САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа № 3
“Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.JS”

Выполнил:
Коротин А.М.

Группа:
K33392

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Мигрировать ранее написанный сайт на фреймворк Vue.JS.

Минимальные требования:

- Должен быть подключён роутер
- Должна быть реализована работа с внешним API
- Разумное деление на компоненты

Ход работы

Для реализации ЛР был выбран frontend-framework Quasar. В приложении были выделены следующие роуты (рисунок 1).

```
const routes = [
  {
    path: '/auth',
    component: () => import('@/layouts/AuthLayout.vue'),
    children: [
      { path: 'login', name: 'Login', component: () => import('@/pages/LoginPage.vue'), meta: { requiresAuth: false } },
      { path: 'register', name: 'Register', component: () => import('@/pages/SignUpPage.vue'), meta: { requiresAuth: false } }
    ]
  },
  {
    path: '/',
    component: () => import("@/layouts/MainLayout.vue"),
    meta: {
      requiresAuth: true
    },
    children: [
      {
        path: 'collection',
        name: 'Collection',
        component: () => import('@/pages/CollectionPage.vue'),
        meta: { requiresAuth: true }
      },
      {
        path: 'search',
        name: 'Search',
        component: () => import('@/pages/SearchPage.vue'),
        meta: { requiresAuth: true },
        props: (route) => ({ query: route.query.q })
      }
    ]
  },
```

Рисунок 1 - Роуты основные роуты приложения

Также, в соответствие с роутами, были выделены 2 layout’а:

- AuthLayout, рисунок 2;
- MainLayout, рисунок 3.

```
<template>
    <q-layout view="hHh lpR fFf">
        <q-page-container>
            <router-view />
        </q-page-container>
    </q-layout>
</template>
```

Рисунок 2 - AuthLayout

Layout авторизации не содержит в себе ничего, кроме основной страницы (router-view). MainLayout же содержит в себе хедер с навигацией, футер с проигрыванием композиций и правый drawer. Данный layout может быть продемонстрирован в среде Quasar Layout Builder:

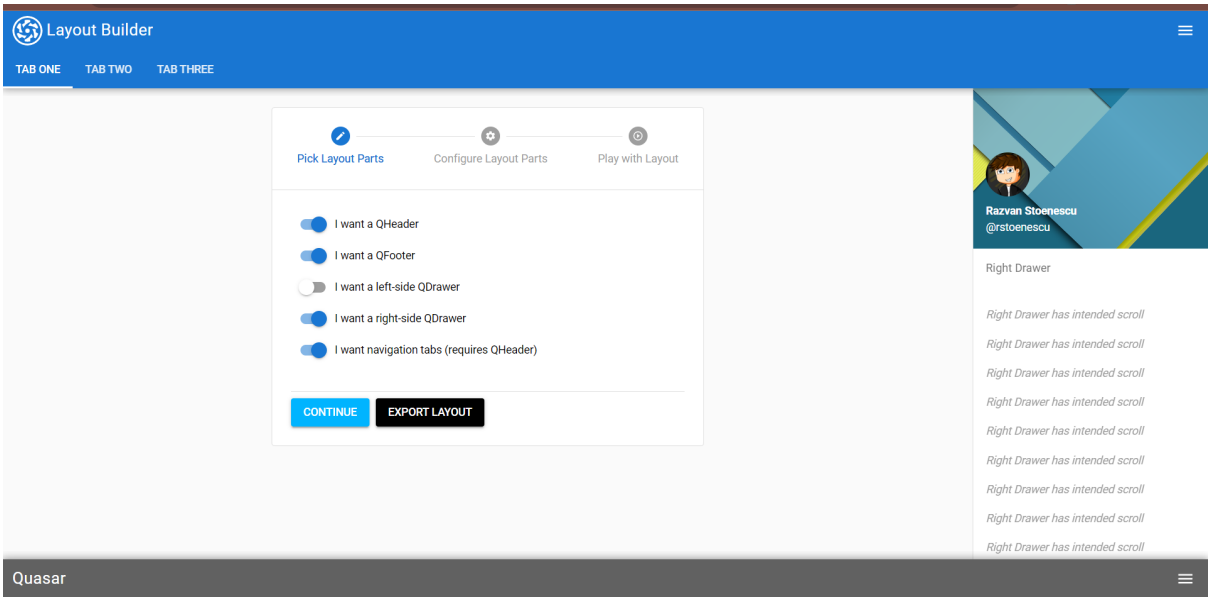

Рисунок 3 - MainLayout

Реальная страница приложения, использующая MainLayout (рисунок 4):

