

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ

УНИВЕРСИТЕТ ИТМО

Факультет инфокоммуникационных технологий

Образовательная программа 09.03.03

Направление подготовки (специальность) Мобильные сетевые технологии

О Т Ч Е Т

о курсовой работе

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

Обучающийся Поляков Андрей Алексеевич К33402

Руководитель: Добряков Давид Ильич

Оценка за курсовую работу ____

Подписи членов комиссии:

____ (Добряков Д. И.)
(подпись)

Дата ____

Санкт-Петербург
2022

| | |
|--|----|
| ВВЕДЕНИЕ | 3 |
| Актуальность | 3 |
| Цели и задачи | 3 |
| ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ | 4 |
| 1.1 Средства разработки | 4 |
| 1.2 Функциональные требования | 4 |
| ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ | 4 |
| 2.1. Проектирование и реализация клиентской части | 4 |
| 2.2. Реализация страниц | 9 |
| ЗАКЛЮЧЕНИЕ | 14 |
| Выводы по работе | 14 |
| СПИСОК ЛИТЕРАТУРЫ | 15 |

ВВЕДЕНИЕ

Актуальность

С учетом распространенности метеорологических станций и интернета на планете земля, люди привыкли полагаться на интернет-ресурсы с прогнозами погоды для составления своего гардероба и определения своего распорядка дня. Использование Vue.js для разработки одностраничного приложения оправдано в первую очередь тем, что одностраничные приложения, как правило, быстрее загружают страницы сайта, а скорость получения информации является важным аспектом жизни человека в информационный век.

Цели и задачи

1. Определение средств разработки
2. Определение функциональных требований
3. Проектирование и реализация клиентской части

ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1.1 Средства разработки

Для реализации клиентской части был использован фреймворк Vue.js. Также для специализированного функционала использовались библиотеки Axios, BootstrapVue, Vue router, Vuetify, vue-toastification и vuex.

Для разработки серверной части я использовал Django и навыки полученные в ходе курса веб разработки.

1.2 Функциональные требования

Функциональные требования по этому проекту заключались в:

- Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.js
- Использование миксинов (необязательно, но желательно)
- Использование Vuex (необязательно, но желательно)
- В проекте должно быть, как минимум, 10 страниц (обязательно)

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

2.1. Проектирование и реализация клиентской части

Используемые стек технологий: vue, vuex, router-vue, axios, vuetify, vue-toastification

```
1  <template>
2    <v-app>
3      <Header/>
4      <v-main class="d-flex align-center text-center">
5        <router-view />
6      </v-main>
7    </v-app>
8  </template>
9
10 <script>
11 import Header from '@components/Header'
12 export default {
13   name: 'App',
14   components: { Header }
15 }
16 </script>
```

(app.vue)

Подключен router для направлений по страницам.

```
1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  import Home from '@/views/Home.vue'
4  import Login from '@/views/Log.vue'
5  import Account from '@/views/Acc.vue'
6  import Weather from '@/views/Search.vue'
7  import Register from '@/views/Reg.vue'
8  import LogOut from '@/views/LogOut.vue'
9
10  Vue.use(VueRouter)
11
12  const routes = [
13    {
14      path: '/',
15      name: 'Home',
16      component: Home
17    },
18    {
19      path: '/login',
20      name: 'Login',
21      component: Login
22    },
23    {
24      path: '/register',
25      name: 'Register',
26      component: Register
27    },
28    {
29      path: '/account',
30      name: 'Account',
31      component: Account
32    },
33    {
34      path: '/search',
35      name: 'Search',
36      component: Weather
37    },
38    {
39      path: '/logout',
40      name: 'Logout',
41      component: LogOut
42    }
  ]
```

```
43   ]  
44  
45   ✓ const router = new VueRouter({  
46     mode: 'history',  
47     base: process.env.BASE_URL,  
48     routes  
49   })  
50  
51   export default router  
52
```

