

内部技术文档 · 开发 / 运维 / 维护

凯迪 ERP 系统技术文档

架构 · 代码结构 · 数据模型 · 自动联动引擎 · 构建部署 · 运维手册 · 踩坑清单

Spring Boot 3.2.5 + Vue 3 + SQLite · 单 jar 部署 · 端口 8090

凯迪科技 · 2026

目录

一	技术架构总览
二	代码结构（前端 + 后端）
三	后端核心约定（ApiResp / 认证 / CORS / 异常）
四	数据模型（核心实体与互链）
五	自动联动引擎（TriggerRuleEngine）
六	通用台账机制（BizRecord）
七	前端机制（导航契约 / 路由 / 组件 / 设计规范）
八	构建与部署（精确命令）
九	运维手册（重启 / 日志 / 数据库 / 隧道）
十	坑与硬约束（必读）

技术架构总览

一套 ERP + OA 一体化平台。前后端打包成单个 jar，同一端口（8090）既提供 SPA 静态资源、又提供 REST API；通过 ngrok 隧道对公网演示。



技术栈

层	技术	说明
前端	Vue 3 + TypeScript + Vite + Element Plus 2.14.1	vue-router createWebHistory (base '/', 干净路径无 #); 图标只用 @element-plus/icons-vue
后端	Spring Boot 3.2.5 + Spring Data JPA	REST 接口前缀 /api/oa/*; 统一返回 ApiResp 信封
数据库	SQLite (org.hibernate.community.dialect.SQLiteDialect)	jdbc:sqlite:./data/oa.db; ddl-auto=update 自动建表加列
实时	WebSocket + yjs CRDT	文档协同实时编辑 (doccollab)
部署	单 jar (gradlew bootJar)	SPA 打进 classpath:/static, 与 API 同端口

关键点：前端 `npm run build` 的产物直接输出到 `oa-backend/src/main/resources/static` (`emptyOutDir=true`)，所以打 jar 时 SPA 被一起封进去——一个 jar 跑全栈。

代码结构

后端 oa-backend (com.kaidi.oa)

包	数量	职责
domain/	90	JPA 实体 (@Entity)。 Contract/Payment/SealUse/Invoice/Project/Bid/FormInstance/AutomationLog/BizRecord ...
repository/	90	Spring Data JpaRepository (findByXxx 派生查询)
web/	98	@RestController, REST CRUD + 业务动作
service/	8	WorkflowService (审批引擎)、TriggerRuleEngine (自动联动)、PaymentService、AuthService、NodeAssigneeResolver、NotificationService、FullTextSearchService、MentionService
common/	5	ApiResp、ApiException、GlobalExceptionHandler、NotFoundException、PasswordUtil (PBKDF2)
config/	5	AuthInterceptor、CorsConfig、JacksonConfig、SpaWebConfig (SPA fallback)、WebSocketConfig

前端 ofbiz-framework/plugins/modern-ui/app/src

data/oaModules.ts	导航契约 (顶部模块 + 子页面), 路由自动生成的唯一来源
router/index.ts	路由表: glob 自动发现 oa/pages/**, 无文件回退 OaGenericPage; 自定义路由手动加
oa/pages/<模块>/<页>.vue	各业务页面 (170 个)
oa/api/	HTTP 封装 (http.ts) + 各模块 API + settingList.ts (通用台账)
oa/engine/	表单引擎 (OaFormWindow/OaFormRenderer) + 模板
components/erp/	MasterDataPage、ErpDataTable、ErpPageHeader、ErpStatusTag、ErpDrawer ...
styles/tokens.css	设计令牌 --erp-* (颜色/字号/间距/圆角)

后端核心约定

统一返回信封 ApiResponse

```
public record ApiResponse<T>(int code, String message, T data) {  
    static ok(data) -> {code:0, message:"ok", data}  
    static error(code,msg)  
}
```

`code==0` 成功, 非 0 是业务错误。前端 http.ts 自动拆信封, `code!=0` 抛 OaApiError。

易错点: 业务错误返回的是 HTTP 200 + body 里 `code!=0` (如未登录 `code:401`)。只有受保护接口未带 token 时才返 HTTP 401。排查时别只看 HTTP 状态, 要看 body 的 code。

