Разработка приложения на react по «Экскурсиям по Дагестану».

Содержание

Введение

Глава 1. Обзор технологий для разработки веб-приложений

1.1 Современные технологии веб-разработки

1.2 Основы работы с React и его преимущества

1.3 MobX: управление состоянием в приложении

1.4 Анализ существующих решений для туристических сайтов

Глава 2. Проектирование и разработка веб-приложения для отрасли туризма.

2.0 Характеристика объекта автоматизации, выбор автоматизируемых бизнес-процессов (на примере компании ООО «Крон»).

* 1. Анализ целевой аудитории и требований к сайту
  2. Изучение существующих разработок в этой сфере и проектирование основных модулей приложения
  3. Формирование базы данных, интерфейса с использованием React
  4. Управление состоянием с MobX и интеграция с внешними сервисами и API

Глава 3. Практические аспекты работы с разработанным веб-приложением.

3.1 Описание технологий работы с программным продуктом

3.2 Специфические особенности разработанного программного кода и его функционала

3.3 Перспективы расширения функционала приложения.

Заключение(выводы и предложения)

Приложение

Список литератур

**Введение**

В современном мире, где информация и услуги все более переходят в цифровую среду, потребность в онлайн-сервисах, способных быстро и удобно предоставлять данные, стремительно возрастает. Туризм и экскурсионные услуги – одна из сфер, где цифровизация позволяет значительно упростить и ускорить взаимодействие с клиентами. Пользователи, находясь в разных уголках мира, могут заранее планировать и бронировать экскурсии, знакомиться с историей и культурой региона, а также получать полную информацию о местных достопримечательностях. В этом контексте разработка веб-приложений, предоставляющих подобные услуги, является крайне востребованной.

Дагестан – это уникальный и многообразный регион России, обладающий богатым культурным и природным наследием. Его разнообразные ландшафты, культурные традиции и исторические достопримечательности привлекают множество туристов из разных стран. Однако из-за ограниченной доступности информационных ресурсов и сложности планирования путешествий, многие туристы сталкиваются с трудностями при организации экскурсий по Дагестану. В то же время, развитие внутреннего туризма в России подчеркивает актуальность создания современных, интерактивных и удобных сайтов для экскурсий, которые способны улучшить пользовательский опыт и предоставить всю необходимую информацию об этом регионе.

**Актуальность темы**

Актуальность разработки сайта для экскурсий по Дагестану заключается в том, что доступ к туристическим услугам и информации в режиме онлайн способствует увеличению туристического потока в регион и развитию местной экономики. Веб-приложения, созданные с использованием современных технологий, таких как React и MobX, позволяют сделать сайт не только функциональным, но и высокопроизводительным и отзывчивым. Эти технологии обеспечивают плавность работы сайта и удобное управление состоянием, что повышает удовлетворенность пользователей.

Кроме того, уникальность Дагестана как региона с древней историей и богатой культурой требует специального подхода к разработке интерфейса. Важно создать сайт, который не только предоставляет информацию, но и вызывает у пользователя интерес к культуре и традициям Дагестана. Внедрение интерактивных карт, галерей, отзывов, а также возможности предварительного бронирования экскурсий делает сайт многофункциональным инструментом, способным привлечь большее количество потенциальных туристов.

**Цель и задачи исследования**

Целью данной работы является разработка веб-сайта для экскурсий по Дагестану на основе технологий React и MobX. Сайт должен обеспечивать пользователям удобный доступ к информации о турах, экскурсиях и местных достопримечательностях, а также предоставлять возможности для бронирования и обратной связи.

Для достижения поставленной цели были определены следующие задачи:

1. Провести анализ современных технологий веб-разработки и выбрать наиболее подходящие для создания туристического сайта.
2. Изучить потребности целевой аудитории и определить ключевые функции сайта.
3. Спроектировать архитектуру приложения и разработать пользовательский интерфейс, который будет сочетать удобство и эстетичность.
4. Реализовать функционал сайта, включая управление состоянием с помощью MobX и интеграцию с внешними API для предоставления интерактивных возможностей.
5. Провести тестирование сайта для обеспечения его стабильности, удобства использования и производительности.
6. Подготовить сайт к развертыванию и запустить его на выбранной платформе.

