Geekbrains

Создание frontend части web-приложения с онлайн-занятиями по фитнесу Fit Steps с помощью библиотеки React

Программа:
Разработчик — Frontend разработка (React)
Савкина Мария Александровна

Санкт-Петербург
2024

Содержание

Введение

В современном мире информационные технологии играют значительную роль во многих сферах деятельности. Особенно это касается сферы фитнеса, где онлайн-тренировки становятся всё более популярными. В связи с этим возникает потребность в создании удобной платформы, предоставляющей услуги онлайн-тренировок по фитнесу.

Так как я сама долгое время занималась фитнесом в зале и перешла на домашние тренировки, я хорошо понимаю актуальность подобного сайта в наши дни и какая у него возможна функциональность.

Такое web-приложение должно быть простым в использовании, интуитивно понятным и предоставлять пользователям широкий спектр возможностей, таких как регистрация, авторизация, поиск тренировок, просмотр расписания занятий.

Для создания подобного веб-приложения можно использовать различные технологии и инструменты. Однако для реализации данного дипломного проекта я остановилась на библиотеке React, которая обладает рядом преимуществ и особенностей, делающих ее идеальным выбором для разработки подобных проектов.

Цель данного дипломного проекта — исследовать возможности и особенности библиотеки React, а также разработать и внедрить веб-приложение для портала по оказанию услуг онлайн-тренировок по фитнесу с проектным названием Fit Steps.

Задачи проекта:

1. Изучить особенности создания быстрой и масштабируемой клиентской части веб-приложения на React.
2. Ознакомиться с основными принципами создания клиентской части веб-приложения на React.
3. Научиться работать со структурой сайта и разрабатывать компоненты на основе этой структуры с помощью библиотеки React
4. Разработать пример веб-приложения Fit Steps, демонстрирующий применение изученных технологий.

Данная дипломная работа подготовлена в рамках итоговой аттестации по технической специализации “Разработчик — Frontend разработка (React)”, поэтому будет затронута только эта часть создания сайта для онлайн-тренировок по фитнесу.

В данном проекте я выполняла роль frontend-разработчика на React и верстальщика на HTML и CSS. Для реализации всего проекта и охвата всех этапов его создания необходимо также участие: проджект-менеджера, маркетолога, аналитика, дизайнера, backend-разработчика и тестировщика.

Для реализации проекта я использовала следующие инструменты и технологии: Visual Studio Code, React, React Router DOM, Webpack, Git, HTML, CSS, SASS, Node.js.

Глава 1. Основы frontend-разработки

1.1 Что такое web-приложение

Одним из ключевых элементов информационных технологий являются веб-приложения. В данной главе будет рассмотрено, что такое веб-приложение, его основные характеристики и отличия от статических сайтов.

Веб-приложение — это функциональная интерактивная компьютерная программа, предназначенная для выполнения определенных задач и доступа к данным через интернет. Оно работает на основе клиент-серверной архитектуры и использует протокол передачи данных HTTP или HTTPS.

Основные характеристики веб-приложений:

- Интерактивность: веб-приложения предоставляют возможность взаимодействия с пользователем, например, заполнение форм, отправка сообщений и получение уведомлений.
- Адаптивность: веб-приложения могут адаптироваться к различным устройствам и разрешениям экрана, обеспечивая удобство использования на разных устройствах.
- Безопасность: веб-приложения используют механизмы аутентификации и авторизации для защиты данных пользователей и предотвращения несанкционированного доступа.

Отличие веб-приложений от статических сайтов заключается в их функциональности и степени взаимодействия с пользователем. Простые сайты предназначены для хранения и отображения информации, в то время как веб-приложения обеспечивают взаимодействие с пользователем и выполнение определённых задач.

Таким образом, веб-приложения представляют собой мощный инструмент для решения различных задач и обеспечения удобного взаимодействия с пользователями.

1.2. Чем занимается frontend-разработчик (HTML, CSS, JavaScript)

Frontend-разработчик создает внешний вид и функциональность веб-страниц, делая их удобными и привлекательными для пользователей. В этой главе будет уделено внимание основным задачам и инструментам, используемым frontend-разработчиками.

1. Верстка и структура страницы

Frontend-разработчик создает структуру и макет страницы с помощью языка гипертекстовой разметки HTML (HyperText Markup Language). Он определяет расположение элементов, таких как заголовки, абзацы, списки и изображения, следуя дизайнерскому макету.

2. Стилизация и оформление

Frontend-разработчик применяет стили CSS (Cascading Style Sheets) для оформления элементов страницы, устанавливает цвета, шрифты и визуальные эффекты. Он также может использовать препроцессоры CSS, такие как Sass и Less, для упрощения и ускорения процесса разработки.

3. Добавление интерактивности

Frontend-разработчик добавляет интерактивность на страницу с помощью языка программирования JavaScript. Он создает пользовательские интерфейсы, обрабатывает события, добавляет анимации и валидацию форм.

4. Взаимодействие с сервером

Frontend-разработчик может работать с API (интерфейс программирования приложений) для получения и отправки данных с сервера. Он использует библиотеки и фреймворки, такие как React, Angular или Vue.js, для упрощения разработки сложных интерфейсов.

5. Тестирование и оптимизация

Frontend-разработчик проводит тестирование своего кода, чтобы убедиться, что все функции работают корректно и без ошибок. Он также оптимизирует производительность страницы, минимизируя размер файлов и ускоряя загрузку контента.

Таким образом, frontend-разработчик играет ключевую роль в создании удобных и функциональных веб-страниц, используя комбинацию HTML, CSS и JavaScript. Он должен обладать навыками верстки, стилизации, программирования и тестирования, чтобы успешно выполнять свои задачи.

1.3 Обзор библиотеки React и её применение в разработке сайтов

React — это библиотека для создания пользовательских интерфейсов, разработанная компанией Facebook. Она использует декларативный подход к созданию компонентов, что позволяет разработчикам сосредоточиться на представлении данных, а не на управлении DOM. React работает на основе принципа виртуального DOM, который позволяет быстро обновлять пользовательский интерфейс без полной перезагрузки страницы.

Преимущества React включают:

- Простота и скорость разработки: React предлагает простой и понятный синтаксис, что облегчает разработку и поддержку кода.
- Высокая производительность: виртуальный DOM обеспечивает быстрое обновление пользовательского интерфейса без перезагрузки страницы.
- Тестирование: React поддерживает тестирование компонентов с помощью библиотеки Jest и библиотеки тестирования React.

Компоненты в React представляют собой строительные блоки пользовательского интерфейса. Они могут быть созданы с использованием функционального или классового подхода. Функциональный подход основан на использовании функций, которые принимают функцию рендеринга в качестве аргумента. Классовый подход использует классы Ecmascript 6 для определения компонентов.

Для управления состоянием приложения в React используется концепция «управляемых компонентов». Управляемые компоненты содержат состояние, которое может изменяться в зависимости от действий пользователя или других событий. Состояние хранится в объекте контекста, который передается через props и state.

React широко используется в разработке сайтов и приложений, особенно крупными компаниями и организациями. Вот некоторые примеры применения React:

- Netflix: стриминговый сервис использует React для создания своего пользовательского интерфейса.
- Новостные ленты социальных сетей: крупные социальные сети, такие как Twitter и Facebook, применяют React для реализации своих новостных лент.
- Яндекс: российский поисковый гигант использует React для создания интерфейсов своих сервисов.
- Uber: компания по вызову такси использует React для разработки своего мобильного приложения.
- Airbnb: популярный сайт для аренды жилья применяет React для создания своего пользовательского интерфейса.

React является мощным инструментом для создания интерактивных и функциональных пользовательских интерфейсов. Его популярность и широкое применение в разработке сайтов и приложений делают его отличным инструментом для веб-разработчиков.

Глава 2. Проектирование frontend-части web-приложения Fit Steps

2.1. Функциональные возможности web-приложения

Функциональные возможности сайта определяются заказчиком, утверждаются в техническом задании и являются основой для дальнейшей разработки любого web-приложения.

Основные функциональные возможности web-приложения Fit Steps (платформа для занятий фитнесом, йогой, гимнастикой дома):

- посещение онлайн-групповых занятий по фитнесу;
- проведение вебинаров по фитнесу и питанию;
- чтение полезных статей;
- регистрация в качестве пользователя с выбором вариантов платной подписки;
- регистрация в качестве тренера для возможности проведения тренировок и публикации собственных статей.

Функционал web-приложения Fit Steps (платформа для занятий фитнесом, йогой, гимнастикой дома) можно разделить на три составляющие: функционал пользователя, тренера, а также функции администрирования.

Функционал пользователя

- Регистрация на площадке с ролью “Пользователь” (регистрация по email);
- Аутентификация на площадке по логину и паролю;
- Профиль с возможностью редактирования данных (ФИО, email, телефон, информация о себе и прочие данные, которые могут появится позже);
- Добавление в избранное онлайн-занятий и вебинаров, чтобы их было проще отслеживать для посещения;
- Удаление из избранного онлайн-занятий и вебинаров;
- Просмотр избранных занятий;
- Приобретение подписки;
- Просмотр занятий на странице с занятием, если приобретена подписка;
- Просмотр вебинаров бесплатно.

Функционал исполнителя

- Регистрация на площадке с ролью “Тренер” (с регистрацией по email);
- Аутентификация на площадке по логину и паролю;
- Профиль с возможностью редактирования данных (ФИО, email, телефон, информация о себе, а также прочие данные, которые могут появится с расширением приложения);
- Создание онлайн-тренировки или вебинара;
- Удаление онлайн-тренировки или вебинара.

Функционал администратора

- Аутентификация на площадке.
- Панель управления пользователями с указанием их ID, email, имя, роль. Создание новых администраторов.
- Профиль выбранного пользователя с подробной информацией и возможностью редактирования полей.
- Панель управления занятиями: ID, email заказчика, имя специалиста, статус.
- Карточка занятия с подробной информацией (описание, файлы, визуализация) с возможностью редактирования полей.

Именно на основе данных возможностей будет разрабатываться логика функций и компонентов.

В данном дипломном проекте будут реализованы функционал пользователя и функционал тренера. Реализация данного функционала предполагает возможность в дальнейшем его расширения для обеспечения большего удобства пользователя (фильтры, сортировка), а также больших возможностей для него (добавление комментариев, выставление оценки тренеру и т.п.).

2.2. Структура web-приложения

Структура web-приложения также определяется и утверждается заказчиком и служит каркасом для проекта. На ее основе можно составить структуру из компонентов для проекта React.

При построении структуры сайта FitSteps необходимо учитывать следующие принципы:

- Простота и ясность. Структура сайта должна быть простой и понятной для пользователей. Это облегчает навигацию и повышает удобство использования сайта.
- Логичность. Элементы структуры сайта должны быть логически связаны друг с другом. Это помогает пользователям быстро находить нужную информацию и облегчает понимание структуры сайта.
- Адаптивность. Структура сайта должна быть адаптирована для мобильных устройств. Это обеспечивает удобство использования сайта на разных устройствах и разрешениях экрана.
- SEO-оптимизация. Структура сайта должна быть оптимизирована для поисковых систем. Это позволяет улучшить видимость сайта в результатах поиска и привлечь больше целевых посетителей.

Основные элементы структуры платформы FitSteps:

Главная страница. Это стартовая страница, на которой пользователи видят основные разделы и подразделы сайта. Главная страница должна быть информативной, привлекательной и удобной для навигации.

Разделы. Разделами сайта Fit Steps являются онлайн-тренировки, вебинары и статьи. Это три основных раздела, в которых сосредоточен главный функционал сайта.

Подразделы. Подразделы внутри разделов детализируют информацию и предоставляют пользователям более углубленный контент. В разделе, посвященном онлайн-занятиям могут быть подразделы для разных видов тренировок (силовые, кардио, йога и т. д.).

Карточки постов. Карточки постов содержат краткое описание, изображение или видео, заголовок, автора или тренера и дату публикации. Они используются для отображения постов на главной странице и в разделах сайта.

Меню навигации. Меню навигации обеспечивает удобный переход между разделами и подразделами сайта. Оно может быть реализовано с помощью меню, ссылок или иконок.

Детальная структура web-платформы находится в Приложении 1.

2.3. Структура компонентов

Структура компонентов в проекте React является важной составляющей для обеспечения эффективной работы и организации кода. Использование функциональных компонентов вместо компонентов класса позволяет добиться лучшей производительности благодаря отсутствию дополнительного слоя абстракции и методов жизненного цикла. По этой причине веб-приложение FitSteps будет реализовано с помощью функциональных компонентов.

Для удобства навигации и структурирования кода компоненты разделены на папки, соответствующие функциям или страницам в веб-приложении. Сначала компоненты делятся на две основные папки: components и pages. Папка components содержит многократно используемые компоненты, такие как header и footer, а папка pages включает компоненты, специфичные для каждой страницы приложения, например, home и about.

Также, чтобы избежать дублирования кода, будет создана папка modules, в которой будут храниться функции, которые используются в различных компонентах.

Детальная организация структуры компонентов находится в Приложении 2.

Глава 3. Подготовка проекта React и используемые дополнительные инструменты и библиотеки

3.1. Создание проекта

Прежде всего, необходимо создать новый проект React. Чтобы это сделать, необходим такой инструмент, как Node.js, благодаря которому можно легко создать React-проект, с помощью специальной команды:

```
npx create-react-app fitsteps
```

После запуска в командной строке этой инструкции, необходимо дождаться загрузки пакета и всех его зависимостей, которые будут установлены в папку fitsteps.

После успешной установки проекта, необходимо удалить файлы, которые устанавливаются с пакетом для демонстрации демо-версии проекта. Эти файлы в проекте Fit Steps использоваться не будут.

Следующим шагом на этапе подготовки проекта будет установка дополнительных пакетов с библиотеками, которые понадобятся при разработке веб-приложения. Для этих целей я установлю react-router-dom, axios, node-sass.

3.2. React Router

Для создания и обработки маршрутов, необходимых для переключения между страницами сайта и создания удобной навигации будет использован react-router-dom.

React Router — это библиотека для маршрутизации в React. Она позволяет переходить от одного представления к другому в приложении на основе действий пользователя, таких как клики на ссылки или кнопки. React Router DOM — это модуль узла, предназначенный для маршрутизации в веб-приложениях на базе React. Он позволяет создавать маршруты для одностраничного приложения React.

Компоненты React Router включают Browser Router, Route, Switch и Link. BrowserRouter — это реализация маршрутизатора, использующая API HTML5 для синхронизации пользовательского интерфейса с URL-адресами. Route — это условный компонент, который отображает компонент на основе определенного URL-адреса. Link

используется для создания ссылок на разные маршруты и реализации навигации по приложению. Switch используется для визуализации только первого маршрута, который соответствует текущему местоположению.

