ПЕРВОЕ ВЫСШЕЕ ТЕХНИЧЕСКОЕ УЧЕБНОЕ ЗАВЕДЕНИЕ РОССИИ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОРНЫЙ УНИВЕРСИТЕТ ИМПЕРАТРИЦЫ ЕКАТЕРИНЫ II

Кафедра информационных систем и вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине: «Проектирование и сопровождение баз данных»

на тему: «Разработка сайта по продаже железнодорожных билетов»

Выполнил: студент гр. ИАС-20 _____ Плотникова П.С.
(шифр группы) (подпись)

Руководитель: ассистент _____ Киба М. Р.
(подпись)

Заключение

_____
_____
_____
_____
_____
_____
_____
_____
_____

_____ _____ _____
(дата) (оценка) (подпись)

Санкт-Петербург
2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сайт, или веб-сайт — одна или несколько логически связанных между собой веб-страниц; также место расположения контента сервера. Обычно сайт в Интернете представляет собой массив связанных данных, имеющий уникальный адрес и воспринимаемый пользователями как единое целое.

Данная курсовая работа посвящена разработке клиентской части для сайта по продаже железнодорожных билетов. В современном мире онлайн-покупка билетов является удобным и быстрым способом приобретения билетов на поезд, поэтому создание удобного и интуитивно понятного интерфейса для пользователей, позволяющего легко и быстро выбирать желаемые маршруты, типы поездов и билеты является важной и актуальной работой. Пользователи ценят удобство, простоту и эффективность процесса покупки билетов онлайн, поэтому качественный интерфейс играет одну из ключевых ролей в привлечении и удержании клиентов.

В рамках данного проекта будут рассмотрены требования к функционалу клиентской части сайта, также будет рассмотрено ее наполнение, структура и удобство. Помимо этого, будет обеспечено корректное взаимодействие с базой данных и изменение созданных backend-разработчиком файлов. Кроме того, будет представлено описание основных возможностей сайта, уровня подготовки пользователя для работы с ним, а также рекомендации по его использованию.

Целью данной курсовой работы является разработка клиентской части для сайта по продаже железнодорожных билетов с интуитивно понятным и удобным интерфейсом для пользователей, обеспечивающего комфортный процесс выбора и приобретения билетов.

Задачами для достижения цели курсовой работы являются:

1. Разработка структуры клиентской части сайта.
2. Наполнение клиентской части необходимым функционалом.
3. Обеспечение корректного взаимодействия с базой данных.
4. Разработка дизайна и внешнего вида сайта.
5. Разработка руководства пользователя.

Работа состоит из введения, трех разделов, заключения и библиографического списка.

1 ОСНОВЫ КЛИЕНТСКОГО ВЗАИМОДЕЙСТВИЯ

1.1 Цель и преимущества веб-сайтов

Веб-сайт как система электронных документов (файлов данных и кода) может принадлежать частному лицу или организации и быть доступным в компьютерной сети под общим доменным именем и IP-адресом или локально на одном компьютере. Для прямого доступа клиентов к сайтам на серверах был специально разработан протокол HTTP.

Клиенты используют сайты как удобный инструмент для различных целей и задач. Они могут посещать сайты для поиска информации о продуктах, услугах или компании, чтобы сделать информированное решение о покупке. Кроме того, клиенты могут совершать покупки онлайн, выбирая нужные товары, оформляя заказ и оплачивая его через сайт.

Таким образом сайты играют ключевую роль в удовлетворении потребностей клиентов, предоставляя им доступ к информации, возможность совершения покупок, взаимодействия с контентом и получения обратной связи. Удобство использования сайта и качественное клиентское обслуживание способствуют повышению уровня удовлетворенности клиентов, увеличению конверсии и повторных покупок, укреплению имиджа компании и формированию положительного отзыва о бренде.

Веб-сайты, в особенности, имеют значительное влияние для компаний и предпринимателей. Они не только служат визитной карточкой для компаний в онлайн-пространстве, но и являются мощным инструментом для привлечения новых клиентов и увеличения продаж. Через веб-сайт компания может представить свои продукты и услуги, принимать заказы и осуществлять продажи онлайн. Веб-сайт улучшает обслуживание клиентов, предоставляя им информацию, возможность задать вопросы и решить проблемы.

Еще одним важным преимуществом веб-сайтов является возможность создания имиджа компании и укрепления доверия у клиентов. Профессионально оформленный и информативный веб-сайт способствует формированию положительного впечатления о компании, ее ценностях и уникальности. Также веб-сайт позволяет компаниям отслеживать и анализировать поведение пользователей, трафик, конверсию и другие метрики, что помогает в оптимизации бизнес-процессов.

Глобальная доступность веб-сайта позволяет компаниям присутствовать на мировом рынке и привлекать клиентов из различных стран и регионов. Это открывает новые возможности для развития бизнеса, увеличения его конкурентоспособности и укрепления позиций на рынке. В целом, веб-сайты играют важную роль в современном бизнесе, обеспечивая компаниям множество преимуществ и возможностей для роста и развития.

1.2 Структура сайта

1.2.1 Общая информация

Веб-сайт – это онлайн-ресурс, который представляет собой виртуальное пространство, доступное через интернет, также он может находиться на локальном сервере. Веб-сайт состоит из нескольких ключевых элементов, каждый из которых играет важную роль в его функционировании.

Первым элементом является доменное имя, которое представляет собой уникальный адрес сайта в сети Интернет. Пользователи вводят доменное имя в адресной строке браузера, чтобы перейти на соответствующий веб-сайт. Дизайн и макет веб-сайта определяют его внешний вид и оформление, включая цветовую гамму, шрифты, расположение элементов и общую структуру страниц. Навигация на сайте представляет собой меню, ссылки и другие элементы, которые помогают пользователям ориентироваться и переходить между различными страницами.

Контент – это информация, тексты, изображения, видео и другие материалы, которые представлены на веб-сайте для привлечения и информирования посетителей. Функциональность включает различные возможности и инструменты на сайте, такие как поиск, фильтрация, формы обратной связи, корзина покупок и другие функции. Адаптивность веб-сайта обеспечивает его корректное отображение на различных устройствах, от компьютеров до мобильных устройств.

Безопасность – это важный аспект веб-сайта, включающий меры защиты от взломов, хакерских атак и утечек данных пользователей. Аналитика предоставляет инструменты для сбора и анализа данных о посещаемости сайта, поведении пользователей, конверсии и других метриках. В целом, веб-сайт представляет собой комплексный инструмент, объединяющий различные элементы и компоненты для обеспечения удобства использования, информативности и эффективности взаимодействия с пользователями.

Структура сайта с точки зрения разработчиков включает в себя несколько ключевых аспектов, которые определяют его функциональность, эффективность и удобство использования.

Первым важным элементом является серверная часть, которая включает в себя сервер, на котором размещается сайт, и языки программирования (например, PHP, Python, Ruby, JavaScript), используемые для создания динамического контента и обработки запросов пользователей.

Другим важным аспектом является клиентская часть, которая включает в себя HTML, CSS и JavaScript - основные технологии для создания веб-страниц, их стилей и интерактивных элементов.

База данных играет ключевую роль в хранении информации, такой как пользовательские данные, заказы, комментарии и другая информация, которая отображается на сайте. Разработчики используют языки запросов к базе данных, такие как SQL, для работы с данными.

