服务端软件开发文档

# 软件简介

服务端软件作为整个人脸识别系统的后台，提供稳定的数据存储服务，并对外暴露一系列HTTP接口供第三方程序调用。在此基础上，提供一套完整的UI界面，进行人员、设备和匹配记录的增删改查。

## 主要功能

|  |  |
| --- | --- |
| 名称 | 说明 |
| 数据存储 | 人员、设备、匹配记录以及各种绑定关系 |
| 对外接口 | REST API + 少量自定义HTTP接口 |
| UI界面 | 人员、设备、匹配记录的管理 |

## 工具介绍

|  |  |
| --- | --- |
| 开发环境 | Windows 10, Linux |
| 开发工具 | Pycharm |
| 部署环境 | Windows, Linux |
| 部署工具 | docker, docker-compose |
| 额外软件 | 关系型数据库(Sqlite, Postgresql), 非关系型数据库(Redis) |
| 编程语言 | Python, HTML, CSS, JavaScript |

# 软件设计

## 整体设计

### 需求分析

|  |  |  |
| --- | --- | --- |
| 功能模块 | 备注 | |
| 人脸数据管理 | 人员录入 | 单个录入 |
| 批量导入 |
| 人员编辑 |  |
| 人员删除 |  |
| 人员查询 |  |
| 设备管理 | 设备注册 |  |
| 设备编辑 |  |
| 设备删除 |  |
| 心跳保持 |  |
| 匹配记录管理 | 记录上传 |  |
| 记录查询 |  |
| 考勤报表 |  |
| 第三方接口 | 增量式数据更新接口 |  |
| 人员数据接口 |  |
| 设备数据接口 |  |

### 交互方式

由于服务器无法主动向Android端发送数据，所有的Android设备均主动定时向服务端软件发送网络请求。

## 详细设计

### 人脸数据管理

|  |  |
| --- | --- |
| 名称 | 备注 |
| 单个人员录入 |  |
| 人员批量导入 | zip压缩文件 |
| 单个人员删除 | 软删除 |
| 人员批量删除 | 软删除 |
| 人员编辑 |  |
| 人员查看 |  |
| 人员查询与筛选 |  |

### 设备管理

|  |  |
| --- | --- |
| 名称 | 备注 |
| 设备注册 | Android端主动调用 |
| 设备编辑 |  |
| 设备查看 |  |
| 设备删除 |  |
| 设备监控 |  |
| 心跳保持 |  |

### 匹配记录管理

|  |  |
| --- | --- |
| 名称 | 备注 |
| 匹配记录上传 | Android端主动调用 |
| 匹配记录查询与筛选 |  |
| 总体考勤报表生成 |  |
| 个人详细出勤报表 |  |

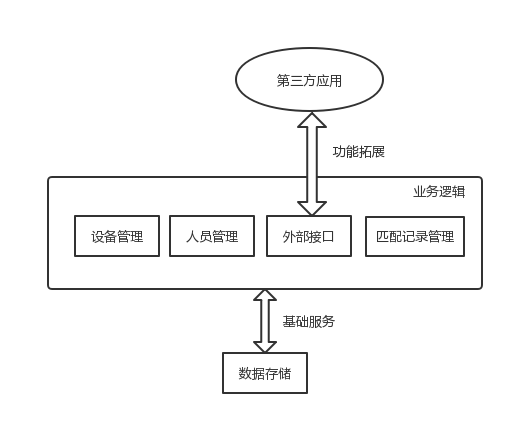
### 第三方接口

|  |  |
| --- | --- |
| 名称 | 备注 |
| 人员数据接口 | 可读可写 |
| 设备数据接口 | 只读 |
| 增量更新接口 | Android端主动调用 |

# 软件实现

## 整体思路

服务端软件采用模块化和层次化的设计原则，将系统分为若干个不同的依赖层次，其中每个层次包含相互独立的软件模块，如下图所示。



其中，数据存储为整个平台的基础服务，所有的业务逻辑均通过数据存储展开；第三方应用（如Android设备）无法直接使用数据存储，但是可以通过外部接口间接调用，这样未来可以通过适当的权限控制来提升平台的安全性。