**Объект и предмет исследования**

Объектом исследования является процесс разработки веб-сайта для туристических услуг, ориентированного на популяризацию экскурсий по Дагестану. Предмет исследования – технологии создания интерфейсов и управления состоянием приложения, а также архитектура взаимодействия компонентов веб-сайта, основанного на React и MobX.

**Методы исследования**

В ходе выполнения данной работы использовались различные методы исследования, включая:

* Анализ существующих решений в сфере туристических веб-приложений и изучение их сильных и слабых сторон.
* Методы проектирования программного обеспечения, такие как разработка архитектуры и пользовательского интерфейса.
* Методы разработки на React и MobX для создания интерактивного и высокопроизводительного сайта.
* Тестирование и отладка кода с использованием современных инструментов, чтобы гарантировать стабильность работы приложения.

**Структура дипломной работы**

Работа состоит из трех основных глав. В первой главе рассмотрены технологии, применяемые в современной веб-разработке, а также обоснован выбор React и MobX для создания сайта. Вторая глава посвящена проектированию и реализации приложения, описаны архитектура сайта и процесс создания основных компонентов. В третьей главе проводится тестирование и подготовка сайта к запуску. В заключении подведены итоги работы, указаны основные достижения и перспективы дальнейшего развития проекта.

**Глава 1. Обзор технологий для разработки веб-приложений**

**1.1 Современные технологии веб-разработки**

Современные технологии веб-разработки прошли значительный путь развития, от создания простых статических страниц до сложных веб-приложений с интерактивными элементами, высокой производительностью и поддержкой масштабируемости. Сегодня разработчики имеют широкий выбор инструментов и подходов, которые позволяют создавать удобные, быстрые и безопасные веб-сайты. Рассмотрим ключевые направления и технологии, используемые в веб-разработке в настоящее время.

### Фронтенд-технологии

Фронтенд-технологии отвечают за отображение информации и взаимодействие пользователя с веб-приложением. Современные фронтенд-инструменты включают в себя не только язык разметки HTML и стилизацию с помощью CSS, но и сложные JavaScript-фреймворки и библиотеки, которые делают приложения более интерактивными и отзывчивыми.

JavaScript является основным языком для создания интерактивных элементов на веб-страницах. В последние годы появились многочисленные библиотеки и фреймворки, такие как **React**, **Vue.js** и **Angular**, которые упрощают процесс создания сложных пользовательских интерфейсов.

React — это библиотека, разработанная Facebook для создания пользовательских интерфейсов. React позволяет разработчикам строить интерфейсы, используя компонентный подход, что упрощает их поддержку и масштабирование. Благодаря виртуальному DOM (виртуальному дереву объектов), React обеспечивает высокую производительность и скорость работы приложений.

**Vue.js** — это прогрессивный JavaScript-фреймворк, который также поддерживает компонентный подход. Vue отличается простотой в освоении, что делает его популярным выбором среди начинающих разработчиков. Он позволяет создавать приложения с низким порогом вхождения, оставаясь при этом мощным инструментом для построения сложных интерфейсов.

**Angular** – это фреймворк от Google, который ориентирован на создание масштабируемых веб-приложений. Angular предоставляет более комплексные возможности, чем React и Vue, и включает встроенные функции для управления состоянием, работы с формами и маршрутизации.

**TypeScript является** надстройкой над JavaScript, которая добавляет статическую типизацию. TypeScript помогает избежать ошибок и улучшает читаемость кода, что особенно полезно в крупных проектах. TypeScript также тесно интегрирован с фреймворком Angular и поддерживается в React и Vue.

### Бэкенд-технологии

Бэкенд-технологии отвечают за серверную логику, обработку данных и взаимодействие с базой данных. Современные бэкенд-решения позволяют разрабатывать масштабируемые и надежные серверные приложения.

Node.js — это среда выполнения JavaScript, которая позволяет запускать JavaScript-код на сервере. Благодаря высокой производительности и поддержке асинхронного программирования, Node.js становится всё более популярным среди разработчиков, особенно для создания приложений в реальном времени.

