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%。

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

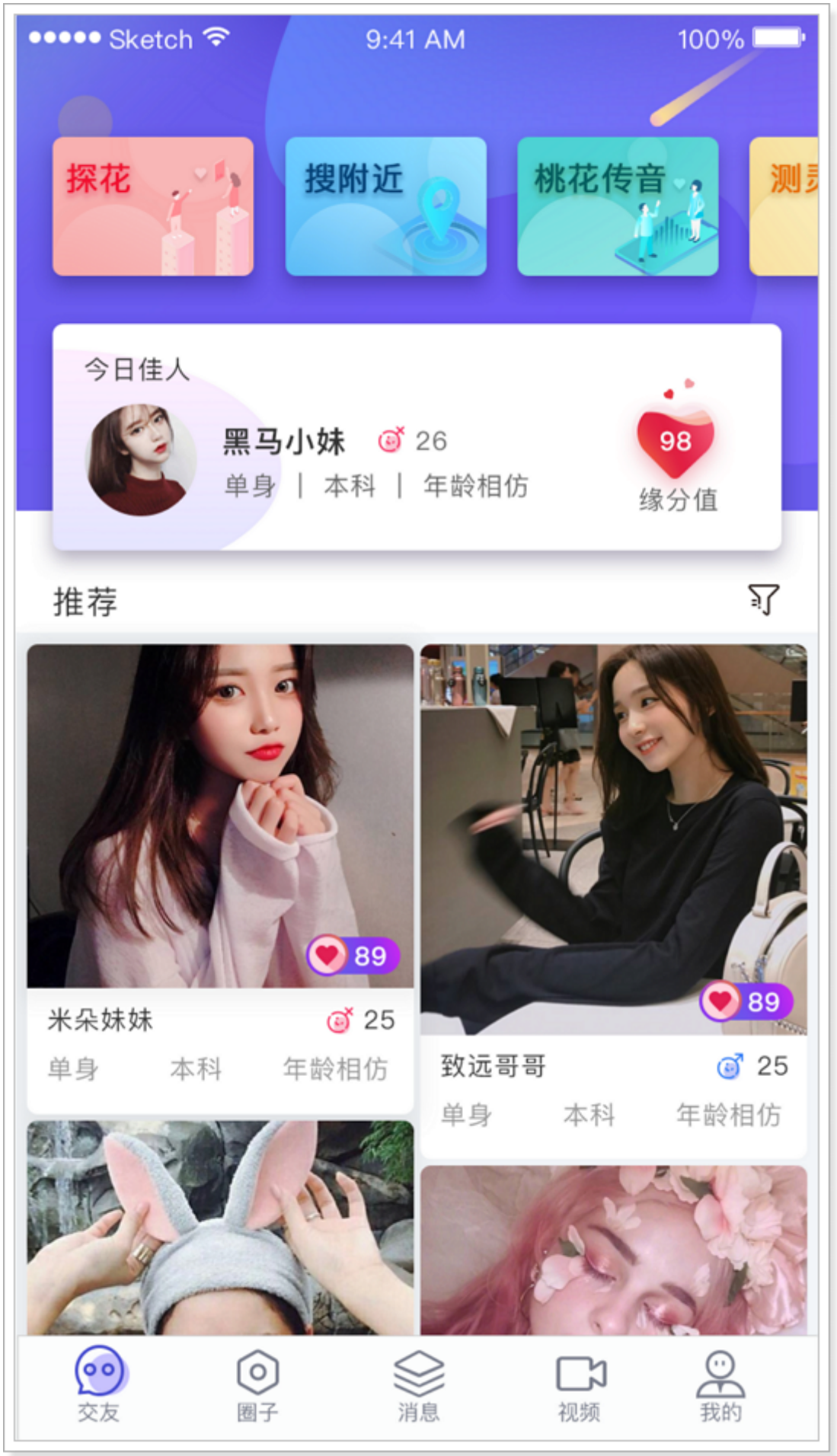
1.5.2、竞品分析

- **花田：**更偏向打造兴趣匹配，并配合线下活动俩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
今日佳人
黑马小妹
26
单身 | 本科 | 年龄相仿
98
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


