“Funtogether”设计原理

负责人： 王朝阳

后端： 李墒菊

前端 1： 李昌瑞

前端 2： 柴沛霖

前端 3： 郭海靖

设计： 谢婉婷

• 柏林圣康

Author:superzhaoyang

2020-12-15

目录

1 引言

1.1 研究意义

在中国存在像“美团外卖”,“饿了么”这样的线上熟人消费平台，但是并不存在线下陌生人的交友并向消费场所导流的平台。我们分析认为，此平台的流量模式具有很大的商业潜力，我们的项目基于此应运而生。

1.2 项目概述

“Funtogether”是一款主攻线下聚会娱乐的 app.。如我们的平台官网所言,“每个人都是茫茫宇宙中的星辰，聚会，让每一个星辰交汇”。我们不仅为陌生人交友娱乐提供了渠道，并通过与商家合作的方式获取返利。

1.3 项目背景

调研发现,欧美国家存在这样的线下聚会 app,而中国则缺乏相应的 app,因此此款 app 的开发对于占据消费市场具有重大意义。

1.4 项目目标

通过 app，用户可以发布组织活动或者参与活动。当报名参加某个活动后，用户可以与其他参加该活动的用户交流。此外，用户还可以通过微信平台分享活动，让更多的人了解、报名参与活动。同时该平台还会提供导航、支付等功能，以方便用户更好地投入到活动中。

1.5 项目价值

在社会价值上，该 app 能够使用户更加轻松地发现与自己具有相同爱好的群体，并与之交流、聚会。人与人之间的交流更加频繁与广泛，有利于社会和谐。在商业价值上，虽然国内的线下聚会的市场具有巨大的潜力，但国内尚未存在一款相关的 app。而此 app 的开发对于挖掘线下聚会市场的潜力和快速占据线下聚会的消费市场具有重大作用。

2 开发计划

2.1 最终呈现形式

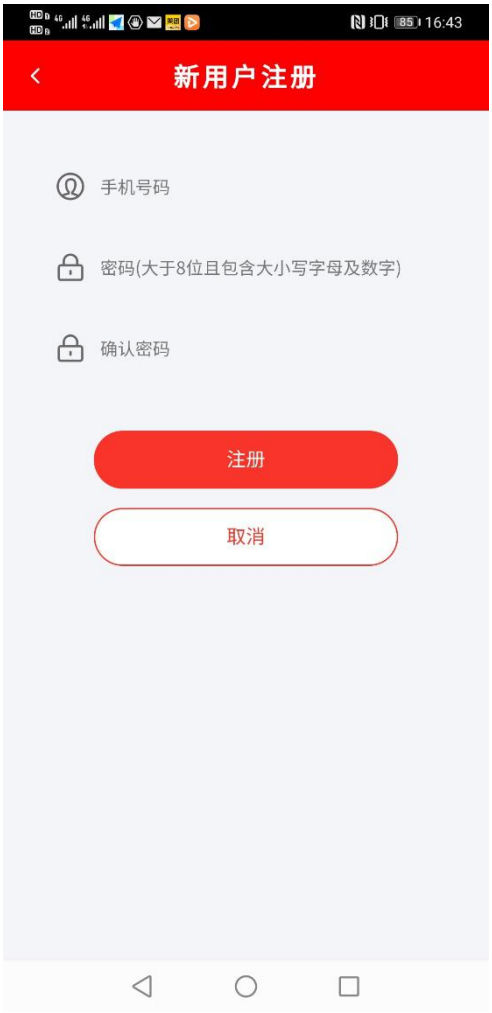
1.登陆功能：支持微信登陆和密码保存等。

2. 注册功能：包含通过短信验证码验证等。

3. 初次微信登录需要绑定账号，通过选择不同的用户类型跳转到不同的页面。

4. 如果是老用户则输入电话进行绑定。

5. 如果是新用户的话则会让用户进行注册。



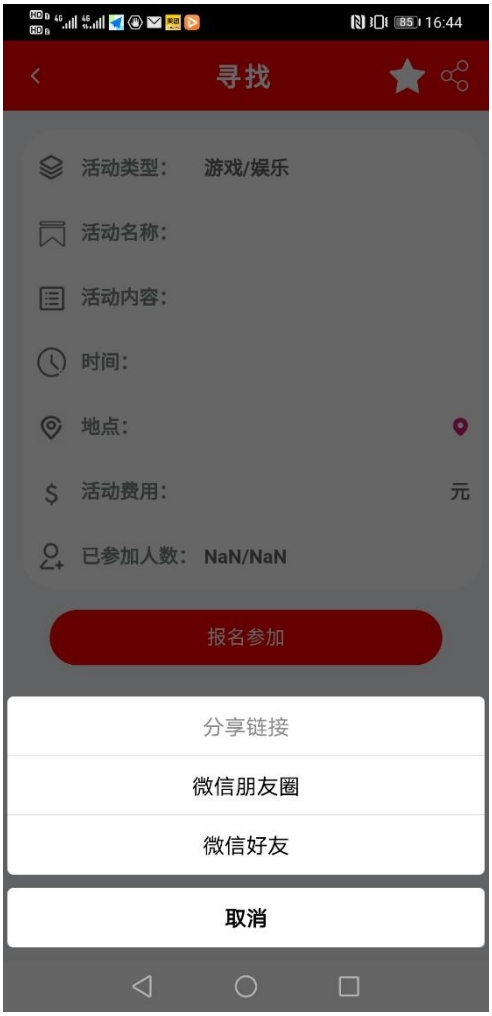
6. 对于初次登录的用户，会弹出编辑资料界使其完善信息。也可以在我的界面打开修改。

7. 寻找所有活动。


18:44
0.2K/s
寻找
游戏/娱乐
GAMES
123
157m
56
123
201m
2
学习/交流
STUDY
弔人瑞昌4...
273m
94
99
2.16km
5
户外/休闲
LEISURE
12343...
484m
58
123搜一...
818m
24
运动/健身
SPORTS
++
245m
43
9090
1.52km
5
自定义
CUSTOM
++
184m
21
弔?
194m
13
寻找
发起
聊天
我的

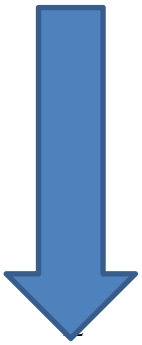
8. 查看活动详细页面并报名。如果点击右上角五角星按钮，则会将活动收藏，如果点击分享按钮，则可以将活动分享给微信好友或微信朋友圈。

