华北电力大学

毕 业 设 计(论文)

、

题　　目　　基于 Vue 和 SSM 的配煤管理系统开发

院　　系　　动力工程系

专业班级　　能源与动力工程专业

　　吴仲华 1701 班

学生姓名　　王艺娴

指导教师　　王春波

二〇二一年六月

基于 Vue 和 SSM 的配煤管理系统开发

摘要

配煤技术作为火电企业合理规划燃料采购、提高生产安全性和运行稳定性的一项关键技术，已成为其可持续发展的重要一环。因此，如何科学、高效地规划配煤方案对火电企业的发展就显得尤为重要。

论文首先通过分析火电企业对配煤管理系统的需求，明确了配煤管理系统应具备的重要功能，并完成了系统数学模型的建立和模块的划分；其次介绍了当前互联网应用主流的前后端框架，并通过引入如 Vue、Spring Boot 等优秀框架，设计了一套前后端分离的 B/S 配煤管理系统；同时引入数据可视化思想，利用 Echarts 可视化图表库为工业应用增添活力，为企业工作人员提供更好的数据理解服务；展示了基于两种单煤的配煤方案的形成及同等环境方案的日供电成本对比，验证了本系统的功能有效性。

论文对配煤管理系统的设计与实现、以及对不同配煤方案的日供电成本的研究，能够实现企业对配煤管理系统网络化、可视化的需求，对企业的配煤技术发展具有借鉴意义。

关键词：配煤管理系统；前后端分离；数据可视化

DEVELOPMENT OF COAL BLENDING MANAGEMENT SYSTEM BASED ON VUE AND SSM

Abstract

Coal blending technology as a key technology for Thermal Power Enterprises to rationally plan fuel purchase, improve production safety and operation stability, has become an important part of its sustainable development. Therefore, how to plan the scheme of coal blending scientifically and efficiently is very important to the development of thermal power enterprises.

Firstly, through analyzing the requirement of coal blending management system in thermal power enterprises, the main function modules of the system are defined. Secondly, the front-end and back-end frame of current Internet application mainstream is introduced, by introducing excellent frameworks such as Vue and Spring Boot, a B/S coal blending management system with separate front and back ends is designed. At the same time, the idea of data visualization is introduced, and the visual chart library of Echarts is used to add vitality to industrial applications. It provides better data understanding service for enterprise staff, demonstrates the formation of coal blending scheme based on two kinds of single coal and the comparison of daily power supply cost of the same environmental scheme, and validates the functional validity of the system.

The research on the design and implementation of coal blending management system and the daily power supply cost of different coal blending schemes can realize the requirement of network and visualization of coal blending management system, it can be used for reference to the development of coal blending technology in enterprises.

Keywords: Coal blending management system; Fore-and-aft separation; Data Visualization

目　　录

1 绪论

1.1 课题背景

电力企业作为国民经济发展的基础和保障，不仅在疫情之下保证了人民的日常用电需求，更为今后的生产力恢复和壮大提供新鲜血液。但随着社会用电需求的不断增大，以及“厂网分家，竞价上网”、“自主经营、自负盈亏”等能源政策改革的推行，如何在确保一定发电质量的前提下尽可能降低发电成本，已成为电力企业生存发展的核心问题。

煤炭作为我国的第一大能源，在一次能源生产中占比超过 75%，一次能源消费张总占比超过 70%，而“富煤、贫油、少气”的能源结构特点也决定了在较长时间内我国以煤为主的能源生产活动。其次，我国的煤炭储量虽然丰富，但煤质参差不齐，燃烧特性相差悬殊，单一煤种在同一燃烧条件下生成的燃烧产物成分不同，形成的燃煤污染物排放水平也不同。同时，由于煤炭市场和电力市场的作用，电力企业难以在合理的成本范围内购入单一煤种进行发电，而只能采取多种煤掺烧的方式进行发电[1]。

近年来，国民经济的高速发展和人民生活水平的显著提高直接导致了我国电力需求的急剧增加，随之而来的环境污染问题也愈发严峻。权威数据显示，我国大气污染物的主要来源就是煤炭燃烧。火电厂不合理的掺烧方式不仅不能保证生产过程连续、稳定地进行，还会降低发电效率，导致更多的污染物的排放。因此，如何在合理的采购成本和发电环节安全稳定经济地运行的条件下，合理、科学地规划燃煤配比方案和企业的燃煤库存，是发电企业需要面对和不断挑战的问题。

“十三五”以来，国家积极推进煤炭消费革命，大力发展燃煤超低排放技术。“十四五”更是实现未来碳达峰、碳中和的关键期。为响应国家对燃煤污染控制的号召，火电企业加强了在配煤决策方面的研究投入，以降低生产成本，提高企业核心竞争力。“配煤”指将若干不同性质的煤按照一定标准的比例进行混合，使混煤在燃烧时服从一定的燃烧规律，减少燃烧过程中氮氧化物、硫氧化物等污染气体的生成，并降低结焦结渣的概率，从而符合锅炉的设计，使燃烧过程更加安全稳定，保证燃煤发电的经济性，降低燃煤污染物的排放，实现燃煤的超净排放。

随着人工智能、大数据、物联网和云存储等高新技术在火电企业中投入应用，智能化的配煤系统已经成为火电企业现代化建设不可或缺的一环。如何使系统的使用更简便、决策更合理，响应更迅速是国内外燃烧管理学者研究的重点课题。

1.2 研究现状

1.2.1 配煤管理系统的国外研究现状

目前，由于配煤技术的发展和国际贸易的加强，国外发达国家多采用优质煤进行燃烧发电，火电企业的发展重心集中在提高配煤掺烧效率、优化煤仓存煤管理和降低燃煤污染

物排放上。

在配煤掺烧方面，学者们主要研究混配比例和配煤决策评价体系。德、美、日等国家首先提出配煤掺烧，并各自取得了较大的技术突破。之后，欧洲多个发达国家、韩国、加拿大也各自掀起了发展配煤技术的浪潮。

德国的配煤研究重点关注混煤的燃烧特性、适合不同混煤的燃烧设备结构。如德国的V.R.Kuh[2]研究了混煤的着火性、可磨性以及配煤掺烧对电厂运行的经济性；J.Zelkowski[3]等研究了不同配煤比例在不同氧浓度下的着火特性。

美国的工业发达，燃煤需求量大，但排污成本也较高。因而企业的研究方向集中于不同的配煤比例对燃煤污染物排放量的影响上。美国开发的 Compliance Advisor 软件，可以根据灰粒获取 SO_2 的能力，做出配煤决策以减少污染物的排放。美国电力工程协会(EPRI)研发出的煤炭质量评估系统(C-QUEL) [4]可以使电厂工作人员在燃烧发电之前，根据系统预测结果，较为准确地获知掺烧过程中设备运行的稳定性、污染物排放量以及发电效益。美国的 Pennsylvania 电厂通过采用 Praxic 公司开发的电厂配煤优化专家系统，每年节约了100~200 万美元的生产成本。

日本由于煤炭资源贫乏，因此更加注重配煤对发电效率的影响。日本石川岛播磨重工业公司 对比研究了单一烟煤和低挥发分混煤的燃烧特性，并在此基础上设计出了适合低挥发分混煤燃烧的 530T/h 功率的锅炉。日本的 Kawasaki 钢铁公司研发的配煤优化专家系统，在两小时内就完成了专业人员需要四天时间才能完成的配煤决策计算，极大地提高了企业生产效率。

欧洲的火力发电企业通常直接燃烧在港口或发电厂配置好的混煤，而混煤的质量主要由煤的特异性决定。因此，A.N. Alekhnovichd[5]等认为，配煤技术是一项重要的、具有经济和生态优势的先进技术。获取实时准确的煤质信息，是配煤决策中的关键一环，也是混煤进行高效、可靠性燃烧的重要保证。

同为煤炭资源匮乏地区的韩国，重点发展了低品质煤的掺烧特性，Yong- Gyun Kim[6]等在研究了低品质煤的燃烧和排放特性，分析了掺混方式对燃烧产物的影响后发现：配煤掺烧对改善低品质煤的利用率，提高发电经济效益具有显著作用。这也为低品质煤在火电企业的应用提供了理论依据。

此外，加拿大通过合理配煤，调节了燃料的燃烧特性和飞灰特性；英国 IEA 煤炭研究所通过研究煤质对 12 个国家 60 多个电厂的影响，明确了配煤技术在火电企业可持续发展中的必要性；西班牙的 Joaquin Ganzales Blas[7]发现，高挥发分和低挥发分煤混合燃烧可显著降低锅炉结渣和磨煤机磨损率，从而提高发电设备的寿命和发电效率，节约发电成本。

1.2.2 配煤管理系统的国内研究现状

20 世纪末期，我国一家重要的煤炭洁净燃烧技术中心通过实地考察，发现了火电企业

在进行选煤、调节煤种配比、计算投资成本等过程中存在的问题。此后，在中心工作人员大量的理论推导和技术实践下，一项创新性的 IOBSC 技术[8]诞生了。这项技术不仅支持不同用户对煤质的不同需求， 同时满足了火电企业在生产活动中对机组运行稳定性和运行工况可调节性的刚需，降低了电厂的生产经营成本。

煤炭科学院北京煤化所依据煤炭的元素分析、工业分析等数据，把混煤的约束条件转化为目标函数需满足的线性方程，构建出了配煤优化方案的数学模型，并以此开发了基于 DOS3.3 的操作平台、 FOXBASE 开发平台的软件。但由于没有考虑煤炭在燃烧过程中的结渣程度、燃烧完全程度和用户对软件交互界面的适应性，此款软件已很少被使用[9]。

戴财胜[10] 开发了一套基于 Delphi 语言的动力配煤系统，此系统可按照煤仓中煤种信息和配煤参数指标，快速准确地计算出满足定制要求的配煤方案，确保了煤企效益的最大化，且已应用于株洲洗煤厂的配煤决策中。马革非[11]利用 Borlandc++ Builder 语言开发工具，开发了应用于某煤炭运销公司的配煤优化系统，并取得了较好的经济收益。

靳智平等人[12]根据山西部分地区电厂和煤矿企业的配煤需求，开发了一套基于 VB 6.0 语言的动力配煤软件，现已经投入使用并取得了实际效益。

梁景坤等人[13]通过将 Microsoft Visual Basic6.0 与 Visual Fortran5.0 结合,在 Windows 9x/2000、 Windows NET4.0 平台上设计了一套实时监测煤燃料变化的软件。通过在系统运行前设定发电成本和效益目标，系统可自动监控煤仓中总体煤质的变化，计算出当前最优的配煤方案并推荐给工作人员。该软件可进行包括配煤比例、成本、效率在内的诸多数据计算，现已被锡林浩特二电厂正式采用。

山东科技大学研发的基于 Microsoft Visual Foxpro6.0 平台的动力配煤优化专家系统，集成了煤种信息库、配煤专业库、权威方案库等诸多数据库，并以此开发了煤种信息管理、煤质计算、个性化配煤、方案查询等主要功能, 为火电企业的配煤生产及管理提供了科学、实用、简便的工具。

邓俊等人[14-16]研发了“焦炉配煤专家控制系统和集散控制系统”。该系统不仅能以接近 90%的准确率预测部分运行数据，还搭建了许多动态模型供企业调整系统参数。如系统的 Server 端搭建有煤炭质量预测模型、配煤比例计算和配煤比例优化模型，在系统 Server 端始终保持运行的状态下, 这些模型可以在数据库与 Client 端之间进行实时的数据交换。系统采用付费数据库 Oracle9i, 开发语言为 Visual c++6.0。Client 端包含账户管理、信息查询、配煤方案优化、参数设置、数据输出等功能，用户可使用多线程技术, 实现多用户同时访问服务器。目前，该系统已正式被湖南某焦化厂采用。

北京科技大学的李胜[17]基于回归模型，建立了可随时间变化的、煤种质量参数与焦炭质量参数的数学函数，该函数体现了系统的实时性。使用 Visual Fox Pro 开发系统软件,通过把有关技术参数指标、对应的数学模型、经济模型建立起来, 形成了实时动态的炼焦优化配煤系统，为焦化厂应用炼焦的配煤原理、焦炭质量的预测分析技术提供了较为全面的

方法。该系统使包钢焦化工厂由过去单纯的凭借经验进行配煤，逐步转变为信息化、现代化方法进行高效配煤的企业。

1.3 课题目的和意义

火电企业作为国家能源供给的支柱，需要时刻注意自身的发电耗损比和污染排放水平。配煤技术虽能有效帮助火电企业关注资源消耗、发电效益和污染水平，但这需建立在时实有效的煤场数据管理和燃煤分析计算上。因此，围绕量、质、价信息对配煤发电进行精细化和智能化的云端管理，对火电企业完善管理体制、保证生产过程安全稳定经济运行具有深远意义。

尽管国内已有针对电厂进行混煤燃烧开发的配煤优化系统，但多数系统只支持 PC 端的访问，且对操作系统具有一定的要求，需用户提前安装一些专门的软件包，因此对用户有一定的使用门槛，不能很好地实现交互需求。而越来越普及的互联网应用无疑是一个创新性的突破口，既能给予普通用户良好的使用体验，又方便高级管理员管理系统数据，保证一定的数据安全，对配煤优化系统的发展具有重要意义。

2 企业配煤系统需求分析

2.1 企业配煤流程

目前企业的配煤生产过程中，配煤管理人员需要依次进行煤种信息检索、单煤选取、人工计算和校验混煤煤质数据、手动存储混煤数据、单方案的生产效益评估、手动设置配煤方案参数并调优、对比不同参数值下的效益，并最终确定最优方案等过程。具体操作过程如图 2-1：

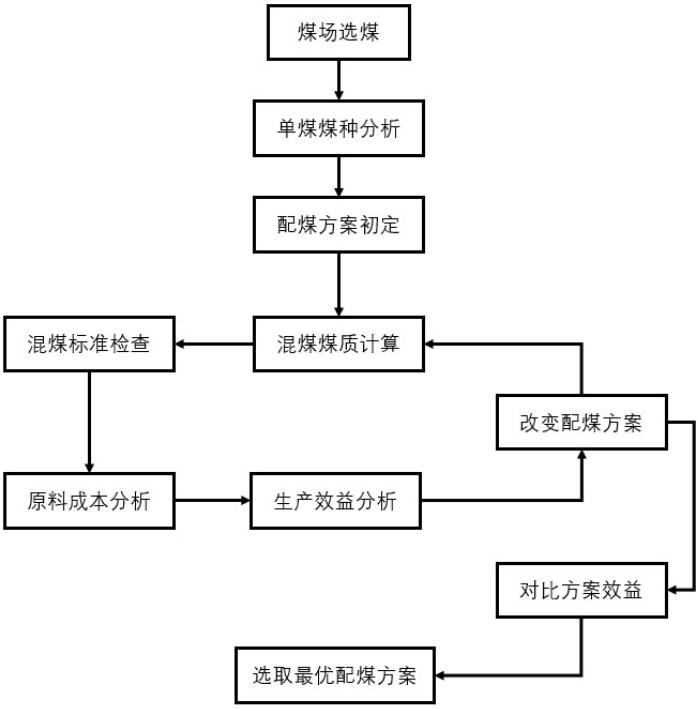


图 2-1 企业配煤方案确定流程

2.2 配煤系统的功能需求

通过分析图 2-1，可以看出企业配煤方案确定流程中存在一定问题，具体表现为以下两点：一是大量重复性、纯计算性的过程：改变配煤方案后需重复进行煤质计算、标准检查、效益分析等工作；二是不确定、不合理的参数范围：工作人员在改变配煤方案参数时，由于人员的专业水平和生产经验各不相同，操作员对不同参数在方案中的实际作用存在误解，使得参数设置不合理，造成方案效益始终较低的现象。

因此，针对以上问题，本文总结了企业配煤管理系统的五个主要需求[18]：

1) 煤仓数据管理。用户可以对煤仓中的单煤数据进行实时的增删改查操作，并且煤仓数据存储需满足一定的约束条件，如关系模型的完整性约束、煤元素分析

数据需保留 2 位小数等。

2) 混煤数据管理。当用户选取 2 种单煤并确定配煤比率后，系统可以自动计算得到混煤的元素分析、工业分析、飞灰成分分析等数据并存储到数据库中，方便后续的标准检查和配煤方案确定。

3) 混煤标准检查。用户选取的混煤，其煤质和飞灰需要符合一定的标准，如挥发分含量至少需 20%。该标准具有默认值，也可由用户自定义修改，以保证混煤燃烧时机组能够安全稳定经济运行，并且污染物排放量达到环保水平。

4) 供电成本分析。用户选择的混煤达标后，用户必须设置上网电价、石灰单价、液氨单价，以供系统进行相关运算和检索。系统将机组参数、混煤标准、单煤煤种等条件相同，而混煤配比不同的方案的供电成本显示给用户，并显示成本与配比的关系图，方便用户进行方案的择优。

5) 用户管理。用户可进行方案的输出、账户密码的修改等操作。

2.3 配煤系统的数学模型

本文基于配煤系统的功能需求，建立了由 2 种单煤混合，多环境参数控制，以降低方案日总供电成本为目标的系统数学模型。模型涉及到的部分计算理论和方法如下[19, 20]。

为了简化配煤问题，当混煤由一定比例的 2 个煤种混合得到时，配煤问题可近似为一个多变量线性回归问题，此时，配煤方案中的混煤标准、脱硫脱硝成本以及厂用电率就成为了影响配煤方案中生产效益的关键因素。

任意选定 2 个单煤并用下标 1 和 2 区别，其对应的煤种单价分别为P_{c1}、P_{c2} 煤 1 的占比为 r，可以得到混煤单价：

$$P_m = P_{c1} \times r_0 + P_{c2} \tag{2-1}$$

P_m——混煤单价（元/t）

P_{c1}——煤 1 单价（元/t）

P_{c2}——煤 1 单价（元/t）

r_0 ——煤 1 占比（%）

对于日发电量 *SP* 一定的机组，当上网电价为P_{power}时，可得到日燃煤成本和日厂用电成本：

$$GP = \frac{24 \times BL}{10} \tag{2-2}$$

GP——日发电量（万度/天）

BL——机组负荷（MW）

由日发电量 SP、厂用电率 r 可得日供电量和日厂用电成本：

$$SP = GP \times (1 - r) \tag{2-3}$$

$$C_{\text{power}} = P_{\text{power}} \times GP \times r \tag{2-4}$$

SP——日供电量（10^4 (kW·h)/天）

C_{power}——日厂用电成本（元/天）

固定锅炉蒸汽流量并初步估计锅炉效率可得混煤煤耗：

$$B_{\text{m}} = \frac{24 \times (2400 D_{\text{gl}} + 207742)}{Q_{\text{arnet}} \times \eta_{\text{gl}} \times 0.01} \times \frac{100}{SP} \tag{2-5}$$

B_{m}——混煤煤耗 (g/ (kW·h))

D_{gl}——锅炉蒸汽流量 (t/h)

η_{gl}——锅炉效率 (%)

由混煤的元素分析结果可得理论空气量和过量空气系数一定时的实际烟气量：

$$V_0 = 0.0889\left(C_{\text{m,ar}} + 0.375 S_{\text{m,ar}}\right) + 0.265 H_{\text{m,ar}} - 0.0333 O_{\text{m,ar}} \tag{2-6}$$

V_0——理论空气量（NH^3/kg）

$$V_{\text{g}} = \frac{1.886(C_{\text{m,ar}} + 0.375 S_{\text{m,ar}}) + 0.8 N_{\text{m,ar}} + 11.1 H_{\text{m,ar}} + 1.24 M_{\text{m,ar}}}{100} + 1.21254 V_0 \tag{2-7}$$

