无代码 / 低代码开发部署一站式平台

（集成 AIGC + MCP「Model Context Protocol」）

产品架构说明文档 V2.0

发布日期：2024-06-XX

────────────────────────────────

目录

1. 背景与目标

2. 关键概念与术语

3. 价值定位与典型用户

4. 总体架构总览

5. 核心子系统详解

6. MCP（Model Context Protocol）深度集成方案

7. AIGC Copilot 体系

8. 端到端应用交付流程

9. 技术选型参考

10. 安全、合规与治理

11. 性能与 SLO 指标

12. 版本规划与实施里程碑

13. 风险与对策

────────────────────────────────

1. 背景与目标

• 背景

– 企业对敏捷数字化的需求迫切，80% 的内部应用可视化即可完成。

– 大模型已具备原型->代码->测试->运维文档的全链路生成能力；但上下文碎片化严重，缺乏标准化注入途径。

– 2024-11-25，Anthropic 发布 “Introducing the Model Context Protocol”，提供跨系统上下文交换统一标准；为 AIGC 能力商品化、模块化提供可能。

• 目标

1. 通过无代码 / 低代码一站式平台，使 70% 业务需求“分钟级”上线。

2. 利用 MCP，将表单、流程、数据库、第三方 SaaS、日志、监控等上下文统一注入到大模型，提高 AIGC Copilot 的准确性 30% 以上。

3. 支持私有化、混合云部署；平台本身对外暴露 MCP Endpoint，方便外部 AI 代理调用。

4. 提供 Marketplace、插件机制，允许生态伙伴快速提供 Context Provider / Consumer。

2. 关键概念与术语

• AIGC：AI Generated Content，包括文案、代码、测试、脚本、图像等。

• MCP（Model Context Protocol）：模型上下文协议。

– Context Provider：暴露上下文（数据库记录、文档、API、实时事件等）

– Context Consumer：AI Model 或代理端，拉取 / 订阅上下文

– MCP Router：负责发现、路由、权限校验、缓存

• 无代码：100% 可视化（拖拽 / 配置 / NL Prompt），无手写代码。

• 低代码：≤20% 定制脚本 / 组件扩展。

3. 价值定位与典型用户

• CIO / CTO：统一 AI 应用开发、上下文治理、模型对接成本 < 30%。

• 业务分析师：自然语言 + 拖拽即可完成业务流程建模。

• 开发者：使用 MCP-SDK 将新系统或数据源 2-5 天内接入平台。

• SRE / 安全：一次性治理 Context 访问权限；模型调用审计可追溯。

4. 总体架构总览

┌──────────────────────────────┐

│ ⑦ 基础设施层：K8s / VM / GPU / 存储 │

│──────────────────────────────│

│ ⑥ 数据智能层：Data Lake、实时 Flink、元数据中心│

│──────────────────────────────│

│ ⑤ 集成层：API Gateway & iPaaS Connector │

│──────────────────────────────│

│ ④ 应用运行层：Micro-Frontend + Serverless │

│──────────────────────────────│

│ ③ AIGC & MCP 层： │

│ • MCP Router & Registry │

│ • Context Provider & Consumer SDK │

│ • Prompt Orchestrator / LLM Router │

│──────────────────────────────│

│ ② 设计层：无代码设计器 + 低代码 IDE │

│──────────────────────────────│

│ ① 体验层：门户、工作空间、Marketplace │

└──────────────────────────────┘

5. 核心子系统详解

5.1 门户 & 多租户中心

– SSO / OIDC，租户隔离；

– 计费、配额、运营统计。

5.2 业务建模套件

– 数据模型设计（ER）；

– 流程引擎（BPMN 2.0）；

– 页面 / 组件 / 仪表盘。

5.3 AIGC Copilot（详见章节 7）

5.4 MCP 子系统（详见章节 6）

5.5 运行时

– 前端：Micro-Frontend (Single-SPA / qiankun)；

– 后端：Knative Function / SpringBoot 微服务；

– 动态侧写（Hot-Reload）提高反馈速度。

5.6 DevOps & Observability

– GitOps（ArgoCD）

– 可观测三件套 + OpenTelemetry

– 自动扩缩容、金丝雀、回滚。

5.7 安全 & 合规

– DevSecOps 左移

– Context 级别访问控制 (ABAC)

– 审计追踪 + 合规报告。

6. MCP（Model Context Protocol）深度集成方案

6.1 架构组件

1) MCP Registry

• 记录 Context Provider/Consumer 元数据（schema、端点、Auth）

2) MCP Router

• 请求路由、版本协商、速率限制、缓存

3) Provider SDK

• GO / Java / Python / Node / Rust，多语言封装

• 自动生成 manifest（YAML / JSON）并注册到 Registry

4) Consumer SDK

• WebSocket & HTTP/2 流式拉取

• 支持 Pull & Push（订阅）模式

5) Security Gateway

• OAuth2 授权；Context 级 token scope；

• 审计 / 加密传输 (TLS-1.3)

