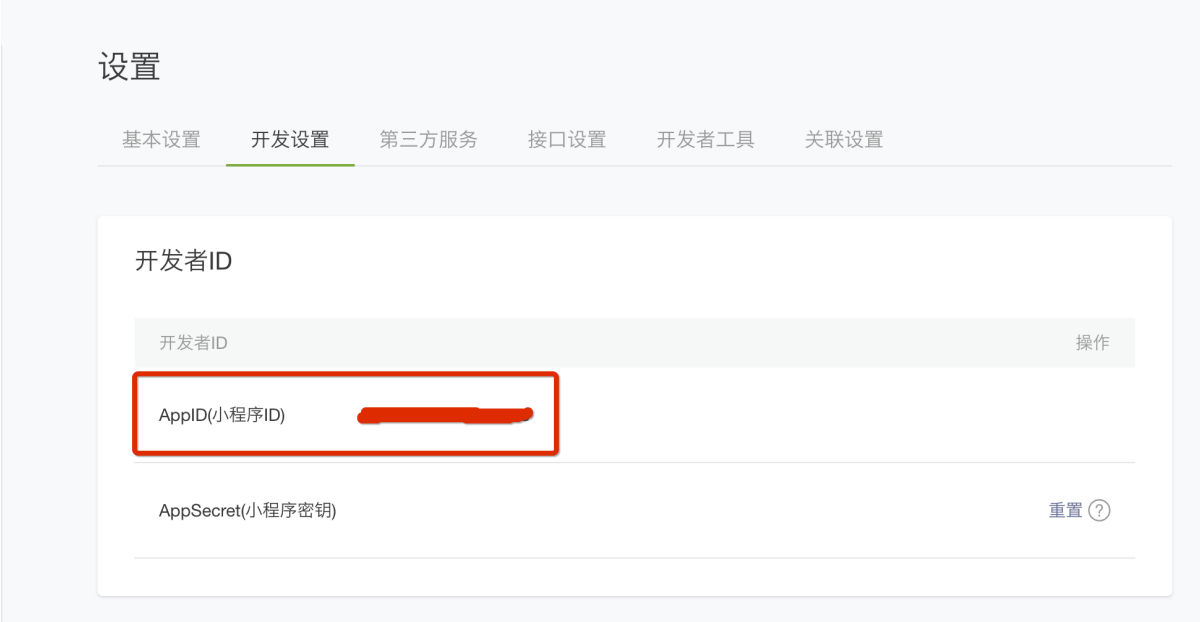
# 基于云开发制作电商小程序

## 准备工作

1. 已经申请小程序，获取小程序 AppID 和 Secret 在小程序管理后台中，【设置】 -> 【开发设置】 下可以获取微信小程序 AppID 和 Secret。



