

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: реализация клиентской части приложения прогноза погоды средствами Vue.js

Обучающийся Литвак Игорь Григорьевич, К33401

Руководитель: Добряков Давид Ильич преподаватель

Оценка за курсовую работу ____

Дата ____

Санкт-Петербург
2020

Оглавление

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. СЕРВЕРНАЯ ЧАСТЬ ПРИЛОЖЕНИЯ.....	5
2.1 Модели	5
2.2 Представления	5
2.3 Авторизация	6
ГЛАВА 3. ПРОЕКТИРОВАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ.....	7
3.1 Авторизация	7
3.2 Хранилище	8
ГЛАВА 4. РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ.....	10
4.1 Регистрация и вход.....	10
4.2 Автоматическое определение местоположения	11
4.3 Добавление любимых городов.....	12
4.4 Настройки пользователя.....	15
Вывод.....	17
Список литературы	18

Введение

Актуальность

Создать сервис погоды с нуля невероятно сложно (необходимо хранить огромную базу, собирать погодные данные и т. п.), однако с использованием внешнего API погоды эта проблема легко решается. Целью данного проекта является доказательство того что полноценное приложение прогноза погоды можно создать с минимальными усилиями и, воспользовавшись ограниченной оберткой внешнего API с кэшированием, можно достигнуть повышенного быстродействия при меньшей стоимости (меньшем числе запросов к API).

Цели и задачи

1. Проработка пользовательского интерфейса;
2. Верстка приложения средствами HTML, CSS, Bootstrap;
3. Реализация принципов доступности при верстке;
4. Реализация интерактивности приложения средствами фронтенд-фреймворка Vue.js;
5. Обеспечение работы системы авторизации, построенной на JWT.

ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1.1 Средства разработки

Используемый стек технологий:

- HTML + CSS + JS
- Vue
- Vue-router
- Vuex
- Vuex-persistedstate
- Axios
- Django + DRF + Djoser (бэкенд)

1.2 Функциональные требования

1. Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.JS
2. Использование миксинов
3. Использование Vuex
4. Минимум 10 страниц

ГЛАВА 2. СЕРВЕРНАЯ ЧАСТЬ ПРИЛОЖЕНИЯ

2.1 Модели

В серверной части реализованы 2 модели

Модель города (название + координаты):

```
6 class City(models.Model):
7     """City"""
8
9     class Meta:
10         verbose_name = "City"
11         verbose_name_plural = "Cities"
12
13         name = models.CharField("City name", max_length=50)
14         lat = models.FloatField("Latitude", validators=[MinValueValidator(-180), MaxValueValidator(180)])
15         lon = models.FloatField("Longitude", validators=[MinValueValidator(-180), MaxValueValidator(180)])
16
17     def __str__(self):
18         return str(self.name)
19
```

Модель любимого города (пользователь + город):

```
21 class FavouriteCity(models.Model):
22     """Favourite city of user"""
23
24     class Meta:
25         verbose_name = "Favourite city"
26         verbose_name_plural = "Favourite cities"
27         unique_together = ["city", "user"]
28
29         city = models.ForeignKey(City, on_delete=models.CASCADE, related_name="favourites")
30         user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="favourites")
31
32     def __str__(self):
33         return f"{self.city.name} ({self.user.username})"
34
```

2.2 Представления

Помимо обычных эндпоинтов для чтения/записи моделей и авторизации реализованы два особых эндпоинта:

- 1) Получение прогноза погоды по выбранному городу (обертка вокруг API OpenWeatherMap с кэшированием)
- 2) Нахождение ближайшего к заданным координатам города

```

18     @method_decorator(cache_page(60))
19     @action(methods=['GET'], detail=True, url_path='forecast')
20     def forecast(self, request, *args, **kwargs):
21         """Cached wrapper around OpenWeatherMap API"""
22         city = self.get_object()
23         print(f"Calling external API for city {city.name}")
24         resp = get_external_api_response(lat=city.lat, lon=city.lon)
25         return Response(resp.json(), status=resp.status_code)
26
27     @swagger_auto_schema(method='GET', query_serializer=LatLonSerializer) # noqa
28     @action(methods=['GET'], detail=False, url_path='closest')
29     def closest(self, request, *args, **kwargs):
30         """Returns city closest to certain coordinates"""
31         query_serializer = LatLonSerializer(data=request.query_params)
32         if not query_serializer.is_valid():
33             return Response(query_serializer.errors, status=400)
34         lat = query_serializer.validated_data['lat']
35         lon = query_serializer.validated_data['lon']
36         best = min(self.queryset, key=lambda city: haversine_distance(lat, lon, city.lat, city.lon))
37         return Response(self.get_serializer_class()(best).data)

```

2.3 Авторизация

Работа с пользователями и авторизация реализована при помощи библиотек Djoser и django-rest-framework-simplejwt. Способ авторизации – JSON Web Token.

ГЛАВА 3. ПРОЕКТИРОВАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ

3.1 Авторизация

Неавторизованные пользователи перенаправляются на страницу логина при помощи хуков навигации vue-router:

```
23 router.beforeEach( guard: function(to, from, next) {  
24   // Redirect to login page if not logged in and trying to access a restricted page  
25   const publicPages = ['/login', '/register'];  
26   const authRequired = !publicPages.includes(to.path);  
27   const loggedIn = store.getters.loggedIn;  
28  
29   if (authRequired && !loggedIn) {  
30     return next( to: '/login');  
31   }  
32  
33   next();  
34 });
```

При истечении действия access JWT токена он автоматически обновляется при помощи refresh JWT токена:

```

18     instance.interceptors.response.use(
19       onFulfilled: (res) => {
20         return res;
21       },
22       onRejected: async (err) => {
23         const originalConfig = err.config;
24
25         if (originalConfig.url !== "/auth/jwt/create" && err.response) {
26           // Access Token was expired
27           if (err.response.status === 401 && !originalConfig._retry) {
28             originalConfig._retry = true;
29
30             try {
31               await store.dispatch({ type: 'refreshToken' });
32               console.log("Refreshed access token");
33               return instance(originalConfig);
34             } catch (_error) {
35               return Promise.reject(_error);
36             }
37           }
38         }
39
40         return Promise.reject(err);
41       }
42     );

