即时通讯(IM) 服务端开发

目录

目录

一、项目整体介绍

1.项目简介

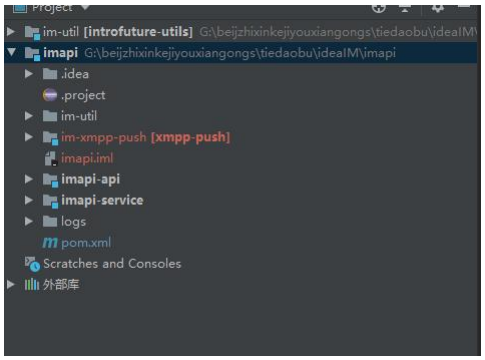


图 1

如上图 1 所示 IM 服务端共 5 个项目

1) imapi 是用来管理 imapi-api 和 imapi-service 项目的没有具体的代码逻 辑,可以关闭该项目。

2) imapi-api 为项目提供数据接口,依赖于 imapi-service, imapi-service 负责接口相关业务处理，数据操作、存储。

3) xmpp-push 是用于 发送服务器发出的消息协议通知

4) push 是用于 用户离线时调用第三方平台 发送离线通知

3) upload 是文件上传服务，用来在聊天的过程中上传用户头像、聊天过程中的图片、文件、语音、视频等。

说明：在单聊及群聊中，上传图片、音视频文件时，我们先调用此接口上传至服务器，形成 URL，这样在发消息时，只要发此 URL 即可，大大节省流量；同时也用于上传头像。Upload 项目下，UploadAvatarServlet：头像上传、UploadServlet：资源上传。

2.服务端架构

1) 项目使用spring boot框架，软件架构如下图：

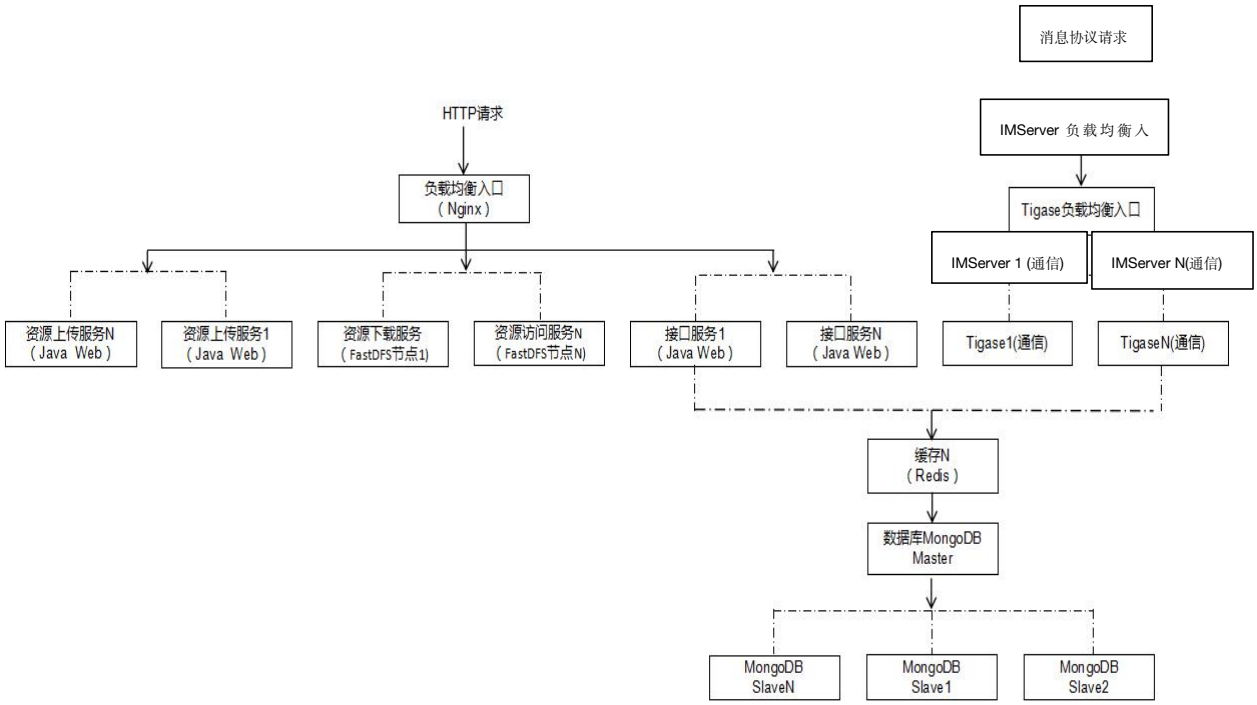


图 2 软件架构

3.框架说明

1) imapi 、imapi-service

使用 Maven 构建, 基于 spring、spring-boot、spring-mvc、jedis、redisson、mongo-java-driver 等框架开发。

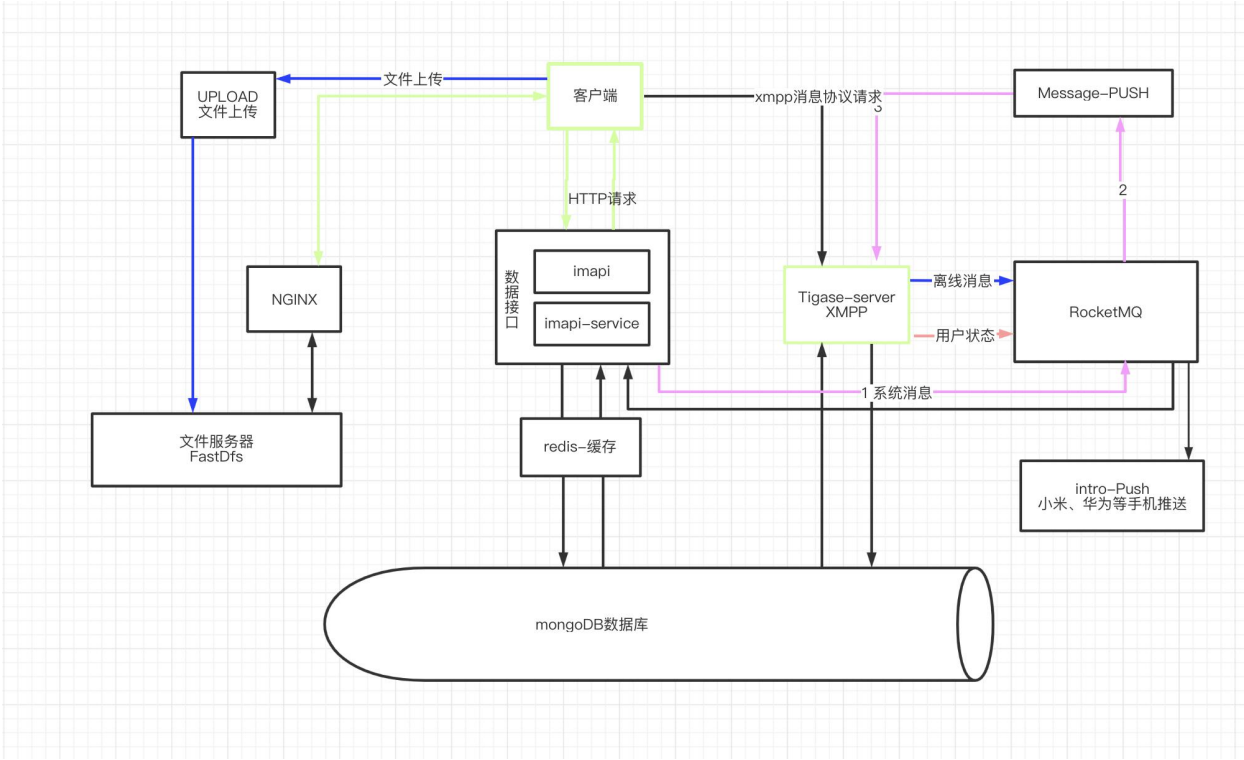
主要框架职责：

spring、spring-boot	项目基础架构
spring-mvc	Http 请求接入
redisson	Redis 缓存访问
morphia、mongo-java-driver	Mongodb 数据库访问

2) upload 文件上传服务

使用 Eclipse Web Project 构建，基于 J2EE Servlet 开发

3.项目关系图



登录过程：客户端调用 http 登录接口进行登录，登录成功后，发送 请求进行 登录。

二、项目类结构及功能

1、imapi

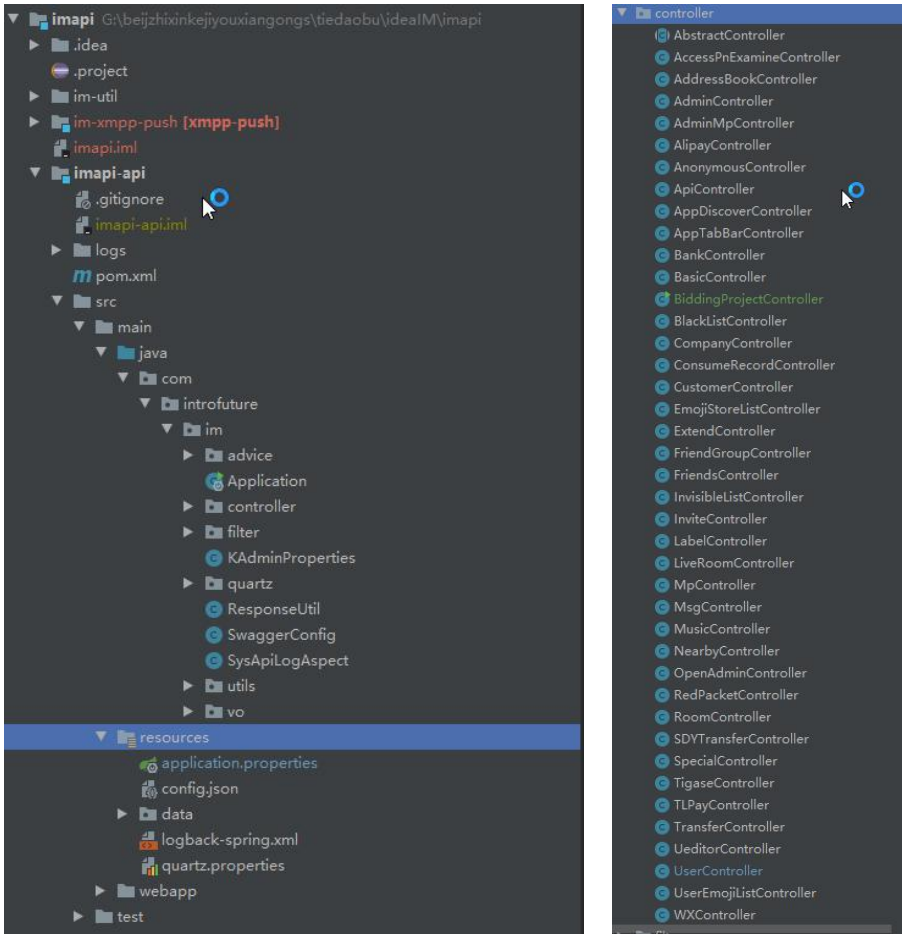


图 3

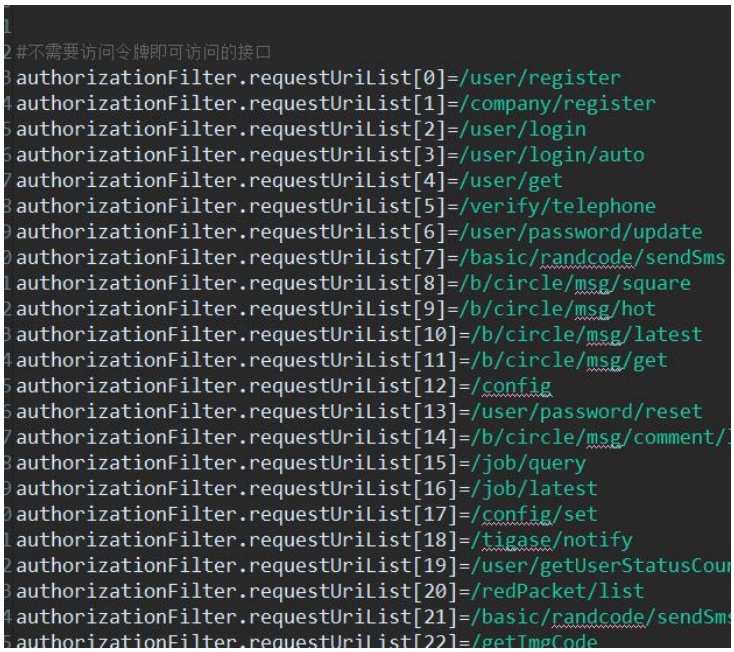


图 4　配置不进行 token 验证的接口

accessToken

accessToken 是通过 UUID 产生的一个唯一的数据，并且不会重复，在每一次登陆的时候都会产生一个 accessToken，并存放在 redis 缓存中，在每次退出的时候都会被销毁。在用户每次请求服务器的方法的时候，我们要求其请求时传入用户的 accessToken，在请求进入方法前都会有一个 AuthorizationFilter 过滤器，去校验判断 accessToken，如果，accessToken 不正确或不存在，则该请求服务器不会响应。

