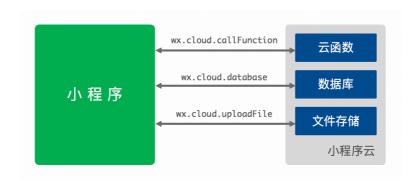
# 在云端使用小程序:云开发开发个人相册

### 准备工作

- 1. 已申请小程序:云开发公测资格的微信小程序账号
- 2. 公测版本的微信开发者工具(下载)
- 3. 个人相册小程序 Demo

#### 实验架构



## 任务一: 创建小程序:云开发环境

任务目标: 创建小程序: 云开发环境, 用于后面储存相册的用户信息和照片。

#### 开发者工具创建项目

打开微信开发者工具,创建一个新的小程序项目,项目目录选择个人相册 Demo 的目录,AppID 填写已经申请公测资格的小程序对应的 AppID。



#### 开通云开发环境

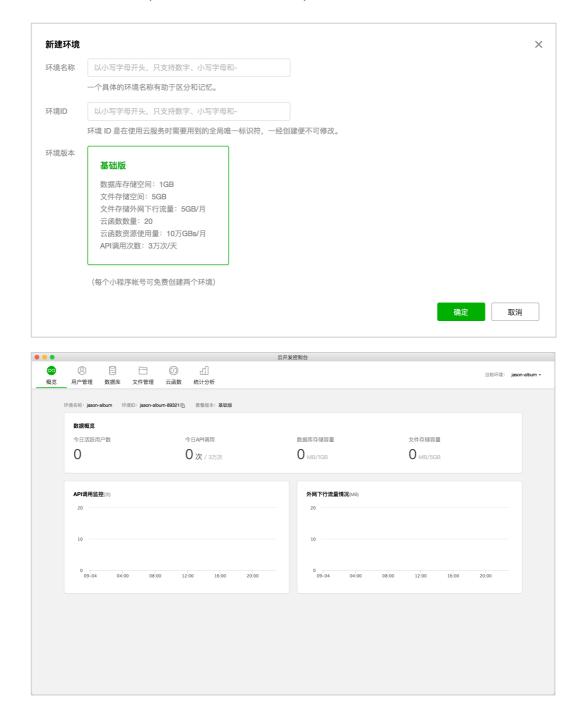
1. 点击开发者工具上的【云开发】按钮



2. 点击【同意】



3. 填写环境名称和环境 ID, 点击【确定】创建环境,即可进入云开发控制台。



#### 创建数据库

相册小程序会使用到云开发提供的数据库能力,数据库使用的是 MongoDB, 需要优先创建一个集合,

方便之后使用。

1. 打开云开发控制台,点击菜单栏中的【数据库】,然后点击左侧边栏的【添加集合】按钮



2. 输入集合名称"user", 然后点击确定即可创建集合。



### 任务二: 搭建个人相册

任务目标: 创建个人相册, 实现照片的上传和存储。

1. 打开项目目录下的 app.js 文件,修改初始化云函数配置中的 env 参数为上一任务中创建的**环境** ID. 并保存。

```
₽←
    Q
                        ≜≡
                                    app.js
                                     1
                                          //app.js
▶ ☐ images
                                         App({
                                     3
                                             onLaunch () {
▶ □ pages
                                                 // 初始化云函数
 .editorconfig
                                                  wx,cloud.init({
                                                     env: '环境ID',
                                     6
 .gitignore
                                                      traceUser:
 README.md
 JS app.js
                                    10
                                                 // 获取用户信息
 {} app.json
                                    11
                                                 wx.getSetting({
  wxss app.wxss
                                    12
                                                     success: res => {
                                    13
                                                         if (res.authSetting['scope
 project.config.json
                                                             // 已经授权,可以直接调用
                                    14
                                                             wx.getUserInfo({
                                    15
                                                                 success: res => {
```

2. 打开 pages/user/user.js 文件, 里面是用户登录所相关的 js 逻辑, 我们需要在文件中的 addUser

#### 函数里添加保存用户信息到数据库的逻辑,代码如下:

```
// 获取数据库实例
const db = wx.cloud.database({})

// 插入用户信息
let result = await db.collection('user').add({
   data: {
      nickName: user.nickName,
      albums: []
   }
})
```

#### 复制粘贴在文件的 70 行:

```
+ Q
                                         user.js
▶ 🗀 images
                                                             wx.switchTab({ url: '/pages/index/index' })
                                          61
▼ D pages
 ▶ 🗀 album
                                          63
                                         64
65
66
                                                    // 如果数据库没有此用户, 则添加
async addUser (user) {
 ▶ 🗀 common
 ▶ 🗀 index
                                                         if (app.globalData.hasUser) {
    return
 ▶ 🗀 photos
                                         68
 ▼ 🗁 user
                                         69
70
71
72
73
74
75
76
77
78
79
80
     JS user.js
                                                         // 获取数据库实例
                                                         const db = wx.cloud.database({})
     () user.json
     <> user.wxml
                                                         // 插入用户信息
                                                         let result = await db.collection('user').add({
     wxss user.wxss
                                                             data: {
 .editorconfig
                                                                  nickName: user.nickName,
 n.gitignore
                                                                 albums: []
 README.md
  JS app.js
                                         81
                                                         app.globalData.nickName = user.nickName
  {} app.json
                                         82
                                                         app.globalData.id = result._id
  wxss app.wxss
                                          84
                                               })
                                          85
```

