Ascent Creator 技术架构文档

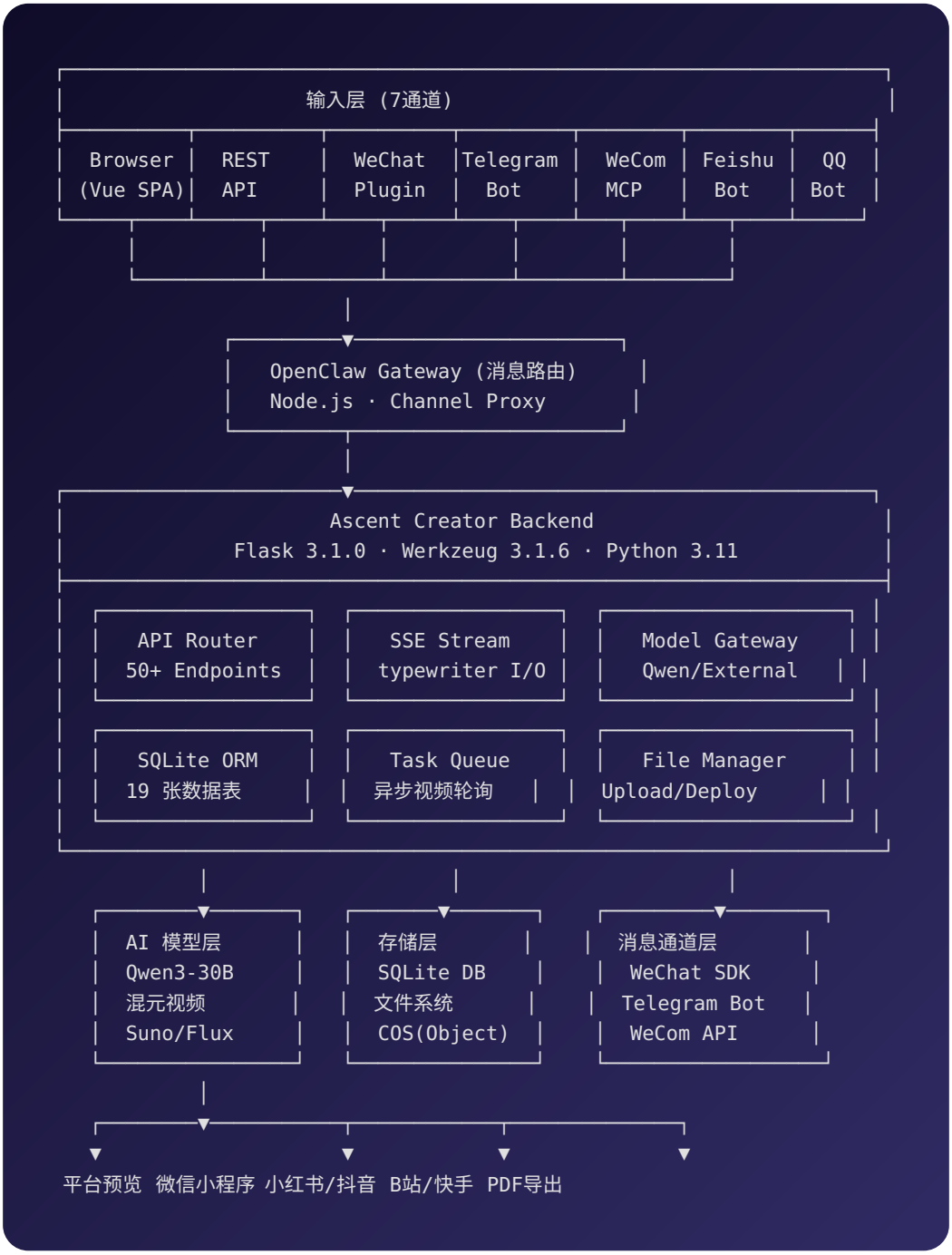
代码结构 · 后端交互 · 数据库设计 · 技术栈 · 输入输出流

⚙ 版本 v1.0 2026-05-31 初稿

文档摘要

本文档从纯技术视角出发，详细阐述 Ascent Creator 平台的技术架构设计。涵盖前端构建体系、后端API规范、数据库ER设计、AI模型网关封装、SSE流式通信协议、消息路由机制、部署拓扑、以及从输入到输出的完整数据流。目标是为开发者和技术维护者提供完整的系统理解和技术参考。

