连接的IP地址为

```java
public static final String BASE_URL = "http://39.105.43.3:8080/";
```

apk下载链接

HaveFun

User相关

```java
package com.hebtu.havefun.controller;

import com.alibaba.fastjson.JSON;
import com.hebtu.havefun.entity.Messages;
import com.hebtu.havefun.service.UserService;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;
import java.util.List;

/**
 * @author PengHuAnZhi
 * @createTime 2020/11/19 11:15
 * @projectName HaveFun
 * @className UserController.java
 * @description TODO
 */
@RestController
@RequestMapping("user")
public class UserController {

    @Resource
    UserService userService;
```

判断是否注册

```java
    /**
     * @param phoneNum 电话号码
     * @return 返回"true"或者"false"
     * @description 判断是否注册
     */
    @RequestMapping("/judgeRegistered")
    public String judgeRegistered(String phoneNum) {
        if (phoneNum == null) {
            System.out.println("judgeRegistered Error");
            return "ErrorParameter";
        }
        if (userService.judgeRegistered(phoneNum)) {
            return "false";
        } else {
            return "true";
        }
    }
```

注册

```java
    /**
     * @param phoneNum 手机号码
     * @param password 密码
     * @return 返回"true"或者"false"
     * @description 注册
     */
    @RequestMapping("/register")
    public String register(String phoneNum, String password) {
        if (phoneNum == null || password == null) {
            System.out.println("register Error");
            return "ErrorParameter";
        }
        if (userService.register(phoneNum, password)) {
            return "true";
        } else {
            return "false";
        }
    }
```

登录

```java
    /**
     * @param phoneNum 手机号
     * @param password 密码
     * @return 返回的是user的对象Json串
     * @description 登录
     */
    @RequestMapping("/login")
    public String login(String phoneNum, String password) {
        if (phoneNum == null || password == null) {
            System.out.println("login Error");
            return "ErrorParameter";
        }
        String user = userService.login(phoneNum, password);
        return "".equals(user) ? "" : user;
    }
```

修改密码

```java
    /**
     * @param phoneNum 电话号码
     * @param password 密码
     * @return 返回"true"或"false"
     * @description 修改密码
     */
    @RequestMapping("/modifyPassword")
    public String modifyPassword(String phoneNum, String password) {
        if (phoneNum == null || password == null) {
            System.out.println("modifyPassword Error");
            return "ErrorParameter";
        }
        if (userService.modifyPassword(phoneNum, password)) {
            return "true";
        } else {
            return "false";
        }
    }
```

报名活动

```
    /**
     * @param activityId 活动id
     * @param id         用户id,注意不是getUserId,是getId
     * @return 返回报名成功"true",如果超出活动人数上限，返回"enough",已经报名了"exists"
     * @description 报名活动
     */
    @RequestMapping("/enrollActivity")
    public String enrollActivity(Integer activityId, Integer id) {
        if (activityId == null || id == null) {
            System.out.println("enrollActivity Error");
            return "ErrorParameter";
        }
        return userService.enrollActivity(activityId, id);
    }
```

取消报名活动

```
    /**
     * @param activityId 活动id
     * @param id         用户id,注意不是getUserId,是getId
     * @return 返回取消报名成功"true"或者失败"false"
     * @description 取消报名活动
     */
    @RequestMapping("/cancelEnrollActivity")
    public String cancelEnrollActivity(Integer activityId, Integer id) {
        if (activityId == null || id == null) {
            System.out.println("enrollActivity Error");
            return "ErrorParameter";
        }
        return "success".equals(userService.cancelEnrollActivity(activityId, id)) ? "true" : "false";
    }
```

收藏或者取消收藏

```
    /**
     * @param activityId 活动的id
     * @param id         用户id,注意不是getUserId,是getId
     * @param collect    是否注册，发送给我的是"false"或者"true"
     * @return
     * @description 收藏或者取消收藏
     */
    @RequestMapping("/changeCollectActivity")
    public String changeCollectActivity(Integer activityId, Integer id, String collect) {
        if (activityId == null || id == null || collect == null) {
            System.out.println("changeCollectActivity Error");
            return "ErrorParameter";
        }
        boolean tag = Boolean.parseBoolean(collect);
        boolean flag = userService.changeCollectActivity(activityId, id, tag);
        return flag ? "true" : "false";
    }
```

