**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**Факультет: Информационные технологии и управление**

**Кафедра: Компьютерная инженерия**

**Предмет: «Веб-системы и технологии»**

Курсовая работа

**Тема** «Разработка онлайн информационной системы для aэропорта»

**Группа:** 680.22

**Курc:** 3

**Специальность: 050616** Информационные технологии

**Студент**: Алимов Зия

**Руководитель:** **асс. Халилов М. Э.**

**Зав.кафедрой: доц.Рагимова Н.А.**

**Баку – 2025**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ**

**ФАКУЛЬТЕТ**: Информационные технологии и управление

**КАФЕДРА**: Компьютерная инженерия

**Группа** 680.22 **курс** III

**Специальность** **050616** Информационные технологии

**Студент** Алимов Зия

**Зачетная книжка**

**Руководитель курсовой работы**  **асс. Халилов М. Э.**

**Срок выдачи**

**Дата сдачи**

**Тема курсовой работы** «Разработка онлайн информационной системы

для аэропорта»

**Отзыв руководителя курсовой**:

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

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

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

**Подпись студента**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись рук.курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись зав.кафедрой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата защиты курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Оценка**\_\_\_\_\_\_\_\_\_\_

**Пред.комиссии**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Члены комиссии**: 1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc198506700)

[ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc198506701)

[1.1 Web-программирование 6](#_Toc198506702)

[1.2 HTML и XHTML 7](#_Toc198506703)

[1.3 Dynamic HTML 10](#_Toc198506704)

[1.3.1 Каскадные таблицы стилей 10](#_Toc198506705)

[1.3.2 JavaScript 11](#_Toc198506706)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 13](#_Toc198506707)

[2.1 Описание предметной области 13](#_Toc198506708)

[2.2 Создание окружения для проекта 14](#_Toc198506709)

[2.3 Верстка HTML-страницы index.html 14](#_Toc198506710)

[2.4 Верстка HTML-страницы flights.html 17](#_Toc198506711)

[2.5 Верстка HTML-страницы contact.html 19](#_Toc198506712)

[2.6 Верстка HTML-страницы login.html 21](#_Toc198506713)

[2.7 Использование стилей CSS 22](#_Toc198506714)

[2.8 JavaScript 39](#_Toc198506715)

[3. РЕЗУЛЬТАТ 41](#_Toc198506716)

[4. ЛИТЕРАТУРА 42](#_Toc198506717)

[5. ПРИЛОЖЕНИЯ 43](#_Toc198506718)

ВВЕДЕНИЕ

Веб-системы и технологии представляют собой совокупность языков программирования, инструментов, платформ и методов, предназначенных для разработки, эксплуатации и сопровождения веб-приложений. Рост популярности интернет-сервисов и активное развитие сетевых технологий способствовали появлению большого количества новых решений, охватывающих как клиентскую, так и серверную стороны веб-разработки.

Внедрение веб-технологий значительно расширило доступ к информационным ресурсам и сделало возможным использование вычислительных систем широким кругом пользователей без специальных технических знаний. Существенное влияние также оказало повышение удобства взаимодействия с программным обеспечением благодаря развитию графических интерфейсов.

Современные веб-технологии обеспечивают функционирование Всемирной паутины (WWW) и включают:

* **Клиентские технологии**, ориентированные на отображение информации и взаимодействие с пользователем (HTML, CSS, JavaScript);
* **Серверные технологии**, обеспечивающие обработку запросов и доступ к базам данных (PHP, CGI и др.).

Типовая модель взаимодействия в веб-среде реализуется по архитектуре «клиент-сервер», при которой клиент отправляет запрос, а сервер возвращает соответствующий ответ.

Данная курсовая работа посвящена разработке информационной системы для аэропорта в формате веб-сайта «Аэропорт Онлайн».

Создание онлайн-системы включало в себя следующие этапы:

1. **Определение целей и функций сайта:**  
   На первом этапе были определены основные функции сайта: отображение расписания вылетов и прилётов, предоставление контактной информации, реализация формы авторизации и удобной навигации.
2. **Дизайн:**  
   Созданы страницы (главная, расписание, контакты, вход), выбраны цвета (синие и белые тона), реализованы элементы интерфейса (таблицы, карточки, кнопки);
3. **Разработка:**  
   Реализация проекта выполнена с использованием HTML, CSS и JavaScript.
   * Для стилизации использован файл style.css, в котором определены стили для всех элементов: шапки, меню, таблиц, форм, карточек, футера и баннера.
4. **Тестирование:**  
   Сайт протестирован в браузерах (Chrome, Firefox, Edge) и на разных устройствах (ПК, планшет, смартфон). Проверены корректность отображения и адаптивность.

Разработанный веб-сайт представляет собой информационную систему, ориентированную на пользователей аэропорта. Сайт обеспечивает доступ к актуальному расписанию рейсов, удобный интерфейс, а также адаптивную верстку под различные устройства. В результате реализованного проекта продемонстрированы основные этапы создания веб-приложения с использованием современных клиентских технологий.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Web-программирование

Web-программирование (или Web-разработка) — это активно развивающееся направление программирования, основная задача которого — создание динамических интернет-приложений. Языки Web-программирования принято делить на две основные категории: клиентские и серверные.

Клиентские языки обрабатываются на стороне пользователя, то есть в его браузере. При этом выполнение скриптов зависит от возможностей браузера: пользователь может отключить их обработку или использовать устаревшую версию браузера, в которой поддержка некоторых языков или их функций отсутствует. В современных браузерах подобные проблемы встречаются редко, а языки программирования обновляются достаточно медленно, поэтому риски несовместимости минимальны. Одной из особенностей клиентского кода является его открытость: любой желающий может просмотреть его, выбрав в браузере пункт меню "Исходный код страницы".

Плюсом клиентских языков является то, что выполнение скриптов не требует обращения к серверу. Например, можно сразу проверить правильность заполнения формы до отправки данных. Однако важно помнить, что средствами клиентских языков нельзя записывать данные непосредственно на сервер. Наиболее распространённым клиентским языком является JavaScript, созданный компанией Netscape совместно с Sun Microsystems. Кроме него, популярен и VBScript, а также современные технологии вроде AJAX, Adobe Flash и Microsoft Silverlight.

Серверные языки программирования предоставляют программистам гораздо больше возможностей. При обращении к веб-странице запрос сначала поступает на сервер, где запускаются необходимые программы, связанные с этой страницей. Лишь после обработки результат отправляется пользователю в виде готового файла. Такие файлы могут иметь расширения HTML, PHP, ASP, Perl, SSI, XML, DHTML, XHTML и другие.

1.2 HTML и XHTML

**HTML (HyperText Markup Language)** — это основной язык разметки документов во Всемирной паутине, который принят как стандарт. Большинство веб-страниц в интернете создаются именно с его помощью (или с использованием его расширенной версии — XHTML).

HTML позволяет управлять внешним видом текста и элементов на веб-странице. Можно изменять цвет, стиль, начертание и размер шрифта, создавать заголовки, абзацы, списки и выделять отдельные символы. Одной из ключевых возможностей HTML является создание гипертекстовых ссылок, которые позволяют переходить между страницами и различными частями одного документа.

Также HTML поддерживает формы для ввода данных (например, для отправки информации через сайт), отображение изображений, аудио и видеофайлов — как средствами браузера, так и с помощью внешних приложений вроде Flash или Java-апплетов.

HTML-документы строятся с помощью **тегов** — специальных служебных конструкций. Все теги записываются внутри угловых скобок < >, а для завершения большинства элементов требуется закрывающий тег с прямым слэшем (/), например: <html>...</html>. В HTML допускается использование как строчных, так и заглавных букв в названиях тегов (в отличие от XHTML).

