

**指导老师：**

**学院：信息工程学院**

**班级：软件1902班**

**学号：**

**姓名：**

**基于微服务架构，前后端分离的**

**权限管理系统**

目录/Contact

第一章：系统介绍 ……………………………………………………………………………………………… 001

第二章：系统安装 ……………………………………………………………………………………………… 001

第三章：SQL设计 ……………………………………………………………………………………………… 001

第四章：后端模块 ……………………………………………………………………………………………… 001

第五章：后端模块 ……………………………………………………………………………………………… 001

第六章：项目总结 ……………………………………………………………………………………………… 001



第一章：系统介绍

Mango后台权限管理系统（以下称为“系统”）是基于Spring Boot、Spring Cloud、Vue.js、Element UI等主流前端/后端技术，采用了前后端分离架构的权限管理系统，采用了较多前卫的框架以及技术。

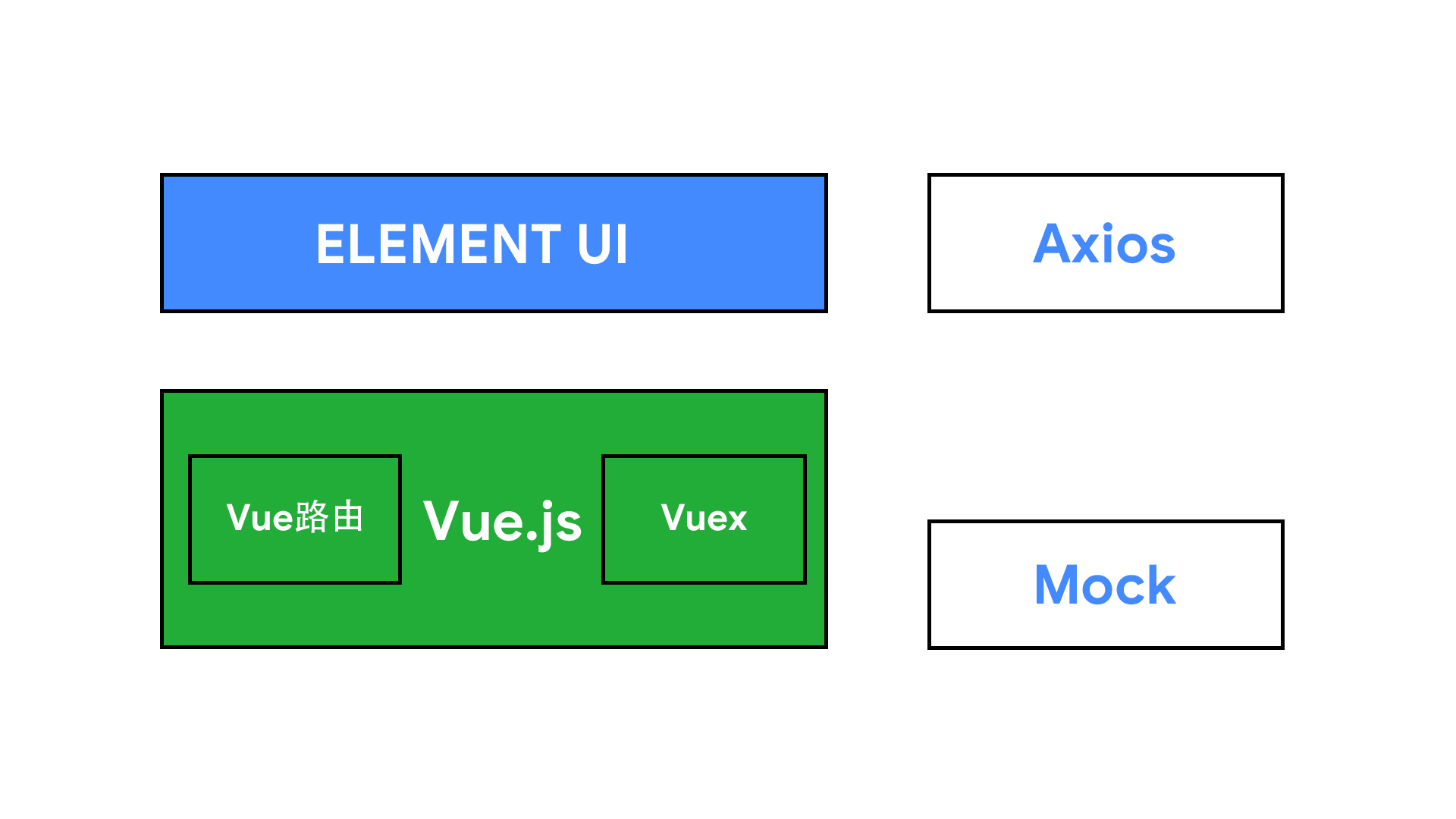
系统架构分为：

|  |  |
| --- | --- |
| 前端 | 后端 |
| Vue.js 核心框架 | Spring Boot |
| ElementUI 核心UI | Spring Security |
| Axios 前后端交互 | Spring Cloud |
| Mock 模拟接口数据 | MyBatis |
|  | Consul注册中心 |
|  | M2构建工具 |
|  | Mysql数据库 |

