# 前端实现与后端接口对比分析报告

版本日期: 2025-12-02

前端状态: React + TypeScript (Mock Backend 模式)

后端参考: 多角色对话系统通信协议说明\_zh.txt

## 一、 总体概况

当前前端应用已完成核心功能模块的 UI 开发与逻辑实现。为了演示和独立运行，前端内部实现了一个 Mock API 层（模拟后端）。在正式联调时，需要将 api 对象中的模拟函数替换为真实的 axios 或 fetch 请求。

**主要差异点摘要：**

1. **角色数据结构简化**：前端大幅简化了角色字段，仅保留名称和提示词。
2. **模板数据结构变更**：前端移除了模板的类型和描述，新增了“预设议题”字段。
3. **流转逻辑字段**：前端使用了 order (序号) 来处理跳转逻辑，后端协议中使用的是 step\_id。

## 二、 模块详细对比

### 1. 角色管理 (Role Management)

前端已根据用户需求进行了简化，与原设计文档差异较大。

| **字段/功能** | **原接口文档定义** | **当前前端实际实现** | **差异/联调建议** |
| --- | --- | --- | --- |
| **数据字段** | name, type, description, style, constraints, focus\_points | **name, prompt** | **重大差异**。  后端需调整数据库设计，或前端在发送请求时将 prompt 拆分/映射到后端字段，其余字段传空值。 |
| **列表查询** | GET /api/roles?type=... | api.getRoles (仅支持关键字搜索，移除了类型筛选) | 后端需支持仅按 name 搜索，忽略 type 参数。 |
| **唯一性校验** | 未明确定义 | 前端实现了**名称唯一性校验** | 后端 API POST/PUT 也应增加同名检测并返回 409 Conflict。 |

### 2. 流程模板 (Flow Templates)

前端对模板结构进行了调整以适应“议题”和“循环”需求。

| **字段/功能** | **原接口文档定义** | **当前前端实际实现** | **差异/联调建议** |
| --- | --- | --- | --- |
| **模板基础信息** | name, type, description, termination\_config | name, **topic (新增)**, is\_active | **1. 新增字段**：后端需在 Template 表中增加 topic (Text) 字段。  **2. 移除字段**：后端需允许 type 和 description 为空或前端传默认值。 |
| **步骤对象** | speaker\_role\_ref, target\_role\_ref | 同左，但 target\_role\_ref 增加了特殊值 **\_\_TOPIC\_\_** | 后端需处理 target\_role\_ref 为 \_\_TOPIC\_\_ 的情况，代表“针对议题发言”。 |
| **流转逻辑** | condition\_config, next\_step\_id | logic\_config: { next\_step\_order, exit\_condition, max\_loops } | **逻辑映射需转换**：  前端使用 order (序号) 指向跳转目标，后端通常使用 step\_id。前端提交保存时，后端需根据 order 解析并建立 step\_id 关联。 |

### 3. 会话管理 (Session Management)

| **字段/功能** | **原接口文档定义** | **当前前端实际实现** | **差异/联调建议** |
| --- | --- | --- | --- |
| **创建会话** | POST Body: { topic, flow\_template\_id, role\_mappings } | 同左，但前端会自动从模板带入 topic | 接口兼容，无重大差异。 |
| **执行下一步** | POST .../run-next-step | 模拟逻辑：自动生成 Mock 回复，前端计算 next\_step\_index | **核心联调点**：  正式环境应由后端计算 Next Step 和生成 Content。  前端会将 UI 上的 topic 数据传递给后端，后端 LLM 在 Step 1 生成时需注入该 Topic 上下文。 |
| **历史记录** | GET /api/sessions | GET /api/sessions?status=finished | 后端需支持 status 过滤参数。 |

## 三、 接口映射表 (Interface Mapping)

开发真实后端对接层时，请参照下表替换 MultiRoleDialogSystem.tsx 中的 api 对象实现。

| **前端 Mock 函数** | **HTTP 方法** | **建议后端路径** | **Payload/Query 备注** |
| --- | --- | --- | --- |
| getRoles | GET | /api/roles | ?keyword={val} |
| saveRole (Create) | POST | /api/roles | { name, prompt } |
| saveRole (Update) | PUT | /api/roles/{id} | { name, prompt } |
| deleteRole | DELETE | /api/roles/{id} | - |
| getFlows | GET | /api/flows | - |
| getFlowDetail | GET | /api/flows/{id} | 返回包含 steps 的完整对象 |
| saveFlow | POST/PUT | /api/flows / /{id} | Body 需包含 topic 字段 |
| getSessions | GET | /api/sessions | ?status=finished (历史记录页用) |
| createSession | POST | /api/sessions | Role Mappings 逻辑保持不变 |
| getSessionDetail | GET | /api/sessions/{id} | - |
| runNextStep | POST | /api/sessions/{id}/run | 后端需返回生成的 Message 和新的 Session State (包含 current\_step\_index) |
| finishSession | POST | /api/sessions/{id}/finish | - |

## 四、 待办事项 (Action Items)

1. **后端数据库变更**：
   * Role 表：废弃 type, style 等字段，新增或复用字段存储 prompt。
   * FlowTemplate 表：新增 topic 字段 (Text/LongText)。
2. **后端逻辑更新**：
   * 在执行会话的 Step 1 时，检查 Template 是否有 topic。如果有，需在 Prompt 中注入“针对议题 {topic} 发言”的指令。
   * 识别 target\_role\_ref = '\_\_TOPIC\_\_' 的特殊标记，调整 Prompt 指令为“点评议题”而非“对某人说话”。
3. **前端对接工作**：
   * 编写 api/client.ts，引入 axios。
   * 将 MultiRoleDialogSystem.tsx 中的 const api = { ... } 替换为真实的 API 调用。