一、技术全景图


输入层 (7通道)
Browser (Vue SPA)
REST API
WeChat Plugin
Telegram Bot
WeCom MCP
Feishu Bot
QQ Bot
OpenClaw Gateway (消息路由)
Node.js · Channel Proxy
Ascent Creator Backend
Flask 3.1.0 · Werkzeug 3.1.6 · Python 3.11
API Router
50+ Endpoints
SSE Stream
typewriter I/O
Model Gateway
Qwen/External
SQLite ORM
19 张数据表
Task Queue
异步视频轮询
File Manager
Upload/Deploy
AI 模型层
Qwen3-30B
混元视频
Suno/Flux
存储层
SQLite DB
文件系统
COS(Object)
消息通道层
WeChat SDK
Telegram Bot
WeCom API
平台预览
微信小程序
小红书/抖音
B站/快手
PDF导出

二、技术栈清单

2.1 前端技术栈

技术	版本	类型	用途说明
Vue 3	3.5.12	框架	Composition API + `<script setup>`，构建复杂 SPA 交互
Vite	5.4.10	构建工具	极速 HMR、按需编译、ESM 原生模块
Vue Router	4.6.4	路由	13 条前端路由，懒加载页面组件
Pinia	3.0.4	状态管理	全局侧边栏状态、页面状态
Tailwind CSS	3.4.17	CSS 框架	原子化 CSS，自定义暗色主题 + 霓虹色系
Lucide Vue Next	1.0.0	图标库	轻量图标，Tree-shaking 友好
Axios	1.16.1	HTTP	API 请求封装，拦截器处理
PostCSS	8.5	CSS 后处理	Autoprefixer + Tailwind 插件
Tailwind Animate	1.0.7	动画	脉冲发光、渐变等自定义动画

2.2 后端技术栈

技术	版本	类型	用途
Flask	3.1.0	Web 框架	REST API，50+ 端点，SSE 流式支持
Werkzeug	3.1.6	WSGI 工具	Flask 底层 WSGI 实现
Flask-CORS	6.0.2	跨域	前后端分离跨域支持
SQLite 3	内置	数据库	零配置关系型数据库，19 张表
Requests	2.33.0	HTTP 客户端	调用外部 AI 模型 API
Gunicorn	23.0.0	WSGI 服务器	生产环境部署（待启用）
Jinja 2	内置	模板引擎	论坛模板渲染
WeasyPrint	68.1	PDF 生成	HTML → PDF 文档导出

2.3 AI 模型对接技术

模型	协议	认证方式	数据格式	调用模式
Qwen3（开源）	OpenAI 兼容	Bearer Token	JSON + SSE	流式 (stream=True)
混元视频（腾讯）	REST	API Key + Key ID	JSON	异步提交+轮询
Suno（音乐）	REST	Cookie + Key	JSON	同步/异步
SDXL / FLUX（图片）	REST	本地直连	JSON/ Binary	同步
外部模型（通用）	OpenAI 兼容	用户配置	JSON + SSE	可配置

2.4 部署与环境

服务器	腾讯云轻量应用服务器，2核2G
操作系统	OpenCloudOS 9 (Linux 6.6)
运行时	Node.js v22.22.0（前端构建）、Python 3.11（后端）
进程管理	nohup + 手动管理（待升级为systemd）
域名/端口	:5000（创作平台）· :5001（论坛）· :3990（Qwen API）

