Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
подпись Л.Н. Храмова
инициалы, фамилия
« 11 » июня 2025 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ «ПУТЕВОДИТЕЛЬ ПО КРАСНОЯРСКОМУ КРАЮ»

Руководитель	11.06.2025 подпись, дата	доцент, канд. пед. наук должность, ученая степень	Е. В. Киргизова инициалы, фамилия
Выпускник	11.06.2025 подпись, дата		М. М. Скок инициалы, фамилия
Нормоконтролер	11.06.2025 подпись, дата		Е. В. Киргизова инициалы, фамилия

Лесосибирск 2025

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка web-приложения «Путеводитель по Красноярскому краю»» содержит 77 страниц текстового документа, 39 иллюстраций, 4 таблицы, 40 использованных источников.

HTML, CSS, JAVASCRIPT, WEB-ПРИЛОЖЕНИЕ, ПУТЕВОДИТЕЛЬ, API.

Цель исследования – теоретически обосновать и разработать web-приложение «Путеводитель по Красноярскому краю», ориентированное на знакомство пользователей с достопримечательностями и на исследование менее известных мест региона.

Объект исследования – путеводитель для информационного сопровождения по Красноярскому краю.

Предмет исследования – процесс проектирования и разработки web-приложения, ориентированного на предоставление туристической информации.

Для достижения поставленной цели необходимо решить следующие задачи:

– на основе анализа туристической отрасли Красноярского края и существующих решений определить требования к функциональности web-приложения;

– разработать структуру и интерфейс web-приложения; средствами JavaScript реализовать навигацию, фильтрацию, голосового помощника и поиск объектов; настроить взаимодействие с голосовым API через Node.js сервер;

– провести тестирование работоспособности web-приложения.

В результате выполнения выпускной квалификационной работы разработано web-приложение «Путеводитель по Красноярскому краю».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня цифровые технологии всё активнее проникают в повседневную жизнь, особенно в такие сферы, как туризм и краеведение. Пользователи всё чаще обращаются к онлайн-сервисам при планировании поездок, выборе маршрутов и изучении достопримечательностей. Быстрый доступ к достоверной информации о достопримечательностях, маршрутах и инфраструктуре помогает путешественникам планировать поездки, а регионам – развивать туристическую отрасль. Красноярский край, обладая уникальным природным и культурным наследием, нуждается в современных инструментах представления своего потенциала.

Печатные издания, такие как классические путеводители, становятся всё менее удобными из-за сложности актуализации данных и отсутствия интерактивных возможностей. Существующие онлайн-сервисы, такие как Google Maps, Яндекс Карты или TravelAsk, предоставляют лишь обобщённую информацию, зачастую не адаптированную под специфику конкретного региона.

Красноярский край входит в число крупнейших субъектов Российской Федерации по площади и отличается разнообразием природных, историко-культурных объектов. Среди них – заповедник «Столбы», плато Путорана, река Енисей, Красноярская ГЭС, музеи, национальные парки и другие достопримечательности. Несмотря на это, многие интересные локации остаются малоизвестными среди туристов из-за отсутствия централизованного и удобного инструмента, способного наглядно и структурировано представить всю необходимую информацию.

Создание web-приложения позволяет решить эту проблему, объединив в одном интерфейсе карту, описания, изображения, поиск по ключевым объектам, фильтрацию и голосового помощника, при этом не требуя установки дополнительного программного обеспечения. Приложение может работать напрямую в браузере, включая автономный режим при локальном запуске.

Большинство современных туристических сервисов перегружены функциональностью, сложны в навигации и не учитывают локальные особенности. Поэтому существует потребность в автономном и интуитивно понятном web-приложении, которое может запускаться в браузере без установки.

Цель исследования – теоретически обосновать и разработать web-приложение «Путеводитель по Красноярскому краю», ориентированное на знакомство пользователей с достопримечательностями и на исследование менее известных мест региона.

Объект исследования – путеводитель для информационного сопровождения по Красноярскому краю.

Предмет исследования – процесс проектирования и разработки web-приложения, ориентированного на предоставление туристической информации.

Для достижения поставленной цели необходимо решить следующие задачи:

– на основе анализа туристической отрасли Красноярского края и существующих решений определить требования к функциональности web-приложения;

– разработать структуру и интерфейс web-приложения; средствами JavaScript реализовать навигацию, фильтрацию, голосового помощника и поиск объектов; настроить взаимодействие с голосовым API через Node.js сервер;

– провести тестирование работоспособности web-приложения.

Методы исследования:

– теоретические: анализ научной, учебной и методической литературы по теме web-разработки, клиент-серверной архитектуры, взаимодействия с API, проектирования пользовательских интерфейсов, изучение документации по HTML, CSS, JavaScript, Node.js, Яндекс.Картам и Yandex SpeechKit.

– эмпирические: проектирование макетов интерфейса и структуры web-приложения, реализация интерфейсной и серверной части, функциональное тестирование реализованных компонентов: фильтрации, поиска, голосового помощника, отображения карты и тестирование пользовательских сценариев работы с приложением.

Результаты исследования представлены на следующих научных мероприятиях:

1. VIII Всероссийская научно-практическая конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла», ЛПИ – филиал СФУ (сертификат участника).

2. IV Всероссийской научно-практической конференции «Современное педагогические образование: теоретический и прикладной аспекты» в секции «Информатика, информационные технологии и экономика», ЛПИ – филиал СФУ (диплом II степени).

3. XXXI Всероссийский смотр-конкурс молодежных IT-проектов «SOFT-ПАРАД 2025» (сертификат участника).

По результатам исследования опубликованы статьи в сборниках конференций:

1. Скок, М. М. Основные аспекты разработки веб-приложения для туристов / М. М. Скок/ // Актуальные проблемы преподавания дисциплин естественнонаучного цикла, Тезисы докладов VIII Всероссийской научно-практической конференции преподавателей, учителей, студентов и молодых ученых, ЛПИ – филиал СФУ – Лесосибирск.

2. Скок, М. М. Разработка web-приложения «Помощник путешественника по Красноярскому краю» / М. М. Скок/ // Тезисы докладов IV Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты», ЛПИ – филиал СФУ – Лесосибирск.

Структура работы – работа состоит из введения, трёх глав, заключения, приложения и списка использованных источников, включающего 40 наименований. Результаты работы представлены в 39 иллюстрациях и 4 таблицах. Общий объем работы – 74 страниц.

1 Теоретические основы проектирования web-приложения

1.1 Основы web-технологий: HTML, CSS и применение JavaScript

Современные web-приложения основываются на трёх базовых технологиях: HTML, CSS и JavaScript. Их совместное применение позволяет создавать функциональные, визуально привлекательные и интерактивные интерфейсы, доступные пользователям в браузере без необходимости установки дополнительного программного обеспечения.

HTML (HyperText Markup Language) – это язык разметки, предназначенный для структурирования содержимого web-страницы. Он определяет семантику документа: заголовки, абзацы, списки, изображения, таблицы и другие элементы. В HTML используются теги, которые описывают структуру информации. Версия HTML5, принятая в 2014 году, ввела множество новых элементов (<section>, <article>, <nav>, <video>, <canvas>) и API, что расширило возможности взаимодействия с web-документами [10, 38].

CSS (Cascading Style Sheets) – каскадные таблицы стилей – позволяют управлять визуальным представлением HTML-документа. С помощью CSS можно задавать размеры элементов, их цвет, расположение, анимации и адаптивность интерфейса для различных устройств. CSS делится на несколько уровней: CSS1, CSS2, CSS3, каждый из которых вносил улучшения и расширения. Особенно значимыми стали технологии Flexbox и CSS Grid, позволяющие создавать сложные адаптивные макеты [8].

Совокупное использование HTML, CSS и JavaScript образует основу фронтенд-разработки. Эти технологии являются стандартом индустрии и поддерживаются всеми современными web-браузерами. Их преимущество заключается в кроссплатформенности: web-приложения можно запускать на любом устройстве с установленным браузером – компьютере, планшете, смартфоне.

С технической точки зрения, HTML создаёт структуру страницы, CSS отвечает за её внешний вид, а JavaScript – за динамику и поведение. Например, при реализации фильтрации достопримечательностей по категориям HTML содержит

блоки с описаниями объектов, CSS отвечает за стилизацию карточек, а JavaScript скрывает или отображает нужные элементы в зависимости от выбора пользователя.

JavaScript – язык программирования, предназначенный для реализации логики поведения web-страницы. Он позволяет реагировать на действия пользователя, изменять содержимое страницы без её перезагрузки, взаимодействовать с сервером и работать с мультимедиа, геолокацией, локальным хранилищем и другими API браузера. JavaScript выполняется непосредственно в браузере и обеспечивает высокий уровень интерактивности [31, 9].

JavaScript выполняется непосредственно в браузере, что позволяет реализовывать отклик на действия пользователя в режиме реального времени. Это делает JavaScript незаменимым при создании пользовательских интерфейсов, особенно в проектах, где важна высокая степень интерактивности.

Возможности JavaScript представлены в таблице 1.

Таблица 1 – Основные возможности JavaScript в web-приложениях

Возможность	**Описание**
Работа с DOM	Изменение структуры, атрибутов и содержимого HTML-страницы динамически.
Обработка событий	Реакция на действия пользователя: нажатия, наведение, ввод с клавиатуры и т. д.
Работа с формами	Проверка введённых данных, отправка на сервер, вывод подсказок и сообщений об ошибках.
AJAX и Fetch API	Асинхронные запросы к серверу без перезагрузки страницы.
Анимации и визуальные эффекты	Создание простых и сложных анимаций, плавных переходов между состояниями.
Работа с API браузера	Доступ к микрофону, геолокации, хранилищу и другим функциям HTML5.
Создание модальных окон и всплывающих подсказок	Повышение удобства и интерактивности интерфейса.

JavaScript позволяет не только реализовать сложное поведение элементов на странице, но и создавать полноценную логику web-приложения. В рамках выпускной квалификационной работы язык JavaScript используется для реализации следующих компонентов:

– Поиск по ключевым словам, реализованный в модуле search.js. Скрипт считывает поисковой запрос из URL, отображает его и фильтрует объекты по названию и описанию.

– Фильтрация по городам, реализованная в файле filter.js. Пользователь выбирает нужные чекбоксы, а затем при клике на кнопку «Показать» выполняется фильтрация карточек featured-item по значению data-city. Дополнительно реализована функция «Выбрать все».

– Голосовой помощник разработан с использованием файлов voice.js и server.js. Пользователь нажимает на кнопку, и текст с экрана передаётся на локальный Node.js-сервер (server.js). Сервер делает запрос к Yandex SpeechKit API для синтеза речи и возвращает аудиофайл, который воспроизводится в браузере [12].

– Логика управления аудиопроигрыванием реализована в voice.js, в которой предусмотрены состояния воспроизведения (воспроизведение, пауза, повтор), визуальные обновления кнопки.

– Взаимодействие с сервером реализуется с использованием fetch. Это позволяет получать аудиофайлы в формате MP3, не перезагружая страницу.

Схема применения JavaScript в web-приложении представлена на рисунке 1:

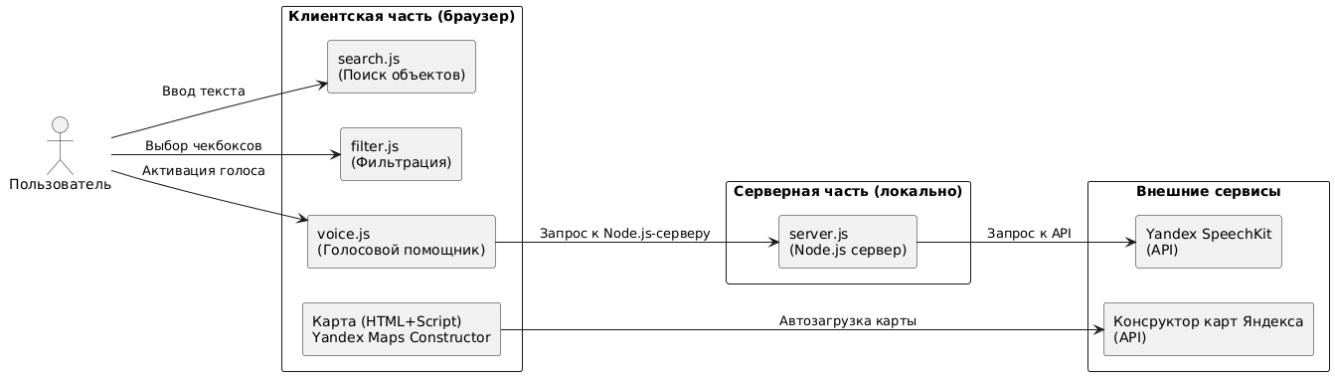


Рисунок 1 – Схема применения JavaScript в web-приложении

JavaScript выступает ключевым элементом в реализации интерактивных функций web-приложения. Он позволяет повысить удобство использования web-

приложения и существенно расширяет его функциональные возможности. Использование JavaScript в сочетании с HTML и CSS обеспечивает полноценную клиентскую логику без необходимости обращения к тяжёлым фреймворкам, что особенно важно при разработке лёгкого, автономного и доступного путеводителя.

1.2 Архитектура web-приложений: многостраничный подход

Архитектура web-приложения определяет, каким образом формируется пользовательский интерфейс, как организовано взаимодействие клиента и сервера, и как реализована логика переходов между страницами. Выбор архитектурного подхода оказывает существенное влияние на производительность, удобство использования, масштабируемость и трудоёмкость поддержки проекта. Существуют два основных архитектурных решения: одностраничные (SPA – Single Page Application) и многостраничные (MPA – Multi Page Application) приложения. В рамках исследования реализована архитектура многостраничного web-приложения (MPA).

Многостраничное приложение представляет собой совокупность отдельных HTML-страниц, каждая из которых отвечает за определённый раздел сайта. Когда пользователь переходит с одной страницы на другую, происходит полная перезагрузка содержимого, и браузер загружает новую страницу. Такой подход традиционно используется в большинстве классических web-сайтов – от информационных порталов до интернет-магазинов.

Архитектура MPA предполагает разделение логики web-приложения по физическим страницам, каждая из которых может иметь уникальный URL, структуру, стили и сценарии. Это упрощает навигацию, повышает читаемость кода и облегчает отладку. Кроме того, многостраничные приложения лучше индексируются поисковыми системами, так как каждая страница представляет собой отдельный HTML-документ с собственным содержанием и мета-данными.

Преимущества многостраничной архитектуры:

1. *Простота реализации.* MPA не требует использования клиентских фреймворков (например, React или Vue), что снижает технический порог входа и упрощает поддержку кода;

2. *Разделение логики.* Каждый HTML-документ реализует отдельный функциональный блок, что способствует модульности и повторному использованию компонентов;

3. *SEO-дружелюбность.* Поисковые роботы получают доступ к готовым страницам без необходимости исполнения JavaScript, что улучшает индексацию и ранжирование;

4. *Совместимость.* MPA не требует поддержки современных API браузеров и подходит даже для устаревших систем;

5. *Оффлайн-доступ.* При локальном запуске или использовании офлайн-копии каждая страница доступна отдельно, что делает архитектуру особенно актуальной для проектов, ориентированных на путешественников в удалённых регионах.

Недостатки многостраничной архитектуры:

1. *Повторная загрузка ресурсов.* При переходах между страницами перезагружаются общие ресурсы (стили, скрипты), что может снизить производительность;

2. *Дольше отклик при навигации.* По сравнению с одностраничными приложениями MPA работают чуть медленнее, так как не используют асинхронную подгрузку контента;

3. *Сложности при добавлении интерактивности.* Для взаимодействия между страницами приходится использовать дополнительные механизмы (например, параметры URL или cookie).

Несмотря на эти ограничения, архитектура MPA является наиболее рациональным выбором. Web-приложение представляет собой набор HTML-страниц, каждая из которых реализует отдельную функциональность.

Каждая страница стилизована с помощью CSS и содержит собственные JavaScript-модули, реализующие функции фильтрации, поиска, отображения карты и голосового помощника. Подобная организация кода обеспечивает удобную структуру, модульность и облегчает дальнейшую поддержку проекта.