Так же используется store для хранения данных о городах пользователя.

```

1  import Vue from 'vue'
2  import Vuex from 'vuex'
3  import axios from 'axios'
4
5  Vue.use(Vuex)
6
7  const store = new Vuex.Store({
8    state: {
9      infos: [],
10     id: null,
11     mainID: null
12   },
13   mutations: {
14     SET_INFOS_TO_STATE: (state, infos) => {
15       state.infos = infos
16     },
17     SET_IDS_TO_STATE: (state, id) => {
18       state.id = id
19     },
20     SET_MAIN_IDS_TO_STATE: (state, mainID) => {
21       state.mainID = mainID
22     }
23   },
24   actions: {
25     GET_INFOS_FROM_API ({ commit }) {
26       const token = localStorage.getItem('token')
27       return axios('http://127.0.0.1:8000/api/cities/preferences/', {
28         method: 'GET',
29         token: token,
30         headers: {
31           Authorization: 'Token ' + token
32         }
33       })
34       .then((infos) => {
35         commit('SET_INFOS_TO_STATE', infos.data)
36         return (infos.data)
37       })
38       .catch((error) => {
39         console.log(error)
40       })
41     },
42     GET_CITY_IDS ({ commit }) {
43       const token = localStorage.getItem('token')
44       return axios('http://127.0.0.1:8000/api/cities/preferences/', {
45         method: 'GET',
46         token: token

```

```

47     headers: {
48       Authorization: 'Token ' + token
49     }
50   })
51   .then((infos) => {
52     for (let i = 0; i < infos.data.length; i += 1) {
53       commit('SET_IDS_TO_STATE', infos.data[i].city.id)
54       return (infos.data[i].city.id)
55     }
56   })
57   .catch((error) => {
58     console.log(error)
59   })
60 },
61 GET_CITY_MAIN_IDS ({ commit }) {
62   const token = localStorage.getItem('token')
63   return axios('http://127.0.0.1:8000/api/cities/preferences/', {
64     method: 'GET',
65     token: token,
66     headers: {
67       Authorization: 'Token ' + token
68     }
69   })
70   .then((infos) => {
71     for (let i = 0; i < infos.data.length; i += 1) {
72       commit('SET_MAIN_IDS_TO_STATE', infos.data[i].id)
73       return (infos.data[i].id)
74     }
75   })
76   .catch((error) => {
77     console.log(error)
78   })
79 },
80 },
81 getters: {
82   INFOS (state) {
83     return state.infos
84   },
85   ID (state) {
86     return state.id
87   },
88   MAINID (state) {
89     return state.mainID
90   }
91 }

```


2.2. Реализация страниц

Всего в приложении реализовано 5 страниц: главная страница, регистрация, вход, личный кабинет, страница поиска.

Главная страница (если мы не вошли или не зарегистрированы)

Прогноз погоды

ВХОД

РЕГИСТРАЦИЯ

Привет!
Это твой прогноз погоды!

Страница регистрации

Прогноз погоды

ВХОД

РЕГИСТРАЦИЯ

Имя пользователя

Пароль

Email

Имя

Фамилия

РЕГИСТРАЦИЯ

Страница входа

Имя пользователя

Пароль

ВХОД

Личный кабинет (если нет городов)

Сохраненные города

Что-то тут пусто :(

Чтобы добавить новые города - перейдите во вкладку "Поиск"

Личный кабинет (с городами)

Сохраненные города

ГОРОД MOSCOW

Текущая температура: -4.3°c

Ощущается как: -8.4°c

Скорость ветра: 2.75 м/с

Удалить город

ГОРОД SAINT PETERSBURG

Текущая температура: -4.92°c

Ощущается как: -9.41°c

Скорость ветра: 3 м/с

Удалить город

Страница поиска погоды в определенном городе (или по геолокации пользователя)

