Saving

「Saving」

Trying to do better

PROFILE / 简介

拥有 **8 年后端开发经验**，熟悉 **Java**、**Golang**、**Kotlin**，擅长 **高性能**、**高可用系统** 的设计与实现。具备良好的 **AI 协同开发能力**，熟悉 **Vibe Coding** 与 **OpenSpec** 规范，熟练使用 **Codex**、**Cursor**、**Claude Code** 等工具，并掌握多种工程提效方法，能够在 **需求分析**、**代码开发**、**重构优化**、**测试调试** 等环节实现高效协同，持续提升研发效率与代码质量。热爱运动，注重自我精进，追求专业能力与作品质量的持续进化。

PERSONAL DETAILS / 个人信息

- 姓名: 吴煌全「Saving」 / 男 / 1997
- 专业: 软件技术
- 毕业院校: 广州城建职业学院
- 期望职位: 高级后端开发 / **TeamLeader**

CONTACT / 联系方式

- Phone: 18316260997
- Wechat: sav7ng
- Email: savingrun@gmail.com
- Github: https://github.com/sav7ng
- WechatBlog: **爱敲代码的猫**

SKILLS / 技能

```
Opensepc Cursor Codex ClaudeCode

Java Kotlin Golang SpringFramework Maven/Gradle

Mybatis Kratos Qmgo MySQL Redis

MongoDB ClickHouse Metabase APISIX RocketMQ

Pulsar Seata XxlJob Sentinel Kafka
```

```
10
11  RPC Docker CI/CD
```

WORK EXPERIENCE / 工作经历

广州速启科技有限责任公司「高级后端开发工程师/2024年07月-至今」

CloudPhone OpenClaw Plugin ▪ 云手机 AI 自动化 ▪ Agent 工具能力开放

- OpenClaw 插件架构

> TypeScript 5.4+ / Node.js (ES2020) / OpenClaw Plugin API / MCP (Model Context Protocol) / Fetch API / AbortController / Crypto (MD5) / GitHub Actions CI/CD / npm Registry

- 项目概况

> CloudPhone 是基于 OpenClaw 插件体系开发的云手机 AI Agent 能力开放插件，为 AI Agent 提供云手机设备管理与 UI 自动化操控能力。项目核心是将云手机 OpenAPI 封装为 14 个标准化的 `cloudphone_*` Agent 工具，涵盖用户信息获取、设备列表与详情查询、电源控制、ADB 连接、坐标点击/长按/滑动、文本输入与清除、系统按键、截图/UI树快照、图片渲染等完整能力。同时内置 `basic-skill` Skill 教学模块，引导 Agent 按照"观察 → 行动 → 验证 → 再观察"的短闭环稳定执行多步 UI 自动化任务。插件通过 npm 包 `@whateverai/cloudphone` 发布，支持 OpenClaw Gateway 一键安装与配置，Agent 即装即用。

- 担任职责

> - Agent 工具体系设计 / 负责设计并实现 14 个 `cloudphone_*` 工具定义，包括工具名、描述、JSON Schema 参数校验与执行函数，遵循 MCP 内容项规范（text / image），统一 Agent 工具调用与响应格式
> - OpenAPI 请求抽象层 / 负责封装统一的 `apiRequest` 请求函数，内置超时控制（AbortController）、鉴权头注入（Authorization ApiKey）、URL 拼接与归一化、业务状态码判定（success / code 多格式兼容）、错误信息提取等能力，屏蔽底层 HTTP 细节
> - 设备管理能力 / 负责实现用户信息获取、设备列表分页与筛选、设备详情、电源控制（开机/关机/重启）、ADB/SSH 连接信息获取等 5 个设备管理工具，支撑 Agent 完成设备选择与状态确认

