

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №3

Выполнила:

Стукалова Александра

Группа К3343

Проверил:

Добряков Д. И.

Санкт-Петербург

2026 г

Введение

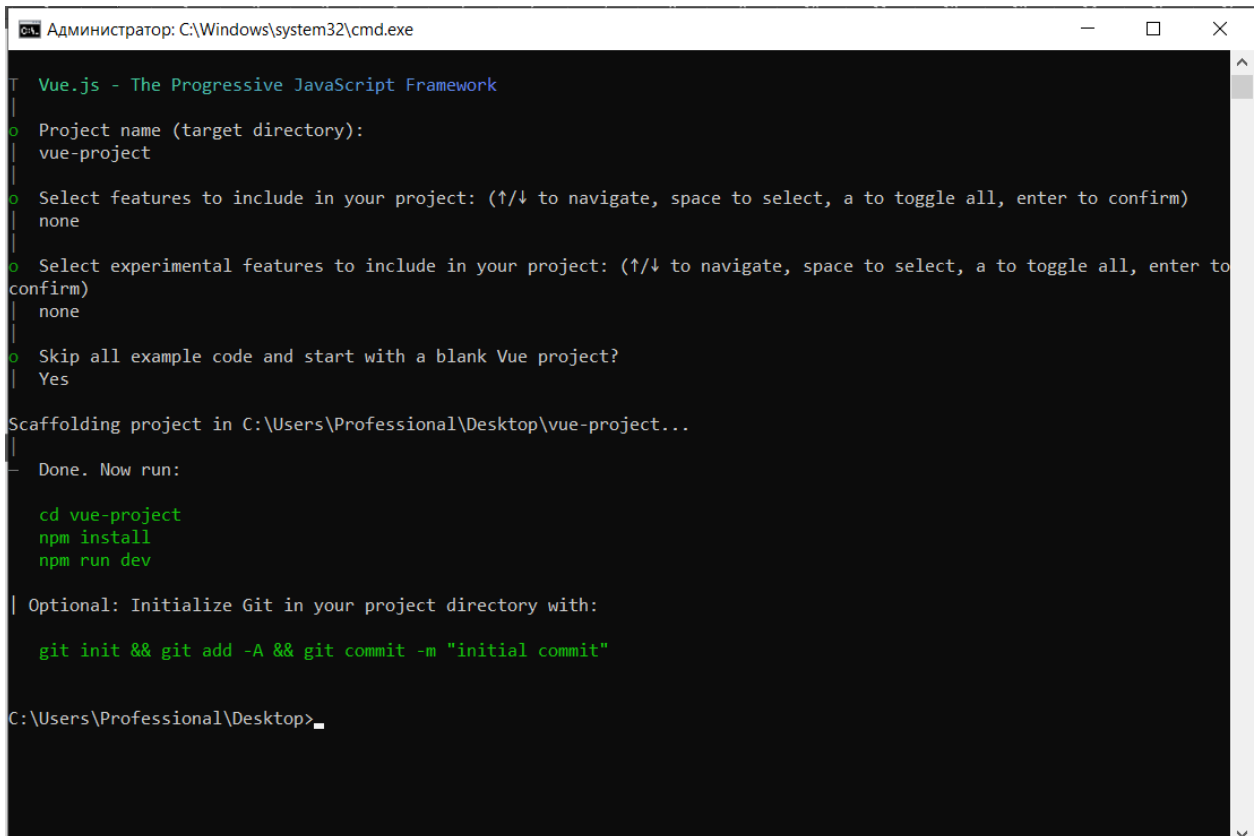
В данной лабораторной работе нужно перенести сайт из первой и второй лабораторных на фреймворк Vue.js и сделать из него одностраничное приложение (SPA).

Что для этого нужно сделать:

1. Подключить роутер, чтобы страницы переключались без перезагрузки
2. Настроить работу с API через axios (вместо fetch)
3. Разбить сайт на компоненты (шапка, подвал, карточки)
4. Вынести повторяющийся код в отдельные файлы composable

В качестве основы я взяла код из второй лабораторной работы, где уже была настроена работа с JSON-server и авторизация.

Для выполнения третьей лабораторной работы я создала новый проект на Vue.js с использованием сборщика Vite. В терминале я выполнила команду `npm create vue@latest`. После завершения создания проекта терминал вывел инструкцию с командами для перехода в папку проекта, установки зависимостей и запуска сервера разработки:

A screenshot of a Windows command prompt window titled "Администратор: C:\Windows\system32\cmd.exe". The window has a dark background with light green text. It shows the output of the Vue CLI command. The text includes: "Vue.js - The Progressive JavaScript Framework", "Project name (target directory): vue-project", "Select features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm) none", "Select experimental features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm) none", "Skip all example code and start with a blank Vue project? Yes", "Scaffolding project in C:\Users\Professional\Desktop\vue-project...", "Done. Now run:", "cd vue-project", "npm install", "npm run dev", "Optional: Initialize Git in your project directory with:", "git init && git add -A && git commit -m 'initial commit'", and the current directory "C:\Users\Professional\Desktop>".

```
Администратор: C:\Windows\system32\cmd.exe

Vue.js - The Progressive JavaScript Framework

Project name (target directory):
vue-project

Select features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm)
none

Select experimental features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm)
none

Skip all example code and start with a blank Vue project?
Yes

Scaffolding project in C:\Users\Professional\Desktop\vue-project...

Done. Now run:

cd vue-project
npm install
npm run dev

Optional: Initialize Git in your project directory with:

git init && git add -A && git commit -m "initial commit"

C:\Users\Professional\Desktop>
```

Далее я создала в папке `src/components` файл `Navbar.vue` (навигационную папку, которая будет отображаться на всех страницах сайта)

В этом компоненте я перенесла код навигации из HTML-файлов предыдущих лабораторных работ, но заменила ссылки а `href="#"` на компоненты `Vue - router-link`. Это для того, чтобы переходы между страницами работали без перезагрузки.

