《巧倩美颜系统》
详细分析与设计

V1.2

目　　录

第一部分 引言

一、编写目的

编写本说明书的目的是为了准确阐述项目概要设计结构，本概要设计说明的作者是【巧倩美颜】项目组，本概要设计说明的确认者是【项目经理】负责人，本概要设计说明的读者是项目所有直接干系人。

二、项目背景

现代人们的生活越来越多姿多彩，很多时候都想保留住美好的瞬间然后分享到朋友圈，然而现在大家对照片的审美越来越高，生活照美颜一下发出去效果会更好， 所以，美图软件横行在自拍党人群中。以前大家只能用 Photoshop 进行修图美化，但是 Photoshop 难度比较大，不容易上手，随后诞生了类似美图秀秀的修图软件，虽然功能强大，但是广告众多，并且不支持自己 diy 贴图或者制作海报，同时功能仅限于美颜照片。

对于办公人士或者学生群体，有时需要对文件进行修图，而市场上还没有一款可以同时兼顾这些个功能的软件。

针对以上现象，我们巧倩美颜项目组准备开发一款面向学生群体和年轻办公人士的美图 APP，帮助他们更好的分享精彩人生。

三、定义

Blurred：图片模糊化处理。图象的模糊一直是图像处理领域一个比较重要的东西，它的用处不仅仅是我们平时 PS 的滤镜，也常常被用来做图片数据的降噪，图片的有损压缩，和图片特征相似匹配的优化工作。

OCR：(Optical Character Recognition，光学字符识别），指电子设备（例如扫描仪或数码相机）检查纸上打印的字符，通过检测暗、亮的模式确定其形状，然后用字符识别方法将形状翻译成计算机文字的过程。

用例：（use case ）或译使用案例、用况，UML 术语，是软件工程或系统工程中对系统如何反应外界请求的描述，是一种通过用户的使用场景来获取需求的技术。每个用例提供了一个或多个场景，该场景说明了系统是如何和最终用户或其它系统互动，也就是谁可以用系统做什么，从而获得一个明确的业务目标。编写用例时要避免使用技术术语，而应该用最终用户或者领域专家的语言。用例一般是由软件开发者和最终用户共同创作的。

事件流：UML 术语。在 UML 建模中，因为用例最终是参与者通过使用用例达成他的目的，用例是达成他的目的的手段，然后又把它分成一些具体的事件流，每个事件流实际上就代表一个具体的目标。

部署：软件部署环节是指将软件项目本身，包括配置文件、用户手册、帮助文档等进行收集、打包、安装、配置、发布的过程。

图像锐化：(image sharpening)是补偿图像的轮廓，增强图像的边缘及灰度跳变的部分，使图像变得清晰，分为空间域处理和频域处理两类。

角点检测：角点通常被定义为两条边的交点，或者说，角点的局部邻域应该具有两个不同区域的不同方向的边界。角点检测(Corner Detection)是计算机视觉系统中获取图像特征的一种方法，广泛应用于运动检测、图像匹配、视频跟踪、三维重建和目标识别等，也可称为特征点检测。

mAP：mean Average Precision，是精确度的平均，是目标检测中模型性能的衡量标准。

IoU：是一种测量在特定数据集中检测相应物体准确度的一个标准。IoU 是一个简单的测量标准，只要是在输出中得出一个预测范围(bounding boxes)的任务都可以用 IoU 来进行测量。

第二部分 项目概述

一款轻量级的 web 端和手机端修图应用，提升美颜软件的易用性，提高美颜软件的服务质量，拓展美颜软件的市场前景，建设一款受市场欢迎，面向现代群体的优质美颜软件。

系统主要包括用户信息管理模块，图片处理模块，图片传输模块三大部分。

#	产品	模块	组件	规格/型号	角色	接入
1	巧倩美颜系统	用户模块	账号注册	判定账号邮箱，提示用户	用户	web、移动
			账户登录	判断账号密码，提示用户	用户	web、移动
2			用户信息管理	修改用户名，修改头像	用户	web、移动
			云相册	查看照片，上传下载照片	用户	web、移动
3		图像处理	照片美化	滤镜，调色，人物美化，添加自定义贴纸	用户	web、移动
4			图片编辑	旋转，裁剪，缩放	用户	web、移动
11		文件处理	照片转扫描件	将用户上传的照片转换为扫描件	用户	web、移动
12			文字 OCR	识别用户上传照片中的文字	用户	web、移动

系统主要面向 P 图零基础的群体，能够提供基本的修图操作，并且具有较高的易用性。同时也提供专业修图操作，能够满足专业人士的需求。界面美观，良性需求需

要质感设计，提供多种主题，满足用户个性化界面需求。实现 7x24 小时连续不间断的业务访问。

第三部分 总体设计

一、技术架构设计

1. 前端：

前端采用 VUE 框架进行开发，利用 vue-cli 进行项目结构搭建，采用 vue-router，vuex 和 axios 组建，完成前端路由，状态管理和与后端的交互；UI 部分利用了 element-ui，保持 UI 风格统一。

2. 后端：

后端采用 Django 框架，由于前端采用了 VUE 进行前后端分离，所以摒弃了 Django 中的模板，利用 Django 的路由功能，配合后端图像处理算法进行后端服务的提供。采用 Django-cors-headers 插件解决跨域问题，数据库采用 MySQL8，配合 Django 的 mysqlclient 驱动进行数据库的 CURD 操作。

3. 部署

利用 Docker，docker-compose 工具，实现快捷部署。在本地搭建好镜像，直接在租用的阿里云上利用 Dockerfile 进行部署。

二、核心控制流程

1 请求路由配置

注：下表给出了系统的路由配置，具体数据规约请见后方交互规约。

编号	系统模块	请求 URL	请求说明
1	用户系统	http://127.0.0.1:8080/user/login/	登录请求
2		http://127.0.0.1:8080/user/register/	注册请求 POST 方法注册
3		http://127.0.0.1:8080/user/register/	获取注册验证码 GET 方法获取
4		http://127.0.0.1:8080/user/change-user-name	修改用户名请求
5		http://127.0.0.1:8080/user/add-album/	新建相册请求
6		http://127.0.0.1:8080/user/delete-album/	删除相册请求
7		http://127.0.0.1:8080/user/change-avatar/	更换头像请求
8		http://127.0.0.1:8080/user/getAllAlbumsInfo/	获取云相册信息请求（全部相册信息）
9		http://127.0.0.1:8080/user/getAllStickersInfo/	获取所有贴纸信息请求
10		http://127.0.0.1:8080/user/getImagesInfo/	获取所有图片信息请求（指定相册）
11		http://127.0.0.1:8080/user/add-image/	上传图片请求
12		http://127.0.0.1:8080/user/add-sticker/	上传贴纸请求
13		http://127.0.0.1:8080/user/change-album-name/	更改相册名请求
14		http://127.0.0.1:8080/user/delete-album/	删除图片请求
15		http://127.0.0.1:8080/user/delete-sticker/	删除贴纸请求
16	用户系统	http://127.0.0.1:8080/i	OCR 识别请求

	图片处理系统	mgProc/ocr-proc/	
17		http://127.0.0.1:8080/imgProc/waterMask/	添加水印请求
18		http://127.0.0.1:8080/imgProc/changeSharp/	调整锐化度请求
19		http://127.0.0.1:8080/imgProc/changeContrast/	调整对比度请求
20		http://127.0.0.1:8080/imgProc/beautify_white/	智能美白请求
21		http://127.0.0.1:8080/imgProc/beautify_buffering/	智能磨皮请求
22		http://127.0.0.1:8080/imgProc/face_makeup/	智能美化(人脸)请求
23		http://127.0.0.1:8080/imgProc/faceswap/	换脸请求
24		http://127.0.0.1:8080/imgProc/picScanner/	图片转扫描件请求
25		http://127.0.0.1:8080/imgProc/stitch/	图片全景拼接请求
26		http://127.0.0.1:8080/imgProc/face_thin/	智能瘦脸请求
27		http://127.0.0.1:8080/imgProc/filter/	智能滤镜请求
28		http://127.0.0.1:8080/imgProc/styleTransform/	风格迁移请求

2 前后端数据交互规约

2.1 用户系统

2.1.1 登录

请求 URL：/user/login/

请求方式：POST

提交数据格式：

```
{
    "user_mail": .... ,
    "user_password": ..... (md5 加盐加密)
}
```

返回数据格式：

```
{
    "error_code": (0 表示成功 1001 表示邮箱不存在 1002 表示密码错误 1006 表示未知错误（可能由于数据库等）),
    "user_data": {
        "user_name": ".....",
        "id": "......",
        "user_mail": "......",
        "user_avatar_url": "......"
    }
}
```

2.1.2 注册

请求 URL: /user/register/

请求方法： POST

提交数据：

{

"user_mail": "......",

"user_name": "......",

"verify_code": ".....",

"user_password": "......"

