2025 年夏季《移动软件开发》实验报告

姓名和学号？	聂宇航，23170001072
本实验属于哪门课程？	中国海洋大学 25 夏《移动软件开发》
实验名称？	实验 2：天气查询小程序
博客地址？	《移动软件开发》实验二实验报告-CSDN 博客
Github 仓库地址？	这个是《移动软件开发》这门课的实验代码与报告

（备注：将实验报告发布在博客、代码公开至 github 是 加分项，不是必须做的）

一、实验目标

1、学习使用快速启动模板创建小程序的方法；2、学习不使用模板手动创建小程序的方法。

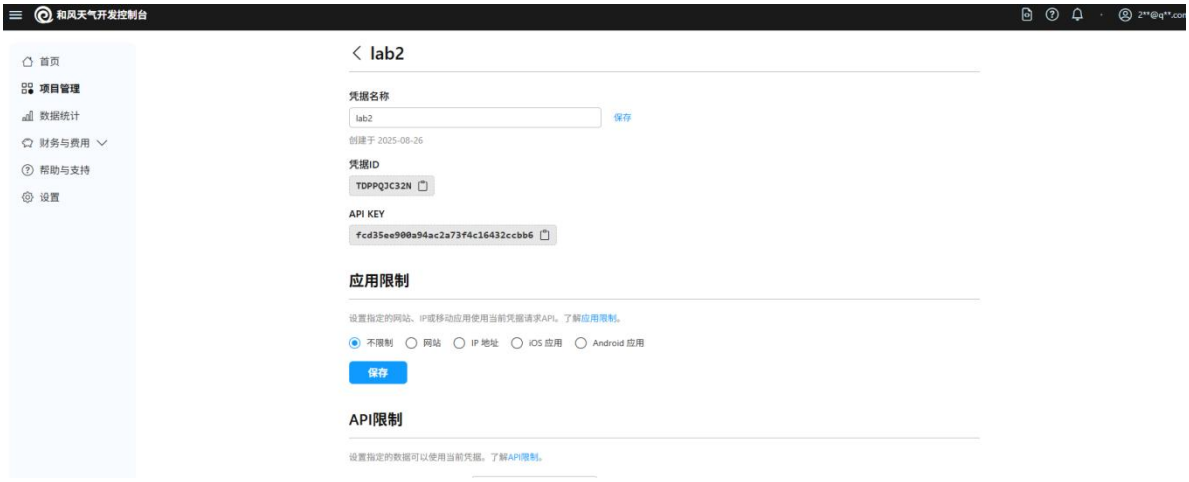
二、实验步骤

1.准备工作：

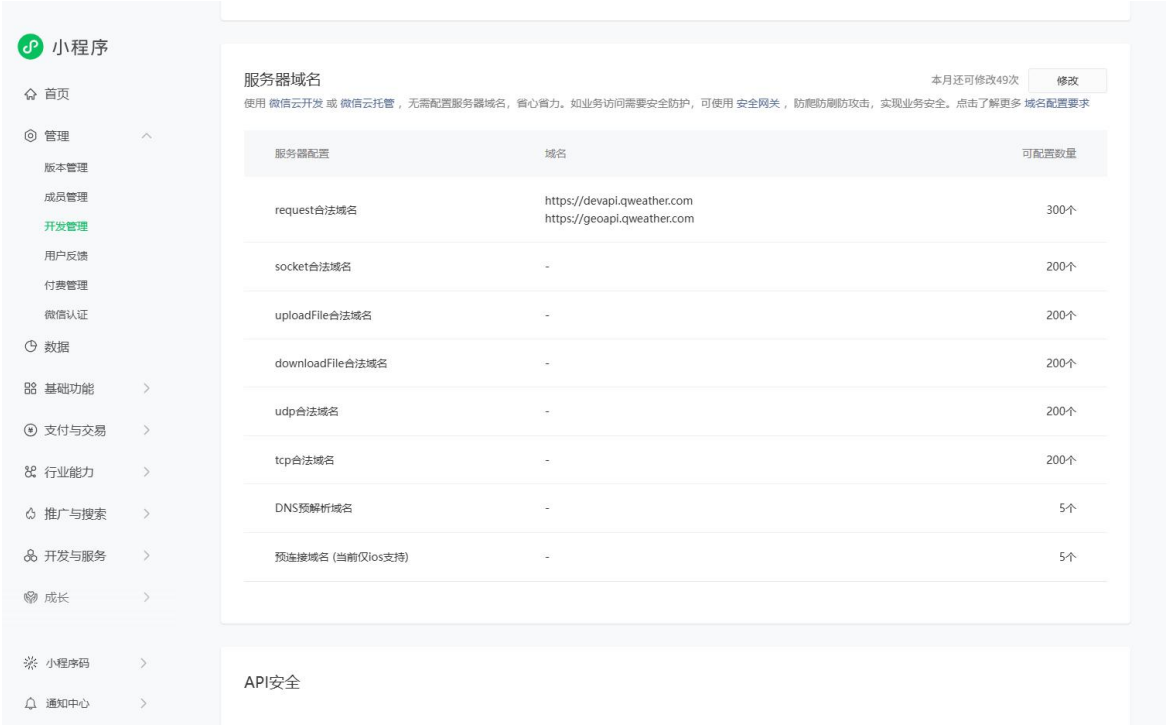
因为我们这个实验是根据和风天气 API 来获取某地的天气情况。

然后去和风天气官网去申请 API 密钥，首先使用邮箱注册并激活账户，接着我们需要查看个人的 key，这在后面的实验中会用到。

在注册过程中我们选择免费用户，然后填写相关的名称，再在如下项目管理中生产一个项目（选择 API KEY 生成项目），那么就会自动生成一个 key。



微信小程序的 wx.request 只能访问 微信公众平台后台 → 开发 → 开发管理 → 开发设置 → 服务器域名 里配置的 request 合法域名。



项目一开始会有默认页面，和实验一一样将默认页面操作

1.创建页面文件

项目创建完毕后,在根目录中会生成文件夹 pages 用于存放页面文件。一般来说首页默认命名为 index,表示小程序运行的第一个页面;其他页面名称可本次只需要保留首页(index)即可。

具体操作如下:

(1)将 app. json 文件内 pages 属性中的"pages/logs/logs"删除,并删除上一行末尾的逗号。

(2)按快捷键 Ctrl+S 保存当前修改。

2.删除和修改文件

具体操作如下:

(1)删除 utils 文件夹及其内部所有内容。

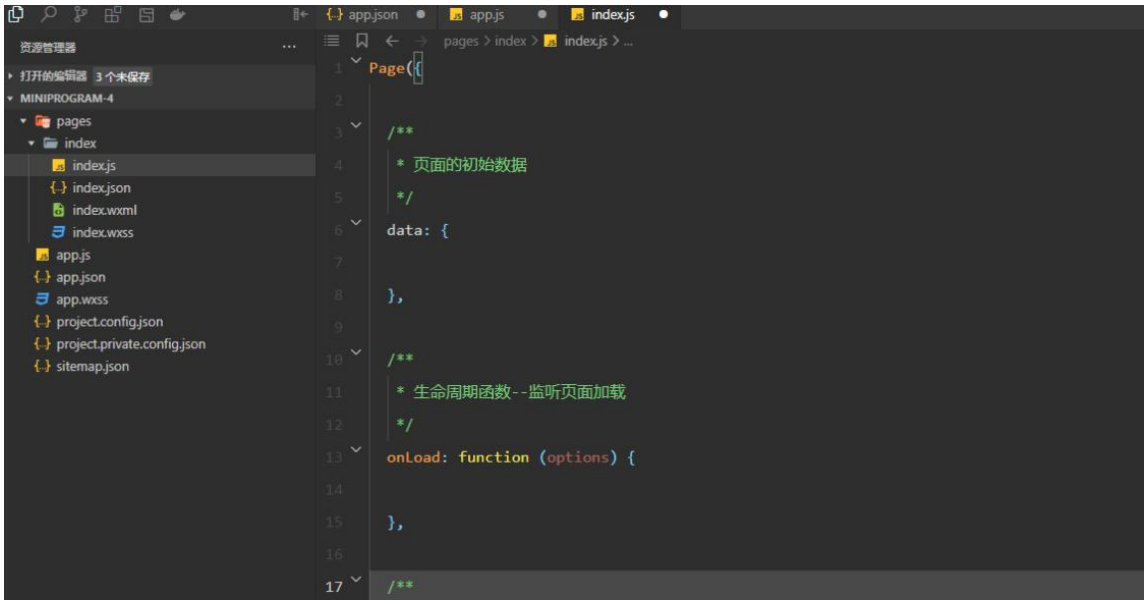
(2)删除 pages 文件夹下的 logs 目录及其内部所有内容。

(3)删除 index. wxml 和 index. wxss 中的全部代码。

(4)删除 index.js 中的全部代码,并且输人关键词“page”找到第二个选项按回车键让其自动补全函数。

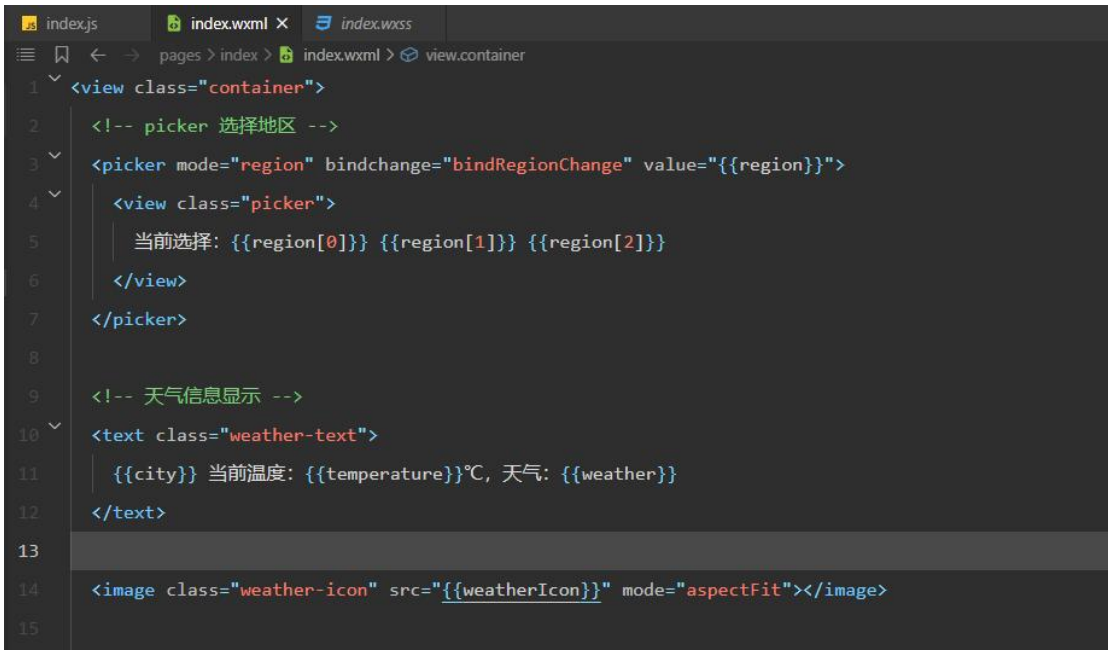
(5)删除 app.wxss 中的全部代码。

(6)删除 app.js 中的全部代码,并且输入关键词“app”找到第二个选项按回车键让其自动补全兩数。



2.页面设计：

wxml 文件：


index.js
index.wxml
index.wxss
pages > index > index.wxml > view.container
<view class="container">
<!-- picker 选择地区 -->
<picker mode="region" bindchange="bindRegionChange" value="{{region}}">
<view class="picker">
当前选择: {{region[0]}} {{region[1]}} {{region[2]}}
</view>
</picker>
<!-- 天气信息显示 -->
<text class="weather-text">
{{city}} 当前温度: {{temperature}}℃, 天气: {{weather}}
</text>
<image class="weather-icon" src="{{weatherIcon}}" mode="aspectFit"></image>

```
<image class="weather-icon" src="{{weatherIcon}}" mode="aspectFit"></image>

<view class="detail">
  <view class="bar">
    <view class="box">湿度: {{humidity}}%</view>
    <view class="box">气压: {{pressure}} hPa</view>
    <view class="box">风向: {{windDirection}}</view>
  </view>
  <view class="bar">
    <view class="box">能见度: {{visibility}} km</view>
    <view class="box">风速: {{windSpeed}} m/s</view>

  </view>
  <view class="bar">
    <view class="box">空气质量: {{category}}</view>
    <view class="box">紫外线: {{uvIndex}}</view>
  </view>
  <view class="bar">
    <view class="box">日出时间: {{sunrise}}</view>
  </view>
  <view class="bar">
    <view class="box">日落时间: {{sunset}}</view>
  </view>
</view>
</view>
```

