САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет инфокоммуникационных технологий

Образовательная программа 09.03.03

Направление подготовки (специальность) Мобильные сетевые технологии

О Т Ч Е Т

по курсовой работе

Тема задания: разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.js

Обучающийся: Кондрашов Егор, K33402

Руководитель: Добряков Д. И., преподаватель

Оценка за курсовую работу ____

Дата ____

Санкт-Петербург
2022

ВВЕДЕНИЕ

Актуальность

Сервисы для бронирования жилья позволяют людям гораздо проще найти привлекательный вариант размещения, в то же время упрощая работу сдающим жильё. В связи с постепенным снятием ограничений, связанных с COVID, сервисы для бронирования жилья возвращают свою востребованность (например, акции booking.com за последние 6 месяцев выросли на 13%).

Использование Vue.js для разработки одностраничного приложения оправдано в первую очередь тем, что одностраничные приложения, как правило, быстрее загружают страницы сайта, что ведёт к лучшему восприятию пользователем процесса взаимодействия с ним.

Цели и задачи

1. Определение средств разработки
2. Определение функциональных требований
3. Проектирование и реализация фронтенда
4. Интеграция Vuex и миксинов

ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1.1 Средства разработки

Для разработки фронтенда был использован фреймворк Vue.js, так как его изучению была посвящена часть курса “Фронд-энд разработка”. Также для специализированного функционала использовались библиотеки Axios, BootstrapVue, Vue router и Vuex

1.2 Функциональные требования

1. Набрать минимально осмысленное число функциональных страниц, покрывающее следующий функционал: логин, регистрация, сброс пароля, редактирование данных профиля, личный кабинет, детальная страница сущности системы (их может быть несколько, по количеству важных сущностей), страница с поиском/фильтрацией по важным сущностям системы (их также может быть несколько).
2. Желательно использовать миксины.
3. Желательно использовать Vuex.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ФРОНТЕНДА

1.1 Проектирование фронтенда

Основной сущностью веб-приложения по выбранной мной тематике является отель. Действия в приложении могут совершать зарегистрированные пользователи, поэтому нужно предусмотреть процессы регистрации и авторизации. У пользователя должен быть личный кабинет. Кроме того, для связи пользователя и отеля существует сущность бронирование, с помощью которой осуществляется главный функционал приложения.

Исходя из этого, был определён следующий список страниц, необходимый к реализации:

1. Регистрация
2. Вход
3. Личный кабинет
4. Изменение информации в ЛК
5. Смена пароля
6. Главная страница/страница поиска по отелям
7. Страница с результатами поиска
8. Страница отеля с возможностью сделать бронирование
9. Мои бронирования

1.2 Реализация фронтенда

На каждой странице приложения должна присутствовать шапка проекта с навигацией. Навигацию между страницами удобно осуществлять с помощью библиотеки Vue Router.

Содержание файла router/index.js, который определяет доступные в приложении маршруты:

```
import Index from "@/views/Index.vue"
import Vue from "vue"
import VueRouter from "vue-router"

Vue.use(VueRouter)
const routes = [
 {
   path: "/",
   name: "Index",
   component: Index
 },
…
```

Теперь менять текущую страницу программно можно следующим образом:

```
this.$router.push({ name: "Index" })
```

Или так, в компоненте b-link:

```
<b-link
      :to="{ name: 'HotelDetail', params: { id: booking.hotel.id }
}"
      class="link-primary"
      >Перейти на страницу отеля</b-link
    >
```

Для простого и быстрого создания адаптивной верстки страницы можно применить библиотеку Bootstrap Vue. Рассмотрим это на примере страницы редактирования профиля:

```
<b-container class="my-profile">
   <b-row class="justify-content-center"
     ><h2>Редактирование профиля</h2></b-row
   >
   <b-row class="justify-content-center">
     <div class="col-10">
       <b-form @submit="onSubmit">
         <b-form-group
           id="email-group"
           label="Адрес электронной почты"
           label-for="email-input"
         >
           <b-form-input
             id="email-input"
```

```
        v-model="profile.email"
        type="email"
        class="col-4"
        placeholder="Адрес электронной почты"
      ></b-form-input>
    </b-form-group>

    <b-form-group
      id="first-name-group"
      label="Имя"
      label-for="first-name-input"
    >
      <b-form-input
        id="first-name-input"
        v-model="profile.first_name"
        type="text"
        class="col-4"
        placeholder="Имя"
      ></b-form-input>
    </b-form-group>

    <b-form-group
      id="last-name-group"
      label="Фамилия"
      label-for="last-name-input"
    >
      <b-form-input
        id="last-name-input"
        v-model="profile.last_name"
        type="text"
        class="col-4"
        placeholder="Фамилия"
      ></b-form-input>
    </b-form-group>

    <b-form-group
      id="middle-name-group"
      label="Отчество"
      label-for="middle-name-input"
    >
      <b-form-input
        id="middle-name-input"
        v-model="profile.middle_name"
```

```
                type="text"
                class="col-4"
                placeholder="Отчество"
              ></b-form-input>
            </b-form-group>

            <b-form-group
              id="birthdate-group"
              label="Дата рождения"
              label-for="birthdate-input"
            >
              <b-form-input
                id="birthdate-input"
                v-model="profile.birthdate"
                type="date"
                class="col-4"
                placeholder="Дата рождения"
              ></b-form-input>
            </b-form-group>

            <b-button class="col-6 btn" type="submit"
              >Изменить данные профиля</b-button
            >
          </b-form>
        </div>
      </b-row>
    </b-container>
```

