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: Мобильные и сетевые технологии

Отчет

О курсовой работе

Тема: Разработка клиентской части сервиса поиска погоды средствами фреймворка Vue.js

Обучающийся: Легин Денис Анатольевич

Уч. Группа: k33402

Руководитель: Добряков Давид Ильич

Оценка за курсовую работу: _____

Подписи членов комиссии: _____
(Добряков. Д.И)

Дата: 28.01.2022

Санкт-Петербург

2022

Оглавление

Введение

Актуальность

С приходом технологий люди всё меньше обращаться за помощью к древним устоям и обычаям, так, например гораздо удобнее взять свое устройство с выходом в интернет и на соответствующем ресурсе посмотреть прогноз погоды на день, чем вспомнить примету и выйдя на улицу смотреть на высоту полета ласточек. Использование Vue.js для разработки одностраничного приложения оправдано тем, что одностраничные приложения, как правило быстрее загружаются браузерами. Помимо этого, Vue.js является очень удобным инструментом для создания сайта

Цели и задачи

1. Определение технологического стека
2. Определение функциональных требований
3. Проектирование и реализация клиентской части

Глава 1. Технологический стек. Функциональные требования

Технологический стек

Для реализации клиентской части был использован фреймворк Vue.js. Помимо этого, для расширения возможностей фрймворка использовались библиотеки: Axios, Bootstrap Vue, Vue router, Vuetify, Vue-toastification, vuex

Для разработки серверной части использовался фреймворк Django и дополнительные библиотеки необходимые для работы с бекендом.

Функциональные требования

Функциональные требования к проекту заключались в:

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

Глава 2. Проектирование и реализация

2.1 Проектирование и реализация клиентской части

App.vue

```vue
<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'
export default {
  name: 'App',
  components: { Header },
  data: () => ({
    //
  })
}
</script>
```

Router.js для маршрутизации между страницами.

```
  {
    path: '/account',
    name: 'Account',
    component: Account
  },
  {
    path: '/search',
    name: 'Search',
    component: Weather
  },
  {
    path: '/logout',
    name: 'Logout',
    component: LogOut
  }
]

const router = new VueRouter( options: {
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

export default router
```

Store для хранения данных о сохраненных городов пользователя.

```
import Vue from 'vue'
import Vuex from 'vuex'
import axios from 'axios'

Vue.use(Vuex)

const store = new Vuex.Store( options: {
  state: {
    infos: [],
    id: null,
    mainID: null
  },
  mutations: {
    SET_INFOS_TO_STATE: (state, infos) => {
      state.infos = infos
    },
    SET_IDS_TO_STATE: (state, id) => {
      state.id = id
    },
    SET_MAIN_IDS_TO_STATE: (state, mainID) => {
      state.mainID = mainID
    }
  },
  actions: {
```

```
actions: {
  GET_INFOS_FROM_API ({ commit }) {
    const token = localStorage.getItem( key: 'token')
    return axios('http://127.0.0.1:8000/api/cities/preferences/', {
      method: 'GET',
      token: token,
      headers: {
        Authorization: 'Token ' + token
      }
    }) AxiosPromise<any>
      .then((infos : AxiosResponse<any> ) => {
        commit('SET_INFOS_TO_STATE', infos.data)
        return (infos.data)
      }) Promise<AxiosResponse<any>>
      .catch((error) => {
        console.log(error)
      })
  },
```

Реализация личного кабинета. Используется v-for для отображения всех городов, сохраненных у пользователя:

```
<template>
  <div class="container">
    <div class="pt-3 d-flex justify-content-center">
      <h2 class="head-text">Сохраненные города</h2>
    </div>
    <div class="v-city_list" v-if="this.$store.state.infos.length > 0">
      <City
        v-for="info in this.$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">
      <p class="head-text">Список пуст</p>
      <p class="head-text">Чтобы добавить новые города - перейдите во вкладку "Поиск"</p>
    </div>
  </div>
</template>

<script>
import ...
export default {
  components: { City },
  name: 'Account',
  data: () => ({
    info: {
      name: ''
    },
    id: '',
```

```vue
<script>
import ...
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()
  }
}
</script>
```

