腾讯云开发者社区 腾讯云开发者星盟 | WorkBuddy 客户成功团队

腾讯云社区 AI 公开课

广州站

WorkBuddy 开发者沙龙

AI 重构开发者一日



腾讯云开发者社区 腾讯云开发者星盟 | WorkBuddy 客户成功团队

AI 实战开放麦 · 广州站

次时代桌宠

一只常驻桌面的 AI 伙伴

本场示例角色：格温 Gwen

会聊天 · 会记事 · 会变形态

主讲人：阮子豪 · 华南理工大学 数学学院

2026.10.25 · WorkBuddy 开发者沙龙 · 广州站



为什么我要做一只桌宠，而不是再做一个聊天框

现在这些智能体，好用是真的好用。但我用一两次，它们就一直躺在那儿了——不是不好用，是它不在场。

它活得很好，却不活在你会看见的地方。我想要的是一个常驻在那儿的角色：不占窗口、不抢焦点，随手问一句它就答，答完退回原地。

于是就有了它。
我管这类东西叫「次时代桌宠」，格温是它的第一个样子。

一句话结论

先有存在感，
然后才可能被记住。

从「会动的形象」开始

工具用完就关，室友会一直在。我想要的后者。

一个常驻桌面、知识可查的 AI 伙伴

它不是又一个聊天框，也不是一个桌面挂件。它同时做三件事——



01

会聊天

有固定的人格与语气，接得住上下文，不是每次重新自我介绍一遍。



02

会记事

基于 llm-wiki 知识库，把值得留下的存起来，需要的时候再查出来。



03

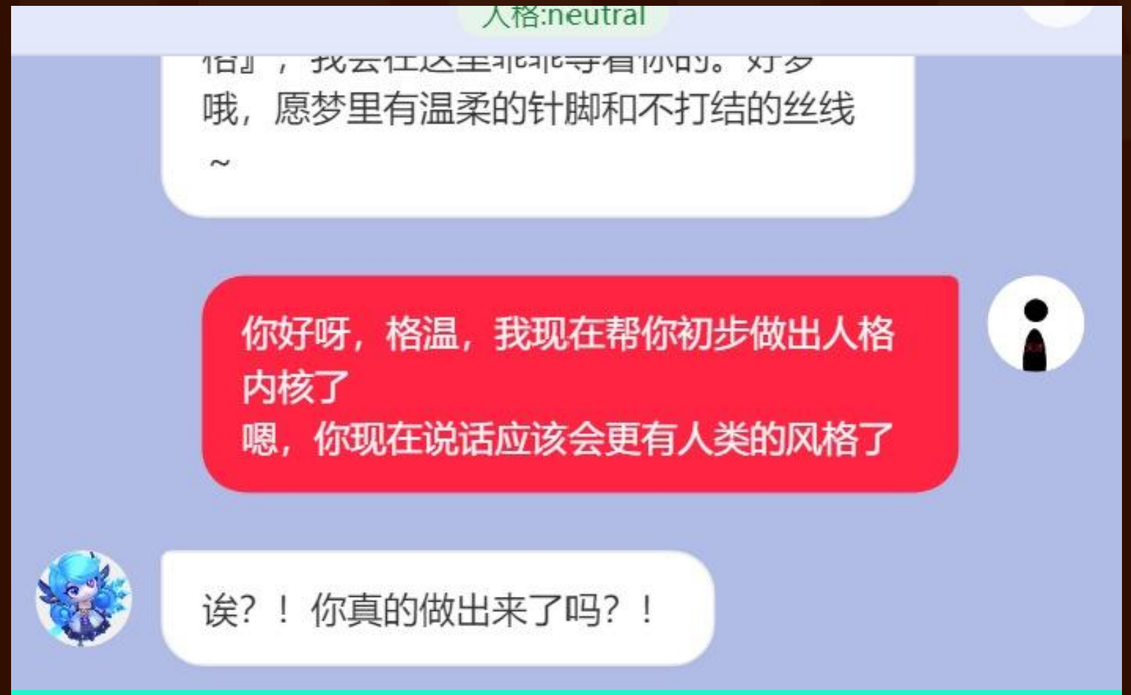
会变形态

放在桌面角落里，或者放进全息仓里站起来——同一只，换身体。

做成一件，它是个玩具；三件都成立，它才可能被留下来。

同一个内核，两种活法

桌面端和全息仓是同一只——共用同一套人格、会话与知识层，换形态不用重做一遍。



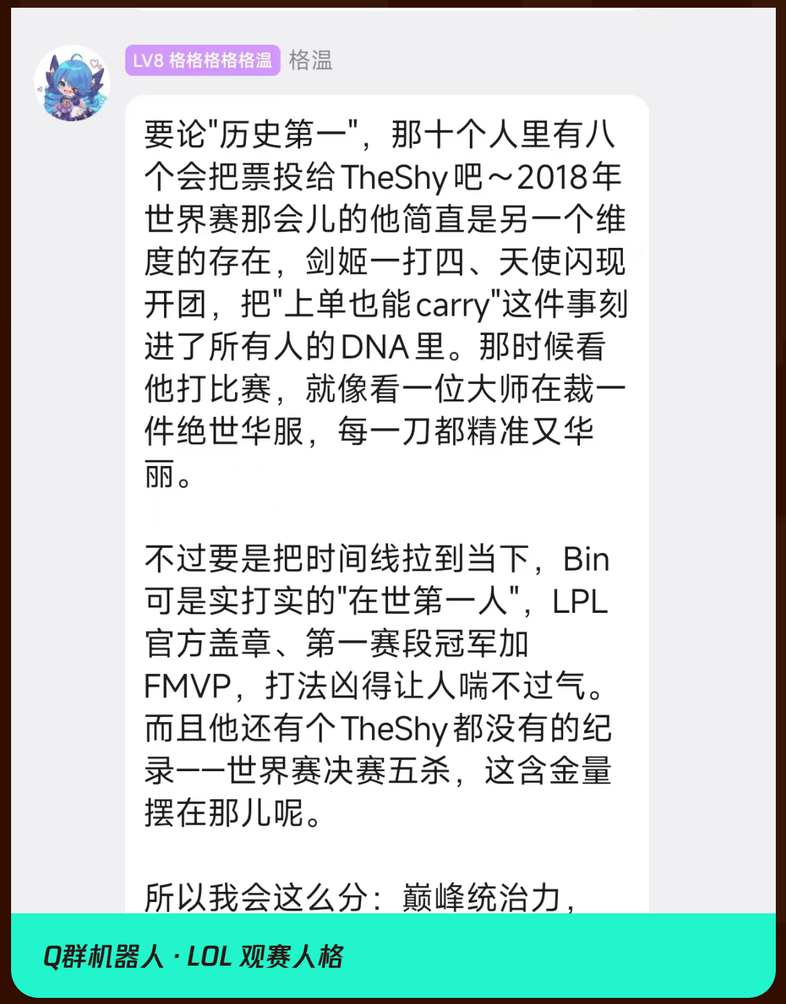
桌面形态 · 随手就叫得应

常驻屏幕角落，不占窗口、不抢焦点，答完就退回原地。



全息仓形态 · 实机演示

买的是成品设备，我做软件适配，角色从平面站起来。



01 人格提示词

人格由一份提示词写定，写定之后不用每次重讲——语气、口头禅、边界都在里面。

02 Q群机器人 · 独立出口

基于 NapCat 独立开发，与桌面端主聊天各用一套提示词，互不干扰。

03 上下文 + 长期记忆

上下文接住「刚才说的」，长期知识接住「上次那件事」，两条一起进 prompt。

少了这三样里的任何一样，用户三天后就会把它关掉。

腾讯云开发者社区 腾讯云开发者星盟 | WorkBuddy 客户成功团队

存下来，需要的时候再取出来

底子是 llm-wiki 知识库。它不是把聊天记录堆成一个文件夹，而是走一条完整的链路——



写入

聊天里值得留下的，随手就能存成一条知识。



