北京海识软件有限公司

技 术 文 档

MyEMS 开源能源管理系统

架构设计文档

文档名称	MyEMS 开源能源管理系统架构设计文档
文档版本	6.9.0
对应系统版本	社区版 v6.9.0
编制日期	2026 年 10 月

目　　录

1 引言 1
1.1 编写目的 1
1.2 适用范围与读者 1
1.3 术语与缩略语 1
1.4 参考资料 2
2 系统概述 2
2.1 系统定位 2
2.2 功能域 3
2.3 系统边界 3
2.4 版本与许可 4
3 架构设计目标与原则 4
3.1 设计目标 4
3.2 设计原则 4
3.3 关键约束 5
4 总体架构 5
4.1 分层逻辑架构 5
4.2 各层职责 6
4.3 组件全景与协作关系 7
4.4 运行时形态 8
5 组件架构设计 9
5.1 协议采集服务 9
5.2 数据规范化服务 9
5.3 数据清洗服务 10
5.4 数据汇总服务 10
5.5 API 服务 10
5.6 展示层 11
5.7 数据库组件 11
5.8 组件间契约 11
6 数据架构设计 12
6.1 数据分层 12
6.2 数据库划分 12
6.3 领域模型 13
6.4 时序表设计 14
6.5 聚合表的命名规则 15
6.6 核算口径 15
6.7 容量与生命周期 16
7 数据流与处理流程 16
7.1 端到端链路 16
7.2 采集与原始入库 16
7.3 规范化与清洗 17
7.4 汇总与派生 17
7.5 查询链路 17

7.6 异常与重算 17
8 接口设计 18
8.1 接口风格 18
8.2 认证与鉴权 19
8.3 接口分组 19
8.4 报表与导出 19
8.5 集成建议 19
9 部署架构 20
9.1 部署单元与端口 20
9.2 典型单机拓扑 20
9.3 现场网络接入 21
9.4 关键配置项 21
9.5 扩展部署 21
10 非功能性设计 22
10.1 性能 22
10.2 可靠性与容错 22
10.3 安全性 22
10.4 可运维性 23
10.5 多语言与多组织 23
11 关键技术决策与权衡 23
12 二次开发与扩展指引 24
12.1 新增采集协议 24
12.2 新增分析报表 24
12.3 新增能源类型 25
12.4 新增业务对象 25
13 风险与演进建议 25
附录 A 默认端口与初始账号 27
附录 B 组件与数据库读写矩阵 28

1 引言

1.1 编写目的

本文档以 MyEMS 社区版的实际代码组织与数据库结构为对象，说明其总体架构、组件划分、数据组织方式、接口约定与部署形态，形成可供架构评审、实施部署、运维排障和二次开发共同使用的技术基线。

文档同时承担两项具体任务：一是把分散在官网功能说明、组件 README、数据库设计文档中的信息收敛为一张完整的架构视图，避免实施人员只能通过读源码理解系统；二是对架构的取舍与代价作出说明，使读者在扩展系统时知道哪些边界可以突破、哪些边界一旦突破会牵连多处。

本文档不替代《安装手册》与《API 手册》。凡本文表述与源码或官方文档不一致之处，以源码与官方文档为准，并在实施评审中记录差异。

1.2 适用范围与读者

本文覆盖社区版全部 11 个组件（含数据库）与 13 个业务库，不含企业版独有能力的具体实现。读者与建议阅读路径如下：技术负责人与架构师重点阅读第 3、4、6、11 章，用于判断技术选型与扩展边界；实施与运维工程师重点阅读第 5、7、9、10 章，用于部署、排障与容量规划；二次开发人员重点阅读第 5、6、8、12 章，用于接口对接与功能扩展。

本文的性能与容量结论为基于数据结构与服务形态的推算，未在真实生产负载下实测，相关段落均已标注“估算”。

1.3 术语与缩略语

表 1 本文使用的核心术语

术语	含义与在系统中的位置
能源分类	按能源种类划分的计量对象，如电、自来水、天然气、冷热水、蒸汽；承载计量单位、千克标准煤与千克二氧化碳排放两项折算因子，对应表 tbl_energy_categories
能耗分项	按用途划分的能耗归属，如空调用电、动力用电、照明用电；挂接在能源分类之下，对应表 tbl_energy_items
空间	组织能源数据的层级载体（园区—建筑—楼层—房间），通过 parent_space_id 自引用构成树，是绝大多数报表的筛选起点

术语	含义与在系统中的位置
计量表	能源计量的最小业务单元，绑定一个能源分类，其下的数据点承载实际读数；另有离线表（人工录入）与虚拟表（按表达式计算）两类派生形态
数据点	采集的最小粒度，包含模拟量、数字量、能耗量、文本量四种类型，按 object_type 与 object_id 归属到计量表或设备
数据源	一组同协议、同连接参数的采集目标，connection 字段以 JSON 保存地址与采集周期
组合设备	由多台设备与计量表组合而成的用能系统（如冷站），用于按系统口径核算输入与输出能耗
8760 模型	以一年 8760 小时为刻度的年度能耗模型，用于预测与预算编制，表名以 _8760 结尾
kgce / kgco2e	千克标准煤与千克二氧化碳当量，是能耗与碳排放的统一折算口径

注：本表为阅读辅助，不构成数据字典；字段级定义见第 6 章与官方数据库设计文档。

1.4 参考资料

[1] MyEMS 官方网站. https://myems.cn

[2] MyEMS 开源仓库（Gitee）. https://gitee.com/myems/myems

[3] MyEMS 社区版功能清单. https://myems.cn/docs/community

[4] MyEMS 系统介绍与兼容性清单. https://myems.cn/docs/intro

[5] MyEMS 数据库设计文档. https://myems.cn/docs/tutorial/database-design

[6] MyEMS API 手册. https://myems.cn/docs/tutorial/api

[7] MyEMS Docker Compose 安装指南. https://myems.cn/docs/installation/docker-compose

[8] MyEMS 管理员使用手册. https://myems.cn/docs/tutorial/admin-guide

[9] GB/T 23331-2020《能源管理体系 要求及使用指南》（等同采用 ISO 50001:2018），2020-11-19 发布，2021-06-01 实施，现行有效。

2 系统概述

2.1 系统定位

MyEMS 是面向用能单位的开源能源管理系统，参照 GB/T 23331-2020（等同采用 ISO 50001:2018）的能源管理体系要求设计，覆盖建筑、工厂、商场、医院、园区与能碳管理中心的电、水、气、冷、热等能源数据的采集、分析与报表，并支持能耗、成本、碳排放三条核算主线。官方资料显示其已积累上千个项目案例并通过 CMA 检测认证，代码以 MIT 协议开源，社区版要求保留版权标识与 LOGO。

与通用组态软件或纯数据平台相比，MyEMS 的重心不在画面组态，而在“计量关系”的建模：空间、计量表、设备、租户、门店、车间、组合设备七类对象共用同一套

聚合结果，使同一份能耗数据可以按组织维度、物理维度与商业维度同时展开。这一取向决定了它的数据模型比一般监测系统更重，也决定了后文的多数架构取舍。

2.2 功能域

按业务闭环划分，系统提供六个功能域。

（1）采集接入域：以 Modbus TCP、BACnet/IP、Siemens S7、OPC UA 四类工业协议接入现场仪表与控制器，并提供离线表与 Excel 导入通道，覆盖无法联网或需人工抄录的计量点。

（2）监测分析域：提供计量表实时分析与趋势、总分表平衡、批量分析、比较分析、电能质量，以及空间与设备的能耗分类、分项、负荷、效率、统计分析，并支持单位面积值与单位产品值两类强度指标。

