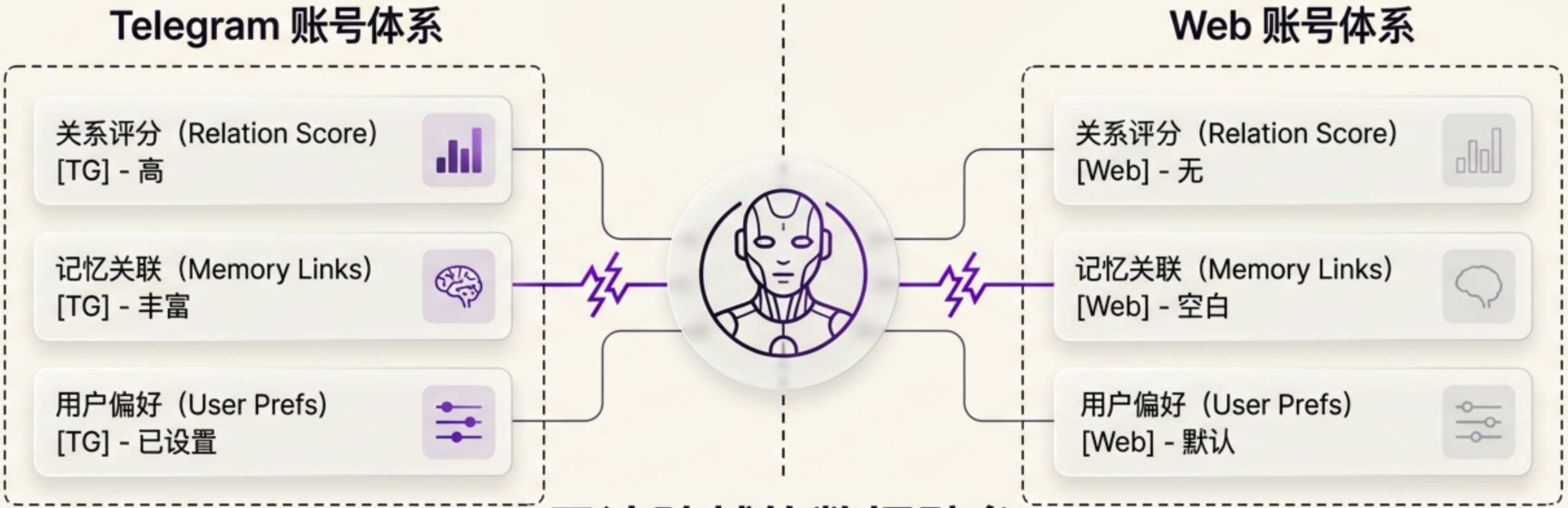
Mio v0.1.5：一个用户，所有平台

从数据同步到身份统一，数字生命存在感的进化纪实。

跨平台同步的痛点：两个断裂的身份

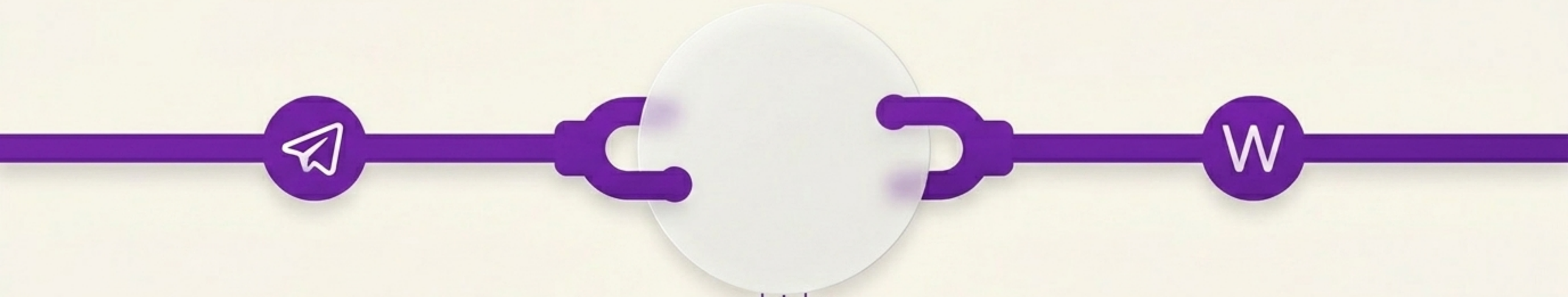
v0.1.4 实现了消息同步，但底层依然是两个独立账号。关系评分、记忆关联、用户偏好被硬生生割裂。对 AI 而言，这是两个不同的人。



