文章编号：2095-2163(2023)03-0093-06 中图分类号：TP311.5 文献标志码：A

基于微服务架构的木材订单系统设计与应用

姚 砺，张海路，徐梦娜，付 帅
（东华大学 计算机科学与技术学院，上海 201620）

摘 要：传统的单体式 ERP 系统开发快、部署简单，但有着内部耦合性高、拓展性差、灵活性低等弊端，随着企业的发展曾经的单体式 ERP 系统已难以满足发展需要。为此，本文在开发某木材大宗商品订单 ERP 系统时，提出了基于微服务架构的设计方案，按照业务功能对微服务进行了划分，并进行了系统整体技术架构设计。系统整合服务注册与配置中心、鉴权中心、微服务网关实现微服务功能，并针对企业需求设计了权限分配策略、使用了分布式事务管理保证数据一致性、使用了负载均衡与服务熔断技术提升了系统可用性。

关键词：微服务；单体架构；Spring Cloud

Design and application of timber order system based on microservice architecture

YAO Li, ZHANG Hailu, XU Mengna, FU Shuai
(College of Computer Science and Technology, Donghua University, Shanghai 201620, China)

【Abstract】 The traditional monolithic ERP system is fast in development and simple in deployment, but has disadvantages such as high internal coupling, poor expansion, and low flexibility. With the development of enterprises, the monolithic ERP system can no longer meet the development needs. In response to this problem, a microservice architecture is introduced to refactor the old system. This paper analyzes the system requirements when developing an ERP system for timber bulk commodity orders. In view of the advantages of low coupling, high expansion and high availability of the microservice architecture, the Spring Cloud microservice architecture is used to reconstruct the system, and to restructure the microservices according to business functions. The microservices are divided by business function, and the overall technical architecture design of the system is carried out. The system integrates service registration and configuration center, authentication center, and micro-service gateway to realize micro-service functions, and designs permission allocation strategy for enterprise needs, uses distributed transaction management to ensure data consistency, and uses load balancing and service fusion technology to improve system availability.

【Key words】 microservice; monolithic architecture; Spring Cloud

0 引 言

随着时代的发展与信息技术的进步，计算机技术在各行各业得到了广泛运用，有力推动了行业数字化产业进程。顺应时代发展的变化，向信息化方向展开变革与创新对所有的企业来说，既是机遇、也是挑战。ERP 系统的应用可以将企业业务实施中的单据记录、数据统计等纸质工作转化为信息化操作，帮助企业进行内部管理，提高企业竞争力。

A 公司作为一家有着跨国木材交易业务的企业，木材商品种类多，总体业务流程复杂，交易金额大、采购订单下达集中且常有业务改动，导致当前传统的单体架构 ERP 系统难以满足企业需要。单体架构将一个应用作为一个整体具有快速开发部署的优势，但是单体架构下系统各模块间的耦合度较高，维护难度大，系统可拓展性低[1]，有着不利于持续开发的劣势。且单体架构下无法对某一模块进行单独的修改和部署[2]，也无法针对某一功能点实现服务器扩容等操作，性能扩展有局限性[3]。

为解决上述问题且满足企业的多变需求，本文在木材订单系统开发中选择使用微服务作为系统的整体架构进行重构，降低了功能模块间的耦合度，提升了系统拓展性，使用了分布式事务保证分布式架构下的数据一致性、针对企业数据安全要求进行了

作者简介：姚 砺（1967-），男，博士，副教授，硕士生导师，主要研究方向：软件测试技术、图像处理；张海路（1998-），男，硕士研究生，主要研究方向：计算机应用、软件开发；徐梦娜（1997-），女，硕士研究生，主要研究方向：计算机应用、软件开发；付 帅（1997-），男，硕士研究生，主要研究方向：计算机应用、软件开发。
通讯作者：姚 砺 Email：yaoli@dhu.edu.cn