1. 最外层容器

```
<view class="container">
  ...
</view>
```

`class="container"` 表示这个容器有样式，在对应的 WXSS 文件中定义布局和样式。

2. 地区选择器（Picker）

```
<picker mode="region" bindchange="bindRegionChange" value="{{region}}">
  <view class="picker">
    当前选择：{{region[0]}} {{region[1]}} {{region[2]}}
```

```
  </view>
</picker>
```

`<picker>` 是选择控件，这里使用 `mode="region"`，用于选择 省/市/区。

`bindchange="bindRegionChange"` 表示选择改变时，会触发页面对应的 `bindRegionChange` 方法。

`value="{{region}}"` 使用了 数据绑定，`region` 是页面的一个数组变量，存储选择的省、市、区。

内部 `<view>` 用来显示当前选中的地区：

```
当前选择：{{region[0]}} {{region[1]}} {{region[2]}}
```

`region[0]`：省

`region[1]`：市

`region[2]`：区

3. 天气信息显示

```
<text class="weather-text">
  {{city}} 当前温度：{{temperature}}℃，天气：{{weather}}
</text>

<image class="weather-icon" src="{{weatherIcon}}" mode="aspectFit"></image>
```

`<text>` 用来显示文字信息。

`{{city}}`：城市名称

`{{temperature}}`：当前温度

`{{weather}}`：天气情况（如晴、雨等）

`<image>` 显示天气图标。

`src="{{weatherIcon}}"`：图标 URL 绑定到页面数据 `weatherIcon`。

`mode="aspectFit"`：图像按比例缩放，整个图像都显示在容器内。

4. 天气详情部分

```
<view class="detail">
  <view class="bar">
```

```
    <view class="box">湿度：{{humidity}}%</view>
    <view class="box">气压：{{pressure}} hPa</view>
    <view class="box">风向：{{windDirection}}</view>
  </view>
  ...
</view>
```

外层 `<view class="detail">` 是详情容器。

内部用多个 `<view class="bar">` 表示每一行的天气信息。

每行有若干 `<view class="box">` 显示具体指标：

湿度 `{{humidity}}%`

气压 `{{pressure}} hPa`

风向 `{{windDirection}}`

能见度 `{{visibility}} km`

风速 `{{windSpeed}} m/s`

空气质量 `{{category}}`

紫外线指数 `{{uvIndex}}`

日出 `{{sunrise}}`

日落 `{{sunset}}`

为了更好的显示，这里用 bar + box 的组合做网格布局，每行显示 2-3 个数据，日出日落单独占一行。

wxss 文件：


index.js
index.wxml
index.wxss
pages > index > index.wxss > ...
.container {
display: flex;
flex-direction: column;
padding: 20rpx;
}
.picker {
margin: 20rpx 0;
padding: 15rpx;
background-color: #f5f5f5;
border-radius: 10rpx;
}
.weather-text {
font-size: 32rpx;
margin: 20rpx 0;
text-align: center;
}
.weather-icon {
width: 150rpx;
height: 150rpx;
margin: 20rpx auto;
}

```
.detail {
  margin-top: 30rpx;
}

.bar {
  display: flex;
  justify-content: space-between;
  margin-bottom: 20rpx;
}

.box {
  flex: 1;
  margin: 0 10rpx;
  padding: 20rpx;
  text-align: center;
  background-color: #e6f7ff;
  border-radius: 12rpx;
  font-size: 28rpx;
}
```

1. .container

```
.container {
  display: flex;
  flex-direction: column;
  padding: 20rpx;
}
```

外层容器使用 Flex 布局，方向为 纵向排列（column）。

`padding: 20rpx`：整个容器内部留 20rpx 的空白。这保证了页面的所有内容从上到下垂直排列，并且不会紧贴屏幕边缘。

2. .picker

```
.picker {
    margin: 20rpx 0;
    padding: 15rpx;
    background-color: #f5f5f5;
    border-radius: 10rpx;
}
```

`margin: 20rpx 0`：上下各 20rpx 的间距。

`padding: 15rpx`：内部内容（文本）留白。

`background-color: #f5f5f5`：浅灰色背景。

`border-radius: 10rpx`：圆角矩形，使界面更柔和。

3. .weather-text

```
.weather-text {
    font-size: 32rpx;
    margin: 20rpx 0;
    text-align: center;
}
```

