计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(4):69−76 [doi: 10.15888/j.cnki.csa.007843]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Python 的高校电子文档管理系统①

黄　昇

(上海旅游高等专科学校 设备处, 上海 201418)
通讯作者: 黄　昇, E-mail: shengshengking@163.com

摘　要: 随着高校采购任务的剧增, 采购业务积累了大量的电子文档资料, 原一站式采购管理平台的文档管理功能已经无法满足现有的工作需要. 综合分析现有文件和资料管理的需求, 并根据文档实际生命周期的业务流程, 确定了系统功能模块的划分. 利用模型驱动工程思想建立系统的对象模型, 使用 Rational 建模工具建立系统类图和时序图来描述系统整体架构和业务逻辑, 选择轻量级 Flask 框架模型进行研发, 采用文档型数据库 MongoDB 解决大并发量和数据服务器的读写压力, 为今后大数据分析提供保障提出 PyPDF 方法解决 PDF 元数据提取功能. 最终解决了电子文档流转最后归档环节缺乏信息化管理的问题.

关键词: 文档管理; 模型驱动; 非关系型数据库; Flask 框架; 元数据

引用格式: 黄昇.基于 Python 的高校电子文档管理系统.计算机系统应用,2021,30(4):69–76. http://www.c-s-a.org.cn/1003-3254/7843.html

Electronic Document Management System in Colleges and Universities Based on Python

HUANG Sheng

(Equipment Department, Shanghai Institute of Tourism, Shanghai 201418, China)

Abstract: Amid the leap in procurement tasks in colleges, a large number of electronic documents have been accumulated. The original one-stop procurement management platform has been unable to manage such a great number of documents. In light of current demand for data management, the system function modules are divided according to the business process throughout the actual life cycle of documents. The object model of the system is built based on the idea of model-driven engineering, and class and sequence diagrams are drawn by rational modeling tools to describe the overall architecture and business logic of the system. Besides, the lightweight flask framework is adopted for research and development, and a document-oriented database (MongoDB) is relied on to tackle the problem of the large concurrent and release the pressure on the data server during reading and writing. For future big data analysis, a PyPDF method is proposed to facilitate the extraction of PDF metadata. As a result, information management is ensured for the final filing during electronic document circulation.

Key words: document management; model driven; non-relational database; flask framework; metadata

随着高校信息化建设的高速发展, 越来越多的电子文档出现在日常的工作中, PDF 作为电子文档归档的首选格式, 在文件格式的保存完整性方面和平台兼容性方面有显著的优势[1]. 本电子文档归档管理系统的研发主旨是将采购管理平台的文档管理与档案归档管理过程合并成一个整体, 解决电子文档流转过程中信息化管理缺失的情况, 同时提出一种对电子文档元数据自动提取来代替传统的手工提取元数据, 并且建立索引库, 为大数据分析奠定了基础, 同时为学校今后的工作决策提供依据.

国内对元数据提取的相关探索起步较晚, 主要研究方向也集中在基于正则表达式和基于规则的元数据

① 收稿时间: 2020-07-27; 修改时间: 2020-08-26; 采用时间: 2020-09-01; csa 在线出版时间: 2021-03-30

提取的相关研究[2–6]. 2001 年贺亚锋首次将元数据提取的相关研究带入到中国[7], 主要对两种常用的基于网站的元数据的自动生成进行了介绍, 并对 ROADS 元数据编辑器和 MeatWeb 元数据生成器做的使用和原理进行了深入的阐释[8,9].

2004 年, 王守芳等提出了如何从 HTML 文件中提取元数据的方案. 该方案主要是基于规则模板, 通过对 HTML 文档进行分词, 配合使用归约算法实现元数据的自动提取[10]. 该方法虽然对元数据提取的准确性却并不太高, 但基本可以实现 HTML 文档元数据的自动提取.

2007 年, 于江德等首次将条件随机场应用在中文论文的元数据提取上, 该方法主要通过利用论文中换行符、回车符等标志性符号对论文内容进行分割, 然后应用条件随机场对分割内容进行元数据抽取[11,12]. 该方法对于学术论文的论文头中的元数据的提取具有比较高的准确度, 可以高达 90%, 但是该方法也局限于论文的头部进行元数据的提取操作.

2017 年, 杜秋霞等为了地名文化遗产的保护将隐马尔可夫模型应用在提取文献中的地名元数据上[13]. 该方法主要通过对电子文档的地名关键词的标注, 然后对文本进行分割, 进而对元数据进行提取. 该方法可以对文献中的地名进行比较细粒度的抽取, 相比传统地名提取的准确度明显提升, 但该方法却无法对消失的地名准确的进行抽取.

通过阅读相关的文献, 研究对比目前流行提取 PDF 元数据的各种方式, 结合实际需求提出一种最适合本课题的提取方法. 通过对 Flask 框架的学习, 以模型驱动工程的思想设计实现一款基于 Python 的能够自动、高效、准确地提取 PDF 中元数据的电子文档归档管理系统.