- UI 交互操控能力 / 负责实现坐标点击、长按、滑动、文本输入、文本清除、系统按键（BACK/HOME/ENTER/RECENT/POWER）等 6 个 UI 交互工具，使 Agent 具备完整的云手机屏幕操控能力
- 状态观测与截图渲染 / 负责实现截图/UI树快照获取、页面等待条件（element_appear/element_disappear/page_stable）、截图 URL 渲染为可展示图片等 3 个状态观测工具，其中图片渲染采用远程拉取 → MD5 哈希去重 → 本地缓存 → MCP image 内容项 + MEDIA 兼容回退的完整链路
- 内置 Skill 设计 / 负责设计并编写 `basic-skill` Skill 教学文档，定义标准工作流（select_device → confirm_online → observe → act → verify → observe_again）、恢复策略、能力边界声明与常见问题排查指南，提升 Agent 在多步 UI 自动化场景下的执行稳定性
- 插件注册与生命周期 / 负责实现插件入口（register 函数），完成运行时配置解析、工具批量注册、执行结果规范化、异常兜底处理与结构化日志输出，确保插件在 OpenClaw Gateway 中稳定加载运行
- CI/CD 与发布 / 负责基于 GitHub Actions 实现自动化构建与 npm 发布流水线，main 分支 push 触发构建，`v*` 标签触发公开发布到 npm Registry

- TECHNICAL POINTS

- 开发过程中结合 Cursor AI 编程助手辅助提效，在工具定义模板生成、请求抽象层搭建、Skill 文档编写及 CI/CD 配置等环节借助 AI 加速迭代，有效提升编码效率与交付节奏
- 基于 OpenClaw Plugin API 实现 14 个标准化 Agent 工具，严格遵循 `name / description / parameters(JSON Schema) / execute` 四元组定义，工具返回值统一采用 MCP 内容项规范（text / image），Agent 即装即用
- 封装统一 apiRequest 抽象层，内置 AbortController 超时中断、Authorization 鉴权头注入、多格式业务状态码兼容判定与结构化错误提取，所有工具共享同一请求链路
- 截图渲染工具实现远程图片拉取 → MD5 哈希去重 → 本地缓存 → MIME 双重推断（Content-Type 优先 + URL 扩展名兜底）→ MCP image + MEDIA 兼容回退的完整图片处理链路
- 设计 basic-skill 内置 Skill，定义"观察 → 行动 → 验证 → 再观察"标准工作流与分层恢复策略（BACK → HOME → 截图 → 重启），在不增加新工具的前提下提升 Agent 多步 UI 自动化执行稳定性
- 基于 GitHub Actions 实现两阶段 CI/CD（build → publish），`v*` 标签推送自动构建并发布到公共 npm Registry，实现版本化自动发布

- 个人收获

> 通过该项目，我深入理解了 AI Agent 工具化开放的工程实践，从"如何将后端 API 能力标准化封装为 Agent 可调用工具"到"如何通过 Skill 引导 Agent 稳定执行多步自动化任务"，掌握了 MCP 协议内容项规范、工具参数 JSON Schema 设计、以及 Agent 行为引导（Skill）与底层能力（Plugin）的职责分离思想。同时通过实际开发 OpenClaw Plugin 并完成 npm 公开发布，熟练掌握了 OpenClaw 插件体系的完整开发流程，包括插件注册生命周期、配置 Schema 声明、Skill 编写规范与 Gateway 集成调试，具备了独立开发与发布 OpenClaw 插件的能力。项目过程中也提升了在 TypeScript 工程化、API 请求抽象、图片处理链路设计、CI/CD 自动化发布以及 AI Agent 生态能力建设等方面的实战经验。

CPC 云手机 AI Agent 编排平台 · 云手机自动化 · OpenAPI / MCP 能力开放

- **分布式微服务架构**

> **Java 17 / Spring Boot 3 / Spring Cloud / Spring AI / OpenAI / MCP / OpenFeign / Nacos / Redis / MySQL / JPA / MyBatis-Plus / Kafka / XXL-JOB / WebFlux / SSE / Spring Security / Swagger / Prometheus**