文本字体 32rpx。

上下各 20rpx 的间距。

`text-align: center`：文字居中显示。

4..weather-icon

```
.weather-icon {
    width: 150rpx;
    height: 150rpx;
    margin: 20rpx auto;
}
```

固定宽高 150rpx。

`margin: 20rpx auto`：上下 20rpx 间距，左右居中。

5. .detail

```
.detail {
  margin-top: 30rpx;
}
```

天气详情整体与上方元素间隔 30rpx。

6. .bar

```
.bar {
  display: flex;
  justify-content: space-between;
  margin-bottom: 20rpx;
}
```

每一行使用 Flex 水平布局。

`justify-content: space-between`：每个 `.box` 在行内均匀分布，左右间距自动拉开。

`margin-bottom: 20rpx`：每行之间的垂直间距。

7. .box

```
.box {
  flex: 1;
  margin: 0 10rpx;
  padding: 20rpx;
  text-align: center;
  background-color: #e6f7ff;
  border-radius: 12rpx;
  font-size: 28rpx;
}
```

`flex: 1`：每个 box 在当前行平分可用空间。

`margin: 0 10rpx`：左右间距 10rpx。

`padding: 20rpx`：内部文字与边框留白。

`text-align: center`：文字居中。

`background-color: #e6f7ff`：淡蓝色背景。

`border-radius: 12rpx`：圆角矩形。

`font-size: 28rpx`：稍小于天气概览字体。

最后得到的页面为：

3.逻辑实现

1. 页面初始化数据（`data`）

`data` 是页面的 状态数据，用于绑定到 WXML。

初始值用占位符 `--`，页面刚加载时显示默认值。

包含：

地区信息：`region`（省市区数组）、`city`（市名）

天气概览：`temperature`、`weather`、`weatherIcon`

气象指标：湿度、气压、风向、能见度、风速

日出日落：`sunrise`、`sunset`

空气质量：`aqi`、`category`

生活指数：`uvIndex`

2. 选择地区事件（`bindRegionChange`）和 按钮选择城市（`chooseCity`）

当 `<picker>` 的值改变时触发。

获取选择的省市区数组 `region`。

取市名 `region[1]` 作为 `city`。

更新页面数据并调用 `getWeather(city)` 获取天气。

使用微信内置选择城市控件 `wx.chooseCity`。

成功后更新 `city` 和 `region`，并获取天气。

3. 页面加载（`onLoad`）

```
onLoad() {
  this.getWeather(this.data.city)
},
```

页面初次加载时，调用 `getWeather` 获取默认城市天气。

5. 获取天气信息（`getWeather`）

index.js ● index.wxml {..} index.json

pages > index > index.js > getNowWeather

```
getWeather(cityName) {
  const key = "58cde137c76f44f5bc7885fc1e711aa9"
  wx.request({
    url: "https://geoapi.qweather.com/v2/city/lookup",
    data: {
      location: cityName,
      key: key
    },
    success: (res) => {
      if (res.data.code === "200" && res.data.location.length > 0) {
        // 找到第一个结果
        const cnCity = res.data.location[0]
        const cityId = cnCity.id
        console.log("城市ID:", cityId, cnCity.name)

        this.getNowWeather(cityId, key)
        this.getSunInfo(cityId, key)
        this.getAirQuality(cityId, key)
        this.getLifeIndex(cityId, key)
      } else {
        wx.showToast({ title: '未找到城市', icon: 'none' })
      }
    },
    fail: (err) => {
      console.error("城市查询失败:", err)
    }
  })
},
```

通过城市名查询城市 ID，然后获取详细天气信息。

调用了四个函数：

`getNowWeather`：实时天气

`getSunInfo`：日出日落

`getAirQuality`：空气质量

`getLifeIndex`：紫外线指数

使用和风天气 API (`qweather.com`)。

错误处理：未找到城市时显示 `Toast`。

5.1 实时天气（`getNowWeather`）

```
// 实时天气
getNowWeather(cityId, key) {
  wx.request({
    url: "https://devapi.qweather.com/v7/weather/now",
    data: { location: cityId, key: key },
    success: (res) => {
      if (res.data.code === "200") {
        let now = res.data.now
        this.setData({
          temperature: now.temp,
          weather: now.text,
          weatherIcon: `/images/${now.icon}.svg`,
          humidity: now.humidity,
          pressure: now.pressure,
          windDirection: now.windDir,
          windSpeed: now.windSpeed,
          visibility: now.vis
        })
      }
    }
  })
},
```

