2025年夏季《移动软件开发》实验报告

姓名：潘奕霖 学号：23020007094

姓名和学号?	潘奕霖，23020007094
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?	中国海洋大学2025年夏季《移动软件开发》实验报告——实验2-CSDN博客
Github仓库地址?	https://github.com/PYL1024/mobile-software-development

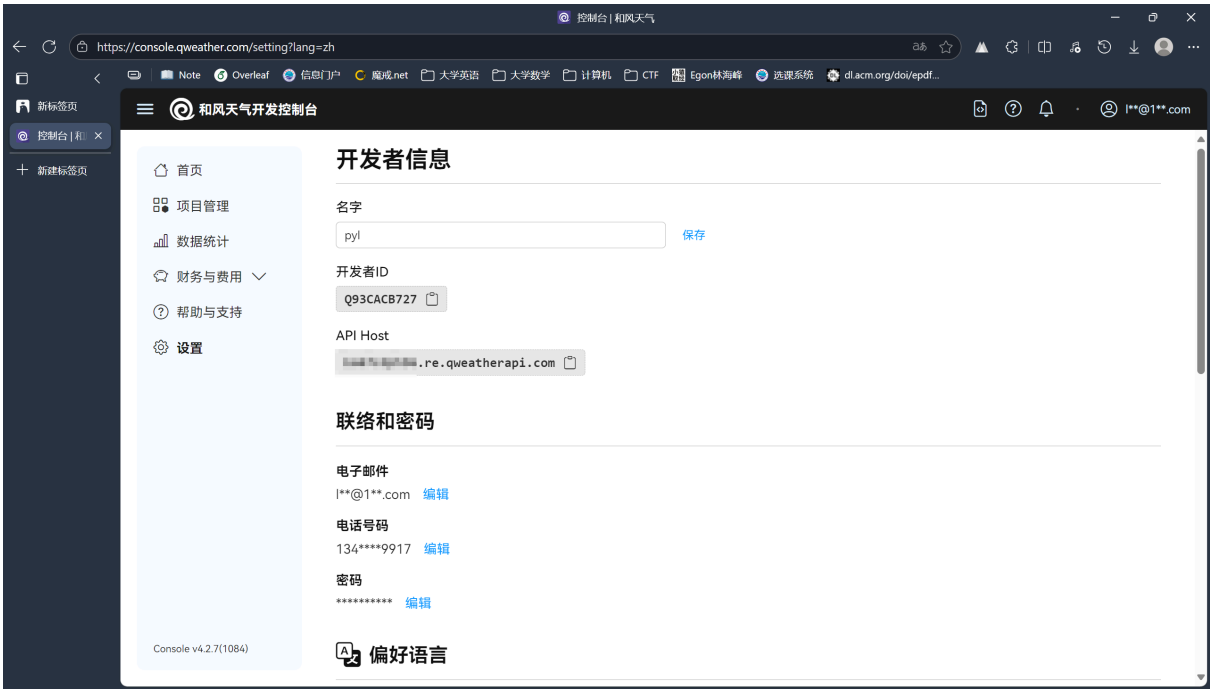
（备注：将实验报告发布在博客、代码公开至 github 是 **加分项**，不是必须做的）