- **项目概况**

> CPC 云手机 AI 编排服务是微服务体系中的独立 AI 能力模块，主要面向云手机自动化与智能任务执行场景。项目核心是将自然语言指令解析为结构化任务，并根据任务类型自动路由到 Agent 执行器或脚本执行服务，完成任务创建、状态流转、执行派发、结果回调、会话沉淀的完整闭环。同时对外提供 HTTP OpenAPI 与 MCP Tool 调用能力，支撑设备管理、设备控制、AI 任务执行与结果获取等业务场景。本人主要参与该项目核心链路的设计与开发，推动 AI 能力与既有微服务、设备能力、异步调度体系的深度融合，提升平台在智能自动化场景下的可复用性与业务落地能力。

- **担任职责**

> - **AI任务解析中心 /** 负责基于 **Spring AI + OpenAI** 实现自然语言任务结构化解析，将用户输入转为任务类型、执行时间、策略、目标应用、动作内容等字段，并结合租户应用库动态注入 Prompt 上下文，提升 AI 指令识别的业务准确性

- **Agent/Script任务编排 /** 负责设计统一任务调度链路，完成任务落库、状态流转、立即执行、定时执行、策略分发等能力，支持根据任务策略自动派发至外部 Agent 执行器或脚本服务
- **云手机开放接口 /** 负责设计并开发云手机设备管理接口，包括设备列表、设备详情、实时状态、电源控制、应用启动、ADB 连接信息获取等能力，支撑外部平台和内部系统统一接入设备控制能力
- **MCP能力开放 /** 负责将 AI 任务能力封装为 MCP Tool，对外以 JSON-RPC 风格暴露任务创建、结果获取等接口，支持智能体工具调用与平台化接入
- **回调闭环与会话沉淀 /** 负责处理 Agent 执行回调与脚本任务回调，更新任务状态、转发业务通知，并将执行结果同步写入会话历史，支持多轮上下文追踪与结果复盘
- **定时任务调度 /** 负责基于 **XXL-JOB** 实现到期任务扫描与自动派发，补齐 AI 能力在延迟执行、预约执行等业务场景下的落地能力
- **租户应用脚本库 /** 负责租户级应用信息管理与快捷命令能力建设，支持通过应用配置快速创建并下发自动化任务，增强 AI 与具体业务应用之间的绑定关系
- **OpenAPI鉴权治理 /** 负责开放接口鉴权方案设计与开发，基于 Redis Token 校验与线程上下文用户态传递实现设备类接口安全访问控制

- **TECHNICAL POINTS**

- 基于 **Spring AI + BeanOutputConverter** 将大模型输出直接转换为结构化任务对象，降低自然语言解析对大量手写规则的依赖，提升 AI 指令到业务任务的落地效率
- 将租户应用库信息动态注入 Prompt 上下文，使模型能够结合实际业务应用完成任务识别与 `agent/script` 策略路由，而不是停留在通用问答层面
- 设计统一异步任务编排链路，打通“任务创建 -> 状态持久化 -> 执行派发 -> 回调处理 -> 通知转发 -> 会话沉淀”完整闭环，增强任务场景下的可观测性与可追踪性
- 基于 **MCP Server + HTTP OpenAPI** 双协议暴露能力，使服务既可被传统业务系统调用，也能被支持 Tool Calling 的智能体直接接入，提升平台通用性与扩展性
- 利用 **Redis Cache + MySQL/JPA** 实现会话与消息持久化机制，对用户消息、回调结果、流式脚本生成结果进行统一沉淀，支撑多轮上下文连续追踪
- 基于 **XXL-JOB** 实现定时任务扫描与派发机制，使 AI 指令具备预约执行、延迟执行和自动触发能力
- 结合 **Kafka / OTA 指令链路 / OpenFeign** 与云手机能力集成，实现设备指令发送、执行协同与外部服务联动，推动 AI 自动化能力真正落到设备执行层

