目录

[1 项目可行性研究报告 6](#_Toc34426718)

[1.1 背景 6](#_Toc34426719)

[1.2 可行性研究前提 6](#_Toc34426720)

[1.2.1 基本要求 6](#_Toc34426721)

[1.2.2 目标 6](#_Toc34426722)

[1.2.3 条件 6](#_Toc34426723)

[1.2.4 可行性研究方法 6](#_Toc34426724)

[1.2.5 评价尺度 7](#_Toc34426725)

[1.3 社会因素方面的可行性 7](#_Toc34426726)

[1.3.1 法律方面的可行性 7](#_Toc34426727)

[1.3.2 使用方面的可行性 7](#_Toc34426728)

[1.4 结论 7](#_Toc34426729)

[2 项目开发计划 7](#_Toc34426730)

[2.1 概述 8](#_Toc34426731)

[2.1.1 工作内容 8](#_Toc34426732)

[2.1.2 产品 8](#_Toc34426733)

[2.1.3 验收标准 9](#_Toc34426734)

[2.2 实施计划 9](#_Toc34426735)

[2.2.1 工作分解与人员分工 9](#_Toc34426736)

[2.2.2 进度计划 9](#_Toc34426737)

[2.2.3 关键问题 10](#_Toc34426738)

[3 需求分析 11](#_Toc34426739)

[3.1 任务概述 11](#_Toc34426740)

[3.1.1 目标 11](#_Toc34426741)

[3.1.2 用户的特点 11](#_Toc34426742)

[3.1.3 假定和约束 11](#_Toc34426743)

[3.2 需求规定 12](#_Toc34426744)

[3.2.1 功能需求 12](#_Toc34426745)

[3.2.2 非性能性需求 14](#_Toc34426746)

[3.3 运行环境规定 14](#_Toc34426747)

[3.3.1 设备 14](#_Toc34426748)

[4 概要设计-服务端 14](#_Toc34426749)

[4.1 总体设计 14](#_Toc34426750)

[4.1.1 需求规定 14](#_Toc34426751)

[4.1.2 运行环境 15](#_Toc34426752)

[4.1.3 基本设计概念和处理流程 15](#_Toc34426753)

[4.1.4 结构 19](#_Toc34426754)

[4.2 接口设计 20](#_Toc34426755)

[4.2.1 用户接口 20](#_Toc34426756)

[4.2.2 外部接口 22](#_Toc34426757)

[4.2.3 内部接口 23](#_Toc34426758)

[4.3 系统数据结构设计 23](#_Toc34426759)

[4.3.1 逻辑结构设计要点 23](#_Toc34426760)

[4.3.2 物理结构设计要点 23](#_Toc34426761)

[4.3.3 数据结构与程序的关系 24](#_Toc34426762)

[4.3.4 系统出错处理设计 24](#_Toc34426763)

[4.3.5 出错信息 24](#_Toc34426764)

[5 概要设计-客户端 24](#_Toc34426765)

[5.1 总体设计 25](#_Toc34426766)

[5.1.1 需求规定 25](#_Toc34426767)

[5.1.2 运行环境 25](#_Toc34426768)

[5.1.3 结构 25](#_Toc34426769)

[5.1.4 基本设计概念和处理流程 26](#_Toc34426770)

[5.2 接口设计 30](#_Toc34426771)

[5.2.1 内部接口 30](#_Toc34426772)

[5.2.2 外部接口 32](#_Toc34426773)

[6 详细设计-服务端 32](#_Toc34426774)

[6.1 程序系统的结构 32](#_Toc34426775)

[6.1.1 整体架构 32](#_Toc34426776)

[6.1.2 图片处理 33](#_Toc34426777)

[6.1.3 用户处理 34](#_Toc34426778)

[6.1.4 数据库逻辑结构 34](#_Toc34426779)

[6.2 PicController 35](#_Toc34426780)

[6.2.1 程序描述 35](#_Toc34426781)

[6.2.2 功能 35](#_Toc34426782)

[6.2.3 输入项 35](#_Toc34426783)

[6.2.4 输出项 36](#_Toc34426784)

[6.2.5 流程逻辑 37](#_Toc34426785)

[6.2.6 接口 38](#_Toc34426786)

[6.3 PicInfoService 38](#_Toc34426787)

[6.3.1 程序描述 38](#_Toc34426788)

[6.3.2 功能 38](#_Toc34426789)

[6.3.3 输入项 39](#_Toc34426790)

[6.3.4 输出项 39](#_Toc34426791)

[6.3.5 流程逻辑 39](#_Toc34426792)

[6.3.6 接口 40](#_Toc34426793)

[6.4 PicInfoServiceImpl 40](#_Toc34426794)

[6.4.1 程序描述 40](#_Toc34426795)

[6.5 PicInfoMapper 41](#_Toc34426796)

[6.5.1 程序描述 41](#_Toc34426797)

[6.5.2 功能 41](#_Toc34426798)

[6.5.3 输入项 41](#_Toc34426799)

[6.5.4 输出项 42](#_Toc34426800)

[6.5.5 流程逻辑 42](#_Toc34426801)

[6.5.6 接口 43](#_Toc34426802)

[6.6 PicStorageService 43](#_Toc34426803)

[6.6.1 程序描述 43](#_Toc34426804)

[6.6.2 功能 44](#_Toc34426805)

[6.6.3 输入项 44](#_Toc34426806)

[6.6.4 输出项 45](#_Toc34426807)

[6.6.5 流程逻辑 45](#_Toc34426808)

[6.6.6 接口 46](#_Toc34426809)

[6.7 PicStorageServiceImpl 47](#_Toc34426810)

[6.7.1 程序描述 47](#_Toc34426811)

[6.8 UserController 47](#_Toc34426812)

[6.8.1 程序描述 47](#_Toc34426813)

[6.8.2 功能 47](#_Toc34426814)

[6.8.3 输入项 47](#_Toc34426815)

[6.8.4 输出项 48](#_Toc34426816)

[6.8.5 流程逻辑 48](#_Toc34426817)

[6.8.6 接口 49](#_Toc34426818)

[6.9 UserService 49](#_Toc34426819)

[6.9.1 程序描述 49](#_Toc34426820)

[6.9.2 功能 49](#_Toc34426821)

[6.9.3 输入项 50](#_Toc34426822)

[6.9.4 输出项 51](#_Toc34426823)

[6.9.5 流程逻辑 51](#_Toc34426824)

[6.9.6 接口 53](#_Toc34426825)

[6.10 UserServiceImpl 54](#_Toc34426826)

[6.10.1 程序描述 54](#_Toc34426827)

[6.11 UserMapper 54](#_Toc34426828)

[6.11.1 程序描述 54](#_Toc34426829)

[6.11.2 功能 54](#_Toc34426830)

[6.11.3 输入项 55](#_Toc34426831)

[6.11.4 输出项 55](#_Toc34426832)

[6.11.5 流程逻辑 56](#_Toc34426833)

[6.11.6 接口 56](#_Toc34426834)

[6.12 JwtUtil 57](#_Toc34426835)

[6.12.1 程序描述 57](#_Toc34426836)

[6.12.2 功能 57](#_Toc34426837)

[6.12.3 输入项 57](#_Toc34426838)

[6.12.4 输出项 57](#_Toc34426839)

[6.12.5 流程逻辑 57](#_Toc34426840)

[7 详细设计-客户端 58](#_Toc34426841)

[7.1 迭代1 58](#_Toc34426842)

[7.1.1 注册登录模块 58](#_Toc34426843)

[7.1.2 图片列表模块 59](#_Toc34426844)

[7.2 迭代2 60](#_Toc34426845)

[7.2.1 导航主页面 60](#_Toc34426846)

[7.2.2 查看大图模块 61](#_Toc34426847)

[7.2.3 用户图片列表模块 62](#_Toc34426848)

[7.3 迭代3 62](#_Toc34426849)

[7.3.1 导航主页面 62](#_Toc34426850)

[7.3.2 查看大图模块 63](#_Toc34426851)

[8 测试分析报告 63](#_Toc34426852)

[8.1 背景 64](#_Toc34426853)

[8.2 测试概要 64](#_Toc34426854)

[8.3 测试结果及发现 65](#_Toc34426855)

[8.3.1 Token获取 65](#_Toc34426856)

[8.3.2 注册功能 66](#_Toc34426857)

[8.3.3 登录功能 68](#_Toc34426858)

[8.3.4 显示图片 70](#_Toc34426859)

[8.3.5 图片删除 71](#_Toc34426860)

[8.3.6 图片上传 71](#_Toc34426861)

[8.3.7 获取用户图片列表 72](#_Toc34426862)

[8.3.8 图片缩略图展示 73](#_Toc34426863)

[8.3.9 下拉刷新 73](#_Toc34426864)

[8.3.10 更改瀑布流布局 74](#_Toc34426865)

[8.3.11 保存图片 75](#_Toc34426866)

[8.3.12 复制外链 76](#_Toc34426867)

[8.3.13 编辑图片 77](#_Toc34426868)

[8.3.14 分享图片 77](#_Toc34426869)

[8.3.15 显示图片信息 78](#_Toc34426870)

[8.4 分析摘要 79](#_Toc34426871)

[8.4.1 能力 79](#_Toc34426872)

[8.4.2 缺陷和限制 79](#_Toc34426873)

[8.4.3 建议 79](#_Toc34426874)

[8.4.4 评价 79](#_Toc34426875)

[9 缺陷报告 80](#_Toc34426876)

[9.1 单元测试测试用例发现的缺陷 80](#_Toc34426877)

[9.1.1 80](#_Toc34426878)

[9.1.2 81](#_Toc34426879)

[9.2 系统测试运行中发现的缺陷 82](#_Toc34426880)

[9.2.1 82](#_Toc34426881)

[9.2.2 83](#_Toc34426882)

[9.2.3 84](#_Toc34426883)

[9.2.4 84](#_Toc34426884)

[10 安装手册 86](#_Toc34426885)

[11 使用手册 87](#_Toc34426886)

[11.1 注册登录 87](#_Toc34426887)

[11.2 查看所有图片 88](#_Toc34426888)

[11.3 上传图片和查看我的上传 88](#_Toc34426889)

[11.4 对图片进行操作 90](#_Toc34426890)

[11.5 更改图片排列列数 92](#_Toc34426891)

[12 编码清单 93](#_Toc34426892)

[12.1 客户端源码 93](#_Toc34426893)

[12.2 服务器端源码 96](#_Toc34426894)

[13 项目总结 97](#_Toc34426895)

# 项目可行性研究报告

执笔:

完成日期:2019-05-05

审查：

## 背景

本项目为一以移动应用为用户入口的公共图床服务。

## 可行性研究前提

### 基本要求

本项目要实现的基本功能有：移动客户端登录、图片浏览、图片上传、图片外链获取等。

基本性能要求：同时处理多个用户的输入输出请求。

系统提供的输出信息，以图片集合为主。特征为请求时间随机，对数据时效性要求较高。

系统接受并存储来自用户的输入，数据以用户信息、用户图片为主。用户输入的特征为：请求时间随机，集中并发程度低。

完成期限：2019-6-13。

### 目标

实现功能需求。

实现性能需求。

具体较好的可拓展性。

### 条件

项目结束时间: 2019-6-13。

首版本集成上线最晚时间:2019-5-30。

应具备对应用内信息进行内容管制的机制，避免违规信息长时间出现。

用户移动设备系统版本假定:Android系统7.0及以上。

### 可行性研究方法

市场相关产品以及同类产品调查。

### 评价尺度

功能开发时间长短

功能使用难易程度

功能响应时间长短

## 社会因素方面的可行性

### 法律方面的可行性

本项目提供的应用服务存在用户上传违法违规信息的情况，应用负有监管主体责任。所以应用具备管控违规信息的机制是项目可行的基本要求之一。

### 使用方面的可行性

本项目提供的应用服务面向消费市场，故项目应采用消费大众普遍接受的交互形式来提供应用服务以确保项目可行性。

## 结论

本项目可以立即开始进行。

# 项目开发计划

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-05 | 制定迭代周期1的项目开发计划 |  |  |
| 2019-05-21 | 制定迭代周期2的项目开发计划 |  |  |
| 2019-06-05 | 制定迭代周期3的项目开发计划 |  |  |

## 概述

### 工作内容

#### 迭代1

* 搭建程序架构
* 集成操作系统/第三方API
* 实现前后端通信
* 实现核心功能需求：
* 用户登录注册
* 图片瀑布流

#### 迭代2

* 实现主要性能需求：
* 本地数据缓存
* 实现主要功能需求：
* APP导航页面UI
* 图片缩略图加快载入
* 图片外链获取
* 图片上传
* 图片删除

#### 迭代3

* 缺陷修复
* 基于用户体验的UI调整
* 实现次要功能需求：
* 保存原图
* 编辑图片
* 图片分享

### 产品

1. 程序：客户端程序app。
2. 文件：Android安装文件apk。
3. 服务：后端向客户端提供图片数据。
4. 非移交的产品：各种文档。

### 验收标准

1. 需求分析：需求分析文档
2. 概要设计：概要设计文档，主要说明程序大致框架流程。
3. 详细设计：详细设计文档，包括类图、api接口说明等。
4. 产品：可于Android7.0以上系统版本运行的Android App，需实现该迭代周期确定的需求。

## 实施计划

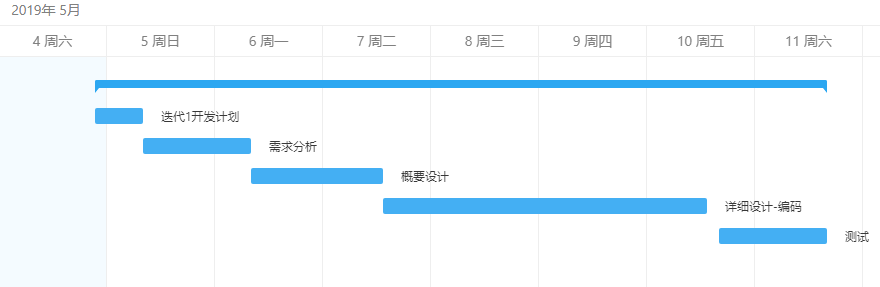
### 工作分解与人员分工

|  |  |  |
| --- | --- | --- |
| 任务 | 执行 | 文档审查 |
| 需求分析 |  |  |
| 概要设计(客户端) |  |  |
| 概要设计(后端) |  |  |
| 详细设计-编码(客户端) |  |  |
| 详细设计-编码(后端) |  |  |
| 测试 |  |  |

### 进度计划

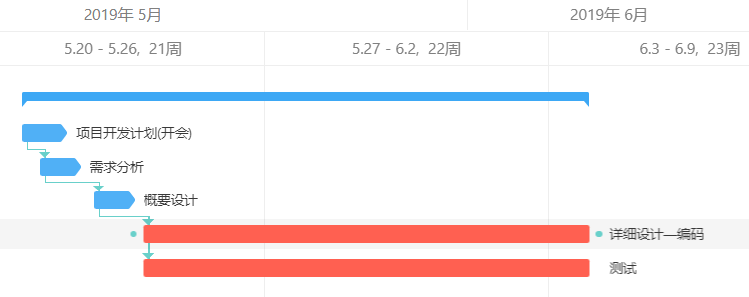
#### 迭代1

迭代1周期设定为一周，内部采用传统的进度流程，需求分析完成后，交付文档给开发人员确认审查，前后端开发人员据此进行概要设计，文档交叉互审，确保前后端交互接口一致，确保团队对功能需求的理解一致。



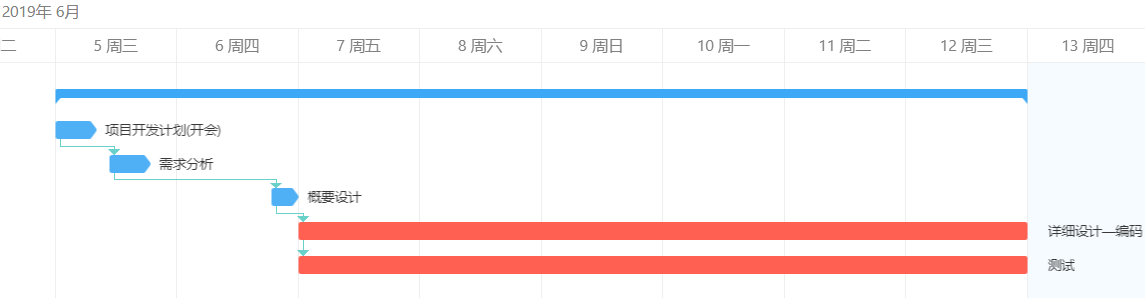
#### 迭代2

迭代周期2功能需求较多设定为两周，由于迭代周期1已经实现了前后端集成系统的可运行可持续集成，故而从第二个迭代周期开始，测试与编码可以并行进行，开发人员每天bulid一次，测试人员将他们集成并收集缺陷而后及时反馈，及时发现不符合需求的功能实现，尽早修改降低修改成本，需求与概要设计也会根据每日build的部分实现效果及时进行适当的调整。



#### 迭代3

迭代周期3主要以修复缺陷与调整用户UI为主。



### 关键问题

#### 迭代1

* 确定程序基本架构，优先考虑可拓展性。
* 基本的图片加载-显示，注意可能产生的性能问题。
* 确定网络通信对安全性的需求程度，过高的安全性要求将增加项目风险。

#### 迭代2

* 于Android系统实现功能需求所需要的非业务逻辑的代码(权限请求等)
* 考虑实现性能需求的方式

#### 迭代3

* 缺陷修复
* 整体性能调优
* 调整交互设计逻辑提高用户体验

# 需求分析

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-06 | 迭代周期1的需求分析 |  |  |
| 2019-05-22 | 迭代周期2的需求分析 |  |  |
| 2019-06-06 | 迭代周期3的需求分析 |  |  |

## 任务概述

### 目标

在Linux + Android平台下，开发使用C/S架构的安卓端图床。初期实现基本功能，详见[3.1]。如果初期顺利完成，则在其基础上进一步开发其图片分享功能，增加以图搜图以及防盗链的功能，增强软件的实用性和安全性。

### 用户的特点

该简易图床APP的用户主要是安卓端有在线图片存储的需求者以及外链分享的网络内容提供者。

### 假定和约束

A．一般约束：参加开发的人数为3人

B．经费：无

C. 期限：8周（2019.4.22-2019.6.14）

D．硬件环境：开发平台为Windows10，运行平台为Ubuntu14.04 64位

## 需求规定

### 功能需求

#### 用户界面操作。

1）注册

输入账号密码之后在无错误的情况下完成注册，注册完成后需要有注册成功的提示并且自动登录。

2）登录

能够用已经注册过的账号密码进行登录，如果账号密码错误显示错误信息。

3）注销

在点击注销按钮后需要有一个提示弹窗来确定是否注销

#### 显示图片

在主界面可以现实图库中所有开放的图片，并且根据双列瀑布流排列

#### 客户端主页面

设计客户端主页面，简洁美观，方便操作。

#### 图片上传功能

用户可以在客户端上传图片，要求大小不超过2MB，先实现单张上传

#### 获取用户图片列表

用户可以单独查看自己上传的所有图片

#### 图片缩略图

要求在主界面显示的所有图片都为缩略图。

#### 图片删除

长按图片后，用户能够选择删除操作，删除自己上传的图片。

#### 下拉刷新

在APP对界面进行下拉操作时，要求更够重新获取一次图片，并且将之前展示图片的缓存刷新。

#### 更改瀑布流布局

更改最开始的双行瀑布流布局，要求更够按照用户的心意进行多行的瀑布流布局

#### 保存图片

长按图片后，用户能够选择保存图片操作，将图库中的任意一张图保存到本地文件

#### 分享图片

长按图片后，用户能够选择分享图片操作，能够将文件夹中的图片分享到类似QQ的应用端

#### 复制外链

长按图片后，用户能够选择复制外链操作，生成一个查看图片的外链

#### 编辑图片

长按图片后，用户能够选择编辑图片操作，调用手机自带的编辑图片功能，对图片进行编辑

#### 图片详细信息

长按图片后，用户能够选择详细信息操作，能够查看图片的详细信息，包括上传时间，上传的用户，图片的大小

### 非性能性需求

#### 输入输出要求

注册账号时账号密码允许由大小写字母，下划线，数字和有限的符号组成。

#### 安全保密性要求

要对注册或者登陆的账号进行加密处理

#### UI设计要求

在力求简单易懂的情况下让界面变得更加美观，容易操作。

## 运行环境规定

### 设备

1. 安卓系统7.0以上的移动设备或者虚拟机
2. 可以运行tomcat的Window/Linux服务器

# 概要设计-服务端

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-06 | 迭代周期1的服务端概要设计 |  |  |
| 2019-05-08 | 前后端接口设计修改 |  |  |
| 2019-05-23 | 迭代周期2的服务端概要设计 |  |  |
| 2019-06-07 | 迭代周期3的服务端概要设计 |  |  |

