Geekbrains

**Разработка интернет-магазина на React с использованием современных технологий и библиотек**

Программа: Разработчик

Специализация: Frontend разработка React

Пургин Антон Витальевич

Санкт-Петербург

2024г.

**Содержание**

[Содержание](#_heading=h.7mt9x4b37ce)

[Введение](#_heading=h.1fob9te)

[Теоретическая и практическая главы](#_heading=h.hst8x8f1r8tz)

[Заключение](#_heading=h.dhwmmf3wq1kk)

[Список используемой литературы](#_heading=h.m8jvbwhv7ujg)

[Приложения](#_heading=h.s6k7t2lwcp0k)

**Введение**

**Глава 1. Теоретические основы разработки на React**

1.1 Обзор React.js: основы, преимущества и недостатки

1.2 Основные компоненты React: State, Props, жизненный цикл компонента

1.3 Роутинг и навигация в React: библиотеки и их применение

1.4 Работа с данными: Fetch API, Redux, Context API

1.5. Современные технологии для React-приложений: Webpack, Babel, ESLint

1.6. Применение Node.js в React.

**Глава 2. Подготовка к ручному тестированию веб-приложения (~15 стр)**

2.1 Составление чек-листов

2.2 Составление тест-кейсов

2.3 Разработка тест-плана

2.4 Тест-дизайн и тест-аналитика

2.5 Баг-репорты и багтрекинговая система

**Заключение (~ 4 стр.)**

**Список используемой литературы**

**Приложения**

**Введение**

В данной работе представлено исследование особенностей разработки приложения на React.js. На сегодняшний день React.js является одним из самых популярных фреймворков для создания высокопроизводительных и интерактивных веб-приложений.

**Цель работы:** Разработать приложение на React, используя современные технологии и библиотеки, с целью демонстрации основных принципов фреймворка и его практического применения.

**Задачи:**

* Изучить теоретические основы разработки интернет-магазинов на React.
* Определить архитектуру и структуру проекта.
* Создать основные компоненты интернет-магазина: карточки товара, корзину, форму поиска, страницу товара.
* Реализовать взаимодействие с API: получение данных, добавление товаров в корзину.
* Проанализировать и оценить полученные результаты.

**Инструменты:** React.js, Node.js, Webpack, Babel, ESLint, Fetch API, Redux, Context API, Git.

**Состав команды:** Пургин Антон Витальевич (Frontend разработка React)

**Глава 1. Теоретические основы разработки на React**

**1.1. Обзор React.js: основы, преимущества и недостатки**

**Введение**

React.js - это популярная JavaScript-библиотека с открытым исходным кодом, разработанная Facebook (сейчас Meta), для создания пользовательских интерфейсов (UI). Она основана на концепции компонентного программирования и предоставляет инструменты для создания интерактивных и динамических веб-приложений.

**Основные концепции React.js:**

* **Компонентный подход:** В React приложения строятся из отдельных, независимых компонентов, которые отвечают за определенные части UI. Каждый компонент представляет собой функцию или класс, принимающий входные данные (props) и возвращающий JSX-код (разметка, похожая на HTML).
* **Виртуальный DOM:** React использует виртуальный DOM (Document Object Model) - внутреннюю структуру данных, которая представляет реальный DOM. Изменения в UI происходят сначала в виртуальном DOM, а затем React эффективно обновляет реальный DOM, что обеспечивает высокую производительность.
* **Односторонний поток данных:** Данные передаются сверху вниз от родительского компонента к дочерним компонентам через props. Это позволяет легко отслеживать изменения данных и избегать нежелательных побочных эффектов.
* **JSX (JavaScript XML):** React использует JSX-синтаксис, который позволяет создавать HTML-подобную разметку непосредственно в JavaScript-коде, что делает код более читаемым и организованным.

**Преимущества React.js:**

* **Высокая производительность:** Благодаря виртуальному DOM, React эффективно обновляет только те элементы UI, которые действительно изменились, что приводит к более быстрому рендерингу.
* **Компонентный подход:** Разделение кода на компоненты позволяет легко повторно использовать код, тестировать его и работать с ним в команде.
* **Односторонний поток данных:** Обеспечивает предсказуемость и легкость отладки, так как изменения в данных распространяются в одном направлении.
* **Большое сообщество:** React имеет большое и активное сообщество разработчиков, что обеспечивает доступ к большому количеству ресурсов, документации, библиотек и компонентов.
* **Удобство использования:** React обладает простой кривой обучения, что делает его доступным для разработчиков с различным уровнем опыта.
* **Поддержка серверного рендеринга:** React позволяет выполнять рендеринг на сервере, что улучшает SEO (Search Engine Optimization) и скорость загрузки приложения.

**Недостатки React.js:**

* **Кривая обучения:** Несмотря на то, что React относительно прост в освоении, некоторые концепции, такие как JSX и управление состоянием, могут быть сложными для начинающих.
* **JSX-синтаксис:** JSX-синтаксис, хотя и является удобным, может быть непривычным для разработчиков, привыкших к стандартному JavaScript-коду.
* **Сложность управления состоянием:** Для сложных приложений с множеством взаимосвязанных компонентов может потребоваться использовать библиотеки управления состоянием, такие как Redux или MobX.
* **Зависимость от JavaScript:** React приложения полностью зависят от JavaScript, что может быть проблемой для браузеров с ограниченной поддержкой JavaScript.

**Заключение:**

React.js - это мощная и гибкая библиотека, которая предоставляет все необходимые инструменты для создания высококачественных веб-приложений. Она обладает множеством преимуществ, таких как высокая производительность, компонентный подход, удобство использования и большая поддержка сообщества. Однако, как и у любой другой технологии, у React.js есть свои недостатки, которые необходимо учитывать при выборе фреймворка для проекта.

**Дополнительная информация:**

* [Официальная документация React](https://reactjs.org/)
* [React на GitHub](https://github.com/facebook/react)
* [React на Stack Overflow](https://stackoverflow.com/questions/tagged/reactjs)

**1.2. Основные компоненты React: State, Props, жизненный цикл компонента**

**1. Компоненты**

В React, приложения строятся из отдельных, независимых компонентов. Компонент - это базовый строительный блок, который представляет собой функцию или класс, принимающий входные данные (props) и возвращающий JSX-код (разметку, похожую на HTML).

**2. Props (Свойства)**

* **Определение:** Props - это не изменяемые данные, которые передаются из родительского компонента в дочерний компонент. Они используются для настройки поведения и внешнего вида дочернего компонента.
* **Пример:**

function Greeting(props) {

return <h1>Привет, {props.name}!</h1>;

}

ReactDOM.render(

<Greeting name="Мир" />,

document.getElementById('root')

);

В этом примере компонент Greeting получает имя через props.name.

**3. State (Состояние)**

* **Определение:** State - это изменяемые данные, хранящиеся внутри самого компонента. Они отвечают за динамическое поведение компонента и его реакцию на события.
* **Пример:**

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

handleClick = () => {

this.setState({ count: this.state.count + 1 });

}

render() {

return (

<div>

<p>Счетчик: {this.state.count}</p>

<button onClick={this.handleClick}>Увеличить счетчик</button>

</div>

);

}

}

ReactDOM.render(

<Counter />,

document.getElementById('root')

);

В этом примере компонент Counter хранит счет в this.state.count. При нажатии на кнопку handleClick обновляет состояние, увеличивая счетчик на 1.

**4. Жизненный цикл компонента**

Жизненный цикл компонента описывает все этапы, которые проходит компонент от создания до уничтожения. Знание жизненного цикла позволяет выполнять различные действия на определенных этапах, например, инициализацию данных, рендеринг, обработку событий и очистку ресурсов.

**Основные методы жизненного цикла:**

* **constructor(props):** Вызывается при создании компонента. Используется для инициализации состояния this.state.
* **static getDerivedStateFromProps(props, state):** Вызывается перед рендерингом, когда props изменились. Можно использовать для обновления состояния на основе новых props.
* **render():** Вызывается для рендеринга компонента. Возвращает JSX-код, представляющий UI компонента.
* **componentDidMount():** Вызывается после того, как компонент был добавлен в DOM. Используется для выполнения асинхронных действий, например, отправки запросов к API.
* **componentDidUpdate(prevProps, prevState):** Вызывается после обновления компонента. Можно использовать для обновления состояния на основе предыдущего состояния и свойств.
* **componentWillUnmount():** Вызывается перед тем, как компонент будет удален из DOM. Используется для очистки ресурсов, например, отмены таймеров или отмены подписок.

**Пример:**

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

componentDidMount() {

console.log('Компонент смонтирован!');

}

componentDidUpdate() {

console.log('Компонент обновлен!');

}

componentWillUnmount() {

console.log('Компонент будет удален!');

}

*// ...*

}

**Важно:**

* Изменение состояния (this.setState()) приводит к перерендерингу компонента.
* Props - это неизменяемые данные, а state - изменяемые.
* Жизненный цикл компонента позволяет выполнять действия на определенных этапах его работы.

**Заключение:**

Props, State и жизненный цикл компонента - это три основных концепции, которые позволяют создавать динамические и интерактивные компоненты в React. Понимание этих концепций является необходимым условием для разработки сложных React-приложений.

**1.3. Роутинг и навигация в React: библиотеки и их применение**

**Введение**

Роутинг в React — это механизм, который позволяет создавать одностраничные приложения (SPA) с несколькими страницами (или представлениями), которые отображаются в зависимости от URL-адреса. Навигация в React — это процесс перехода между различными страницами приложения.

**Библиотеки для роутинга в React**

Существует несколько популярных библиотек для реализации роутинга в React:

* **React Router:** Одна из самых популярных и широко используемых библиотек. Предлагает богатый набор функций, включая:
  + **Nested Routing:** Создание вложенных маршрутов для организации приложения.
  + **Dynamic Routing:** Создание маршрутов, которые могут принимать параметры, например, ID продукта или категорию.
  + **URL-параметры:** Доступ к параметрам, переданным в URL, через useParams hook.
  + **History:** Доступ к истории навигации, для перехода назад или вперед.
* **React Router Dom:** Специализированная версия React Router для веб-приложений. Предоставляет компоненты, такие как <BrowserRouter>, <Routes>, <Route>, <Link>, которые легко интегрируются с React-приложениями.
* **React Router Native:** Вариант React Router для разработки мобильных приложений с использованием React Native.

**Применение библиотек роутинга**

**Пример с React Router Dom:**

import { BrowserRouter, Routes, Route, Link } from 'react-router-dom';

function HomePage() {

return <h1>Домашняя страница</h1>;

}

function AboutPage() {

return <h1>О нас</h1>;

}

function ProductPage() {

return <h1>Страница товара</h1>;

}

function App() {

return (

<BrowserRouter>

<div>

<nav>

<ul>

<li>

<Link to="/">Домашняя</Link>

</li>

<li>

<Link to="/about">О нас</Link>

</li>

</ul>

</nav>

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/about" element={<AboutPage />} />

<Route path="/product/:id" element={<ProductPage />} />

</Routes>

</div>

</BrowserRouter>

);

}

ReactDOM.render(<App />, document.getElementById('root'));

В этом примере:

* <BrowserRouter> оборачивает приложение, чтобы обеспечить роутинг.
* <Routes> определяет набор маршрутов.
* <Route> определяет отдельный маршрут с path, element (компонент, который отображается при совпадении URL) и children (дочерние маршруты).
* <Link> создает ссылку, которая перенаправляет пользователя на определенный URL, используя to атрибут.
* path="/product/:id" определяет динамический маршрут, который принимает параметр id из URL.

