

详细设计说明书

1 引言

1.1 编写目的

本详细设计说明书,主要是为了让软件的使用者能快速便捷地了解软件的功能以及如何操作,并对使用者在使用过程中可能出现的问题进行解答与解析。配合本文使用,软件的使用者可以不用理解软件内部的架构,而是从软件外部显示出来的部分来观察,快速上手,并且可以避免因为操作错误或缺漏带来的软件错误。

1.2 背景

说明:

软件名称为 MyWeatherApp,顾名思义,这个程序的应用之处在于获知搜索地区当天以及后面两天的天气状况。值得一提的是,除此之外我们还添加了对搜索地区的地图、路面情况的展示。

本项目的任务提出者是任仕杰,由小组成员共同开发;目标用户是所有急需获得信息的人,他们可以使用该软件查询天气情况及路面信息,例如他们在出门之前使用软件,软件会给予他们不小的帮助;运行该程序系统的环境最好为 WIN10 系统,使用 VS2015 及以上的版本进行编译运行。

1.3 定义

本文件中用到专门术语的定义和外文首字母组词的原词组。

SQL: Structured Query Language

结构化查询语言,用于支持数据库应用。

UWP: Universal Windows Platform

Windows 通用应用平台,不同于传统 PC 上的 exe 应用,它并不是为某一个终端而设计,而是可以在所有 windows10 的设备上运行。UWP 是项目自适应设计的一个重要前提。

PCL: Point Cloud Library

是在吸收了前人点云相关研究基础上建立起来的大型跨平台开源 C++ 编程库，它实现了大量点云相关的通用算法和高效数据结构，涉及到点云获取、滤波、分割、配准、检索、特征提取、识别、追踪、曲面重建、可视化等。支持多种操作系统平台，可在 Windows、Linux、Android、Mac OS X、部分嵌入式实时系统上运行。













1.4 参考资料

1、Imagine Cup 2016 微软“创新杯”全球学生科技大赛中国区比赛项目计划书

Imagine Cup 2016 微软“创新杯”全球学生科技大赛 中国区比赛项目计划书

参赛信息	
参赛队伍名称	miraculous yousters
参赛作品名称	MY weather forecast and maps
队长姓名	任仕杰
学校名称	中山大学
联系电话	13590430809
电子邮箱	409871419@qq.com

2、现代操作系统应用开发课件：

	01-UWPOverview	2016/1/7 18:54	PPTX 演示文稿	4,962 KB
	02-XAMLcontrols	2016/1/7 18:54	PPTX 演示文稿	5,541 KB
	03-PageNavigation	2016/1/7 18:54	PPTX 演示文稿	1,324 KB
	04-AdaptiveUI	2016/1/7 18:54	PPTX 演示文稿	52,459 KB
	05-AdaptiveCode	2016/1/7 2:54	PPTX 演示文稿	1,352 KB
	06-XAMLDataBinding	2016/1/7 2:54	PPTX 演示文稿	1,046 KB
	07-XAMLPerformance	2016/1/7 2:54	PPTX 演示文稿	761 KB
	10-ApplicationLifecycle	2016/1/7 2:54	PPTX 演示文稿	3,231 KB
	11-BackgroundExecution	2016/1/7 2:54	PPTX 演示文稿	2,238 KB
	13-LiveTiles-new	2017/3/16 20:31	PPTX 演示文稿	4,307 KB
	13-LiveTilesNotifications	2016/1/7 2:54	PPTX 演示文稿	5,214 KB
	15-AppToAppCommunication	2017/3/16 20:56	PPTX 演示文稿	2,027 KB

