文章编号：2096-1472(2021)-10-59-04
DOI:10.19644/j.cnki.issn2096-1472.2021.010.014

基于Spring Cloud的工厂可视化管理系统的设计与实现

黄 静，吴 涵

(浙江理工大学信息学院，浙江 杭州 310018)
✉syhj_sy@163.com; wuhan489200@163.com

摘 要：为解决当下工厂管理成本高、效率低等问题，提出并实现了基于Spring Cloud的工厂可视化管理系统。采用Spring Cloud框架，根据工厂管理需求，划分出多个微服务并各自独立开发、部署，微服务之间通过相应的API(Application Programming Interface)进行相互调用，系统中包含Redis缓存、服务熔断等处理机制。采用该系统可实现对工厂进行可视化管理，模块化的设计使得工厂管理更有条理性。经过测试，该系统操作简明，流程明晰，集成度高，线上操作可视化，可拓展性高，可应对工厂管理流程中复杂的业务需求，减少了工厂管理成本，提高了工厂管理效率。

关键词：微服务；Spring Cloud；工厂管理；可视化
中图分类号：TP311.5 **文献标识码**：A

Design and Implementation of Factory Visual Management System based on Spring Cloud

HUANG Jing, WU Han

(*School of Information*, *Zhejiang Sci-Tech University*, *Hangzhou* 310018, *China*)
✉syhj_sy@163.com; wuhan489200@163.com

Abstract: Aiming at high cost and low efficiency of factory management, this paper proposes to implement a factory visual management system based on Spring Cloud. Using the Spring Cloud framework, according to the requirements of factory management, multiple microservices are divided, developed and deployed independently. The microservices call each other through the corresponding API (Application Programming Interface). The system includes processing mechanism such as Redis caching, service fuse, etc.. Application of this system can realize visual management of the factory. Modular design makes the factory management more organized. Testing results show that the system is simple in operation, clear in process, highly integrated; it can be visually operated online and has high scalability. It can deal with complex business needs in the factory management process, so to reduce factory management cost and improve factory management efficiency.

Keywords: microservices; Spring Cloud; factory management; visualization

1 引言(Introduction)

在数字化转型[1]的背景下，工厂的发展规模进一步扩大，竞争愈发激烈，给车间生产管理和人员管理带来了严峻的考验。如何能够更及时、更直观地了解工厂的生产状况以做出合理的管控，从而提高生产效率，增强自身的竞争力，成为工厂急需解决的问题[2]。

本文提出的基于Spring Cloud的工厂可视化管理系统，采用前后端分离的开发模式，以Spring Cloud为框架，构建该系统的微服务架构；采用Vue.js和uni-app框架构建该平台PC端和移动端架构；以微服务的形式将服务端切分成若干个独立的服务进行部署运行，前端发送请求至服务端得到响应数据，再利用Element UI和Echarts相关组件对相应数据进行图表展示。经过测试、部署和试用，该系统运行稳定且具有很强的跨平台性能，提升了工厂管理效率和生产效率。

2 微服务架构(Microservice architecture)

随着互联网技术的不断发展，服务端架构不断更新，

基金项目：浙江省重点研发计划项目(2021C01048)。

开始由单体架构发展至微服务架构[3]。单体架构一般含有数据层、业务层、视图层三部分，即MVC(Model View Controller)结构。采用单体架构，在项目的初期，由于所有的业务逻辑都写在一个应用中，会使得开发、测试和部署变得简单高效[4]。但随着业务的不断扩大、需求不断增加，代码会变得愈发臃肿，导致系统可维护性大大降低，甚至会出现修改一个小功能时，由于所有功能模块的耦合，导致系统崩溃。此外，在代码高度耦合且臃肿的情况下，如何实现最大程度的功能优化也变得棘手。单体架构示意图如图1所示。