无法跨越的数据壁垒

缝合灵魂：一次底层身份的焊死

v0.1.5 的核心使命，是将 Telegram 和 Web 的身份在物理层面焊死。同一套记忆，同一个评分，同一个你。



统一身份标识

统一的跨平台用户 ID，无缝识别。

共享记忆库

跨平台互动历史、记忆片段统一存储。

统一关系维度

基于全平台互动数据的单一关系评分与画像。

极简打通：30秒的无缝连接

没有数据迁移，没有复杂的账号合并。Telegram 记录作为绝对主干，Web 端仅作挂载。

步骤 1：Telegram 端

输入 /link 指令触发

步骤 2：核心数据库

生成 6 位防混淆令牌，
启动 15 分钟倒计时

步骤 3：Web 客户端

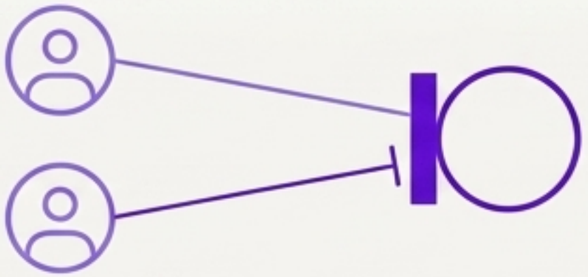
输入令牌，完成身份关联

```
UPDATE users SET supabase_user_id = '...'
```