1 系统需求分析

目前线下归档流程仍旧是文档在各部门之间通过复印和填写文档属性表格资料的方式传递. 职能部门为执行科研项目建立文档库, 将其他各类格式的电子文档和实体文档转换或扫描成 PDF 格式保存, 同时在项目执行过程中不断更新该文档库, 项目完成后发起归档任务将数据迁移至档案部门审核, 然后根据项目分类存入对应的档案系统中[14–16]. 期间需要经历很多繁琐的流程, 而且还有诸多弊端: (1) 项目庞大, 涉及的供应商有多家, 文档的完整性无法保证; (2) 项目执行过程只有职能部门参与其中, 对归档工作没有一个过程把控的机制, 档案部门只是在归档任务发起才参与进来; (3) 归档任务通常集中在年末, 会积压大量的资料, 这就势必在文档流转审核过程中产生错误和审核不严格的情况.

因此, 需要将归档管理功能一并纳入一站式采购平台且做进一步的完善. (1) 增加文档数据导入功能, 保证项目执行期间文档实时保存; (2) 增加监督审核机制, 保证上传电子文档的准确性, 避免项目后期返工的情况; (3) 增加电子文档元数据提取功能, 解决目前针对海量数据缺乏大数据分析的情况. 这样电子文档管理和归档管理就涵盖了整个文档生命周期, 如图 1 所示.

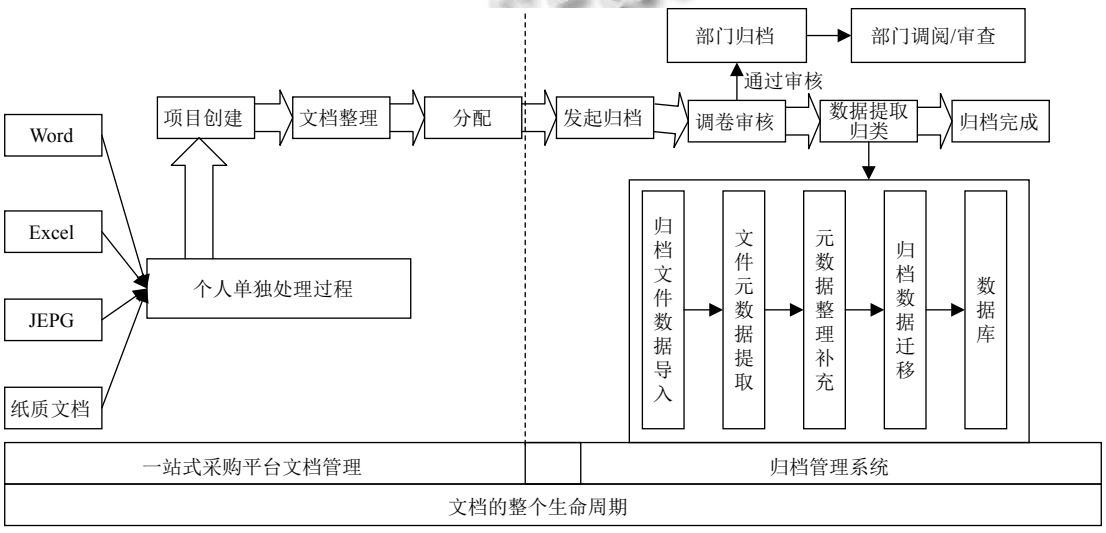

图 1 文档生命周期

2 系统概要设计

2.1 系统基本思想

(1) 系统以形成高校一体化的信息高度集成为基准, 设计标准数据接口防止信息“孤岛”的产生, 做到新系统与一站式采购管理平台的无缝结合[17–19].

(2) 出于对安全性的考虑, 系统对数据库管理要采取必要的定期自动数据备份、防灾预案和数据恢复等措施; 在数据传输方面要充分利用校园数据交互中心的标准接口, 确保系统数据的安全性、可靠性和一致性[20]; 对所有用户的权限必须要有有效的管控机制(如: 归档角色、审批权限).

(3) 设计阶段充分考虑后期需求的变化, 系统后台配置应具备灵活性, 例如需要增加新的文档属性项时只要通过系统后台配置即可, 无需修改系统程序和数据结构.

2.2 系统功能设计

根据需求分析和设计思路可以得到系统的用例图, 如图 2 所示, 将本系统分为项目文档整理、检索与统计、移交接收管理和系统管理 4 个功能区.

