## 数据可视化模块

#### 4.1 设计思路

后端：接到GET请求后，取出数据库表中的所有数据，根据machine\_model字段的值来判断不同的光伏板，每一个值对应一个Machine实体类，Machine实体类包含machineName属性和timeNodeList属性，其中timeNodeList属性表示该板对应的时间点列表，每个时间点列表包含若干TimeNode类，TimeNode类包含电流列表和电压列表，以此包含每个光伏板的所有数据。将数据处理完成后，将每个光伏板对应的Machine类包装成列表，最后通过fastJSON转成JSON格式返回。

前端：在每次打开页面时，向后端发起数据请求，将得到的JSON进行遍历，根据光伏板的数量初始化ECharts表格，再变量每个光伏板的时间点和故障类型，得到每个光伏板的第一张折线图。监听ECharts每个节点的点击事件，点击时显示一个子组件，将时间点的IV数值列表传入子组件，在子组件中展示I-V曲线。

#### 4.2 设计优化

原来的设计思路是网页打开时一次请求所有数据，经过测试当数据点在七百个时一次传输就已经需要传输1.5M的JSON文件，这样很容造成用户使用卡顿；思考之后我们优化了方案，页面打开时仅传递数据时间点、故障类型等信息，而每个时间点的电流电压信息在用户点击具体点时再次请求，如此优化了网络请求，极大得减轻了用户的带宽压力。

因为一次请求数据无法实现动态实时刷新页面数据，而碍于HTTP协议的单工通信，无法实现从服务端主动发送数据到前端，所以我们更换了请求数据的方式，采用支持全双工通信的TCP协议，利用WebSocket来完成通信。每当有数据更新时，服务端主动将数据发送给前端，而前端接受到数据之后重新渲染图表，达到了实时刷新数据的效果。为了达到监控数据库数据更新的目的，我们采用了在后端开线程轮询数据库的方式，每隔一定时间查询一次数据，若数据与之前保存的数据不相同，说明有数据更新，则将其群发给所有连接。

#### 4.3 项目部署

利用阿里云轻量应用服务器，将数据库部署在云端，实现算法部分与数据展示部分共用数据库。后端通过Maven打成jar包之后，在服务器上运行。前端通过npm打包成静态网页文件在服务器上直接访问。

#### 4.4 遇到的问题

1、Axios只允许异步GET请求，导致页面加载之后因为未获取到数据导致表格无法加载。

解决方案：深入了解Vue页面的生命周期，同时将初始化表格的代码放入GET请求的回调函数中。

2、对ECharts折线图节点的监听事件不太了解。

解决方案：阅读ECharts官方文档，逐步调试。

3、I-V曲线因为结点太多导致曲线图部分节点被自动隐藏，无法准确表达曲线趋势。

解决方案：增加缩放工具栏，支持对I-V曲线的任意缩放，实现可以查看到所有的结点。

4、轮询数据库导致资源消耗过大。

解决方案：所有连接共用线程，一个线程轮询，发现新信息群发给所有连接。

#### 4.5 技术栈

* 前端框架使用Vue3
* UI框架使用Element Plus
* 数据可视化使用EChart5
* 后端框架使用Spring boot
* 使用Axios完成HTTP请求
* 使用WebSocket完成TCP请求

#### 4.6 部署方式

* 前端通过npm打包成静态网页文件
* 后端通过maven打成jar包在服务器上运行

#### 4.7 代码链接

<https://github.com/RoseSavona/srtp.git>

#### 4.8 心得

这次项目是我第一次做后端开发，所以花费了一定时间在入门SpringBoot上，因为我们的项目在后端也仅需要读取数据之后处理返回，所以还是进行得比较顺利，在前端方面，虽然我对Vue的掌握也仅处于入门阶段，但因为在其他的项目已经把要踩的坑都踩完了，所以前端开发也比较顺利。这也是我第一次独立部署一个网页，经过这次学习，我深入地感受了一整个项目从立项到部署上线完整的流程。