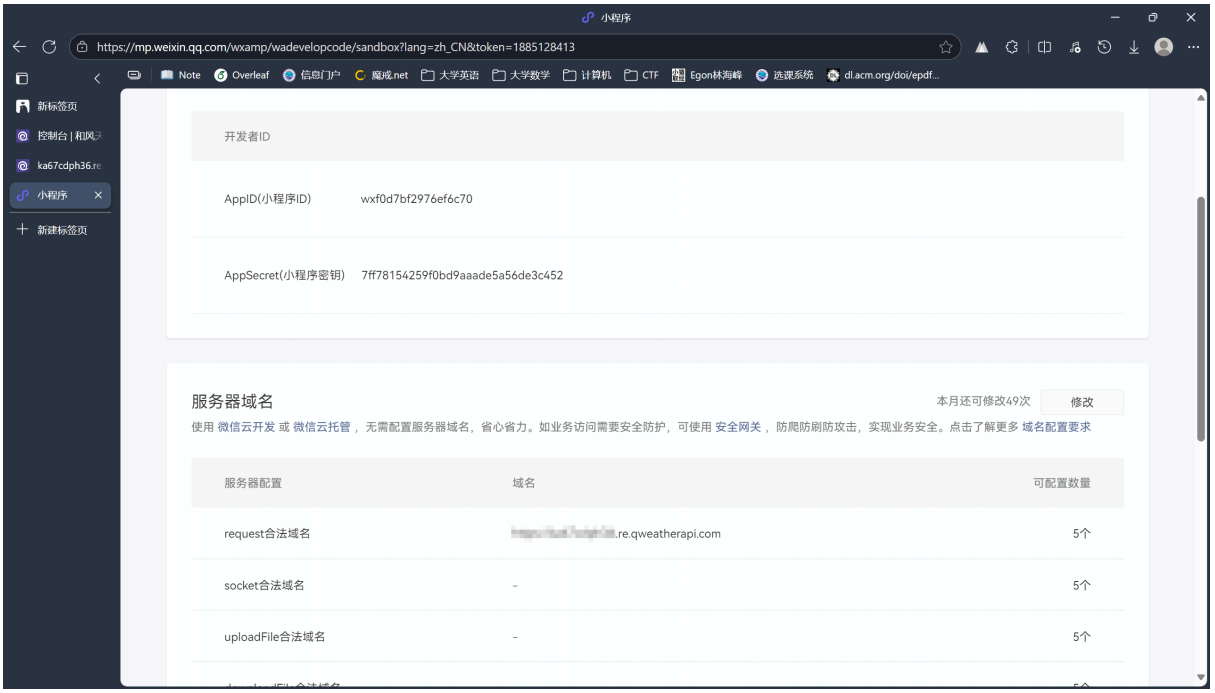
一、实验目标

1、构建一个天气查询小程序。

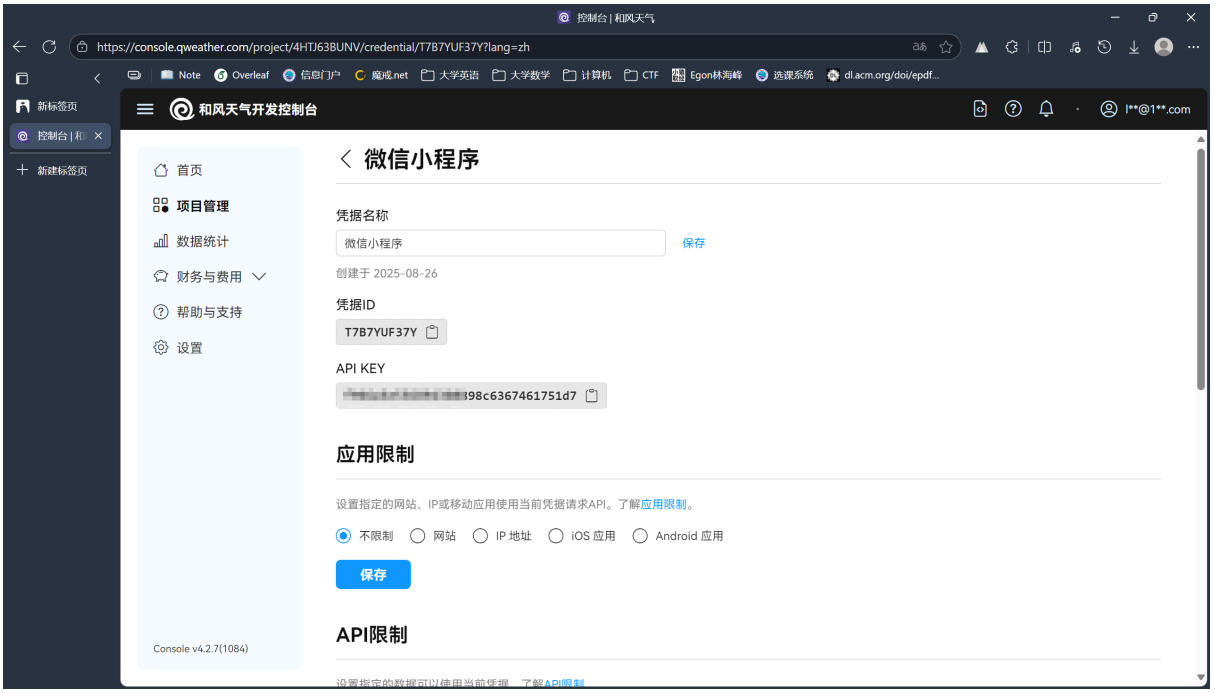
二、实验步骤

1、注册和风天气账户，获取查询所需的凭据。使用和风天气 API 进行查询需要提供 API Host 和项目凭据。API Host 位于开发控制台的设置栏，类似于 `abc1234xyz.def.qweatherapi.com` 。为了让小程序能访问该网站，需要在小程序后台配置服务器域名。