Компонент city

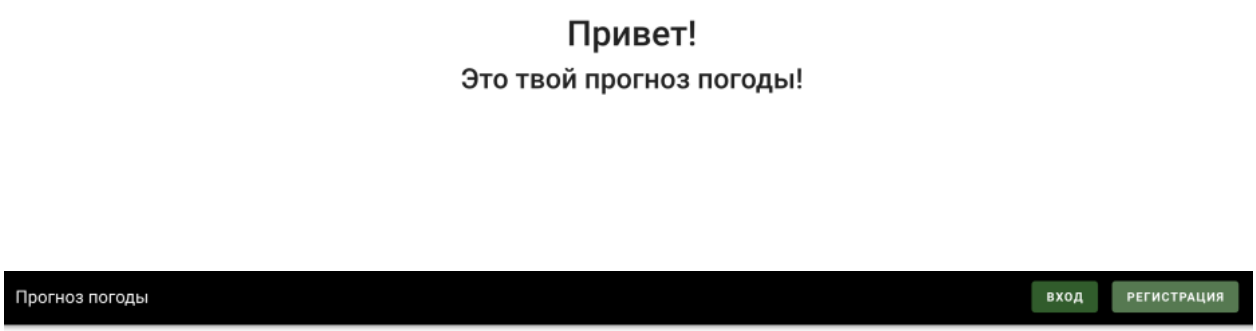
```vue
<template>
<div class="container">
  <div class="city">
     <v-card
        class="my-3"
        id="weather-form"
        elevation="2"
       >
       <v-list-item-content>
        <div class="text-overline" v-if="info.name">Город {{ info.name }}</div>
        <v-list-item-subtitle v-if="info.name">Текущая 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/с</v-list-item-subtitle>
       </v-list-item-content>
     <button class="btn btn-success mb-3" id="delete" type="submit" v-if="info.name" v-on:click="removeCity(id)">Удалить город</button>
     </v-card>
  </div>
</div>
</template>
```

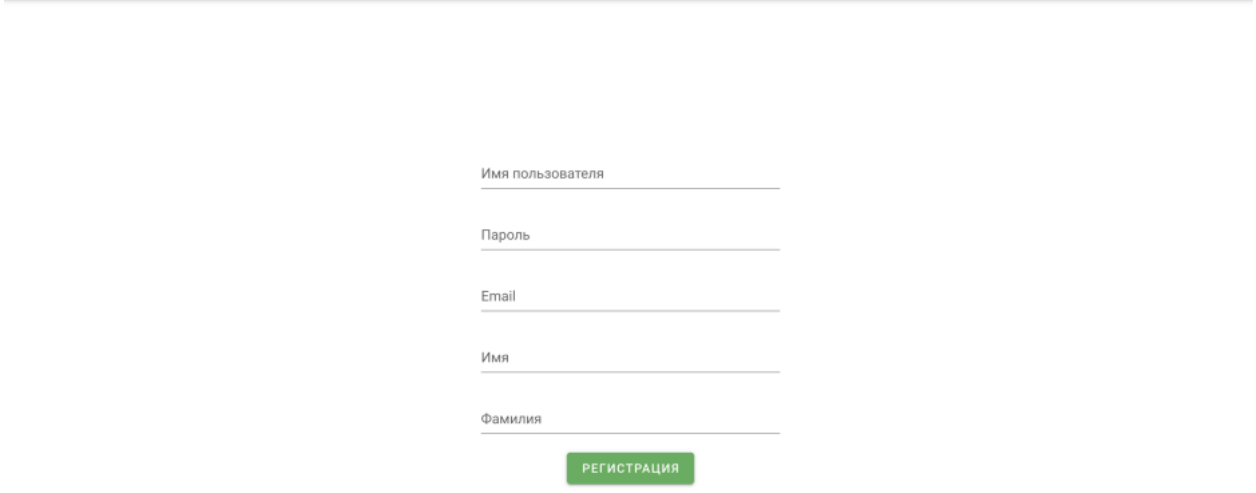
```vue
<script>
import axios from 'axios'
export default {
  name: 'City',
  props: [
    'info',
    'id'
  ],
  methods: {
    getWeather () {
      axios
        .get( url: 'https://cors-anywhere.herokuapp.com/http://api.openweathermap.org/data/2.5/weather?id=' + this.info.city.id + '&appid=62f76307202d2bbb00f83a4de8ac7393
        .then(response => (this.info = response.data))
    },
    async removeCity (id) {
      try {
        const token = localStorage.getItem( key: 'token')
        if (token) {
          this.axios.defaults.headers.common.Authorization = `token ${token}`
        }
        const response = await axios
          .delete( url: 'http://127.0.0.1:8000/api/cities/preferences/' + id + '/',  config: {
            token: token
          }, this.info.id)
        location.reload()
        if (response.status !== 201) {
          throw new Error(response.status)
        }
        const response = await axios
          .delete( url: 'http://127.0.0.1:8000/api/cities/preferences/' + id + '/',  config: {
            token: token
          }, this.info.id)
        location.reload()
        if (response.status !== 201) {
          throw new Error(response.status)
        }
      } catch (e) {
        console.error('AN API ERROR', e)
      }
    }
  },
  mounted () {
    this.getWeather()
  }
}
</script>

<style scoped>
    .cardWeather{
        background: #FFFFFF;
        box-shadow: 2px 5px 10px rgba(0, 0, 0, 0.1);
        border-radius: 10px;
        margin-top: 50px;
        margin-bottom: 10px;
        padding: 10px;
    }
</style>
```