**Express.js** – это фреймворк для создания приложений на Node.js. Express.js предоставляет набор инструментов для построения серверной части приложений и широко используется для разработки RESTful API.

Django и Flask — это популярные фреймворки для разработки на языке Python. Django предоставляет структурированную среду и включает множество встроенных функций, что ускоряет процесс разработки. Flask, в свою очередь, является более лёгким фреймворком и предоставляет разработчикам большую свободу в выборе инструментов.

### Управление состоянием

С увеличением сложности приложений управление состоянием стало ключевым аспектом в разработке. Современные веб-приложения часто имеют сложные структуры данных и зависят от множества пользовательских взаимодействий, поэтому управление состоянием позволяет эффективно синхронизировать данные и интерфейс.

* **Redux** – это одна из наиболее популярных библиотек для управления состоянием в React-приложениях. Redux поддерживает централизованное хранилище, что упрощает обмен данными между компонентами.
* **MobX** – это альтернатива Redux, которая предоставляет более гибкий и упрощенный способ управления состоянием. MobX позволяет легко отслеживать изменения и автоматизировать обновление интерфейса при изменении данных.
* **Context API** – встроенная в React технология для передачи данных между компонентами. Она полезна для более простых случаев и позволяет избежать использования сторонних библиотек для управления состоянием.

### Базы данных

Базы данных хранят и обрабатывают данные, используемые веб-приложениями. Современные веб-приложения используют как реляционные базы данных, так и NoSQL-хранилища в зависимости от требований проекта.

* **PostgreSQL и MySQL – э**то реляционные базы данных, которые поддерживают ACID-транзакции и обеспечивают надежное хранение структурированных данных. Они широко применяются в веб-разработке, особенно для крупных и корпоративных проектов.
* MongoDB — это NoSQL-база данных, которая работает с данными в формате JSON. Она подходит для работы с большими объемами данных и часто используется в приложениях, где требуется высокая масштабируемость и гибкость.

### Системы управления версиями и CI/CD

Современная разработка невозможна без использования систем управления версиями и автоматизированных инструментов для развертывания и тестирования.

* **Git и GitHub/GitLab** – Git является стандартом управления версиями, а GitHub и GitLab предоставляют платформы для хранения и совместной работы над кодом.
* **CI/CD-платформы** – инструменты, такие как Jenkins, GitHub Actions и GitLab CI, позволяют автоматически тестировать и развертывать приложения при каждом изменении в коде, что значительно ускоряет процесс разработки и повышает надежность программного обеспечения.

### Контейнеризация и облачные технологии

Современные веб-приложения часто развертываются в контейнерах и облачных средах, что упрощает их масштабирование и управление.

* Docker предоставляет контейнеры, которые позволяют разрабатывать, тестировать и развертывать приложения в изолированных средах, что повышает стабильность и воспроизводимость.
* **Облачные платформы (AWS, Google Cloud, Microsoft Azure) –** эти платформы предлагают инфраструктуру и сервисы для хостинга и управления веб-приложениями, а также встроенные инструменты для анализа, мониторинга и автоматического масштабирования.

**1.2 Основы работы с React**

React — это популярная JavaScript-библиотека для создания пользовательских интерфейсов, разработанная компанией Facebook. Она фокусируется на построении интерактивных и динамичных веб-приложений, делая акцент на упрощении работы с пользовательским интерфейсом за счет использования компонентного подхода. React предоставляет мощные возможности для создания масштабируемых и отзывчивых приложений, что делает его одним из самых востребованных инструментов среди веб-разработчиков.

### **React использует компонентный** подход, который представляет собой переиспользуемую и независимую часть интерфейса. Компоненты можно создавать для любых элементов интерфейса: кнопок, форм, заголовков, целых страниц и т.д. Это позволяет строить сложные интерфейсы, комбинируя простые компоненты в более крупные структуры.