Компонент содержит логотип и ссылки на все основные разделы: главную, вход, регистрацию, личный кабинет, тренировки и блог. Стили для него уже есть в подключенном файле `style.css`, поэтому внутри компонента я оставила пустой блок стилей:

```
JS main.js  ▾ Navbar.vue ×
components ▾ ▾ Navbar.vue
1  <template>
2    <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
3      <div class="container">
4        <router-link class="navbar-brand" to="/">FitnessPlatform</router-link>
5        <div class="navbar-nav ms-auto">
6          <router-link class="nav-link" to="/">Главная</router-link>
7          <router-link class="nav-link" to="/login">Вход</router-link>
8          <router-link class="nav-link" to="/register">Регистрация</router-link>
9          <router-link class="nav-link" to="/profile">Личный кабинет</router-link>
10         <router-link class="nav-link" to="/workouts">Тренировки</router-link>
11         <router-link class="nav-link" to="/blog">Блог</router-link>
12       </div>
13     </div>
14   </nav>
15 </template>
16
17 <script setup>
18   // пока пусто
19 </script>
20
21 <style scoped>
22 </style>
```

В файле `src/router/index.js` я описала все маршруты, которые будут в приложении. Каждому пути соответствует своя страница, которая будет загружаться при переходе по этому адресу:

```
JS main.js | App.vue | Navbar.vue | JS index.js X
router > JS index.js > routes
1 import { createRouter, createWebHistory } from 'vue-router'
2
3 // Здесь будет импорт страниц
4 import HomeView from '../views/HomeView.vue'
5
6 const router = createRouter({
7   history: createWebHistory(import.meta.env.BASE_URL),
8   routes: [
9     {
10      path: '/',
11      name: 'home',
12      component: HomeView
13    },
14    {
15      path: '/login',
16      name: 'login',
17      // компонент создадим позже
18      component: () => import('../views/LoginView.vue')
19    },
20    {
21      path: '/register',
22      name: 'register',
23      component: () => import('../views/RegisterView.vue')
24    },
25    {
26      path: '/profile',
27      name: 'profile',
28      component: () => import('../views/ProfileView.vue')
29    },
30    {
31      path: '/workouts',
32      name: 'workouts',
33      component: () => import('../views/WorkoutsView.vue')
34    }
35  ]
36 })
```

После описания маршрутов я подключила роутер в main.js с помощью `app.use(router)`, чтобы он стал доступен во всем приложении:

```
JS main.js X | Footer.vue components | App.vue | Footer.vue \ | HomeView.vue | LoginView.vue • | RegisterView.vue |
JS main.js > ...
1 import { createApp } from 'vue'
2 import App from './App.vue'
3 import router from './router'
4 import './assets/style.css'
5
6 const app = createApp(App)
7 app.use(router)
8 app.mount('#app')
```

В файле `App.vue` я добавила `router-view`. Это место, куда роутер будет подставлять нужную страницу в зависимости от текущего адреса:

```
JS main.js  App.vue  NavBar.vue  JS index.js  ...
App.vue
1 <template>
2   <div id="app">
3     <Navbar />
4     <router-view />
5     <Footer />
6   </div>
7 </template>
8
9 <script setup>
10 import Navbar from './components/Navbar.vue'
11 import Footer from './components/Footer.vue'
12 </script>
13
14 <style>
15
16 </style>
```

Я запустила проект командой `npm run dev` в терминале, после чего становится доступен по адресу <http://localhost:5173/>:

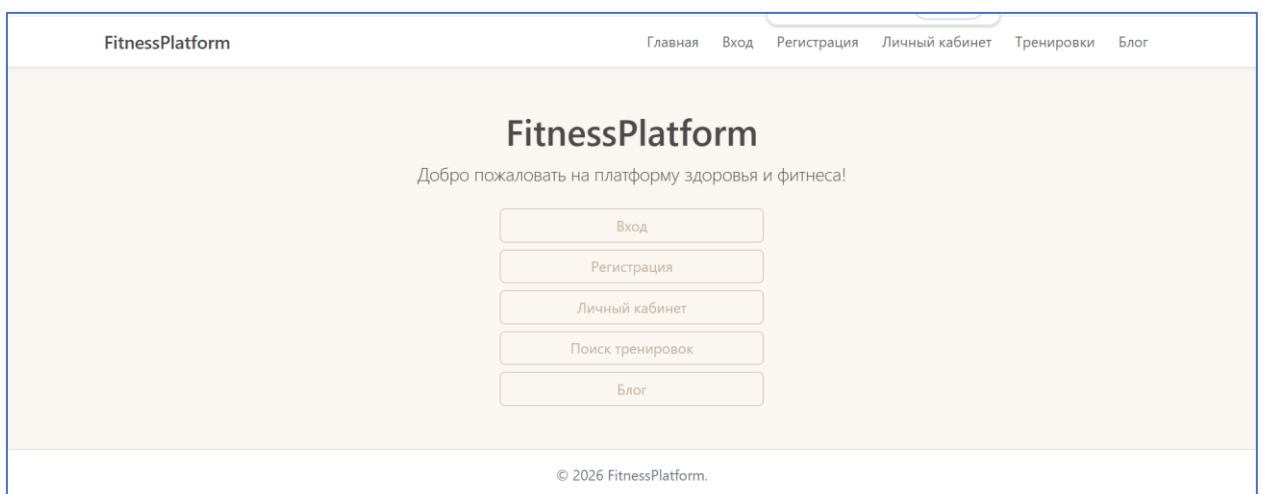
```
C:\Users\Professional\Desktop\vue-project>npm run dev

> vue-project@0.0.0 dev
> vite

VITE v7.3.1 ready in 748 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  Vue DevTools: Open http://localhost:5173/__devtools__/ as a separate window
  Vue DevTools: Press Alt(⌘)+Shift(⇧)+D in App to toggle the Vue DevTools
  press h + enter to show help
```