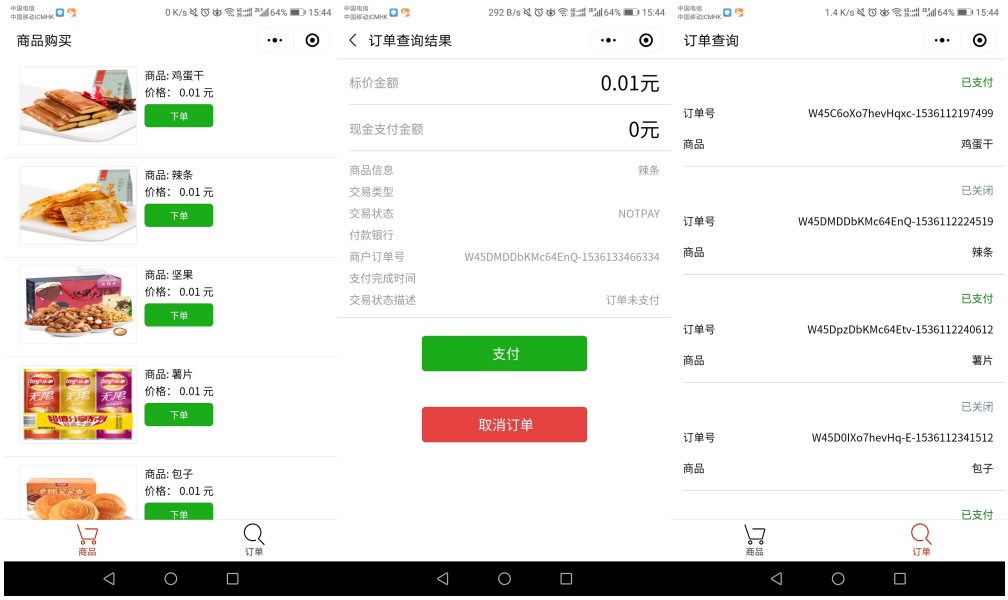

2. 微信支付商户号，获取商户号和商户密钥

在微信支付商户管理平台中，【账户中心】 -> 【商户信息】 下可以获取微信支付商户号。


在【账户中心】 -> 【API安全】 下可以设置商户密钥。

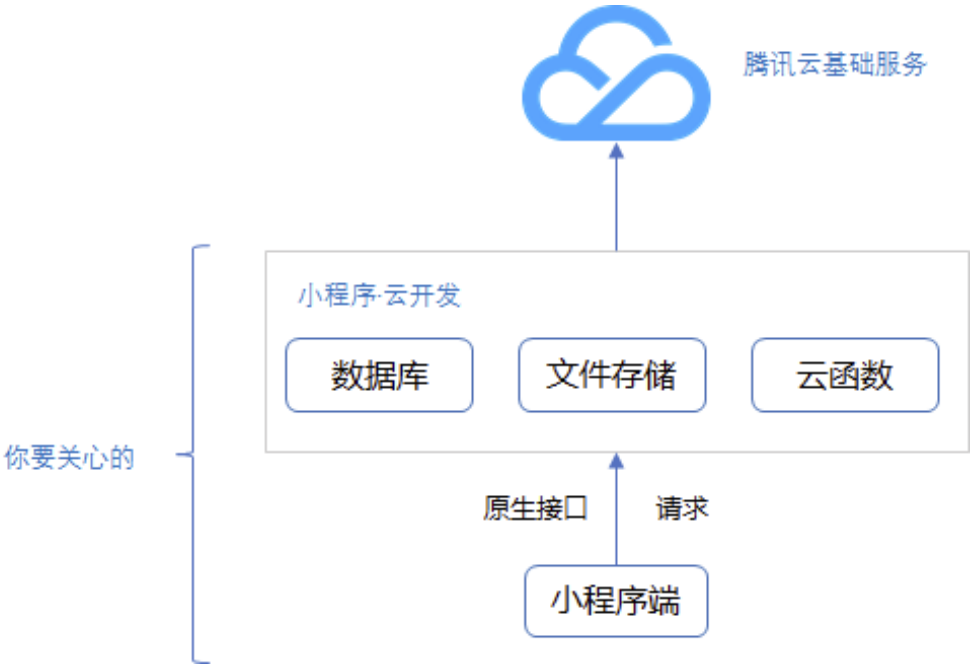

3. 微信开发者 IDE(下载)
4. 下载电商小程序代码包
5. 运行环境 Node8.9 或以上

# 效果预览




# 知识点

1. 学习如何用云开发控制台上传图片、录入商品数据。
2. 学习如何用云开发插入、读取数据。
3. 学习如何用 wx-js-utils 和云函数实现小程序微信支付逻辑。
4. 了解微信支付相关文档 小程序侧 和 微信支付侧
5. 了解微信小程序模板消息相关文档 小程序模板消息