认证

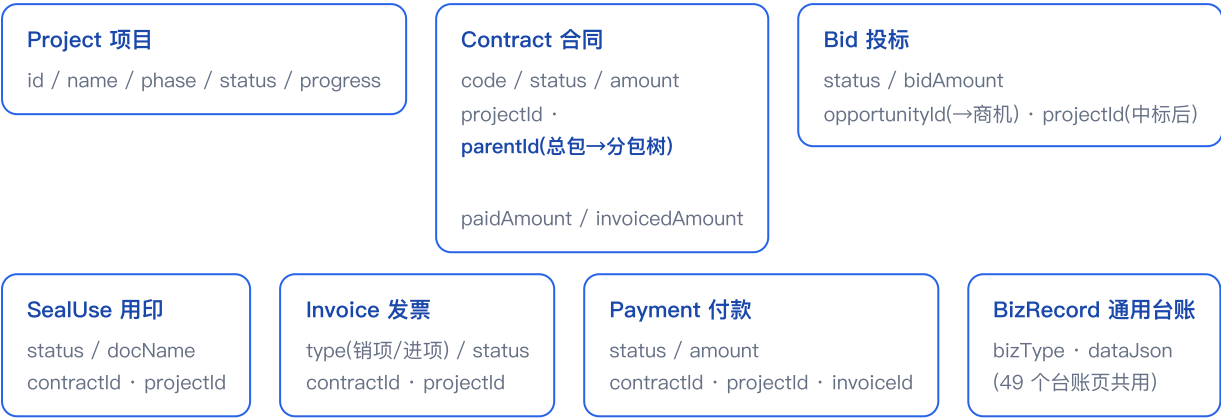
- 登录 `POST /api/oa/auth/login`, 请求体字段是 `loginName` / `password` (不是 `username`)
- 返回 token, 前端存 `localStorage` `oa.token`, 每次请求带 `Authorization: Bearer <token>`
- `AuthInterceptor` 守 `/api/oa/**`, 白名单: `login` / `session` / `logout`
- 密码 PBKDF2-HMAC-SHA256: `pbkdf2$120000$<saltB64>$<hashB64>`

CORS (隧道用)

- `CorsConfig.allowedOriginPatterns` 含 `"*"` ——隧道 Origin 才不被拦 (curl 不带 Origin 永远通, 浏览器才触发)
- 认证用 Bearer token 不用 cookie, 所以反射任意 Origin 安全
- 前端所有请求带 `ngrok-skip-browser-warning: true` 头, 绕过 ngrok-free 的浏览器拦截页

数据模型（核心实体与互链）

复杂领域走专用实体，靠外键字段互链；通用台账走 BizRecord。核心互链字段：



互链关系

关系	字段
项目 ← 合同/投标/发票/付款/用印	各实体的 projectId 指向 Project.id
合同 ← 用印/发票/付款	各实体的 contractId 指向 Contract.id
总包 → 分包（合同树）	Contract.parentId 自引用
商机 → 投标 → 项目	Bid.opportunityId / Bid.projectId

这些互链字段就是「项目360 一屏聚合」和「自动联动回写」的基础——加新关联只要补一个 xxxId 字段 + repo.findByXxxId。

自动联动引擎（TriggerRuleEngine）

系统「会自己跑」的核心。某审批事项**办结**时自动写下游单据/状态。

触发链路



`fire()` 按 `category + templateName` 关键词命中一条规则 → 写下游实体 + 一条 `AutomationLog`（数据中心可查）。

规则表

关键词	下游动作（含链式接力）
付款/报销/请款	生成「待付」付款单（钱权分离，不放款）
收款/到账/回款	生成「待开」销项发票草稿（用项目反查合同拿甲方）
用印/盖章	用印台账置「已用印」
验收/结项/竣工	项目置「验收」 + 链式：自动生成项目档案
立项	新建项目台账
供应商/准入	供应商置「合格」
合同	合同置「已生效」 + 链式：用章中心生成待用印单

另有 `BidController`（中标→建项目+总包合同）、`OpportunityController`（商机→投标）、`InvoiceController`（开票→回写合同已开票额）。

懂³的坑：真实表单按 `field.id`（英文 `contractName`）提交，规则按中文 `label`（合同名称）查找。
`enrichWithLabels()` 读模板 `schema` 把值按中文 `label` 补一份键——没有这步，自动联动在真实表单上全部静默失效。

加一条新规则

- ① 在 `fire()` 加 `else if (containsAny(tag, "关键词"))` 分支
- ② 写 `ruleXxx()` 方法（写下游 + `safeLog`）
- ③ 在 `ruleKeyFor()` 映射动作→稳定 `ruleKey`（幂等键，`(instanceId,ruleKey)` 只触发一次）
- ④ 注意顺序：具体关键词在前（如收款/验收要排在「立项」之前，避免「项目发起」类目误命中）

通用台账机制（BizRecord）

49 个补齐的部室台账页（运营工艺/HR资质/财务应收/行政后勤...）共用一套行级真后端，而非各写一个实体。

domain/BizRecord	行级实体：id + bizType（台账类型）+ dataJson（灵活字段）+ 时间戳
web/BizRecordController	REST：GET/POST /api/oa/biz/{type}、PUT/DELETE /biz/{type}/{id}，对外扁平化为 {id, ...字段}
前端 oa/api/settingList.ts	settingListStore(bizType, seed) → 调 /biz/{type}，喂给 MasterDataPage；首次空则把 seed 播种成真记录

每条台账记录 = biz_record 表一行真实数据，可查询、可统计，统一 REST。复杂领域（合同/付款/审批/用印）仍走各自专用实体 + 链式自动化，不混进 BizRecord。