> - 通过 OpenAPI 鉴权拦截器结合 Redis 用户态数据完成统一鉴权，避免开放设备接口被未授权访问，提升接口安全性与可控性

- **个人收获**

> 通过该项目，我对 AI 大模型在企业级业务系统中的工程化落地有了更深入的理解，不再局限于“调用模型并返回结果”，而是进一步掌握了从 Prompt 设计、结构化任务解析、异步任务编排、执行回调闭环到能力平台化开放的完整实现思路。项目过程中，我提升了在 AI 工程化、微服务协同、云手机自动化、多租户业务抽象、接口安全治理以及上下文会话设计等方面的实战能力，也让我对“如何将大模型能力真正转化为稳定、可执行、可追踪的业务能力”有了更成熟的认识。

CPC 3.0 多租户云机管理平台 / 云手机 SaaS 平台

- **分布式微服务架构**

> **Spring Boot 2.6 / Spring Cloud 2021 / Java 17 / Nacos / OpenFeign / MyBatis-Plus / MySQL / Redis / Redisson / MongoDB / ShardingSphere / Kafka / XXL-Job / Prometheus / Netty / MQTT / WebSocket / Dubbo(部分模块) / Swagger**

- **项目概况**

> CPC 3.0 是一套围绕云机/云手机资源管理、租户隔离、设备路由、用户接入、订单计费和 SaaS 配置构建的多模块后端平台。整体仓库由 `lib` 公共能力层、`worker` 业务微服务层、`master` 平台管理入口、`mopenapi` 多开放接口层以及 `gateway` 工具服务组成，覆盖管理后台、客户端 API、开放平台、worker 集群协同、设备接入、异步任务、监控统计和第三方云厂商对接等场景。平台核心业务围绕租户中心、用户中心、订单与计费、设备与云机中心、网关与路由、SaaS 定制配置、开放平台能力、IoT/Netty 通信链路、消息与短信、监控指标与任务调度展开，具备较强的多模块协同和分布式架构特征。

- **担任职责**

> - 平台架构治理 / 参与 CPC 3.0 多模块微服务平台的架构拆分与服务边界设计，围绕 `lib` 公共能力、`worker` 业务中心、`portalapi` 平台入口、`mopenapi` 开放能力和工具服务形成统一的分层协作模式，沉淀可复用的通用组件和服务模板

- 租户与用户中心 **/** 负责或参与多租户能力建设，围绕租户配置、用户信息、登录记录、验证码、用户反馈、租户与设备绑定等能力进行设计与实现，支撑 SaaS 场景下多租户隔离和业务配置扩展
- 订单与计费链路 **/** 参与订单、商品、支付回调、套餐和计费相关服务建设，打通订单中心、租户系统和上层业务模块之间的协同链路，支撑平台资源订购、续费、套餐配置和业务计费场景
- 设备与云机中心 **/** 负责或参与云机设备、资源池、共享存储、应用上下架、批量换机、重置、Root、截图、文件上传、设备定位修改等能力建设，覆盖云机生命周期管理和批量运维操作场景
- 网关与路由体系 **/** 参与 `cpc-center-gateway`、`mopenapi` 和 worker 路由关系的设计与实现，围绕租户到 worker、设备到 worker 的域名路由、请求转发、接口签名和跨 worker 数据同步能力建设开放平台调用链路
- 开放平台与客户端接口 **/** 参与 `clientapi`、`boxapi`、`backendapi`、`openapi`、`mopenapi` 等 API 服务建设，对外提供客户端、盒子端、平台端和开放接口能力，支撑多端接入与服务复用
- IoT 与设备通信 **/** 参与 `kite-netty`、`kite-proxy`、`kite-manager`、`kite-monitor`、`iotapi` 等服务协同建设，支撑设备接入、代理转发、集群管理、MQTT/WebSocket/Netty 通信、运行监控和设备状态采集等场景
- 任务调度与异步链路 **/** 参与 Kafka 消息消费、定时任务、批处理任务和跨模块异步通知能力建设，将资源分配、应用上下架、设备回收、消息通知、数据同步等流程解耦为可追踪、可重试的任务链路
- 第三方生态对接 **/** 参与 Huawei、Tencent、Volcengine、Baidu、Ali、Firebase、Migu、Line、蜂助手(FHelper) 等第三方云厂商或生态能力的接入，支撑多云资源管理、设备同步、账号对接、回调通知和扩展能力集成
- 监控与数据能力 **/** 参与 Prometheus 指标采集、运行状态统计、资源池统计、云机服务状态记录、埋点日志和业务数据接口建设，支撑平台可观测性与数据分析场景

