小程序云开发·基于全套云能力的图片分享社区

在学习了小程序云开发的基础知识后，读者不妨尝试综合应用云数据库、云函数和云存储制作一款图片分享社区小程序。用户可以上传图片到云存储空间中，也可以查看其他用户上传的图片列表，并进行图片分享、下载和全屏预览等。

本章学习目标

- 综合应用小程序云开发的基础知识创建图片分享社区小程序；
- 掌握云数据集创建、云存储管理和云函数调用等知识。

19.1 初始化项目

19.1.1 创建云模板项目



视频讲解

首先需要创建一个云开发项目，在任意盘符下创建一个空白文件夹（例如cloudPhoto），然后填入 AppID 和选中“小程序·云开发”，如图 19-1 所示。

接着删除其中无用的模板代码：

（1）删除 cloudFunctions 文件夹下的默认云函数 login 的全部内容。

（2）找到 app.json 文件，打开并删除其中的页面引用，只保留第一个 pages/index/index。

（3）打开硬盘中的 pages 文件夹，删除 index 以外的所有目录。

（4）进入 index 文件夹，删除多余的图片，以及 JS、WXML 和 WXSS 文件中的全部代码（JS 文件删除干净后输入关键词 page 让它自动生成 Page({})函数的相关代码）。

（5）删除 app.wxss 中的代码。

（6）删除 image 文件夹中的所有图片。

（7）删除 style 文件夹，此时项目清理全部完成。

19.1.2 部署云数据库



视频讲解

具体操作如下：

（1）打开云开发控制台，创建一个新的数据集，例如 photos。

（2）检查 photos 数据集的权限，确认是“所有用户可读，仅创建者及管理

员可写”，完成。



图 19-1　云模板项目填写效果示意图

效果如图 19-2 所示。

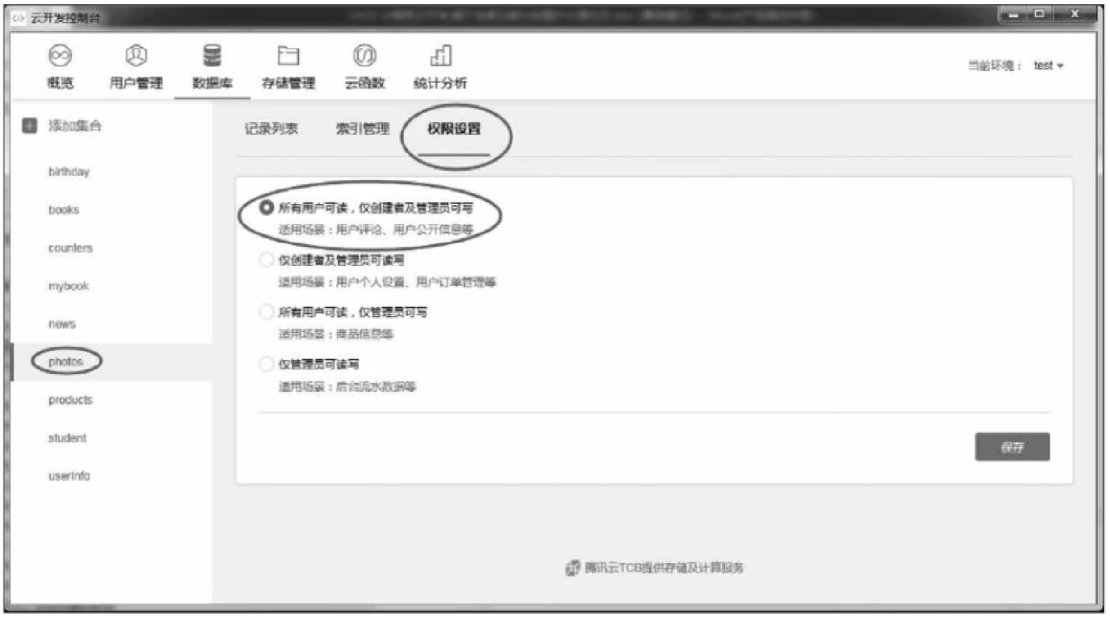


图 19-2　在云开发控制台添加数据集 photos 并设置权限

19.1.3　创建页面文件



视频讲解

本项目有 4 个页面文件，分别是 index（首页）、homepage（个人主页）、detail（图片展示页）和 add（上传图片页）。打开 app.json 文件，在 pages 属性中追加除 index 之外的其他页面的路径描述，如图 19-3 所示。

保存后会自动生成后面 3 个页面，此时 pages 文件夹内部的目录结构如图 19-4 所示。此时文件配置就全部完成，19.2 节将正式进行页面布局和样式设计。

```
app.json
1  {
2    "pages": [
3      "pages/index/index",
4      "pages/homepage/homepage",
5      "pages/detail/detail",
6      "pages/add/add"
7    ],
```

图 19-3　在 app.json 中添加页面配置的代码

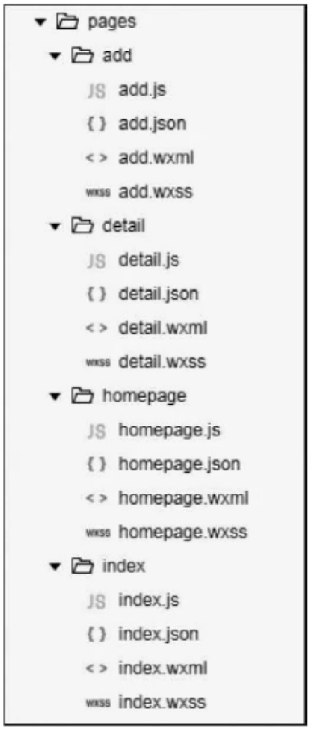


图 19-4　页面文件创建完成

19.2　视图设计

19.2.1　导航栏设计



视频讲解

用户可以通过在 app.json 中对 window 属性进行重新配置来自定义导航栏效果。

更改后的 app.json 文件代码如下：

```
1.  {
2.    "pages":[代码略],
3.    "window":{
4.      "navigationBarBackgroundColor":"#F6F6F6",
5.      "navigationBarTitleText":"图片分享社区",
6.      "navigationBarTextStyle":"black"
7.    }
8.  }
```

上述代码可以更改导航栏背景色为浅灰色、字体为黑色，效果如图 19-5 所示。

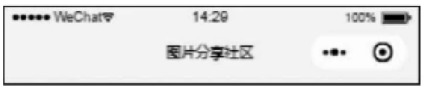


图 19-5　自定义导航栏效果

19.2.2　页面设计



1　首页设计

首页主要是图片展示区域，用于展示多名作者上传的图片作品。每张图

片都以卡片的形式出现，卡片包括页眉、主体和页脚部分。除此之外还需要在右上角放置一个浮动按钮，点击此按钮可以上传图片到云存储空间。



图 19-6　首页设计图

页面设计如图 19-6 所示。

计划使用如下组件。

- 整体容器：容器组件（< view >），并定义 class = 'container'；
- 整张卡片：容器组件（< view >），并定义 class = 'card'；
- 卡片页眉、主体和页脚：容器组件（< view >），并分别定义 class = 'card-head'、'card-body' 和 'card-foot'；
- 作者头像图标和上传的图片：图片组件（< image >）；
- 作者昵称、所在地以及图中上传时间：文本组件（< text >）；
- 浮动按钮：按钮组件（< button >）。

WXML（pages/index/index. wxml）代码如下：

```
1.  < view class = 'container'>
2.    <!-- 卡片 -->
3.    < view class = 'card'>
4.      <!-- 卡片页眉 -->
5.      < view class = 'card - head'>
6.        < image class = 'avatar' src = '/images/avatar. jpg' mode = 'widthFix'></image >
7.        < view class = 'title'>
8.          < text class = 'nickName'>周小贝</text >
9.          < text class = 'address'> Anhui, China </text >
10.       </view >
11.     </view >
12.
13.     <!-- 卡片主体 -->
14.     < view class = 'card - body'>
15.       < image src = '/images/pic001. jpg' mode = 'widthFix'></image >
16.     </view >
17.
18.     <!-- 卡片页脚 -->
19.     < view class = 'card - foot'>
20.       < text > 2019 - 06 - 21 </text >
21.     </view >
22.   </view >
23. </view >
24.
25. <!-- “上传图片”按钮 -->
26. < view class = 'floatBtn'>
27.   < button size = 'mini' type = 'primary'>上传图片</button >
28. </view >
```