}

返回数据：

{

"error_code": ""(0 表示成功，1003 表示验证码错误, 1006 未知错误（可能是数据库等）　　1007 表示验证码过期)

}

2.1.3 请求发送验证码

请求 URL：/user/register/

请求方法：GET

提交数据：?usermail="....."

返回数据：

{

"error_code": (0 表示成功，1004 表示邮箱已注册，1005 表示发送错误)

}

2.1.4 修改用户名

请求 URL: /user/change-user-name

请求方法：POST

提交数据：

{

“user_id”：".....",

"user_new_name": "....."

}

返回数据：

{

"error_code": (0 表示成功，1006 表示失败（未知错误，可能由于数据库）),

“user_new_data”：{

"user_new_name": "......."

}

}

2.1.5 新建相册

请求 URL：/user/add-album/

请求方法：POST

前端发送数据格式　　# 前端判断相册不重名

{

'user_id': xxx,

'album_name': "xxx"　　# 前端验证不可为空

}

后端返回格式

{

```
    'error_code': xxx,              #   0:新建成功
    'album_info': {
        "id": '',
        "name": '',
        "date": "2020-07-23",
        "size": "",
        "url": ""
    }
}
```

2.1.6 删除相册

请求 URL：/user/delete-album/

请求方法：POST

前端发送数据格式

```
{
    'user_id': xxx,
    'album_name': "xxx"             # 前端验证不可为空
}
```

后端返回格式

```
msg = {
    'error_code': xxx          # (0 表示成功 1006 表示数据库连接错误)
}
```

2.1.7 更换头像

请求 URL: /user/change-avatar/

请求方式：POST

提交数据

返回数据：

```
{
    "error_code": (0 没有错误，1006 未知错误)
    "user_new_data": {
        "user_new_avatar_url": "....."
    }
}
```

2.1.8 获取云相册信息（全部相册信息）

请求 URL：/user/getAllAlbumsInfo/

请求方式：GET

提交数据：

```
{
     “user_id": xxx
}
```

返回数据：

```
{
    "error_code": 0,
    "AlbumsNum": 2,
    "AlbumsInfo": [
```

```
        {
            "id": '',
            "name": '',
            "date": "2020-07-23",
            "size": "",
            "url": ""
        },
        ...
    ]
}
```

2.1.9 获取所有贴纸信息

请求 URL：/user/getAllStickersInfo/

请求方式：GET

提交数据：

```
{
    “user_id": xxx
}
```

返回数据：　关注字段名！！（与所有相册信息格式相同）

```
{
    "error_code": 0,                    # 0 表示成功  1006 表示数据库错误
    "StickersNum": 0,
    "StickersInfo": [
```

```
        {
            "id": '',
            "name": '',
            "date": "2020-07-23",
            "size": "",
            "url":'',
        },
        ...
    ]
}
```

2.1.10 获取所有图片信息（指定相册）

请求 URL：/user/getImagesInfo/

请求方式：GET

提交数据：

```
{
    “user_id": xxx
    “album_id”  :xxx
}
```

返回数据：　关注字段名！！（与所有相册信息格式相同）

```
{
    "error_code": 0,                # 0 表示成功  1006 表示数据库错误
    "ImagesNum": 0,
```

```
    "ImagesInfo": [
        {
            "id": '',
            "name": '',
            "date": "2020-07-23",
            "size": "",
            "url": ""
        },
        ...
    ]
}
```

2.1.11 上传图片

请求 URL：/user/add-image/

请求方式：POST

提交数据：

```
{
    'user_id': xxx,
    'album_id': xxx
}
```

注：提交时加上 FILE

返回数据：

```
{
```

"error_code": 0,　　# 0 表示成功 1006 表示数据库错误

"imageInfo": {

"id": '',

"name": '',

"date": "2020-07-23",

"size": "",

"url": ""

}

}

2.1.12 上传贴纸

请求 URL：/user/add-sticker/

请求方式：POST

提交数据：

{

'user_id': xxx,

}

注：提交时加上 FILE

返回数据：

{

"error_code": 0,　　# 0 表示成功 1006 表示数据库错误

"stickerInfo": {

"id": '',

"name": '',

"date": "2020-07-23",

"size": "",

"url": ""

}

}

2.1.13 更改相册名

请求 Url: /user/change-album-name/

请求方式：POST

提交数据

{

“user_id": xxx,

“album_id” : xxx,

"album_name": xxx, # 原名

"album_new_name": xxx # 新名

}

返回数据

{

"error_code": 0,

"AlbumInfo": {

"id": '',

"name": '',

```
            "date": "2020-07-23",
            "size": "",
            "url": ""
        }
    }
```

2.1.14、删除图片

请求 URL：/user/delete-image/

请求方式：POST

提交数据：

```
{
    user_id: "",
    album_id: "",
    image_id: ""
}
```

返回数据：

```
{
    'error_code': 0                    # 0 成功 1006 数据库出错
    'album_cover_url': "xxx"
}
```

2.1.15、删除贴纸

请求 URL：/user/delete-sticker/

请求方式：POST

提交数据：

```
{
    user_id: "",
    sticker_id: ""
}
```

返回数据：

```
{
    error_code: 0 成功 1006 数据库出错
}
```

2.2 图片处理

2.2.1 OCR 识别

请求 URL：/imgProc/ocr-proc/

请求方式：POST

提交数据：

```
{
    'editedImg': "xxxxx"     注：base64 编码
}
```

返回数据：

```
{
    "error_code": 0,          # 0 表示成功  1008 表示图片处理异常
    "ocr_result": {
```

```
        "words_result_num": 1,                # 几行文字
        "words_result": [                     # 每行文字一个{}，合成一个[]
            {
                "words": "巧倩美颜"
            }
        ]
    }
}
```

2.2.2 水印

请求 URL：/imgProc/waterMark/

请求方式：POST

提交数据：

```
{
    'editedImg': "xxxxx",              注：base64 编码
    'text': "xxx"                      水印文字
}
```

返回数据：

```
{
    'error_code': xxx,                 # 0 表示成功  1008 表示处理异常
    'img_proc_result': "xxx"           base64 码
}
```

2.2.3 锐化

请求 URL：/imgProc/changeSharp/

请求方式：POST

提交数据：

{

'editedImg': "xxxxx", 注：base64 编码

'sharp_degree': xxx 锐化度 0--100

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.4 对比度调整

请求 URL：/imgProc/changeContrast/

请求方式：POST

提交数据：

{

'editedImg': "xxxxx", 注：base64 编码

'contrast_degree': xxx 对比度 -100--100

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.5 美白

请求 URL：/imgProc/beautify_white/

请求方式：POST

提交数据：

{

'editedImg': "xxxxx", 注：base64 编码

'white_degree': xxx 美白度 0--100

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.6 磨皮

请求 URL：/imgProc/beautify_buffering/

请求方式：POST

提交数据：

{

'editedImg': "xxxxx", 注：base64 编码

'buffer_degree': xxx 磨皮程度 0-10 推荐 3（3 已经磨得挺狠了我觉得）

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.7 智能美化（人脸）

请求 URL：/imgProc/face_makeup/

请求方式：POST

提交数据：

{

'editedImg': "xxxxx", 注：base64 编码

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.8 换脸

请求 URL：/imgProc/faceswap/

请求方式：POST

提交数据：

{

'headImg': "xxxxx", 用这张图片的头 注：base64 编码

'faceImg': "xxxxx" 用这张图片的脸 注：base64 编码

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.9 图片转扫描件

请求 URL：/imgProc/picScanner/

请求方式：POST

提交数据：

{

'editedImg': "xxxxx" 注：base64 编码

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.10 图片拼接

请求 URL：/imgProc/stitch/

请求方式：POST

提交数据：

```
{
    'img_num': xxx,                    拼接的图片数量
    'editedImg0': "xxxxx",             注：base64 编码
    'editedImg1': "xxxxx"
}
```

