密级：公开

高校固定资产管理系统的设计与实现

Design and Implementation of University Fixed Assets Management System

学　　院：软件学院

学　　号：161203726

专业班级：软件工程 1607 班

学生姓名：刘靖诗

指导教师：邵中

2020年06月

摘 要

高校的固定资产是发展高等教育的物质基础，我国经济正在不断发展，各所高校也在不断扩招，高校的办学规模不断扩大，办学规模的扩大必然会导致高校的固定资产的种类和数量不断增多，固定资产的管理也会变得越来越复杂，传统的固定资产模式已经无法方便快捷的完成对固定资产的管理。在信息化时代背景下，高校固定资产的管理已从传统模式转变为信息化管理模式。

本文以 PC 端和移动端相结合的方式进行固定资产的管理，利用 GPS 定位以及二维码管理，为每个固定资产生成单独的二维码及其位置信息。将二维码的加密、资产图片的自动审核作为设计重点。采用 B/S 架构，以浏览器、微信小程序、手机 APP 作为客户端，以 Apache、Tornado 作为 Web 服务器，使用 MVC 设计模式以及 Vue 单页应用进行开发，并申请 SSL 证书，使用 HTTPS 协议进行 Request 请求，WSS 协议进行 WebSocket 连接，保证数据传输的安全性，采用 MySQL 数据库进行数据存储，前端使用 Bootstrap+Vue.js+layui 作为主体框架，uni-app 进行移动端开发。

系统实现了资产信息管理、资产盘点功能、审批管理功能以及部门管理、职位管理、人员管理、用户管理等基本功能。可以在移动端首页发布帖子和回复帖子，进行用户之间交流。

关键词：固定资产；管理系统；二维码；图片相似度；Socket；Vue；Tornado；

Abstract

The fixed assets of colleges and universities are the material basis for the development of higher education, Our economy is developing continuously, Universities are also expanding their enrollments, The scale of colleges and universities is constantly expanding, The expansion of school scale will inevitably lead to the variety and quantity of fixed assets of colleges and universities, The management of fixed assets will also become more complex, The traditional model of fixed assets has been unable to complete the management of fixed assets conveniently and quickly. Under the background of information age, The management of fixed assets in colleges and universities has changed from the traditional mode to the information management mode.

This paper combines PC terminal and mobile terminal to manage fixed assets, GPS positioning and QR code management, Generate a separate QR code and its location information for each fixed asset. Take the encryption of QR code and automatic audit of asset pictures as the key points of design. B/S architecture is adopted, With browser, WeChat small program, mobile phone APP as the client, Apache, Tornado as the Web server, Use MVC design pattern and Vue single page application for development, And apply for an SSL certificate, HTTPS protocol is used for Request and WSS protocol for WebSocket connection, Ensure the security of data transmission, MySQL database is used for data storage, The front end USES Bootstrap+ Vue.js+ layui as the main frame, Uni-app carries out mobile terminal development.

The system has realized the functions of asset information management, asset inventory, examination and approval management, department management, position management, personnel management, user management and other basic functions. You can post and reply posts on the home page of the mobile terminal for communication between users.

Keywords: Fixed assets; Management system; QR code; Image similarity; Socket; Vue; Tornado;

目 录

第1章 引 言

1.1 研究背景与意义

近几年随着我国经济的突飞猛进，高等教育事业得到了蓬勃发展。高校固定资产规模随之扩大，资产构成日趋复杂，导致管理难度也越来越大。加强高校固定资产管理不仅是为了确保国有资产的安全完整，也是高校自身发展的内在需要。因此，进一步提高高校固定资产管理水平和能力显得至关重要。

当前信息化技术已经被逐步引入到高校的固定资产管理工作中，在台账建档、资产调拨、折损计算等业务工作中取得了明显的效益，大幅度地提高了工作效率和质量。但是多数现有系统所提供的业务支持不够全面，偏重于资产登记、调拨、使用和折损处置等环节，而在清查、盘点、实物核对这一类需要大量人员参与的业务工作中却缺乏有效的工作模式和功能支撑，导致上述工作依然需要通过少数人员的繁重劳动方可完成，未能从根本上解决高校固定资产管理所面临的痛点问题。尽管目前以 RFID 智能标签结合移动射频采集设备的解决方案可以在一定程度上缓解资产清查盘点工作的困境，但相对高昂的应用成本则成为其应用推广所面临的主要障碍。

本课题以图形二维码和智能手机取代 RFID 标签及专用射频采集设备，给出一种低成本且便于广泛参与的高校固定资产管理解决方案及配套软件系统。课题的研究及实践进一步提升固定资产管理水平和质量，提高资产清查盘点的准确性，降低应用成本和工作人员的劳动强度，具有明确的实用价值和现实意义。

1.2 国内外研究现状

分别从同类产品的应用现状和相关技术研究进展两个方面对与课题相关的现存工作进行调研分析，分类归纳如下。

1.2.1 应用现状

固定资产管理信息系统是以固定资产为管理对象，以资产台帐为基础，以统筹使用，防止流失，加强管理监督并提升使用效率为目的的管理系信息系统。当前市场上存在大量的同类产品，本文重点对其中与移动端应用功能支持有关的 3 款产品进行分析。

“易点固定资产管理系统”由易点易动公司开发，主要功能分为资产管理、耗材管理、财务管理以及审批管理四大模块。该系统支持移动端 APP 接入，可以使用手机扫码完成资产盘点，但无法对固定资产地理位置进行 GPS 定位；后台可提供对 RFID 标签的功能支持，但需要单独购买相应设备。

“联想百应资产管理系统”是由联想公司开发的一款同类应用系统。功能分为资产管理和耗材管理两大模块，无法使用手机扫码进行资产盘点，但可以申请工作人员代盘点，需要支付代盘点费用，也可以直接使用联想商城购买耗材。

“简普固定资产管理软件”是一款由 ASP/C#开发的固定资产管理软件，其功能分为个人办公、资产管理和耗材管理三大模块，无法使用手机扫码进行资产盘点，界面简单，但支持支持盘点任务的短信通知。

1.2.2 研究现状

大部分固定资产管理相关系统采用 PC 端 B/S 架构模式以及 RFID 自动识别技术，在 2013 年以前，多数为 Java 开发的 C/S 架构模式，目前，已有小部分系统采用了移动端的形式。

在 2019 年张慕然[1]基于 ASP.NET 平台和 SQLServer 数据库实现浏览器与服务器(Browser/Server，B/S)架构的数据库系统，采用 Knockout 作为 JavaScript 库，EasyUI 作为 UI 库，采用面向对象的设计思想建立资产管理系统的数据模型、功能模型和分层模型和规范的业务管理流程。切实做到资产管理落实到人、落实到点、物物可查、事事能循。

姜宇飞[2]基于 Java 平台采用了 Spring MVC 开发模式，以 Apache Tomcat 为服务器，采用 MySQL 关联数据库，并通过 MyBatis 技术实现数据库的链接，实现了设备资产从采购到使用到维护到报废整个生命周期的一体化管理，使得设备资产的各个管理环节完美衔接，形成了更加科学合理的管理流程。

张华[3]在 2018 年利用非对称加密算法对 QR 二维码加密，生成了基于 DES 加密的 QR 二维码，并将私钥放置于服务端，公钥放置于客户端，在用户扫描二维码时进行解密，有效杜绝二维码的篡改、伪造等系列问题，提高里面二维码的安全性。

高硕[4]在 2019 年在其开发的基于内容的图像检索系统中，根据提取的主要特征，采用欧氏距离、直方图相交等方法对图像相似性进行度量，可方便地替换为马氏距离、棋盘格距离、切比雪夫距离、直方图相交法等多种度量方法，计算被检索图像与当前图像的相似度。

1.3 研究内容及主要工作

课题的主要目标是在 PC 端与移动端（手机 APP、微信小程序）实现一套完整的高校固定资产管理系统，可以投入实际运营。对于移动端而言，实现资产信息查看、资产入库、资产盘点、资产的借用与归还，在资产盘点的过程中，对于用户拍摄的照片进行裁剪，并与资产入库的图片进行相似度对比，防止用户胡乱拍摄。并提供了小型论坛交流场景，方便询问一些关于资产使用方面的注意事项，管理员也可以利用该场景发布一些通知。

系统 PC 端主要采用 ThinkPHP5 作为开发框架，使用 Bootstrap4 配合 Vue.js 以及 layui 组件完成 PC 端设计，数据处理采用 PHP 脚本语言，外加阿里云提供的短信接口 API 以及地图接口 API，完成了高校固定资产管理系统的 PC 端。

1.4 实验环境与条件

系统移动端使用微信开发平台以及 HBuilder X 配合生产原生 APP 以及微信小程序的发布，开发语言采用 JavaScript 和 Python；系统 PC 端使用 PHP 语言开发。服务端实验环境服务器采用 CentOS7.4，客户端环境采用 Windows10 Google Chrome、iOS10、Android9。

第 2 章　相关技术

系统 PC 端基于 ThinkPHP5 框架，采用 MVC 设计模式，使用 Apache 作为 Web 服务器软件，使用 Bootstrap+Vue.js+layui+jQuery+ECharts 进行前端页面的开发，系统移动端基于 Tornado 作为服务端开发框架，uni-app 作为客户端开发框架，数据库采用 MySQL，PC 端使用 ThinkPHP 自带的 Db 类进行数据库操作，移动端使用本人基于 pymysql 封装的 Ljsmysql 类进行数据库操作，下面将对系统的主要技术进行介绍。