**Ключевые компоненты React Router:**

* <BrowserRouter> - Создает контекст роутинга для веб-приложений.
* <Routes> - Определяет набор маршрутов для приложения.
* <Route> - Определяет отдельный маршрут с path и element (компонент, который отображается при совпадении URL).
* <Link> - Создает ссылку, которая перенаправляет пользователя на определенный URL.
* useParams - Hook для доступа к параметрам, переданным в URL.
* useNavigate - Hook для программирования навигации.

**Заключение**

Роутинг и навигация в React являются важной частью разработки одностраничных приложений. Библиотеки для роутинга, такие как React Router, позволяют легко создавать сложные приложения с несколькими страницами, динамическими маршрутами и удобной навигацией.

**Дополнительная информация:**

* [Документация React Router](https://reactrouter.com/docs/en/v6)
* [Примеры использования React Router](https://reactrouter.com/docs/en/v6/examples)

**1.4. Работа с данными: Fetch API, Redux, Context API**

В React работа с данными является неотъемлемой частью создания динамических приложений. Для управления данными и их передачи между компонентами используются различные инструменты.

**1. Fetch API**

* **Определение:** Fetch API — это встроенный в браузер API для отправки запросов к серверу и получения данных.
* **Применение:** Fetch API позволяет отправлять HTTP-запросы (GET, POST, PUT, DELETE) для получения данных с сервера, а также для отправки данных на сервер.
* **Пример:**

async function fetchProducts() {

const response = await fetch('https://api.example.com/products');

const data = await response.json();

return data;

}

fetchProducts().then(products => {

console.log(products);

});

В этом примере:

* fetch — функция для отправки запроса к API.
* response.json() — метод для преобразования полученного ответа в JSON-формат.
* then — метод для обработки полученных данных.

**2. Redux**

* **Определение:** Redux — это библиотека для управления состоянием приложения, которая помогает централизовать и организовать данные. Она использует модель однонаправленного потока данных:
  + **Actions:** Объекты, которые описывают событие, произошедшее в приложении.
  + **Reducers:** Функции, которые обновляют состояние приложения на основе полученных действий.
  + **Store:** Единый источник данных, доступный всем компонентам приложения.
* **Преимущества:**
  + **Централизованное управление состоянием:** Данные хранятся в одном месте, что облегчает отладку и тестирование.
  + **Предсказуемость:** Изменения состояния происходят предсказуемо, что делает приложение более стабильным.
  + **Повторное использование:** Reducers можно повторно использовать в разных частях приложения.
* **Пример:**

*// Actions*

const ADD\_ITEM = 'ADD\_ITEM';

const REMOVE\_ITEM = 'REMOVE\_ITEM';

*// Reducer*

const cartReducer = (state = [], action) => {

switch (action.type) {

case ADD\_ITEM:

return [...state, action.payload];

case REMOVE\_ITEM:

return state.filter(item => item.id !== action.payload);

default:

return state;

}

};

*// Store*

const store = createStore(cartReducer);

*// Component*

function Cart() {

const cartItems = useSelector(state => state);

*// ...*

return (

*// ...*

);

}

В этом примере:

* ADD\_ITEM и REMOVE\_ITEM - типы действий.
* cartReducer — функция, которая обновляет состояние корзины.
* store — объект, хранящий состояние корзины.
* useSelector — hook, который позволяет получить доступ к состоянию приложения из компонента.

**3. Context API**

* **Определение:** Context API — это встроенный в React механизм для обмена данными между компонентами, которые не связаны напрямую через props.
* **Преимущества:**
  + **Простой способ обмена данными:** Не нужно передавать props через множество уровней вложенности компонентов.
  + **Глобальная доступность данных:** Данные, хранящиеся в контексте, доступны всем компонентам, которые подписаны на этот контекст.
* **Пример:**

const ThemeContext = createContext({

theme: 'light',

toggleTheme: () => {}

});

function App() {

const [theme, setTheme] = useState('light');

const toggleTheme = () => {

setTheme(prevTheme => prevTheme === 'light' ? 'dark' : 'light');

};

return (

<ThemeContext.Provider value={{ theme, toggleTheme }}>

{/\* ... \*/}

</ThemeContext.Provider>

);

}

function MyComponent() {

const { theme, toggleTheme } = useContext(ThemeContext);

*// ...*

return (

*// ...*

);

}

В этом примере:

* ThemeContext — контекст для хранения темы.
* App — компонент, который предоставляет контекст для своих потомков.
* MyComponent — компонент, который получает доступ к контексту и использует его для управления темой.

**Заключение**

Fetch API, Redux и Context API — это мощные инструменты для управления данными в React-приложениях. Fetch API обеспечивает взаимодействие с сервером, Redux помогает централизовать и управлять состоянием, а Context API позволяет легко передавать данные между компонентами. Выбор инструмента зависит от сложности проекта, объема данных и архитектуры приложения.

**1.5. Современные технологии для React-приложений: Webpack, Babel, ESLint**

**Введение**

Современная разработка React-приложений требует использования дополнительных инструментов, которые оптимизируют процесс, повышают качество кода и обеспечивают совместимость с различными браузерами. К ключевым технологиям, применяемым в React-проектах, относятся Webpack, Babel и ESLint.

**1. Webpack**

* **Определение:** Webpack — это модульный бандлер, который собирает все файлы проекта, включая JavaScript, CSS, изображения и другие ресурсы, в один или несколько файлов (бандлов).
* **Преимущества:**
  + **Модульность:** Webpack позволяет разбить приложение на отдельные модули, что улучшает организацию кода и его повторное использование.
  + **Оптимизация:** Webpack оптимизирует бандлы, сжимает код и изображения, что ускоряет загрузку приложения.
  + **Поддержка различных типов файлов:** Webpack работает с различными типами файлов, включая JavaScript, CSS, HTML, изображения, шрифты и др.
  + **Разделение кода:** Webpack может разделить приложение на несколько бандлов, которые загружаются только при необходимости.
* **Пример конфигурации:**

const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {

entry: './src/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist'),

},

module: {

rules: [

{

test: /\.(js|jsx)$/,

exclude: /node\_modules/,

use: 'babel-loader',

},

{

test: /\.css$/,

use: ['style-loader', 'css-loader'],

},

{

test: /\.(png|jpg|gif|svg)$/,

use: [

{

loader: 'file-loader',

options: {

outputPath: 'assets/images',

},

},

],

},

],

},

plugins: [

new HtmlWebpackPlugin({

template: './public/index.html',

}),

],

};

В этом примере:

* entry - точка входа для сборки проекта.
* output - настройки выходного файла.
* module - настройки для обработки различных типов файлов.
* plugins - плагины для расширения функциональности Webpack.

**2. Babel**

* **Определение:** Babel — это транспилятор, который преобразует JavaScript-код, написанный с использованием современных функций языка (ES6+), в совместимый код для старых браузеров (ES5).
* **Преимущества:**
  + **Совместимость:** Babel позволяет использовать новейшие JavaScript-фичи, такие как классы, стрелочные функции, let/const, модули ES6, которые не поддерживаются всеми браузерами.
  + **Улучшенный синтаксис:** Babel обеспечивает более простой и читаемый синтаксис JavaScript.
  + **Повышение производительности:** Babel оптимизирует код, удаляя ненужные элементы и улучшая его эффективность.
* **Пример конфигурации:**

{

"presets": [

[

"@babel/preset-env",

{

"targets": "defaults"

}

],

"@babel/preset-react"

],

"plugins": [

"@babel/plugin-transform-runtime"

]

}

В этом примере:

* presets - наборы плагинов для преобразования кода.
* plugins - дополнительные плагины для расширения функциональности Babel.

**3. ESLint**

* **Определение:** ESLint — это инструмент для статического анализа кода, который выявляет синтаксические ошибки, предупреждения о стиле кодирования и другие проблемы в JavaScript-коде.
* **Преимущества:**
  + **Обнаружение ошибок:** ESLint позволяет обнаруживать ошибки на ранних этапах разработки, что упрощает отладку и предотвращает появление дефектов.
  + **Улучшение качества кода:** ESLint обеспечивает соблюдение единого стиля кодирования, что повышает читаемость и понимание кода.
  + **Предотвращение дублирования кода:** ESLint может выявлять дублирование кода и предлагать способы его оптимизации.
* **Пример конфигурации:**

module.exports = {

env: {

browser: true,

es2021: true,

},

extends: ['eslint:recommended', 'plugin:react/recommended'],

parserOptions: {

ecmaFeatures: {

jsx: true,

},

ecmaVersion: 12,

sourceType: 'module',

},

plugins: ['react'],

rules: {

'react/prop-types': 'off',

'react/react-in-jsx-scope': 'off',

},

};

В этом примере:

* env - определяет среду, в которой выполняется код.
* extends - наборы правил для линтинга.
* parserOptions - настройки синтаксического анализа.
* plugins - дополнительные плагины для линтинга.
* rules - настройки для конкретных правил линтинга.

**Заключение**

Webpack, Babel и ESLint — это мощные инструменты, которые значительно упрощают процесс разработки React-приложений, повышают качество кода и обеспечивают его совместимость с различными браузерами. Их использование является стандартной практикой в современном React-разработке.

**Дополнительная информация:**

* [Документация Webpack](https://webpack.js.org/)
* [Документация Babel](https://babeljs.io/)
* [Документация ESLint](https://eslint.org/)

**1.6. Применение Node.js в React.**

**1. Введение**

React.js - популярная библиотека JavaScript для создания пользовательских интерфейсов. Она предоставляет инструменты для построения интерактивных и динамических веб-приложений. Node.js, в свою очередь, является средой выполнения JavaScript, которая позволяет выполнять код JavaScript вне браузера.

Сочетание React.js и Node.js позволяет создавать полнофункциональные веб-приложения, где React.js используется для фронтенда, а Node.js для бэкенда.

**Преимущества:**

* **Повышенная производительность:** React.js использует виртуальный DOM, что значительно ускоряет рендеринг.
* **Удобство разработки:** Компонентный подход в React.js позволяет разбивать сложные интерфейсы на более мелкие, легко управляемые части.
* **Обширный набор инструментов:** Экосистема Node.js предоставляет множество пакетов, которые упрощают процесс разработки.
* **Создание полнофункциональных приложений:** Сочетание React.js и Node.js позволяет строить как фронтенд, так и бэкенд одного приложения, используя один язык программирования.

**Цель работы:**

Изучить возможности применения Node.js в React.js и рассмотреть практические примеры.

**2. Node.js: основы**

Node.js - это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом, которая позволяет выполнять JavaScript-код вне браузера. Она основана на движке JavaScript V8, разработанном OpenAI для браузера Chrome.

**Ключевые особенности Node.js:**

* **Неблокирующая модель ввода-вывода:** Node.js использует асинхронный ввод-вывод, что позволяет ему эффективно обрабатывать множество запросов одновременно.
* **Событийно-ориентированная архитектура:** Код в Node.js выполняется в ответ на события, что позволяет оптимизировать использование ресурсов.
* **Модульность:** Node.js предоставляет модульную систему, которая позволяет разбивать приложения на небольшие, независимые части.
* **Большое сообщество:** Node.js имеет активное и быстро растущее сообщество разработчиков, предоставляющее поддержку и ресурсы.

