阿创

CRM系统v3.0产品需求文档

CRM 系统 v3.0 产品需求文档（PRD）

逆向梳理版 — 从项目起源到当前实现的完整技术复盘

版本：v3.0（草案，供架构重构参考）

创建：2026-05-26

作者：阿创（基于与老张的全部沟通 + 代码逆向梳理）

目录

1. 文档概要

1.1 编写目的

本 PRD 从项目第一天沟通到当前 v2.0 实现进行完整逆向梳理，产出目的：

1. 全景回顾 — 完整记录老张与阿创自 2026-04-05 起的全部 CRM 相关决策、讨论、演进
2. 架构冻结 — 将当前分散在 4 个 skill + 1 个独立项目 + 多个部署中的 CRM 能力统一归档
3. 问题诊断 — 系统分析 CRM 表现不太稳定的根因
4. 重构蓝图 — 为 v3.0 迁移到公司高可用架构提供完整参考

1.2 当前版本 vs 目标版本

对比维度	v2.0（当前）	v3.0（目标）
架构	FastAPI + 单机	公司高可用架构
数据库	SQLite WAL	待定
部署	uvicorn --workers 4	高可用集群
分布	分散在 4 个 skill + 1 项目	统一服务
API 文档	无标准化	待定
监控	无	待定
容灾	systemd 3s 重启	待定

1.3 阅读对象

- 老张（最终决策者）
- 马坚（架构设计/后端实现）
- 阿创（开发文档统一归档）

2. 项目全历程时间线

还原从第一次讨论到当前实现的全过程，包含每次关键决策和方向调整。

Phase 0：概念期（2026-04 初）

老张初始需求：做一套客户经营的东西 — 基于社交网络的客户关系管理

第一次沟通要点（2026-04-05）：

#	决策/讨论	说明
1	战略方向：双向奔赴的智能体经济	供给侧智能体经营智能商品，需求侧智能体是个人AI助理
2	"能聊天不UI，能文件不数据库"	技术选型原则（后期被实际数据量打破）
3	身份(role) vs 商品关系(relation) 分离	role 由 user-onboard 管，relation 由 client-journey 管
4	数字员工本身就是智能商品	两个成交模式：卖给人 / 卖给人的AI
5	AI助手模板集中管理	放阿创服务器，不分散到目标服务器

Phase 1：client-journey 技能诞生（2026-04-05）

产物：skills/client-journey/ v1.0 → v2.0

核心设计：一个人可以和多个商品有不同关系。例如张总可同时是 medclaw 的 customer、bossclaw 的 prospect、healthclaw-personal 的 user。

文件存储（不是数据库）：

- other-contacts/{contactId}/profile.json
- product_relations.jsonl（append-only）
- interactions.jsonl
- recommendations.jsonl
- notes.md

关系阶段（9 阶段）：

stranger → icebreak → exploring → trusting → first_deal → nurturing → bonded → dormant → lost

Phase 2：支付与商业化的到来（2026-04 中旬）

背景：需要实际收费，必须引入数据库。

关键决策链：

1. 要收钱了 → 需要记录谁买了、谁推荐了
2. 阿创实现 order_db.py（SQLite）
3. 引入推荐关系（两级分销）
4. 引入微信支付 / JKZL 公司支付对接

时间	决策	影响
04-10	保证金模式 ¥599（加好友满 60 人退款 ¥600）	设计 trial_lifecycle + user_accounts
04-12	两级分销合规：直接 20%/25%，间接 10%	referrals 表 + commissions 表
04-13	33 天提现等待期	withdrawable_at 字段
04-15	推广资格 = 续费后激活	referral_qualifications 表
04-15	启航金（一次性付费，加好友退完为止）	is_paid_trial 字段
04-19	小薇健康管家上线 → 推广大使前置条件	prerequisite 配置
04-24	企业版 990 元，加 99 好友退 990 元	扩展 refund 配置
04-29	从社交与业务助手拆分为独立项目	projects/CRM/

Phase 3：从 skill 演化为独立服务（2026-04-29）

背景：数据量增长 + 多 AI 助手需要共享 CRM 数据。

架构演进轨迹：

1. Phase 1: other-contacts/{contactId}/product_relations.jsonl（纯文件）
2. Phase 2: order_db.py（SQLite，单机单文件）
3. Phase 2.5: FastAPI + uvicorn --workers 4（多进程 Web 服务）
4. Phase 2.6: systemd 守护 + Nginx 反向代理
5. Phase 2.7: JKZL 公司支付接口对接
6. Phase 2.8: v2.0 实体模型（person_id 体系）扩展