Sketch
9:41 AM
100%
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

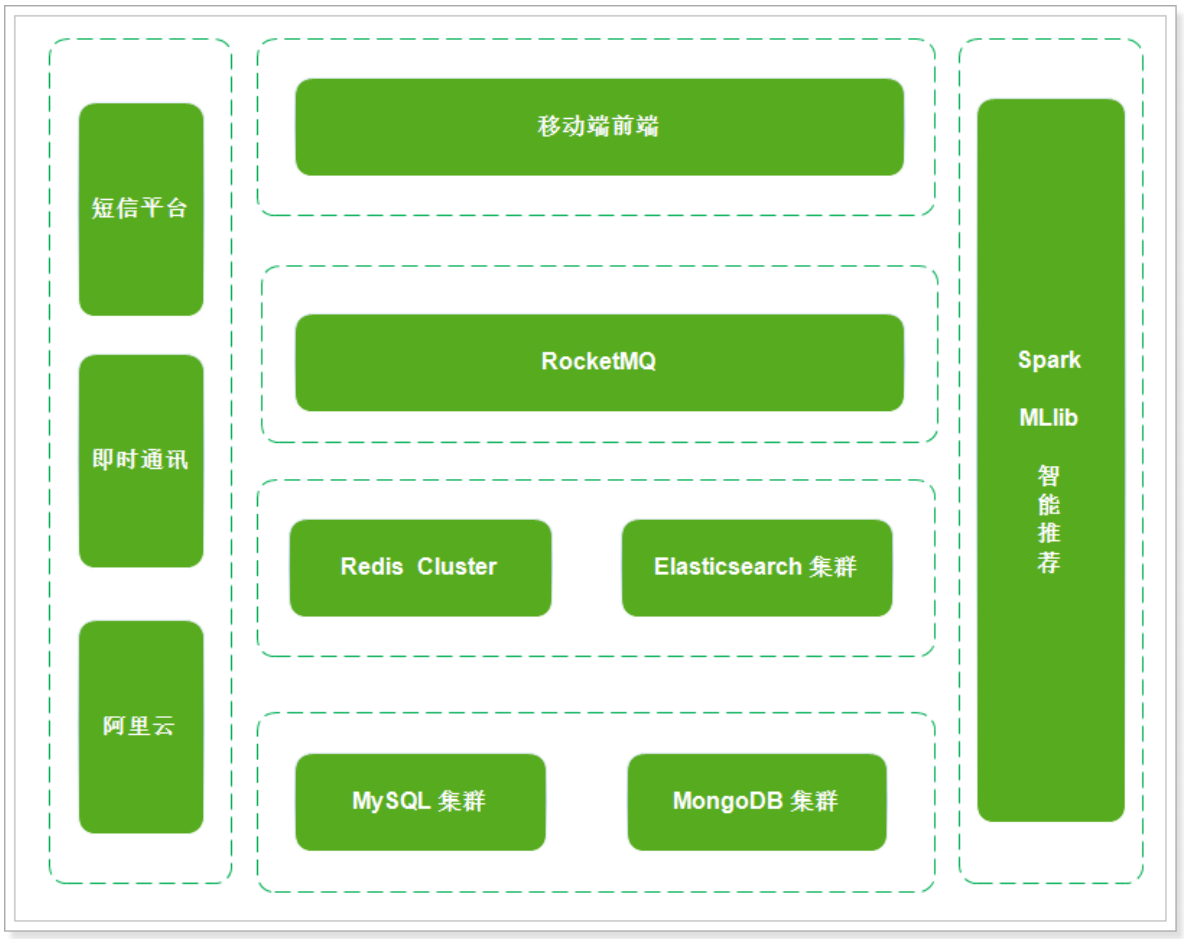
前端：

- 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

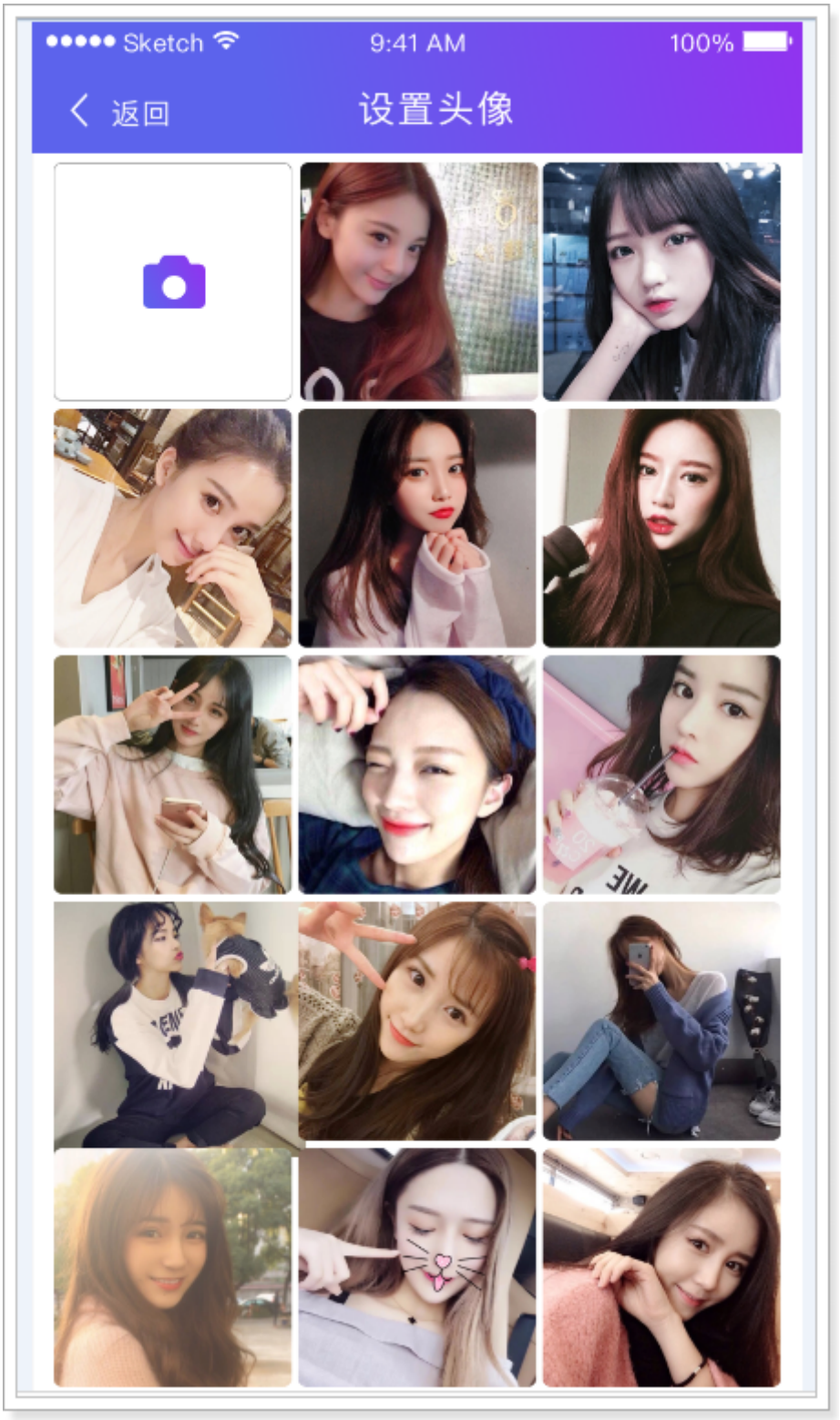
业务说明：

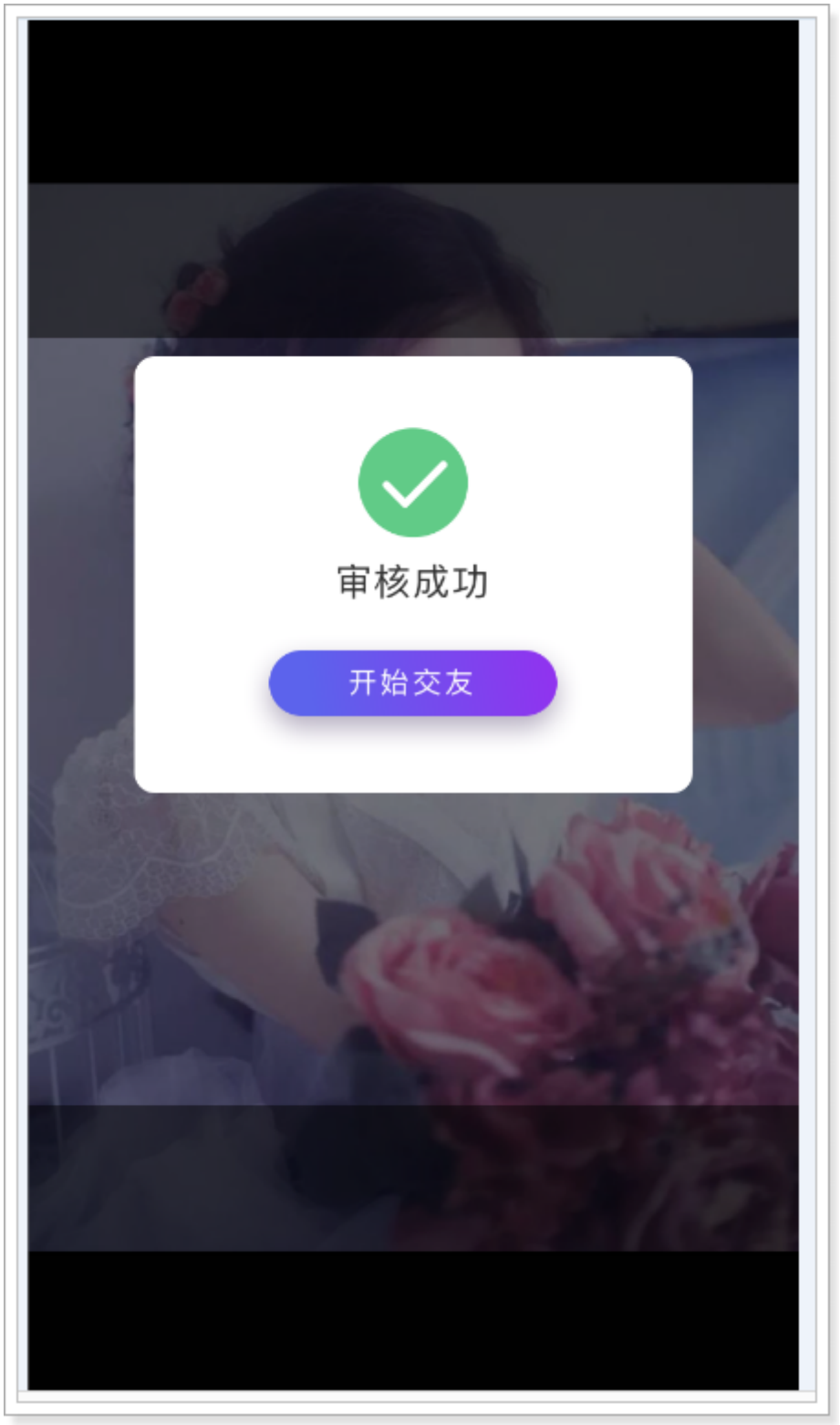
用户通过手机验证码进行登录，如果是第一次登录则需要完善个人信息，在上传图片时，需要对上传的图片做人像的校验，防止用户上传非人像的图片作为头像。流程完成后，则登录成功。

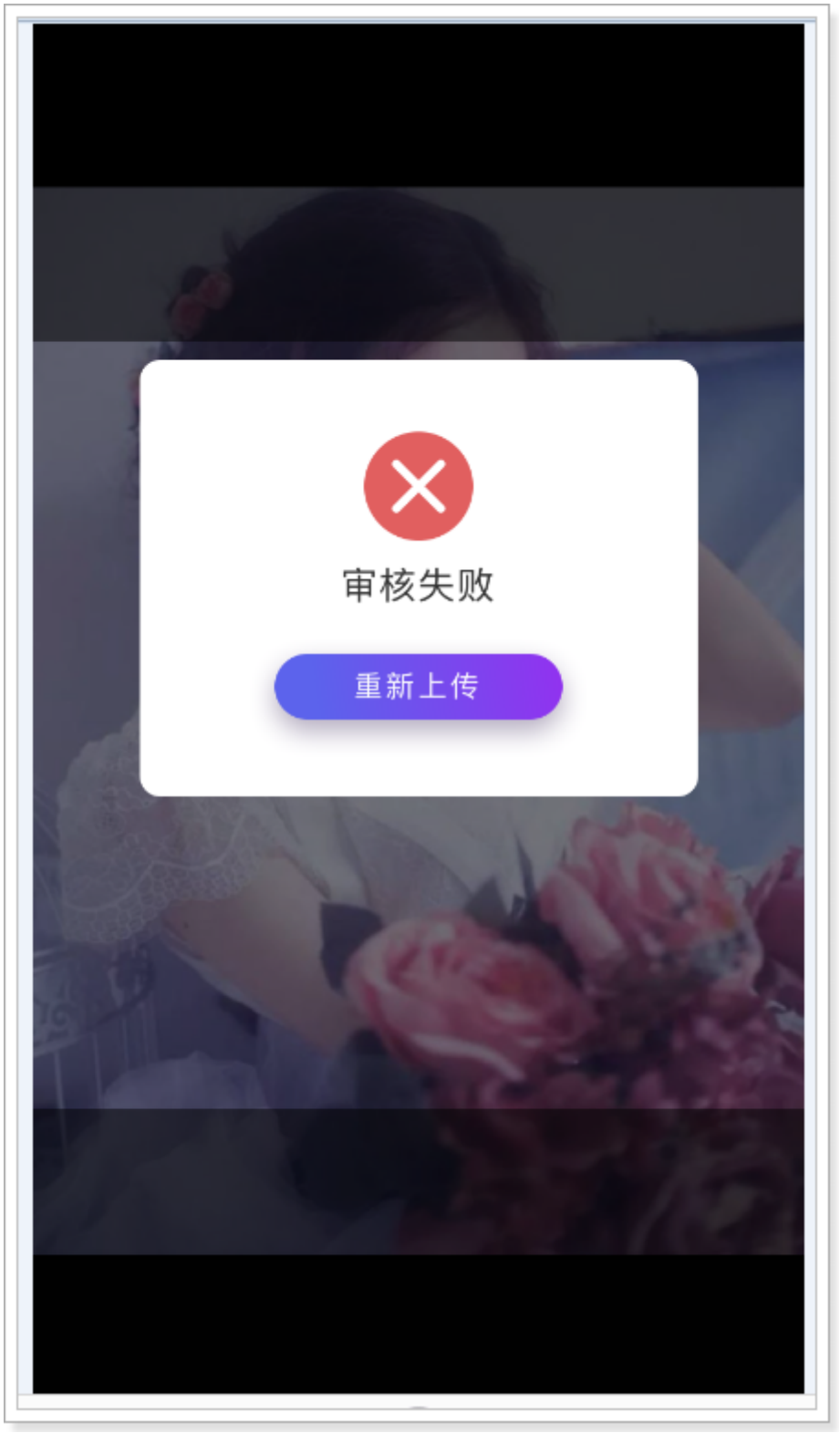

Sketch
9:41 AM
100%
手机号登录注册
+86
输入手机号码
获取验证码


Sketch
9:41 AM
100%
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）


Sketch
9:41 AM
100%
填写资料
提升我的魅力
昵称设置
你的生日
现居城市
设置头像


Sketch
9:41 AM
100%
返回
设置头像


审核成功
开始交友



2.3、交友

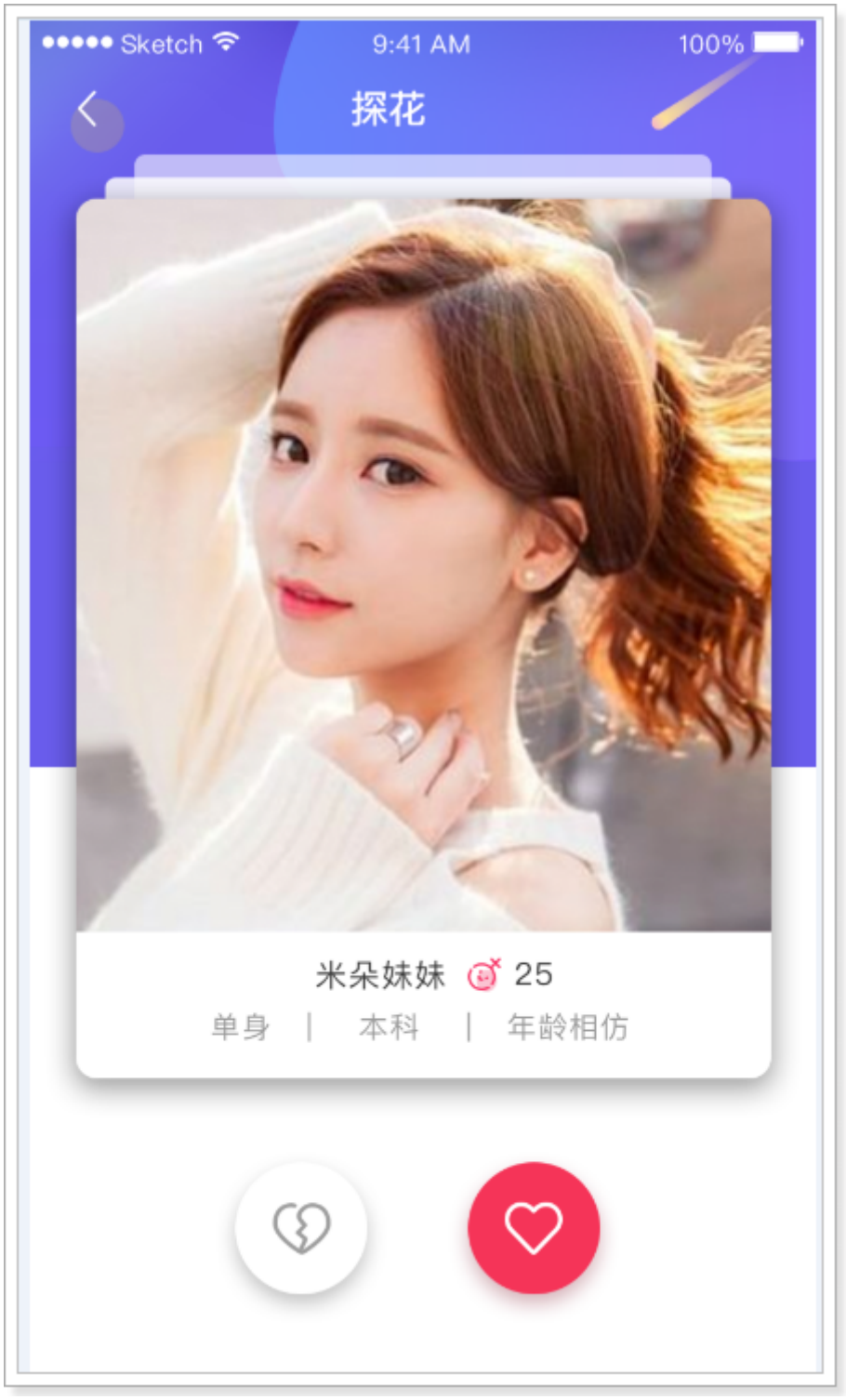
交友是探花项目的核心功能之一，用户可以查看好友，添加好友，搜索好友等操作。

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



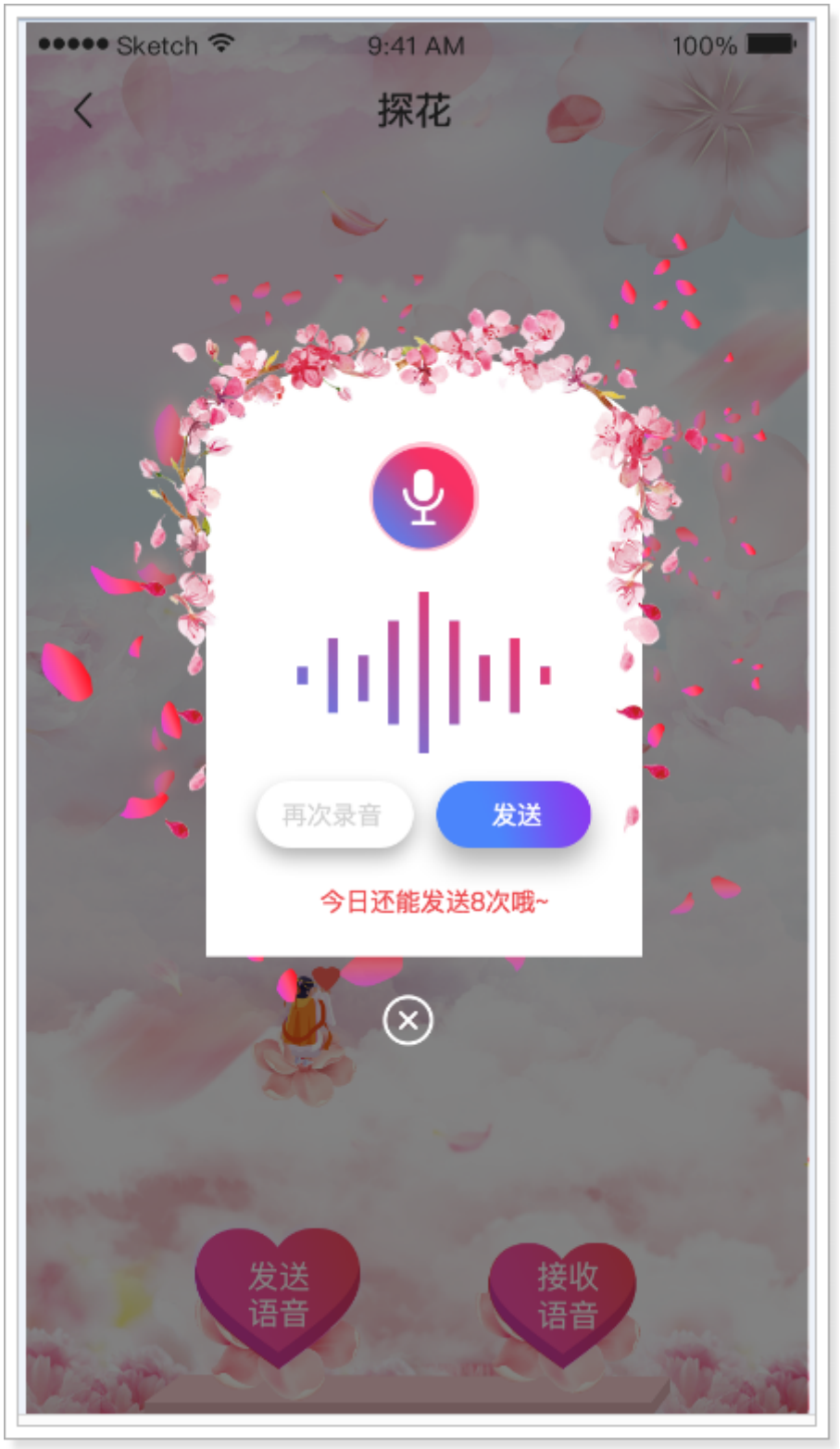
> 说明：左划喜欢，右划不喜欢，每天限量不超过100个，开通会员可增加限额。双方互相喜欢则配对成功。
>
> 实现：数据来源推荐系统计算后的结果。