返回数据：

```
{
    'error_code': xxx,                 # 0 表示成功  1008 表示处理异常
    'img_proc_result': "xxx"           base64 码
}
```

2.2.11 瘦脸

请求 URL：/imgProc/face_thin/

请求方式：POST

提交数据：

```
{
    'editedImg': "xxxxx",              注：base64 编码
}
```

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

2.2.12 滤镜

请求 URL：/imgProc/filter/

请求方式：POST

提交数据：

{

'operation': xxx 详见下方表

'editedImg': "xxxxx" 注：base64 编码

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

operation 对应表：

0 --------> 毛玻璃

1 --------> 儿童蜡笔画

2 --------> 褪色

3 --------> 去雾

4 --------> 调和浮雕

5 --------> 怀旧

6 --------> 素描

7 --------> 中国剪纸

8 --------> 浮雕

9 --------> 碧绿碧绿诡异诡异的滤镜

2.2.13 风格迁移

请求 URL：/imgProc/styleTransform/

请求方式：POST

提交数据：

{

'style_num': xxx 详见下方表

'editedImg': "xxxxx" 注：base64 编码

}

返回数据：

{

'error_code': xxx, # 0 表示成功 1008 表示处理异常

'img_proc_result': "xxx" base64 码

}

0-----cartoon：卡通画风格

1-----pencil：铅笔风格

2-----color_pencil：彩色铅笔画风格

3-----warm：彩色糖块油画风格

4-----wave：神奈川冲浪里油画风格

5-----lavender：薰衣草油画风格

6-----mononoke：奇异油画风格

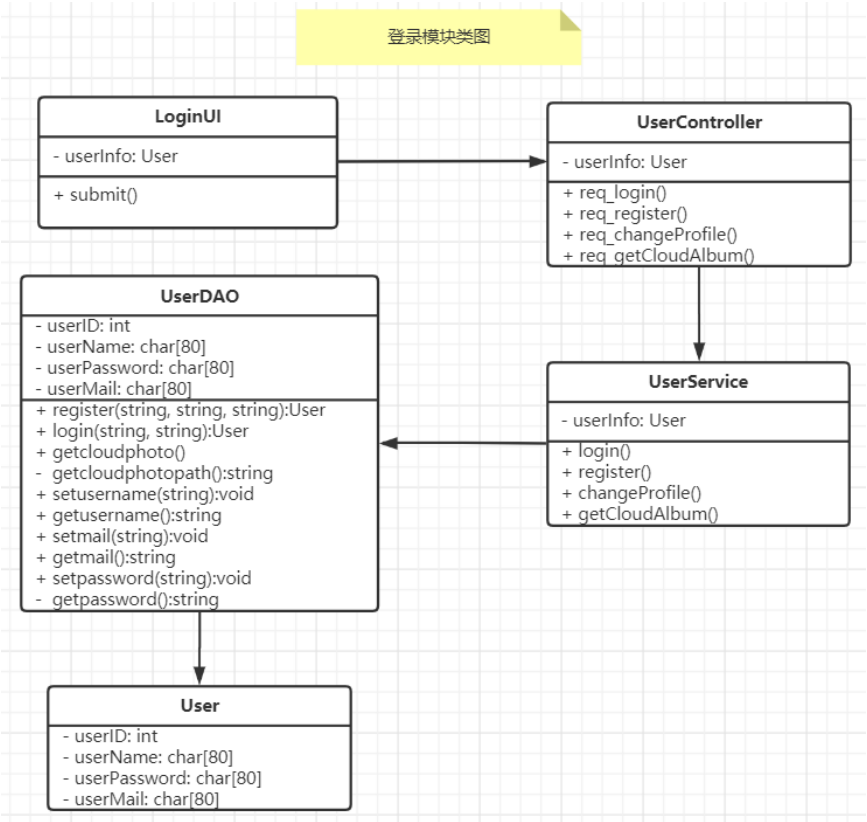
7-----scream：呐喊油画风格

8-----gothic：哥特油画风格

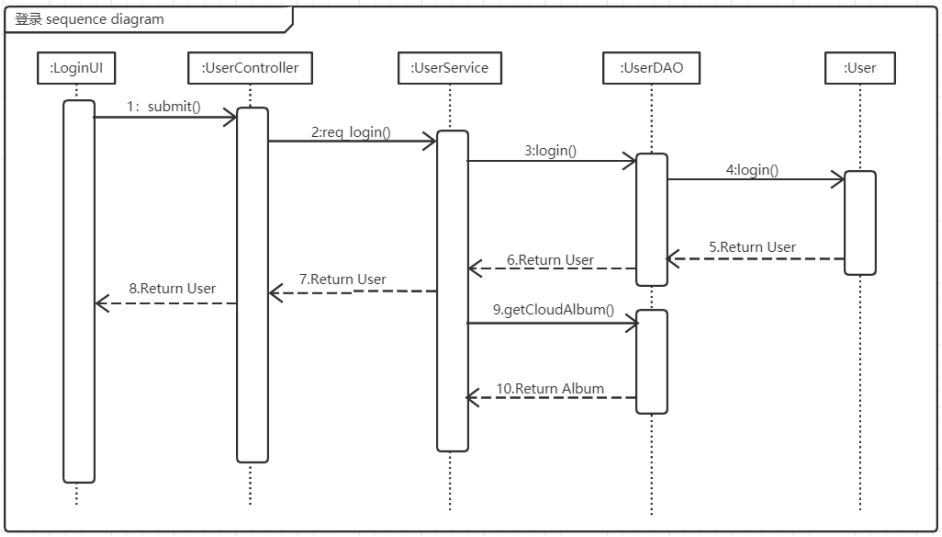
第五部分 单元模块设计

一、登录模块

1. 类图设计



2. 时序图



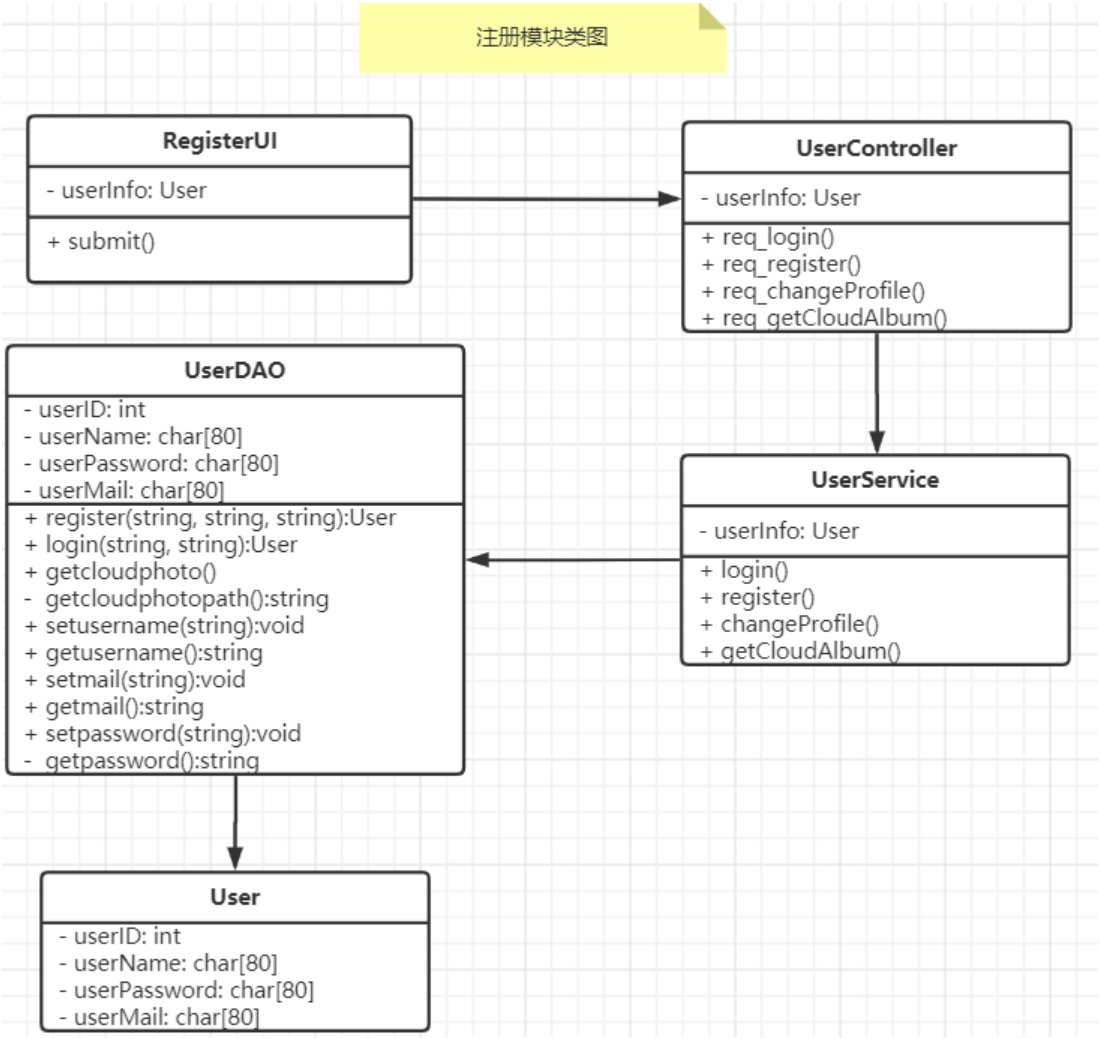
3. 流程描述

LoginUI 浏览器登录界面类 submit(); 用户在登录界面提交登录请求，返回用户对象
UserController 用户功能管理类

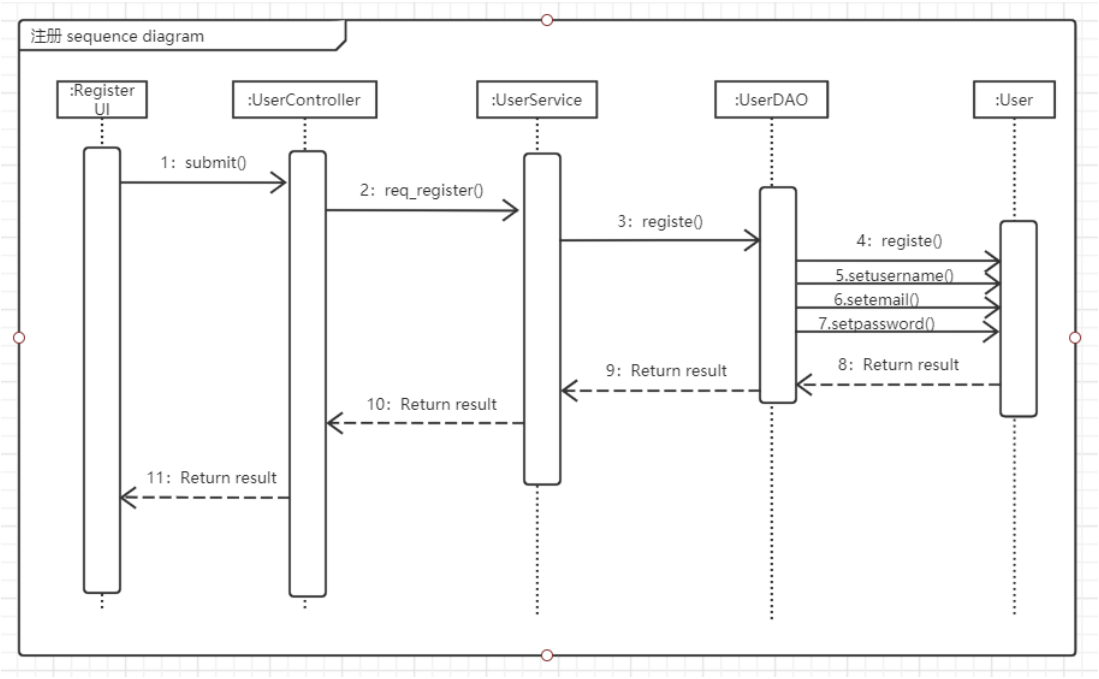
req_login(); 用户提交登录请求，返回用户对象
UserService 用户服务类 login(); 提供用户登录服务，返回用户对象 getCloudAlbum(); 登录后返回用户个人云相册
UserDAO 用户数据访问类 login(); 访问用户信息服务，创建新用户，返回用户对象
User 用户类 用于存储用户信息，用户登录后返回该用户对象