三、完整代码结构

```
Ascent Creator Platform
├── package.json                ← 前端依赖：Vue 3 + Vite + Tailwind
├── vite.config.js              ← Vite 配置：proxy → :5000
├── tailwind.config.js          ← 暗色主题 + 霓虹色系（12种主题色）
├── postcss.config.js           ← PostCSS + Autoprefixer
├── index.html                  ← SPA 入口
│
├── src/                        ← 前端源码（Vue 3 Composition API）
```

```
│   ├── main.js                    ← 入口：挂载App + Router + Pinia
│   ├── App.vue                    ← 根组件
│   ├── style.css                  ← 全局样式：Tailwind + 自定义霓虹玻璃
│   │
│   ├── components/                ← 通用组件
│   │   ├── AppLayout.vue          ← 全局布局（响应式侧边栏）
│   │   ├── ShareDialog.vue        ← 分享弹窗
│   │   └── KnowledgeBasePanel.vue ← 知识库面板
│   │
│   ├── views/                     ← 页面组件（14个工坊/页面）
│   │   ├── Dashboard.vue          ← 仪表盘（统计卡片+快捷入口）
│   │   ├── WritingStudio.vue      ← AI写作工坊（35KB，最重）
│   │   ├── ChatStudio.vue         ← AI聊天助手（34KB，SSE流式+多媒体）
│   │   ├── NovelStudio.vue        ← AI小说工坊（30KB，五步设定+流式章节）
│   │   ├── CodeStudio.vue         ← AI应用工坊（22KB，代码沙箱+部署）
│   │   ├── VoiceStudio.vue        ← AI语音工坊（19KB，TTS播放）
│   │   ├── AvatarStudio.vue       ← 数字人工坊（14KB，VRM/Live2D）
│   │   ├── VideoStudio.vue        ← AI视频工坊（14KB，异步+轮询）
│   │   ├── Gallery.vue            ← 作品广场（14KB，Grid展示）
│   │   ├── MusicStudio.vue        ← AI音乐工坊（12KB，Suno集成）
│   │   ├── AssetLibrary.vue       ← 资料库（8KB）
│   │   ├── ImageStudio.vue        ← AI图片工坊（8KB，SDXL/Flux）
│   │   ├── ModelHub.vue           ← 模型管理（5KB，配置自定义模型）
│   │   └── CodeStudio-backup.vue  ← 旧版备份
│   │
│   ├── router/index.js            ← 13条路由，懒加载
│   ├── services/api.js            ← Axios封装，所有API调用
│   └── stores/app.js              ← Pinia Store：侧边栏状态
│
├── backend/                       ← 后端源码（Flask 3.1.0, 2828行）
│   ├── app.py                     ← 主应用（50+API端点，SSE流式）
│   ├── config.py                  ← 模型网关配置（159行）
│   ├── configs/
│   │   ├── video_api.json         ← 混元视频 Key/Endpoint
│   │   └── suno_api.json          ← Suno音乐 Key/Endpoint
│   ├── knowledge/                 ← 小说知识库（用户上传）
│   │   └── default/
│   ├── generated/                 ← 生成文件缓存
│   ├── uploads/                   ← 用户上传文件
│   └── deployed/                  ← 已部署的应用
│
├── dist/                          ← 构建产物（~500KB gzip）
│   └── assets/
│       ├── index-xxx.js           ← 主包（~124KB gzip: 47KB）
│       ├── api-xxx.js             ← API层（~43KB gzip: 17KB）
│       ├── ChatStudio-xxx.js      ← 聊天助手（~24KB）
│       ├── NovelStudio-xxx.js     ← 小说工坊（~22KB）
│       └── ...其他视图懒加载~
```

```
│
├── forum/                          ← 论坛系统（位于 workspace）
│   └── forum.py                    ← 3729行，独立Flask实例
│
├── forum.db                        ← SQLite 数据库（19张表）
├── downloads_whitelist.json        ← 下载白名单
│
├── .vscode/extensions.json         ← 推荐扩展
└── README.md
```

四、后端交互详解

4.1 API 端点总表（50+ 端点）

分组	方法	端点	说明
□聊天	GET	/api/chat/characters	获取角色列表
	POST	/api/chat/characters	创建角色
	GET	/api/chat/characters/:id	角色详情
	PUT	/api/chat/characters/:id	更新角色
	DELETE	/api/chat/characters/:id	删除角色
	POST	/api/chat/completions	□ SSE 流式聊天
□小说	POST	/api/novel/create	创建小说
	GET	/api/novel/list	小说列表
	GET	/api/novel/:id	小说详情
	POST	/api/novel/:id/setup	保存设定（5步向导）
	GET	/api/novel/:id/chapter/stream	□流式章节生成
	POST	/api/novel/:id/character	创建/批量生成角色
	POST	/api/novel/:id/delete	删除小说（含章节角色）
	GET/POST	/api/novel/:id/knowledge	知识库管理
	POST	/api/novel/:id/ai-suggest	AI 建议

