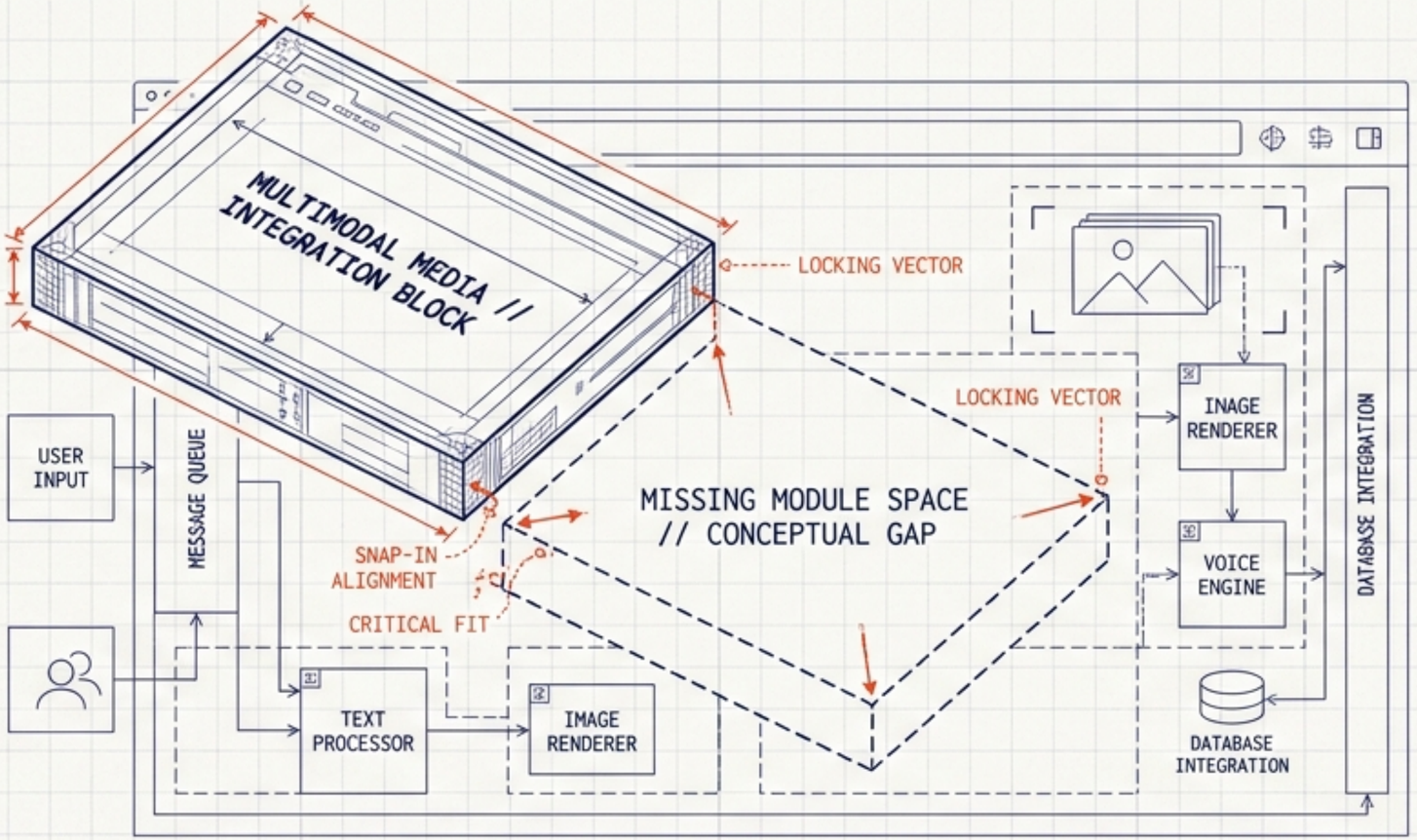
v0.0.5 补齐多模态的最后一块



[STATUS] PUBLISHED // [TAGS] AI, AGENTS, MIO

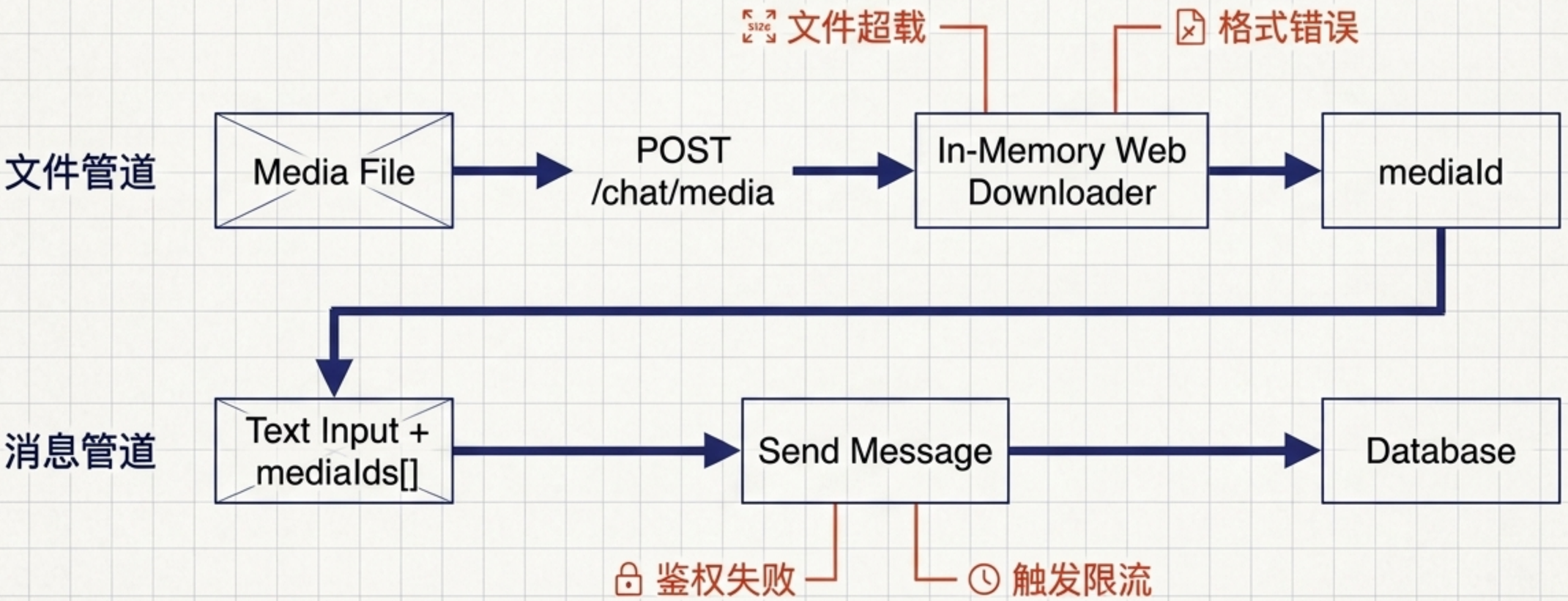
38_FILES // 17_SEC_FIXES // 28_MIN

一台没有摄像头的手机

	Telegram Mobile	Mio Web v0.0.4
[TEXT] 文字	✔ [OK]	✔ [OK]
[STREAM] 流式回复	✔ [OK]	✔ [OK]
[PHOTO] 图片	✔ [OK]	✘ [ERR]
[VOICE] 语音	✔ [OK]	✘ [ERR]
[EMOJI] 表情	✔ [OK]	✘ [ERR]