```

3.2 Хранилище

Конфигурация Vuex:


```

45  const store = new Vuex.Store( options: {
46    state: {
47      user: null,
48      selectedCity: null,
49      favouriteCities: [],
50      rawFavouriteCities: [],
51      allCities: [],
52      todayWeather: null,
53      weekWeather: null
54    },
55    getters: {
56      nonFavouriteCities(state) {...},
59      loggedIn(state) {...}
62    },
63    mutations: {
64      setSelectedCity(state, payload) {...},
70      setFavouriteCities(state, payload) {...},
75      setAllCities(state, payload) {...},
78      setForecast(state, payload) {...},
99      setUser(state, payload) {...},
102     unsetUser(state) {...}
109   },

```

```

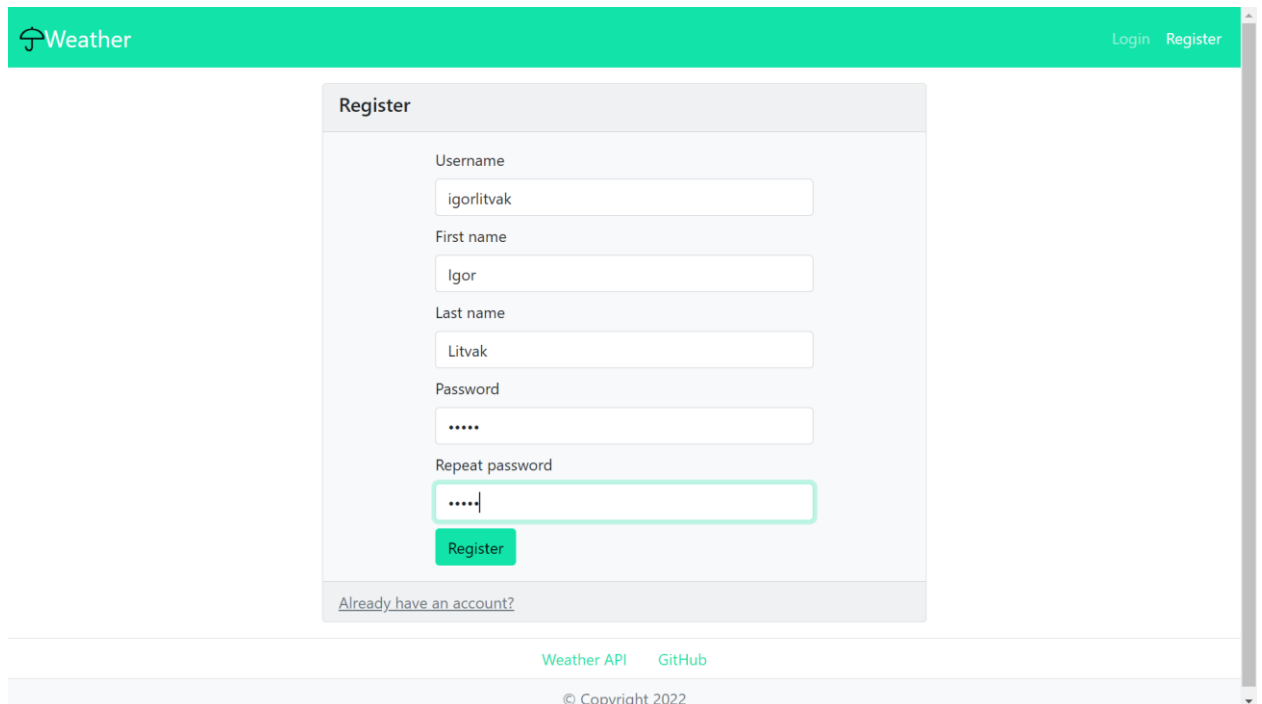
110   actions: {
111     selectCity(context, city_id) {...},
118     selectClosestCity(context, location) {...},
137     listFavouriteCities(context) {...},
152     listAllCities(context) {...},
163     addFavouriteCity(context, city_id) {...},
178     deleteFavouriteCity(context, city_id) {...},
193     getForecast(context, city_id) {...},
204     getCurrentUser(context) {...},
222     login(context, payload) {...},
238     refreshToken() {...},
251     logout(context) {...},
259     register(context, payload) {...},
270     changeName(context, payload) {...},
288     changePassword(context, payload) {...}
306   },
307   plugins: [createPersistedState( options: {
308     storage: window.sessionStorage,
309   })]
310 })
311 export default store;