**Пример:**

*// Пример создания сервера HTTP в Node.js*

const http = require('http');

const server = http.createServer((req, res) => {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('Hello, world!');

});

server.listen(3000, () => {

console.log('Сервер запущен на порту 3000');

});

Этот код создает HTTP-сервер, который слушает порт 3000. Когда к серверу поступает запрос, он отправляет в ответ текст “Hello, world!”.

**3. React.js: основы**

React.js - это библиотека JavaScript для построения пользовательских интерфейсов, разработанная Facebook. Она использует компонентный подход, где интерфейс приложения делится на небольшие, независимые блоки.

**Ключевые особенности React.js:**

* **Виртуальный DOM:** React.js использует виртуальный DOM для оптимизации рендеринга. Он сравнивает виртуальный DOM с реальным и обновляет только те части, которые изменились.
* **Однонаправленный поток данных:** Данные в React.js передаются сверху вниз по иерархии компонентов, что упрощает управление состоянием приложения.
* **JSX:** React.js использует язык JSX, который позволяет встраивать HTML-код в JavaScript.
* **Использование компонентов:** Компонентный подход в React.js позволяет разбивать интерфейс приложения на более мелкие, управляемые блоки.

**Пример:**

*// Пример создания простого компонента в React.js*

function Welcome(props) {

return <h1>Привет, {props.name}!</h1>;

}

ReactDOM.render(

<Welcome name="Мир" />,

document.getElementById('root')

);

Этот код создает компонент Welcome, который принимает имя в качестве аргумента и выводит приветственное сообщение. Затем компонент рендерится в HTML-элемент с ID root.

**4. Интеграция Node.js и React.js**

Node.js и React.js могут быть интегрированы для создания полнофункциональных веб-приложений. Node.js может использоваться для построения бэкенда, обработки запросов, управления данными и предоставления API для React.js-фронтенда.

**Типичная архитектура:**

* **Фронтенд (React.js):** Отвечает за пользовательский интерфейс и взаимодействие с пользователем.
* **Бэкенд (Node.js):** Обрабатывает запросы от фронтенда, управляет данными, предоставляет API.
* **База данных:** Хранит данные приложения.

**Пример:**

**Фронтенд (React.js):**

*// Компонент для получения данных от API*

function PostList() {

const [posts, setPosts] = useState([]);

useEffect(() => {

fetch('/api/posts')

.then(response => response.json())

.then(data => setPosts(data));

}, []);

return (

<ul>

{posts.map(post => (

<li key={post.id}>{post.title}</li>

))}

</ul>

);

}

**Бэкенд (Node.js):**

*// API для получения списка постов*

const express = require('express');

const app = express();

app.get('/api/posts', (req, res) => {

const posts = [

{id: 1, title: 'Заголовок 1'},

{id: 2, title: 'Заголовок 2'},

];

res.json(posts);

});

app.listen(3000, () => {

console.log('Сервер запущен на порту 3000');

});

В этом примере React.js-компонент PostList использует fetch для получения данных от Node.js-API /api/posts. Node.js-сервер отвечает на запрос, отправляя JSON-массив с данными о постах.

5. Примеры использования

**5.1. Создание блогов:**

Node.js может использоваться для создания REST API для управления постами, комментариями и пользователями. React.js может использоваться для построения фронтенда с интерфейсом для написания постов, просмотра и редактирования статей.

**5.2. Разработка онлайн-магазинов:**

Node.js может использоваться для создания API для управления продуктами, корзинами и заказами. React.js может использоваться для построения фронтенда с каталогом товаров, корзиной и формой оформления заказа.

**5.3. Создание чатов:**

Node.js может использоваться для создания сервера WebSockets, который будет обрабатывать сообщения от пользователей. React.js может использоваться для построения фронтенда с чатом.

**5.4. Разработка игр:**

Node.js может использоваться для создания сервера игры, который будет обрабатывать логику игры и взаимодействие между игроками. React.js может использоваться для построения графического интерфейса игры.

**6. Заключение**

Сочетание Node.js и React.js предоставляет мощные инструменты для создания современных веб-приложений. Node.js обеспечивает эффективную обработку запросов и управление данными, а React.js позволяет создавать интерактивные и динамические пользовательские интерфейсы.

**Преимущества данного сочетания:**

* **Повышенная производительность:** Благодаря использованию виртуального DOM, асинхронного ввода-вывода и других оптимизаций.
* **Удобство разработки:** Компонентный подход React.js и модульная система Node.js упрощают процесс разработки.
* **Создание полнофункциональных приложений:** Сочетание React.js и Node.js позволяет строить как фронтенд, так и бэкенд одного приложения, используя один язык программирования.

**В будущем:**

* Ожидается дальнейшее развитие и совершенствование обоих технологий.
* Появление новых инструментов и библиотек, упрощающих разработку с использованием Node.js и React.js.
* Расширение сферы применения Node.js и React.js для создания различных типов веб-приложений.

**Глава 2. Пошаговое описание создания сайта интернет-магазина экскурсий на React.js**

**2.1. Создание проекта**

* **Инициализация проекта:**
  + Откройте командную строку (терминал).
  + Перейдите в желаемую директорию для проекта.
  + Выполните команду npx create-react-app my-diplom .
  + Это создаст новую директорию с базовой структурой проекта React.
* **Переход в директорию проекта:**
  + cd my-diplom
* **Запуск проекта:**
  + npm start
  + Это запустит сервер разработки, и мы сможем открыть сайт в браузере по адресу http://localhost:3000.

**2.2. Создание компонентов**

* **Header:** Создаем компонент Header.js в директории src/components с следующим кодом:

import React, { useState } from 'react'

import { FaShoppingCart } from "react-icons/fa";

import Order from './Order';

const showOrders = (*props*) => {

    let summa = 0

*props*.orders.forEach(*el* => summa += Number(*el*.price))

    return (

        <div>

            {*props*.orders.map(*el* => (

                <Order *onDelete*={*props*.onDelete} *key*={*el*.id} *item*={*el*} />

            ))}

            <p *className*='summa'>Сумма: {new Intl.NumberFormat().format(summa)}$</p>

        </div>

    )

}

const showNothing = () => {

    return (

        <div *className*='empty'>

            <h2>Товаров нет</h2>

        </div>

    )

}

export default function Header(*props*) {

    let [cartOpen, setCartOpen] = useState(false)

    return (

        <header>

            <div *className*='container'>

                <div *className*='logo'>

                    <svg *height*="99px" *viewBox*="0 0 4338.3 1883.5" *xmlns*="http://www.w3.org/2000/svg">

                        <path *d*="m2573.8 826.55-116.8-124.1-153.2 161.76c-26.87 5.3-52.78 14.56-77.29 28.11-34.51 19.08-62.92 45.1-84.82 77.11-21.84-31.63-50.03-57.4-84.11-76.53-41.27-23.15-86.55-34.13-133.84-34.13-19.94 0-39.39 1.86-58.07 5.82h-33.3l504.69-532.64 402.67 422.23c-2.91 0.71-5.81 1.45-8.71 2.2-46.34 12.14-90.62 30.39-131.76 54.96l-25.46 15.21z" *fill*="#FFED00"></path>

                        <path *d*="m3523.1 864.59v85.07c-20.78-22.97-45.46-41.96-73.57-56.85-44.59-23.62-93.34-34.04-143.62-34.04-40.57 0-80.13 6.79-117.36 21.73l16.53-36.37-53.75-32.56c-41.07-24.88-85.44-43.12-131.87-55.21-47.3-12.32-95.82-18.28-144.69-18.28-35.98 0-71.77 3.3-107.02 10.02l-147.95-208.42 486.14-539.68 810.65 864.59h-393.51z" *fill*="#FFED00"></path>

                        <path d="m-0 817.53h117.78v412.96h255.19v95.97h-372.97v-508.93zm825.2 117.78v391.15h-108.33v-45.08c-28.11 33.93-68.83 50.9-122.14 50.9-36.84 0-70.16-8.24-99.97-24.72s-53.2-39.99-70.16-70.53c-16.96-30.53-25.45-65.92-25.45-106.15s8.48-75.61 25.45-106.15c16.96-30.53 40.35-54.04 70.16-70.53 29.81-16.48 63.13-24.72 99.97-24.72 49.92 0 88.94 15.76 117.05 47.26v-41.44h113.42zm-210.84 303.9c28.6 0 52.34-9.81 71.25-29.45 18.9-19.63 28.35-45.92 28.35-78.88s-9.45-59.26-28.35-78.88c-18.91-19.63-42.65-29.45-71.25-29.45-29.08 0-53.07 9.81-71.98 29.45-18.9 19.63-28.35 45.92-28.35 78.88s9.45 59.26 28.35 78.88c18.91 19.63 42.9 29.45 71.98 29.45zm816.47-309.72c48.95 0 87.85 14.42 116.69 43.26s43.26 72.1 43.26 129.78v223.93h-113.42v-206.48c0-31.02-6.42-54.16-19.27-69.43-12.84-15.27-31.14-22.9-54.89-22.9-26.65 0-47.74 8.61-63.25 25.81-15.51 17.21-23.26 42.78-23.26 76.71v196.3h-113.42v-206.48c0-61.56-24.72-92.33-74.16-92.33-26.17 0-47.01 8.61-62.53 25.81-15.51 17.21-23.26 42.78-23.26 76.71v196.3h-113.42v-391.15h108.33v45.08c14.54-16.48 32.35-29.08 53.44-37.81 21.08-8.73 44.23-13.09 69.43-13.09 27.63 0 52.59 5.46 74.89 16.36s40.23 26.78 53.8 47.62c16-20.36 36.23-36.11 60.71-47.26s51.26-16.73 80.33-16.73zm492.94 0c36.35 0 69.43 8.36 99.24 25.08 29.81 16.73 53.2 40.23 70.16 70.53s25.45 65.56 25.45 105.79-8.48 75.49-25.45 105.79c-16.96 30.29-40.35 53.8-70.16 70.53-29.81 16.72-62.89 25.08-99.24 25.08-49.92 0-89.18-15.76-117.78-47.26v182.49h-113.42v-532.2h108.33v45.08c28.11-33.93 69.07-50.9 122.87-50.9zm-19.63 309.72c29.08 0 52.95-9.81 71.61-29.45 18.66-19.63 27.99-45.92 27.99-78.88s-9.33-59.26-27.99-78.88c-18.66-19.63-42.53-29.45-71.61-29.45s-52.95 9.81-71.61 29.45c-18.66 19.63-28 45.92-28 78.88s9.33 59.26 28 78.88c18.66 19.63 42.53 29.45 71.61 29.45zm687.06-303.9v391.15h-108.33v-45.08c-28.11 33.93-68.83 50.9-122.14 50.9-36.84 0-70.16-8.24-99.97-24.72s-53.2-39.99-70.16-70.53c-16.96-30.53-25.45-65.92-25.45-106.15s8.48-75.61 25.45-106.15c16.96-30.53 40.35-54.04 70.16-70.53 29.81-16.48 63.13-24.72 99.97-24.72 49.92 0 88.94 15.76 117.05 47.26v-41.44h113.42zm-210.84 303.9c28.6 0 52.34-9.81 71.25-29.45 18.9-19.63 28.35-45.92 28.35-78.88s-9.45-59.26-28.35-78.88c-18.91-19.63-42.65-29.45-71.25-29.45-29.08 0-53.07 9.81-71.98 29.45-18.9 19.63-28.35 45.92-28.35 78.88s9.45 59.26 28.35 78.88c18.91 19.63 42.9 29.45 71.98 29.45zm694.33-279.18c-43.62-25.21-90.4-41.44-140.32-48.71v415.14h-117.78v-415.14c-49.92 7.27-96.94 23.5-141.05 48.71l-39.99-87.97c34.9-20.84 72.7-36.6 113.42-47.26s82.64-16 125.78-16c43.62 0 85.91 5.33 126.87 16 40.95 10.66 78.64 26.42 113.05 47.26l-39.99 87.97zm231.2 372.25c-41.2 0-78.16-8.61-110.88-25.81-32.72-17.21-58.28-41.08-76.7-71.61s-27.63-65.19-27.63-103.97 9.21-73.43 27.63-103.97c18.42-30.53 43.99-54.41 76.7-71.61 32.72-17.21 69.68-25.81 110.88-25.81s78.03 8.61 110.51 25.81c32.47 17.2 57.92 41.08 76.34 71.61s27.63 65.19 27.63 103.97-9.21 73.43-27.63 103.97c-18.42 30.53-43.87 54.41-76.34 71.61-32.48 17.21-69.31 25.81-110.51 25.81zm0-93.06c29.08 0 52.95-9.81 71.61-29.45 18.66-19.63 27.99-45.92 27.99-78.88s-9.33-59.26-27.99-78.88c-18.66-19.63-42.53-29.45-71.61-29.45s-53.07 9.81-71.98 29.45c-18.9 19.63-28.35 45.92-28.35 78.88s9.45 59.26 28.35 78.88c18.91 19.63 42.9 29.45 71.98 29.45zm684.15-303.9v391.15h-107.6v-46.53c-15.03 16.96-32.96 29.93-53.8 38.9-20.84 8.96-43.38 13.45-67.62 13.45-51.37 0-92.09-14.79-122.14-44.35-30.05-29.57-45.07-73.43-45.07-131.59v-221.02h113.42v204.3c0 63.01 26.42 94.52 79.25 94.52 27.14 0 48.95-8.84 65.43-26.54 16.48-17.69 24.72-43.99 24.72-78.88v-193.39h113.42zm213.75 51.62c13.57-18.9 31.87-33.2 54.89-42.89s49.56-14.54 79.61-14.54v104.7c-12.6-0.97-21.08-1.46-25.45-1.46-32.47 0-57.92 9.09-76.34 27.26-18.42 18.18-27.63 45.44-27.63 81.79v184.67h-113.42v-391.15h108.33v51.62z" fill="#fff"></path>

