即应——匹配信息发布平台

软件架构文档

版本 <1.1>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
| <01/07/2019> | <1.0> | 添加简介 | 855小组 |
| <09/07/2019> | <1.1> | 了解技术栈之后，完善架构设计 | 855小组 |
| <01/09/2019> | <1.2> | 项目完善后，对原有架构做稍微修改 | 855小组 |
|  |  |  |  |

目录

1. 简介 2

1.1 目的 2

1.2 参考资料 2

2. 用例视图 2

3. 逻辑视图 2

3.1 概述 2

3.2 在构架方面具有重要意义的设计包 2

4. 进程视图 2

5. 部署视图 2

6. 实现视图 2

7. 数据视图（可选） 2

8. 核心算法设计（可选） 2

软件架构文档 （简化版）

# 简介

## 目的

本文档将从架构方面对系统进行综合概述，其中会使用多种不同的架构视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

## 参考资料

1. 即应匹配信息发布平台需求规约，1.4版，summer855小组
2. 即应匹配信息发布平台架构设计模型 OOM， 1.4版， summer855小组

# 用例视图



上图为“即应——拼单/活动一体化信息平台”的用例视图，所有usecase为普通用户的核心功能。

其中用户核心功能如下：

**发布网购拼单/外卖拼单/拼车/简单活动信息**是用户使用即应的最主要原因，用户通过此功能发布信息，从而找到理想的用户一起进行拼单/活动，从而减免邮费，获取更低的价格，通过更便捷的方式，来找到理想的朋友。

**浏览网购拼单/外卖拼单/拼车/简单活动**是用户寻找理想的条目的途径，用户通过搜索等方法找到想要参加的条目，然后申请加入。

**接受网购拼单/外卖拼单/拼车/简单活动申请**，用户可以再次接受申请，也可以拒绝申请。

**浏览我的网购拼单/外卖拼单/拼车/简单活动**，用户可以查看自己参与的条目，并对其中的条目进行管理，包括退出等

**网购拼单/外卖拼单/拼车/简单活动成员管理**是信息发布者的权限，发布者可以对参与进来的用户网友进行筛选，可以统一用户的加入，也可以踢出用户。

**利用活动群聊进行即时通讯，**用户加入某个活动后，可以利用平台的群聊方便地进行沟通。

**个人信息管理，**用户可以在平台修改自己的个人信息（如昵称、专业、生日）等。

**查看浏览历史记录，**用户可以查看自己最近浏览过的活动。

**浏览推荐活动，**根据用户的用户画像，平台将推给用户其可能感兴趣的活动。

**关注/查看关注者/查看关注者的活动，**用户可以在平台关注其他用户，关注后可以收到其他用户活动的推送。

管理员核心功能如下：

**管理用户**，包括封禁，解除封禁，处理用户的举报

**管理平台内网购拼单/外卖拼单/拼车/活动信息**，可以对已发布的信息进行过滤，对于违规的内容，违规的信息进行删除，禁止等操作

# 逻辑视图

## 概述



微服务架构下，将系统拆分成细粒度的微服务。

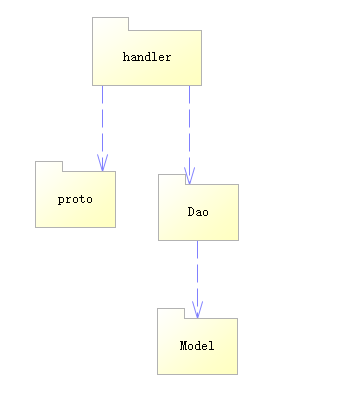
即应匹配信息发布平台拆分成活动服务、用户服务、认证服务、反馈服务、推荐服务，每个服务专注与服务本身的事情，当需要调用其他服务的时候通过Api gateway进行调用。

反向代理（reverse proxy）用于统一请求到服务器的接口，所有的请求都发向这个路由，反向代理通过判断路由Endpoint，来将请求转发到对应的端点

群聊服务是基于XMPP协议，通过UDP发送广播消息，这一部分对并发要求比较高，设计比较困难，后期再进行完善细化

## 在构架方面具有重要意义的设计包

### Activity service



Activity service的包名就是activity。

其中主要负责的逻辑是对Activity添加，删除，修改，查询。添加活动会存入一个聊天的群聊记录，这个群聊和活动一对一存在。

### Auth service

Auth service的包名是auth-service。

其主要的逻辑就是对登录进行验证，查询用户在线状态，登录后返回jwt

### User service

User service的包名是user

其主要的逻辑就是注册，修改用户信息，禁用用户，删除用户，查询用户信息

### Feedback service

Feedback service的包名就是feedback

其主要的逻辑是处理（查询、存储、统计）用户对其他用户的反馈。

### Recommendation service

Recommendation service的包名就是recommendation

其主要的逻辑是在后台对收集的用户信息建立用户画像，从而利用协同过滤等方法对用户参与的activity进行推荐

### Group chat service

Group chat service是一个比较大的构件

其主要的逻辑是在用户发布一个活动后，建立一个生命周期和活动同步的群聊服务，参与到该活动中的用户可以进行群聊商讨活动策略，安排等

[对于每个重要的包，都用一个小节来加以说明，其中应包括该包的名称、简要说明以及显示该包中所有重要的类和包的图。

对于该包中的每个重要类，应包括其名称、简要说明，还可选择包括对其部分主要职责、操作和属性的说明。]

# 进程视图

[本节说明将系统分解为轻量级进程（单个控制线程）和重量级进程（成组的轻量级进程）的情况。本节的内容按照各个通信或交互的进程组来进行组织。说明进程之间的主要通信模式，例如消息传递、中断和会合。]



Nginx反向代理服务器是程序的入口，前端发送请求至这个入口，nginx进程如理请求，及那个其分发到其他进程，根据路由格式分发到micro api gateway或者群聊的控制器中。

Micro api通过proto协议，使用grpc调用，通过Http/2协议调用其他进程。Micro api收到nginx反向代理服务器的请求后，如果服务可用，主线程就会进行到load balance处，默认使用roundRobin策略进行负载均衡，然后调用其他进程服务，最后返回给调用者。如果服务不可访问，则主线程进入到Fault tolerance，返回预先定义好的容错机制进行处理。

# 部署视图



Server中包含多个服务，除了GroupChat以外的其他服务分布在一个Server集群上。其中GroupChat（群聊）服务不属于微服务架构中的部分，该群聊服务通过XMPP协议建立连接，发送聊天信息。

群聊服务部署在一个单独的slave上，并对外部直接开放端口。

# 实现视图



一个包即为一个构件，所有构件对API gateway提供proto接口，并将接口实现。API Gateway监听http请求，针对对应的请求，调用相应的controller，然后通过proto调用其他构件实现的方法

# 数据视图（可选）



持久性数据主要包含Activity和User的信息，User信息存储在Mysql中，Activity存储在MongoDB中。

# 核心算法设计（可选）

[对系统中的核心算法进行设计。如果没有什么重要的算法，那么本节就为可选。]