**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 автовокзалов»

**Группа:** 680.22

**Курc:** 3

**Специальность:** Информационные технологии

**Студент**: Исмаилов Хаган

**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** Исмаилов Хаган

**Зачетная книжка**

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

**Срок выдачи**

**Дата сдачи**

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

для автовокзалов»

**Отзыв руководителя курсовой**:

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

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

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

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

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

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

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

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

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

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

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

**СОДЕРЖАНИЕ:**

[ВВЕДЕНИЕ 3](#_Toc197017420)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc197017421)

[1.1 Web-программирование 6](#_Toc197017422)

[1.2 HTML и XHTML 6](#_Toc197017423)

[**1.3 Dynamic HTML** 10](#_Toc197017424)

[**1.3.1 Каскадные таблицы стилей** 10](#_Toc197017425)

[1.3.2 JavaScript 11](#_Toc197017426)

[1.4 Bootstrap 12](#_Toc197017427)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ КУРСОВОЙ РАБОТЫ 14](#_Toc197017428)

[**2.1 ВВЕДЕНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ.** 14](#_Toc197017429)

[**2.2 СТРУКТУРА И ФУНКЦИОНАЛЬНОСТЬ WEB-ИНТЕРФЕЙСА** 18](#_Toc197017430)

[2.3. HTML-компоненты информационной системы 19](#_Toc197017431)

[**2.3.1 Структура HTML-документа** 19](#_Toc197017432)

[**2.3.2. Шапка сайта (header)** 20](#_Toc197017433)

[**2.3.3. Основная секция поиска рейсов** 21](#_Toc197017434)

[**2.3.4. Секция расписания с таблицей** 22](#_Toc197017435)

[**2.3.5. Секция "Мои билеты"** 23](#_Toc197017436)

[**2.4. СТИЛЕВОЕ ОФОРМЛЕНИЕ С ИСПОЛЬЗОВАНИЕМ CSS** 26](#_Toc197017437)

[**2.4.1. Основные переменные и сброс стилей** 26](#_Toc197017438)

[**2.4.2. Стилизация шапки сайта** 27](#_Toc197017439)

[**2.4.7 Медиа-запросы для адаптивности** 30](#_Toc197017440)

[2.5 JAVASCRIPT: ОСНОВНЫЕ КОНЦЕПЦИИ И ПРИМЕНЕНИЕ 32](#_Toc197017441)

[2.5.1 Обработчики событий и DOM-манипуляции 32](#_Toc197017442)

[2.5.2 Функции как основные строительные блоки 33](#_Toc197017443)

[2.5.3 Обработка форм и предотвращение стандартного поведения 34](#_Toc197017444)

[2.5.4 Делегирование событий и работа с DOM-иерархией 35](#_Toc197017445)

[2.5.5 Управление видимостью и интерфейсом пользователя 36](#_Toc197017446)

[2.5.6 Работа с датами 37](#_Toc197017447)

[3. РЕЗУЛЬТАТ 37](#_Toc197017448)

[4. ЛИТЕРАТУРА 40](#_Toc197017449)

[5. ПРИЛОЖЕНИЯ 41](#_Toc197017450)

# 

# **ВВЕДЕНИЕ**

Веб-системы и веб-технологии представляют собой совокупность инструментов, языков программирования и платформ, применяемых для создания, развертывания и поддержки веб-приложений. За годы развития они значительно эволюционировали, чтобы соответствовать растущим требованиям пользователей к онлайн-сервисам. Это развитие привело к появлению множества новых языков, фреймворков, библиотек и баз данных.

Появление веб-технологий сделало компьютерные системы доступными для более широкой аудитории, включая людей, ранее далёких от сферы информационных технологий. Всё больше пользователей из разных социальных групп подключаются к интернету для поиска информации, расширяя границы применения сети.

Существенное влияние на популярность веб-приложений оказало упрощение пользовательских интерфейсов. Ранее развитие компьютерной техники ограничивалось такими параметрами, как производительность процессоров, объём памяти и пропускная способность каналов связи, при этом удобство взаимодействия с системой оставалось на втором плане. Сложность интерфейсов сдерживала внедрение новых технологий. Лишь после того, как пользовательские интерфейсы стали понятны и удобны для обычных пользователей, произошёл стремительный рост интереса к возможностям компьютерных систем и интернета.

Веб-технологии объединяют весь комплекс средств, обеспечивающих работу Всемирной паутины (WWW). Поскольку работа в интернете строится на взаимодействии двух сторон — клиента и сервера — веб-технологии делятся на две группы:

1. Технологии клиентской стороны (client-side).
2. Технологии серверной стороны (server-side).

Клиентские технологии включают в себя инструменты создания веб-страниц, такие как HTML, CSS, JavaScript и другие. Серверные технологии отвечают за обработку данных и работу с базами данных через интернет с помощью таких средств, как CGI, PHP и других серверных скриптов.

Принцип взаимодействия между клиентом и сервером реализуется в архитектуре "клиент-сервер", на основе которой построено большинство современных веб-сервисов, включая WWW.

В рамках данной курсовой работы рассматривается процесс создания онлайн информационной системы для автовокзалов.

Проектирование и разработка веб-сайта для автовокзала будет включать в себя следующие этапы:

1. **Анализ требований**: На этом этапе определяется функционал сайта — предоставление расписания автобусов, информации о маршрутах, возможности бронирования билетов, отображение доступных сервисов на вокзале.
2. **Проектирование**: Создание структуры сайта, разработка макетов страниц, определение навигации, подбор цветовой схемы и шрифтов для удобства восприятия информации пользователями.
3. **Разработка**: Реализация сайта с использованием современных веб-технологий, таких как HTML, CSS, JavaScript, а также возможное подключение серверных технологий для работы с базами данных расписаний и бронирования билетов.
4. **Тестирование**: Проверка функциональности веб-сайта на различных устройствах и браузерах, устранение выявленных ошибок и оптимизация работы ресурса.

Онлайн-информационная система автовокзала должна быть максимально удобной для пользователей: обеспечивать быстрый доступ к расписанию рейсов, простой поиск маршрутов и билетов, а также предоставлять актуальную информацию о работе вокзала.

**Целью курсовой работы** является создание сайта для автовокзала, который будет сочетать в себе удобство использования, безопасность, современный и понятный интерфейс.

# **1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1** **Web-программирование**

**Веб-программирование (или веб-разработка)** — это активно развивающаяся отрасль программирования, сосредоточенная на создании динамических интернет-приложений. Языки веб-программирования делятся на две основные категории: *клиентские и серверные*.

Клиентские языки работают на стороне пользователя, обычно в браузере. Это означает, что выполнение скриптов зависит от возможностей браузера, а пользователь может изменить настройки так, чтобы скрипты вообще не обрабатывались. Кроме того, в случае использования устаревших браузеров могут возникать проблемы с поддержкой новых языков или их версий, на которые ориентировался разработчик. Однако современные браузеры обычно лишены таких недостатков, а глобальные изменения в языках происходят довольно редко — раз в несколько лет. Большинство эффективных технологий уже давно широко известны.

*Клиентский код* открыт для просмотра — любой пользователь может увидеть его через меню браузера, выбрав «Просмотреть исходный код страницы». Плюсом клиентских языков является то, что обработка данных происходит без обращения к серверу. Например, скрипт может сразу проверить корректность заполнения формы и вывести сообщение об ошибке, не отправляя данные на сервер. Однако есть и ограничение: с помощью клиентских языков невозможно сохранить или изменить данные на сервере.

Самым популярным клиентским языком считается JavaScript, разработанный компанией Netscape совместно с Sun Microsystems. Также достаточно известен язык VBScript. В последние годы популярность приобрели такие технологии, как AJAX, Adobe Flash, Microsoft Silverlight и другие.

Серверные языки программирования дают разработчикам больше возможностей. При переходе пользователя на страницу, запрос сначала обрабатывается на сервере: выполняются все связанные с ней скрипты, а затем результат в виде готового файла отправляется пользователю через сеть. Такой файл может иметь расширения HTML, PHP, ASP, Perl, SSI, XML, DHTML или XHTML.

## **1.2****HTML и XHTML**

HyperText Markup Language (HTML) — это стандартный язык разметки документов, предназначенный для отображения информации во Всемирной паутине. Поскольку большинство веб-страниц в Интернете создано с использованием HTML (или его расширения XHTML), его изучению уделяется особое внимание.

HTML предоставляет средства форматирования текста и других элементов веб-страницы, позволяя изменять цвет, начертание, стиль и название шрифта. С помощью HTML можно выделять фрагменты текста (например, ставить ударение в словах, задавать заголовки страниц или абзацев), создавать гипертекстовые ссылки для перехода между документами, реализовывать формы для ввода данных, а также встраивать мультимедийные объекты, которые могут быть обработаны как встроенными средствами браузера (например, изображения и аудиофайлы), так и внешними приложениями (например, Flash-роликами или Java-апплетами).