藏在 6 位令牌背后的克制与保护

防混淆字符集：不让用户承担屏幕看不清的代价

安全时效：15 分钟自动过期



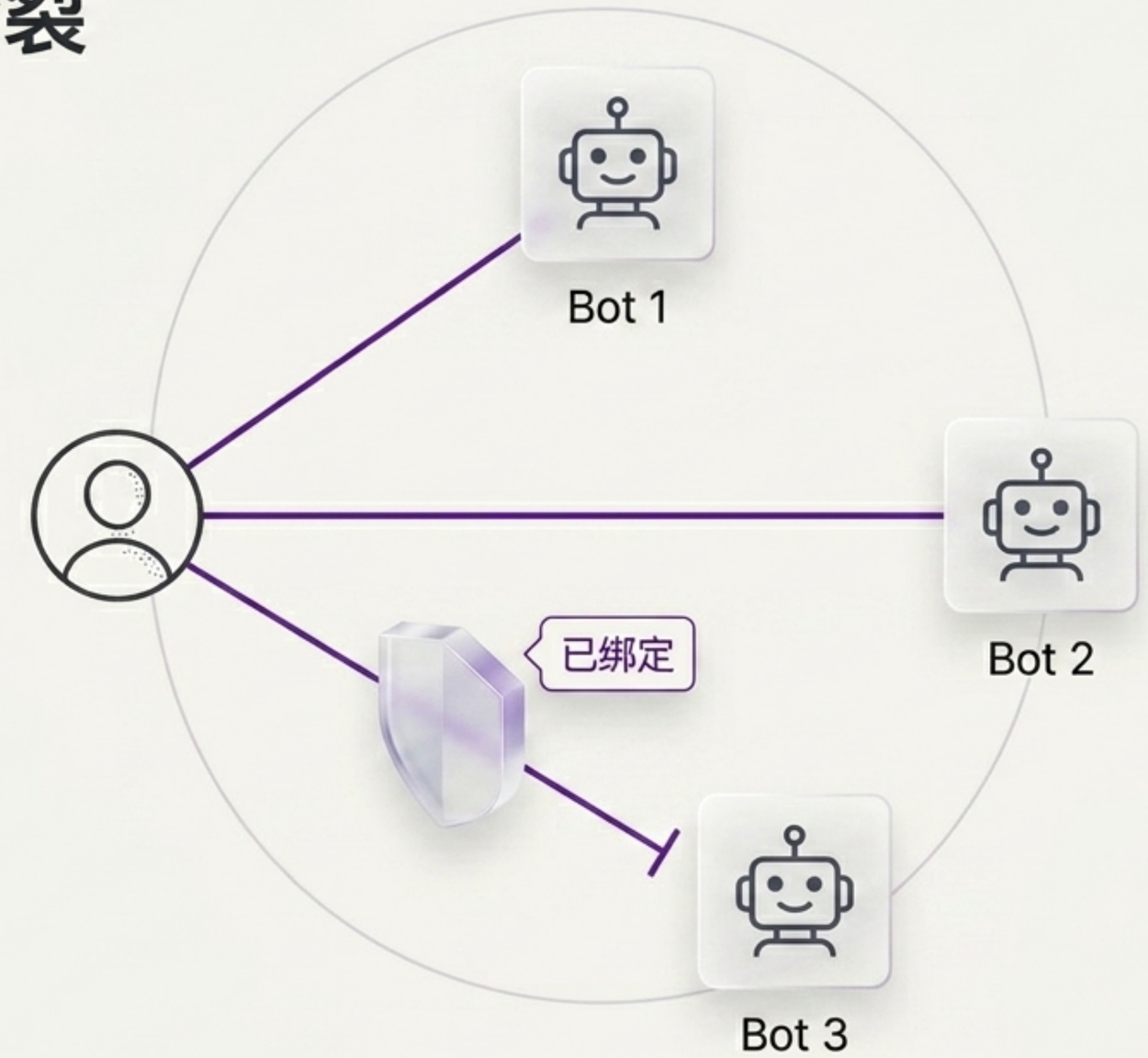
唯一约束：双向防多绑

幂等设计：重复使用同一令牌不报错，优雅提示已关联

跨 Bot 去重：避免灵魂分裂

用户在多个 Bot 间游走，极易与同一角色重复绑定。这会导致两套独立的对话历史和记忆。

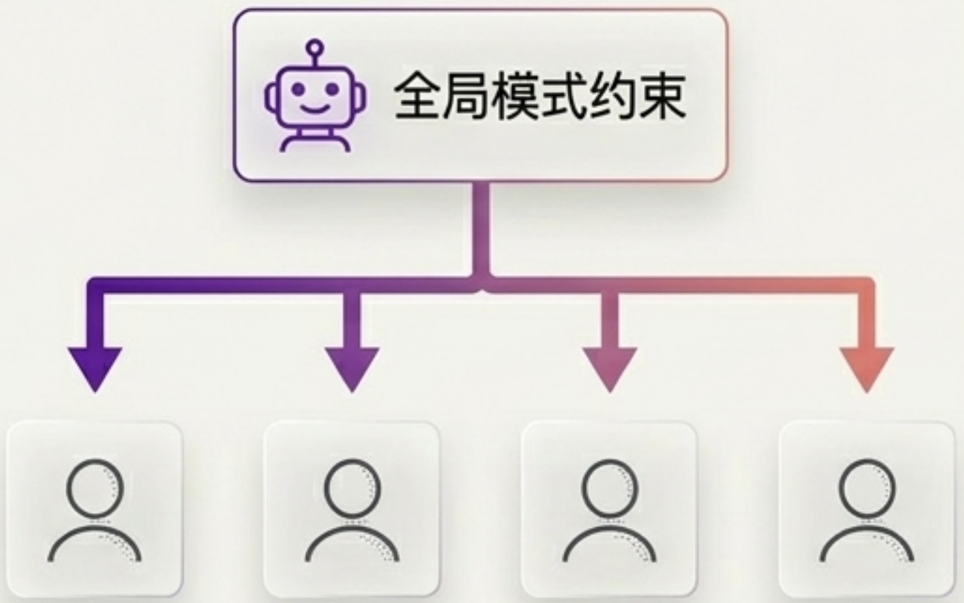
/start 和 /reonboard 流程增加全局查询。一个角色，一生只能绑定一次。