2.1 主要相关算法

2.1.1 二维码加密算法

本文采用的二维码加密算法是一种基于 AES 和 RSA 的混合二维码加密算法，通过加密手段将二维码的内容加密，这样使用正常的扫码工具扫码只会显示一堆乱码。

AES 算法是一种新型加密方法，具有更加可靠的加密过程和更加适合的密钥长度[5]，而 RSA 算法是一种非对称加密算法。本文则是使用了一种 AES 和 RSA 混合的加密算法。

AES 算法需要一个秘钥进行加密以及解密，而为了避免秘钥泄露所以 AES 算法的秘钥采用随机生成的方式，即每个二维码的秘钥都不一样，使用该秘钥对二维码内容进行加密。而 RSA 算法则需要公钥以及私钥两种秘钥，一种加密必须使用另一种解密，接下来则用 RSA 算法的私钥对 AES 算法的秘钥进行加密，最后将前后得到的两种密文以特定方式组合在一起生产加密后的二维码。

2.1.2 资产图片核对算法

本文采用颜色直方图计算相似度的算法，通过计算盘点时的资产图片和资产入库时的图片的相似度，初步判断该资产是否通过盘点。

颜色直方图是在许多图像检索系统中已获得广泛采用的颜色特征，算法简单方便[6]。

单一通道的直方图相似值是通过 OpenCV 获取颜色直方图数据，再使用巴氏系数算法计算出相似程度。

巴氏系数算法的公式如下：

$$\rho(p,\ p') = \sum_{i=1}^{N} \sqrt{p(i)p'(i)}$$

式中 p、p`分别表示原始与待比较的图像直方图数据。

本文通过计算 RGB 每个通道的相似值，然后取其平均值作为两张图片的相似度，通过大量实验数据得出资产图片核对的相似度阈值。

2.2 主要前端技术

2.2.1 WebSocket 通讯技术

Websocket 是 HTML5 新标准中的通信机制，能够实现稳定全双工实时通信,具有简洁高效的特点[7]。提供了 WebSocket 接口的相关定义，允许服务端主动向客户端推送数据，不必使用 Ajax 轮询来获取服务端主动发送的数据。在 WebSocket API 中，浏览器和服务器只需要要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。本文的移动端和服务端就是使用了 WebSocket 通讯技术。

2.2.2 Vue.js 框架

Vue.js 是一个非常典型的 MVVM 框架，模型只是一个 JavaScript 对象，如果修改了模型视图会自动更新，这种设计让状态的管理变得简单而且直观[8]。使用 Object.defineProperty，通过设定对象属性的 getter 和 setter 方法来监听相应的数据变化，通过 getter 进行依赖收集，每个 setter 方法都是一个观察者，在数据发生变化时触发相应的监听回调，通知订阅者进行视图更新。这样就实现了数据的双向绑定。本文主要在表单上使用 Vue.js 实现数据的双向绑定。

2.2.3 uni-app

uni-app 是基于 Vue.js 框架所开发的跨平台应用前端[9]。开发者编写一套代码，可发布到 iOS、Android、H5、以及各种小程序、快应用等多个平台。uni-app 是目前跨端成熟度和案例数量最多的框架。每一个页面对应一个.vue 文件，里面分为 HTML、CSS 和 JavaScript 三部分，HTML 和 CSS 控制着该页面的内容与样式，JavaScript 部分包含诸多生命周期函数，监听 APP 从打开到关闭的每一步动作。本文的移动端则使用 uni-app 进行开发。

2.2.4 ljspopup

ljspopup 是本人使用 JavaScript 语言基于 jQuery 编写的弹出层组件，界面风格模仿了 Mac OS 的窗口，与 Mac OS 的窗口有 90%的相似度。调用时只需要一句 JavaScript 代码即可实现弹出层，具有关闭、最大化、最小化、移动等功能，可以无限层弹出，最小化后可以在右下角还原。

2.3 后端主要技术

2.3.1 Tornado 框架

Tornado 全称 Tornado Web Server，是一个用 Python 语言写成的 Web 服务器兼 Web 应用框架，由 FriendFeed 公司在自己的网站 FriendFeed 中使用，被 Facebook 收购以后框架以开源软件形式开放给大众[10]。

Tornado 采用异步非阻塞 IO 的处理方式，处理速度非常快，具有很强的抗负载能力。本文主要使用 Tornado 中的 WebSocket 创建 Socket 服务端，配合使用 Apache 做反向代理服务器部署。

2.3.2 OpenCV

OpenCV 是一个开源的可以跨平台的计算机视觉库，可以运行在多种操作系统中，并且为多种计算机语言提供了接口，可以实现计算机视觉、图像处理方面的很多算法[11]。本文的图片核对算法就是利用了 OpenCV 的 Python 接口来进行图片处理以及颜色直方图的数据处理

2.3.3 ThinkPHP5 框架

ThinkPHP5 是一个免费开源、快捷简单的轻量级 PHP 开发框架，基于 MVC 架构模式，高度封装了数据库的 CURD 操作和常用 API[12]。本文 PC 端则使用了 ThinkPHP5 进行开发，Model 层和 Controller 层用来编写后端代码，View 编写前端代码，由于使用了 ThinkPHP5 的 Route，所以在前端并未使用 Vue.js 的 vue-router。

2.3.4 Ljsmysql 数据库操作类

Ljsmysql 是本人使用 Python 语言基于 pymysql 编写的 MySQL 数据库操作类，使用风格模仿了 ThinkPHP5 中的 Db 数据库操作类，无需编写 SQL 语句，只需要使用 insert、delete、update、select、order、page 等函数即可实现最基本的增删改

查以及排序和分页查询等操作。

第 3 章　系统需求分析与总体设计

在系统开发之前，必须对系统的需求以及总体设计进行必要的分析与规划，本章就会对系统需求分析、系统总体设计以及数据库设计进行详细的介绍。

3.1 系统需求分析

3.1.1 业务流程分析

高校固定资产管理基本的业务流程图如图 3-1 所示。

管理员用户需要先进行资产登记，也就是资产入库，登记资产的相关信息，系统会自动生成二维码，工作人员下载后打印贴到相应的固定资产上面，即可完成资产入库。

当所有的资产全部登记到系统之后，学生和老师会进行资产的借用/归还操作，使用手机扫描资产二维码或者使用电脑登陆 PC 端后台，选择借用或者归还，然后等待管理员审核通过。

对于管理员用户来说，可以直接将资产借用给某人，也可以选择审核通过某人的借用或归还的申请，资产的状态就会从审核中变成在用或闲置。等到了需要进行资产盘点的时候，管理员需要在后台创建盘点单，指定盘点人员。接到盘点任务的用户需要进行资产盘点，对盘点任务的资产进行扫码拍照，等待管理员审核，管理员需要对系统无法判断的资产进行核对，如果资产不合格资产需要用户重新进行盘点，等到所有资产全部盘点合格之后盘点任务完成。

对于系统管理用户来说，可以使用普通管理员用户和普通用户的全部功能。并可以进行系统管理，包括高校的各个部门，人员的职位与信息的管理，以及用户添加、修改、删除以及权限的管理。同时还可以查看用户的详细操作日志。