Также важным аспектом структуры сайта является архитектура информации, которая определяет логическую организацию контента на сайте, навигацию и взаимосвязи между различными разделами и страницами. Четкая и интуитивно понятная архитектура информации способствует удобству использования сайта для пользователей.

1.2.2 Взаимодействие клиентов с разрабатываемым сайтом

Пользователь при использовании сайта по продаже железнодорожных билетов должен иметь возможность просматривать доступные маршруты, рейсы, города, вокзалы. Также необходимо иметь возможность искать интересующие пользователя маршруты, выбирать подходящий рейс и покупать на него билет. Пользователю необходимо вводить свои данные для покупки билета, поэтому ему должен быть доступен ввод данных на сайте, которые в дальнейшем добавятся в базу данных. После покупки билета пользователь должен получать подтверждение об успешной покупке билета.

1.2.3 Описание структуры разрабатываемого сайта

Представим ситуацию взаимодействия клиентов с сайтом по продаже железнодорожных билетов. Клиент заходит на сайт и видит удобный интерфейс, где может выбрать пункт отправления, прибытия и дату. Он вносит необходимую информацию и получает список доступных билетов с указанием времени, городов, класса обслуживания и других деталей.

Все вышеперечисленные действия должны быть на главной странице сайта, поэтому она должна содержать кнопку поиска доступных рейсов и форму поиска рейсов с полями ввода:

1. Города отправления;
2. Города прибытия;
3. Дату поездки.

Далее все найденные рейсы должны выводится так же на главной странице после области с поиском и содержать данные о городах, вокзалах и времени отправления и

прибытия, о времени в пути. В рейсе должна быть возможность выбора типа вагона. Выбор должен осуществляться нажатием кнопки с именованием типа вагона и информацией об оставшемся количестве мест.

После выбора рейса и типа вагона пользователю необходимо проверить всю выбранную информацию и приступить к заполнению личных данных для билета. Этот этап должен происходить на новой странице. То есть после нажатия на кнопку с типом вагона на определенном рейсе, должна открываться новая страница с информацией о рейсе, предоставленной ранее, типе вагона, номере вагона, цене самого билета.

На этой же странице после отображения информации должна находиться форма для заполнения личными данными покупателя.

В форме необходимо иметь поля для ввода:

1. Фамилии;
2. Имени;
3. Отчества;
4. Даты рождения;
5. Серии паспорта;
6. Номера паспорта;
7. Номера телефона;
8. Электронной почты.

После заполнения формы должна находиться кнопка для перехода на следующую страницу покупки билета. На новой странице должна быть выведена вся информация о рейсе, выбранном месте, покупателе и цене. Только после повторной проверки всех данных покупатель может купить билет, нажав на кнопку для покупки. После нажатия этой кнопки вся информация занесется в базу данных для регистрации покупки билета.

Помимо функции покупки билета на сайте необходимо иметь информацию о городах и вокзалах, в которых покупается билет. Покупатель должен иметь возможность без формы на главной странице посмотреть все имеющиеся маршруты, города и вокзалы. Поэтому необходимо также создать страницы с этой информацией.

На странице «Вокзалы и маршруты» нужно указать перечень городов, для которых сайт продает билеты. Ниже списка городов должны быть все имеющиеся на данный момент маршруты. Надпись каждого города должна вести на новую страницу с вокзалами. Если вокзалов у города несколько, то на новой странице так же выводится перечень вокзалов выбранного города, при переходе на которые выводилась бы страница вокзала. Если же у города всего один вокзал, то ссылка должна вести сразу на страницу этого вокзала.

Конечная страница с вокзалом должна отображать все маршруты, имеющиеся на этом вокзале. Сначала должны быть выведены маршруты, направляющиеся из выбранного вокзала, затем маршруты, направляющиеся в него. На каждой странице города или вокзала обязательно должна быть ссылка на предыдущую страницу, таким образом пользователю будет удобнее переходить между страницами.

Также на сайте необходимо иметь информацию, не относящуюся к рейсам, билетам и вокзалам, а специальные страницы для ознакомления пользователя с работой сайта. Такая информация может находиться на страницах «Помощь и руководство» и «Карта сайта».

На странице «Помощь и руководство» пользователи должны иметь возможность найти ответы на часто задаваемые вопросы, инструкции по использованию функционала сайта, руководства пользователя и другую полезную информацию. Это позволяет пользователям быстро решать проблемы, возникающие при использовании сайта, обучаться новым возможностям и повышать уровень удовлетворенности от взаимодействия с ресурсом. Предоставление доступной и полезной информации на странице «Помощь и руководство» способствует улучшению пользовательского опыта и повышению уровня комфорта при работе с сайтом.

Как видно из названия страницы, в «Карте сайта» должна находиться карта сайта, которая представляет собой виртуальную модель структуры веб-ресурса. Она визуально отображает иерархию всех доступных страниц и разделов сайта. Карта обычно представляется в виде древовидной диаграммы или списком ссылок, позволяющих пользователям легко найти нужную информацию и перейти к соответствующим разделам.

Карта сайта обычно включает в себя основные категории и подкатегории контента, ссылки на все страницы сайта. Она может быть представлена как для посетителей сайта, чтобы помочь им быстро найти нужную информацию, так и для поисковых систем, чтобы облегчить индексацию контента и улучшить SEO-показатели веб-ресурса.

Для удобства пользования сайтом на нем должно быть меню, чтобы, находясь в любой части сайта, можно было перейти на другие страницы или вернуться на главную.

Для меню можно выделить 4 основные страницы сайта:

1. Главная страница с поиском рейсов;
2. Страница «Вокзалы и маршруты»;
3. «Руководство и помощь»;
4. «Карта сайта» со всеми ссылками.

Описание структуры разрабатываемого сайта дает представление об объеме работы, об информации, с которой придется работать в базе данных, о функциональных возможностях сайта и о количестве взаимодействии интерфейса с базой данных.

2 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ

2.1 Инструменты для разработки

Сайт создавался и использовался в учебных целях на локальном хосте компьютера. Для хранения и использования файлов проекта выступил OpenServer - программная среда для веб-разработки. Данный программный комплекс включает в себя тщательно подобранный набор серверного программного обеспечения, а также удобную управляющую утилиту с широкими возможностями по администрированию и настройке всех имеющихся компонентов. На сегодняшний день OSPanel широко используется веб-разработчиками со всего мира с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях. Он позволяет быстро развернуть рабочую среду и сразу начать изучение веб-технологий без сложных манипуляций по установке и настройке большого количества программного обеспечения.

Для работы над проектом в OpenServer были использованы приложение PHPMyAdmin для работы с базой данных и сервер Apache, а также дополнительно установленный на компьютер и настроенный Node.js и фреймворк Express.

Node (или более формально Node.js) - кросплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык JavaScript. Среда исполнения предназначена для использования вне контекста браузера (т.е. выполняется непосредственно на компьютере или на серверной ОС). Таким образом, среда исключает API-интерфейсы JavaScript для браузера и добавляет поддержку более традиционных OS API-интерфейсов, включая библиотеки HTTP и файловых систем.

Node был разработан для оптимизации пропускной способности и масштабируемости в веб-приложениях и очень хорошо справляется со многими распространёнными проблемами веб-разработки (например, веб-приложения реального времени). JavaScript является относительно новым языком программирования и имеет преимущества от улучшения дизайна языка по сравнению с другими традиционными языками для веб-серверов (например, Python, PHP, и т.д.).

Менеджер пакетов Node (NPM) обеспечивает доступ к сотням тысяч многоразовых пакетов. Он также имеет лучшее в своём классе разрешение зависимостей и может также использоваться для автоматизации большинства инструментов построения.