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

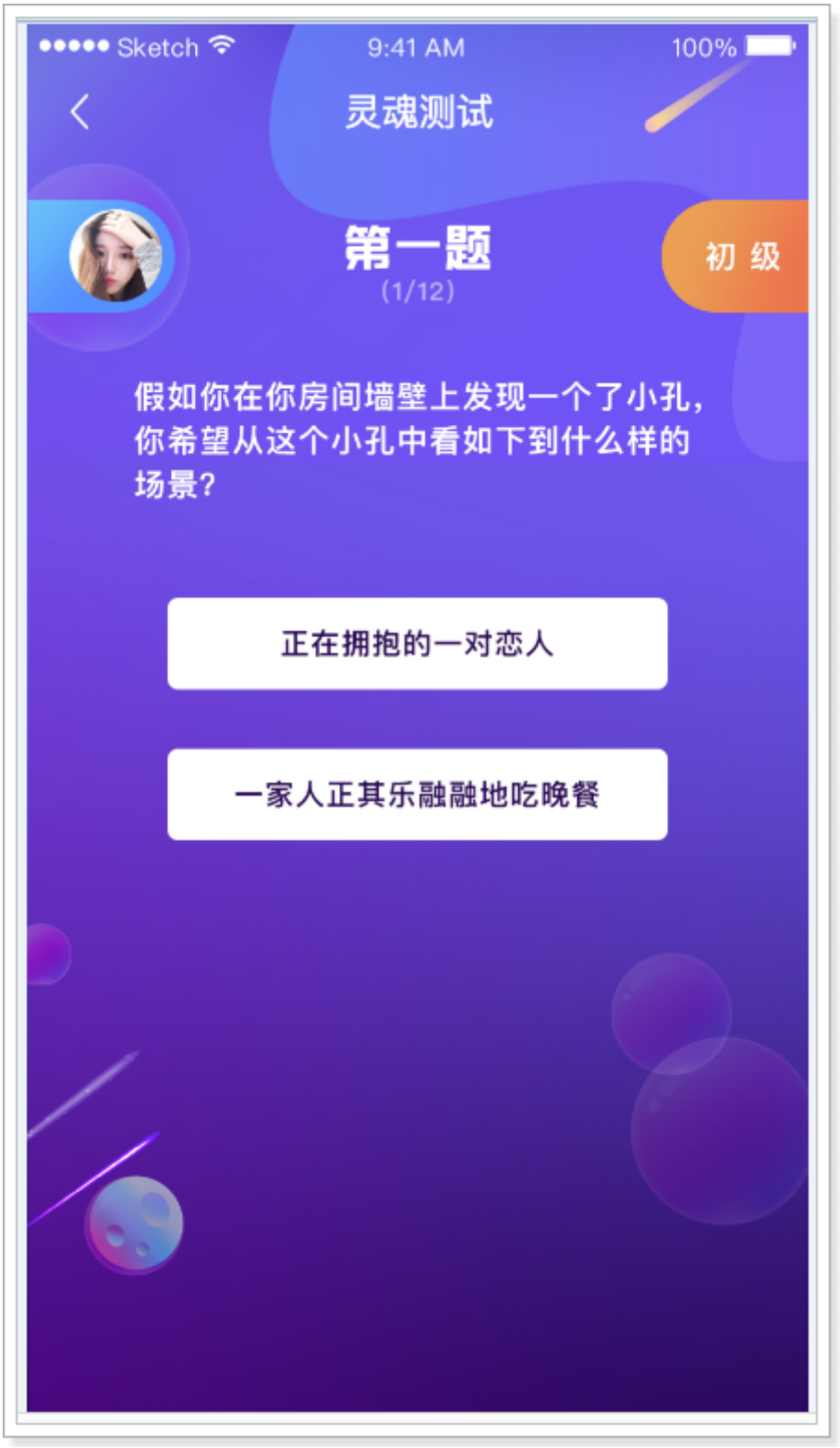
功能类似QQ中的漂流瓶，用户可以发送和接收语音消息，陌生人就会接收到消息。



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


Sketch
9:41 AM
100%
灵魂测试
初级灵魂题
开始测试


Sketch
9:41 AM
100%
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

Sketch 9:41 AM 100%

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

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
互相关注
345
喜欢
113
粉丝
我的动态
谁看过我
钱包
通用设置
客服在线

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。

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <packaging>pom</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.0.RELEASE</version>
    </parent>

    <groupId>cn.itcast.tanhua</groupId>
    <artifactId>itcast-tanhua</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!-- 集中定义依赖版本号 -->
    <properties>
        <mysql.version>5.1.47</mysql.version>
        <jackson.version>2.9.9</jackson.version>
        <druid.version>1.0.9</druid.version>
        <servlet-api.version>2.5</servlet-api.version>
        <jsp-api.version>2.0</jsp-api.version>
        <joda-time.version>2.5</joda-time.version>
        <commons-lang3.version>3.3.2</commons-lang3.version>
        <commons-io.version>1.3.2</commons-io.version>
        <mybatis.version>3.2.8</mybatis.version>
        <mybatis.mybatis-plus>3.1.1</mybatis.mybatis-plus>
        <lombok.version>1.18.4</lombok.version>
    </properties>

    <!--通用依赖-->
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
            <!-- mybatis-plus插件依赖 -->
            <dependency>
                <groupId>com.baomidou</groupId>
                <artifactId>mybatis-plus</artifactId>
                <version>${mybatis.mybatis-plus}</version>
            </dependency>
            <!-- MySql -->
            <dependency>
                <groupId>mysql</groupId>
```

```
                <artifactId>mysql-connector-java</artifactId>
                <version>${mysql.version}</version>
            </dependency>

            <dependency>
                <groupId>org.mongodb</groupId>
                <artifactId>mongodb-driver-sync</artifactId>
                <version>3.9.1</version>
            </dependency>
            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <optional>true</optional>
                <version>${lombok.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.commons</groupId>
                <artifactId>commons-lang3</artifactId>
                <version>${commons-lang3.version}</version>
            </dependency>
            <!--RocketMQ相关依赖-->
            <dependency>
                <groupId>org.apache.rocketmq</groupId>
                <artifactId>rocketmq-spring-boot-starter</artifactId>
                <version>2.0.0</version>
            </dependency>
            <dependency>
                <groupId>org.apache.rocketmq</groupId>
                <artifactId>rocketmq-client</artifactId>
                <version>4.3.2</version>
            </dependency>
            <dependency>
                <groupId>org.apache.commons</groupId>
                <artifactId>commons-lang3</artifactId>
                <version>3.7</version>
            </dependency>
            <!-- Jackson Json处理工具包 -->
            <dependency>
                <groupId>com.fasterxml.jackson.core</groupId>
                <artifactId>jackson-databind</artifactId>
                <version>${jackson.version}</version>
            </dependency>
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>${druid.version}</version>
            </dependency>
            <dependency>
                <groupId>commons-codec</groupId>
                <artifactId>commons-codec</artifactId>
                <version>1.11</version>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <!-- java编译插件 -->
```

```
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.2</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
        </plugins>
    </build>

</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。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>itcast-tanhua</artifactId>
        <groupId>cn.itcast.tanhua</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>itcast-tanhua-sso</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus</artifactId>
        </dependency>
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>${mybatis.mybatis-plus}</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
```

```
            <artifactId>lombok</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
        </dependency>


        <!--RocketMQ相关-->
        <dependency>
            <groupId>org.apache.rocketmq</groupId>
            <artifactId>rocketmq-spring-boot-starter</artifactId>
            <version>2.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.rocketmq</groupId>
            <artifactId>rocketmq-client</artifactId>
            <version>4.3.2</version>
        </dependency>
    </dependencies>

</project>
```

3.1.3、部署基础服务

使用docker进行部署，采用Ubuntu操作系统。

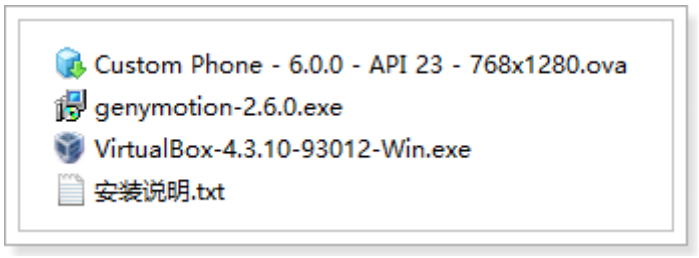
```
#部署Redis集群，该集群有3个节点
docker create --name redis-node01 --net host -v redis-node01:/data
redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-01.conf
--port 6379

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

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

#启动容器
docker start redis-node01 redis-node02 redis-node03

#进入redis-node01容器进行操作
docker exec -it redis-node01 /bin/bash
```

```
#组件集群
redis-cli --cluster create 192.168.31.81:6379 192.168.31.81:6380
192.168.31.81:6381 --cluster-replicas 0

#查询集群信息

127.0.0.1:6379> CLUSTER NODES
4f4fddc825e2387783fff9c972409b264e4df5d5 192.168.31.81:6381@16381 master -
0 1563956537241 3 connected 10923-16383
0616e00533a16e931f8dfb2e8844c35ca5721dc8 192.168.31.81:6380@16380 master -
0 1563956538243 2 connected 5461-10922
498b986e07731cead17ad1c62aa95dba6513c7b0 192.168.31.81:6379@16379
myself,master - 0 1563956537000 1 connected 0-5460
```

```
#部署RocketMQ
#拉取镜像
docker pull foxiswho/rocketmq:server-4.3.2
docker pull foxiswho/rocketmq:broker-4.3.2

#创建nameserver容器
docker create -p 9876:9876 --name rmqserver \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt" \
-v rmqserver-logs:/opt/logs \
-v rmqserver-store:/opt/store \
foxiswho/rocketmq:server-4.3.2

#创建broker.conf文件
vim /itcast/rmq/rmqbroker/conf/broker.conf
brokerIP1=192.168.31.81
namesrvAddr=192.168.31.81:9876
brokerName=broker_tanhua

#创建broker容器
docker create -p 10911:10911 -p 10909:10909 --name rmqbroker \
-e "JAVA_OPTS=-Duser.home=/opt" \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-v /itcast/rmq/rmqbroker/conf/broker.conf:/etc/rocketmq/broker.conf \
-v rmqbroker-logs:/opt/logs \
-v rmqbroker-store:/opt/store \
foxiswho/rocketmq:broker-4.3.2