На рисунке 2 представлена типовая структура многостраничного приложения. Каждая HTML-страница загружается отдельно при переходах, подключает собственные CSS и JavaScript файлы, а также может обращаться к серверу или API при необходимости. Такой подход обеспечивает простоту, модульность и предсказуемость архитектуры:

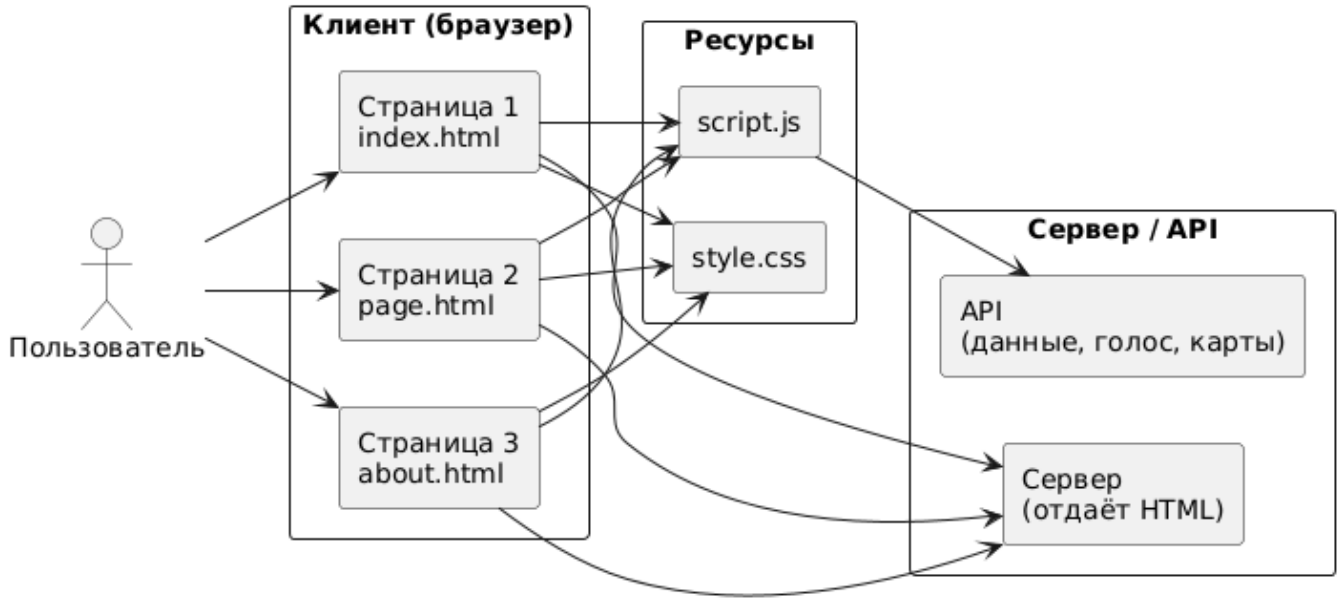


Рисунок 2 – Общая структура многостраничного web-приложения (MPA)

Выбранный архитектурный подход позволяет развернуть сайт как офлайн-приложение, что особенно важно для пользователей, находящихся вне зоны стабильного интернета. При этом страницы могут кэшироваться локально и быть доступны при необходимости без подключения к сети.

Таким образом, многостраничная архитектура соответствует целям работы и обеспечивает надёжную навигацию, логическую структуру, а также высокую доступность информации при минимальной технической сложности реализации.

1.3 Локальные и удалённые API: принципы работы

Современные web-приложения всё чаще строятся на модульной архитектуре с активным использованием API – интерфейсов прикладного программирования (от англ. Application Programming Interface). API позволяют различным программным компонентам взаимодействовать друг с другом, упрощая структуру кода и обеспечивая масштабируемость решений.

API можно разделить на локальные (внутренние) и удалённые (внешние). Локальные API работают на стороне сервера или клиента и используются внутри приложения. Удалённые API предоставляются сторонними сервисами через интернет и требуют сетевого доступа.

API обычно работают по протоколу HTTP/HTTPS, используя методы GET, POST, PUT, DELETE. Web-клиенты (например, JavaScript в браузере) могут отправлять запросы к API, получая или передавая данные. Для обмена чаще всего используется формат JSON или URL-кодированные параметры.

В таблице 2 представлены основные различия между локальных и удалённых API.

Таблица 2 – Сравнение локальных и удалённых API

Характеристика	Локальный API	Удалённый API
Расположение	На сервере или в том же приложении	Внешние сервисы (облако, сторонние платформы)
Скорость отклика	Быстрая (локальная сеть)	Зависит от интернета и удалённого сервера
Пример использования	Node.js сервер (server.js)	Yandex SpeechKit
Доступность офлайн	Возможна	Требует интернет соединения
Уровень контроля	Полный (разработка и поддержка)	Ограниченный

В рамках исследования реализовано взаимодействие с двумя типами API: удалённый API синтеза речи (Yandex SpeechKit) и удалённый API картографического сервиса (Яндекс.Карты)

Для голосового помощника web-приложение использует связку из клиентского JavaScript-модуля voice.js и локального сервера server.js,

разработанного на Node.js. Пользователь инициирует голосовой вывод описания объекта путём нажатия на кнопку «Голосовой помощник». Процесс реализуется следующим образом:

1. voice.js формирует POST-запрос, содержащий текст из DOM-элемента (#textToSpeak).

2. Запрос отправляется на локальный сервер (http://localhost:3000/tts), обрабатываемый скриптом server.js.

3. Сервер пересылает данные на удалённый API Yandex SpeechKit, используя токен и параметры (язык, голос, формат).

4. API Яндекса возвращает аудиофайл в формате MP3.

5. Сервер пересылает аудиопоток обратно в браузер, где воспроизводится.

Отображение интерактивной карты реализовано с использованием внешнего API Яндекс Карт, подключаемого в HTML-документ в виде виджета. Виджет создаётся заранее через визуальный Конструктор карт Яндекса и не требует дополнительной логики на стороне JavaScript или Node.js. Это решение удобно для работы, так как требуется быстрое и визуально качественное отображение географических объектов.

Использование двух различных API – для речи и карт позволило реализовать:

– Преобразование текстовой информации в речь;

– Отображение географических объектов на страницах web-приложения.

API значительно улучшит пользовательский опыт и расширит функциональные возможности web-приложения.

1.4 Роль серверной части на Node.js в клиент-серверной архитектуре

Клиент-серверная архитектура – одна из базовых моделей взаимодействия в web-приложении. Она предполагает наличие двух независимых компонентов:

– Клиентская часть (frontend) – отображает интерфейс и обрабатывает взаимодействие с пользователем;

– Серверная часть (backend) – обрабатывает запросы, выполняет бизнес-логику, обращается к базам данных или внешним API.

Сервер является промежуточным звеном, которое принимает запросы от клиента, обрабатывает данные и возвращает результат. Такой подход позволяет изолировать конфиденциальные операции (например, хранение токенов API), оптимизировать структуру кода и повысить безопасность [6].

Node.js – это среда выполнения JavaScript вне браузера, построенная на движке V8 от Google. Она обеспечивает высокую производительность и масштабируемость благодаря неблокирующей архитектуре [40].

Node.js активно используется в современных web-приложениях благодаря следующим преимуществам:

– Единый язык разработки (JavaScript) на клиенте и сервере;

– Высокая производительность при работе с сетевыми запросами;

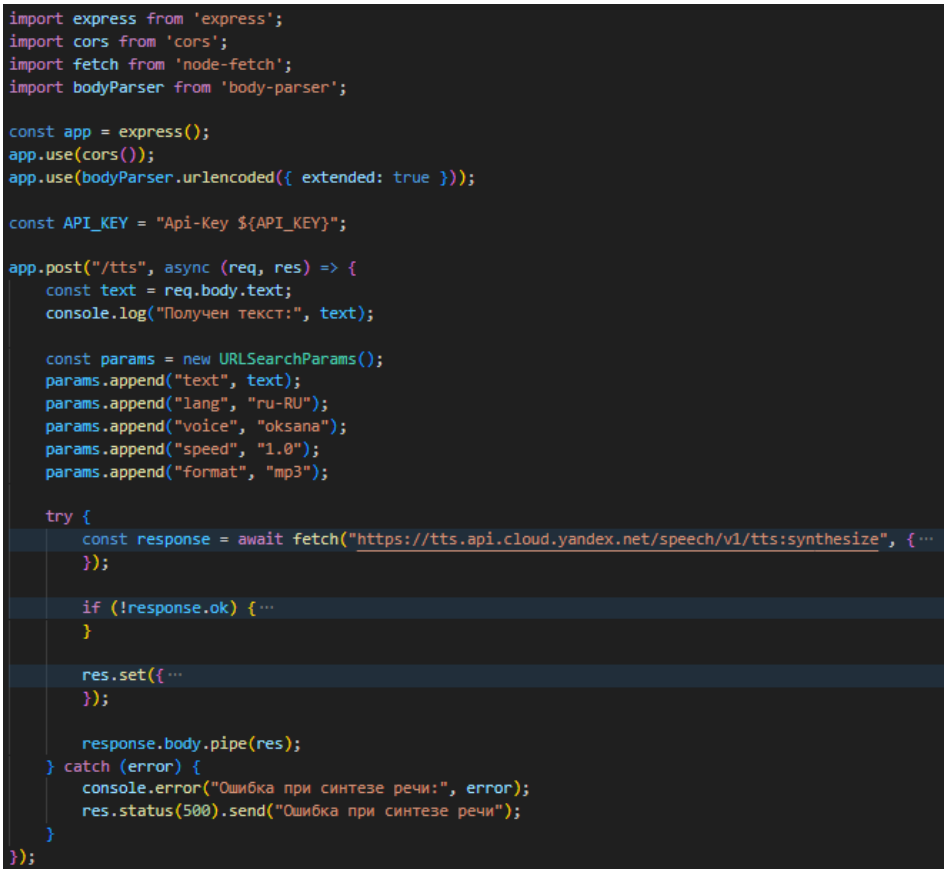
– Наличие большого количества библиотек в экосистеме;

– Поддержка асинхронных операций и событийно-ориентированной модели.

В рамках исследования Node.js выполняет роль промежуточного сервера между клиентским приложением и внешним API Yandex SpeechKit. Таким образом, Node.js позволил нам защитить API-ключ от утечки (ключ хранится на сервере), обработать параметры запроса (язык, формат, голос), преобразовать ответ от стороннего API в потоковый формат и вернуть клиенту аудиофайл без необходимости хранения на сервере.

Выбор Node.js для серверной части обусловлен рядом преимуществ. Во-первых, единый язык программирования для клиентской и серверной сторон упрощает сопровождение и ускоряет разработку. Во-вторых, Node.js обеспечивает высокую производительность и подходит для работы с HTTP-запросами и внешними API, включая потоковую передачу данных. В-третьих, сервер позволяет скрыть API-ключи от клиента, обеспечивая безопасность. Благодаря простоте запуска и доступности большого числа библиотек, Node.js упрощает разработку и расширение функционала [26].

На рисунке 3 представлен фрагмент кода, реализующий обработку POST-запроса к Yandex SpeechKit:



```
import express from 'express';
import cors from 'cors';
import fetch from 'node-fetch';
import bodyParser from 'body-parser';

const app = express();
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));

const API_KEY = "Api-Key ${API_KEY}";

app.post("/tts", async (req, res) => {
    const text = req.body.text;
    console.log("Получен текст:", text);

    const params = new URLSearchParams();
    params.append("text", text);
    params.append("lang", "ru-RU");
    params.append("voice", "oksana");
    params.append("speed", "1.0");
    params.append("format", "mp3");

    try {
        const response = await fetch("https://tts.api.cloud.yandex.net/speech/v1/tts:synthesize", {···
        });

        if (!response.ok) {···
        }

        res.set({···
        });

        response.body.pipe(res);
    } catch (error) {
        console.error("Ошибка при синтезе речи:", error);
        res.status(500).send("Ошибка при синтезе речи");
    }
});
```

Рисунок 3 – Фрагмент кода (server.js)

Этот код демонстрирует, как сервер принимает текст от пользователя, пересылает его в Yandex SpeechKit API и возвращает результат в виде MP3-аудиопотока без промежуточного сохранения файла. Такое решение является эффективным, безопасным и технологически современным.

Использование Node.js позволило реализовать компактную, управляемую и безопасную серверную часть в web-приложении. Благодаря асинхронной архитектуре и возможности работы с внешними API, Node.js идеально подошёл для задач озвучивания текста в браузере через голосового помощника. Это решение оптимизирует распределение задач между клиентом и сервером, и создаёт основу для дальнейшего расширения возможностей web-приложения.

2 Проектирование путеводителя по Красноярскому краю

2.1 Анализ целевой аудитории и существующих решений

Разработка web-приложения, ориентированного на массовое использование, начинается с анализа предполагаемой аудитории. Это позволяет адаптировать не только интерфейс, но и функциональность, структуру контента и способы взаимодействия с пользователем.

Целевой аудиторией путеводителя по Красноярскому краю являются пользователи, заинтересованные в ознакомлении с достопримечательностями того или иного региона. Основу аудитории составляют:

– Туристы и гости края, приезжающие в командировки, на отдых или на экскурсии;

– Жители Красноярского края, желающие открыть для себя новые локации;

– Студенты или школьники, которым необходим наглядный материал для учебных проектов;

– Преподаватели, экскурсоводы, краеведы, использующие структурированную информацию.

На основе этих групп можно выделить несколько типичных портретов пользователей, которые отображены на рисунке 4:

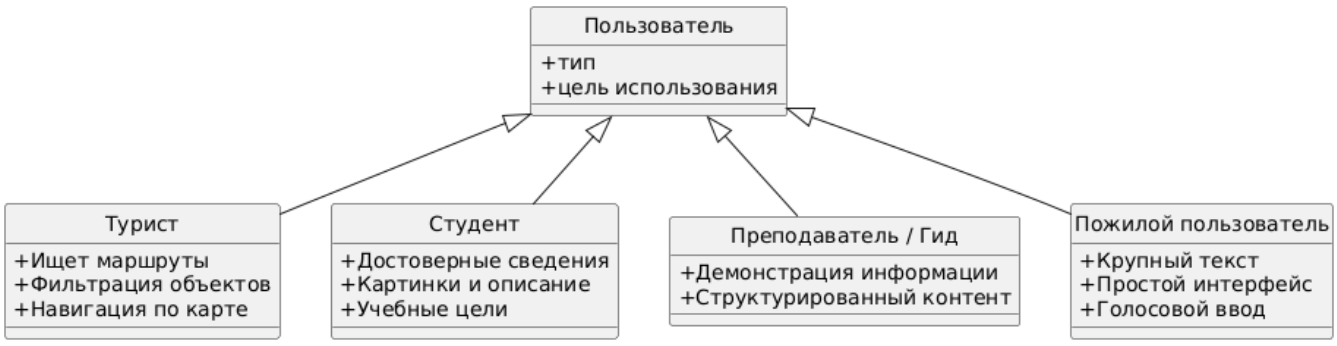


Рисунок 4 – Целевая аудитория web-приложения-путеводителя

Основываясь на диаграмме определились ключевые потребности и требования целевой аудитории:

– Простота интерфейса – минимум отвлекающих элементов, логичная структура страниц, крупные кнопки и понятные подписи;

– Доступ к информации без технических барьеров – web-приложение должно работать в любом современном браузере без необходимости установки дополнительных компонентов;

– Интерактивность – возможность искать объекты, фильтровать их по категориям, взаимодействие с картой и голосовым помощником;

– Географическая наглядность – карта с расположением объектов;

– Возможность голосового взаимодействия – особенно актуально для пользователей с ограничениями зрения или для быстрой навигации.