   拆分动作（2026-04-29）：

   旧: projects/社交与业务助手/instance/CRM/

   新：projects/CRM/ 包含 app.py、order_db.py、order_manager.py、jkzl_payment.py、wechat_pay.py、app_pay_endpoints.py、bus_callbacks.py、order_models.py、const.py、templates/、config/、data/

Phase 4：v2.0 实体模型升级（2026-05）

背景：跨企微应用 contactId 不统一 → 引入 person_id（unionId）

> 核心问题：同一个真人在不同企微应用下 contactId 不同（如文波在阿创 contactId=zkpw5lzq，在小薇=jvw2pdlv）。引入 person_id = hostingWorkContact.unionId 作为跨应用唯一标识。

新增表（v2.0）：persons（真人大表）、agents（AI 助手实例）、wecom_accounts（企微号）、person_wecom_relations（联系关系）。旧表改造：所有业务表增加 person_id 字段。

Phase 5：当前状态（2026-05-26）

CRM 相关能力散布范围（问题所在）：

位置	内容	状态
projects/CRM/	核心数据库 + API + 支付	□ 独立运行
skills/client-journey/	客户经营流程 + 脚本	□ 独立 skill
skills/followup-tracker/	跟进任务管理	⚠ 低活跃
projects/*/instance/skills/crm-daily-cycle/	CRM 日清日结（3 副本）	⚠ 多个副本

projects/社交与业务助手/instance/CRM/	旧 CRM 残留	□ 已废弃
projects/CRM/crm.service	systemd 指向旧路径	□ 路径不匹配
skills/smart-product-creator/	商品创建→同步 CRM	⚠ 松耦合
skills/jkzl-order-pay-AiAgentService/	下单支付 skill	⚠ 双通道

3. 当前系统架构（逆向工程）

3.1 物理部署

客户/企微 → Nginx（80/443 反向代理）→ FastAPI（4 个 uvicorn worker）→ SQLite WAL（/data/orders.db）→ JKZL 公司支付系统（RPC 调用）+ 微信支付（JSAPI 直连）

部署信息：

- 服务器：47.98.181.220（内网 172.19.35.238:8899）
- 运行时：systemd → uvicorn --workers 4
- 保活：systemd Restart=always, RestartSec=3
- 用户：admin（但 crm.service 路径指向旧目录）
- 数据库：SQLite WAL mode（单文件）

3.2 逻辑架构（4 层）

接入层：FastAPI + uvicorn (4 workers)，80+ API 端点

业务逻辑层：order_manager.py (1000+ 行)—订单管理、推荐关系、佣金结算、试用生命周期、社交关系、客户经营、JKZL 支付流程、账户管理、提现

数据访问层：order_db.py (1000+ 行)—SQLite WAL 连接管理，15+ 张表的 CRUD 操作，复杂查询（转化漏斗、统计报表）

外部集成层：jkzl_payment.py → JKZL RPC；wechat_pay.py → 微信 JSAPI；bus_callbacks.py → Bus.* 回调；tutu-cli → 企微网关

3.3 技能-项目关系图谱

CRM 能力散布在 8 个组件中，引用同一个数据库但分布在独立的 skill 和项目中，没有统一的服务契约。

- client-journey (skill) → order_db.py
- CRM 服务 (FastAPI) → order_db.py
- jkzl-order-pay-AiAgentService (skill) → order_db.py

- 3 × crm-daily-cycle（企业版/ai-trial/社交）各有自己的 SKILL.md + 脚本

4. 完整组件清单

4.1 核心代码（projects/CRM/）

文件	行数	职责	稳定性
order_db.py	~1200	15+ 表的 CRUD + 复杂查询	⚠ refund 逻辑冗余
order_manager.py	~1000	业务编排（下单/支付/佣金/结算/试用/社交）	⚠ 新旧混合
app.py	~960	FastAPI 路由 + 支付页面 + 回调处理	⚠ 页面与 API 混用
jkzl_payment.py	~400	JKZL RPC 对接（注册/下单/支付/签名）	□ 稳定
wechat_pay.py	~200	微信 JSAPI 支付	⚠ 证书文件未就绪
app_pay_endpoints.py	~150	额外支付端点	⚠ 部分重复
bus_callbacks.py	~200	健康之路订单回调处理	⚠ 格式待确认
order_models.py	~60	Pydantic 请求模型	□
const.py	~10	CRM 渠道号 9002027	□
crm-nginx.conf	~50	Nginx 反向代理配置	□ server_name 占位符
crm.service	~20	systemd 服务配置	□ 路径指向旧目录
templates/	—	落地页/支付页	⚠ 硬编码
config/order-config.json	—	产品+支付+合规配置	□ 灵活

