

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО»**

Отчет
по лабораторной работе №2
по дисциплине «**Фронт-энд разработка**»

Автор: Шляхов Денис Олегович

Факультет: ИКТ

Группа: К33402

Преподаватель: Добряков Д.И.

ИТМО

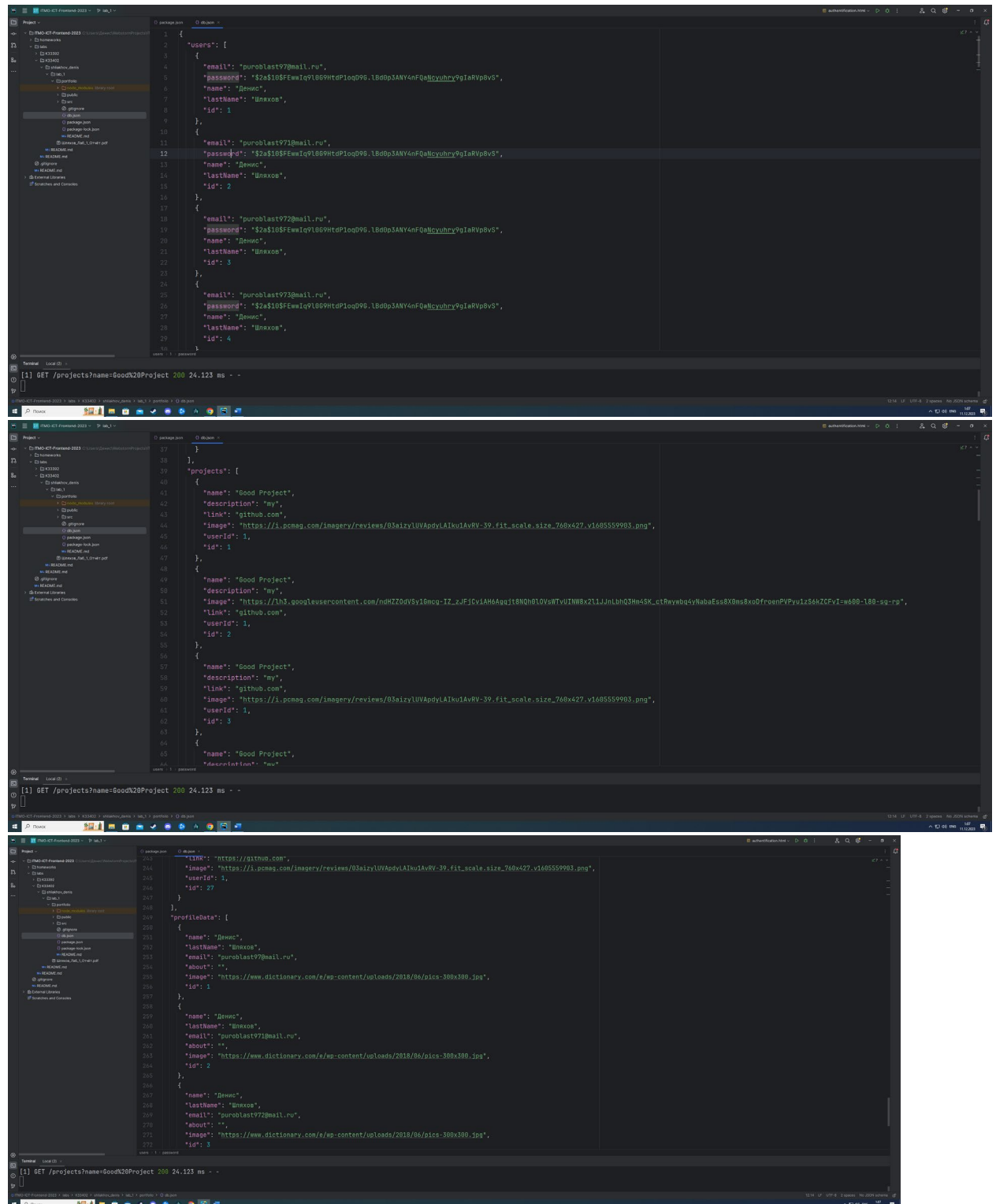
Санкт-Петербург 2023

Задача :

Расширить функционал к первой лабораторной. Связать все данные с бд и добавить запросы к API.