Сайт пока что выглядит так:



Для работы с API я установила библиотеку `axios`:

```
C:\Users\Professional\Desktop\vue-project>npm install axios

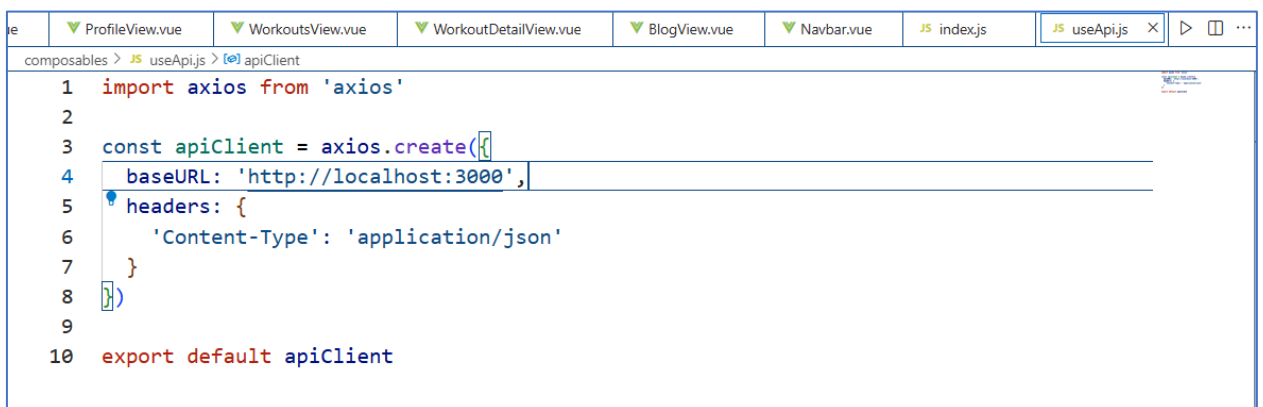
added 23 packages, and audited 149 packages in 4s

37 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Professional\Desktop\vue-project>
```

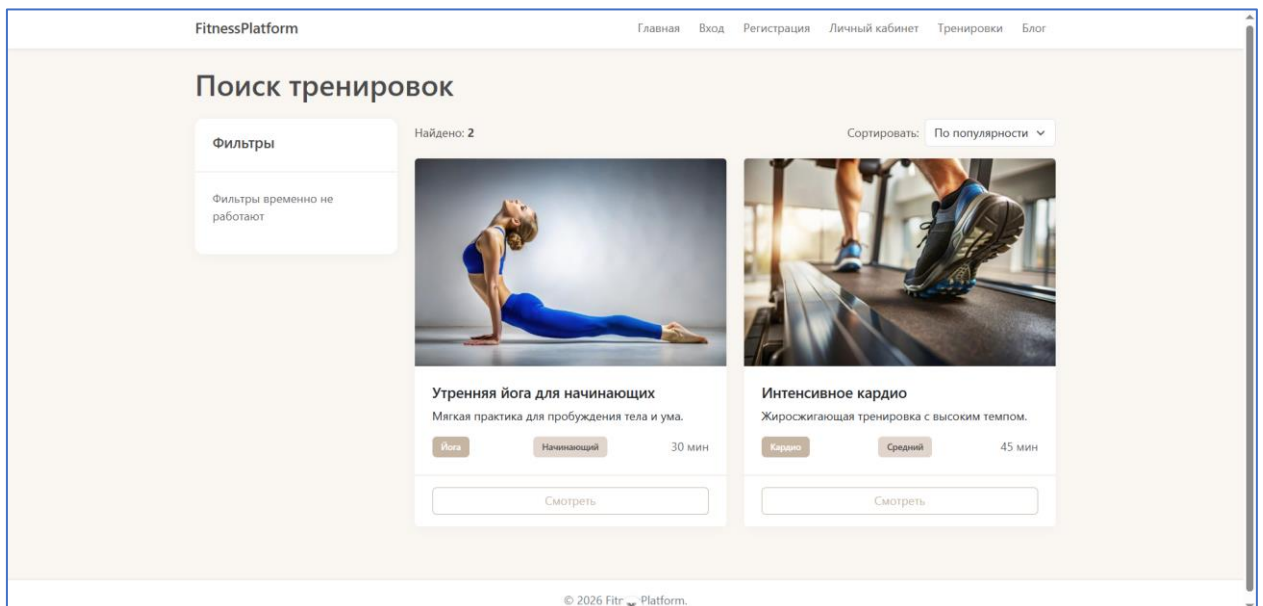
Для работы с сервером я создала отдельный файл useApi.js в папке composables. В нем настраивается axios: указывается наш URL (<http://localhost:3000>) и заголовки запросов. Все компоненты могут импортировать этот файл и делать запросы к API, не дублируя настройки:



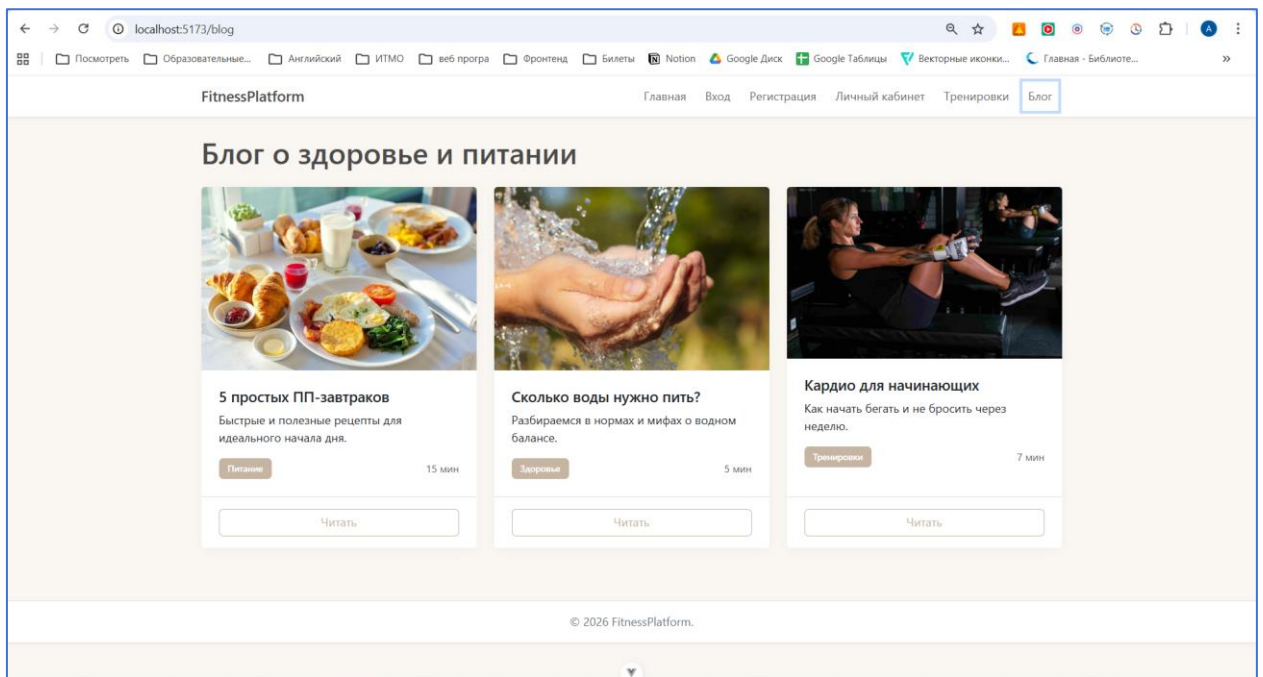
```
composables > JS useApi.js > apiClient
1 import axios from 'axios'
2
3 const apiClient = axios.create({
4   baseURL: 'http://localhost:3000',
5   headers: {
6     'Content-Type': 'application/json'
7   }
8 })
9
10 export default apiClient
```

Для загрузки тренировок я создала composable-функцию useWorkouts. В ней используются переменные workouts (список тренировок), loading (состояние загрузки) и error (ошибка):

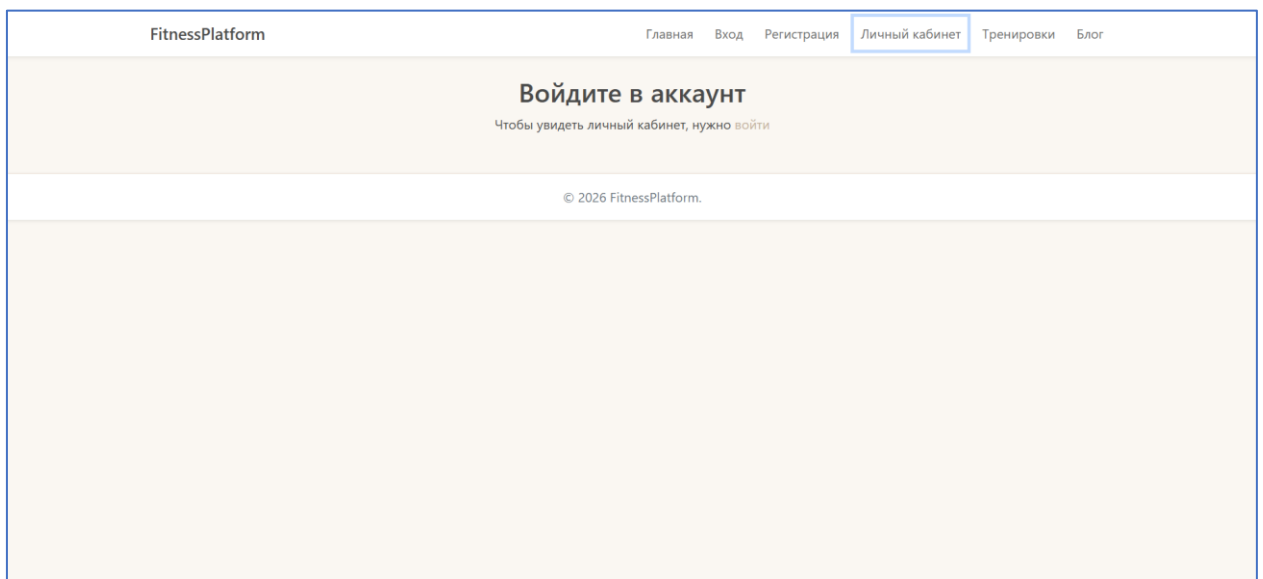
```
WorkoutsView.vue | WorkoutDetailView.vue | BlogView.vue | NavBar.vue | JS index.js | JS useApi.js | JS useWorkouts.js X
composables > JS useWorkouts.js > useWorkouts
1 import { ref } from 'vue'
2 import apiClient from './useApi'
3
4 export function useWorkouts() {
5   const workouts = ref([])
6   const loading = ref(false)
7   const error = ref(null)
8
9   const fetchWorkouts = async () => {
10     loading.value = true
11     error.value = null
12     try {
13       const response = await apiClient.get('/workouts')
14       workouts.value = response.data
15     } catch (err) {
16       error.value = 'Ошибка загрузки тренировок'
17       console.error(err)
18     } finally {
19       loading.value = false
20     }
21   }
22
23   return {
24     workouts,
25     loading,
26     error,
27     fetchWorkouts
28   }
29 }
```