4.2 关联 Skill

Skill	关联方式	说明

client-journey	脚本调用 CRM API	客户经营流程定义，9 阶段 + 标签体系
followup-tracker	独立文件系统	跟进任务管理，低活跃
crm-daily-cycle ×3	脚本直接查 DB	日清日结集成，3 个副本不一致
jkzl-order-pay-AiAgentService	独立下单流程	下单入口，与 CRM 下单流程并行
smart-product-creator	--sync-crm 参数	商品创建时同步 CRM 配置
user-onboard	调用 /api/social/add	添加好友触发社交关系 + 退款
agent-marketplace	引用商品配置	营销场景关联 CRM 商品数据

4.3 数据库表（18 张）

#	表名	用途	生成方式
1	orders	订单记录	下单创建，支付更新
2	referrals	推荐关系（两级）	部署时自动记录
3	commissions	佣金记录	支付成功后生成
4	payment_logs	支付回调日志	每次回调追加
5	user_accounts	用户账户（余额）	首次支付创建
6	account_transactions	账户流水（append-only）	每笔变动追加
7	deployments	部署记录	部署时创建
8	trial_lifecycle	试用生命周期	试用开始创建
9	social_connections	社交关系	添加好友创建
10	client_journey	客户关系经营	AI 助手推进更新
11	client_product_journey	客户产品经营	推送产品更新
12	referral_qualifications	推广资格	续费后激活
13	persons	真人大表（v2.0 新增）	注册时创建

14	agents	AI 助手实例	部署时注册
15	wecom_accounts	企微号	绑定企微注册
16	person_wecom_relations	人-企微关系（v2.0 新增）	绑定创建
17	referral_relations	通用推荐关系	v2.late 新增
18+	referral_qualification_history	资格变更历史	每次变更追加

5. 数据模型全景图

5.1 核心业务实体关系（v2.0）

person (unionId) → person_wecom_relations → wecom_account (botSenderId) → contact_id (external_userid)

核心业务表通过 contact_id 关联：

- user_accounts → account_transactions
- orders → referrals → commissions → referral_qualifications
- trial_lifecycle
- social_connections
- client_journey → client_product_journey

5.2 核心业务流

注册 → 试用 → 加好友（每个好友退 ¥10，仅付费试用）→ 付费（激活推广资格+佣金）

5.3 资金流

用户 → ¥599 保证金 → user_accounts.balance（¥0 起始）→ 加好友 → friend_refund → balance += ¥10 → 满 60 好友 → balance ≈ ¥600 → 33 天等待期 → withdrawable = true → 微信原路退回 ¥599 → 续费 → 激活推广资格 → 可推广获佣金

6. API 端点清单

全部 40+ API 端点（按功能分组）：

6.1 系统

端点	方法	说明
/health	GET	健康检查

6.2 产品

端点	方法	说明
/api/products	GET	产品列表
/api/jkzl/goods	GET	JKZL 商品列表

6.3 订单

端点	方法	说明
/api/order/{order_id}	GET	订单详情
/api/order/create	POST	创建订单
/api/order/create-with-jkzl	POST	下单+JKZL支付

6.4 部署

端点	方法	说明
/api/deployment/check	GET	检查部署
/api/deployment/info	GET	部署信息
/api/deployment/list	GET	部署列表

/api/deployment/create	POST	创建部署
/api/deployment/status	POST	更新状态
/api/deployment/register	POST	注册推广关系

6.5 试用

端点	方法	说明
/api/trial/status	GET	试用状态
/api/trial/convert	POST	试用转正
/api/trial/manage	POST	试用管理

6.6 客户/推荐

端点	方法	说明
/api/customer/profile	GET	客户全景档案
/api/referrer/stats	GET	推荐人统计
/api/referrer/downline	GET	下游客户