```

ГЛАВА 4. РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ

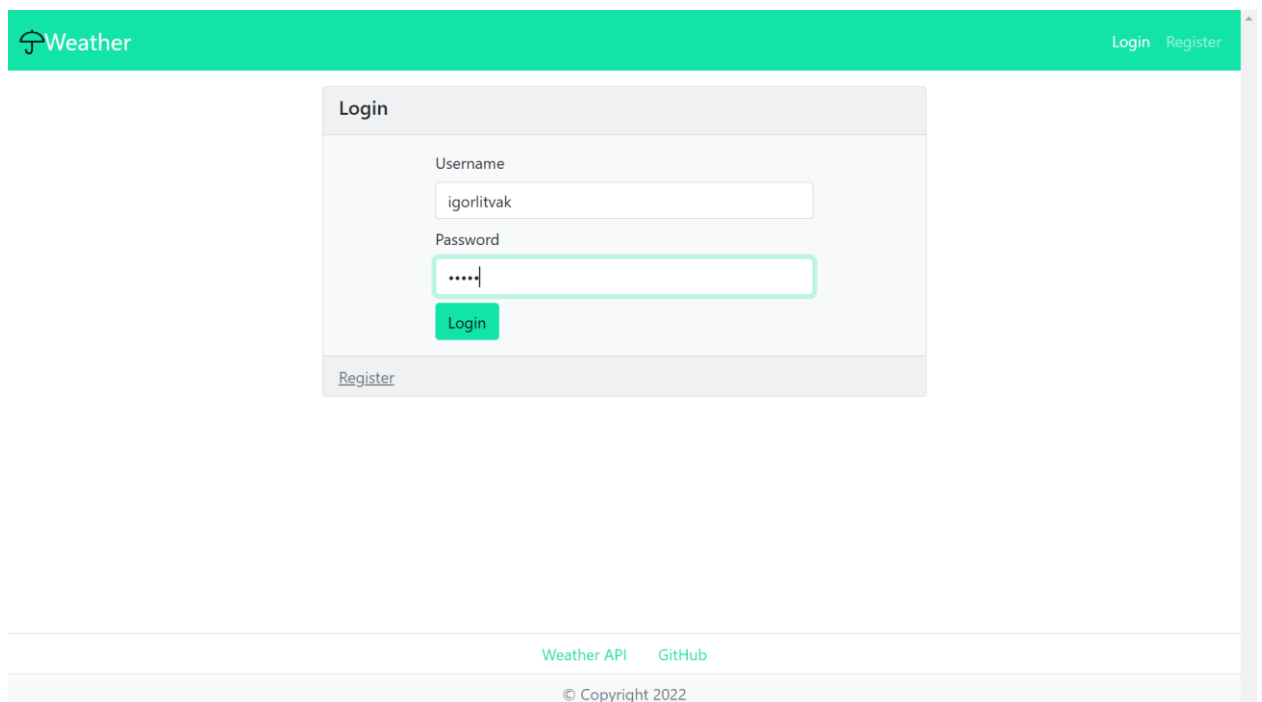
4.1 Регистрация и вход

Страница регистрации



The screenshot shows the 'Register' page of the 'Weather' application. The page has a teal header with the 'Weather' logo and 'Login Register' links. The main content area is a light gray box titled 'Register' containing several input fields: 'Username' (filled with 'igorlitvak'), 'First name' (filled with 'Igor'), 'Last name' (filled with 'Litvak'), 'Password' (filled with '.....'), and 'Repeat password' (filled with '.....'). A teal 'Register' button is at the bottom of the form. Below the form is a link 'Already have an account?'. The footer contains links for 'Weather API' and 'GitHub', and a copyright notice '© Copyright 2022'.

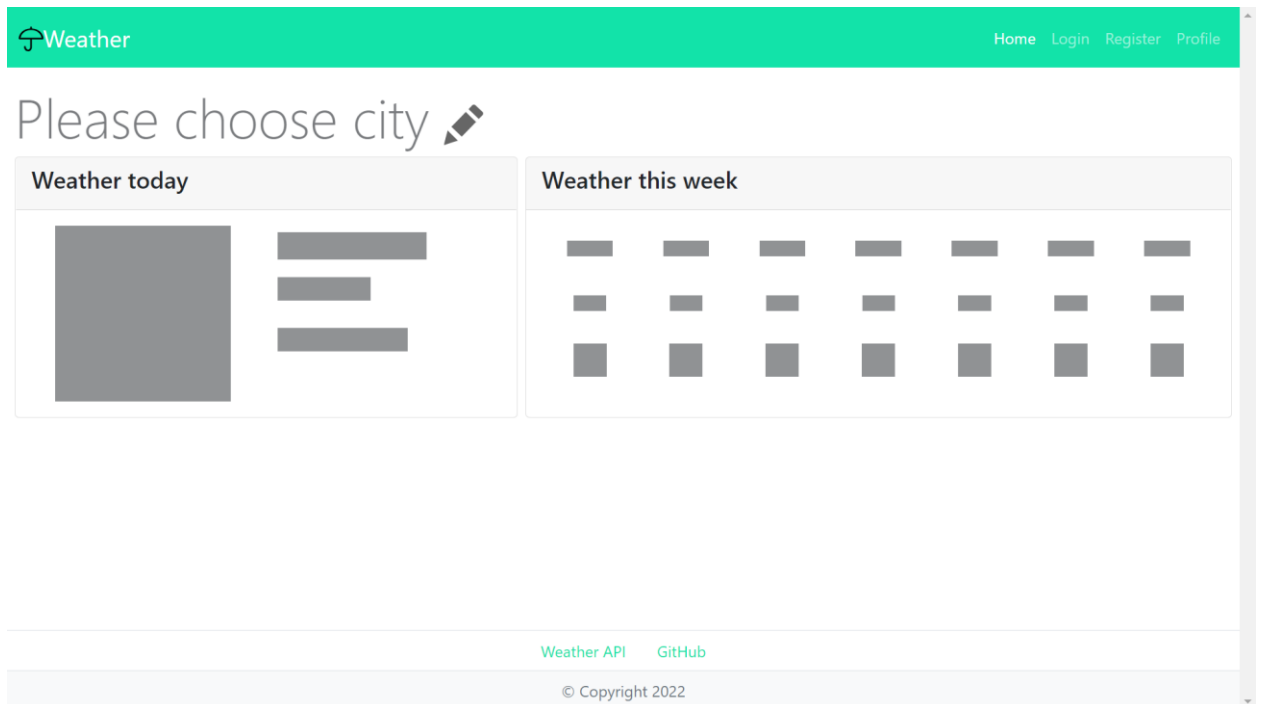
Пройдя регистрацию, пользователь может войти на сайт через страницу логина:



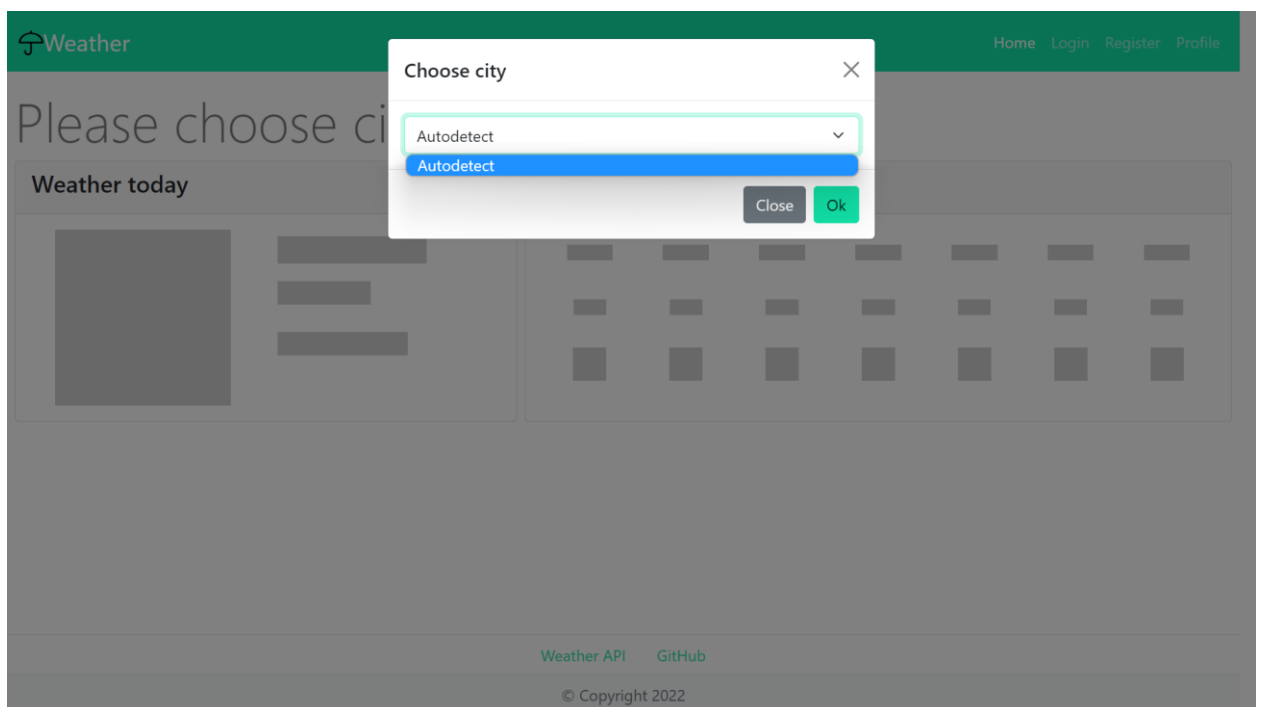
The screenshot shows the 'Login' page of the 'Weather' application. The page has a teal header with the 'Weather' logo and 'Login Register' links. The main content area is a light gray box titled 'Login' containing two input fields: 'Username' (filled with 'igorlitvak') and 'Password' (filled with '.....'). A teal 'Login' button is at the bottom of the form. Below the form is a link 'Register'. The footer contains links for 'Weather API' and 'GitHub', and a copyright notice '© Copyright 2022'.