### JSX (JavaScript XML), синтаксис, похожий на HTML, который позволяет создавать элементы интерфейса прямо в коде JavaScript. JSX делает код более понятным и наглядным, так как разработчики видят, как будет выглядеть интерфейс непосредственно в коде. Пример:

### const MyComponent = () => {

### return ( <div> <h1>Hello, React!</h1> </div> );

### };

### React использует виртуальное представление DOM (Virtual DOM), чтобы оптимизировать обновления на странице. Когда происходит изменение состояния или данных в приложении, React сначала обновляет Virtual DOM, а затем только изменяет реальные элементы в браузере, которые были изменены. Это значительно повышает производительность, так как позволяет избегать ненужных перерисовок.

### В React используется **управление состоянием,** каждое приложение имеет свое состояние, представляющее данные, которые могут изменяться в ходе работы приложения. Состояние компонента можно задавать с помощью хуков, таких как useState, что позволяет динамически обновлять содержимое интерфейса в ответ на действия пользователя.

### Для создания многостраничных приложений используется библиотека React Router, которая позволяет управлять переходами между страницами и отображать разные компоненты в зависимости от маршрута, по которому перешел пользователь.

### С появлением хуков (hooks) в версии React 16.8 появилась возможность управлять состоянием и жизненным циклом компонентов, не используя классовые компоненты. Хуки, такие как useState, useEffect, useContext и другие, позволяют добавлять функциональность в компоненты-функции.

### React идеально подходит для создания одностраничных приложений (SPA), где динамическое изменение содержимого выполняется без перезагрузки страницы. Примеры таких приложений включают социальные сети, личные кабинеты, панели управления и интерактивные веб-приложения.

### Приложения, требующие быстрой реакции на пользовательские действия, например, интернет-магазины, чаты, сервисы потокового видео.

С помощью React Native разработчики могут создавать мобильные приложения для iOS и Android, используя тот же синтаксис и подход, что и в React.

**Преимущества React**

React позволяет разбивать пользовательский интерфейс на небольшие, независимые компоненты. Каждый компонент отвечает за свою часть функционала, что упрощает чтение и поддержку кода. Например, компонент кнопки можно использовать во многих местах, изменяя только его свойства. Это обеспечивает высокую степень переиспользуемости кода и модульности проекта.

React использует Virtual DOM для быстрого обновления интерфейса. При изменении данных библиотека создает копию DOM в памяти, сравнивает её с текущим состоянием и вносит минимальные изменения в реальный DOM. Такой подход позволяет избегать ненужных операций, обеспечивая высокую производительность даже в сложных приложениях с большим количеством интерактивных элементов.

React подходит для различных платформ и сценариев. На его основе можно создавать как веб-приложения, так и мобильные приложения (с помощью React Native). Например, разработчики могут писать один и тот же код интерфейса, который будет работать как на сайте, так и в мобильных приложениях, что сокращает время и ресурсы на разработку.

JSX объединяет JavaScript и разметку, позволяя создавать элементы интерфейса прямо в коде. Это улучшает читаемость и визуализацию структуры приложения. Например, разработчики сразу видят, как будет выглядеть страница, без необходимости перемещаться между файлами с HTML и JavaScript.

React предоставляет только базовые возможности для построения интерфейсов. Это оставляет свободу выбора дополнительных библиотек для управления состоянием (Redux, MobX), маршрутизации (React Router) и серверного рендеринга (Next.js). Такая гибкость делает React универсальным для разных типов проектов.

Поддержка современного управления состоянием с помощью хуков,с появлением хуков, таких как useState, useEffect и useContext, функциональные компоненты могут обрабатывать сложную логику без использования классов. Это делает код компактнее, легче читаемым и проще в тестировании. Например, с помощью useEffect можно легко управлять побочными эффектами, такими как запросы к серверу или обработка событий.

React использует однонаправленный поток данных, при котором информация передается сверху вниз (от родительских компонентов к дочерним). Это делает поведение приложения предсказуемым и уменьшает вероятность ошибок. Например, любое изменение состояния в родительском компоненте автоматически обновляет только те дочерние компоненты, которые зависят от этих данных.

Возможности для SEO и серверного рендеринга, традиционно JavaScript-приложения имели проблемы с SEO, так как поисковые роботы не могли анализировать контент, создаваемый динамически. React решает эту проблему с помощью инструментов серверного рендеринга, таких как Next.js. Это позволяет генерировать HTML-код на сервере, делая страницы быстрее загружаемыми и доступными для индексации.

