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

УНИВЕРСИТЕТ ИТМО

Факультет инфокоммуникационных технологий

Образовательная программа 09.03.02

Направление подготовки (специальность) Мобильные сетевые технологии

О Т Ч Е Т

о курсовой работе

Тема задания: Реализация клиентской части приложения прогноза погоды посредством Vue.Js

Обучающийся Кузгиев Адам, К33402

Руководитель: Добряков Д. И., преподаватель

Оценка за курсовую работу ____

Подписи членов комиссии:

_____ (Добряков Д. И.)
(подпись)

Дата ____

Санкт-Петербург
2021

ВВЕДЕНИЕ

Актуальность

Не знаю, насколько актуален сайт прогноза погоды для других людей, но я часто сталкивался с такой проблемой, что с ПК или ноутбука я не могу найти нужную и важную мне информацию быстро, часто такие сайты заполнены ненужно информации Из-за чего ценная информация теряется из виду

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 Средства разработки

Используемый стек технологий:

- axios
- vue
- vue-router
- vuetify
- vuex

2 Функциональные требования

Функциональные требования по этому проекту заключались в наличие 9 компонентов в

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

3 Проектирование и реализация клиентской части

Main.js
Тут подключен router, store, vuetify

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from '@/store/index'
import Vuetify from 'vuetify'
import 'vuetify/dist/vuetify.min.css'


Vue.use(Vuetify);
Vue.config.productionTip = false

new Vue({
  router,
  vuetify : new Vuetify(),
  store,
  render: h => h(App),
}).$mount( elementOrSelector: '#app')
```

Рассмотрим роутер
Реализовано 3 страницы, главная страница, детальная страница погоды, 404.
Пути:

```
Vue.use(VueRouter)

const routes = [
    {
        path: '/',
        name: 'Home',
        component: Home
    },
    {
        path: '/weather/:id',
        name: 'Weather',
        component: Weather
    },
    {
        path: '*',
        component: NotFound
    }
]

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

export default router
```

Рассмотрим app vue

```
mounted() {
  if (this.$store.state.city.data.length !== this.$store.state.weather.data.length) {
    for (var i = this.$store.state.city.data.length - 1; i >= 0; i--) {
      if (this.$store.state.city.data[i]) {
        this.$store.dispatch( type: 'ADD_WEATHER', this.$store.state.city.data[i].id)
        this.$store.commit( type: 'REMOVE_CITY', this.$store.state.city.data[i].id)
      }
    }
  }
  if (Object.keys(this.$store.state.location).length === 0) {
    this.$store.dispatch( type: 'ADD_LOCATION')
  } else {
    this.$store.dispatch( type: 'ADD_LOCAL_WEATHER')
  }
}
```

Header

```
<template>
    <v-app>
      <Header @openSnackbar="openSnackbar"></Header>
      <v-main>
        <v-container>
          <router-view></router-view>
        </v-container>
      </v-main>
      <div>
        <v-snackbar
                right
                :color="snackbarData.color"
                v-model="snackbar"
                v-if="snackbarData.text"
        >
          {{ snackbarData.text }}
          <template v-slot:action="{ attrs }">
            <v-btn
                    text
                    v-bind="attrs"
                    @click="snackbar=false"
            >
              Close
            </v-btn>
          </template>
        </v-snackbar>
      </div>
    </v-app>
</template>
```

Контейнер, который рендерит views в зависимости от роутера

```
<v-main>
  <v-container>
    <router-view></router-view>
  </v-container>
</v-main>
```

Модальное окно

```
<div>
  <v-snackbar
          right
          :color="snackbarData.color"
          v-model="snackbar"
          v-if="snackbarData.text"
  >
    {{ snackbarData.text }}
    <template v-slot:action="{ attrs }">
      <v-btn
              text
              v-bind="attrs"
              @click="snackbar=false"
      >
        Close
      </v-btn>
    </template>
  </v-snackbar>
</div>
```

Функция открытия модального окна, которая передается в хедер

```
methods: {
  openSnackbar(data = {color: 'primary', text: ''}) {
    this.snackbarData = data
    this.snackbar = true
  }
},
```

При создании корневой компоненты