9.分享活动：将活动分享给好友或朋友圈。

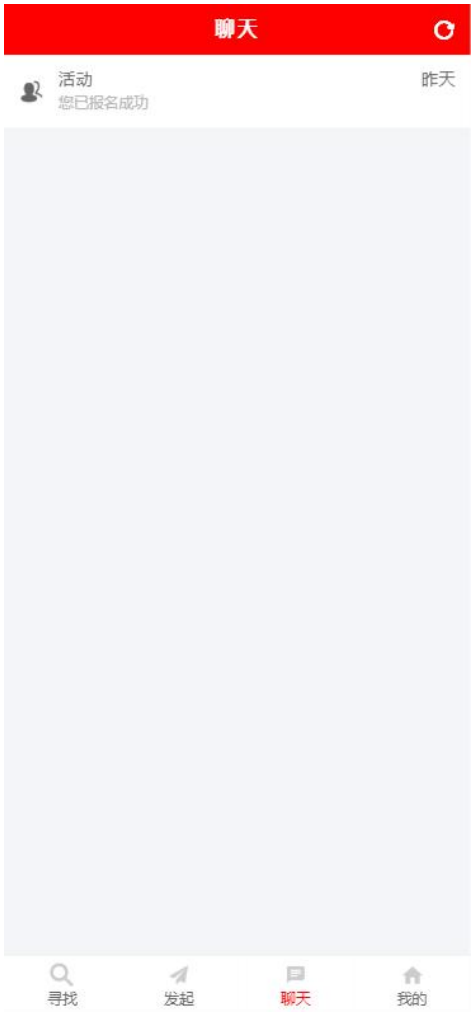


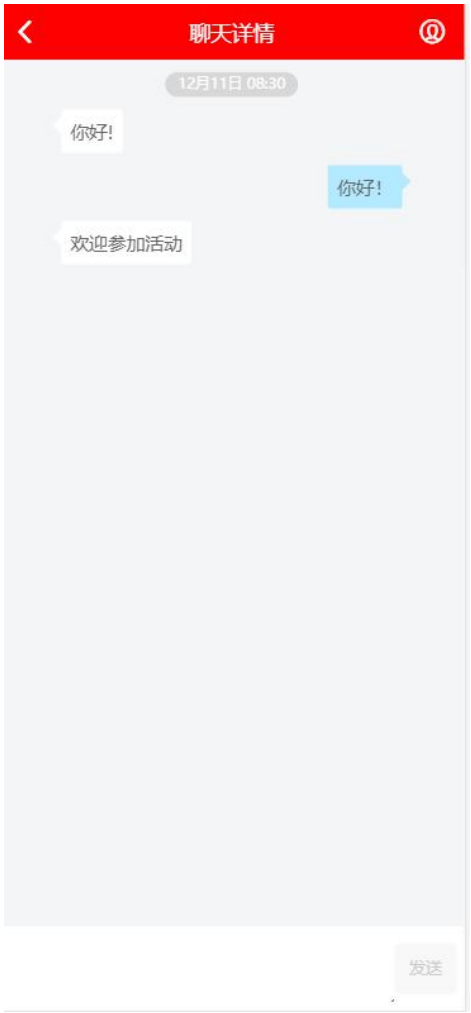
10. 发布活动：需要填写详细信息，之后进行发布

11. 利用百度地图选择发布地点：可搜索，也可以点击。



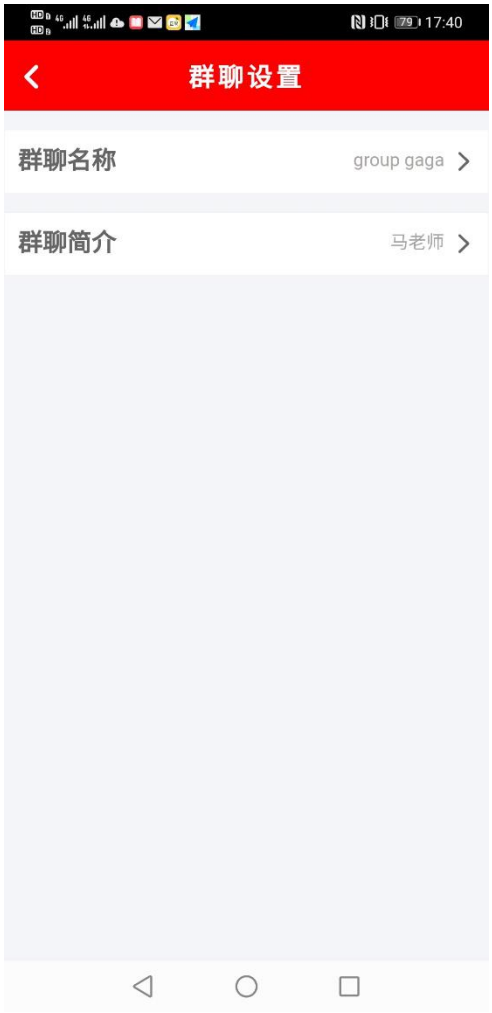
12.聊天列表，当报名成功后，便可在活动群聊天。





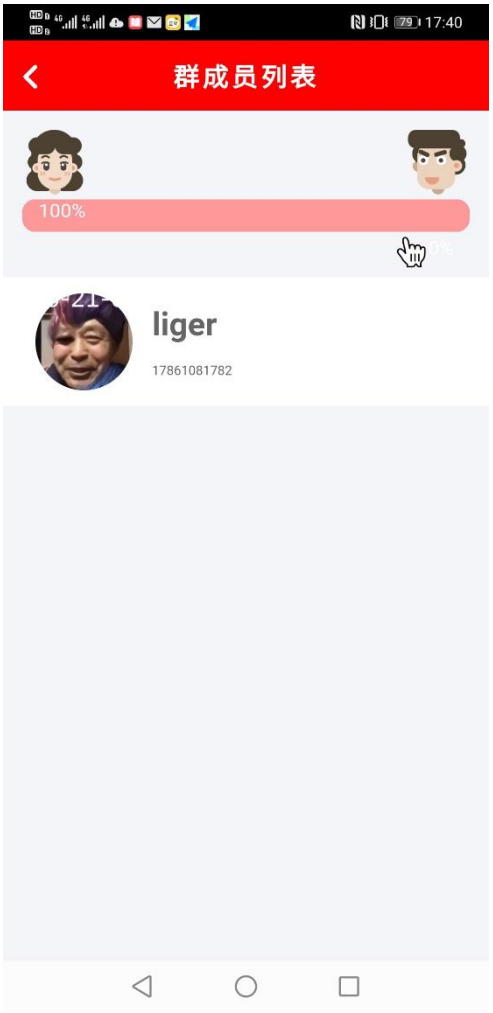
13. 聊天界面。点击右上角可查看或管理群信息。

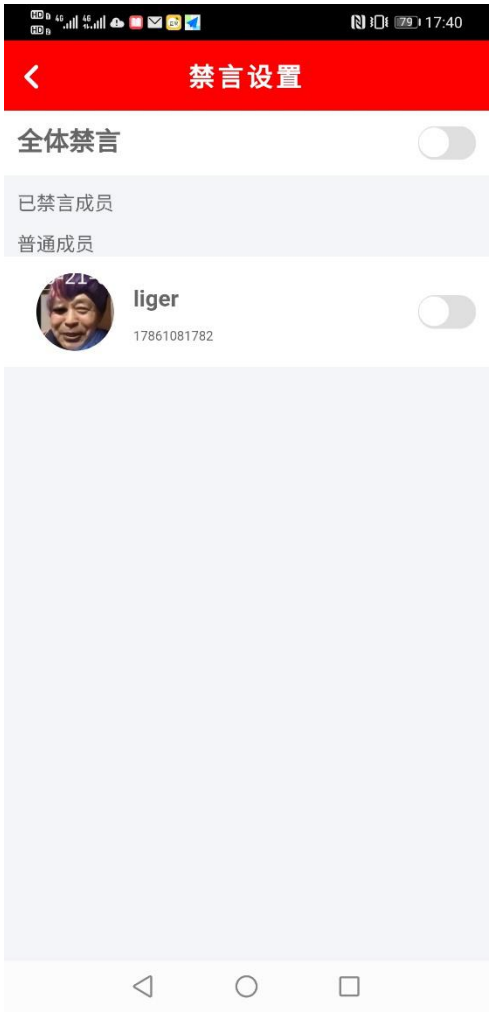
14. 群聊信息的查看或管理。包括修改群聊名称、头像、简介，禁言群成员，查看群聊成员信息等。



15. 修改群聊名称、简介。

16. 查看群成员信息，包括性别比例等。





17. 禁言群成员功能。

18. 我的界面：可以查看收藏的活动和参与的活动；编辑个人资料；个人账户管理等。

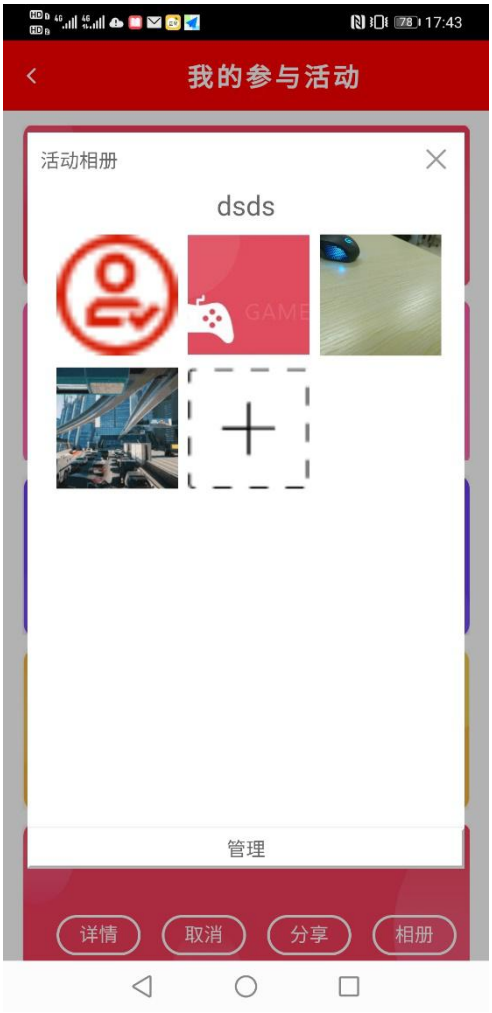

17:43
我的
coc
查看或更改个人资料
2693.20
提现
我的钱包
3
我的收藏活动
36
我的参与活动
我的荣耀系统
版本更新
查看头像
更改头像
取消
寻找
发起
聊天
我的