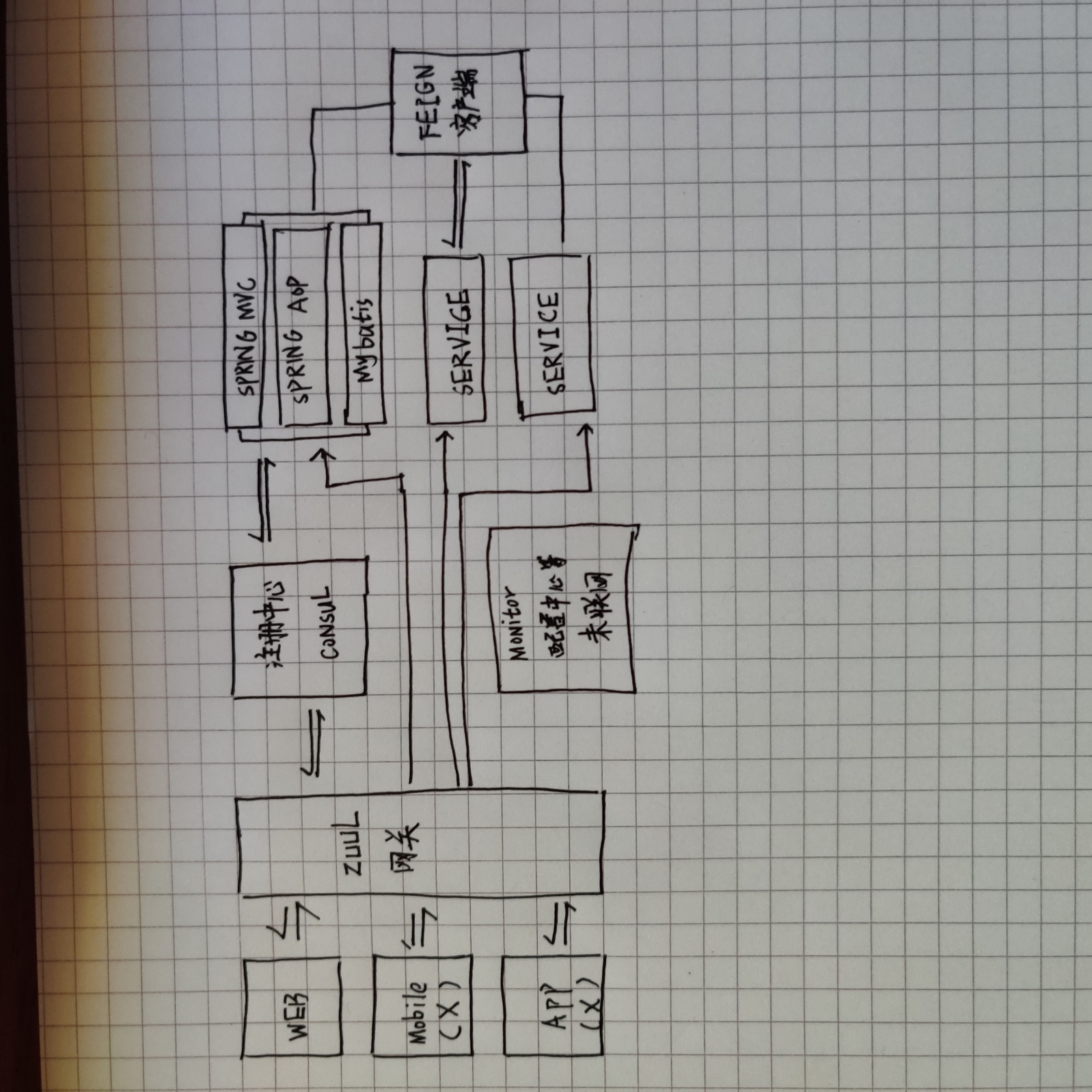
系统功能有：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能模块 | 完成√未完成 × | 是否有未解决BUG | 备注 |
| 用户登录 |  |  |  |
| 用户管理 |  |  |  |
| 机构管理 |  |  |  |
| 角色管理 |  |  |  |
| 菜单管理 |  |  |  |
| 字典管理 |  |  |  |
| 配置管理 |  | 不记录 |  |
| 登录日志 |  | 不记录 |  |
| 操作日志 |  | 不记录 |  |
| 在线用户 |  | 显示错误 |  |
| 数据监控 |  |  |  |
| 聚合文档 |  |  |  |
| 备份还原 |  | 没集成到前端 |  |
| 主题切换 |  | 无法保存切换颜色 |  |
| 服务治理 |  | 未嵌套到前端 |  |
| 服务监控 |  | 未嵌套到前端 |  |
| 服务消费 |  | 未嵌套到前端 |  |
| 服务熔断 |  |  |  |
| 服务网关 |  | zuul无法正常使用 |  |
| 服务消费 |  | 一堆BUG |  |