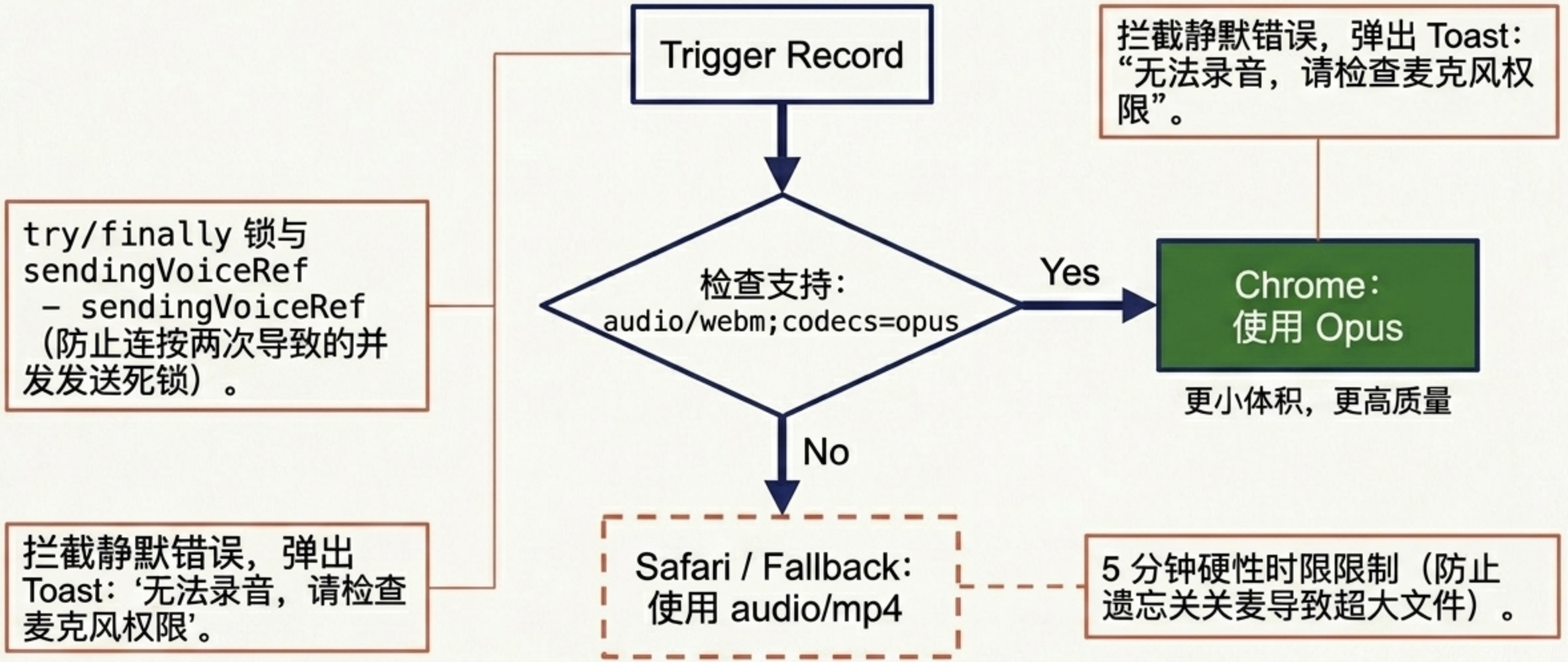
Web 端 UI 虽已重建，但在此之前只能打字。它像是一个结构完整，却无法说话、无法看见的残缺系统。v0.0.5 将彻底填平这个体验鸿沟。

