<Mardrobe>

软件架构文档

版本 <5.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <12/7/2019> | <1.0> | <从构架方面对系统进行综合概述> | <张帅，余张辉，李佳睿，高治远> |
| <19/7/2019> | <2.0> | <针对第二次迭代内容进行扩展> | <张帅，余张辉，李佳睿，高治远> |
| <20/7/2019> | <3.0> | <针对第三次迭代内容进行扩展> | <张帅，余张辉，李佳睿，高治远> |
| <26/7/2019> | <4.0> | <针对第四次迭代内容进行的扩展> | <张帅，余张辉，李佳睿，高治远> |
| <29/7/2019> | <4.1> | <针对答辩意见进行的修改> | <张帅，余张辉，李佳睿，高治远> |
| <08/9/2019> | <5.0> | <针对第四次迭代内容进行的扩展> | <张帅，余张辉，李佳睿，高治远> |

目录

1. 简介 3

1.1 目的 3

1.2 参考资料 3

2. 用例视图 3

3. 逻辑视图 3

3.1 概述 3

3.2 在构架方面具有重要意义的设计包 3

4. 进程视图 3

5. 部署视图 3

6. 实现视图 3

7. 数据视图（可选） 3

8. 核心算法设计（可选） 3

软件架构文档 （简化版）

# 简介

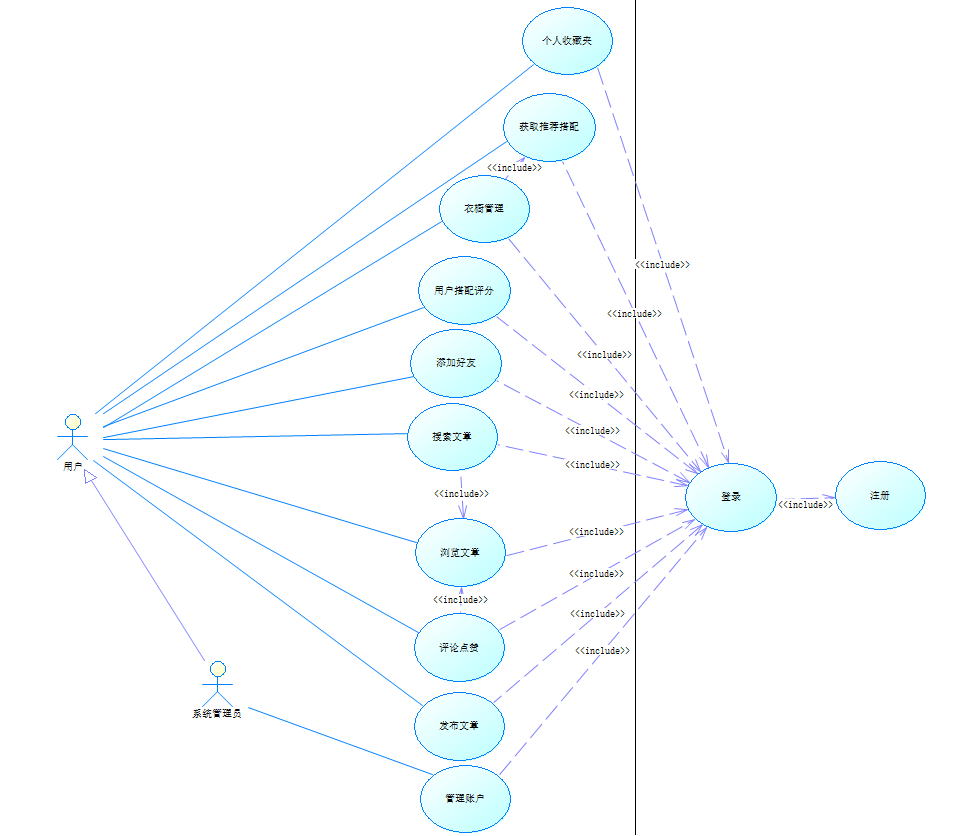
## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架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