**Пример базовой структуры HTML-документа:**

<HTML>

<HEAD>

<TITLE>Заголовок страницы</TITLE>

</HEAD>

<BODY>

Тело документа

</BODY>

</HTML>

Теги могут быть вложены друг в друга. Некоторые из них (например, <br>) являются самозакрывающимися и не требуют отдельного закрытия.

Атрибуты тегов позволяют задавать их свойства, например цвет текста, размер шрифта, ссылки на изображения. Атрибуты указываются внутри открывающего тега.

**Пример ссылки с атрибутом:**

<a href="http://www.google.com">Перейти на Google</a>

**Основные группы тегов HTML:**

Гиперссылки: <a href="URL" target="\_blank">Текст ссылки</a>, где target определяет, где откроется ссылка (в новом окне, в текущем и т.д.).

**Текстовые элементы:**

Заголовки <h1>…<h6> для выделения текста разного уровня важности.

Абзацы <p>, перенос строки <br>, горизонтальные линии <hr>.

Блоки и области: <div>, <span>, цитаты <blockquote>, текст с сохранением форматирования <pre>.

**Форматирование текста:**

Курсив <i>, жирный <b>, подчёркивание <u>, зачёркивание <s> или <strike>.

Элементы акцента <em>, <strong>.

Изменение размера шрифта <big>, <small>, подстрочный <sub> и надстрочный текст <sup>.

Изменение шрифта <font color="цвет" face="шрифт" size="размер">...</font>.

**Списки:**

Нумерованные <ol> и ненумерованные <ul> списки, элементы списка <li>.

**Изображения:**

Вставка картинок тегом <img src="путь" alt="описание" title="подсказка" width="ширина" height="высота" align="обтекание">.

**Таблицы:**

Создание таблиц через теги <table>, строки <tr>, заголовки столбцов <th>, ячейки <td>.

**Формы ввода данных:**

Создание форм <form>, текстовых полей <input>, больших текстовых областей <textarea>, выпадающих списков <select> и элементов списка <option>.

**Объекты и скрипты:**

Вставка мультимедийных объектов через <embed>, Java-апплетов через <applet>, сценариев через <script>.

**Особые символы HTML:** Некоторые символы нельзя напрямую использовать в коде страницы, например:

&amp; — амперсанд &

&lt; — символ «меньше» <

&gt; — символ «больше» >

**Обязательная структура любой HTML-страницы:**

Она начинается с тегов <html>...</html>, включает секцию заголовка <head>...</head>, и основное содержимое в секции <body>...</body>.

**XHTML (Extensible Hypertext Markup Language)** — это более строгая версия HTML. В XHTML все теги обязательно должны быть закрыты. Так же все теги и атрибуты пишутся только маленькими буквами. XHTML предъявляет более строгие требования к синтаксису (например, нельзя использовать необработанные знаки < и &). При ошибке браузер обязан остановить загрузку страницы. Еще стоит отметить, что XHTML поддерживает расширение языка через пользовательские теги и структуры.

Таким образом, HTML предлагает широкие возможности для форматирования информации на веб-страницах, а XHTML — обеспечивает строгую структуру и уменьшает количество ошибок при создании сайтов.

1.3 Dynamic HTML

**Dynamic HTML (DHTML)** — это общее название для технологий, объединяющих язык разметки HTML, каскадные таблицы стилей CSS, скриптовые языки программирования (чаще всего **JavaScript**) и объектную модель документа (**DOM**).

DHTML позволяет создавать интерактивные веб-страницы, обрабатывать пользовательские запросы и формировать динамическое содержимое без необходимости обращения к серверу или базе данных. С помощью DHTML можно реализовать интерфейсы перетаскивания объектов (Drag'n'Drop), а также создавать игры и другие интерактивные сервисы прямо в браузере пользователя.

Важным преимуществом DHTML является то, что для его работы требуется только браузер, способный обрабатывать HTML, CSS и JavaScript. Никаких дополнительных серверных компонентов не требуется.

1.3.1 Каскадные таблицы стилей

**Каскадные таблицы стилей (CSS)** — это технология, предназначенная для управления внешним видом элементов HTML-документа. С помощью CSS задаются такие характеристики, как шрифт, цвет, отступы, размеры и расположение элементов на странице.

Главная идея CSS заключается в разделении структуры документа (HTML) и его оформления (CSS). Благодаря этому подходу веб-разработчики могут легче изменять внешний вид страниц без необходимости вмешательства в их содержимое. Это также помогает уменьшить избыточность кода и облегчает поддержку сайта.

Стили можно подключать несколькими способами:

* **Встроенные стили** — добавляются прямо в HTML-документ внутри тега <style>:

<style type="text/css">

body {

color: green;

}

</style>

* **Внешние таблицы стилей** — подключаются отдельным файлом с помощью тега <link>:

<link rel="stylesheet" type="text/css" href="style.css" />

* **Inline-стили** — указываются непосредственно в атрибуте style HTML-элемента:

<p style="font-size: 21px; color: green;">Текст</p>

Также каждый браузер имеет собственные встроенные стили по умолчанию.

Поскольку одному элементу могут соответствовать несколько стилей, CSS использует принцип каскадности: более специфичные и более поздние правила имеют больший приоритет.

Важное преимущество внешних CSS-файлов — ускорение загрузки страниц. Браузер загружает таблицу стилей один раз и затем использует её из кэша при переходе между страницами сайта.

Однако CSS имеет и свои недостатки, основной из которых — разное отображение в разных браузерах из-за особенностей их работы с CSS-правилами.

1.3.2 JavaScript

**JavaScript** — это объектно-ориентированный скриптовый язык программирования, предназначенный для создания динамического поведения на веб-страницах.

**Для размещения JavaScript-кода используется тег <script>:**

<script type="text/javascript">

alert('Hello, World!');

</script>

**Также скрипты могут быть вынесены в отдельные файлы и подключаться через атрибут src:**

<script type="text/javascript" src="script.js"></script>

**Благодаря JavaScript разработчики могут:**

* автоматизировать заполнение форм,
* скрывать или показывать элементы,
* управлять поведением веб-приложений на стороне клиента.

Язык довольно прост в освоении и предлагает широкие возможности для создания интерактивных интерфейсов.

Что касается инструментов для разработки на JavaScript, то нет какого-то одного "лучшего" редактора. Многие используют модуль для **Eclipse** или встроенные отладчики в браузерах, которые позволяют удобно проверять и тестировать скрипты прямо во время работы.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Описание предметной области

**Предметная область** проекта охватывает разработку онлайн-информационной системы для аэропорта, предназначенной для предоставления пассажирам актуальной информации о рейсах, услугах и навигации по территории аэропорта. Цель — создание удобного и функционального ресурса, способствующего повышению качества обслуживания и информированности пассажиров.

**Архитектура сайта** включает логичную структуру представления информации и продуманную навигацию, обеспечивающую быстрый доступ к основным разделам. Грамотно спроектированная структура сокращает время поиска данных и способствует улучшению пользовательского опыта, повышая лояльность и эффективность взаимодействия с сервисами аэропорта.

При проектировании учитывались как восприятие информации пользователями, так и требования к продвижению ресурса в цифровой среде. Ключевыми элементами структуры стали: расписание рейсов, информация об услугах и терминалах, формы поиска и обратной связи.

В рамках реализации использовались следующие технологии:

* **HTML (HyperText Markup Language)** — базовый язык разметки, применяемый для создания структуры веб-страниц, содержащих сведения о рейсах, терминалах и сервисах.
* **CSS (Cascading Style Sheets)** — язык описания внешнего вида элементов сайта, позволяющий реализовать адаптивный и визуально единый интерфейс: цветовую схему, типографику, компоновку страниц.
* **JavaScript** — язык программирования для создания динамических и интерактивных элементов. Используется для обновления информации о рейсах в реальном времени, работы форм и повышения удобства взаимодействия с сайтом без перезагрузки страниц.