Ход работы:

Для начала я создал сервер на Json server + Json server auth. В ней создал три сущности : users , projects, profileData



```
1 {
2   "users": [
3     {
4       "email": "puroblast77@mail.ru",
5       "password": "$2a$10$FwEwIq9l869HtdPloqP99.1Bd0p3ANy4nFQeJcyvhrY9g1aRv0dVS",
6       "name": "Денис",
7       "lastName": "Унхос",
8       "id": 1
9     },
10    {
11      "email": "puroblast77@mail.ru",
12      "password": "$2a$10$FwEwIq9l869HtdPloqP99.1Bd0p3ANy4nFQeJcyvhrY9g1aRv0dVS",
13      "name": "Денис",
14      "lastName": "Унхос",
15      "id": 2
16    },
17    {
18      "email": "puroblast77@mail.ru",
19      "password": "$2a$10$FwEwIq9l869HtdPloqP99.1Bd0p3ANy4nFQeJcyvhrY9g1aRv0dVS",
20      "name": "Денис",
21      "lastName": "Унхос",
22      "id": 3
23    },
24    {
25      "email": "puroblast77@mail.ru",
26      "password": "$2a$10$FwEwIq9l869HtdPloqP99.1Bd0p3ANy4nFQeJcyvhrY9g1aRv0dVS",
27      "name": "Денис",
28      "lastName": "Унхос",
29      "id": 4
30    }
31  ]
32 }
33
34 "projects": [
35   {
36     "name": "Good Project",
37     "description": "my",
38     "link": "https://i.pinimg.com/imagery/reviews/83aizyUVApdyAIkuI4vRV-39_fit_scale_size_768x427_v1685559903.png",
39     "userId": 1,
40     "id": 1
41   },
42   {
43     "name": "Good Project",
44     "description": "my",
45     "image": "https://lh3.googleusercontent.com/nwM2Z0dV5yI0mcu-TZ_xJf3CvIAH6Agj18NQH0LOv8WTvUINBx2LJJnLhQ3Hm4SK_ctReybeqyWabaEsa8X0ms8xoDfrenPVPvui156hZCFvIw600-180-sg-rp",
46     "link": "github.com",
47     "userId": 1,
48     "id": 2
49   },
50   {
51     "name": "Good Project",
52     "description": "my",
53     "link": "github.com",
54     "image": "https://i.pinimg.com/imagery/reviews/83aizyUVApdyAIkuI4vRV-39_fit_scale_size_768x427_v1685559903.png",
55     "userId": 1,
56     "id": 3
57   },
58   {
59     "name": "Good Project",
60     "description": "my",
61     "id": 4
62   }
63 ]
64
65 "profileData": [
66   {
67     "name": "Денис",
68     "lastName": "Унхос",
69     "email": "puroblast77@mail.ru",
70     "about": "",
71     "image": "https://www.dictionary.com/s/wp-content/uploads/2018/06/pics-300x300.jpg",
72     "id": 1
73   },
74   {
75     "name": "Денис",
76     "lastName": "Унхос",
77     "email": "puroblast77@mail.ru",
78     "about": "",
79     "image": "https://www.dictionary.com/s/wp-content/uploads/2018/06/pics-300x300.jpg",
80     "id": 2
81   },
82   {
83     "name": "Денис",
84     "lastName": "Унхос",
85     "email": "puroblast77@mail.ru",
86     "about": "",
87     "image": "https://www.dictionary.com/s/wp-content/uploads/2018/06/pics-300x300.jpg",
88     "id": 3
89   }
90 ]
```

[1] GET /projects?name=Good20Project 200 24.123 ms - -

Для отправки запросов я решил поэкспериментировать и использовать 2 подхода : Для авторизации `createAsyncThunk` вместе с `axios` и `slice`. Для всего остального уже использовал RTK QUERY. Я заметил что у первого варианта больше возможностей кастомизации обработки респонсов, но также много бойлерплейт кода , который автоматически создается во втором варианте(например `slice`)

```
4  const authSlice = createSlice({
5    name: 'auth',
6    initialState,
7    reducers: {},
8    extraReducers : (builder) => {
9      builder.addCase(authUser.pending, (state) => {
10        state.status = 'loading'
11        state.error = null
12        state.token = null
13      })
14
15      builder.addCase(authUser.rejected, (state, action) => {
16        state.status = 'error'
17        state.error = action.payload
18      })
19      builder.addCase(authUser.fulfilled, (state, action) => {
20        state.status = "connect"
21        state.token = action.payload.accessToken
22        state.user = action.payload.user
23      })
24    }
25  })
```

```
5+ usages  Purobiast+1
export const authUser = createAsyncThunk(
  'post/authUser',
  async (payload, {rejectWithValue}) => {
    try {
      const {user , params} = payload

      const res = await axios.post(`http://localhost:8080/${params}`, user)

      if (res.status !== 201 && params === "register") {
        throw new Error('Ошибка при регистрации')
      }

      if (res.status !== 200 && params === "login") {
        throw new Error('Ошибка при входе')
      }

      return res.data
    } catch (e) {
      return rejectWithValue(e.message)
    }
  }
)
```

```

const rememberedKeys = ['authSlice'];

const reducers = combineReducers( {
  authSlice,
  [projectsApi.reducerPath] : projectsApi.reducer
})

const store = configureStore({
  reducer: rememberReducer(reducers),
  enhancers: [rememberEnhancer(
    window.localStorage,
    rememberedKeys,
    {persistWholeStore: true}
  )],
  middleware: (getDefaultMiddleware) => getDefaultMiddleware().concat(projectsApi.middleware)
})