暂无

# 用例视图



Actor：

（1）用户：包含系统管理员在内的一切可以使用app的人

（2）系统管理员：有权限对后台账户进行访问和修改的人

Use Case：

（1）注册：用户填写信息来获取使用app的账户

（2）登录：用户通过账号密码来进入自己app的专属空间

（3）获取推荐搭配：用户选取一件并得到推荐搭配

（4）衣橱管理：用户上传衣服并进行管理

（5）添加好友：用户之间搜索好友并添加

（6）搜索文章：用户输入文字，系统根据输入内容查找对应的文章

（7）浏览文章：用户在社区中浏览所有发布的文章

（8）评论点赞：用户对浏览的文章进行评论、点赞或分享

（9）发布文章：用户在社区中发布自己的搭配图片、文字，并可以进行管理

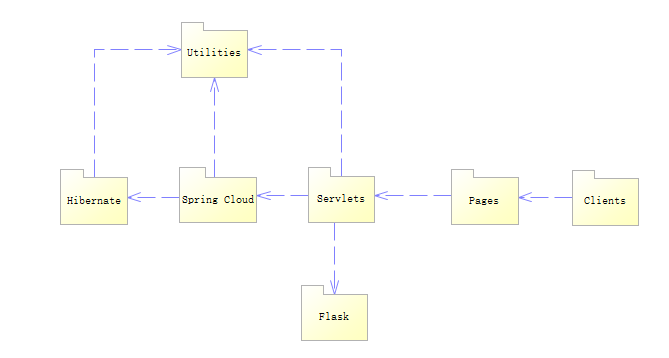
（10）管理账户：系统管理员进入后台管理账户数据

（11）用户搭配评分：系统对用户给出的搭配进行评分

（12）个人收藏夹：用户收藏喜爱的系统搭配

# 逻辑视图

## 概述



总体如上，分为Clients包，Pages包，Servlets包, Spring Cloud包，Hibernate包，Utilities包，Flask包。

Clients包，包括客户端中运行的脚本；

Pages包，包括所有静态页面及静态素材的类；

Servlets包,包括相应客户端HTTP请求的类;