修改个性签名

```
    /**
     * @param id                用户id,注意不是getUserId,是getIds
     * @param personalSignature 新的个性签名
     * @return 返回"true"或者"false"
     * @description 修改个性签名
     */
    @RequestMapping("/modifyPersonalSignature")
    public String modifyPersonalSignature(Integer id, String personalSignature) {
        if (id == null || personalSignature == null) {
            System.out.println("modifyPersonalSignature Error");
            return "ErrorParameter";
        }
        return userService.modifyPersonalSignature(id, personalSignature) ? "true" : "false";
    }
```

获取消息列表，分页显示

```
    /**
     * @param sender   发送者id,注意不是getUserId,是getId
     * @param receiver 接收者id,注意不是getUserId,是getId
     * @param pageNum  当前页码
     * @param pageSize 页面大小
     * @return 返回List<Messages>集合Json,没有消息则为empty
     * @description 获取消息列表，分页显示
     */
    @RequestMapping("/getMsg")
    public String getMsg(Integer sender, Integer receiver, Integer pageNum, Integer pageSize) {
        if (sender == null || receiver == null || pageNum == null || pageSize == null) {
            System.out.println("getMsg Error");
            return "ErrorParameter";
        }
        List<Messages> messagesList = userService.getMsg(sender, receiver, pageNum, pageSize);
        return messagesList != null ? JSON.toJSONString(messagesList) : "empty";
    }
```

判断是否关注

```
    /**
     * @param followId   当前用户的id，即关注者id，注意不是getUserId,是getId
     * @param followedId 对方id，即被关注者id，注意不是getUserId,是getId
     * @return 返回是否关注 "true"或者"false"
     * @description 判断是否关注
     */
    @RequestMapping("/judgeFollow")
    public String judgeFollow(Integer followId, Integer followedId) {
        if (followId == null || followedId == null) {
            System.out.println("judgeFollow Error");
            return "ErrorParameter";
        }
        return userService.judgeFollow(followId, followedId) ? "true" :
"false";
    }
```

关注用户

```
    /**
     * @param followId   当前用户的id，即关注者id，注意不是getUserId,是getId
     * @param followedId 对方id，即被关注者id，注意不是getUserId,是getId
     * @return 返回关注成功还是失败true或者false
     * @description 关注用户
     */
    @RequestMapping("/followUser")
    public String followUser(Integer followId, Integer followedId) {
        if (followId == null || followedId == null) {
            System.out.println("followUser Error");
            return "ErrorParameter";
        }
        return userService.followUser(followId, followedId) ? "true" :
"false";
    }
```

取消关注用户

```
    /**
     * @param followId   当前用户的id，即关注者id，注意不是getUserId,是getId
     * @param followedId 对方id，即被关注者id，注意不是getUserId,是getId
     * @return 返回取消关注成功还是失败true或者false
     * @description 取消关注用户
     */
    @RequestMapping("/unFollowUser")
    public String unFollowUser(Integer followId, Integer followedId) {
        if (followId == null || followedId == null) {
            System.out.println("unFollowUser Error");
            return "ErrorParameter";
        }
        return userService.unFollowUser(followId, followedId) ? "true" :
"false";
```

```java
    }
```

获取粉丝数量

```java
    /**
     * @param id 当前用户的id
     * @return 返回粉丝数量
     * @description 获取用户被多少人关注，即粉丝数量
     */
    @RequestMapping("/getFollowedCount")
    public String getFollowedCount(Integer id) {
        if (id == null) {
            System.out.println("getFollowedCount Error");
            return "ErrorParameter";
        }
        Long count = userService.getFollowedCount(id);
        return count + "";
    }
```

获取关注人的数量

```java
    /**
     * @param id 当前用户的id
     * @return 返回关注人数
     * @description 获取用户关注了多少人
     */
    @RequestMapping("/getFollowCount")
    public String getFollowCount(Integer id) {
        if (id == null) {
            System.out.println("getFollowCount Error");
            return "ErrorParameter";
        }
        Long count = userService.getFollowCount(id);
        return count + "";
    }
```