19. 设置个人头像

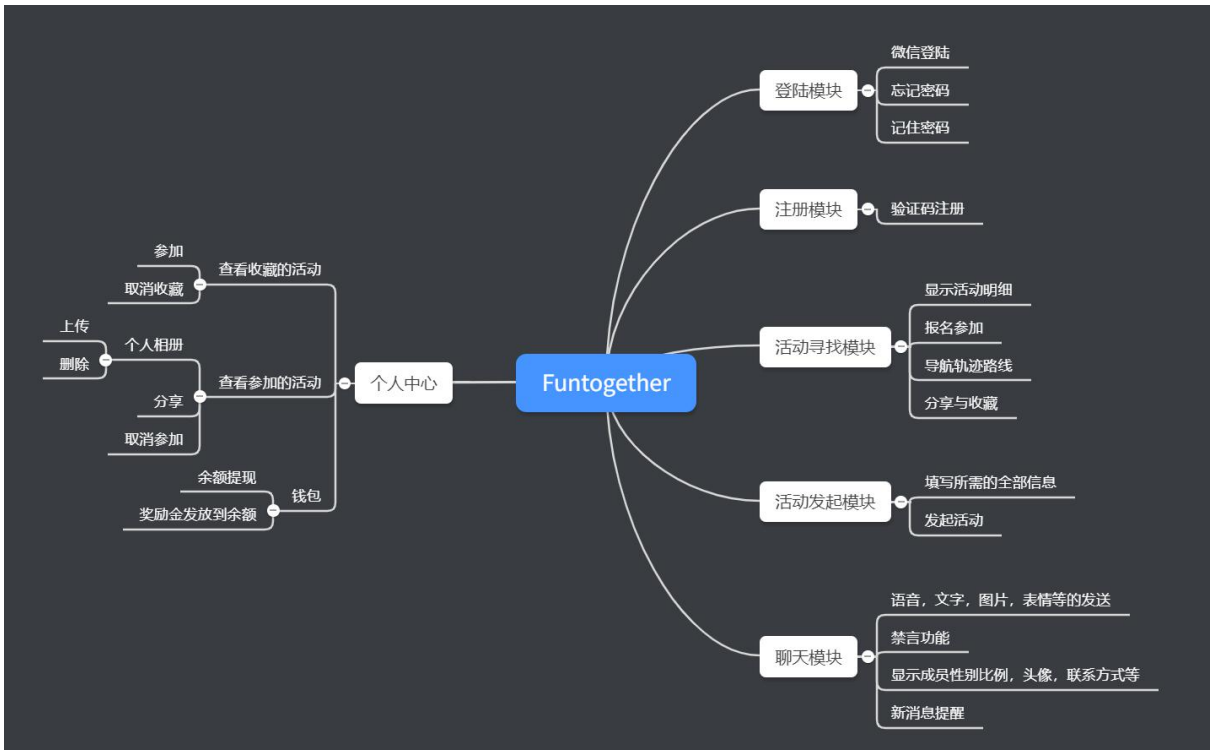
20.我的收藏活动界面，可以将活动分享到微信或者取消收藏。

21. 我的参与活动界面，包括查看活动详情、取消活动报名、分享活动、上传或查看相册等。

22. 活动相册界面，包括查看照片或上传相册等



2.2 主要功能描述



2.3 运行环境

- 安卓 > 4.0
- ios > 9.0
- 浏览器端

2.4 验收标准

- 无逻辑 bug
- 可承受 200 人左右的并发量

3 前端设计概要

3.1 Bui 框架

3.1.1 Bui 介绍

BUI 是用来快速构建界面交互的 UI 框架，专注 webapp 开发，开发者只需关注业务的开发，界面的布局及交互交给 BUI，开发出来的应用，可以嵌入平台

(微信公众号, 微信小程序 webview, 聆客, 钉钉, 淘宝, 支付宝等), 亦可以跟其它第三方平台打包成独立应用(Bingotouch , Cordova , Dcloud , APICloud , Appcan 等), 最终可以全跨平台展示。(包括 Ipad)

BUI 支持两种开发方式, 多页开发跟单页开发基本保持一致。本 app 采用单页开发, 单页开发不受平台限制,保持一致交互体验。

在 BUI 官网下载 BUI 单页开发包, 解压,进入工程目录, 通过这种运行, app.json 可以配置域名代理,解决移动端调试的跨域问题. 生成的 dist 目录为最终要打包的目录, src 目录保持源文件的方式。

3.1.2 路由初始化

打开编辑 src/index.html , body 下只有一个 div,这个便是路由最外层结构.

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
    <title>BUI单页工程</title>
    <meta name="format-detection" content="telephone=no" />
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">
    <link rel="stylesheet" href="css/bui.css" />
</head>

<body>
    <!-- 第1步：开启单页路由 -->
    <div id="bui-router"></div>
    <script src="js/zepto.js"></script>
    <script src="js/bui.js"></script>
    <!-- 初始化单页 -->
    <script src="index.js"></script>
</body>

</html>
```

src/index.js为加载 **index.html** 的 **js** 文件，可进行 **app** 相关的事件类定义，本 app 绑定页面的所有按钮有 href 跳转以及统一绑定页面所有的后退按钮。

```
window.router = bui.router();


bui.ready(function() {
    // 初始化路由
    router.init({
        id: "#bui-router",
        progress: true,
        hash: true,
    })

    // 绑定事件
    bind();

    // 事件类定义
    function bind() {
        // 绑定页面的所有按钮有href跳转
        bui.btn({ id: "#bui-router", handle: ".bui-btn" }).load();

        // 统一绑定页面所有的后退按钮
        $("#bui-router").on("click", ".btn-back", function(e) {
            // 支持后退多层,支持回调
            bui.back();
        });

    }


})
```

*

当路由初始化以后,会自动查找首页 main 模块，这个模块是内部定义好的，默认指向路径 pages/main/main.html。本 app main.html 页面为登录页面，关于本 app 的 html 文件采用的单页模板，在此进行简单介绍，此处之外不再赘述。

单页模板不需要引入一堆脚本样式，跟组件模板一致，就是一个简单 html 结构.

单页模板的命名跟模块的命名默认保持一致，路径一致的方式。以下是简单例子

```
<div class="bui-page bui-box-vertical">
    <header>
        <!-- 固定顶部区 -->
        <div class="bui-bar">
            <div class="bui-bar-left">
                <a class="bui-btn" onclick="bui.back();"><i class="icon-back"></i></a>
            </div>
            <div class="bui-bar-main">单页模板</div>
            <div class="bui-bar-right">
            </div>
        </div>
    </header>
    <main>
      <!-- 固定中间滚动内容区 -->
    </main>
    <footer>
      <!-- 固定底部   -->
    </footer>
</div>
```

此外，模板里面可以增加 css 样式，只需写在最上方即可

```
<style>
  .bui-page .bui-bar {
    background:red;
  }
</style>
```

此处需注意，由于 bui.page 是全局 class，像上面的写法会影响全局. 应该在 bui-page 加多一个独有的样式. 才能避免相互影响，例如：

```
<style>
  .page-home .bui-bar {
    background:red;
  }
  .page-home main {
    background:#ddd;
  }
</style>
```

```
<div class="bui-page bui-box-vertical page-home">
```

关于 bui 的各种布局以及各种控件用法，例如滑动交互控件、点击控件等，此处不再进行介绍，可在 bui 官网进行查阅并学习使用即可。

跳转 main.html，自动加载相同名字的 js 文件。单页开发所有组件的使用,默认都是在模块 loader.define 里面。关于 loader.define，loader.define 定义一个匿名模块,一个 js 文件里面只能有一个 loader.define 的匿名模块，即：

```
loader.define(function(require,exports,module){

    // 以下几个参数非必须,如果前面加载了依赖,则这三个参数后移;
    // require : 相当于 loader.require, 获取依赖的模块
    // exports : 如果没有return 可以采用这种方式输出模块
    // module : 拿到当前模块信息

    // 第一次加载会执行一次

    // 模块如果需要给其它模块加载,通过 return 的方式抛出来
    return {};
})
```

本 app 的 js 文件基本采用以下结构

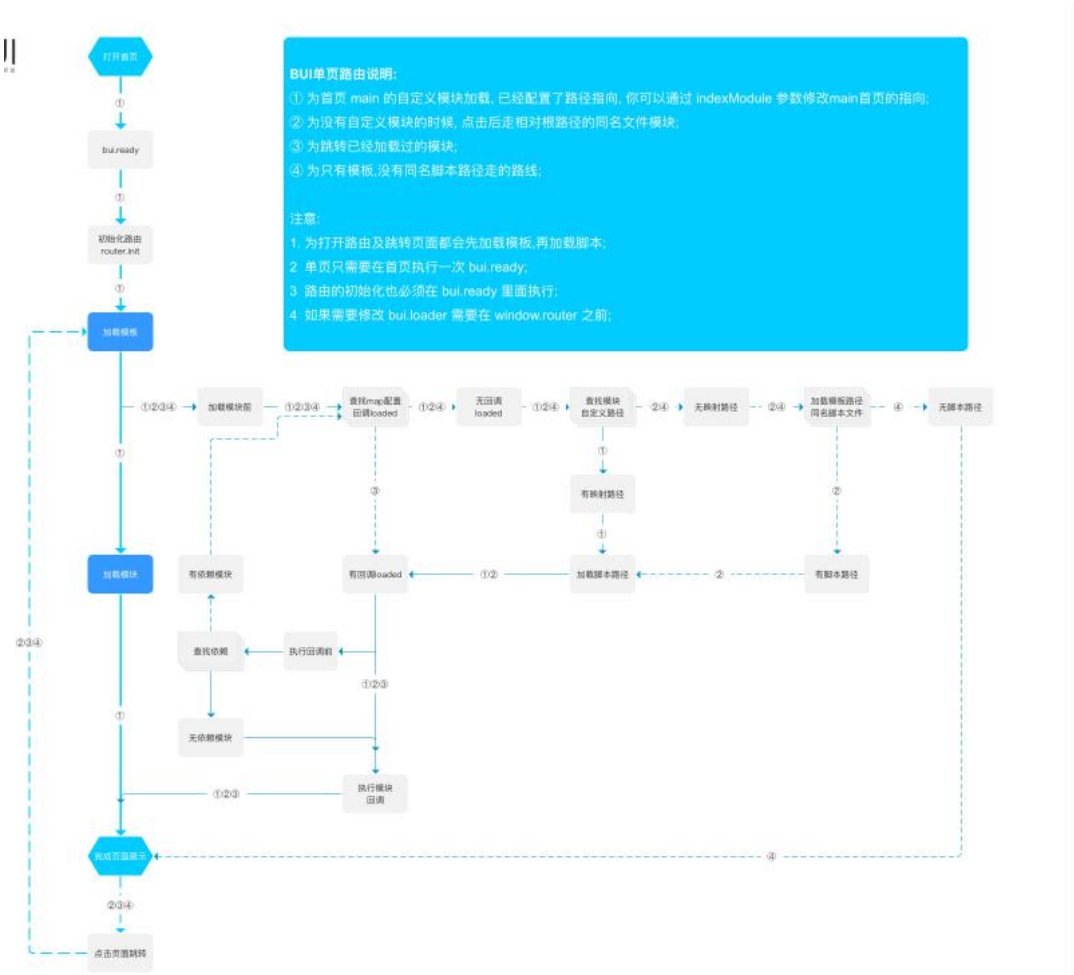
```
loader.define({
    depend: ['pages/config/config'],
    loaded: function(config,require,exports,module,global){


        this.pageview = {
            init: function(e){

            },

    };

        this.pageview.init()
        return this.pageview;
    }
})
```

利用 depend 可以加载出其他模块，但要加载的模块需要通过 return 的方式抛出来，

例如本 app 的 config 模块：

