

**研 究 生 毕 业 论 文**

**（申请工程硕士学位）**

|  |  |
| --- | --- |
| **论文题目** | Tradeshift电子发票系统发票生成  相关模块的设计与实现 |
| **作者姓名** | 花 霞 |
| **学科、专业名称** | 工程硕士(软件工程方向) |
| **研究方向** | 软件工程 |
| **指导教师** | 刘峰 讲师 |

**2018年 4月 20日**

**学 号： MF1632024**

**论文答辩日期： 2018 年 5 月 16 日**

**指 导 教 师： （签字）**

**Tradeshift电子发票系统发票生成相关**

**模块的设计与实现**

|  |  |
| --- | --- |
| **作 者:** | **花 霞** |
| **指导教师:** | **刘峰 讲师** |

|  |
| --- |
| **南京大学研究生毕业论文** |
| **(申请工程硕士学位)** |

|  |
| --- |
| **南京大学软件学院** |
| **2018年4月** |

**The Design and Implementation of Invoice Generation Modules in Tradeshift’s Electronic Invoice System**

**Hua, Xia**

**Submitted in partial fulfillment of the requirements for the degree of Master of Engineering**

Supervised by

Lecturer **Liu, Feng**

Software Institute

**NANJING UNIVERSITY**

Nanjing, China

April, 2018

# 摘 要

发票是生活消费以及商家在交易过程中的重要交易凭证。传统纸质发票由机打或手写的方式生成，消耗着巨大的社会成本且不便于汇总、保存等。电子发票通过无纸化的形式来实现交易信息的传递和保存以减少管理成本。电子发票不仅可以帮助企业管理业务网络中的交易和资金流信息，而且是企业信用的重要凭证。Tradeshift公司为企业提供了电子发票的解决方案，把采购商和供应商连接在同一个网络平台中，为用户提供从采购到开票的电子化流程。

本文主要描述了Tradeshift电子发票系统的相关设计与实现方式。Tradeshift电子发票系统从功能上被划分为公司信息模块、邮件模块、公司网络模块、对账单模块、形式发票模块、价格协商模块、预制发票模块、直连开票模块、应用库模块。本文首先描述了电子发票相关的行业背景和Tradeshift电子发票系统的项目背景；然后介绍了Tradeshift电子发票系统所使用的数据存储技术和系统开发技术；接着描述了电子发票系统的需求和发票生成相关模块的分析与设计，并对数据库设计进行了介绍；最后，本文描述了电子发票系统发票生成相关模块的实现方式和实现结果。

本文主要描述了电子发票生成的相关模块，其中对账单模块主要处理采购商上传的元数据的解析和存储；形式发票模块和预制发票模块实现在开具正式发票前，为买卖双方创建形式发票和预制发票；价格协商模块为开票商品提供价格协商和确认的功能。

**关键词：**电子发票、对账单、形式发票、预制发票、微服务

# Abstract

Invoice plays an important role as a transaction certificate for life consumption and business transactions. Traditional paper invoices are produced by machine or handwritten, which consumes huge social cost and cannot meet the demand of summary and preservation. Electronic invoices realize information transmission and storage in a paperless way, reducing management costs. Electronic invoices would not only help companies manage transaction in their business networks, but they are also important credentials for corporate credit. Tradeshift provides solutions for enterprise billing by connecting both buyers and suppliers to the same network platform.

This thesis mainly describes the design and implementation of Tradeshift electronic invoice system. The electronic invoice system is divided into company information module, mail module, company network module, billing module, proforma invoice module, bargaining module, premade invoice module, directly invoicing module and application library module. This thesis first describes the background of the Tradeshift electronic invoice system, and then introduces the data storage technology and the system development technology used in the Tradeshift electronic invoice system. Then it describes the requirements of the Tradeshift electronic invoice system and the analysis and design of the relevant modules of the invoice generation, and introduces the database design in detail. Finally, this thesis describes the implementation and results of the relevant modules of the invoice generation of the electronic invoice system.

This thesis mainly describes the related modules of invoice generation. The billing module mainly deals with the parsing and storage of metadata uploaded by buyers. Proforma invoice module and premade invoice module can be used to create proforma invoices and prefabricated invoices. The bargaining module provides a process for negotiating and confirming the price of a proforma invoice.

**Keywords**: electronic invoice, bills, proforma invoices, premade invoices, micro service

# 目录

摘 要 I

Abstract II

目录 III

图目录 VI

表目录 VIII

第一章 引言 1

1.1 行业背景 1

1.2 项目背景 2

1.3 本文的主要工作 3

1.4 本文组织结构 4

第二章 技术综述 5

2.1 数据存储技术 5

2.1.1 Elasticsearch 5

2.1.2 Riak 6

2.1.3 MyBatis 8

2.2 系统开发技术 9

2.2.1 Microservice 9

2.2.2 RESTful 10

2.2.3 OAuth2.0 11

2.2.4 HornetQ 12

2.2.5 SpringMVC 13

2.3 本章小结 14

第三章 电子发票系统的分析与设计 15

3.1 项目概述 15

3.2 系统需求分析 16

3.2.1 系统用例图 16

3.2.2 功能需求 17

3.3 系统总体结构与模块设计 22

3.3.1 总体结构 22

3.3.2 对账单模块 23

3.3.3 形式发票模块 24

3.3.4 价格协商模块 27

3.3.5 预制发票模块 28

3.4 数据库设计 30

3.4.1 数据库总体设计 30

3.4.2 proformaline表设计 32

3.4.3 mockinvoiceline表设计 33

3.4.4 fapiao表设计 33

3.5 本章小结 34

第四章 电子发票系统主要功能的实现 35

4.1 电子发票系统的主要功能概述 35

4.2 电子发票系统的整体设计 36

4.2.1 电子发票系统的整体框架 36

4.2.2 电子发票生成相关模块的类图 37

4.3 对账单模块 38

4.3.1 对账单模块设计描述 38

4.3.2 对账单模块实现代码 39

4.3.3 对账单模块实现结果 40

4.4 形式发票模块 41

4.4.1 形式发票模块设计描述 41

4.4.2 形式发票模块实现代码 42

4.4.3 形式发票模块实现结果 43

4.5 价格协商模块 44

4.5.1 价格协商模块设计描述 44

4.5.2 价格协商模块实现代码 46

4.5.3 价格协商模块实现结果 46

4.6 预制发票模块 47

4.6.1 预制发票模块设计描述 47

4.6.2 预制发票模块实现代码 48

4.6.3 预制发票模块实现结果 50

4.7 本章小结 52

第五章 总结与展望 53

5.1 总结 53

5.2 进一步展望 54

参考文献 55

致谢 59

版权及论文原创性说明 60

# 图目录

图2.1 Elasticsearch单个文档读取流程示意图 6

图2.2 Elasticsearch集群扩张逻辑示意图 6

图2.3 Riak集群关系图 7

图2.4 MyBatis应用流程图 8

图2.5 微服务拆分功能组件图 9

图2.6 REST架构图 10

图2.7 OAuth2.0运行流程图 11

图2.8 点对点消息队列模型图 12

图2.9 springmvc执行流程 14

图3.1 Tradeshift电子发票系统 15

图3.2 Tradeshift电子发票系统系统用例图 16

图3.3 Tradeshift电子发票系统总体结构图 22

图3.4 发票生成相关模块数据流转图 22

图3.5 上传对账单活动图 24

图3.6 创建形式发票活动图 25

图3.7 对账单商品信息相关字段 26

图3.8 价格协商活动图 27

图3.9 拆票活动图 29

图3.10 Tradeshift电子发票系统发票生成模块数据库设计图 30

图4.1 Tradeshift发票系统组织结构 36

图4.2 电子发票生成相关模块类图 37

图4.3 上传对账单时序图 38

图4.4 V4 PROFORMA监听接收对账单事件 39

图4.5 对账单列表界面 40

图4.6 生成形式发票时序图 41

图4.7 创建形式发票接口 42

图4.8 对账单文档创建形式发票界面 43

图4.9 勾选创建形式发票界面 44

图4.10 价格协商时序图 45

图4.11 保存议价记录代码 46

图4.12 价格协商图 47

图4.13 拆票时序图 48

图4.14 拆分发票实现 49

图4.15 合并发票算法实现 50

图4.16 设置开票限额界面 51

图4.17 预制发票列表界面 51

图4.18 预制发票详情界面 52

# 表目录

表3.1建立公司网络用例描述 18

表3.2 上传对账单用例描述 18

表3.3 创建形式发票用例描述 19

表3.4 编辑形式发票用例描述 19

表3.5 议价用例描述 20

表3.6 生成预制发票用例描述 20

表3.7生成增值税专用发票用例描述 21

表3.8发票录入用例描述 21

表3.9 proformaline表设计 32

表3.10 mockinvoiceline表设计 33

表3.11 fapiao表设计 34

# 第一章 引言

## 1.1 行业背景

发票是个人在生活消费以及商家在交易过程中所开具和收取的业务凭证，它起着报销、记账、核税、保修和维权等非常重要的作用。发票不仅是会计财务核算的原始依据，也成为税务对纳税人进行检查征收的重要依据 [张媛媛, 2010]。发票使用范围非常广泛，在各种日常生活场景中都扮演着重要的角色，然而纸质发票保管困难，数据追溯难度大，纸质发票的大量印制和购置持续消耗着庞大的社会成本。

为了解决纸质发票带来的多种问题，电子发票逐渐得到了推广。自2011年6月《中国电子发票蓝皮书》公布以来，电子发票的推广就拉开了序幕[潘陶, 2016]。近年来，国家逐年出台多项政策，从多个维度和方面帮助电子发票在全国的推广和使用[戴新竹等, 2016]。自2015年以来，国家对电子发票在政策层面的扶持进一步加速。2015年，财政部在最新的《会计档案管理办法》中允许将电子发票作为报销凭证，并且首次把电子档案纳入了会计档案范围 [新华社, 2015]，有效地解决了电子发票在实际使用过程中遇到的多种问题，有利于推动电子发票的不断发展。

电子发票和传统纸质发票相比有着很多优点。电子发票系统搭建完成后，电子发票通过无纸化的方式降低开票成本。同时，采购商和供应商能够通过线上电子平台直接进行联系和交互从而减少了许多繁杂的线下操作，这样更能够满足于高速发展的电子商务的需求，提高业务办理的效率，保证跨地域服务的实时性。线上的电子化数据的存储能够帮助税务监控系统获取真实的、实时的交易数据，有利于交易信息的追溯，利于税务相关数据的收集和分析。另外，电子发票数据无法轻易修改和复制，有很好的防伪功能，与传统纸质发票相比，电子发票将对税务监管有着明显的促进作用[汪周丹, 2016]。

## 1.2 项目背景

Tradeshift是一个为企业级用户提供数字化服务的平台，一个将全球采购商和供应商联系在一起的全球在线市场[Tradeshift, 2018]。Tradeshift为各类企业提供从购买商品到支付、从完成交易到开出交易凭证的整套解决方案。同时，Tradeshift也提供供应商管理服务和金融服务，让企业及其商业伙伴不仅可以在Tradeshift商业平台上获得基本的商业应用服务，而且企业也可以定制许多满足特殊需求的增值服务。Tradeshift主要为企业级用户提供多种电子化服务，帮助企业实现数字化的交易和协作。

Tradeshift电子发票系统主要为企业级用户提供上下游的协同服务，帮助不同类型、不同规模的企业实现与上游供应商，下游经销商之间的信息传递的无纸化，保证跨地域服务的实时性，真正赋予企业采购生产、商品交付、收付款和开票等各个环节的数字化、无纸化的体验。Tradeshift电子发票系统将采购商与供应商通过电子化的方式连接起来，让采购商与供应商更加高效便捷的交流和协作，提高买卖双方交互的便捷性和实时性。

Tradeshift平台为企业级用户提供了一个从购买产品到开出增值税发票的标准、可靠的流程，方便用户对采购、销售、开票流程进行统一准确的管理。Tradeshift电子发票系统提供的总体的开票流程为：企业开票方即供应商把交易信息的电子化数据上传至电子发票平台；电子发票平台接收电子数据，根据待开票数据开具电子发票，通过线上无纸化的方式将电子发票发送给企业受票方即采购商，并在系统中保存交易和发票数据；外围系统税务局能够通过电子发票平台获取和分析交易信息，同时也能够通过电子发票系统获得海量电子交易数据进行分析，进行税务监控。

此外，Tradeshift电子发票系统功能是可定制化的。电子发票系统从功能上被划分为公司信息模块、邮件模块、公司网络模块、对账单模块、形式发票模块、价格协商模块、预制发票模块、直连开票模块、应用库模块。用户可以在基本功能的基础上通过Tradeshift 应用库中的应用来扩展和电子发票相关的其他功能，来满足其更复杂的业务需求。

## 1.3 本文的主要工作

本文所研究的Tradeshift电子发票系统，是一个把采购商和供应商连接起来的电子化平台，能够为客户提供从采购到付款、从交易到开出交易凭证的整套解决方案。电子发票系统以云平台的形式存在，所有的采购商和供应商全部都在一个公网网络中，这种网络既可以是一个由采购商为主导的私有采购平台，又可以转化成一个公开的任何人都可以加入的采购平台[罗东, 2010]。Tradeshift电子发票系统不仅可以为在平台上的采购记录开出增值税电子发票，同时也为采购商提供了上传对账单的接口，可以直接对采购商上传的采购文档进行解析并创建形式发票、预制发票，增值税电子发票。

电子发票系统从功能上被划分为公司信息模块、邮件模块、公司网络模块、对账单模块、形式发票模块、议价模块、预制发票模块、直连开票模块、应用库模块。本文主要针对Tradeshift电子发票系统发票生成相关模块进行描述。包括对采购商上传原始数据的处理，形式发票创建，价格协商，预制发票创建等多个功能模块进行了详细描述。本文首先描述了电子发票相关的行业背景，对纸质发票存在的问题以及电子发票的推广和发展过程进行了简单介绍，并介绍了Tradeshift公司以及Tradeshift电子发票系统的项目背景和项目概述。然后介绍了Tradeshift电子发票系统所使用的数据存储技术和系统开发技术。接着描述了电子发票系统的功能需求和发票生成相关模块的分析与详细设计，并对数据库的表设计进行了介绍。最后，本文描述了电子发票系统发票生成相关模块的实现方式和实现结果。

电子发票系统基于微服务的方式，采用springMVC模式的设计思想，在系统的数据展示层应用多线程机制和消息队列异步技术等提高了请求的响应速度。采用非关系型的数据库、全文本搜索引擎来提高数据的存储和查询效率。在数据查询和操作方面，采用Mybatis持久层框架来建立Java对象和数据库记录之间的映射关系。当用户对系统数据进行访问和获取的过程中，系统采用OAuth2.0标准协议来验证用户身份以及获取用户授权，以此来进行访问控制，并对数据进行加密，来提高系统的安全性[魏成坤等, 2016]。

## 1.4 本文组织结构

本文主要基于Tradeshift电子发票系统进行展开，并对电子发票生成相关模块进行了重点描述。本文首先介绍电子发票系统相关的行业背景和项目背景，介绍了电子发票平台在开票过程中使用到的多种技术。然后对Tradeshift电子发票系统的功能模块进行了详细的介绍，并重点描述了发票生成相关模块业务流程、设计和实现。

第一章为引言部分。主要介绍了电子发票的行业背景、项目背景、论文的主要工作以及组织结构。

第二章为技术综述。主要介绍在电子发票系统中所使用到的技术和技术应用方式。包括了Elasticsearch、Riak、MyBatis、Microservice、RESTful、HornetQ、SpringMVC。

第三章为Tradeshift电子发票系统的分析与设计。首先展示了电子发票系统的用例图，然后对关键功能需求进行详细描述，并对发票生成相关模块的业务流程和设计思路按照功能模块进行了概述，对系统的数据库设计进行了介绍。

第四章为电子发票系统的发票生成相关模块的设计和实现。在描述需求和业务流程的基础上，详细阐述了模块的设计和实现，描述各个微服务间的交互过程。

第五章为总结与展望。对Tradeshift电子发票系统实现过程以及在论文期间所做的工作进行总结，提出项目中可以进一步完善的部分，对项目的未来进行展望。

# 第二章 技术综述

## 2.1 数据存储技术

### 2.1.1 Elasticsearch

Tradeshift电子发票系统根据客户的不同需求提供不同的定制服务。随着平台的快速发展，加入平台的供应商和采购商逐渐增加，发票等文档数据会以几何级别的数量提升。为了保证在高数量级数据上查询操作的实时性，系统使用Elasticsearch来完成搜索服务[Lei et al., 2015]。

ElasticSearch是一个带有实时分析功能的分布式搜索引擎，它不仅提供全文本检索功能，还提供分布式实时文档存储功能[周映等, 2015]。ElasticSearch可以通过扩展服务器的数量来存储和处理PB级的数据。Lucene是开源的引擎工具包，提供全文搜索功能，Elasticsearch基于Lucene对接口进行封装后提供高效率的检索方式。

Elasticsearch能够自动重新分片，采用多分片的方式保证数据的安全性。当有海量数据需要处理时，由于内存的限制、磁盘处理能力的不足，无法快速地响应客户端的请求，Elasticsearch会将数据切分为较小的分片进行处理。在ElasticSearch中分片是一个底层的工作单元，每个分片能够被独立地存储在多个不同的服务器上。当用户查询的数据对应的索引存储在多个分片上时，ElasticSearch会将查询内容发送给这些有关联的分片，并将查询结果组合处理后返回[宏朴, 2016]。

ElasticSearch中每个节点都存有其他所有文档的位置信息，当用户需要检索数据时，用户的查询请求会发送到集群的协调节点，协调节点转发这些请求到其他节点并收集数据返回给客户端。读取单个文档流程如图2.1 所示，客户端向NODE1发起请求，NODE1会成为协调节点，向存储请求文档的节点NODE2发起请求，NODE2将请求结果返回给NODE1后，NODE1最后将结果返回给客户端。