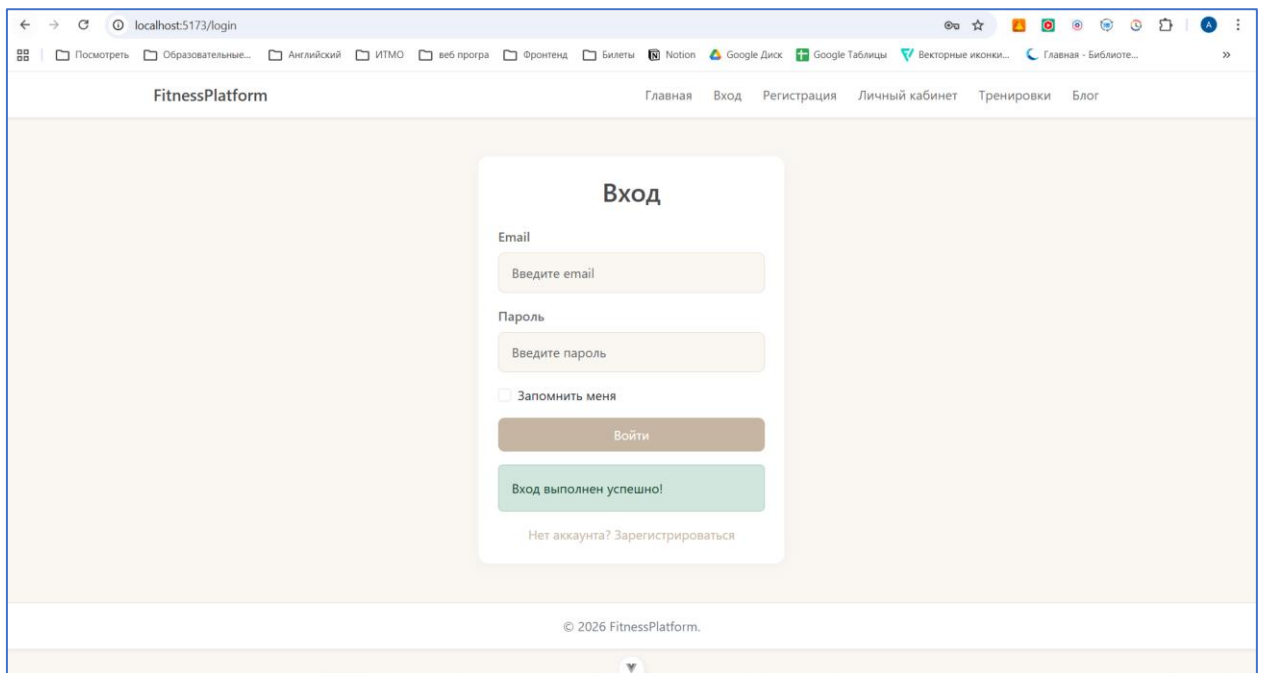


На странице блога я сделала то же самое, что и с тренировками. Я создала файл useBlog.js, который загружает статьи с сервера. Когда открываешь страницу, появляются карточки с картинками, заголовками и временем чтения. Данные берутся из того же db.json, что и во второй лабораторной:

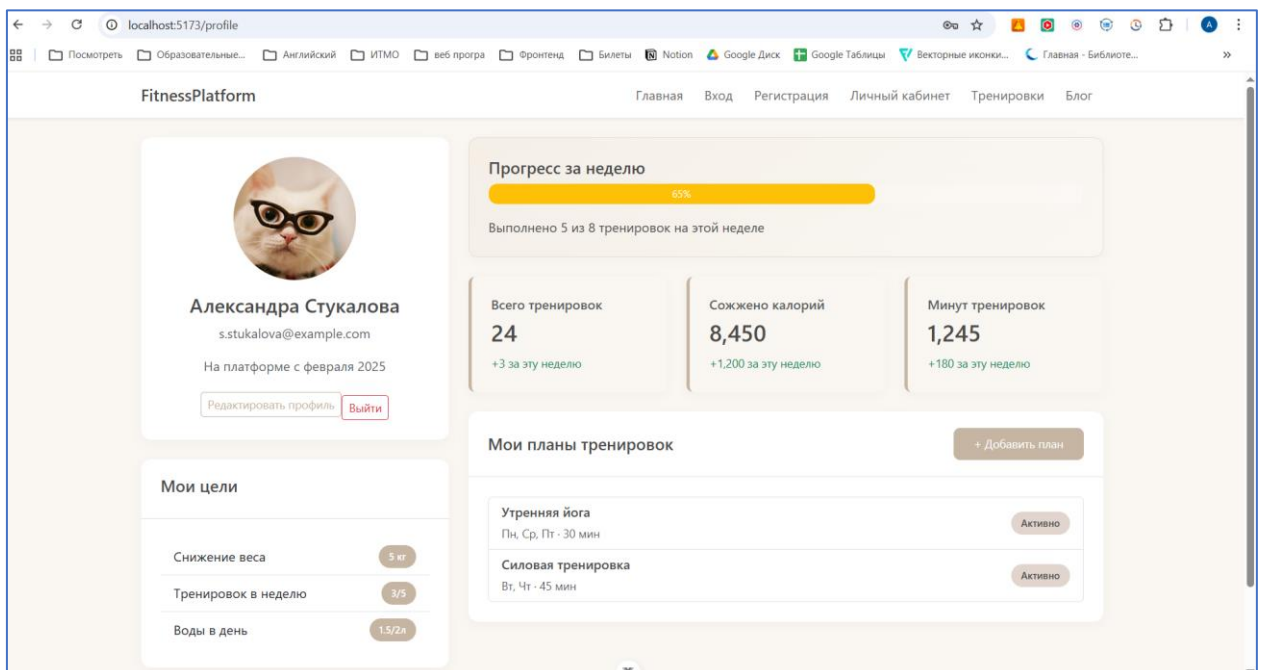


Для авторизации я написала файл `useAuth.js`. Там есть две основные функции: `login` и `register`. Когда пользователь вводит email и пароль на странице входа, отправляется запрос на сервер. Если пользователь найден и пароль подходит, то вход выполняется. При регистрации проверяется, нет ли уже такого email, и если все хорошо, то новый пользователь добавляется в базу. Чтобы при перезагрузке страницы не приходилось заходить заново, я добавила сохранение в `localStorage`.



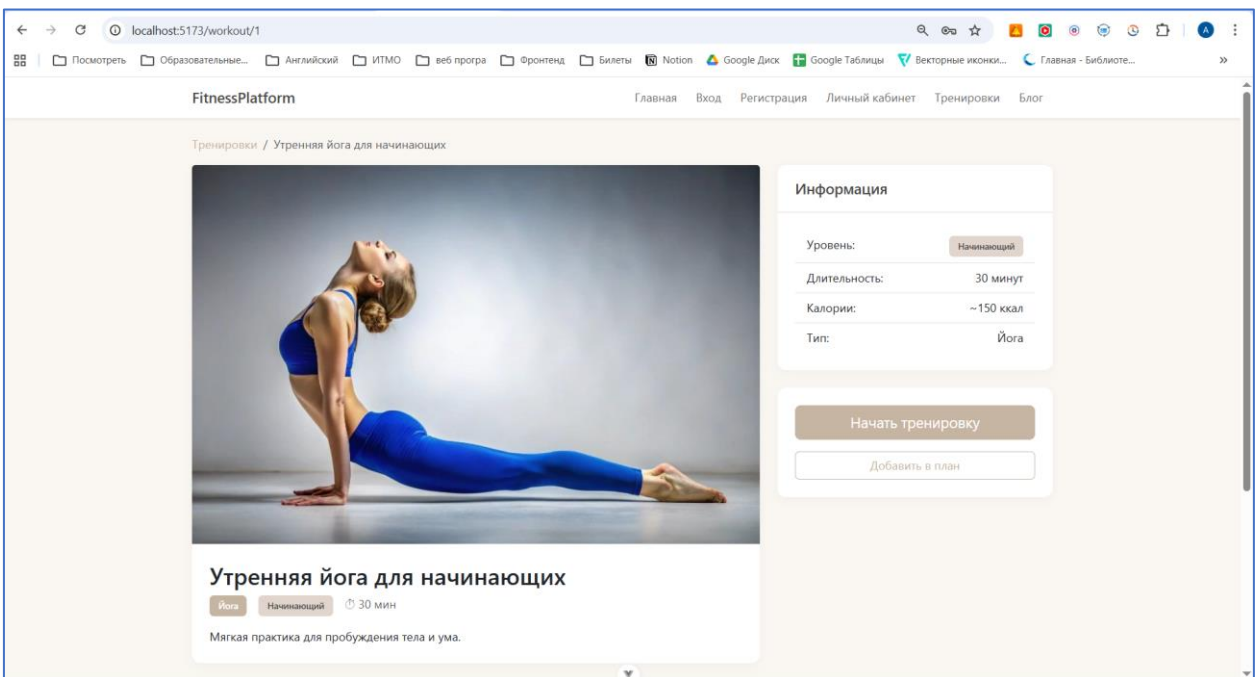
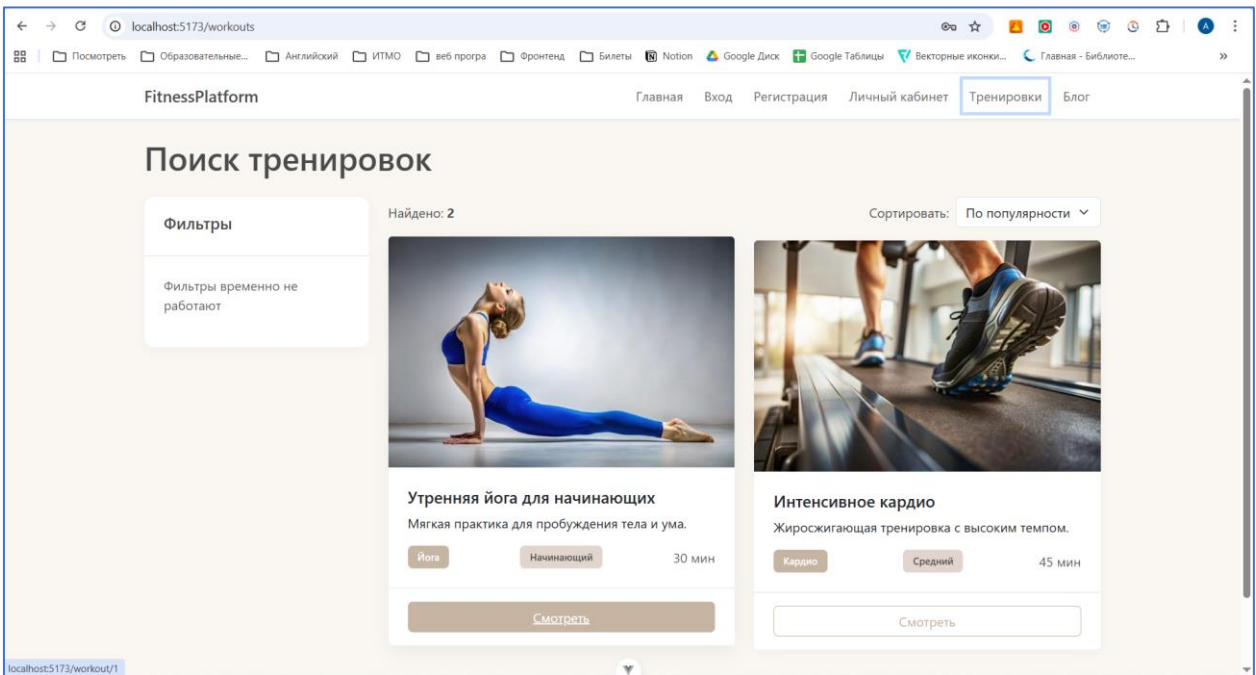