获取用户信息

```java
    /**
         * @param id 用户的id
         * @return 返回user对象的JSON串,没找到对应的用户则是"false"
         * @description 根据id获取用户的信息，用于点击某个用户头像进入个人主页
         */
    @RequestMapping("/getUser")
    public String getUser(Integer id) {
        if (id == null) {
            System.out.println("getUser Error");
            return "ErrorParameter";
        }
        User user = userService.getUser(id);
        return user != null ? JSON.toJSONString(user) : "false";
    }
```

获取当前用户的关注列表

```java
    /**
     * @param id 当前用户的id
     * @return 返回一个用户列表的Json串，如果为空则返回empty
     * @description 获取当前用户的关注列表
     */
    @RequestMapping("/getFollowUsers")
    public String getFollowUsers(Integer id) {
        if (id == null) {
            System.out.println("getFollowUsers Error");
            return "ErrorParameter";
        }
        List<User> userList = userService.getFollowUsers(id);
        return userList.size() != 0 ? JSON.toJSONString(userList) : "empty";
    }
```

获取当前用户的粉丝列表

```java
    /**
     * @param id 当前用户的id
     * @return 返回一个用户列表的Json串，如果为空则返回empty
     * @description 获取当前用户的粉丝列表
     */
    @RequestMapping("/getFollowedUsers")
    public String getFollowedUsers(Integer id) {
        if (id == null) {
            System.out.println("getFollowedUsers Error");
            return "ErrorParameter";
        }
        List<User> userList = userService.getFollowedUsers(id);
        return userList.size() != 0 ? JSON.toJSONString(userList) : "empty";
    }
```

更改用户头像

```java
    /**
     * @param headPortrait 使用OkHttp3传输图片，类型为MultipartFile，名称为headPortrait
     * @param id           用户id
     * @return 返回修改成功还是失败"true"或者"false"
     * @description 更改用户头像
     */
    @RequestMapping("/modifyUserHeadPortrait")
    public String modifyUserHeadPortrait(MultipartFile headPortrait, Integer id) {
        if (headPortrait == null || id == null) {
            System.out.println("modifyUserHeadPortrait Error");
            return "ErrorParameter";
        }
        return userService.modifyUserHeadPortrait(headPortrait, id) ? "true" : "false";
    }
```

更改用户昵称

```java
    /**
     * @param userName 用户新昵称，请求参数名称为userName
     * @param id       用户id
     * @return 返回修改成功还是失败"true"或者"false"
     * @description 更改用户昵称
     */
    @RequestMapping("/modifyUserName")
    public String modifyUserName(String userName, Integer id) {
        if (userName == null || id == null) {
            System.out.println("modifyUserName Error");
            return "ErrorParameter";
        }
        return userService.modifyUserName(userName, id) ? "true" : "false";
    }
```

更改用户性别

```java
    /**
     * @param sex 用户性别字段，1或者0
     * @param id  用户id
     * @return 返回修改成功还是失败"true"或者"false"
     * @description 修改用户性别
     */
    @RequestMapping("/modifyUserSex")
    public String modifyUserSex(Integer sex, Integer id) {
        if (sex == null || id == null) {
            System.out.println("modifyUserName Error");
            return "ErrorParameter";
        }
        return userService.modifyUserSex(sex, id) ? "true" : "false";
    }
```

身份证号认证

```java
    /**
     * @param id             用户id
     * @param residentIdCard 身份证号
     * @param realName       真实姓名
     * @return 返回是否认证成功，"true"或者"false"
     * @description 根据用户id设置用户身份证号
     */
    @RequestMapping("/idCardAuthentication")
    public String idCardAuthentication(Integer id, String residentIdCard,
String realName) {
        if (id == null || residentIdCard == null || realName == null) {
            System.out.println("idCardAuthentication Error");
            return "ErrorParameter";
        }
        return "true".equals(userService.idCardAuthentication(id,
residentIdCard, realName)) ? "true" : "false";
    }
}
```