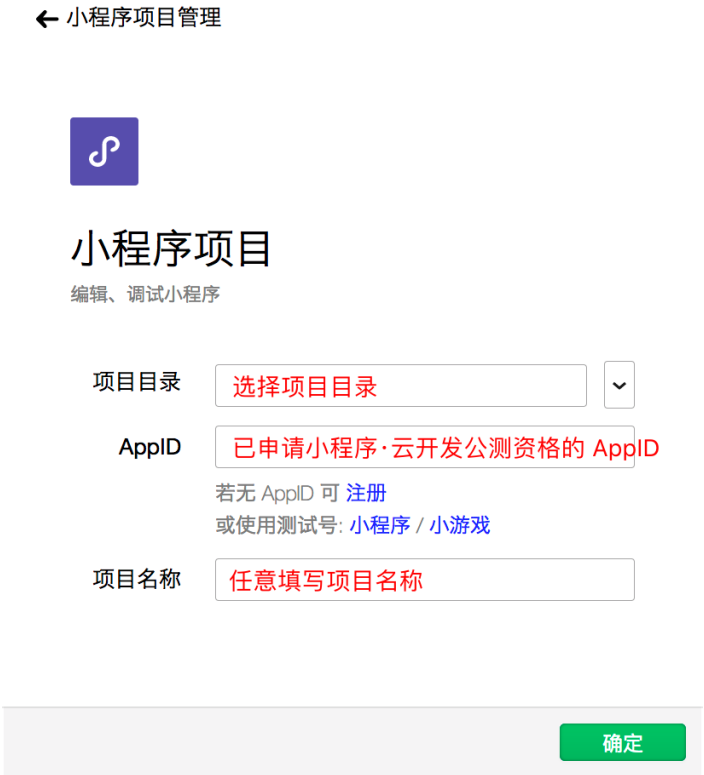



# 任务一：创建小程序·云开发环境

**任务目标**: 创建小程序·云开发环境，用于后面存储信息和开发云函数。

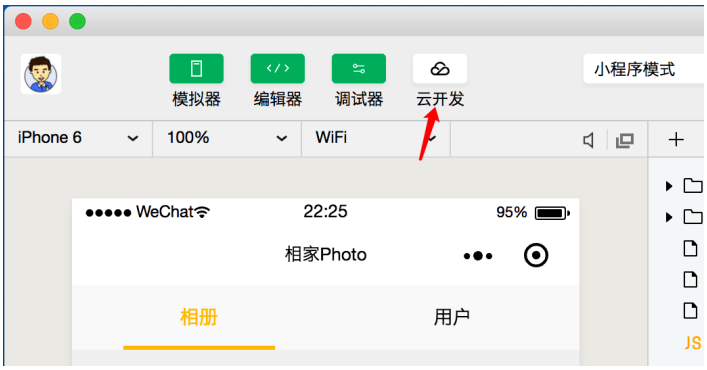
## 开发者工具创建项目

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

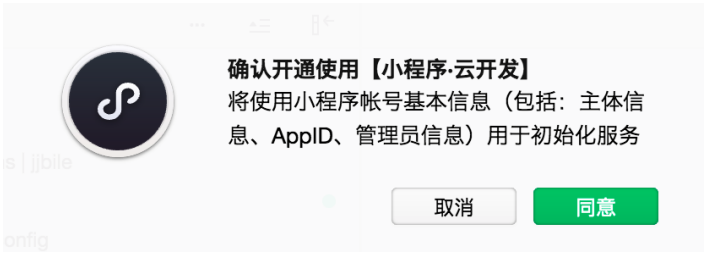



## 开通云开发环境

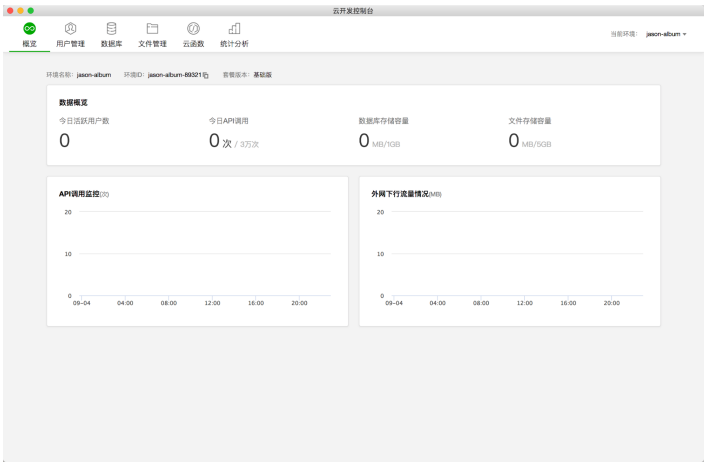