核心架构决策：两阶段上传隔离风险

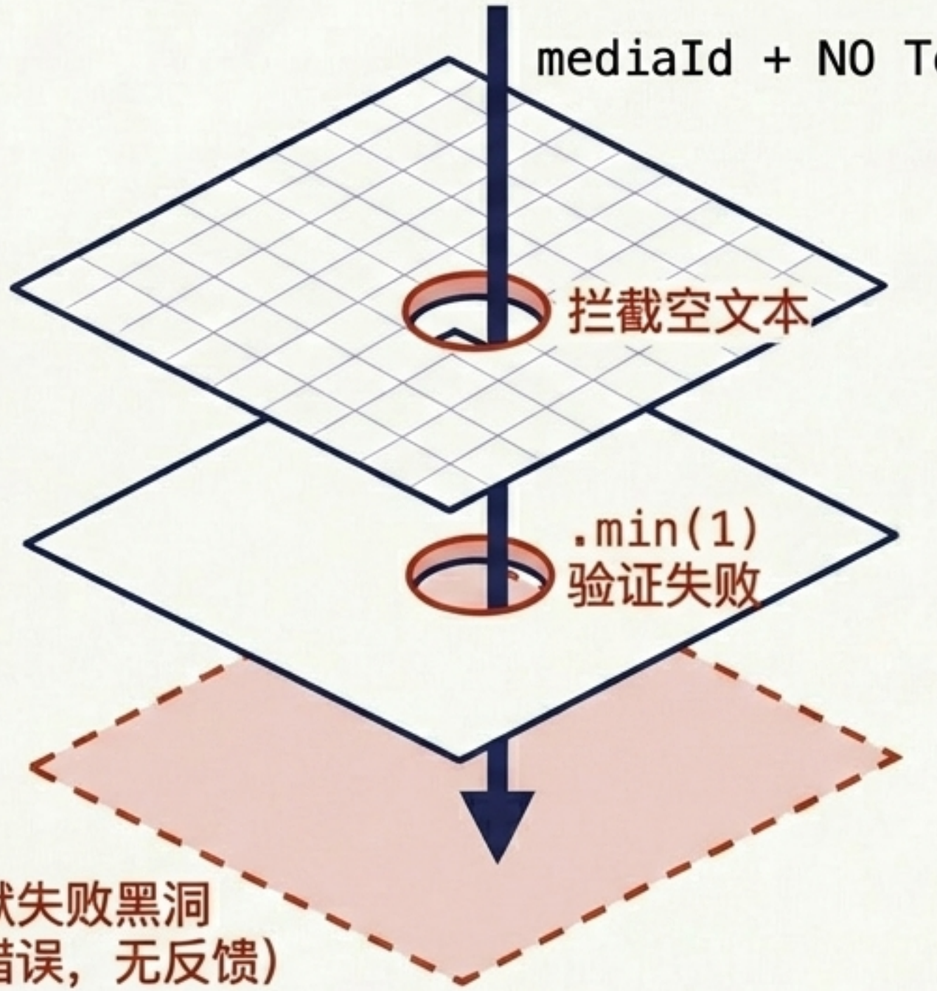


核心优势：一条管道处理两种输入。复用已有的 Telegram processMedia() 逻辑，实现零重复代码。分离上传与发送，允许优雅的错误处理和发送前预览。