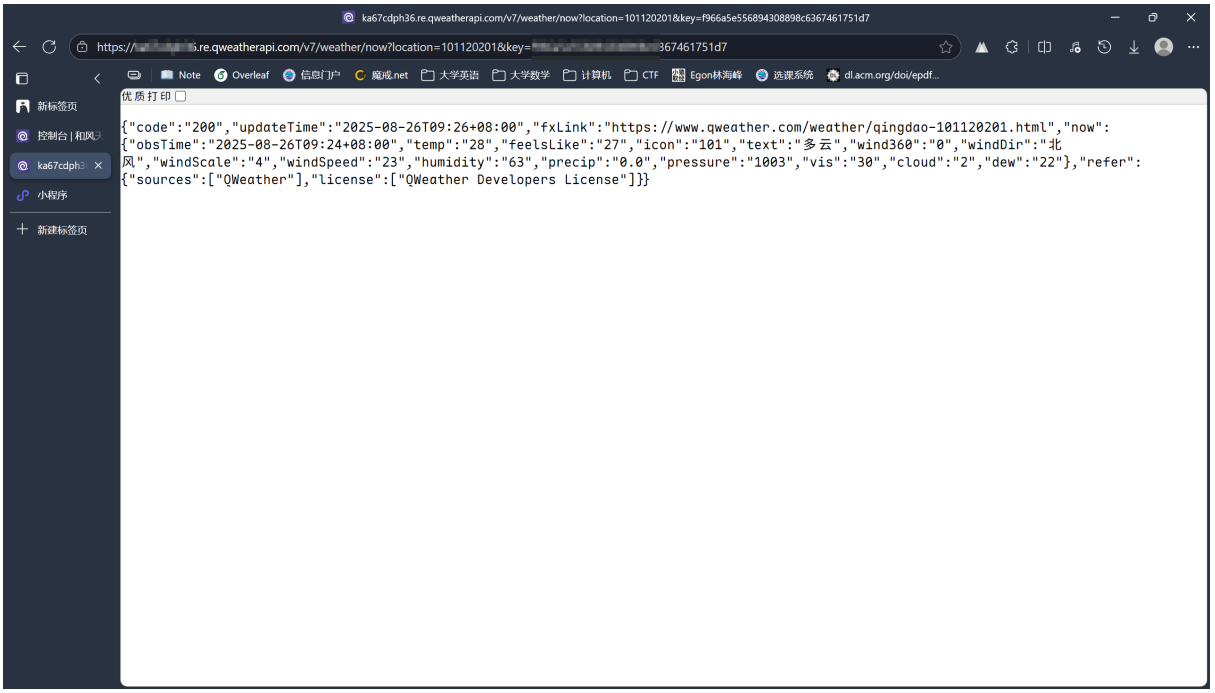




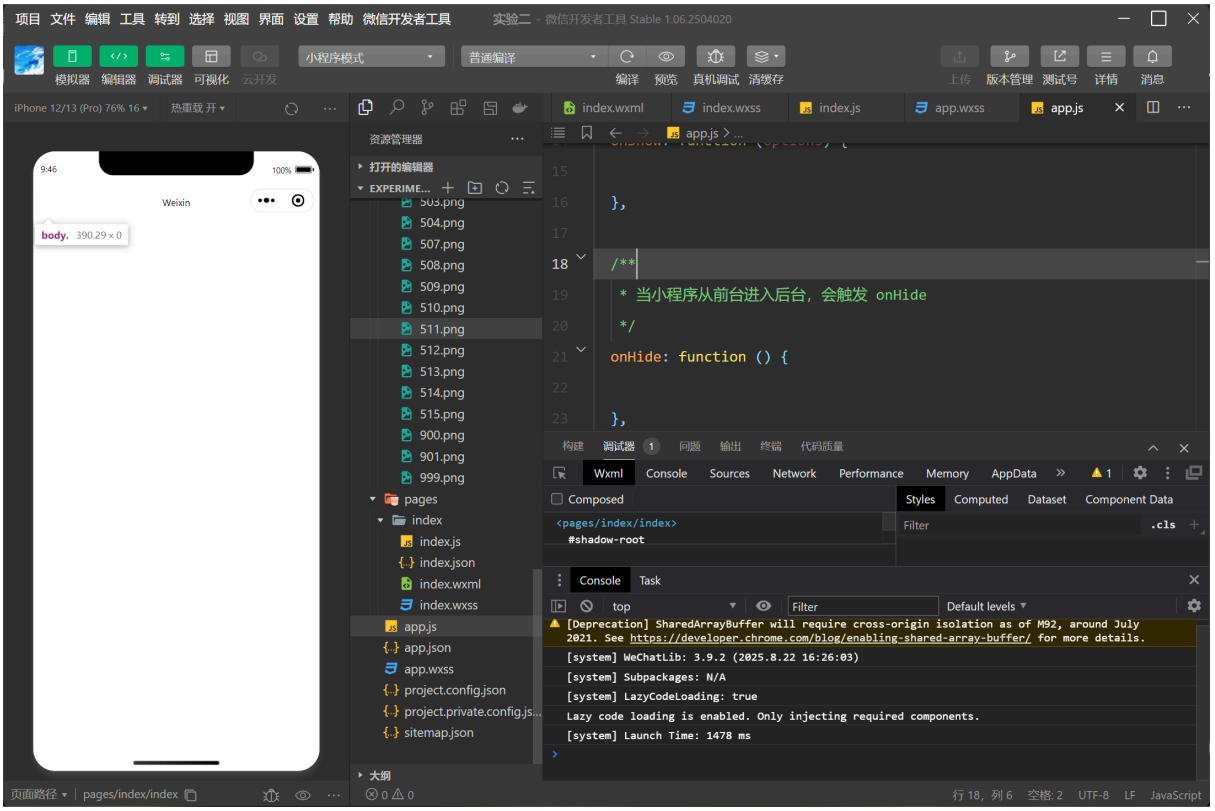
要获取项目凭据，需要创建一个项目，并为此创建凭据，身份认证方式选择API KEY，使用更加简单。



取得凭据后，可以按照和风天气的文档尝试构建请求URL，例如查询北京的实时天气 `https://abcxyz.qweatherapi.com/v7/weather/now?location=101010100&key=1234abdc'`，尝试查询青岛的实时天气。