二、注册模块

1. 类图设计



2. 时序图

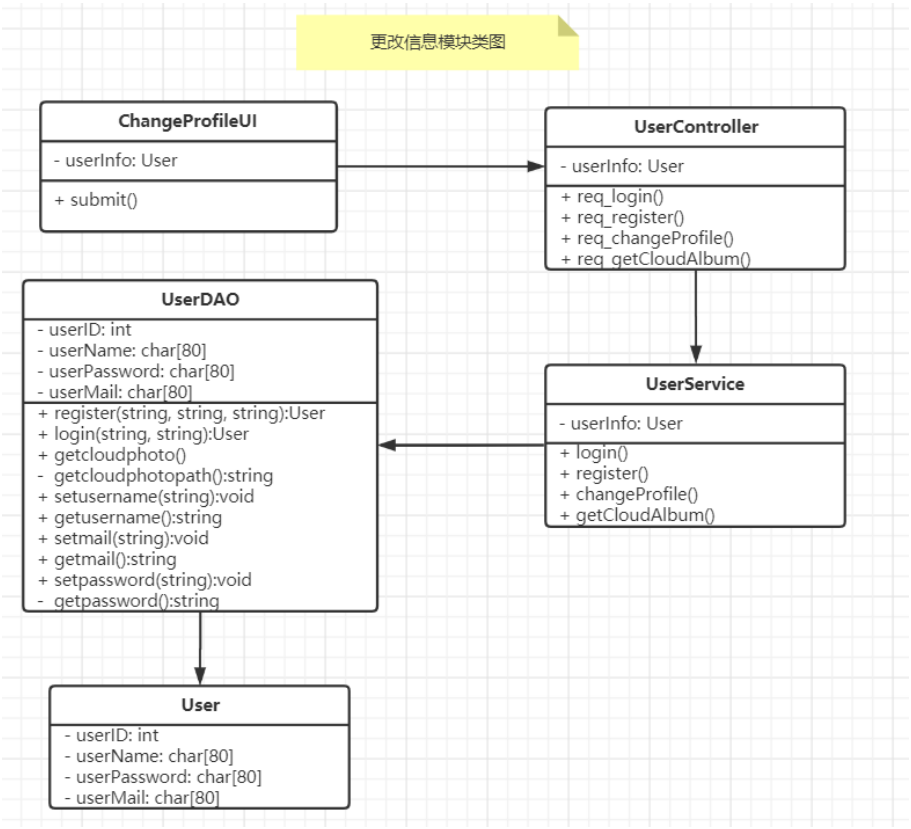


3．流程描述

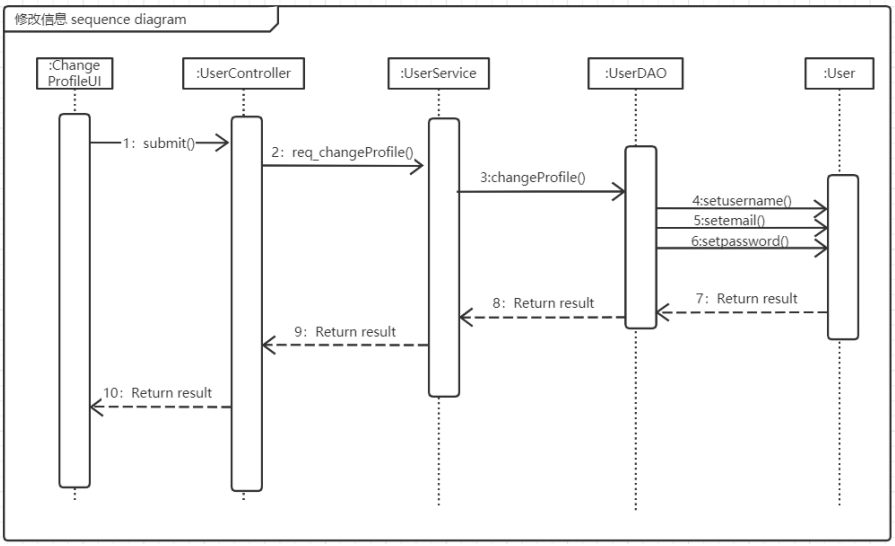
RegisterUI 浏览器注册界面类 submit(); 用户在注册界面提交注册请求，返回注册结果
UserController 用户功能管理类 req_login(); 用户提交注册请求，返回注册结果
UserService 用户服务类 register(); 提供用户注册服务，返回注册结果
UserDAO 用户数据访问类 register(); 访问用户信息服务，创建新用户，返回注册结果 setusername(); 设置用户名 setmail(); 设置用户邮箱 setpassword(); 设置用户密码
User 用户类 用于存储用户信息，用户注册后返回注册结果

三、修改信息模块

1．类图设计



2. 时序图



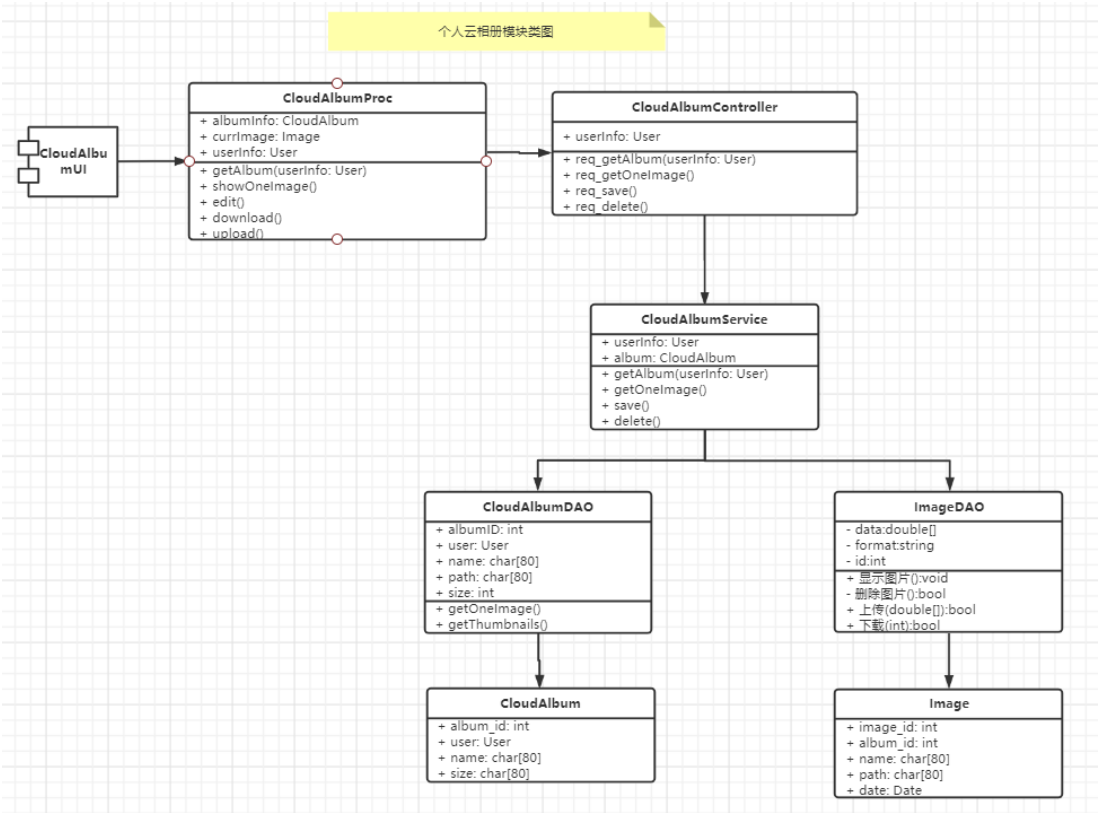
3.流程描述

ChangeProfileUI 浏览器用户信息界面类 submit(); 用户在信息界面提交修改信息请求，返回修改结果

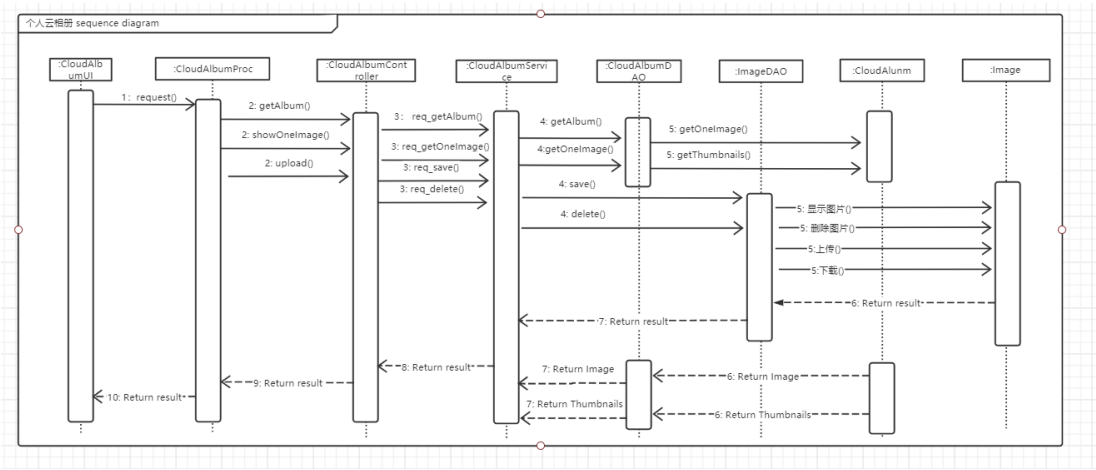
UserController 用户功能管理类 req_changeProfile(); 用户提交修改信息请求，返回修改结果
UserService 用户服务类 changeProfile(); 提供用户信息修改服务，返回修改结果
UserDAO 用户数据访问类 setusername(); 设置用户名 setmail(); 设置用户邮箱 setpassword(); 设置用户密码 返回修改结果
User 用户类 用于存储用户信息，用户修改信息后返回修改信息结果