```
mounted() {
  if (this.$store.state.city.data.length !== this.$store.state.weather.data.length) {
    for (var i = this.$store.state.city.data.length - 1; i >= 0; i--) {
      if (this.$store.state.city.data[i]) {
        this.$store.dispatch( type: 'ADD_WEATHER', this.$store.state.city.data[i].id)
        this.$store.commit( type: 'REMOVE_CITY', this.$store.state.city.data[i].id)
      }
    }
  }
  if (Object.keys(this.$store.state.location).length === 0) {
    this.$store.dispatch( type: 'ADD_LOCATION')
  } else {
    this.$store.dispatch( type: 'ADD_LOCAL_WEATHER')
  }
```

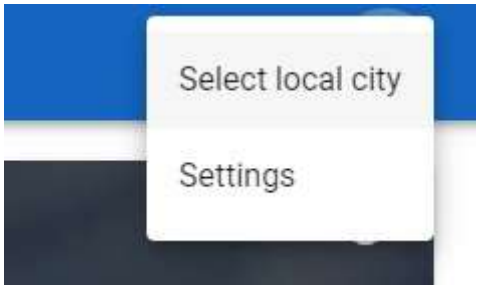
Если в локальном хранилище ничего не хранится, то вызывается метод добавления локации, которая переписывает store, если разрешена геопозиция

```
actions: {
    ADD_LOCATION(state) {
        if (!("geolocation" in navigator)) {
            return
        }
        navigator.geolocation.getCurrentPosition( successCallback: res => {
            state.commit('ADD_LOCATION', res)
            state.dispatch('ADD_LOCAL_WEATHER')
            state.commit('ADD_GETTING_LOCATION', true)
        }, errorCallback: () => {
            state.commit('ADD_LOCATION', {})
            state.commit('ADD_GETTING_LOCATION', false)
        })
    },
```

и вызывает метод получения данных из API openweathermap.org при успешном выполнении которого записывается в store