Документ HTML представляет собой совокупность элементов, начало и конец которых отмечаются специальными служебными символами — тегами. Теги заключаются в угловые скобки: <тег> для начала элемента и </тег> для его завершения. Пример базовой структуры документа:

html

<HTML>

<HEAD>

<TITLE>Заголовок страницы</TITLE>

</HEAD>

<BODY>

</BODY>

</HTML>

HTML является регистронезависимым языком: теги могут быть записаны как строчными, так и прописными буквами. Кроме того, допускается вложенность тегов.

Существуют также пустые теги, не содержащие вложенного содержимого, такие как <BR>, который обозначает переход на новую строку. Элементы разметки могут содержать атрибуты, определяющие их свойства (например, размер или цвет шрифта). Атрибуты указываются внутри начального тега.

Пример использования атрибута:

html

<a href="http://www.yandex.ru">Пример ссылки с атрибутом href.</a>

**Классификация тегов HTML**

1. **Гипертекстовые ссылки** Тег <A> используется для создания ссылок:

html

<A HREF="filename" TARGET="\_self">текст ссылки</A>

где filename — путь к файлу или URL-адрес, а атрибут target определяет окно или фрейм для открытия документа (\_self, \_blank, \_top, \_parent).

1. **Форматирование текста**
   * Заголовки различных уровней: <H1>, <H2>, ..., <H6>.
   * Новый абзац: <P>.
   * Перенос строки: <BR>.
   * Горизонтальная линия: <HR>.
   * Выделение цитат: <BLOCKQUOTE>.
   * Вывод текста без форматирования: <PRE>.
   * Блочные (<DIV>) и строчные (<SPAN>) контейнеры, часто применяемые с каскадными таблицами стилей (CSS).
2. **Теги для стилизации текста**
   * Курсив: <I>, выделение важности: <EM>.
   * Жирное начертание: <B>, усиление важности: <STRONG>.
   * Подчёркивание: <U>, зачёркивание: <S> или <STRIKE>.
   * Изменение размера шрифта: <BIG>, <SMALL>.
   * Анимация текста: <BLINK>, <MARQUEE>.
   * Вывод подстрочного (<SUB>) и надстрочного (<SUP>) текста.

Для задания дополнительных атрибутов шрифта используется тег <FONT>, позволяющий определить цвет (COLOR), название шрифта (FACE) и размер (SIZE).

1. **Списки**
   * Нумерованные и маркированные списки создаются при помощи тегов <UL>, <OL>, <LI>. Пример маркированного списка:

<UL>

<LI>Первый элемент списка</LI>

<LI>Второй элемент списка</LI>

</UL>

1. **Работа с мультимедийными объектами**
   * Вставка объектов: <EMBED>, <APPLET>, <SCRIPT>.
   * Вставка изображений осуществляется с помощью тега <IMG>, где задаются атрибуты SRC, ALT, TITLE, WIDTH, HEIGHT, ALIGN.
2. **Таблицы**
   * Создание таблиц: <TABLE>.
   * Строки таблицы: <TR>.
   * Ячейки данных: <TD>, заголовки столбцов: <TH>.
   * Дополнительные атрибуты: BORDER, CELLSPACING, WIDTH, HEIGHT.
3. **Формы**
   * Создание форм: <FORM>.
   * Поля ввода: <INPUT>, многострочные поля: <TEXTAREA>.
   * Выпадающие списки: <SELECT> с пунктами <OPTION>.
4. **Вывод специальных символов** Некоторые символы необходимо экранировать:
   * Амперсанд (&) — &amp;
   * Меньше (<) — &lt;
   * Больше (>) — &gt;

**Структура базовой HTML-страницы**

Каждая HTML-страница должна начинаться и завершаться тегами <html> и </html>, между которыми размещаются разделы <head> и <body>, содержащие метаданные и основное содержимое соответственно.

**XHTML: особенности и различия с HTML**

Extensible Hypertext Markup Language (XHTML) представляет собой расширение HTML с более строгими правилами синтаксиса:

* Все теги должны быть закрытыми. Самозакрывающиеся теги оформляются с помощью символа / перед угловой скобкой (<br />).
* Теги и атрибуты пишутся только строчными буквами.
* Специальные символы (например, < и &) должны быть экранированы даже в URL.
* При обнаружении синтаксических ошибок браузер обязан прекратить обработку документа и сообщить об ошибке.

Таким образом, XHTML отличается от HTML более строгими требованиями к структуре документа и поддержкой расширяемости за счёт возможности определения собственных тегов.

### **1.3 Dynamic HTML**

Dynamic HTML (DHTML) — это сочетание HTML, CSS, скриптового языка (чаще всего JavaScript) и объектной модели документов. DHTML позволяет создавать интерактивные веб-страницы, обрабатывать данные запроса и формировать динамический ответ без обращения к серверу. С его помощью можно реализовать функции Drag'n'Drop, разработать игры и другие интерактивные сервисы. Всё, что требуется — это браузер, поддерживающий работу с DHTML.

### **1.3.1 Каскадные таблицы стилей**

Каскадные таблицы стилей (CSS, Cascading Style Sheets) предназначены для описания внешнего вида элементов HTML-страницы — шрифтов, цветов, расположения и других свойств. Их основная задача — отделить содержимое от оформления, упростить управление дизайном и уменьшить дублирование кода.

CSS-стили можно подключать разными способами:

* встроенные стили внутри тега <style>,
* отдельные CSS-файлы через <link>,
* inline-стили через атрибут style внутри HTML-элементов.

Браузеры также применяют свои стандартные стили по умолчанию.

Каждое правило CSS включает селектор и блок свойств:

css

селектор {

свойство1: значение;

свойство2: значение;

}

В случае конфликтов между стилями применяется принцип каскадности: более поздние правила имеют приоритет.

До появления CSS оформление задавалось прямо в HTML. С внедрением CSS стало возможным централизованное управление стилем, лёгкая адаптация под разные устройства (экран, печать, мобильные устройства) и ускорение загрузки страниц за счёт кэширования стилей.

Недостатки CSS связаны с разным отображением в браузерах. Для работы с CSS доступны редакторы вроде **TopStyle**, который проверяет синтаксис и совместимость стилей в разных браузерах. Однако большинство HTML-редакторов уже имеют встроенную поддержку CSS, поэтому специализированные программы используются реже.

## **1.3.2 JavaScript**

Хотя JavaScript не входит в стандарты W3C, в рамках DHTML он играет важную роль.  
JavaScript — это объектно-ориентированный скриптовый язык, предназначенный для работы с объектами внутри приложений. Он широко применяется в разработке веб-страниц. По синтаксису JavaScript напоминает язык C, но имеет существенные отличия: объекты и их типы можно создавать во время выполнения кода, происходит автоматическое приведение типов и управление памятью. Также поддерживаются анонимные функции.  
Имена переменных в JavaScript чувствительны к регистру и могут содержать буквы, подчёркивания, символ доллара и цифры (но не начинаться с цифры). При этом язык не имеет прямого доступа к файловой системе, сетевым ресурсам и потокам ввода/вывода — это обеспечивает безопасность скриптов, исполняемых в браузере. Также JavaScript ограничивает доступ к содержимому других страниц.

Скрипты подключаются через теги <script></script>, например:

<script type="text/javascript">

alert('Hello, World!');

</script>

Или через внешний файл:

<script type="text/javascript" src="путь\_к\_файлу.js"></script>

JavaScript открывает широкие возможности для динамического изменения контента страниц: автоматическое заполнение форм, изменение стилей, скрытие и отображение элементов, управление поведением веб-приложений.

Хотя выделить лучший редактор для JavaScript сложно, популярной является среда Eclipse с поддержкой JavaScript через плагины. Также современные браузеры оснащены встроенными средствами для отладки скриптов.

## **1.4 Bootstrap**

Bootstrap — это бесплатный и открытый фреймворк для HTML, CSS и JavaScript, используемый для быстрой верстки адаптивных сайтов и веб-приложений. Он включает готовые шаблоны оформления для форм, кнопок, навигации, типографики и других интерфейсных элементов.

Bootstrap особенно полезен, если:

* сайт состоит из множества страниц,
* страницы строятся из базовых компонентов,
* не планируется частых редизайнов,
* важна скорость и стандартизация верстки.

По сути, Bootstrap — это набор файлов, после подключения которых становятся доступны многочисленные готовые классы и компоненты для ускорения верстки.

Классы Bootstrap делятся на три основные категории:

* для построения сетки страницы,
* для оформления контента (текст, изображения, таблицы),
* для вспомогательных задач (скрытие элементов, отступы, выравнивание).

Кроме классов, Bootstrap предлагает готовые компоненты: хлебные крошки, кнопки, выпадающие списки, модальные окна, формы, панели навигации и многое другое.

Для работы с Bootstrap понадобится текстовый редактор (например, Visual Studio Code, Atom, Sublime Text) или IDE и современный браузер.

Bootstrap можно установить несколькими способами:

1. Подключить через BootstrapCDN — достаточно вставить ссылку в HTML-шаблон.
2. Скачать готовые CSS- и JS-файлы и подключить их локально.
3. Скачать исходные файлы для последующей компиляции.
4. Установить через менеджеры пакетов (yarn, npm, RubyGems, NuGet, Composer).

