

**软件工程综合课程设计**

――开发手册

**题 目**  图像识别处理系统

**院 系**  计算机科学与技术学院

**专 业**  人工智能

**学生姓名**  颜劭铭，黄喆琨，阳松霖

邹逸飞，王赟

**学 号**  162050127；162050114；162050113；

162050112；162050130

**2022 年 12 月 16 日**

目录

[1. 引言 4](#_Toc18754)

[1.1. 编写目的 4](#_Toc18134)

[1.2. 软件背景 4](#_Toc16401)

[2. 工具安装及环境配置 4](#_Toc29486)

[2.1. MySQL 4](#_Toc24408)

[2.2. Python 5](#_Toc9352)

[2.3. Vue.js 5](#_Toc10592)

[3. 开发环境搭建 5](#_Toc2386)

[3.1. 开发工具配置 6](#_Toc6907)

[3.2. 导入工程 8](#_Toc6830)

[4. 运行系统 9](#_Toc25083)

[4.1. 修改数据库链接 9](#_Toc19349)

[4.2. 发信人/收件人设置 9](#_Toc22580)

[4.3. 端口配置 10](#_Toc32323)

[4.4. 整体运行 11](#_Toc28610)

[5. 部署系统 12](#_Toc23017)

[6. 编程规约 12](#_Toc25209)

[6.1. 命名规约 12](#_Toc10991)

[6.1.1 模型名称 12](#_Toc17808)

[6.1.2 配置对象命名 12](#_Toc2640)

[6.1.3 常量名 13](#_Toc27672)

[6.2. 常量规约 13](#_Toc11480)

[6.3. 注释规约 13](#_Toc5565)

[6.4. 其他 13](#_Toc332)

[7. 异常日志 13](#_Toc5337)

[7.1. 异常处理 14](#_Toc15547)

[7.1.1 异常捕获 14](#_Toc11752)

[7.1.2 异常处理 14](#_Toc11926)

[7.2. 日志处理 14](#_Toc17374)

[8. MySQL数据库规约 15](#_Toc16955)

[8.1. 建表规约 15](#_Toc16371)

[8.2. 索引规约 15](#_Toc530)

[8.3. SQL规约 16](#_Toc22735)

[9. 工程规约 16](#_Toc20601)

[9.1. 应用分解 16](#_Toc15438)

[9.2. 数据源 17](#_Toc27907)

[9.3. 打卡记录 17](#_Toc30772)

[9.4. 项目 18](#_Toc20708)

[10. 安全规约 18](#_Toc8569)

# 引言

以编写目的和软件背景两个维度展开，对整个项目的的开发目的，初步预期，和系统主题进行一个方向的定调。

## 编写目的

当对图像识别处理系统添加新的功能时，开发人员可根据该文档更快更有效地开发。

## 软件背景

图像识别处理系统的基础应用环境是在宿舍楼下或者写字楼下面对学生或员工进行打卡。我们提供了基础的登录，注册功能（包含管理员权限），对账号正确登录却不满足对应脸型的非对象目标会向对应的管理员邮箱发送警告邮件。

网站开发所依赖的工具及软件环境包括MySQL，JavaScript，BoostStrap，Jinjia2，Python，vue，npm, Chrome。

# 工具安装及环境配置

利用MySQL、Python、Java、JavaScript、Vue.js等工具的来完成对FaceRecognition整体的架构，通过不同工具相配合形成FaceRecognition的具体功能实现，并打包形成较为完备的界面和交互系统，根据用户所需以达成所求。

## MySQL

MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。

MySQL官方网站https://www.mysql.com/。

* 版本

社区版8.0.27

* 端口

3306

* 账户

admin 123456（默认初始密码可更改为您设置的密码）

* 后端依赖数据

名称为Face或Face\_bi。

* 测试源数据库

自行构建并测试，名称端口可随意，但不可同时为Face和3306。

## Python

Python由[荷兰](https://baike.baidu.com/item/%E8%8D%B7%E5%85%B0/190469?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/Python/_blank)数学和计算机科学研究学会的吉多·范罗苏姆于1990年代初设计，作为一门叫做ABC语言的替代品。Python提供了高效的高级数据结构，还能简单有效地面向对象编程。Python语法和动态类型，以及解释型语言的本质，使它成为多数平台上写脚本和快速开发应用的编程语言，随着版本的不断更新和语言新功能的添加，逐渐被用于独立的、[大型项目](https://baike.baidu.com/item/%E5%A4%A7%E5%9E%8B%E9%A1%B9%E7%9B%AE/3986637?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/Python/_blank)的开发。