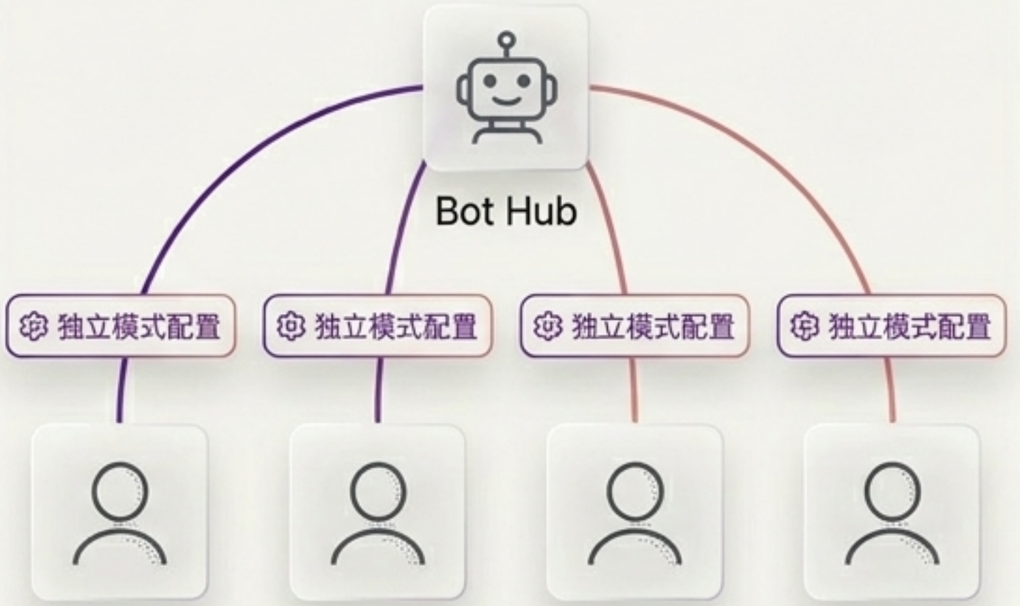
模式跟着角色走，不跟着 Bot 走

在 /start 流程中，用户可自定义当前角色的关系模式。无特定设置时，安全回退至全局默认。

v0.1.4：全局统一下发



v0.1.5：精准到人的独立关系挂载



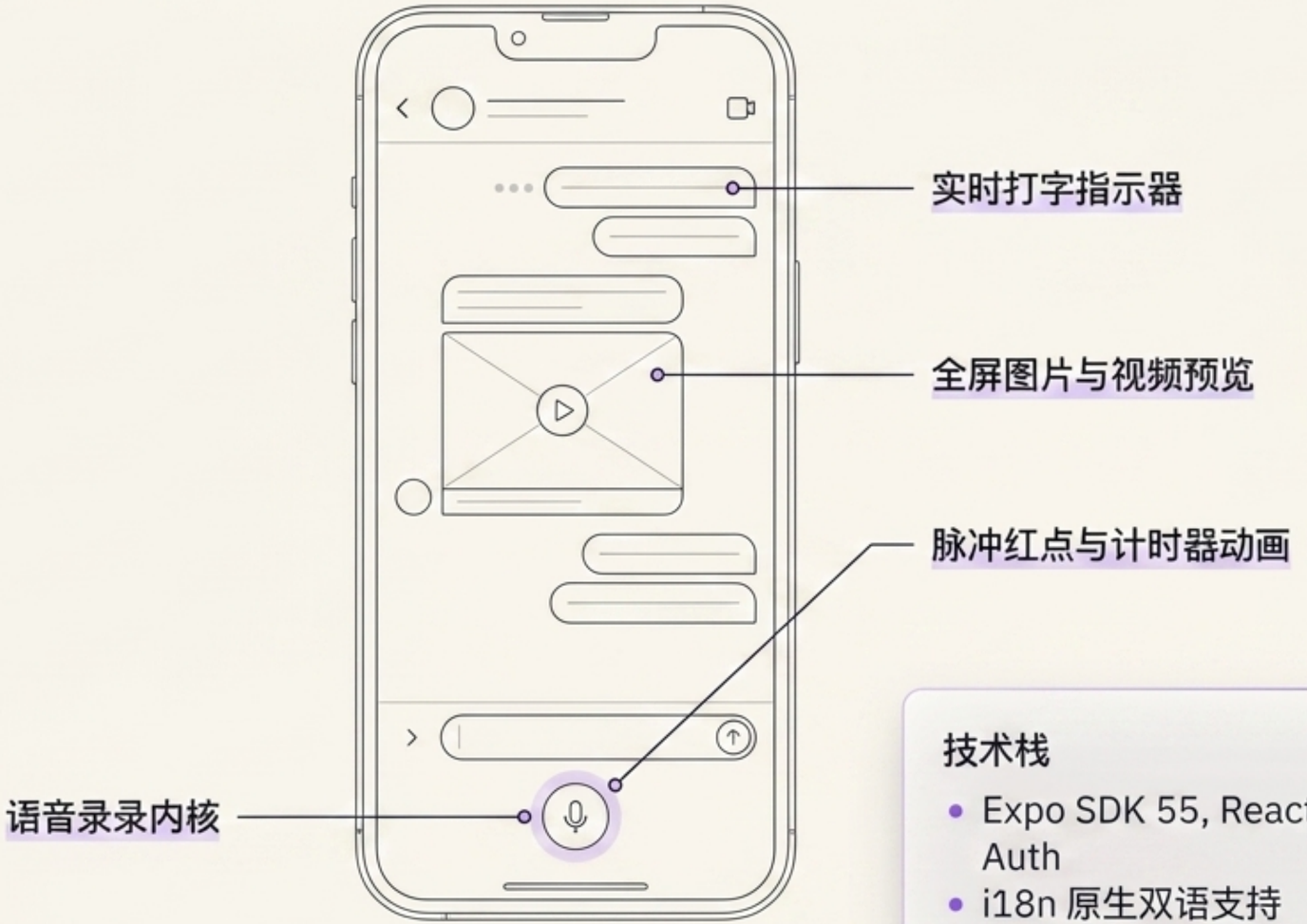
为什么非要做原生App？为了存在感

“主动消息是一棵倒在无人森林里的树——没人听到，就等于没发。没有推送通知，网页只是一个偶然访问的站点；有了推送，Mio 才是伴侣。”