## 开发思路

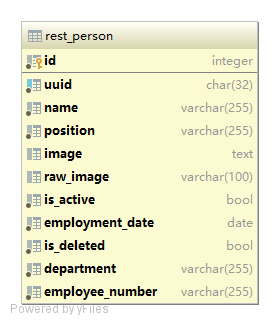
服务端软件采用Python的重量级Web框架Django开发，而前端页面采用HTML，CSS，和JavaScript配合Django模板系统完成。

## 数据存储

数据存储模块直接使用Django自带的ORM系统，将Python类定义与数据库表格关联起来，更便于数据迁移和代码开发与修改，具体内容可以参考Django官方文档中的models概念：<https://docs.djangoproject.com/en/2.1/topics/db/models/>

### 人员

人员信息的代码在rest/models.py中，其数据库中的表格设计如下：



其中，uuid为识别人员的全局唯一标识符，raw\_image为用户上传的图片文件，image为图片的base64字符串，服务端统一使用image字段进行操作，上传的文件会自动转换成base64字符串；此外，is\_deleted表示软删除标志位，true表示已经软删除，平台将不再显示与之相关的任何信息。

### 匹配记录

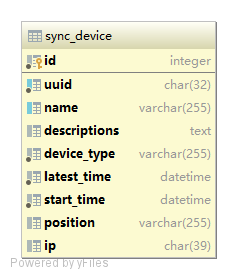
匹配记录的代码在rest/models.py中，其在数据库中的表格设计如下：



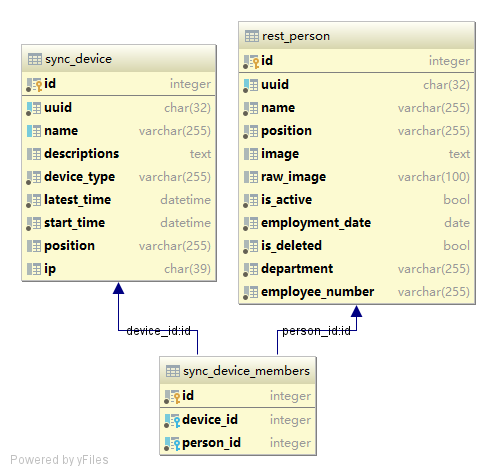
其中，每一条匹配记录与一个人员进行主键关联，可以通过人员与匹配记录的外键引用关系得到一个人员的所有匹配记录。

### 终端设备

设备信息代码在sync/models.py中，其数据库中的表格设计如下：



其中，uuid为识别人员的全局唯一标识符；此外，设备与人员通过一个额外的表格进行关联，形成多对多的关系：



因此，每一个设备可以关联多个人员，而每个人员可以绑定至多个设备终端。

## 人员管理

人员管理模块的代码在enhanced\_ui/views中，url在enhanced\_ui/urls.py中，其功能直接依赖于数据存储模块。在数据存储模块的Django Query API基础上开发后台逻辑，并使用HTML+CSS+Javascript开发相应的前端网页界面。

### 添加人员

添加人员功能的模板为enhanced\_ui/templates/enhanced\_ui/face\_create.html，对应的模块路径为enhanced\_ui.views.face.FaceCreateView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/#createview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/" \l "createview)

### 编辑人员

编辑人员功能的模板为enhanced\_ui/templates/enhanced\_ui/face\_update.html，对应的模块路径为enhanced\_ui.views.face.FaceUpdateView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/#updateview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/" \l "updateview)

此外，编辑人员页面中包含一个额外的多选框，用来绑定人员与设备，多选框的模板与代码均在上述的文件中，其调用的接口为faces/set\_devices/，可参考enhanced\_ui.views.face.set\_devices。

### 删除人员

删除人员功能的模板为enhanced\_ui/templates/enhanced\_ui/face\_delete.html，对应的模块路径为enhanced\_ui.views.face.FaceDeleteView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/#deleteview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/" \l "deleteview)

### 查看人员

查看人员功能的模板为enhanced\_ui/templates/enhanced\_ui/face\_detail.html，对应的模块路径为enhanced\_ui.views.face.FaceDetailView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-display/#detailview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-display/" \l "detailview)

### 人员列表

人员列表功能的模板为enhanced\_ui/templates/enhanced\_ui/face\_list.html，对应的模块路径为enhanced\_ui.views.face.FaceListView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-display/#listview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-display/" \l "listview)