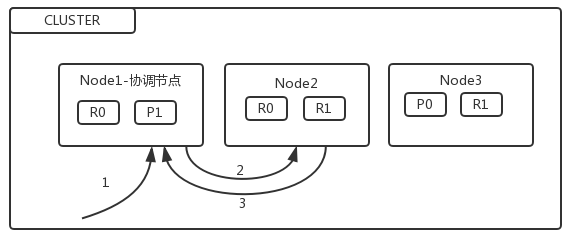


图2.1 Elasticsearch单个文档读取流程示意图

当数据以几何数增加时，需要对集群进行扩张，按集群节点来均衡分配分片，从而对索引和检索过程进行负载均衡，具体过程如图2.2所示。

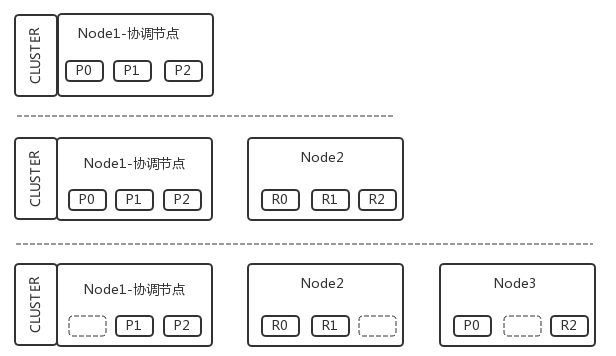


图2.2 Elasticsearch集群扩张逻辑示意图

### 2.1.2 Riak

随着互联网的快速发展，传统的关系型数据库会出现性能瓶颈问题，影响系统的响应和处理速度等。而使用非关系型数据库可以很好地应对关系型数据库带来的很多问题，能够解决不同数据类型的海量数据的存储和处理问题[吕冬雪, 2016]。Riak是一种非关系型数据库，它没有主节点的概念，因此在解决故障方面具有很好的伸缩性和可扩展性[Jain et al., 2015]。

Riak是一个开放源代码的分布式 key-value数据库，在性能方面表现优异，能够通过读写多个服务器来保证当网络出现问题或者机器宕机时服务依然可用。Riak按照一致性哈希的思想来实现存储数据的复制及分片[苏跃明等, 2017]。一致性哈希算法可以使在删除或增加一个节点时，最低限度地修改已有映射关系，实现在动态的网络拓扑中进行分布存储和路由。Riak基于一致性哈希算法实现了负载均衡，当有节点从哈希环删除后，被删除节点上的数据在经过计算后被重新分配到整个环上的其它节点上。

在Riak中，每个服务器节点的地位都是平等的，在去中心化的集群中能够随意地进行增加和移除节点的操作并据此将数据重新分摊到各个节点上，这能够让 Riak 更加方便灵活地进行故障转移和扩展，避免单点故障[SIDDIQA et al., 2017]。Riak通过将数据均匀分布在集群内的每一个节点上，保证当机器增加时，系统性能能够近似线性增长。客户端对Riak集群进行访问和实际的Riak集群之间的关系如图2.3所示。每一个Riak节点都与一个实际的Riak集群中的节点相对应。

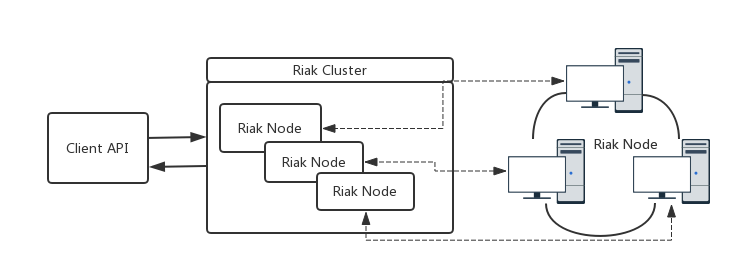


图2.3 Riak集群关系图

Riak对于数据存储没有确定的格式要求，可以存储大小不同、类型不同的各种文档数据。在电子发票系统中，使用Riak存储用户上传的对账单。系统支持不同采购商上传不同类型不同大小的对账单。

### 2.1.3 MyBatis

MyBatis是一个支持SQL定制化功能，支持高级映射和存储过程的持久层框架。MyBatis允许开发人员通过注解或简单的XML就可以实现对数据访问的配置，将程序中的Java对象映射为数据库中存储的记录[MyBatis, 2018]。

MyBatis从架构层面可以被分为基础层、数据处理层和API接口层这几个部分[荣艳冬, 2015]。

(1)基础层：基础层是数据处理层的支撑，主要提供加载配置文件、管理数据库连接、处理事务的缓存和管理等多种功能。

(2)数据处理层：主要用于对数据进行处理和执行SQL语句，实现Java实体和数据库存储的记录之间的映射关系和数据的增删改查操作。

(3)API 接口层：提供了许多封装好的API，为开发人员提供了更为高效简单的方式对数据库进行访问和操作，提高开发效率。

MyBatis主要基于XML配置实现Java实体和数据库记录之间的映射关系[文欢欢等, 2015]，MyBatis的配置应用流程非常方便简洁，如图2.4所示，开发人员编写主配置文件，然后根据配置文件创建session，根据配置文件实现Java实体和数据库存储记录间的映射关系，并对数据库进行访问，最后当数据访问结束后，关闭session。

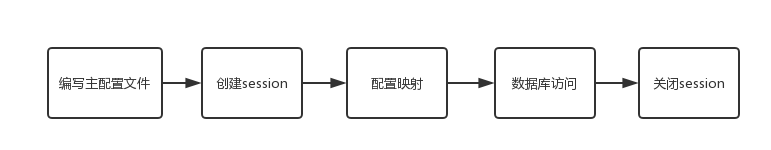


图2.4 MyBatis应用流程图

MyBatis是免费的开源软件，是一个不依赖于第三方包的轻量级框架，学习成本低。MyBatis支持动态SQL，动态语句是其最强大的功能之一，开发人员能够自由便捷地使用SQL对数据进行查询等操作，并根据实际需求封装成定制化的持久层 [丁春玲等, 2017]。

## 2.2 系统开发技术

### 2.2.1 Microservice

Microservice（微服务架构）是将业务系统进行组件化和服务化开发，将一个完整的业务系统分解成数个能够独立设计、开发、运行和维护的小组件。这些组件之间通过给定的规则协议来实现协同工作，完成组件间的交互和集成[张晶等, 2017]。拆分出来的每个组件都能够作为一个独立的应用，每个组件的内部实现方式互不影响。每个独立的应用不仅实现自身需要完成的业务需求，将自身所提供的功能向外发布为服务外，还需要按照给定的规则协议来使用其它外部应用提供的服务。

微服务架构将系统按照业务功能进行划分，应用程序在设计阶段就可以根据业务功能或业务流程分割成多个可以独立运行的服务。然后再应用相同的规则将系统所需要的服务拼装起来，形成一个完整可用的系统[李春阳等, 2017]。如图2.5所示，在微服务架构中，不同的功能组件对应于不同的服务，只需要对该业务功能的服务进行扩展就可以实现对特定业务的扩展，从而避免对整个应用程序进行修改，提高了整个系统的可扩展性。同时，以业务功能为导向的微服务架构使管理员可以通过当前的资源状况对运算资源进行重新分配和部署。

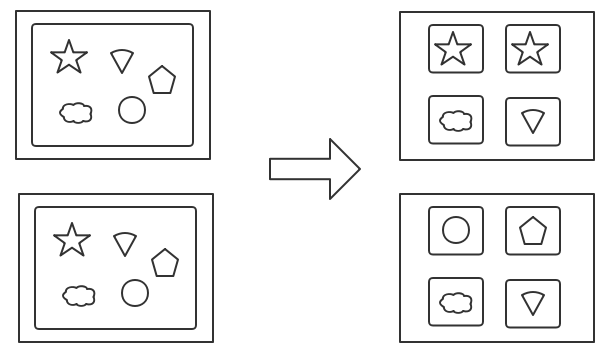


图2.5 微服务拆分功能组件图

### 2.2.2 RESTful

经过十多年的发展，REST（Representational State Transfer，表现层状态转化）成为了Web开发技术的重要组成部分。REST主要通过统一规则的接口和无状态的URI来实现对资源的访问。

RESTful 是符合REST约束和规则的架构设计风格，它聚焦于资源和资源的访问方式 [Garriga et al., 2016]。RESTful架构风格规定，请求数据的增删查改操作对应于HTTP中的GET、POST、PUT和DELETE方法。通过RESTful统一的架构设计风格定义了数据访问和请求的接口，GET方法用来获取资源，POST方法用来新建或更新资源，PUT方法用来更新资源，DELETE方法用来删除资源。

在RESTful架构风格中，统一资源定位符（URI）是每一个资源的地址或识别符，通过访问URI就可以对资源进行定位和获取。REST具有无状态的特点，所要访问的资源都根据URI来进行定位，而不受其他资源影响，不会随着其他资源的改变发生改变 [王建斌等, 2010]。REST的无状态的特性使得系统在调用接口来获取或操作资源的时候，能够不用关注上下文环境和当前所处状态，极大的降低了请求的复杂度。

如图2.6所示，为REST架构图，客户端（Client）通过RESTful API向服务器端（Server）发送请求，获取资源[王仲洲等, 2016]。

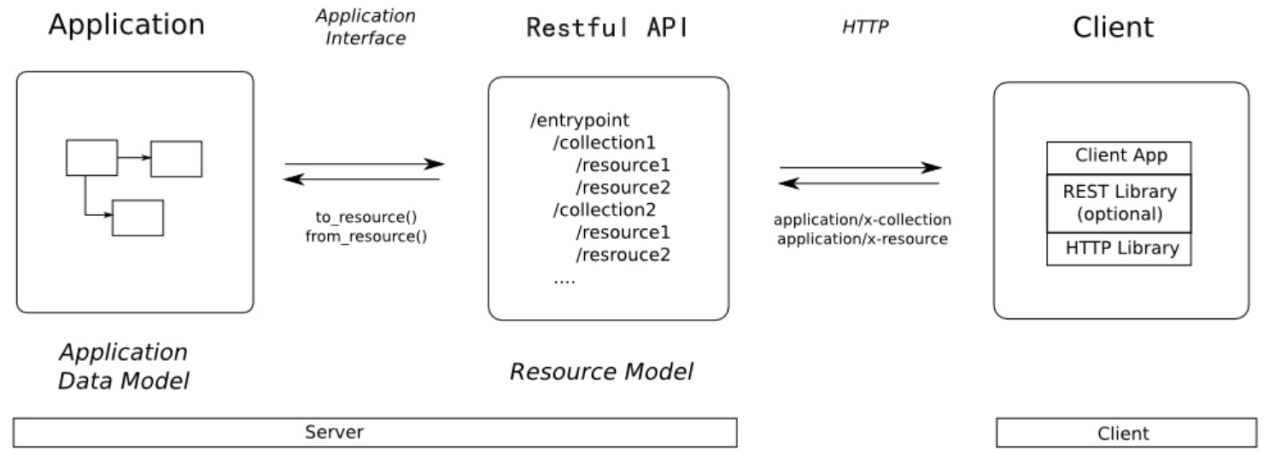


图2.6 REST架构图

### 2.2.3 OAuth2.0

OAuth2.0是一个开放标准协议，允许用户在不提供账号密码的情况下授权第三方网站获取用户在其他服务提供者上维护的部分可共享数据[XIAO et al., 2017]。

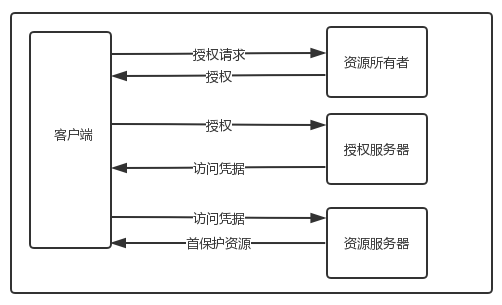


图2.7 OAuth2.0运行流程图

在OAuth中至少存在用户、应用服务器和认证服务器这三种角色。OAuth的主要作用就是在认证服务器的限制下让客户端能够安全、正确地获得用户的授权，以此来访问应用服务器上的资源。

OAuth2.0的运行流程图如图2.7所示。OAuth2.0一共提供了四种不同的授权方式[刘姚, 2017]。

授权码授权：授权码授权能够使用户同时获得访问凭据和刷新凭据。基于HTTP重定向的方式要求客户端可以操作资源所有者的用户代理并且可以获取从授权服务器重定向发送的请求。

隐式授权：隐式授权针对已知的公开客户端优化，只可以获得访问凭据，不允许刷新凭据。与授权码授权相同，基于HTTP重定向的方式要求客户端可以操纵资源拥有者的用户代理并且可以接收从授权服务器重定向发送的请求。它与授权码授权方式主要区别在于客户端可以直接从授权请求获取访问凭据而不用为授权和访问凭据分别单独发送请求。

资源所有者密码凭据授权：资源所有者密码凭据授权可以被运用于被完全信任的应用，例如可以用于一些通过可交互的表单获取用户凭证的应用。

客户端凭据授权：客户端凭据授权只针对于完全受信任的客户端，是指客户端能够只根据客户端自身凭证来获得访问凭证，客户端能够按照自身的需求来获取受限数据。

### 2.2.4 HornetQ

消息队列中间件运用于异步处理、应用耦合、流量削锋等多个场景，是大型分布式系统中不可或缺的重要中间件[于吉鹏, 2014]。在高并发环境下，大量的同步请求处理可能会导致堵塞问题。例如当数据库需要执行大量的插入、更新等操作时，容易造成大量的行锁表锁，最后将可能导致请求过多而发生阻塞从而触发连接过多的错误。为了解决上述类似阻塞问题，可以通过使用消息队列来实现请求的异步处理以缓解和分散系统的压力。

如图2.8为点对点消息队列模型图，异步的操作允许消息的生产者和消费者可以处于不用的状态，生产者只需要在消息产生时将请求消息发送给特定的队列，当消息消费者空闲时就会从队列中接受消息处理并确认。

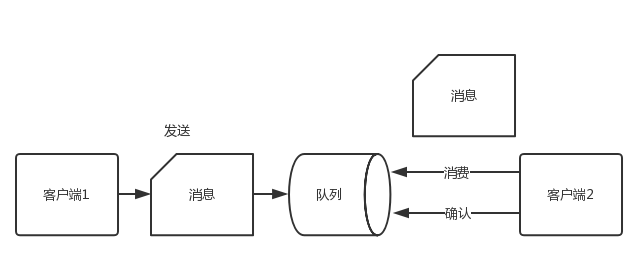


图2.8 点对点消息队列模型图

HornetQ提供消息传递和消息队列模型，是一种通过消息传递机制以实现与平台无关的数据传递的消息服务中间件，以数据通信为基础来实现分布式系统的集成。在开发过程中，HornetQ完全使用简单的Java对象从而可以尽量不依赖于第三方包，能够更加方便独立地被使用。HornetQ对Java消息服务有很好的支持，并且支持集群和多个协议，是一个灵活、高可用、可伸缩的异步消息系统。一张形式发票可能包含数千行的数据，在供应商向对应的采购商发送形式发票时，通过异步消息系统能够有效避免阻塞情况，给用户实时性的响应结果，提升用户体验。

### 2.2.5 SpringMVC

Spring是一个轻量级的的容器框架，它主要的功能特性是控制反转和面向切面。Spring的控制反转能够让容器对对象进行管理，使开发人员在开发过程中，不需要自己主动创建或查询依赖的对象，而是通过容器控制的方法传入[Spring, 2018]。Spring的面向切面编程的支持使程序在运行时能够动态地将代码加载到指定的位置，实现对多个业务逻辑部分的解耦。

Spring不仅能够在应用中单独使用，也能够和Hibernate等ORM框架，和Struts等MVC框架实现集成后使用。SpringMVC是一个轻量级的Web框架，运用了MVC的设计模式，将Spring与MVC框架进行集成，使SpringMVC拥有非常灵活简便的配置功能，具有可适配、非侵入的特性，通过简单的JavaBean就可以实现对Java类的配置。SpringMVC架构将数据展示、业务处理和流程控制分离，实现了对Web层的职责的解耦 [薛峰等, 2012]。

SpringMVC由验证器、控制器、模型对象、命令对象、处理程序映射视图解析器等数个部分组成，每个组成部分的功能实现都由一个特定的对象来执行和实现。如图2.9所示，DispatcherServlet 是SpringMVC的集中访问点，主要扮演着职责调度和任务分配的角色，实现了视图解析、语言环境处理、主题解析等多种功能。

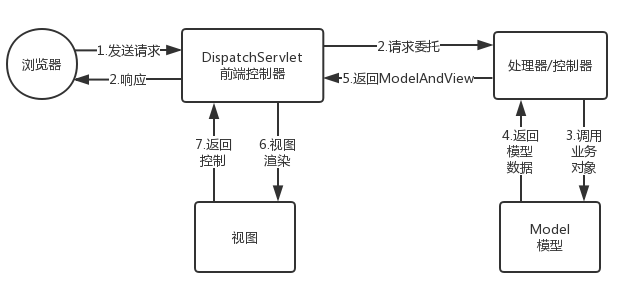


图2.9 springmvc执行流程

## 2.3 本章小结

本章主要介绍了Tradeshift电子发票系统在实现过程中主要涉及到的相关技术。电子发票系统使用了Riak、Elasticsearch、MyBatis来实现部分数据存储和查询，提高数据操作效率。电子发票系统基于微服务来实现，使系统的升级和扩展更为简便。电子发票系统接口的设计满足REST约束条件和原则，通过统一规则的接口和无状态的URI来实现对资源的访问。通过OAuth2.0的授权认证，来保证资源访问的权限。在系统消息传递过程中，使用到了HornetQ，通过消息队列中间件进行数据通信。在各个服务的开发中，使用了springMVC框架，将数据展示层、业务逻辑层和控制层相分离，简化了开发过程。

# 第三章 电子发票系统的分析与设计

## 3.1 项目概述

为了克服传统纸质发票的缺陷，Tradeshift公司开发了电子发票系统。Tradeshift为企业提供从商品购买到支付、从完成交易到开出交易凭证的一整套的解决方案。电子发票具有统一规定的格式，相比传统纸质发票，电子发票在页面左上角新增了二维码标识，用户能够使用普通A4纸而不需要用专用纸打印发票。Tradeshift电子发票平台为企业级的用户提供了上下游协同服务，帮助不同类型、不同规模的企业级用户实现与供应商和经销商之间的信息传递，提高买卖双方交互的便捷性和实时性，实现企业从采购到开票等多个环节的数字化、无纸化流程。

Tradeshift电子发票系统主要为用户提供从交易记录到电子化发票的整套解决方案。电子发票系统由公司信息模块、邮件模块、公司网络模块、对账单模块、形式发票模块、价格协商模块、预制发票模块、直连开票模块、应用库模块组成。其中，核心功能主要包括为采购商提供上传采购记录的功能，为供应商提供创建形式发票，创建预制发票，直连开票的功能。通过电子化的数据，方便用户对发票进行查看和管理。