#启动容器
docker start rmqserver rmqbroker

#停止删除容器
docker stop rmqbroker rmqserver
docker rm rmqbroker rmqserver

#部署RocketMQ的管理工具
docker pull styletang/rocketmq-console-ng:1.0.0

#创建并启动容器
docker run -e "JAVA_OPTS=-Drocketmq.namesrv.addr=192.168.31.81:9876 -
Dcom.rocketmq.sendMessageWithVIPChannel=false" -p 8082:8080 -t
styletang/rocketmq-console-ng:1.0.0
```

3.1.4、搭建客户端

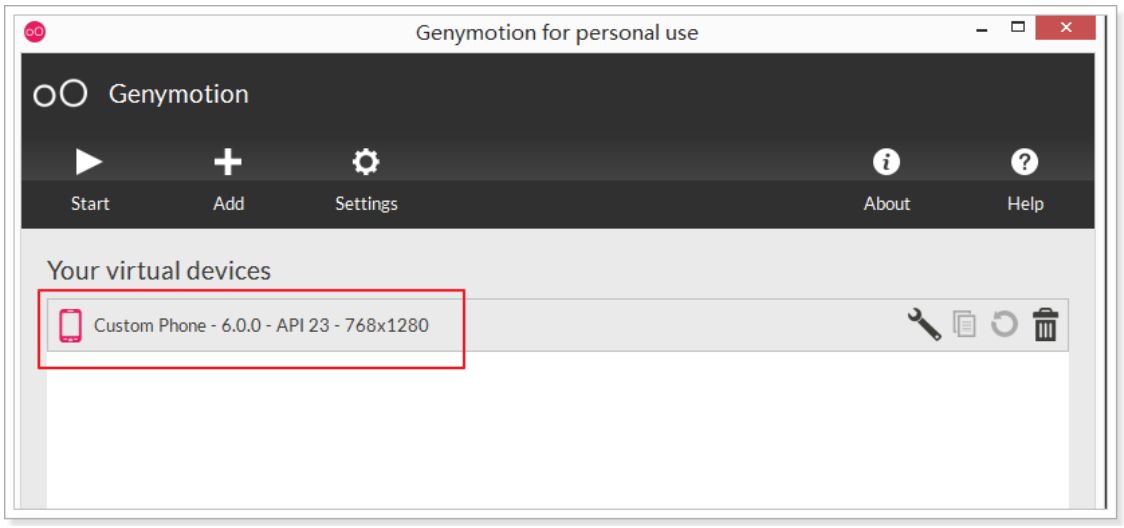
客户端由前端团队进行开发，前端提供apk进行对接，所以，需要我们安装安卓的模拟器进行测试。



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


Genymotion for personal use
Genymotion
Start
Add
Settings
About
Help
Your virtual devices
Custom Phone - 6.0.0 - API 23 - 768x1280

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：

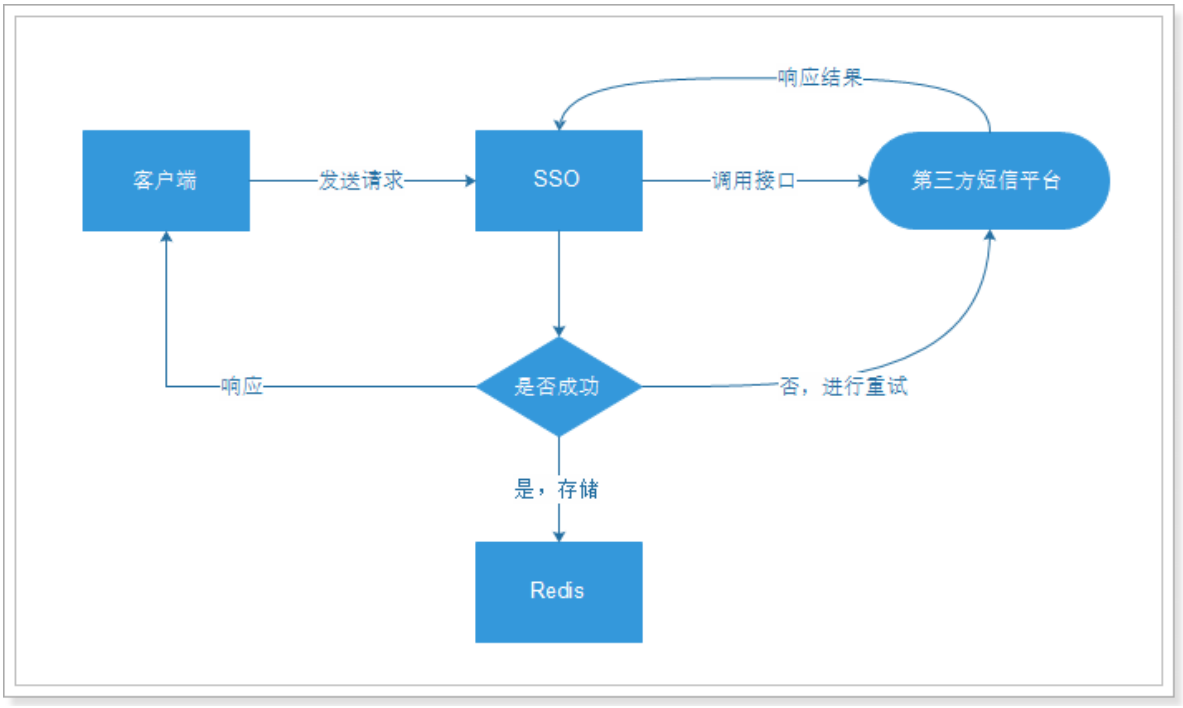
3.2、实现分析

需要提供4个接口服务：

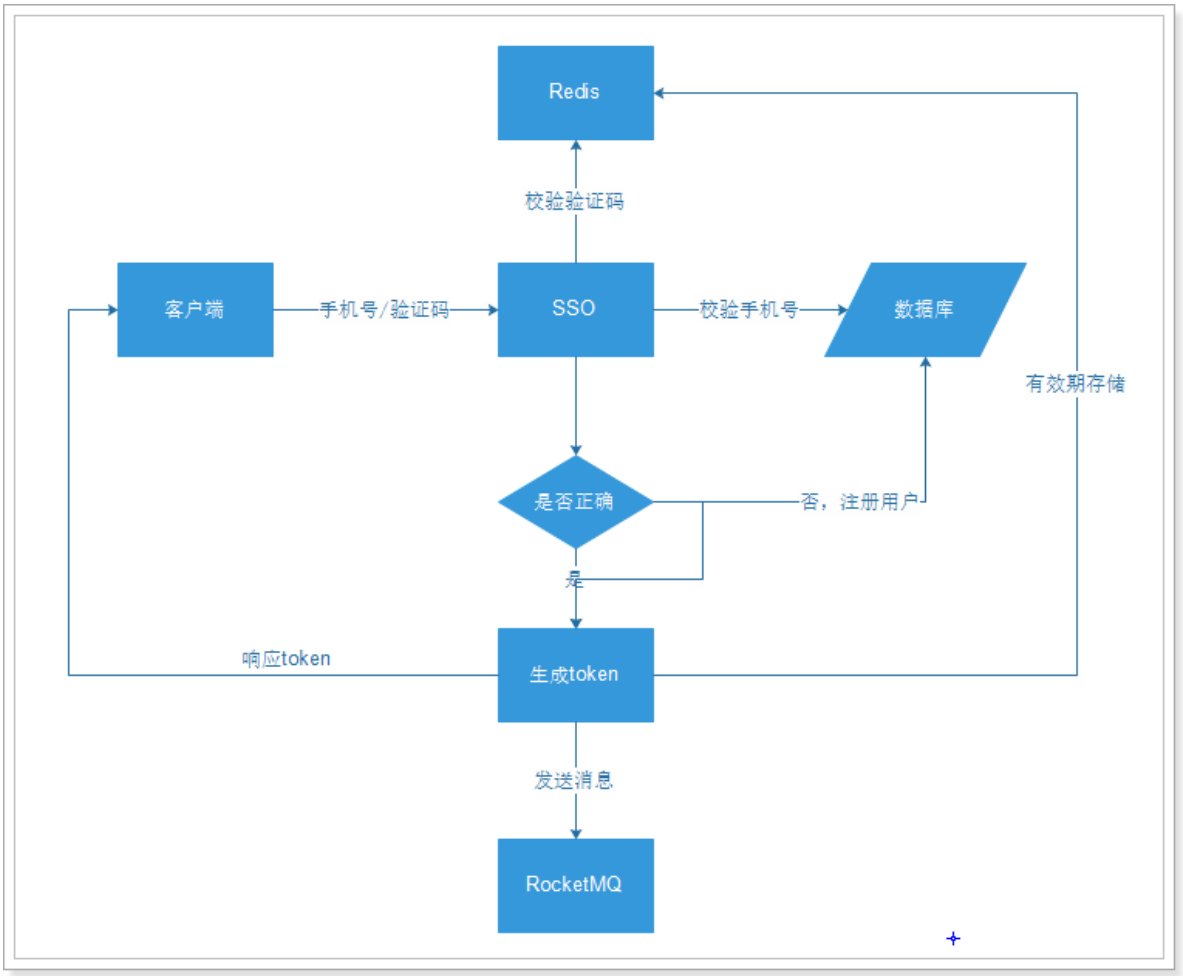
- 发送手机验证码
  - 需要对接第三方短信平台进行发送验证码
  - 选用云之讯平台进行发送 https://www.ucpaas.com/ （选用其他接口也可以）
- 校验用户登录
  - 后台需要验证手机号与验证码是否正确
  - 校验成功之后，需要按照JWT规范进行返回响应
- 首次登录完善个人信息

- 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：

```sql
CREATE TABLE `tb_user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `mobile` varchar(11) DEFAULT NULL COMMENT '手机号',
  `password` varchar(32) DEFAULT NULL COMMENT '密码，需要加密',
  `created` datetime DEFAULT NULL,
  `updated` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `mobile` (`mobile`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='用户表';

CREATE TABLE `tb_user_info` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `user_id` bigint(20) NOT NULL COMMENT '用户id',
  `nick_name` varchar(50) DEFAULT NULL COMMENT '昵称',
  `logo` varchar(100) DEFAULT NULL COMMENT '用户头像',
  `tags` varchar(50) DEFAULT NULL COMMENT '用户标签：多个用逗号分隔',
  `sex` tinyint(1) DEFAULT '3' COMMENT '性别，1-男，2-女，3-未知',
  `age` int(11) DEFAULT NULL COMMENT '用户年龄',
  `edu` varchar(20) DEFAULT NULL COMMENT '学历',
  `city` varchar(20) DEFAULT NULL COMMENT '居住城市',
  `birthday` varchar(20) DEFAULT NULL COMMENT '生日',
  `cover_pic` varchar(50) DEFAULT NULL COMMENT '封面图片',
  `industry` varchar(20) DEFAULT NULL COMMENT '行业',
  `income` varchar(20) DEFAULT NULL COMMENT '收入',
  `marriage` varchar(20) DEFAULT NULL COMMENT '婚姻状态',
  `created` datetime DEFAULT NULL,
  `updated` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户信息表';

```

3.4、编写配置

application.properties：

```properties
spring.application.name = itcast-tanhua-sso

server.port = 18080

spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/tanhua?useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=root


# 枚举包扫描
mybatis-plus.type-enums-package=com.tanhua.sso.enums
# 表名前缀
mybatis-plus.global-config.db-config.table-prefix=tb_
# id策略为自增长
mybatis-plus.global-config.db-config.id-type=auto


```

```
# Redis相关配置
spring.redis.jedis.pool.max-wait = 5000ms
spring.redis.jedis.pool.max-Idle = 100
spring.redis.jedis.pool.min-Idle = 10
spring.redis.timeout = 10s
spring.redis.cluster.nodes =
192.168.31.81:6379,192.168.31.81:6380,192.168.31.81:6381
spring.redis.cluster.max-redirects=5

# RocketMQ相关配置
spring.rocketmq.nameServer=192.168.31.81:9876
spring.rocketmq.producer.group=tanhua

#itcast_tanhua
jwt.secret=76bd425b6f29f7fcc2e0bfc286043df1

#虹软相关配置
arcsoft.appid=*****
arcsoft.sdkKey=****
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。

```
package com.tanhua.sso.enums;

import com.baomidou.mybatisplus.core.enums.IEnum;

public enum SexEnum implements IEnum<Integer> {

    MAN(1,"男"),
    WOMAN(2,"女"),
    UNKNOWN(3,"未知");

    private int value;
    private String desc;

    SexEnum(int value, String desc) {
        this.value = value;
        this.desc = desc;
    }

    @Override
    public Integer getValue() {
        return this.value;
    }

    @Override
    public String toString() {
        return this.desc;
    }
}

```

3.5.2、User、UserInfo

```java
package com.tanhua.sso.pojo;

import com.baomidou.mybatisplus.annotation.FieldFill;
import com.baomidou.mybatisplus.annotation.TableField;

import java.util.Date;


public abstract class BasePojo {

    @TableField(fill = FieldFill.INSERT) //自动填充
    private Date created;
    @TableField(fill = FieldFill.INSERT_UPDATE)
    private Date updated;
}

```

```java
package com.tanhua.sso.pojo;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class User extends BasePojo {

    private Long id;
    private String mobile; //手机号

    @JsonIgnore
    private String password; //密码，json序列化时忽略

}

```

```java
package com.tanhua.sso.pojo;

import com.tanhua.sso.enums.SexEnum;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserInfo extends BasePojo {

    private Long id;
    private Long userId; //用户id
```

```
    private String nickName; //昵称
    private String logo; //用户头像
    private String tags; //用户标签：多个用逗号分隔
    private SexEnum sex; //性别
    private Integer age; //年龄
    private String edu; //学历
    private String city; //城市
    private String birthday; //生日
    private String coverPic; // 封面图片
    private String industry; //行业
    private String income; //收入
    private String marriage; //婚姻状态

}

```

3.5.3、MyMetaObjectHandler

对自动填充字段的处理：

```
package com.tanhua.sso.handler;

import com.baomidou.mybatisplus.core.handlers.MetaObjectHandler;
import org.apache.ibatis.reflection.MetaObject;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class MyMetaObjectHandler implements MetaObjectHandler {

    @Override
    public void insertFill(MetaObject metaObject) {
        Object created = getFieldValByName("created", metaObject);
        if (null == created) {
            //字段为空，可以进行填充
            setFieldValByName("created", new Date(), metaObject);
        }

        Object updated = getFieldValByName("updated", metaObject);
        if (null == updated) {
            //字段为空，可以进行填充
            setFieldValByName("updated", new Date(), metaObject);
        }
    }

    @Override
    public void updateFill(MetaObject metaObject) {
        //更新数据时，直接更新字段
        setFieldValByName("updated", new Date(), metaObject);
    }
}

```

3.5.4、UserMapper

```java
package com.tanhua.sso.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.tanhua.sso.pojo.User;

public interface UserMapper extends BaseMapper<User> {

}
```