对于不同的用户，系统在任何操作之前都会对其权限进行盘点，权限不足则会得到提示。游客的权限最低，只可以扫码查看资产的基本信息。

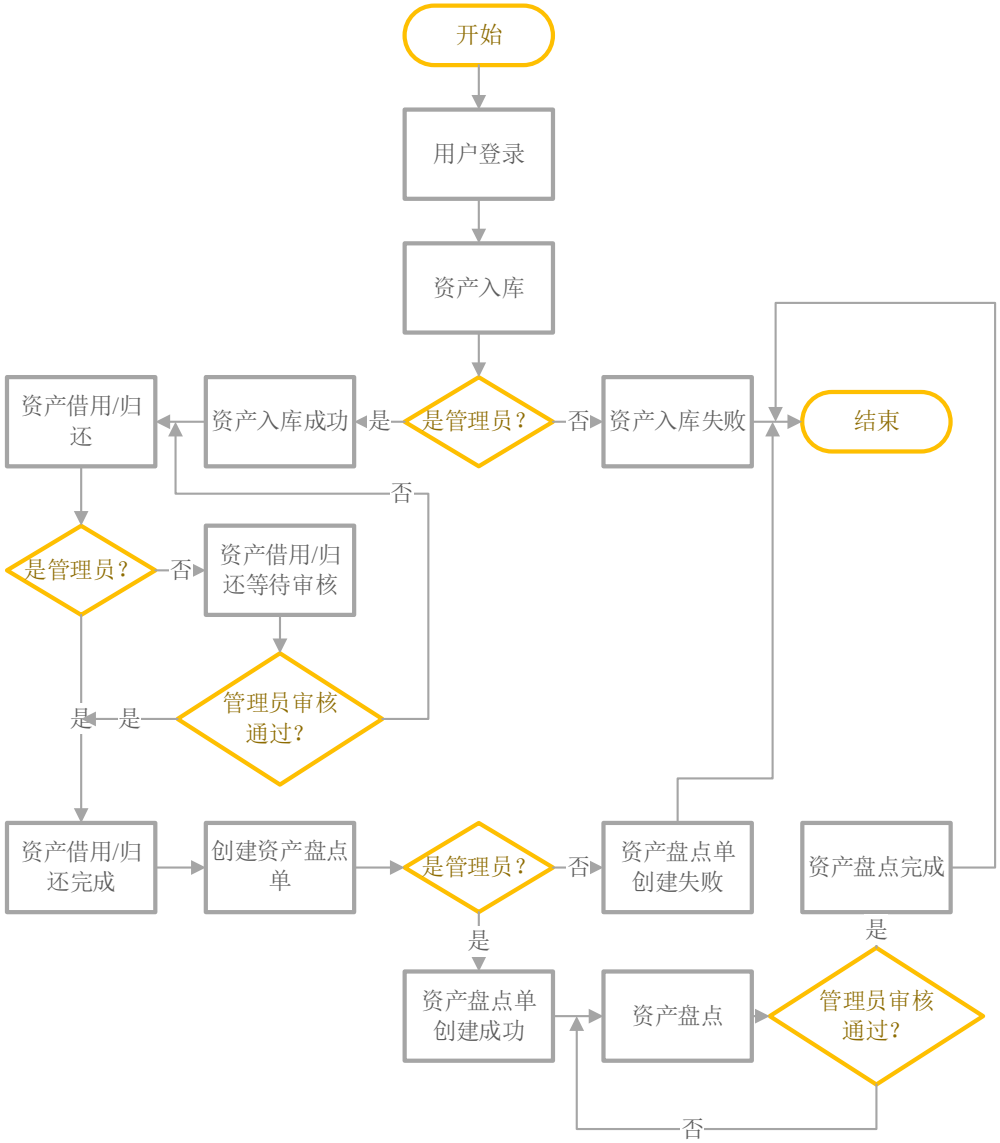

图 3 - 1 业务流程图

3.1.2 系统需求定义

根据系统的业务流程分析，本系统主要由四类角色构成：游客，高校普通用户，高校普通管理员用户，高校系统管理员用户。系统的用例图如图 3-2 所示。

(1) 游客

一般的游客指的是未登录用户，在本系统中游客除了未登录用户还可以指未绑定在职人员的用户，游客只可以使用 APP 扫码查看资产基本信息，不可以登录后台。

(2) 高校普通用户

高校普通用户指的是普通学生和教师，就是在高校中并没有资产管理的人员，该类人员为普通用户，普通用户处理可以查看资产基本信息外，还可以进行资产的借用与归还，在接到资产盘点的任务后，进行资产盘点，但所进行的操作都需要管理员进行审核才可以通过。

(3) 高校普通管理员用户

高校普通管理员用户指的是在高校中有资产进行管理的人员，或被赋予资产管理权限的人员，普通管理员用户在普通用户的基础上，还可以资产入库，即资产的增删改查，可以进行盘点单的创建。

(4) 高校系统管理员用户

高校系统管理员用户指的是整个高校固定资产管理系统的管理者，在普通管理员用户的基础上，还可以记性部门管理、职位管理、人员管理、用户管理等操作。

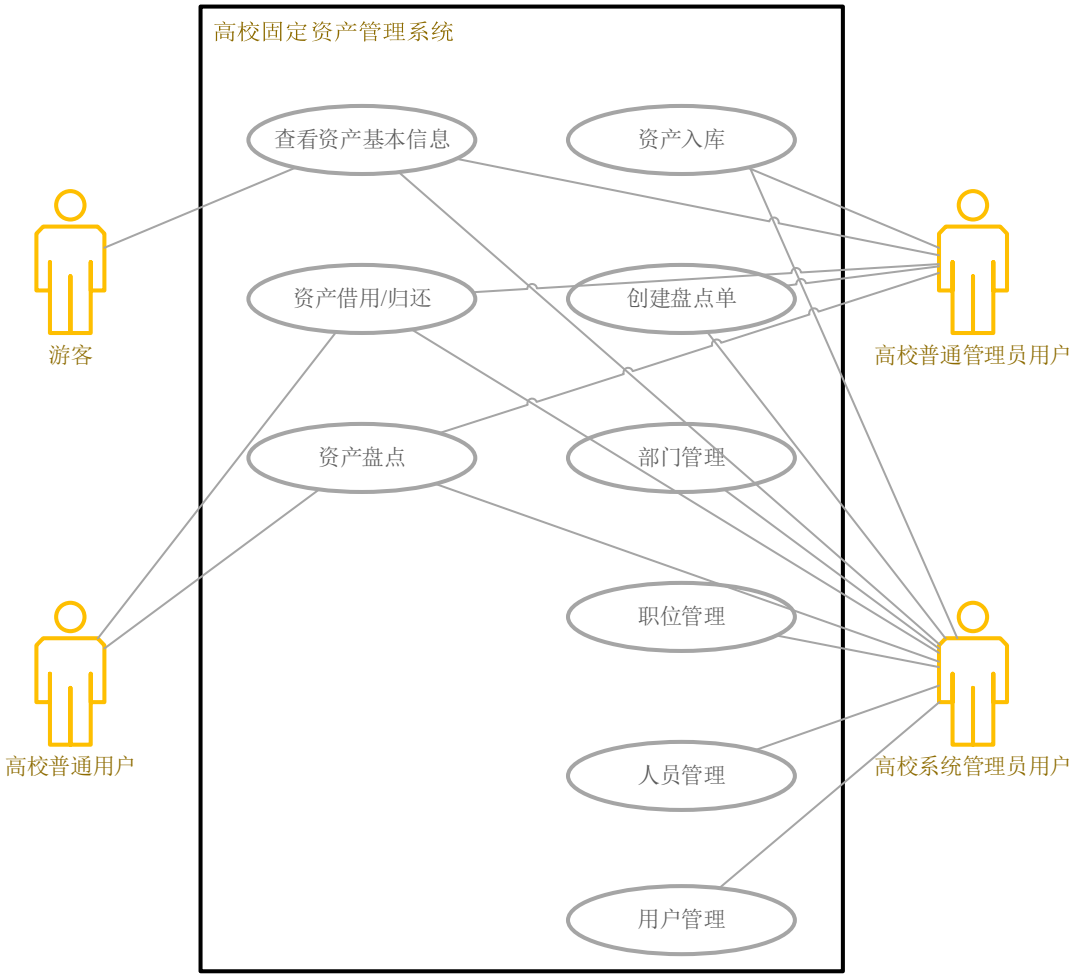

图 3 - 2 系统用例图

3.2 系统总体设计

系统总体设计是对整个系统的设计与实现进行合理的安排，下面将从体系结构设计和功能结构设计进行说明。

3.2.1 体系结构设计

系统基于 B/S 结构设计，分为 PC 端和移动端，PC 端采用 MVC 设计方式，移动端则是 Vue 单页应用。PC 端和移动端分别有不同服务端和客户端，部署关系如图 3-3 所示。其中 Apache 服务器使用 PHP 语言编写，通过 MySQL 数据库进行数据存储，Tornado 服务器采用 Python 语言开发，小程序客户端和手机 APP 客户端与 Tornado 服务器通过 WebSocket 连接，PC 客户端通过 HTTPS 协议请求 Apache 服务器。

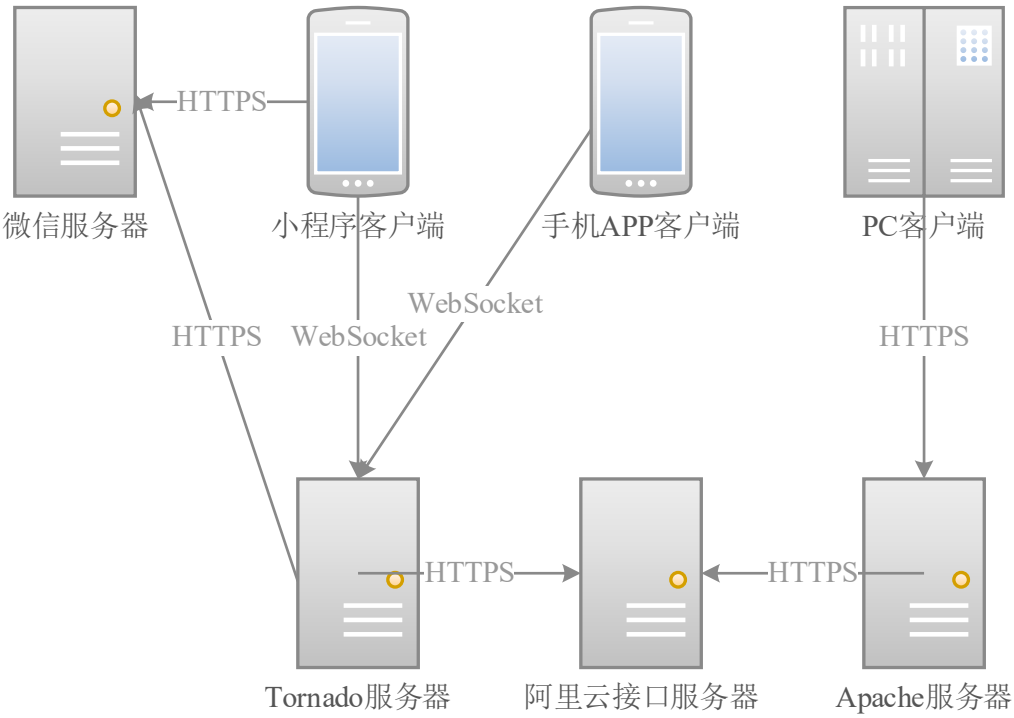

图 3 - 3 部署关系图

3.2.2 功能结构设计

高校固定资产管理系统在功能结构上主要分为 4 个功能模块，分别是：资产管理、资产盘点、审批管理、系统管理。下面对其功能进行一一介绍。

(1) 资产管理

资产管理是对固定资产进行管理，由系统管理员登录 PC 端后台根据已有固定资产的数量批量生产二维码，再将二维码打印贴在固定资产上，管理员再通过移动端操作，先扫码再填写资产信息最后拍照保存，如此完成资产信息的录入。管理员也可以在 PC 端后台直接对固定资产进行增删改查操作，也可在移动端扫码进行固定资产信息的修改。普通用户可以通过移动端扫码的方式查看资产信息，以及借用或者归还资产。普通用户和管理员可以申请资产报废或者维修，管理员在后台可以进行资产调拨。

