Girls Help - 微信云开发后端API调用文档

一、数据库设计

1. 用户集合 (users)

```
{
  _id: ObjectId,
  _openid: String,          // 微信云开发自动获取
  userId: String,           // 用户唯一ID
  nickName: String,         // 昵称
  avatarUrl: String,        // 头像URL
  isAnonymous: Boolean,     // 是否匿名
  resources: [String],      // 可提供的资源 ['卫生巾', '纸巾', '暖宝宝']
  showOnMap: Boolean,       // 是否在地图上显示
  stats: {
    helpGiven: Number,      // 帮助他人次数
    helpReceived: Number    // 获得帮助次数
  },
  currentLocation: {
    latitude: Number,
    longitude: Number,
    accuracy: Number,
    updateTime: Date
  },
  privacyOffset: Number,    // 隐私偏移量（米）
  joinTime: Date,
  lastActiveTime: Date
}
```

2. 求助请求集合 (help_requests)

```
{
  _id: ObjectId,
  _openid: String,          // 求助者openid
  requestId: String,        // 请求唯一ID
  type: String,             // 类型: 'pad', 'tissue', 'safety', 'other'
  note: String,             // 补充说明
  status: String,           // 'pending', 'matched', 'active', 'completed', 'cancelled'
  location: {
    latitude: Number,
    longitude: Number,
    accuracy: Number
  },
  matchedHelperId: String,  // 匹配的帮助者ID
  createTime: Date,
  matchTime: Date,          // 匹配时间
  completeTime: Date        // 完成时间
}
```

3. 位置更新集合 (user_locations)

```
{
  _id: ObjectId,
  _openid: String,
  location: {
    latitude: Number,
    longitude: Number,
    accuracy: Number
  },
  showOnMap: Boolean,
  resources: [String],
  isHelper: Boolean,        // 当前是否可作为帮助者
  isSeeker: Boolean,        // 当前是否需要帮助
  updateTime: Date
}
```

4. 联系记录集合 (contact_records)

```
{
  _id: ObjectId,
  fromOpenid: String,       // 发起者
  toOpenid: String,         // 接收者
  type: String,             // 'help_request', 'help_offer', 'response'
  status: String,           // 'pending', 'accepted', 'rejected', 'completed'
  message: String,
  createTime: Date,
  responseTime: Date
}
```

二、云函数API列表

API 1: 微信登录

云函数: login

调用位置: pages/Login/index.js:174 (sendToServer方法)

请求参数:

```
{
  code: String,           // wx.login()获取的code
  userInfo: {
    nickName: String,
    avatarUrl: String,
    isAnonymous: Boolean
  }
}
```

返回数据:

```
{
  success: Boolean,
  token: String,          // 自定义token（可用openid+时间戳生成）
  userInfo: {
    userId: String,
    nickName: String,
```

```
  avatarUrl: String,
  isAnonymous: Boolean,
  joinDays: Number,
  resources: [String],
  showOnMap: Boolean
 }
}
```

云函数实现要点:

```
// 云函数入口
exports.main = async (event, context) => {
 const { code, userInfo } = event;
 const wxContext = cloud.getWXContext();

 // 1. 检查用户是否存在
 const userResult = await db.collection('users').where({
   _openid: wxContext.OPENID
 }).get();

 let user;
 if (userResult.data.length === 0) {
   // 2. 新用户，创建记录
   user = {
     _openid: wxContext.OPENID,
     userId: 'user_' + Date.now() + '_' + Math.random().toString(36).substr(2,
9),
     nickName: userInfo.nickName || '姐妹',
     avatarUrl: userInfo.avatarUrl || '',
     isAnonymous: userInfo.isAnonymous || false,
     resources: [],
     showOnMap: true,
     stats: { helpGiven: 0, helpReceived: 0 },
     joinTime: new Date(),
     lastActiveTime: new Date()
   };
   await db.collection('users').add({ data: user });
 } else {
   // 3. 老用户，更新最后活跃时间
   user = userResult.data[0];
   await db.collection('users').doc(user._id).update({
     data: { lastActiveTime: new Date() }
   });
 }

 // 4. 生成token
 const token = Buffer.from(wxContext.OPENID + ':' +
Date.now()).toString('base64');

 return {
   success: true,
   token: token,
   userInfo: {
     userId: user.userId,
     nickName: user.nickName,
```

```
    avatarUrl: user.avatarUrl,
    isAnonymous: user.isAnonymous
  }
};
```