3.5.5、MyApplication

SpringBoot的启动类。

```java
package com.tanhua.sso;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@MapperScan("com.tanhua.sso.mapper") //设置mapper接口的扫描包
@SpringBootApplication
public class MyApplication {

    public static void main(String[] args) {
        SpringApplication.run(MyApplication.class, args);
    }
}

```

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/

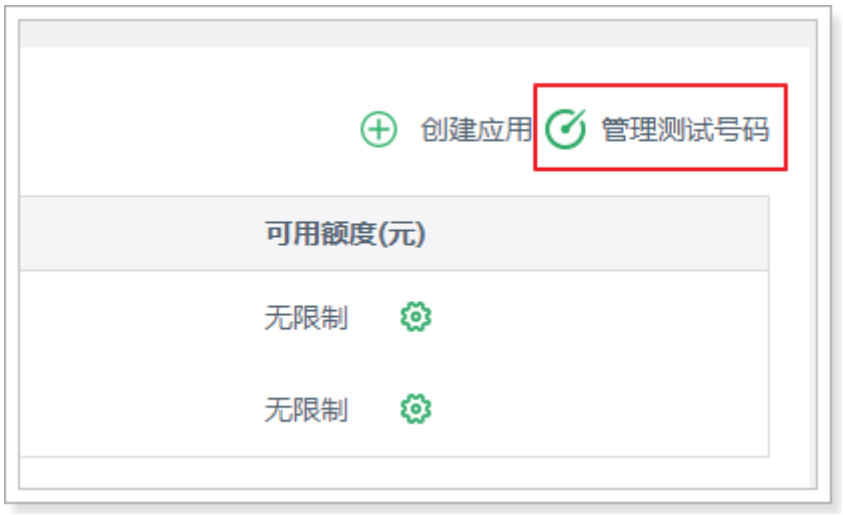
> 选择原因：注册赠送10元，无实名认证也可以发验证码短信。

3.6.1、创建应用

自行注册、登录后进行创建应用：

创建完成：

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）：



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	待审核	

共有1条，每页显示 15 条 1/1 跳转

> 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（位置）	39467b989d087c2d92c6132184a365d8
token	String	必填	用户密钥“Auth Token”，在开发者控制台获取（位置）	23f757bad208226ec301e117e40006ed
appid	String	必填	创建应用时系统分配的唯一标示（位置）	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（位置）	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

3.6.4、编写代码

配置RestTemplateConfig:

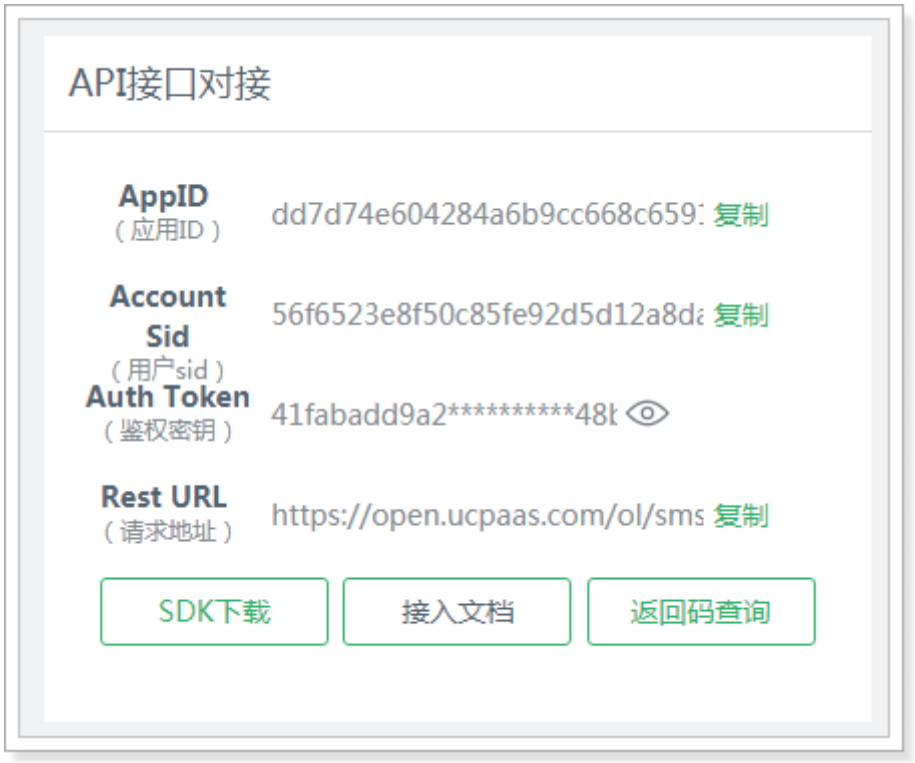
```
package com.tanhua.sso.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.client.ClientHttpRequestFactory;
import org.springframework.http.client.SimpleClientHttpRequestFactory;
import org.springframework.web.client.RestTemplate;
import org.springframework.http.converter.StringHttpMessageConverter;

import java.nio.charset.Charset;

@Configuration
public class RestTemplateConfig {

    @Bean
    public RestTemplate restTemplate(ClientHttpRequestFactory factory) {
        RestTemplate restTemplate = new RestTemplate(factory);
        // 支持中文编码
        restTemplate.getMessageConverters().set(1, new
StringHttpMessageConverter(Charset.forName("UTF-8")));
        return restTemplate;

    }

    @Bean
    public ClientHttpRequestFactory simpleClientHttpRequestFactory() {
        SimpleClientHttpRequestFactory factory = new
SimpleClientHttpRequestFactory();
        factory.setReadTimeout(5000);//单位为ms
        factory.setConnectTimeout(5000);//单位为ms
        return factory;
    }
}
```

编写SmsService：

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

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.commons.lang3.RandomUtils;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

```

```java
@Service
public class SmsService {

    @Autowired
    private RestTemplate restTemplate;

    private static final ObjectMapper MAPPER = new ObjectMapper();


    /**
     * 发送验证码短信
     *
     * @param mobile
     */
    public String sendSms(String mobile) {
        String url = "https://open.ucpaas.com/ol/sms/sendsms";
        Map<String, Object> params = new HashMap<>();
        params.put("sid", "********");
        params.put("token", "********");
        params.put("appid", "********");
        params.put("templateid", "******");
        params.put("mobile", mobile);
        // 生成4位数验证
        params.put("param", RandomUtils.nextInt(100000, 999999));
        ResponseEntity<String> responseEntity =
this.restTemplate.postForEntity(url, params, String.class);

        String body = responseEntity.getBody();

        try {
            JsonNode jsonNode = MAPPER.readTree(body);
            //000000 表示发送成功
            if (StringUtils.equals(jsonNode.get("code").textValue(),
"000000")) {
                return String.valueOf(params.get("param"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        return null;

    }

}

```

相关参数的信息查询：

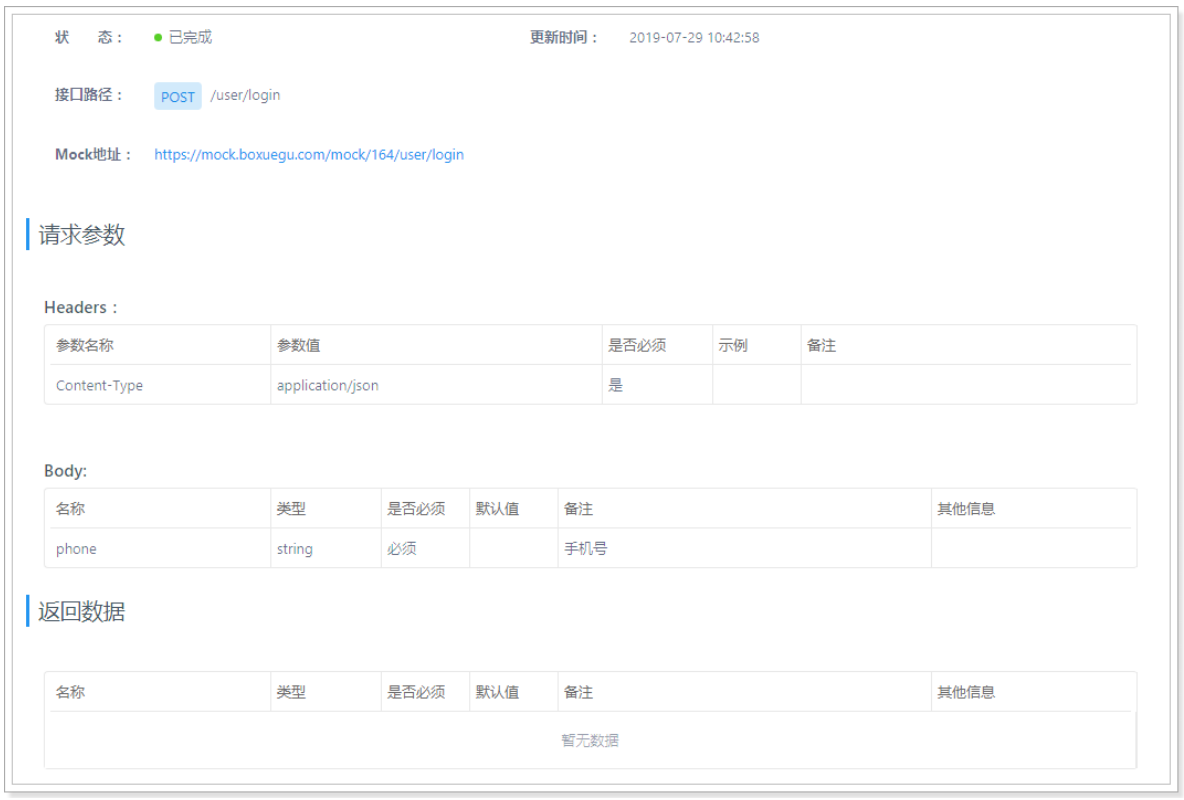


3.6.5、编写测试用例

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

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

@SpringBootTest
@RunWith(SpringJUnit4ClassRunner.class)
public class TestSmsService {

    @Autowired
    private SmsService smsService;

    @Test
    public void sendSms(){
        this.smsService.sendSms("158****7944");
    }
}

```

3.6.6、mock接口

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



3.6.6、编写接口服务

编写ErrorResult：

```
package com.tanhua.sso.vo;

import lombok.Builder;
import lombok.Data;

@Data
@Builder
public class ErrorResult {

    private String errCode;
    private String errMessage;
}

```

SmsController：

```
package com.tanhua.sso.controller;

import com.tanhua.sso.service.SmsService;
import com.tanhua.sso.vo.ErrorResult;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

```

```
import java.util.Map;

@RestController
@RequestMapping("user")
public class SmsController {

    private static final Logger LOGGER =
LoggerFactory.getLogger(SmsController.class);

    @Autowired
    private SmsService smsService;

    /**
     * 发送验证码
     *
     * @return
     */
    @PostMapping("login")
    public ResponseEntity<Object> sendCheckCode(@RequestBody Map<String,
Object> param) {
        ErrorResult.ErrorResultBuilder resultBuilder =
ErrorResult.builder().errCode("000000").errMessage("发送短信验证码失败");
        try {
            String phone = String.valueOf(param.get("phone"));
            Map<String, Object> sendCheckCode =
this.smsService.sendCheckCode(phone);
            int code = ((Integer) sendCheckCode.get("code")).intValue();
            if (code == 3) {
                return ResponseEntity.ok(null);
            }else if(code == 1){

 resultBuilder.errCode("000001").errMessage(sendCheckCode.get("msg").toStri
ng());
            }
        } catch (Exception e) {
            LOGGER.error("发送短信验证码失败", e);
        }

        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(resultBuilder.
build());
    }

}

```

SmsService：

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

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.commons.lang3.RandomUtils;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
```

```java
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import java.io.IOException;
import java.time.Duration;
import java.util.HashMap;
import java.util.Map;

@Service
public class SmsService {

    private static final Logger LOGGER =
LoggerFactory.getLogger(SmsService.class);

    @Autowired
    private RestTemplate restTemplate;

    private static final ObjectMapper MAPPER = new ObjectMapper();

    @Autowired
    private RedisTemplate<String, String> redisTemplate;

    /**
     * 发送验证码
     *
     * @param mobile
     * @return
     */
    public Map<String, Object> sendCheckCode(String mobile) {
        Map<String, Object> result = new HashMap<>(2);
        try {
            String redisKey = "CHECK_CODE_" + mobile;
            String value = this.redisTemplate.opsForValue().get(redisKey);
            if (StringUtils.isNotEmpty(value)) {
                result.put("code", 1);
                result.put("msg", "上一次发送的验证码还未失效");
                return result;
            }
            String code = this.sendSms(mobile);
            if (null == code) {
                result.put("code", 2);
                result.put("msg", "发送短信验证码失败");
                return result;
            }

            //发送验证码成功
            result.put("code", 3);
            result.put("msg", "ok");

            //将验证码存储到Redis,2分钟后失效
            this.redisTemplate.opsForValue().set(redisKey, code,
Duration.ofMinutes(2));

            return result;
        } 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 JWTS 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。

