github仓库：https://github.com/lwq-NEWCEO/CloudComputer2025.git

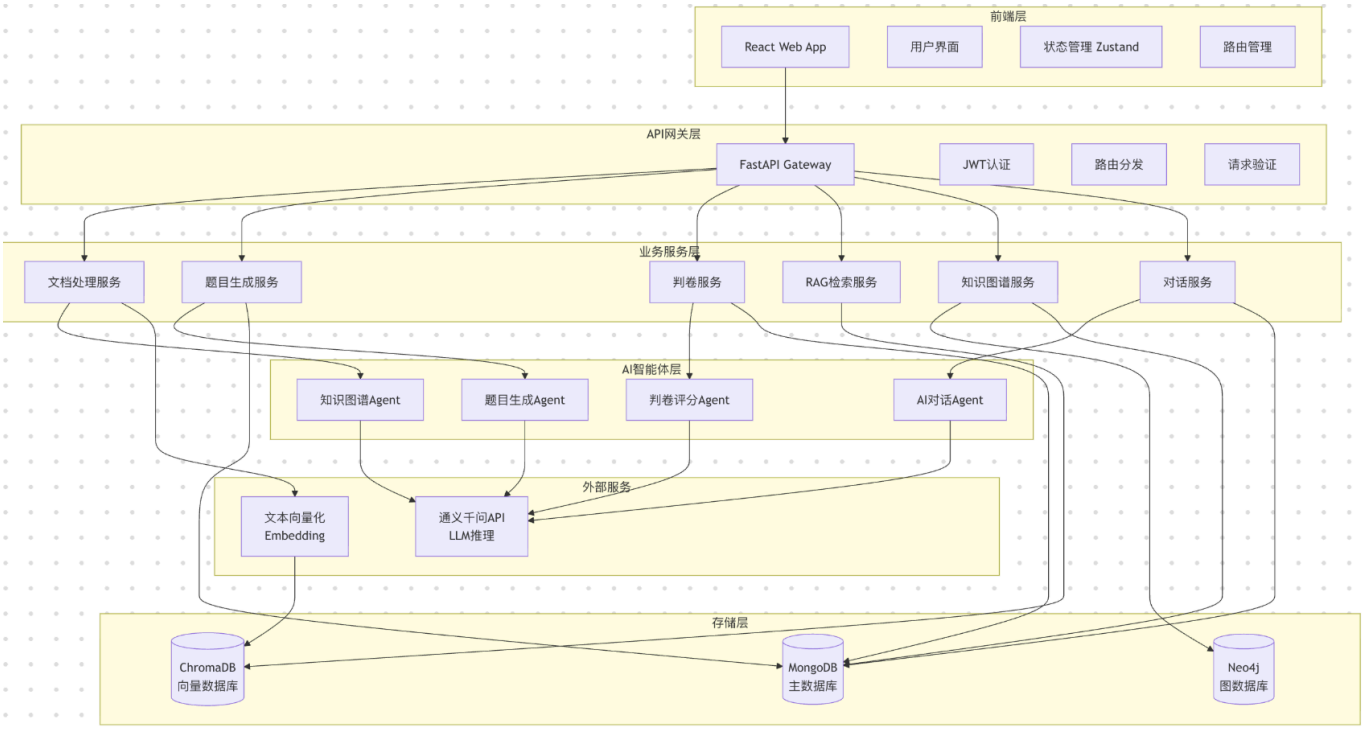
分工Part1

负责登录注册前后端
上传文件前后端
根据文件出选择问答题前后端
错题本前后端
知识图谱构建前后端

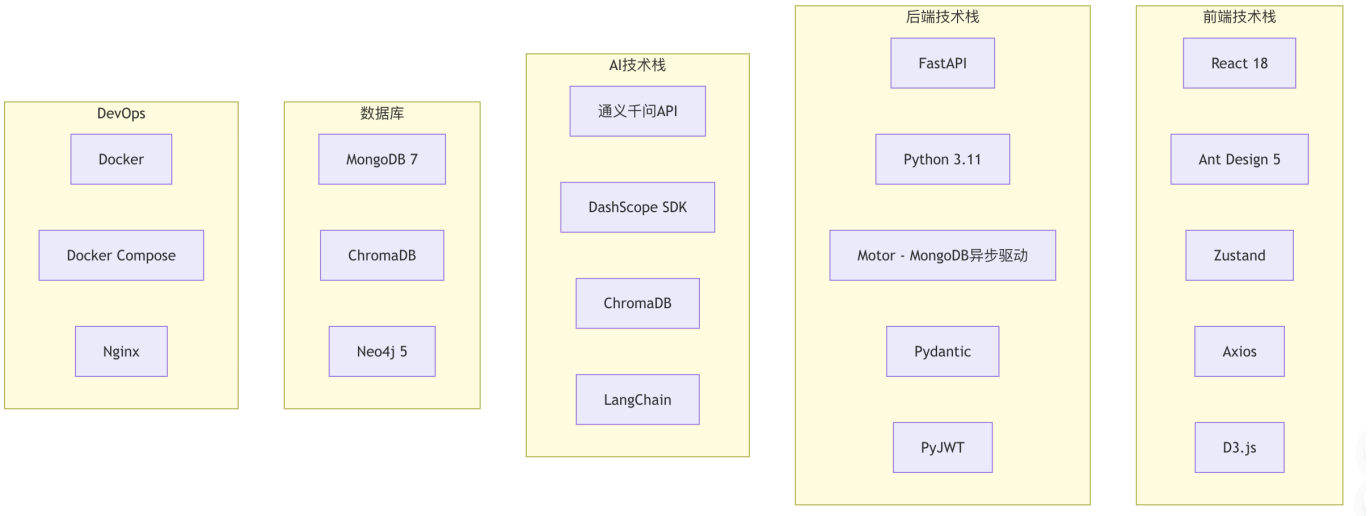
1. 系统架构设计

1.1 整体架构

本系统采用**微服务架构**，结合**RAG（检索增强生成）和多智能体协作**技术，构建了一个智能化的学习评估平台。整体流程如下：



1.2 三层架构详解



第一层：前端层

- **框架**: React 18 (函数式组件 + Hooks)
- **UI库**: Ant Design 5.x (企业级UI组件)
- **状态管理**: Zustand (轻量级状态管理)
- **路由**: React Router v6
- **HTTP客户端**: Axios (带拦截器的请求封装)

关键特性:

- 组件化开发，高度复用
- 响应式设计，支持多端适配
- 实时反馈（答题结果、AI分析）
- 可视化知识图谱（D3.js）

第二层：后端服务层

API服务 (FastAPI)

```
app/
├── routes/ # API路由
│ ├── auth.py # 认证授权
│ ├── upload.py # 文件上传
│ ├── questions.py # 题目生成
│ ├── grading.py # 自动判卷
│ ├── knowledge.py # 知识图谱
│ ├── chat.py # AI对话
│ └── errorbook.py # 错题本
├── services/ # 业务服务
│ ├── question_generator.py # 题目生成服务
