

**“互联网+”综合研究实训设计报告**

( 2025--2026学年第一学期)

题 目：《安燃云——城/园区燃气漏损监测与应急平台》

学 院：计算机科学与技术学院（人工智能学院）

专 业： 计算机科学与技术

班 级： 2班

学 号： 2023337621278

学生姓名： 李星阳

指导教师： “互联网+”综合研究实训教学组

日期： 2025年 12 月 28 日

目 录

[一、 设计目的 5](#_Toc1726)

[二、 团队成员及分工 7](#_Toc1978)

[1. 团队成员： 7](#_Toc2267)

[2. 具体分工： 7](#_Toc4071)

[三、 设计思路 8](#_Toc22378)

[1. 操作系统： 8](#_Toc14113)

[2. 数据库工具： 9](#_Toc20651)

[3. 集成开发环境与工具： 9](#_Toc1966)

[4. 用户界面设计应用框架： 10](#_Toc3763)

[4.1现代化前端技术栈： 10](#_Toc4268)

[4.2轻量级WebGIS解决方案： 10](#_Toc13333)

[4.3初期MVP策略：响应式Web表单 10](#_Toc17035)

[4.4中期演进路线：微信小程序 10](#_Toc3850)

[5. 支持移动端应用： 10](#_Toc23579)

[6. 技术路线图： 11](#_Toc20142)

[四、 需求分析 11](#_Toc21841)

[1. 需求分析描述： 11](#_Toc9032)

[2. 用例图： 13](#_Toc23792)

[3.现阶段其他工作进展情况： 15](#_Toc10685)

[五、 设计方案 15](#_Toc26807)

[1.系统业务流程 15](#_Toc17301)

[1.1 整体业务流程说明（跨 C 端 + B 端） 15](#_Toc5062)

[1.2系统业务流程图 18](#_Toc18499)

[2.系统功能结构（模块分层） 19](#_Toc9553)

[2.1 功能结构介绍 19](#_Toc8585)

[（一）表现层（UI 层 / 客户端） 19](#_Toc10724)

[（二）业务应用层（后端服务与领域模块） 20](#_Toc13368)

[（三）支撑与数据层 21](#_Toc7103)

[2.2系统功能结构图 21](#_Toc16274)

[2.3类图（实体类图） 21](#_Toc15122)

[2.4数据库E-R图 23](#_Toc28952)

[六、开发成果展示 24](#_Toc32037)

[1. 系统启动与访问方式 24](#_Toc6038)

[2. 登录与权限验证 25](#_Toc29100)

[3. 态势总览 25](#_Toc30622)

[4. 告警管理 26](#_Toc14299)

[5. 工单闭环处置 27](#_Toc5337)

[6. 设备管理与数据查看 28](#_Toc10484)

[7. 异味快报（居民上报） 29](#_Toc7887)

[8. 管网仿真与故障注入 31](#_Toc19557)

[9. 周报导出（PDF） 32](#_Toc25888)

[10. 移动端成果展示（居民端 + 工人端） 33](#_Toc2476)

[10.1 居民端（公众参与） 33](#_Toc5691)

[10.2 工人端（运维处置） 35](#_Toc28325)

[11. 关键代码展示 38](#_Toc25879)

[七、总结 42](#_Toc21733)

[1. 项目完成情况 42](#_Toc21888)

[2. 主要亮点与创新点 42](#_Toc18255)

[3. 不足与改进方向 43](#_Toc11248)

[4. 总结 43](#_Toc20754)

**图目录**

[图1-1 总体思路图 12](#_Toc19098)

[图1-2 期望图 14](#_Toc16626)

[图3-1 技术工具图 15](#_Toc31120)

[图3-2 数据库工具 16](#_Toc26738)

[图3-3 集成开发环境与工具 16](#_Toc20109)

[图3-4 技术主流程 18](#_Toc19753)

[图3-5 技术路线图 18](#_Toc8953)

[图4-1 安燃云平台需求用例图 20](#_Toc16203)

[图4-2 安燃云平台业务流程图 21](#_Toc6401)

[图4-3 安燃云平台ER图 21](#_Toc10252)

[图4-4 安燃云平台模块图 22](#_Toc28122)

[图5-1 系统业务流程图 25](#_Toc11602)

[图5-2 系统功能结构图 28](#_Toc16174)

[图5-3 web端类图 29](#_Toc20480)

[图5-4 移动端类图 29](#_Toc25805)

[图6-1 安燃云Pro登录界面 32](#_Toc16712)

[图6-2 态势总览页面（地图+设备点位+管网） 32](#_Toc20462)

[图6-3 设备详情弹窗（点击点位查看实时指标） 33](#_Toc5230)

[图6-4 告警列表页面 33](#_Toc30817)

[图6-5 AI分析生成告警原因与影响 34](#_Toc30357)

[图6-6 工单列表页面 34](#_Toc25043)

[图6-7 工单状态更新 34](#_Toc23658)

[图6-8 设备管理页面（设备列表） 35](#_Toc24283)

[图6-9 新增设备信息表单 35](#_Toc10139)

[图6-10 设备时序数据趋势展示（MiniChart） 36](#_Toc6596)

[图6-11 异味快报上报页面 37](#_Toc2663)

[图6-12 异味快报提交结果反馈 38](#_Toc14690)

[图6-13 管网仿真可视化界面 39](#_Toc23008)

[图6-14 故障注入后的状态变化展示 39](#_Toc28661)

[图6-15 报表导出页面（生成与下载周报） 40](#_Toc13295)

[图6-16 导出的周报PDF示例 40](#_Toc29009)

[图6-17 移动端登录界面（居民身份入口） 41](#_Toc16254)

[图6-18 居民端首页（便民服务与运行概况） 41](#_Toc18073)

[图6-19 居民端异味上报表单（地址与异常选择） 41](#_Toc17682)

[图6-20 居民端异味上报表单（坐标获取与示例填写） 41](#_Toc5180)

[图6-21 居民端异味上报提交（联系方式与立即上报） 42](#_Toc29996)

[图6-22 附近管网状态查询（监测点/管段概况） 42](#_Toc7296)

[图6-23 上报记录回显（最近一次上报详情） 42](#_Toc26567)

[图6-24 移动端登录界面（抢修/巡检身份入口） 43](#_Toc8984)

[图6-25 工人工作台首页（待处理工单与快捷入口） 43](#_Toc14054)

[图6-26 工作台告警提醒与推荐巡检路线 43](#_Toc4781)

[图6-27 抢修任务列表（今日工单与状态流转） 43](#_Toc16499)

[图6-28 巡检与管网（路线生成与风险段提示） 44](#_Toc8130)

[图6-29 今日巡检计划（打卡式勾选执行） 44](#_Toc1058)

[图6-30 告警中心（搜索与多维筛选） 44](#_Toc23436)

[图6-31 告警中心列表（严重告警展示） 44](#_Toc25947)

[图6-32 工作台工单概览与设备状态统计 45](#_Toc32441)

**《安燃云——城/园区燃气漏损监测与应急平台》实训设计报告**

# 设计目的

本项目旨在构建面向城市与园区燃气运营单位、物业管理方与居民用户的“安燃云”一体化平台，在存量管网与低成本边缘传感器条件下，形成“更快发现、更准定位、更规范闭环”的用气安全能力。平台遵循“互联网+公共安全治理”的总体思路，以数据贯通为基础、以物理约束为核心、以应急流程为骨架，贯穿监测—识别—告警—处置—复测—归档全流程。项目在技术上坚持“仿真先行、实网替换”的工程路线：以可视化管网模拟器和仿真传感器快速搭建可运行的端到端闭环，再在不改变接口与流程的前提下替换为真实设备与网关接入，确保早期可验证性与后期可演进性。

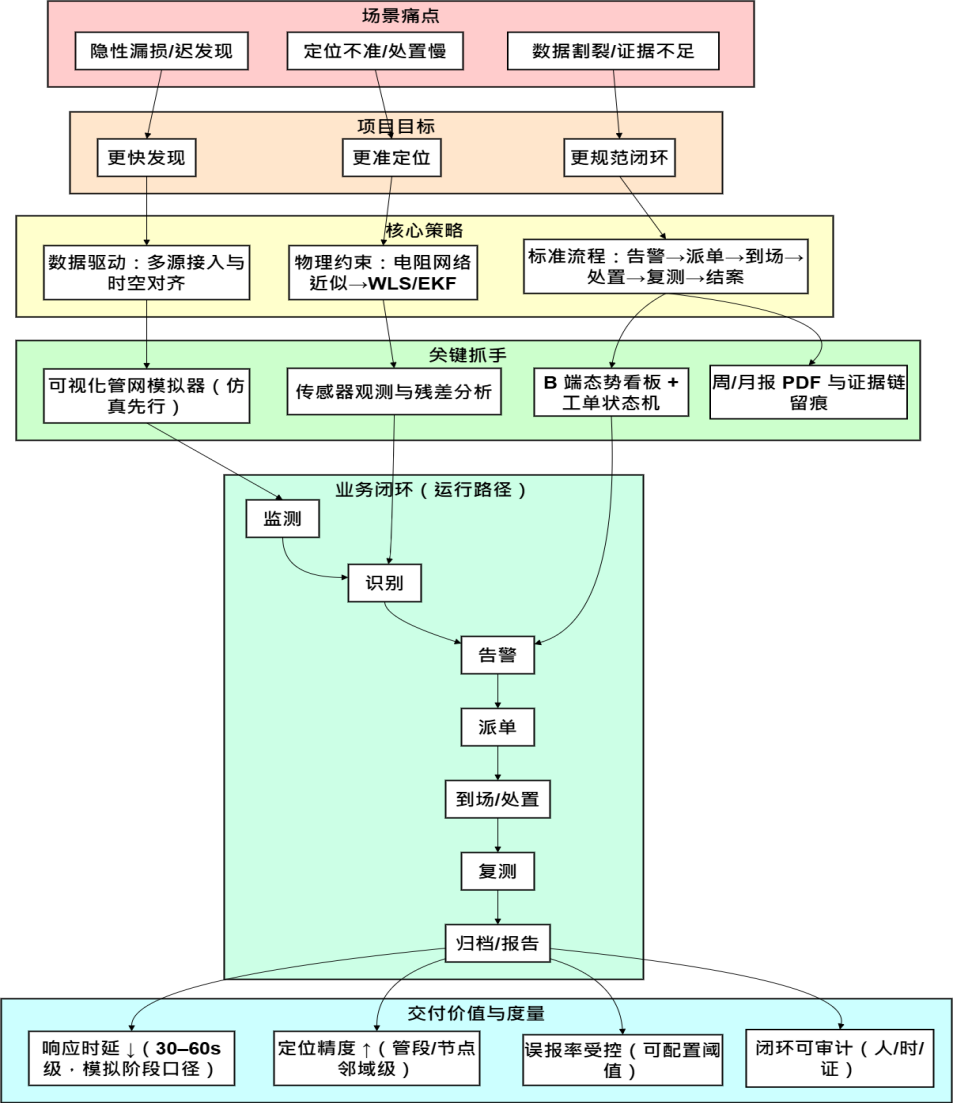


图1-1 总体思路图

从城市治理视角，设计目的首先在于打通多源异构数据的壁垒，包括 SCADA、压力/流量传感器、巡检记录与摄像头抓拍等，使之在统一的数据底座与时序存储中实现时空对齐与质量评估。其次是在算法层面对管网进行近似建模：以电阻网络近似求解节点压力与边流向，结合传感节点观测与残差分析形成可解释的异常检出，并为后续的加权最小二乘（WLS）与扩展卡尔曼滤波（EKF）等状态估计方法预留接口与数据口径。再次是在业务层将“告警—派单—到场—处置—复测—结案”固化为标准化的、可审计的流程，确保每一环节有证据链、有责任人、有时间戳，既满足日常运行，也能支撑演练与复盘。

从使用者体验与价值交付看，B 端侧重态势看板与工单闭环，要求调度人员能够直观看到压力分布、流向走势与告警热区，并一键从告警生成工单、跟踪进度、导出报表；C 端以“安燃云·家”实现居民参与的最低门槛，支持异味快报、用量趋势与安全体检等轻量功能，让“人人可上报、处处可响应”。平台同时考虑对外联动的可扩展性，预留阀控、短信/电话网关与门禁等一体化接口，在合规前提下实现“发现即联动、处置可追溯”。

在工程可实施性方面，项目将“可运行的最小系统”作为第一目标：后端采用 FastAPI + SQLAlchemy 构建轻量服务，启用 SQLite 的 WAL 模式降低并发写入冲突；前端采用 React + Vite + Leaflet 实现 WebGIS、工单流转与报表下载；内置可视化管网模拟器以秒级频率推送节点压力与边流量，配合 WebSocket 实时刷新告警列表。

从“互联网+”的落地价值看，项目并非单点功能的叠加，而是以平台化方式把“数据接入—治理—建模—告警—处置—报告”串成一条可复用、可迁移的链路：一方面服务于园区/校园等典型场景的安全运营，另一方面形成可推广的工程范式，为后续与政企现有系统的对接（如城市运行管理中心、应急平台）、与更多算法插件（视觉仪表识别、因果链推理与风险画像）的融合打下标准化基础。最终目标是以较低的改造成本，为更多场景提供“可上线、可审计、可演进”的燃气安全治理能力，并在长期运维中沉淀形成可度量的风险下降曲线与处置效率提升曲线。

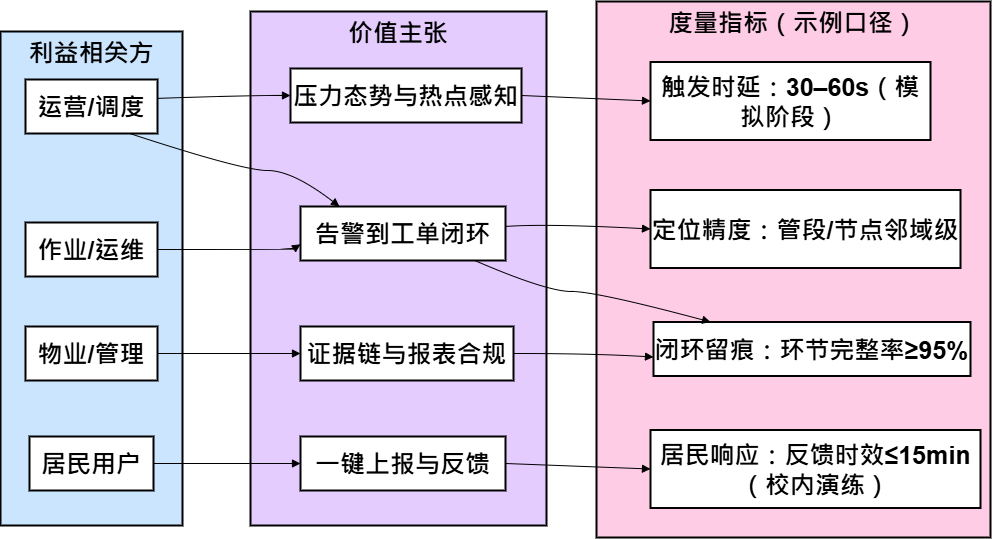


图1-2 期望图

# 团队成员及分工

## 团队成员：

组长：李星阳  
组员：杨修，黄红洲

## 具体分工：

**组长：李星阳**

工作：在算法侧，完成管网电阻网络近似求解，输出节点压力与边流量的时序流；实现基于观测残差与滑动窗口的多级告警触发，并给出可解释字段（触发阈值、残差大小、传感器ID、建议定位区段）。预留 WLS/EKF 接口与参数模板，形成从“近似线性→WLS/EKF”的平滑升级路径。在前端侧，负责态势看板上的**压力/流向叠加层**与**时序曲线联动**等关键组件开发，对接 WebSocket 事件流，保证“地图—列表—曲线”三向联动的一致性与性能。牵头接口评审与技术风险处置（如并发写入、时序乱序、告警抖动）。