Для установки библиотеки необходимо в терминале запустить следующую команду: **npm install react-router-dom .**

3.3. Axios

Для обработки запросов к серверу, таких как получение данных, их изменение, добавление и удаление будет использоваться axios.

Axios — это библиотека JavaScript, используемая для выполнения HTTP-запросов из браузера или среды выполнения JavaScript, например Node.js. Она обеспечивает простой и удобный способ взаимодействия с веб-серверами и получения данных с удаленных ресурсов.

Функции Axios включают поддержку множества платформ, промисов, обработку запросов и ответов, установку заголовков, манипуляцию с данными и обработку ошибок.

Библиотека предоставляет следующие возможности:

- Отправка XMLHttpRequest из браузера.
- Отправка HTTP-запросов из Node.js.
- Поддержка Promise API.
- Перехват запросов и ответов.
- Преобразование данных запросов и ответов.
- Отмена/прерывание запроса.
- Автоматический разбор/парсинг данных в формате JSON.
- Автоматическая защита от XSRF.

Для установки библиотеки необходимо в терминале запустить следующую команду: `npm install axios .`

3.4. Node Sass

Чтобы работа со стилями была наиболее удобной и простой будет использована библиотека node-sass, позволяющая работать с препроцессорами.

Node Sass — это инструмент, который позволяет использовать язык Sass в проектах на языке JavaScript, написанных с использованием Node.js. Он работает на основе библиотеки LibSass, которая является реализацией языка Sass на C++.

Node Sass выполняет следующие функции:

- Компиляция Sass в CSS: преобразует код Sass в CSS, что позволяет использовать стили в веб-приложениях.
- Использование переменных: позволяет передавать значения переменных между файлами Sass.
- Вложенность и миксины: обеспечивают возможность организации кода и повторного использования стилей.
- Поддержка различных операционных систем: работает на Windows, macOS и Linux.

После установки данной библиотеки, при запуске проекта файлы Sass будут компилироваться в CSS.

Для установки библиотеки необходимо в терминале запустить следующую команду: `npm install node-sass .`

3.5. Git

Для успешной работы с проектом также необходимо контролировать все изменения, чтобы было возможно откатиться к любой версии проекта. Для этих целей будет использоваться Git.

Git — это распределенная система контроля версий, которая позволяет отслеживать изменения в файлах, хранить их версии и быстро возвращаться к предыдущим состояниям. Git работает на основе принципа снимка, сохраняя полный образ файла в момент сохранения.

Git состоит из трех основных компонентов:

- Git-репозиторий — это место, где хранятся все файлы проекта и история их изменений.
- Git-клиент — это программа, которая позволяет работать с репозиторием, например, добавлять, изменять и удалять файлы.
- Git-сервер — это служба, которая предоставляет доступ к репозиторию другим пользователям и разработчикам.

Git использует модель ветвления для работы с несколькими версиями кода. Ветви позволяют разработчикам работать над разными задачами одновременно, не мешая друг другу. Когда работа над веткой завершена, её можно объединить с основной веткой или опубликовать на GitHub.

GitHub — это веб-сервис, который предоставляет хостинг для Git-репозиториев и инструменты для совместной разработки. Он позволяет разработчикам делиться своими проектами с коллегами, получать отзывы и вносить изменения в реальном времени.

После того, как все необходимые библиотеки установлены, нужно инициализировать проект и разместить его на git-сервисе.

Для этих целей на платформе GitHub.ru я создала публичный репозиторий https://github.com/mashhroma/fitSteps и связала его с проектом на локальном ПК.

Теперь проект настроен и можно приступать с созданию компонентов.

3.6. Хуки UseState и Use Context

Прежде чем начать разрабатывать компоненты приложения, которые отвечают за визуализацию и взаимодействие с пользователем, необходимо создать компонент-обертку, который будет передавать нуждающимся в этом компонентам данные, приходящие из серверной части. В данном веб-приложении это реализовано с помощью встроенных хуков Use State и Use Context.

Use State — это функциональный хук в React, который позволяет создавать и обновлять состояние компонента. Он принимает начальное значение состояния в качестве аргумента и возвращает массив из двух элементов: текущее состояние и функция для его обновления.

Use Context — это хук в React, который позволяет передавать данные между компонентами без использования props. Он состоит из двух частей: Context API и useContext().

Context API создаёт контекст, который содержит данные и предоставляет доступ к ним для дочерних компонентов. Для создания контекста используется встроенная фабричная функция createContext().

useContext() — это хук, который возвращает значение контекста. Он гарантирует повторный рендеринг компонента при изменении значения контекста.

Чтобы использовать Context API, необходимо:

1. Создать контекст с помощью createContext().
2. Предоставить контекст дочерним компонентам с помощью компонента Context.Provider.
3. Использовать хук useContext() для получения значения контекста и повторного рендеринга компонента при изменении значения.

Глава 4. Реализация frontend-части web-приложения Fit Steps с использованием React

4.1. Компонент ContextProvider и используемые данные

Мой самый первый компонент создаем с помощью хуков Use State и Use Context. Контекст создаем для следующих данных, которые нужно будет передавать дочерним элементам:

- Данные об онлайн-занятиях, статьях, вебинарах - DataContext;
- Данные о типах тренировок - TypesContext;
- Данные о профилях (пользователи и тренеры)- ProfilesContext;
- Данные о залогиненном пользователе - ActiveUserContext;
- Данные о залогиненном тренере - ActiveCoachContext;

Внутри компонента ContextProvider с помощью хука Use State создаем пары переменных и функций для их изменения, которые будут передаваться через контекст другим компонентам.

Данные об онлайн-занятиях, вебинарах, статьях, профилях хранятся на сервере. Для целей данного дипломного проекта все эти данные были размещены на платформе Mockapi.io, которая предоставляет ограниченную возможность бесплатно размещать JSON-файлы, к которым можно обращаться с помощью стандартных серверных запросов (получить, отправить, редактировать или удалить объект по id).

Чтобы получить эти данные и сохранить в переменные, используется get-метод из библиотеки axios. Т.к. запрос идет с сервера, необходимо дождаться получения данных, а затем сохранять их в переменные. Таким образом, запрос является асинхронным, поэтому его необходимо обернуть в try-catch, а также указать async-await в функции, которая получает и обрабатывает данные с сервера.

Помимо этого, запрос к серверу необходимо делать внутри хука Use Effect. Это необходимо для того, чтобы обращение к серверу произошло лишь один раз - при первой загрузке веб-приложения и не происходило постоянно.

Данные о типах онлайн-занятий получаем из локального JSON-файла, так как они не будут меняться.

Данные о залогиненном пользователе и залогиненном тренере будут приходить из LocalStorage.

В итоге компонент ContextProvider возвращает “матрешку” из провайдеров ранее созданных контекстов, в которые передаются созданные переменные. Внутри матрешки будут все остальные компоненты веб-приложения.

Код компонента представлен в Приложении 3.

4.2. Компонент Header

Одним из важнейших компонентов в веб-приложении является компонент Header. Данный компонент будет присутствовать на каждой странице сайта и обладает существенным влиянием на его восприятие и именно на нем присутствуют главный элемент дизайна - логотип, от которого зависит вся стилистика веб-приложения.

Компонент <Header /> состоит из следующих элементов:

- Логотип FitSteps, который является ссылкой на главную страницу веб-приложения. Для реализации функционала ссылки будет использоваться компонент Link из библиотеки react-router-dom.
- Компонент <Navigation /> , который содержит навигацию по разделам веб-приложения.
- Компонент <Authorization />, который отображает текущего активного пользователя, либо кнопки авторизации и регистрации.

Для того, чтобы компонент функционировал, в нем указаны следующие зависимости:

- Link из библиотеки 'react-router-dom',
- Navigation,
- Authorization.

Код компонента представлен в Приложении 4.

4.3. Компонент Navigation

Компонент Navigation представляет собой меню из трех основный разделов сайта: онлайн-занятия, вебинары, статьи. Он размещен в компоненте <Header />. При нажатии на пункт меню осуществляется переход на соответствующий раздел веб-приложения при помощи компонента Link. Это единственная зависимость, которая используется в этом компоненте.

Код компонента представлен в Приложении 5.

4.4. Компонент Authorization

Еще один компонент, который размещен в компоненте <Header />. С помощью пропсов, в данный компонент передаются функции, которые при клике на соответствующие кнопки из этого компонента открывают либо форму аутентификации, либо форму регистрации.

В компоненте также есть функция, которая удаляет активного пользователя. Чтобы эта функция работала, в компонент передаются активный пользователь с помощью Use State и Use Context. Функция активируется кнопкой “выйти”. Она устанавливает значение активного пользователя как null и удаляет его из localStorage.

Компонент возвращает два варианта разметки, в зависимости от того, есть ли залогиненный пользователь. Если нет, что возвращаются две кнопки: “войти” и “зарегистрироваться”. Если пользователь залогинен - то возвращается ссылка на страницу пользователя и кнопка “выйти”.

Код компонента представлен в Приложении 6.

4.5. Компонент LoginForm

Компонент LoginForm с помощью стилей оформлен как модальное окно. Оно появляется при клике на кнопку “войти” из компонента <Authorization /> или из компонента <CoachAboutPage />.

В данном компоненте есть функция, которая проверяет введенные данные из формы. Для этого из контекста в компонент передаются все профили. Функция ищет совпадения по email. Если профиль нашелся и пароль введен корректно, то устанавливается или

залогиненный пользователь в соответствующую переменную, также переданную из контекста, или залогиненный тренер. Это зависит от того, из какого компонента была открыта форма.

Возвращает данный компонент разметку с формой аутентификации.

Код компонента представлен в Приложении 7.

4.6. Компонент Registration Form

Компонент RegistrationForm имеет точно такое же по оформлению модальное окно, как и в компоненте <LoginForm />. В нем также содержаться из контекста данные о всех профилях, но используются они в функции, которая проверяет существует ли с введенным email уже профиль в веб-приложении. Форма также может открываться из компонентов <Authorization /> и <CoachAboutPage />, поэтому перед проверкой email осуществляется сортировка по нужной роли профиля.

Вначале запускается функция, которая проверяет правильно ли заполнена форма. Затем специальная функция проверяет зарегистрирован ли пользователь, проверяя есть ли введенный email в массиве с профилями. Если все хорошо, то новый профиль добавляется на сервер с помощью post-запроса, а также в контексте устанавливается новое значение массива из профилей.

Возвращает данный компонент разметку с формой регистрации.

Код компонента представлен в Приложении 8.

4.7. Компонент Submenu

Помимо основной навигации по трем разделам, для удобства пользователей на каждой странице веб-приложения имеется дополнительная навигация с подразделами онлайн-занятий, чтобы можно было легко зайти именно в тот подраздел, который интересует больше всего. Для этого создан компонент Submenu, который принимает данные о типах онлайн-занятий из контекста и возвращает их, обернутыми в компонент Link.

Код компонента представлен в Приложении 9.

4.8. Компонент Subheader

Компонент Subheader располагается под <Submenu />. Этот компонент отображает название открытой в данный момент страницы. Чтобы сформировать название страницы, в компоненте есть функция. Благодаря use Location компонент получает из url путь к странице, из которого, благодаря методу split(), получаем первое значение после слеша. Функция сравнивает его из спомощью switch получает название, которое функция возвращает.

Код компонента представлен в Приложении 10.

4.9. Компонент Footer

Компонент Footer также отображается на всех страницах сайта. Он располагается снизу и содержит ссылки на все основные страницы сайта, созданные с помощью компонента Link из библиотеки ‘react-router-dom’.

Код компонента представлен в Приложении 11.

4.10. Компонент Breadcrumbs

Компонент Breadcrumbs отображается на страницах разделов онлайн-занятий, вебинаров и статей. Он представляет собой навигацию по подразделам в виде меню “хлебные крошки”.

Для реализации этого функционала, также как и в компоненте Subheader, вначале с помощью use Location нужно получить URL текущей страницы, а затем с помощью метода split() делим этот URL на части, из которых получаем массив. Затем компонент каждой части массива присваивает заголовок, который будет отображаться в меню в виде ссылки.

В итоге компонент возвращает навигацию, в которой есть ссылка на основной раздел, подраздел и саму статью или занятие, в зависимости от того, какая открыта текущая страница.

Код компонента представлен в Приложении 12.

4.11. Компонент Home

Компонент Home является компонентом, который отображает главную страницу сайта. Этот компонент состоит из пяти дочерних компонентов, которые формируют эту страницу:

- <BannerAbout /> возвращает слайдер с рекламными фото;
- <SubscribeHome /> возвращает разметку со слоганом и ссылкой на страницу с подпиской;
- <Top3Articles /> возвращает подзаголовок, 3 различные статьи, а также дает возможность перейти в главный раздел “Статьи”. Этот компонент использует для отображения элементов другой компонент, который возвращает разметку с карточкой статьи <ArticlePreview />, о котором пойдет речь позже.
- <Top5Streams /> также возвращает подзаголовок, 5 различных вебинаров, а также ссылку на главный раздел “Вебинары”. Для отображения элементов использует компонент <ArticleStream />, о котором также речь пойдет позже.
- <Top5Workouts /> работает по тому же принципу: возвращает подзаголовок, 5 различных онлайн-занятий и ссылку на главный раздел “Онлайн-занятия”. Для отображения элементов использует компонент <ArticleWorkout />.

Коды компонентов, которые используются для главной страницы:

- Приложении 13 <Home />;
- Приложении 14 <BannerAbout />;
- Приложении 15 <SubscribeHome />;
- Приложении 16 <Top3Articles />;
- Приложении 17 <Top5Streams />;
- Приложение 18 <Top5Workouts />.

4.12. Компонент Workouts

Компонент Workouts является одним из важнейших компонентов сайта, так как он отображает основной раздел веб-приложения - “Онлайн-занятия”. В этом компоненте есть также компонент <Breadcrumbs />, принцип работы которого был описан ранее.

Таким образом, компонент <Workouts /> возвращает меню с “хлебными крошками” и перечень онлайн-занятий либо по определенному типу, либо всех имеющихся.

Для отображения перечня с онлайн-занятиями используется метод map(), с помощью которого мы проходимся по массиву из всех онлайн-занятий, которые получили из контекста и для каждого занятия отображаем на странице карточку.

Для отображения карточек занятий используется компонент <Workout Preview />. В этот компонент с помощью пропсов передается текущее занятие, полученное благодаря методу map(). Далее из объекта текущего занятия можно легко получить свойства, которые нужно отобразить на карточке: название, расписание и т.п. В объекте указан id тренера, который его ведет. Для отображения на карточке этого тренера, компонент из списка с профилями, который получает из контекста, находит данные нужного тренера. Вся карточка является ссылкой на онлайн-занятие. А имя и фамилия тренера - ссылкой на тренера.