1. 点击开发者工具上的【云开发】按钮




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。

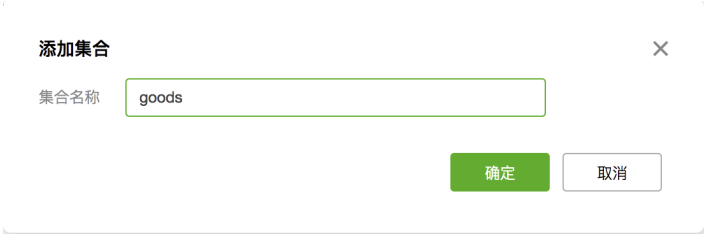



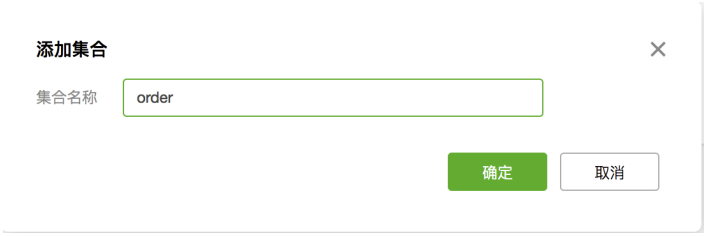

## 创建数据库

电商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. 分别输入集合名称 "goods" 和 "order"，然后点击确定即可创建集合。