};

API 2: 创建求助请求

云函数: createHelpRequest

调用位置: pages/home/index.js:64 (submitRequest方法)

请求参数:
{
  token: String,
  type: String,   // 'pad', 'tissue', 'safety', 'other'
  note: String,   // 补充说明
  location: {
    latitude: Number,
    longitude: Number,
    accuracy: Number
  }
}

返回数据:
{
  success: Boolean,
  requestId: String,
  status: String,   // 'pending'
  estimatedTime: Number // 预计匹配时间（秒）
}

API 3: 获取附近用户

云函数: getNearbyUsers

调用位置: pages/map/index.js:114 (generateMarkers方法)

请求参数:
{
  token: String,
  location: {
    latitude: Number,
    longitude: Number
  },
  radius: Number,   // 搜索半径（米），默认2000
  limit: Number   // 返回数量限制，默认20
}

返回数据:

```
{
  success: Boolean,
  users: [
    {
      userId: String,
      type: String,          // 'helper' | 'seeker'
      location: {
        latitude: Number,    // 隐私模糊后的位置
        longitude: Number
      },
      distance: Number,      // 距离（米）
      provide: [String],     // 可提供的资源
      need: String           // 需要的帮助
    }
  ]
}
```

云函数实现要点（位置模糊处理）：

```
// 计算两点距离
function getDistance(lat1, lon1, lat2, lon2) {
  const R = 6371000;
  const dLat = (lat2 - lat1) * Math.PI / 180;
  const dLon = (lon2 - lon1) * Math.PI / 180;
  const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
          Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
          Math.sin(dLon/2) * Math.sin(dLon/2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
  return R * c;
}

// 位置模糊处理
function blurLocation(lat, lon, offsetMeters) {
  const angle = Math.random() * Math.PI * 2;
  const offset = Math.random() * offsetMeters;
  const dLat = (Math.sin(angle) * offset) / 111000;
  const dLon = (Math.cos(angle) * offset) / (111000 * Math.cos(lat * Math.PI / 180));
  return { lat: lat + dLat, lon: lon + dLon };
}
```

API 4: 获取用户资料

云函数: getUserProfile

调用位置: pages/profile/index.js:onLoad

请求参数:

```
{
  token: String
}
```

返回数据:
```
{
  success: Boolean,
  user: {
    userId: String,
    nickName: String,
    avatarUrl: String,
    joinDays: Number,
    resources: [String],
    showOnMap: Boolean,
    stats: {
      helpGiven: Number,
      helpReceived: Number
    }
  }
}
```

API 5: 更新用户资源

云函数: updateUserResources

调用位置: pages/profile/index.js (需要新增资源编辑功能)

请求参数:
```
{
  token: String,
  resources: [String]    // ['卫生巾', '纸巾', '暖宝宝', '热水']
}
```

返回数据:
```
{
  success: Boolean
}
```

API 6: 更新隐私设置

云函数: updatePrivacySetting

调用位置: pages/profile/index.js:20 (toggleMapVisibility方法)

请求参数:
```
{
  token: String,
  showOnMap: Boolean
}
```

返回数据:
```
{
  success: Boolean
}
```

API 7: 更新用户位置

云函数: updateUserLocation

调用位置: app.js:134, pages/map/index.js:54

请求参数:

```
{
  token: String,
  location: {
    latitude: Number,
    longitude: Number,
    accuracy: Number
  }
}
```

返回数据:

```
{
  success: Boolean
}
```

API 8: 联系用户

云函数: contactUser

调用位置: pages/map/index.js:333 (contactPerson方法)

请求参数:

```
{
  token: String,
  targetUserId: String,
  type: String          // 'help_request' | 'help_offer'
}
```

返回数据:

```
{
  success: Boolean,
  contactId: String,
  message: String
}
```

API 9: 取消求助请求

云函数: cancelHelpRequest

调用位置: pages/home/index.js:166 (onLongPress方法)

请求参数:

```
{
  token: String,
  requestId: String
}
```

返回数据:
{
  success: Boolean
}

API 10: 完成互助

云函数: completeHelp

调用位置: pages/home/index.js:47 (handleStartRequest方法 - active状态点击)