（3）核算域：将能耗值按电价折算为成本、按排放因子折算为碳排放，输出报告期与基准期的对比结果；同时支持能耗计划、能耗基线与能耗预测。

（4）报表输出域：所有分析页可导出为真实 xlsx 文件，内含表格与曲线图、柱状图、饼图，而非 CSV 文本；并提供高级报表的模板化、定时生成与邮件分发。

（5）配置管理域：维护空间树、计量表与数据点、数据源与协议、设备与组合设备、成本中心、电价、工作日历、用户与权限等主数据。

（6）扩展域：社区版已提供基于 DeepSeek 与 OpenAI API 的人工智能分析结论与节能建议；企业版另提供光伏、储能、充电桩、微电网、虚拟电厂、设备控制、故障诊断与工单等能力。

2.3 系统边界

系统上游是现场仪表、PLC、楼宇自控系统与光伏储能等设备的监控接口，下游是浏览器、移动端、大屏与通过 REST API 集成的第三方系统。边界之内的职责是“取数、算数、看数、出报表”；边界之外不承担 SCADA 画面组态、不承担贸易结算级的计量检定、不内置消息中间件与流式计算引擎。

这一定位带来一个直接后果：系统对现场设备的实时性要求以“采集周期”为单位（数据源级配置，示例配置为 60 秒）；所有跨组件的数据交换以数据库表为介质，没有事件总线。理解这一点，是理解第 4 章运行时形态与第 11 章技术决策的前提。

2.4 版本与许可

仓库当前发行版为 v6.9.0，累计发布 49 个发行版。项目节奏为每月 1 个小版本、每年 1 个大版本，升级时需按 CHANGELOG 与升级文档执行库结构变更。

许可为 MIT，允许商用与修改，但社区版要求保留 MyEMS 相关版权信息及 LOGO，不得修改或删除品牌标识；更换品牌名称与标志为社区版明确不支持的能力。二次开发与贴牌交付时，这是必须先确认的合规边界。

3 架构设计目标与原则

3.1 设计目标

（1）多协议、多能源统一接入。异构仪表与控制器应在同一套数据点模型下被规范化，新增协议不应牵动上层逻辑。

（2）核算可追溯。任一能耗、成本、碳排放数字都应能回溯到具体的计量表、数据点与时间段，并保留数据质量标记。

（3）多维组织。同一份能耗需同时支撑空间、设备、租户、门店、车间、组合设备六类视角，且支持层级递归查询。

（4）低门槛交付。单机通过 docker-compose 即可完成部署，兼容多种国产操作系统与 MySQL 系数据库，适应私有化与离线环境。

（5）可扩展与可集成。对外以 REST API 与 JSON 导入导出为集成面，对内以表结构为扩展面，允许在不改动核心服务的前提下增加采集协议与报表。

3.2 设计原则

（1）一数一源。每个测量值只在一个数据点产生，之后的所有派生值均可由其重新计算得出；系统中不存在需要人工维护的第二份原始数据。

（2）算存分离、按用途分库。原始时序值、标准化能耗值、多维聚合值、派生核算值分置于不同数据库，避免单库膨胀，也使各库可按自身负载独立扩容。

（3）维度预物化。聚合结果按“对象×方向×分类×粒度”预先落表，用存储换查询响应，报表页因此不必在请求期做大规模扫描。

（4）时间统一 UTC。所有时间字段以 UTC 存储，展示层再转换为本地时间，使跨时区项目与跨库比对不产生歧义。

（5）配置驱动。采集目标、采集周期、点位地址、聚合范围均来自数据库配置，服务进程本身尽量无状态，便于容器化与重启恢复。

（6）可重算。任一计算环节都应支持按时间段重跑，并以 is_published、is_bad 等标记控制数据是否进入对外报表，从而在发现口径错误后不必重建整库。

3.3 关键约束

技术栈被明确限定：后端为 Python，API 采用 Falcon 框架并以 gunicorn 或 waitress 承载；前端为 React（Web）与 AngularJS 1.x（Admin）；数据库为 MySQL 系，官方兼容清单列出 MySQL 8.0/5.7、MariaDB 11/10、Percona Server、SingleStore 8.0/7.8、GoldenDB 3.0/2.0、OceanBase 4.2/3.2，企业版另支持 SQL Server、Oracle 与 GreatDB。

数据库兼容清单既是能力也是约束：为实现跨库兼容，系统只能使用各库共有的 SQL 子集，不能依赖特定厂商的窗口函数、JSON 函数或分区语法。这一约束直接解释了第 6 章中“用表结构而非 SQL 表达业务语义”的设计取向。

4 总体架构

4.1 分层逻辑架构

系统在逻辑上分为五层：采集层负责把现场信号变为库中的原始值；存储层按用途持有原始值、聚合值与派生值；计算层按时间节拍把原始值加工为多维口径的能耗、成本与碳排；服务层以 REST 接口对外提供查询、计算与导出；展示层按角色分为分析端与管理端。跨层的通用能力（权限、配置、多语言、日志、容器化、多库适配）作为横向支撑存在。

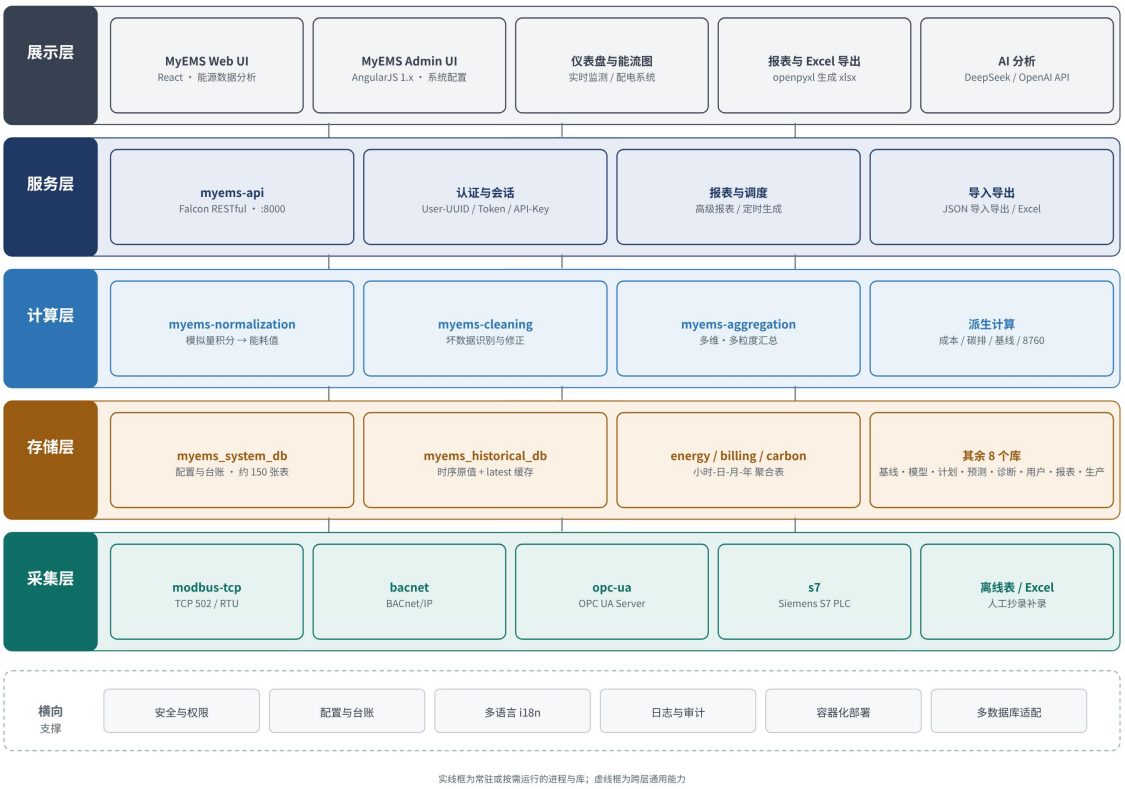


图 1 MyEMS 分层逻辑架构

分层并非纯逻辑划分，而是与部署单元基本对应：采集层与计算层的每个组件都是独立进程或独立容器，可以单独启停、单独扩容；服务层是唯一的对外 HTTP 入口；展示层是两组静态资源。这种“一层多进程”的组织方式，使单机部署天然形成进程级隔离，某个采集协议异常不会拖垮 API。