收稿日期：2022-04-22

权限分配与权限鉴定设计并为保证系统安全性,使用了负载均衡与服务熔断技术保证了系统的高可用。

1 系统需求分析

系统将要实现从海外采购订单下达到木材海运至国内仓库的整体业务流程管理。订单将区分运营公司、项目与币种,不同币种的单据间还需要进行金额换算,同时不同的订单还将对应不同的运输方式,由此又会有包含陆上卡车运输及海上货船运输在内的不同物流环节参与其中。系统需要负责整个业务流程中大量的单据生成、管理、木材的状态位置去向查询、各环节费用登记、物流过程中的状态变化跟踪。需要做好用户的角色权限、数据权限划分,不同部门之间的员工拥有不同的权限,保证不同用户权限间的数据隔离。同时要考虑市场旺季时系统的访问稳定性与数据一致性问题,以及在订单、运输等业务流程变动时系统的快速改动上线与维护,保证系统的高稳定、高可用、高拓展。木材订单系统功能模块如图 1 所示。

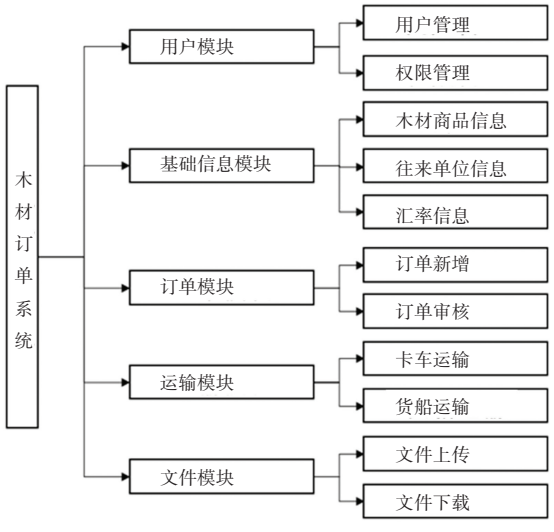


图 1 木材订单系统功能模块

Fig. 1 Functional modules of the timber order system

按照业务功能主要可分为以下模块:

(1)用户模块:负责用户的登录注册、用户基本信息以及用户权限的管理,是用户登录系统的入口。

(2)基础信息模块:负责对业务流程中涉及的木材商品信息、往来单位信息、港口信息、汇率信息等一系列对业务推进至关重要的基础信息的管理。

(3)订单模块:负责原木、板材等各种种类木材采购订单的新增、编辑、审核、统计等操作,不同运营公司的员工对订单需要有权限区分。

(4)运输模块:采购流程中的商品运输功能,包含卡车运输和货船运输两部分。

(5)文件模块:负责上传纸质合同、海关报表及其他相关文件扫描件的留档保存。

2 基于微服务的木材订单系统架构设计

原先的订单系统采用传统单体架构,单体架构即指将一个工程的所有模块的源码配置等内容在一起打包和部署。系统的所有代码都集中在同一个项目中,易于开发,便于管理,本地即可完整启动,所以也更加便于测试。传统的单体架构应用开发速度快、开发成本低,适合应用在项目工程开发初期[4]。随着系统模块增多,代码量越来越大,多个模块相互耦合导致代码结构混乱,代码逻辑难以厘清,拓展性下降。由于耦合度高,模块之间的逻辑边界模糊不清,难以对模块进行有效划分,相互依赖关系较难梳理,出错时难以定位出错点,且单个模块出现问题可能导致整个系统崩溃。同时一旦功能点出现变更,将会很难发现哪些模块会受到影响,不能有效组织测试,给系统的维护、拓展都会带来影响[5]。为解决上述问题引入了微服务作为系统架构进行开发。

2.1 微服务架构简介

微服务 2014 年由 Martin Fowler 正式提出[6]。与单体架构不同,微服务架构将一个完整的应用程序拆分为一套可以独立部署和拓展的小服务[7],当一个服务出现问题就会影响到与之有关联的其他服务,而对系统整体影响较小。不同的服务间相互独立,可以由不同的语言编写,由不同的团队进行管理维护。划分执行的各个服务间应有较为清晰的模块边界且各自负责一个单一的职责与功能,每个小服务在各自的进程中独立运行且各服务间可使用统一的轻量级通信协议相互通信,使其可以组成一个功能完整的系统[8]。各个服务的部署和拓展都可以独立进行,服务间有着清晰的模块边界,具有单一性职责,避免了庞大系统下耦合性过高的问题。微服务架构思想的应用可以使得大型的软件系统更方便地实现业务拓展。

