****



**研 究 生 毕 业 论 文**

**（申请工程硕士学位）**

|  |  |
| --- | --- |
| **论文题目** | 德荣工业现场总线布线系统的  设计与实现 |
| **作者姓名** | 李行行 |
| **学科、专业名称** | 工程硕士(软件工程方向) |
| **研究方向** | 软件工程 |
| **指导教师** | 葛季栋　副教授 |

**2021 年 4 月 日**

**学 号： MF1932089**

**论文答辩日期： 2021 年 月 日**

**指 导 教 师： （签字）**

**德荣工业现场总线布线系统的**

**设计与实现**

|  |  |
| --- | --- |
| **作 者:** | **李行行** |
| 指导教师: | **葛季栋　副教授** |

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

|  |
| --- |
| **南京大学软件学院** |
| **2021年4月** |

# 第一章 引言

## 1.1 项目背景

# 技术综述

# 德荣工业现场总线布线系统的分析与设计

## 3.1 项目总体规划

本系统作为德荣工业推出的工业现场总线布线助手平台，目的是为了让客户（电气工程师或公司采购人员）能够根据自己的需求，添加各类产品绘制出模拟组网图，并生成对应的产品清单，该系统组合了德荣工业的多种IO总线模块产品，可以轻而易举的帮助用户搭建工业现场级的IO组网解决方案，增强用户与德荣工业公司的双向交流，为用户提供专业的，系统的组网，采购及服务体验．

工业领域是以严谨著称的，经过几百年的发展，已经建立起来了完善的标准体系．本系统作为一款工业软件，也要遵守工业领域严谨的标准体系．由于德荣工业公司研发生产了各种类别的产品，如智能集线器，IO预铸连接器，通讯电缆，传感器，执行器等，同类产品也根据所采用的协议，接口尺寸，数字信号类型，额定电压电流，端口规格，材质等划分出了不同的型号，不同的工业产品之间的通讯是有严格的规定的，这也就为客户采购选择产品带来了很多困难，需要耗费很多时间进行选择，最终可能还会出现所选产品之间无法通讯甚至无法连接的情况．

为了解决上述问题，德荣工业现场布线系统研发了如下功能，根据通讯协议划分出不同的模块，根据信号类型划分出不同的系列，客户在绘制组网图时，选择一个模块新建一个画板，该画板中只能添加采用该协议的产品，用户可以在画板中自由拖到产品组件进行组合，同时也可以使用系统提供的模板进行快速组装，当选择一个产品组件后，系统会推荐与其相通讯的其他组件供客户挑选，当两端的产品组件都选择完毕后，中间的连接器会自动确定，并显示型号，用户只需要再选择或输入连接器长度即可．这就避免了用户选择的产品无法通讯甚至连接的问题．另外，该系统提供了组网图检测的功能，可以检测出各个模块中是否可以正常运行，例如电压电流的检测，如果检测不通过会给出相应的提示，当所有检测都通过后，系统会为组网图生成产品清单，客户可以提交到询价平台进行询价下订单．

作为系统的管理员用户，该系统为其开发了后台管理平台，主要包括产品组件管理，报表管理，用户管理等模块，其中产品组件管理中包含组件的添加，修改及删除等功能，方便对新研发出的产品快速添加到系统中可供客户组网．报表管理中包含产品报表的分析与统计，主要是从客户组网后生成的产品清单中分析出客户对各类产品的喜好或需求程度，以及客户的采购率，从而调整公司对各类产品的生产比重．最后用户管理模块，包含流量统计，获客分析，反馈管理等功能，能够监控布线系统的使用状态以及获取客户对该系统的改进意见。

## 3.2 系统功能性需求分析

### 3.2.1 系统用例图



图3.1 系统用例图

本系统主要面向两类用户，如图3.1所示，第一类用户是德荣工业的客户，主要是工业互联网领域的其他公司的负责采购的人员或电气工程师，第二类用户是系统管理员．为客户提供的功能模块主要有组网图设计，图文管理和收藏管理．其中组网图设计中包含三个模块，Modbus模块，CC-Link模块和空白文件模块．图文管理包括文件的导入，导出和删除．系统管理员的主要职责是产品组件管理，统计分析和反馈管理，其中产品组件管理包括组件组件类别管理和组件管理，统计分析包括报表分析和流量分析．

下面使用用例描述表对用例图中提及的主要模块进行详细描述．

### 3.2.2 组网图设计模块功能需求

现场总线布线系统的组网图设计模块是该系统的主要功能，用户可以使用该功能设计并绘制出工业现场总线的拓扑图，该模块提供了新建空白文件，新建Modbus协议文件，新建CC-Link协议文件等子模块．在创建的空白文件中，用户可以随意通过点击或拖动在画布中添加产品组件，绘制连线，形成一张工业现场组网图，当然在添加产品时会受到协议或其他属性的限制，避免组网图中的组件无法连接或通讯．在创建的Modbus协议文件下，将只会显示采用Modbus协议的产品，同理，在CC-Link协议文件下，显示的是采用CC-Link协议的产品组件．

在每一个文件下，为其提供了特定的若干模板，提供模板的目的是为了让用户更加方便快速的绘制出组网图，模板中已固定添加了一些产品组件，并预留了一些需要用户自行添加产品组件的位置；每个文件下还包含增加模板模块，删除模板模块，模板模块命名，校验电流，生成并导出产品清单，文件属性设置，文件编辑（包括撤销，恢复，剪切，复制，粘贴等）等多种操作，并且能在画布文件中添加或删除产品组件，查看产品组件参数，更改产品组件外观样式（如颜色，大小，背景，文字描述等），本节以组网图设计模块为例进行具体的用例说明，用例描述如表3.1所示．

表3.1 组网图设计模块用例描述表

|  |  |
| --- | --- |
| ID | UC1 |
| 名称 | 设计组网图 |
| 参与者 | 公司采购人员或电气工程师 |
| 描述 | 用户在组网图设计模块可以完成对工业现场总线的布线设计，校验以及生成产品清单． |
| 优先级 | 高 |
| 触发条件 | 用户在导航栏选择新建文件并点击 |
| 前置条件 | 用户完成登录并通过了权限验证 |
| 后置条件 | 无 |
| 正常流程 | 1.用户选择一个文件类别  1.1 选择创建空白文件  1.2 选择创建Modbus协议文件  1.3 选择创建CC-Link协议文件  2.添加模板  2.1 添加M8集线器单通道模板  2.2 添加M8集线器双通道模板  2.3 添加M12集线器单通道模板  2.4 添加M12集线器双通道模板  3.在画布模板中添加产品组件，通过拖动或者双击弹窗的形式添加  3.1 在模板中添加产品组件时提供筛选功能，根据产品的极性，类型，线制，接头，额定电流等条件进行筛选  4.电流校验，针对某个绘制好的模块或者整个文件进行电流校验  5.查看并导出产品清单，跳转采购系统进行询价 |
| 异常流程 | 1. 如果用户没有登录，用户无法创建文件，提示用户登录 2. 若用户已创建文件超过10个，禁止用户再次创建新文件，提示用户删除之前的文件再进行创建 3. 添加的产品组件和已有组件不匹配，无法通讯，提示用户无法进行添加 4. 电流校验不通过，提示用户修改 5. 产品清单生成失败或无法导出 |

### 3.2.3 图文管理模块功能需求

图文管理模块用来帮助用户进行管理自己创建的文件，包括导入文件，导出文件，删除文件，查看文件，下载为图片，上传至服务器，设为共享，取消共享等多种功能，其中文件导出功能主要是为了节省服务器存储空间，同时用户又不愿将绘制的组网图文件舍弃，可以通过文件导出功能，将文件保存到用户本地，当用户再次使用到该文件时，再通过文件导入功能进行导入，导入之后可以查看并编辑文件；导出的文件对用户来说是不可读的，所以提供了下载为图片的功能，方便用户在本地进行查看．对于用户自己创建的文件用户可以选择设为共享，这样该文件将会在共享图文区显示，作为一种在特定场景下的布线解决方案，供其他用户参考和使用．本节以图文管理模块为例进行具体的用例说明，用例描述如表3.2所示．

表3.2 图文管理用例描述表

|  |  |
| --- | --- |
| ID | UC2 |
| 名称 | 帮助用户进行文件管理 |
| 参与者 | 公司采购人员或电气工程师 |
| 描述 | 用户在该模块下可以对创建的文件进行管理，包括导入，导出，删除，下载为图片，设为共享等多种功能 |
| 优先级 | 中 |
| 触发条件 | 用户点击导航栏的图文管理 |
| 前置条件 | 用户完成登录并通过了权限验证 |
| 后置条件 | 无 |
| 正常流程 | 1.用户进入图文管理页面  2.用户选择一个文件对其操作  2.1 单击文件打开并查看，可对其进行重新编辑  2.2 导出文件  2.2.1 点击导出按钮  2.2.2 选择导出的路径  2.2.3 点击确认导出成功  2.3 点击设为共享按钮将文件分享  2.4 下载为图片  2.4.1 点击保存为图片按钮  2.4.2 选择保存路径  2.4.3 点击确认图片下载成功  2.5 点击取消共享按钮将文件设为私有  2.６ 点击删除按钮将文件删除 |
| 异常流程 | 1.用户没有登录，无法进入图文管理页面，并提示用户登录  2.导出文件失败，返回提示信息  3.转换为图片并下载失败，提示用户相关错误信息 |

### 3.2.4 收藏管理模块功能需求

收藏管理模块是为用户提供共享文件收藏管理的，通常，在热门图文页面存在大量用户共享出来的文件，也包括德荣工业绘制出组网图并分享出的文件，为特定场景下的行业解决方案，供其他用户参考或使用．用户可以将自己喜欢或者对自己有用的组网图设计文件收藏起来，提供排序和关键词搜索等功能，方便之后参考和使用．表3.3是收藏管理模块的用例描述表．

表3.3收藏管理模块用例描述表

|  |  |
| --- | --- |
| ID | UC3 |
| 名称 | 为用户提供对收藏的共享文件的管理功能 |
| 参与者 | 公司采购人员或电气工程师 |
| 描述 | 用户在热门图文页面选择需要收藏的文件，在收藏管理页面对其管理 |
| 优先级 | 中 |
| 触发条件 | 用户点击导航栏的我的收藏 |
| 前置条件 | 用户完成登录并通过了权限验证 |
| 后置条件 | 无 |
| 正常流程 | 1.用户在共享图文页面找到自己需要收藏的文件，点击收藏按钮  2.用户在收藏管理页面根据关键词或时间进行搜索  3.搜索到结果后，单击可查看具体内容  4.点击取消按钮可取消收藏 |
| 异常流程 | 1.用户没有进行登录认证，无法进行收藏，点击收藏按钮提示其登录  2.用户输入的搜索关键字不合法  3.收藏或取消收藏失败，提示其具体错误信息 |

### 3.2.5产品组件管理模块功能需求

产品组件管理模块是德荣工业布线系统后台管理中最重要的一个模块，该模块主要作用是管理员用来向系统中添加新产品或修改产品信息，供用户在绘制拓扑图时使用该组件．

产品组件管理模块包括组件分类和组件管理，由于产品数量非常多，需要根据其功能对其分类，如集线器产品，终端产品，连接器产品等，分类后，方便用户进行选择．对组件类别的管理通常是添加类别，修改类别，删除类别以及搜索等常用操作．

组件管理为管理员提供了查看，添加，修改，搜索和删除组件的功能，添加产品组件时需要添加产品名称，产品料号，产品图片，选择分类，产品极性，接头尺寸，额定电流，产品描述，产品备注等各种属性．本节以组件管理模块为例进行具体的用例说明，用例描述如表3.4所示．

