

微波研发部门研发管理平台

详细实施方案

文档编号: RDP-PLAN-2026-001

版本: V1.0

编制日期: 2026年2月20日

密级: 内部公开

目录

- 方案概述
- 总体架构设计
- 技术选型方案
- 各模块实施方案
- 开源项目选型与对比
- 部署架构方案
- 分期实施计划
- 风险与应对

1. 方案概述

1.1 建设思路

本方案采用"**微服务 + 模块化前端**"的架构策略，在统一的技术栈和基础设施之上，将9大功能模块设计为独立可部署的子系统。每个模块可由不同团队并行开发，通过统一API网关和消息总线实现模块间通信。

1.2 核心设计原则

- **离线优先 (Offline-First)**：所有依赖打包至局域网，零互联网依赖
- **模块独立 (Module Independence)**：每个功能模块独立仓库、独立部署、独立升级
- **开源优先 (Open-Source First)**：优先采用成熟开源项目，减少自研工作量
- **渐进交付 (Progressive Delivery)**：四期分步交付，每期可独立验收使用
- **本地集成 (Local Integration)**：深度融合Altium Designer、Obsidian、Zotero等本地工具

2. 总体架构设计

2.1 系统总体架构图

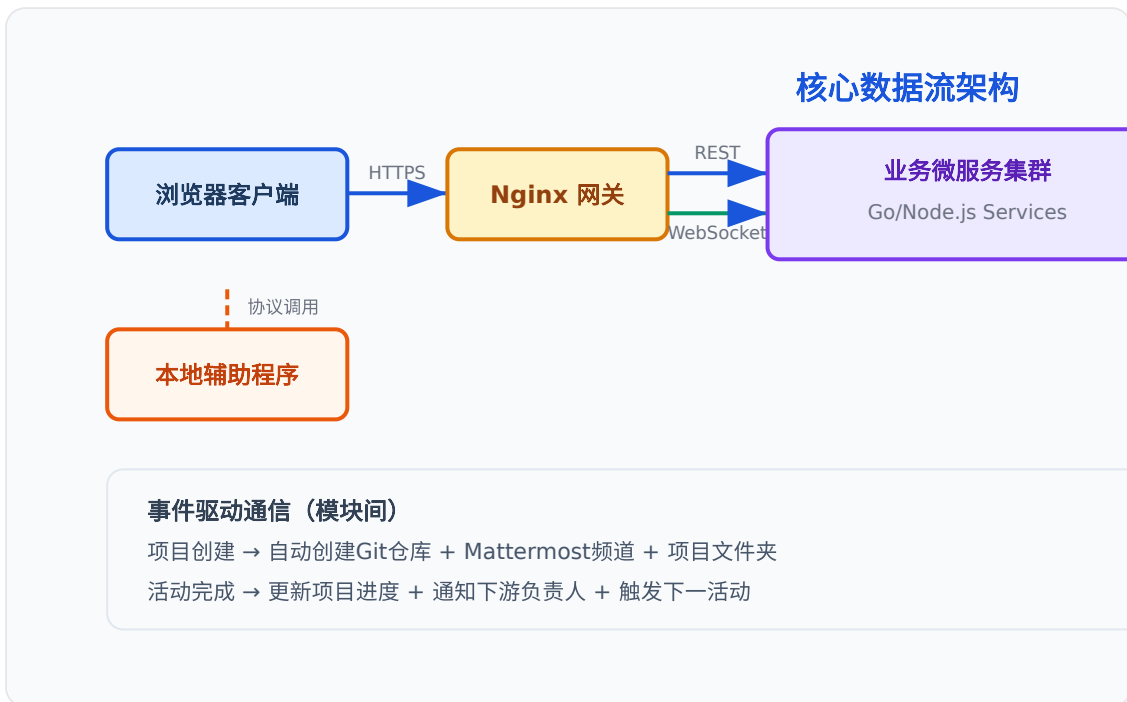


2.2 微前端架构说明

采用 **Webpack Module Federation**（或qiankun）实现微前端架构。主应用（Shell）负责门户框架、路由、认证和全局状态；各子应用（Remote）为独立的功能模块，可以独立开发、构建和部署。