Express - самый популярный веб-фреймворк для Node. Он является базовой библиотекой для ряда других популярных веб-фреймворков Node.

2.2 Файловая структура проекта

Файлы кода — документы, написанные на языках HTML, CSS и JavaScript. Файлы HTML нужны, чтобы все элементы страницы — заголовки, картинки, текст — были показаны в правильном порядке. CSS — это язык стилей, с его помощью задают внешний вид элементов — например, цвет и размер шрифта. На JavaScript пишут сценарии поведения элементов. Например, с его помощью можно сделать так, чтобы кнопка меняла цвет, когда на неё наводят курсор.

Так как для создания сайта используются Node.js и Express, которые работают с backend частью, то все файлы в проекте имеют расширение .js – файлы с JavaScript, .ejs – файлы с HTML. Исключением являются файлы со стилями - .css, изображения - .png.

Иерархия файлов выглядит следующим образом:

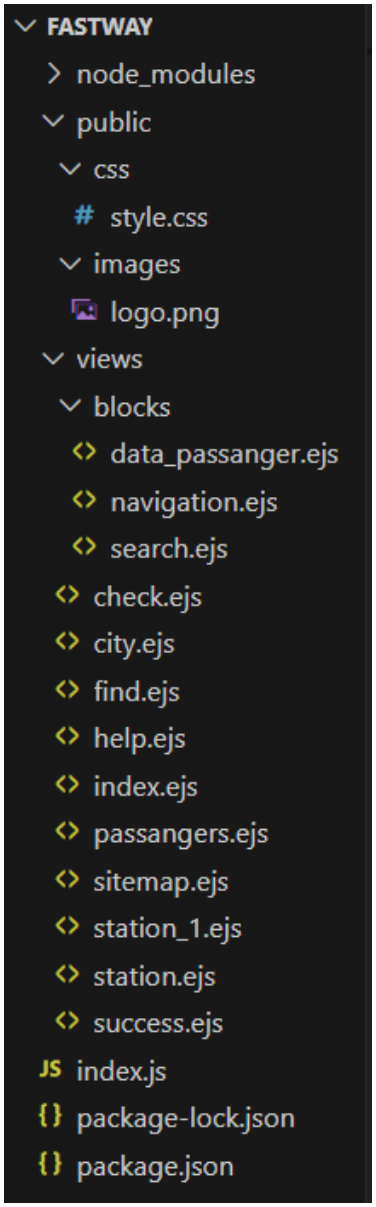

Рисунок 1. Иерархия файлов в проекте

В корневой папке FastWay лежат все файлы и файл запуска работы сайта – index.js. В папке node_modules хранятся все модули для Node.js и Express. В папке public хранятся файлы со стилями и изображениями. В папке views хранятся основные файлы сайта, каждый файл является страницей или шаблоном для загрузки нескольких одинаковых страниц. В папке blocks – файлы с шаблонным кодом, которые подключаются в других файлах для сокращения написания кода и использования повторяющихся элементов.

Перед началом работы необходимо включить OpenServer и запустить файл index.js через консоль.

2.3 Вывод данных из базы данных

Для начала нужно создать меню, чтобы при разработке сайта и в дальнейшем было удобно и комфортно пользоваться сайтом и страницами. Для этого создается шаблон navigation.ejs, включающий в себя, на момент окончания работы, 4 основные страницы сайта, перед этим создаются get-запросы для страниц «Вокзалы и маршруты», «Помощь и руководство», «Карта сайта». Дополнительно в шаблон добавляется логотип и название сайта. Шаблон подключается к каждой странице: <%- include ('blocks/navigation') -%>.

```
app.get('/sitemap', (req, res) => {
    res.render('sitemap');
});

app.get('/help', (req, res) => {
    res.render('help');
});
```

Рисунок 2. Get-запрос для страниц «Помощь и руководство» и «Карта сайта»

```
<header class="header">
    <div class="header-content">
        <img src="/images/logo.png" class="logo">
        <div>
            <h1><span class="gradient-text">FastWay</span>
                <span class="additional-text">Поиск дешевых Ж/Д билетов</span></h1>
        </div>
    </div>
    <nav class="navigation">
        <div class="nav-wrapper">
            <a href="/" class="nav-btn">Поиск билетов</a>
            <a href="/station" class="nav-btn">Вокзалы и маршруты</a>
            <a href="/help" class="nav-btn">Помощь и руководство</a>
            <a href="/sitemap" class="nav-btn">Карта сайта</a>
        </div>
    </nav>
</header>
```

Рисунок 3. Файл navigation.ejs с шапкой сайта

Для функционирования сайта необходимо передавать нужные данные из базы данных на страницы сайта. В моем случае нужно передать информацию о доступных маршрутах, городах и вокзалах на страницу «Вокзалы и маршруты». Для этого был создан

get-запрос, в котором описано, что если перейти по адресу /station, то будет собрана вся информация о маршрутах и городах из базы данных и передана в файл station.ejs.

```
app.get('/station', (req, res) => {
    let sql_count_stations = 'SELECT City.id_city, City.name_city, COUNT(Station.id_station) AS station_count FROM Station INNER JOIN Ci
    let sql_stations = 'SELECT name_station, id_city FROM Station';
    let sql_flight = "SELECT Flight.flight_number, Flight.time_way, Flight.time_start, Flight.time_end, Flight.number_train, Train_categ
    connection.query(sql_count_stations, (error, results_count) => {
        if (error) {
            console.error('Ошибка при запросе к базе данных:', error);
            return res.status(500).send('Ошибка при запросе к базе данных');
        } else {
            connection.query(sql_stations, (error, results_stations) => {
                if (error) {
                    console.error('Ошибка при запросе к базе данных:', error);
                    return res.status(500).send('Ошибка при запросе к базе данных');
                } else {
                    connection.query(sql_flight, (error, results_flight) => {
                        if (error) {
                            console.error('Ошибка при запросе к базе данных:', error);
                            return res.status(500).send('Ошибка при запросе к базе данных');
                        }
                        const flights_all = results_flight.map(flight => ({
                            flight_number: flight.flight_number,
                            time_way: flight.time_way,
                            time_start: flight.time_start,
                            time_end: flight.time_end,
                            number_train: flight.number_train,
                            name_train_category: flight.name_train_category,
                            city_1: flight.city_1,
                            city_2: flight.city_2,
                            station_1: flight.station_1,
                            station_2: flight.station_2
                        }));
                        res.render('station', {count_station: results_count, stationName: results_stations, flights_all: flights_all});
                    });
                }
            });
        }
    });
});
```

Рисунок 4. Get-запрос для файла station.ejs

Файл station.ejs получает данные и эта страница выводится на сайте. В файле имеются 2 блока – города и маршруты. Городам добавляется ссылка либо на страницу со списком его вокзалов, если у него их несколько, либо сразу на страницу вокзала, если он один.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <%- include ('blocks/navigation') -%>
    <div class="container">
        <div class="flight-info" style="gap: 8px; width: 800px;">
            <h2>Города</h2>
            <% count_station.forEach(city => { %>
                <% if (city.station_count > 1) { %>
                    <span><a href="/station/city_<%= city.name_city %>"><%= city.name_city %></a></span>
                <% } else { %>
                    <% const station = stationName.find(station => station.id_city === city.id_city); %>
                    <% if (station) { %>
                        <span><a href="/station/city_<%= city.name_city %>/station_<%= station.name_station %>"><%= city.name_city %></a></span>
                    <% } else { %>
                        <span><a href="/station/city_<%= city.name_city %>">No station found for <%= city.name_city %></a></span>
                    <% } %>
                <% } %>
            <% }); %>
        </div>
```

Рисунок 5. Файл station.ejs, вывод городов