**组员1：杨修**

工作：在算法侧，完成管网电阻网络近似求解，输出节点压力与边流量的时序流；实现基于观测残差与滑动窗口的多级告警触发，并给出可解释字段（触发阈值、残差大小、传感器ID、建议定位区段）。预留 WLS/EKF 接口与参数模板，形成从“近似线性→WLS/EKF”的平滑升级路径。在前端侧，负责态势看板上的**压力/流向叠加层**与**时序曲线联动**等关键组件开发，对接 WebSocket 事件流，保证“地图—列表—曲线”三向联动的一致性与性能。牵头接口评审与技术风险处置（如并发写入、时序乱序、告警抖动）。

**组员3：黄红洲**

工作：基于 FastAPI + SQLAlchemy 构建微服务，启用 SQLite 的 WAL 模式降低“database is locked”概率；设计并实现**告警→工单**状态机（待派/在场/处置/复测/结案）与审计字段；提供统一分页/过滤规范与版本化路由 /api/v1/\*；集成 ReportLab 的周/月报导出；实现 CORS 白名单、JWT/角色权限与 WebSocket 推送；编写 Windows 一键启动脚本与 .env.example，保障“下载即运行”。

# 设计思路

系统采用“可运行最小闭环 + 可演进技术路线”的思路先搭后优。后端以 FastAPI + SQLAlchemy 构建轻量服务，SQLite WAL 模式保障单机并发写入稳定性；内置电阻网络近似的管网模拟器，以节点压力与边流量的线性方程组快速求解，周期推送到前端，形成流向可视化与异常触发；前端以 React + Vite + Tailwind 落地，WebGIS 选用 Leaflet，态势/工单/设备与时序曲线统一在 B 端集成；居民侧先以 Web 表单模拟“异味快报”，后续再扩展到小程序；报表采用 ReportLab 生成可下载 PDF。部署上本地一键脚本优先，云端可平滑迁移至容器编排。随着数据与设备完善，仿真将逐步被真实接入替代，状态估计从近似线性走向 WLS/EKF 以及分区解耦，异常检测从阈值/残差扩展到时序模型，并在保障安全与隐私合规的前提下对接阀控、短信/电话网关与摄像头联动。

## 操作系统：

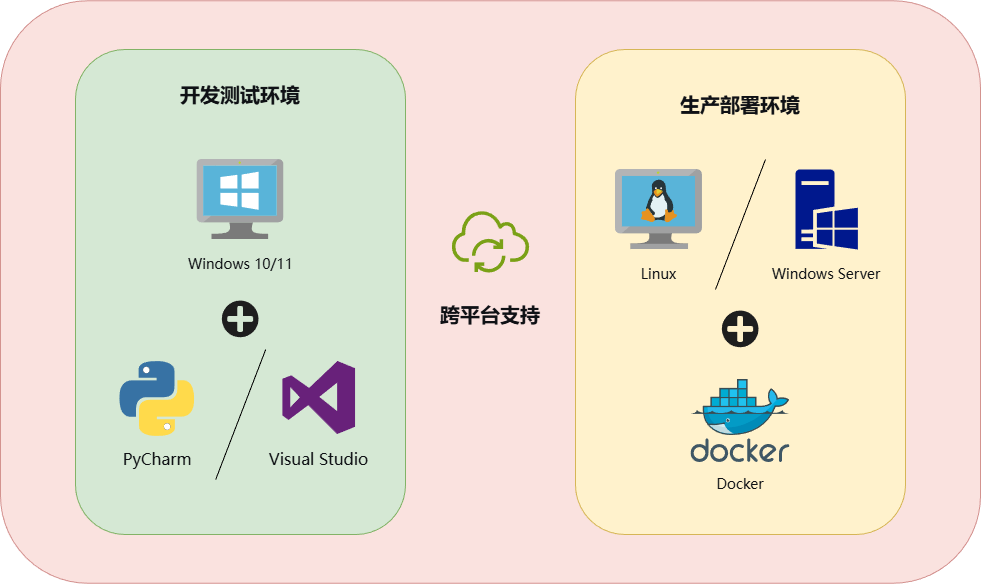


图3-1 技术工具图

**开发与测试环境：** 以 **Windows 10/11** 作为主要开发和测试平台，确保系统在该系列操作系统上的兼容性与稳定性。

**生产环境：** 系统设计为跨平台，可无缝部署于主流Linux发行版（如 Ubuntu、CentOS）以及Windows Server环境，为后续云端容器化部署奠定基础。

## 数据库工具：

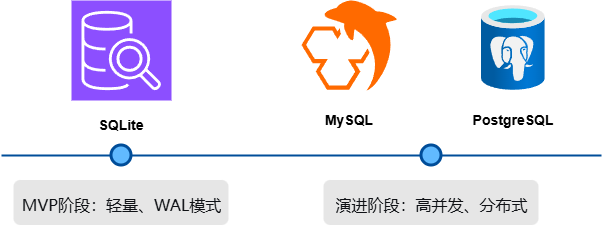
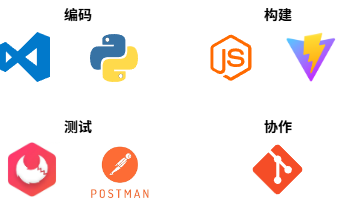


图3-2 数据库工具

**核心数据库：** 项目初期采用 **SQLite** 作为关系型数据库。通过启用 **WAL（预写式日志）模式**，显著提升读写并发性能与数据崩溃恢复能力，完美契合MVP阶段对轻量、高效和单文件部署的需求。

**数据迁移与演进：** 使用 SQLAlchemy ORM 进行数据建模与操作，抽象了底层数据库差异。随着业务规模扩大，可通过更换数据库连接驱动，平滑迁移至 **PostgreSQL** 或 **MySQL** 等更强大的数据库系统，而无需重写主要业务代码。

## 集成开发环境与工具：



**图3-3 集成开发环境与工具**

**后端开发：** 使用 **VS Code** 或 **PyCharm** 作为主要的集成开发环境，二者均提供对Python和FastAPI框架的出色支持。

**前端开发：** 使用 **Node.js** 作为运行时，**Vite** 作为前端构建工具，以实现极速的热重载和高效的打包。

**接口测试与协作：** 使用 **Postman** 或 **Apifox** 进行API接口的调试、测试与自动化，并用于生成和维护API文档，促进前后端协作。

## 用户界面设计应用框架：

B端平台作为系统的管理中枢，其设计核心在于高效、清晰与强大的数据整合与可视化能力。

### 4.1现代化前端技术栈：

我们采用以 React 为核心的前沿技术栈。通过函数式组件和 Hooks 模式构建声明式用户界面，使代码逻辑更清晰、更易于复用与测试。构建工具选用 Vite，凭借其极速的热重载（HMR）和高效的打包能力，为开发者提供流畅的开发体验。样式方案上，采用 Tailwind CSS 这种原子化、Utility-First 的CSS框架，它允许我们通过组合低效用类来快速构建定制化且保持一致性的用户界面，极大地提升了UI开发的灵活性与效率。

### 4.2轻量级WebGIS解决方案：

在地理信息可视化方面，我们选用轻量级开源库 Leaflet。它将作为管网系统的“数字地图”，通过集成各类插件，我们能够实现以下核心功能：

* **管网拓扑可视化：** 将抽象的管网数据转化为直观的地图图层。
* **设备资产标注：** 对阀门、传感器、水泵等设备进行精准定位与状态标识。
* **实时数据驱动：**通过自定义矢量图层或动画插件，动态展示水流的实时方向与压力分布，将数据变化转化为直观的视觉态势。
* **丰富的交互操作**： 支持地图的缩放、平移、点位查询、框选以及基于地图的工单创建与派发。
* C端应用面向广大居民用户，其策略核心在于快速验证、低成本迭代与最大化用户触达。

### 4.3初期MVP策略：响应式Web表单

在项目初期，为以最小成本快速验证“居民上报”这一核心业务逻辑的可行性，我们优先采用响应式Web表单进行模拟。该方案可直接复用B端平台的React与Tailwind技术栈，无需开辟新的技术战线，能够迅速落地。一个适配手机和电脑的H5页面，足以收集用户反馈、跑通上报流程，并为后续开发积累真实数据需求。

### 4.4中期演进路线：微信小程序

在业务模式得到验证后，为触达更广泛的用户群体并提升用户体验，我们将全面演进至微信小程序。小程序依托微信生态，具备无需下载、即用即走的天然优势，并通过微信授权登录、消息模板等功能，极大地优化了用户注册与通知流程。此举旨在利用微信的庞大流量与便捷生态，构建一个服务精准、体验流畅的居民端入口，完成从“功能验证”到“服务深化”的关键转变。

## 支持移动端应用：

在移动互联时代，本系统的设计高度重视移动场景下的用户体验，针对不同用户群体的核心需求，制定了清晰而务实的移动端支持方案。

对于B端管理平台而言，移动化的核心价值在于为管理人员提供“无处不在的态势感知”与“快速应急响应”能力。平台采用先进的响应式Web设计，其界面布局能够智能地适应从桌面大屏到平板、手机等多种设备屏幕。这意味着，无论是在办公室的电脑前，还是在巡检现场的平板上，或是在应急指挥途中使用手机，用户都能获得布局合理、操作便捷的连贯性体验。当然，我们也清醒地认识到移动设备在处理复杂任务时的局限性。因此，在移动端的功能设计上，我们采取了“聚焦核心”的策略，优先保障实时数据监控、报警消息的即时推送与确认、以及工单的查阅与简短批复等高频、轻量级操作。这种设计确保了管理人员在移动状态下能高效完成最紧迫的任务，而将复杂的配置与深度分析工作留给桌面环境，实现了效率与功能完整性的最佳平衡。

对于C端居民应用，移动端则从“可选项”升级为“主战场”。我们的策略遵循一条渐进式演进路径。在项目初期，我们将首先推出响应式H5页面作为服务载体。这一选择确保了服务的普惠性——任何智能手机用户无需下载安装，通过浏览器即可即时访问报事、查询等基础功能，极大地降低了用户使用门槛，有利于快速验证核心业务模式并积累早期用户。随着服务的深化和用户习惯的养成，我们将顺势推出微信小程序作为下一代移动端主力。小程序不仅能够提供近乎原生应用的流畅交互体验，更重要的是，它能无缝融入微信社交生态，借助微信的授权登录、消息模板和社交分享能力，实现用户的便捷获取与服务的精准触达，最终完成从“功能可用”到“体验优秀”的质的飞跃。

## 技术路线图：

在总体方向上，路线分为“当前可运行版本（已实现）”与“目标生产化版本（迭代中）”。当前版本以内置仿真和本地时序为主，目标版本在数据侧引入 Flink+Kafka 的流式接入与规则引擎（Drools+LLM 自然语言规则），在检测侧融合 LSTM/XGBoost 与 BGE 文本告警抽取，在处置侧以 LLM + 图计算输出因果链，并形成周/月报自动生成。

技术主流程（目标形态）：



图3-4 技术主流程

分层架构（当前已实现的最小闭环）：

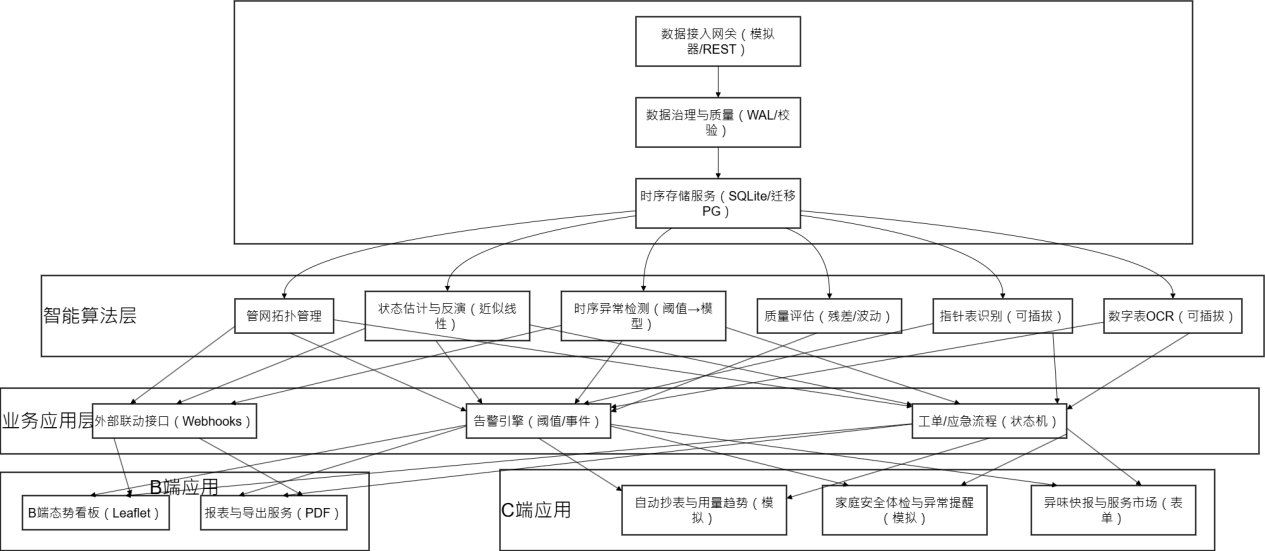


图3-5 技术路线图

# 需求分析

## 需求分析描述：

在典型园区/校园燃气场景中，气源经主干/支线输送至各楼栋终端。运营方需要实时掌握压力分布和流向变化，并尽早识别异常；物业需要查看事件闭环与可追溯工单；居民需要能在闻到异味或发现异常时迅速上报并获得反馈。由于项目初期缺乏完整设备与实时数据，系统通过仿真器 + 可替代流程保证“能用、好用”，同时保留与真实网关、SCADA和物联网设备的扩展接口，以支持从模拟流量逐步切换至真实数据接入。

