# 详细设计说明书

## 0. 文档元信息

* 文档版本：v1.0
* 作者/评审人：系统架构师团队
* 创建/更新日期：2025-09-01
* 关联需求/任务：飞机发动机QEC拆装智能化系统二期建设
* 适用范围：整个系统四大功能模块

## 1. 引言

### 1.1 背景与目标

本文档旨在详细描述飞机发动机QEC拆装智能化系统二期建设的完整详细设计方案，为开发人员、测试人员和运维人员提供技术实现指导。

系统通过引入人工智能技术，提高维修效率、优化资源配置，实现维修工作智能化管理的升级。系统主要包括维修知识库管理、维修规划及资源智能化、生产看板以及发动机检测辅助四大功能模块，旨在构建完善的维修知识体系，实现维修计划的智能生成与调整，优化资源配置，并提供多维度的生产状态可视化展示。

### 1.2 范围与不在范围

**在范围内：**

* 维修知识库管理：知识构建、检索、问答等全功能
* 维修规划及资源智能化：计划生成、资源配置、冲突解决
* 生产看板：多维度可视化展示、实时监控
* 发动机检测辅助：智能检测、报告生成、预警管理
* 系统集成：与SAP、文档管理等外部系统的接口
* 基础设施：数据存储、安全、监控等支撑服务

**不在范围内：**

* SAP系统内部功能实现
* 物流运输环节的业务流程
* 硬件设备的具体选型和采购
* 第三方系统的内部实现

### 1.3 术语与缩写

| 术语/缩略语 | 定义 |
| --- | --- |
| QEC | Quick Engine Change，快速发动机更换装置 |
| KB | Knowledge Base，知识库 |
| RAG | Retrieval-Augmented Generation，检索增强生成 |
| NLP | Natural Language Processing，自然语言处理 |
| API | Application Programming Interface，应用程序接口 |
| ETL | Extract Transform Load，数据提取转换加载 |
| VDB | Vector Database，向量数据库 |
| LLM | Large Language Model，大语言模型 |
| MMR | Maximal Marginal Relevance，最大边际相关性 |

### 1.4 参考资料（SRS/PRD/ADR/接口标准/规约）

1. 《飞机发动机QEC拆装智能化系统需求规格说明书》
2. 《飞机发动机QEC拆装智能化系统概要设计说明书》
3. 《飞机发动机QEC拆装业务流程文档》
4. 《机务维修AI辅助系统功能结构规划》
5. 《航空维修行业相关标准和规范》
6. RESTful API设计规范
7. 企业信息安全规范

### 1.3.1 术语与命名规范

* 资源标识：统一使用小写蛇形命名，dataset\_id, document\_id, segment\_id, child\_chunk\_id
* API 路径：以名词复数作为集合资源名，层级关系用子路径体现，如 /v1/datasets/{dataset\_id}/documents/{document\_id}
* 状态字段：使用全大写下划线命名的常量值，如 UPLOADED, PARSING, INDEXING, ENABLED
* 时间字段：统一使用 ISO8601，字段名以 \_at 结尾，UTC 存储
* 枚举字段：文档化列出可选值与默认值，后端侧使用 CHECK 约束或枚举类型
* 错误码：业务错误码为 6 位数字，按模块分配段，如 41xxxx(文件与文档), 42xxxx(检索), 43xxxx(问答)

## 2. 总体设计

### 2.1 架构视图（C4）



### 2.1.1 系统容器视图（C4-Container）

* 决策ADR-004：父子分段作为默认分段模式/原因：兼顾精准召回与上下文完整性/备选：仅通用分段/影响：问答准确率与可解释性提升/日期：2025-09-01
* 决策ADR-005：采用混合检索（语义+关键词）与可选重排序/原因：覆盖缩写/ID/专有名词等关键词优势，同时保留语义能力/备选：仅语义或仅关键词/影响：召回覆盖率提升，首字节延迟可控/日期：2025-09-01
* 决策ADR-006：引入多级缓存（本地内存+分布式KV）/原因：缓解热点查询与重复检索/备选：仅分布式缓存/影响：P95 延迟优化，成本下降/日期：2025-09-01
* 决策ADR-007：统一异步任务队列+DLQ/原因：提升稳定性与可恢复性/备选：同步处理/影响：峰值可用性提升/日期：2025-09-01
* 决策ADR-008：统一错误码与幂等键/原因：便于跨服务一致性与重试安全/备选：各自定义/影响：异常恢复效率提升/日期：2025-09-01



### 2.2 运行环境与技术栈

* **语言/框架**：Python/Flask，前端Next.js
* **运行环境**：Docker容器/Linux/16GB内存/8核CPU/高速网络
* **外部依赖**：关系型数据库/向量数据库/消息队列/对象存储/搜索引擎/AI推理服务

### 2.3 设计约束与关键决策（ADR）

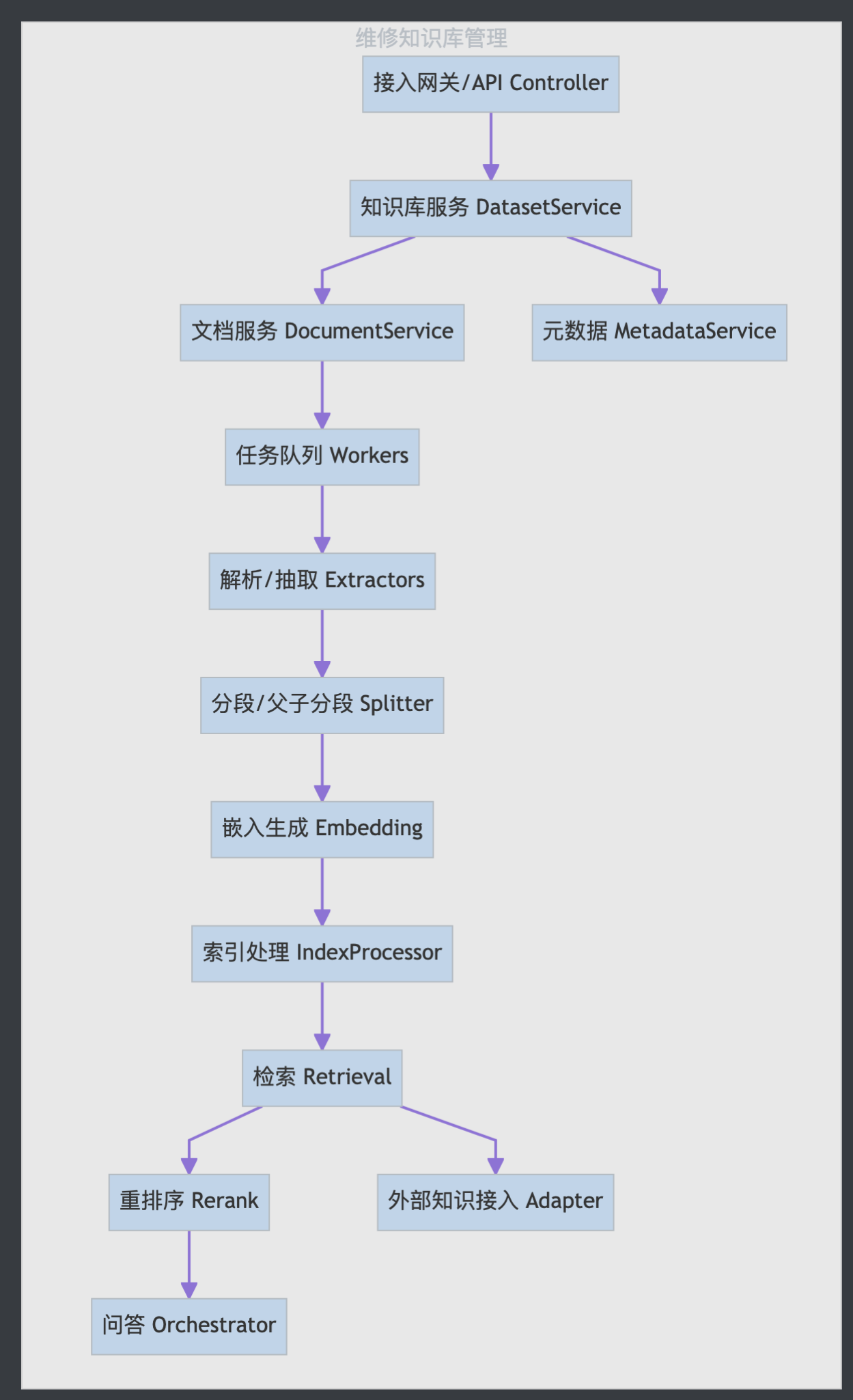
* **决策ADR-001**：采用检索增强生成(RAG)架构/原因：结合向量检索与生成模型优势/备选：纯检索或纯生成/影响：提升答案质量与可追溯性/日期：2025-09-01
* **决策ADR-002**：使用向量数据库存储知识嵌入/原因：支持语义检索与快速相似度计算/备选：传统全文检索/影响：检索精度显著提升/日期：2025-09-01
* **决策ADR-003**：采用分段化知识管理/原因：便于精确引用与上下文控制/备选：整篇文档检索/影响：提升引用精确性/日期：2025-09-01

## 3. 详细设计

### 3.1 模块清单

| 模块 | 职责 | 主要接口 | 数据实体 | 关键流程 |
| --- | --- | --- | --- | --- |
| 维修知识库管理 | 知识构建、检索、问答服务 | 知识库CRUD、问答API、检索服务 | Dataset、Document、Segment、Message | 知识接入→解析分段→向量索引→智能问答 |

#### 3.2.0 模块组件与关系（更新）



* 设计要点：支持通用分段与父子分段两种模式；提供混合检索（向量+关键词）与重排序；全过程异步化与可回溯日志。

| 维修规划及资源智能化 | 计划生成、资源配置、冲突解决 | 计划API、资源管理API、调度API | Plan、Resource、Task、Schedule | 需求分析→计划生成→资源分配→冲突解决 |  
| 生产看板 | 多维度可视化展示、实时监控 | 看板API、数据聚合API、通知API | Dashboard、Metric、Alert、Report | 数据采集→实时计算→可视化展示→告警通知 |  
| 发动机检测辅助 | 智能检测、数据分析、预警管理 | 检测API、分析API、预警API | Detection、Analysis、Warning、Report | 检测规划→数据采集→智能分析→预警报告 |

### 3.2 知识库构建与维护模块

#### 3.2.1 职责与边界

**核心职责**：

* 实现维修手册、历史维修记录、维修经验与结构化工程文件等多来源知识的接入、分段、标注、检索参数配置与全生命周期维护
* 提供知识库的创建、更新、删除和版本管理功能
* 支持多种文档格式的解析和标准化处理
* 建立知识分段的向量索引和元数据索引

**边界定义**：

* 处理范围：TXT/MD/PDF/HTML/XLSX/DOCX/CSV等格式文档
* 不处理：音频/视频文件的直接解析（可接受转录文本）
* 分段模式：
  + 通用分段：自定义分隔符/最大长度/重叠长度；适用常规文档
  + 父子分段：子块精确检索 + 父块上下文补全；适用手册类、结构清晰文档
* 索引方式：
  + 高质量索引：向量索引（语义检索，支持重排序）
  + 经济索引：关键词倒排（用于低成本/精确词匹配场景）
* 检索模式：语义/全文/混合检索；支持TopK、Score阈值与重排模型
* 支持文档大小：单文件≤100MB，批量上传≤1GB
* 支持知识库规模：单库≤10万文档，≤1000万分段

##### 3.2.1.1 自研能力与创新点

* 自适应分段器：基于文档结构与标点统计自动选择分段粒度，支持父子分段动态阈值与重叠比例自调优
* 语义去噪与重复检测：结合指纹哈希与相似度阈值，消除重复段落与模板性水印文本
* 多路召回融合：关键词检索与语义检索并行，按可学习权重融合，支持基于反馈的在线学习
* 上下文预算器：依据模型上下文窗口与TopK、引用密度自动分配上下文预算，保证答案可解释性
* 引用可追溯：回答内置可点击引用锚点，定位至文档/章节/分段，提升审计与复核效率
* 命中测试与A/B评价：内置检索命中测试与问答离线评估集，支持策略灰度与快速回滚
* 可靠性增强：队列DLQ+幂等键+补偿事务，跨组件失败可恢复，提供端到端重试视图

#### 3.2.2 接口定义（REST API）

| 接口 | 方法 | 路径 | 鉴权 | 入参 | 出参 | 错误码 |
| --- | --- | --- | --- | --- | --- | --- |
| 创建知识库 | POST | /v1/datasets | Bearer Token | name, permission, description | dataset\_id, status | 400, 401, 409 |
| 知识库列表 | GET | /v1/datasets | Bearer Token | page, limit, name\_filter | datasets[], total, page\_info | 400, 401 |
| 删除知识库 | DELETE | /v1/datasets/{dataset\_id} | Bearer Token | dataset\_id | success | 400, 401, 404 |
| 上传文档 | POST | /v1/datasets/{dataset\_id}/document/create-by-file | Bearer Token | multipart file, metadata | document\_id, status | 400, 401, 413, 415 |
| 文档列表 | GET | /v1/datasets/{dataset\_id}/documents | Bearer Token | page, limit, status\_filter | documents[], total | 400, 401, 404 |
| 更新文档 | POST | /v1/datasets/{dataset\_id}/documents/{document\_id}/update-by-file | Bearer Token | multipart file | document\_id, status | 400, 401, 404 |
| 删除文档 | DELETE | /v1/datasets/{dataset\_id}/documents/{document\_id} | Bearer Token | document\_id | success | 400, 401, 404 |
| 索引状态 | GET | /v1/datasets/{dataset\_id}/documents/{batch}/indexing-status | Bearer Token | batch\_id | indexing\_status, progress, error | 400, 401, 404 |

| 通过文本创建文档 | POST | /v1/datasets/{dataset\_id}/document/create-by-text | Bearer Token | name, text, process\_rule, retrieval\_model | document\_id, status | 400, 401, 413, 415 |  
| 通过文本更新文档 | POST | /v1/datasets/{dataset\_id}/documents/{document\_id}/update-by-text | Bearer Token | name, text, process\_rule | document\_id, status | 400, 401, 404 |

**示例请求**：

POST /v1/datasets HTTP/1.1  
Content-Type: application/json  
Authorization: Bearer eyJ0eXAiOiJKV1Q...  
  