表3.4 产品组件管理模块用例描述表

|  |  |
| --- | --- |
| ID | UC4 |
| 名称 | 产品组件管理 |
| 参与者 | 系统管理员 |
| 描述 | 系统管理员通过该模块可以对产品组件进行管理，包括组件分类，组件添加，组件编辑，组件删除等功能 |
| 优先级 | 高 |
| 触发条件 | 用户点导航栏产品组件模块 |
| 前置条件 | 用户登录布线系统的后台管理系统并通过权限认证 |
| 后置条件 | 无 |
| 正常流程 | 1.用户选择产品组件类别  1.1 用户点击添加按钮添加类别  1.2 用户输入关键词点击搜索按钮搜索类别  1.3 用户点击编辑按钮进行更改  1.4 用户点击删除按钮将该类别删除，如果该类别下有产品组件，将提示用户是否同时删除  2.用户点击组件管理  2.1 用户输入关键字点击搜索按钮进行搜索  2.2 用户点击编辑按钮进行修改组件信息  2.3 用户点击删除按钮将某个产品组件的信息删除  2.4 用户点击添加按钮，选择已有产品分类，添加产品组件信息，如果产品分类不存在，需要去1.1中添加产品分类 |
| 异常流程 | 1.用户没有权限登录，则提示无权访问  2.输入非法关键词搜索，给出错误提示  3.后端接口错误，给出相应错误代码和错误提示 |

### 3.2.6统计分析模块功能需求

统计分析模块是后台管理中非常重要的一个模块，包括报表统计与分析，流量统计与分析两个子模块，是企业用来分析用户数据，了解用户行为的非常重要的一个渠道。

1. 报表统计与分析

　　　报表统计与分析模块依据的数据主要来自用户绘制组网图产生的产品清单，可根据单独用户或全部用户数据进行分析与统计。针对单独用户，可以统计出该用户所设计的组网图中各种文件类型的数量及比例，各种模板类型的数量及比例，各种产品组件类型的数量及比例，针对以上三组数据，同样适用于针对全部用户进行统计。工业软件不像互联网软件那样有着大量的用户基数，而每一个该系统的使用者背后通常代表着一家公司进行采购，所以对单个用户和全部用户的数据统计都同样重要，更有利于了解用户对各类产品的需求，同时有利于发掘和发展潜在客户。

表3.5 报表统计与分析模块用例描述表

|  |  |
| --- | --- |
| ID | UC5 |
| 名称 | 产品报表统计与分析 |
| 参与者 | 系统管理员 |
| 描述 | 系统管理员通过该模块对用户的组网图文件中使用的产品组件进行统计和分析 |
| 优先级 | 中 |
| 触发条件 | 用户点导航栏统计分析模块，选择报表统计与分析 |
| 前置条件 | 用户登录布线系统的后台管理系统并通过权限认证 |
| 后置条件 | 无 |
| 正常流程 | 1. 系统管理员选择用户，单个用户（支持搜索）或全部用户 2. 管理员选择时间区间 3. 系统管理员选择统计类别，分为文件类型统计、模板类型统计和组件类型统计 4. 管理员根据显示出的图表查看各类型的数量及比例，点击图表可以查看数据详情 |
| 异常流程 | 1.用户没有权限登录，则提示无权访问  2.输入非法关键词搜索，给出错误提示  3.后端接口错误，给出相应错误代码和错误提示 |

1. 流量统计与分析

流量统计与分析虽然不如报表统计与分析模块重要，但是也是企业用来了解用户行为习惯的一个重要渠道，主要用来统计每天的用户访问量以及访问时长，针对每天访问量可以做成趋势图，了解每天、每周、每月的用户增长率，以及在不同形式的宣传活动中获得的新的用户访问量，如参加工业博览会设置展台，或在其他平台投放广告，查看当天的用户访问增量，以此判断该宣传是否达到预期效果。针对访问时长，可以计算出平均访问时长，最长访问时长，最短访问时长，以及各时长所占比例，以此判断用户对该系统的兴趣程度，计算用户留存率。

表3.6 流量统计与分析模块用例描述表

|  |  |
| --- | --- |
| ID | UC6 |
| 名称 | 用户流量统计与分析 |
| 参与者 | 系统管理员 |
| 描述 | 系统管理员通过该模块对工业布线系统的访问量、用户使用时长及用户增长量统计和分析 |
| 优先级 | 中 |
| 触发条件 | 用户点导航栏统计分析模块，选择流量统计与分析 |
| 前置条件 | 用户登录布线系统的后台管理系统并通过权限认证 |
| 后置条件 | 无 |
| 正常流程 | 1. 管理员点击访问量分析    1. 选择时间区间查看用户访问量曲线    2. 选择具体时间查看每天访问量    3. 点击导出按钮导出访问量数据表格 2. 管理员点击使用时长分析    1. 选择日期范围，统计平均使用时长，最长使用时长，最短使用时长    2. 选择时长区间，统计各区间用户数及所占比例    3. 点击导出按钮导出使用时长数据表格 3. 管理员点击常用用户分析    1. 选择月份，查看常用用户数量    2. 查看常用用户使用次数排名    3. 查看常用用户使用时长排名    4. 点击导出按钮导出常用用户数据表格 |
| 异常流程 | 1.用户没有权限登录，则提示无权访问  2.输入非法关键词搜索，给出错误提示  3.后端接口错误，给出相应错误代码和错误提示 |

### 3.2.7留言管理模块功能需求

用户留言管理模块是企业与用户沟通的一个重要通道，由于目前德荣工业还处于成立初期阶段，还不具备完善的客服系统，也无法雇佣太多客服在线解答用户的问题，用户留言管理模块就显得尤为重要，通过用户留言的方式了解和解答用户的问题，听取用户对于产品和该系统提出的宝贵建议，便于系统的改进与升级，进而为用户提供更优质的服务，优化用户体验，最终也有利于公司的利益与发展。

表3.7 留言管理模块用例描述表

|  |  |
| --- | --- |
| ID | UC5 |
| 名称 | 用户留言管理模块 |
| 参与者 | 系统管理员，用户 |
| 描述 | 用户通过工业布线系统中的留言反馈功能向公司系统管理员提出建议或疑问，由系统管理员解答或将反馈意见提交给开发者，由开发进行系统优化。 |
| 优先级 | 高 |
| 触发条件 | 用户点击导航栏中留言管理模块 |
| 前置条件 | 用户需登录工业布线系统并通过权限认证，系统管理员需登录后台管理系统并通过权限认证 |
| 后置条件 | 无 |
| 正常流程 | 1. 用户点击新增留言按钮    1. 用户选择留言类别，包括产品问题，系统问题，其他问题等    2. 用户填写留言标题    3. 用户选择将留言设为公开或私密    4. 用户填写具体的留言内容，并提交留言 2. 用户查看留言    1. 用户根据关键词筛选留言（包括自己的留言和全部公开留言）    2. 点击留言查看具体内容及回复 3. 系统管理员根据留言类型，留言时间等条件筛选留言    1. 管理员点击留言查看具体内容    2. 管理员回复留言问题 |
| 异常流程 | 1.用户没有权限登录，则提示无权访问  2.输入非法关键词搜索，给出错误提示  3.后端接口错误，给出相应错误代码和错误提示 |

## 系统非功能性需求分析

由于工业软件的特殊性，对规范性、性能、易操作性、可扩展性，可维护性等有着更高的要求，下面将从这几个维度进行分析。

1. 规范性：工业系统已严谨著称，在本系统中，产品组件涉及到多种通讯协议，以及电路参数，对产品组件的连接以及指标计算要严格按照普遍认可的标准进行，同时系统的设计也要符合工业软件的设计标准。
2. 性能：对于首页加载时间95%不能超过2s，对于打开组网拓扑图时90%的情况下不能超过3s，对于组网图设计过程中添加删除产品组件时，99%的情况下不能超过1s，对于其他模块不考虑网络问题响应时间不应超过5s。
3. 易操作性：系统主要面向用户为电气工程师，应符合他们绘制网络拓扑图的习惯，提供多种方式绘制，如单击，双击弹窗，拖拽等，增加一些自动生成的功能，如根据终端设备自动生成电缆型号，减少用户的操作。页面尽量简洁明了，导航清晰，各产品组件参数齐全，减少用户学习如何使用该系统的成本。
4. 可扩展性：德荣工业处于快速发展阶段，会不断有新的产品被研发出来，也会有新的业务需求，所以系统应支持良好的可扩展性。同时随着用户增长，业务的复杂，也应支持系统的拆分和升级，如拆分为分布式架构或微服务架构。
5. 可维护性：系统设计及代码编写要符合规范，当系统出现错误时，便于错误定位与排查，方便运维人员进行系统的维护。

## 系统总体设计

由于德荣工业目前仍处于成立初期，公司的门户网站、购物网站以及该现场总线布线系统是传统行业通过互联网的一次创新与探索，目标是以快速实现，快速上线为主，同时对系统的稳定性以及易维护性提出更高要求，基于此该系统以采用单体架构为主，单体架构系统完全能够满足近一两年内的用户量及数据量的增长，同时单体架构也是统治历史最久，应用范围最广的一种架构风格，这种单体架构不仅易于开发、测试、部署以及维护，而且各模块和方法的调用是基于进程内的调用，避免了进程间通信的复杂性以及风险，单体架构反而拥有更高的运行效率。

单体架构中的单体是相对于时下较为流行的微服务而言的，从单体架构的纵向角度来看，依然是可分层的，通常分为Model层、Service层、Controller层及View层。该系统使用前后端分离架构，前端相当于View层用来展示数据及与用户交互，后端包含Model层、Service层及Controller层进行与数据库交互与业务处理。

考虑到目前公司已有的门户网站和采购系统，为了实现用户在各系统之间跳转，增加用户体验不需要重新登录，采用统一权限认证服务，即CAS单点登录系统。如图3.2所示，为系统的整体架构图。如图所示，工业布线系统和后台管理系统是分离的，属于两个服务，这样做的目的是当后台管理系统升级时，无需将工业布线系统停止，管理系统的异常宕机也不会波及到布线系统，反之亦然。单点登录系统也单独抽离了出来，虽然现在各个系统处于同一个域名下，可以利用Cookie和Session共享来简单实现单点登录，但随着公司的发展，有可能存在多个域名，也会有新的系统接入，那时就会存在Cookie跨域问题，因此，一开始就将单点登录服务抽离出来，作为中央认证服务(CAS)，方便系统的扩展。也为已有的采购系统接入CAS减少了工作量。

图3.2 系统整体架构图

如下图3.3，是该系统的逻辑视图，用来描述该系统的主要功能，为用户提供了哪些服务，以及各模块的约束和边界。

组网图设计模块：可以创建空白文件开始设计，同时支持创建Modbus协议文件和CC-Link协议文件，每一种协议文件下提供不同的模板和产品组件。每一种文件中都包含多种操作，如模板添加删除，电流校验，组件添加删除及查看详细参数，生成并导出产品清单，以及常用快捷键等。



图3.3 系统逻辑视图

图文管理：主要是对自己创建的组网图进行管理，包括导入导出、查看、删除、保存、共享、下载为图片等多种功能，图文数据是以json的格式保存到非关系型数据库MongoDB中，由于服务器容量限制，不允许用户无限的保存文件到服务器，每个用户最多保存十个文件，但是该模块提供了文件导入导出功能，用户可将创建的文件已json的格式导出到本地，当下次需要时，再导入系统，可以重新解析为组网图，并且可以进行编辑。当然用户也可以将文件以图片的形式保存到本地，方便查看。

收藏管理：在共享区会存在大量的组网图，作为优秀的行业解决方案，其他用户可以参考，点赞和收藏。在收藏管理中可以对收藏的组网图进行管理。

留言管理：用户通过该模块对产品或系统提出建议，也可以发布到公共留言区同大家一起交流，可以回复其他人的留言。

组件类别管理：为了方便对产品组件的管理，从功能、极性等维度将组件分为多个类别，对类别提供新增，删除，修改参数等操作。

产品组件管理：通过在后台管理系统中添加，修改或删除组件，布线系统中就会产生相应的变化，用户能够使用这些产品组件进行组网图设计。

报表统计分析：提供了文件类别统计，模板类别统计，组件类别统计等功能，可针对单个用户或全体用户数据进行统计分析，对于公司用来了解用户需求和发展潜在客户有着重要作用。

