САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Екушев Владислав

Группа К32402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Нужно привязать то, что Вы делали в ЛР1 к внешнему API средствами fetch/axios/xhr

Вариант: Разработка интерактивного сайта для управления умным домом

Ход работы

Была установлена библиотека vite-plugin-mock-server для того, чтобы сымитировать работу сервера.

```ts
import { MockHandler } from 'vite-plugin-mock-server'
import { devices } from './devices'
import { Device } from 'src/types/Device'
import { User } from 'src/types/User'
import { UserGetSelfRes, UserLoginReq, UserLoginRes } from
import type { DeviceGetRes, FavoriteDeviceReq } from 'src/types/api'

const API_DELAY = 500

const DEFAULT_USER: User = {
  id: '1',
  devices,
  avatarUrl: '/avatar.png',
  email: 'v.ekushev@vk.team',
  fullname: 'Владислав Екушев',
  password: Buffer.from('nobodyknowsthis').toString('base64'),
}

let user: User | null = null

const mocks: MockHandler[] = [
  {
    pattern: '/api/user',
    handle: (req, res) => {
      if (!user) {
        res.writeHead(401, { 'Content-Type': 'application/json' })
        return setTimeout(
          () => res.end(JSON.stringify({ user: null })),
          API_DELAY
        )
      }

      const result: UserGetSelfRes = {
        user: {
          avatarUrl: user.avatarUrl,
          id: user.id,
          email: user.email,
          fullname: user.fullname,
        },
      }

      res.writeHead(200, { 'Content-Type': 'application/json' })
      setTimeout(() => res.end(JSON.stringify(result)), API_DELAY)
    },
  },
  ...
]

export default mocks
```

Для хранения состояния выбрал redux и RTK Query, на скриншоте представлен код для стора с авторизацией. В нём создаём асинхронный thunk для получения данных о юзере. Если данных не вернётся, приложение откроет страницу авторизации.

```ts
import { createAsyncThunk, createSlice, PayloadAction } from '@reduxjs/toolkit'
import { ApiUser, UserGetSelfRes } from 'src/types/api'
import { FetchingState } from 'src/types/FetchingState'
import axios from 'axios'
import { API_URL } from 'src/config/constants'

type AuthState = {
  user: ApiUser | null
  state: FetchingState
  currentRequestId: string | null
}

const initialState: AuthState = {
  user: null,
  state: FetchingState.IDLE,
  currentRequestId: null,
}

export const fetchUserSelf = createAsyncThunk<
  ApiUser | null,
  void,
  { state: { auth: AuthState } }
>('auth/getUserSelf', async (_, { getState, requestId }) => {
  const { state, currentRequestId } = getState().auth

  if (state !== FetchingState.PENDING || requestId !== currentRequestId) {
    return null
  }

  const response = await axios<UserGetSelfRes>({
    url: API_URL + '/user',
    method: 'GET',
  })

  return response.data.user
})

const slice = createSlice({
  name: 'auth',
  initialState,
  reducers: {
    setUser: (state, { payload }: PayloadAction<ApiUser | null>) => {
      state.user = payload
    },
    setUserFetchingState: (
      state,
      { payload }: PayloadAction<FetchingState>
    ) => {
      state.state = payload
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(fetchUserSelf.pending, (state, action) => {
        if (state.state === FetchingState.IDLE) {
          state.state = FetchingState.PENDING
          state.currentRequestId = action.meta.requestId
        }
      })
      .addCase(fetchUserSelf.fulfilled, (state, action) => {
        const { requestId } = action.meta
        if (
          state.state === FetchingState.PENDING &&
          state.currentRequestId === requestId
        ) {
          state.state = FetchingState.FULFILLED
          state.user = action.payload || null
          state.currentRequestId = null
        }
      })
      .addCase(fetchUserSelf.rejected, (state, action) => {
        const { requestId } = action.meta
        if (
          state.state === FetchingState.PENDING &&
          state.currentRequestId === requestId
        ) {
          state.state = FetchingState.REJECTED
          state.currentRequestId = null
          state.user = null
        }
      })
  },
})

export const { setUser, setUserFetchingState } = slice.actions

export default slice.reducer
```

```tsx
React.useEffect(() => {
  // Unauthorized state, show login page
  if (
    (userFetchState === FetchingState.FULFILLED && !user) ||
    userFetchState === FetchingState.REJECTED
  ) {
    navigate(getRouteByAlias('login').path)
  }
}, [navigate, user, userFetchState])
```

В файле src/api/index.ts находится слайс rtk-query. В нём находятся запросы на получение и изменение данных

```tsx
import {
  DevicesGetRes,
  DevicesGetReq,
  FavoriteDeviceReq,
  FavoriteDeviceRes,
} from 'src/types/api'
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'
import { API_URL } from 'src/config/constants'
import { UserLoginReq, UserLoginRes } from 'src/types/api/User'

const baseQuery = fetchBaseQuery({
  baseUrl: API_URL,
})

export const apiSlice = createApi({
  baseQuery,
  tagTypes: ['Device', 'User'],
  endpoints: (builder) => ({
    login: builder.mutation<UserLoginRes, UserLoginReq>({
      query: (body) => ({
        url: '/login',
        method: 'POST',
        body,
      }),
      invalidatesTags: (result, _error, _arg) =>
        result?.success ? [{ type: 'User', id: result.user.id }] : [],
    }),
    getDevices: builder.query<DevicesGetRes, DevicesGetReq>({ ···
    }),
    getFavoritesDevices: builder.query<DevicesGetRes, DevicesGetReq>({ ···
    }),
    toggleFavoriteDevice: builder.mutation<
      FavoriteDeviceRes,
      FavoriteDeviceReq
    >({ ···
    }),
  }),
})

export const {
  useLoginMutation,
  useGetDevicesQuery,
  useGetFavoritesDevicesQuery,
  useToggleFavoriteDeviceMutation,
} = apiSlice
```

Пример запроса на получение данных в коде (получение списка девайсов):

```
45  const Favorites: React.FC = () => {
46    const navigate = useNavigate()
47    const { data, isSuccess, isLoading } = useGetFavoritesDevicesQuery({})
48    const devices = useMemo(
49      () => (isSuccess ? data?.devices : []),
50      [data?.devices, isSuccess]
51    )
```

Пример запроса на изменение в коде (добавление девайса в избранное):

```
const toggleFavoriteDevice = () => {
  if (!id) return

  favoriteToggle({ id })
    .unwrap()
    .then((data) => {
      setIsFavorite(data.state)
    })
}
```

Вывод

В ходе лабораторной работы был реализован стор для хранения состояния приложения через Redux и RTK Query, был реализован mock-сервер с тестовыми данными и тестовым API. Была реализована логика авторизации, получения данных о девайсах и изменение девайса (добавление в избранное).