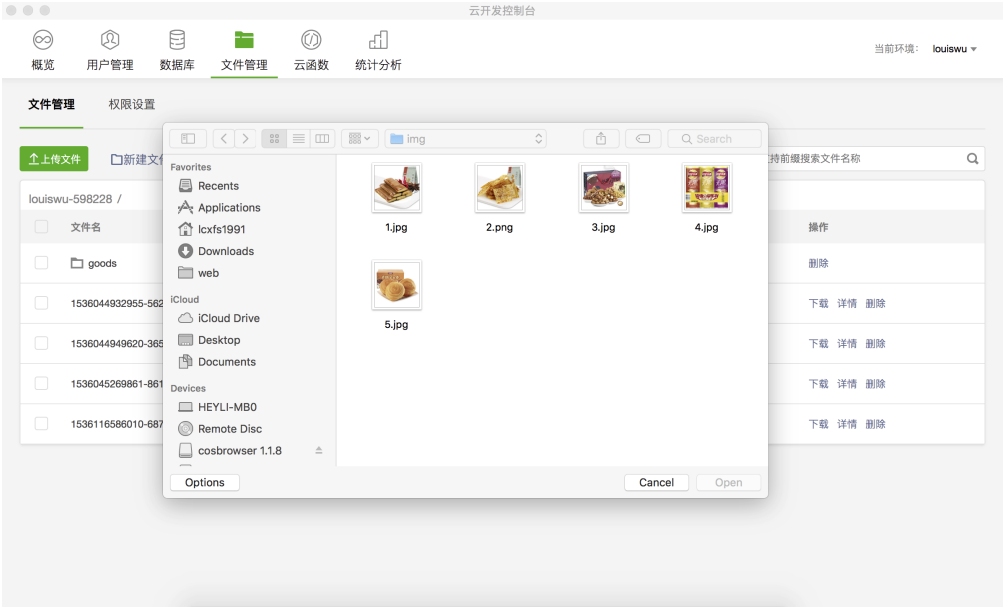


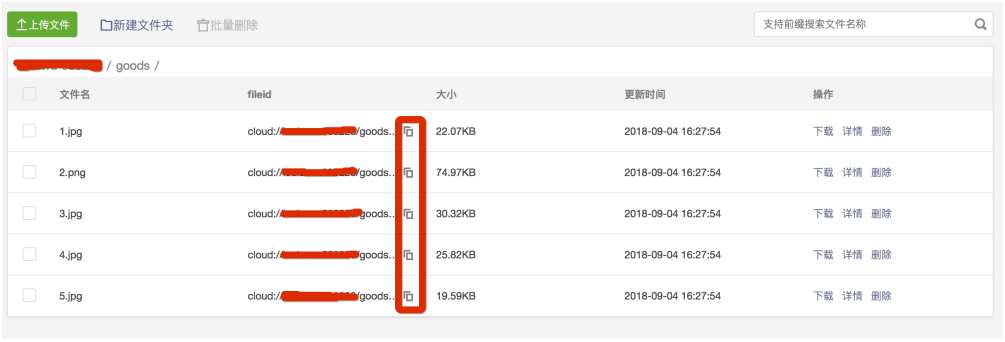

3. 要将 "goods" 集合的权限，设置为 "所有用户可读，仅创建者及管理员可写"。


## 录入商品数据

1. 在云开发控制台，点击菜单栏中的 【文件管理】，然后点击左侧的【新建文件夹】，新建一个目录 `goods`，点击进入 `goods` 目录后，点击左侧的【上传文件】按钮，将小程序项目目录 `./img` 中的商品图片，都选择进行上传。

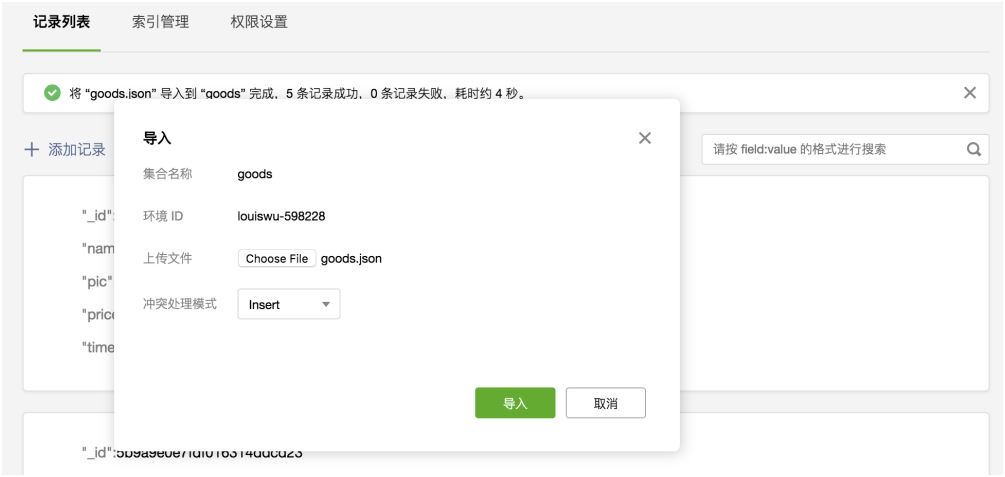



上传完毕后，分别点击红框框选的复制按钮，将这些文件的 `fileID` 保存下来。




打开 `cloud/database/goods.json`，将保存下来的文件 `fileID`，依照顺序，填入数据的 `pic` 字段中。

2. 返回【数据库】，在 `goods` 这个 `collection` 下，点击【导入】，选择 `cloud/database/goods.json` 文件进入数据批量导入。