V_{g}——过量空气系数为 1.4 时的实际烟气量（NH^3/kg）

由混煤的硫分可估计 SO_x 产量和日脱硫成本：

$$SO_x = 1821.4 S_{\text{m,ar}} + 376.48 \tag{2-8}$$

$$D_{\text{lime}} = \frac{1.02(SO_x - 20) \times B_{\text{m}} \times SP \times V_{\text{g}}}{64 \times 10^6} \tag{2-9}$$

SO_x——预计 SO_x 产量（NH^3/kg）

D_{lime}——预计石灰石耗量（t/天）

由混煤的氮分可估计 NO_x 产量和日脱硝成本：

$$NO_x = 212 + 155 N_{\text{m,ar}} \tag{2-10}$$

$$D_{\text{NH}_3} = \frac{1224(NO_x - 35) \times B_{\text{m}} \times SP \times V_{\text{g}}}{64 \times 10^6} \tag{2-11}$$

NO_x——预计 NO_x 产量（NH^3/kg）

D_{NH_3}——预计液氨耗量（t/天）

由日燃料成本、日脱硝成本、日脱硫成本和日厂用电成本可得日总供电成本：

$$C_{\text{coal}} = P_{\text{m}} \times \frac{B_{\text{m}} \times SP}{100} \tag{2-12}$$

$$C_{\text{lime}} = P_{\text{lime}} \times D_{\text{lime}} \tag{2-13}$$

$$C_{\text{NH}_3} = P_{\text{NH}_3} \times D_{\text{NH}_3} \tag{2-14}$$

$$C_{\text{total}} = C_{\text{coal}} + C_{\text{lime}} + C_{\text{NH}_3} + C_{\text{power}} \tag{2-15}$$

C_{coal}——日燃煤成本（元/天）

C_{lime}——日脱硫成本（元/天）

C_{NH_3}——日脱硝成本（元/天）

C_{total}——日总供电成本（元/天）

3 配煤管理系统的关键技术

3.1 开发路线

随着 Web 应用的普及及其功能的复杂化，“从无到有”的原生态开发模式逐渐暴露出许多缺点，如工程结构混乱、代码复用率低下等问题。因此许多优秀团队在基于大量开发经验的基础上，形成了满足特定业务需求的工程项目结构，称为“框架”。它预定义了项目的整体结构，模块的分割和协作以及控制流程等，以便于开发者能集中精力于应用本身的业务逻辑。基于计算机基础知识和项目开发基础知识，系统采用前后端分离的开发路线（如图 3-1），并引入主流的开发框架以获得更好的开发效果。

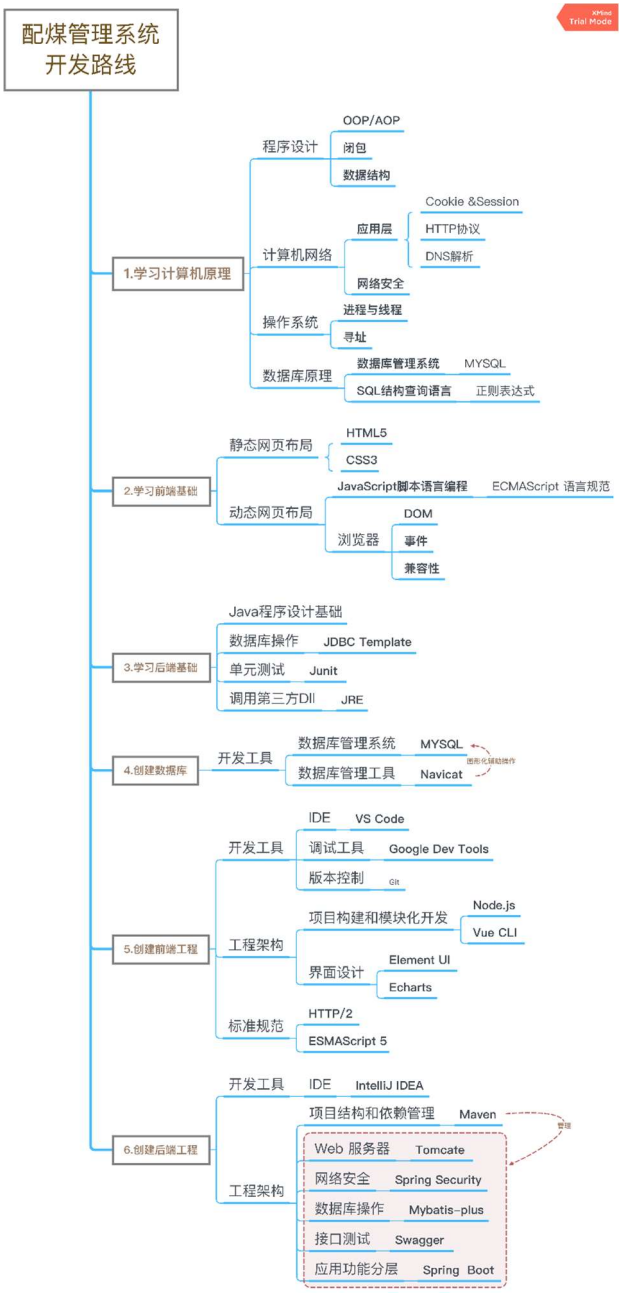


图 3-1 系统开发路线

3.2 前端主要技术

3.2.1 Vue

Vue 是一套开源的、用于构建用户界面的渐进式前端框架[21]。Vue 预先通过 JavaScript 进行各种计算，优化最终作用于 DOM 的各种操作，提高了页面的运行效率，为开发者节省大量操作 DOM 的精力，使开发者能专注于业务逻辑的开发。通过创建 Vue 实例，可实现嵌套的、复用的组件化开发。

Vue CLI（Command-Line Interface）是 Vue 官方发布的一个用于快速搭建 Vue 开发环境依赖的集成式框架。它自动为 Vue 项目规划了合理的代码目录结构（如图 3-2），包括项目 webpack 的依赖包文件夹 node_modules，开发文件夹 src，公共组件文件夹 components，浏览器内路由跳转文件夹 router，项目配置信息文件 package.json 等。

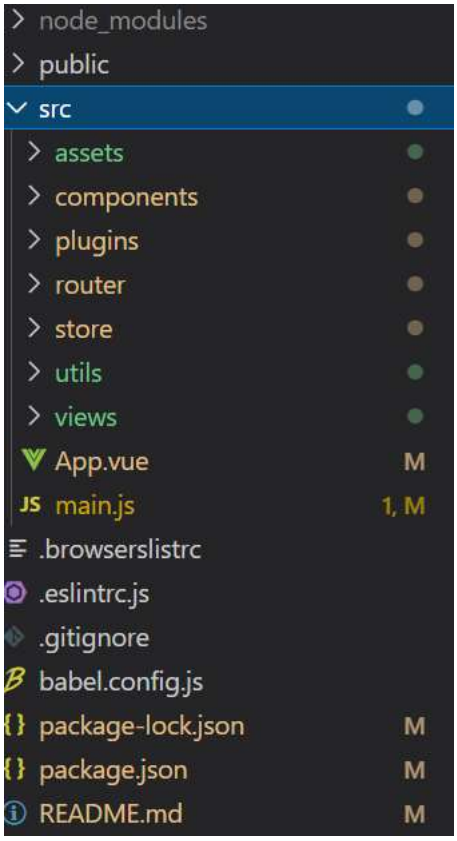


图 3-2 vue-cli 生成的项目目录结构

Vue CLI 还提供可选功能的 npm 包，例如 Babel/TypeScript 转译、ESLint 集成、单元测试和 end-to-end 测试等。开发者只需在对应的目录结构下开发项目文件，即可实现复杂的项目功能，如 webpack 打包、浏览器热加载和跨域访问等。此外，开发者还可以在终端，用 vue ui 命令调出 vue-cli 的图形化用户界面（如图 3-3），并在此页面上进行项目依赖包的管理和配置。

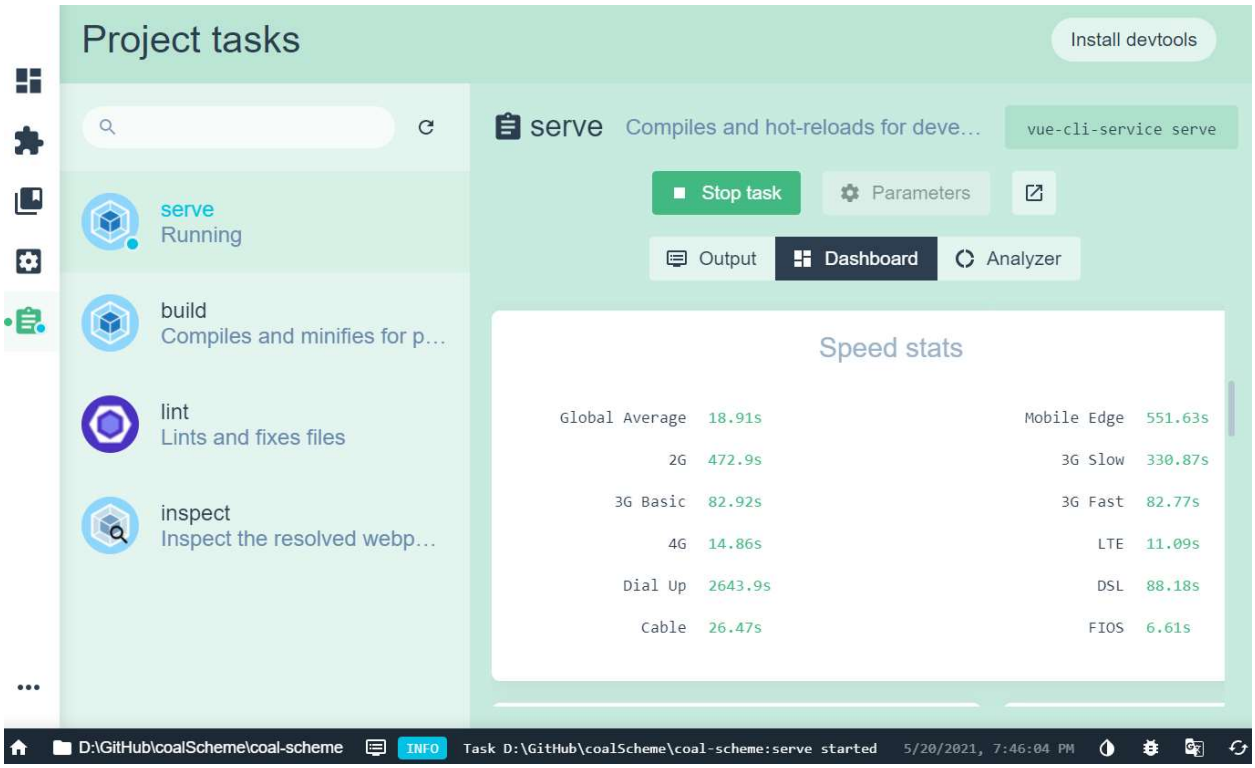


图 3-3 Vue UI 图形化控制面板

3.2.2 Element UI

Element UI（如图 3-4）是由饿了么团队开发的，一套为开发者准备的基于 Vue 2.0 前端框架的桌面端组件库[22]。其组件的设计思路与现实生活的流程、逻辑保持一致，如页面布局按照容器思想，包含 header、main、aside、footer。其组件名称和属性名称的语言表达清晰，能让开发者快速理解组件功能，进行有效开发。同时其简洁直观的设计让用户能够快速上手操作进而做出决策；丰富的信息提示能让用户获得更友好的交互体验。

图 3-4 Element UI 组件化开发库

3.2.3 Echarts

Apache ECharts™ 是一个开源的 JavaScript 可视化图表库[23]，它涵盖了针对各行业可视化需求的数据图表框架，不仅提供数据过滤、聚类、回归等多维度数据分析方式，还支持响应式设计，并且提供了灵活的配置项，方便开发者定制图表。此外，它强大的浏览器兼容性（如 IE8/9/10/11、 Chrome、 Firefox、 Safari 等），使用户可以在 PC 和移动设备上流畅查看由 Echarts 开发的图表。

ECharts 的基本图表类型包括: 线系、柱系、散点系、饼状图、关系图系列、多维数据平行系列、漏斗系列等。开发者可通过下载官方源码或在 node 环境中安装 echarts 把 echarts 文件引入项目，随后在 JavaScript 脚本文件中进行 echarts 实例的创建和配置，即可生成数据图表（如图 3-5）。

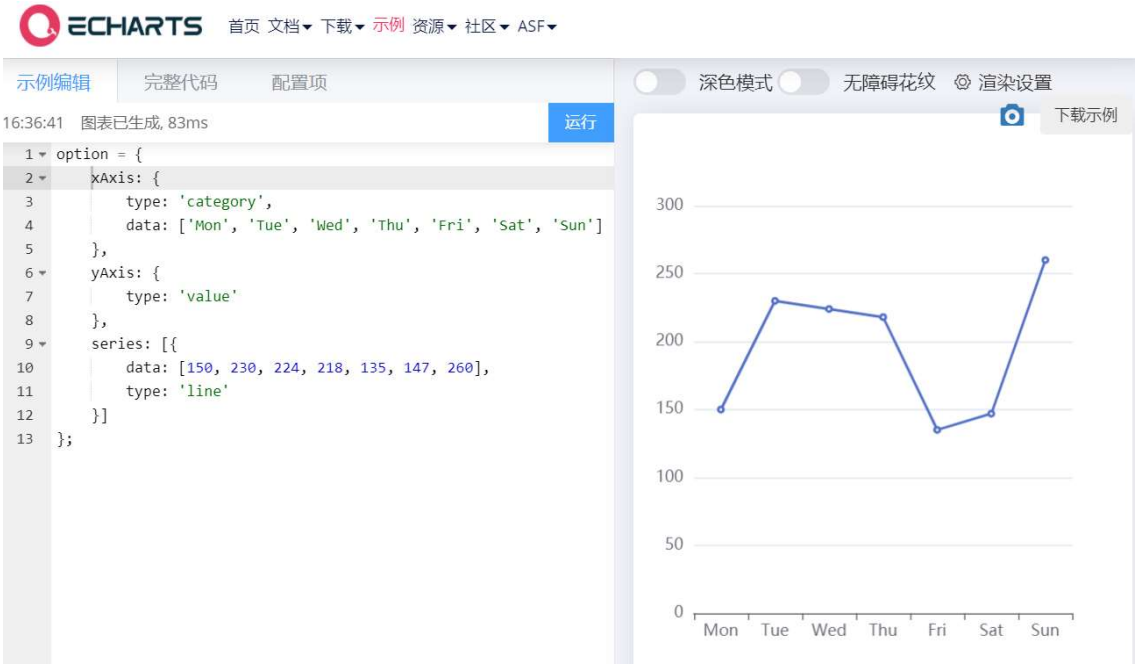


图 3-5 用 JavaScript 编写的自定折线图

本文使用 Vue CLI 集成的 echarts 插件进行 echarts 图表的开发，具体使用步骤如下：

6) 在 vue ui 控制面板上安装相关插件，并将 echarts 包引入项目入口文件 main,js

7) 提前创建一个 DOM 节点对象，用于绑定 echarts 对象进行图表的渲染。

8) 在 Vue 实例创建过程中定义钩子函数，用于修改数据图表的配置项，包括横纵坐标、数据源、图表的动态效果、图表打印工具等

3.3 后端主要技术

3.3.1 Spring 框架

Spring 是由 Rod Johnson 组织和开发的，一个分层的轻量级开源框架，它以 IoC(Inversion of Control，控制反转)和 AOP(Aspect Oriented Programming，面向切面编程)为内核，在业务逻辑层可以管理事务、记录日志；在持久层可以整合 MyBatis-Plus 技术，极大地降低了企业软件开发的复杂度[24]。

Spring 框架采用分层结构（如图 3-6），各功能由 Core Container、Data Access/Integration、Web、Messaging 等多个模块分别实现，各个模块相对独立并以层级依赖，大大降低了组件之间的耦合性，方便开发者使用。

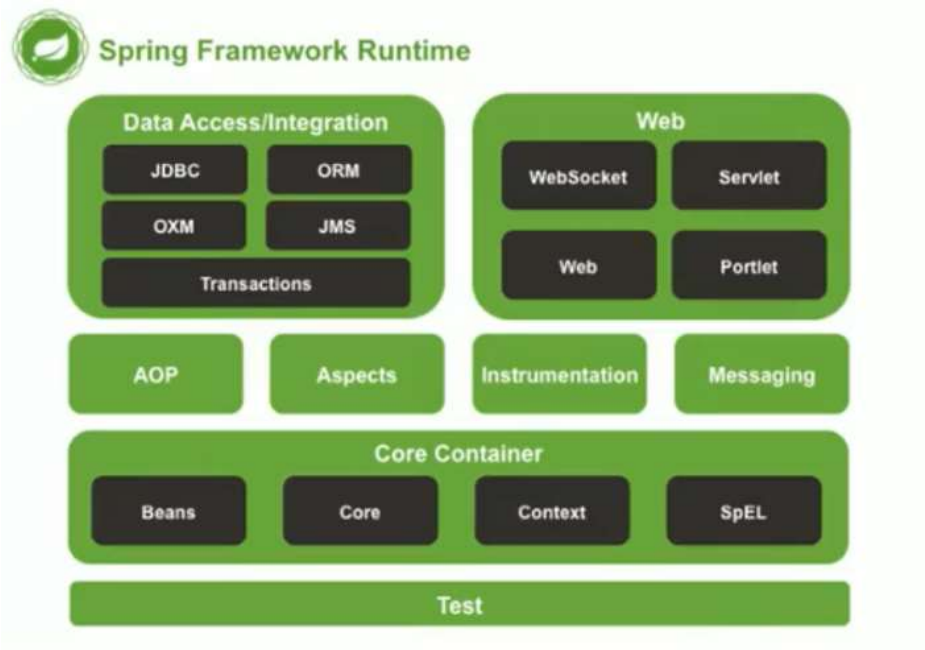


图 3-6 Spring 体系结构

关于 Spring 的优点主要有以下几个方面:

1) 解耦组件、方便开发和维护。开发者可以将所有对象的创建以及依赖关系的维护都交给 Spring 容器，这样对象之间的耦合度会大大降低，方便开发者随时开发新组件和后期系统的维护。

2) 程序复用性高。Spring 的 AOP 内核使它可以将一些通用任务，如安全、事务、日志等进行统一处理，减少了大量重复的代码

3) 可扩展性强。Spring 允许用户在 Spring 框架的基础上添加其它功能的框架，如用于测试的 Junit4 框架、用于数据库操作的 MyBatis-Plus 框架，满足了开发者采用统一框架实现个性化功能的需求。

Spring Security 是一个为基于 Spring 框架开发的企业应用提供声明式安全访问控制的安全框架，它定义了认证、授权、拦截等功能，开发者只需继承封装类的详细信息，即可

实现用户安全管理。

SpringBoot 是由 Pivotal 团队提供的，包含了 Spring 框架所有功能，同时集成了大量常用的第三方库配置。在 Spring Boot 中这些第三方库几乎可以零配置使用，大大简化了 Spring 项目的搭建和开发过程，方便开发者快速搭建和扩展项目。本文通过在 Spring Boot 中集成常用的第三方库（如图 3-7）如 Spring JDBC, Spring Security 等，快速实现系统复杂功能的需求。