│ ├── grading_service.py # 判卷服务
│ ├── rag_service.py # RAG检索服务
│ ├── knowledge_graph.py # 知识图谱服务
│ └── chat_service.py # 对话服务
└── models/ # 数据模型
```

第三层：存储层

多数据库架构:

1. **MongoDB** - 主数据库
   - 用户数据、题目库、答题记录
   - 错题本、知识点
2. **ChromaDB** - 向量数据库
   - 文档向量存储
   - 相似度检索
   - RAG知识检索
3. **Neo4j** - 图数据库
   - 知识图谱存储
   - 知识点关系网络
   - 学习路径推荐

2. 云原生组件

2.1 容器化架构 (Docker)

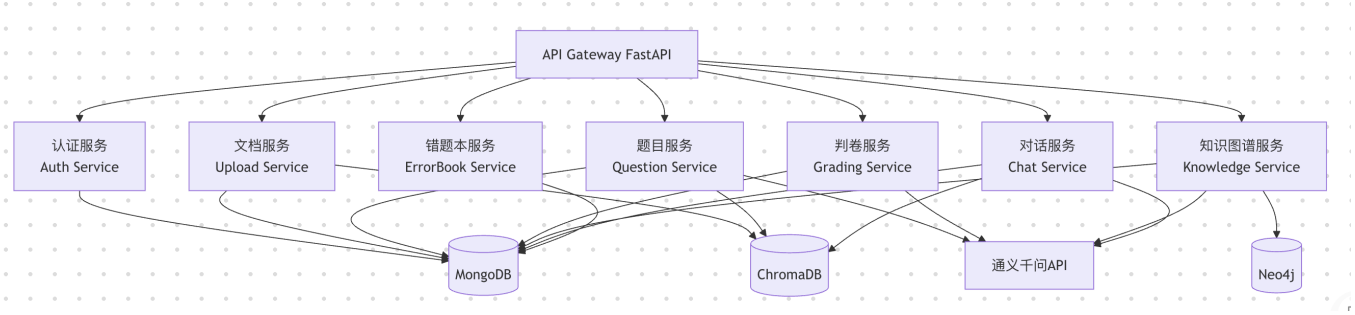
Docker Compose 编排:

```
services:

# 前端服务
frontend:
image: node:18-alpine
ports: ["3000:3000"]

# 后端服务
backend:
image: python:3.11-slim
ports: ["8000:8000"]
depends_on:
```

```
- mongodb
- chroma
mongodb:
image: mongo:7
volumes: [mongodb_data:/data/db]

# ChromaDB 向量数据库
chroma:
image: chromadb/chroma:latest
volumes: [chroma_data:/chroma/chroma]

# Neo4j 图数据库
neo4j:
image: neo4j:5
environment:
NEO4J_AUTH: neo4j/password
```