前端架构示意：



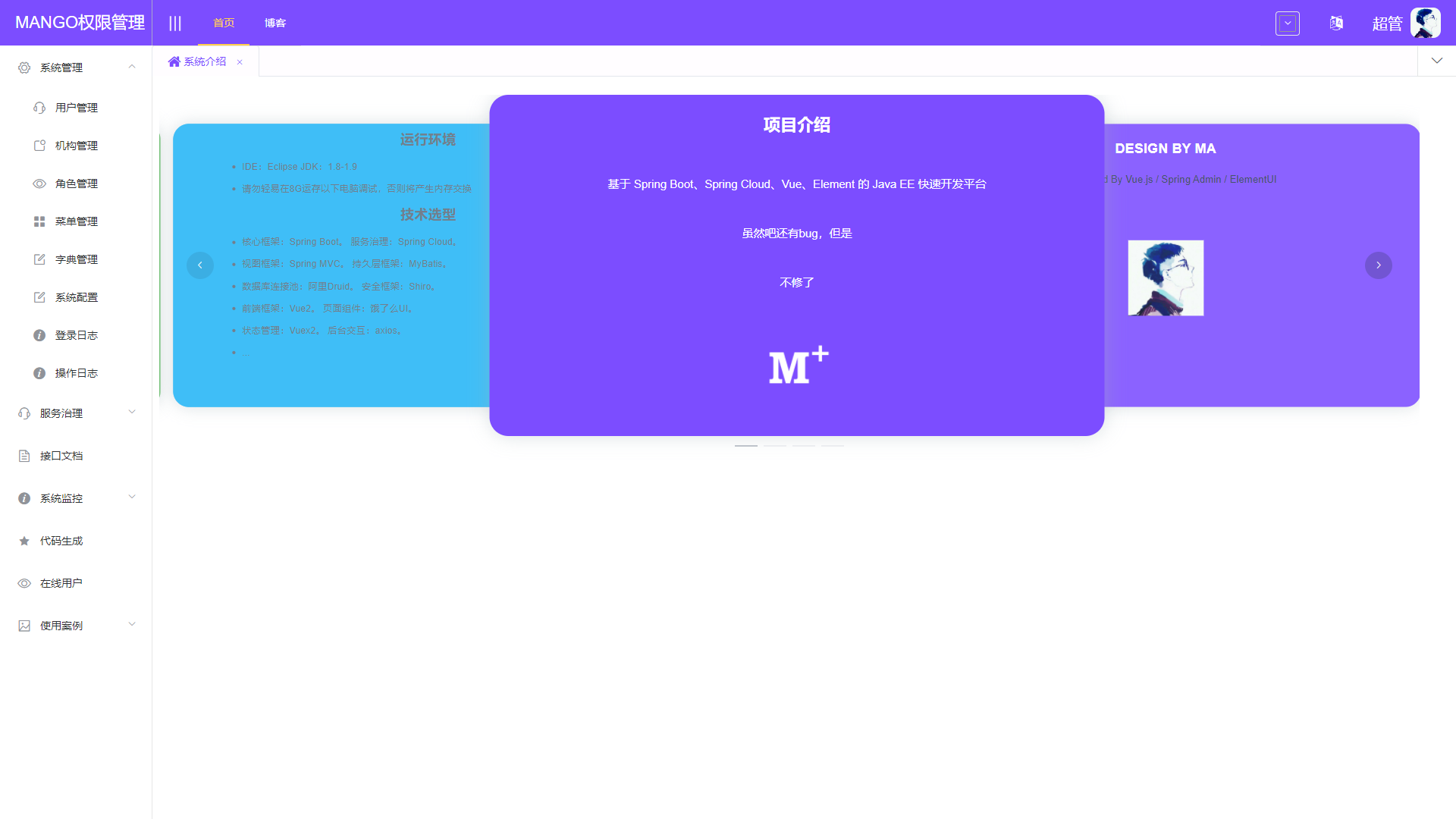
后端结构示意：手绘更舒服更直观



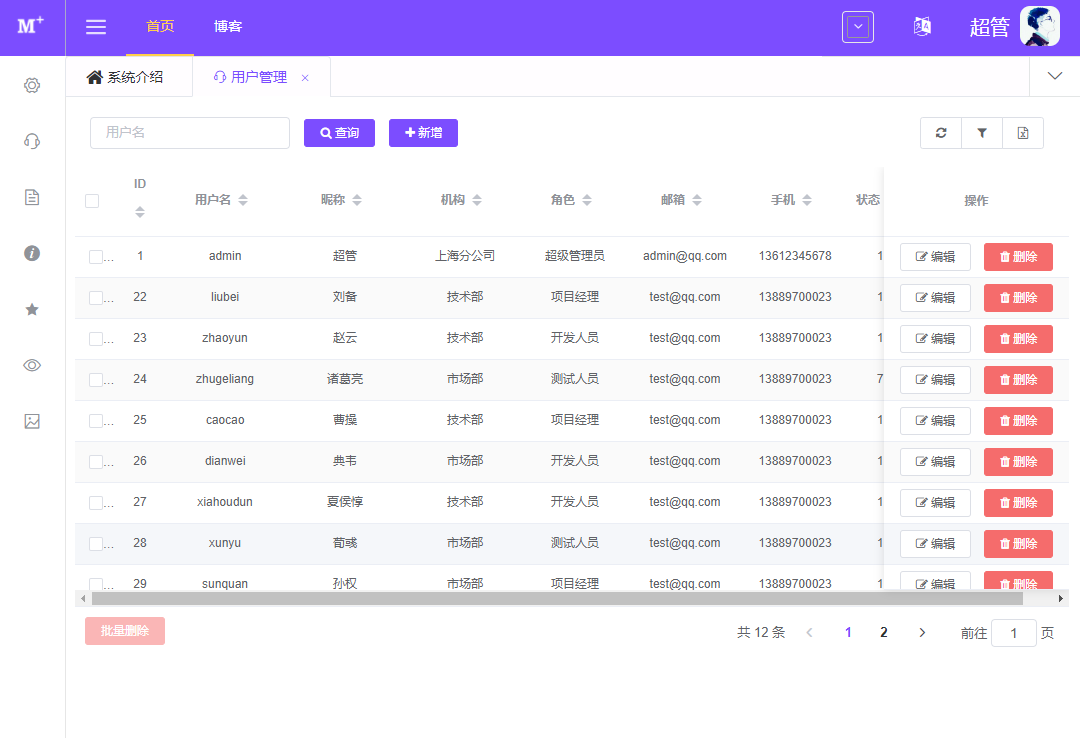
登录页面



主界面



用户管理页面



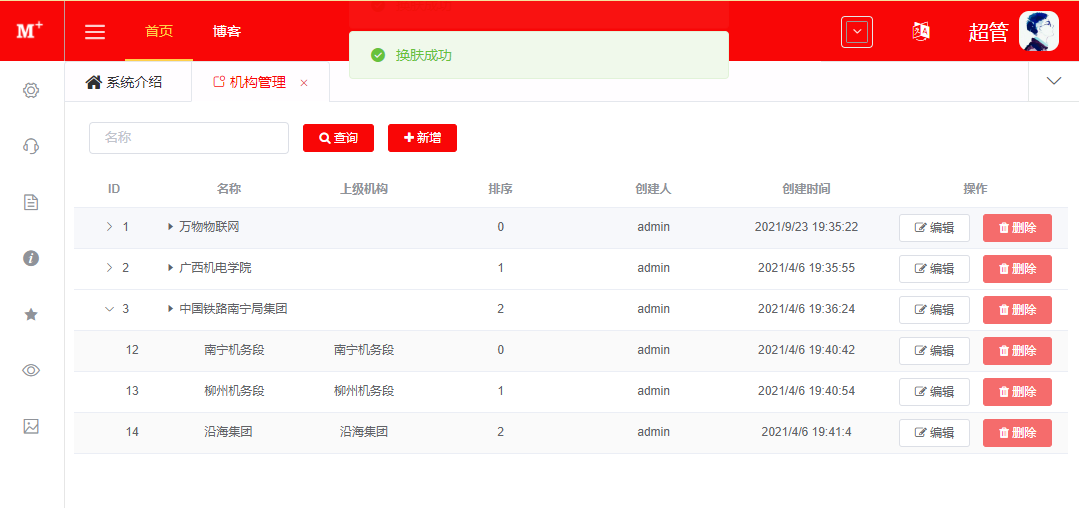
角色管理页面



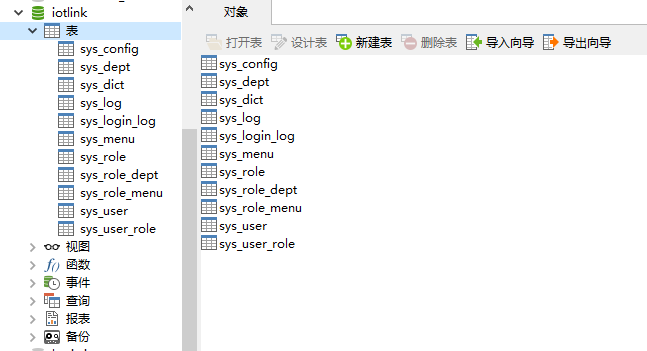
用户面板



换皮肤功能



数据库表





第二章：系统安装

系统安装，需要提前准备NPM环境

1. 下载Node.js进行安装

<https://nodejs.org/dist/v14.17.0/node-v14.17.0-x64.msi>

1. 导入SQL文件到数据库，在Resoure文件夹配置数据库链接、密码
2. 下载Conusl进行运行，Consul.exe项目内已附带。

打开CMD命令，CD到consul.exe所在文件夹，运行命令启动注册中心

consul agent -dev

必须先启动注册中心，否则将会报错

1. 导入项目到Eclipse，使用导入MAVEN方式导入以下全部项目，然后运行绿色标出项目

Admin

Backup

Common

Config

Consumer

Core

Hystrix

Monitor

Pom

Product

Product2

Zipkin

Zuul

其中Admin、backup、monitor必须运行，如果运行全部项目请保证电脑拥有8GB左右运行内存，全部展开将会占用至少4096MiB运行内存

1. 使用CMD命令，CD到Mango-前端项目文件夹，执行

npm install

进行模块安装，再执行

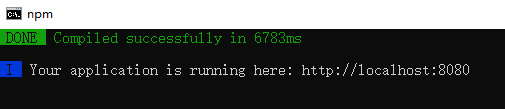