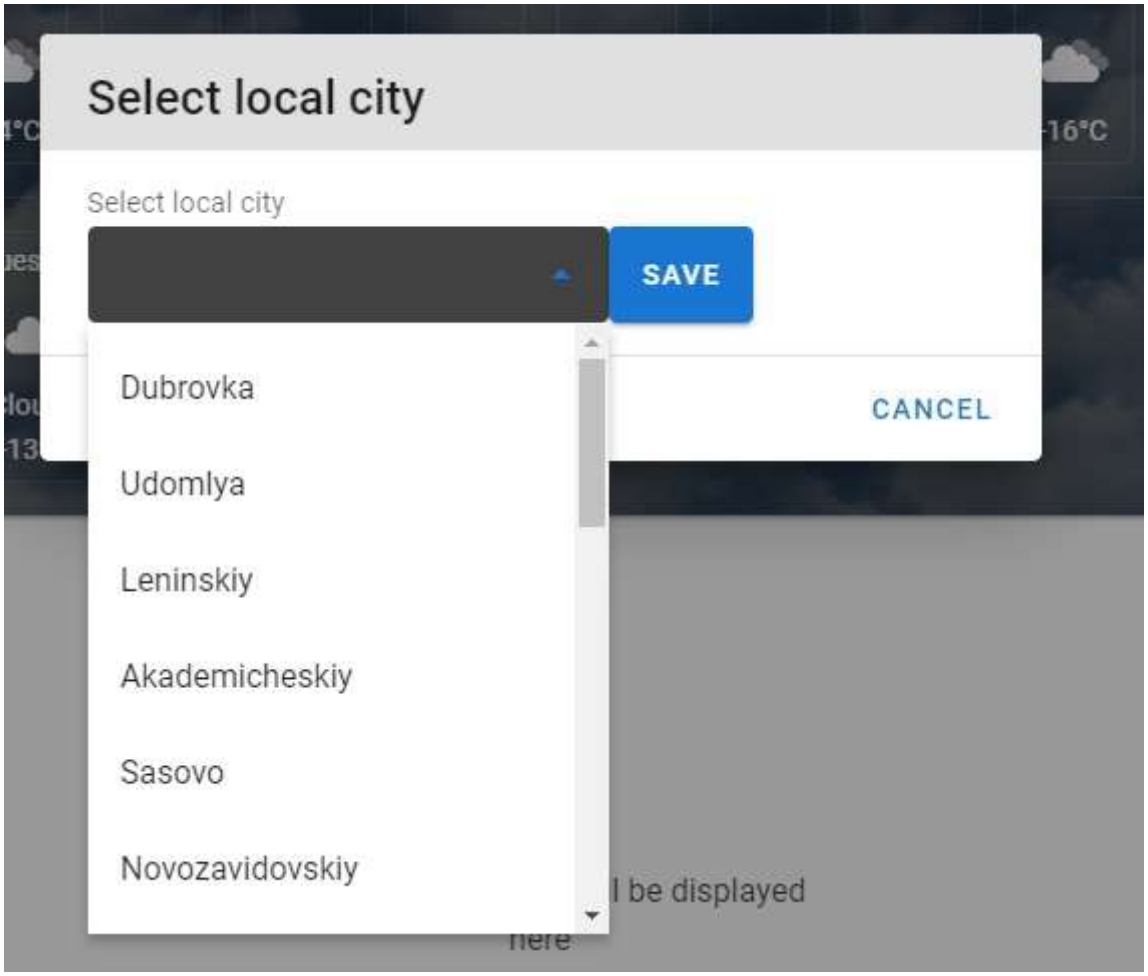
```
ADD_LOCAL_WEATHER(state) {
    state.commit('ADD_LOCAL_WEATHER_MODULE', {}, {root: true})
    if (state.rootState.location.coords) {
        axios.get( url: 'https://api.openweathermap.org/data/2.5/onecall', config: {
            params: {
                lat: state.rootState.location.coords.latitude,
                lon: state.rootState.location.coords.longitude,
                exclude: 'minutely',
                appid: '7bec99c4096420145182740l996c14fc',
                units: state.rootState.units
            }
        }).then((res : AxiosResponse<any> ) => {
            state.commit('ADD_LOCAL_WEATHER_MODULE', res.data, {root: true});
        }).catch((err) => {
            if (!err.response) {
                state.commit('ADD_LOCAL_WEATHER_MODULE', {cod: 0}, {root: true})
            }
        })
    } else {
        state.commit('ADD_LOCAL_WEATHER_MODULE', {cod: 404}, {root: true})
    }
},
```

в Хедере есть функции



С помощью этих функций можно изменить локальный город, который будет закреплен и настройки метрической системы

Вот как реализован выбор города



он из себя представляет vuetify компоненты,

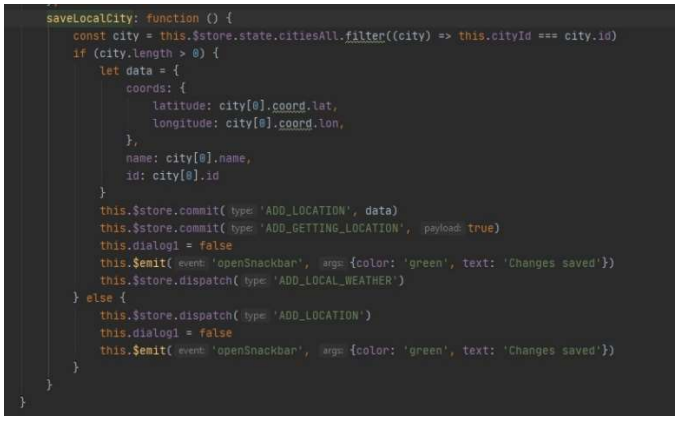
```
<v-card>
    <v-card-title class="text-h5 grey lighten-2">
        Select local city
    </v-card-title>