2、创建项目并下载图片。按照实验文档指示删除和修改项目文件，并将天气图标添加至项目文件。



3、页面整体设计。先设计导航栏，修改 app.json 自定义导航栏内容和颜色。

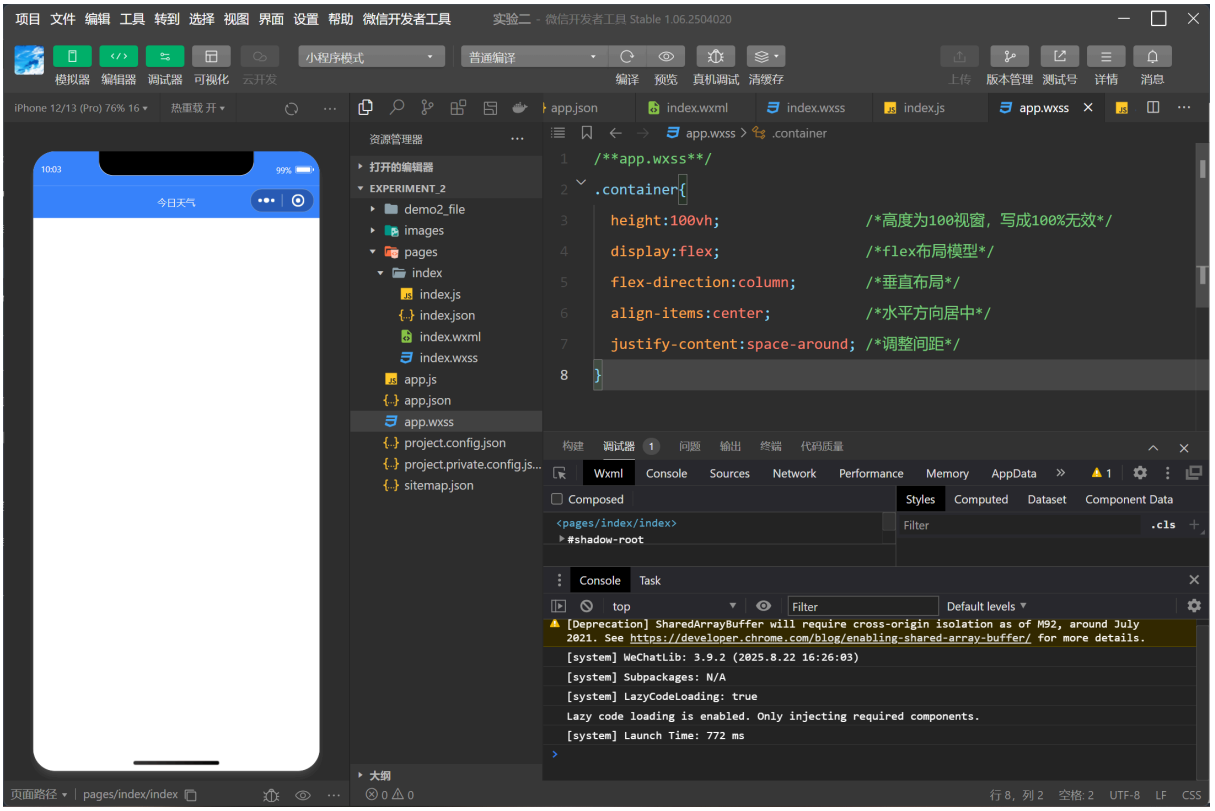
```
"window": {
  "navigationBarTextStyle": "white",
  "navigationBarTitleText": "今日天气",
  "navigationBarBackgroundColor": "#3883FA"
},
```

主页面包括四个区域，分别用于地区选择、显示当前城市温度和天气状态、显示当前城市天气图标以及分多行显示其他天气信息。先定义页面容器 <view> ，对其样式进行设计。不过页面当前没有内容，无法看到布局效果。

```
//index.wxml
<view class='container'>
</view>
```

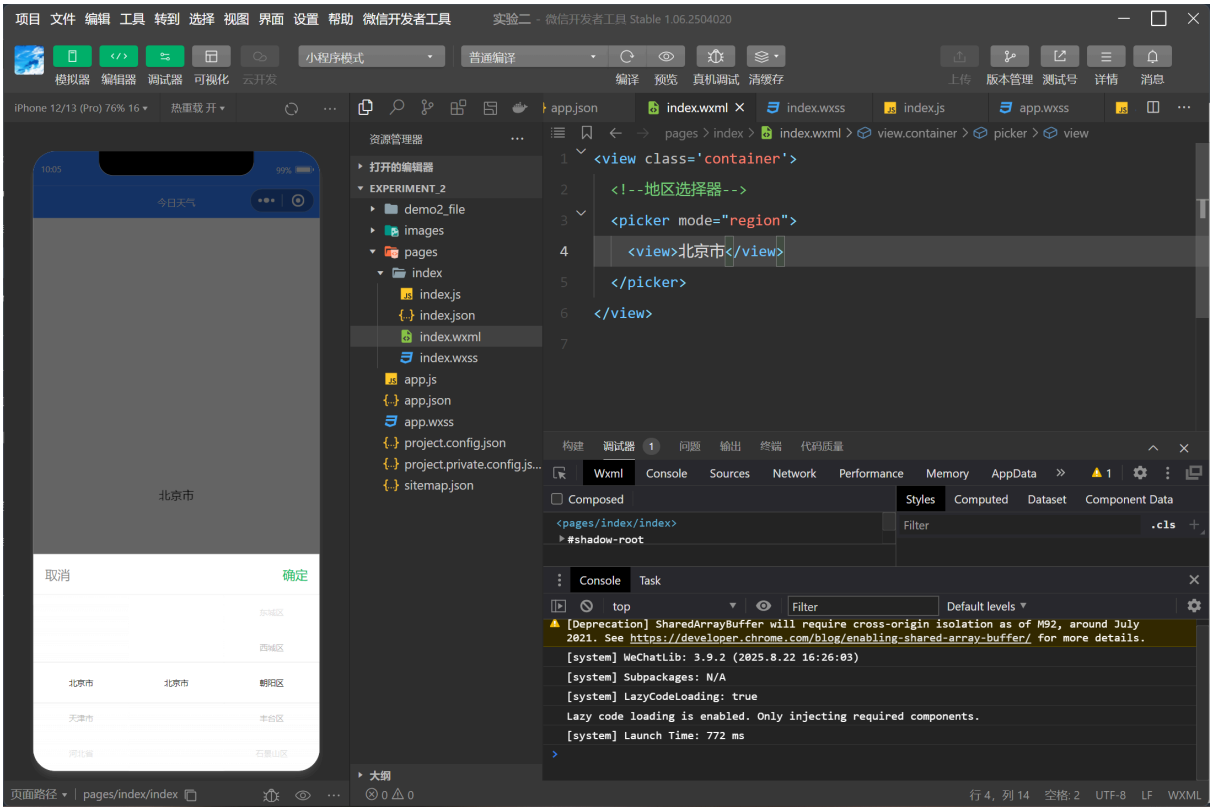
```
/*app.wxss*/
.container{
  height:100vh;              /*高度为100视窗，写成100%无效*/
  display:flex;             /*flex布局模型*/
```

```
  flex-direction:column;      /*垂直布局*/
  align-items:center;         /*水平方向居中*/
  justify-content:space-around; /*调整间距*/
}
```