2.2 微服务设计

服务拆分原则:



每个服务职责单一：

- **文档处理服务**: 文件解析、分块、向量化
- **题目生成服务**: 基于知识点智能出题
- **判卷服务**: 多智能体协作评分
- **RAG检索服务**: 语义搜索、上下文检索
- **知识图谱服务**: 知识点关系、薄弱点分析
- **AI对话服务**: 智能答疑、学习指导

2.3 云服务集成

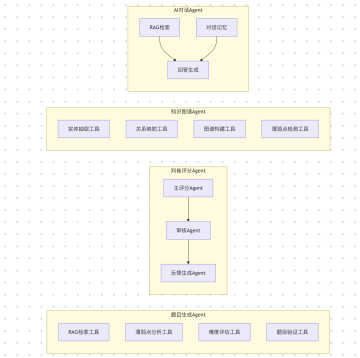
通义千问 API (Qwen):

```
# 配置

QWEN_API_KEY = "sk-xxx"
QWEN_MODEL = "qwen-plus"
QWEN_EMBEDDING_MODEL = "text-embedding-v3"

# 调用示例
from dashscope import Generation

response = Generation.call(
model='qwen-plus',
messages=[
{'role': 'system', 'content': '你是一个专业的出题专家'},
{'role': 'user', 'content': '根据知识点生成题目...'}
],
temperature=0.7
)
```

3. LLM Agent智能体设计

3.1 Agent架构概览

本系统采用**多智能体协作架构**，每个Agent负责特定任务：



3.2 Agent 1: 题目生成Agent (Question Generator)

核心职责: 基于知识库内容和用户薄弱点，智能生成高质量练习题

工具链设计:

```
class QuestionGeneratorAgent:
tools = [
RAGRetrievalTool, # RAG检索工具
WeakPointAnalyzer, # 薄弱点分析工具
DifficultyEvaluator, # 难度评估工具
QuestionValidator # 题目验证工具
]
```

工作流程:

```
1. 知识检索阶段
└> RAG检索相关知识点
└> 分析用户错题记录
└> 识别薄弱知识领域
2. 题目构思阶段
└> 确定考察知识点
└> 设计题目类型（选择/简答/代码）
└> 生成题干和选项
3. 质量保障阶段
└> 验证题目准确性
└> 评估难度等级
└> 生成标准答案和解析
4. 个性化调整
└> 根据用户水平调整
└> 聚焦薄弱点强化
```

3.3 Agent 2: 判卷评分Agent (Grading Agent)

核心职责: 多维度评估学生答案，提供详细反馈

多Agent协作机制:

```
class GradingSystem:
agents = [
PrimaryGrader, # 主评分员
VerifierGrader, # 审核评分员
FeedbackGenerator # 反馈生成器
]
```

评分流程 (Prometheus评估法):

```
┌─────────────────────────────────────┐
│ 主评分Agent (Primary Grader) │
│ │
│ 评分维度: │
│ - 准确性 (40%) │
│ - 完整性 (30%) │
│ - 清晰度 (20%) │
│ - 深度 (10%) │
└──────────┬──────────────────────────┘

│ 初始评分结果

▼

┌─────────────────────────────────────┐
│ 审核Agent (Verifier Grader) │
│ │
│ 审核内容: │
│ - 评分是否公正 │
│ - 是否有偏差 │
│ - 给出修正建议 │
└──────────┬──────────────────────────┘

│ 审核后的最终分数

▼

┌─────────────────────────────────────┐
│ 反馈生成Agent (Feedback Generator) │
│ │
│ 生成内容: │
│ - 优点分析 │
│ - 错误指出 │
│ - 改进建议 │
│ - 薄弱知识点 │
└─────────────────────────────────────┘
```