```
<h2>Маршруты</h2>

    <% flights_all.forEach(flight => { %>
        <div class="flight">
        <% if (flight.name_train_category.charAt(0) === 'Л') { %>
            <div class="flight-info" style="width: 900px;">
                <div class="flight-info-train">
                    <span style="font-weight: bold; color: rgb(3, 3, 195);">
                        <%= 'Ласточка' %>
                    </span>
        <% } else if (flight.name_train_category.charAt(0) === 'С') { %>
            <div class="flight-info" style="width: 900px;">
                <div class="flight-info-train">
                    <span style="font-weight: bold; color: rgb(213, 0, 0);">
                        <%= 'Сапсан' %>
                    </span>
        <% } else if (flight.name_train_category.charAt(0) === 'К') { %>
            <div class="flight-info"  style="width: 900px;">
                <div class="flight-info-train">
                    <span style="font-weight: bold; color: rgb(1, 104, 1);">
                        <%= 'Стандартный' %>
                    </span>
        <% } %>

        <span>Рейс: <%= flight.flight_number %></span>
        <span>Поезд: <%= flight.number_train %></span>

        </div>
        <div class="flight-info-city-time">
            <div class="flight-info-out">
                <span style="font-weight: bold; color:#016d8e;">
                    <%= flight.city_1 %><br>Вокзал <%= flight.station_1 %></span>
                <span><%= flight.time_start %></span>
            </div>
            <div class="flight-info-time">
                <span>------------------------></span>
                <span>Время в пути: <%= flight.time_way %></span>
            </div>
            <div class="flight-info-in">
                <span style="font-weight: bold; color:#016d8e;">
                    <%= flight.city_2 %><br>Вокзал <%= flight.station_2 %></span>
                <span><%= flight.time_end %></span>
```

Рисунок 6. Файл station.ejs, вывод маршрутов

Как уже было сказано, если у города несколько вокзалов, то ссылка города переводит пользователя на страницу города с перечнем вокзалов. Для перехода по ссылке создается get-запрос, передающий список вокзалов в файл city.ejs.

```
app.get('/station/city_:cityName', (req, res) => {
    const cityName = req.params.cityName;
    let sql_station_name = 'SELECT station.name_station FROM station JOIN city O
    connection.query(sql_station_name, [cityName], (err, results) => {
        if (err) {
            console.error('Ошибка при запросе данных из базы данных:', err);
            res.status(500).send('Ошибка при запросе данных из базы данных');
            return;
        }
            res.render('city', { cityName: cityName, stations: results });
    });
});
```

Рисунок 7. Get-запрос для файла city.ejs

```ejs
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <%- include ('blocks/navigation') -%>
    <div class="container">
        <div class="flight-info" style="gap: 8px; width: 800px;">
        <a href="#" onclick="history.back(); return false;">← Назад</a>
        <h2>Вокзалы города <%= cityName %></h2>
            <% stations.forEach(station => { %>
                <span><a href="/station/city_<%= cityName %>/station_<%= station.name_station %>">
                    <%= station.name_station %></a></span>
            <% }); %>
        </div>
    </div>
</body>
</html>
```

Рисунок 8. Файл city.ejs, вывод вокзалов

Далее необходимо создать get-запрос для перехода на страницы вокзалов и просмотра маршрутов у них. Данные передаются в файл station_1.ejs.

```js
app.get('/station/city_:cityName/station_:stationName', (req, res) => {
    const cityName = req.params.cityName;
    const stationName = req.params.stationName;
    let sql_flight_station = "SELECT Flight.flight_number, Flight.time_way, Flight.time_start, Flight.time_end, Flight.number_train
    connection.query(sql_flight_station, [stationName, stationName], (err, results) => {
        if (err) {
            console.error('Ошибка при запросе данных из базы данных:', err);
            res.status(500).send('Ошибка при запросе данных из базы данных');
            return;
        }
        const flights = results.map(flight => ({
            flight_number: flight.flight_number,
            time_way: flight.time_way,
            time_start: flight.time_start,
            time_end: flight.time_end,
            number_train: flight.number_train,
            name_train_category: flight.name_train_category,
            city_1: flight.city_1,
            city_2: flight.city_2,
            station_1: flight.station_1,
            station_2: flight.station_2
        }));
        const flights_out = flights.filter(flight => flight.station_1 === stationName);
        const flights_in = flights.filter(flight => flight.station_2 === stationName);
        res.render('station_1', { cityName: cityName, stationName: stationName, flights_out: flights_out, flights_in: flights_in});
    });
});
```

Рисунок 9. Get-запрос для файла station_1.ejs

В файле station_1.ejs написан код для вывода маршрутов из данного вокзала и в него. Код в двух случаях аналогичен.

```
<h3>Рейсы из вокзала:</h3>
<% if (flights_out.length > 0) { %>
    <% flights_out.forEach(flight => { %>
        <div class="flight">
            <% if (flight.name_train_category.charAt(0) === 'Л') { %>
                <div class="flight-info" style="gap: 8px; width: 900px;">
                    <div class="flight-info-train">
                        <span style="font-weight: bold; color: rgb(3, 3, 195);">
                            <%= 'Ласточка' %>
                        </span>
            <% } else if (flight.name_train_category.charAt(0) === 'С') { %>
                <div class="flight-info" style="gap: 8px; width: 900px;">
                    <div class="flight-info-train">
                        <span style="font-weight: bold; color: rgb(213, 0, 0);">
                            <%= 'Сапсан' %>
                        </span>
            <% } else if (flight.name_train_category.charAt(0) === 'К') { %>
                <div class="flight-info" style="gap: 8px; width: 900px;">
                    <div class="flight-info-train">
                        <span style="font-weight: bold; color: rgb(1, 104, 1);">
                            <%= 'Стандартный' %>
                        </span>
            <% } %>

            <span>Рейс: <%= flight.flight_number %></span>
            <span>Поезд: <%= flight.number_train %></span>

            </div>
            <div class="flight-info-city-time">
                <div class="flight-info-out">
                    <span style="font-weight: bold; color: #016d8e;">
                        <%= flight.city_1 %><br>Вокзал <%= flight.station_1 %></span>
                    <span><%= flight.time_start %></span>
                </div>
                <div class="flight-info-time">
                    <span>------------------------></span>
                    <span>Время в пути: <%= flight.time_way %></span>
                </div>
                <div class="flight-info-in">
                    <span style="font-weight: bold; color: #016d8e;">
                        <%= flight.city_2 %><br>Вокзал <%= flight.station_2 %></span>
                    <span><%= flight.time_end %></span>
```

Рисунок 8. Файл station_1.ejs, вывод маршрутов из вокзала

Get-запрос для страницы «Помощь и руководство» был описан ранее. При переходе по ссылке запускается файл help.ejs. Там находится текст с информацией для пользователя.

```
<html lang="ru">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <%- include ('blocks/navigation') -%>
    <div class="container">
        <div class="flight-info" style="gap: 8px; width: 800px;">
            <h2>Помощь и руководство</h2>
            <span>Добро пожаловать на сайт по продаже жд билетов!<br><br>

                Поиск и выбор билетов:
                <br> - Используйте удобный поиск для нахождения подходящего маршрута и билетов.
                <br> - Напишите города отправления и прибытия, дату и нажмите на кнопку "Поиск".
                <br> - Выберите рейс и тип вагона и нажмите на него.