语音录制：跨浏览器兼容的雷区



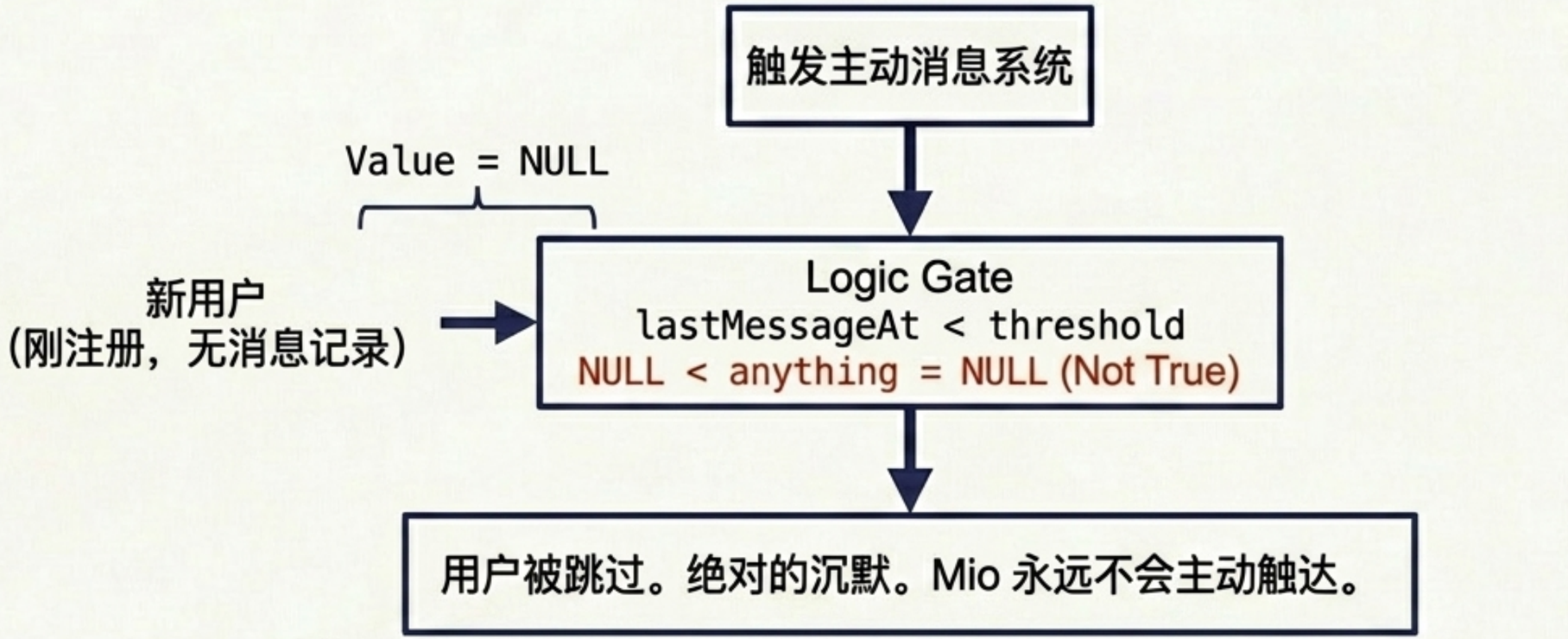
深度解剖：消失的语音消息



修复 1：后端更新为 .refine()（存在 mediaIds 时接受空文本）

修复 2：消除竞态条件，移除 setTimeout(500)，直接返回 MediaAttachment 对象以实现即时同步。

心跳 Bug：致命的第一印象



修复方案仅仅是一行代码：加注 `OR lastMessageAt IS NULL`

缺失这一行，系统对所有新用户的首次主动触达将永远失效。最微小的代码漏洞，造就了最差的产品体验。

基础设施：从开发脚本到真实业务

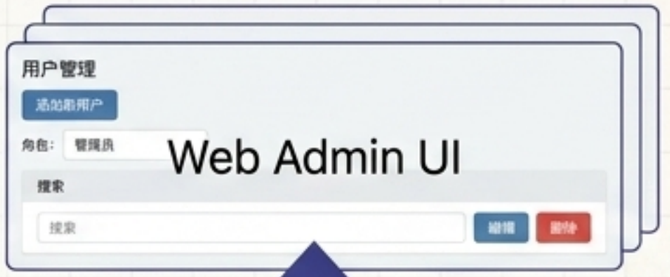
Before

.env

ALLOWED_TELEGRAM_IDS

添加/移除用户需重新部署
(Re-deploy required)

After



In-Memory Cache

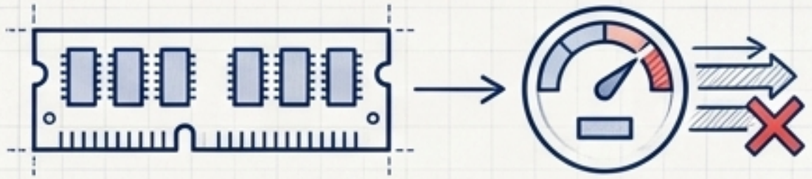
telegram_allowlist

users.role (Migration 0003)

“开发者工具”与“真实产品”的分水岭。提供完整的 GET/POST/DELETE API，由 Admin Guard 中间件保护，保留环境变量作为纯数据库为空时的兜底 (Fallback)。

