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
ОТЧЕТ
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.)
Дата

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

```
<template>
       <v-app >
         <Header/>
         <v-main class="d-flex align-center text-center">
          <router-view />
         </v-main>
       </v-app>
     </template>
     <script>
     import Header from '@/components/Header'
11
12
     export default {
       name: 'App',
13
14
       components: { Header }
     </script>
```

(app.vue)

Подключен router для направлений по страницам.

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

Так же используется store для хранения данных о городах пользователя.

```
import Vue from 'vue'
     import Vuex from 'vuex'
     import axios from 'axios'
     Vue.use(Vuex)
     const store = new Vuex.Store({
       state: {
         infos: [],
         id: null,
11
         mainID: null
12
       },
       mutations: {
         SET_INFOS_TO_STATE: (state, infos) => {
15
          state.infos = infos
         SET_IDS_TO_STATE: (state, id) => {
17
          state.id = id
20
         SET_MAIN_IDS_TO_STATE: (state, mainID) => {
21
           state.mainID = mainID
22
23
       },
       actions: {
25
         GET_INFOS_FROM_API ({ commit }) {
26
           const token = localStorage.getItem('token')
           return axios('http://127.0.0.1:8000/api/cities/preferences/', {
28
             method: 'GET',
29
             token: token,
             headers: {
               Authorization: 'Token ' + token
             }
           })
             .then((infos) => {
               commit('SET_INFOS_TO_STATE', infos.data)
               return (infos.data)
             })
             .catch((error) => {
               console.log(error)
             })
40
         GET_CITY_IDS ({ commit }) {
           const token = localStorage.getItem('token')
            return axios('http://127.0.0.1:8000/api/cities/preferences/', {
             method: 'GET',
```

```
headers: {
                Authorization: 'Token ' + token
50
            })
              .then((infos) => {
                for (let i = 0; i < infos.data.length; i += 1) {</pre>
                  commit('SET_IDS_TO_STATE', infos.data[i].city.id)
53
54
                  return (infos.data[i].city.id)
              })
              .catch((error) => {
              console.log(error)
              })
60
          GET_CITY_MAIN_IDS ({ commit }) {
62
            const token = localStorage.getItem('token')
            return axios('http://127.0.0.1:8000/api/cities/preferences/', {
64
              method: 'GET',
              token: token,
              headers: {
                Authorization: 'Token ' + token
            })
70
              .then((infos) => {
71
                for (let i = 0; i < infos.data.length; i += 1) {</pre>
                  commit('SET_MAIN_IDS_TO_STATE', infos.data[i].id)
72
73
                  return (infos.data[i].id)
74
75
              })
76
              .catch((error) => {
              console.log(error)
78
              })
79
        },
        getters: {
82
          INFOS (state) {
            return state.infos
84
          },
          ID (state) {
            return state.id
86
87
          },
         MAINID (state) {
           return state.mainID
```

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)



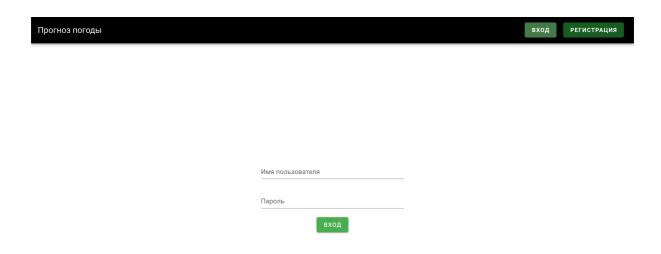
Привет!

Это твой прогноз погоды!

Страница регистрации

Прогноз погоды		l	вход	РЕГИСТРАЦИЯ
	Имя пользователя			
	Пароль			
	Email			
	Имя			
	Фамилия			
	РЕГИСТРАЦИЯ			

Страница входа



Личный кабинет (если нет городов)

