

# 课程介绍 《探花交友》

---

- 项目介绍
- 项目的功能介绍
- 工程搭建
- 短信验证码
- 阿里云OSS服务应用
- 人脸识别
- 完善个人信息

## 1、项目介绍

---

### 1.1、项目背景

在线社交是互联网时代的产物，已成为互联网用户的基础需求之一。移动互联网自2003年起快速发展，促使在线社交逐渐从PC端转移至移动端。移动社交最初以熟人社交为主，以维系熟人关系、共享资源信息的形式存在。随着人们交友需求的延伸，移动社交开始向陌生人社交、兴趣社交等垂直方向发展，形式丰富多样。

### 1.2、市场分析

探花交友项目定位于 **陌生人交友市场**。

- 根据《2018社交领域投融资报告》中指出：虽然相比2017年，投融资事件减少29.5%，但是融资的总额却大幅增长，达到68%。
- 这些迹象说明：社交领域的发展规模正在扩大，而很多没有特色的产品也会被淘汰。而随着那些尾部产品的倒下，对我们来说就是机会，及时抓住不同社交需求的机会。以社交为核心向不同的细分领域衍生正在逐渐走向成熟化。
- 而我们按照娱乐形式和内容为主两个维度，将社交行业公司分类为：即时通信、内容社群、陌生人社交、泛娱乐社交以及兴趣社交几个领域。
- 而在2018年社交的各个细分领域下，均有备受资本所关注的项目，根据烯牛数据2018年的报告中，也同样指出：内容社交及陌生人社交为资本重要关注领域，合计融资占比达73%。

## 2018年国内社交领域融资事件数量分布

犀牛数据  
RHINO DATA



数据来源：犀牛数据

根据市场现状以及融资事件来看：陌生人社交、内容社群、兴趣社交在2019年仍然保持强劲的动力，占到近70%的比例，它们仍然是资本市场主要关注领域。从增长率来看陌生人社交的增长速度远远大于其他几类，因此我们要从这个方向入手。

### 1.3、目标用户群体

从整体年龄段来看：目前目标用户群体主要以30岁以下为主，其中以18-25岁年龄群体为主要受众人群。

- **上班群体**：热衷于通过分享内容或表达“个人情绪”在陌生人面前建立特殊的人设，并借此提升自我价值扩大自己的交际圈；
- **学生群体**：追求个性选择，更倾向找到有共同话题的陌生人对象并建立长期的关系，乐于展现自我；
- **文艺群体**：拥有自己独特的爱好且拥有特别的个人追求，追求文艺圈子内的交流，希望通过分享结交更多好友；
- **沟通弱势群体**：对现有长期保持线上对社交模式表现无力且无效，渴望有更加有效且安全的社交方式出现，解决目前单调乏味的沟通方式；

### 1.4、使用场景

#### 用户场景一：

关键词：内向、社交障碍、不主动

大学二年级的陈烨是一位品学兼优且容貌昳丽的小女生，但从小到大的朋友特别少。在聚会时大家都觉得她很高冷，但是陈烨只是不会找时机插不上话，偶尔说上几句也是如细雨飘过。在各类群体社交场合也难以融入人群。

后来，看到室友小白在玩一款陌生人社交软件并引起了她的兴趣，她可以在软件中建立一个内向真实的自己，尝试学会更主动更热情地去了解他人。

但是，玩了一段时间后发现很多陌生人都不愿意与她长聊，或者说聊久了无话可说缺乏话题逐渐变成了好友列表里的一个摆设。

在某乎的某个回答中她看到探花交友App，抱着试一试的心态也尝试着体验了一番，从一开始的每天匹配随心聊天到后来认识到几个有共同爱好的朋友。这同时也让她在社交中慢慢提升自己变得更好。

## **用户场景二：**

关键词：分享、互动、娱乐

陈莹是一位初入职场的新人，喜欢看书、听音乐、创作、拍照....几乎对什么都感兴趣，在毕业后她发现认识新朋友，和新朋友一起出去玩的机会越来越少了。朋友圈里的大家都是二点一线的生活，陈莹喜欢晒生活，说趣闻，发心情。但是，对于朋友圈这个“大杂烩”来说，她不想暴露太多的自我。

在一个偶然的机会，她看到微信公众号有一篇关于社交产品的推文，一向对此嗤之以鼻的她突然来了点兴趣。在用了一段时间后，她发现：她每天可以将自己不愿意分享到朋友圈里的内容，分享到社交产品里。而且发几条，发的内容是什么，她也可以经常将自己所想，所写，所拍都上传到“圈子”里。

对于懂这些东西的人，他们会主动的聚集过来讨论。因此，她也加入到某个兴趣小组，时不时与他们在线上探讨一些问题。陈莹不但找到了属于她自己的社交圈子，同时也找到一个可以随时随地分享点滴的平台。

## **用户场景三：**

关键词：脱单、脱单、脱单

作为一个直男，宋沌堪称直男教学书一般的案例，他的行为类似下图：

曾谈过几次恋爱，都以迅速失败告终。作为一个长相一般，身家一般，谈吐一般的综合表现男来说，他基本把自己定义成街上一抓一大把的类型。但是，作为一个直男的他也是有个异性梦，每天都梦想着有一个女友，所以他也不断在尝试。

他几乎下载了市面上所有的社交产品，摆上了经过“特殊处理”的自拍照，时不时更新自己的动态。但即便如此，宋沌依然没有几个异性聊友，宋沌也反省过自己，主要是自己每次图一时新鲜，聊一段时间就不感兴趣了，而且由于自己比较害羞所以聊天也容易尬聊。

在朋友的介绍下，他下载了探花APP，由于属于陌生人社交，宋沌可以不用有太多的思想压力，经过几天的好友配对，找到了合适的朋友，每天发一些日常生活的消息，也能获得更多的关注，自信心逐渐增长，聊天技巧也有所提升。

## **1.5、竞争对手分析**

### **1.5.1、竞品选择**

根据我们的市场调研以及分析：从产品细分领域以及对应的产品定位来选择，我们选择了社交范围内的兴趣社交App作为竞品分析的案例。

其中，我们发现：市面上的兴趣社交产品还是较多的，例如花田、soul、探探、陌陌等等，最终我们选择了**花田、SOUL和陌陌**。

竞品分析			
竞品名称	slogan	产品定位	产品特色
花田	你的爱情在花田	花田交友是国内最高效的恋爱交友社区	智能匹配
			缘分电话亭
			恋爱研究院
SOUL	跟随灵魂找到你	匹配追求灵魂交流的弱社交，寻找属于自己的 soul mate	灵魂自测游戏
			3D 星球邂逅
			匿名的私密 BLOG
陌陌	很高兴认识你	陌陌是一款基于地理位置的移动社交工具	多人交友、快聊
			直播短视频
			视频交友

### 1.5.2、竞品分析

- **花田**：更偏向打造兴趣匹配，并配合线下活动两者结合提升产品服务。給每一个热爱青年文化的用户营造出归属感，并促使用户自主的生产内容，形成一个良性的娱乐社交平台。
- **SOUL**：更注重用户灵魂（内涵）的产品，一定程度上，SOUL摒弃了传统社交的以颜值优先，内容其次的特点。将自身的个性以及特点先展现出去，然后再以内部算法为匹配手段，通过图文内容进行用户交流。
- **陌陌**：陌陌是一款基于地理位置的移动社交工具。使用者可以通过陌陌认识附近的人，免费发送文字消息、语音、照片以及精准的地理位置和身边的人更好的交流；可以使用陌陌创建和加入附近的兴趣小组、留言及附近活动和陌陌吧。

三款产品各具风格，各有特点，但有一点是三款产品都有一个核心观点，就是：弱化肤浅的目的，利用人类自带的自我认识的本能来结识陌生人。总结而言，就是：希望满足用户『探索自我』的娱乐性。

## 1.6、项目简介

探花交友是一个陌生人的在线交友平台，在该平台中可以搜索附近的人，查看好友动态，平台还会通过大数据计算进行智能推荐，通过智能推荐可以找到更加匹配的好友，这样才能增进用户对产品的喜爱度。探花平台还提供了在线即时通讯功能，可以实时的与好友进行沟通，让沟通随时随地的进行。

••••• Sketch ⚡

9:41 AM

100% 🔋

探花



搜附近



桃花传音



测灵



今日佳人



黑马小妹 ♂ 26

单身 | 本科 | 年龄相仿



缘分值

推荐



89

米朵妹妹

♂ 25

单身 本科 年龄相仿



89

致远哥哥

♂ 25

单身 本科 年龄相仿



交友



圈子



消息



视频



我的

## 推荐 好友



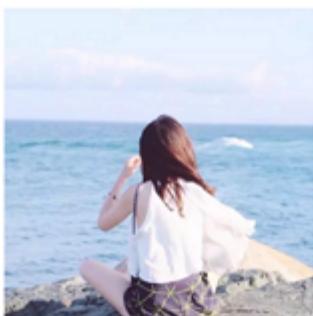
米朵妹妹

25

单身 | 本科 | 年龄相仿

:::

下属去看脱发……请病假合适吗……



距离1.2km 10分钟前

18

30

45



夏了的夏天

25

单身 | 本科 | 年龄相仿

:::

有家的地方，没有工作，有工作的地方没有家，他乡容不下的灵魂，故乡安置不了的肉身，谁能留下我这尊大可爱~



发布



交友



圈子



消息



视频



我的

## 1.7、技术方案

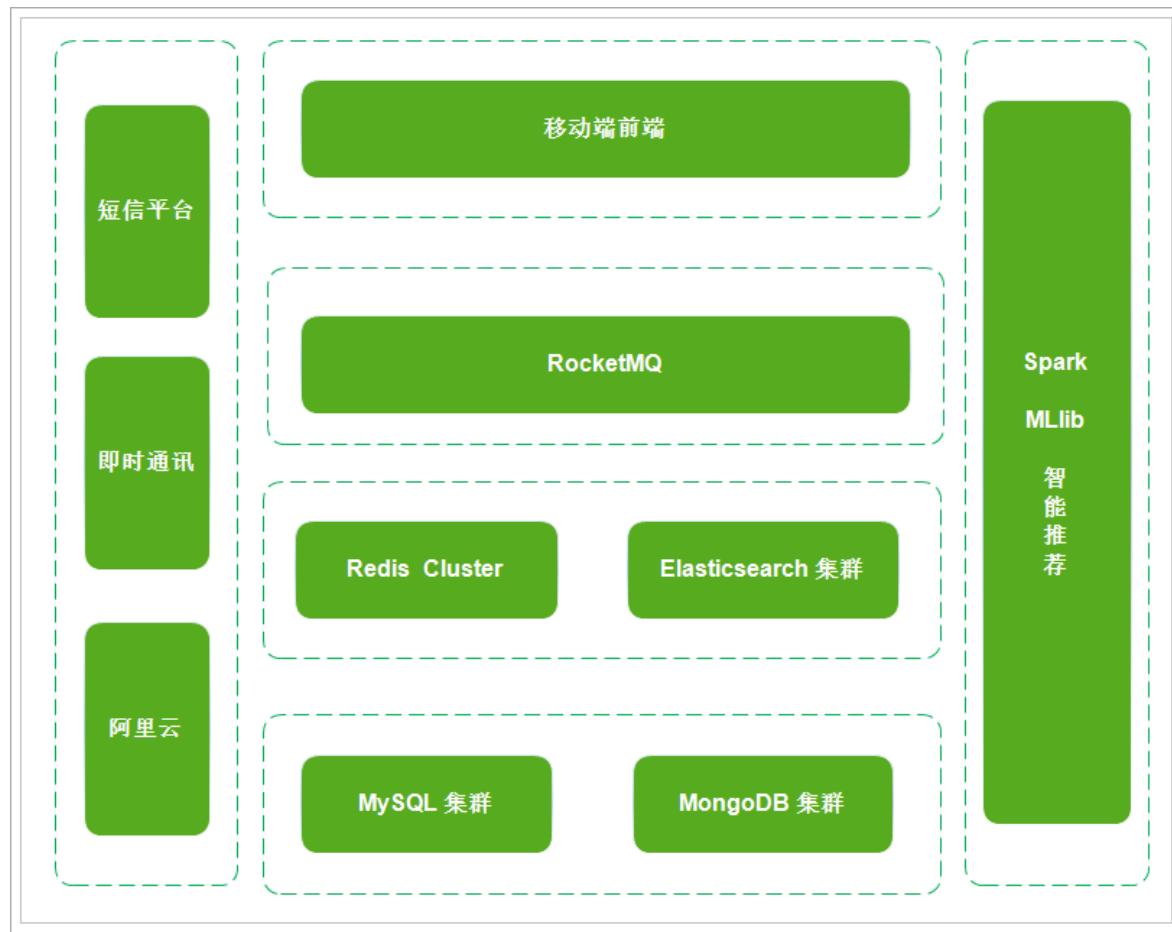
前端：

- flutter + android + 环信SDK + redux + shared\_preferences + connectivity + iconfont + webview + sqflite

后端：

- Spring Boot + SpringMVC + Mybatis + MybatisPlus + Dubbo
- Elasticsearch geo 实现地理位置查询
- MongoDB 实现海量数据的存储
- Redis 数据的缓存
- cdn 加速静态文件的加载
- Spark + MLlib 实现智能推荐
- 第三方服务 环信即时通讯
- 第三方服务 阿里云 OSS

## 1.8、技术架构



## 1.9、技术解决方案

- 使用Elasticsearch geo实现附近的人的解决方案
- 使用Spark + Mllib实现智能推荐的解决方案
- 使用MongoDB进行海量数据的存储的解决方案
- 使用采用分布式文件系统存储小视频数据的解决方案
- 使用虹软开放平台进行人脸识别的解决方案

## 1.10、技术亮点

- 采用Elasticsearch geo实现地理位置查询
- 采用RocketMQ作为消息服务中间件
- 采用MongoDB进行海量数据的存储
- 采用Spark + Mllib实现智能推荐

- 采用环信服务实现即时通讯
- 采用分布式文件系统存储小视频数据
- 采用CDN技术加速静态资源以及小视频的加载
- 采用Apache Dobbo作为微服务架构技术
- 采用SpringBoot + Mybatis实现系统主架构
- 采用Redis集群实现缓存的高可用

## 2、功能介绍

探花交友平台，涵盖了主流常用的一些功能，如：交友、聊天、动态等。

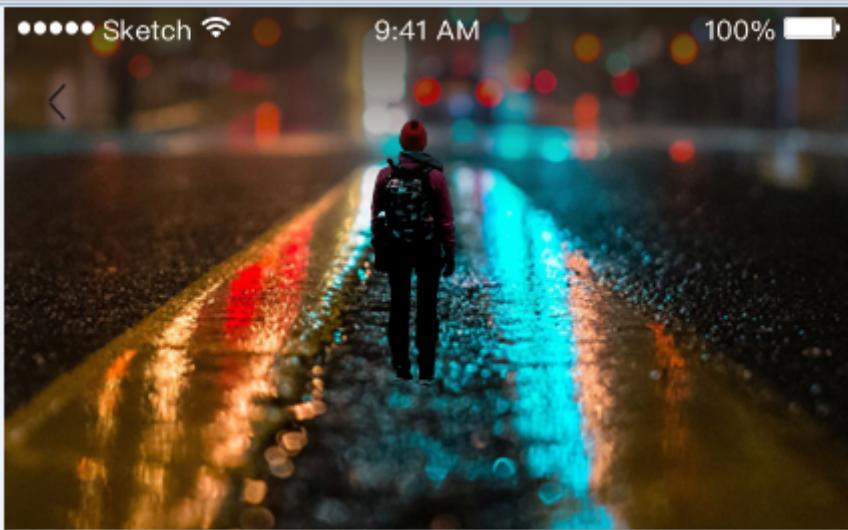
### 2.1、功能列表

功能	说明	备注
注册、登录	用户无需单独注册，直接通过手机号登录即可	首次登录成功后需要完善个人信息
交友	主要功能有：测灵魂、桃花传音、搜附近、探花等	
圈子	类似微信朋友圈，用户可以发动态、查看好友动态等	
消息	通知类消息 + 即时通讯消息	
小视频	类似抖音，用户可以发小视频，评论等	显示小视频列表需要进行推荐算法计算后进行展现。
我的	我的动态、关注数、粉丝数、通用设置等	