Также у компонента <Workout Preview /> есть функционал, отвечающий за добавление или удаление любимый занятий. Если в данный момент существует залогиненный пользователь - то на карточке отображается сердечко для добавления в избранное. Если занятие добавлено - отображается красное сердечко.

Помимо объекта с текущим занятием, компонент <Workout Preview /> также принимает данные ширины и высоты отображаемого элемента. Это нужно из-за того, что этот компонент может использоваться на разных страницах сайта и на каждой из них он нужен разного размера.

Код компонентов, которые используются для раздела “Онлайн-занятия”:

- Приложение 19 <Workouts />;
- Приложение 20 <Workout Preview />.

4.13. Компонент WorkoutDetails

Компонент WorkoutDetails отображает полную информацию по конкретному онлайн-занятию. Для этого компонент получает из URL благодаря useParams() аргумент, который в роутере прописан для id онлайн-занятия. Компонент берет из контекста данные по онлайн-занятиям и находит в них онлайн-занятие с полученным id и возвращает разметку с информацией о занятии.

При реализации коммерческого проекта на данной странице будет реализован переход на сторонний ресурс, который будет транслировать онлайн-занятие, а также будет возможность загружать на эту страницу видеоплеера с прошедшими занятиями.

Компонент <Breadcrumbs /> также содержится на странице.

Если в данный момент существует залогиненный пользователь - то на странице отображается сердечко для добавления в избранное.

Код компонента представлен в Приложении 21.

4.14. Компоненты Streams, Stream Preview, Stream Details

Компоненты Streams, Stream Preview, Stream Details дают аналогичный функционал, что и соответствующие компоненты для онлайн-занятий. Отличием является то, что в “Вебинарах” нет подразделов, а также нет возможности добавить вебинар в избранное.

Компонент <StreamPreview /> не имеет расписания, всего этого компонент отображает дату и время проведения вебинара.

Код компонентов, которые используются для раздела “Вебинары” и отдельной страницы с вебинаром::

- Приложение №22 <Streams />;
- Приложение №23 <StreamPreview />;
- Приложение №24 <StreamsDetails />.

4.15. Компоненты Articles, Article Preview, Article Details

Компоненты Articles, Article Preview, Article Details, в свою очередь также не отличаются существенно от предыдущих компонентов. “Статьи” также не имеют подразделов и функционала с добавлением в избранное.

Компонент <Article />s выводит статьи не по несколько штук в строке, а по одной.

Компоненты <ArticlePreview />, <ArticleDetails /> вместо тренера указывают автора, и только дату публикации. Основной контент - это текст статьи.

Код компонентов, которые используются для раздела “Статьи” и отдельной страницы со статьей:

- Приложение №25 <Articles />;
- Приложение №26 <ArticlePreview />;
- Приложение №27 <ArticleDetails />.

4.16. Компонент User Account

Компонент User Account представляет собой страницу залогиненного пользователя. Зайти на нее можно из компонента <Authorization />, в котором появляется ссылка на данную страницу, если текущий активный пользователь установлен.

Для работы с такими значениями как имя, фамилия, день рождения, пароль и прочие, в компоненте есть несколько пар с переменными и функциями, использующими хук Use State(), которые по умолчанию принимают в качестве значения соответствующие свойства из объекта активного пользователя. Этот объект компонент получает из контекста. Все эти поля пользователь должен иметь возможность изменить. Именно для этого необходимо их отслеживание и переустановка. Для реализации изменения активного пользователя при изменении какого-либо поля, в компоненте находятся функции, которые реагируют на события onChange и onClick и меняют данные в контексте, localStorage, а также передают изменения на сервер.

Так как идет отслеживание каждого поля в отдельности, для каждого созданы по несколько функций в компоненте. Одна следит за изменением, другая по клику меняет объект, третья запрещает редактировать поле.

Также на данной странице отображаются все онлайн-занятия, которые добавил в избранное текущий пользователь. Для этого сначала создается массив с занятиями с помощью фильтра. Из всех онлайн-занятий добавляются те, у которых id совпадает с id из массива favorite из объекта пользователя. Затем для их отображения используется метод map(), который пробегается по полученному массиву с занятиями и отображает их с помощью уже знакомого компонента <WorkoutPreview />.

Код компонента представлен в Приложении 28.

4.17. Компоненты Coach Account, Coach Public Page

Компонент Coach Account работает аналогично компоненту <UserAccount />, но только из контекста берется объект с активным тренером, если он установлен. Попасть на страницу этого компонента можно по ссылке, которая находится в компоненте <CoachAboutPage />. На этой странице также отображается список занятий, но только тех, которые данный тренер ведет. Для этого создается массив, в который с помощью метода filter() добавляются те онлайн занятия, в которых coachId совпадает с id текущего тренера.

В будущем планируется добавить на эту страницу функционал с возможностью удалять и создавать свои онлайн-занятия.

Компонент CoachPublicPage отображает данные о конкретном тренере. Для этого компонент получает id тренера из URL благодаря useParams(). Компонент берет из контекста массив с профилями и находит в нем тренера с данным id.

Компонент возвращает примерно ту же разметку, что и компонент CoachAccount, кроме приватных данных (электронная почта, день рождения, пароль). Также на данной странице отсутствуют функции, связанные с изменением данных. Ссылка на данный компонент есть в компонентах статей, вебинаров и онлайн-занятий.

Код этих компонентов:

- Приложение №29 <CoachAccount />;
- Приложение №30 <CoachPublicPage />.

4.18. Компонент CoachAboutPage

Компонент Coach About Page возвращает разметку, в которой присутствуют:

- кнопка для открытия формы входа в личный кабинет, либо ссылка на страницу личного кабинета тренера, если он залогинен. Для входа используется описанный ранее компонент <LoginForm />.
- кнопка для открытия формы регистрации из уже знакомого компонента <RegistrationForm />.
- Списки с описанием тренера и преимуществами сотрудничества.

Переход на страницу компонента происходит из нижней части web-приложения, за которую отвечает компонент <Footer />.

Код компонента представлен в Приложении 31.

4.19. Компонент Subscriptions

Компонент Subscription представляет собой список из трех вариантов подписок, которые обернуты в компонент Link. Никакой логики данный компонент не содержит, только ссылки для перехода на страницу оплаты. В эту ссылку передается название подписки, которая соответствует ссылке, на которую нажал пользователь

Код компонента представлен в Приложении 32.

4.20. Компонент PaymentPage

Компонент Payment Page с помощью Use Params() получает из URL, сформированный на странице с подпиской, название подписки. В зависимости от названия, с помощью switch устанавливается информация с названием подписки и суммой, которые выбрал пользователь. Если все верно, пользователь нажимает на кнопку “оплатить”. В дальнейшем по клику на эту кнопку будет реализован переход на сторонний ресурс для оплаты подписки. Переход на эту страницу возможен только из компонента <Subscription />.

Код компонента представлен в Приложении 33.

4.21. Простые текстовые компоненты

В данном веб-приложении также имеются компоненты, которые отвечают за страницы, которые обязательно должны быть на сайте, но, при этом, эти компоненты только возвращают разметку, не имея какого-то дополнительного функционала:

- Компонент About возвращает разметку с информацией о данной платформе. В компоненте также присутствуют ссылки на три основных раздела сайта. Переход на эту страницу осуществляется из компонента <Footer />.
- Компонент Contacts возвращает разметку с контактной информацией, а также форму обратной связи. В будущем в этом компоненте будет реализована логика отправки данных формы на сторонний ресурс. Переход на эту страницу осуществляется из компонента <Footer />.

- Компонент CoachOffer возвращает разметку с договором оферты с тренером. Так как коммерческого использования на данный момент нет - оферта пустая. Переход на эту страницу осуществляется из компонента <Footer /> и из компонента <CoachAboutPage />.
- Компонент UserOffer - возвращает разметку с договором оферты с пользователем. Она также пока не заполнена. Переход на эту страницу осуществляется из компонента <Footer /> и из компонента <PaymentPage />.
- Компонент ConfidentialPolicy - возвращает разметку с политикой конфиденциальности платформы. Переход на эту страницу осуществляется из компонента <Footer />.
- Компонент UserAgreement - возвращает разметку с пользовательским соглашением с пользователями платформы. Переход на эту страницу осуществляется из компонента <Footer />.
- Компонент ErrorPage - необходим для отображения разметки при переходе на несуществующие URL сайта. Он возвращает разметку с информацией, что страница не существует.

Код описанных выше компонентов:

- Приложение №34 <About />;
- Приложение №35 <Contacts />;
- Приложение №36 <CoachOffer />;
- Приложение №37 <UserOffer />;
- Приложение №38 <ConfidentialPolicy />;
- Приложение №39 <UserAgreement />;
- Приложение №40 <ErrorPage />.

4.22. Компонент App

Компонент App консолидирует все компоненты внутри себя и собирает веб-приложение. Все компоненты, которые используются в приложении импортируются в компонент <App /> .

Также Компонент App содержит всю логику маршрутизации между страницами веб-приложения, для чего в него импортируются BrowserRouter как Router, Routes, Route из библиотеки “react-router-dom”.

Возвращает компонент матрешку из всех компонентов. На самом верхнем уровне представлен компонент <ContextProvider />, который передает все контексты во все компоненты веб-приложения.

Далее внутри используется компонент <Router />, внутри которого уже представлены остальные компоненты, которые в себе содержат ссылки. Таким образом, внутри <Router /> вложены компоненты, которые уже возвращают разметку и создают внешний вид приложения. Это компоненты: <Header />, <Submenu />, <Subheader />, <Footer />, <Routes />, <RegistrationForm />, <LoginForm />.

Компонент <Routes /> включает в себя компоненты <Routes />, которые связывают компоненты с тем URL путем, по которому они будут отображаться между компонентами <Subheader /> и <Footer />.

Помимо них, на сайте присутствуют компоненты, которые отображаются только после определенных событий - это формы регистрации <RegistrationForm /> и входа на сайт <LoginForm />. Для контроля за отображением этих компонентов в <App /> присутствуют дополнительные пары переменных и функций из хука Use State(), которые принимают булево значение. В зависимости от него, компонент или отображается, или нет. Для изменения состояния в компоненте есть специальные функции, которые следят за событиями кликов в веб-приложении.

Код компонента представлен в Приложении 41.

4.23. Компонент index

Компонент index финализирует весь проект.

Чтобы применились все стили, в компонент импортируется файл index.scss.

Затем мы находим в файле index.html, который находится в папке public, элемент div с id равным root и присваиваем это значение константе root. После чего вызывается всего лишь один метод у данной константы - render, внутри которого будет находится компонент <App />.

Таким образом происходит рендеринг всего проекта.

Код компонента представлен в Приложении 42.

4.24. Функции

Для того, чтобы исключить ненужное дублирование кода, те функции, которые могут использоваться в различных компонентах были собраны в отдельные файле в папке modules.

Это функции, отвечающие за работу с формами, создание профиля, поиск профиля, поиск занятия или вебинара и другие.

Код данных функций представлен в Приложении 43.

Заключение

Работа над данным дипломным проектом дала возможность, в первую очередь, оценить полученные знания и навыки в ходе обучения и научится их применять на практике.

Важным аспектом при реализации проекта был полученный опыт поиска новой информации. Поиск в интернете ресурсов, которые помогали в написании кода, занял существенную часть времени в подготовке проекта. Поэтому приобретенный навык быстро находить информацию в свободных источниках считаю одним из ключевых.

Помимо этого, в ходе подготовки проекта я научилась формировать структуру проекта и познакомилась с различными методиками по его организации. Поняла, насколько важно, чтобы структура файлов и папок была удобной и понятной.

При написании и тестировании кода я периодически сталкивалась с возникающими ошибками. Этот опыт помог мне научится читать ошибки, распознавать их, чтобы быстрее и продуктивнее решать проблемы, которые к ним приводят.

Для целей этого проекта необходимо было сымитировать работу бекэнд части. Для этих целей я решила опробовать специальный сервер mockAPI.io . Благодаря этому я сумела лучше понять взаимодействие с серверной частью приложения со стороны фронтенда. Также был получен большой опыт взаимодействия с JSON-файлами и объектами.

Разработанная платформа FitSteps является веб-приложением с готовым функционалом, демонстрирующим как именно может работать данный сайт. При наличии интереса к этой платформе - в нее быстро можно встроить работу со сторонними платными ресурсами, необходимыми для того, чтобы возможно было использовать платформу в коммерческих целях.

Функционал веб-приложения FitSteps также может быть существенно расширен. Можно добавить фильтрацию, сортировку, поиск по сайту, рейтинг для занятий, статей, тренеров, возможность оставлять комментарии. Также возможно будет добавить раздел с обсуждениями тренировок. На странице пользователя можно сделать календарь с тренировками, а также добавить больше полей с личной информацией.

Библиотека React позволяет масштабировать и улучшать приложение крайне эффективно, благодаря тому, что проект разделен на отдельные компоненты.

Ну и конечно, в современной жизни очень важно, чтобы популярная платформа имела свое мобильное приложение, чтобы было удобно ею пользоваться с помощью смартфона. Это важное улучшение, которое стоит поставить в приоритет, при планировании расширения проекта.

Таким образом, после создания собственного веб-приложения FitSteps на React я получила важный практический опыт реализации собственного проекта, а также почувствовала большую уверенность в своих силах, навыках и готовности к работе в сфере фронтенд-разработки.