Поиск

москва

Поиск



Город: Москва

Текущая температура: -4.3 °c

Ощущается как: -8.4 °c

Скорость ветра: 2.75 м/с

Сохранить город

Демонстрация работы библиотеки vue-toastification

Поиск

москва

Поиск



Город: Москва

Текущая температура: -4.3 °c

Ощущается как: -8.4 °c

Скорость ветра: 2.75 м/с

Сохранить город

Город добавлен

Рассмотрим страницу личного кабинета:

```

1  <template>
2    <div class="container">
3      <div class="pt-3 d-flex justify-content-center">
4        <h2 class="head-text">Сохраненные города</h2>
5      </div>
6      <div class="v-city_list" v-if="this.$store.state.infos.length > 0">
7        <City
8          v-for="info in this.$store.state.infos"
9          :key="info.id"
10         :info="info"
11         :id="info.id"
12       />
13      </div>
14      <div class="mt-3 no_cities" v-else-if="this.$store.state.infos.length === 0">
15        <p class="head-text">Что-то тут пусто :(</p>
16        <p class="head-text">Чтобы добавить новые города – перейдите во вкладку "Поиск"</p>
17      </div>
18    </div>
19  </template>
20
21  <script>
22    import City from './City.vue'
23    import { mapActions } from 'vuex'
24    export default {
25      components: { City },
26      name: 'Account',
27      data: () => ({
28        info: {
29          name: ''
30        },
31        id: '',
32        mainID: ''
33      }),
34      methods: {
35        ...mapActions([
36          'GET_INFOS_FROM_API',
37          'GET_CITY_IDS',
38          'GET_CITY_MAIN_IDS'
39        ])
40      },
41      mounted () {
42        this.GET_INFOS_FROM_API()
43        this.GET_CITY_IDS()
44        this.GET_CITY_MAIN_IDS()
45      }
46    }

```

Используется v-for для отображения всех городов, сохраненных у пользователя.

Рассмотрим компоненту city:

```

1 <template>
2 <div class="container">
3   <div class="city">
4     <v-card
5       class="my-3"
6       id="weather-form"
7       elevation="2"
8     >
9       <v-list-item-content>
10        <div class="text-overline" v-if="info.name">Город {{ info.name }}</div>
11        <v-list-item-subtitle v-if="info.name">Текущая температура: {{ info.main.temp }}°C</v-list-item-subtitle>
12        <v-list-item-subtitle v-if="info.name">Ощущается как: {{ info.main.feels_like }}°C</v-list-item-subtitle>
13        <v-list-item-subtitle v-if="info.name">Скорость ветра: {{ info.wind.speed }} м/с</v-list-item-subtitle>
14      </v-list-item-content>
15      <button class="btn btn-success mb-3" id="delete" type="submit" v-if="info.name" v-on:click="removeCity(id)">Удалить город</button>
16    </v-card>
17  </div>
18 </div>
19 </template>
20
21 <script>
22 import axios from 'axios'
23 export default {
24   name: 'City',
25   props: [
26     'info',
27     'id'
28   ],
29   methods: {
30     getWeather () {
31       axios
32         .get('http://api.openweathermap.org/data/2.5/weather?id=' + this.info.city.id + '&appid=62f76307202d2bbb00f83a4de8ac7393&units=metric')
33         .then(response => (this.info = response.data))
34     },
35     async removeCity (id) {
36       try {
37         const token = localStorage.getItem('token')
38         if (token) {
39           this.axios.defaults.headers.common.Authorization = `token ${token}`
40         }
41         const response = await axios
42           .delete('http://127.0.0.1:8000/api/cities/preferences/' + id + '/', {
43             token: token
44           }, this.info.id)
45         location.reload()
46         if (response.status !== 201) {
47           if (response.status !== 201) {
48             throw new Error(response.status)
49           }
50         } catch (e) {
51           console.error('AN API ERROR', e)
52         }
53       },
54       mounted () {
55         this.getWeather()
56       }
57     }
58   }
59 </script>

```

В эту компоненту поступает информация о городе, погоду для которого нужно отобразить. Для поиска погоды идет обращение через axios к внешнему открытому api openweathermap. Так же в этой компоненте реализовано удаление городов из личного кабинета пользователя.

ЗАКЛЮЧЕНИЕ

Выводы по работе

Благодаря выполненной работе я теперь понимаю как взаимодействует клиентская и серверная части между собой. Также я получил навыки взаимодействия с фреймворком Vue и некоторыми его библиотеками (vuex, vue-router, vuetify, vue-toastification).

СПИСОК ЛИТЕРАТУРЫ

1. Документация Django — <https://docs.djangoproject.com/en/4.0/>
2. Документация Django REST Framework — <https://www.django-rest-framework.org>
3. Документация Vue.JS - <https://ru.vuejs.org/v2/guide/>
4. Документация Vuetify - <https://vuetifyjs.com/en/>
5. Документация vue-toastification - <https://github.com/Maronato/vue-toastification/tree/main>
6. Документация Vue-Router - <https://router.vuejs.org/guide/>