

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 3

Выполнил: Зайцев Кирилл
Дмитриевич

Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

Задача 1

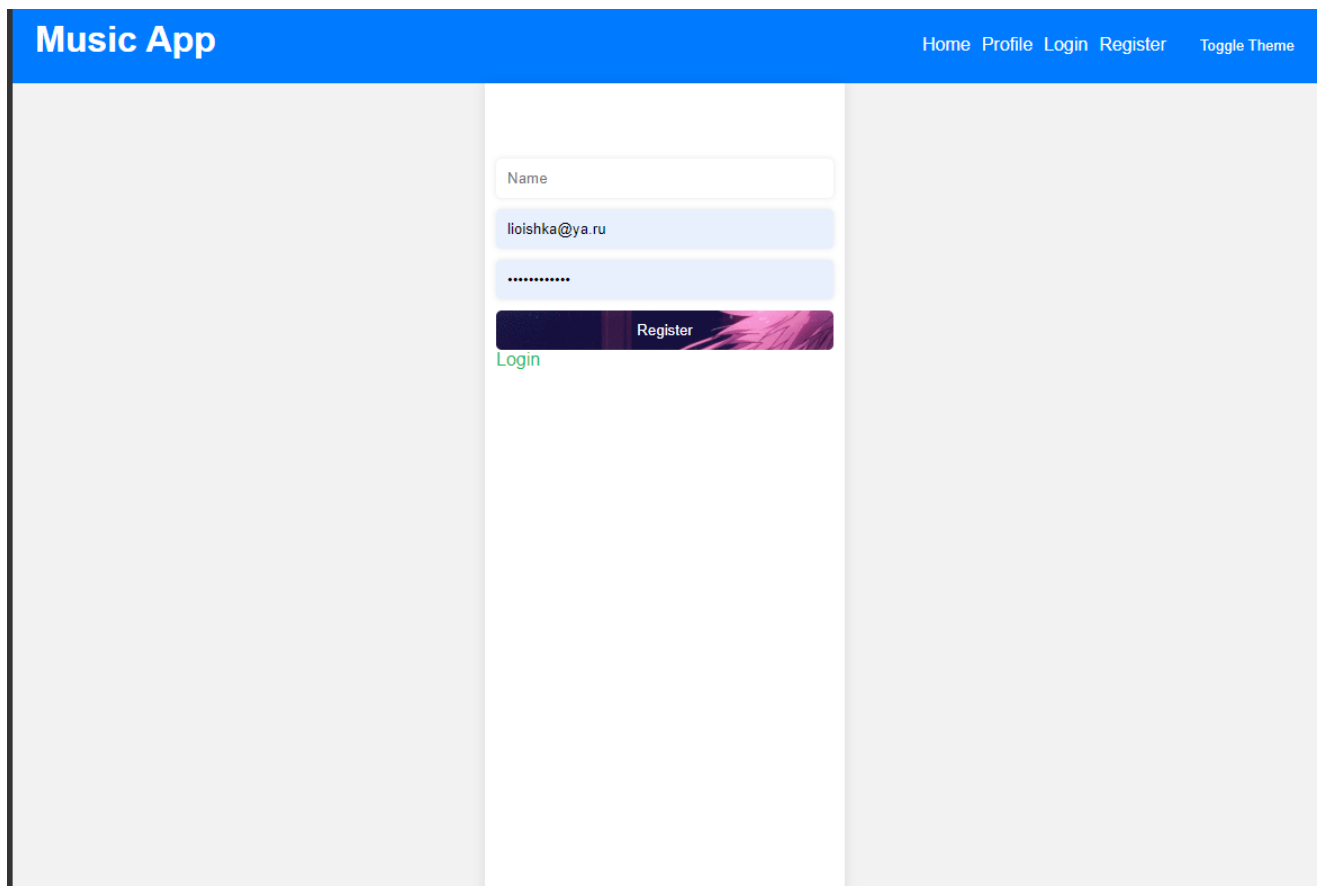
Мигрировать ранее написанный сайт на фреймворк Vue.JS.

Минимальные требования:

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

Ход работы:

Так как задачей было мигрировать сайт на Vue, основные страницы, такие как вход, регистрация, страница профиля и главная страница были переписаны для нового стека(рисунки1-4).