所在包	类名	说明
	Application	应用启动类
	KAdminProperties	用于读取后台管理员配置信息
	ResponseUtil	响应的统一处理
advice	ExceptionHandlerAdvice	异常处理
filter	AuthorizationFilter	请求认证拦截器
	AuthorizationFilterProperties	用于读取拦截配置信息（如上图 4）
controller	AbstractController	数据接口父类
	AdminController	后台管理相关数据接口
	AdminMpController	公众号管理数据接口
	BasicController	验证码、短信、配置等基础数据接口
	CompanyController	公司组织架构模块相关数据接口
	ConsumeRecordController	消费记录相关数据接口

	CustomerController	客服模块相关数据接口
	FriendsController	好友模块相关数据接口
	LiveRoomController	直播房间相数据接口
	MpController	公众号相关数据接口
	MsgController	朋友圈相关数据接口
	NearbyController	附近的人相关的数据接口
	RedPacketController	红包相关的数据接口
	RoomController	群组相关的数据接口
	TigaseController	Tigase 支持接口（单聊记录、群聊记录等）
	UserController	用户模块相关数据接口
	~~WXController~~	~~微信相关数据接口~~

2、imapi-service

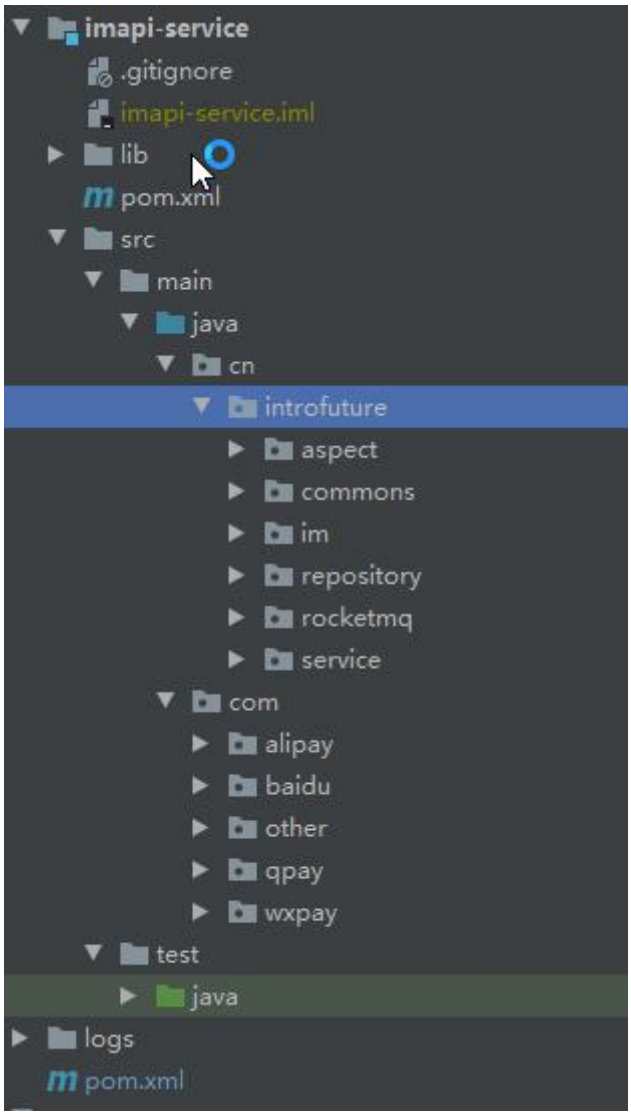
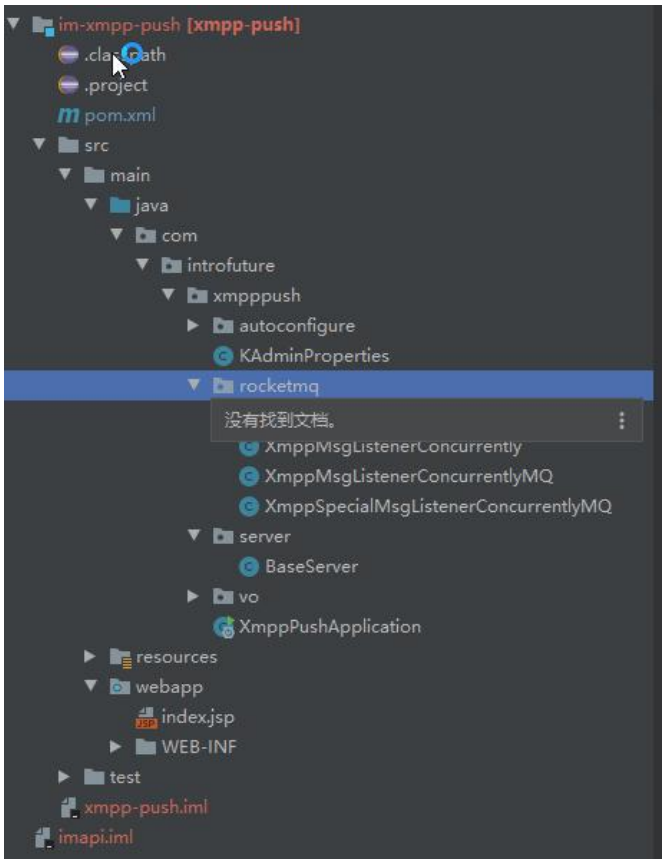


图 5

2、Imapi-xmpp



所在包	类名/文件名	说明
apns	voippush.p12	ios 语音、视频推送所需要的证书文件
cn.introfuture.commons.autoconfigure	CommAutoConfiguration	配置管理器
	KApplicationProperties	用于配置读取
	KMongoAutoConfiguration	Mongodb 配置管理
	KRedisAutoConfiguration	Redis 配置管理
cn.introfuture.commons.constants	KConstants	常量
	KConstantsUtil	
	KKeyConstant	获取键值相关的常量和静态方法 （键值：redis 里对应数据的 key）
	MsgType	消息类型的常量
cn.introfuture.commons.ex	ServiceException	Service 异常
cn.introfuture.commons.support.jackson	ObjectIdDeserializer	MongoDB ObjectId 对象反序列化
	ObjectIdSerializer	MongoDB ObjectId 对象序列化
cn.introfuture.commons.support.mongo	MongoOperator	MongoDB 操作符常量
cn.introfuture.commons.support.spring	SpringBeansUtils	

cn.introfuture.commons.support.spring.converter	MappingFastjsonHttpMessageConverter	http 消息转换器
cn.introfuture.commons.support.spring.ds	DbContextHolder	
cn.introfuture.commons.utils	Base64	Base64 编解码工具
	BeanUtils	
	Blowfish	Blowfish 加解密
	CollectionUtil	集合工具
	ConfigUtil	
	DateUtil	日期工具
	DES	DES 加密解密（消息内容加密解密）
	FileUtil	文件工具 主要用于：删除文件(图片、视频、语音)
	HttpClientUtil	http 客户端工具
	HttpUtil	http 工具类
	JSONUtil	Json 工具类
	KTaglibUtils	
	MapUtil	
	Md5Util	Md5 工具类
	ParamsSign	HTTP 请求参数签名
	PayConstants	支付宝服务窗环境常量
	ReflectionUtils	反射工具类
	ReqUtil	请求工具类
	StringUtil	字符工具类
	ThreadUtil	线程工具类
	ValidateCode	图片验证码生成器
	ValueUtil	值类型转换
	WebappUtil	
	WebNetEncode	字符编码工具
	ZHUtils	繁体简体转换
cn.introfuture.commons.vo	JSONMessage	Restful 接口返回值对象
cn.introfuture.commons	IdWorker	
cn.introfuture.im.example (用于数据传递的实体类)	AddCommentParam	添加评论参数
	AddGiftParam	添加礼物参数
	AddMsgParam	添加朋友圈消息参数
	BaseExample	基础数据传递实体
	BasePoi	
	BindExample	
	~~CompanyExample~~	~~公司数据传递实体~~
	ContactsParam	
	KSession	
	MessageExample	
	NearbyUser	附近用户数据传递实体类

	PageExample	列表分页相关数据传递类
	UserExample	用户数据传递类
	UserQueryExample	用于用户查询参数传递
cn.introfuture.im.scheduleds	CommTask	定时任务（用户统计 红包退回等）
	MsgListScheduled	朋友圈消息定时刷新
cn.introfuture.im.service （业务接口）	AdminManager	后台管理业务接口
	CompanyManager	公司业务接口
	CustomerManager	客服模块业务接口
	FriendsManager	好友业务接口
	MPService	公众号业务接口
	RoomManager	群组业务接口
	UserManager	用户业务接口
cn.introfuture.im.service.impl	AdminManagerImpl	后台管理业务类
	CompanyManagerImpl	公司相关业务类
	ConsumeRecordManagerImpl	消费记录相关业务类
	CustomerManagerImpl	客服模块相关业务类
	FriendsManagerImpl	好友相关业务类
	LiveRoomManagerImpl	直播房间业务类
	MongoRepository	
	MPServiceImpl	公众号相关业务类
	RedPacketManagerImpl	红包相关业务类
	RoomManagerImplForIM	聊天群组相关业务类
	UserManagerImpl	用户相关业务类
cn.introfuture.im.utils	KSessionUtil	
	ConstantUtil	
cn.introfuture.im.vo (实体类，和数据库对应)	Areas	地区 --对应库-- **tb_areas**
	Black	黑名单 --对应库--**black**
	Comment	评论 --对应库-- **s_comment**
	CommonText	常用语 --对应库-- **commonText**
	CompanyVO	公司 --对应库-- **company**
	Config	配置 --对应库-- **config**
	~~Constant~~	--对应库-- **tb_constants**
	ConsumeRecord	消费记录 --对应库-- **ConsumeRecord**
	Course	课程 --对应库-- **course**
	CourseMessage	课程消 --对应库-- **courseMessage**
	Customer	客服模块访客 --对应库-- **customer**
	DepartmentVO	部门 --对应库-- **department**
	Emoji	收藏（聊天收藏图片/表情） --对应库-- **emoji**
	Employee	员工 --对应库-- **employee**
	Fans	粉丝 --对应库-- **u_fans**
	Friends	好友 --对应库-- **u_friends**
	Gift	礼物 --对应库-- **s_gift**
	Givegift	赠送礼物记录 --对应库-- **givegift**

	Language	语言
	LiveRoom	直播间 --对应库-- **LiveRoom**
	~~Member~~	
	Menu	公众号菜单 --对应库-- **mp_menu**
	Msg	朋友圈消息 --对应库-- **s_msg**
	MsgNotice	推送通知 --对应库-- **Notice**
	NewFriends	新朋友 --对应库-- **NewFriends**
	PageVO	
	PhotoVO	图片
	Praise	点赞 --对应库-- **s_praise**
	RedPacket	红包 --对应库-- **RedPacket**
	RedReceive	收红包记录 --对应库-- **RedReceive**
	Report	举报 --对应库-- **Report**
	Room	群组 --对应库-- **shiku_room**
	SmsRecord	短信记录--对应库-- **SmsRecord**
	User	用户--对应库-- **user**
	UserStatusCount	用户状态统计--对应库-- **UserStatusCount** **（用于后台用户走势图）**
	WxUser	微信公众号群聊模块用户--对应库-- **wxuser**
cn.introfuture.repository	BaseRepository	基础数据操作接口
	CompanyRepository	组织架构模块公司数据操作接口
	CustomerRepository	客服模块数据操作接口
	DepartmentRepository	组织架构模块部门数据操作接口
	EmployeeRepository	组织架构模块员工数据操作接口
	FriendsRepository	好友模块数据操作接口
	MongoRepository	数据操作接口父类
	MsgCommentRepository	朋友圈评论数据操作接口
	MsgGiftRepository	礼物数据操作接口
	MsgListRepository	消息数据操作接口
	MsgPraiseRepository	点赞数据操作接口
	MsgRepository	朋友圈数据操作接口
	UserRepository	用户数据操作接口
cn.introfuture.repository.mongo	BaseRepositoryImpl	基础数据操作类
	CompanyRepositoryImpl	公司数据操作类
	ConsumeRecordRepositoryImpl	消费记录数据操作类
	CustomerRepositoryImpl	客服模块数据操作类
	DepartmentRepositoryImpl	部门数据操作类
	EmployeeRepositoryImpl	员工数据操作类
	FriendsRepositoryImpl	好友数据操作类
	MsgCommentRepositoryImpl	朋友圈评论数据操作类
	MsgGiftRepositoryImpl	礼物数据操作类
	MsgListRepositoryImpl	消息数据操作类