此外，因为人员列表可能包含较多信息，需要一个合适的分页策略，所以这一部分的页面使用ajax调用了系统的对外数据接口，而没有使用Django最简单的模板渲染方式，接口信息见后面章节。

### 批量上传人员信息

人员列表功能的模板为enhanced\_ui/templates/enhanced\_ui/face\_batch\_create.html，对应的模块路径为enhanced\_ui.views.face.BatchUpload，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/#formview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/" \l "formview)

其中，限定了上传的文件为zip格式，且内部文件结构固定，模板文件在examples/demo.zip可供参考。

## 设备管理

设备管理模块的代码在sync/views.py中，url在sync/urls.py中，其功能直接依赖于数据存储模块。在数据存储模块的Django Query API基础上开发后台逻辑，并使用HTML+CSS+Javascript开发相应的前端网页界面。

### 设备注册

设备注册接口为安卓端主动调用的接口/sync/devices/create\_or\_update\_device/，具体接口输入输出文件可以在examples文件夹中找到。

### 编辑设备

编辑设备功能的模板为sync/templates/sync/device\_update.html，对应的模块路径为sync.views.DeviceUpdateView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/#updateview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/" \l "updateview)

此外，编辑设备页面中只包含一个的多选框，用来绑定人员与设备，多选框的模板与代码均在上述的文件中，其调用的接口为sync /devices/set\_people/，可参考sync.views.set\_people。

### 删除设备

删除设备功能的模板为sync/templates/sync/device\_delete.html，对应的模块路径为sync.views.DeviceDeleteView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/#deleteview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-editing/" \l "deleteview)

### 查看设备

查看设备功能的模板为sync/templates/sync/device\_detail.html，对应的模块路径为sync.views.DeviceDetailView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-display/#detailview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/generic-display/" \l "detailview)

### 设备列表

设备列表功能得模板为sync/templates/sync/device\_list.html，对应的模块路径为sync.views.DeviceListView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/#templateview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/" \l "templateview)

此外，为了实现高效的设备监控动态效果，此页面使用了vue.js进行前端开发，并配合心跳保持接口和Websocket进行完整的功能实现。

### 设备监控（心跳保持）

心跳保持接口为安卓端主动周期性调用的接口/sync/ heart\_beats /，具体接口输入输出文件可以在examples文件夹中找到。

## 匹配记录管理

### 匹配记录上传

匹配记录上传接口为安卓端主动调用的接口/sync/ record/，具体接口输入输出文件可以在examples文件夹中找到。

### 匹配记录查询

匹配记录列表功能的模板为enhanced\_ui/templates/enhanced\_ui/record\_list.html，对应的模块路径为enhanced\_ui.views.record.RecordListView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/#templateview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/" \l "templateview)

因为匹配记录列表可能包含较多信息，需要一个合适的分页策略，所以这一部分的页面使用ajax调用了系统的对外数据接口，而没有使用Django最简单的模板渲染方式，接口信息见后面章节。

### 考勤报表生成

考勤报表功能的模板为enhanced\_ui/templates/enhanced\_ui/report.html，对应的模块路径为enhanced\_ui.views.report.ReportView，如有具体疑问请参考Django官方文档[https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/#templateview](https://docs.djangoproject.com/en/2.1/ref/class-based-views/base/" \l "templateview)

## 外部接口

外部数据接口采用REST API的形式进行开发，使用了djangorestframework这个第三方库开发，具体细节可以参考官方文档<https://www.django-rest-framework.org/>

其中，外部接口的url定义在pengfeng/urls.py中。

### 人员数据接口

人员数据接口的实现主要在rest/serializers.py，rest/ordering.py和rest/views.py中。rest/serializers.py依附于rest/models.py，主要定义了应该对外暴露人员数据的哪些字段。

rest/ordering.py主要定义了如何对人员数据进行排序，这一块主要是为了符合一个前端控件bootstrap-table，其说明在<https://github.com/wenzhixin/bootstrap-table>，一般不推荐对着进行修改，如果想加入新的功能直接定义新的接口即可。

rest/views.py定义了人员数据接口的详细实现，rest.views.PersonViewSet为目前内部平台使用的接口，而rest.views.CustomPersonViewSet与rest.views.PersonViewSet几乎相同，只是使用了UUID作为识别实体的唯一标识。