Music App

Home Profile Login Register Toggle Theme

Name

lioishka@ya.ru

Register

Login

Рисунок 1 – Регистрация

[Register](#)

Рисунок 2 – вход

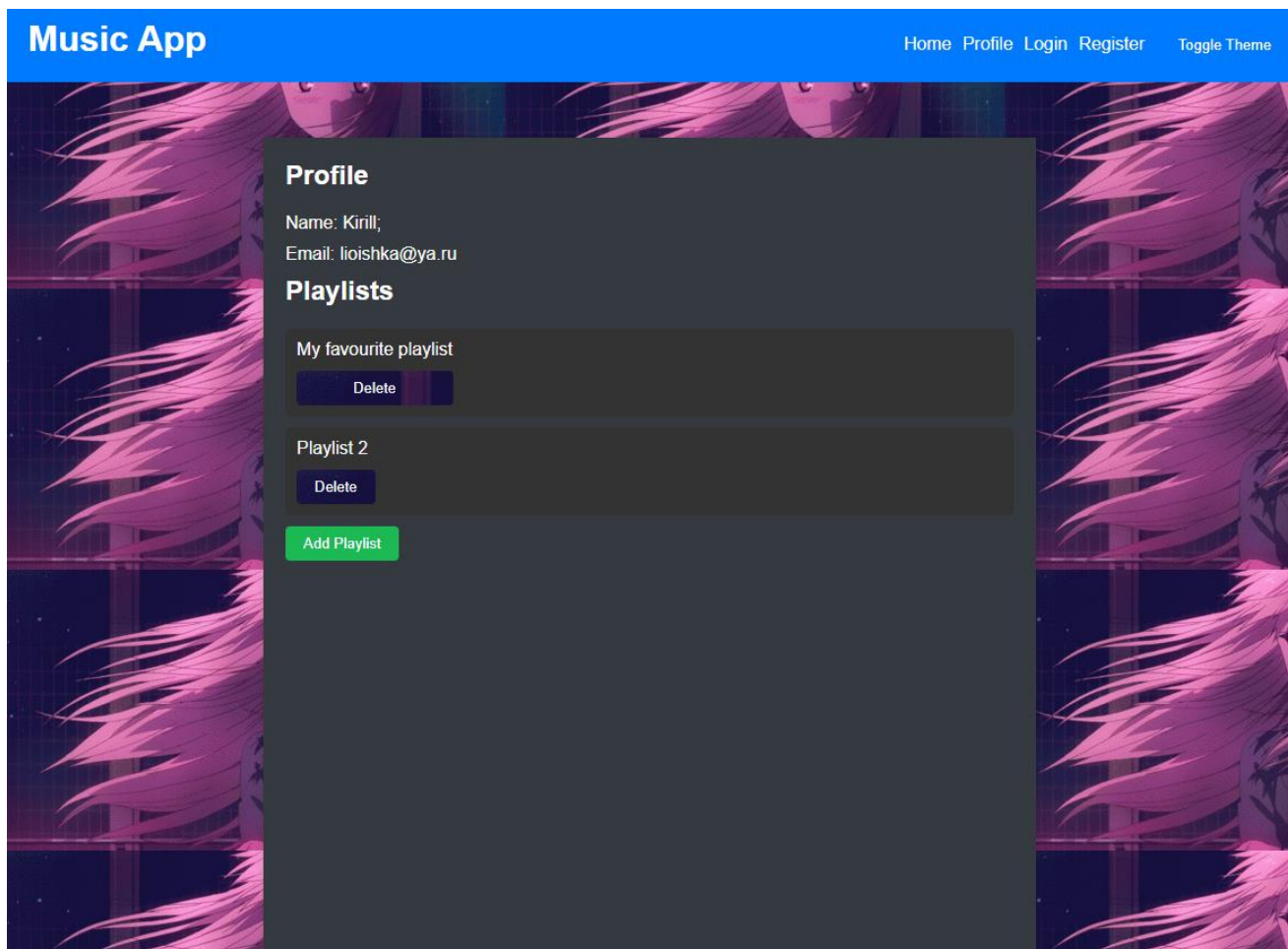