分组	方法	端点	说明
□ 视频	POST	/api/video/generate	提交视频生成（异步）
	GET	/api/video/status/:task_id	查询生成状态
	GET	/api/video/history	历史记录
	GET/POST	/api/video/config	API 配置管理
□ 音乐	POST	/api/music/generate	生成音乐
	GET	/api/music/list	音乐列表
	GET/POST	/api/music/config	音乐配置
□ 数字人	GET	/api/avatar/models	模型列表
	GET	/api/avatar/list	数字人列表
	POST	/api/avatar/create	创建数字人
□ 部署	POST	/api/deploy/html	部署 Web 应用
	GET	/api/deploy/list	部署列表
□ 作品	GET	/api/gallery/works	作品列表
	POST	/api/gallery/work/:id/like	点赞
	POST	/api/gallery/work/share	分享
⚙	GET	/api/health	健康检查

4.2 SSE 流式通信协议

平台的核心交互方式之一是 SSE（Server-Sent Events）流式输出，用于实现「打字机效果」。以下是完整协议规范：

```
// 请求格式（前端 → 后端） POST /api/chat/completions Content-Type: application/
json Accept: text/event-stream { "messages": [ {"role": "system", "content":
"角色设定..."}, {"role": "user", "content": "你好"} ], "model":
"qwen3_coder_30b_a3b_instruct...", "stream": true, "character_id": 1 // 可选,
指定聊天角色 } // 响应格式（后端 → 前端，SSE逐字流） data: {"content": "你"} data:
{"content": "好"} data: {"content": "！"} data: {"content": "我"} data:
{"content": "是"} data: {"content": "As"} data: {"content": "cent"} data:
{"content": "小"} data: {"content": "助"} data: {"content": "手"} data:
{"content": "。"} data: {"content": "\n"} data: {"content": "有"} data:
```

```
{"content": "什"} data: {"content": "么"} data: {"content": "可"} data:
{"content": "以"} data: {"content": "帮"} data: {"content": "你"} data:
{"content": "的"} data: {"content": "吗"} data: {"content": "？"} data:
[DONE] // 流结束标记 // 前端解析逻辑 const reader = resp.body.getReader() const
decoder = new TextDecoder() let buffer = '' for(;;) { const {done, value} =
await reader.read() if (done) break buffer += decoder.decode(value, {stream:
true}) for (const line of buffer.split('\n')) { if (line.startsWith('data: '))
{ const data = line.slice(6).trim() if (data === '[DONE]') continue const
parsed = JSON.parse(data) const content = parsed.content // □ 直接取 content 字
段 // ↑ 注意：不是 choices[0].delta.content (OpenAI格式) updateUI(content) } } }
```

4.3 视频生成异步轮询

腾讯混元视频 API 使用异步提交 + 轮询模式：

```
Step 1: 提交任务 POST https://tokenhub.tencentmaas.com/v1/api/video/submit
Authorization: Bearer sk-xxx X-Api-Key-Id: ak-xxx { "model": "hy-video-1.5",
"prompt": "一只可爱的小猪在唱歌" } → Response: {"id": "145255...", "status":
"queued"} Step 2~N: 轮询状态（每10-30秒） POST https://tokenhub.tencentmaas.com/
v1/api/video/query { "model": "hy-video-1.5", "id": "145255..." } → Response:
{"status": "in_progress"} Step N+1: 完成 → Response: { "status": "completed",
"data": { "url": "https://cos.ap-guangzhou...mp4" } } Step N+2: 下载并存储 → 下
载视频 → 保存到 ai_video_history → 返回给前端/微信
```

4.4 聊天角色注入机制

AI 聊天助手通过动态 system prompt 注入角色设定：

```
# 后端角色注入逻辑 (app.py) if character_id: row = db.execute("SELECT * FROM
ai_chat_characters WHERE id=?", (character_id,)) char = dict(row) # 构造角色
system prompt system_prompt = '''你正在扮演以下角色，必须完全以这个身份说话。 【角色设
定】 姓名：{name} 外貌：{appearance} 性格：{personality} 背景故事：{background} 对话
风格：{style} 【知识库】{knowledge} ''' # 注入到消息列表首位 messages.insert(0,
{"role": "system", "content": system_prompt}) # 为防止Qwen忽略system角色，追加用户
消息强化 messages.append({ "role": "user", "content": "从现在开始，你是{name}，不是
其他任何AI。请以这个身份和我对话。" })
```