### 匹配记录数据接口

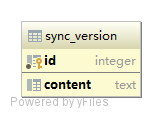
与人员数据接口相同，匹配记录接口也实现于rest/serializers.py，rest/ordering.py和rest/views.py中，其细节略有不同。

### 设备数据接口

设备数据接口的实现主要在sync/serializers.py和rest/views.py中，使用设备UUID作为全局标识符，且所有字段都是只读的。

### 数据同步接口

数据同步接口实现在sync/sync\_tool.py和sync/views.py中，主要目的是实现增量式数据更新。数据库中保存一个Version表格，每次关于人员的操作都会记录在表格中，服务器根据设备提供的版本号将更新的数据以增量式的形式返回给设备，版本数据库表格设计如下：



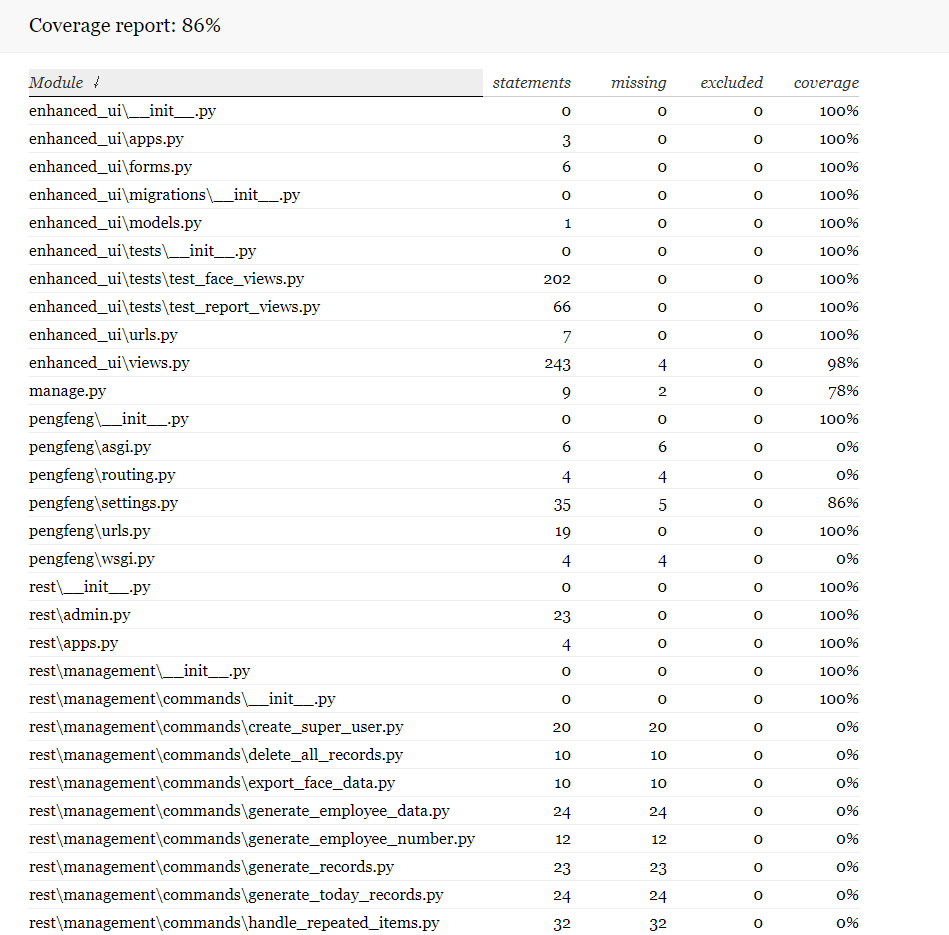
其中，具体的设计在sync.views.get\_version\_difference中。

# 软件测试

平台软件的测试分为单元测试和功能测试，单元测试是针对部分代码模块进行的小范围局部测试，功能测试则是针对平台的具体功能，直接在网页端进行人工测试。此外，平台只有通过了单元测试和功能测试，才能继续进行功能开发。

## 单元测试

测试脚本文件夹为enhanced\_ui/tests，rest/tests和sync/tests，里面有详细的测试脚本文件和测试注释，开发的时候可以参考。测试全部通过，且代码覆盖率可以达到86%。