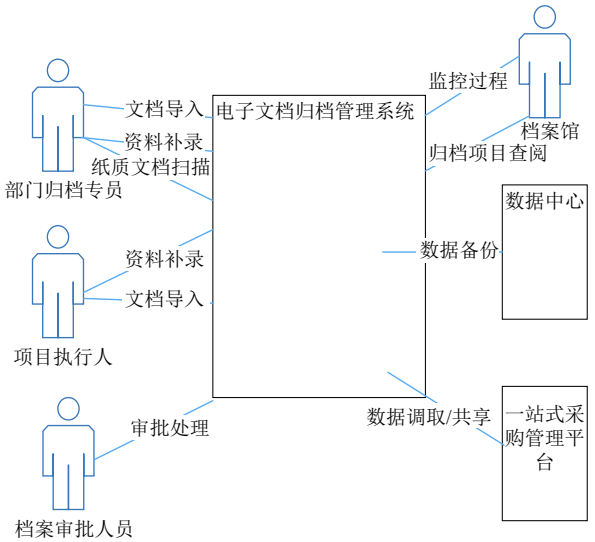

图 2 归档系统用例图

(1) “项目文档整理”包含 6 项子功能: 文档导入(自动导入、人工录入、本地导入), 文档补充, 项目信息补录, 删除文档, 文档格式转换, 文档元数据提取.

(2) “检索与统计”包含 4 项子功能: 归档任务进度查询, 电子文档查询, 数据统计, 报表管理 (包含模板管理).

(3) “移交接收管理”包含 3 项子功能: 预审, 资料移交审核, 归档内容审核.

(4) “系统管理”包含 3 项子功能: 用户和权限配置, 文件扩展属性管理, 各类标准接口配置.

2.3 数据库设计

在大数据背景下, 要求应用系统具有高性能、弱事务的特性, 因此数据结构需要以横向扩展的方式进行分布式存储, 数据模式多元化, 数据相对独立存在. 通过功能业务分析电子文档归档系统并非事务性系统, 为了使本系统在扩展性、并发处理和读/写方面更有优势, 而且需要考虑到系统后期的升级与功能扩展, 摒弃使用传统关系数据库, 采用 MongoDB 半结构化的非关系型数据库, 它有着分布式的存储架构, 这样数据之间分散存储更容易扩展, 数据库不需要事先定义数据字段, 可以随时自定义写入数据的格式. NoSQL 来处理大量多元化数据存储运算与高并发访问有更显著的效果[21–23]. 数据库采用 1 主节点+1 副节点+1 仲裁节点的基本架构, 以减轻数据服务器的访问压力, 同时提升容灾能力, 如图 3 所示.

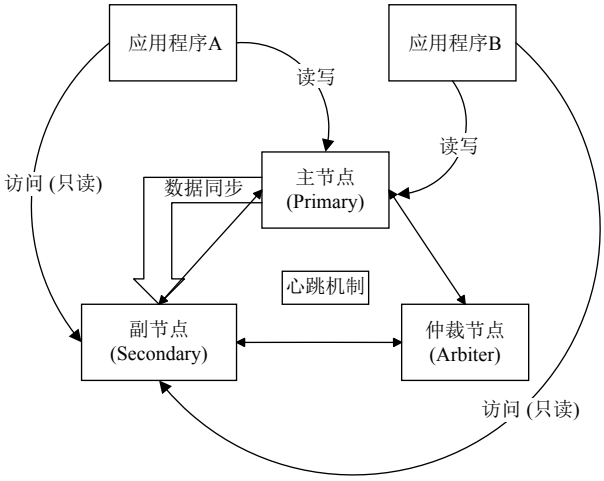

图 3 数据库存储架构

3 系统对象模型的建立

3.1 业务逻辑分析

通过对实际业务进行系统建模更有利于把握系统的整体格局, 在更高的抽象水平上考虑系统的设计, 而不是程序编码, 这样可以降低前期出错率, 缩短系统实现的周期[24,25]. 由于篇幅有限仅以文档迁移导入为例, 其包括的主要功能: 文档锁定, 项目移交接收, 文档信息补充, 本地导入, 具体业务逻辑如表 1 所示.

表 1 文档迁移导入主要业务逻辑

序号	操作名	逻辑处理内涵
1	文档锁定	①在归档任务发起的情况下, 系统将该项目资料库的文档进行锁定封存, 并整体迁移至归档库. ②锁定后的文档无法操作, 并自动转换PDF格式. 元数据提取过程执行平台的依赖性, 采取单独开发文档格式转换和元数据提取插件功能块来实现.
2	项目移交接收	①项目移交任务审核过程中, 系统允许“文档迁移导入”、“文档信息补录”和“本地导入”等操作. ②点击“文档迁移导入”, 弹出归档库项目的选择/查询页面, 选中要迁移的归档项目, 点击“确定迁移”. ③文档导入操作在送审之前允许多次操作. “确定迁移”即为进入送审阶段, 导入功能被锁定, 只能对文档信息进行补录, 只有审批驳回情况下能够解锁并重新导入.
3	文档信息补充	如归档的文件信息不完整, 具体补录操作如下: ①根据归档项目的类型弹出填写相关信息的输入界面. ②根据归档项目的类型补录上传相关的文档. ③补录的归档项目资料信息将同步写入数据库. 文档则要通过审核流程才能上传.
4	本地导入	在项目移交审核不合格, 还需补充文档时, 从本地客户端直接上传至归档库对应项目, 无需再上传至项目库然后迁移, 同时上传过程中完成格式转换和元数据提取.