6.7 社交关系

端点	方法	说明
/api/social/direct	GET	直接好友
/api/social/indirect	GET	间接好友
/api/social/stats	GET	社交统计
/api/social/add	POST	加好友
/api/social/remove	POST	删好友
/api/social/friend-refund	GET	好友退款状态

6.8 转化统计

端点	方法	说明
/api/conversion/stats	GET	转化率
/api/conversion/curve	GET	转化曲线
/api/conversion/funnel	GET	转化漏斗

6.9 账户

端点	方法	说明
/api/account/{contact_id}	GET	账户信息
/api/account/{contact_id}/transactions	GET	流水
/api/account/refund	POST	好友退款
/api/account/withdrawable	GET	可提现金额
/api/account/withdraw	POST	提现

6.10 支付

端点	方法	说明
/pay	GET	支付页
/pay/success	GET	支付成功页
/api/pay/jsapi	POST	JSAPI 下单
/api/pay/jsapi/mock	POST	Mock 支付
/api/pay/wechat/oauth	GET	微信 OAuth
/api/jkzl/payment-callback	POST	JKZL 回调
/api/jkzl/poll-payment-status	POST	轮询支付状态
/api/jkzl/check-expired	POST	检查超时订单

6.11 客户经营

端点	方法	说明
/api/client-journey/update	POST	更新客户阶段
/api/client-journey	GET	查询客户阶段
/api/client-product-journey/update	POST	更新产品阶段
/api/client-product-journey	GET	查询产品阶段
/api/client-push-history	GET	推送历史
/api/clients/list	GET	客户清单

6.12 Bus.* 回调

端点	方法	说明
/callback/bus/orderCompletePay	POST	支付完成
/callback/bus/cancelBus	POST	取消订单
/callback/bus/undoBus	POST	撤销订单
/callback/bus/orderoverTime	POST	订单超时
/callback/bus/getGoodsPrice	POST	查询价格

7. 业务规则全集

7.1 定价规则

产品	月付	保证金	退款上限	直接佣金	间接佣金
AI助手（个人版）	¥599/月	¥599	加满60人退¥600	¥120/月	¥60/月
AI助手企业版	¥990/月	¥990	加满99人退¥990	¥200/月	¥100/月
小薇健康管家	¥99/月	¥99	—	¥20(20%)/月	¥10(10%)/月
小薇健康管家（年付）	¥990/年	¥99	—	¥200(25%)/年	¥100(10%)/年
个人启航票	¥599 一次性	—	—	—	—
企业启航票	¥990 一次性	—	—	—	—

7.2 推广佣金规则

1. 两级分销：直接 = level 1（20-25%），间接 = level 2（10%）
2. 自动记录：部署时自动记录推荐关系（不需要邀请码）
3. 佣金状态：pending → paid（结算日批处理）
4. 结算日：每月 5 号（bonusPaymentDay=5）
5. 推广资格：续费后激活，不续费自动暂停
6. 前置条件：小薇推广需要先有AI助手推广资格

7.3 试用保证金规则

1. 保证金 ¥599 → 公司账户（不在用户余额）
2. 加好友退费：每加 1 个好友退 ¥10（仅付费试用）

3. 退款上限：个人版 60 人退 ¥600，企业版 99 人退 ¥990
4. 提现等待期：付款时间 + 33 天
5. 启航金模式：一次性付费，加好友退完为止

7.4 社交关系规则

1. 好友关系：owner → friend（单向，但实际双向触达）
2. 间接好友：好友的好友（按 owner 的 social_connections 递归查询 depth=2）
3. 退款触发：每次 add_social_connection 自动触发 friend_refund 逻辑
4. 接触上限：个人版最多退 60 个好友，企业版最多退 99 个

7.5 客户经营阶段

关系阶段（client_journey）：stranger → icebreak → exploring → trusting → first_deal → nurturing → bonded → dormant → lost

产品阶段（client_product_journey）：stranger → introduced → exploring → trial_active → converted → rejected

8. 已知问题与不稳定点

8.1 架构层面（重点关注）

> 最严重的两个根因：SQLite 单文件写锁（4 个 worker 抢 1 个 DB 文件）+ 单点故障（单服务器无容灾）

#	问题	严重度	影响
A1	SQLite 单文件并发瓶颈	□ 严重	高并发下单时可能堵塞/写冲突
A2	单点故障（SPOF）	□ 严重	服务器宕机则全部 CRM 不可用
A3	无数据库备份策略	□ 中	数据丢失无法恢复
A4	无连接池管理	□ 中	连接数随并发线性增长
A5	Nginx 配置未上线	□ 中	HTTPS 未配置
A6	systemd 配置路径错误	□ 中	重启可能失败