**2. ПРАКТИЧЕСКАЯ ЧАСТЬ КУРСОВОЙ РАБОТЫ**

**2.1 ВВЕДЕНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ.**

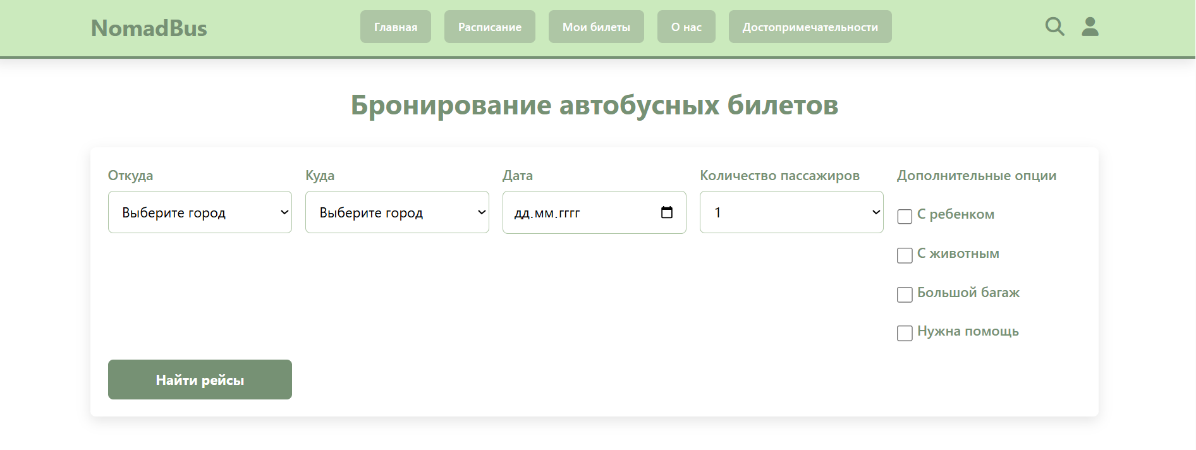
Данная практическая часть курсовой работы посвящена разработке информационной системы "NomadBus" для автовокзалов. Система представляет собой веб-приложение, позволяющее пользователям осуществлять поиск, фильтрацию и бронирование автобусных билетов между различными городами. В работе представлена полная реализация клиентской части приложения с применением современных технологий веб-разработки: HTML5, CSS3 и JavaScript.

Основная цель проекта заключается в создании удобного и интуитивно понятного интерфейса для пассажиров автобусов, который позволит быстро находить необходимую информацию о рейсах и приобретать билеты онлайн. Разработанная система решает следующие задачи:

* Предоставление актуальной информации о расписании автобусных рейсов
* Поиск маршрутов по различным параметрам (город отправления, город прибытия, дата, время)
* Бронирование и оплата билетов через интернет
* Хранение и управление приобретенными билетами
* Информирование о туристических достопримечательностях в городах назначения

В ходе разработки был применен подход, ориентированный на пользователя (user-centered design), что позволило создать интуитивно понятный интерфейс с учетом потребностей и ожиданий конечного пользователя.

Разработанная информационная система имеет многоуровневую архитектуру, где клиентская часть представлена веб-интерфейсом, с которым непосредственно взаимодействует пользователь. Именно эта часть и является предметом данной курсовой работы. В реальных условиях такая система взаимодействует с серверной частью, которая обеспечивает обработку запросов, хранение данных и интеграцию с платежными системами.



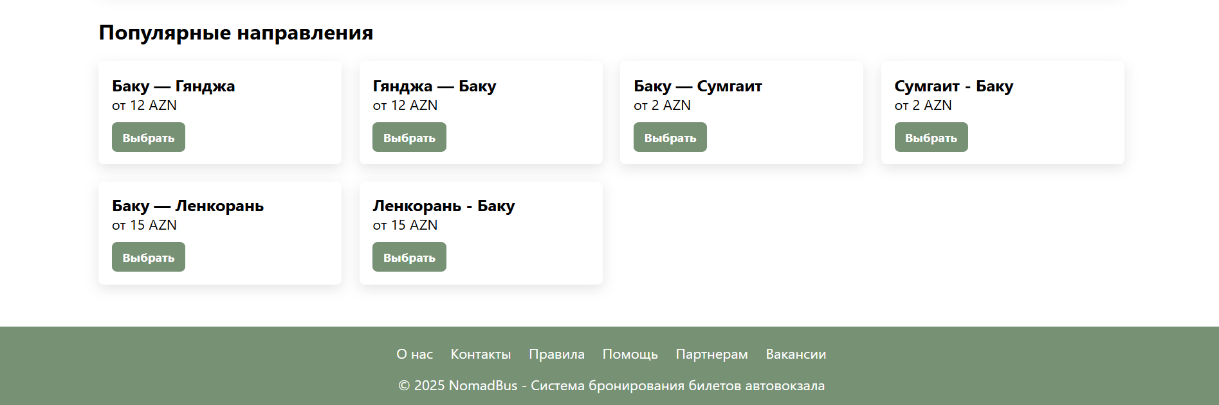


Рис 2.1 – «Главная страница сайта»

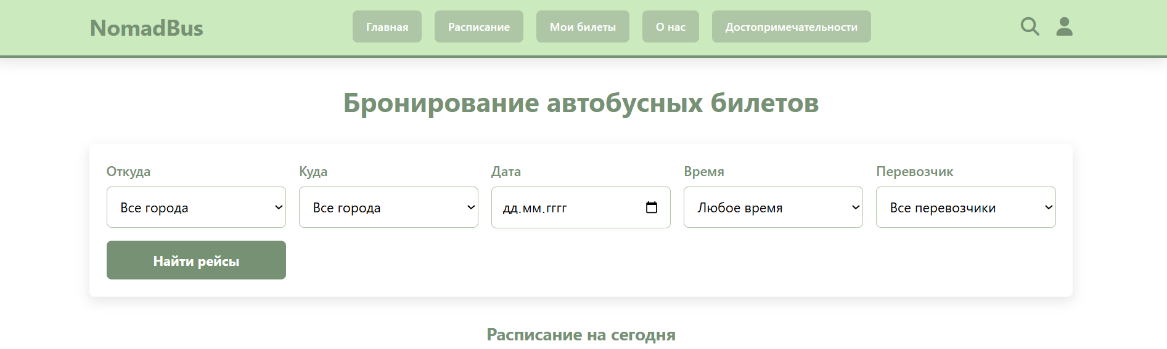




Рис. 2.2 – «Страница с расписанием»

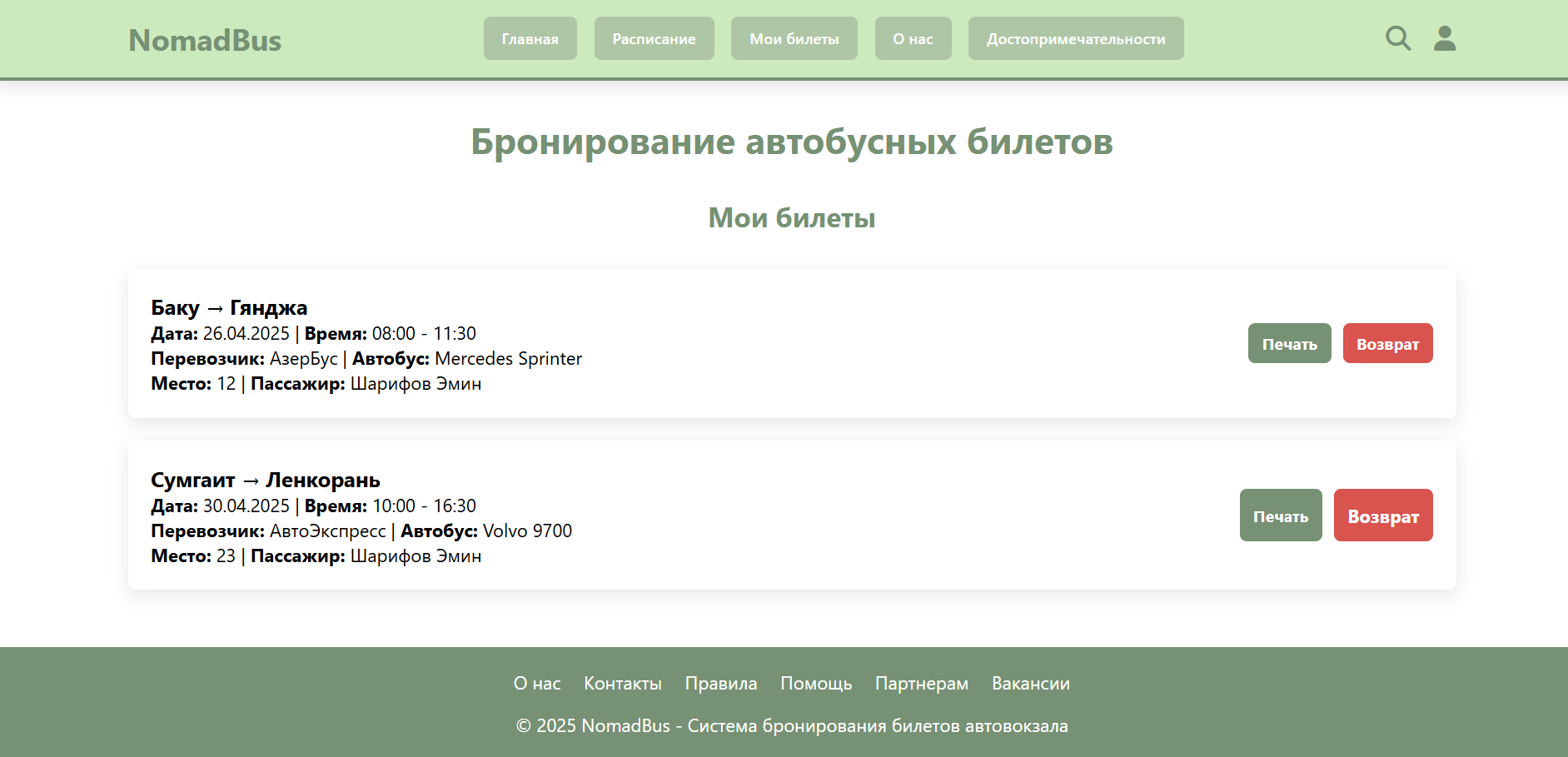


Рис 2.3 – Страница с «Моими билетами»