由于整体容器、卡片样式也会在其他页面使用，所以将这两部分样式代码写到公共样式表 app. wxss 中，具体代码如下：

```
1.  /* 1. 整体容器样式 */
2.  .container {
3.    display: flex;
4.    flex - direction: column;
```

```
/* 浮动按钮样式 */
.floatBtn{
  position: fixed;
  top: 50rpx;
  right: 50rpx;
}
```

上述代码表示无论首页如何下拉滚动，浮动按钮都永远固定在距离顶端和右边 50rpx 的位置上不变。当前效果如图 19-7 所示。

需要注意的是，这里所展示的头像、昵称、所在地、图片和上传日期都是为了演示效果临时填写的，后续会换成实际录入的信息，并形成卡片列表效果。



视频讲解

2 个人主页设计

个人主页除了顶部会显示当前作者的头像、昵称外，下方仍然是和首页一样布局的图片展示区域，仅限于展示当前作者上传的图片作品。

页面设计如图 19-8 所示。

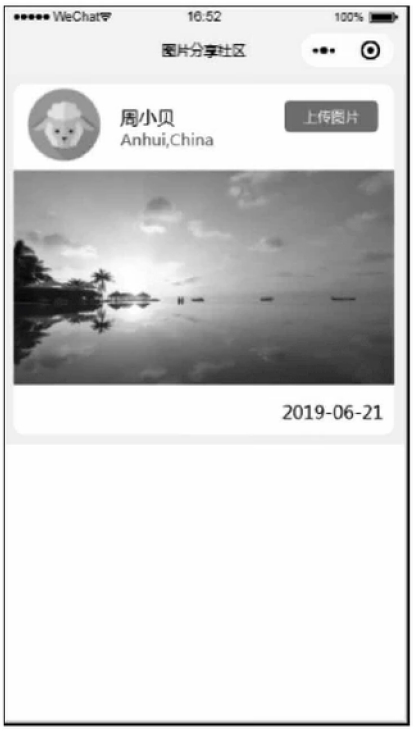


图 19-7　首页效果图



图 19-8　个人主页设计图

计划使用如下组件。

- 整体容器：容器组件(< view >)，并定义 class＝'container'；
- 顶端头像和昵称区域：容器组件(< view >)，并定义 class＝'avatarBox'；
- 整张卡片：容器组件(< view >)，并定义 class＝'card'；
- 卡片页眉、主体和页脚：容器组件(< view >)，并分别定义 class＝'card-head'、'card-body'和'card-foot'；
- 作者头像图标和上传的图片：图片组件(< image >)；
- 作者昵称和上传时间：文本组件(< text >)。

WXML(pages/homepage/homepage.wxml)代码如下：

```
<view class = 'container'>
  <!-- 头像和昵称 -->
  <view class = 'avatarBox'>
    <image src = '/images/avatar.jpg'></image>
    <text>周小贝个人主页</text>
  </view>

  <!-- 卡片 -->
  <view class = 'card'>
    <!-- 卡片页眉 -->
    <view class = 'card - head'>
      <image class = 'avatar' src = '/images/avatar.jpg' mode = 'widthFix'></image>
      <view class = 'title'>
        <text class = 'nickName'>周小贝</text>
        <text class = 'address'> Anhui, China </text>
      </view>
    </view>

    <!-- 卡片主体 -->
    <view class = 'card - body'>
      <image src = '/images/pic01.jpg' mode = 'widthFix'></image>
    </view>

    <!-- 卡片页脚 -->
    <view class = 'card - foot'>
      <text> 2019 - 06 - 21 </text>
    </view>
  </view>
</view>
```

整体容器和卡片的样式代码与首页相同，可以自动继承于公共样式表 app.wxss，无须重复声明，而顶部头像和昵称的相关样式写到对应的 homepage.wxss 中即可。

WXSS(pages/homepage/homepage.wxss)代码如下：

```
/* 头像区域 */
.avatarBox{
  display: flex;
  flex - direction: column;
  align - items: center;
  padding - top: 100rpx;
}
/* 头像区域中的头像图片 */
.avatarBox image{
  width: 200rpx;
  height: 200rpx;
  border - radius: 50 % ;
}
/* 头像区域中的昵称文本 */
.avatarBox text{
  margin: 30rpx 0;
}
```

当前效果如图 19-9 所示。

同样这里所展示的头像、昵称、图片以及上传日期都是为了演示效果临时填写的，后续会换成实际录入的信息，并形成卡片列表效果。

3 图片展示页设计



视频讲解

当用户点击卡片主体中的图片时会跳转到图片展示页查看完整图片效果，该页面主要包含完整图片和 3 个按钮，按钮分别用于下载图片、分享给好友以及全屏预览图片。

页面设计如图 19-10 所示。



图 19-9　个人主页效果图



图 19-10　图片展示页设计图

计划使用如下组件。

- 顶端图片展示：图片组件(<image>)；
- 按钮区域：容器组件(<view>)；
- 3 个按钮：按钮组件(<button>)。

WXML 文件(pages/detail/detail.wxml)代码如下：

```
1.  <!-- 顶端图片展示 -->
2.  <image src = '/images/pic01.jpg' mode = 'widthFix'></image>
3.
4.  <!-- 按钮区域 -->
5.  <view>
6.    <button type = 'primary' plain>下载到本地</button>
7.    <button type = 'primary' plain>分享给好友</button>
8.    <button type = 'primary' plain>全屏预览</button>
9.  </view>
```

WXSS(pages/detail/detail.wxss)代码如下：

```
1.  /* 顶端展示图片样式 */
2.  image{
3.    width: 750rpx;
4.  }
5.
```

```
/* 按钮外层 view */
view{
  padding: 0 50rpx;
}

/* 按钮样式 */
button{
  margin: 10rpx;
}
```

当前效果如图 19-11 所示。

这里的图片路径是为了演示效果临时填写的，后续会换成实际需要展示的图片。



视频讲解

4 上传图片页设计

点击首页中的“上传图片”按钮可以跳转到上传图片页，该页面主要包含按钮、标题和九宫格图片。页面设计如图 19-12 所示。



图 19-11　图片展示页效果图



图 19-12　上传图片页设计图

计划使用如下组件。

- 整体容器：容器组件(< view >)，并定义 class='container'；
- 按钮：按钮组件(< button >)；
- 标题：文本组件(< text >)；
- 九宫格图片展示：图片组件(< image >)。

WXML 文件(pages/add/add. wxml)代码如下：

```
<view class = 'container'>
  <!--“点击此处上传图片”按钮 -->
  <button type = 'warn'>点击此处上传图片</button>

```

```
    <!-- 已上传图片区域 -->
    <view class = 'photoBox'>
      <!-- 标题 -->
      <text >已上传图片历史记录</text >

      <!-- 图片集 -->
      <view >
        <image src = '/images/pic01.jpg'></image >
        <image src = '/images/pic01.jpg'></image >
        <image src = '/images/pic01.jpg'></image >
        <image src = '/images/pic01.jpg'></image >
        <image src = '/images/pic01.jpg'></image >
        <image src = '/images/pic01.jpg'></image >
      </view >
    </view >
</view >
```

整体容器的样式代码大部分与首页相同，可以自动继承于公共样式表 app.wxss，无须重复声明。在对应的 add.wxss 文件中还可以追加其他样式代码。

WXSS(pages/add/add.wxss)代码如下：

```
/ * 已上传图片区域 * /
.photoBox{
  margin - top: 50rpx;
  display: flex;
  flex - direction: column;
}
/ * 标题样式 * /
.photoBox text{
  margin - bottom: 30rpx;
  text - align: center;
}
/ * 图片样式 * /
.photoBox image{
  float: left;
  width: 250rpx;
  height: 250rpx;
}
```