两轮完整安全审计与 17 项防御加固

内存保护（Memory）

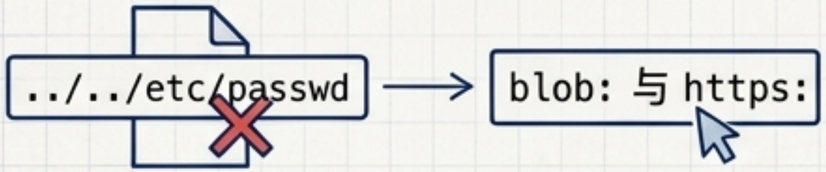


500MB 全局内存上限 + 每用户最多 3 个待发文件，拦截恶意资源消耗。

身份校验（Identity）

Web downloader 闭包捕获 `userId`，严格校验媒体文件所有权，拒绝暴力猜测 `mediaId`。

数据清洗（Sanitization）



文件名消毒拦截 `../.../etc/passwd` 路径遍历；URL Scheme 严格限制为 `blob:` 与 `https:` 。

深度验证（Validation）

弃用扩展名信任，引入 `file-type` 库进行底层 Magic Byte MIME 签名校验 + 反射错误截断防信息泄露。

并行审计 Agent 逐文件扫描。17 个修复项层层叠加，构筑“能跑”到“能安全上线”的鸿沟。

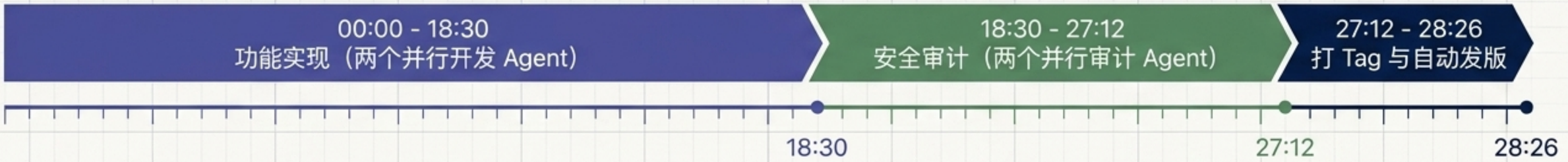
极限开发：28 分钟的奇迹

3,009 行代码新增

38 个文件改动

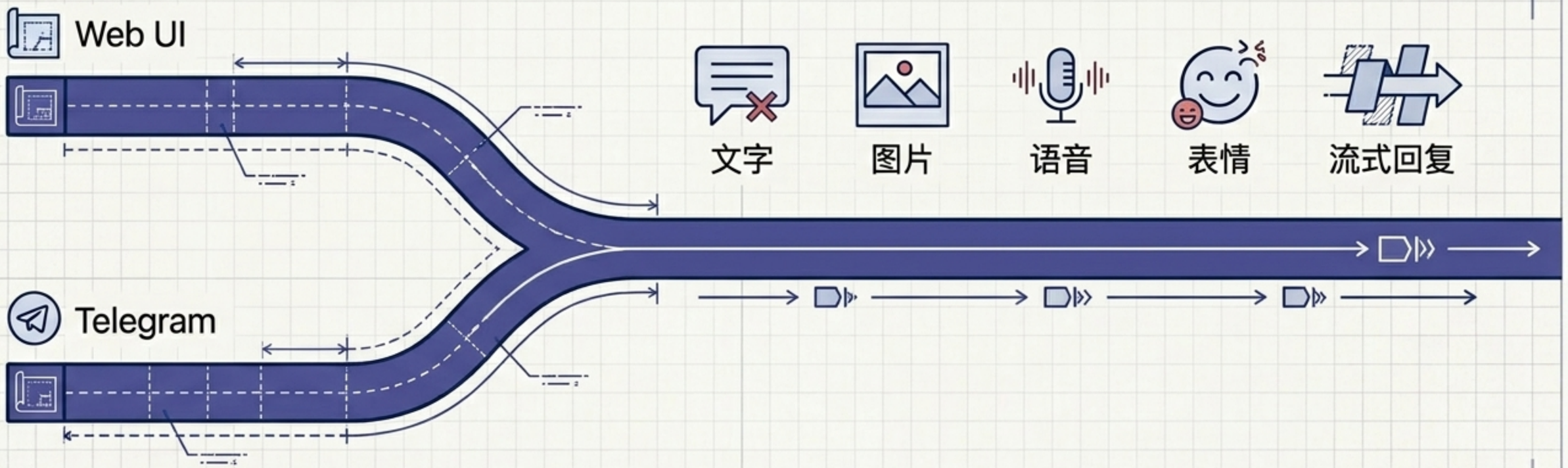
17 项安全修复

12/12 测试通过



引入懒加载表情选择器（@emoji-mart/react）、两阶段媒体上传、完整数据库后台。
全部工程仅耗时 28 分钟。

统一的语言，全新的起点



媒体支持是最后的缺口。现在，Web 端不再是二等公民。两个渠道在核心交互上完全对齐。

共同语言的基石已然筑就，通往跨渠道高级特性的道路正式开启。