    <v-card-text class="py-4">
        <div>{{ this.$store.state.location.id ? 'Change local city' : 'Select local city' }}</div>
        <div class="d-flex align-stretch flex-wrap">
            <v-responsive max-width="260">
                <v-autocomplete
                        v-model="cityId"
                        :items="$store.state.citiesAll.filter(x=>x.country==='RU')"
                        item-text="name"
                        item-value="id"
                        flat
                        hide-no-data
                        hide-details
                        label="Search"
                        solo-inverted
                ></v-autocomplete>
            </v-responsive>
            <div>
                <v-btn @click="saveLocalCity" height="100%" color="primary">Save</v-btn>
                <v-btn v-if="$store.state.location.id" @click="delLocalCity" height="100%"
                       color="error">Delete
                </v-btn>
            </div>
        </div>
    </v-card-text>

    <v-divider></v-divider>

    <v-card-actions>
        <v-spacer></v-spacer>
        <v-btn
                color="primary"
                text
                @click="dialog1 = false"
        >
            Cancel
```

где массив данных, данные из store
которые получаются чтением файла списка всех городов

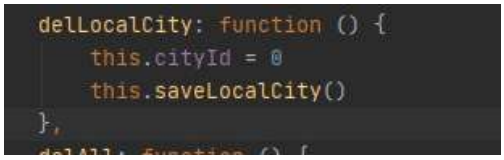
```
state: () => ({
    citiesAll: require("@/assets/city.list.min.json"),
    cities: require("@/assets/city.list.min.json"),
```

при сохранении города вызывается метод добавления локального города, также реализован метод удаления города

удаление города, обнуляет ID выбранного города, и вызывает метод для добавления



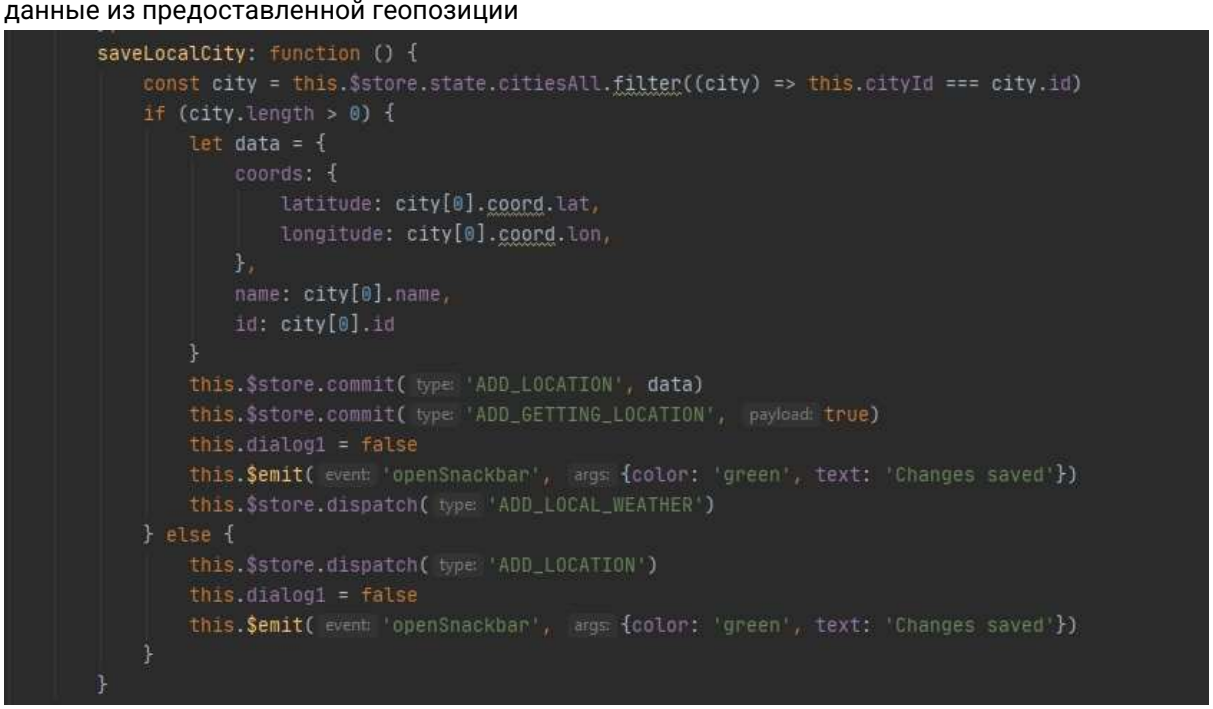
```
saveLocalCity: function () {
    const city = this.$store.state.citiesAll.filter((city) => this.cityId === city.id)
    if (city.length > 0) {
        let data = {
            coords: {
                latitude: city[0].coord.lat,
                longitude: city[0].coord.lon,
            },
            name: city[0].name,
            id: city[0].id
        }
        this.$store.commit( type: 'ADD_LOCATION', data)
        this.$store.commit( type: 'ADD_GETTING_LOCATION', payload: true)
        this.dialog1 = false
        this.$emit( event: 'openSnackbar', args: {color: 'green', text: 'Changes saved'})
        this.$store.dispatch( type: 'ADD_LOCAL_WEATHER')
    } else {
        this.$store.dispatch( type: 'ADD_LOCATION')
        this.dialog1 = false
        this.$emit( event: 'openSnackbar', args: {color: 'green', text: 'Changes saved'})
    }
}
```