2.2 Создание окружения для проекта

В качестве среды разработки выбран редактор Visual Studio Code. Для проекта создана папка с названием “Visuals”, в которой будут размещаться все необходимые файлы сайта. Поскольку сайт будет состоять из шести страниц, первым создан файлы **index.html**. Далее были созданы две дополнительные папки: **css** и **images**. В папке **css** размещён файл стилей под названием **style.css**, а в папку **images** будут добавляться все изображения, используемые на сайте. Рисунок 2.1 показывает папку со всеми файлами

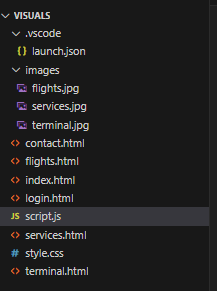


Рис. 2.1

2.3 Верстка HTML-страницы index.html

Откроем файл index.html и пропишем стандартную структуру HTML5:

<!DOCTYPE html>

<html lang="ru">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Главная — Аэропорт Онлайн</title>

  <link rel="stylesheet" href="style.css">

</head>

* **<!DOCTYPE html>** сообщает браузеру, что документ написан на HTML5.
* **<html lang="ru">** задаёт язык страницы.
* **<meta charset="UTF-8">** гарантирует корректное отображение кириллицы.
* **<meta name="viewport" content="width=device-width, initial-scale=1.0">** позволяет адаптировать страницу под экраны разных устройств.
* **<link rel="stylesheet" href="style.css">** подключает ваш CSS.

**Навигация (navbar)**

    <header>

    <div class="header-container">

      <div class="logo"><h1>Аэропорт Онлайн</h1> </div>

      <nav>

        <ul>

          <li><a href="index.html" class="active">Главная</a></li>

          <li><a href="flights.html">Расписание рейсов</a></li>

          <li><a href="contact.html">Контакты</a></li>

          <li><a href="login.html"> Войти </a></li>

        </ul>

      </nav>

    </div>

  </header>

* <ul> + <li> + <a> — пункты меню, которые при клике переходят на нужные страницы.

**Баннер с поиском рейса**

    <section class="welcome-banner">

      <h1>Добро пожаловать в аэропорт "Онлайн"</h1>

      <p>Получите актуальную информацию о рейсах, услугах и сервисах аэропорта</p>

      <div class="search-flight">

        <select>

          <option value="departure">Вылет</option>

          <option value="arrival">Прилет</option>

        </select>

        <input type="text" placeholder="Номер рейса или направление">

        <button>Найти рейс</button>

      </div>

    </section>

Сразу после шапки добавляем секцию с ключевым сообщением и формой поиска рейсов:

* Класс .welcome-banner задаёт фоновую картинку или цвет.
* Внутри <div class= “search flight” , можно выбрать между вылетом и прилетом, так же присутсвует поисковая строка для получения информации о рейсе

**Карточки-переходы**

Ниже располагаем блок с карточками для быстрого доступа к ключевым разделам:

    <section class="card-container">

      <div class="card">

        <img src="images/terminal.jpg" alt="Терминал">

        <h3>Наш терминал</h3>

        <p>Обзор нашего современного терминала.</p>

        <a href="terminal.html">Подробнее</a>

      </div>

      <div class="card">

        <img src="images/flights.jpg" alt="Расписание рейсов">

        <h3>Расписание рейсов</h3>

        <p>Узнайте расписание рейсов прямо сейчас.</p>

        <a href="flights.html">Подробнее</a>

      </div>

      <div class="card">

        <img src="images/services.jpg" alt="Наши услуги">

        <h3>Наши услуги</h3>

        <p>Мы предлагаем широкий спектр услуг.</p>

        <a href="services.html">Подробнее</a>

      </div>

    </section>

* Каждая карточка — это ссылка <a> с изображением, заголовком <h3> и описанием <p>.
* Класс .card-container выстраивает их в ряд.

**Footer**

В конце файла добавляем футер:

  <footer>

    <p>&copy; 2025 Аэропорт Онлайн. Все права защищены.</p>

  </footer>

2.4 Верстка HTML-страницы flights.html

**Структура и назначение страницы flights.html**

Файл flights.html представляет собой отдельную страницу сайта, предназначенную для отображения расписания вылетов и прилётов в аэропорту. Она оформлена в соответствии с общей стилистикой проекта и включает как статические элементы интерфейса, так и динамически обновляемые данные о рейсах.

**Основная структура документа**

**Основной контент (<main>)**

Внутри <main> размещены две ключевые секции:

**Секция "Вылеты"**

  <main>

    <section>

      <h2>Вылеты</h2>

      <table>

        <thead>

          <tr>

            <th>Рейс</th><th>Направление</th><th>Время</th><th>Гейт</th><th>Статус</th>

          </tr>

        </thead>

        <tbody>

          <tr>

            <td>SU123</td><td>Москва</td><td>12:30</td><td>A5</td>

            <td class="status scheduled">По расписанию</td>

          </tr>

          <tr>

            <td>DP456</td><td>Рим</td><td>15:00</td><td>B2</td>

            <td class="status delayed">Задерживается</td>

          </tr>

          <tr>

            <td>UT789</td><td>Сеул</td><td>18:45</td><td>C1</td>

            <td class="status cancelled">Отменён</td>

          </tr>

        </tbody>

      </table>

    </section>

* Заголовок секции: **"Вылеты"**.
* Таблица содержит информацию о рейсах на вылет: номер рейса, направление, время вылета, номер выхода на посадку (гейт) и текущий статус рейса.
* Статусы рейсов (По расписанию, Задерживается, Отменён) дополнительно стилизованы с использованием CSS-классов (scheduled, delayed, cancelled).

**Секция "Прилёты"**

    <section>

      <h2>Прилёты</h2>

      <table>

        <thead>

          <tr>

            <th>Рейс</th><th>Из</th><th>Время</th><th>Гейт</th><th>Статус</th>

          </tr>

        </thead>

        <tbody>

          <tr>

            <td>BA321</td><td>Баку</td><td>11:20</td><td>D3</td>

            <td class="status scheduled">По расписанию</td>

          </tr>

          <tr>

            <td>AF654</td><td>Пекин</td><td>14:10</td><td>E4</td>

            <td class="status landed">Приземлился</td>

          </tr>

          <tr>

            <td>LH987</td><td>Франкфурт</td><td>17:05</td><td>F1</td>

            <td class="status enroute">В пути</td>

          </tr>

        </tbody>

      </table>

    </section>

**Описание:**

* Заголовок секции: **"Прилёты"**.
* Таблица содержит информацию о прибывающих рейсах: номер рейса, пункт вылета, время прибытия, номер выхода и текущий статус рейса.
* Используются аналогичные классы для статусов (scheduled, landed, enroute).

**Оформление статусов рейсов**

Каждому статусу рейса присваивается свой CSS-класс, который определяет цвет фона и внешний вид ячейки.  
Это позволяет пользователю быстро ориентироваться в расписании:

* **По расписанию** — зелёный цвет (.scheduled);
* **Задерживается** — жёлтый цвет (.delayed);
* **Отменён** — красный цвет (.cancelled);
* **Приземлился** — голубой цвет (.landed);
* **В пути** — фиолетовый цвет (.enroute).

2.5 Верстка HTML-страницы contact.html

**Структура и назначение страницы contact.html**

Страница contact.html предназначена для предоставления пользователям контактной информации, отображения местоположения аэропорта на карте, а также для организации обратной связи посредством специальной формы.

**Основной контент страницы (<main>)**

Содержимое страницы разделено на несколько логических секций (<section>), каждая из которых выполняет свою функцию.

