

* 1. Слайдер с карточками мест

**Основной код блока и его стиль**

* Routes: Компонент, который определяет коллекцию маршрутов.
* Route path="/": Маршрут, соответствующий корневому каталогу приложения (URL-адрес без дополнительных путей).
* element={...}: Определяет, какой компонент будет отображаться для данного маршрута. В данном случае отображаются компоненты секции "Интересные места Екатеринбурга".
* section: Этот блок представляет собой секцию, которая содержит информацию о интересных местах Екатеринбурга. Секция имеет заголовок и карусель карточек.
* h2: Заголовок секции, выровненный по центру для лучшего визуального восприятия.
* Card-slider: Контейнер, в котором расположены карточки с информацией о местах и кнопки для прокрутки.
* Card: Компонент, который отображает информацию о месте. Каждая карточка содержит изображение и текст с описанием места. Карточки оформлены с отступами, тенями и закругленными углами для создания приятного интерфейса.
* Card Image: Изображение места, которое адаптируется под размеры карточки, сохраняя свои пропорции.
* Hover Effect: При наведении курсора на карточку, она увеличивается, создавая эффект зуммирования.
* ArrowButton: Кнопки для прокрутки карточек влево и вправо. Кнопки стилизованы минималистично, чтобы не отвлекать внимание от основного контента.
* /reviews/:placeTitle: это дополнительный маршрут, который используется для отображения страницы с отзывами о конкретном месте. Он принимает параметр "placeTitle", который представляет собой название места.
* Reviews: это компонент, отображающий страницу с отзывами.
* places: массив объектов, каждый из которых представляет собой информацию о конкретном месте, включая:
* title: название места;
* description: описание места;
* address: адрес места;
* image: путь к изображению места.
* handlePrevClick:
* Функция изменяет текущее значение индекса на предыдущий.
* Если текущий индекс равен 0, то функция устанавливает индекс последнего элемента в массиве places. Это позволяет зациклить навигацию.
* Если текущий индекс не равен 0, то он уменьшается на единицу.
* handleNextClick:
* Функция изменяет текущее значение индекса на следующий.
* Если текущий индекс равен индексу последнего элемента в массиве places, то функция устанавливает индекс на 0. Это также позволяет зациклить навигацию.
* Если текущий индекс не равен индексу последнего элемента, то он увеличивается на единицу.

*Код:*

**App.js**

<!-- Используется Routes для Главной страницы, повторяться в последующих описаниях интерфейса НЕ будет -->

<Routes>

<Route path="/" element={

<>

<section className="section" id="places">

<h2>Интересные места Екатеринбурга</h2>

<div className="card-slider">

<ArrowButton direction="left" onClick={handlePrevClick} />

<Card place={places[currentIndex]} />

<ArrowButton direction="right" onClick={handleNextClick} />

</div>

</section>

<Route path="/reviews/:placeTitle" element={<Reviews />} />

</Routes>

const places = [

{

title: 'Гринвич',

description: 'Крупнейший ТЦ Екатеринбурга.',

address: 'ул. 8 Марта, 46',

image: '/images/place1.jpg'

},

{

title: 'ЦПКиО им. В.В. Маяковского ',

description: 'Один из самых популярных парков в Екатеринбурге',

address: 'ул. Мичурина, 230',

image: '/images/place2.jpg'

},

{

title: 'Гостиница Высоцкий',

description: 'Красивый вид, открывается весь Екатеринбург',

address: 'ул. Малышева, 51',

image: '/images/place3.jpg'

}

];

const handlePrevClick = () => {

setCurrentIndex((prevIndex) => (prevIndex === 0 ? places.length - 1 : prevIndex - 1));

};

const handleNextClick = () => {

setCurrentIndex((prevIndex) => (prevIndex === places.length - 1 ? 0 : prevIndex + 1));

};

**App.css**

<!-- Используется заголовок h2 для Главной страницы, повторяться в последующих описаниях интерфейса НЕ будет -->

h2 {

text-align: center;

}

.card {

display: flex;

flex-direction: column;

margin-bottom: 20px;

padding: 20px;

background-color: rgba(255, 255, 255, 0.9);

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);

height: 750px;

overflow: hidden;

transition: transform 0.3s ease-in-out;

}

.card img {

border-radius: 10px;

margin-bottom: 20px;

max-width: 100%;

height: auto;