此项目我们使用Anaconda提供的虚拟Python环境。

版本为3.6。

## Vue.js

Vue.js是一套构建用户界面的渐进式框架。Vue 只关注视图层， 采用自底向上增量开发的设计。

Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

命令行下安装npm install -g @vue/cli。

* 1. **Flask**

Flask是一个使用python编写的轻量级Web应用框架，其WEGI工具箱采用Werkzeug，模板引擎使用Jinja2。Flask实用BDB授权。

工作原理如下图所示Flask的基本模式为在程序里将一个视图函数分配给一个URL，每当用户访问这个URL时，系统就会执行给该URL分配好的视图函数，获取函数的返回值并将其显示到浏览器上。下图1为Flask原理示意图。

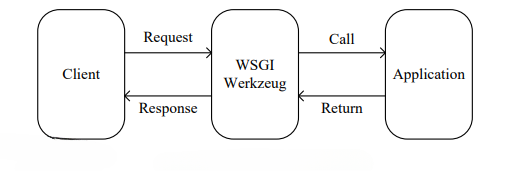


图1 Flask原理图

# 开发环境搭建

分为开发工具的配置及导入工程两个模块，根据对前文所用到工具进行相对应的配置，能使整个系统初步运转，再根据工程的导入，实现个性化的创新过程。

## 开发工具配置

* 配置Python
* 首先选择Anaconda