流量统计分析：管理员通过该模块查看每天用户访问量，访问时长，常用用户等信息，是公司宣传推广效果的最直接反馈。

如图3.4系统物理视图所示，系统采用Nginx作为静态服务器和反向代理，在前后端分离项目中，前端采用Angular框架，构建后将生成的静态文件（html,js,css等）放到静态服务器上，如图，三个子系统的前端文件都放置在静态服务器上，通过Nginx映射到不同的端口上，同时Nginx还具备路由功能，通选择各自的后台服务，也可以通过Nginx对业务服务集群进行负载均衡。对于各个后端服务，采用同一个缓存服务Redis，用来缓存用户信息及热点数据，使用Mysql存储关系型数据，如用户表，使用MongoDB保存非关系型、拥有层次结构的数据，如产品组件表，组网图数据表等，以上缓存和数据库初步设计均为单体结构。



图3.4 系统物理视图

图3.5 系统开发视图

如图3.5系统开发视图所示，如图所示，对系统层次进行划分，每个层次的系统只能和同层或更低层的子系统进行通信，从上往下，层次越低，通用性越强，这样当需求变化时，只需改动一小部分代码即可。最上层为数据可视化服务，主要是使用前端Angular框架，依赖了Ant-design，Echarts等开源库进行图表展示，引入le5le进行客户端存储，引入topology开源的源代码，并在此基础上进行扩展和修改，实现拓扑图的绘制功能。服务端模块分为独立的组件管理模块，组网图设计模块，报表分析和流量分析模块，这些模块可以独立开发，互不干扰，它们共用基础服务，包括权限认证服务、日志服务、错误码映射服务等，再底层是缓存服务，用来连接Redis，包括一些增删改查的操作，使用缓存服务可以使热点数据快速返回，减少用户等待时间。最底层是数据库服务，连接MongoDB和MySQL并进行一些操作，MongoDB对于存储层次化的数据比较擅长，可以提高数据访问速度，MySQL存储关系型数据，可以进行一些表的连接查询操作。

## 系统模块设计

### 3.5.1 单点登录设计

### 3.5.2 组件管理模块设计

组件管理模块是后台管理中一个重要的模块，主要用来配置布线系统中可用的产品组件，组件管理分为组件类别管理和产品组件管理两部分。

1. 组件类别管理

组件类别管理是为了方便产品组件管理，以及在布线系统中可以方便查找使用，根据产品的用途、材质、电流、极性等特性进行分类分组，不同类别下的产品组件适用于不同的场景。

图3.6 组件类别管理类图

如图3.6所示，为组件类别管理中的核心类图，GroupService对外提供了六个接口，getAll方法用来获取所有类别信息，save方法用来新建类别时进行插入到数据库，findGroupById方法通过group\_id进行搜索，搜索结果是唯一的。findGroupByName是通过类别名称关键字进行模糊查询，搜索结果不唯一，返回List<Group>类型。deleteGroupById方法是通过group\_id进行删除分类信息，updateGroupById方法是根据group\_id字段更新分类信息。GroupServiceImpl实现GroupService接口中的方法，这些方法均通过GroupDao进行对数据库操作。组件类别参数如表3.1所示。

表3.1 组件类别参数

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 |
| Group\_id | String | 分组的主键，不可重复 |
| Group\_name | String | 分组名称 |
| Group\_des | String | 分组的详细描述，用来描述该分组的适用场景 |
| Group\_imgUrl | String | 分组的图标地址 |

1. 产品组件管理

管理员通过产品组件管理实现对产品组件的添加，修改，删除等操作，当新的产品生产出来后，通过该模块将产品信息注册到系统中，用户在进行组网图设计时就可以选择该产品进行绘图，生成产品清单，然后询价购买。



图3.7 产品组件管理类图

如图3.7所示，接口ProductService是产品组件管理接口，提供了七种方法供控制层调用，getAllByGroupId方法是通过分组类别获取产品组件，返回结果类型为List<Product>，save方法用来保存产品组件信息，findProductByName方法通过产品名称模糊查询，返回结果不唯一，findPrductById通过产品组件id查找，返回结果唯一。deleteProductById通过产品id删除产品组件，deleteProductsByGroupId方法是通过分类Id删除该类别下的所有产品组件，updateProductById方法通过产品id更新组件信息。这些方法均调用ProductDao类中对应的方法进行数据库操作。表3.2为产品组件参数表。

表3.2 产品组件参数表

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 |
| Product\_id | String | 产品组件的唯一标识 |
| Group\_id | String | 产品所在分组的唯一标识。 |
| Product\_name | String | 产品组件的名称。 |
| Product\_imgUrl | String | 产品组件的图片地址，该图片指的是在产品栏中显示的图片。 |
| Product\_style | Object | 在产品栏中显示的产品图片风格，包含两个参数，宽width和高height，类型均为int，单位为px; |
| Product\_data | Object | 产品组件的详细数据，包含在画布中存在的外观参数以及产品属性。 |

### **3.5.3 组网图设计模块设计**

组网图设计模块是现场总线布线系统最核心的模块，该模块可以实现拓扑图的绘制功能，拓扑图绘制是在画布上进行的，而为了提升性能，对画布进行逻辑分层，可分为离屏层、选中层、浮动层三层。离屏层比较稳定，主要是包含绘制的所有组件以及电缆的数据，一旦添加上，变动较少。选中层是当鼠标点击选中时所表现出来的图层，会显示出组件的控制点，进行旋转与缩放。浮动层是当鼠标移动到组件时所表现出来的图层，主要用于显示锚点和连接电缆。

根据面向对象的思想进行设计，将画布中所有的组件抽象出两类，节点类Node和连线类Line。所有的产品组件都可抽象为节点，所有用于连接组件的电缆均可抽象为连线。除此之外，还应抽象出更基础的点类Point和画笔类Pen，Node和Line均继承与Pen，表3.3为画笔类Pen的参数说明。

表3.3 画笔类参数表

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 默认值 | 参数说明 |
| id | String | null | 唯一标识，不可为空，系统自动生成 |
| type | int | 0 | 代表是节点还是连线，0是节点，1是连线 |
| text | String | null | 节点或连线上的文字 |
| font | Object |  | 文字的风格，包括文字颜色color，大小size，对齐方式textAlign，背景颜色background |
| name | String | null | 名称 |
| rect | Object |  | 该组件在画布中所占区间大小和位置，有四个参数，x,y,width,height。 |
| locked | Boolean | false | 是否锁定，锁定之后不可移动和更改组件参数。 |

节点类Node继承自上面的画笔类，除了包含画笔类Pen中的所有参数之外，还拥有一些特有的参数，如下，表3.4为节点类参数说明。

表3.4 节点类参数表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | | 参数类型 | 默认值 | 参数说明 |
| image | | String | null | 所包含的图片地址 |
| imageWidth | | int | 25 | 图片宽度 |
| imageHeight | | int | 25 | 图片高度 |
| mubanId | | String | null | 节点所在的模板id |
| backgroundColor | | String |  | 节点背景颜色 |
| anchors | | Point[] |  | 节点上的锚点坐标 |
| children | | Pen[] |  | 该节点的子节点或子连线，处于该节点内部 |
| attribute | productName | String | null | 产品名称 |
| channelType | int | 1 | 产品通道类型，1代表单信号，2代表双信号 |
| signalType | int | 1 | 产品信号类型，1代表NPN，2代表PNP |
| portType | int | 1 | 端口类型，1代表M8系列，2代表M12系列 |
| property | jixing | String | null | 终端组件极性 |
| leixing | int | 1 | 终端组件类型，1代表输入型，2代表输出型 |
| xianzhi | int | 3 | 终端组件芯数，有3芯、4芯和5芯 |
| jiekou | int | 1 | 接口类型，1代表M8,2代表M12 |
| current | int | 0 | 终端组件额定电流 |
| name | String | null | 终端产品中文名称 |
| desc | String | null | 终端产品描述信息 |

连线类Line同样继承自画笔类Pen，除了画笔类中已有属性，还包含一些独有属性，如下表3.5所示。

表3.5 连线类属性表

|  |  |  |
| --- | --- | --- |
| **属性名称** | **属性类型** | **属性说明** |
| from | Point | 连线的起点 |
| to | Point | 连线的终点 |
| controlPoints | Points[] | 连线的控制点位，控制点位通常为转折点 |
| fromArrow | String | 连线起点箭头图标 |
| toArrow | String | 连线终点箭头图标 |
| length | int | 连线长度（并非实际长度，仅是电缆的属性） |
| color | String | 连线颜色 |

定义完这些基础类之后，要对画布进行设计，画布分为离屏层，选中层和浮动层，离屏层包含拓扑图中的所有数据，主要参数为画布dom元素canvas,所有节点数据nodes和所有连线数据lines。主要方法除了构造函数用来初始化画布大小以及画布背景颜色和水印外，还包含能够改变画布大小的方法resize以及能够重新渲染画布的方法render。选中层也抽象出类，成员变量除了canvas，nodes和lines外，还包含控制旋转的控制点rotatePoints和控制大小的控制点sizePoints以及旋转角度rotate。成员函数出构造函数以及resize和render之外，还有getPoints用来计算节点的顶点，resizeNodes用来改变节点大小，moveNodes用来移动节点，updateLines用来更新连线，add方法用来添加节点或连线到选中层等。最后是浮动层，该类的成员变量包括canvas，line，node，hoverLineCP(鼠标悬停在节点上的锚点,即连线起点)，dockAnchor(连线终点的锚点)。成员函数包括render，resize，constructor,lineFrom(用来设置连线的起点)，lineTo(用来设置连线的终点)，lineMove(用来移动连线)等。图3.8为详细的组网图设计类图。



图3.8 组网图设计核心类图

如图3.8所示，组网图数据抽象出TopologyData类，该类中核心参数有pens列表，为节点和连线的集合，除此之外是组网图文件背景颜色backgroundColor，最后是文件类型fileType，文件类型分为空白文件，Modbus协议文件，CC-Link协议文件等。Topology类对外提供了若干操作画布和组件的方法，是绘制组网图的核心类，该类组合了离屏层、选中层和浮动层，将这三个类的方法进一步封装对外提供，主要的方法有添加节点、删除节点、移动节点、添加连线、删除连线、移动连线、打开文件、保存文件等操作，合理的调用这些操作，就可以实现组网图的绘制功能。而将文件持久化的操作，需要将文件数据保存到数据库中，除了保存到topologyData表中的数据，还需要保存文件的概要信息到topology表中，具体属性如表3.6所示。

表3.6 topology属性表

|  |  |  |
| --- | --- | --- |
| **属性名称** | **属性类型** | **属性描述** |
| id | String | 唯一标识，和topologyData中的id相同且一一对应 |
| name | String | 文件名称 |
| desc | String | 文件描述 |
| image | String | 文件缩略图保存路径 |
| userId | int | 文件拥有者id |
| userName | String | 文件拥有者姓名 |
| createdAt | Date | 创建时间 |
| shared | boolean | 是否为共享文件 |

除了组网图的绘制功能之外，还应该生成图中使用的产品清单，根据模块进行划分，方便询价与购买。产品清单是在绘制组网图过程中动态生成的，当添加节点或连线时，就将对应的组件添加到产品清单中，删除节点或连线时，再将产品组件移除。如表3.7为产品清单属性参数。

表3.7 ReportList属性表

|  |  |  |
| --- | --- | --- |
| **属性名称** | **属性类型** | **属性描述** |
| id | String | 产品清单唯一标识，和组网图文件id相同，为一对一关系。 |
| userId | String | 产品清单所属用户。 |
| data | List<Module> | 产品清单实际数据。 |
| createTime | Date | 创建时间。 |

产品清单实际数据data是Module列表，Module为模块数据，每一个模块对应一个单独的对象，Module属性列表如表3.8所示。

表3.8 Module属性表

|  |  |  |  |
| --- | --- | --- | --- |
| **属性名称** | | **属性类型** | **属性描述** |
| id | | String | 模块的唯一标识。 |
| mubanName | | String | 该模块自定义的名称。 |
| mubanType | | String | 模块的类型。 |
| reportNodes | pen\_id | String | 节点或连线的唯一标识 |
| pen\_name | String | 节点或连线的名称 |
| pen\_type | String | 节点或连线类型，由编号表示 |
| remarks | String | 备注 |