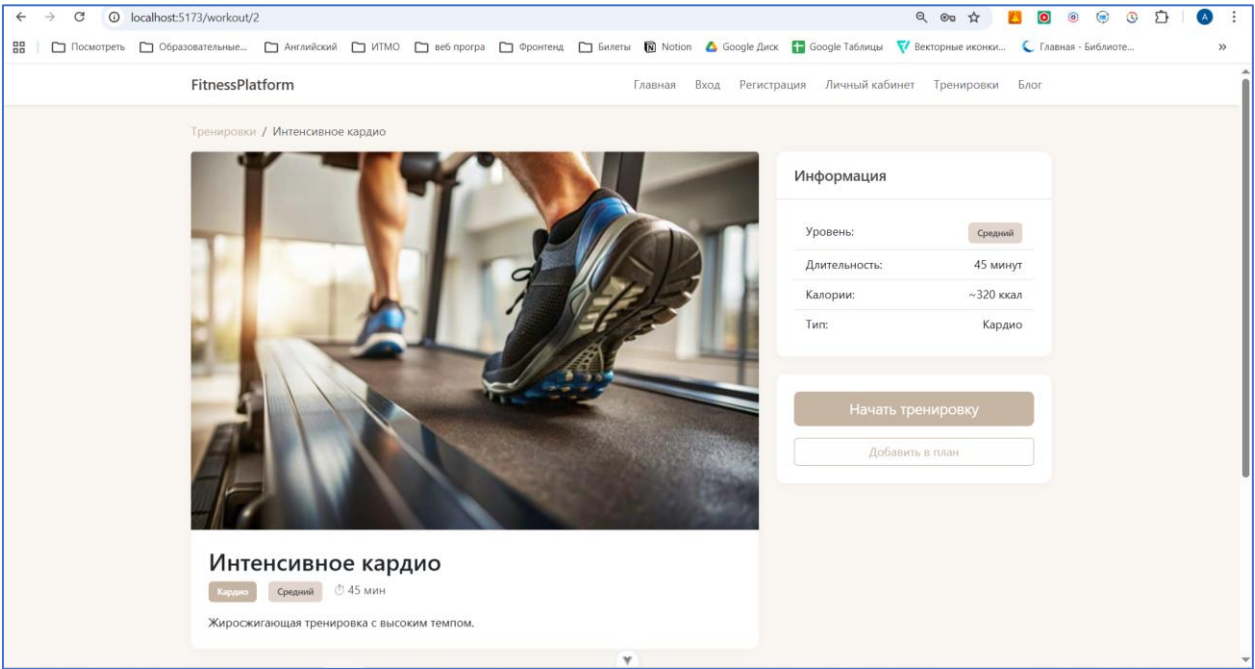
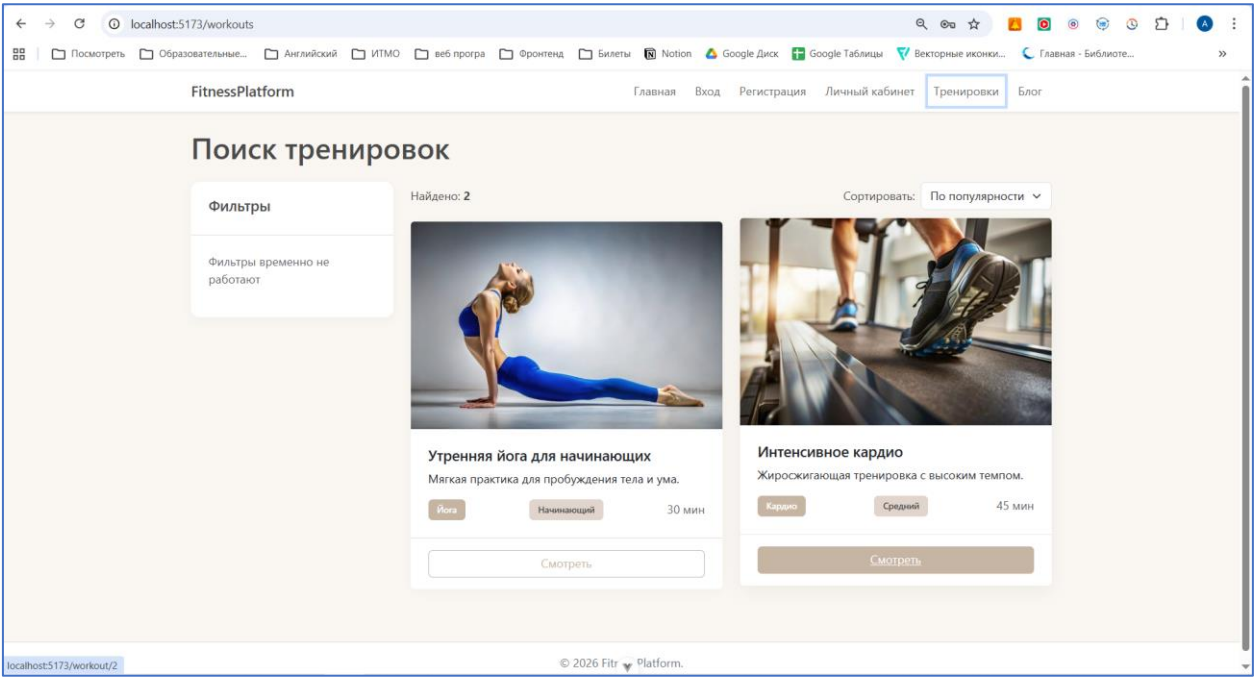
На странице профиля если пользователь зарегистрирован, то показывается его имя, email и другая информация. Если нет, то появляется сообщение, что нужно войти. Данные для профиля пока статичные, как в первой лабораторной.