因此，本系统需求核心围绕三条主线展开：

1）数据侧：先以仿真器推送压力/流量，后期替换为真实传感器、摄像头与表计数据。

2）算法侧：先以电阻网络与阈值触发支撑异常检测，再逐步演进为 WLS/EKF 状态估计。

3）业务侧：形成“监测—告警—派单—到场—处置—复测—结案”的可运行闭环，并同时支持C端居民上报事件。

* 1. **功能性需求**

系统围绕“监测—识别—告警—处置—复核—留痕”建立最小可运行业务闭环（MVP），需求如下：

（1）数据接入与治理

初期支持仿真器定时推送节点压力与管段流量。

中期支持网关/REST/文件导入与标准工业协议（OPC UA、Modbus TCP、MQTT）接入。

数据治理需完成时间对齐、异常过滤、缺失补偿与质量评分，用于保障后续分析输入可靠性。

（2）状态估计与异常识别

初期采用电阻网络近似模型，对压力/流量分布进行快速求解。

支持基于传感节点偏差的简单异常触发。

随数据积累逐步引入 WLS/EKF 状态估计，对模型偏差进行动态修正。

（3）智能告警与工单流程

异常触发后系统自动生成告警并进入状态机管理。

工单流程包含“待派—在场—处置—复测—结案”五个阶段，每个阶段自动记录时间戳与操作人。

B 端可在态势看板与工单模块中查看进度和现场回传数据。

（4）居民异味快报与C端上报

居民可通过网页（H5）提交异味上报、附带图片/描述。

上报事件直接进入 B 端告警流程，与运维事件统一管理。

后期演进至微信小程序，实现更低门槛触达与消息通知。

（5）可视化态势与报表导出

基于 Leaflet 的轻量级 GIS 组件展示实时流向、压力分布与事件位置。

报表功能可按周/月自动生成 PDF，包括告警统计、工单完成率与关键指标。

* 1. **非功能性需求**

（1）性能与实时性

监测数据刷新频率满足“秒级”感知。

告警生成时间控制在 1–2 秒内（仿真环境下）。

工单与消息流转保持低延迟，满足应急响应要求。

（2）系统稳定性与可维护性

SQLite 启用 WAL 模式提升并发写入稳定性。

API 必须保持幂等性，为后期迁移至 PostgreSQL 做准备。

后端日志记录关键操作轨迹，用于故障排查与审计。

（3）安全性与权限控制

使用 JWT 鉴权，区分管理员、调度员、操作员等角色。

后端启用 CORS 白名单策略；用户数据与图像进行必要的脱敏处理。

敏感操作（结案、复测确认）需进行角色校验。

（4）前端适配性

B 端采用响应式设计，适配桌面、平板与手机查看。

C 端 H5 页面适配不同分辨率，为后期小程序迁移打基础。

（5）可扩展性与兼容性

数据模型与 API 保持可拓展结构，不因后期设备接入或算法升级而产生破坏性修改。

系统可无缝迁移至 Docker/K8s 环境。

## 用例图：

1. **安燃云平台需求用例图**

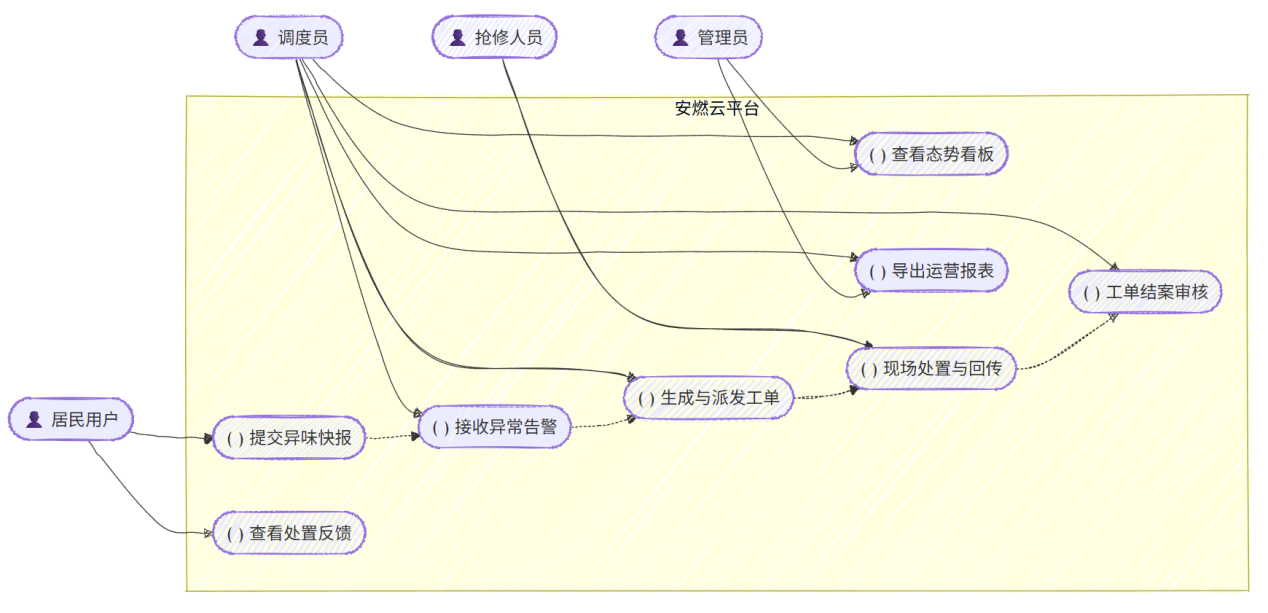
****

图4-1 安燃云平台需求用例图

1. **安燃云平台业务流程图**

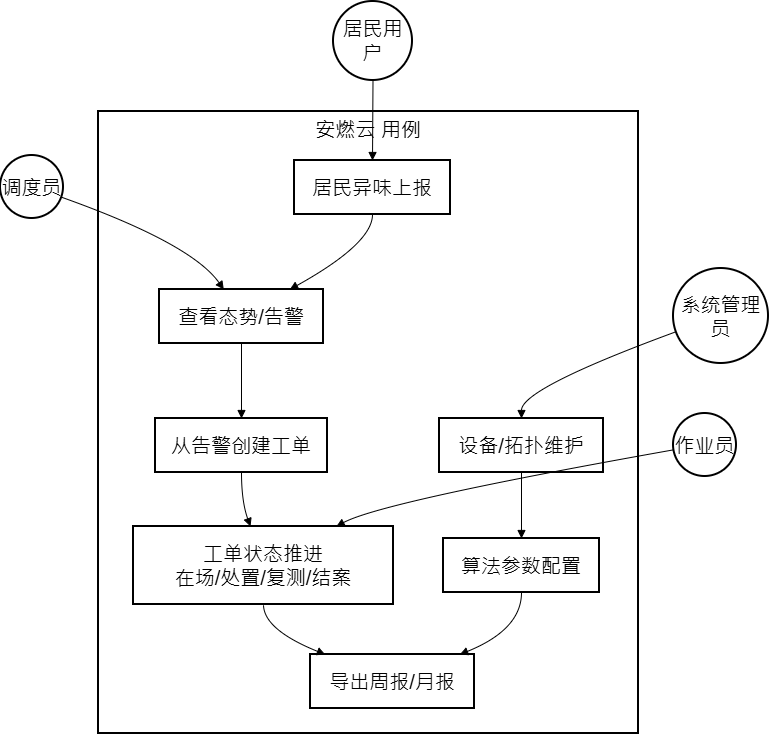


图4-2 安燃云平台业务流程图

1. **安燃云平台ER图**

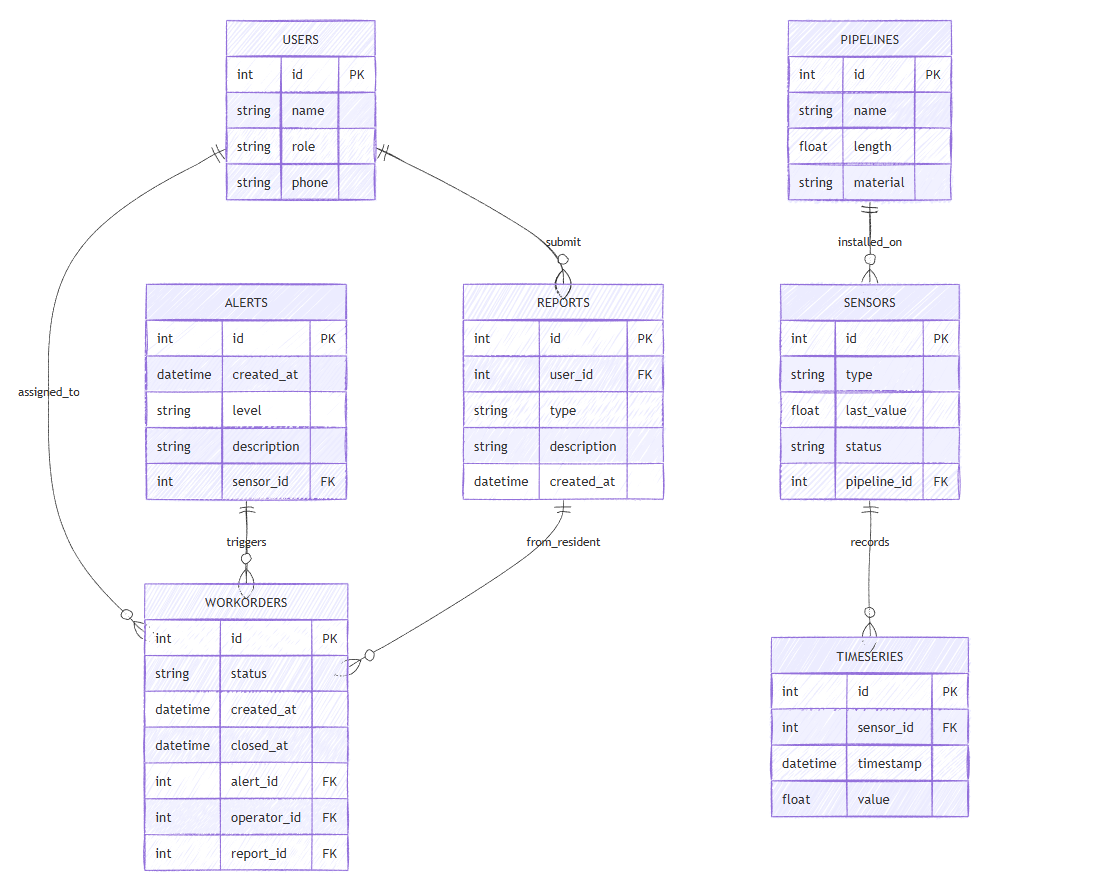
****

图4-3 安燃云平台ER图

1. **安燃云平台模块图**

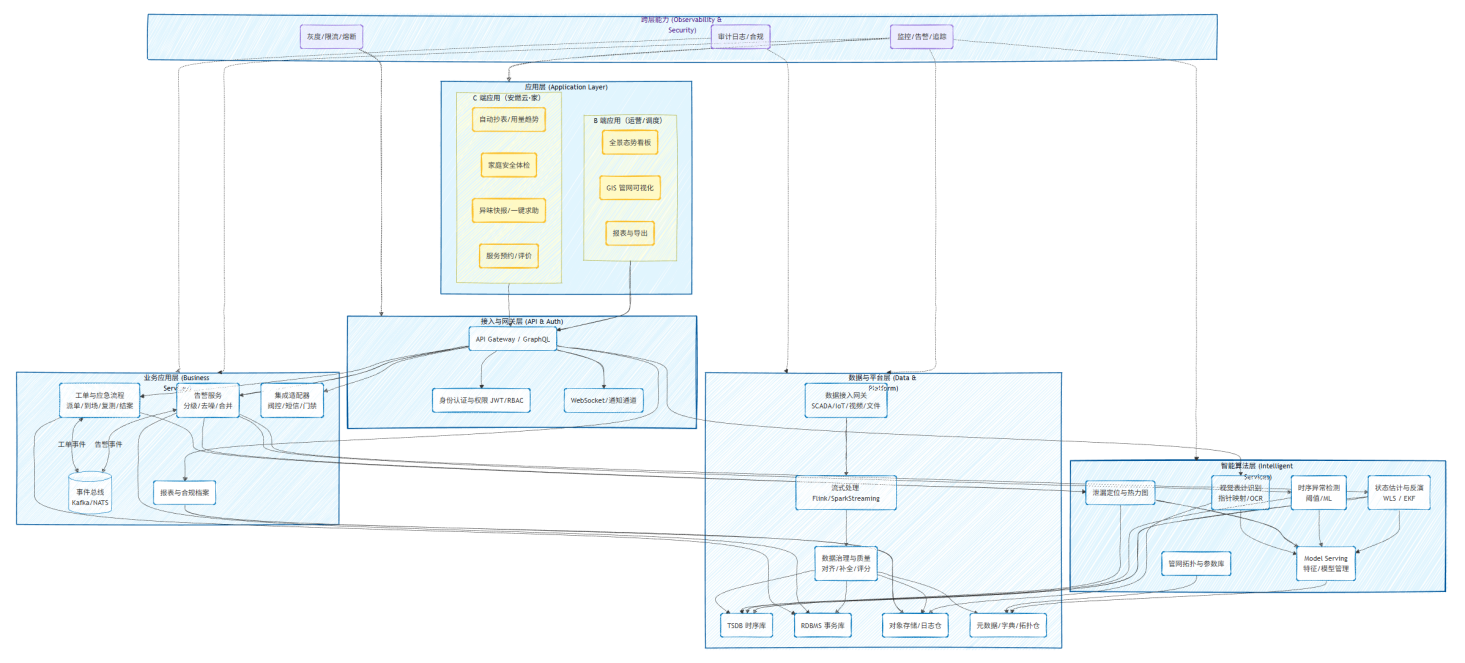
****

图4-4 安燃云平台模块图

## 3.现阶段其他工作进展情况：

当前已完成以下部分，为项目后续迭代提供基础：

（1）后端最小可运行系统（MVP）已落地

FastAPI + SQLite + SQLAlchemy 全流程打通

设备、监测数据、告警、工单、居民上报等核心表结构已实现

基于仿真的压力/流量推送机制可正常运行

（2）前端 B 端平台已实现基础功能

GIS 地图展示与设备点位初步可视化

告警列表、工单列表、事件时间轴等关键模块已可操作

工单派发与状态变更流程已跑通

（3）C 端异味快报已跑通最小流程

居民上报页面（H5）已开发

上报内容自动写入后端并触发 B 端工作流

（4）可复用的架构体系已搭建

数据接入/算法/业务/可视化等模块边界已明确

具备从模拟 → 真实设备接入的扩展空间

前后端文档与接口调试环境完备（Apifox/Postman）

# 设计方案

## 1.系统业务流程

### 1.1 整体业务流程说明（跨 C 端 + B 端）

系统的主业务链条是：

1.C 端居民侧

* 居民通过 App 登录/注册（走 auth + residents）。
* 打开「首页」看到自己家庭用气概览（用量趋势 + 最近告警 + 安全体检结果）。
* 如果闻到异味或发现异常，可以发起「异味快报 / 安全隐患上报」，写入告警表或居民上报表。