### **3.5.4 报表统计模块设计**

根据需求分析可知，报表统计分析模块是统计分析的一个子模块，主要的功能是将用户创建组网图所产生的数据进行统计分析，包括文件类型统计，模板类型统计，产品组件类型统计，通过对用户的这些数据进行分析，进而了解用户需求，了解各模板和产品组件的使用率，从而改进产品和系统。

（1）文件类型统计部分，主要数据来源与topology和topologyData中的数据，如图3.9文件类型数据统计类图所示，Controller层提供了五个方法供前端调用，getFileTypeByUserId方法用来查询每个用户创建的文件类型及其数量，返回数据类型为Map<String,Integer>类型，key为文件类型，value为对应的数量，getNumByFileTypeAndUserId方法用来查询每个用户下指定类型的文件数量，getNumByUserId方法用来查询每个用户创建的所有文件数量，getFileTypeByUserIdAndCreateAt方法用来查询每个用户指定时间段创建的各类型的文件数量，getAllFileTypeByCreateAt用来统计全部用户指定时间段创建各类型的文件数量。以上方法都通过组合调用TopologyService和TopologyDataService实现，再由Service层调用Dao层从数据库中获取数据，类图中省略了Dao层的类。



图3.9 文件类型数据统计类图

（2）模板类型统计部分，所依赖的数据来源于reportList数据表，reportList中的数据data是模板module的集合，获取模板类型数据的类图如图3.10所示，ModulesController为前端提供了5种方法供其调用，getMubanTypeByUserId方法通过用户Id查询每个用户使用各类型模板的数量，返回结果类型为Map<String,Integer>,map的key为MubanType，value为对应的数量。getMubanTypeByUserIdAndCreateTime增加了创建时间作为筛选条件，查询各时间区间的数据。getNumByMubanTypeAndUserId方法用来获取各用户使用特用模板的数量，getMubanTypeByCreateTime方法通过时间区间获取该时间段所有用户使用各类模板的数量情况，getNumByMubanType方法用来获取某类模板被使用情况。通过以上数据，可视化出各类图表展示出来供管理员查看。



图3.10 模板类型数量统计类图

1. 产品组件类型统计部分，用来统计各组件被使用的情况，数据同样来源于ReportList，由于数据具有分层结构，且读取和更改都较为频繁，使用mongoDB数据库保存较好，使用聚合查询也可以满足查询的需求，组件类型统计类图如图3.11所示。ReportNodeController对外提供了五种方法供前端调用，getPenTypeByUserId方法用来获取各用户所使用的产品组件的类别及其数量，getPenTypeByUserIdAndCreateTime方法增加了时间区间作为筛选条件，用来查询某个时间段某个用户所使用的产品组件的类别及其数量。getNumByPenTypeAndUserId方法用来查询某用户某种产品使用的次数，getPenTypeByCreateTime用来查询某个时间段所有用户使用的各类别的产品数量，getNumByPenType用来查询某类型组件被使用的次数。以上所有方法均通过调用ReportListService中的相关方法实现，ReportListService依赖了ReportList,Module和ReportNode三个类，这三个类之间存在聚合关系。而前端程序通过以上所有数据，形成各类图表，有利用管理员了解各产品受欢迎程度，以及用来分析某用户需求。

图3.11 产品组件统计数据类图

### **3.5.5 流量统计模块设计**

根据需求分析可知，流量统计模块主要功能分为三个部分，用户访问量统计、用户访问时长统计和常用用户统计。下面对该模块进行详细设计。

用户访问量统计，是以天为基本单位，保存每天访问的用户数量，即DAU(日活跃用户数)，以userId作为区分，同一userId在同一天内重复登录访问只算一次。可以通过Redis来实现，当用户登录成功进入系统首页时，触发一次请求去增加访问量，具体做法是使用Redis中的Set数据结构，利用其自动去重功能，可以记录每天登录的userId，然后启动定时任务，每日24点统计Set集合的长度，就可以得出今日访问的用户数，再将其持久化到Mysql数据库中，同时将set集合置为空。Redis是基于内存的，增删改查等操作都非常迅速，避免了对数据库的频繁操作，对系统的性能影响较小。

用户访问时长即用户在站点停留的时间（Time On Site)，用来统计某日期区间内的平均使用时长，最长使用时长，最短使用时长，以及各时长区间内的用户数，如30s以内、30s-1min、1min-5min、...、1h以上等区间。对页面停留时间和站点停留时间的统计，无法做到非常精确，主要设计思路为监听进入页面和离开页面的时间，获取两个时间戳，由离开时间减去进入时间，就为在该页面停留时间，将各页面停留时间相加，就是用户在站点停留时间。由于会频繁的进行时间的累加，使用Redis保存效率更好，使用String数据结构，key为userId，value为访问时长，当每一次用户离开页面时，都会计算本次的访问时长，并发送给服务端，服务端获取到用户id和时长，如果Redis中存在该userId的key，则将时长进行累加，如果没有该userId的记录，则插入该条记录。启动定时任务，在每日24时将每个用户的访问时间保存到mysql中进行持久化，同时将Redis中的数据清空。Mysql中的用户访问时长表user\_visit属性如表3.9所示。

表3.9 用户访问时长属性表

|  |  |  |
| --- | --- | --- |
| **属性名称** | **属性类型** | **属性说明** |
| user\_id | int | 用户唯一标识 |
| visit\_date | String | 用户访问日期，dd-MM-yyyy格式 |
| visit\_duration | Long | 用户访问时长 |

常用用户统计，管理员选择日期区间后，对在该时间段内访问的用户进行访问次数排行，显示出访问次数前20的用户信息及其对应的访问次数。还可以对该时间段内用户访问时长进行排行，同样显示出排名前20的用户，显示出用户信息及其访问时长。该部分的数据主要来自于user\_visit表，统计访问次数排行，先根据日期筛选数据，然后根据userId进行分组计算记录条数，最后倒序排列取出前20条记录的userId，再关联用户表获取用户信息即可，访问时长排行类似。

针对以上三种统计，设计出如下类图，如图3.12所示。VisitRedisService类是通过Redis进行实时统计用户访问量和用户访问时长，addUserId()方法是当用户登录时，就用户userId添加到Redis中的Set中。每当用户离开页面时，前端计算出该次访问时长，通过addDuration方法更新Redis中Map<userId,time>，其中time类型为Long,代表时间戳差值。



图3.12 流量统计类图

VisitMysqlService中的方法是用来将Redis中的数据持久化到Mysql中，使用定时任务每天的23:59执行，saveDAU方法用来持久化当天的访问量，取出Redis中set中的数据个数，和日期一起保存到数dau据表中。saveTOS方法用来持久化站点停留时间，同样访问Redis获取map中的数据，将每一条记录封装成userVisit保存到user\_visit数据表中。VisitService方法对外提供了获取数据的接口，getDauSumByDate(date1,date2)方法用来获取日期date1至date2之间的总用户访问量，getDauByDate(date1,date2)方法用来获取date1至date2之间每天的用户访问量。getAvgTime()方法用来获取平均访问时长，getLongestTime()用来获取最长访问时长，getShortestTime()用来获取最短访问时长，getCountByTime(date1,date2,time1,time2)用来获取在日期date1至date2之间，访问时长在time1至time2之间的用户数。getUserByVisitTime(date1,date2)用来获取日期date1至date2之间用户访问次数排名，getUserByVisitDuration(date1,date2)方法用来获取日期date1至date2之间的用户访问时长排行。

## 本章小结

本章主要描述了系统各模块的需求分析以及详细设计。首先介绍了项目的总体规划，然后绘制出系统的用例图，介绍了系统主要面向的用户角色以及各角色相关的用例和模块，紧接着以用例描述表的形式分别介绍了组网图设计模块、图文管理模块、收藏管理模块、产品组件管理模块、统计分析模块以及留言管理模块的功能性需求，然后介绍了系统非功能性需求。接着介绍系统的总体设计，以各种视图进行辅助说明。最后，对各个模块进行详细设计，以类图和数据结构表的形式展现，着重设计了单点登录、组件管理模块、组网图设计模块、报表统计模块以及流量统计模块，对于图文管理、收藏管理以及留言管理三个模块，由于比较简单，文中省略了对它们的详细设计部分。

# 德荣工业现场总线布线系统的实现

## 4.1 单点登录的实现

## 4.2 组网图设计模块的实现

根据3.5.3节的组网图设计模块的详细设计，对组网图中的连线和节点等抽象出了类，同时对画布也抽象出了三层，为了减少设计组网图时前端与后端的交互，增加使用时的流畅度，将这些类都通过TypeScript语言定义在前端，只有保存和获取等操作向后端发送请求，其余的添加、删除、移动节点和连线，更改节点和连线参数等操作均在前端对组网图数据topologyData进行操作完成。

关于节点的实现，节点类定义部分代码如下：

|  |
| --- |
| export class Node extends Pen {  z: number;  zRotate = 0;  // 产品的硬件属性  attribute: {  productName: string;  channelType: number;  signalType: number;  portType: number;  feature: number;  configuration: number;  modelsType: string;  };  plugType?:number; //plug是单通道中的还是双通道中的  image: string;  imageWidth: number;  imageHeight: number;  anchors: Point[] = [];//锚点  rotatedAnchors: Point[] = [];  ......  } |

图 4.1 Node类代码图

为了将以上的定义在绘制成图像展示，在Node中需要定义draw函数来绘制，核心代码如下：

|  |
| --- |
| draw(ctx: CanvasRenderingContext2D) {  if (!drawNodeFns[this.name]) {  return;  }  // 绘制形状  drawNodeFns[this.name](ctx, this);  // 绘制文字  if (this.name !== 'text' && this.text) {  text(ctx, this);  }  // 绘制图片.  if (this.image) {  this.drawImg(ctx);  return;  }  } |

图4.2 Node类中draw方法代码

其中绘制形状需要根据组件名称来判断，调用drawNodeFns函数，将ctx和该节点对象传入即可，绘制图片和文字也类似。

针对德荣工业生产的各种产品，需要每一种产品注册为一种节点，方可在系统中使用，注册函数代码如下：

|  |
| --- |
| export function registerNode(  name: string,//节点名称，产品的唯一标识  drawFn: (ctx: CanvasRenderingContext2D, node: Rect | Node) => void,  anchorsFn?: (node: Node) => void,//注册锚点  iconRectFn?: (node: Node) => void,//注册图片或图标  textRectFn?: (node: Node) => void,//注册文字  force?: boolean  ) {  if (drawNodeFns[name] && !force) {  return false;  }  drawNodeFns[name] = drawFn;  anchorsFns[name] = anchorsFn;  iconRectFns[name] = iconRectFn;  textRectFns[name] = textRectFn;  return true;  }  // 注册分线盒  registerNode('fenxianhe',(ctx:CanvasRenderingContext2D,node:Rect)=>{},fenXianHeAnchors, fenxianheRect, fenxianheTextRect); |

图4.3 节点注册函数

以分线盒产品为例，注册为节点，其中fenXianHeAnchors为注册分线盒组件锚点的函数，fenxianheRect为注册分线盒区域范围的函数，fenxianheTextRect用来定义文本区域。

注册好各类产品节点之后，需要在画布中显示，画布中的三层所对应的三个类中，都有渲染画面的函数render，以浮动层为例，部分代码如下：

|  |
| --- |
| render(ctx: CanvasRenderingContext2D) {  ctx.save();  // 绘制节点边框  if (this.node && !this.data.locked) {  this.root = this.getRoot(this.node) || this.node;  if (this.root) {  ctx.save();  ctx.strokeStyle = this.options.dragColor;  ctx.beginPath();  ctx.strokeRect(this.root.rect.x, this.root.rect.y, this.root.rect.width, this.root.rect.height);  ctx.restore();  }  //省略  }  //绘制锚点  if (this.dockAnchor) {  ctx.beginPath();  ctx.arc(this.dockAnchor.x, this.dockAnchor.y, 4, 0, Math.PI \* 2);  ctx.fill();  }  //省略  ctx.restore();  } |