五、数据库设计

5.1 ER 关系图（核心模块）

5.2 核心表 DDL

```
-- AI 小说主表 CREATE TABLE ai_novels ( id INTEGER PRIMARY KEY AUTOINCREMENT,
title TEXT, genre TEXT, theme TEXT, style TEXT, target_audience TEXT,
core_idea TEXT, setting TEXT, outline TEXT, protagonist TEXT, setup_step
INTEGER DEFAULT 0, setup_completed INTEGER DEFAULT 0, current_chapter INTEGER
```

```
DEFAULT 0, total_chapters INTEGER DEFAULT 0, status TEXT DEFAULT 'draft',
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ); -- AI 聊天角色 CREATE TABLE ai_chat_characters ( id
INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, avatar TEXT, personality TEXT,
appearance TEXT, background TEXT, knowledge TEXT, expressions TEXT, greeting
TEXT, style TEXT DEFAULT 'normal', created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP, updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
```

六、模型网关封装

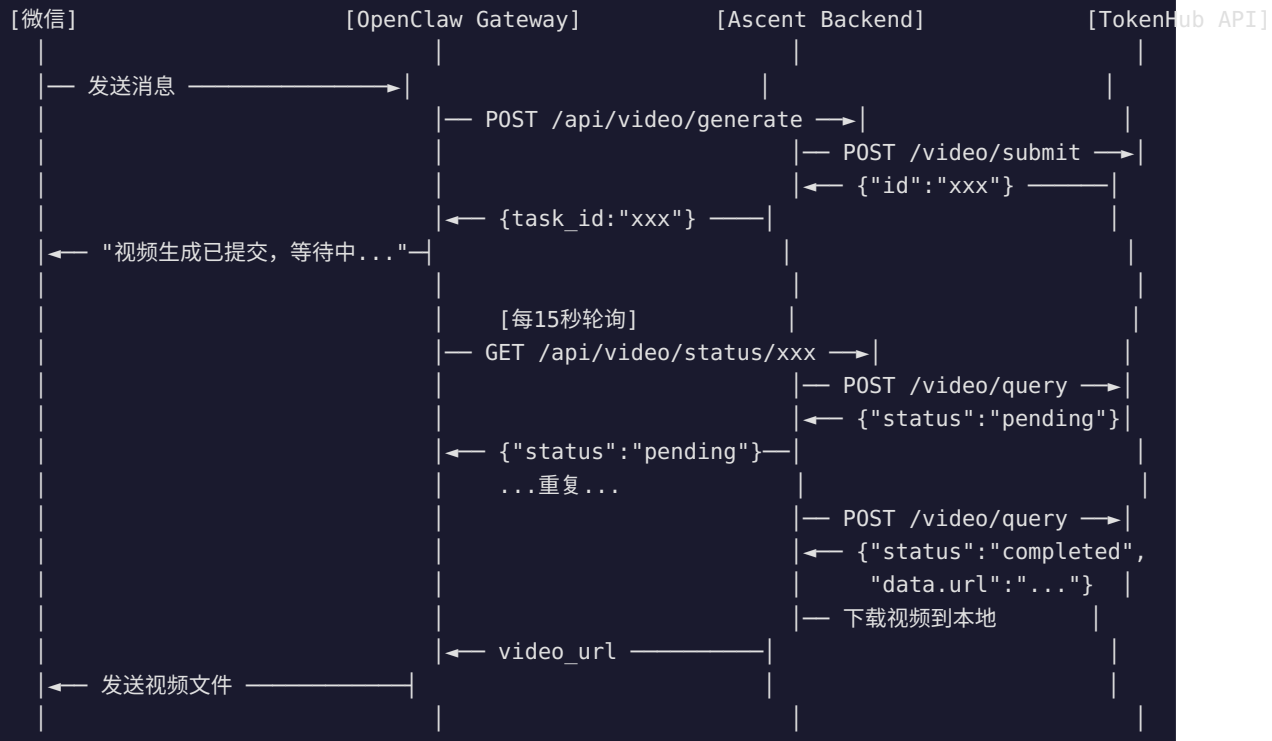
所有 AI 模型调用通过统一的**「模型网关层」**进行抽象和管理，这是平台实现模型集成设计的核心技术组件。平台集成各类开源、闭源模型，在模型管理里面管理各类模型。用户可以使用我们自带的开源模型（如 Qwen3-30B），也可以使用自己从云服务商购买的闭源模型（如GPT-4、通义千问等），只需在模型管理中配置API Base和Key即可使用：

```
┌─────────────────────────────────────────────┐ │ 模型网关抽象层 │ │
call_llm_stream(messages, model_id) │
├─────────────────────────────────────────────┤ │ │ │ 1. 从 config.MODELS 获取模型
配置 │ │ → api_base, api_key, temperature │ │ → max_tokens, top_p,
frequency_penalty│ │ │ 2. 构造统一请求 payload │ │ → {"model": ...,
"messages": ..., │ │ "stream": true, "temperature": ...} │ │ │ 3. 添加认证头
│ │ → headers["Authorization"] = Bearer │ │ │ 4. 发起流式请求 + 逐行 yield │ │
→ requests.post(stream=True) │ │ → for line in resp.iter_lines(): │ │ yield
"data: " + line │ │ │ 5. 统一响应格式 │ │ → data: {"content": "逐字内容"} │ │ →
data: [DONE] │ └─────────────────────────────────────────────┘ // config.py 模型配
置示例（可动态扩展） MODELS = { "qwen3_coder_30b_a3b_instruct...":
{ "display_name": "Qwen3-Coder-30B", "type": "code", "context_length": 262144,
"capabilities": ["chat", "code", "writing"], "completion_options":
{ "temperature": 0.6, "max_tokens": 4096, "top_p": 0.95, }, }, # 用户可以在此添
加自定义模型： # "my_gpt4": { # "display_name": "我的GPT-4o", # "type": "llm", #
"api_base": "https://api.openai.com/v1", # "api_key": "sk-xxx", #
"context_length": 128000, # ... # } }
```