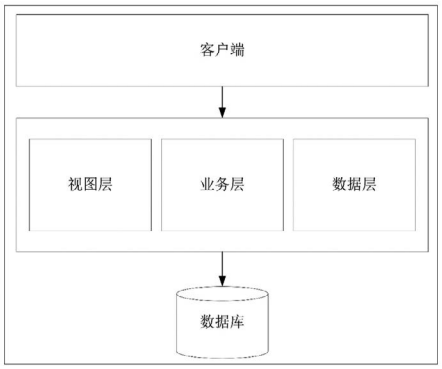

图1 单体架构示意图

Fig.1 Schematic diagram of single structure

微服务架构是近年来出现的一种新的系统开发架构，其核心思想在于通过将业务功能和需求分解到各个不同的服务中进行管理，实现对业务和代码的整体解耦[5]。各个不同的服务可以独立开发、测试、部署及迭代，还可以通过定义完善的API相互通信，每个服务的内部实现细节均对其他服务隐藏。微服务是松耦合的，是有功能意义的服务，且微服务能够使用不同的语言开发。每个微服务都有自己的存储能力，可以有自己的数据库，也可以统一数据库。微服务架构示意图如图2所示。

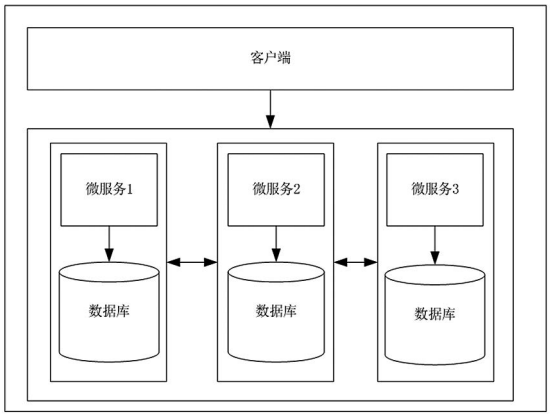

图2 微服务架构示意图

Fig.2 Schematic diagram of microservice architecture

3 系统相关技术选型(System related technology selection)

本系统采用前后端分离的开发模式。后端架构采用微服务的思想进行设计，采用Spring Boot作为后端框架，结合Spring Cloud构建微服务；数据库采用MySQL 5.1.6实现，数据缓存采用Redis实现。前端架构分为PC端和移动端两部分，分别采用Vue.js和uni-app作为开发框架。

Spring[6]框架是当下最流行的轻量级Java服务端开发框架，为传统开发复杂臃肿、耦合度高的问题提供了一套完整的解决方案。但随着互联网产品功能日益复杂，Spring也无法满足开发者的需求，在这样的环境下，Spring Boot应运而生。Spring Boot基于Spring 4.0设计，不仅继承了Spring原有的优秀特性，还实现了自动化配置，大大简化了服务端开发过程，得到越来越多开发者的青睐。

Spring Cloud[7]是分布式服务治理框架，适于高效快速地构建分布式微服务系统。Spring Cloud集成了很多优秀的组件，比如Eureka服务注册与发现组件、Zuul网关服务组件、Ribbon负载均衡组件、Feign服务调用组件、Config配置组件等。这些组件都包含在Spring Cloud中，使用时简单配置即可，大大提高了开发效率。

Redis[8](Remote Dictionary Server)是一个开源的使用ANSIC语言编写的key-value的跨平台的非关系型数据库。Redis基于内存运行，性能高效，支持分布式，理论上可以无限扩展。相比于其他类型的数据库，Redis最受开发者青睐的便是强大的高并发读写能力；当面对海量的数据时，开发者可将短期内不会发生变化的数据存入Redis中，从而减轻数据库的压力，提高系统响应速度，增强用户体验。

Vue.js[9]是一套用于构建用户界面的渐进式框架。与其他大型框架不同的是，Vue.js被设计为可以自底向上逐层应用。Vue.js的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链及各种支持类库结合使用时，Vue.js也完全能够为复杂的单页应用提供驱动。

uni-app[10]是一个使用Vue.js开发所有前端应用的框架，开发者可以只编写一套代码，便可发布到iOS、Android及各种小程序等平台。为了实现一套代码多端发布，并且综合考虑代码编译效率、系统性能等因素，uni-app开发需遵循严格的开发规范，例如页面布局优先采用flex布局等，在此不再赘述。