4.2 各层职责

采集层。每个协议一个独立服务，按数据源配置周期性读取寄存器或节点，将结果写入原始时序库。服务不解释业务含义，只做协议解析与类型转换；点位与业务对象的归属关系由配置库中的 object_type 与 object_id 决定。

存储层。以 13 个数据库承载不同性质的数据：配置库保存主数据，历史库保存原始时序值，能耗库、计费库、碳排库保存聚合结果，其余库保存基线、模型、计划、预测、诊断、用户、报表与生产数据。库与库之间不做外键约束，关联在应用层完成。

计算层。三个服务构成一条串行流水线：规范化把原始值转换为按小时的能耗值，清洗对异常值打标或修正，汇总再按七类对象与四种粒度物化。三者均以“时间段”为输入单位，因此具备重跑能力。

服务层。myems-api 是唯一的数据出口，负责鉴权、资源路由、报表生成与 Excel 导出；它不持有业务状态，所有状态落在数据库与会话表中，因此可水平扩展为多实例（共享同一数据库）。

展示层。Web UI 面向能源分析人员，提供图表、报表与导出；Admin UI 面向系统管理员，维护主数据与采集配置；两者均为静态站点，经 nginx 托管并反向代理至 API。

4.3 组件全景与协作关系

组件之间没有直接的进程间调用，全部通过库表耦合：采集服务写历史库，计算服务读历史库并写结果库，API 读全部库并对外呈现。配置库是被引用最多的节点，采集、计算、鉴权、报表四个环节都要读它。

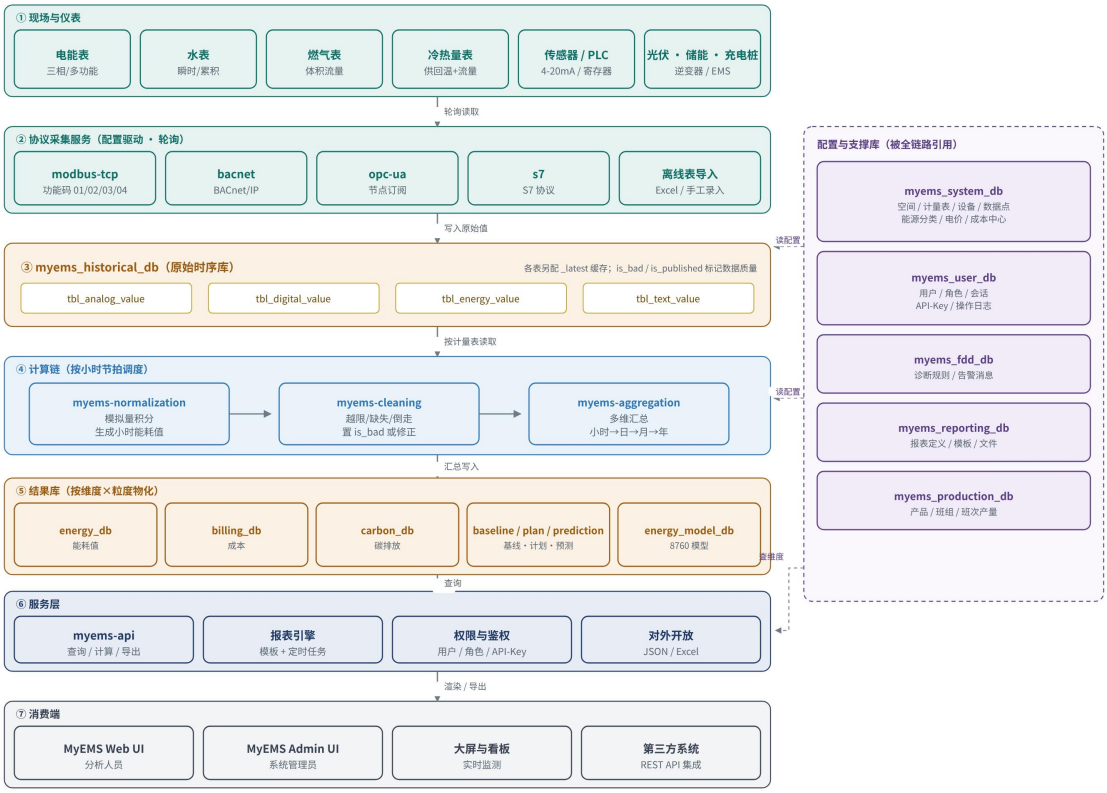


图 2 端到端数据流转与组件协作关系

这种“共享数据库”的集成方式，优点是部署简单、无中间件依赖、离线环境可运行；代价是组件间契约隐藏在表结构中，表结构变更即成为跨组件的不兼容变更。第 11 章将其列为一项需要长期管理的技术债务。

表 2 MyEMS 组件清单

组件	运行形态	主要技术栈	核心职责	端口
myems-database	初始化脚本集	MySQL DDL	创建 13 个库与全部表、初始数据	—
myems-api	常驻服务	Python、Falcon、gunicorn/waitress、openpyxl	REST 接口、鉴权、报表与 Excel 导出	8000
myems-admin	静态站点	AngularJS 1.x、nginx	主数据与采集配置管理	8001
myems-web	静态站点	React 16、ECharts、Chart.js、i18next、nginx	能源数据分析与可视化	80
myems-modbus-tcp	常驻采集	Python、modbus_tk、schedule	Modbus TCP/RTU 数据采集	—
myems-bacnet	常驻采集	Python、BACnet/IP	楼宇自控设备采集	—
myems-opc-ua	常驻采集	Python、OPC UA	从 OPC UA 服务器取数	—
myems-s7	常驻采集	Python、S7 协议	西门子 PLC 采集	—
myems-normalization	定时计算	Python、mysql-connector、schedule	原始值→小时能耗值	—
myems-cleaning	定时计算	Python、schedule	坏数据识别、修正与标记	—
myems-aggregation	定时计算	Python、mysql-connector	多维多粒度汇总与派生核算	—

注：端口为默认值，可在 nginx.conf 与容器编排中调整；采集与计算服务不监听端口，仅在容器网络内向数据库与现场设备发起连接。

4.4 运行时形态

组件按运行时特征分为三类，这一分类直接决定运维策略。

（1）请求响应型：myems-api 与两组前端。特征是耗时短、并发随用户数波动、故障可即时感知。运维重点是进程存活、连接池与慢查询。

（2）周期轮询型：四个采集服务。特征是长时间挂起、按数据源配置的间隔唤醒、单次失败不影响下一周期。运维重点是网络可达性与采集失败的连续性。

（3）定时批处理型：规范化、清洗、汇总。特征是按时段成批处理、对数据库压力集中、失败会累积滞后。运维重点是调度是否卡滞与数据是否按时段补齐。

三类组件的故障表现不同：API 故障立即被用户发现，采集故障表现为数据缺失且往往数天后才在报表中暴露，批处理故障则表现为聚合表停止增长。第 10 章的监控建议据此分层设置。

5 组件架构设计

5.1 协议采集服务

四个采集服务共用一套骨架：从配置库读取协议匹配的数据源，解析 connection 中的地址与周期，按点位地址逐项读取，转换后写入历史库对应表。服务之间不共享代码，但契约一致，因此新增协议时可以按同一骨架复制实现。

以 Modbus TCP 为例，数据源连接配置保存主机地址、端口与采集周期，点位地址配置保存从站号、功能码、寄存器起始偏移、寄存器数量、数据格式与字节交换开关，功能码支持 01、02、03、04 与 23，字节序由 Python struct 格式串描述，byte_swap 用于处理 32 位或 64 位数据的高低位交换。