8.2 代码质量层面

#	问题	严重度	说明
B1	person_id 解析冗余	□ 中	_resolve_person_id() 和 _resolve_to_person_id() 功能相同
B2	退款配置逻辑复杂	□ 中	_get_refund_config() 多个 fallback 路径容易出错
B3	JKZL 回调格式未确认	□ 中	大量等陈悦确认的字段映射

B4	新旧调用方式共存	□ 中	同时支持旧函数签名和新 Pydantic model
B5	页面与 API 混用	□ 中	app.py 同时包含 API 路由和 HTML 渲染
B6	crm-daily-cycle 三副本不一致	□ 中	三个副本存在细微差异

8.3 稳定性隐患（CRM 不太稳定的直接根因）

#	隐患	场景	实际影响
C1	SQLite 写锁	同时下单+回调+cron检查到期	事务冲突导致 HTTP 500
C2	内存增长	长时间运行后连接未释放	内存泄漏
C3	JKZL RPC 超时	tutu-cli 调用 30 秒超时	请求排队积压
C4	时钟不同步	33 天提现依赖系统时间	提现判断错误
C5	幂等性不足	JKZL 回调重试可能重复处理	订单状态异常
C6	事务边界不清晰	social+refund 在同一事务中	部分失败不一致

8.4 运维层面

- D1：无日志归档 — /var/log/ 无 CRM 独立日志
- D2：无监控告警 — systemd 重启不通知任何人
- D3：无性能指标 — 无法知道 QPS/响应时间/p99
- D4：无数据库维护 — VACUUM/一致性检查从未执行
- D5：到期提醒无确认 — trial_expired 通知发送状态未验证

9. v3.0 高可用架构方案

9.1 总体架构建议

> 利用公司高可用基础设施，实现负载均衡→无状态应用层→数据层→观测层的完整架构

负载均衡层：Nginx / ALB → 多副本 CRM 服务

无状态应用层：CRM API Service（containerized, 多副本）

- 订单服务 (Order Service)
- 账户服务 (Account Service)
- 社交关系服务 (Social Service)
- 客户经营服务 (Client Journey Service)
- 支付网关 (Payment Gateway)

数据层：关系型 DB（主从/集群）+ KV 缓存（Redis）+ 消息队列（MQ）

观测层：日志(ELK) + 指标(Prometheus) + 告警

9.2 关键架构决策

决策项	v2.0（当前）	v3.0（建议）	理由
数据库	SQLite WAL	待评估（公司DB）	SQLite有写锁，不支持HA
服务形态	单体 FastAPI	微服务/模块化	解耦后可独立扩缩容
部署	systemd + uvicorn	容器化（Docker）	多副本部署，滚动更新
缓存	无	Redis（可选）	高频查询可缓存
队列	无	MQ（可选）	支付回调等异步事件

监控	无	Prometheus+Grafana	提前发现性能问题
日志	journald	ELK/Loki	日志查询和告警
API文档	无	OpenAPI/Swagger	自动生成文档

9.3 模块拆分建议

将当前单体 app.py + order_manager.py + order_db.py 拆分为 5 个模块：

- crm-api-gateway/ — ★ 统一入口
- order/ — 订单 → 支付 → 佣金
- account/ — 账户 → 流水 → 提现
- social/ — 社交关系 → 好友退款
- journey/ — 客户经营 → 阶段管理
- payment/ — 支付网关（微信/JKZL）

9.4 数据库方案建议

关键指标参考：orders 表 ~200 条（月增长 ~100），account_transactions ~500 条（月增长 ~300），social_connections ~200 条（月增长 ~100）

> 当前数据量不大，可以先确保 HA 而非先迁移数据库。优先解决：单点故障 + 并发写锁 + 备份。估算未来 12 个月数据量（约 3000 条/表），仍可用轻量方案。

9.5 公司高可用能力对接

公司层能力	对接方式	优先级
负载均衡	通过公司 LB 入口转发现有多副本	P0
数据库	评估公司 DB 集群或 RMS	P0
监控告警	接入公司 Prometheus	P1
容器平台	评估公司 K8s 或 ECS	P1