# 任务二：读取数据与发起订单

**任务目标**: 利用云开发的小程序端接口读取商品数据和在云函数发起微信支付订单

读取数据

1. 将下面代码，输入到 `client/pages/list/index.js` 中的 `getGoodsList` 方法中，通过此方法，可以获取所有商品的数据。

```
// 利用云开发新接口，读取所有商品数据
const db = wx.cloud.database();
const result = await db.collection('goods').get();

let data = result.data || [];

this.setData({
    goods: data
});
```

## 发起订单

1. 将下面代码，输入到 `client/pages/list/index.js` 中的 `makeOrder` 方法中，通过此方法，可以调用云函数进行订单发起。

```
wx.showLoading({
    title: '正在下单',
});

// 利用云开发新接口，调用云函数发起订单
let id = e.target.dataset.goodid;
const { result } = await wx.cloud.callFunction({
    name: 'pay',
    data: {
        type: 'unifiedorder',
        data: {
            goodId: id
        }
    }
});

const data = result.data;

wx.hideLoading();

wx.navigateTo({
    url: `/pages/result/index?id=${data.out_trade_no}`
});
```

2. 创建云函数 `pay`

在 `cloud/functions/pay` 目录下，运行以下命令，安装依赖。

```
npm i --production
```

3. 填写微信商户与微信小程序相关配置 新建 `cloud/functions/pay/config/index.js`，并填入小程序的 `AppId`，还有微信支付的商户号 `MCHID` 和 商户密钥 `KEY`：

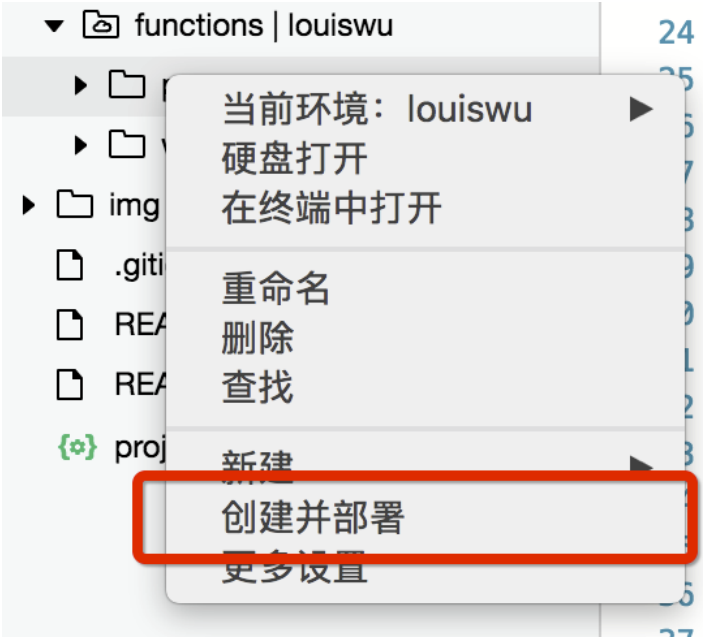
```
module.exports = {
  mpAppId: '', // 小程序 AppID
  envName: '', // 小程序云开发环境ID
  MCHID: '', //商户号
  KEY: '', // 商户密钥
  TIMEOUT: 10000 // 毫秒
};
```