## 总体设计

### 需求规定

本迭代周期的目标是建立注册/登陆，图片展示相关的数据库，实现功能良好的后端api。

### 运行环境

Ubuntu 14.04

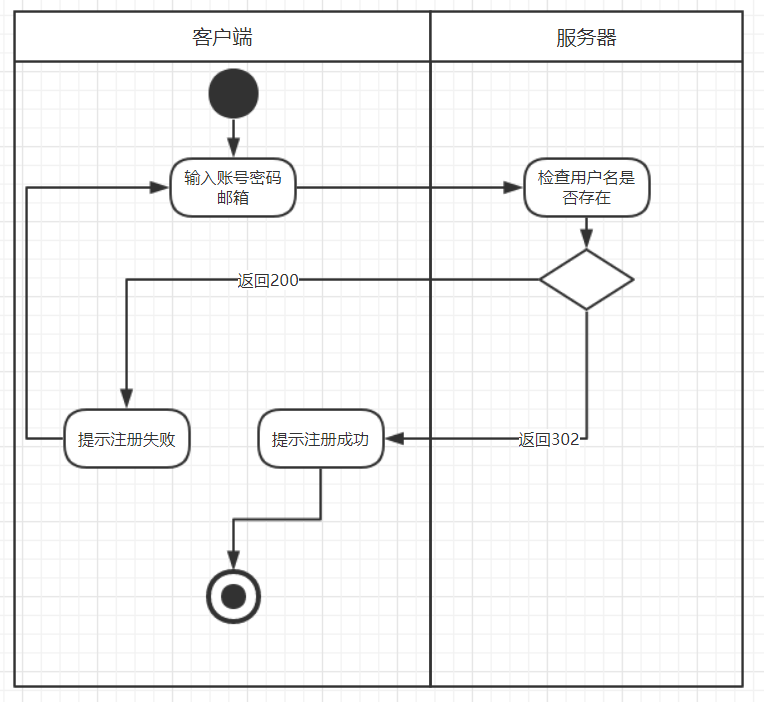
### 基本设计概念和处理流程

#### 登陆功能



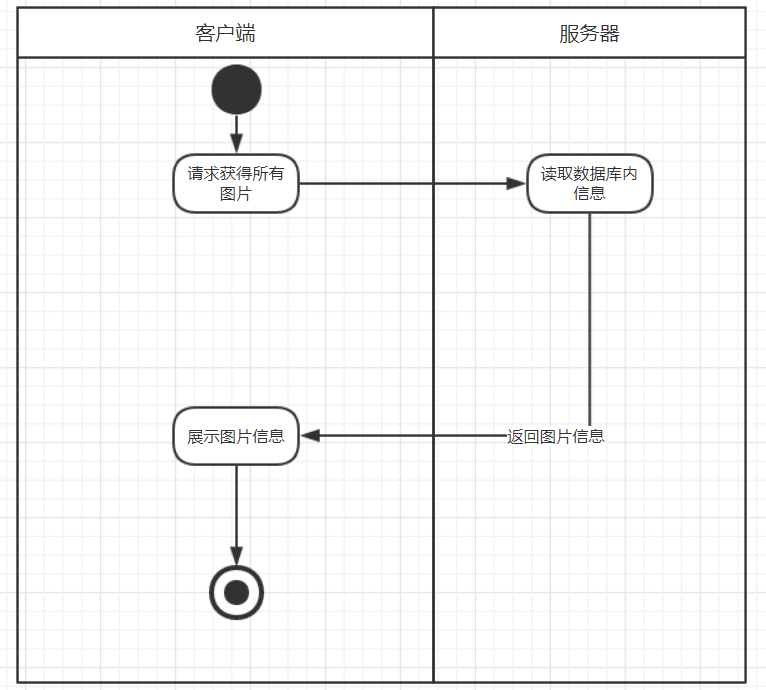
客户端传入账号密码，服务端检查是否合法，若合法传回302状态码并传回生成的jwt token

#### 注册功能



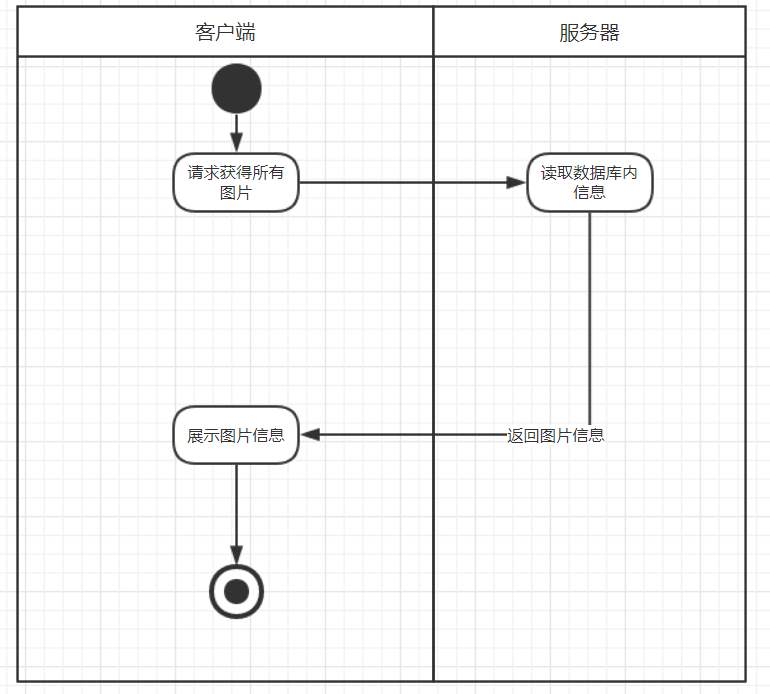
客户端提交用户名，密码，邮箱，客户端判断是否重名并返回对应状态码

#### 获得所有图片



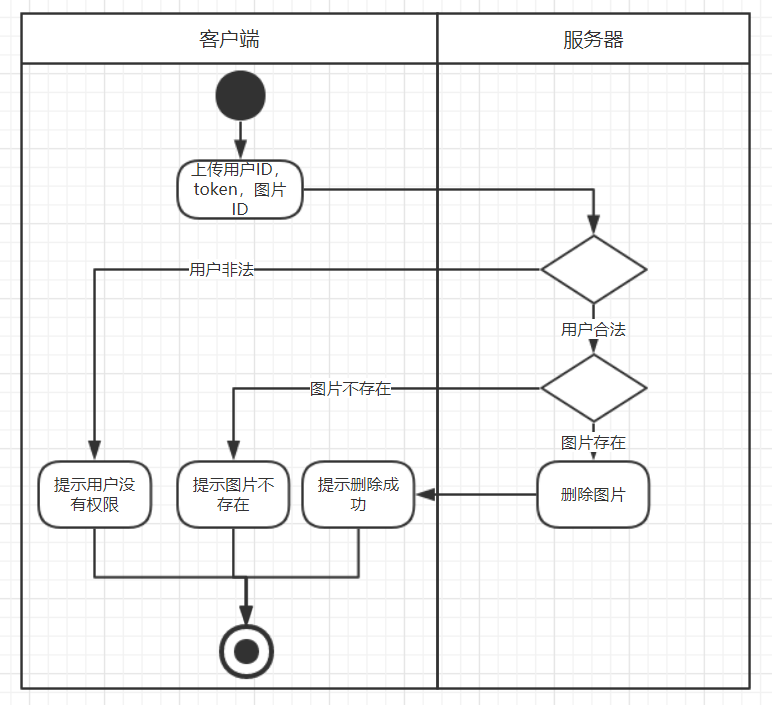
请求所有图片信息

#### 获得用户拥有图片



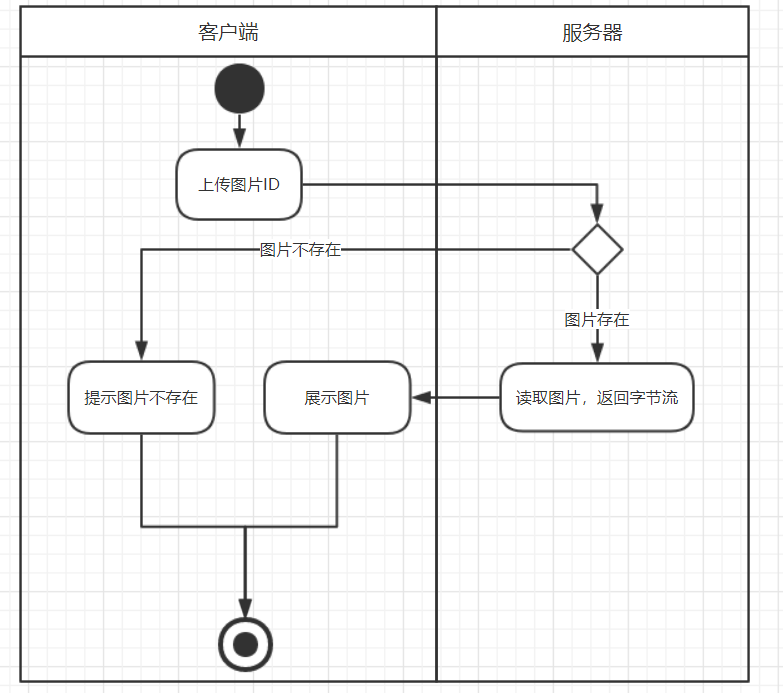
获得用户拥有的图片信息，上传用户名，返回图片信息JSON

#### 删除图片



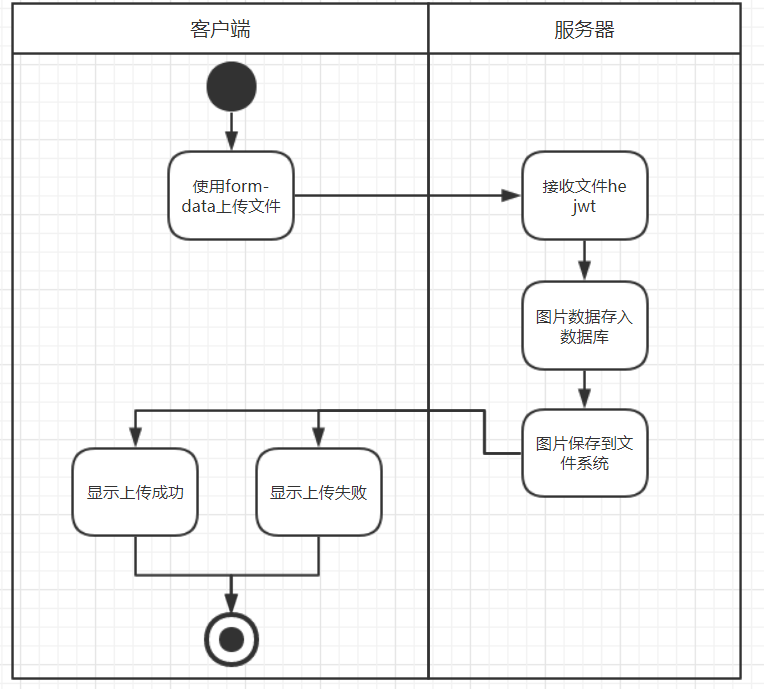
上传用户ID，token和图片ID，判断用户是否合法，合法则删除。

#### 根据图片ID获得图片/缩略图



根据图片ID返回图片，需判断图片是否存在

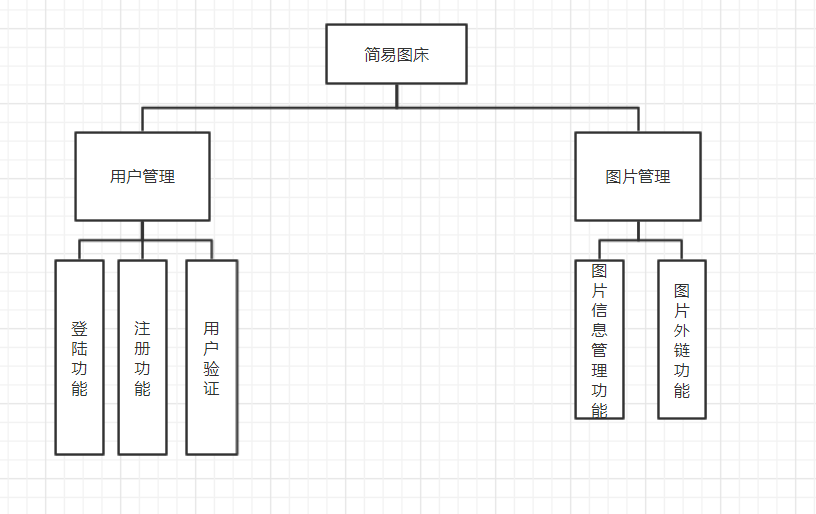
#### 图片上传



Android端提交二进制文件，服务

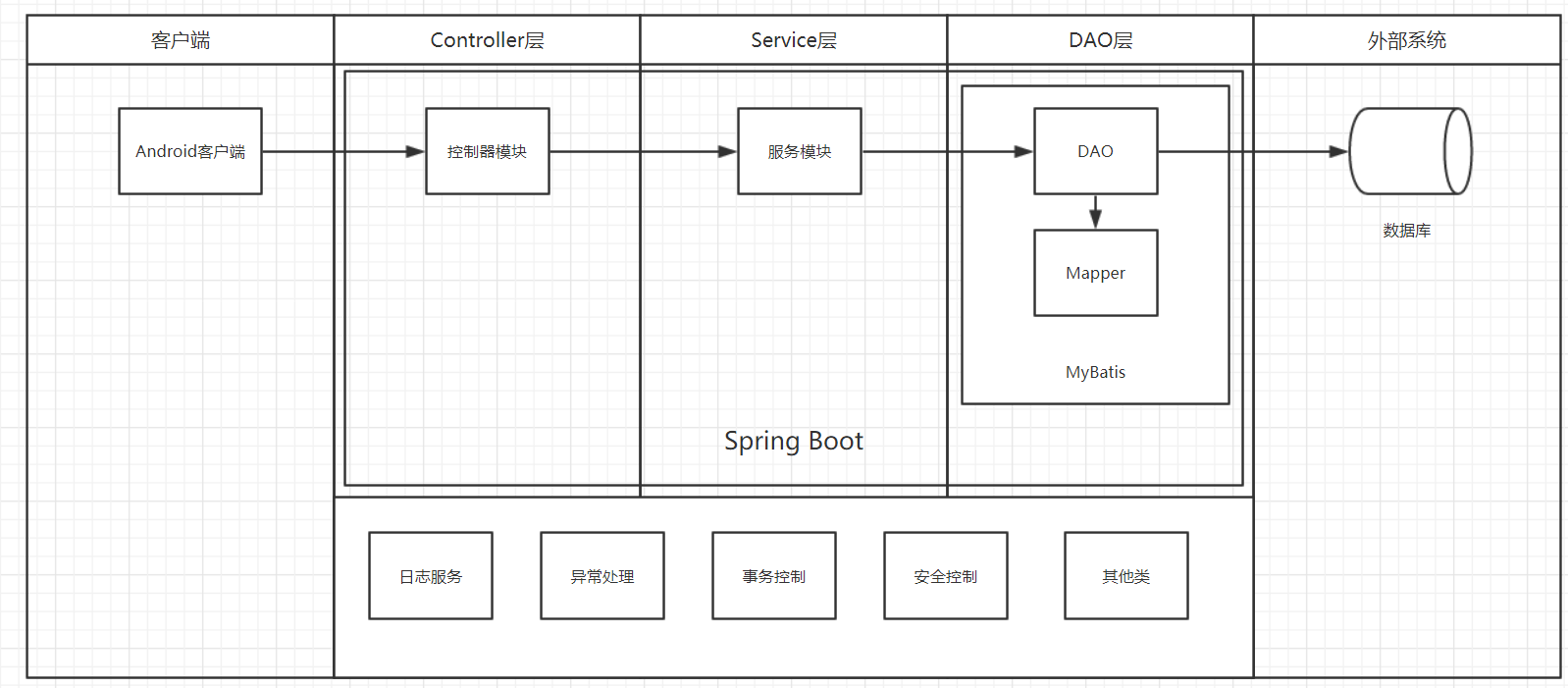
### 结构

用一览表及框图的形式说明本系统的系统元素（各层模块、子程序、公用程序等）的划分，扼要说明每个系统元素的标识符和功能，分层次地给出各元素之间的控制与被控制关系.

系统功能分为用户管理及图片管理功能。

用户管理主要负责登陆，注册以及api访问时用户验证的功能

图片管理负责图片的信息管理以及图片外链功能。



客户端：客户通过Android操作系统的客户端实现对建议图床功能的实现。

Controller层：对客户端发来的请求进行判断，并根据请求返回对应数据。

Service层：业务逻辑的具体实现。通过对DAO层的返回值进行组合，加工，最后将结果返回到Controller层。

DAO层：向业务层提供统一的资源访问接口。封装对数据库的访问操作

外部系统；MySQL数据库

## 接口设计

### 用户接口

**接口说明：**登陆接口

**请求方式：**POST /signin

**请求参数：**JSON或表单

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| username | str | 是 | 用户名 |
| password | str | 是 | 用户密码 |

**返回Header：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码（200失败，302成功） |

**返回数据：**JSON

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| jwtToken | str | 是 | jwt验证token，每次请求api都需要 |

**接口说明：**注册接口

**请求方式：**POST /signup

**请求参数：**JSON或表单

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| username | str | 是 | 用户名 |
| password | str | 是 | 用户密码 |
| email | str | 否 | 用户邮箱 |

**返回Header:**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码（200失败，302成功） |

**接口说明：**请求所有图片信息接口

**请求方式：**GET /allPicInfo

**请求参数：**无

**返回Header：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码 |

**返回数据：**JSON

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| picInfoList | picInfo[] | 是 | 包含图片信息 |

**示例：**

[

{

“picId”:”xasdfu3829”,

“picName”:”aaa.png”,

“picLastModify”:”2019-05-01”,

“picType”:”png”,

“picSize”:”1294”,

“userName”:”test”

“height”:”80”,

“width”:”90”

}

]

**接口说明：**请求某用户所有图片信息接口

**请求方式：**GET /picInfo/{username}

**请求参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| username | str | 是 | 用户名 |
| jwtToken | str | 是 | jwtToken用于验证，Header |

**返回Header：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码 |

**返回数据：**JSON

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| picInfoList | picInfo[] | 是 | 包含图片信息 |

**示例：**

[

{

“picId”:”xasdfu3829”,

“picName”:”aaa.png”,

“picLastModify”:”2019-05-01”,

“picType”:”png”,

“picSize”:”1294”,

“userName”:”test”

“height”:”80”,

“width”:”90”

}

]

**接口说明：**根据图片ID获得图片接口

**请求方式：**GET /pic/{picId}

**请求参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| picId | str | 是 | 图片ID |

**返回Header：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码 |

**返回数据：**byte[]

**接口说明：**根据图片ID获得缩略图接口

**请求方式：**GET /thumbnail/{picId}

**请求参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| picId | str | 是 | 图片ID |

**返回Header：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码 |

**返回数据：**byte[]

**接口说明：**删除图片接口

**请求方式：**DELETE /pic/{picId}

**请求参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| picId | str | 是 | 用户名 |
| jwtToken | str | 是 | jwtToken用于验证，放在Header |

**返回Header：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码 |

**接口说明：**上传图片

**请求方式：**POST /upload

**请求参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| file | MultipartFile | 是 | 上传文件 |

**返回Header**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 说明 |
| status | int | 是 | 状态码 |

### 外部接口

MySQL数据库接口：实现与MySQL的交互

### 内部接口

UserMapper：DAO层接口，获取基本的用户信息

PicInfoMapper：DAO层接口，获取基本的图片信息

UserService：Service层接口，实现用户登录，合法性查询等业务逻辑

PicInfoService：Service层接口，实现图片信息管理的业务逻辑

PicService：Servcie层接口，实现图片文件管理

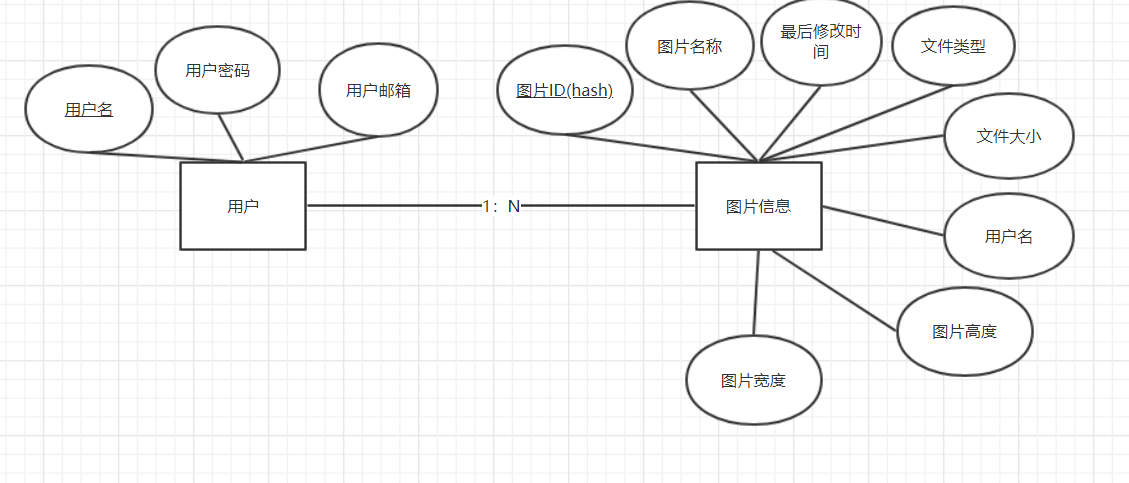
UserController：Controller层接口，处理User相关的请求