请求参数:
{
  token: String,
  requestId: String
}

返回数据:
{
  success: Boolean,
  message: String
}

三、前端调用示例

封装云函数调用方法

建议在项目中创建 utils/cloud.js:

```
// utils/cloud.js

// 云函数调用封装
function callCloudFunction(name, data) {
  return new Promise((resolve, reject) => {
    wx.cloud.callFunction({
      name: name,
      data: data,
      success: res => {
        if (res.result.success) {
          resolve(res.result);
        } else {
          reject(res.result);
        }
      },
      fail: err => {
        reject(err);
      }
    });
  });
}
```

```
// 获取token
function getToken() {
  return wx.getStorageSync('token') || getApp().globalData.token;
}

module.exports = {
  callCloudFunction,
  getToken,
```

```
  // 认证相关
  login: (code, userInfo) => callCloudFunction('login', { code, userInfo }),

  // 求助相关
  createHelpRequest: (type, note, location) =>
    callCloudFunction('createHelpRequest', {
      token: getToken(),
      type,
      note,
      location
    }),

  cancelHelpRequest: (requestId) =>
    callCloudFunction('cancelHelpRequest', {
      token: getToken(),
      requestId
    }),

  completeHelp: (requestId) =>
    callCloudFunction('completeHelp', {
      token: getToken(),
      requestId
    }),

  // 附近用户
  getNearbyUsers: (location, radius = 2000, limit = 20) =>
    callCloudFunction('getNearbyUsers', {
      token: getToken(),
      location,
      radius,
      limit
    }),

  // 用户相关
  getUserProfile: () =>
    callCloudFunction('getUserProfile', {
      token: getToken()
    }),

  updateUserResources: (resources) =>
    callCloudFunction('updateUserResources', {
      token: getToken(),
      resources
    }),

  updatePrivacySetting: (showOnMap) =>
```

```
  callCloudFunction('updatePrivacySetting', {
    token: getToken(),
    showOnMap
  }),

updateUserLocation: (location) =>
  callCloudFunction('updateUserLocation', {
    token: getToken(),
    location
  }),

// 联系相关
contactUser: (targetUserId, type) =>
  callCloudFunction('contactUser', {
    token: getToken(),
    targetUserId,
    type
  }),

// 情绪支持
emotionSupport: (message) =>
  callCloudFunction('emotionSupport', {
    token: getToken(),
    message
  })
```

};

四、具体页面修改点

pages/Login/index.js

修改位置: 第174行 sendToServer 方法

// 替换原有的模拟代码
const cloud = require('../../utils/cloud.js');

sendToServer: function (code, userInfo) {
 const that = this;
 this.setData({ loadingText: '正在连接服务器...' });

```
cloud.login(code, userInfo)
  .then(res => {
    // 保存token和用户信息
    wx.setStorageSync('token', res.token);
    wx.setStorageSync('userInfo', res.userInfo);

    app.globalData.userInfo = res.userInfo;
    app.globalData.token = res.token;

    that.requestLocationPermission();
  })
  .catch(err => {
    console.error('Login failed:', err);
    that.handleError('登录失败，请重试');
```

```
  });
```

}

---

pages/home/index.js

修改位置: 第64行 submitRequest 方法

const cloud = require('../../utils/cloud.js');

submitRequest: function (type, note) {
 const that = this;
 const app = getApp();

```
this.setData({ helpStatus: 'requesting' });
app.globalData.helpStatus = 'requesting';

// 获取当前位置
wx.getLocation({
  type: 'gcj02',
  success: (locRes) => {
    const location = {
      latitude: locRes.latitude,
      longitude: locRes.longitude,
      accuracy: locRes.accuracy
    };

    cloud.createHelpRequest(type, note, location)
      .then(res => {
        // 保存requestId
        that.requestId = res.requestId;

        // 轮询检查匹配状态
        that.pollMatchStatus();
      })
      .catch(err => {
        console.error('Create help request failed:', err);
        wx.showToast({ title: '请求失败，请重试', icon: 'none' });
        that.setData({ helpStatus: 'idle' });
        app.globalData.helpStatus = 'idle';
      });
  },
  fail: () => {
    wx.showToast({ title: '获取位置失败', icon: 'none' });
    that.setData({ helpStatus: 'idle' });
    app.globalData.helpStatus = 'idle';
  }
});
```