(2) 资产盘点

管理员登录 PC 端后台创建盘点单，在盘点单创建成功后，指定的盘点用户和管理员可以查看相应的盘点单，需要在截止日期之前进行资产盘点，用户将盘点单对应的资产进行扫码拍照以及 GPS 定位，并将数据上传，由系统初步处理后交于管理员审核，待管理员审核通过后本次盘点结束。

(3) 审批管理

在普通用户进行申请操作时（借用归还固定资产、资产盘点），管理员需要记性审批，普通用户在固定资产的借用与归还时需要管理员进行审核，管理员给予通过方可成功，反正操作失败。在进行资产的盘点之后，系统的初步检查结果会出现盘点审批中，管理员对系统无法判断的资产进行审批，管理员给予通过后该资产盘点任务结束。

(4) 系统管理

系统管理只有系统管理员可以进行操作，包括部门管理、职位管理、人员管理以及用户管理，可以对部门、职位、人员以及用户进行最基本的增删改查，对用户进行权限管理，也可以查看其它用户的操作日志，可以详细的查看其它用户在什么时候对某数据库某表某记录进行了什么操作。

3.3 数据库设计

本系统中使用 MySQL 数据库为高校固定资产管理系统提供数据存储服务，主要包括资产管理、借用归还、资产盘点、部门管理、职位管理、人员管理、用户管理、日志管理、帖子管理等，下面对一些主要的表进行介绍。

(1) 资产信息表（fams_asset）

资产信息表用于存放资产的信息，具体结构如表 3-1 所示。

表 3-1 资产信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	as_id	int	10	主键	主键
2	as_no	varchar	64	资产编号	唯一
3	as_name	varchar	64	资产名称	非空
4	as_price	double	10,2	资产价格	非空
5	cate_id	int	11	类别id	非空
6	sta_no	varchar	255	状态编号	非空
7	as_import_time	datetime	0	资产入库时间	非空
8	as_image	varchar	255	资产照片	非空
9	as_local_id	int	11	资产地点	非空
10	as_qrcode	varchar	255	资产二维码	非空
11	as_exist	int	11	记录是否存在	非空

(2) 借用归还表（fams_borrowlend）

借用归还表用于存放资产借用以及归还的相关信息，具体结构如表 3-2 所示。

表 3-2 借用归还表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	bl_id	int	10	主键	主键
2	as_no	varchar	64	资产编号	非空
3	u_id	int	11	用户id	非空
4	b_time	datetime	0	领用时间	非空
5	l_time	datetime	0	归还时间	非空
6	bl_ok	int	11	该记录已完成	非空
7	bl_exist	int	11	记录是否存在	非空

(3) 资产盘点表（fams_check）

资产盘点表用于存放资产盘点单，不存放资产盘点的内容，用于记录盘点单创建和截止的日期以及盘点状态，具体结构如表 3-3 所示。

表 3-3 资产盘点表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	c_id	int	10	主键	主键
2	c_title	varchar	255	盘点单名称	非空
3	u_id	int	11	指定盘点人	非空
4	c_time	datetime	0	盘点单创建日期	非空
5	c_e_time	datetime	0	盘点限期	非空
6	c_sta_no	varchar	16	盘点单状态	非空
7	c_exist	int	11	记录是否存在	非空

(4) 资产盘点单表（fams_check_content）

资产盘点单表用于存放资产盘点的数据，记录每一个资产的盘点状态，具体结构如表 3-4 所示。

表 3-4 资产盘点单表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	c_c_id	int	10	主键	主键
2	c_id	int	11	所属盘点单id	非空
3	as_no	varchar	64	资产编号	非空
4	c_c_image	varchar	255	盘点图片	非空
5	c_c_sta_no	varchar	16	盘点状态编号	非空
6	c_c_exist	int	11	记录是否存在	非空

(5) 部门信息表（fams_department）

部门信息表用于存放高校各个部门的结构，具体结构如表 3-5 所示。

表 3-5 部门信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	dep_id	int	10	主键	主键
2	dep_no	varchar	32	部门编号	唯一
3	dep_name	varchar	40	部门名称	非空
4	up_dep_id	int	11	上级部门id	非空
5	dep_remark	varchar	255	备注	可空
6	dep_exist	int	11	记录是否存在	非空

(6) 地点信息表（fams_local）

地点信息表用于存放高校各个部门地点的 GPS 数据，具体结构如图 3-6 所示。

表 3-6 地点信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	local_id	int	10	主键	主键
2	local_name	varchar	255	地点名称	非空
3	local_data	varchar	255	GPS相关数据	非空
4	local_exist	int	11	记录是否存在	非空

(7) 日志信息表（fams_log）

日志信息表用于存放用户的操作，便于管理员查看，具体结构如表 3-7 所示。

表 3-7 日志信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	log_id	int	10	主键	主键
2	log_category_no	varchar	32	日志类别编号	非空
3	log_action_no	varchar	32	日志操作编号	非空
4	u_id	int	11	用户id	非空
5	log_datetime	datetime	0	操作时间	非空
6	common_id	int	11	公用id	非空
7	table_name	varchar	64	表名	非空
8	table_main_key	varchar	32	关联的表的主键	非空
9	log_exist	int	11	记录是否存在	非空

(8) 人员信息表（fams_person）

人员信息表用于存放高校的在职人员信息，如学生和教师，具体结构如表 3-8 所示。

表 3-8 人员信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	p_id	int	10	主键	主键
2	p_no	varchar	32	工号	唯一
3	dep_id	int	11	部门主键	非空
4	pos_id	int	11	职位主键	非空
5	p_name	varchar	40	姓名	非空
6	p_sex	char	2	性别	非空
7	p_ic	varchar	20	身份证号	非空
8	p_email	varchar	40	邮箱	非空
9	p_exist	int	11	记录是否存在	非空

(9) 职位信息表（fams_position）

职位信息表示用于存放高校各个职位的名称以及标号，具体结构如表 3-9 所示。

表 3-9 职位信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	pos_id	int	10	主键	主键
2	pos_no	varchar	32	职位编号	唯一
3	pos_name	varchar	40	职位名称	非空
4	pos_remark	varchar	255	备注	可空
5	pos_exist	int	11	记录是否存在	非空

(10) 资产维修表（fams_repair）

支持维修表用于存放资产维修的信息，具体结构如表 3-10 所示。

表 3-10 资产维修表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	rep_id	int	10	主键	主键
2	as_no	varchar	64	资产编号	非空
3	rep_price	double	10,2	维修费用	非空
4	rep_time	datetime	0	维修时间	非空
5	rep_exist	int	11	记录是否存在	非空

(11) 用户信息表（fams_user）

用户信息表用于存放用户的信息，具体结构如表 3-11 所示。

表 3-11 用户信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	u_id	int	10	主键	主键
2	p_id	int	11	人员表主键	非空
3	u_phone	varchar	16	用户电话	非空
4	u_openid	varchar	64	微信openid	非空
5	power_no	varchar	16	用户权限编号	非空
6	u_head	varchar	255	用户头像	非空
7	u_money	int	11	用户积分	非空
8	u_login_code	varchar	255	用户登录码	非空
9	u_exist	int	11	记录是否存在	非空

(12) 帖子信息表（fams_bbs）

帖子信息表用于存放移动端首页所发布帖子的信息，具体结构如表 3-12 所示。

表 3-12 帖子信息表结构

序号	数据库字段	字段类型	字段长度	存储内容	备注
1	bbs_id	int	10	主键	主键
2	u_id	int	11	用户id	非空
3	up_bbs_id	int	11	上级id	非空
4	bbs_text	text	0	帖子内容	非空
5	bbs_time	datetime	0	发布时间	非空
6	bbs_exist	int	11	记录是否存在	非空

第 4 章　主要功能算法的设计与实现

4.1 基于 AES 和 RSA 的混合二维码加密算法

4.1.1 算法的设计

基于 AES 和 RSA 的混合二维码加密算法流程图如图 4-1 所示，核心内容是对于每一个二维码的内容使用不同 AES 秘钥进行 AES 加密，而这个 AES 秘钥也会进行 RSA 加密放置于二维码中，最后得到加密二维码。

为了保证 AES 秘钥的随机性，将会对每一个 AES 秘钥进行随机轮数的生成组合排列，确保无法被暴力破解，然后利用得到的 AES 秘钥对二维码内容进行 AES 加密，得到加密后的二维码内容，再将 AES 秘钥使用 RSA 私钥进行加密得到加密后的 AES 秘钥，再对加密后的二维码内容以及加密后的 AES 秘钥进行 base64 编码，然后再将得到的两个 base64 编码序列化成 json 字符串，最后将 json 字符串再次进行 base64 编码后生成加密后的二维码。

