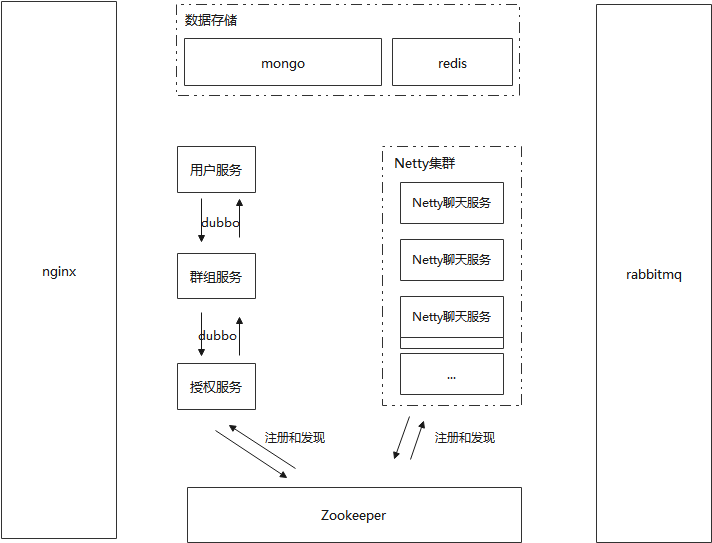
# 文件上传平台

## 项目背景

一个基于 Netty 和 Dubbo 的分布式聊天项目，支持用户授权验证、群聊、私聊的IM系统

## 整体架构分析

### 2.1 整体架构图

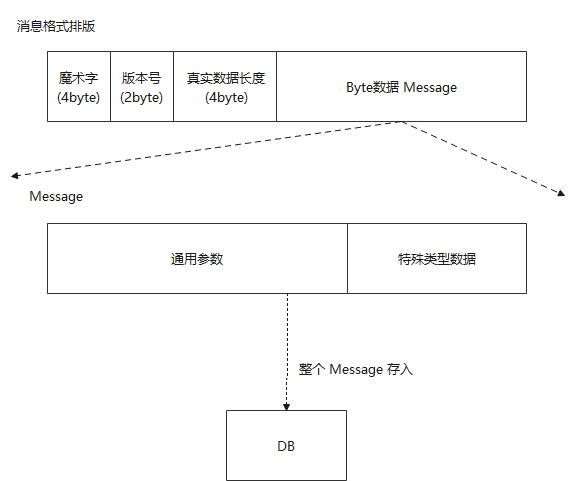


### 2.2 消息协议

在长连接通信中，为了避免TCP粘包和半包，客户端和服务器需要使用固定的格式进行消息通信。关键字段为：

* 魔术字段：用于确认这个协议包是否是我的服务器或客户端发来的，即暗码。默认用4个字节表示，默认用 bird
* 版本号：版本号用于标识当前这个消息的版本是多少，这是为了在以后的升级中，对消息做版本兼容使用。使用short类型(2个字节)表示
* 真实消息数据：表明之后的真实消息数据有多大，默认用 int 类型(4个字节表示)

**整体消息格式如下：**



### 2.3 用户连接处理

用于不仅仅要在 api 模块进行连接，还要与 netty 通信服务建立长连接

### 2.4 token 校验方案

由于有多个微服务同时对外部提供访问，若是每个微服务都向认证服务器验证 token，会对认证服务器造成性能压力。因此选择了 jwt 无状态令牌，当用户登录时，用户服务向验证服务请求令牌，并存入 Redis 中，设置有效期限。当用户持有 jwt 令牌访问时，各个服务去访问 Redis 取出对应的令牌，若能取出则是正确的token，同时判断若token的有效时间仅剩下半分钟，则刷新这个 token 的过期时间。