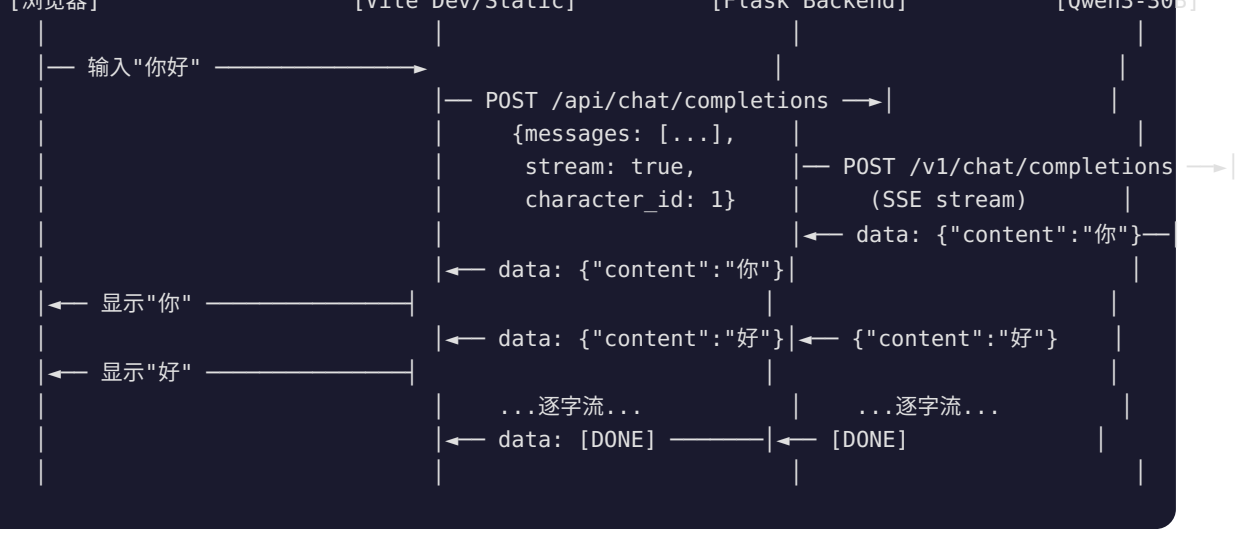
七、输入输出数据流

7.1 完整数据流图

场景：用户在微信上发送「帮我生成一只小猪唱歌的动画视频」



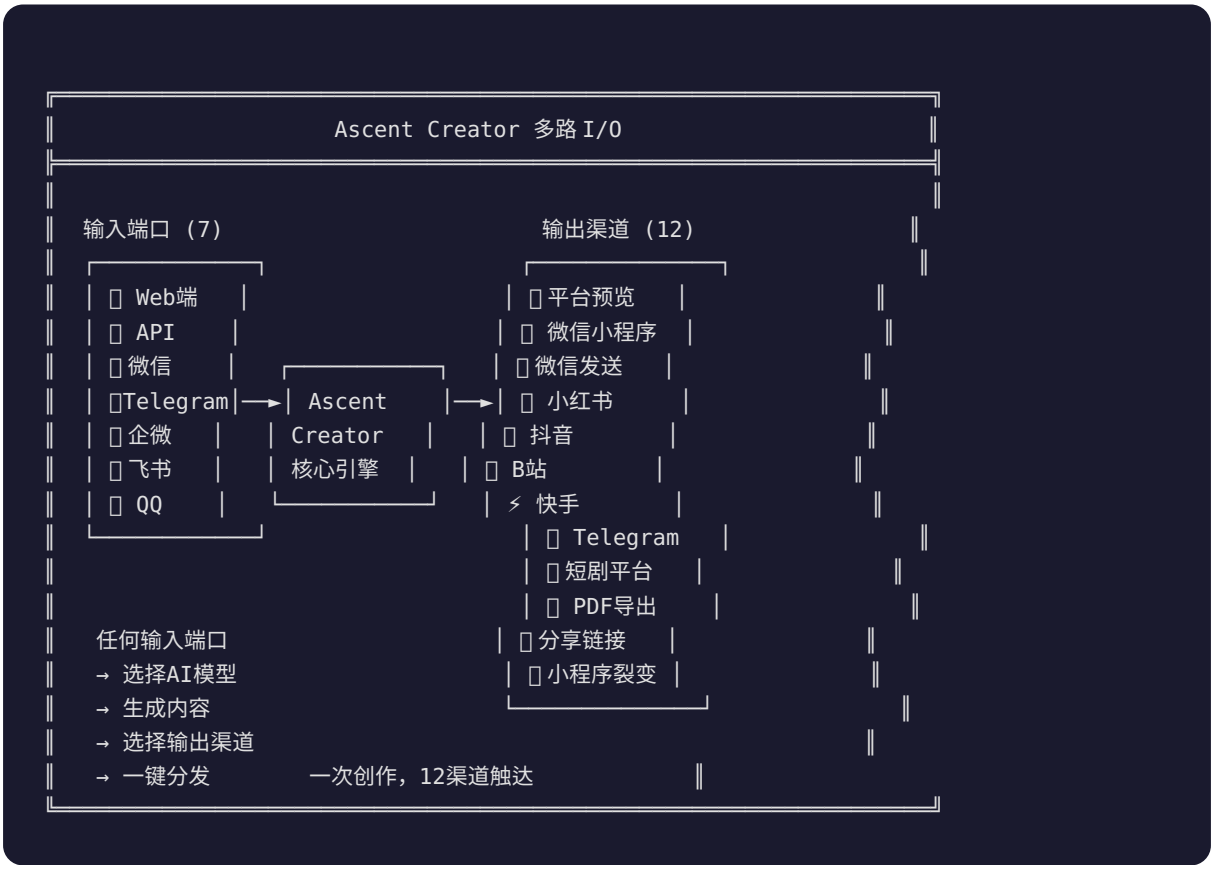
场景：用户在Web端与AI聊天助手对话



7.2 输入 → 输出映射表

输入方式	AI 模型	处理耗时	输出格式	交付渠道
□ 文字输入	Qwen3-30B	实时SSE	流式文本	Web · 微信 · Telegram
□ 图片描述	SDXL/Flux	5-15秒	PNG/JPG	Web · 微信 · 小红书
□ 视频描述	混元hy-v1.5	30-120秒	MP4	Web · 微信 · 抖音 · B站
□ 音乐描述	Suno v4	20-60秒	MP3/WAV	Web · 微信
□ 文字转语音	TTS模型	3-10秒	MP3	Web · 微信
□ 小说设定	Qwen3-30B	实时SSE	流式文本(章节)	Web · PDF导出
□ 应用描述	Qwen3-30B	10-30秒	HTML/JS/CSS	Web · 部署链接
□ 数字人	VRM/Live2D	即时	3D模型/动画	Web嵌入