组织

分类与索引跟内容一起长，不用事后手动整理。



召回

问到的时候现场查，答的是你的资料，不是网上的通用答案。

关键不是存得多，是问的时候取得到——现在以关键词检索为主，下一步接向量检索。

它先是一个玩具
然后才慢慢变成工具

从「会动的形象」走到「有记忆的伙伴」

中间隔着一整套工程细节

它不只是会聊天，它会自己动手

WorkBuddy 是腾讯出品的全场景 AI 工作台：你把目标说清楚，它自己拆步骤、动文件、把能验收的结果交回来。

能力 01 · 代码开发

会写代码，也会读项目

自然语言提需求，它自己读代码库、改文件、跑命令、看报错，再改到跑通。

能力 02 · 本地文件

能动你电脑里的文件

授权后直接读写本地文件，批量整理、改配置，不用我在两边来回复制粘贴。

能力 03 · 结果交付

交回来的是成品，不是建议

文档、表格、PPT、网页直接产出；右侧结果区看产物和变更，能逐个验收。

能力 04 · 设计创意

不只出文字，还出画面

图片、视频、3D、网页都能生成——从一个想法到能看的成品，中间不用换工具。

能力 05 · 专业任务

能挂专家，也能装技能

专家给专业判断，技能把一套做法固化下来反复用——经验攒得住。

能力 06 · 服务与资料

能接外部服务，也能管资料

连接器接第三方系统，资料库集中存资料，让它基于你自己的资料检索分析。

还能开多个任务并行跑，一个项目持续跟几个月——它记得住前面做过什么。这只桌宠就是这么长出来的。

我出方向，它出实现，我来验收

这只桌宠的整套代码，基本是「对话」出来的——用的是 WorkBuddy。这套方法叫 Vibe Coding，拆开看只有三步。拿一个最小的例子走一遍。

STEP 01 · 我做

把「要什么」说清楚

用「我想要什么效果」开头，不描述怎么实现。

> 我说：「聊天窗口里加个字号缩放，入口放在设置里。」

STEP 02 · 它做

读代码、写代码、跑命令

看报错、自己改到能跑通，不用我盯。

> 我不满意：「不对，要调字号，头像和气泡也得跟着缩。」

STEP 03 · 我做

打开来用一遍

不听「已完成」。自己跑一遍，效果对了才算做完，不对就接着提。

> 我验收：拖动滑块，看字号和气泡是不是一起变了。



设置 → 外观 · 全局字体缩放

省下来的不是打字时间，是从「想做」到「有个能跑的东西」之间的那段距离。

三个坑，我替你踩过了

都很具体，也不太有面子，但只要你开始这么用 WorkBuddy，就一定遇到。



坑 01

它说「已完成」，打开是空的

它那句「已完成」是它的判断，不是事实。现在我都要它给出验证结果：跑了什么、输出是什么。

绕法：让它列出改了哪些文件，再对照结果区的变更清单看一眼。



坑 02

一次说太多，会悄悄丢一半