下图2为Anaconda初始界面。

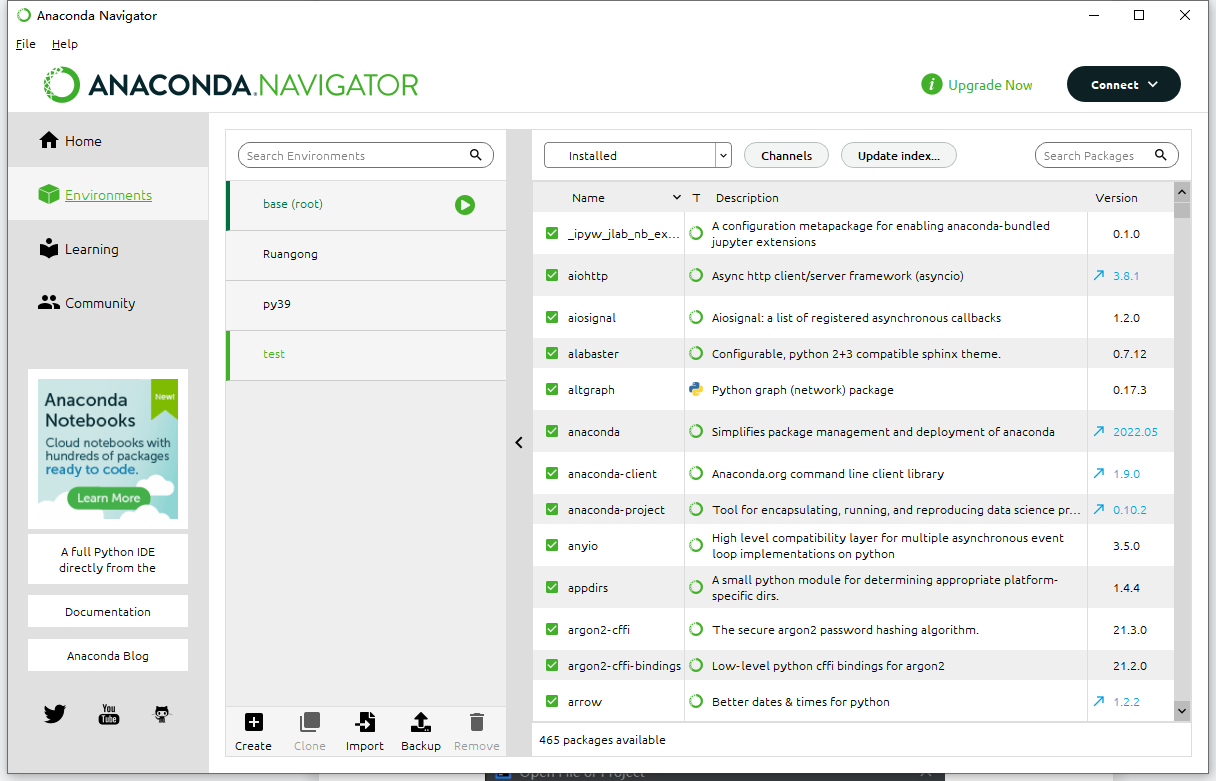


图2 Anaconda视图

我们选择下方Create按钮，选择我们所需的Python版本3.6，名称随意。（尽量简单好记方便调用）。

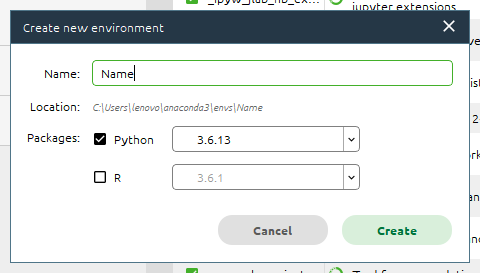


图3 环境视图

上图3为进入环境后命名过程。在安装了环境后，我们就要安装对应的库我们先通过对应的指令进入虚拟环境。

我们运用“conda activate”和“activate XXX”（XXX为你设定的环境名）来进入虚拟环境。下图4为进入cmd后界面。

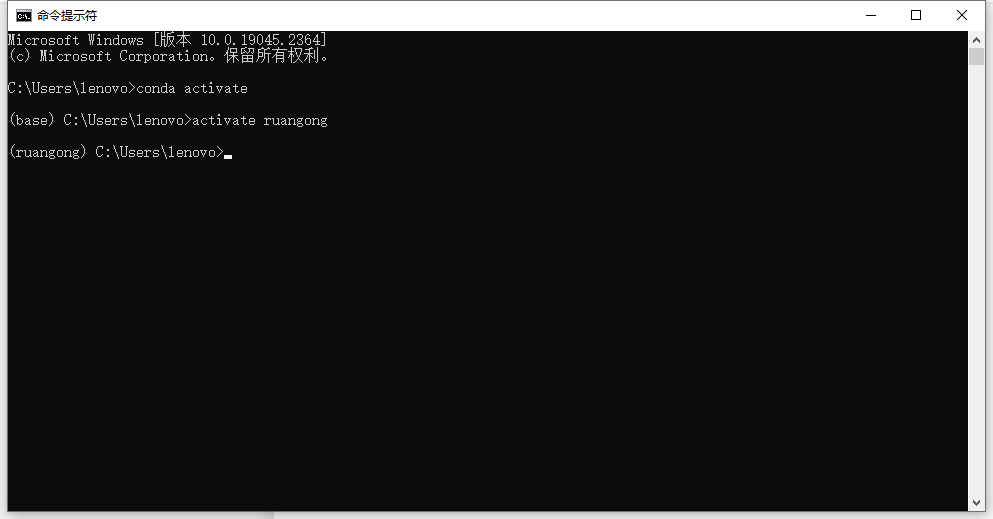


图4 命令行

之后我们运用pip install xxx（此处xxx为不同库的名称）来安装。

所有要求安装的库名已在.requirement文档中给出。

* 安装过程中可能出现的问题

1. 关于包安装速度过慢，大量红色报错问题。

解决方案：换用国内镜像源，在指令后增加“-i http://pypi.douban.com/simple/ --trusted-host=pypi.douban.com/simple”即可解决（该指令是换用国内豆瓣源）。

