### （四）软件设计与流程

本系统的软件设计围绕地理信息可视化、区域控制与智能分析等核心功能展开，采用模块化架构，前后端分离设计，以实现高效、可扩展的交互体验与系统响应。下文将从系统架构、关键模块、前后端交互流程以及界面逻辑四个方面进行阐述。

#### 1. 系统架构

本系统采用 **B/S 架构**（Browser/Server），整体设计分为三层：

**前端显示层**：基于 Vue.js 框架开发，配合 Element-UI 实现界面组件构建，并使用 Leaflet 插件进行地图渲染；

**后端服务层**：采用 Node.js + Axios 提供 REST API 接口，处理控制命令、数据查询与用户管理；

**数据支持层**：整合 sqlite ，用于结构化信息（如用户、事件、控制区域）与非结构化数据（如地图标注、历史轨迹）存储。

#### 2. 核心功能模块设计

系统主要功能模块如下：

**地图控制模块**：用户可通过左侧导航栏“地图控制”切换底图类型（如卫星图、矢量图），并可进行缩放、拖动等基本交互。

**预警区域绘制模块**：支持用户在地图上手动绘制矩形或多边形区域，用于定义预警或监控区域，信息同步提交后端。

**控制区坐标配置模块**：允许精确设定或读取指定区域的经纬度坐标点，完成控制区标定。

**反击功能管理模块**：可视化展示反制链路状态，包括“后台地图”、“干扰模块”、“主节点”连接情况与检测统计信息。

**AI 学习与事件展示模块**：通过数据驱动方式学习历史攻击模式，实现事件分类展示与特征分析。

#### 3. 前后端交互流程

系统交互通过 RESTful API 完成以下主要流程：

**用户登录与认证**：前端发送登录请求，后端验证身份并返回会话token；

**地图加载与图层请求**：前端初始化加载地图后，从后端请求相关图层信息（如预警区、多边形边界）；

**控制操作同步**：如添加控制区、更新设备状态等操作，将以JSON形式通过 POST 请求提交到后端；

**数据轮询与状态刷新**：界面右侧状态面板定时从后端获取最新设备连接状态、检测次数等信息；

**事件记录与AI学习**：每次检测记录与反击响应将发送至AI模块进行样本积累，前端支持分页加载事件。

#### 4. 前端流程设计

前端界面设计以用户体验为中心，采用响应式布局，主要逻辑流程如下：

**初始化加载阶段**：

创建Leaflet实例，设置视图中心与缩放级别；

加载初始图层与系统状态；

**用户交互阶段**：

左侧导航栏控制系统模式（如地图绘制、AI学习等）；

地图点击或绘制事件触发弹窗或浮动层，进行坐标信息录入；

控制按钮（如上传控制区、查看详情）绑定事件回调，调用后端接口；

**状态反馈阶段**：

页面右侧以模块卡片形式展示设备状态与检测数据，绿色/红色图标指示连接情况；

未检测到数据时显示“No Data”，检测次数动态刷新。

#### 5. 关键技术点

地图组件采用 **Leaflet.js**，通过引入瓦片图与自定义图层实现对地理信息的高效加载；

多边形绘制与区域判断通过状态转移完成，通过Pinia进行状态管理；

状态监测模块通过WebSocket或轮询方式获取最新设备/检测信息；

系统支持国际化扩展（语言切换按钮位于右上角），便于部署于不同语言环境下。