PicController：Controller层接口，处理Pic相关的请求

## 系统数据结构设计

### 逻辑结构设计要点

#### 系统E-R图



### 物理结构设计要点

给出本系统内所使用的每个数据结构中的每个数据项的存储要求，访问方法、存取单位、存取的物理关系（索引、设备、存储区域）、设计考虑和保密条件。

用户表（user）

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 备注 |
| user\_name | varchar(255) | 主键 |
| user\_pw | varchar(255) | NOT NULL |
| user\_mail | varchar(255) |  |

图片表(picInfo)

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 备注 |
| pic\_id | varchar(255) | 主键 |
| pic\_name | varchar(255) | NOT NULL |
| pic\_last\_modify | date | NOT NULL |
| pic\_type | varchar(255) | NOT NULL |
| pic\_size | bigint | NOT NULL |
| user\_name | varchar(255) | 外键 ref user(username) |
| width | int | NOT NULL |
| height | int | NOT NULL |

### 数据结构与程序的关系

通过MyBatis提供的Mapper进行数据库访问

### 系统出错处理设计

### 出错信息

用一览表的方式说朗每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。

|  |  |  |
| --- | --- | --- |
| 错误信息 | 含义 | 处理办法 |
| IOException | 文件读写出错 | 抛出异常并返回对应数据到客户端 |
| TypeMisMatchException | 文件非图片文件 | 抛出异常并返回对应数据到客户端 |

# 概要设计-客户端

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-07 | 迭代周期1的客户端概要设计 |  |  |
| 2019-05-23 | 迭代周期2的客户端概要设计 |  |  |
| 2019-06-07 | 迭代周期3的客户端概要设计 |  |  |

## 总体设计

### 需求规定

迭代周期1客户端主要实现基本的：

* 注册与登录功能前端界面、
* 图片浏览瀑布流列表
* 查看大图

迭代周期2客户端主要实现基本的

* 抽屉主页导航布局
* 图片上传功能
* 获取属于某用户上传的所有图片
* 图片缩略图功能
* 图片删除功能

迭代周期3客户端主要实现

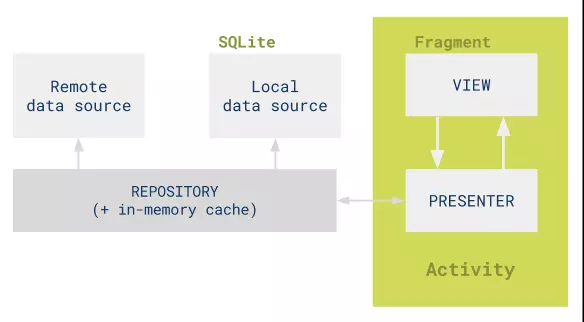
* 下拉刷新图片列表
* 改变图片列表瀑布流布局
* 保存原图
* 分享图片
* 编辑图片

### 运行环境

Android 7.0及以上系统

### 结构

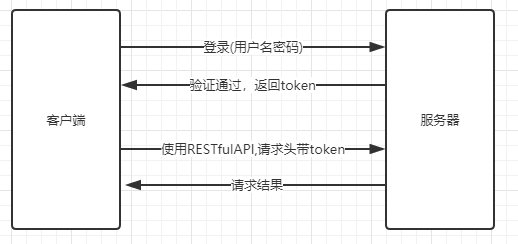
1. Android客户端开发架构采用Google给出的mvp示例模式,大体结构如下:

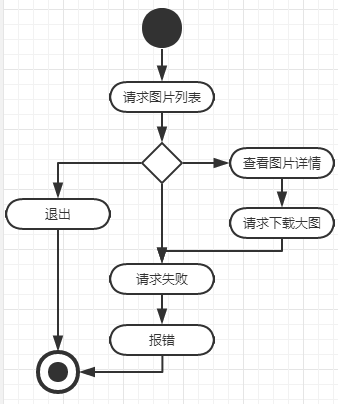


按功能点分类，每个需求功能点由一组model-presenter-view实现。

### 基本设计概念和处理流程

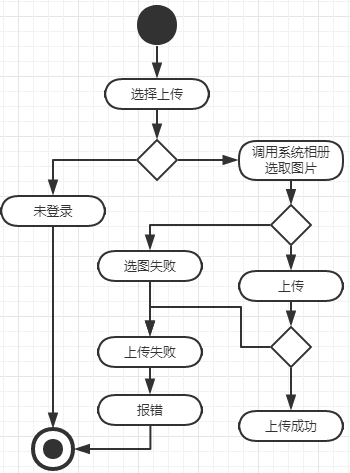
#### 迭代1

* 前后端交互接口采用RESTful风格设计api接口
* 登录采用token机制去维持客户端登录状态:
* 客户端获取图片功能基本处理流程如下:

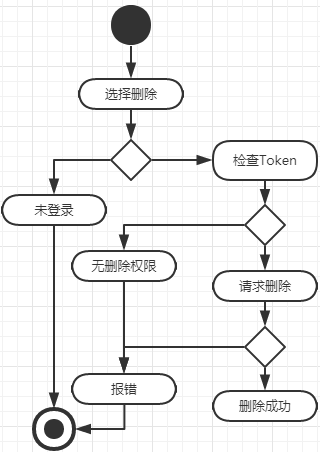


#### 迭代2

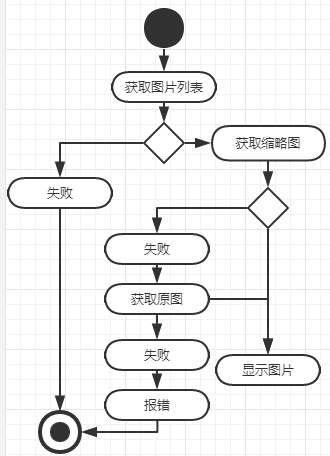
* 图片上传基本流程如下:



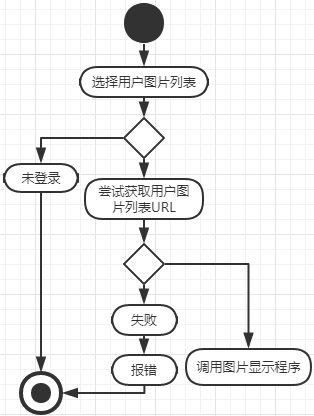
* 图片删除基本流程如下:



* 加载缩略图基本流程如下:

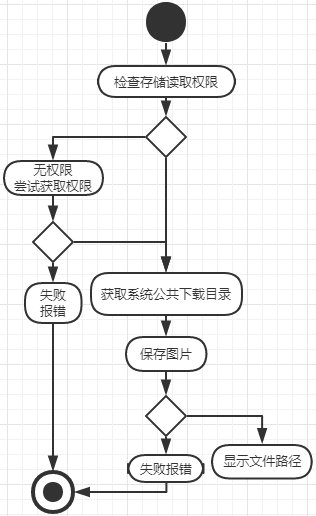


* 获取属于用户的图片列表

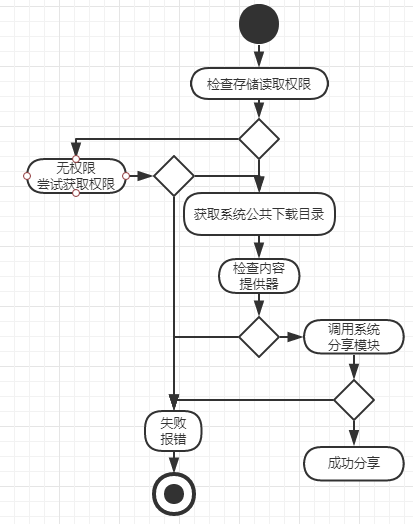


#### 迭代3

* 保存原图基本流程:



* 分享图片基本流程:



* 编辑图片基本流程如下:



## 接口设计

### 内部接口

#### 迭代1

* 每个Presenter都应实现的公共接口BasePresenter:

void start();

* 每个View都应实现的公共接口BaseView<T>:

boolean isActive();

void setPresenter(T presenter);

* 远程登录用的LoginDataSource <T>接口:

boolean isLoggedIn();

void logout();

T login(String username, String password);

* 登录后本地保存、获取、删除token的TokenRepositrory接口:

String getToken();

void saveToken();

void delToken();

* 图片列表Presenter(继承BasePresenter)自身应实现的接口:

void openPictureDetails();

* 图片列表View(继承BaseView)自身应实现的接口:

void showPictures();

void setProgressIndicator(boolean active);

void showLoadingPicturesError();

* 从拉取图片列表回调接口FetchPicsCallback:

void onDataLoaded(List<PicInfo> picList);

void onDataNotAvailable();

* 从拉取单张图片回调接口GetPicCallback:

void onDataLoaded(PicInfo pic);

void onDataNotAvailable();

* 图片数据源接口

void fetchPicList(String listURL, FetchPicsCallback callback);

void getPic(String picId, GetPicCallback callback);

void refresh();

#### 迭代2

* 传回调接口UpPicCallback：

**void onPicUploaded(response);**

**void onUploadFail(error);**

* 上传接口PicUpload:

**void upload(file, sourceid,jwt,UpPicCallback);**

* 删除回调接口DelPicCallback:

**void onDataLoaded(response);**

**void onDataNotAvailable(error);**

* 数据源接口PicDataSource添加删除

**void deletePic(picId, jwt, DelPicCallback);**

* 图片数据添加缩略图URL信息
* 新添加一个PicUserRemoteRepo 继承原有的图片仓库类实现数据源接口PicDataSource

#### 迭代3

* 图片编辑、删除、分享、保存的入口都设计在查看大图的UI界面的长按菜单中

故为查看大图模块的View- Presenter两对interface拥有以下接口函数

View：

**void showToast(msg);**

**void showShare(shareIntent); //用于调用系统分享视图**

**void showEdit(editIntent); //用于调用系统编辑图片视图**

Presenter：

**void savePic(PicInfo);**

**void sharePic(PicInfo);**

**void editPic(PicInfo);**

**void deletePic(PicInfo);**

**boolean isLoginIn();**

**boolean checkUser(username);**

* 下拉更新监听器的接口OnRefreshListener:

**void onRefresh();**

查看大图模块的View对象实现该接口

* 根据Android系统构建一个内容提供器以向其它应用发送图片

### 外部接口

* Android系统提供的api
* 服务端提供的api(见服务端概要设计)
* http网络通讯框架提供的api
* Google Material-Design组件包提供的Api（下拉刷新）

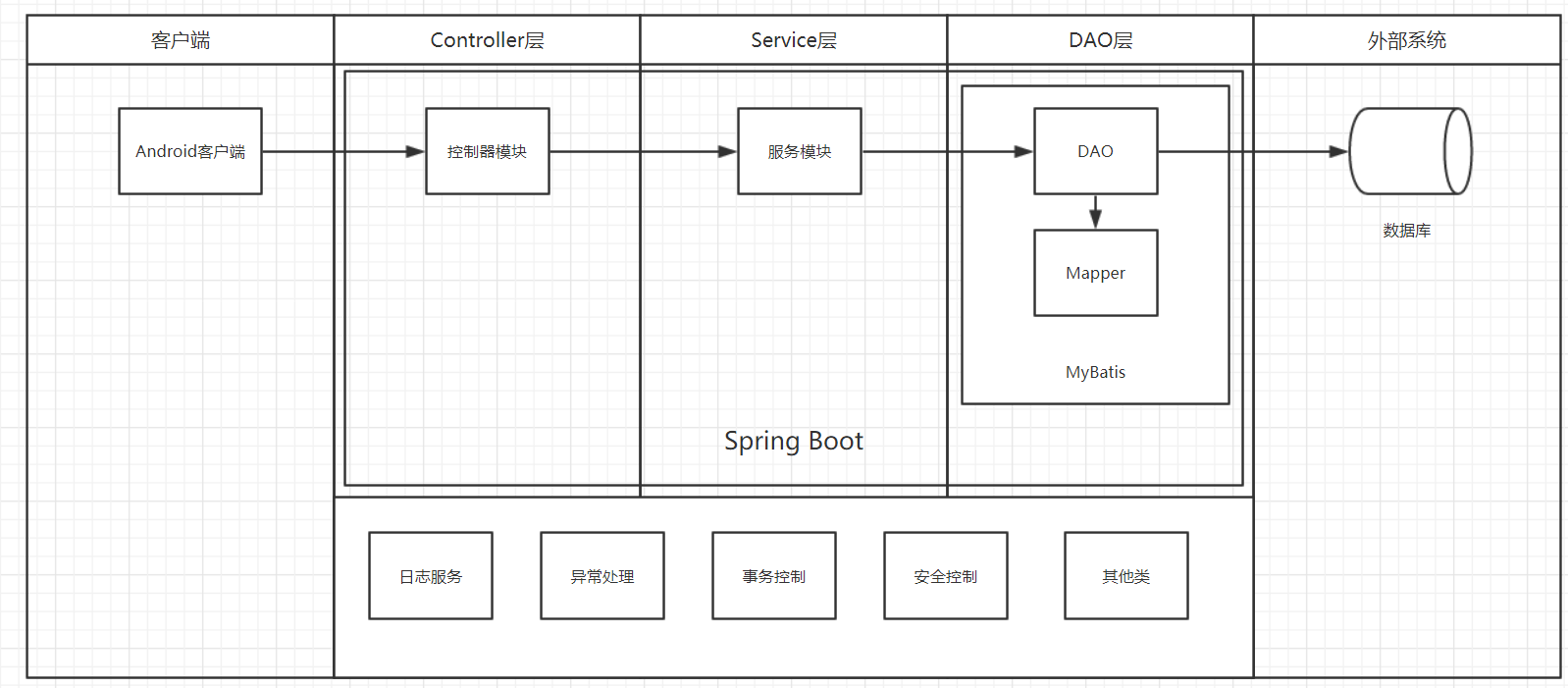
# 详细设计-服务端

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-09 | 迭代周期1的服务端详细设计 |  |  |
| 2019-05-28 | 迭代周期2的服务端详细设计 |  |  |
| 2019-06-11 | 迭代周期3的服务端详细设计 |  |  |

## 程序系统的结构

### 整体架构

本系统使用Spring boot+MyBatis+MySQL搭建后台架构



客户端：客户通过Android操作系统的客户端实现对建议图床功能的实现。

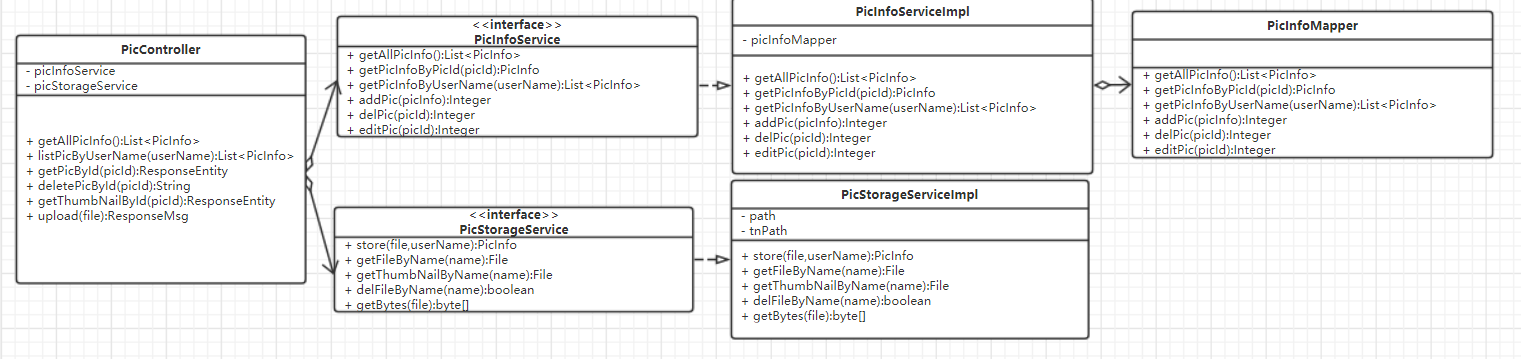
Controller层：对客户端发来的请求进行判断，并根据请求返回对应数据。

Service层：业务逻辑的具体实现。通过对DAO层的返回值进行组合，加工，最后将结果返回到Controller层。

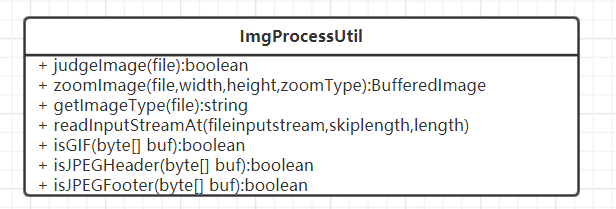
DAO层：向业务层提供统一的资源访问接口。封装对数据库的访问操作

外部系统；MySQL数据库

### 图片处理

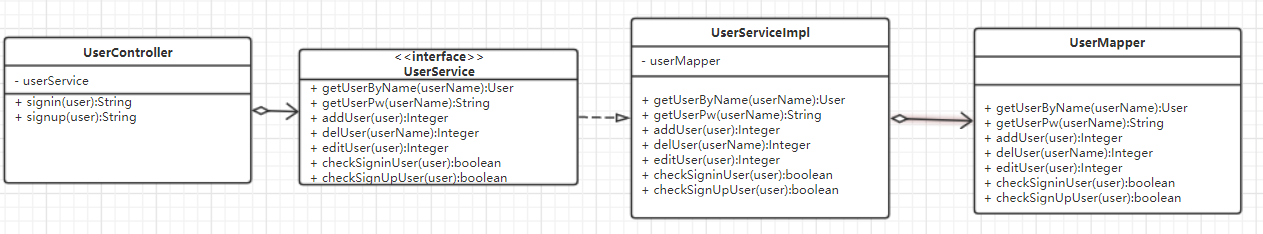


使用Controller-service-DAO传统的三层架构，PicController接收图片相关请求，PicInfoService提供图片信息处理，PicInfoMapper提供数据库接口。PicStorageService提供图片存储服务



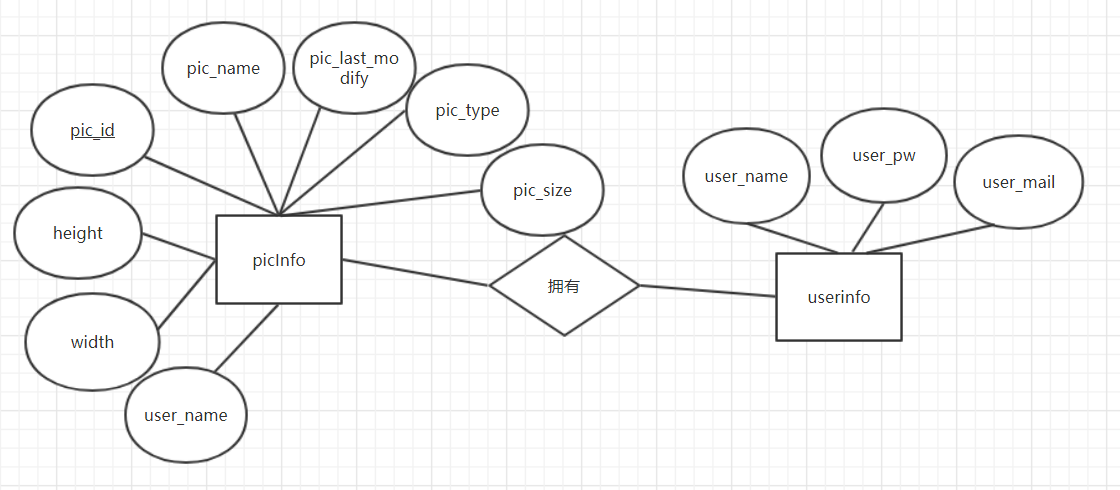
ImgProcessUtil提供了判断图片格式的功能

### 用户处理



同样Controller-service-DAO架构，UserController处理用户请求，UserService处理用户服务，UserMapper提供数据库接口