```
loader.define(function(require,exports,module,global){

    var pageview = {};


    pageview.apiurl = "";

    return pageview;
})
```

在 js 文件中即可通过 config.apiurl 获取 config 模块的 apiurl 内容。



此图更好地展示路由加载的完整过程.

除了 **main**, 正常我们都是创建匿名模块,这样只要通过路径跳转,就会自动加载相对路径同名的模块。关于页面跳转，我们将在下面介绍。

3.1.3 页面跳转

```
bui.load({
    url: "",
    param:{}
})
```

其中 **url** 为请求的地址，这里必须为相对路径，**params** 为传过去另外页面的参数，本 **app** 基本采用这种页面跳转方式。

例：

```
bui.load({url:"pages/personalcontent/personalcontent",
param: {
    flag:"2",
    personlocation:$("#mine_my_big_part2_div1").attr('value')
}
```

接收参数：

```
var getParams = bui.getPageParams();
getParams.done(function(result){
    var personlocation=result['personlocation']
    var flag=result['flag']
})
```

3.1.4 页面后退

bui.back();很好的完成页面后退。

```
<a class="bui-btn" onclick="bui.back();"><i class="icon-back"></i></a>
```

3.1.5 数据请求

```
bui.ajax({
    url: "",
    data: {}
}).then(function(res){
    // 成功回调
    console.log(res)
},function(res,status){
    // 失败回调
    console.log(status);
})
```

本 app 采用 bui.ajax(option)与服务器进行数据交互。

例：

```
bui.ajax({
    url: config.apiurl +"api/v1.0/mychange",
    data: {},
    method: "post",
    headers: {
        'Content-Type': 'application/json'
    }
    }).done(function(res){
        console.log(res);

  }).fail(function(res,status){
      console.log(status);
      // status = "timeout" || "error" || "abort", "parsererror"
      })
```

需和服务端维护人员商议出传输的数据字段，以成功的完成数据交互。

3.1.6 数据存储

bui.storage(option)，bui.storage 是基于 localStorage 及 sessionStorage 封装的，主要解决两者之间的 API 统一问题，并且支持 JSON 存储，以及支持限制多少条数据等问题，常用来做历史记录，默认返回的是一个数组。本 app 主要是用 bui.storage 实现记住用户账号密码功能。
例：

```
var storage = bui.storage();
pageview.bs = {

    islogin: function(){
        if(storage.get("islogin", 0) == true)
            return true;
        return false;
    },
    getuser: function(){
        var data = {
            mobile: storage.get("user", 0),
            password: storage.get("password", 0)
        }
        return data;
    },
    saveuser: function(user, password){
        storage.remove("user");
        storage.remove("password");
        storage.set("user", user);
        storage.set("password", password);
    },
    clearuser: function(){
        storage.clear();
    },
    login: function () {
        storage.remove("islogin");
        storage.set("islogin", true);
    },
    logout: function(){
        storage.remove("islogin");
        storage.set("islogin", false);
    },
    save: function () {
        storage.remove("issave");
        storage.set("issave", true);
    },
    unsave: function () {
        storage.remove("issave");
        storage.set("issave", false);
    },
    issave: function(){
        if(storage.get("issave", 0) == true)
            return true;
        return false;
    }
}
```

3.1.7 相关控件

关于本 app 使用的相关控件，在此处举几例进行简单介绍

下拉菜单控件

例如发起活动界面的下拉框

```
<div id="uiDropdown" class="bui-dropdown">
  <div class="bui-btn bui-box my_box" >
    <div class="span1" id="type_name"><!--下拉菜单填写处--></div>
      <i class="icon-dropdown" id="select_dropdown"></i>
  </div>
</div>
```

相关的 js 代码

```
//实现下拉列表
var uiDoropdown = bui.dropdown({
    id: "#uiDropdown",
    data: [{
            name:"自定义",
        value:"自定义"
    },{
            name:"游戏/娱乐",
        value:"游戏/娱乐"
    },{
            name:"户外/休闲",
        value:"户外/休闲"
    },{
            name:"学习/交流",
        value:"学习/交流"
    },{
            name:"运动/健身",
        value:"运动/健身"
    }],
    //设置relative为false,二级菜单继承父层宽度
    relative: false,
    callback: function (e) {
    }
});
```

下拉刷新控件

例如我的参与活动的下拉刷新

```
<div id="scrolllist" class="bui-scroll">
    <div class="bui-scroll-head"></div>
    <div class="bui-scroll-main">
        <ul class="bui-list bui-list-thumbnail">



        </ul>
    </div>
    <div class="bui-scroll-foot"></div>
</div>
```

相关 js 代码

```
//滑动页面布局加载
var uiList = bui.list({
    id: "#scrollList",
    url: config.apiurl + "api/v1.0/myact",
    pageSize: 1000000, // 当pageSize 小于返回的数据大小的时候,则认为是最后一页,接口返回的数据最好能返回空数组,而不是null
    data: {},
    //如果分页的字段名不一样,通过field重新定义
    field: {
        page: "page",
        size: "pageSize",
        data: "activities"
    },
    template: function(data) {

    },
    onBeforeRefresh: function() {
        console.log("brefore refresh")
    },
    onBeforeLoad: function() {
        console.log("brefore load")
    },
    onRefresh: function() {
        // 刷新以后执行
        startButton();
        console.log("refreshed")
    },
    onLoad: function() {
        // 刷新以后执行
        startButton();
        console.log("loaded")

    }
```

对话框控件

例如相册弹出对话框

```
<!-- 中间自定义弹出框结构    -->
<div id="dialogCenter" class="bui-dialog" style="display: none;">
    <div class="bui-dialog-head">活动相册</div>
    <div class="bui-dialog-main">
        <h1 id="activity_name"></h1>
        <div id="buiPhoto" class="bui-upload bui-fluid-space-4">

            <div class="container clearfix" id="panelfix">

                <div class="img-listsadd" id="add">
                    <img src="images/home/add_photo.png" multiple/>
                </div>

            </div>


        </div>
    </div>
    <div class="bui-dialog-foot">
        <div class="bui-box">
            <div class="span1" id="delpanel">
                <button class="del_btn1" id="delmanager">管理</button>
            </div>
        </div>
    </div>
    <div class="bui-dialog-close"><i class="icon-close"></i></div>
</div>
```

相关 **js** 代码

```
// 自定义居中弹出框
var uiDialog = bui.dialog({
    id: "#dialogCenter",
    height: 400,
    fullscreen:true,
    style: { left: "5%",right:"5%", top: "10%",bottom:"10%"},
    // mask: false,
    callback: function(e) {
        console.log(e.target)
    },
    onClose:function(){
        $("#delpanel").html(`<button class="del_btn1" id="delmanager">管理</button>`)
        $list_container.html(previous);
    }
});
uiDialog.open();
```

底部弹出菜单控件

例如点击头像事件

相关 js 代码

```
// 上拉菜单 js 初始化:
var uiActionsheet = bui.actionsheet({
    trigger: "#mine_my_big_part1",
    appendTo: "#home",
    buttons: [{ name: "查看头像", value: "check" }, { name: "更改头像", value: "photo" }],
    callback: function(e) {
        var ui = this;
        var val = $(e.target).attr("value");

        switch (val) {

            case "photo":


                break;
            case "check":


                break;
            case "cancel":
                ui.hide();
                break;
        }
    }
})
```

注：此块内容对 BUI 进行简单概述，以及对 app 中的部分操作以及些许细节进行介绍以便读者理解，具体请参照项目源码和 BUI 官网进行解读。

3.2 百度地图 api

本 app 采用百度地图提供的 **JavaScript API** 实现 app 中地图部分的功能。

百度地图 JavaScript API 是一套由 JavaScript 语言编写的应用程序接口，可帮助您在网站中构建功能丰富、交互性强的地图应用，支持 PC 端和移动端基于浏览器的地图应用开发,且支持 HTML5 特性的地图开发。

JavaScript API GL v1.0 是基于 WebGL 全新开发的地图 API 接口，包含了 3D 地图的渲染、基本控件、覆盖物。但对手机性能要求较高，故我们选择 JavaScript API v3.0 版本。

针对于本 app 的特点与百度题图 api 提供的服务，我们实现了：

1. **活动详情：查看活动地理位置，显示该地点的相关信息**
   - **实现效果：**

- **具体功能：**

1. **为用户提供当前位置与活动地点的距离**

实现：结合百度 JavaScript API 提供在 Web 端获取当前位置的方法，获取当前位置。并根据活动地点经纬度信息计算距离。

相关函数：

- `BMap.Geolocation()`类中 `getCurrentPosition(function(r){})`函数获取用户当前位置

```
var map = new BMap.Map("allmap");
var point = new BMap.Point(116.331398,39.897445);
map.centerAndZoom(point,12);

var geolocation = new BMap.Geolocation();
geolocation.getCurrentPosition(function(r){
if(this.getStatus() == BMAP_STATUS_SUCCESS){
        var mk = new BMap.Marker(r.point);
        map.addOverlay(mk);
        map.panTo(r.point);
        alert('您的位置: '+r.point.lng+','+r.point.lat);
    }
    else {
        alert('failed'+this.getStatus());
    }
});
```

- BMap.Map()类中
  getDistance(point1, point2)函数
  根据两点经纬度，得到两点的距离