2.监测与仿真引擎

* 后端的 net\_sim 持续模拟/采集每个 Device 的时序数据（TimePoint：压力/流量/温度）。
* net / 告警规则引擎定期扫描数据：超阈值、波动异常、状态异常时生成告警 Alert。
* 告警会同时推送到：
  + B 端 Web 的「态势总览」「告警列表」；
  + C 端居民 App 的「异常提醒 / 消息中心」。

3.B 端调度中心

* 调度员在 Web 端 Dashboard 上看到：
  + 园区地图（Leaflet）：管网线路 + 设备点位 + 告警点高亮。
  + 告警列表：按等级/时间/区域筛选。
* 选择需要现场处理的告警时：
  + 「从告警创建工单」→ 写入 WorkOrder（初始 status=pending 或 new）。
  + 在工单界面指定负责人、要求到场时间等（写入 assignee、assigned\_at 等字段）。

4.抢修/运维人员处理工单

* 运维人员在 B 端（或后续工人端 App）看到自己名下的工单列表。
* 流程大致为：
  + 接单 → 到达现场（on\_site）；
  + 执行检查/处置（processing，填写处置说明、记录措施、照片等）；
  + 复测确认（retest，记录复测压力/泄漏检测结果）；
  + 如果一切正常，申请结案（closed），形成完整的时间线。

5.调度中心审核与归档

* 调度员在 Web 工单详情页上复核现场记录（处置说明、时间链路、责任人等），确认无残余风险后批准结案。
* 系统将告警与工单绑定归档，用于事后追踪和审计。

6.报表与数据分析

* reports 模块对 Alert 与 WorkOrder 做聚合统计：
  + 按时间段统计告警数量、处理时长、逾期情况等。
  + 给出周报 / 月报，支持导出。
* B 端可以在页面看到图表化的 KPI（MTTA/MTTR、一段时间内的告警趋势、区域风险分布等）。

### 1.2系统业务流程图

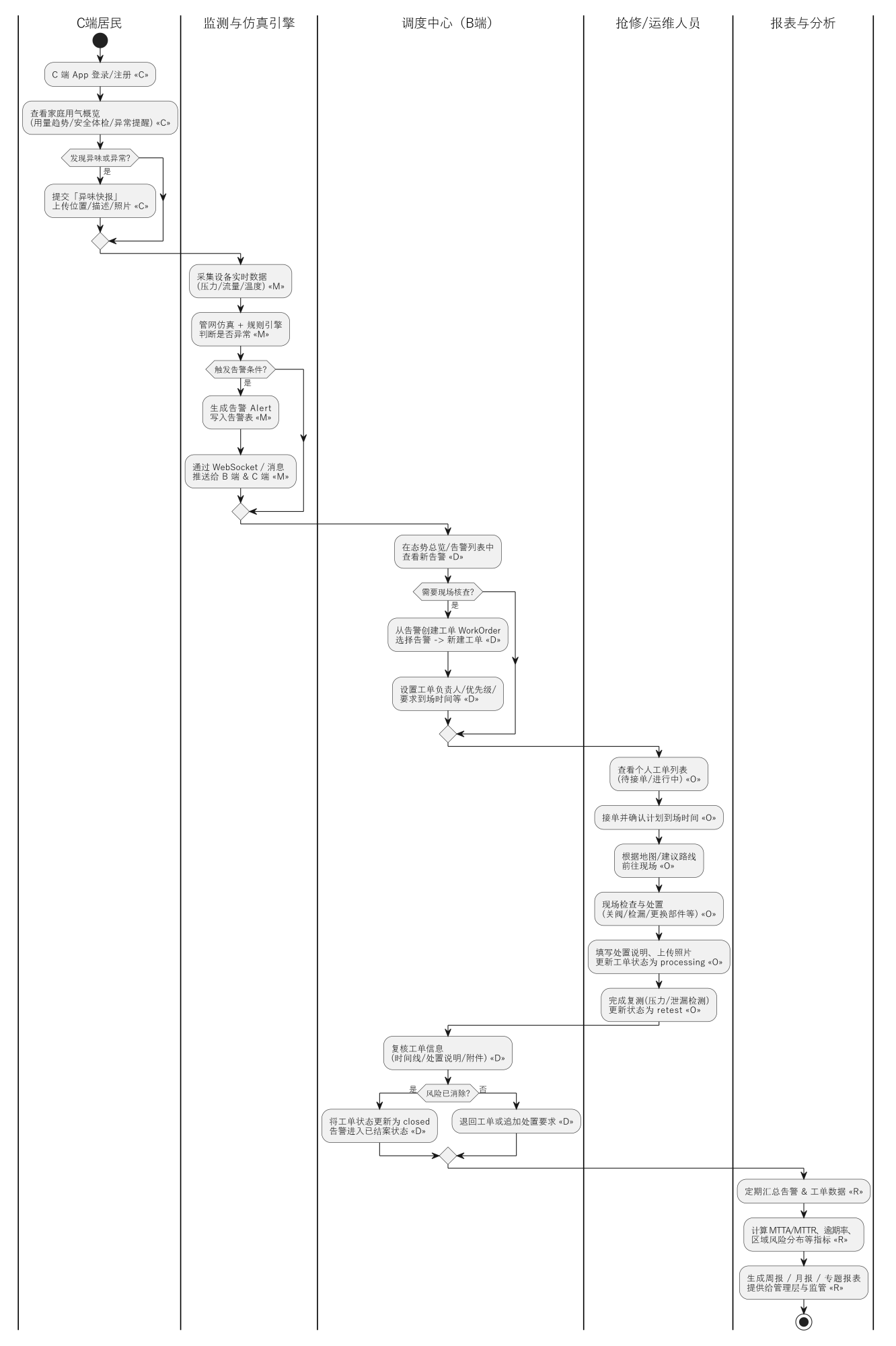


图5-1 系统业务流程图

## 2.系统功能结构（模块分层）

### 2.1 功能结构介绍

系统功能结构图建议使用三层结构 + 多角色视角来组织：表现层 → 业务应用层 → 支撑与数据层。

### （一）表现层（UI 层 / 客户端）

**1.B 端 Web 运维管理端**

* 态势总览（Dashboard）
  + Leaflet 地图展示园区管网与监测点。
  + 告警热度、重点区域标记。
* 告警管理
  + 告警列表（筛选/排序）。
  + 告警详情（趋势曲线、上下游设备、关联居民上报）。
* 工单管理
  + 工单列表（按状态、负责人、时间过滤）。
  + 工单详情与处置流转表单。
* 报表与统计
  + 周报/月报生成。
  + 指标看板（MTTA/MTTR、结案率、重复告警 TopN）。
* 运维人员个人中心
  + 个人信息、角色权限。
  + 操作日志、个人工单统计。

1. **C 端 居民移动端**

* 首页/用气总览
  + 当前所在家庭/房间的用气总览、余额/账单入口。
* 自动抄表与用量趋势
  + 近 7/30/90 天的用气曲线。
  + 分时段用气分析与节能建议。
* 家庭用气安全体检
  + 自查问卷、评分结果与整改建议。
* 异常提醒/告警列表
  + 与本住户相关的异常提醒（如长时间未关闭、疑似泄漏等）
* 异味快报
  + 上报位置、图片、文字描述，触发居民侧告警。
* 个人中心
  + 绑定住址、家人联系人。
  + 消息通知、隐私与授权设置。

### （二）业务应用层（后端服务与领域模块）

**1.用户与权限管理模块（Auth）**

* 用户登录认证（用户名/密码、Token）。
* 角色管理：管理员、调度员、运维工人、居民。
* 访问控制：不同角色访问不同接口与页面。

**2.设备与管网管理模块（Devices & Net）**

* 设备档案管理：设备类型、安装位置、所属管段、状态（启用/停用）。
* 设备状态管理：记录设备在线状态、最近心跳时间。
* 管网拓扑：管段、节点、阀门等的拓扑关系（为地图与仿真服务）

**3.监测与仿真模块（Monitoring & Simulation）**

* 时序数据存储（TimePoint）。
* net\_sim 仿真模块模拟管网压力/流量，并写入时序数据。
* 实时数据推送：通过 WebSocket（ws\_net）将最新数据推送到前端地图/趋势图。

**4.告警管理模块（Alerts）**

* 告警规则：阈值告警、趋势告警、组合规则（例如“压力持续下降 + 居民异味上报”）。
* 告警记录：增、删、改、查，支持状态更新（未确认/已确认/已关联工单）。
* 告警与设备/居民上报的关联关系维护。

**5.工单管理模块（WorkOrders）**

* 工单创建：从告警创建、手工创建。
* 状态机与流转：new → assigned → on\_site → processing → retest → closed。
* 处置表单：派单信息、现场记录、处置过程、复测数据、结案总结。
* 与告警联动：工单关闭时联动更新告警状态。

**6.居民与户号管理模块（Residents）**

* 居民基本信息与住址信息。
* 与设备、告警、工单的关联（哪个住户被哪些告警影响）。
* C 端上报记录（异味快报、体检记录）。

**7.报表与分析模块（Reports）**

* 周报/月报自动生成。
* 统计指标：告警/工单数量、分级分布、结案率、重复告警排行。
* 风险分析：高风险区域和高频隐患类型挖掘。

### （三）支撑与数据层

**1.数据库与模型层**

* models.py 中的 ORM 模型：User、Device、Alert、WorkOrder、TimePoint、Resident 等。
* db.py 提供会话管理和数据库初始化逻辑，当前使用 SQLite（可替换为 MySQL/PostgreSQL）。

**2.Schema 与数据传输对象**

* schemas.py 中的 Pydantic 模型负责前后端的数据结构约定，如 DeviceOut、AlertOut、WorkOrderOut、WorkOrderStatusUpdate 等。
* 实现前端与后端之间的数据验证与序列化。

**3.消息与推送**

* WebSocket 通道为前端 maps/dashboard 提供实时态势数据与告警推送。

**4.配置与公共组件**

* 日志、配置文件、中间件（如 CORS、中间错误处理）等为整个系统提供基础支撑。

### 2.2系统功能结构图

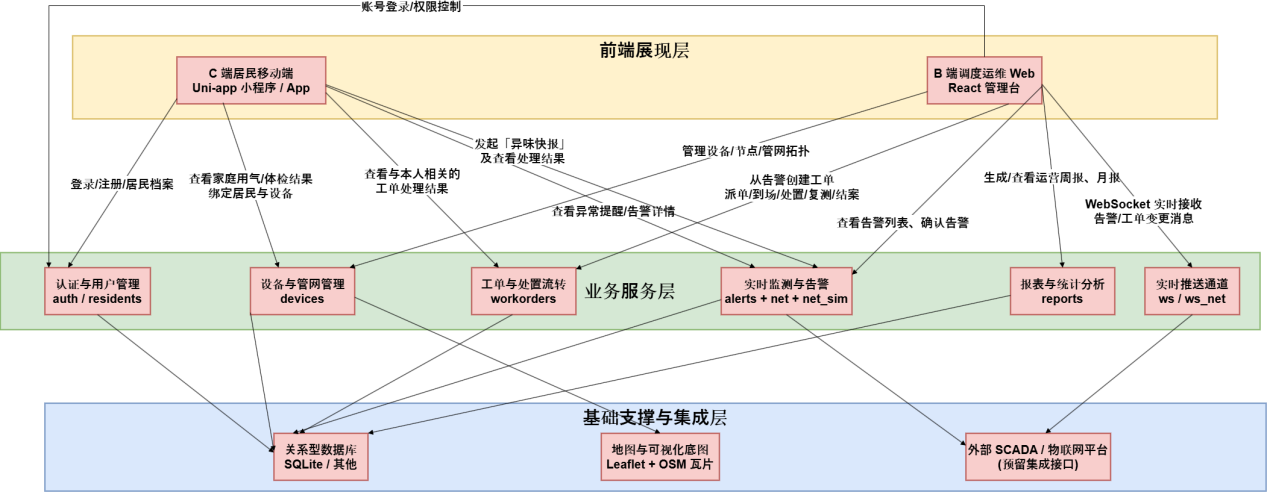


图5-2 系统功能结构图

### 2.3类图（实体类图）

系统采用面向对象的前端架构设计，旨在构建一套资产可视化、实时数据监控、异常智能响应、工单闭环处置与公众协同参与于一体的燃气安全综合监管平台。类图从用户交互视角出发，划分为网页管理端与移动作业端两大子系统，分别面向监管人员与一线运维/居民用户，通过共享核心数据模型与服务接口，实现业务逻辑的统一与交互体验的差异化，详细模块设计如下：