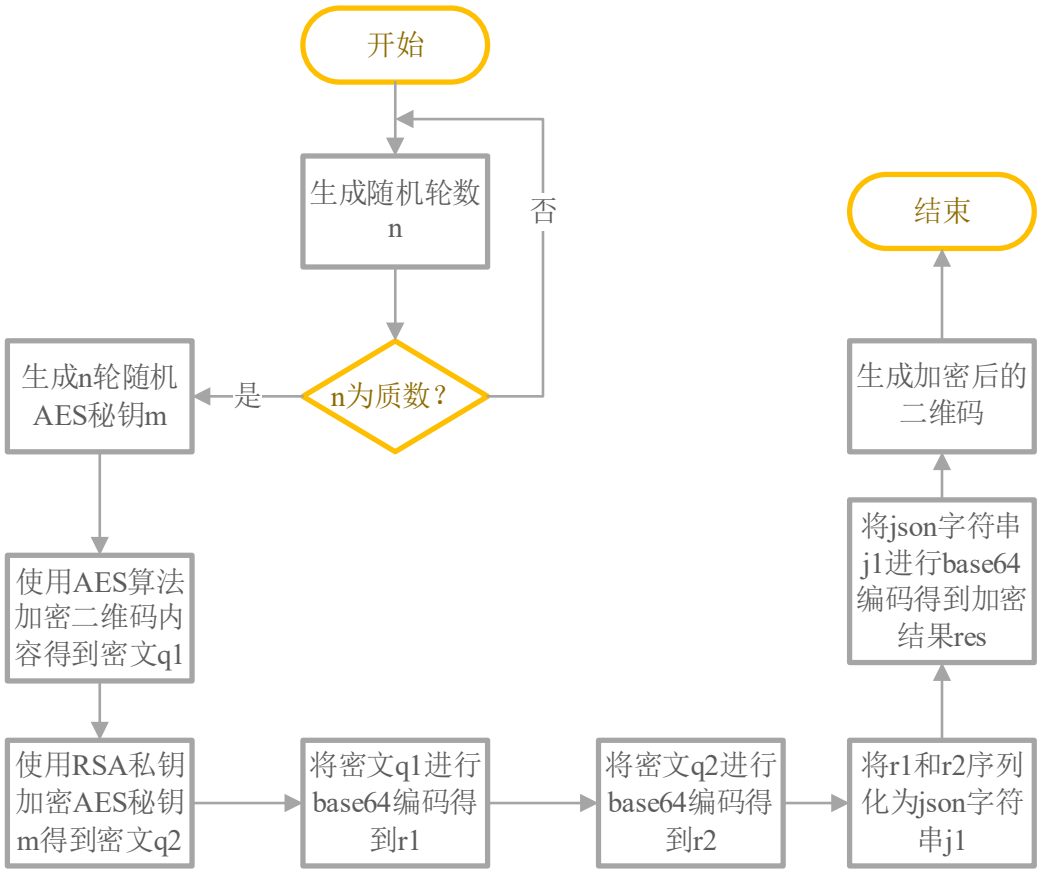

图 4 - 1　二维码加密算法流程图

基于 AES 和 RSA 的混合二维码解密算法流程图如图 4-2 所示，过程基本就是将加密的过程进行逆向操作，重点在于对于随机轮数的判断，如果随机轮数不是质数或者随机轮数不存在则说明二维码内容被篡改。

算法现将密文进行 base64 反编码，再判断随机轮数是否为质数，是质数则继续进行 base64 反编码，得到 AES 秘钥的密文和二维码内容的密文，再使用 RSA 的公钥对 AES 秘钥的密文进行 RSA 解密得到 AES 秘钥，最后使用 AES 秘钥对二维码内容的密文进行 AES 解密的到二维码明文。

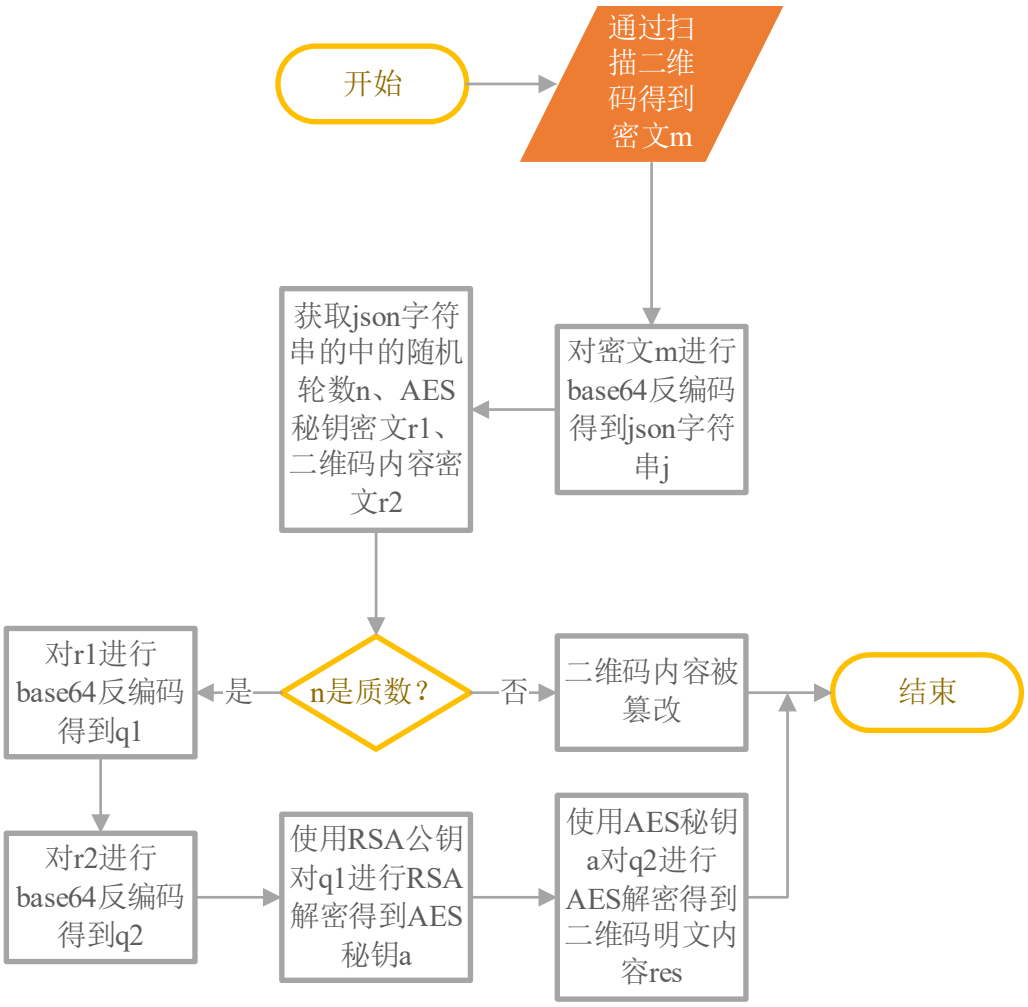

图 4 - 2 二维码解密算法流程图

4.1.2 算法的实现

该算法有两部分，分别为加密算法和解密算法，由于二维码的生成是在 PC 端进行，所以使用 PHP 语言实现加密算法，而扫描二维码是在移动端进行，所以使用 Python 语言实现解密算法。

(1) 加密算法

加密算法首先需要生成一个随机整数 n，该步骤只需要设定一个随机数种子后生成随机整数即可，而后生成随机 AES 秘钥，代码如下：

```
private function createAESm() {
    $no = "";
    $pattern = '1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLOMNOPQRSTUVWXYZ';
    $returnStr = '';
    for($i = 0; $i < rand(10,100); $i ++) {
        $returnStr .= $pattern[mt_rand(0, 61)];
    }
    $no = $returnStr . time();
    $returnStr = '';
    for($i = 0; $i < rand(10,100); $i ++) {
        $returnStr .= $pattern[mt_rand(0, 61)];
    }
    $no .= $returnStr;
    return strtoupper(md5($no));
}
```

然后使用得到 AES 秘钥对二维码内容进行加密，代码如下：

```
/**
 * $data: 要加密的内容
 * $aesM: 要使用的 AES 秘钥
 */
private function aesEncode($data, $aesM){
    $method = 'AES-128-ECB';
    $options = 0;
    $iv = '';
    return openssl_encrypt($data, $method, $aesM, $options, $iv);
}
```

然后使用提前生成好的 RSA 私钥对 AES 秘钥进行加密，代码如下：

```
/**
 * $data: 要加密的内容
 * $rsaPrivateKey: 要使用的 RSA 秘钥数据字符串
 */
private function rsaEncode($data, $rsaPrivateKey) {
    $search = [
        "-----BEGIN RSA PRIVATE KEY-----",
        "-----END RSA PRIVATE KEY-----",
        "\n",
        "\r",
```

```
            "\r\n"
        ];
        $rsaPrivateKey = str_replace($search, "", $rsaPrivateKey);
        $privateKey = $search[0] . PHP_EOL . wordwrap($rsaPrivateKey,
64, "\n", true) . PHP_EOL . $search[1];
        openssl_private_encrypt($data, $rst, $privateKey);
        return $rst;
    }
```

然后对两个密文进行 base64 编码再组合成 json 字符串再次进行 base64 编码，得到最终结果，代码如下：

```
    /**
     * $q1: 加密后的 AES 秘钥
     * $q2: 加密后的二维码内容
     * $n: 随机轮数
     */
    private function createResult($r1, $r2, $n) {
        $jsonData = [
            "n" => $n,
            "r1" => base64_encode($q1),
            "r2" => base64_encode($q2)
        ];
        $jsonString = json_encode($jsonData);
        $result = base64_encode($jsonString);
        return $result;
    }
```

然后将得到的结果生成二维码，代码如下：

```
    public static function createQrcode($value) {
        vendor('phpqrcode');
        $errorCorrectionLevel = 'L';
        $matrixPointSize = 5;
        $filename = 'qrcode/'.$value.'.png';
        \QRcode::png($value,    $filename,    $errorCorrectionLevel,
$matrixPointSize, 2);
    }
```

(2) 解密算法

解密算法需要先从密文中提取出随机轮数、密文 AES 秘钥、密文二维码内容，代码如下：

```
    def loadData(data):
        jsonString = base64.b64decode(data)
```

```
    jsonData = json.loads(jsonString)
    n = jsonData['n']
    q1 = base64.b64decode(jsonData['r1'])
    q2 = base64.b64decode(jsonData['r2'])
    return n, q1, q2
```

