项目核心架构与
技术实施全景导航

目录

项目概述与定位

All-in-One 个人工作台定位
采用 Go 语言开发，编译为单二进制文件，部署零依赖，开箱即用，极简维护成本。

多维能力深度融合引擎
集成任务管理、个人经验库、AI 智能执行、任务调度器与网络代理，打破工具壁垒。

统一的数字生活入口
作为日常 Homepage、程序 Launcher 以及任务调度中心，构建个人生产力闭环。

精炼的技术基座
~1.46w 行 Go 代码，14 张 SQLite 轻量化表结构，91 个 HTTP 路由构建微内核架构。

全端环境支持
原生适配 macOS、Linux、Windows 三大桌面平台，保持一致的交互与数据体验。

极简三层架构设计

Presentation Layer: Web UI
现代化 Web 界面交互，响应式布局，支持多终端浏览器访问，提供直观的操作体验。

Core Layer: Single Binary
单进程集成所有业务逻辑，内置 HTTP 服务、AI 调度、代理转发等核心能力，无外部依赖。

Storage Layer: SQLite
嵌入式数据库引擎，文件级存储，免配置，易迁移，确保数据安全与便携性。

技术栈

后端核心架构 (Go Ecosystem)

Go 1.22+ 标准库

基于原生 net/http 与 mux 路由，拒绝重型框架，利用 Goroutine 实现高并发处理，确保服务核心轻量化与稳定性。

SQLite 纯 Go 驱动

采用 modernc.org/sqlite 驱动，完全消除 CGO 依赖，实现单文件交叉编译，适配多平台部署，简化运维复杂度。

WebSocket 实时推送

集成 gorilla/websocket 库，建立全双工通信通道，实现终端输出实时回传与指令即时下发，保障交互低延迟。

调度与终端支持

利用 cron/v3 实现进程内任务调度；基于 creack/pty 实现真 PTY 伪终端，完美兼容各类 CLI 工具的交互行为。

前端极简方案 (Vanilla Stack)