- **TECHNICAL POINTS**

- 基于 **Spring Boot 2.6 + Spring Cloud 2021 + Nacos + OpenFeign** 构建多模块微服务体系，将管理后台、客户端 API、开放平台、任务中心、网关中心、设备中心和数据中心按职责拆分，降低单体耦合度，提升跨团队协作和独立迭代能力
- 在仓库层面形成统一的 **api / biz / dao / service** 分层规范，并通过 `lib/cpc-core` 中的 `starter` 模块沉淀 Redis、MyBatis、Kafka、MongoDB、Prometheus、XXL-Job、ShardingSphere、Feign、锁、认证等公共能力，提升重复业务场景的复用效率

> - 利用 **Redis + Redisson** 实现分布式锁、状态控制和并发保护，在设备分配、换机、共享、回调、回收、备份恢复等涉及同一用户设备或资源池的流程中减少并发覆盖和脏写风险
> - 借助 **mopenapi 路由表 + worker 域名映射 + 跨 worker 同步接口** 构建设备与租户的分布式路由能力，使开放平台请求能够根据租户、设备、区域和 worker 关系进行动态转发，支撑多 worker 集群下的业务调度与扩展
> - 使用 **Kafka + XXL-Job** 处理设备分配、回收、应用上下架、日志采集、统计任务、通知任务等异步流程，通过任务化和消息化方式降低核心链路阻塞，提升复杂业务流程的可拆解性与可恢复性
> - 结合 **MongoDB + ShardingSphere + MyBatis-Plus** 处理结构化与非结构化数据场景，在租户配置、日志记录、统计数据、设备关系、业务主表等不同类型数据间实现较灵活的数据组织方式
> - 在设备侧通过 **Netty / MQTT / WebSocket** 支撑 IoT 与云机通信链路，配合 `kite-*` 系列服务完成设备接入、代理转发、集群管理、状态采集、指令调度和监控统计，形成从设备接入到服务治理的一体化能力
> - 通过 **Prometheus 指标监控 + 业务统计 REST 接口** 建立平台可观测性能力，对设备数、服务状态、资源池、采集日志、云机流媒体状态等指标进行采集和暴露，支持后续监控告警与分析场景
> - 基于 **开放平台 + 多云厂商适配** 形成对 Huawei、Tencent、Volcengine、Baidu、Ali、Firebase、Migu、Line、FHelper 等生态能力的扩展接口，提升平台对不同设备供应商、短信/消息、云资源和外部系统的兼容性

- **个人收获**

> 通过参与 CPC 3.0 这类多租户云机平台的建设，进一步加深了我对微服务架构拆分、公共能力沉淀、跨模块协同和复杂业务边界划分的理解；在租户、设备、路由、开放平台、任务调度和监控链路等场景中，提升了我对分布式系统设计、并发控制、异步任务编排、跨 worker 数据同步和平台型后端建设的整体把控能力。同时也强化了我在多云厂商接入、IoT/Netty 通信链路、SaaS 配置扩展、接口复用和可观测性建设方面的实践经验，使我在面对复杂平台型项目时，能够更系统地从架构、业务、稳定性和可扩展性角度进行思考与落地。