当前效果如图 19-13 所示。

为了体现多张图片设计效果，这里重复了 6 次< image >组件的相关代码，仅作为临时参考。后续将读取用户真实上传的图片数量，展示历史记录。

此时页面布局与样式设计就已完成，19.3 节将介绍如何进行逻辑处理。

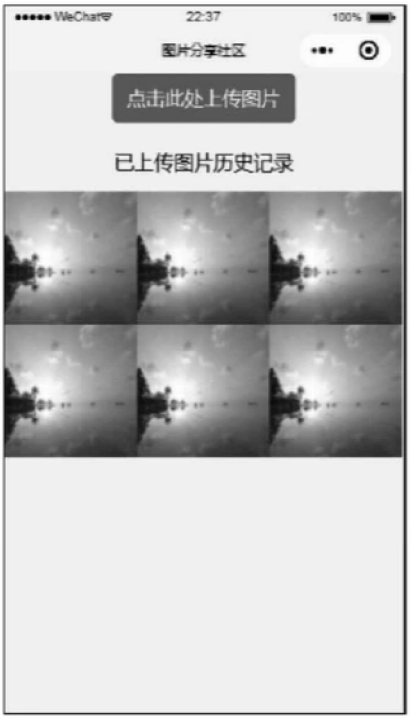


图 19-13　上传图片页效果图

19.3　逻辑实现

19.3.1　用户个人信息获取逻辑

在首页中点击“上传图片”按钮时需要实现两个功能，一是获取当前用户个人信息(包括基础信息和 openid)；二是能够跳转到上传图片页。

1 获取用户基础信息



视频讲解

用户基础信息主要包括用户的微信昵称、头像、性别、所在地等内容，为按钮组件(<button>)添加 open-type 属性就可以用来获取这些信息。

修改 index.wxml 文件中浮动按钮的代码，为其添加点击后获取用户个人信息的功能。

相关 WXML(pages/index/index.wxml)代码片段如下：

```
1.  <!--“上传图片”按钮 -->
2.  <view class = 'floatBtn'>
3.    <button size = 'mini' type = 'primary' open - type = 'getUserInfo' bindgetuserinfo =
'getUserInfo'>上传图片</button>
4.  </view>
```

其中 open-type 的属性值为固定写法，用于表示按钮的特殊用法类型；bindgetuserinfo 的属性值为自定义函数，用于获取到用户信息后的进一步处理，开发者也可以更改为其他名称。

在 JS 文件中添加自定义函数 getUserInfo，相关 JS(pages/index/index.js)代码片段如下：

```
1.  var app = getApp()
2.
3.  Page({
4.    /**
5.      * 自定义函数 -- 获取用户个人信息
6.      */
7.    getUserInfo: function(e) {
8.      //尝试打印输出个人信息，测试是否获取成功
9.      console.log(e.detail.userInfo)
10.     //将用户个人信息存放到全局变量 userInfo 中
11.     app.globalData.userInfo = e.detail.userInfo
12.   },
13. })
```

点击右上角的按钮触发获取用户信息事件，此时 Console 控制台的运行效果如图 19-14 所示。

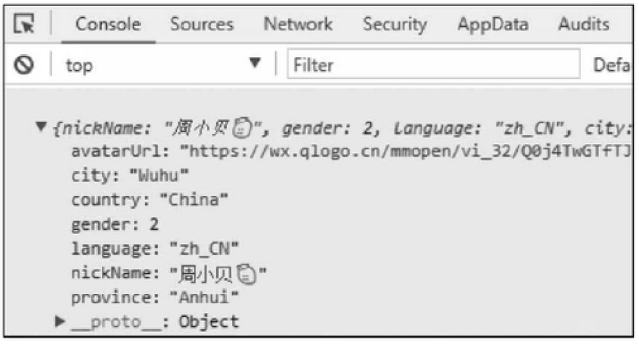


图 19-14 Console 控制台显示用户基础信息



视频讲解

由图可见，点击按钮后可以将用户的信息打印出来了，说明该功能生效。此时可以去掉或者注释掉其中的 console.log(e.detail.userInfo)语句，该语句仅在开发过程中临时使用，用于测试是否获取到数据。

2 获取用户 openid 信息

实际上每个用户还有一个专属用户编号，称为 openid，该编号信息是不会

改变的，因此开发者可以通过 openid 的值来确定用户身份。

可以通过自定义云函数来实现用户 openid 信息的获取。对项目中的 cloudfunctions 右击，选择“新建 Node.js 云函数”，然后自定义云函数名称，例如 getOpenid。

修改云函数中 index.js 文件的代码如下：

```
1.  //云函数入口文件
2.  const cloud = require('wx-server-sdk')
3.
4.  cloud.init()
5.
6.  //云函数入口函数
7.  exports.main = async (event, context) => {
8.    const wxContext = cloud.getWXContext()
9.
10.   return {
11.     openid: wxContext.OPENID
12.   }
13. }
```

修改完毕后右击该云函数，选择“上传并部署：云端安装依赖”等待上传到云开发控制台。上传成功后可以在云开发控制台查看，如图 19-15 所示。

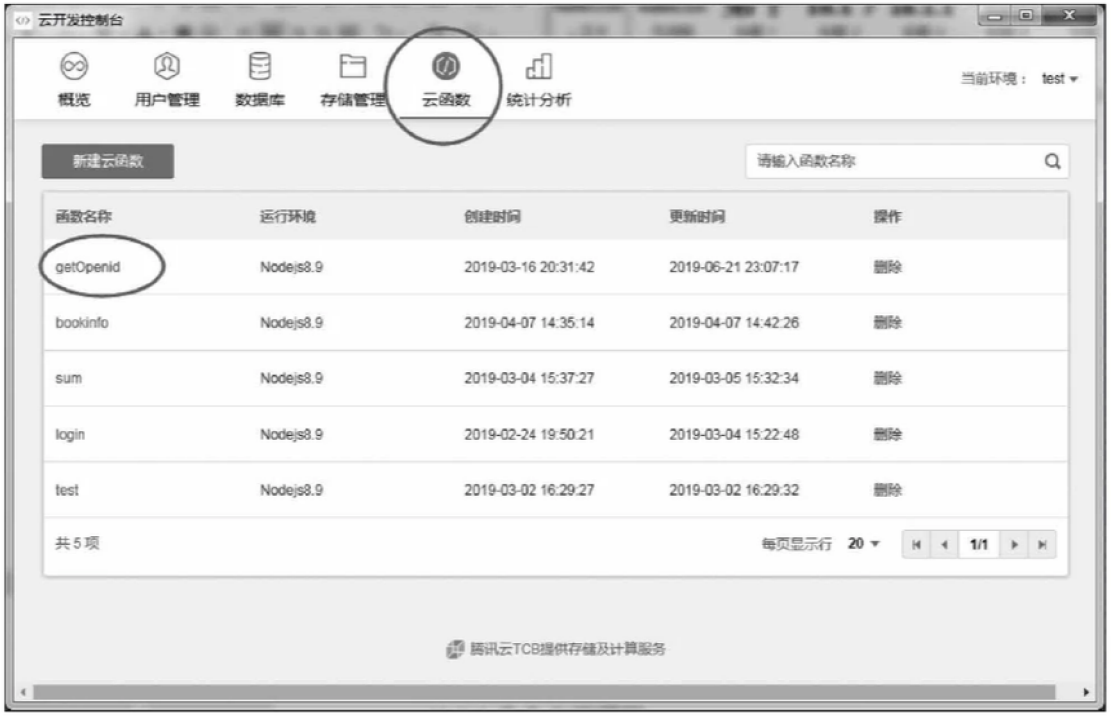


图 19-15　云开发控制台显示云函数信息

现在就可以在小程序中调用该云函数获取当前用户的 openid 信息了。修改 index.js 中的自定义函数 getUserInfo，相关 JS(pages/index/index.js)代码片段如下：

```
1.  Page({
2.    /**
3.     * 自定义函数 -- 获取用户个人信息
4.     */
5.    getUserInfo: function(e) {
```

```
      //尝试打印输出个人信息,测试是否获取成功
      //console.log(e.detail.userInfo)
      //将用户个人信息存放到全局变量 userInfo 中
      app.globalData.userInfo = e.detail.userInfo

      //检测是否已经获取过了用户 openid 信息
      if (app.globalData.openid == null) {
        //如果是第一次登录则使用云函数获取用户 openid
        wx.cloud.callFunction({
          name: 'getOpenid',
          complete: res => {
            //尝试打印输出个人信息,测试是否获取成功
            console.log(res.result.openid)
            //将用户 openid 信息存放到全局变量 openid 中
            app.globalData.openid = res.result.openid
          }
        })
      }
    },
  })
```

点击右上角的按钮触发获取用户信息事件，此时 Console 控制台的运行效果如图 19-16 所示。



图 19-16　Console 控制台显示用户 openid 信息

由该图可见，点击按钮后可以将用户的 openid 信息打印出来了，说明该功能生效。此时可以去掉或者注释掉其中的 console.log(res.result.openid)语句，该语句仅在开发过程中临时使用，用于测试是否获取到数据。