原生无框架开发

坚持使用纯 Vanilla JS 与 CSS，拒绝 React/Vue 等重型框架，减少运行时体积，代码结构清晰，维护成本极低。

零构建单页架构

采用“单 HTML + 多 View JS”模块化组织方式，无需 Webpack/Vite 等构建工具，文件即改即用，开发链路极简。

Go:embed 资源内嵌

利用 Go 1.16+ 的 go:embed 特性，在编译期将所有前端静态资源打包进二进制文件，实现服务端“单文件部署”。

F5 即时热更新

开发模式下，修改前端代码后只需按下 F5 刷新浏览器即可生效，无需重启服务或等待打包，极大提升开发迭代效率。

整体架构：7 Tab + 5 Widget

总览看板
全局数据总视图，核心指标可视化呈现

任务管理
任务全生命周期追踪，进度实时更新

经验知识库
沉淀历史方案，支持文档快速检索

自动化流
配置脚本任务，实现流程自动触发

系统配置
环境变量与权限管理，灵活适配需求

AI 对话
集成大模型能力，辅助决策与问答

代理服务
网关路由分发，保障接口安全稳定

WebSocket 实时推送
建立长连接通道，实现任务状态、通知消息的毫秒级实时同步，提升交互即时性。

HTTP REST API 交互
标准化接口设计，支持CRUD操作及复杂业务逻辑调用，确保系统间数据交互的可靠性。

快捷链接
聚合高频访问入口，一键直达核心功能与外部资源，提升操作效率。

目录快捷
可视化文件目录树，快速定位关键文档与项目资源，管理更直观。

Todo 清单
基于Markdown语法的轻量级待办管理，支持快速编辑与状态标记同步。

调度摘要
展示周期性任务执行计划，监控Cron表达式状态，防止任务遗漏。

最近执行
回溯任务执行日志，查看运行耗时与结果状态，快速排查异常问题。

数据基石：底层由 14 张 SQLite 表结构支撑，实现业务数据的高效存储与关联查询。

总览仪表盘

全维任务状态全景监控

覆盖任务全生命周期，实时追踪待认领、待执行、执行中、待交互、已完成及异常六大核心状态，状态流转清晰可溯。

核心调度器实时心跳监测

可视化呈现调度器“运行中/停止”关键状态，系统核心进程健康度一目了然，异常状态即时触发视觉预警，保障任务引擎稳定。

动态刷新与趋势洞察

最近执行列表按3秒间隔自动刷新，最新任务按时间倒序置顶展示；内置近7日任务执行统计图表，直观呈现业务量波峰波谷与执行效率趋势，辅助决策分析。

数据可视化看板预览

任务管理：AI 驱动的任务执行

结构化任务定义

构建标准化任务模板，涵盖标题、执行描述、优先级分级（P0-P2）及明确的验收标准，确保AI执行目标无歧义。

多环境AI执行引擎

深度兼容 Claude、CBC、Shell 等多种 CLI 工具，AI 自动解析自然语言指令并适配对应环境执行，降低人工操作成本。

全链路状态闭环

实现任务状态全生命周期管理：待认领 → 执行中 → 已完成 / 已归档，系统自动记录节点变更，可视化追踪执行轨迹。

LLM-as-a-Judge 智能评估

利用大模型对执行结果进行语义理解与自动打分，输出结构化的质量评估报告，替代人工复核，确保结果的客观性。

本地与Agent协同

轻量任务本地直连执行，复杂耗时任务智能分发至远程Agent集群，平衡资源利用效率。

知识资产双向沉淀

任务与经验库深度绑定，执行过程中的成功策略与避坑指南自动归档，构建可复用的企业知识库。

智能任务中心看板

刷新状态

任务 #001：API 接口自动化测试覆盖

P0 紧急　执行中　AI Agent-02

任务 #002：用户行为数据清洗与分析

P1 高优　待认领　Shell CLI

查看详情　立即认领　归档任务

经验库：最佳实践沉淀

多维定位：可复用的技术资产库
整合 SOP 标准作业程序、问题排查手册与技术知识库，沉淀团队过往实战经验，避免重复踩坑，实现知识资产化管理。

结构化分类：按技术域聚合内容
基于 Module 维度进行划分（如 xworkbench、linux-perf、system 等），建立清晰的层级目录，让技术经验归属一目了然。

全要素字段：还原真实技术场景
不仅包含文字描述，更关联场景背景、关键标签、工具用法、报错日志样本及核心代码片段，全方位还原问题与解决方案。

任务智能关联
执行任务时自动注入相关历史经验与解决方案，辅助即时决策。

多维快速检索
支持按技术模块、关键词、场景标签进行模糊与精确搜索，高效定位。

技术分类	核心关键词	典型适用场景
Linux-Perf 性能调优模块	CPU负载高 火焰图分析 sysstat工具	生产环境API响应延迟，需定位内核态与用户态进程资源占用瓶颈，快速生成分析报告。
XWorkbench 研发工作台	CI/CD流水线 环境隔离 依赖管理	多语言项目构建失败排查，解决依赖冲突与环境变量注入异常，标准化发布流程。
System-Admin 系统运维	日志轮转 权限管控 服务自启	处理系统盘空间告警，配置logrotate策略，以及突发故障后关键服务的自动恢复脚本。

自动化：Cron 定时任务

灵活的 Cron 语法

支持标准 5 字段（分/时/日/月/周）配置，同时内置 @every、@hourly、@daily 等快捷预设，满足从高频到低频的各类调度需求。

多类型任务执行器

内置 Shell 脚本、Claude AI 调用、CBC 容器执行三种核心执行器，无需额外适配即可对接底层系统命令与上层智能服务逻辑。

AI 模型精细化配置

每个定时任务可独立指定调用的 AI 模型版本（如 Claude 3 Opus/Sonnet）及参数（温度、TopP），实现任务与模型能力的精准匹配。

全状态调度管控

提供启动、停止、重载配置、立即触发执行等完整操作接口。支持动态调整任务状态，无需重启服务即可让配置变更生效。

任务调度监控看板（实时视图）

任务名称	Cron 表达式	类型	执行状态 / 下次执行
数据归档 每日数据备份	`0 0 * * *` 每天凌晨执行	Shell	运行中 下次: 2026-05-20 00:00:00
智能周报 生成业务总结	`@weekly` 每周一 09:00	Claude	待执行 下次: 2026-05-26 09:00:00