### 数据库逻辑结构



数据字典

picinfo

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 约束 |
| pic\_id | varchar(255) | primary key |
| pic\_name | varchar(255) |  |
| pic\_last\_modify | date |  |
| pic\_type | varchar(255) |  |
| pic\_size | bigint |  |
| user\_name | varchar(255) | ref userinfo(user\_name) |
| height | int |  |
| width | int |  |

Userinfo

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 约束 |
| user\_name | varchar(255) | primary key |
| user\_pw | varchar(255) |  |
| user\_mail | varchar(255) |  |

## PicController

### 程序描述

本程序是Spring boot架构的Controller部分，目的是接收android端提交与图片相关的请求并根据请求调用对应服务。Spring boot的Controller使用的是单实例，无状态，多线程的处理方式，其源程序常驻进程，自身支持了多线程的处理方式。故在设计该程序时直接使用了其设计方式。

### 功能

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 请求参数 | 功能目的 |
| getAllPicInfo | 无 | List<PicInfo>所有图片信息 | …/allPicInfo(get方法) | 输出所有图片信息 |
| listPicByUserName | username:用户名 | List<PicInfo>属于用户的图片信息 | …/picInfo/{username}(get方法) | 输出用户拥有的图片信息 |
| getPicById | picId:图片id | 二进制图片比特流 | …/pic/{picid}(get方法) | 返回获得所求图片对应比特流 |
| deletePicById | picId:图片id | 无 | …/pic/{picid}(delete方法) | 删除数据库中对应id图片 |
| getThumbNailById | picId:图片id | 二进制图片比特流 | …/thumbnail/{picId}(get方法) | 获得图片缩略图 |
| upload | file:二进制图片 | 状态码 | /upload(post方法) | 上传图片 |

### 输入项

函数名称：GetAllPicInfo

输入：无

函数名称：listPicByUserName

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 在数据库中存在即可 | get中携带参数 |

函数名称：getPicById

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picid | string | 在数据库中存在即可 | get中携带参数 |

函数名称：deletePicById

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 在数据库中存在即可 | delete中携带参数 |

函数名称：getThumbNailById

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 在数据库中存在即可 | get中携带参数 |

函数名称：upload

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| file | 二进制字节流 | 长度大于8字节，格式须为图片 | 使用form-data提交 |

### 输出项

函数名称：GetAllPicInfo

返回类型：json

返回格式

[

{

picId:,

picName:,

picLastModify:,

picType:,

picSize:,

userName:,

width:,

height:

}

]

函数名称：listPicByUserName

返回类型：json

返回格式

[

{

picId:,

picName:,

picLastModify:,

picType:,

picSize:,

userName:,

width:,

height:

}

]

函数名称：getPicById

返回类型：二进制比特流

返回格式：无

函数名称：deletePicById

返回类型：http状态码

返回格式：无

函数名称：getThumbNailById

返回类型：二进制比特流

返回格式：无

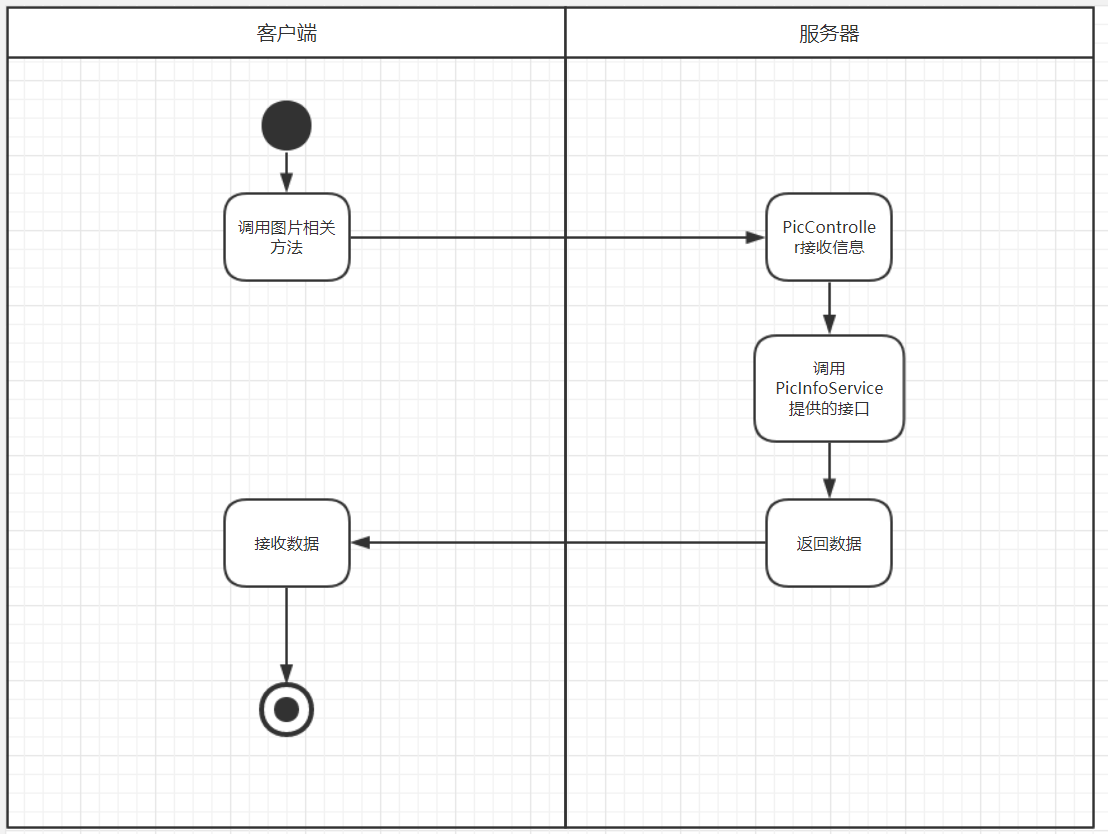
函数名称：upload

返回类型：http状态码

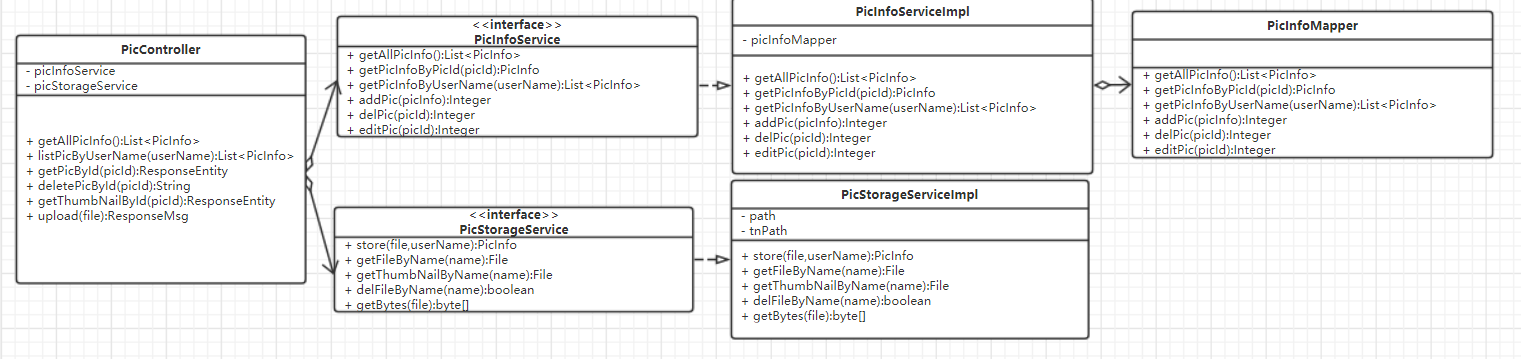
返回格式：无

### 流程逻辑

GetAllPicInfo，listPicByUserName，getPicById，deletePicById，getThumbNailById，upload



### 接口



在PicController中持有了两个接口，PicInfoService，PicStorageService以支持依赖注入

## PicInfoService

### 程序描述

本程序是Service层，实现了对图片信息操作业务逻辑（根据用户名批量获得图片信息，根据图片ID获得图片信息，增删改图片信息）的实现。本模块由Spring提供的依赖注入注入到PicInfoController中。由Spring管理其生命周期。

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| getAllPicInfo | 无 | List<PicInfo>所有图片信息 | 输出所有图片信息 |
| getPicInfoByPicId | picId:图片ID | PicInfo图片ID为picId的图片信息 | 根据图片id输出图片信息 |
| getPicInfoByUserName | userName:用户名 | List<PicInfo>：用户拥有的图片信息 | 获得用户拥有的图片信息 |
| addPic | picInfo：需要添加的图片信息 | Integer：1为成功，0为失败 | 添加图片信息 |
| delPic | picId:图片id | Integer：1为成功，0为失败 | 删除图片信息 |
| editPic | picInfo:需要修改的图片信息 | Integer：1为成功，0为失败 | 上传图片 |

### 输入项

函数名称：getAllPicInfo

输入：无

函数名称：getPicInfoByPicId

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 数据库中存在即可 | PicController传入参数 |

函数名称：getPicInfoByUserName

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| userName | string | 在数据库中存在即可 | PicController传入参数 |

函数名称：addPic

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picInfo | PicInfo | 内容不得为空 | PicController传入参数 |

函数名称：delPic

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 在数据库中存在即可 | PicController传入参数 |

函数名称：editPic

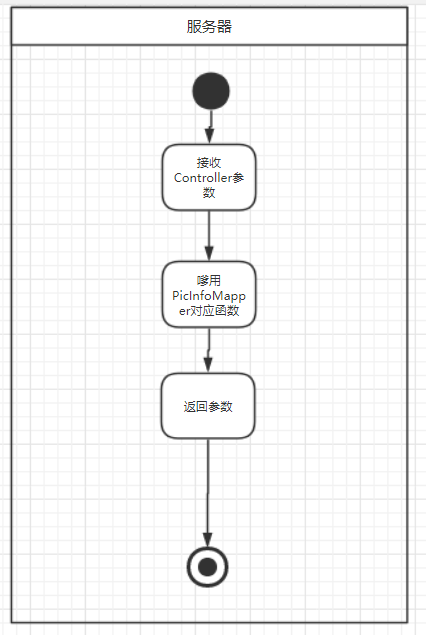
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picInfo | PicInfo | 内容不得为空 | PicController传入参数 |

### 输出项

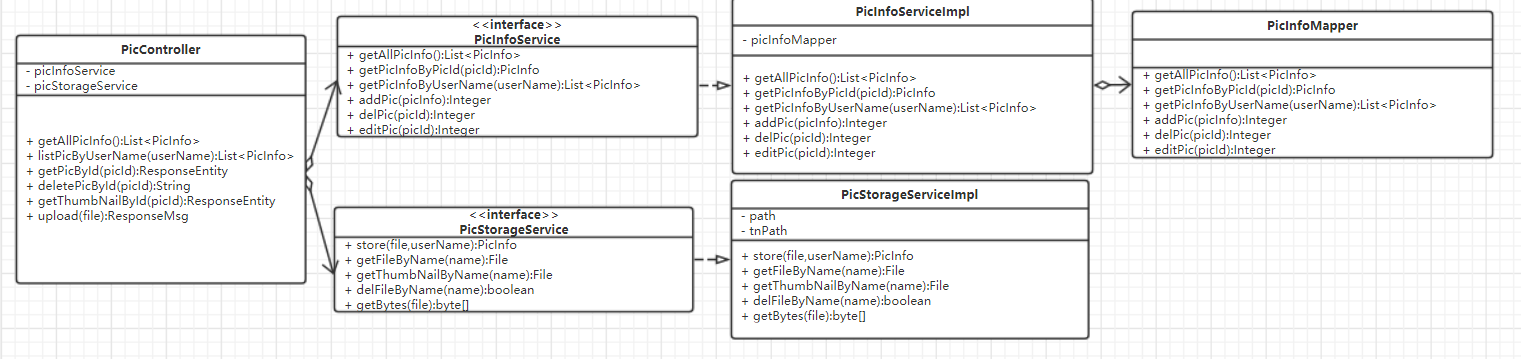
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| getAllPicInfo | List<PicInfo> | 无 | 参数传递 |
| getPicInfoByPicId | PicInfo | 无 | 参数传递 |
| getPicInfoByUserName | List<PicInfo> | 无 | 参数传递 |
| addPic | Integer | 无 | 参数传递 |
| delPic | Integer | 无 | 参数传递 |
| editPic | Integer | 无 | 参数传递 |

### 流程逻辑

getAllPicInfo，getPicInfoByPicId，getPicInfoByUserName，addPic，delPic，editPic



### 接口



在PPicInfoService中持有PicInfoMapper，其中PicInfoMapper为DAO层接口

## PicInfoServiceImpl

### 程序描述

为PicInfoService实现类，功能参见4PicInfoService

## PicInfoMapper

### 程序描述

本程序是三层架构的DAO层，主要负责与数据库进行简单交互。由于PicInfoService实现的即是对数据库的增删改查，所以本模块功能与PicInfoService相同。DAO层使用MyBatis与数据库Mysql交互。

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| getAllPicInfo | 无 | List<PicInfo>所有图片信息 | 输出所有图片信息 |
| getPicInfoByPicId | picId:图片ID | PicInfo图片ID为picId的图片信息 | 根据图片id输出图片信息 |
| getPicInfoByUserName | userName:用户名 | List<PicInfo>：用户拥有的图片信息 | 获得用户拥有的图片信息 |
| insert | picInfo：需要添加的图片信息 | Integer：1为成功，0为失败 | 添加图片信息 |
| delete | picId:图片id | Integer：1为成功，0为失败 | 删除图片信息 |
| update | picInfo:需要修改的图片信息 | Integer：1为成功，0为失败 | 上传图片 |

### 输入项

函数名称：getAllPicInfo

输入：无

函数名称：getPicInfoByPicId

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 数据库中存在即可 | PicController传入参数 |

函数名称：getPicInfoByUserName

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| userName | string | 在数据库中存在即可 | PicController传入参数 |

函数名称：insert

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picInfo | PicInfo | 内容不得为空 | PicController传入参数 |

函数名称：delete

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 在数据库中存在即可 | PicController传入参数 |

函数名称：update

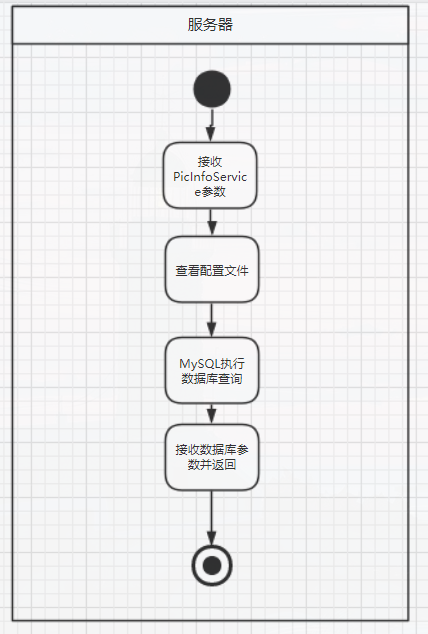
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picInfo | PicInfo | 内容不得为空 | PicController传入参数 |

### 输出项

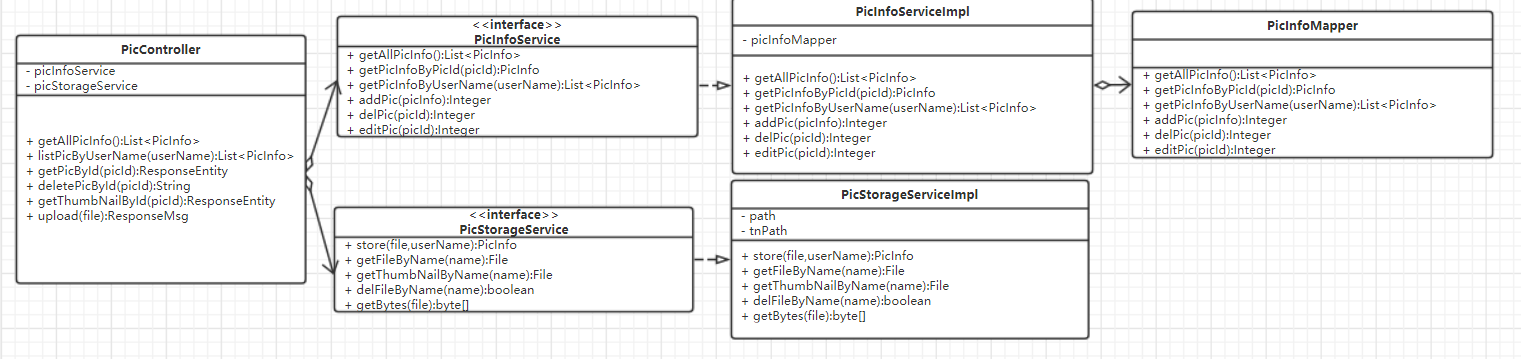
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| getAllPicInfo | List<PicInfo> | 无 | 参数传递 |
| getPicInfoByPicId | PicInfo | 无 | 参数传递 |
| getPicInfoByUserName | List<PicInfo> | 无 | 参数传递 |
| insert | Integer | 无 | 参数传递 |
| delete | Integer | 无 | 参数传递 |
| update | Integer | 无 | 参数传递 |

### 流程逻辑

getAllPicInfo，getPicInfoByPicId，getPicInfoByUserName，insert，delete，update



### 接口



## PicStorageService

### 程序描述

本程序是三层架构的service层，负责储存图片，存储缩略图，获得图片文件的功能。

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| store | file:文件  userName：用户名 | PicInfo图片信息 | 存储图片 |
| getFileByName | name:图片ID | file:文件 | 根据图片ID获得文件信息 |
| getThumbNailByName | name:图片ID | file:文件 | 根据图片ID获得文件信息 |
| delFileByName | name:图片ID | Boolean:删除是否成功 | 根据图片ID删除图片 |
| getBytes | file:文件 | byte[]:文件数组 | 获得文件比特流 |

### 输入项

函数名称：store

输入：

file:文件

userName：用户名

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| file | File | 合法文件 | PicController传入参数 |
| userName | string | 长度>0 |  |

函数名称：getFileByName

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 无 | PicController传入参数 |

函数名称：getThumbNailByName

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 无 | PicController传入参数 |

函数名称：delFileByName

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| picId | string | 数据库中须有该文件 | PicController传入参数 |

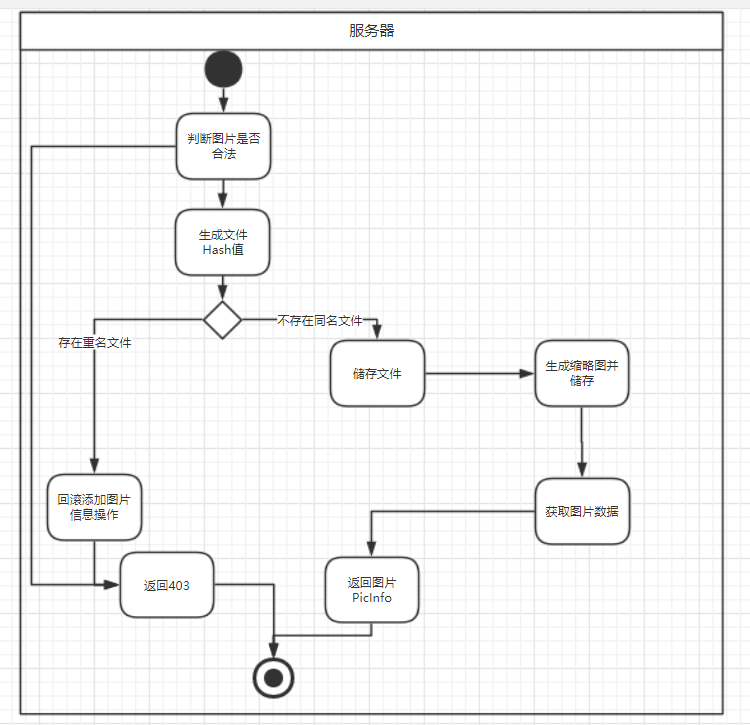
函数名称：getBytes

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| file | File | 文件需合法 | PicController传入参数 |