3.4 Agent 3: 知识图谱Agent (Knowledge Graph Agent)

核心职责: 自动构建知识关系网络，识别学习盲点

工具链:

```
class KnowledgeGraphAgent:
tools = [
EntityExtractor, # 实体抽取
RelationshipMapper, # 关系映射
GraphBuilder, # 图谱构建
WeakPointDetector # 薄弱点检测
]
```

知识提取流程:

```
async def extract_knowledge_points(content: str) -> List[KnowledgePoint]:

# Prompt设计

prompt = f"""从以下内容提取知识点并建立关系:
{content}
返回JSON格式:
{{
"knowledge_points": [
{{
"name": "知识点名称",
"description": "简要描述",
"category": "分类",
"keywords": ["关键词"],
"related_to": ["相关知识点"]
}}
]
}}"""
response = await llm.generate(prompt)
kps = parse_json(response)
```

```
# 构建Neo4j图谱
for kp in kps:
await create_node(kp)
for related in kp['related_to']:
await create_relationship(kp, related)

return kps
```

4. Prompt工程与模板

4.1 题目生成Prompt模板

单选题生成模板:

```
SINGLE_CHOICE_PROMPT = """你是一个专业的出题专家。请根据以下知识内容生成高质量的单选题。

【知识库内容】
{context}

【薄弱知识点】

{weak_points}

【出题要求】

- 题目数量: {count}

- 难度等级: {difficulty}

- 题型: 单选题

- 每题包含: 题干、4个选项(A/B/C/D)、正确答案、详细解析

- 选项设计: 包含常见错误认知和混淆项

【输出格式】(严格JSON)

{{

"questions": [

{{

"content": "题目内容",

"options": [

{{"key": "A", "content": "选项A", "is_correct": false}},

{{"key": "B", "content": "选项B", "is_correct": true}},

{{"key": "C", "content": "选项C", "is_correct": false}},

{{"key": "D", "content": "选项D", "is_correct": false}}

],

"correct_answer": "B",

"explanation": "详细解析",

"knowledge_points": ["涉及的知识点"],

"difficulty": "medium"

}}

]

}}


请生成题目:"""
```

简答题生成模板:

```
SHORT_ANSWER_PROMPT = """你是一个教育专家。请根据知识内容设计开放性简答题。

【知识内容】

{context}

【设计要求】

- 题目数量: {count}

- 难度: {difficulty}

- 考察理解深度，而非死记硬背

- 提供评分标准（分点给分）

【输出格式】

{{

"questions": [

{{

"content": "请解释XXX概念，并举例说明其应用场景",

"correct_answer": "标准答案（包含关键得分点）",

"explanation": "详细解析",

"knowledge_points": ["核心概念", "应用场景"],

"scoring_rubric": [

{{"point": "正确定义概念", "score": 40}},

{{"point": "举例说明", "score": 30}},

{{"point": "分析应用场景", "score": 30}}

]

}}

]

}}"""
```

4.2 判卷评分Prompt模板

主评分Agent Prompt:

```
PRIMARY_GRADER_PROMPT = """你是一个公正的评卷专家。使用Prometheus评估法评分。

【评分标准】

1. 准确性(40%): 答案是否正确反映了核心概念

2. 完整性(30%): 答案是否覆盖了所有要点

3. 清晰度(20%): 表达是否清晰有逻辑

4. 深度(10%): 是否有深入理解的体现

【评分规则】

- 90-100: 完全正确，表述清晰，有深度

- 70-89: 基本正确，有小错误或遗漏

- 50-69: 部分正确，有明显错误或遗漏

- 30-49: 大部分错误，但有部分正确内容

- 0-29: 完全错误或未作答

【题目】
```

```
{question}

【标准答案】

{correct_answer}

【参考资料】

{context}

【学生答案】

{user_answer}

【输出格式】(JSON)

{{

"score": 85,

"is_correct": true,

"accuracy_score": 38,

"completeness_score": 25,

"clarity_score": 16,

"depth_score": 6,

"feedback": "总体反馈",

"correct_parts": ["正确的部分"],

"incorrect_parts": ["错误的部分"],

"missing_parts": ["遗漏的部分"],

"improvement_suggestions": ["改进建议1", "建议2"],

"weak_points": ["薄弱知识点"]

}}

请评估:"""
```

审核Agent Prompt:

```
VERIFIER_PROMPT = """你是一个评分审核专家。审核评分是否公正。


【原始评分】
分数: {initial_score}
反馈: {initial_feedback}


【题目和答案】
题目: {question}
标准答案: {correct_answer}
学生答案: {user_answer}


【审核任务】
1. 检查评分是否过严或过宽
2. 是否遗漏了学生答案的亮点
3. 是否对错误过于苛刻
4. 给出你认为合理的分数



【输出】(JSON)

{{
"score": 80,
"agreement": "agree/disagree",
"reason": "审核理由",
"adjustments": "建议调整的部分"
}}"""
```