### 2.2、注册登录

业务说明：

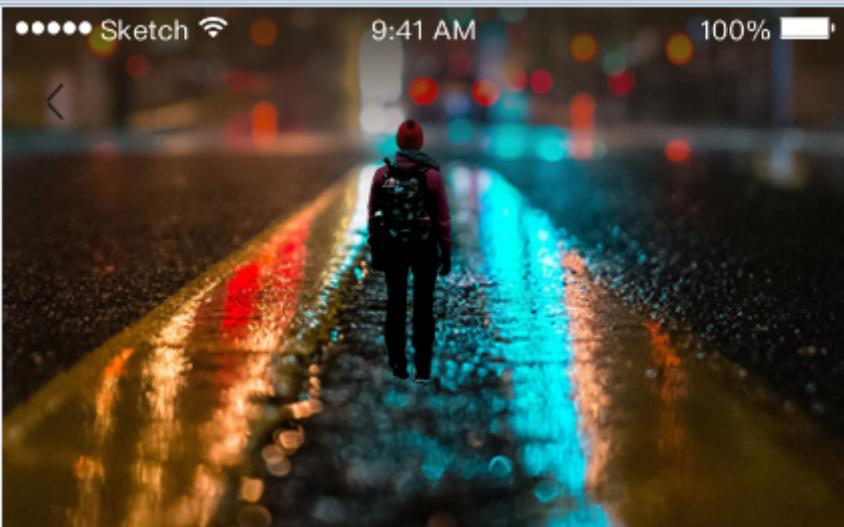
用户通过手机验证码进行登录，如果是第一次登录则需要完善个人信息，在上传图片时，需要对上传的图片做人像的校验，防止用户上传非人像的图片作为头像。流程完成后，则登录成功。



## 手机号登录注册

+86  输入手机号码

获取验证码



输入6位验证码

已发到: +86 16678231276

5

3

6

1

4

7

短信验证码错误

重新获取 (50s)



## 填写资料 提升我的魅力



昵称设置

你的生日

992cf0

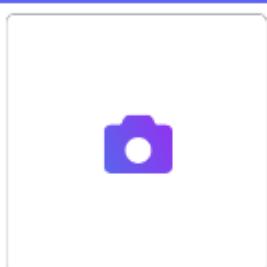


现居城市

设置头像

&lt; 返回

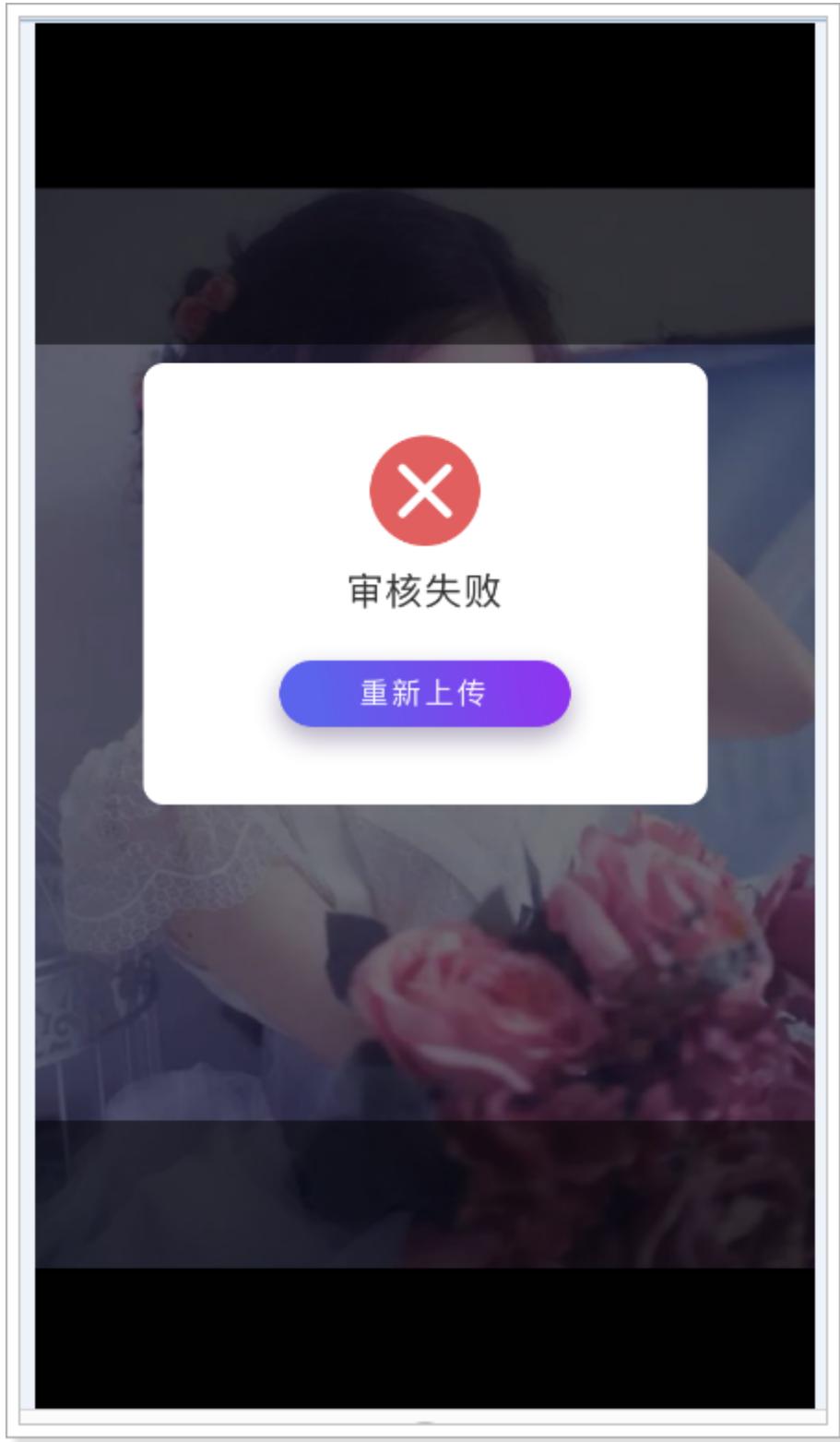
## 设置头像





审核成功

开始交友



## 2.3、交友

交友是探花项目的核心功能之一，用户可以查看好友，添加好友，搜索好友等操作。

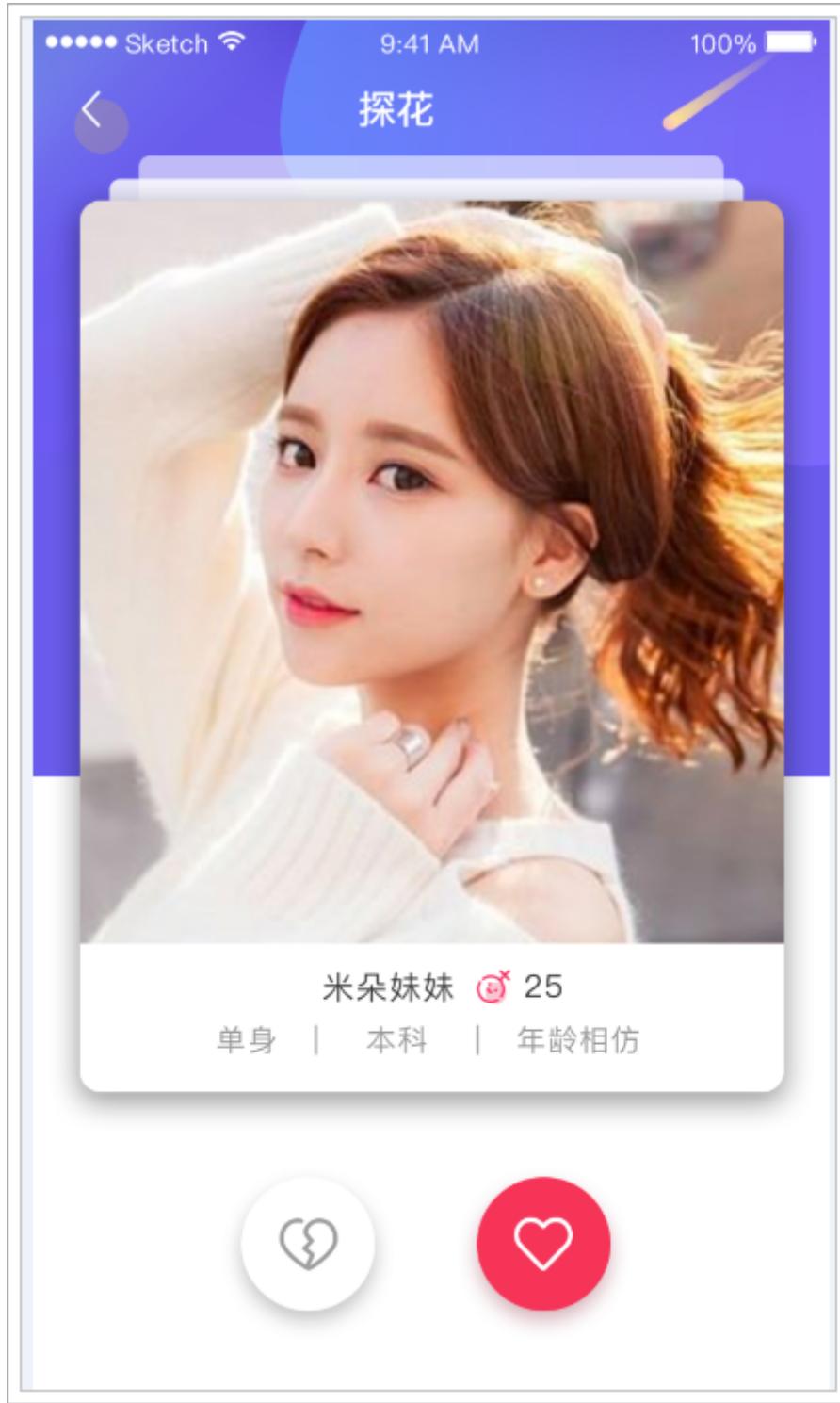


### 2.3.1、首页

在首页中，主要功能有“今日佳人”、“推荐”、“最近访客”等

- 今日佳人
  - 按照“缘分值”进行匹配，将“缘分值”最高的用户展现出来
- 推荐
  - 按照“缘分值”进行推荐，由后台的推荐系统计算得出，展现出来
- 最近访客
  - 显示最近来看“我”的用户

### 2.3.2、探花



说明：左划喜欢，右划不喜欢，每天限量不超过100个，开通会员可增加限额。双方互相喜欢则配对成功。

实现：数据来源推荐系统计算后的结果。

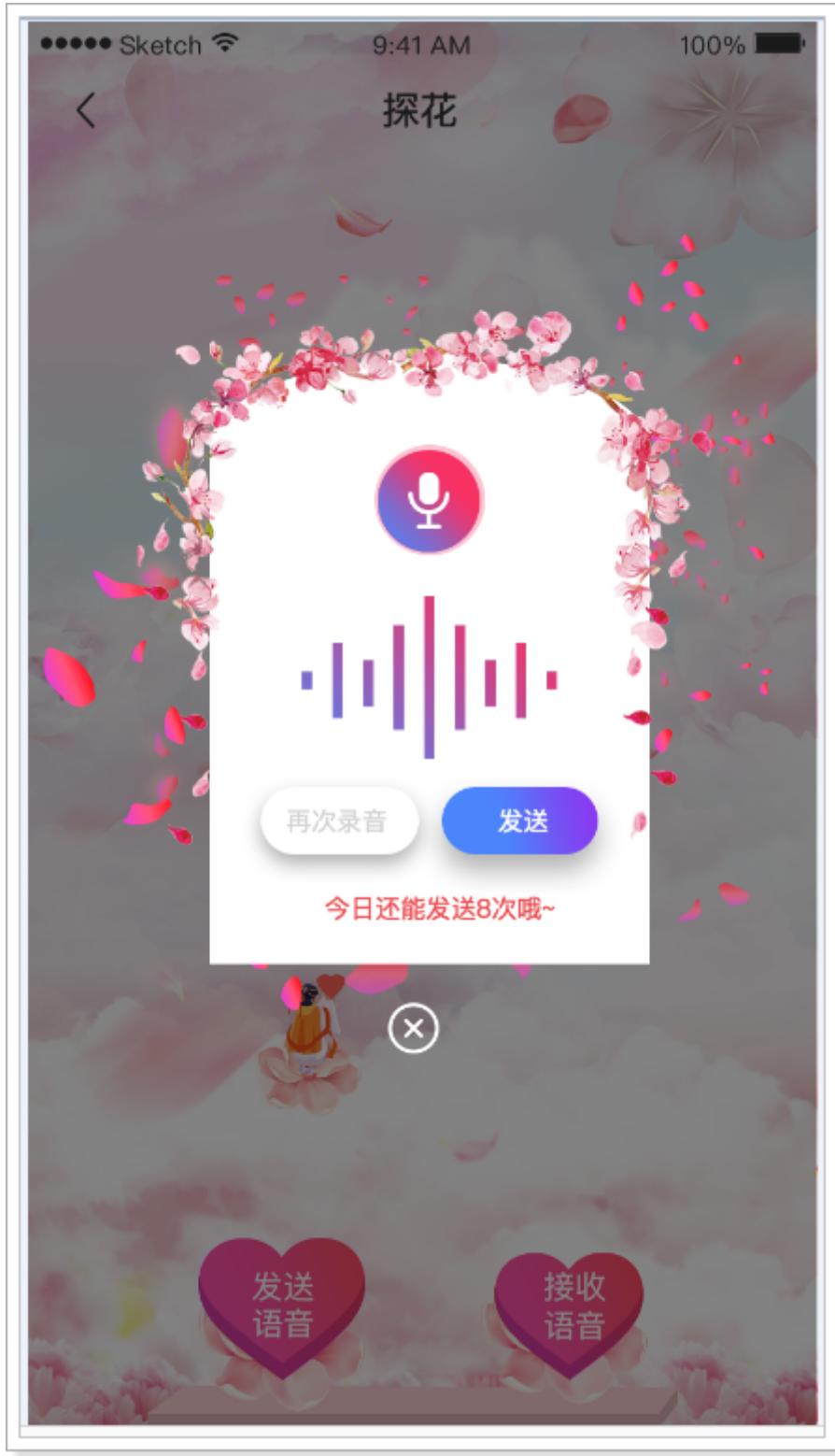
### 2.3.3、搜附近



根据用户当前所在的位置进行查询，并且在10km的范围内进行查询，可以通过筛选按钮进行条件筛选。

#### 2.3.4、桃花传音

功能类似QQ中的漂流瓶，用户可以发送和接收语音消息，陌生人就会接收到消息。



### 2.3.5、测灵魂

1. 测试题用于对用户进行分类，每次提交答案后更新用户属性
2. 测试题在后台进行维护
3. 测试题测试完后产生结果页可以进行分享
4. 测试题为顺序回答，回答完初级题解锁下一级问题
5. 点击锁定问题 显示提示 请先回答上一级问题



## 灵魂测试



初级灵魂题



开始测试



## 灵魂测试



## 第一题

(1/12)

初级

假如你在你房间墙壁上发现一个了小孔，  
你希望从这个小孔中看如下到什么样的  
场景？

正在拥抱的一对恋人

一家人正其乐融融地吃晚餐



## 2.4、圈子

- 1、推荐频道为根据问卷及喜好推荐相似用户动态
- 2、显示内容为用户头像、用户昵称、用户性别、用户年龄、用户标签和用户发布动态
- 3、图片最多不超过6张或发布一个小视频
- 4、动态下方显示发布时间距离当时时间，例如10分钟前、3小时前、2天前，显示时间进行取整
- 5、动态下方显示距离为发布动态地与本地距离
- 6、显示用户浏览量
- 7、显示点赞数、评论数 转发数

## 推荐 好友

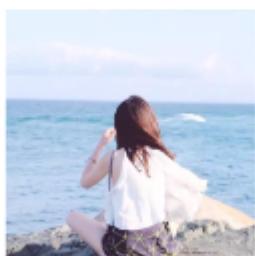


米朵妹妹 25

单身 | 本科 | 年龄相仿

:::

下属去看脱发.....请病假合适吗.....