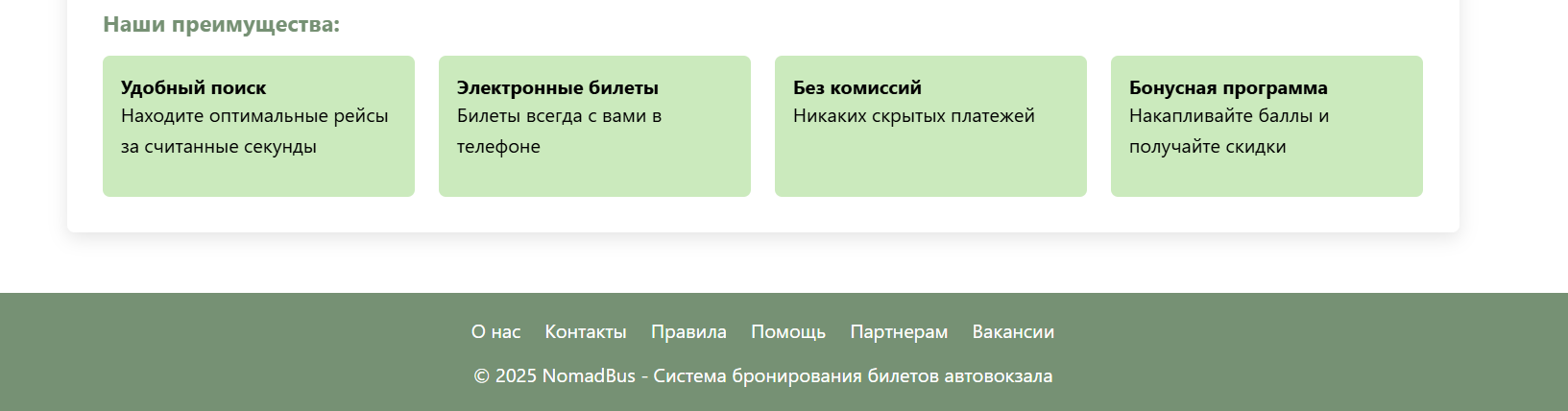
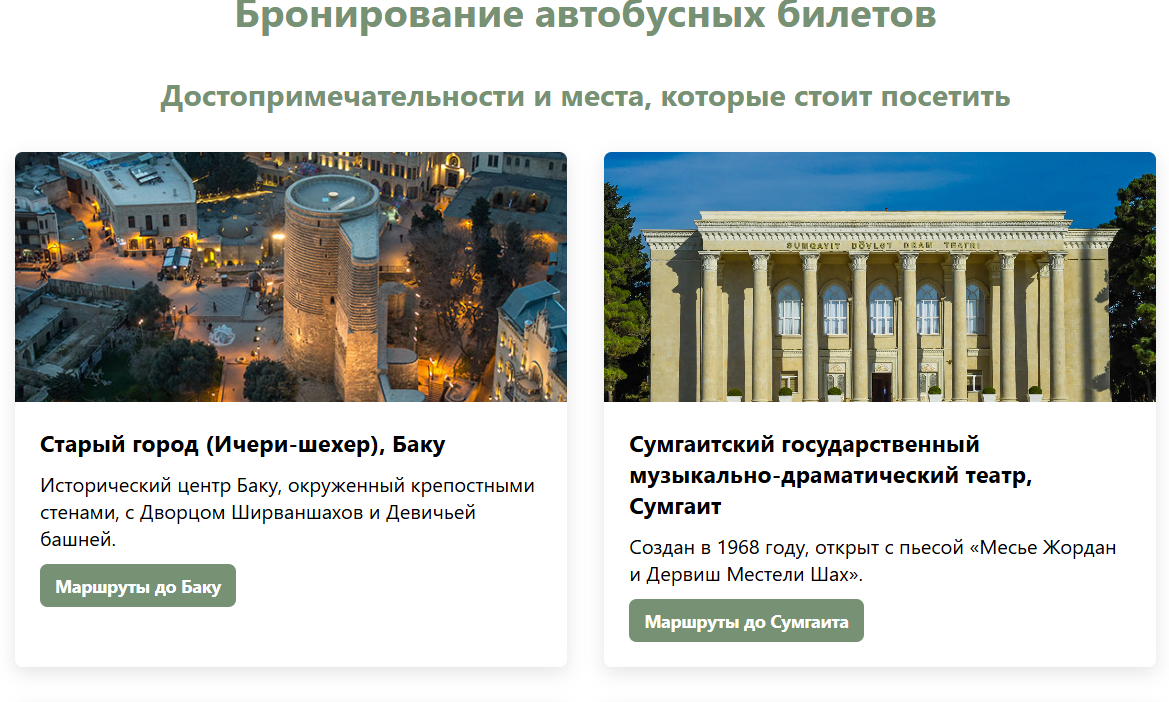


Рис 2.4 – Страница «О нас»





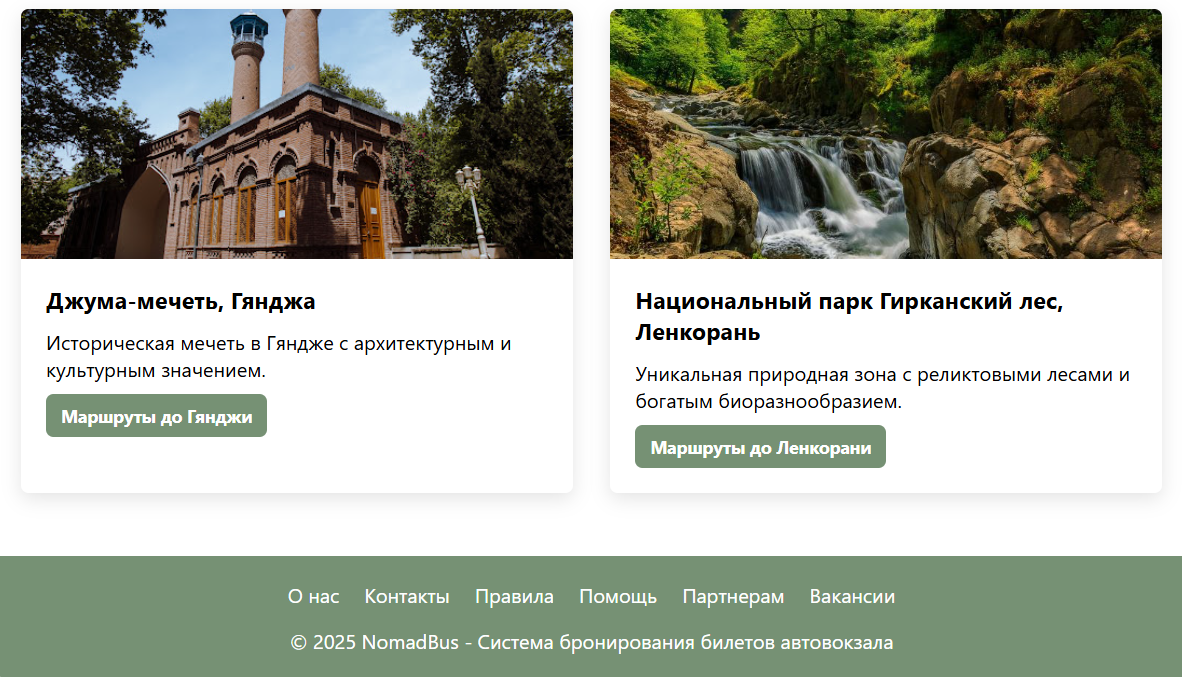


Рис 2.5 – Страница «Достопримечательности»

**2.2 СТРУКТУРА И ФУНКЦИОНАЛЬНОСТЬ WEB-ИНТЕРФЕЙСА**

Разработанное веб-приложение состоит из нескольких функциональных разделов, каждый из которых отвечает за определенный аспект работы системы:

1. **Главная страница** - включает форму поиска рейсов и блок с популярными направлениями. Данный раздел служит отправной точкой для пользователей, предоставляя им наиболее востребованные функции.
2. **Расписание** - отображает доступные рейсы с возможностью фильтрации по различным параметрам, таким как:
   * Город отправления
   * Город прибытия
   * Дата поездки
   * Время отправления
   * Перевозчик
3. **Мои билеты** - представляет приобретенные пользователем билеты и предоставляет возможность их управления (печать, возврат).
4. **О нас** - содержит информацию о сервисе, его преимуществах и миссии, что повышает доверие пользователей к системе.
5. **Достопримечательности** - предлагает информацию о туристических местах в городах назначения, что является дополнительной ценностью для путешественников.
6. **Процесс покупки билета** - пошаговая форма для приобретения билета, включающая следующие этапы:
   * Выбор маршрута (откуда, куда, дата)
   * Выбор конкретного рейса
   * Ввод данных пассажира
   * Оплата

Веб-интерфейс разработан с учетом современных тенденций в UI/UX дизайне и обеспечивает следующие характеристики:

* **Удобство использования (usability)** - интуитивно понятная навигация, логичная структура, последовательные и понятные пользовательские сценарии
* **Доступность (accessibility)** - корректная семантическая разметка, логический порядок табуляции
* **Адаптивность (responsive design)** - корректное отображение на устройствах с различными размерами экрана
* **Эстетичность** - гармоничная цветовая схема, согласованное оформление элементов, современный внешний вид

**2.3. HTML-компоненты информационной системы**

**2.3.1 Структура HTML-документа**

html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

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

<title>NomadBus - Система бронирования билетов автовокзала</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

</head>

**Описание основных элементов:**

* <!DOCTYPE html> - объявление типа документа для HTML5
* <html> - корневой элемент страницы, lang="ru" указывает язык содержимого
* <head> - контейнер для метаданных страницы
* <meta> - метатеги:
  + charset="UTF-8" задаёт кодировку символов
  + viewport настраивает отображение для мобильных устройств
* <title> - заголовок страницы в браузере
* <link> - подключение внешних ресурсов, здесь - библиотеки иконок Font Awesome

**2.3.2. Шапка сайта (header)**

html

<header>

<div class="container">

<nav class="navbar">

<a href="#" class="logo">NomadBus</a>

<ul class="nav-links">

<li><button onclick="showSection('home-section')">Главная</button></li>

*<!-- Другие пункты меню -->*

</ul>

<div class="header-icons">

<div class="icon-wrapper" id="search-icon">

<i class="fas fa-search"></i>

<div class="modal" id="search-modal">

*<!-- Модальное окно поиска -->*

</div>

</div>

*<!-- Иконка пользователя -->*

</div>

</nav>

</div>

</header>

**Описание элементов:**

* <header> - семантический тег для шапки сайта
* <div> - универсальный блочный элемент, используется для группировки и стилизации
* <nav> - семантический тег для навигационного меню
* <a> - гиперссылка, здесь используется для логотипа
* <ul> и <li> - ненумерованный список и его элементы для навигационных ссылок
* <button> - интерактивная кнопка с атрибутом onclick для вызова JavaScript-функции
* <i> - элемент для иконок (используется Font Awesome)
* Атрибуты:
  + class - указывает CSS-классы для стилизации
  + id - уникальный идентификатор для JavaScript-манипуляций

**2.3.3. Основная секция поиска рейсов**

html

<section id="home-section" class="page-content">

<div class="filter-section">

<form class="filter-form" id="main-search-form">

<div class="form-group">

<label for="from">Откуда</label>

<select id="from" class="form-control">

<option value="">Выберите город</option>

<option value="baku">Баку</option>

*<!-- Другие города -->*

</select>

</div>

*<!-- Другие поля формы -->*

<div class="form-group">

<button type="submit" class="btn search-btn">Найти рейсы</button>

</div>

</form>

</div>

</section>

**Описание элементов:**

* <section> - семантический тег для обозначения разделов страницы
* <form> - контейнер для элементов ввода, отправляемых на сервер
* <label> - текстовая метка для поля формы, атрибут for связывает с элементом ввода
* <select> - выпадающий список для выбора одного из вариантов
* <option> - вариант выбора в выпадающем списке
* Атрибуты:
  + type="submit" - определяет кнопку отправки формы
  + id - уникальный идентификатор элемента

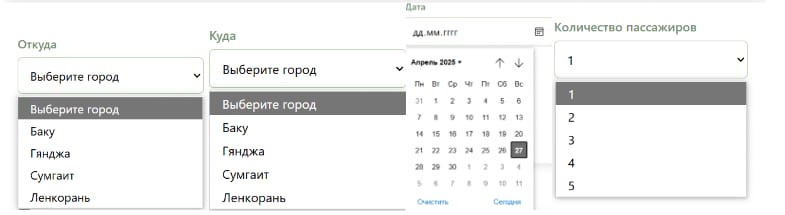


Рис. 2.6 – Меню выбора для поездки

**2.3.4. Секция расписания с таблицей**

html

<section id="schedule-section" class="page-content" style="display: none;">

*<!-- Форма фильтрации -->*

<div class="schedule-container">

<div class="schedule-date">

<span>Расписание на сегодня</span>

</div>

<div class="table-wrapper">

<table class="schedule-table">

<thead>

<tr>

<th>Маршрут</th>

*<!-- Другие заголовки -->*

</tr>

</thead>

<tbody>

<tr>

<td>Баку → Гянджа</td>

*<!-- Другие ячейки -->*

<td><button class="ticket-btn">Выбрать</button></td>

</tr>

*<!-- Другие строки -->*

</tbody>

</table>

</div>

</div>

</section>

**Описание элементов:**

* <span> - строчный элемент для оформления текста внутри блоков
* <table> - таблица для структурированного отображения данных
* <thead> - группа заголовков таблицы
* <tbody> - основное содержимое таблицы
* <tr> - строка таблицы
* <th> - ячейка заголовка таблицы
* <td> - ячейка данных таблицы
* Атрибут style="display: none;" - встроенный CSS для скрытия элемента

**2.3.5. Секция "Мои билеты"**

html

<section id="tickets-section" class="page-content" style="display: none;">

<h2 style="text-align: center; margin-bottom: 30px; color: var(--color-accent);">Мои билеты</h2>

<div class="ticket-card">

<div class="ticket-details">

<h3>Баку → Гянджа</h3>

<p><strong>Дата:</strong> 26.04.2025 | <strong>Время:</strong> 08:00 - 11:30</p>

*<!-- Другие детали билета -->*

</div>

<div class="ticket-actions">

<button class="btn btn-sm">Печать</button>

<button class="btn btn-sm" style="background-color: #d9534f;">Возврат</button>

</div>

</div>

</section>

**Описание элементов:**

* <h2>, <h3> - заголовки разных уровней
* <p> - параграф текста
* <strong> - выделение текста жирным начертанием
* Встроенные стили (style="...") - директивы CSS внутри HTML-элемента:
  + text-align - выравнивание текста
  + margin-bottom - отступ снизу
  + color - цвет текста
  + background-color - цвет фона

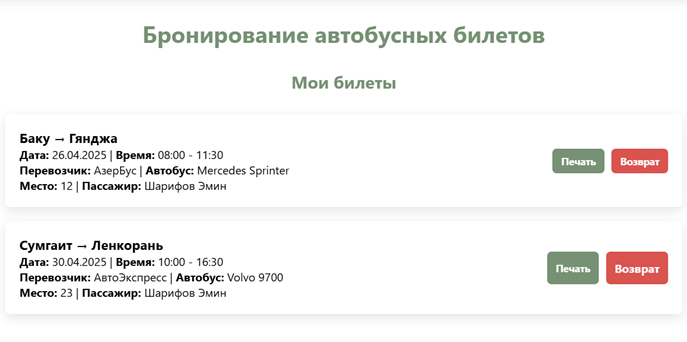


Рис. 2.7 – «Мои билеты»

В остальных секциях ("О нас", "Достопримечательности", "Покупка билета") используются уже описанные выше HTML-элементы с различным содержимым и атрибутами. Вместо повторения полных описаний приведем только новые элементы:

* <ul style="margin-left: 20px;"> - маркированный список с отступом слева
* <li> - элементы списка в разделе "О нас"
* Стилизация через свойство grid-template-columns - создание адаптивной сетки в секции "Достопримечательности"
* <footer> - семантический тег для нижней части страницы
* &copy; - HTML-сущность для символа авторского права (©)