**为什么选择JWT:**

JWT 中可以包含少量的用户数据，且加解密效率高，当服务接收到 jwt后，可以尝试解析得到对应的数据内容用来处理。

**token 刷新：**

使用判断，避免大量的写入操作影响 redis 性能

## 项目模块功能分析

### 3.1 用户模块

3.1.1 基础功能

* 登录、注册、忘记密码
* 添加好友、删除好友
* 创建群组、解散群组
* 个人信息查看和修改
* 用户默认设置

### 3.2 群组模块

3.2.1 基础功能

* 群内身份管理
* 创建群组
* 解散群组
* 邀请入群
* 退出群组
* 踢出群组
* 群内昵称
* 群内设置
* 成员管理

### 3.3 权限验证模块

3.3.1 基础功能

* 用户的 session 管理和验证

### 3.4 消息模块

3.4.1 基础功能

* 私聊
* 群聊
* 控制消息(群组或后台)
* 系统消息

### 3.5 后台模块

3.5.1 基础功能

* 用户管理
* 群组管理
* 实时数据系统
* 日志系统
* 压测系统

## 数据库表设计

### 4.1 im\_restapi 库

#### 4.1.1 服务器配置表(sysConfig)

主要存储默认的服务器配置，其中的限制以服务器的优先级最高。同时客户端需要根据服务器的配置切换不同的界面。每次服务器修改配置，需要通知客户端重新登录

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| keepAliveTimeout | int | 心跳包间隔时间(ms) |
| roomMemLimit | int | 默认创建群组的最大成员数 |
| isAccountLogin | int | 是否使用账号登录 0 账号注册 1 手机号注册 |
| canMemberChat | int | 是否允许群成员私聊(0 不允许 1 允许) |
| canMemberInvite | int | 是否允许普通群成员邀请好友 |
| canMemberShareFile | int | 是否允许普通群成员上传群文件 |
| canMemberShow | int | 是否允许普通群成员查看群内成员 |
| userNameSearch | int | 通过用户名搜索 (0 不允许 1 模糊搜索 2精确搜索) |
| nickNameSearch | int | 通过昵称搜索 (0 不允许 1 模糊搜索 2精确搜索) |
| isSaveRequestLog | int | 是否开启请求日志 |
| isFriendVerify | int | 添加好友是否需要好友验证 |

#### 4.1.2 客户端配置表(clientConfig)

客户端配置表是用于用户端获取config 数据时返回给客户端的，包括请求地址、长连接地址、资源地址等，以及对应的app 版本号。但由于有些配置在客户端和服务器都需要处理，故实际返回给用户应该是客户端配置+服务器部分配置

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| apiUrl | String | api 的请求地址，也可能是网关地址 |
| socketUrl | String | 通讯长连接地址 |
| uploadUrl | String | 资源上传地址 |
| downloadUrl | String | 资源下载地址 |
| multiDeviceLogin | int | 是否允许多设备登录 |
| iosVersion | String | ios当前最新版本号 |
| iosDisable | String | ios 最低支持的版本号(低于此版本号就不再支持) |
| windowsVersion | String | windows当前最新版本号 |
| windowsDisable | String | windows 最低支持的版本号(低于此版本号就不再支持) |
| androidVersion | String | android当前最新版本号 |
| androidDisable | String | android 最低支持的版本号(低于此版本号就不再支持) |
| macVersion | String | mac当前最新版本号 |
| macDisable | String | mac 最低支持的版本号(低于此版本号就不再支持) |
| webVersion | String | web当前最新版本号 |
| webDisable | String | web 最低支持的版本号(低于此版本号就不再支持) |

#### 4.1.3 用户表(user)

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| uId | int | 用户id |
| account | String | 账号 |
| encryAccount | String | 加密的通讯账号 |
| password | String | 密码 |
| username | String | 用户昵称 |
| role | int | 账号的身份：普通用户=0 公众号=1 |
| sex | int | 性别：男=0 女=1 |
| status | int | 用户状态 正常=1 冻结=2 删除=3 |
| createTime | long | 账号的创建时间 |
| modifyTime | long | 最近修改时间 |
| lastLoginTime | long | 最后一次登录的时间 |
| platform | int | 最后一次登录的平台  未知设备：1  Window： 2  Android: 3  IOS: 4  Web: 5 |
| appVersion | String | 最后一次登录的app的版本 |
| ip | String | 最后一次登录的ip |

### 4.2 im\_room 库

#### 4.2.1 群组表(room)

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| rId | ObjectId | 群组唯一id |
| name | String | 群组名称 |
| desc | String | 群描述 |
| lastNotice | String | 房间的最新公告 |
| notices | List<Object(Notice)> | 公告列表 |
| memSize | int | 当前成员数 |
| maxMemSize | int | 允许的最大成员数 |
| member | Object(Member) | 自己在群内的身份 |
| createTime | long | 创建时间 |
| modifyTime | long | 修改时间 |
| isNeedVerify | int | 入群是否需要验证 |
| isShowMember | int | 是否显示群成员(0不显示 1显示) |