npm run dev

进行启动项目，启动完成后可在

<http://127.0.0.1:8080/>

执行项目登录

出现以下界面即为运行成功

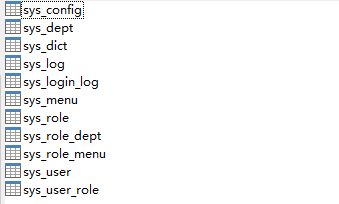






第三章：SQL设计

数据库表结构非常简单，简单解析一下



sys\_user 用户表：用户树表

sys\_role 角色表：角色树表

sys\_dept 机构表：机构树表

sys\_menu 菜单表：菜单目录、菜单、操作按钮，可以进行权限控制菜单可用

sys\_user\_role 用户角色表：也是中间表

sys\_role\_menu 角色菜单表：角色和菜单的中间关联表

sys\_role\_dept 角色机构表：角色和机构的中间表，关联表

sys\_dict 字典表：存储常用的枚举类型数据，节省系统资源

sys\_config 配置表：存储系统配置信息

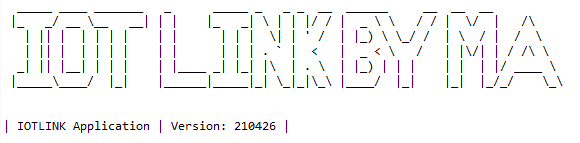
sys\_log 操作日志表：记录用户日常操作消息

sys\_login\_log 登录日志表：记录用户登入登出状态



第四章：后端模块

每个模块都有独立的banner文件，存放于/resource文件夹的banner.txt内，效果如下：

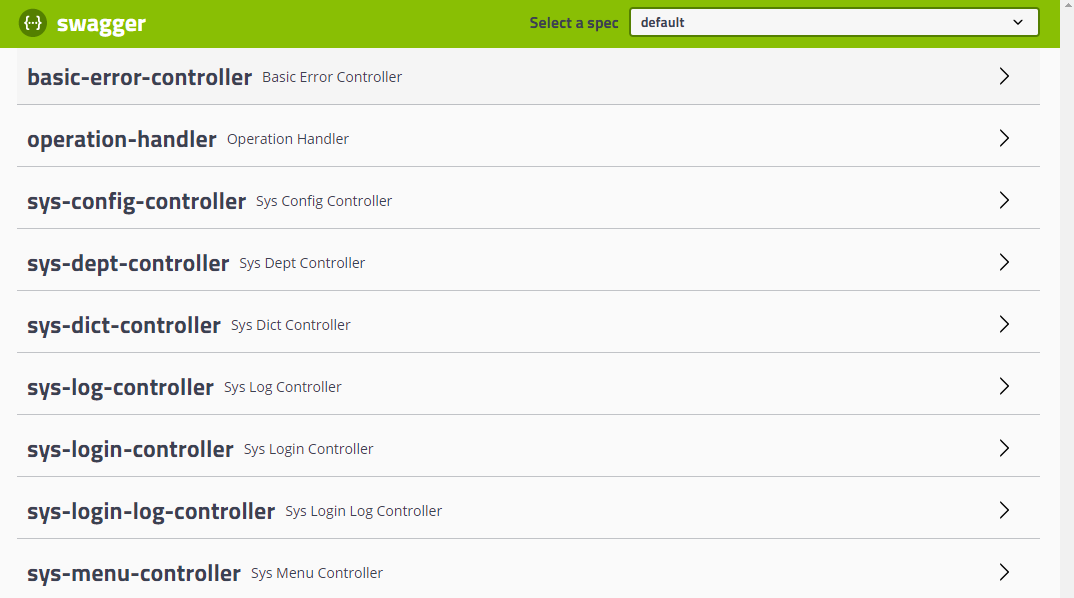


版面有限，项目就不一一列举剖析，挑几个特别的模块讲解吧。

1. Swagger文档，配置文件在admin项目内的Config包

接口地址<http://127.0.0.1:8001/swagger-ui.html>

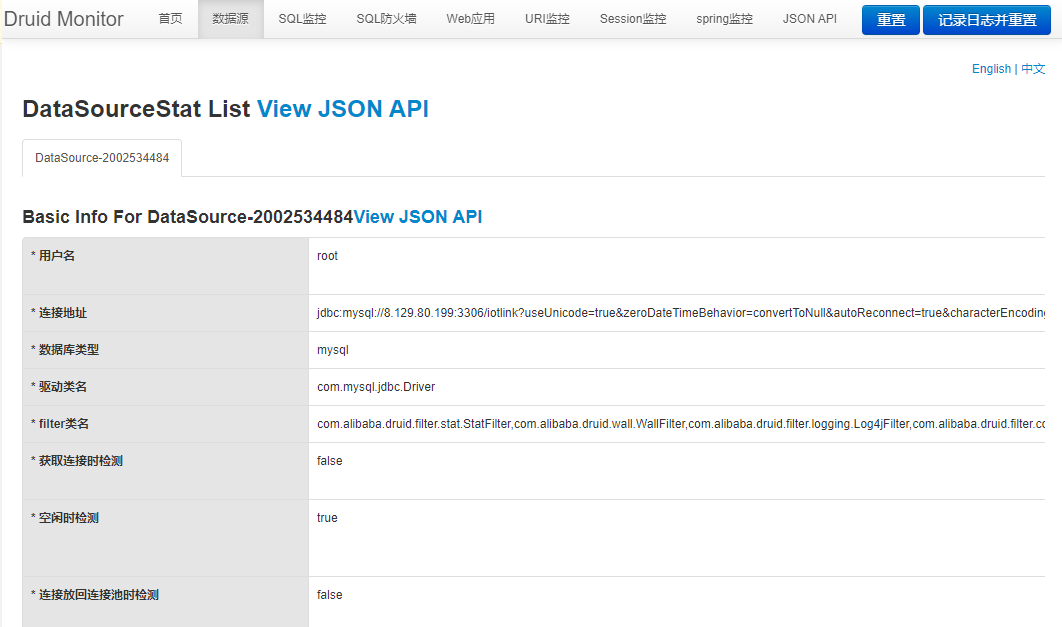
这个接口文档用来测试是最合适不过的