所有任务执行记录全量留存，关联 Execution ID 可追溯输入输出日志、AI 调用耗时及错误堆栈，确保调度过程可审计、可排查。

AI 对话：多 Tab PTY 终端

真 PTY 原生内核

基于 xterm.js 与 creack/pty 深度构建，实现与系统内核直连的原生终端交互，拒绝模拟层延迟。

5 路独立会话 Tab

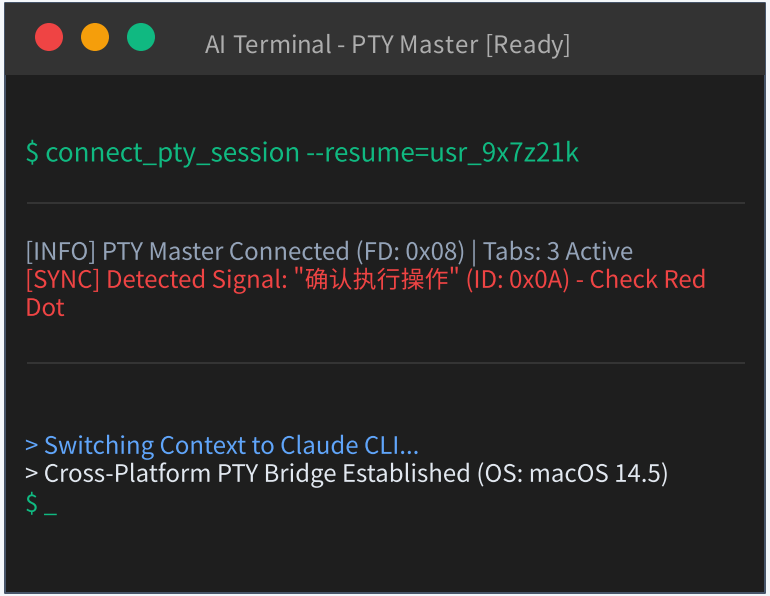
支持最多 5 个并行独立终端会话，会话间数据隔离，支持快速切换与后台任务挂起，提升操作效率。

多模型 CLI 无缝切换

内置 Claude / CBC / Codex / System Shell 等多环境入口，一键切换执行上下文，适配不同任务场景。

18 种信号自动预警

覆盖中英文场景下的 18 种关键确认信号，系统自动识别并触发红点提示，确保关键操作不被忽略。



支持 Resume_UUID 会话链延续，完美兼容 macOS/Linux 真 PTY 与 Windows ConPTY，打破平台壁垒。

真实环境模拟：终端界面展示了 PTY 连接状态、信号检测反馈以及 CLI 上下文切换的实时交互过程。

代理：命令执行 + HTTP 转发 + 远程 Agent

命令执行引擎

多环境原生支持
无缝执行本地 Shell 或 PowerShell 脚本，兼容类 Unix 与 Windows 系统环境。

完整输出捕获
精准返回 Stdout、Stderr 标准流及 Exit Code 状态码，便于脚本执行结果校验。

安全可控执行
支持自定义工作目录与超时限制，基于 relay.api_key 实现严格的 API 鉴权访问。

HTTP 智能转发

全方法请求透传
完美支持 GET、POST、PUT、DELETE 等 HTTP 协议全方法，透明转发请求至目标 URL。

自动化鉴权集成
无需手动配置，系统自动携带认证头信息，简化下游服务的身份验证流程。

全链路监控统计
实时记录并统计总请求数、成功响应数及失败请求数，可视化展示转发健康度。

远程 Agent 集群管理

生命周期自动化
Agent 自动注册、心跳保活与离线剔除，确保集群节点状态实时同步。

任务自动认领机制
支持 auto_claim 模式，空闲 Agent 自动拉取并执行任务，最大化资源利用率。

资产与能力管控
统一管理 Agent 版本、执行能力集，并支持绑定目录快捷方式快速访问资源。

工具赋能：提供一键生成 Linux 调用脚本功能，轻松实现跨机器、跨网络的代理服务调用，降低部署与集成门槛。

系统配置：统一配置管理

单一权威配置源

采用 config.json 作为系统唯一的配置入口，移除复杂的 fallback 优先级链设计，确保配置读取逻辑极简、透明且无歧义，消除多源配置冲突风险。

全维度用户偏好定制

支持深度个性化配置：涵盖默认终端选择、CLI执行优先级设定、AI自治功能开关及Todo数据存储路径映射，让系统行为完全贴合用户操作习惯。

标准化部署环境管控

集中定义终端类型枚举值、注册可用大模型列表、统一管理第三方代理密钥。通过中心化部署配置，实现多环境参数的一致性与安全性管控。

模型能力解耦配置

严格分离执行模型(default)与评估模型(eval_default)的配置节点，支持双模型独立调度。允许在不同任务场景下灵活切换模型，互不干扰。

config.json Structure

```
{
  "system":{
    "default_terminal":"bash", // 终端配置
    "ai_autonomy":true
  },
  "models":{
    "exec":"gpt-4o","eval":"claude-3"
}
```