React имеет огромное сообщество, которое регулярно создает новые инструменты, библиотеки и учебные материалы. Это делает процесс обучения и разработки проще. Проблемы, возникающие у разработчиков, как правило, уже решены другими, и найти готовое решение не составляет труда.

React был создан и активно поддерживается компанией Meta (Facebook). Это обеспечивает надежность и регулярные обновления библиотеки. Многие крупные компании, такие как Airbnb, Netflix, Uber и Twitter, используют React, что подтверждает его эффективность и востребованность.

Совместимость с TypeScript для типизации,React легко интегрируется с TypeScript, что делает код более безопасным и защищенным от ошибок. Типизация помогает избежать проблем на этапе разработки и упрощает поддержку больших проектов.

Улучшение пользовательского опыта (UX), React позволяет создавать отзывчивые, интерактивные приложения с плавной анимацией и мгновенными ответами на действия пользователя. Например, в интернет-магазинах с React пользователи могут добавлять товары в корзину, изменять параметры и фильтры, не дожидаясь полной перезагрузки страницы.

React подходит как для небольших проектов, так и для крупных корпоративных систем. Его архитектура позволяет добавлять новые компоненты и функции без значительных изменений в существующем коде.

React-приложения легко тестировать благодаря модульной структуре и поддержке таких инструментов, как Jest и React Testing Library. Каждый компонент можно тестировать отдельно, что сокращает количество багов в приложении.

React можно интегрировать в существующий проект. Например, если у вас уже есть приложение на другой технологии, вы можете начать использовать React для определенных частей интерфейса, таких как формы или таблицы, без необходимости переписывать весь код.

**1.3 MobX: управление состоянием в приложении**

MobX — это библиотека для управления состоянием в JavaScript-приложениях, которая особенно популярна среди разработчиков, использующих React. Она предлагает простой и гибкий подход к управлению состоянием, автоматизируя отслеживание изменений и обновление интерфейса. В отличие от Redux, MobX использует реактивный подход: когда данные изменяются, интерфейс обновляется автоматически, что делает разработку более интуитивной и упрощает работу с данными.

**Основные принципы работы MobX**

MobX работает на основе четырех основных концепций: обсерваблы (observables), вычисления (computeds), действия (actions) и реакции (reactions).

Обсерваблы (observables) – это состояния или данные, которые MobX отслеживает для автоматического обновления интерфейса. Любые изменения в обсерваблах приводят к тому, что MobX обновляет все связанные с ними компоненты. В MobX обсерваблы могут представлять собой как простые переменные, так и объекты или массивы. Пример:

class TodoStore {

todos = [];

constructor() {

makeAutoObservable(this);

}

addTodo(title) {

this.todos.push({ title, completed: false });

}

toggleTodo(index) {

this.todos[index].completed = !this.todos[index].completed;

}

get completedTodosCount() {

return this.todos.filter(todo => todo.completed).length;

}

}

const todoStore = new TodoStore();

export default todoStore;

Вычисления (computeds) – это производные значения, которые зависят от обсерваблов. Например, если у нас есть список задач, мы можем создать вычисляемое свойство, которое будет показывать количество выполненных задач. MobX автоматически обновит это значение при изменении обсерваблов.

import { observer } from 'mobx-react-lite';

import todoStore from './TodoStore';

const TodoList = observer(() => {

const addTodo = () => {

const title = prompt('Введите название задачи:');

if (title) todoStore.addTodo(title);

};

return (

<div>

<h1>Список задач</h1>

<ul>

{todoStore.todos.map((todo, index) => (

<li key={index} onClick={() => todoStore.toggleTodo(index)}>

{todo.title} {todo.completed ? '(Выполнено)' : ''}

</li>

))}

</ul>

<button onClick={addTodo}>Добавить задачу</button>

<p>Выполнено задач: {todoStore.completedTodosCount}</p>

</div>

);

});

export default TodoList;

Действия (actions) представляют собой методы, которые изменяют состояние. В MobX любое изменение состояния должно происходить через действие, что делает код более организованным и упрощает его отладку. Действия также помогают отделить логику изменения состояния от отображения.