**Контактная информация**

    <section>

      <h2>Контактная информация</h2>

      <p><strong>Адрес:</strong> г. Баку</p>

      <p><strong>Телефон:</strong> +994 55 427 10 90</p>

      <p><strong>Email:</strong> info@airport.az</p>

    </section>

* Содержит основные данные для связи: адрес, телефон и электронную почту.
* Важные слова (Адрес, Телефон, Email) выделены тегом <strong>, что делает их более заметными для пользователя.

**Часы работы**

    <section>

      <h2>Часы работы</h2>

      <p>Понедельник–Воскресенье: 06:00–23:00</p>

    </section>

* Представлены часы работы организации, что позволяет пользователю заранее знать время приёма и обслуживания.

**Карта местоположения**

    <section>

      <h2>Где мы на карте</h2>

      <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d15486.783373727614!2d50.049444349961675!3d40.46035108650008!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x4030606eefea1ed1%3A0x41cc40173c5f5edc!2z0JzQtdC20LTRg9C90LDRgNC-0LTQvdGL0Lkg0LDRjdGA0L7Qv9C-0YDRgiDQuNC80LXQvdC4INCT0LXQudC00LDRgNCwINCQ0LvQuNC10LLQsA!5e1!3m2!1sru!2saz!4v1745689723128!5m2!1sru!2saz" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade">

      </iframe>

    </section>

* Используется встроенный фрейм <iframe>, через который подключается карта Google Maps.
* Свойства карты:
  + width="600" и height="450" — размеры карты на странице.
  + border:0; — убирает рамку вокруг карты.
  + allowfullscreen и loading="lazy" — улучшают производительность и позволяют отображать карту полноэкранно при необходимости.
* Карта отображает точное местоположение аэропорта.

**Социальные сети**

    <section>

      <h2>Мы в социальных сетях</h2>

      <p>

        <a href="#">Facebook</a> |

        <a href="#">Instagram</a> |

        <a href="#">Twitter</a>

      </p>

    </section>

Представлены ссылки на страницы аэропорта в социальных сетях. Ссылки разделены вертикальными чертами (|) для удобства восприятия.

**Форма обратной связи**

    <section>

      <h2>Форма обратной связи</h2>

      <form>

        <label for="name">Имя:</label><br>

        <input type="text" id="name" name="name" required><br>

        <label for="email">Email:</label><br>

        <input type="email" id="email" name="email" required><br>

        <label for="message">Сообщение:</label><br>

        <textarea id="message" name="message" rows="5" required></textarea><br>

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

      </form>

Форма включает следующие элементы:

* **Поле ввода имени** (<input type="text" id="name" required>);
* **Поле ввода электронной почты** (<input type="email" id="email" required>);
* **Текстовое поле для сообщения** (<textarea id="message" rows="5" required>);
* **Кнопка отправки** (<button type="submit">Отправить</button>).

Атрибут required делает все поля обязательными для заполнения. Такая форма позволяет пользователям отправлять свои вопросы или предложения напрямую через сайт.

2.6 Верстка HTML-страницы login.html

**Структура и назначение страницы login.html**

Страница login.html предназначена для организации процесса авторизации пользователей на сайте "Аэропорт Онлайн". Она представляет собой компактную форму входа с минималистичным дизайном и поддержкой клиентской валидации данных.

**Основная структура документа**

**Основной контент страницы (<main>)**

    <main>

        <div class="login-form-container">

            <h2>Вход в систему</h2>

            <form id="login-form">

                <label for="username">Логин</label>

                <input type="text" id="username" name="username" required>

                <label for="password">Пароль</label>

                <input type="password" id="password" name="password" required>

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

            </form>

            <p>Нет аккаунта? <a href="register.html">Зарегистрируйтесь</a></p>

        </div>

    </main>

* **Блок .login-form-container**  
  — визуально выделенная область формы входа, стилизованная с помощью CSS (background-color: #fff;, скругление углов, тень).
* **Заголовок формы (<h2>)**  
  — "Вход в систему" — информирует пользователя о назначении формы.
* **Форма авторизации (<form id="login-form">)**  
  Включает три основных элемента:
  + **Поле для ввода логина** (<input type="text">).
  + **Поле для ввода пароля** (<input type="password">).
  + **Кнопку отправки** (<button type="submit">Войти).

Каждый элемент формы сопровождается меткой (<label>), что повышает доступность сайта для пользователей и адаптивных технологий (например, экранных читалок). Поля ввода имеют атрибут required, что позволяет браузеру автоматически проверять, чтобы они не оставались пустыми.

**Подключение скрипта JavaScript**

<script src="script.js"></script>

Подключение внешнего файла JavaScript (script.js) позволяет добавить интерактивность форме авторизации.Данный скрипт обрабатывает отправку формы, выполняет валидацию данных и перенаправляет пользователя после успешного входа.

2.7 Использование стилей CSS

**Общие стили оформления**

В начале оформления сайта устанавливаются базовые стили для всего документа. Для этого задаются следующие свойства:

body {

  font-family: 'Segoe UI', 'Roboto', 'Arial', sans-serif;

  margin: 0;

  padding: 0;

  background-color: #f8f9fa;

  color: #333333;

}

Шрифт устанавливается семейством 'Segoe UI', 'Roboto' или 'Arial', в зависимости от доступности на устройстве пользователя. Также убираются стандартные отступы и поля браузера (margin и padding), задаётся светлый фон и основной цвет текста.

**Оформление шапки сайта**

Для элемента <header> задаются стили, обеспечивающие его визуальное отделение от остальной части страницы:

header {

  background-color: #0056b3;

  padding: 10px 20px;

  box-shadow: 0 2px 10px rgba(0,0,0,0.1);

}

Таким образом, используется насыщенный синий цвет фона, а также применяется лёгкая тень для создания эффекта возвышенности.

Контейнер внутр. и шапки оформляется следующим образом:

.header-container {

  max-width: 1200px;

  margin: 0 auto;

  display: flex;

  align-items: center;

  justify-content: space-between;

}

Это позволяет ограничить максимальную ширину содержимого, выровнять элементы по центру и равномерно распределить их пространство.

Логотип сайта выделяется белым цветом и увеличенным размером шрифта:

.logo h1 {

  color: white;

  font-size: 28px;

  margin: 0;

}

**Стилизация меню навигации**

Навигационное меню реализуется с использованием списка <ul>. Убираются стандартные маркеры и обеспечивается горизонтальное выравнивание элементов:

nav ul {

  list-style: none;

  display: flex;

  margin: 0;

  padding: 0;

}

Элементы списка получают отступы слева:

nav ul li {

  margin-left: 25px;

}

Ссылки в меню оформляются следующим образом:

nav ul li a {

  text-decoration: none;

  color: white;

  font-weight: 600;

  font-size: 18px;

  padding: 8px 12px;

  border-radius: 6px;

  transition: background-color 0.3s, color 0.3s;

}

Таким образом обеспечивается отсутствие подчёркивания ссылок и добавляется плавный переход цвета при наведении.

При наведении курсора или активном состоянии ссылки происходит инверсия цвета:

nav ul li a:hover,

nav ul li a.active {

  background-color: white;

  color: #0056b3;

}

В результате оглавление сайта будет выглядить следущим образом (рис. 2.2) :

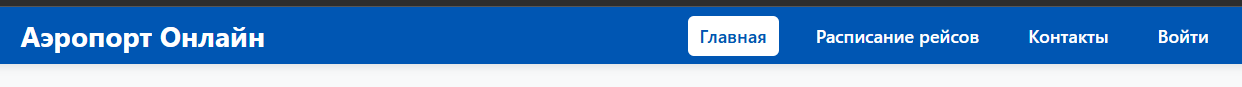


Рис. 2.2

**Баннер приветствия**

Для оформления приветственного баннера используется следующий CSS-код:

.welcome-banner {

  background-color: #666;

  color: white;

  padding: 50px 20px;

  text-align: center;

  border-radius: 10px;

  margin-bottom: 40px;

}

Данный элемент представляет собой блок с фоном тёмно-серого цвета и белым текстом, выравненным по центру. Устанавливаются значительные внутренние отступы для увеличения визуального пространства, а также закругление углов посредством border-radius: 10px. Отступ снизу (margin-bottom) отделяет баннер от следующего содержимого страницы.