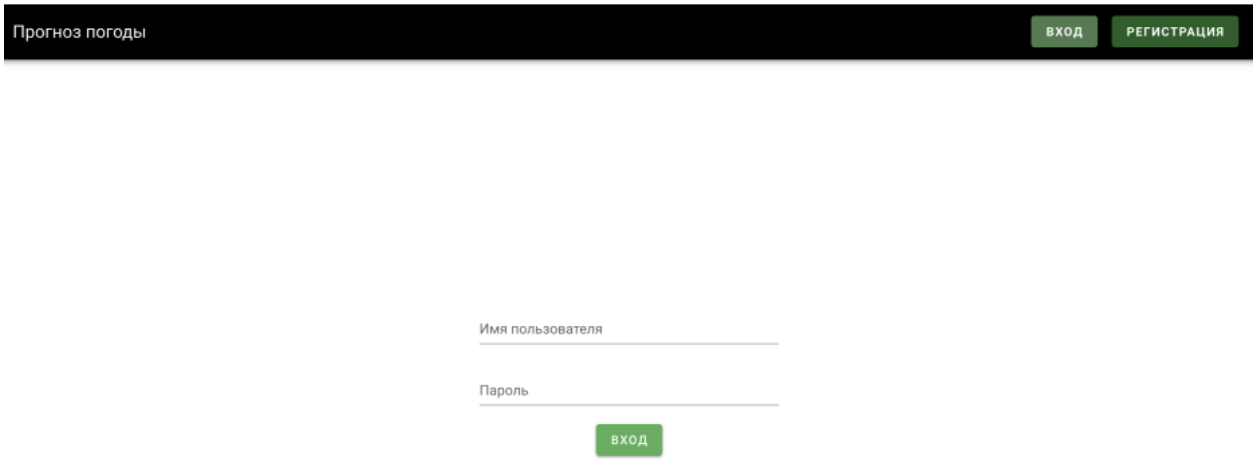
В эту компоненту поступает информация о городе, погоду для которого нужно отобразить. Для поиска погоды идет обращение через axios к внешнему открытому api openweathermap. Так же в этой компоненте реализовано удаление городов из личного кабинета пользователя