```
var distance = map.getDistance(point1,point2)
```

2. **显示活动地点相关信息（活动地点名称，活动地点地址等）**

实现：结合百度 JavaScript API 提供的检索 POI 服务。检索服务提供某一特定地区的兴趣点查询服务(POI:Point of Interest，感兴趣点)。允许设置检索城市、检索圆形区域 POI、检索结果详情。

相关函数：

- BMap.LocalSearch()类中
  search()函数
  根据活动地点名获得检索结果详情

```
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(116.404, 39.915), 11);
var options = {
    onSearchComplete: function(results){
        if (local.getStatus() == BMAP_STATUS_SUCCESS){
            // 判断状态是否正确
            var s = [];
            for (var i = 0; i < results.getCurrentNumPois(); i ++){
                s.push(results.getPoi(i).title + ", " +
results.getPoi(i).address);
            }
            document.getElementById("log").innerHTML = s.join("<br>");
        }
    }
 };
var local = new BMap.LocalSearch(map, options);
local.search("公园");
```

3. 为用户显示活动地点在地图上的位置

相关函数：

- BMap.Marker()类创建点

```
// 向地图添加标注
var map = new BMap.Map("container");
var point = new BMap.Point(116.404, 39.915);
map.centerAndZoom(point, 15);
var marker = new BMap.Marker(point);        // 创建标注
map.addOverlay(marker);                     // 将标注添加到地图中
```

```
// 定义标注图标
var map = new BMap.Map("container");
var point = new BMap.Point(116.404, 39.915);
map.centerAndZoom(point, 15);  // 编写自定义函数，创建标注
function addMarker(point, index){  // 创建图标对象
    var myIcon = new BMap.Icon("markers.png", new BMap.Size(23, 25), {
        // 指定定位位置。
        // 当标注显示在地图上时，其所指向的地理位置距离图标左上
        // 角各偏移10像素和25像素。您可以看到在本例中该位置即是
        // 图标中央下端的尖角位置。
        anchor: new BMap.Size(10, 25),
        // 设置图片偏移。
        // 当您需要从一幅较大的图片中截取某部分作为标注图标时，您
        // 需要指定大图的偏移位置，此做法与css sprites技术类似。
        imageOffset: new BMap.Size(0, 0 - index * 25)   // 设置图片偏移
    });
    // 创建标注对象并添加到地图
    var marker = new BMap.Marker(point, {icon: myIcon});
    map.addOverlay(marker);
}
// 随机向地图添加10个标注
var bounds = map.getBounds();
var lngSpan = bounds.maxX - bounds.minX;
var latSpan = bounds.maxY - bounds.minY;
for (var i = 0; i < 10; i ++) {
    var point = new BMap.Point(bounds.minX + lngSpan * (Math.random() * 0.7
+ 0.15),
                                    bounds.minY + latSpan * (Math.random() *
0.7 + 0.15));
    addMarker(point, i);
}
```

```
// 监听标注事件
marker.addEventListener("click", function(){
    alert("您点击了标注");
});
```

```
// 可拖拽的标注
marker.enableDragging();
marker.addEventListener("dragend", function(e){
    alert("当前位置: " + e.point.lng + ", " + e.point.lat);
})
```

- BMap.Map()类创建地图
  addOverlay()函数添加点

```
map.addOverlay(marker);                    // 将标注添加到地图中
```

2. 利用百度地图选择发布地点：可以搜索，也可以点击

- 实现效果：

- 具体功能：

1. 用户通过触发点击事件，选择发布活动的地点

实现：结合百度 JavaScript API 提供的检索 POI 服务。为用户显示选中地点的详情信息

相关函数：

- `BMap.GeoCoder()`类中 `getLocation()`函数 获取点击点详细 POI 信息

```
var get_address=new BMap.Geocoder();
get_address.getLocation(new BMap.Point(e.point.lng-0.00005,e.point.lat-0.00005),
function(m){
    //获取详细POI信息
    var t=m.surroundingPois;
```

2. **用户通过搜索框搜索活动地点，选择发布活动地点**

实现：结合百度 JavaScript API 提供的检索服务。为用户显示解锁结果

相关函数：

- BMap.LocalSearch()类中
  search()函数
  根据活动地点名获得检索结果详情

4 后端设计概要

4.1 服务器框架

4.1.1 Flask 简介

(1) **Flask 是使用 Python 编写的轻量级 Web 应用框架，基于 WerkzeugWSGI 工具箱和 Jinja2 模板引擎。**使用 BSD 授权。Flask 也被称为“microframework”，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。然而，Flask 保留了扩增的弹性，可以用 Flask-extension 加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

(2) **Flask 是目前十分流行的 web 框架，采用 Python 编程语言来实现相关功能。**它被称为微框架(microframework)，“微”并不是意味着把整个 Web 应用放入到一个 Python 文件，微框架中的“微”是指 Flask 旨在保持代码简洁且易于扩展，Flask 框架的主要特征是核心构成比较简单，但具有很强的扩展性和兼容性。

2. **后端设计**

(1) Flask 自身不会提供表单验证功能，在项目实施过程中可以自由配置，从而为应用程序开发提供数据库抽象层基础组件，支持进行表单数据合法性验证、文件上传处理、用户身份认证和数据库集成等功能。

(2) Flask 主要包括 Werkzeug 和 Jinja2 两个核心函数库，它们分别负责业务处

理和安全方面的功能，这些基础函数为 web 项目开发过程提供了丰富的基础组件。Werkzeug 库十分强大，功能比较完善，支持 URL 路由请求集成，一次可以响应多个用户的访问请求；支持 Cookie 和会话管理，通过身份缓存数据建立长久连接关系，并提高用户访问速度；支持交互式 Javascript 调试，提高用户体验；可以处理 HTTP 基本事务，快速响应客户端推送过来的访问请求。

(3) Jinja2 库支持自动 HTML 转移功能，能够很好控制外部黑客的脚本攻击。系统运行速度很快，页面加载过程会将源码进行编译形成 python 字节码，从而实现模板的高效运行；模板继承机制可以对模板内容进行修改和维护，为不同需求的用户提供相应的模板。目前 Python 的 web 框架有很多。除了 Flask，还有 django、Web2py 等等。和其他的轻量级框架相比较，Flask 框架有很好的扩展性，这是其他 Web 框架不可替代的。

(4) 本项目为基于 Flask Web 框架开发的线下交友活动的软件平台，后端采用 Flask-SQLAlchemy 扩展连接 MySQL 数据库与 redis 数据库对用户的个人信息、活动信息及聊天详情信息等模块进行存储，通过增删查改等基本操作与前端进行交互，实时传递信息。

4.2 数据库设计

4.2.1 Mysql 介绍

(1) MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。

(2) MySQL 是一种关系型数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

(3) MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

4.2.2 Redis 介绍

(4) redis 是一个 key-value 存储系统，它支持存储的 value 类型相对更多，包括 string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和 hash（哈希类型）。这些数据类型都支持 push/pop、add/remove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis 支持各种不同方式的排序。与 memcached 一样，为了保证效率，数据都是缓存在内存中。区别的是 redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且

在此基础上实现了 master-slave(主从)同步。

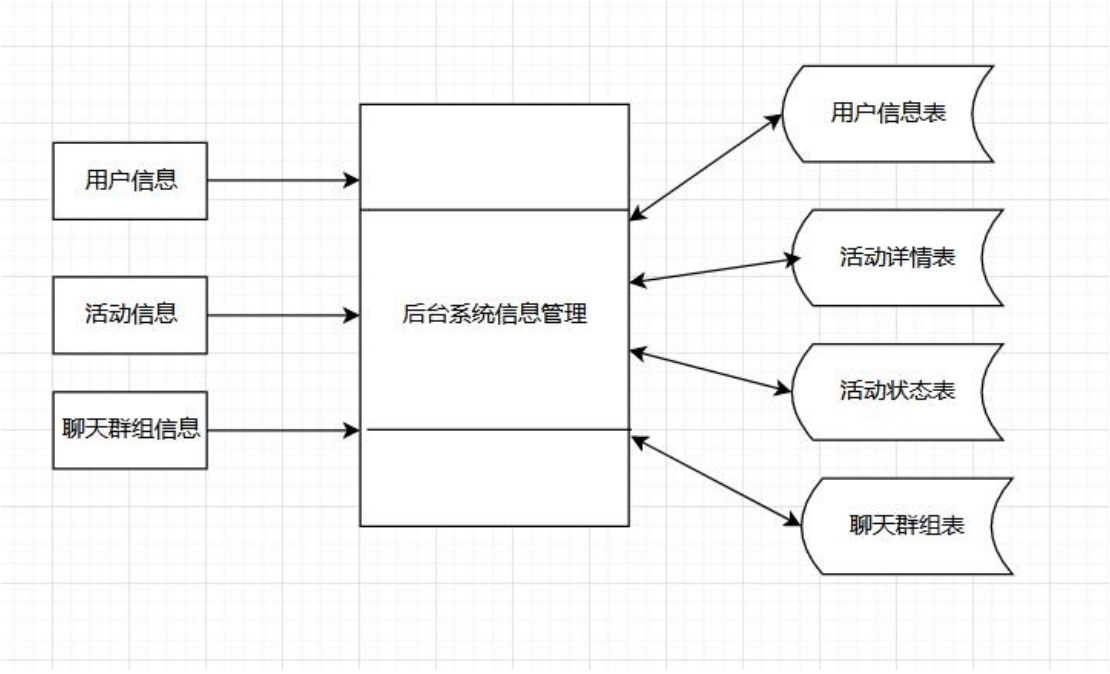
(5) Redis 是一个高性能的 key-value 数据库。 redis 的出现，很大程度补偿了 memcached 这类 key/value 存储的不足，在部 分场合可以对关系数据库起到很好的补充作用。它提供了 Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang 等客户端，使用很方便。

(6) Redis 支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得 Redis 可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

4.2.3 本项目后端数据库设计

本项目后端采用 Flask-SQLAlchemy 扩展连接 MySQL 数据库与 redis 数据库进行信息的存储处理。

（1）数据流图



（2）数据结构

数据结构	含义说明	组成
用户信息	包含用户注册信息、个人账号详细信息	用户编号，名称，电话号码，密码，头像，地址，性别，年龄，微信id
参与活动信息	包含用户参与的活动信息	活动编号，参与时间，参与人数，参与者编号
收藏活动信息	包含用户收藏的活动信息	活动编号，参与时间，参与人数，收藏者编号
活动发起信息	包含用户发起活动的详细信息	活动编号，发起时间，人数限制，发起者编号
聊天群组信息	包含聊天群组的详情信息	群组编号，创建时间，参与人员，参与者总数

（3）数据项

数据项	含义说明	类型	长度	备注
用户编号	唯一标识用户	int	32位整数大小	
用户名称	表示用户名称	string	32位	
用户电话	用户电话号码	string	11位	
用户密码	用户注册登录密码	string	128位	hash操作
openid	唯一标识用户微信	string	40	
活动编号	唯一标识活动	int	32位整数大小	
活动类型	标识活动类别	int	32位整数大小	
活动地址	标识活动地址	string	128位	
照片路由	照片路径	string	128位	
群组编号	唯一标识群组	int	32位整数大小	
群组头像路由	群组头像路径	string	128位	

（4）数据存储

表名称	字段信息	数据类型	是否主键	外键设置
User	id	int	是	
	name	string		
	password	string		
	mobile	string		
	avatar_url	string		
	account	int		
	favorite_count	int		
	sex	string		
	introduction	string		
	loc	string		
	age	string		
	openid	string		

表名称	字段信息	数据类型	是否主键	外键设置
ActivityType	id	int	是	
	type	string		

表名称	字段信息	数据类型	是否主键	外键设置
Activity	id	int	是	
	name	string		
	typeid	string		ActivityType.id
	pageview	int		
	cost	int		
	peoplecount	int		
	address	string		
	time	string		
	content	string		
	uid	string		
	detailaddress	string		
	longitude	float		
	latitude	float		
	city	string		
	peoplesurplus	int		

表名称	字段信息	数据类型	是否主键	外键设置
Glory	userid	int	是	User.id
	favorite_act_id	int	是	Activity.id
	score	int		
表名称	字段信息	数据类型	是否主键	外键设置
UserActivity	userid	int	是	
	actid	int	是	
	score			
表名称	字段信息	数据类型	是否主键	外键设置
Photo	userid	int	是	User.id
	actid	int	是	Activity.id
	photo_url	string	是	
表名称	字段信息	数据类型	是否主键	外键设置
Group	groupid	int	是	
	leaderid	int		
	groupname	string		
	avatar_url	string		
	act_id	int		
	introduction	string		
表名称	字段信息	数据类型	是否主键	外键设置
Member	userid	int	是	
	groupid	int	是	
	name	string		

表名称	字段信息	数据类型	是否主键	外键设置
FavouriteActivity	userid	int	是	User.id
	actid	int	是	Activity.id
	is_involved	boolean		
表名称	字段信息	数据类型	是否主键	外键设置
InvolveActivity	userid	int	是	User.id
	actid	int	是	Activity.id

5 相关技术依赖

融云：第三方聊天 api,官网：https://www.rongcloud.cn/
容连云：第三方及时短信 api，官网：https://www.yuntongxun.com/
百度地图:第三方地图 api，官网： http://lbsyun.baidu.com/

6 运行及打包方法

6.1 前端

6.1.1 Bui 如何运行项目

1. **简述：** buijs 是基于 zepto 框架(一款轻量级并且类似 jQuery 框架的 JavaScript 框架)的 JavaScript 框架。在运行 bui 程序之前需要先安装好 buijs 框架。

2. **准备工作：**需要再 node 环境下进行安装 buijs，所以需要先安装好 nodejs 并配置好环境变量，这一点可以参照网上博客。

3. **安装：**安装好 nodejs 后，在 Windows 下使用管理员模式打开终端，具体操作是按键盘的 win+s 键搜索 cmd,点击以管理员模式运行。输入 npm install -g buijs。在 Mac 环境下打开终端，输入 sudo npm install -g buijs。这一步是安装 buijs，需要花些时间等待安装。