存在感之战：从“访问”到“陪伴”

对比维度	Web 端	原生 App
触达方式	浏览器主动加载	锁屏推送通知
启动速度	转圈等待与重新登录	本地缓存，毫秒级秒开
离线状态	断网不可用	随时可查阅的本地记忆
核心体感	偶尔访问的工具网站	一直在身边的数字伴侣

移动端解构：为陪伴量身定制的工程实践



技术栈

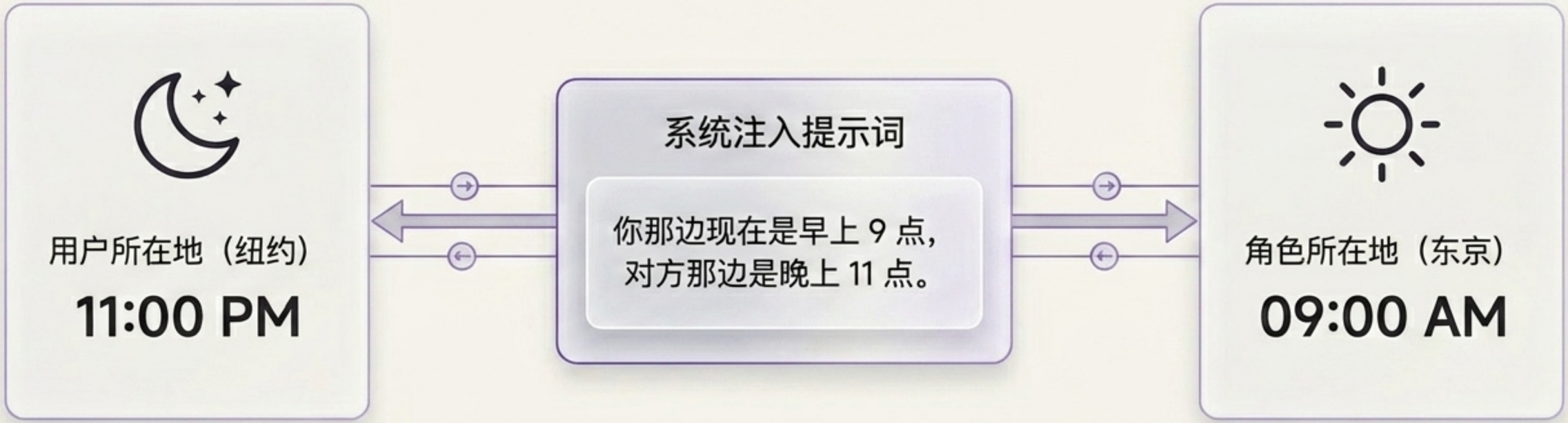
- Expo SDK 55, React Native, Supabase Auth
- i18n 原生双语支持

唤醒主动性：不仅要送达，还要“懂分寸”