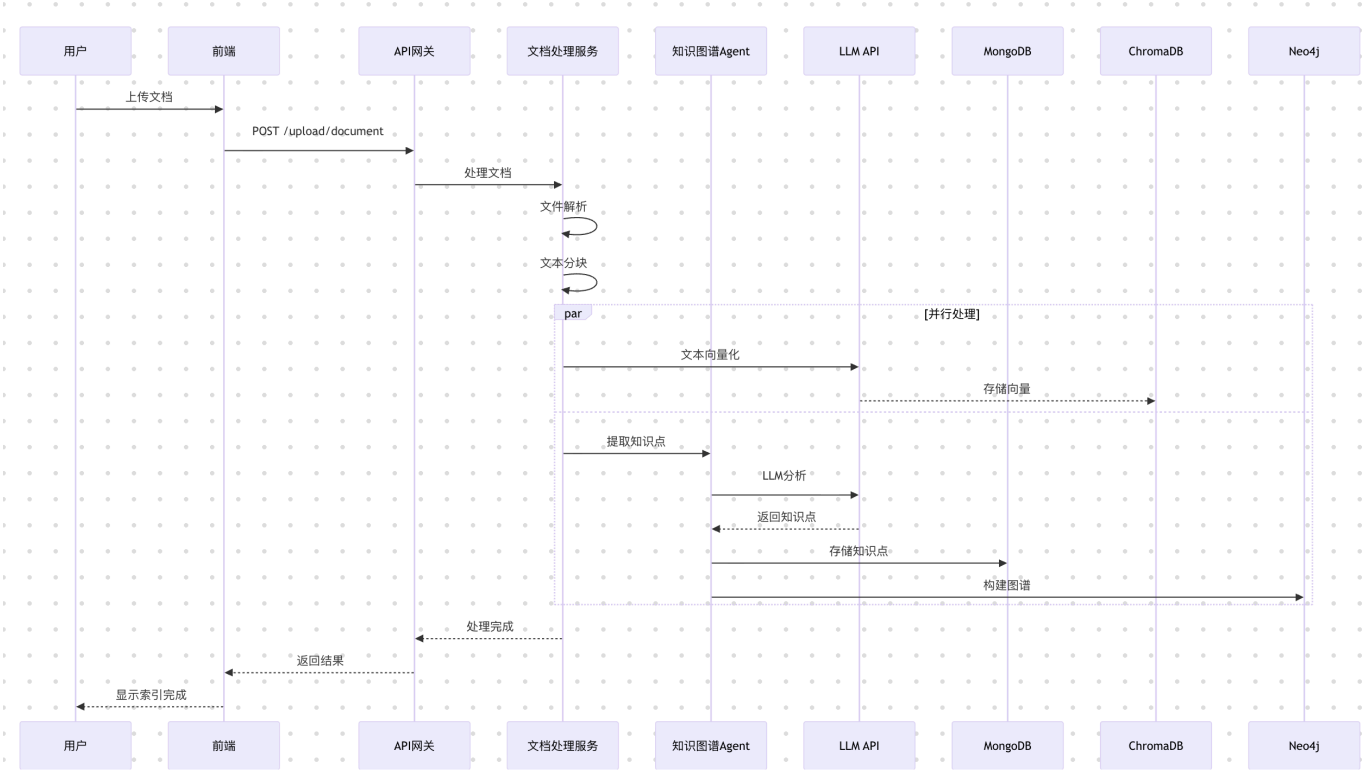
4.3 知识图谱提取Prompt

```
KNOWLEDGE_EXTRACTION_PROMPT = """你是一个知识图谱专家。从文本中提取知识点并建立关系。

【文本内容】
{content}

【提取要求】
1. 识别所有重要概念（名词、术语）
2. 提取概念定义和描述
3. 识别概念之间的关系（依赖、包含、关联）
4. 提取关键词和分类

【输出格式】(严格JSON)
{{
"knowledge_points": [
{{
"name": "数据库索引",
"description": "提高查询效率的数据结构",
"category": "数据库",
"keywords": ["B树", "哈希", "性能优化"],
"related_to": ["查询优化", "存储引擎"]
}},
{{
"name": "查询优化",
"description": "提升SQL执行效率的技术",
"category": "数据库",
"keywords": ["执行计划", "索引选择"],
"related_to": ["数据库索引", "SQL语句"]
}}
],
"relationships": [
{{
"source": "数据库索引",
"target": "查询优化",
"type": "ENABLES"
}}]
}}

请提取:"""
```