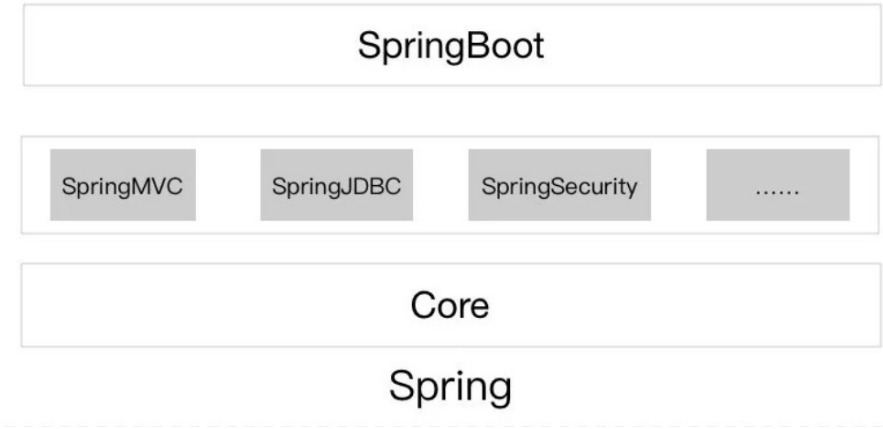


图 3-7 Spring Boot 与 Spring 的关系

3.3.2 MyBatis-Plus 框架

MyBatis 是一个支持自定义 SQL，存储过程和高级映射的一流持久化框架[25]。MyBatis 几乎消除了所有 JDBC（Java DataBase Connection） 代码、参数手动设置和结果检索。MyBatis 可以使用简单的 XML 或注释进行配置，并将原语、 Map 接口和 javapojo (普通的旧 Java 对象)映射到数据库记录。

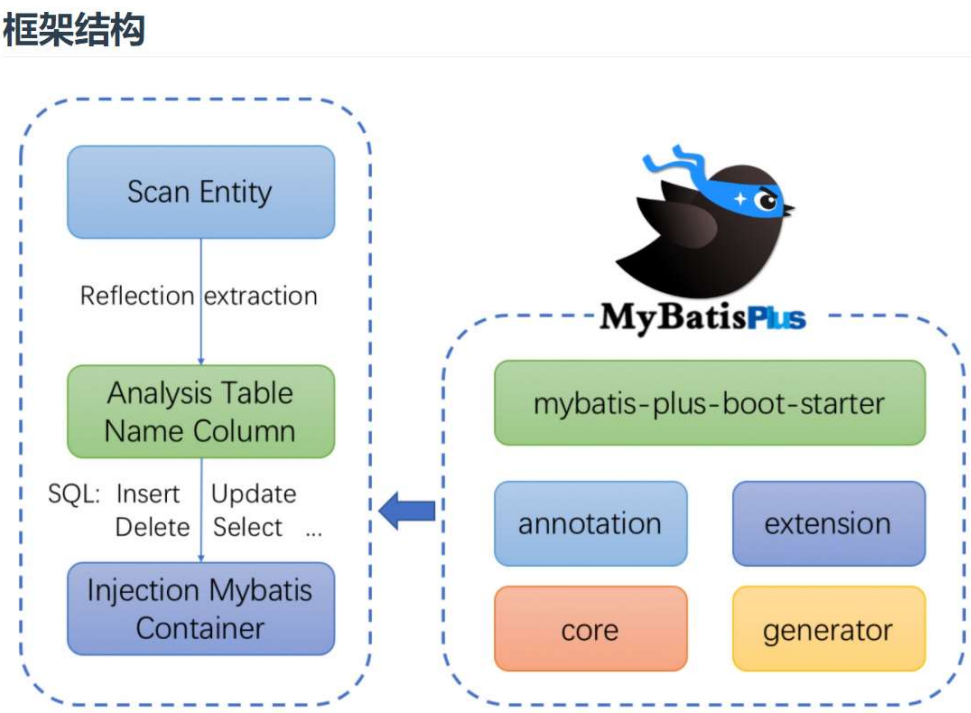


图 3-8 MyBatis-Plus 框架结构

Mybatis-Plus 是一个基于 MyBatis 的增强工具，它可以自动注入基本的 SQL 片段，具有强大而灵活的条件包装器，且有许多有用的插件(例如代码生成器、自动分页、性能分析等等)，从而进一步节省大量的开发时间。

本文设计的系统以企业实际业务需求为导向，利用集成式前端框架 Vue 和集成式后端框架 SSM（Spring Boot + Spring Security+Mybatis-Plus）快速开发满足特定业务场景的系统功能，如：前端界面的数据可视化、后端数据库的操作等。

4 系统设计与实现

4.1 系统设计架构

由第 2 章的分析可知，配煤管理系统要实现煤仓数据管理、混煤数据管理、混煤标准检查、供电成本分析、用户管理这六大功能模块，其中各个模块具有相对独立性但又互相依赖。

因为前端主要负责与用户的交互和数据信息的展示，而后端主要负责业务逻辑的处理和数据库的操作，所以为了方便后续的团队开发，提升系统开发进度，本文设计采用前后端分离式的系统架构，前后端独立开发和部署。其中，前端通过开启 Mock 模拟接口数据，避免对后台接口开发的依赖；后端通过 Swagger 框架进行接口测试，避免对前端访问请求的依赖。正式测试时，前后端通过发送 http 请求和响应进行数据交互。

系统的分层架构，用户在浏览器访问前端提供的登录界时，经浏览器向服务器后端发送资源请求，到服务器底层访问数据库获取到数据，再到服务器后端把处理好的数据作为响应返回到前端，最后前端把响应内容处理成易于理解的文字和图表，并由客户端解析展现给用户上。

访问层指发送 http 请求的客户端，如电脑的网页浏览器。展示层指前端工程中与用户交互的界面内容，即用户看到的文字图表、可点击的按钮等。服务层指在 Vue CLI 4.5 的框架下由 Vue 管理的路由跳转、跨域操作、HTTP 请求等操作，其主要负责处理前端内部业务逻辑和后端的交互。控制器层是后端工程中处理 http 请求的服务器入口层，即前端与后端交互的接口层（API），其包含请求的方式、请求的 URL 路径、请求需传入的参数、响应返回的参数、是否支持跨域访问、是否经过拦截器的等操作；

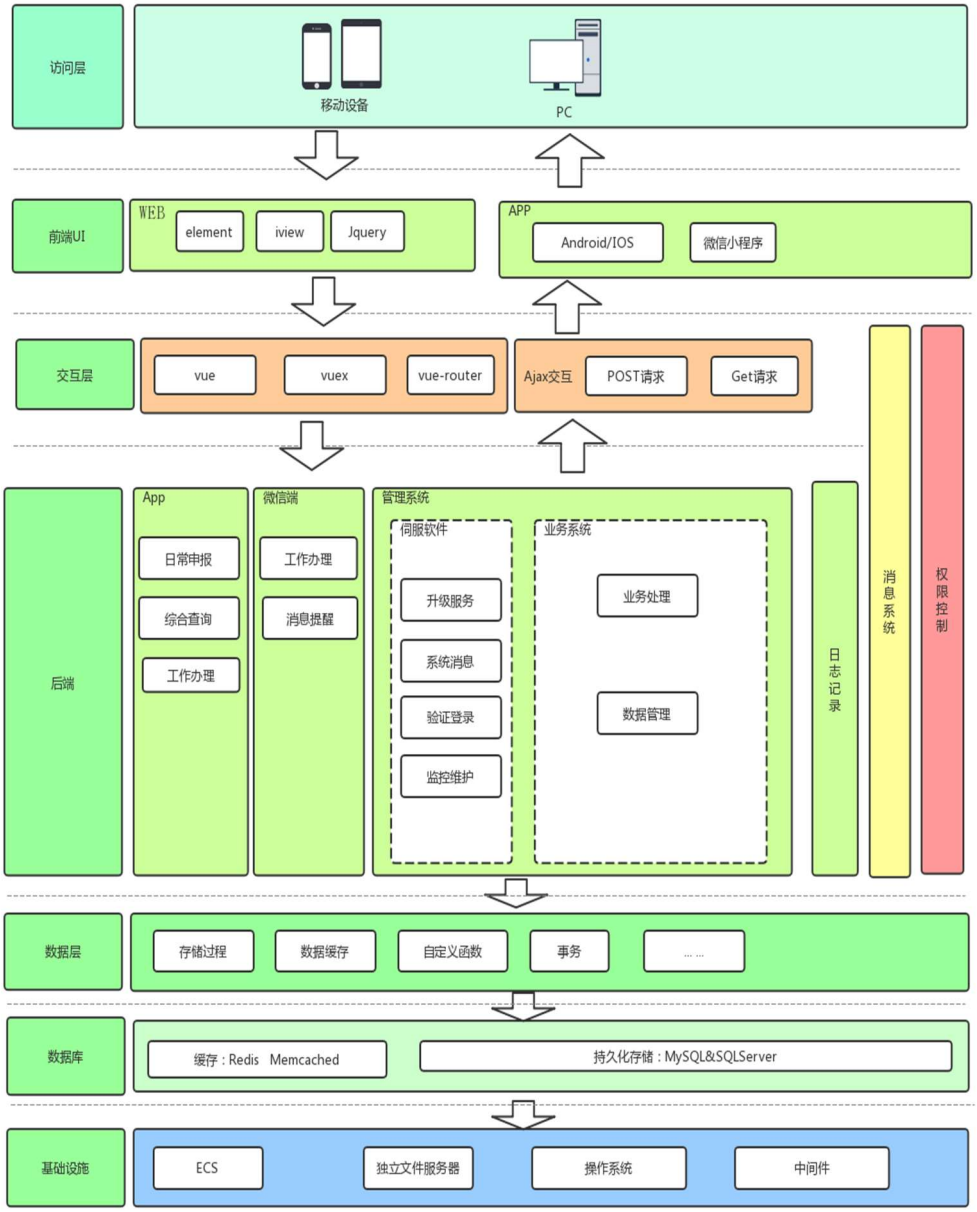


图 4-1 前后端分离的系统设计架构

4.1.1 前端架构

前端采用 MVVM 架构、模块化、组件化的分层设计模式。MVVM 就是通过 Vue 框架的 Vue Model 层，监听用户对 View 层 DOM 节点的操作，然后反馈到 JavaScript 的 Model 层处理操作触发的事件，再通过 Vue Model 层，把事件结果渲染到 Vue 对象管理的 DOM 节点上，形成逻辑闭环，使开发者无需操作 DOM，可专注于 JS 中对事件的处理逻辑。（如图 4-2）

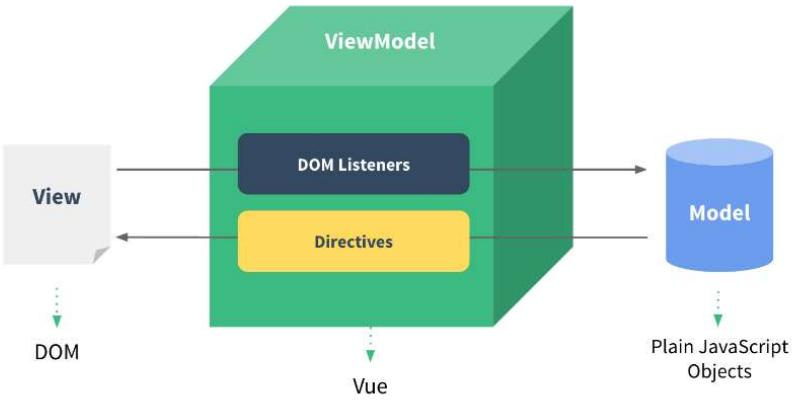


图 4-2 MVVM 架构

本文使用 Vue CLI 创建分层的前端工程项目。Vue CLI 依赖于 Node 环境，Node 环境中的各种工具包能帮助开发者把开发阶段浏览器不支持的 ES6 语法、.vue 文件、.less 文件转换为浏览器兼容的 ES5 语法、html 文件和 js 文件。前端分层架构包括浏览器层、展示层、服务层。Vue CLI 内部封装包括构建部署依赖、页面容器等，开发者可根据需要自定义 util 工具类、

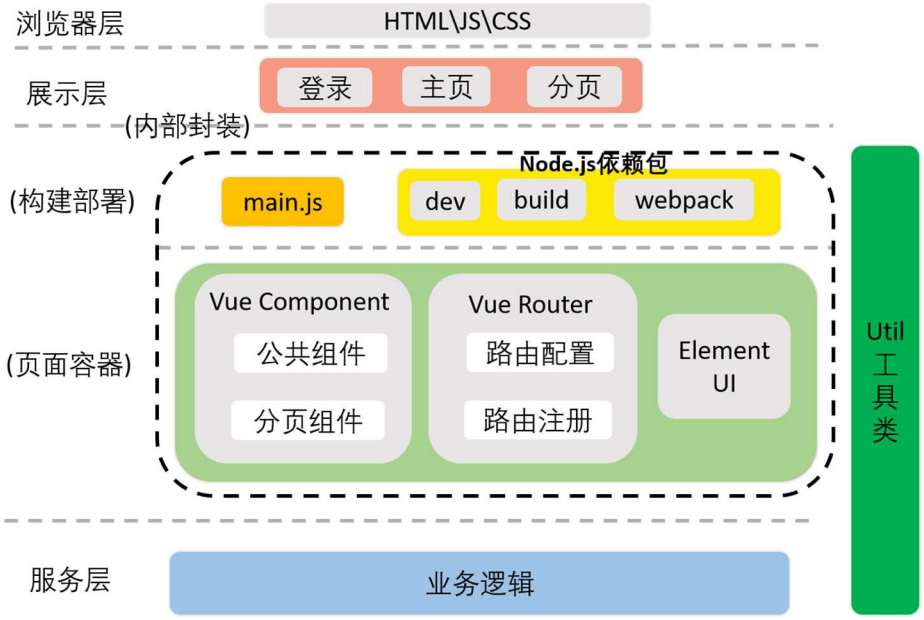


图 4-3 前端分层架构

4.1.2 后端架构

后端采用利用 AOP（面向切面编程）的思想，将有关数据的处理流程形成一条处理流水线，采用自高向低的指令控制方式：Controller（控制层）—Service（业务层）—Mapper（映射层）—Pojo（持久层）—数据库层，完成每次数据处理流程（如图 4-4）。

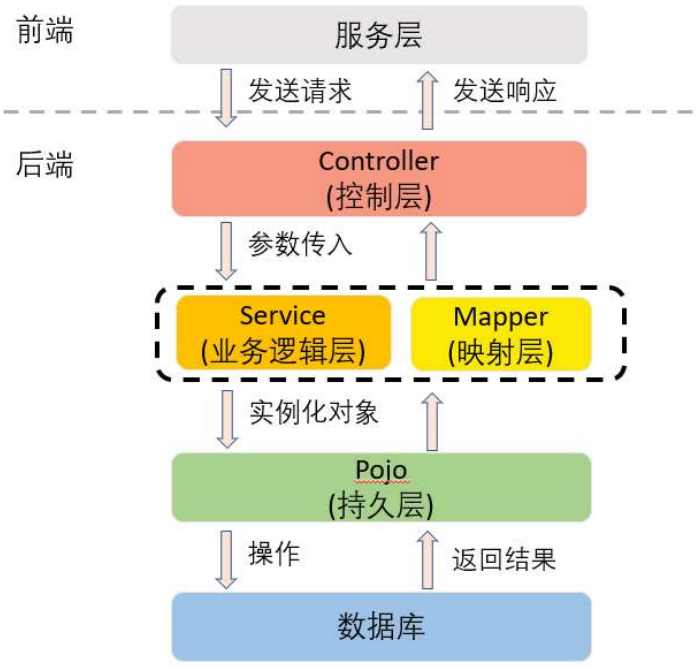


图 4-4 后端分层架构

4.2 数据库实现

基于已有煤仓数据，要实现对煤种信息的增删改查、混煤煤质的分析计算以及配煤方案的信息存取，需要设计出配煤管理系统的数据库表。

根据《阿里巴巴 Java 开发手册（第 2 版）》[26]中对数据库表设计的有关建议，本文的数据库表设计遵循以下原则：

1) 数据库名、表名、字段名均采用小写字母或增加下划线、数字的形式命名；

2) 默认采用的字符集为 utf8mb4，排序规则为 utf8mb4_general_ci，引擎为 InnoDB；

3) 表间关系通过外键关联，关联表必须设置外每张表必须含有主键、每个字段必须添加注释；

4) 本系统涉及数据库脚本默认采用 MYSQL[27]。

4.2.1 数据库表关系

数据库 coal_scheme_db 共包含 5 张表，分别为用户管理表 user、单煤信息表 coal_inplo、单煤煤质表 coal_anlysis、混煤信息表 mixed_analysis、配煤方案表 coal_scheme。

其中，单煤信息表 coal_inplo 通过 coal_id 与单煤煤质表 coal_anlysis 关联，混煤信息表 mixed_analysis 通过 coal_id 与单煤煤质表 coal_anlysis 关联，配煤方案表 coal_scheme 通过 mixed_name 与混煤信息表 mixed_analysis 关联。

4.2.2 数据库表结构

1) 创建用户管理表

用户管理表保存了已注册用户的信息，包括编号、用户名、密码、权限水平、是否启用、备注字段，主键为编号。SQL 代码如下：

```
/*Table structure for table `user` */
DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (
  `user_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '管理员编号',
  `user_name` VARCHAR(50) DEFAULT NULL COMMENT '用户名',
   `user_password` VARCHAR(100) DEFAULT NULL COMMENT '密码',
   `user_level` INT(1) DEFAULT '1' NULL COMMENT'权限等级',
  `enabled` TINYINT(1) DEFAULT '1' COMMENT '是否启用',
  `remark` VARCHAR(255) DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户管理';

/*Data for the table `user` */
INSERT INTO user(user_id, user_name, user_password, user_level) VALUES
(1, 'Jone', '123', 1),
(2, 'Jack', '123', 2);
```

2) 创建单煤信息表

单煤信息表保存了煤种的简要信息，包括编号、名称、单价、储量、产地字段。其中，该表的主键为煤种编号，该表为单煤煤质表的参考表，SQL 代码如下：

```
CREATE TABLE coal_inplo (
coal_id INT(11) NOT NULL AUTO_INCREMENT COMMENT'煤种编号',
coal_name VARCHAR(50) NOT NULL COMMENT'煤种名称',
coal_price DECIMAL(5) DEFAULT 0 COMMENT'煤种价格',
coal_load DECIMAL(8) DEFAULT 0 COMMENT'煤种储量',
coal_origin VARCHAR(50) DEFAULT 'ABC' COMMENT'煤种产地',
PRIMARY KEY(coal_id)
```

)ENGINE=INNODB DEFAULT CHARSET=utf8mb4 COMMENT='单煤信息 '

3) 创建单煤煤质表

单煤煤质表保存了煤种的元素分析数据、工业分析数据以及飞灰成分分析数据，包括收到基全水分、可燃基挥发分、飞灰中三氧化二铝含量等字段。其中，该表的主键和外键均为煤种编号，SQL 代码如下：

```
/*创建单煤煤质表*/
CREATE TABLE coal_analysis (
coal_id INT(11) NOT NULL AUTO_INCREMENT COMMENT'煤种编号',
coal_m_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基全水分',
coal_m_ad decimal(4, 2) DEFAULT 0 COMMENT'空干基水分',
coal_a_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基灰分',
coal_v_daf decimal(4, 2) DEFAULT 0 COMMENT'可燃基挥发分',
coal_v_ad decimal(4, 2) DEFAULT 0 COMMENT'空干基挥发分',
coal_qnet decimal(6, 2) DEFAULT 0 COMMENT'低位发热量',
coal_c_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基碳',
coal_h_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基氢',
coal_n_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基氮',
coal_s_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基全硫',
coal_o_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基氧',
ash_dt_ar decimal(6, 2) DEFAULT 0 COMMENT'灰变形温度',
ash_st_ar decimal(6, 2) DEFAULT 0 COMMENT'灰软化温度',
ash_ft_ar decimal(6, 2) DEFAULT 0 COMMENT'灰流动温度',
ash_sio2_ar decimal(4, 2) DEFAULT 0 COMMENT'二氧化硅',
ash_al2o3_ar decimal(4, 2) DEFAULT 0 COMMENT'三氧化二铝',
ash_fe2o3_ar decimal(4, 2) DEFAULT 0 COMMENT'三氧化二铁',
ash_tio2 decimal(4, 2) DEFAULT 0 COMMENT'二氧化钛',
ash_cao decimal(4, 2) DEFAULT 0 COMMENT'氧化钙',
ash_mgo decimal(4, 2) DEFAULT 0 COMMENT'氧化镁',
ash_so3 decimal(4, 2) DEFAULT 0 COMMENT'三氧化硫',
ash_k2o decimal(4, 2) DEFAULT 0 COMMENT'氧化钾',
ash_na20 decimal(4, 2) DEFAULT 0 COMMENT'氧化钠',
CONSTRAINT pk_coal_id PRIMARY KEY(coal_id),
CONSTRAINT fk_coal_inplo FOREIGN KEY(coal_id) REFERENCES coal_inplo(coal_id)
```

```
)ENGINE=INNODB DEFAULT CHARSET=utf8mb4 COMMENT='单煤煤质 ';
```