## 功能测试

|  |  |  |
| --- | --- | --- |
| 测试名称 | 测试步骤 | 合格判据 |
| 登录系统 | 1. 浏览器输入站点网址 2. 在登录框输入账户信息 3. 点击登陆按钮 | 成功跳转至网站页面 |
| 员工列表 | 1. 登陆系统 2. 进入人员管理页面 | 正常显示人员列表 |
| 添加员工 | 1. 进入员工列表页面 2. 点击添加人员按钮 3. 输入人员信息 4. 点击确认按钮 | 新增人员显示在人员列表中 |
| 编辑员工 | 1. 进入员工列表页面 2. 点击编辑人员按钮 3. 输入人员信息 4. 点击确认按钮 | 编辑后的人员被正确保存 |
| 查看员工 | 1. 进入员工列表页面 2. 点击查看人员按钮 | 显示人员的信息和出勤记录等内容 |
| 删除员工 | 1. 进入员工列表页面 2. 点击删除按钮 3. 确认删除 | 删除人员成功 |
| 批量删除员工 | 1. 进入员工列表页面 2. 左侧多选框进行选择 3. 点击批量删除按钮 | 批量删除成功 |
| 批量上传员工 | 1. 进入员工列表页面 2. 点击批量上传按钮 3. 上传压缩文件 4. 点击确认按钮 | 人员上传成功 |
| 匹配记录查询 | 1. 进入匹配记录列表 2. 点击列标题进行排序 3. 搜索框输入信息进行搜索 | 操作无异常 |
| 添加终端 | 1. 使用Android设备注册终端 2. 进入设备列表 | 设备被添加到设备列表 |
| 编辑终端 | 1. 进入设备列表 2. 点击编辑按钮 3. 绑定人员信息 4. 点击更新按钮 | 人员绑定成功 |
| 删除终端 | 1. 进入设备列表 2. Android断开连接 3. 出现删除按钮 4. 点击删除按钮 5. 确认删除 | 设备删除成功 |
| 查看终端 | 1. 进入设备列表 2. 点击查看按钮 | 显示设备信息 |
| 考勤报表 | 1. 进入考勤报表页面 2. 输入查询日期 3. 点击查询按钮 | 显示出勤信息，并且出勤人员可以查看当天详细记录 |
| 个人报表 | 1. 进入人员列表 2. 点击查看按钮 3. 出勤记录框上面选择日期范围 4. 点击查询按钮 | 显示个人报表，并包含详细的出勤信息 |

# 附录

## 测试报告表

|  |  |  |
| --- | --- | --- |
| 测试类别 | 测试名称 | 测试结果 |
| 单元测试 | 单元测试 | 91个单元测试，91个通过，0个未通过 |
| 功能测试 | 登录系统 | 通过 |
| 员工列表 | 通过 |
| 添加员工 | 通过 |
| 编辑员工 | 通过 |
| 查看员工 | 通过 |
| 删除员工 | 通过 |
| 批量删除员工 | 通过 |
| 批量上传员工 | 通过 |
| 匹配记录查询 | 通过 |
| 添加终端 | 通过 |
| 编辑终端 | 通过 |
| 删除终端 | 通过 |
| 查看终端 | 通过 |
| 考勤报表 | 通过 |
| 个人报表 | 通过 |

## 第三方库依赖

第三方依赖可以参考requirements.txt文件：

|  |
| --- |
| django  djangorestframework  djangorestframework-bulk  djangorestframework-filters  drf-yasg  django-picklefield  channels  channels\_redis  psycopg2-binary  django-filter==2.0.0  daphne  numpy  Pillow  requests  openpyxl  arrow  pyzip  coverage  business\_calendar |

## 注意事项

1. 服务器程序完全参照Django官方文档进行开发，官方文档有详细的说明。
2. 在使用websocket时，依赖了channels库。也因为使用了websocket，因此无法采用传统的方法进行部署，详情见<https://channels.readthedocs.io/en/stable/>中的部署模块。
3. 服务器代码采用python3.5进行开发，所以python2可能会出现兼容问题。
4. 目前服务器程序兼容各类关系型数据库，可以在settings.py里面设置。

## 修订记录

|  |  |
| --- | --- |
| 修订人 | 修订日期 |
| 盛武斌 | 2019.1.25 |