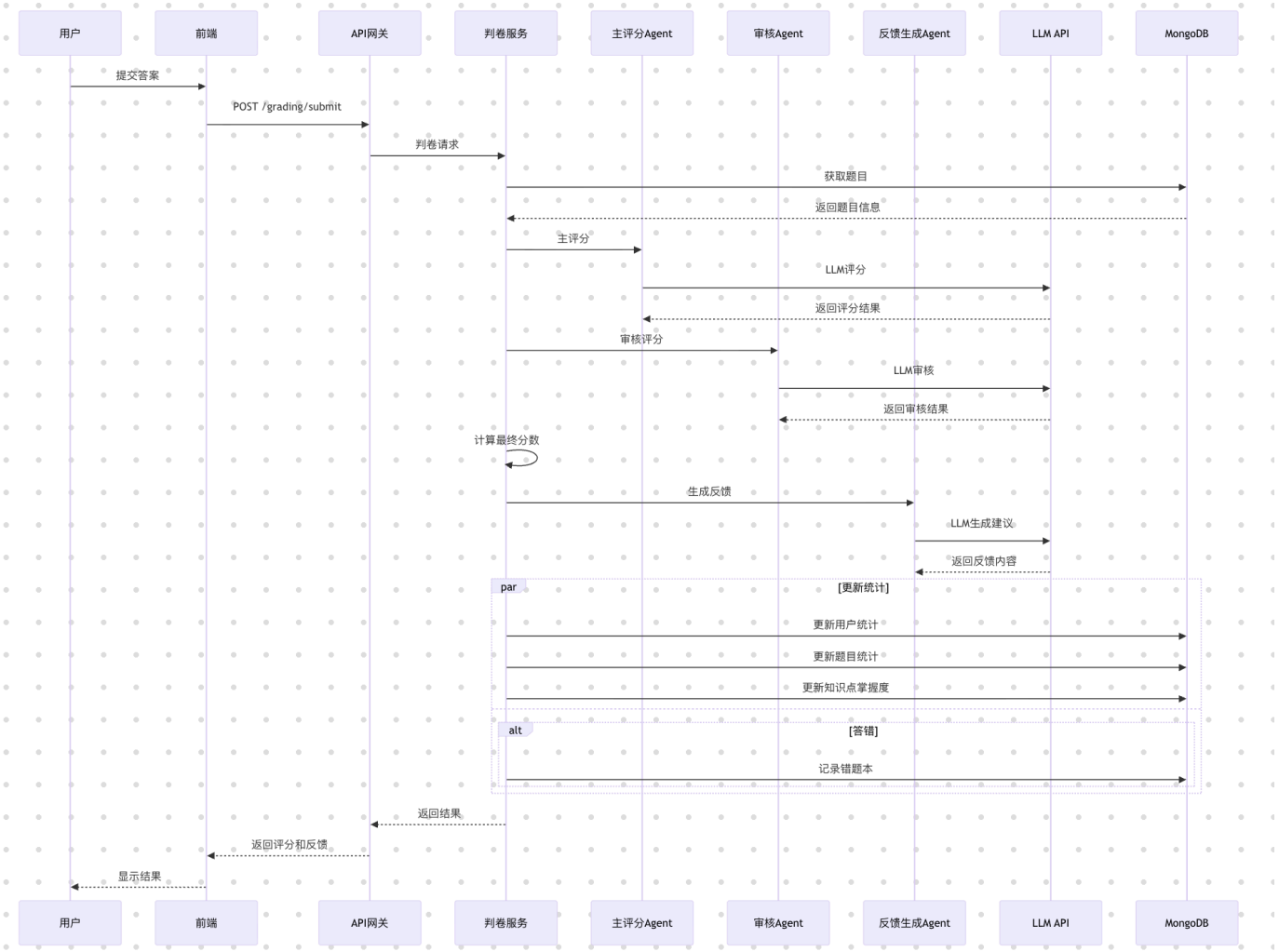
5. 数据流向与调用逻辑

5.1 核心业务流程

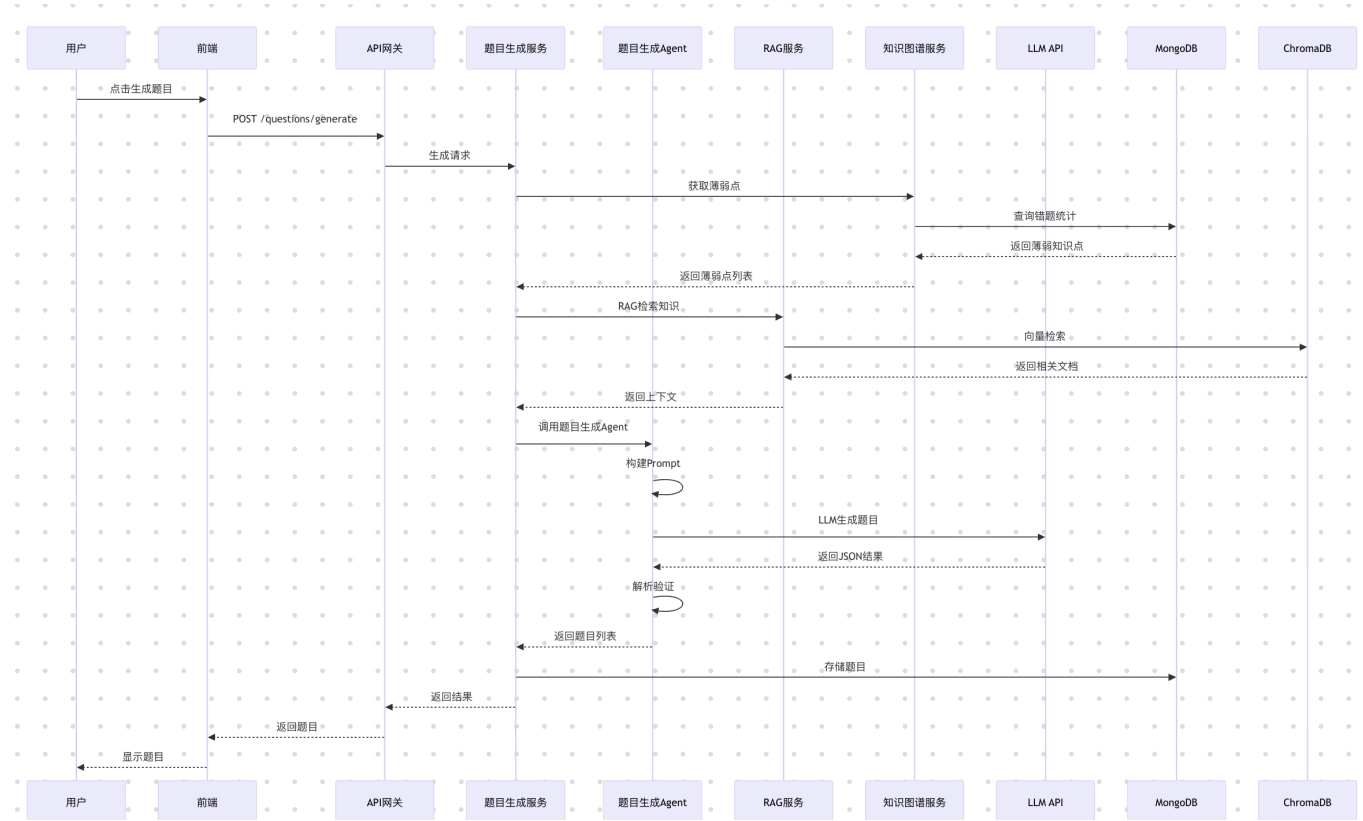
流程1: 文档上传与知识提取



流程2: 智能题目生成与智能判卷



流程3: 知识图谱



5.2 关键数据结构

题目数据模型

```
{
"_id": ObjectId("..."),
"knowledge_base_id": "kb_xxx",
"user_id": "user_xxx",
"question_type": "single_choice", # 单选/多选/简答/代码
"difficulty": "medium", # easy/medium/hard
"content": "题目内容",
"options": [ # 选择题选项
{"key": "A", "content": "...", "is_correct": false},
{"key": "B", "content": "...", "is_correct": true}
],
"correct_answer": "B",
"explanation": "详细解析",
"knowledge_points": ["索引", "查询优化"],
"times_answered": 0,
"times_correct": 0,
"created_at": ISODate("...")
}
```

判卷结果模型

```
{
"question_id": "xxx",
"is_correct": true,
"score": 85.5,
"user_answer": "学生的答案",
"correct_answer": "标准答案",
"feedback": "总体评价",
"explanation": "题目解析",
"improvement_suggestions": [
"建议1: 加强对XXX概念的理解",
"建议2: 注意XXX细节"
],
"weak_points_identified": ["概念理解", "应用能力"],
"test_results": [...], # 代码题的测试结果"
code_analysis": "..." # 代码质量分析
}
```