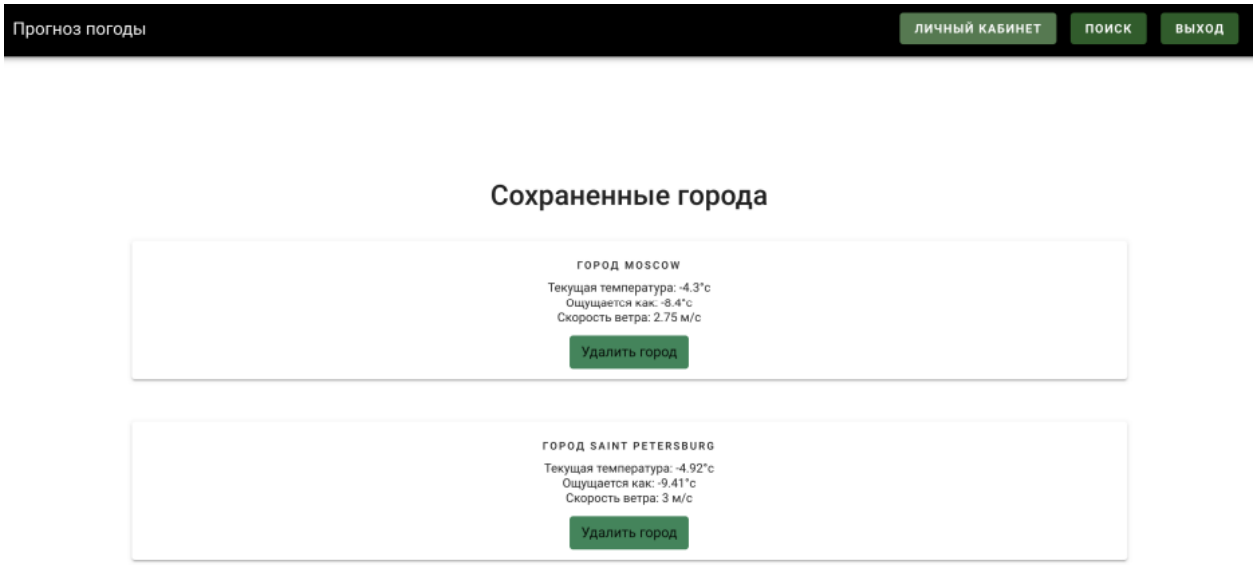
2.2 Реализация страниц

В приложении реализовано 5 полноценных страниц: главная, регистрация, вход, личный кабинет, поиск. Реализовано именно такое количество страниц за неимением достаточного количества творческого ресурса для воображения новых страниц, которые скорее всего были бы бесполезными с точки зрения user experience.

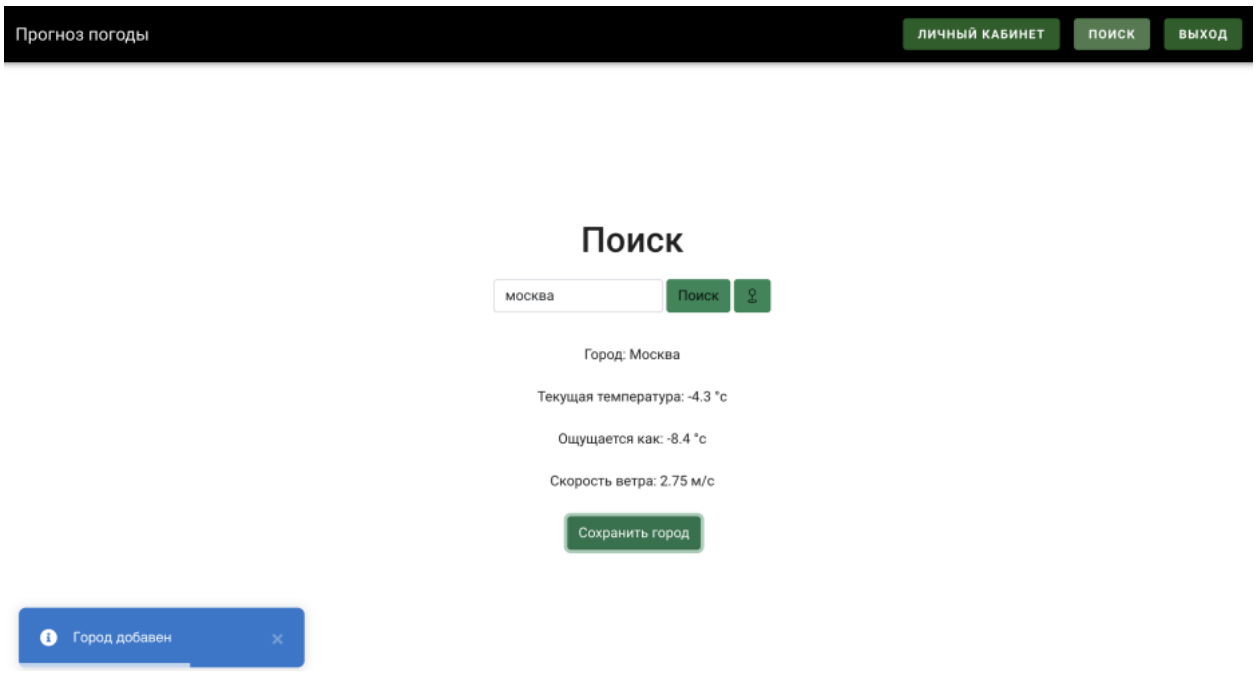








В случае отсутствия сохраненных городов, будет отображена соответствующая надпись с подсказкой для пользователя, как добавить города в этот список



Поиск работы по городу, с всплывающим toastification

Заключение

Выводы по работе

В ходе работе дополнил свои знания полученные при выполнении лабораторной работы 3, а также научился работать с библиотеками vuex, vuetify, vue-toastification. А также теперь имею представления о том, как клиентская и серверная части сайта взаимодействуют между собой.

Список литературы:

1. Документация Vue.JS - https://ru.vuejs.org/v2/guide/

2. Документация Vuetify - https://vuetifyjs.com/en/

3. Документация vue-toastification - https://github.com/Maronato/vue-toastification/tree/main

4. Документация Vue-Router - https://router.vuejs.org/guide