图4.4 浮动层渲染画布函数

最终分线盒产品在画布上显示的效果如图4.5所示，图为鼠标浮动到节点上时所展现出的效果，非浮动状态下只会显示图片和文字，隐藏边框及锚点，各部分图中已经标注。

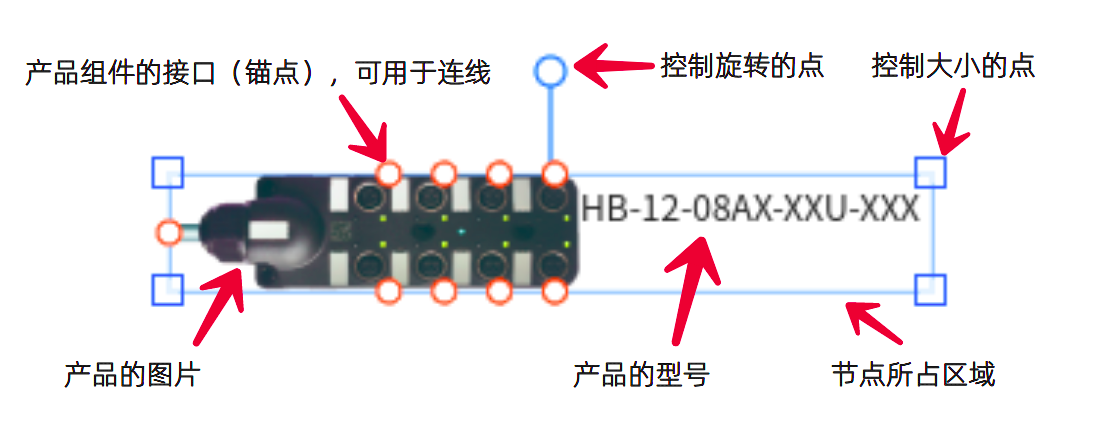


图 4.5 分线盒产品节点实现图

当选中一个节点时，右侧会显示出该节点的外观属性以及数据参数，对于外观属性可以随意更改，对于数据参数有些可以更改有些不可更改，具体实现效果如图4.6所示为分线盒节点的相关属性。

图4.6 分线盒节点的外观及数据参数图

前端页面部分整体的布局如图4.7所示，最上面部分为导航栏，包括文件处理，编辑操作，预览视图以及分享按钮等，左边是组件栏，根据组件用途分为多个类别，中间为画布部分，画布最上方有增加模板、删除模板以及校验电流按钮。最右边为属性栏，根据点击事件会分别显示出文件属性、产品属性、产品清单等多种数据。

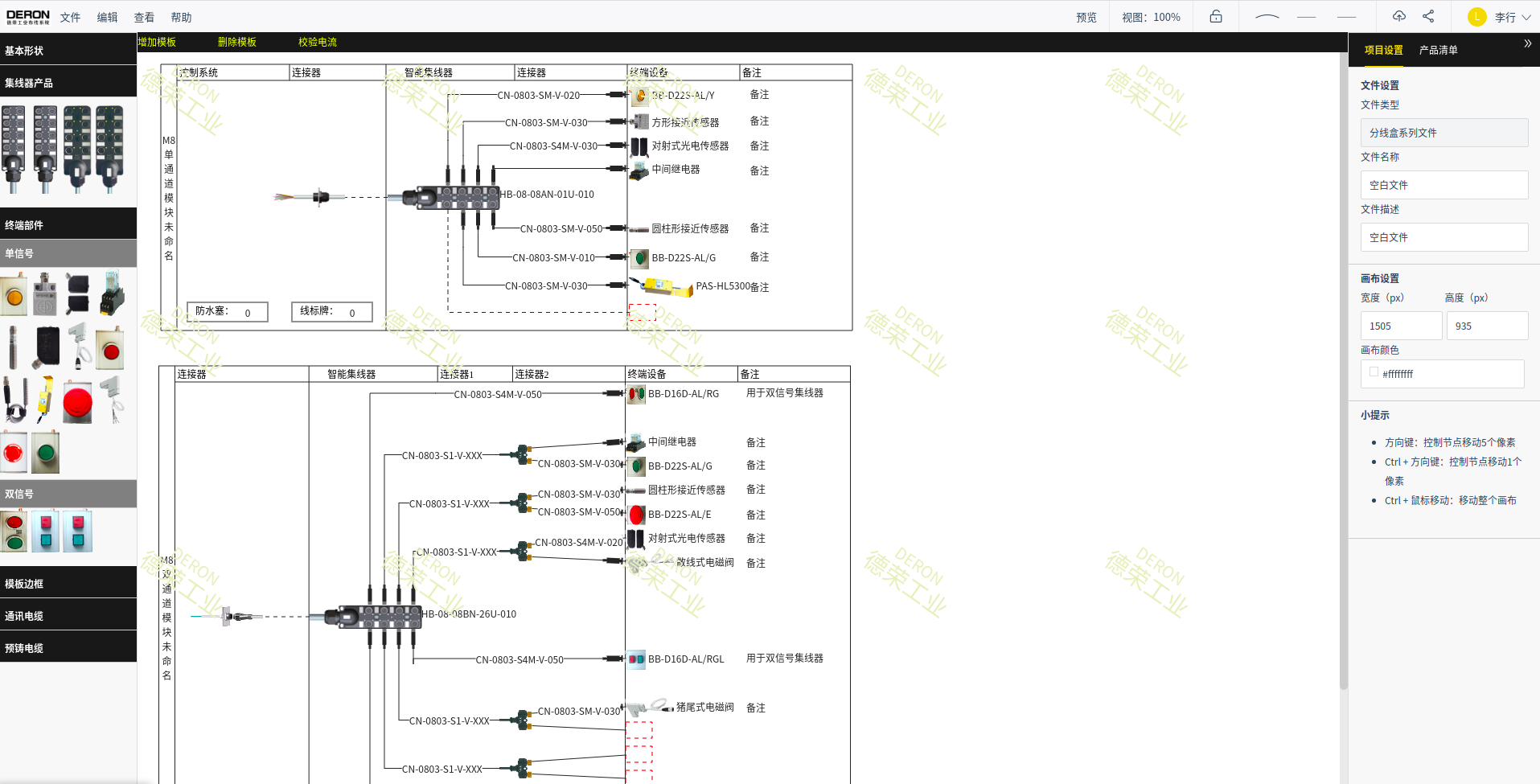


图4.7 布线系统整体效果图

在每一个需要添加节点的位置双击，都会出现如图4.8所示的弹窗，用来根据条件筛选出想要添加的节点，点击下方产品图片就可以将其添加到画布上，除此之外还可以从左边组件栏拖动组件添加到画布中。



图4.8 组件筛选弹窗实现图

## 4.3 图文管理模块的实现

图文管理模块中，主要是用于管理个人创建的一些文件，该部分的接口定义在topologyService中，以查找自己创建的文件列表为例，由于创建文件会有很多，为了方便展示，加入分页功能，并根据创建时间进行排序，如图4.9是其findTopologiesByUserId函数具体的代码实现。

|  |
| --- |
| public Map<String,Object> findTopologiesByUserId(Integer userId, Integer pageIndex,Integer pageCount){  SpringbootPageable pageable = new SpringbootPageable();  PageModel pm=new PageModel();  //根据创建时间排序  List<Sort.Order> orders = new ArrayList<Sort.Order>();  orders.add(new Sort.Order(Sort.Direction.DESC, "createdAt"));  Sort sort = Sort.by(orders);  Criteria criteria = new Criteria();  //userid查询  criteria.and("userId").is(userId);  Query query = new Query(criteria);  // 开始页  pm.setPagenumber(pageIndex);  // 每页条数  pm.setPagesize(pageCount);  // 排序  pm.setSort(sort);  pageable.setPage(pm);  // 查询出一共的条数  Long count = mongoTemplate.count(query, Topologie.class);  // 查询  List<Topologie> list = mongoTemplate.find(query.with(pageable), Topologie.class);  // 将集合与分页结果封装  Map<String,Object> smap = new HashMap<>();  smap.put("list",list);  smap.put("count",count);  return smap;  } |

图4.9 获取个人文件列表的实现代码

实现效果图如图4.10所示



4.10 图文管理实现效果图

图文管理中删除图文的操作，是通过TopologyController中的deleteTopologyByIdAndUserId方法实现的，该方法调用需要删除文件所对应图片，文件基本信息数据以及文件组网图数据，具体代码如下：

|  |
| --- |
| @CrossOrigin  @ResponseBody  @Transactional  @DeleteMapping(path = "/deleteTopology/{id}")  Public int deleteTopologyByIdAndUserId(HttpServletRequest httpServletRequest,@PathVariable(name = "id") String id){  Integer userId = null;  // 从 http 请求头中取出 token  String token = httpServletRequest.getHeader("Authorization");  // 获取 token 中的 user id  try {  userId = Integer.parseInt(JWT.decode(token).getAudience().get(0));  } catch (JWTDecodeException j) {  throw new RuntimeException("401");  }  //删除对应的图片  Topologie t = topologieService.findTopologieById(id);  String imageName = t.getImage();  File file = new File(fileurl+imageName);  if (file.exists() && file.isFile()) {  if (file.delete()) {  log.info("删除单个文件" + fileurl+imageName + "成功！");  } else {  log.error("删除单个文件" + fileurl+imageName + "失败！");  }  }  int ret = 1;  try{  topologieService.deleteTopologieByIdAndUserId(id,userId);  topologieDataService.deleteTopologieDataById(id);  }catch (Exception e){  ret = 0;  log.error(e.toString());  }  return ret;  } |

图4.11 图文管理删除文件具体代码

对于绘制好的组网图，用户可以以图片的格式下载到本地，但是会添加上德荣工业公司的水印以及特制的边框，如图4.9所示。

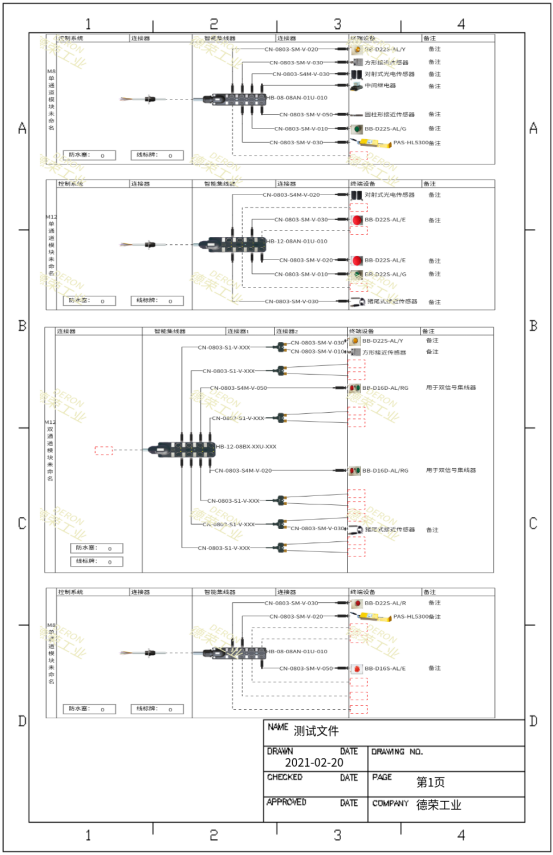


图4.9 组网图下载为图片效果图

## 4.4 收藏管理模块的实现

收藏管理模块主要用来管理用户收藏的一些文件，用户在热门图文区域搜索到自己感兴趣的图文，点击收藏按钮进行收藏，方便之后再次查看。该部分的实现接口定义在CollectionService中，由于添加收藏取消收藏等操作较为频繁，如果直接使用MySQL数据库会增加其压力，所以通过Redis来实现收藏和点赞等操作，再启动定时任务定时将收藏和点赞数据进行持久化。以收藏为例，具体的Controller层代码如下：

|  |
| --- |
| @CrossOrigin  @ResponseBody  @RequestMapping(path = "/addCollection")  public String addCollection(HttpServletRequest httpServletRequest, @RequestParam("topologyId")Boolean topologyId){  // 从 http 请求头中取出 token  Integer userId = null;  String token = httpServletRequest.getHeader("Authorization");  // 获取 token 中的 user id  try {  userId = Integer.parseInt(JWT.decode(token).getAudience().get(0));  } catch (JWTDecodeException j) {  throw new RuntimeException("401");  }  //调用collectionService中的addCollection方法，传入用户Id，被收藏的图文  boolean res = collectionService.addCollection(userId,topologyId);  return res;  } |