В процессе анализа существующих туристических web-приложений рассмотрено решение, представленное на сайте rus-trip.ru. Ресурс предлагает пользователям информацию о маршрутах, достопримечательностях и культурных особенностях различных регионов России. Основной акцент сделан на предоставление текстовой информации и изображений, что обеспечивает базовое представление о туристических объектах. На рисунке 5 представлен скриншот страницы города Красноярска:

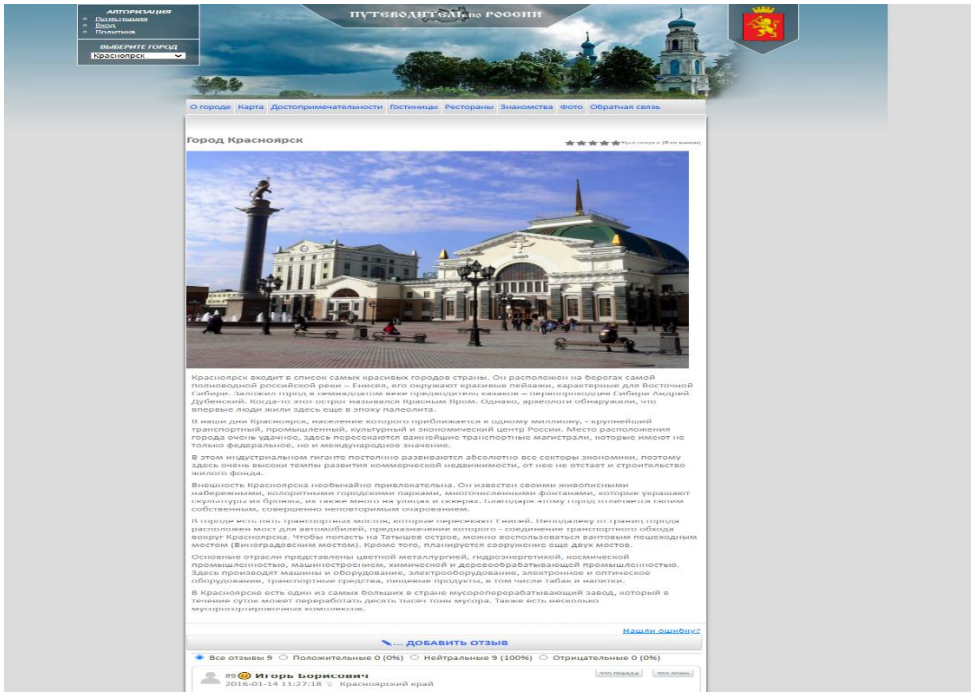


Рисунок 5 – Скриншот с сайта rus-trip.ru

В отличие от rus-trip.ru, разработанное web-приложение ориентировано на интерактивное взаимодействие с пользователем. Ключевыми отличиями являются:

– *Страницы объектов:* возможность просмотра объекта на карте, с возможностью построения маршрута и удобной навигацией по странице;

– *Голосовой помощник:* интеграция с Yandex SpeechKit для озвучивания информации, что повышает доступность для пользователей с ограничениями по зрению;

– *Многостраничная архитектура (MPA):* каждая страница отвечает за отдельный функциональный блок, что упрощает навигацию и поддержку кода.

Таким образом, web-приложение предоставляет динамичный и персонализированный опыт для пользователя, что особенно актуально в современных условиях цифровизации туристических услуг. А анализ аудитории позволяет сформировать техническое и визуальное задание на разработку. В следующих параграфах будет рассмотрено, каким образом эти требования воплощаются в архитектуре, интерфейсе и функциональности электронного путеводителя по Красноярскому краю.

2.2 Требования к функциональности web-приложения

Проектирование web-приложения начинается с постановки функциональных требований, вытекающих из анализа целевой аудитории. Эти требования определяют, какие действия должен уметь выполнять пользователь и какие функции должен реализовать сайт для их поддержки [18].

Функциональные возможности делятся на основные и вспомогательные (улучшающие пользовательский опыт, но не критичны для базовой работы web-приложения).

Основные функциональные требования:

1. Просмотр списка городов, культурных достопримечательностей, природы, развлечений – карточки с названием, описанием и изображением.

2. Поиск по ключевым словам – ввод строки и динамическое отображение результатов.

3. Фильтрация по категориям – сортировка объектов по городам (природа, культура, развлечения).

4. Просмотр подробной информации – переход на отдельную страницу объекта с расширенным описанием, интерактивной картой и другими объектами.

5. Интерактивная карта – отображение объектов на карте с возможностью масштабирования.

6. Голосовой помощник – озвучивание информации по нажатию кнопки.

Вспомогательные функции:

1. Навигационное меню – быстрый переход по разделам web-приложения (города, природа, культура и т.д.).

2. Поддержка локального запуска – возможность использовать web-приложение без подключения к интернету.

На рисунках 6, 7, 8 и 9 представлены ключевые элементы функциональности web-приложения: навигация и поиск, система фильтрации по категориям, использование голосового помощника (TTS API), а также интеграция карты, обеспечивающей визуальное представление объектов:

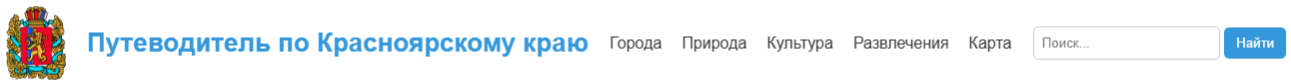


Рисунок 6 – Шапка web-приложения с логотипом, меню и поиском

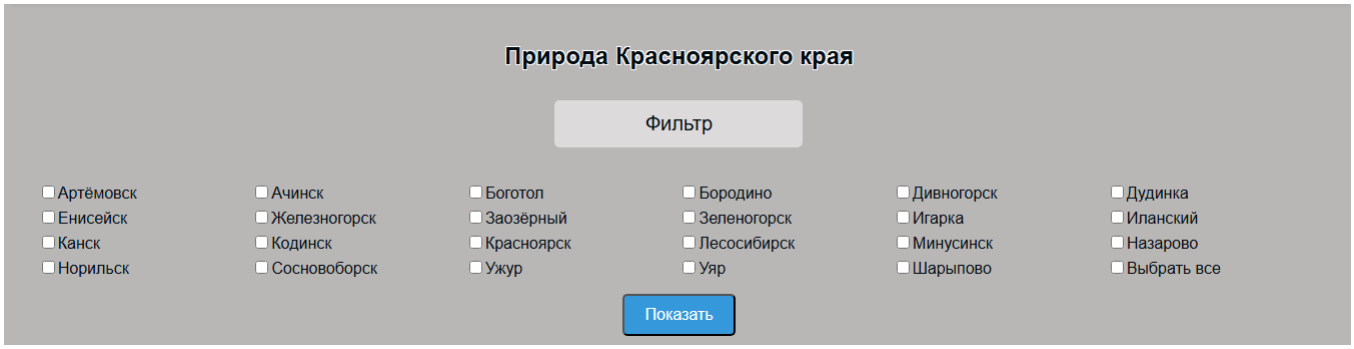


Рисунок 7 – Блок с чекбоксами для выбора категории

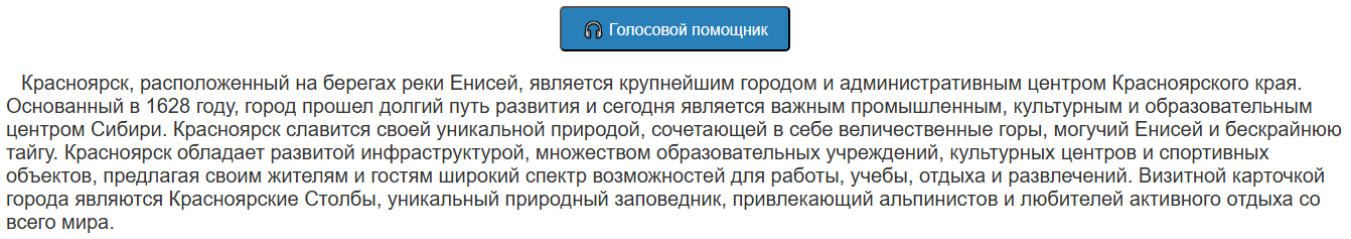


Рисунок 8 – Описание объекта с кнопкой голосового помощника

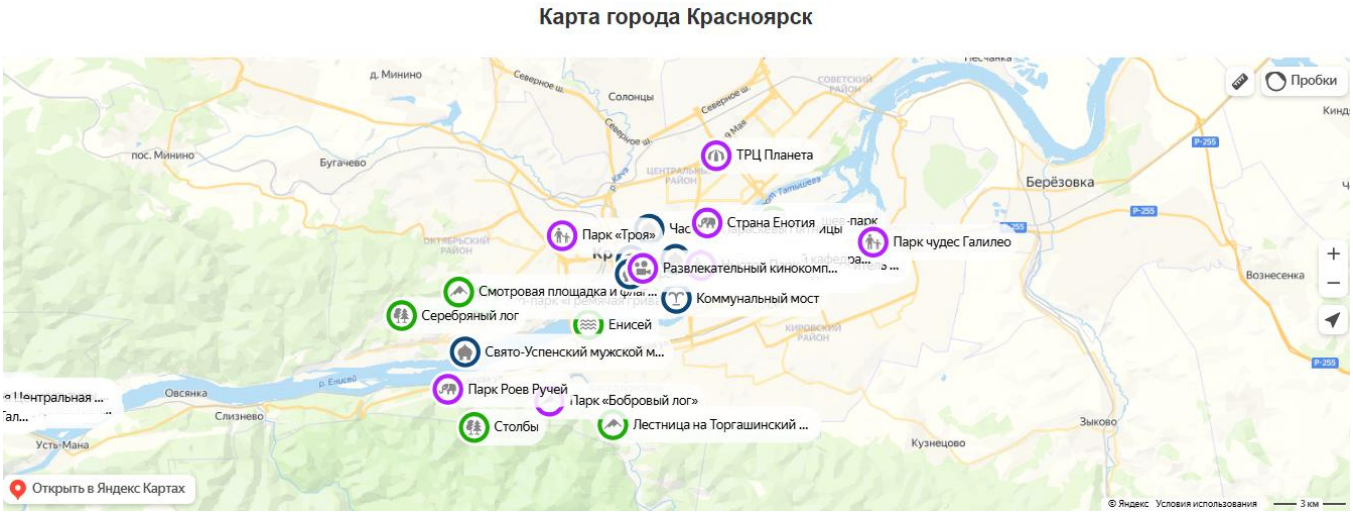


Рисунок 9 – Встроенная карта с объектами

Каждое требование связано с конкретной группой пользователей – фильтрация и карта необходимы туристам для планирования маршрутов, голосовой помощник улучшает доступность для пожилых людей или слабовидящих, поиск – для студентов и преподавателей, работающих с учебным материалом, отдельные страницы с описаниями – для всех, кто хочет прочитать дополнительную информацию об интересующем объекте.

Сформулированные функциональные требования определяют архитектуру сайта, структуру интерфейсов и выбор технологий. Их реализация рассмотрена в следующих разделах, где описывается структура web-приложения, проектирование страниц и механизмы клиент-серверного взаимодействия.

2.3 Структура web-приложения «Путеводитель по Красноярскому краю»

Структура web-приложения определяет организацию навигации, размещение информации и взаимодействие между разделами. Для путеводителя по Красноярскому краю логика построения должна быть максимально интуитивной и удобной для пользователей с разным уровнем цифровой подготовки.

Web-приложение реализовано по принципу многостраничной архитектуры (MPA), где каждая HTML-страница представляет собой отдельный функциональный модуль: главная страница, каталог достопримечательностей (города, культура, природа, развлечения) с фильтрацией, страница с отображением поиска, а также страницы с подробным описанием объектов, голосовым помощником и интерактивной картой.

На рисунке 10 представлена структурная схема web-приложения, отражающая основные разделы и связи между ними.

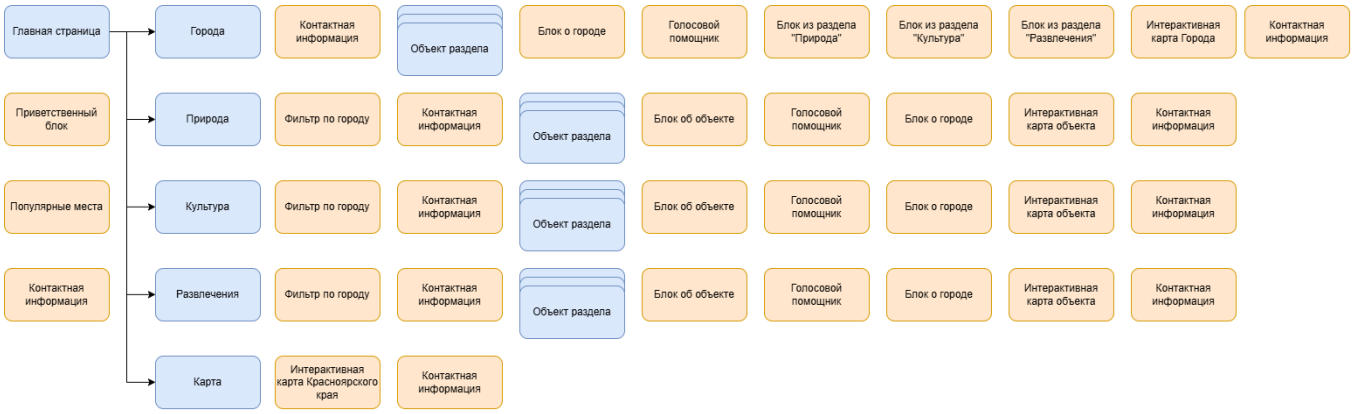


Рисунок 10 – Структурная схема web-приложения

Главная страница выполняет функцию приветственного экрана и точки входа, на которой размещаются: логотип и название проекта, основное меню навигации («Природа», «Культура», «Города», «Развлечения», «Карта»), поисковая строка, приветственный блок, шапка с навигацией перехода к категориям, популярные места, контактная информация.

Каждая категория («Природа», «Культура», «Города», «Развлечения») имеет собственную страницу, на которой представлена сетка карточек. Каждая карточка содержит: изображение объекта, название, краткое описание, кнопку «Подробнее», ведущую на отдельную страницу объекта.

Все категории (кроме категории «Города») оформлены в виде фильтруемых списков, что упрощает навигацию и делает интерфейс удобным даже при большом количестве объектов. Фильтрация по категориям и городам осуществляется через чекбоксы и кнопку «Показать» (реализовано в filter.js). Это позволяет отсекать нерелевантные объекты.

На странице поиска пользователю отображаются результаты, соответствующие введённому запросу. Поиск реализуется с использованием JavaScript (файл search.js) без перезагрузки страницы.

На отдельной странице представлена встроенная интерактивная карта, созданная через Яндекс.Карты с помощью конструктора. На карте размещены метки с указанием достопримечательностей.

Каждый объект имеет собственную страницу, где представлено: полное описание, изображение, кнопка активации голосового помощника, блок с переходами к другим объектам, интерактивная карта, блок с адресом и возможностью построения маршрутов.

Разработанная структура web-приложения обеспечивает простоту навигации, логичное размещение информации и комфортную работу с контентом. Такое построение удобно для пользователей всех возрастов и категорий, включая туристов, учащихся и преподавателей.

Каркас интерфейса (Wireframe) представляет собой схематичное изображение будущих web-страниц без оформления и графики. Основная задача – показать расположение ключевых элементов интерфейса, таких как меню, блоки контента, поиск, фильтры, кнопки [19].

Wireframe используется на этапе проектирования для того, чтобы:

– Логически структурировать информацию на странице;

– Обеспечить удобство пользовательского взаимодействия;

– Заранее оценить, как будет работать навигация и переходы;

– Согласовать структуру с техническими возможностями реализации.

В рамках разработки путеводителя созданы макеты четырёх ключевых страниц: «Главная страница», «Страница каталога объектов», «Страница города», «Страница объекта».