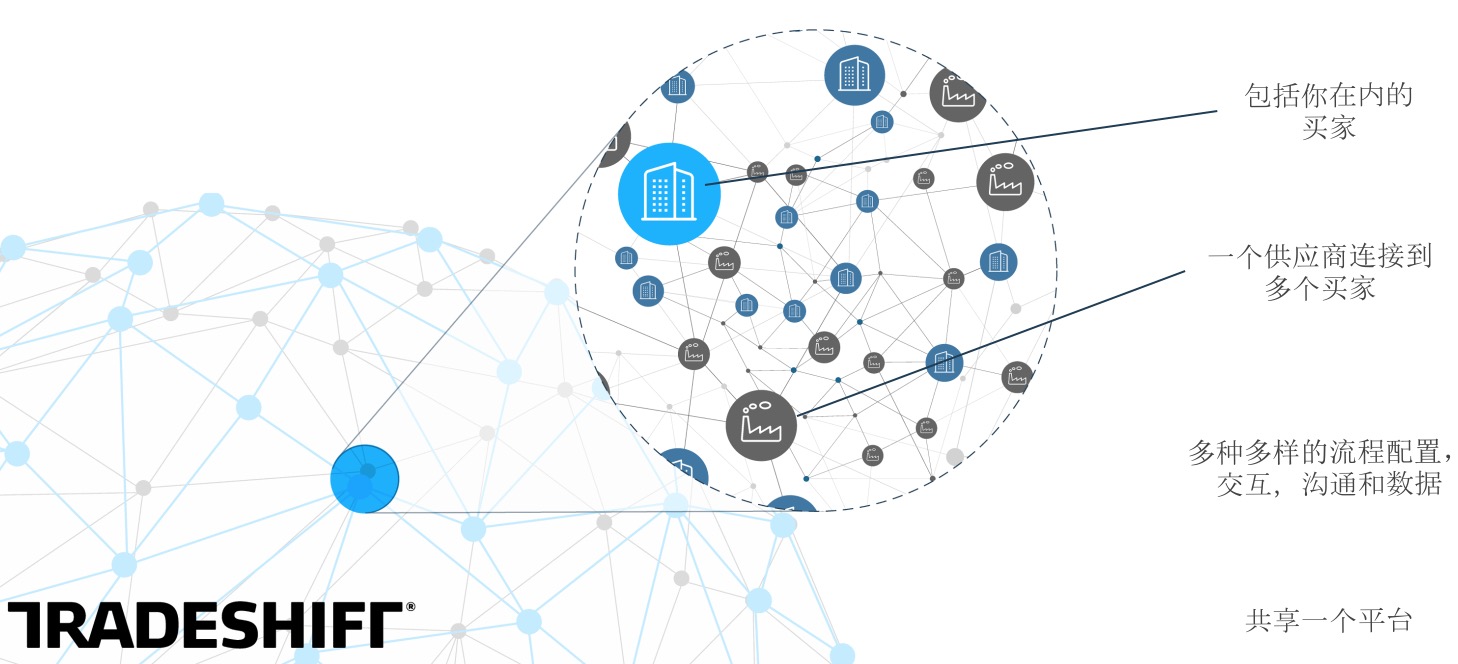


图3.1 Tradeshift电子发票系统

## 3.2 系统需求分析

### 3.2.1 系统用例图

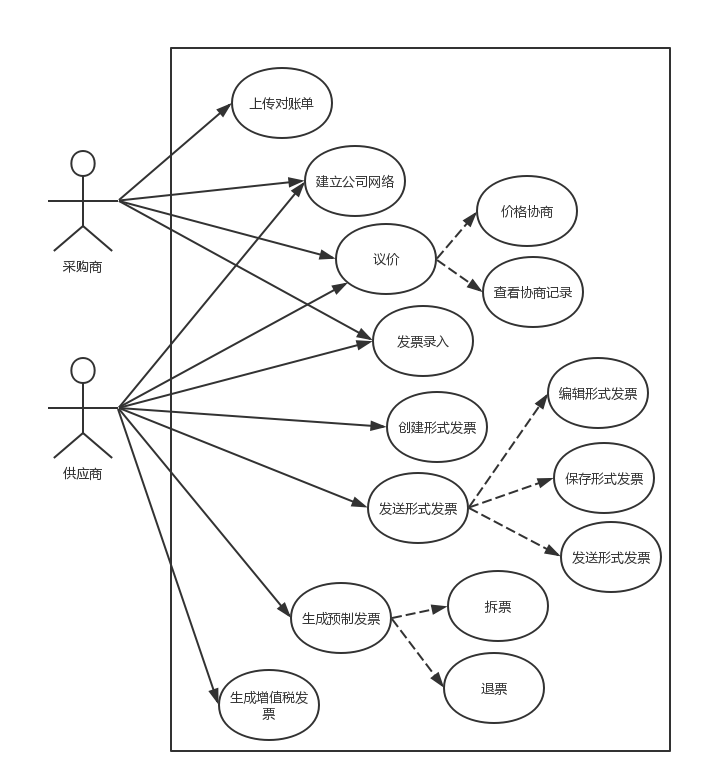


图3.2 Tradeshift电子发票系统系统用例图

根据需求分析，Tradeshift电子发票系统系统用例图如图3.2所示，主要分为采购商和供应商两种角色。采购商主要负责开票元数据的整理和发票的录入，供应商主要针对采购商上传的待开票数据生成增值税专用发票。用户在平台上注册，并填写包括企业名称，企业申请的税号，地址等必要信息。企业级用户可以通过搜索对方的公司名称查询公司并请求建立关联。为了保证用户数据的安全性，采购商和供应商之间需要建立连接，即加入相同的公司网络，才能进行交互。当双方建立连接之后，采购商可以通过将采购记录整理成符合规范的对账单，并将对账单发送给供应商。供应商接收对账单，进行创建形式发票、编辑形式发票、保存形式发票、发送形式发票的操作。采购商收到形式发票后，采购双方会对形式发票中每一行记录的商品价格进行协商，双方就一个形式发票可以进行多次协商，直至双方都接受形式发票中的所有行价格。在协商过程中，系统会显示协商历史，负责价格协商的用户能够根据协商历史给出新的定价。当形式发票价格被双方接受后，供应商可以选择将形式发票创建为预制发票。预制发票的样式类似于增值税发票，增值税发票会有限额规定，所以在创建预制发票时，会对超过限额的形式发票进行拆票处理，拆分成多张金额小于限额的预制发票。因为买卖双方在交易过程中可能存在退货的操作，当采购商就退货商品进行重传对账单时，系统会将已经开出但未被归集的预制发票进行退票处理，重新计算发票金额。当供应商对预制发票查看确认后，就可以进行生成增值税发票操作，开出正式的发票。当供应商和采购商将增值税发票录入系统后，则流程结束，发票被置为归集状态。

### 3.2.2 功能需求

根据图3.2的系统用例图，具体分析Tradeshift电子发票系统的功能需求，如表3.1-表3.8所示。其中表3.2-3.6描述了发票生成相关模块的用例。

形式发票是一种非正式发票，供应商可以通过形式发票对采购商购买的商品进行报价[李海珍, 2012]。预制发票的展示形式类似于增值税专用发票，在Tradeshift电子发票系统中，预制发票是用来产生和下载增值税专用发票的数据载体，形式和增值税发票相近，预制发票帮助建立增值税发票与采购商上传的对账单数据的对应关系，能够方便用户在之后进行数据跟踪，利于查账。[詹伟等, 2014]。

其中，发票相关数据，包括对账单、形式发票等都以通用商业语言的方式进行承载。通用商业语言（UBL）是一个开放标准的XML文件[UBL, 2018]，通常被用于电子商务采购和运输，如采购订单、发票库、物流运输和货运单。 它的目的是消除数据的不一致性，使文档按照统一的标准进行保存，为中小企业提供电子商务的切入点[夏萍等, 2014]。在Tradeshift电子发票系统对外暴露的上传对账单接口中，需要用户上传符合通用商业语言标准的XML文件才能正确解析。

表3.1建立公司网络用例描述

|  |  |
| --- | --- |
| ID | UC1 |
| 名称 | 建立公司网络 |
| 详细描述 | 商家可以在形式发票平台搜索到其他注册商家并申请建立联系。如果申请通过，企业双方就会建立起公司网络。建立起联系的买卖双方可以进行发送对账单等操作。 |
| 优先级 | 高 |
| 数据输入 | 用户要建立关系的公司信息 |
| 数据输出 | 返回网络建立结果信息 |

表3.2 上传对账单用例描述

|  |  |
| --- | --- |
| ID | UC2 |
| 名称 | 上传对账单 |
| 详细描述 | 采购商可以通过API的方式上传对账单；  采购商在上传的文档数据中，需要定义文档的唯一document\_id，定义上传文档的数据类型，添加采购商和供应商的信息，包括买卖双方的名称、税号和地址信息以及交易商品信息。上传文档的主体内容包含多条商品交易记录，每一条商品记录中包括该行在对账单中的行号，交易记录的唯一标识transaction\_id、商品名称、物料名称、税收编码、商品单价、购买数量、税率、价格总额等。上传成功后，系统会根据transaction\_id进行重传判断，如果发生负量的重传并且相同transaction\_id的记录已经被形式发票占用，则将形式发票状态置为冻结。 |
| 优先级 | 高 |
| 数据输入 | 对账单文档数据 |
| 数据输出 | 对账单上传结果，包括上传的文档id |

表3.3 创建形式发票用例描述

|  |  |
| --- | --- |
| ID | UC3 |
| 名称 | 创建形式发票 |
| 详细描述 | 当供应商基于新上传的对账单数据创建或者对冻结的形式发票进行应用变更操作时会发生创建形式发票操作。  1.基于新上传的对账单：供应商可以针对整个对账单，也可以对多个对账单中的每一行记录按行勾选，当某条记录被选中时，系统会根据设置的规则，加入关联的行，并排除一些与选中记录不应该放在一起开票的记录。用户勾选结束后即可创建形式发票。2.应用变更：对账单发生负量的重传时，对应的形式发票会被冻结，需要进行应用变更操作。应用变更操作会将冻结形式发票中对应的对账单数据和重传的对账单数据作为创建形式发票的元数据。  系统根据对账单id和行id生成对应的形式发票文档，根据已有的物料名称、商品名称、商品价格、税收编码的映射关系替换所创建的形式发票中的商品部分信息，以文档的方式保存在系统中并显示在界面上。 |
| 优先级 | 高 |
| 数据输入 | 对账单id，若勾选开票，则会输入选中的记录的line\_id |
| 数据输出 | 以通用商业语言方式承载的形式发票 |

表3.4 编辑形式发票用例描述

|  |  |
| --- | --- |
| ID | UC4 |
| 名称 | 发送形式发票 |
| 详细描述 | 供应商在创建形式发票后可以对形式发票进行编辑和保存并发送给采购商。供应商可以删除某行记录，或者编辑某行记录的商品名称、商品价格、商品数量、税率、税收编码，相同商品名称的商品会进行批量更新，每一行显示的合计价格会根据用户的修改重新计算并刷新。  编辑完成后可以将形式发票保存为草稿，但不发送给对应的采购商，系统会释放在编辑过程中删除的行，使该行可以被重新勾选以创建形式发票。同时系统会根据编辑后的形式发票更新商品映射关系并保存形式发票文档。编辑完成后供应商也可以直接点击发送按钮，系统会释放行、保存形式发票文档并更新映射关系。然后将形式发票发送给对应的采购商。 |
| 优先级 | 高 |
| 数据输入 | 通用商业语言文档，形式发票id，文档类型 |
| 数据输出 | 保存或者发送的结果 |

表3.5 议价用例描述

|  |  |
| --- | --- |
| ID | UC5 |
| 名称 | 议价 |
| 详细描述 | 供应商对新创建的形式发票中的每一行记录的商品价格进行编辑，发送给采购商。在系统将形式发票发送给采购商审批之前，系统会先在workflow模块对价格做一个比对，如果供应商发送的形式发票中的每一种商品的价格都低于采购商之前给定的参考价格，那就价格审批直接通过，不需要再让采购商进行审批操作。否则就发送给采购商进入双方议价流程。  采购商会对形式发票中价格不一致的行进行审批。需要审批的行都会显示供应商的定价和采购商之前定义的参考价格。采购商可以选择接受供应商价格，或者拒绝后线下和供应商协商价格并在系统中填入新的价格，参考价格也会更新为采购商新定的价格。采购商分为两级审批，第一级是采购员对商品价格进行审批，第二级是财务会计对商品价格进行审批，只有当二级审批通过后，双方价格协商流程结束。否则，系统会将审批后的形式发票退还给供应商，供应商重新编辑价格后发送。双方能够就价格进行多次协商和审批操作，审批流程以workflow模块自动接收或者采购商二级审批通过结束。 |
| 优先级 | 高 |
| 数据输入 | 通用商业语言文档，形式发票id，文档类型 |
| 数据输出 | 审批数据和审批结果 |

表3.6 生成预制发票用例描述

|  |  |
| --- | --- |
| ID | UC6 |
| 名称 | 生成预制发票 |
| 详细描述 | 供应商在核对好形式发票后，可以创建预制发票。预制发票样式类似于正式的增值税发票。预制发票为供应商开出正式发票前提供一个参考，帮助供应商对开票商品、金额等进行确认，以避免开出废票的情况。在生成预制发票过程中，需要根据商品分类，先将商品根据能否开在同一张增值税发票中进行划分，然后将相同的商品合并。对于划分结果，根据发票限额分别进行拆票。在拆票过程结束后，会生成一张或者多张预制发票。 |
| 优先级 | 高 |
| 数据输入 | 通用商业语言形式承载的形式发票文档 |
| 数据输出 | 通用商业语言形式承载的预制发票文档 |

表3.7生成增值税专用发票用例描述

|  |  |
| --- | --- |
| ID | UC7 |
| 名称 | 生成增值税专用发票 |
| 详细描述 | 供应商可以将发票数据导出到本地，然后将导出后的数据导入开票系统，进行开票。供应商也可以直接使用平台上与开票软件进行集成的功能，在系统中直接开票，进行增值税发票查看和打印。 |
| 优先级 | 高 |
| 数据输入 | 符合开票系统规范的发票数据 |
| 数据输出 | 返回增值税专用发票数据 |

表3.8发票录入用例描述

|  |  |
| --- | --- |
| ID | UC8 |
| 名称 | 发票录入 |
| 详细描述 | 开出的增值税发票可以通过手动输入或者影像文件的方式录入到系统中。供应商在平台上通过开票接口开票或者线下手动开票后，需要在系统上将发票录入，进行归集操作。采购商收到发票后，在系统上进行录入操作，以用于查看、管理、抵扣进项发票。 |
| 优先级 | 高 |
| 数据输入 | 系统中保存的增值税专用发票信息；用户上传的发票信息 |
| 数据输出 | 返回发票录入的结果和状态 |

## 3.3 系统总体结构与模块设计

### 3.3.1 总体结构

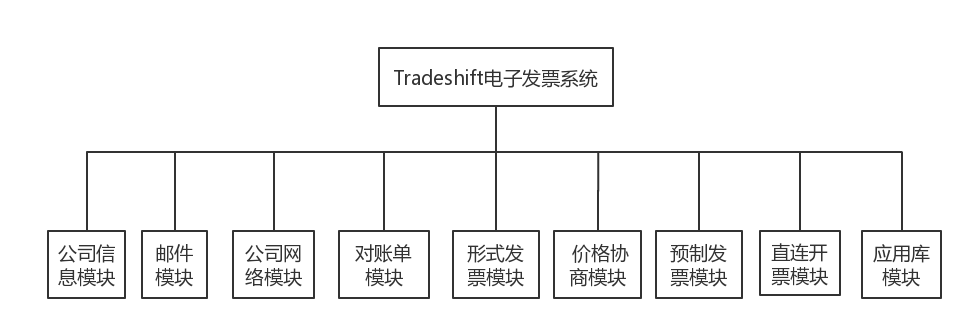


图3.3 Tradeshift电子发票系统总体结构图

如图3.3所示，电子发票系统主要由公司信息模块、邮件模块、公司网络模块、对账单模块、形式发票模块、价格协商模块、预制发票模块、直连开票模块、应用库模块组成。本文主要围绕发票生成相关模块进行重点描述，包括对账单模块、形式发票模块、价格协商模块、预制发票模块。

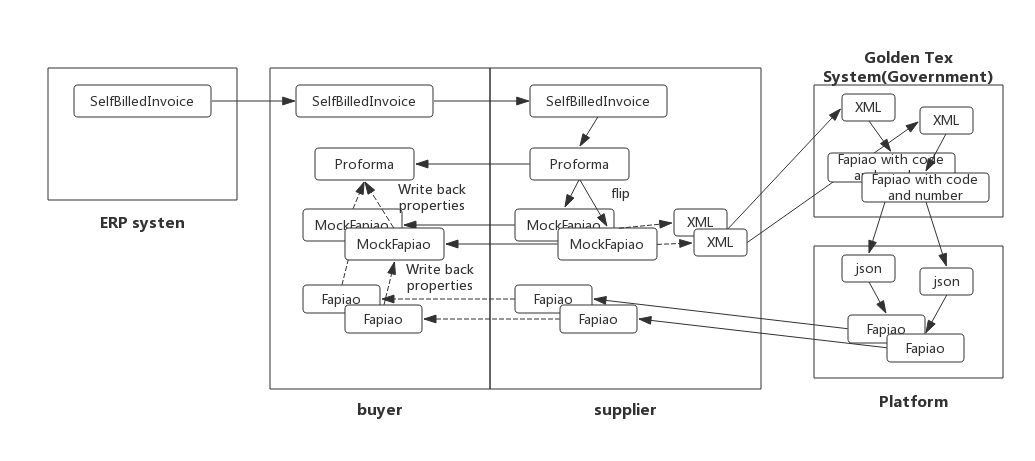


图3.4 发票生成相关模块数据流转图

图3.4显示了在系统中发票生成相关模块的数据的流转方式。在ERP系统中存储采购商上传的对账单，目前系统中包含SelfBilledInvoice类型的文档的对账记录，以满足通用商业语言标准的XML文件进行存储。采购商buyer上传对账单数据，将SelfBilledInvoice发送给供应商supplier，供应商接收后创建形式发票，并将形式发票发送给采购商。当采购商接受形式发票后，供应商将形式发票开票为预制发票，并发送给采购商。供应商进行直连开票，预制发票数据转化为xml通过国税开放的开票接口进行开票操作，创建增值税发票，并以json的格式回传增值税发票的号码和代码。在电子发票系统中对回传数据进行处理，在系统中形成电子版的增值税发票，并发送给采购商。

### 3.3.2 对账单模块

采购商可以将企业内部系统与电子发票开放的API接口进行集成，将对账单上传至电子发票平台。一旦上传成功，平台会对对账单进行解析和处理，并自动将对账单数据分发到对应的供应商账号内。当供应商打开对账单文档的详情页面，在此页面上，供应商可以打印对账单文档、下载对账单的PDF或UBL文档、创建形式发票等。