}

.card-content {

flex: 1;

display: flex;

flex-direction: column;

justify-content: space-between;

}

.card:hover {

transform: scale(1.05);

}

.arrow-button {

background: none;

border: none;

font-size: 20px;

color: #5162e5;

cursor: pointer;

}

.arrow-button.left {

margin-right: 10px;

}

.arrow-button.right {

margin-left: 10px;

}

.card-slider {

display: flex;

align-items: center;

justify-content: center;

}

**Описание компонента Card**

* Card: Этот компонент отображает информацию о конкретном месте.
* useNavigate: Используется для навигации к странице с отзывами о месте.
* handleReviewsClick: Функция, которая вызывается при нажатии на кнопку "Отзывы" и перенаправляет пользователя на страницу с отзывами.
* Card Layout: Карточка включает изображение, заголовок, описание, адрес и кнопку для перехода на страницу отзывов.

*Код:*

**Card.js**

import React from 'react';

import { useNavigate } from 'react-router-dom';

function Card({ place }) {

const navigate = useNavigate();

const handleReviewsClick = () => {

navigate(`/reviews/${place.title}`);

};

return (

<div className="card">

<img src={place.image} alt={place.title} />

<div className="card-content">

<h3>{place.title}</h3>

<p>{place.description}</p>

<p>Адрес: {place.address}</p>

<button onClick={handleReviewsClick}>Отзывы</button>

</div>

</div>

);

}

export default Card;

**Описание компонента ArrowButton**

* ArrowButton: Кнопка для прокрутки карточек влево и вправо.
* direction: Определяет направление стрелки, которая отображается на кнопке ('left' для '<' и 'right' для '>').
* onClick: Обработчик клика, который вызывается при нажатии на кнопку.

*Код:*

**ArrowButton.js**

import React from 'react';

function ArrowButton({ direction, onClick }) {

return (

<button className={`arrow-button ${direction}`} onClick={onClick}>

{direction === 'left' ? '<' : '>'}

</button>

);

}

export default ArrowButton;

**Реализация адаптивности блока**

Для обеспечения корректного отображения на различных устройствах используются медиа-запросы:

1. @media screen and (max-width: 992px):

* Уменьшаются внутренние отступы у элементов с классом .card до 15px, чтобы карточки стали более компактными на устройствах средней ширины.

1. @media screen and (max-width: 768px):

* Внутренние отступы у .card уменьшаются до 10px, что позволяет сэкономить место на экранах меньшей ширины.

1. @media screen and (max-width: 576px):

* Карточки .card получают автоматическую высоту, чтобы адаптироваться под размер экрана.
* Изображения в .card img адаптируются по ширине и высоте для лучшего отображения на мобильных устройствах, убирается максимальная высота.

*Код:*

**App.css**

/\* Media Queries for Responsiveness \*/

@media screen and (max-width: 992px) {

.card {

padding: 15px;

}

}

@media screen and (max-width: 768px) {

.card {

padding: 10px;

}

}

@media screen and (max-width: 576px) {

.card {

padding: 10px;

height: auto;

}

.card img {

width: 100%;

height: auto;

max-height: none;

}

}

* 1. Форма Отзывов на место

**Основной код блока и его стиль**

* section: Этот элемент создаёт секцию страницы для формы отзыва.
* Идентификатор suggest позволяет ссылаться на эту секцию в навигации и стилизации.
* form: Форма для ввода отзыва. Обработчик handleNewPlaceSubmit вызывается при отправке формы.
* input: Поле ввода для названия места. Атрибуты name, placeholder, и required обеспечивают корректную работу формы и удобство для пользователя. value и onChange связаны с состоянием компонента для управления вводом данных. Также используется для описания впечатлений и имени пользователя.
* button: Кнопка для отправки формы. Тип submit инициирует отправку формы и вызывает обработчик handleNewPlaceSubmit.
* form: Устанавливает, что элементы формы располагаются вертикально друг под другом.
* form input, form textarea: Задаёт отступы, внутренние отступы, границы и радиус границ для полей ввода и текстовой области. Отступы снизу обеспечивают визуальное разделение между элементами.
* form button: Настраивает внешний вид кнопки отправки, включая отступы, отсутствие границы, радиус границы, цвет фона, цвет текста и указатель курсора. Переход background-color задаёт плавное изменение цвета фона при наведении курсора.
* form button:hover: Определяет изменение цвета фона кнопки при наведении курсора, улучшая пользовательский интерфейс.
* handleNewPlaceChange: Обработчик изменения значения полей ввода. Обновляет состояние newPlaceData при каждом изменении поля ввода.
* handleNewPlaceSubmit: Обработчик отправки формы. Предотвращает стандартное поведение формы (перезагрузка страницы), выводит данные, введенные пользователем, а затем сбрасывает состояние newPlaceData до первоначального состояния.
* e.preventDefault(): Предотвращает стандартное поведение формы (перезагрузка страницы).
* axios.post: Асинхронный запрос к серверу для отправки данных формы (newPlaceData) на указанный URL (http://localhost:5000/api/reviews).
* console.log(response.data): Выводит ответ сервера в консоль для отладки.
* alert: Показывает пользователю сообщение об успешной отправке отзыва, включая введенные данные.
* catch (error): Обрабатывает ошибки, возникающие при отправке данных на сервер, выводит ошибку в консоль и показывает сообщение об ошибке пользователю.
* setNewPlaceData: Сбрасывает состояние newPlaceData до первоначального состояния после успешной отправки формы или при возникновении ошибки.