```

5+ usages Denis Shliakhov

export default store

```

export const projectsApi = createApi({
  reducerPath: 'projectsApi',
  tagTypes: ['Project', 'Profile'],
  baseQuery: fetchBaseQuery({baseUrl: "http://localhost:8080/"}),
  endpoints: build => ({
    getProjects: build.query({
      query: (userId) => ({
        url: `projects?userId=${userId}`,
        method: "GET"
      }),
      providesTags: ["Project"]
    }),
    addProject: build.mutation({
      query: (project) => ({
        url: 'projects',
        method: 'POST',
        body: project,
      }),
      invalidatesTags: ["Project"]
    }),
    getOneProject: build.query({
      query: (projectName) => ({
        url: `projects?name=${projectName}`,
        method: "GET"
      })
    })
  }),
});

```

```
getUsers: build.query({
  query: () => ({
    url: 'users',
    method: "GET"
  })
}),
editProject: build.mutation({
  query: (project) => ({
    url: `projects/${project[1]}`,
    method: "PUT",
    body: project[0]
  }),
  invalidatesTags: ["Project"]
}),
deleteProject: build.mutation({
  query: (projectId) => ({
    url: `projects/${projectId}`,
    method: "DELETE"
  }),
  invalidatesTags: ["Project"]
}),
editProfileData: build.mutation({
  query: (user) => ({
    url: `profileData/${user[1]}`,
    method: "PUT",
    body: user[0]
  }),

```

```
    getProfileData: build.query({
      query: (profileId) => ({
        url: `profileData/${profileId}`,
        method: "GET"
      }),
      providesTags: ["Profile"]
    })
  })
})
```

4 usages  Puroblast

```
export const {
  useGetProjectsQuery,
  useAddProjectMutation,
  useGetOneProjectQuery,
  useGetUsersQuery,
  useEditProjectMutation,
  useDeleteProjectMutation,
  useEditProfileDataMutation,
  useAddProfileDataMutation,
  useGetProfileDataQuery
} = projectsApi
```


Пример использования хуков и запроса авторизации:

```
2 usage  Puroblast
const requestAuthUser = async (data) => {
  await dispatch(authUser({user: data, params: status})).then(value => {
    if (!value.error) {
      if (status === "register") {
        const profileData = {
          name: data.name,
          lastName: data.lastName,
          email: data.email,
          about: "",
          image: ""
        }
        addProfileData(profileData)
      }
      reset()
      setHide(true)
      navigate("/")
    } else {
      setHide(false)
    }
  })
}
```

```
const [editProfileData] = useEditProfileDataMutation()
const {data, isLoading} = useGetProfileDataQuery(userId)

const {
  register,
  handleSubmit
} = useForm({
  mode: "onBlur", values: {
    name: data ? data.name : "",
    lastName: data ? data.lastName : "",
    email: data ? data.email : "",
    about: data ? data.about : "",
    image: data ? data.image : "",
  }
})

1 usage  Puroblast
const handleEditProfileData = async (data) => {
  await editProfileData([data, userId])
  setDisabled(true)
}

if (isLoading) {
  return <Container>
    <Row>
      <Col>
        <h1>Загрузка...</h1>
      </Col>
    </Row>
  </Container>
}
```

Итог:

Все данные связаны с бд, кладутся в `redux` через запросы и обновляются при изменении в базе данных с помощью `provideTags`. Также данные авторизации хранятся в `localStorage` и подгружаются в редакс сразу при инициализации, из-за чего после рефреша или перезахода не надо заного авторизоваться.