



雾霾探测系统设计

组号:

姓名:

学号:

2024 年 5 月

一 问题描述

雾霾的频繁出现已严重影响人们的出行和健康状况，对此，为了将雾霾的影响降到最低，人们能够在出行前查看雾霾指数并采取相应的措施变得尤为重要。为满足这一需求，我们设计了一款手机端的雾霾应用程序探测系统。

这款雾霾应用程序探测系统旨在提供给用户实时的雾霾信息。系统采用先进的气象传感器和数据分析技术，能够准确地监测和预测雾霾状况。用户可以通过手机端的应用程序随时查看当前所在地区的雾霾指数，并了解雾霾的程度和对健康的影响。

总之，这款手机端雾霾应用程序探测系统通过综合分析多种因素，为用户提供实时的雾霾信息。希望通过这个系统的使用，人们能够更好地了解 and 应对雾霾问题，保护自己和家人的健康。

二 方案设计

总体架构

典型的 C/S 架构，客户端负责发送经纬度数据，服务端负责根据经纬度数据，从 OpenWeather 服务器获取天气信息和空气质量信息。

服务端

服务端负责根据经纬度数据，使用 OpenWeather 的 api，从 OpenWeather 服务器获取天气信息和空气质量信息。具体实现方案使用 go+gin+gorm 实现。可以设置多种体系架构为编译目标，具有良好的兼容性。

客户端

客户端负责定位并接受用户选择后发送经纬度数据，并接收服务端返回的天气信息和空气质量信息并显示。

代码逻辑实现采用 Flutter 框架实现，使用 Flutter 的 geoLocator 库实现定位，Flutter 可生成 Windows,Linux,Android,iOS,Web 等平台的目标软件，具备良好的跨平台能力。同时解决因为手机像素的不同，显示结果的不同。

显示界面元素包括：天气和空气质量指数，空气质量指数历史折线图。其中支持实时刷新显示天气和空气质量指数。

界面设计方案使用 MaterialDesignUI3 设计，首页添加城市列表，点击首页右下角添加城市，第二屏在点击首页城市后显示点击的城市的天气详情。第三屏可以设置默认值以外的服务器 url，便于调试开发。

三 数据获取

地图数据

地图数据素材来自[OpenStreetMap](https://www.openstreetmap.org/)，国内访问可能存在被 GFW 阻拦情况，请注意，具体调用可见下代码

```
1 TileLayer(  
2     urlTemplate: 'https://tile.openstreetmap.org/{z}/{x}/{y}.png',  
3     userAgentPackageName: 'dev.fleaflet.flutter_map.example',  
4 ),
```

定位数据

定位数据使用 Flutter 的 `geolocator` 库实现。见下代码

```
1 FloatingActionButton(  
2   onPressed: () async {  
3     Position position = await Geolocator.getCurrentPosition(  
4       desiredAccuracy: LocationAccuracy.high);  
5     setState(() {  
6       selectedPosition =  
7         LatLng(position.latitude, position.longitude);  
8       selectedMarker = Marker(  
9         point: selectedPosition,  
10        builder: (context) => const Icon(  
11          Icons.location_on,  
12          size: 50,  
13          color: Colors.red,  
14        ),  
15      );  
16    });  
17  },  
18  child: const Icon(Icons.gps_fixed),  
19 ),
```

天气与空气质量数据

客户端

客户端使用 Flutter 框架实现，根据用户选择的位置以及配置的服务器地址，请求获得对应的天气与空气质量信息。具体实施方案见下代码

```
1 Future<dynamic> fetchWeather(double lat, double lon) async {  
2   final prefs = await SharedPreferences.getInstance();  
3   final serverUrl = prefs.getString("serverUrl") ?? "http://localhost:8901";  
4   final url = Uri.parse('$serverUrl/api/v1/weather/by_pos');  
5   final response = await http.post(  
6     url,  
7     headers: {'Content-Type': 'application/json'},  
8     body: json.encode({  
9       'lat': lat.toString(),  
10      'lon': lon.toString(),  
11    }),  
12  );  
13  if (response.statusCode == 200) return json.decode(response.body);  
14 }
```

服务端

服务端使用 go+gin+gorm 实现, 使用 OpenWeather 的 api, 从 OpenWeather 服务器获取天气信息和空气质量信息。具体实现方案见下代码

```
1 func GetWeatherByPos(lat, lon, apiKey string) (*model.Weather, error) {
2     response, err := http.Get(fmt.Sprintf("%s?lat=%s&lon=%s&appid=%s&units=metric",
3         weatherUrl, lat, lon, apiKey))
4
5     if err != nil {
6         return nil, err
7     }
8     defer response.Body.Close()
9     if response.StatusCode != http.StatusOK {
10        return nil, fmt.Errorf("failed to get weather data. Status: %s",
11            response.Status)
12    }
13    weather := new(model.Weather)
14    if err := json.NewDecoder(response.Body).Decode(weather); err != nil {
15        return nil, err
16    }
17    return weather, nil
18 }
```