Список литературы

- *Книга: Веру Леа. "Секреты CSS.Идеальные решения ежедневных задач". Москва: Питер, 2016. 336 стр.*
- *Книга: Флэнаган Дэвид. "JavaScript. Полное руководство". Москва: "Диалектика", 2021. 720 стр.*
- *Книга: Хорстман Кэй С. "Современный JavaScript для нетерпеливых". Москва: ДМК Пресс, 2021. 297 стр.*
- *Книга: Чиннатамби Кирупа. "Изучаем React". Москва: Эксмо, 2019. 368 стр.*
- *https://axios-http.com/docs/intro*
- *https://github.com/mockapi-io/docs/wiki*
- *https://learn.javascript.ru/*
- *https://learn.microsoft.com/ru-ru/devops/develop/git/what-is-git*
- *https://react.dev/learn*
- *https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html*
- *https://www.thirdrocktechkno.com/blog/best-react-websites/*

Приложения

Приложение 1. Структура web-приложения Fit Steps

- Основная часть
    - Главная
        - Рекламный баннер (слайдер из 5 фото)
        - О подписке кратко
        - Топ-5 онлайн-занятий
        - Топ-5 вебинаров
        - Топ-3 статей
    - Занятия
        - Хаб с онлайн-тренировками
        - Страница онлайн-занятия
    - Вебинары
        - Хаб с вебинарами
        - Страница вебинара
    - Занятия
        - Хаб со статьями
        - Страница статьи
    - Вход (авторизация)
        - Регистрация
        - Вход в личный кабинет
    - Подвал сайта
        - Контакты
        - О платформе
        - Подписки
        - Информация для пользователей (оферта)
        - Информация для специалистов (оферта)
        - Политика конфиденциальности
        - Пользовательское соглашение
        - Страница о тренере
    - Служебные страницы
        - Страница тренера (публичная)

        - Оплата подписки с выбором типа и срока
        - Страница с ошибкой 404
- Кабинет пользователя
    - О себе
    - Информация о подписке
    - Избранное
- Кабинет тренера
    - О себе
    - Баланс, вывод средств
    - Занятия, которые ведет тренер

Приложение 2. Организации компонентов

```
SRC
│   App.js
│   index.js
│   index.scss
│
├───components
│       ArticlePreview.js
│       Authorization.js
│       Breadcrumbs.js
│       Footer.js
│       Header.js
│       LoginForm.js
│       Navigation.js
│       RegistrationForm.js
│       StreamPreview.js
│       Subheader.js
│       Submenu.js
│       WorkoutPreview.js
│
├───contexts
│       ContextProvider.js
│       data.json
│       types.json
│
├───modules
│       formModules.js
│       workoutsFunctions.js
│
└───pages
    │   About.js
    │   ArticleDetails.js
```

```
│   Articles.js
│   CoachAboutPage.js
│   CoachAccount.js
│   CoachOffer.js
│   CoachPublicPage.js
│   ConfidentialPolicy.js
│   Contacts.js
│   ErrorPage404.js
│   PaymentPage.js
│   Streams.js
│   StreamsDetails.js
│   Subscriptions.js
│   UserAccount.js
│   UserAgreement.js
│   UserOffer.js
│   WorkoutDetails.js
│   Workouts.js
│
└───Home
    BannerAbout.js
    index.js
    SubscribeHome.js
    Top3Articles.js
    Top5Streams.js
    Top5Workouts.js
```

Приложение 3. Компонент <ContextProvider />

```
import { createContext, useEffect, useState } from 'react';
import axios from 'axios';
import types from './types.json';

export const DataContext = createContext();
export const TypesContext = createContext();
export const ProfilesContext = createContext();
export const ActiveUserContext = createContext();
export const ActiveCoachContext = createContext();

export const ContextProvider = ({ children }) => {
    const [dataItems, setDataItems] = useState([]);
    const [workouts, setWorkouts] = useState([]);
    const [streams, setStreams] = useState([]);
    const [articles, setArticles] = useState([]);
    const [profiles, setProfiles] = useState([]);
    const [isLoading, setIsLoading] = useState(true);
                        const      [activeUser,      setActiveUser]      =
useState(JSON.parse(localStorage.getItem('activeUser')));
                       const      [activeCoach,      setActiveCoach]      =
useState(JSON.parse(localStorage.getItem('activeCoach')));

    useEffect(() => {
        async function fetchData() {
            try {
                           const  [dataResponse,  profilesResponse]  =  await
Promise.all([

axios.get('https://66598df5de346625136ceaa4.mockapi.io/data'),

axios.get('https://66598df5de346625136ceaa4.mockapi.io/profiles'),
                ]);

                setProfiles(profilesResponse.data);
                setDataItems(dataResponse.data);

            } catch (error) {
                alert('Ошибка при запросе данных ;(');
                console.error(error);
            }
        }
```

```
        fetchData();
    }, []);

    useEffect(() => {
        if (dataItems.length > 0) {
                setWorkouts(() => dataItems.filter(obj => obj.category ===
'workout'));
                 setStreams(() => dataItems.filter(obj => obj.category ===
'stream'));
                setArticles(() => dataItems.filter(obj => obj.category ===
'article'));
            setIsLoading(false);
        }
    }, [dataItems]);

    const editActiveUser = (newUser) => {
        setActiveUser(activeUser => activeUser = newUser);
    }

    const editActiveCoach = (newCoach) => {
        setActiveCoach(activeCoach => activeCoach = newCoach);
    }

    return (isLoading ? <div>Идет загрузка</div> :
             <DataContext.Provider value={{ dataItems, workouts, streams,
articles }}>
            <TypesContext.Provider value={types}>
                  <ProfilesContext.Provider value={{ profiles, setProfiles
}}>
                        <ActiveUserContext.Provider value={{ activeUser,
setActiveUser, editActiveUser }}>
                       <ActiveCoachContext.Provider value={{ activeCoach,
setActiveCoach, editActiveCoach }}>
                            {children}
                      </ActiveCoachContext.Provider>
                    </ActiveUserContext.Provider>
                </ProfilesContext.Provider>
            </TypesContext.Provider>
        </DataContext.Provider>
    );
};
```

Приложение 4. Компонент <Header />

```
import React from 'react';
import { Link } from 'react-router-dom';
import { Navigation } from './Navigation';
import Authorization from './Authorization';

export default function Header({ toggleUserRegForm, toggleUserLoginForm })
{
    return (
        <header className='Header'>
              <Link to='/'><a><img className='Logo' src='/images/logo.png'
/></a></Link>
            <Navigation />
                      <Authorization  toggleUserRegForm={toggleUserRegForm}
toggleUserLoginForm={toggleUserLoginForm} />
        </header>
    )
}
```

Приложение 5. Компонент <Navigation />

```
import { Link } from 'react-router-dom';

export function Navigation() {
    return (
        <ul className="menu">
            <Link to={`/workouts`}><li
className="menu__link">уроки</li></Link>
            <Link to={`/streams`}><li
className="menu__link">вебинары</li></Link>
            <Link to={`/articles`}><li
className="menu__link">статьи</li></Link>
        </ul>
    );
}
```

Приложение 6. Компонент <Authorization />

```
import { useContext } from "react";
import { ActiveUserContext } from "../contexts/ContextProvider";
import { Link } from "react-router-dom";

export default function Authorization({ toggleUserRegForm,
toggleUserLoginForm }) {
    const { activeUser, setActiveUser } = useContext(ActiveUserContext);

    const removeActiveUser = () => {
        if (activeUser) {
            localStorage.removeItem('activeUser');
            setActiveUser(null);
        }
    }

    return (
        <div>
            {activeUser ?
                <div className="authorization">
                    <Link to={`/users/${activeUser.id}`}>
                        <span>
                            <img src={activeUser.avatar} alt="Profile" />
                                                    <h4>{activeUser.name}
{activeUser.surname}</h4>
                        </span>
                    </Link>
                                    <Link to='/'><button className='Register'
onClick={removeActiveUser}>выйти</button></Link>
                </div> :
                <div className="authorization">
                                                <button  className='Login'
onClick={toggleUserLoginForm}>войти</button>
                                            <button  className='Register'
onClick={toggleUserRegForm}>зарегистрироваться</button>
                </div>}
        </div>
    )
}
```

Приложение 7. Компонент <LoginForm />

```
import { useContext, useState } from "react";
import { ActiveCoachContext, ActiveUserContext, ProfilesContext } from
"../contexts/ContextProvider";
import { findProfile, getProfileDataFromForm, isFormCorrect,
isPasswordCorrect } from "../modules/formModules";

export default function LoginForm({ role, closeLoginForm }) {
    const { profiles } = useContext(ProfilesContext);
    const { setActiveUser } = useContext(ActiveUserContext);
    const { setActiveCoach } = useContext(ActiveCoachContext);
    const [message, setMessage] = useState('');
    let profile = {};

    const loginProfile = (e) => {
        e.preventDefault();
        const { elements } = e.target.parentElement;

        const trueForm = isFormCorrect(elements);
        if (!trueForm) {
            setMessage('Необходимо заполнить все поля формы');
        }

        if (trueForm) {
                const profileLoginData = getProfileDataFromForm(elements,
role);
            profile = findProfile(profiles, profileLoginData.email, role);

            if (profile) {
                if (isPasswordCorrect(profile, profileLoginData.password))
{
                    localStorage.setItem(role === 'coach' ? 'activeCoach'
: 'activeUser', JSON.stringify(profile));
                    if (role === 'user') {
                        setActiveUser(profile);
                    }
                    if (role === 'coach') {
                        setActiveCoach(profile);
                    }
                    setMessage('');
                    closeLoginForm();
                } else {
```

```
                    setMessage('Неверный пароль');
                }
            } else {
                setMessage('Пользователь не найден');
            }
        }
    }

    return (
        <form className="form reg">
                        <h3>Войти как {role === 'coach' ? 'тренер' :
'пользователь'}</h3>
                    <input className="form__input" type="text" name="email"
placeholder="E-mail" required />
             <input className="form__input" type="password" name="password"
placeholder="Пароль" required />
                                            <button    className="button"
onClick={loginProfile}>Войти</button>
            <span>{message}</span>
        </form>
    )
}
```

Приложение 8. Компонент <RegistrationForm />

```
import { useContext, useState } from "react";
import { ActiveCoachContext, ActiveUserContext, ProfilesContext } from
"../contexts/ContextProvider";
import { findProfile, createProfile, isFormCorrect } from
"../modules/formModules";
import axios from "axios";

export default function RegistrationForm({ role, closeRegForm }) {
    const { profiles, setProfiles } = useContext(ProfilesContext);
    const { setActiveUser } = useContext(ActiveUserContext);
    const { setActiveCoach } = useContext(ActiveCoachContext);
    const [message, setMessage] = useState('');

    const createUser = (e) => {
```

```
        e.preventDefault();
        const { elements } = e.target.parentElement;

        const trueForm = isFormCorrect(elements, role);
        if (!trueForm) {
            setMessage('Необходимо заполнить все поля формы');
        }

        if (trueForm) {
            const newProfile = createProfile(elements, role, profiles);
            const checkProfile = findProfile(profiles, newProfile.email,
role);

            console.log(newProfile);

            if (checkProfile) {
                setMessage('Пользователь уже существует!');
            } else if (newProfile.password !== newProfile.checkPassword) {
                setMessage('Пароли не совпадают.')
            }
            else {

axios.post(`https://66598df5de346625136ceaa4.mockapi.io/profiles/`,
newProfile);
                setProfiles([...profiles, newProfile])
                setMessage('ВЫ УСПЕШНО ЗАРЕГИСТРИРОВАЛИСЬ!');
                localStorage.setItem(role === 'coach' ? 'activeCoach' :
'activeUser', JSON.stringify(newProfile));
                if (role === 'user') {
                    setActiveUser(newProfile);
                }
                if (role === 'coach') {
                    setActiveCoach(newProfile);
                }
                setTimeout(() => {
                    closeRegForm();
                }, 1000);
            }
        }
    }

    return (
        <form className="form reg">
```

```
            <h3>Регистрация {role === 'coach' ? 'тренера' :
'пользователя'}</h3>
            <input className="form__input" type="text" name="name"
placeholder="Имя" required />
            <input className="form__input" type="text" name="surname"
placeholder="Фамилия" required />
            <input className="form__input" type="email" name="email"
placeholder="E-mail" required />
            <input className="form__input" type="password" name="password"
placeholder="Пароль" required />
            <input className="form__input" type="password"
name="checkPassword" placeholder="Повтор пароля" required />
            <button className="button"
onClick={createUser}>Зарегистрироваться</button>
            <span>{message}</span>
        </form>
    )
}
```

Приложение 9. Компонент <Submenu />

```
import { useContext } from "react";
import { Link } from "react-router-dom";
import { TypesContext } from '../contexts/ContextProvider';

export default function Submenu() {
    const workoutTypes = useContext(TypesContext);

    return (
        <nav>
            <ul className="submenu">
                                {workoutTypes.map(type  =>  <li  key={type.id}
className="submenu__link"><Link
to={`/workouts/${type.path}`}>{type.name}</Link></li>)}
            </ul>
        </nav>
    )
}
```

Приложение 10. Компонент <Subheader />

```
import { useLocation } from "react-router-dom";

export default function Subheader() {
    const { pathname } = useLocation();
    const clear = pathname.split('/')[1];
    let title = '';
    switch (clear) {
        case '':
            title = '';
            break;
        case 'workouts':
            title = 'Онлайн-занятия';
            break;
        case 'streams':
            title = 'Вебинары';
            break;
        case 'articles':
            title = 'Статьи';
            break;
        case 'about':
            title = 'О платформе';
            break;
        case 'contacts':
            title = 'Контакты';
            break;
        case 'subscriptions':
            title = 'Подписки';
            break;
        case 'payment_subscription':
            title = 'Оплата подписки';
            break;
        case 'coach_about':
            title = 'Стать тренером';
            break;
        case 'users':
            title = 'Страница пользователя';
            break;
        case 'profiles':
            title = 'Страница тренера';
            break;
        case 'coaches':
```

```
            title = 'Страница тренера';
            break;
        case 'public_coaches':
            title = 'Тренер платформы';
            break;
        case 'user_offer':
            title = 'Оферта с пользователем';
            break;
        case 'coach_offer':
            title = 'Оферта с тренером';
            break;
        case 'user_agreement':
            title = 'Пользовательское соглашение';
            break;
        case 'confidential_policy':
            title = 'Политика конфиденциальности ';
            break;
        case 'email_confirm':
            title = 'Регистрация завершена';
            break;
        default:
            title = 'Ooops...';
            break;
    }

    return (
        <div className='subheader'>
            <h1 className="subheader__title">{title}</h1>
        </div>
    )
}
```

Приложение 11. Компонент <Footer />

```
import React from "react";
import { Link } from "react-router-dom";

export default function Footer() {
```

```
    return (
        <footer className="footer">
            <menu className="footer__menu">
                <ul className="footer__menu-left">
                                        <li><Link  to='/about'>О  платформе
FitSteps.ru</Link></li>
                                      <li><Link  to='/contacts'>Контактная
информация</Link></li>
                                   <li><Link  to='/subscriptions'>Оформить
подписку</Link></li>
                                     <li><Link  to='/user_offer'>Оферта  с
пользователем</Link></li>
                                    <li><Link  to='/coach_offer'>Оферта  с
тренером</Link></li>
                               <li><Link  to='/user_agreement'>Пользовательское
соглашение</Link></li>
                                  <li><Link  to='/confidential_policy'>Политика
конфиденциальности</Link></li>
                </ul>
                <ul className="footer__menu-right">
                    <li><Link to='/workouts'>Занятия</Link></li>
                    <li><Link to='/streams'>Вебинары</Link></li>
                    <li><Link to='/articles'>Статьи</Link></li>
                </ul>
                <ul className="footer__menu-coach">
                    <li><Link to='/coach_about'>Стать тренером</Link></li>
                </ul>
            </menu>
            <div className="footer__copyright">
                <b>&copy; 2024 Это учебный проект Марии Савкиной</b><br />
                 Вся информация на сайте несет исключительно информационный
характер и создана для имитации работающего сайта в рамках дипломного
проекта по специализации Frontend React.</div>
        </footer>
    )
}
```