四、个人云相册模块

1. 类图设计



2. 时序图



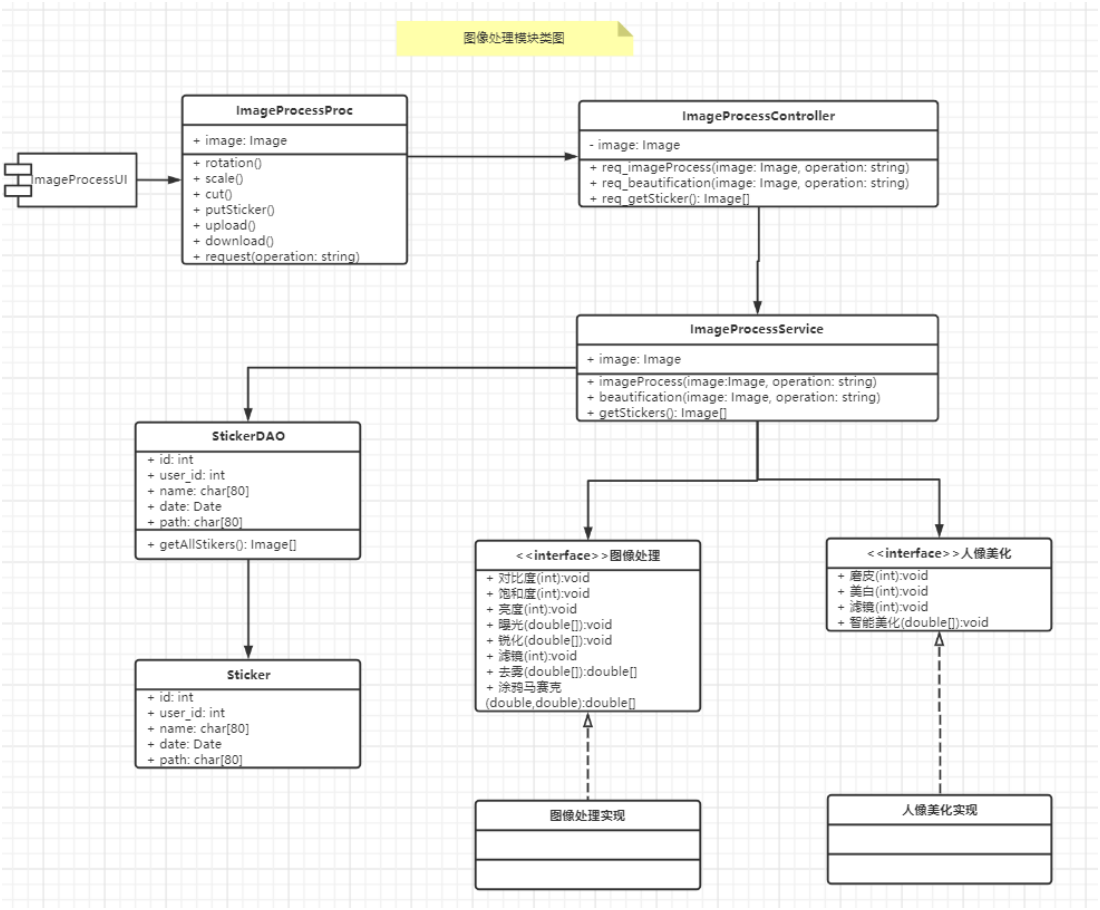
3.流程描述

CloudAlbumUI 浏览器云相册功能界面 request; 用户请求获取云相册内容
CloudAlbumProc 云相册功能类 getAlbum(); 请求获取云相册内容（缩略图） showOneImage(); 请求获取一张图片（完整图） edit(); 请求编辑图片（包含删除操作） upload(); 上传图片 download(); 下载图片
CloudAlbumController 云相册功能控制类 req_getAlbum(); 提交获得云相册内容的请求，返回云相册内容（缩略图） req_getOneImage(); 提交获取一张图片的请求，返回指定图片（完整图） req_sava(); 提交保存图片的请求，用于上传图片 req_delete(); 提交删除图片的请求
CloudAlbumService 拼接服务类 getAlbum(); 提供获取云相册内容服务，返回云相册内容（缩略图） getOneImage(); 提供获取一张图片的服务，返回一张图片（完整图） save(); 提供保存图片服务，返回执行结果 delete(); 提供删除图片服务，返回执行结果
CloudAlbumDAO 云相册数据访问类 getOneImage(); 访问一张图片，返回一张图片（完整图） getThumbnails(); 访问一组缩略图，返回一组缩略图
ImageDAO 图片数据访问类 删除图片(); 删除一张图片，返回执行结果 上传(); 上传一张图片，返回执行结果

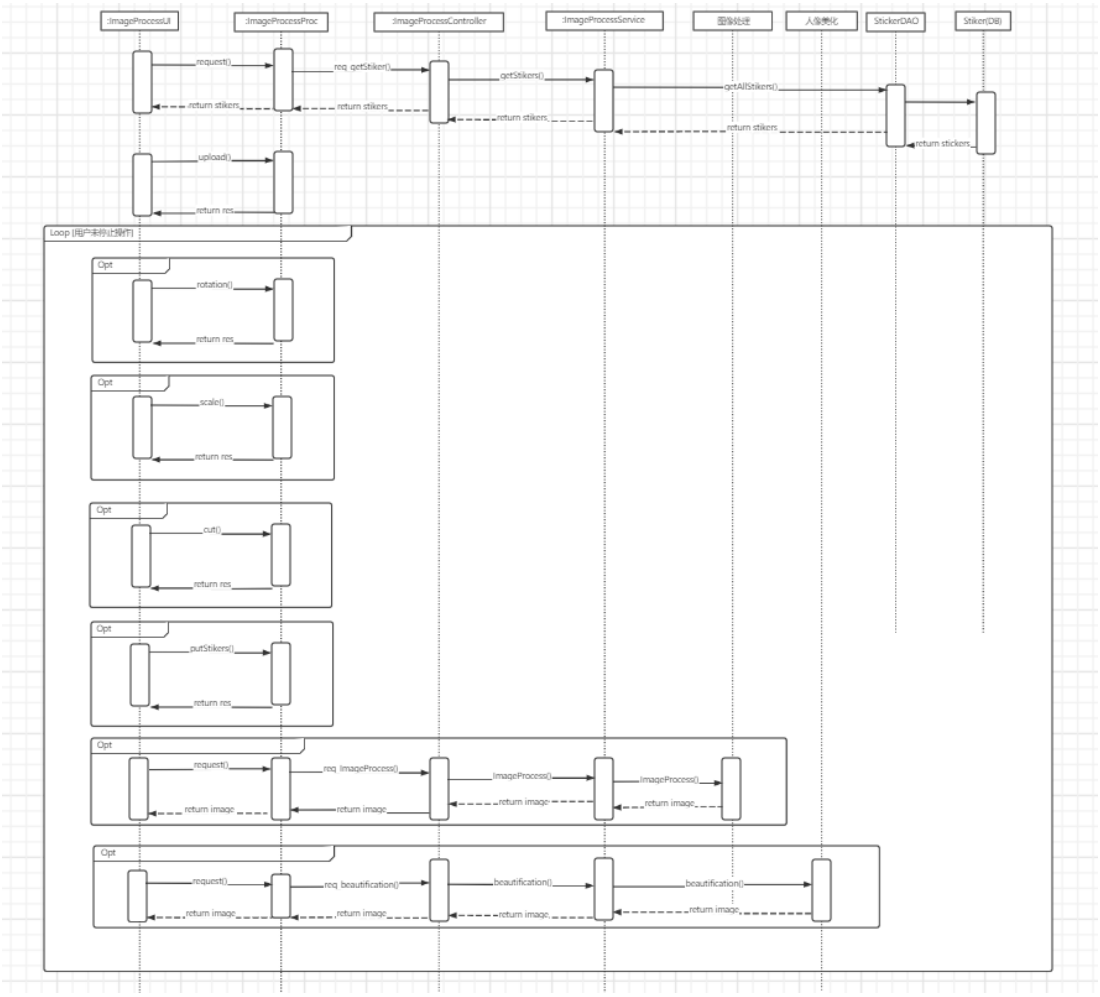
下载(); 下载一张图片，返回一张图片数据（完整图）
CloudAlbum　云相册类 存储云相册信息
Image　　图片类 存储图片信息