	MsgPraiseRepositoryImpl	点赞数据操作类
	MsgRepositoryImpl	朋友圈数据操作类
	RedPacketRepositoryImpl	红包数据操作类
	UserRepositoryImpl	用户数据操作类
cn.introfuture.service	ApnsPushService	集成苹果 Apns 推送
	BaiduPushService	集成百度推送
	HWPushService	集成华为推送通知栏消息
	HWPushTransService	华为透传消息推送
	KSMSServiceImpl	短信发送，当前集成的是天天国际短信
	KServiceImpl	服务用于调用 tigase 聊天/建群等
	XMPushService	集成小米推算通知栏消息
	XMPushTransService	集成小米透传消息
com.alipay.config	AlipayConfig	支付宝配置信息
com.alipay.sign	Base64	
	RSA	RSA 签名
com.alipay.util	AlipayCore	支付宝接口公用函数类 该类是请求、通知返回两个文件所调用的公用函数核心处理文件，不需要修改
	AlipayNotify	支付宝通知处理
	AliPayParam	支付工具类，用于生成订单/解析返回数据等
	AliPayUtil	
	UtilDate	
com.wxpay.utils	GetWxOrderno	获取支付订单号/支付连接等
	JsonResult	
	MD5Util	MD5 工具类
	RequestHandler	支付请求
	ResponseData	支付响应
	Sha1Util	创建 SHA1 签名
	WXNotify	微信支付通知处理
	WxPayDto	微信支付订单
	WxPayResult	微信支付回调
	WXPayUtil	支付工具类
com.wxpay.utils.http	HttpClientConnectionManager	Http 客户端连接管理
	HttpConnect	Http 连接
	HttpRequest	Http 请求对象的封装
	HttpResponse	Http 返回对象的封装
	HttpResultType	表示 Http 返回的结果字符方式
	MySSLSocketFactory	
	TrustAnyTrustManager	

4、upload 上传服务

所在包	类名	说明
com.introfuture.commons	SystemConfig	配置读取
com.introfuture.commons.utils	DateUtils	时间工具类
	FileUtils	文件工具类
	ThumUtils	缩略图工具类
com.introfuture.commons.vo	FileType	文件类型
	JMessage	JSON 模型
	UploadItem	上传文件模型
com.introfuture.servlet	AmrToMp3Servlet	Amr 转 MP3 格式
	DeleteFileServlet	删除文件
	UploadAvatarServlet	头像上传
	UploadHadImgServlet	
	UploadifyAvatarServlet	jquery-upload 头像上传
	UploadifyServlet	jquery-upload 文件上传
	~~UploadResumeAvatarServlet~~	~~简历上传~~
	UploadServlet	文件上传
	SystemConfig-win.properties	windows 配置
	SystemConfig.properties	linux 配置

三、主要模块介绍

1、用户模块

1. **注册新用户**：

请求 UserController 中的 registe()方法，首先会调用 userRepository.addUser()方法在业务服务器中添加用户，然后调用中的 register()方法注册用户到 tigase 中。

2. **用户登陆**：

请求 UserController 中的 login()方法，密码采用 md5 加密方式，登陆成功后会返回登陆用户数据，并会产生登陆用户的 accessToken，用户在请求其他方法时需要验证该 accessToken 是否有效。accessToken 是通过 UUID 产生的一个唯一的数据，并且不会重复，再用每一次登陆的时候都会产生一个 accessToken，并存放在 redis 缓存中，在每次退出的时候都会被销毁。

3. **用户登出**：

请求 UserController 中的 logout()方法，会将 redis 中的 accessToken 删除。这样用户没有 accessToken，就进行操作。

2、朋友模块

1.加好友：

请求 FriendsController 中的/friends/add 方法，会直接成为好友。而请求/friends/attention/add 方法可以让用户增加好友验证机制，只有在用户同意的情况下，才能添加好友，否则只是关注。

2.删除好友：

请求 FriendsController 中的/friends/delete 方法，该方法会根据当前用户 userId 和用户好友 userId 删除双方所有关系，将数据库中的 u_fans 和 u_friends 表中数据删除。

3.好友列表：

请求 FriendsController 中的/friends/attention/list 方法。该方法会根据当前用户的 userId 去数据库查询好友 u_friends 表，返回 list 集合。

3、群组模块

1.新建房间：

请求 RoomController 中的/room/add 方法，该方法会调用 RoomManager.add()方法。会把该房间的信息保存到 tigase 中。同时也会把房间创建者存入 tigase 房间成员表，如果创建房间时有邀请好友则会把邀请的好友存入 tigase 房间成员表中。同时会以 1005 号发送推送，新成员 type=907 的消息推送。

2.删除房间：

请求 RoomController 中的/room/delete 方法，该方法会调用 RoomManager.delete()方法。通过 roomId 删除房间，然后会以 1005 号发送 chatType=1 的删除房间 type=903 的通知，再删除房间成员。

3.加入房间：

请求 RoomController 中的/room/join 方法，该方法会调用 RoomManager.join()方法，首先会判断改用户是否在改群组，如果不在则会在成员表中添加一条数据。然后会以 1005 号发送 chatType=1 的新增成员 type=907 的消息推送。

4.退出房间：

请求 RoomController 中的/room/member/delete 方法，该方法会调用 RoomManager.deleteMember()方法，在成员表中删除该成员，然后会以 1005 号发送 chatType=1 的退出成员 type=904 的消息。

5.房间列表：

请求 RoomController 中等/room/list 方法，该方法会调用 RoomManager.seleteList()方法。

6.房间成员列表：

请求 RoomController 中的/room/member/list 方法，该方法会调用 RoomManager.getMemberList()方法。

4、公司模块

1.**创建公司**：请求 CompanyController 中的/org/company/create 方法，该方法会调用 companyManager.createCompany()方法，首先该方法会检查是否有重名的公司，如果没有然后会在数据库中添加一条公司记录，然后会默认的给该公司添加一个根部门，再默然添加创建两个部门（人事部、财务部）；然后会将创建者默认添加到人事部，权限为最高。

2.**删除公司**：请求 CompanyController 中的/org/company/delete 方法，该方法会调用 CompanyManager.deleteCompany()方法，通过 userId 和 comId 删除公司。首先判断删除者是否为创建者，只有创建者才能删除。

5、关键词过滤

通过配置 tigase 中 etc 下的 init-mongo.properties 文件中的—confirm-open-keyword 的值 1 为打开，0 为关闭，动态的选择打开或是关闭关键词过滤功能。在 tigase 中的 introfuture/keywordFilter 方法，所有发送的消息都经过该方法，但是只对 type=1 的消息，也就是文本消息做相应的处理。在数据库中有一张 notKeyword 关键词库表，在有消息进入方法时，会将词库的内容与发送的消息内容进行比对，如果在消息内容中发现关键词则返回 null，对方就不会收到该条消息。

6、后台管理模块

AdminController 后台管理的类，所有后台操作都会调其中的方法。

后台用户登陆：请求 AdminController 中的 login() 方法,login 方法有两个，method={RequestMethod.GET} 代表 get 请求，返回 login 页面。Method={RequestMethod.POST}代表 POST 请求，用校验用户登陆。

主要分为如下几大模块方便用户管理：

1. **用户在线走势图**：console/userStatus
   a) 可以对用户在线状态进行监控，了解用户使用情况
2. **红包列表**：console/redPacketList
   请求 AdminController 中的 /redPacketList 方法，该方法会调用 RedPacketManager.getRedPacketList()方法，到数据库查询出所有用户发送的红包列表。
3. **单聊聊天记录管理**：console/chat_logs_all
   a) 请求 AdminController 中的/chat_logs_all 方法，该方法会查询出所有的单聊消息
4. **群组聊天记录管理**：/console/groupchat_logs_all
   a) 请求 AdminController 中的/groupchat_logs_all 方法，查询出群组中每个人的聊天消息
5. Config **表配置**：/config/set
   通过 Config 表可以修改连接的机器的 IP 或者是域名(例：192.168.0.128)，直播的地址(例:rtmp://live.hkstv.hk.lxdns.com:1935/live/)，接口地址(例：http://192.168.0.128:8092/)，头像访问的地址(例：http://file.youjob.co/)，资

源访问的地址(例:http://file.youjob.co/)，资源上传的地址(例:http://upload.youjob.co/)，freeswitch 地址(例:120.24.211.24)，是否打开红包等。

6. **房间管理**：/console/roomList
   a) 房间列表：请求 AdminController 中的/roomList 方法，查询出系统中所有的群组。同时该方法也可以通过房间名字 roomName 条件查询。
   b) 新建房间：请求 AdminController 中的/addRoom 方法，该方法会返回 editRoom.jsp 视图。进入 editRoom.jsp 提交数据，进入/addRoom 方法，该方法只允许 POST 请求进入，然后调用 roomManager.add()方法，将群组数据保存在数据库。
   c) 删除房间：请求 AdminController 中的/deleteRoom 方法，该方法会调用 roomManager.delete()方法，通过 roomId 删除房间，再删除房间成员。
   d) 成员管理：请求 AdminController 中的/roomUserManager 方法，该方法会调用 roomManager.getMemberListByPage()方法，查询出该房间的所有成员。
   e) 删除成员：请求 AdminController 中的/deleteMember 方法，该方法会调用 roomManager.deleteMenber()，将该成员删除，并且以 10005 号发送 type=904 的消息。
7. **用户管理**：console/userList
   a) 用户列表：请求 AdminController 中的/userList 方法，该方法允许传入多个条件，进行条件查询，当不传入多个值的时候，会查询出系统中所有的用户。用户可以根据用户名、用户在线状态等进行条件查询。
   b) 删除用户：请求 AdminController 中的/delete 方法，该方法首先会删除数据库中的 user 数据, 然后会删除 tigase 中的 user 数据, 然后会删除用户关系表。

8. **提示信息管理**: console/messageList
   a) 查询出系统中对用户不同的错误操作进行的不同返回
9. **关键词管理**: console/keywordfilter
   a) 在启用关键词过滤功能后，可以对关键词进行增删改查。
   b) 关键词列表: 请求 AdminController 中的/keywordfilter 方法, 该方法会去 tigase 数据库中查询 notKeyword 表，如果支持条件查询，如果参数 word 不为空，则通过 word 查询。
   c) 新增词库：请求 AdminController 中的/keywordEdit 方法，该方法会返回 addkeyword.jsp，当保存数据的时候，会调用/addkeyword 方法，将敏感词存入数据库。
   d) 删除关键词：请求 AdminController 中的/deletekeyword 方法，该方法会通过 id 删除对应数据。

四、开放平台

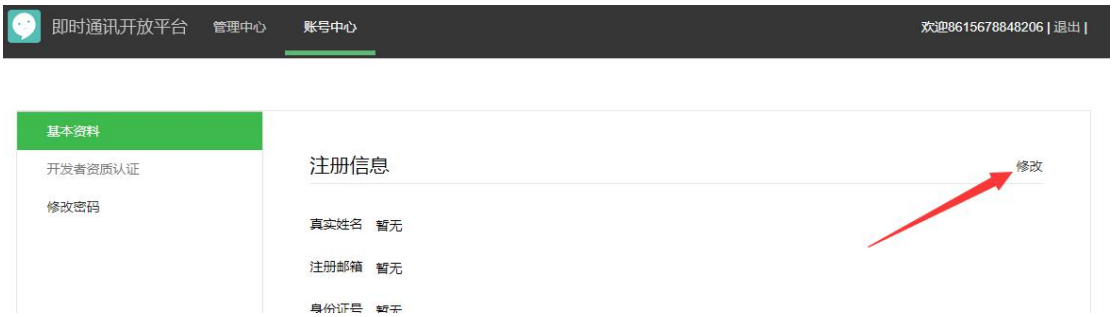
开放平台是一个支持第三方应用使用即时通讯做登录、分享的平台，总共分为两个部分，前台申请和后台审核。