### 输出项

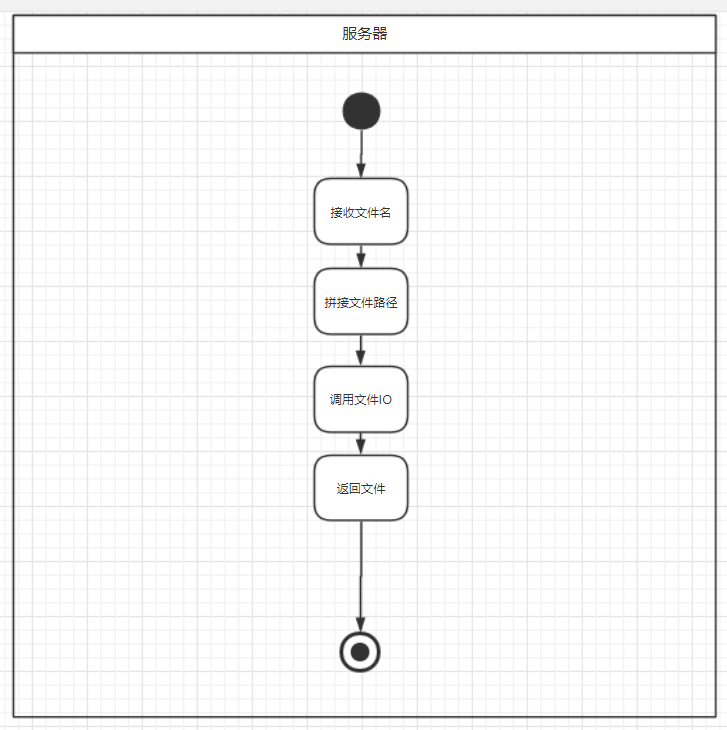
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| store | PicInfo | 无 | 参数传递 |
| getFileByName | file | 无 | 参数传递 |
| getThumbNailByName | file | 无 | 参数传递 |
| delFileByName | Boolean | 无 | 参数传递 |
| getBytes | byte[] | 无 | 参数传递 |

### 流程逻辑



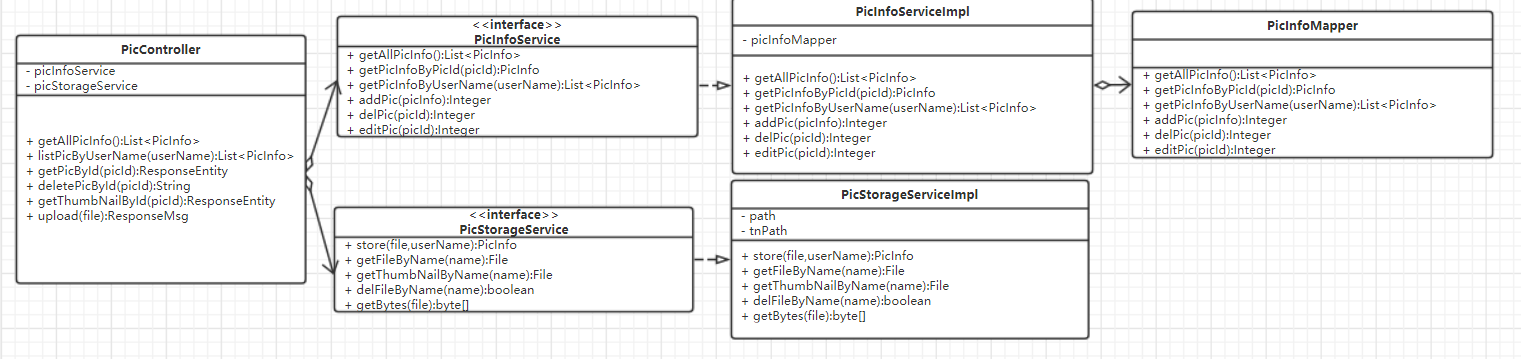
Store存储时实现判断图片是否合法（是否为png，jpg，gif文件，使用标志位而非后缀）判断是否存在同名文件。如果出现上述问起或其他IO问题，返回403，否则我们生成图片的Hash值（MD5），然后使用thumbnails生成缩略图并返回图片数据

getFileByName，getThumbNailByName，delFileByName



这三个函数都是读取文件的，故可以直接使用

### 接口



## PicStorageServiceImpl

### 程序描述

本程序是PicStorageService的实现类，具体功能参考7 PicStorageService

## UserController

### 程序描述

本程序是Spring boot架构的Controller部分，目的是接收android端提交与用户相关的请求并根据请求调用对应服务。本层实现了登陆，注册，注销功能，

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| signin | username:用户名  password:密码 | String：jwtToken | 登陆 |
| signup | username:用户名  password:密码 |  | 注册 |

### 输入项

函数名称：signin

输入：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | post参数输入 |
| password | string | 非空 | post 参数输入 |

函数名称：signup

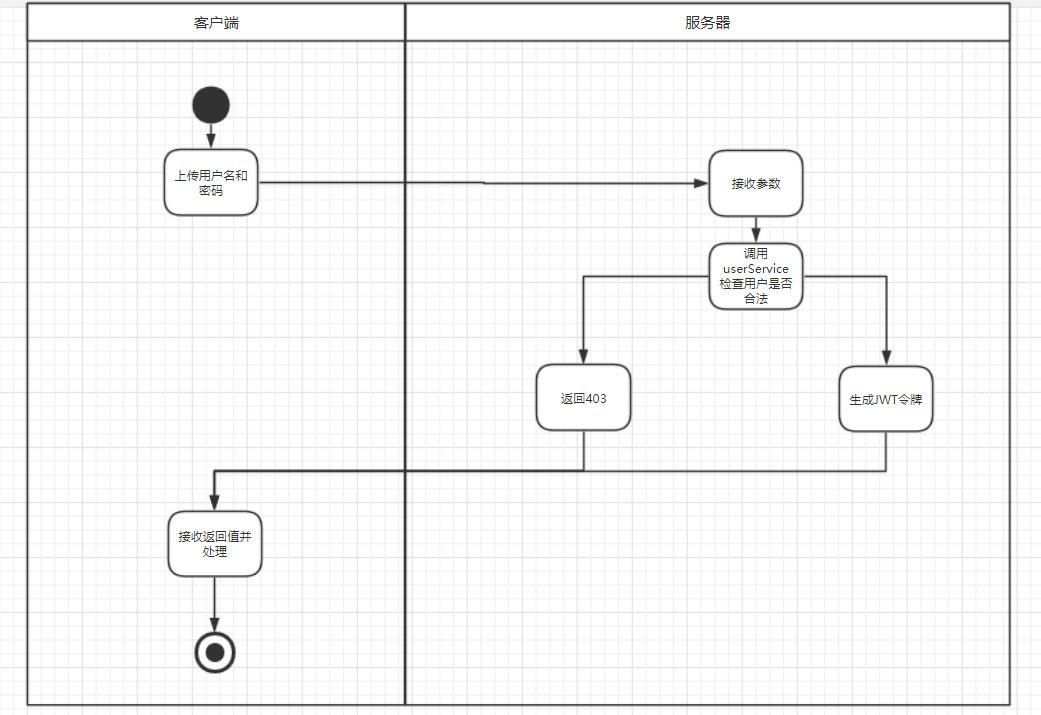
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | post参数输入 |
| password | string | 非空 | post参数输入 |

### 输出项

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| signin | string | 无 | 参数传递 |
| signup | 无 | 无 | 无 |

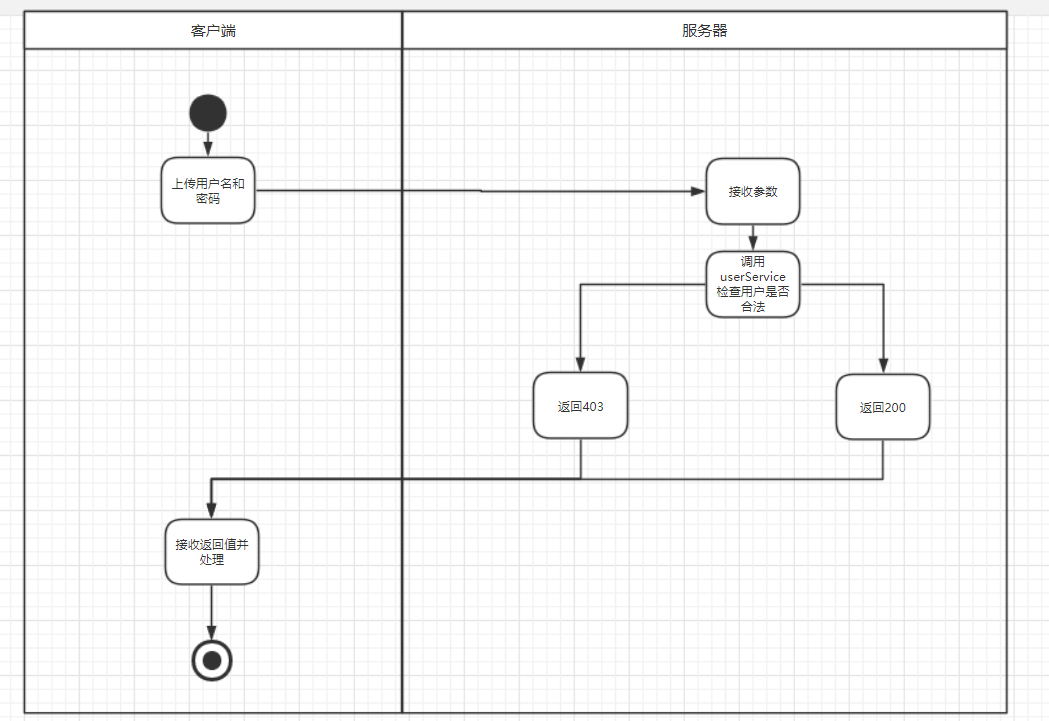
### 流程逻辑

Signin



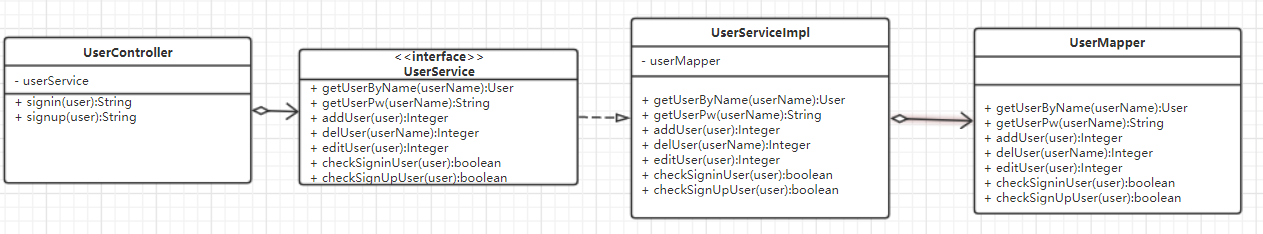
登陆时检查用户名密码是否匹配，用户是否合法，如果非法，返回403，如果合法则返回JWT令牌

Signup



注册时检查用户名是否已经存在即可

### 接口



## UserService

### 程序描述

本程序是Spring boot架构的Service部分，目的是检查用户合法性并实现增删改查用户功能

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| getUserByName | username:用户名 | User：用户 | 通过用户名获得用户 |
| getUserPw | username:用户名 | string：用户密码 | 通过用户名获得密码 |
| addUser | user：用户信息 | Integer：sql执行返回值 | 添加用户 |
| delUser | username:用户名 | Integer：sql执行返回值 | 删除用户 |
| editUser | user：用户信息 | Integer：sql执行返回值 | 编辑用户 |
| checkSigninUser | user：用户信息 | Boolean：登陆用户合法性 | 检查登陆用户合法性 |
| checkSignUpUser | user：用户信息 | Boolean：注册用户合法性 | 检查注册用户合法性 |

### 输入项

函数名称：getUserByName

输入：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |

函数名称：getUserPw

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |

函数名称：addUser

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

函数名称：delUser

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |
| password | string | 非空 | 参数传递 |

函数名称：editUser

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

函数名称：checkSigninUser

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

函数名称：checkSignUpUser

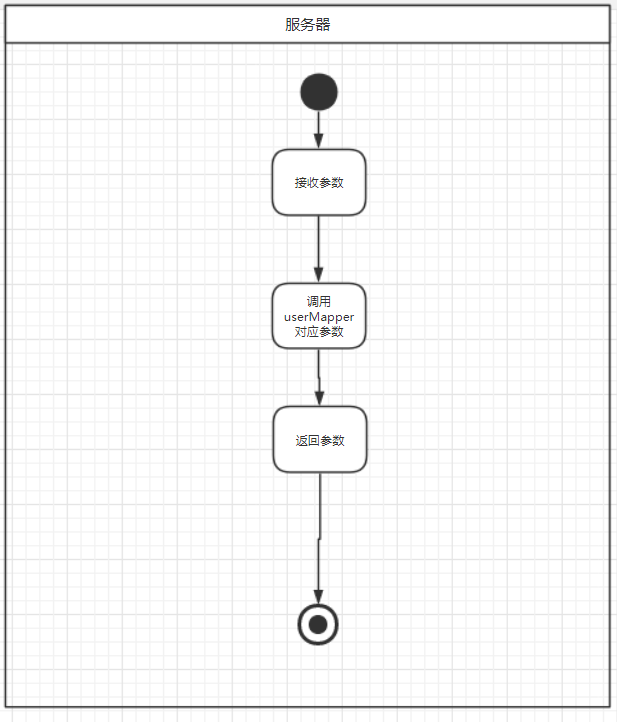
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

### 输出项

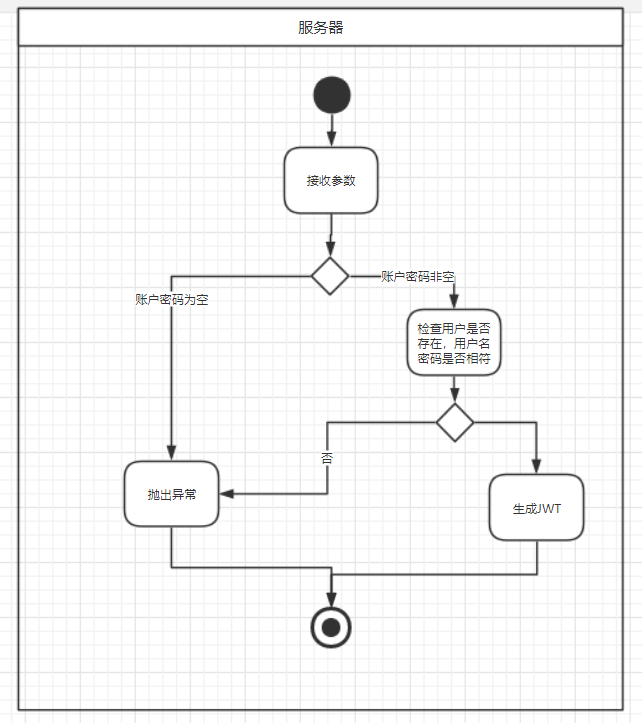
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| getUserByName | string | 无 | 参数传递 |
| getUserPw | string | 无 | 参数传递 |
| addUser | Integer | 0-1 | 参数传递 |
| delUser | Integer | 0-1 | 参数传递 |
| editUser | Integer | 0-1 | 参数传递 |
| checkSigninUser | boolean | true/false | 参数传递 |
| checkSignUpUser | boolean | true/false | 参数传递 |

### 流程逻辑

getUserByName, getUserPw, addUser, delUser, editUser

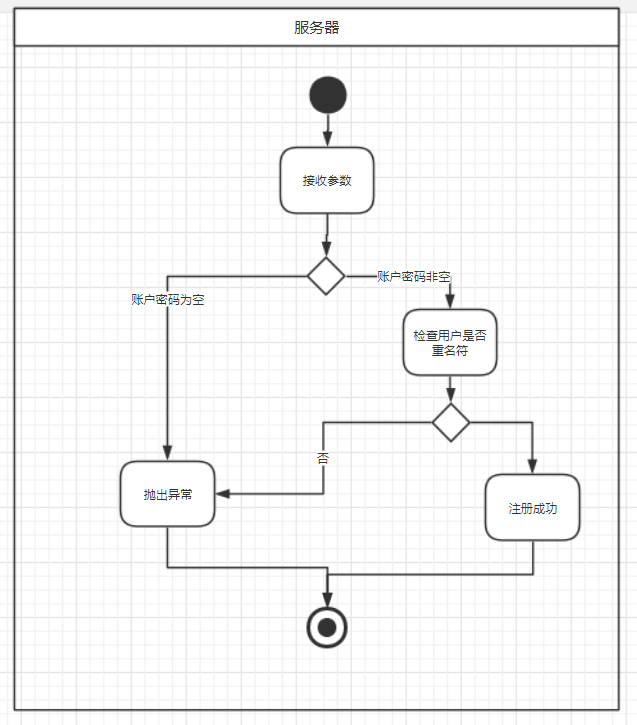


上述函数是调用DAO层函数的中间过程，直接调用DAO层



checkSigninUser

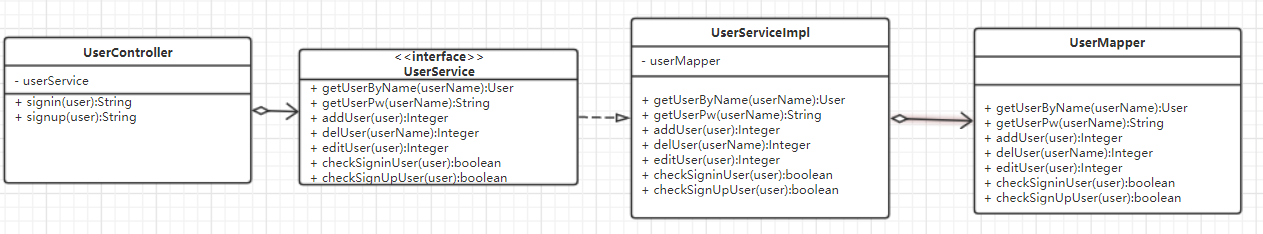
检查帐户密码是否为空，用户是否存在，账户密码是否相符，如果成功返回JWT，否则抛出异常



checkSignUpUser

检查账目密码是否维空，检查用户是否重名，成功则注册成功，否则抛出异常

### 接口



## UserServiceImpl

### 程序描述

本程序是UserService接口的实现，详细功能见10 UserService

## UserMapper

### 程序描述

本程序是Spring boot的DAO层，实现有关user的增删改查，负责与mysql交互

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| getUserByName | username:用户名 | User：用户 | 通过用户名获得用户 |
| getUserPw | username:用户名 | string：用户密码 | 通过用户名获得密码 |
| insert | user：用户信息 | Integer：sql执行返回值 | 添加用户 |
| delete | username:用户名 | Integer：sql执行返回值 | 删除用户 |
| update | user：用户信息 | Integer：sql执行返回值 | 编辑用户 |
| checkSigninUser | user：用户信息 | Boolean：登陆用户合法性 | 检查登陆用户合法性 |
| checkSignUpUser | user：用户信息 | Boolean：注册用户合法性 | 检查注册用户合法性 |

### 输入项

函数名称：getUserByName

输入：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |

函数名称：getUserPw

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |

函数名称：insert

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

函数名称：delete

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |
| password | string | 非空 | 参数传递 |

函数名称：update

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

函数名称：checkSigninUser

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

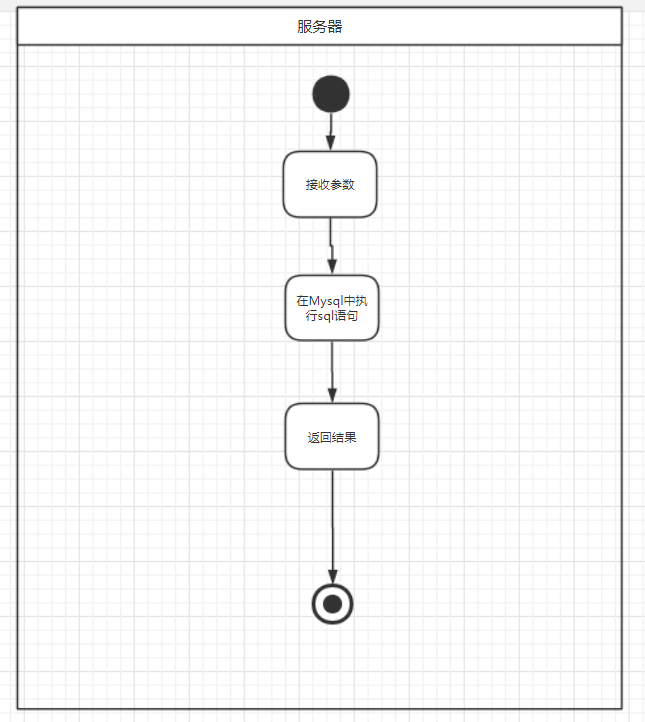
函数名称：checkSignUpUser

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| user | User | 非空 | 参数传递 |

### 输出项

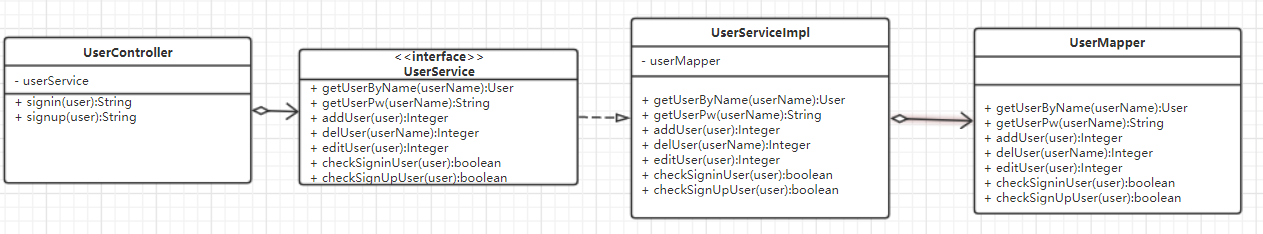
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| getUserByName | string | 无 | 参数传递 |
| getUserPw | string | 无 | 参数传递 |
| insert | Integer | 0-1 | 参数传递 |
| delete | Integer | 0-1 | 参数传递 |
| update | Integer | 0-1 | 参数传递 |
| checkSigninUser | boolean | true/false | 参数传递 |
| checkSignUpUser | boolean | true/false | 参数传递 |