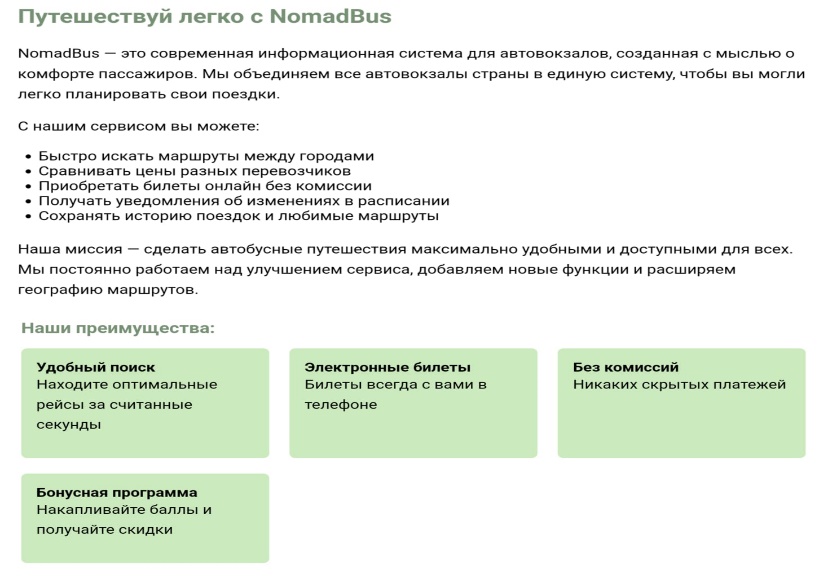


Рис 2.8 –«О нас»

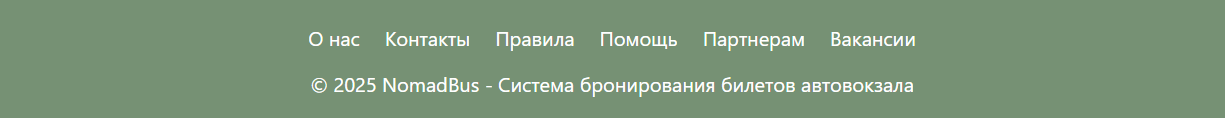


Рис. 2.9 – «Нижняя панель сайта»

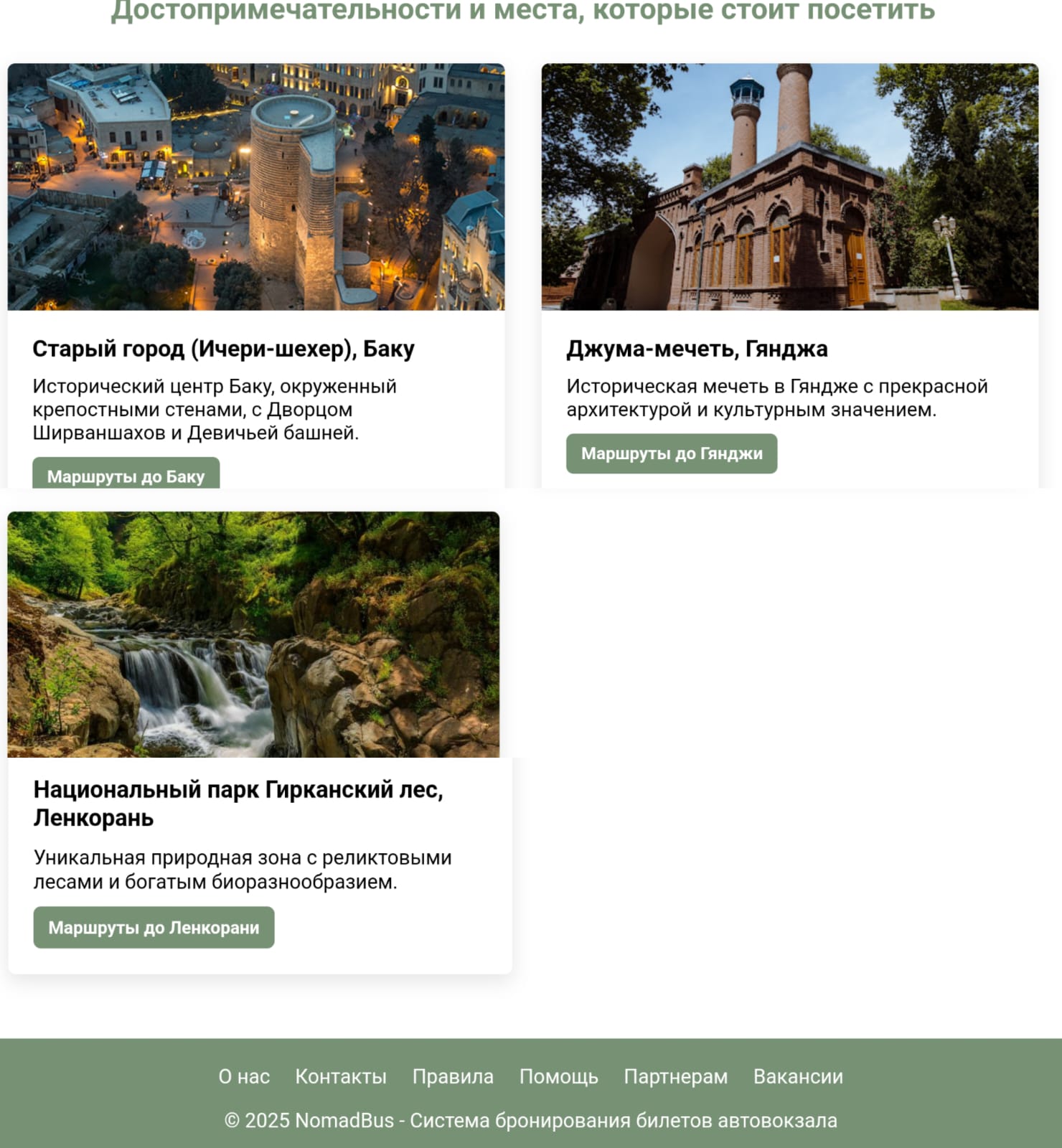


Рис. 2.10 – «Достопримечательности»

**2.4. СТИЛЕВОЕ ОФОРМЛЕНИЕ С ИСПОЛЬЗОВАНИЕМ CSS**

**2.4.1. Основные переменные и сброс стилей**

:root {

--color-primary: #cbeabd;

--color-secondary: #aec6a5;

*/\* Другие переменные \*/*

}

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

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

}

**Описание элементов:**

* **:root** - псевдокласс, представляющий корневой элемент документа
* **--color-\*** - CSS-переменные (custom properties) для центрального управления цветами
* **\*** - универсальный селектор, применяется ко всем элементам
* **CSS**-свойства:
  + **margin/padding** - внешние/внутренние отступы
  + **box-sizing** - модель расчета размеров элемента
  + **font-family** - семейство шрифтов

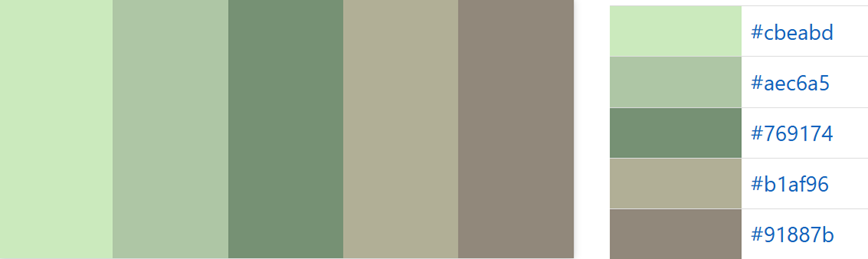


Рис. 2.11 - цветовая гамма страницы

**2.4.2. Стилизация шапки сайта**

header {

width: 100%;

background-color: var(--color-primary);

padding: 15px 0;

border-bottom: 3px solid var(--color-accent);

box-shadow: var(--shadow-standard);

position: relative;

}

**Описание основных CSS-свойств:**

* width - ширина элемента
* background-color - цвет фона
* padding - внутренние отступы
* border-bottom - нижняя граница элемента
* box-shadow - тень вокруг элемента
* position - метод позиционирования элемента
* var(--переменная) - использование переменной CSS

css

.navbar {

display: flex;

justify-content: space-between;

align-items: center;

}

**Описание свойств Flexbox:**

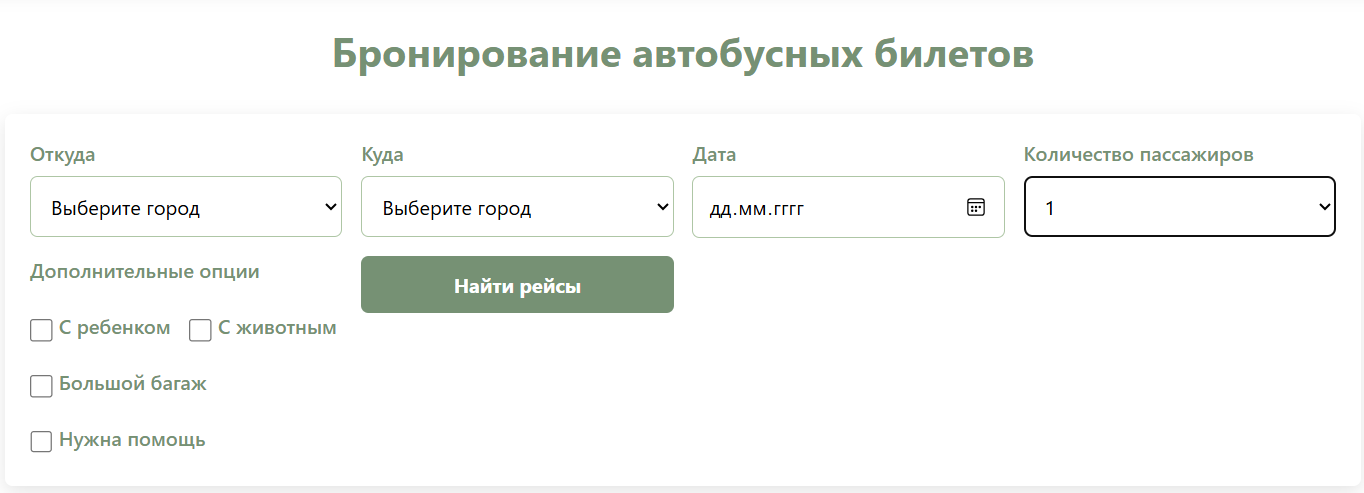
* display: flex - включает режим гибкого контейнера
* justify-content - выравнивание элементов по главной оси
* align-items - выравнивание элементов по поперечной оси