Приложение 12. Компонент <Breadcrumbs />

```
import { Link, useLocation } from "react-router-dom";
```

```
export default function Breadcrumbs({ items, types = null }) {
    const { pathname } = useLocation();
    const pathParts = pathname
        .split('/')
        .filter(part => part !== '');
    let link = '';

    return (
        <ul className="breadcrumbs">
            {
                pathParts.map((part, index) => {
                    link += ('/' + part);
                    let subtitle = '';
                    if (index === 0) {
                        switch (part) {
                            case 'workouts':
                                subtitle = 'Все занятия';
                                break;
                            case 'streams':
                                subtitle = 'Все вебинары';
                                break;
                            case 'articles':
                                subtitle = 'Все статьи';
                                break;
                            default:
                                break;
                        }
                    }
                    if (index === 1) {
                        if (pathParts[0] === 'workouts') {
                                subtitle = types.find(type => type.path ===
part).name;
                        } else {
                                 subtitle = items.find(item => item.id ===
part).title;
                        }
                    }
                    if (index > 1 && pathParts[0] === 'workouts') {
                                subtitle = items.find(item => item.id ===
part).title;
                    }
```

```
                                                return  <li  key={index}><Link
to={link}>{subtitle}</Link></li>
                })
            }
        </ul>
    )
}
```

Приложение 13. Компонент <Home />

```
import { useContext } from "react";
import { DataContext } from "../../contexts/ContextProvider";
import BannerAbout from "./BannerAbout";
import SubscribeHome from "./SubscribeHome";
import Top5Workouts from "./Top5Workouts";
import Top3Articles from "./Top3Articles";
import Top5Streams from "./Top5Streams";

export default function Home() {
    const { workouts } = useContext(DataContext);
    const { streams } = useContext(DataContext);
    const { articles } = useContext(DataContext);

    return (
        <div>
            <BannerAbout />
            <Top5Workouts workouts={workouts} />
            <SubscribeHome />
            <Top5Streams streams={streams} />
            <Top3Articles articles={articles} />
        </div>
    )
}
```

Приложение 14. Компонент <BannerAbout />

```
import { useEffect, useState } from "react";
```

```
export default function BannerAbout() {
    const images = ['01', '02', '03', '04', '05'];
    const [imagePath, setImagePath] = useState('./images/banner/01.jpg');
    const [imageIndex, setImageIndex] = useState(0);

    const renderBannerPath = () => {
        setTimeout(() => {
            setImagePath(`./images/banner/${images[imageIndex]}.jpg`);
             setImageIndex((imageIndex < images.length - 1) ? imageIndex +
1 : 0);
        },
            4000);
    };

    useEffect(() => {
        renderBannerPath();
    }, [imageIndex]);

    return (
        <div className='Banner'>
            <img src={imagePath} alt="Баннер" />
        </div>
    )
}
```

Приложение 15. Компонент <SubscribeHome />

```
import { Link } from "react-router-dom";

export default function SubscribeHome() {
    return (
        <div className='SubscribeHome'
            style={{
                background: "url('./images/subbanner.jpg')",
                backgroundRepeat: "no-repeat",
                backgroundSize: "100%"
            }}>
            <h3>Узнай больше о всех возможностях с платной подпиской!</h3>
```

```
            <p>Все подробности!</p>
            <Link to='/subscriptions'>тут</Link>
        </div>
    )
}
```

Приложение 16. Компонент <Top3Articles />

```
import { Link } from "react-router-dom"
import ArticlePreview from "../../components/ArticlePreview"

export default function Top3Articles({ articles }) {
    return (
        <div className="top green">
            <h3 className="top__title">Последние статьи</h3>
            <ul className='top__list'>
                {articles.map((article, index) => {
                    if (index < 3) return (
                        <li key={article.id}><ArticlePreview
article={article} width={400} height={360} /></li>
                    )
                })}
            </ul>
            <div className="top__all"><Link to='/articles'>Показать
все</Link></div>
        </div>
    )
}
```

Приложение 17. Компонент <Top5Streams />

```
import { Link } from "react-router-dom";
import StreamPreview from "../../components/StreamPreview";

export default function Top5Streams({ streams }) {
    return (
        <div className="top blue">
            <h3 className="top__title">ТОП-5 вебинаров</h3>
```

```
            <ul className='top__list'>
                {streams.map((stream, index) => {
                    if (index < 5) return (
                        <li key={stream.id}><StreamPreview stream={stream}
width={230} height={360} /></li>
                    )
                })}
            </ul>
                    <div  className="top__all"><Link  to='/streams'>Показать
все</Link></div>
        </div>
    )
}
```

Приложение 18. Компонент <Top5Workouts />

```
import { Link } from "react-router-dom"
import WorkoutPreview from "../../components/WorkoutPreview"

export default function Top5Workouts({ workouts }) {
    return (
        <div className="top pink">
            <h3 className="top__title">ТОП-5 онлайн-занятий</h3>
            <ul className='top__list'>
                {workouts.map((workout, index) => {
                    if (index < 5) return (
                                        <li  key={workout.id}><WorkoutPreview
workout={workout} width={230} height={380} /></li>
                    )
                })}
            </ul>
                    <div  className="top__all"><Link  to='/workouts'>Показать
все</Link></div>
        </div>
    )
}
```

Приложение 19. Компонент <Workouts />

```
import { useContext } from "react";
import { useParams } from "react-router-dom";
import { TypesContext, DataContext } from "../contexts/ContextProvider";
import Breadcrumbs from "../components/Breadcrumbs";
import WorkoutPreview from "../components/WorkoutPreview";

export default function Workouts() {
    const types = useContext(TypesContext);
    const { workouts } = useContext(DataContext);
    const { typePath } = useParams();

    let filteredWorkouts = [];
    if (typePath) {
        const type = types.find(type => type.path === typePath);
           filteredWorkouts = workouts.filter(workout => workout.type ===
type.name);
    } else {
        filteredWorkouts = workouts;
    }

    return (
        <section>
            <Breadcrumbs items={workouts} types={types} />
            <div className="content">
                <ul className='workouts'>
                                    {filteredWorkouts.map(workout => <li
key={workout.id}><WorkoutPreview workout={workout} /></li>)}
                </ul>
            </div>
        </section>
    )
}
```

Приложение 20. Компонент <WorkoutPreview />

```
import React, { useEffect, useState } from "react";
import { useContext } from "react";
import { ActiveUserContext, ProfilesContext, TypesContext } from
"../contexts/ContextProvider";
```

```
import { Link } from 'react-router-dom';
import { getCoach, getScheduleHTML, getTypePath, renderFavorite,
toggleFavorite } from '../modules/workoutsFunctions';
import axios from "axios";

export default function WorkoutPreview({ workout, width = 400, height =
500 }) {
    const types = useContext(TypesContext);
    const { profiles } = useContext(ProfilesContext);
    const typePath = getTypePath(workout, types);
    const schedule = getScheduleHTML(workout);
    const coach = getCoach(workout, profiles);
    const { activeUser, editActiveUser } = useContext(ActiveUserContext);
    const [favoriteIcon, setFavoriteIcon] =
useState(renderFavorite(activeUser, workout.id));

    const handleFavorite = () => {
        const newUser = toggleFavorite(activeUser, workout.id);
        editActiveUser(newUser);
        localStorage.setItem('activeUser', JSON.stringify(newUser));

axios.put(`https://66598df5de346625136ceaa4.mockapi.io/profiles/${newUser.
id}`, newUser);
        setFavoriteIcon(renderFavorite(activeUser, workout.id));
    }

    useEffect(() => {
        setFavoriteIcon(renderFavorite(activeUser, workout.id));
    }, [activeUser, workout.id]);

    return (
        <article>
            {activeUser &&
                <div className="favorite"
onClick={handleFavorite}>{favoriteIcon}</div>}

            <Link to={`/workouts/${typePath}/${workout.id}`}>
                <div className="workouts__item" style={{ width:
`${width}px`, height: `${height}px` }}>
                    <span>{workout.type}</span>
                    <b>{workout.title}</b>
                    <img className='workouts__preview' src={workout.image}
alt="Превью" />
```

```
                    <div>
                        <h4>Расписание:</h4>
                        <span>{schedule}</span>
                    </div>
                    <span>Продолжительность: {workout.duration * 60}
мин.</span>
                    <h4>Тренер: <Link
to={`/coaches/${workout.coachId}`}>{coach.name}
{coach.surname}</Link></h4>
                </div>
            </Link>
        </article>
    );
}
```

Приложение 21. Компонент <WorkoutDetails />

```
import React, { useEffect, useState } from "react";
import { useContext } from "react";
import { useParams, Link } from "react-router-dom";

import Breadcrumbs from "../components/Breadcrumbs";
import { getCoach, getDescription, getScheduleHTML, getClosestStreamDate,
renderFavorite, toggleFavorite } from '../modules/workoutsFunctions';
import { ActiveUserContext, ProfilesContext, TypesContext, DataContext }
from "../contexts/ContextProvider";
import axios from "axios";

export default function WorkoutDetails() {
    const { id } = useParams();

    const types = useContext(TypesContext);
    const { profiles } = useContext(ProfilesContext);
    const { activeUser, editActiveUser } = useContext(ActiveUserContext);

    const { workouts } = useContext(DataContext);
    const workout = workouts.find(workout => workout.id === id);
    const schedule = getScheduleHTML(workout);
    const coach = getCoach(workout, profiles);
    const description = getDescription(workout, types);
    const streamDate = getClosestStreamDate(workout);
```

```
    const [favoriteIcon, setFavoriteIcon] =
useState(renderFavorite(activeUser, workout.id));

    const handleFavorite = () => {
        const newUser = toggleFavorite(activeUser, workout.id);
        editActiveUser(newUser);
        localStorage.setItem('activeUser', JSON.stringify(newUser));

axios.put(`https://66598df5de346625136ceaa4.mockapi.io/profiles/${newUser.
id}`, newUser);
        setFavoriteIcon(renderFavorite(activeUser, workout.id));
    }

    useEffect(() => {
        setFavoriteIcon(renderFavorite(activeUser, workout.id));
    }, [activeUser, workout.id]);

    if (!workout) {
        return <div>Такого занятие нет.</div>
    }

    return (
        <section className="text">
            <Breadcrumbs items={workouts} types={types} />

            <h1 className="workouts__title">Курс занятий:
"{workout.title}"</h1>
            {activeUser &&
                <div className="favorite"
onClick={handleFavorite}>{favoriteIcon}</div>}
            <div className="favorite-details" onClick={handleFavorite}>
                {favoriteIcon}
            </div>
            <div className="workouts__info">
                <div className="workouts__left">
                    <p className="workouts__block">Дата ближайшего
занятия: {streamDate}</p>
                    <div className="workouts__block">
                        <h3>Расписание:</h3>
                        <ul>
                            {schedule}
                        </ul>
```

```
                    </div>
                    <p className="workouts__block">Продолжительность:
{workout.duration * 60} мин.</p>
                </div>
                <div className="workouts__right">
                    <p className="workouts__block">{description}</p>
                    <p className="workouts__block">Тренер: <Link
to={`/coaches/${workout.coachId}`}>{coach.name} {coach.surname}</Link></p>
                </div>
            </div>
            <div className="workouts__block">Ссылка на ближайшее
онлайн-занятие: {workout.streamUrl}</div>

            <img className="workouts__img" src={workout.image}
alt="Превью" />
            <div className="workouts__block"><Link to='/workouts'>К
занятиям</Link></div>
        </section>
    )
}
```

Приложение 22. Компонент <Streams />

```
import React, { useContext } from "react";
import Breadcrumbs from "../components/Breadcrumbs";
import { DataContext } from "../contexts/ContextProvider";
import StreamPreview from "../components/StreamPreview";

export default function Streams() {
    const { streams } = useContext(DataContext);

    return (
        <section>
            <Breadcrumbs items={streams} />
            <div className="content">
                <ul className='workouts'>
                                                {streams.map(stream  =>  <li
key={stream.id}><StreamPreview stream={stream} /></li>)}
                </ul>
            </div>
```

```
        </section>
    )
}
```

Приложение 23. Компонент <StreamPreview />

```
import { useContext } from "react";
import { ProfilesContext } from "../contexts/ContextProvider";
import { Link } from 'react-router-dom';
import { getCoach } from '../modules/workoutsFunctions';

export default function StreamPreview({ stream, width = 400, height = 500
}) {
    const { profiles } = useContext(ProfilesContext);
    const [date, time] = stream.date.split('-');
    const coach = getCoach(stream, profiles);

    return (
        <article>
            <Link to={`/streams/${stream.id}`}>
                            <div  className="workouts__item"  style={{  width:
`${width}px`, height: `${height}px` }}>
                    <b>{stream.title}</b>
                      <img className='workouts__preview' src={stream.image}
alt="Превью" />
                    <div>
                        <h4>Дата: {date}</h4>
                        <span>Время: {time}</span>
                    </div>
                            <span>Продолжительность: {stream.duration * 60}
мин.</span>
                                                                <h4>Тренер:   <Link
to={`/coaches/${stream.coachId}`}>{coach.name} {coach.surname}</Link></h4>
                </div>
            </Link>
        </article>
    );
}
```

Приложение 24. Компонент <StreamsDetails />

```
import { useContext } from "react";
import { useParams, Link } from "react-router-dom";
import Breadcrumbs from "../components/Breadcrumbs";
import { getCoach } from '../modules/workoutsFunctions';
import { ProfilesContext, DataContext } from
"../contexts/ContextProvider";

export default function StreamsDetails() {
    const { id } = useParams();
    const { profiles } = useContext(ProfilesContext);
    const { streams } = useContext(DataContext);
    const stream = streams.find(stream => +stream.id === +id);
    const [day, time] = stream.date.split('-');
    const coach = getCoach(stream, profiles);

    if (!stream) {
        return <div>Такого вебинара нет.</div>
    }

    return (
        <section className="text">
            <Breadcrumbs items={streams} />

            <h1 className="workouts__title">Вебинар: "{stream.title}"</h1>
            <div className="workouts__info">
                <div className="workouts__left">
                            <p  className="workouts__block">Дата  вебинара:
{day}</p>
                    <p className="workouts__block">Время начала вебинара:
{time}</p>
                        <p  className="workouts__block">Продолжительность:
{stream.duration * 60} мин.</p>
                </div>
                <div className="workouts__right">
                                                                    <p
className="workouts__block">{stream.description}</p>
                                <p  className="workouts__block">Тренер:  <Link
to={`/coaches/${stream.coachId}`}>{coach.name} {coach.surname}</Link></p>
                </div>
            </div>
```

```
                    <div  className="workouts__block">Ссылка  на  вебинар
{stream.streamUrl}</div>

            <img className="workouts__img" src={stream.image} alt="Превью"
/>
                <div  className="workouts__block"><Link  to='/streams'>К
вебинарам</Link></div>

        </section>
    )
}
```