数据源 connection：{"host":"10.9.67.99","port":502,"interval_in_seconds":60}

点位 address：{"slave_id":1,"function_code":3,"offset":0,
"number_of_registers":2,"format":"<f","byte_swap":true}

这一设计把“协议差异”压缩为两段 JSON，使现场调试可以在管理界面完成，而不必改代码。

采集服务的一个关键约束是：修改数据源或点位配置后必须重启对应服务。原因是配置在进程启动时载入内存，服务不做配置热更新。实施时应把“改配置—重启服务—核对数据”作为标准动作写入作业指导书，避免配置生效与否成为排障盲区。

5.2 数据规范化服务

规范化服务把原始点值转换为具有业务意义的能耗值。原始表按点存储瞬时读数，而报表需要的是“某计量表在某小时消耗了多少能源”，两者之间需要按计量表与点位的归属关系做聚合与差分。服务以小时为基本段落，读取时段内的原始值，计算后写入 tbl_energy_value，并同步更新对应的 _latest 表。

规范化是口径的第一次收口：此前的数据只是电压、电流、流量脉冲，此后的数据才具备能源分类与单位。因此，计量表与点位的绑定关系如果建错，错误会一路传导到成本与碳排，且不会在任何环节报错。实施阶段应把“表—点绑定核对”作为验收前置项。

5.3 数据清洗服务

清洗服务处理三类问题：越限值（超出量程或物理可能范围）、缺失值（时段内无数据）、倒走值（累积量出现回退，常见于换表或通信异常）。处理方式分标记与修正两种：可直接判定的异常置 is_bad 标记，使其不参与后续汇总；可推断的异常按规则修正。是否对外发布由 is_published 控制。

这一层的价值在于把“数据质量”显式化。多数能源系统的报表争议并非算法错误，而是坏数据未被隔离；MyEMS 通过两个布尔标记把质量问题留在数据中，使报表可以同时给出“值”与“可信度”。

5.4 数据汇总服务

汇总服务是数据放大的环节，也是系统性能的关键。它读取规范化后的能耗值，按七类对象（计量表、设备、组合设备、空间、门店、租户、车间，其中计量表另含离线表与虚拟表）与四种粒度（小时、日、月、年）分别计算，写入能耗库；再以同一份能耗值乘以电价写入计费库、乘以排放因子写入碳排库。

汇总的乘法派生是本系统的重要简化：成本与碳排不是独立采集的数据，而是能耗的线性变换。这使得电价或排放因子调整后，只需重跑对应时段即可修正历史，而不必重新采集。代价是分时电价、阶梯电价等非线性规则必须在汇总阶段解析，不能简单按单一系数相乘。

5.5 API 服务

API 服务基于 Falcon 实现，是全部数据的唯一出口。资源按名词复数组织，计量表、空间、设备、组合设备、数据点、数据源、电价与命令各有独立资源路径；层级关系用路径嵌套表达，例如组合设备下的计量表；非 CRUD 动作以子路径动词表达，如运行高级报表、下发命令、复制计量表、导出与批量导入。

服务同时承担报表与导出职责。Excel 导出由 openpyxl 在服务端生成真实 xlsx，包含表格与图表对象；高级报表支持模板、定时生成与邮件分发，其表达式以 JSON 保存在报表定义中，用于指定报表的空间范围与运行条件，运行时由服务解析。

服务本身不持有业务状态，会话落在用户库的 tbl_sessions 中，API 密钥落在 tbl_api_keys 中，因此多实例部署只需共享同一数据库。这一特性在多项目集中部署时尤为有用。

5.6 展示层

Web UI 基于 React 16 构建，图表使用 ECharts 与 Chart.js，地图使用 Mapbox GL 与 Leaflet，国际化使用 i18next，表格使用 react-bootstrap-table。工程通过 craco 覆盖默认的 CRA 配置，构建产物为静态文件，由 nginx 托管。地图能力需要在 src/config.js 中配置 mapboxToken 并将 showOnlineMap 置为 true，离线环境应关闭该开关。

Admin UI 基于 AngularJS 1.x，承担主数据维护。两套前端技术栈并存是历史沿革的结果：分析端持续演进，配置端保持稳定。这带来一个现实约束：新增配置类功能需按 AngularJS 写法实现，新增分析类功能按 React 写法实现，团队需要同时具备两种栈的维护能力。

值得注意的是，Admin UI 的配置项与后端的表结构几乎一一对应，界面字段即表字段。这降低了理解成本，也意味着表结构变更往往需要同步修改管理界面。

5.7 数据库组件

数据库不是被动存储，而是系统的集成总线。组件之间通过表约定协作：写入方保证字段语义与时间口径，读取方按约定解析。因此表结构在本系统中具有接口地位，任何字段改名、类型变更都应视为跨组件的不兼容变更。

所有库统一使用 utf8mb4 字符集、utf8mb4_unicode_ci 排序规则与 InnoDB 引擎；命名遵循库名 myems_{功能}_db、表名 tbl_{实体名}、字段小写下划线、索引 tbl_{表名}_index_{序号} 的规范。统一规范的价值在于：当需要跨库拼装数据时，字段名与含义可以预期。

5.8 组件间契约

(1) 时间。所有时间字段为 UTC，历史表用 utc_date_time，聚合表用 start_datetime_utc 表示时段起点。展示层负责转换为本地时间。跨时区项目的日、月边界因此以 UTC 切分，若业务要求按本地日历切分，需要在报表侧校正。

(2) 数值。统一使用 DECIMAL(21,6)，可容纳 6 位小数，足以表达 kWh、m^3、元与 kgCO2e 的常见精度，同时避免浮点误差累积。

(3) 标识。所有实体表含 id（BIGINT 自增）与 uuid（CHAR(36)）。id 用于库内关联，uuid 用于跨系统集成与导入导出时的对象对齐，避免自增主键在不同环境间冲突。

(4）质量与状态。is_bad、is_published、is_counted、is_enabled 四个布尔位分别控制数据可信度、发布状态、是否计入统计与配置启用状态；消息类表使用 status 字符串表达 new→sent→acknowledged 等流转。

6 数据架构设计

6.1 数据分层

系统数据按加工深度分为四层，每层的表结构、写入者与查询者都不同。

L0 原始点值。位于历史库，按点与时标记录测量值，是全部数据的唯一来源，只增不改（除清洗修正）。

L1 标准化能耗值。仍位于历史库的 tbl_energy_value，但已按计量表口径归集，具备能源分类与单位。

L2 维度聚合值。位于能耗库，按对象、方向、分类与粒度物化，是报表的主要读取对象。

L3 派生核算值。位于计费库、碳排库、基线库、计划库、预测库与模型库，由 L2 乘以系数或经算法生成。

分层的实际意义在于重算成本：口径变化若只涉及系数，重跑 L3 即可；若涉及计量表归属，需从 L1 起重跑；只有采集错误才需要回到 L0 补数。清楚每层边界，可以把一次口径调整的停机时间控制在小时级。

6.2 数据库划分

13 个库按用途划分为四组：配置组（system、user）、原始组（historical）、聚合组（energy、billing、carbon）、派生与支撑组（baseline、model、plan、prediction、fdd、reporting、production）。

表 3 数据库划分与读写关系

数据库	用途	规模特征	主要写入者	主要读取者
myems_system_db	配置与台账：能源分类、分项、空间、计量表、设备、组合设备、数据点、数据源、成本中心、电价、配电系统、能流图、命令与控制模式	表最多（约 150 张），数据量小、关系复杂	Admin UI 经 API	采集、计算、API
myems_historical_db	原始时序值与最新值缓存，含离线表、成本、修复、计划四类文件表	数据量最大，按点与时标增长	采集服务、导入	规范化、API 实时分析