                <br><br> Оформление заказа:
                <br>- Перейдите к выбранному рейсу и типу вагона и убедитесь, что выбранные билеты указаны корректно.
                <br>- Введите свои личные и контактные данные в форму.
                <br>- После ввода данных нажмите на кнопку "Продолжить".
                <br>- Проверьте свои данные и данные билета.
                <br>- Произведите оплату.
                <br><br>
                <br>Если у вас возникли вопросы или проблемы, обратитесь в службу поддержки сайта для получения помощи.
                <br><br>
                <br>Спасибо, что выбрали наш сайт для покупки ж/д билетов. Приятной поездки!</span>
        </div>
</div>
</body>
</html>
```

Рисунок 9. Файл help.ejs, вывод вспомогательной информации

Get-запрос для страницы «Карта сайта» был описан ранее. При переходе по ссылке запускается файл sitemap.ejs.

```
<h2>Карта сайта</h2>
<ul>
    <li class="level-1">
        <a href="/">Главная (Поиск билетов и покупка)</a>
    </li>
    <li class="level-1">
        <a href="/station">Вокзалы и маршруты</a>
    </li>
    <li class="level-2">
        <a href="/station/city_Адлер/station_Адлер">Вокзал Адлер</a>
    </li>
    <li class="level-2">
        <a href="/station/city_Казань/station_Казань">Вокзал Казань</a>
    </li>
    <li class="level-2">
        <a href="/station/city_Краснодар/station_Краснодар">Вокзал Краснодар</a>
    </li>
    <li class="level-2">
        <a href="/station/city_Москва">Город Москва</a>
    </li>
    <li class="level-3">
        <a href="/station/city_Москва/station_Белорусский">Вокзал Белорусский</a>
    </li>
```

Рисунок 10. Файл sitemap.ejs, вывод ссылок сайта

2.4 Дизайн сайта

Дизайн сайта необходим для создания привлекательного и удобного веб-ресурса, который привлечет внимание пользователей, обеспечит удобство использования, повысит конверсию и укрепит образ бренда. Дизайн способствует удержанию посетителей на сайте, облегчает навигацию, делает контент доступным.

Вся разметка и стили находятся в файлах с расширением .ejs и .css. На предыдущих изображениях можно увидеть html-разметку в файлах .ejs.

Для компании продажи ж/д билетов необходим логотип и название. Как было выяснено ранее, название компании FastWay. Название и логотип располагаются в шапке сайта вместе с меню.

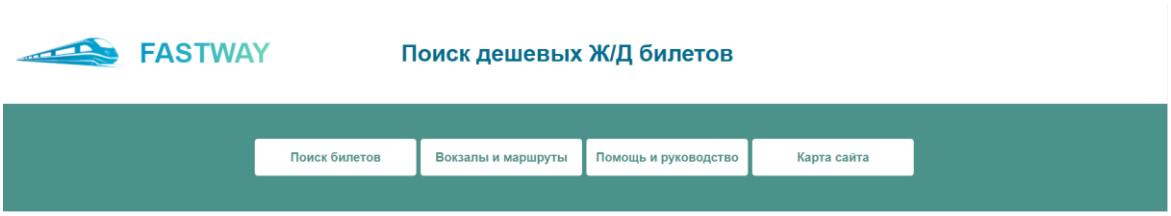

Рисунок 11. Шапка сайта

Форма поиска должна быть удобной, поэтому загружена библиотека flatpickr для отображения календаря у окошка с датой.

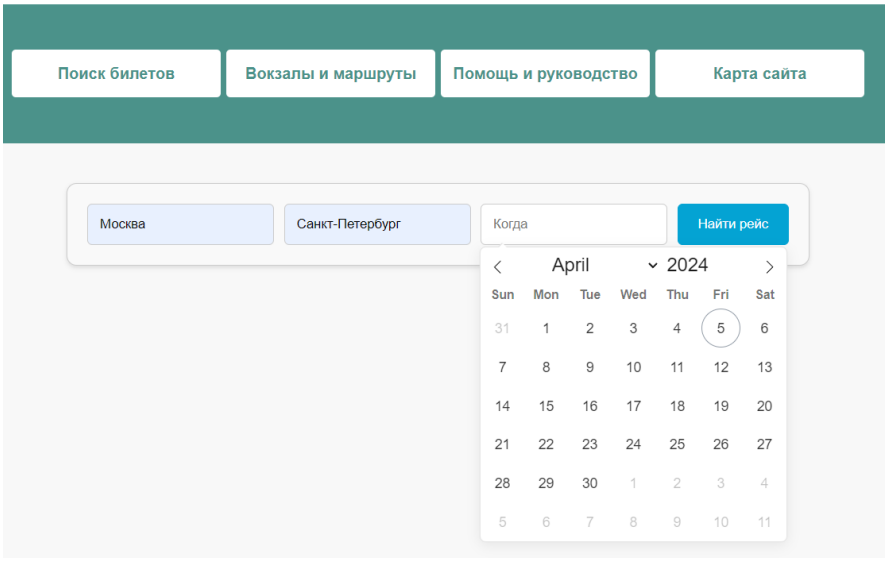

Рисунок 12. Форма поиска рейсов

Рейсы выводятся с большим количеством информации, поэтому для них нужно подобрать дизайн, который будет читаемым для покупателя. Для удобства типы поездов указаны разными цветами.

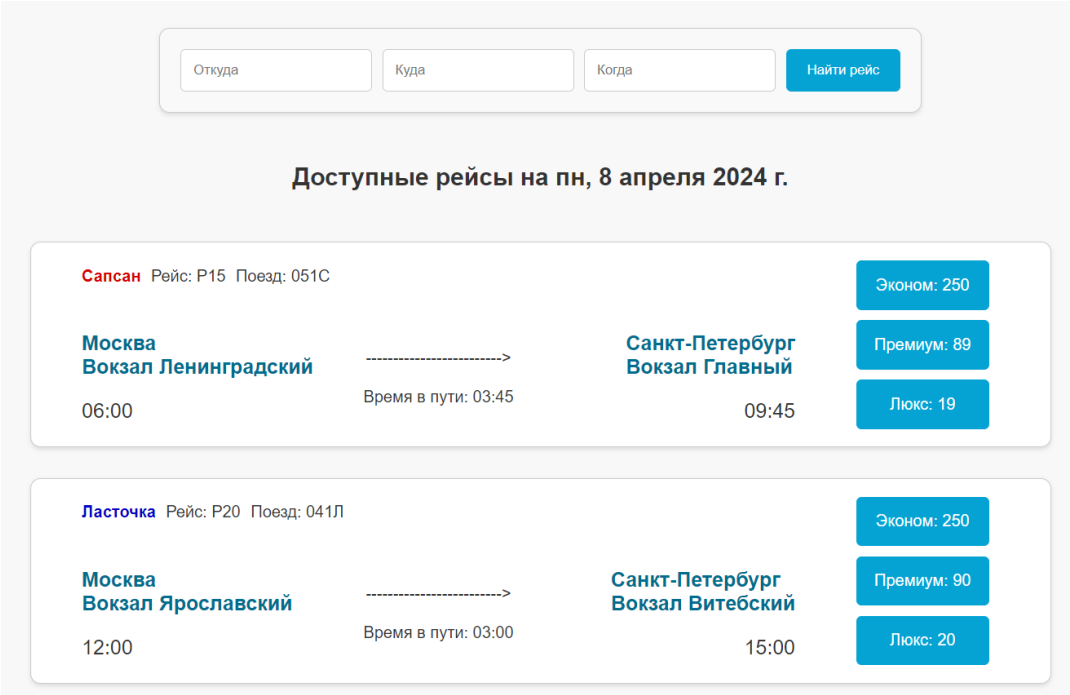