1. 阿里Druid数据源，这是一个连接池组件，提供sql监控和扩展，对于需要SQL优化的项目也挺方便的

接口地址：<http://127.0.0.1:8001/druid/index.html>

特别注意，这个组件还需要Log4j的lib，需要在pom.xml引入



1. CORS实现 跨域解决方案，配置文件详见admin项目/config/WebMvcConfigurer

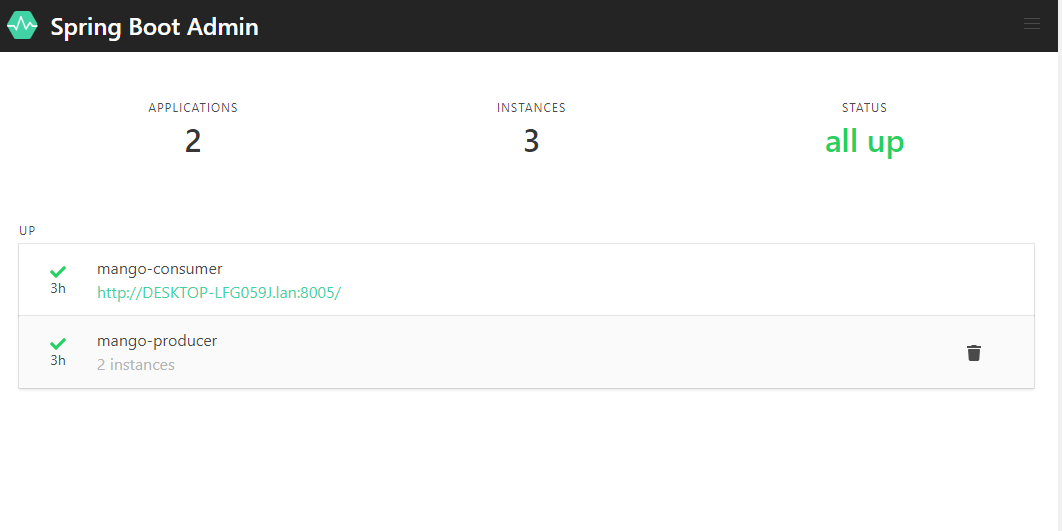
没前端的，也没怎么验证是否在工作

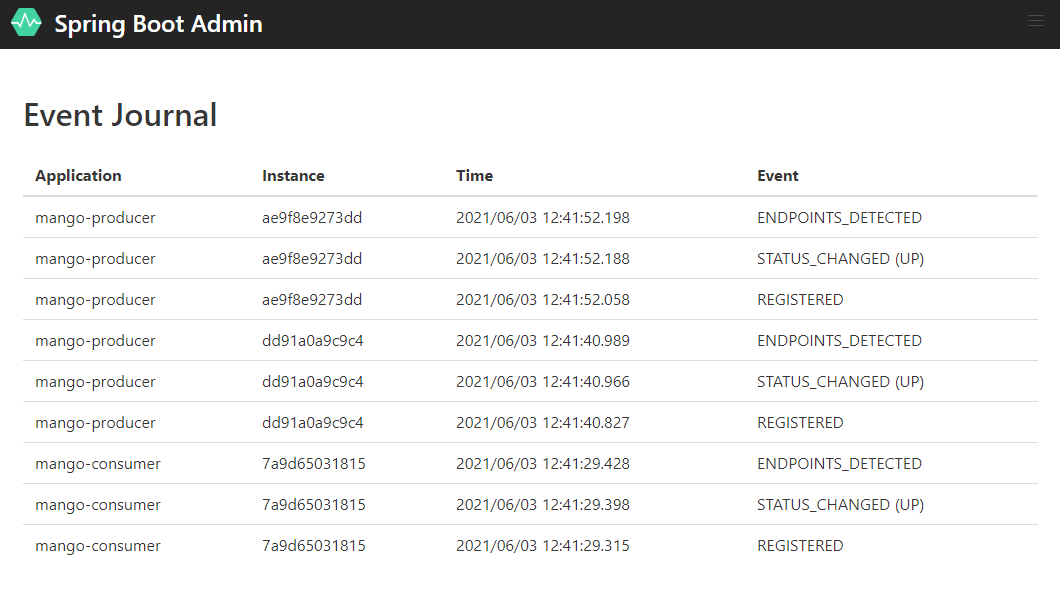
1. 导出Excel表

（配置文件在common项目下）没完成。

1. Kaptcha登录验证码，集成在admin项目，登陆时可见，配置文件也在config包下
2. 备份还原项目（未完成）
3. 系统服务监控，基于springAdmin的一个小监控插件，需要在yml文件里注册到

接口地址<http://127.0.0.1:8000/#/applications>

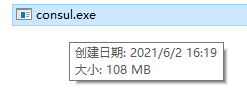


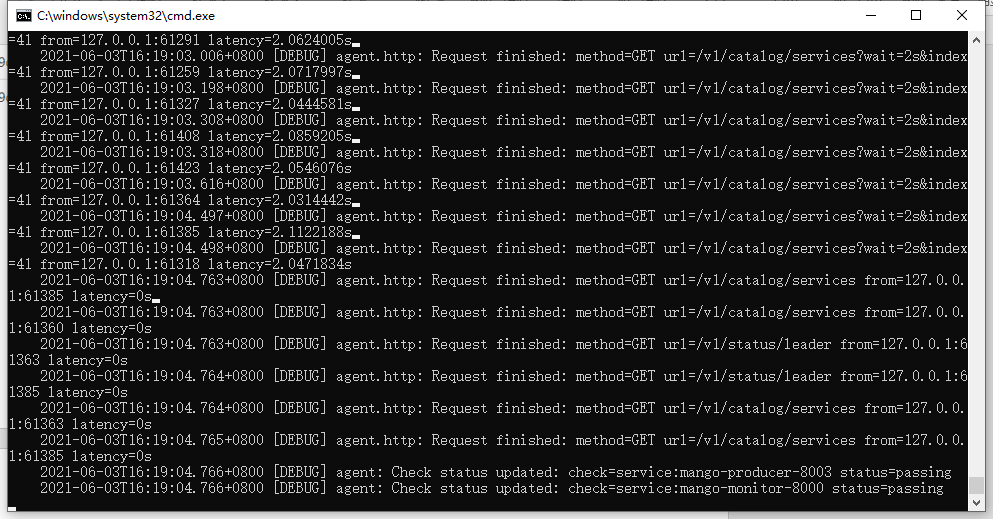


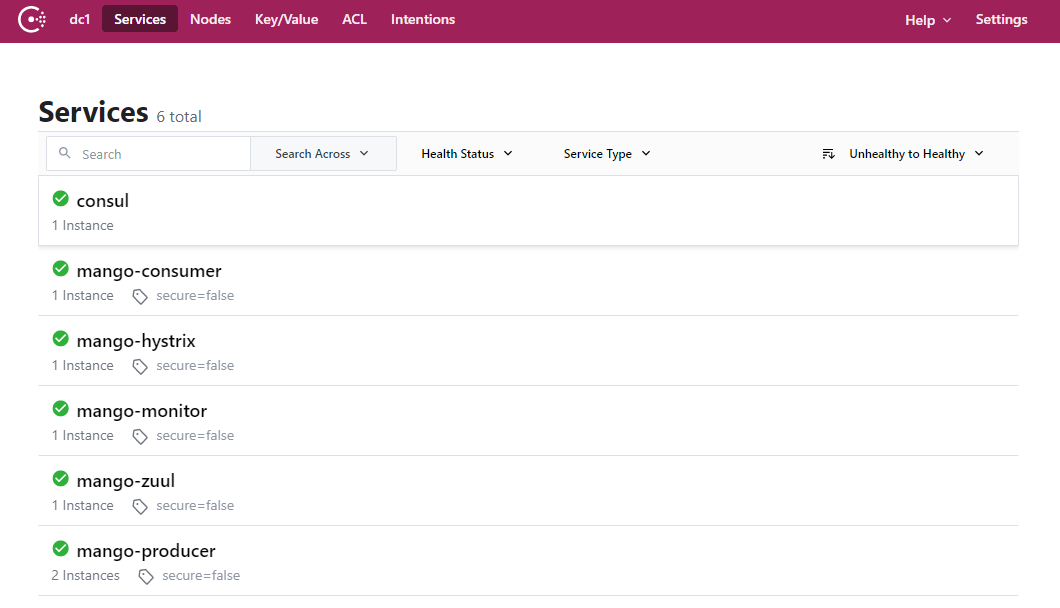
⑧注册中心，主要使用健康检查功能，可以看见目前注册到注册中心的项目（在线项目）

运行方式：cd到目录，执行命令consul agent -dev运行

<http://127.0.0.1:8500/ui/dc1/services>



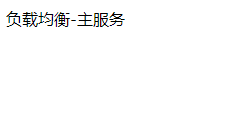
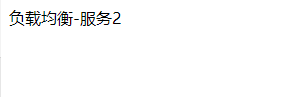