数据库	用途	规模特征	主要写入者	主要读取者
myems_energy_db	多维能耗聚合，按小时/日/月/年物化	行数由对象与维度组合决定	汇总服务	API 报表
myems_billing_db	成本聚合，结构与能耗库一致，值已按电价折算	同上	汇总服务	API 成本报表
myems_carbon_db	碳排放聚合，结构与能耗库一致，值已按因子折算	同上	汇总服务	API 碳排报表
myems_energy_baseline_db	能耗基线，用于节能效果对比	较小	汇总/导入	API 分析
myems_energy_model_db	8760 小时年度能耗模型	每对象 8760 行	汇总/生成任务	API 预测与预算
myems_energy_plan_db	能耗计划与目标值	较小	计划文件导入	API 对比分析
myems_energy_prediction_db	能耗预测值	较小	预测任务	API
myems_fdd_db	故障诊断规则与多通道消息（Web、邮件、短信、微信）	小，但规则表达式复杂	规则引擎、API	API、消息分发
myems_user_db	用户、权限、会话、API 密钥、操作日志、通知	小，安全要求高	API	API 鉴权
myems_reporting_db	报表定义、模板文件、生成文件与邮件	小，含二进制附件	报表任务	API
myems_production_db	产品、班组、班次与产量，用于单位产品能耗	小	Admin UI、导入	API 强度分析

6.3 领域模型

领域模型的核心是一条“归属链”：数据源产生数据点，数据点归属计量表或设备，计量表归属空间并绑定能源分类，空间通过父指针构成树，树上再挂门店、租户、车间等业务子类。组合设备把多台设备与计量表聚合为一个系统。基础字典（能源分类、分项、成本中心、电价、产品、联系人）被上述对象以外键方式引用。

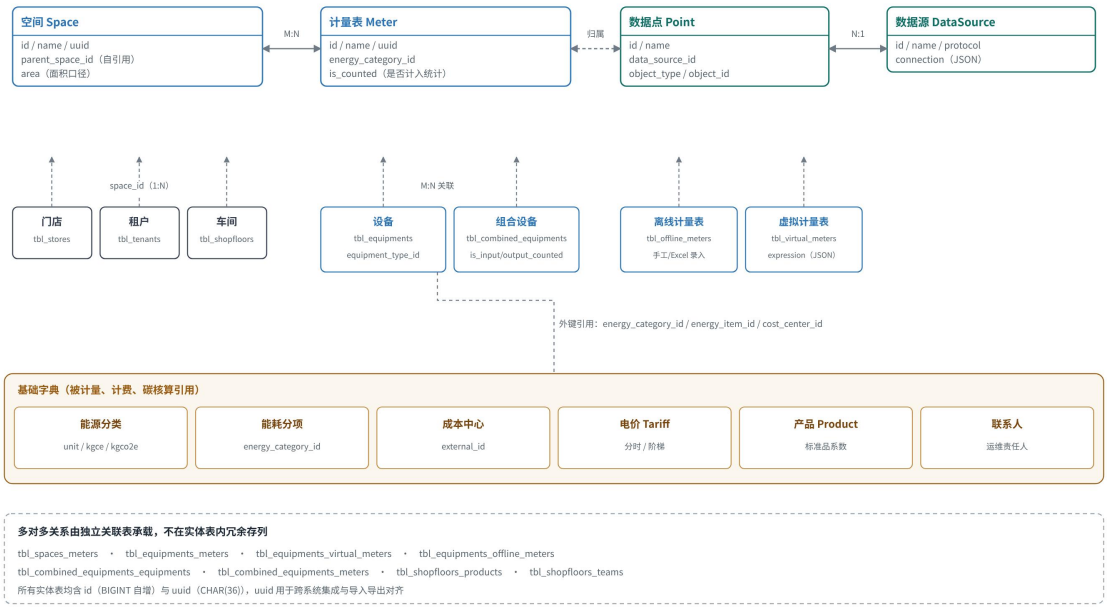


图 3 核心领域模型示意

模型刻意回避了两类常见做法：一是不在实体表内冗余存储关联列，多对多关系一律由独立关联表承载（如 tbl_spaces_meters、tbl_equipments_meters），使一个计量表可以同时服务多个空间视角；二是不为每类对象建独立的事实表族，而是复用同一套聚合表命名规则，使报表逻辑可以按对象类型参数化。

这一模型的代价是查询需要多跳关联。例如“某楼层本月照明用电”需要空间树递归、关联表展开、能源分项过滤三步，这也是系统大量使用预聚合而非实时 JOIN 的原因。

6.4 时序表设计

历史库的四类值表（模拟量、数字量、能耗量、文本量）结构高度同构：point_id + utc_date_time + actual_value，辅以 is_bad 与 is_published。每类表都配一张 _latest 表缓存最新值，用于实时分析页与设备状态刷新，避免在千万级历史表上做 MAX 聚合。

索引策略为复合索引 (point_id, utc_date_time) 与单列索引 (utc_date_time)：前者服务“某点某时段”的常规查询，后者服务“全库某时段”的批量加工。这一对索引分别对应在线查询与离线计算两类访问模式，是时序表设计中少见的双入口优化。

文件类数据（离线表文件、成本文件、修复文件、计划文件、报表文件与模板）以 LONGBLOB 直接存入数据库，字段含 file_name、uuid、upload_datetime_utc、status。优点是备份与迁移只需处理数据库，缺点是库体积随附件增长，实施时应约定单个文件上限与清理策略。

6.5 聚合表的命名规则

聚合表按 tbl_{对象类型}_{方向}_{分类维度}_{粒度} 命名，对象类型含 meter、offline_meter、virtual_meter、equipment、combined_equipment、space、store、tenant、shopfloor；方向含 input、output；分类维度含 category（能源分类）与 item（能耗分项）；粒度含 hourly、daily、monthly、yearly。

表 4 聚合表命名矩阵（示例）

对象类型	方向与分类维度	示例表名
计量表	无方向与维度	tbl_meter_hourly
离线表 / 虚拟表	无方向与维度	tbl_virtual_meter_hourly
设备	input 或 output / category 或 item	tbl_equipment_input_category_hourly
组合设备	input 或 output / category 或 item	tbl_combined_equipment_output_item_hourly
空间	input 或 output / category 或 item	tbl_space_input_category_daily
门店 / 租户 / 车间	input / category 或 item	tbl_tenant_input_item_monthly
光伏 / 储能 / 微电网	充电、放电、购电、售电	tbl_energy_storage_container_charge_hourly

需要说明的是，实际建表是上述组合的子集，而非完整笛卡尔积；系统按业务需要选择对象与维度，避免无效表膨胀。理解命名规则的价值在于：看到表名即可推断其维度含义，新增对象时也能按规则延展，而不必翻阅建表脚本。

6.6 核算口径

能耗折算为标准煤、成本折算为货币、碳排放折算为二氧化碳当量，三者共用同一份能耗值，差别仅在系数。千克标准煤与千克二氧化碳排放两项因子保存在能源分类上（tbl_energy_categories.kgce 与 kgco2e），成本系数保存在电价表。

产品内置的示例数据中，电的折算因子为 0.123 kgce/kWh 与 0.928 kgCO2e/kWh。这组数值属于示例口径，实施时必须按项目所在地的统计口径、电网排放因子与合同电价重新维护，否则所有碳排与成本报表都会系统性偏离。口径变更应以“配置修改 + 时段重算”的方式留痕，而不是直接改历史值。

成本核算支持分时与阶梯等复杂电价，规则在汇总阶段解析；碳排放支持按能源分类设置不同因子，但因子变更后的历史追溯需依赖重算，系统不内置因子版本表，这是实施中需要注意的边界。

6.7 容量与生命周期

原始时序库是唯一随点位与采集频率线性增长的库。以 3000 个数据点、15 分钟采集为例，单日新增 28.8 万行，年增约 1.05 亿行；按每行 40 字节（含索引）估算，年增约 3.9 GiB。若采集周期缩短至 1 分钟，年增将放大 15 倍，此时必须按时间分区或定期归档。