2.2 微服务划分

为降低订单系统耦合度,便于系统的拓展与维护,将微服务按照系统的业务逻辑服务以及系统功能进行拆分。业务服务负责实现采购系统的具体业务功能,系统功能服务则要在抽取出整个系统的通用功能后来确保得到好的效果。微服务的划分要保证各个服务之间的逻辑边界清晰,每个服务负责实

现一个功能模块，各个服务的业务功能划分要明确，确保服务是高内聚、低耦合的[9]，划分的服务之间通过可以跨平台/跨语言的轻量级通信接口实现通信[10]，便于后期拓展与修改。微服务划分见表 1。

表 1　微服务划分

Tab. 1　Microservices division

服务类型	服务名称
系统功能服务	服务注册中心
	服务配置中心
	服务网关
	权限鉴定中心
业务逻辑服务	用户信息服务
	商品信息服务
	单位信息服务
	订单服务
	卡车运输服务
	海上运输服务
	文件管理服务

2.3　系统总体架构设计

系统的微服务框架将使用 Spring Cloud[11] 搭建，约定大于配置的优势，大大简化了微服务系统的开发。Spring Cloud 可以与其它 Spring 项目兼容，快速结合，使得在整合构建微服务系统的组件与框架上更加灵活、快捷[12]。系统总体架构如图 2 所示。对于总体架构的研发设计内容，文中给出剖析论述如下。

（1）功能服务与业务服务。订单系统内的微服务分为系统功能微服务与业务逻辑微服务两大类。服务使用 Spring Boot 框架进行实际功能开发，配合 Feign 实现跨服务远程调用。使用 LogBack 日志框架实现系统操作日志的记录，使用 Spring Security + OAuth2.0 + JWT（JSON Web Token）实现权限鉴定。

（2）服务发现与服务配置。木材订单系统由数个不同的微服务实例组成，且在企业实际应用中根据流量压力情况的不同，服务实例数量也是会动态变化的。每一个服务实例可能会有不同的网络地址和端口号，在服务实例频繁变动的情况下人工管理与配置是不现实的。此时就需要有一个模块对生产者服务与消费者服务实行集中管理，解决服务间的互通问题。

各业务服务与系统功能服务都会有其对应的配置文件，而随着项目的开发、部署环境变化，这些配置文件也会衍生出不同版本。若按照传统的单体架构的模式在服务中管理各自的配置文件，数量巨大的配置文件管理工作将十分棘手，因此一个可以统一管理各种配置、参数配置的中心模块是不可或缺的。