如图3.5所示，采购商上传对账单的主要过程如下。首先对上传的文档数据做合法性检查，检查是否符合通用商业语言文档规范，检查文档id的唯一性。然后对文档解析，获得采购商商品采购数据。对每一行的商品采购记录，根据transactionid来判断是新的记录还是重传记录。如果是新的商品记录，则直接将记录新增到待开票数据中。如果是已有的商品记录，说明供应商历史上传的对账单的记录需要修改，或者商品发生进货、退货情况的时候，会发生重传来更新商品的采购数据。根据累积入库量、累积退库量、开票累积量这些字段来计算新的商品采购数量。

如果商品是进货操作，则直接计算出增量，将新增数量的商品记录添加到待开票数据中。如果是退货操作，则需要对之前上传的相同transactionid的记录进行查询，判断之前的记录是否已经在形式发票中被占有。如果未被占有，则直接将退货操作对应的负量记录添加到待开票数据中；如果已经被占有，在将记录增加到待开票数据之前，需要将对应的形式发票状态设置为冻结。当供应商在接下来的流程中操作形式发票时，需要供应商通过应用变更操作创建新的形式发票并删除被冻结的形式发票。

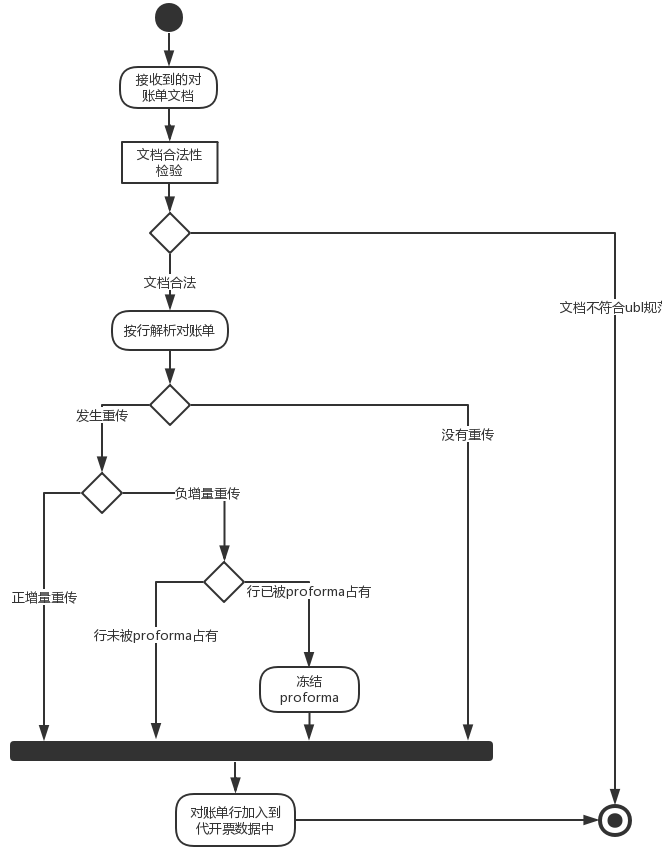


图3.5 上传对账单活动图

### 3.3.3 形式发票模块

形式发票包含着详细的采购信息，包括商品名称、规格型号、数量、单价、税率、总金额等信息，是在供应商开具正式发票之前的一种形式化的文档。供应商将形式发票发送给买家，针对采购的商品进行报价后，经过买家同意后，即可开具正式发票。

供应商可以在开具正式发票前，在电子发票平台上创建形式发票，将采购商上传的文档数据解析后以清晰的方式在界面上展示，并对形式发票进行编辑后发送给采购商，这样可以避免因错开发票导致的效率降低。

待开票的对账单数据，可以由上传的对账单勾选获得，或者通过应用变更操作从冻结的形式发票中获得。系统根据对账单行记录，根据alongwithkey，transactionid等字段，获得创建的形式发票所包含的所有行，并根据物料名称、商品名称和税收编码的映射关系替换行中的部分信息，最后创建完整的形式发票。具体流程如图3.6所示。供应商通过勾选任意行对账单数据、选择全部对账单行数据、选择单个对账单文档、从冻结的形式发票中获得对账单数据这四种方式进行创建形式发票。

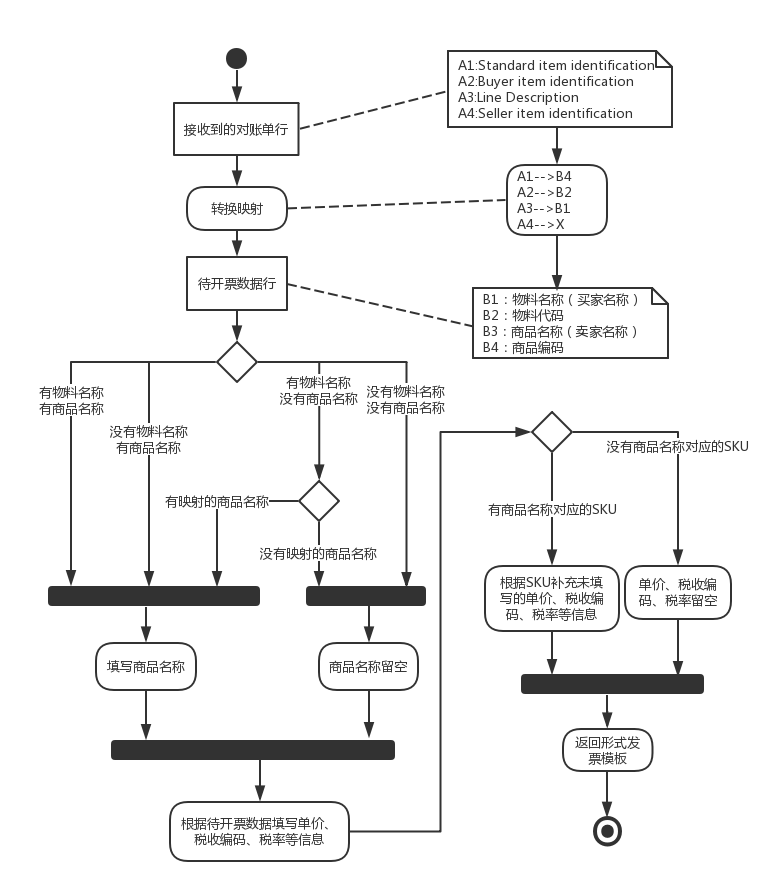


图3.6 创建形式发票活动图

当创建形式发票时，因为采购商上传的对账单中的商品名称不符合增值税发票中的商品名称要求，所以需要在对形式发票处理时，将不符合标准的商品名称替换，给每一行商品记录都添加正确的商品名称和税收分类编码这两项。这两项数据第一次由供应商手工输入，一旦维护过，映射关系会存储在数据库中，后续同一物料编码上传时，就会根据映射关系找出维护过的商品名称和税收分类编码。

当系统接收到对账单后，会根据对账单行中的相关字段进行映射转换，转换成可开票数据行。如图3.7所示，为对账单中商品相关信息。对账单中的Standard item identification、buyer item identification、line description、seller item identification这四个字段分别对应商品编码、物料代码、物料名称、商品名称。物料代码和物料名称是采购商在对账单中定义的，商品名称是供应商在发票中定义的可开票名称。多个不同的物料名称可以对应于同一个商品名称。

<cac:Item>

<cbc:Description>qt型纸箱</cbc:Description>

<cbc:Name>qt型纸箱</cbc:Name>

<cac:BuyersItemIdentification>

<cbc:ID schemeAgencyID="9" schemeID="GTIN">A001013293</cbc:ID>

</cac:BuyersItemIdentification>

<cac:SellersItemIdentification>

<cbc:ID schemeAgencyID="9" schemeID="GTIN">A000000001</cbc:ID>

</cac:SellersItemIdentification>

<cac:StandardItemIdentification> <cbc:ID schemeAgencyID="9"

schemeID="GTIN">1090602050000000000</cbc:ID>

</cac:StandardItemIdentification>

</cac:Item>

图3.7 对账单商品信息相关字段

当供应商创建形式发票时，如果待开票行数据中已经填写了商品名称，则直接使用已有的商品名称；若待开票行数据中只有物料名称，没有商品名称，则需要通过物料名称，根据系统中保存的映射关系得到相应的商品名称；若映射关系不存在，则商品名称留空，需要供应商在编辑形式发票页面进行填写。最后，根据映射关系获得对应商品信息，补充待开票数据中未填写的单价、税收编码、税率等信息，以文档的形式返回形式发票。

### 3.3.4 价格协商模块

当采购商上传开票数据时，系统会将文档中采购商的采购价格提取出来，存入价格参照表中。供应商在给采购商发送形式发票前，需要填写商品的出售价格。然后通过价格协商模块，买卖双方协商价格，只有当双方对商品价格有了统一的定价，供应商才能根据形式发票创建预制发票。价格协商的具体业务过程如图3.8所示。

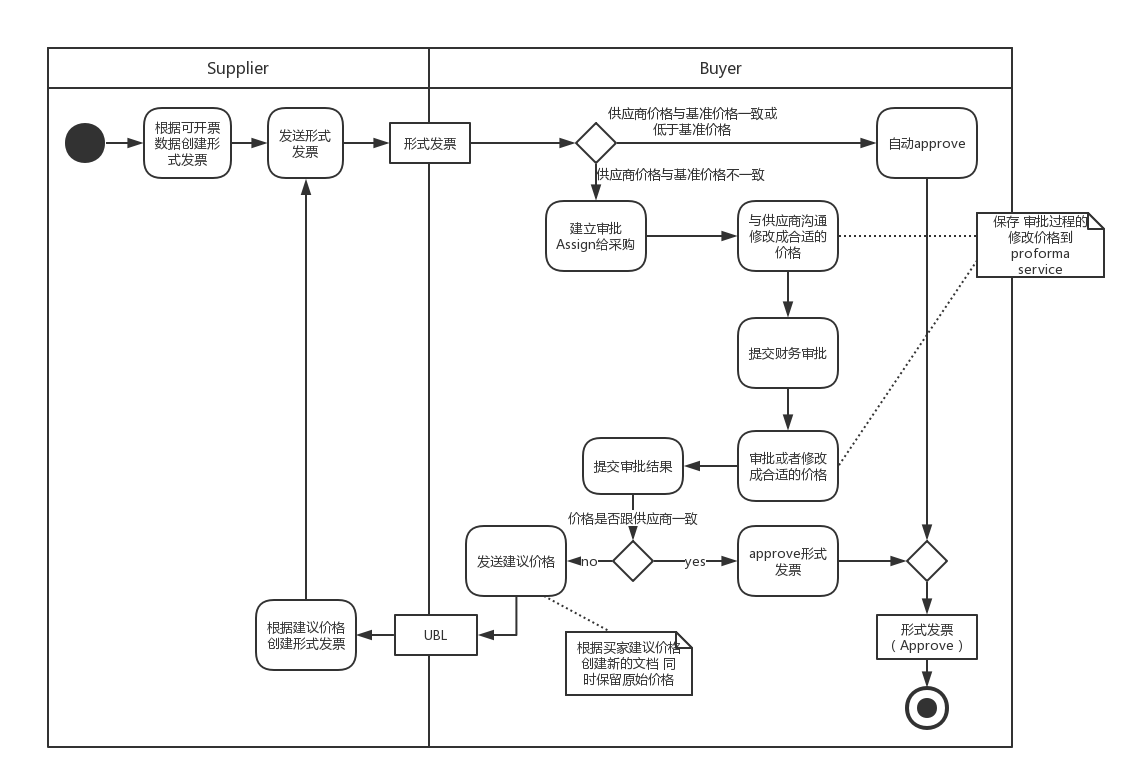


图3.8 价格协商活动图

供应商根据待开票数据创建形式发票，确定好价格后将形式发票发送给采购商。采购商接收到形式发票后，系统会首先将每行商品的价格与采购商在对账单中上传的基准价格进行对照，如果整个形式发票中，供应商制定的商品价格都不大于采购商的基准价格，则系统直接通过形式发票，将形式发票的状态置为采购商接受状态而不需要采购商进行人工审批的操作。当供应商提供的价格高于基准价格时就会进入价格协商审批流程。

审批流程分为采购和财务两级审批。采购商的采购账号会收到审批任务，可以在页面中查看不需要审批的行和需要议价的商品。采购通过在线下或者线上沟通的方式，重新确定价格。当采购拒绝供应商的价格，则价格仍以系统中的基准价格为准；当采购接受供应商的价格，或者重新定义一个新的价格，基准价格更新为当前定义的价格。当采购对形式发票中的商品审批完成后，议价任务会流转到下一级的财务账号。财务会对采购审批的结果进行审核，通过采购的审批结果或者对采购的审批结果进行修改，然后提交审批结果。系统会自动对审批结果进行检查，如果新定义的价格与供应商一致，则系统通过形式发票，将形式发票置为采购商接受状态。如果价格不一致，则系统会将新的建议价格添加到形式发票中，以UML文档的形式发送给供应商。供应商可以在系统中看到根据新的建议价格生成的形式发票，供应商需要对价格进行确认或修改并重新发送形式发票。买卖双方可以就形式发票中的商品价格进行多次协商，在协商过程中，可以查看历史协商记录，并重新定义新的价格。当供应商和采购商就形式发票中所有商品的价格达成一致时，议价流程结束，形式发票被采购商接受。

### 3.3.5 预制发票模块

当供应商创建形式发票，并发送给采购商被接受后，供应商可以根据形式发票创建预制发票。预制发票是增值税发票的一种数据承载方式，形式和增值税发票相近，预制发票帮助建立增值税发票与采购商上传的对账单数据的对应关系，能够方便用户在之后进行数据跟踪，利于查账。另外，预制发票能够提供对增值税发票的预览，帮助供应商在直连开票前对发票信息进行查看和确认，避免误开票的情况。

拆票的主要流程如图3.9所示。首先根据proformaid查询拆票状态表，获得当前形式发票的拆票状态，共有未拆票，拆票中，已拆票三种状态。如果是拆票中，则返回拆票中的状态；如果是已拆票，则从预制发票表中根据proformaid查询预制发票并返回。如果状态是未拆票，则进入拆票主流程。根据proformaid获取待拆票形式发票文档。判断形式发票中是否存在重传的记录，如果发生了重传且原来的形式发票已创建过预制发票，则找到形式发票中重传的记录对应的预制发票，将其打上标记，将预制发票退回并按照新的数量重新创建预制发票。如果没有发生过重传，则对形式发票直接进行拆票操作。根据形式发票对应的待开票数据sourceline表中的groupkey，将形式发票中的记录分组，同组中的记录表示可以放入同一张预制发票中。然后对于每一组，根据mergekey将相同的商品进行合并为一条记录并重新计算金额。将记录分组合并之后，针对每一组的商品记录金额进行拆票。

不同的企业能够按照其实际的经营状况向主管税务机关申请开具不同开票额度的增值税专用发票。每个供应商会申请不同的发票开票限额。系统会根据供应商在应用库模块中的开票管理app中设定的开票限额，将每一组的商品记录进行拆票。如果某一行的商品金额超过了限定的金额，则会进行拆分，拆成多张预制发票。如果多行记录都小于限定金额，则会进行合并，将多行记录生成为一张预制发票。最后，系统会保存形式发票和预制发票的对应关系，并将拆票结果返回。