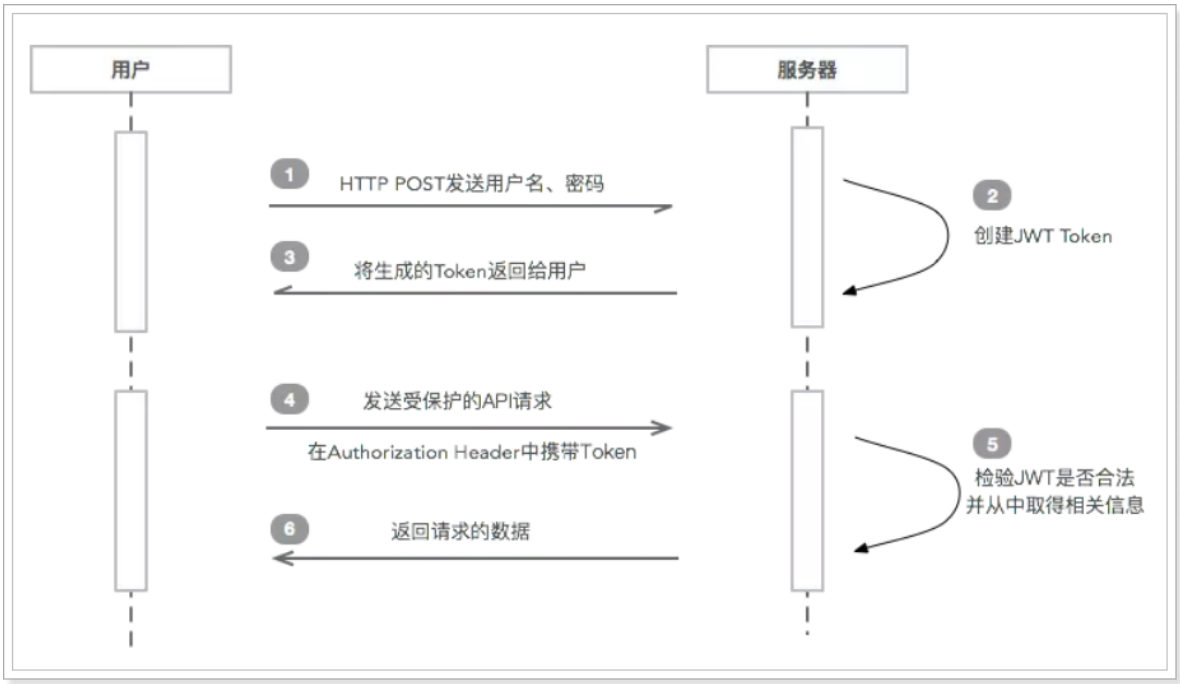
在身份验证过程中, 当用户使用其凭据成功登录时, 将返回 JSON Web token, 并且必须在本地保存 (通常在本地存储中)。

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：

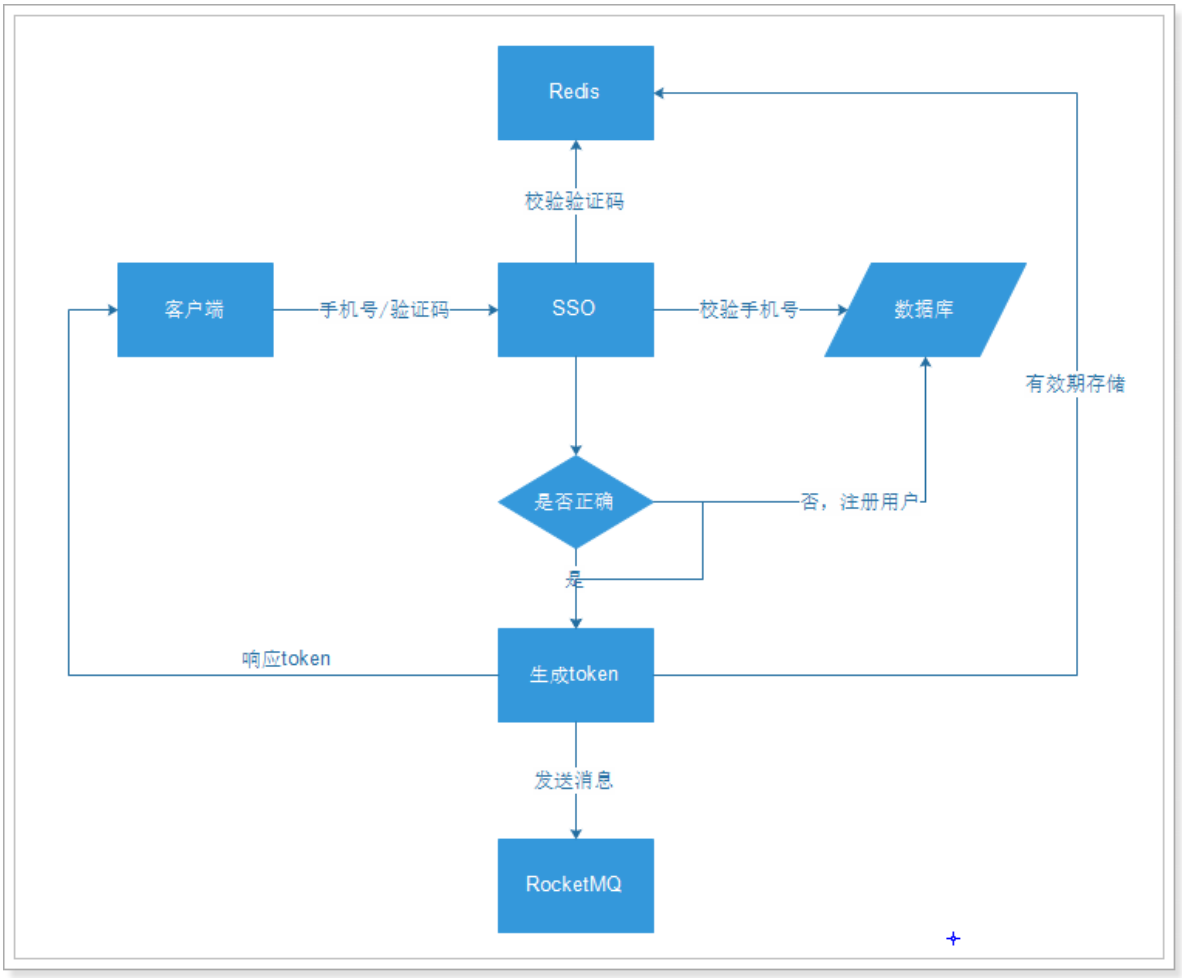
```xml
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

编写测试用例：

```java
public class TestJWT {

    public static void main(String[] args) {

        String secret = "itcast";

        Map<String, Object> claims = new HashMap<String, Object>();
        claims.put("mobile", "12345789");
        claims.put("id", "2");

        // 生成token
        String jwt = Jwts.builder()
                .setClaims(claims) //设置响应数据体
                .signWith(SignatureAlgorithm.HS256, secret) //设置加密方法和加密盐
                .compact();

        System.out.println(jwt);
//eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUiOiIxMjM0NTc4OSIsImlkIjoiMiJ9.VivsfLzrsKFOJo_BdGIf6cKY_7wr2jMOMOIGaFt_tps

        // 通过token解析数据
        Map<String, Object> body = Jwts.parser()
                .setSigningKey(secret)
                .parseClaimsJws(jwt)
                .getBody();

        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.eyJleHAiOjE1NjI4MjkzMzYsInVzZXJfaWQiOiIxIn0.Mbzn6LzsLrkVWEbhexR3lTYDZjxqIcqW11rJxDQ6Ewk
isNew	boolean	必须	true	是否新用户	

3.7.2、UserController

```java
package com.tanhua.sso.controller;

import com.tanhua.sso.pojo.User;
import com.tanhua.sso.service.UserService;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("user")
public class UserController {

    @Autowired
    private UserService userService;


    /**
     * 登录
     *
     * @return
     */
    @PostMapping("loginVerification")
    public ResponseEntity<Object> login(@RequestBody Map<String, Object> param) {
        try {
            String mobile = param.get("phone").toString();
            String code = param.get("verificationCode").toString();
            String token = this.userService.login(mobile, code);

            Map<String, Object> result = new HashMap<>(2);

            if (StringUtils.isNotEmpty(token)) {
                String[] ss = StringUtils.split(token, '|');
                String isNew = ss[0];
                String tokenStr = ss[1];

                result.put("isNew", isNew);
                result.put("token", tokenStr);
                return ResponseEntity.ok(result);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        // 登录失败，验证码错误
        ErrorResult errorResult = ErrorResult.builder().errCode("000000").errMessage("验证码错误").build();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorResult);
    }

}
```

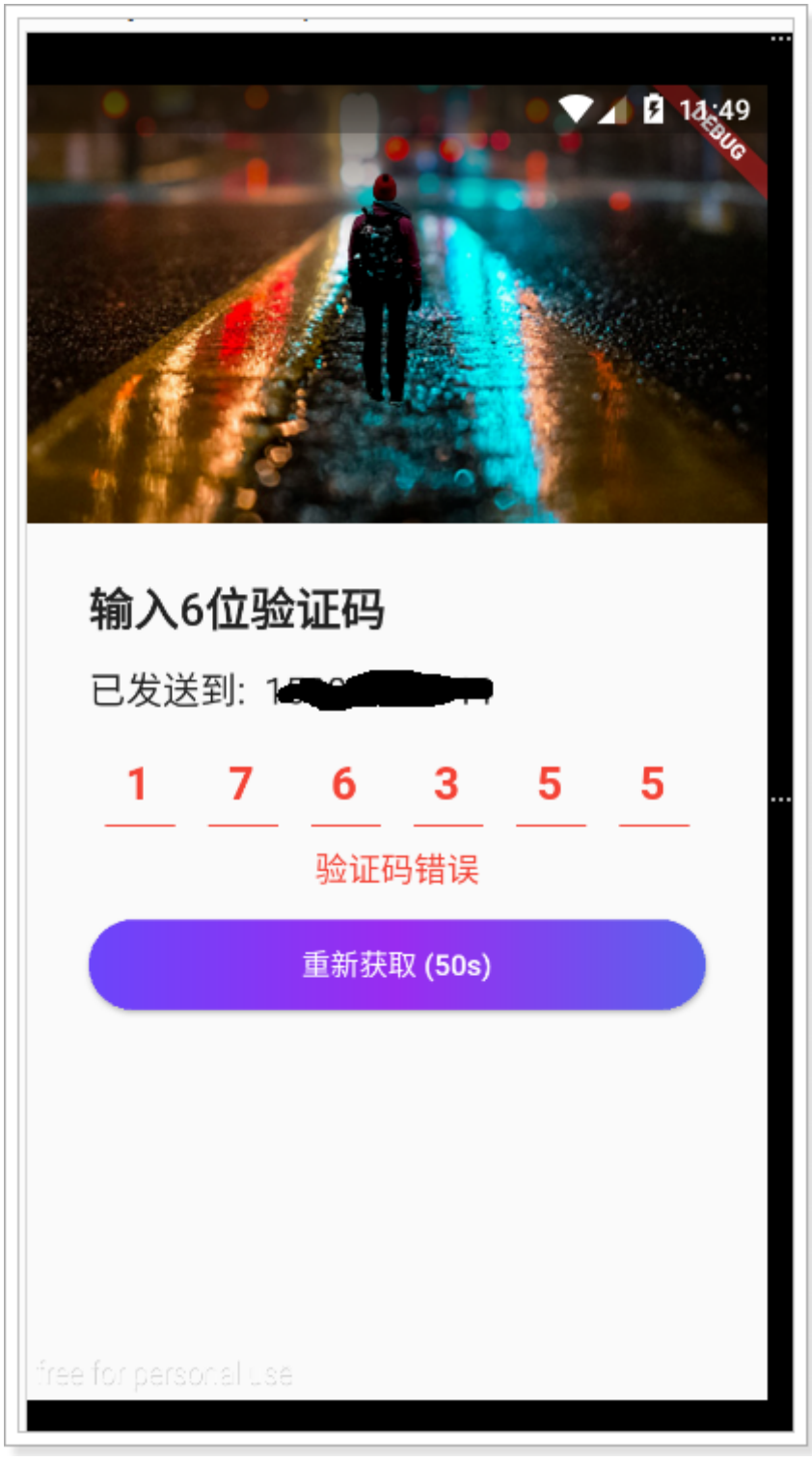
```
53
```

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.Jwts;
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 =
   LoggerFactory.getLogger(UserService.class);
31
32     @Autowired
33     private UserMapper userMapper;
34
35     @Autowired
36     private RedisTemplate<String, String> redisTemplate;
37
38     private static final ObjectMapper MAPPER = new ObjectMapper();
39
40     @Autowired
41     private RocketMQTemplate rocketMQTemplate;
42
43     @Value("${jwt.secret}")
44     private String secret;
45
46     public String login(String mobile, String code) {
47
48         Boolean isNew = false; //是否为新注册
49
50         //校验验证码
51         String redisKey = "CHECK_CODE_" + mobile;
52         String value = this.redisTemplate.opsForValue().get(redisKey);
```

```
        if (!StringUtils.equals(value, code)) {
            return null; //验证码错误
        }