五、图像处理模块

1. 类图设计



2. 时序图



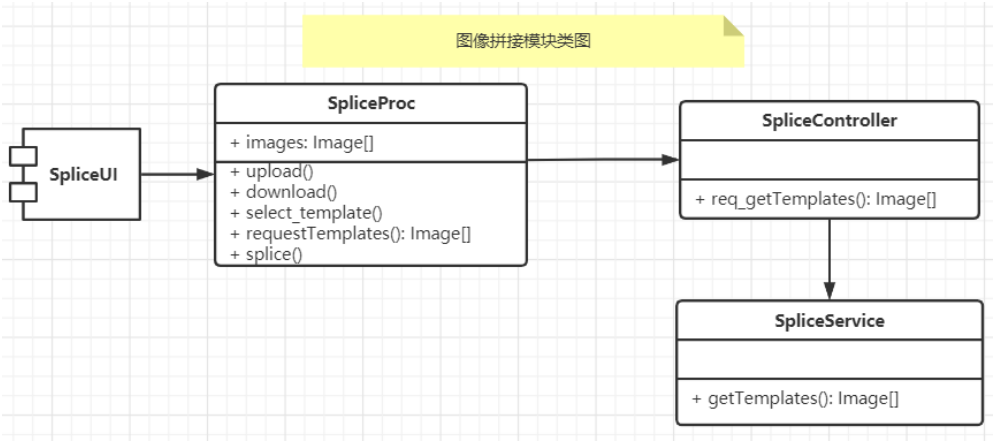
3. 流程说明

ImageProcessUI: 图像处理 UI
ImageProcessProc: 图像处理 UI 的控制类 request(): 根据参数不同请求不同的后端服务，开始时请求所有的可用贴纸，还可以请求复杂的图片处理服务（风格迁移等） upload(): 用户选择图片并上传至浏览器缓存 rotation(): 对用户上传的图片进行旋转(不需要后端服务) scale(): 对用户上传的图片进行缩放(不需要后端服务) cut(): 对用户上传的图片进行裁剪(不需要后端服务) download():下载浏览器缓存中的图片
ImageProcessController: 后端图像处理控制器 req_ImageProcess(): 向服务实现模块请求图片处理服务 req_beautification(): 向服务实现模块请求人像美化服务 req_getStikers(): 向服务实现模块请求所有贴纸

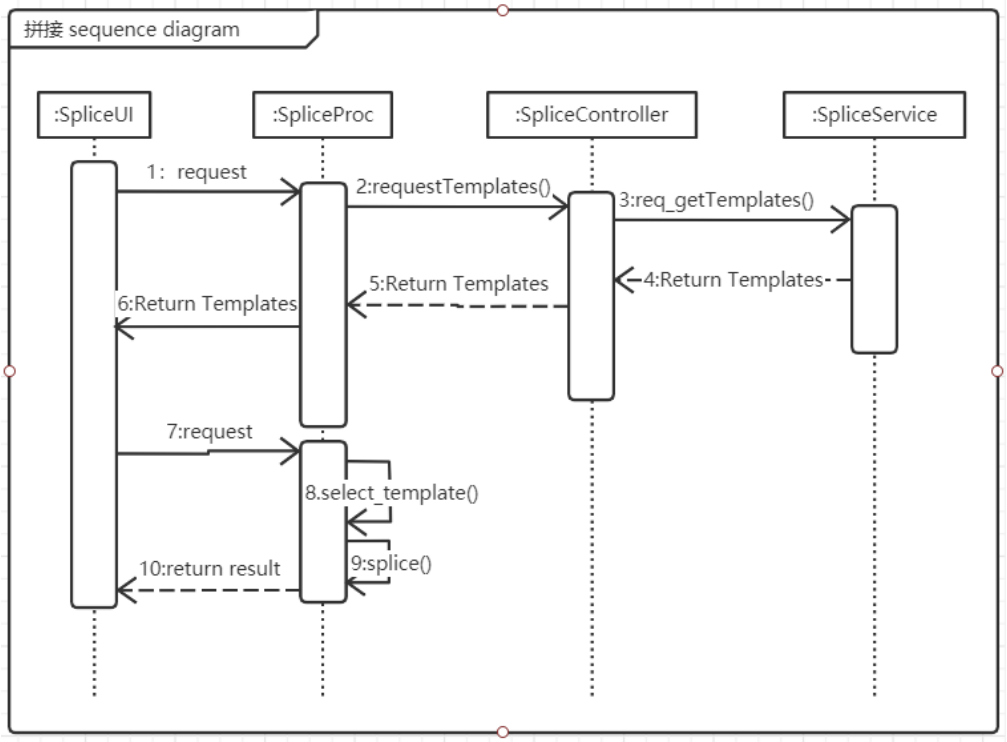
ImageProcessService: 后端服务实现程序 ImageProcess(): 调用图像处理模块中的相应功能对图片进行处理 beautification(): 调用人像美化模块中的相应功能对图片进行处理
图像处理：图像处理模块接口
人像处理：人像处理模块接口
StickerDAO: 与用户相关联的贴纸类 getAllStickers(): 获取所有与该用户相关的贴纸
Sticker: 贴纸实体类