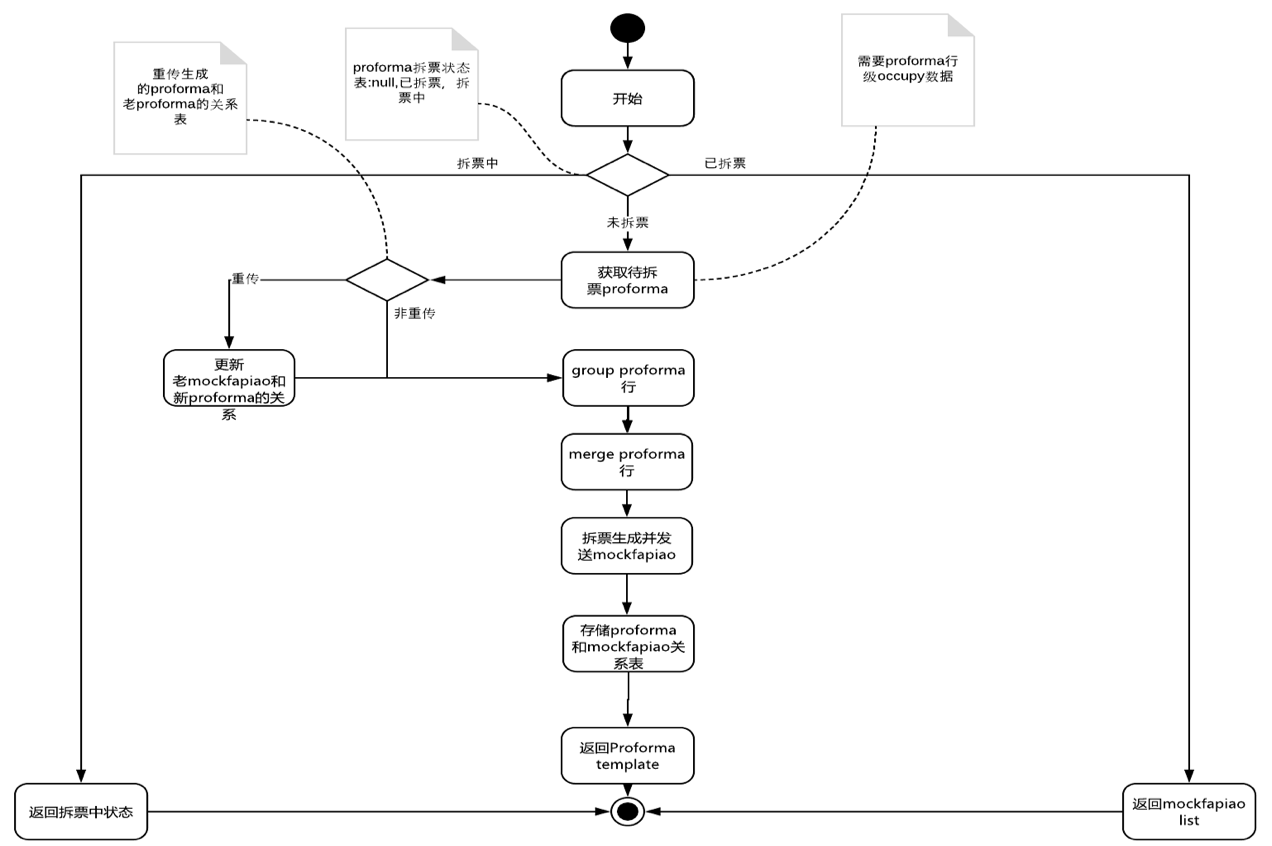


图3.9 拆票活动图

## 3.4 数据库设计

### 3.4.1 数据库总体设计

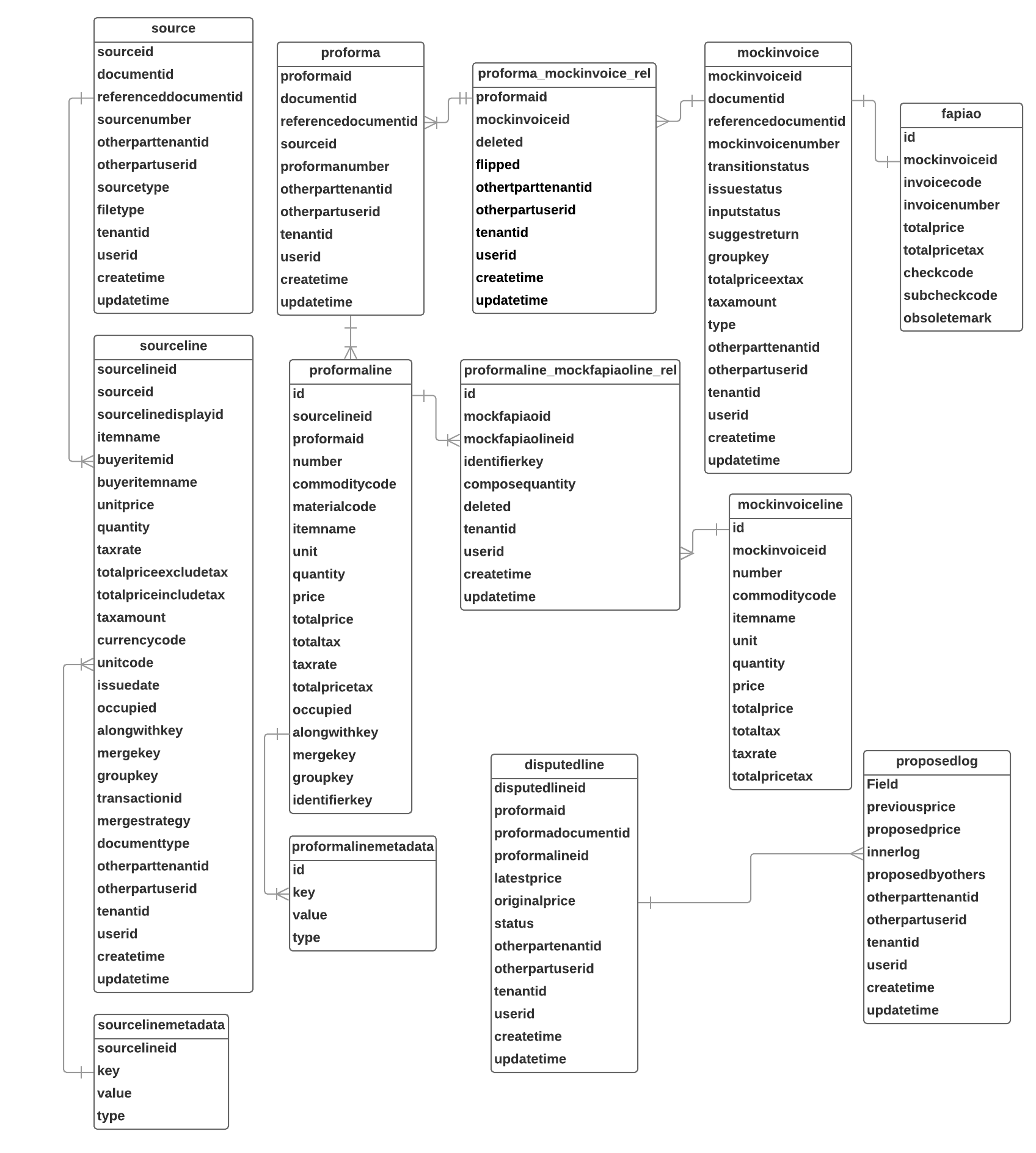


图3.10 Tradeshift电子发票系统发票生成模块数据库设计图

图3.10是Tradeshift电子发票系统发票生成模块的数据库设计图。

sourceline：在对账单上传后，会将对账单逐行解析，插入sourceline实体表中，会包括关联文档的id信息，商品名称、价格、税率以及多个key值。其中alongwithkey表示当对账单中的某一行被勾选时，拥有相同alongwithkey的行记录会被同时勾选以创建形式发票。 mergekey表示形式发票中哪几行是同类商品，应该在创建预制发票时合并计算为预制发票中的一行。groupkey表示形式发票中的哪些行在拆票过程中可以放入同一张预制发票中，不同的groupkey对应的行不能放于同一张预制发票中。transactionid是上传的行记录的唯一标识，根据transactionid判断新上传的对账单中的行记录是新增加的纪录，还是重传的记录。重传表示之前上传的行记录信息有误，需要通过重传的数据替换原来上传的行数据。

proforma：对账单创建形式发票后，会将形式发票的相关数据插入proforma实体表中。

proformaline：在生成形式发票时，会将形式发票中每一行的详细信息插入proformaline实体表中。

mockinvoice：对形式发票进行拆票后，会生成预制发票，预制发票相关数据会插入mockinvoice实体表中，会包括关联文档的id、文档状态等信息。其中，transactionstatus表示预制发票的状态，issuestatus表示预制发票是否已导出开票，suggestreturn表示当采购商发生重传，应用变更后，对已经生成的预制发票标记为建议退票状态，以解决重传发生的退货情况，减少开红票的情况。

mockinvoiceline：在生成预制发票时，会将预制发票中每一行的详细信息插入mockinvoiceline实体表中。

fapiao：预制发票生成正式的增值税发票，数据会存入fapiao实体表中，包括发票的编码，发票的总价和关联的预制发票id等。

disputeline：用于保存议价过程中形式发票中每一行商品的参考价格，其中originalPrice和lastPrice初始时保存的是买家上传的原始数据中的价格，lastPrice会在议价过程中根据买家更新的价格而更新，作为议价过程中的参考价格。

proposedlog:用于保存整个议价过程的价格变化记录，当买卖双方进行协商议价时，显示价格协商历史记录。

### 3.4.2 proformaline表设计

表3.9的profromaline表用于存储形式发票的行信息，形式发票中每一行在页面显示的信息包括商品名称、商品编码、计量单位、商品数量、商品单价、商品税率、行商品总金额、行商品总税额、价税合计等。用户在界面上对形式发票行数据进行更新后都会更新proformaline表中对应的行数据。

表3.9 proformaline表设计

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 描述 |
| id | UUID | 形式发票行数据唯一标识 |
| sourcelineid | UUID | 形式发票行数据关联的明细行唯一标识 |
| proformaid | UUID | 形式发票数据唯一标识 |
| number | INTEGER | 明细行号 |
| commoditycode | VARCHAR(50) | 商品编码 |
| materialcode | VARCHAR(100) | 物料代码/物料名称 |
| itemname | VARCHAR(100) | 商品名称 |
| unit | VARCHAR(50) | 计量单位 |
| quantity | NUMBERIC(21,10) | 商品数量 |
| price | NUMBERIC(21,16) | 商品单价 |
| totalprice | NUMBERIC(21,2) | 行级别商品总金额 |
| totaltax | NUMBERIC(21,2) | 行级别商品总税额 |
| taxrate | NUMBERIC(6,2) | 行级别商品税率 |
| totalpricetax | NUMBERIC(6,2) | 价税合计 |
| occupied | BOOLEAN | 形式发票行占用标志 |
| alongwithkey | VARCHAR(200) | 形式发票行数据归属关键字 |
| mergekey | VARCHAR(200) | 形式发票行数据合并关键字 |
| groupkey | VARCHAR(200) | 形式发票行数据分组关键字 |
| transactionid | VARCHAR(200) | 形式发票行数据重传关键字 |

### 3.4.3 mockinvoiceline表设计

表3.10的mockinvoiceline表主要用来存储预制发票的行信息，预制发票中的每一行在界面中显示的信息包括了商品编码、商品名称、计量单位、商品数量、商品单价、行级别商品总金额、行级别商品总税额、行级别商品税率、价税合计等。当形式发票进行拆票操作后，会根据拆票算法将发票拆分并将拆分后的信息存入mockinvoiceline表。

表3.10 mockinvoiceline表设计

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 描述 |
| id | UUID | 发票行数据唯一标识 |
| mockinvoiceid | UUID | 发票数据唯一标识 |
| number | INTEGER | 明细行号 |
| commoditycode | VARCHAR(50) | 商品编码 |
| itemname | VARCHAR(100) | 商品名称 |
| unit | VARCHAR(50) | 计量单位 |
| quantity | NUMERIC(21, 2) | 商品数量 |
| price | NUMERIC(21, 16) | 商品单价 |
| totalprice | NUMERIC(21, 2) | 行级别商品总金额 |
| totaltax | NUMERIC(21, 2) | 行级别商品总税额 |
| taxrate | NUMERIC(6, 2) | 行级别商品税率 |
| totalpricetax | NUMERIC(21, 2) | 价税合计 |

### 3.4.4 fapiao表设计

表3.11的fapiao表主要用于存储增值税专用发票信息，当用户选择预制发票直连开票时，调用税务系统的开票接口，开出增值税专用发票后会将增值税发票的四字段信息（发票号码，发票代码，开票日期，校验码后六位）回写到Tradeshift电子发票系统中。

表3.11 fapiao表设计

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 描述 |
| id | UUID | 发票数据唯一标识 |
| premadefapiaoid | UUID | 预制发票数据唯一标识 |
| invoicecode | VARCHAR(30) | 发票代码 |
| invoicenumber | VARCHAR(30) | 发票号码 |
| totalprice | NUMERIC(21, 2) | 合计金额 |
| totalpricetax | NUMERIC(21, 2) | 价税合计 |
| checkcode | VARCHAR(30) | 发票校验码 |
| subcheckcode | VARCHAR(30) | 发票校验码后6位 |
| obsoletemark | SMALLINT | 发票数据作废标识 0：作废 1：正常 |

## 3.5 本章小结

本章主要介绍了Tradeshift电子发票系统的分析和设计过程，首先描述了系统的用例图，然后对发票生成模块的相关用例进行详细描述，包括上传对账单、创建形式发票、发送形式发票、差价协商、创建预制发票。然后，根据需求分析结果，本文给出了电子发票系统的整体架构，将系统按照模块进行划分，并从活动图切入详细介绍了电子发票生成相关模块的职责以及包含的功能点。最后，详细介绍了电子发票系统发票生成相关模块的数据库设计，并列出了多张关键表的设计。

# 第四章 电子发票系统主要功能的实现

## 4.1 电子发票系统的主要功能概述

Tradeshift电子发票系统主要目标是为用户提供从采购到开票的一体化电子流程。系统应用库模块中提供了供应商市场app，供应商可以通过平台上传商品，定制商品价格、数量、税率等属性。采购商可以直接通过平台采购并记录数据进行开票，也可以通过上传线下的采购数据进行开票。供应商可以根据采购商上传的对账单数据，创建形式发票、预制发票和增值税发票。本文对发票生成相关模块进行了重点描述。

上传对账单主要通过为采购商提供后台接口，以通用商业语言文档的形式上传线下采购数据，系统保存数据并发送给供应商。系统对对账单文档进行解析，将每一条交易记录进行解析，在解析过程中进行重传判断，根据不同的重传情况进行不同的数据处理，最后将对账单数据按行存入数据库表中。对账单上传后系统就会同步返回上传结果，然后将文档放入消息队列，使用HornetQ实现异步发送文档。

创建形式发票主要根据采购商上传的对账单创建电子文档，它包含商品名称、规格型号、数量、单价、税率、总金额等信息。形式发票是一种非正式的发票，供应商可以通过形式发票对采购商采购商品进行报价。

价格协商是指供应商通过形式发票对采购商购买的商品进行报价后，采购商和供应商可以就商品价格进行多次协商，通过对文档价格进行修改并发送的方式进行协商。当双方对形式发票中所有商品价格协商一致时，则形式发票可以用于创建预制发票。

创建预制发票是指根据供应商设置的开票限额对形式发票进行拆票处理，根据设置的条件将形式发票拆分成类似增值税发票的多张预制发票。预制发票的展示形式与增值税发票相近，建立增值税发票与采购商上传的对账单数据的对应关系，能够方便用户在之后进行数据跟踪。预制发票能够提供对增值税发票的预览，帮助供应商在直连开票前对发票信息进行查看和确认，避免误开票的情况。

## 4.2 电子发票系统的整体设计

### 4.2.1 电子发票系统的整体框架

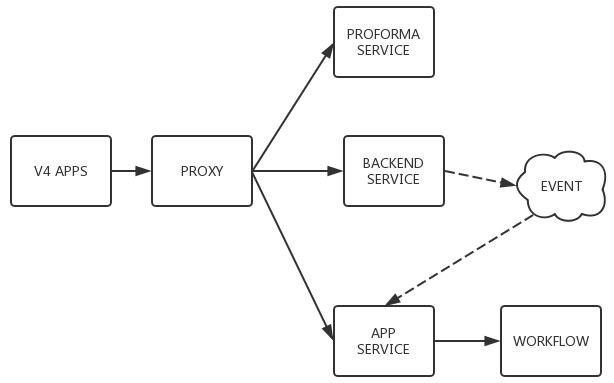


图4.1 Tradeshift发票系统组织结构

Tradeshift电子发票系统基于微服务的架构来实现，图4.1描述了Tradeshift电子发票系统的组织结构。

V4 APPS：主要负责前端的呈现，包括前端页面展示和部分逻辑的处理等。

PROXY：所有的外部请求都需要通过PROXY转发到各个微服务。

PROFORMA SERVICE：主要用于处理电子发票相关的操作，包括创建形式发票，编辑形式发票，保存形式发票，创建预制发票等。

BACKEND SERVICE：主要用于处理用户请求的后端操作，例如保存、管理文档和文档状态，管理公司信息等。

APP SERVICE：主要用于应用库模块，管理系统中的APP，包括APP的版本，用户激活情况等。

WORKFLOW：WORKFLOW是通过监听BACKEND SERVICE发出的事件，来触发一系列工作流，而这个工作流的过程是可插拔可配置的。

### 4.2.2 电子发票生成相关模块的类图

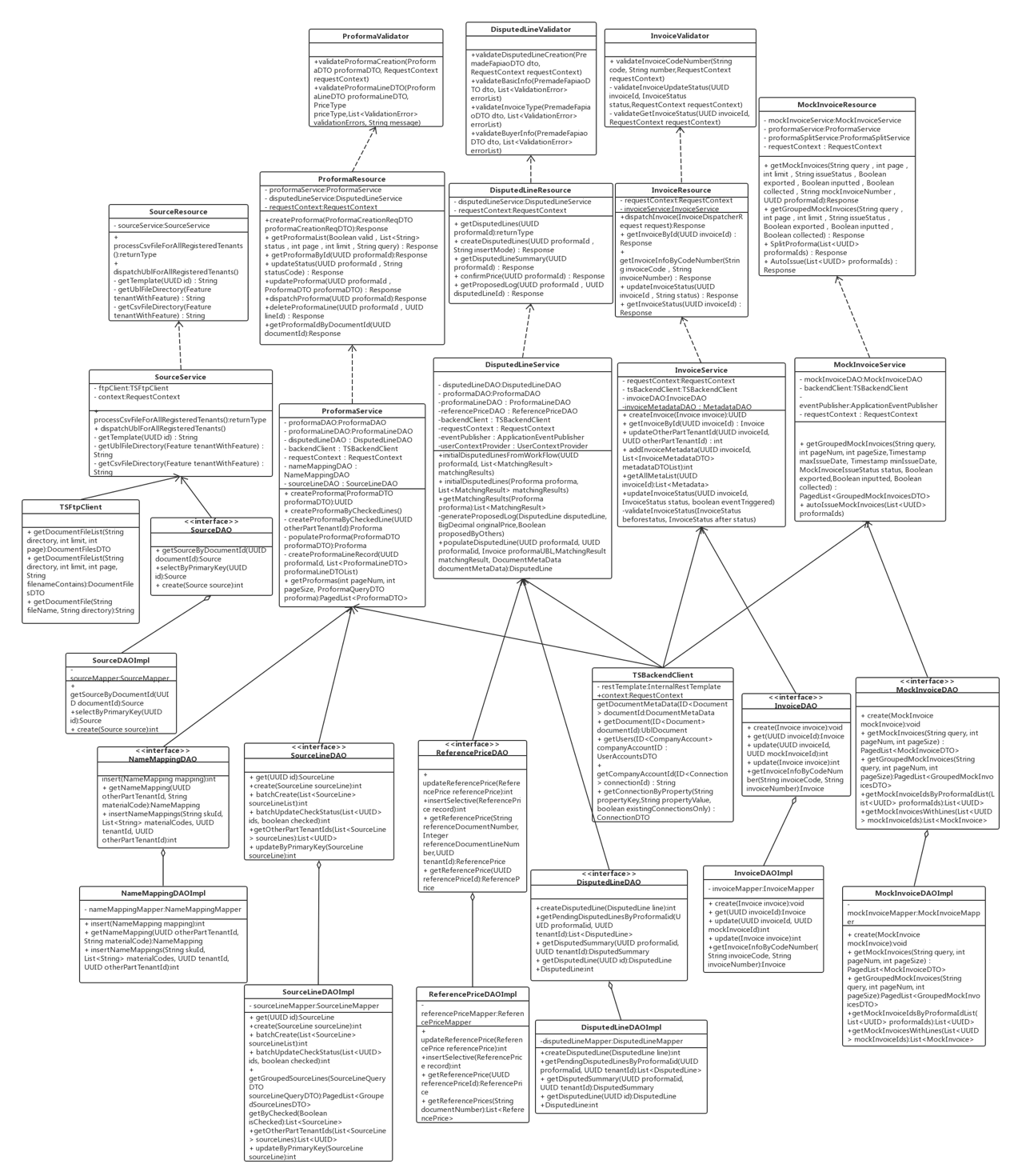


图4.2 电子发票生成相关模块类图

如图4.2所示，是Tradeshift电子发票生成相关模块的类图。Resource类主要提供响应前端请求的Restful接口，包括处理待开票原始数据的SourceResource类，处理形式发票相关操作的ProformaResource类，处理议价相关流程的DisputedLineResource类，处理拆票流程的InvoiceResource类，处理预制发票流程的MockInvoiceResource类。Resouce类在获得请求参数后会首先调用Validator相关类对参数进行合法性检验。当请求参数通过合法性检验后，Resource类会分别调用相对应的Service类。Service相关类主要实现业务流程的具体处理，包括正常业务流程的实现和异常流程的处理等。除了电子发票生成模块相关的SourceService、ProformaService、DisputedLineService、InvoiceServic、MockinvoiceService这些类之外，还包括TSFtpClient类来处理待开票元数据的FTP读写问题，TSBackendClient类来处理需要调用Backend-Service服务的接口。Service类调用DAO相关类接口对数据库进行访问，对数据对象进行增删查改的操作。