Рисунок 3 – профиль

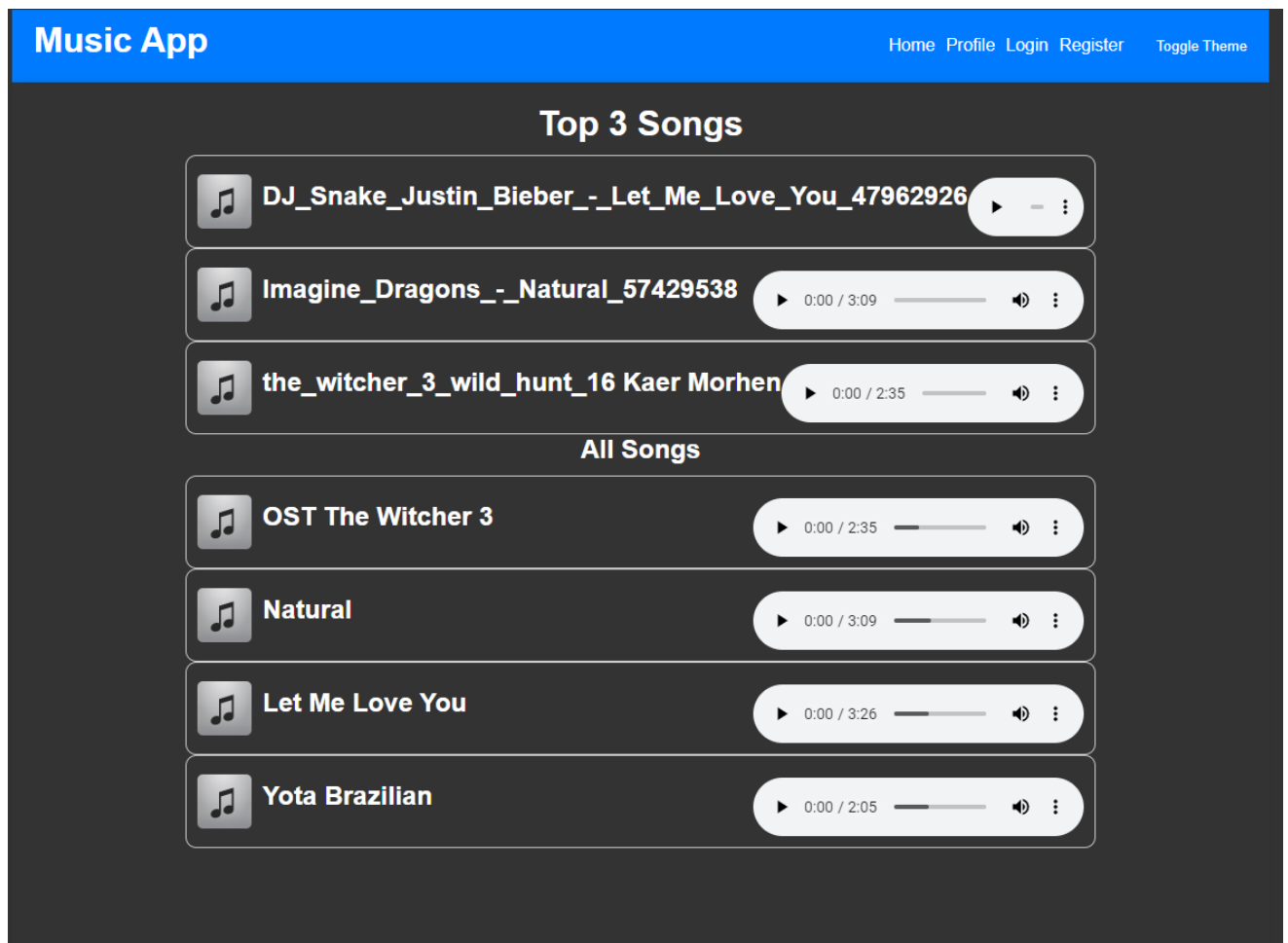


Рисунок 4 – главная страница

Поскольку код всех файлов получился довольно объёмным, то пробежимся лишь по самым примечательным, на мой взгляд:

-store – отвечает за сохранение состояний сайта, например, таких как тема, и за локальный обмен данными.

```
import { createStore } from 'vuex';

export const store = createStore({
  state() {
    return {
      user: null,
      isDarkTheme: localStorage.getItem('isDarkTheme') === 'true'
    };
  },
  mutations: {
    setUser(state, user) {
      state.user = user;
    },
    toggleTheme(state) {
      state.isDarkTheme = !state.isDarkTheme;
      localStorage.setItem('isDarkTheme', state.isDarkTheme.toString());
    }
  }
});
```

```

    },
    actions: {
      // Добавляем действие для загрузки данных пользователя
      async fetchUser({ commit }) {
        try {
          const response = await fetch('/api/user'); // Предполагается, что
          // сервер предоставляет данные о пользователе по этому пути
          const data = await response.json();
          commit('setUser', data);
        } catch (error) {
          console.error('Error fetching user data:', error);
        }
      },
      // Добавляем действие для обновления плейлистов пользователя
      async updatePlaylists({ commit }) {
        try {
          const response = await fetch('/api/playlists'); //
          // Предполагается, что сервер предоставляет данные о плейлистах пользователя по
          // этому пути
          const data = await response.json();
          commit('updateUserPlaylists', data);
        } catch (error) {
          console.error('Error updating user playlists:', error);
        }
      },
      toggleTheme({ commit }) {
        commit('toggleTheme');
      }
    },
    getters: {
      isDarkTheme: state => state.isDarkTheme
      // Добавьте геттеры для данных пользователя, статуса аутентификации и т.
      // д.
    }
  }
});

```

-router – для перенаправления пользователя между разделами сайта

```

import { createRouter, createWebHistory } from 'vue-router';
import Home from '../components/Home.vue';
import Login from '../components/Login.vue';
import Register from '../components/Register.vue';
import Profile from '../components/Profile.vue';

const routes = [
  { path: '/', component: Home, props: true },
  { path: '/login', component: Login, props: true },
  { path: '/register', component: Register, props: true },
  { path: '/profile', component: Profile, props: true }
];

export const router = createRouter({
  history: createWebHistory(),
  routes
});

```

-useTheme.js – файл отвечающий за смену темы сайта и передачу данных о текущем состоянии темы

```
import { ref, watch } from 'vue';

export function useTheme() {
  const isDarkTheme = ref(localStorage.getItem('isDarkTheme') === 'true');

  function toggleTheme() {
    isDarkTheme.value = !isDarkTheme.value;
    localStorage.setItem('isDarkTheme', isDarkTheme.value.toString());
  }

  // Добавляем слежение за изменением значения isDarkTheme
  watch(isDarkTheme, (newValue, oldValue) => {
    applyTheme(newValue);
  });

  function applyTheme(isDark) {
    const themeClass = isDark ? 'dark-theme' : 'light-theme';
    const oppositeThemeClass = isDark ? 'light-theme' : 'dark-theme';

    // Устанавливаем новую тему на body
    document.body.classList.add(themeClass);
    document.body.classList.remove(oppositeThemeClass);

    // Добавляем класс с анимацией для плавной смены темы
    document.body.classList.add('theme-transition');

    // Удаляем класс с анимацией после завершения перехода
    setTimeout(() => {
      document.body.classList.remove('theme-transition');
    }, 1000); // Время анимации в миллисекундах
  }

  applyTheme(isDarkTheme.value); // Применяем текущую тему

  return { isDarkTheme, toggleTheme };
}
```


Вывод:

Проделанная работа была направлена на миграцию ранее написанного сайта на фреймворк Vue.js с соблюдением минимальных требований, включая подключение роутера, работу с внешним API, разумное деление на компоненты и использование композиционных функций (composables). Подключение роутера: Был использован Vue Router для управления маршрутизацией в приложении. Это позволило организовать навигацию между различными страницами. Работа с внешним API: Для работы с внешним API был реализован метод `fetchTopSongs`, который получает топовые песни с сервера. Это обеспечивает динамическую загрузку данных с удаленного источника. Разумное деление на компоненты: Весь проект был разделен на компоненты для повышения его читаемости, масштабируемости и обеспечения лучшей организации кода. Каждый компонент отвечает за конкретный функциональный блок и может быть переиспользован. Использование композиционных функций (composables): Для управления темой приложения был использован `composable useTheme`, который позволяет управлять темой и предоставляет доступ к текущей теме через компоненты. В целом, проект успешно соответствует требованиям и обеспечивает хорошую основу для дальнейшего развития. Использование Vue.js значительно улучшает организацию кода, делает приложение более масштабируемым и поддерживаемым.