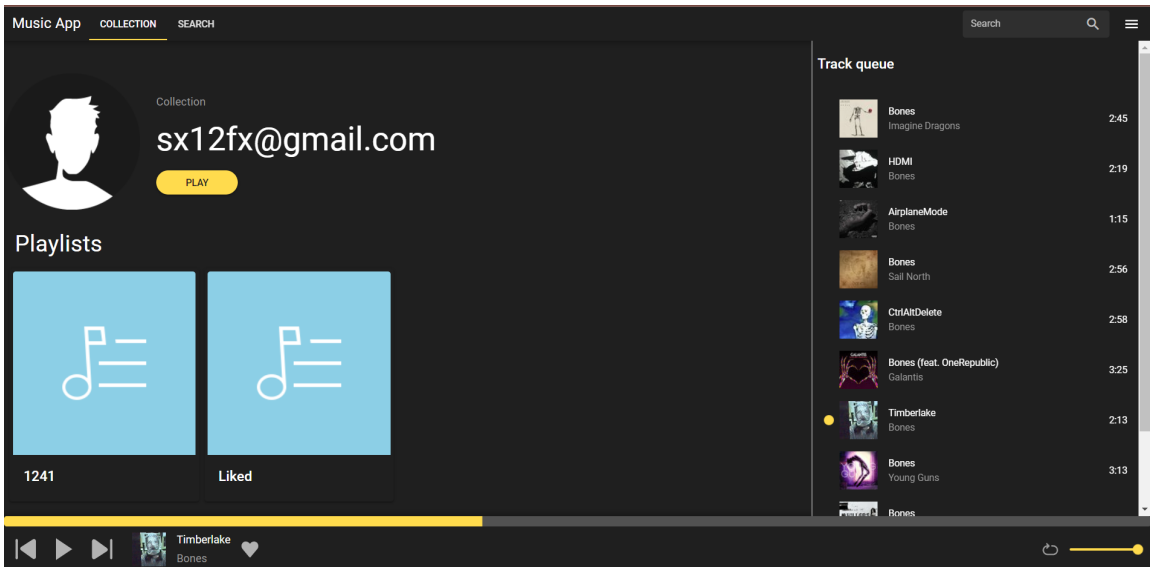

Рисунок 4 - Страница коллекции

Как и в предыдущих работах, используется работа со внешним API Deezer.com и json-server для авторизации и хранения информации о пользователях. Информация о композициях, получаемая с сервиса Deezer (рисунок 5).

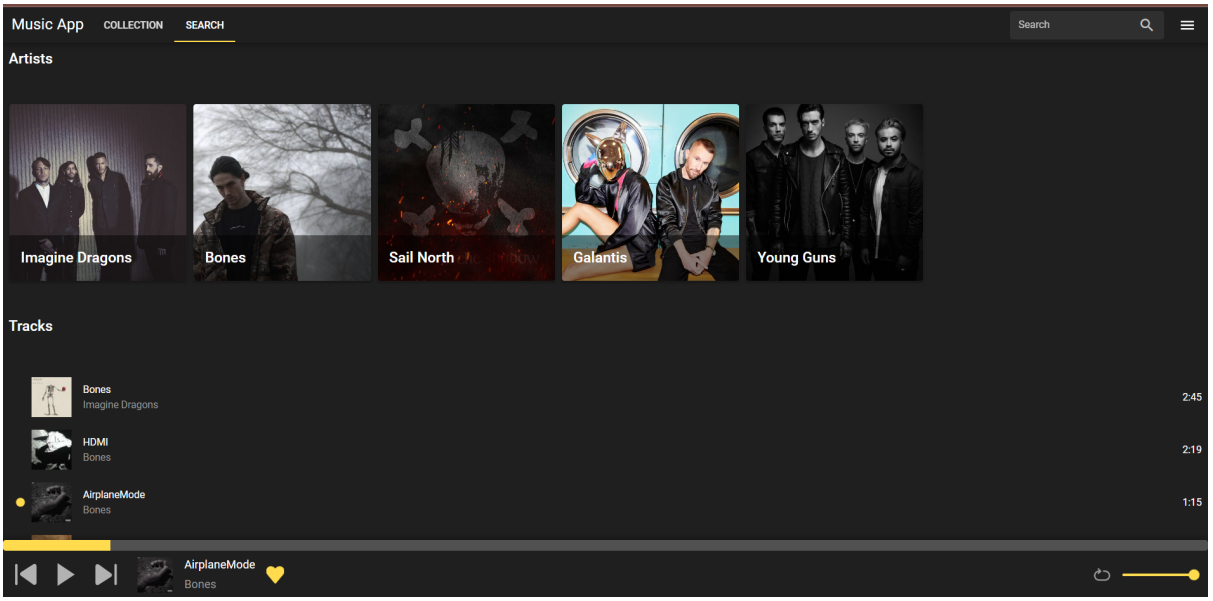

Рисунок 5 - Страница поиска композиций

Приведем некоторые компоненты, выделенные при разработке пользовательского интерфейса:

- PlaylistCard.vue, рисунок 6;
- ArtistCard.vue, рисунок 7;
- TrackCard.vue, рисунок 8.

```vue
<template>
    <q-card shrink class="playlist-card border-hover bg-primary col-xs-6 col-sm-3 col-md-2 col-lg-2 q-pa-xs cursor-pointer">
        <img src="@/assets/collection/playlist-cover_no_cover.png">
        <q-card-section>
            <div class="playlist-caption block text-h6 ellipsis overflow-hidden">{{ name }}</div>
        </q-card-section>
    </q-card>
</template>
<script>
export default {
    props: {
        name: {
            type: String,
            required: true
        }
    }
}
</script>
<style scoped lang="scss">
0 references
@import '@/css/quasar.variables.scss';

1 reference
.playlist-card {
    color: $text;
}
</style>
```

Рисунок 6 - PlaylistCard.vue

Рисунок 7 - ArtistCard.vue

Исходный код компоненты TrackCard.vue слишком объемный, его можно найти в Pull Request лабораторной работы.

Взаимодействие с внешними API с использованием axios производились следующим образом:

```
const apiUrl = 'http://localhost:3000';
export const api = axios.create({ baseURL: apiUrl });
```

Рисунок 8 - Создание axios instance

```
export class AuthApi {
    constructor(api) {
        this.api = api
    }

    async login(data) {
        return this.api({
            url: '/login',
            method: 'POST',
            data,
            headers: {
                'Content-Type': 'application/json'
            }
        })
    }

    async signUp(data) {
        return this.api({
            url: '/register',
            method: 'POST',
            data,
            headers: {
                'Content-Type': 'application/json'
            }
        })
    }

}
```

Рисунок 9 - Класс для работы с API на примере авторизации

Вывод:

В ходе выполнения лабораторной работы я познакомился с frontend фреймворками Vue.js и Quasar Framework. Полученные знания являются основополагающими при дальнейшем изучении frontend разработки.