⑨负载均衡

<http://127.0.0.1:8005/feign/call>

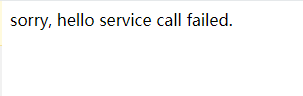
可以启动全部项目，然后访问以上链接，刷新时可以看到轮询的负载均衡服务，主要是测试product和product2的负载均衡，仅做测试，未作实际应用：）



⑩ 服务熔断器（Hystrix）已经设计并连接完成，会自动检测项目是否在线，不在线的会返回服务不可用，服务回复后可以自动恢复，做集群必备。

<http://127.0.0.1:8005/feign/call> 反复刷新

停掉其中一个product会提示不可用，而另一个并不会受影响，而且另一个修好后会自动恢复。

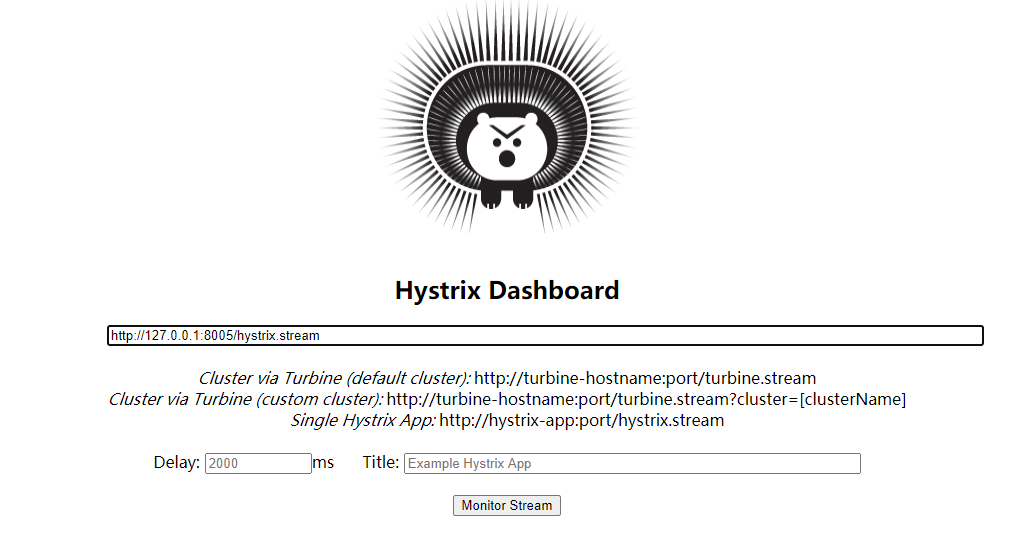


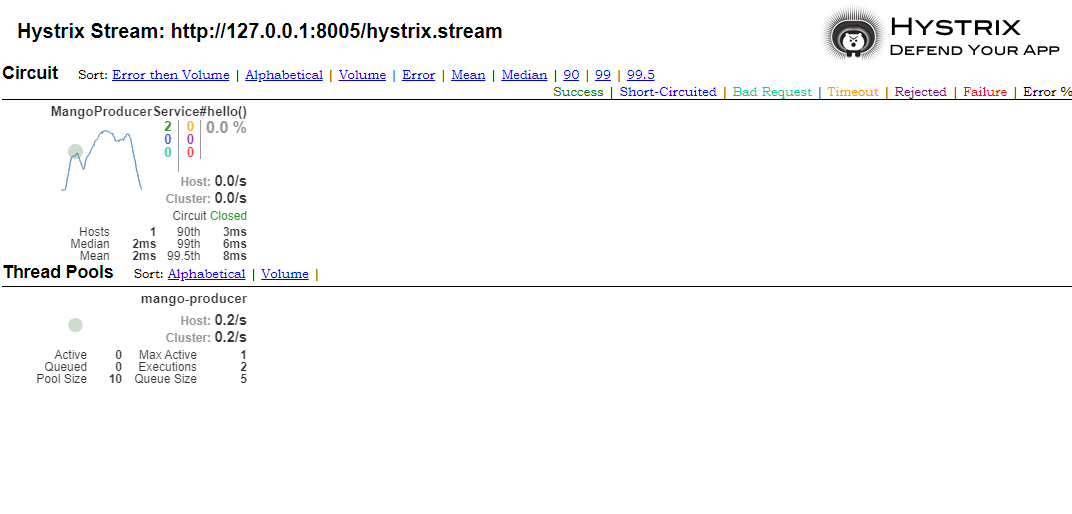
还有一个hystrix面板，地址是<http://127.0.0.1:8501/hystrix>