3 首页跳转上传图片页



视频讲解

首页中的“上传图片”按钮点击后除了能够获取用户个人信息以外，还需要跳转到上传图片页，以便进行本地图片的上传。

修改 index.wxml 文件中浮动按钮的代码，为其添加自定义点击事件 goToAdd。

相关 WXML(pages/index/index.wxml)代码片段如下：

```
<!--“上传图片”按钮 -->
<view class='floatBtn'>
    <button size='mini' type='primary' bindtap='goToAdd' open-type='getUserInfo' bindgetuserinfo='getUserInfo'>上传图片</button>
</view>
```

在 JS 文件中添加自定义函数 goToAdd，相关 JS(pages/index/index.js)代码片段如下：

```
Page({
  /**
   * 自定义函数 -- 跳转上传图片页
   */
```

```
5.    goToAdd: function(options) {
6.      wx.navigateTo({
7.        url: '../add/add',
8.      })
9.    },
10. })
```

为了测试跳转后新页面是否也可以读取到用户个人信息数据，可以临时在 add.js 文件的 onLoad 函数中添加 console.log()语句尝试打印输出用户的基础信息和 openid。

相关 JS(pages/add/add.js)代码片段如下：

```
1.  var app = getApp()
2.  Page({
3.    /**
4.     * 生命周期函数--监听页面加载
5.     */
6.    onLoad: function(options) {
7.      //尝试输出用户基础信息
8.      console.log(app.globalData.userInfo)
9.      //尝试输出用户 openid
10.     console.log(app.globalData.openid)
11.   },
12. })
```

点击首页右上角的按钮跳转新页面，此时 Console 控制台的运行效果如图 19-17 所示。

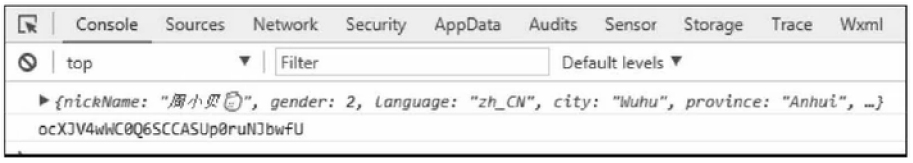


图 19-17　Console 控制台显示用户个人信息

由图可见，点击按钮跳转上传图片页后可以将用户基础信息和 openid 都打印出来了，说明该功能生效。此时可以去掉或者注释掉这两个 console.log()语句，该语句仅在开发过程中临时使用，用于测试是否获取到数据。

19.3.2　上传图片页逻辑

为了使首页可以显示真实上传的图片素材，可以先将上传图片页的功能实现，然后传若干张测试图片等待首页展示使用。

1 图片上传



视频讲解

图片上传成功后需要用到云数据库中的 photos 数据集记录图片基础信息，例如图片地址、添加日期、上传者信息等。因此首先在 add.js 文件顶端声明对 photos 数据集的引用，相关 JS(pages/index/index.js)代码片段如下：

```
1.  const db = wx.cloud.database()
2.  const photos = db.collection('photos')
3.
4.  Page({
5.      ...
6.  })
```

其中 const photos 是自定义名称，开发者可以自行更改。

上传图片成功后还需要记录上传日期，因此在 add.js 文件的顶端自定义函数 formatDate，用于格式化显示当天的日期。相关 JS(pages/index/index.js)代码片段如下：

```
1.  //格式化当前日期
2.  function formatDate() {
3.    var now = new Date()
4.    var year = now.getFullYear()
5.    var month = now.getMonth() + 1
6.    var day = now.getDate()
7.
8.    if (month < 10) month = '0' + month
9.    if (day < 10) day = '0' + day
10.
11.   return year + '-' + month + '-' + day
12. }
13.
14. Page({
15.     ...
16. })
```

修改 add.wxml 中按钮的代码，为其添加自定义点击事件 upload，用于上传图片。

相关 WXML(pages/add/add.wxml)代码片段修改后如下：

```
1.    <!--“点击此处上传图片”按钮 -->
2.    <button type = 'warn' bindtap = 'upload'>点击此处上传图片</button>
```

然后在 add.js 中添加自定义函数 upload，相关 JS(pages/add/add.js)代码片段如下：

```
1.  Page({
2.    /**
3.     * 自定义函数 -- 上传图片
4.     */
5.    upload: function() {
6.      //选择图片
7.      wx.chooseImage({
8.        count: 1,
9.        sizeType: ['compressed'],
10.       sourceType: ['album', 'camera'],
11.       success: function(res) {
12.         //loading 提示框表示正在上传图片
13.         wx.showLoading({
14.           title: '上传中',
15.         })
16.         //获取图片临时地址
17.         const filePath = res.tempFilePaths[0]
18.
19.         //自定义云端的图片名称
20.         const cloudPath = Math.floor(Math.random() * 1000000) + filePath.match(/.[^.]
+?$/)[0]
21.         //上传图片到云存储空间中
22.         wx.cloud.uploadFile({
23.           cloudPath,
24.           filePath,
25.           success: res => {
```

```
            //提示上传成功
            wx.showToast({
              title: '上传成功!',
              duration: 3000
            })

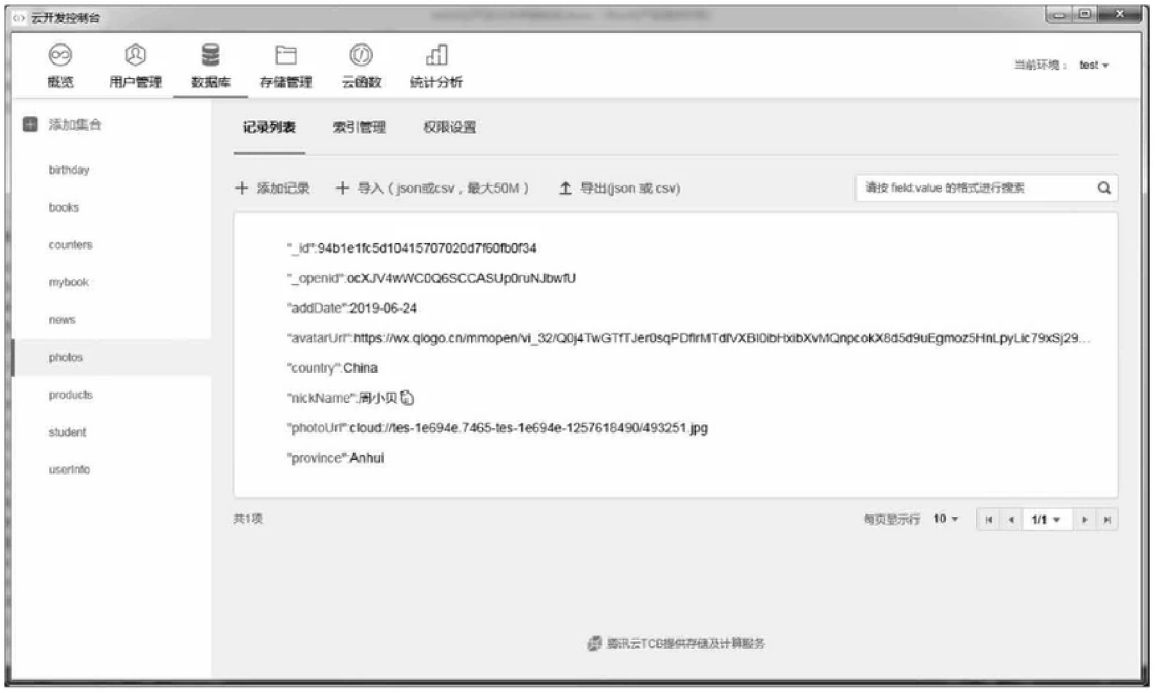
            //获取用户个人基础信息
            let userInfo = app.globalData.userInfo
            //获取当天日期
            let today = formatDate()

            //往云数据集中添加一条记录
            photos.add({
              data: {
                photoUrl: res.fileID,
                avatarUrl: userInfo.avatarUrl,
                country: userInfo.country,
                province: userInfo.province,
                nickName: userInfo.nickName,
                addDate: today
              },
              success: res => {
                console.log(res)
              },
              fail: e => {
                console.log(e)
              }
            })
          },
          fail: e => {
            //提示上传失败
            wx.showToast({
              icon: 'none',
              title: '上传失败',
            })
          }
        })
      },
      fail: e => {
        console.error(e)
      },
      complete: () => {
        //上传完成后关闭 loading 提示框
        wx.hideLoading()
      }
    })
  },
})
```