4、实现地区选择功能。在 index.wxml 的 <view> 内部加入如下代码构建一个选择器，就可以实现地区选择的功能。

```
<!--地区选择器-->
<picker mode="region">
  <view>北京市</view>
</picker>
```



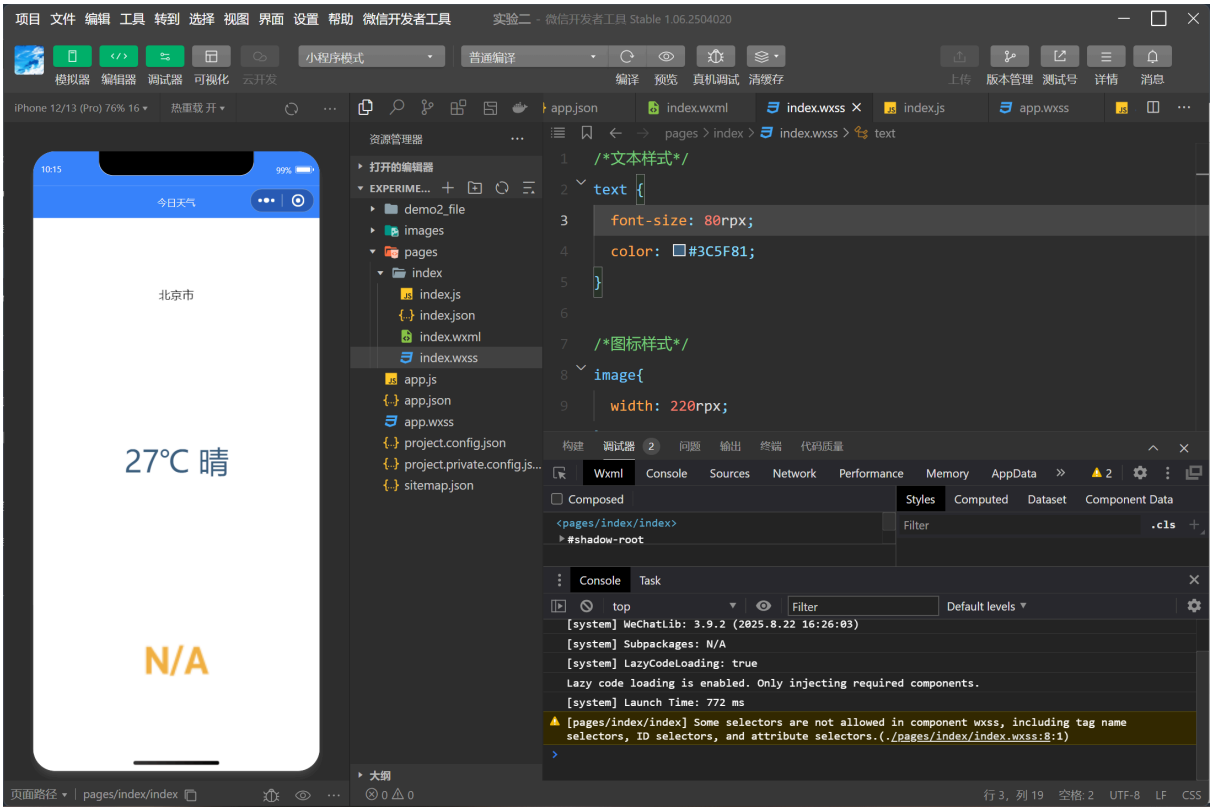
5、显示城市温度、天气状态以及天气图标。在 index.wxml 的 <view> 中添加温度文本和天气文本，并从项目的 images 文件夹中获取天气图标。

```
<!--温度和天气-->
<text>27°C 晴</text>
<!--天气图标-->
<image src='/images/weather_icon/999.png' mode="widthFix"></image>
```