广州大事件科技网络有限公司「后端开发组长/2020年11月-2024年07月」

贪吃商城小程序 ▪ 移动端 ▪ 商家端

- **分布式微服务架构**

> **SpringCloud / MybatisPlus / MySql(polarDB) / Redis / MongoDB / ClickHouse / ES / RocketMQ / Pulsar / Sentry / Seata / XxlJob / Sentinel / OpenFeign / APISIX / Swagger / Kubernetes / 分布式发号器 / Jenkins**

- **项目概况**

> 贪吃商城 boomsj.com 是自研从0到1，现注册会员400W，月流水平均2.5KW的本地生活服务电商平台。依托微信十亿生态，及大事件流量优势，以技术为驱动力，帮助本地商家精准匹配消费群体，降低营销门槛，为本地商户打造一站式推广服务。从0到1的参与到项目开发中，在其中负责系统架构的讨论以及担任相关模块设计开发，并且参与需求评审参与需求讨论原型设计是否合理，并且在其中独立对发号器进行调研测试开发设计，负责多渠道数据的用户行为数据埋点与数据统计系统设计开发。协助组员工作分配，以及完成需求开发方案。

- **担任职责**

> - 用户中心 **/** 负责用户接入兼容多端(网页/微信/抖音/支付宝)中统一登录注册的SAAS用户层功能，和用户服务中心的设计与开发，确保高流量场景下用户的正常登录注册;用户等级功能模块开发，用户任务系统，行为事件系统
> - 消息中心 **/** 个人负责担任设计，开发主导角色，进行融合统一第三方消息发送渠道(短信/APP极光推送/站内信/微信公众号模板消息/微信小程序订阅消息/支付宝小程序推送)，设计其中管理关键字与事件中心进行配合进行管理，解决商场架构中消息多渠道统一发送的难点，该服务在 架构中承担着小程序/APP多端的消息推送，使商场的转化率提高，更好触达以激活用户
> - APISIX网关 **/** 负责网关是后端服务的统一入口，所有的客户端都通过统一的网关接入微服务，网关通过负载均衡将客户端请求转发到具体的后端服务，在网关层处理所有的非业务功能;实现Route(路由)，Token身份认证，用户鉴权，较验租户ID，全局异常，限流处理功能
> - OpenAPI网关 **/** 负责开发第三方的关鉴权，参数校验，API请求次数，黑白名单功能
> - 微信中心 **/** 负责微信服务设计开发重构;开发微信开放平台的第三方平台，实现微信多媒体平台(公众号)授权管理，监听授权公众号事件如扫码，自动回复客服消息实现公众号引流功能，并且对起做缓存性能优化;开发对接微信生成小程序UrlLink功能，URL跳转打开小程序场景;公众号带参二维码生成跳转功能添加转发长内容转换，解决回调信息长度超出限制问题;订阅消息发送功能
> - 渠道码数据统计中心 **/** 负责交易分析图表支付总金额，支付人数，单均支付金额和人均消费次数标识指数开发;负责开发渠道码功能，并且把推广码兼容升级到渠道

码相同功能，实现渠道ID绑定小程序的用户行为记录链路功能;负责开发使用ClickHouse实现渠道码查看埋点数据分析相关功能，可以详细看到该渠道码带来的访问人数，访问次数，客单价，下单人数，下单订单数，下单件数，下单金额，支付人数，支付订单数，支付件数，支付金额，新增用户数

- 第三方服务(腾讯广告附近推/哔啦啦) **/** 负责开发对接微信第三方附近推朋友圈广告功能，广告投放流程开发，以及浏览/购券/核销的用户行为数据回传流程开发，集成渠道码特性，实现观察链路;对接哔啦啦第三方，实现与其门店管理和扫码核销功能
- 商品订单模块 **/** 后期人员调整负责商品订单模块，对商品库存以及订单过期分账回补库存等业务开发
- CodeReview **/** 审核代码合并请求，指出其他同学在开发代码中和优化问题，对项目方案进行整体把关