Activity相关

```java
package com.hebtu.havefun.controller;

import com.alibaba.fastjson.JSON;
import com.google.gson.Gson;
import com.hebtu.havefun.entity.activity.Activity;
import com.hebtu.havefun.entity.activity.ActivityDetail;
import com.hebtu.havefun.s  ervice.ActivityService;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;
import java.util.List;

/**
 * @author PengHuAnZhi
 * @createTime 2020/11/21 15:23
 * @projectName HaveFun
 * @className ActivityController.java
 * @description TODO
 */
@RestController
@RequestMapping("activity")
public class ActivityController {

    @Resource
    ActivityService activityService;
```

获取轮播图在服务器上的地址

```java
    /**
     * @return 返回的是一个json串，解析出来是一个list<String>,字符串分别是图片在服务器上的地址
     * @description 获取轮播图在服务器上的地址
     */
    @RequestMapping("/getRotationChartPictures")
    public String getRotationChartPictures() {
        return new Gson().toJson(activityService.getRotationChartPictures());
    }
```

获取活动列表

```java
    /**
     * @param activityKind 热门或者近期，1或者0
     * @param pageNum      页码
     * @param pageSize     页大小
     * @return List<Activity>集合的JSON串,没有活动则是"empty"
     * @description 获取活动列表
     */
    @RequestMapping("/getActivityList")
    public String getActivityList(Integer activityKind, Integer pageNum, Integer pageSize) {
        if (activityKind == null || pageNum == null || pageSize == null) {
            System.out.println("getActivityList Error");
            return "ErrorParameter";
        }
        String activities = activityService.getActivityList(activityKind, pageNum, pageSize);
        return "empty".equals(activities) ? "" : activities;
    }
```

获取活动详细信息

```java
    /**
     * @param activityId 活动的id
     * @return 返回ActivityDetail的json串,如果没有找到对应的活动详细信息则返回空串""
     * @description 获取活动详细信息
     */
    @RequestMapping("/getActivityDetail")
    public String getActivityDetail(Integer activityId) {
        if (activityId == null) {
            System.out.println("getActivityDetail Error");
            return "ErrorParameter";
        }
        String activityDetail = activityService.getActivityDetail(activityId);
        return activityDetail != null ? activityDetail : "";
    }
```

判断是否收藏

```java
    /**
     * @param id         用户id,注意不是getUserId,是getId
     * @param activityId 活动的id
     * @return 返回true或者false的字符串
     * @description 判断是否收藏
     */
    @RequestMapping("/judgeCollected")
    public String judgeCollected(Integer id, Integer activityId) {
        if (id == null || activityId == null) {
            System.out.println("judgeCollected Error");
            return "ErrorParameter";
        }
        boolean flag = activityService.judgeCollected(id, activityId);
        return flag ? "true" : "false";
    }
```

获取收藏的活动列表

```java
    /**
     * @param id       用户id,注意不是getUserId,是getId
     * @param pageNum  页码
     * @param pageSize 页大小
     * @return 返回List<UserCollectActivity>集合,如果没有就返回"empty"
     * @description 获取收藏的活动列表
     */
    @RequestMapping("/getCollectedActivities")
    public String getCollectedActivities(Integer id, Integer pageNum, Integer pageSize) {
        if (id == null || pageNum == null || pageSize == null) {
            System.out.println("getCollectedActivities Error");
            return "ErrorParameter";
        }
        return activityService.getCollectedActivities(id, pageNum, pageSize);
    }
```

添加活动

```
    /**
     * @param files              客户端传递若干MultipartFile文件，要求请求的参数名字都
     *                           是file（就是要重复），这边接收就是自动加入List<MultipartFile>中
     * @param activityDetailJson 客户端将封装好的ActivityDetail类转换为Json串发送过来，
     *                           注意ActivityDetail里面的Activity类也得把数据都封装好，
     *                           添加操作，不需要设置id值，因为数据库id是自增的
     * @return 返回一个添加成功或者失败"true"或者"false"
     * @description 添加活动，接受一个ActivityDetail类的Json串数据，且参数名称为activityJson
     */
    @RequestMapping("/addActivity")
    public String addActivity(@RequestParam("file") List<MultipartFile> files, String activityDetailJson) {
        if (activityDetailJson == null) {
            System.out.println("addActivity Error");
            return "ErrorParameter";
        }
        return activityService.addActivity(files, activityDetailJson) ? "true" : "false";
    }
```