其他镜像源

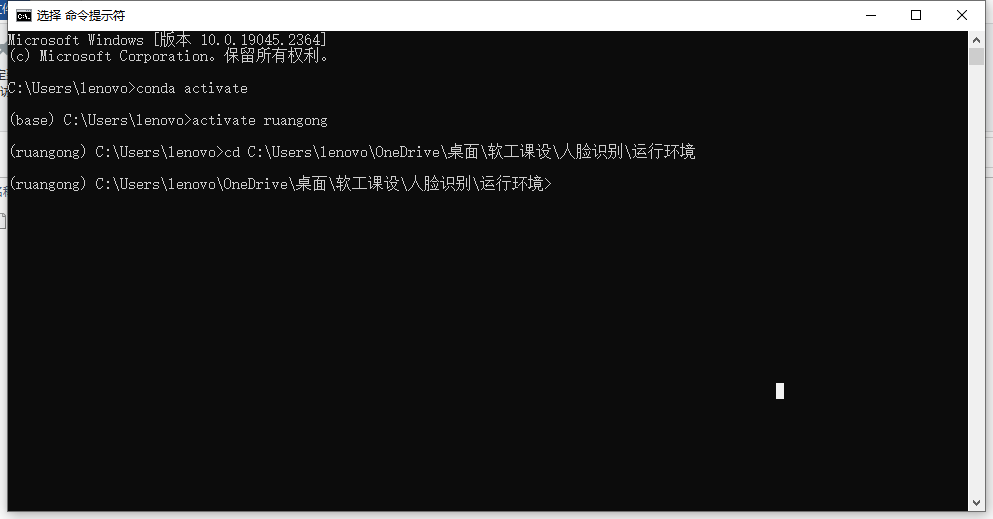
清华：https://pypi.tuna.tsinghua.edu.cn/simple/

阿里：http://mirrors.aliyun.com/pypi/simple/

中科大：http://pypi.mirrors.ustc.edu.cn/simple/

1. 关于在下载库dlib==19.8.1时无法直接pip下载

我们在提交的文件中已经给出了该库的安装包，我们通过cd指令跳转到改文件位置，输入指令“pip install dlib-19.8.1-cp36-cp36m-win\_amd64.whl”。

 图5 Dlib库安装

上图5为输入界面。输入指令后即可成功安装，至此，Python环境配置完毕。

## 导入工程

* 导入工程

在VSCode中选择打开文件夹，选择对应的路径，或者直接ctrl+O打开路径。下图6为示例路径。



图6 工程导入

在工程视图中就可以看到整个项目的代码。下图7为整体工程视图。

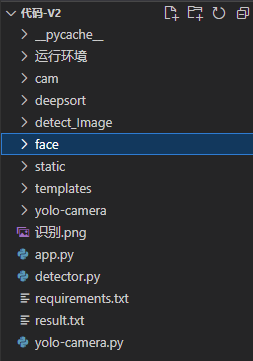


图7 工程视图

* 配置Python

我们在右下角的选项中点击Python版本处。下图8为版本选取栏。

6

图8 配置

在顶部选择我们需要的版本。

下图9为版本选取区域展示以及示例图，如此处我们选择“ruangong”环境。

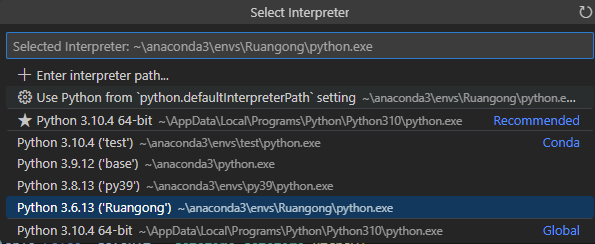


图9 选择正确环境

# 运行系统

遵循修改数据库链接、发信人/收信人设置、端口配置和整体运行四个步骤，并能在运行系统的同时与部署相勾连，以期能更快的完成对整个系统的部署，并根据端口的配置提升其各个变量的匹配程度。