获取当前温度、天气描述、图标、湿度、气压、风向、风速、能见度。

更新 `data`，页面绑定自动刷新。

5.2 日出日落（`getSunInfo`）

```
// 日出日落
getSunInfo(cityId, key) {
  const date = new Date()
  const yyyy = date.getFullYear()
  const mm = String(date.getMonth() + 1).padStart(2, '0')
  const dd = String(date.getDate()).padStart(2, '0')
  const today = `${yyyy}${mm}${dd}`

  wx.request({
    url: "https://devapi.qweather.com/v7/astronomy/sun",
    data: { location: cityId, date: today, key: key },
    success: (res) => {
      if (res.data.code === "200") {
        this.setData({
          sunrise: res.data.sunrise,
          sunset: res.data.sunset
        })
      }
    }
  })
},
```

获取今日日出 `sunrise` 和日落 `sunset`。

使用 API `/v7/astronomy/sun`。

5.3 空气质量（`getAirQuality`）

```
// 空气质量
getAirQuality(cityId, key) {
  wx.request({
    url: "https://devapi.qweather.com/v7/air/now",
    data: { location: cityId, key: key },
    success: (res) => {
      if (res.data.code === "200") {
        let air = res.data.now
        this.setData({
          aqi: air.aqi,
          category: air.category,
        })
      }else {
        this.setData({
          category: '良好'
        })
      }
    }
  })
},
```

获取 AQI 和空气质量等级 `category`。

如果 API 返回失败，则默认 `category` 为“良好”。

5.4 生活指数（紫外线）（`getLifeIndex`）

```
// 生活指数（紫外线）
getLifeIndex(cityId, key) {
  wx.request({
    url: "https://devapi.qweather.com/v7/indices/1d",
    data: {
      location: cityId,
      type: 5,   // type=5 表示紫外线指数
      key: key
    },
    success: (res) => {
      if (res.data.code === "200" && res.data.daily.length > 0) {
        let uv = res.data.daily[0]
        this.setData({
          uvIndex: uv.category
        })
      }
    }
  })
}
})
```

获取紫外线指数（`type=5` 表示 UV 指数）。

更新 `uvIndex`。

三、程序运行结果

列出程序的最终运行结果及截图。

在一进入的时候，调用 onLoad 函数，对默认城市进行查询（青岛市）

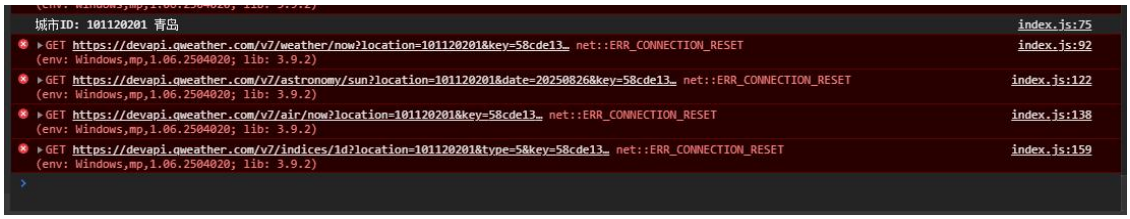
然后我们可以点击最上方的 【当前选择：山东省 青岛市 黄岛区】来改变城市，比如我现在可以选【新疆维吾尔自治区 乌鲁木齐市 天山区】：

然后我们就可以看到乌鲁木齐的天气了：

四、问题总结与体会

描述实验过程中所遇到的问题，以及是如何解决的。有哪些收获和体会，对于课程的安排有哪些建议。

永无止境的网络错误



因为报警为：

`request:fail url not in domain list`

这是 微信小程序网络请求的域名白名单问题。 微信小程序要求所有 `wx.request` 请求的域名必须先配置到 小程序后台 → 开发管理 → 开发设置 → 服务器域名。

解决办法：

登录 微信公众平台

找到 开发 → 开发管理 → 开发设置 → 服务器域名

在 request 合法域名 里添加以下域名（和风天气用到的）：

```
https://geoapi.qweather.com
https://devapi.qweather.com
```

但还是失败了，纯自觉感觉是我的 key 的问题，我把同学的 key 拿来用，发现我的失败了，但我的同学的 key 成功了，非常的诡异啊，从这之后我就用我同学的 key 了