7.3 多路输入输出拓扑



八、消息通道架构

8.1 OpenClaw Gateway 消息路由

平台的消息通道层基于 OpenClaw Gateway 实现，它是一个独立于后端服务的消息路由中间件：

通道	实现方式	消息格式	文件支持	状态
微信	企业微信个人号插件	文本/图片/视频/文件	□ 视频/图片/语音	□ 运行中
Telegram	Telegram Bot API	文本/媒体/内联键盘	□ 全部	□ 开发中
企业微信	MCP Server 协议	JSON-RPC	□ 文档	□ 规划中
飞书	飞书 Bot API	消息卡片/文本	□ 全部	□ 规划中
QQ	QQ Bot (Go-Mirai)	文本/图片	□ 图片	□ 规划中

8.2 后端启动流程

```
// 启动顺序 1. Qwen 模型服务（vllm serve qwen3...）→ :3990 2. Ascent 平台后端（python3 backend/app.py）→ :5000 3. 论坛服务（python3 forum.py）→ :5001 4. OpenClaw Gateway（消息通道管理） // 启动命令 cd /root/models_20260529220733/ai-creator-platform nohup python3 backend/app.py > /tmp/app.log 2>&1 & cd /root/.openclaw/workspace nohup python3 -c " from forum import app app.run(host='0.0.0.0', port=5001, debug=False) " > /tmp/forum_5001.log 2>&1 & // 构建前端 cd /root/models_20260529220733/ai-creator-platform npm run build # → dist/ 前端静态文件 // 健康检查 curl http://localhost:5000/api/health → 200 OK curl http://localhost:5001/ → 200 HTML
```

九、前端架构详解

9.1 组件层次结构

```
App.vue ← 根组件 └── AppLayout.vue ← 全局布局（侧边栏+顶栏+主内容区） ├── router-view（仪表盘）← Dashboard.vue ├── router-view（写作）← WritingStudio.vue ├── router-view（图片）← ImageStudio.vue ├── router-view（视频）← VideoStudio.vue | ├── ShareDialog.vue ← 分享弹窗（复用组件） ├── router-view（音乐）←
```

```
MusicStudio.vue ├── router-view (语音) — VoiceStudio.vue ├── router-view (聊天)
— ChatStudio.vue │ └── 多媒体内联渲染 (IMAGE/VIDEO/HTML) ├── router-view (小说) —
NovelStudio.vue │ ├── KnowledgeBasePanel.vue ← 知识库面板 (复用组件) ├── router-
view (应用) — CodeStudio.vue ├── router-view (数字人) — AvatarStudio.vue ├──
router-view (模型管理) — ModelHub.vue └── router-view (作品广场) — Gallery.vue
```

十、附录

10.1 服务地址

Ascent Creator 平台	http://82.156.214.59:5000
学习论坛	http://82.156.214.59:5001
Qwen 模型 API	http://123.58.111.244:3990/v1
数据库路径	/root/.openclaw/workspace/forum.db

10.2 技术栈速查表

```
┌──────────────┬──────────────────────────────────────────────┐ │ 前端框架 │ Vue 3.5 +
Vite 5.4 + Pinia 3 │ │ CSS 方案 │ Tailwind 3.4 + 自定义暗色主题 │ │ 前端路由 │ Vue
Router 4.6 (13条) │ │ HTTP 请求 │ Axios 1.16 (统一拦截器) │ │ 图标 │ Lucide Vue
Next 1.0 (40+图标) │ ├──────────────┼──────────────────────────────────────────────┤ │
后端框架 │ Flask 3.1 (2828行, 50+API) │ │ 数据库 │ SQLite 3 (19张表) │ │ AI 协议 │
OpenAI API 兼容 + SSE 流式 │ │ 外部调用 │ Requests 2.33 │ │ PDF 生成 │ WeasyPrint
68.1 │ ├──────────────┼──────────────────────────────────────────────┤ │ 构建产物 │
~500KB gzip (首页124KB, 其余懒加载) │ │ 部署方式 │ nohup + Flask Dev Server │ │ 操
作系统 │ OpenCloudOS 9 (Linux 6.6) │ │ 运行环境 │ Node 22 + Python 3.11 │ │ 消息
通道 │ OpenClaw Gateway (Node.js) │
└──────────────┴──────────────────────────────────────────────┘
```

⚙ Ascent Creator 技术架构

平台：:5000 · 论坛：:5001