                        <path d="m106.25 1567c24.01 0 45.53 5.37 64.58 16.1 19.04 10.74 33.99 25.63 44.84 44.67s16.28 40.57 16.28 64.58-5.43 45.53-16.28 64.58c-10.85 19.04-25.8 33.93-44.84 44.67-19.04 10.73-40.57 16.1-64.58 16.1-21.46 0-40.8-3.69-58-11.08s-31.57-18.12-43.11-32.2l35.66-33.24c17.08 19.16 37.97 28.74 62.67 28.74 17.78 0 32.84-5.19 45.19-15.58s20.37-24.36 24.06-41.9h-98.34v-42.93h97.64c-3.92-16.62-12-29.9-24.24-39.82-12.23-9.93-27.01-14.89-44.32-14.89-24.7 0-45.59 9.58-62.67 28.74l-35.66-33.24c11.54-14.08 25.91-24.81 43.11-32.2s36.53-11.08 58-11.08zm277.06 175.9h-28.05v70.64h-54.02v-186.29h54.02v72.37h29.09l50.55-72.37h57.48l-65.1 89.34 69.6 96.95h-63.71l-49.86-70.64zm252.47 73.41c-19.85 0-37.68-4.1-53.5-12.29-15.81-8.2-28.16-19.57-37.05-34.11s-13.33-31.05-13.33-49.52 4.45-34.97 13.33-49.51c8.89-14.54 21.24-25.91 37.05-34.11 15.81-8.19 33.64-12.29 53.5-12.29 19.62 0 36.76 4.1 51.42 12.29 14.66 8.2 25.34 19.91 32.03 35.15l-41.9 22.51c-9.69-17.08-23.66-25.62-41.9-25.62-14.08 0-25.74 4.62-34.97 13.85-9.24 9.23-13.85 21.81-13.85 37.74s4.62 28.51 13.85 37.74c9.23 9.24 20.89 13.85 34.97 13.85 18.47 0 32.43-8.54 41.9-25.63l41.9 22.86c-6.7 14.77-17.37 26.31-32.03 34.63-14.66 8.31-31.8 12.46-51.42 12.46zm227.89-73.41h-28.05v70.64h-54.02v-186.29h54.02v72.37h29.09l50.55-72.37h57.48l-65.1 89.34 69.6 96.95h-63.71l-49.86-70.64zm352.89-115.65v155.13c0 34.16-8.66 59.56-25.97 76.18s-42.47 24.93-75.49 24.93c-17.54 0-34.11-2.2-49.69-6.58-15.58-4.39-28.68-10.73-39.3-19.04l21.46-38.78c7.85 6.47 17.43 11.55 28.74 15.24s22.74 5.54 34.28 5.54c18.01 0 31.16-4.15 39.48-12.46 8.31-8.31 12.46-21.01 12.46-38.09v-6.23c-6.93 6.93-15.12 12.23-24.58 15.93-9.46 3.69-19.62 5.54-30.47 5.54-24.47 0-43.86-6.98-58.18-20.95-14.31-13.97-21.46-34.92-21.46-62.85v-93.49h54.02v85.53c0 30.24 12.58 45.36 37.74 45.36 12.93 0 23.32-4.21 31.16-12.64 7.85-8.43 11.77-20.95 11.77-37.57v-80.68h54.02zm176.29-2.77c19.85 0 37.17 3.92 51.94 11.77s26.2 19.04 34.28 33.59c8.08 14.54 12.12 31.51 12.12 50.9 0 18.93-3.81 35.6-11.43 50.03s-18.35 25.63-32.2 33.59-29.78 11.94-47.79 11.94c-23.77 0-42.36-7.5-55.75-22.51v86.92h-54.02v-159.63c0-19.16 4.27-36.01 12.81-50.55s20.55-25.86 36.01-33.93c15.47-8.08 33.47-12.12 54.02-12.12zm-2.42 147.51c14.08 0 25.51-4.73 34.28-14.2 8.77-9.46 13.16-21.82 13.16-37.05 0-15.47-4.39-27.99-13.16-37.57s-20.2-14.37-34.28-14.37-25.45 4.73-34.11 14.2c-8.66 9.46-12.99 21.93-12.99 37.4s4.27 27.94 12.81 37.4 19.97 14.2 34.28 14.2zm256.97 44.32c-19.85 0-37.68-4.1-53.5-12.29-15.81-8.2-28.16-19.57-37.05-34.11s-13.33-31.05-13.33-49.52 4.45-34.97 13.33-49.51c8.89-14.54 21.24-25.91 37.05-34.11 15.81-8.19 33.64-12.29 53.5-12.29 19.62 0 36.76 4.1 51.42 12.29 14.66 8.2 25.34 19.91 32.03 35.15l-41.9 22.51c-9.69-17.08-23.66-25.62-41.9-25.62-14.08 0-25.74 4.62-34.97 13.85-9.24 9.23-13.85 21.81-13.85 37.74s4.62 28.51 13.85 37.74c9.23 9.24 20.89 13.85 34.97 13.85 18.47 0 32.43-8.54 41.9-25.63l41.9 22.86c-6.7 14.77-17.37 26.31-32.03 34.63-14.66 8.31-31.8 12.46-51.42 12.46zm329.69-189.06v186.29h-51.59v-21.81c-6.93 8.08-15.41 14.2-25.45 18.35-10.04 4.16-20.72 6.23-32.03 6.23-24.47 0-43.92-7.04-58.35-21.12-14.42-14.08-21.64-34.98-21.64-62.68v-105.26h54.02v97.3c0 30.01 12.58 45.01 37.74 45.01 13.16 0 23.66-4.21 31.51-12.64s11.77-20.95 11.77-37.57v-92.1h54.02zm264.93 0v186.29h-51.59v-21.81c-6.93 8.08-15.41 14.2-25.45 18.35-10.04 4.16-20.72 6.23-32.03 6.23-24.47 0-43.92-7.04-58.35-21.12-14.42-14.08-21.64-34.98-21.64-62.68v-105.26h54.02v97.3c0 30.01 12.58 45.01 37.74 45.01 13.16 0 23.66-4.21 31.51-12.64s11.77-20.95 11.77-37.57v-92.1h54.02zm321.37 88.99c21.93 7.38 32.9 22.16 32.9 44.32 0 16.85-6.35 29.89-19.04 39.13s-31.62 13.85-56.78 13.85h-99.38v-186.29h95.22c23.32 0 41.55 4.21 54.71 12.64s19.74 20.26 19.74 35.49c0 9.24-2.37 17.37-7.1 24.41s-11.48 12.52-20.26 16.45zm-92.8-12.47h40.86c18.93 0 28.39-6.7 28.39-20.08 0-13.16-9.46-19.74-28.39-19.74h-40.86v39.82zm45.36 73.06c19.85 0 29.78-6.81 29.78-20.43 0-7.15-2.31-12.41-6.93-15.76s-11.89-5.02-21.81-5.02h-46.4v41.21h45.36zm448.8 15.93c-11.54 7.85-26.03 13.97-43.46 18.35-17.43 4.39-35.95 6.58-55.57 6.58-24.47 0-45.25-3.06-62.33-9.18s-30.01-14.66-38.78-25.63-13.16-23.49-13.16-37.57c0-12.93 3.58-24.35 10.73-34.28s16.97-17.31 29.43-22.16c-9.46-5.08-16.79-12-21.99-20.78-5.19-8.77-7.79-18.58-7.79-29.43 0-13.16 4.04-25.16 12.12-36.01s20.31-19.5 36.7-25.97c16.39-6.46 36.47-9.69 60.25-9.69 15.7 0 31.04 1.56 46.05 4.68 15 3.12 28.05 7.33 39.13 12.64l-15.58 42.94c-21.24-9.93-43.75-14.89-67.52-14.89-18.01 0-31.57 2.71-40.69 8.14-9.12 5.42-13.68 12.75-13.68 21.98 0 8.54 3.06 15.06 9.18 19.57 6.12 4.5 15.41 6.75 27.88 6.75h66.13v44.32h-69.6c-14.08 0-24.93 2.42-32.55 7.27-7.61 4.85-11.43 11.89-11.43 21.12 0 9.69 4.9 17.25 14.72 22.68 9.81 5.42 24.87 8.13 45.18 8.13 14.31 0 28.62-1.9 42.94-5.71 14.31-3.81 26.43-9.18 36.36-16.1l17.31 42.25zm208.84-121.19h-95.57v141.97h-53.67v-186.29h149.24v44.32zm242.08-44.32v186.29h-51.59v-21.81c-6.93 8.08-15.41 14.2-25.45 18.35-10.04 4.16-20.72 6.23-32.03 6.23-24.47 0-43.92-7.04-58.35-21.12-14.42-14.08-21.64-34.98-21.64-62.68v-105.26h54.02v97.3c0 30.01 12.58 45.01 37.74 45.01 13.16 0 23.66-4.21 31.51-12.64s11.77-20.95 11.77-37.57v-92.1h54.02zm266.32 0v186.29h-54.02v-141.97h-77.57v141.97h-53.67v-186.29h185.25zm319.64-2.77c23.32 0 41.84 6.87 55.58 20.6 13.74 13.74 20.6 34.34 20.6 61.81v106.65h-54.02v-98.34c0-14.77-3.06-25.8-9.18-33.07s-14.83-10.91-26.14-10.91c-12.7 0-22.74 4.1-30.13 12.29-7.38 8.2-11.08 20.37-11.08 36.53v93.49h-54.02v-98.34c0-29.31-11.77-43.97-35.32-43.97-12.46 0-22.39 4.1-29.78 12.29-7.39 8.2-11.08 20.37-11.08 36.53v93.49h-54.02v-186.29h51.59v21.47c6.93-7.85 15.41-13.85 25.45-18.01s21.07-6.23 33.07-6.23c13.16 0 25.05 2.59 35.66 7.79 10.62 5.19 19.16 12.75 25.63 22.68 7.61-9.69 17.25-17.2 28.91-22.51s24.41-7.96 38.26-7.96zm245.54 148.54c9.69 0 18.3-1.44 25.8-4.33s14.49-7.44 20.95-13.68l28.74 31.16c-17.54 20.09-43.16 30.13-76.87 30.13-21.01 0-39.59-4.1-55.75-12.29-16.16-8.2-28.62-19.57-37.4-34.11-8.77-14.54-13.16-31.05-13.16-49.52 0-18.24 4.33-34.68 12.98-49.34 8.66-14.66 20.55-26.08 35.66-34.28 15.12-8.19 32.15-12.29 51.08-12.29 17.78 0 33.93 3.75 48.48 11.25 14.54 7.5 26.14 18.35 34.8 32.55s12.98 30.99 12.98 50.38l-137.81 26.66c3.93 9.24 10.1 16.16 18.53 20.78s18.76 6.93 30.99 6.93zm-7.96-107.69c-13.62 0-24.7 4.39-33.24 13.16s-13.04 20.89-13.51 36.36l90.72-17.66c-2.54-9.7-7.73-17.43-15.58-23.2s-17.31-8.66-28.39-8.66z" fill="#fff"></path>