关键设计决策：选择微前端架构而非单体前端的核心原因是——需求明确要求"多个团队独立开发模块"。Module Federation 允许每个模块作为独立仓库开发，运行时动态加载，无需统一构建。

2.3 数据流架构



3. 技术选型方案

3.1 核心技术栈

层次	技术选型	版本	选型理由
前端 框架	React 18 + TypeScript	18.x / 5.x	原department_homepage技术栈一致；生态最大；TypeScript保障代码质量
前端 UI	Ant Design 5 / Shadcn UI	5.x	企业级组件丰富；中文支持好；Ant Design Pro生态完善
构建 工具	Vite 5	5.x	department_homepage已采用；HMR极速；Module Federation支持
微前 端	Webpack Module Federation / qiankun	—	支持多团队独立开发部署
后端 语言	Go (Gin/Fiber)	1.22+	性能优异；单二进制部署；与Casdoor/Gitea同技术栈
后端 补充	Node.js (Express/Fastify)	20 LTS	流程引擎、文件处理等场景
关系 数据 库	PostgreSQL	16.x	功能强大；JSON支持好；开源社区活跃
缓存	Redis	7.x	会话管理；消息队列；热点缓存
搜索 引擎	MeiliSearch	1.x	轻量级；中文分词优秀；离线部署简单（Rust编写）

对象存储	MinIO	latest	S3兼容；轻量；适合局域网
Git服务	Gitea	1.22+	53.8K Stars；轻量级Git服务；Go编写；完善API
容器化	Docker + Docker Compose	24.x	统一部署环境；一键启停；服务隔离
反向代理	Nginx	1.25+	成熟稳定；SSL终止；负载均衡

3.2 开源组件集成矩阵

功能模块	集成方式	开源项目	Stars
用户认证/RBAC	独立服务 + API集成	Casdoor + Casbin	13K + 19.8K
即时通信	独立部署 + iframe/API集成	Mattermost	35.4K
Git版本管理	独立服务 + API调用	Gitea	53.8K
知识库	参考架构 + 自研	Wiki.js / BookStack	27K / 17K
全文搜索	独立服务 + SDK集成	MeiliSearch	52K
甘特图	前端组件嵌入	gantt-task-react	1K
流程引擎	嵌入式引擎	Flowable (Java) / 自研 (Go)	9K
组织架构图	前端组件嵌入	d3-org-chart	2.3K
图表可视化	前端组件嵌入	Apache ECharts	64K
Markdown编辑	前端组件嵌入	ByteMD / Milkdown	4K+

文件预览	前端组件嵌入	react-file-viewer / OnlyOffice	—
论坛	参考架构 + 自研	Flarum (参考)	16K
对象存储	独立服务	MinIO	52K

4. 各模块实施方案

4.1 门户界面模块

实施策略

基于 `Paniche/department_homepage` 的 React + Vite + TypeScript 技术栈进行扩展，增加工作台功能。采用微前端架构作为Shell应用，加载各子模块。

参考开源项目

Ant Design Pro 37K+

蚂蚁金服出品的企业级中后台前端解决方案，提供完整的工作台布局、权限管理、国际化等企业级功能。

github.com/ant-design/ant-design-pro

React

TypeScript

Ant Design

UmiJS

参考价值：工作台布局、全局搜索、消息通知中心的交互设计

vue-vben-admin 31.6K

Vue3 + Vite + TypeScript + Shadcn UI 构建的现代化后台管理模板，支持多种UI库切换。

github.com/vbenjs/vue-vben-admin

Vue 3

Vite

TypeScript

Shadcn

参考价值：微前端集成模式、多主题方案、Tab页签式工作区

vue-element-admin 90.3K

最流行的Vue管理后台模板，基于Element UI，内置i18n、权限管理、动态路由等企业级功能。

github.com/PanJiaChen/vue-element-admin

Vue

Element UI

Vuex

i18n

参考价值：中文企业级后台的标杆设计，权限模型、仪表盘布局

页面结构设计

```
src/shell/
├── layouts/
│   ├── MainLayout.tsx      # 顶部导航 + 侧边栏 + 内容区
│   ├── PortalLayout.tsx    # 门户首页布局
│   └── WorkbenchLayout.tsx  # 工作台布局
├── pages/
│   ├── portal/              # 部门首页（新闻、公告、荣誉）
│   ├── workbench/          # 个人工作台（待办、我的项目、消息）
│   └── search/              # 全局搜索结果页
├── components/
│   ├── GlobalSearch.tsx    # 全局搜索组件
│   ├── NotificationCenter.tsx # 消息通知中心
│   └── QuickActions.tsx     # 快捷操作面板
└── micro-fe/
    └── loader.ts            # 微前端子应用加载器
```