同时需要在 index.wxss 中对文本和图标的样式进行定义。

```
/*文本样式*/
text {
  font-size: 80rpx;
  color: #3C5F81;
}

/*图标样式*/
image{
  width: 220rpx;
}
```



6、实现多行天气信息。在 index.wxml 的 <view> 中使用 bar 形成多行，使用 box 分割每行，实现六宫格的效果。

```
<!--多行天气信息-->
<view class='detail'>
  <view class='bar'>
    <view class='box'>湿度</view>
    <view class='box'>气压</view>
    <view class='box'>能见度</view>
  </view>
  <view class='bar'>
    <view class='box'> 0 % </view>
    <view class='box'> 0 hPa </view>
    <view class='box'> 0 km </view>
  </view>
  <view class='bar'>
    <view class='box'>风向</view>
    <view class='box'>风速</view>
    <view class='box'>风力</view>
  </view>
  <view class='bar'>
```

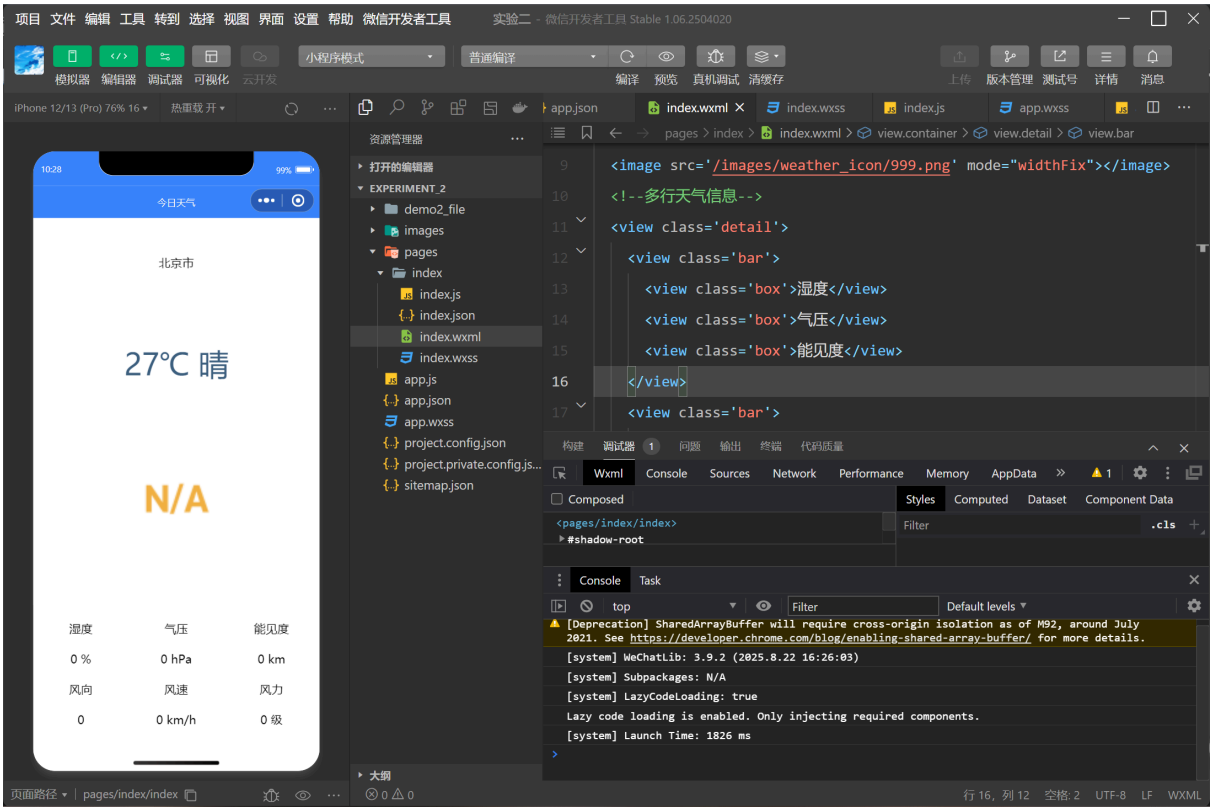
```
    <view class='box'> 0 </view>
    <view class='box'> 0 km/h</view>
    <view class='box'> 0 级</view>
  </view>
</view>
```

同样需要在 index.wxss 中添加代码，定义整个区域的元素排列方式以及单元行和单元格的位置细节。

```
/*多行天气样式*/
.detail {
  width: 100%;
  display: flex;
  flex-direction: column;
}

/*单元行样式*/
.bar {
  display: flex;
  flex-direction: row;
  margin: 20rpx 0;
}

/*单元格样式*/
.box {
  width: 33.3%;
  text-align: center;
}
```



7、更新省、市、区信息。修改 <picker> 中“北京市”为 {{region}}，然后为 <picker> 组件添加自定义 bindchange 事件，监听选项变化。修改 index.wxml 代码如下。