## 修改数据库链接

首先，在MySQL中新建数据库，此处我们选择默认数据库，名称为Root，密码可以设定为XXX（XXX为你设置的密码，给出代码中默认为123456）如果你对密码做出修改，请修改代码中50行“mysql：//root：XXX@localhost/xxx”。

其中XXX为你设置的密码，xxx为你设置的存储成员的数据表单名（默认为test）。

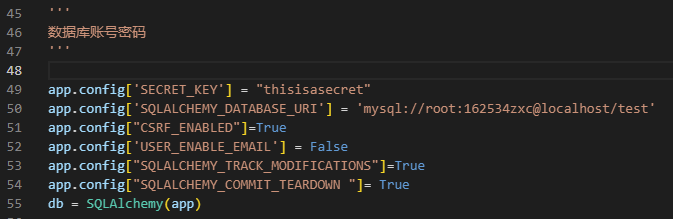


图10 数据库链接

## 发信人/收件人设置

* 发信人设置

实现方式为Flask\_mail，此处默认发件人为“409054990@qq.com”，若想改为用自己设定的邮箱，可更改56行代码将邮箱替换为自己的，同时更改57行“MAIL\_PASSWORD”部分为自己的密码（注意，该密码并不是你的邮箱登录密码而是smtp服务密码，具体可在对应邮箱设置处了解详情）。下图11为发信设置处代码。

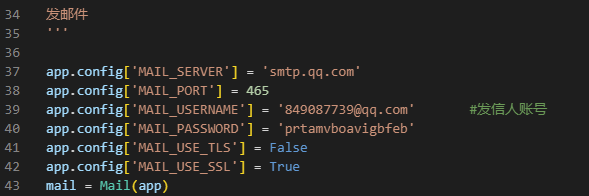


图11 发信设置

* 收信人设置

在更改88行的RECIPENTS处可更改收件人（默认为849087739@qq.com）修改为自己所需的收件人邮箱即可。



图12 收信人

## 端口配置

* FLASK接口定义

以打卡页面为例，除此以外还有图像识别、视频识别、摄像头识别、上传头像界面。下图13为打卡界面接口代码展示。



图13 打卡

* Vue界面实现

我们截取部分打卡界面为例。同样下图14展示此处Vue界面的接口代码。

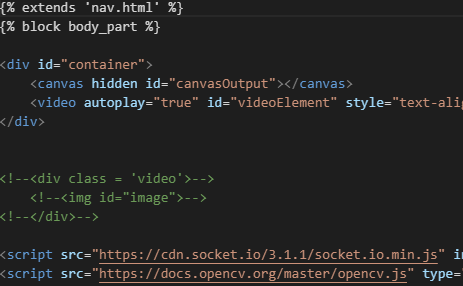


图14 Vue代码

* 后端接口定义

1. 寻找函数处理需求。
2. 请求响应和会话。
3. 重新定向与错误处理。
4. 准备交互。
5. 打通前后端交互流程。
6. 关于SQLAIchemy和PostMan的运用此处不再赘述，是两个很有用的工具。下图15为后端连接端口代码展示。