注意：支付功能需要相关资质，暂时无法使用

1. 开放平台访问地址：http://imapi.lveliao.com/open/login

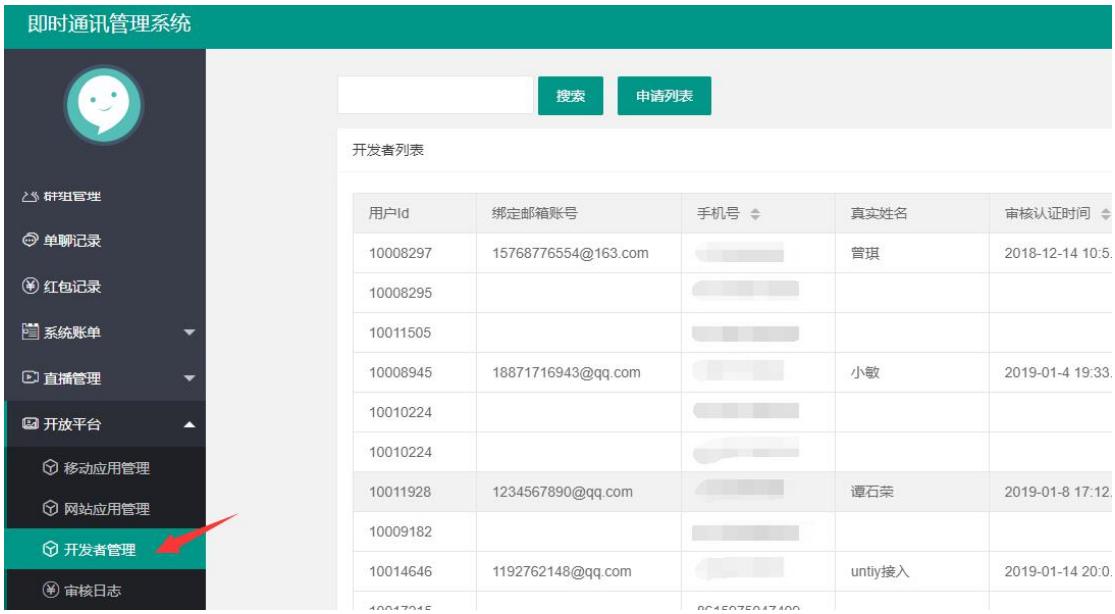
2.使用在 IM 系统中注册的账号进行登录

3.登录后，在账号中心--基本资料中点击修改完善个人信息



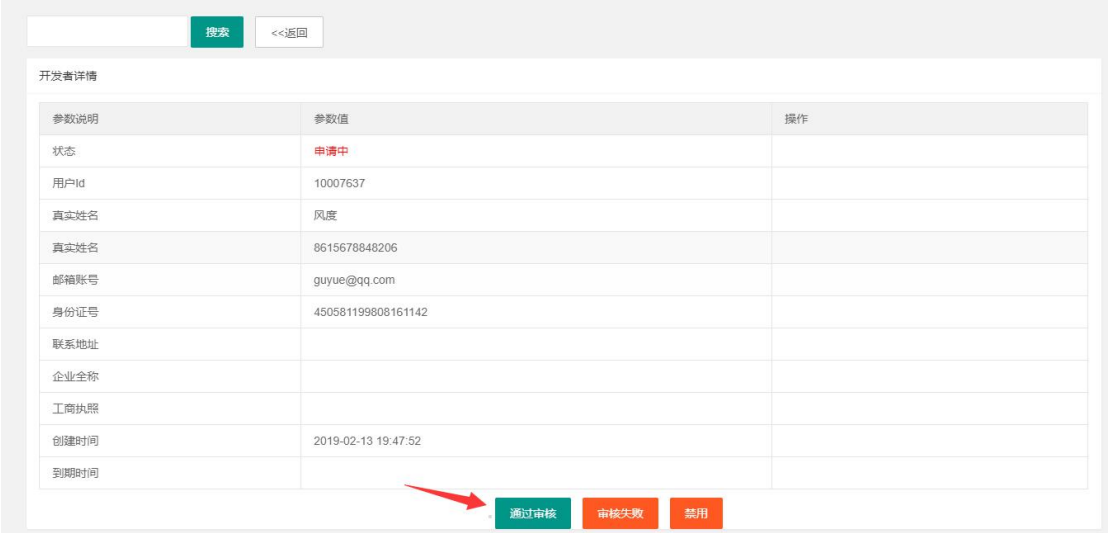
4.资料填写完毕后在开发者资质认证中点击提交申请，进行提交审核

5.此时使用管理员账号登录管理后台，在 开放平台--开发者管理中点击**申请列表**，可以看到刚刚的申请记录



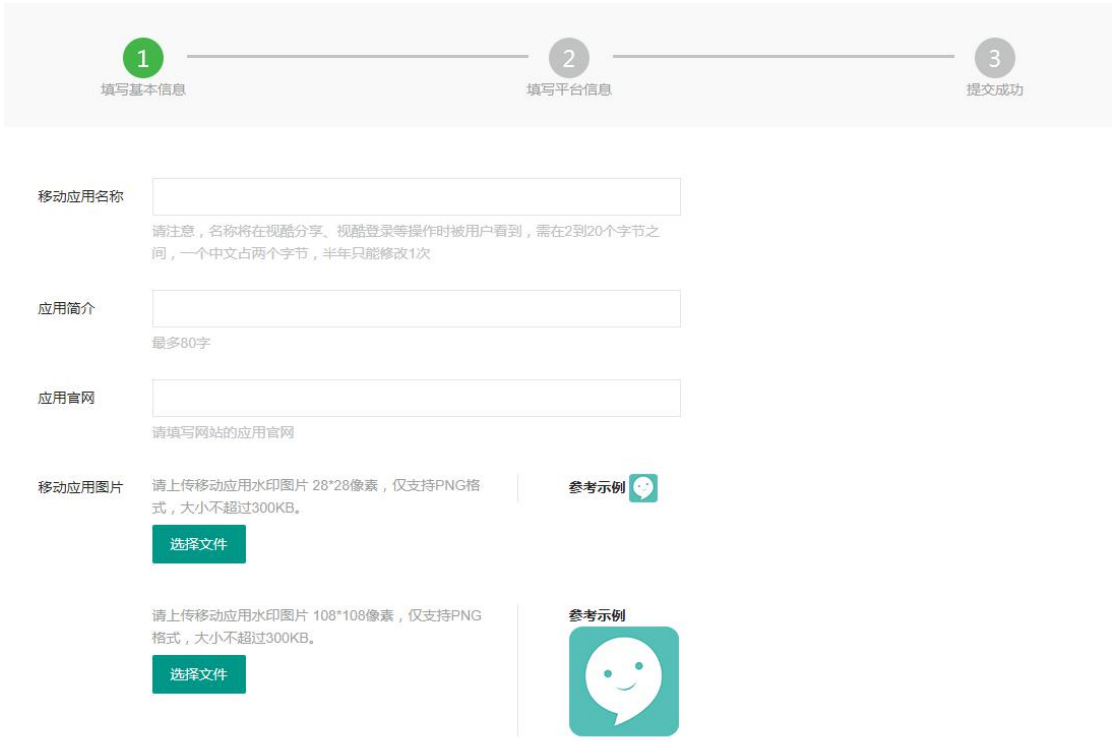
6.在申请列表列表操作项中点击详情按钮

7.在打开的详情页面可以通过下方的操作按钮进行操作，这里点击通过审核



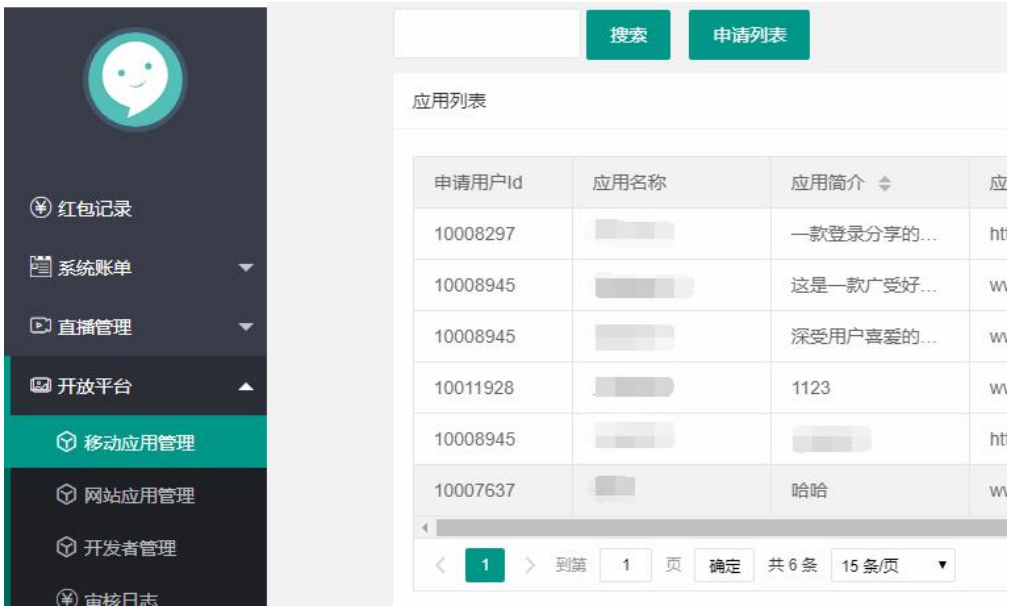
8.此时来到开放平台，可以看到验证状态为已认证

9.在通过审核后，选择创建应用类型（移动应用和网页应用），填写相关信息，请注意填写的格式要正确，最后提交到后台审核

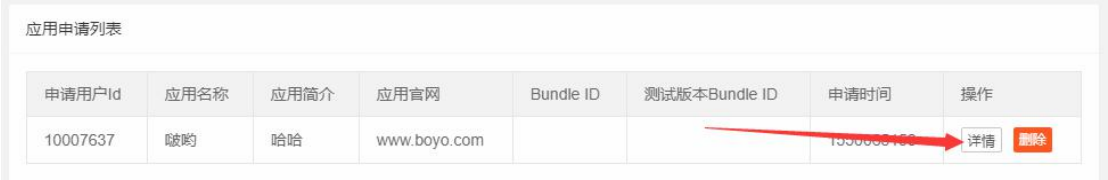


10.填写完成后提交审核

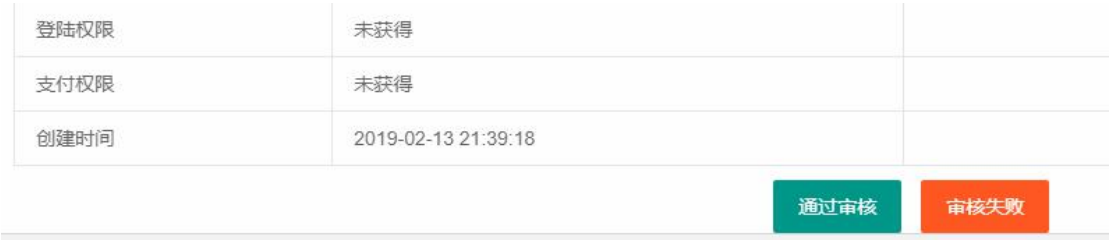
11.进入管理后台，开放平台--移动应用管理，点击申请列表



12.在对应的应用申请记录操作项中点击详情按钮



13.在打开的页面中底部有操作按钮，这里选择通过审核



14.此时在开放平台，管理中心，可以看到应用处于正常状态，点击查看按钮

15.在打开的应用详情页面可以看到应用的 AppId，和 AppSecret,在下方接口权限中选择需要使用的接口点击申请开通

16.点击申请后，在管理后台，开放平台--移动应用管理--应用详情界面 对该应用的权限进行审核，这里选择审核通过

17.接口权限审核通过后，即可进行接入使用

18.第三方应用通过得到的 appId 和 secret 请求 /open/authorization 接口得到授权信息。通过授权后根据第三应用的需要请求服务器登录、分享接口

1. 第三方应用请求接口说明

App 授权校验接口 /open/authorization

请求参数:

参数名	参数说明	是否必填
access_token	访问令牌	Y
appId	申请的 appId	Y
appSecret	申请的 app 唯一标识	Y
time	请求接口时间	Y

secret	普通接口加密规则	Y

返回值：

flag：1：校验成功　0：校验失败

2. 登录分享接口校验　/open/authInterface

请求参数：

参数名	参数说明	是否必填
access_token	访问令牌	Y
userId	用户 Id	Y
appId	申请的 appId	Y
appSecret	申请的 app 唯一标识	Y
time	请求接口时间	Y
secret	普通接口加密规则	Y
type	校验权限类型 1.登录　2.分享　3.支付	Y

返回值：

Map:{“flag”:1,”userId”:md5(userId) }

五、数据结构说明

注：MongoDb 支持以 json 串的形式保存自定义类型的数据

5.1 数据模型说明

说明：

视酷：imapi-service 项目 cn.introfuture.im.vo 包中有 Entity 注解的类为数据模型

1、如下图所示，@Entity 声明类为一个 MongoDB 集合，value 属性设置集合名；@Id 声明字段为文档的主键；@Indexed 声明字段为一个索引；

2、详细文档参考：https://github.com/mongodb/morphia/wiki/EntityAnnotation

```java
package cn.xyz.mianshi.vo;

import org.bson.types.ObjectId;

@Entity(value = "i_notice", noClassnameStored = true)
public class NoticeVO {
    private @Id ObjectId noticeId;
    private String body;
    private @Indexed int companyId;
    private int status;
    private long time;
    private String title;
    private @Indexed int userId;
```

imapi 库

1.config -----配置表

注: App 在启动的时候首先会调用 /config 接口，返回的数据就是该表中的数据

类型	字段名	字段释义	相关说明及用途
long	id	配置 id	
String	Domain	主机域名	这两处均为 tigase 所在机器的域名或 ip
String	Host	主机地址	
String	apiUrl	接口地址	
String	downloadAvatarUrl	头像访问地址	
String	downloadUrl	资源访问地址	
String	uploadUrl	资源上传地址	
String	liveUrl	直播推流地址	
~~String~~	~~freeswitch~~	~~Freeswitch 地址~~	~~用于旧版本的视频服务器，现已不用~~
String	jitsiServer	视频服务器 url	
String	meetingHost	视频会议主机 IP	
int	displayRedPacket	是否开放 IOS 红包	
int	fileValidTime	聊天内容的 文件有效期	默认 -1 -1=永久
int	chatRecordTimeOut	聊天记录过期时间	-1 为永久 数值 为天数
int	closeTelephoneFind	关闭手机号搜索用户	关闭则不用使用手机号搜索用户 0 =开启 1=关闭 默认开启
int	showContactsUser	显[illegible]	默认值为 1 0=显示 1=不显示
String	helpUrl	帮助[illegible]	