## 4.3 对账单模块

### 4.3.1 对账单模块设计描述

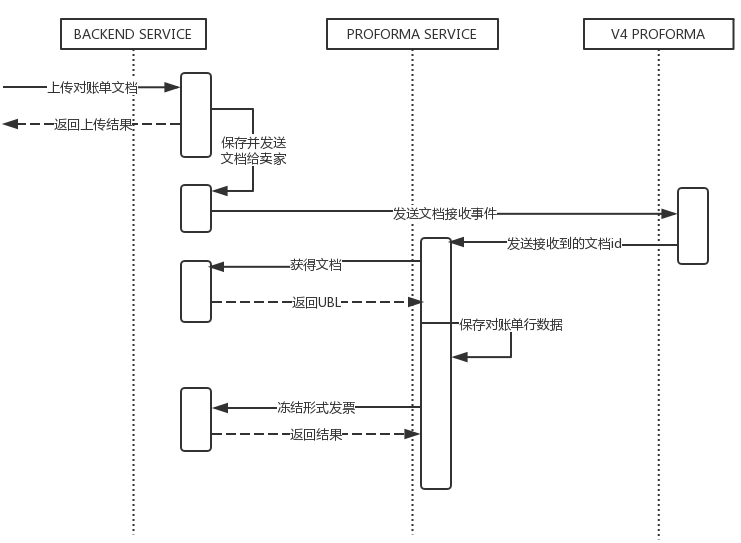


图4.3 上传对账单时序图

上传对账单模块主要为采购商提供上传开票数据的接口，时序图如图4.3所示。上传对账单模块主要依赖BACKEND，PROFORMA SERVICE，V4 PROFORMA三个微服务组件来实现。采购商通过BACKEND向外暴露的后端接口上传对账单，上传过程中BACKEND会对上传的对账单的格式和信息进行合法性验证，当验证通过，则将对账单发送给对账单中标记的供应商。在上传对账单的过程中使用了消息队列并应用了发布订阅的设计模式[钱剑锋, 2011]。当BACKEND SERVICE将文档发送给供应商后，作为订阅者的V4 PROFORMA就会接收事件，开始处理对账单文档。V4 PROFORMA会调用PROFORMA SERVICE服务对对账单文档进行处理。PROFORMA SERVICE将根据获得的docID向BACKEND SERVICE请求完整的以通用商业语言文档方式存储的对账单，然后将对账单文档按行解析成元数据并存入数据库。在解析行的过程中，对重传的行进行检查，调用BACKEND SERVICE对重传的行进行重新计算并将包含重传行的形式发票进行冻结等处理。

### 4.3.2 对账单模块实现代码

runtime.addEventListener('AFTER\_DOCUMENT\_RECEIVE', function(evt) {

var documentId = \_.get(evt, 'data.msg.ObjectId');

var documentType = \_.get(evt, 'data.msg.Metadata.DocumentType.type');

if (

\_.isEmpty(documentId) ||\_.isEmpty(documentType) ||

!\_.includes([SELFBILLEDINVOICE, RECEIPTADVICE], documentType) ||

\_.isEqual(documentId, oldDocumentId)

) {

return;

}

);

oldDocumentId = documentId;

putLinesToProformaService(documentId, documentType)

.then(() => {

oldDocumentId = '';

);

})

.catch(err => {

oldDocumentId = '';

);

})

.done();

});

图4.4 V4 PROFORMA监听接收对账单事件

如图4.4是V4 PROFORMA服务对对账单的状态进行事件监听，通过addEventListener方法对’ AFTER\_DOCUMENT\_RECEIVE’这一事件进行监听。当对账单被接收后，即系统成功将对账单发送给接收方后，V4 PROFORMA将监听到对账单接收事件，然后根据传递的参数，获得对账单文档id和文档类型，然后对文档相关信息进行合法性检验后。若文档不合法，则直接返回不做后续处理；若通过合法性检验，则调用PROFORMA SERVICE的相应接口对对账单进行后续处理，包括更新对账单状态等。

### 4.3.3 对账单模块实现结果

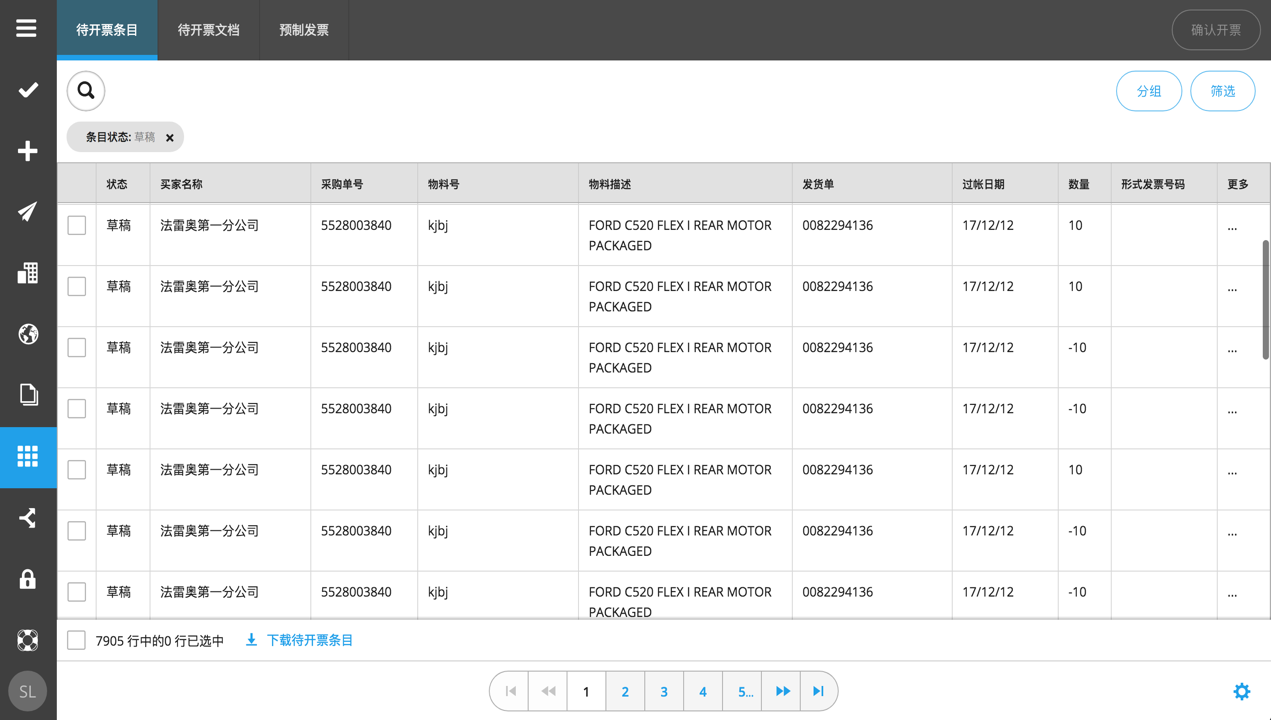


图4.5 对账单列表界面

当采购商成功上传对账单发送给供应商后，供应商可以在待开票数据列表中查看对账单列表，包括待开票条目状态、买家名称、采购单号、物料号、物料描述、发货单、过账日期、数量、形式发票号码等信息。

如图4.5所示，查询状态为草稿的待开票条目，因为条目为草稿状态，未创建成形式发票，所以对应的形式发票号码为空。当开票条目中的数量为正数时，表示发生了进货；当开票条目数量为负数时，表示发生了退货。

## 4.4 形式发票模块

### 4.4.1 形式发票模块设计描述

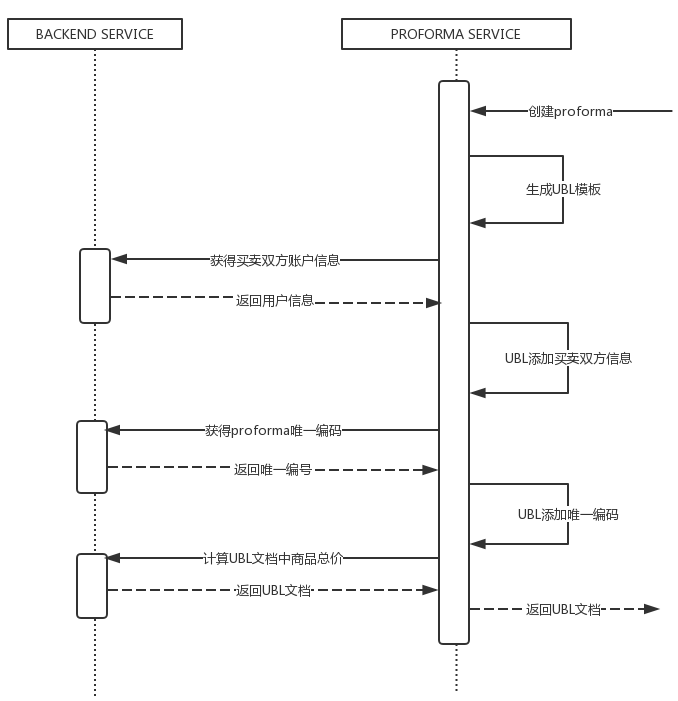


图4.6 生成形式发票时序图

如图4.6所示，为供应商选择待开票数据创建形式发票的时序图。

当用户勾选对账单进行创建形式发票时会调用PROFROMA SERVICE的createProforma接口。createProforma方法中会判断用户是通过哪种勾选方式创建形式发票，包括多行勾选、全部勾选和对账单勾选，解析出需要开票的行列表。PROFROMA SERVICE会根据勾选中的行的lineid列表，根据alongwithkey和transactionid查询与勾选的行拥有相同alongwithkey但还未被其他形式发票占用的行记录，加入到待开票数据行中，在PROFROMA SERVICE中根据商品-物料映射关系，转换映射关系，添加缺省的商品名称、商品价格、税率、税收编码等信息，生成UBL模板。PROFROMA SERVICE向BACKEND SERVICE请求用户账户信息和形式发票唯一编码，将数据添加到UBL文档中。最后，调用BACKEND SERVICE的calculate方法，计算UBL中每一行的商品金额和所有行的总额和税额，并将拼装完成的UBL文档返回。

### 4.4.2 形式发票模块实现代码

Response createProforma(@QueryParam("mode") ProformaCreationMode mode,

CreateProformaDto createProformaDto) {

if (StringUtils.isBlank(mode.getMode())) {

LOGGER.error("Proforma Creation Mode: {} is mandatory.", mode);

throw new ValidationException(requestContext);

}

UBLDocument template;

switch (mode) {

case PICK:

List<ID<InvoiceLine>> lineIdList

= createProformaDto.getInvoiceLineIds();

template = proformaService.createProformaByPick(lineIdList);

case FLIP:

ID<Document> sourceDocId

= createProformaDto.getDocumentId();

template = proformaService.createProformaByFlip(sourceDocId);

case PICK\_ALL:

List<String> documentTypes

= createProformaDto.getDocumentTypes();

template = proformaService.createProformaByPickAll(documentTypes);

case APPLY\_CHANGE:

ID<Document> proformaId

= createProformaDto.getSourceDocumentId();

template

= proformaService.createProformaByApplyChange(proformaId);

}

return Response.ok().entity(template).build();

}

图4.7 创建形式发票接口

如图4.7所示，是创建形式发票的后端接口的具体实现代码。接口根据前端传入的参数来判断创建形式发票的模式，包括勾选部分商品条目进行创建，勾选全部商品条目进行创建，选择完整对账单进行创建，重传对账单进行创建。接口实现会根据不同的创建方式调用对应Service类中的不同方法，并将创建结果返回给前端。

### 4.4.3 形式发票模块实现结果

供应商可以针对整个对账单进行创建形式发票操作，如图4.8所示。待开票文档列表中包括了文档号码、开票类型、序列号、买家名称、文档状态、创建日期等。用户可以根据文档号码查看文档详情并创建形式发票。

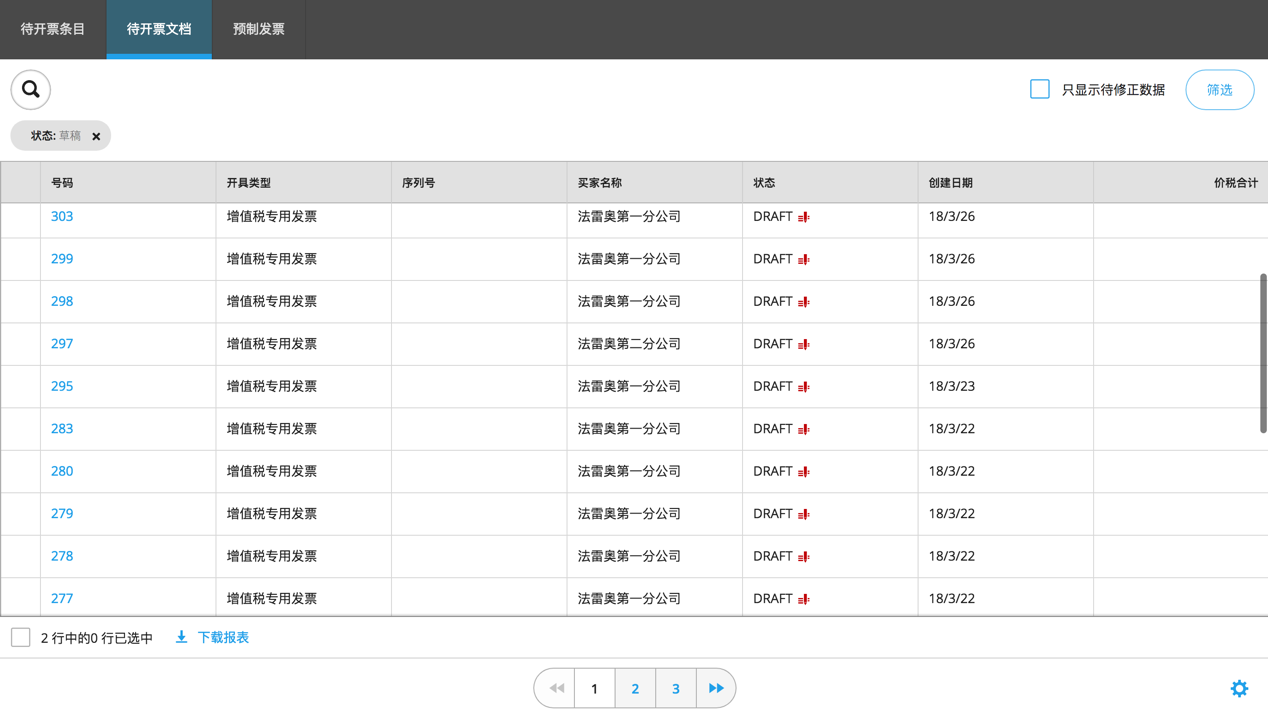


图4.8 对账单文档创建形式发票界面

供应商可以按需要勾选对账单中的部分行创建形式发票，如图4.9所示。用户可以勾选对账单条目中的部分条目，并点击右上角的确认开票按钮进行创建形式发票的操作。其中，“只显示待修正数据”选项表示显示需要修改的对账单文档。当对账单没有通过合法性检验，比如税号、公司名称、公司地址不合法，则需要用户对待开票文档进行修改后才能创建和发送形式发票。

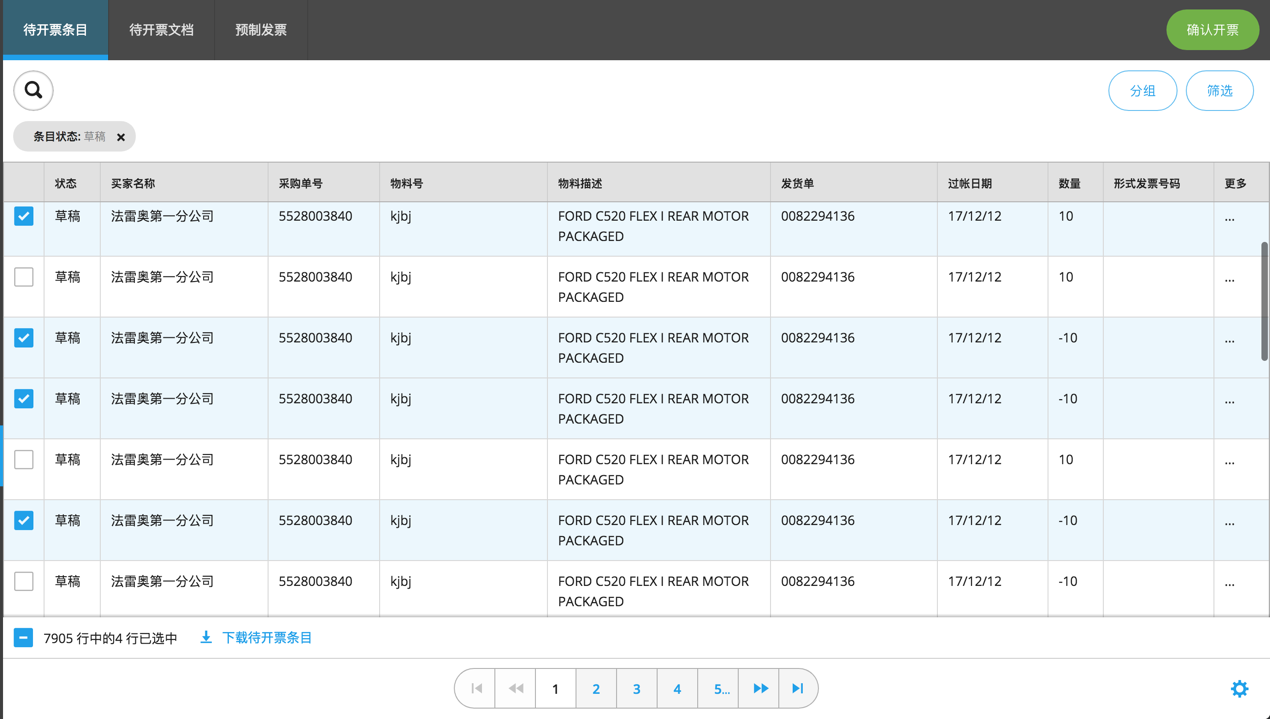


图4.9 勾选创建形式发票界面

## 4.5 价格协商模块

### 4.5.1 价格协商模块设计描述

供应商创建形式发票，定义价格后发送给采购商。当买卖双方对形式发票中商品价格存在差异时，会进入议价流程。图4.10的协商差价时序图，展示了协商差价模块中各个微服务间的一个请求过程。首先，供应商发送形式发票，从PROFORMA SERVICE发送文档到BACKEND SERVICE。WORKFLOW服务会监听BACKEND SERVICE的接收文档接口，当BACKEND SERVICE接收到一份文档时，WORKFLOW服务会拉取文档进行信息匹配，并将匹配结果返回给PROFORMA SERVICE。PROFROMA SERVICE会保存匹配结果，如果供应商在文档中填写的商品价格和WORKFLOW中由采购商定义的商品参考价格匹配失败，则进入议价流程。采购商可以调用PROFORMA SERVICE的接口查看供应商发送的形式发票及相关的匹配失败记录。采购商就匹配失败的记录与供应商进行线下和线上的价格协商后更新价格发送给供应商并存储记录在PROFORMA SERVICE中。供应商对差异价格进行审核和编辑，然后将新的文档发送给采购商。