**Форма поиска авиарейсов**

Форма поиска авиарейсов (.search-flight) реализована с использованием следующего стиля:

.search-flight {

  background: white;

  padding: 20px;

  display: inline-flex;

  gap: 10px;

  border-radius: 6px;

  align-items: center;

}

Контейнер формы представлен как инлайн-флекс контейнер, в котором элементы (поля ввода и кнопка поиска) располагаются в одну строку с промежутками (gap) между ними. Форма имеет светлый фон, скруглённые углы и выравнивание по вертикали за счёт align-items: center.

Поля ввода и выпадающие списки внутри формы имеют следующие стили:

.search-flight select,

.search-flight input {

  padding: 10px;

  font-size: 1em;

  border: 1px solid #ccc;

  border-radius: 5px;

}

Кнопка поиска выделена синим цветом и имеет эффект затемнения при наведении:

.search-flight button {

  background-color: #0d47a1;

  color: white;

  border: none;

  padding: 10px 20px;

  font-size: 1em;

  border-radius: 5px;

  cursor: pointer;

}

.search-flight button:hover {

  background-color: #093170;

}

Таким образом, форма поиска органично вписывается в общую концепцию сайта, поддерживая фирменный цветовой стиль.

Конечный вид баннера приветсвия и поиска рейсов можно увидеть на рисунке .2.3:

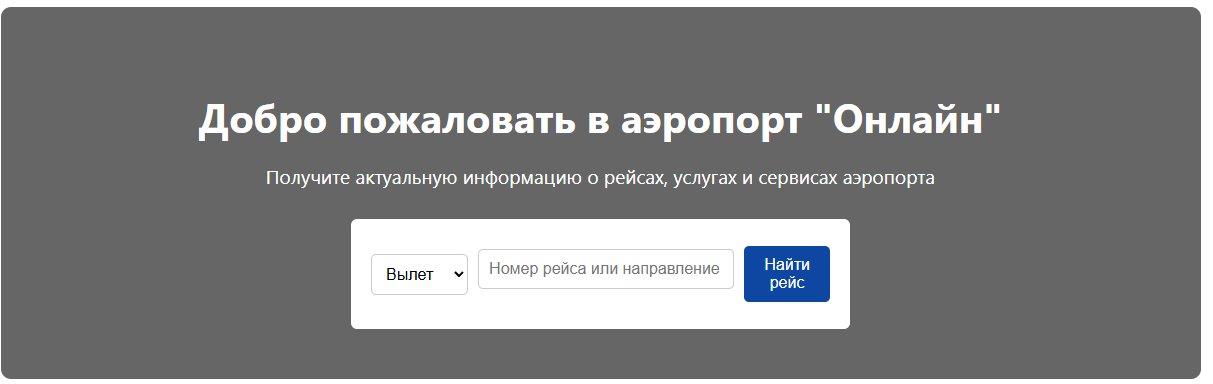


Рис.2.3

**Основной контент страницы**

Основной блок сайта стилизован в виде "карточки":

main {

  padding: 50px 20px;

  max-width: 1200px;

  margin: 30px auto;

  background-color: white;

  border-radius: 12px;

  box-shadow: 0 4px 12px rgba(0,0,0,0.08);

}

Это обеспечивает аккуратное выделение основного контента за счёт белого фона, скруглённых углов и лёгкой тени.

**Оформление карточек продуктов**

Карточки расположены в ряд с использованием технологии Flexbox:

.card-container {

  display: flex;

  justify-content: space-between;

  gap: 20px;

  margin-top: 40px;

}

Отдельная карточка имеет следующий стиль:

.card {

  background-color: white;

  border-radius: 10px;

  box-shadow: 0 2px 15px rgba(0, 0, 0, 0.1);

  width: 30%;

  overflow: hidden;

  transition: transform 0.3s, box-shadow 0.3s;

}

Добавлены скругления, эффект лёгкого подъема при наведении и плавная анимация трансформации.

Изображение внутри карточки адаптируется по размеру:

.card img {

  width: 100%;

  height: 200px;

  object-fit: cover;

}

Тексты в карточках имеют компактные отступы и аккуратное цветовое оформление.

Кнопка внутри карточки выделяется цветом и изменяется при наведении:

.card a {

  display: block;

  text-align: center;

  padding: 12px;

  background-color: #0056b3;

  color: white;

  text-decoration: none;

  border-radius: 0 0 10px 10px;

  transition: background-color 0.3s;

}

.card:hover {

  transform: scale(1.05);

  box-shadow: 0 5px 20px rgba(0, 0, 0, 0.2);

}

Конечный вид представлен на рисунке 2.4 :

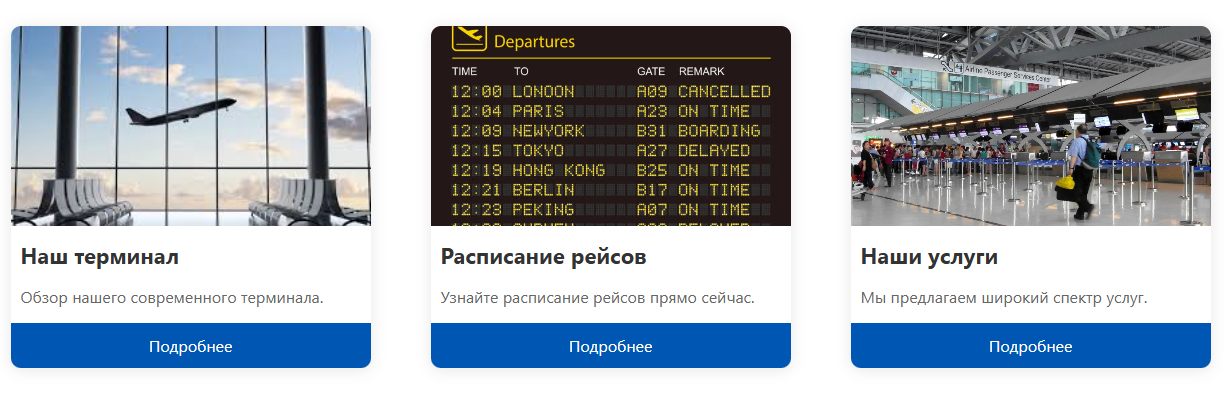


Рис.2.4

**Оформление таблиц данных**

Таблицы, находящиеся в разделе ”Расписание рейсов”, являются важным элементом для структурированного отображения информации на сайте. Для повышения их читаемости и визуальной привлекательности были применены следующие стили CSS.

**Общие стили для таблицы**

table {

  width: 100%;

  border-collapse: collapse;

  margin-top: 30px;

}

* border-collapse: collapse; — объединяет границы ячеек, устраняя двойные линии между ними. Без этого свойства у таблицы могли бы отображаться двойные границы между ячейками, что ухудшает визуальное восприятие данных.

**Стилизация границ таблицы**

table, th, td {

  border: 1px solid #dee2e6;

}

* Данное правило применяется одновременно к самой таблице (table), заголовкам столбцов (th) и обычным ячейкам (td).
* border: 1px solid #dee2e6; — задаёт тонкую, одинарную границу серого цвета вокруг всех элементов таблицы.  
  Цвет выбран светло-серым (#dee2e6), чтобы визуально отделить ячейки друг от друга, сохраняя при этом лёгкость общего дизайна без тяжёлых визуальных перегородок.