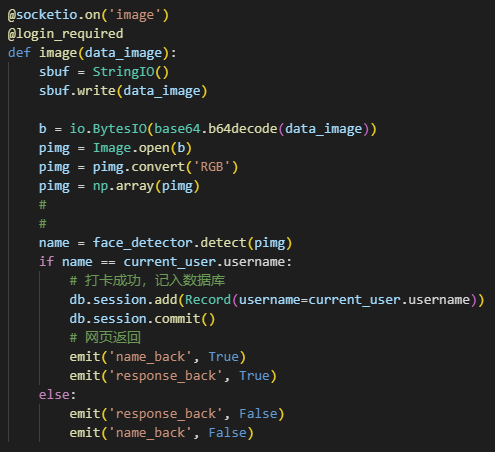


图15 后端接口

## 整体运行

在终端输入“python app.py migrate”得到以下网址。下图16为运行指令后终端给出的结果。

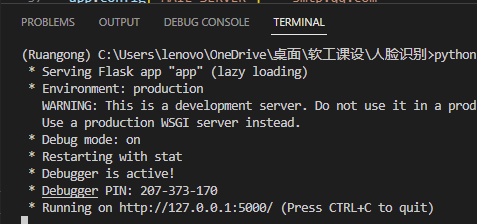


图16运行结果

点击最后一行的网址即可进入结果界面

# 部署系统

Flask基本的机构树为

* App.py：项目管理文件，通过它管理其他项目。
* Static：存放静态文件。
* Templates：放置html模板文件。

步骤为

1. 配置sql数据库，加载配置文件并针对整个项目定义全局db。
2. 定义model模型与数据库交互。
3. 声明蓝图并将其注册进入app中。
4. 通过路由器传递参数。

# 编程规约

分为命名规约、常量规约、注释规约及其他四个板块，实现对各个模块间串联的媒介，限定不同模块间的接口，从而能更好的对项目的各个模块和脉络有清晰的层次。

## 命名规约

选择一种命名约定并遵循它。只要保持一致，就可以是camelCase（小驼峰），PascalCase（大驼峰），snake\_case（下划线）或其他约定形式。许多编程语言在命名约定方面都有自己的传统。因此在我们的系统中也对名称进行了相关规约。

## 6.1.1 模型名称

* 函数名统一使用驼峰命名法，通常由2~3个单词组成，每个单词首字母大写。
* 函数名能简洁直接的表明函数体的功能。
* 有明确功能定义的模型放在统一的文件夹下并以统一的规则命名，如组件Logout, Menu, Link。

## 6.1.2 配置对象命名

* 函数名统一使用驼峰命名法，通常由2~3个单词组成，每个单词首字母大写。
* 函数名能简洁直接的表明函数体的功能。
* 有明确功能定义的模型放在统一的文件夹下并以统一的规则命名。

## 6.1.3 常量名

常量名尽量不使用除字母和数字外的符号，纯字母名称优先，常量名全部大写。

## 常量规约

不允许任何魔法值（即未经定义的常量）直接出现在代码中；按常量功能进行归类，分成不同的类对常量进行维护，不要仅用一个常量类维护所有常量。

## 注释规约

注释规约属于对格式一致化的要求，在编写程序的过程中，往往是跨时间和设备的，因此同组哪怕是同一个人之间都可能会因为注释的编写对程序编写的体验和速率造成较大影响。

* 关于类或类中方法的注释一律使用#context的形式，不允许使用//。
* 所有的抽象方法，除了需要在注释中说明参数，返回值，异常外，还需说明方法所实现的功能；代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑等的修改；注释掉的代码需附上注释说明而不是简单的注释掉。
* 与其使用半吊子的英文，不如用中文注释把内容说清楚，专有名词与关键字保持与英文原文一致即可。

## 其他

这部分主要包括一些与信安相关的错误和小问题的修改和规约。

* 后台输送给页面的变量必须加$!{var}——中间的感叹号。
* 任何数据结构的构造或初始化，都应指定大小，避免数据结构无限增长吃光内存；对于“明确停止使用的代码和配置”，如方法、变量、类、配置文件、动态配置属性等需从程序中清理出去，避免造成过多垃圾。

# 异常日志

分为异常处理部分和日至处理部分，整个系统不可避免的会存在异常操作，这也是我们在不断改进完善的动力，在通过对异常日志的记录和处理，能更好的规避异常操作的同时，对系统的鲁棒性有进一步的提升。

## 异常处理

关于异常处理的规约，使得对异常的处理能更快捷和有效，在此主要囊括捕获和处理两个模块，并根据异常的不同类型进行自适应的处理和完善。

## 7.1.1 异常捕获

flask内部异常通过继承这个HTTPException类来处理。

* Get\_headers方法定义了这个返回的响应头，返回的是html文档。
* Get\_body方法定义了返回的响应体，对应的也是一段html内容。

最后在response中将响应体状态码，响应头定义好返回。

分析至此，其实这个HTTPException中做的事也不难理解，就是定义好响应体，状态码，还有响应头，做了一个返回。当然这个类返回是html类型的，现在前后端分离交互都是json形式的返回，所以我们可以继承自这个类，定义我们自己的异常处理类。