表 2 整个系统的类定义集合 (类集)

序号	类的英文名称	类的说明
1	AmObject	对象抽象类, 系统所有模型类的基类
2	AmObjectDAO	对象数据操作抽象类, 其他模型数据操作的基类
3	ConDefiner	公共类(包含系统通用类, 翻页, 日期格式等等)
4	ModelManager	EJB类
5	AmAuthority	权限验证类, 负责所有和权限相关的操作
6	AmRules	归档类型类
7	AmRulesDAO	归档类型操作类
8	AmModelProperty	文件属性类
9	AmModelPropertyDAO	文件属性操作类
10	AmInitTask	归档任务类
11	AmInitTaskDAO	归档任务操作类
12	AmProcTask	过程任务类
13	AmProcTaskDAO	过程任务操作类
14	AmRollInfo	案卷类
15	AmProcOpinion	审批意见类
16	AmProcOpinionDAO	审批意见操作类
17	AmDocModel	文件模型类
18	AmDocModelDAO	文件模型操作类
19	AmDocType	文件类型模型类
20	AmDocTypeDAO	文件类型操作类

3.2 类定义与类间关系

根据模型驱动工程的思想方法, 首先建立系统的对象模型, 接着通过对象模型建立系统类集, 并对每个类都定义属性和操作方法, 如表 2 所示.

(1) ModelManager 属于 EJB 类, 封装的组件为前台与服务端的交互提供数据访问接口.

(2) 公共类 (ConDefiner), 它主要封装了基本查询、第三方插件调用和翻页等方法, 前台只需要实例化这个类就能继承并使用.

(3) 对象抽象类: 将实体类 AmObject 类作为系统 Model 类的基类, Model 类就是把数据库的字段映射为各类中各个对象的属性, 为模型操作类提供数据来源.

(4) 数据操作抽象类: 将 AmObjectDAO 类作为数据操作类的基类, 除了从父类 AmObject 继承的一些通用对象数据操作, 还自定义特殊的数据对象操作方法. 如图 4 所示.

3.3 数据操作类的逻辑实现

利用 UML 序列图能描述对象间的交互和消息传递顺序的特性, 完成系统核心功能模块对象间的输入输出. 以文档迁移导入、文档数据补充、本地导入和移交接收的逻辑实现为例.

文档迁移导入设计思路: 根据前台应用操作通过 EJB 类调用文件模型操作类的具体方法, 数据库返回对应的项目列表. 对项目列表内的项目文件进行锁定操作, 调取文件模型操作类的 Lock() 方法. 前台应用根据界面操作选择锁定后的项目, 调取文件模型操作类的 Move() 迁移方法, 对项目列表内的对象迁移至归档任务库, 如果迁移未成功的文件仍保存在项目资料库, 文档迁移导入模块序列图如图 5 所示.

数据补录的设计思想: 前台应用通过 EJB 层调取文件模型操作类的 ReInput() 方法, 同时利用对象模型类显示对应的操作界面, 并写入到数据库. 本地导入的设计思想: 前台应用通过 EJB 层调取文件模型操作类的 LocalImport() 方法, 本地导入数据直接写入归档任务库. 移交接收的设计思想: 前台应用通过 EJB 层验证用户身份权限, 同时调取归档任务操作类的 Check() 方法, 如果操作成功则返回审批意见信息并写入归档任务库, 等待后续的归档完成操作, 否则返回驳回信息并告知原因.

最后通过使用 UML 建模工具 IBM Rational Architect 绘制系统主要功能模块的时序图和类图, 有助于完成后续系统框架设计和编码工作.

