# 环境与项目初始化（后端 + Flutter）

本项目最终形态为 Android/iOS App。后端技术选型以 Python + FastAPI 为主，队列用 Celery（Redis/RabbitMQ），HTTP 调用用 httpx，重试用 tenacity/backoff；不引入 LangChain，先以“轻封装 + 明确契约”实现可控、可测试的最小可用链路。

## 技术选型与原则

* 后端：Python 3.11 + FastAPI + Uvicorn/Gunicorn
* 任务队列：Celery + Redis/RabbitMQ（解析/生成/评测等耗时任务）
* 网络/重试：httpx + tenacity/backoff（超时、重试、退避、速率限制）
* 数据校验：Pydantic（结构化出入参）
* 向量库（起步→进阶）：pgvector → Qdrant Cloud（或 MongoDB Atlas Vector）
* 模型调用：优先托管 API（OpenAI/Anthropic/Gemini/国内可替换）；嵌入用 text-embedding-3-large/-small
* 抽象层：自定义 LLMProvider（complete/chat/embed）与 VectorStoreRepository，保留后续平滑迁移空间
* 不使用：LangChain（当前阶段），等 RAG/工具链复杂后再择机按需引入模块

## 后端初始化（Python + FastAPI）

* 依赖安装（示例）：

python -m venv .venv && source .venv/bin/activate
pip install --upgrade pip
pip install fastapi uvicorn[standard] pydantic-settings httpx tenacity backoff
pip install celery[redis] redis # 或 rabbitmq: pip install celery
pip install orjson python-dotenv
pip install pytest pytest-asyncio httpx[http2]
# 可选观测：
pip install opentelemetry-sdk opentelemetry-instrumentation-fastapi opentelemetry-exporter-otlp

* 目录结构建议：

server/
app/
main.py # FastAPI 入口
api/ # 路由层（ingest/personalize/materials/...）
services/
llm\_provider.py # LLMProvider 接口与多厂商实现
embedder.py # 嵌入封装
rag.py # 可选 RAG 检索
tasks/
worker.py # Celery 实例
ingest\_pdf.py # 解析任务
personalize.py # 个性化改写任务
materials.py # 素材生成任务
scoring.py # 评测任务
models/ # Pydantic 模型
repos/ # VectorStoreRepository/DB 访问
config.py # 设置/环境变量
tests/

## 环境变量（.env.example）

# LLM 相关
LLM\_PROVIDER=openai # openai|anthropic|google|qwen ...
LLM\_MODEL=gpt-4o-mini # 主模型
EMBEDDING\_MODEL=text-embedding-3-large
OPENAI\_API\_KEY=
ANTHROPIC\_API\_KEY=
GOOGLE\_API\_KEY=
QWEN\_API\_KEY=
# 队列与存储
REDIS\_URL=redis://localhost:6379/0 # 或 RABBITMQ\_URL=amqp://...
DB\_URL=postgresql://user:pass@host:5432/db
VECTOR\_DB=pgvector # pgvector|qdrant|mongodb
PG\_DSN=postgresql://user:pass@host:5432/db
QDRANT\_URL=
QDRANT\_API\_KEY=
MONGODB\_URI=
MONGODB\_DB=
# 其他
MODEL\_SERVE\_MODE=api # api|local，已确定使用 api
RATE\_LIMIT\_RPS=1
LOG\_LEVEL=INFO

## 运行与测试（后端）

# 开发启动
uvicorn app.main:app --reload
# 启动 Celery worker
celery -A app.tasks.worker worker -l info
# 测试
pytest -q

## Flutter 初始化（客户端）

* 提示词：
* “使用 flutter create app 初始化；引入 flutter\_riverpod、dio、freezed、json\_serializable、go\_router、hive/sqflite、mocktail、flutter\_test。”
* 命令示例：

flutter --version | cat
flutter create learn\_your\_way\_app
cd learn\_your\_way\_app
flutter pub add flutter\_riverpod dio go\_router freezed\_annotation json\_annotation
flutter pub add --dev build\_runner freezed json\_serializable mocktail flutter\_lints
flutter pub add hive hive\_flutter # 或 sqflite

* 基本验证：

flutter analyze | cat
flutter test | cat
flutter run -d ios # 或 android/macos

## 持续集成（要点）

* 客户端：flutter analyze、flutter test、golden/集成测试
* 后端：pytest + 覆盖率门槛；构建镜像并部署测试环境
* 公共：固定评测基准（个性化/测验/导图）与分数阈值，作为质量门禁