距离1.2km 10分钟前

18

30

45



夏了的夏天 25

单身 | 本科 | 年龄相仿

:::

有家的地方，没有工作，有工作的地方没有家，他  
乡容不下的灵魂，故乡安置不了的肉身，谁能留下  
我这尊大可爱~



发布



交友



圈子



消息



视频



我的



米朵妹妹 25  
单身 | 本科 | 年龄相仿

下属去看脱发.....请病假合适吗.....



最新评论 99+



春花多时有  
最美的风景配最美的人~

11

08:27



清水河上住  
灯泡

11

08:27



清水河上住  
灯泡

11

08:27



清水河上住  
灯泡

11

08:27



清水河上住  
灯泡

11

08:27

清水河上住

发布评论



发送

## 2.5、消息

消息包含通知类的消息和好友消息。



## 2.6、小视频

用户可以上传小视频，也可以查看小视频列表，并且可以进行点赞操作。



## 2.7、我的

显示关注数、喜欢数、粉丝数、我的动态等信息。

Sketch

9:41 AM

100%



谁动了我的冬天

26

北京

113

345

113

互相关注

喜欢

粉丝



我的动态



谁看过我



钱包



通用设置



客服在线





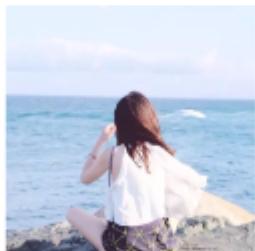
## 我的动态



米朵妹妹 25

单身 | 本科 | 年龄相仿

下属去看脱发……请病假合适吗……



距离1.2km 10分钟前

18

30



夏了的夏天 25

单身 | 本科 | 年龄相仿

有家的地方，没有工作，有工作的地方没有家，他乡容不下的灵魂，故乡安置不了的肉身，谁能留下我这尊大可爱~



### 3、注册登录

业务说明：

用户通过手机验证码进行登录，如果是第一次登录则需要完善个人信息，在上传图片时，需要对上传的图片做人像的校验，防止用户上传非人像的图片作为头像。流程完成后，则登录成功。

#### 3.1、搭建工程

##### 3.1.1、itcast-tanhua

itcast-tanhua是父工程，集中定义了依赖的版本以及所需要的依赖信息。

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5             http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7     <packaging>pom</packaging>
8
9     <parent>
10        <groupId>org.springframework.boot</groupId>
11        <artifactId>spring-boot-starter-parent</artifactId>
12        <version>2.1.0.RELEASE</version>
13    </parent>
14
15    <groupId>cn.itcast.tanhua</groupId>
16    <artifactId>itcast-tanhua</artifactId>
17    <version>1.0-SNAPSHOT</version>
18
19    <!-- 集中定义依赖版本号 -->
20    <properties>
21        <mysql.version>5.1.47</mysql.version>
22        <jackson.version>2.9.9</jackson.version>
23        <druid.version>1.0.9</druid.version>
24        <servlet-api.version>2.5</servlet-api.version>
25        <jsp-api.version>2.0</jsp-api.version>
26        <joda-time.version>2.5</joda-time.version>
27        <commons-lang3.version>3.3.2</commons-lang3.version>
28        <commons-io.version>1.3.2</commons-io.version>
29        <mybatis.version>3.2.8</mybatis.version>
30        <mybatis.mybatis-plus>3.1.1</mybatis.mybatis-plus>
31        <tombok.version>1.18.4</tombok.version>
32    </properties>
33
34    <!--通用依赖-->
35    <dependencies>
36        <dependency>
37            <groupId>org.springframework.boot</groupId>
38            <artifactId>spring-boot-starter-test</artifactId>
39            <scope>test</scope>
40        </dependency>
41        <dependency>
42            <groupId>junit</groupId>
43            <artifactId>junit</artifactId>
44            <version>4.12</version>
45            <scope>test</scope>
46        </dependency>
47    </dependencies>
48    <dependencyManagement>
49        <dependencies>
50            <!-- mybatis-plus插件依赖 -->
51            <dependency>
52                <groupId>com.baomidou</groupId>
53                <artifactId>mybatis-plus</artifactId>
54                <version>${mybatis.mybatis-plus}</version>
55            </dependency>
56            <!-- MySql -->
57            <dependency>
58                <groupId>mysql</groupId>
```

```
58         <artifactId>mysql-connector-java</artifactId>
59         <version>${mysql.version}</version>
60     </dependency>
61
62     <dependency>
63         <groupId>org.mongodb</groupId>
64         <artifactId>mongodb-driver-sync</artifactId>
65         <version>3.9.1</version>
66     </dependency>
67     <dependency>
68         <groupId>org.projectlombok</groupId>
69         <artifactId>lombok</artifactId>
70         <optional>true</optional>
71         <version>${lombok.version}</version>
72     </dependency>
73     <dependency>
74         <groupId>org.apache.commons</groupId>
75         <artifactId>commons-lang3</artifactId>
76         <version>${commons-lang3.version}</version>
77     </dependency>
78     <!--RocketMQ相关依赖-->
79     <dependency>
80         <groupId>org.apache.rocketmq</groupId>
81         <artifactId>rocketmq-spring-boot-starter</artifactId>
82         <version>2.0.0</version>
83     </dependency>
84     <dependency>
85         <groupId>org.apache.rocketmq</groupId>
86         <artifactId>rocketmq-client</artifactId>
87         <version>4.3.2</version>
88     </dependency>
89     <dependency>
90         <groupId>org.apache.commons</groupId>
91         <artifactId>commons-lang3</artifactId>
92         <version>3.7</version>
93     </dependency>
94     <!-- Jackson Json处理工具包 -->
95     <dependency>
96         <groupId>com.fasterxml.jackson.core</groupId>
97         <artifactId>jackson-databind</artifactId>
98         <version>${jackson.version}</version>
99     </dependency>
100    <dependency>
101        <groupId>com.alibaba</groupId>
102        <artifactId>druid</artifactId>
103        <version>${druid.version}</version>
104    </dependency>
105    <dependency>
106        <groupId>commons-codec</groupId>
107        <artifactId>commons-codec</artifactId>
108        <version>1.11</version>
109    </dependency>
110  </dependencies>
111 </dependencyManagement>
112
113 <build>
114   <plugins>
115     <!-- java编译插件 -->
```

```

116     <plugin>
117         <groupId>org.apache.maven.plugins</groupId>
118         <artifactId>maven-compiler-plugin</artifactId>
119         <version>3.2</version>
120         <configuration>
121             <source>1.8</source>
122             <target>1.8</target>
123             <encoding>UTF-8</encoding>
124         </configuration>
125     </plugin>
126 </plugins>
127 </build>
128
129 </project>

```

### 3.1.2、itcast-tanhua-sso

该工程是实现单点登录的，为前端提供接口服务。

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5 http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <parent>
7         <artifactId>itcast-tanhua</artifactId>
8         <groupId>cn.itcast.tanhua</groupId>
9         <version>1.0-SNAPSHOT</version>
10    </parent>
11    <modelVersion>4.0.0</modelVersion>
12
13    <artifactId>itcast-tanhua-sso</artifactId>
14
15    <dependencies>
16        <dependency>
17            <groupId>org.springframework.boot</groupId>
18            <artifactId>spring-boot-starter-web</artifactId>
19        </dependency>
20        <dependency>
21            <groupId>com.baomidou</groupId>
22            <artifactId>mybatis-plus</artifactId>
23        </dependency>
24        <dependency>
25            <groupId>com.baomidou</groupId>
26            <artifactId>mybatis-plus-boot-starter</artifactId>
27            <version>${mybatis.mybatis-plus}</version>
28        </dependency>
29        <dependency>
30            <groupId>mysql</groupId>
31            <artifactId>mysql-connector-java</artifactId>
32        </dependency>
33        <dependency>
34            <groupId>com.alibaba</groupId>
35            <artifactId>druid</artifactId>
36        </dependency>
37        <dependency>
38            <groupId>org.projectlombok</groupId>

```

```

38         <artifactId>lombok</artifactId>
39     </dependency>
40     <dependency>
41         <groupId>org.apache.commons</groupId>
42         <artifactId>commons-lang3</artifactId>
43     </dependency>
44     <dependency>
45         <groupId>com.fasterxml.jackson.core</groupId>
46         <artifactId>jackson-databind</artifactId>
47     </dependency>
48     <dependency>
49         <groupId>org.springframework.boot</groupId>
50         <artifactId>spring-boot-starter-data-redis</artifactId>
51     </dependency>
52     <dependency>
53         <groupId>commons-codec</groupId>
54         <artifactId>commons-codec</artifactId>
55     </dependency>
56
57
58     <!--RocketMQ相关-->
59     <dependency>
60         <groupId>org.apache.rocketmq</groupId>
61         <artifactId>rocketmq-spring-boot-starter</artifactId>
62         <version>2.0.0</version>
63     </dependency>
64     <dependency>
65         <groupId>org.apache.rocketmq</groupId>
66         <artifactId>rocketmq-client</artifactId>
67         <version>4.3.2</version>
68     </dependency>
69   </dependencies>
70
71 </project>

```

### 3.1.3、部署基础服务

使用docker进行部署，采用Ubuntu操作系统。

```

1 #部署Redis集群，该集群有3个节点
2 docker create --name redis-node01 --net host -v redis-node01:/data
redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-01.conf
--port 6379
3
4 docker create --name redis-node02 --net host -v redis-node02:/data
redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-02.conf
--port 6380
5
6 docker create --name redis-node03 --net host -v redis-node03:/data
redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-03.conf
--port 6381
7
8 #启动容器
9 docker start redis-node01 redis-node02 redis-node03
10
11 #进入redis-node01容器进行操作
12 docker exec -it redis-node01 /bin/bash

```

```

13 #组件集群
14 redis-cli --cluster create 192.168.31.81:6379 192.168.31.81:6380
15 192.168.31.81:6381 --cluster-replicas 0
16
17 #查询集群信息
18
19 127.0.0.1:6379> CLUSTER NODES
20 4f4fddc825e2387783ffff9c972409b264e4df5d5 192.168.31.81:6381@16381 master -
21 0 1563956537241 3 connected 10923-16383
22 0616e00533a16e931f8dfb2e8844c35ca5721dc8 192.168.31.81:6380@16380 master -
23 0 1563956538243 2 connected 5461-10922
24 498b986e07731cead17ad1c62aa95dba6513c7b0 192.168.31.81:6379@16379
25 myself, master - 0 1563956537000 1 connected 0-5460

```

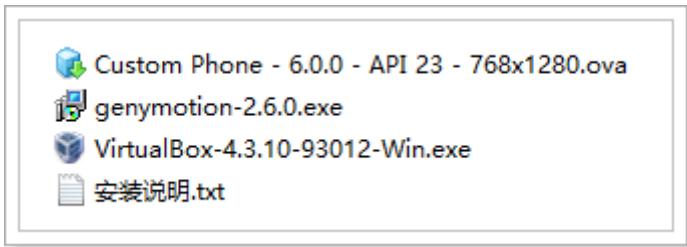
```

1 #部署RocketMQ
2 #拉取镜像
3 docker pull foxiswho/rocketmq:server-4.3.2
4 docker pull foxiswho/rocketmq:broker-4.3.2
5
6 #创建nameserver容器
7 docker create -p 9876:9876 --name rmqserver \
8 -e "JAVA_OPTS=-Duser.home=/opt" \
9 -v rmqserver-logs:/opt/logs \
10 -v rmqserver-store:/opt/store \
11 foxiswho/rocketmq:server-4.3.2
12
13 #创建broker.conf文件
14 vim /itcast/rmq/rmqbroker/conf/broker.conf
15 brokerIP1=192.168.31.81
16 namesrvAddr=192.168.31.81:9876
17 brokerName=broker_tanhua
18
19 #创建broker容器
20 docker create -p 10911:10911 -p 10909:10909 --name rmqbroker \
21 -e "JAVA_OPTS=-Duser.home=/opt" \
22 -e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
23 -v /itcast/rmq/rmqbroker/conf/broker.conf:/etc/rocketmq/broker.conf \
24 -v rmqbroker-logs:/opt/logs \
25 -v rmqbroker-store:/opt/store \
26 foxiswho/rocketmq:broker-4.3.2
27
28 #启动容器
29 docker start rmqserver rmqbroker
30
31 #停止删除容器
32 docker stop rmqbroker rmqserver
33 docker rm rmqbroker rmqserver
34
35 #部署RocketMQ的管理工具
36 docker pull styletang/rocketmq-console-ng:1.0.0
37
38 #创建并启动容器
39 docker run -e "JAVA_OPTS=-Drocketmq.namesrv.addr=192.168.31.81:9876 -Dcom.rocketmq.sendMessageWithVIPChannel=false" -p 8082:8080 -t
40 styletang/rocketmq-console-ng:1.0.0

```

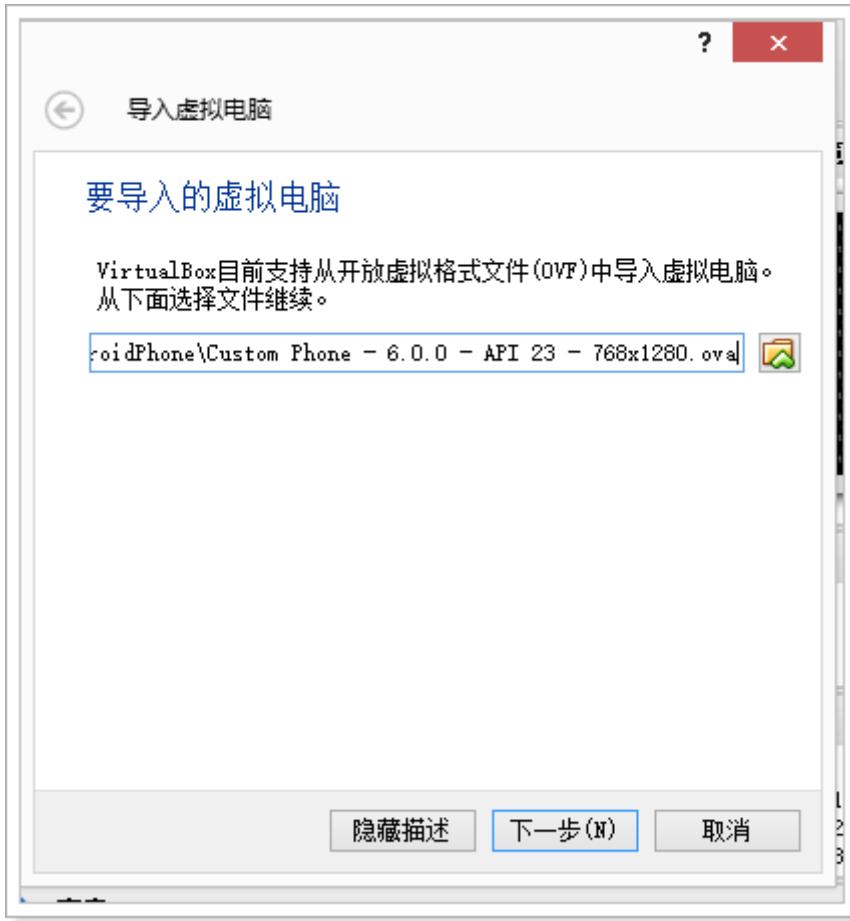
### 3.1.4、搭建客户端

客户端由前端团队进行开发，前端提供apk进行对接，所以，需要我们安装安卓的模拟器进行测试。

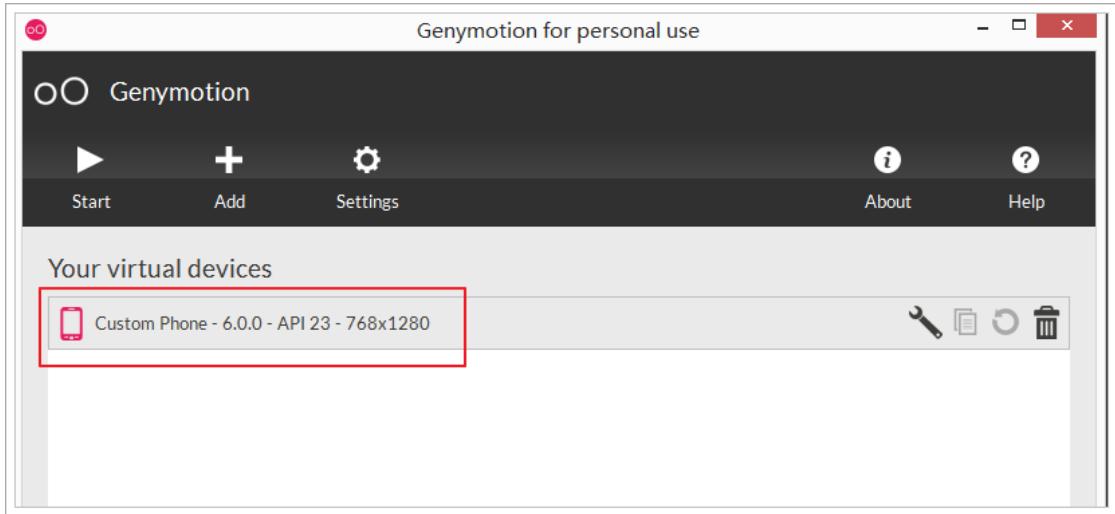


安装步骤：

- 首先安装VirtualBox-4.3.10-93012-Win.exe
- 其次安装genymotion-2.6.0.exe
- 安装完成后，打开VirtualBox，将Custom Phone - 6.0.0 - API 23 - 768x1280.ova导入到虚拟机