{  
 "name": "维修手册知识库",  
 "permission": "only\_me",  
 "description": "X100发动机维修手册集合"  
}

**示例响应**：

{  
 "code": 0,  
 "message": "success",  
 "data": {  
 "id": "kb\_abc123",  
 "name": "维修手册知识库",  
 "permission": "only\_me",  
 "document\_count": 0,  
 "created\_at": "2024-12-19T10:30:00Z"  
 }  
}

#### 3.2.3 数据结构与存储

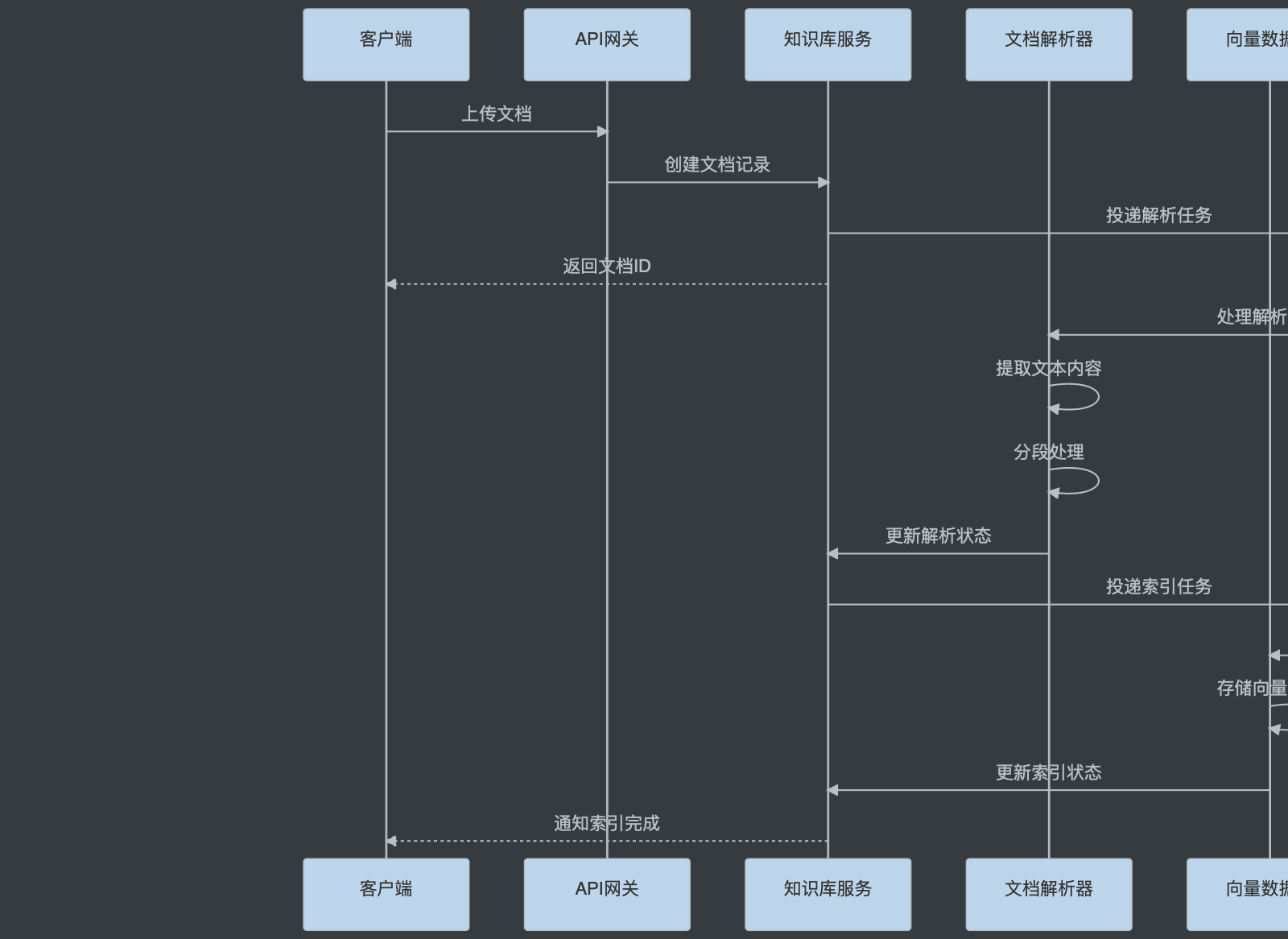
**逻辑模型与实体**：

-- 知识库主表  
CREATE TABLE datasets (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name VARCHAR(255) NOT NULL,  
 permission VARCHAR(50) NOT NULL CHECK (permission IN ('only\_me', 'all', 'partial')),  
 description TEXT,  
 created\_by UUID NOT NULL,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 UNIQUE(name, created\_by)  
);  
  
-- 文档表  
CREATE TABLE documents (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 dataset\_id UUID NOT NULL REFERENCES datasets(id) ON DELETE CASCADE,  
 name VARCHAR(500) NOT NULL,  
 file\_id VARCHAR(255),  
 status VARCHAR(50) NOT NULL DEFAULT 'uploaded'  
 CHECK (status IN ('uploaded', 'parsing', 'indexing', 'enabled', 'disabled', 'archived')),  
 word\_count INTEGER DEFAULT 0,  
 parsing\_completed\_at TIMESTAMP WITH TIME ZONE,  
 indexing\_completed\_at TIMESTAMP WITH TIME ZONE,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 文档分段表  
CREATE TABLE segments (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 document\_id UUID NOT NULL REFERENCES documents(id) ON DELETE CASCADE,  
 position INTEGER NOT NULL,  
 content TEXT NOT NULL,  
 word\_count INTEGER NOT NULL,  
 tokens INTEGER NOT NULL,  
 keywords TEXT[],  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 索引设计  
CREATE INDEX idx\_documents\_dataset\_id ON documents(dataset\_id);  
CREATE INDEX idx\_documents\_status ON documents(status);  
CREATE INDEX idx\_segments\_document\_id ON segments(document\_id);  
CREATE INDEX idx\_segments\_keywords ON segments USING GIN(keywords);

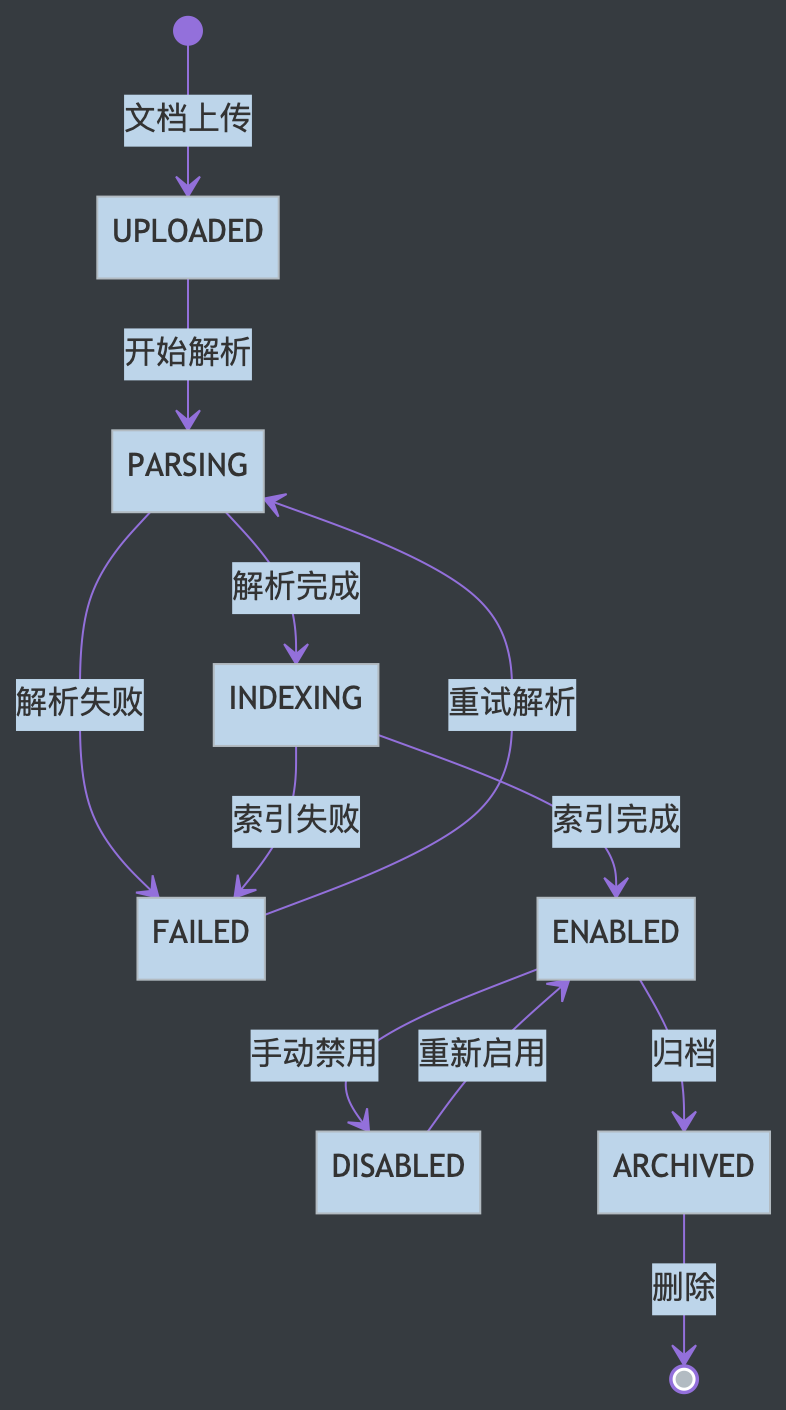
**缓存键设计与过期策略**：

* 知识库配置：kb:config:{dataset\_id} (TTL: 1小时)
* 文档状态：doc:status:{document\_id} (TTL: 30分钟)
* 检索结果：search:{query\_hash}:{dataset\_id} (TTL: 15分钟)

#### 3.2.4 核心流程



#### 3.2.5 状态机



#### 3.2.6 错误处理与重试

**错误分类、错误码、幂等策略**：

| 错误类型 | 错误码 | 描述 | 重试策略 | 幂等处理 |
| --- | --- | --- | --- | --- |
| 文件格式错误 | 415001 | 不支持的文件格式 | 不重试 | 返回相同错误 |
| 文件过大 | 413001 | 文件大小超限 | 不重试 | 返回相同错误 |
| 解析失败 | 500001 | 文档内容解析失败 | 指数退避，最多3次 | 基于文档ID幂等 |
| 索引失败 | 500002 | 向量索引构建失败 | 指数退避，最多5次 | 基于分段ID幂等 |
| 知识库名称冲突 | 409001 | 知识库名称已存在 | 不重试 | 返回现有知识库ID |

**补偿事务**：

* 文档上传失败：清理已上传的文件和数据库记录
* 索引构建失败：保留文档记录，标记为失败状态，支持重新索引

#### 3.2.7 配置项

| 键 | 默认值 | 范围/限制 | 作用 | 生效方式 |
| --- | --- | --- | --- | --- |
| MAX\_FILE\_SIZE | 100MB | 1MB~500MB | 单文件最大大小 | 重启生效 |
| MAX\_BATCH\_SIZE | 1GB | 100MB~5GB | 批量上传最大大小 | 重启生效 |
| PARSING\_TIMEOUT | 300s | 60s~1800s | 文档解析超时时间 | 实时生效 |
| INDEXING\_BATCH\_SIZE | 100 | 10~1000 | 批量索引分段数量 | 实时生效 |
| RETRY\_MAX\_ATTEMPTS | 3 | 1~10 | 最大重试次数 | 实时生效 |
| CACHE\_TTL | 3600s | 300s~86400s | 缓存过期时间 | 实时生效 |

#### 3.2.8 性能与容量规划

**QPS/吞吐/延迟目标**：

* 文档上传：50 QPS，响应时间 ≤ 2秒
* 文档列表查询：200 QPS，响应时间 ≤ 500ms
* 文档解析：单文档10MB ≤ 30秒
* 向量索引：1000分段/分钟