### 流程逻辑



由于该层是DAO层，所以所有函数直接映射到sql语句

### 接口



## JwtUtil

### 程序描述

本程序是生成JWT密匙部分，使用HS512加密时间戳。

### 功能

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名 | 输入 | 输出 | 功能目的 |
| generateToken | username:用户名 | token：认证token | 获得token |
| parseToken | token： 认证token | Claims：负荷 | 认证token |

### 输入项

函数名称：generateToken

输入：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| username | string | 非空 | 参数传递 |

函数名称：parseToken

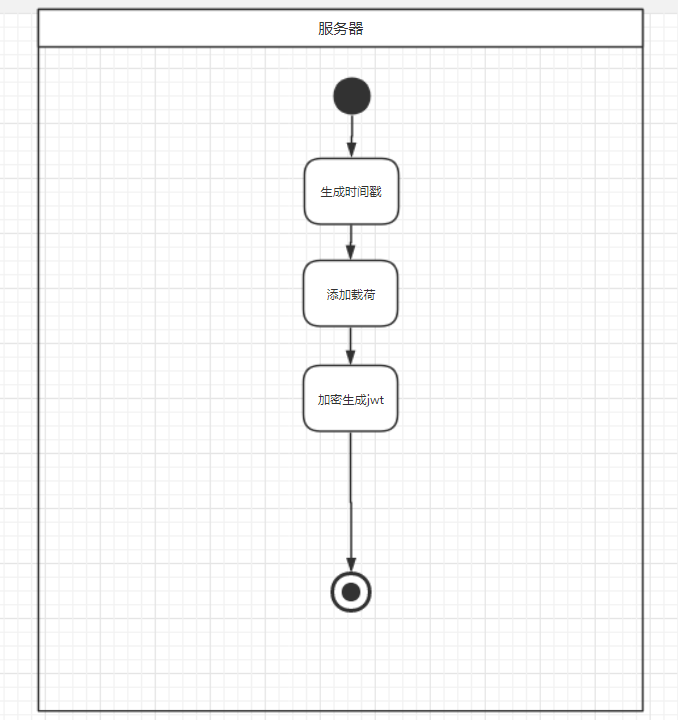
|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输入方式 |
| token | string | 非空 | 参数传递 |

### 输出项

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 数据类型 | 有效范围 | 输出方式 |
| generateToken | string | 无 | 参数传递 |
| parseToken | Claims | 无 | 参数传递 |

### 流程逻辑

generateToken



Jwt利用非对称加密实现生成token。本程序对时间戳进行了加密

# 详细设计-客户端

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-09 | 迭代周期1的客户端详细设计 |  |  |
| 2019-05-28 | 迭代周期2的客户端详细设计 |  |  |
| 2019-06-11 | 迭代周期3的客户端详细设计 |  |  |

## 迭代1

### 注册登录模块

登录注册模块注重的是User信息的获取与储存，其依赖模块结构如下图。

UserToken是一个持有User信息的POJO，User信息在系统中传递都是以UserToken对象的形式。模块主要由解耦的三个部分组成：

本地数据库存取部分：AppDataBase UserTokenDAO接口 UserTokenDAO

UI界面：LoginActivity 通过Injection注入的LoginRepositrory响应注册、登录操作。

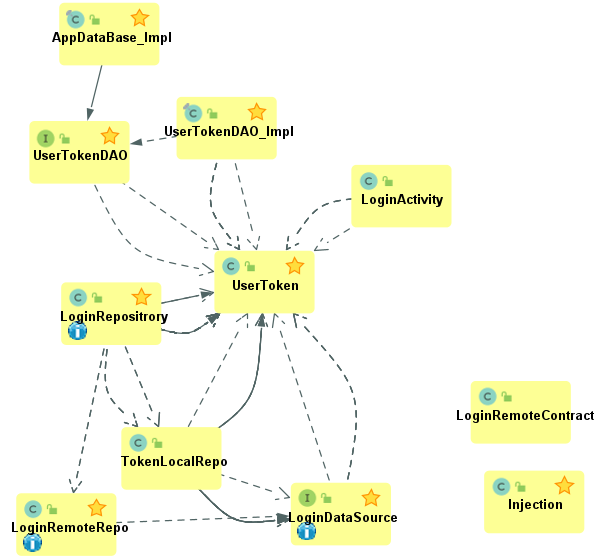
负责信息获取、缓存、更新、删除等操作的仓库部分：

LoginRemoteContract 定义远程客户端Api的各种接口给http框架实现

LoginRemoteRepo通过由http框架提供注入的http客户端提供远程信息获取

TokenLocalRepo通过系统注入的数据库DAO实现提供本地持久化

LoginRepositrory持有一个TokenLocalRepo与LoginRemoteRepo负责将用户信息缓存在内存，以及在适当的时机从远程更新信息、将信息写入本地数据库，向上提供注册、登录逻辑。



### 图片列表模块

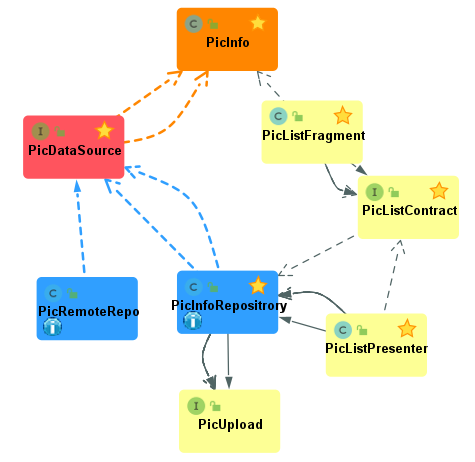
图片列表模块依照MVP的模式实现，

PicListFragment实现UI通过PicListContract接口与Presenter解耦、通过PicDataSourece接口与PicInfoRepositrory、PicRemoteRepo解耦。

PicRemoteRepo负责获取远程图片列表信息。

PicInfoRepositrory通过Injection注入一个PicRemoteRepo对象，同时通过PicUpload接口实现上传服务（本迭代未具体实现该功能），向PicListPresenter提供图片列表信息获取服务。

PicInfoRepositrory会适当的缓存、更新图片列表信息。

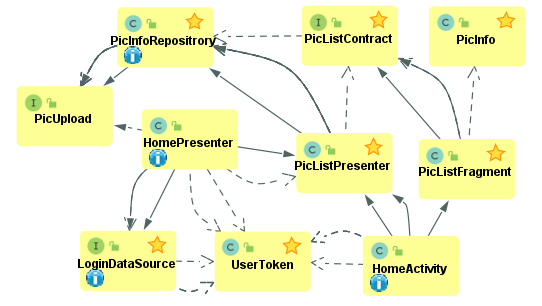


## 迭代2

### 导航主页面

导航页面包含各种功能界面的入口，如显示所有图片、登录/注册功能、上传功能、显示本用户上传的图片、设置等。而HomePresenter的主要任务是在启动时读取持久化数据检查登录状态，以及将各种功能分发给正确的模块/对象。

其中上传功能的具体实现由PicInfoRepository承担(实现PicUpload接口)，该对象并不与HomePresenter直接耦合，而是通过Injection类注入到HomePresenter持有的PicUpload引用。



### 查看大图模块

查看大图模块是从图片列表模块跳转而来，显示某张图片的原图

本迭代周期的两个重要功能需求：

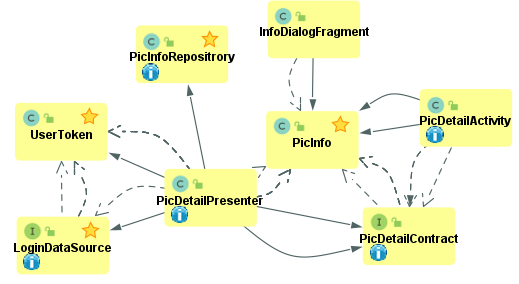
* 查看图片详情
* 删除图片

的入口将设计为大图UI中长按弹出的菜单选项。

在查看大图模块依照MVP模式实现，通过PicDetailContract接口实现了PicDetaiActivty与PicDetailPresenter的解耦。

PicDetailPresenter负责获取图片信息、加载源图片、以及根据外部注入的向PicInfoRepository发出删除请求（在删除前会通过LoginDataSource获取当前用户并检查是否符合权限），而Repository将以恰当的方式向远程服务器发起删除请求，并更新各数据信息缓存。

InfoDialogFragment是接受外部注入的Picinfo对象并展示图片详细信息的浮动视图，通过PicDetaiActivty的菜单选项触发显示。

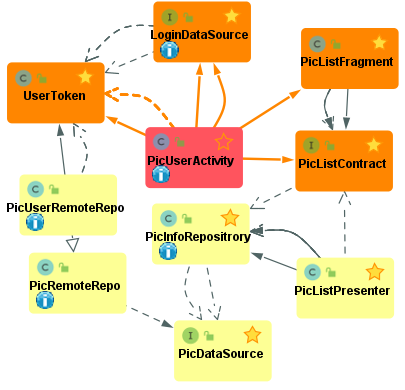


### 用户图片列表模块

图片列表模块依照MVP模式实现，本模块主体部分复用了原图片列表模块的PicListFragment-PicListPresenter的组合，PicUserActivity在此处充当控制器的作用。

不同在于注入的远程数据源依赖，PicUserRemoteRepo继承了PicRemoteRepo，构造时需要传入一个UserToken对象。

PicUserActivity通过Injection获取一个LoginDataSource的实现，并获取当前登录对象信息（UserToken对象），而后将该对象传递给Injection以获取一个PicInfoRepositrory对象并注入到PicListPresenter，该PicInfoRepositrory持有的一个远程数据源PicDataSource引用，该引用的依赖由Injection构造PicUserRemoteRepo并传入。



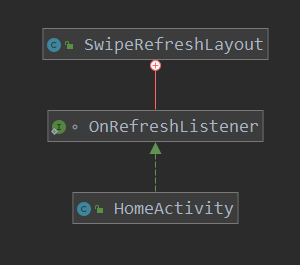
## 迭代3

### 导航主页面

迭代3更新：

* 下拉刷新图片列表

下拉刷新需要引入MaterialDesign构件包中的下拉刷新视图，HomeActivity需要实现该构件中的onRefreshListerner接口来实现下拉更新动画的控制。



* 改变图片列表瀑布流布局

该功能通过为图片列表视图添加setLayoutSpanCount(int)接口实现，在HomeActivity触发相应点击事件时调用该接口。

### 查看大图模块

迭代3更新了以下功能需求：

* 保存原图
* 分享图片
* 编辑图片

需要做出的变动有：在浮动菜单中添加功能入口、

为PicDetailPresenter添加实现如下接口（通过PicDetailContract）：

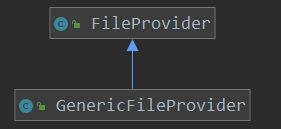
void savePic(PicInfo picInfo);

void sharePic(PicInfo picInfo);

void editPic(PicInfo picInfo);

除此之外还需要继承实现系统的文件内容提供器并在应用配置中声明

以实现向其它应用提供图片文件



# 测试分析报告

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 修改内容 | 执笔 | 审查 |
| 2019-05-14 | 制定迭代周期1的项目测试报告 |  |  |
| 2019-06-03 | 制定迭代周期2的项目测试报告 |  |  |
| 2019-06-12 | 制定迭代周期3的项目测试报告 |  |  |

## 背景

软件名称：《android简易图床》

测试环境：在win10系统中，运用Junit对分模块进行逐一测试

运行环境：安卓端

预计出现的误差：在前端和后端的通讯过程可能会出现一些不可预料的故障

解决方法：在整个软件项目完成后用安卓前端进行再一次测试

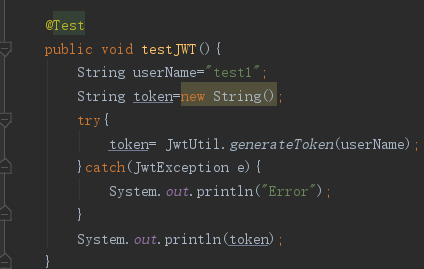
## 测试概要

|  |  |  |
| --- | --- | --- |
| 功能 | 单元设计 | 单元测试 |
| Token获取 | 登录时候用户保护用户信息自动根据时间等因素生成 | 假定一个UserName来获取token，将测试结果与预期结果进行对比 |
| 注册功能 | 在客户端界面向系统发送注册操作请求，并调用方法进行处理，处理成功后反馈给客户 | 模拟客户端界面向系统发送多组添加用户命令，输入数个符合或者不符合输入要求的username和Password（等服务器开放后进行客户端层面的再次测试） |
| 显示图片 | 请求数据库中所有公开的图片，并进行自我放缩 | 重复多次关闭打开APP，观察是否能够正常显示图片 |
| 登录功能 | 向系统发送客户的登录操作请求并调用方法进行处理，处理成功后反馈给客户 | 模拟从客户端界面使用多组设计好的账号密码进行登录（等服务器开放后进行客户端层面的再次测试） |
| 图片删除 | 向系统发送删除图片请求，在判断图片所属和发送者是否拥有删除权限后调用方法进行处理，处理成功或者失败后反馈客户 | 模拟拥有不同图库的数个客户，选择或者批量选择删除图库中的拥有的照片 |
| 图片上传 | 向系统发送上传图片请求，在判断图片所属和发送者是否拥有上传权限后调用方法进行处理，处理成功或者失败后反馈客户 | 模拟客户大单张以及批量上传不同大小，不同格式的图片。 |
| 获取用户图片列表 | 用户登录后会自动向系统发送现实用户自己图片列表的请求，系统判断权限和ID后进行返回 | 模拟多个客户向后台发送获取图片列表请求 |
| 图片缩略图展示 | 将从数据库返回来的图片进行缩略处理，显示在界面上的图片改为缩略图，点击缩略图旁边的按钮图加载请求 | 模拟多个客户向后台发送获取图片列表请求 |
| 分享图片 | 向服务端发送请求，调用方法获得对应图片的二进制流再调用方法转换成其他应用端可以识别的uri | 对多张不同格式的图片进行分享，并且分享到不同的应用端 |
| 保存图片 | 动态对本地文件管理申请读取和写入文件请求，从后端数据库下载图片写入文件 | 实体APP测试，对多张不同格式的图片进行保存到本地操作 |
| 图片详细信息 | 向系统发送获取对应图片的详细信息请求，调用方法处理完成后进行返回显示 | 对图库内的多张不同格式的图片进行详细信息查看，观察是否有信息错误 |
| 复制外链 | 生成一个能够查看图片的链接 | 对图库内的多张图片进行进行外链，观察是否能过够在电脑端打开 |
| 编辑图片 | 发出编辑请求，调用方法唤醒手机自带的编辑系统进行编辑。 | 对多张图片进行编辑 |
| 下拉刷新 | 对后台系统重新发送获取图片请求。 | 多个用户在同一时刻上床图片，下拉刷新后观察是否显示了所有图片 |

## 测试结果及发现

### Token获取

Jnuit代码：



|  |
| --- |
| **测试用例 3.11: Token获取** |
| 综述:   1. 目的：保证用户登陆时的安全性 |
| 步骤：  1.打开测试APP  2.另Username = test |
| 期望输出:   * 1. 一串由大小写英文和下划线等组成的随机数列 |
| 实际输出:  正常 |

### 注册功能

Jnuit代码：



测试用例

|  |
| --- |
| **测试用例 3.21: 注册功能** |
| 综述:   1. 目的：用户能够自我注册账号密码 2. 条件：注册所用的用户名密码皆不存在 |
| 步骤：  1.编写测试代码  2.模拟前台后后台发送注册（最后使用成品APP进行测试）  3.输入测试用例 Username=test2 Userpw=test2 |
| 期望输出:  注册成功 |
| 实际输出:  正常。C:\Users\lenovo\Desktop\软工课设\Documents\Tencent Files\1484405121\Image\C2C\66A10AA0E141373BCD56D9E7D01EC138.jpg |

|  |
| --- |
| **测试用例 3.22: 注册功能** |
| 综述:   1. 目的：用户能够自我注册账号密码 2. 条件：注册所用的用户名已经存在 |
| 步骤：  1.打开APP  2.测试用例为 Username=test2 Userpw=test2 |
| 期望输出:  注册失败 |
| 实际输出:  正确 |

|  |
| --- |
| **测试用例 3.23: 注册功能** |
| 综述:   1. 目的：用户能够自我注册账号密码 2. 条件：账号密码都为空 |
| 步骤：  1.打开APP  3.测试用例为 Username=“” Userpw=“” |
| 期望输出:  注册失败 |
| 实际输出:  错误，详情见缺陷报告[9.1.1](#_5.1.1) |

### 登录功能

|  |
| --- |
| **测试用例 3.31: 用户登录** |
| 综述:   1. 目的：用户能够通过正确的账号密码登录 2. 条件：账号密码已经存在系统中 |
| 步骤：  1.打开APP  2.测试用例为 Username=test2 Userpw=test2 |
| 期望输出:  登录成功 |
| 实际输出:  C:\Users\lenovo\Desktop\软工课设\Documents\Tencent Files\1484405121\Image\C2C\1CBA2CE8443FDB21E0A46EF0C28F2A07.jpg正常。 |

|  |
| --- |
| **测试用例 3.32: 用户登录** |
| 综述:   1. 目的：账号存在但是密码错误的情况下不能登录 2. 条件：账号密码已经存在系统中 |
| 步骤：  1.打开APP  2.测试用例为 Username=test2 Userpw=test3 |
| 期望输出:  登陆失败 |
| 实际输出:  正常。 |

|  |
| --- |
| **测试用例 3.33: 用户登录** |
| 综述:   1. 目的：验证账号不存在的情况下不能登录 2. 条件：测试用例的账号密码不存在后台数据库中 |
| 步骤：  1打开APP  2.测试用例为 Username=test3 Userpw=test3 |
| 期望输出:  登陆失败，提示账号密码错误 |
| 实际输出:  正常。 |

### 显示图片

|  |
| --- |
| **测试用例 3.41: 显示图片列表** |
| 综述:   1. 目的：可以通过客户端查看所有图片的信息，并且观察图片是否按照双盒瀑布流分布 |
| 步骤：   * 1. 进入客户端界面   2. 点击显示图片列表 |
| 期望输出:  获得系统所有图片列表，并且按照一定瀑布流排序 |
| 实际输出:  正常。 |

### 图片删除

|  |
| --- |
| **测试用例 3.51: 图片删除** |
| 综述:   1. 目的：验证用户是否能够正常单张删除自己图库内的照片 2. 条件：测试的用户账号内存在一张以上照片 |
| 步骤：  1.登录账号  2.选择单张照片进行删除 |
| 期望输出:  删除成功，用户图库内已经找不到删除的图片 |
| 实际输出:  正常。 |

### 图片上传

|  |
| --- |
| **测试用例 3.61: 图片上传** |
| 综述:   1. 目的：验证用户是否能够正常单张上传图片到自己图库内 2. 条件：事先准备不同大小，不同规格的图片（jpg,png,gif）。 |
| 步骤：  1.登录账号  2.选择手机内单张照片进行上传 |
| 期望输出:  上传成功，并且用户图片列表中已经存在上传的图片，并且能够点击查看 |
| 实际输出:  正常 |

### 获取用户图片列表

|  |
| --- |
| **测试用例 3.71: 获取用户图片列表** |
| 综述:   1. 目的：验证用户是否能够正常获取自己图库内的所有图片 2. 条件：在测试用户对应的数据库中事先准备不同大小，不同规格的图片（jpg,png,gif）。 |
| 步骤：  1.登录测试账号  2.进行图片获取 |
| 期望输出:  获取成功 |
| 实际输出:  正常 |