## 7.1.2 异常处理

* 异常分为客户端异常和服务端异常，flask 中的异常处理分为三步走：异常注册、异常触发、异常处理。先看段代码，边测边写。
* 捕获异常是为了处理掉异常，不要捕获了却什么都不处理而抛弃之，如果不想处理异常，请将该异常抛给它的调用者。最外层的业务使用者，必须处理异常，将其转化为用户可以理解的内容。
* 客户端全局异常捕获，@app.errorhandler 向 flask 注册了 HTTPException 异常以及该异常的处理方法 handle\_exception(e)。客户端的所有请求他都囊括了，404、405、400 等等，一旦这些异常触发，都会执行到 handle\_exception(e) 方法中，最后将异常信息返回给客户端。
* 还可以在 HTTPException 基础上[自定义异常](https://so.csdn.net/so/search?q=%E8%87%AA%E5%AE%9A%E4%B9%89%E5%BC%82%E5%B8%B8&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/yy_diego/article/details/_blank)信息，比如在参数不符合要求，抛出异常。当子类异常与父类异常同时存在时，优先让子类异常捕获。

## 日志处理

主要包括对日志的存储空间和时间的规约，并且对日志的保存和管理员运作访问进行明确的认知和界定。

* 日志模块基础配置，如：日志存放地址、日志记录格式、日志等级
* LogHandle 封装: 自定义日志的格式、等级设置;分别使用了以文件大小分割和时间分割日志方式

# MySQL数据库规约

分为建表规约、SQL规约和索引规约，主要体现在导入时数据库的一致性，便于对数据后续属性、数据等相关变量的操作，也能够清晰数据源导入的方向，为后续的建模提供基础。

## 建表规约

通过对表格进行各个维度的规约，对于系统的提升主要体现在界面清晰度、编写舒适度和程序的流畅度的这三个方面。建表规约如下：

* 表名、字段名必须使用小写字母或数字；禁止出现数字开头，禁止两个下划线中间只出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。
* 表名不使用复数名词；库名与应用名称尽量一致；单表行数超过500万行或者单表容量超过2GB，才推荐进行分库分表；禁用保留字，如 desc 、 range 、 match 、 delayed 等，请参考 MySQL 官方保留字。
* 小数类型为 decimal ，禁止使用 float 和 double， float 和 double 在存储的时候，存在精度损失的问题，很可能在值的比较时，得到不正确的结果。如果存储的数据范围超过 decimal 的范围，建议将数据拆成整数和小数分开存储。
* varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度大于此值，定义字段类型为 text ，独立出来一张表，用主键来对应，避免影响其它字段索引效率。
* 合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检索速度。

## 索引规约

索引规约的任务在各个步骤都极为重要，若是没有进行协调统一的处理，接口的连接部分可能会出现对不齐或是对不上的阻滞，并会使得不同组员的合作出现问题。索引规约如下：

* 业务上具有唯一特性的字段，即使是组合字段，也必须建成唯一索引。
* 超过三个表禁止join。需要join的字段，数据类型保持绝对一致；多表关联查询时，保证被关联的字段需要有索引。
* 利用覆盖索引来进行查询操作，来避免回表操作；如果有 order by 的场景，请注意利用索引的有序性。 order by 最后的字段是组合索引的一部分，并且放在索引组合顺序的最后，避免出现 file \_ sort 的情况，影响查询性能。正例： where a =? and b =? order by c; 索引： a \_ b \_ c。
* 反例：索引中有范围查找，那么索引有序性无法利用，如： WHERE a >10 ORDER BY b; 索引a \_ b 无法排序。
* 在varchar字段上建立索引时，必须指定索引长度，没必要对全字段建立索引，根据实际文本区分度决定索引长度。