На рисунке 11 представлены основные элементы главной страницы: шапка web-приложения (логотип, название, меню, поисковая строка), приветственный текст, блок с объектами – интересными местами края, подвал с контактной информацией:

Рисунок 11 – Макет главной страницы

На рисунке 12 отображены основные элементы страницы с объектами: шапка web-приложения (логотип, название, меню, поисковая строка), панель фильтрации, сетка карточек объектов с заголовками, описанием и фото, кнопки «Подробнее» на карточках, подвал с контактной информацией:

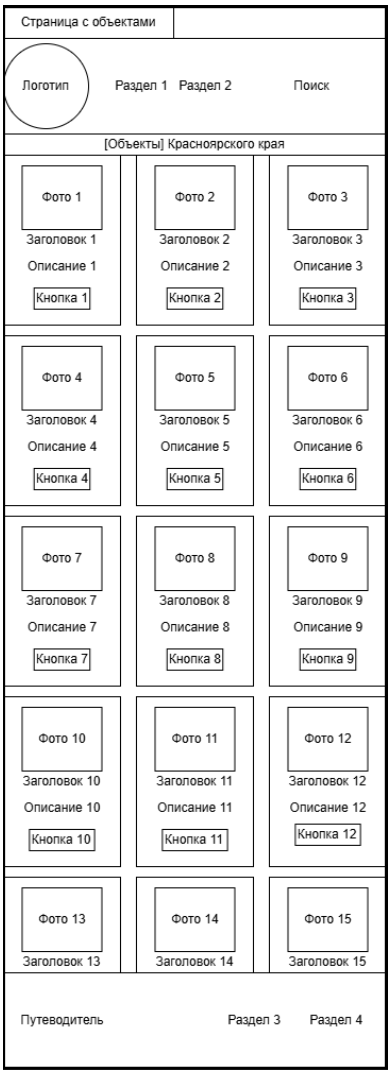


Рисунок 12 – Макет страницы с объектами

На рисунке 13 визуализированы основные элементы страницы города: шапка web-приложения (логотип, название, меню, поисковая строка), фото города, подробное описание, кнопка «Голосовой помощник», навигация к другим объектам, встроенная карта, сетка карточек объектов с заголовками, описанием и фото, кнопки «Подробнее» на карточках, подвал с контактной информацией:

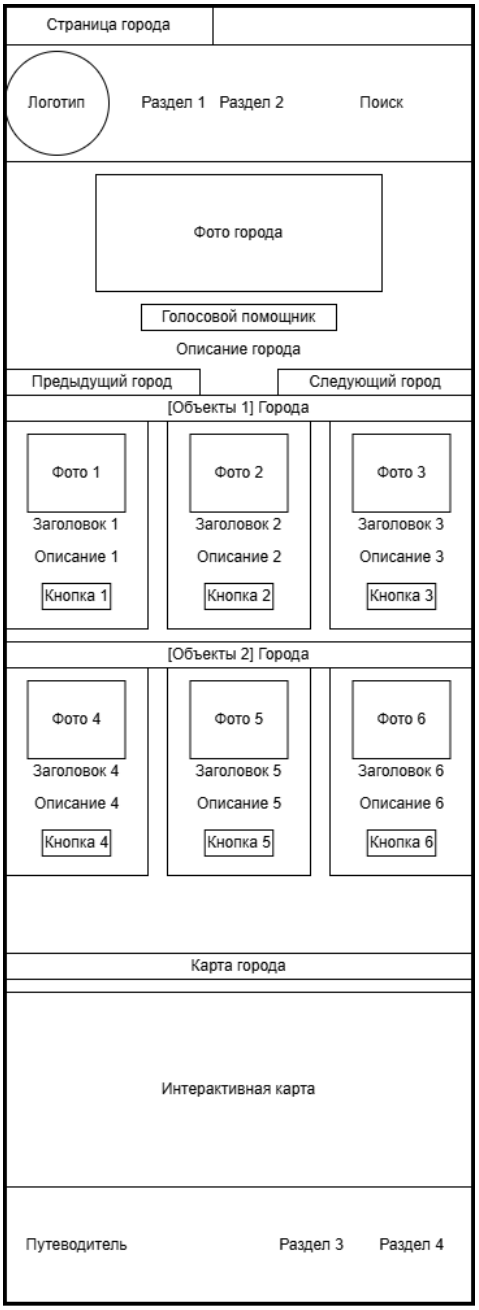


Рисунок 13 – Макет страницы города

На рисунке 14 представлены основные элементы страницы объекта: шапка web-приложения (логотип, название, меню, поисковая строка), изображение объекта, его описание, интерактивная кнопка «Голосовой помощник», карточка города с информацией, блок с адресом, встроенная карта, подвал с контактной информацией:

Рисунок 14 – Макет страницы объекта

Каркасные схемы страниц позволяют на этапе проектирования заранее продумать удобство работы с интерфейсом. Wireframe также облегчает этап верстки и помогает избежать перегрузки страниц избыточными элементами.

2.4 Проектирование голосового помощника и интерактивной карты

Интеграция голосового помощника в web-приложение направлена на расширение доступности и удобства использования нашего web-сервиса. Особенно это актуально для пользователей с ограничениями зрения, пожилых людей, а также для тех, кто предпочитает аудио формат восприятия информации. Внедрение

голосовой помощника повышает интерактивность web-приложения и делает взаимодействие с ним более интересным.

В рамках исследования реализован голосовой помощник, выполняющий озвучивание текстовой информации на странице по запросу пользователя.

Функция голосового помощника активируется вручную с помощью кнопки на странице объекта. Основной сценарий использования включает следующие этапы:

1. Пользователь открывает страницу с описанием достопримечательности или города.

2. Нажимает кнопку «Голосовой помощник» (событие onclick).

3. JavaScript-код (файл voice.js) считывает текст из элемента страницы (#textToSpeak).

4. Сформированный запрос отправляется на локальный сервер Node.js (server.js), размещённый на localhost:3000.

5. Сервер делает POST-запрос в Yandex SpeechKit API, используя токен авторизации и параметры синтеза речи.

6. В ответ возвращается MP3-файл, который немедленно воспроизводится в браузере с помощью HTML5 <audio>.

На рисунке 15 изображена схема сценария взаимодействия пользователя с голосовым помощником:

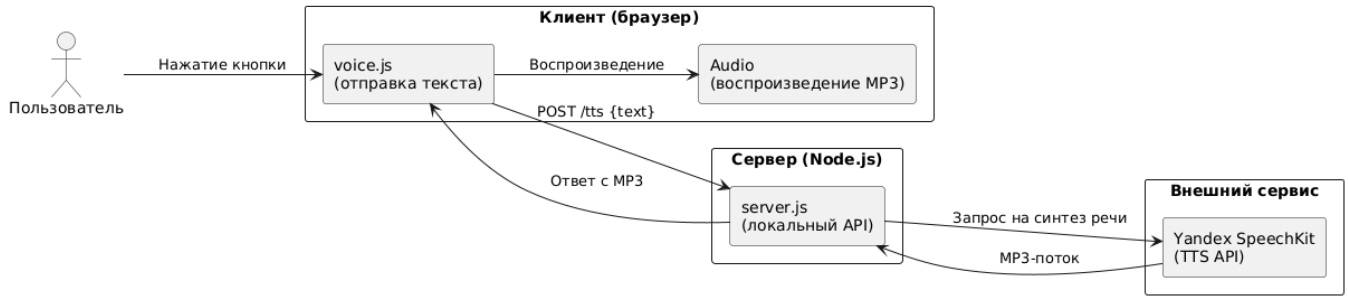


Рисунок 15 – Сценарий взаимодействия с голосовым помощником

Тексты для озвучивания заранее встроены в HTML-страницу как абзацы (<p>), не требуют извлечения из базы данных или сторонних источников. Это

упрощает реализацию, делает помощника автономным и надёжным. В таблице 3 показано как происходит взаимодействие элементов командами в интерфейсе.

Таблица 3 – Основные элементы интерфейса голосового помощника и их функции

Элемент UI	Назначение	Реализовано в
Кнопка «Голосовой помощник»	Запускает процесс озвучивания	voice.js
Текстовый блок	Источник текста для озвучивания	HTML (#textToSpeak)
HTML5 Audio Player	Воспроизводит полученный MP3	voice.js

На рисунке 16 представлен фрагмент реализации кода голосового помощника для электронного путеводителя:

```
const textToSpeak = document.getElementById("textToSpeak");
const speechBtn = document.getElementById("speechBtn");

let audio = null;
let isSpeaking = false;
let isPaused = false;

async function textToSpeechYandex(text) {
    const url = "http://localhost:3000/tts"; // Запрос к локальному серверу

    const params = new URLSearchParams({
        text: text,
        lang: "ru-RU",
        voice: "oksana", // другие: zahar, ermil, jane и др.
        format: "mp3"
    });

    const response = await fetch(url, {
        method: "POST",
        headers: {
            "Content-Type": "application/x-www-form-urlencoded"
        },
        body: params
    });

    if (!response.ok) {···
    }

    const audioBlob = await response.blob();
    const audioUrl = URL.createObjectURL(audioBlob);
    audio = new Audio(audioUrl);
```

Рисунок 16 – Фрагмент реализации кода (voice.js)

Реализация голосового помощника имеет как преимущества, так и ограничения. Преимущества заключаются в следующем: простота и автономность,

отсутствие необходимости в дополнительных наборах инструментов для разработки, защита API-ключа через Node.js, высокое качество синтеза речи (TTS от Яндекса).

Недостатки реализации: нет распознавания речи или интерактивных диалогов, отсутствие кэширования аудио, требуется интернет-соединение для TTS

Голосовой помощник, реализованный в рамках web-приложения, выступает в роли вспомогательного средства, повышающего доступность контента и удобство взаимодействия с web-приложением. Простота реализации и использование проверенных технологий (JavaScript, Node.js, Yandex SpeechKit) делают систему надёжной, автономной и лёгкой для внедрения даже при минимальных технических ресурсах.

Интерактивная карта также является важной частью web-приложения, позволяющая пользователям визуально ориентироваться в пространстве, находить интересующие объекты и быстро получать доступ к их описанию. Для реализации данной функциональности выбрано встраиваемое JavaScript-API от Яндекс.Карт, так как он не требует сложной интеграции, хорошо документирован и предоставляет необходимый набор функций [17].

Карта подключается на каждой HTML-странице с городом или объектом через специальный скрипт-конструктор, который представлен на рисунке 17:

Рисунок 17 – Скрипт подключения карты

В конструкторе:

– <ID> – идентификатор предварительно созданной карты в визуальном конструкторе Яндекс.Карт;

– Все настройки (метки, масштаб, центр) задаются через web-интерфейс конструктора;

– Полученный код вставляется непосредственно в HTML, без необходимости взаимодействия с сервером.

На рисунке 18 изображен интерфейс создания и получение JavaScript кода, который находится на HTML-странице:

Рисунок 18 – Интерфейс Конструктора карт Яндекса

Преимущества выбранного подхода:

1. Простота интеграции – не требует написания JS-кода, карта генерируется через конструктор карт.

2. Визуальная наглядность – пользователь сразу видит местоположение объектов

3. Автономность – может работать при локальной установке web-приложения.

4. Быстрая настройка – добавление новых точек выполняется через интерфейс конструктора

Использование визуального конструктора Яндекс.Карт позволило быстро и эффективно интегрировать карту в web-приложение без каких-либо затрат. Это соответствует концепции лёгкого и доступного интерфейса для широкой аудитории пользователей.

2.5 Сценарии использования

Сценарии использования описывают типовые действия, которые пользователь может выполнять в рамках web-приложения. Эти сценарии необходимы для понимания логики взаимодействия между пользователем и системой, а также служат основой для проектирования структуры интерфейса и определения необходимой функциональности [4, 24].

Для наглядного представления сценариев используется диаграмма вариантов использования (Use Case Diagram), разработанная в соответствии с методологией UML (Unified Modeling Language). Такая диаграмма отображает внешних и внутренних участников системы, а также перечень доступных им действий, обеспечивающих реализацию ключевых пользовательских задач [27].

Выделим три основные категории участников, взаимодействующих с системой:

1. Пользователь – основной участник, взаимодействующий с интерфейсом путеводителя.

2. Локальный сервер – внутренний технический компонент, обрабатывающий TTS-запросы.

3. Внешний API (Яндекс.Карты, Yandex SpeechKit) – дополнительные сервисы, с которыми работает приложение.

На рисунке 19 представлена диаграмма основных сценариев использования web-приложения:

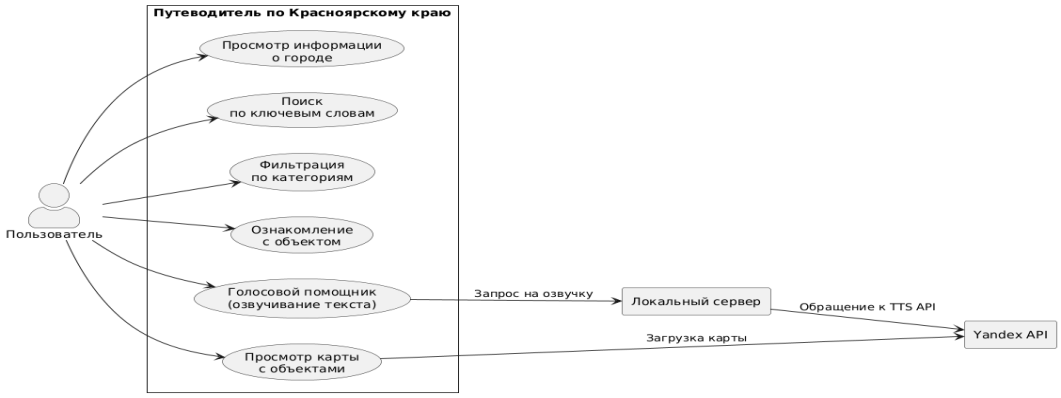


Рисунок 19 – Диаграмма вариантов использования web-приложения

Сценарии использования отражают ключевые задачи, которые выполняет web-приложение: поиск, фильтрация, просмотр объектов, озвучивание текста и навигация по карте. На основе этих сценариев построена вся логика интерфейса и структуры приложения. Диаграмма наглядно показывает, что функциональность web-приложения полностью ориентирована на реальные задачи пользователя, а структура системы формируется исходя из ключевых сценариев взаимодействия.

3 Разработка web-приложения «Путеводитель по Красноярскому краю»

3.1 Создание страниц на HTML и подключение CSS: структура и оформление

Разработка web-приложения начинается с создания HTML-страниц – основного каркаса пользовательского интерфейса. HTML (HyperText Markup Language) позволяет структурировать содержимое, задавая расположение элементов, их семантическую значимость и иерархию. С помощью CSS (Cascading Style Sheets) эта структура получает визуальное оформление: цвет, шрифт, отступы, фон и другие параметры отображения. В данной работе использована связка HTML5 и CSS3 [29].

Каждая страница построена по единому шаблону, обеспечивающему единообразие навигации и восприятия:

– <head>: содержит метаинформацию, заголовок страницы, подключение внешнего файла стилей style.css;

– <header>: включает логотип, название проекта, навигационное меню и строку поиска. Используется класс container для выравнивания содержимого по центру и flex-распределения элементов;

– <main>: основной контент страницы. Здесь размещаются карточки объектов, заголовки разделов, кнопки и изображения;

– <footer>: контактная информация и ссылки на вспомогательные страницы, например, «О сайте» и «Контакты».

В рамках исследования реализовано несколько типов HTML-страниц, каждая из которых выполняет определённую функцию и имеет свой макет. Блоки <header> и <footer> одинаковые у всех страниц, основные различия происходят в блоке <main>. На рисунке 20 представлен фрагмент кода <header> и <footer>:

```
<header>
    <div class="container">
        <img src="img/index/logo.png" alt="Путеводитель по Красноярскому краю">
        <div class="logo">
            <a href="index.html">Путеводитель по Красноярскому краю</a>
        </div>
        <nav>
            <ul>
                <li><a href="cities.html">Города</a></li>
                <li><a href="nature.html">Природа</a></li>
                <li><a href="culture.html">Культура</a></li>
                <li><a href="fun.html">Развлечения</a></li>
                <li><a href="map.html">Карта</a></li>
            </ul>
        </nav>
        <div class="search">
            <form action="search.html" method="GET">
                <input type="text" placeholder="Поиск..." name="q">
                <button type="submit">Найти</button>
            </form>
        </div>
    </div>
</header>

<main>···
</main>
<footer>
    <div class="container">
        <p>&copy; 2025 Путеводитель по Красноярскому краю</p>
        <nav>
            <ul>
                <li><a href="about.html">О сайте</a></li>
                <li><a href="contacts.html">Контакты</a></li>
            </ul>
        </nav>
    </div>
</footer>
```

Рисунок 20 – Фрагмент кода <header> и <footer>

Все страницы используют единый файл стилей style.css, подключаемый в теге <head>: <link rel="tylesheet" href="style.css">

Главная страница (index.html) выполняет функцию приветственного экрана и служит точкой входа в web-приложение. Блок <main> включает два ключевых раздела: <section class="welcome"> – приветственный текст с заголовком и кнопкой «Узнать больше», <section class="featured"> – популярные места Красноярского края, отображаемые в виде карточек. На рисунке 21 показано, как происходит оформление карточки популярного объекта:

```html
<section class="featured">
    <div class="container">
        <h2>Популярные места</h2>
        <div class="featured-grid">
            <div class="featured-item">
                <img src="img/nature/stolb.jpg" alt="Столбы">
                <h3>Столбы</h3>
                <p>Красноярские Столбы – уникальное явление. О Столбах написано множество книг и статей, снято
                    множество фильмов. Им посвящены стихи, о них написаны песни, они служат источником
                    вдохновения художников.</p>
                <a href="nature/stolb.html" class="button">Подробнее</a>
            </div>
            <div class="featured-item">
                <img src="img/cities/yeniseysk.jpg" alt="Енисейск">
                <h3>Енисейск</h3>
                <p>Город в Красноярском крае, расположенный на левом берегу реки Енисей. Население — 17 537 чел.
                    Основан в 1619 году.</p>
                <a href="cities/yeniseysk.html" class="button">Подробнее</a>
            </div>
            <div class="featured-item">
                <img src="img/culture/theatre-square.jpg" alt="Красноярский театр оперы и балета">
                <h3>Красноярский театр оперы и балета</h3>
                <p>В афише театра представлена почти вся мировая оперная и балетная классика. Кроме оперы и
                    балета, в театре проходят концертные программы, творческие встречи с актёрами, спектакли
                    театров из российских городов.</p>
                <a href="culture/theatre-square.html" class="button">Подробнее</a>
            </div>
        </div>
    </div>
</section>
```