- **TECHNICAL POINTS**

- 灵活使用「策略+工厂+模版」设计模式进行对用户统一登录注册功能和消息中心进行开发设计，实现多渠道登录注册业务和多渠道发送信息，大大降低代码复杂程度提高可读性和拓展性，一套接口，多处实现，方便后期拓展渠道与业务
- 使用RocketMQ实现异步通信、解耦系统组件、实现分布式事件系统，服务与用户任务/行为模块，多处服务行为触发后进行统一消费处理，以及保障失败重试和使用数据库操作保障消费的幂等性，实现该服务模块的高吞吐量，高可靠性，防止业务高并发场景下对服务产生高负载或者打崩服务的情况
- 引入APISIX替换Spring Cloud Gateway，由于业务发展后期发现在高并发场景下性能相对较低，因为它使用了Reactor异步编程模型，而不是传统的阻塞式IO。虽然它在并发连接数方面表现良好，但在单个请求的吞吐量上可能相对较低。所以将其废弃，并且引入使用了OpenResty架构性能更强的APISIX，它具有灵活的路由、负载均衡、熔断、限流、认证授权、动态配置等特性，并且可以通过插件进行功能扩展和定制化，在其之下使用了它的动态配置etcd开发动态用户黑名单功能实现实时拉黑用户功能，并且定制开发授权Token验签插件，用于APISIX中的路由配置中，开发中参与多次开源社区讨论以及bug反馈
- 设计业务用户埋点数据模型，利用ClickHouse列式数据库管理系统，进行打点记录以及统计，因为其的并行查询处理能力提高查询性能和吞吐量，业务服务使用Pulsar配合前端进行消息传递分批上报，并在其业务中进行打点数据统计实现多维度的数据展示
- 利用MongoDB单文档级别的原子性操作能力和多文档事务功能实现商品库存扣减以及回补功能
- 集成对接微信第三方开放平台，实现统一管理众多公众号和小程序的业务功能

> - 熟悉实用Arthas进行对线上服务进行诊断性能/bug排除，达到快速解决问题以及优化服务接口性能

- **个人收获:**

> 提升了我对系统架构的大局观的思考和微服务分布式架构的更加深入的了解以及不同业务划分不同的服务。使我在现实商城中高并发高流量的场景中处理方案得到更多的实践，对场景有更完善的考虑与处理方案;提升了对SAAS多租户系统设计能力;提升会员系统的设计;提升了代码重构和设计模式的能力;在我主要负责的用户中心/消息中心中实践融合了多种设计模式架构，使代码可复用性/拓展性提升，以及在其中提高了对业务的设计能力。

贪吃商城Disco社区项目

- **分布式微服务架构**

> Golang / Kratos / Protobuf / Kafka / Redis / MongoDB / prometheus / goscrew / OpenAPI / Swagger / Wire / Docker / XxlJob / APISIX

- **项目概括**

> 吃喝玩乐的社交平台，与贪吃商城相互结合，为其补齐社交部分，内设吃喝玩乐帖子发布以及评论，贪吃商场商品转卖以及拼单等功能。

- **担任职责**

> - 项目架构设计以及把控，前期Go相关的架构**Kratos**, `Gin` , `go-micro` , `go-zero` 等调研和选择，以及多次进行GO相关开发分析会议，并且技术文档沉淀
> - 负责用户模块/商城核销码转卖功能开发，贪吃商城用户授权注册社区会员功能，会员等级/收藏夹等功能开发
> - 推行内部开源 `goscrew` 螺丝钉开发，类似Java中的 `hutool` 轮子工具包

- **个人收获**

> 在个人时间中学习的 `golang` 得到实践，在公司中进行推行其开发的优点并落地，增加go分布式架构系统的开发经验，开发过程中学习并实践 `DDD` 领域模型开发，高内聚，