Рисунок 13. Вывод рейсов на главной странице

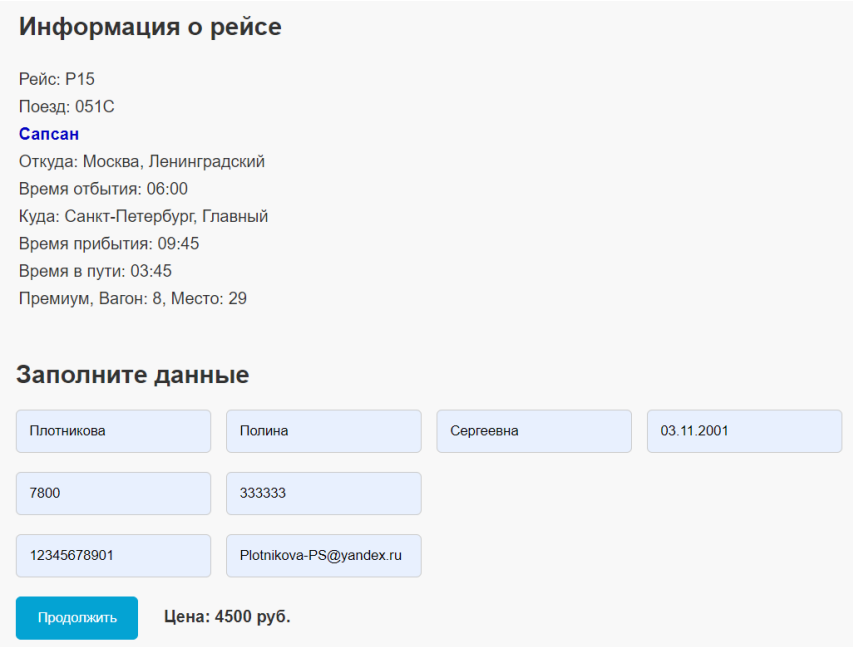

Рисунок 14. Проверка информации и ввод личных данных

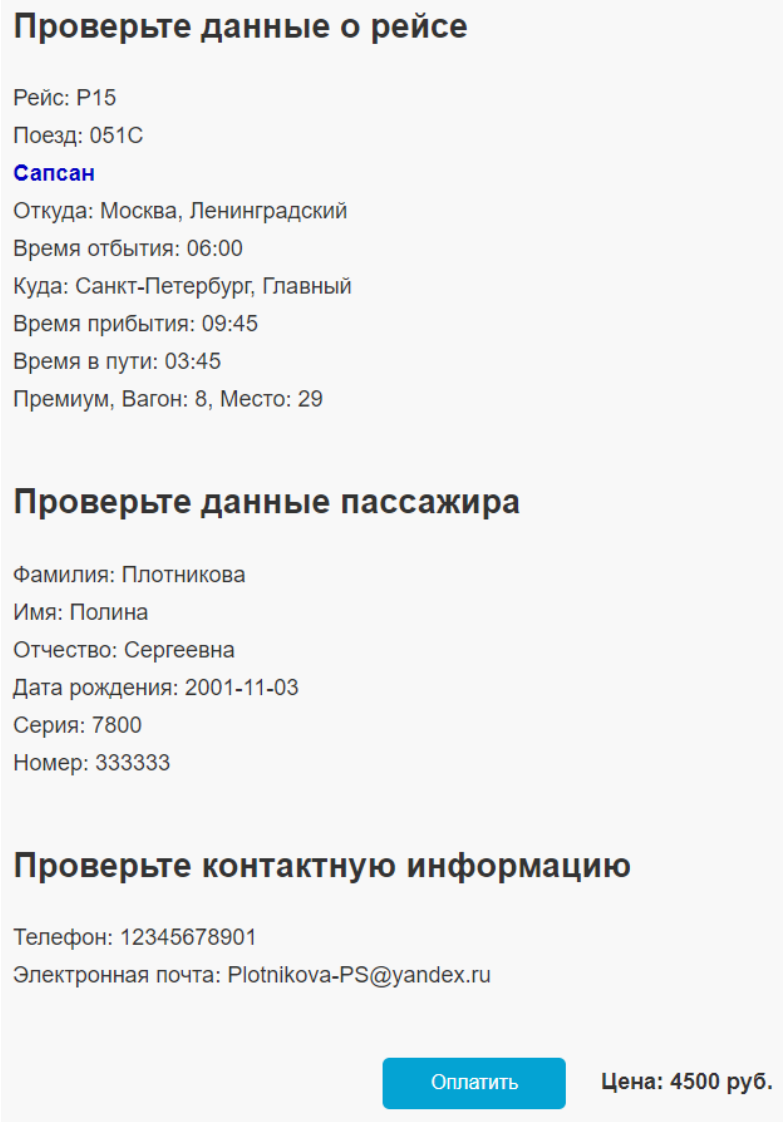

Рисунок 15. Повторная проверка информации и покупка билета

После оплаты появляется подтверждающее покупку сообщение.

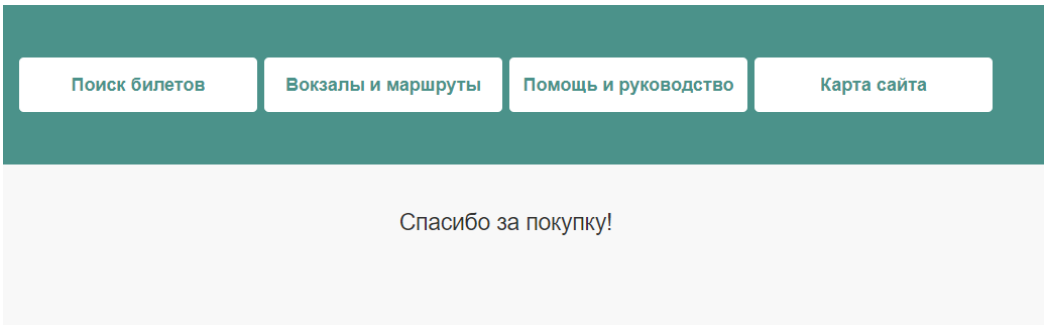

Рисунок 16. Подтверждение покупки

Далее будут показаны изображения вкладки «Вокзалы и маршруты».

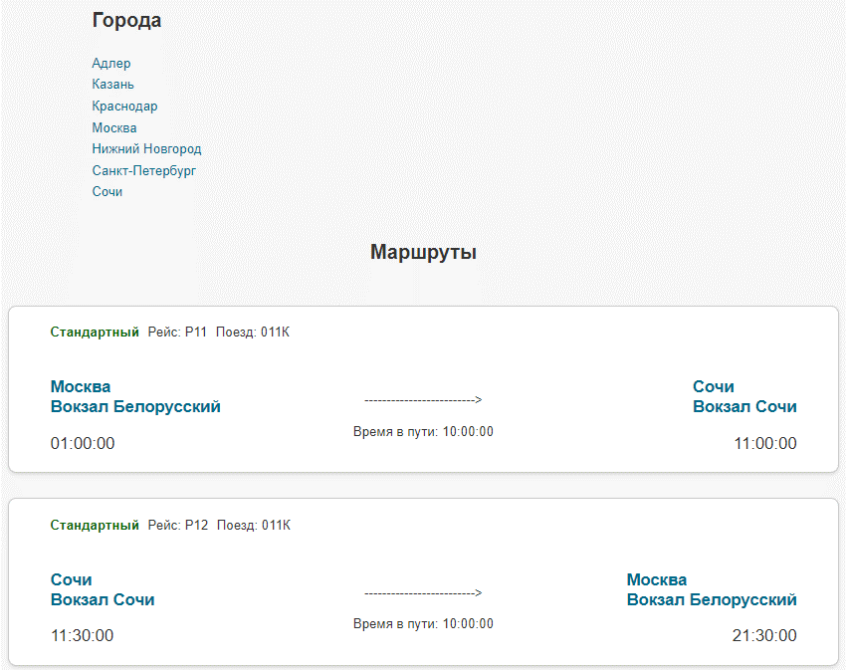

Рисунок 17. «Вокзалы и маршруты»

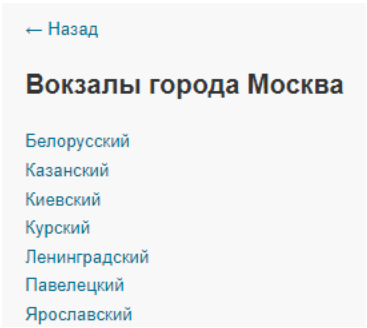

Рисунок 18. Список вокзалов города

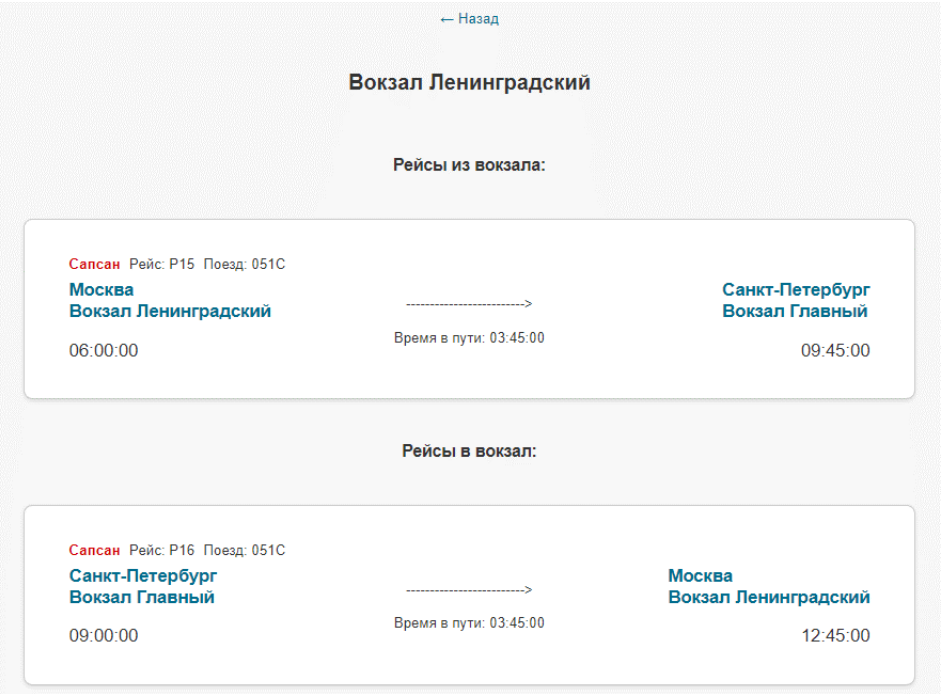

Рисунок 19. Маршруты вокзала

Далее рассмотрим страницу «Карта сайта».

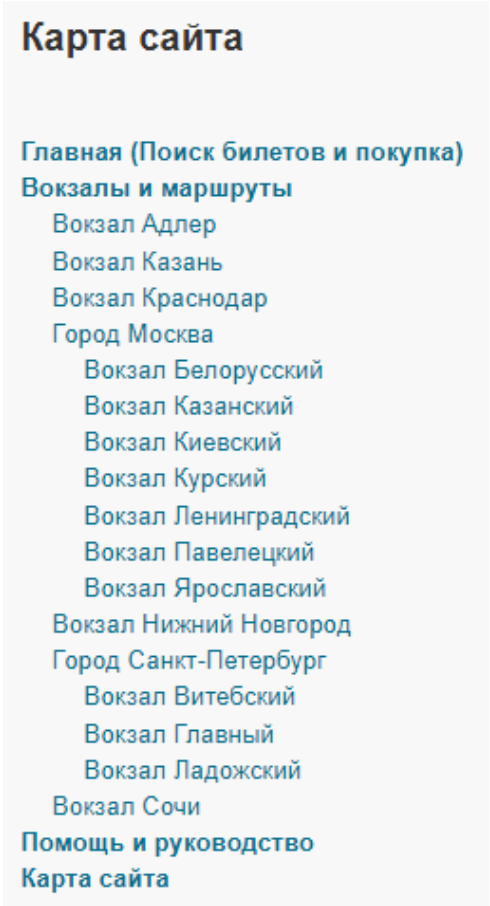

Рисунок 20. «Карта сайта»

Далее рассмотрим страницу «Помощь и руководство».

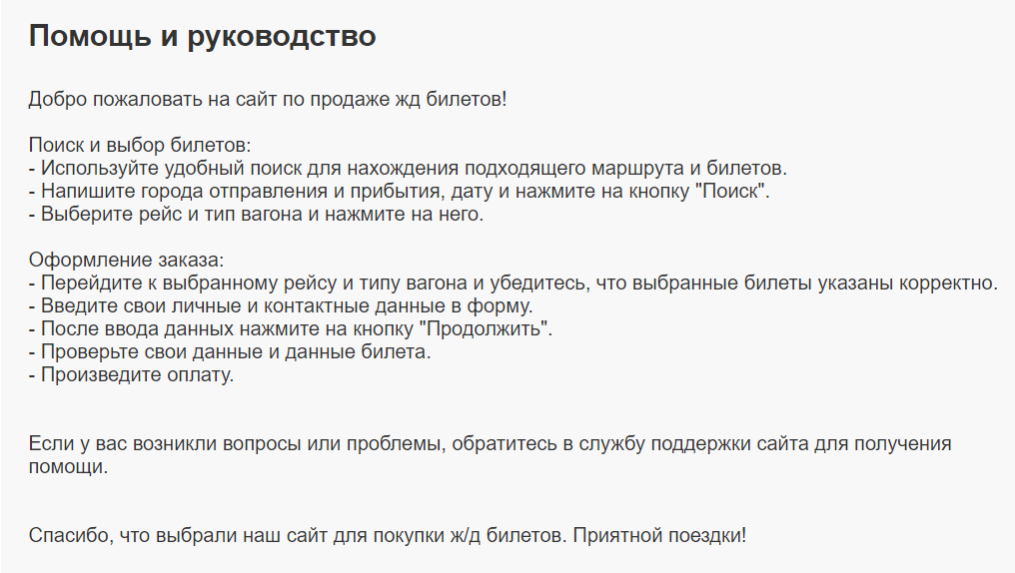

Рисунок 21. «Помощь и руководство»

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Введение

1.1 Область применения

Сайт по продаже ж/д билетов предназначен для удобного и быстрого приобретения билетов на поезд. Пользователи могут выбирать желаемый маршрут, дату и класс обслуживания, просматривать маршруты.

1.2 Краткое описание возможностей

- Поиск и выбор желаемого маршрута и даты отправления.
- Просмотр расписания поездов и доступных вариантов билетов.
- Оплата билетов.
- Уведомления о статусе заказа и изменениях в расписании.

1.3 Уровень подготовки пользователя

Сайт должен быть дружелюбным к пользователям с разным уровнем подготовки. Для основного функционала (поиск, выбор билетов, покупка) достаточно базовых навыков работы с компьютером и интернетом.

1.4 Перечень эксплуатационной документации:

- Руководство пользователя: описание функционала сайта, инструкции по поиску и бронированию билетов, оплате, сохранению данных и прочее.
- Политика конфиденциальности: информация о защите данных пользователей и использовании их личной информации.
- Условия использования: правила пользования сайтом, ответственность сторон, правила бронирования и отмены заказов.

2. Назначение и условия применения

Средство автоматизации, представленное сайтом по продаже ж/д билетов, предназначено для автоматизации процесса поиска, выбора, бронирования и оплаты желаемых железнодорожных билетов. Оно обеспечивает пользователям возможность быстрого доступа к информации о расписании поездов, наличии билетов, ценах, классах обслуживания и других параметрах, что значительно упрощает процесс планирования поездок и совершения покупок.

Условия, при соблюдении которых обеспечивается применение средства автоматизации в соответствии с назначением:

1) Сайт должен быть доступен на различных устройствах (компьютеры, планшеты, смартфоны) и совместим с различными браузерами.