4 系统设计与实现(System design and implementation)

4.1 系统总体架构设计

本系统总体架构图如图3所示。

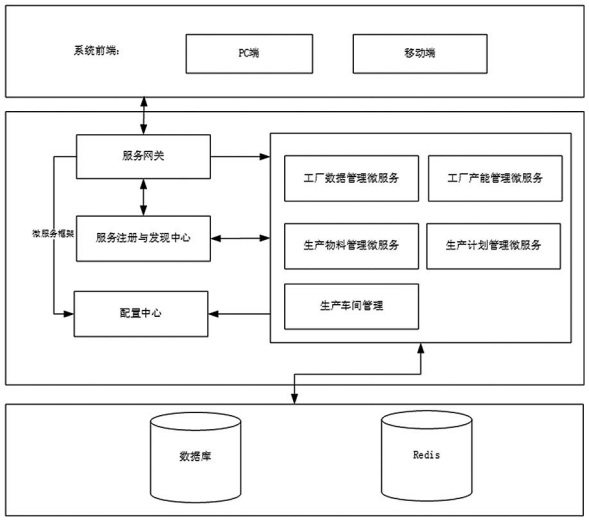

图3 系统总体架构图

Fig.3 System overall architecture diagram

本系统设计主要分为三部分，具体如下：

(1)数据库、Redis缓存：本平台数据库采用MySQL 5.1.6，将工厂管理系统的所有数据都存放在其中。MySQL[11]是一种开放源代码的关系型数据库管理系统，使用者可以根据个性化需求对其进行修改，由于其体积小、速度快、成本低等特点，已成为当下最流行的关系型数据库管理系统之一。本系统将员工、车间、班组、设备、物料、订单、生产工单等所有工厂生产数据信息都存储在数据库中。由于工厂生产过程中所涵盖的数据量会日积月累，愈发庞大，从而导致系统数据访问慢等问题。为了解决该问题，在系统设计过程中，将访问频率高、体量庞大的数据以缓存的形式存入Redis中，并设置相关限定条件，如数据过期时间等，从而实时可控地更新数据，增强数据读写性，提升系统响应速度。

(2)服务端：根据功能需求将服务端进行服务划分，首先构建出基本的微服务框架，包含服务网关、服务注册与发现中心、配置中心；其次根据工厂生产整体流程划分出五个微服务，分别是工厂数据管理微服务、工厂产能管理微服务、生产物料管理微服务、生产计划管理微服务、生产车间管理微服务，各个模块对应工厂管理的各个流程。

(3)系统前端：本系统前端包含PC端和移动端两种，PC端主要采用Vue.js框架、Element UI和Echarts实现，移动端主要采用uni-app框架、uView、uCharts和相关开源组件实现，PC端和移动端以图表的形式完整清晰地展示相关数据。

4.2 系统数据库设计

本系统数据库采用MySQL 5.1.6，在此由于篇幅原因仅对部分库表的结构设计进行说明。数据库部分表设计如图4和图5所示，其中工厂车间信息表(workshop)存放工厂中所有车间信息；设备信息表(machine)存储各个车间中的设备信息；车间班组信息表(team)存储各个车间所辖班组信息；班组人员信息表(employee)存储各个班组中的人员信息；工序工价表(processprice)存储每一道工序的工价信息；工序报工信息表(reportprocess)存储已报工的工序信息；工序派工信息表(assignprocess)存储工序派工信息；生产计划信息表(schedule)存储工厂生产计划信息；产品订单表(productionorder)主要存储所有的产品订单信息；工厂表(plants)存放各个工厂基本信息；公司表(company)存储公司基本信息；用户表(users)存储用户基本信息。plants表和company表是多对一的关系，每个公司下包含一个及以上工厂，两表通过外键company_id关联。plants表和users表是一对多的关系，每个工厂下都有若干个用户，两表通过外键plant_id关联。

