САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Жаров Александр Павлович

Группа:
K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

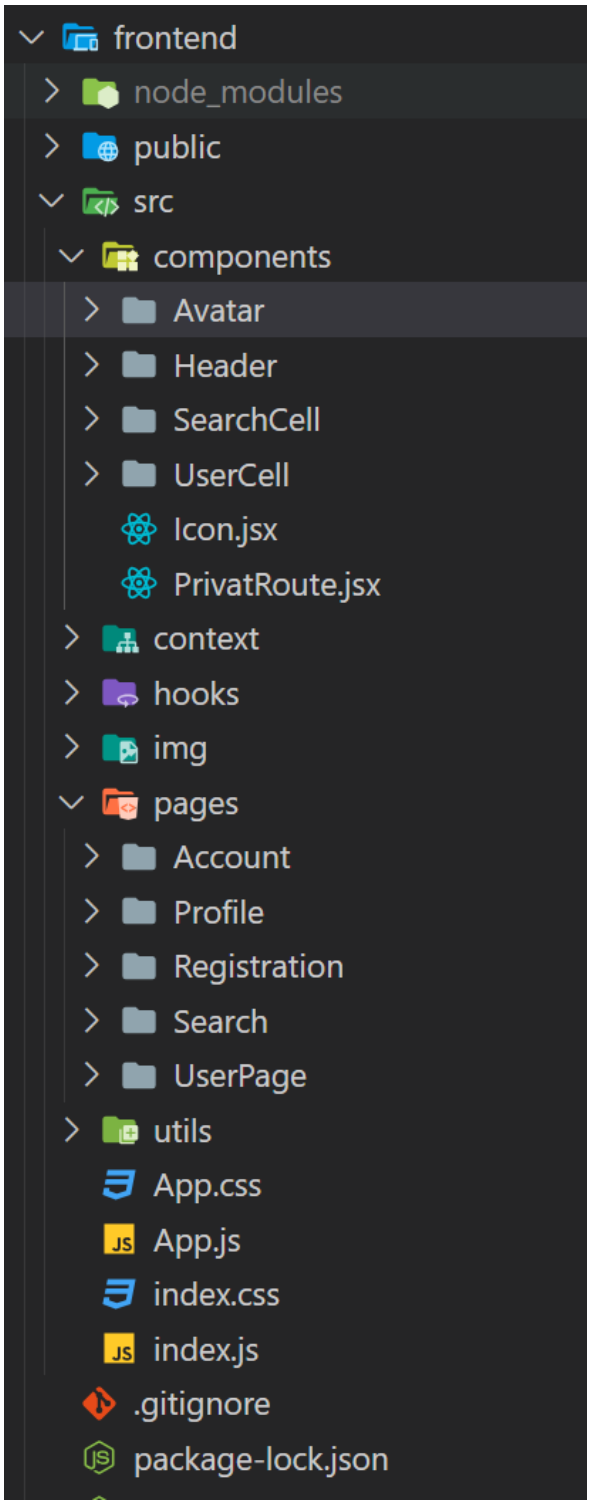
2023 г.

Задача

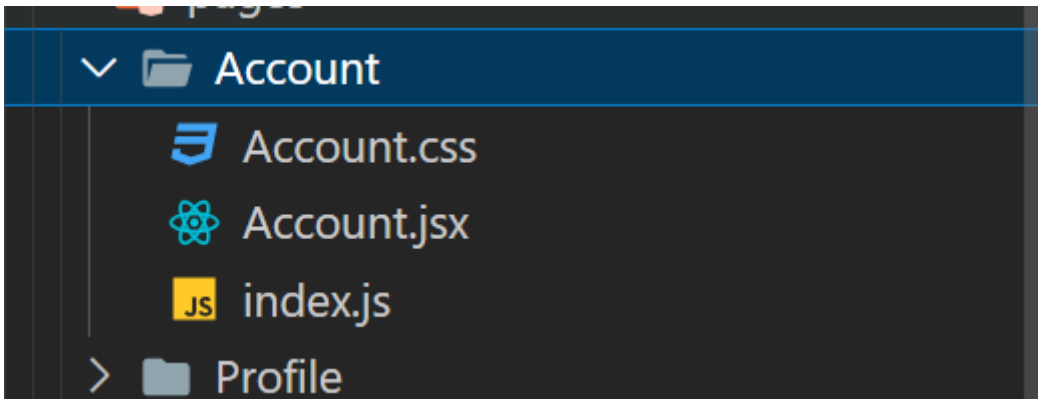
Создать одностраничный сайт на React

Ход работы

Создадим шаблон приложения и продумаем разделение на компоненты



Компоненты, содержащие код страницы располагаем в pages, реиспользуемые более мелкие компоненты положим в components



В каждой папке с компонентом располагается код с самим компонентом .jsx, стили для него .css и index.js для удобного импорта компонента.

Для роутинга используем библиотеку react touter. А для авторизации напишем собственный кастомный защищенный роут.

```jsx
import { Navigate, Outlet } from "react-router-dom";
import useAuth from "../hooks/useAuth";
import Registration from "../pages/Registration";

export const PrivateRoute = () => {
  const { isAuthenticated } = useAuth();

  return isAuthenticated ? (
    <Outlet />
  ) : (
    <Navigate
      to="/registration/registration"
      element={<Registration />}
      replase
    />
  );
};

```

Здесь мы проверяем авторизован ли пользователь и если он не авторизован редиректим его на страницу регистрации скрывая основные роуты.

В противном случае разрешаем пользователю доступ к основным страницам.

```
  return (
    <div className="App">
      {!isHeaderDisplayNone && <Header />}
      <Routes>
        <Route path="/">
          <Route path="/registration/:isLogin" element={<Registration />} />

          <Route element={<PrivateRoute />}>
            <Route index element={<Profile />} />
            <Route path="/account" element={<Account />} />
            <Route path="/search/:query" element={<Search />} />
            <Route path="/user/:id" element={<UserPage />} />
          </Route>
        </Route>
      </Routes>
    </div>
  );
}
```

Применяем наш PrivatRoute.

Далее необходимо реализовать взаимодействие с внешним api. Для этого используем встроенную библиотеку fetch.

```
export async function fetchAuthUser(token) {
  const res = await fetch("http://localhost:3030/api/get-user", {
    method: "GET",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    },
  }).then(async (response) => {
    const data = await response.json();
    if (response.status === 403) {
      throw new Error(data.message);
    }
    return data;
  });

  return res;
}
```

Вот пример запроса на получения информации о пользователе при авторизации. В хедер передаем токен и обрабатываем ответ или ошибку с сервера.

```
React.useEffect(() => {
  const token = getCookie("token");
  if (token && !isAuthenticated) {
    navigate("/");
    fetchAuthUser(token)
      .then((user) => {
        onAuth(user);
      })
      .catch((error) => {
        console.log(error);
        navigate("/registration/login");
        onLogout();
      });
  }
}, [isAuthenticated, setAuth, setUser, onAuth, onLogout, navigate]);
```

Вот так выглядит выполнение предыдущего запроса в коде, передаем в функцию токен и передаем переменную user в функцию авторизации.

Вывод:

В ходе работы мы создали одностраничный сайт, разделили код на компоненты, настроили роутинг, а также реализовали взаимодействие с внешним api.