2) Сайт должен корректно функционировать под различными операционными системами (Windows, MacOS, Linux) и использовать современные технологии для обеспечения безопасности и стабильной работы.

3) Пользователь должен иметь доступ к информации о поездках, билетах, оплатах и других данных через интерфейс сайта.

4) Пользователи сайта должны иметь базовые навыки работы с интернетом и понимания процесса покупки билетов онлайн. Специалисты, поддерживающие работу сайта, должны быть грамотно подготовлены по вопросам технической поддержки и обслуживания пользователей.

3. Подготовка к работе

Для открытия сайта на локальном сервере:

1) Скачать Openserver, Node.js;

2) Установить пакеты из архива;

3) Запустить package.json в консоли;

4) Запустить Openserver;

5) Запустить Node.js в консоли.

Проверить работу проекта можно открыв в браузере localhost:3000/.

1) Попробовать выполнить поиск и выбор желаемого маршрута, даты и класса обслуживания.

2) Попробовать забронировать билет и осуществить оплату.

3) Администратору проверить изменения в базе данных после покупки билета.

4. Описание операций

1) Поиск рейсов при помощи ввода интересующих данных по рейсу в форме.

Записанные в форму данные после нажатия кнопки «найти» отправляются на сервер. Затем эти данные используются для вывода рейсов, удовлетворяющих введенным данным.

2) Выбор рейса с типом вагона.

При нажатии на кнопку с определенным типом вагона данные рейса отправляются на сервер. Затем сервер передает эти данные на страницу вывода информации о рейсе и ввода личных данных покупателя в виде той самой информации о рейсе.

3) Ввод личной и контактной информации для покупки билета.

Введенные в форму данные после нажатия кнопки «Продолжить» отправляются на сервер. Затем сервер передает эти данные на страницу вывода информации о рейсе и личных и контактных данных покупателя.

4) Покупка билета.

При нажатии кнопки «Оплатить» все данные со страницы с информацией о рейсе и покупателе переносятся на сервер, где эти данные добавляются в базу данных как купленный билет. Также добавляется покупатель, если он новый, меняется количество свободных мест на рейсе в определенном вагоне.

5) Вывод всех имеющихся маршрутов и городов.

При переходе на страницу сервер отправляет из базы данных все города и маршруты.

6) Вывод вокзалов и маршруты вокзалов.

При переходе на страницу города сервер получает либо название города, либо название вокзала, и отправляет из базы данных либо все вокзалы города, либо маршруты вокзала.

5. Аварийные ситуации

При ошибке в данных пассажиров необходимо связаться с службой поддержки для внесения корректировок. При ошибках на страницах сайта также связаться со службой поддержки. При отсутствии открытия сайта проверить подключение ко всем имеющимся программам и верверу.