4.2 Автоматическое определение местоположения

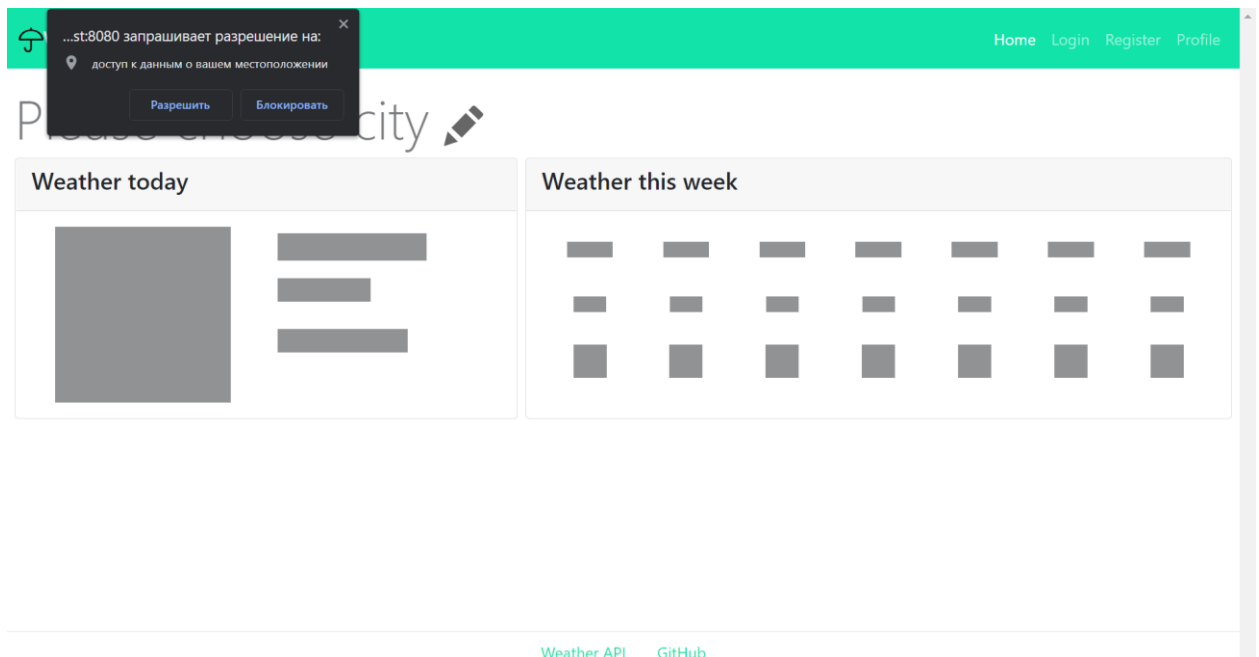
У нового пользователя не будет выбран город и для его выбора необходимо нажать на “Please choose city”



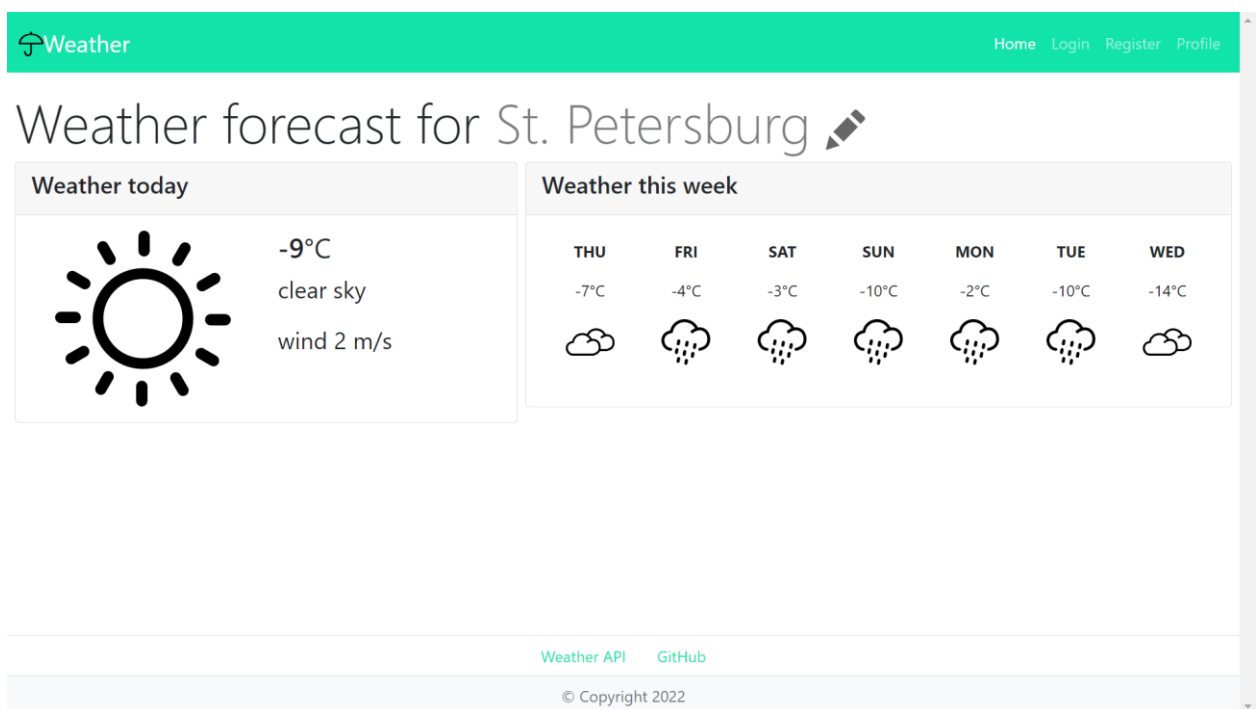
Пользователь еще не выбрал любимые города, поэтому на данный момент может воспользоваться только автоматическим определением местоположения.