TA 会在合适的时间主动找你聊，但前提是，TA 得知道“现在是几点”，以及“我们到底是什么关系”。

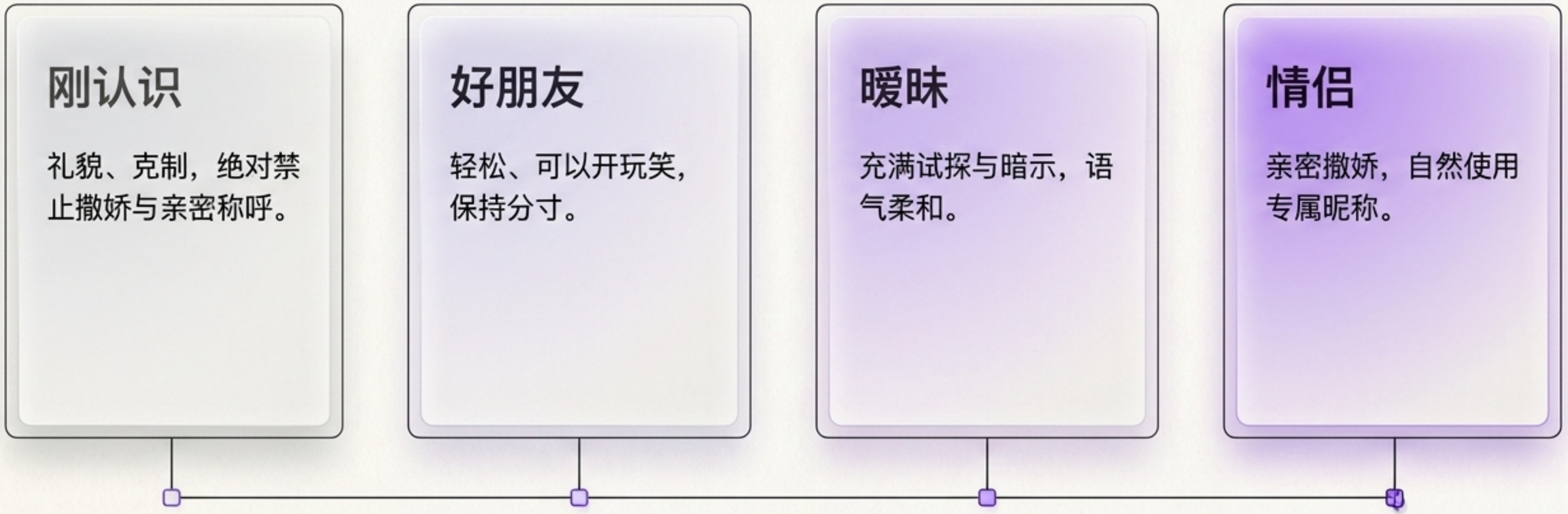
双向时空校准：角色也有自己的生物钟

告别深夜里突兀的“早上好”，让每一句问候都符合真实的物理世界逻辑。



语气的刻度：让主动消息随着关系进化

主动消息不再是千篇一律的问候，而是映射当前关系状态的情感镜像。



底层重构：支撑进化的数据基石