6. Рекомендации по освоению

Описание контрольного примера:

1) Войдите на сайт покупки ж/д билетов;

2) Выберите маршрут и дату поездки;

3) Выберите желаемый тип вагона;

4) Укажите контактные данные и данные пассажиров;

5) Проверьте все данные пассажира и рейса;

6) Произведите покупку;

7) Получите подтверждение покупки.

Правила запуска и выполнения контрольного примера (на локальном сервере):

1) Для запуска контрольного примера откройте браузер и введите адрес сайта покупки ж/д билетов localhost:3000/.

2) Следуйте шагам контрольного примера, выбирая необходимые опции и вводя запрашиваемые данные;

3) Убедитесь, что все данные введены корректно и подтвердите заказ;

4) Проверьте получение подтверждения покупки билета;

5) При необходимости свяжитесь с технической поддержкой сайта для решения возможных проблем.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была достигнута поставленная цель и получены ценные практические навыки в области проектирования и создания сайта.

В процессе работы были получены навыки работы с интерфейсом сайтов. Были улучшены навыки в анализе информации, обработке данных и их использовании. Также были улучшены навыки в обработке запросов пользователей и предоставлении информации, передачи данных между страницами сайта.

В целом, работа над этой курсовой работой помогла расширить свои знания и навыки в области веб-разработки, улучшить способность анализировать и решать проблемы, а также научиться работать с данными и информацией в сфере создания интернет-ресурсов.

Полученный опыт и знания окажутся полезными для дальнейшей профессиональной деятельности в сфере информационных технологий.

СПИСОК ЛИТЕРАТУРЫ

1. Тузовский А. Ф. Проектирование и разработка web-приложений: учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2022. — 218 с. — ISBN 978-5-534-00515-8.

2. Титов, В. А. Разработка WEB-сайта средствами языка HTML : учебное пособие / В. А. Титов, Г. И. Пещеров. – Москва : Институт мировых цивилизации, 2018. – 184 с. I – ISBN 978-5-9500469-3-3.

3. Полуэктова, Н. Р. Разработка веб-приложений: учебное пособие для вузов / Н. Р. Полуэктова. - Москва : Юрайт, 2023. - 204 с. - URL: https://urait.ru/bcode/519714 (дата обращения: 21.11.2023). - ISBN 978-5-534- 13715-6.

4. Роббинс, Дженнифер Нидерст. Веб-дизайн для начинающих : HTML, CSS, JavaScript и веб-графика : пер. с англ. / Дженнифер Нидерст Роббинс. - 5-е изд. - Санкт-Петербург : БХВ- Петербург, 2021. - 912 с., 1-44 с. цв. ил. : ил. - ISBN 978-1-491-96020-2.

5. 10 шагов, чтобы создать сайт с нуля – URL: https://contextonline.ru/blog/10-shagov-chtoby-sozdat-sayt-s-nulya/ (дата обращения 07.02.2023).

6. Команда для создания сайта — кто нужен? – URL: https://wezom.com.ua/blog/komanda-dlya-sozdaniya-sajta-kto-nuzhen (дата обращения 08.03.2023).