#### 4.2.2 群公告表(room\_notice)

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| nId | String | 公告唯一id |
| rId | ObjectId | 群组的id |
| content | String | 群公告内容 |
| pubUserId | int | 发布人id |
| pubUserName | String | 发布人昵称 |
| createTime | long | 发布时间 |
| modifyTime | long | 最近修改时间 |
| status | int | 公告状态 1正常 2删除 |

#### 4.2.3 群聊消息表(mucmsg\_[rid])

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

通过 mucmem\_[对应群组的rid] 来存储群内成员，当一个群组被解散后，一个月或几个月后就可以删除数据，避免占用 mongodb 的内存空间

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| rId | ObjectId | 成员所属的群组id |
| uId | int | 成员对应的用户id |
| nickName | String | 入群时的用户昵称 |
| remarkName | String | 群内昵称 |
| role | int | 成员角色：创建者1 管理员2 普通成员3 |
| offLineNoPushMsg | int | 是否开启免打扰 0不开启 1开启 |
| noSayingTime | long | 是否被禁言 0 正常 -1 永久禁言 >0 有禁言时间 |
| noSayingStart | long | 当成员被禁言时，禁言时间计算的起始时间 |
| createTime | Long | 成员的创建时间 |
| modifyTime | Long | 修改时间 |
| isTopChat | Long | 是否开启置顶聊天 不开启0 开启>0，根据开启时间大小来置顶排序 |
| joinInfo | String | 进群方式 |

### 4.3 im\_room\_member 库

#### 4.3.1 群成员表(mucmem\_[rid])

通过 mucmem\_[对应群组的rid] 来存储群内成员，当一个群组被解散后，一个月或几个月后就可以删除数据，避免占用 mongodb 的内存空间

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| rId | ObjectId | 成员所属的群组id |
| uId | int | 成员对应的用户id |
| nickName | String | 入群时的用户昵称 |
| remarkName | String | 群内昵称 |
| role | int | 成员角色：创建者1 管理员2 普通成员3 |
| offLineNoPushMsg | int | 是否开启免打扰 0不开启 1开启 |
| noSayingTime | long | 是否被禁言 0 正常 -1 永久禁言 >0 有禁言时间 |
| noSayingStart | long | 当成员被禁言时，禁言时间计算的起始时间 |
| createTime | Long | 成员的创建时间 |
| modifyTime | Long | 修改时间 |
| isTopChat | Long | 是否开启置顶聊天 不开启0 开启>0，根据开启时间大小来置顶排序 |
| joinInfo | String | 进群方式 |

### 4.4 im\_friend 库

#### 4.3.1 私聊消息表(muc\_friend\_msg)

私聊消息要存两条，to 和 from 相反的两条！

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### 4.5 im\_userId\_roomIds 库

#### 4.3.1 用户id和群id表(xxxx 用户id取模10)

私聊消息要存两条，to 和 from 相反的两条！

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 项目流程设计

### 3.1 注册

支持用户通过手机号注册，为了能后期兼容短信验证码方式，手机号格式暂时限定于中国手机号格式。密码暂无限制

### 3.2 登录

支持用户通过账号、密码登录，但以后要兼容第三方登录以及短信验证方式登录

### 3.3 找回密码

一期先让用户输入账号就能修改密码，后期接入短信平台，通过验证手机号的方式登录

### 3.4 上传/下载

支持用户上传和下载文件，暂不做格式限制，但上传大小限制为100M，后期增加断点续传和实时进度反馈

### 3.5 查看上传列表

支持查看自己上传了的文件列表，并能对其进行操作。

支持文件名搜索，和限定上传时间范围

### 3.6 删除自己上传的文件

支持用户将自己上传的文件删除，但应提供给用户两种，一种是彻底删除，另一种是放进回收站。二者都不显示在自己的上传文件列表中，但彻底删除的文件不能恢复，且不会在垃圾桶列表中展示

### 3.7 垃圾桶

用户在删除文件时，没有选择彻底删除，则会被放进垃圾桶中。垃圾桶中的的文件可以被恢复(即文件依旧先保存着)。同时提供彻底删除选项。

### 3.8 彻底删除

用户可以在我的存储中删除文件时，以及垃圾桶中，彻底删除某个文件，这个文件记录将不会再显示我的存储中

## 项目结构

前端： upload-client-web 用户页面

upload-manager-web 管理页面

后端： upload-parent

---- upload-user 用户服务

---- upload-file 文件服务

---- upload-manager 管理端服务