财顾报告生成智能体：后端（Java）及算法端（Python）功能模块描述与实现思路

1. 文档范围与分工边界

侧	核心职责
Java 业务后端	**任务与状态的真相源**；鉴权与租户；人工卡点持久化；行内集成与 OSS；编排何时调用算法端；结果落库与导出。
Python 算法 / Agent 服务	**无状态或弱状态**的「智能计算」：**多段 LLM、结构化大纲、RAG、提示词组装与校验**；不替代 Java 做过账式持久化主流程。
运行支撑	关系库、缓存、MQ、对象存储、模型集群；为 Java / Python 共用，但**事务边界与审计主记录在 Java**。

下文按架构图中的 **Java 子图**、**Python 子图**、**运行支撑** 分别展开。

2. Java 业务后端（Spring Boot）各模块

2.1 GW｜接入：鉴权、审计、租户、限流

要实现的功能

- **鉴权与身份**：对接行内 SSO / JWT / 网关透传头，识别用户与组织机构。
- **租户上下文**：解析并注入 `tenant_id` / `org_id`（或默认租户），贯穿后续 ORCH、REPO、ADAPTER 调用链。
- **审计**：记录关键 API 的访问主体、资源、时间、结果；满足后续合规扩展。
- **限流与防护**：接口级 / 租户级限流，防止任务创建洪峰或暴力重试打穿后端与算法端。

实现思路（概要）

- Spring Security + Filter 链：认证完成后将 `TenantContext` 、 `UserPrincipal` 放入 **SecurityContext** 或 **请求级 ThreadLocal**，并由 WebMvc 拦截器向下传递。
- 限流：网关（若有）+ 应用内 **Redis 令牌桶 / 滑动窗口**；与租户配额策略表或配置中心联动。
- 审计：AOP 切面 + 异步写审计表；TraceId 与日志 MDC 对齐。

2.2 ORCH｜任务状态机：大纲 → 数据 → 成文 → 完成

要实现的功能

- 维护报告任务全生命周期：与需求文档一致的状态（如：待生成大纲、待确认大纲、待准备数据、待确认数据、待生成报告、生成中、待查看结果、完成、失败等）。
- **驱动阶段迁移**：在满足前置条件（含人工确认落库）后，触发下一阶段；失败时进入可恢复或失败终态。
- **与异步解耦**：长耗时阶段（整段大纲生成、全文段落批量）可 **投递 MQ**，由 Worker 拉活后继续调 ORCH 或直连 ADAPTER/SVC。
- **编排 Python 调用时机**：仅在允许的阶段调用 ADAPTER；对 Python 返回做**业务校验**后写 REPO。

实现思路（概要）

- **枚举 + 显式转移表** 或 **Spring State Machine**：转移函数内只做「校验 → 调用领域服务 → 持久化状态」；避免在 Controller 写长流程。
- **应用服务分层**（参考 `finrep-application/orchestration` ）： `command` 表示写操作、与事务边界对齐；幂等键防重复提交。
- MQ：消息体带 `taskId` 、 `stage` 、 `tenantId` ；Consumer **幂等消费**（基于 DB 状态或 Redis 去重）。

2.3 HITL｜人工卡点持久化：大纲确认 / 数据确认

要实现的功能

- **大纲确认**：持久化业务人员在**预置知识体系范围内**的勾选、取消、顺序调整结果，形成**最终知识单元清单**版本，供后续数据准备与成文唯一依据。
- **数据确认**：持久化每知识单元的「系统自动取数结果 + 文本补录」合并后的**数据包**及确认状态。
- **版本与可追溯**：可选记录确认人、确认时间、与模型草稿的差异标记，便于审计与问题复盘。

实现思路（概要）

- 独立聚合表：任务 ID + 阶段 + 版本号；确认提交为**一次事务**：更新清单/数据包 + 推进 ORCH 状态。
- 服务端强校验：禁止未配置知识单元、禁止越权改写结构树；与领域文档「受控生成」一致。
- 不将「确认权」下放 Python；Python 只消费**已确认**快照或 Java 明确下发的只读 DTO。