四 结果展示与分析

服务端初始化

服务端默认使用 debug 模式, 输出对应部署的日志信息。

```
[minami@KotoriNoArch build]$ ./fog-forcast-linux-amd64
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env:   export GIN_MODE=release
- using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /api/v1/city          --> main.main.GetAll[...].func2 (3 handlers)
[GIN-debug] POST   /api/v1/city/by_pos   --> main.main.handleGetCityByPos.func3 (3 handlers)
[GIN-debug] POST   /api/v1/weather/by_city --> main.main.handleGetWeather.func4 (3 handlers)
[GIN-debug] POST   /api/v1/weather/by_pos --> main.main.handleGetWeather.func5 (3 handlers)
[GIN-debug] POST   /api/v1/aqi/by_pos    --> main.main.handleGetAQIByPos.func6 (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8901
```

服务端处理请求

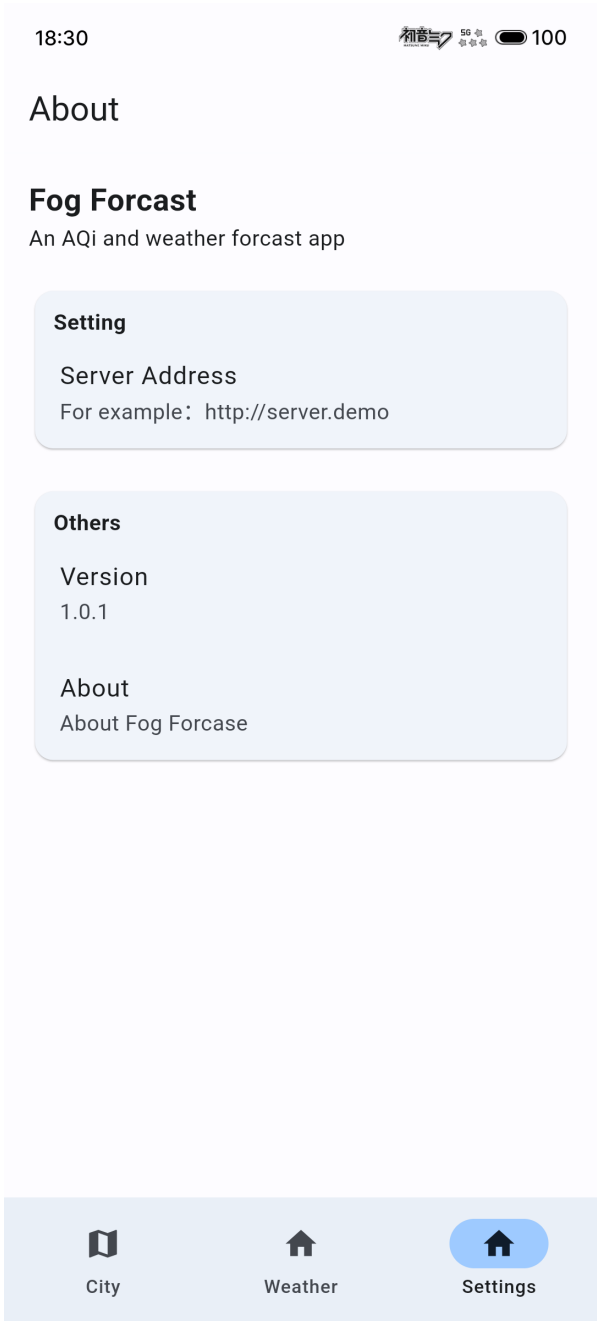
当服务端获取到客户端对应的请求时, 会将对应请求的 api, respond 的状态信息等打印

```
[GIN] 2024/05/05 - 01:47:17 | 200 | 1.005775685s | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 01:47:18 | 200 | 1.009187097s | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
[GIN] 2024/05/05 - 01:47:34 | 200 | 631.72228ms | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 01:47:35 | 200 | 746.422393ms | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
[GIN] 2024/05/05 - 01:48:55 | 200 | 509.867882ms | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 01:48:56 | 200 | 995.104125ms | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
[GIN] 2024/05/05 - 02:19:07 | 200 | 1.212266689s | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 02:19:07 | 200 | 806.164335ms | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 02:19:07 | 200 | 1.133691531s | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 02:19:08 | 200 | 978.61945ms | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
[GIN] 2024/05/05 - 02:19:09 | 200 | 1.05608979s | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
[GIN] 2024/05/05 - 02:19:09 | 200 | 1.163124944s | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
[GIN] 2024/05/05 - 03:05:36 | 200 | 930.543121ms | 10.199.26.188 | POST | "/api/v1/weather/by_pos"
[GIN] 2024/05/05 - 03:05:37 | 200 | 1.025164578s | 10.199.26.188 | POST | "/api/v1/aqi/by_pos"
```