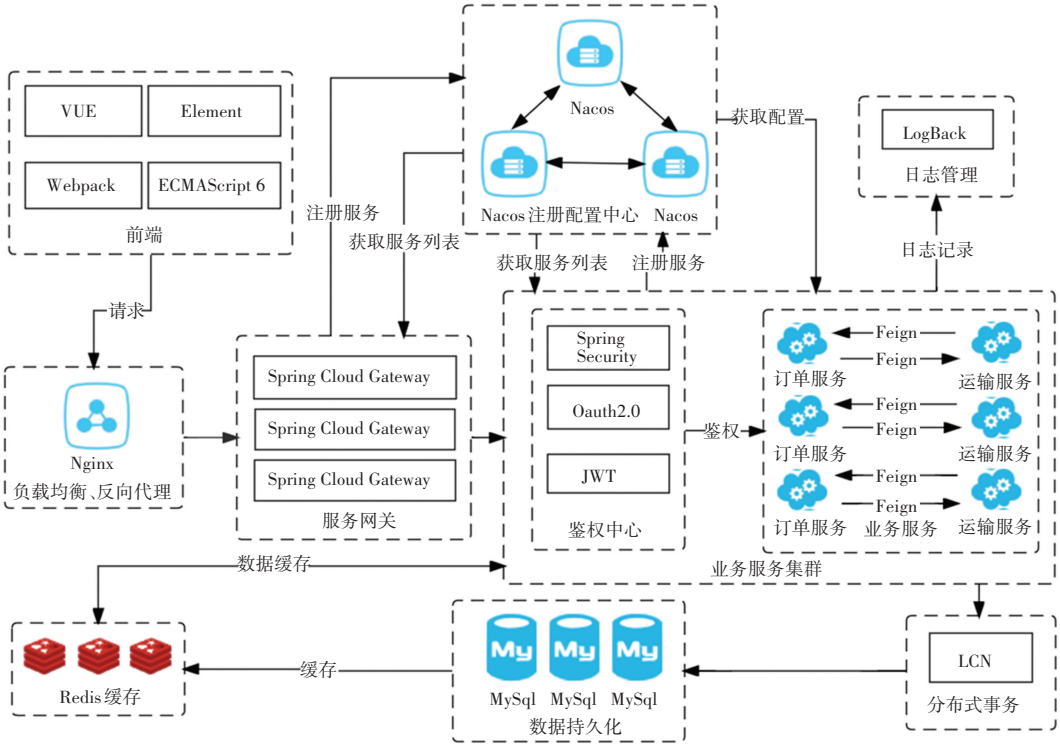
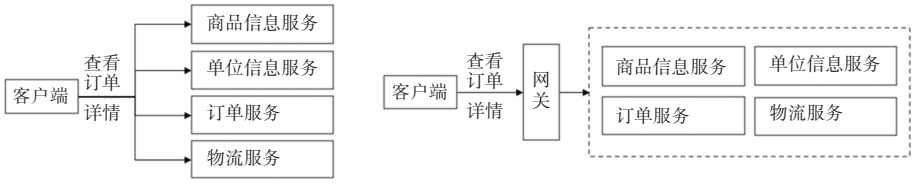


图 2　系统总体架构

Fig. 2　Overall architecture of the system

为解决上述问题,使用 Nacos 实现系统的服务注册与配置中心,以完成微服务的发现、配置以及管理工作,实现服务发现与服务检查、动态配置管理、动态 DNS 服务、服务和元数据管理在内的四大功能。动态管理的特点使得服务注册与配置中心可以动态监听配置参数,发布新配置后不必重新部署,可以使最新的配置应用于相关服务[13],提升了开发效率,减轻了系统维护对企业使用的影响。

(3)服务网关。构建服务网关作为木材订单系统的用户请求入口,负责接收所有外界请求并向相应的微服务进行路由转发,起到将系统内外部隔离的作用。作为流量转发的中心,网关对所有服务的需要对外提供的 API 接口做统一管理,令外部请求只与网关、而非与多个微服务进行交互,降低客户端调用的复杂度。系统中的微服务接口调用详见图 3。



(a) 微服务接口调用 (b) 增加网关后接口调用

图 3 微服务接口调用

Fig. 3 Microservice interface call

本系统选择使用 Spring Cloud Gateway 作为服务网关组件,其基于 Filter 拦截器链的工作方式,使得系统可以在此处拓展限流、鉴权、监控等功能,完善系统的整体稳定性、安全性与可用性。同时,引入 Sentinel 结合网关实现服务熔断,并使用 Nginx 实现系统负载均衡。

(4)远程调用组件。微服务系统为了降低耦合性,微服务之间是相互独立的,但是不可避免地需要相互通信传递信息,此时就需要通过服务间远程调用组件来解决这一课题[14]。

使用 Feign 作为服务间的远程调用组件。Feign 是声明式的模板化的 Http 客户端,在服务生产者创建好相应接口并添加注解后远程服务就可以被消费者调用,使得在调用远程服务时就会更加简单。Feign 默认整合了 Ribbon 组件,使其实现了负载均衡功能。

(5)数据持久化。订单系统的持久层使用 MySql 为单个微服务实现数据的存储,使用 Redis 作为数据缓存中间件,提高数据访问速度,减轻数据库访问负担。使用 MyBatis 持久层框架,简化了 JDBC 操作以及参数设置。为保证分布式系统下的数据一致性,使用 Tx-LCN 框架实现分布式事务管理。