2.4 SVC｜取数编排、OSS、导出、行内 HTTP（工商/指标等）

要实现的功能

- **取数编排**：按「知识单元 × 数据绑定」拉起 HTTP 调用、指标平台查询；将异构结果规整为可展示、可合并的字段（供前端与 Prompt 占位符）。
- **OSS**：上传材料存储、预签名下载；与任务、租户路径策略绑定。
- **行内 HTTP**：工商、风险、其他资质类接口；统一超时、重试、熔断、证书。
- **导出**：将已定稿报告生成 DOCX/PDF（或异步任务）；敏感留痕与文件留存策略由行内规范决定。

实现思路（概要）

- **端口-适配器**：application 定义 `DataSourcePort`，infrastructure 里按类型实现（REST、指标 SDK、Mock）。
- 连接池与隔离：按下游系统分 **RestTemplate/WebClient Bean**，避免单点超时拖垮全站。
- 导出：模板 + **docx4j/POI**；大文件走异步 + 任务状态回写。

2.5 REPO｜任务 / 大纲 / 数据包 / 报告落库

要实现的功能

- 持久化任务主档、一级/二级大纲结果、最终知识单元清单、各单元数据包、单段生成结果、完整报告正文与元数据。
- 支持按 **租户 + 任务 ID** 查询与列表分页；关键中间结果可 JSON 列存储并可追溯。
- 与 ORCH 状态**原子一致**：状态推进与结果写入同事务或**外显状态机 + Outbox**（若 MQ 与库强一致）。

实现思路（概要）

- MyBatis / JPA + **Flyway/Liquibase**；热点表预留归档策略。
- 大文本：可考虑与 OSS 协同（正文过大时），MVP 以库内 CLOB/JSON 为主亦可。
- **租户列 + 索引**：所有业务表强制 `tenant_id`，避免交叉租户访问。

2.6 ADAPTER｜Agent 适配层：HTTP/gRPC 调用 Python

要实现的功能

- 将 Java 内部的「阶段请求」转换为 Python 算法服务约定的 **OpenAPI / gRPC** 请求。
- 注入 **服务间认证**、**TraceId**、**TenantId**、**TaskId**、**幂等键**；处理超时、重试、熔断与**可映射错误码**。
- 对 Python 返回的 JSON / 二进制做**反序列化与契约校验**（字段缺失、类型不符打业务可理解错误）。

实现思路（概要）

- **OpenFeign / WebClient** + Resilience4j；超时：大纲可能 30–120s，段落可更短，分级配置。
- DTO 与 `finrep-contract/openapi` 对齐；可选代码生成减少手写漂移。
- 严禁前端直连 Python；所有流量经 Java，便于审计与脱敏。

3. Python 算法 / Agent 服务各模块

3.1 AG｜报告 Agent：阶段 Skills（L1 / L2 / 段落）

要实现的功能

- **一级大纲（L1）**：基于任务上下文与一级章节候选，输出**结构化 JSON**（章节是否呈现、段落规模枚举、整体撰写逻辑说明等），与需求文档 Prompt 约束一致。
- **二级大纲（L2）**：按一级章节循环或批处理，输出**末级知识单元结构**（章节编号树、纳入节点、结构逻辑说明）；供 Java 转 UI 树并人工确认。
- **段落生成（Section）**：按单个知识单元：拼装 system/user messages、注入数据包与模板逻辑，调用 MW，返回**单段正文**；可多次调用由 Java 拼接顺序。
- **可选**：LangGraph 将上述 Skills 连成内部子图；**卡点仍以 Java 状态为准**，Python 侧以「无状态请求」为主。

实现思路（概要）

- FastAPI **Router 分路由**：`/v1/outline/l1` 、 `/v1/outline/l2` 、 `/v1/section` 等；Handler 薄，**Skills 包**内写核心逻辑。
- 每个 Skill：**构建 Prompt**（调用 PROMPT）→ **调 MW** → **解析与校验**（Pydantic + JSON Schema）→ 返回 DTO。

- 超长上下文：按阶段拆分调用，不在单次请求堆砌全篇材料；与 Java「分段生成」分工一致。

3.2 RAG｜解析、分块、检索、拼装

要实现的功能

- **解析**：对上传材料（PDF/Word 等）提取文本（可异步 ingestion）；遵守行内病毒扫描与格式白名单。
- **分块**：按章节或固定窗口 + overlap 切分；可带 metadata（文件名、页码）。
- **检索**：向量检索 +（可选）关键词；返回 Top-K 片段。
- **拼装**：将片段整理为「可注入 Prompt 的引用块」或结构化 `citations`，避免模型胡编无据（策略按产品定）。