},

// 轮询匹配状态
pollMatchStatus: function () {
  const that = this;
  const app = getApp();
  let pollCount = 0;
  const maxPolls = 20; // 最多轮询20次（100秒）

```
  const poll = () => {
    pollCount++;

    if (pollCount > maxPolls) {
      // 超时
      that.setData({ helpStatus: 'idle' });
      app.globalData.helpStatus = 'idle';
      wx.showToast({ title: '暂无附近姐妹响应', icon: 'none' });
      return;
    }

    // 调用获取状态的云函数
    cloud.callCloudFunction('getHelpRequestStatus', {
      token: cloud.getToken(),
      requestId: that.requestId
    }).then(res => {
      if (res.status === 'matched' || res.status === 'active') {
        that.setData({ helpStatus: 'active' });
        app.globalData.helpStatus = 'active';
        wx.showToast({
          title: '附近有姐妹响应了你的请求！',
          icon: 'success',
          duration: 5000
        });
      } else {
        // 继续轮询
        setTimeout(poll, 5000);
      }
    }).catch(() => {
      setTimeout(poll, 5000);
    });
  };

  setTimeout(poll, 5000); // 5秒后开始轮询
```

}

---

pages/map/index.js

修改位置: 第114行 generateMarkers 方法

const cloud = require('../../utils/cloud.js');

generateMarkers: function () {
  const that = this;
  const { userLocation } = this.data;

```
if (!userLocation) {
  this.generateDefaultMarkers();
  return;
}

wx.showLoading({ title: '加载附近姐妹...' });

cloud.getNearbyUsers(userLocation, 2000, 20)
  .then(res => {
    const markers = res.users.map((user, index) => ({
      id: index + 1,
      longitude: user.location.longitude,
      latitude: user.location.latitude,
      type: user.type,
      distance: user.distance,
      provide: user.provide ? user.provide.join(',') : '',
      need: user.need || '',
      width: 32,
      height: 32,
      iconPath: user.type === 'helper'
        ? '/images/marker-helper.png'
        : '/images/marker-seeker.png',
      alpha: 0.9,
      customCallout: {
        anchorY: 0,
        anchorX: 0,
        display: 'BYCLICK',
        textAlign: 'center',
        bgColor: user.type === 'helper' ? '#FFA4A4' : '#BADFDB',
        color: '#333',
        fontSize: 12,
        borderRadius: 8,
        padding: 8,
        content: user.type === 'helper' ? '🤝' : '🆘'
      }
    }));

    // 添加用户当前位置标记
    markers.push({
      id: 0,
      longitude: userLocation.longitude,
      latitude: userLocation.latitude,
      type: 'user',
      width: 24,
      height: 24,
      iconPath: '/images/marker-user.png',
      alpha: 1,
      zIndex: 100
    });

    that.setData({ markers });
    wx.hideLoading();
  })
  .catch(err => {
    console.error('Get nearby users failed:', err);
```

```
    wx.hideLoading();
    that.generateDefaultMarkers(); // 失败时使用默认数据
  });
```

}

---

pages/profile/index.js

修改位置: 整个页面需要重写

// pages/profile/index.js
const cloud = require('../../utils/cloud.js');

Page({
 data: {
  user: {
   name: '',
   avatar: '',
   joinDays: 0,
   helpGiven: 0,
   helpReceived: 0,
   resources: [],
   showOnMap: true
  },
  loading: true
 },

```
onLoad: function (options) {
  this.loadUserProfile();
},

onShow: function () {
  // 页面显示时刷新数据
  if (!this.data.loading) {
    this.loadUserProfile();
  }
},

// 加载用户资料
loadUserProfile: function () {
  const that = this;
  that.setData({ loading: true });

  cloud.getUserProfile()
    .then(res => {
      that.setData({
        user: res.user,
        loading: false
      });
    })
    .catch(err => {
      console.error('Get user profile failed:', err);
      that.setData({ loading: false });
```

```
    });
},

// 切换地图显示状态
toggleMapVisibility: function () {
  const that = this;
  const newShowOnMap = !this.data.user.showOnMap;

  cloud.updatePrivacySetting(newShowOnMap)
    .then(res => {
      that.setData({
        'user.showOnMap': newShowOnMap
      });
      wx.showToast({
        title: newShowOnMap ? '已开启地图显示' : '已关闭地图显示',
        icon: 'success'
      });
    })
    .catch(err => {
      wx.showToast({ title: '设置失败', icon: 'none' });
    });
},

// 编辑可提供资源
editResources: function () {
  const that = this;
  const currentResources = this.data.user.resources;

  // 显示多选对话框
  const allResources = ['卫生巾', '纸巾', '暖宝宝', '热水', '充电宝', '巧克力', '雨
伞'];
  const items = allResources.map(r => ({
    name: r,
    checked: currentResources.includes(r)
  }));

  wx.showActionSheet({
    itemList: allResources,
    itemColor: '#000000',
    success: (res) => {
      // 简化处理：实际需要自定义多选组件
      const selected = [allResources[res.tapIndex]];
      that.updateResources(selected);
    }
  });
},

// 更新资源
updateResources: function (resources) {
  const that = this;

  cloud.updateUserResources(resources)
    .then(res => {
      that.setData({
        'user.resources': resources
```

```
      });
      wx.showToast({ title: '已更新', icon: 'success' });
    })
    .catch(err => {
      wx.showToast({ title: '更新失败', icon: 'none' });
    });
},

// 关于我们
showAbout: function () {
  wx.showModal({
    title: '关于 Girls Help',
    content: 'Girls Help 是一款专为女性用户设计的即时互助小程序。',
    showCancel: false
  });
}
```