Для отправки формы на бэкенд, используем библиотеку axios. Данные из формы получаем с помощью директивы v-model:

```
  data() {
    return {
      profile: Object
    }
  },
  methods: {
    async onSubmit(event) {
      event.preventDefault()
      try {
        const config = {
          headers: {
            Authorization: this.$store.getters.authHeader
          }
```

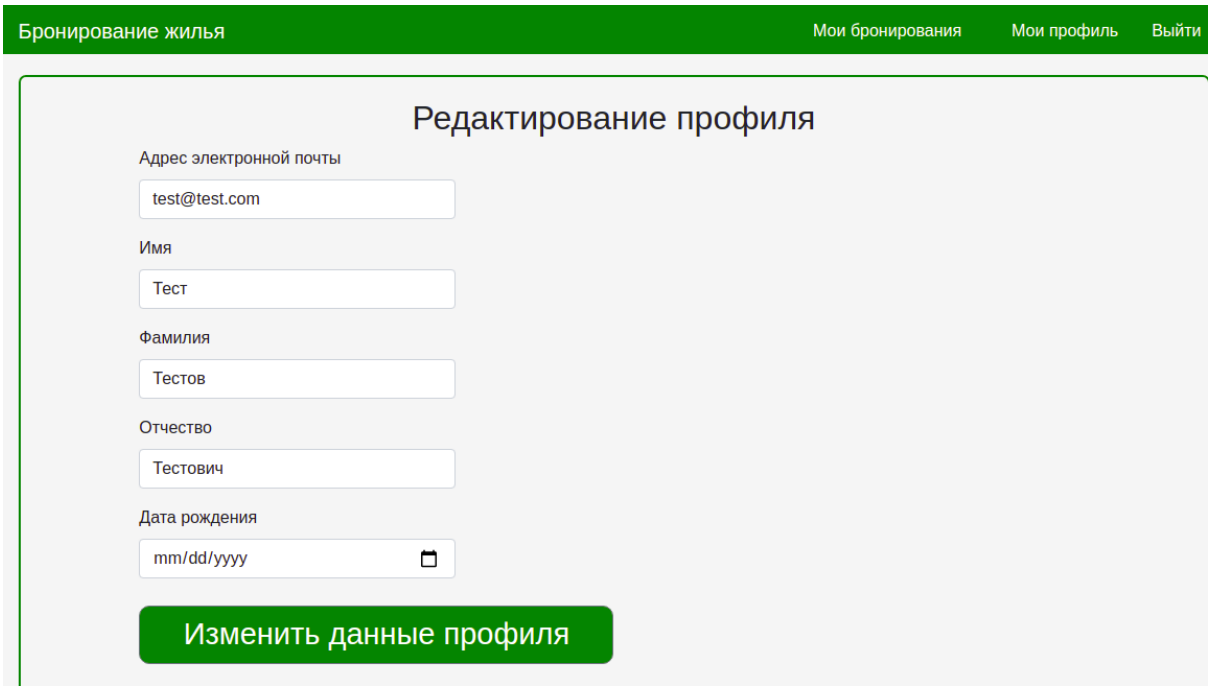
```
        }

        const resp = await this.axios.patch(
          "http://localhost:8000/api/users/me",
          this.profile,
          config
        )

        if (resp.status !== 200) {
          throw new Error(resp.error)
        }

        this.$router.push({ name: "MyProfile" })
      } catch (e) {
        console.error("Error from API: ", e)
      }
    }
  },
```

Соберём представление редактирования профиля из компонент EditProfileMain.vue и AppHeader.vue:

```
<template>
 <section class="page-content">
   <app-header />
   <edit-profile-main />
 </section>
</template>

<script>
import AppHeader from "../components/AppHeader"
import EditProfileMain from "../components/EditProfileMain.vue"

export default {
 name: "EditProfile",

 components: {
   AppHeader,
   EditProfileMain
 }
}
</script>

<style>
```

```
</style>
```

Скриншот получившейся страницы:



Аналогично были реализованы остальные страницы.