**并发处理**：

* 支持100个并发上传任务
* 解析队列：10个worker并行处理
* 索引队列：5个worker并行处理

**扩展策略**：

* 水平扩展：API服务无状态，支持负载均衡
* 存储扩展：数据库分片，向量库集群
* 计算扩展：解析和索引服务独立扩容

#### 3.2.9 安全与合规

**鉴权鉴权**：

* JWT Token身份验证
* 基于角色的权限控制（RBAC）
* 知识库级别的访问控制

**加密与审计**：

* 文档存储AES-256加密
* 敏感元数据脱敏显示
* 操作日志审计记录：创建、更新、删除、访问

**权限最小化**：

* 用户仅能访问授权的知识库
* API接口按功能最小权限设计
* 数据库连接使用专用账户

#### 3.2.10 可观测性

**日志等级与结构**：

{  
 "timestamp": "2024-12-19T10:30:00Z",  
 "level": "INFO",  
 "service": "knowledge-service",  
 "module": "document-upload",  
 "trace\_id": "abc123",  
 "user\_id": "user456",  
 "dataset\_id": "kb\_789",  
 "message": "Document uploaded successfully",  
 "duration\_ms": 1200,  
 "file\_size": 2048576  
}

**指标（Prometheus格式）**：

* knowledge\_document\_upload\_total{status="success|failed"} - 文档上传计数
* knowledge\_document\_parsing\_duration\_seconds - 解析耗时分布
* knowledge\_indexing\_queue\_size - 索引队列长度
* knowledge\_storage\_usage\_bytes{dataset\_id} - 存储空间使用

**告警阈值**：

* 文档上传失败率 > 5%
* 解析队列积压 > 100个任务
* 索引构建失败率 > 2%
* 存储使用率 > 80%

### 3.3 智能问答服务模块

#### 3.3.1 职责与边界

**核心职责**：

* 实现面向维修场景的自然语言问题解析与答案生成
* 结合内部维修知识库进行检索增强生成(RAG)
* 提供可解释的答案与引用追溯功能
* 支持多轮对话和上下文理解

**边界定义**：

* 处理自然语言问题，长度≤2000字符
* 支持文本、图片、文档等多模态输入
* 答案生成时间：简单问题≤3秒，复杂问题≤10秒
* 支持的语言：中文、英文

#### 3.3.2 接口定义（REST API）

| 接口 | 方法 | 路径 | 鉴权 | 入参 | 出参 | 错误码 |
| --- | --- | --- | --- | --- | --- | --- |
| 发送问题 | POST | /v1/chat-messages | Bearer Token | query, inputs, files, response\_mode | answer, conversation\_id, message\_id | 400, 401, 429 |
| 停止响应 | POST | /v1/chat-messages/{task\_id}/stop | Bearer Token | task\_id | success | 400, 401, 404 |
| 上传文件 | POST | /v1/files/upload | Bearer Token | multipart file | file\_id, name, size | 400, 401, 413, 415 |
| 历史消息 | GET | /v1/messages | Bearer Token | conversation\_id, limit, before\_id | messages[], has\_more | 400, 401, 404 |
| 会话列表 | GET | /v1/conversations | Bearer Token | limit, before\_id | conversations[], has\_more | 400, 401 |
| 消息反馈 | POST | /v1/messages/{message\_id}/feedbacks | Bearer Token | rating, content | success | 400, 401, 404 |

**示例请求**：

POST /v1/chat-messages HTTP/1.1  
Content-Type: application/json  
Authorization: Bearer eyJ0eXAiOiJKV1Q...  
  
  
#### 3.3.11 检索增强生成（RAG）细节  
  
- 候选召回：  
 - 语义通道：查询向量化后在VDB中近邻检索，召回TopK\_s  
 - 关键词通道：倒排索引检索，召回TopK\_k  
 - 融合：按归一化分数执行MMR或学习到的权重融合，得到候选集K  
- 重排序：  
 - 基于启发式权重（语义相似、关键词覆盖、段落层级、时间衰减）或训练好的rerank模型重排  
- 上下文构建：  
 - 父子合并：命中子块时，自动选取对应父块，控制上下文预算与冗余去重  
 - 预算器：根据模型上下文窗口、TopK、引用密度分配tokens上限  
- 生成与引用：  
 - 提示词模板注入引用占位符，生成后将段落引用替换为可追溯链接  
- 质量反馈闭环：  
 - 收集点击/评分/命中率指标，离线驱动召回权重/阈值自动微调  
  
{  
 "query": "X100发动机启动失败怎么处理？",  
 "inputs": {  
 "device\_model": "X100",  
 "manual\_version": "v2.0"  
 },  
 "response\_mode": "streaming",  
 "conversation\_id": "conv\_abc123"  
}

**示例响应**：

{  
 "event": "message",  
 "conversation\_id": "conv\_abc123",  
 "message\_id": "msg\_def456",  
 "answer": "根据X100发动机维修手册，启动失败的排查步骤如下：\n\n1. 检查燃油供应系统...",  
 "retriever\_resources": [  
 {  
 "segment\_id": "seg\_789",  
 "document\_name": "X100维修手册v2.0",  
 "content": "启动系统故障排除...",  
 "score": 0.95,  
 "position": {"page": 156, "section": "3.2.4"}  
 }  
 ],  
 "metadata": {  
 "usage": {"tokens": 1250, "cost": 0.025}  
 }  
}

#### 3.3.3 数据结构与存储

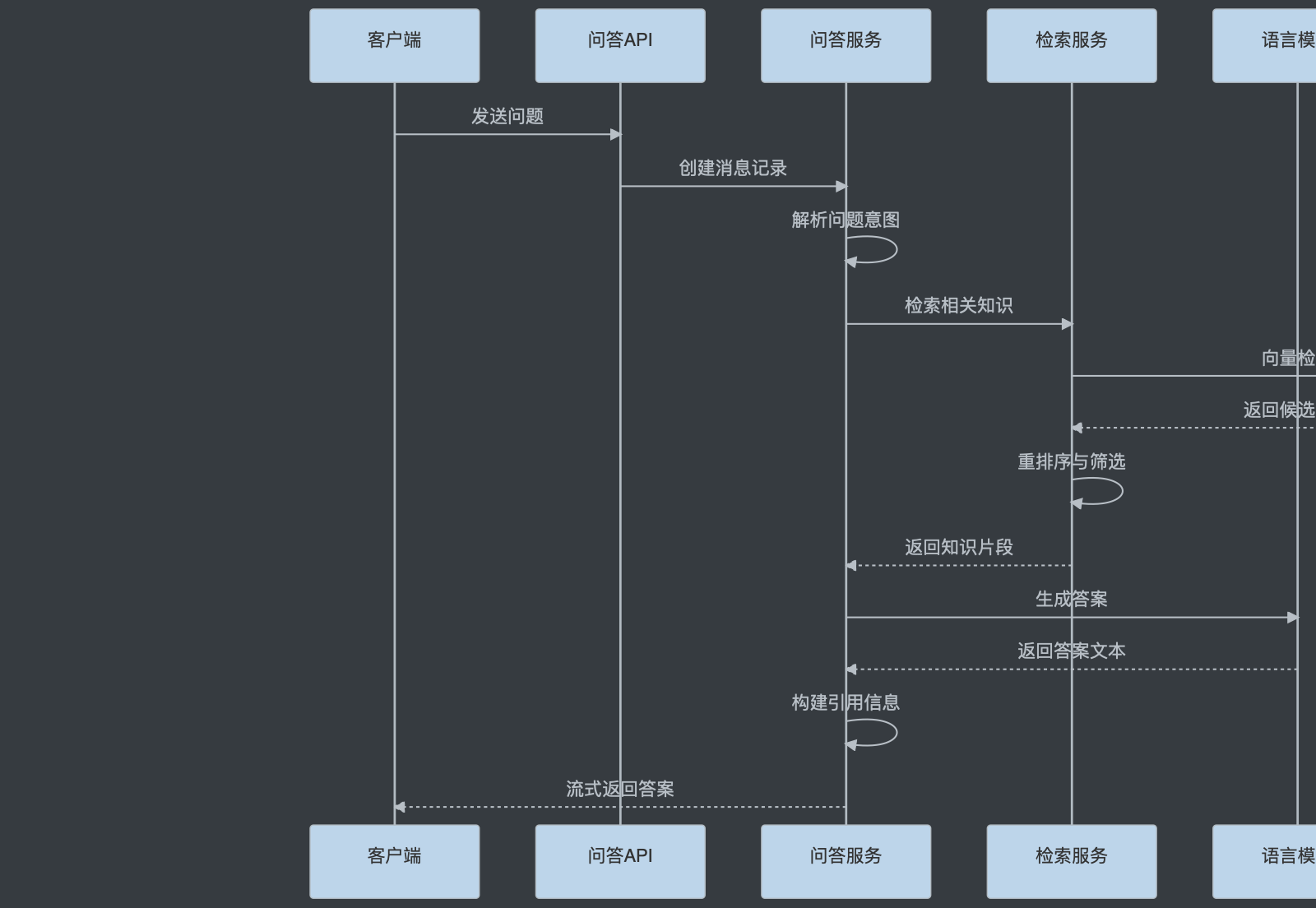
**逻辑模型与实体**：

-- 会话表  
CREATE TABLE conversations (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID NOT NULL,  
 name VARCHAR(255),  
 inputs JSONB,  
 status VARCHAR(50) DEFAULT 'active' CHECK (status IN ('active', 'archived')),  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 消息表  
CREATE TABLE messages (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 conversation\_id UUID NOT NULL REFERENCES conversations(id) ON DELETE CASCADE,  
 parent\_message\_id UUID REFERENCES messages(id),  
 query TEXT,  
 answer TEXT,  
 error\_message TEXT,  
 usage JSONB,  
 retriever\_resources JSONB,  
 agent\_thoughts JSONB,  
 message\_tokens INTEGER DEFAULT 0,  
 answer\_tokens INTEGER DEFAULT 0,  
 total\_tokens INTEGER DEFAULT 0,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 消息反馈表  
CREATE TABLE message\_feedbacks (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 message\_id UUID NOT NULL REFERENCES messages(id) ON DELETE CASCADE,  
 rating VARCHAR(20) CHECK (rating IN ('like', 'dislike')),  
 content TEXT,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 索引设计  
CREATE INDEX idx\_conversations\_user\_id ON conversations(user\_id);  
CREATE INDEX idx\_messages\_conversation\_id ON messages(conversation\_id);  
CREATE INDEX idx\_messages\_created\_at ON messages(created\_at);  
CREATE INDEX idx\_message\_feedbacks\_message\_id ON message\_feedbacks(message\_id);

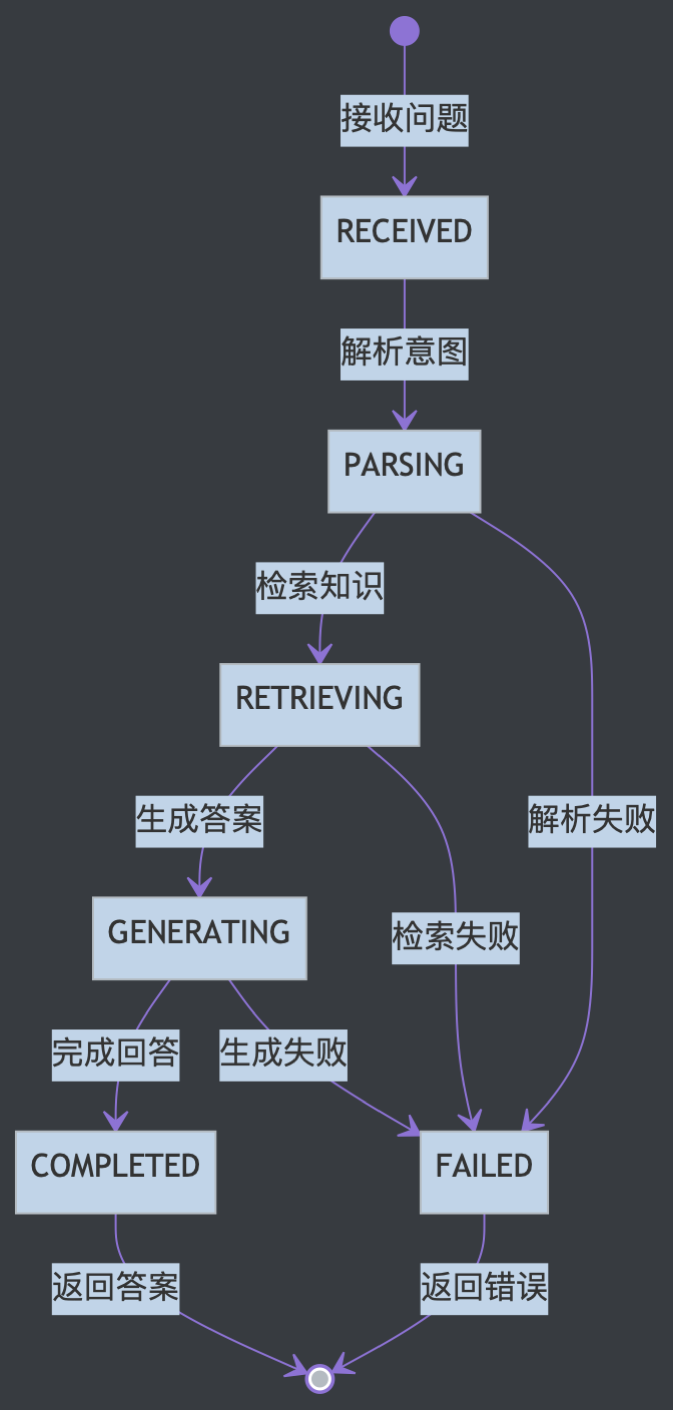
**缓存键设计与过期策略**：

* 会话上下文：conv:context:{conversation\_id} (TTL: 2小时)
* 用户偏好：user:pref:{user\_id} (TTL: 24小时)
* 问答缓存：qa:cache:{query\_hash}:{inputs\_hash} (TTL: 1小时)