```
delLocalCity: function () {
    this.cityId = 0
    this.saveLocalCity()
},
```

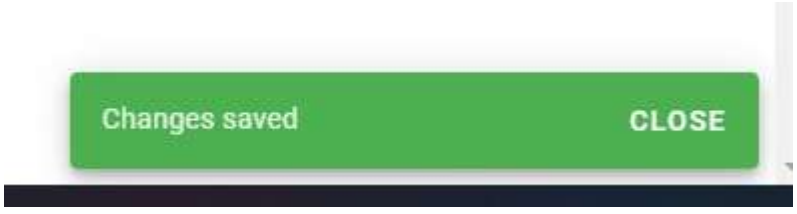
где прописано 2 эндпоинта

поиск города по id, если есть, то вызывается новый запрос с данными текущего города, иначе данные из предоставленной геопозиции



```
saveLocalCity: function () {
    const city = this.$store.state.citiesAll.filter((city) => this.cityId === city.id)
    if (city.length > 0) {
        let data = {
            coords: {
                latitude: city[0].coord.lat,
                longitude: city[0].coord.lon,
            },
            name: city[0].name,
            id: city[0].id
        }
        this.$store.commit( type: 'ADD_LOCATION', data)
        this.$store.commit( type: 'ADD_GETTING_LOCATION', payload: true)
        this.dialog1 = false
        this.$emit( event: 'openSnackbar', args: {color: 'green', text: 'Changes saved'})
        this.$store.dispatch( type: 'ADD_LOCAL_WEATHER')
    } else {
        this.$store.dispatch( type: 'ADD_LOCATION')
        this.dialog1 = false
        this.$emit( event: 'openSnackbar', args: {color: 'green', text: 'Changes saved'})
    }
}
```

При успешном добавлении вызывается успешное уведомление



Также реализована смена метрической системы, которая вызывает обновление всех данных на новую систему

```
changeUnits: function () {
    this.$store.commit( type: 'ADD_UNITS', this.units)
    this.dialog2 = false
    this.$emit( event: 'openSnackbar',  args: {color: 'green', text: 'Changes saved'})
    this.$store.dispatch( type: 'ADD_LOCAL_WEATHER')
    this.$store.dispatch( type: 'ADD_FULL_WEATHER')
    for (var i = this.$store.state.city.data.length - 1; i >= 0; i--) {
        this.$store.dispatch( type: 'ADD_WEATHER', this.$store.state.city.data[i].id)
    }
},
```

где метрическая система передается как параметр

```
actions: {
    ADD_WEATHER(state, id) {
        state.commit('ADD_WEATHER_MODULE', {id: id}, {root: true})
        if (id) {
            axios.get( url: 'https://api.openweathermap.org/data/2.5/weather',  config: {
                params: {
                    id: id,
                    appid: '7bec99c4096420145182740l996c14fc',
                    units: state.rootState['units']
                }
            }).then(res => {
                state.commit('ADD_WEATHER_MODULE', res.data, {root: true})
            }).catch((err) => {
                if (!err.response) {
                    state.commit('ADD_WEATHER_MODULE', {id: id, cod: 0}, {root: true})
                }
            })
        } else {
            state.commit('ADD_WEATHER_MODULE', {cod: 404}, {root: true})
        }
    },
```

Рассмотрим главную страницу:

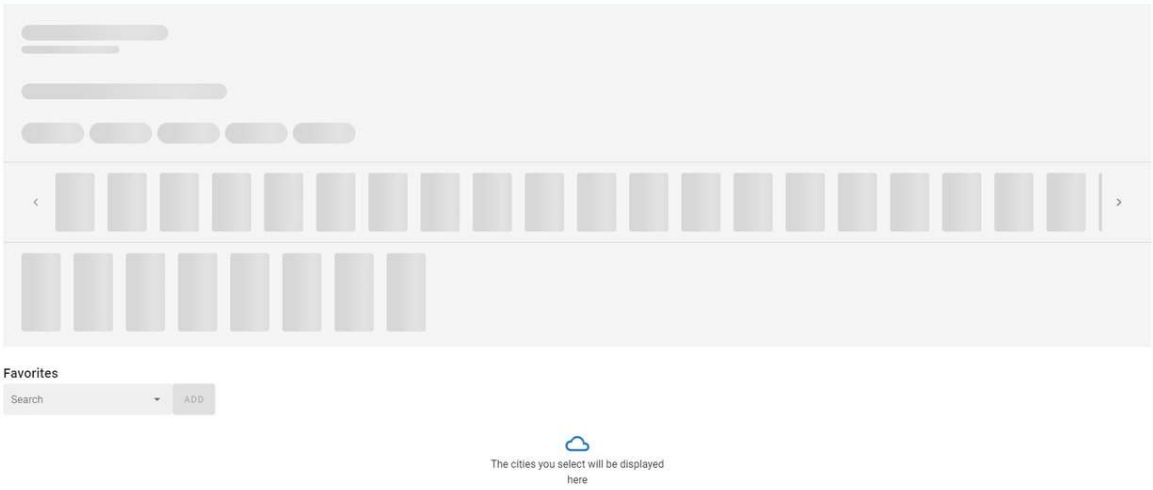
```
<v-row class="mb-2">
  <v-col v-if="$store.state.gettingLocation">
    <template v-if="$store.state.weather.local_weather.cod === 404">
      <NotFound></NotFound>
    </template>
    <template v-else-if="$store.state.weather.local_weather.cod === 0">
      <v-card class="pa-4">
        <NetworkError
            @callback="()=>{$store.dispatch( type: 'ADD_LOCAL_WEATHER')}">
        </NetworkError>
      </v-card>
    </template>
    <template v-else-if="Object.keys($store.state.weather.local_weather).length !== 0">
      <CardFullWeather :weather="$store.state.weather.local_weather"
                       :city="{name:$store.state.location.name}"
                       :path="'ADD_LOCAL_WEATHER'"></CardFullWeather>
    </template>
    <template v-else>
      <CardFullWeatherSkeletonLoader></CardFullWeatherSkeletonLoader>
    </template>
  </v-col>
```

Если в итоге запросов в store код 404, до выводится страница 404,
Если 0, то компонент ошибки запроса,
где предлагается вызвать данные снова чуть позже