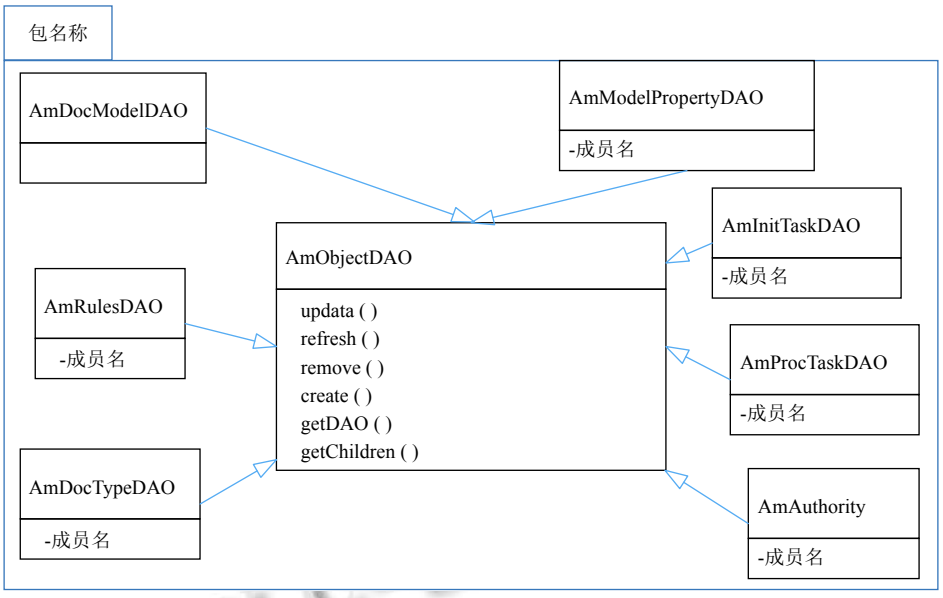

图 4 模型数据操作类中各类间的继承关系

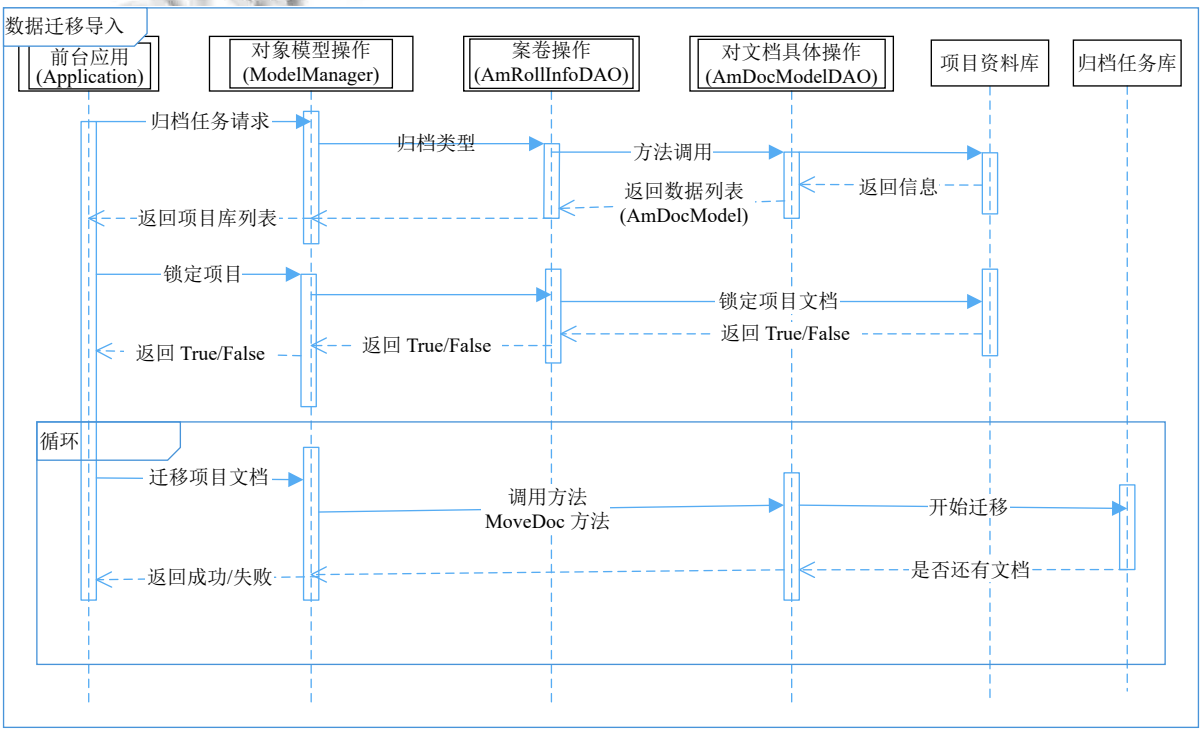

图 5 文档迁移导入时序图

4 系统核心功能实现

4.1 系统框架设计

目前基于 Python 的 Web 开发的框架有很多, 例如: Flask、Django 和 Web2Py 等. Django 如同 Java 的 EJB (Enterprise+JavaBeans+JavaEE 服务器端组件模型) 多被用于大型网站的开发, 而且所有资源每次都要全部加载, 造成一定资源的浪费. 对于大多数的小型网站的开发, Web2Py 的管理接口没有权限, 没有内建的单元测试, 不方便系统的调试. Flask 的优点是保持代码简洁且具有很强的扩展性和兼容性, 通过框架后台的自由配置, 可以使归档管理系统支持表单验证、权限判断、数据库操作等基本功能, 更符合本系统的实

际需求[26,27].

本系统项目归档操作, 元数据提取和审核管理作为核心功能模块, 数据传输/转换接口则作为与一站式采购管理平台及其他信息系统间联系的桥梁, 如图 6 所示. 利用 Flask 框架的核心库 Werkzeug 的 Cookies 和 Session 组件解决多个用户快速响应客户端推送过来的访问请求, 提高用户访问速度. 同时, 系统构建 HTML 页面和数据绑定模式, 使用 knockout.js (一个基于 MVVM 模式的 JavaScript 库), 通过将 UI 和基础 JavaScript 模型绑定, 做到模型和 UI 同步更新. 通过调用 Jinja2 提高系统安全, 将变量名中含 HTML 自动转义, 但如果是安全的变量名则利用 safe 过滤器标记为安全, 这样能够很好控制外部的脚本攻击, 而且也避免全部转义所带来的资源占用.