                    </svg>

                    <ul>

                        <li><FaShoppingCart *onClick*={() => setCartOpen(cartOpen = !cartOpen)} *className*={`shop-cart-button ${cartOpen && 'active'}`} /></li>

                        <li><a *href*='#about'>О&nbsp;нас</a></li>

                        <li><a *href*='#excursions'>Экскурсии</a></li>

                        <li><a *href*='#contacts'>Контакты</a></li>

                    </ul>

                    {

                        cartOpen && (

                            <div *className*="shop-cart">

                                {*props*.orders.length > 0 ?

                                    showOrders(*props*) : showNothing()}

                            </div>

                        )

                    }

                </div>

            </div>

        </header>

    )

* **About:** Создаем компонент About.js в директории src/components с следующим кодом:

import React from 'react'

export default function About() {

    return (

        <section *className*='about container' *id*='about'>

            <h2>О&nbsp;нас</h2>

            <p>

                Приветствуем вас на&nbsp;странице компании Лампатур! На&nbsp;нас лежит ответственная миссия&nbsp;&mdash; организовать для вас интересные, безопасные и&nbsp;лучшие экскурсии в&nbsp;Египте. Основные направления&nbsp;&mdash; курорты Хургада и&nbsp;Шарм-Эль-Шейх.

            </p>

            <h3>Почему к&nbsp;нам постоянно обращаются, а&nbsp;вам стоит доверить организацию отдыха Лампатур:</h3>

            <ul>

                <li>проверяем все экскурсии лично&nbsp;&mdash; проходим по&nbsp;маршрутам, общаемся с&nbsp;гидами и&nbsp;водителями;</li>

                <br></br>

                <li>проверяем все экскурсии лично&nbsp;&mdash; проходим по&nbsp;маршрутам, общаемся с&nbsp;гидами и&nbsp;водителями;</li>

                <br></br>

                <li>реагируем на&nbsp;жалобы и&nbsp;недовольства туристов&nbsp;&mdash; исправляем недочеты, чтобы делать экскурсии лучше (мы&nbsp;понимаем, что не&nbsp;всегда всё может пройти идеально, но&nbsp;держим руку на&nbsp;пульсе).</li>

            </ul>

            <p>

                У нас есть варианты отдыха для всех туристов: можете походить по Каирскому музею, поплавать в Красном море или расслабиться в спа. Все программы поездок — увлекательные и разнообразные. Еще можно отправиться в исторические места Египта, чтобы посмотреть Пирамиды, увидеть знаменитого Сфинкса и побывать в долине мертвых в Луксоре.

                <br></br>

                <br></br>

                Знаем, что дайвинг у многих туристов — основная причина приехать сюда, где больше не встретить столь разнообразный и красивый подводный мир. Даже если не умеете плавать, вы всё равно можете погрузиться с опытным инструктором под воду и увидеть собственными глазами рифы и проплывающих рядом рыб.

                <br></br>

                <br></br>

                Мы работаем с увлеченными гидами, которые любят свою страну за историю, культуру и традиции. Многие из них имеют регалии историков, поэтому заскучать вам не удастся. Еще местные гиды помогают нам придумывать маршруты, чтобы за короткое время показать вам как можно больше интересного.

                <br></br>

                <br></br>

                А представьте, что всё внимание гида сосредоточено только на вас: он подсказывает, как сфотографироваться, где сделать удачные кадры и отвечает на все каверзные вопросы, связанные с поездкой и достопримечательностями. Всё это доступно во время индивидуальных поездок, которые мы организуем для вас.

            </p>

            <h3>С точки зрения технических нюансов тоже всё в порядке</h3>

            <ul>

                <li>без проблем принимаем оплату российскими картами;</li>

                <br></br>

                <li>поможем дополнить ваш отдых экскурсиями, когда вы уже находитесь в Египте (учтите, что предложения от отелей будут дороже в 2-3 раза);</li>

                <br></br>

                <li>оплата происходит в день экскурсии (исключения — индивидуальные поездки, где 30% стоимости будет в качестве предоплаты);</li>

                <br></br>

                <li>приобретаемая страховка действует на всей территории Египта (не слушайте местных зазывал, которые могут убеждать в обратном);</li>

                <br></br>

                <li>вопросы, связанные с нахождением в Египте (вызов такси, обмен валюты, поиск недвижимости для проживания), помогаем решить в нашем Телеграм-чате.</li>

            </ul>

            <p>Ждем ваших вопросов и заранее желаем отличного отдыха в Египте, ваш ЛампаТур!</p>

        </section>

    )

}

* **Categories:** Создаем компонент Categories.js в директории src/components с следующим кодом:

import React, { Component } from 'react'

export class Categories extends Component {

    constructor(*props*) {

*super*(*props*)

        this.state = {

          categories: [

            {

                key: 'all',

                name: 'Все'

            },

            {

                key: 'Исторические',

                name: 'Исторические'

            },

            {

                key: 'Морские',

                name: 'Морские'

            },

            {

                key: 'Развлечения',

                name: 'Развлечения'

            }

          ]

        }

    }

  render() {

    return (

      <div *className*='categories container' *id*='excursions'>

        <h2>Экскурсии</h2>

        {this.state.categories.map(*el* => (

            <div *key*={*el*.key} *onClick*={() => this.props.chooseCategory(*el*.key)}>{*el*.name}</div>

        ))}

      </div>

    )

  }

}

export default Categories

* **Excursions:** Создаем компонент Excursions.js в директории src/components с следующим кодом:

import React, { Component } from 'react'

import Excursion from './Excursion'

export class Excursions extends Component {

    render() {

        return (

            <main *className*='container' >

                {this.props.items.map(*el* => (

                    <Excursion *onShowItem*={this.props.onShowItem} *key*={*el*.id} *item*={*el*} *onAdd*={this.props.onAdd}/>

                ))}

            </main>

        )

    }

}

export default Excursions

* **Excursion:** Создаем компонент Excursion.js в директории src/components с следующим кодом:

import React, { Component } from 'react'

import { MdOutlineWatchLater } from "react-icons/md";

import { CiCalendar } from "react-icons/ci";

export class Excursion extends Component {

    render() {

        return (

            <div *className*='excursion'>

                <img *src*={"./img/" + this.props.item.img} *onClick*={() => this.props.onShowItem(this.props.item)} />

                <h4>{this.props.item.category}</h4>

                <h3>{this.props.item.title}</h3>

                <p><CiCalendar/> {this.props.item.date}</p>

                <p><MdOutlineWatchLater/> {this.props.item.time}</p>

                <b>{this.props.item.price}$</b>

                <div *className*='add-to-cart'*onClick*={() => this.props.onAdd(this.props.item)}>+</div>

            </div>

        )

    }

}

export default Excursion

* **Footer:** Создаем компонент Footer.js в директории src/components с следующим кодом:

import React from 'react'

export default function Footer() {

  return (

    <footer *className*='container'>

2024 &laquo;LampaTour&raquo;. Все права защищены&nbsp;&copy;.

Разработка сайта&nbsp;&mdash; <a *href*='http://anton-purgin-portfolio.ru/' *target*='blank'>Anton Purgin</a>.