支持 UI 可视化编辑与 JSON 批量导入导出

代码组织：清晰的分层结构

CMD / Server：应用入口与交互层

main.go & pty.go
项目唯一的程序入口，聚合所有 HTTP Handler 路由；封装 Unix PTY 伪终端实现，处理终端交互底层逻辑。

index.html & Static
SPA 单页应用的 HTML 入口；Static 目录存放编译后的前端资源，包含通用 API 工具函数、UI 组件与 7 个核心视图模块。

该层级专注于**对外交互与资源展示**，不涉及核心业务逻辑，确保应用入口简洁，前端资源与后端服务解耦，易于部署与维护。

INTERNAL：核心领域与业务逻辑层

Backend (数据层)
封装 14 张核心数据表的 Repository 接口，负责数据持久化、查询与事务管理，隔离底层数据库差异。

Config (配置中心)
统一管理系统配置的加载、解析与热更新；支持环境变量、配置文件与远程配置源的无缝切换。

Executor
子进程执行器，安全隔离并调度外部命令，处理 IO 重定向与生命周期管理。

Hub & Relay
WebSocket 消息中枢与流量代理层，负责实时消息广播、连接管理与数据转发。

Eval & Sched
AI 代码评估器与任务调度器，实现复杂逻辑编排、定时任务与智能分析功能。

设计优势：通过**CMD 层收敛入口**，INTERNAL 层封装核心逻辑，严格遵循**Clean Architecture**原则，确保高内聚低耦合。

核心链路：AI 任务执行流程

01. 用户交互触发
用户在前端操作界面点击「▶ 运行」按钮，直观发起任务执行的核心指令请求。

02. 后端接口分发
系统网关接收请求，标准化转发至 POST /api/tasks/{id}/run 接口进行参数校验与分发。

03. 上下文 Prompt 构造
智能融合 Task 基础元数据与历史 Experience 经验库数据，动态生成高质量的执行上下文 Prompt。

04. 执行命令动态构建
通过 runner.BuildCommand 适配器，自动生成适配 Claude、CBC 或 Shell 环境的可执行命令。

05. 子进程同步执行
调用 executor.Run 启动隔离的子进程环境，同步执行构建好的业务逻辑，确保稳定性。

06. 执行状态持久化落库
实时更新 executions 表记录，状态流转：running → success / failed / timeout，留存全量日志。

07. WebSocket 实时状态广播
服务端通过 WebSocket 向 executions 订阅频道推送状态变更消息，实现数据实时触达。

08. 前端动态渲染展示
前端监听器捕获广播消息，即时更新任务列表与日志详情页，提供无感知的实时交互体验。

评估链路独立机制

解耦执行与评估
评估逻辑走独立的 evaluator.Evaluate 流程，不阻塞主任务执行，保证系统吞吐量。

模型灵活配置
支持独立配置评估专用模型（如 GPT-4o、Claude 3），针对不同任务类型定制评估标准。

评估结果独立归档，支持与主任务结果关联分析，形成完整的任务闭环数据。

数据模型：14 张 SQLite 表

核心业务表 (8张)

tasks (任务主表)
承载24个核心字段，定义任务生命周期状态，驱动整体业务流转的基石。

experiences & executions
经验库沉淀历史方案，执行记录表全链路追踪任务运行轨迹与输出详情，实现有据可查。

scheduled_tasks & agents
支持定时自动化执行，管理远程Agent节点，构建分布式任务处理能力。

evaluations & shortcuts
AI量化评估任务质量，整合web链接与目录快捷方式，提升操作便捷性。

关联与审计表 (4张)

