Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Отчет по лабораторной работе №4

по дисциплине «Программирование на JavaScript»

«Знакомство с React»

Преподаватель Сайчик Е.Д.

Студент гр. РИМ-240950 Зверев А. Д.

Екатеринбург

2025

Цель работы: ознакомиться с библиотекой React и способами создания приложения с ее использованием

Задачи:

1. Ознакомиться с процессом создания приложения на React
2. Ознакомиться с концепциями CSR / SSR на теоретическом уровне
3. Ознакомиться с возможными вариантами создания приложения на React

Ознакомиться с процессом создания приложения на React

Чистый React – это библиотека для JS, позволяющая в удобной форме создавать компоненты страницы и эффективно взаимодействовать с DOM браузера.

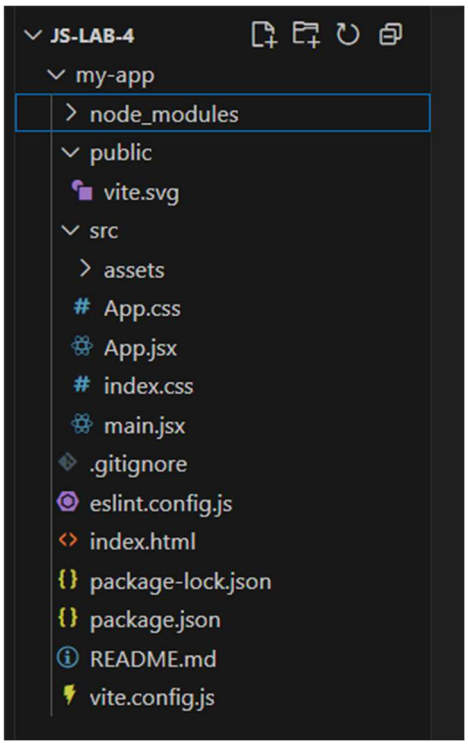
Он позволяет описывать компоненты в синтаксисе, похожем на HTML – JSX. Но важно понимать, что код JSX в последствии конвертируется в обычный JS, например, через babel. То есть все то же самое можно написать и через JS, используя api браузера, но это будет не так просто.

Для разработки с использованием React можно выбрать два варианта – чистый React сборщиком, например Vite, или использование фреймворков на основе React - App Router в Next. Js, React Router (фреймворки уже подразумевают использование сборщиков).

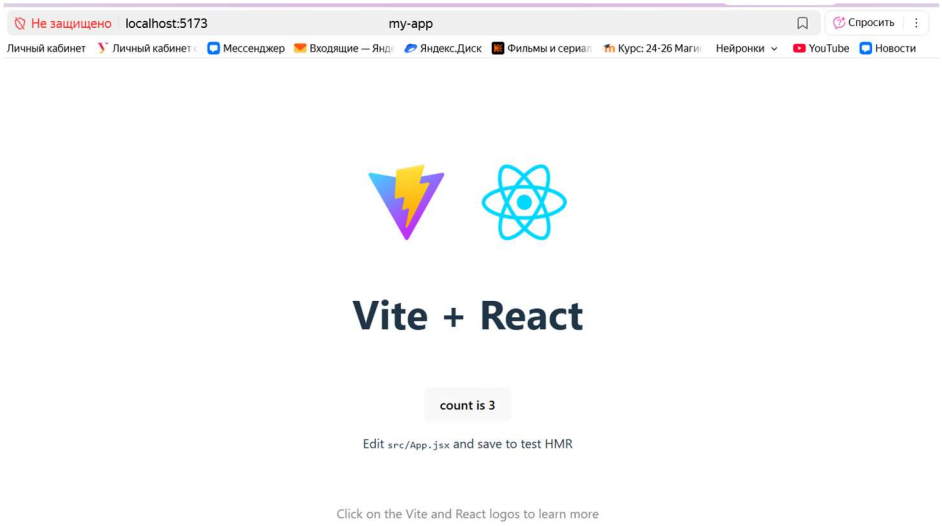
Для первого способа нужно выполнить команду npm create vite@latest my-app -- --template react:

- npm create vite@latest — это способ вызвать пакет create-vite для создания нового проекта с помощью Vite. Ключевое слово create используется с npm, чтобы инициализировать проект из шаблона, а @latest указывает использовать последнюю доступную версию инструмента.
- my-app — имя создаваемой папки/проекта, куда Vite разместит начальные файлы.
- -- --template react — указывает, что проект нужно создать на шаблоне React, то есть с предустановленной конфигурацией и файлами, необходимыми для React-приложения.

В итоге после небольшого опроса от vite получим следующую структуру проекта, который уже можно запустить командой npm run dev. Vite подготовил для нас множество файлов для конфигурации проекта с начальными настройками в соответствии с лучшими практиками сообщества. Дальше мы можем дополнять проект своими компонентами, зависимостями и прочим, что понадобится в работе.