群组表[illegible]

群共享表

短信国家表

短信记录表

粉丝表

String	videoLen	录制视频时长	
String	audioLen	录制语音时长	
String	shareUrl	分享 url	
String	softUrl		
int	distance		
int	androidVersion	Android 版本号	
int	iosVersion	Ios 版本号	
String	androidAppUrl	Android App 下载地址	
String	iosAppUrl	Ios 更新地址	
String	androidExplain	Android 更新说明	
String	iosExplain	Ios 更新说明	

2.user -----用户表

类型	字段名	字段释义	相关说明及用途
int	userId	用户 Id	
String	userKey	用户唯一标识	
String	username	用户名	
String	password	密码	用户注册时由客户端将密码 MD5 加密后传到服务端
int	userType	用户类型	1=普通用户；2=公众号 ；3=微信公众号；4=客服账号
String	telephone	完整电话号码	加上区号后的，如 8615217009761
String	name	姓名	
String	nickname	昵称	
String	description	签名、说说、备注	
Long	birthday	生日	
int	sex	性别	
Loc	loc	地理位置	Loc 对象包含两个属性: double lng //经度 double lat //纬度
int	countryId	国家 Id	
int	provinceId	省份 Id	
int	cityId	城市 Id	
int	areaId	地区 Id	
int	level	等级	
int	vip	Vip 级别	
int	friendsCount	好友数量	
int	fansCount	粉丝数量	
int	attCount	关注数	
long	createTime	注册时间	

long	modifyTime	更新时间	
String	idcard	身份证号码	
String	idcardUrl	身份证图片 url	
int	isAuth	是否认证	0 ： 没有认证 1：已认证
int	status	状态	0 ： 正常
LoginLog	loginLog	用户上一次登录时的信息	
UserSettings	setting	用户设置	UserSettings 包含如下属性 int allowAtt = 1 // 允许关注，0 为不允许 int allowGreet=1 //允许打招呼，0 为不允许 int friendsVerify=1 //加好友需要验证，0 不需要
int	onlinestate	在线状态	离线 0 在线 1
String	appId	应用包名	ios 需要判断的包名
String	connectionId	连接 Id	
long	active	最后出现时间	
String	areaCode	区号	
String	phone	电话号码	不带区号
double	balance	用户余额	
double	totalRecharge	充值总金额	
int	msgNum	未读消息数量	
double	totalConsume	消费总金额	
int	offlineNoPush	消息免打扰	1 为开启 0 为关闭
int	num	创建房间次数	
int	multipleDevices	多设备登陆	0 关闭 1 开启
int	isPasuse	是否暂停	0：正常 1：暂停 （用于客服模块，暂停后不再为客服人员分配客户）

3. u_friends -----好友库 用户 Id/10000 分表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 Id	
int	blacklist	是否拉黑	1=是；0=否
int	isBeenBlack	是否被拉黑	1=是；0=否
int	offlineNoPushMsg	消息免打扰	1=是；0=否
long	createTime	建立关系时间	
long	modifyTime	修改时间	
long	lastTalkTime	最后沟通时间	
long	msgNum	未读消息数量	

int	status	状态	1=关注；2=好友；0=陌生人
String	toNickname	好友昵称	
int	toUserId	好友 Id	
int	userId	用户 Id	
String	remarkName	备注	
	~~companyId~~		
int	chatRecordTimeOut	聊天记录过期时间	默认值：-1 -1 为永久 数值为天数

4. black ----- 房间黑名单

注：目前用于记录直播房间踢出的用户

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	群组黑名单记录 id	
int	userId	被踢用户 id	
ObjectId	roomId	群组 Id	
long	time	创建时间	

5. `shiku_newFirend`-----新的朋友库 用户 Id/10000 分表

如图所示：NewFriends 表记录的数据用于 App 内新的朋友功能

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	关系 Id	
int	userId	用户 Id	
int	toUserId	好友 Id	
long	createTime	建立关系时间	
long	modifyTime	修改时间	
int	type	消息 Type	
String	content	消息内容	
int	direction	方向	0=发出去的；1=收到的

6. redPacket -----红包表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	
int	userId	发送者 userId	
String	userName	红包发送者昵称	
String	greetings	祝福语	
long	sendTime	发送时间	
int	type	红包类型	1：普通红包 2：拼手气红包 3:口令红包
int	count	红包个数	
int	receiveCount	已领取个数	
double	money	红包总金额	
double	over	红包剩余金额	
long	outTime	超时时间	
int	status	红包状态	
List<Integer>	userIds	领取者的 userId 集合	
String	roomJid	群组 jid	表示红包发送到哪个群组

7. commonText -----常用语表

注: 目前常用语用于客服模块，用于让客服人员添加常用的回复语句

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	常用语 id	
int	createUserId	创建者用户 id	
String	content	常用语内容	
long	createTime	创建时间	
int	modifyUserId	修改者用户 Id	
ObjectId	companyId	公司 id	

8. city -----城市表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	该条记录的 id 由 mongodb 自动生成
int	id	城市 id	
String	RANK	级别	
String	zh	城市名	
double	LAT	纬度	
double	LNG	经度	
String	PINYIN	城市名拼音	
int	IS_OPEN	是否开启	
String	DIVISION_STR		
String	IS_FOREIGN	是否是国外	
String	big5	城市名繁体	
int	userId	用户 Id	
long	modifyTime	修改时间	
int	status	状态	

9. company -----公司表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	公司 Id	
String	companyName	公司名称	
int	createUserId	创建者 userId	
int	deleteUserId	删除者 userId	默认值:0 当用户执行删除公司操作后，将删除者 userId 存入，不会真的删除数据。以此字段判断公司是否被“删除”
List<ObjectId>	rootDpartId	根部门 Id	可能有多个
long	createTime	创建时间	
long	deleteTime	删除时间	
String	noticeContent	公司公告内容	
long	noticeTime	公告时间	
int	empNum	公司员工总数	
int	type	类型	5：默认加入的公司

10. department -----部门表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	部门 id	
ObjectId	companyId	部门所属公司 id	
ObjectId	parentId	上一级的部门 id	
String	departName	部门名称	
int	createUserId	创建者 userId	
long	createTime	创建时间	
int	empNum	部门员工数	

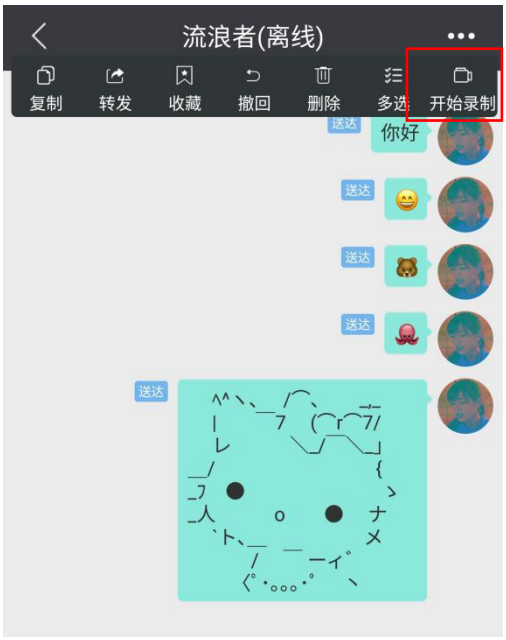
11. employee -----员工表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	员工 id	
int	userId	对应的用户的 userId	
ObjectId	departmentId	所属的部门 id	
ObjectId	companyId	所属的公司 id	
int	role	角色值	
String	position	职位	

12. ConsumeRecord -----消费记录表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	
String	tradeNo	交易单号	
int	userId	用户 Id	
double	money	金额	
long	time	时间	
int	type	类型	1：进账 2:出账
String	desc	消费备注	如：余额充值、红包发送、红包退款等
int	payType	支付方式	1：支付宝 2：微信 3：余额
int	status	交易状态	0：创建 1：支付完成 2：交易完成 -1：交易关闭

13. course -----课程表



在聊天界面，长按某条消息，点击开始录制，会从此条消息开始将录制者发送的消息进行记录，直到点击停止录制后将记录的消息保存成一个课程。

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	课程 Id	该字段和实体类 Course 的 courseId 对应
int	userId	课程所属用户 Id	
List<String>	messageIds	消息 id 集合	
long	createTime	创建时间	
long	updateTime	修改时间	
String	courseName	课程名称	
String	roomJid	房间 jid	

14. courseMessage -----课程消息表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	课程消息 Id	
int	userId	用户 Id	
String	courseId	课程 Id	
long	createTime	创建时间	
String	message		

15. customer -----访客表

注：用于客服模块记录访客

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	
int	customerId	访客 Id	相当于用户 userId
String	userKey	访客标识	userKey 由 ip 地址 MD5 加密得到
String	ip	IP 地址	
~~String~~	~~macAddress~~	~~Mac 地址~~	暂时没有用到
long	createTime	注册时间	
String	companyId	公司 id	

16. emjor

类型	字段名	字段释义	相关说明及用途
ObjectId	_id		
int	userId		
String	url		
long	createTime		
long	index		

17. emoji ----- 收藏表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	收藏 id	对应 Emoji 实体类的 emojiId
int	userId	所属用户 userId	
int	type	收藏类型	1.图片 2.视频 3.文件 4.语音 5.文本 6.表情
String	url	图片、视频等的 url	
long	createTime	创建时间	
String	msgId	对应的那条消息的 Id	
String	msg	对应的那条消息	
String	roomJid	群组 jid	

18. friendGroup -----好友分组

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	分组 id	对应实体 FriendGroup 里的 groupId

String	groupName	分组名称	
int	userId	分组所属的用户 userId	
long	createTime	分组创建时间	
List<Integer>	userIdList	分组中好友的 userId 集合	用于记录将哪些好友放在该分组中

19. givegift -----赠送礼物表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	礼物 Id	对应实体类 Givegift 中的 giftId
int	count	礼物数量	
int	id	送礼物记录 Id	
double	price	礼物价格	
long	time	送礼时间	
int	userId	送礼物用户 Id	
int	toUserId	接收礼物用户 Id	

20. idx_user -----自增长计数表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 Id	
int	id	userId 当前值	记录当前的 userId 最大值，用于 userd 最大值
String	stub		
int	call	语音会议标识当前值	记录语音会议标识当前值，用于语音会议标识的自增长
int	videoMeetingNo	视频会议标识当前值	记录视频会议标识当前值，用于视频会议标识的自增长

21. LiveRoom ----- 直播房间

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	直播间 Id	由 mongodb 自动生成，对应实体类 LiveRoom 中的 roomId
int	userId	直播间创建者 userId	
String	nickName	创建者昵称	

String	name	直播间名称	
String	notice	直播间公告	
String	url	直播间推流地址	
String	img	房间封面 url	
int	numbers	直播间人数	
int	status	直播状态	1:开启直播 0:关闭直播
long	createTime	创建时间	
String	jid	直播间 jid	

22. LiveRomMember -----直播房间成员表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 Id	
ObjectId	roomId	直播间 id	
int	userId	对应的 userId	
String	nickName	成员昵称	
int	type	类型	1 为创建者 2 为管理员 3 为成员
int	state	状态	为了判断是否被禁言 1 为禁言，0 为未禁言
int	number	送礼物的数量	
long	createTime	创建时间	
int	online	是否在线	1 为在线 0 为退出

23. message -----错误信息表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 Id	由 mongodb 自动生成
String	code	错误码	
String	zh	错误信息中文	
String	en	错误信息英文	
String	type	类型	
String	big5	错误信息繁体	

24. mp_menu -----公众号菜单表

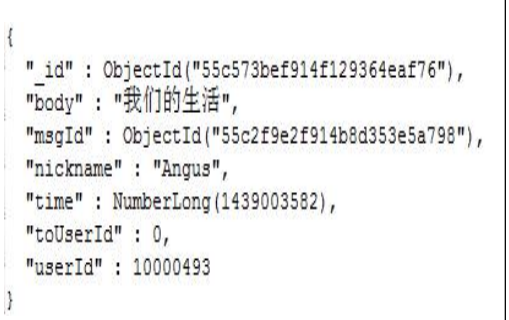
类型	字段名	字段释义	相关说明及用途
long	_id	菜单 Id	

long	parentId	菜单父级 Id	菜单数据示例： 
int	userId	创建菜单的用户 userId	
String	name	菜单名称	
String	url	菜单连接的 url	
int	index	编号	
String	menuId	用来标识某一个接口	
String	desc	描述	

25. notKeyword -----敏感词

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	敏感词 Id	
String	word	敏感词	

26. s_comment -----朋友圈评论表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	由 mongodb 自动生成
String	body	评论内容	记录示例： 
ObjectId	msgId	评论所属消息Id	
String	nickname	评论用户昵称	
long	time	评论时间	
int	toUserId	被回复用户Id	
int	userId	评论用户 Id	
~~String~~	~~toBody~~	~~被回复内容~~	
String	toNickname	被回复人用户昵称	

27. s_gift -----礼物表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	礼物 id	

String	name	礼物名称	礼物数据示例： 
String	photo	礼物图片 url	
double	price	礼物单价	
int	type	礼物类型	