> 低耦合到开发模式

唐朝科技(广州)有限公司「后端开发工程师2019年08月-2020年10月」

斑马同校小程序

- **技术架构**

> Spring / SpringMVC / SpringBoot / MybatisPlus / MySql / Maven / Swagger / Lombok / JsonWebToke(JWT) / Shiro

- **项目概况**

> 校园小程序，提供给学生发布代拿快递，代跑购物，代课信息和点外卖功能，并在斑马同校骑手端进行任务抢单和完成新订单，骑手钱包管理，还有校园合伙人功能，提供给合伙人管理该学校的基本权限和功能。

- **担任职责**

> 开发外卖模块的开发担任负责人，对接微信支付，还有微信小程序和公众号的授权，与推送信息，与处理新需求与解决并且优化代码中遗留的问题，并且用Swagger搭建Api文档提高后端与前端开发对接速度。后台审核外卖模块中的商家与商品模块，用户提现审核模块，给后台系统集成Shiro安全框架用于后台多用户权限管理功能。

- **个人收获**

> 熟练运用于对接微信接口，可快速开发与对接，还有对外卖类型的项目有更深层的考虑，从表设计到项目模块落地实现需要考虑的种种问题，学习到Shiro从0到1集成系统并且利用它开发权限管理系统有了自己的了解与认识，对Shiro的框架知识和源码的有了更近一步的了解。

睿妃医美公众号平台

- **技术架构**

> Spring / SpringMVC / SpringBoot / SpringSecurity / MybatisPlus / MySql / Redis / Maven / Swagger2 / Lombok / JsonWebToke(JWT) / Docker

- **项目概况**

> 医美共享队列金平台，消费返利平台

- **担任职责**

> 项目原先是属于PHP开发语言，我负责主导整个项目转变重构成Java语言，利用自己编写的开源脚手架框架**IRON**有maven版本和gradle版本，原有数据库结构分析优化，业务逻辑重新分析优化，并且把项目进行了Docker容器化，使用DockerFile和Git实现CI/CD（DevOps思想），持续构建交付，减少开发后要部署的操作，加快发开发速度，解决项目中共享金队列分佣奖励的繁琐点，使其按规定的用户奖励相应的金额等。

- **个人收获**

> 对Docker理解更深一步，熟系了DockerFile编写规则和编写基本的文件，但是Docker打包优化这块我还有待提高，比如基础镜像需要怎么选择要选着alpine版本等，构建后需要`rm -rf`删除掉构建产生无需的文件减少Docker容器的体积。

广州市网地软件科技有限公司「后端开发工程师2018年04月-2019年08月」

威时沛运货运(广州)有限公司易贸通系统

- **技术架构**

> Spring / SpringMVC / ibatis / JSP / oracle

- **项目概况**

> 配合威时沛运的仓储管理，商品保税，飞机退运等境外物流功能，从前端多元化客户应用，到后端云平台大数据计算，全方位覆盖国际贸易所有环节，自动适应客户个性化偏好，为客户智慧化匹配最优解决方案。

- **担任职责**

> 负责项目的新科宇航模块的费用分摊核算功能，更好的计算出货物在空运费，物流费，税费中分别是多少和不同本币和源币之间的汇率转换，还负责项目中的实际入仓

> 记录模块的导出Excel功能和打印航空标签功能

- **个人收获**

> 对境外物流系统更加深入的了解，体会到了需求理解的重要性，不能盲目开始写代码，一定要弄清楚整个模块的流程之后再进行开发。

ARTICLES / 文章

- 分布式系统架构中使用发号器
- Git分支命名规范
- 微信小程序云开发—云函数连接MySQL
- Collections的singleton,singletonList,singletonMap

OPEN SOURCES / 开源项目

- WeHalo 简约风的微信小程序版博客 

- AgentDroid AI移动设备自动化服务 

THANKS FOR YOUR / 致谢

感谢您花时间阅读我的简历，期待能有机会和您共事。