1. **网页管理端：资产全景监控与智能告警中枢**

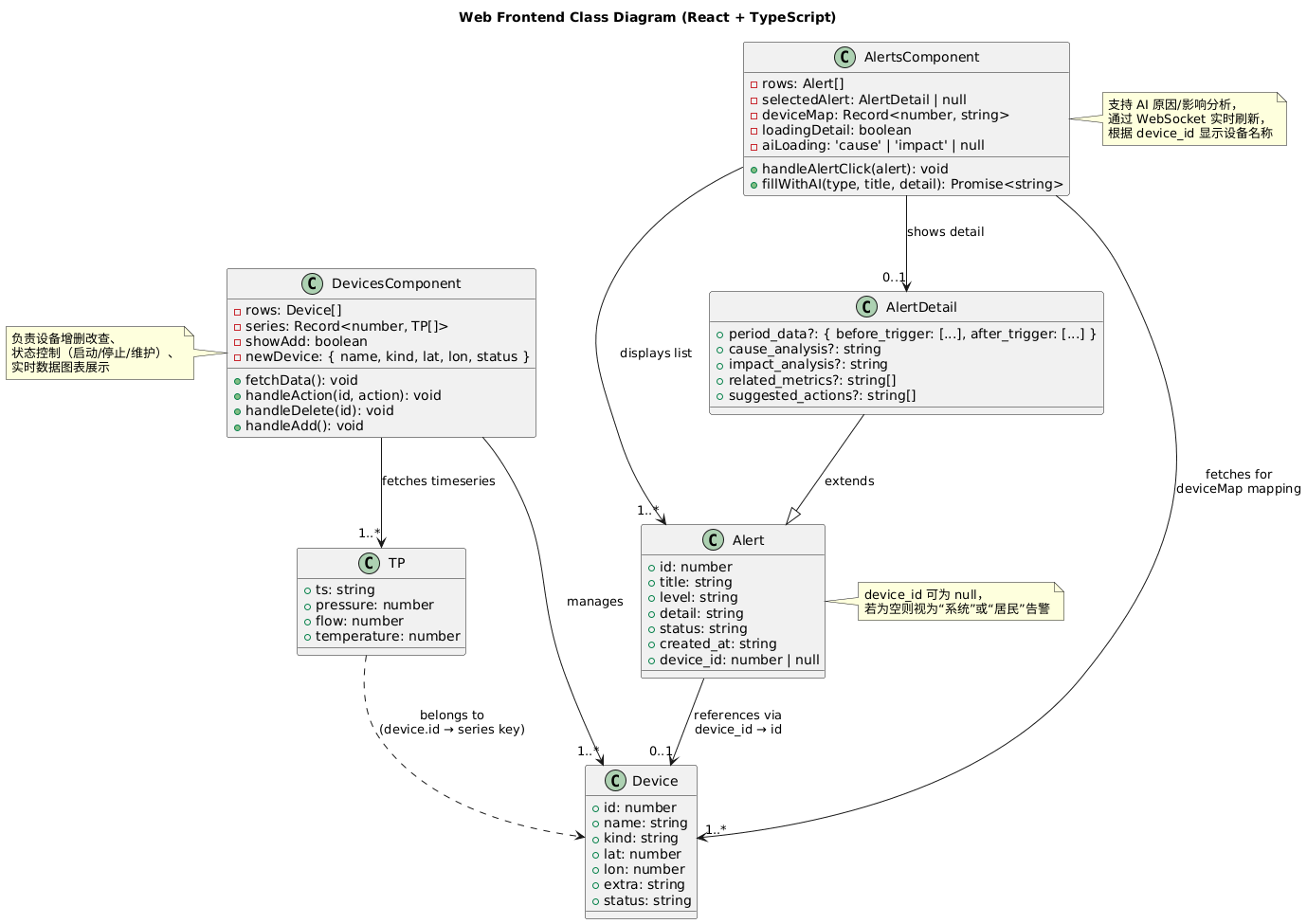
****

图5-3 web端类图

核心组件：DevicesComponent（设备管理组件）与 AlertsComponent（告警监控组件）构成网页端的双引擎。

资产与遥测可视化：DevicesComponent 管理多个 Device（设备实体），每个设备关联一系列 TP（TimePoint，时间序列数据点），用于展示压力、流量、温度等实时曲线，支持设备启停、维护状态切换及增删改查操作，实现对燃气管网物理资产的全生命周期可视化管控。

多源告警融合：AlertsComponent 聚合来自物理设备的真实告警（Alert）与数字孪生仿真节点的预测性风险信号，通过 WebSocket 实现告警列表的动态刷新，并支持 AI 辅助分析。每条 Alert 可扩展为 AlertDetail，记录异常成因与影响评估，形成“监测—识别—研判”一体化的智能监控闭环。

**（2）移动端应用：现场作业协同与公众参与入口**

双角色支持：系统为运维工人与社区居民提供定制化视图，实现“专业处置”与“社会共治”双轨并行。

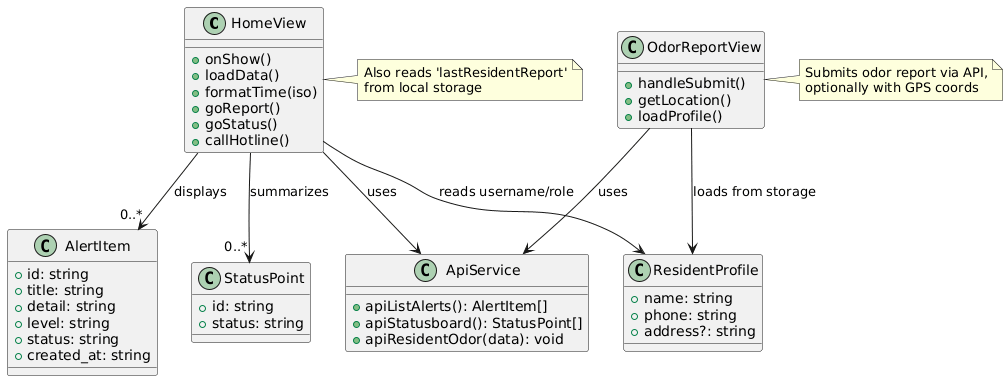


图5-4 移动端类图

工单驱动的运维闭环：WorkerHomeView 展示分配给当前用户的 WorkOrder（工单）列表，按状态（待到场、处理中、已关闭）分类统计；WorkOrderDetailView 呈现工单详情、标准操作流程（SOPItem）及完整时间线（TimelineStep），引导工人按规范完成从到场、处理到复测的全流程操作，确保处置过程可追溯、可验证。

居民异味上报通道：ResidentHomeView 为公众提供风险感知入口，展示近期告警与设施状态；OdorReportView 允许居民填写 ResidentReport（异味上报表），自动采集位置信息并提交至后台，其内容可作为潜在告警触发源（may\_trigger 关系），辅助系统提升泄漏点定位精度，体现“人人都是安全员”的社会治理理念。

**（3）共享核心服务：统一数据与交互支撑**

两端共用一套共享核心模块，包括：

（1）数据模型：WorkOrder、AlertItem、StatusPoint、ResidentReport 等实体，确保业务语义一致；

（2）基础服务：ApiService 封装统一 RESTful 接口，AuthManager 管理用户认证，NavigationService 控制页面路由，FormatUtils 提供状态流转与时间格式化等通用逻辑。

该设计既保证了前后端数据契约的稳定性，又实现了代码复用与维护效率的最大化。

综上，本前端架构通过分层解耦、角色定制、服务共享的设计原则，有效支撑了燃气安全监管平台“监、管、控、报、处”五位一体的业务需求，为系统高可用性与用户体验一致性提供了坚实保障

### 2.4数据库E-R图

本系统采用型数据库设计，旨在构建一套资产管理、实时数据、异常关系响应、工单闭环处理为一体的监控燃气安全综合监管平台。ER图主要由资产层、数据层、业务层和警报四大核心模块构成，各设备间通过严谨的外键约束关联，确保了数据的一致性和可靠性。

详细模块设计如下：

（1）资产与遥测模块 (Asset & Telemetry)

核心实体：（Device设备表）是系统的物理基础，记录了燃气管网中的传感器、阀门、泵站等关键设施的静态属性（如经纬度坐标lat/lon、设备类型kind）。

相互数据：Device与TimePoint相互数据表呈一对多关系。系统持续采集上传设备的压力、流量、温度等遥测数据，存储于TimePoint中，用于历史趋势分析和实时状态监测。

（2）智能监控与另外中心（监控和警报）

一组聚合：Alert（一组表）是异常处理的中枢。它不仅接收来自物理Device的触发信号，还通过SimulationNode（仿真节点表）接收来自数字邻居生仿真引擎的预测性同样。

多源融合：该设计支持“虚实结合”的监控模式，无论是物理设备的真实故障，还是仿真模拟的潜在风险，均统一汇聚至统一中心进行分级处理（level：严重/警告）。

（3）业务闭环与公众参与（工作流程和参与）

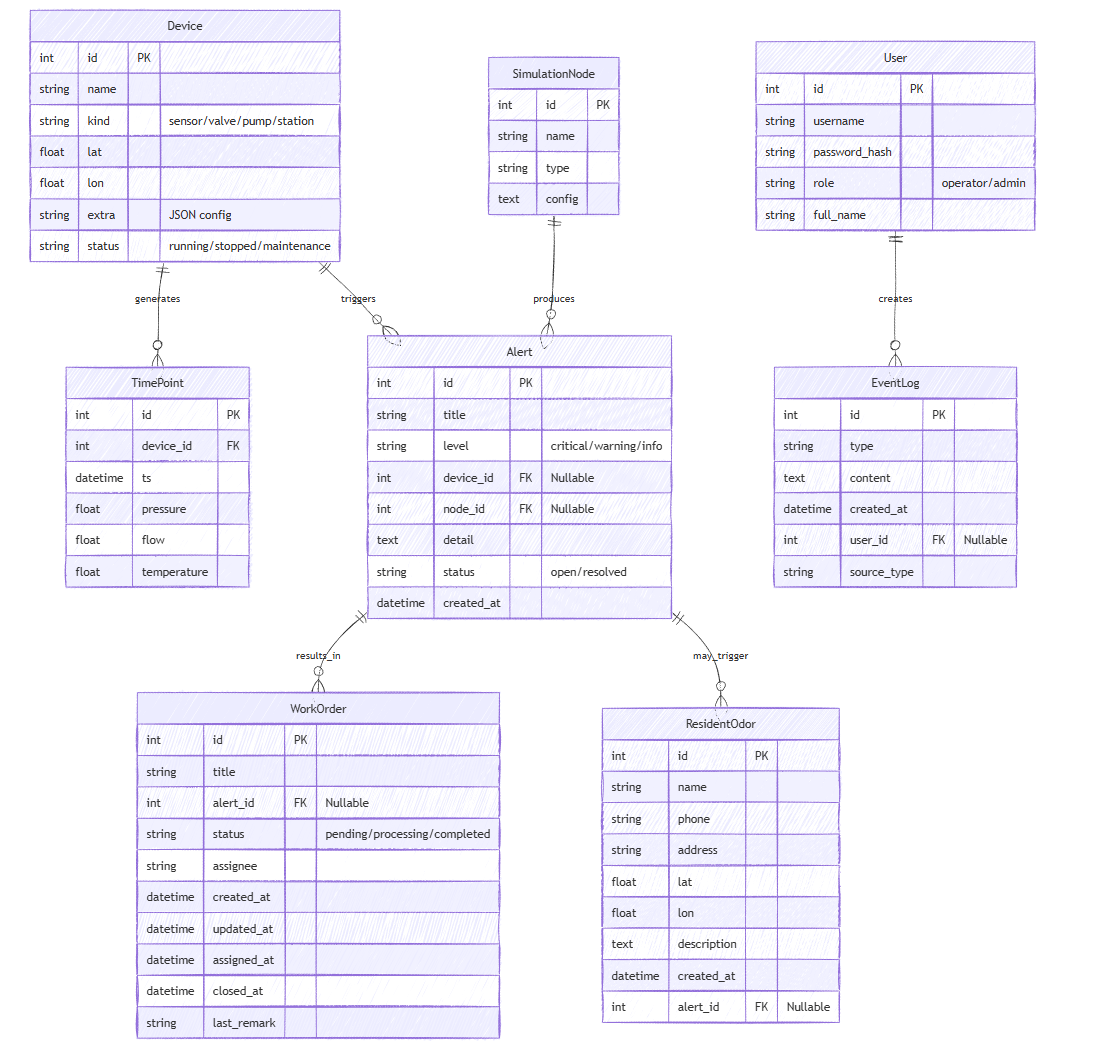
运维闭环：Alert与WorkOrder（工单表）形成一对多的业务流转关系。当系统确认一致有效后，可生成对应的维修工单，记录指派人员、处理进展及反馈，实现从“发现问题”到“解决问题”的全环闭环流程管理。

公众上报：（ResidentOdor居民异味上报表）体现了系统的社会治理功能。居民提交的异味线索可与系统Alert进行关联，通过辅助判断（may\_trigger关系），提高漏洞定位的准确性。

（4）系统管理与审计（行政与审计）

用户权限：User（用户表）定义了系统操作员和管理员的角色权限。

操作审计：为了满足安全审计需求，User与EventLog（日志表）关联，记录关键操作行为（如登录、派单、设备控制），确保系统操作可追溯。

图5-5 安燃云Pro数据库设计ER图

# 六、开发成果展示

## 1. 系统启动与访问方式

（1）后端服务：在 backend 目录启动 FastAPI（uvicorn），提供鉴权、设备/告警/工单/居民上报、仿真数据、周报导出等接口。

（2）前端页面：在 frontend 目录启动 Vite + React 项目，通过浏览器访问“安燃云 Pro”管理端。

（3）演示账号：管理员 admin/admin123；调度员 dispatcher/dispatcher123；运维人员 operator/operator123。不同角色可验证权限控制与功能边界。

## 2. 登录与权限验证



图6-1 安燃云Pro登录界面

## 3. 态势总览

态势总览页以地图为核心，叠加燃气设备点位与管网连线，并提供“常规/压力/流量”等视图切换。点击设备可查看压力（kPa）、流量（m³/h）、状态与最后检查时间等关键指标，实现对园区用气安全的可视化巡检。