- 启动模拟器





下面尝试安装apk，将资料中的 tanhua-test.apk 拖拽到模拟器窗口进行安装，安装完成：



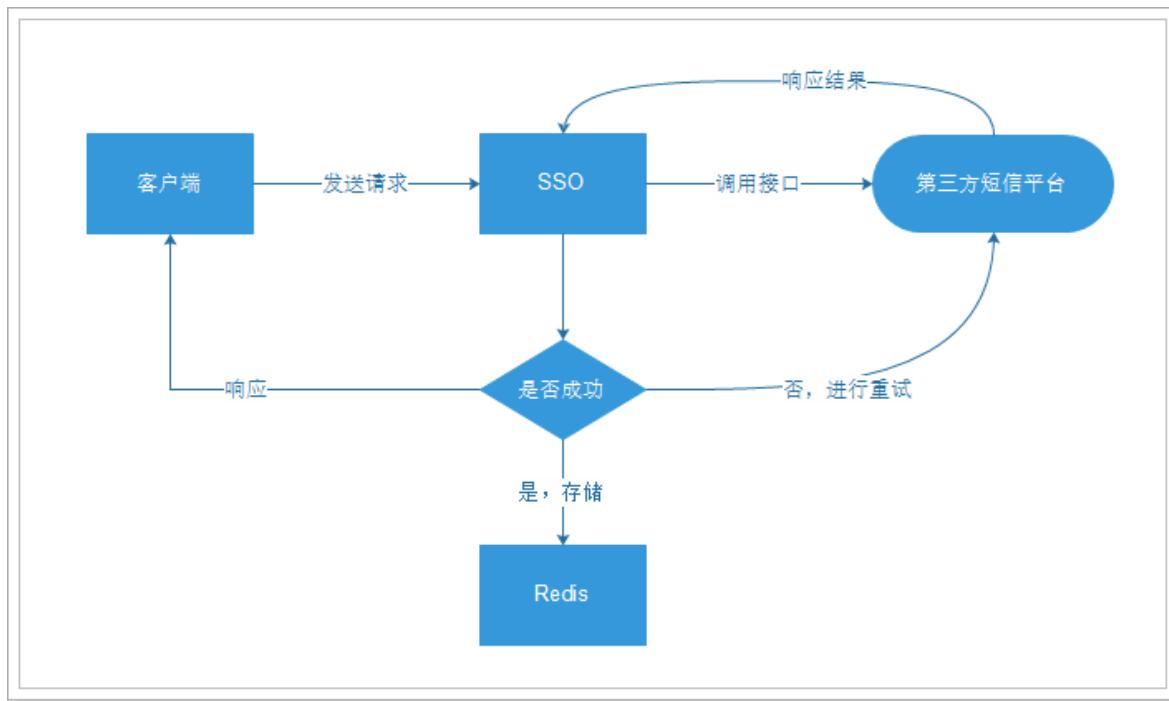
### 3.2、实现分析

需要提供4个接口服务：

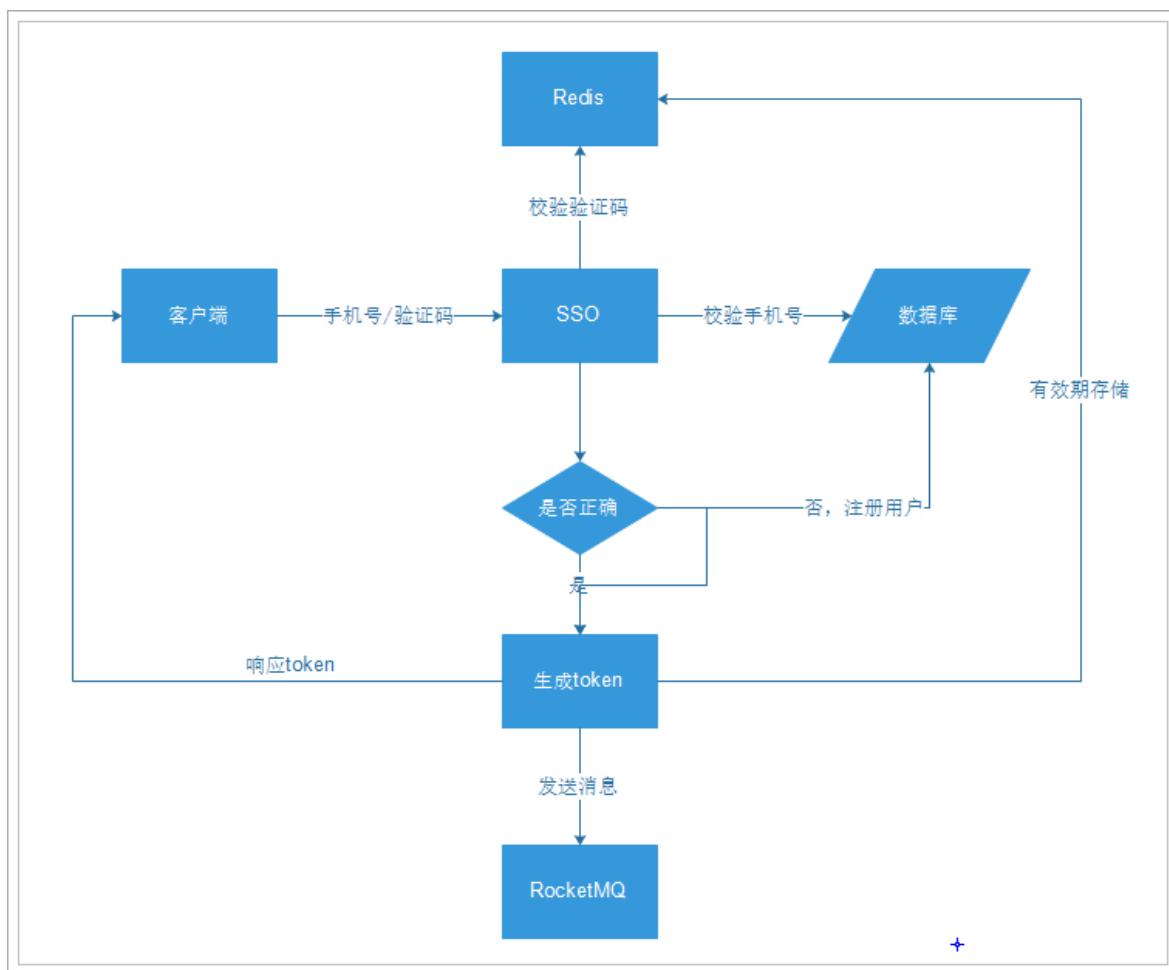
- 发送手机验证码
  - 需要对接第三方短信平台进行发送验证码
  - 选用云之讯平台进行发送 <https://www.ucpaas.com/> (选用其他接口也可以)
- 校验用户登录
  - 后台需要验证手机号与验证码是否正确
  - 校验成功之后，需要按照JWT规范进行返回响应
- 首次登录完善个人信息

- 校验token是否有效
  - 校验存储到Redis中的token是否有效

发送手机验证码流程：



校验用户登录流程：



### 3.3、数据库表

数据库使用的mysql：

```

1 CREATE TABLE `tb_user` (
2     `id` bigint(20) NOT NULL AUTO_INCREMENT,
3     `mobile` varchar(11) DEFAULT NULL COMMENT '手机号',
4     `password` varchar(32) DEFAULT NULL COMMENT '密码, 需要加密',
5     `created` datetime DEFAULT NULL,
6     `updated` datetime DEFAULT NULL,
7     PRIMARY KEY (`id`),
8     KEY `mobile` (`mobile`) USING BTREE
9 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='用户表';
10
11 CREATE TABLE `tb_user_info` (
12     `id` bigint(20) NOT NULL AUTO_INCREMENT,
13     `user_id` bigint(20) NOT NULL COMMENT '用户id',
14     `nick_name` varchar(50) DEFAULT NULL COMMENT '昵称',
15     `logo` varchar(100) DEFAULT NULL COMMENT '用户头像',
16     `tags` varchar(50) DEFAULT NULL COMMENT '用户标签: 多个用逗号分隔',
17     `sex` tinyint(1) DEFAULT '3' COMMENT '性别, 1-男, 2-女, 3-未知',
18     `age` int(11) DEFAULT NULL COMMENT '用户年龄',
19     `edu` varchar(20) DEFAULT NULL COMMENT '学历',
20     `city` varchar(20) DEFAULT NULL COMMENT '居住城市',
21     `birthday` varchar(20) DEFAULT NULL COMMENT '生日',
22     `cover_pic` varchar(50) DEFAULT NULL COMMENT '封面图片',
23     `industry` varchar(20) DEFAULT NULL COMMENT '行业',
24     `income` varchar(20) DEFAULT NULL COMMENT '收入',
25     `marriage` varchar(20) DEFAULT NULL COMMENT '婚姻状态',
26     `created` datetime DEFAULT NULL,
27     `updated` datetime DEFAULT NULL,
28     PRIMARY KEY (`id`),
29     KEY `user_id` (`user_id`)
30 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户信息表';
31

```

## 3.4、编写配置

application.properties :

```

1 spring.application.name = itcast-tanhua-sso
2
3 server.port = 18080
4
5 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
6 spring.datasource.url=jdbc:mysql://127.0.0.1:3306/tanhua?
useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries
=true&useSSL=false
7 spring.datasource.username=root
8 spring.datasource.password=root
9
10
11 # 枚举包扫描
12 mybatis-plus.type-enums-package=com.tanhua.sso.enums
13 # 表名前缀
14 mybatis-plus.global-config.db-config.table-prefix=tb_
15 # id策略为自增长
16 mybatis-plus.global-config.db-config.id-type=auto
17
18

```

```
19 # Redis相关配置
20 spring.redis.jedis.pool.max-wait = 5000ms
21 spring.redis.jedis.pool.max-idle = 100
22 spring.redis.jedis.pool.min-idle = 10
23 spring.redis.timeout = 10s
24 spring.redis.cluster.nodes =
25     192.168.31.81:6379,192.168.31.81:6380,192.168.31.81:6381
26     spring.redis.cluster.max-redirects=5
27
28 # RocketMQ相关配置
29 spring.rocketmq.nameServer=192.168.31.81:9876
30     spring.rocketmq.producer.group=tanhua
31
32 #itcast_tanhua
33 jwt.secret=76bd425b6f29f7fcc2e0bfc286043df1
34
35 #虹软相关配置
36 arcsoft.appid=*****
37 arcsoft.sdkKey=*****
38 arcsoft.libPath=F:\\code\\WIN64
```

## 3.5、编写基础代码

### 3.5.1、SexEnum

用户的性别用枚举进行表示。

```
1 package com.tanhua.sso.enums;
2
3 import com.baomidou.mybatisplus.core.enums.IEnum;
4
5 public enum SexEnum implements IEnum<Integer> {
6
7     MAN(1,"男"),
8     WOMAN(2,"女"),
9     UNKNOWN(3,"未知");
10
11     private int value;
12     private String desc;
13
14     SexEnum(int value, String desc) {
15         this.value = value;
16         this.desc = desc;
17     }
18
19     @Override
20     public Integer getValue() {
21         return this.value;
22     }
23
24     @Override
25     public String toString() {
26         return this.desc;
27     }
28 }
```

### 3.5.2、User、UserInfo

```
1 package com.tanhua.sso.pojo;
2
3 import com.baomidou.mybatisplus.annotation.FieldFill;
4 import com.baomidou.mybatisplus.annotation.TableField;
5
6 import java.util.Date;
7
8
9 public abstract class BasePojo {
10
11     @TableField(fill = FieldFill.INSERT) //自动填充
12     private Date created;
13     @TableField(fill = FieldFill.INSERT_UPDATE)
14     private Date updated;
15 }
16
```

```
1 package com.tanhua.sso.pojo;
2
3 import com.fasterxml.jackson.annotation.JsonIgnore;
4 import lombok.AllArgsConstructor;
5 import lombok.Data;
6 import lombok.NoArgsConstructor;
7
8 @Data
9 @NoArgsConstructor
10 @AllArgsConstructor
11 public class User extends BasePojo {
12
13     private Long id;
14     private String mobile; //手机号
15
16     @JsonIgnore
17     private String password; //密码, json序列化时忽略
18
19 }
20
```

```
1 package com.tanhua.sso.pojo;
2
3 import com.tanhua.sso.enums.SexEnum;
4 import lombok.AllArgsConstructor;
5 import lombok.Data;
6 import lombok.NoArgsConstructor;
7
8 @Data
9 @NoArgsConstructor
10 @AllArgsConstructor
11 public class UserInfo extends BasePojo {
12
13     private Long id;
14     private Long userId; //用户id
```

```
15     private String nickName; //昵称
16     private String logo; //用户头像
17     private String tags; //用户标签: 多个用逗号分隔
18     private SexEnum sex; //性别
19     private Integer age; //年龄
20     private String edu; //学历
21     private String city; //城市
22     private String birthday; //生日
23     private String coverPic; // 封面图片
24     private String industry; //行业
25     private String income; //收入
26     private String marriage; //婚姻状态
27
28 }
29
```

### 3.5.3、MyMetaObjectHandler

对自动填充字段的处理：

```
1 package com.tanhua.sso.handler;
2
3 import com.baomidou.mybatisplus.core.handlers.MetaObjectHandler;
4 import org.apache.ibatis.reflection.MetaObject;
5 import org.springframework.stereotype.Component;
6
7 import java.util.Date;
8
9 @Component
10 public class MyMetaObjectHandler implements MetaObjectHandler {
11
12     @Override
13     public void insertFill(MetaObject metaObject) {
14         Object created = getFieldValByName("created", metaObject);
15         if (null == created) {
16             //字段为空, 可以进行填充
17             setFieldValByName("created", new Date(), metaObject);
18         }
19
20         Object updated = getFieldValByName("updated", metaObject);
21         if (null == updated) {
22             //字段为空, 可以进行填充
23             setFieldValByName("updated", new Date(), metaObject);
24         }
25     }
26
27     @Override
28     public void updateFill(MetaObject metaObject) {
29         //更新数据时, 直接更新字段
30         setFieldValByName("updated", new Date(), metaObject);
31     }
32 }
33
```

### 3.5.4、UserMapper

```
1 package com.tanhua.sso.mapper;
2
3 import com.baomidou.mybatisplus.core.mapper.BaseMapper;
4 import com.tanhua.sso.pojo.User;
5
6 public interface UserMapper extends BaseMapper<User> {
7
8 }
```

### 3.5.5、MyApplication

SpringBoot的启动类。

```
1 package com.tanhua.sso;
2
3 import org.mybatis.spring.annotation.MapperScan;
4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6
7 @MapperScan("com.tanhua.sso.mapper") //设置mapper接口的扫描包
8 @SpringBootApplication
9 public class MyApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(MyApplication.class, args);
13     }
14 }
15 }
```

## 3.6、短信验证码

短信验证码选用云之讯第三方短信平台：<https://www.ucpaas.com/>

选择原因：注册赠送10元，无实名认证也可以发验证码短信。



### 3.6.1、创建应用

自行注册、登录后进行创建应用：



创建完成：

应用名称	上线状态	可用额度(元)
tanhua	未上线	无限制
hello-001	未上线	无限制

默认情况下，短信只能发送给自己注册的手机号，为了测试方便需要添加测试手机号（最多6个）：

This section allows managing test numbers. It features a 'Create Application' button and a 'Manage Test Numbers' button, with the latter being highlighted by a red box. Below are two rows for setting available credit: 'Unlimited' with a gear icon, and another 'Unlimited' entry with a gear icon.