六、图像拼接模块

1. 类图设计



2. 时序图

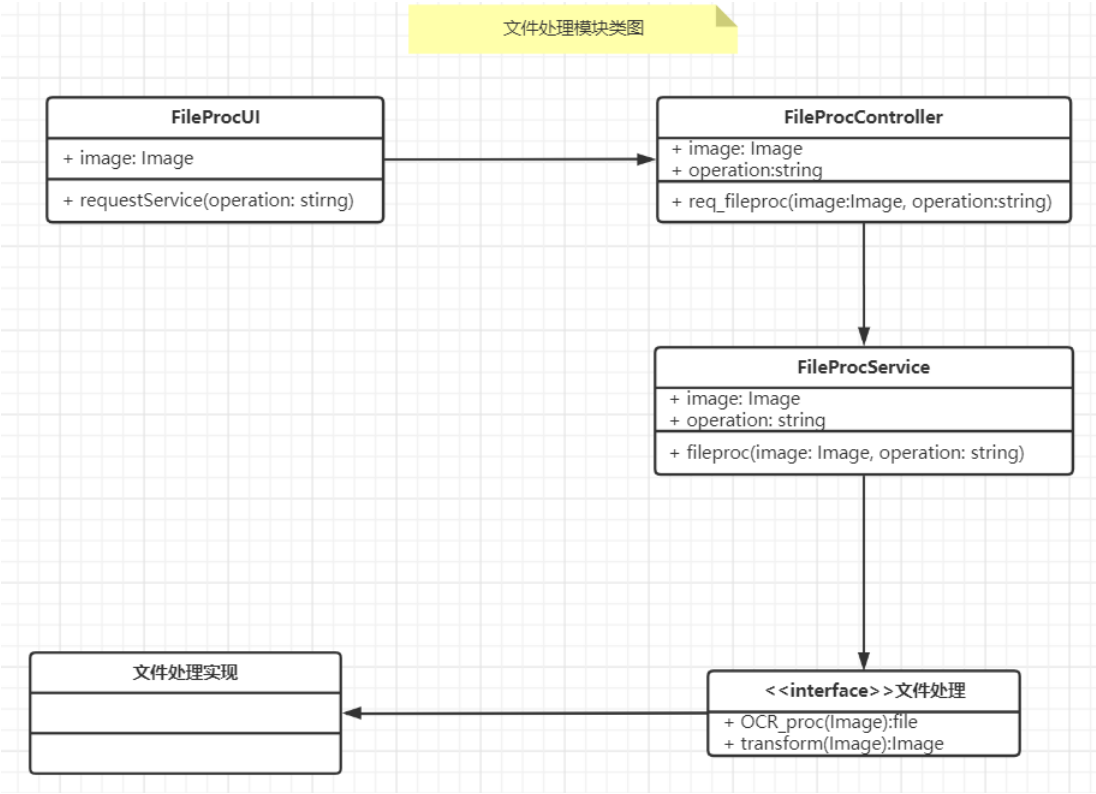


3.流程描述

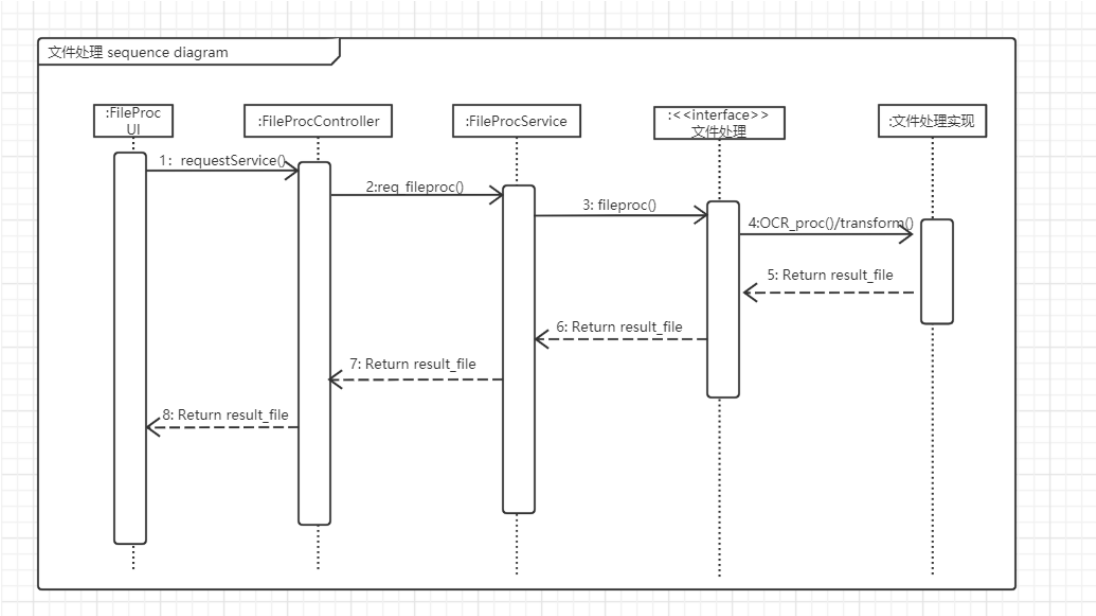
SpliceUI 浏览器拼接功能界面 request; 用户在拼接功能界面提交图片处理请求，返回拼接模板，返回拼接结果
SpliceProc 拼接功能类 requestTemplates(); 提交获得拼接模板的请求，返回拼接模板 select_templates(); 用户选择拼接模板 splice(); 对多图片进行拼接，返回拼接结果
SpliceController 拼接功能控制类 req_getTemplates(); 提交获得拼接模板的请求，返回拼接模板
SpliceService 拼接服务类 返回拼接模板

七、文件处理模块

1. 类图设计



2. 时序图



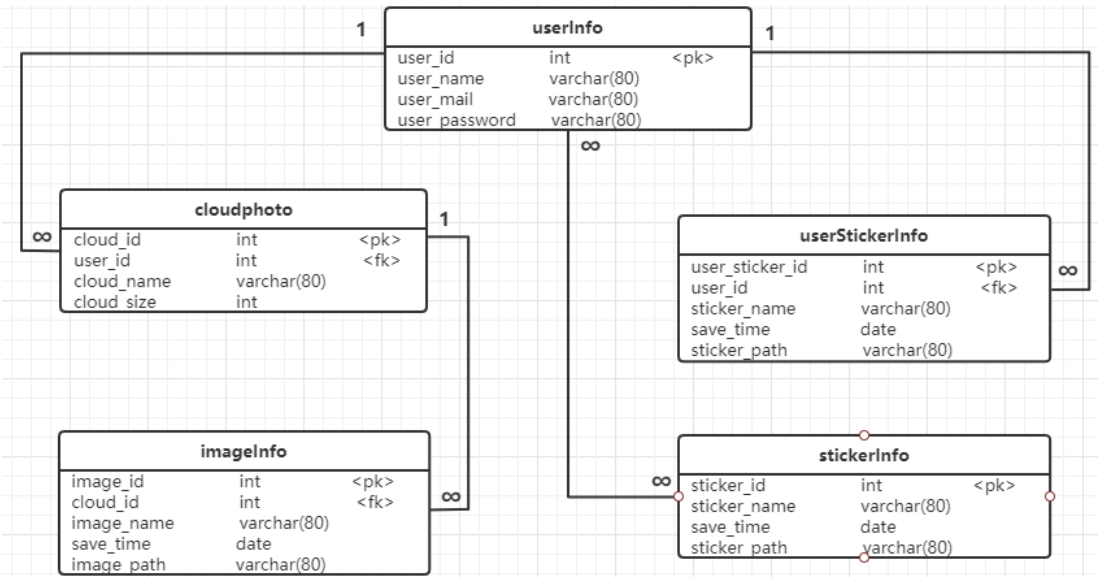
3.流程描述

FileProcUI 浏览器文件处理界面类 requestService(); 用户在文件处理界面提交需要进行文件处理的图片，返回结果文件

FileProcController 文件处理功能管理类 req_fileproc(); 用户提交需要进行文件处理的图片，返回结果文件
FileProcService 文件处理服务类 fileproc(); 提供文件处理服务，返回结果文件
<<interface>> 文件处理 文件处理接口 OCR_proc(); 进行图片 OCR 识别，返回结果文件 transform(); 进行图片转扫描件，返回结果文件
文件处理实现类 用于文件处理，返回结果文件

第六部分 数据库设计

一、数据库整体结构图



整体结构表格清单

序号	表名	注释
1	userInfo	用户信息表
2	cloudphoto	云相册信息表
3	imageInfo	图片信息表
4	stickerInfo	贴纸信息表
5	userStickerInfo	用户贴纸信息表

1 userInfo 表结构

序号	列名	数据类型	注释	约束
1	user_id	INT	记录编号	PK
2	user_name	VARCHAR(80)	记录用户名	not null unique
3	user_mail	VARCHAR(80)	记录用户邮箱	not null unique check(user_email like %@%.com)
4	user_password	VARCHAR(80)	记录用户密码	not null check(length(user_password)>=6)

2 cloudphoto 表结构

序号	列名	数据类型	注释	约束
1	cloud_id	INT	记录相册编号	PK
2	user_id	INT	记录所属用户编号	FK
3	cloud_name	VARCHAR(80)	记录相册名称	default CONCAT(cloud_id, '')
4	cloud_size	INT	记录相册大小(存放图片数量)	default 0 check(cloud_size>=0)

3 imageInfo 表结构

序号	列名	数据类型	注释	约束
1	image_id	INT	记录图片编号	PK
2	cloud_id	INT	记录所属相册编号	FK
3	image_name	VARCHAR(80)	记录图片名称	default CONCAT(image_id, '')
4	save_time	DATE	记录图片存储的时间	default DATE()
5	image_path	VARCHAR(80)	记录图片存储路径	not null unique

4 stickerInfo 表结构

序号	列名	数据类型	注释	约束
1	sticker_id	INT	记录贴纸编号	PK
3	sticker_name	VARCHAR(80)	记录贴纸名称	default CONCAT(sticker_id, '')
4	save_time	DATE	记录贴纸存储的时间	default DATE()
5	sticker_path	VARCHAR(80)	记录贴纸存储路径	not null unique

5 userStickerInfo 表结构

序号	列名	数据类型	注释	约束
1	user_sticker_id	INT	记录贴纸编号	PK
2	user_id	INT	记录所属用户编号	FK
3	sticker_name	VARCHAR(80)	记录贴纸名称	default CONCAT(sticker_id, '')
4	save_time	DATE	记录贴纸存储的时间	default DATE()
5	sticker_path	VARCHAR(80)	记录贴纸存储路径	not null unique

6 外键清单

外键名称	父表	父键列	子表	外键列	关系	说明
cloudphoto_ibfk_1	userInfo	user_id	cloudphoto	user_id	1..*	一个用户可以拥有多个云相册，一个云相册只可以对应一个用户
imagemsg_ibfk_1	cloudphoto	cloud_id	imageInfo	cloud_id	1..*	一个相册可以有多张图片，一张图片只可以放在一个相册里
userstickerinfo_ibfk_1	userInfo	user_id	userstickerInfo	user_id	1..*	一个用户可以有多张自定义贴纸，一张自定义贴纸只可以对应一个用户