4. 将下面代码，输入到 `cloud/functions/pay/index.js` 中的 `switch` 代码块中，通过此 `case`，可以进行订单的发起。

```
case 'unifiedorder': {
    // 在云函数参数中，提取商品 ID
    const { goodId } = data;

    // 查询该商品 ID 是否存在于数据库中，并将数据提取出来
    let goods = await goodCollection.doc(goodId).get();

    if (!goods.data.length) {
        return new Res({
            code: 1,
            message: '找不到商品'
        });
    }

    // 在云函数中提取数据，包括名称、价格才更合理安全，
    // 因为从端里传过来的商品数据都是不可靠的
    let good = goods.data[0];

    // 拼凑微信支付统一下单的参数
    const curTime = Date.now();
    const tradeNo = `${goodId}-${curTime}`;
    const body = good.name;
    const spbill_create_ip = ip.address() || '127.0.0.1';
    // 云函数暂不支付 http 触发器，因此这里回调 notify_url 可以先随便填。
    const notify_url = 'http://www.qq.com'; //'127.0.0.1';
    const total_fee = good.price;
    const time_stamp = '' + Math.ceil(Date.now() / 1000);
    const out_trade_no = `${tradeNo}`;
    const sign_type = WXPayConstants.SIGN_TYPE_MD5;

    let orderParam = {
        body,
        spbill_create_ip,
        notify_url,
        out_trade_no,
        total_fee,
```

```
        openid,
        trade_type: 'JSAPI',
        timeStamp: time_stamp,
    };

    // 调用 wx-js-utils 中的统一下单方法
    const {
        return_code,
        ...restData
    } = await pay.unifiedOrder(orderParam);

    let order_id = null;

    if (return_code === 'SUCCESS'&& restData.result_code === 'SUCCESS') {
        const {
            prepay_id,
            nonce_str
        } = restData;

        // 微信小程序支付要单独进地签名，并返回给小程序端
        const sign = WXPayUtil.generateSignature({
            appId: mpAppId,
            nonceStr: nonce_str,
            package: `prepay_id=${prepay_id}`,
            signType: 'MD5',
            timeStamp: time_stamp
        }, KEY);

        let orderData = {
            out_trade_no,
            time_stamp,
            nonce_str,
            sign,
            sign_type,
            body,
            total_fee,
            prepay_id,
            sign,
            status: 0, // 订单文档的status 0 未支付 1 已支付 2 已关闭
            _openid: openid,
        };

        let order = await orderCollection.add(orderData);

        order_id = order.id;
    }

    return new Res({
        code: return_code === 'SUCCESS' ? 0 : 1,
        data: { out_trade_no, time_stamp, order_id, ...restData }
    });
}
```

5. 上传云函数 在微信开发者工具中，右键点击云函数 `pay`，选取好环境后，选取好环境后，点击【创建并部署】。




## 任务二：发起支付与模板消息通知

**任务目标**: 利用云函数发起支付与发送模板消息

### 发起支付

1. 将下面代码，输入到 `client/pages/result/index.js` 中的 `pay` 方法中，通过此方法，可以调起微信支付。

```
let orderQuery = this.data.order;
let out_trade_no = this.data.out_trade_no;
let _this = this;

const {
    time_stamp,
    nonce_str,
    sign,
    sign_type,
    prepay_id,
    body,
    total_fee
} = orderQuery;

wx.requestPayment({
    timeStamp: time_stamp,
    nonceStr: nonce_str,
    package: `prepay_id=${prepay_id}`,
    signType: 'MD5',
    paySign: sign,
    async success(res) {
    wx.showLoading({
```

```
        title: '正在支付',
    });

    wx.showToast({
        title: '支付成功',
        icon: 'success',
        duration: 1500,
        async success() {
        _this.getOrder();

        await wx.cloud.callFunction({
            name: 'pay',
            data: {
            type: 'payorder',
            data: {
                body,
                prepay_id,
                out_trade_no,
                total_fee
            }
            }
        });
        wx.hideLoading();
        }
    });
    },
    fail: function (res) { }
})
```

2. 将下面代码，输入到 `cloud/functions/pay/index.js` 中的 `switch` 代码块中，通过此 `case`，可以进行进行微信支付。

```
case 'payorder': {
    // 从端里出来相关的订单相信
    const {
        out_trade_no,
        prepay_id,
        body,
        total_fee
    } = data;

    // 到微信支付侧查询是否存在该订单，并查询订单状态，看看是否已经支付成功了。
    const { return_code, ...restData } = await pay.orderQuery({
        out_trade_no
    });

    // 若订单存在并支付成功，则开始处理支付
    if (restData.trade_state === 'SUCCESS') {
        let result = await orderCollection
          .where({ out_trade_no })
          .update({
```