**添加号码**

输入真实号码

输入格式例如：13800138000 或 075523456789

图片验证码  请输入图片验证码 

验证方式  短信验证码  或  语音验证码

输入获取的验证码  请输入验证码

确定  取消

绑定号码		操作
15800000000	44(注册手机)	--
17000000000		<input type="button"/> 删除

### 3.6.2、创建短信模板

发送短信需要创建短信模板，模板中采用{1}、{2}的形式作为参数占位。

**模板管理**

产品总览

▼ 发送短信

- 群发助手
- 发送管理

**模板库**

▼ 短信包

- 我的套餐包
- 国际费率

▼ 统计中心

- 发送详情
- 数据统计

▼ 用户群组

通讯录

**编辑模板**

模板类型  验证类模板  会员服务模板  通知类模板

适用于注册、登录、找回密码、身份验证、号码绑定身份验证的短信模板

从模板库中导入

**【探花交友】 您的验证码是：{1} (如非本人操作请忽略)。**

共 28 个字 (含签名括号)，1条短信(因含有变量,按实际发送长度计费)

为保障信息安全，ucpaas.com将为您提供完善的审核服务，帮助客户在符合法律、法规基础上使用通信服务。审核时间为：工作日的9点至18点，每30分钟审核一次，休息日（含假期）的每天12点前集中审核。

**添加签名**  探花交友

**\* 模板名称**  探花交友

模板ID	模板类型	资费项	单价	模板名称	模板内容	创建时间	审核状态	操作
487656	验证码模板			探花交友	【探花交友】您的验证码是:{1} (如非本人操作请忽略)。	2019-08-28 19:08:28	待审核 <span style="color: red;">?</span>	<span style="color: green;">编辑</span>

共有1条，每页显示 15 条 1/1

模板处于待审核状态是不可以使用的，需要审核通过后才能使用，审核通过后会有短信通知。

### 3.6.3、发送短信api

地址：<http://docs.ucpaas.com/doku.php?id=%E7%9F%AD%E4%BF%A1:sendsms>

参数：

参数	类型	约 束	描述	示例
sid	String	必 填	用户的账号唯一标识 “Account Sid” , 在开发者控制台获取 <a href="#">填 (位置)</a>	39467b989d087c2d92c6132184a365d8
token	String	必 填	用户密钥 “Auth Token” , 在开发者控制台获取 ( <a href="#">位置</a> )	23f757bad208226ec301e117e40006ed
appid	String	必 填	创建应用时系统分配的唯一标示 ( <a href="#">位置</a> )	2d92c6132139467b989d087c84a365d8
templateid	String	必 填	可在后台短信产品→选择接入的应用→短信模板-模板ID , 查看 该模板ID ( <a href="#">位置</a> )	154501
param	String	选 填	模板中的替换参数 , 如该模板不存在参数则无需传该参数或者参 数为空 , 如果有多个参数则需要写在同一个字符串中 , 以英文逗 号分隔 (如：“a,b,c” ) , 参数中不能含有特殊符 号“【】”和“”	87828,3
mobile	String	必 填	接收的单个手机号 , 暂仅支持国内号码	18011984299
uid	String	选 填	用户透传ID , 随状态报告返回	2d92c6132139467b989d087c84a365d8

响应：

JSON 响应示例			
参数	类型	描述	示例
code	String	状态码	000000
msg	String	对应状态码的信息	例如：“发送成功”，“手机号码不合法”
count	String	计费条数	1
create_date	String	创建时间,格式YYYY-MM-DD hh:mm:ss	2017-08-28 19:08:28
smsid	String	短信唯一id	f96f79240e372587e9284cd580d8f953
mobile	String	手机号码	18088888888
uid	String	请求时透传的uid	2d92c6132139467b989d087c84a365d8

返回码：[http://docs.ucpaas.com/doku.php?id=error\\_code](http://docs.ucpaas.com/doku.php?id=error_code)

### 3.6.4、编写代码

配置RestTemplateConfig:

```
1 package com.tanhua.sso.config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.http.client.ClientHttpRequestFactory;
6 import org.springframework.http.client.SimpleClientHttpRequestFactory;
7 import org.springframework.web.client.RestTemplate;
8 import org.springframework.http.converter.StringHttpMessageConverter;
9
10 import java.nio.charset.Charset;
11
12 @Configuration
13 public class RestTemplateConfig {
14
15     @Bean
16     public RestTemplate restTemplate(ClientHttpRequestFactory factory) {
17         RestTemplate restTemplate = new RestTemplate(factory);
18         // 支持中文编码
19         restTemplate.getMessageConverters().set(1, new
20 StringHttpMessageConverter(Charset.forName("UTF-8")));
21         return restTemplate;
22     }
23
24     @Bean
25     public ClientHttpRequestFactory simpleClientHttpRequestFactory() {
26         SimpleClientHttpRequestFactory factory = new
27 SimpleClientHttpRequestFactory();
28         factory.setReadTimeout(5000); //单位为ms
29         factory.setConnectTimeout(5000); //单位为ms
30         return factory;
31     }
32 }
```

编写SmsService :

```
1 package com.tanhua.sso.service;
2
3 import com.fasterxml.jackson.databind.JsonNode;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import org.apache.commons.lang3.RandomUtils;
6 import org.apache.commons.lang3.StringUtils;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.stereotype.Service;
10 import org.springframework.web.client.RestTemplate;
11
12 import java.io.IOException;
13 import java.util.HashMap;
14 import java.util.Map;
```

```
16 @Service
17 public class SmsService {
18
19     @Autowired
20     private RestTemplate restTemplate;
21
22     private static final ObjectMapper MAPPER = new ObjectMapper();
23
24
25     /**
26      * 发送验证码短信
27      *
28      * @param mobile
29      */
30     public String sendSms(String mobile) {
31         String url = "https://open.ucpaas.com/ol/sms/sendsms";
32         Map<String, Object> params = new HashMap<>();
33         params.put("sid", "*****");
34         params.put("token", "*****");
35         params.put("appid", "*****");
36         params.put("templateid", "*****");
37         params.put("mobile", mobile);
38         // 生成4位数验证
39         params.put("param", RandomUtils.nextInt(100000, 999999));
40         ResponseEntity<String> responseEntity =
41             this.restTemplate.postForEntity(url, params, String.class);
42
43         String body = responseEntity.getBody();
44
45         try {
46             JsonNode jsonNode = MAPPER.readTree(body);
47             //000000 表示发送成功
48             if (StringUtils.equals(jsonNode.get("code").textValue(),
49                 "000000")) {
50                 return String.valueOf(params.get("param"));
51             }
52         } catch (IOException e) {
53             e.printStackTrace();
54         }
55
56         return null;
57     }
58 }
59 }
```

相关参数的信息查询：

## API接口对接

**AppID** (应用ID) dd7d74e604284a6b9cc668c659 [复制](#)

**Account**  
**Sid** (用户sid) 56f6523e8f50c85fe92d5d12a8da [复制](#)

**Auth Token** (鉴权密钥) 41fabadd9a2\*\*\*\*\*48l 

**Rest URL** (请求地址) <https://open.ucpaas.com/ol/sms> [复制](#)

[SDK下载](#)[接入文档](#)[返回码查询](#)

### 3.6.5、编写测试用例

```
1 package com.tanhua.sso.service;
2
3 import org.junit.Test;
4 import org.junit.runner.RunWith;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
8
9 @SpringBootTest
10 @RunWith(SpringJUnit4ClassRunner.class)
11 public class TestSmsService {
12
13     @Autowired
14     private SmsService smsService;
15
16     @Test
17     public void sendSms(){
18         this.smsService.sendSms("158****7944");
19     }
20 }
21 }
```

### 3.6.6、mock接口

地址：<https://mock.boxuegu.com/project/164/interface/api/11011>

状态： ● 已完成      更新时间： 2019-07-29 10:42:58

接口路径： POST /user/login

Mock地址： <https://mock.boxuegu.com/mock/164/user/login>

### 请求参数

Headers :

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body:

名称	类型	是否必须	默认值	备注	其他信息
phone	string	必须		手机号	

### 返回数据

名称	类型	是否必须	默认值	备注	其他信息
暂无数据					

## 3.6.6、编写接口服务

编写ErrorResult :

```

1 package com.tanhua.sso.vo;
2
3 import lombok.Builder;
4 import lombok.Data;
5
6 @Data
7 @Builder
8 public class ErrorResult {
9
10     private String errCode;
11     private String errMessage;
12 }
13

```

SmsController :

```

1 package com.tanhua.sso.controller;
2
3 import com.tanhua.sso.service.SmsService;
4 import com.tanhua.sso.vo.ErrorResult;
5 import org.slf4j.Logger;
6 import org.slf4j.LoggerFactory;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpStatus;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.PostMapping;
11 import org.springframework.web.bind.annotation.RequestBody;
12 import org.springframework.web.bind.annotation.RequestMapping;
13 import org.springframework.web.bind.annotation.RestController;
14

```

```

15 import java.util.Map;
16
17 @RestController
18 @RequestMapping("user")
19 public class SmsController {
20
21     private static final Logger LOGGER =
LoggerFactory.getLogger(SmsController.class);
22
23     @Autowired
24     private SmsService smsService;
25
26     /**
27      * 发送验证码
28      *
29      * @return
30      */
31     @PostMapping("login")
32     public ResponseEntity<Object> sendCheckCode(@RequestBody Map<String,
Object> param) {
33         ErrorResult.ErrorResultBuilder resultBuilder =
ErrorResult.builder().errCode("000000").errMessage("发送短信验证码失败");
34         try {
35             String phone = String.valueOf(param.get("phone"));
36             Map<String, Object> sendCheckCode =
this.smsService.sendCheckCode(phone);
37             int code = ((Integer) sendCheckCode.get("code")).intValue();
38             if (code == 3) {
39                 return ResponseEntity.ok(null);
40             } else if (code == 1) {
41
resultBuilder.errCode("000001").errMessage(sendCheckCode.get("msg").toString());
42
}
43         } catch (Exception e) {
44             LOGGER.error("发送短信验证码失败", e);
45         }
46
47         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(resultBuilder.
build());
48     }
49
50 }
51

```

SmsService :

```

1 package com.tanhua.sso.service;
2
3 import com.fasterxml.jackson.databind.JsonNode;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import org.apache.commons.lang3.RandomUtils;
6 import org.apache.commons.lang3.StringUtils;
7 import org.slf4j.Logger;
8 import org.slf4j.LoggerFactory;
9 import org.springframework.beans.factory.annotation.Autowired;

```

```
10 import org.springframework.data.redis.core.RedisTemplate;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.stereotype.Service;
13 import org.springframework.web.client.RestTemplate;
14
15 import java.io.IOException;
16 import java.time.Duration;
17 import java.util.HashMap;
18 import java.util.Map;
19
20 @Service
21 public class SmsService {
22
23     private static final Logger LOGGER =
24     LoggerFactory.getLogger(SmsService.class);
25
26     @Autowired
27     private RestTemplate restTemplate;
28
29     private static final ObjectMapper MAPPER = new ObjectMapper();
30
31     @Autowired
32     private RedisTemplate<String, String> redisTemplate;
33
34     /**
35      * 发送验证码
36      *
37      * @param mobile
38      * @return
39      */
40     public Map<String, Object> sendCheckCode(String mobile) {
41         Map<String, Object> result = new HashMap<>(2);
42         try {
43             String redisKey = "CHECK_CODE_" + mobile;
44             String value = this.redisTemplate.opsForValue().get(redisKey);
45             if (StringUtils.isNotEmpty(value)) {
46                 result.put("code", 1);
47                 result.put("msg", "上一次发送的验证码还未失效");
48                 return result;
49             }
50             String code = this.sendSms(mobile);
51             if (null == code) {
52                 result.put("code", 2);
53                 result.put("msg", "发送短信验证码失败");
54                 return result;
55             }
56
57             //发送验证码成功
58             result.put("code", 3);
59             result.put("msg", "ok");
60
61             //将验证码存储到Redis,2分钟后失效
62             this.redisTemplate.opsForValue().set(redisKey, code,
63             Duration.ofMinutes(2));
64
65             return result;
66         } catch (Exception e) {
```

```

66     LOGGER.error("发送验证码出错! " + mobile, e);
67
68     result.put("code", 4);
69     result.put("msg", "发送验证码出现异常");
70     return result;
71 }
72 }
73 }
74 }
75 }
76

```

## 3.7、JWT

### 3.7.1、简介

JSON Web token简称JWT，是用于对应用程序上的用户进行身份验证的标记。也就是说，使用JWT的应用程序不再需要保存有关其用户的 cookie 或其他session数据。此特性便于可伸缩性，同时保证应用程序的安全。

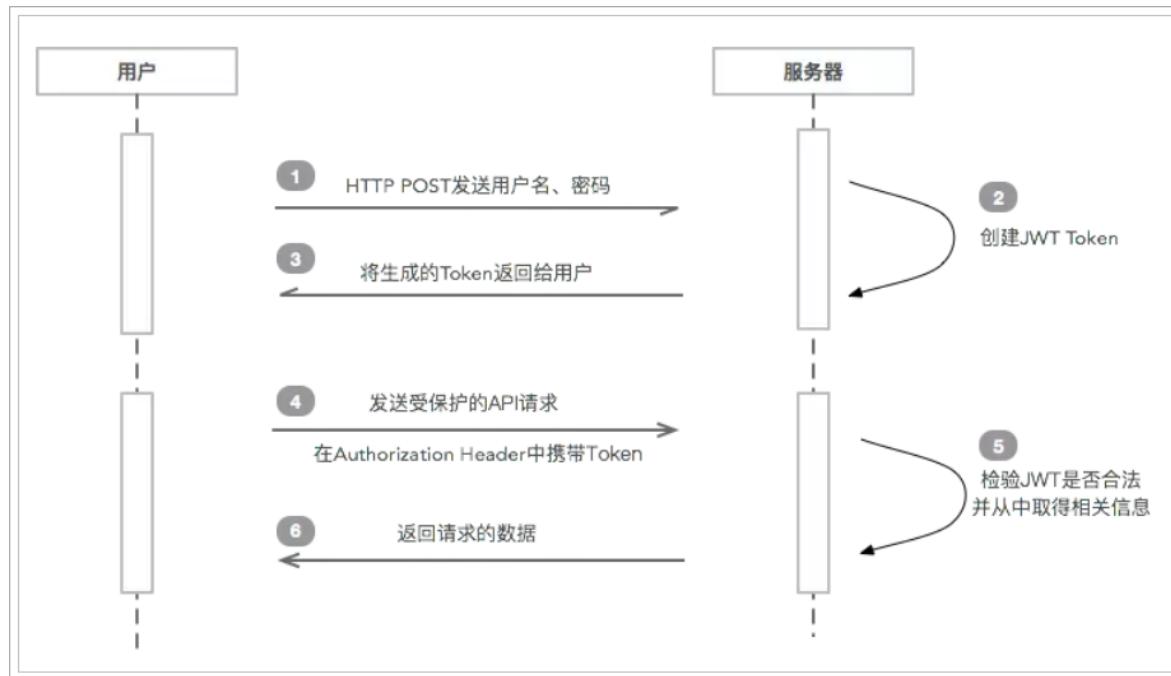
在身份验证过程中，当用户使用其凭据成功登录时，将返回 JSON Web token，并且必须在本地保存（通常在本地存储中）。