#### 3.3.4 核心流程



#### 3.3.5 状态机



#### 3.3.6 错误处理与重试

**错误分类、错误码、幂等策略**：

| 错误类型 | 错误码 | 描述 | 重试策略 | 幂等处理 |
| --- | --- | --- | --- | --- |
| 问题格式错误 | 400001 | 问题内容为空或格式错误 | 不重试 | 返回相同错误 |
| 模型服务超时 | 504001 | 语言模型响应超时 | 指数退避，最多2次 | 基于消息ID幂等 |
| 检索服务异常 | 500003 | 知识库检索服务异常 | 线性退避，最多3次 | 基于查询哈希幂等 |
| 配额不足 | 429001 | 用户请求配额不足 | 不重试 | 返回配额重置时间 |
| 会话不存在 | 404001 | 指定会话ID不存在 | 不重试 | 创建新会话 |

**补偿事务**：

* 生成失败：保留用户问题，标记答案状态为失败
* 检索异常：降级为基础回答模式，不提供引用

#### 3.3.7 配置项

| 键 | 默认值 | 范围/限制 | 作用 | 生效方式 |
| --- | --- | --- | --- | --- |
| MAX\_QUERY\_LENGTH | 2000 | 100~5000 | 问题最大长度 | 实时生效 |
| RETRIEVAL\_TOP\_K | 8 | 1~50 | 检索返回片段数 | 实时生效 |
| MIN\_RELEVANCE\_SCORE | 0.3 | 0.1~0.9 | 最低相关度阈值 | 实时生效 |
| GENERATION\_TIMEOUT | 30s | 10s~120s | 答案生成超时时间 | 实时生效 |
| MAX\_TOKENS | 2048 | 512~4096 | 最大生成token数 | 实时生效 |
| CONVERSATION\_TTL | 7200s | 3600s~86400s | 会话上下文保持时间 | 实时生效 |

#### 3.3.8 性能与容量规划

**QPS/吞吐/延迟目标**：

* 问答请求：100 QPS，响应时间 ≤ 3秒(简单问题)
* 流式响应：首字节延迟 ≤ 1秒
* 历史消息查询：500 QPS，响应时间 ≤ 200ms
* 并发会话：支持1000个活跃会话

**并发处理**：

* 问答服务：20个worker并行处理
* 检索服务：异步调用，无阻塞
* 生成服务：支持流式处理

**扩展策略**：

* 水平扩展：问答服务无状态设计
* 缓存优化：多级缓存减少重复计算
* 模型优化：支持模型负载均衡

#### 3.3.9 安全与合规

**鉴权鉴权**：

* JWT Token身份验证
* 用户级别的问答配额控制
* 敏感问题内容过滤

**加密与审计**：

* 会话内容传输加密
* 问答历史审计日志
* 用户隐私数据脱敏

**权限最小化**：

* 用户仅能访问自己的会话
* 知识库访问权限继承
* API调用频率限制

#### 3.3.10 可观测性

**日志等级与结构**：

{  
 "timestamp": "2024-12-19T10:30:00Z",  
 "level": "INFO",  
 "service": "qa-service",  
 "module": "chat-completion",  
 "trace\_id": "xyz789",  
 "user\_id": "user456",  
 "conversation\_id": "conv\_123",  
 "message\_id": "msg\_456",  
 "message": "Answer generated successfully",  
 "duration\_ms": 2800,  
 "tokens\_used": 1250,  
 "relevance\_score": 0.87  
}

**指标（Prometheus格式）**：

* qa\_messages\_total{status="success|failed|timeout"} - 问答消息计数
* qa\_response\_duration\_seconds - 响应时间分布
* qa\_tokens\_usage\_total - Token使用量
* qa\_relevance\_score\_histogram - 相关度分数分布

**告警阈值**：

* 问答失败率 > 3%
* 平均响应时间 > 5秒
* Token使用率 > 90%
* 相关度分数 < 0.5的比例 > 20%

### 3.4 检索与定位服务模块

#### 3.4.1 职责与边界

**核心职责**：

* 实现"按手册定位"的精准检索与跳转功能
* 提供关键词/语义/混合检索与父子分段上下文合并
* 支持多路召回、重排序与去重优化
* 提供检索质量监控与策略调优能力

**边界定义**：

* 支持向量检索、全文检索、混合检索三种模式
* 单次检索返回分段数：1~50个可配置
* 检索响应时间：≤1秒
* 支持外部知识库联合检索

#### 3.4.2 接口定义（REST API）

| 接口 | 方法 | 路径 | 鉴权 | 入参 | 出参 | 错误码 |
| --- | --- | --- | --- | --- | --- | --- |
| 知识检索 | POST | /v1/retrieval/search | Bearer Token | query, dataset\_ids, mode, top\_k, score\_threshold | segments[], total\_found, search\_id | 400, 401, 404 |
| 手册定位 | POST | /v1/retrieval/locate | Bearer Token | query, device\_model, manual\_version, section\_hint | location\_info, context\_preview, reference\_link | 400, 401, 404 |
| 检索测试 | POST | /v1/retrieval/hit-testing | Bearer Token | query, dataset\_id, retrieval\_model | segments[], scores[], debug\_info | 400, 401, 404 |
| 策略配置 | POST | /v1/retrieval/strategy | Bearer Token | dataset\_id, mode, parameters | strategy\_id, status | 400, 401, 404 |
| 外部检索 | POST | /v1/retrieval/external | Bearer Token | query, external\_kb\_id, max\_segments, min\_relevance | segments[], source\_info | 400, 401, 408 |

**示例请求**：

POST /v1/retrieval/locate HTTP/1.1  
Content-Type: application/json  
Authorization: Bearer eyJ0eXAiOiJKV1Q...  
  
{  
 "query": "如何复位燃油控制单元？",  
 "device\_model": "X100",  
 "manual\_version": "v2.0",  
 "section\_hint": "故障排除"  
}

**示例响应**：

{  
 "code": 0,  
 "message": "success",  
 "data": {  
 "location\_info": {  
 "manual\_name": "X100维修手册v2.0",  
 "chapter": "第3章 故障排除",  
 "section": "3.2.4",  
 "page": 156,  
 "relevance\_score": 0.95  
 },  
 "context\_preview": "燃油控制单元复位程序：\n1. 关闭主电源开关\n2. 等待30秒...",  
 "reference\_link": "/datasets/kb\_123/documents/doc\_456/segments/seg\_789"  
 }  
}

#### 3.4.3 数据结构与存储

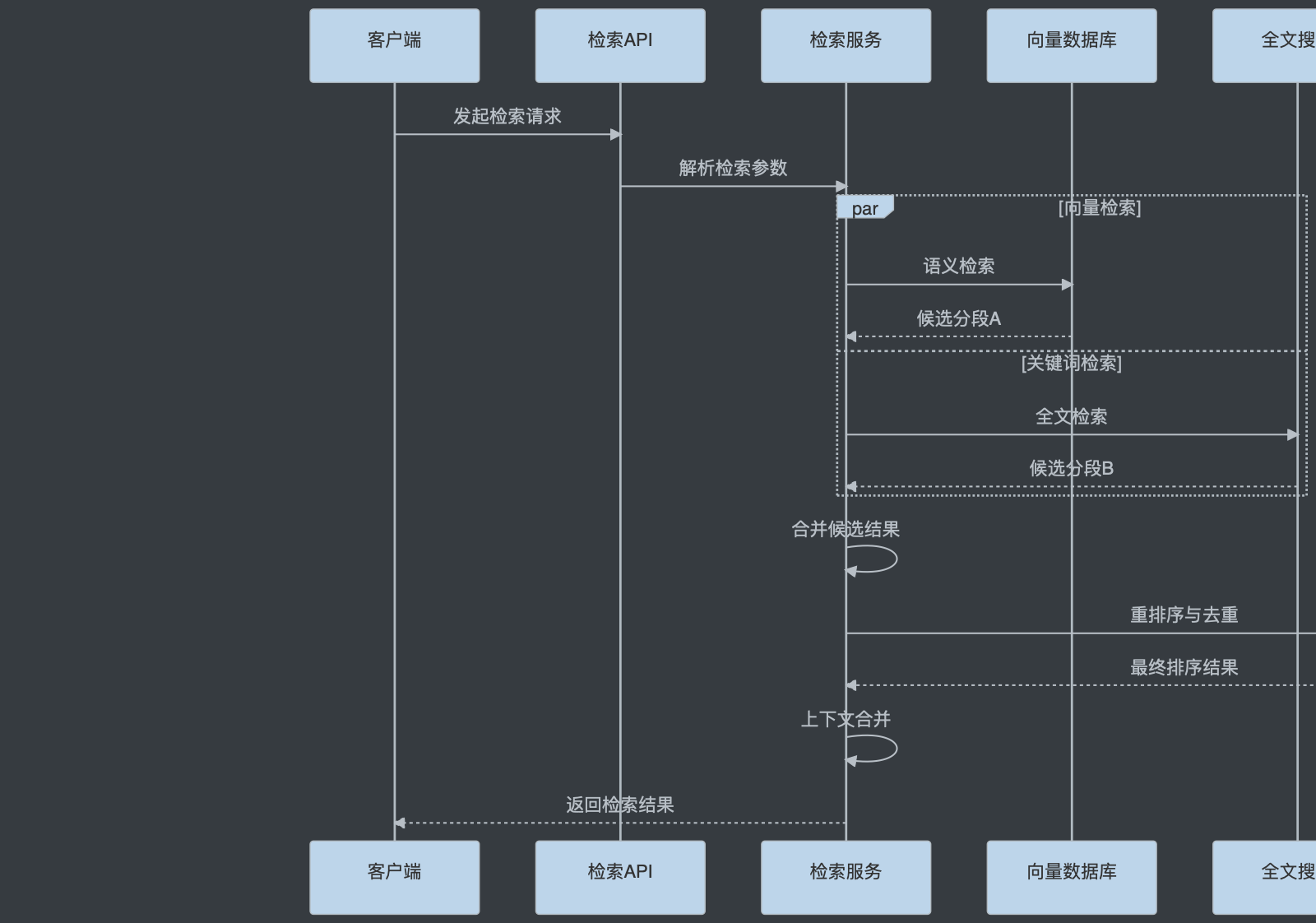
**逻辑模型与实体**：

-- 检索记录表  
CREATE TABLE retrieval\_records (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 dataset\_id UUID NOT NULL,  
 query TEXT NOT NULL,  
 mode VARCHAR(50) NOT NULL CHECK (mode IN ('vector', 'keyword', 'hybrid')),  
 parameters JSONB,  
 results\_count INTEGER DEFAULT 0,  
 duration\_ms INTEGER,  
 user\_id UUID,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 检索策略表  
CREATE TABLE retrieval\_strategies (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 dataset\_id UUID NOT NULL,  
 name VARCHAR(255) NOT NULL,  
 mode VARCHAR(50) NOT NULL,  
 parameters JSONB NOT NULL,  
 is\_active BOOLEAN DEFAULT false,  
 created\_by UUID NOT NULL,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
  
### 3.4.11 检索评测与观测  
- 线下评测：命中率(Recall@K)、精确率(Precision@K)、nDCG、覆盖率、重复率  
- 线上指标：首字节延迟、P95 响应、TopK命中位置分布、重排提升率  
- 诊断工具：查询回放、阈值灵敏度曲线、权重热力图、错误用例库  
  
 UNIQUE(dataset\_id, name)  
);  
  
-- 测试用例表  
CREATE TABLE test\_cases (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 dataset\_id UUID NOT NULL,  
 query TEXT NOT NULL,  
 expected\_segments UUID[],  
 expected\_score\_threshold FLOAT DEFAULT 0.5,  
 tags TEXT[],  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 索引设计  
CREATE INDEX idx\_retrieval\_records\_dataset\_id ON retrieval\_records(dataset\_id);  
CREATE INDEX idx\_retrieval\_records\_created\_at ON retrieval\_records(created\_at);  
CREATE INDEX idx\_retrieval\_strategies\_dataset\_id ON retrieval\_strategies(dataset\_id);  
CREATE INDEX idx\_test\_cases\_dataset\_id ON test\_cases(dataset\_id);