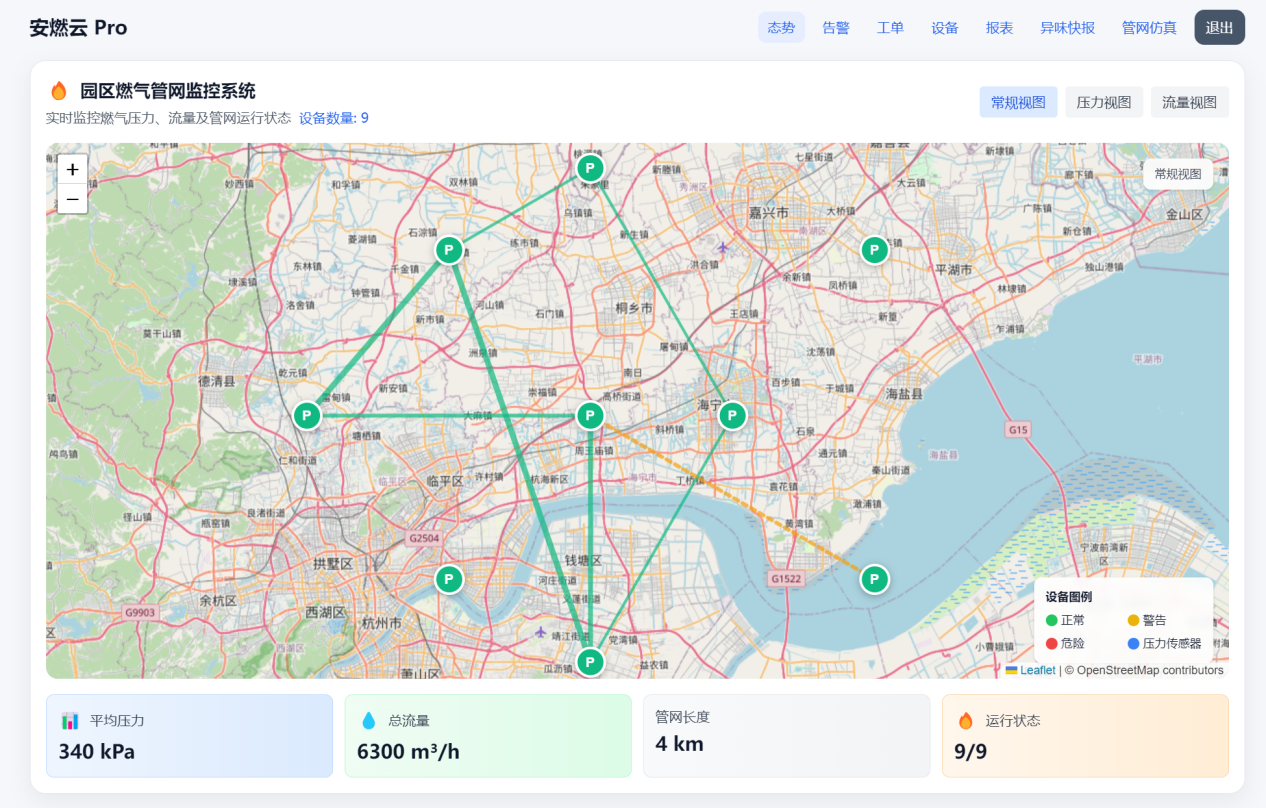


图6-2 态势总览页面（地图+设备点位+管网）

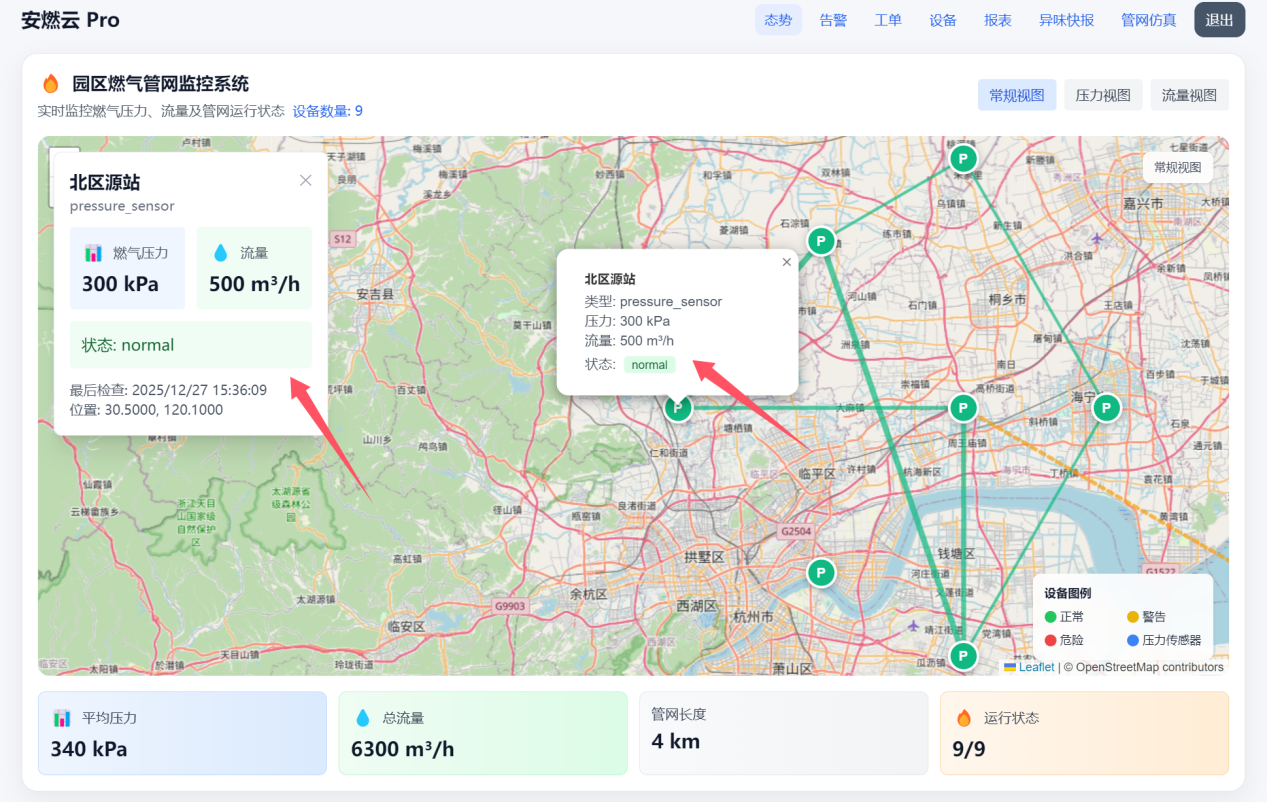


图6-3 设备详情弹窗（点击点位查看实时指标）

## 4. 告警管理

告警模块提供告警列表与详情查看能力，可按等级、状态、时间等维度快速定位风险点。系统支持从告警进入处置流程（生成工单/状态流转），将“发现问题”与“处理闭环”串联起来。在此基础上，系统引入基于历史数据与规则模型的 AI 分析机制，对告警触发时的运行参数、环境特征及行为模式进行综合分析，自动生成告警产生的可能原因及其对系统运行、安全与业务连续性的影响评估。

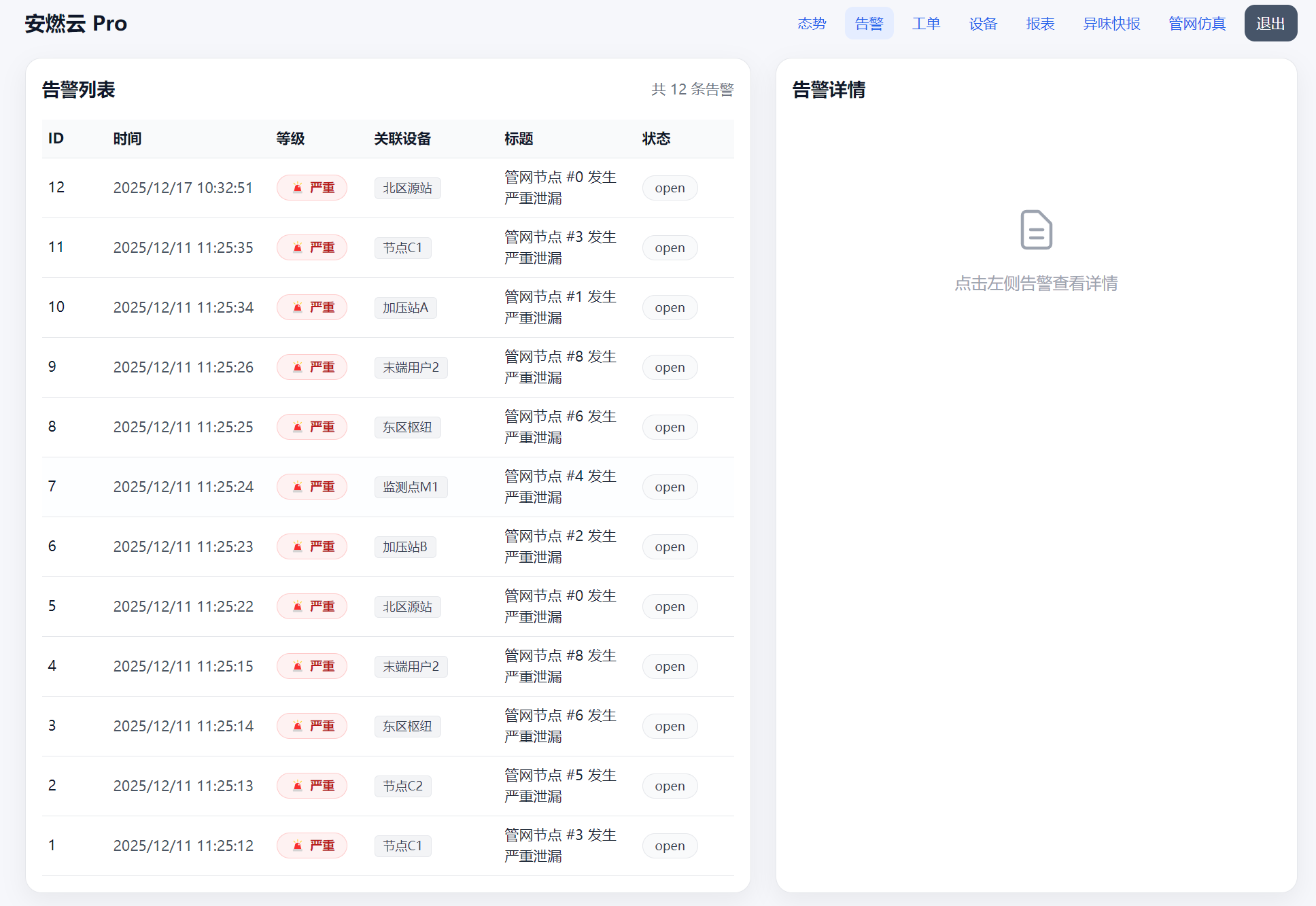


图6-4 告警列表页面



图6-5 AI分析生成告警原因与影响

## 5. 工单闭环处置

工单模块用于跟踪处置过程，支持工单创建、指派、状态更新与备注记录。典型流程为：从告警创建工单 → 指派责任人 → 现场处理 → 回填结果 → 完成关闭。通过统一的状态枚举与时间字段（created\_at/updated\_at），保证流程可追溯。



图6-6 工单列表页面



图6-7 工单状态更新

## 6. 设备管理与数据查看

设备模块用于维护燃气监测设备的基础信息（名称、类型、经纬度、扩展字段、状态等），并展示设备的时序数据（压力/流量/温度等）。支持新增设备、定位获取、列表管理与趋势小图展示，便于运维人员快速判断设备健康度与波动异常。