首先打开<http://127.0.0.1:8005/feign/call>地址，

再在刚刚打开的面板输入http://127.0.0.1:8005/hystrix.stream链接

点击monitor stream，然后不停的刷新<http://127.0.0.1:8005/feign/call>地址，可以看见实时状态





以上就是项目内大部分的SpringCloud组件了，如果实际应用该项目可以直接把他们配置的一些BUG修一修，应该就可以用到生产了。

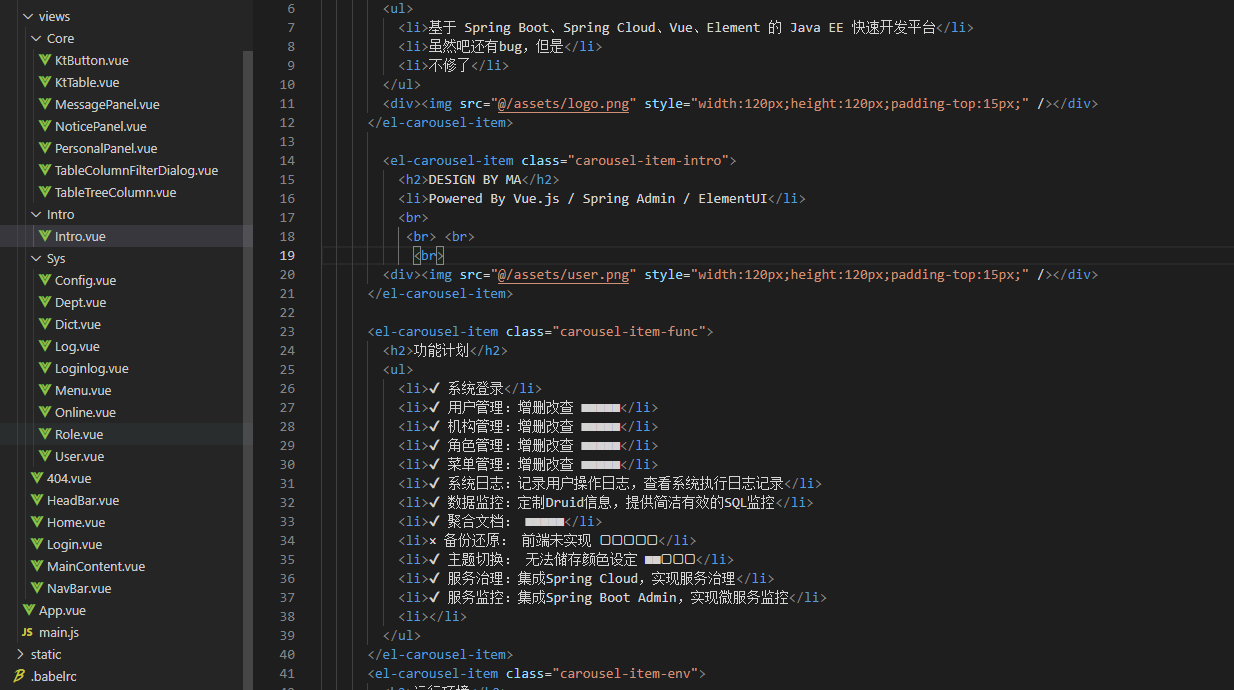


第四章：前端模块

本项目前端使用Vue.js，Vuex等技术实现，主要UI是目前大家都在用的饿了么UI，比较简洁也比较美观大方，唯一美中不足的部分是本人还未透彻的学习Vue，所以部分代码也未能正常运行😊

部分代码截图：

Intro 主页滚动的项目介绍



机构表的代码

<template>

  <div class="page-container">

    <!--工具栏-->

    <div class="toolbar" style="float:left;padding-top:10px;padding-left:15px;">

        <el-form :inline="true" :model="filters" :size="size">

            <el-form-item>

                <el-input v-model="filters.name" placeholder="用户名"></el-input>

            </el-form-item>

            <el-form-item>

                <kt-button icon="fa fa-search" :label="$t('action.search')" perms="sys:log:view" type="primary" @click="findPage(null)"/>

            </el-form-item>

        </el-form>

    </div>

    <!--表格内容栏-->

    <kt-table

    :data="pageResult" :columns="columns" :showOperation="showOperation" @findPage="findPage">

    </kt-table>

  </div>

</template>

表单代码

<script>

引入三大组件，略

export default {

    components: {

        KtTable,

        KtButton

    },

    data() {

        return {

            size: 'small',

            filters: {

                name: ''

            },

            columns: [

                {prop:"id", label:"ID", minWidth:60},

                {prop:"userName", label:"用户名", minWidth:100},

                // {prop:"operation", label:"操作", minWidth:120},

                {prop:"method", label:"方法", minWidth:180},

                {prop:"params", label:"参数", minWidth:220},

                {prop:"ip", label:"IP", minWidth:120},

                {prop:"time", label:"耗时", minWidth:80},

                {prop:"createBy", label:"创建人", minWidth:100},

                {prop:"createTime", label:"创建时间", minWidth:120, formatter:this.dateFormat}

                // {prop:"lastUpdateBy", label:"更新人", minWidth:100},

                // {prop:"lastUpdateTime", label:"更新时间", minWidth:120, formatter:this.dateFormat}

            ],

            pageRequest: { pageNum: 1, pageSize: 10 },

      pageResult: {},

      showOperation:false

        }

    },

    methods: {

        // 获取分页数据

        findPage: function (data) {

            if(data !== null) {

                this.pageRequest = data.pageRequest

            }

            this.pageRequest.params = [{name:'userName', value:this.filters.name}]

            this.$api.log.findPage(this.pageRequest).then((res) => {

                this.pageResult = res.data

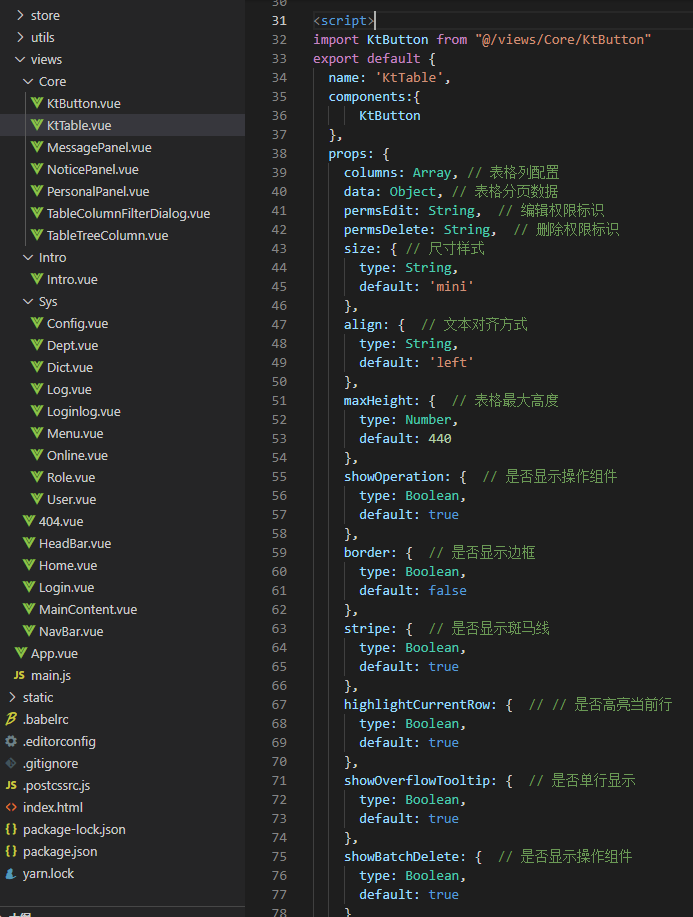
            }).then(data!=null?data.callback:'')