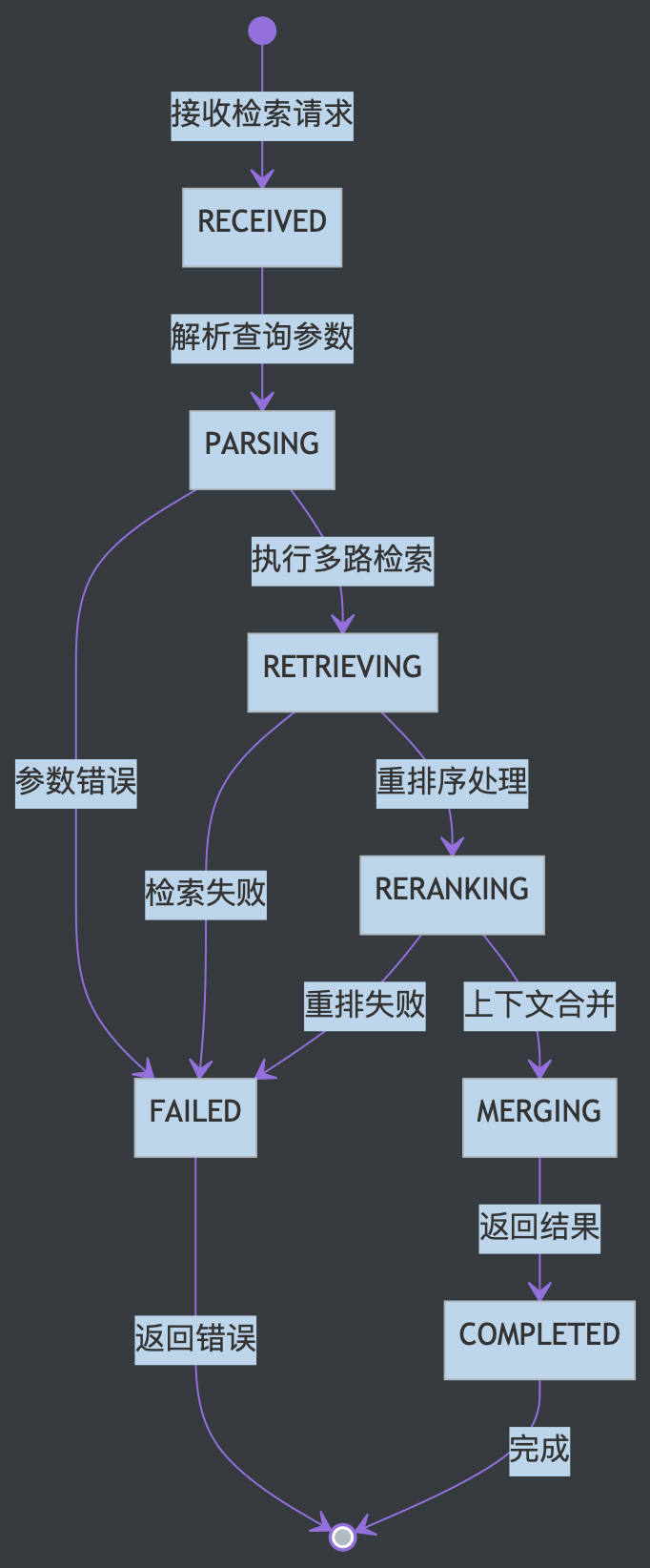
**缓存键设计与过期策略**：

* 检索结果：retrieval:cache:{query\_hash}:{params\_hash} (TTL: 15分钟)
* 策略配置：retrieval:strategy:{dataset\_id} (TTL: 1小时)
* 向量嵌入：embedding:cache:{text\_hash} (TTL: 24小时)

#### 3.4.4 核心流程



#### 3.4.5 状态机



#### 3.4.6 错误处理与重试

**错误分类、错误码、幂等策略**：

| 错误类型 | 错误码 | 描述 | 重试策略 | 幂等处理 |
| --- | --- | --- | --- | --- |
| 查询参数错误 | 400002 | 检索参数格式错误 | 不重试 | 返回相同错误 |
| 向量库连接失败 | 500004 | 向量数据库不可用 | 指数退避，最多3次 | 基于查询哈希幂等 |
| 检索超时 | 504002 | 检索操作超时 | 线性退避，最多2次 | 降级策略 |
| 知识库不存在 | 404002 | 指定知识库不存在 | 不重试 | 返回空结果 |
| 外部KB超时 | 408001 | 外部知识库响应超时 | 降级为内部检索 | 记录降级日志 |

**补偿事务**：

* 检索失败：降级为基础关键词检索
* 重排失败：返回原始检索结果
* 外部KB失败：仅使用内部知识库

#### 3.4.7 配置项

| 键 | 默认值 | 范围/限制 | 作用 | 生效方式 |
| --- | --- | --- | --- | --- |
| DEFAULT\_TOP\_K | 10 | 1~50 | 默认返回分段数 | 实时生效 |
| MIN\_SCORE\_THRESHOLD | 0.3 | 0.0~1.0 | 最低相关度阈值 | 实时生效 |
| SEARCH\_TIMEOUT | 5s | 1s~30s | 检索超时时间 | 实时生效 |
| MAX\_CONTEXT\_MERGE | 3 | 1~10 | 最大上下文合并数 | 实时生效 |
| CACHE\_ENABLED | true | true/false | 是否启用结果缓存 | 实时生效 |
| EXTERNAL\_KB\_TIMEOUT | 3s | 1s~10s | 外部KB超时时间 | 实时生效 |

#### 3.4.8 性能与容量规划

**QPS/吞吐/延迟目标**：

* 检索请求：300 QPS，响应时间 ≤ 1秒
* 手册定位：200 QPS，响应时间 ≤ 800ms
* 检索测试：50 QPS，响应时间 ≤ 2秒
* 批量检索：支持100个并发请求

**并发处理**：

* 检索服务：30个worker并行处理
* 向量检索：异步并发调用
* 重排服务：CPU密集型，独立扩容

**扩展策略**：

* 水平扩展：检索服务无状态，支持负载均衡
* 缓存优化：多级缓存(内存+Redis)减少延迟
* 向量库优化：分片与副本提高可用性

#### 3.4.9 安全与合规

**鉴权鉴权**：

* JWT Token身份验证
* 知识库级别的检索权限控制
* API访问频率限制

**加密与审计**：

* 检索历史审计记录
* 敏感查询内容脱敏
* 结果访问权限验证

**权限最小化**：

* 用户仅能检索授权知识库
* 检索结果按权限过滤
* 外部KB访问权限控制

#### 3.4.10 可观测性

**日志等级与结构**：

{  
 "timestamp": "2024-12-19T10:30:00Z",  
 "level": "INFO",  
 "service": "retrieval-service",  
 "module": "vector-search",  
 "trace\_id": "ret123",  
 "user\_id": "user456",  
 "dataset\_id": "kb\_789",  
 "query\_hash": "abc123",  
 "message": "Retrieval completed successfully",  
 "duration\_ms": 800,  
 "results\_count": 8,  
 "top\_score": 0.89  
}

**指标（Prometheus格式）**：

* retrieval\_requests\_total{mode="vector|keyword|hybrid", status="success|failed"} - 检索请求计数
* retrieval\_duration\_seconds{mode} - 检索延迟分布
* retrieval\_results\_count{dataset\_id} - 返回结果数分布
* retrieval\_cache\_hit\_ratio - 缓存命中率

**告警阈值**：

* 检索失败率 > 2%
* 平均检索延迟 > 1.5秒
* 缓存命中率 < 60%
* 向量库连接失败率 > 1%

### 3.4.1 元数据管理子模块

#### 3.4.1.1 职责与边界

**核心职责**：

* 管理知识库全局与文档级元数据字段定义
* 支持按来源/型号/版本/时间/标签等维度筛选
* 提供可追溯引用的元数据标注功能
* 实现元数据的批量编辑和版本管理

**边界定义**：

* 支持string/number/time三种字段类型
* 元数据字段数量：单知识库≤50个
* 批量标注：单次≤1000个文档
* 字段名命名规范：小写字母/数字/下划线

#### 3.4.1.2 接口定义（REST API）

| 接口 | 方法 | 路径 | 鉴权 | 入参 | 出参 | 错误码 |
| --- | --- | --- | --- | --- | --- | --- |
| 新增元数据字段 | POST | /v1/datasets/{dataset\_id}/metadata | Bearer Token | name, type, is\_required | field\_id, status | 400, 401, 409 |
| 更新元数据字段 | PATCH | /v1/datasets/{dataset\_id}/metadata/{field\_id} | Bearer Token | name, type, is\_required | field\_id, status | 400, 401, 404 |
| 删除元数据字段 | DELETE | /v1/datasets/{dataset\_id}/metadata/{field\_id} | Bearer Token | field\_id | success | 400, 401, 404, 409 |
| 元数据字段列表 | GET | /v1/datasets/{dataset\_id}/metadata | Bearer Token | - | fields[], total | 400, 401, 404 |
| 批量标注文档 | POST | /v1/datasets/{dataset\_id}/documents/metadata | Bearer Token | document\_ids[], field\_values | updated\_count | 400, 401, 413 |

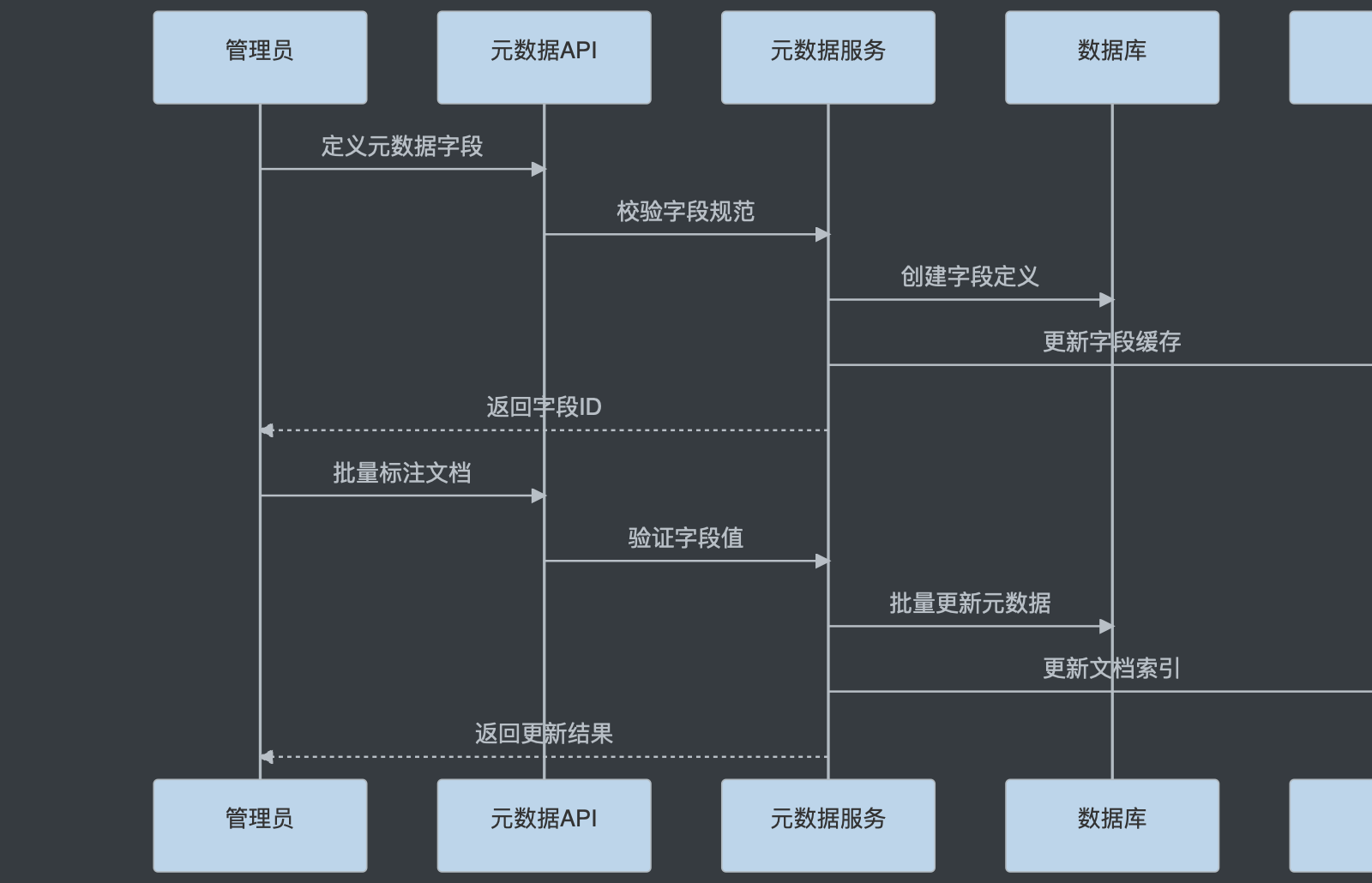
**示例请求**：

POST /v1/datasets/{dataset\_id}/metadata HTTP/1.1  
Content-Type: application/json  
Authorization: Bearer eyJ0eXAiOiJKV1Q...  
  