4) 创建混煤信息表

混煤信息表保存了一定比例两种单煤种混合形成的混煤煤种信息，包含混煤编号、方案名称、煤种 1 编号、煤种 2 编号、煤种 1 比率、混煤可燃基挥发分、混煤飞灰软化温度等字段。其中，该表的方案编号为主键，煤种 1 编号和煤种 2 编号为外键；该表是配煤方案表的参考表，SQL 代码如下：

```
/*创建混煤信息表*/
CREATE TABLE mixed_analysis
(
    mixed_id INT(11) NOT NULL AUTO_INCREMENT COMMENT'混煤编号',
    mixed_name VARCHAR(50) NOT NULL UNIQUE COMMENT'方案名称',
    coal1_id INT(11) NOT NULL COMMENT'煤种 1 编号',
    coal2_id INT(11) NOT NULL COMMENT'煤种 2 编号',
    coal1_ratio DECIMAL(4, 2) DEFAULT 80 COMMENT'煤种 1 比率(%)',
    mixed_c_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基碳(%)',
    mixed_h_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基氢(%)',
    mixed_o_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基氧(%)',
    mixed_n_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基氮(%)',
    mixed_s_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基全硫(%)',
    mixed_s_ad decimal(4, 2) DEFAULT 0 COMMENT'空干基硫(%)',
    mixed_a_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基灰分(%)',
    mixed_m_ar decimal(4, 2) DEFAULT 0 COMMENT'收到基全水分(%)',
    mixed_m_ad decimal(4, 2) DEFAULT 0 COMMENT'空干基水分(%)',
    mixed_v_daf decimal(4, 2) DEFAULT 0 COMMENT'可燃基挥发分(%)',
    mixed_qnet_ar decimal(6, 2) DEFAULT 0 COMMENT'收到基低位发热量(kcal)',
    ash_sio2_ar decimal(4, 2) DEFAULT 0 COMMENT'二氧化硅(%)',
    ash_al2o3_ar decimal(4, 2) DEFAULT 0 COMMENT'三氧化二铝(%)',
    ash_fe2o3_ar decimal(4, 2) DEFAULT 0 COMMENT'三氧化二铁(%)',
    ash_tio2 decimal(4, 2) DEFAULT 0 COMMENT'二氧化钛(%)',
    ash_cao decimal(4, 2) DEFAULT 0 COMMENT'氧化钙(%)',
    ash_mgo decimal(4, 2) DEFAULT 0 COMMENT'氧化镁(%)',
    ash_so3 decimal(4, 2) DEFAULT 0 COMMENT'三氧化硫(%)',
    ash_k2o decimal(4, 2) DEFAULT 0 COMMENT'氧化钾(%)',
```

```
    soft_t decimal(6, 2) DEFAULT 0 COMMENT'软化温度(°C)',
    CONSTRAINT pk_mixed_id PRIMARY KEY(mixed_id),
    CONSTRAINT fk_coal1_id FOREIGN KEY(coal1_id) REFERENCES coal_analysis(coal_id),
    CONSTRAINT fk_coal2_id FOREIGN KEY(coal2_id) REFERENCES coal_analysis(coal_id)
)ENGINE=INNODB DEFAULT CHARSET=utf8mb4 COMMENT='混煤信息表 ';
```

5) 创建配煤方案表

配煤方案表保存了用户在配煤过程中设置的有关燃煤超洁排放的煤质标准（如最小可燃基挥发分）、燃煤超洁排放的材料成本（如石灰单价）、燃煤机组热力参数（如机组负荷）以及预计的供电信息（如发电量、供电成本），其中，该表的方案编号为主键，混煤编号为外键，SQL 代码如下：

```
/*创建配煤方案表*/
CREATE TABLE coal_scheme
(
    scheme_id INT(11) NOT NULL AUTO_INCREMENT COMMENT'方案编号',
    mixed_id INT(11) NOT NULL COMMENT'混煤编号',
    min_v_daf decimal(4, 2) DEFAULT 0 COMMENT'最小可燃基挥发分(%)',
    max_m_ar decimal(4, 2) DEFAULT 0 COMMENT'最大收到基水分(%)',
    max_a_ar decimal(4, 2) DEFAULT 0 COMMENT'最大收到基灰分(%)',
    max_s_ad decimal(4, 2) DEFAULT 0 COMMENT'最大空干基硫分(%)',
    min_qnet_ar decimal(6, 2) DEFAULT 0 COMMENT'最小收到基低位发热量(kcal)',
    min_soft_t decimal(6, 2) DEFAULT 0 COMMENT'最小软化温度(°C)',
    sox decimal(5, 1) DEFAULT 0 COMMENT'SOx 估计(mg/Nm3)',
    nox decimal(5, 1) DEFAULT 0 COMMENT'NOx 估计(mg/Nm3)',
    t_env decimal(4, 2) DEFAULT 20 COMMENT'环境温度(°C)',
    boiler_eff decimal(4, 2) DEFAULT 0 COMMENT'锅炉效率(%)',
    mixed_consump_kwh decimal(5, 2) DEFAULT 0 COMMENT'混煤煤耗(g/kwh)',
    stand_consump_kwh decimal(5, 2) DEFAULT 0 COMMENT'标煤煤耗(g/kwh)',
    boilder_load decimal(5, 2) DEFAULT 0 COMMENT'机组负荷(MW)',
    steam_flow decimal(5, 2) DEFAULT 0 COMMENT'主汽流量(t/h)',
    mixed_price decimal(5, 2) DEFAULT 0 COMMENT'混煤单价(元/t)',
    lime_price decimal(3, 1) DEFAULT 0 COMMENT'石灰单价(元/t)',
```

```
nh3_price decimal(6, 2) DEFAULT 0 COMMENT'液氨单价(元/t)',
power_price decimal(5, 4) DEFAULT 0 COMMENT'上网电价(元/kwh)',
lime_consump decimal(3, 1) DEFAULT 0 COMMENT'石灰耗量(t/天)',
lime_cost decimal(5, 1) DEFAULT 0 COMMENT'石灰成本(元/天)',
nh3_consump decimal(3, 2) DEFAULT 0 COMMENT'液氨耗量(t/天)',
nh3_cost decimal(5, 1) DEFAULT 0 COMMENT'脱硝成本(元/天)',
power_generate decimal(4, 1) DEFAULT 0 COMMENT'日发电量(万度/天)',
ratio_own decimal(2, 1) DEFAULT 0 COMMENT'厂用电率(%)',
power_supply decimal(5, 2) DEFAULT 0 COMMENT'日供电量(万度/天)',
power_cost_kwh decimal(5, 4) DEFAULT 0 COMMENT'供电成本(元/kwh)',
power_cost_day decimal(5, 4) DEFAULT 0 COMMENT'生产成本(元/天)',
CONSTRAINT pk_scheme_id PRIMARY KEY(scheme_id),
/*foreign key 主表字段属性至少为 unque, 否则不能设置为 fk*/
CONSTRAINT fk_mixed_id FOREIGN KEY(mixed_id) REFERENCES mixed_analysis(mixed_id)
)ENGINE=INNODB DEFAULT CHARSET=utf8mb4 COMMENT='配煤方案管理';
```

4.2.3 数据库操作异常的后端处理

当前端用户存在数据库不允许的操作，如删除主表的某条记录。因为该条记录被参考表关联，因此会出现数据库操作异常。通过在后端工程的 exception 文件夹下创建 GlobalException 类，系统统一处理数据库操作异常时抛出的信息并返回到前端，使前端用户能了解哪些操作是不被系统允许的，从而获得更好使用体验。后端 Java 代码如下:

```
package buzz.freelearner.server.exception;

import buzz.freelearner.server.pojo.ResponseBean;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.sql.SQLException;
import java.sql.SQLIntegrityConstraintViolationException;

/**
 * 全局异常处理
 * @author eva
 * @create 2021/4/28
 * @RestControllerAdvice 自定义异常拦截类
 */
@RestControllerAdvice
```

```
public class GlobalException {
    @ExceptionHandler(SQLException.class)
    public ResponseBean mysqlException(SQLException exception){
        //对数据库的操作违反了数据库的唯一约束条件
        //如删除主表记录或插入 Unique 字段重复的数据
        if(exception instanceof SQLIntegrityConstraintViolationException){
            return ResponseBean.error("数据已存在且存在关联数据!");
        }
        return ResponseBean.error("数据库异常，操作失败!");
    }

}
```

4.3 系统安全模块实现

系统的安全内容主要包含认证（Authentication）和授权（Authorization）两个方面。认证是为了保护系统的隐私数据与资源，在用户身份合法时才可以访问系统资源，如用户需先登录账户才能进入系统。授权是为了更加细化系统资源的保护，用户有一定权限时才能访问系统的某些资源，如普通用户无法修改上网电价。

4.3.1 用户登录前端实现

前端在公共组件 component 中创建登录界面 Login.vue，作为用户访问系统和退出系统时的入口页面，并将其导入前端入口函数 main.js 中，结构层和行为层核心代码如下：

```
<template>
  <div class="login_container">
    <div class="login_box">
      <!—登录账户头像 -->
      <div class="avatar_box">
        <img src="../assets/logo.png" alt="" />
      </div>
      <!—输入用户名和密码的表单 -->
      <div>
        <el-form
          ref="ruleFormRef"
          :model="ruleForm"
          :rules="rules"
          label-width="0px"
          class="demo-ruleForm input-form"
        >
          <!--username -->
          <el-form-item prop="username">
            <el-input
```

```
          placeholder="请输入用户名"
          prefix-icon="el-icon-user-solid"
          v-model="ruleForm.username"
        ></el-input>
      </el-form-item>
      <!-- password -->
      <el-form-item prop="password">
        <el-input
          placeholder="请输入密码"
          prefix-icon="el-icon-key"
          v-model="ruleForm.password"
          type="password"
        ></el-input>
      </el-form-item>
      <!-- button -->
      <el-form-item>
        <el-button class="input-button" type="primary" @click="submitForm()"
          >登录</el-button
        >
        <!-- @click 事件绑定 resetForm 方法 传入表单名字 ruleFormRef-->
        <el-button class="input-button" @click="resetForm()"
          >重置</el-button
        >
      </el-form-item>
    </el-form>
  </div>
 </div>
</div>
</template>

<script>
export default {
 name: "Login",
 data() {
  return {
   //表单数据
   ruleForm: {
    username: "",
    password: "",
   },
   //表单验证规则
   rules: {
    username: [
```

```
        { required: true, message: "用户名不能为空", trigger: "blur" }, //'blur'为鼠标失焦时
        //{ min: 3, max: 5, message: 'Length should be 3 to 5', trigger: 'blur' }
      ],
      password: [
        { required: true, message: "密码不能为空", trigger: "blur" },
        //{ min: 3, max: 5, message: 'Length should be 3 to 5', trigger: 'blur' }
      ],
    },
  };
 },
 methods: {
  //预验证: 提交前检查验证规则
  submitForm() {
   //查看预验证
   this.$refs.ruleFormRef.validate((valid) => {
    //预验证通过，则 post request (ps:valid 为预验证结果)
    if (valid)
     this.$axios.defaults.baseURL = "http://localhost:8081";
     this.postRequest("/login", this.ruleForm).then((response) => {
      if (response) {
       //拿到 token 并存在 sessionStorage 内，对应 keyName 为"tokenStr"
       //以此作为系统内访问的令牌
       const tokenStr =
        response.object.tokenHead + response.object.token;
       window.sessionStorage.setItem("tokenStr", tokenStr);
       //1.1.2 跳转 url
       //默认 replace 的端口号是 8080, 不用 push，避免返回/login 页面
       this.$router.replace("/home");
      }
      //else 在 api.js 统一处理
     });
    } else {
     this.$message.error("输入非法!");
     return false;
    }
   });
  },
  //重置
  resetForm() {
   this.$refs.ruleFormRef.resetFields();
  },
 },
};
```

```
</script>
```

4.3.2 用户登录和请求拦截的后端实现

后端采用 Spring Security 结合 JWT（JSON Web Token）实现用户认证（能否登陆）和用户授权（能否使用某些功能）这两个主要的安全访问控制功能。首先，需在 service 模块的 pom.xml 中添加依赖包：

```
<!--Spring Security to generate JWT-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.0</version>
</dependency>
```

其次，在开发文件 src/main/java 下创建 security 包，添加用于配置 Spring Security 的类 SecurityConfig、用于处理用户未登录或 token 过期的类 RestAuthorizationEntryPoint、用于处理用户已登录但无权限访问时的类 RestfulAccessDeniedHandler、自动生成 token 的工具类 JwtTokenUtils、用于拦截请求并判断是否放行的类 JwtAuthenticationTokenFilter，用户请求拦截流程如图 4-5。

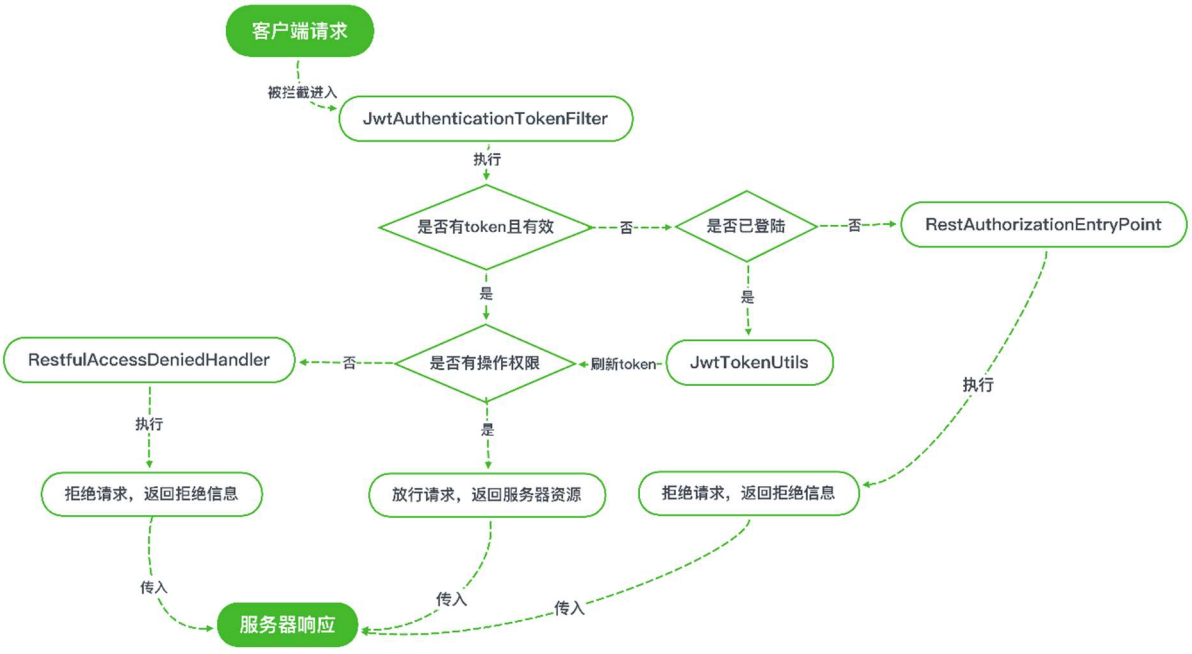


图 4-5 用户请求拦截流程

其中，SecurityConfig 是使用 Spring Security 进行安全管理的关键。SecurityConfig 需继

承 WebSecurityConfigurerAdapter 并重写其配置方法 configure。SecurityConfig 的核心代码如下：

```
/** Security configuration
 * @author eva
 * @create 2021/4/24 download
 */
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    //注入类
    //自定义的 IUserService 接口
    @Autowired
    private IUserService userService;
    //前端用户未登录或 token 过期时访问系统接口，后端的返回结果
    @Autowired
    private RestAuthorizationEntryPoint restAuthorizationEntryPoint;
    //前端用户无权限时访问系统接口，后端的返回结果
    @Autowired
    private RestfulAccessDeniedHandler restfulAccessDeniedHandler;

    //重写 userDetailsService
    @Override
    @Bean
    public UserDetailsService userDetailsService(){
        //1.重写 username
        return username ->{
            //由 IUserService 内定义的接口获取 LoginUser 的 username
            User user = userService.getUserByUserName(username);
            if(user != null){ return user;}
            return null;
        };
    }

    //解码用户登录密码的解码器
    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }
    //请求拦截器：检查请求头中的 Token 并判断是否过滤
    @Bean
    public JwtAuthenticationTokenFilter jwtAuthenticationTokenFilter(){ return new
JwtAuthenticationTokenFilter();}
```

```
    //2.放入重写的 userDetailsService
    @Override
    protected void configure(AuthenticationManagerBuilder authBuilder){
        try {
            //获取用户 info 时，隐藏 password

authBuilder.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());
        } catch (Exception e) {
            System.out.println("无法执行重写的 userDetailsService");
        }
    }

    //需要放行的一些特殊路径，不走拦截器
    @Override
    public void configure(WebSecurity web){
        //勿漏根路径"/",暂时全放行
        web.ignoring().antMatchers(
                "/**"
        );
    }

   //核心配置方法
    @Override
    protected void configure(HttpSecurity http){
        //使用 token，关闭 csrf
        try {
            http.csrf()
                    .disable()
                    //使用 token，关闭 session
                    .sessionManagement()
                    .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                    .and()
                    .authorizeRequests()
                    //除了请求特殊路径，其他请求均需要授权
                    .anyRequest()
                    .authenticated()
                    .and()
                    //禁用缓存
                    .headers()
                    .cacheControl();
            //添加 jwt filter
```

```
                http.addFilterBefore(jwtAuthenticationTokenFilter(),
UsernamePasswordAuthenticationFilter.class);
                //添加自定义未授权和未登录的自定义返回结果
                //详见 RestfulAccessDeniedHandler, RestAuthorizationEntryPoint definition
                http.exceptionHandling()
                        .accessDeniedHandler(restfulAccessDeniedHandler)
                        .authenticationEntryPoint(restAuthorizationEntryPoint);
            } catch (Exception e) {
                System.out.println();
            }
        }
}
```