3 系统关键技术

3.1 权限安全设计

由于该企业跨国木材业务订单金额大,有一定数据保密需求,企业内部区分运营分公司,不同部门员工对不同单据的操作权限也不相同,需要做好用户权限隔离与用户身份认证检查,所以就用户权限设计和权限鉴定中心两方面进行了系统的权限安全设计。具体设计可详述如下。

(1)用户权限设计。木材订单系统使用基于角色的访问控制模型[15](RBAC)实现系统的功能权限分配策略,同一个角色拥有相同的功能权限。使用用户-角色-权限的权限设置方式,不直接将权限与用户绑定,而是将权限与角色相关联、再将角色与用户相关联,令用户和权限不直接关联,使一个用户可以拥有被赋予的不同角色的所有权限,有利于权限分配的多样化。用户权限设计框架如图 4 所示。

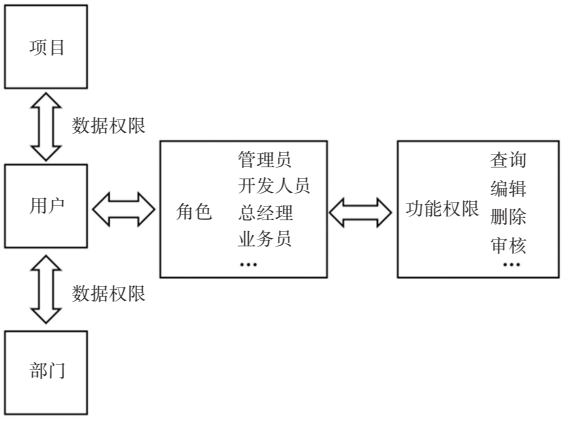


图 4 用户权限设计

Fig. 4 User permission design

用户所在部门职位不同、项目不同,可查看的数据也不同。分部门的数据权限分为全部部门数据、

本部门数据、本部门及下属部门数据、自定义部门数据四种粒度，项目组不做细分。对需要有数据权限要求的数据表中都加入项目 ID 以及部门 ID 两个字段，通过使用 MyBatis 持久化框架下的拦截器对数据进行过滤，实现数据权限的限制。

(2)权限鉴定中心。为保证木材订单系统的数据安全性，使用 Spring Security 框架与 OAuth2.0 协议[16]构建权限鉴定中心微服务，利用 JWT 实现 Token 令牌并使用非关系型数据库 Redis 缓存 Token 令牌，以避免用户登录时频繁访问数据库，减轻数据库压力，实现用户登录以及用户操作时的权限鉴定。权限鉴定流程如图 5 所示。对此可做探讨表述如下。

① 用户登录鉴权：客户端发送携带用户登录信息的请求，由 Gateway 网关获取，并转发至鉴权中心，由 TokenEndPoint 中的/oauth/token 接口接收。将传来的登录信息同数据库中的用户账号密码匹配校验，若成功匹配则通过 Feign 远程调用用户管理微服务，获取角色以及功能权限列表等详细用户信息；若匹配失败则向客户端返回错误提示。此时可查询 Redis 缓存，查看是否存在当前用户有效 Token，若存在则直接向客户端返回 Token，若不存在则调用 createAccessToken 方法生成新 Token 后再返回客户端。