### 图片缩略图展示

|  |
| --- |
| **测试用例 3.81: 图片缩略图展示** |
| 综述:   1. 目的：验证从后端返回的照片是否进行缩略图处理 2. 条件：在的数据库中事先准备不同大小，不同规格的图片（jpg,png,gif）。 |
| 步骤：  1.登录测试账号  2.进行图片获取 |
| 期望输出:  获取成功，界面现实的图片都以缩略图表示 |
| 实际输出:  正常 |

### 下拉刷新

|  |
| --- |
| **测试用例 3.91: 下拉刷新** |
| 综述:   1. 目的：验证下拉是否能够刷新另一个用户上传的图片 2. 条件：多个用户按顺序上传。 |
| 步骤：  1.登录两个测试账号  2.分别上传图片 |
| 期望输出:  刷新列表成功 |
| 实际输出:  正常 |

### 更改瀑布流布局

|  |
| --- |
| **测试用例 3.101: 更改瀑布流布局** |
| 综述:   1. 目的：验证是否能自由变化多条瀑布流布局 2. 条件：有足够多的图片制成 |
| 步骤：  1.登录测试账号  2.选择改变瀑布流排序列数 |
| 期望输出:  成功更改 |
| 实际输出:  正常 |

### 保存图片

|  |
| --- |
| **测试用例 3.11.1: 保存图片** |
| 综述:   1. 目的：将图片保存到本地 2. 条件：数据库中已经存在图片 |
| 步骤：  1.登录测试账号  2.选择图片进行保存 |
| 期望输出:  成功保存，并且显示保存路径 |
| 实际输出:  错误，详情见缺陷报告[9.1.2](#_Bv) |

|  |
| --- |
| **测试用例 3.11.2: 保存图片** |
| 综述:   1. 目的：将图片保存到本地 2. 条件：数据库中已经存在图片 |
| 步骤：  1.登录测试账号  2.选择图片进行保存 |
| 期望输出:  成功保存，并且显示保存路径 |
| 实际输出:  正常 |

### 复制外链

|  |
| --- |
| **测试用例 3.12.1:** 复制外链 |
| 综述:   1. 目的：生成并复制图片的外链，能够在PC端网页打开 2. 条件：数据库中已经存在图片 |
| 步骤：  1.登录测试账号  2.选择图片进行复制外链 |
| 期望输出:  复制成功 |
| 实际输出:  正常 |

### 编辑图片

|  |
| --- |
| **测试用例 3.13.1:** 编辑图片 |
| 综述:   1. 目的：能够唤出手机自带的编辑系统对图片进行编辑并且保存 2. 条件：数据库中已经存在图片 |
| 步骤：  1.登录测试账号  2.选择图片进行编辑 |
| 期望输出:  成功编辑并且保存在手机的一个文件中 |
| 实际输出:  正常 |

### 分享图片

|  |
| --- |
| **测试用例 3.14.1:** 分享图片 |
| 综述:   1. 目的：能够将图片分享到QQ的应用端 2. 条件：数据库中已经存在图片 |
| 步骤：  1.登录测试账号  2.选择图片进行分享 |
| 期望输出:  分享成功 |
| 实际输出:  正常 |

### 显示图片信息

|  |
| --- |
| **测试用例 3.15.1:** 分享图片 |
| 综述:   1. 目的：能够获得图片的详细信息 2. 条件：数据库中已经存在图片 |
| 步骤：  1.登录测试账号  2.选择图片点击查看详细信息 |
| 期望输出:  出现图片的对应信息 |
| 实际输出:  正常 |

## 分析摘要

### 能力

经测试本软件的基本功能都能够正常运行，具有良好的运行效果，能力姣好。

### 缺陷和限制

经测试，本软件缺少批量上传和批量删除的能力，在上传图片较多的情况下可能存在重复操作浪费时间的工作，不同品牌的手机对权限的验证存在少许区别，部分手机可能需要自己主动开启读取文件权限，功能和美观性可能存在局限性，需要继续添加辅助性功能，比如UI背景的更换，个人信息的编辑等等

### 建议

若有多余时间可以对上述缺陷和限制进行弥补和实现。

### 评价

该软件已经达到的预期目的，能够正常使用。

# 缺陷报告

## 单元测试测试用例发现的缺陷

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **测试人** |  | **时间** | 2019-05-7 |
| **软件名称** | android简易图床 | **编号/版本** | **第一次迭代** |
| **测试环境** | **Android8.0** | | |
| **测试用例** | **测试用例3.2.3** | | |
| 1. **前提**   点击注册按钮，切数据库中不存在全为空的账号密码   1. **操作步骤**   在账号密码处全部都为空，不输入任何数据，直接点击提交  **3、期望结果**  报错：注册失败  **4、实际结果**  弹出添加成功，并且能够登陆**。** | | | |
| **错误类别： 致命性错误（） 功能性错误（X） 建议性错误（）** | | | |
| **优先级：P2** | | | |
| **严重程度：2** | | | |
| **测试结果分析与建议：对注册的账号密码多进行一次筛选和判定** | | | |

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **测试人** |  | **时间** | 2019-06-7 |
| **软件名称** | android简易图床 | **编号/版本** | **第三次迭代** |
| **测试环境** | **Android8.0** | | |
| **测试用例** | **测试用例3.11.1** | | |
| **1.前提**  登陆APP，且图库内存在图片  **2.操作步骤**  点击一张图片进行保存  **3、期望结果**  自动建立一个所属APP的文件夹并且保存成功，提示保存的目录  **4、实际结果**  无法打开文件夹 | | | |
| **错误类别： 致命性错误（X） 功能性错误（） 建议性错误（）** | | | |
| **优先级：P1** | | | |
| **严重程度：2** | | | |
| **测试结果分析与建议：权限请求存在问题，建议所有操作都进行一次动态权限申请。** | | | |

## 系统测试运行中发现的缺陷

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **测试人** |  | **时间** | 2019-05.22 |
| **软件名称** | android简易图床 | **编号/版本** | **第一次迭代** |
| **测试环境** | **Android8.0** | | |
| **测试用例** | **无** | | |
| **1.前提**  APP内已经存在属于多个用户的多张图片  **2.操作步骤**  打开用户图片  退回主页  **3、期望结果**  界面显示所有图片  **4、实际结果**  只显示了对应用户的图片 | | | |
| **错误类别： 致命性错误（X） 功能性错误（） 建议性错误（）** | | | |
| **优先级：P2** | | | |
| **严重程度：2** | | | |
| **测试结果分析与建议：修改显示权限** | | | |

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **测试人** |  | **时间** | 2019-05.29 |
| **软件名称** | android简易图床 | **编号/版本** | **第二次迭代** |
| **测试环境** | **Android8.0** | | |
| **测试用例** | **无** | | |
| **1.前提**  第一次安装APP并且登录  **2.操作步骤**  点击我的上传  **3、期望结果**  出现对应账号内所有图片  **4、实际结果**  出现了用于测试的图片Mock | | | |
| **错误类别： 致命性错误（） 功能性错误（X） 建议性错误（）** | | | |
| **优先级：P3** | | | |
| **严重程度：3** | | | |
| **测试结果分析与建议：软件设置为读取本地保存的True or False，第一次安装手机内并没有所以读取了默认值，默认值为true，建议将默认值设置为False。** | | | |

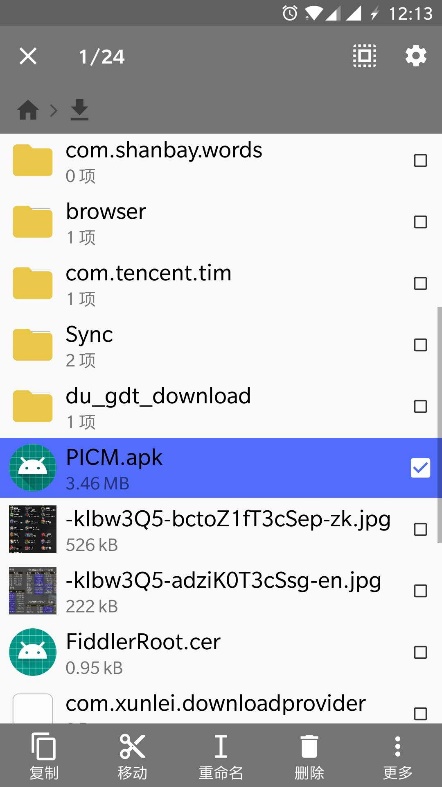
### 

|  |  |  |  |
| --- | --- | --- | --- |
| **测试人** |  | **时间** | 2019-05.29 |
| **软件名称** | android简易图床 | **编号/版本** | **第二次迭代** |
| **测试环境** | **Android8.0** | | |
| **测试用例** | **无** | | |
| **1．前提**  APP内已经存在属于多个用户的多张图片  **2.操作步骤**  打开APP获取图片  **3、期望结果**  界面显示所有图片  **4、实际结果**  没有显示图片 | | | |
| **错误类别： 致命性错误（X） 功能性错误（） 建议性错误（）** | | | |
| **优先级：P1** | | | |
| **严重程度：1** | | | |
| **测试结果分析与建议：权限存在一定问题** | | | |

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **测试人** |  | **时间** | 2019-05.29 |
| **软件名称** | android简易图床 | **编号/版本** | **第二次迭代** |
| **测试环境** | **Android8.0** | | |
| **测试用例** | **无** | | |
| **1．前提**  APP内已经存在属于多个用户的多张图片  **2.操作步骤**  打开APP获取图片  **3、期望结果**  界面显示所有图片  **4、实际结果**  没有显示图片 | | | |
| **错误类别： 致命性错误（X） 功能性错误（） 建议性错误（）** | | | |
| **优先级：P1** | | | |
| **严重程度：1** | | | |
| **测试结果分析与建议：权限存在一定问题** | | | |

# 安装手册

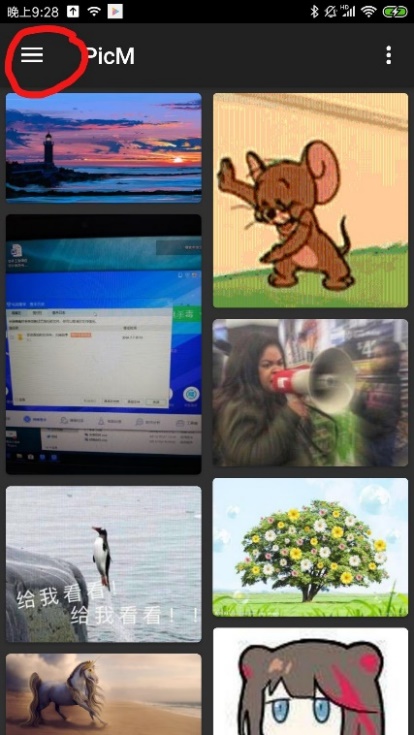
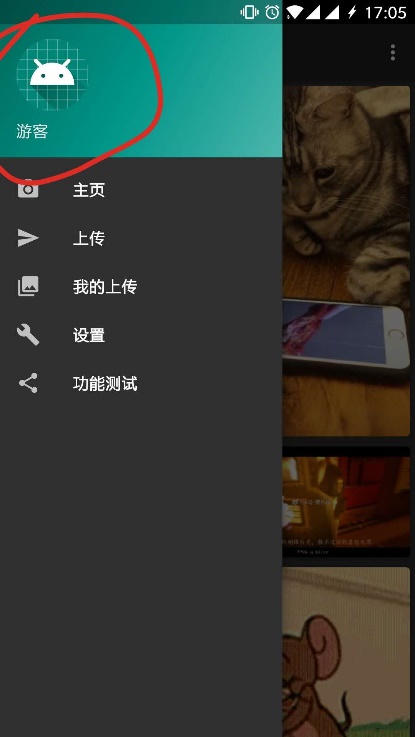
将apk下载到手机内部存储，在文件管理器中打开apk安装程序

打开apk后可能会提示未知来源，这是当前使用的文件管理器没有安装apk的权限，点击设置-允许来自此来源的应用进行授权操作。返回安装界面点击安装，等待安装完成。



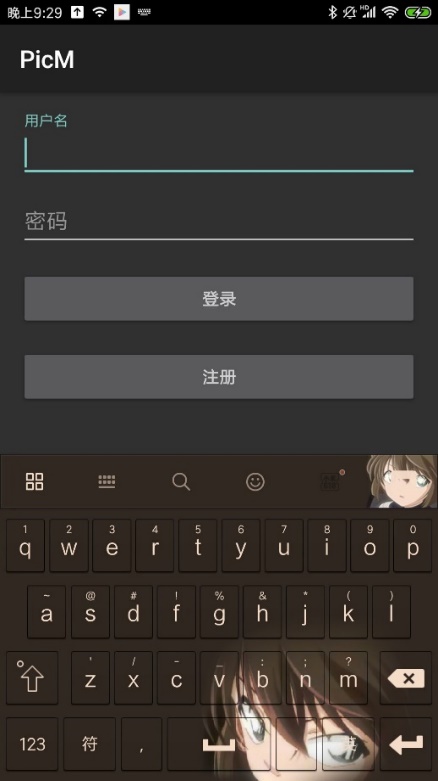
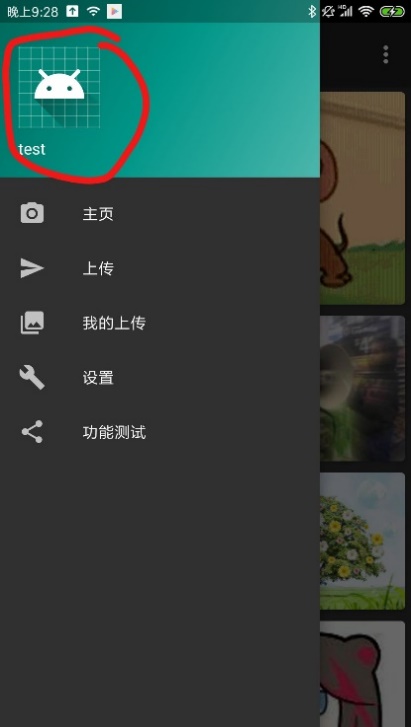
# 使用手册

## 注册登录

点击红圈处或者右拉可以唤出侧边栏。

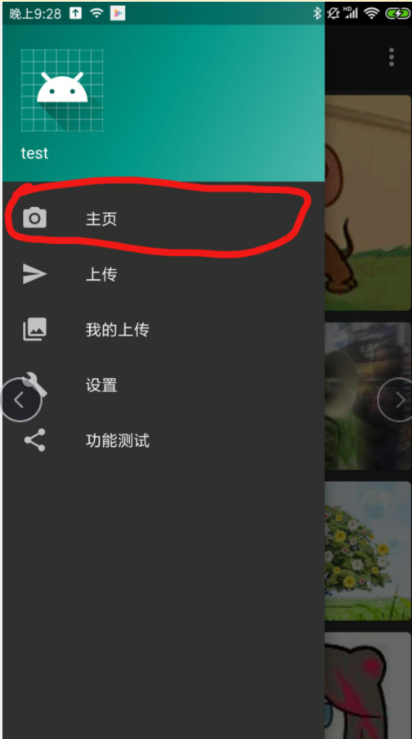
点击侧边栏的红圈处可以呼出注册登录界面。

输入用户名和密码，如果是登录就直接点击登录，如果是注册就点击注册，点击注册后可以自动登录

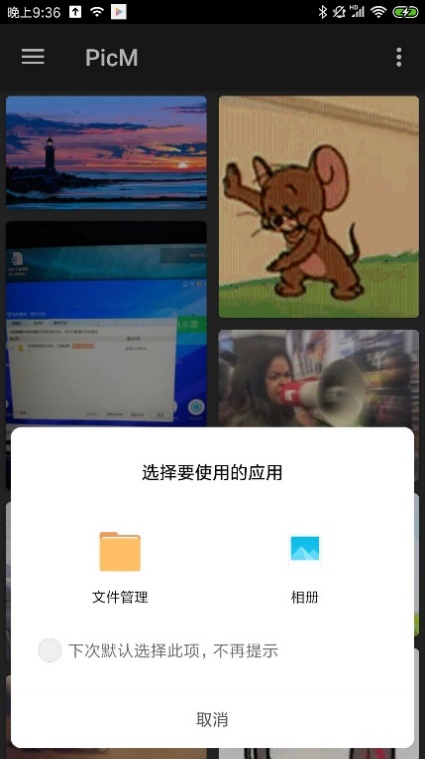
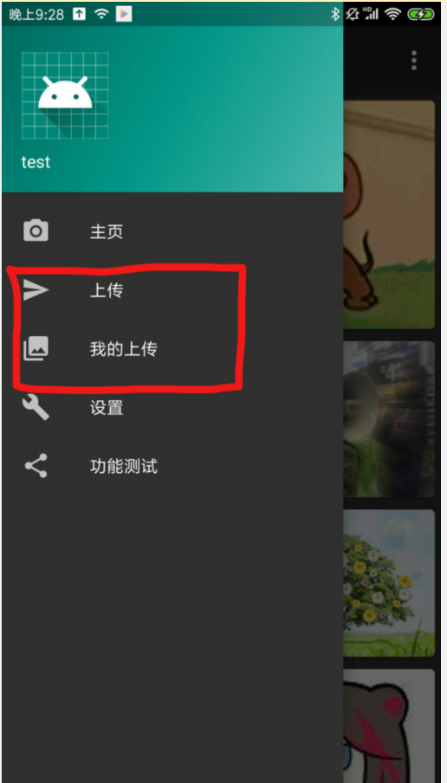
登录后的在侧边栏可以查看当前登录用户，此时点击头像可以注销你的账户(上述右图)。

## 查看所有图片



登录成功后点击红圈所在的主页按钮就可以看到所有图片，并且下拉屏幕可以刷新图片

## 上传图片和查看我的上传



点击上传按钮会调用可以的相册程序进行图片选择

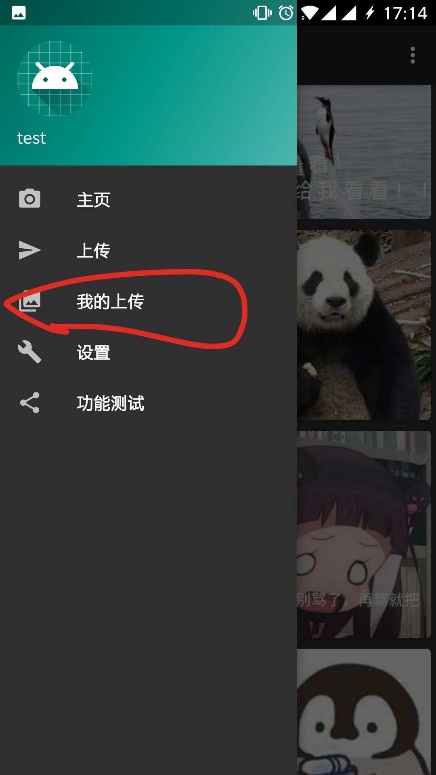
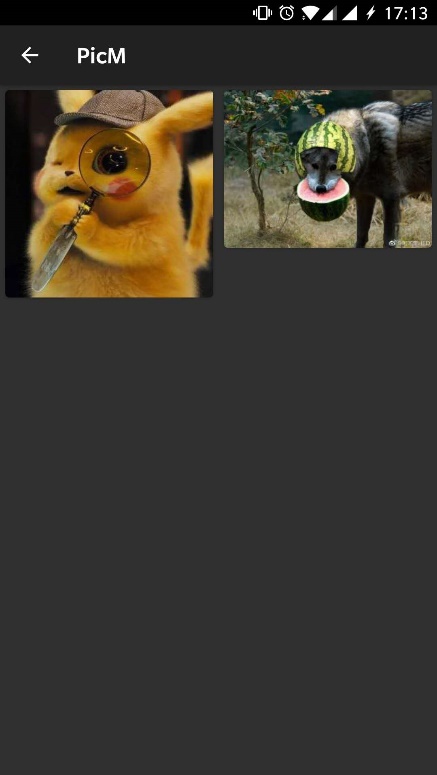
小米手机上系统显示如上右图