4.4 煤仓数据管理模块实现

数据管理模块的实现主要由煤仓数据的增删改查和混煤数据的计算、存储及可视化两部分构成。

4.4.1 煤仓数据的增删改查

前端工程中，通过在 view 层创建 CoalInfo 模块获取用户需要执行的增删改查操作。

首先，将创建的 CoalAnalysis 模块导入路由跳转文件 index.js，并在页面容器 Vue-Router 中注册跳转路由"/coal/list"。在设计展示层界面时，通过引入 Element UI 组件库的 Card、Form、FormItem、Input、Table、TableColumn 等组件，实现展示层的表格、新建煤种信息表单、删除煤种信息提示框、修改煤种信息表单、增删改查按钮等。以查询煤种信息和增加煤种信息为例，主要代码如下：

```
<template>
  <div>
    <el-card shadow="never">
      <!-- 添加煤种按钮-->
      <el-button type="success" @click="addFormVisible = true" class="add-btn"
        >添加煤种</el-button
      >
      …
       <!—新建煤种信息的弹出表单 -->
      <el-dialog title="新建煤种信息" :visible.sync="addFormVisible">
        <el-form :model="newCoal">
          <el-form-item label="煤种编号" :label-width="formLabelWidth">
            <el-input v-model="newCoal.coalId" autocomplete="off"></el-input>
```

```
        </el-form-item>
        <el-form-item label="煤种名称" :label-width="formLabelWidth">
          <el-input v-model="newCoal.coalName" autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="煤种单价" :label-width="formLabelWidth">
          <el-input v-model="newCoal.coalPrice" autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="煤种存量" :label-width="formLabelWidth">
          <el-input v-model="newCoal.coalLoad" autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="煤种产地" :label-width="formLabelWidth">
          <el-input
            v-model="newCoal.coalOrigin"
            autocomplete="off"
          ></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addFormVisible = false">取  消</el-button>
        <el-button type="primary" @click="addCoal">确  定</el-button>
      </div>
    </el-dialog>

    <!—监控煤种表格选项变化
    @selection-change:每次勾选变化时都会触发这个事件，以保证选取 2 个煤种
    -->
    <el-table
      ref="multipleTable"
      @selection-change="handleSelectionChange"
      :data="coalList"
      tooltip-effect="dark"
      style="width: 100%"
    >
      <el-table-column
```

```
            type="selection"
            width="55"
            :selectable="handleDisabled"
          ></el-table-column>
          <el-table-column prop="coalId" label="煤种编号" width="100">
          </el-table-column>
          <el-table-column prop="coalName" label="煤种名称" width="150">
          </el-table-column>
          <el-table-column prop="coalPrice" label="煤种单价" width="150">
          </el-table-column>
          <el-table-column prop="coalLoad" label="煤种存量" width="150">
          </el-table-column>
          <el-table-column prop="coalOrigin" label="煤种产地" width="120">
          </el-table-column>
          <el-table-column label="操作" width="180">
            <template slot-scope="scope">
              <el-button
                size="mini"
                @click="editCoal(scope.row)"
                type="primary"
                icon="el-icon-edit"
                circle
              ></el-button>
              <el-button
                size="mini"
                @click="deleteCoal(scope.row)"
                type="danger"
                icon="el-icon-delete"
                circle
              ></el-button>
            </template>
          </el-table-column>
        </el-table>
…
```

```
            </el-card>
          </div>
        </template>
```

其次，在服务层实现与后端的交互，包括页面初始化时向服务器后端发送请求，以获取煤种信息的 initCoalList()方法、添加煤种信息的 addCoal()方法、删除煤种信息的 deleteCoal()方法、修改煤种信息的 updateCoal()方法等以查询煤种信息和增加煤种信息为例，主要代码如下：

```
//回调钩子函数 mounted()
mounted() {
    //mounted 在 DOM 渲染之前创建，如果更新 mounted 必须刷新页面
    this.initCoalList();
  },

  methods: {
    initCoalList() {
      //跨域获取 coalList
      this.getRequest("/coal/list/get").then((response) => {
        if (response) {
          this.coalList = response;
        }
      });
    },
    //添加煤种信息
    addCoal() {
      if (this.newCoal) {
        this.postRequest("/coal/list/add", this.newCoal).then((response) => {
          if (response) {
            //添加完成后重新获取 coalList
            this.initCoalList();
            this.addFormVisible = false;
          }
        });
      }
},
…
},
…
```

后端工程中，为了快速并正确执行用户对数据库的增删改查操作，必须首先生成对应数据库表的实体类，再编写相应的 SQL 语句操作数据库。

本文在对应的持久层创建了实体类 CoalInplo 和 CoalAnalysis 类来操作 coal_inplo 表和

coal_analysis 表中的记录（类代码详见附录）。由于后端集成了 Mybatis-plus 框架，通过使用 Mybatis-plus 封装的通用 Service CURD 来获取数据库中的数据可极大地节省编写 SQL 语句的事件，提高开发效率。当请求的是数据库数据时，Service 层将把获取到的数据库数据以 JSON 格式封装到响应中返回给前端；当请求的执行数据库操作时，Controller 层将通过 ResponseBean 把提示消息以 JSON 格式封装到响应中返回给前端。以查询煤种信息和增加煤种信息为例，主要代码如下：

```
/**
 * <p>
 * 煤种信息 CURD 控制层
 * </p>
 *
 * @author eva
 * @since 2021-04-24
 */
@RestController
@CrossOrigin
@RequestMapping("/coal/list")
public class CoalInploController {
    //注入类
    @Autowired
    private ICoalInploService coalInploService;

    //api
    //api 说明
    @ApiOperation(value = "获取所有煤种")
    //api  请求方式及地址 url
    @GetMapping("/get")
    public List<CoalInplo> getCoalList(){
        return coalInploService.list();
    }

    @ApiOperation(value ="添加煤种")
    @PostMapping("/add")
    public ResponseBean addCoalInplo(@RequestBody CoalInplo coalInplo){
        //save  成功则返回 success
        if(coalInploService.save(coalInplo)){
            return ResponseBean.success("添加成功！ ");
        }
        return ResponseBean.error("操作失败!");
    }
}
```

4.4.2 混煤数据的计算、存储及可视化

前端工程中，通过在 view 层创建 CoalAnalysis 模块实现混煤数据的操作。

与 CoalInfo 模块的实现类似，首先将创建的 CoalAnalysis 模块导入路由跳转文件，再注册跳转路由"/coal/analysis"。由于煤质数据的可视化需要用到 Echarts 图表库中的 echarts 实例，因此需要先在项目入口函数文件 main.js 中导入 echarts 模块，具体代码如下：

```
//导入 5.x 版本 Echarts
import * as echarts from "echarts";
Vue.prototype.$echarts = echarts;
```

基于前端的分层架构模式，在结构层，用 Element UI 的 Slider 组件实现混煤比例的改变；在行为层，通过异步获取后端煤质数据并在 Vue 实例的钩子函数中初始化煤质金字塔图表；此后，在结构层监听用户拖动比例调节滑块的事件，行为层动态计算并展示混煤数据；最后，在用户进行环境参数设置前，向后端发送混煤数据并存入数据库，实现混煤数据的计算、存储及可视化。CoalAnalysis 的核心代码如下：

```
//结构层
<template>
  <div>
    <el-card>
      <div class="block">
        <span class="demonstration slider-info"
          >当前混煤比例：{{ ratio }}%</span
        >
        <!-- 属性说明
          v-model="ratio"动态绑定 slider 的值到 this.ratio
          @change="updateMixed" 监听鼠标拖动事件 -->
        <el-slider
          v-model="ratio"
          :step="10"
          show-stops
          ref="slider_ref"
          @change="updateMixed"
        ></el-slider>
        <el-button type="success" @click="saveMixedData"
          >保存混煤数据</el-button
        >
      </div>
    </el-card>
    <el-card>
```

```
        <!-- 为 echarts 准备一个 DOM -->
        <div class="main" ref="funnels"></div>
      </el-card>
    </div>
</template>

//methods 中异步获取煤质数据并渲染到 DOM 的方法
     // 跨域获取 coalAnalysis
    async initAnalysis(dataOption, ratio) {

     /**1.1 设置跨域 url**/
     /**1.2 发起跨域请求获取后台数据
     传入参数: 储存在 sessionStorage 的参数 coalIds(由后台接口决定传参数格式和内容),
     传入格式详见 CoalInplo.vue 中定义的 toAnalyse()
     **/
     const result = await this.getRequest(
      "/coal/analysis/getTwo/" + window.sessionStorage.getItem("coalIds")
     );
     //2 设置煤种参数
     //2.1 单煤参数
     result.forEach((resultItem, rId) => {
      dataOption.series
        .filter(function (seriesItem, sId) {
         return rId == sId;
        })
        .map(function (seriesItem) {
         seriesItem.name = window.sessionStorage.getItem(resultItem.coalId);
         seriesItem.data.push({
          value: resultItem.coalCAr,
          name: "收到基 C",
         });
         seriesItem.data.push({
          value: resultItem.coalHAr,
          name: "收到基 H",
         });
         seriesItem.data.push({
          value: resultItem.coalOAr,
          name: "收到基 O",
         });
         seriesItem.data.push({
          value: resultItem.coalSAr,
          name: "收到基 S",
         });
```

```
          seriesItem.data.push({
            value: resultItem.coalAAr,
            name: "收到基 A",
          });
          seriesItem.data.push({
            value: resultItem.coalMAr,
            name: "收到基 M",
          });
        });
      });
      //保存请求数据备用
      this.responseData = result;

      //2.2 默认比例计算混煤参数并设置
      this.getMixedData(dataOption, ratio);
      //注意必须在异步函数之内，先用请求到的数据默认初始化 funnelChart
      //否则在 mounted 内的语句作为主线程会先于异步请求执行，其初始化 funnelChart 先于异步函数
      //this.funnelChart.setOption(dataOption);
    },

    //计算混煤参数并设置
    getMixedData(dataOption, ratio) {
      //不能直接用 this.ratio 计算，否则会 undefined, 为什么我也不知道
      //不知道为什么用 dataOption.series.filter 拿到的 data 是 undefined, 如果不这样会 undefined
      let coal3Data = [];
      let count = 0;
      //计算混煤数据
      dataOption.series.reduce(function (lastCoal, currentCoal) {
        //技巧糖：设置锚点
        count++;
        if (count == 2) {
          //2.2.1 设置混合煤名称
          dataOption.series.find((item, id) => {
            if (id == 2) {
              //默认 30%煤种 1
              item.name =
                lastCoal.name +
                ratio +
                "% + " +
                currentCoal.name +
                (100 - ratio) +
```

```
            "%";
          }
        });
        //2.2.2 设置混合煤煤质数据
        lastCoal.data.forEach((coal1_data) => {
          //遍历找到单煤煤质对应的值
          currentCoal.data
            .filter(function (coal2_data) {
              return coal1_data.name == coal2_data.name;
            })
            //计算混煤质
            .map(function (coal2_data) {
              let coal1_value = parseFloat(coal1_data.value);
              let coal2_value = parseFloat(coal2_data.value);
              let mixed_value =
                (coal1_value * ratio + (100 - ratio) * coal2_value) / 100;
              mixed_value = Math.round(mixed_value * 100) / 100;
              coal3Data.push({
                value: mixed_value,
                name: coal1_data.name,
              });
            });
        });
      }
      return currentCoal;
    }, null);
    //为混煤 data 绑定计算的数据
    dataOption.series.find((item, id) => {
      if (id == 2) {
        item.data = coal3Data;
      }
    });
    //每次算完混煤数据后实时刷新 funnelChart 设置
    this.funnelChart.setOption(dataOption);
  },

  //监听 slider 拖动鼠标，鼠标松开后触发 updateMixed
  updateMixed() {
    //用已经获取完后台数据的 this.dataOption 再初始化 Vue 对象，计算 MixedData
    this.getMixedData(this.dataOption, this.ratio);
},
…
```

后端工程主要实现混煤数据的补充计算（如飞灰软化温度）和存储。通过在 maven 工

程中引入 JNA 依赖，并在对应 Service 层内定义调用第三方 DLL 库的实例方法计算飞灰软化温度，核心代码如下：

```
@ApiOperation(value ="计算混煤数据并存储")
@PostMapping("/add")
public ResponseBean addMixedCoal(@RequestBody MixedAnalysis mixedAnalysis){
    //计算软化温度 softT 并设置
    double softT = SoftTDll.softTDll.hrdyc(mixedAnalysis.getAshAl2o3Ar().doubleValue(),
            mixedAnalysis.getAshSio2Ar().doubleValue(),
mixedAnalysis.getAshCao().doubleValue(),
            mixedAnalysis.getAshFe2o3Ar().doubleValue(),
mixedAnalysis.getAshMgo().doubleValue(),

mixedAnalysis.getAshTio2().doubleValue(),mixedAnalysis.getAshSo3().doubleValue(),
            mixedAnalysis.getAshK2o().doubleValue());
    mixedAnalysis.setSoftT(new BigDecimal(softT).setScale(2,
BigDecimal.ROUND_HALF_UP));
    //如果混煤数据未存在则保存
    //否则抛出 SQLIntegrityConstraintViolationException
    if(mixedAnalysisService.save(mixedAnalysis)){
        return ResponseBean.success("保存混煤数据成功！");
    }
    return ResponseBean.error("操作失败！");
}
```

4.5 方案参数控制模块实现

环境参数控制模块由混煤标准检查和供电成本分析两个部分组成。该模块是用户和系统共同优化配煤方案的第一步，因此是配煤管理系统的核心模块之一。

4.5.1 混煤标准检查

基于响应式网页的设计模式和配煤管理系统的需求分析，用户在网页上要能随时调整设置的环境参数，并将其作为配煤方案的参数之一。因此，前端结构层需根据用户的设置，基于用户相应提示。只有当前混煤数据通过检查后，浏览器才会将当前混煤参数和环境参数将发送至后端，去初始化配煤方案，否则系统将强制用户修改环境参数或退回上一过程重新调整混煤数据。前端功能通过 StandSetting.vue 模块实现，以挥发分 Vdaf 的参数设置为例，其核心代码如下：

```
<!—结构层： 挥发分 Vdaf 标准设置-->
  <div class="dashboard-box">
    <el-tag type="success">挥发分 Vdaf</el-tag>
    <el-progress
```

```
        type="dashboard"
        :percentage="sVdafValue"
        :color="sVdafColors"
        ref="sVdafRef"
      ></el-progress>
      <div>
        <el-input-number
          v-model="sVdafValue"
          @change="sVdafChange"
          :min="20"
        ></el-input-number>
      </div>
    </div>

//行为层：挥发分 Vdaf 标准检查
sVdafChange(inputValue) {
    if (parseFloat(inputValue) <= parseFloat(this.result.mixedVDaf)) {
      this.sVdafValue = inputValue;
      this.updateStandard();
      return 1;
    } else {
      this.$message({
        type: "error",
        message: "超出最小 Vdaf，请重新设置标准或重选混煤方案",
      });
      return 0;
    }
  },

//服务层：系统自动检查 6 个标准
checkStandard() {
    //判断 6 个标准是否全满足
    if (
      this.sVdafChange(this.sVdafValue) &&
      this.sMarChange(this.sMarValue) &&
      this.sAarChange(this.sAarValue) &&
      this.sSadChange(this.sSadValue) &&
      this.qarnetChange(this.sQarValue) &&
      this.softTChange(this.sSoftTValue)
    ) {
      const scheme = {};
      scheme.maxAAr = this.sAarValue;
      scheme.maxMAr = this.sMarValue;
```

```
      scheme.maxSAd = this.sSadValue;
      scheme.minQnetAr = this.sQarValue;
      scheme.minVDaf = this.sVdafValue;
      scheme.mixedId = this.result.mixedId;
      //符合标准，存储标准数据到数据库
      this.postRequest("/mixed-analysis/add-scheme", scheme).then(
        (response) => {
          if (response) {
            window.sessionStorage.setItem("schemeId", parseInt(response));
          }
        }
      );
      this.$message({
        type: "success",
        message: "混煤符合煤质标准, 可进行成本分析",
      });
    }
  },
```

后端工程主要实现配煤方案的初始化，包括燃料成本、锅炉效率、日供电量、标煤煤耗和混煤煤耗等数据计算。为了简化方案设置，突出后文中同等环境方案中混煤配比的作用，本文设计的方案部分参数采用默认值，详见表 4-1。方案涉及的计算公式详见 2.3 节，后端核心代码详见附录。

表 4-1 配煤方案部分默认参数

厂用电率，%	主蒸汽流量，t/h	环境温度，℃	日满负荷发电时间，h
6.5	540	20	24

4.5.2 日供电成本分析

由于日供电成本受上网电价、脱硫脱硝成本等因素的影响，因此配煤方案的日供电成本分析需要用户在结构层输入上网电价、液氨单价、石灰石单价，随后由浏览器发送成本参数到后端执行相应的成本计算，并更新相应配煤方案的参数。

前端由 CostSetting.vue 模块实现，其展示层代码如下：

```
<template>
  <div>
    <el-card>
      <div class="input-price">
        <el-tag type="danger">混煤单价(元/t)</el-tag>
        <el-input-number
          :precision="1"
          :step="0.1"
```

```
      :min="0"
      v-model="mixedPrice"
      disabled
    >
    </el-input-number>
  </div>
  <div class="input-price">
    <el-tag type="warning">石灰石单价(元/t)</el-tag>
    <el-input-number
      :precision="1"
      :step="0.1"
      :min="0"
      v-model="limePrice"
    >
    </el-input-number>
  </div>
  <div class="input-price">
    <el-tag>液氨单价(元/t)</el-tag>
    <el-input-number
      :precision="2"
      :step="0.01"
      :min="0"
      v-model="nh3Price"
    >
    </el-input-number>
  </div>
  <div class="input-price">
    <el-tag type="success">上网电价(元/kwh)</el-tag>
    <el-input-number
      :precision="4"
      :step="0.001"
      :min="0"
      v-model="powerPrice"
    >
    </el-input-number>
  </div>
  <el-button type="success" @click="setPrice" class="analysis-btn"
    >分析成本</el-button
  >
</el-card>
<el-card>
  <div class="pie-chart" ref="pieChart"></div>
</el-card>
```

```
  </div>
</template>
```

后端由控制层的 CoalSchemeController 类计算并更新配煤方案参数，核心代码如下：

```
@ApiOperation(value ="计算成本占比随配比变化")
@PostMapping("/get-data")
public CoalScheme getSchemeData(@RequestBody Price price){

    CoalScheme scheme = coalSchemeService.getById(price.getSchemeId());
    if(scheme != null){
        //计算脱硫、脱硝成本、总成本
        double limePrice = price.getLimePrice();
        double nh3Price = price.getNh3Price();
        double powerPrice = price.getPowerPrice();
        //脱硫成本
        double limeCost = limePrice * scheme.getLimeConsump().doubleValue();
        //脱硝成本
        double nh3Cost = nh3Price * scheme.getNh3Consump().doubleValue();
        //总成本 = 燃料成本 + 脱硫成本 + 脱硝成本 + 厂用电成本
        //日供电成本(元/day)
        double powerCostDay = scheme.getMixedConsumpKwh().doubleValue() *
scheme.getPowerSupply().doubleValue() *
                0.01 * scheme.getMixedPrice().doubleValue() + limeCost + nh3Cost +
                100 * scheme.getPowerGenerate().doubleValue() *
scheme.getRatioOwn().doubleValue() * powerPrice;
        //日供电成本(元/kwh)
        double powerCostKwh = powerCostDay / (scheme.getPowerSupply().doubleValue() *
10000);

        //存储数据
        scheme.setLimePrice(new
BigDecimal(limePrice).setScale(1,BigDecimal.ROUND_HALF_UP));
        scheme.setNh3Price(new
BigDecimal(price.getNh3Price()).setScale(2,BigDecimal.ROUND_HALF_UP));
        scheme.setPowerPrice(new
BigDecimal(price.getPowerPrice()).setScale(4,BigDecimal.ROUND_HALF_UP));
        scheme.setLimeCost(new
BigDecimal(limeCost).setScale(1,BigDecimal.ROUND_HALF_UP));
        scheme.setNh3Cost(new
BigDecimal(nh3Cost).setScale(1,BigDecimal.ROUND_HALF_UP));
        scheme.setPowerCostKwh(new
BigDecimal(powerCostKwh).setScale(4,BigDecimal.ROUND_HALF_UP));
        scheme.setPowerCostDay(new
BigDecimal(powerCostDay).setScale(1,BigDecimal.ROUND_HALF_UP));
```

```
            coalSchemeService.updateById(scheme);
            return scheme;
        }

        return null;
}
```