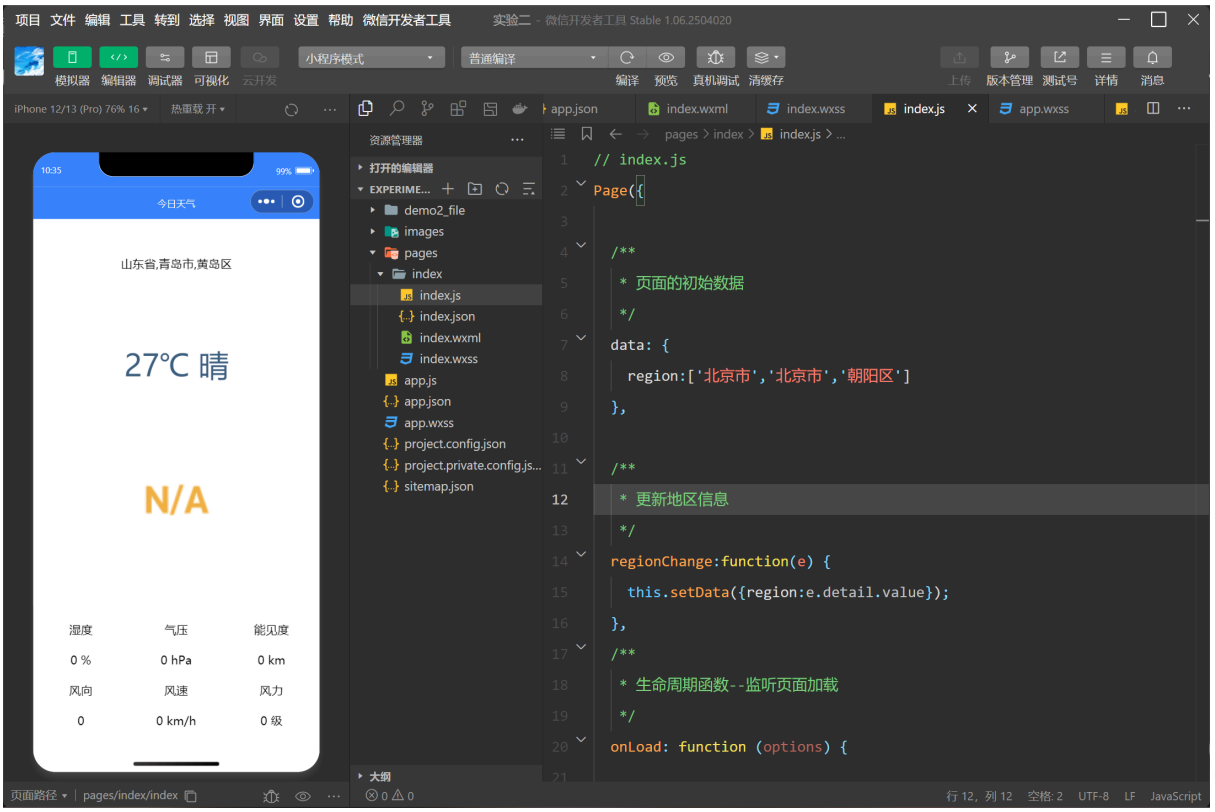
```
<picker mode="region" bindchange="regionChange">
  <view>{{region}}</view>
</picker>
```

在 index.js 中添加对应的处理逻辑。

```
/**
 * 页面的初始数据
 */
data: {
  region:['北京市','北京市','朝阳区']
},

/**
 * 更新地区信息
 */
regionChange:function(e) {
```

```
    this.setData({region:e.detail.value});
  },
```



8、获取实况天气数据。不同于实验文档中实时天气的获取方式，和风天气目前需要先提供城市名称利用 `/geo/v2/city/lookup` 进行搜索，获取城市的 locationID 值，再利用 locationID 调用 `/v7/weather/now` 获取城市的天气数据。

