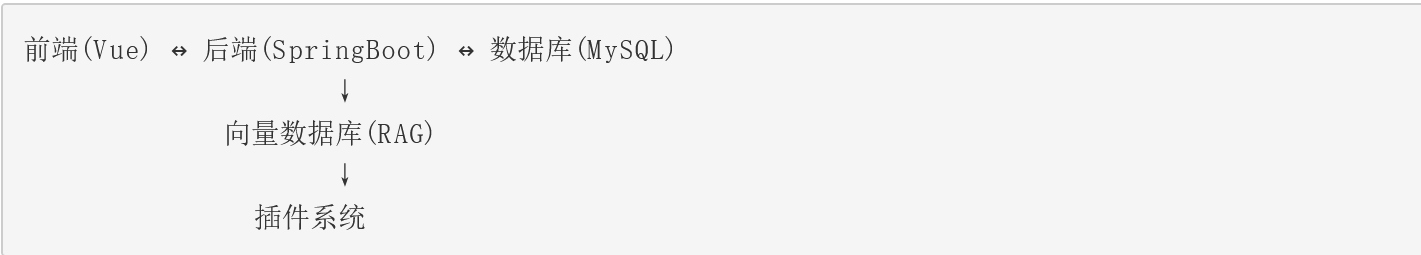


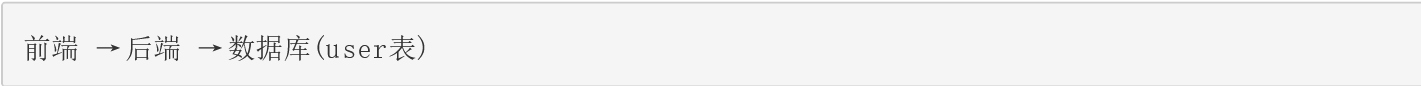
我来从前后端数据传输的角度，详细分析智能体 workflow 平台的完整创作过程。

1. 系统架构概览



2. 用户注册登录阶段

数据传输流程：



涉及数据表：

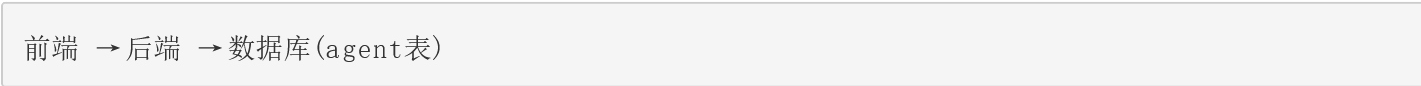
```
-- 用户表
CREATE TABLE users (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(50) UNIQUE,
  email VARCHAR(100),
  password_hash VARCHAR(255),
  created_at TIMESTAMP,
  updated_at TIMESTAMP
);
```

数据传输：

- 注册： {username, email, password} → 后端 → 密码加密 → 数据库
- 登录： {username, password} → 后端 → 验证 → 返回 token

3. 智能体创建阶段

数据传输流程：



涉及数据表：

```

-- 智能体基础表
CREATE TABLE agents (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT,
    name VARCHAR(100),
    description TEXT,
    avatar_url VARCHAR(255),
    status ENUM('draft', 'testing', 'published'),
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- 智能体配置表
CREATE TABLE agent_configs (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    agent_id BIGINT,
    system_prompt TEXT,
    temperature DECIMAL(3, 2),
    max_tokens INT,
    rag_enabled BOOLEAN,
    plugins_enabled BOOLEAN,
    FOREIGN KEY (agent_id) REFERENCES agents(id)
);

```

数据传输：

```

// 前端 → 后端
{
    "name": "客服助手",
    "description": "处理客户咨询的智能助手",
    "systemPrompt": "你是一个专业的客服助手...",
    "temperature": 0.7,
    "ragEnabled": true,
    "pluginsEnabled": true
}

```

4. 知识库管理阶段

数据传输流程：

```

前端 → 后端 → 文件存储 → 文本处理 → 向量数据库
                        ↓
                    数据库(knowledge_base表)

```

涉及数据表：

-- 知识库表

```
CREATE TABLE knowledge_bases (  
    id BIGINT PRIMARY KEY AUTO_INCREMENT,  
    agent_id BIGINT,  
    name VARCHAR(100),  
    description TEXT,  
    created_at TIMESTAMP,  
    FOREIGN KEY (agent_id) REFERENCES agents(id)  
);
```

-- 文档表

```
CREATE TABLE documents (  
    id BIGINT PRIMARY KEY AUTO_INCREMENT,  
    kb_id BIGINT,  
    filename VARCHAR(255),  
    file_path VARCHAR(500),  
    file_size BIGINT,  
    file_type VARCHAR(50),  
    chunk_count INT,  
    status ENUM('uploading', 'processing', 'completed', 'failed'),  
    created_at TIMESTAMP,  
    FOREIGN KEY (kb_id) REFERENCES knowledge_bases(id)  
);
```

-- 文档分块表（元数据）

```
CREATE TABLE document_chunks (  
    id BIGINT PRIMARY KEY AUTO_INCREMENT,  
    document_id BIGINT,  
    chunk_index INT,  
    content TEXT,  
    token_count INT,  
    vector_id VARCHAR(100), -- 向量数据库中的ID  
    FOREIGN KEY (document_id) REFERENCES documents(id)  
);
```