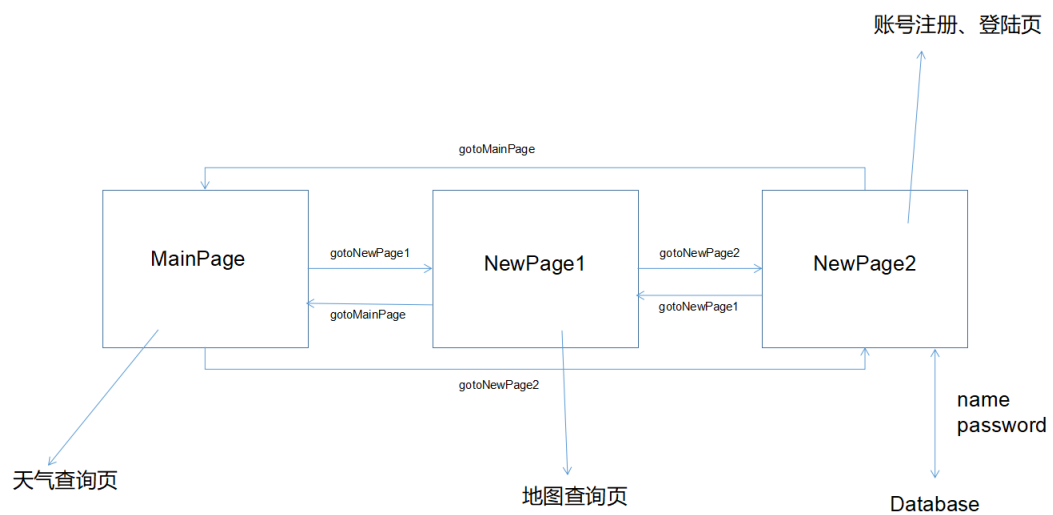
3、VS2015 系统软件开发参考链接

<https://msdn.microsoft.com/zh-cn/library>

4、课程作业及 demo

		2017-homework6	2
		2017-homework5	2
		2017-homework4	2
		2017homework3v2	2
2017-week6-课件	20	2017-homework3-demo2	2
2017week4demos	20	2017-homework2	2
2017week4demos (1)	20	2017-homework1	2
2017week3demo	20		

2 程序系统的结构





NewPage2



3 程序设计说明

程序共有三个页面：

天气查询页作为主页面，可以查询天气。

地图查询页可以根据地点查询地图。

登录页可以登录 app，并收藏地点。

- 页面导航：页面之间的跳转，通过侧边栏的几个按钮实现
- 数据绑定：在第二个页面，绑定了 checkbox 的 isChecked 属性和按钮的 content 属性，方便用户对查询进行选择

在 NewPage1.xaml.cs 里面的 searchMap 函数

```
/*
 * 分为地图查找和交通路况显示，由checkboxbox控制
 */
private async void searchMap(object sender, RoutedEventArgs e) { ... }
```

- 后台运行与程序生命周期：挂起时保存页面信息（如查询框中的信息），重新运行后恢复信息

在 MainPage.xaml.cs、NewPage1.xaml.cs、NewPage2.xaml.cs 三个文件中的 OnNavigatedTo、OnNavigatedFrom 函数实现后台运行和程序生命周期、程序间通信，并且有挂起判定，在挂起时保存页面信息。

- 程序间通信：分享天气信息

MainPage.xaml.cs 的 shareWeather 是点击分享时的 click 事件，将调用起分享的 UI。

MainPage.xaml.cs 的 OnShareRequested 则是实现分享时的程序间通信功能。

- 动态磁贴：每次查询都会进行更新，把当日天气、明后日天气装载到动态磁贴中，循环展示

实现主要是用 fileOpenPicker 和 Bitmap 数据流来实现图片的读取，LiveTile 主要是根据返回的 Json 文件键值对修改 XML 结点树上的对应值，动态的实现是通过开启通知队列并向通知队列里面 push 新的通知来实现动态更新。各个函数具体如下：

searchWeather:

天气查找按钮的点击事件,当查询得出正确结果时，更新磁贴

最初的磁贴初始化：

```
if (weatherSearchBlock.Text == "")
{
    string url = "http://restapi.amap.com/v3/ip?output=xml&key=5fd3b8bd943a505ccfec387943bba945";
    HttpClient client = new HttpClient();
    string result = await client.GetStringAsync(url);
    XmlDocument document = new XmlDocument();
    document.LoadXml(result);
    XmlNodeList list = document.GetElementsByTagName("status");
    XmlNode node = list.Item(0);
    string querySuccess = node.InnerText;
    if (querySuccess == "0")
    {
        XmlNodeList li = document.GetElementsByTagName("info");
        XmlNode no = li.Item(0);
        var i = new MessageDialog(no.InnerText).ShowAsync();
    } else
    {
        list = document.GetElementsByTagName("adcode");
        node = list.Item(0);
        string adcode = node.InnerText;

        string url2 = "http://restapi.amap.com/v3/weather/weatherInfo?key=5fd3b8bd943a505ccfec387943bba945&extensions=all&city=" + adcode;
        string result2 = await client.GetStringAsync(url2);
        JObject jo = (JObject)JsonConvert.DeserializeObject(result2);
        string querySuccess2 = jo["status"].ToString();
    }
}
```

当前日期的天气显示:

```
currentWeatherBlock.Text += ja2[0]["date"].ToString() + "\n";
currentWeatherBlock.Text += "星期" + ja2[0]["week"].ToString() + "\n";
currentWeatherBlock.Text += ja2[0]["dayweather"].ToString() + "\n";
if (ja2[0]["dayweather"].ToString() == "晴")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/晴.jpg"));
    icon1.Source = image;
} else if (ja2[0]["dayweather"].ToString() == "小雨")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/小雨.jpg"));
    icon1.Source = image;
} else if (ja2[0]["dayweather"].ToString() == "中雨" || ja2[0]["dayweather"].ToString() == "大雨")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/大雨.jpg"));
    icon1.Source = image;
} else if (ja2[0]["dayweather"].ToString() == "暴雨" || ja2[0]["dayweather"].ToString() == "大暴雨")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/暴雨.jpg"));
    icon1.Source = image;
} else if (ja2[0]["dayweather"].ToString() == "雾")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/雾.jpg"));
    icon1.Source = image;
} else if (ja2[0]["dayweather"].ToString() == "多云")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/多云.jpg"));
    icon1.Source = image;
} else
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/阴.jpg"));
    icon1.Source = image;
}
currentWeatherBlock.Text += "日间温度: " + ja2[0]["daytemp"].ToString() + "\n";
currentWeatherBlock.Text += "夜间温度: " + ja2[0]["nighttemp"].ToString() + "\n";
```

显示出当前城市, 最后更新时间, 当前日期, 星期, 并获得天气情况并加载对应图片资源, 显示日间温度与晚间温度。

对之后两个天气情况的磁贴:

```
weatherForecast1.Text += ja2[1]["date"].ToString() + "\n";
weatherForecast1.Text += "星期" + ja2[1]["week"].ToString() + "\n";
weatherForecast1.Text += ja2[1]["dayweather"].ToString() + "\n";
if (ja2[1]["dayweather"].ToString() == "晴")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/晴.jpg"));
    icon2.Source = image;
} else if (ja2[1]["dayweather"].ToString() == "小雨")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/小雨.jpg"));
    icon2.Source = image;
} else if (ja2[1]["dayweather"].ToString() == "中雨" || ja2[0]["dayweather"].ToString() == "大雨")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/大雨.jpg"));
    icon2.Source = image;
} else if (ja2[1]["dayweather"].ToString() == "暴雨" || ja2[0]["dayweather"].ToString() == "大暴雨")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/暴雨.jpg"));
    icon2.Source = image;
} else if (ja2[1]["dayweather"].ToString() == "雾")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/雾.jpg"));
    icon2.Source = image;
} else if (ja2[1]["dayweather"].ToString() == "多云")
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/多云.jpg"));
    icon2.Source = image;
} else
{
    BitmapImage image = new BitmapImage(new Uri("ms-appx:Assets/阴.jpg"));
    icon2.Source = image;
}
```