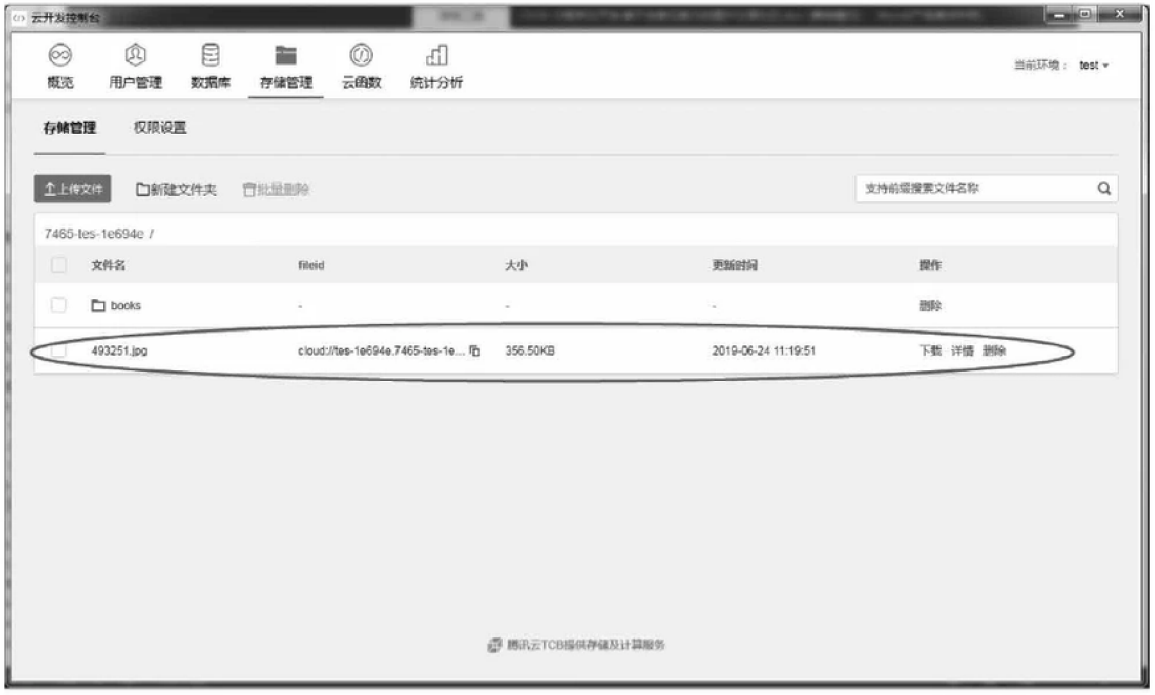
上述代码在自定义云端图片名称时用了随机数 Math.floor(Math.random() * 1000000),表示图片名称为随机生成的 0～1 000 000 的向下取整的数,但是在图片非常多的情况下难免会碰巧有重复的,开发者也可以思考其他自定义的图片名称。

此外还需要注意的是,一定要从首页中点击“上传图片”按钮跳转到当前页面再进行测试,否则可能无法获取到用户的个人信息数据。

点击首页右上角的按钮跳转到当前页面，并从本地任意上传一张图片。此时云开发控制台的运行效果如图 19-18 所示。



(a) 云数据库的photos集中出现记录



(b) 云存储中出现已上传的图片

图 19-18　云开发控制台显示信息

由图可见，本地图片上传成功后会分别出现在云数据库和云存储中。为了后续的图片展示功能，不妨再继续上传若干张本地图片进行测试。



视频讲解

2 展示图片历史记录

在往云数据库中添加记录时，会自动登记当前用户的 openid 并记录成“_openid”字段，因此只要筛选出 openid 属性值与当前用户匹配的所有数据即

可展示图片历史记录。

在 add.js 文件中添加自定义函数 getHistoryPhotos，用于获取当前用户已经上传过的图片数据。相关 JS(pages/add/add.js)代码片段如下：

```
Page({
  /**
   * 自定义函数 -- 获取已上传图片历史记录
   */
  getHistoryPhotos: function() {
    //获取当前用户的 openid
    let openid = app.globalData.openid

    //从云数据集查找当前用户的上传记录
    photos.where({
      _openid:openid
    }).get({
      success:res =>{
        this.setData({historyPhotos:res.data})
      }
    })
  },
})
```

上述代码表示从云数据集 photos 中查找当前用户上传的图片记录，并将获取到的记录集存到变量 historyPhotos 中，该名称可以由开发者自定义。

修改 add.wxml 文件中显示图片历史记录的代码，使用 wx:for 循环来展示实际图片。

相关 WXML(pages/add/add.wxml)代码片段如下：

```
<!-- 图片集 -->
<view>
  <block wx:for = '{{historyPhotos}}' wx:key = 'history{{index}}'>
    <image src = '{{item.photoUrl}}'></image>
  </block>
</view>
```

最后在 add.js 文件的 onLoad 和 upload 函数中分别引用 getHistoryPhotos，表示打开当前页面和上传新图片后都更新显示图片历史记录。

相关 JS(pages/add/add.js)代码片段如下：

```
Page({
  /**
   * 自定义函数 -- 上传图片
   */
  upload: function() {
    //选择图片
    wx.chooseImage({
      count: 1,
      sizeType: ['compressed'],
      sourceType: ['album', 'camera'],
      success: function(res) {
        ...
      },
      fail: e => {
        console.error(e)
      },
```

```
      complete: () => {
        //上传完成后关闭 loading 提示框
        wx.hideLoading()
        //更新图片历史记录
        this.getHistoryPhotos()
      }
    })
  },

  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    //更新图片历史记录
    this.getHistoryPhotos()
  },

})
```



图 19-19　Console 控制台显示用户个人信息

在已经有上传图片的前提条件下打开上传图片页，此时运行效果如图 19-19 所示。

由图可见，当前页面可以成功显示云存储中的图片，此时上传图片页的逻辑全部完成。

19.3.3　首页逻辑



视频讲解

1 图片列表展示

首先在 index.js 文件顶端声明对 photos 数据集的引用，相关 JS(pages/index/index.js)代码片段如下：

```
const db = wx.cloud.database()
const photos = db.collection('photos')

Page({
    ...
})
```

其中 const photos 是自定义名称，开发者可以自行更改。

在 index.js 文件的 onShow 函数中追加对云端数据库的读取代码，这样从其他页面返回首页时也可以刷新图片列表。相关 JS(pages/index/index.js)代码片段如下：

```
Page({
  /**
   * 生命周期函数--监听页面显示
   */
  onShow: function() {
    //获取图片列表(按照添加日期降序排列)
    photos.orderBy('addDate','desc').get({
      success: res => {
        this.setData({
          photoList: res.data
        })
      }
```

```
13.      })
14.    },
15.  })
```

修改 index.wxml 文件中卡片的相关代码，使用 wx:for 循环语句显示实际图片信息。相关 WXML(pages/index/index.wxml)代码片段如下：

```
1.    <!-- 卡片 -->
2.    <view class = 'card' wx:for = '{{photoList}}' wx:key = 'photo{{index}}'>
3.      <!-- 卡片页眉 -->
4.      <view class = 'card - head'>
5.        <image class = 'avatar' src = '{{item.avatarUrl}}' mode = 'widthFix'></image>
6.        <view class = 'title'>
7.          <text class = 'nickName'>{{item.nickName}}</text>
8.          <text class = 'address'>{{item.province}},{{item.country}}</text>
9.        </view>
10.     </view>
11.
12.     <!-- 卡片主体 -->
13.     <view class = 'card - body'>
14.       <image src = '{{item.photoUrl}}' mode = 'widthFix'></image>
15.     </view>
16.
17.     <!-- 卡片页脚 -->
18.     <view class = 'card - foot'>
19.       <text>{{item.addDate}}</text>
20.     </view>
21.   </view>
22. </view>
```

为了显示出上传日期的区别，开发者可以临时手动修改云数据集里面 addDate 字段的值，此时运行效果如图 19-20 所示。

由图可见，首页已经可以成功显示当前云存储中的图片和上传者信息，并且按照上传日期将最新的图片显示在最上面。



视频讲解

2 点击头像跳转个人主页

修改 index.wxml 文件中卡片页眉的相关代码，使用< navigator >组件实现头像图片的导航功能。相关 WXML(pages/index/index.wxml)代码片段如下：

```
1.      <!-- 卡片页眉 -->
2.      <view class = 'card - head'>
3.        <navigator url = '../homepage/homepage?id = {{item._openid}}'>
4.          <image class = 'avatar' src = '{{item.avatarUrl}}' mode = 'widthFix'></image>
5.        </navigator>
6.        <view class = 'title'>
7.          <text class = 'nickName'>{{item.nickName}}</text>
8.          <text class = 'address'>{{item.province}},{{item.country}}</text>
9.        </view>
10.     </view>
```