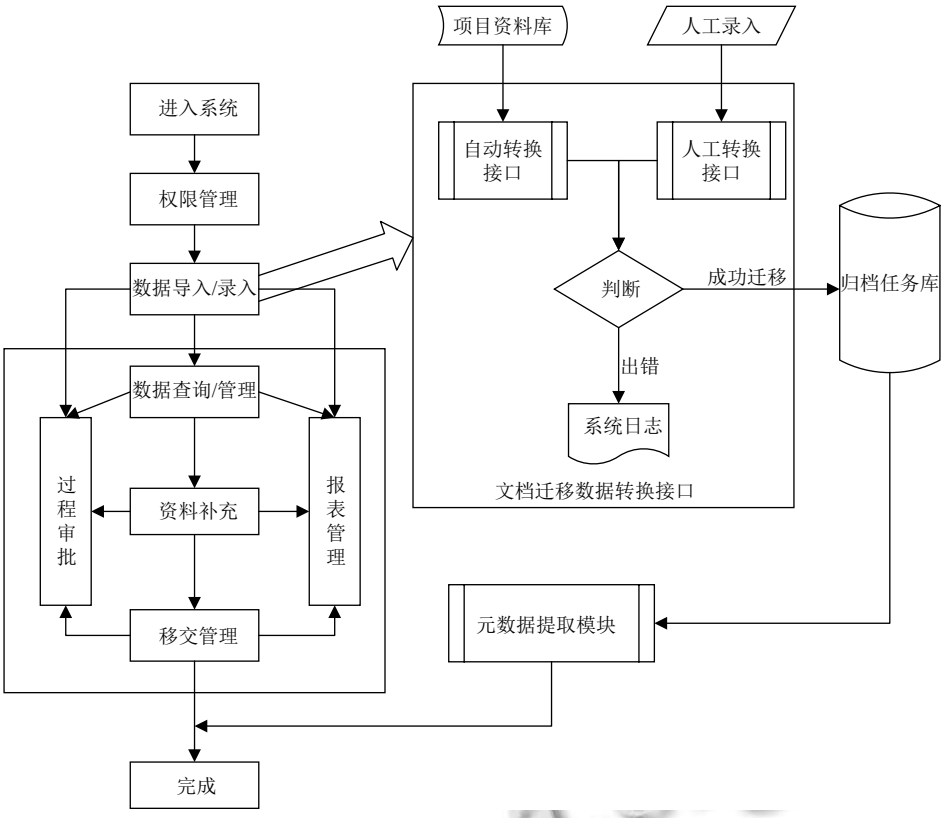

图 6 归档系统架构图

4.2 元数据提取模块设计实现

对 PDF 文档进行数据提取的方法有很多, 比如使用 OCR 文字识别软件对 PDF 文档中的关键信息进行提取[28], 利用 Adobe Acrobat 所提供的接口编写 Plug-in 程序实现对 PDF 元数据提取[29], 或者 Adobe Acrobat X Pro 自带的工具对页面文本进行识别提取[30]. 但这些方法对 PDF 文档的操作太过于繁琐, 后续还要人工对所提取的元数据做进一步处理, 故只适用于处理少量文档的情况, 对于体量庞大的归档文件元数据提取则不合适. 目前比较流行的做法是采取调用已有的 PDF 类库对大量文档进行批量操作, 基于 Java 类库比较常用的两种操作是 iText 和 PDFBox.

iText 是 sourceforge 的一个开源项目, 通过 iText 不仅可以生成 PDF 或 rtf 的文档, 也可以将 XML、HTML 文件转化为 PDF 文件[31], 但 iText 在提取元数据时只能先将 PDF 转换为纯文本后进行文本提取, 这样 PDF 元数据的提取不准确. 而且 itextpdf 类本身并不支持中文, 需要借助第三方 jar (iTextAsian.jar) 来实现, 有部分版本升级后还需要更改中文包的名称和存放路径才能正常使用.

Apache PDFBox 是支持操作 PDF 的开源工具库[32–35]. PDFBox 库提供一个特殊的对象, 该方法涉及 Lucence 的搜索引擎库, LucencePDFDocument.getDocument() 方法将指定 PDF 文档, 提取其内容 (包含作者信息和关键词等元数据) 并创建一个 Lucence 文档对象, 这样添加到 Lucence 索引中的这些元数据方便进行跟踪, 但由于 Lucence 创建的索引只支持文本索引, 而且创

建全文索引也消耗大量资源.

PyPDF2 是基于 Python 开发的函数库, 它提供对 PDF 进行提取元数据和图片、拆分或合并等基本操作, 同时还能编写脚本完成对 PDF 文档的批量操作, PyPDF2 包可在任何 Python 平台上运行, 而且不依赖于其他外部库的配合. 它可以完全在 StringIO 对象而不是文件流上工作, 允许在内存中进行 PDF 操作提高执行效率, 而且新版本 PyPDF4 功能更趋于完善, 相对比较前两种 PDF 元数据提取的方式, 此方法更符合本系统的要求, 综上所述, 因此决定使用基于 Python 的 PyPDF2 来解决 PDF 元数据提取的功能.