Для определения координат пользователя используется Geolocation API браузера, требующий в большинстве браузеров разрешения.

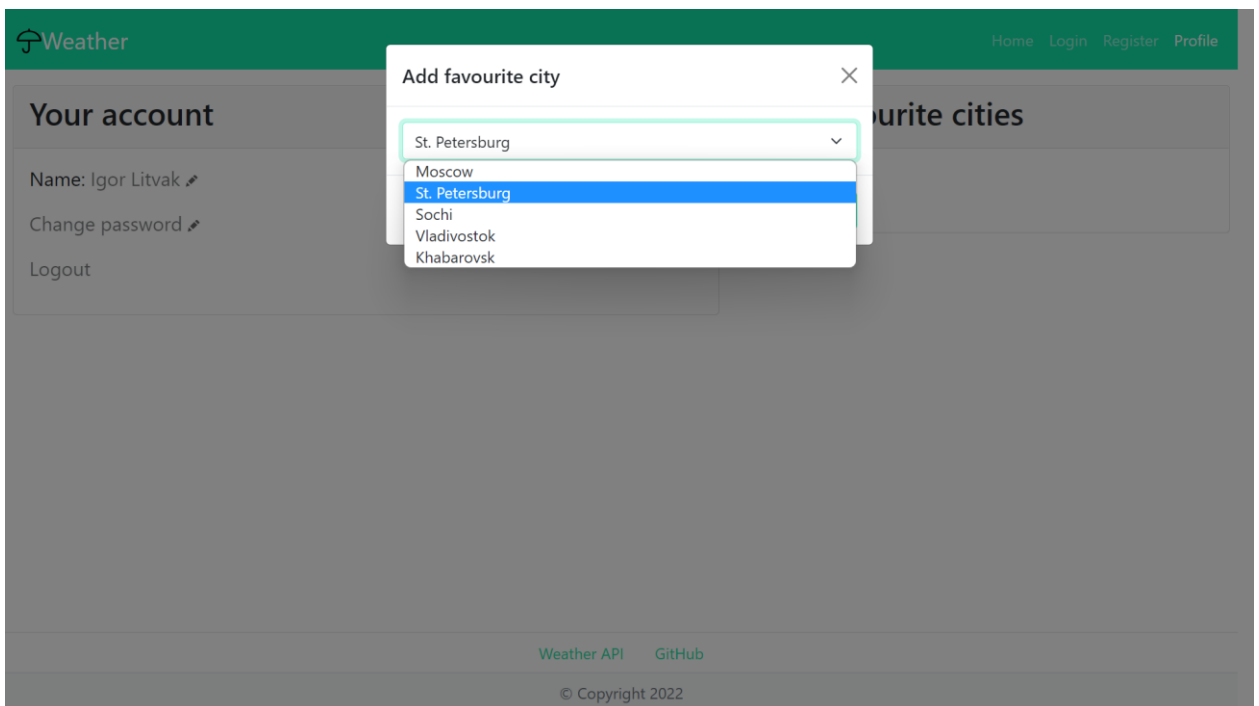
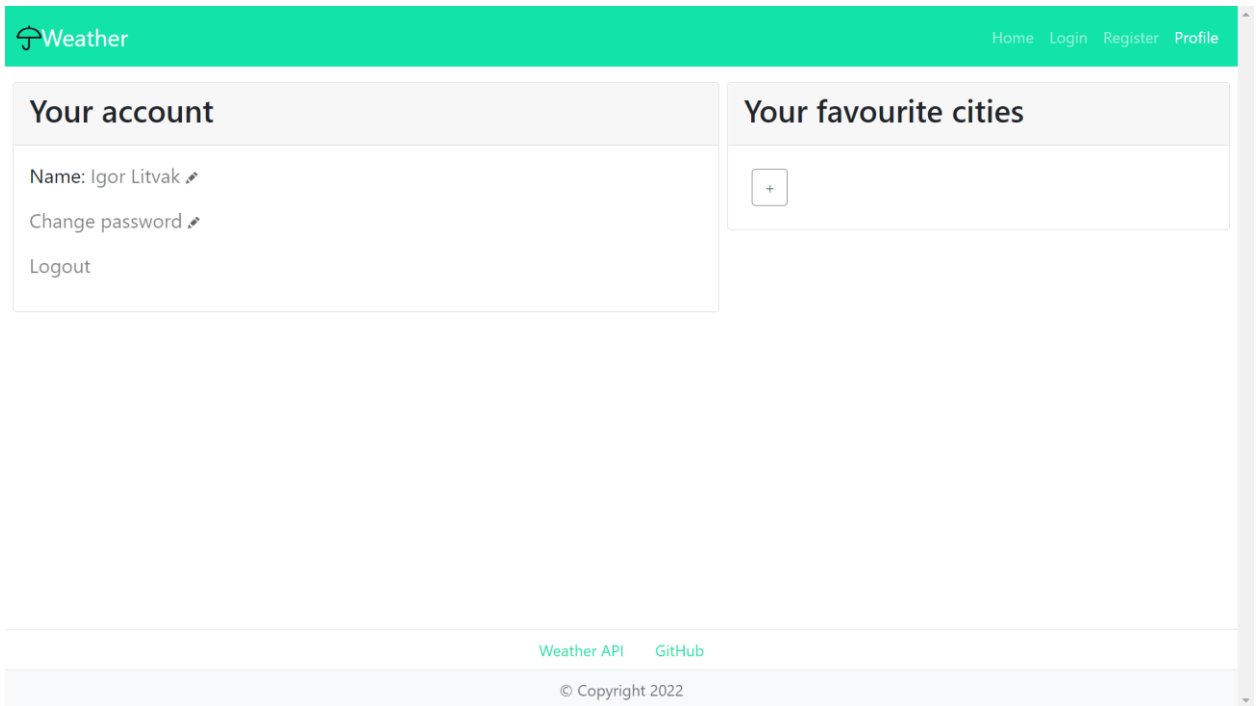