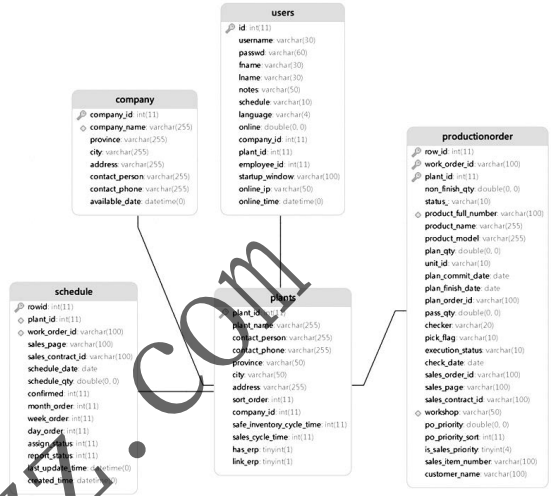

图4 数据库部分表及对应关系图

Fig.4 Part of the database table and the corresponding diagram

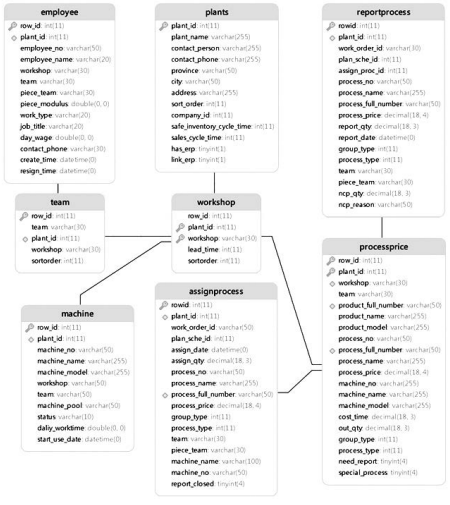

图5 数据库部分表及对应关系图

Fig.5 Part of the database table and the corresponding diagram

4.3 系统服务端设计与实现

4.3.1 微服务框架设计

本系统微服务框架采用Spring Cloud的微服务架构进行搭建。当接口对服务端发起请求，每个请求都必须先经过网关，本系统服务端网关采用的是Spring Cloud全家桶中的微服务API网关Zuul[12]。所有从设备或网站来的请求都会经过Zuul到达后端的Netflix应用程序。作为一个边界性质的应用程序，Zuul提供了动态路由、监控、弹性负载和安全功能。

HTTP和TCP的客户端负载均衡工具采用的是Spring Cloud Ribbon[13]，实现了服务端的高可用，缓解了网络请求压力。

服务调用采用的是Spring Cloud Feign[14]，它是一种声明式的Web Service客户端，简化了微服务之间的调用。在Feign中进行服务熔断的配置，本系统采用Spring Cloud Hystrix[15]实现熔断机制，当系统某个服务发生故障或异常时，会直接熔断这个服务，而不会一直等待该服务超时，防止整个系统发生故障[16]。

配置中心[17]采用的是Spring Cloud Config，它是一个解决分布式系统的配置管理方案，是微服务架构中不可或缺的一部分。如果在微服务架构中不采用Spring Cloud Config作为配置中心会导致配置文件分散，不利于维护等问题。采用Spring Cloud Config后，可以根据需求动态修改各种配置参数，且修改实时生效。

服务注册与发现中心采用的是Spring Cloud Eureka[18]，它是Netflix开发的服务发现框架，本身是一个基于REST的服务，主要用于定位运行在AWS域中的中间层服务，以达到负载均衡和中间层服务故障转移的目的。