采购商和供应商之间就价格在PROFROMA SERVICE中进行多次协商，直到双方价格协商一致，审批通过，文档状态被置为协商通过。供应商在查看协商通过的形式发票时，可以将商品价格更新为协商之后的价格，创建新的形式发票并通过BACKEND SERVICE发送给采购商。

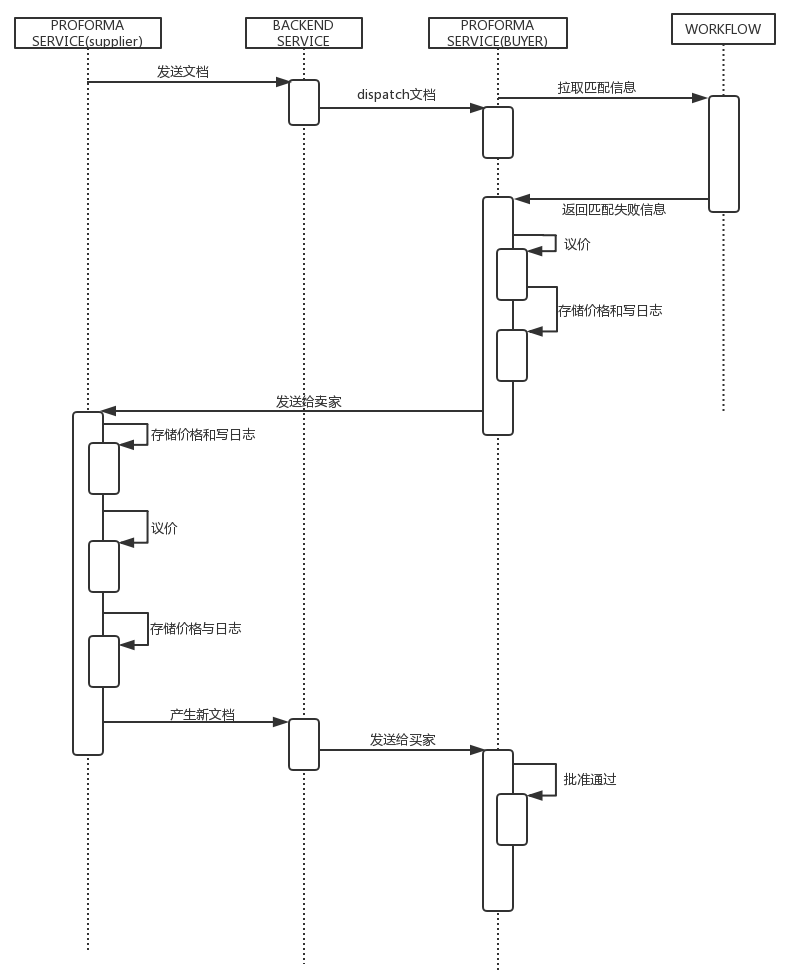


图4.10 价格协商时序图

### 4.5.2 价格协商模块实现代码

采购商和供应商在价格协商的流程中，都会针对一张形式发票上所有有争议的商品价格进行价格确认，然后将修改价格后的形式发票发送给对方。在发送过程中，WORKFLOW模块会对形式发票中的价格进行自动校验，与WORKFLOW中预先保存的参考价格进行对比，系统会保存比对结果和形式发票信息保存至disputedline表中，用于保存最新的价格以及议价历史记录。保存争议信息的具体实现如图4.11所示。

void saveDisputedLines(Proforma proforma,

List<MatchingResult> matchingResults) {

DocumentMetaData metaData =

backendClient.getDocumentMetaData(proforma.getDocumentId());

Invoice proformaUBL=

backendClient.getDocument(proforma.getDocumentId());

if (matchingResults == null || metaData == null || proformaUBL == null) {

throw new ObjectNotFoundException(requestContext, null, null);

}

for (MatchingResult matchingResult : matchingResults) {

DisputedLine disputedLine

= populateDisputedLine(proforma.getId(),proforma.getIid(),

proformaUBL, matchingResult, metaData);

disputedLineDAO.createDisputedLine(disputedLine);

}

}

图4.11 保存议价记录代码

### 4.5.3 价格协商模块实现结果

供应商和采购商会就商品价格进行协商，图4.12显示了用户就形式发票中存在价格争议的商品价格进行协商的界面图，图中的Source Price指的是采购商上传的对账单中商品的原始价格，Price offered by the other party指的是对方给出的协商价格。

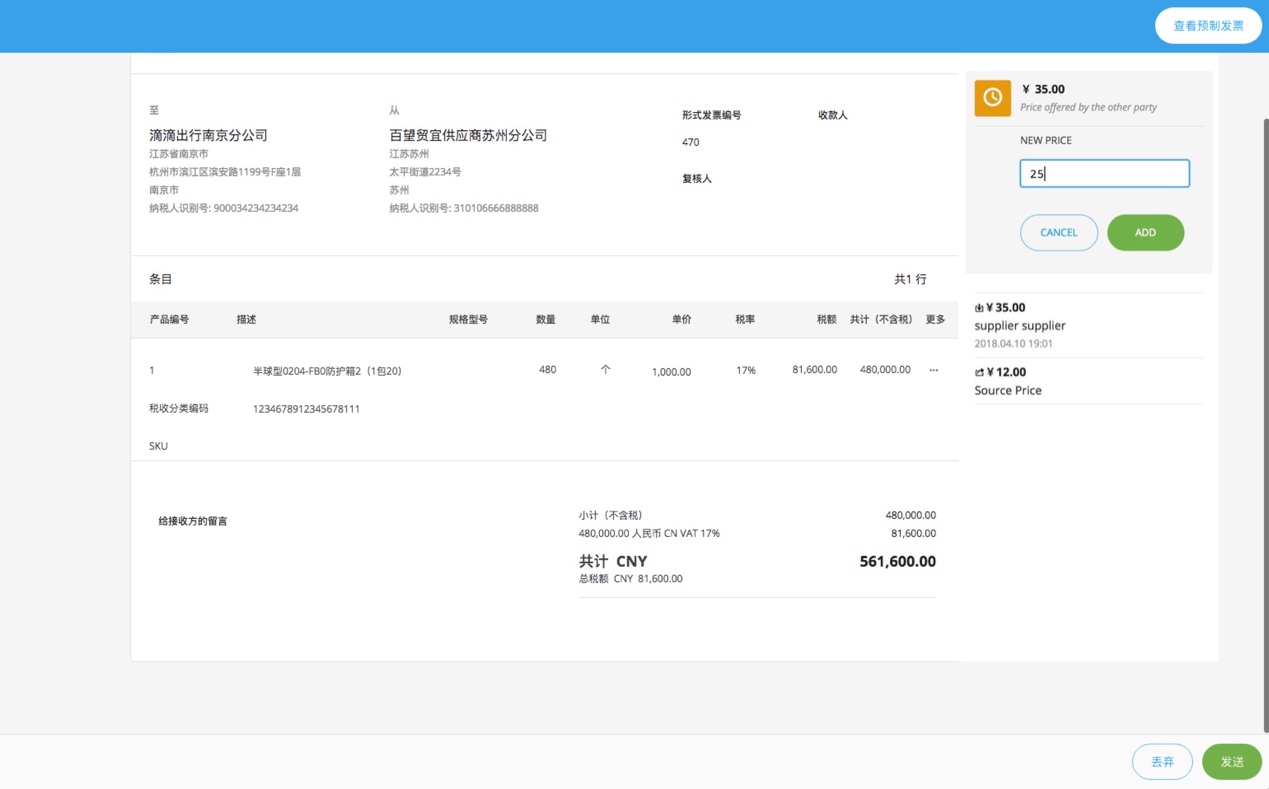


图4.12 价格协商图

## 4.6 预制发票模块

### 4.6.1 预制发票模块设计描述

如图4.13 所示，和拆票模块相关的主要微服务包括PROFORMA SERVIE和BACKEND SERVICE。PROFORMA SERVICE主要处理和电子发票相关的核心业务，BACKEND SERVICE主要实现对文档和相关状态的维护。当供应商对形式发票创建预制发票时，会调用PROFORMA SERVICE的拆票接口。PROFORMA SERVICE会通过文档id向BACKEND SERVICE拉取形式发票文档，然后根据形式发票记录中的一些特定字段，包括group key、merge key等进行拆票操作。首先将记录根据group key进行分组，然后根据merge key将相同的记录进行合并。在分组合并之后，针对每一组的多条记录根据开票限额进行拆票和合并操作。在拆分的过程中，例如开票限额是10万，某一条记录是25万，那么会将记录拆分为10万，10万，5万三张预制发票。在合并过程中，主要用到了贪心算法的思想[武炳杰, 2016]。例如某一组中现在有1，2，3，5，6，8这几条记录，首先将记录降序排序为8，6，5，3，2，1。然后将记录放入一个桶列表中，首先将8放入第一个桶，然后将6从第一个桶开始尝试，如果放入桶中后，桶中的合计金额小于开票限额，则表示可以放入，所以6不能放入第一个桶中，将6放入第二个空的桶中。最后的结果为第一个桶中为8，2；第二个桶中为6，3，1；第三个中为5。每一个桶分别对应于一张预制发票。最后将拆票结果传给Backend Service微服务，由Backend Service对发票文档和发票状态进行维护。

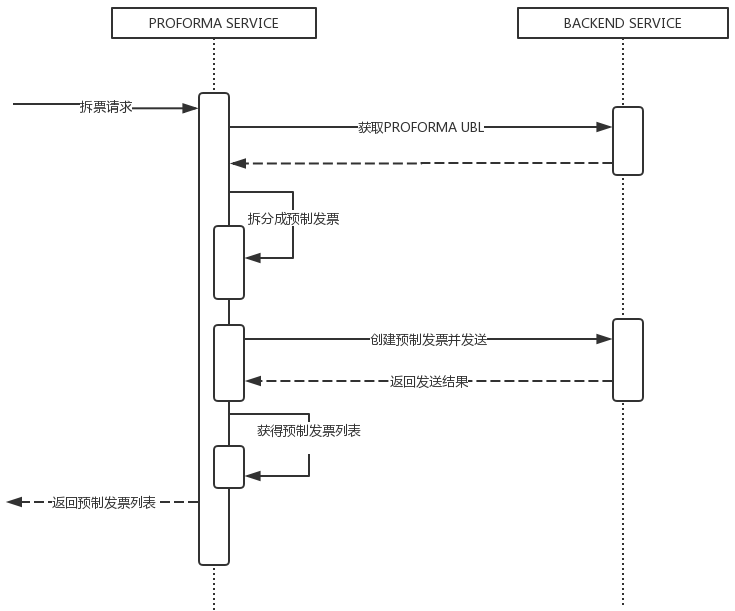


图4.13 拆票时序图

### 4.6.2 预制发票模块实现代码

如图4.14所示，divideProformaLine（）方法主要处理了拆分发票的流程。若一张形式发票中同组商品总金额超过了公司申请的增值税发票的金额上限则在创建预制发票的过程中需要进行拆分操作。根据限额和商品单价计算出每一张预制发票的商品数量上限。然后根据每一张预制发票的数量限额进行拆票操作。

图4.14 拆分发票实现

void divideProformaLine (InvoiceLine line, ArrayList<InvoiceLine> splittedLinelist,

ArrayList<InvoiceLine> noSplittedLinelist, BigDecimal limit,

List<ProformaLineDTO> proformaLines) {

for (ProformaLineDTO proformaLine : proformaLines) {

LineNatureType lineNature

= LineNatureType.getTypeFromName(proformaLine.getLineNature());

if (LineNatureType.GENERAL\_LINE == lineNature) {

generalLines.add(proformaLine);

} else if(LineNatureType.DISCOUNT\_LINE == lineNature) {

discountLines.put(proformaLine.getDiscountLineNumber(),

proformaLine);

}

}

BigDecimal lineAmount = line.getAmount();

ArrayList<InvoiceLine> resultLines = new ArrayList<>();

BigDecimal totalQuantity = line.getQuantity();

BigDecimal Quantity = calculateLineQuantity(totalQuantity, lineAmount, limit);

BigDecimal unitPrice = line.getUnitPrice();

BigDecimal LineAmount =LineQuantity.multiply(unitPrice);

while (totalQuantity.compareTo(eachLineQuantity) >= 0) {

totalQuantity = totalQuantity.subtract(eachLineQuantity);

InvoiceLine dividedLine = line.clone();

dividedLine.setLineAmount(eachLineAmount);

dividedLine.setQuantity(eachLineQuantity);

}

if (totalQuantity.compareTo(BigDecimal.ZERO) > 0) {

InvoiceLine dividedProformaLine = proformaLine.clone();

dividedLine.setLineAmount(totalQuantity.multiply(unitPrice).setScale(2, DEFAULT\_ROUNDING\_RULE));

dividedProformaLine.setQuantity(totalQuantity);

}

}

若一张形式发票中商品金额都小于开票金额，则需要进行合并操作，以减少开出的增值税发票的数目。在对多行商品记录合并之前，首先需要对形式发票中的商品记录进行分组，按照groupkey、mergekey等多个关键词对商品记录分组后，针对同一组的多条商品记录进行合并操作。合并操作的具体实现代码如图4.15。

图4.15 合并发票算法实现

boolean addIntoMockinvoice (InvoiceLine line, List<BigDecimal> amountCapacity,

List<BigDecimal> taxCapacity, List<Invoice> mockinvoices) {

for (Invoice mockinvoice:mockinvoices) {

BigDecimal totalTax = line.getQuantity().multiply(line.getPrice())

.multiply(line.getTaxRate().divide(HUNDRED)).setScale(2,

DEFAULT\_ROUNDING\_RULE);

line.setTotalTax(totalTax);

line.setTotalPriceTax(totalTax.add(line.getTotalPrice()));

}

BigDecimal lineTaxAmount = line.getTaxAmount();

BigDecimal lineAmount = line.getLineAmount();

for (int i = 0; i < fapiaoAmountCapacity.size(); i++) {

if (lineAmount.compareTo(fapiaoAmountCapacity.get(i)) <= 0 &&

lineTaxAmount.compareTo(taxCapacity.get(i)) <= 0) {

amountCapacity.set(i, amountCapacity.get(i).subtract(lineAmount));

taxCapacity.set(i, taxCapacity.get(i).subtract(lineTaxAmount));

mockinvoices.get(i).withInvoiceLine(line);

if (mockinvoices.get(i).getInvoiceLine().size() == 200) {

amountCapacity.set(i, BigDecimal.ZERO);

taxCapacity.set(i, BigDecimal.ZERO);

}

return true;

}

}

return false;

}

### 4.6.3 预制发票模块实现结果

不同的企业能够按照其实际的经营状况向主管税务机关申请开具不同开票额度的增值税专用发票。如图4.16所示，供应商可以在应用库的开票管理应用中设置增值税专用发票开票的开票限额。当用户根据形式发票创建预制发票时，会根据用户设置的开票限额进行拆分，每张预制发票的金额都尽可能大但不超过增值税发票最大金额。

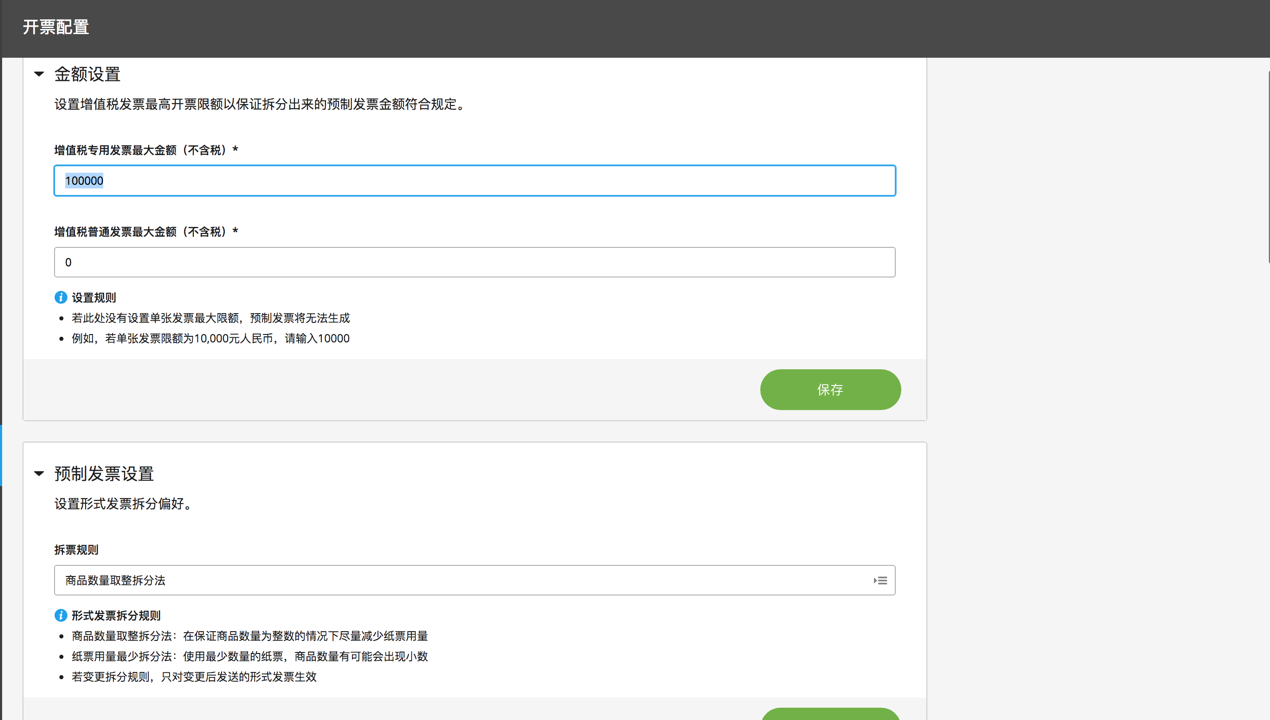


图4.16 设置开票限额界面

如图4.17所示，将形式发票根据预先设置好的开票限额进行拆票，得到预制发票列表，包括发票类型、号码、对应的形式发票号码、状态、买家名称、价税合计等。用户可以针对预制发票创建增值税专用发票，如果创建失败则在预制发票列表中显示“开具失败”状态；如果增值税专用发票开具成功，则会在预制发票列表对应行上显示“已开具”状态，并显示税务开票接口返回的包含发票代码、发票号码和开具日期的发票信息。