task_experiences (多对多关联)
建立任务与经验的关联关系，打破数据孤岛，构建可复用的知识网络，赋能后续任务决策。

task_events (审计日志)
详尽记录任务状态变更、关键操作触发点及时间戳，确保操作留痕，满足审计与排查需求。

task_comments & execution_comments
提供多层次的协作沟通机制，分别针对任务本身和单次执行过程进行评论反馈，沉淀隐性知识。

系统元数据表 (2张)

app_meta (应用元数据)
存储应用的核心配置参数、运行环境信息、版本号及关键开关状态。作为系统的"配置中枢"，确保应用启动与运行的一致性，支持灵活的参数动态调整，无需修改代码即可适配不同部署环境。

skill_versions (技能版本管理)
管理 AI 技能模型的迭代历史、版本号、发布时间及内容摘要。实现技能的灰度发布与回滚机制，保障系统功能升级的稳定性，同时记录技能演进轨迹。

采用轻量化 SQLite 分层设计，结构清晰解耦，在保证数据一致性的同时，最大化提升系统运行效率与可维护性。

开发指南：如何扩展功能

01. 新增 API 端点

在**cmd/server/main.go**中注册 HandleFunc 路由；若需数据交互，在 APIServer struct 中添加 repo 依赖；编写具体的 handleXxx 业务逻辑函数；前端通过 view JS 调用**fetchJSON**完成接口通信。

02. 新增前端 Tab 页

在**index.html**中添加页面容器 div 及导航菜单项 nav-item；新建**static/js/views/xxx.js**实现视图逻辑；在 api.js 的 switchTab 方法中补充新 Tab 的加载与切换控制逻辑。

03. 变更数据库 Schema

修改**internal/backend/repo.go**的 InitSchema；新表使用 CREATE TABLE IF NOT EXISTS 保证幂等；新增字段采用 migrateTasksColumns 模式，通过 PRAGMA 探测字段存在性后执行 ALTER TABLE 进行无损迁移。

04. 扩展系统配置项

在**internal/config/config.go**结构体中添加新字段，并同步更新 DefaultConfig() 中的默认值；前端通过 GET /api/config 获取配置，PUT /api/config 提交更新，实现配置的持久化与实时生效。

代码示例：任务执行核心

```
// 1. 构造执行命令，支持多种配置选项
cmd, err:= runner.BuildCommand(
"claude", "sonnet", "", prompt,
runner.WithActionReport(),
)
// 2. 执行任务，流式回调处理实时输出
result, err:= executor.Run(ctx, execDir, cmd,
func(chunk []byte) { /* WS 实时推送前端 */ }
)
```

多 CLI 类型适配

统一接口兼容 claude、cbc、shell 三种核心执行环境，通过参数化配置实现底层调用的无缝切换。

流式实时响应

利用 onChunk 回调函数捕获实时输出，通过 WebSocket 即时推送到前端，保障用户体验的连贯性。

全链路状态管理

构建 Running → Success/Failed/Timeout/Cancelled 完整状态机，确保任务状态流转清晰可追溯。

兜底与异步评估

Ctx 超时后自动兜底写入 stderr；任务评估逻辑独立解耦，支持异步调用与模型的灵活配置。

参考代码位置：cmd/server/main.go - handleTaskRun 函数实现核心调度逻辑。

总结与展望

核心优势回顾

极简部署体验
单二进制文件分发，零系统依赖，仅需配置可选 AI CLI 即可快速启动服务。

全平台无缝覆盖
完美适配 macOS、Linux、Windows 三大主流操作系统，统一交互逻辑，降低学习成本。

架构与交互升级
模块化设计易于扩展，配合实时 WebSocket 推送，实现毫秒级流畅交互反馈。

AI 原生自治闭环
内置任务执行、效果评估与自主决策闭环，不仅是工具，更是智能协作者。

未来演进方向

构建多 Agent 协作网络
突破单 Agent 能力边界，实现任务拆解、多角色分工与结果聚合的群体智能协作。

开放插件生态系统
提供标准化 API 接口，支持第三方技能与业务逻辑无缝接入，丰富平台服务能力。

多端适配与可视化
完善移动端/PWA适配，升级数据看板，以图表化方式直观呈现 AI 执行轨迹与成效。

深度团队协作赋能
引入权限管理、任务共享与实时协作功能，让 AI 能力服务于整个团队的工作流。

感谢聆听，欢迎交流！