前端机制

导航契约 oaModules.ts

顶部导航 + 每个模块的子页面 (key/label/kind/path) 都在这一个文件。路由从它自动生成，加页面通常只改这里。

路由自动发现

```
// router/index.ts
const pages = import.meta.glob('../oa/pages/**/*.vue')
// 有 oa/pages/<模块>/<key>.vue 就用它，没有则回退 OaGenericPage
// 带参/不进导航的页 (/orgview/dept、/collab/handle) 手动加进 routes[]
```

通用列表页 MasterDataPage

传 `columns / load / create / update / remove / createFields` 就得到「列表 + 搜索 + 新建 + 详情抽屉 + 增删改」。绝大多数台账页就是它 + 一个 store。

设计规范（硬约束）

- 颜色/字号/间距/圆角一律用 `tokens.css` 的 `--erp-*` 令牌
- 绝不用渐变色（纯色，蓝色主调）；绝不用 emoji（图标只用 `@element-plus/icons-vue`）
- 状态着色走 `ErpStatusTag`（语义映射：失败红/完成绿/进行中蓝/待办黄）

构建与部署（精确命令）

① 构建前端（产物自动进 static）

```
cd ofbiz-framework/plugins/modern-ui/app
npm run build # = vue-tsc --noEmit && vite build
# 输出到 oa-backend/src/main/resources/static (emptyOutDir)
```

② 打 jar（把 SPA 封进去）

```
cd oa-backend
JAVA_HOME=/Users/qiu/Desktop/ERP/.jdk/jdk-17.0.19+10/Contents/Home \
./gradlew bootJar
# 产物: oa-backend/build/libs/oa-backend-0.1.0.jar
```

③ 运行（单 jar 跑全栈）

```
JAVA_HOME=/Users/qiu/Desktop/ERP/.jdk/jdk-17.0.19+10/Contents/Home \
java -jar build/libs/oa-backend-0.1.0.jar
# http://localhost:8090 — SPA + API 同端口
```

改了前端必须先 **npm run build** 再 **bootJar**（jar 在打包时封入 static，光改前端不重打 jar 不生效）。改了后端只需 **bootJar**。

运维手册

重启（关键：只杀 8090 自己的进程）

```
PID=$(lsof -ti tcp:8090 -sTCP:LISTEN)
ps -p $PID -o command= | grep oa-backend-0.1.0.jar # 确认是本服务
kill $PID
# 再 nohup java -jar ... 启动
```

绝不用 `pkill java` / `killall java` ——会杀掉机器上别的 Java 进程（如 IDE/构建）。永远只杀 8090 LISTEN 的那个 pid。

日志	<code>/tmp/oa-backend.log</code> （nohup 重定向）
数据库	<code>oa-backend/data/oa.db</code> （SQLite 单文件，备份直接拷贝该文件）
建表/加列	<code>ddl-auto=update</code> 启动时自动同步（加实体字段会自动加列，不丢数据）
JDK	<code>/Users/qiu/Desktop/ERP/.jdk/jdk-17.0.19+10/Contents/Home</code>
公网隧道	ngrok http 8090（演示用）；隧道挂了重起 ngrok 即可，不用重启服务
账号	admin / zhangwei / lina ...，初始密码 123456

坑与硬约束（必读）

坑	对策
vite 白屏：manualChunks 单拆 vendor-vue，Vue 与耦合库（tiptap/yjs/vuedraggable）跨块 TDZ 「Cannot access 'X' before initialization」	Vue + 所有 Vue 耦合库必须同一个 chunk。构建绿≠能跑，构建后必须浏览器验 #app 已挂载
SQLite 多表启动崩溃	已配 hbm2ddl.jdbc_metadata_extraction_strategy=individually
SQLite 写争用：快速连续 PUT 会 500	操作间隔 / busy_timeout；批量写别并发轰
自动联动静默失效：表单 field.id 英文 vs 规则中文 label	TriggerRuleEngine.enrichWithLabels() 读模板 schema 补中文 label 键
排查方向错：只看 HTTP 状态	业务错误是 HTTP 200 + body code≠0，要看 body
误杀进程	重启只杀 8090 LISTEN pid，绝不 pkill/killall java
脚本批量生成页面	cwd 要在 oa/pages（否则相对 import 失效）；数组元素逗号分隔；加类型注解 :MdColumn[]/:MdField[]；MdField 无 'date' 类型用 text
ngrok-free 拦截页	所有请求带 ngrok-skip-browser-warning 头

铁律：① 绝不渐变色 ② 绝不 emoji（图标用 @element-plus/icons-vue）③ 重启只杀 8090 pid ④ 改前端必重打 jar ⑤ 构建后必浏览器验挂载。

一个 jar 跑全栈，靠流程引擎自己接力

前端 Vue 打进后端 jar，单端口 8090；90 实体 / 98 控制器；
自动联动引擎 + 通用台账 BizRecord 是两个最该先读懂的设计。