上述代码表示点击头像图片可以跳转到 homepage 页面。由于未来实际使用该小程序时有可能首页会呈现多名不同用户的头像信息，所以在< navigator >组件的 url 属性值中携带自



(a) 页面初始效果　　(b) 下拉滚动效果

图 19-20　首页图片列表展示

定义参数 id，用于记录当前被点击用户的 openid。

后续的代码逻辑需要在个人主页(homepage)中实现。

3 点击图片跳转图片展示页

修改 index.wxml 文件中卡片主体的相关代码，使用< navigator >组件实现上传图片的导航功能。相关 WXML(pages/index/index.wxml)代码片段如下：



视频讲解

```
1.    <!-- 卡片主体 -->
2.    < view class = 'card - body'>
3.      < navigator url = '../detail/detail?id = {{item._id}}'>
4.        < image src = '{{item.photoUrl}}' mode = 'widthFix'></image >
5.      </navigator >
6.    </view >
```

上述代码表示点击展示图片可以跳转到 detail 页面。由于首页会呈现不同图片信息，所以在< navigator >组件的 url 属性值中携带自定义参数 id，用于记录当前被点击图片的 id。

后续的代码逻辑需要在图片展示页(detail)中实现。

19.3.4　个人主页逻辑



视频讲解

1 获取被点击用户上传的图片列表

从首页跳转到个人主页时会携带被点击用户的 openid 值，所以可以去云数据集查找该用户上传的所有图片记录。

首先在 homepage.js 文件顶端声明对云数据集 photos 的引用，相关 JS

(pages/homepage/homepage.js)代码片段如下：

```
const db = wx.cloud.database()
const photos = db.collection('photos')

Page({
    ...
})
```

其中 const photos 是自定义名称，开发者可以自行更改。

然后在 homepage.js 文件的 onLoad 函数中追加对云数据集 photos 的读取代码。

相关 JS(pages/index/index.js)代码片段如下：

```
Page({
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    //获取被点击用户的 openid
    let openid = options.id

    //显示 loading 提示框
    wx.showLoading({
      title: '数据加载中',
    })

    //查找云数据集 photos 中该用户的图片上传记录
    photos.orderBy('addDate', 'desc').where({
      _openid: openid
    }).get({
      success: res => {
        //获取数据记录
        this.setData({
          photoList: res.data
        })
        //关闭 loading 提示框
        wx.hideLoading()
      }
    })
  },
})
```

最后修改 homepage.wxml 中卡片的代码，使用 wx:for 循环语句显示实际图片信息。

相关 WXML(pages/homepage/homepage.wxml)代码片段如下：

```
<!--卡片-->
<view class='card' wx:for='{{photoList}}' wx:key='photo{{index}}'>
  <!--卡片页眉-->
  <view class='card-head'>
    <image class='avatar' src='{{item.avatarUrl}}' mode='widthFix'></image>
    <view class='title'>
      <text class='nickName'>{{item.nickName}}</text>
      <text class='address'>{{item.province}},{{item.country}}</text>
    </view>
  </view>

  <!--卡片主体-->
```

```
13.      <view class = 'card - body'>
14.        <image src = '{{item.photoUrl}}' mode = 'widthFix'></image>
15.      </view>
16.
17.      <!-- 卡片页脚 -->
18.      <view class = 'card - foot'>
19.        <text>{{item.addDate}}</text>
20.      </view>
21.    </view>
```

此时运行效果如图 19-21 所示。



(a) 页面初始效果　　(b) 下拉滚动效果

图 19-21　个人主页的图片列表展示

由图可见，个人主页已经可以成功显示云存储中的图片，并且按照上传日期将最新的图片显示在最上面。

由于尚未实现顶端用户头像和昵称更新的逻辑，所以目前仍然是测试图片效果。



视频讲解

2 显示顶端用户头像和昵称

修改 homepage.wxml 文件中顶端头像和昵称的代码，使用 photoList 数据中的任意一条均可获取头像和昵称属性值。相关 WXML（pages/homepage/homepage.wxml）代码片段如下：

```
1.    <!-- 头像和昵称 -->
2.    <view class = 'avatarBox'>
3.      <image src = '{{photoList[0].avatarUrl}}'></image>
4.      <text>{{photoList[0].nickName}}个人主页</text>
5.    </view>
```

此时运行效果如图 19-22 所示。

由图可见，此时已经可以成功显示当前用户的个人信息了。



图 19-22　个人主页的顶端头像和昵称展示

3 点击图片跳转图片展示页



视频讲解

修改 homepage.wxml 文件中卡片主体的相关代码，使用< navigator >组件实现上传图片的导航功能。相关 WXML(pages/homepage/homepage.wxml)代码片段如下：

```
1.      <!-- 卡片主体 -->
2.      < view class = 'card - body'>
3.          < navigator url = '../detail/detail?id = {{item._
id}}'>
4.              < image src = '{{item.photoUrl}}' mode =
'widthFix'></image >
5.          </navigator >
6.      </view >
```

上述代码表示点击展示图片可以跳转到 detail 页面。由于首页会呈现不同图片信息，所以在< navigator >组件的 url 属性值中携带自定义参数 id，用于记录当前被点击图片的 id。

后续的代码逻辑需要在图片展示页(detail)中实现。

19.3.5 图片展示页逻辑



视频讲解

1 显示当前图片

从首页或个人主页的图片卡片跳转到图片展示页时均会携带被点击图片的 id，所以可以去云数据集查找该图片的记录。

首先在 detail.js 文件顶端声明对云数据集 photos 的引用，相关 JS(pages/detail/detail.js)代码片段如下：

```
1.  const db = wx.cloud.database()
2.  const photos = db.collection('photos')
3.
4.  Page({
5.      ...
6.  })
```

其中 const photos 是自定义名称，开发者可以自行更改。

然后在 detail.js 文件的 onLoad 函数中追加对云数据集 photos 的读取代码。

相关 JS(pages/detail/detail.js)代码片段如下：

```
1.  Page({
2.    /**
3.     * 生命周期函数 -- 监听页面加载
4.     */
5.    onLoad: function(options) {
6.      //根据图片 id 获取云数据集中的图片记录
7.      photos.doc(options.id).get({
8.        success: res => {
```

```
9.            this.setData({ photo: res.data })
10.         }
11.       })
12.     },
13. })
```

最后修改 detail.wxml 中图片的代码，显示实际图片信息。

相关 WXML(pages/detail/detail.wxml)代码片段如下：

```
1.  <!-- 顶端图片展示 -->
2.  <image src = '{{photo.photoUrl}}' mode = 'widthFix'></image>
```

从首页点击任意图片，此时运行效果如图 19-23 所示。



(a) 首页效果　　(b) 图片展示页效果

图 19-23　图片展示页显示顶端图片

由图可见，此时点击首页卡片列表中的任意图片均可在图片展示页显示正确的画面。实际上从个人主页点击图片进入图片展示页也是一样的效果，这里不再另行演示。

2 下载图片到本地设备



视频讲解

修改 detail.wxml 中相关按钮的代码，为其添加点击事件 downloadPhoto。

相关 WXML(pages/detail/detail.wxml)代码片段如下：

```
1.  <!-- 按钮区域 -->
2.  <view>
3.     <button type = 'primary' plain bindtap = 'downloadPhoto'>下载到本地</
button>
4.     <button type = 'primary' plain>分享给好友</button>
5.     <button type = 'primary' plain>全屏预览</button>
6.  </view>
```

然后在 detail.js 文件中追加自定义函数 downloadPhoto，相关 JS(pages/detail/detail.js)

代码片段如下：

```
Page({
  /**
   * 自定义函数 -- 下载图片到本地
   */
  downloadPhoto: function() {
    //从云存储中进行图片下载
    wx.cloud.downloadFile({
      fileID: this.data.photo.photoUrl,
      success: res => {
        //保存图片到本地相册
        wx.saveImageToPhotosAlbum({
          filePath: res.tempFilePath,
          success: res => {
            wx.showToast({
              title: '保存成功!',
            })
          },
          fail: err => {
            wx.showToast({
              title: '保存失败!',
              icon: 'none'
            })
          }
        })
      },
      fail: err => {
        console.log(err)
      }
    })
  },
})
```