Рисунок 21 – Фрагмент кода оформления карточки

Элементы .featured-grid оформлены с помощью CSS-сетки, обеспечивающей равномерное распределение карточек по странице. Скриншот главной страницы приведен на рисунке 22:

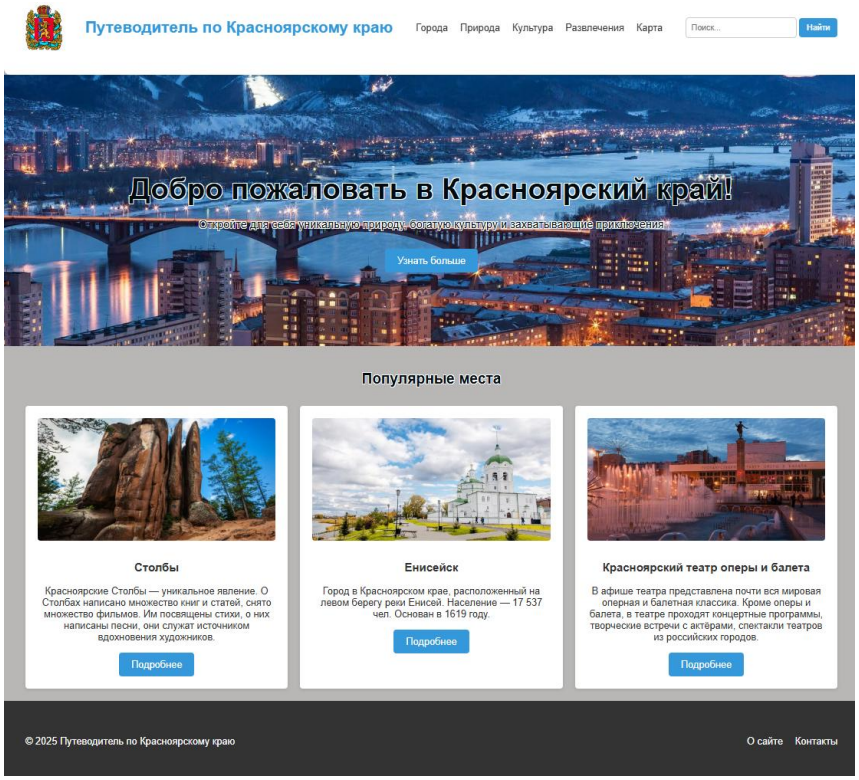


Рисунок 22 – Главная страница web-приложения с карточками популярных объектов

Страница с городами (cities.html) представляет каталог населённых пунктов Красноярского края. Структура страницы аналогична главной, но с акцентом на список городов. Карточка города оформляется таким же образом, как и на главной странице, каждая из которых содержит фото, название, краткое описание и кнопку перехода. Скриншот страницы с городами представлен на рисунке 23:

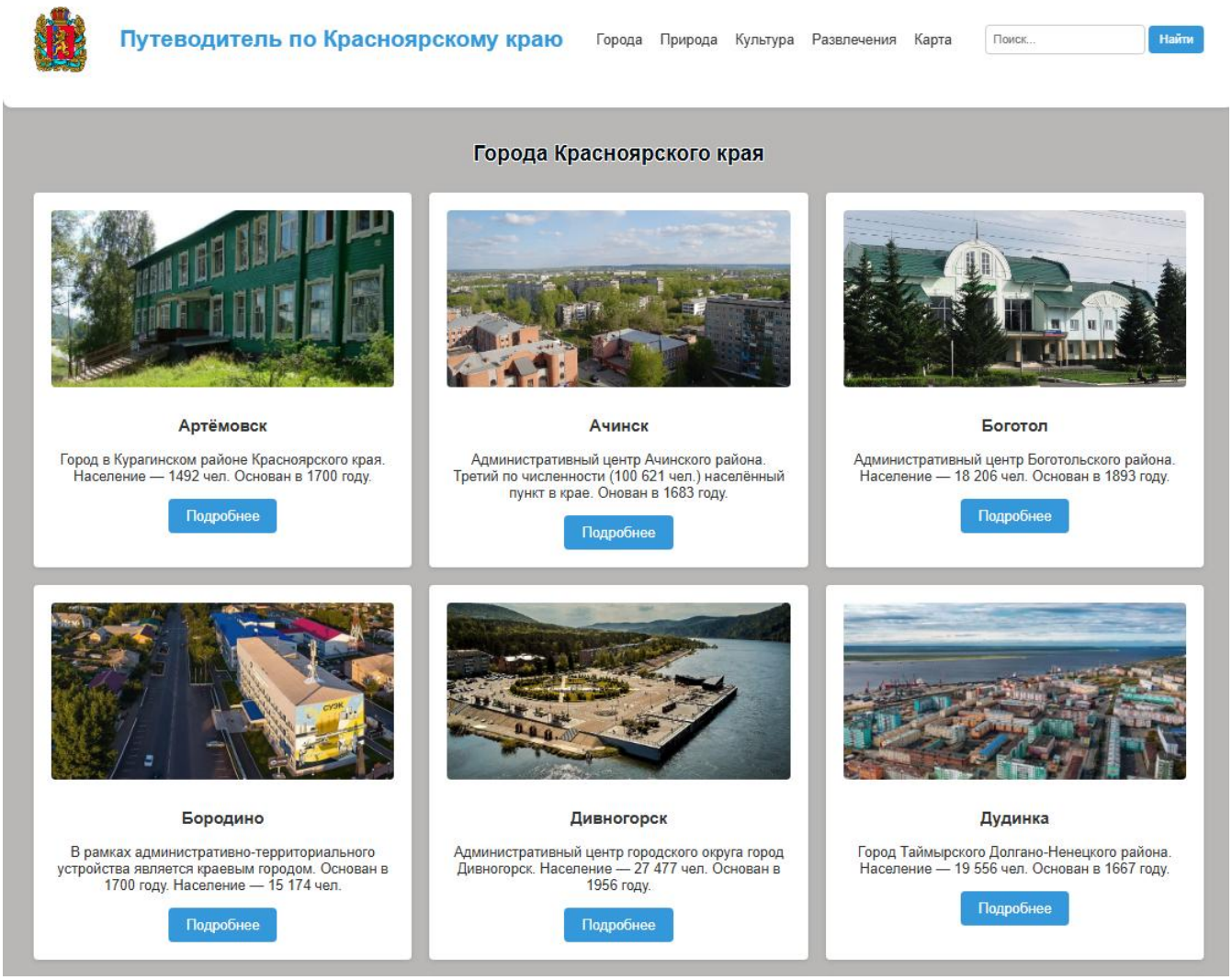


Рисунок 23 – Страница с городами Красноярского края

Каждая тематическая категория (культура, природа, развлечения) реализована как отдельная страница (culture.html, nature.html, fun.html). Структура аналогична структуре страницы с городами, главное отличие заключается в том, что на этих страницах используется фильтрация, она реализована через JavaScript-скрипт filter.js, который скрывает/отображает карточки по выбранным чекбоксам. Скриншот страницы «Культура» представлен на рисунке 24:

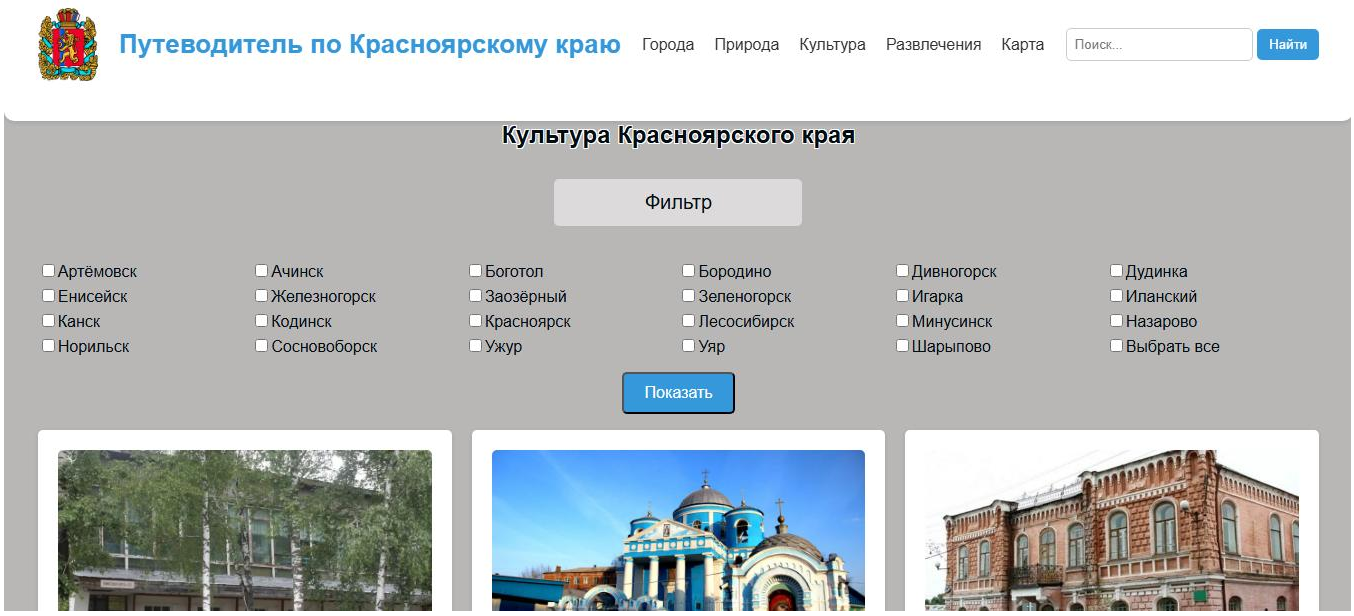


Рисунок 24 – Страница «Культура Красноярского края»

Каждый город реализован как отдельная страница, например, sosnovoborsk.html содержащий: заголовок города, изображение города, полное описание (внутри <p>), кнопка голосового помощника, реализованная с использованием voice.js, ссылки на соседние города: «Предыдущий / Следующий», встроенная карта через API Яндекс.Карт, блоки карточек отдельных категорий (например, «Природа в городе Сосновоорск»). Скриншот отображения карты города представлен на рисунке 25, а скриншот страницы «Сосновоборск» на рисунке 26:

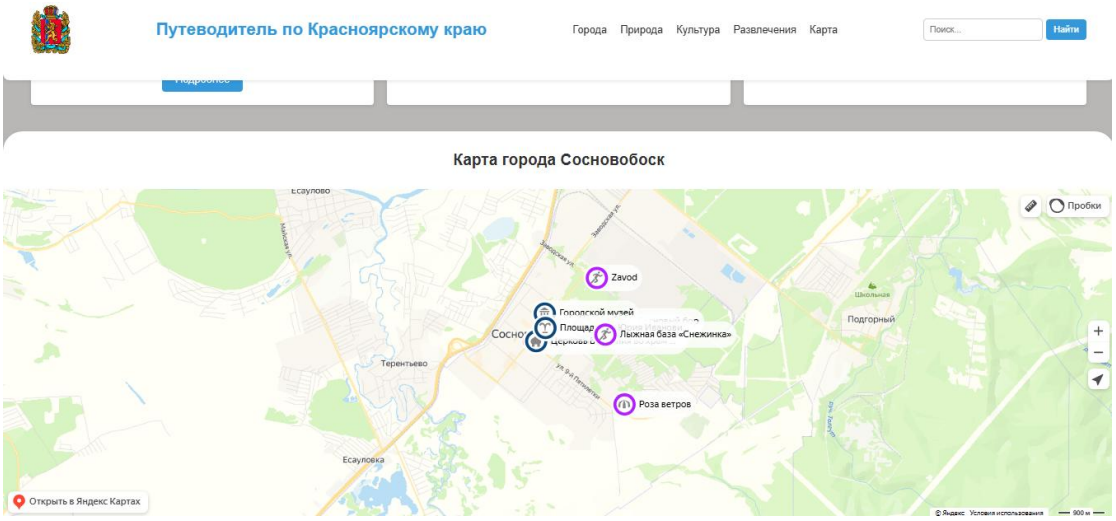


Рисунок 25 – Карта города Сосновоборск

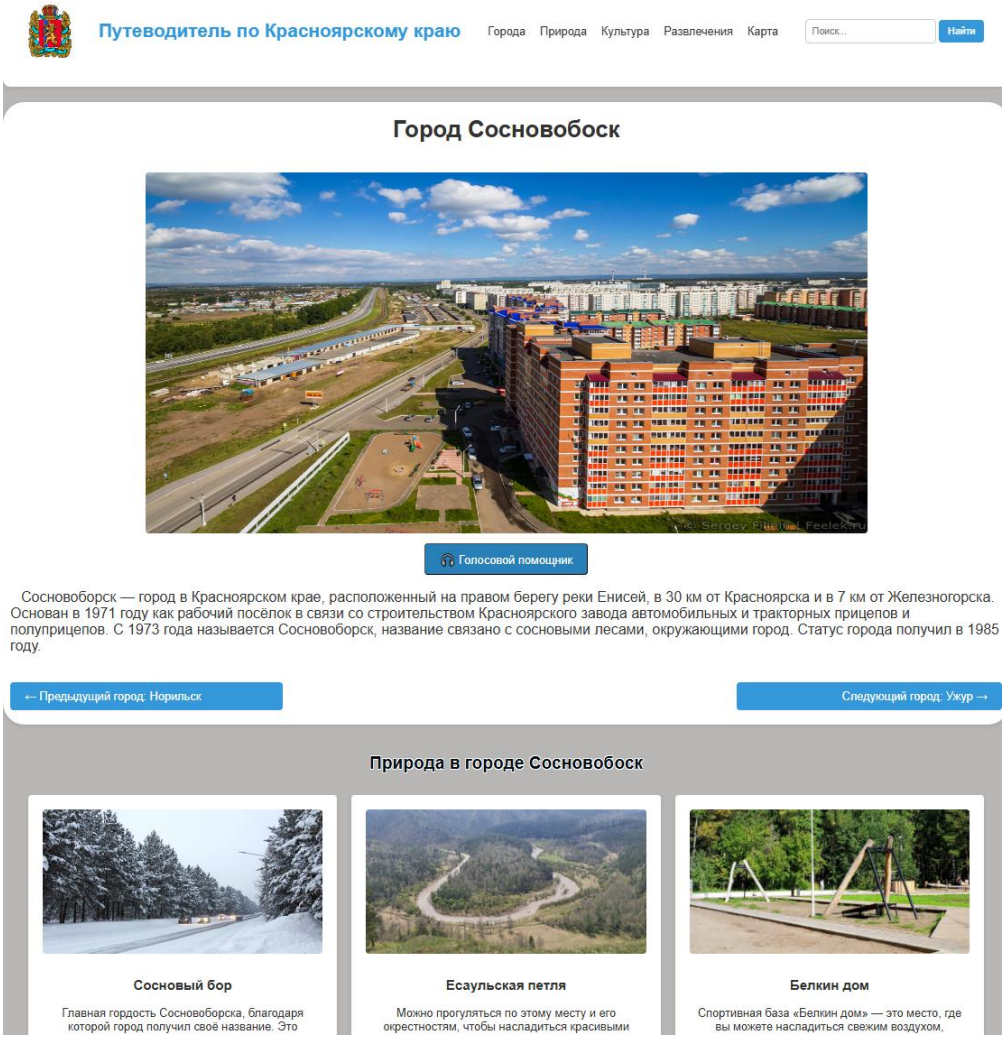


Рисунок 26 – Страница «Сосновоборск»

Каждый объект в населенном пункте реализован как отдельная страница (например, theatre-square.html), которая содержит: заголовок объекта, изображение объекта, полное описание (внутри <p>), кнопка голосового помощника, реализованная с использованием voice.js, блок с адресом и возможностью построить маршрут (нажав на 1 из иконок), блок с городом, встроенная карта через API Яндекс.Карт. Скриншот страницы «Красноярский театр оперы и балеты» представлен на рисунке 27:

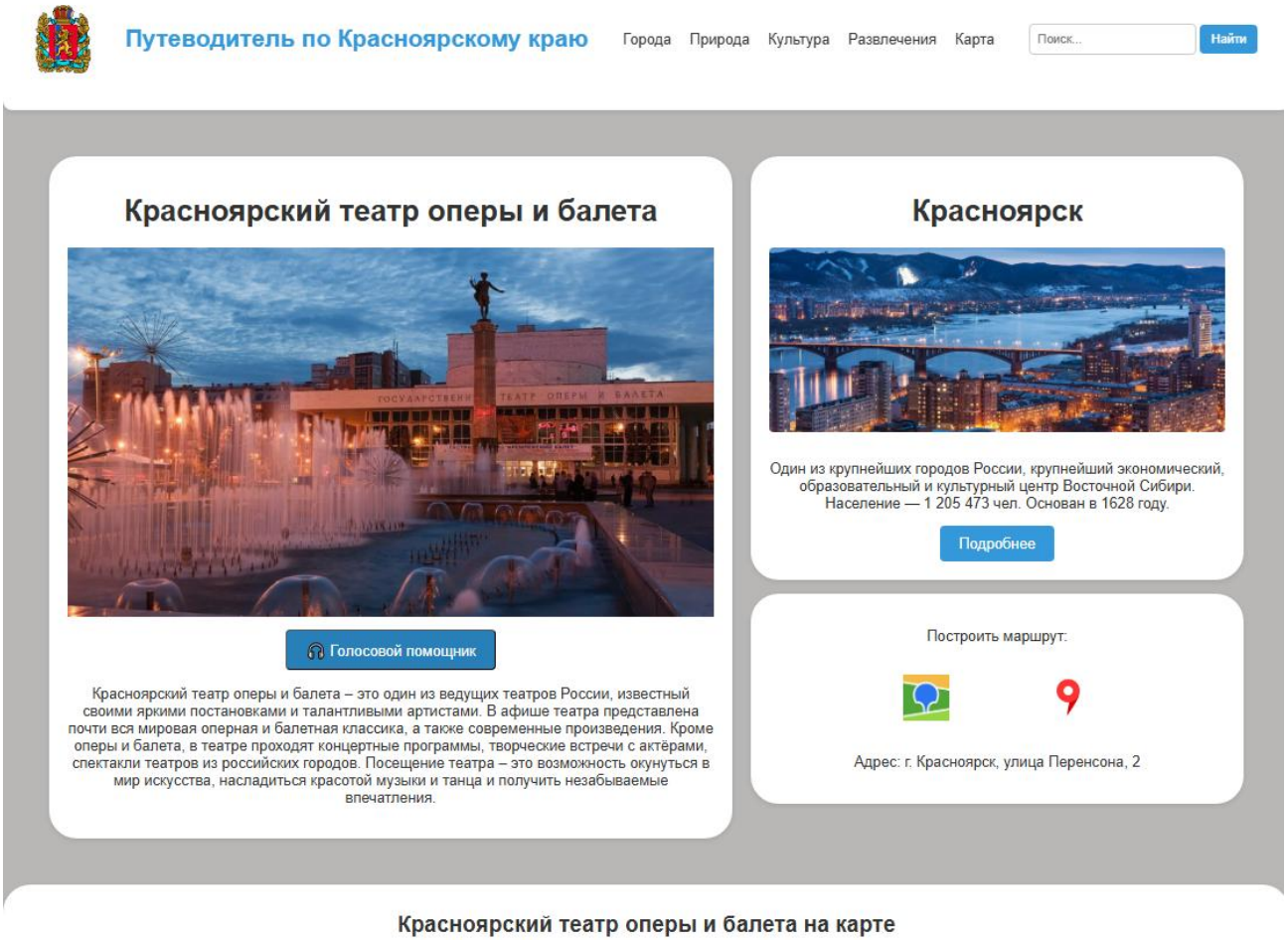


Рисунок 27 – Страница «Красноярский театр оперы и балеты»

Функция построения маршрута реализована через переход по ссылкам на 2GIS и Яндекс.Карты с указанием координат объекта. Рисунок 28 демонстрирует программную реализацию данной функции, а рисунок 29 – результат отображения маршрута при активации ссылки:

```
<div class="content-enter">
    <p>Построить маршрут:</p>
    <p> <a
            href="https://2gis.ru/krasnoyarsk/directions/points/%7C92.868542%2C56.008645%3B9856906994660 37?m=92.877234%2C56.008144%2F14.3"><img
                src="../img/index/2gis.webp" width="50" height="50"></a>
        <a
            href="https://yandex.ru/maps/62/krasnoyarsk/?ll=92.887110%2C56.009465&mode=routes&rtext=~56.008646%2C92.868409&rtt=auto&ruri=~ymapsbm1%3A%2
                src="../img/index/yandex.jpg" width="50" height="50"></a>
    </p>
    <p>Адрес: г. Красноярск, улица Перенсона, 2</p>
</div>
```

Рисунок 28 – Фрагмент кода «Построить маршрут»

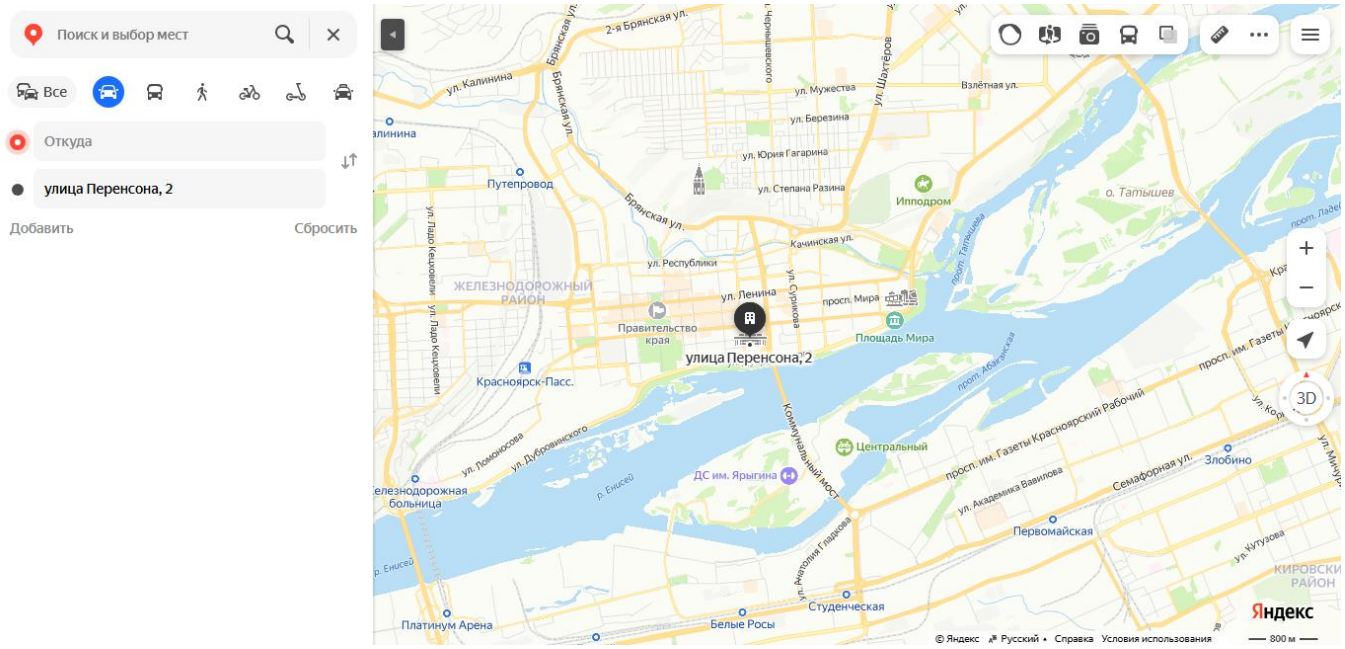


Рисунок 29 – Карта с возможностью построения маршрута

Файл style.css содержит более 300 строк кода и определяет внешний вид всех элементов web-приложения. Ниже рассмотрим ключевые блоки:

1. Сетка карточек объектов (.featured-grid) формирует три колонки, между которыми задано расстояние 20px. Фрагмент кода реализации представлен на рисунке 30:

```
.featured-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
}
```

Рисунок 30 – Фрагмент кода .featured-grid

2. Карточки (.featured-item) получают светлый фон, тень, закругления и отступы, что визуально отделяет каждый объект и улучшает восприятие. Фрагмент кода реализации представлен на рисунке 31:

```css
.featured-item {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    text-align: center;
}
```

Рисунок 31 – Фрагмент кода .featured-item

3. Кнопки стилизованы под современный интерфейс: синие, с белым текстом и закруглёнными краями. При наведении они становятся чуть темнее. Фрагмент кода реализации представлен на рисунке 32:

```css
.button {
    background-color: #3498db;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    display: inline-block;
}

.button:hover {
    background-color: #2980b9;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    display: inline-block;
}
```

Рисунок 32 – Фрагмент кода .button и .button:hover

На рисунке 33 представлены результаты отображения фрагментов кода:

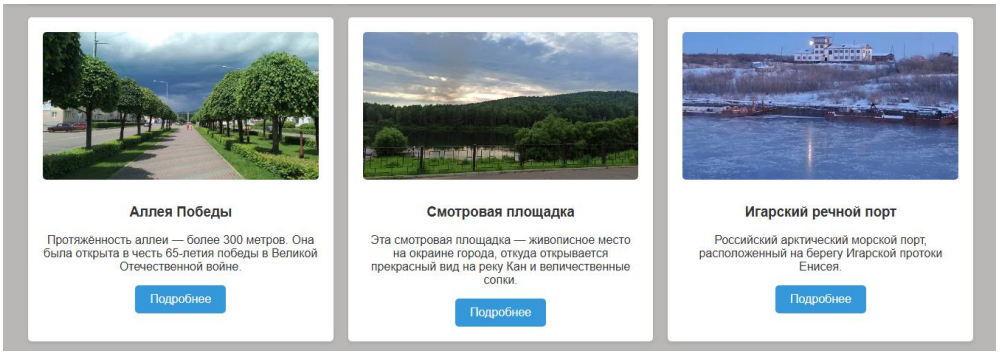


Рисунок 33 – Карточки с элементами

4. Шапка web-приложения – светлый фон, синие ссылки, position: sticky делает шапку фиксированной при прокрутке, а box-shadow визуально отделяет её от основного контента. Фрагмент кода данной реализации представлен на рисунке 34:

```
header {
    position: sticky;
    top: 0;
    z-index: 10;
    background-color: #fff;
    padding: 20px 0;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    border-bottom-right-radius: 10px;
    border-bottom-left-radius: 10px;
}
```

Рисунок 34 – Фрагмент кода .header

5. Поиск и фильтрация – поисковая строка и фильтры реализованы как формы с полями ввода и кнопками, стилизованными под общий дизайн web-приложения. Это делает интерфейс единообразным и интуитивно понятным.

Таким образом, использование HTML и CSS позволяет создать адаптивный, визуально удобный и легко расширяемый пользовательский интерфейс. Структура страниц логична, единообразна и поддерживает принципы модульной архитектуры.

3.2 Реализация интерактивности с помощью JavaScript

Для повышения интерактивности и удобства взаимодействия с web-приложением, в рамках исследования реализованы ключевые функции на языке JavaScript. Он позволяет реализовать клиентскую функциональность, необходимую для динамического взаимодействия с интерфейсом, отправки запросов и интеграции внешних сервисов.

В разработке web-приложения «Путеводитель по Красноярскому краю» активно используются три JavaScript-модуля:

1. Модуль фильтрации (filter.js) – этот скрипт позволяет пользователю отфильтровать карточки объектов по городам. Он реализует: обработку чекбоксов

(отдельных и «Выбрать все»), фильтрацию элементов .featured-item по значению атрибута data-city, скрытие нерелевантных карточек при помощи style.display. Ключевая логика реализована через прослушивание событий click и change и работу с DOM. Также предусмотрена обработка случая, если карточке не указан город.

2. Модуль поиска (search.js) – позволяет отобразить карточки, соответствующие ключевому слову, введённому пользователем. Поиск производится по заголовкам и описаниям карточек. Алгоритм поиска включает следующие этапы: получение поискового запроса из строки, приведение текста и поискового слова к нижнему регистру, поиск в содержимом карточек и отображение только тех, где найдено совпадение. Результаты поиска отображаются на отдельной странице search.html.

3. Модуль голосового помощника (voice1.js) – одна из ключевых функций web-приложения, его реализация построена на взаимодействии с локальным сервером Node.js и удалённым API Yandex SpeechKit (TTS). Скрипт получает текст из элемента с id="textToSpeak", отправляет POST-запрос с параметрами (язык, голос, текст) получает mp3-файл с озвучкой и воспроизводит его в браузере управляет состояниями воспроизведения: запуск, пауза, продолжение, завершение.

Эти модули значительно повышают удобство использования web-приложения, его интерактивность и доступность, особенно для пользователей с ограничениями по зрению или людей, находящихся в дороге. Применение JavaScript позволило создать автономное web-приложение, без использования сторонних библиотек и фреймворков.

3.3 Работа с Node.js сервером: обработка запросов от клиента и взаимодействие с API

Для обеспечения голосового функционала и защиты конфиденциальных ключей используется собственный сервер, реализованный на платформе Node.js.

Это позволяет организовать промежуточный уровень между браузером и внешними API.

Сервер выполняет две ключевые задачи:

1. Обрабатывает POST-запрос от клиента, содержащий текст для озвучивания.

2. Осуществляет запрос к API синтеза речи Yandex SpeechKit, получает аудиофайл и пересылает его обратно в браузер.

Клиент отправляет текст с помощью fetch и URLSearchParams [7]. Ниже на рисунке 35 приведён фрагмент кода серверного обработчика:

```
try {
    const response = await fetch("https://tts.api.cloud.yandex.net/speech/v1/tts:synthesize", {
        method: "POST",
        headers: {
            "Authorization": API_KEY,
            "Content-Type": "application/x-www-form-urlencoded"
        },
        body: params.toString()
    });

    if (!response.ok) {
        const errorText = await response.text();
        console.error("Ошибка от Яндекса:", errorText);
        return res.status(response.status).send(errorText);
    }

    res.set({
        'Content-Type': 'audio/mpeg',
        'Transfer-Encoding': 'chunked'
    });

    response.body.pipe(res);
} catch (error) {
    console.error("Ошибка при синтезе речи:", error);
    res.status(500).send("Ошибка при синтезе речи");
}
```

Рисунок 35 – Фрагмент кода серверного обработчика

Преимущества такой реализации заключаются в следующем:

– Безопасность – ключ API хранится на сервере, недоступен клиенту;

– Потоковая передача – pipe обеспечивает немедленную отправку аудиофайла без временного хранения;

– Масштабируемость – возможность расширение под другие API (например, для перевода, анализа текста и др.).

Подключение к серверу происходит через команду node server.js, которая представлена на рисунке 36:

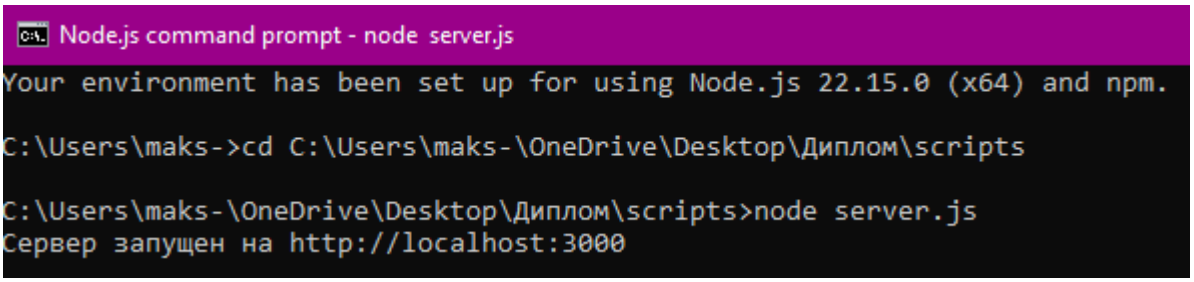


Рисунок 36 – Команды в Node.js

Node.js является идеальным выбором для серверной части проекта, позволив реализовать: безопасную обработку POST-запросов, подключение к Yandex SpeechKit без необходимости дополнительной авторизации клиента, простую архитектуру с быстрым откликом.

Серверная логика является связующим звеном между интерфейсом пользователя и внешними сервисами, обеспечивая удобство, безопасность и гибкость web-приложения

3.4 Тестирование функций web-приложения

После завершения основной разработки web-приложения необходимо провести тестирование. Тестирование позволяет убедиться, что web-приложение стабильно работает в различных условиях и соответствует требованиям, которые описаны в параграфе 2.2.

В процессе разработки использовалось ручное функциональное тестирование. Ручное тестирование позволило отследить поведение каждого элемента на разных страницах, тогда как оценка удобства проводилась с точки зрения логики переходов и визуальной читаемости интерфейса. Такой подход особенно актуален при разработке пользовательских интерфейсов для широкого круга пользователей без опыта работы с цифровыми сервисами.

Цели тестирования: проверить корректность отображения страниц и элементов интерфейса, оценить работоспособность поисковой и фильтрационной

систем, убедиться в стабильной работе голосового помощника, проверить корректную загрузку карт и меток.