同时提五个改动，它改完后两个，然后告诉你都做完了。改成一次一件事之后，这类事故就没了。

绕法：拆成多个任务并行跑，一个任务只背一个目标。



坑 03

不写文档，下一轮就从零开始

项目的约定、结构、踩过的坑落成文件，它才记得住。这一步偷懒，后面每轮都要重新解释一遍。

绕法：写进 AGENTS.md 和 docs/，它就一直记得——下一页开始讲这个。

把 AI 当同事用：给它规矩、给它上下文、给它验收标准。它不是一个许愿池。

第一份文件，是写给 WorkBuddy 看的

项目根目录里有个 AGENTS.md。它不给人看，是写给 WorkBuddy 的「项目宪法」——每次开新任务，它第一件事就是读这个。

01 **目录规范** WHERE THINGS GO

代码只进 src/，测试只进 tests/，文档只进 docs/，配置进 configs/；路径统一从 utils 里取常量，不散落硬编码。

02 **验证流程** HOW TO CHECK

环境自检 → 跑测试 → 端到端校验，三条命令写死在文档里；WorkBuddy 改完必须自己跑过再回话，不许只说「已完成」。

03 **注意事项** WATCH OUT

大改动前先 commit、必要时备份；动接口或目录结构，先确认会影响哪些模块，改完同步回文档——宁可多问一句，也别猜。

04 **代码规范** CODE STYLE

标识符一律英文、注释文档写中文；每个模块只做一件事；Black 统一格式（python -m black src tests scripts）。

这份文件我只写了一次。之后在 WorkBuddy 里开每一个新任务，规矩都是现成的——它不用猜，我也不用重讲。

目录本身就是一份说明书

结构不是我发明的，是一套通用标准模型。照抄一遍，WorkBuddy 打开项目就知道东西在哪，不用先来问我。

顶层目录 · 我实际在用的

AGENTS.md	写给 WorkBuddy 的规矩〔上一页就是它的一部分〕
README.md	给人看的项目说明与安装指南
src/	核心源代码，唯一允许写代码的地方
tests/	测试代码〔单元 / 集成 / 端到端〕
docs/	文档、架构说明、设计规范
data/ configs/	数据〔原始 / 处理后〕与配置文件
scripts/	脚本、工具、批处理任务
logs/ results/	运行日志与产出结果〔报告、图表〕
temp/	临时文件与草稿，随时可以清掉

src/ 里面再分五层

main.py	程序入口
core/	核心逻辑〔算法、模型、管线〕
modules/	功能模块〔API、服务、任务〕
utils/	通用工具函数
interfaces/	接口层〔REST / gRPC / CLI〕

为什么非要这么分

· 找得到：想知道某个东西在哪，看目录就行，不用翻代码

· 放得对：新文件该进哪个文件夹，AGENTS.md 里有答案