```vue
<template>
    <div>
        <v-icon
                large
                color="primary"
        >
            mdi-wifi-off
        </v-icon>
        <v-list-item two-line class="px-0">
            <v-list-item-content>
                <v-list-item-title>
                    No internet connection
                </v-list-item-title>
                <v-list-item-subtitle>
                    Try again later
                </v-list-item-subtitle>
            </v-list-item-content>
        </v-list-item>
        <v-btn color="primary" @click="$emit( event: 'callback')">To retry</v-btn>
    </div>
</template>


<script>
    export default {
        name: 'NetworkError',
        props: ['callback']
    }
</script>
```

Если же, в локальном хранилище содержится нужный ключ, то рендерится погода,

Пока ответ не придет, выводится имитатор загрузки CardFullWeatherSkeletonLoader
Который имитирует загрузку данных



Также ниже имеется функция добавления города в локальный store, которая хранит данные о выбранных городах. Отправка данных вызывает
добавление города, запись его в store и в локальное хранилище

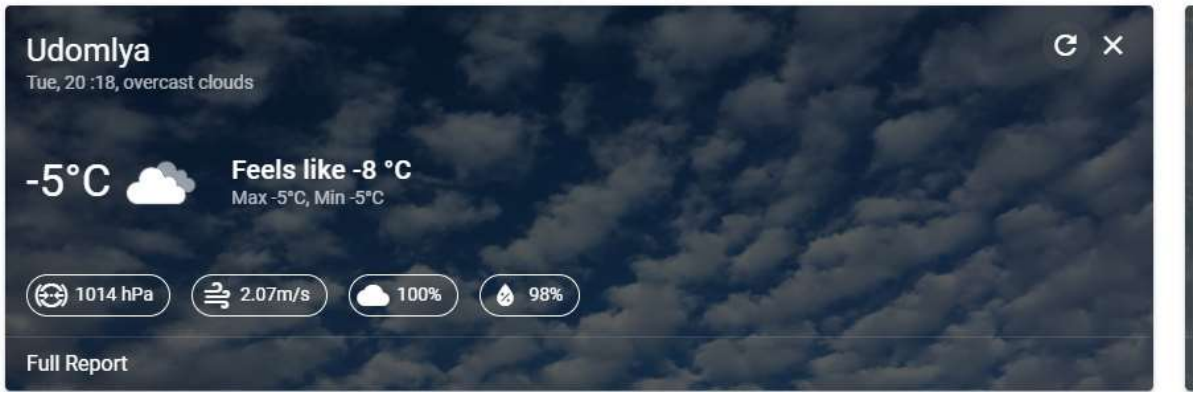
```
ADD_CITY: (state, data) => {
    state.commit('ADD_CITY_MODULE', data, {root: true})
    if (data) {
        state.commit('REMOVE_CITY', data.id, {root: true})
    }
},
```