## SQL规约

SQL规约主要体现在数据的导入过程中，系统与数据库的交互过程，并且SQL的由于语句方面的格式限定需要定义规约先行的规范来保证系统架构的合理展开。SQL规约如下：

* 不要使用count(列名)或count(常量)来替代count()；表达是与否概念的字段，必须使用 is \_ xxx 的方式命名，数据类型是 unsigned tinyint（1 表示是，0 表示否） ，此规则同样适用于 odps 建表。
* 表名、字段名必须使用小写字母或数字 ； 禁止出现数字开头，禁止两个下划线中间只出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。

正例： getter \_ admin ， task \_ config ， level 3\_ name。

反例： GetterAdmin ， taskConfig ， level 3 name。

* 当某一列的值全是NULL时，count(col)的返回结果为0，但sum(col)的返回结果为NULL，因此使用sum()时需注意NPE问题。
* 为便于测试，不要使用外键，所有外键会在测试时删除；varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度大于此值，定义字段类型为 text ，独立出来一张表，用主键来对应，避免影响其它字段索引效率。
* 禁止使用存储过程，存储过程难以调试和扩展，更没有移植性；单表行数超过 500 万行或者单表容量超过 2 GB ，才推荐进行分库分表。
* 所有的字符存储与表示，均以utf-8编码。
* 数据订正时，删除和修改记录时，要先select，避免出现误删除，确认无误才能执行更新语句。

# 工程规约

体现在应用分解、数据源、打卡记录和项目这四个部分，最直白的体现就是在整个工程项目的存储处理上都是有说法和讲究的，这也是我们小组最后梳理和再运行部分来完善工程的体系架构的关键。

## 应用分解

根据web架构的启发，对于整个工程的应用分层包括Flask、Vue、和app.py这三个部分定义都实行了较为严格的调用。

* 路由 - 去找函数处理请求。
* 请求、响应和会话。
* 重定向与错误处理。
* 前端简单制作form表单 - 准备交互。
* 打通前后端交互流程。

## 数据源

数据源部分主要体现在和SQL语句数据库的相关联动，以及对应的操作处理方法，在导入数据库和数据库各个功能驱动上体现数据源前端的路由效果。

* 数据源部分实现数据库中数据的取用，需要配置数据库地址，数据库驱动，数据库账户。配置好数据源后即可根据数据库中的记录绘制图表并生成报表。
* 数据源前端路由包括添加，修改，删除，索引，显示，查询，同步源数据，链接，保存。
* 数据源配置信息通过json传递。

表1 数据源具体信息

|  |  |
| --- | --- |
| Test | 数据源名称 |
| conn\_url | 数据库conn链接 |
| Admin | 管理员 |
| db\_type | 数据库类型 |
| username | 数据库账户 |
| password | 数据库账户 |

## 打卡记录

打卡记录部分也是图像识别处理系统可视化处理的关键所在，主要体现在打卡记录路由的展示。

* 打卡记录可以实现对于以往所有打卡信息的查询。
* 打卡记录通过json传递。

表2 打卡记录具体信息

|  |  |
| --- | --- |
| create\_date | 创建时间 |
| Admin | 管理员 |
| field\_cnames | 数据表名 |
| is\_public | 是否公开 |
| report\_id | 报表唯一标识id |
| report\_sql | sql查询语句 |

## 项目

项目部分是对整个系统的包装和提取，并且对整个工程文件的格式化和系统化处理也是尤为关键的部分，同时不同项目间的权限问题也值得注意。

* 用户可以创建一个项目，在项目里同步实现绘图和报表生成，不同项目之间数据不共享互不干扰，但数据源可共享。报表可选择为公开，与其他人一同编辑。
* 项目包含的路由有创建，更新，查询，删除，索引。
* 项目部分通过json传递。

表3 项目具体信息

|  |  |
| --- | --- |
| created\_at | 创建时间 |
| Admin | 管理员 |
| creator\_username | 创建者名称 |
| is\_public | 是否公开 |
| project\_id | 标识 |
| project\_name | 名称 |

# 安全规约

体现在隶属于用户个人的页面或者功能必须进行权限控制校验、用户敏感数据禁止直接展示，必须对展示数据脱敏、禁止向HTML页面输出未经安全过滤或未正确转义的用户数据这三个板块，是对系统鲁棒性的又一次提高和检验。