        QueryWrapper<User> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("mobile", mobile);
        User selectUser = this.userMapper.selectOne(queryWrapper);
        if (selectUser == null) {
            // 该手机号未注册，进行注册操作
            User user = new User();
            user.setMobile(mobile);
            user.setPassword(DigestUtils.md5Hex(secret + "_123456"));// 默认密码
            this.userMapper.insert(user);
            selectUser = user;
            isNew = true;
        }

        Map<String, Object> claims = new HashMap<String, Object>();
        claims.put("mobile", mobile);
        claims.put("id", selectUser.getId());

        // 生成token
        String token = Jwts.builder()
                .setClaims(claims) //设置响应数据体
                .signWith(SignatureAlgorithm.HS256, secret) //设置加密方法和加密盐
                .compact();

        //将用户数据写入到redis中
        String redisTokenKey = "TOKEN_" + token;
        try {
            this.redisTemplate.opsForValue().set(redisTokenKey, MAPPER.writeValueAsString(selectUser), Duration.ofHours(1));
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }

        try {
            // 发送登录成功的消息
            Map<String, Object> msg = new HashMap<>();
            msg.put("userId", selectUser.getId());
            msg.put("date", new Date());
            this.rocketMQTemplate.convertAndSend("tanhua-sso-login", msg);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return isNew + "|" + token;
    }

}

```

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

```xml
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>2.8.3</version>
</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：

```
aliyun.endpoint = http://oss-cn-shanghai.aliyuncs.com
aliyun.accessKeyId = LTAIQvQNjLEmJEoa
aliyun.accessKeySecret = nhB33fbK3LSiHj2JgTFc6TbFSCfgaG
aliyun.bucketName=itcast-tanhua
aliyun.urlPrefix=http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/
```

AliyunConfig：

```
package com.tanhua.sso.config;

import com.aliyun.oss.OSSClient;
import lombok.Data;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

@Configuration
@PropertySource("classpath:aliyun.properties")
@ConfigurationProperties(prefix = "aliyun")
@Data
public class AliyunConfig {

    private String endpoint;
    private String accessKeyId;
    private String accessKeySecret;
    private String bucketName;
    private String urlPrefix;

    @Bean
    public OSSClient ossClient() {
        return new OSSClient(endpoint, accessKeyId, accessKeySecret);
    }

}
```

4.1.3、PicUploadService

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

import com.aliyun.oss.OSSClient;
import com.tanhua.sso.config.AliyunConfig;
import com.tanhua.sso.vo.PicUploadResult;
```

```java
import org.apache.commons.lang3.RandomUtils;
import org.apache.commons.lang3.StringUtils;
import org.joda.time.DateTime;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.ByteArrayInputStream;

@Service
public class PicUploadService {

    // 允许上传的格式
    private static final String[] IMAGE_TYPE = new String[]{".bmp", ".jpg",
            ".jpeg", ".gif", ".png"};

    @Autowired
    private OSSClient ossClient;

    @Autowired
    private AliyunConfig aliyunConfig;

    public PicUploadResult upload(MultipartFile uploadFile) {

        PicUploadResult fileUploadResult = new PicUploadResult();

        //图片做校验，对后缀名
        boolean isLegal = false;

        for (String type : IMAGE_TYPE) {
            if
(StringUtils.endsWithIgnoreCase(uploadFile.getOriginalFilename(),
                    type)) {
                isLegal = true;
                break;
            }
        }

        if (!isLegal) {
            fileUploadResult.setStatus("error");
            return fileUploadResult;
        }

        // 文件新路径
        String fileName = uploadFile.getOriginalFilename();
        String filePath = getFilePath(fileName);

        // 上传到阿里云
        try {
            // 目录结构：images/2018/12/29/xxxx.jpg
            ossClient.putObject(aliyunConfig.getBucketName(), filePath, new
                    ByteArrayInputStream(uploadFile.getBytes()));
        } catch (Exception e) {
            e.printStackTrace();
            //上传失败
            fileUploadResult.setStatus("error");
            return fileUploadResult;
        }
```

```
        // 上传成功
        fileUploadResult.setStatus("done");
        fileUploadResult.setName(this.aliyunConfig.getUrlPrefix() +
filePath);

 fileUploadResult.setUid(String.valueOf(System.currentTimeMillis()));

        return fileUploadResult;
    }

    private String getFilePath(String sourceFileName) {
        DateTime dateTime = new DateTime();
        return "images/" + dateTime.toString("yyyy")
                + "/" + dateTime.toString("MM") + "/"
                + dateTime.toString("dd") + "/" +
System.currentTimeMillis() +
                RandomUtils.nextInt(100, 9999) + "." +
                StringUtils.substringAfterLast(sourceFileName, ".");
    }

}
```

所需其他的代码：

PicUploadResult:

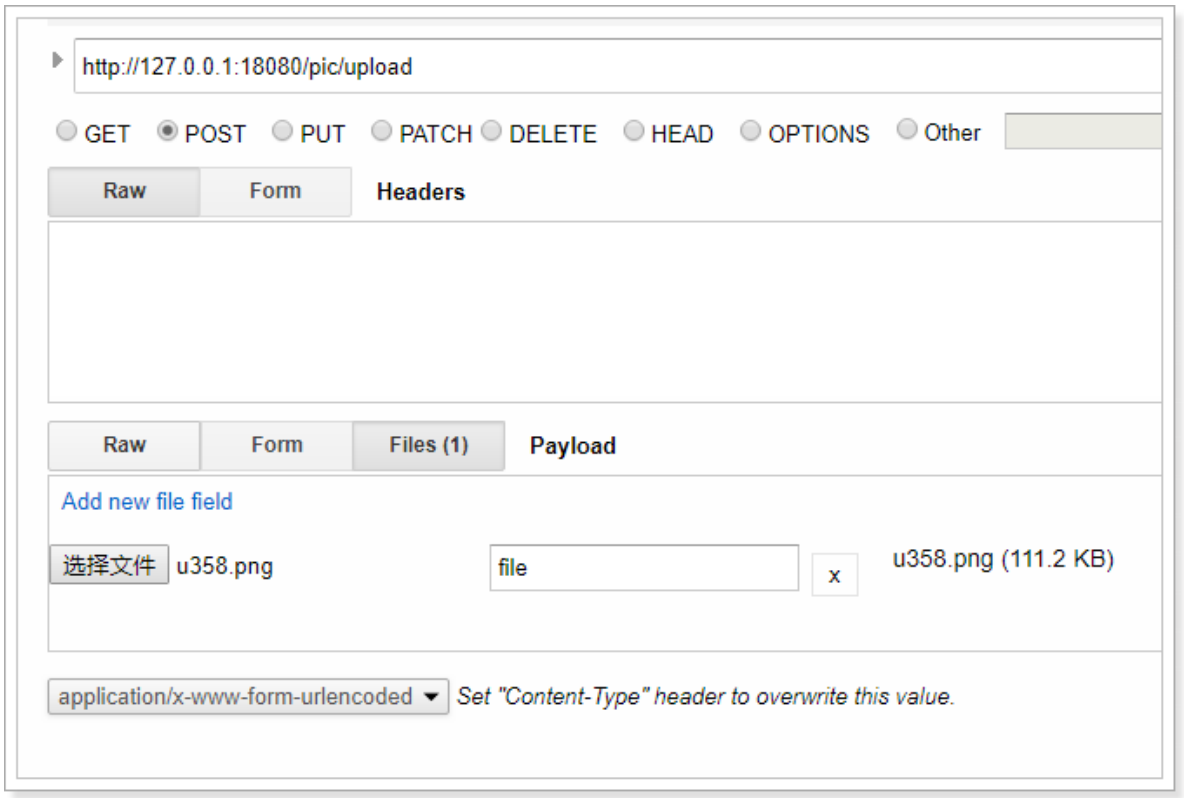
```
package com.tanhua.sso.vo;

import lombok.Data;

@Data
public class PicUploadResult {

    // 文件唯一标识
    private String uid;
    // 文件名
    private String name;
    // 状态有：uploading done error removed
    private String status;
    // 服务端响应内容，如：'{"status": "success"}'
    private String response;

}
```

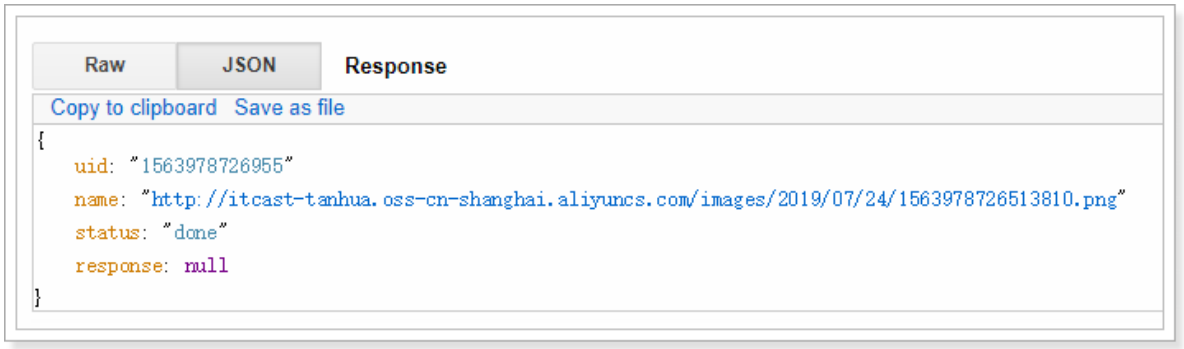
4.1.4、PicUploadController

```
package com.tanhua.sso.controller;

import com.tanhua.sso.service.PicUploadService;
import com.tanhua.sso.vo.PicUploadResult;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

@RequestMapping("pic/upload")
@Controller
public class PicUploadController {

    @Autowired
    private PicUploadService picUploadService;

    @PostMapping
    @ResponseBody
    public PicUploadResult upload(@RequestParam("file") MultipartFile
multipartFile) {
        return this.picUploadService.upload(multipartFile);
    }
}

```

4.1.5、测试

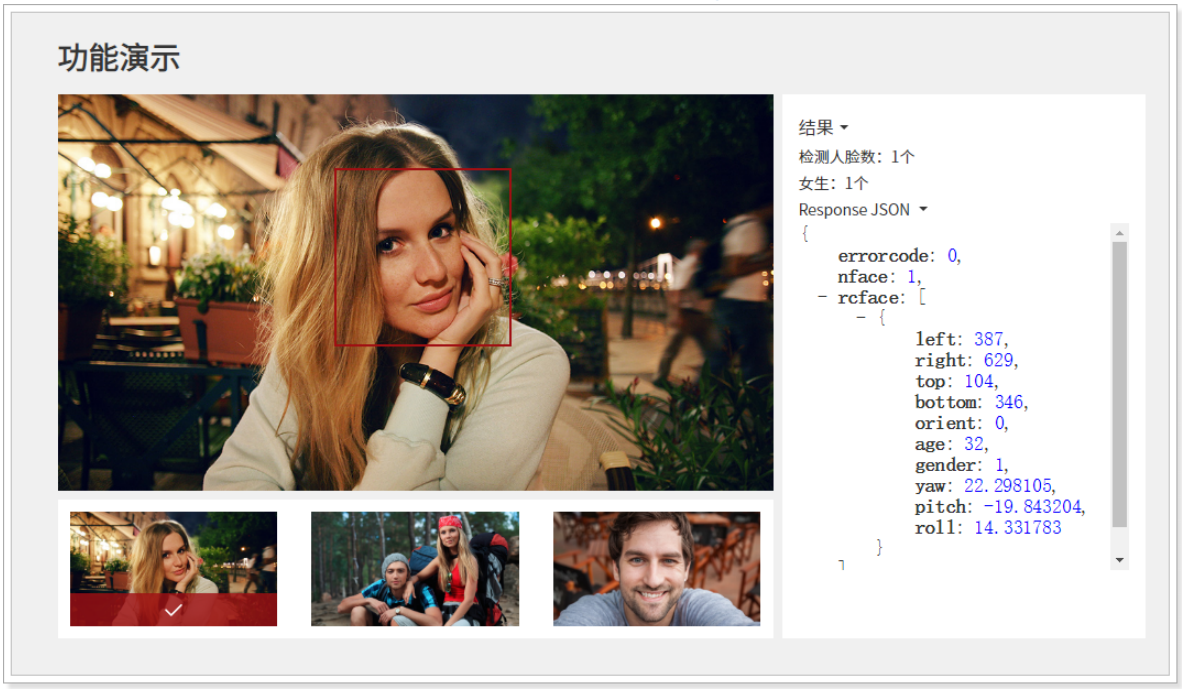





详情
快速使用图片服务
文件名
images/2019/07/24/1563978726513810.png
ETag
565CBA3BF42E17783A9174AF38AF6EAA
使用 HTTPS
URL
https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/07/24/1563978726513810.png
下载
打开文件 URL
复制文件 URL
复制文件路径
类型
image/png
设置 HTTP 头

4.2、人脸识别

人脸识别技术采用虹软开放平台实现（免费使用）。官网：https://www.arcsoft.com.cn/



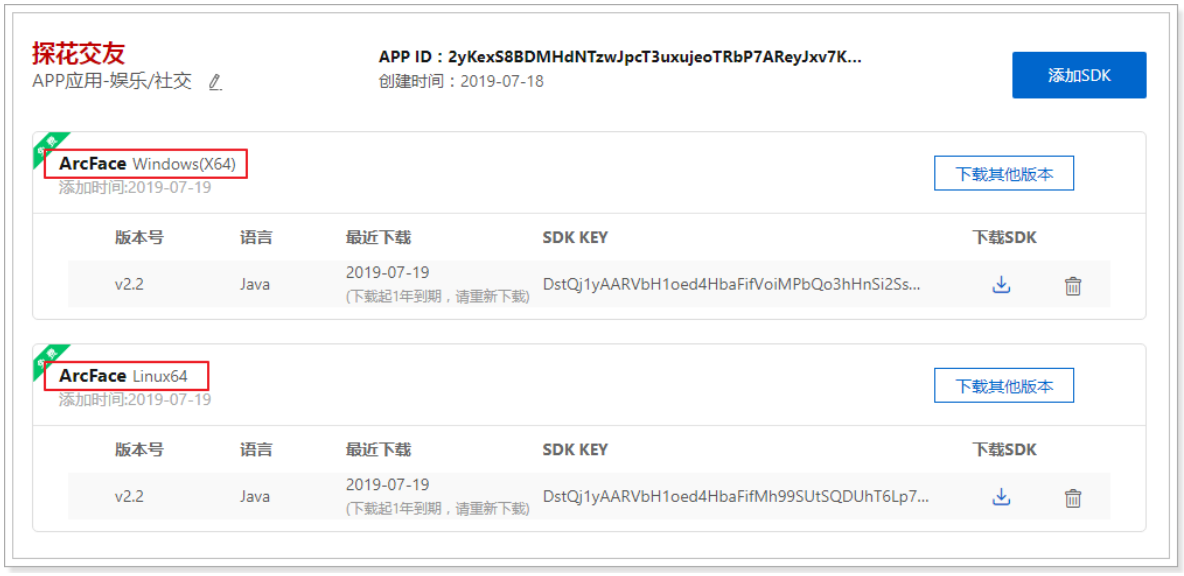
4.2.1、使用说明

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

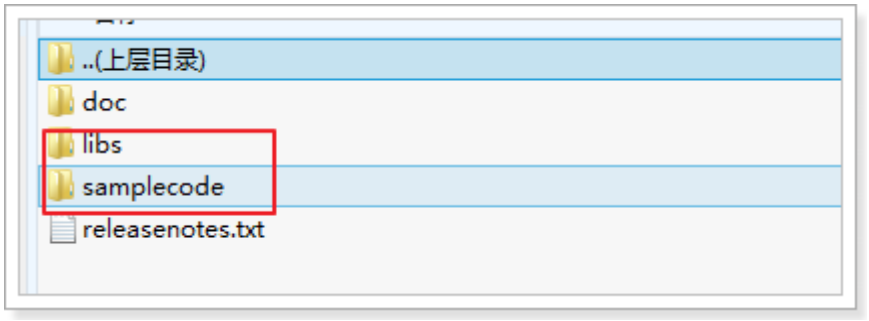
注册完成后进行登录，然后进行创建应用：

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。

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。



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：