数据传输流程（这个没写全）：

1. 上传文档：

```
// 前端 → 后端 (FormData)  
{  
    "file": File对象,  
    "kbId": 123,  
    "agentId": 456  
}
```

2. 文档处理状态:

```
// 后端 → 前端 (WebSocket/轮询)
{
  "documentId": 789,
  "status": "processing",
  "progress": 75,
  "chunksProcessed": 150
}
```

5. workflows 配置阶段

数据传输流程:

前端 → 后端 → 数据库(workflow表)

涉及数据表:

-- 工作流表

```
CREATE TABLE workflows (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  agent_id BIGINT,
  name VARCHAR(100),
  description TEXT,
  config JSON, -- 存储工作流节点配置
  version INT,
  is_active BOOLEAN,
  created_at TIMESTAMP,
  FOREIGN KEY (agent_id) REFERENCES agents(id)
);
```

-- 工作流节点表

```
CREATE TABLE workflow_nodes (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  workflow_id BIGINT,
  node_id VARCHAR(50), -- 前端生成的节点ID
  node_type ENUM('llm', 'rag', 'plugin', 'condition', 'input', 'output'),
  position_x INT,
  position_y INT,
  config JSON, -- 节点具体配置
  FOREIGN KEY (workflow_id) REFERENCES workflows(id)
);
```

-- 工作流边表

```
CREATE TABLE workflow_edges (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
```

```
workflow_id BIGINT,  
source_node_id VARCHAR(50),  
target_node_id VARCHAR(50),  
source_handle VARCHAR(50),  
target_handle VARCHAR(50),  
FOREIGN KEY (workflow_id) REFERENCES workflows(id)  
);
```

数据传输：

```
// 前端 → 后端（保存工作流）  
{  
  "agentId": 456,  
  "name": "客服工作流",  
  "nodes": [  
    {  
      "id": "node-1",  
      "type": "input",  
      "position": { "x": 100, "y": 100 },  
      "data": { "label": "用户输入" }  
    },  
    {  
      "id": "node-2",  
      "type": "rag",  
      "position": { "x": 300, "y": 100 },  
      "data": {  
        "knowledgeBaseId": 123,  
        "topK": 5,  
        "scoreThreshold": 0.7  
      }  
    }  
  ],  
  "edges": [  
    {  
      "id": "edge-1",  
      "source": "node-1",  
      "target": "node-2",  
      "sourceHandle": "output",  
      "targetHandle": "input"  
    }  
  ]  
}
```

6. 插件管理阶段

涉及数据表：

-- 插件表

```
CREATE TABLE plugins (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  name VARCHAR(100),  
  description TEXT,  
  author VARCHAR(100),  
  version VARCHAR(20),  
  endpoint_url VARCHAR(500),  
  parameters JSON,  
  is_active BOOLEAN  
);
```

-- 智能体插件关联表

```
CREATE TABLE agent_plugins (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  agent_id BIGINT,  
  plugin_id BIGINT,  
  config JSON, -- 插件具体配置  
  execution_order INT,  
  FOREIGN KEY (agent_id) REFERENCES agents(id),  
  FOREIGN KEY (plugin_id) REFERENCES plugins(id)  
);
```

7. 智能体测试阶段

数据传输流程：

用户输入 → 前端 → 后端 → workflow引擎 → RAG/插件 → LLM → 返回结果

测试会话数据：

-- 测试会话表

```
CREATE TABLE test_sessions (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  agent_id BIGINT,  
  user_id BIGINT,  
  session_id VARCHAR(100),  
  created_at TIMESTAMP,  
  FOREIGN KEY (agent_id) REFERENCES agents(id),  
  FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

-- 对话记录表

```
CREATE TABLE chat_messages (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  session_id BIGINT,
```

```
role ENUM('user', 'assistant', 'system'),
content TEXT,
metadata JSON, -- 包含RAG检索结果、插件调用记录等
created_at TIMESTAMP,
FOREIGN KEY (session_id) REFERENCES test_sessions(id)
);
```

实时测试数据传输：

```
// 用户输入 → 后端
{
  "sessionId": "session-123",
  "agentId": 456,
  "message": "请问产品价格是多少？",
  "workflowId": 789
}

// 后端处理过程数据
{
  "ragResults": [
    {
      "content": "产品A价格是100元...",
      "score": 0.85,
      "source": "price_doc.pdf"
    }
  ],
  "pluginExecutions": [
    {
      "pluginName": "价格查询",
      "input": {"product": "产品A"},
      "output": {"price": 100, "currency": "CNY"},
      "success": true
    }
  ]
}

// 最终响应 → 前端
{
  "sessionId": "session-123",
  "message": "产品A的价格是100元...",
  "sources": ["price_doc.pdf", "价格查询插件"],
  "timestamp": "2024-01-20T10:30:00Z"
}
```

8. 完整数据流总结

1. 用户管理流：注册 → 登录 → 身份验证

2. **智能体创建流**: 基础信息 → 配置 → 保存
3. **知识库流**: 上传文档 → 分块处理 → 向量化存储
4. **workflow设计流**: 节点配置 → 连接关系 → 版本管理
5. **插件集成流**: 插件选择 → 参数配置 → 执行顺序
6. **测试执行流**: 用户输入 → workflow执行 → 多步骤处理 → 结果返回

这个架构支持灵活的智能体创作，同时保证了数据的完整性和可追溯性。