ГЛАВА 3. ИНТЕГРАЦИЯ VUEX И МИКСИНОВ

1.1 Интеграция Vuex

Vuex - реактивное хранилище для Vue-приложений. В моем приложении Vuex используется для хранения JWT (JSON Web Token), необходимого для осуществления авторизированных запросов.

Будем хранить в состоянии токен и статус авторизации:

```
    state: {
       status: "",
       token: localStorage.getItem("token") || ""
     },
```

Для входа (получения токена) будет использоваться действие login, которое шлет запрос на бэкенд:

```
    login({ commit }, user) {
         return new Promise((resolve, reject) => {
           commit("auth_request")
           const params = new URLSearchParams()
           params.append("username", user.username)
           params.append("password", user.password)
           const config = {
             headers: {
               "Content-Type": "application/x-www-form-urlencoded"
             }
           }
                axios.post("http://localhost:8000/api/auth/jwt/login",
params, config)
             .then(resp => {
               const token = resp.data.access_token
               localStorage.setItem("token", token)
               commit("auth_success", { token: token })
               resolve(resp)
             })
             .catch(err => {
               commit("auth_error")
               localStorage.removeItem("token")
               reject(err)
             })
         })
       }
```

Для непосредственного изменения токена в состоянии используется мутация:

```
mutations: {
   auth_request(state) {
     state.status = "loading"
   },
   auth_success(state, payload) {
     state.status = "success"
     state.token = payload.token
   },
   auth_error(state) {
     state.status = "error"
   },
   logout(state) {
     state.status = ""
     state.token = ""
   }
 },
```

Получать токен для авторизации в запросах из компонент можно с помощью геттера:

```
 getters: {
   isLoggedIn: state => !!state.token,
   authStatus: state => state.status,
   authHeader: state => `Bearer ${state.token}`
 },
```

Теперь проверять авторизацию можно следующим образом:

```
<make-booking-form v-if="this.$store.getters.isLoggedIn" />
```

1.2 Интеграция миксинов

Миксины во Vue.js - способ дать доступ разным компонентам к общему функционалу и не повторять один и тот же код в разных местах.

В моем приложении, на странице “Мой профиль”, а также на странице “Редактировать профиль” необходимо получать информацию о профиле текущего пользователя. Для этого можно использовать миксин.

Код файла src/mixins/getMyProfile.js:

```
export default {
 name: "getMyProfile",
```

```
  methods: {
    async getMyProfile() {
      try {
        const config = {
          headers: {
            Authorization: this.$store.getters.authHeader
          }
        }
        const resp = await this.axios.get(
          "http://localhost:8000/api/users/me",
          config
        )

        if (resp.status !== 200) {
          throw new Error(resp.error)
        }

        this.profile = resp.data
      } catch (e) {
        console.error("Error from API: ", e)
      }
    }
  }
}
```

Теперь подключаем этот миксин в файлы необходимых компонент следующим образом:

```
<script>
import getMyProfile from "@/mixins/getMyProfile.js"

export default {
 name: "EditProfileMain",
 mixins: [getMyProfile],
```

ЗАКЛЮЧЕНИЕ

Выводы по работе

В ходе выполнения этой работы я получил практический опыт использования Vue.js для создания Single Page Application. Также я разобрался, как для реализации своего приложения применить популярные библиотеки: Vue Router для навигации, Axios для отправки запросов на бэкенд, Bootstrap Vue для создания адаптивной верстки, Vuex для хранения состояния авторизации. Кроме того, я стал использовать больше инструментов Vue.js: директивы v-if/v-else, миксины.

СПИСОК ЛИТЕРАТУРЫ

1. Документация Vue.js [Электронный ресурс] https://vuejs.org/v2/guide/
2. Документация Vue Router [Электронный ресурс] https://router.vuejs.org/guide/
3. Документация Vuex [Электронный ресурс] https://vuex.vuejs.org/ru/guide/
4. Документация BootstrapVue [Электронный ресурс] https://bootstrap-vue.org/docs
5. Документация Axios [Электронный ресурс] https://axios-http.com/docs/intro