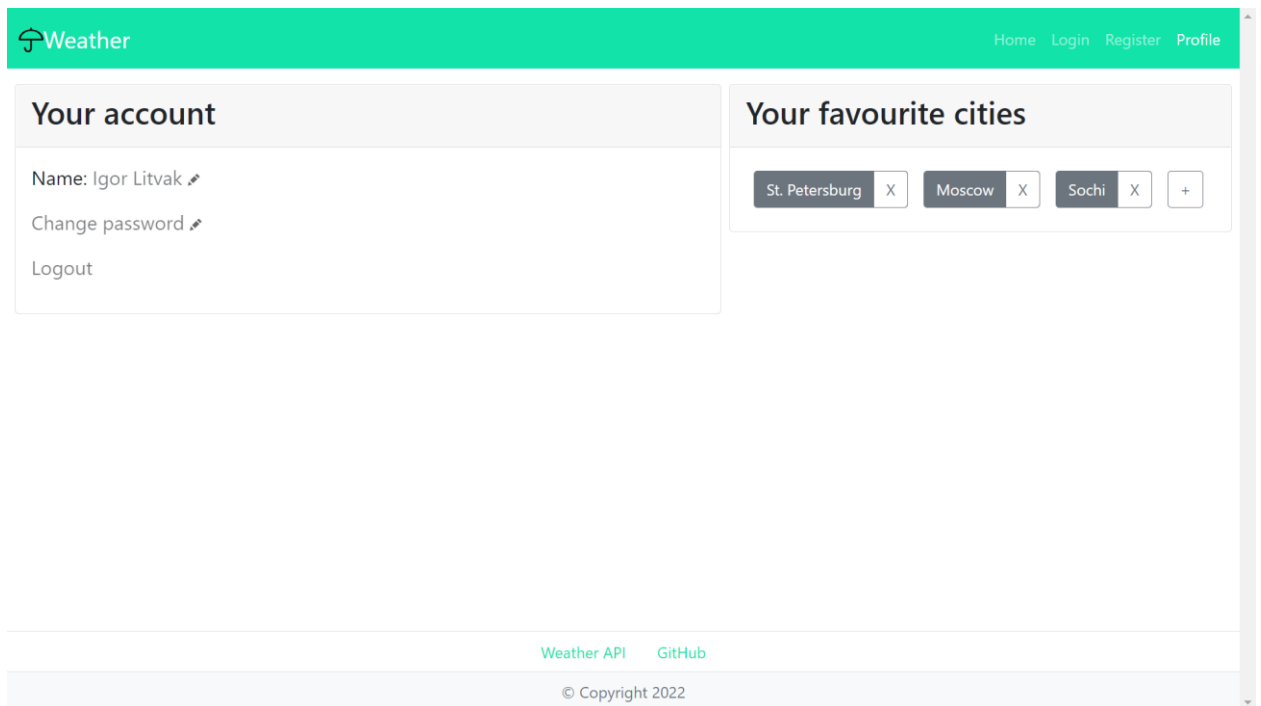
После получения разрешения запрос с координатами пользователя отправляется на бэкенд и возвращается ближайший к данным координатам город.



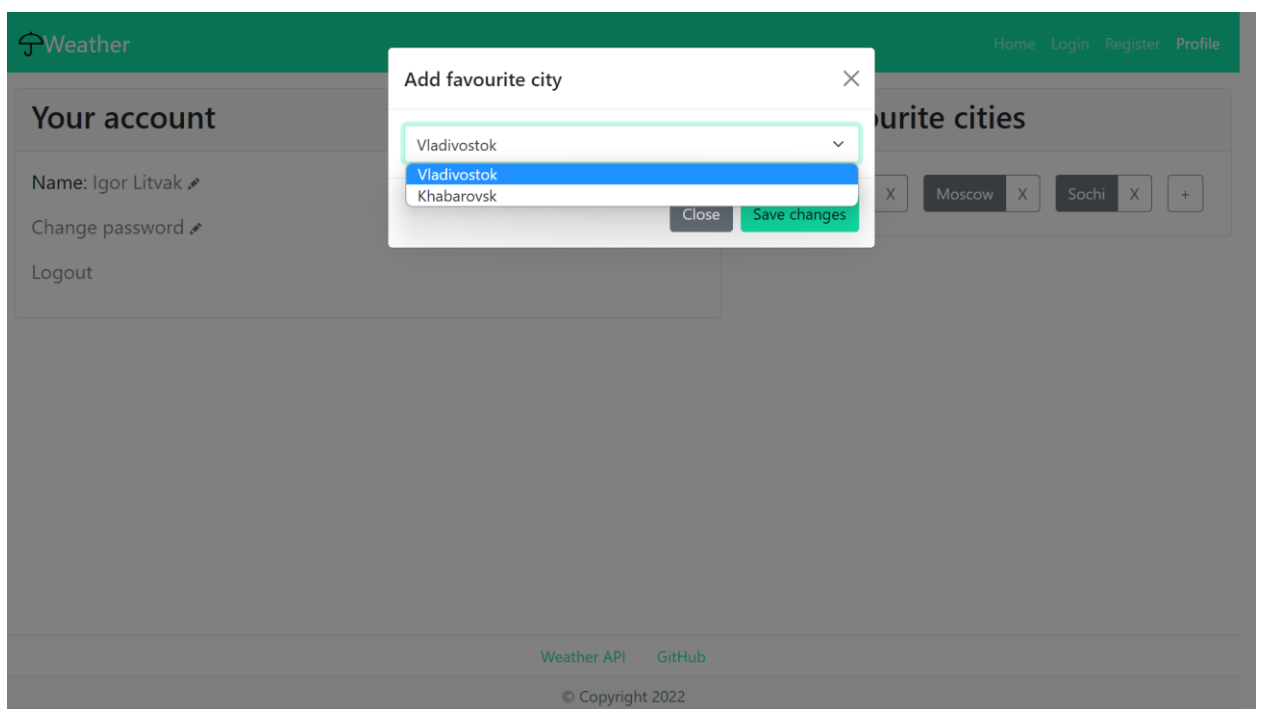
4.3 Добавление любимых городов

В случае, если пользователь хочет просматривать погоду других городов, он может добавить их в любимые города в личном кабинете.