*Код:*

**App.js**

<section className="section" id="suggest">

<h2>Оставить свой отзыв на место</h2>

<form onSubmit={handleNewPlaceSubmit}>

<input type="text" name="title" placeholder="Название места" required value={newPlaceData.title} onChange={handleNewPlaceChange} />

<input type="text" name="description" placeholder="Впечатления" required value={newPlaceData.description} onChange={handleNewPlaceChange} />

<input type="text" name="address" placeholder="Имя" required value={newPlaceData.address} onChange={handleNewPlaceChange} />

<button type="submit">Отправить</button>

</form>

</section>

const [newPlaceData, setNewPlaceData] = useState({

title: '',

description: '',

name: ''

});

const handleNewPlaceSubmit = async (e) => {

e.preventDefault();

try {

const response = await axios.post('http://localhost:5000/api/reviews', newPlaceData);

console.log(response.data); // Вывод ответа сервера в консоль

alert(`Отзыв успешно отправлен!\nНазвание места: ${newPlaceData.title}\nВпечатления: ${newPlaceData.description}\nИмя: ${newPlaceData.name}`);

} catch (error) {

console.error('Ошибка при отправке отзыва:', error);

alert('Произошла ошибка при отправке отзыва.');

}

setNewPlaceData({

title: '',

description: '',

name: ''

});

};

const handleNewPlaceChange = (e) => {

const { name, value } = e.target;

setNewPlaceData(prevState => ({

...prevState,

[name]: value

}));

};

**App.css**

<!-- Используется различных стили css для форм на Главной странице, повторяться в последующих описаниях интерфейса НЕ будет -->

form {

display: flex;

flex-direction: column;

}

form input,

form textarea {

margin-bottom: 1rem;

padding: 0.5rem;

border: 1px solid #ccc;

border-radius: 5px;

}

form button {

padding: 0.5rem;

border: none;

border-radius: 5px;

background-color: rgb(124, 183, 225);

color: #fff;

cursor: pointer;

transition: background-color 0.3s;

}

form button:hover {

background-color: #5162e5;

}

* 1. Форма обратной связи

**Основной код блока и его стиль**

* section: Этот элемент создает секцию страницы для формы обратной связи.
* Идентификатор feedback позволяет ссылаться на эту секцию в навигации и стилизации.
* form: Форма для ввода отзыва. Обработчик handleSubmit вызывается при отправке формы.
* input (name): Поле ввода для имени пользователя. Атрибуты name, placeholder, и required обеспечивают корректную работу формы и удобство для пользователя. value и onChange связаны с состоянием компонента для управления вводом данных. Также input используется для ввода сообщения.
* input (email): Поле ввода для email пользователя с аналогичными атрибутами.
* textarea: Поле для ввода сообщения пользователя. Атрибуты аналогичны полям ввода.
* button: Кнопка для отправки формы. Тип submit инициирует отправку формы и вызывает обработчик handleSubmit.
* handleChange: Обработчик изменения значения полей ввода. Обновляет состояние formData при каждом изменении поля ввода.
* handleSubmit: Обработчик отправки формы. Предотвращает стандартное поведение формы (перезагрузка страницы), выводит данные, введенные пользователем, а затем сбрасывает состояние formData до первоначального состояния.