28. s_msg -----朋友圈消息

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	朋友圈消息 id	
Body	body	消息内容对象	Body **对象包含的属性：** String address //发送朋友圈的地址 List<Resource> audios //音频列表 List<Resource> images //图片列表 List<Resource> videos //视频列表 String remark //备注 String text //文本 Long time //发送时间 String title //朋友圈消息标题 ---Resource 对象包含的属性： String oUrl //原图 url String tUrl //缩略图 url Long size //视频、语音的大小 Int length //播放时长(秒)
int	cityId	消息所属城市 Id	
Count	count	计数对象	注：Count 对象包含的属性： Long collect //收藏数 Long comment //评论数 Long forward //转发数（App 内） Long money //金币数 Long play //播放数 Long praise //点赞数 Long share //分享数（App 外） Long total // 参考数
int	flag	标记	~~1=求职消息、2=招聘消息~~

			3=普通消息 （1、2 暂时没有用到）
double	latitude	纬度	
double	longitude	经度	
String	nickname	昵称	
long	time	发朋友圈消息时间	
int	userId	发朋友圈消息用户 id	
int	visible	是否可见 默认值 1	1：公开，2：私密，3：部分可见选中的朋友的可见，4：不给谁看，5：@谁看
String	model	手机型号	如：iPhone 7、HWI-TL00 等
String	location	地址信息	
List<Integer>	userRemindLook	选中不可见的朋友的 userId 集合	
List<Integer>	userLook	选中可见的朋友 userId 集合	
List<Integer>	userNotLook	@提醒的朋友 userId 集合	

29. s_praise ---- 朋友圈点赞表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	赞的 id	对应 Praise 实体类的 praiseId
ObjectId	msgId	赞所属的朋友圈 消息 id	
String	nickname	点赞用户的昵称	
long	time	点赞时间	
int	userId	点赞用户 userId	

30. shildedFriend

类型	字段名	字段释义	相关说明及用途
ObjectId	_id		
int	userId		
int	toUserId		

31. sms_country ----- 短信国家表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	由 mongodb 自动生成
String	en	英文名	
String	zh	中文名	
String	prefix	国家区号	
double	price	短信价格	
String	big5	繁体名	
int	userId	创建者 id	0：系统
long	modifyTime	修改时间	
int	status	状态	状态 0 or null 正常 公用 -1 伪删除 1 ：个人生效 2：临时状态
int	id	国家 id	

32. SmsRecord -----短信记录表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	短信记录 id	
String	areaCode	区号	
String	telephone	电话号码	
String	code	验证码	
String	country	国家	
String	content	短信内容	
double	price	短信价格	
long	time	发送时间	
String	msgId	短信 id	这里是第三方短信平台的 id

33. tb_areas -----地区表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 id	由 mongodb 自动生成
int	parent_id	父级 id	
int	type	类型	
String	zh	中文名称	
String	zip	邮政编码	
double	LAT	纬度	
double	LNG	经度	
String	PINYIN	拼音名称	中国的市、区、县一级

String	code	国家代码	该值只有国家一级的具有，其它的为 null
String	ab	所属国家英文简称	
String	en	英文名称	
String	big5	繁体名称	
int	userId	创建者 userId	0：系统
int	modifyTime	修改时间	
int	status	状态	状态 0 or null 正常 公用 -1 伪删除 1 ：个人生效 2：临时状态
int	id	地区 id	

34. tb_constants -----常量表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 Id	
int	id	常量 Id	
int	parent_id	父级常量 Id	
String	zh	常量名中文	
int	more		
String	en	常量名英文	
String	big5	常量名繁体	
long	userId	创建者 Id	0：系统
long	modifyTime	修改时间	
int	status	状态	0 正常 公用 -1 伪删除 1 ：个人生效 2：临时状态

群组及消息库

1. shiku_msgs -----单聊消息库 按照用户 Id 分库

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	消息记录 id	由 mongodb 自动生成
String	message	消息体	Json 串格式 示例：{"content":"4","deleteTime":-1,"fileSize":0,"fileTime":0,"fromUserId":"10005629","fromUserName":"5778","isEncrypt":false,"isReadDel":false,"location_x":0.0,"location_y":0.0,"messageHead":{"

			chatType":1,"from":"10005629/android","messageId":"bf3355cc799743fba54d0d640760f22e","offline":false,"to":"10000"},"timeSend":1564737521378,"toUserId":"10000","toUserName":"客服公众号","type":1}
int	direction	消息方向	0=发出的；1=收到的
long	receiver	接收者 userId	
String	receiver_jid	接收者 jid	10008297/ios
long	sender	接收者 userId	
String	sender_jid	接收者 jid	
long	ts	时间	毫秒，当前使用的时间字段
int	type		
int	contentType	消息类型	
String	messageId	消息 id	消息 id 由客户端发送消息时通过 uuId 产生
double	timeSend	发送时间	毫秒
String	context	消息内容	

imRoom 库

1. shiku_room -----群组表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	群组 id	
String	jid	群组 jid	
String	name	群组名称	
String	desc	群组描述	
String	subject	群组主题	
int	category	群组分类	
List<String>	tags	群组标签	
String	call	语音通话标识	
String	videoMeetingNo	视频聊天标识	
int	showRead	是否显示已读人数	群主设置 群内消息是否发送已读 回执 用于显示已读人数
int	userSize	当前成员数	
int	maxUserSize	最大成员数	默认值： 1000
Member	member	自己	

List<Member>	members	成员列表	
int	countryId	国家 id	
int	provinceId	省份 id	
int	cityId	城市 Id	
int	areaId	地区 id	
double	longitude	经度	
double	latitude	纬度	
int	userId	创 建 者 userd	
String	nickname	创建者昵称	
long	createTime	创建时间	
int	modifier	修 改 者 userId	
long	modifyTime	修改时间	
int	s	状态	1:正常
Notice	notice	公告	"notice" : { "roomId" : ObjectId("59ba72c94adfdc3d64d8fb7f"), "text" : "No notice", "userId" : 10004761, "nickname" : "0003", "time" : NumberLong(1505444179), "_id" : ObjectId("59bb41534adfdc1d3a0ad884") }
int	isLook	是否可见	0 为可见 1 为不可见
int	isNeedVerify	加群是否需要通过验证	0：不要 1：要
int	showMember	显示群成员给普通用户看	1 显示 0 不显示
int	allowSendCard	允许发送名片好友	1 允许 0 不允许
int	allowHostUpdate	是否允许群主修改 群属性	1 允许 0 不允许

2.mucmsg_** ----- 群聊消息记录表

注：这里 ** 表示群的 jid，每个群对应一个消息表

类型	字段名	字段释义	相关说明及用途

ObjectId	_id	记录 id	由 mongoDB 自动生成
String	message	消息体	格式为 json 串
int	event_type		
String	message	完整的消息	xml 格式
~~String~~	~~nickname~~	~~发送者昵称~~	~~当前没有用到~~
int	public_event		
String	room_jid	群组 jid	
String	sender_jid	发送者 jid	100052/ios
long	sender	发送者 userId	
long	ts	消息发送时间	毫秒 当前版本使用
int	contentType	消息类型	如：1=文本 2：语音 详细类型请参照 **九/消息类型介绍**
String	messageId	消息 id	客户端在发送消息时通过 uuid 产生
double	timeSend	旧版消息发送时间	毫秒
String	context	消息内容	

3. shiku_room_member ----- 群组成员库 按照用户 Id 分表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	成员记录 id	
Objectd	roomId	群组 id	
int	userId	成员 userId	
String	nickname	成员昵称	
int	role	成员角色	1=创建者、2=管理员、3=成员
int	sub	订阅群信息	0=否、1=是
int	offlineNoPushMsg	消息免打扰	1=是；0=否
long	talkTime	会话时间	大于当前时间时禁止发言
long	active	最后一次互动时间	
long	createTime	创建时间	
long	modifyTime	修改时间	
String	call	语音通话标识符	
String	videoMeetingNo	视频会议标识符	

4. shiku_room_notice -----群公告表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	公告 Id	
ObjectId	roomId	群组 id	
String	text	公告内容	
int	userId	公告发布者 userId	
String	nickname	发布者昵称	
long	time	发布时间	

5. shiku_room_share ------群共享表

类型	字段名	字段释义	相关说明及用途
ObjectId	_id	记录 Id	
ObjectId	roomId	群组 Id	
String	name	文件名称	
String	url	文件 url	
long	time	发送时间	
int	userId	发消息的用户 id	
String	nickname	发送者昵称	
int	type	文件类型	
float	size	文件大小	

六、项目中集成的第三方服务

1. 发送短信服务

说明：接入第三方短信服务，主要用来给在用户注册、修改密码时给用户发送短信验证码。

项目中目前集成了天天国际短信，和阿里云短信,选择其中一家注册配置即可。

1) **阿里云短信：** https://www.aliyun.com/product/sms

请参照如下文档说明在阿里云短信控制台进行配置

流程指引文档：
https://help.aliyun.com/document_detail/59210.html?spm=5176.sms-account.0.0.76291cbeYoeYvk

申请好后将配置写入到 imapi 的配置文件中

```
## 阿里云短信服务
im.smsConfig.product=Dysmsapi
im.smsConfig.domain=dysmsapi.aliyuncs.com
im.smsConfig.accesskeyid=
im.smsConfig.accesskeysecret=
im.smsConfig.signname=视酷IM
im.smsConfig.chinase_templetecode=
im.smsConfig.english_templetecode=
```

2)**天天国际短信：** 网址 ： m.isms360.com
如需集成其它短信在 KSMSServiceImpl 类中修改即可

注：openSMS = 0:关闭短信 1：开启短信

```
## SMS Properties
im.smsConfig.openSMS=0
im.smsConfig.host=m.isms360.com
im.smsConfig.port=8085
im.smsConfig.api=/mt/MT3.ashx
im.smsConfig.username=username
im.smsConfig.password=password
```

图 6 短信配置

注： 如使用默认集成的天天国际短信，请在该平台注册账号 http://m.isms360.com
然后联系平台客服，设置短信模板如： 【视酷 IM】,您的验证码为:33333
并让客服开通，以便正常使用。

所在类：KSMSServiceImpl

```
private String sendSmsToMs360(String telephone,String areaCode,String code){
    String ip = smsConfig.getHost();
        int port = smsConfig.getPort();
        // HTTP 请求工具
        //"/mt/MT3.ashx"
        HttpClientUtil util = new HttpClientUtil(ip,  port,smsConfig.getApi());
        String user = smsConfig.getUsername();//你的用户名
        String pwd = smsConfig.getPassword();//你的密码：
        String ServiceID = "SEND"; //固定，不需要改变
        String dest = telephone; // 你的目的号码【收短信的电话号码】
        String sender = "";// 你的原号码,可空【大部分国家原号码带不过去，只有少数国家支持透传，所有一般为空】
        String msg ="[SHIKU IM], Your verification code is:"+code;//你的短信内容
        if("86".equals(areaCode)||"886".equals(areaCode)||"852".equals(areaCode))
         msg = "[视酷IM],您的验证码为:"+code;
        // codec=8 Unicode 编码, 3 ISO-8859-1, 0 ASCII
      // 短信内容 HEX 编码，8 为 UTF-16BE HEX 编码， dataCoding = 8 ,支持所有国家的语言，建议直接使用 8
        String hex = WebNetEncode.encodeHexStr(8, msg);
        hex = hex.trim() + "&codec=8";
        //System.out.println("POST MT3");
        // HTTP 封包请求，util.sendPostMessage  返回结果，
      // 如果是以 “-” 开头的为发送失败，请查看错误代码，否则为MSGID

        String msgId=util.sendGetMessage(user, pwd, ServiceID, dest, sender, hex);

        System.out.println("msgid =  "+msgId);
      //发送短信记录保存到数据库
        saveSMSToDB(telephone, areaCode, code, msg, msgId);

        return msgId;
}
```

图 7 调用天天国际短信

2. 消息推送服务

1) 百度云推送主要用于 Ios

1. 首先在百度云推送平台注册账号　　链接：http://push.baidu.com/console/app

2. 点击创建新应用，在输入框中填写名称（这里的名称仅用于区分）

3. 选择类型（Android /IOS ），并填写相关信息

注意：百度推送需要为 Android 和 IOS 分别创建一个应用

Android 填写应用的包名即可

IOS 需要上传推送证书

4.申请完成后分别将得到的 Android 和 IOS 的 ApiKey 和 SecretKey 配置到代码中

所在类：BaiduPushService

注：需要自行注册百度开发者平台账号，获取相应配置替换到代码中。
百度云推送网址：http://push.baidu.com/ 在百度云推送页面中申请安卓和 iOS 应用得到两个 app_key 和 secret_key

```
private Aps aps;
private String description;// 内容
private String title;// 标题

private String name;
private int status;// 状态：0=未读；1=已读
private long time;
private String toName;
private int type;//消息类型
private long toUserId; // 用户Id（个人用户或企业用户）
private long userId;// 用户Id（个人用户或企业用户）
```