6.2 数据流示例

a. 低代码页面保存 → 触发事件 “page.updated” 通过 Provider-SDK 发布到 MCP.

b. AIGC Copilot (Consumer) 订阅同租户 “page.\*” 事件 → 将页面 DSL、ER 图、用户角色等上下文注入 Prompt → 自动生成 UI 代码补丁。

c. 补丁经用户确认后合并 PR → ArgoCD 自动部署。

6.3 MCP 与平台 DSL 的映射

– Context Type = {PageDSL, BPMN, SQLSchema, LogEntry, Metric, DocChunk, SaaSRecord}

– Schema 使用 JSON-Schema 2020-12；

– 大容量二进制（如图像）存储至对象存储，MCP message 保存 presigned URL。

6.4 性能与缓存

– Router 内置 Redis / Memcached；

– 长连接 multiplex，支持批量上下文。

6.5 开发者体验

– “一键生成 Provider” CLI：扫描 OpenAPI / DB schema → 自动发布到 Registry；

– UI 控制台：可视化选择上下文注入模型的顺序与条件（if / else / Top-k 过滤）。

7. AIGC Copilot 体系

7.1 功能

• NL → 原型 / 表单 / 流程 / ER

• 语义 → SQL / API / 单元测试

• 语义 → Helm / Terraform

• 语义 → 观察性告警规则

7.2 Prompt Orchestrator

– Prompt Template + 动态插槽（由 MCP Router 获取的上下文）

– Chain-of-Thought / Tool-use planning

– 支持 Function Calling (OpenAI) 与 Toolformer 风格

7.3 模型路由

– 公有云 LLM：OpenAI GPT-4o, Claude, Gemini

– 私有化 LLM：Qwen-72B-Instruct, Yi-34B, Baichuan-13B

– 选择策略：成本、延迟、保密级别

7.4 反馈闭环

– 用户评分 + 隐式指标 (diff size, revert rate)

– RAG 微调 / Delta-fine-tune 定期迭代

8. 端到端应用交付流程

1) 业务人员在门户输入需求。

2) MCP Router 汇聚：

– 历史工单、行业知识库、权限模型 → 作为 Context。

3) AIGC Copilot 生成初版 UI/流程；用户拖拽微调。

4) 开发者按需编写 JS/Python 业务脚本。

5) Git Commit → CI → 单测（AIGC 自动生成并跑通）

6) GitOps 推送 → ArgoCD 部署到 K8s；Service Mesh 完成灰度。

7) 运行期日志 / 指标 通过 Provider-SDK 注入 MCP；

Copilot 监测性能下降自动建议缓存、索引优化。

9. 技术选型参考

• Portal：Next.js 或 Vue3 + Pinia

• 无代码内核：Alibaba LowCodeEngine / 拓端自研

• BPMN：Camunda 8（Zeebe）

• MCP 实现：

– Transport：gRPC-web + HTTP/2

– Registry：PostgreSQL + Redis

– SDK：基于 Protobuf / AsyncAPI

• LLM 调用：LangChain / LlamaIndex + Azure / Vertex / Bedrock

• K8s：v1.29 + Containerd

• Serverless：Knative 1.12 + KEDA 2.x

• Observability：Prometheus / Thanos / Loki / Tempo

10. 安全、合规与治理

• Zero-Trust 网络，mTLS 全链路加密

• Context 级别 RBAC + ABAC（Attribute: org, project, dataSensitivity）

• DLP：对出入模型的文本/文件进行敏感信息脱敏

• DevSecOps：SAST, SCA, IaC-Scan, RASP

• 合规：GDPR, ISO-27001, 等保 3.0

11. 性能与 SLO

• MCP Router P99 延迟 < 50 ms

• 平台崩溃恢复 (RTO) < 10 min

• AIGC Copilot 首次建议可用率 ≥ 85%

• SLA：99.95% 月度可用性

12. 版本规划

• V0.5（MVP，3 个月）：

– 无代码 + AIGC 基本生成功能

– MCP Router / Registry 雏形，支持页面 & 数据模型上下文

• V1.0（6 个月）：

– Provider Marketplace、Consumer SDK GA

– 私有化大模型适配

• V1.5（9 个月）：

– RAG + Online Learning

– Context 速率自动调整 (Adaptive Streaming)

• V2.0（12 个月）：

– 国际化、多云联邦

– 审计与合规自动报告

13. 风险与对策

1) MCP 标准变化快 → 抽象内部 Adapter 层，紧跟版本迭代。

2) 跨系统 Context 质量低 → 引入数据评分体系 + 自动清洗。

3) LLM 成本不可控 → 预算告警 + 动态路由至本地模型。

4) 安全泄露 → DLP、最小权限、定期渗透测试。

5) 生态冷启动 → 开源 SDK、插件分成、黑客松。

────────────────────────────────

本文件为内部架构蓝图，可根据业务优先级与资源灵活裁剪。