此时运行效果如图 19-24 所示。







(a) 点击按钮进行图片下载



(b) 保存成功提示

图 19-24　下载图片到本地设备效果

当前是计算机端模拟运行效果，可以自选图片另存为地址；如果用手机测试会将图片直接保存到手机相册中。

3 分享图片给好友



视频讲解

修改 detail.wxml 中相关按钮的代码，为其添加 open-type 属性。

相关 WXML(pages/detail/detail.wxml)代码片段如下：

```
1.  <!-- 按钮区域 -->
2.  <view>
3.    <button type = 'primary' plain bindtap = 'downloadPhoto'>下载到本地</button>
4.    <button type = 'primary' plain open - type = 'share'>分享给好友</button>
5.    <button type = 'primary' plain>全屏预览</button>
6.  </view>
```

此时点击“分享给好友”按钮的效果就等同于点击右上角的“…”，都可以进行页面分享。

如果用户需要自定义分享的内容，可以在 detail.js 文件中修改自带的 onShareAppMessage 函数，相关 JS(pages/detail/detail.js)代码片段如下：

```
1.  Page({
2.    /**
3.     * 用户点击右上角分享
4.     */
5.    onShareAppMessage: function() {
6.      return {
7.        title: '给你分享一张好看的图片',
8.        path: 'pages/detail/detail?id = ' + this.data.photo._id
9.      }
10.   }
11. })
```

此时运行效果如图 19-25 所示。



(a) 点击按钮进行图片分享　　(b) 图片分享对话框

图 19-25　分享图片效果

当前是计算机端模拟运行效果，如果用手机测试还可以选择微信好友列表进行发送。

4 全屏预览图片



视频讲解

修改 detail.wxml 中相关按钮的代码，为其添加点击事件 previewPhoto。相关 WXML(pages/detail/detail.wxml)代码片段如下：

```
1.  <!-- 按钮区域 -->
2.  <view>
3.    <button type = 'primary' plain bindtap = 'downloadPhoto'>下载到本地</button>
4.    <button type = 'primary' plain open - type = 'share'>分享给好友</button>
5.      <button type = 'primary' plain bindtap = 'previewPhoto'>全屏预览</button>
6.  </view>
```

然后在 detail.js 文件中追加自定义函数 previewPhoto，相关 JS(pages/detail/detail.js)代码片段如下：

```
1.  Page({
2.    /**
3.     * 自定义函数 -- 全屏预览图片
4.     */
5.    previewPhoto: function() {
6.      //定义图片 URL 地址
7.      var urls = [this.data.photo.photoUrl]
8.      //全屏预览图片
9.      wx.previewImage({
10.       urls: urls,
11.     })
12.   },
13. })
```

此时运行效果如图 19-26 所示。



(a) 点击按钮进行图片全屏预览　　(b) 全屏预览效果

图 19-26　全屏预览图片效果

当前是计算机端模拟运行效果，如果用手机预览，长按还可以保存图片到手机相册中。

19.4 完整代码展示

19.4.1 应用文件代码展示

app.json 文件的完整代码如下：

```
{
  "pages": [
    "pages/index/index",
    "pages/homepage/homepage",
    "pages/detail/detail",
    "pages/add/add"
  ],
  "window": {
    "navigationBarBackgroundColor": "#F6F6F6",
    "navigationBarTitleText": "图片分享社区",
    "navigationBarTextStyle": "black"
  }
}
```

app.wxss 文件的完整代码如下：

```
/*1.整体容器样式*/
.container {
  display: flex;
  flex-direction: column;
  align-items: center;
  background-color: whitesmoke;
  min-height: 100vh;
}

/*2.卡片区域整体样式*/
.card {
  width: 710rpx;
  border-radius: 20rpx;
  background-color: white;
  margin: 20rpx 0;
}
/*2-1 卡片页眉*/
.card-head {
  display: flex;
  flex-direction: row;
  align-items: center;
}

/*2-1-1 头像图片*/
.avatar {
  width: 150rpx;
  border-radius: 50%;
  margin: 0 20rpx;
}

```

```
/*2-1-2 文字信息*/
.title {
  display: flex;
  flex-direction: column;
}
/*2-1-2(1) 昵称*/
.nickName {
  font-size: 35rpx;
  margin: 0 10rpx;
}
/*2-1-2(2) 所在地区*/
.address {
  font-size: 30rpx;
  color: gray;
  margin: 0 10rpx;
}
/*2-2 卡片主体*/
.card-body image {
  width: 100%;
}
/*2-3 卡片页脚*/
.card-foot{
  font-size: 35rpx;
  text-align: right;
  margin: 20rpx;
}
```

app.js 文件的完整代码如下(该文件代码为云模板自动生成)：

```
App({
  onLaunch: function() {

    if (!wx.cloud) {
      console.error('请使用 2.2.3 或以上的基础库以使用云能力')
    } else {
      wx.cloud.init({
        traceUser: true,
      })
    }

    this.globalData = {}
  }
})
```

19.4.2 云函数文件代码展示

云函数 getOpenid 的 JS 文件(getOpenid/index.js)的完整代码如下：

```
//云函数入口文件
const cloud = require('wx-server-sdk')

cloud.init()

//云函数入口函数
exports.main = async (event, context) => {
```

```
    const wxContext = cloud.getWXContext()

    return {
      openid: wxContext.OPENID
    }
}
```

19.4.3 页面文件代码展示

1 首页代码展示

WXML 文件(pages/index/index.wxml)的完整代码如下：

```
<view class = 'container'>
  <!-- 卡片 -->
  <view class = 'card' wx:for = '{{photoList}}' wx:key = 'photo{{index}}'>
    <!-- 卡片页眉 -->
    <view class = 'card - head'>
      <navigator url = '../homepage/homepage?id = {{item._openid}}'>
        <image class = 'avatar' src = '{{item.avatarUrl}}' mode = 'widthFix'></image>
      </navigator>
      <view class = 'title'>
        <text class = 'nickName'>{{item.nickName}}</text>
        <text class = 'address'>{{item.province}},{{item.country}}</text>
      </view>
    </view>

    <!-- 卡片主体 -->
    <view class = 'card - body'>
      <navigator url = '../detail/detail?id = {{item._id}}'>
        <image src = '{{item.photoUrl}}' mode = 'widthFix'></image>
      </navigator>
    </view>

    <!-- 卡片页脚 -->
    <view class = 'card - foot'>
      <text>{{item.addDate}}</text>
    </view>
  </view>
</view>

<!-- "上传图片"按钮 -->
<view class = 'floatBtn'>
   <button size = 'mini' type = 'primary' bindtap = 'goToAdd' open - type = 'getUserInfo'
bindgetuserinfo = 'getUserInfo'>上传图片</button>
</view>
```

WXSS 文件(pages/index/index.wxss)的完整代码如下：

```
/* 浮动按钮样式 */
.floatBtn{
  position: fixed;
  top: 50rpx;
  right: 50rpx;
}
```

JS 文件(pages/index/index.js)的完整代码如下：

```
const db = wx.cloud.database()
const photos = db.collection('photos')
var app = getApp()

Page({
  /**
   * 自定义函数 -- 跳转上传图片页
   */
  goToAdd: function(options) {
    wx.navigateTo({
      url: '../add/add',
    })
  },

  /**
   * 自定义函数 -- 获取用户个人信息
   */
  getUserInfo: function(e) {
    //尝试打印输出个人信息,测试是否获取成功
    //console.log(e.detail.userInfo)

    //将用户个人信息存放到全局变量 userInfo 中
    app.globalData.userInfo = e.detail.userInfo

    //检测是否已经获取过了用户 openid 信息
    if (app.globalData.openid == null) {
      //如果是第一次登录,则使用云函数获取用户 openid
      wx.cloud.callFunction({
        name: 'getOpenid',
        complete: res => {
          //尝试打印输出个人信息,测试是否获取成功
          //console.log(res.result.openid)
          //将用户 openid 信息存放到全局变量 openid 中
          app.globalData.openid = res.result.openid
        }
      })
    }
  },
  /**
   * 生命周期函数 -- 监听页面显示
   */
  onShow: function() {
    //获取图片列表(按照添加日期降序排列)
    photos.orderBy('addDate','desc').get({
      success: res => {
        this.setData({
          photoList: res.data
        })
      }
    })
  }
```