图 10 推送消息属性

```
// EMPTY，安卓，iOS
// public static final String[] APP_ID = new String[] { "", "7134076",
// "7134076" };
public static final String APPSTORE_APPID="com.shiku.im.push";
public static final String APPSTORE_APPKEY="YWCjFscGk7cv3RlEtaxoypzt0sipp6vw";
public static final String APPSTORE_SECRET_KEY="Y5tsqtPYDtTHFC1NjkccEYqyLVZ9jGnh";
public static final String[] APP_KEY = new String[] { "", "8h0GOjOlgP8dXRzp9nG1dGBT", "7LlWDe0AZGKILS4Tq5cMNMum" };
public static final String REST_URL = "http://api.tuisong.baidu.com";
public static final String[] SECRET_KEY = { "", "348afe4f67d9053b604f8a38d1cfa98b",
      "7fd2005f78131ba9b7affb454c221847" };
```

图 11 配置

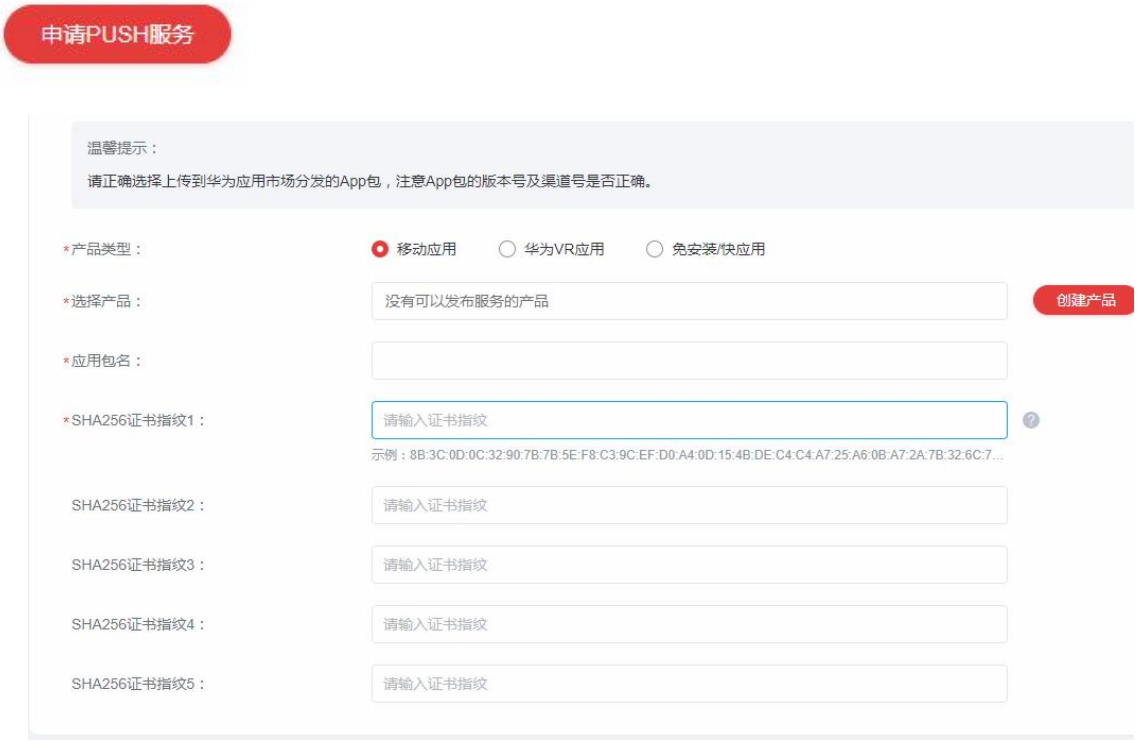
2) 华为推送

1. 首先在华为开发者平台注册账号：链接 https://developer.huawei.com/consumer/cn/

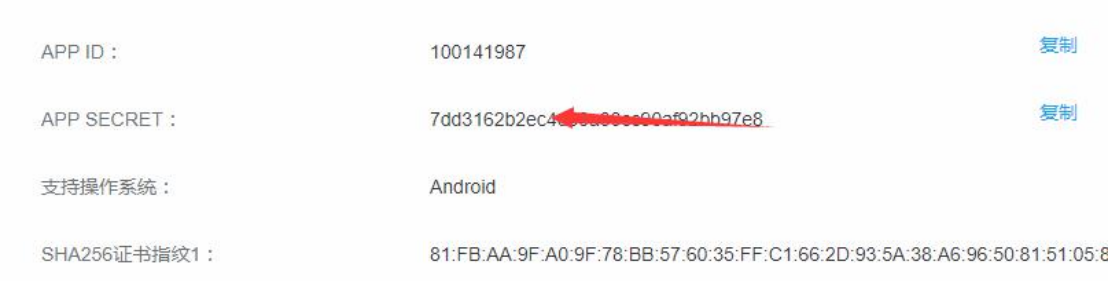
然后进行身份认证，认证成功后在开发服务中点击进入 push 推送服务

注：如只需使用推送服务可以用个人身份进行认证，如果后续需要在该平台上线建议使用企业身份进行认证

2. 点击申请 PUSH 服务，按提示填写相关资料



3.信息填写完毕后将获取到的 appId 和 App Secret 配置到代码中即可



所在类：HWPushService && HWPushTransService

```
//华为推送集成通知栏消息
public class HWPushService {
     private static String appSecret = "7dd3162b2ec4d39a30cc90af92bb97e8";
     private static  String appId = "100141987";
     private static  String tokenUrl = "https://login.cloud.huawei.com/oauth2/v2/token";
     private static  String apiUrl = "https://api.push.hicloud.com/pushsend.do";
     private static  String accessToken;
     private static  long tokenExpiredTime;

     /*public static void main(String[] args) throws IOException{
           sendPushMessage("0865217039424357300001122700CN01");
     }*/
```

华为通知栏推送

```
//华为透传消息推送
public class HWPushTransService {
     private static String appSecret = "7dd3162b2ec4d39a30cc90af92bb97e8";
        private static  String appId = "100141987";//用户在华为开发者联盟申请的appId和appSecret（会员中心->我的
        private static  String tokenUrl = "https://login.cloud.huawei.com/oauth2/v2/token";//获取认证To
        private static  String apiUrl = "https://api.push.hicloud.com/pushsend.do"; //应用级消息下发API
        private static  String accessToken;//下发通知消息的认证Token
        private static  long tokenExpiredTime;  //accessToken的过期时间

       /* public static void main(String[] args) throws IOException
        {
            sendPushMessage("0865217039424357300001122700CN01");
        }*/
```

图 12 华为透传消息

3) 小米推送

1. 首先进入小米开放平台注册账号　链接：https://dev.mi.com/console/man/

注：如只需使用推送服务可以用个人身份进行认证，如果后续需要在该平台上线建议使用企业身份进行认证

2. 认证成功后，在消息推送中创建应用，并填写对应的数据

3. 应用创建成功后，将产生的 AppKey AppSecret 配置到代码中即可

注：确保推送处于已启用状态

应用名称	平台类型	启用状态
视酷即时通讯		已启用

所在类：XMPushService && XMPushTransService

```
//小米通知栏推送集成
public class XMPushService {

 public static Sender sender=new Sender("wetU6ay+22KQ6H8qF4AFiw==");//申请到的AppSecret

 public final static String PACKAGE_NAME="com.sk.weichat";//app的包名

  public static void pushToRegId(MsgNotice notice,String callNum){
      if(StringUtils.isEmpty(notice.getText()))
          notice.setText("收到一条消息...");
```

微信支付回调接口在　ConsumeRecordController　类中 /user/recharge/wxPayCallBack

```java
@RequestMapping(value="/user/recharge/wxPayCallBack",method=RequestMethod.POST)
public void wxPayCallBack(HttpServletRequest request,
        HttpServletResponse response) throws IOException {
    //把如下代码贴到的你的处理回调的servlet 或者.do 中即可明白回调操作
    logger.info("微信支付回调数据开始");
    BufferedOutputStream out = null;
    String inputLine;
    String notityXml = "";
    String resXml = "";
    try {
        while ((inputLine = request.getReader().readLine()) != null) {
            notityXml += inputLine;
        }
        request.getReader().close();

        Map<String,String> m = WXNotify.parseXmlToList2(notityXml);
        logger.info("接收到的报文：" + m);
            String tradeNo=m.get("out_trade_no");
            ConsumeRecord entity=service.getConsumeRecordByNo(tradeNo);
            if(null==entity)
                logger.info("交易订单号不存在！-----"+tradeNo);
            else if(0!=entity.getStatus())
                logger.info(tradeNo+"===status==="+entity.getStatus()+"=======交易已处理或已取消！");
            else if("SUCCESS".equals(m.get("result_code"))){
                boolean flag=Double.valueOf(m.get("cash_fee"))==entity.getMoney()*100;
                if(flag){
                     //logger.info("支付金额比较"+m.get("cash_fee")+"=="+entity.getMoney()*100+"===
                    WxPayResult wpr = WXPayUtil.mapToWxPayResult(m);
                    //支付成功
                    resXml = "<xml>" + "<return_code><![CDATA[SUCCESS]]></return_code>"
                    + "<return_msg><![CDATA[OK]]></return_msg>" + "</xml> ";
                    entity.setStatus(KConstants.OrderStatus.END);
```

七、消息类型介绍

Type 值	所代表的消息类型
0	不显示的无用类型
1	文本
2	图片
3	语音
4	位置
5	动画表情（Gif）
6	视频
7	音频

8	名片
9	文件
10	自己添加的消息类型,代表系统的提示
26	已读回执
28	红包
80	单条图文
81	多条图文
82	链接
83	领取红包
84	戳一戳
85	合并转发
100	询问是否准备好语音通话
101	已准备好语音通话
102	接受语音通话
103	拒绝语音通话 或 取消拨号
104	结束语音通话
110	询问是否准备好视频通话
111	已准备好视频通话
112	接受视频通话
113	拒绝视频通话 或 取消拨号
114	结束视频通话
115	邀请加入视频会议
116	加入视频会议
117	退出视频会议
118	踢出视频会议
120	邀请加入语音会议
121	加入语音会议
122	退出语音会议
123	踢出语音会议
124	轮麦
125	取消轮麦
200	多点登录验证在线

201	正在输入消息
202	消息撤回
301	朋友圈点赞
302	朋友圈评论
27	通知评论消息
304	朋友圈提醒谁看
401	上传群文件
402	删除群文件
403	下载群文件
500	打招呼
501	同意加好友
502	回话
503	新关注
504	删除关注
505	彻底删除
506	新推荐好友
507	黑名单
508	直接成为好友
509	取消黑名单
901	修改昵称
902	改房间名
903	删除房间
904	删除成员
905	新公告
906	禁言
907	增加新成员
915	设置群已读消息 只有接收
910	发送弹幕
911	发送礼物
912	发送爱心
913	设置管理员