{  
 "name": "device\_model",  
 "type": "string",  
 "is\_required": true,  
 "description": "设备型号信息"  
}

#### 3.4.1.3 数据结构与存储

-- 元数据字段定义表  
CREATE TABLE metadata\_fields (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 dataset\_id UUID NOT NULL REFERENCES datasets(id) ON DELETE CASCADE,  
 name VARCHAR(50) NOT NULL,  
 type VARCHAR(20) NOT NULL CHECK (type IN ('string', 'number', 'time')),  
 is\_required BOOLEAN DEFAULT false,  
 description TEXT,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 UNIQUE(dataset\_id, name)  
);  
  
-- 文档元数据值表  
CREATE TABLE document\_metadata\_values (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 document\_id UUID NOT NULL REFERENCES documents(id) ON DELETE CASCADE,  
 field\_id UUID NOT NULL REFERENCES metadata\_fields(id) ON DELETE CASCADE,  
 value TEXT NOT NULL,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP,  
 UNIQUE(document\_id, field\_id)  
);  
  
-- 索引设计  
CREATE INDEX idx\_metadata\_fields\_dataset ON metadata\_fields(dataset\_id, name);  
CREATE INDEX idx\_doc\_metadata\_field\_value ON document\_metadata\_values(field\_id, value);  
CREATE INDEX idx\_doc\_metadata\_document ON document\_metadata\_values(document\_id);

#### 3.4.1.4 核心流程



### 3.4.2 外部知识库接入子模块

#### 3.4.2.1 职责与边界

**核心职责**：

* 通过标准化API接入外部知识库
* 实现统一的检索结果格式转换
* 提供外部知识与内部知识的合并重排
* 监控外部知识库的可用性和性能

**边界定义**：

* 支持HTTP/HTTPS协议的外部API
* 单次检索返回：≤50个分段
* 超时控制：连接超时≤3秒，读取超时≤10秒
* 支持基本认证和Token认证

#### 3.4.2.2 接口定义（REST API）

| 接口 | 方法 | 路径 | 鉴权 | 入参 | 出参 | 错误码 |
| --- | --- | --- | --- | --- | --- | --- |
| 配置外部知识库 | POST | /v1/external-kb/config | Bearer Token | endpoint, credential, kb\_id | config\_id, status | 400, 401, 422 |
| 外部检索 | POST | /v1/external-kb/search | Bearer Token | query, kb\_id, max\_segments | segments[], source\_info | 400, 401, 408, 422 |
| 测试连接 | POST | /v1/external-kb/test | Bearer Token | config\_id | connectivity, response\_time | 400, 401, 408 |
| 外部库列表 | GET | /v1/external-kb/configs | Bearer Token | - | configs[], total | 400, 401 |

#### 3.4.2.3 数据结构与存储

-- 外部知识库配置表  
CREATE TABLE external\_kb\_configs (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name VARCHAR(255) NOT NULL,  
 endpoint VARCHAR(500) NOT NULL,  
 credential\_type VARCHAR(50) NOT NULL CHECK (credential\_type IN ('basic', 'token', 'none')),  
 credential\_data JSONB,  
 external\_kb\_id VARCHAR(255),  
 timeout\_seconds INTEGER DEFAULT 10,  
 max\_segments INTEGER DEFAULT 20,  
 is\_active BOOLEAN DEFAULT true,  
 created\_by UUID NOT NULL,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 外部检索记录表  
CREATE TABLE external\_retrieval\_logs (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 config\_id UUID NOT NULL REFERENCES external\_kb\_configs(id),  
 query TEXT NOT NULL,  
 response\_time\_ms INTEGER,  
 status VARCHAR(20) NOT NULL,  
 error\_message TEXT,  
 segments\_count INTEGER DEFAULT 0,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);

### 3.4.3 关键词与数据索引子模块

#### 3.4.3.1 职责与边界

**核心职责**：

* 建立和维护关键词体系（核心词/同义词/禁用词）
* 构建多维数据索引支持精确筛选
* 提供检索策略的参数配置和调优
* 监控索引质量和检索效果

**边界定义**：

* 词表规模：核心词≤10000个，同义词≤50000个
* 索引维度：≤10个元数据维度
* 策略版本：支持≤5个历史版本回滚
* 重建频率：增量更新≤1小时，全量重建≤24小时

#### 3.4.3.2 接口定义（REST API）

| 接口 | 方法 | 路径 | 鉴权 | 入参 | 出参 | 错误码 |
| --- | --- | --- | --- | --- | --- | --- |
| 管理词表 | POST | /v1/datasets/{dataset\_id}/keywords | Bearer Token | core\_terms[], synonyms[], stopwords[] | version\_id, status | 400, 401, 413 |
| 配置索引策略 | POST | /v1/datasets/{dataset\_id}/index-strategy | Bearer Token | dimensions[], weights, params | strategy\_id, status | 400, 401, 422 |
| 重建索引 | POST | /v1/datasets/{dataset\_id}/rebuild-index | Bearer Token | strategy\_id, mode | task\_id, estimated\_time | 400, 401, 409 |
| 索引状态查询 | GET | /v1/datasets/{dataset\_id}/index-status | Bearer Token | - | status, progress, metrics | 400, 401, 404 |

### 3.4.4 权限与审计子模块

#### 3.4.4.1 职责与边界

**核心职责**：

* 基于角色的知识库访问权限控制
* 关键操作的审计日志记录
* 操作权限的实时验证
* 审计数据的查询和导出

**边界定义**：

* 角色类型：所有者/管理员/编辑/只读
* 权限粒度：知识库级+文档级+分段级
* 审计保留：≥12个月，支持导出
* 访问日志：记录所有API调用和数据访问

#### 3.4.4.2 权限矩阵

| 角色 | 知识库管理 | 文档管理 | 元数据管理 | 索引重建 | 检索测试 | 审计查看 |
| --- | --- | --- | --- | --- | --- | --- |
| 所有者 | 全部权限 | 全部权限 | 全部权限 | 允许 | 允许 | 允许 |
| 管理员 | 查看/编辑 | 全部权限 | 全部权限 | 允许 | 允许 | 允许 |
| 编辑 | 查看 | 增删改查 | 标注编辑 | 受限 | 允许 | 受限 |
| 只读 | 查看 | 查看 | 查看 | 否 | 查看 | 否 |

#### 3.4.4.3 审计日志设计

-- 操作审计日志表  
CREATE TABLE audit\_logs (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID NOT NULL,  
 action VARCHAR(50) NOT NULL,  
 resource\_type VARCHAR(50) NOT NULL,  
 resource\_id UUID,  
 old\_values JSONB,  
 new\_values JSONB,  
 ip\_address INET,  
 user\_agent TEXT,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 访问日志表  
CREATE TABLE access\_logs (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID NOT NULL,  
 endpoint VARCHAR(200) NOT NULL,  
 method VARCHAR(10) NOT NULL,  
 status\_code INTEGER NOT NULL,  
 response\_time\_ms INTEGER,  
 dataset\_id UUID,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP  
);

### 3.4.5 文档与分段维护子模块

#### 3.4.5.1 职责与边界

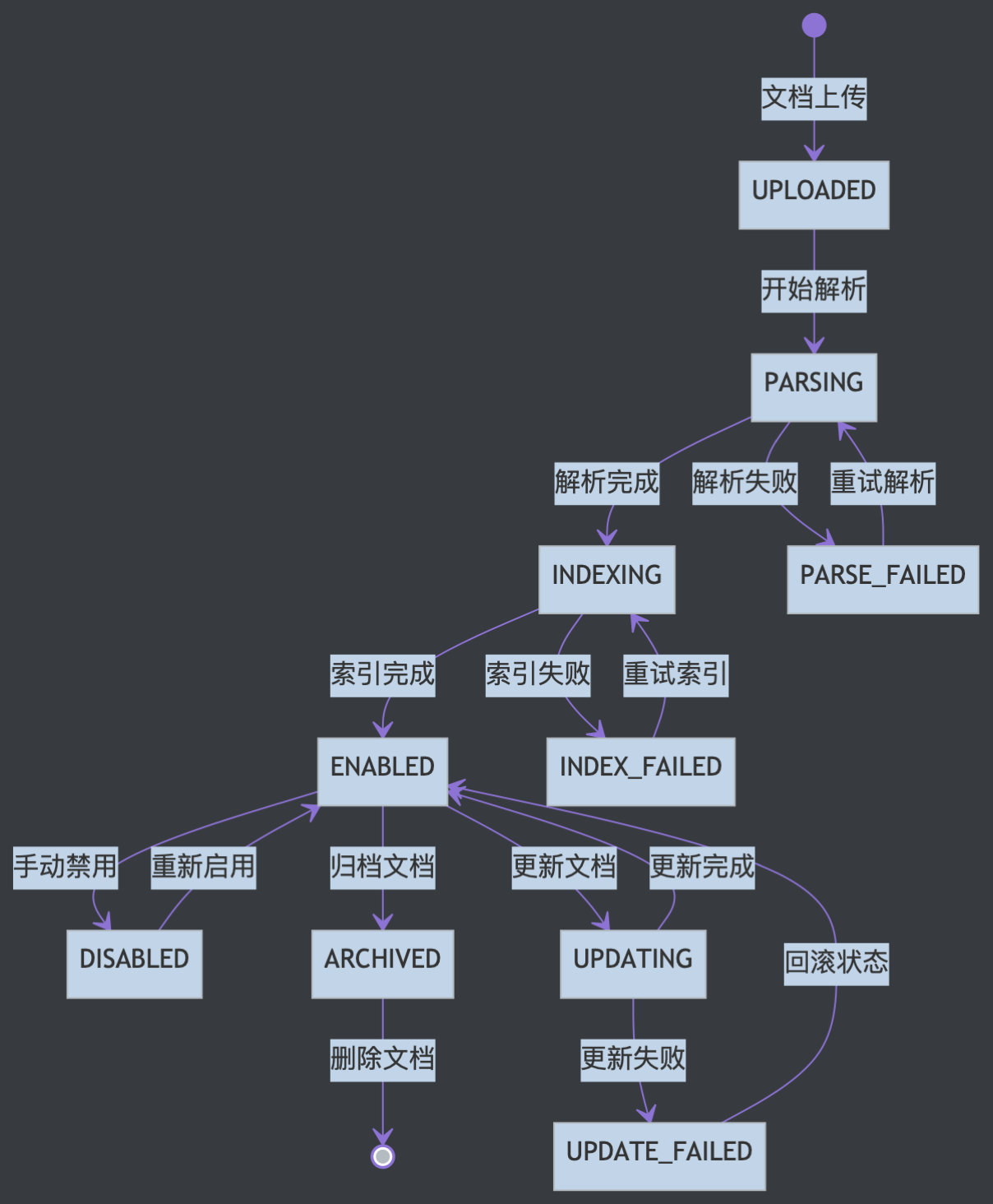
**核心职责**：

* 文档状态的生命周期管理
* 分段的增删改查和批量操作
* 文档更新时的索引重建
* 分段质量的监控和优化

**边界定义**：

* 文档状态：uploaded → parsing → indexing → enabled/disabled/archived
* 分段操作：支持单个和批量操作
* 更新频率：文档更新触发增量索引
* 质量监控：分段相关度、长度分布等指标