每当用户要访问受保护的路由或资源（端点）时，用户代理(user agent)必须连同请求一起发送 JWT，通常在授权标头中使用Bearer schema。后端服务器接收到带有 JWT 的请求时，首先要做的是验证token。

### 3.7.2、格式

- JWT就是一个字符串，经过加密处理与校验处理的字符串，形式为：A.B.C
- A由JWT头部信息header加密得到
- B由JWT用到的身份验证信息json数据加密得到
- C由A和B加密得到，是校验部分

### 3.7.3、流程



### 3.7.4、示例

导入依赖：

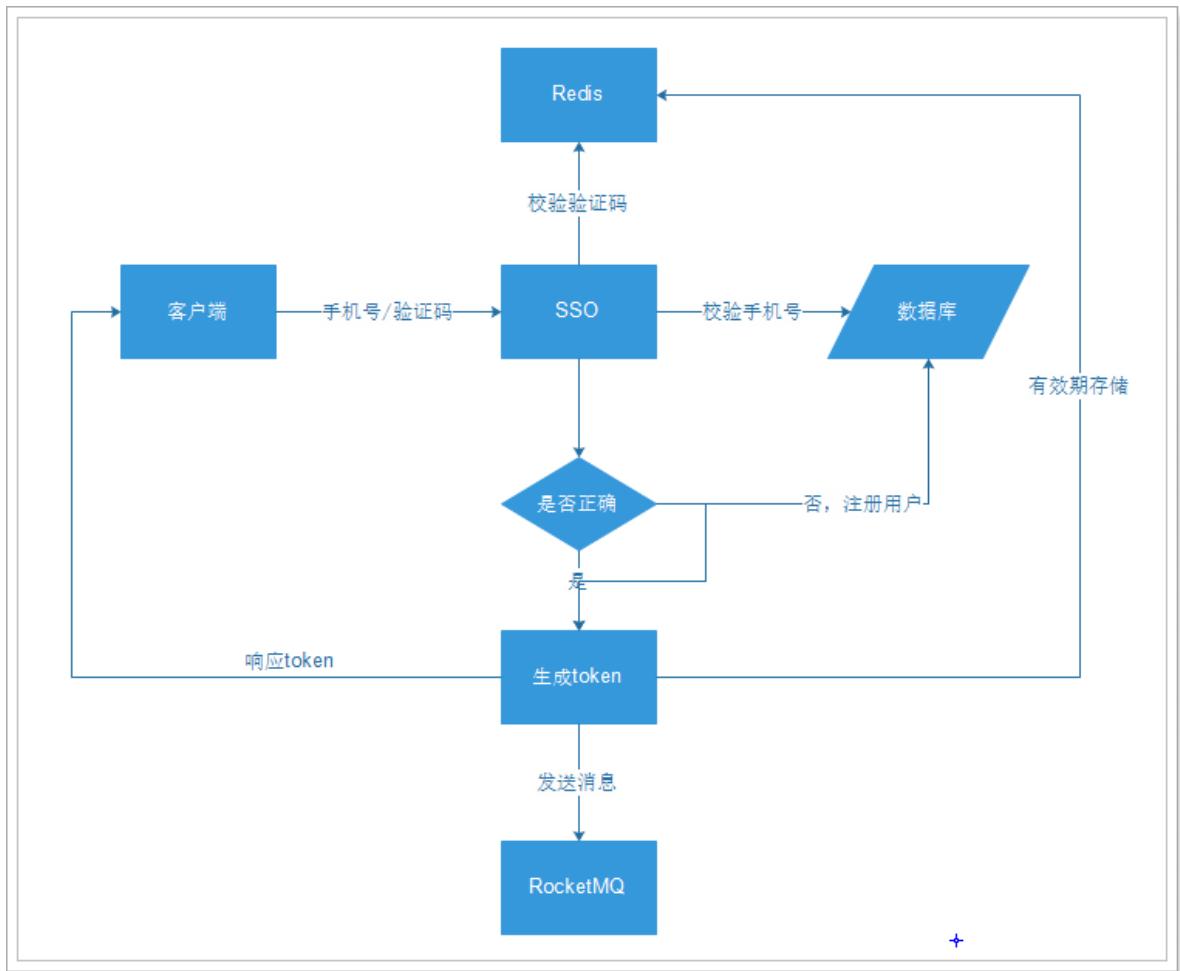
```
1 <dependency>
2   <groupId>io.jsonwebtoken</groupId>
3   <artifactId>jjwt</artifactId>
4   <version>0.9.1</version>
5 </dependency>
```

编写测试用例：

```
1 public class TestJWT {
2
3   public static void main(String[] args) {
4
5     String secret = "itcast";
6
7     Map<String, Object> claims = new HashMap<String, Object>();
8     claims.put("mobile", "12345789");
9     claims.put("id", "2");
10
11    // 生成token
12    String jwt = Jwts.builder()
13      .setClaims(claims) //设置响应数据体
14      .signWith(SignatureAlgorithm.HS256, secret) //设置加密方法和加
密盐
15      .compact();
16
17    System.out.println(jwt);
18    //eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUoIixMjM0NTc4OSIsImlkIjoIMj9.VivsfLzrsKF
19    OJo_BdGif6cKY_7wr2jMOMOIGaFt_tps
20
21    // 通过token解析数据
22    Map<String, Object> body = Jwts.parser()
23      .setSigningKey(secret)
24      .parseClaimsJws(jwt)
25      .getBody();
26
27    System.out.println(body); //{"mobile":12345789, "id":2}
}
```

## 3.8、用户登录

用户接收到验证码后，进行输入验证码，点击登录，前端系统将手机号以及验证码提交到SSO进行校验。



### 3.7.1、mock接口

接口地址：<https://mock.boxuegu.com/project/164/interface/api/12593>

Mock地址：<https://mock.boxuegu.com/mock/164/user/loginVerification>

**请求参数**

Headers :

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body:

名称	类型	是否必须	默认值	备注	其他信息
phone	string	必须		手机号	
verificationCode	string	必须		验证码	

**返回数据**

名称	类型	是否必须	默认值	备注	其他信息
token	string	必须		登录成功返回令牌Token	枚举: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOiJE1NjI4MjkzMzYsInVzZXJfaWQiOiIxIn0.Mbzn6LzsLrkVWEbhexR3lTYDZjxq/cqW11rjxDQ6Ewk
isNew	boolean	必须	true	是否新用户	

### 3.7.2、UserController

```
1 package com.tanhua.sso.controller;
2
3 import com.tanhua.sso.pojo.User;
4 import com.tanhua.sso.service.UserService;
5 import org.apache.commons.lang3.StringUtils;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.web.bind.annotation.*;
8
9 import java.util.HashMap;
10 import java.util.Map;
11
12 @RestController
13 @RequestMapping("user")
14 public class UserController {
15
16     @Autowired
17     UserService userService;
18
19
20     /**
21      * 登录
22      *
23      * @return
24      */
25     @PostMapping("loginVerification")
26     public ResponseEntity<Object> login(@RequestBody Map<String, Object>
param) {
27         try {
28             String mobile = param.get("phone").toString();
29             String code = param.get("verificationCode").toString();
30             String token = this.userService.login(mobile, code);
31
32             Map<String, Object> result = new HashMap<>(2);
33
34             if (StringUtils.isNotEmpty(token)) {
35                 String[] ss = StringUtils.split(token, '|');
36                 String isNew = ss[0];
37                 String tokenStr = ss[1];
38
39                 result.put("isNew", isNew);
40                 result.put("token", tokenStr);
41                 return ResponseEntity.ok(result);
42             }
43         } catch (Exception e) {
44             e.printStackTrace();
45         }
46
47         // 登录失败，验证码错误
48         ErrorResult errorResult =
49             ErrorResult.builder().errCode("000000").errMessage("验证码错误").build();
50         return
51             ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorResult);
52     }
53 }
```

### 3.7.3、UserService

```

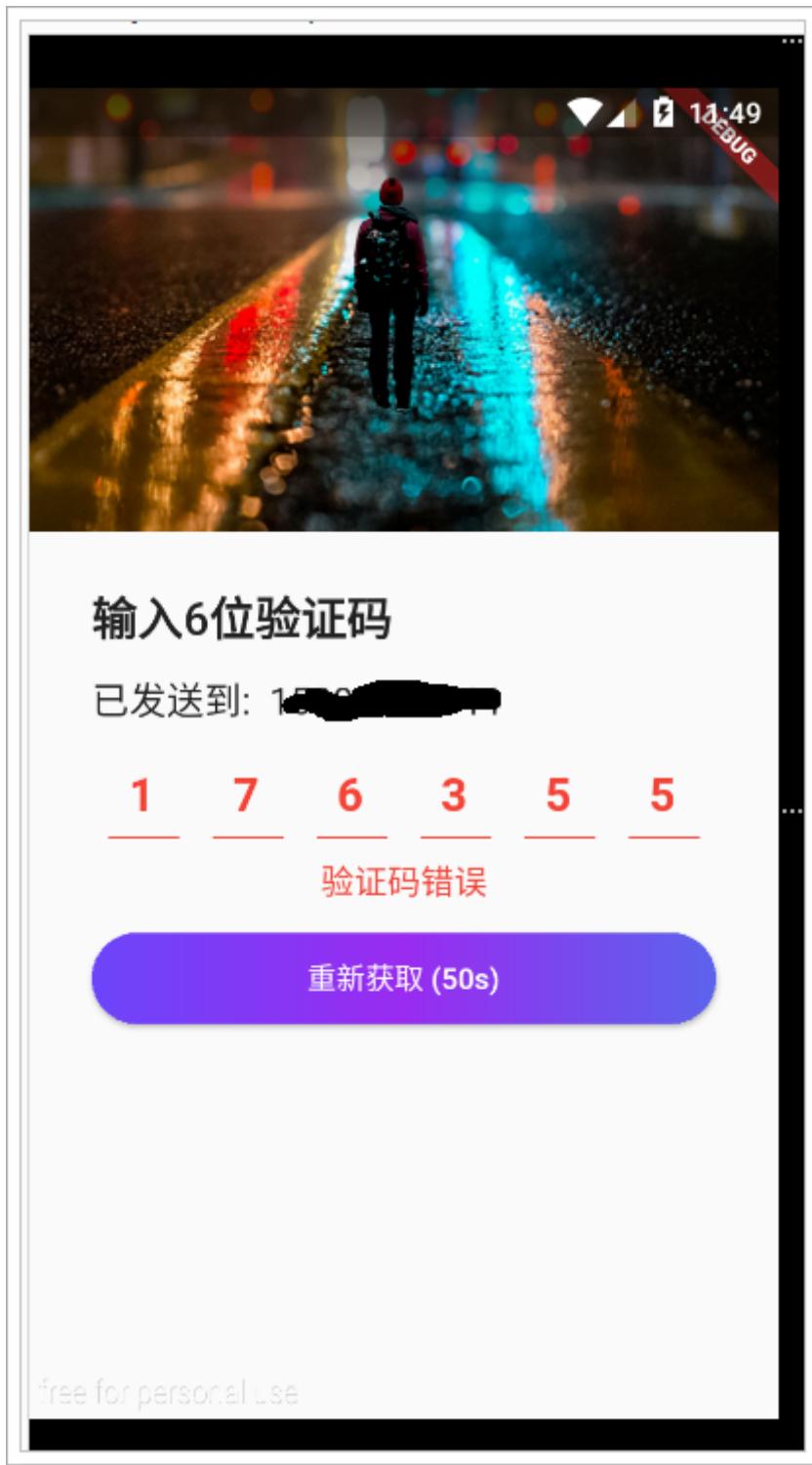
1 package com.tanhua.sso.service;
2
3 import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
4 import com.fasterxml.jackson.core.JsonProcessingException;
5 import com.fasterxml.jackson.databind.ObjectMapper;
6 import com.tanhua.sso.mapper.UserMapper;
7 import com.tanhua.sso.pojo.User;
8 import io.jsonwebtoken.Jwt;
9 import io.jsonwebtoken.SignatureAlgorithm;
10 import org.apache.commons.codec.digest.DigestUtils;
11 import org.apache.commons.lang3.StringUtils;
12 import org.apache.rocketmq.spring.core.RocketMQTemplate;
13 import org.slf4j.Logger;
14 import org.slf4j.LoggerFactory;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.beans.factory.annotation.Value;
17 import org.springframework.data.redis.core.RedisTemplate;
18 import org.springframework.stereotype.Service;
19
20 import java.io.IOException;
21 import java.time.Duration;
22 import java.util.Date;
23 import java.util.HashMap;
24 import java.util.Map;
25 import java.util.concurrent.TimeUnit;
26
27 @Service
28 public class UserService {
29
30     private static final Logger LOGGER =
31         LoggerFactory.getLogger(UserService.class);
32
33     @Autowired
34     private UserMapper userMapper;
35
36     @Autowired
37     private RedisTemplate<String, String> redisTemplate;
38
39     private static final ObjectMapper MAPPER = new ObjectMapper();
40
41     @Autowired
42     private RocketMQTemplate rocketMQTemplate;
43
44     @Value("${jwt.secret}")
45     private String secret;
46
47     public String login(String mobile, String code) {
48
49         Boolean isNew = false; //是否为新注册
50
51         //校验验证码
52         String redisKey = "CHECK_CODE_" + mobile;
53         String value = this.redisTemplate.opsForValue().get(redisKey);
54
55         if (value == null) {
56             return "验证码错误";
57         }
58
59         if (StringUtils.equals(value, code)) {
60             isNew = true;
61         }
62
63         Map<String, Object> map = new HashMap<>();
64         map.put("mobile", mobile);
65         map.put("secret", secret);
66
67         Jwt jwt = Jwt.createJWT();
68         jwt.setSubject("User");
69         jwt.setIssuedAt(new Date());
70         jwt.setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60));
71
72         String token = jwt.sign();
73
74         map.put("token", token);
75
76         String json = MAPPER.writeValueAsString(map);
77
78         return json;
79     }
80
81     public void logout(String token) {
82
83         String redisKey = "CHECK_CODE_" + mobile;
84
85         this.redisTemplate.delete(redisKey);
86
87     }
88
89     public void updateToken(String token) {
90
91         String redisKey = "CHECK_CODE_" + mobile;
92
93         this.redisTemplate.delete(redisKey);
94
95         String value = this.redisTemplate.opsForValue().get(token);
96
97         if (value == null) {
98             return;
99         }
100
101         String[] split = value.split("_");
102
103         if (split.length != 2) {
104             return;
105         }
106
107         String mobile = split[0];
108         String secret = split[1];
109
110         if (!StringUtils.equals(secret, this.secret)) {
111             return;
112         }
113
114         this.redisTemplate.delete(redisKey);
115
116         String redisKey2 = "TOKEN_" + token;
117
118         this.redisTemplate.set(redisKey2, value);
119
120     }
121
122     public void checkToken(String token) {
123
124         String redisKey = "TOKEN_" + token;
125
126         String value = this.redisTemplate.opsForValue().get(redisKey);
127
128         if (value == null) {
129             return;
130         }
131
132         String[] split = value.split("_");
133
134         if (split.length != 2) {
135             return;
136         }
137
138         String mobile = split[0];
139         String secret = split[1];
140
141         if (!StringUtils.equals(secret, this.secret)) {
142             return;
143         }
144
145         this.redisTemplate.delete(redisKey);
146
147         String redisKey2 = "CHECK_CODE_" + mobile;
148
149         this.redisTemplate.set(redisKey2, value);
150
151     }
152
153 }
```

```

53     if (!StringUtils.equals(value, code)) {
54         return null; //验证码错误
55     }
56
57     QueryWrapper<User> queryWrapper = new QueryWrapper<>();
58     queryWrapper.eq("mobile", mobile);
59     User selectUser = this.userMapper.selectOne(queryWrapper);
60     if (selectUser == null) {
61         // 该手机号未注册, 进行注册操作
62         User user = new User();
63         user.setMobile(mobile);
64         user.setPassword(DigestUtils.md5Hex(secret + "_123456")); // 默认密码
65         this.userMapper.insert(user);
66         selectUser = user;
67         isNew = true;
68     }
69
70     Map<String, Object> claims = new HashMap<String, Object>();
71     claims.put("mobile", mobile);
72     claims.put("id", selectUser.getId());
73
74     // 生成token
75     String token = Jwts.builder()
76         .setClaims(claims) //设置响应数据体
77         .signWith(SignatureAlgorithm.HS256, secret) //设置加密方法和
加密盐
78         .compact();
79
80     //将用户数据写入到redis中
81     String redisTokenKey = "TOKEN_" + token;
82     try {
83         this.redisTemplate.opsForValue().set(redisTokenKey,
MAPPER.writeValueAsString(selectUser), Duration.ofHours(1));
84     } catch (JsonProcessingException e) {
85         e.printStackTrace();
86     }
87
88     try {
89         // 发送登录成功的消息
90         Map<String, Object> msg = new HashMap<>();
91         msg.put("userId", selectUser.getId());
92         msg.put("date", new Date());
93         this.rocketMQTemplate.convertAndSend("tanhua-sso-login", msg);
94     } catch (Exception e) {
95         e.printStackTrace();
96     }
97     return isNew + "|" + token;
98 }
99
100}