```
ADD_CITY_MODULE: (state, data) => {
    if (data) {
        state.data.unshift(data)
    }
    localStorage.setItem('cities', JSON.stringify(state.data))
},
```

Новый город, добавленный в избранное, выглядит так

Логика поведения погоды для выбранного города повторяет поведение для локальной погоды

Есть компонент CardWeatherSkeletonLoader имитирующий загрузку для данного компонента, CardWeather имеет 2 метода, обновления данных, и удаление города, которые вызывают соответственные экшены и мутации



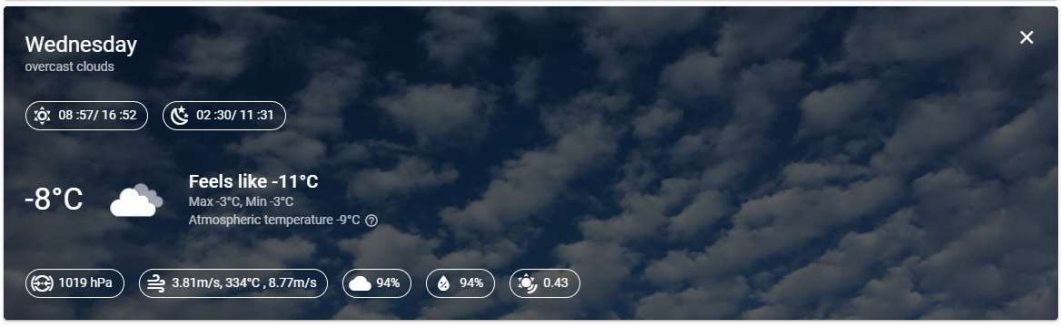

```
methods: {
    refresh_weather: function () {
        this.$store.dispatch( type: 'ADD_WEATHER', this.weather.id)

    },
    delete_weather: function () {
        this.$store.commit( type: 'REMOVE_WEATHER_MODULE', this.weather.id)
        this.$store.dispatch( type: 'REMOVE_CITY', this.weather.id)
    }
}
```

Также есть ссылка на страницу детальной погоды выбранного города

```
<v-card-actions>
    <router-link :to="'/weather/'+weather.id" class="text-decoration-none white--text px-2">
        Full Report
    </router-link>
</v-card-actions>
```

Данная страница полностью повторяет верстку и логику компонента CardFullWeather
есть кнопка обновления данных, карусель прогноза погода на 24 часа

и данные на всю неделю
При клике на час, и неделю открываются соответственные компоненты



также передается функция скрытия данных компонентов, которые возвращают данные в исходное состояние

```
delete_weather: function () {
    this.typeWeather = -1
    this.typeWeatherData = {}
},
```

Внутри компоненты реализован метод watch, который следит за изменением переданной переменной

```
watch: {
    weather: function () {
        this.time = new Date( value: this.weather.dt * 1000)
    }
}
```

также передается функция скрытия данных компонентов, которые возвращают данные в исходное состояние

```
delete_weather: function () {
    this.typeWeather = -1
    this.typeWeatherData = {}
},
```

Логика на этом заканчивается, весь остальной код отвечает за стилизацию элементов, например

```
<v-card
        class="mx-auto ma-2 pa-2"
        dark
        color="grey"
        :style="{
    backgroundPosition:'center',
    backgroundSize:'cover',
    backgroundImage: `linear-gradient( rgba(0,0,0,0.65), rgba(0,0,0,0.65)),
    url(${require('@/assets/img/'+weather.weather[0].main.toLowerCase()+'.jpg')})`
    }"
>
```

Который показывают нужную картинку исходя из данных с сервера:

ЗАКЛЮЧЕНИЕ

Выводы по работе

Цели работы были достигнуты. Были освоены необходимые средства разработки, а именно: axios, Vue, vue-router, vuetify, vuex.

Определены функциональные требования к проекту, а также спроектирована и реализована непосредственно клиентская часть приложения прогноза погоды.

СПИСОК ЛИТЕРАТУРЫ

1. Документация Vuex https://vuex.vuejs.org/ru/
2. Документация Vue-router https://router.vuejs.org/ru/api/
3. Документация VueJs https://vuejs.org/