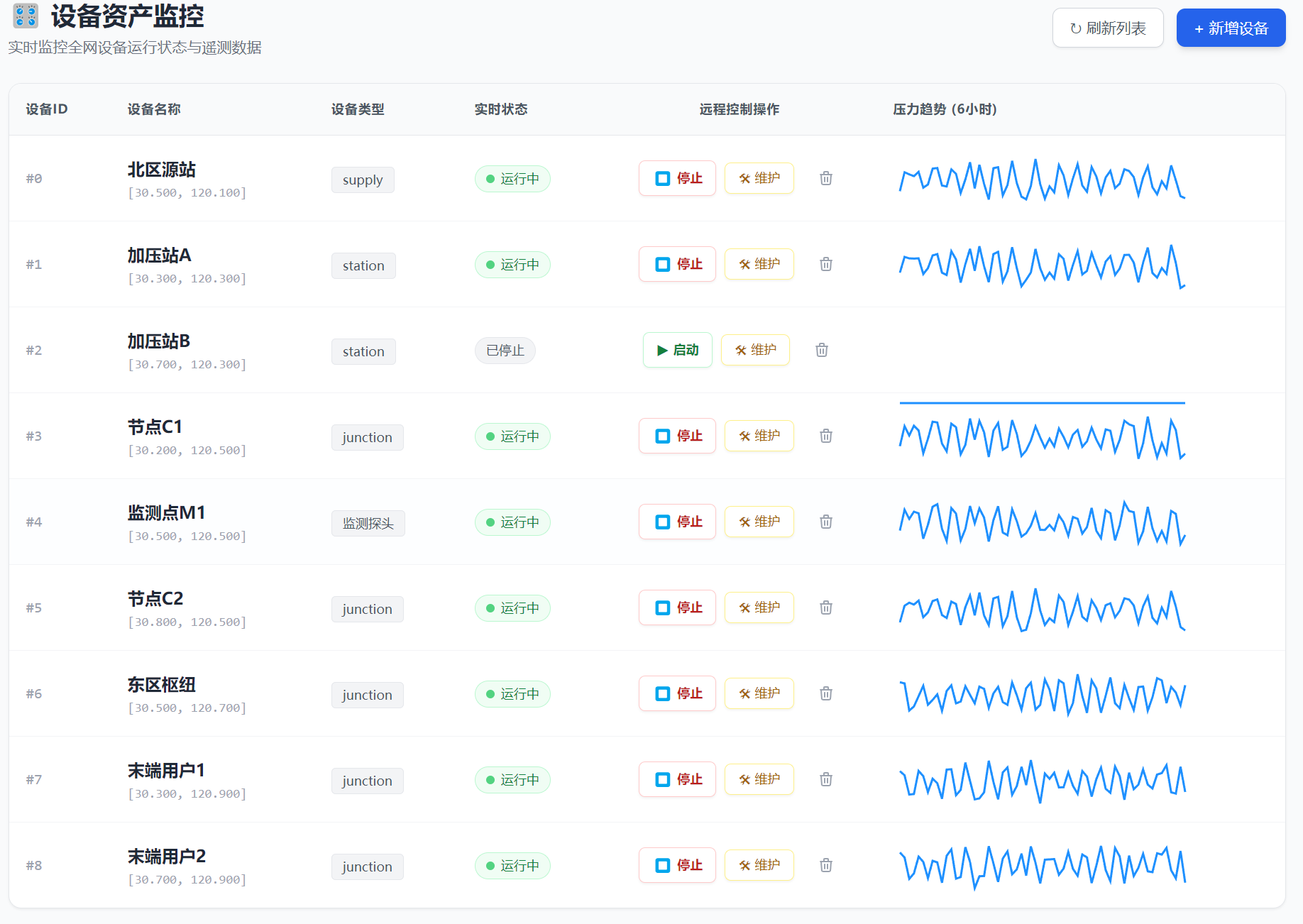


图6-8 设备管理页面（设备列表）



图6-9 新增设备信息表单

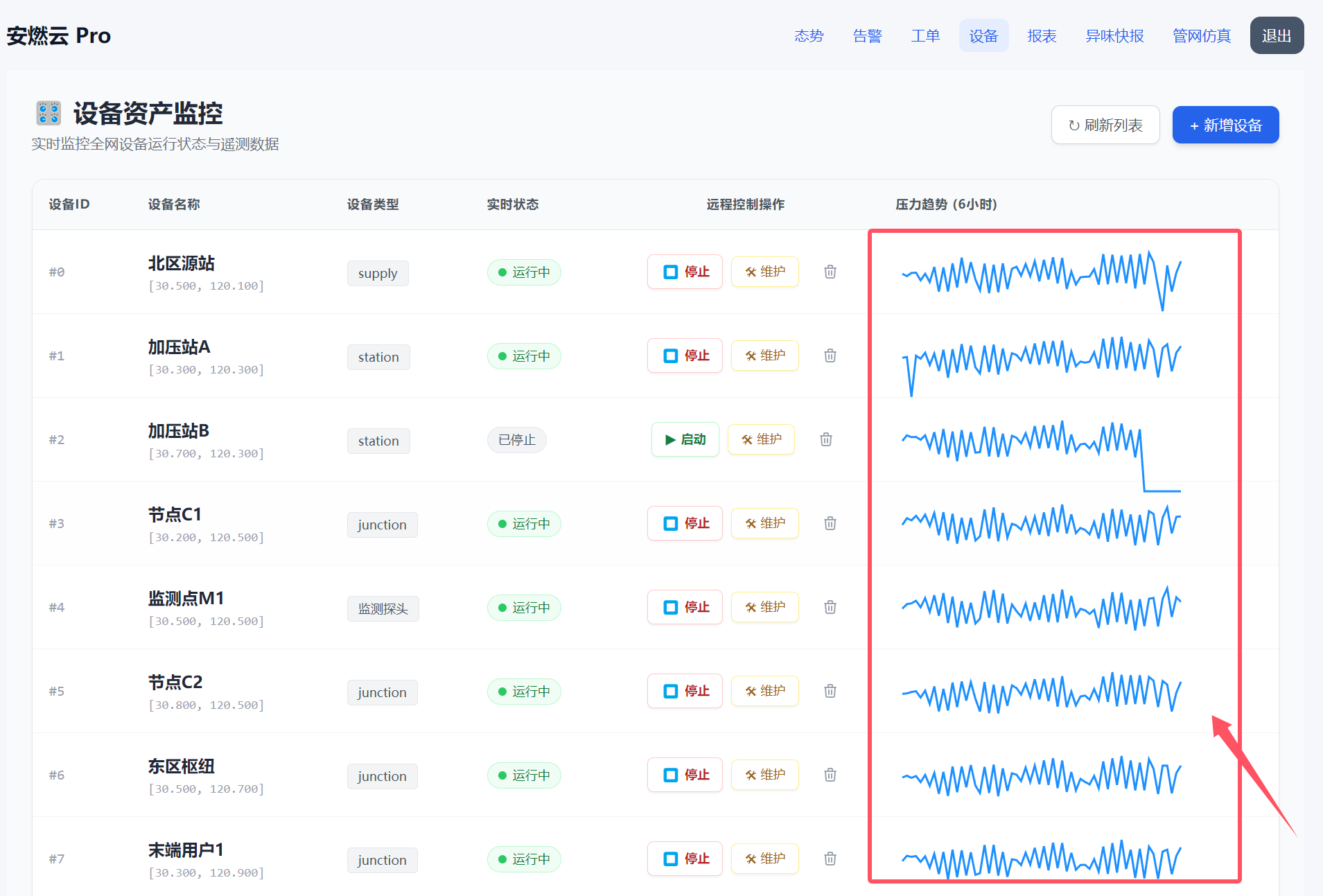


图6-10 设备时序数据趋势展示（MiniChart）

## 7. 异味快报（居民上报）

异味快报模块面向居民侧提供快速上报入口，填写姓名、电话、地址、经纬度与描述信息，可辅助调度中心快速定位线索并与系统告警联动，提高问题发现的覆盖率与响应速度。前端对经纬度范围进行校验，减少无效上报。



图6-11 异味快报上报页面

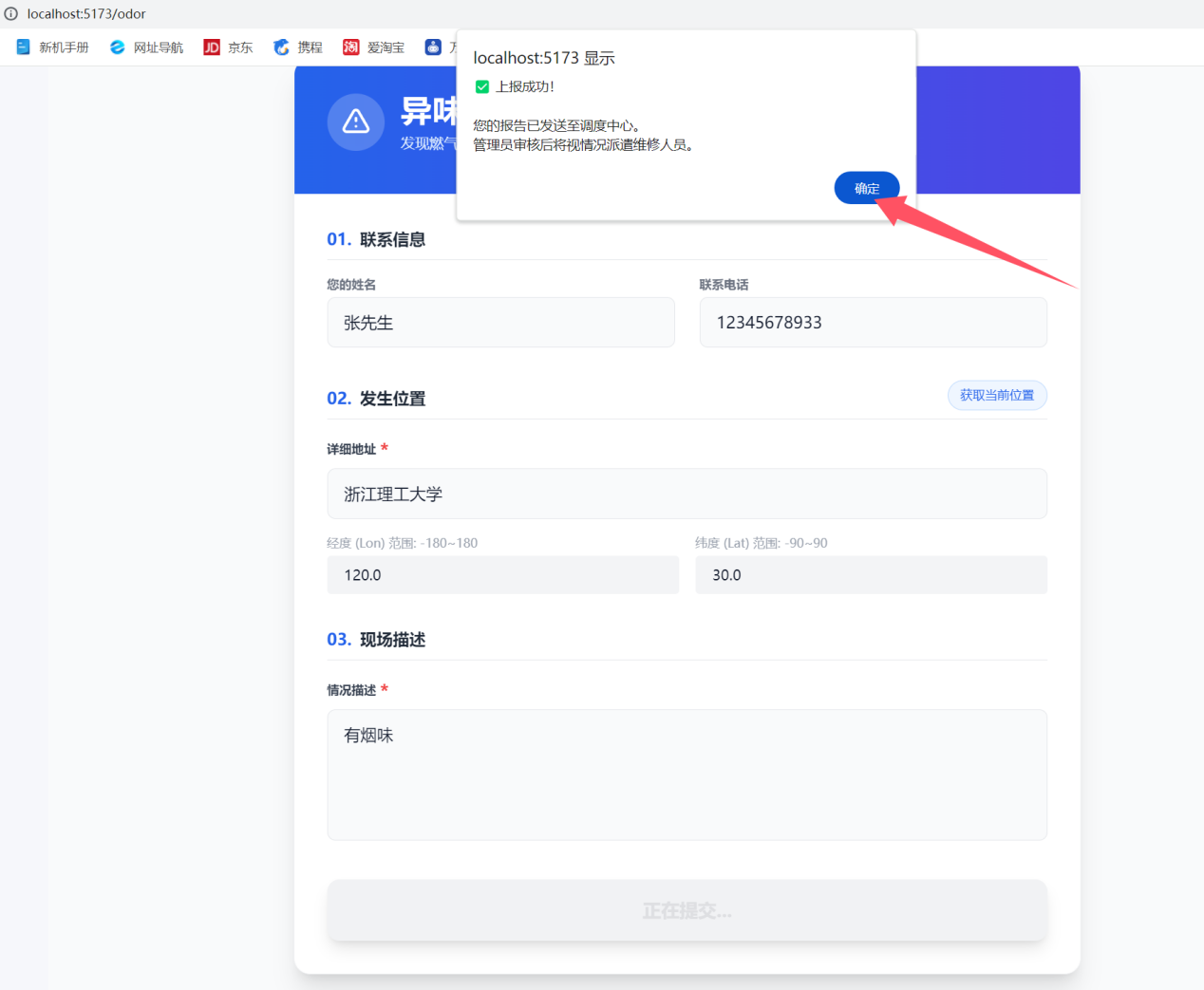


图6-12 异味快报提交结果反馈

## 8. 管网仿真与故障注入

管网仿真模块在前端以 Canvas 方式绘制管网节点与边，并以粒子流动与数值状态展示压力/流量变化。支持对指定节点注入“泄漏”等故障，用于演示数字孪生场景下的预测性告警与联动处置，为后续接入真实管网模型奠定基础。

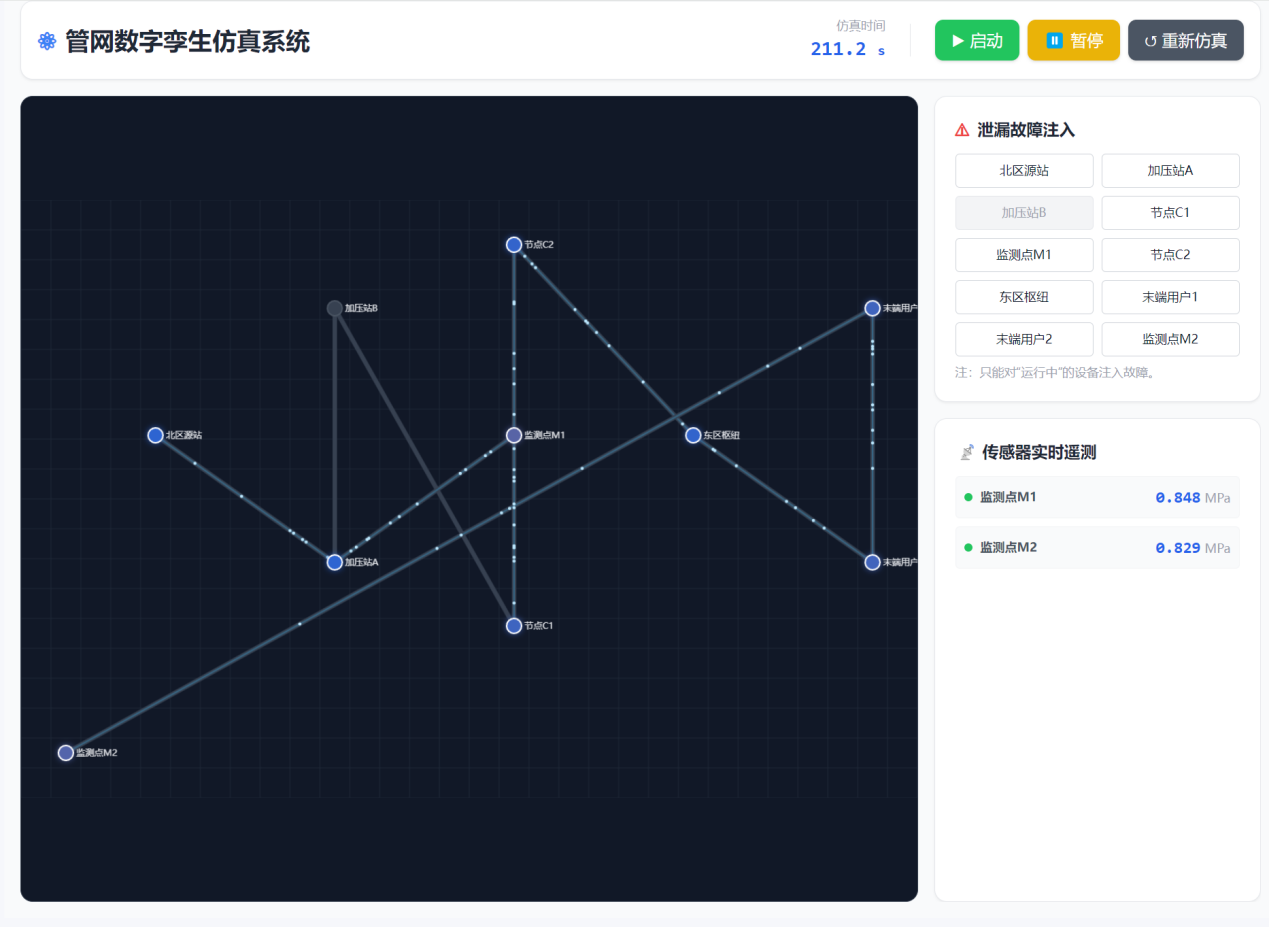


图6-13 管网仿真可视化界面

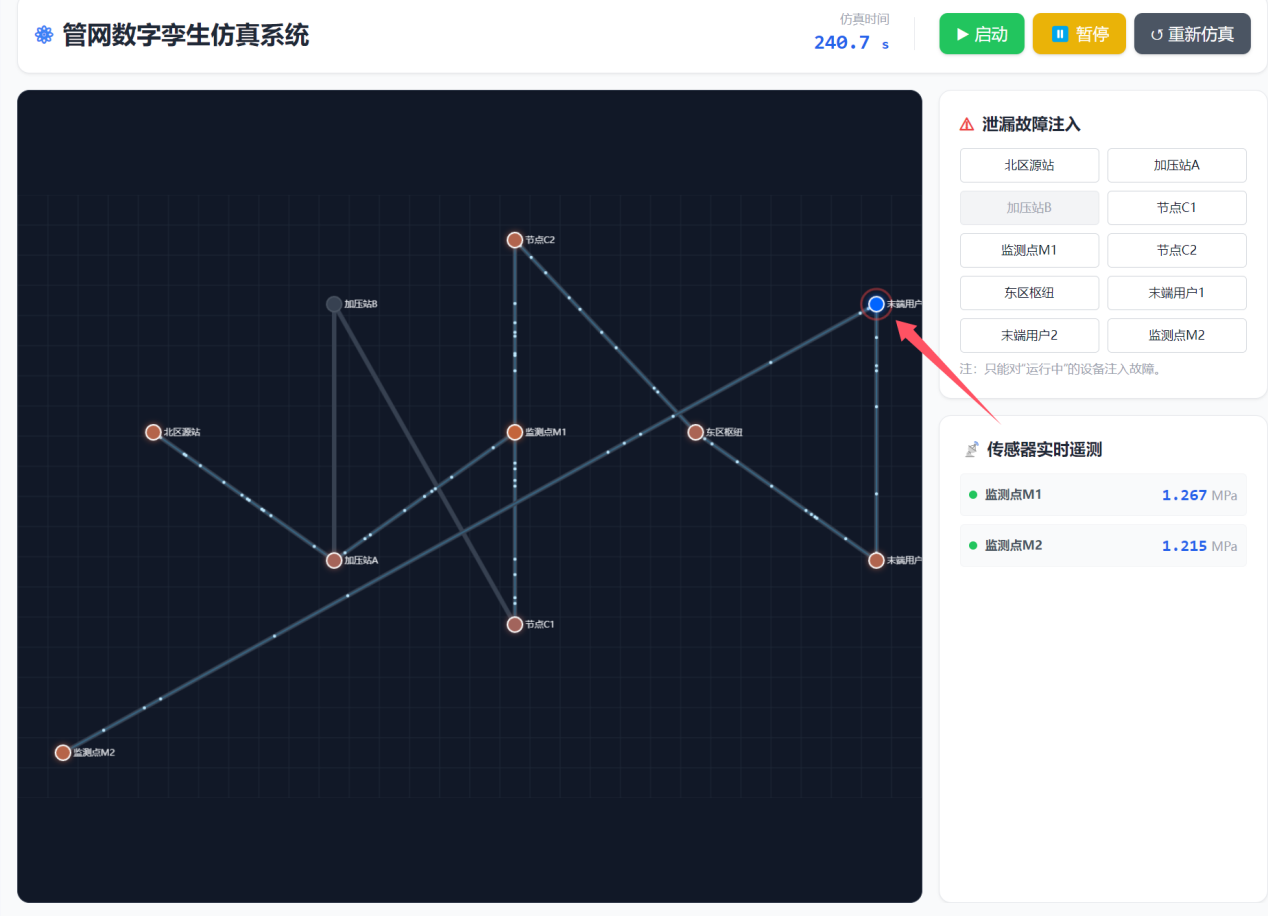


图6-14 故障注入后的状态变化展示

## 9. 周报导出（PDF）

系统支持一键生成周报 PDF，用于对本周期告警、工单处置与关键指标进行留痕与归档，便于管理人员复盘与审计。

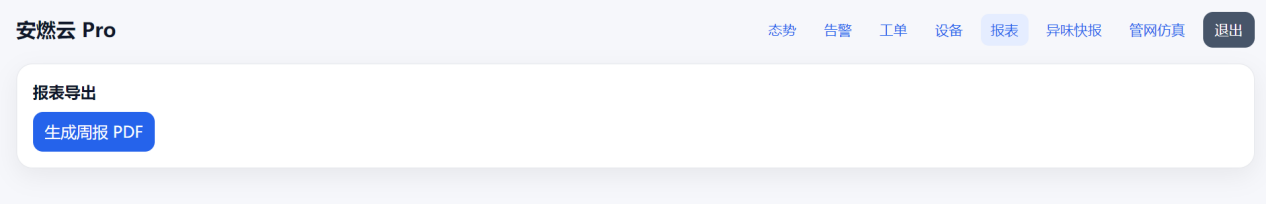


图6-15 报表导出页面（生成与下载周报）



图6-16 导出的周报PDF示例

## 10. 移动端成果展示（居民端 + 工人端）

为扩展燃气安全的触达范围与处置效率，系统在管理端（Web）之外补充了移动端页面：居民端用于公众参与线索上报与周边状态查询；工人端用于抢修/巡检人员接收告警、查看工单与现场处置留痕。