*Код:*

**App.js**

<section className="section" id="feedback">

<h2>Обратная связь</h2>

<form onSubmit={handleSubmit}>

<input type="text" name="name" placeholder="Ваше имя" required value={formData.name} onChange={handleChange} />

<input type="email" name="email" placeholder="Ваш email" required value={formData.email} onChange={handleChange} />

<textarea name="message" placeholder="Ваше сообщение" required value={formData.message} onChange={handleChange}></textarea>

<button type="submit">Отправить</button>

</form>

</section>

const handleSubmit = (e) => {

e.preventDefault();

alert(`Имя: ${formData.name}\nEmail: ${formData.email}\nСообщение: ${formData.message}`);

setFormData({

name: '',

email: '',

message: ''

});

};

const handleChange = (e) => {

const { name, value } = e.target;

setFormData(prevState => ({

...prevState,

[name]: value

}));

};

function App() {

const [currentIndex, setCurrentIndex] = useState(0);

const [formData, setFormData] = useState({

name: '',

email: '',

message: ''

});

* 1. Футер сайта

**Основной код блока и его стиль**

* footer: Этот элемент создает секцию внизу страницы для размещения информации о политике конфиденциальности и условиях использования.
* small: Этот элемент используется для отображения текста меньшего размера.
* a: Ссылки на страницы политики конфиденциальности и условий использования. Атрибут href указывает, что ссылки ведут на соответствующие документы.
* .footer: Класс для футера. Задает выравнивание текста по центру и верхний отступ для создания пространства между футером и предыдущим элементом.
* .footer a: Класс для ссылок в футере. Убирает стандартное подчеркивание текста и задает цвет текста.

*Код:*

**App.js**

<footer className="footer">

<small><a href="#!">Политика конфиденциальности</a> / <a href="#!">Условия использования</a></small>

</footer>

**App.css**

.footer {

text-align: center;

margin-top: 20px;

}

.footer a {

text-decoration: none;

color: #333;

}

* 1. Блок Отзывов

**Основной код блока и его стиль**

**Описание компонента Reviews**

* useParams: Хук из React Router для извлечения параметров из URL. В данном случае получаем placeTitle.
* useState: Хук для управления состоянием компонента.
* useEffect: Хук для выполнения побочных эффектов в функциональном компоненте. Здесь используется для загрузки отзывов при монтировании компонента.
* axios: Библиотека для выполнения HTTP-запросов.
* fetchReviews: Функция для отправки запроса на сервер и обновления состояния отзывов.
* reviews.map: Итерация по массиву отзывов и отображение каждого отзыва в виде блока с автором и текстом отзыва.

*Код:*

**Reviews.js**

import React, { useEffect, useState } from 'react';

import { useParams } from 'react-router-dom';

import axios from 'axios';

function Reviews() {

const { placeTitle } = useParams();

const [reviews, setReviews] = useState([]);

const [loading, setLoading] = useState(true);

const [error, setError] = useState(null);

useEffect(() => {

const fetchReviews = async () => {

try {

const response = await axios.get(`http://localhost:5000/api/reviews/${placeTitle}`);

setReviews(response.data);

setLoading(false);

} catch (err) {

setError(err.message);

setLoading(false);

}

};

fetchReviews();

}, [placeTitle]);

if (loading) return <p>Загрузка...</p>;

if (error) return <p>Ошибка: {error}</p>;

return (

<div className="reviews">

<h2>Отзывы о {placeTitle}</h2>

{reviews.length === 0 ? (

<p>Нет отзывов для этого места.</p>

) : (

reviews.map((review) => (

<div key={review.id} className="review">

<p><strong>{review.author}</strong>: {review.text}</p>

</div>

))

)}

</div>

);

}

export default Reviews;

# РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ

**База данных**

Схема представлена на Рисунке 10.

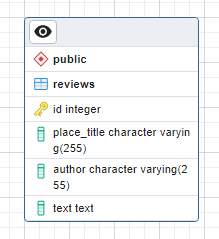


Рисунок 10 − Схема БД

В нашем приложении используется одна таблица в базе данных, которая содержит отзывы пользователей о различных местах. Назовем эту таблицу "reviews".