Приложение 25. Компонент <Articles />

```
import React, { useContext } from "react";
import Breadcrumbs from "../components/Breadcrumbs";
import { DataContext } from "../contexts/ContextProvider";
import ArticlePreview from "../components/ArticlePreview";

export default function Articles() {
    const { articles } = useContext(DataContext);

    return (
        <section>
            <Breadcrumbs items={articles} />
            <div className="content">
                <ul className='workouts'>
                    {articles.map(article => <li
key={article.id}><ArticlePreview article={article} height={700} /></li>)}
                </ul>
            </div>
        </section>
    )
}
```

Приложение 26. Компонент <ArticlePreview />

```
import { useContext } from "react";
```

```
import { ProfilesContext } from "../contexts/ContextProvider";
import { Link } from 'react-router-dom';
import { getCoach } from '../modules/workoutsFunctions';

export default function ArticlePreview({ article, width = 800, height =
600 }) {
    const { profiles } = useContext(ProfilesContext);
    const coach = getCoach(article, profiles);

    return (
        <article>
            <Link to={`/articles/${article.id}`}>
                <div className="workouts__item" style={{ width:
`${width}px`, height: `${height}px` }}>
                    <b>{article.title}</b>
                    <img className='workouts__preview' src={article.image}
alt="Превью" />
                    <div>
                        <h4>Дата: {article.date}</h4>
                    </div>
                    <h4>Автор: <Link
to={`/coaches/${article.coachId}`}>{coach.name}
{coach.surname}</Link></h4>
                </div>
            </Link>
        </article>
    );
}
```

Приложение 27. Компонент <ArticleDetails />

```
   import { useContext } from "react";
import { useParams, Link } from "react-router-dom";
import Breadcrumbs from "../components/Breadcrumbs";
import { getCoach } from '../modules/workoutsFunctions';
import { ProfilesContext, DataContext } from
"../contexts/ContextProvider";

export default function ArticleDetails() {
```

```
    const { id } = useParams();
    const { profiles } = useContext(ProfilesContext);
    const { articles } = useContext(DataContext);
    const article = articles.find(article => article.id === id);
    const coach = getCoach(article, profiles);

    if (!article) {
        return <div>Такой статьи нет.</div>
    }

    return (
        <section className="text">
            <Breadcrumbs items={articles} />

            <h1 className="workouts__title">Статья: "{article.title}"</h1>
            <p className="workouts__block">Дата: {article.date}</p>
            <p className="workouts__block">Автор: <Link
to={`/coaches/${article.coachId}`}>{coach.name} {coach.surname}</Link></p>
            <p className="workouts__block">{article.text}</p>
            <div className="workouts__block"><Link to='/articles'>К
статьям</Link></div>

        </section>
    )
}
```

Приложение 28. Компонент <UserAccount />

```
import React, { useContext, useState } from "react";
import       {       ActiveUserContext,       DataContext       }       from
"../contexts/ContextProvider";
import { Link } from "react-router-dom";
import WorkoutPreview from "../components/WorkoutPreview";
import axios from "axios";

export default function UserAccount() {
    const { activeUser, editActiveUser } = useContext(ActiveUserContext);
    const { workouts } = useContext(DataContext);
```

```
            const   myWorks   =   activeUser.favorites.map(favorite   =>
workouts.find(workout => +workout.id === favorite));

    const [allowEditName, setAllowEditName] = useState(false);
    const [name, setName] = useState(activeUser.name)

    const [allowEditSurname, setAllowEditSurname] = useState(false);
    const [surname, setSurname] = useState(activeUser.surname)

    const [allowEditBirthday, setAllowEditBirthday] = useState(false);
    const [birthday, setBirthday] = useState(activeUser.birthday)

    const [allowEditEmail, setAllowEditEmail] = useState(false);
    const [email, setEmail] = useState(activeUser.email)

    const [allowEditAbout, setAllowEditAbout] = useState(false);
    const [about, setAbout] = useState(activeUser.about)

    const [allowEditPassword, setAllowEditPassword] = useState(false);
    const [password, setPassword] = useState(activeUser.password);
    const [checkPassword, setCheckPassword] = useState('');

    const setEditedData = (newUser) => {
        editActiveUser(newUser);
        localStorage.setItem('activeUser', JSON.stringify(newUser));

axios.put(`https://66598df5de346625136ceaa4.mockapi.io/profiles/${newUser.
id}`, newUser);
    }

    const editName = (e) => {
        onChangeAllowEditName()
        setName(e.target.value);
    }
    const onChangeAllowEditName = () => {
        setAllowEditName(true);
        document.getElementById('name').classList.add('white');
    }
    const onChangeName = () => {
        if (name.length >= 2) {
            setAllowEditName(false);
            const newUser = activeUser;
            newUser.name = name;
```

```
            setEditedData(newUser);
            document.getElementById('name').classList.remove('white');
        }
    }

    const editSurname = (e) => {
        onChangeAllowEditSurname()
        setSurname(e.target.value);
    }
    const onChangeAllowEditSurname = () => {
        setAllowEditSurname(true);
        document.getElementById('surname').classList.add('white');
    }
    const onChangeSurname = () => {
        if (surname.length >= 2) {
            setAllowEditSurname(false)
            const newUser = activeUser;
            newUser.surname = surname;
            setEditedData(newUser);
            document.getElementById('surname').classList.remove('white');

        }
    }

    const editBirthday = (e) => {
        onChangeAllowEditBirthday()
        setBirthday(e.target.value);
    }
    const onChangeAllowEditBirthday = () => {
        setAllowEditBirthday(true);
        document.getElementById('birthday').classList.add('white');

    }
    const onChangeBirthday = () => {
        setAllowEditBirthday(false)
        const newUser = activeUser;
        newUser.birthday = birthday;
        setEditedData(newUser);
        document.getElementById('birthday').classList.remove('white');
    }

    const editEmail = (e) => {
        onChangeAllowEditEmail()
```

```
        setEmail(e.target.value);
    }
    const onChangeAllowEditEmail = () => {
        setAllowEditEmail(true);
        document.getElementById('email').classList.add('white');

    }
    const onChangeEmail = () => {
        if (email.length >= 2) {
            setAllowEditEmail(false)
            const newUser = activeUser;
            newUser.email = email;
            setEditedData(newUser);
            document.getElementById('email').classList.remove('white');

        }
    }

    const editAbout = (e) => {
        onChangeAllowEditAbout()
        setAbout(e.target.value);
    }
    const onChangeAllowEditAbout = () => {
        setAllowEditAbout(true);
        document.getElementById('about').classList.add('white');

    }
    const onChangeAbout = () => {
        if (about.length >= 2) {
            setAllowEditAbout(false)
            const newUser = activeUser;
            newUser.about = about;
            setEditedData(newUser);
            document.getElementById('about').classList.remove('white');
        }
    }

    const editPassword = (e) => {
        if (allowEditPassword) {
            setPassword(e.target.value);
        }
    }
    const checkingPassword = (e) => {
```

```
        if (allowEditPassword) {
            setCheckPassword(e.target.value);
        }
    }
    const onChangeAllowEditPassword = () => {
        setAllowEditPassword(true);
    }
    const onChangePassword = () => {
        if (password.length >= 5 && password === checkPassword) {
            setAllowEditPassword(false)
            const newUser = activeUser;
            newUser.password = password;
            setEditedData(newUser);
        }
    }

    return (
        <div className="personal">
            <section className="personal__data">
                <div className="personal__avatar">
                    <img src={activeUser.avatar || null} alt="Avatar" />
                </div>
                <div className="personal__info">
                    <div className="personal__subscribe">
                        <span>Активная подписка:
                                        {activeUser.subscribe.isActive ? `до
${activeUser.subscribe.endDate} г.` : ' НЕТ'}
                        </span>
                        <span>
                                {activeUser.subscribe.Type ? `Тип подписки:
${activeUser.subscribe.type}`     :     <Link     to="/subscriptions"><button
className="button">оформить подписку</button></Link>}
                        </span>
                    </div>
                    <div className="personal__info-item">
                        <span>Имя</span>
                            <input id="name" type="text" value={name}
onChange={editName} />
                        {allowEditName ?
                                                <img width={20} height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeName} /> :
                                                <img width={20} height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditName} />}
```

```
                    </div>
                    <div className="personal__info-item">
                        <span>Фамилия</span>
                            <input id="surname" type="text" value={surname}
onChange={editSurname} />
                        {allowEditSurname ?
                                                <img  width={20}  height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeSurname} /> :
                                                <img  width={20}  height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditSurname} />}
                    </div>
                    <div className="personal__info-item">
                        <span>День рождения</span>
                          <input id="birthday" type="date" value={birthday}
onChange={editBirthday} />
                        {allowEditBirthday ?
                                                <img  width={20}  height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeBirthday} /> :
                                                <img  width={20}  height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditBirthday} />}
                    </div>
                    <div className="personal__info-item">
                        <span>Эл. почта</span>
                               <input id="email" type="text" value={email}
onChange={editEmail} />
                        {allowEditEmail ?
                                                <img  width={20}  height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeEmail} /> :
                                                <img  width={20}  height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditEmail} />}
                    </div>
                    {allowEditPassword ?
                        <form className="form">
                             <input className="form__input" type="password"
value={password} placeholder="Введите пароль" onChange={editPassword} />
                            <input className="form__input" type="password"
value={checkPassword}            placeholder="Повторите             пароль"
onChange={checkingPassword} />
                            {password === checkPassword && password.length
>= 5 ?
                                                <img  width={20}  height={20}
src="/images/edit.png" alt="Edit" onClick={onChangePassword} /> :
```

```
                                        <span className="error">Пароли не
совпадают или указан пароль менее 5 символов</span>}
                        </form> :
                                            <button  className="change-pass"
onClick={onChangeAllowEditPassword}>Изменить пароль</button>
                    }
                </div>
                <div className="personal__about">
                    <h3>Обо мне</h3>
                            <textarea  id="about"  type="text"  value={about}
onChange={editAbout} placeholder="Напишите о себе" />
                    {allowEditAbout ?
                                                <img  width={30}  height={30}
src="/images/checked.png" alt="Edit" onClick={onChangeAbout} /> :
                                                <img  width={30}  height={30}
src="/images/edit_big.png" alt="Edit" onClick={onChangeAllowEditAbout} />}
                </div>

            </section>
            <section className="favorites">
                <h3>Мои занятия</h3>
                {myWorks.length > 0 ?
                    <ul className="workouts">
                                                {myWorks.map(workout  =>  <li
key={workout.id}><WorkoutPreview       workout={workout}       width={200}
height={400} /></li>)}
                    </ul> :
                     <div className="text"><p>Вы пока не добавили ни одного
онлайн-занятия  в  избранное.  Выберите  занятие  <Link  to='/workouts'>в
разделе "онлайн-занятий".</Link></p>
                    </div>}

            </section>
        </div>
    )
}
```