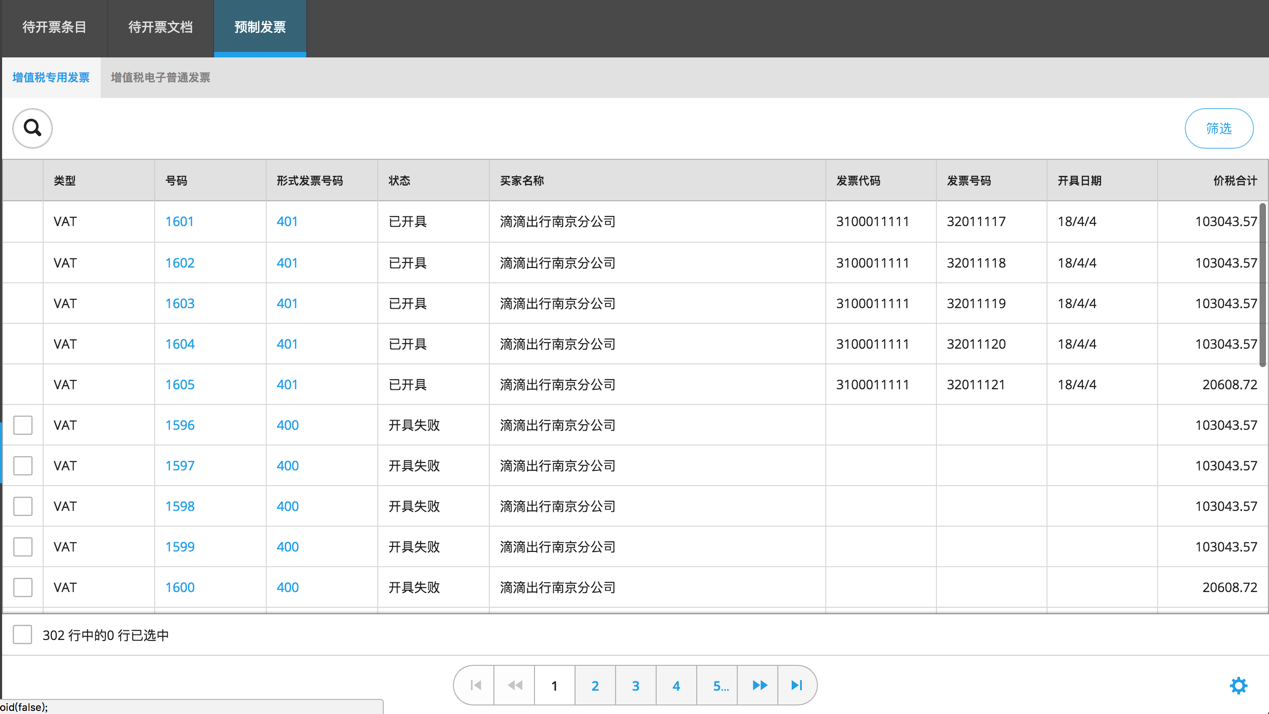


图4.17 预制发票列表界面

如图4.18所示，在预制发票列表中，如果该行状态显示为“已开具”，用户可以根据预制发票号码选择预制发票查看详情。

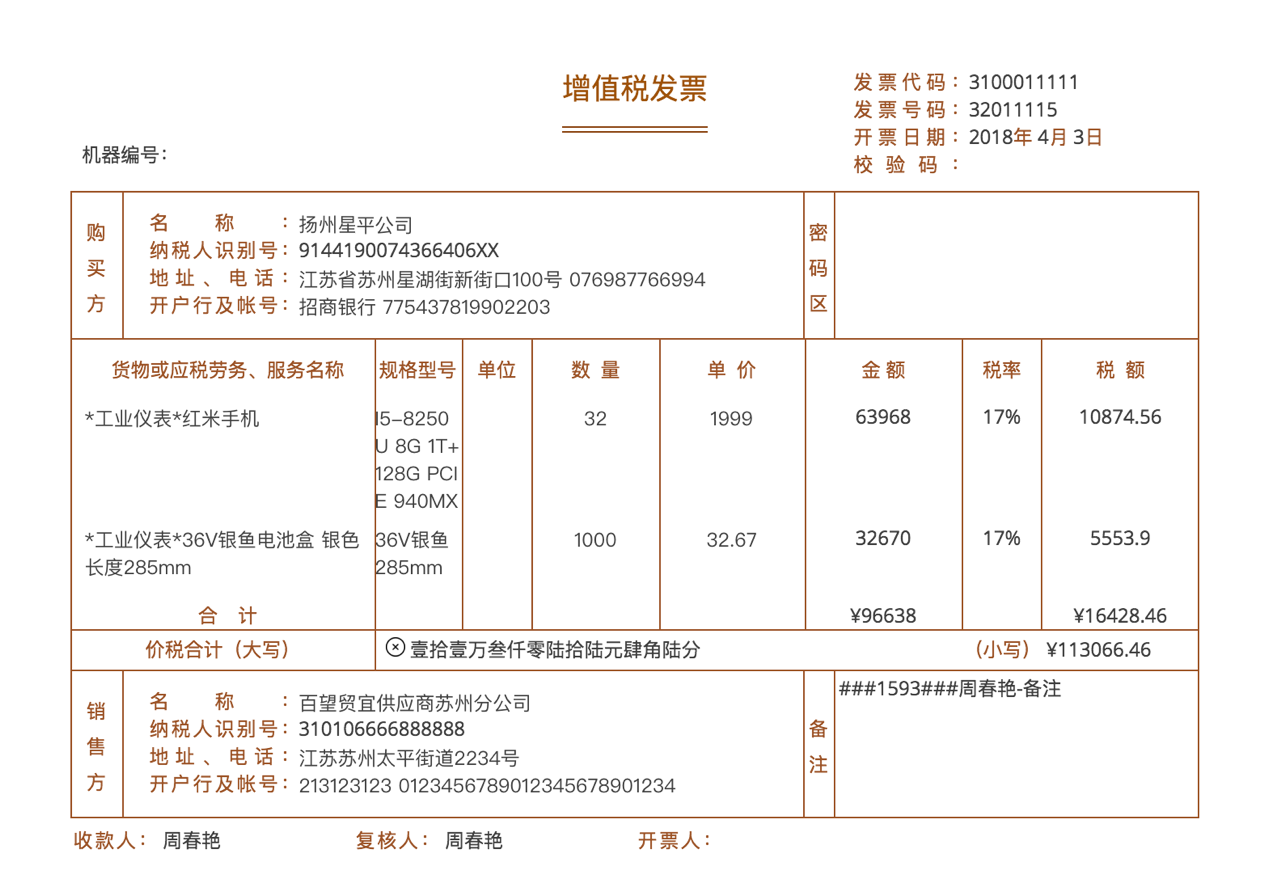


图4.18 预制发票详情界面

## 4.7 本章小结

本章主要讲述了Tradeshift电子发票系统发票生成相关模块的设计和实现，包括了上传对账单、创建形式发票、协商价格、创建预制发票。通过每个模块的时序图，描述了每个模块在不同微服务间，主要包括V4 PROFORMA、PROFORMA SERVICE和BACKEND SERVICE这几个微服务之间的请求和交互方式，并通过部分关键代码来描述了模块的实现过程。系统已经上线并使用，在本章中通过界面截图展示了发票生成相关模块的实现结果。

# 第五章 总结与展望

## 5.1 总结

Tradeshift电子发票系统主要为企业级用户提供上下游的协同服务，帮助不同类型、不同规模的企业实现与上游供应商，下游经销商之间的信息传递的无纸化，保证跨地域服务的实时性，为供应商和采购商提供了从销售记录到增值税正规发票的一系列处理流程。本人有幸参与到了电子发票系统的开发工作，包括上传对账单，创建形式发票，协商价格，创建预制发票等多个电子发票生成相关模块，主要负责功能模块的后台接口的实现。

本文主要基于Tradeshift电子发票系统的设计与实现进行展开，并对电子发票生成相关模块进行了重点描述。本文首先介绍了电子发票相关的行业背景，对传统发票向电子发票的发展过程展开了介绍，并描述了Tradeshift公司以及Tradeshift电子发票系统的项目背景和项目概述，阐述了电子发票系统主要解决的痛点和提供的服务。然后介绍了Tradeshift电子发票系统所使用的数据存储技术和系统开发技术，包括Riak、MyBatis、Restful、OAuth2.0等多种技术。接着描述了电子发票系统的功能需求和发票生成相关模块的分析与详细设计，并对存储发票相关数据的数据库的表设计进行了介绍。最后，本文通过时序图描述了电子发票生成相关模块的实现逻辑，通过部分关键代码描述了电子发票系统发票生成相关模块的实现方式，并通过已经上线系统的部分截图展示了多个功能模块的实现结果。

Tradeshift电子发票系统，通过无纸化的方式对发票信息进行传递，能够有效地节约社会成本、提高买卖双方的交互效率。Tradeshift电子发票系统从功能上被划分为公司信息模块、邮件模块、公司网络模块、对账单模块、形式发票模块、价格协商模块、预制发票模块、直连开票模块、应用库模块。本文详细地描述了对账单模块、形式发票模块、价格协商模块、预制发票模块这四个电子发票生成相关的模块。

## 5.2 进一步展望

Tradeshift电子发票系统目前已经上线，为包括迪卡侬、沃尔沃、浙江大华等多家公司提供发票处理业务，提高企业业务办理和开票的效率，保证跨地域服务的实时性。但在用户使用过程和产品开发过程中，可以发现产品仍然存在着需要进一步完善的地方。

Tradeshift电子发票系统是针对公司隔离的，系统会根据公司需求提供定制化的功能。在之后的开发维护中，需要进一步提高系统的可扩展性，保证当新的需求或者新的公司使用系统的时候能够在不影响其他公司已有功能的基础上，方便地在原有系统上扩展功能。比如，随着电子发票系统的不断发展，接入系统的企业不断增加，系统可以在下一个迭代中，增加对多种不同的待开票数据类型的接收，除了接口上传文档外增加包括FTP方式上传等多种待开票数据上传方式，以匹配不同公司的ERP系统中存储的交易记录。

另外，系统中存储的发票文件指数级增长时，需要不断提高海量数据的查询、更新等效率，提高系统的性能。比如，当用户选择了形式发票状态、创建日期、文档名称关键字进行模糊搜索时，需要保证用户在对数据进行复杂查询时的系统响应效率。

# 参考文献

[Garriga et al., 2016] M. Garriga, C. Mateos, A. Flores, A. Cechich and A. Zunino, RESTful service composition at a glance: A survey*, Journal of Network and Computer Applications*, 2016, 2015.11.020, 32-53.

[Jain et al., 2015] P. Jain, A. Goel and S.C. Gupta, Monitoring of Riak CS Storage Infrastructure, *Procedia Computer Science*,2015, 2015.06.016, 137-146.

[Lei et al., 2015] X.F. Lei，Z. Wang and Y.Z. He，The Data Management and Real-time Search Based on Elasticsearch， *Computer Science and Electronic Technology International Society*，pages 825-829,2015.

[MyBatis, 2018] [http://www.MyBatis.org/MyBatis-3/zh/index.html](http://www.mybatis.org/mybatis-3/zh/index.html) MyBatis maintained by [MyBatis.org](http://www.mybatis.org/),2018.

[SIDDIQA et al., 2017] Aisha SIDDIQA,Ahmad KARIM and Abdullah GANI,大数据存储技术综述（英文）, *Frontiers of Information Technology & Electronic Engineering*，2017， 2017年08期, 1041-1072.

[Spring, 2018] <https://spring.io> Spring maintained by Pivotal Software,2018.

[Tradeshift, 2018] <https://cn.tradeshift.com/press/> Tradeshift maintained by Tradeshift，2018.

[UBL, 2018] <http://ubl.xml.org> UBL maintained by OASIS,2018.

[XIAO et al., 2017] M.H. XIAO，D.L. CHENG，W. LI，Y.N. LI，X.Q. LIU and Y.T. MEI, Formal Analysis and Verification of OAuth 2.0 Protocol Improved by Key Cryptosystems, *电子学报(英文版)*，2017，2017年03期, 477-484.

[戴新竹等, 2016] 戴新，竹黄洵， 税收征管语境下的电子发票应用，*税务研究*，2016，2016年03期，70-74。

[丁春玲等, 2017] 丁春玲，路志强，彭伟， Java反射机制在数据持久层轻量级ORM框架中的应用研究，*西安文理学院学报(自然科学版)*，2017，2017年01期，39-42。

[宏朴, 2016] 宏朴，*基于Lucene的搜索引擎的研究与实现*，硕士论文，大连理工大学数学科学学院，2016。

[李春阳等, 2017] 李春阳，刘迪，崔蔚，李晓珍，李春岐， 基于微服务架构的统一应用开发平台，*计算机系统应用*，2017，2017年04期，43-48。

[李海珍, 2012] 李海珍， 形式发票在外贸实务中的用途及其存在的问题，*对外经贸实务*，2012，2012年12期，74-76。

[刘姚, 2017] 刘姚，基于Spring和OAuth2.0的第三方授权框架，*计算机技术与发展*，2017，2017年03期，167-170。

[罗东, 2010] 罗东，*基于供应链管理的电子采购系统若干问题研究*,电子科技大学信息与软件工程学院，2010。

[吕冬雪, 2016] 吕冬雪, 基于大数据环境的NoSQL技术分析, *电子设计工程*，2016， 2016年14期, 33-36+39。

[潘陶, 2016] 潘陶， 电子发票推广模式浅析，*财会学习*，2016 ，2016年17期，161。

[钱剑锋, 2011] 钱剑锋，*分布式语义发布/订阅系统的若干关键技术研究*，浙江大学计算机科学与技术学院，2011。

[荣艳冬, 2015] 荣艳冬， 关于Mybatis持久层框架的应用研究，*信息安全与技术*，2015，2015年12期，86-88。

[苏跃明等, 2017] 苏跃明，李晨，田丽华， 基于分片一致性哈希负载均衡策略与应用，*计算机技术与发展*，2017，2017年11期，62-65+70。

[王建斌等, 2010] 王建斌，胡小生，李康君，赵靓， REST风格和基于SOAP的Web Services的比较与结合，*计算机应用与软件*，2010，2010年09期，297-300。

[王仲洲等, 2016] 王仲洲，杨晓洪，王剑平，张果， 基于REST风格的WEB API架构研究， *微处理机*， 2016，2016年05期，52-55。

[汪周丹, 2016] 汪周丹， 浅析电子发票的影响，*中国乡镇企业会计*，2016，2016年07期，84-85。

[魏成坤等, 2016] 魏成坤，刘向军，石兆军， OAuth2.0协议的安全性形式化分析，*计算机工程与设计*，2016，2016年07期，1746-1751。

[文欢欢等, 2015] 文欢欢，刘振宇，吴霖，基于Mybatis和JDBC的分页查询研究，*电脑知识与技术*，2015，2015年25期，165-167。

[武炳杰, 2016] 武炳杰,贪心算法在组合问题中的运用, *中等数学High-School Mathematics*，2016， 2016年05期，6-10。

[夏萍等, 2014] 夏萍，张强，叶静 ， UBL在货物运输过程信息交换中的应用，*物流技术Logistics Technology*，2014，2014年01期，318-320。

[新华社, 2015] 新华社，国务院印发《指导意见》 加快构建大众创业万众创新支撑平台，*现代企业Modern Enterprise*，2015，2015年10期，1。

[薛峰等, 2012] 薛峰，梁锋，徐书勋，王彪任， 基于Spring MVC框架的Web研究与应用， *合肥工业大学学报(自然科学版)*，2012，2012年03期，337-340。

[于吉鹏, 2014] 于吉鹏， 复杂结构的消息队列应用研究，*科技信息*，2014，2014年12期，45-46+49。

[詹伟等, 2014] 詹伟，郭峰，董亮，张科，王逸兮， *2014电力行业信息化年会论文集*，106-108，人民邮电出版社电信科学编辑部，2014。

[张晶等, 2017] 张晶，王琰洁，黄小锋，一种微服务框架的实现，*计算机系统应用*，2017，2017年04期，82-86。

[张媛媛, 2010] 张媛媛， *普通发票管理的现状与对策研究*，山东大学政治学与公共管理学院，2010。

[周映等, 2015] 周映，韩晓霞 ，ElasticSearch在电子商务系统中的应用实例，*信息技术与标准化*，2015，2015年05期，72-74。

# 致谢

研究生时代已经接近尾声，在此向老师、同学和家人在我完成论文过程中对我不断地鼓励与支持表示由衷的谢意。

感谢我的家人和同学，是他们在我遇到问题时给予了我鼓励和肯定，让我能够以好的心态解决在实习和论文编写中遇到的困难。感谢在实习期间遇到的同事，通过与他们的沟通交流让我能够更加清晰地了解项目的需求和业务逻辑。在项目的设计和实现中遇到的难题，他们都能热心的为我提供意见和解决方案，帮助我在项目实践中不断学习和提高专业技能。

感谢学院和老师们，学院为我提供了良好的学习环境，让我收获了大量的专业知识，并能够将所学的知识运用到实践中。在老师们的教导下，我才能够非常顺利地参与到项目开发中，并最终完成这篇论文。特别要感谢刘峰老师在我编写过程中对我的指导与帮助，在完成论文时，多次和我交流，热情认真的帮助我解决相关问题。

最后，感谢所有帮助和支持过我的人们，衷心地感谢你们为我做的一切。我祝愿你们在未来的每一天里幸福而健康。同时，还要感谢评阅本论文以及答辩委员会的各位老师，谢谢您们辛劳的付出。

# 版权及论文原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：

日期： 年 月 日