914	加入直播间

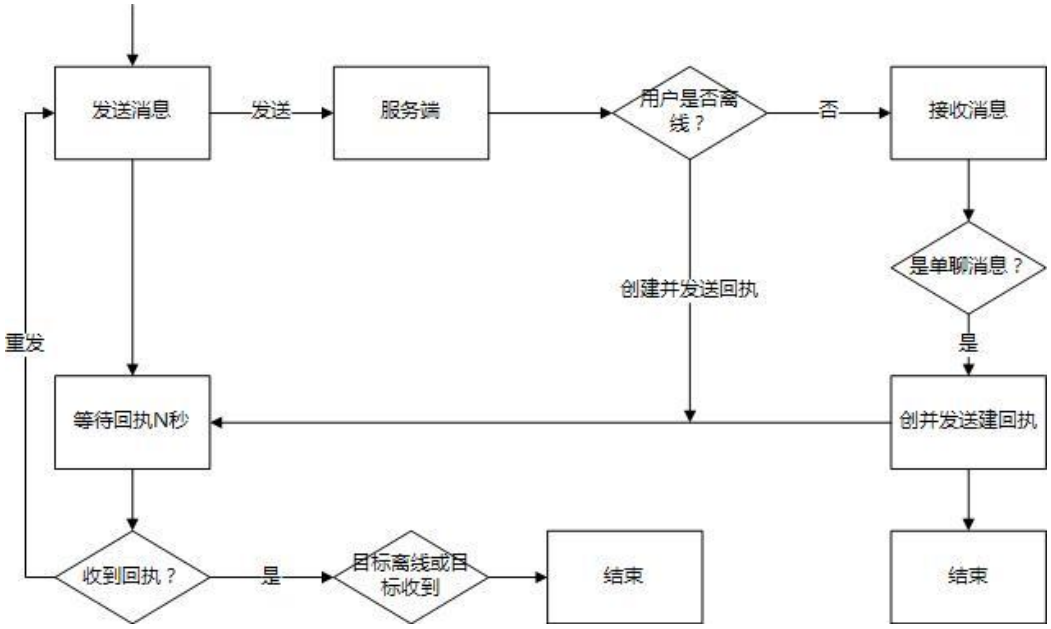
八、消息回执

1.服务端收到一条消息 回给客户端发送一条消息回执

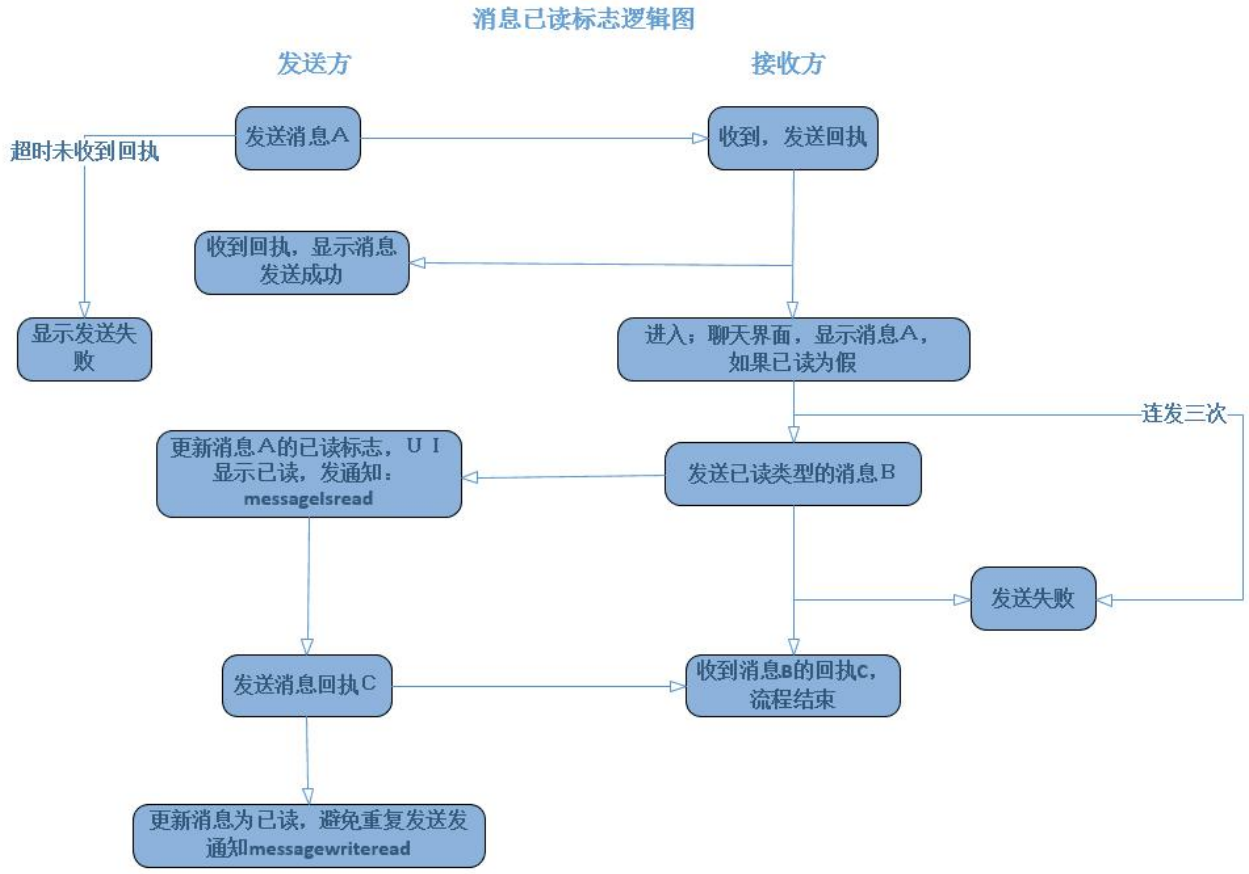
2.客户端缓存一个待发送消息列表，如果收到回执，即从列表删除这一条。如果在 N 秒内迟迟收不到回执，则重发。秒数可以根据消息类型自定义长短，如文字建议为 15 秒，大图片建议为 45 秒。

接收方下登录时通过获取离线消息方式获取离线期间未收到的消息。接收方在线情况下，发送方每发送一条消息，接收方接到消息后马上发送回执给发送方，发送方在 N 秒内未收到回执则认为消息发送失败，客户线需要重新发送或提示用户消息发送失败。

单聊回执流程图



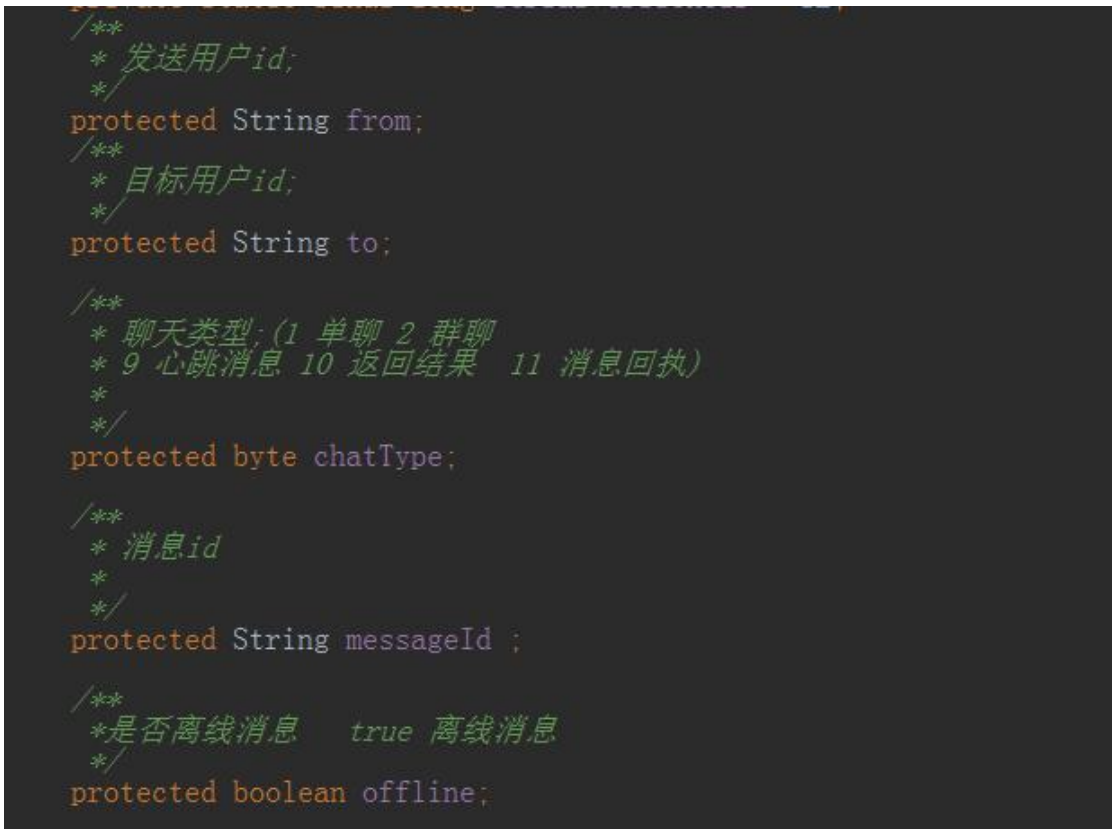
九、已读标志逻辑



十通讯模块开发说明

1.消息协议说明

MessageHead 消息协议头
所有的聊天都包含一个 MessageHead
MessageHead 的参数说明



```java
/**
 * 发送用户id;
 */
protected String from;
/**
 * 目标用户id;
 */
protected String to;

/**
 * 聊天类型;(1 单聊 2 群聊
 * 9 心跳消息 10 返回结果  11 消息回执)
 *
 */
protected byte chatType;

/**
 * 消息id
 *
 */
protected String messageId ;

/**
 *是否离线消息   true 离线消息
 */
protected boolean offline;
```

ChatMessage 参数说明

```java
/**
 * 业务发送者
 */
private String fromUserId;
/**
 *发送者用户名
 */
private String fromUserName;
/**
 * 业务接受者
 */
private String toUserId;
/**
 * 接受者用户名
 */
private String toUserName;

/**
 * 消息类型;
 *
 * 具体参考 xmpp协议文档
 */
private short type;

/**
 *
 * 消息发送时间  毫秒单位
 */
protected long timeSend;

/**
 * 消息内容;
 */
private String content;
```

```java
/**
 * 多用于 消息发到哪个群组 jid;
 * 具体参考 xmpp协议文档
 */
private String objectId;

/**
 * 多用于 文件名称
 * 具体参考 xmpp协议文档
 */
private String fileName;

// 是否加密传输
@JSONField(name="isEncrypt")
private boolean isEncrypt;

// 消息到期时间(当前时间+消息保存天数=到期时间)
private long deleteTime;
@JSONField(name="isReadDel")
private boolean isReadDel;  // 是否阅后即焚

//  文件大小 单位字节
private long fileSize;
// 文件播放时长  录音时长，视频时长
private long fileTime;

// 1.当为地理位置时，有效 2.特殊：当为图片时，该值为图片的宽度
private double location_x;
// 1.当为地理位置时，有效 2.特殊：当为图片时，该值为图片的高度
private double location_y;
```

2. 协议核心类及方法说明

IMClient 发送消息链接对象

```
/**
 *
 * @param ip    IM 服务器 Ip
 * @param port IM 服务器端口
 * @param clientHandler 客户端消息处理监听
 * @param clientListener 客户端事件处理监听
 */
public void initIMClient(String ip,int port,BaseClientHandler
clientHandler,BaseClientListener clientListener)
```

发送消息方法

public void sendMessage(AbstractMessage message)

BaseClientHandler 消息监听对象

方法说明

```
/**
 * 收到服务器的消息回执   代表消息已经到达服务器
 * @param messageId
 */
public   void handlerReceipt(String messageId)

/**
 * 收到服务器的 消息
```

```
 * 需要处理自己的逻
 * 重写这个方法
 * @param message
 */
public    void handlerMessage(AbstractMessage message)
```

```
/**
 * 收到登陆结果
 * @param message
 */

public void handlerLoginResult(AuthRespMessage message)
```

下图为登陆结果参数说明

```
/**
 * 登陆结果  1 登陆 成功     0 登陆失败
 */
private byte status;


/*
 * 提示信息
 */
private String arg;

private String token;

/**
 * 在线设备列表  android,ios,web
 */
private String resources;
```

BaseClientListener　链接事件监听对象

方法说明

与服务器建立链接成功
public void onAfterConnected
发送消息回调事件
public void onAfterSent
断开链接事件

public void onBeforeClose

发送消息流程说明

参考 com.shiku.push.rocketmq.SocketImTest

1.初始化链接对象

```
        /**
 * 创建 链接对象
 */
IMClient client=new IMClient();
/**
 * 赋值用户 Id
 */
client.setUserId(userId);
/*
 * 赋值用户 Resource
 * 服务端系统登陆统一使用  Server
 * */
client.setResource("Server");
/*
 * 赋值 发送心跳 时间 毫秒
 * */
client.setPingTime(60000);
BaseClientHandler clientHandler=new BaseClientHandler() {
    @Override
    public void handlerReceipt(String messageId) {
    System.out.println("handlerReceipt ===> "+messageId);
    /*
     * 收到服务端的消息回执
     * 代表消息已经到达服务器
     * */

    }

    @Override
    public void handlerMessage(AbstractMessage message) {
        super.handlerMessage(message);
        /*收到消息 处理自己的业务逻辑 */
    }
};
BaseClientListener clientListener=new BaseClientListener() {
    @Override
    public AuthMessage authUserMessage(ChannelContext channelContext, BaseIMClient client) {
```

```
            /*
             * 建立链接成功后,会调用这个方法
             * 生成一个登陆授权的协议发送给服务端
             **/
            MessageHead messageHead=new MessageHead();
            messageHead.setChatType((byte)1);
            channelContext.userid=userId;
            messageHead.setFrom(userId+"/Server");
            AuthMessage authMessage=new AuthMessage();
            //authMessage.setPassword();
            /*
             * 服务器 登陆 token  统一使用 配置文件
             * 获取的 Token 和 IMServer 配置一致
             * 客户端 登陆 Token 为 登陆接口返回的 访问令牌
             **/
            authMessage.setToken("token");
            authMessage.setMessageHead(messageHead);
            return authMessage;
        }
    };
    /*初始化链接对象,会自动发送登陆协议*/
    client.initIMClient("localhost",5666,clientHandler,clientListener);
```

2.发送消息

```
MessageHead messageHead=new MessageHead();
        /*
        * client.getConnStr() = 10005/Server
        * */
        messageHead.setFrom(client.getConnStr());
        messageHead.setTo("100020");
        /*发送单聊消息*/
        messageHead.setChatType(ChatType.CHAT);
        messageHead.setMessageId(StringUtil.randomUUID());

        ChatMessage message=new ChatMessage();
        message.setContent("你好 ！ ");
        message.setFromUserId(client.getUserId());
        message.setFromUserName("服务器管理员");
```

```
message.setToUserId("100020");
message.setToUserName("100020 用户");

/*文本消息*/
message.setType((short) 1);
message.setTimeSend(System.currentTimeMillis());

message.setEncrypt(false);
message.setReadDel(false);
client.sendMessage(message);
```