Структура таблицы "reviews" (Рисунок 11):

* id: Уникальный идентификатор отзыва (тип данных: SERIAL, PRIMARY KEY).
* place\_title: Название места, к которому относится отзыв (тип данных: VARCHAR).
* author: Автор отзыва (тип данных: VARCHAR).
* text: Текст отзыва (тип данных: TEXT).

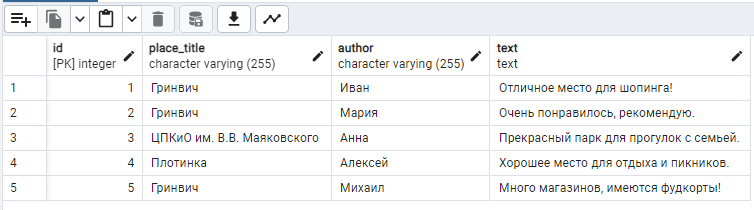


Рисунок 11 − Таблица "reviews"

Описание серверной части приложения, включая основные компоненты и их функциональность:

**db.js**

const { Pool } = require('pg');

const pool = new Pool({

user: '-',

host: '-',

database: 'interesting\_places\_db',

password: '-',

port: 5432,

});

module.exports = pool;

Этот модуль представляет собой настройки и подключение к базе данных PostgreSQL с использованием пакета pg. В файле определен объект pool, который предоставляет пул соединений к базе данных. Этот объект экспортируется из модуля для использования в других частях приложения.

**server.js**

const express = require('express');

const cors = require('cors');

const pool = require('./db');

const app = express();

app.use(cors());

app.use(express.json());

app.get('/api/reviews/:placeTitle', async (req, res) => {

const placeTitle = req.params.placeTitle;

try {

const reviews = await pool.query('SELECT \* FROM reviews WHERE place\_title = $1', [placeTitle]);

res.json(reviews.rows);

} catch (err) {

console.error('Error executing query', err);

res.status(500).send('Server Error');

}

});

app.post('/api/reviews', async (req, res) => {

const { title, description, name } = req.body;

try {

const newReview = await pool.query(

'INSERT INTO reviews (place\_title, text, author) VALUES ($1, $2, $3)',

[title, description, name]

);

res.json(newReview.rows[0]);

} catch (err) {

console.error('Error executing query', err);

res.status(500).send('Server Error');

}

});

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

Этот файл является основным файлом серверной части приложения на Node.js с использованием Express и PostgreSQL. Он управляет HTTP-запросами, обрабатывает данные и взаимодействует с базой данных.

1. Что делает этот файл:

* Создание и настройка сервера:
* Express: Создает серверное приложение.
* CORS: Использует для разрешения кросс-доменных запросов.
* express.json(): Настраивает middleware для автоматического парсинга JSON-тел запросов.

1. Определение маршрутов (routes):

**GET /api/reviews/:placeTitle:**

* Извлекает параметр placeTitle из URL.
* Выполняет SQL-запрос для получения всех отзывов с данным placeTitle.
* Возвращает результаты запроса в формате JSON.

**POST /api/reviews:**

* Извлекает данные из тела запроса (название места, описание и имя автора).
* Выполняет SQL-запрос для вставки нового отзыва в базу данных.
* Возвращает данные о созданном отзыве в формате JSON.

1. Запуск сервера:

* Слушает входящие запросы на определённом порту и выводит сообщение в консоль о запуске сервера.

Что использует этот файл:

* Express: Веб-фреймворк для создания и управления серверными маршрутами и middleware.
* CORS: Middleware для разрешения кросс-доменных запросов.
* PostgreSQL (pool): Объект для управления подключениями к базе данных и выполнения SQL-запросов.
* Middleware:
* express.json(): Парсит JSON-тела запросов.
* cors(): Разрешает кросс-доменные запросы.

# ЗАКЛЮЧЕНИЕ

В ходе работы было разработано веб-приложение "Твой ЕКБ", которое позволяет пользователям просматривать и оставлять отзывы о различных местах в городе. Были выполнены все поставленные задачи, включая анализ предметной области, проектирование архитектуры, выбор инструментальных средств, разработка и тестирование клиентской и серверной частей приложения, а также разработка отчета по проекту. Итоговый результат представляет собой функциональное веб-приложение, доступное для использования.

Приложение готово к использованию и может быть развернуто на хостинге для предоставления информации пользователям. В дальнейшем можно рассмотреть возможность добавления новых функций, таких как система рейтингов, фильтры и поиск по ключевым словам.