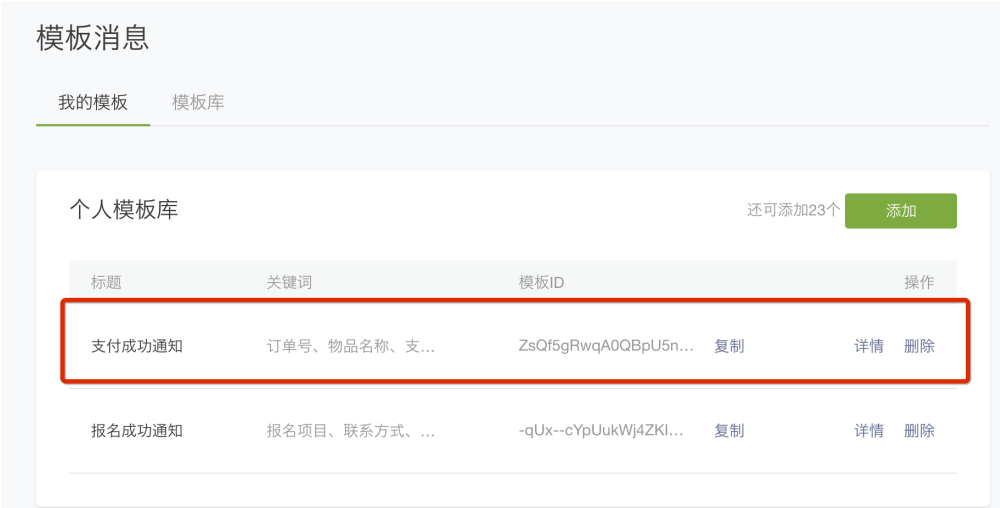
```
            status: 1,
            trade_state: restData.trade_state,
            trade_state_desc: restData.trade_state_desc
          });


        let curDate = new Date();
        let time = `${curDate.getFullYear()}-${curDate.getMonth() +
          1}-${curDate.getDate()}
${curDate.getHours()}:${curDate.getMinutes()}:${curDate.getSeconds()}`;

        // 调用另一个云函数，发送模板消息，通知用户已经支付成功了
        // 如果在实验中拿不到模板消息的模板 id，这段可以暂时去掉
        let messageResult = await app.callFunction({
          name: 'wxmessage',
          data: {
            formId: prepay_id,
            openId: userInfo.openId,
            appId: userInfo.appId,
            page: `/pages/result/index?id=${out_trade_no}`,
            data: {
              keyword1: {
                value: out_trade_no // 订单号
              },
              keyword2: {
                value: body // 物品名称
              },
              keyword3: {
                value: time// 支付时间
              },
              keyword4: {
                value: (total_fee / 100) + "元" // 支付金额
              }
            }
          }
        });

    }

    return new Res({
        code: return_code === 'SUCCESS' ? 0 : 1,
        data: restData
    });
}
```

## 模板消息通知

1. 配置小程序相关信息 新建 `cloud/functions/wxmessage/config/index.js`，并填写小程序相关配置。

```
module.exports = {
  secret: '', // 小程序 secret
  templateId: '' // 模板 id
};
```

模板 id 可以在小程序管理后台的【模板消息】-> 【我的模板】中获取，如果仍未添加，则可以在【模板库】中先添加 支付成功通知 模板。




2. 在 cloud/functions/wxmessage 目录下，运行以下命令，安装依赖。安装完毕后，在开发者工具中右键点击该目录，点击上传并创建云函数。

```
npm i --production
```

## 预览

实验完成，可以在微信开发者工具中，用手机微信扫一扫预览效果。