    </footer>

  )

}

* **Contacts:** Создаем компонент Contacts.js в директории src/components с следующим кодом:

import React from 'react'

export default function Contacts() {

  return (

    <div *className*='contacts container' *id*='contacts'>

    <div *className*='logo'>

        <svg *height*="99px" *viewBox*="0 0 4338.3 1883.5" *xmlns*="http://www.w3.org/2000/svg">

            <path *d*="m2573.8 826.55-116.8-124.1-153.2 161.76c-26.87 5.3-52.78 14.56-77.29 28.11-34.51 19.08-62.92 45.1-84.82 77.11-21.84-31.63-50.03-57.4-84.11-76.53-41.27-23.15-86.55-34.13-133.84-34.13-19.94 0-39.39 1.86-58.07 5.82h-33.3l504.69-532.64 402.67 422.23c-2.91 0.71-5.81 1.45-8.71 2.2-46.34 12.14-90.62 30.39-131.76 54.96l-25.46 15.21z" *fill*="#FFED00"></path>

            <path *d*="m3523.1 864.59v85.07c-20.78-22.97-45.46-41.96-73.57-56.85-44.59-23.62-93.34-34.04-143.62-34.04-40.57 0-80.13 6.79-117.36 21.73l16.53-36.37-53.75-32.56c-41.07-24.88-85.44-43.12-131.87-55.21-47.3-12.32-95.82-18.28-144.69-18.28-35.98 0-71.77 3.3-107.02 10.02l-147.95-208.42 486.14-539.68 810.65 864.59h-393.51z" *fill*="#FFED00"></path>

            <path d="m-0 817.53h117.78v412.96h255.19v95.97h-372.97v-508.93zm825.2 117.78v391.15h-108.33v-45.08c-28.11 33.93-68.83 50.9-122.14 50.9-36.84 0-70.16-8.24-99.97-24.72s-53.2-39.99-70.16-70.53c-16.96-30.53-25.45-65.92-25.45-106.15s8.48-75.61 25.45-106.15c16.96-30.53 40.35-54.04 70.16-70.53 29.81-16.48 63.13-24.72 99.97-24.72 49.92 0 88.94 15.76 117.05 47.26v-41.44h113.42zm-210.84 303.9c28.6 0 52.34-9.81 71.25-29.45 18.9-19.63 28.35-45.92 28.35-78.88s-9.45-59.26-28.35-78.88c-18.91-19.63-42.65-29.45-71.25-29.45-29.08 0-53.07 9.81-71.98 29.45-18.9 19.63-28.35 45.92-28.35 78.88s9.45 59.26 28.35 78.88c18.91 19.63 42.9 29.45 71.98 29.45zm816.47-309.72c48.95 0 87.85 14.42 116.69 43.26s43.26 72.1 43.26 129.78v223.93h-113.42v-206.48c0-31.02-6.42-54.16-19.27-69.43-12.84-15.27-31.14-22.9-54.89-22.9-26.65 0-47.74 8.61-63.25 25.81-15.51 17.21-23.26 42.78-23.26 76.71v196.3h-113.42v-206.48c0-61.56-24.72-92.33-74.16-92.33-26.17 0-47.01 8.61-62.53 25.81-15.51 17.21-23.26 42.78-23.26 76.71v196.3h-113.42v-391.15h108.33v45.08c14.54-16.48 32.35-29.08 53.44-37.81 21.08-8.73 44.23-13.09 69.43-13.09 27.63 0 52.59 5.46 74.89 16.36s40.23 26.78 53.8 47.62c16-20.36 36.23-36.11 60.71-47.26s51.26-16.73 80.33-16.73zm492.94 0c36.35 0 69.43 8.36 99.24 25.08 29.81 16.73 53.2 40.23 70.16 70.53s25.45 65.56 25.45 105.79-8.48 75.49-25.45 105.79c-16.96 30.29-40.35 53.8-70.16 70.53-29.81 16.72-62.89 25.08-99.24 25.08-49.92 0-89.18-15.76-117.78-47.26v182.49h-113.42v-532.2h108.33v45.08c28.11-33.93 69.07-50.9 122.87-50.9zm-19.63 309.72c29.08 0 52.95-9.81 71.61-29.45 18.66-19.63 27.99-45.92 27.99-78.88s-9.33-59.26-27.99-78.88c-18.66-19.63-42.53-29.45-71.61-29.45s-52.95 9.81-71.61 29.45c-18.66 19.63-28 45.92-28 78.88s9.33 59.26 28 78.88c18.66 19.63 42.53 29.45 71.61 29.45zm687.06-303.9v391.15h-108.33v-45.08c-28.11 33.93-68.83 50.9-122.14 50.9-36.84 0-70.16-8.24-99.97-24.72s-53.2-39.99-70.16-70.53c-16.96-30.53-25.45-65.92-25.45-106.15s8.48-75.61 25.45-106.15c16.96-30.53 40.35-54.04 70.16-70.53 29.81-16.48 63.13-24.72 99.97-24.72 49.92 0 88.94 15.76 117.05 47.26v-41.44h113.42zm-210.84 303.9c28.6 0 52.34-9.81 71.25-29.45 18.9-19.63 28.35-45.92 28.35-78.88s-9.45-59.26-28.35-78.88c-18.91-19.63-42.65-29.45-71.25-29.45-29.08 0-53.07 9.81-71.98 29.45-18.9 19.63-28.35 45.92-28.35 78.88s9.45 59.26 28.35 78.88c18.91 19.63 42.9 29.45 71.98 29.45zm694.33-279.18c-43.62-25.21-90.4-41.44-140.32-48.71v415.14h-117.78v-415.14c-49.92 7.27-96.94 23.5-141.05 48.71l-39.99-87.97c34.9-20.84 72.7-36.6 113.42-47.26s82.64-16 125.78-16c43.62 0 85.91 5.33 126.87 16 40.95 10.66 78.64 26.42 113.05 47.26l-39.99 87.97zm231.2 372.25c-41.2 0-78.16-8.61-110.88-25.81-32.72-17.21-58.28-41.08-76.7-71.61s-27.63-65.19-27.63-103.97 9.21-73.43 27.63-103.97c18.42-30.53 43.99-54.41 76.7-71.61 32.72-17.21 69.68-25.81 110.88-25.81s78.03 8.61 110.51 25.81c32.47 17.2 57.92 41.08 76.34 71.61s27.63 65.19 27.63 103.97-9.21 73.43-27.63 103.97c-18.42 30.53-43.87 54.41-76.34 71.61-32.48 17.21-69.31 25.81-110.51 25.81zm0-93.06c29.08 0 52.95-9.81 71.61-29.45 18.66-19.63 27.99-45.92 27.99-78.88s-9.33-59.26-27.99-78.88c-18.66-19.63-42.53-29.45-71.61-29.45s-53.07 9.81-71.98 29.45c-18.9 19.63-28.35 45.92-28.35 78.88s9.45 59.26 28.35 78.88c18.91 19.63 42.9 29.45 71.98 29.45zm684.15-303.9v391.15h-107.6v-46.53c-15.03 16.96-32.96 29.93-53.8 38.9-20.84 8.96-43.38 13.45-67.62 13.45-51.37 0-92.09-14.79-122.14-44.35-30.05-29.57-45.07-73.43-45.07-131.59v-221.02h113.42v204.3c0 63.01 26.42 94.52 79.25 94.52 27.14 0 48.95-8.84 65.43-26.54 16.48-17.69 24.72-43.99 24.72-78.88v-193.39h113.42zm213.75 51.62c13.57-18.9 31.87-33.2 54.89-42.89s49.56-14.54 79.61-14.54v104.7c-12.6-0.97-21.08-1.46-25.45-1.46-32.47 0-57.92 9.09-76.34 27.26-18.42 18.18-27.63 45.44-27.63 81.79v184.67h-113.42v-391.15h108.33v51.62z" fill="#fff"></path>