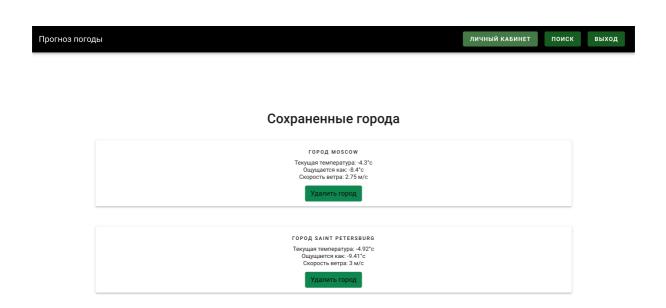


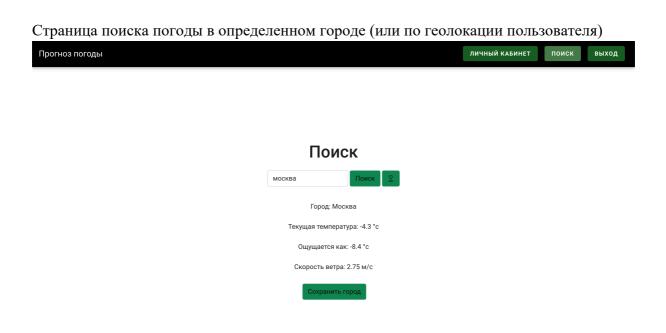
Сохраненные города

Что-то тут пусто :(

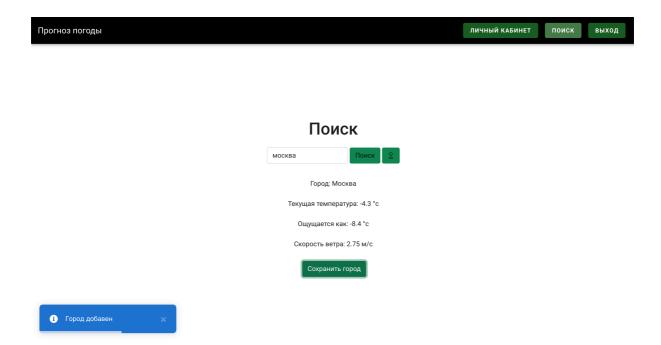
Чтобы добавить новые города - перейдите во вкладку "Поиск"

Личный кабинет (с городами)





Демонстрация работы библиотеки vue-toastification



Рассмотрим страницу личного кабинета:

```
<template>
  <div class="container">
    <div class="pt-3 d-flex justify-content-center">
     <h2 class="head-text">Сохраненные города</h2>
    <div class="v-city_list" v-if="this.$store.state.infos.length > 0">
       v-for="info in this.$store.state.infos"
       :key="info.id"
       :info="info"
       :id="info.id"
    </div>
    <div class="mt-3 no_cities" v-else-if="this.$store.state.infos.length === 0">
     Что-то тут пусто :(
     Чтобы добавить новые города - перейдите во вкладку "Поиск"
    </div>
  </div>
</template>
<script>
import City from './City.vue'
import { mapActions } from 'vuex'
export default {
 components: { City },
  name: 'Account',
  data: () => ({
    info: {
     name: ''
    id: '',
   mainID: ''
  methods: {
    ...mapActions([
      'GET_INFOS_FROM_API',
      'GET_CITY_IDS',
      'GET_CITY_MAIN_IDS'
  mounted () {
    this.GET_INFOS_FROM_API()
    this.GET_CITY_IDS()
    this.GET_CITY_MAIN_IDS()
```

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

Рассмотрим компоненту city:

```
<div class="city">
                    class="my-3"
                    id="weather-form'
                   elevation="2"
                    <div class="text-overline" v-if="info.name">Город {{ info.name }}</div>
                   <v-list-item-subtitle v-if="info.name">Текущая температура: {{ info.main.temp }}°c</v-list-item-subtitle>
<v-list-item-subtitle v-if="info.name">Ощущается как: {{ info.main.feels_like }}°c</v-list-item-subtitle>
<v-list-item-subtitle v-if="info.name">Скорость ветра: {{ info.wind.speed }} м/c</v-list-item-subtitle>
               -koutton class="btn btn-success mb-3" id="delete" type="submit" v-if="info.name" v-on:click="removeCity(id)">Удалить город</button>
          props: [
          methods: {
            getWeather () {
                 .get('http://api.openweathermap.org/data/2.5/weather?id=' + this.info.city.id + '&appid=62f76307202d2bbb00f83a4de8ac7393&units=metric
                 .then(response => (this.info = response.data))
            async removeCity (id) {
                 const token = localStorage.getItem('token')
if (token) {
                    this.axios.defaults.headers.common.Authorization = `token ${token}`
                 const response = await axios
   .delete('http://127.0.0.1:8000/api/cities/preferences/' + id + '/', {
                    token: token
}, this.info.id)
                 location.reload()
                  if (response.status !== 201) {
46
47
48
49
50
51
52
53
54
55
56
57
58
                    throw new Error(response.status)
               } catch (e) {
                 console.error('AN API ERROR', e)
          mounted () {
            this.getWeather()
```

В эту компоненту поступает информация о городе, погоду для которого нужно отобразить. Для поиска погоды идет обращение через axios к внешнему открытому арі орепweathermap. 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/