与之前基本相同，只是去除了当前城市和最后更新时间。

经过查询之后的磁贴更新：

```
} else
{
    string url = "http://restapi.amap.com/v3/config/district?key=5fd3b8bd943a505ccfec387943bba945&subdistrict=0&showbiz=false&extensions=base&keywords=" + wea
    HttpClient client = new HttpClient();
    string result = await client.GetStringAsync(url);
    JObject jo = (JObject)JsonConvert.DeserializeObject(result);
    if (jo["status"].ToString() == "0" || jo["count"].ToString() == "0")
    {
        var i = new MessageDialog("查无此地区").ShowAsync();
    } else
    {
        JArray ja = (JArray)jo["districts"];
        string adcode = ja[0]["adcode"].ToString();
        string url2 = "http://restapi.amap.com/v3/weather/weatherInfo?key=5fd3b8bd943a505ccfec387943bba945&extensions=all&city=" + adcode;
        string result2 = await client.GetStringAsync(url2);
        JObject jo2 = (JObject)JsonConvert.DeserializeObject(result2);
        string querySuccess2 = jo2["status"].ToString();
        if (querySuccess2 == "0")
        {
            var i = new MessageDialog(jo2["info"].ToString()).ShowAsync();
        }
        else
    }
}
```

更换城市，更新最后时间，得出相关数据之后。磁贴的显示与初始化相同，不在赘述。

天气板块在整个磁贴的初始化与更新：

```
if (currentWeatherBlock.Text != "")
{
    TileUpdateManager.CreateTileUpdaterForApplication().Clear();
    XmlDocument xml = await XmlDocument.LoadFromFileAsync(await StorageFile.GetFileFromApplicationUriAsync(new Uri("ms-appx:///tile.xml")));
    string changeContent = xml.GetXml().Replace("Text", currentWeatherBlock.Text);
    XmlDocument newXml = new XmlDocument();
    newXml.LoadXml(changeContent);
    TileNotification update = new TileNotification(newXml);
    Random a = new Random();
    string id = a.Next(1000).ToString();
    update.Tag = id;
    TileUpdateManager.CreateTileUpdaterForApplication().Update(update);
}
```

通过读取 tile.xml 来进行详细的初始化要求，并进行更新。天气预报也是同样的处理。到此为止就是 serachWeather 函数的具体分析。

OnLaunched：

用于当该应用被打开时的一些操作

初始化：

```
/// <param name="e">Details about the launch request and process.</param>
protected override void OnLaunched(LaunchActivatedEventArgs e)
{
    TileUpdateManager.CreateTileUpdaterForApplication().EnableNotificationQueue(true);
    NewPage2.userName = "";
    MainPage.currentCity = "";
    NewPage1.currentCity = "";

    #if DEBUG
    if (System.Diagnostics.Debugger.IsAttached)
    {
        this.DebugSettings.EnableFrameRateCounter = true;
    }
    #endif
}
```

通过利用 frame 引导第一页的内容：

```
Frame rootFrame = Window.Current.Content as Frame;

// Do not repeat app initialization when the window already has content,
// just ensure that the window is active
if (rootFrame == null)
{
    // Create a Frame to act as the navigation context and navigate to the first page
    rootFrame = new Frame();

    rootFrame.NavigationFailed += OnNavigationFailed;

    if (e.PreviousExecutionState == ApplicationExecutionState.Terminated)
    {
        if (ApplicationData.Current.LocalSettings.Values.ContainsKey("NavigationState"))
        {
            rootFrame.SetNavigationState((string)ApplicationData.Current.LocalSettings.Values["NavigationState"]);
        }
    }

    // Place the frame in the current window
    Window.Current.Content = rootFrame;
}
```