            <path d="m106.25 1567c24.01 0 45.53 5.37 64.58 16.1 19.04 10.74 33.99 25.63 44.84 44.67s16.28 40.57 16.28 64.58-5.43 45.53-16.28 64.58c-10.85 19.04-25.8 33.93-44.84 44.67-19.04 10.73-40.57 16.1-64.58 16.1-21.46 0-40.8-3.69-58-11.08s-31.57-18.12-43.11-32.2l35.66-33.24c17.08 19.16 37.97 28.74 62.67 28.74 17.78 0 32.84-5.19 45.19-15.58s20.37-24.36 24.06-41.9h-98.34v-42.93h97.64c-3.92-16.62-12-29.9-24.24-39.82-12.23-9.93-27.01-14.89-44.32-14.89-24.7 0-45.59 9.58-62.67 28.74l-35.66-33.24c11.54-14.08 25.91-24.81 43.11-32.2s36.53-11.08 58-11.08zm277.06 175.9h-28.05v70.64h-54.02v-186.29h54.02v72.37h29.09l50.55-72.37h57.48l-65.1 89.34 69.6 96.95h-63.71l-49.86-70.64zm252.47 73.41c-19.85 0-37.68-4.1-53.5-12.29-15.81-8.2-28.16-19.57-37.05-34.11s-13.33-31.05-13.33-49.52 4.45-34.97 13.33-49.51c8.89-14.54 21.24-25.91 37.05-34.11 15.81-8.19 33.64-12.29 53.5-12.29 19.62 0 36.76 4.1 51.42 12.29 14.66 8.2 25.34 19.91 32.03 35.15l-41.9 22.51c-9.69-17.08-23.66-25.62-41.9-25.62-14.08 0-25.74 4.62-34.97 13.85-9.24 9.23-13.85 21.81-13.85 37.74s4.62 28.51 13.85 37.74c9.23 9.24 20.89 13.85 34.97 13.85 18.47 0 32.43-8.54 41.9-25.63l41.9 22.86c-6.7 14.77-17.37 26.31-32.03 34.63-14.66 8.31-31.8 12.46-51.42 12.46zm227.89-73.41h-28.05v70.64h-54.02v-186.29h54.02v72.37h29.09l50.55-72.37h57.48l-65.1 89.34 69.6 96.95h-63.71l-49.86-70.64zm352.89-115.65v155.13c0 34.16-8.66 59.56-25.97 76.18s-42.47 24.93-75.49 24.93c-17.54 0-34.11-2.2-49.69-6.58-15.58-4.39-28.68-10.73-39.3-19.04l21.46-38.78c7.85 6.47 17.43 11.55 28.74 15.24s22.74 5.54 34.28 5.54c18.01 0 31.16-4.15 39.48-12.46 8.31-8.31 12.46-21.01 12.46-38.09v-6.23c-6.93 6.93-15.12 12.23-24.58 15.93-9.46 3.69-19.62 5.54-30.47 5.54-24.47 0-43.86-6.98-58.18-20.95-14.31-13.97-21.46-34.92-21.46-62.85v-93.49h54.02v85.53c0 30.24 12.58 45.36 37.74 45.36 12.93 0 23.32-4.21 31.16-12.64 7.85-8.43 11.77-20.95 11.77-37.57v-80.68h54.02zm176.29-2.77c19.85 0 37.17 3.92 51.94 11.77s26.2 19.04 34.28 33.59c8.08 14.54 12.12 31.51 12.12 50.9 0 18.93-3.81 35.6-11.43 50.03s-18.35 25.63-32.2 33.59-29.78 11.94-47.79 11.94c-23.77 0-42.36-7.5-55.75-22.51v86.92h-54.02v-159.63c0-19.16 4.27-36.01 12.81-50.55s20.55-25.86 36.01-33.93c15.47-8.08 33.47-12.12 54.02-12.12zm-2.42 147.51c14.08 0 25.51-4.73 34.28-14.2 8.77-9.46 13.16-21.82 13.16-37.05 0-15.47-4.39-27.99-13.16-37.57s-20.2-14.37-34.28-14.37-25.45 4.73-34.11 14.2c-8.66 9.46-12.99 21.93-12.99 37.4s4.27 27.94 12.81 37.4 19.97 14.2 34.28 14.2zm256.97 44.32c-19.85 0-37.68-4.1-53.5-12.29-15.81-8.2-28.16-19.57-37.05-34.11s-13.33-31.05-13.33-49.52 4.45-34.97 13.33-49.51c8.89-14.54 21.24-25.91 37.05-34.11 15.81-8.19 33.64-12.29 53.5-12.29 19.62 0 36.76 4.1 51.42 12.29 14.66 8.2 25.34 19.91 32.03 35.15l-41.9 22.51c-9.69-17.08-23.66-25.62-41.9-25.62-14.08 0-25.74 4.62-34.97 13.85-9.24 9.23-13.85 21.81-13.85 37.74s4.62 28.51 13.85 37.74c9.23 9.24 20.89 13.85 34.97 13.85 18.47 0 32.43-8.54 41.9-25.63l41.9 22.86c-6.7 14.77-17.37 26.31-32.03 34.63-14.66 8.31-31.8 12.46-51.42 12.46zm329.69-189.06v186.29h-51.59v-21.81c-6.93 8.08-15.41 14.2-25.45 18.35-10.04 4.16-20.72 6.23-32.03 6.23-24.47 0-43.92-7.04-58.35-21.12-14.42-14.08-21.64-34.98-21.64-62.68v-105.26h54.02v97.3c0 30.01 12.58 45.01 37.74 45.01 13.16 0 23.66-4.21 31.51-12.64s11.77-20.95 11.77-37.57v-92.1h54.02zm264.93 0v186.29h-51.59v-21.81c-6.93 8.08-15.41 14.2-25.45 18.35-10.04 4.16-20.72 6.23-32.03 6.23-24.47 0-43.92-7.04-58.35-21.12-14.42-14.08-21.64-34.98-21.64-62.68v-105.26h54.02v97.3c0 30.01 12.58 45.01 37.74 45.01 13.16 0 23.66-4.21 31.51-12.64s11.77-20.95 11.77-37.57v-92.1h54.02zm321.37 88.99c21.93 7.38 32.9 22.16 32.9 44.32 0 16.85-6.35 29.89-19.04 39.13s-31.62 13.85-56.78 13.85h-99.38v-186.29h95.22c23.32 0 41.55 4.21 54.71 12.64s19.74 20.26 19.74 35.49c0 9.24-2.37 17.37-7.1 24.41s-11.48 12.52-20.26 16.45zm-92.8-12.47h40.86c18.93 0 28.39-6.7 28.39-20.08 0-13.16-9.46-19.74-28.39-19.74h-40.86v39.82zm45.36 73.06c19.85 0 29.78-6.81 29.78-20.43 0-7.15-2.31-12.41-6.93-15.76s-11.89-5.02-21.81-5.02h-46.4v41.21h45.36zm448.8 15.93c-11.54 7.85-26.03 13.97-43.46 18.35-17.43 4.39-35.95 6.58-55.57 6.58-24.47 0-45.25-3.06-62.33-9.18s-30.01-14.66-38.78-25.63-13.16-23.49-13.16-37.57c0-12.93 3.58-24.35 10.73-34.28s16.97-17.31 29.43-22.16c-9.46-5.08-16.79-12-21.99-20.78-5.19-8.77-7.79-18.58-7.79-29.43 0-13.16 4.04-25.16 12.12-36.01s20.31-19.5 36.7-25.97c16.39-6.46 36.47-9.69 60.25-9.69 15.7 0 31.04 1.56 46.05 4.68 15 3.12 28.05 7.33 39.13 12.64l-15.58 42.94c-21.24-9.93-43.75-14.89-67.52-14.89-18.01 0-31.57 2.71-40.69 8.14-9.12 5.42-13.68 12.75-13.68 21.98 0 8.54 3.06 15.06 9.18 19.57 6.12 4.5 15.41 6.75 27.88 6.75h66.13v44.32h-69.6c-14.08 0-24.93 2.42-32.55 7.27-7.61 4.85-11.43 11.89-11.43 21.12 0 9.69 4.9 17.25 14.72 22.68 9.81 5.42 24.87 8.13 45.18 8.13 14.31 0 28.62-1.9 42.94-5.71 14.31-3.81 26.43-9.18 36.36-16.1l17.31 42.25zm208.84-121.19h-95.57v141.97h-53.67v-186.29h149.24v44.32zm242.08-44.32v186.29h-51.59v-21.81c-6.93 8.08-15.41 14.2-25.45 18.35-10.04 4.16-20.72 6.23-32.03 6.23-24.47 0-43.92-7.04-58.35-21.12-14.42-14.08-21.64-34.98-21.64-62.68v-105.26h54.02v97.3c0 30.01 12.58 45.01 37.74 45.01 13.16 0 23.66-4.21 31.51-12.64s11.77-20.95 11.77-37.57v-92.1h54.02zm266.32 0v186.29h-54.02v-141.97h-77.57v141.97h-53.67v-186.29h185.25zm319.64-2.77c23.32 0 41.84 6.87 55.58 20.6 13.74 13.74 20.6 34.34 20.6 61.81v106.65h-54.02v-98.34c0-14.77-3.06-25.8-9.18-33.07s-14.83-10.91-26.14-10.91c-12.7 0-22.74 4.1-30.13 12.29-7.38 8.2-11.08 20.37-11.08 36.53v93.49h-54.02v-98.34c0-29.31-11.77-43.97-35.32-43.97-12.46 0-22.39 4.1-29.78 12.29-7.39 8.2-11.08 20.37-11.08 36.53v93.49h-54.02v-186.29h51.59v21.47c6.93-7.85 15.41-13.85 25.45-18.01s21.07-6.23 33.07-6.23c13.16 0 25.05 2.59 35.66 7.79 10.62 5.19 19.16 12.75 25.63 22.68 7.61-9.69 17.25-17.2 28.91-22.51s24.41-7.96 38.26-7.96zm245.54 148.54c9.69 0 18.3-1.44 25.8-4.33s14.49-7.44 20.95-13.68l28.74 31.16c-17.54 20.09-43.16 30.13-76.87 30.13-21.01 0-39.59-4.1-55.75-12.29-16.16-8.2-28.62-19.57-37.4-34.11-8.77-14.54-13.16-31.05-13.16-49.52 0-18.24 4.33-34.68 12.98-49.34 8.66-14.66 20.55-26.08 35.66-34.28 15.12-8.19 32.15-12.29 51.08-12.29 17.78 0 33.93 3.75 48.48 11.25 14.54 7.5 26.14 18.35 34.8 32.55s12.98 30.99 12.98 50.38l-137.81 26.66c3.93 9.24 10.1 16.16 18.53 20.78s18.76 6.93 30.99 6.93zm-7.96-107.69c-13.62 0-24.7 4.39-33.24 13.16s-13.04 20.89-13.51 36.36l90.72-17.66c-2.54-9.7-7.73-17.43-15.58-23.2s-17.31-8.66-28.39-8.66z" fill="#fff"></path>

        </svg>

    </div>

    <a *href*='lampatour@gmail.com'>lampatour@gmail.com</a>

    <b>+20-15-0105-5199</b>

    <p>круглосуточно</p>

</div>

  )

}

* **Order:** Создаем компонент Order.js в директории src/components с следующим кодом:

import React, { Component } from 'react'

import { FaTrash } from 'react-icons/fa'

export class Order extends Component {

    render() {

        return (

            <div *className*='item'>

                <img *src*={"./img/" + this.props.item.img} />

                <h3>{this.props.item.title}</h3>

                <b>{this.props.item.price}$</b>

                <FaTrash *className*='delete-icon' *onClick*={() => this.props.onDelete(this.props.item.id)} />

            </div>

        )

    }

}

export default Order

* **ShowFullItem:** Создаем компонент ShowFullItem.js в директории src/components с следующим кодом:

import React, { Component } from 'react'

import { MdOutlineWatchLater } from "react-icons/md";

import { CiCalendar } from "react-icons/ci";

export class ShowFullItem extends Component {

    render() {

        return (

            <div *className*='full-item'>

                <div>

                    <img *src*={"./img/" + this.props.item.img} *onClick*={() => this.props.onShowItem(this.props.item)} />

                    <h3>{this.props.item.title}</h3>

                    <p><CiCalendar /> {this.props.item.date}</p>

                    <p><MdOutlineWatchLater /> {this.props.item.time}</p>

                    <b>{this.props.item.price}$</b>

                    <div *className*='add-to-cart' *onClick*={() => this.props.onAdd(this.props.item)}>+</div>

                </div>

            </div>

        )

    }

}

export default ShowFullItem

**2.3. Создание главной страницы**

* **App.js:** Изменяем код компонента App.js в директории src следующим образом:

import Footer from "./components/Footer";

import Header from "./components/Header";

import Excursions from "./components/Excursions";

import Contacts from "./components/Contacts";

import Categories from "./components/Categories";

import ShowFullItem from "./components/ShowFullItem";

class App extends React.Component {

  constructor(*props*) {

*super*(*props*)

    this.state = {

      orders: [],

      currehtItems: [],

      items: [

        {

          id: 1,

          title: 'VIP Луксор с Долиной Царей',

          img: '1.jpg',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Исторические',

          price: '1500'

        },

        {

          id: 2,

          title: 'Абу Дабаб — пляж гигантских черепах',

          img: '2.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Морские',

          price: '1500'

        },

        {

          id: 3,

          title: 'Асуан + Абу - Симбел',

          img: '3.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Исторические',

          price: '1500'

        },

        {

          id: 4,

          title: 'Батискаф + снорклинг в Хургаде',

          img: '4.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Исторические',

          price: '1500'

        },

        {

          id: 5,

          title: 'Бухта дельфинов',

          img: '5.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Морские',

          price: '1500'

        },

        {

          id: 6,

          title: 'Гранд аквариум и зоопарк в Хургаде',

          img: '6.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Развлечения',

          price: '1500'

        },

        {

          id: 7,

          title: 'Дайв - Сафари',

          img: '7.jpeg',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Морские',

          price: '1500'

        },

        {

          id: 8,

          title: 'Дельфинарий',

          img: '8.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Развлечения',

          price: '1500'

        },

        {

          id: 9,

          title: 'Джип - сафари на LAND CRUISER',

          img: '9.webp',

          date: 'ежедневно',

          time: '18 часов',

          category: 'Развлечения',

          price: '1500'

        }

      ],

      ShowFullItem: false,

      fullItem: {}

    }

    this.state.currehtItems = this.state.items

    this.addToOrder = this.addToOrder.bind(this)

    this.deleteOrder = this.deleteOrder.bind(this)

    this.chooseCategory = this.chooseCategory.bind(this)

    this.onShowItem = this.onShowItem.bind(this)

  }

  render() {

    return (

      <div *className*="wrapper">

        <Header *orders*={this.state.orders} *onDelete*={this.deleteOrder} />

        <About />

        <Categories *chooseCategory*={this.chooseCategory} />

        <Excursions *onShowItem*={this.onShowItem} *items*={this.state.currehtItems} *onAdd*={this.addToOrder} />

        {this.state.ShowFullItem && <ShowFullItem *onAdd*={this.addToOrder} *onShowItem*={this.onShowItem} *item*={this.state.fullItem} />}

        <Contacts />

        <Footer />

      </div>

    );

  }

  onShowItem(*item*) {

    this.setState({fullItem: *item*})

    this.setState({ShowFullItem: !this.state.ShowFullItem})

  }

  chooseCategory(*category*) {

    if (*category* === 'all') {

      this.setState({currehtItems: this.state.items})

      return

    }

    this.setState({

      currehtItems: this.state.items.filter(*el* => *el*.category === *category*)

    })

  }

  deleteOrder(*id*) {

    this.setState({ orders: this.state.orders.filter(*el* => *el*.id !== *id*) })

  }

  addToOrder(*item*) {

    let isInArray = false;

    this.state.orders.forEach(*el* => {

      if (*el*.id === *item*.id)

        isInArray = true

    })

    if (!isInArray)

      this.setState({ orders: [...this.state.orders, *item*] })