2 个人主页代码展示

WXML 文件(pages/homepage/homepage.wxml)的完整代码如下：

```
<view class='container'>
  <!-- 头像和昵称 -->
  <view class='avatarBox'>
    <image src='{{photoList[0].avatarUrl}}'></image>
    <text>{{photoList[0].nickName}}个人主页</text>
  </view>

  <!-- 卡片 -->
  <view class='card' wx:for='{{photoList}}' wx:key='photo{{index}}'>
    <!-- 卡片页眉 -->
    <view class='card-head'>
      <navigator url='../user/user?id={{item._openid}}'>
        <image class='avatar' src='{{item.avatarUrl}}' mode='widthFix'></image>
      </navigator>
      <view class='title'>
        <text class='nickName'>{{item.nickName}}</text>
        <text class='address'>{{item.province}},{{item.country}}</text>
      </view>
    </view>

    <!-- 卡片主体 -->
    <view class='card-body'>
      <navigator url='../detail/detail?id={{item._id}}'>
        <image src='{{item.photoUrl}}' mode='widthFix'></image>
      </navigator>
    </view>

    <!-- 卡片页脚 -->
    <view class='card-foot'>
      <text>{{item.addDate}}</text>
    </view>
  </view>
</view>
```

WXSS 文件(pages/homepage/homepage.wxss)的完整代码如下：

```
/*头像区域*/
.avatarBox{
  display: flex;
  flex-direction: column;
  align-items: center;
  padding-top: 100rpx;
}
/*头像区域中的头像图片*/
.avatarBox image{
  width: 200rpx;
  height: 200rpx;
  border-radius: 50%;
}
/*头像区域中的昵称文本*/
.avatarBox text{
  margin: 30rpx 0;
}
```

JS 文件(pages/homepage/homepage.js)的完整代码如下：

```
const db = wx.cloud.database()
const photos = db.collection('photos')

Page({
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    //获取被点击用户的 openid
    let openid = options.id

    //显示 loading 提示框
    wx.showLoading({
      title: '数据加载中',
    })

    //查找云数据集 photos 中该用户的图片上传记录
    photos.orderBy('addDate', 'desc').where({
      _openid: openid
    }).get({
      success: res => {
        //获取数据记录
        this.setData({
          photoList: res.data
        })
        //关闭 loading 提示框
        wx.hideLoading()
      }
    })
  }
})
```

3 图片展示页代码展示

WXML 文件(pages/detail/detail.wxml)的完整代码如下：

```
<!-- 顶端图片展示 -->
<image src='{{photo.photoUrl}}' mode='widthFix'></image>

<!-- 按钮区域 -->
<view>
  <button type='primary' plain bindtap='downloadPhoto'>下载到本地</button>
  <button type='primary' plain open-type='share'>分享给好友</button>
    <button type='primary' plain bindtap='previewPhoto'>全屏预览</button>
</view>
```

WXSS 文件(pages/detail/detail.wxss)的完整代码如下：

```
/* 顶端展示图片样式 */
image{
  width: 750rpx;
}
/* 按钮外层 view */
view{
  padding: 0 50rpx;
```

```
}
/*按钮样式*/
button{
  margin: 10rpx;
}
```

JS 文件(pages/detail/detail.js)的完整代码如下：

```
const db = wx.cloud.database()
const photos = db.collection('photos')

Page({
  /**
   * 自定义函数--下载图片到本地
   */
  downloadPhoto: function() {
    //从云存储中进行图片下载
    wx.cloud.downloadFile({
      fileID: this.data.photo.photoUrl,
      success: res => {
        //保存图片到本地相册
        wx.saveImageToPhotosAlbum({
          filePath: res.tempFilePath,
          success: res => {
            wx.showToast({
              title: '保存成功!',
            })
          },
          fail: err => {
            wx.showToast({
              title: '保存失败!',
              icon: 'none'
            })
          }
        })
      },
      fail: err => {
        console.log(err)
      }
    })
  },
  /**
   * 自定义函数--全屏预览图片
   */
  previewPhoto: function() {
    //定义图片URL地址
    var urls = [this.data.photo.photoUrl]
    //全屏预览图片
    wx.previewImage({
      urls: urls,
    })
  },
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
```

```
    //根据图片 id 获取云数据集中的图片记录
    photos.doc(options.id).get({
      success: res => {
        this.setData({ photo: res.data })
      }
    })
  },
  /**
   * 用户点击右上角分享
   */
  onShareAppMessage: function() {
    return {
      title: '给你分享一张好看的图片',
      path: 'pages/detail/detail?id=' + this.data.photo._id
    }
  }
})
```

4 上传图片页代码展示

WXML 文件(pages/add/add.wxml)的完整代码如下：

```
<view class='container'>
  <!--“点击此处上传图片”按钮 -->
  <button type='warn' bindtap='upload'>点击此处上传图片</button>

  <!-- 已上传图片区域 -->
  <view class='photoBox'>
    <!-- 标题 -->
    <text>已上传图片历史记录</text>

    <!-- 图片集 -->
    <view>
      <block wx:for='{{historyPhotos}}' wx:key='history{{index}}'>
        <image src='{{item.photoUrl}}'></image>
      </block>
    </view>
  </view>
</view>
```

WXSS 文件(pages/add/add.wxss)的完整代码如下：

```
/* 已上传图片区域 */
.photoBox{
  margin-top: 50rpx;
  display: flex;
  flex-direction: column;
}
/* 标题样式 */
.photoBox text{
  margin-bottom: 30rpx;
  text-align: center;
}
/* 图片样式 */
.photoBox image{
  float: left;
  width: 250rpx;
```

```
  height: 250rpx;
}
```

JS 文件(pages/add/add.js)的完整代码如下：

```
const db = wx.cloud.database()
const photos = db.collection('photos')
var app = getApp()

//格式化当前日期
function formatDate() {
  var now = new Date()
  var year = now.getFullYear()
  var month = now.getMonth() + 1
  var day = now.getDate()

  if (month < 10) month = '0' + month
  if (day < 10) day = '0' + day

  return year + '-' + month + '-' + day
}

Page({
  /**
   * 自定义函数 -- 上传图片
   */
  upload: function() {
    //选择图片
    wx.chooseImage({
      count: 1,
      sizeType: ['compressed'],
      sourceType: ['album', 'camera'],
      success: function(res) {
        //loading 提示框表示正在上传图片
        wx.showLoading({
          title: '上传中',
        })
        //获取图片临时地址
        const filePath = res.tempFilePaths[0]

        //自定义云端的图片名称
        const cloudPath = Math.floor(Math.random() * 1000000) + filePath.match(/\.[^.]+?$/)[0]
        //上传图片到云存储空间中
        wx.cloud.uploadFile({
          cloudPath,
          filePath,
          success: res => {
            //提示上传成功
            wx.showToast({
              title: '上传成功!',
              duration: 3000
            })

            //获取用户个人基础信息
            let userInfo = app.globalData.userInfo
```

```
51.             //获取当天日期
52.             let today = formatDate()
53.
54.             //往云数据集中添加一条记录
55.             photos.add({
56.               data: {
57.                 photoUrl: res.fileID,
58.                 avatarUrl: userInfo.avatarUrl,
59.                 country: userInfo.country,
60.                 province: userInfo.province,
61.                 nickName: userInfo.nickName,
62.                 addDate: today
63.               },
64.               success: res => {
65.                 console.log(res)
66.               },
67.               fail: e => {
68.                 console.log(e)
69.               }
70.             })
71.           },
72.           fail: e => {
73.             //提示上传失败
74.             wx.showToast({
75.               icon: 'none',
76.               title: '上传失败',
77.             })
78.           }
79.         })
80.       },
81.       fail: e => {
82.         console.error(e)
83.       },
84.       complete: () => {
85.         //上传完成后关闭 loading 提示框
86.         wx.hideLoading()
87.         //更新图片历史记录
88.         this.getHistoryPhotos()
89.       }
90.     })
91.   },
92.   /**
93.    * 自定义函数 -- 获取已上传图片历史记录
94.    */
95.   getHistoryPhotos: function() {
96.     //获取当前用户的 openid
97.     let openid = app.globalData.openid
98.
99.     //从云数据集查找当前用户的上传记录
100.    photos.where({
101.      _openid: openid
102.    }).get({
103.      success: res => {
104.        this.setData({
105.          historyPhotos: res.data
```

```
        })
      }
    })
  },
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    //尝试输出用户基础信息
    //console.log(app.globalData.userInfo)
    //尝试输出用户 openid
    //console.log(app.globalData.openid)

    //更新图片历史记录
    this.getHistoryPhotos()
  }
})
```