首先, 通过 pig install pycharm PyPDF2 安装此模块“PyPDF2”. 然后导入模块“import PyPDF2”和“import sys”, 通过定义一个变量, 将 PDF 文件路径赋值给变量. 调用 open() 用“rb”二进制方式读取文件, 读取的内容传给 PyPDF2.PdfFileReader(), 初始化一个 PdfFileReader 对象. 利用 PdfFileReader 对象的 getDocumentInfo() 方法得到 PDF 文件元数据, 接着利用 for 语句遍历字典的键值对. 此时 docInfo 的实例包含了大部分信息, 可以使用这些属性从文档中获取所需的其余元数据, 将这些数据存放至数据库以备将来使用. 尝试导入单个 PDF 文档验证程序的可行性和元数据提取准确度, 结果如图 7 所示. 最后, 添加 OptionParser 方法使脚本只解析我指定的文件元数据, 同时完善代码将提取到的元数据按一定的格式显示, 部分代码如下所示:

```
def main():
parser = optparse.OptionParser('usage %prog + -F <PDF file name>')
parser.add_option('-F' , dest='filename' , type='string' , help='specify PDF file name') (options, args) = parser.parse_args()
fileName = options.filename
if fileName == None:
print(parser.usage)
exit(0)
else:
printMeta(fileName)
if __name__ == '__main__':
main()
```

本应用通过校园身份统一认证系统验证才可登录如图 8 所示, 登录后根据用户权限显示不同的首页界面, 如图 9 所示为具有文档审核权限的用户.

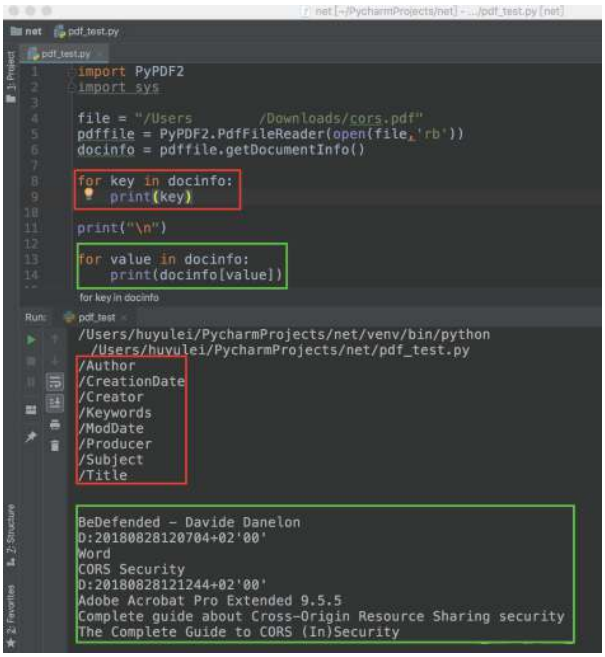

图 7 提取 PDF 元数据

图 8 登陆页面

图 9 档案审批用户首页

5 结束语

针对目前学校电子文档管理和归档管理的现状, 以及整个电子文件生命周期的深入分析, 提出了对现有一站式采购管理平台进一步完善的设计方案. 本系统遵循模型驱动工程的研发过程, 从需求分析、系统设计、系统建模和核心功能逻辑设计本文做了详细介

绍. 此方案为全校提供了一个功能完整, 数据安全, 高效, 顺畅的一体化归档平台, 为解决学校归档业务全部流程信息化的最终目的打下了基础.

下一阶段研究方向, 对系统的几个方面进行改进: (1) 提高 PDF 元数据提取技术的精准度, 目前的元数据的提取不够全面, 不利于资源搜索的效率, 需要进一步优化相关算法. (2) 文档转换要增加 OFD 格式, OFD 标准是我国 2016 年自主研发的版式文件格式, 现在还在大力推广中, 今后很有可能替代 PDF 成为我国电子文档归档标准格式. (3) 进一步提升本系统响应时间、吞吐率、并发用户数等方面的性能.

参考文献

1 张国民. 浅谈文档类电子文件格式及其特点. 兰台世界, 2012, (2): 9–10. [doi: 10.3969/j.issn.1006-7744.2012.02.006]

2 赵庆峰, 鞠英杰. 国内元数据研究综述. 现代情报, 2003, 23(11): 42–45. [doi: 10.3969/j.issn.1008-0821.2003.11.018]

3 王红滨, 刘大昕. 元数据提取综述. 黑龙江省计算机学会 2009 年学术交流年会论文集. 哈尔滨. 2010.4.

4 江亮. 2011–2015 年国内外元数据研究现状和宏观分析. 图书馆杂志, 2016, 35(9): 38–49.