Рис. 2.12 - Стилизация навигационных элементов

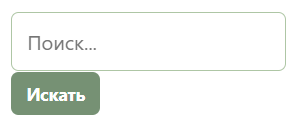
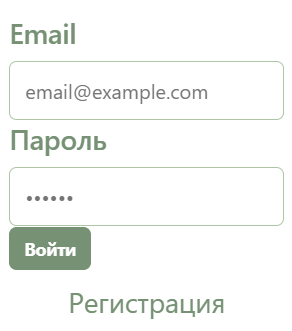
В остальных разделах CSS используются аналогичные принципы стилизации. Ключевые особенности:

* **Стилизация форм:**
  + display: grid - макет на основе сетки
  + grid-template-columns - определение столбцов сетки
  + gap - промежутки между элементами сетки

****Рис 2.13 - Стилизация бронивания автобусных билетов

**Стилизация кнопок:**

* transition - плавное изменение свойств при наведении
* :hover - псевдокласс для стилей при наведении курсора

****Рис 2.14 - представление авторизации Рис 2.15 - блок отображаения при нажатии на на лупу.

**Стилизация таблиц:**

* position: sticky - фиксация заголовков при прокрутке
* white-space: nowrap - запрет переноса строк



Рис 2.16 - Таблица расписания с выделенной строкой

**2.4.7 Медиа-запросы для адаптивности**

@media (max-width: 1000px) {

.navbar {

flex-direction: column;

gap: 15px;

}

.nav-links {

flex-direction: column;

gap: 10px;

width: 100%;

}

.nav-links button {

width: 100%;

}

.filter-form {

grid-template-columns: 1fr;

}

.header-icons {

margin-top: 15px;

}

.modal {

right: -50px;

}

.ticket-card {

flex-direction: column;

gap: 15px;

}

.ticket-actions {

width: 100%;

justify-content: space-between;

}

}

**Описание:**

* Использование медиа-запросов для адаптации интерфейса под мобильные устройства
* Изменение направления flexbox с горизонтального на вертикальное для меню и карточек билетов
* Переключение формы поиска на одну колонку на узких экранах
* Адаптация расположения модальных окон и иконок
* Растяжение кнопок на всю ширину экрана для удобства касания на мобильных устройствах

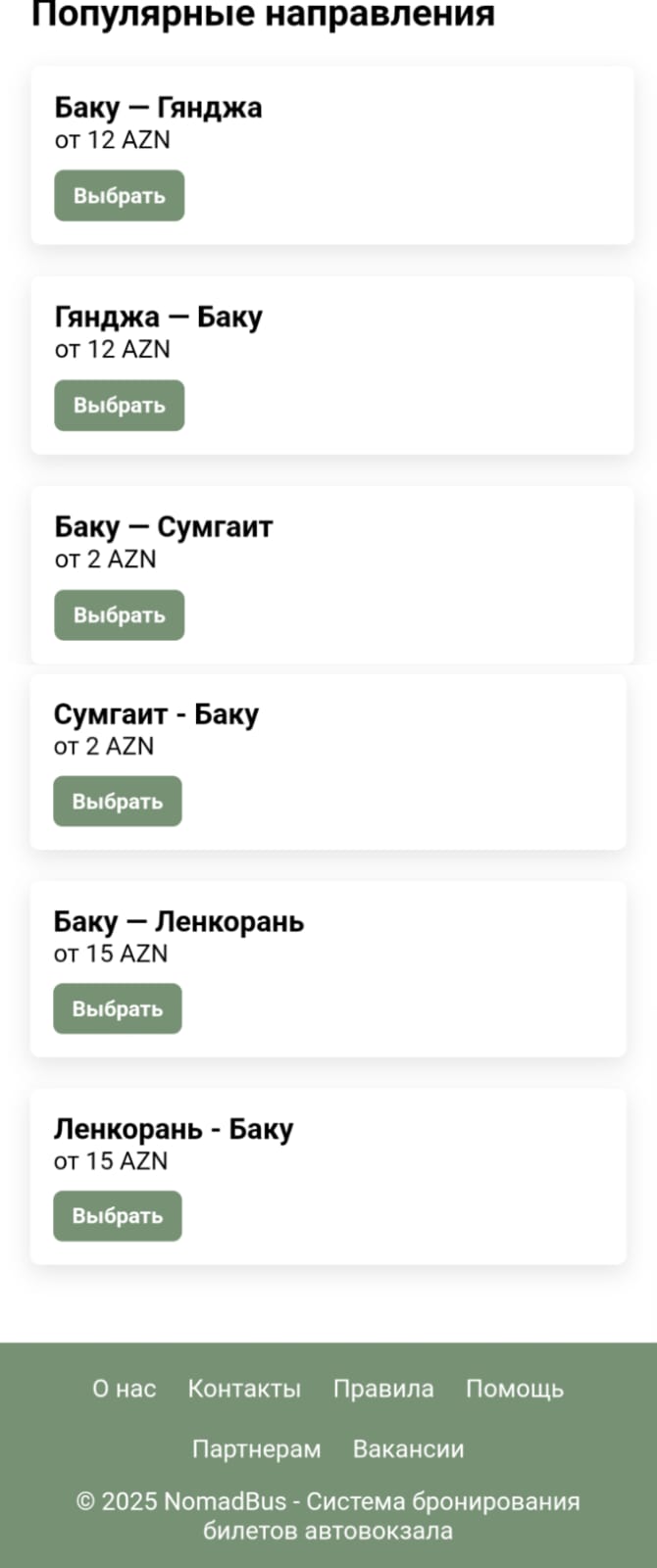
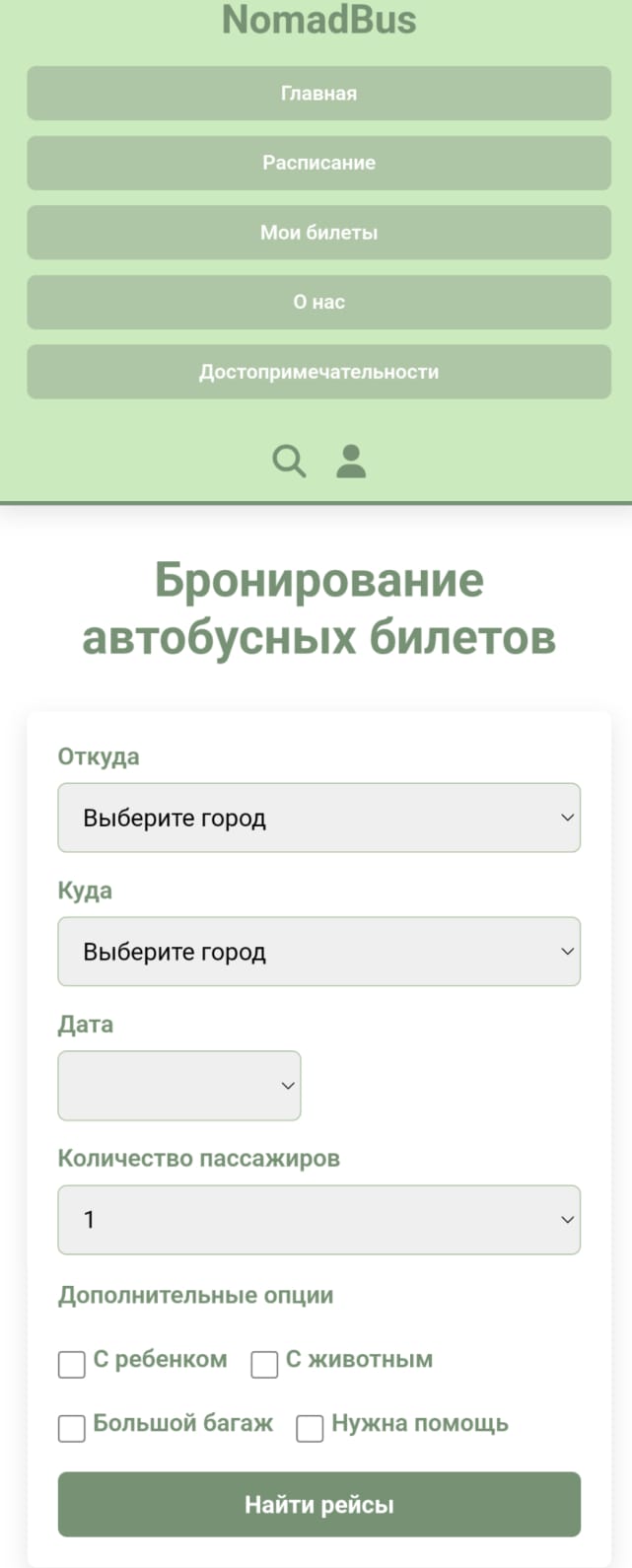


Рис 2.17 – «Мобильное представление страницы»

* 1. **JAVASCRIPT: ОСНОВНЫЕ КОНЦЕПЦИИ И ПРИМЕНЕНИЕ**

JavaScript - это мощный язык программирования, который в основном используется для создания интерактивных веб-страниц. Давайте разберем ключевые элементы и принципы его работы на основе предоставленного вами кода.