```
D:\>npm install -g buijs
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
D:\install\Nodejs\node_global\buijs -> D:\install\Nodejs\node_global\node_modules\buijs\buijs-cli.js
+ buijs@1.6.40
added 290 packages from 143 contributors in 105.593s
```

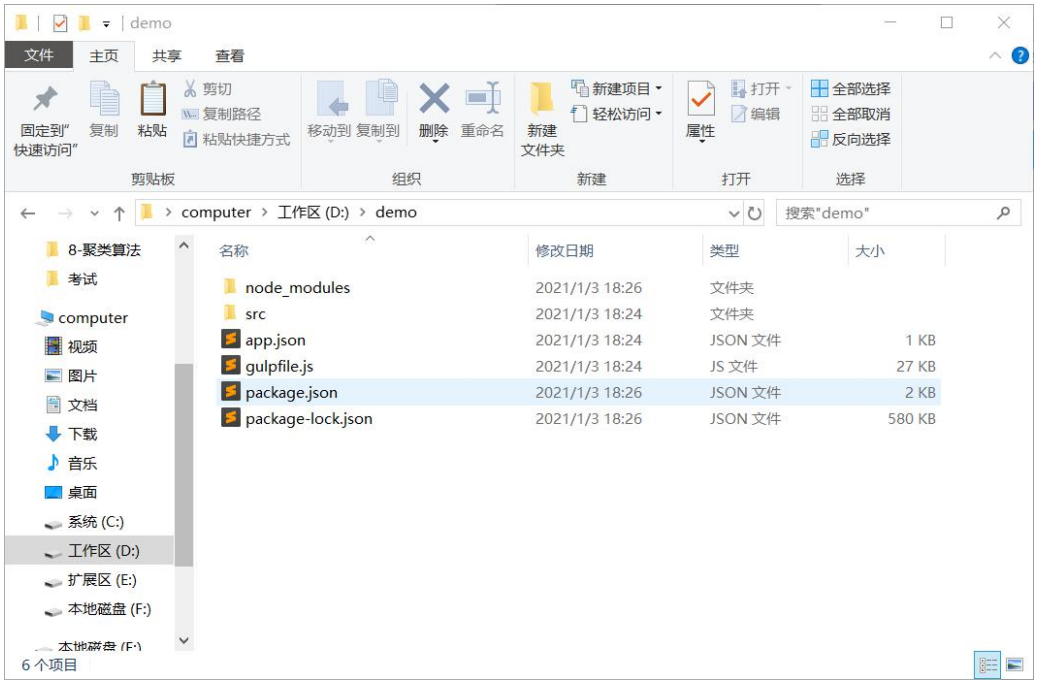
注：如果出现了停顿，按回车键即可

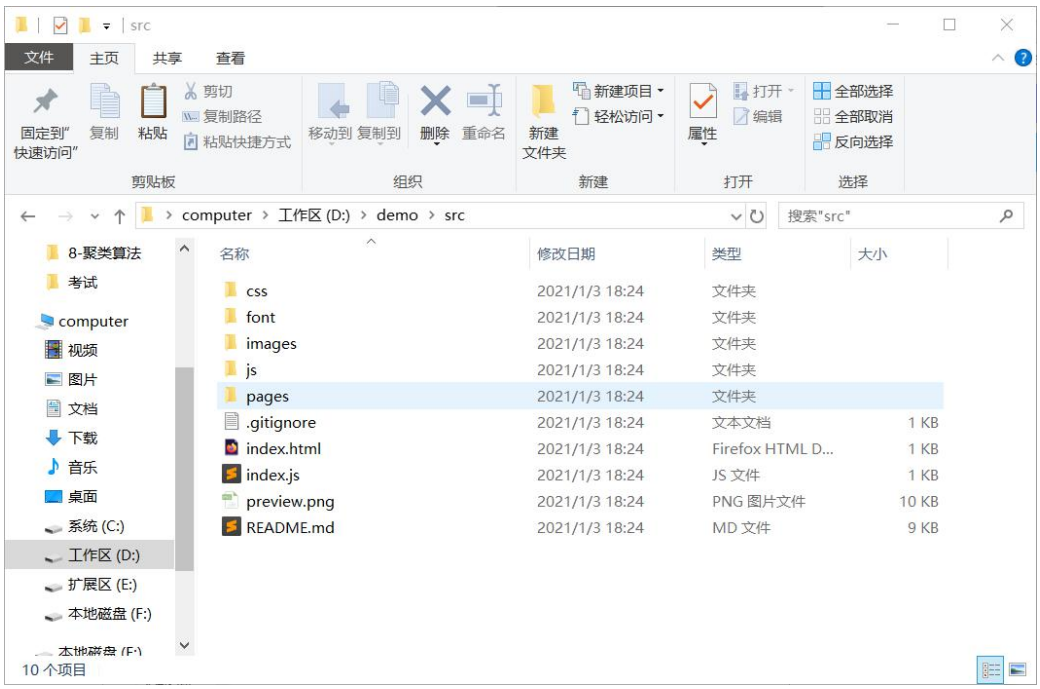
4.创建 bui 工程：安装好 buijs 框架后开始创建 bui 工程。在终端输入 buijs create demo。代表的是在当前目录下创建一个名叫 demo 的工程文件。之后接着输入 cd demo 表示进入 demo 工程文件夹。之后再输入 npm install，表示安装工程所需的依赖文件。此时 bui 工程创建完毕。

```
D:\>buijs create demo
INFO change repo gitee
INFO Creating project...
INFO Fetching release: latest...
INFO Already cached release.
INFO Copying default template file...
INFO Copy Dev done.
INFO Project created done.
```

```
D:\>cd demo

D:\demo>npm install
npm WARN deprecated gulp-util@2.2.20: gulp-util is deprecated - replace it, following the guidelines at https://
om/gulpjs/gulp-util-ca3b1f9f9ac5
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less depe
.
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regressio
sed in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/de
ues/797)
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Up
 fsevents 2.