#### 3.4.5.2 状态机设计



### 3.5 维修规划及资源智能化模块

#### 3.5.1 职责与边界

**核心职责**：

* 智能化生成和调整维修计划
* 优化航材、人员和工具等资源配置
* 处理工作协同冲突和资源调度
* 提供维修需求分析和计划执行跟踪

**边界定义**：

* 处理维修指令接收和计划生成
* 支持多维度资源配置优化
* 计划执行时间：简单计划≤10秒，复杂计划≤60秒
* 支持1000+并发资源调度请求

**详细设计**：

待补充 - 包含计划生成算法、资源优化策略、冲突解决机制等详细设计

### 3.6 生产看板模块

#### 3.6.1 职责与边界

**核心职责**：

* 提供多维度的生产状态可视化展示
* 实时监控维修计划执行情况
* 展示资源使用状态和工位情况
* 提供预警和告警功能

**边界定义**：

* 支持计划看板、进度看板、资源看板、工位看板
* 数据刷新频率：≤5秒
* 支持大屏显示和移动端适配
* 并发查看用户：≤500人

**详细设计**：

待补充 - 包含看板组件设计、数据聚合策略、实时通信机制等详细设计

### 3.7 发动机检测辅助模块

#### 3.7.1 职责与边界

**核心职责**：

* 智能化检测任务管理和路径规划
* 实时检测结果处理和分析
* 自动化报告生成和历史数据分析
* 异常预警和风险评估

**边界定义**：

* 支持多种检测设备接入
* 检测数据处理：实时流式处理
* 报告生成时间：≤30秒
* 支持历史数据回溯分析

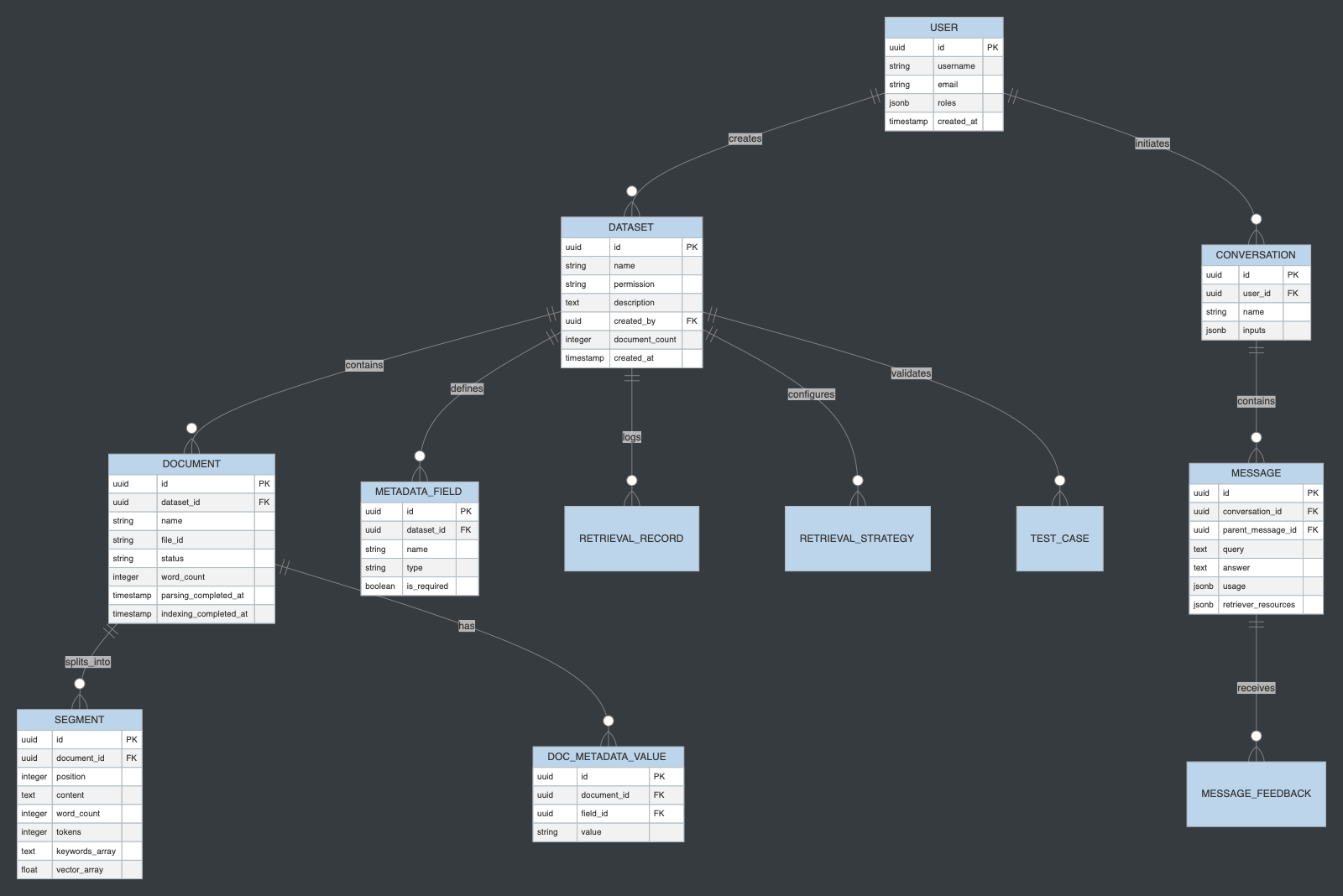
**详细设计**：

待补充 - 包含检测设备集成、数据分析算法、预警规则引擎等详细设计

## 4. 数据设计

### 4.1 逻辑数据模型

整个系统的核心数据实体及其关系如下（当前重点展示维修知识库管理模块，其他模块的数据模型待补充）：



### 4.2 物理数据模型与迁移策略

**数据库选择**：PostgreSQL 15+ with pgvector extension

* 支持关系型数据存储和向量数据索引
* 提供JSONB字段类型支持元数据灵活存储
* 具备完善的事务ACID保证

**分区策略**：

* messages表按创建时间月份分区，提高查询性能
* retrieval\_records表按数据集ID分区，便于数据管理
* segments表考虑按文档数量进行垂直分片

**索引设计**：

-- 组合索引优化查询  
CREATE INDEX idx\_segments\_document\_vector ON segments USING ivfflat(vector);  
CREATE INDEX idx\_messages\_conversation\_time ON messages(conversation\_id, created\_at DESC);  
CREATE INDEX idx\_documents\_dataset\_status ON documents(dataset\_id, status);  
  
-- 元数据查询优化  
CREATE INDEX idx\_doc\_metadata\_value\_field ON doc\_metadata\_value(field\_id, value);  
CREATE INDEX idx\_metadata\_field\_dataset ON metadata\_field(dataset\_id, name);

**迁移策略**：

* 使用Alembic进行数据库版本管理
* 采用蓝绿部署，零停机迁移
* 大表迁移采用分批处理，避免长时间锁表

### 4.3 数据流设计

**主要数据流向**：

1. **知识库构建数据流**：

* 文档上传 → 文件存储 → 解析队列 → 文本提取 → 分段处理 → 向量嵌入 → 索引存储

1. **问答交互数据流**：

* 用户问题 → 意图解析 → 知识检索 → 重排序 → 答案生成 → 引用追踪 → 结果返回

1. **质量监控数据流**：

* 检索日志 → 指标计算 → 质量评估 → 策略调优 → 效果验证 → 配置更新

## 5. 接口设计汇总

### 5.1 REST API统一规范

**版本控制**：

* URL路径版本：/v1/, /v2/
* 向后兼容性：保持旧版本至少6个月
* 废弃通知：通过响应头X-API-Deprecated提前通知

**鉴权机制**：

* Bearer Token认证：Authorization: Bearer <JWT\_TOKEN>
* Token有效期：24小时，支持刷新
* 权限粒度：接口级+资源级双重控制

### 7.1.1 Docker Compose 参考（本地/一体化）

* 最低配置：2C/4GiB（建议8GiB以上），Linux / macOS / Windows(WSL2)
* 组件清单：api、worker、web、rdb、vdb、redis、nginx、sandbox、ssrf\_proxy（按需）
* 启动步骤：复制.env → 调整端口/密钥 → docker compose up -d → 访问 /install 初始化管理账户
* 变量同步：跟随 .env.example 变更更新本地 .env

### 7.1.2 生产集群部署要点

* 拆分职责：API/Worker/Web 独立伸缩；VDB/DB/Redis 独立高可用
* 健康检查：/health, /ready, /health/deep 全覆盖
* 持久化：数据库、向量库、对象存储使用持久卷，定期快照
* 安全：仅开放必要端口；内网访问中间件；密钥集中管理
* 升级：蓝绿或滚动；变更.env同步；向量库兼容性验证

**请求/响应格式**：

// 统一请求格式  
{  
 "data": { /\* 业务数据 \*/ },  
 "meta": { /\* 元数据，如分页信息 \*/ }  
}  
  
// 统一响应格式  
{  
 "code": 0, // 业务状态码，0表示成功  
 "message": "success", // 状态描述  
 "data": { /\* 响应数据 \*/ },  
 "meta": {  
 "request\_id": "req\_123456", // 请求追踪ID  
 "timestamp": "2024-12-19T10:30:00Z",  
 "version": "v1.0"  
 }  
}

**分页规范**：

* 使用limit和offset参数
* 响应包含total, page\_size, page\_number
* 最大每页限制：100条记录

**错误处理**：

* HTTP状态码与业务错误码分离
* 错误响应包含error\_code, error\_detail, suggestions
* 支持国际化错误信息

### 5.2 API接口清单

| 模块 | 接口分组 | 接口数量 | 主要功能 |
| --- | --- | --- | --- |
| 维修知识库管理 | 知识库CRUD | 8 | 创建、查询、更新、删除知识库 |
|  | 文档管理 | 12 | 上传、解析、索引、状态管理 |
|  | 分段操作 | 6 | 分段增删改查、批量操作 |
|  | 元数据管理 | 8 | 字段定义、批量标注、查询 |
|  | 智能问答 | 6 | 问答、会话、历史、反馈 |
|  | 检索服务 | 5 | 向量检索、手册定位、测试 |
|  | 外部知识库 | 4 | 配置、检索、测试、管理 |
|  | 质量监控 | 4 | 测试评估、指标监控、策略配置 |
|  | 权限审计 | 6 | 权限控制、审计日志、访问记录 |
| 维修规划及资源智能化 | 计划管理 | 待补充 | 计划生成、调整、执行跟踪 |
|  | 资源管理 | 待补充 | 航材、人员、工具配置 |
|  | 冲突解决 | 待补充 | 资源冲突检测与解决 |
| 生产看板 | 看板展示 | 待补充 | 计划、进度、资源、工位看板 |
|  | 数据聚合 | 待补充 | 实时数据收集与计算 |
|  | 告警通知 | 待补充 | 异常告警与通知推送 |
| 发动机检测辅助 | 检测管理 | 待补充 | 任务创建、路径规划 |
|  | 数据分析 | 待补充 | 实时分析、历史分析 |
|  | 预警管理 | 待补充 | 异常检测、预警推送 |

### 5.3 限流策略

**API限流规则**：

* 用户级限流：1000 req/hour
* IP级限流：5000 req/hour
* 接口级限流：根据业务重要性分级
* 突发流量：允许短时间超限，后续限制

**限流实现**：

* 使用Redis + Lua脚本实现分布式限流
* 支持滑动窗口和令牌桶算法
* 提供限流状态查询接口

## 6. 非功能需求

### 6.1 可用性/SLA指标

* 系统可用性：99.9%（年计划外停机时间≤8.76小时）
* 服务恢复时间：平均故障恢复时间(MTTR) ≤ 30分钟
* 故障检测时间：平均故障检测时间(MTTD) ≤ 5分钟
* API可用性：核心接口99.95%，非核心接口99.9%

### 6.2 性能指标

**响应时间要求**：

* 知识库列表查询：≤200ms (P95)
* 文档上传接口：≤2s (P95)
* 智能问答接口：≤3s (P95)
* 检索服务接口：≤1s (P95)

**吞吐量要求**：

* 知识库管理：200 QPS
* 智能问答服务：100 QPS
* 检索服务：300 QPS
* 并发用户：支持1000个活跃用户

### 6.2.1 性能基线与容量冗余

* 基线模型：以P50/P95为主，异常值剔除；周更趋势对比
* 容量冗余：按峰值×1.5规划；凌晨离线批量任务限流
* 关键优化点：
  + 向量检索：启用近似索引、分片+副本、压缩量化（如PQ/IVF）
  + 全文检索：冷热分层、前缀索引、字段选择性优化
  + 队列：任务批处理、优先级队列、消费者自动伸缩
  + 缓存：查询结果/embedding缓存，多级缓存穿透保护

### 6.3 扩展性要求

**水平扩展能力**：

* API服务：无状态设计，支持负载均衡
* 数据库：读写分离，支持分片扩展
* 向量库：支持集群模式
* 缓存：Redis集群部署

**数据扩展能力**：

* 支持TB级知识库数据存储
* 支持百万级分段向量索引
* 支持万级并发检索请求

### 6.4 可靠性要求

**数据备份策略**：

* 数据库：每日增量备份，每周全量备份
* 文件存储：异地冗余存储，3副本保证
* 向量数据：定期快照备份
* 配置数据：版本化管理，支持快速回滚

**容灾恢复**：

* RTO（恢复时间目标）：≤4小时
* RPO（恢复点目标）：≤1小时
* 支持跨机房容灾部署
* 关键服务双活架构

### 6.5 安全与隐私要求

**数据安全**：

* 传输加密：TLS 1.3
* 存储加密：AES-256
* 密钥管理：独立密钥管理服务
* 敏感数据脱敏：自动化脱敏规则

**访问安全**：

* 身份认证：JWT + 多因素认证
* 权限控制：RBAC细粒度权限
* 操作审计：完整审计日志
* 防护措施：防SQL注入、XSS、CSRF

### 6.6 可维护性要求

**监控与告警**：

* 系统监控：CPU、内存、磁盘、网络
* 应用监控：响应时间、错误率、吞吐量
* 业务监控：知识库使用量、问答质量
* 告警通知：多渠道告警，升级机制

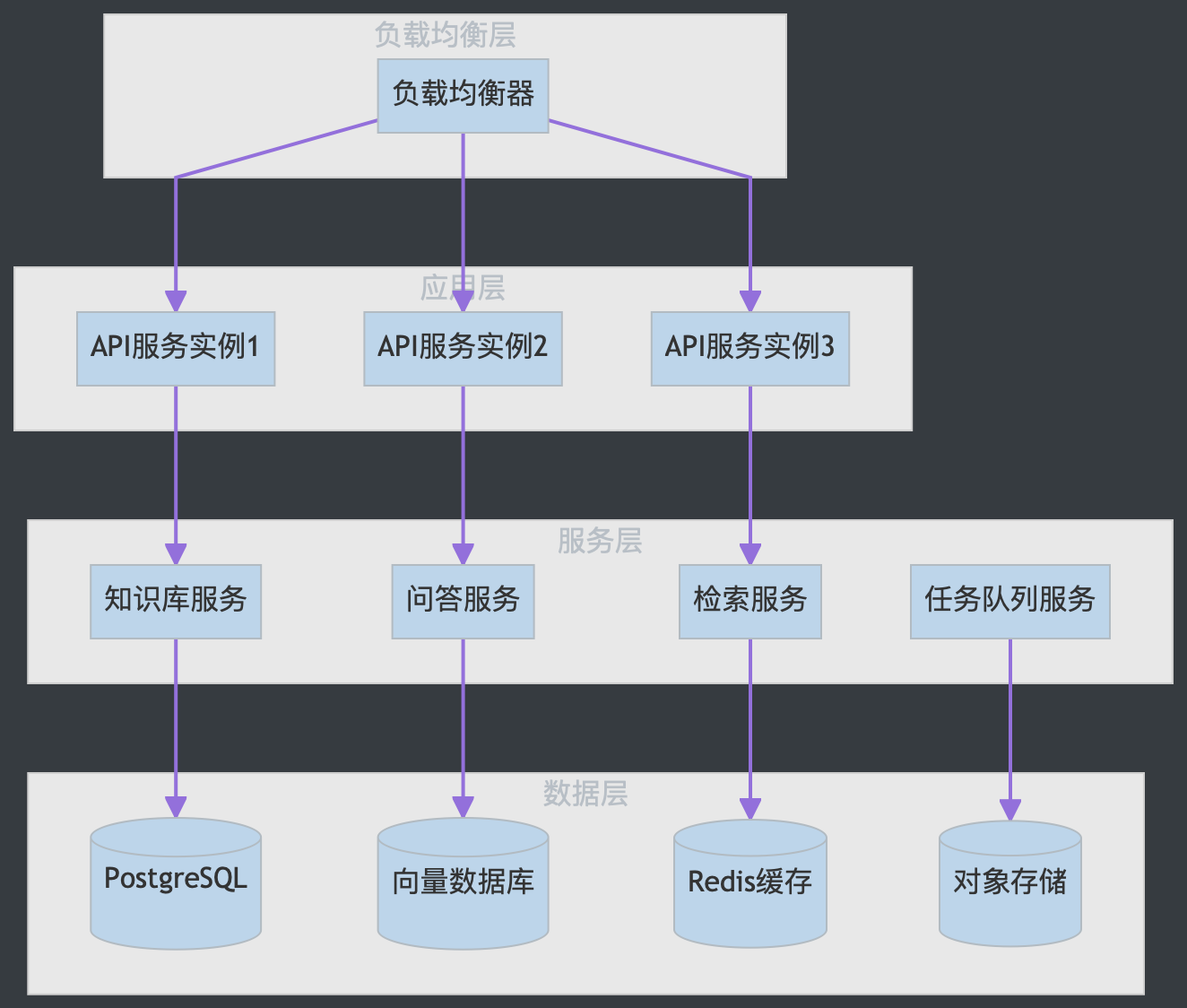
**日志管理**：

* 结构化日志：JSON格式统一输出
* 日志等级：ERROR、WARN、INFO、DEBUG
* 日志聚合：ELK Stack集中管理
* 日志保留：业务日志30天，审计日志1年