聚合库的规模不随点数增长，而由对象数与维度组合决定。以 500 个对象、4 种粒度、2 个分类维度估算，单维度年增约 438 万行（小时粒度），属于可控规模。这一特性意味着：系统扩容的主要压力来自采集侧，而非报表侧。

生命周期建议分层设定：原始值保留满足审计与重算所需的最短周期（常见 1 至 3 年），聚合值与派生值长期保留，文件类数据按项目约定定期清理。由于系统未内置自动归档策略，清理需以数据库作业方式自行建立，并在清理前确认重算窗口的需求。

7 数据流与处理流程

7.1 端到端链路

一条数据从现场到报表，经历七个环节：现场读取、协议解析、原始入库、规范化、清洗、汇总、查询呈现（详见图 2）。每个环节都有明确的输入与输出表，环节之间以时段对齐，因此任一环节都可以独立重跑而不影响上下游。

链路的时序特征值得注意：采集是分钟级，规范化、清洗与汇总是小时级，报表是请求级。三者节拍不同，意味着报表看到的最新数据通常滞后一个时段。这一滞后是设计使然，而非故障，实施时应在用户培训中明确，避免“实时数据不准”的误判。

7.2 采集与原始入库

采集服务按数据源配置的间隔唤醒，逐点位读取并写入历史库，同时刷新 _latest 表。写入以“点 + 时标”为键，重复写入同一时标的行为取决于实现是否幂等；从运维

角度应按“一个点位只由一个采集服务负责”的规则划分，避免双写造成同点同时标冲突。

7.3 规范化与清洗

规范化按小时段落读取原始值，结合计量表与点位的归属关系计算时段能耗，写入能耗值表。清洗紧随其后，对越限、缺失、倒走三类异常分别处理：能判定的打标，能推断的修正。二者的顺序不可颠倒——先归集再清洗，才能以计量表口径判断“这个值对这块表是否异常”。

7.4 汇总与派生

汇总阶段完成三件事：按对象与维度计算能耗值，按电价计算成本，按因子计算碳排。三件事共用一次能耗读取，分别写入三个库，保证同一时段三套数据的一致性。基线、计划、预测与 8760 模型在此之后生成，它们依赖历史聚合值，属于二级派生。

7.5 查询链路

用户请求经 nginx 转发至 API，API 校验身份后按资源路径访问对应库：配置类数据读配置库，实时值读历史库的 _latest 表，报表数据读聚合库。Excel 导出在服务端生成文件流直接返回，不落盘。

查询性能的关键在于“永远不要在请求期做重计算”。报表页之所以能快速响应，是因为它读取的是已物化的聚合表；一旦需求超出预置维度（例如任意时间窗的任意组合），响应时间将取决于历史库扫描，实施时应提前评估。

7.6 异常与重算

系统对异常的基本策略是“标记而非丢弃”，对错误的策略是“按段重算而非整库重建”。重算的边界由数据分层决定（见 6.1 节）：系数错误重跑派生层，归属错误重跑标准层，采集缺失则回到原始层补数后逐层重跑。

重算期间应通过 is_published 控制对外可见性，避免用户在重算过程中看到半成品数据。这一机制目前依赖人工操作，系统未提供统一的重算编排界面，是运维复杂度较高的环节。

8 接口设计

8.1 接口风格

API 遵循 REST 风格，资源用名词复数，层级用路径嵌套，非 CRUD 动作以子路径动词表达。响应统一为 JSON；列表接口不支持复杂查询语法，筛选以路径参数与少量查询参数完成。这一取向使接口易于被脚本与第三方系统调用，也限制了它在复杂检索场景下的表达力。

导入导出是接口体系中的特色能力：任意配置对象都提供 /export 与 /import 两个端点，导出为包含完整拓扑的 JSON（如组合设备导出包含其设备、计量表、离线表、虚拟表、参数与命令），导入时按 uuid 对齐对象。这使得跨环境迁移配置成为可能，也是多项目复制标准配置的主要手段。

接口从客户端经反向代理进入服务层、再落到数据库的完整链路，以及鉴权环节在其中的位置，如图 4 所示。

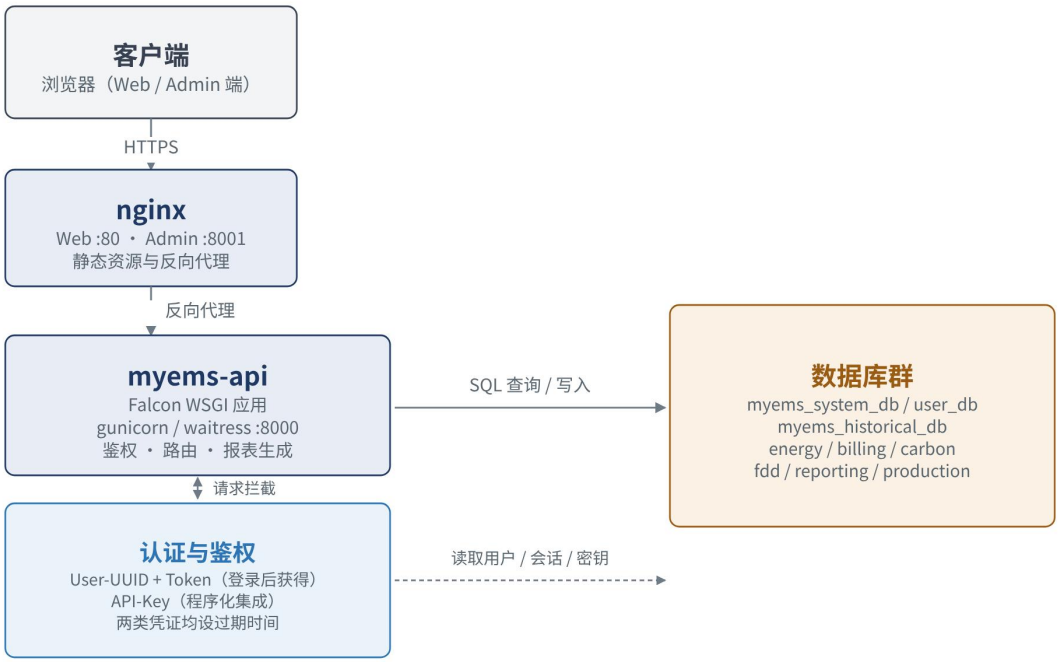


图 4 REST API 认证与调用链

8.2 认证与鉴权

所有请求须携带 User-UUID 与 Token 两个头部，Token 在登录后获得并带过期时间；程序化集成可使用 API-Key 替代，密钥同样设过期时间。三个凭证均由 API 在鉴权环节校验，失败返回 401。另有 API-Key 与 Token 并存的使用方式，用于同一账号下的多系统接入。

权限模型以用户为粒度：用户表含 is_admin、is_read_only 与 privilege_id，权限内容以 JSON 保存在 tbl_privileges 中。这意味着系统支持“管理员/只读/自定义权限”三档，但没有组织级的行级隔离；多租户场景下的数据隔离依赖空间树与视图，而非数据库级隔离。

8.3 接口分组

按资源性质可分为五组：核心配置（空间、计量表、设备、组合设备、数据点、数据源、成本中心、电价、联系人、命令、控制模式）、能源数据（各类分析接口与实时值）、报表（高级报表定义、运行、模板）、系统（用户、权限、版本、通知）、生产（产品、班组、班次、产量）。

接口的粒度与表结构高度对齐：一个资源通常对应一张主表加若干关联表，因此接口文档可以直接作为数据模型的补充说明。对接方在阅读接口时，应同时理解关联表的存在，否则容易在“创建对象”与“绑定关系”两步之间遗漏。

8.4 报表与导出

报表分两类：页面级报表由前端调用分析接口渲染，导出时由 API 生成 xlsx；高级报表由报表定义驱动，支持模板文件、定时生成、邮件分发，生成结果以文件形式存入报表库。前者面向即时查看，后者面向周期性报送，二者共用同一套聚合数据。