然后，判断随机轮数是否为质数，确定二维码是否被篡改，再使用提前生成好的 RSA 公钥对 AES 秘钥的密文进行 RSA 解密，代码如下：

```
# data: 要解密的密文
# rsaPublicKey: 要使用的 RSA 公钥
def rsaDecode(data, rsaPublicKey):
    content = rsa.decrypt(data, rsaPublicKey)
    rst = content.decode("utf-8")
    return rst
```

最后，使用刚得到了 AES 秘钥对二维码密文进行 AES 解密，代码如下：

```
# data: 要解密的密文
# aesM: 要使用的 AES 秘钥
def aesDecode(data, aesM):
    aes = AES.new(aesM, AES.MODE_ECB)
    base64Dec = base64.decodebytes(data.encode(encoding='utf-8'))
    rst = str(aes.decrypt(base64Dec), encoding='utf-8')
    return rst
```

4.2 三通道颜色直方图的资产图片核对算法

4.2.1 算法的设计

三通道颜色直方图的资产图片核对算法流程图如图 4-3 所示，核心内容是通过巴氏系数公式计算 RBG 三通道的颜色直方图数据的相似度的平均值，通过大量数据得到资产核对所需要的阈值。算法采用 Python 语言编写，只用到了计算机视觉库 OpenCV。

首先读取图片并重置两图片的尺寸，确保两图片尺寸相同，这样得到的直方图数据的长度就相同，然后对两图片进行 RGB 通道的拆分，再分别对三通道的数据进行计算，最后取平均值作为结果。

单通道相似度计算需要先获取两图片的颜色直方图数据，对于重合的数据设置为 1，不重合的数据使用巴氏系数公式进行计算，最后得到单通道的相似度。

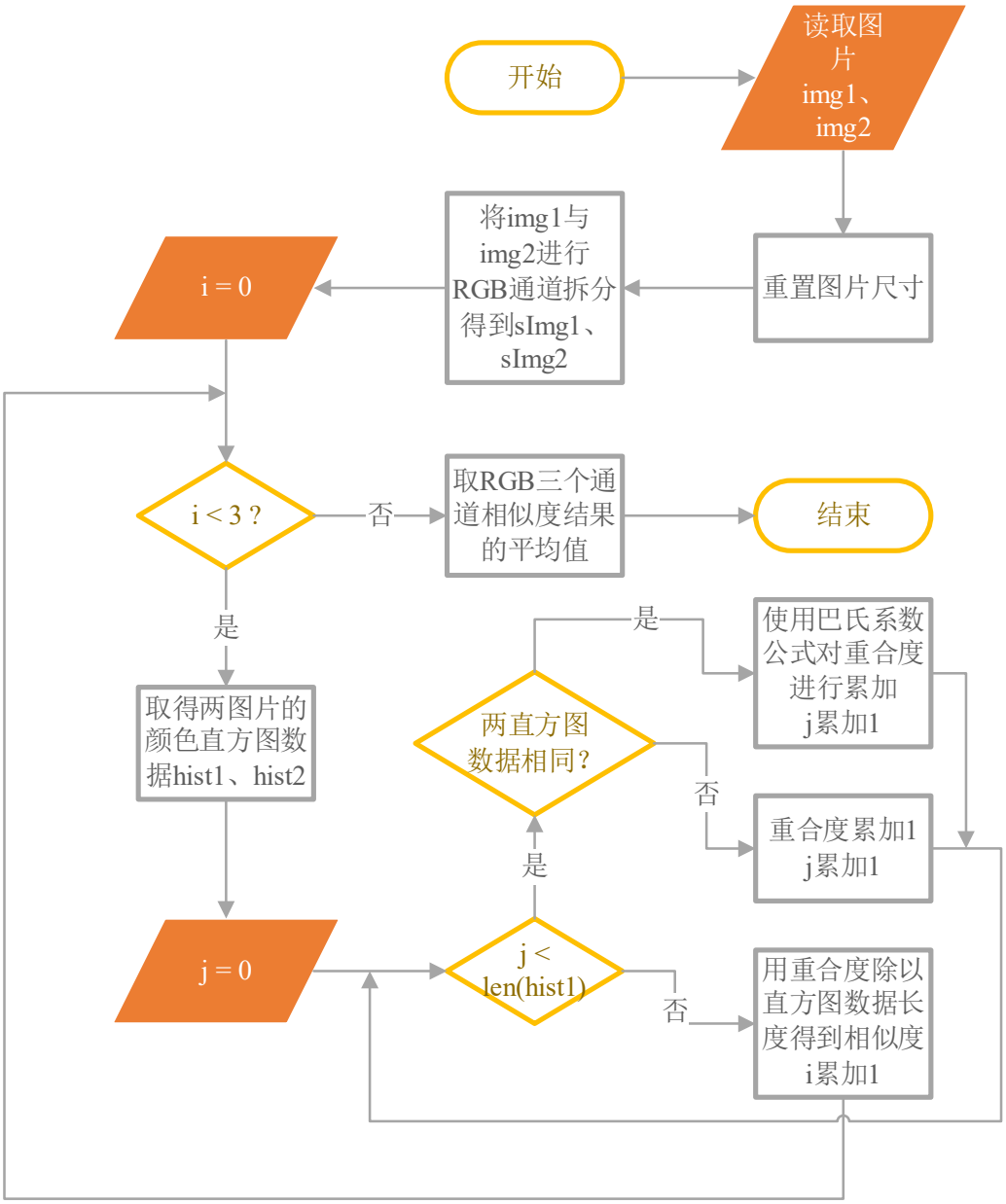

图 4 - 3 资产图片核对算法流程图

4.2.2 算法的实现

该算法是在使用移动端进行资产盘点时使用，所以使用 Python 语言实现该算法。

首选需要读取两张图片，代码如下：

```
# path1：图片 1 路径
```

```
# path2: 图片 2 路径
def openImage(path1, path2):
    image1 = cv2.imread(path1)
    image2 = cv2.imread(path2)
    return image1, image2
```

然后统一图片的尺寸，代码如下：

```
# image1: 图片 1 的 OpenCV 对象
# image2: 图片 2 的 OpenCV 对象
# size: 要统一的图片尺寸
def resizeImage(image1, image2, size=(256, 256)):
    image1 = cv2.resize(image1, size)
    image2 = cv2.resize(image2, size)
    return image1, image2
```

再对 RGB 通道进行拆分，代码如下：

```
# image1: 图片 1 的 OpenCV 对象
# image2: 图片 2 的 OpenCV 对象
def splitImage(image1, image2):
    image1 = cv2.split(image1)
    image2 = cv2.split(image2)
    return image1, image2
```

计算单通道相似值的代码如下：

```
# image1: 图片 1 单通道的 OpenCV 对象
# image2: 图片 2 单通道的 OpenCV 对象
def calculate(image1, image2):
    hist1 = cv2.calcHist([image1], [0], None, [256], [0.0, 255.0])
    hist2 = cv2.calcHist([image2], [0], None, [256], [0.0, 255.0])
    degree = 0
    for i in range(len(hist1)):
        if hist1[i] != hist2[i]:
            degree = degree + \
                (1 - abs(hist1[i] - hist2[i]) / max(hist1[i],
hist2[i]))
        else:
            degree = degree + 1
    degree = degree / len(hist1)
    return degree
```

最后取三通道相似值的平均数，代码如下：

```
# image1: 图片 1 分割后的 OpenCV 对象
# image2: 图片 2 分割后的 OpenCV 对象
def calcResult(image1, image2):
```

```
result = 0
for im1, im2 in zip(image1, image2):
    result += calculate(im1, im2)
result = result / 3
return result
```

4.3 自制弹出层的设计与实现

4.3.1 自制弹出层的设计

ljspopup 是本人自制的仿 Mac OS 窗口风格的弹出层组件，采用 JavaScript 语言基于 jQuery 编写，弹出后的样式如图 4-4 所示。

图 4 - 4 弹出层界面

可以在同一个页面弹出多个弹出层，后弹出的弹出层会在先弹出的上层，如图 4-5 所示。

图 4 - 5 多个弹出层

弹出层可以进行移动，将 move 属性设置为 false 则不可以移动，将鼠标指向标题，按住鼠标拖动即可移动弹出层位置， 弹出层可以最大化、最小化，将 min 属性设置为 false 禁止最小化，将 max 属性设置为 false 禁止最大化，由于是仿造 Mac OS 窗口风格，弹出层左上角有红。橙、绿三种颜色的按钮，红色按钮为关闭弹出层，橙色按钮为最小化弹出层，绿色按钮为最大化弹出层，如图 4-6 所示。

图 4 - 6 最小化弹出层

默认根据内容调整弹出层调整大小，也可以设置 width、height 属性调整大小，默认弹出层居中，也可以设置 top、left 属性调整弹出层位置，弹出层内容可以为文字、HTML 内容、元素 id。

而弹出层的弹出、关闭、最小化、最大化等操作，是通过 jQuery 自定义动画实现，弹出首选设置弹出层的宽和高都为 0，然后再扩大到预设宽高的 110%，最后恢复到预设的宽高，这样就实现了动态的弹出效果，关闭则是将宽高直接缩小为 0，而最小化则是先计算出最小化的位置再将宽高以及位置通过 jQuery 动画缩小，实现动态的最小化，而这些操作都可以禁用，如图 4-7 是禁用最大化、最小化、移动以及不显示遮罩层的效果图。