## 7. 部署与运维

### 7.1 部署架构

**容器化部署架构**：



**部署环境规划**：

| 环境 | 用途 | 配置 | 数据保留 |
| --- | --- | --- | --- |
| 开发环境 | 功能开发测试 | 2核4GB×3, 单副本 | 7天 |
| 测试环境 | 集成测试验证 | 4核8GB×3, 高可用 | 30天 |
| 预生产环境 | 性能测试验证 | 生产配置×0.5, 高可用 | 90天 |
| 生产环境 | 正式服务 | 8核16GB×5, 三副本 | 永久 |

### 7.2 部署配置

**Docker容器配置**：

# docker-compose.yml  
version: '3.8'  
services:  
 knowledge-api:  
 image: knowledge-base:v1.0  
 ports:  
 - "8000:8000"  
 environment:  
 - DATABASE\_URL=postgresql://user:pass@postgres:5432/knowledge\_db  
 - REDIS\_URL=redis://redis:6379/0  
 - VECTOR\_DB\_URL=http://vectordb:8080  
 healthcheck:  
 test: ["CMD", "curl", "-f", "http://localhost:8000/health"]  
 interval: 30s  
 timeout: 10s  
 retries: 3  
 deploy:  
 replicas: 3  
 resources:  
 limits:  
 memory: 2G  
 cpus: "1.0"  
 reservations:  
 memory: 1G  
 cpus: "0.5"

**Kubernetes部署配置**：

# k8s-deployment.yml  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: knowledge-api  
 labels:  
 app: knowledge-api  
spec:  
 replicas: 3  
 selector:  
 matchLabels:  
 app: knowledge-api  
 template:  
 metadata:  
 labels:  
 app: knowledge-api  
 spec:  
 containers:  
 - name: api  
 image: knowledge-base:v1.0  
 ports:  
 - containerPort: 8000  
 env:  
 - name: DATABASE\_URL  
 valueFrom:  
 secretKeyRef:  
 name: db-secret  
 key: database-url  
 resources:  
 requests:  
 memory: "1Gi"  
 cpu: "500m"  
 limits:  
 memory: "2Gi"  
 cpu: "1000m"  
 livenessProbe:  
 httpGet:  
 path: /health  
 port: 8000  
 initialDelaySeconds: 30  
 periodSeconds: 10  
 readinessProbe:  
 httpGet:  
 path: /ready  
 port: 8000  
 initialDelaySeconds: 5  
 periodSeconds: 5

### 7.3 运维监控

**健康检查**：

* 应用健康检查：/health 端点检查服务状态
* 就绪检查：/ready 端点检查依赖服务连通性
* 深度检查：/health/deep 端点检查业务功能可用性

**监控指标**：

# prometheus配置  
global:  
 scrape\_interval: 15s  
  
scrape\_configs:  
 - job\_name: 'knowledge-api'  
 static\_configs:  
 - targets: ['knowledge-api:8000']  
 metrics\_path: /metrics  
 scrape\_interval: 10s  
  
 - job\_name: 'postgres'  
 static\_configs:  
 - targets: ['postgres-exporter:9187']  
  
 - job\_name: 'redis'  
 static\_configs:  
 - targets: ['redis-exporter:9121']

**日志收集**：

# filebeat配置  
filebeat.inputs:  
- type: container  
 paths:  
 - /var/lib/docker/containers/\*/\*.log  
 processors:  
 - add\_docker\_metadata:  
 host: "unix:///var/run/docker.sock"  
 - decode\_json\_fields:  
 fields: ["message"]  
 target: ""  
 overwrite\_keys: true  
  
output.elasticsearch:  
 hosts: ["elasticsearch:9200"]  
 index: "knowledge-base-logs-%{+yyyy.MM.dd}"

### 7.4 备份与恢复

**数据备份策略**：

#!/bin/bash  
# 数据库备份脚本  
pg\_dump -h postgres -U knowledge\_user -d knowledge\_db \  
 --no-password --clean --if-exists \  
 --format=custom \  
 --file=/backup/knowledge\_db\_$(date +%Y%m%d\_%H%M%S).dump  
  
# 向量数据备份  
curl -X POST "http://vectordb:8080/collections/knowledge/snapshots" \  
 -H "Content-Type: application/json" \  
 -d '{"snapshot\_name": "daily\_'$(date +%Y%m%d)'"}'  
  
# 文件备份到对象存储  
aws s3 sync /data/files s3://knowledge-backup/files/$(date +%Y%m%d)/

**恢复流程**：

1. 确认备份文件完整性
2. 停止应用服务
3. 恢复数据库：pg\_restore -d knowledge\_db backup.dump
4. 恢复向量数据：从快照恢复向量索引
5. 恢复文件数据：从对象存储下载文件
6. 启动应用服务并验证

## 8. 测试设计

### 8.1 测试覆盖范围

**功能测试**：

* 知识库CRUD操作完整性测试
* 文档上传、解析、索引流程测试
* 智能问答准确性与响应时间测试
* 检索功能精确度与召回率测试

**性能测试**：

* 并发用户访问压力测试
* 大文件上传性能测试
* 海量数据检索性能测试
* 系统资源使用率测试

**安全测试**：

* 身份认证与授权测试
* API安全漏洞扫描
* 数据传输加密验证
* 敏感信息泄露检测

### 8.2 关键测试用例

| 用例类别 | 用例名称 | 前置条件 | 测试步骤 | 期望结果 | 自动化 |
| --- | --- | --- | --- | --- | --- |
| 知识库管理 | 创建知识库 | 用户已登录 | 1.填写名称描述 2.选择权限 3.提交创建 | 知识库创建成功，返回ID | 是 |
| 文档处理 | 大文件上传 | 准备100MB PDF文件 | 1.选择文件 2.上传 3.等待解析 | 30秒内解析完成 | 是 |
| 智能问答 | 维修问题问答 | 知识库已建立 | 1.输入"发动机故障" 2.等待回答 | 3秒内返回相关答案 | 是 |
| 检索服务 | 手册定位 | 手册已上传索引 | 1.输入定位查询 2.获取结果 | 1秒内返回章节位置 | 是 |
| 性能测试 | 并发问答 | 100个虚拟用户 | 1.同时发起问答 2.记录响应时间 | 95%请求<5秒 | 是 |

### 8.3 自动化测试

**单元测试**：

# 示例单元测试  
import pytest  
from knowledge\_service import DatasetService  
  
class TestDatasetService:  
 def test\_create\_dataset(self):  
 service = DatasetService()  
 dataset = service.create\_dataset(  
 name="测试知识库",  
 permission="only\_me",  
 description="单元测试用知识库"  
 )  
 assert dataset.id is not None  
 assert dataset.name == "测试知识库"  
  
 def test\_upload\_document(self):  
 service = DatasetService()  
 dataset = service.create\_dataset("测试库", "only\_me")  
  
 with open("test.pdf", "rb") as f:  
 document = service.upload\_document(  
 dataset\_id=dataset.id,  
 file=f,  
 metadata={"type": "manual"}  
 )  
 assert document.status == "uploaded"

**集成测试**：

# API集成测试  
import requests  
  
def test\_qa\_flow():  
 # 创建知识库  
 response = requests.post("/v1/datasets", json={  
 "name": "集成测试库",  
 "permission": "only\_me"  
 })  
 dataset\_id = response.json()["data"]["id"]  
  
 # 上传文档  
 with open("manual.pdf", "rb") as f:  
 files = {"file": f}  
 response = requests.post(  
 f"/v1/datasets/{dataset\_id}/document/create-by-file",  
 files=files  
 )  
 assert response.status\_code == 200  
  
 # 等待索引完成  
 time.sleep(30)  
  
 # 测试问答  
 response = requests.post("/v1/chat-messages", json={  
 "query": "如何启动发动机？",  
 "inputs": {"device\_model": "X100"}  
 })  
 assert response.status\_code == 200  
 assert "启动" in response.json()["data"]["answer"]

## 9. 风险与问题

### 9.1 技术风险

| 风险项 | 风险等级 | 影响范围 | 缓解策略 | 应急预案 |
| --- | --- | --- | --- | --- |
| 向量数据库性能瓶颈 | 高 | 检索服务 | 分片部署，读写分离 | 降级为关键词检索 |
| 大语言模型服务中断 | 中 | 问答功能 | 多供应商备用 | 切换备用模型 |
| 文档解析失败率高 | 中 | 知识库构建 | 多解析器并行 | 人工处理失败文档 |
| 数据库连接池耗尽 | 中 | 整体服务 | 连接池监控 | 重启服务释放连接 |

### 9.2 业务风险

| 风险项 | 风险等级 | 影响范围 | 缓解策略 | 应急预案 |
| --- | --- | --- | --- | --- |
| 问答准确率不达标 | 高 | 用户体验 | 持续优化模型 | 人工客服兜底 |
| 知识库数据泄露 | 高 | 安全合规 | 加密存储传输 | 立即隔离受影响系统 |
| 用户采纳率低 | 中 | 项目价值 | 用户培训推广 | 调整产品策略 |
| 维修手册版本混乱 | 中 | 数据质量 | 版本管理流程 | 临时锁定问题版本 |

### 9.3 开放问题与待确认项

**待确认问题**：

1. 大语言模型的具体选型和配置参数
2. 向量数据库的分片策略和副本数量
3. 文档解析服务的具体实现方案
4. 与SAP系统的集成接口规范

**待补充项**：

1. 详细的错误码定义和国际化方案
2. 生产环境的灾备切换流程
3. 数据迁移工具的开发计划
4. 用户权限模型的细化设计

## 10. 变更记录

| 版本 | 日期 | 变更内容 | 作者 |
| --- | --- | --- | --- |
| v1.0 | 2025-09-01 | 初版发布，完成整体系统架构和维修知识库管理模块详细设计 | 系统架构师团队 |
|  |  | 包含四大功能模块总体设计，维修知识库管理模块详细实现 |  |
|  |  | 定义了数据模型、接口规范、部署方案等技术基础架构 |  |
|  |  | 为其他三个模块预留了详细设计占位符，待后续补充完善 |  |

## 附录

### 附录A. 术语表

| 术语 | 英文 | 定义 |
| --- | --- | --- |
| 检索增强生成 | RAG (Retrieval-Augmented Generation) | 结合检索和生成的AI技术，通过检索相关知识来增强答案生成质量 |
| 向量嵌入 | Vector Embedding | 将文本转换为高维向量表示，用于语义相似度计算 |
| 分段 | Segment | 将长文档切分为的较小文本块，便于检索和引用 |
| 重排序 | Reranking | 对初步检索结果进行二次排序，提升相关性 |

### 附录B. 参考链接

1. [PostgreSQL官方文档](https://www.postgresql.org/docs/)
2. [FastAPI框架文档](https://fastapi.tiangolo.com/)
3. [Docker容器化指南](https://docs.docker.com/)
4. [Kubernetes部署文档](https://kubernetes.io/docs/)
5. [Prometheus监控指南](https://prometheus.io/docs/)

### 附录C. 关联代码/配置路径

基于项目代码结构，主要模块对应的实现路径：

**维修知识库管理模块**：

* 知识库管理服务：api/services/dataset\_service.py
* 文档处理服务：api/services/file\_service.py
* 检索服务：api/core/rag/retrieval/, api/core/rag/datasource/retrieval\_service.py
* 问答服务：api/services/app\_generate\_service.py
* 向量存储服务：api/services/vector\_service.py
* 元数据服务：api/services/metadata\_service.py
* 外部知识服务：api/services/external\_knowledge\_service.py
* 知识实体：api/services/entities/knowledge\_entities/
* 检索测试服务：api/services/hit\_testing\_service.py
* 数据模型：api/models/dataset.py
* API控制器：api/controllers/console/datasets/, api/controllers/service\_api/dataset/
* 任务队列：api/tasks/add\_document\_to\_index\_task.py, api/tasks/document\_indexing\_task.py
* RAG核心：api/core/rag/embedding/, api/core/rag/extractor/, api/core/rag/index\_processor/

**维修规划及资源智能化模块**：

* 待补充：计划生成服务、资源管理服务、调度算法等实现路径

**生产看板模块**：

* 待补充：看板服务、数据聚合服务、实时通信等实现路径

**发动机检测辅助模块**：

* 待补充：检测服务、分析引擎、预警服务等实现路径