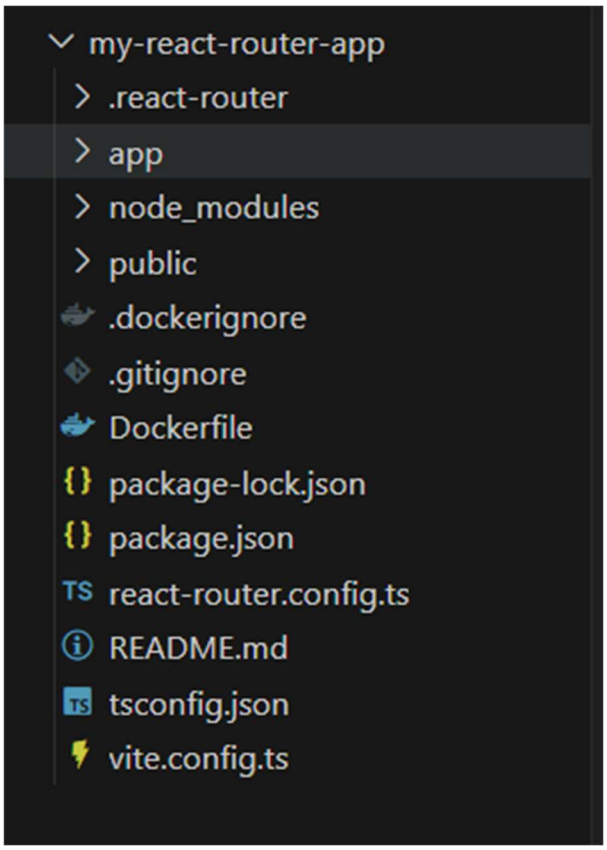


По адресу http://localhost:5173/ увидим следующий демо-проект



Второй способ создания приложения – использование фреймворков, например React Router. Создать приложение можно командой npx create-react-router@latest my-react-router-app

Получим следующую структуру проекта. Напоминает предыдущую, но содержит дополнительные файлы. Например, добавлен файл конфигурации для TS, определен базовый dockerfile для контейнеризации, а также появились файлы для определения путей в нашем приложении



React Router организует маршруты в приложении и контролирует, какой компонент показывать пользователю при изменении адреса браузера, обеспечивая плавный пользовательский опыт и быстрый переход между "страницами" без фактической загрузки новых HTML-документов с сервера.

Подходы к рендеру приложения CSR и SSR

CSR – Client-Side Rendering. Подход при котором отрисовка приложения происходит на стороне клиента, то есть в веб-браузере пользователя. Осуществляется это за счет того, что сервер передает пользователю по сети только минимальный html документ и исполняемый код JS, который исполняется браузером пользователя и наполняет html всем нужным контентом.

Преимущества CSR	Недостатки CSR
Интерактивность и отзывчивость – Переключение между страницами без перезагрузки – Используется "виртуальный DOM", что ускоряет обновления Минимальная нагрузка на сервер – Сервер отдает только статические файлы Современная архитектура – Удобно разделять фронтенд и бэкенд (через API)	Медленный первый рендер – Пользователь видит пустой экран, пока JS не загрузится и не выполнится Проблемы с SEO – Поисковые боты могут не дожидаться выполнения JS – Метаданные (title, description) генерируются динамически Большой бандл JS – Чем больше логики на клиенте, тем больше трафика

Server-Side Rendering — это способ, при котором сервер формирует полноценную HTML-страницу до того, как она отправляется пользователю. Браузер получает готовый контент, который виден сразу, а потом подключается JS для интерактивности.

Логика работы следующая – пользователь делает запрос, сервер запускает JS-приложение (требуется сервер для исполнения JS, что не нужно в случае с CSR), формируется заполненный html документ и отправляется клиенту

Преимущества SSR	Недостатки SSR
Быстрый первый рендер – Пользователь сразу видит контент SEO-оптимизация – HTML уже содержит нужный контент, мета-теги и данные Улучшенный UX – Первая отрисовка — мгновенная, без «пустого экрана»	Повышенная нагрузка на сервер – Каждая страница требует вычислений и сборки HTML Сложность архитектуры – Нужно согласовать клиентскую и серверную логику Возможные задержки при навигации – Если не настроен кеш или частичный рендеринг

Ссылка на GitHub

https://github.com/UnderAlex59/JS-Lab-4

Выводы

В ходе выполнения ЛР ознакомились с основными концепциями библиотеки React, технологиями для сборки приложений, а также фреймворками в основе которых используется React.

С помощью встроенных в vite и React Router генераторов проектов создали два примера «пустых» приложений и сравнили их стандартное наполнение.