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:

Тюмин Никита

К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Выбранный вариант: Музыкальный плеер.

Задача: Мигрировать ранее написанный сайт на фреймворк Vue.JS.

Ход работы

В работе использовался vue-router для маршрутизации:

@/routes/routes.js

```
import HomePage from "../views/HomePage.vue";
import ProfilePage from "../views/ProfilePage.vue";
import SearchPage from "../views/SearchPage.vue";

import LoginPage from "../views/auth/LoginPage.vue";
import RegisterPage from "../views/auth/RegisterPage.vue";

import NotFoundPage from "../views/NotFoundPage.vue";

import SecureComponent from "../components/SecureComponent.vue";
import PublicComponent from "../components/PublicComponent.vue";

export default [
    {
        path: '',
        meta: { requiresAuth: true },
        component: SecureComponent,
        children: [
            {
                path: '',
                name: 'home',
                component: HomePage
            },
            {
                path: '/profile',
                name: 'profile',
                component: ProfilePage
            },
            {
                path: '/search',
                name: 'search',
                component: SearchPage
            },
        ],
    },
    {
        path: '',
        meta: { blockedForAuthenticated: true },
        component: PublicComponent,
        children: [
            {
                path: '/login',
                name: 'login',
                component: LoginPage,
            },
            {
                path: '/register',
                name: 'register',
                component: RegisterPage,
            },
```

```
        ],
    },
    {
        name: '404',
        path: '/:catchAll(.*)',
        component: NotFoundPage
    }
]
```

@/router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import routes from './routes.js'
import { auth } from '@/firebase/firebase'
import db from "../services/db";
import useAuthStore from "../pinia/auth";

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: routes,
})

router.beforeEach(async (to, from) => {
  let user
  let authStore = useAuthStore()

  user = authStore.user
  if (!user) {
    user = await new Promise((resolve, reject) =>
      auth.onAuthStateChanged(
        user => {
          if (user) {
            resolve(user)
          } else {
            resolve(null)
          }
        },
        error => reject(error)
      )
    )
    if (user) {
      let userData = await db.find('users', user.uid)
      authStore.setUser(userData)
    }
  }

  if(to.meta.requiresAuth && !user) {
    return {
      name: 'login'
    }
  }
  if(to.meta.blockedForAuthenticated && user) {
    return {
      name: 'home'
    }
  }
})

export default router
```

Перед рендером страницы роутер проверяет, аутентифицирован ли пользователь, и разрешает или блокирует доступ к защищенным страницам соответственно.

В качестве менеджера состояния используется pinia. Всего в приложении два store'а для работы с проигрываемыми треками и аутентификацией пользователя.

@/pinia/auth.js:

```
import { defineStore } from 'pinia'

import { auth } from '@/firebase/firebase'
import {browserSessionPersistence, setPersistence, signInWithEmailAndPassword,
signOut} from "firebase/auth";
import router from "../router";

const useAuthStore = defineStore('auth', {
  state: () => ({
    user: null,
  }),
  getters: {
    isAuthenticated() {
      return this.user !== null
    }
  },
  actions: {
    setUser(user) {
      this.user = user
    },
    login(user) {
      setPersistence(auth, browserSessionPersistence)
        .then(() => {
          signInWithEmailAndPassword(auth, user.email, user.password).then(res =>
{
            router.push({name: 'home'})
          }).catch(err => {
            console.log(err)
          })
        })
        .catch((error) => {
          const errorCode = error.code;
          const errorMessage = error.message;
        });
    },
    logout() {
      signOut(auth).then(() => {
        router.push({name: 'login'})
        this.$reset()
      }).catch((error) => {
        // An error happened.
      });
    }
  },
  persist: true,
})

export default useAuthStore
```

@/pinia/player.js:

```
import { defineStore } from 'pinia'

const usePlayerStore = defineStore('player', {
  state: () => ({
    currentSong: null,
    playlist: [],
  }),
  getters: {
    getCurrentSong() {
      return this.playlist[this.currentSong]
    },
  },
  actions: {
    setCurrentSong(song) {
      this.playlist = [song]

      this.currentSong = 0
    },
    addToPlaylist(song) {
      this.playlist.push(song)
    },

    setCurrentPlaylist(playlist, play = true) {
      this.playlist = playlist

      if (play) {
        this.currentSong = 0
      }
    },

    playNextSong() {
      if (this.playlist.length > this.currentSong + 1) {
        this.currentSong += 1
      }
    },
    playPreviousSong() {
      if (this.currentSong > 0) {
        this.currentSong -= 1
      }
    },
  },
  persist: true,
})

export default usePlayerStore
```

Для сохранения состояния используется библиотека pinia-plugin-persistedstate. Сохранение состояние позволяет сохранять сессию аутентификации и проигрываемый плейлист после перезагрузки страницы.

Логика проигрывания треков и добваления их в очередь использовалась в различных компонентах, поэтому было решено вынести ее в composable-функцию:

@/composables/songActions:

```
import usePlayerStore from "../src/pinia/player";

export function useSongActions(props) {
  const playerStore = usePlayerStore()

  function play() {
    playerStore.setCurrentSong(props.song)
  }

  function addToCurrentPlaylist() {
    playerStore.addToPlaylist(props.song)
  }

  return { play, addToCurrentPlaylist }
}
```

Пример использования:

@/components/songs/SongCardComponent.vue:

```
<script setup>
import {useSongActions} from "../../../composables/songActions";
import {defineProps} from "vue";

const props = defineProps({
  song: {type: Object, required: true},
})

const { play, addToCurrentPlaylist } = useSongActions(props)
</script>
```

Вывод:

В ходе Лабораторной работы был реализован музыкальный плеер на Vue.js, с использованием роутера, pinia и composable-функций.