4.6 配煤方案对比模块实现

由于第 2 章的需求分析可知，配煤管理系统的目的之一是找到生产效益最高，即日供电成本最低的配煤方案。为了研究相同的两种单煤在掺烧时，混煤配比的影响，本文进行日供电成本对比的配煤方案需满足以下条件：

1) 在满足用户设定的相同混煤标准的前提之下，煤种 1 比率不同的配煤方案

2) 其他配煤自定义参数（如最大干燥无灰基挥发分标准、石灰石单价等）和机组参数（如日发电量、锅炉效率等）均与当前配煤方案相同的其他配煤方案。

以下称这些方案为同等环境方案。后端把同等环境方案的数据返回到前端，并由前端在展示层输出方案不同组分日供电成本的占比饼图，以及日供电成本随配比变化的趋势图。前端部分代码如下：

```
mounted() {
  //绑定 echarts 要挂载的 DOM 对象
  this.pieChart = this.$echarts.init(this.$refs.pieChart);
  const pieChart = this.pieChart;
  setTimeout(function () {
    let option = {
      legend: {},
      tooltip: {
        trigger: "axis",
        showContent: false,
      },
          //数据区域缩放控件
      dataZoom: [
        {
          id: "dataZoomX",
          type: "slider",
          xAxisIndex: [0],
          filterMode: "filter",
        },
        {
```

```
    id: "dataZoomY",
    type: "slider",
    yAxisIndex: [0],
    filterMode: "empty",
  },
],
dataset: {
  source: [],
},
xAxis: {
  type: "category",
  name: "煤种 1 占比: 1/10",
  nameLocation: "center",
},
yAxis: {
  gridIndex: 0,
  name: "日供电成本：元/天",
},
grid: { top: "55%" },
series: [
  {
    type: "line",
    smooth: true,
    seriesLayoutBy: "row",
    emphasis: { focus: "series" },
  },
  …
  {
    type: "pie",
    id: "pie",
    radius: "30%",
    center: ["50%", "25%"],
    emphasis: { focus: "data" },
    label: {
      formatter: "{b}: {@2} (%)",
    },
    encode: {
      itemName: "product",
      value: "2",
      tooltip: "2",
    },
  },
],
```

```
    };

    pieChart.on("updateAxisPointer", function (event) {
      var xAxisInfo = event.axesInfo[0];
      if (xAxisInfo) {
        var dimension = xAxisInfo.value + 1;
        pieChart.setOption({
          series: {
            id: "pie",
            label: {
              formatter: "{b}: {@[" + dimension + "]} (%)",
            },
            encode: {
              value: dimension,
              tooltip: dimension,
            },
          },
        });
      }
    });
    pieChart.setOption(option);
});
```

5 系统使用说明及测试

5.1 测试参数

表 5-1 测试登录账户

用户编号	用户名	用户密码	用户权限
1	Jone	123	管理员

表 5-2 方案涉及的部分单煤信息

煤种名称	煤种单价，元/t	干燥无灰基挥发分，%	收到基水分，%	收到基灰分，%	空干基硫分，%	收到基低位发热量，kcal
（范平）梅花井	336	44.94	8.32	17.05	0.30	4876.30
（范平）羊肠湾	338	34.71	13.20	13.44	0.44	4935.30

表 5-3 设定的混煤数据标准值

最小干燥无灰基挥发分，%	最大收到基水分，%	最大收到基灰分，%	最大空干基硫分，%	最小收到基低位发热量，kcal	最小飞灰软化温度，℃
20	12	20	0.68	4400	1300

表 5-2 混煤方案部分数据

煤种 1 占比	干燥无灰基挥发分，%	收到基水分，%	收到基灰分，%	空干基硫分，%	飞灰软化温度，℃	混煤单价，元/t	液氨单价，元/t	石灰石单价，元/t
0.3	37.78	11.74	14.52	0.42	1332.68	337.2	2457	53.8

5.2 使用及测试过程

5.2.1 用户登录

用户需先将包含用户名、密码、用户权限、是否启用的账户信息录入数据库，才能使用系统资源。其中，录入数据库的用户名和密码必须非空，以提高系统安全性。用户输入非法或输入与数据库中不一致的账户信息时，都将被系统拒绝访问（如图 5-1）。只有当用

户输入与数据库中完全一致的账户信息时，用户成功登录系统（如图 5-2），之后用户可使用账户授权的系统功能。

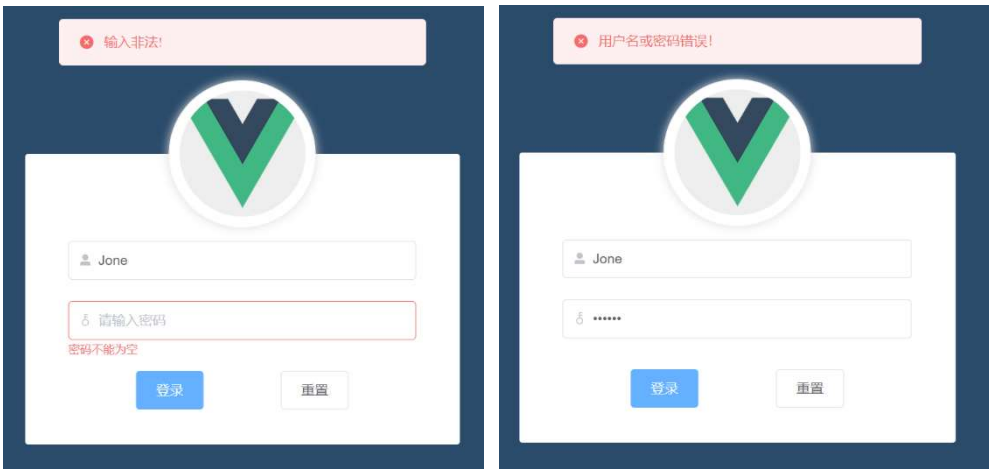


图 5-1 系统拒绝访问的情况

图 5-2 用户成功登录界面

5.2.2 煤种预选

用户点击煤种预选选项卡，即可跳转到煤种预选页面（如图 5-3）。该页面展示了煤仓中默认的所有煤种信息。当用户权限为管理员时，用户才可修改默认的煤种信息。任何用户都给可以增删改自定义的煤种信息（如图 5-4）。

图 5-3 煤种预选界面

图 5-4(a) 新建煤种信息

图 5-4(b) 删除煤种信息

使用管理员账户修改默认煤种界面如图 5-5，修改完成后点击“确定”按钮，即永久修改煤仓中的默认煤种信息。

图 5-5 管理员权限下的修改煤种信息界面

用户需先在复选框内勾选两种单煤，然后点击“分析煤质含数据”按钮，方可跳转至“煤质分析”界面（如图 5-6），否则系统将会提示用户进行煤种预选。测试时选择的两种单煤的煤种编号分别为 53 和 55。

图 5-6（a） 选择两种单煤分析煤质

图 5-6（b） 测试时选定的两种单煤

5.2.3 煤质分析

当用户点击“分析煤质数据”按钮后，界面将跳转至“煤质分析”界面。此时，系统会自动调取用户选取的 2 种单煤的煤质数据，并以“金字塔型”的数据可视化方式，展示对应煤种的元素分析数据（如图 5-7）。每一个数据金字塔，自上而下以升序形式表示了对应煤种中的不同元素含量。当用户鼠标悬停在数据图上方时，系统会响应式地显示当前数据所属的煤种及其数据值。

图 5-7 金字塔型数据图

本文通过调研企业常见配煤比例，设定系统的默认 2 种单煤配比为 7：3。用户仍可根据自身配煤的需求，拖动混煤比例调节滑块改变混煤比例（如图 5-8），系统将实时计算并显示用户指定配比下的混煤煤质信息，提供给用户实时、生动的数据可视化服务。本文测试时，采用系统默认的混煤配比，即：（范平）梅花井：（范平）羊肠湾 ＝3:7

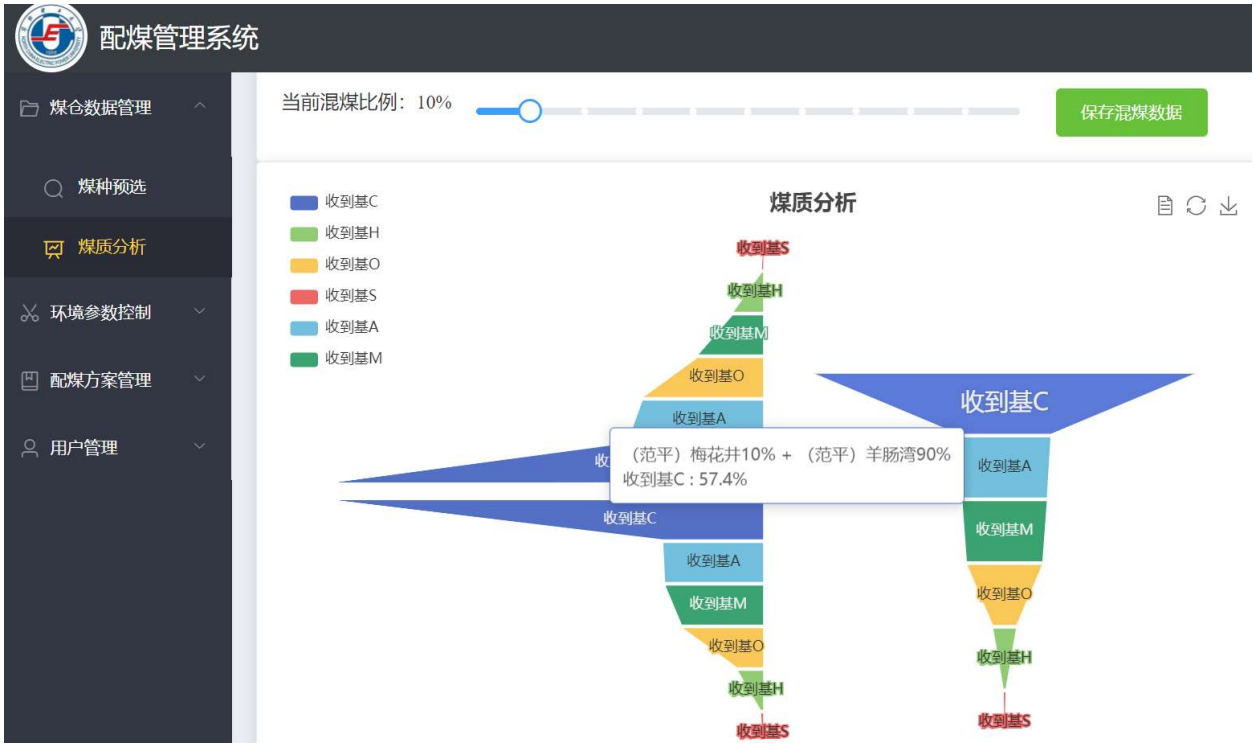


图 5-8（a）（范平）梅花井：（范平）羊肠湾 = 1:9

图 5-8（b）（范平）梅花井：（范平）羊肠湾 = 9:1

此外，用户还可以查看并下载文本格式的煤质分析数据（如图 5-9），方便用户进行校验和记录。

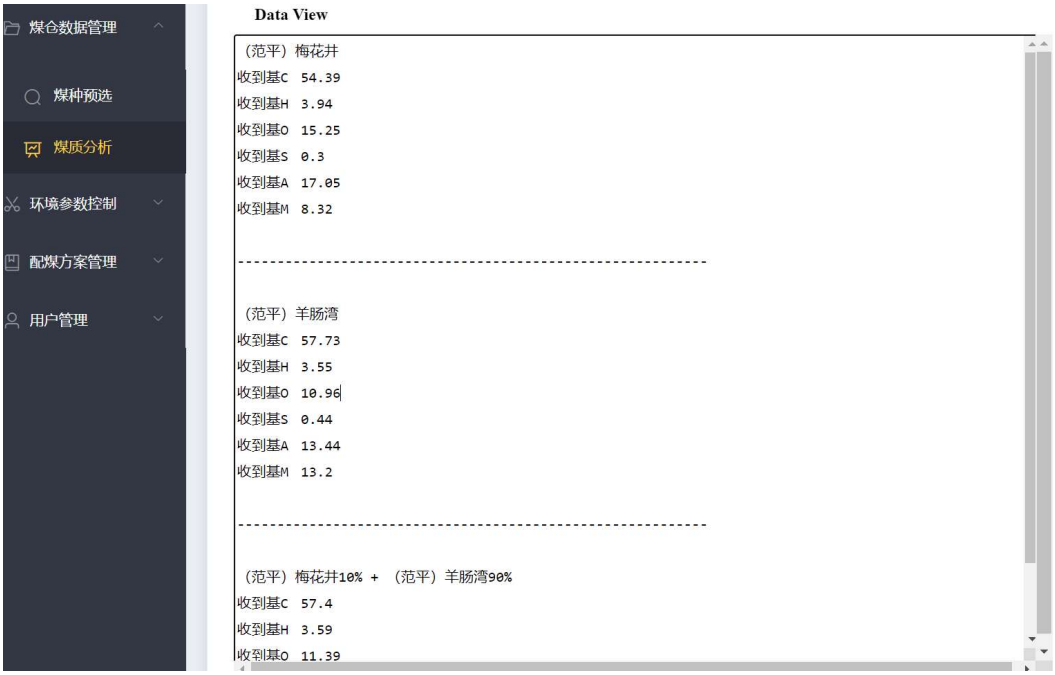


图 5-9 单煤和混煤的煤质分析数据

用户需先点击“保存混煤数据”按钮，保存一定配比的混煤数据（如图 5-10），此后系统将自动跳转到混煤标准检查界面，否则系统将采用默认的混煤配比进行混煤标准检查。

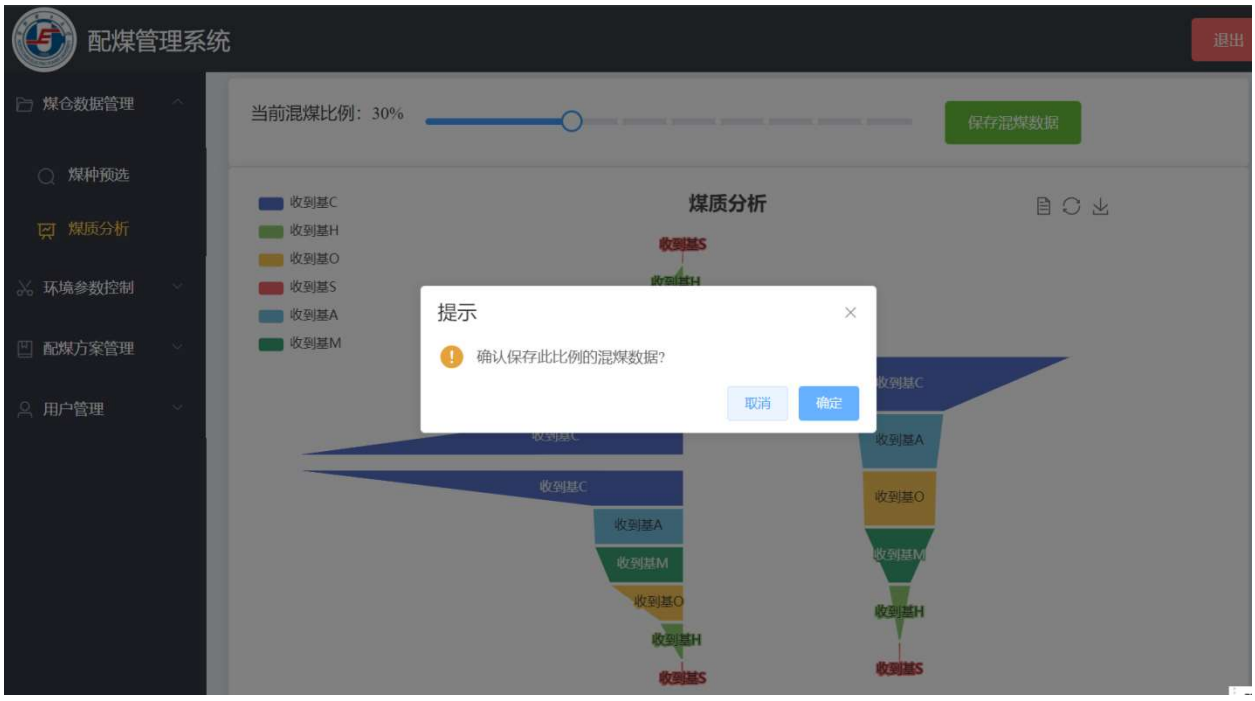


图 5-10 保存指定配比的混煤数据

5.2.4 混煤标准检查

图 5-11 混煤标准检查界面

首先，用户根据机组运行和燃烧排放标准中对混煤成分的一系列要求，自行设置包括煤挥发分含量、水分含量、灰分含量、硫分含量、低位发热量、飞灰软化温度这 6 项标准（如图 5-12），测试时的混煤标准数据可查看表 5-2。

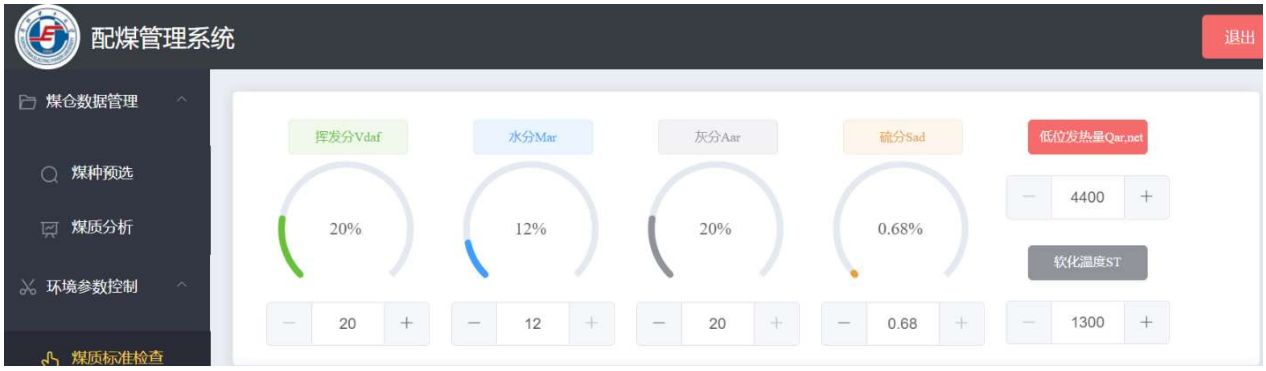


图 5-12 混煤标准设置面板

用户设置的 6 项标准值将实时显示在“混煤标准对比图”中，图中的每一行分别表示一项标准值，从上到下依次为最小干燥无灰基挥发分标准、最大收到基水分标准、最大收到基灰分标准、最大空干基硫分标准、最小收到基低位发热量标准、最小飞灰软化温度标准。图中的蓝色图标表示当前的混煤数据，绿色图标表示设定的标准数据（如图 5-13）。

图 5-13 混煤标准对比图

其次，用户设定的标准值将实时更新到对比图中，方便用户查看。如果标准设置不符合当前混煤数据，系统则提示用户修改标准或重新选择配煤方案（图 5-14）。为了确保混煤符合设定标准，同时方便用户进行标准检查，用户可点击“自动检查混煤标准”按钮，由系统自动检查当前混煤是否符合标准。