图 4 - 7 禁用效果图

4.3.2 自制弹出层的实现

弹出层只有 ljspopup.js 和 ljspopup.css 两个文件，JS 文件用于编写弹出层的逻辑算法，CSS 文件用于编写弹出层样式。

因为可以弹出多个弹出层，所以定义数组 popup_data 用于存储每一个弹出层数据，这样就可以记录已经创建的弹出层的个数，避免弹出层的重复创建，浪费系统资源造成卡顿，而且还可以记录弹出层的位置信息和大小，方便最大化、最小化、还原时的数据初始化，调用弹出层是的参数为一个字典，所以弹出的函数定义应为：

```
function ljspopup(data = {}){}
```

data 即为传入的参数，默认为空字典，即为缺省数据，一般调用时会传入 el、title 等参数，调用此函数并传入参数，首先创建该弹出层的 sign，也就是唯一编号，并设置在 id 中，用于辨认弹出层，并判断该弹出层是否已经被创建，代码如下：

```
var panel_sign = "ljs-sign-" + $.md5(JSON.stringify(data));
var index = sign_find(panel_sign);
```

如果弹出层已经被创建则直接显示弹出层即可，如该弹出层没有被创建，则应先创建弹出层，要想创建弹出层，首先需要对参数进行初始化，确保每个参数都有一个缺省值，代码如下：

```
var new_data = {
    el: data.el ? data.el : "",
    title: data.title ? data.title : "",
    width: data.width || data.width == 0 ? data.width : -1,
    height: data.height || data.height == 0 ? data.height : -1,
    top: data.top || data.top == 0 ? data.top : -1,
    left: data.left || data.left == 0 ? data.left : -1,
    min: data.min != undefined ? data.min : true,
    max: data.max != undefined ? data.max : true,
    mask: data.mask != undefined ? data.mask : true,
    move: data.move != undefined ? data.move : true,
    can_move: true,
    sign: panel_sign
};
```

然后创建弹出层所需的元素，即创建 DOM，使用原生 JS 中的 document.createElement()方法创建，并保存在相应变量中，代码如下：

```
var ljs_panel = document.createElement("div");
var ljs_panel_header = document.createElement("div");
var ljs_panel_header_three_btn = document.createElement("div");
var ljs_panel_header_close = document.createElement("div");
var ljs_panel_header_min = document.createElement("div");
var ljs_panel_header_max = document.createElement("div");
var ljs_panel_header_title = document.createElement("div");
var ljs_panel_body = document.createElement("div");
```

再对已经创建好的弹出层元素赋予 CSS 样式，利用 jQuery 将 DOM 对象转换为 jQuery 对象，利用 jQuery 中的 addClass()对其添加预设好的 CSS 样式，代码如下：

```
$(ljs_panel).addClass("ljs-panel");
$(ljs_panel).addClass(panel_sign);
$(ljs_panel_header).addClass("ljs-panel-header");
$(ljs_panel_header_three_btn).addClass("ljs-panel-header-three-b
tn");
$(ljs_panel_header_close).addClass("ljs-panel-header-close");
$(ljs_panel_header_min).addClass("ljs-panel-header-min");
if (!new_data.min) {

$(ljs_panel_header_min).addClass("ljs-panel-header-btn-disabled");
}
$(ljs_panel_header_max).addClass("ljs-panel-header-max");
if (!new_data.max) {

$(ljs_panel_header_max).addClass("ljs-panel-header-btn-disabled");
}
$(ljs_panel_header_title).addClass("ljs-panel-header-title");
if (!new_data.move) {
    $(ljs_panel_header_title).css("cursor", "default")
```

```
    }
    $(ljs_panel_body).addClass("ljs-panel-body");
```

然后对弹出层元素赋予内容并添加到<body>中，利用 jQuery 中的 html 方法设置 DOM 对象的 innerHTML 属性，并使用 jQuery 中的 append 方法代替 DOM 对象的 appendChild 方法向文档中添加 DOM 元素，代码如下：

```
    $(ljs_panel_header_close).html("×");
    $(ljs_panel_header_min).html(" - ");
    $(ljs_panel_header_max).html(" + ");
    $(ljs_panel_header_title).html(new_data.title);
    $(ljs_panel_body).append($(new_data.el));
    $(ljs_panel_header_three_btn).append(ljs_panel_header_close);
    $(ljs_panel_header_three_btn).append(ljs_panel_header_min);
    $(ljs_panel_header_three_btn).append(ljs_panel_header_max);
    $(ljs_panel_header).append(ljs_panel_header_three_btn);
    $(ljs_panel_header).append(ljs_panel_header_title);
    $(ljs_panel).append(ljs_panel_header);
    $(ljs_panel).append(ljs_panel_body);
    $("body").append(ljs_panel);
```

然后对弹出层的位置进行调整，并添加到数组 popup_data 中，代码如下：

```
    $(ljs_panel).find(".ljs-panel-body").children().show();
    new_data.width = new_data.width < 0 ? $(ljs_panel).width() :
new_data.width;
    new_data.height = new_data.height < 0 ? $(ljs_panel).height() :
new_data.height;
    new_data.width = new_data.width > $(window).width() ?
$(window).width() : new_data.width;
    new_data.height = new_data.height > $(window).height() ?
$(window).height() : new_data.height;
    popup_data.push(new_data);
```

然后判断是否显示遮罩层，首先查看<body>中是否存在遮罩层元素，如果不存在则创建遮罩层元素并添加到<body>中，然后根据弹出层的属性判断是否显示

遮罩层，代码如下：

```
if (!el_exist(".mask")) {
    var mask = document.createElement("div");
    $(mask).addClass("mask");
    $("body").append(mask);
}
if (data.mask) {
    $(".mask").show();
}
```

然后读取弹出层配置，弹出层的配置即使传入的参数 data，读取配置用于计算弹出动画的元素位置等信息，代码如下：

```
var width = data.width;
var height = data.height;
var window_width = $(window).width();
var window_height = $(window).height();
var top = data.top >= 0 ? (data.top + (data.height / 2)) : window_height / 2;
var left = data.left >= 0 ? (data.left + (data.width / 2)) : window_width / 2;
var width_out = width / 10;
var height_out = height / 10;
```

最后以 jQuery 动画的方式将弹出层显示出来，代码如下：

```
$panel.css({
    width: 0,
    height: 0,
    top: top,
    left: left,
    display: "flex"
}).animate({
    width: "+=" + (width + width_out) + "px",
    height: "+=" + (height + height_out) + "px",
    top: "-=" + (height + height_out) / 2 + "px",
```

```
        left: "-=" + (width + width_out) / 2 + "px"
    }, 150).animate({
        width: "-=" + width_out + "px",
        height: "-=" + height_out + "px",
        top: "+=" + height_out / 2 + "px",
        left: "+=" + width_out / 2 + "px"
    }, 100);
```

至此，弹出层显示完毕，然后需要进行事件监听，监听关闭的代码如下：

```
    $(document).on("click", ".ljs-panel-header-close", function () {
        closeLjspopup($(this).parent().parent().parent());
    });
    function closeLjspopup($panel) {
        var width = $panel.width();
        var height = $panel.height();
        var width_out = width / 10;
        var height_out = height / 10;
        $(".mask").hide();
        $panel.animate({
            width: "+=" + width_out + "px",
            height: "+=" + height_out + "px",
            top: "-=" + width_out / 2 + "px",
            left: "-=" + height_out / 2 + "px"
        }, 100).animate({
            width: 0,
            height: 0,
            top: "+=" + (height + height_out) / 2 + "px",
            left: "+=" + (width + width_out) / 2 + "px"
        }, 150, function () {
            $panel.hide();
            if ($panel.parent().attr("class") != undefined &&
$panel.parent().attr("class").indexOf("ljs-panel-min-area") >= 0) {
                $panel.css("position", "fixed");
```

```
            $panel.parent().remove($panel);
            $("body").append($panel);
        }
    });
}
```

弹出层最小化需要在网页最下面创建一个区域用于存放最小化的弹出层，采用从右到左，从下到上的方式排列，代码如下：

```
if (el_exist(".ljs-panel-min-area")) {
    $(".ljs-panel-min-area").show();
} else {
    var ljs_panel_min_area = document.createElement("div");
    $(ljs_panel_min_area).addClass("ljs-panel-min-area");
    $("body").append(ljs_panel_min_area);
}
```

然后，根据已有最小化弹出层数量和屏幕的宽度计算此弹出层需要取得绝对位置，为最小化动画做准备，代码如下：

```
var min_width = 200;
var min_line_height = 38;
var window_width = $(window).width();
var window_height = $(window).height();
var width = $panel.width();
var height = $panel.height();
var width_out = width / 10;
var height_out = height / 10;
var now_number = $(".ljs-panel-min-area").children().length;
var max_number = Math.floor(window_width / min_width);
var c = now_number / max_number;
var now_line = c < Math.floor(c) ? Math.floor(c) : Math.floor(c) +
1;
var now_col = (now_number % max_number) + 1;
```

最后，进行 jQuery 动画，再将此弹出层加入底部区域，代码如下：

```
$(".mask").hide();
```

```
    var sign = get_sign($panel);
    var index = sign_find(sign);
    popup_data[index].can_move = false;
    $panel.animate({
        width: "+=" + width_out + "px",
        height: "+=" + height_out + "px",
        top: "-=" + width_out / 2 + "px",
        left: "-=" + height_out / 2 + "px"
    }, 100).animate({
        width: 0,
        height: 0,
        top: (window_height - (now_line * min_line_height -
min_line_height / 2)) + "px",
        left: (window_width - (now_col * min_width - min_width / 2)) +
"px"
    }, 150, function () {
        $panel.css("position", "static");
        $(".ljs-panel-min-area").append($panel);
        $panel.find(".ljs-panel-header-title").css("justify-content",
"flex-start");
    }).animate({
        width: min_width + "px",
        height: min_line_height + "px"
    }, 100);
```