Spring Cloud包，包含使用Spring框架实现的业务逻辑类；

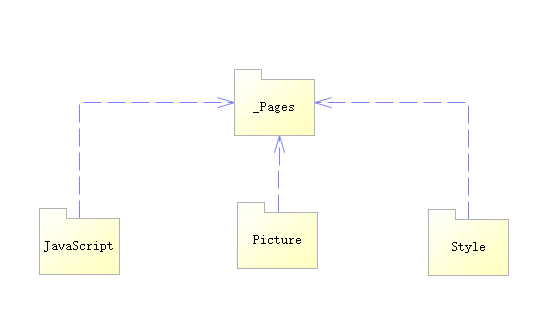
Hibernate包，包括使用O/R映射机制实现数据持久性的类；

Utilities包，包括贯穿系统各个部分诸如事务服务类和数据库连接服务类等公共服务类；

Flask包，包括搭配推荐的后端python脚本

## 在构架方面具有重要意义的设计包

**Pages包：包括所有静态页面及静态素材的类；**



JavaScript：包含所有的js类，负责渲染静态文件

Picture：包含所有的静态图片

Style：包含所有的样式类

**Spring Cloud包，包含使用Spring框架实现的业务逻辑类；**

****

UserController:负责控制用户管理有关的通信

MessageController:负责控制消息管理有关的通信

WardrobeController:负责控制衣橱管理有关的通信

CommunityController:负责控制社区管理有关的通信

FavoriteController:负责控制收藏夹管理有关的通信

UserService:负责用户管理有关的业务逻辑

MessageService:负责消息管理有关的业务逻辑

WardrobeService:负责衣橱管理有关的业务逻辑

CommunityService:负责社区管理有关的业务逻辑

FavoriteService:负责收藏夹管理有关的业务逻辑

**Hibernate包，包括使用O/R映射机制实现数据持久性的**

****

UserRepository:负责User类有关的数据持久化

MessageReposity:负责Message类有关的数据持久化

WardrobeReposity:负责Wardrobe类有关的数据持久化

CommunityReposity:负责Community类有关的数据持久化

FavoriteReposity:负责Favorite类有关的数据持久化

**Utilities包，包括贯穿系统各个部分诸如事务服务类和数据库连接服务类等公共服务类**



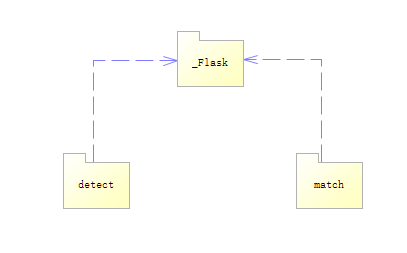
UserEntity:User的实体类，储存用户的相关信息

MessageEntity:Message的实体类，储存消息的相关信息

WardrobeEntity:Wardrobe的实体类，储存我的衣橱的相关信息

CommunityEntity:Community的实体类，储存社区的相关信息

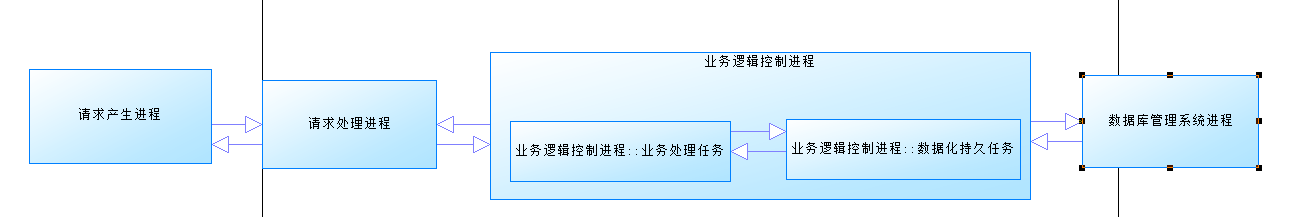
FavoriteEntity:Favorite的实体类，储存收藏夹的相关信息

**Flask包，包括搭配推荐的后端python脚本**

detect:用于从图片中剪切衣服的脚本模块

match:用于提取特征并和数据库中匹配的脚本模块

# 进程视图



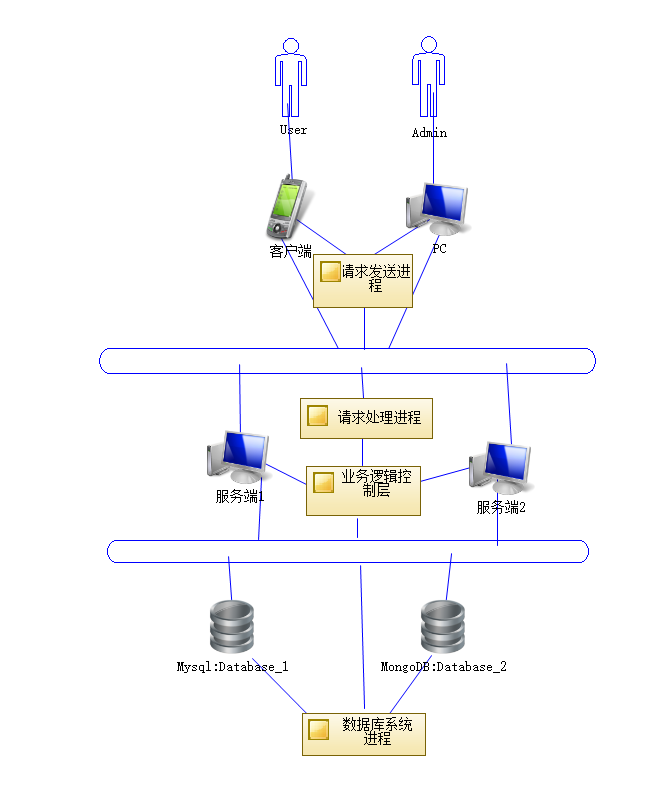
请求产生进程：用户通过Android app访问系统，对用户的不同操作发出相应的请求

请求处理进程：接收来自app的http请求，处理后转发给业务逻辑层

业务逻辑控制进程：执行业务逻辑，将数据变化发送给数据库管理系统进程，将处理结果组装成http响应后发送给客户端

数据库管理系统进程：管理储存在数据库中的数据，如社区文章和用户信息等

# 部署视图



客户端：手机Android客户端，用户在此操作

PC：管理员账户管理界面，可以对所有用户账户进行查询、封禁、添加和重置密码等操作

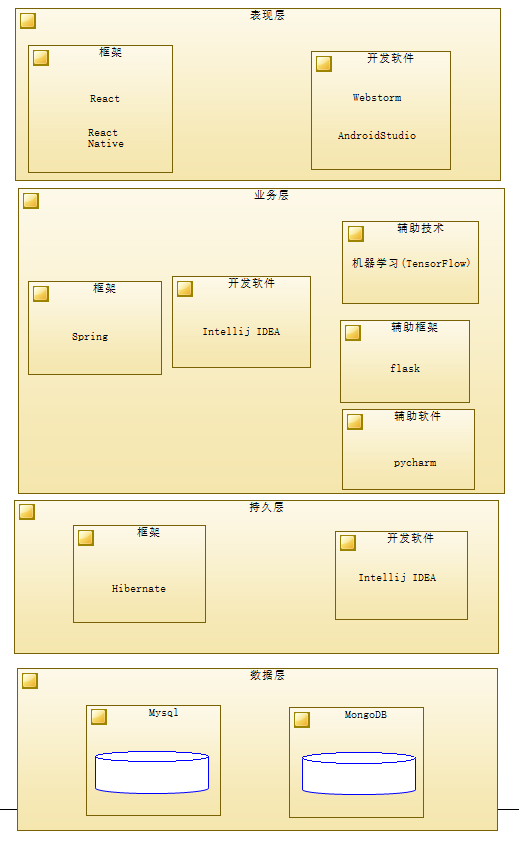
服务端1：对软件除了搭配推荐外的注册登录、添加好友等功能的请求进行业务逻辑处理，并返回处理结果