图 5-14 混煤标准检查不通过

当混煤满足所有设定的标准后，系统将提示用户可进行不同方案的成本分析（如图 5-

15），具体分析可在供电成本分析中查看。

图 5-15 混煤标准检查通过

5.2.5 供电成本分析

用户在完成混煤标准设定和检查后，系统将自动跳转至“供电成本分析”页面。页面由成本设定面板和成本占比趋势图构成（如图 5-16）。用户需先在成本设定面板设定成本参数，设定完成后点击“分析”成本按钮，系统将自动搜集同等环境的方案，并计算在 178MW 的负荷下、环境温度为 20℃时，采用相应配煤方案的供电成本，并对不同方案进行对比分析。

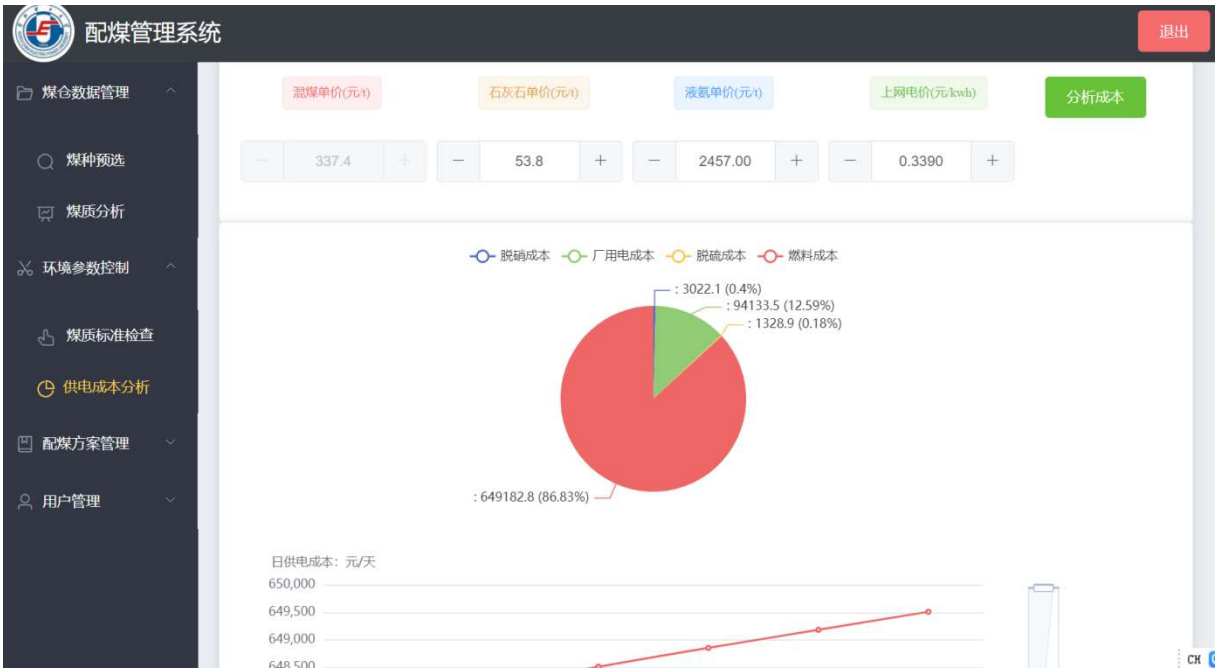


图 5-16 供电成本分析界面

成本设定面板（如图 5-17）包括混煤单价、石灰石单价、液氨单价、上网电价四项，其中混煤单价由系统计算得到，无序用户设定，其余三项由用户根据实际市场价格设定。

图 5-17 成本设定面板

用户在成本设定面板上设置好成本后，点击“分析成本”按钮，系统会自动去数据库中查找同等环境方案。所有同等环境方案的信息将显示在成本占比趋势图中（如图 5-18）。横轴所取的每一个值表示一个配煤方案，一个方案对应 4 个供电成本数值。从图中我们可以发现，煤种 1 占比从 1 到 9 并非都能取到，因为在同等环境方案中部分混煤配比不满足用户设定的混煤标准。

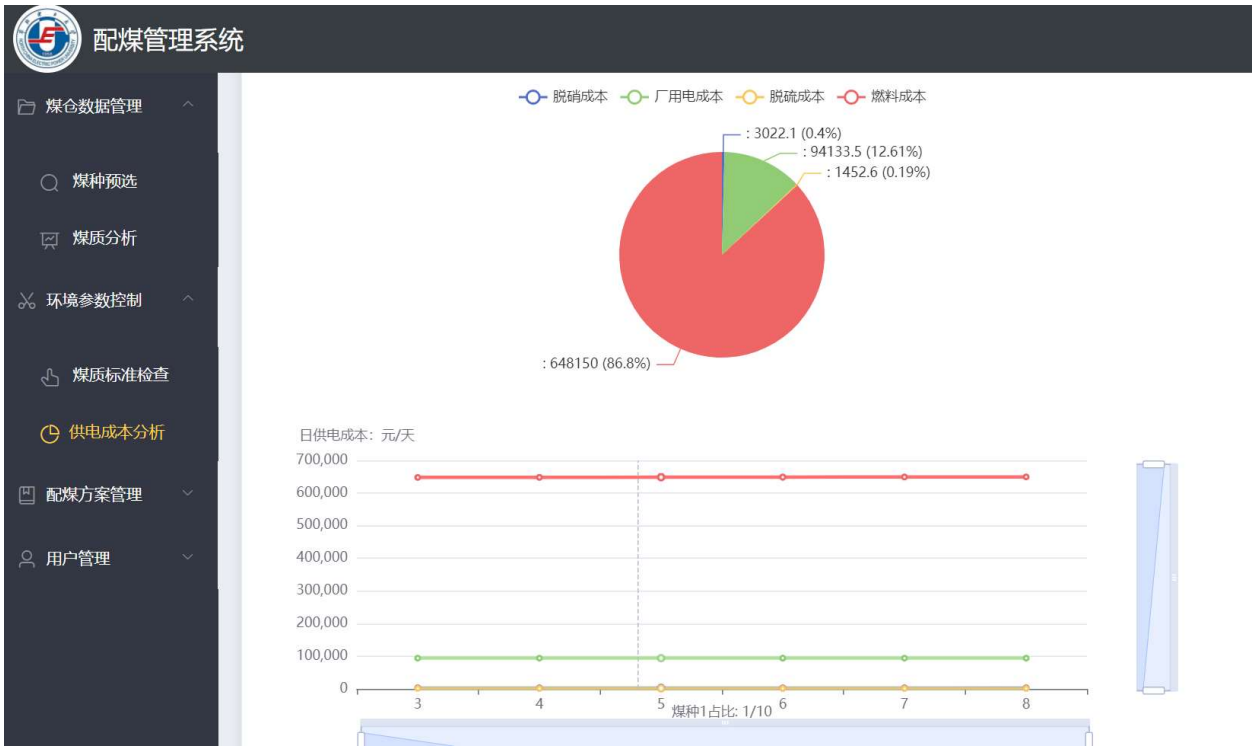
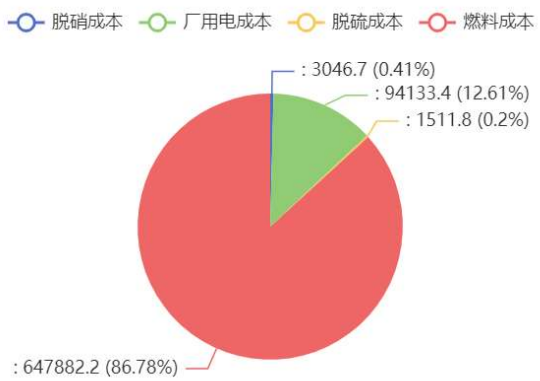
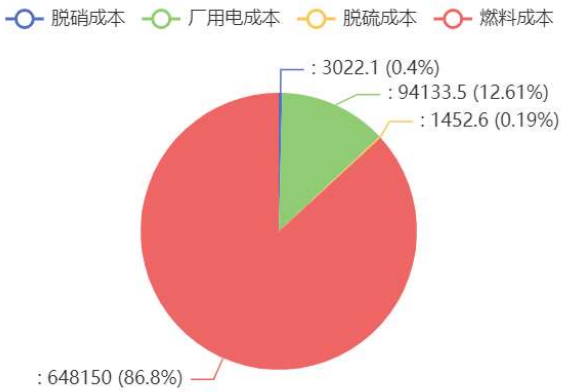


图 5-18 同等环境方案的成本占比趋势图

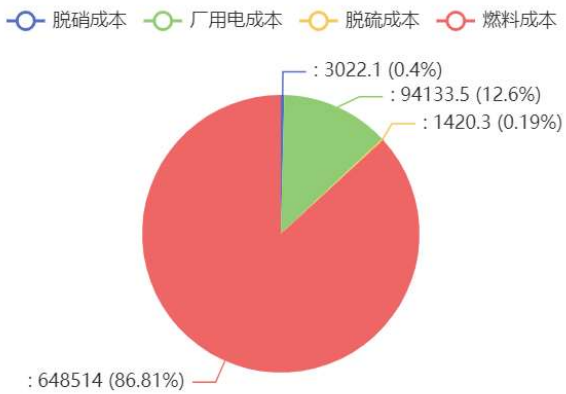
成本占比趋势图由日供电成本饼图和日供电成本折线图组成。饼图显示的是一个同等环境方案的供电总成本及其组成部分的占比情况（如图 5-19（a））。供电总成本主要由燃煤成本、厂用电成本、脱硝成本、脱硫成本四部分组成，用户可以通过点击折线图中代表不方案的数据点来动态显示方案的供电成本占比情况（如图 5-19（b~f））。



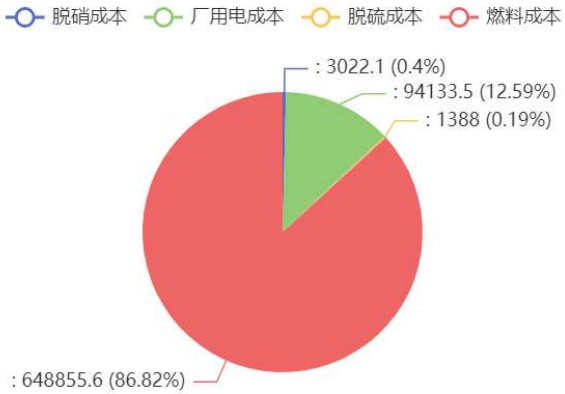
(a) （范平）梅花井：（范平）羊肠湾 = 3:7



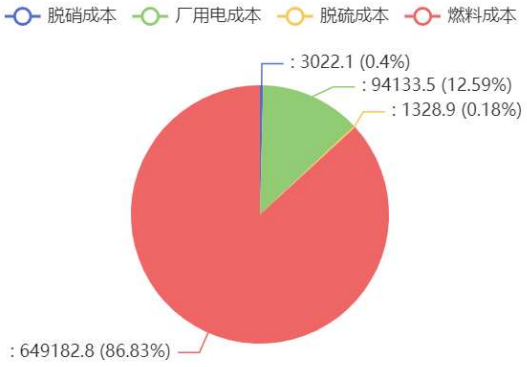
(b) （范平）梅花井：（范平）羊肠湾 = 4:6



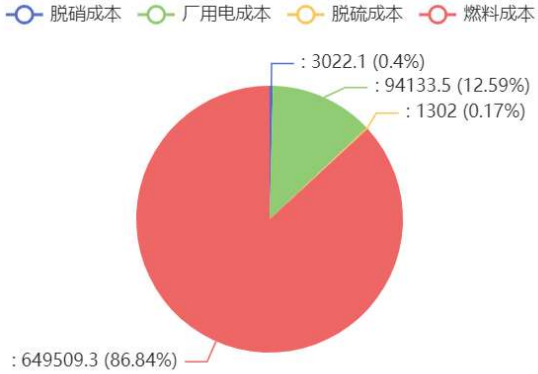
(c) （范平）梅花井：（范平）羊肠湾 = 5:5



(d) （范平）梅花井：（范平）羊肠湾 = 6:4



(e) （范平）梅花井：（范平）羊肠湾 = 7:3



(f) （范平）梅花井：（范平）羊肠湾 = 8:2

图 5-19 同等环境的供电成本占比情况

用户可以通过拖动数据区域的缩放滑块，进一步查看随煤种 1 占比改变时，不同日供电成本组分的变化趋势。从图 5-20 和表 5-4 可以看出，在日供电量、厂用电率、混煤标准、单煤煤种等条件均相同的情况，仅改变煤种 1 的占比，燃料成本以百元的量级不断增加，厂用电成本和脱硝成本几乎不变，脱硫成本以十元的量级减少。因此对于同等环境的配煤方案，选择混煤配比为：（范平）梅花井：（范平）羊肠湾 = 3:7 的方案时，生产效益最高。

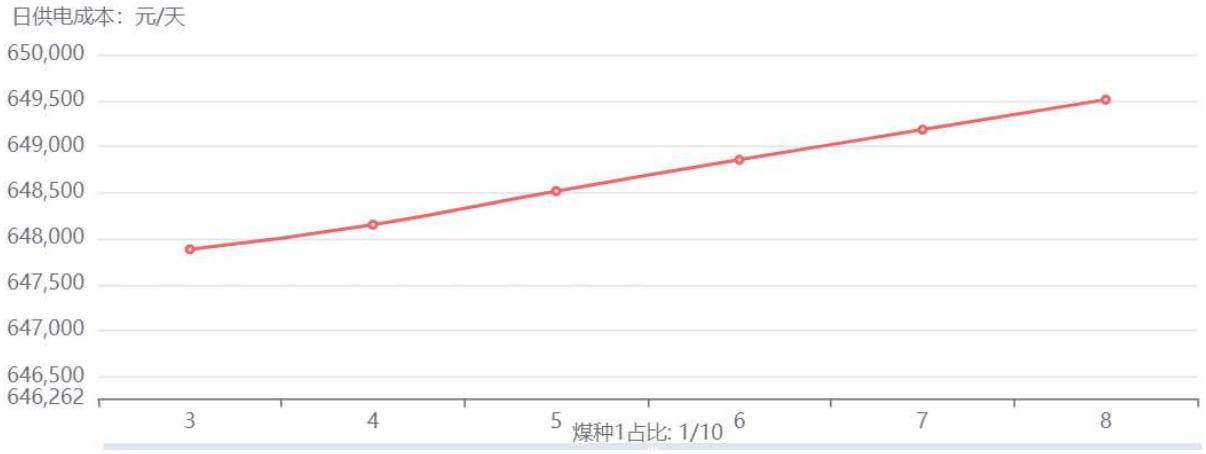


图 5-20（a）日燃料成本变化

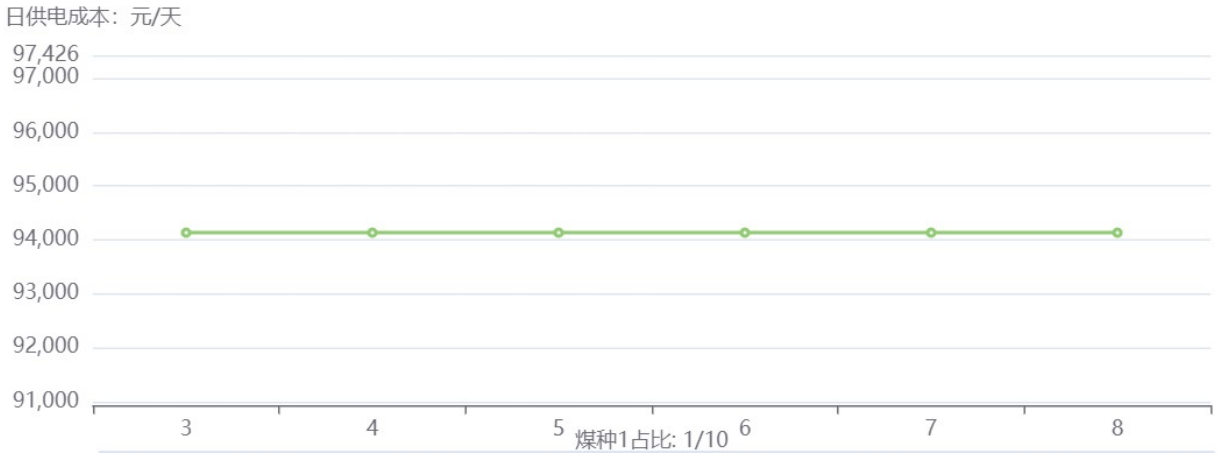


图 5-20（b）日厂用电成本变化

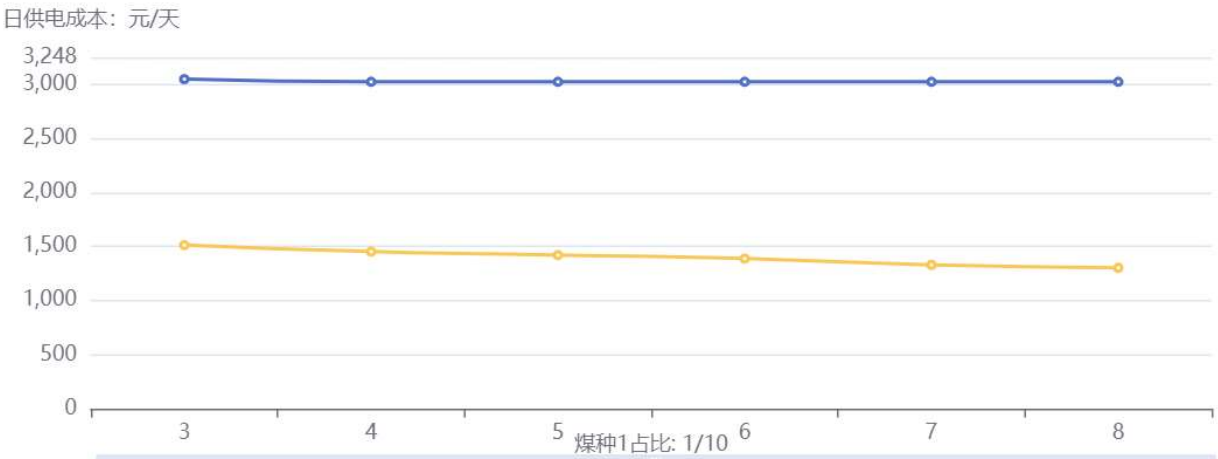


图 5-20（c）日脱硝和脱硫成本变化

表 5-4 同等环境方案的日供电成本

煤种 1 占比	燃煤成本（元/天）	厂用电成本（元/天）	脱硝成本（元/天）	脱硫成本（元/天）	供电总成本（元/天）
0.3	647882.2	94133.4	3046.7	1511.8	746574.1
0.4	648150.0	94133.5	3022.1	1452.6	746758.2
0.5	648514.0	94133.5	3022.1	1420.3	747089.9
0.6	648855.6	94133.5	3022.1	1388.0	747399.2
0.7	649182.8	94133.5	3022.1	1328.9	747667.3
0.8	649509.3	94133.5	3022.1	1302.0	747966.9

6 总结与展望

6.1 测试结论

通过分析第 5 章最后的测试数据和结果，可以发现

1) 对于同等环境方案，单位质量混煤中煤种 1 占比的越大，方案的日总供电成本差距最高可达近千元；
2) 尽管单煤之间水分、灰分、硫分含量有一定差距，但当单煤单价的差异不大时，造成日总供电成本差异的主要因素为燃料单价，即单煤的挥发分含量。厂用电成本和脱硫脱硝成本占比很小。
3) 在优化配煤方案时，对于同等环境方案，在满足混煤标准的前提下，尽可能增大挥发分含量较大的单煤比例将有利于降低日总供电成本。

6.2 工作总结

本文主要研究了火电企业配煤管理系统的设计架构和互联网化。配煤管理系统作为火电企业现代化建设不可或缺的一环，其智能化和移动互联化将极大地提高企业生产效率。因此，本文从配煤意义入手，通过分析配煤系统的研究现状，确立了企业对配煤管理的功能需求，从而建立起基于前后端分离的系统技术路线，最后针对具体功能进行系统设计架构，实现并测试了基于 Vue 和 SSM 的配煤管理系统的完整功能。

本系统界面简洁美观，符合用户的实际生产逻辑。对本文定义的同等环境配煤方案，系统以得到最低的方案日总供电成本为目标，一方面给企业降低生产成本提供量化参考，另一方面将设计配煤方案的所经历的各流程可视化，让工作人员能更深入地理解配煤方案中的参数设置和优化原理，具有应用于实际生产的价值。

6.3 未来工作展望

本文通过分析火电企业的实际配煤生产需求，开发出了一款作为互联网产品的配煤指导系统，但从搜索算法的优化还是系统功能的延伸来看，该系统仍有很大的发展空间：

1) 搜索算法的优化。系统目前只比较了同等环境方案下，日供电成本的占比情况和随配比改变的成本变化趋势，应用场景单一。未来可针对非同等环境方案下的多应用场景，引入不同的方案搜索算法，为用户提供更多配煤参考信息；
2) 系统功能的延伸。为进一步结合企业的实际生产需求，增加用户友好性，系统还应考虑开发用户管理、机组能耗分析等模块，以培养用户粘性。

参考文献

[1] DU S-W, CHEN W-H, LUCAS J A. Pretreatment of biomass by torrefaction and carbonization for coal blend used in pulverized coal injection [J]. Elsevier, 2014, 161.