当导航的栈无法恢复到第一页的状态时，将通过需要的信息来作为导航，来完成新请求页面的配置：

```
if (e.PrelaunchActivated == false)
{
    if (rootFrame.Content == null)
    {
        // When the navigation stack isn't restored navigate to the first page,
        // configuring the new page by passing required information as a navigation
        // parameter
        rootFrame.Navigate(typeof(MainPage), e.Arguments);
    }
    // Ensure the current window is active
    Window.Current.Activate();
}

LoadDatabase();
```

- 文件管理：打开一个选择图片的 filePicker，供用户选择头像

selectPicture:

用于文件管理，选择图片添加到头像框

```
private async void selectPicture(object sender, RoutedEventArgs e)
{
    FileOpenPicker picker = new FileOpenPicker();
    picker.ViewMode = PickerViewMode.List;
    picker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;
    picker.FileTypeFilter.Add(".jpg");
    picker.FileTypeFilter.Add(".png");
    picker.FileTypeFilter.Add(".jpeg");
    StorageFile file = await picker.PickSingleFileAsync();
    if (file != null)
    {
        using (IRandomAccessStream fileStream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read))
        {
            BitmapImage bitmapImage = new BitmapImage();
            //这里是选定的图片的宽度，根据UI修改一下
            //bitmapImage.DecodePixelWidth = 400;
            await bitmapImage.SetSourceAsync(fileStream);
            //图片控件的名称为Icon
            Icon.Source = bitmapImage;
        }
    }
    else
    {
        var message = new MessageDialog("Did not Pick anything !").ShowAsync();
    }
}
```

通过打开文件，筛选 jpg, png, jpeg 3 种格式的图片进行上传，并更新为头像。如果选取失败，则返回错误提示。

- 数据库：用于管理用户信息和城市信息

Prepare

```
using (var statement = db.Prepare("SELECT * FROM Favourites WHERE UserName LIKE ?"))
{
    statement.Bind(1, NewPage2.userName);
    while (SQLiteResult.ROW == statement.Step())
    {
        result += (string)statement[2] + "\n";
    }
}
```


在 prepare () 中使用的是标准的 SQL 指令。

对于具体的数值类的量就用? 代替，同时需要额外添加绑定指令 Bind()，把 prepare () 中用? 表示的量替换成相对应的数值量，而且替换是是将 NewPage2.userName[即 Bind()中第 1 个参数]替换成第 1[即 Bind()中第 0 个参数的数值]个?，然后在执行 exec () 时就可以更新到数据库中。

- 网络访问：调用高德 API 并解析，使它反馈到页面上，并有一定的异常处理

addFavourite

在该项目中有两个 addFavourite 函数，分别在两个地方出现：MainPage.xaml.cs 和 NewPage1.xaml.cs。NewPage1 的 addFavourite 函数与 MainPage 的 addFavourite 函数相同。

addFavourite:添加收藏

判断是否登录和是否选中地点

```
private void addFavourite(object sender, RoutedEventArgs e)
{
    if (NewPage2.userName == "")
    {
        var i = new MessageDialog("未登录").ShowAsync();
        return;
    }

    if (currentCity == "")
    {
        var i = new MessageDialog("未选中收藏地点").ShowAsync();
    } else
    {
```