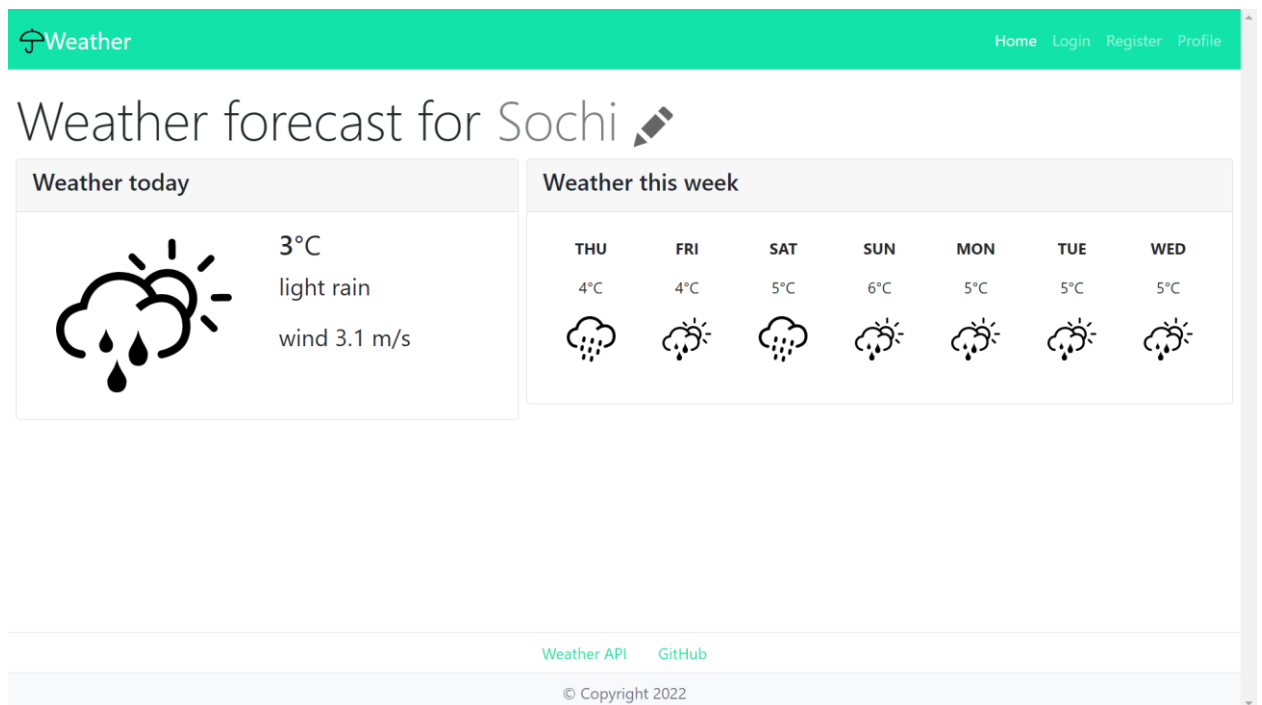
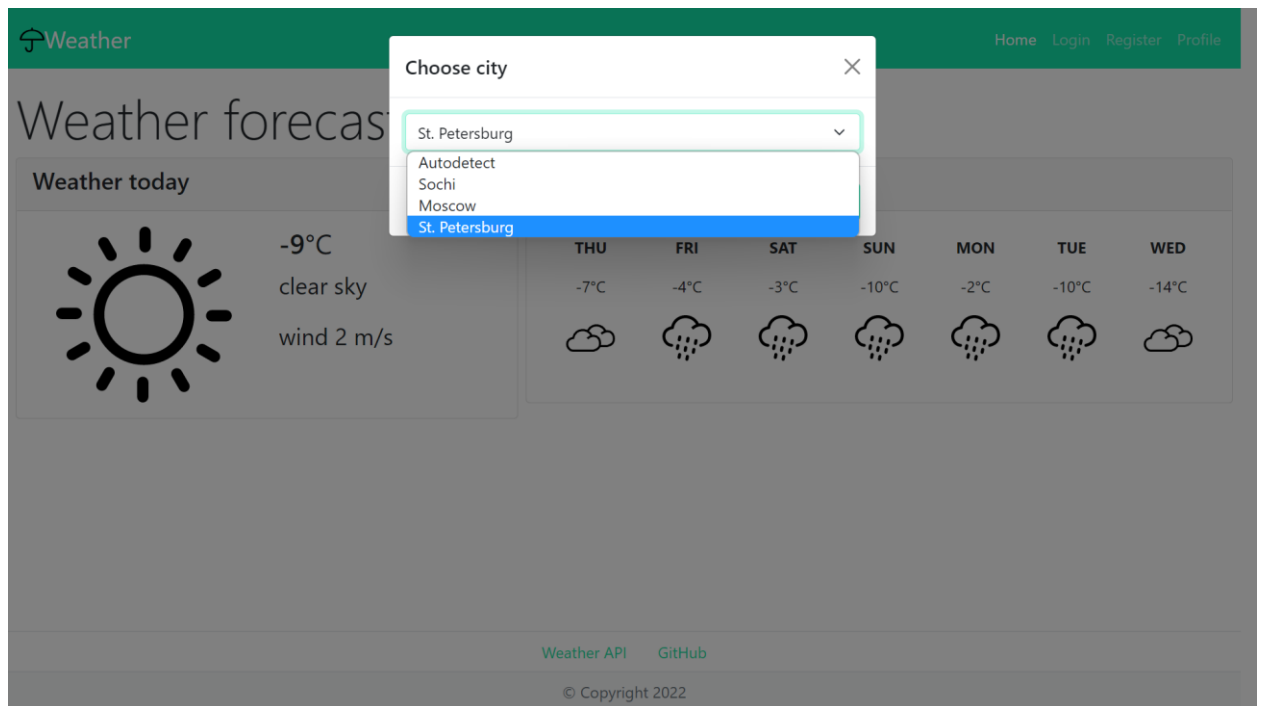




Добавить город в любимые дважды нельзя.