图4.10 CollectionController中的addCollection方法的实现代码

如图4.10所示，添加收藏和取消收藏调用同一个方法，当返回结果为true，代表收藏成功，当返回结果为false，代表收藏失败或取消成功。前端传入参数为图文标识topologyId，再通过从token中取出用户标识userId，通过collectionService中的addCollection方法操作Redis。

|  |
| --- |
| public boolean addCollection(Integer userId, String topologyId){  //拼接key  String key = "topology\_user\_collect\_"+userId;  String key2 = "topologyVo\_collection\_";  //如果该用户收藏过此图文，则取消收藏  if(redisTemplate.hasKey(key)&&redisTemplate.opsForList().range(key,0,-1).contains(topologyId)){  //将图文id从列表中移除  redisTemplate.opsForList().remove(key,0,topologyId);  //该文章的收藏数-1  redisTemplate.opsForZSet().incrementScore(key2,topologyId,-1);  }else {  //添加收藏  redisTemplate.opsForList().leftPush(key,topologyId);  //该文章收藏数+1  redisTemplate.opsForZSet().incrementScore(key2,topologyId,1);  }  return redisTemplate.opsForList().range(key,0,-1).contains(topologyId);  } |

图4.11 CollectionService中的addCollection方法代码实现

如图4.11所示，用到了Redis中两种数据结构，一种为List，主要用于暂时保存每个用户收藏的文件id，key为关键词+userId，value为文件id列表，采用List的原因是可以记录插入的顺序，把最近收藏的放在前面。另一种数据结构为zset，即有序集合，用来暂时保存每个图文被收藏的次数，zset参数为(K key, V value, double score)，key为自定义关键词，value为图文id，score为被收藏次数，使用Zset的原因是其可以根据收藏量score进行排序，显示出最受欢迎的图文。当用户点击收藏按钮时，会先从Redis中根据key找到用户收藏列表，判断列表中是否有该图文Id，如果存在说明已经收藏过，那么这次点击事件就是取消收藏事件，会将列表中该图文id删除，同时也将zset中该图文id所对应的score减一。如果list中不存在该topologyId，说明之前没有收藏过，这次点击事件为收藏操作，需要将其添加到列表最前面，然后将该图文的收藏量加一。以用户userId为key的收藏列表不会出现并发安全问题，但是以图文id为key的集合，在进行score更新时会存在并发安全问题，所以使用原子操作incrementScore可以解决。具体实现的效果图如图4.12所示。



图 4.12 热门图文实现效果图

## 4.5 组件管理模块的实现

组件管理模块分为组件分类管理和产品组件管理，具体的接口定义如表4.1所示。

表4.1 产品组件管理模块接口定义

|  |  |  |
| --- | --- | --- |
| 模块 | 接口名称 | 功能描述 |
| 组件分类管理 | getAllGroup | 获取所有分类信息 |
| getGroupByName | 根据名称获取分类信息 |
| saveGroup | 保存分类信息 |
| updateGroupById | 根据id更新分类信息 |
| deleteGroupById | 根据id删除分类信息，也会将其分类下的所有产品删除 |
| 产品组件管理 | saveProduct | 保存产品信息 |
| saveProductByGroup | 向group组内添加或更改产品信息 |
| getProductByGroup | 根据group获取到该分组中的所有产品信息 |
| saveImg | 上传产品图片 |
| deleteProductById | 根据产品id删除产品信息 |
| updateProductById | 根据产品id更新产品信息 |
| getProductByName | 通过产品名称搜索产品信息 |

组件管理模块中，做重要的是添加组件信息，然后布线系统前端通过获取到组件信息进行注册，用户才可以使用，图4.13为向分类中添加产品组件的前端代码实现，图4.14为后端代码实现部分。如图4.14所示，保存时需要先判断指定的分类是否还存在，如果不存在提示用户先创建分类，如果存在，判断该产品组件是否有productId，没有需要自动生成UUID作为productId。

|  |
| --- |
| async saveProduct(value: any) {  let id = '';  let groupId = '';  if (groupId === null || groupId === '') {  groupId = Store.get('component\_id');  }  const data = {  //数据部分省略  };  const product = new Merchant(data);  const ret = await this.productService.saveProduct(product);  if (ret.code === 200) {  //省略保存产品对应锚点数据的代码  this.\_notification.create('success', '保存成功', '组件保存成功,两秒后自动跳转');  setTimeout( \_ => {  this.router.navigate(['../'], { relativeTo: this.route });  }, 2000);  }else if (ret.code === -1) {  this.\_notification.create('error', '保存失败', ret.data.value);  }  } |

图4.13 保存产品组件前端代码实现

|  |
| --- |
| @CrossOrigin  @ResponseBody  @RequestMapping(value = "/saveProductbygroup")  public Response saveProductByGroup(@RequestBody Product product){  String groupid = product.getGroupId();  Response r = new Response();  //查询分类是否存在  if(groupid == null || groupid==""){  r.setCode(-1);  r.setData(new retInfo("error","group不存在，请先创建分类"));  return r;  }  //如果是新增产品组件，自动生成组件id  if(product.getId()==null || product.getId()==""){  product.setId(uuid.getUUID4());  }  String pid = this.productService.saveProduct(product);  r.setCode(200);  r.setData(new retInfo("id",pid));  return r;  } |

图4.14 保存产品组件后端代码实现

如图4.15为产品组件管理实现页面，可以在其页面进行搜索，删除指定组件，编辑组件信息，查看产品具体信息等功能。

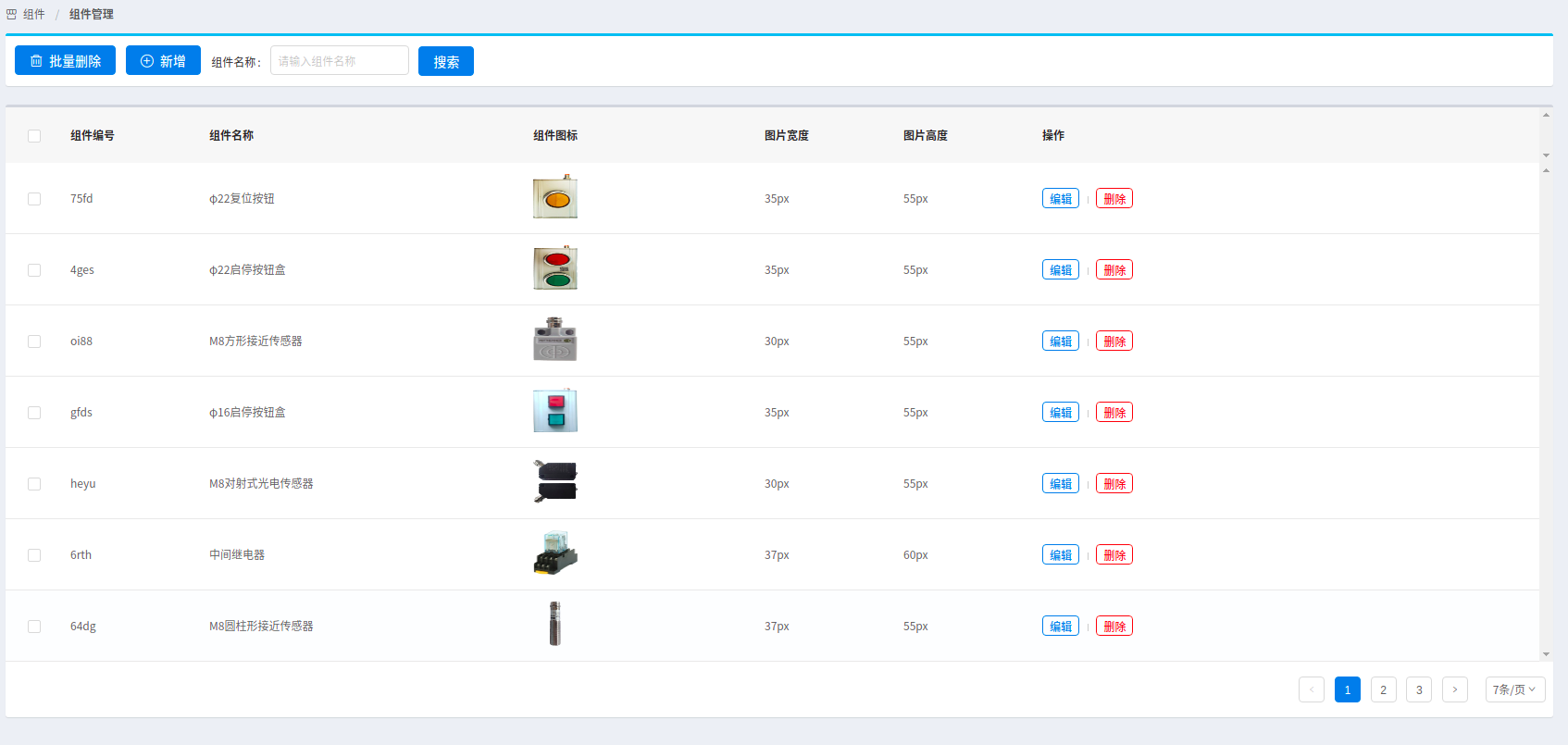


图4.15 产品组件管理实现页面

如图4.16为新增或修改产品信息实现页面。



图4.16为新增或修改产品信息实现页面

## 4.6 统计分析模块的实现

由第三章的模块设计可知，统计分析模块分为报表统计和流量统计两部分，下面详细介绍这两部分的实现。

1. 报表统计模块

报表统计模块中主要有三个子功能，TopologyService中定义了文件类型统计的服务端接口，ReportListService中定义了模板类型统计和产品组件统计的服务端接口，控制层分别由TopologyController、ModulesController和ReportNodeController来相应前端的请求，并对数据处理后返回给前端。如图4.17为TopologyController中的部分方法说明。

表4.2 文件类型统计模块的方法定义

|  |  |  |
| --- | --- | --- |
| 模块 | 方法名称 | 功能描述 |
| 文件类型统计模块 | getFileTypeByUserId | 查询每个用户创建的文件类型及其数量 |
| getFileTypeByUserIdAndCreateAt | 查询每个用户指定时间段创建的各类型的文件数量 |
| getNumByFileTypeAndUserId | 查询每个用户下指定类型的文件数量 |
| getNumByUserId | 查询每个用户创建的所有文件数量 |
| getAllFileTypeByCreateAt | 统计全部用户指定时间段创建各类型的文件数量 |

以文件类型统计为例，具体介绍其实现，如图4.17所示，为前端调用后端接口获取数据的代码，传入参数parm中包括起始时间以及用户id，0代表全部用户。获取到后端返回结果后，判断是否获取数据成功，如果成功将数据分组并其填充到Echart的option中。

|  |
| --- |
| this.orderService.getAllFileType(parm).subscribe(res => {  if(res.code==200){  this.option.series[0].data = this.series\_data;  let item= res.data;  let data = [];    //填充数据  data.push({value:item.KB,name:'空白文件'});//空白文件数量  data.push({value:item.MB,name:'ModBus/TCP协议文件'});  data.push({value:item.CC,name:'CC-Link IE协议文件'});  data.push({value:item.PR,name:'Profinet协议文件'});  data.push({value:item.EC,name:'EtherCAT协议文件'});  data.push({value:item.EN,name:'EtherNet/IP协议文件'});  //将数据赋值到Echart的option中，实现图形展示  this.FileOption.series[0].data = data;  }  // 改变对象的引用，触发输入属性的 ngOnChanges 事件  this.FileOption = Object.assign({}, this.FileOption);  }); |