Приложение 29. Компонент <CoachAccount />

```
import React, { useContext, useState } from "react";
import { ActiveCoachContext, DataContext } from
"../contexts/ContextProvider";
import { Link } from "react-router-dom";
import WorkoutPreview from "../components/WorkoutPreview";
import axios from "axios";

export default function CoachAccount() {
    const { activeCoach, setActiveCoach, editActiveCoach } =
useContext(ActiveCoachContext);

    const { workouts } = useContext(DataContext);
    const myWorks = workouts.filter(workout => +workout.coachId ===
+activeCoach.id);

    const [allowEditName, setAllowEditName] = useState(false);
    const [name, setName] = useState(activeCoach.name)

    const [allowEditSurname, setAllowEditSurname] = useState(false);
    const [surname, setSurname] = useState(activeCoach.surname)

    const [allowEditBirthday, setAllowEditBirthday] = useState(false);
    const [birthday, setBirthday] = useState(activeCoach.birthday)

    const [allowEditEmail, setAllowEditEmail] = useState(false);
    const [email, setEmail] = useState(activeCoach.email)

    const [allowEditAbout, setAllowEditAbout] = useState(false);
    const [about, setAbout] = useState(activeCoach.about)

    const [allowEditPassword, setAllowEditPassword] = useState(false);
    const [password, setPassword] = useState(activeCoach.password);
    const [checkPassword, setCheckPassword] = useState('');

    const removeActiveCoach = () => {
        if (activeCoach) {
            localStorage.removeItem('activeCoach');
            setActiveCoach(null);
        }
    };
```

```
    const setEditedData = (newCoach) => {
        editActiveCoach(newCoach);
        localStorage.setItem('activeCoach', JSON.stringify(newCoach));

axios.put(`https://66598df5de346625136ceaa4.mockapi.io/profiles/${newCoach
.id}`, newCoach);
    }

    const editName = (e) => {
        onChangeAllowEditName();
        setName(e.target.value);
    }
    const onChangeAllowEditName = () => {
        setAllowEditName(true);
        document.getElementById('name').classList.add('white');
    }
    const onChangeName = () => {
        if (name.length >= 2) {
            setAllowEditName(false);
            const newCoach = activeCoach;
            newCoach.name = name;
            setEditedData(newCoach);
            document.getElementById('name').classList.remove('white');
        }
    }

    const editSurname = (e) => {
        onChangeAllowEditSurname();
        setSurname(e.target.value);
    }
    const onChangeAllowEditSurname = () => {
        setAllowEditSurname(true);
        document.getElementById('surname').classList.add('white');
    }
    const onChangeSurname = () => {
        if (surname.length >= 2) {
            setAllowEditSurname(false)
            const newCoach = activeCoach;
            newCoach.surname = surname;
            setEditedData(newCoach);
            document.getElementById('surname').classList.remove('white');
        }
```

```
    }

    const editBirthday = (e) => {
        onChangeAllowEditBirthday();
        setBirthday(e.target.value);
    }
    const onChangeAllowEditBirthday = () => {
        setAllowEditBirthday(true);
        document.getElementById('birthday').classList.add('white');

    }
    const onChangeBirthday = () => {
        setAllowEditBirthday(false)
        const newCoach = activeCoach;
        newCoach.birthday = birthday;
        setEditedData(newCoach);
        document.getElementById('birthday').classList.remove('white');
    }

    const editEmail = (e) => {
        onChangeAllowEditEmail();
        setEmail(e.target.value);
    }
    const onChangeAllowEditEmail = () => {
        setAllowEditEmail(true);
        document.getElementById('email').classList.add('white');

    }
    const onChangeEmail = () => {
        if (email.length >= 2) {
            setAllowEditEmail(false)
            const newCoach = activeCoach;
            newCoach.email = email;
            setEditedData(newCoach);
            document.getElementById('email').classList.remove('white');

        }
    }

    const editAbout = (e) => {
        onChangeAllowEditAbout();
        setAbout(e.target.value);
    }
```

```
    const onChangeAllowEditAbout = () => {
        setAllowEditAbout(true);
        document.getElementById('about').classList.add('white');

    }
    const onChangeAbout = () => {
        if (about.length >= 2) {
            setAllowEditAbout(false)
            const newCoach = activeCoach;
            newCoach.about = about;
            setEditedData(newCoach);
            document.getElementById('about').classList.remove('white');
        }
    }

    const editPassword = (e) => {
        if (allowEditPassword) {
            setPassword(e.target.value);
        }
    }
    const checkingPassword = (e) => {
        if (allowEditPassword) {
            setCheckPassword(e.target.value);
        }
    }
    const onChangeAllowEditPassword = () => {
        setAllowEditPassword(true);
    }
    const onChangePassword = () => {
        if (password.length >= 5 && password === checkPassword) {
            setAllowEditPassword(false)
            const newCoach = activeCoach;
            newCoach.password = password;
            setEditedData(newCoach);
        }
    }

    return (
        <div className="personal">
            <section className="personal__data">
                <div className="personal__avatar">
                    <img src={activeCoach.avatar || null} alt="Avatar" />
                </div>
```

```
                <div className="personal__info">
                    <div className="personal__subscribe">
                          <span>Текущий баланс: {activeCoach.balance > 0 ?
activeCoach.balance : 0} руб.</span>
                        <span>
                                        {activeCoach.balance > 0 && <button
className="button">вывести средства</button>}
                        </span>
                    </div>
                    <div className="personal__info-item">
                        <span>Имя</span>
                                <input id="name" type="text" value={name}
onChange={editName} />
                        {allowEditName ?
                                                <img width={20} height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeName} /> :
                                                <img width={20} height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditName} />}
                    </div>
                    <div className="personal__info-item">
                        <span>Фамилия</span>
                           <input id="surname" type="text" value={surname}
onChange={editSurname} />
                        {allowEditSurname ?
                                                <img width={20} height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeSurname} /> :
                                                <img width={20} height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditSurname} />}
                    </div>
                    <div className="personal__info-item">
                        <span>День рождения</span>
                          <input id="birthday" type="date" value={birthday}
onChange={editBirthday} />
                        {allowEditBirthday ?
                                                <img width={20} height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeBirthday} /> :
                                                <img width={20} height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditBirthday} />}
                    </div>
                    <div className="personal__info-item">
                        <span>Эл. почта</span>
                               <input id="email" type="text" value={email}
onChange={editEmail} />
```

```
                        {allowEditEmail ?
                                                <img  width={20}  height={20}
src="/images/checked.png" alt="Edit" onClick={onChangeEmail} /> :
                                                <img  width={20}  height={20}
src="/images/edit.png" alt="Edit" onClick={onChangeAllowEditEmail} />}
                    </div>
                    {allowEditPassword ?
                        <form className="form">
                             <input className="form__input" type="password"
value={password} placeholder="Введите пароль" onChange={editPassword} />
                             <input className="form__input" type="password"
value={checkPassword}            placeholder="Повторите            пароль"
onChange={checkingPassword} />
                             {password === checkPassword && password.length
>= 5 ?
                                                 <img  width={20}  height={20}
src="/images/edit.png" alt="Edit" onClick={onChangePassword} /> :
                                            <span  className="error">Пароли не
совпадают или указан пароль менее 5 символов</span>}
                        </form> :
                                              <button  className="change-pass"
onClick={onChangeAllowEditPassword}>Изменить пароль</button>
                    }
                                    <Link  to='/'><button  className='button'
onClick={removeActiveCoach}>выйти из профиля</button></Link>
                </div>
                <div className="personal__about">
                    <h3>Обо мне</h3>
                            <textarea  id="about"  type="text"  value={about}
placeholder="Напишите о себе" onChange={editAbout} />
                    {allowEditAbout ?
                                                <img  width={30}  height={30}
src="/images/checked.png" alt="Edit" onClick={onChangeAbout} /> :
                                                <img  width={30}  height={30}
src="/images/edit_big.png" alt="Edit" onClick={onChangeAllowEditAbout} />}
                </div>

            </section>
            <section className="favorites">
                <h3>Мои занятия</h3>
                {myWorks.length > 0 ?
                    <ul className="workouts">
```

```
                                                {myWorks.map(workout  =>  <li
key={workout.id}><WorkoutPreview        workout={workout}        width={200}
height={400} /></li>)}
                    </ul> :
                    <p>У Вас пока нет занятий.</p>}

            </section>
        </div>
    )
}
```

Приложение 30. Компонент <CoachPublicPage />

```
import { useContext } from "react";
import { useParams } from "react-router-dom";
import        {        ProfilesContext,        DataContext        }        from
"../contexts/ContextProvider";
import WorkoutPreview from "../components/WorkoutPreview";

export default function CoachPublicPage() {
    const { workouts } = useContext(DataContext);
    const { profiles } = useContext(ProfilesContext);

    const { id } = useParams();
    const coach = profiles.find(coach => coach.id === id);
      const coachWorkouts = workouts.filter(workout => workout.coachId ===
coach.id);

    return (
        <div className="personal">
            <section className="personal__data">
                <div className="personal__avatar">
                    <img src={coach.avatar || null} alt="Avatar" />
                </div>
                <div className="personal__info">
                    <div className="personal__info-item">
                        <span>Имя</span>
                        <input id="name" type="text" value={coach.name} />
```

```
                    </div>
                    <div className="personal__info-item">
                        <span>Фамилия</span>
                                        <input  id="surname"  type="text"
value={coach.surname} />
                    </div>
                </div>
                <div className="personal__about">
                    <h3>О тренере</h3>
                      <textarea id="about" type="text" value={coach.about}
/>
                </div>
            </section>
            <section className="favorites">
                <h3>Занятия, которые проводит тренер</h3>
                {coachWorkouts.length > 0 ?
                    <ul className="workouts">
                                        {coachWorkouts.map(workout  =>  <li
key={workout.id}><WorkoutPreview        workout={workout}        width={200}
height={400} /></li>)}
                    </ul> :
                    <p>У тренера пока нет занятий.</p>}
            </section>
        </div>
    )
}
```

Приложение 31. Компонент <CoachAboutPage />

```
import { Link } from "react-router-dom";
import { ActiveCoachContext } from "../contexts/ContextProvider";
import { useContext } from "react";

export     default     function     CoachAboutPage({     toggleCoachRegForm,
toggleCoachLoginForm }) {
                const     {     activeCoach,     setActiveCoach     }     =
useContext(ActiveCoachContext);
```

```
    const removeActiveCoach = () => {
        if (activeCoach) {
            localStorage.removeItem('activeCoach');
            setActiveCoach(null);
        }
    };

    return (
        <section className="text">
            {activeCoach ?
                <div className="coach coach__button">
                      <Link to={`/profiles/${activeCoach.id}`}><h3>Страница
тренера: {activeCoach.name} {activeCoach.surname}</h3></Link>
                                                <button  className='button'
onClick={removeActiveCoach}>выйти</button>
                </div> :
                <div className="coach coach__button">
                                                <button  className="button"
onClick={toggleCoachLoginForm}>Войти в личный кабинет</button>
                </div>}

            <div className="coach coach__banner">
                <ul>
                    <li>Удобная площадка</li>
                    <li>Создавайте тренировки по своим методикам</li>
                    <li>Выбирайте любое время для своего расписания</li>
                             <li>Пишите статьи и поднимайте рейтинг своих
занятий</li>
                    <li>Чем больше посещений - тем выше доход</li>
                    <li>Удобный вывод денежных средств</li>
                </ul>
                                <p>Подробная информация указана в <Link
to='/coach_offer'>договоре оферты с тренерами</Link>.</p>
            </div>
                            <div  className="coach  coach__button"><button
className="button"  onClick={toggleCoachRegForm}>Зарегистрируйся  и  стань
тренером!</button></div>
            <div className="coach coach__conditions">
                <h3>Требования к тренерам:</h3>
                <ul>
                    <li>Подтверждение профильного образования</li>
                     <li>Необходимо быть зарегистрированным как самозанятый
или быть индивидуальным предпринимателем.</li>
```

```
                            <li>Наличие технической возможности проводить
онлайн-трансляции (компьютер, видеокамера, микрофон, интернет и т.п.)</li>
                </ul>
            </div>
            <div className="coach coach___todo">
                    <h3>Чтобы стать тренером достаточно сделать несколько
простых шагов:</h3>
                <ul>
                        <li>Зарегистрируйтесь на нашей платформе в качестве
тренера</li>
                    <li>В личном кабинете заполните анкету</li>
                    <li>Укажите свои реквизиты, платежные данные</li>
                            <li>Отправьте нам Ваши документы о профильном
образовании</li>
                            <li>Создайте свою тренировку и укажите в ней
расписание</li>
                </ul>
            </div>
        </section>
    )
}
```

Приложение 32. Компонент <Subscriptions />

```
import { Link } from "react-router-dom";

export default function Subscriptions() {
    return (
        <section className="text">
            <h2>Варианты подписок</h2>
            <ul className="subscriptions">
                <li className="subscriptions__item">
                    <h4>Месяц</h4>
                    <p>199 руб.</p>
                    <Link to='/payment_subscription/monthly'>
                        <button className="button">Подписаться</button>
                    </Link>

                </li>
```

```
                <li className="subscriptions__item">
                    <h4>Полгода</h4>
                    <p>999 руб.</p>
                    <Link to='/payment_subscription/half_yearly'>
                        <button className="button">Подписаться</button>
                    </Link>
                </li>
                <li className="subscriptions__item">
                    <h4>Год</h4>
                    <p>1899 руб.</p>
                    <Link to='/payment_subscription/yearly'>
                        <button className="button">Подписаться</button>
                    </Link>
                </li>
            </ul>
        </section>
    )
}
```

Приложение 33. Компонент <PaymentPage />

```
import { Link, useParams } from "react-router-dom";

export default function PaymentPage() {
    const { subscribeType } = useParams();
    let price = 0;
    let type = '';
    switch (subscribeType) {
        case 'monthly':
            price = 199;
            type = 'Ежемесячная';
            break;
        case 'half_yearly':
            price = 999;
            type = 'Полугодовая';
            break;
        case 'yearly':
```

```
            price = 1899
            type = 'Годовая';
            break;
        default:
            break;
    }

    return (
        <section className="text">
            <h2>Вы выбрали подписку "{type}"!</h2>
            <h3>Стоимость подписки: {price} руб.</h3>
                        <p>Перед оплатой просьба ознакомиться с <Link
to='/user_offer'>договором оферты</Link>.</p>
            <form className="form" action="#">
                <span>Сумма к оплате: </span>
                  <input className="form__input" type="text" value={price}
/>
                <p>
                    <input type="checkbox" name="agree" id="agree" />
                     <label htmlFor="agree">Я согласен с условиями договора
оферты.</label>
                </p>
                <button className="button">Оплатить</button>
            </form>
                                        <h3>Вернуться к выбору <Link
to='/subscriptions'>подписки</Link>.</h3>
        </section>
    )
}
```

Приложение 34. Компонент <About />

```
import { Link } from 'react-router-dom';

export default function About() {
    return (
        <section className='text'>
```

```
            <p>Меня зовут Мария Савкина, мне 39 лет и я влюблена в фитнес и здоровый образ жизни. Эта любовь сподвигла меня на создание этой платформы.</p>
                <p>Идея подобного сайта зародилась у меня давно. Еще в то время, когда я только решилась на то, чтобы начать заниматься фитнесом. Я смотрела ролики, читала статьи про фитнес, но, при этом, чувствовала себя крайне неуютно от того, что мне не хватало общения с единомышленниками, не хватало понимания того, у кого можно спросить житейского и профессионального совета, с кем поделиться своими мыслями и страхами.</p>

            <p> На просторах интернета я не могла найти то, что мне нужно и тогда подумала, что в мире очень не хватает своего интернет-комьюнити. Позже я начала ходить в фитнес-клуб, затем вернулась к домашним тренировкам, которые делаю по тем крупицам информации, которые нахожу в интернете.</p>

                <p>Платформа FitSteps.ru призвана помочь решить все эти задачи. Тут есть три основных раздела:</p>
            <ul>
                <li><Link to='/workouts'>Онлайн-занятия</Link> по фитнесу, йоге, растяжке, силовым нагрузкам, джумбе и прочее. Здесь по подписке можно посещать онлайн-тренировки с живыми специалистами, которым после занятия можно задать вопросы.</li>
                <li><Link to='/streams'>Вебинары.</Link> В данном разделе можно абсолютно бесплатно посещать бесплатные вебинары с обсуждениями фитнеса и питания.</li>
                <li><Link to='/articles'>Статьи.</Link> Тут можно почитать полезные советы и другую интересную информацию, которую публикуют наши тренеры.</li>
            </ul>

            <p>Пока мы только начинаем и развиваем наше дружное комьюнити, но очень надеемся, что нас будет все больше и больше и мы сможем неограниченно делится своим опытом о том, как быть красивыми, стройными и здоровыми</p>
        </section>
    )
}
```

Приложение 35. Компонент <Contacts />

```
export default function Contacts() {
    return (
        <section className='text'>
            <h3>Уважаемые посетители сайта!</h3>
                <p>По любым вопросам Вы всегда можете обратиться к нам по телефону, электронной почте, либо оставить сообщение, заполнив форму ниже.</p>
            <p>+7 800 000-00-00</p>
            <p>info@fitsteps.ru</p>
            <form className='form'>
                        <input className='form__input' type='text' required placeholder='Ваше имя' />
                    <input className='form__input' type='email' required placeholder='Ваша эл.почта' />
                            <input  className='form__input'  type='number' placeholder='Ваш телефон' />
                    <textarea  className='form__input' placeholder='Введите Ваше сообщение'></textarea>
                <button className='button'>Отправить</button>
            </form>
        </section>
    )
}
```

Приложение 36 Компонент <CoachOffer />

```
export default function CoachOffer() {
    return (
        <section className="text">
            <p>В данном разделе будет опубликована Оферта с тренером.</p>
        </section>
    )
}
```