获取报名的活动列表

```
    /**
     * @param id       用户id,注意不是getUserId,是getId
     * @param pageNum  页码
     * @param pageSize 页大小
     * @return 返回List<UserCollectActivity>集合,如果没有就返回"empty"
     * @description 获取报名的活动列表
     */
    @RequestMapping("/getEnterActivities")
    public String getEnterActivities(Integer id, Integer pageNum, Integer pageSize) {
        if (id == null || pageNum == null || pageSize == null) {
            System.out.println("getEnterActivities Error");
            return "ErrorParameter";
        }
        return activityService.getEnterActivities(id, pageNum, pageSize);
    }
```

获取发布的活动列表

```java
    /**
     * @param id       用户id,注意不是getUserId,是getId
     * @param pageNum  页码
     * @param pageSize 页大小
     * @return 返回List<UserCollectActivity>集合,如果没有就返回"empty"
     * @description 获取发布的活动列表
     */
    @RequestMapping("/getPublishActivities")
    public String getPublishActivities(Integer id, Integer pageNum, Integer pageSize) {
        if (id == null || pageNum == null || pageSize == null) {
            System.out.println("getPublishActivities Error");
            return "ErrorParameter";
        }
        return activityService.getPublishActivities(id, pageNum, pageSize);
    }
```

根据若干条件筛选活动

```java
   /**
     * @description 根据若干条件筛选活动,每个参数都给我传过来，如果用户没有做出对应的选择，String类型传empty，Integer类型传-1
     * @param typeName 小类活动的名称
     * @param lowCost 价格区间的低区间
     * @param highCost 价格区间的高区间
     * @param howManyDays 近多少天
     * @param city 市
     * @param county 区
     * @param pageNum 页码
     * @param pageSize 页大小
     * @return 返回一个List<Activity>集合的JSON串，如果集合为空，返回字符串"empty"
     */
    @RequestMapping("/screenActivities")
    public String screenActivities(String typeName, Integer lowCost, Integer highCost, Integer howManyDays, String city, String county, Integer pageNum, Integer pageSize) {
        if (howManyDays == null || pageNum == null || pageSize == null || typeName == null || lowCost == null || highCost == null || city == null || county == null) {
            System.out.println("screenActivities Error");
            return "ErrorParameter";
        }
        String activityList = activityService.screenActivities(howManyDays, typeName, lowCost, highCost, city, county, pageNum, pageSize);
        return !"empty".equals(activityList) ? JSON.toJSONString(activityList) : "empty";
    }
```

判断用户是否报名活动

```java
    /**
     * @param id         用户id
     * @param activityId 活动id
     * @return 返回已报名"true"，未报名"false"
     * @description 根据用户id和活动id查询用户是否报名这个活动
     */
    @RequestMapping("/judgeEnterActivity")
    public String judgeEnterActivity(Integer id, Integer activityId) {
        if (id == null || activityId == null) {
            System.out.println("judgeEnterActivity Error");
            return "ErrorParameter";
        }
        return activityService.judgeEnterActivity(id, activityId);
    }
```

修改活动信息

```java
    /**
     * @param activityDetailJson 修改后的ActivityDetail对象的JSON串
     * @return 返回修改成功"true"
     * @description 修改活动信息
     */
    @RequestMapping("/modifyActivity")
    public String modifyActivity(String activityDetailJson) {
        if (activityDetailJson == null) {
            System.out.println("modifyActivity Error");
            return "ErrorParameter";
        }
        return activityService.modifyActivity(activityDetailJson);
    }
```

删除活动

```java
    /**
     * @param activityId 活动id
     * @return 返回删除成功"true"或者"false"
     * @description 删除活动，数据库状态字段置0
     */
    @RequestMapping("/deleteActivity")
    public String deleteActivity(Integer activityId) {
        if (activityId == null) {
            System.out.println("deleteActivity Error");
            return "ErrorParameter";
        }
        return "true".equals(activityService.deleteActivity(activityId)) ?
"true" : "false";
    }
}
```