8.5 集成建议

对外集成建议遵循三条：一是为第三方单独创建只读账号与独立 API-Key，避免共用管理员密钥；二是在 nginx 层做限流与访问控制，API 自身不内置限流；三是批量取数优先使用聚合接口与报表导出，避免直接调用实时值接口做长周期扫描。

若需与集团级平台对接，推荐以“日粒度聚合 + 空间树”为最小数据集，既能满足管理口径，又能把接口调用量控制在可预期范围。

9 部署架构

9.1 部署单元与端口

系统提供容器化与原生安装两条路径：容器化用 docker-compose 一次拉起全部组件，适合快速交付；原生安装以 systemd 托管各 Python 服务，适合离线或需精细控制的场景。两条路径的组件构成一致，差别只在进程管理方式。

表 5 部署单元与默认端口

部署单元	默认端口	对外暴露建议	说明
myems-web（nginx）	80	对外	静态资源 + 反向代理至 API
myems-admin（nginx）	8001	对内或 VPN	静态资源 + 反向代理至 API
myems-api	8000	内网	REST 服务，仅供 nginx 与内部调用
MySQL	3306	内网	13 个库，禁止公网暴露
采集服务（4 个）	无	—	需可达现场设备网段（502 等）
计算服务（3 个）	无	—	仅需可达数据库

9.2 典型单机拓扑

典型部署由三部分组成：应用主机承载全部容器与共享卷，数据主机承载数据库，现场网络承载仪表与控制器。应用主机暴露 80 与 8001 两个端口，8000 与 3306 收敛在内网；采集容器额外需要一条到现场设备网段的路由。

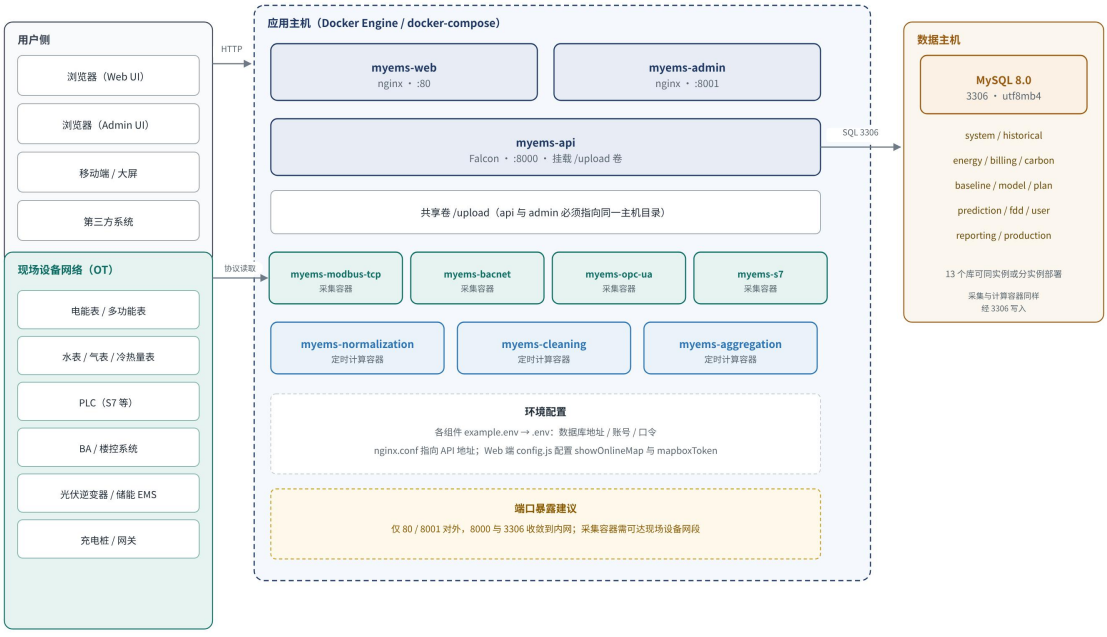


图 5 典型单机部署拓扑

共享卷 /upload 是容易被忽略的部署要点：api 与 admin 必须挂载同一主机目录，否则管理界面上传的模板文件在 API 侧不可见，报表生成会因找不到模板而失败。容器化部署时，两个容器的 volume 源路径必须一致。

9.3 现场网络接入

采集侧是唯一需要穿透到 OT 网络的组件。工程上建议为采集主机设置双网卡或独立采集网关：一侧连 IT 网络访问数据库，一侧连 OT 网络读取设备，两侧之间不做路由转发。若条件受限只能单网卡，至少应在交换机层用 ACL 限制采集主机仅能访问已登记的仪表地址与端口。

对于只有串口的仪表，需在现场侧部署 Modbus RTU 转 TCP 的网关，再由 modbus-tcp 服务采集。这类网关的并发连接数与超时设置直接影响采集周期，是常见的性能瓶颈点。

9.4 关键配置项

部署涉及四类配置：各组件的 .env（数据库地址、账号、口令，由 example.env 复制修改）；nginx.conf（指向 API 地址，web 与 admin 各一份）；Web 端的 src/config.js（showOnlineMap 与 mapboxToken，离线环境应关闭在线地图）；docker-compose 中的 upload 卷路径（Windows 与 Linux 路径写法不同）。

配置错误是部署阶段最主要的失败原因，且表现具有迷惑性：数据库地址错误会表现为 API 启动即退出，而 nginx 配置错误则表现为页面可打开但数据为空。建议在部署后按“API /version → 登录 → 打开任一分析页 → 检查一条采集数据”四步验证。

9.5 扩展部署

横向扩展遵循“按负载类型分别扩展”的原则。采集侧可按网段或协议拆分多实例，各实例读同一份数据源配置，按数据源归属划分职责；API 侧可部署多实例并由 nginx 负载均衡，共享同一数据库；计算侧通常无需多实例，瓶颈在数据库写入而非 CPU。

数据库侧的扩展空间取决于所选产品：MySQL 可通过只读副本分担查询，历史库可独立实例化以隔离采集写入压力。系统本身不内置读写分离逻辑，需要在 API 连接配置层实现，这属于需要改造的扩展点，而非开箱能力。

10 非功能性设计

10.1 性能

系统的性能策略可以概括为“把计算前移到批处理”。报表请求只做范围过滤与求和，不做跨表大规模聚合，因此响应时间与数据量弱相关，而与所选时段和对象数量相关。实时页则通过 _latest 表避免历史表扫描。

潜在瓶颈有三处：一是历史库写入，采集点位上万时单表写入会成为瓶颈，需要按点位分表或分库；二是汇总任务，小时任务若单次执行超过一小时将形成滞后累积；三是 Excel 导出，大范围导出在服务端生成图表对象，内存占用随报表规模上升。

上述判断为基于结构的推断，未经生产负载实测。实际容量的经验阈值应在项目试运行阶段通过采集行数增长与任务耗时曲线确定。

10.2 可靠性与容错

组件均为无状态或弱状态进程，崩溃后重启即可恢复；采集失败由下一周期自然补齐，计算失败可在下一轮按段重跑。数据可靠性依赖数据库自身的事务与备份，系统不提供跨库一致性保障，因此跨库数据可能出现短暂不一致。

单机部署存在明显的单点：数据库。容器化部署虽提升了组件隔离度，但未解决数据库高可用。对连续性要求高的项目，应在数据库层采用主从或集群方案，并把应用侧连接切换到可写节点。

10.3 安全性

系统内置的安全机制包括：口令以 salt + 哈希存储，不存明文；支持账户与口令的过期时间；支持登录失败次数限制；API-Key 与 Token 均设过期时间；关键操作写入操作日志表（含请求方法、资源类型、资源 ID 与请求体）。

需要项目侧补齐的部分同样明确：默认口令 administrator/!MyEMS1 必须在首次登录后修改；API 未内置限流，需在 nginx 层配置；传输层 TLS 由反向代理承担；数据库账号应按最小权限划分，避免应用账号拥有 DDL 权限。