实现思路（概要）

- Embedding 与向量库：**pgvector / Milvus**；索引可按 **task_id** 隔离集合，降低 MVP 复杂度。
- **ingestion** 与 **query** 路径分离：Java 材料落 OSS 后，传 **只读 URL 或 objectId** 给 Python；Python 拉取后解析入索引。
- 与 `skills/rag_retrieve` 配合：返回检索结果 DTO，由 AG 或 SVC 决定谁来拼进最终 Prompt。

3.3 PROMPT｜Jinja / 模板、JSON Schema 校验

要实现的功能

- 管理 **Jinja2 / 字符串模板**：变量来自任务字段、大纲阶段输出、数据包、RAG 引用。
- **结构校验**：大纲类输出走 **JSON Schema**；解析失败触发重试或带原因返回 Java（便于产品提示「请重试」）。
- **版本化**：模板带版本号；请求可带 `prompt_version` 与 Java 记录一致，便于回归与排障。

实现思路（概要）

- `prompts/templates` 存放文件；`builders` 用代码组装多段 message（system/developer/user）。
- **校验库**：`jsonschema` 或 Pydantic v2；对模型输出先 `json.loads` 再 validate。
- 与需求文档附录 Prompt **逐条对齐**，变更走 Code Review + 版本表。

3.4 MW｜模型网关：OpenAI 兼容、多模型路由、重试降级

要实现的功能

- 统一通过 **OpenAI 兼容 HTTP** 访问行内模型集群；屏蔽路径差异（chat/completions 等以实际网关为准）。
- **多模型路由**：按请求元数据（阶段 L1/L2/section、租户默认、成本档位、 `logical_model` ）选择 **base_url + model 名**。
- **重试与降级**：429/5xx 时退避；主模型失败切换备用；**不向调用方暴露密钥**。
- **可观测**：记录延迟、token（若返回）、route_id；与 TraceId 关联。

实现思路（概要）

- 薄封装 `httpx` / 官方 OpenAI SDK（兼容模式）；**Router** 读配置表或 YAML。
- **Semaphore / 本地并发上限** 保护单机；全局限流与租户配额可与 Java/Redis 协调（双端取严）。
- 结构化输出：优先 `response_format` /约束 Prompt；仍要做服务端 Schema 校验。

4. 运行支撑（架构图中 RUN）与两端关系

4.1 关系库 DB

功能：持久化任务与全流程中间结果（主权威在 Java）。
思路：Python **不写主事务库**（除非只读分析库或独立向量元数据表）；避免双写一致性问题。

4.2 Redis（锁、配额、进度缓存）

功能：分布式锁（同任务阶段防重入）、租户/API 限流计数、热点任务进度缓存。
思路：Java 为主写方；Python 如需限流可读共享 Redis 配置（可选）。

4.3 MQ（削峰）

功能：长耗时「生成大纲 / 生成全文」**异步化**，缓和 LLM 突增压。
思路：**建议由 Java 生产与消费**，在 Consumer 中调 ADAPTER→Python，状态仍在 Java 提交；避免 Python Consumer 与 Java 状态机**双头驾驶**。

4.4 OSS

功能：材料与导出文件；租户/任务维度路径。
思路：Java 管上传凭证与元数据；Python **仅按需拉取** Java 颁发的只读链接。

4.5 OpenAI 兼容模型集群 LLM

功能：实际推理算力。
思路：**仅由 Python MW 直连**（或经行内统一推理网关）；Java 不重复直连模型，避免双套路由逻辑。

5. 端到端协作要点

1. **状态与持久化以 Java 为准**；Python 输出一律经 Java 校验后再落库。
2. **人工卡点（大纲/数据）只在 Java + 前端闭环**；Python 不拥有「确认后」的写权限。
3. **ADAPTER 是集成关键**：契约稳定、超时合理、可观测一致，比争论「Agent 在不在 Java」更能决定上线质量。
4. **MW 与 PROMPT 是质量与成本枢纽**：多模型路由 + 强 Schema + 模板版本化，是控制「可控生成」的技术抓手。
5. **RAG 与 SVC 取数分工**：结构化指标走 Java 行内接口；非结构化长文档走 Python RAG；避免重复造轮子或双源冲突。