```

### 3.7.4、测试



## 4、完善个人信息

用户在首次登录时需要完善个人信息，包括性别、昵称、生日、城市、头像等。

其中，头像数据需要做图片上传，这里采用阿里云的OSS服务作为我们的图片服务器，并且对头像要做人脸识别，非人脸照片不得上传。

### 4.1、图片上传

#### 4.1.1、导入依赖

```
1 <dependency>
2   <groupId>com.aliyun.oss</groupId>
3   <artifactId>aliyun-sdk-oss</artifactId>
4   <version>2.8.3</version>
5 </dependency>
```

## 4.1.2、OSS配置

需要购买阿里云服务，购买方法此处省略。

aliyun.properties：

```
1 aliyun.endpoint = http://oss-cn-shanghai.aliyuncs.com
2 aliyun.accessKeyId = LTAIqvQNjLEmJEoa
3 aliyun.accessKeySecret = nhB33fbK3LSiHj2JgTFC6TbFSCfgaG
4 aliyun.bucketName=itcast-tanhua
5 aliyun.urlPrefix=http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/
```

AliyunConfig：

```
1 package com.tanhua.sso.config;
2
3 import com.aliyun.oss.OSSClient;
4 import lombok.Data;
5 import org.springframework.boot.context.properties.ConfigurationProperties;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8 import org.springframework.context.annotation.PropertySource;
9
10 @Configuration
11 @PropertySource("classpath:aliyun.properties")
12 @ConfigurationProperties(prefix = "aliyun")
13 @Data
14 public class AliyunConfig {
15
16     private String endpoint;
17     private String accessKeyId;
18     private String accessKeySecret;
19     private String bucketName;
20     private String urlPrefix;
21
22     @Bean
23     public OSSClient ossclient() {
24         return new OSSClient(endpoint, accessKeyId, accessKeySecret);
25     }
26
27 }
```

## 4.1.3、PicUploadService

```
1 package com.tanhua.sso.service;
2
3 import com.aliyun.oss.OSSClient;
4 import com.tanhua.sso.config.AliyunConfig;
5 import com.tanhua.sso.vo.PicUploadResult;
```

```
6 import org.apache.commons.lang3.RandomUtils;
7 import org.apache.commons.lang3.StringUtils;
8 import org.joda.time.DateTime;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Service;
11 import org.springframework.web.multipart.MultipartFile;
12
13 import java.io.ByteArrayInputStream;
14
15 @Service
16 public class PicuploadService {
17
18     // 允许上传的格式
19     private static final String[] IMAGE_TYPE = new String[]{"bmp", ".jpg",
20         ".jpeg", ".gif", ".png"};
21
22     @Autowired
23     private OSSClient ossClient;
24
25     @Autowired
26     private AliyunConfig aliyunConfig;
27
28     public PicuploadResult upload(MultipartFile uploadFile) {
29
30         PicuploadResult fileuploadResult = new PicuploadResult();
31
32         //图片做校验，对后缀名
33         boolean isLegal = false;
34
35         for (String type : IMAGE_TYPE) {
36             if
37                 (StringUtils.endsWithIgnoreCase(uploadFile.getOriginalFilename(),
38                     type)) {
39                     isLegal = true;
40                     break;
41                 }
42
43         if (!isLegal) {
44             fileuploadResult.setStatus("error");
45             return fileuploadResult;
46         }
47
48         // 文件新路径
49         String fileName = uploadFile.getOriginalFilename();
50         String filePath = getFilePath(fileName);
51
52         // 上传到阿里云
53         try {
54             // 目录结构: images/2018/12/29/xxxx.jpg
55             ossClient.putObject(aliyunConfig.getBucketName(), filePath, new
56                 ByteArrayInputStream(uploadFile.getBytes()));
57         } catch (Exception e) {
58             e.printStackTrace();
59             //上传失败
60             fileuploadResult.setStatus("error");
61             return fileuploadResult;
62         }
63     }
64 }
```

```

63     // 上传成功
64     fileUploadResult.setStatus("done");
65     fileUploadResult.setName(this.aliyunConfig.getUrlPrefix() +
66     filePath);
67
68     fileUploadResult.setuid(String.valueOf(System.currentTimeMillis()));
69
70     return fileUploadResult;
71 }
72
73 private String getFilePath(String sourceFileName) {
74     DateTime dateTime = new DateTime();
75     return "images/" + dateTime.toString("yyyy")
76         + "/" + dateTime.toString("MM") + "/"
77         + dateTime.toString("dd") + "/" +
78     System.currentTimeMillis() +
79         RandomUtils.nextInt(100, 9999) + "." +
80         StringUtils.substringAfterLast(sourceFileName, ".");
81 }
82

```

所需其他的代码：

PicUploadResult:

```

1 package com.tanhua.sso.vo;
2
3 import lombok.Data;
4
5 @Data
6 public class PicuploadResult {
7
8     // 文件唯一标识
9     private String uid;
10    // 文件名
11    private String name;
12    // 状态有: uploading done error removed
13    private String status;
14    // 服务端响应内容, 如: '{"status": "success"}'
15    private String response;
16
17 }
18

```

#### 4.1.4、PicUploadController

```

1 package com.tanhua.sso.controller;
2
3 import com.tanhua.sso.service.PicUploadService;
4 import com.tanhua.sso.vo.PicUploadResult;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.PostMapping;

```

```
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestParam;
10 import org.springframework.web.bind.annotation.ResponseBody;
11 import org.springframework.web.multipart.MultipartFile;
12
13 @RequestMapping("pic/upload")
14 @Controller
15 public class PicUploadController {
16
17     @Autowired
18     private PicUploadService picuploadService;
19
20     @PostMapping
21     @ResponseBody
22     public PicUploadResult upload(@RequestParam("file") MultipartFile
multipartFile) {
23         return this.picuploadService.upload(multipartFile);
24     }
25 }
26
```

#### 4.1.5、测试

http://127.0.0.1:18080/pic/upload

POST

Raw Form Headers

Add new file field

选择文件 u358.png file x u358.png (111.2 KB)

application/x-www-form-urlencoded Set "Content-Type" header to overwrite this value.

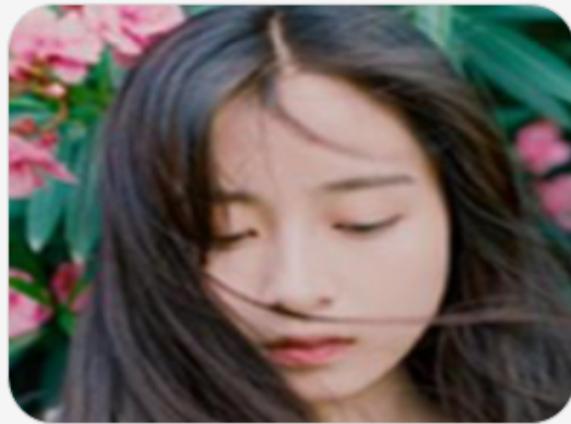
Raw JSON Response

Copy to clipboard Save as file

```
{
    uid: "1563978726955"
    name: "http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/07/24/1563978726513810.png"
    status: "done"
    response: null
}
```

详情

⑦ 快速使用图片服务 



文件名 images/2019/07/24/1563978726513810.png

ETag 565CBA3BF42E17783A9174AF38AF6EAA

使用 HTTPS 

URL https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/07/24/1563978726513810.png

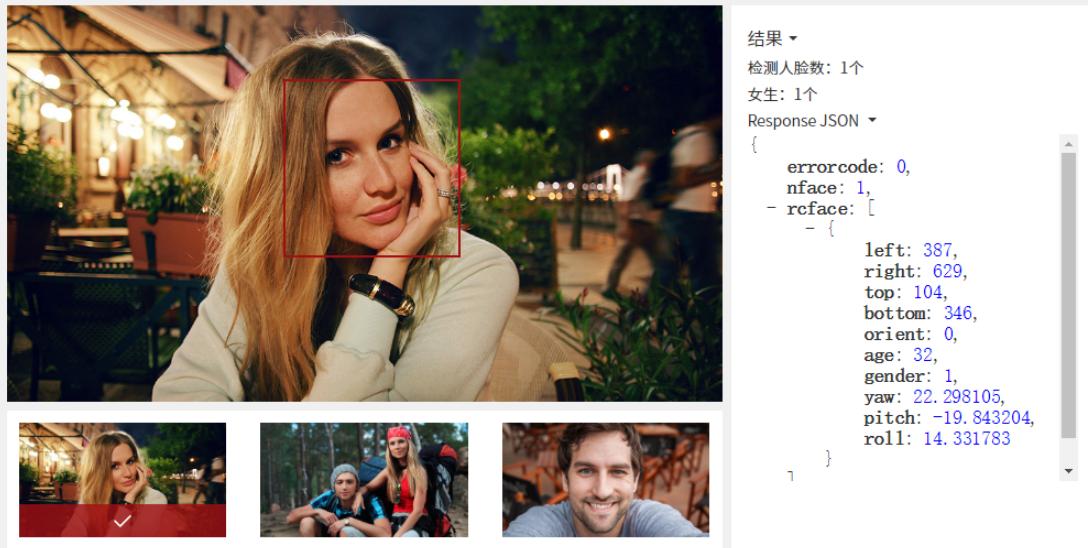
[下载](#) | [打开文件 URL](#) | [复制文件 URL](#) | [复制文件路径](#)

类型 image/png

[设置 HTTP 头](#)

## 4.2、人脸识别

## 功能演示



### 4.2.1、使用说明

使用虹软平台需要先注册开发者账号：<https://ai.arcsoft.com.cn/ucenter/user/userlogin>



注册完成后进行登录，然后进行创建应用：

### 添加应用

\* 应用名称:

\* 应用场景:

\* 应用行业:

应用描述:

**立即创建** **取消**

创建完成后，需要进行实名认证，否则相关的SDK是不能使用的。

### 账号管理

#### 基本信息



用户名 : 用户55254425 [编辑](#)

用户名ID : 55254425

电话号码 : 176\*\*\*\*9189 [编辑](#)

邮箱 : [编辑](#)

[修改密码](#)

#### 实名认证

 您的虹软账号已经通过个人认证！ [查看认证信息](#) [升级到企业认证>](#) [?](#)

实名认证后即可下载对应平台的SDK，我们需要下载windows以及linux平台。

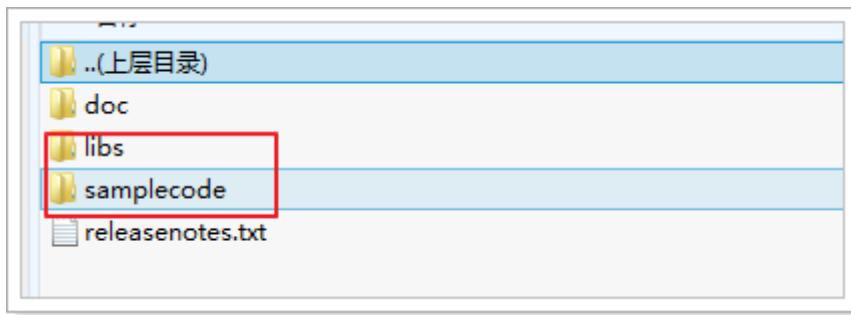
探花交友 APP ID : 2yKexS8BDMHdNTzwJpcT3uxujeoTRbP7AReyJxv7K...  
APP应用-娱乐/社交 创建时间 : 2019-07-18 添  
加SDK

ArcFace Windows(X64)	下载其他版本
添加时间:2019-07-19	
版本号 语言 最近下载 SDK KEY	下载SDK
v2.2 Java 2019-07-19 (下载起1年到期,请重新下载) DstQj1yAARVbH1oed4HbaFifVoiMPbQo3hHnSi2Ss...	

ArcFace Linux64	下载其他版本
添加时间:2019-07-19	
版本号 语言 最近下载 SDK KEY	下载SDK
v2.2 Java 2019-07-19 (下载起1年到期,请重新下载) DstQj1yAARVbH1oed4HbaFifMh99SUtSQDUhT6Lp7...	

打开解压包，可以看到有提供相应的jar包以及示例代码：



## 4.2.2、安装jar到本地仓库

进入到libs目录，需要将arcsoft-sdk-face-2.2.0.1.jar安装到本地仓库：

```
1 mvn install:install-file -DgroupId=com.arcsoft.face -DartifactId=arcsoft-sdk-face -Dversion=2.2.0.1 -Dpackaging=jar -Dfile=arcsoft-sdk-face-2.2.0.1.jar
```

安装成功后，即可通过maven坐标引用了：

```
1 <dependency>
2   <groupId>com.arcsoft.face</groupId>
3   <artifactId>arcsoft-sdk-face</artifactId>
4   <version>2.2.0.1</version>
5 </dependency>
```

## 4.2.3、开始使用

说明：虹软的SDK是免费使用的，但是首次使用时需要联网激活，激活后可离线使用。使用周期为1年，1年后需要联网再次激活。

配置：application.properties

```
1 #虹软相关配置(在虹软应用中找到对应的参数)
2 arcsoft.appid=*****
3 arcsoft.sdkKey=*****
4 arcsoft.libPath=F:\\code\\WIN64
```