知识图谱节点

```
# MongoDB存储

{
"_id": ObjectId("..."),
"knowledge_base_id": "kb_xxx",
"name": "数据库索引",
"description": "提高查询效率的数据结构",
"category": "数据库",
"keywords": ["B树", "哈希", "性能"],
"error_count": 3, # 错误次数
"correct_count": 7, # 正确次数
"mastery_level": 0.7, # 掌握度 = correct/(correct+error)
"neo4j_id": "uuid-xxx" # Neo4j节点ID
}
# Neo4j存储
(kp:KnowledgePoint {
id: "uuid-xxx",
name: "数据库索引",
error_count: 3,
mastery_level: 0.7
})
-[:RELATED_TO]->
(kp2:KnowledgePoint {name: "查询优化"})
```

6. 核心功能实现

6.1 RAG检索系统

向量检索流程:

```
class RAGService:
def __init__(self):
self.chroma_client = chromadb.Client()
self.embedding_model = QwenEmbedding()
async def search(
self,
kb_id: str,
query: str,
top_k: int = 5
) -> List[dict]:

# 1. 获取collection
collection = self.chroma_client.get_collection(f"kb_{kb_id}")
# 2. 查询向量化
query_vector = await self.embedding_model.embed(query)
# 3. 相似度检索
results = collection.query(
query_embeddings=[query_vector],
n_results=top_k,
include=['documents', 'metadatas', 'distances']
)
# 4. 格式化返回
documents = []
for i, doc in enumerate(results['documents'][0]):
documents.append({
'content': doc,
metadata': results['metadatas'][0][i],
'score': 1 - results['distances'][0][i] # 转为相似度
})
return documents
```

6.2 知识图谱可视化

D3.js力导向图:

```
// 前端实现
const renderGraph = (nodes, edges) => {
const simulation = d3.forceSimulation(nodes)
.force('link', d3.forceLink(edges).id(d => d.id))
.force('charge', d3.forceManyBody().strength(-400))
.force('center', d3.forceCenter(width/2, height/2))
// 节点颜色根据掌握度
const getColor = (mastery, errors) => {
if (errors >= 5) return '#f44336' // 红色-很多错误
if (errors >= 3) return '#ff9800' // 橙色-一些错误
if (mastery >= 0.8) return '#4caf50' // 绿色-已掌握
if (mastery >= 0.5) return '#2196f3' // 蓝色-学习中
return '#9e9e9e' // 灰色-未练习

}
// 节点大小根据重要度
const getSize = (node) => 20 + node.error_count * 2

}
```

6.3 错题本与薄弱点分析

智能推荐算法:

```
async def get_weak_points(kb_id: str, user_id: str) -> List[dict]:

# 1. 统计错题中的知识点
pipeline = [
{'$match': {'user_id': user_id, 'mastered': False}},
{'$unwind': '$knowledge_points'},
{'$group': {
'_id': '$knowledge_points',
'error_count': {'$sum': 1}
}},
{'$sort': {'error_count': -1}},
{'$limit': 10}
]
error_stats = await db.error_records.aggregate(pipeline).to_list()

# 2. 获取知识点详情

weak_points = []
```

```
for stat in error_stats:
kp = await db.knowledge_points.find_one({
'knowledge_base_id': kb_id,
'name': stat['_id']
})

if kp:
weak_points.append({
'name': kp['name'],
'error_count': stat['error_count'],
'mastery_level': kp.get('mastery_level', 0),
'error_rate': stat['error_count'] /
(kp['error_count'] + kp['correct_count'])
})
return weak_points
```

总结

技术亮点

1. **多智能体协作**: 题目生成、判卷评分、知识提取、AI辅导等多个Agent协同工作
2. **RAG增强**: 结合向量检索和LLM生成，确保答案准确性和相关性
3. **知识图谱**: 自动构建知识关系网络，可视化学习盲点
4. **个性化学习**: 基于错题分析和薄弱点识别，定制学习路径
5. **云原生架构**: 容器化部署，微服务设计，易于扩展

技术栈总览

层级	技术	用途
前端	React + Ant Design	UI 界面开发与交互
后端	FastAPI + Python	提供 API 接口服务
数据库	MongoDB + ChromaDB + Neo4j	多类型数据存储
AI	通义千问 API	大语言模型推理
容器化	Docker + Docker Compose	项目部署与运维管理
知识检索	RAG (ChromaDB)	基于语义的知识搜索
图谱	Neo4j + D3.js	知识关系可视化与存储