### 10.1 居民端（公众参与）

居民端面向公众用户，提供异味快报上报、风险线索提交与周边安全信息查看等功能，便于快速采集线索并形成闭环反馈。

|  |  |
| --- | --- |
| 图6-17 移动端登录界面（居民身份入口） | 图6-18 居民端首页（便民服务与运行概况） |
| 图6-19 居民端异味上报表单（地址与异常选择） | 图6-20 居民端异味上报表单（坐标获取与示例填写） |

|  |  |
| --- | --- |
| 图6-21 居民端异味上报提交（联系方式与立即上报） | 图6-22 附近管网状态查询（监测点/管段概况） |
| 图6-23 上报记录回显（最近一次上报详情） |  |

### 10.2 工人端（运维处置）

工人端面向巡检/抢修人员，支持接收告警与工单、查看现场处置要点、回填处置过程与结果，实现一线运维的移动化闭环。

|  |  |
| --- | --- |
| 图6-24 移动端登录界面（抢修/巡检身份入口） | 图6-25 工人工作台首页（待处理工单与快捷入口） |
| 图6-26 工作台告警提醒与推荐巡检路线 | 图6-27 抢修任务列表（今日工单与状态流转） |
| 图6-28 巡检与管网（路线生成与风险段提示） | 图6-29 今日巡检计划（打卡式勾选执行） |
| 图6-30 告警中心（搜索与多维筛选） | 图6-31 告警中心列表（严重告警展示） |

|  |  |
| --- | --- |
| 图6-32 工作台工单概览与设备状态统计 |  |
|  |  |

## 11. 关键代码展示

本节选取项目中与“鉴权登录、告警查询、实时推送、管网仿真”直接相关的关键代码片段，用于说明核心实现思路。为便于阅读，仅保留主流程与关键语句，省略了部分非核心细节（以注释省略号表示）。

**代码6-1 后端服务启动与模块挂载（main.py）**

*# backend/app/main.py*

from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

from fastapi.staticfiles import StaticFiles

from .settings import settings

from .db import Base, engine

*# 导入各个业务模块路由*

from .routers import auth as r\_auth, devices as r\_devices, alerts as r\_alerts, \

    workorders as r\_workorders, reports as r\_reports, ws as r\_ws, net as r\_net

from .services import sim\_watcher, net\_sim

*# 1. 数据库表结构初始化*

Base.metadata.create\_all(bind=engine)

app = FastAPI(title=settings.APP\_NAME)

*# 2. 跨域中间件配置*

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=settings.CORS\_ORIGINS,

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

*# 3. 挂载静态资源（用于访问生成的PDF报表）*

import os

os.makedirs("uploads/reports", exist\_ok=True)

app.mount("/uploads", StaticFiles(directory="uploads"), name="uploads")

*# 4. 注册路由*

app.include\_router(r\_auth.router)

app.include\_router(r\_devices.router)

app.include\_router(r\_alerts.router)

app.include\_router(r\_net.router)

*# ... 其他路由省略*

*# 5. 系统启动事件钩子*

@app.on\_event("startup")

async def on\_startup():

    from .auth import create\_user\_if\_absent

    from .db import SessionLocal

*# 初始化默认用户（管理员/调度员/操作员）*

    with SessionLocal() as db:

        create\_user\_if\_absent(db)

*# 启动管网仿真引擎线程*

    net\_sim.start()

*# 启动数据监控与入库线程*

    sim\_watcher.start()

代码亮点：

1.生命周期管理：利用 on\_event("startup") 钩子函数，将“数据库种子数据填充”、“仿真引擎启动”、“监控线程启动”等操作与 Web 服务启动过程绑定，实现了“一键启动，全栈就绪”。

2.静态资源挂载：通过 StaticFiles 将本地 uploads 目录映射为 Web 路由，巧妙解决了生成的 PDF 报表无法通过前端直接访问的问题。

3.模块化路由设计：主文件仅负责“组装”，具体的业务逻辑被拆分到 routers 目录下的不同模块中，降低了代码耦合度。

**代码6-2 登录鉴权与JWT Token生成（auth.py）**

*# backend/app/auth.py*

from fastapi import Depends, HTTPException

from fastapi.security import OAuth2PasswordBearer

from jose import jwt

*# 采用 pbkdf2\_sha256 算法替代 bcrypt，解决环境兼容性问题*

from passlib.hash import pbkdf2\_sha256

from datetime import datetime, timedelta

from sqlalchemy.orm import Session

from .models import User

oauth2\_scheme = OAuth2PasswordBearer(tokenUrl="/auth/login")

def create\_user\_if\_absent(db: Session):

    """自动创建默认的三类角色账号"""

    if not db.query(User).filter\_by(username="admin").first():

        db.add(User(username="admin", password\_hash=pbkdf2\_sha256.hash("admin123"), role="admin", full\_name="Administrator"))

*# ... 其他角色创建逻辑省略*

    db.commit()

def authenticate(db: Session, username: str, password: str):

    """用户登录验证核心逻辑"""

    u = db.query(User).filter(User.username==username).first()

*# 验证哈希密码是否匹配*

    if not u or not pbkdf2\_sha256.verify(password, u.password\_hash):

        return None

    return u

def create\_token(username: str, role: str) -> Token:

    """生成 JWT 令牌，内含用户角色信息"""

    expire = datetime.utcnow() + timedelta(minutes=settings.ACCESS\_TOKEN\_EXPIRE\_MINUTES)

    payload = {"sub": username, "role": role, "exp": expire}

    token = jwt.encode(payload, settings.SECRET\_KEY, algorithm="HS256")

    return Token(access\_token=token)

代码亮点：

1.算法升级适配：项目中特意选用了 pbkdf2\_sha256 哈希算法，相比于传统的 bcrypt，它在 Windows 开发环境下具有更好的兼容性，解决了依赖库安装报错的痛点。

2.无状态认证：采用 JWT 标准，将用户身份（sub）和权限角色（role）直接封装在 Token 中，服务端无需存储 Session，极大地提升了系统的并发处理能力。

3.自动化运维：create\_user\_if\_absent 函数确保了系统首次部署时，数据库中必然存在可用的管理员账号，无需手动修改数据库。

**代码6-3 告警列表联合查询与返回结构（alerts.py）**

*# backend/app/routers/alerts.py*

@router.get("/", response\_model=list[AlertOut])

def list\_alerts(db: Session = Depends(get\_db), user=Depends(get\_current\_user)):

*# 构建联合查询语句 (Left Outer Join)*

*# 关联 Alert 表与 Device 表，逻辑：*

*# 告警可能关联 device\_id，也可能关联 node\_id (仿真节点)，需使用 OR 条件匹配*

    stmt = (

        select(Alert, Device.name)

        .outerjoin(Device, or\_(Alert.device\_id == Device.id, Alert.node\_id == Device.id))

        .order\_by(Alert.created\_at.desc())

        .limit(200)

    )

    rows = db.execute(stmt).all()

*# 数据组装：将查到的 Device.name 动态注入到返回结果中*

    results = []

    for alert, dev\_name in rows:

        results.append(AlertOut(

            id=alert.id,

            title=alert.title,

            level=alert.level,

            device\_id=alert.device\_id,

            detail=alert.detail,

            status=alert.status,

            created\_at=alert.created\_at,

*# 如果没关联到设备（如系统级告警），显示默认值*

            device\_name=dev\_name or "未知设备/手动上报"

        ))

    return results

代码亮点：

1.复杂 ORM 查询：使用了 SQLAlchemy 的 outerjoin 配合 or\_ 逻辑，解决了本项目中特有的“双外键关联”问题（告警源可能是物理设备，也可能是仿真节点），避免了分别查询带来的性能损耗。

2.数据扁平化处理：在后端完成了数据的清洗和组装，将关联表的 device\_name 直接合并入返回对象，前端无需再发起额外的 API 请求去查询设备名称，显著提升了页面加载速度。

3.健壮性设计：加入了 dev\_name or "..." 的空值处理逻辑，确保即使设备已被删除，告警列表依然能正常显示，不会导致页面崩溃。

**代码6-4 WebSocket广播器与推送机制（broadcast.py / ws.py）**

*# backend/app/utils/broadcast.py*

class Broadcaster:

    """WebSocket 广播器：用于向多个连接的前端客户端同时发送消息"""

    def \_\_init\_\_(self):

        self.\_clients: Set[WebSocket] = set()

        self.\_lock = Lock()  *# 异步锁，确保并发安全*

    async def register(self, ws: WebSocket):

        await ws.accept()

        async with self.\_lock:

            self.\_clients.add(ws)

    async def publish(self, data: Any):

        """向所有在线用户广播数据"""

        dead = []

*# 复制集合快照进行遍历*

        clients\_snapshot = list(self.\_clients)

        for ws in clients\_snapshot:

            try:

                await ws.send\_json(data)

            except Exception:

*# 捕获发送失败的连接（用户已断开）*

                dead.append(ws)

*# 统一清理死连接*

        if dead:

            async with self.\_lock:

                for ws in dead:

                    self.\_clients.discard(ws)

*# 全局单例：分别用于告警推送和仿真数据推送*

alertcaster = Broadcaster()

netcaster = Broadcaster()

代码亮点：

1.高并发连接管理：使用 Python 的 set 集合存储 WebSocket 连接对象，配合 asyncio.Lock 异步锁，确保在多用户同时连接或断开时，连接池的状态是线程安全的。

2.自我修复机制：在推送消息的过程中，自动检测并收集“死连接”，并在发送结束后统一清理。这种“惰性清理”策略有效防止了因客户端异常掉线导致的服务器资源泄露。

3.单例模式应用：通过全局实例 alertcaster 和 netcaster，在整个应用生命周期内复用广播器，任何业务模块只需导入实例即可发送消息，解耦了业务逻辑与通信逻辑。

**代码6-5 管网仿真故障注入与状态更新（net\_sim.py / NetSim.tsx）**

*# backend/app/services/net\_sim.py*

class NetSimulation:

    def \_loop(self):

        """仿真计算主循环（运行在独立线程中）"""

        while self.running:

            if self.paused:

                time.sleep(0.5)

                continue

            with self.lock:

                self.t += 0.1

*# 1. 基础压力波动模拟（正弦波）*

                base\_pressure = 1.0 + 0.3 \* math.sin(self.t \* 0.5)

                for n in self.nodes:

                    nid = n["id"]

*# 仅计算运行中的设备*

                    if self.node\_status.get(nid) != 'running':

                        self.node\_pressures[nid] = 0.0

                        continue

*# 2. 计算节点压力：基础波 + 相位偏移 + 随机噪声*

                    phase = (nid % 5) \* 0.5

                    noise = random.uniform(-0.05, 0.05)

                    p = base\_pressure + 0.1 \* math.sin(self.t \* 0.8 + phase) + noise

*# 3. 故障注入逻辑：如果该节点正在泄漏，压力骤降至20%*

                    if self.leak\_node == nid:

                        p \*= 0.2

                    self.node\_pressures[nid] = max(0.1, p)

*# 4. 跨线程广播状态*

            self.\_broadcast()

            time.sleep(0.1) *# 控制仿真频率约为 10Hz*

代码亮点：

1.数学建模仿真：摒弃了简单的随机数，而是采用 sin 正弦函数叠加随机噪声的方式，模拟出真实世界中燃气管道压力随时间周期性波动（如昼夜峰谷）的特性，数据曲线平滑且真实。

2.故障动态注入：代码中内置了 leak\_node 检测逻辑，允许用户在运行时动态注入泄漏故障。故障一旦触发，压力值实时发生非线性突变（p \*= 0.2），实现了“所见即所得”的推演效果。

3.线程间通信桥梁：由于仿真跑在同步的 threading.Thread 中，而 WebSocket 推送是异步的 async/await，代码中巧妙使用了 asyncio.run\_until\_complete 来打通同步线程与异步网络层，确保高频仿真数据能实时推送到前端。

说明：以上代码片段与第六部分对应功能页面一一匹配，用于支撑“登录鉴权、告警管理、实时推送、仿真注入”等核心成果展示。

# 七、总结

## 1. 项目完成情况

本项目围绕“燃气安全监测与应急处置”的核心目标，完成了从需求分析、系统架构设计、数据库建模到前后端功能实现的完整闭环。管理端实现了态势总览、告警管理、工单闭环、设备管理、异味快报与管网仿真等关键模块；后端以 FastAPI 提供统一接口与鉴权，数据库采用 SQLite/ORM 建模，能够支撑课程设计阶段的快速迭代与演示验证。

## 2. 主要亮点与创新点

（1）“告警-工单”闭环：通过告警触发与工单流转，将发现问题、派单处置、回填结果串成可追溯流程。

（2）“虚实结合”的数据思路：既可接入真实设备数据，也提供仿真引擎与故障注入用于演示预测性风险。

（3）可视化表达：Dashboard 以地图叠加设备与管网，结合趋势小图与状态标识，增强信息密度与可读性。

（4）面向公众参与：异味快报提供居民线索上报入口，可与后台告警关联，提高发现覆盖率。

（5）周报归档：支持一键生成 PDF 周报，便于周期复盘与管理留痕。

（6）接入大模型：接入大模型API，助力根因分析与风险处置。

## 3. 不足与改进方向

（1）权限与审计粒度可进一步细化：目前以角色为主的权限控制可扩展为“菜单-接口-字段”级别，并完善操作审计维度。

（2）数据规模与性能：后续可引入更完善的时序数据存储与缓存机制，并在高并发场景下迁移至 PostgreSQL 等数据库。

（3）告警规则与模型：现阶段以阈值/简单规则为主，后续可引入更复杂的异常检测模型与多源融合策略。

（4）移动端与现场端：可扩展微信小程序/工人端，实现接单、到场打卡、拍照回传与离线处理等能力。

（5）地图与GIS能力：可叠加更真实的管线数据、分区管理、缓冲区分析与风险热力图，提升决策支持效果。

## 4. 总结

总体而言，本系统已实现课程设计要求的核心功能与可运行演示，形成了较完整的业务闭环与可扩展的技术架构。后续可在真实数据接入、规则模型增强、权限审计完善与移动端落地等方面持续迭代，使其更贴近实际燃气安全生产与城市治理场景。