客户端界面



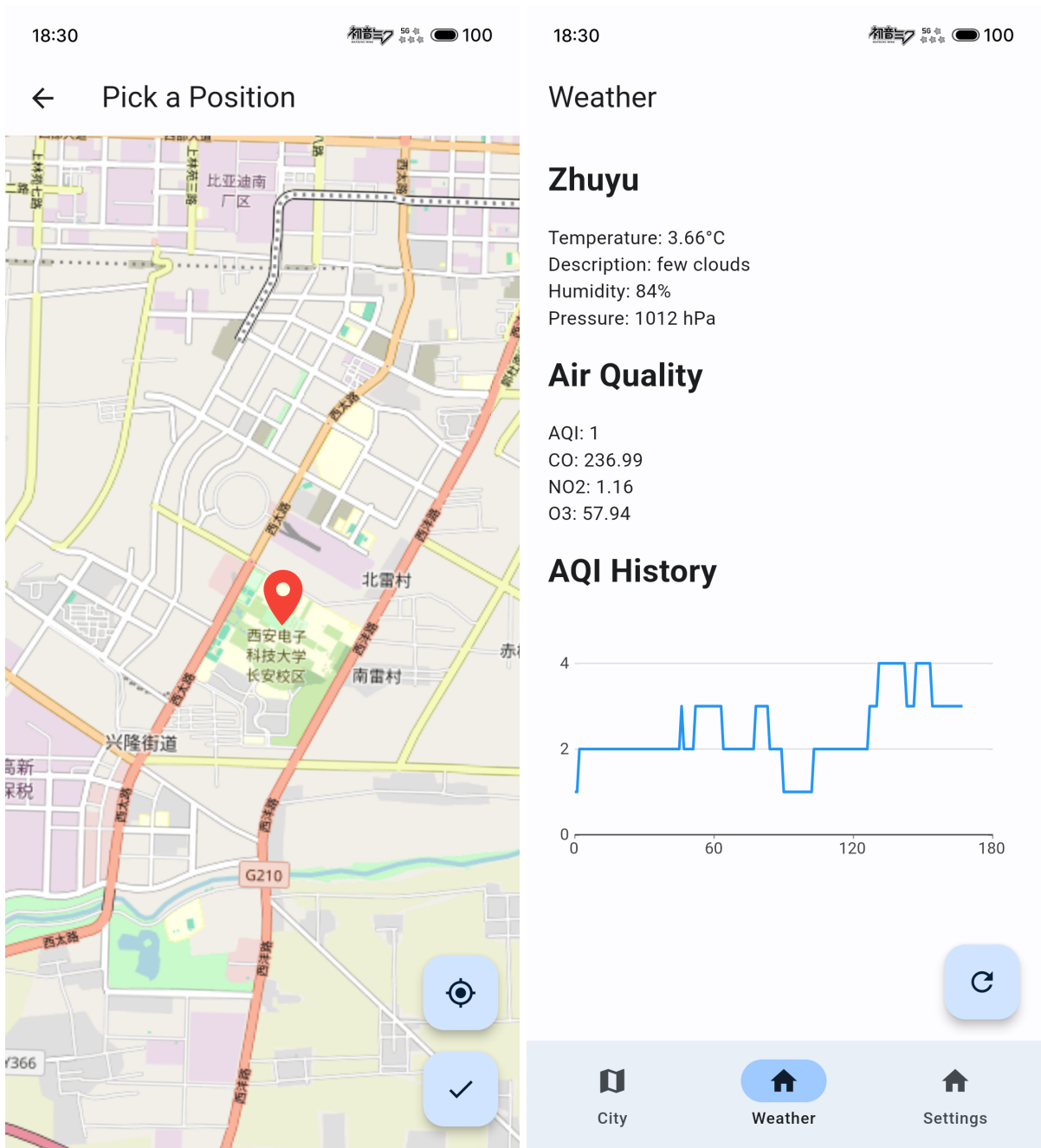
(a) 城市界面



(b) 设置界面

操作流程

初始情况下，城市界面为空，在点击加号添加城市后，会进入地图，且初始位置为用户所在位置。由于设备，网络等各种问题，精确度目前测试到市级别。在确定地点后，点击确定向 cities 页面添加成功城市，点击对应城市可进入城市天气详情界面。



(a) 选择地点

(b) 查看天气详情

五 小组成员分工

| 成员 | 分工 |
|------|---------|
| 神里绫华 | 前端 |
| 那维莱特 | 后端 |
| 芙宁娜 | 测试及报告编写 |

表 1: 成员分工

六 心得与体会

困了，gpt，建议根据赛博分工一人一个 gpt