        },

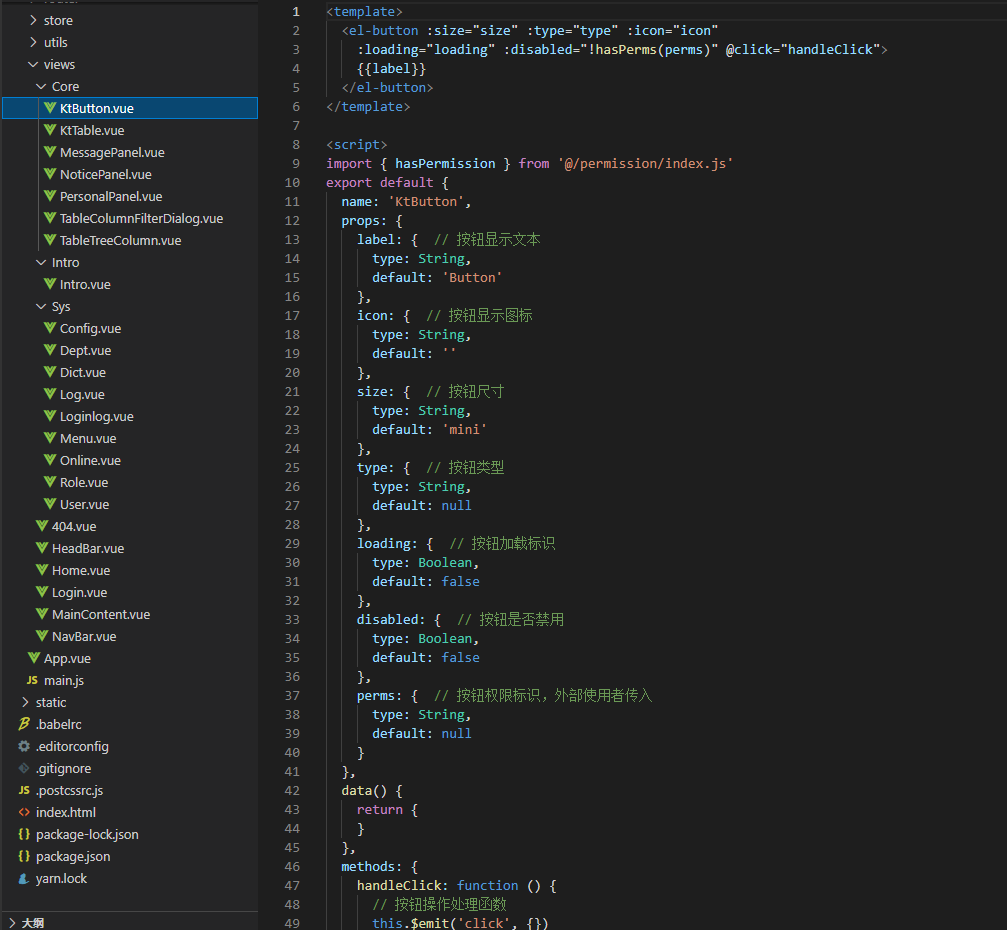
        ……

}</style>

核心组件：表格组件



核心组件：按钮配置



Axios交互模块的请求拦截器

export default function $axios(options) {

  return new Promise((resolve, reject) => {

    const instance = axios.create({

      baseURL: config.baseUrl,

      headers: config.headers,

      timeout: config.timeout,

      withCredentials: config.withCredentials

    })

    // request 请求拦截器

    instance.interceptors.request.use(

      config => {

        let token = Cookies.get('token')

        // 发送请求时携带token

        if (token) {

          config.headers.token = token

        } else {

          // 重定向到登录页面

          router.push('/login')

        }

        return config

      },

      error => {

        // 请求发生错误时

        console.log('request:', error)

        // 判断请求超时

        if (error.code === 'ECONNABORTED' && error.message.indexOf('timeout') !== -1) {

          console.log('timeout请求超时')

        }

        // 需要重定向到错误页面

        const errorInfo = error.response

        console.log(errorInfo)

        if (errorInfo) {

          error = errorInfo.data  // 页面那边catch的时候就能拿到详细的错误信息,看最下边的Promise.reject

          const errorStatus = errorInfo.status; // 404 403 500 ...

          router.push({

            path: `/error/${errorStatus}`

          })

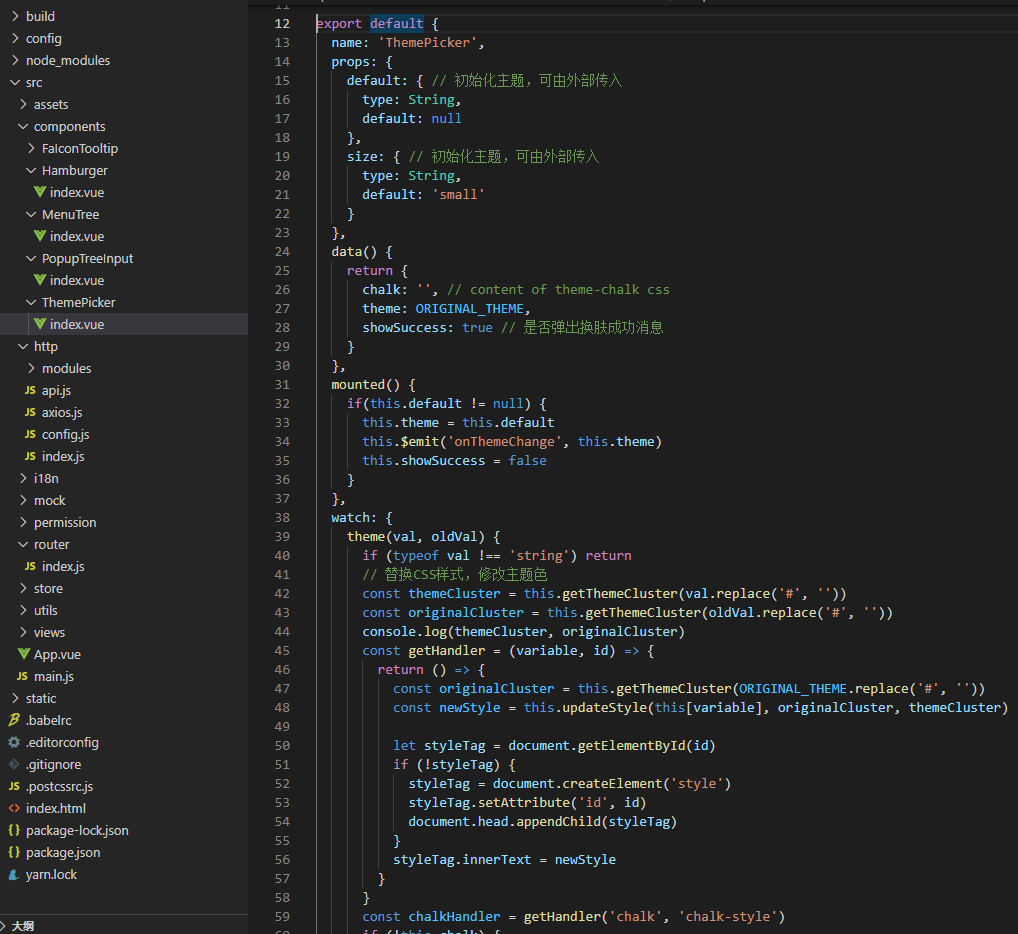
        }

        return Promise.reject(error) // 在调用的那边可以拿到(catch)你想返回的错误信息

      }

    )

主题切换组件





第五章：项目总结

总的来说做这个项目，对于我来说并不完美，期间遇到许多BUG，一直无法修复，比如使用Mybatis生成器进行生成SQLMAP，不知道为什么我的配置总能出错误，后面多方寻求解决方案，几乎把CSDN翻了个遍，同期博客园又在维护，给我解决问题带来不少麻烦。

同时这个项目是我尝试使用Vue和springcloud技术进行制作的第一个，虽然有教材指导，但也并不详细，在出问题的时候教材基本没帮上忙，比如Sass居然在高版本NPM中有兼容问题，尝试了很多次，甚至也重装了系统，也还是不行，有那么几次我都有些相放弃了，但是庆幸的是我没有这么做。我在群里问了杨鸿几个学长，被他一语点破，瞬间解决问题。

我做这个项目，几乎不叫面向对象了，完全就是面向CSDN和面向学长了，哈哈，没有他们的帮助，可能我也无法完成这个项目了。

总的来说，做这个项目，虽然并不特别成功，但是反复失败中我学会了解决更多问题，也逐渐把SpringCloud的概念摸清了大概，希望在我接下来的实训中这些知识能受用无穷吧，我也会继续学习这方面的知识，让它成为我的一个亮点把！加油！

马壹元

2021.06.03