Приложение 37. Компонент <UserOffer />

```
export default function UserOffer() {
    return (
        <section className="text">
            <p>В данном разделе будет опубликована Оферта с
пользователем.</p>
        </section>
    )
}
```

Приложение 38. Компонент <ConfidentialPolicy />

```
export default function ConfidentialPolicy() {
    return (
        <section className="text">
            <h2>Политика конфиденциальности сайта FitSteps.ru</h2>

            <h3>1. Введение</h3>
                <p>FitSteps.ru (далее — Сайт) стремится обеспечить защиту
персональных данных своих пользователей и соблюдает требования
законодательства Российской Федерации в области защиты персональных
данных. Настоящая Политика конфиденциальности (далее — Политика)
устанавливает правила сбора, хранения, обработки и использования
персональных данных пользователей Сайта.</p>

            <h3>2. Сбор и использование персональных данных</h3>
            <p>Сайт может собирать следующие виды персональных данных:</p>
            <ul>
                <li>фамилия, имя, отчество;</li>
                <li>дата рождения;</li>
                <li>пол;</li>
                <li>место жительства;</li>
                <li>контактный телефон;</li>
                <li>адрес электронной почты;</li>
                <li>история посещений Сайта;</li>
                <li>данные о совершенных покупках и заказах;</li>
```

```
                    <li>другая информация, предоставляемая пользователями
добровольно.</li>
            </ul>
               <p>Сбор персональных данных осуществляется путем заполнения
форм на Сайте, предоставления информации при регистрации, оформлении
заказов и обратной связи с администрацией Сайта.</p>

            <h3>3. Обработка и хранение персональных данных</h3>
             <p>Персональные данные пользователей обрабатываются и хранятся
на серверах, расположенных на территории Российской Федерации. Сайт
принимает необходимые меры для защиты персональных данных от
несанкционированного доступа, изменения и уничтожения.</p>

            <h3>4. Передача персональных данных</h3>
              <p>Персональные данные могут быть переданы следующим третьим
лицам:</p>
            <ul>
                     <li>партнерам и контрагентам, оказывающим услуги по
обработке заказов, доставке товаров, технической поддержке и другим
услугам, связанным с функционированием Сайта;</li>
                   <li>государственным органам и службам в соответствии с
законодательством Российской Федерации;</li>
                  <li>рекламодателям и рекламным агентствам для проведения
рекламных кампаний и анализа эффективности рекламы.</li>
            </ul>

            <h3>5. Права пользователей</h3>
            <p>Пользователи имеют право:</p>
            <ul>
                     <li>получать информацию о своих персональных данных,
хранящихся на Сайте;</li>
                  <li>требовать исправления или удаления своих персональных
данных в случае их недостоверности или неактуальности;</li>
                  <li>возражать против обработки своих персональных данных,
если это не предусмотрено законодательством Российской Федерации или если
обработка данных нарушает права и свободы пользователя.</li>
            </ul>

            <h3>6. Изменения в Политике</h3>
                  <p>Администрация Сайта оставляет за собой право вносить
изменения в Политику конфиденциальности по мере необходимости. Новая
редакция Политики размещается на Сайте и вступает в силу с момента её
публикации.</p>
```

```
        </section>
    )
}
```

Приложение 39. Компонент <UserAgreement />

```
export default function UserAgreement() {
    return (
        <section className="text">
            <h2>Пользовательское соглашение сайта FitSteps.ru</h2>
            <h3>1. Общие положения</h3>
                 <p> 1.1. Настоящее пользовательское соглашение (далее — Соглашение) регулирует отношения между пользователем и сайтом FitSteps.ru (далее — Сайт).</p>
              <p>1.2. Используя Сайт, пользователь соглашается с условиями Соглашения и обязуется их соблюдать.</p>
            <h3>2. Предоставление информации</h3>
              <p>2.1. Сайт предоставляет пользователю информацию о здоровом образе жизни, физической активности и питании.</p>
               <p>2.2. Информация на Сайте предоставляется исключительно в информационных целях и не должна рассматриваться как медицинская консультация или рекомендация.</p>
            <h3>3. Конфиденциальность</h3>
                 <p>3.1. Пользователь гарантирует, что предоставленная им информация является достоверной и актуальной.</p>
               <p>3.2. Сайт сохраняет конфиденциальность личной информации пользователя, включая его имя, адрес электронной почты и другую информацию, предоставленную добровольно.</p>
            <h3>4. Ответственность</h3>
             <p>4.1. Пользователь несёт ответственность за свои действия на Сайте и соблюдение законодательства Российской Федерации.</p>
              <p>4.2. Сайт не несёт ответственности за возможные убытки или ущерб, возникшие в результате использования или неиспользования информации, предоставленной на Сайте.</p>
            <h3>5. Изменение Соглашения</h3>
                <p>5.1. Сайт имеет право изменять условия Соглашения без предварительного уведомления пользователя.</p>
            <h3>6. Разрешение споров</h3>
```

```
                    <p>6.1. Все споры и разногласия, возникающие между пользователем и Сайтом, решаются путём переговоров.</p>
                    <p>6.2. В случае невозможности достижения согласия спор передаётся на рассмотрение суда общей юрисдикции по месту нахождения Сайта.</p>
                <h3>7. Заключительные положения</h3>
                    <p>7.1. Настоящее Соглашение вступает в силу с момента его принятия пользователем и действует бессрочно.</p>
                    <p>7.2. Соглашение составлено в двух экземплярах, имеющих одинаковую юридическую силу, по одному для каждой из сторон.</p>
            </section>
        )
}
```

Приложение 40. Компонент <ErrorPage />

```
export default function ErrorPage() {
    return (
        <section className="text">
            <p>Ошибка 404. Страница не найдена.</p>
        </section>
    )
}
```

Приложение 41. Компонент <App />

```
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { ContextProvider } from './contexts/ContextProvider';
import { useState } from 'react';

import Header from './components/Header';
import Submenu from './components/Submenu';
import Subheader from './components/Subheader';
import Footer from './components/Footer';
```

```
import Home from './pages/Home';
import Workouts from './pages/Workouts';
import WorkoutDetails from './pages/WorkoutDetails';
import Streams from './pages/Streams';
import StreamsDetails from './pages/StreamsDetails';
import Articles from './pages/Articles';
import ArticleDetails from './pages/ArticleDetails';

import About from './pages/About';
import Contacts from './pages/Contacts';
import Subscriptions from './pages/Subscriptions';
import UserAgreement from './pages/UserAgreement';
import UserOffer from './pages/UserOffer';
import CoachOffer from './pages/CoachOffer';
import ConfidentialPolicy from './pages/ConfidentialPolicy';
import ErrorPage from './pages/ErrorPage404';
import PaymentPage from './pages/PaymentPage';
import CoachAboutPage from './pages/CoachAboutPage';
import CoachPublicPage from './pages/CoachPublicPage';
import RegistrationForm from './components/RegistrationForm';
import LoginForm from './components/LoginForm';
import UserAccount from './pages/UserAccount';
import CoachAccount from './pages/CoachAccount';


function App() {
  const [coachRegFormVisibility, setCoachRegFormVisibility] =
useState(false);
  const toggleCoachRegForm = () => {
    setCoachRegFormVisibility(!coachRegFormVisibility);
  }

  const [regFormVisibility, setRegFormVisibility] = useState(false);
  const toggleUserRegForm = () => {
    setRegFormVisibility(!regFormVisibility);
  };

  const [loginFormVisibility, setLoginFormVisibility] = useState(false);
  const toggleUserLoginForm = () => {
    setLoginFormVisibility(!loginFormVisibility);
  };
```

```
  const [coachLoginFormVisibility, setCoachLoginFormVisibility] =
useState(false);
  const toggleCoachLoginForm = () => {
    setCoachLoginFormVisibility(!coachLoginFormVisibility);
  };

  const closeForm = (e) => {
    if (!e.target.classList.contains('reg')) {
      if (coachRegFormVisibility) {
        toggleCoachRegForm();
      }
      if (regFormVisibility) {
        toggleUserRegForm();
      }
      if (loginFormVisibility) {
        toggleUserLoginForm();
      }
      if (coachLoginFormVisibility) {
        setCoachLoginFormVisibility();
      }
      document.querySelector('.App').classList.remove('formVisibility');
    }
  }

  return (
    <ContextProvider>
      <Router>
        <div className="App" onClick={closeForm}>
          <Header toggleUserRegForm={toggleUserRegForm}
toggleUserLoginForm={toggleUserLoginForm} />
          <Submenu />
          <Subheader />
          <main>
            <Routes>
              <Route path='/' element={<Home />} />
              <Route path='/workouts' element={<Workouts />} />
              <Route path='/workouts/:typePath' element={<Workouts />} />
              <Route path='/workouts/:typePath/:id'
element={<WorkoutDetails />} />
              <Route path='/streams' element={<Streams />} />
              <Route path='/streams/:id' element={<StreamsDetails />} />
              <Route path='/articles' element={<Articles />} />
              <Route path='/articles/:id' element={<ArticleDetails />} />
```

```
              <Route path='/about' element={<About />} />
              <Route path='/contacts' element={<Contacts />} />
              <Route path='/subscriptions' element={<Subscriptions />} />
              <Route path='/coach_about' element={<CoachAboutPage
toggleCoachRegForm={toggleCoachRegForm}
toggleCoachLoginForm={toggleCoachLoginForm} />} />
              <Route path='/users/:id' element={<UserAccount />} />
              <Route path='/coaches/:id' element={<CoachPublicPage />} />
              <Route path='/profiles/:id' element={<CoachAccount />} />
              <Route path='/user_offer' element={<UserOffer />} />
              <Route path='/coach_offer' element={<CoachOffer />} />
              <Route path='/user_agreement' element={<UserAgreement />} />
              <Route path='/confidential_policy'
element={<ConfidentialPolicy />} />
              <Route path='/payment_subscription/:subscribeType'
element={<PaymentPage />} />
              <Route path='*' element={<ErrorPage />} />
            </Routes>
          </main>
          <Footer />
          {(loginFormVisibility || coachRegFormVisibility ||
regFormVisibility || coachLoginFormVisibility) && (<div
className='overlay'></div>)}
        </div>
        {coachRegFormVisibility && <RegistrationForm role='coach'
closeRegForm={toggleCoachRegForm} />}
        {regFormVisibility && <RegistrationForm role='user'
closeRegForm={toggleUserRegForm} />}
        {loginFormVisibility && <LoginForm role='user'
closeLoginForm={toggleUserLoginForm} />}
        {coachLoginFormVisibility && <LoginForm role='coach'
closeLoginForm={toggleCoachLoginForm} />}
      </Router>
    </ContextProvider >
  );
}

export default App;
```

Приложение 42. Компонент <Index />

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.scss';
import App from './App';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Приложение 43. Функции

```
const isFormCorrect = (elements) => {
       return  Array.from(elements).filter(element  =>  element.required  &&
element.value === '').length === 0;
};

const createProfile = (elements, role, profiles) => {
    let userData = {
        id: Math.max(...profiles.map(profile => +profile.id)) + 1,
        role: role,
        avatar: '/images/default_avatar.webp',
        about: 'Напишите немного о себе'
    };
    Array.from(elements).forEach(element => {
        if (element.name) {
            userData[element.name] = element.value
        };
    });
    if (role === 'user') {
        userData = {
            ...userData,
```

```
            favorites: [],
            subscribe: {
                isActive: false,
                type: null,
                startDate: null,
                endDate: null
            }
        }
    }
    if (role === 'coach') {
        userData = {
            ...userData,
            balance: 500
        }
    }
    return userData;
};

const getProfileDataFromForm = (elements) => {
    let userData = {};
    Array.from(elements).forEach(element => {
        if (element.name) {
            userData[element.name] = element.value
        };
    });
    return userData;
};

const findProfile = (profiles, email, role) => {
    return profiles
        .filter(currProfile => currProfile.role === role)
        .find(currProfile => currProfile.email === email);
};

const isPasswordCorrect = (profile, password) => {
    return profile.password === password;
};

export     {     isFormCorrect,     getProfileDataFromForm,     findProfile,
isPasswordCorrect, createProfile };

const weekDays = {
    1: 'Понедельник',
```

```
    2: 'Вторник',
    3: 'Среда',
    4: 'Четверг',
    5: 'Пятница',
    6: 'Суббота',
    7: 'Воскресенье'
};

const getTypePath = (workout, types) => {
    return types.find(currType => currType.name === workout.type).path;
}

const getScheduleHTML = (workout) => {
    const scheduleSplit = workout.schedule
        .split(',')
        .map(day => day.split('-'));

    const scheduleItem = scheduleSplit.map((day, index) => {
        const weekDay = weekDays[parseInt(day[0])];
        const startWorkout = day[1];
        return <li key={index}>{weekDay}: {startWorkout}</li>
    })

    return scheduleItem;
}

const getCoach = (dataItem, profiles) => {
    return profiles.find(profile => profile.id === dataItem.coachId);
}

const getDescription = (workout, types) => {
    return types.find(type => type.name === workout.type).description;
}

const getClosestStreamDate = (workout) => {
    const today = new Date().getDay();
    const scheduleSplit = workout.schedule
        .split(',')
        .map(day => day.split('-'));
    let diff = 100;
    scheduleSplit.forEach(currDay => {
        let currDiff = currDay[0] - today;
        if (currDiff < 0) {
```

```
            currDiff += 7;
        }
        if (currDiff < diff) {
            diff = currDiff;
        }
    });

    const streamDay = new Date(Date.now() + diff * 24 * 3600 * 1000);
    const month = streamDay.getMonth() + 1;

     const stringStreamDay = `${streamDay.getDate()}.${month < 10 ? '0' +
month.toString() : month}.${streamDay.getFullYear()}`

    return stringStreamDay;
}

const isFavorite = (activeUser, workoutId) => {
    if (activeUser) {
        if (activeUser.favorites.includes(+workoutId)) {
            return true;
        }
    }
    return false;
}

const renderFavorite = (activeUser, workoutId) => {
         return  <img  width={30}  height={30}  src={isFavorite(activeUser,
workoutId)    ?    '/images/favorite.svg'    :    '/images/unfavorite.svg'}
alt="Добавить или удалить из избранного" />
}

const toggleFavorite = (activeUser, workoutId) => {
    const editedUser = activeUser;
    if (activeUser) {
        let newFavorites = [];
        if (isFavorite(activeUser, workoutId)) {
                    newFavorites  =  activeUser.favorites.filter(favorite  =>
favorite !== +workoutId);
        } else {
            newFavorites = [...activeUser.favorites, +workoutId]
        }
        editedUser.favorites = newFavorites;
    }
```

```
    return editedUser;
}

export { getTypePath, getScheduleHTML, getCoach, getDescription,
getClosestStreamDate, renderFavorite, toggleFavorite };
```