图4.17 文件类型统计前端实现代码

如图4.18为文件类型统计后端的实现代码，getAllFileTypeByCreateAt方法接受三个参数，开始时间，结束时间和用户userId，判断userId是否为0，为0则代表统计所有用户数据，通过调用TopologyService中的相应方法获取到文件id列表，判断列表是否为空，若不为空，则通过topologieDataService中的getFileTypeById方法迭代获取所有文件id对应的文件类型，将每种类型所对应的数量通过ConcurrentHashMap保存，封装成Response对象返回给前端。

|  |
| --- |
| @CrossOrigin  @ResponseBody  @RequestMapping(path = "/getAllFileType")  public Response getAllFileTypeByCreateAt(  @RequestParam("startTime")Date startTime,  @RequestParam("endTime")Date endTime,  @RequestParam("userId")int userId){  Response response = new Response();  int code = 200;  List<String> topologyIdList = new ArrayList<>();  //如果userId为0，搜索全部用户，否则指定用户  if(userId==0){  topologyIdList = topologieService.getIdByCreateAt(startTime,endTime);  }else{  topologyIdList=topologieService  .getIdByUserIdAndCreateAt(userId,startTime,endTime);  }  //用来保存每种类型文件的数量  Map<String,Integer> map = new ConcurrentHashMap<>();  if(topologyIdList.size()>0){  for(String id:topologyIdList){  String type = topologieDataService.getFileTypeById(id);  //如果map中还未存储type为Key的数据，就默认为0再加一，如果已  //经存在则先取出值再加一，然后更新。  map.put(type,map.getOrDefault(type,0)+1);  }  }else{  code = -1;  }  response.setCode(code).setData(map);  return response;  } |

图4.18 文件类型统计后端实现代码

前端获取到所需数据之后，通过Echart中的图表组件展示出来，如图4.19所示为文件类型统计界面。

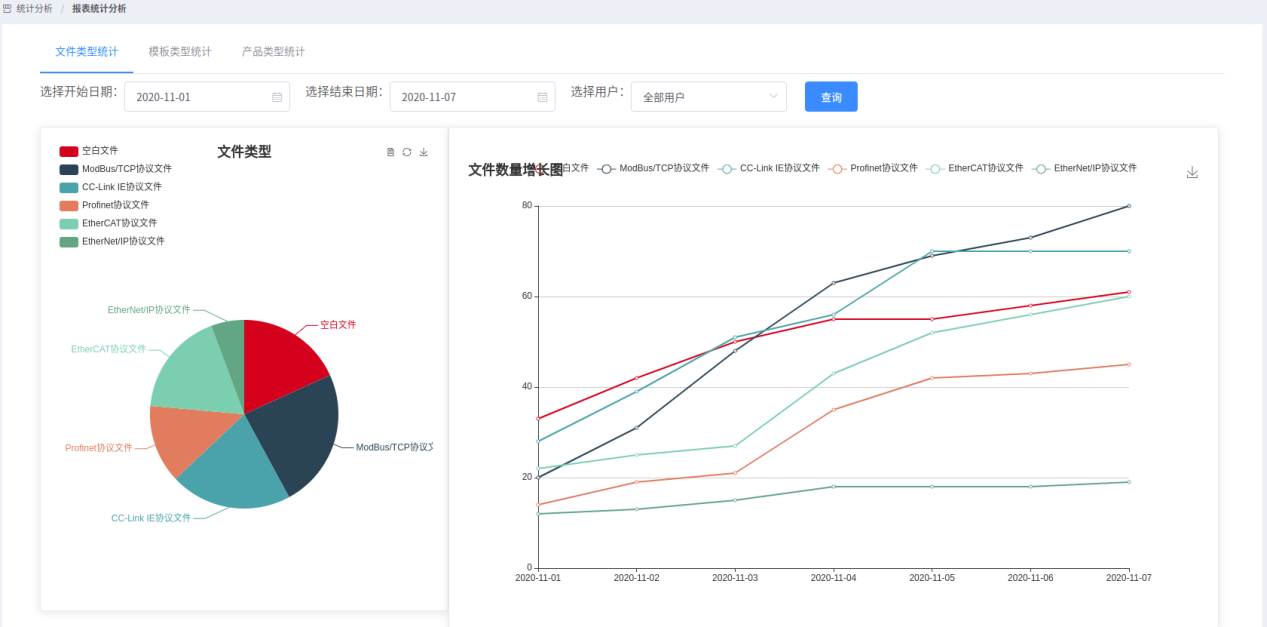


图4.19 文件类型统计界面

1. 流量统计模块

由第三章需求分析和详细设计可知，流量统计部分分为用户访问量统计，用户访问时长统计和常用用户统计三个部分，这三个部分的接口全部定义在VisitService中，如表4.3为流量统计模块服务端接口定义表。

表4.3流量统计模块接口定义表

|  |  |  |
| --- | --- | --- |
| 模块 | 接口名称 | 功能描述 |
| 用户访问量统计 | getDauSumByDate | 获取某个时间段的用户总访问量 |
| getDauByDate | 获取每天的用户访问量 |
| 用户访问时长统计 | getAvgTime | 用来获取某段时间内的平均访问时长 |
| getLongestTime | 用来获取某段时间内的最长访问时长 |
| getShortestTime | 用来获取某段时间内的最短访问时长 |
| getCountByTime | 用来获取各访问时长区间内的用户数 |
| 常用用户统计 | getUserByVisitTime | 获取某时间段内的用户访问次数排行 |
| getUserByVisitDuration | 获取某段时间内的访问时长排行 |

用户访问量统计部分的实现，是在Redis中维护一个Set数据结构，每当用户登录成功，从单点登录系统CAS跳转到该系统，在加载首页时，就会向后端发送请求，通过VisitRedisService中的addUserId方法向Redis中添加userId，使用Set是为了保证每个用户每天至多只统计一次。之后设置定时任务VisitMysqlService中的saveDAU方法，在每天快结束时将该日的访问量持久化到Mysql中的visit表并将Set置空。

用户访问时长统计部分，同样将数据临时保存到Redis中，key为visit\_time\_userId，value为当日该用户访问时长。时长获取通过window的 onload 和onbeforeunload 事件来监听页面的进入和离开，通过document.visibilitychange事件监听浏览器中tab的更换，通过angular中的ngOnInt和ngOnDestroy来监听页面的初始化和销毁。每当用户进入页面时记录当前时间戳作为开始时间，用户从页面离开、tab更换或页面被销毁时记录结束时间戳，差值为访问时长，在被触发的方法中将数据传递给服务端。服务端VisitRedisService中的addDuration方法判断Redis中是否有该userId为标识的key，若有则取出值，加上传入的时间戳作为新值再保存到Redis中，若没有则直接保存。再使用定时任务VisitMysqlService中的saveTOS方法将每个用户当日的访问时间持久化到Mysql中的user\_visit表并删除Redis中相应数据。

常用用户统计是根据以上持久化到的user\_visit表中的数据进行查询，根据访问次数排行首先以日期为条件筛选，然后再以userId分组，利用聚合函数count查询出每个用户对应的访问次数，降序排列取出前20，再与user表关联取出完整用户信息，访问时长排行查询过程类似。

具体实现以用户访问量为例，如图4.20为用户访问量统计实现代码。

|  |
| --- |
| //VisitRedisService中的addUserId方法，记录登录的用户。  public void addUserId(int userId){  String key = Config.DAU\_KEY;  redisTemplate.opsForSet().add(key,userId);  }  //VisitMysqlService中的saveDAU方法，每天23:59:00执行，将当日访问量持久化。  @Async //使用多线程执行定时任务  @Scheduled(cron = "0 59 23 \* \* ?")  public void saveDAU(){  String key = Config.DAU\_KEY;  Date date = new Date();  SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy");  String day = formatter.format(date);  Long count = redisTemplate.opsForSet().size(key);  try {  visitMysqlDao.saveDAU(day,count);  redisTemplate.delete(key);  }catch (Exception e){  log.error("持久化失败");  }  } |

图4.20 用户访问量统计实现代码

上面代码实现了每日访问量的获取与保存，还需要使用Echart库进行数据可视化，前端调用后端接口获取数据并填充页面的代码如图4.21所示。

|  |
| --- |
| visitNum() {  this.visitService.getVisitNum().subscribe(res => {  if (res.code === 200) {//获取数据成功  const dayList = [];  const visit= [];  const VisitNum = res;  VisitNum.forEach(item => {  dayList.push(item.date);  visit.push(item.count);  });  this.option.xAxis.data = dayList;  this.option.series[0].data = visit;  // 改变对象的引用，触发输入属性的 ngOnChanges 事件  this.option = Object.assign({}, this.option);  }else {  const \_noticeService: NoticeService = new NoticeService();  \_noticeService.notice({  body: '获取数据失败！',  theme: 'error'  });  }  });  } |

图4.21 用户访问量统计前端代码实现

实现效果图如图4.22所示，根据日期查询出该时间段的每日访问量，并计算该时间段的最大值，最小值以及平均值。

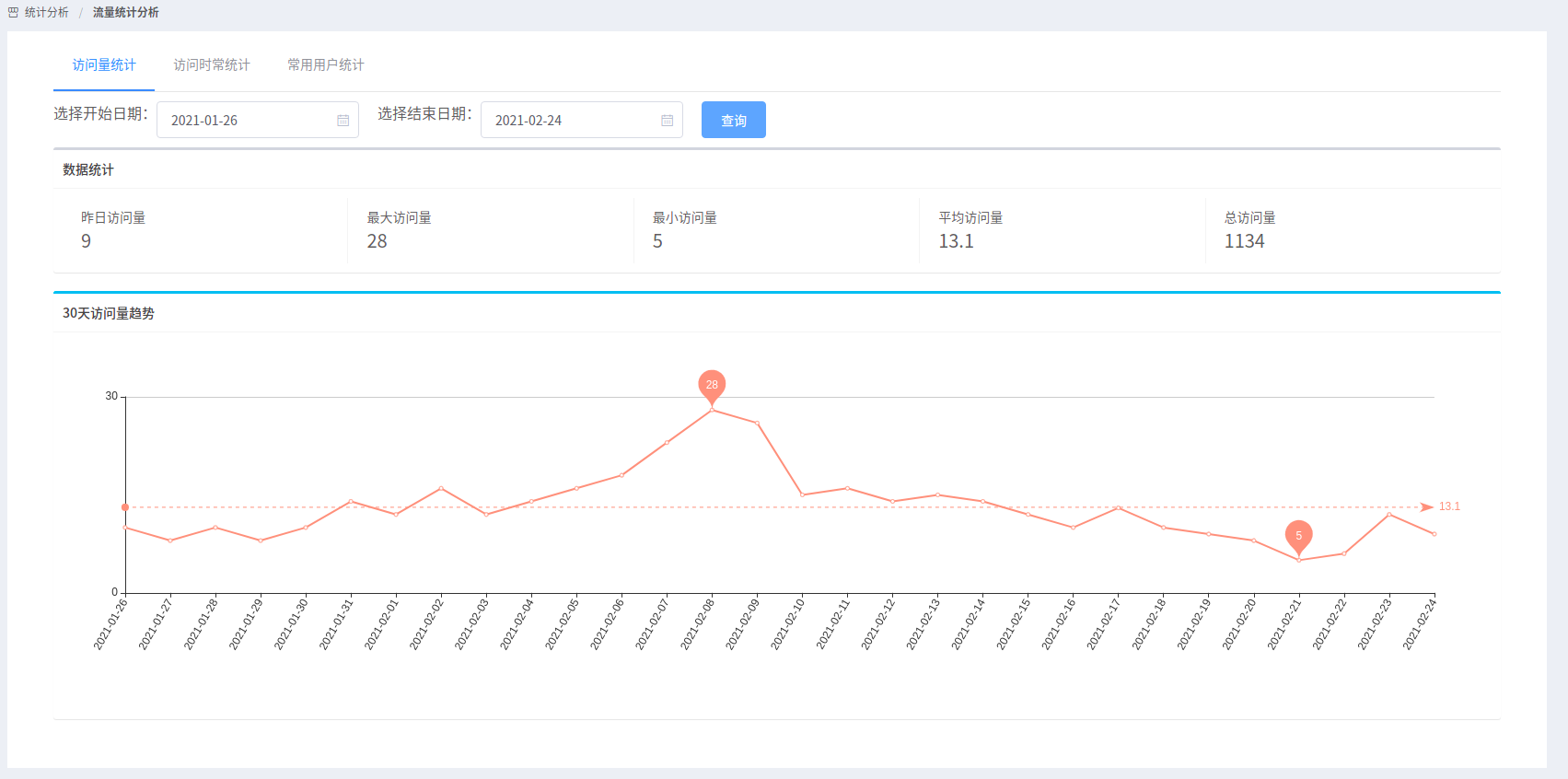


图4.22 用户访问量统计展示界面

## 4.7 留言管理模块的实现、

留言管理模块的接口定义在MessageService中，接口的实现由MessageServiceImpl类完成，通过调用MessageDao实现对数据库的增删改查操作，如表4.4为留言管理模块接口定义表。