});

五、app.js 配置

初始化云开发

在 app.js 开头添加:

```
// app.js
App({
  onLaunch: function (options) {
    // 初始化云开发
    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({
        env: 'your-env-id',  // 替换为你的云环境ID
        traceUser: true
      });
    }
```

```
    this.checkUpdate();
    this.initUserInfo();
},

// ... 其他代码保持不变
```

});

修改 app.json 配置

```
{
  "pages": [
    "pages/Login/index",
    "pages/home/index",
    "pages/map/index",
    "pages/profile/index"
```

```
  ],
  "window": {
    "backgroundTextStyle": "light",
    "navigationBarBackgroundColor": "#FFB6C1",
    "navigationBarTitleText": "Girls Help",
    "navigationBarTextStyle": "white"
  },
  "tabBar": {
    "color": "#666666",
    "selectedColor": "#FFB6C1",
    "backgroundColor": "#ffffff",
    "borderStyle": "black",
    "list": [
      {
        "pagePath": "pages/home/index",
        "text": "求助",
        "iconPath": "images/tab-home.png",
        "selectedIconPath": "images/tab-home-active.png"
      },
      {
        "pagePath": "pages/map/index",
        "text": "附近",
        "iconPath": "images/tab-map.png",
        "selectedIconPath": "images/tab-map-active.png"
      },
      {
        "pagePath": "pages/profile/index",
        "text": "我的",
        "iconPath": "images/tab-profile.png",
        "selectedIconPath": "images/tab-profile-active.png"
      }
    ]
  },
  "permission": {
    "scope.userLocation": {
      "desc": "你的位置信息将用于显示附近的姐妹"
    }
  },
  "cloud": true,
  "style": "v2",
  "sitemapLocation": "sitemap.json"
}
```

六、云函数创建清单

在微信开发者工具中创建以下云函数:

cloudfunctions/
├── login/
│ └── index.js
├── createHelpRequest/
│ └── index.js
├── cancelHelpRequest/
│ └── index.js
├── completeHelp/
│ └── index.js
├── getHelpRequestStatus/
│ └── index.js
├── getNearbyUsers/
│ └── index.js
├── getUserProfile/
│ └── index.js
├── updateUserResources/
│ └── index.js
├── updatePrivacySetting/
│ └── index.js
├── updateUserLocation/
│ └── index.js
├── contactUser/
│ └── index.js
└── emotionSupport/
└── index.js

每个云函数的 index.js 基本模板:

// cloudfunctions/login/index.js
const cloud = require('wx-server-sdk');
cloud.init({ env: cloud.DYNAMIC_CURRENT_ENV });
const db = cloud.database();

exports.main = async (event, context) => {
const wxContext = cloud.getWXContext();

```
try {
  // 业务逻辑...

  return {
    success: true,
    // 返回数据...
  };
} catch (err) {
  console.error(err);
  return {
    success: false,
    error: err.message
  };
}
```

};

七、安全注意事项

1. Token验证: 每个云函数都应验证token的有效性
2. openid获取: 使用 cloud.getWXContext() 获取用户openid，不要依赖前端传入
3. 位置隐私: 服务器返回的位置必须经过模糊处理
4. 数据权限: 确保用户只能修改自己的数据
5. 频率限制: 对创建求助请求等操作添加频率限制