5 龚立群, 马宝英, 常晓荣. 科技文献元数据自动抽取研究述评. 计算机系统应用, 2013, 22(3): 11–15. [doi: 10.3969/j.issn.1003-3254.2013.03.003]

6 李雪驹, 王智广, 鲁强. 一种规则与 SVM 结合的论文抽取方法. 计算机技术与发展, 2017, 27(10): 24–29. [doi: 10.3969/j.issn.1673-629X.2017.10.006]

7 贺亚锋. Web 站点元数据自动生成工具介绍. 图书馆杂志, 2001, 20(1): 28–30. [doi: 10.3969/j.issn.1000-4254.2001.01.010]

8 DC-ROADS: ROADS as a (Dublin Core) metadata management environment. http://www.ukoln.uk/roads/metadata. (1991-05-26).

9 Meta Web Project (MWP). http://www.dstc.edu.au/RDU/MetaWeb. (1991-05-07).

10 王守芳, 狄涤, 潘金贵. 基于自动规约规则的 HTML 文档元数据提取. 模式识别与人工智能, 2005, 18(4): 405–411. [doi: 10.3969/j.issn.1003-6059.2005.04.004]

11 于江德, 樊孝忠, 尹继豪. 基于条件随机场的中文科研论文信息抽取. 华南理工大学学报 (自然科学版), 2007, 35(9): 90–94, 106.

12 张玲, 黄铁军, 高文. 基于隐马尔可夫模型的引文信息提取. 计算机工程, 2003, 29(20): 33–34, 54. [doi: 10.3969/j.issn.1000-3428.2003.20.015]

13 杜秋霞, 王洪国, 邵增珍, 等. 基于混合 HMM 的文献元数据地名抽取方法研究. 计算机与数字工程, 2017, 45(1): 101–106. [doi: 10.3969/j.issn.1672-9722.2017.01.022]

14 徐雪荣, 弓淑芬. 浅析政府采购项目档案管理的信息化发展. 中国管理信息化, 2018, 21(20): 183–184. [doi: 10.3969/j.issn.1673-0194.2018.20.083]

15 于明鹤, 聂铁铮, 李国良. 数据管护技术及应用. 大数据, 2019, 5(6): 2019048.

16 薛四新. 电子档案单轨制管理的关键问题研究. 浙江档案, 2020, (7): 17–20.

17 田士兵. 电子文件归档与管理系统构建. 中国档案, 2017, (2): 64–65.

18 苏冠贤. 办公自动化系统与档案管理系统优化整合模式研究. 档案学研究, 2017, (5): 86–91.

19 吴志杰, 王强. 组织机构视角下的业务系统电子文件归档: 问题、理念与策略框架. 档案学通讯, 2020, (4): 79–86.

20 姜宝, 刘志祥, 彭辉. 广东省政务信息资源安全共享管理研究. 电子产品可靠性与环境试验, 2017, 35(3): 55–59. [doi: 10.3969/j.issn.1672-5468.2017.03.011]

21 Michael McLaughlin. Oracle9i Web 开发指南. 北京: 机械工业出版社, 2003.

22 张雯杰, 蔡佳玲. MongoDB 从入门到商业实战. 北京: 电子工业出版社, 2019.

23 Bryla B. Oracle Database 12c DBA 官方手册. 明道洋, 译. 北京: 清华大学出版社, 2016.

24 伊恩·萨默维尔. 软件工程. 程成, 译. 北京: 机械工业出版社, 2017.

25 拉曼. UML 和模式应用. 李洋, 郑奕, 译. 北京: 机械工业出版社, 2006.

26 Lutz M. Python 学习手册. 李军, 刘红伟, 译. 北京: 电子工业出版社, 2018.

27 李辉. Flask Web 开发实战-入门、进阶与原理解析. 北京: 机械工业出版社, 2018.

28 陈云榕, 刘立柱, 丁志鸿. PDF 文件中关键信息的提取与组织方法研究. 计算机工程与设计, 2007, 28(7): 1688–1690. [doi: 10.3969/j.issn.1000-7024.2007.07.062]

29 李贵林, 李建中, 杨艳. 用 Plug-in 实现对 PDF 文件的信息提取. 计算机应用, 2003, 23(2): 110–112.

30 刘华中. 面向 PDF 文档的论文元数据提取方法研究 [硕士学位论文]. 秦皇岛: 燕山大学, 2012.

31 刘健. 国外元数据研究前沿与热点可视化探讨 [硕士学位论文]. 南京: 南京大学, 2013.

32 牛永洁, 薛苏琴. 基于 PDFBox 抽取学术论文信息的实现. 计算机技术与发展, 2014, 24(12): 61–63, 68.

33 陈文慧. 高校综合电子文档管理系统的设计与实现 [硕士学位论文]. 长沙: 湖南大学, 2018.

34 钱远鹏. 基于 SWT 元数据提取的研究与实现 [硕士学位论文]. 北京: 北京邮电大学, 2018.

35 邹名璐, 罗元. 电子文件归档管理系统的总体设计. 计算技术与自动化, 2018, 37(3): 175–179.