4.2 用户管理模块

实施策略

采用 **Casdoor**（13K Stars）作为身份认证和用户管理基础，结合 **Casbin**（19.8K Stars）实现细粒度RBAC权限控制。在此基础上自研GitHub风格的个人Profile页面。

Casdoor 🌟 13K

UI优先的IAM/SSO平台，内置组织/团队管理、多因素认证、130+社交登录支持。Go语言编写，可独立部署或嵌入。

github.com/casdoor/casdoor

Go React PostgreSQL OAuth 2.1

集成方式：独立Docker部署，通过OAuth2/OIDC与主平台对接；组织/角色数据通过API同步

d3-org-chart 2.3K

基于D3.js的高度可定制组织架构图组件，支持React/Vue/Angular，可展开折叠、缩放、自定义节点样式。

github.com/bumbeishvili/org-chart

D3.js React SVG

用途：组织架构树可视化，团队成员关系展示

个人Profile页面设计（类GitHub）



4.3 项目管理模块

实施策略

以自研为主，参考 **Plane**（45.8K Stars）和 **OpenProject**（14.4K Stars）的功能架构。甘特图采用 **ganttask-react** 组件。文件管理通过 **Gitea** API实现Git版本控制。

Plane 🌟 45.8K

现代化的开源项目管理工具，支持Issue跟踪、Sprint、甘特图、文档等。API-first设计，完美的自托管方案。

github.com/makeplane/plane

TypeScript

React

Django

PostgreSQL

参考价值：项目看板、Issue管理、Sprint面板的UI/UX设计；API设计规范

OpenProject 🌟 14.4K

企业级开源项目管理软件，被西门子、德铁等大型企业使用。强大的甘特图、BPMN工作流、时间跟踪等功能。

github.com/opf/openproject

Ruby

Angular

PostgreSQL

参考价值：甘特图交互设计、WBS分解、资源日历、MS Project导入导出

禅道 (Zentao) 🌟 1.6K (开源版)

国产开源项目管理软件，16年迭代，覆盖产品、项目、质量、效能管理，100万+团队使用。IPD版本直接支持IPD流程。

github.com/easysoft/zentaopms | zentao.net

PHP

JavaScript

MySQL

参考价值：IPD流程模板、中国企业项目管理最佳实践、批量导入导出

Gitea 🌟 53.8K

轻量级自托管Git服务，Go编写，单二进制部署。提供完整的Git仓库管理、API、WebHook等功能。

github.com/go-gitea/gitea

Go

Git

SQLite/PG

REST API

集成方式：每个项目自动创建Gitea仓库；通过API实现文件上传/下载/版本管理；自动commit

项目创建向导流程



4.4 项目开发模块

实施策略

自研为主，流程引擎参考 **Flowable**（9K Stars）的BPMN规范设计轻量级Go实现。流程可视化采用 **bpmn.js**。本地工具集成通过自研**桌面辅助程序**（Electron/Tauri）实现自定义协议注册和文件监听。