Web-приложение было запущено локально. Серверная часть, реализованная с использованием Node.js, обеспечивала передачу данных для работы голосового помощника. Тестирование проводилось на всех ключевых страницах – главной, страницах объектов, категорий и городов. Основные сценарии тестирования представлены в таблице 4.

Таблица 4 – Тестирование ключевых функций

Функция	**Проверяемый аспект**	**Ожидаемый результат**
Поиск по ключевым словам	Ввод текста и отображение в search.html	Отображаются соответствующие карточки
Фильтрация по городам	Выбор чекбоксов и нажатие кнопки «Показать»	Отображаются карточки только выбранных чекбоксов
Голосовой помощник	Нажатие кнопки, воспроизведение речи	Корректная озвучка текста, без ошибок
Подключение карты	Загрузка встроенного скрипта карт на страницах объектов	Карта отображается, метки на месте
Кнопки «Подробнее»	Переход по ссылке из карточки объекта	Открывается соответствующая страница
Кнопки «Предыдущий/Следующий»	Навигация между городами	Переход на соответствующую страницу города
Маршруты в 2ГИС и Яндекс.Картах	Переход по внешним ссылкам	Открывается правильная карта в новом окне

На рисунках 37, 38, 39 изображены скриншоты, которые подтверждают работоспособность некоторых элементов тестирования:

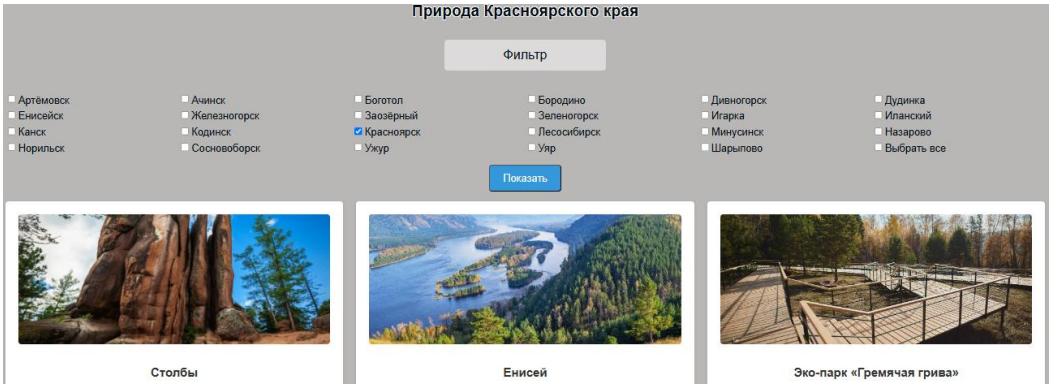


Рисунок 37 – Тестирование фильтрации объектов

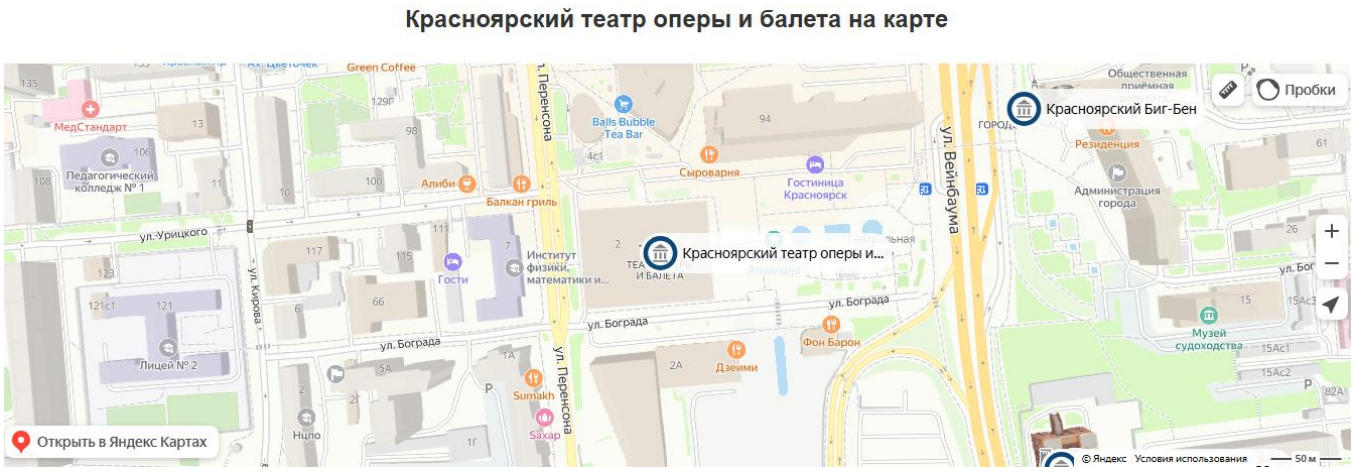


Рисунок 38 – Тестирование загрузки карты на странице объекта

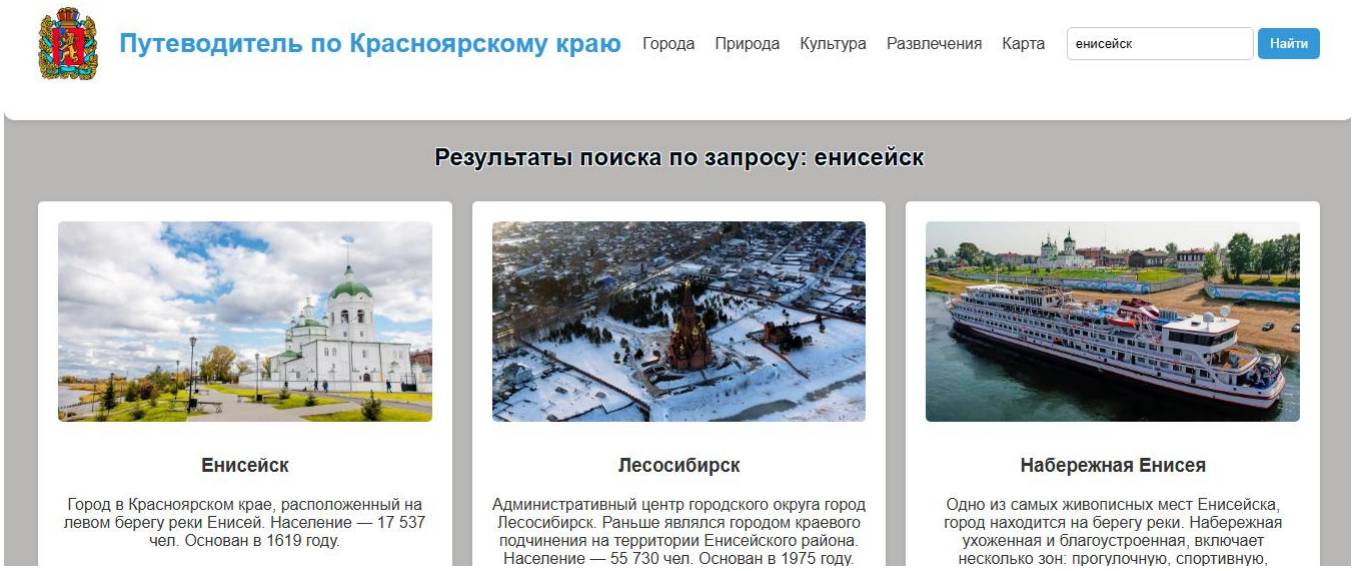


Рисунок 39 – Тестирование поиска по ключевому слову

Проведённое тестирование показало, что web-приложение стабильно функционирует в основных браузерах – Google Chrome, Mozilla Firefox и Яндекс.Браузере. Все ключевые функции, включая фильтрацию, поиск, отображение карты и работу голосового помощника, выполняются корректно. В процессе тестирования были выявлены и устранены мелкие недочеты: некорректное позиционирование кнопки голосового помощника, частичное перекрытие текста, неправильное отображение результатов поиска. После внесения правок все указанные проблемы устранены, и приложение продемонстрировало стабильную работу.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы разработано web-приложение «Путеводитель по Красноярскому краю», предназначенное для информирования пользователей о природных, культурных и туристических достопримечательностях региона. Основная задача проекта – создание, доступного и функционального web-инструмента, обеспечивающего комфортную навигацию, поиск информации, фильтрацию объектов и голосовое сопровождение – была успешно реализована. В ходе выполнения работы достигнуты следующие результаты:

– Изучены современные технологии web-разработки, включая HTML5, CSS3 и JavaScript, рассмотрены архитектурные подходы (MPA и SPA), принципы клиент-серверного взаимодействия и работа с API;

– Проанализирована целевая аудитория использования путеводителя и сформулированы функциональные требования с ориентацией на простоту интерфейса, доступность и визуальную наглядность;

– Разработана структура web-приложения, выполнено проектирование макетов страниц, сценариев взаимодействия с пользователем и клиент-серверной архитектуры;

– Реализована функциональность: фильтрация объектов по городам, поиск по ключевым словам, интеграция голосового помощника на основе TTS API (Yandex SpeechKit), подключение интерактивной карты через Яндекс.Карты;

– Проведено тестирование основных функций web-приложения, подтверждена их стабильная работа и удобство использования в различных браузерах и условиях;

– Приложение полностью реализовано с использованием собственных HTML-страниц, единого CSS-файла и модулей на JavaScript, без применения внешних фреймворков, что делает его лёгким и автономным.

Разработанное web-приложение может использоваться учителями при подготовке уроков краеведения, туристическими агентствами, а также

индивидуальными путешественниками, планирующими маршруты по Красноярскому краю.

Таким образом, в рамках исследования решены все поставленные задачи: от проектирования интерфейса и архитектуры до реализации и тестирования функционального web-приложения. Полученное решение отвечает современным требованиям, легко расширяется и может служить основой для дальнейших усовершенствований и внедрения в практическую сферу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аквино, К. Front-end. Клиентская разработка для профессионалов / К. Аквино. – СПБ: Питер, 2017. – 512 с.

2. Андерсон, С. Приманка для пользователей. Создаем привлекательный сайт / С. Андерсон. – М. : Питер, 2021. – 793 с.

3. Архитектура фронтенда и какой она должна быть / Habr: [сайт]. – URL: https://habr.com/ru/articles/667214/ (дата обращения: 21.05.2025).

4. Блаха, М. UML 2.0. Объектно-ориентированное моделирование и разработка / М. Блаха, Д. Рамбо – СПб.: Питер, 2021. – 545 с.

5. Браун, Э. Изучаем JavaScript. Руководство по созданию современных веб-сайтов / Э. Браун. – Москва: Альфа-книга, 2017. – 368 с.

6. Васильков, А. В. Информационные системы и их безопасность: Учебное пособие / А. В. Васильков, А. А. Васильков. – М. : Форум, 2017. – 528 с.

7. Введение в fetch / Habr: [сайт]. – URL: https://habr.com/ru/articles/252941/ (дата обращения: 21.05.2025).

8. Веб-технологии для разработчиков – CSS / MDN Web Docs: [сайт]. – URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата обращения: 21.05.2025).

9. Веб-технологии для разработчиков – JavaScript / MDN Web Docs: [сайт]. – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 21.05.2025).

10. Веб-технологии для разработчиков – HTML / MDN Web Docs: [сайт]. – URL: https://developer.mozilla.org/ru/docs/Web/HTML (дата обращения: 21.05.2025).

11. Гвоздева, В. А. Информатика, автоматизированные информационные технологии и системы / В. А. Гвоздева. – М. : Форум, 2018. – 58 с.

12. Главная страница технологии Yandex SpeechKit / YandexCloud: [сайт]. – URL: https://yandex.cloud/ru/services/speechkit?utm_referrer=about%3Ablank (дата обращения: 21.05.2025).

13. Гришин, В. Н. Информационные технологии в профессиональной деятельности: учебник / В. Н. Гришин, Е. Е. Панфилова. – М. : ИНФРА-М, 2015. – 416 с.

14. Дакетт, Д. Основы веб-программирования с использованием HTML / Джон Дакетт. – Москва: Эксмо, 2020. – 239 с.

15. Дакетт, Д. Разработка и дизайн веб-сайтов / Джон Дакетт. – Москва: Эксмо, 2018. – 250 с.

16. Диаграмма вариантов использования (UseCase diagram) / Ensi: [сайт]. – URL: https://docs.ensi.tech/analyst-guides/tools/diagrams/uml/use-case-diagram (дата обращения: 21.05.2025).

17. Документация для разработчиков / API Яндекс.Карты: [сайт]. – URL: https://yandex.ru/maps-api/docs/ (дата обращения: 21.05.2025).

18. Заботина, Н. Н. Проектирование информационных систем: учеб пособие / Н. Н. Заботина. – М. : ИНФРА-М, 2011. – 331 с.

19. Как выбрать подходящую архитектуру внешнего интерфейса? / Tonai Blog: [сайт]. – URL: https://tonai.github.io/blog/posts/front-end-architecture/ (дата обращения: 21.05.2025).

20. Кириченко, А. В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков / А. В. Кириченко, А. П. Никольский, Е. В. Дубовик. – Санкт-Петербург: Наука и техника, 2021. – 432 с. – ISBN 978- 5-94387-271-6.

21. Кислицын, Е. В. Современные технологии разработки программного обеспечения: учеб. пособие / Е. В. Кислицын, М. А. Панов. – Екатеринбург: Изд-во УрГЭУ, 2021. – 176 с.

22. Климов, А. JavaScript на примерах / А. Климов. – СПБ: БХВПетербург: Эксмо, 2018. – 336 с.

23. Коэн, Исси. Полный справочник по HTML, CSS и JavaScript / Исси Коэн. – Паблишерз: Эксмо, 2017. – 246 с.

24. Леоненков, А. В. Самоучитель UML 2. / А. В. Леоненков. – СПб: БХВ-Петербург, 2007. – 568 с.

25. Лоре, А. Проектирование веб-API / А. Лоре. – Москва: ДМК-Пресс, 2020. – 440 с.

26. О технологии «Синтез речи» / YandexCloud: [сайт]. – URL: https://yandex.cloud/ru/docs/speechkit/tts/ (дата обращения: 21.05.2025).

27. Паршин, К. А. Методы и средства проектирования информационных систем и технологий: учеб.-метод. пособие / К. А. Паршин. – Екатеринбург: УрГУПС, 2018. – 129 с.

28. Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера, 5 изд / Н.А. Прохоренок. – СПб. : БХВ-Петербург, 2019. – 912 с.

29. Следующий этап развития Веба / Habr: [сайт]. – URL: https://habr.com/ru/companies/timeweb/articles/695798/ (дата обращения: 21.05.2025).

30. Фаулер, М. UML Основы: Краткое руководство по стандартному языку объектного моделирования / М. Фаулер. – СПб: Символ-Плюс, 2004. – 192 с.

31. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган. – СПб: Символ-Плюс, 2008. – 992 с.

32. Фрейн, Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – Санкт-Петербург: Питер, 2018. – 304 с.

33. Фримен, Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон. – Москва: Эксмо, 2007. – 2010 с.

34. Хассан, Г. UML. Проектирование систем реального времени, распределённых и параллельных приложений. / Г. Хассан. – ДМК Пресс: БХВ-Петербург, 2022. – 702 с.

35. Хэррон, Д. Node.js Разработка серверных веб-приложений на JavaScript / Д. Хэррон. – Москва: ДМК, 2014. – 144 с.

36. Эспозито, Д. Разработка современных веб-приложений. Анализ предметных областей и технологий / Д. Эспозито – Москва: Вильямс, 2017. – 464 с.