4.4 自制 Python 数据库访问类的设计与实现

4.4.1 自制 Python 数据库访问类的设计

在编写 Python 代码的过程中对数据的操作必不可少，对于一个项目，如果涉及到大量 SQL 语句，会导致程序的可读性变差，出现 bug 也不容易排查，所以封装一个好的数据库访问类尤为重要。

Ljsmysql 是本人使用 Python 语言基于 pymysql 编写的 MySQL 数据库访问类，

无需编写 SQL 语句即可实现数据库增删改查操作。使用 connect 函数进行数据库连接，使用 table 函数选择要操作的表，使用 where 函数设置条件，使用 select、find 函数进行数据查找，使用 insert 函数插入数据，使用 update 函数修改数据，使用 delete 函数删除数据，使用 order 函数进行排序。例如：`Ljsmysql.table('test').where('id', 1).select()`即可查找符合条件的记录，`Ljsmysql.table('test').insert({'id': 2, 'name': 'ljs'})`即可插入记录。

4.4.2 自制 Python 数据库访问类的实现

访问数据库首先需要进行数据库的连接，封装为 connect 函数，代码如下：

```
@staticmethod
def connect(localhost="127.0.0.1", username="root", password="", dbname="mysql", encoding="utf8"):
    Ljsmysql.db = pymysql.connect(
        localhost, username, password, dbname, charset=encoding)
    Ljsmysql.cursor = Ljsmysql.db.cursor(pymysql.cursors.DictCursor)
```

然后使用 table 函数来选择要操作的表，返回一个 Table 对象，再根据需要进行操作，where 函数可传递多样值：

(1) 向 where 函数传递两个值

例如：`where('id', 1)`

第一个值是字段名，第二个值是值，两者是‘=’关系。

(2) 向 where 函数传递三个值

例如：`where('id', '<>', 1)`

这里第二个值为字段与值之间的关系，不再只限于‘=’，例如：‘>’，‘<’，‘<>’，‘like’，‘not like’。

(3) 向 where 中传递字典

例如：
```
where({
    'id': 1,
    'name': 'ljs'
})
```

这样实现了多条件，条件之间关系为 AND，字段与值之间关系为‘=’。

(4) 向 where 中传递列表

例如：where([

['id', '<>', 1],

['name', 'like', '_js']

])

这样就拓展了字段与值之间的关系，不再只限于‘=’。

where 函数实现代码如下：

```
def where(self, p1, p2=False, p3=False):
    if p2 == False:
        if isinstance(p1, dict):
            tmpList = []
            for k, v in p1.items():
                tmpList.append("{0} = '{1}'".format(k, v))
            self.whereSql = "where {0}".format(" and ".join(tmpList))
        if isinstance(p1, list):
            tmpList = []
            for p in p1:
                if len(p) == 2:
                    tmpList.append("{0} = '{1}'".format(p[0], p[1]))
                if len(p) == 3:
                    tmpList.append("{0} {1} '{2}'".format(p[0], p[1],
p[2]))
            self.whereSql = "where {0}".format(" and ".join(tmpList))
    if p2 != False and p3 == False:
        self.whereSql = "where {0} = '{1}'".format(p1, p2)
    if p3 != False:
        self.whereSql = "where {0} {1} '{2}'".format(p1, p2, p3)
    return self
```

其余的 select、update、delete、insert 就是将条件拼接成 SQL 语句并执行，以 update 函数举例，实现代码如下：

```
def update(self, p1, p2=False):
    if p2 != False:
        updateSql = "set {0} = '{1}'".format(p1, p2)
```

```
        else:
            tmpList = []
            for k, v in p1.items():
                tmpList.append("{0} = '{1}'".format(k, v))
            updateSql = "set {0}".format(",".join(tmpList))
        sql = "update {0} {1} {2}".format(self.tableName, updateSql,
self.whereSql)
        Ljsmysql.exec(sql)
        Ljsmysql.db.commit()
```

第5章 结 论

设计并实现了高校固定资产管理系统。服务端部署在阿里云服务器，使用Apache 和 Tornado 作为 Web 服务器。客户端以网站、微信小程序、手机 APP 的方式进行发布，提供了资产管理、盘点管理、审批管理、系统管理 4 个功能模块，移动端还可以进行帖子发布与回复。系统后台采用 PHP 和 Python 语言编写，PHP 后台运行在 Apache 中，Python 后台采用 Tornado 框架运行，可稳定运行，达到了课题预期设计目标。

本文的主要设计工作结论如下：

(1) 设计并实现了基于 AES 和 RSA 的混合二维码加密算法，使资产信息更加安全，无法被暴力破解。

(2) 设计并实现了三通道颜色直方图的资产图片核对算法，在用户进行资产盘点后，对所盘点的资产的图片进行核对，减少了管理员的工作量。

(3) 申请了 SSL 证书，采用 HTTPS 协议和 WSS 协议进行 Request 请求以及 WebSocket 连接，保证了数据传输的安全性。

(4) 设计并实现了帖子功能，用户之间可以发帖以及恢复，自行解决固定资产的使用问题，减少了管理员的工作量。

课题后续的研究重点内容包括：

(1) 可以将帖子功能发展成为论坛，加入发送表情、图片功能，并加入文章发表功能，评论点赞功能，采用 Markdowm 格式进行文章的编写。

(2) 对资产折旧的算法进行设计，目前系统为加入资产折旧功能，后续将对资产折旧的算法进行设计并实现。

(3) 将财政管理加入该系统中，学生可以查看奖学金的事宜，教师可以查看奖金的事宜，可以申请固定资产的购买。

参考文献

[1] 张慕然. 高校固定资产管理系统设计与实现[D]. 长江: 长江大学, 2019.

[2] 姜宇飞. 设备资产管理的设计与实现[D]. 青岛: 青岛大学, 2019.

[3] 张华. 基于非对称加密算法的 QR 二维码[J]. 电子技术与软件工程, 2018(05): 29-29.

[4] 高硕. 基于内容的图像检索系统研究与实现[D]. 唐山: 华北理工大学, 2019.

[5] 刘海峰, 刘洋, 梁星亮. 一种结合优化后 AES 与 RSA 算法的二维码加密算法[J]. 陕西科技大学学报, 2019, 37(06): 153-159.

[6] 王宁邦, 徐博, 夏百川, 夏娜, 郃永航, 李琼. 一种颜色直方图计算相似度的资产图片核对算法[J]. 智能计算机与应用, 2018, 8(02): 59-62, 67.

[7] 李先懿, 郭正光. 基于 Websocket 的车联网报警推送系统[J]. 计算机系统应用, 2020, 29(03): 127-131.

[8] 柴青山. 基于 MVVM 模式的 Vue.js 框架在物流软件自动化测试系统中的应用研究[D]. 北京: 北京邮电大学, 2019.

[9] 许溜溜. 基于 HBuilder 快速开发移动端 APP 的设计与实现[J]. 电脑知识与技术, 2020, 16(10): 74-75.

[10] 林国梁. 基于 Web 的移动终端取证管理系统的设计与实现[D]. 厦门: 厦门大学, 2018.

[11] 李全全. Python OpenCv 在智慧党建人脸识别中的应用[J]. 中国有线电视, 2020, (02): 167-171.

[12] 田维铝, 邓梅, 王晓华, 马宁. 置管病人管理系统中 H5、TP5 关键技术的研究与实现[J]. 电脑编程技巧与维护, 2020, (04): 86-90.

[13] 刘晶. 高校固定资产管理系统的设计与实现[D]. 大连: 大连理工大学, 2018.

[14] Seth James Nielson, Christopher K. Monson. Asymmetric Encryption: Public/Private Keys[M]. Berkeley: Apress, 2019.

[15] Stepan Sivkov, Leonid Novikov, Galina Romanova, Anastasia Romanova, Denis Vaganov, Marat Valitov, Sergey Vasiliev. The algorithm development for operation of a computer vision system via the OpenCV library[J]. Procedia Computer Science, 2020, (10): 169-169.

[16] Guerrero-Sanchez Alma E, Rivas-Araiza Edgar A, Gonzalez-Cordoba Jose Luis, Toledano-Ayala Manuel, Takacs Andras. Blockchain Mechanism and Symmetric Encryption in A Wireless Sensor Network.[J]. Sensors (Basel, Switzerland), 2020, 20(10): 2798-2798.

[17] S. A. Malakh, V. V. Servakh. Maximization of Unit Present Profit in Inventory Management Systems[J]. Automation and Remote Control, 2020, 81(3): 843–852.

[18] 孙霓刚, 陈宣任, 朱浩然. 一种基于整数多项式环上的非对称全同态加密方案[J]. 现代电子技术, 2020, 43(05): 86-91.

[19] 毕红棋, 陈露. 基于混合算法的加密与解密的应用研究[J]. 现代信息科技, 2020, 4(03): 145-147, 150.

[20] 邵凯, 毛云龙. 二维码加密技术的研究[J]. 数字技术与应用, 2020, 38(02): 189-190.

致　谢

感谢我的导师邵中老师。在老师的悉心指导和严格要求下完成了本论文，导师的心血和汗水凝聚在本论文的各个方面：课题选择、方案论证到具体设计和实现。在四年的大学学习和生活期间，导师的精心教导让我更好的掌握和运用专业知识，并在论文中得以体现，不仅如此，日常的学习中也始终能够感受到导师无私的关怀。在此向邵中老师表示深深的感谢和崇高的敬意。

同时也感谢同学以及其他各位老师在思路想法上对我提供的帮助。感谢室友大学四年来的陪伴，共同走过这美好的大学生活。