点击选择小米系统相册

显示如下画面

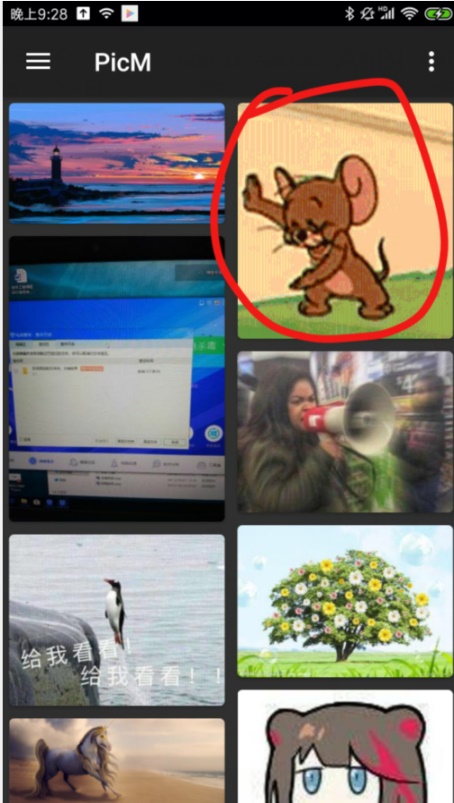


选择并点击你心仪的图片就可以上传。

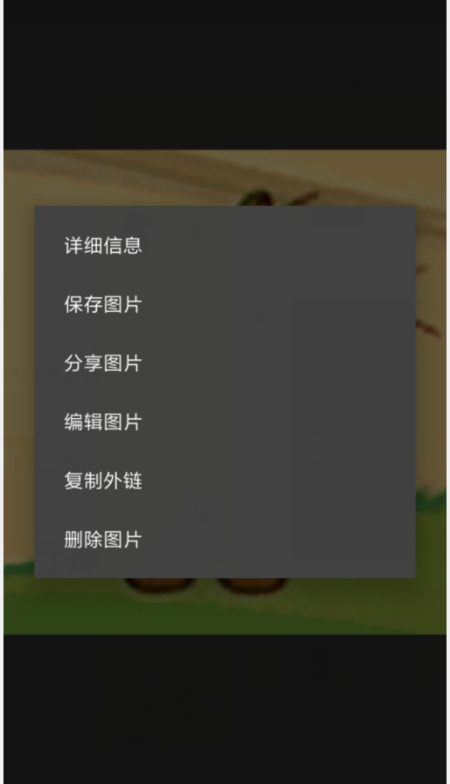
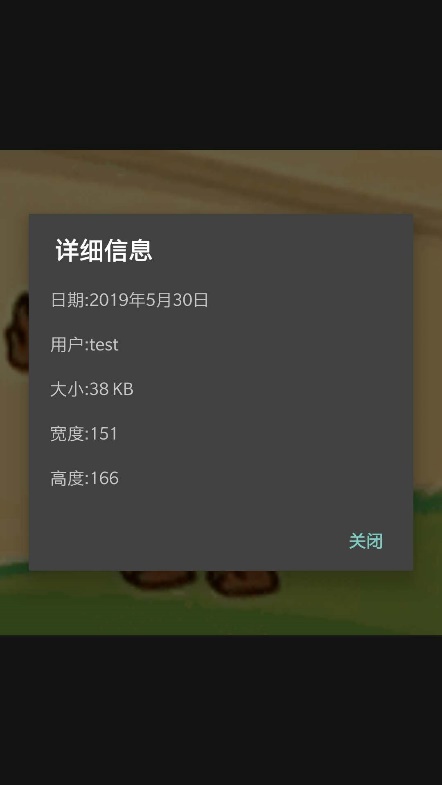
在侧边栏中点击我的上传可以查看当前用户上传的图片。

## 对图片进行操作

轻轻点击一张图片可以全屏显示图片原图。

在全屏视图下长按图片可以解锁新的操作：

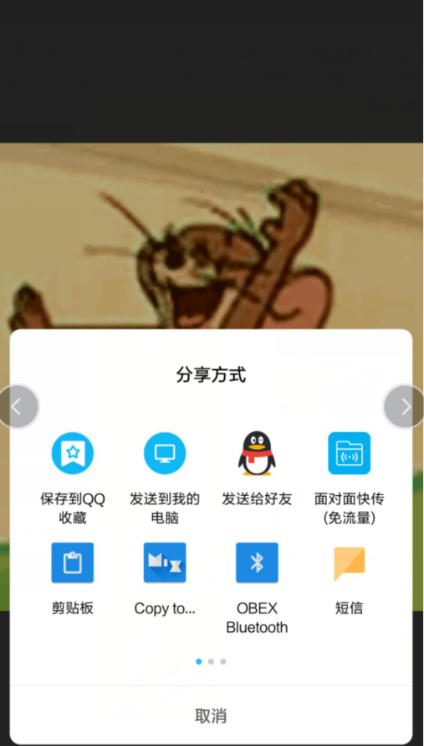
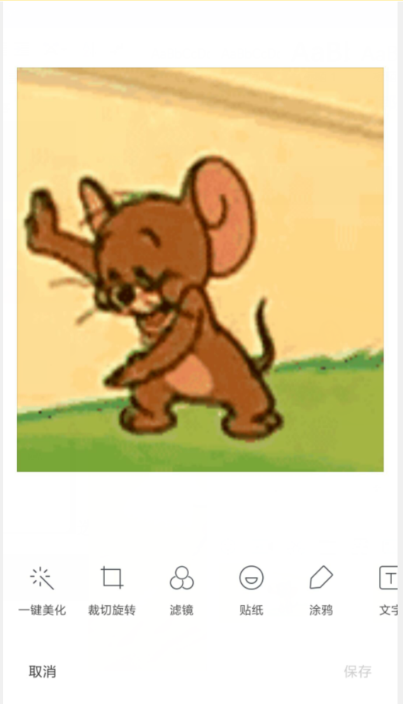
 

点击你可以得到图片的详细信息，如上右图。

点击可以将这张图片保存到你的手机文件夹中

点击会跳出如下左图界面（演示为小米系统UI）

选择你想选择的分享方式就可以分享了。

点击会出现可以调用图片编辑应用

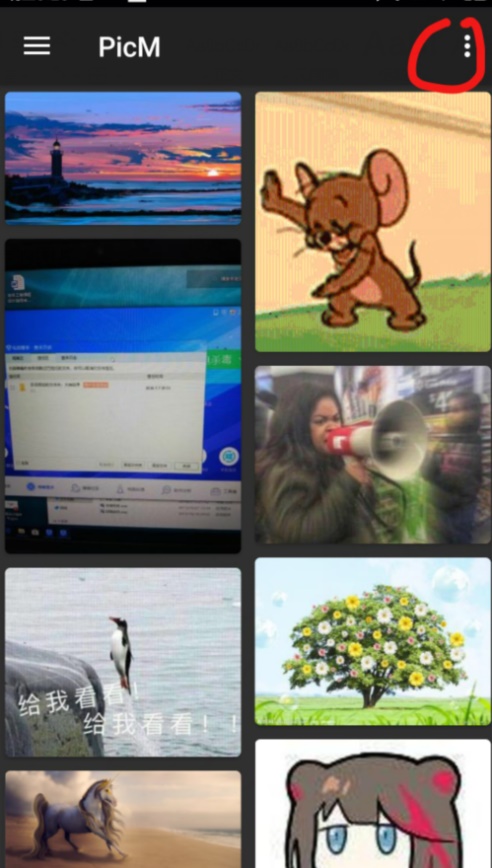
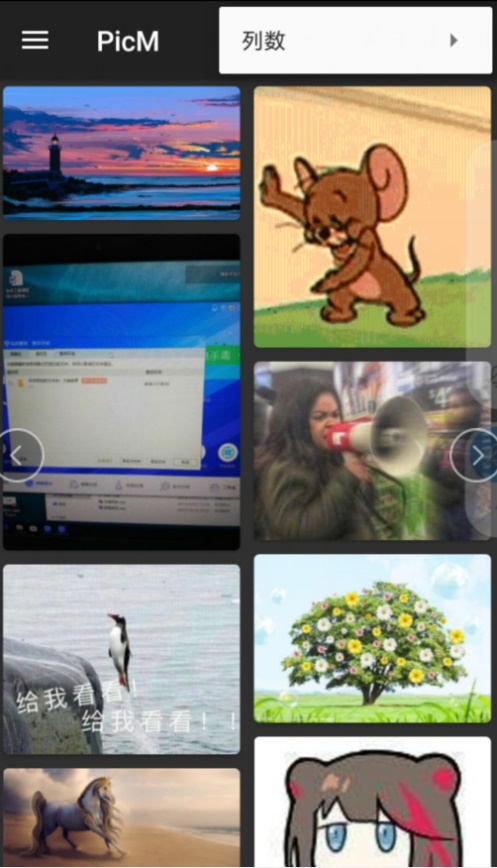
如上右图为小米系统自带的图片编辑程序

编辑完成后可以选择点击按钮来保存到本地。

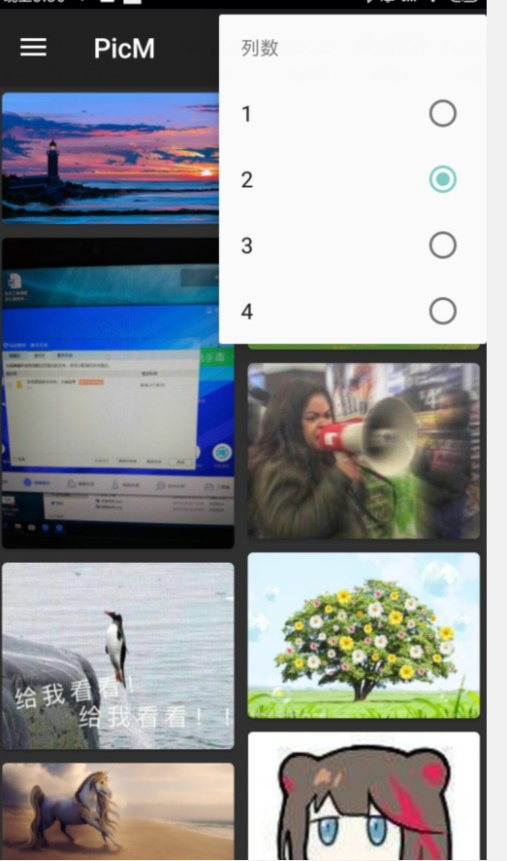
点击在粘贴板上会获得该图片的公开访问链接，直接去你想要分享的地方粘贴就可以了。

点击可以删除图片，你只能够删除当前登录用户所上传的图片。

## 更改图片排列列数

点击右上角的红圈展开菜单可以看到右图所示，点击列数展开菜单



在菜单中选择你想要更改的列数就可以改变了瀑布流列数。

# 编码清单

## 客户端源码

..**\PicM\app\src\main\**

**│ AndroidManifest.xml**

**│**

**├───java**

**│ └───com**

**│ └───dhu777**

**│ └───picm**

**│ │ BasePresenter.java**

**│ │ BaseView.java**

**│ │ MainActivity.java**

**│ │**

**│ ├───data**

**│ │ │ LoginDataSource.java**

**│ │ │ LoginRepositrory.java**

**│ │ │ PicDataSource.java**

**│ │ │ PicInfoRepositrory.java**

**│ │ │**

**│ │ ├───entity**

**│ │ │ BaseResponse.java**

**│ │ │ PicInfo.java**

**│ │ │ UserToken.java**

**│ │ │**

**│ │ ├───local**

**│ │ │ AppDataBase.java**

**│ │ │ PicLocalRepo.java**

**│ │ │ TokenLocalRepo.java**

**│ │ │ UserTokenDAO.java**

**│ │ │**

**│ │ └───remote**

**│ │ LoginRemoteContract.java**

**│ │ LoginRemoteRepo.java**

**│ │ PicRemoteContract.java**

**│ │ PicRemoteRepo.java**

**│ │ PicUpload.java**

**│ │ PicUserRemoteRepo.java**

**│ │**

**│ ├───home**

**│ │ HomeActivity.java**

**│ │ HomePresenter.java**

**│ │ SettingsActivity.java**

**│ │**

**│ ├───login**

**│ │ LoginActivity.java**

**│ │**

**│ ├───mock**

**│ │ │ Injection.java**

**│ │ │**

**│ │ └───data**

**│ │ FakeLoginRepositrory.java**

**│ │**

**│ ├───picdetail**

**│ │ InfoDialogFragment.java**

**│ │ PicDetailActivity.java**

**│ │ PicDetailContract.java**

**│ │ PicDetailPresenter.java**

**│ │**

**│ ├───piclist**

**│ │ PicListContract.java**

**│ │ PicListFragment.java**

**│ │ PicListPresenter.java**

**│ │ PicTestActivity.java**

**│ │ PicUserActivity.java**

**│ │**

**│ └───util**

**│ ComAppGlideModule.java**

**│ ComUtil.java**

**│ DataHolder.java**

**│ GenericFileProvider.java**

**│ ReLoginException.java**

**│**

**└───res**

**├───drawable**

**│ ic\_launcher\_background.xml**

**│ ic\_menu\_camera.xml**

**│ ic\_menu\_gallery.xml**

**│ ic\_menu\_manage.xml**

**│ ic\_menu\_send.xml**

**│ ic\_menu\_share.xml**

**│ ic\_menu\_slideshow.xml**

**│ side\_nav\_bar.xml**

**│**

**├───drawable-v24**

**│ ic\_launcher\_foreground.xml**

**│**

**├───layout**

**│ activity\_home.xml**

**│ activity\_login.xml**

**│ activity\_main\_test.xml**

**│ activity\_pic\_detail.xml**

**│ activity\_pic\_list.xml**

**│ app\_bar\_home.xml**

**│ content\_home.xml**

**│ dialog\_pic\_detail.xml**

**│ nav\_header\_home.xml**

**│ piclist\_item.xml**

**│ piclis\_frag.xml**

**│ settings\_activity.xml**

**│**

**├───menu**

**│ activity\_home\_drawer.xml**

**│ activity\_pic\_detail\_context.xml**

**│ home.xml**

**│**

**├───mipmap-anydpi-v26**

**│ ic\_launcher.xml**

**│ ic\_launcher\_round.xml**

**│**

**├───mipmap-hdpi**

**│ ic\_launcher.png**

**│ ic\_launcher\_round.png**

**│**

**├───mipmap-mdpi**

**│ ic\_launcher.png**

**│ ic\_launcher\_round.png**

**│**

**├───mipmap-xhdpi**

**│ ic\_launcher.png**

**│ ic\_launcher\_round.png**

**│**

**├───mipmap-xxhdpi**

**│ ic\_launcher.png**

**│ ic\_launcher\_round.png**

**│**

**├───mipmap-xxxhdpi**

**│ ic\_launcher.png**

**│ ic\_launcher\_round.png**

**│**

**├───values**

**│ colors.xml**

**│ dimens.xml**

**│ strings.xml**

**│ styles.xml**

**│**

**├───values-v21**

**│ styles.xml**

**│**

**└───xml**

**provider\_paths.xml**

**root\_preferences.xml**

## 服务器端源码

**..\piccloud\src\main\**

**│**

**├─java**

**│ └─cn**

**│ └─dhu**

**│ └─piccloud**

**│ │ PiccloudApplication.java**

**│ │**

**│ ├─Controller**

**│ │ PicController.java**

**│ │ UserController.java**

**│ │**

**│ ├─Exception**

**│ │ AuthFailException.java**

**│ │ TypeMisMatchException.java**

**│ │**

**│ ├─Interceptor**

**│ │ InterceptorRegister.java**

**│ │ LoginInterceptor.java**

**│ │**

**│ ├─Mapper**

**│ │ PicInfoMapper.java**

**│ │ UserMapper.java**

**│ │**

**│ ├─Pojo**

**│ │ PicInfo.java**

**│ │ ResponseMsg.java**

**│ │ User.java**

**│ │**

**│ ├─Service**

**│ │ PicInfoService.java**

**│ │ PicInfoServiceImpl.java**

**│ │ PicStorageService.java**

**│ │ PicStorageServiceImpl.java**

**│ │ UserService.java**

**│ │ UserServiceImpl.java**

**│ │**

**│ └─Util**

**│ HashUtil.java**

**│ ImgProcessUtil.java**

**│ JwtUtil.java**

**│**

**└─resources**

**│ application.yml**

**│**

**├─mybatis**

**│ │ mybatis-config.xml**

**│ │**

**│ └─mapper**

**│ PicInfoMapper.xml**

**│ UserMapper.xml**

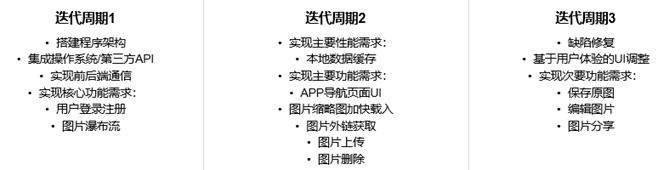
**│**

**├─static**

**└─templates**

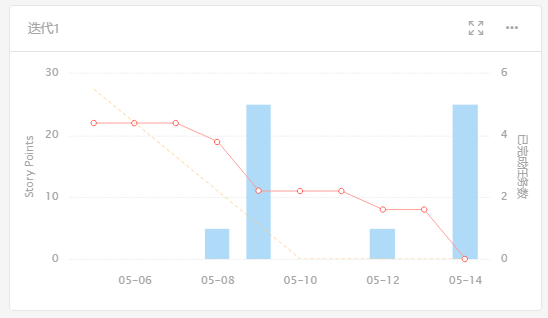
# 项目总结

**本项目开发采用Scrum敏捷开发的模式进行开发，采用**[**Teambition**](https://www.teambition.com/)**作为项目管理工具。项目切割为3个迭代周期：**

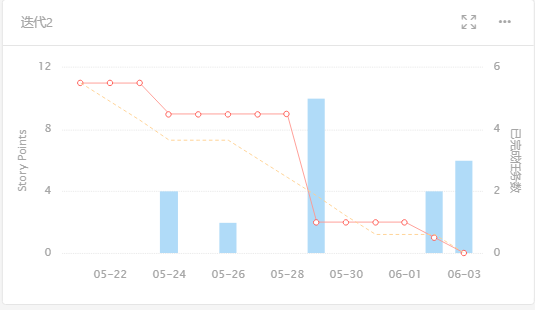


**以下是每个周期具体执行情况：**

**（燃尽图中黄色虚线为理想状况，红色实线为实际执行，StoryPoints为与需求相关的任务设置，测试任务与缺陷修复任务在测试完成，测试文档提交后才计入已完成状态）**

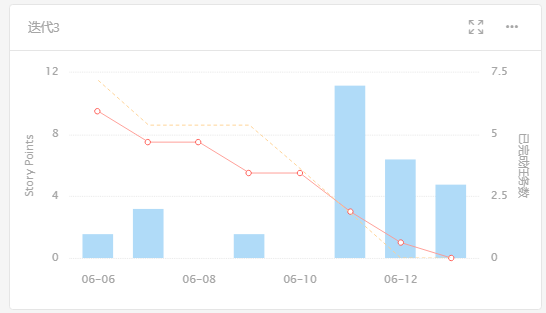
**迭代周期1的燃尽图**

**第一个迭代周期执行效果并不理想，周期结束时间延后了3天，个人认为最主要的原因在于：第一个迭代周期的重点是技术选型与框架搭建，需要时间进行程序调试、阅读文档。在做项目开发计划以及任务分配时，没有考虑到这部分时间只以实现功能需求为标志分配了一周的时间，上述燃尽图两个有较大项目进展的时间点对应的是两个功能需求的实现。实际执行过程中前期主要时间花在网络框架的搭建，中期则在图片显示框架的集成花了较多的时间。**



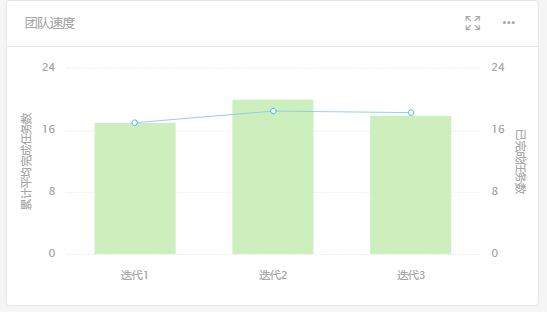
**迭代周期2的燃尽图**

**第二个迭代周期实际执行情况比第一个周期好上不少，至少按时结束并交付。在第一个周期完成后，前后端代码可以做到可持续集成，保证每次代码提交后app都能正常运行，且为客户端设计mock数据，故在不产生接口变动的情况下前后端可以根据自身实际情况推进项目开发。本周期存在的问题是对需求的优先级划分均匀而忽视了一些需求需要相对较长的调试、测试时间：数据缓存功能从一开始就开始做了，但堆积到29号才正式处于已完成状态。**



**迭代周期3的燃尽图**

**最后一个周期有了前两个周期的计划与执行经验，需求实现的整体执行效果较好。但详细设计文档的变更较多，最终交付时间较晚。**



**整体团队速度图**

**整体而言，前期团队花了些时间磨合以及适应敏捷开发过程，但随着项目进行，团队执行、实现需求的效率持续提高。就最终交付的APP的实际运行状况而言，较好的实现了初期设想的绝大部分功能需求。**