**Оформление заголовков таблицы**

th {

  background-color: #0056b3;

  color: white;

  padding: 15px;

  font-size: 16px;

  text-align: center;

}

**Оформление содержимого ячеек таблицы**

td {

  padding: 15px;

  text-align: center;

  background-color: #f8f9fa;

  transition: background-color 0.3s;

}

* padding: 15px; — задаёт внутренние отступы в ячейках, что делает текст менее сжатым и увеличивает удобство восприятия данных.
* transition: background-color 0.3s; — добавляет плавный переход при изменении цвета фона ячейки, улучшая эстетическое восприятие интерфейса.

**Эффект при наведении на ячейку**

td:hover {

  background-color: #e2e6ea;

}

* При наведении курсора мыши на любую ячейку таблицы цвет её фона меняется на чуть более тёмный оттенок серого (#e2e6ea).
* Этот эффект "подсветки" делает взаимодействие с таблицей более динамичным и наглядным: пользователь интуитивно видит, на какую строку он навёлся.
* Благодаря свойству transition (заданному ранее), смена цвета происходит плавно, что создаёт современный и аккуратный пользовательский опыт.

Особое внимание уделено стилизации таблиц, отображающих статусы рейсов. Для ячеек со статусами (.status) предусмотрено использование цвето вых обозначений в зависимости от состояния рейса:

.status {

  color: white;

  font-weight: bold;

  border-radius: 4px;

  padding: 4px 8px;

}

**Статусы окрашены в цвета соответсвующие их состоянию:**

.status.scheduled {

  background-color: #28a745;

}

.status.delayed {

  background-color: #ffc107;

  color: #333;

}

.status.cancelled {

  background-color: #dc3545;

}

.status.landed {

  background-color: #17a2b8;

}

.status.enroute {

  background-color: #6f42c1;

}

Таким образом, пользователь может мгновенно визуально определить состояние рейса благодаря цветовой кодировке.

Результат изображен на рисунке 2.5:

A screenshot of a computer

AI-generated content may be incorrect.

Рис.2.5

**Стилизация форм ввода данных**

Формы ввода данных на веб-странице играют важную роль в процессе взаимодействия пользователя с сайтом. Для повышения удобства заполнения и эстетической привлекательности формы были оформлены следующим образом.

**Стилизация полей ввода и текстовых областей**

input, textarea {

  width: 100%;

  padding: 15px;

  margin-top: 10px;

  margin-bottom: 20px;

  border: 1px solid #ced4da;

  border-radius: 8px;

  font-size: 16px;

}

* width: 100%; — задаёт ширину элементов на всю доступную ширину контейнера, что позволяет полям адаптироваться под размеры формы и обеспечивать симметричный внешний вид.
* padding: 15px; — добавляет внутренние отступы внутри полей ввода и текстовых областей, увеличивая площадь активной зоны для ввода текста и улучшая эргономику.
* margin-top: 10px; margin-bottom: 20px; — устанавливает внешние отступы, отделяя поля друг от друга и создавая логичную структуру формы.
* border: 1px solid #ced4da; — задаёт светло-серую границу вокруг полей, обеспечивая чёткое, но ненавязчивое выделение границ полей ввода.
* border-radius: 8px; — делает углы скруглёнными, что визуально смягчает внешний вид формы и соответствует современным тенденциям веб-дизайна.
* font-size: 16px; — обеспечивает оптимальный размер текста для ввода, способствующий удобочитаемости на различных устройствах.

Таким образом, поля ввода и текстовые области становятся интуитивно понятными и удобными для использования.

**Эффект при фокусе на полях ввода**

input:focus, textarea:focus {

  outline: none;

  border-color: #0056b3;

  box-shadow: 0 0 8px rgba(0,86,179,0.2);

}

**Подробное объяснение:**

* outline: none; — убирает стандартную браузерную подсветку при фокусе на поле, заменяя её пользовательским стилем.
* border-color: #0056b3; — изменяет цвет границы поля ввода на насыщенный синий, который соответствует общей цветовой гамме сайта.
* box-shadow: 0 0 8px rgba(0,86,179,0.2); — добавляет мягкую синюю тень вокруг поля ввода. Это визуально подчёркивает активное поле, привлекая внимание пользователя, но без излишней агрессивности.

Таким образом, пользователь сразу видит, в каком поле он находится, а форма становится более отзывчивой и приятной для работы.

**Оформление кнопок**

Кнопки являются важнейшими элементами управления на сайте, отвечающими за подтверждение действий пользователей (например, отправку формы).

**Базовое оформление кнопок**

button {

  background-color: #0056b3;

  color: white;

  padding: 15px 30px;

  border: none;

  border-radius: 8px;

  cursor: pointer;

  font-size: 16px;

  transition: background-color 0.3s, transform 0.2s;

}

* обеспечивая высокий контраст и хорошую читаемость.
* padding: 15px 30px; — создаёт внутренние отступы вокруг текста кнопки, увеличивая активную область для нажатия и делая кнопку визуально более крупной.
* border-radius: 8px; — скругляет углы кнопки, гармонируя с оформлением полей ввода.
* cursor: pointer; — изменяет курсор при наведении на кнопку на "указатель" (руку), что интуитивно подсказывает пользователю о возможности взаимодействия.
* transition: background-color 0.3s, transform 0.2s; — обеспечивает плавность анимации при наведении, улучшая пользовательский опыт.

**Анимация кнопок при наведении**

button:hover {

  background-color: #004094;

  transform: translateY(-2px);

}

* background-color: #004094; — при наведении цвет фона кнопки становится немного темнее. Это создаёт эффект интерактивности и помогает пользователю понять, что элемент активен.
* transform: translateY(-2px); — при наведении кнопка слегка приподнимается вверх на 2 пикселя. Это имитирует эффект "нажатия" или "отзывчивости", делая интерфейс более живым и современным.

Таким образом, анимация кнопок обеспечивает не только визуальное улучшение интерфейса, но и способствует интуитивному взаимодействию пользователей с сайтом.

**Блок авторизации**

Блок авторизации представлен отдельной секцией с собственным оформлением:

**Базовое оформление контейнера формы**

.login-form-container {

  max-width: 400px;

  margin: 50px auto;

  padding: 30px;

  background-color: #ffffff;

  border-radius: 12px;

  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);

}

* **margin: 50px auto;** — задаёт автоматические отступы слева и справа (auto) для центрирования формы по горизонтали, а также верхний и нижний отступ в 50 пикселей для отделения формы от других элементов страницы.
* **box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);** — добавляет лёгкую тень вокруг формы, создавая эффект "приподнятости" элемента над остальной частью страницы.

**Стилизация заголовка формы**

.login-form-container h2 {

  text-align: center;

  color: #0056b3;

  margin-bottom: 20px;

}

**Стилизация меток (label)**

.login-form-container label {

  font-size: 16px;

  margin-bottom: 8px;

  color: #333;

  display: block;

}

* **display: block;** — делает метку блочным элементом, позволяя корректно размещать её над полем ввода.

**Оформление полей ввода**

.login-form-container input {

  width: 100%;

  padding: 15px;

  margin-bottom: 20px;

  border: 1px solid #ced4da;

  border-radius: 8px;

  font-size: 16px;

  color: #333;

}

* **border: 1px solid #ced4da;** — задаёт светло-серую рамку вокруг поля, визуально выделяя область для ввода.
* **border-radius: 8px;** — скругляет углы полей, что соответствует общей стилистике контейнера.
* **color: #333;** — устанавливает тёмно-серый цвет текста, обеспечивая комфортное восприятие информации.