Реакции (reactions) представляют собой автоматические эффекты, которые выполняются при изменении обсерваблов. Например, можно использовать реакции для синхронизации состояния приложения с локальным хранилищем или для выполнения сетевых запросов, когда данные меняются.

**1.4 Анализ существующих решений для туристических сайтов**

**Функционал туристических сайтов**

Основные функции туристических сайтов направлены на удовлетворение потребностей пользователей в поиске информации, бронировании и планировании поездок.

Среди ключевых функций можно выделить:

Поиск туров и экскурсий – удобный и быстрый поиск по направлениям, категориям (исторические, приключенческие, семейные и т.д.), датам и другим параметрам.

Система бронирования, возможность забронировать тур или экскурсию онлайн с выбором дат, мест и добавлением услуг (трансфер, проживание, питание).

Карты с основными достопримечательностями, маршруты путешествий и подробное описание туров и экскурсий.

Возможность оставлять отзывы и просматривать оценки от других пользователей, что помогает потенциальным клиентам выбрать подходящий тур.

Удобный интерфейс для управления бронированиями, отслеживания предстоящих поездок, сохранения избранных туров и просмотра истории.

Взаимодействие с соцсетями для публикации отзывов, фотографий и вдохновляющего контента.

Особенно важно для привлечения международных туристов.

Поддержка разных методов оплаты, включая кредитные карты, электронные кошельки и оплату через банки.

Адаптивный дизайн и приложения для смартфонов, позволяющие пользователям комфортно пользоваться сайтом на любых устройствах.

**Структура туристических сайтов**

Типичная структура туристического сайта отражает его основную цель: сделать путешествия доступными и понятными для пользователя.

Обычно содержит информацию о популярных направлениях, специальных предложениях и актуальных новостях, чтобы привлечь внимание пользователя.

Страница с удобной системой фильтров по направлениям, категориям, цене и другим параметрам.

Подробное описание, фото и видео, информация о местах, которые будут посещены, и услуги, включенные в стоимость.

Полезные статьи, путеводители и советы по путешествиям для вдохновения и привлечения аудитории.

Страница с контактной информацией и часто задаваемыми вопросами, а также возможностью связаться с поддержкой через чат или мессенджеры.

Доступ к истории заказов, настройкам профиля и платежным данным.

**Дизайн туристических сайтов**

Дизайн туристического сайта должен быть ярким, визуально привлекательным и вдохновляющим, так как путешествия часто связаны с эмоциями и впечатлениями.

Качественные фотографии, видеоролики и интерактивные карты помогают пользователю лучше представить себе поездку.

Простая и удобная навигация, где можно быстро найти нужную информацию, например, о турах или маршрутах.

Яркие и приглушенные цвета, приятные для глаз, помогают создать впечатление надежности и профессионализма.

Важность адаптивного дизайна трудно переоценить, поскольку пользователи все чаще заходят на сайты с мобильных устройств.

**Примеры популярных туристических сайтов**

TripAdvisor – мировой лидер в сфере туризма, TripAdvisor предлагает пользователям большое количество отзывов, рейтинг отелей, ресторанов, экскурсий и других объектов. Сайт прост в навигации и включает в себя обширный каталог туристических услуг с поддержкой бронирования и информационными страницами.

Airbnb – специализируется на предоставлении уникальных предложений по аренде жилья и организацию экскурсий. Airbnb имеет интуитивный интерфейс, поддерживает систему отзывов и предложений, а также включает мощные фильтры для поиска жилья и активности.

Expedia – один из крупнейших сайтов для планирования путешествий, который предоставляет функции бронирования отелей, билетов на самолеты, аренды автомобилей и организации туров. Интерфейс прост и удобен, а также поддерживает интеграции с внешними сервисами.

Booking.com – основное направление сайта — бронирование жилья, однако он также включает экскурсии и туры. Простой и понятный интерфейс позволяет легко находить нужные предложения и быстро бронировать жилье.

Глава 2. Проектирование и разработка веб-приложения для отрасли туризма.