  }

}

export default App;

**2.4. Стиль сайта**

* **index.css:**

@import url("https://fonts.googleapis.com/css2?family=Montserrat:wght@100..900&display=swap");

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    background: #fff;

    color: #222;

    font-family: "Montserrat", sans-serif;

    font-weight: 300;

}

h2,

h3 {

    font-weight: 600;

}

h2 {

    font-size: 32px;

}

h3 {

    font-size: 25px;

}

.container {

    max-width: 1280px;

    margin: 0 auto;

}

header {

    width: 100%;

    height: 100vh;

    background: url("./img/hurgada.jpg") no-repeat;

    background-size: cover;

    background-position: center;

}

header .logo {

    padding: 50px;

    display: flex;

    align-items: center;

    justify-content: space-between;

    position: relative;

}

header .shop-cart-button {

    cursor: pointer;

    transition: color, transform 500ms ease;

    color: #fff;

}

header .shop-cart-button:hover,

header .shop-cart-button.active {

    color: #ffe500;

    transform: scale(1.5);

}

header .shop-cart {

    position: absolute;

    top: 120px;

    right: 0;

    width: 600px;

    background: #fafafa;

    -webkit-box-shadow: 4px 5px 15px -7px #606060;

    box-shadow: 4px 5px 15px -7px #606060;

    padding: 20px;

    padding-bottom: 0;

}

header .shop-cart .empty h2 {

    text-align: center;

    margin-bottom: 20px;

    font-size: 20px;

}

header .shop-cart .item {

    width: 100%;

    float: left;

    margin-bottom: 20px;

}

header .shop-cart .item img {

    width: 70px;

    float: left;

    margin-right: 20px;

}

header .shop-cart .item h3 {

    font-size: 20px;

    margin-bottom: 10px;

}

header .shop-cart .item b {

    color: #797979;

    font-weight: 600;

    padding-top: 10px;

}

header .shop-cart .item .delete-icon {

    color: #ca5252;

    float: right;

    position: relative;

    top: -15px;

    cursor: pointer;

    transition: color, transform 500ms ease;

}

header .shop-cart .item .delete-icon:hover {

    color: #d83030;

    transform: scale(1.5);

}

header .shop-cart .summa {

    float: left;

    width: 100%;

    font-weight: 600;

    font-size: 20px;

    margin-bottom: 20px;

}

header ul {

    display: flex;

    gap: 30px;

}

header ul li {

    list-style-type: none;

}

header ul li a {

    text-decoration: none;

    color: #fff;

    transition: color 500ms ease;

}

header ul li a:hover {

    color: #ffe500;

}

.about {

    margin-top: 50px;

    margin-bottom: 50px;

    padding: 0 20px;

}

.about h2 {

    margin-bottom: 30px;

}

.about h3 {

    margin-top: 30px;

    margin-bottom: 20px;

}

.about ul {

    margin-left: 20px;

    margin-bottom: 20px;

}

.categories {

    padding: 0 20px;

}

.categories h2 {

    margin-bottom: 30px;

}

.categories div {

    display: inline-block;

    max-width: max-content;

    border: 1px solid #ffe500;

    padding: 5px;

    background: #ffe500;

    border-radius: 5px;

    color: #333;

    margin-bottom: 25px;

    margin-right: 20px;

    font-weight: 600;

    cursor: pointer;

    transition: all 500ms ease;

}

.categories div:hover {

    transform: scale(1.1);

}

.full-item {

    position: fixed;

    top: 0;

    right: 0;

    left: 0;

    bottom: 0;

    background: rgba(0, 0, 0, 0.8);

    z-index: 9999;

    overflow: auto;

}

.full-item div {

    position: relative;

    width: 600px;

    margin: 10% auto;

    background: #47b2ff;

    border-radius: 10px;

    overflow: hidden;

    padding-bottom: 20px;

}

main {

    padding: 0 20px;

    width: 100%;

    display: flex;

    flex-wrap: wrap;

    justify-content: space-between;

}

main .excursion {

    position: relative;

    width: 30%;

    margin-bottom: 50px;

    background: #47b2ff;

    border-radius: 10px;

    overflow: hidden;

    padding-bottom: 20px;

}

main .excursion img,

.full-item img {

    width: 100%;

    border-radius: 10px 10px 0 0;

    transition: transform 500ms ease;

    cursor: pointer;

}

.full-item img {

    padding: 30px;

}

main .excursion img:hover,

.full-item img:hover {

    transform: scale(1.05);

}

main h4,

.full-item h4,

main h3,

.full-item h3,

main p,

.full-item p,

main b,

.full-item b {

    margin: 10px 20px;

    color: #fff;

}

main h3,

.full-item h3 {

    height: 70px;

}

main p,

.full-item p {

    margin-bottom: 50px;

}

main b,

.full-item b,

main h4,

.full-item h4 {

    max-width: max-content;

    border: 1px solid #ffe500;

    padding: 5px;

    background: #ffe500;

    border-radius: 5px;

    color: #333;

}

main b,

.full-item b {

    margin: 0;

    position: absolute;

    bottom: 20px;

    left: 20px;

}

main h4,

.full-item h4 {

    position: absolute;

    margin: 0;

    top: 10px;

    left: 10px;

}

main .add-to-cart,

.full-item .add-to-cart {

    position: absolute;

    right: 20px;

    bottom: 20px;

    background: #ffe500;

    width: 35px;

    height: 35px;

    text-align: center;

    line-height: 35px;

    border-radius: 50%;

    cursor: pointer;

    font-weight: 600;

    transition: transform 500ms ease;

}

.full-item .add-to-cart {

    bottom: -40px;

}

main .add-to-cart:hover,

.full-item .add-to-cart:hover {

    transform: scale(1.5) translateY(-5px);

}

.contacts {

    display: flex;

    align-items: center;

    justify-content: space-around;

    padding: 50px;

    background: #47b2ff;

    border-radius: 10px;

    font-weight: 600;

    color: #fff;

}

.contacts a {

    color: #fff;

    text-decoration: none;

}

.contacts b {

    color: #ffe500;

}

footer {

    padding: 30px 20px;

    text-align: center;

}

footer a {

    text-decoration: none;

    color: #222;

    transition: color 500ms ease;

}

footer a:hover {

    color: #ffe500;

}

@media (max-width: 850px) {

    .contacts {

        flex-direction: column;

        gap: 30px;

    }

}

@media (max-width: 730px) {

    header .logo, .contacts {

        flex-direction: column;

        gap: 30px;

    }

    main {

        justify-content: space-around;

    }

    main .excursion {

        width: 47%;

    }

}

@media (max-width: 420px) {

    main {

        justify-content: center;

    }

    main .excursion {

        width: 96%;

    }

}

**2.5. Передача данных:**

* + В App.js передаем excursions в Excursions компонент:

<Excursions excursions={this.state.excursions} />

* **Отображение данных:**
  + В Excursions.js используем excursions для rendering:

export class Excursions extends Component {

    render() {

        return (

            {

*this*.*props*.*items*.*map*(*el* => (

                    <Excursion *key*={*el*.id} *item*={*el*} />

                ))

            }

              </main >

            );

    }

}

**2.6. Реализация функциональности**

* **Добавление в корзину:**
  + В Excursion.js добавляем onClick handler для button:

<button onClick={() => this.props.onAdd(this.props.item)}>Добавить в корзину</button>

* + В App.js добавьте state с orders и addToOrder function:

class App extends React.Component {

    constructor(*props*) {

*super*(*props*);

        this.state = {

            orders: [],

*// ...*

        };

*// ...*

    }

    addToOrder(*item*) {

*// ... (код для добавления товара в заказы)*

    }

*// ...*

}

* **Отображение корзины:**
  + В Header.js добавляем state с cartOpen и onClick handler для cart icon:

let [cartOpen, setCartOpen] = useState(false);

<li><FaShoppingCart *onClick*={() => setCartOpen(cartOpen = !cartOpen)} *className*={`shop-cart-button ${cartOpen && 'active'}`} /></li>

* + В Header.js добавляем conditional rendering для cart:

{

    cartOpen && (

        <div *className*="shop-cart">

            {props.orders.length > 0 ?

                showOrders(props) : showNothing()}

        </div>

    )

}

* **Удаление из корзины:**
  + В Order.js добавляем onClick handler для delete icon:

<FaTrash *className*='delete-icon' *onClick*={() => this.props.onDelete(this.props.item.id)} />

* + В App.js добавляем deleteOrder function:

class App extends React.Component {

*// ...*

    deleteOrder(*id*) {

*// ... (код для удаления товара из заказа)*

    }

*// ...*

}

**2.7. Дополнительная функциональность**

* **Фильтрация:**
  + В Categories.js добавляем onClick handler для categories:

{

    this.state.categories.map(*el* => (

        <div *key*={*el*.key} *onClick*={() => this.props.chooseCategory(*el*.key)}>{*el*.name}</div>

    ))

}

* + В App.js добавляем chooseCategory function:

class App extends React.Component {

*// …*

    chooseCategory(*category*) {

*// … (код для фильтрации экскурсий)*

    }

*// …*

}

* **Отображение полной информации:**
  + В Excursion.js добавляем onClick handler для img:

<img *src*={"./img/" + this.props.item.img} *onClick*={() => this.props.onShowItem(this.props.item)} />

* + В App.js добавляем state с fullItem и onShowItem function:

class App extends React.Component {

*// ...*

    onShowItem(*item*) {

*// ... (код для отображения полной информации об экскурсии)*

    }

*// ...*

}

* + В App.js добавляем ShowFullItem компонент:

{this.state.ShowFullItem && <ShowFullItem *onAdd*={this.addToOrder} *onShowItem*={this.onShowItem} *item*={this.state.fullItem} />}

# На этом фронтенд часть сайта закончена.

# Заключение

Разработка сайта интернет-магазина экскурсий на React.js — это увлекательный и результативный процесс, который позволяет создавать современные, функциональные и привлекательные веб-приложения. В процессе работы мы изучили основы React.js, реализовали основные компоненты сайта, интегрировали данные, добавили функциональность для добавления экскурсий в корзину и их удаления, а также реализовали фильтрацию по категориям.

Благодаря компонентной архитектуре React, мы смогли организовать код проекта четко и структурировано. Виртуальный DOM позволил нам создать сайт, который работает быстро и эффективно.

В дальнейшем, мы можем расширить функциональность сайта, добавив:

* **Интеграцию с платежной системой:** Для удобного приема платежей от клиентов.
* **Админ-панель:** Для управления товарами, заказами и пользователями.
* **Мобильную версию:** Для удобного доступа к сайту с мобильных устройств.
* **Интеграцию с социальными сетями:** Для упрощения входа и взаимодействия с пользователями.

Опыт разработки этого проекта показал, что React.js — это мощный инструмент для создания современных веб-приложений. Он позволяет легко создавать интерактивные и динамичные сайты, которые соответствуют современным стандартам качества.

# Список используемой литературы

* [https://ru.react.js.org](https://ru.react.js.org/)
* <https://nodejs.org/en>
* [https://react.dev](https://react.dev/)

# Приложения

* <https://github.com/Drfinik/my-diplom>
* <https://my-diplom-80793q2vk-antonios-projects-c6ddee39.vercel.app>
* my diplom.rar