服务端2：对上传衣服进行匹配推荐，传回推荐的衣服（整套）图片

Mysql：存储用户信息

MongoDB：存储包含图片的搭配库、社区动态等数据

# 实现视图



软件共分为四层：

表现层：用户端使用AndroidStudio软件进行开发,运用react native框架，js语言编写

管理员界面使用软件Webstorm软件进行开发,运用react框架，js语言编写

业务层：对于搭配推荐的算法部分采用辅助技术机器学习，通过pycharm软件进行开发，引入 TensorFlow相关包，使用框架flask，python语言编写；

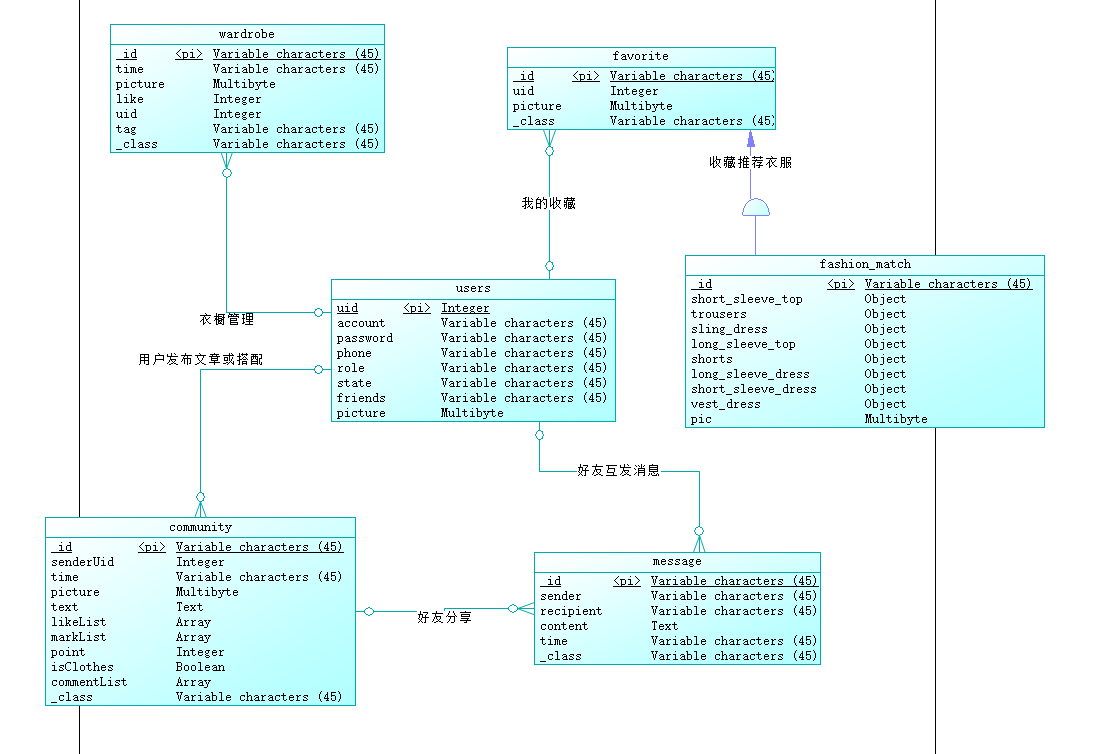
对于剩余的用户管理，社区管理，消息管理，收藏夹管理，衣橱管理部分的业务逻辑，使用Intellij Idea软件开发，采用Spring框架，java语言编写