**Анимация полей ввода при фокусе**

.login-form-container input:focus {

  outline: none;

  border-color: #0056b3;

  box-shadow: 0 0 8px rgba(0,86,179,0.2);

}

* **outline: none;** — убирает стандартную браузерную обводку вокруг активного поля ввода.
* **box-shadow: 0 0 8px rgba(0,86,179,0.2);** — добавляет лёгкую синюю тень вокруг активного поля, создавая эффект интерактивности и улучшая пользовательский опыт.

**Оформление кнопки авторизации**

.login-form-container button {

  width: 100%;

  padding: 15px;

  background-color: #0056b3;

  color: white;

  border: none;

  border-radius: 8px;

  cursor: pointer;

  font-size: 16px;

  transition: background-color 0.3s, transform 0.2s;

}

* **cursor: pointer;** — при наведении курсор меняется на "указатель" (руку), что визуально подсказывает пользователю о возможности нажатия.
* **transition: background-color 0.3s, transform 0.2s;** — добавляет плавную анимацию для фона и эффекта подъёма при наведении.

**Эффект при наведении:**

.login-form-container button:hover {

  background-color: #004094;

  transform: translateY(-2px);

}

* При наведении фон кнопки становится темнее (#004094), а сама кнопка визуально приподнимается вверх на 2 пикселя (translateY(-2px)), что делает интерфейс более динамичным и отзывчивым.

**Оформление вспомогательных ссылок в форме**

.login-form-container p {

  text-align: center;

  margin-top: 20px;

  color: #333;

}

.login-form-container p a {

  color: #0056b3;

  text-decoration: none;

}

.login-form-container p a:hover {

  text-decoration: underline;

}

* **Абзац (p)** выравнивается по центру и отделяется от кнопки с помощью верхнего отступа.
* **Ссылки (a)** окрашены в фирменный синий цвет и при наведении подчёркиваются (text-decoration: underline), что традиционно указывает на возможность перехода.

Результатиллюстрирован на рисунке 2.6**:**

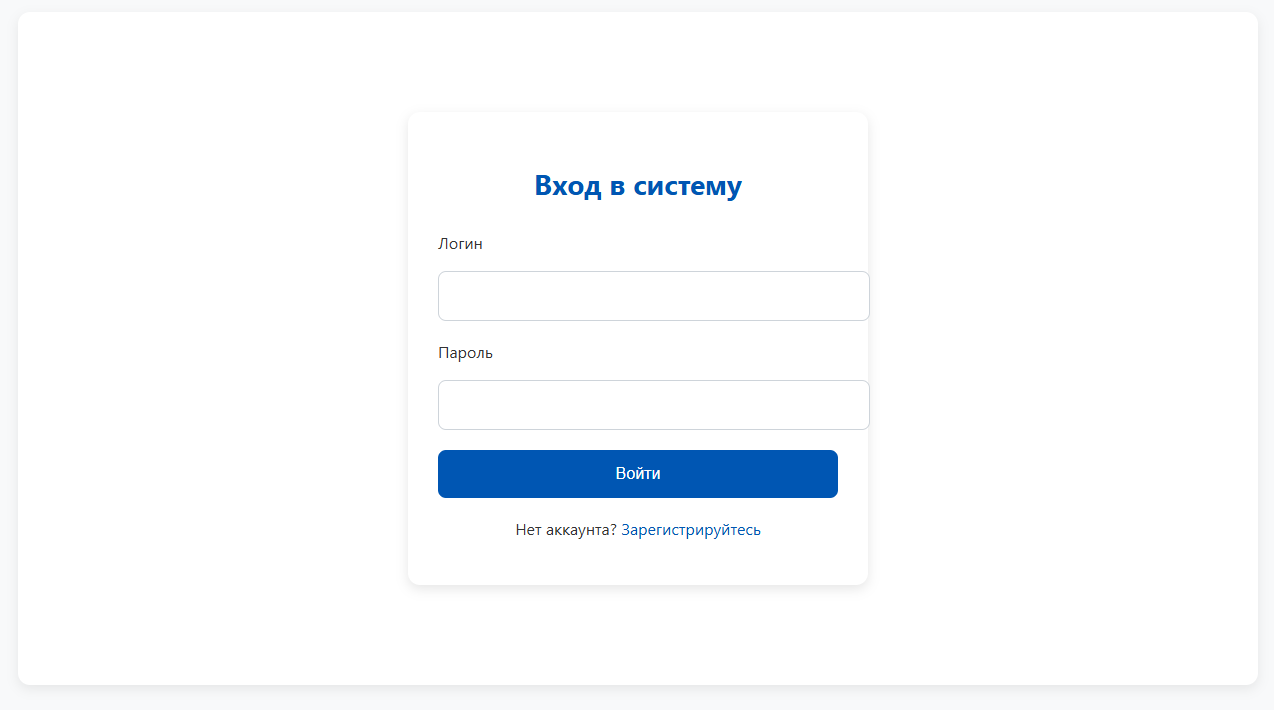


Рис.2.6

**Секции услуг**

Секции, описывающие предоставляемые услуги, оформлены следующим образом:

.services-section {

  margin-top: 20px;

}

.services-section h2 {

  font-size: 24px;

  color: #0056b3;

}

.services-section p {

  font-size: 16px;

  color: #555;

  line-height: 1.6;

}

.services-list {

  list-style-type: none;

  padding: 0;

  margin: 0;

}

.services-list li {

  font-size: 16px;

  color: #333;

  margin-bottom: 10px;

}

Таким образом, текстовый контент в секциях имеет стандартные отступы, оптимальную читаемость и выстроен в логическую структуру. Сами списки услуг оформлены без стандартных маркеров (list-style-type: none).

Конечный вид можно увидеть на рисунке 2.7 **:**

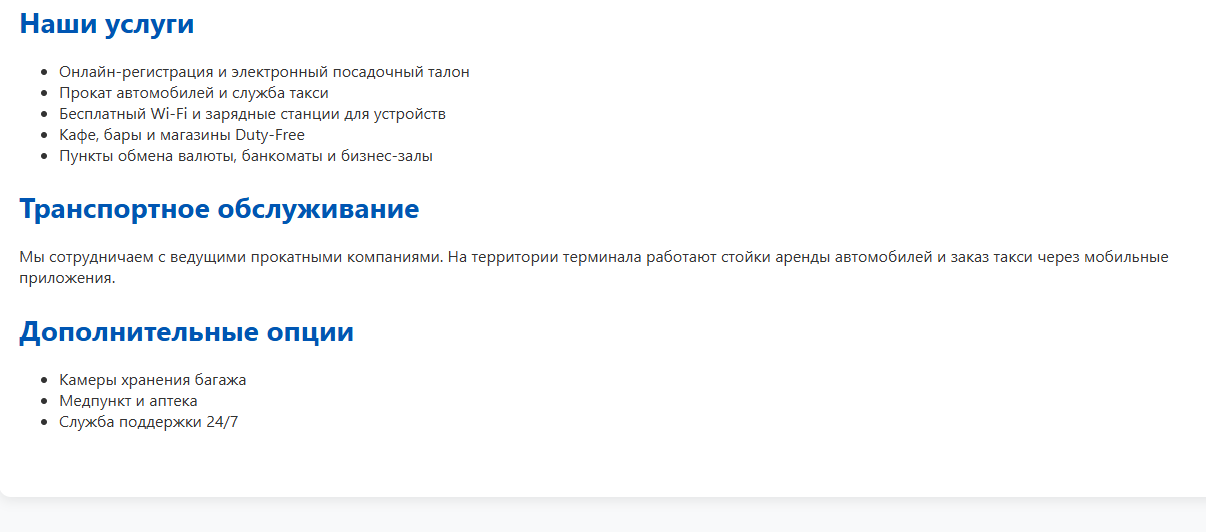


Рис.2.7

**Оформление секции "Терминал"**

На веб-странице важное место занимает секция, посвящённая описанию терминалов (например, терминалов аэропорта, вокзала или торгового центра). Эта секция предназначена для представления текстовой информации и соответствующих иллюстраций в структурированном и эстетичном виде.