```
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：

```
<dependency>
    <groupId>com.arcsoft.face</groupId>
    <artifactId>arcsoft-sdk-face</artifactId>
    <version>2.2.0.1</version>
</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

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

FaceEngineService：

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

import com.arcsoft.face.EngineConfiguration;
import com.arcsoft.face.FaceEngine;
import com.arcsoft.face.FaceInfo;
import com.arcsoft.face.FunctionConfiguration;
import com.arcsoft.face.enums.DetectMode;
import com.arcsoft.face.enums.DetectOrient;
import com.arcsoft.face.enums.ErrorInfo;
import com.arcsoft.face.enums.ImageFormat;
import com.arcsoft.face.toolkit.ImageFactory;
import com.arcsoft.face.toolkit.ImageInfo;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import java.io.File;
import java.util.ArrayList;
import java.util.List;

@Service
public class FaceEngineService {

    private static final Logger LOGGER =
LoggerFactory.getLogger(FaceEngineService.class);

    @Value("${arcsoft.appid}")
    private String appid;

    @Value("${arcsoft.sdkKey}")
    private String sdkKey;

    @Value("${arcsoft.libPath}")
    private String libPath;

    private FaceEngine faceEngine;

    @PostConstruct
    public void init() {
        // 激活并且初始化引擎
        FaceEngine faceEngine = new FaceEngine(libPath);
        int activeCode = faceEngine.activeOnline(appid, sdkKey);
        if (activeCode != ErrorInfo.MOK.getValue() && activeCode !=
ErrorInfo.MERR_ASF_ALREADY_ACTIVATED.getValue()) {
            LOGGER.error("引擎激活失败");
            throw new RuntimeException("引擎激活失败");
        }

        //引擎配置
        EngineConfiguration engineConfiguration = new
EngineConfiguration();
        //IMAGE检测模式，用于处理单张的图像数据

 engineConfiguration.setDetectMode(DetectMode.ASF_DETECT_MODE_IMAGE);
        //人脸检测角度，逆时针0度
```

```
 engineConfiguration.setDetectFaceOrientPriority(DetectOrient.ASF_OP_0_ONLY);

        //功能配置
        FunctionConfiguration functionConfiguration = new FunctionConfiguration();
        functionConfiguration.setSupportAge(true);
        functionConfiguration.setSupportFace3dAngle(true);
        functionConfiguration.setSupportFaceDetect(true);
        functionConfiguration.setSupportFaceRecognition(true);
        functionConfiguration.setSupportGender(true);
        functionConfiguration.setSupportLiveness(true);
        functionConfiguration.setSupportIRLiveness(true);

 engineConfiguration.setFunctionConfiguration(functionConfiguration);

        //初始化引擎
        int initCode = faceEngine.init(engineConfiguration);

        if (initCode != ErrorInfo.MOK.getValue()) {
            LOGGER.error("初始化引擎出错!");
            throw new RuntimeException("初始化引擎出错!");
        }

        this.faceEngine = faceEngine;
    }

    /**
     * 检测图片是否为人像
     *
     * @param imageInfo 图像对象
     * @return true:人像，false:非人像
     */
    public boolean checkIsPortrait(ImageInfo imageInfo) {
        // 定义人脸列表
        List<FaceInfo> faceInfoList = new ArrayList<FaceInfo>();
        faceEngine.detectFaces(imageInfo.getImageData(), imageInfo.getWidth(), imageInfo.getHeight(), ImageFormat.CP_PAF_BGR24, faceInfoList);
        return !faceInfoList.isEmpty();
    }

    public boolean checkIsPortrait(byte[] imageData) {
        return this.checkIsPortrait(ImageFactory.getRGBData(imageData));
    }

    public boolean checkIsPortrait(File file) {
        return this.checkIsPortrait(ImageFactory.getRGBData(file));
    }

}

```

4.2.4、测试

```
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 =
   this.faceEngineService.checkIsPortrait(file);
22         System.out.println(checkIsPortrait); // true|false
23     }
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
  - 

- 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

```
package com.tanhua.sso.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.tanhua.sso.pojo.UserInfo;

public interface UserInfoMapper extends BaseMapper<UserInfo> {
}

```

4.3.2、UserInfoService

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

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.tanhua.sso.enums.SexEnum;
import com.tanhua.sso.mapper.UserInfoMapper;
import com.tanhua.sso.pojo.User;
import com.tanhua.sso.pojo.UserInfo;
import com.tanhua.sso.vo.PicUploadResult;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.util.Map;

@Service
public class UserInfoService {

    private static final Logger LOGGER =
LoggerFactory.getLogger(UserInfoService.class);

    @Autowired
    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,
    String> param, @RequestHeader("Authorization") String token) {
32          try {
33              Boolean saveUserInfo = this.userInfoService.saveUserInfo(param,
    token);
34              if (saveUserInfo) {
35                  return ResponseEntity.ok(null);
36              }
37          } catch (Exception e) {
38              e.printStackTrace();
39          }
40
41          ErrorResult errorResult =
    ErrorResult.builder().errCode("000000").errMessage("发生错误").build();
```

```
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorResult);
    }

    /**
     * 上传头像
     *
     * @param file
     * @param token
     * @return
     */
    @RequestMapping("loginReginfo/head")
    @PostMapping
    public ResponseEntity<Object> saveLogo(@RequestParam("headPhoto") MultipartFile file, @RequestHeader("Authorization") String token) {
        try {
            Boolean bool = this.userInfoService.saveLogo(file, token);
            if(bool){
                return ResponseEntity.ok(null);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        ErrorResult errorResult = ErrorResult.builder().errCode("000000").errMessage("图片非人像，请重新上传!").build();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(errorResult);
    }

}

```

4.4.4、测试

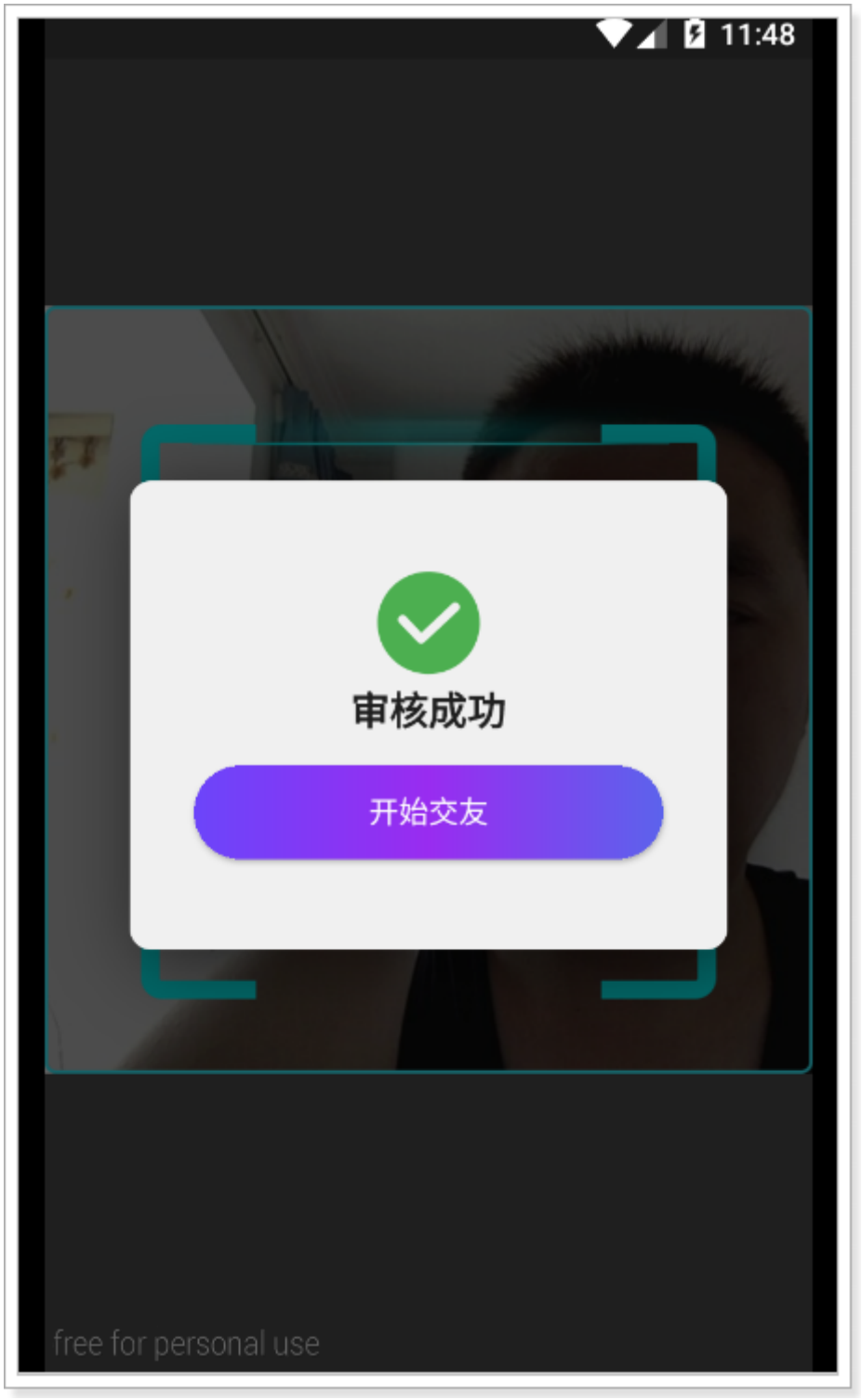
输入6位验证码

已发送到: 1[redacted]

5 8 1 2 3

重新获取 (50s)


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

```
    public User queryUserByToken(String token) {
        try {
            String redisTokenKey = "TOKEN_" + token;
            String cacheData =
this.redisTemplate.opsForValue().get(redisTokenKey);
            if (StringUtils.isEmpty(cacheData)) {
                return null;
            }
```

```
            // 刷新时间
            this.redisTemplate.expire(redisTokenKey, 1, TimeUnit.HOURS);
            return MAPPER.readValue(cacheData, User.class);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

5.2、UserController

```
    /**
     * 根据token查询用户数据
     *
     * @param token
     * @return
     */
    @GetMapping("{token}")
    public User queryUserByToken(@PathVariable("token") String token) {
        return this.userService.queryUserByToken(token);
    }
```

5.3、测试