Flowable Engine 9K

高性能BPMN 2.0流程引擎，支持流程设计、人工任务、定时任务、事件驱动等完整 workflow 能力。

github.com/flowable/flowable-engine

Java

BPMN 2.0

Spring Boot

REST API

参考价值：BPMN流程建模规范、活动-网关-事件模型、流程实例管理API设计

n8n 175K+

可视化 workflow 自动化平台，400+集成，支持自托管。可作为流程开发类项目的开发环境。

github.com/n8n-io/n8n

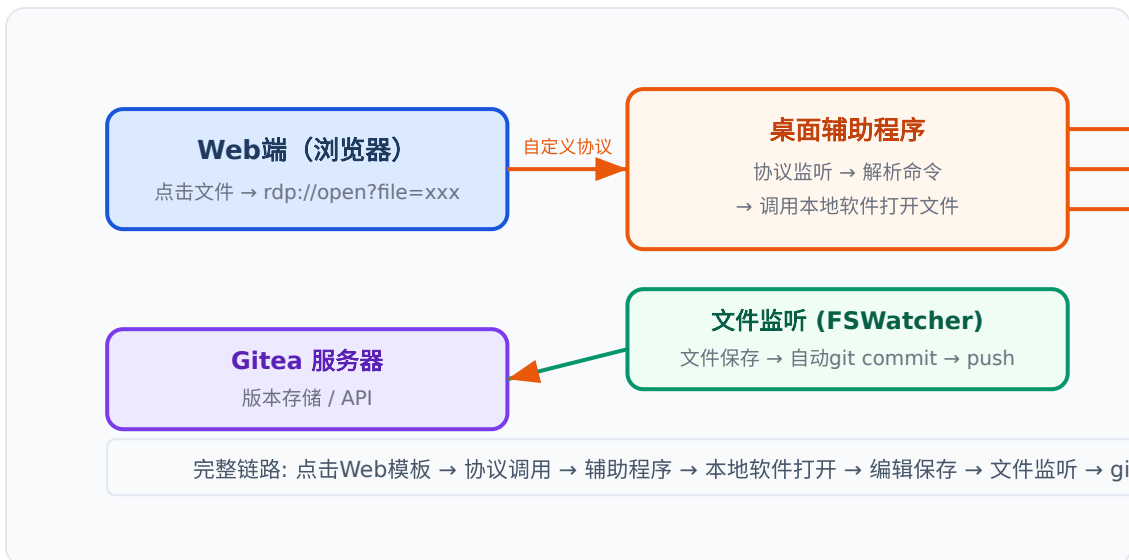
TypeScript

Vue

Node.js

集成方式：流程开发类项目直接嵌入n8n的流程设计器进行可视化流程编排

本地工具集成架构



4.5 产品/技术货架模块

实施策略

自研为主，产品展示UI参考电商平台的卡片式浏览和筛选交互。产品数据管理参考 **Part-DB**（电子元器件数据库）。技术树可视化采用 **ECharts Tree** 或 **D3.js**。

Part-DB 📦 1.5K

开源电子元器件数据库管理系统，支持参数化搜索、分类管理、数据手册关联、库存管理等。非常适合硬件研发的元器件货架场景。

github.com/Part-DB/Part-DB-server

PHP

Symfony

MySQL

参考价值：电子元器件分类体系、参数化搜索、数据手册管理的交互设计

4.6 知识库模块

实施策略

知识库底层架构参考 **Wiki.js**（27K Stars）或 **BookStack**（17K Stars），在此基础上自研Obsidian Vault同步和Zotero引用集成。全文搜索基于 **MeiliSearch**。