Оформление данной секции в проекте реализовано следующим образом.

**Стилизация текстового блока описания терминала**

.terminal-description {

  margin-top: 20px;

}

.terminal-description h2 {

  font-size: 24px;

  color: #0056b3;

}

**Таким образом**, заголовок ясно выделяется и привлекает внимание пользователя.

**Основной текст описания терминала:**

.terminal-description p {

  font-size: 16px;

  line-height: 1.6;

  color: #555;

}

* line-height: 1.6; — увеличивает межстрочный интервал, что значительно улучшает читаемость абзацев, особенно на мобильных устройствах.

Таким образом, текст в секции "Терминал" оформлен так, чтобы быть легко читаемым и приятным для глаз.

**Стилизация списков описания терминала:**

.terminal-description ul {

  list-style-type: square;

  margin-left: 20px;

}

.terminal-description li {

  font-size: 16px;

  color: #333;

  margin-bottom: 8px;

}

* list-style-type: square; — задаёт для маркированного списка квадратные маркеры, что визуально отличает списки терминала от стандартных круглых маркеров HTML и делает их более строгими.
* margin-bottom: 8px; — добавляет небольшой отступ между элементами списка, улучшая визуальное восприятие и облегчая восприятие информации пользователем.

Результат предоставлен на рисунку 2.8

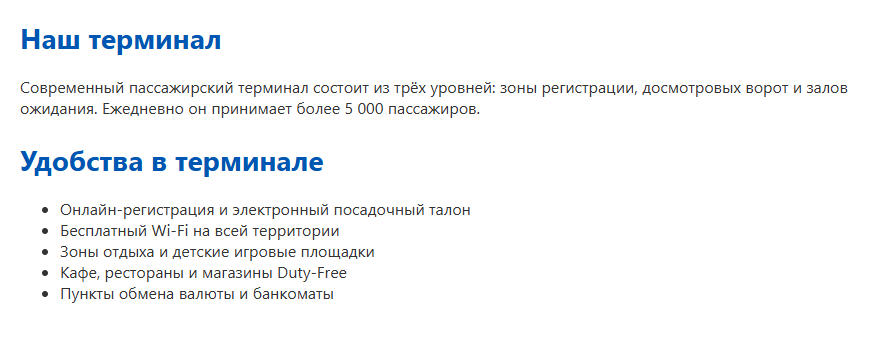


Рис. 2.8

**Оформление подвала сайта (footer)**

<footer> выполняет важную роль в структуре веб-страницы: он содержит дополнительную информацию, ссылки на условия использования, контакты, авторские права и другие вспомогательные элементы.  
Его оформление должно быть одновременно незаметным (чтобы не отвлекать внимание от основного контента) и в то же время соответствовать общей стилистике сайта.

В разработанном проекте стилизация подвала реализована следующим образом:

footer {

  background-color: #0056b3;

  color: white;

  text-align: center;

  padding: 20px;

  margin-top: 50px;

  border-top: 3px solid #004094;

}

* **padding: 20px;**  
  Добавляет внутренние отступы вокруг содержимого футера. Это предотвращает прилипание текста к краям блока и визуально выделяет текст на фоне.
* **border-top: 3px solid #004094;**  
  Добавляет верхнюю границу футеру, толщиной в 3 пикселя и более тёмного синего оттенка (#004094), чем фон футера. Эта тонкая полоса отделяет подвал от основного содержимого страницы и визуально усиливает его восприятие как самостоятельного блока.

Результат показан на рисунке 2.9:

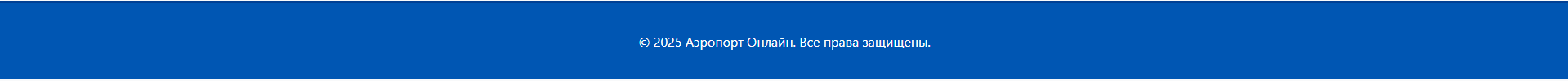


Рис. 2.9

2.8 JavaScript

**Работа скрипта авторизации пользователей**

Для реализации базовой проверки введённых данных при авторизации на сайте был использован JavaScript-скрипт, представленный в следующем виде:

document.getElementById('login-form').addEventListener('submit', function(event) {

    event.preventDefault();

    const username = document.getElementById('username').value;

    const password = document.getElementById('password').value;

    if (username === 'user' && password === 'password123') {

        alert('Авторизация успешна');

        window.location.href = 'index.html';

    } else {

        alert('Неверный логин или пароль');

    }

});

**Назначение обработчика события отправки формы**

document.getElementById('login-form').addEventListener('submit', function(event) {

* document.getElementById('login-form') — находит HTML-элемент формы по его атрибуту id="login-form".
* .addEventListener('submit', function(event) {...}) — добавляет обработчик события отправки формы (submit).
* Таким образом, когда пользователь нажимает кнопку "Войти", данная функция будет выполнена.

**Предотвращение стандартного поведения формы**

event.preventDefault();

По умолчанию при отправке формы браузер перезагружает страницу или перенаправляет её.

* Вызов preventDefault() отменяет это поведение, позволяя обработать данные формы на стороне клиента без перезагрузки страницы.

**Получение введённых пользователем данных**

const username = document.getElementById('username').value;

const password = document.getElementById('password').value;

* С помощью document.getElementById('username') и document.getElementById('password') получаются элементы формы (поля ввода имени пользователя и пароля).
* .value возвращает значения, введённые пользователем.
* Эти значения сохраняются в переменные username и password для последующей проверки.

**Проверка введённых данных**

    if (username === 'user' && password === 'password123') {

        alert('Авторизация успешна');

        window.location.href = 'index.html';

    } else {

        alert('Неверный логин или пароль');

    }

* Выполняется условная проверка введённых данных:
  + Если имя пользователя равно "user", а пароль — "password123", пользователь считается успешно авторизованным.
  + В противном случае выводится сообщение об ошибке.

3. РЕЗУЛЬТАТ

В ходе выполнения курсовой работы были изучены и практически применены основные принципы создания веб-сайта:

1. Было разработано несколько веб-страниц с использованием языка разметки HTML, что позволило сформировать структуру и наполнение сайта.
2. Внешнее оформление страниц было реализовано посредством каскадных таблиц стилей (CSS), что обеспечило единое стилистическое оформление и адаптивный дизайн проекта.
3. Создана адаптивная веб-страница с применением медиа-запросов для корректного отображения на различных устройствах.
4. Интерактивные элементы сайта были реализованы с использованием языка программирования JavaScript, что позволило добавить динамическое поведение и повысить удобство взаимодействия пользователей с системой.

В результате проведённой работы был создан современный, функциональный и эстетически привлекательный веб-сайт для аэропорта. Разработанная система соответствует предъявленным требованиям, обеспечивает удобную навигацию, предоставляет актуальную информацию о рейсах и услугах, а также отвечает современным тенденциям веб-дизайна и требованиям к адаптивности.

4. ЛИТЕРАТУРА

1. Дакетт Дж. HTML и CSS. Разработка и дизайн веб-сайтов. – М.: Эксмо, 2019.
2. Робсон Э., Фримен Э. Изучаем HTML, XHTML и CSS. – СПб.: Питер, 2018.
3. Фланаган Д. JavaScript. Подробное руководство. – СПб.: Символ-Плюс, 2017.
4. Бьюли А. Изучаем SQL. – СПб.: Символ-Плюс, 2019.

5. ПРИЛОЖЕНИЯ

Репозиторий:

<https://github.com/ziyafantaziya/Coursework>