> gifsicle@4.0.1 postinstall D:\demo\node_modules\gifsicle
> node lib/install.js

  ‼ getaddrinfo ENOENT raw.githubusercontent.com
  ‼ gifsicle pre-build test failed
  i compiling from source
```

4. **编辑 bui 工程：**此处介绍一下 bui 工程结构。进入名为 demo 的 bui 工程文件夹后，gulpfile.js 为入口文件，package.json 为 npm 依赖配置文件，app.json 为入口文件，dist/为编译打包最终要部署的目录，src/index.html 为入口文件，src/index.js 为入口的脚本，src/css/bui.css 为 BUI 库的样式文件 ，src/css/style.css 为当前应用的样式文件，src/font/为字体图标目录，src/images/为应用图片目录，src/scss/为样式源文件，样式最好放这里可以分模块管理，src/js/zepto.js 为 bui 的依赖库，src/js/plugins/fastclick.js 为移动端快速点击的插件，src/js/bui.js 位 BUI 交互控件库，src/pages/为模块目录，src/pages/main 为默认 main 模块，src/pages/main/main.html 为默认 main 模块模板，src/pages/main/main.js 为默认 main 模块定义脚本。





6. 运行 bui 工程：在 demo 的路径下的 src 内运行终端。也可以可以直接打开终端，输入 cd xxx/demo/src，xxx 表示您的 demo 工程所在路径路径。然后输入 npm run dev。之后会自动调用浏览器开始运行您的 bui 项目。

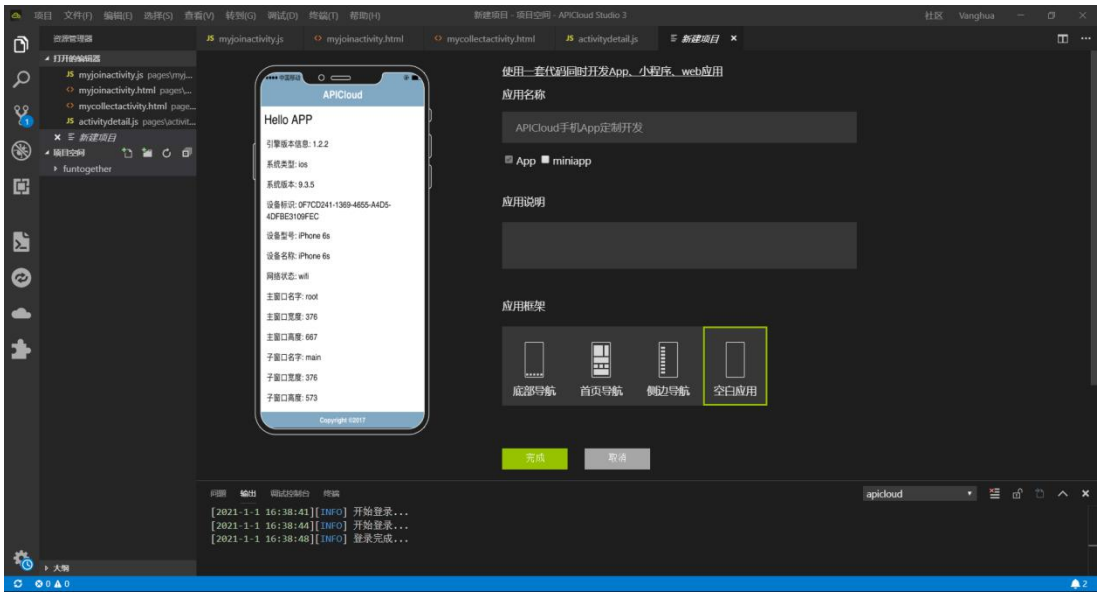
```
C:\Users\Administrator>D:

D:\>cd demo/src

D:\demo\src>npm run dev

> buiapp@1.0.0 dev D:\demo
> gulp dev

[18:29:48] Using gulpfile D:\demo\gulpfile.js
[18:29:48] Starting 'dev'...
[18:29:48] Starting 'move'...
[18:29:48] Finished 'move' after 84 ms
[18:29:48] Starting 'html'...
[18:29:48] Finished 'html' after 8.45 ms
[18:29:48] Starting 'css'...
[18:29:48] Finished 'css' after 5.82 ms
[18:29:48] Starting 'images'...
[18:29:48] Finished 'images' after 21 ms
[18:29:48] Starting 'less'...
[18:29:48] Finished 'less' after 14 ms
[18:29:48] Starting 'babel'...
[18:29:49] Finished 'babel' after 839 ms
[18:29:49] Starting 'browserify'...
[18:29:49] Finished 'browserify' after 9.13 ms
[18:29:49] Starting 'server-sync'...
```

6.1.2 如何借助第三方平台将 web 程序打包成 app（Android 和 Ios）

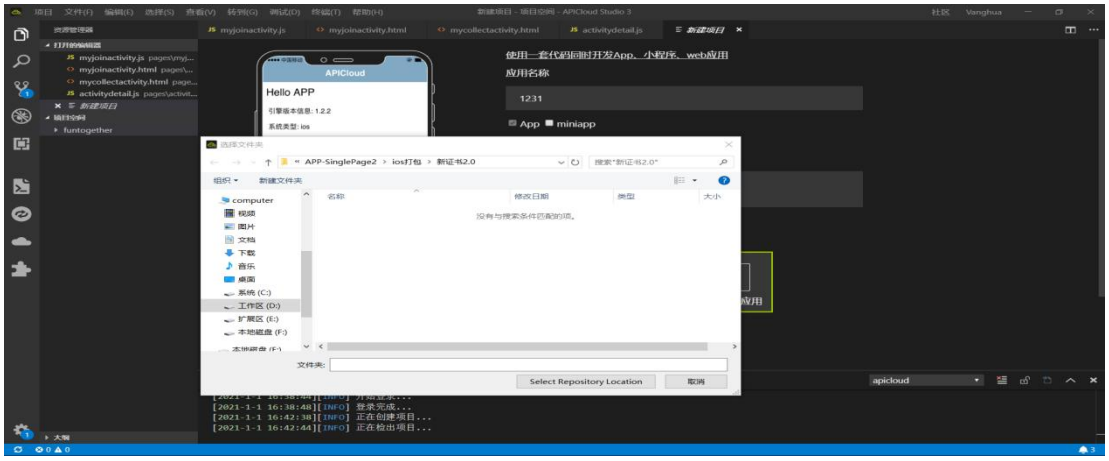
1. 准备打包平台：在 apicloud.com 上注册一个账户。然后在 apicloud.com/studio3#downloadBtn 上下载 apicloud studio，这是一个集成开发环境。

2.准备适应 apicloud 的编译好的 bui 项目：打开终端，先进入您的 bui 项目。输入 cd xxx/demo,xxx 表示您的 demo 工程所在路径路径。然后输入 buijs create -p apicloud。表示转化为适应 apicloud 的项目。终端会提示是否重写一些文件，此时直接输入 Y 表示同意即可。此时项目已经适应 apicloud 平台了，然后输入 cd src 表示进入 demo 中的 src 文件夹。然后输入 npm run build。表示编译项目。编译结束后退出终端。之后拷贝 demo 文件下的 dist 文件夹中的所有内容，这里面的内容是编译后的项目。

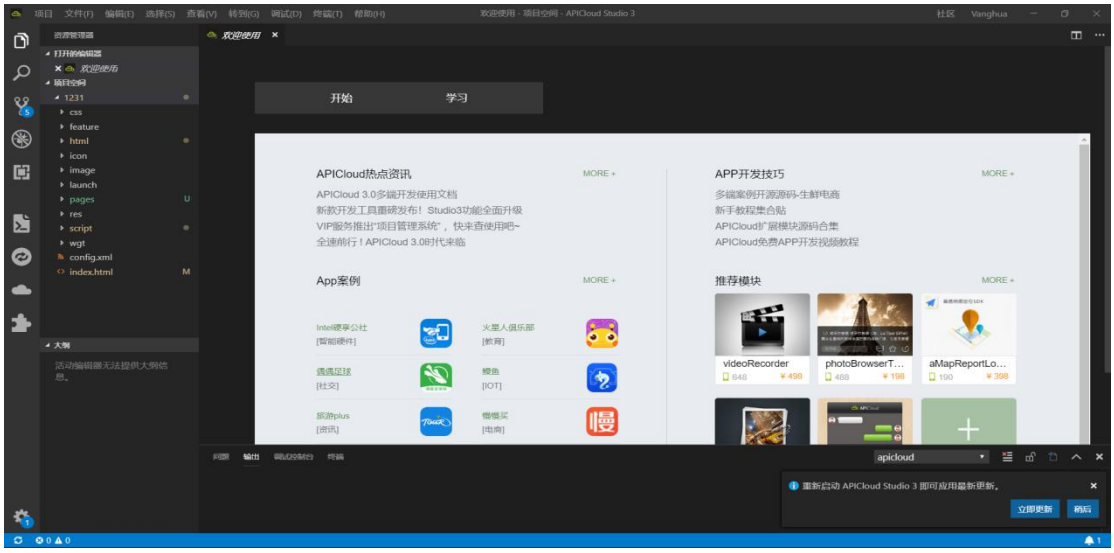
3.准备打包：打开下载好的 apicloud studio，点击左上角的新建项目(需要登录，步骤一已经注册好了账户)。然后如图操作



注意右侧选择 app 按钮，并选择空白应用，然后完善项目名称和应用说明。最后点击右侧的完成按钮。

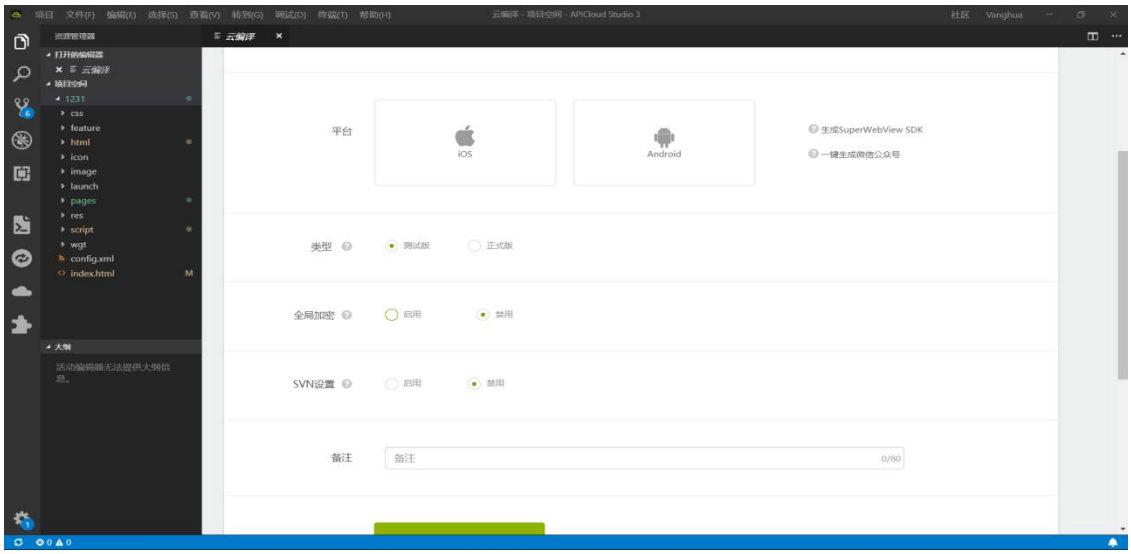


然后如图选择您想把您的待打包项目保存的路径。



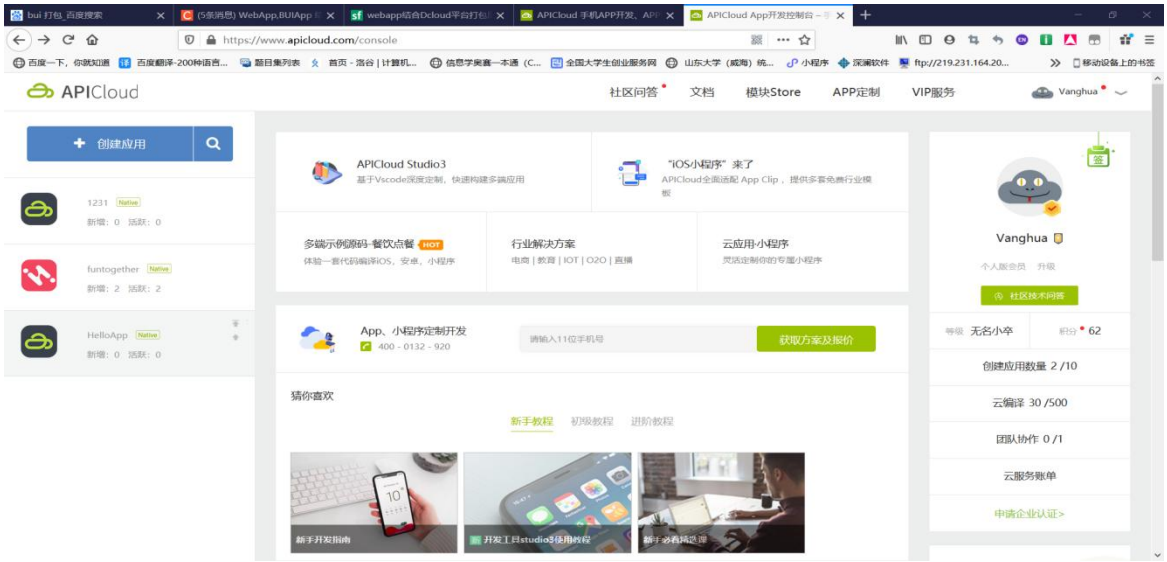
然后在左侧，右键您的项目名。如图，此时右键“1231”这个项目，然后把刚刚

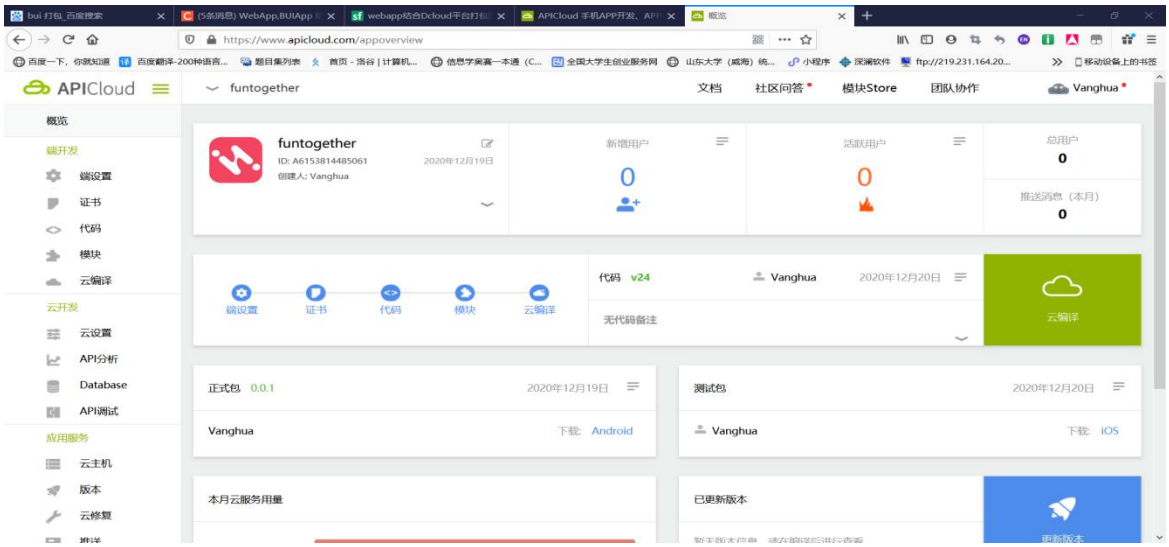
拷贝好的 dist 文件夹中的内容拷贝进来。之后再右键这个项目，点击提交云端，表示项目已经在云端更新。然后再次右键项目，点击云编译，会出现下图。



然后选择您想打包的平台，安卓还是苹果。然后选择一下此页面的其它按钮，如应用名称等等。最后点击下面的绿色按钮就开始打包了。打包结束会自动弹出 console 框，提示您安装包的下载地址。

4. 打包后的管理： 或者也可以登录 apicloud 官网，在这个位置 https://www.apicloud.com/console 编辑您的应用的启动界面等信息。具体内容如下图所示。





6.2 后端

6.2.1 如何运行本项目

在环境配置无误后（环境配置请查看 6.2.2），首先对数据库进行初始化：
首先依次执行如下三条命令：

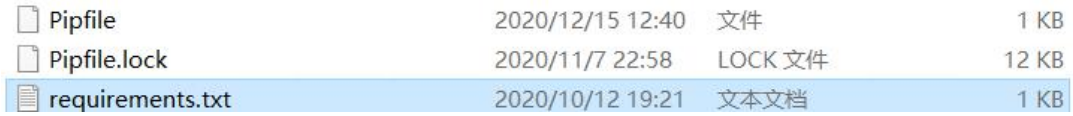
```
1.python manage.py db init
2.python manage.py db migrate
3.python manage.py db upgrade
```

随后,再执行运行命令：

```
python manage.py runserver -h 0.0.0.0
```

不报错则说明正常启动

6.2.2 如何给 flask 项目安装虚拟环境



1. 在有 requiremetns.txt 的目录下运行 `pipenv install` 命令，进行虚拟环境安装;
2. 安装结束后，运行 `pipenv shell` 进入虚拟环境;
3. 进入之后，继而运行 `python manage.py runserver -h 0.0.0.0`。