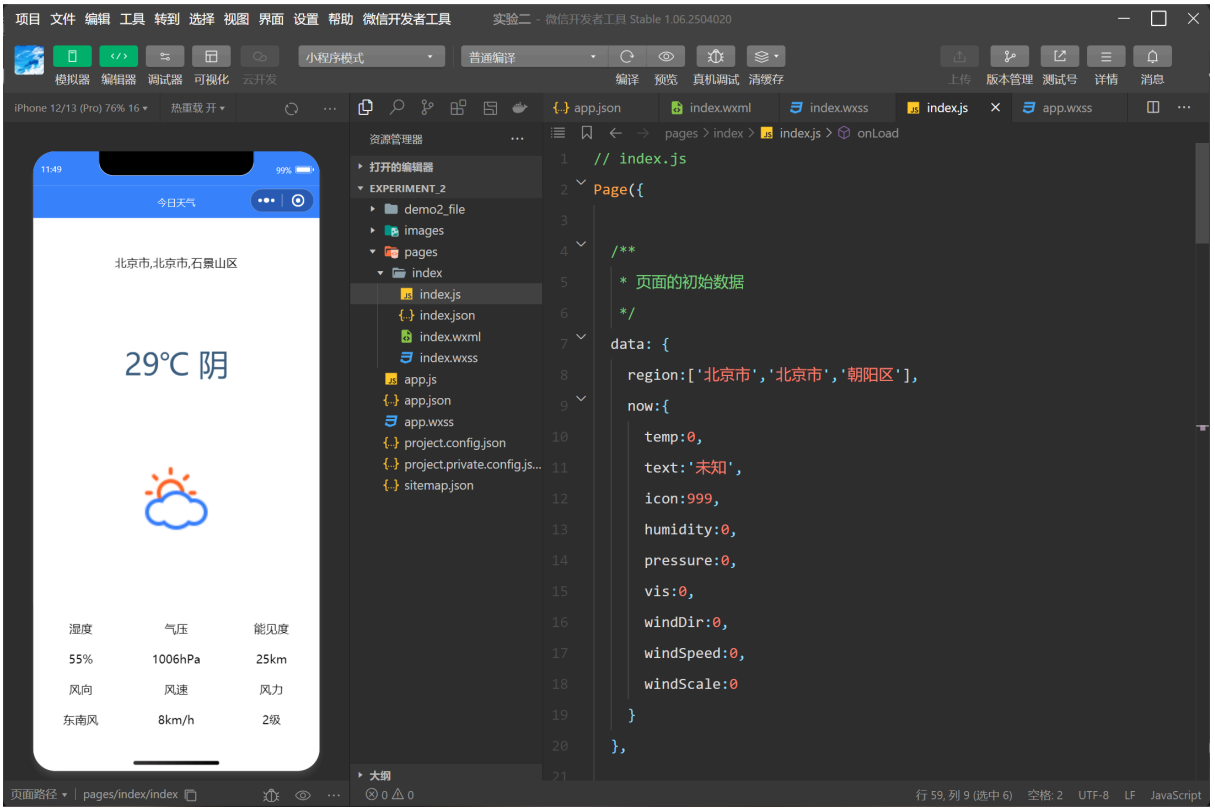
获取天气的函数如下，`that.data.region[2]` 可以将城市精确到县区，获取到的天气数据存放于 `now` 变量中。

```
/**
 * 获取天气信息
 */
getWeather:function() {
  var that = this;
  wx.request({
    url: 'https://ka67cdph36.re.qweatherapi.com/geo/v2/city/lookup',
    data: {
      location:that.data.region[2],
      key:'f966a5e556894308898c6367461751d7'
    },
```

```
    success(res1) {
      // const cityData = res1.data.location[0];
      wx.request({
        url: 'https://ka67cdph36.re.qweatherapi.com/v7/weather/now',
        data: {
          location:res1.data.location[0].id,
          key:'f966a5e556894308898c6367461751d7'
        },
        success:function(res2) {
          that.setData({now:res2.data.now});
        }
      })
    }
  })
},
```

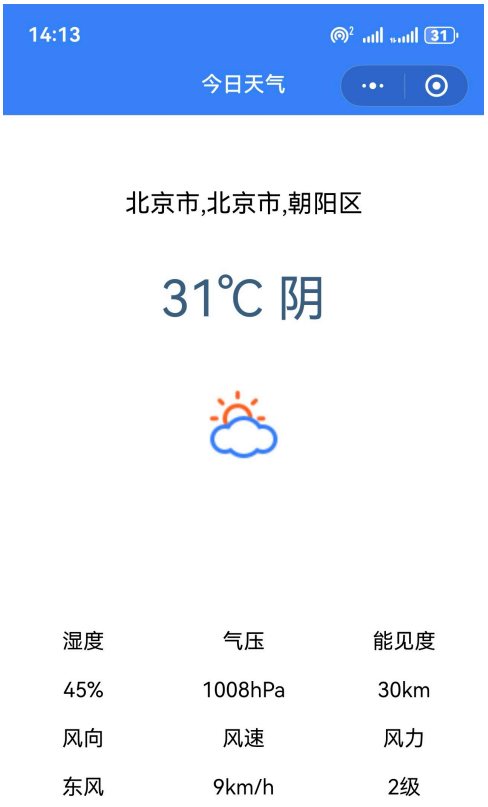
在更改城市和小程序加载时，需要调用 `getWeather` 函数，获取天气数据，因此要在 `regionChange` 和 `onLoad` 末尾添加 `this.getWeather();`

接下来可以将获取的数据显示出来，用 `{{now.xxx}}` 替换 index.wxml 中原本的固定数据，要注意变量名有所更改，和实验文档中的不同。可以在 index.js 中为这些变量设定一个初值，用于网络不佳时显示。



三、程序运行结果

按照实验文档实现的小程序在真机上运行的效果如下图，相对比较简单，也不太美观。



对页面的设计进行了一定优化，调整了文字位置和大小，为不同区域设置了背景。

四、问题总结与体会

遇到的几个问题：

遇到的最大问题是和风天气修改了调用 API 的方式，导致实验文档上的方法不再适用。一个问题是需要 API Host 和 API KEY 配合才能构建请求 URL，而不是文档上的认证 key。另一个问题是获取天气数据的方式变得不那么直接了，需要先根据城市名查询到城市 ID，再查询天气，具体的流程在网上搜索比较新的教程可以看到。

体会：

小程序界面的构建难度并不大，困难的是逻辑实现。尤其是和风天气的 API 与实验文档不同，只能在查看和风天气的文档后仿照着网上的教程来。小程序提供的组件还是比较方便的，比如内置的选择器，能够满足不同类型的选择需求，也能让选择的结果通过事件和其他组件联动，实现改变地区刷新天气的效果。其内置的 `request` 函数也能非常方便地构建请求，获取服务器的信息，在获取数据上相当方便。