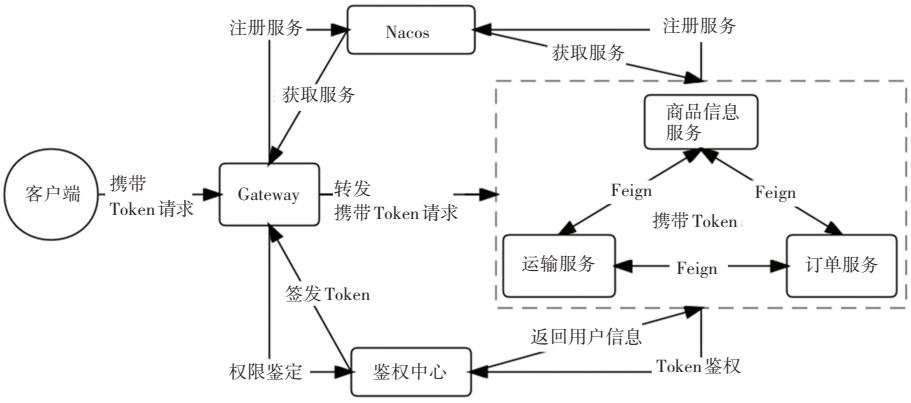


图 5　权限鉴定流程

Fig. 5　Permission authentication process

② 用户操作鉴权：客户端向微服务发送请求时携带 Token 信息，Gateway 拦截到请求后向鉴权中心发送请求，根据 Token 的合法性判断是否为用户的有效请求。鉴权中心将接收到的 Token 同 Redis 缓存中的 Token 进行对比，合法性检查合格后放行，将请求转发到具体微服务。微服务间通过 Feign 进行远程调用时需要权限鉴定的同样在请求时携带 Token，鉴权中心检验其合法性，合格后返回用户角色权限列表。在请求进入 Controller 层接口前判断当前用户是否拥有权限，符合条件的才能进入接口并调用后续方法，否则返回权限错误信息。

3.2　分布式事务

木材采购系统采用了微服务架构，避免了单体架构中的可维护性差、架构扩展性差等问题，按照单一职责的原则划分了微服务，每个服务有各自的数据库、且事务是相互独立的，只在本服务内生效，而一次业务操作中难免同时涉及多个微服务，产生多个独立事务。要想使跨服务业务逻辑最终获得成功，就必须保证跨服务调用的事务与数据一致性[17]。分布式事务处理时序如图 6 所示。

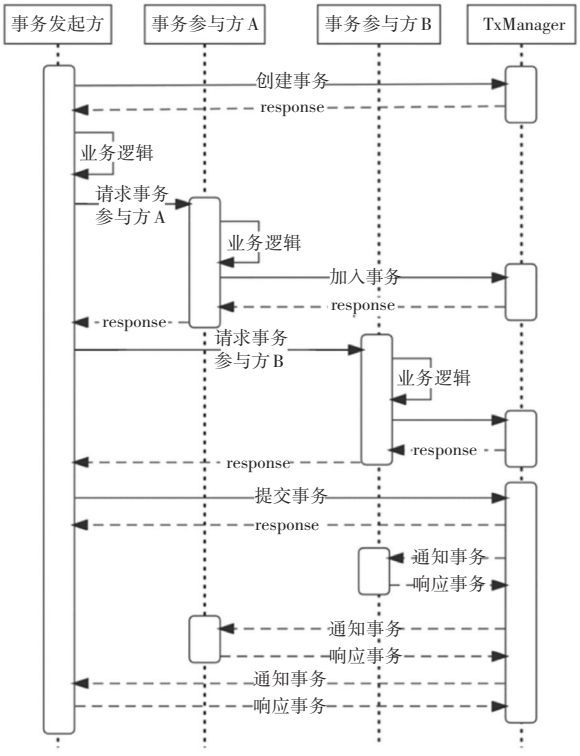


图 6　分布式事务处理时序图

Fig. 6　Sequence diagram of distributed transaction processing

系统使用 Tx-LCN 框架实现分布式事务的管理。事务管理中不创建新事务,而是通过创建并管理所有微服务事务信息的事务组。在事务发起方执行业务代码前,调用 TxManager 模块创建事务组对象,并得到事务标识 GroupId。在事务发起方之后参与的事务都将加入同一个事务组,每一个参与方的事务执行结果都需要通知 TxManager。当所有事务发起方的所有业务代码执行结束后,TxManager 根据事务执行情况通知事务组中的每一个事务参与方提交或回滚事务,并将最终结果返回事务发起方,保障数据一致性。

3.3 系统可用性

木材订单系统在市场旺季时会有大量订单或运输服务的高并发请求,单体架构旧系统无法进行针对性的处理,可能引发服务超时、报错、请求失败等影响企业业务运营的问题。引入微服务架构后,为满足系统可用性采取以下措施:

(1)负载均衡。为了维持木材订单系统后台在高并发情况下的稳定性,需要将客户端流量以资源利用最大化的方式分配到具体服务中,也就是实现负载均衡。在订单系统的网关集群前加入反向代理服务器 Nginx,流量由此转发到网关集群。在 Nginx 中添加服务器网关端口集群,根据网关服务实例的连接数目进行负载均衡,平衡请求数量的分配,保证网关的稳定运行。内部服务间远程调用时在 Feign 中集成 Ribbon 组件,通过轮询机制实现负载均衡。

(2)服务熔断。为避免在服务调用链中因某一环节服务无法正常工作,导致其调用者请求超时、甚至因为请求堆积发生雪崩效应造成整个系统崩溃,此时应该将出故障的服务或接口隔离出来,断绝其与外部访问的联系,直到恢复正常。通过在网关中加入 Sentinel 实现服务熔断以达到目标需求,本系统中设置使用平均响应时间超过阈值 800 ms 时进行熔断。

4 结束语

本文使用 Spring Cloud 微服务框架设计重构木材订单系统。相较于传统单体架构,微服务高内聚低耦合的系统特点更有利于根据用户具体需求对系统做针对性修改,更有利于企业的发展需求。本文根据系统功能以及业务功能划分了微服务,每一个服务负责单一职责,并在此基础上进行了系统整体架构设计,加入了 RBAC 权限分配与权限鉴定中心配合解决系统数据权限问题,保障企业数据安全;通过分布式事务管理保证在微服务架构下的数据一致性;在企业应用中使用了负载均衡与服务熔断,尽力保障采购系统在高并发或其他突发情况下的可用性与健壮性。本木材订单系统已在企业投入使用,并取得良好效果,有效辅助了企业业务运行。

参考文献

[1] THÖNES J. Microservices[J]. IEEE Software,2015,32(1):116.

[2] 代飞,刘国志,李章,等. 微服务技术:体系结构、通信和挑战[J]. 应用科学学报,2020,38(05):761-778.

[3] 杨强根,王晓蕊,马维峰,等. 基于微服务架构的地质灾害监测预警预报系统设计[J]. 地球科学,2021,46(04):1505-1517.

[4] DRAGONI N, GIALLORENZO S, LAFUENTE A, et al. Microservices:Yesterday, today, and tomorrow[J]. Present and Ulterior Software Engineering, 2017(6): 195-216.

[5] 辛园园,钮俊,谢志军,等. 微服务体系结构实现框架综述[J]. 计算机工程与应用,2018,54(19):10-17.

[6] LEWISM F J. Microservices a definition of this new architectural term [EB/OL]. [2014]. https://martinfowler. com/articles/microservices.html.

[7] 冯志勇,徐砚伟,薛霄,等. 微服务技术发展的现状与展望[J]. 计算机研究与发展,2020,57(05):1103-1122.

[8] RICHARDSONC.) Microservices) patterns:) With) examples) in) Java[M]. New) York:PHManning) Publications,) 2018.

[9] 杨秦. 基于微服务架构的云平台服务端的设计与实现[D]. 成都:电子科技大学,2020.

[10] 吴化尧,邓文俊. 面向微服务软件开发方法研究进展[J]. 计算机研究与发展,2020,57(03):525-541.

[11] COSMINA I. Spring microservices with spring cloud [M]// Pivotal certified professional spring developer exam. Berkeley, CA:Apress, 2017: 435-459.

[12] VMWARE INC. Spring Cloud [EB/OL]. [2023]. http://projects.spring.io/spring-cloud/.

[13] 张悦. 基于云原生的微服务开发运维一体化平台设计与实现[D]. 济南:山东大学,2021.

[14] 方意,朱永强,宫学庆. 微服务架构下的分布式事务处理[J]. 计算机应用与软件,2019,36(01):152-158.

[15] 蔡婷,聂清彬,欧阳凯,等. 基于角色扩展的 RBAC 模型[J]. 计算机应用研究,2016,33(03):882-885.

[16] FERRY E, RAW J O, CURRAN K. Security evaluation of the OAuth 2.0 framework[J] . Information and Computer Security, 2015,23(1):73-101.

[17] 周文坤,乔运华,侯佳佳,等. 微服务架构的 ERP 应用系统的优势及挑战[J]. 制造业自动化,2020,42(06):123-124,132.