消息队列	评估公司 MQ	P2
CI/CD	接入公司部署流水线	P2

10. 迁移路线图

Phase 1：止血（1-2 天）

> 解决 CRM 不太稳定的根因 — 不需要架构改造，纯运维和代码修复

#	任务	说明	负责人
1	SQLite WAL 参数调优	PRAGMA busy_timeout, mmap_size 等	阿创
2	数据库自动备份	cron 定时备份 orders.db 到 OSS	阿创
3	crm.service 路径修复	指向 projects/CRM/ 而非旧目录	阿创
4	Nginx 配置上线	配置真实域名 + HTTPS	阿创/运维
5	JKZL 回调字段确认	找陈悦确认回调 payload 格式	马坚/阿创
6	修复 person_id 解析冗余	合并两个函数	阿创
7	三副本 crm-daily-cycle 统一	指定一个标准版本	阿创

Phase 2：高可用架构迁移（1-2 周）

#	任务	说明	负责人
1	CRM 服务容器化	Dockerfile + docker-compose	马坚/阿创
2	对接公司 HA 基础设施	负载均衡 + 多副本部署	马坚

3	数据库方案确认	迁移到公司DB或继续SQLite+副本	马坚/老张
4	服务拆分为 5 模块	按 9.3 拆分	阿创
5	OpenAPI 文档自动生成	FastAPI 自带 swagger	阿创
6	渠道号 9002027 多环境	dev/test/prod 环境隔离	马坚

Phase 3：观测与运维（1 周）

#	任务	说明	负责人
1	接入公司 Prometheus	请求量/错误率/响应时间	马坚
2	关键告警配置	服务宕机/支付失败/超时订单堆积	阿创
3	日志结构化管理	统一 JSON 日志格式	阿创
4	数据库定期维护	VACUUM + 数据一致性检查	阿创
5	幂等性改造	JKZL/微信回调接口等幂	阿创
6	事务边界优化	关键路径的事务粒度	阿创

Phase 4：能力扩展（2-4 周）

#	任务	说明	负责人
1	客户端 SDK	Python SDK 封装 CRM API	阿创
2	统一 payment skill	合并 jkzl-order-pay + CRM 支付	阿创
3	推广数据分析面板	转化漏斗/佣金排行榜等	阿创
4	定时结算自动化	每月 5 号自动执行结算	阿创
5	多商品自动计费	按产品差异化计费逻辑	阿创
6	数据迁移方案	从 SQLite 迁移到目标 DB	马坚/阿创

附录 A：争议/待决决策

#	待决策事项	讨论背景	建议
D1	数据库选型	SQLite够用但有写锁，公司DB强但引入运维成本	如果HA架构提供DB服务则用公司DB，否则先优化SQLite
D2	微服务 vs 模块化单体	当前数据量不大，微服务增加复杂度	先模块化单体，按9.3拆包不拆进程
D3	容器 vs 直接部署	容器化对于HA滚动更新更方便	建议容器化，对接公司HA基建
D4	client-journey skill是否并入CRM	当前是独立skill调用CRM API	建议保留skill但统一调用SDK
D5	JKZL支付是保留还是统一到CRM	两条支付路径并存	建议CRM + JKZL作为payment provider之一

附录 B：关键文件定位（供开发参考）

文件	路径
主应用	projects/CRM/app.py
数据库层	projects/CRM/order_db.py
业务逻辑层	projects/CRM/order_manager.py
JKZL 支付	projects/CRM/jkzl_payment.py
微信支付	projects/CRM/wechat_pay.py
支付端点	projects/CRM/app_pay_endpoints.py
Bus 回调	projects/CRM/bus_callbacks.py
数据模型	projects/CRM/order_models.py
常量	projects/CRM/const.py
Nginx 配置	projects/CRM/crm-nginx.conf
systemd 服务	projects/CRM/crm.service
产品配置	projects/CRM/config/order-config.json
客户经营 skill	skills/client-journey/SKILL.md
跟进 skill	skills/followup-tracker/SKILL.md
日清日结副本1	projects/AI助手企业版/instance/skills/crm-daily-cycle/
日清日结副本2	projects/ai-trial/instance/skills/crm-daily-cycle/
日清日结副本3	projects/社交与业务助手/instance/skills/crm-daily-cycle/

本 PRD 基于老张与阿创 2026-04-05 至 2026-05-26 期间的全部沟通、代码实现记录逆向梳理。

版本：v3.0-draft | 生成：2026-05-26 | 用途：下一版本重构参考