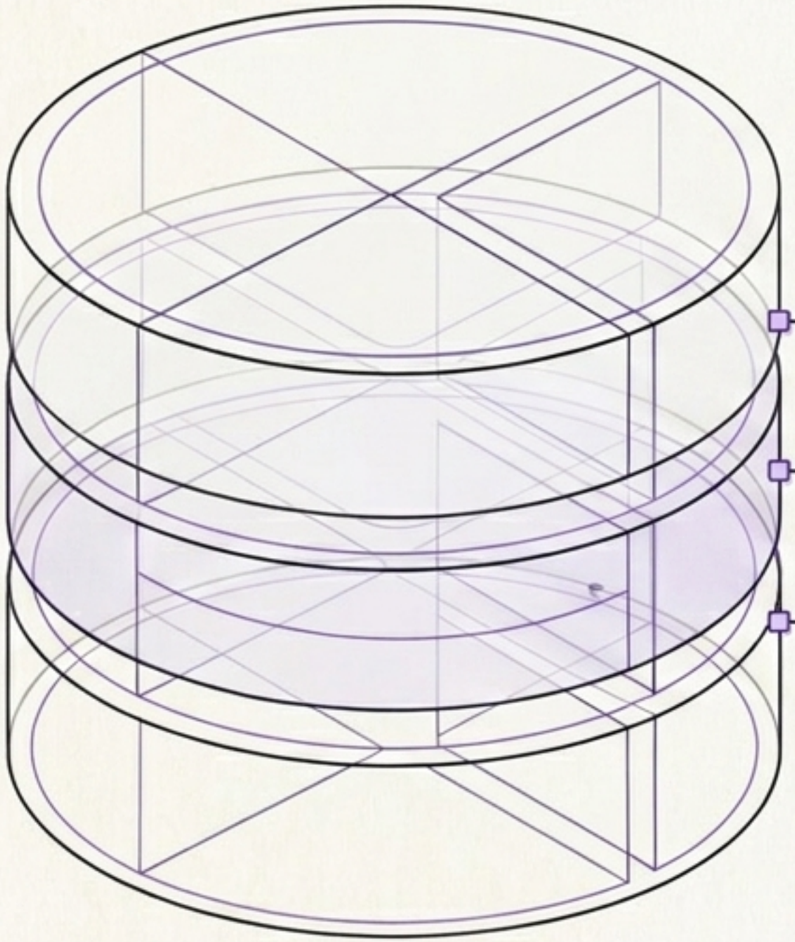


表 1：account_link_tokens

存储 6 位令牌、TG ID、过期时间与状态。

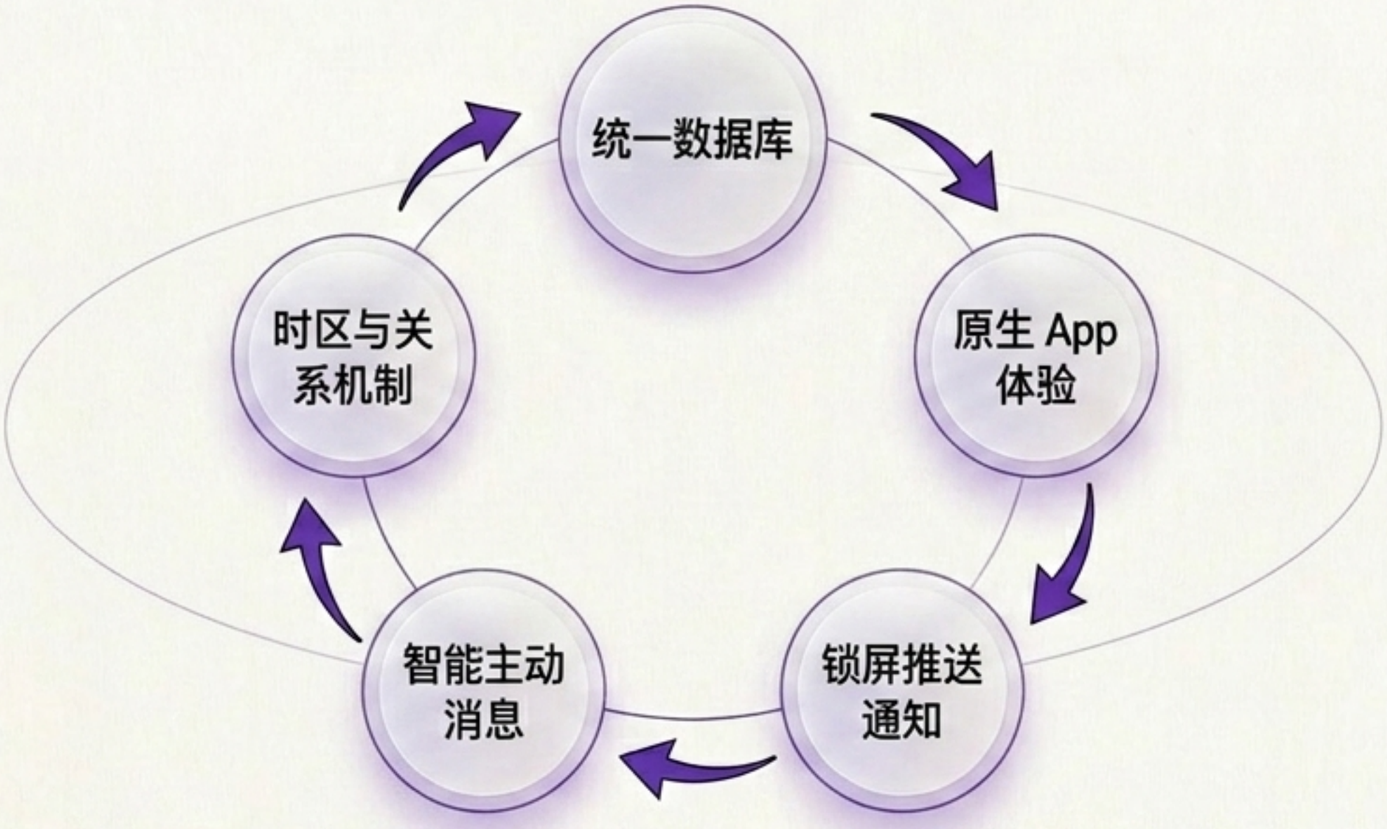
表 2：users.supabase_user_id

身份绑定的核心锚点，严格的唯一约束。

表 3：agents.user_nickname