FaceEngineService :

```
1 package com.tanhua.sso.service;
2
3 import com.arcsoft.face.EngineConfiguration;
4 import com.arcsoft.face.FaceEngine;
5 import com.arcsoft.face.FaceInfo;
6 import com.arcsoft.face.FunctionConfiguration;
7 import com.arcsoft.face.enums.DetectMode;
8 import com.arcsoft.face.enums.DetectOrient;
9 import com.arcsoft.face.enums.ErrorInfo;
10 import com.arcsoft.face.enums.ImageFormat;
11 import com.arcsoft.face.toolkit.ImageFactory;
12 import com.arcsoft.face.toolkit.ImageInfo;
13 import org.slf4j.Logger;
14 import org.slf4j.LoggerFactory;
15 import org.springframework.beans.factory.annotation.Value;
16 import org.springframework.stereotype.Service;
17
18 import javax.annotation.PostConstruct;
19 import java.io.File;
20 import java.util.ArrayList;
21 import java.util.List;
22
23 @Service
24 public class FaceEngineService {
25
26     private static final Logger LOGGER =
27     LoggerFactory.getLogger(FaceEngineService.class);
28
29     @Value("${arcsoft.appid}")
30     private String appid;
31
32     @Value("${arcsoft.sdkKey}")
33     private String sdkKey;
34
35     @Value("${arcsoft.libPath}")
36     private String libPath;
37
38     private FaceEngine faceEngine;
39
40     @PostConstruct
41     public void init() {
42         // 激活并且初始化引擎
43         FaceEngine faceEngine = new FaceEngine(libPath);
44         int activeCode = faceEngine.activeOnline(appid, sdkKey);
45         if (activeCode != ErrorInfo.MOK.getValue() && activeCode != ErrorInfo.MERR ASF_ALREADY_ACTIVATED.getValue()) {
46             LOGGER.error("引擎激活失败");
47             throw new RuntimeException("引擎激活失败");
48         }
49
50         //引擎配置
51         EngineConfiguration engineConfiguration = new
52         EngineConfiguration();
53         //IMAGE检测模式，用于处理单张的图像数据
54
55         engineConfiguration.setDetectMode(DetectModeASF_DETECT_MODE_IMAGE);
56         //人脸检测角度，逆时针0度
```

```

54     engineConfiguration.setDetectFaceOrientPriority(DetectOrient.ASF_OP_0_ONL
Y);
55
56     //功能配置
57     FunctionConfiguration functionConfiguration = new
FunctionConfiguration();
58     functionConfiguration.setSupportAge(true);
59     functionConfiguration.setSupportFace3dAngle(true);
60     functionConfiguration.setSupportFaceDetect(true);
61     functionConfiguration.setSupportFaceRecognition(true);
62     functionConfiguration.setSupportGender(true);
63     functionConfiguration.setSupportLiveness(true);
64     functionConfiguration.setSupportIRLiveness(true);
65
66     engineConfiguration.setFunctionConfiguration(functionConfiguration);
67
68     //初始化引擎
69     int initCode = faceEngine.init(engineConfiguration);
70
71     if (initCode != ErrorInfo.MOK.getValue()) {
72         LOGGER.error("初始化引擎出错!");
73         throw new RuntimeException("初始化引擎出错!");
74     }
75
76     this.faceEngine = faceEngine;
77 }
78 /**
79 * 检测图片是否为人像
80 *
81 * @param imageInfo 图像对象
82 * @return true:人像, false:非人像
83 */
84 public boolean checkIsPortrait(ImageInfo imageInfo) {
85     // 定义人脸列表
86     List<FaceInfo> faceInfoList = new ArrayList<FaceInfo>();
87     faceEngine.detectFaces(imageInfo.getImageData(),
imageInfo.getWidth(), imageInfo.getHeight(), ImageFormat.CP_PAF_BGR24,
faceInfoList);
88     return !faceInfoList.isEmpty();
89 }
90
91 public boolean checkIsPortrait(byte[] imageData) {
92     return this.checkIsPortrait(ImageFactory.getRGBData(imageData));
93 }
94
95 public boolean checkIsPortrait(File file) {
96     return this.checkIsPortrait(ImageFactory.getRGBData(file));
97 }
98
99 }
100

```

#### 4.2.4、测试

```
1 package com.tanhua.sso.service;
```

```

2
3 import org.junit.Test;
4 import org.junit.runner.RunWith;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
8
9 import java.io.File;
10
11 @SpringBootTest
12 @RunWith(SpringJUnit4ClassRunner.class)
13 public class TestFaceEngineService {
14
15     @Autowired
16     private FaceEngineService faceEngineService;
17
18     @Test
19     public void testCheckIsPortrait(){
20         File file = new File("F:\\1.jpg");
21         boolean checkIsPortrait =
22             this.faceEngineService.checkIsPortrait(file);
23         System.out.println(checkIsPortrait); // true|false
24     }
25 }
```

## 4.3、实现完善个人信息

mock接口：

- 完善个人信息
  - <https://mock.boxuegu.com/project/164/interface/api/28553>

Mock地址： <https://mock.boxuegu.com/mock/164/user/loginRegInfo>

请求参数

Headers :				
参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body:

名称	类型	是否必须	默认值	备注	其他信息
gender	string	必须		性别 man woman	
nickname	string	必须		昵称	
birthday	string	必须		生日 年月日	
city	string	必须		城市	
header	string	非必须		用户头像	

- 上传头像
  - <https://mock.boxuegu.com/project/164/interface/api/39725>

Mock地址： <https://mock.boxuegu.com/mock/164/user/loginRegInfo/head>

**请求参数**

Headers :

参数名称	参数值	是否必须	示例	备注
Content-Type	multipart/form-data	是		

Body:

参数名称	参数类型	是否必须	示例	备注
headPhoto	文件	是		头像图片

**返回数据**

名称	类型	是否必须	默认值	备注	其他信息
暂无数据					

### 4.3.1、UserInfoMapper

```

1 package com.tanhua.sso.mapper;
2
3 import com.baomidou.mybatisplus.core.mapper.BaseMapper;
4 import com.tanhua.sso.pojo.UserInfo;
5
6 public interface UserInfoMapper extends BaseMapper<UserInfo> {
7 }
8

```

### 4.3.2、UserInfoService

```

1 package com.tanhua.sso.service;
2
3 import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
4 import com.tanhua.sso.enums.SexEnum;
5 import com.tanhua.sso.mapper.UserInfoMapper;
6 import com.tanhua.sso.pojo.User;
7 import com.tanhua.sso.pojo.UserInfo;
8 import com.tanhua.sso.vo.PicUploadResult;
9 import org.apache.commons.lang3.StringUtils;
10 import org.slf4j.Logger;
11 import org.slf4j.LoggerFactory;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.stereotype.Service;
14 import org.springframework.web.multipart.MultipartFile;
15
16 import java.util.Map;
17
18 @Service
19 public class UserInfoService {
20
21     private static final Logger LOGGER =
22         LoggerFactory.getLogger(UserInfoService.class);
23
24     @Autowired
25     private UserInfoMapper userInfoMapper;

```

```
25  
26     @Autowired  
27     private UserService userService;  
28  
29     @Autowired  
30     private FaceEngineService faceEngineService;  
31  
32     @Autowired  
33     private PicUploadService picuploadService;  
34  
35     /**  
36      * 完善个人信息  
37      *  
38      * @return  
39      */  
40     public Boolean saveUserInfo(Map<String, String> param, String token) {  
41         User user = this.userService.queryUserByToken(token);  
42         if (user == null) {  
43             return false;  
44         }  
45  
46         UserInfo userInfo = new UserInfo();  
47         userInfo.setUserId(user.getId());  
48         userInfo.setSex(StringUtils.equals(param.get("gender"), "man") ?  
SexEnum.MAN : SexEnum.WOMAN);  
49         userInfo.setNickName(param.get("nickname"));  
50         userInfo.setBirthday(param.get("birthday"));  
51         userInfo.setCity(param.get("city"));  
52  
53         // 保存UserInfo数据到数据库  
54         this.userInfoMapper.insert(userInfo);  
55  
56         return true;  
57     }  
58  
59  
60     public Boolean saveLogo(MultipartFile file, String token) {  
61         User user = this.userService.queryUserByToken(token);  
62         if (user == null) {  
63             return false;  
64         }  
65  
66         try {  
67             //校验头像是否为人像  
68             boolean isPortrait =  
this.faceEngineService.checkIsPortrait(file.getBytes());  
69             if (!isPortrait) {  
70                 return false;  
71             }  
72         } catch (Exception e) {  
73             LOGGER.error("检测人像图片出错!", e);  
74             return false;  
75         }  
76         // 图片上传到阿里云OSS  
77         PicUploadResult uploadResult = this.picuploadService.upload(file);  
78  
79         UserInfo userInfo = new UserInfo();  
80         userInfo.setLogo(uploadResult.getName());
```

```

81
82     QueryWrapper queryWrapper = new QueryWrapper();
83     queryWrapper.eq("user_id", user.getId());
84
85     this.userInfoMapper.update(userInfo, queryWrapper);
86
87     return true;
88 }
89
90

```

### 4.3.3、UserInfoController

```

1 package com.tanhua.sso.controller;
2
3 import com.tanhua.sso.service.UserInfoService;
4 import com.tanhua.sso.vo.ErrorResult;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.messaging.handler.annotation.Header;
9 import org.springframework.web.bind.annotation.*;
10 import org.springframework.web.multipart.MultipartFile;
11
12 import javax.servlet.http.HttpServletRequest;
13 import java.util.Map;
14
15 @RestController
16 @RequestMapping("user")
17 public class UserInfoController {
18
19     @Autowired
20     private UserInfoService userInfoService;
21
22     /**
23      * 完善个人信息
24      *
25      * @param param
26      * @param token
27      * @return
28      */
29     @RequestMapping("loginReginfo")
30     @PostMapping
31     public ResponseEntity<Object> saveUserInfo(@RequestBody Map<String,
32     String> param, @RequestHeader("Authorization") String token) {
33         try {
34             Boolean saveUserInfo = this.userInfoService.saveUserInfo(param,
35             token);
36             if (saveUserInfo) {
37                 return ResponseEntity.ok(null);
38             }
39         } catch (Exception e) {
40             e.printStackTrace();
41         }
42
43         ErrorResult errorResult =
44         ErrorResult.builder().errCode("000000").errMessage("发生错误").build();
45
46     }
47
48 }

```

```
42     return
43     ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorResult);
44 }
45 /**
46 * 上传头像
47 *
48 * @param file
49 * @param token
50 * @return
51 */
52 @RequestMapping("loginReginfo/head")
53 @PostMapping
54 public ResponseEntity<Object> saveLogo(@RequestParam("headPhoto")
55 MultipartFile file, @RequestHeader("Authorization") String token) {
56     try {
57         Boolean bool = this.userInfoService.saveLogo(file, token);
58         if(bool){
59             return ResponseEntity.ok(null);
60         }
61     } catch (Exception e) {
62         e.printStackTrace();
63     }
64     ErrorResult errorResult =
65     ErrorResult.builder().errCode("000000").errMessage("图片非人像，请重新上
66     传!").build();
67     return
68     ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorResult);
69 }
```

#### 4.4.4、测试



输入6位验证码

已发送到: 1750000000

5      8      1      2      3

---

---

---

---

---

重新获取 (50s)

free for personal use

11:47

填写资料  
提升我的魅力



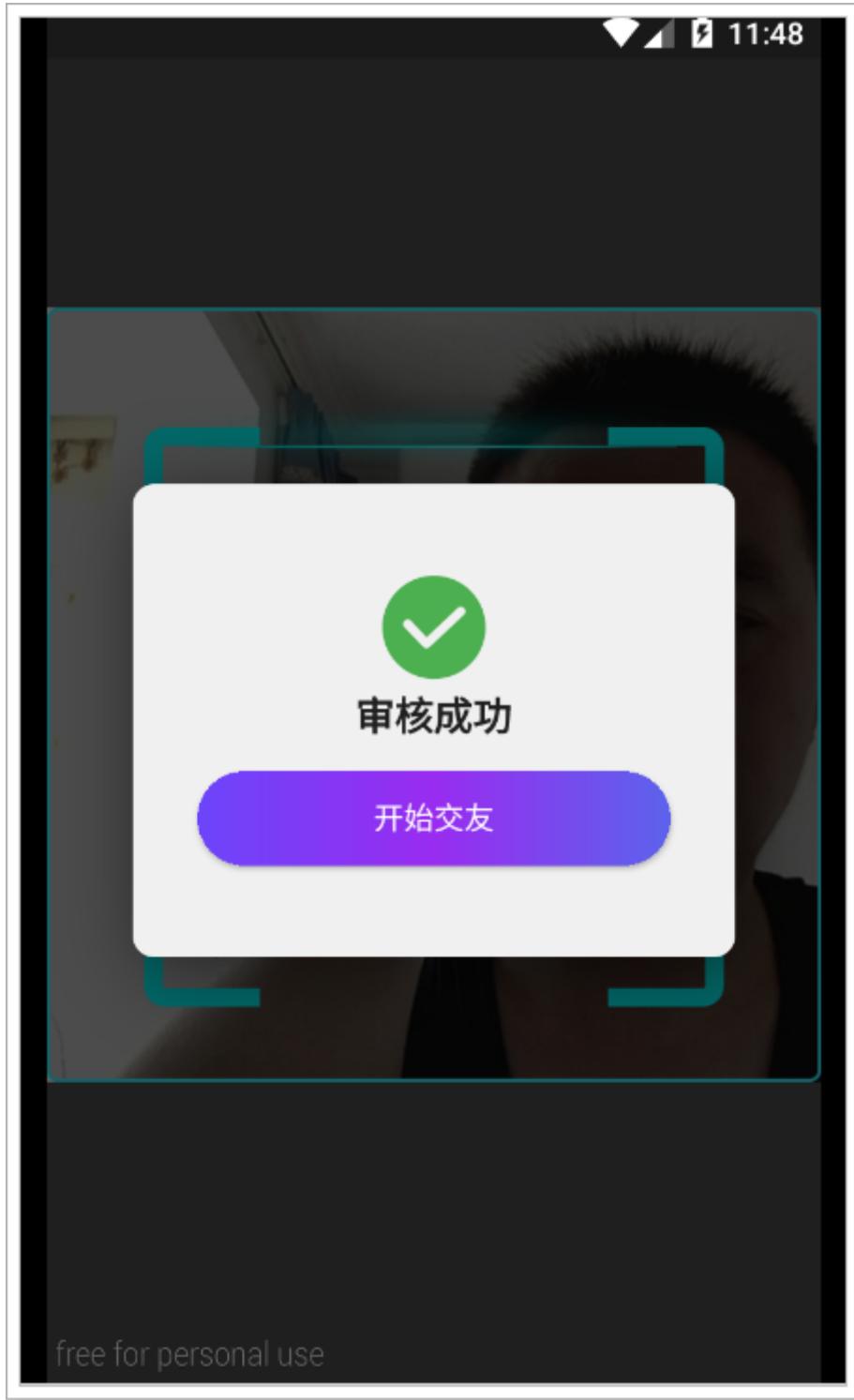
波仔

2018-07-30

北京市-北京城区-东城区

设置头像

free for personal use



## 5、检查登录状态

为其他系统提供根据token来查询用户信息的接口。

### 5.1、UserService

```
1  public User queryUserByToken(String token) {  
2      try {  
3          String redisTokenKey = "TOKEN_" + token;  
4          String cacheData =  
5          this.redisTemplate.opsForValue().get(redisTokenKey);  
6          if (StringUtils.isEmpty(cacheData)) {  
7              return null;  
8          }  
9      } catch (Exception e) {  
10          log.error("根据token查询用户失败,token:{}", token);  
11          return null;  
12      }  
13  }
```

```
8 // 刷新时间
9     this.redisTemplate.expire(redisTokenKey, 1, TimeUnit.HOURS);
10    return MAPPER.readValue(cacheData, User.class);
11 } catch (Exception e) {
12     e.printStackTrace();
13 }
14 return null;
15 }
```

## 5.2、UserController

```
1 /**
2  * 根据token查询用户数据
3  *
4  * @param token
5  * @return
6  */
7 @GetMapping("{token}")
8 public User queryUserByToken(@PathVariable("token") String token) {
9     return this.userService.queryUserByToken(token);
10}
```

## 5.3、测试

The screenshot shows a browser-based REST client interface. At the top, there is a URL input field containing `http://127.0.0.1:18080/user/eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUiOiIxNTgwMDgwNzk0NCIsImlkjoxfQ.X11vF45M6Ec_mlgcKcCWr0ekySVRd80LT7qd24t5hPo`. Below the URL are several radio buttons for selecting HTTP methods: GET (selected), POST, PUT, PATCH, DELETE, HEAD, OPTIONS, and Other. Underneath these buttons are three tabs: Raw, Form, and Headers. The Headers tab is currently active, showing the following header information:

Status	200	Loading time: 24 ms
Request headers	User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36 Content-Type: text/plain; charset=utf-8 Accept: */*	
Response headers	Content-Disposition: inline;filename=f.txt Content-Type: text/plain; charset=UTF-8 Content-Length: 31 Date: Thu, 25 Jul 2019 08:34:16 GMT	

At the bottom of the Headers section, there are three tabs: Raw, Parsed, and Response. The Response tab is selected, displaying the JSON response:

```
{"id":1,"mobile":"156...~244"}
```

Below the JSON response, there is a note: "Code highlighting thanks to [Code Mirror](#)".