持久层：在Intellij Idea软件中进行开发，使用Hibernate框架，java语言编写

数据层：使用mysql存储固定结构的表，如users

使用mongoDB存储文档型、图片型数据，如发布的动态、收藏夹

# 数据视图（可选）



实体类：

Users：用户信息

Community：社区文章

Message：好友消息

Wardrobe：我的衣橱

Favorite：个人收藏的搭配

Fashion——match：穿衣搭配库

联系：

衣橱管理：一个用户可以在衣橱中上传多件衣服

用户发布文章或搭配：一个用户可以在社区中发布多篇文章

好友分享：一条动态可以在多条私信中分享

好友互发消息：一个用户可以给所有自己的好友发消息

我的收藏：一个用户可收藏多件搭配

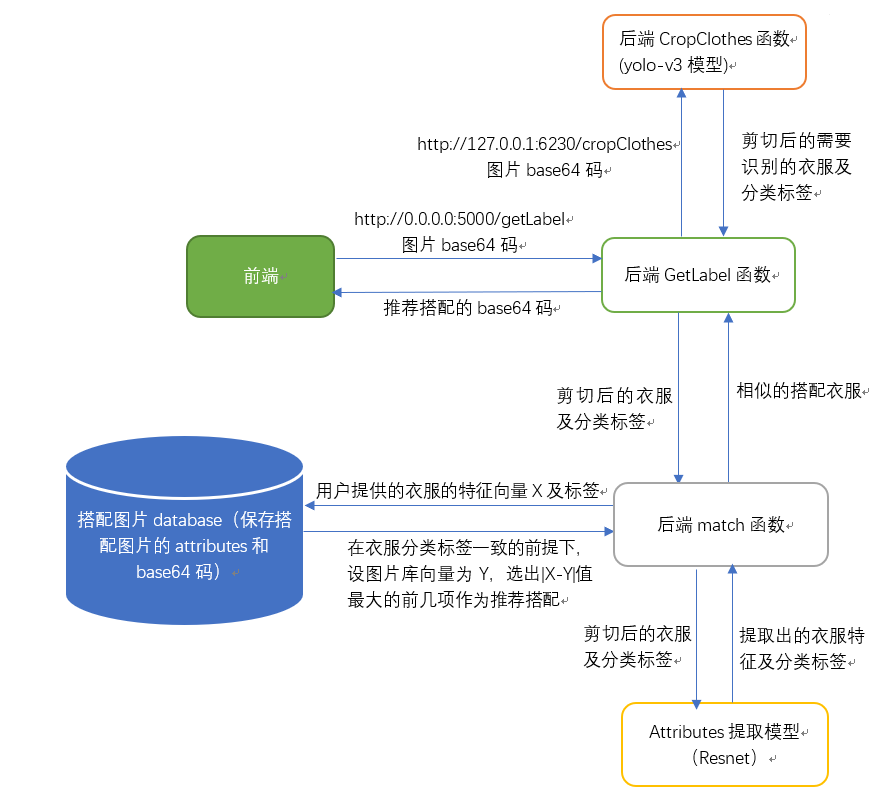
收藏推荐衣服：用户从搭配库给出的衣物中挑选喜欢的加入收藏

# 核心算法设计（可选）

参考DeepFashion的数据集，采用yolo-v3的Detect服装识别截取模型，以及用于特征提取的算法模型，最后由计算服装间特征的相似度的搭配算法在数据库中得到用户端收到的搭配推荐。

**剪切与分类标签：**将图片通过yolo模型后，会将可能是衣服的部分框取起来，并给出对应的置信度。通过置信度的筛选，将最有可能是衣服的框保留下来，通过cv库将该部分图片截取出来。在通过模型后，每个框还计算了一个13维的向量，其中的数字（score）分别对应于短袖上装、短裤等衣物种类。对于之前选出的置信度最高的框，取最高score对应的衣服种类作为该衣服识别出的标签。

**匹配算法：**将输入衣服的特征与数据库中的已有特征用余弦相似度进行比较，选出相似度最高的前五套搭配作为最终的推荐搭配。



**流程简介**：

1. 前端发送图片的base64码到用于匹配的后端getLabel函数接口
2. getLabel函数调用用于分类识别的后端cropClothes函数接口，传入base64码，通过yolo-v3模型检测后输出剪切后的衣服图片base64码和对应的分类标签
3. getLabel函数调用match函数，针对剪切后的衣服和分类标签在match函数中进行匹配
4. match函数把输入的衣服图片通过Resnet模型提取出对应的特征向量X
5. 和数据库交互，数据库中每张搭配存有对应的衣服种类、图片的base64码和对应的特征向量Y。在匹配时首先保证衣服种类一致，然后计算X与Y的余弦距离作为误差loss，按照loss大小取最相似的前十张搭配图片传回到getLabel函数，再传回到前端显示