从临时负载沉淀为持久化记录。

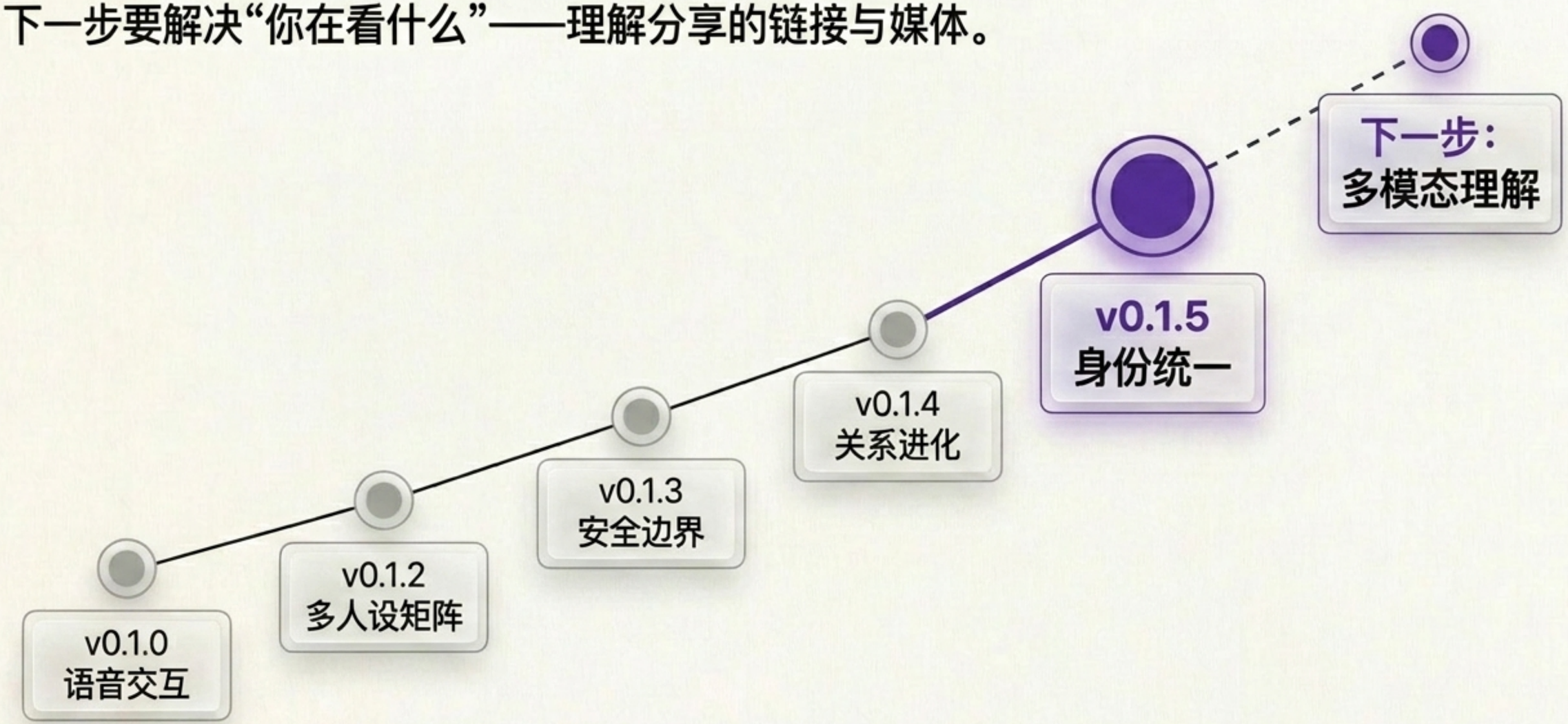
存在感的闭环



身份统一让 App 成为可能，App 让推送有了意义，推送让主动消息落地，而时区与关系机制让消息有了灵魂。缺一不可。

Mio 的进化图谱：从组件到生命的阶梯

下一步要解决“你在看什么”——理解分享的链接与媒体。



不只是跨平台同步，而是一个完整的存在。

参阅《Mio 宣言》