**2.5.1 Обработчики событий и DOM-манипуляции**

javascript

document.addEventListener('DOMContentLoaded', function() {

const searchIcon = document.getElementById('search-icon');

const searchModal = document.getElementById('search-modal');

searchIcon.addEventListener('click', function(e) {

e.stopPropagation();

searchModal.style.display = searchModal.style.display === 'block' ? 'none' : 'block';

userModal.style.display = 'none';

});

});

**Описание элементов JavaScript:**

* document.addEventListener() - регистрация обработчика события
* 'DOMContentLoaded' - событие завершения загрузки DOM
* function() {...} - анонимная функция-обработчик
* const - объявление константной переменной
* document.getElementById() - поиск элемента по идентификатору
* e.stopPropagation() - остановка всплытия события
* Тернарный оператор условие ? значение1 : значение2

**2.5.2 Функции как основные строительные блоки**

javascript

function showSection(sectionId) {

*// Скрываем все секции*

const sections = document.querySelectorAll('.page-content');

sections.forEach(section => {

section.style.display = 'none';

});

*// Показываем выбранную секцию*

document.getElementById(sectionId).style.display = 'block';

*// Прокручиваем страницу вверх*

window.scrollTo(0, 0);

}

**Описание:**

* Функция showSection() принимает параметр sectionId и использует его для показа определённой секции страницы
* document.querySelectorAll() выбирает все элементы, соответствующие CSS-селектору
* sections.forEach() применяет указанную функцию к каждому элементу коллекции
* Стрелочная функция section => {...} - современный синтаксис для компактных анонимных функций
* window.scrollTo() программно прокручивает страницу в указанную позицию

**2.5.3 Обработка форм и предотвращение стандартного поведения**

javascript

const mainSearchForm = document.getElementById('main-search-form');

mainSearchForm.addEventListener('submit', function(e) {

e.preventDefault();

*// Получаем значения полей*

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

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

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

*// Проверяем заполнение обязательных полей*

if (!from || !to || !date) {

alert('Пожалуйста, заполните все обязательные поля');

return;

}

*// Переходим к расписанию и фильтруем результаты*

showSection('schedule-section');

filterSchedule(from, to, date);

});

**Описание:**

* e.preventDefault() отменяет стандартное поведение браузера (в данном случае перезагрузку страницы при отправке формы)
* .value получает введённые пользователем данные из полей формы
* Условная конструкция if() проверяет заполнение необходимых полей
* alert() выводит всплывающее уведомление для пользователя
* Вызов других функций (showSection, filterSchedule) демонстрирует модульный подход

**2.5.4 Делегирование событий и работа с DOM-иерархией**

javascript

const ticketButtons = document.querySelectorAll('.ticket-btn');

ticketButtons.forEach(button => {

button.addEventListener('click', function() {

*// Если кнопка не отключена (есть места)*

if (!button.hasAttribute('disabled')) {

*// Переходим к процессу покупки билета*

showSection('purchase-section');

*// Заполняем данные маршрута на основе выбранного рейса*

const row = button.closest('tr');

const route = row.cells[0].textContent;

const departure = row.cells[1].textContent;

*// Предзаполняем форму покупки билета*

const [from, to] = route.split('→').map(city => city.trim());

}

});

});

**Описание:**

* button.hasAttribute() проверяет наличие атрибута у элемента
* button.closest('tr') находит ближайший родительский элемент указанного типа (строку таблицы)
* row.cells[0].textContent получает текстовое содержимое конкретной ячейки таблицы
* Деструктуризация массива const [from, to] и цепочка методов .split('→').map(city => city.trim()) демонстрируют современный подход к обработке строк

**2.5.5 Управление видимостью и интерфейсом пользователя**

javascript

function filterSchedule(from, to, date, time, carrier) {

*// В реальном приложении здесь был бы AJAX-запрос к серверу*

*// Для демонстрации просто покажем уведомление*

let message = 'Поиск рейсов с параметрами:\n';

message += from ? `Откуда: ${from}\n` : '';

message += to ? `Куда: ${to}\n` : '';

message += date ? `Дата: ${date}\n` : '';

*// Обновляем заголовок расписания*

document.querySelector('.schedule-date span').textContent = date

? `Расписание на ${formatDate(date)}`

: 'Расписание на сегодня';

console.log(message);

}

**Описание:**

* Функция принимает множество опциональных параметров
* Строковые шаблоны с использованием обратных кавычек и ${переменная} упрощают форматирование строк
* Условные добавления к строке через тернарный оператор
* document.querySelector() находит первый элемент, соответствующий CSS-селектору
* console.log() выводит информацию в консоль браузера для отладки

**2.5.6 Работа с датами**

javascript

function formatDate(dateString) {

const date = new Date(dateString);

return date.toLocaleDateString('ru-RU');

}

**Описание:**

* Создание объекта Date из строки
* Форматирование даты с учетом локали (в данном случае русскоязычный формат)

***Общая структура и подход***

В представленном коде используется модульный подход с разделением функциональности:

1. Инициализация и настройка обработчиков событий при загрузке страницы
2. Объявление функций для управления интерфейсом
3. Обработка пользовательского ввода и форм
4. Манипуляции с DOM для обновления содержимого страницы

Это типичный подход для JavaScript в браузере, где основная задача - обеспечение интерактивности веб-страницы и реагирование на действия пользователя без перезагрузки страницы

1. **РЕЗУЛЬТАТ**

На основании всего кода, веб-сайта и предметной области, связанной с обработкой данных и версткой веб-страниц, можно сделать несколько ключевых выводов и результатов. Вот основные моменты.

1. **Техническое качество и адаптивность интерфейса:**

Сайт построен на семантической HTML-разметке с гибкой, адаптивной вёрсткой (Flexbox, Grid, медиазапросы), что обеспечивает корректное отображение на разных устройствах. CSS организован структурно, что позволяет масштабировать проект и при необходимости перейти на препроцессоры (SASS). Используются ARIA-атрибуты и поддержка клавиатуры, что повышает доступность.

1. **Безопасность, обработка данных и производительность:**

Формы имеют двухуровневую валидацию, а также предусмотрены меры защиты от XSS, CSRF и SQL-инъекций. Оптимизация загрузки (Lazy Loading, минимизация ресурсов, использование CDN) повышает скорость сайта. Также реализовано логирование и корректная обработка ошибок, что важно для стабильности.

**Итог:**

Наш сайт демонстрирует уверенное движение в сторону профессиональной системы для отображения и управления данными автовокзала. С учётом дальнейшей доработки по направлениям адаптивности, безопасности, оптимизации и доступности — он может стать надёжным инструментом для пользователей и сотрудников вокзала.

1. **ЛИТЕРАТУРА**

* Брюс Лоусон, Реми Шарп - “Изучаем HTML5. Библиотека специалиста (ориг. название: "Introducing HTML5 ", New Riders)
* Кристофер Шмитт, Кайл Симпсон - "HTML5: Рецепты программирования"
* Эрик А. Майер - "CSS. Карманный справочник", Вильямс, 2017
* Кит Грант - "CSS для профи (ориг. название: "CSS in Depth", Manning)
* <https://html5book.ru/>
* Выразительный JavaScript. Современное веб-программирование. Третье издание

Марейн Хавербеке

* Фотографии – [Джума-мечеть](https://lh5.googleusercontent.com/proxy/uqhxR7KwSb3U7uttJHLHZI-hkYOe_yf05f-ehXwi-7FH052g1Ecl8O-yWEwzaWRtvExV61Ikp1hqKw5FoxQ0V9K8BwxBRFJl), [Ичери-Шехер](https://azerbaijan.az/uploads/news-files/melumatlar/Paytaxt/VUQ_6844.jpg), [Парк в Лянкорани](https://lankaransprings.az/photos/products/1626097993.AA29011D7F00010136D339971807B51D.jpg)
* [Вдохновение и прототип](https://freight.ady.az/ru?_gl=1*vg1dpr*_gcl_au*NjU2Njc3MjAxLjE3NDU2NTkyNDE.)

1. **ПРИЛОЖЕНИЯ**

Репозиторий, содержит код страницы, курсовую, а также презентацию по теме «Разработка онлайн информационной системы для автовокзала»

<https://github.com/madtayari/NomadBusProject>