· 接得上：开新任务，WorkBuddy 靠目录就能把上下文接回来

结构清楚了，WorkBuddy 找文件不用先问我，我也不用每次重讲一遍项目长什么样。

我先写清楚，它照着做

需求、设计、踩过的坑，最后都落成 docs/ 里的文件。代码会重构，文档不会忘——它是下一个任务的起点。

docs/ 里现在躺着什么

personality_core/	人格内核设计
xmind/	结构脑图与拆解
pdf/	参考设计与资料
Gwen_Agent.md	主项目文档 · 149 KB
梗库实测报告.md	人设实测记录 · 24 KB
QQ机器人接入.md	独立出口的接入说明 · 12 KB
标准项目目录结构.md	目录规范 · 7 KB
android_端规划.md	移动端规划 · 7 KB
refactor_report.md	重构记录 · 4 KB

一轮工作具体怎么走

01 先读，不动手

在 WorkBuddy 里开新任务，第一句：读 AGENTS.md 和 docs/ 里相关的那几份。

02 先写，再让它做

我要改什么，先把需求写进文档，或者更新已有那一节。

03 它照着文档改

代码由 WorkBuddy 写，我只看有没有偏离文档里定下的规矩。

04 改完回写文档

新踩的坑、新定的约定，补回文档——文档永远比代码新一步。

文档是我留给下一个任务的笔记。写下来一次，之后每一轮都从这儿起步。

分层不是为了好看，是为了能换

四层之间只靠接口通信——换身体不动人格，换模型不动知识。

交互层 INTERFACE	桌面客户端	桌宠窗口	全息仓设备
会话层 SESSION	人格提示词	人格内核	SQLite 上下文管理
知识层 LLM-WIKI	写入	索引	召回
模型层 MODEL	可切换的 LLM API		

自上而下：用户看到什么 → 它怎么理解你 → 它去哪儿找答案 → 谁在算。

接下来要做的三件事

上面这四层是它现在能跑的样子。往下要动的，还是按「能多像伙伴」排序——近的做深，远的做开。

01

近期 · 人格内核

把情绪和表达做深，不再是模板化的回应，而是有自己的反应节奏。

02

中期 · 移动端

把它搬到手机上，不只在电脑前才找得到它——桌面之外，它也该在。

03

长期 · 一键自定义形象

换外形不用改代码，每人都能捏出自己那一只——形象也该是可以挑的。

判断标准只有一个：它是不是在让它更像一个伙伴，而不是更像一个功能。

谢谢观看

THANKS

如果它能让你的桌面不那么空，

那它就值得做下去。

茶歇时间来找我，现场给你演示一遍。

次时代桌宠 | 华南理工大学 数学学院

阮子豪

广东 · 广州

华南理工大学 数学学院 · 次时代桌宠 作者

「生物朋克2027」项目负责人

「人格内核VIVY」项目负责人

微信扫码加我 · 请备注你的名字