Wiki.js 📦 27K

功能最完善的开源Wiki引擎，支持Markdown/WYSIWYG编辑、Git同步、多语言、模块化存储、全文搜索等。

github.com/requarks/wiki

Node.js

Vue

PostgreSQL

Git Sync

参考价值：知识分类体系、Git同步机制、全文搜索集成、权限管理

BookStack 📦 17K

层级化知识管理平台（Shelves → Books → Chapters → Pages），非常适合标准规范、制度文件等结构化文档管理。

github.com/BookStackApp/BookStack

PHP

Laravel

MySQL

LDAP

参考价值：书架→书→章→页的层级结构，完美对应知识库的分类体系需求

Quartz (Obsidian发布) 📦 10K

将Obsidian Vault发布为静态网站，支持Wiki链接、Callout、嵌入、图谱等Obsidian特有语法。

github.com/jackyzha0/quartz

TypeScript

Markdown

Static Site

参考价值：Obsidian Markdown渲染引擎的实现参考，Wiki链接解析逻辑

4.7 即时通信模块

实施策略

直接部署 **Mattermost** 团队版（MIT许可），通过iframe嵌入或API集成到平台工作台。利用Webhook实现项目状态变更自动推送。

Mattermost 📦 35.4K

企业级开源协作平台，Go+React，支持频道、私聊、文件共享、700+集成、Bot框架等。被美国空军、CERN等机构使用。

github.com/mattermost/mattermost

Go

React

PostgreSQL

WebSocket

集成方式：Docker独立部署 → Nginx反向代理集成 → SSO与Casdoor对接 → Webhook接收项目事件

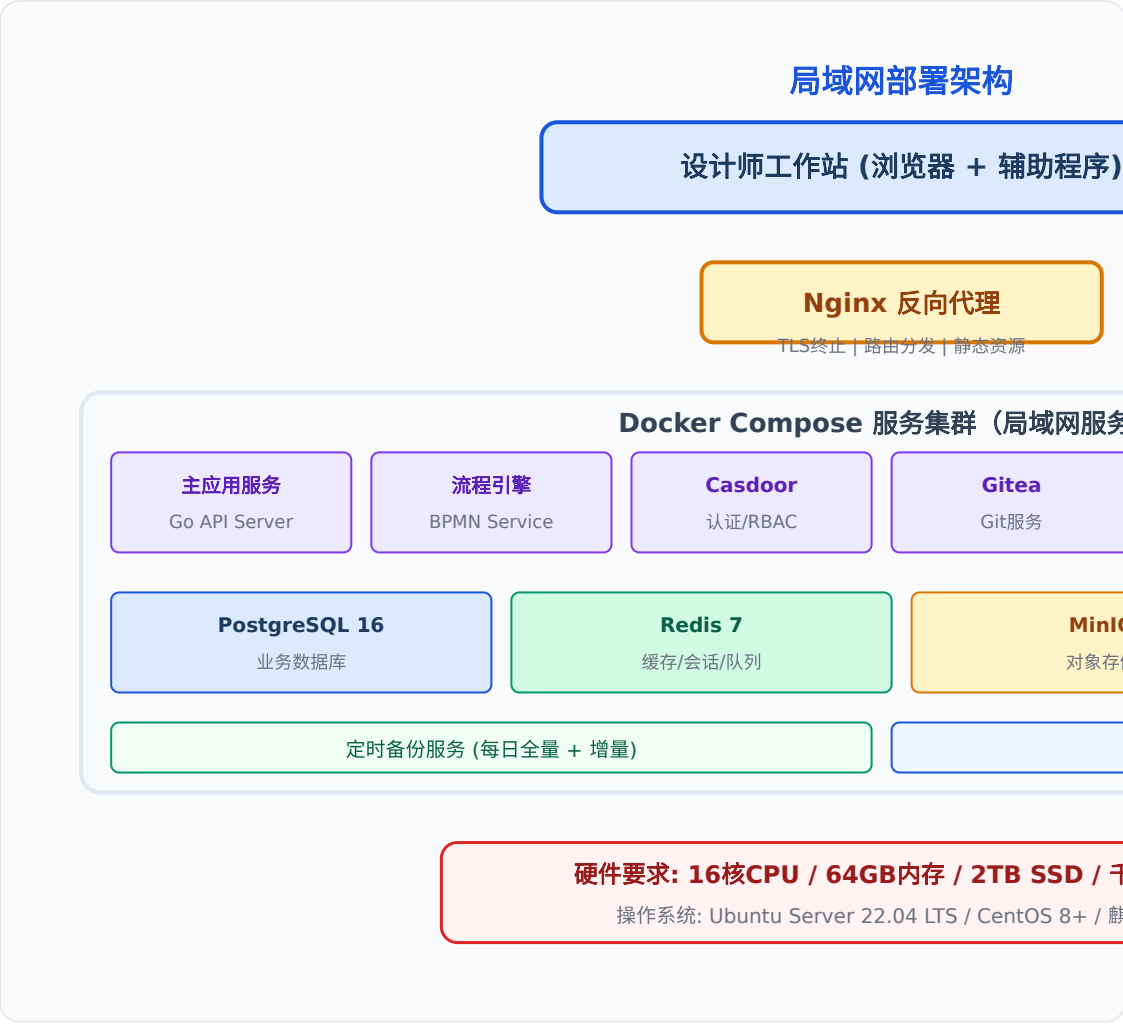
5. 开源项目选型对比总览

功能域	推荐方案	备选方案	决策理由
前端框架	React + Vite + TS	Vue 3 + Vite	与原department_homepage一致；React生态更大
UI组件库	Ant Design 5	Shadcn/Element Plus	企业级组件最丰富；中文支持最好
用户认证	Casdoor (13K)	Keycloak (32.9K)	Go技术栈统一；UI更友好；资源占用更小
权限引擎	Casbin (19.8K)	自研RBAC	成熟稳定；多模型支持；Go原生
Git服务	Gitea (53.8K)	Gogs (47.6K)	功能更完善；API更丰富；社区更活跃
即时通信	Mattermost (35.4K)	Rocket.Chat	Go+React技术栈；安全性强；API丰富
知识库	自研（参考Wiki.js）	BookStack直接集成	需深度定制Obsidian/Zotero集成
搜索引擎	MeiliSearch (52K)	Elasticsearch	轻量级；中文分词好；Rust编写性能高
对象存储	MinIO (52K)	SeaweedFS	S3兼容；部署简单；社区活跃
甘特图	ganttask-react (1K)	frappe-gantt	TypeScript/React原生；可定制性好

流程引擎	自研轻量BPMN(Go)	Flowable (Java)	Go技术栈统一；按需裁剪；避免Java依赖