[2] 谢海深, 孟军波, 刘永新, et al. 配煤优化模型及其专家系统的设计 [J]. 计算机与应用化学, 2007, (05): 621-4.

[3] BARRETT R E E A. Engineering Foundation Third Conference on Slagging and Fouling Due to Impurities in Combustion Gases [J]. Slagging and Fouling as Related to Coal and Boiler Parameters, 1984.

[4] 曾琴琴. 火电厂配煤优化方法研究 [D]; 哈尔滨工业大学, 2006.

[5] ALEKHNOVICH A, BOGOMOLOV V. Use of coal blends at thermal power plants [J]. Power Technology and Engineering, 2010, 44(3): 213-9.

[6] BAYER A K, RADEMACHER M, RUTHERFORD A. Development and perspectives of the Australian coal supply chain and implications for the export market [J]. Zeitschrift für Energiewirtschaft, 2009, 33(3): 255-67.

[7] J.R. K. Coal and Coal/Biomass-Based Power Generation [Z]. Global Climate Change - The Technology Challenge. Advances in Global Change Research; Springer, Dordrecht. 2011

[8] 成玉琪，杜铭华，李文华，徐振刚. 优化动力配煤是符合中国燃煤特点的洁净煤技术 [J]. 洁净煤技术, 1997, (01): 9-12+27.

[9] 唐春潮, 陈鸿复, 高秋潮. 专家系统与模式识别在炼焦配煤中的应用 [J]. 燃料与化工, 1995, 26(06): 273-8.

[10] 戴财胜. 动力配煤的理论与应用研究 [D]; 中国矿业大学（北京）, 2000.

[11] 马革非. 动力配煤最优配煤指标及配煤方案的确定 [J]. 中州煤炭, 2000, (05): 3.

[12] 靳智平, 李东雄. 用 VB6.0 开发动力配煤适用软件 [J]. 电力学报, 2002, (03): 186-8+94.

[13] 梁景坤. 基于煤质工程分析的动力配煤优化研究 [D]; 华北电力大学（北京）, 2004.

[14] 邓俊. 炼焦配煤智能优化模型及其应用研究 [D]; 中南大学, 2007.

[15] 李超斌. 配煤过程控制与智能优化系统的设计与应用 [D]; 中南大学, 2007.

[16] 郑志军, 郑守淇. 用基于实数编码的自适应遗传算法进化神经网络 [J]. 计算机工程与应用, 2000, (09): 36-7.

[17] 李胜. 包钢配煤系统技术经济研究 [J]. 钢铁, 2003, (11): 75-7.

[18] 张曦木. 火电厂燃煤配比与采购计划优化方法与系统开发 [D]; 东北大学, 2014.

[19] 樊泉桂等. 锅炉原理（第二版） [M]. 中国电力出版社, 2014.

[20] 王春波, 陈亮, 任育杰, et al. 基于高温除尘的燃煤电站多污染物协同控制技术 [J]. 华

北电力大学学报(自然科学版), 2017, 44(06): 82-92.
[21]YOU 等 E. Vue.js 官方文档 [EB/OL] 2021, https://cn.vuejs.org.
[22]饿了么团队. Element UI 官方文档 [EB/OL] 2021, https://element.eleme.cn/#/zh-CN.
[23]Echarts 官方文档 [EB/OL]. 2021, https://echarts.apache.org/zh/api.html#echarts
[24]TEAM T S. Spring 官方文档 [EB/OL] 2021, https://spring.io/guides.
[25]MyBatis-Plus 官方文档 [EB/OL]. 2021[https://baomidou.com/.
[26]杨冠宝. 阿里巴巴 Java 开发手册(第 2 版) [M]. 电子工业出版社, 2020.
[27]FORTA 英 B. MYSQL 必知必会 [M]. 人民邮电出版社, 2009.
[28]邢诚. 动力配煤原理在火力发电厂中的应用 [D]; 华北电力大学（北京）, 2006.
[29]李超斌, 吴敏. 涟钢焦化配煤智能优化配比控制系统 [J]. 装备制造技术, 2007, (02): 76-8.
[30]李悦. 煤化工企业配煤系统的设计与实现 [D]; 西安科技大学, 2014.
[31]焦涵宇. 基于机组负荷的火电厂优化配煤系统分析与设计 [D]; 华北电力大学, 2015.
[32]张广宏. 数字燃煤环境下发电企业燃煤库存管理优化研究 [D]; 华北电力大学(北京), 2016.
[33]李婷. 气化用煤配煤模型及动态配煤系统研究 [D]; 西安科技大学, 2017.
[34]韩云昕. 火力发电企业数字化煤场的设计与实现 [D]; 华中科技大学, 2019.
[35]徐鹏涛. 基于 Vue 的前端开发框架的设计与实现 [D]; 山东大学, 2020.
[36]张卫华. 基于 SSM 的权限管理系统及数据可视化 [D]; 北京邮电大学, 2020.

附录 A　后端部分代码

A-1 持久层实体类

第 4 章中系统后端持久层中，用于操作 coal_inplo 表和 coal_analysis 表中的记录的实体 CoalInplo 类和 CoalAnalysis 类来：

```
//CoalInplo 类
package buzz.freelearner.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.experimental.Accessors;

import java.io.Serializable;
import java.math.BigDecimal;

/**
 * <p>
 *  煤种查询
 * </p>
 *
 * @author eva
 * @since 2021-04-24
 */
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@ApiModel(value="CoalInplo 对象", description="煤种查询")
public class CoalInplo implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "煤种编号")
    @TableId(value = "coal_id", type = IdType.AUTO)
    private Integer coalId;

    @ApiModelProperty(value = "煤种名称")
    private String coalName;
```

```
    @ApiModelProperty(value = "煤种价格")
    private BigDecimal coalPrice;

    @ApiModelProperty(value = "煤种储量")
    private BigDecimal coalLoad;

    @ApiModelProperty(value = "煤种产地")
    private String coalOrigin;
}

//CoalAnalysis 类
package buzz.freelearner.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.experimental.Accessors;

import java.io.Serializable;
import java.math.BigDecimal;

/**
 * <p>
 *  煤质管理
 * </p>
 *
 * @author eva
 * @since 2021-04-24
 */
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@ApiModel(value="CoalAnalysis 对象", description="煤质管理")
public class CoalAnalysis implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "煤种编号")
    @TableId(value = "coal_id", type = IdType.AUTO)
    private Integer coalId;
```

```
@ApiModelProperty(value = "收到基全水分")
private BigDecimal coalMAr;

@ApiModelProperty(value = "空干基水分")
private BigDecimal coalMAd;

@ApiModelProperty(value = "收到基灰分")
private BigDecimal coalAAr;

@ApiModelProperty(value = "可燃基挥发分")
private BigDecimal coalVDaf;

@ApiModelProperty(value = "空干基挥发分")
private BigDecimal coalVAd;

@ApiModelProperty(value = "低位发热量")
private BigDecimal coalQnet;

@ApiModelProperty(value = "收到基碳")
private BigDecimal coalCAr;

@ApiModelProperty(value = "收到基氢")
private BigDecimal coalHAr;

@ApiModelProperty(value = "收到基氮")
private BigDecimal coalNAr;

@ApiModelProperty(value = "收到基全硫")
private BigDecimal coalSAr;

@ApiModelProperty(value = "收到基氧")
private BigDecimal coalOAr;

@ApiModelProperty(value = "灰变形温度")
private BigDecimal ashDtAr;

@ApiModelProperty(value = "灰软化温度")
private BigDecimal ashStAr;

@ApiModelProperty(value = "灰流动温度")
private BigDecimal ashFtAr;
```

```
    @ApiModelProperty(value = "二氧化硅")
    private BigDecimal ashSio2Ar;

    @ApiModelProperty(value = "三氧化二铝")
    private BigDecimal ashAl2o3Ar;

    @ApiModelProperty(value = "三氧化二铁")
    private BigDecimal ashFe2o3Ar;

    @ApiModelProperty(value = "二氧化钛")
    private BigDecimal ashTio2;

    @ApiModelProperty(value = "氧化钙")
    private BigDecimal ashCao;

    @ApiModelProperty(value = "氧化镁")
    private BigDecimal ashMgo;

    @ApiModelProperty(value = "三氧化硫")
    private BigDecimal ashSo3;

    @ApiModelProperty(value = "氧化钾")
    private BigDecimal ashK2o;

    @ApiModelProperty(value = "氧化钠")
    private BigDecimal ashNa20;
}
```

A-2 控制层功能类

第 4 章中，根据预选取的两种单煤数据和煤种配比，实现混煤数据计算和存储、初步储存配煤方案等功能：

```
package buzz.freelearner.server.controller;

import buzz.freelearner.server.dll.Fanping50Dll;
import buzz.freelearner.server.dll.SoftTDll;
import buzz.freelearner.server.mapper.CoalSchemeMapper;
import buzz.freelearner.server.pojo.CoalScheme;
import buzz.freelearner.server.pojo.MixedAnalysis;
import buzz.freelearner.server.pojo.ResponseBean;
import buzz.freelearner.server.service.ICoalInploService;
import buzz.freelearner.server.service.ICoalSchemeService;
```

```
import buzz.freelearner.server.service.IMixedAnalysisService;
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import io.swagger.annotations.ApiOperation;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.math.BigDecimal;

/**
 * <p>
 *  混煤煤质管理  前端控制器
 * </p>
 *
 * @author eva
 * @since 2021-05-08
 */
@RestController
@CrossOrigin
@RequestMapping("/mixed-analysis")
public class MixedAnalysisController {

    //注入类
    //注意每个注入类都要加@Autowired
    @Autowired
    private IMixedAnalysisService mixedAnalysisService;
    @Autowired
    private ICoalInploService coalInploService;
    @Autowired
    private ICoalSchemeService coalSchemeService;
    @Autowired
    private CoalSchemeMapper coalSchemeMapper;


    @ApiOperation(value ="计算混煤数据并存储")
    @PostMapping("/add")
    public ResponseBean addMixedCoal(@RequestBody MixedAnalysis mixedAnalysis){
        //计算软化温度 softT 并设置
        double                                          softT                                          =
SoftTDll.softTDll.hrdyc(mixedAnalysis.getAshAl2o3Ar().doubleValue(),
                mixedAnalysis.getAshSio2Ar().doubleValue(),
mixedAnalysis.getAshCao().doubleValue(),
                mixedAnalysis.getAshFe2o3Ar().doubleValue(),
mixedAnalysis.getAshMgo().doubleValue(),
```

```
mixedAnalysis.getAshTio2().doubleValue(),mixedAnalysis.getAshSo3().doubleValue(),
                    mixedAnalysis.getAshK2o().doubleValue());
        mixedAnalysis.setSoftT(new                    BigDecimal(softT).setScale(2,
BigDecimal.ROUND_HALF_UP));
        //如果混煤数据未存在则保存
        //否则抛出 SQLIntegrityConstraintViolationException
        if(mixedAnalysisService.save(mixedAnalysis)){
            return ResponseBean.success("保存混煤数据成功！");
        }
        return ResponseBean.error("操作失败！");
    }

    @ApiOperation(value ="根据混煤名称获取数据")
    @GetMapping("/get")
    public MixedAnalysis getMixedCoal(String mixedName){
        QueryWrapper<MixedAnalysis> queryWrapper = new QueryWrapper<>();
        return mixedAnalysisService.getOne(queryWrapper.eq("mixed_name", mixedName));
    }

    @ApiOperation(value = "初步存储配煤方案")
    @PostMapping("/add-scheme")
    public int addSimpleScheme540(@RequestBody CoalScheme coalScheme){

        MixedAnalysis                    mixedAnalysis                    =
mixedAnalysisService.getById(coalScheme.getMixedId());
        //煤价
        double                    coal1Price                    =
coalInploService.getById(mixedAnalysis.getCoal1Id()).getCoalPrice().doubleValue();
        double                    coal2Price                    =
coalInploService.getById(mixedAnalysis.getCoal2Id()).getCoalPrice().doubleValue();
        double ratio = mixedAnalysis.getCoal1Ratio().doubleValue() * 0.01;
        double mixedPrice = coal1Price * ratio + coal2Price * (1-ratio);
        coalScheme.setMixedPrice(new                    BigDecimal(mixedPrice).setScale(2,
BigDecimal.ROUND_HALF_UP));
        //锅炉效率、环境温度
        double mixedQnet = mixedAnalysis.getMixedQnetAr().doubleValue() * 4.187;
        double[] result = new double[10];
        double tEnv = 20.0;
        Fanping50Dll.fanping50dll.fanping50(mixedAnalysis.getMixedCAr().doubleValue(),
                mixedAnalysis.getMixedHAr().doubleValue(),
mixedAnalysis.getMixedOAr().doubleValue(),
                mixedAnalysis.getMixedNAr().doubleValue(),
```

```
mixedAnalysis.getMixedSAr().doubleValue(),
                    mixedAnalysis.getMixedMAr().doubleValue(),
mixedAnalysis.getMixedAAr().doubleValue(),
                    mixedAnalysis.getMixedVDaf().doubleValue(), mixedQnet, tEnv, result);
            double boilerEff = result[0] - 1;
            coalScheme.setBoilerEff(new                    BigDecimal(boilerEff).setScale(2,
BigDecimal.ROUND_HALF_UP));
            coalScheme.setTEnv(new                         BigDecimal(tEnv).setScale(2,
BigDecimal.ROUND_HALF_UP));

            //厂用电率、机组负荷（540t/h）、日发电量(固定)
            double ratioOwn = 0.065;
            //机组负荷
            double boilerLoad = 178.00;
            double powerGenerate = boilerLoad * 24 / 10;
            double steamFlow = 540.00;
            coalScheme.setRatioOwn(new              BigDecimal(ratioOwn              *
100).setScale(2,BigDecimal.ROUND_HALF_UP));
            coalScheme.setBoilderLoad(new
BigDecimal(boilerLoad).setScale(2,BigDecimal.ROUND_HALF_UP));
            coalScheme.setSteamFlow(new
BigDecimal(steamFlow).setScale(2,BigDecimal.ROUND_HALF_UP));
            //小数位超出 setScale 范围则必须设置圆整
            coalScheme.setPowerGenerate(new
BigDecimal(powerGenerate).setScale(2,BigDecimal.ROUND_HALF_UP));
            //日供电量
            double powerSupply = powerGenerate * (1 - ratioOwn);
            coalScheme.setPowerSupply(new
BigDecimal(powerSupply).setScale(2,BigDecimal.ROUND_HALF_UP));
            //混煤煤耗、标煤煤耗
            double mixedConsumpKwh = (24 * (steamFlow * (3416 - 1016) + 431 * (3533-3051))
/
                    (mixedQnet * boilerEff * 0.01)) * 100 / powerSupply;
            double standConsumpKwh = (mixedConsumpKwh * mixedQnet) / 29273;
            coalScheme.setMixedConsumpKwh(new
BigDecimal(mixedConsumpKwh).setScale(2,BigDecimal.ROUND_HALF_UP));
            coalScheme.setStandConsumpKwh(new
BigDecimal(standConsumpKwh).setScale(2,BigDecimal.ROUND_HALF_UP));

            //估计 SOx、NOx(SOx 有较大误差)
            double SOx = 1821.4 * mixedAnalysis.getMixedSAr().doubleValue() + 376.48;
            coalScheme.setSox(new
BigDecimal(SOx).setScale(1,BigDecimal.ROUND_HALF_UP));
```

```
            double NOx = 212 + 155 * mixedAnalysis.getMixedNAr().doubleValue();
            coalScheme.setNox(new
BigDecimal(NOx).setScale(1,BigDecimal.ROUND_HALF_UP));
            //理论空气量 v0, 实际烟气量 vg
            double v0 = 0.0889 * (mixedAnalysis.getMixedCAr().doubleValue() +
                        0.375 * mixedAnalysis.getMixedSAr().doubleValue()) +
                        0.265 * mixedAnalysis.getMixedHAr().doubleValue() - 0.0333 *
mixedAnalysis.getMixedOAr().doubleValue();
            double vg = (1.886 * (mixedAnalysis.getMixedCAr().doubleValue() +
                        0.375 * mixedAnalysis.getMixedSAr().doubleValue()) + 0.8 *
mixedAnalysis.getMixedNAr().doubleValue() +
                        11.1 * mixedAnalysis.getMixedHAr().doubleValue() + 1.24 *
mixedAnalysis.getMixedMAr().doubleValue()) / 100
                        + (0.79 + 0.0161 + 0.4 * 1.0161) * v0;
            //估计石灰耗量、液氨耗量
            double limeConsump = ((SOx - 20) * 1.02 * mixedConsumpKwh * powerSupply *
vg)/(64 * Math.pow(10, 6));
            double nh3Consump = ((NOx - 35) * 72 * 17 * mixedConsumpKwh * powerSupply *
vg)/(46 * Math.pow(10, 10));
            coalScheme.setLimeConsump(new
BigDecimal(limeConsump).setScale(1,BigDecimal.ROUND_HALF_UP));
            coalScheme.setNh3Consump(new
BigDecimal(nh3Consump).setScale(2,BigDecimal.ROUND_HALF_UP));

            if(coalSchemeService.save(coalScheme)){
                return coalScheme.getSchemeId();}
            return 0;
        }
}
```

致谢

衷心感谢导师王春波教授对我的谆谆教诲和无私帮助。师从王老师多年，我深深折服于老师高屋建瓴的学术眼光、严谨的治学态度和果断的学术魄力。老师的言传身教让我受益匪浅、感激万分。

同时也感谢陈亮老师、赵启智师兄、郑菲师姐在此期间对我的给予的帮助和支持，让我收获了不少深刻的启发和教训，从而在正确的道路上保持希望。

其次，尤其感谢我的父母，他们在我处于瓶颈时给予我莫大的关怀和鼓励，让我感受到后背坚实的安全感，从而能全力朝着自己的目标努力。

此外，特别感谢本项目所应用到的所有开源技术和 B 站、GitHub 等平台上公开的教学资源，正是这群无私的开源者为我们这些学习者开辟了新的学习道路，让我们有机会一窥新兴技术的玄机，而他们的开源思想也将永远激励无数后浪，为科技产业的发展注入源源不断的活力！

最后，向为我提供学习平台的社会、学校以及所有关心、支持和帮助过我的师长、同学、朋友和家人致以最崇高的感谢和祝福！