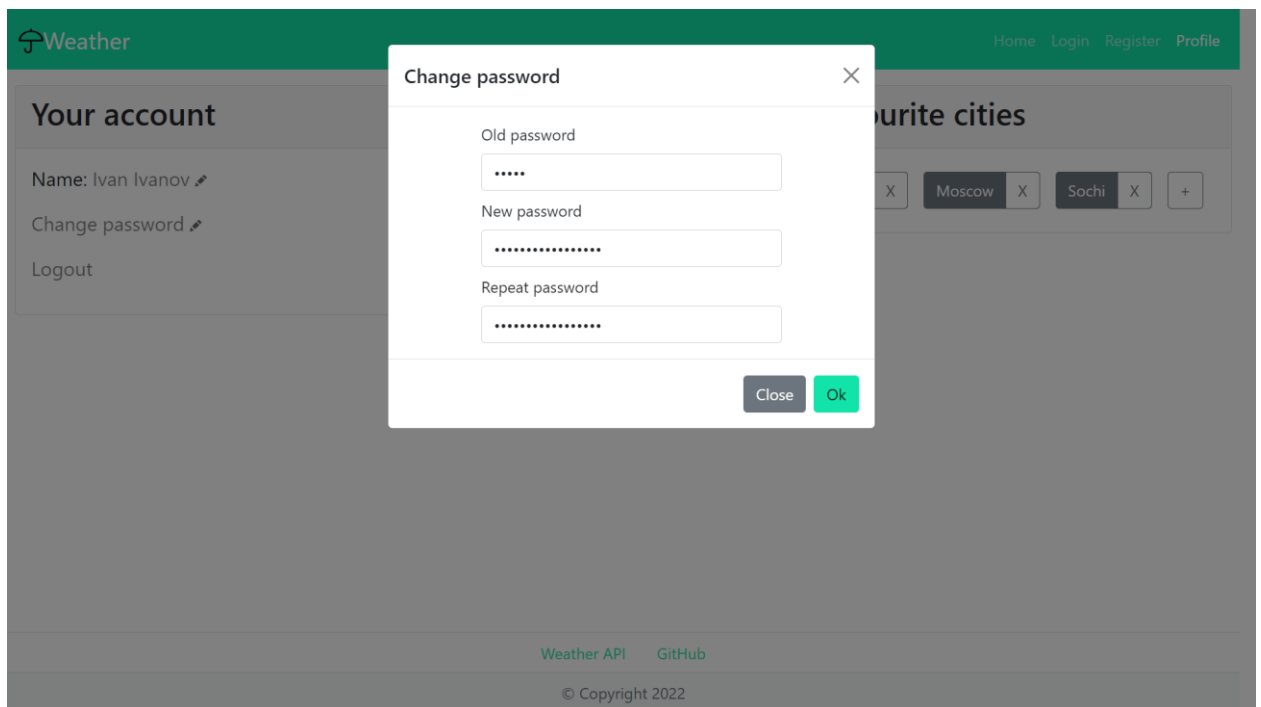
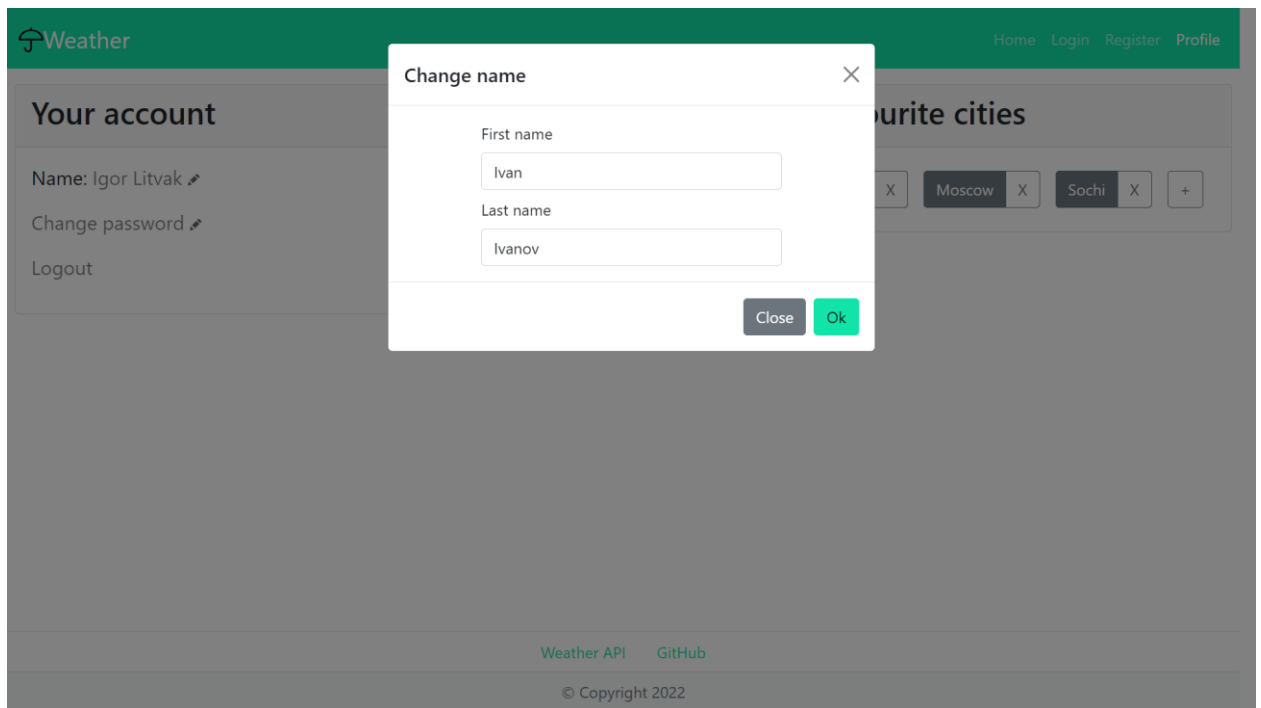


После добавления городов в любимые их можно будет выбрать на главной странице на ряду с автоматическим определением координат.



4.4 Настройки пользователя

На личной странице пользователя пользователь может сменить свое имя и пароль:



Вывод

Выполнив данный проект, я изучил новый для себя стек технологий (Vue, Axios и т. д.), и хорошо понял принципы фулстак-разработки. Реализованное мной приложение потребовало работать с такими вещами как CORS, JWT, AJAX и другими как со стороны бэкенд, так и фронтенд разработки, что является бесценным опытом в веб-разработке.

Список литературы

1. Документация Django [Электронный ресурс] —
<https://docs.djangoproject.com/en/4.0/>
2. Документация Django REST Framework [Электронный ресурс] —
<https://www.django-rest-framework.org/>
3. Документация Vue [Электронный ресурс] —
<https://vuejs.org/v2/guide/>
4. Документация Vue-router [Электронный ресурс] —
<https://router.vuejs.org/guide/>
5. Документация Vuex [Электронный ресурс] —
<https://vuex.vuejs.org/guide/>
6. Документация Axios [Электронный ресурс] —
https://axios-http.com/docs/api_intro
7. Документация Geolocation API [Электронный ресурс] —
https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API