表4.4 留言管理模块接口定义表

|  |  |  |
| --- | --- | --- |
| 模块 | 接口名称 | 功能描述 |
| 留言管理模块 | addMessage | 用户添加留言 |
| deleteMessageById | 用户删除自己创建的留言 |
| answerMessage | 用户回复别人的留言 |
| getMessageByTitle | 通过标题关键字模糊查询留言 |
| getMessageByType | 通过留言类型进行筛选 |

如表4.4所示，addMessage方法用来新增留言，需要的参数封装为对象Message，Message中属性包括创建者、创建时间、留言标题、类型、是否公开以及留言内容等。deleteMessageById方法通过留言唯一标识删除留言，当然只可以删除自己创建的留言。answerMessage方法实现回复别人的留言功能，将留言message\_Id、回复的内容、回复者user\_id、回复时间等信息保存到数据库answer表中。getMessageByTitle方法实现了对留言的查询，通过留言标题中的关键字可以查询出若干相关留言。getMessageByType方法是通过留言的类型，如产品问题、账户问题、系统问题等进行筛选。如图4.23为添加留言控制层的代码实现。

|  |
| --- |
| @CrossOrigin  @ResponseBody  @RequestMapping(value = "/addMessage")  public Response addMessage(  HttpServletRequest httpServletRequest,  @RequestBody Message message){  Response response = new Response();  int code = 200;  Integer userId = null;  String token = httpServletRequest.getHeader("Authorization");// 从 http 请求头中取出 token  // 获取 token 中的 user id  try {  userId = Integer.parseInt(JWT.decode(token).getAudience().get(0));  } catch (JWTDecodeException j) {  throw new RuntimeException("401");  }  //添加创建者  message.setUserId(userId);  //生成message\_id  String message\_id = uuid.getUUID4();  message.setMessageId(message\_id);  //添加创建时间  message.setCreateTime(new Date().getTime());  try{  messageService.addMessage(message);  }catch (Exception e){  code = -1;  //错误日志  }  response.setCode(code);  return response;  } |

图4.23 添加留言的代码实现

如图对前端传入的Message对象赋值message\_id，user\_id和createTime，然后调用messageService的addMessage方法将数据保存到数据库中。下图4.24为添加留言实现界面。



图4.24 添加留言实现界面

## 4.7 本章小结

本章介绍了工业布线系统各模块的实现，包括组网图设计模块、图文管理模块、收藏管理模块、组件管理模块、统计分析模块以及留言管理模块，通过接口定义、关键方法的代码实现和关键部分的页面展示来分别介绍各模块的实现方式，同时阐述了开发过程中所遇到的各种难点以及最终解决方式。

# 第五章　德荣工业现场总线布线系统的测试

## **5.1 系统测试概述**

### **5.1.1 系统测试环境**

德荣工业现场总线布线系统的测试环境如表5.1所示。

表5.1 系统测试环境配置表

|  |  |
| --- | --- |
| 环境名称 | 环境详细说明 |
| 部署环境 | 腾讯云Ubuntu服务器、Nginx、JDK8 |
| 开发环境 | Ubuntu18、JDK8 |
| 单元测试环境 | Ubuntu18、JUnit4 |
| 功能测试环境 | Ubuntu18+Chrome、Windows10+Chrome |
| 性能测试环境 | Ubuntu18、JMeter |
| 数据库 | Redis、MySQL、MongoDB |

系统的部署环境采用的是腾讯云服务器，配置为4核8G+200G数据盘，需要安装的软件和工具包括Nginx、JDK8、Redis、MongoDB、MySQL等，将后端程序代码编译为Jar包，由于SpringBoot已内置了Tomcat，可直接使用命令运行后端服务。前端代码编译之后放入服务器中指定文件夹，通过在nginx.conf中配置虚拟主机，监听指定端口，并跳转到前端代码所在位置，充当静态服务器，当横向拓展时可用于负载均衡。单元测试环境为Ubuntu18系统和JUnit4。功能测试环境由于用户多Windows用户，另外使用Windows10系统和Chrome浏览器。性能测试主要使用工具为JMeter。缓存使用Redis，使用MySQL和MongoDB对数据持久化。

5.1.2 测试工作描述

德荣工业现场总线布线系统的测试工作主要包括单元测试、功能测试、定时任务测试以及性能测试

1. 单元测试

单元测试是指对软件中的最小的可测试单元进行检查与验证【百科】，这里的单元通常指类，为每一个类创建一个对应的测试类，为类中的每个方法创建一个测试函数，用来测试方法执行结果和预期结果是否一致。每完成一个类就要完成其测试用例的编写，单元测试越早越好。

1. 功能测试

功能测试也叫黑盒测试或数据驱动测试，只需要考虑需要测试的各个功能，不需要考虑整个软件的内部结构及代码【百科】，用例设计通常包括等价类划分、边界值分析法、错误推测法等。本文将采用等价类划分法和边界值分析法两种方法进行功能测试，等价类划分是将所有可能的输入划分为若干子集，然后从子集中选出几个具有代表性的数据作为测试用例【】。边界值分析法是等价类划分法的补充，用来测试输入参数的边界情况。

1. 定时任务测试

系统中存在多个定时任务，对于定时任务需要测试其业务逻辑是否正确，开始执行的时间是否正确以及测试执行时长。系统中使用的定时任务多在每日0点之前触发，主要用来将Redis中的数据持久化到Mysql中，同时清空Redis。对于业务逻辑的测试，需要保证数据的正确性、完整性以及一致性。对于触发时间的测试需要保证cron表达式书写正确。

1. 性能测试

性能测试主要是通过测试工具模拟各种条件对系统的各项指标进行测试，性能测试可分为基准测试、负载测试、压力测试、稳定性测试、并发测试等多种测试类型【】。本文主要采用负载测试和压力测试对系统性能进行测试，负载测试是通过不断增加负载，找到系统性能指标变化的拐点，从而预估系统所能承受的最大并发数。压力测试指的是在一定的负载条件下长时间运行，查看系统是否正常运行，各项指标是否正常。压力测试又分为高负载下长时间的稳定性压力测试和极限负载情况下的破坏性测试【】。

## **5.2 单元测试**

极限编程的思想要求在编写代码之前先编写好测试代码，写一点测试一点，能够快速的发现问题，减少追踪错误的时间。Java中最常用的单测试框架是Junit,它在测试驱动开发方面有着重要发展，本文将采用Junit4进行单元测试，针对每一个类创建一个对应的测试类，对类中的大部分接口或方法进行单元测试，特别简单的方法除外。Junit中提供了注解，在测试方法上标注@Test，使用@Befor标注需要在测试方法执行前执行的方法，使用@After标注需要在测试方法执行后执行的方法，同时Junit还提供了大量的断言方法，用于判断是否相等的assertEquals方法，用于判断是否为真的assertTrue方法等。

测试报告如表5.2所示。

表5.2 系统单元测试报告表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 包名 | 单元测试数量 | 失败 | 通过 | 通过率 | 代码覆盖率 |
| com.deron.topology.controller | 80 | 0 | 80 | 100% | 86% |
| Com.deron.topology.service | 42 | 0 | 42 | 100% | 62% |

## **5.3 功能测试**

功能测试是为了验证软件应用程序的功能是否符合预期，主要通过用户图形界面、程序接口进行验证，接下来对每个模块中的关键功能点进行测试，列出测试用例表。

### **5.3.1 组网图设计模块功能测试**

组网图设计模块主要功能点包括各种类型文件的创建、保存、打开、导入、下载为JSON、下载为PNG等，各种模板的添加、删除、重命名等，各种组件和连线的添加、删除、属性更改、复制粘贴等，产品清单的生成、下载等，还有电流校验等功能。

如表5.3为组网图设计模块的测试用例，所有测试结果和预期效果相同，测试全部通过。

表5.3组网图设计模块测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试功能点 | 前置条件 | 测试步骤 | 预期结果 | 测试结果 |
| 新建文件 | 用户进入首页或在画布页面 | 用户点击导航创建文件，选择文件协议类型 | 创建出相应协议文件的空白画布 | 通过 |
| 打开最近文件 | 用户进入画布页面 | 用户下拉文件导航栏，点击打开最近文件，在列表中选择需要打开的文件 | 重置画布，显示打开的文件内容 | 通过 |
| 项目设置 | 用户进入画布页面 | 用户点击右侧项目设置，可对文件名称，文件描述，画布宽高以及颜色设置 | 文件名称文件描述实现数据绑定，画布宽高以及颜色设置可以随即在画布中显示效果 | 通过 |
| 添加模板 | 用户已创建或已打开文件 | 用户下拉画布上方的添加模板，点击需要添加的模板类型 | 在画布的空白处新增模板，新增的模板类型和选择的相同 | 通过 |
| 删除模板 | 用户已在文件中添加过模板 | 用户下拉删除模板下拉框，点击需要删除的模板名称 | 画布中相应的模板被删除，在该模板下方的组件或上移 | 通过 |
| 添加组件 | 用户已创建或打开文件 | 用户选择左侧产品栏，拖拽出选择的组件，添加在画布中的相应位置。或用户双击模板上的预留区域，在弹窗 中根据条件筛选出组件，点击添加 | 组件可以成功添加到画布中，如果有连线，会自动与连线吸附在一起 | 通过 |
| 添加连线 | 用户已创建或打开文件 | 用户在某个组件锚点引出连线，连接两个组件 | 组件被成功连接，并根据起点和终点组件类型，自动生成连线接头和型号 | 通过 |
| 快捷键 | 用户进入画布页面 | 用户选中需要操作的组件或连线，可使用快捷键进行复制，粘贴，删除，剪切等操作 | 组件被成功添加和删除 | 通过 |
| 组件属性设置 | 用户进入画布页面 | 用户点击组件，右侧栏中会显示该组件的外观属性以及产品参数，可对外观属性进行更改 | 外观属性的更改会实时显示在画布中 | 通过 |
| 电流校验 | 用户已经设计好拓扑图 | 点击画布上方校验电流，可选择某个模块校验，也可校验整个文件 | 会显示每个模块入电流和出电流，计算是否通过校验 | 通过 |

### **5.3.2 图文管理模块功能测试**

图文管理模块只要功能点为

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 导入文件 | 用户进入画布页面 | 用户下拉文件导航栏，点击导入文件 | 弹出文件框，选择需要导入的json文件，点击之后再画布中显示文件内容 | 通过 |
| 下载为图片 | 用户已创建好拓扑图 | 用户下拉导航栏，点击下载为图片，输入相关信息 | 文件以图片的形式被成功下载到本地，并添加特制边框 | 通过 |

### **5.3.3 收藏管理模块功能测试**

### **5.3.4 组件管理模块功能测试**

### **5.3.5 统计分析模块功能测试**

### **5.3.2 留言管理模块功能测试**

## **5.4 定时任务测试**

## **5.5 性能测试**

## **5.4 本章小结**