从合规角度看，系统记录了操作日志与数据质量标记，具备基本的审计线索，但不提供完整的等保要求的日志留存、双因素认证与数据脱敏能力，涉密或强监管场景需另行加固。

10.4 可运维性

运维面的友好之处在于“配置即数据”：绝大多数运行参数存在数据库或 .env 中，备份数据库即备份了业务配置；日志为单文件输出（myems-*.log），排障时可快速定位。不友好之处在于缺少统一的任务编排与健康检查接口，采集与计算任务的执行状态需要从数据增长与日志中推断。

建议补充三类监控：采集连续性（按点位统计最近写入时间）、聚合时效（各粒度最新时段）、接口健康（/version 与关键查询的响应时间）。这三项覆盖了系统最主要的静默故障模式。

10.5 多语言与多组织

前端使用 i18next 管理文案，仓库 README 提供二十余种语言版本，说明项目具备国际化基础；后端以 UTF-8 存储多语言名称，界面语言由前端资源决定。多组织能力通过空间树与门店、租户、车间子类实现，属于“同一实例内的逻辑隔离”，而非独立实例或独立库的物理隔离。

这一设计在小规模多项目场景下足够，但在集团级应用中会带来权限粒度与数据量的双重压力：权限只能到用户级，无法限制“某用户仅可见某分公司”；所有项目共用一套聚合表，单表行数随项目数线性增长。

11 关键技术决策与权衡

架构的价值不在于选了什么，而在于清楚地知道为此放弃了什么。下表列出八项关键决策及其代价，供评审时逐项确认。

表 6 关键技术决策与代价

决策	选择	理由	代价与应对
数据组织	按用途拆分为 13 个库	隔离采集写入、聚合膨胀与配置读取三类负载，各库可独立扩容	跨库无法 JOIN，关联在应用层完成
聚合策略	按对象×方向×分类×粒度预物化	以存储换响应，报表页无需实时聚合	新增维度需同步建表
组件集成	以数据库表为集成总线，不引入消息中间件	部署简单、可离线、无额外运维组件	无事件溯源与重放能力
时间口径	全库统一 UTC 存储	跨时区与跨库比对无歧义，夏令时问题在展示层统一处理	日、月边界按 UTC 切分，与本地日历存在偏差
采集方式	周期性轮询，周期按数据源配置	兼容存量仪表与老旧 PLC，无需设备侧改造	实时性受周期限制
前端技术栈	分析端 React，配置端 AngularJS 1.x	分析端持续演进，配置端保持稳定，降低回归风险	双栈并行增加维护成本
数据库兼容	支持 MySQL 系多产品及部分国产库	适应私有化与信创环境，交付灵活	只能使用各库共有的 SQL 子集，无法利用厂商特性
部署形态	单机 docker-compose 优先	降低交付门槛，适合项目制实施	高可用与读写分离需项目侧另行设计

若把这张表压缩成一句话：MyEMS 用“结构简单、部署轻量”换取了“实时性与水平扩展能力有限”，用“预计算”换取了“口径变更的重算成本”。评审时应重点确认项目是否接受这两项交换。

12 二次开发与扩展指引

12.1 新增采集协议

新增协议的标准路径是复制现有采集服务骨架：以 myems-modbus-tcp 为模板，保留“读配置—连接—采集—写库—更新 latest”的主循环，替换协议实现与点位地址解析。随后在配置库的协议字典中登记协议名，在 Admin UI 增加对应表单，最后补充容器编排与 systemd 单元。

实施时有三个易错点：点位地址解析必须容错，单个坏点位不应中断整轮采集；写入必须遵循既有表结构与 UTC 口径，否则下游规范化读不到数据；服务需要按“配置改动后重启”的既有约定设计，不要引入半热更新，以免与其他采集服务行为不一致。

12.2 新增分析报表

新增报表的最小改动面是“API 资源 + 前端页面”：在 API 侧新增资源类，复用既有聚合表查询；在 Web 侧新增路由与图表组件。若报表需定期报送，则改为配置高级

报表：以 JSON 表达式描述范围，绑定模板文件，交由报表任务定时生成并邮件分发。

报表口径应尽量落在既有维度内。一旦需要“任意时间窗 × 任意对象组合 × 任意分项”的自由组合，就应新建对应的物化表或引入独立的分析库，而不是让 API 在请求期扫描历史库。

12.3 新增能源类型

新增能源类型是成本最低的扩展：在能源分类表增加一行，填写单位、千克标准煤与千克二氧化碳排放因子，随后建立对应计量表并绑定点位即可。聚合表按分类维度组织，不需要为新能源类型单独建表。

需要注意两点：因子必须按项目所在地口径维护，不能沿用示例值；若新类型需要新的计量单位或换算（如蒸汽按质量与焓值折算），应在采集或规范化阶段完成单位统一，避免把换算逻辑散落到报表中。

12.4 新增业务对象

新增业务对象（例如“园区中的某类功能区”）是成本最高的扩展：需要新增实体表与关联表、新增聚合表族、在汇总服务中增加计算分支、在 API 中增加资源、在两套前端中增加管理界面。其成本与“新增能源类型”不在同一量级，应作为版本级需求排期。

判断是否需要新增对象的标准是：该对象是否有独立的统计口径与权限诉求。若只是同一批计量表的另一种分组方式，优先通过空间树或关联表表达，而不是新增对象类型。

13 风险与演进建议

以下判断基于公开资料与结构分析，属于本文评估意见，不代表项目路线图。

（1）前端双栈的长期维护风险。AngularJS 1.x 已停止官方支持，配置端的安全补丁与浏览器兼容将逐步受限。建议新增配置功能优先在 React 侧实现，并以“配置端逐步迁移”作为中长期目标。

(2) 实时性天花板。轮询式采集与小时级计算决定了系统的实时能力上限。若项目要求秒级或分钟级闭环控制，应引入独立的实时通道（如 MQTT 或时序库），而不是改造现有链路。

(3) 聚合表膨胀与重算窗口。对象数量增长会带来表数量与行数的双重增长，重算窗口可能超过调度周期。建议引入按时间分区与增量重算，把“全段重跑”改为“按段补算”。

(4) 数据库单点与多租户弱隔离。集中部署多个项目时，数据库成为单点，权限也只能到用户级。建议在集团级场景中按项目分实例部署，或引入独立的元数据服务做行级隔离。

(5) 安全默认值与接口治理。默认口令、无限流、无内建 TLS 是交付阶段的常见缺口，应在实施清单中固化为必检项，避免依赖实施人员经验。

(6) 数据归档缺失。系统未内置历史数据归档策略，长期运行后原始库将持续膨胀。建议在项目侧建立归档作业，并在归档前明确重算窗口与审计要求。

附录 A 默认端口与初始账号

下表为社区版默认值，仅用于首次部署验证。生产环境必须在首次登录后修改口令，并按第 10.3 节调整暴露面。

表 A.1 默认端口与初始账号

项目	默认值	说明
Web UI 端口	80	分析端入口
API 端口	8000	REST 服务
Admin UI 端口	8001	配置端入口
数据库端口	3306	MySQL 默认端口
Admin 初始账号	administrator / !MyEMS1	首次登录后必须修改
Web 初始账号	administrator / !MyEMS1	与分析端权限模型共用用户表

附录 B 组件与数据库读写矩阵

下表汇总各组件对数据库的读写关系，可作为权限划分与备份策略的依据。

表 B.1 组件与数据库读写矩阵

组件	system	historical	energy	billing	carbon	user	reporting
采集服务	读	写	—	—	—	—	—
规范化	读	读/写	—	—	—	—	—
清洗	读	读/写	—	—	—	—	—
汇总	读	读	写	写	写	—	—
API	读/写	读/写	读/写	读	读	读/写	读/写
报表任务	读	读	读	读	读	—	写

注：本表为逻辑读写关系，“写”指该组件可变更对应库中的业务表；实际权限应按最小必要原则在数据库账号层收敛。