若登录并且选中了地点，则进行收藏操作，即在数据库中进行插入 ID，用户名和收藏地点的操作

```
    } else
    {
        string id = Guid.NewGuid().ToString();
        var db = App.connection.GetInstance().conn;
        using (var statement = db.Prepare("INSERT INTO Favourites (Id, UserName, City) VALUES (?, ?, ?)"))
        {
            statement.Bind(1, id);
            statement.Bind(2, NewPage2.userName);
            statement.Bind(3, currentCity);
            statement.Step();
        }
    }
}
```

showCollection

在该项目中有两个 showCollection 函数，分别出现在两处：NewPage1.xaml.cs 和

MainPage.xaml.cs。NewPage1 的 showCollection 函数与 MainPage 的 showCollection 函数相同。

showCollection:显示收藏的城市

先判断是否登录。若已登录，从数据库中调取登录用户收藏的所有城市。

```
private void showCollection(object sender, RoutedEventArgs e)
{
    if (NewPage2.userName == "")
    {
        var j = new MessageDialog("未登录").ShowAsync();
        return;
    }

    var db = App.connection.GetInstance().conn;
    string result = "收藏的地点:  \n";
    using (var statement = db.Prepare("SELECT * FROM Favourites WHERE UserName LIKE ?"))
    {
        statement.Bind(1, NewPage2.userName);
        while (SQLiteResult.ROW == statement.Step())
        {
            result += (string)statement[2] + "\n";
        }
    }

    var i = new MessageDialog(result).ShowAsync();
}
```

searchMap

分为地图查找和交通路况显示，由 checkbox 控制。

交通路况显示的情况下,需判断能否查询到此地区，如果能，则向对应网址发送 GET 请求，并根据请求生成地图。

```
if (jo["status"].ToString() == "0" || jo["count"].ToString() == "0")
{
    var i = new MessageDialog("查无此地区").ShowAsync();
} else
{
    JArray ja = (JArray)jo["geocodes"];
    string location = ja[0]["location"].ToString();
    currentCity = ja[0]["formatted_address"].ToString();

    string url2 = "http://restapi.amap.com/v3/staticmap?key=5fd3b8bd943a505ccfec387943bba945&locat
    HttpResponseMessage response = await client.GetAsync(url2);
    BitmapImage bitmap = new BitmapImage();
    Stream stream = await response.Content.ReadAsStreamAsync();
    IInputStream inputStream = stream.AsInputStream();
    using (IRandomAccessStream randomAccessStream = new InMemoryRandomAccessStream())
    {
        using (IOutputStream outputStream = randomAccessStream.GetOutputStreamAt(0))
        {
            await RandomAccessStream.CopyAsync(inputStream, outputStream);
            randomAccessStream.Seek(0);
            bitmap.SetSource(randomAccessStream);
            map.Source = bitmap;
        }
    }
}
```

交通路况不显示的情况与显示的情况相比，代码基本相同，只是发送 GET 请求的网址不同。

Login

Login: 登录

需判断用户名和密码是否为空。

```
if (nameBlock.Text == "")
{
    var i = new MessageBox("用户名不能为空").ShowAsync();
    return;
}
if (passwordBlock.Password == "")
{
    var i = new MessageBox("密码不能为空").ShowAsync();
    return;
}
```

用户名和密码都不为空情况下，查找数据库是否有用户名与输入的用户名一致，返回 result 和 pw。根据 result 判断数据库中是否有改用户名。根据 pw 判断密码是否正确。

```
using (var statement = db.Prepare("SELECT * FROM Info WHERE UserName LIKE ?"))
{
    statement.Bind(1, nameBlock.Text);
    while (SQLiteResult.ROW == statement.Step())
    {
        result += (string)statement[0];
        pw += (string)statement[1];
    }
}

if (result == "")
{
    using (var statement2 = db.Prepare("INSERT INTO Info (UserName, Password) VALUES (?, ?)"))
    {
        statement2.Bind(1, nameBlock.Text);
        statement2.Bind(2, passwordBlock.Password);
        statement2.Step();
    }
    userName = nameBlock.Text;
    var i = new DialogResult("注册成功").ShowAsync();
} else
{
    if (passwordBlock.Password == pw)
    {
        userName = nameBlock.Text;
        var i = new DialogResult("登录成功").ShowAsync();
    } else
    {
        var i = new DialogResult("登录失败，密码不正确").ShowAsync();
    }
}
```