图表库	Apache ECharts (64K)	Chart.js	雷达图、树图、热力图等全场景覆盖
组织架构	d3-org-chart (2.3K)	自研SVG	开箱即用；React集成好；交互丰富
论坛	自研（参考Flarum）	Discourse Docker 部署	轻量化；与知识库联动；技术栈统一

6. 部署架构方案

6.1 局域网部署架构图



6.2 Docker Compose 服务编排示意

```
# docker-compose.yml 核心服务定义
services:
  nginx:          # 反向代理 + 前端静态资源
    image: nginx:1.25-alpine
    ports: ["443:443", "80:80"]

  rdp-api:        # 主业务API服务 (Go)
```

```
build: ./services/api
depends_on: [postgres, redis]

rdp-workflow:  # 流程引擎服务 (Go)
build: ./services/workflow

casdoor:      # 认证服务
image: casbin/casdoor:latest

gitea:        # Git服务
image: gitea/gitea:1.22
volumes: [gitea-data:/data]

mattermost:   # 即时通信
image: mattermost/mattermost-team-edition:latest

meilisearch:  # 搜索引擎
image: getmeili/meilisearch:v1

postgres:     # 数据库
image: postgres:16-alpine
volumes: [pg-data:/var/lib/postgresql/data]

redis:        # 缓存
image: redis:7-alpine

minio:        # 对象存储
image: minio/minio:latest
volumes: [minio-data:/data]
```

7. 分期实施计划

7.1 四期实施路线图



7.2 团队分工建议

团队	人数	负责模块	技术栈要求
前端团队A	2-3人	门户Shell + 用户管理 + 项目管理前端	React + TypeScript + Ant Design
前端团队B	2-3人	项目开发 + 货架模块 + 知识库前端	React + TypeScript + ECharts
后端团队	3-4人	API服务 + 流程引擎 + 数据库	Go + PostgreSQL + Redis

	人		
集成/DevOps	1-2人	Mattermost/Gitea/Casdoor集成 + 部署	Docker + Nginx + Shell
桌面端	1人	辅助程序开发	Electron/Tauri + Rust/Node.js

8. 风险与应对

风险	等级	应对措施
离线环境依赖管理复杂	高	建立离线镜像仓库；所有npm/go依赖预下载打包；Docker离线镜像
微前端技术复杂度高	中	先以Monorepo开发，成熟后再拆分微前端；保留降级方案
本地软件集成兼容性	中	桌面辅助程序做好版本适配；提供手动上传兜底方案
开源项目版本升级风险	低	锁定经过验证的版本号；建立内部Fork仓库
数据迁移和历史数据	中	设计通用导入接口；提供Excel/CSV批量导入工具
用户接受度和培训成本	中	分批培训；提供操作视频和帮助文档；设置"意见反馈"入口