保存文件后,就实现了用户登录的能力。页面会自动刷新,点击页面上的登录按钮,即可在页面上看到自己的昵称和头像。



3. 实现了用户登录的能力,我们需要接着来实现照片上传的能力。照片选择和上传的相关代码在

pages/photos/add.js 中,打开文件,找到 uploadPhoto 的函数,即可看到函数接收了一个 filePath 的参数,他是用户选择照片时照片的本地临时路径,我们需要使用云能力将图片上传到文件储存中,代码如下:

```
// 调用 wx.cloud.uploadFile 上传文件
return wx.cloud.uploadFile({
    cloudPath: `${Date.now()}-${Math.floor(Math.random(0, 1) * 10000000)}.png`,
    filePath
})
```

粘贴代码在文件的 66 行:

```
add.js ×

ror (const temprileratn or temprileratns) {
   items.push({
       src: tempFilePath
   }
+ Q
                                                  user,is
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
70
71
72
73
74
75
76
77
78
79
▶ 🗀 images
▼ 🗁 pages
 ▶ □ album
 ▶ 🗀 common
                                                                                this.setData({ photosNew: items })
 ▶ 🗀 index
                                                                     })
 ▼ 🗁 photos
      JS add.js
                                                                // 上传图片
      {} add.ison
                                                                 uploadPhoto (filePath) {
      <> add.wxml
                                                                     return wx.cloud.uploadFile({
    cloudPath: `${Date.now()}-${Math.floor(Math.random(0, 1) * 10000000)}.png`,
      wxss add.wxss
                                                                           filePath
      JS photos.js
                                                                     })
      {} photos.json
                                                               },
      <> photos.wxml
                                                                // 预览图片
       wxss photos.wxss
                                                                 previewImage (e) {
                                                                     const current = e.target.dataset.sr any
const photos = this.data.photosNew.map(photo => photo.src)
 ▶ 🗀 user
  .editorconfig
                                                                      wx.previewImage({
    current: current.src,
  gitignore .
  README.md
                                                                           urls: photos
```

4. 保存文件后,上传的能力就完成了。文件上传后,和登录一样,我们需要将上传好的文件信息存储在数据库中,这个逻辑在 pages/photos/add.js 文件的 addPhotos 函数完成。相册的数据存储在用户信息中,函数已经帮我们完成了对用户信息的更新,我们只需要完成用户信息的更新即可、代码如下:

```
// 写入集合
db.collection('user').doc(app.globalData.id).update({
    data: {
        albums: db.command.set(app.globalData.allData.albums)
    }
}).then(result => {
    console.log('写入成功', result)
    wx.navigateBack()
})
```

粘贴代码到文件的102行:

```
... ≛≡ 8←
▶ 🗀 images
                                  90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
118
119
111
111
▼ 🗁 pages
                                            ▶ 🗀 album
▶ 🗀 common
▶ 🗀 index
▼ 🗁 photos
   JS add.js
   {} add.json
   <> add.wxml
                                                app.globalData.allData.albums[this.data.albumIndex].photos = [...oldPhotos, ...albumPhotos]
   was add.wxss
   JS photos.js
                                                <> photos.wxml
   wss photos.wxss
                                               )).then(result => {
    console.log('写入成功', result)
    wx.navigateBack()
 ▶ 🗀 user
                                  113
114
115
116
 .gitignore
 README.md
 JS app.js
 () app.json
                                                                                                                     行110,列23 JavaScrip
                                 /pages/photos/add.js 3.0 KB
```

保存文件后,页面会自动刷新,上传功能就完成啦。

5. 提供了上传的能力,我们还需要删除的能力,代码已经提前写好了长按弹出菜单的功能,但是删除文件的代码还未完成。需要使用 wx.cloud.deleteFile 去删除文件,这个逻辑在 pages/photos/photos.js 文件的 deleteFile 函数完成,代码如下:

```
// 删除文件
return wx.cloud.deleteFile({
   fileList: [fileId]
}).then(res => {
   this.saveImageDelele(fileId)
})
```

粘贴代码到文件的 47 行:

```
+ Q
                      }
▶ 🗀 images
                                                         1)
                                          40
▼ 🗁 pages
                                          41
42
 ▶ 🗀 album
                                          43
                                                     // 删除照片
 ▶ 🗀 common
                                          44
45
                                                     async deleteFile (idx) {
                                                         const fileId = this.data.photoIds[idx]
 ▶ 🗀 index
                                          46
47
 ▼ 🗁 photos
                                                          return wx.cloud.deleteFile({
     JS add.js
                                          48
49
50
51
52
                                                              fileList: [fileId]
     {} add.json
                                                         }).then(res => {
     <> add.wxml
                                                             this.saveImageDelele(fileId)
                                          53
54
55
     JS photos.js
                                                     async saveImageDelele (fileId) {
                                          56
57
                                                         const photos = app.globalData.allData.albums[this
const newFileIds = this.data.photoIds.filter(id =
     <> photos.wxml
     wxss photos.wxss
                                          58
                                                         const newPhotos = photos.filter(photo => !!~newFi
```

保存文件后,页面会自动刷新。至此,我们已经完成了一个简单的小程序的搭建,并将最核心的能力 使用小程序·云开发完成,快使用微信开发者工具或者扫描开发二维码用手机体验吧~