数据缓存采用Redis[19]，它是一个开源的高性能的key-value非关系型数据库。Redis支持数据持久化，可以将内存中的数据存入磁盘，再次加载时可以使用；Redis具有丰富的数据类型，除了简单的key-value类型，还包含string、list、hash等数据类型；Redis读数据的速度可达到110,000 次/秒，写数据的速度可达到81,000 次/秒。在本系统中，运用Redis实现业务数据的缓存，将体量庞大且短时间内不会发生变化的数据存入其中，减小了数据库压力，提升了系统响应速度，增强了用户体验。

4.3.2 微服务功能设计

根据工厂生产管理整体流程，本系统划分出五个微服务，分别是工厂数据管理微服务、工厂产能管理微服务、生产物料管理微服务、生产计划管理微服务、生产车间管理微服务，以下为各个微服务具体的功能描述。

(1)工厂数据管理微服务

服务消费者可通过调用工厂数据管理微服务，在数据库中查询出对应的员工、车间、班组、设备等信息并返回给前端进行展示；还可接收前端对相应信息的编辑修改请求，将符合要求的数据写入数据库并完成数据库的更新。

(2)工厂产能管理微服务

服务消费者可通过调用工厂数据管理微服务，在数据库中查询出对应的生产车间、班组、员工的工作日历信息，也可根据前端对工作日历做出的相应编辑修改操作，将接收到的编辑修改数据写入数据库并完成数据的更新。此外，通过调用工厂产能管理微服务还可查询到工厂的产能信息并返回至前端进行展示。

(3)生产物料管理微服务

服务消费者可通过调用生产物料管理微服务，在数据库中查询出工厂的物料库存数据并返回这些数据，供前端展示。

(4)生产计划管理微服务

服务消费者可通过调用生产计划管理微服务，在数据库中查询出销售订单、生产工单、历史生产计划等信息并返回；服务消费者还可通过调用生产计划管理微服务，实现对物料的齐套计算，以及生产月计划、周计划、日计划生成功能，生产计划管理微服务会将这些信息写入数据库，供前端查询展示使用。

(5)生产车间管理微服务

服务消费者可通过调用生产车间管理微服务，在数据库中查询各个车间下的生产任务进度、生产计划、工序派工和报工等信息，并返回至前端进行展示；另可根据前端相关数据的编辑设置操作，将相应数据写入数据库并完成数据库的更新。

4.4 系统前端设计与实现

PC端和移动端共设计了五个模块，分别是工厂数据管理、工厂产能管理、生产物料管理、生产计划管理、生产车间管理。

4.4.1 PC端设计与实现

PC端采用Vue.js框架设计，使用Element UI编写PC端静态页面，图表采用Echarts组件生成。PC端页面路由采用Vue Router实现，状态管理采用Vuex。各个模块功能设计如下：

(1)工厂数据管理

工厂数据管理主要包含员工、车间及班组、设备三个子页面。其中员工子页面中可以查看所有员工入职、离职、所属车间、班组等信息，且信息可编辑修改；车间及班组子页面可以查看工厂下所有车间及班组信息，且信息可编辑修改；设备子页面可查看工厂所有设备的运行情况、负荷等信息，且可手动添加设备。

(2)工厂产能管理

工厂产能管理主要包含工作日历和产能列表两个子页面。其中工作日历子页面可以查看对应车间、对应车间下的班组、对应班组下的员工或设备的工作时间安排，还可以手动调整这些时间安排；产能列表子页面可查看对应车间、班组、员工、设备的产能信息。

(3)生产物料管理

生产物料管理实现对工厂生产物料库存信息的监控，主要包含物料库存状态看板和物料库存水位看板两个子页面。

(4)生产计划管理

生产计划管理主要包含销售订单状态看板、生产工单状态看板、计划历史和工作台四个子页面。销售订单状态看板、生产工单状态看板、计划历史三个子页面实现对销售订单、生产工单、计划历史的详细信息的查看；工作台子页面实现物料齐套计算和月计划、周计划、日计划任务下达确认功能，当计划下达确认后生成相应的生产计划。

(下转第58页)