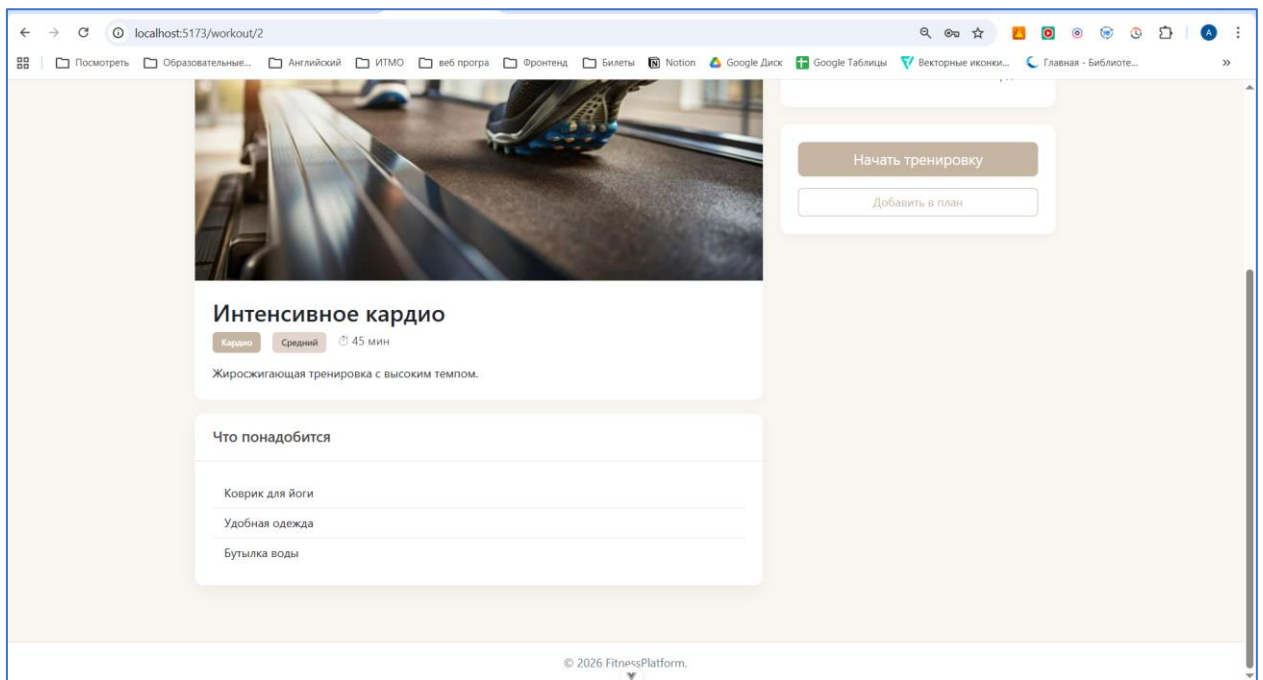


Когда на странице тренировок нажимаешь «Смотреть», открывается новая страница с подробной информацией о тренировке. Я сделала отдельный composable useWorkoutDetail, который загружает с сервера только одну

тренировку по ее ID. На странице отображается картинка, описание, уровень сложности, длительность и другая информация.







Проект загружен на GitHub и доступен для просмотра в репозитории.

Для запуска проекта локально нужно:

1. Скачать папку lab_3
2. Установить зависимости: `npm install`
3. Запустить JSON-server: `json-server --watch db.json --port 3000`
4. Запустить проект: `npm run dev`
5. Перейти по ссылке: <http://localhost:5173/>