37. HTML5BOOK: официальный сайт. – URL: https://html5book.ru (дата обращения 21.05.2025).

38. HTML Tutorial/W3Schools: [сайт]. – URL: https://www.w3schools.com/html/ (дата обращения: 21.05.2025).

39. JavaScript Tutorial /W3Schools: [сайт].– URL: https://www.w3schools.com/js (дата обращения 21.05.2025).

40. Node.js: официальный сайт. – URL: https://nodejs.org/en/ (дата обращения: 21.05.2025).

ПРИЛОЖЕНИЕ А

Листинг кода JavaScript

web-приложения «Путеводитель по Красноярскому краю»

1. Скрипт фильтрации – filter.js

```
document.addEventListener('DOMContentLoaded', function () {
    const filterButton = document.getElementById('filterButton');
    const checkboxList = document.getElementById('checkboxList');
    const featuredGrid = document.getElementById('featuredGrid');
    const featuredItems = featuredGrid.querySelectorAll('.featured-item');
    const selectAllCheckbox = document.getElementById('selectAll');

    // Функция для фильтрации
    function filterItems() {
        const checkedCities =
Array.from(checkboxList.querySelectorAll('input[type="checkbox"]:checked'))
            .filter(checkbox => checkbox.id !== 'selectAll') // Исключаем "Выбрать все"
            .map(checkbox => checkbox.value);

        featuredItems.forEach(item => {
            const city = item.dataset.city;
            if (checkedCities.length === 0 || checkedCities.includes(city)) {
                item.style.display = '';
            } else {
                item.style.display = 'none';
            }
        });
    }

    // Обработчик для кнопки «Показать»
    filterButton.addEventListener('click', function (event) {
        event.preventDefault();
        filterItems();
```

```
    });

    // Обработчик для «Выбрать все»
    selectAllCheckbox.addEventListener('change', function () {
        const checkboxes = checkboxList.querySelectorAll('input[type="checkbox"]');
        checkboxes.forEach(checkbox => {
            checkbox.checked = selectAllCheckbox.checked;
        });
    });

    const featuredItemsArray = Array.from(document.querySelectorAll('.featured-item'));
    featuredItemsArray.forEach(item => {
        if (!item.dataset.city) {
            console.warn("Элементу .featured-item не хватает data-city:", item);
        }
    });
});
```

2. Скрипт поиска – search.js

```
document.addEventListener('DOMContentLoaded', function () {
    const searchInput = document.getElementById('searchInput');  // Получаем поле поиска
    const searchQueryDisplay = document.getElementById('searchQuery'); // Получаем элемент для отображения запроса
    const featuredGrid = document.getElementById('featuredGrid');
    const featuredItems = featuredGrid.querySelectorAll('.featured-item');

    // Функция для получения параметра из URL
    function getParameterByName(name, url = window.location.href) {
        name = name.replace(/[\[\]]/g, '\\$&');
        var regex = new RegExp('[?&]' + name + '(=([^&#]*)|&|#|$)'),
            results = regex.exec(url);
```

```
    if (!results) return null;
    if (!results[2]) return '';
    return decodeURIComponent(results[].replace(/\+/g, ' '));
}

// Получаем поисковой запрос из URL
const query = getParameterByName('q');

// Если поисковой запрос есть, отображаем его и выполняем поиск
if (query) {
    searchInput.value = query; // Заполняем поле поиска
    searchQueryDisplay.textContent = query; // Отображаем запрос
    filterItems(query); // Выполняем поиск
}

function filterItems(query) {
    query = query.toLowerCase(); // Приводим запрос к нижнему регистру

    featuredItems.forEach(item => {
        const title = item.querySelector('h3').textContent.toLowerCase(); // Приводим заголовок к нижнему регистру
        const description = item.querySelector('p').textContent.toLowerCase(); // Приводим описание к нижнему регистру

        if (title.includes(query) || description.includes(query)) {
            item.style.display = '';
        } else {
            item.style.display = 'none';
        };

        searchButton.addEventListener('click', function (event) {
            event.preventDefault(); // Предотвращаем отправку формы
```

```
            const query = searchInput.value;
            searchQueryDisplay.textContent = query; //  Показываем запрос
            filterItems(query);
        });

        searchInput.addEventListener('input', function () {
            const query = searchInput.value;
            searchQueryDisplay.textContent = query;
            filterItems(query);
        });
    });
  }
})
```

3. Скрипт голосового помощника – voice.js

```
const textToSpeak = document.getElementById("textToSpeak");
const speechBtn = document.getElementById("speechBtn");

let audio = null;
let isSpeaking = false;
let isPaused = false;

async function textToSpeechYandex(text) {
  const url = "http://localhost:3000/tts"; // Запрос к локальному серверу

  const params = new URLSearchParams({
    text: text,
    lang: "ru-RU",
    voice: "oksana", // другие: zahar, ermil, jane и др.
    format: "mp3"
  });
```

```
const response = await fetch(url, {
    method: "POST",
    headers: {
        "Content-Type": "application/x-www-form-urlencoded"
    },
    body: params
});

if (!response.ok) {
    const errorText = await response.text();
    console.error("Ошибка от сервера:", errorText);
    speechBtn.innerHTML = "&#127911 Ошибка";
    return;
}

const audioBlob = await response.blob();
const audioUrl = URL.createObjectURL(audioBlob);
audio = new Audio(audioUrl);

// События
audio.onplay = () => {
    isSpeaking = true;
    isPaused = false;
    speechBtn.innerHTML = "&#9208 Остановить";
};

audio.onpause = () => {
    isPaused = true;
    isSpeaking = false;
    speechBtn.innerHTML = "&#9654 Продолжить";
};
```

```
    audio.onended = () => {
        isSpeaking = false;
        isPaused = false;
        speechBtn.innerHTML = "&#127911 Голосовой помощник";
    };

    audio.play();
}

speechBtn.addEventListener("click", async (e) => {
    e.preventDefault();

    const text = textToSpeak.textContent.trim();
    if (!text) return;

    if (!audio) {
        await textToSpeechYandex(text);
    } else {
        if (isSpeaking) {
            audio.pause();
        } else if (isPaused) {
            audio.play();
        } else {
            await textToSpeechYandex(text); // проигрываем заново, если всё завершено
        }
    }
});

window.addEventListener("load", () => {
    if (audio) {
        audio.pause();
```

```
        audio = null;
    }
    isSpeaking = false;
    isPaused = false;
    speechBtn.innerHTML = "&#127911 Голосовой помощник";
});
```

4. Скрипт сервера – server.js

```
import express from 'express';
import cors from 'cors';
import fetch from 'node-fetch';
import bodyParser from 'body-parser';

const app = express();
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));

const API_KEY = "Api-Key AQVNwtmQXyD8q8BjCwxIro9veIae3vCdolwm9-6f";

app.post("/tts", async (req, res) => {
    const text = req.body.text;
    console.log("Получен текст:", text);

    const params = new URLSearchParams();
    params.append("text", text);
    params.append("lang", "ru-RU");
    params.append("voice", "oksana");
    params.append("speed", "1.0");
    params.append("format", "mp3");

    try {
```

```
        const response = await fetch("https://tts.api.cloud.yandex.net/speech/v1/tts:synthesize", {
            method: "POST",
            headers: {
                "Authorization": API_KEY,
                "Content-Type": "application/x-www-form-urlencoded"
            },
            body: params.toString()
        });

        if (!response.ok) {
            const errorText = await response.text();
            console.error("Ошибка от Яндекса:", errorText);
            return res.status(response.status).send(errorText);
        }

        res.set({
            'Content-Type': 'audio/mpeg',
            'Transfer-Encoding': 'chunked'
        });
        response.body.pipe(res);
    } catch (error) {
        console.error("Ошибка при синтезе речи:", error);
        res.status(500).send("Ошибка при синтезе речи");
    }
});
const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Сервер запущен на http://localhost:${PORT}`);
});
```

ПРИЛОЖЕНИЕ Б

Листинг кода HTML и CSS web-приложения

1. Фрагмент листинга кода главной страницы – index.html

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Путеводитель по Красноярскому краю</title>
    <link rel="stylesheet" href="style.css">
    <link rel="shortcut icon" href="img/index/logo.png">
</head>
<body>
    <header>
        <div class="container">
            <img src="img/index/logo.png" alt="Путеводитель по Красноярскому краю">
            <div class="logo">
                <a href="index.html">Путеводитель по Красноярскому краю</a>
            </div>
            <nav>
                <ul>
                    <li><a href="cities.html">Города</a></li>
                    <li><a href="nature.html">Природа</a></li>
                    <li><a href="culture.html">Культура</a></li>
                    <li><a href="fun.html">Развлечения</a></li>
                    <li><a href="map.html">Карта</a></li>
                </ul>
            </nav>
            <div class="search">
                <form action="search.html" method="GET">
                    <input type="text" placeholder="Поиск..." name="q">
                    <button type="submit">Найти</button>
```

```
            </form>
        </div>
    </div>
</header>
<main>
    <section class="welcome">
        <div class="container">
            <h1>Добро пожаловать в Красноярский край!</h1>
            <p>Откройте для себя уникальную природу, богатую культуру и захватывающие приключения</p>
            <a href="index0.html" class="button">Узнать больше</a>
        </div>
    </section>
    <section class="featured">
        <div class="container">
            <h2>Популярные места</h2>
            <div class="featured-grid">
                <div class="featured-item">
                    <img src="img/nature/stolb.jpg" alt="Столбы">
                    <h3>Столбы</h3>
                    <p>Красноярские Столбы – уникальное явление. О Столбах написано множество книг и статей, снято
                        множество фильмов. Им посвящены стихи, о них написаны песни, они служат источником
                        вдохновения художников.</p>
                    <a href="nature/stolb.html" class="button">Подробнее</a>
                </div>
                <div class="featured-item">
                    <img src="img/cities/yeniseysk.jpg" alt="Енисейск">
                    <h3>Енисейск</h3>
                    <p>Город в Красноярском крае, расположенный на левом берегу реки Енисей. Население – 17 537 чел.
                        Основан в 1619 году.</p>
```

```
                <a href="cities/yeniseysk.html" class="button">Подробнее</a>
            </div>
            <div class="featured-item">
                <img src="img/culture/theatre-square.jpg" alt="Красноярский театр оперы и балета">
                <h3>Красноярский театр оперы и балета</h3>
                <p>В афише театра представлена почти вся мировая оперная и балетная классика. Кроме оперы и
                    балета, в театре проходят концертные программы, творческие встречи с актёрами, спектакли
                    театров из российских городов.</p>
                <a href="culture/theatre-square.html" class="button">Подробнее</a>
            </div>
        </div>
    </div>
  </section>
</main>

<footer>
  <div class="container">
    <p>&copy; 2025 Путеводитель по Красноярскому краю</p>
    <nav>
      <ul>
        <li><a href="about.html">О сайте</a></li>
        <li><a href="contacts.html">Контакты</a></li>
      </ul>
    </nav>
  </div>
</footer>
```

2. Фрагмент листинга кода страницы с объектами – cities.html

```
<main>
  <section class="featured">
```

```
        <div class="container">
            <h2>Города Красноярского края</h2>
            <div class="featured-grid">
                <div class="featured-item">
                    <img src="img/cities/artemovsk.webp" alt="Артёмовск">
                    <h3>Артёмовск</h3>
                    <p>Город в Курагинском районе Красноярского края. Население – 1492 чел. Основан в 1700 году.</p>
                    <a href="cities/artemovsk.html" class="button">Подробнее</a>
                </div>
                  //Другие города
                <div class="featured-item">
                    <img src="img/cities/sharipovo.webp" alt="Шарыпово">
                    <h3>Шарыпово</h3>
                    <p>Город краевого подчинения в Красноярском крае России, административный центр Шарыповского
                        района и городского округа город Шарыпово. Население – 33 961 чел. Основан в 1806 году.</p>
                    <a href="cities/sharipovo.html" class="button">Подробнее</a>
                </div>
            </div>
        </div>
        </div>
    </section>
</main>
```

3. Фрагмент листинга кода страницы объекта – theatre-square.html

```
<main>
    <div class="cont">
        <div class="content-enter">
            <h1>Красноярский театр оперы и балета</h1>
            <img class="first-image" src="../img/culture/theatre-square.jpg"
```

```
            alt="Красноярский театр оперы и балета">
        <button id="speechBtn" type="button">&#127911 Голосовой помощник</button>
        <p id="textToSpeak">Красноярский театр оперы и балета – это один из ведущих театров России, известный
            своими яркими постановками и талантливыми артистами. В афише театра представлена почти вся мировая
            оперная и балетная классика, а также современные произведения. Кроме оперы и балета, в театре
            проходят концертные программы, творческие встречи с актёрами, спектакли театров из российских
            городов. Посещение театра – это возможность окунуться в мир искусства, насладиться красотой музыки и
            танца и получить незабываемые впечатления.</p>
    </div>
    <div>
        <div class="content-enter">
            <div class="city">
                <h1>Красноярск</h1>
                <img src="../img/index/welcome.avif" alt="Красноярск">
                <p>Один из крупнейших городов России, крупнейший экономический, образовательный и культурный
                    центр Восточной Сибири. Население – 1 205 473 чел. Основан в 1628 году.</p>
                <a href="../cities/krasnoyarsk.html" class="button">Подробнее</a>
            </div>
        </div>
        <div class="content-enter">
            <p>Построить маршрут:</p>
            <p> <a
href="https://2gis.ru/krasnoyarsk/directions/points/%7C92.868542%2C56.008645%3B9856906994660
37?m=92.877234%2C56.008144%2F14.3"><img
                    src="../img/index/2gis.webp" width="50" height="50"></a>
                <a
href="https://yandex.ru/maps/62/krasnoyarsk/?ll=92.887110%2C56.009465&mode=routes&rtext=~56.0
```

```
08646%2C92.868409&rtt=auto&ruri=~ymapsbm1%3A%2F%2Fgeo%3Fdata%3DCgg1NjQyNzUwNhJE0KDQvtGB0YHQuNGPLCDQmtGA0LDRgdC90L7Rj9GA0YHQuiwg0YPQu9C40YbQsCDQn9C10YDQtdC90YHQvtC90LAsIDIiCg2gvLlCFdoIYEI%2C&z=13.03"><img
                        src="../img/index/yandex.jpg" width="50" height="50"></a>
                </p>
                <p>Адрес: г. Красноярск, улица Перенсона, 2</p>
            </div>
        </div>
    </div>
    <div class="content">
        <h2>Красноярский театр оперы и балета на карте</h2>
        <script type="text/javascript" charset="utf-8" async
            src="https://api-
maps.yandex.ru/services/constructor/1.0/js/?um=constructor%3A0447f36a3e240a5d91fc4b811f21146918169a4aaa627bac5c76363a4227b64f&amp;width=100%25&amp;height=400&amp;lang=ru_RU&amp;scroll=true"></script>
    </div>
```

4. Фрагмент листинга кода оформления – style.css

```
body {
    font-family: sans-serif;
    margin: 0;
    padding: 0;
    background-color: #b9b6b6;
    color: #333;
}

.container {
    width: 95%;
    margin: 0 auto;
    padding: 20px;
}
```

```
a {
    text-decoration: none;
    color: #3498db;
}

a:hover,
summary:hover {
    color: #2980b9;
}

.button {
    background-color: #3498db;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    display: inline-block;
}

.button:hover {
    background-color: #2980b9;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    display: inline-block;
}

header {
    position: sticky;
    top: 0;
    z-index: 10;
    background-color: #fff;
    padding: 20px 0;
```

```
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    border-bottom-right-radius: 10px;
    border-bottom-left-radius: 10px;
}

header img {
    width: 60px;
    margin-right: 20px;
}

header .container {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

header .logo a {
    font-size: 26px;
    font-weight: bold;
}

header nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex;
}

header nav ul li {
    margin-left: 20px;
}
```

```
header nav ul li a {
    color: #333;
}

header .search input[type="text"] {
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
    margin-left: 20px;
}

header .search button {
    background-color: #3498db;
    color: #fff;
    border: none;
    padding: 8px 12px;
    border-radius: 5px;
    cursor: pointer;
}

.welcome {
    background-image: url('img/index/welcome.avif');
    background-size: cover;
    background-position: center;
    text-align: center;
    padding: 100px 0;
}

.welcome h1 {
    font-size: 48px;
    margin-bottom: 20px;
    color: #010c13;
```

```
    text-shadow:
        -1px -1px 0 #fff,
        1px -1px 0 #fff,
        -1px 1px 0 #fff,
        1px 1px 0 #fff;
}

.welcome p {
    font-size: 18px;
    margin-bottom: 30px;
    color: #010c13;
    text-shadow:
        -1px -1px 0 #fff,
        1px -1px 0 #fff,
        -1px 1px 0 #fff,
        1px 1px 0 #fff;
}

.featured h2 {
    text-align: center;
    margin-bottom: 30px;
    font-size: 24px;
    color: #010c13;
    text-shadow:
        -0.7px -0.7px 0 #fff,
        0.7px -0.7px 0 #fff,
        -0.7px 0.7px 0 #fff,
        0.7px 0.7px 0 #fff;
}

.featured-grid {
    display: grid;
```

```
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
}

.featured-item {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    text-align: center;
}

.featured-item img {
    width: 100%;
    height: 200px;
    object-fit: cover;
    border-radius: 5px;
    margin-bottom: 10px;
}

footer {
    background-color: #333;
    color: #fff;
    padding: 20px 0;
    text-align: center;
}

.footer {
    position: fixed;
    bottom: 0;
    left: 0;
    right: 0;
```

```
}

footer .container {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

footer nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex;
}

footer nav ul li {
    margin-left: 20px;
}

footer nav ul li a {
    color: #fff;
}
```