서비스 설정 및 사용 가이드

이 문서는 서비스를 처음 설정하고 실행하는 사용자를 위한 가이드입니다.

1. 사전 준비 사항

- **Python 3.11** 이상이 설치되어 있어야 합니다.
- **Docker** (선택 사항: 컨테이너 환경에서 실행할 경우)

2. 설치 및 환경 설정

터미널을 열고 다음 순서대로 명령어를 입력하세요.

2.1. 디렉토리 이동

먼저 `RAG` 디렉토리로 이동합니다.

```
cd src/RAG
```

2.2. 가상환경 생성 및 활성화

Python 패키지 충돌을 방지하기 위해 가상환경을 사용하는 것이 좋습니다.

```
# 가상환경 생성 (venv 이름은 자유롭게 변경 가능, 예: .venv)
python -m venv .venv

# 가상환경 활성화
# Windows:
.venv\Scripts\activate

# Mac/Linux:
source .venv/bin/activate
```

2.3. 의존성 패키지 설치

필요한 라이브러리를 설치합니다.

```
pip install -r requirements.txt
```

2.4. 환경 변수 설정 (.env)

`RAG` 디렉토리 안에 `.env` 파일을 생성하고 다음 내용을 작성해야 합니다. `OPENAI_API_KEY`는 필수입니다.

- **`.env` 파일 예시:**

```
OPENAI_API_KEY=sk-proj-xxxxxxxxxxxxxxxxxxxxxxxx
REDIS_URL=redis://localhost:6379/0
```

> **참고:** 로컬에서 Redis 없이 단순 테스트만 하려는 경우, 코드 상에서 Redis 연결 실패 시 어떻게 처리되는지 확인이 필요할 수 있습니다. 기본 설정은 Docker 내부 네트워크용 주소로 되어 있으므로 로컬 실행 시 `REDIS_URL`을 수정해야 합니다.

3. 서비스 실행

3.1. 로컬에서 실행 (Uvicorn)

개발 모드로 서버를 실행합니다.

```
uvicorn api.rag_service:app --host 0.0.0.0 --port 8000 --reload
```

실행 후 `http://localhost:8000/docs` 로 접속하면 API 문서를 확인할 수 있습니다.

3.2. Docker로 실행

Docker를 사용하여 이미지를 빌드하고 실행할 수도 있습니다.

```
# 이미지 빌드
docker build -t rag-service .

# 컨테이너 실행 (포트 8000번)
docker run -p 8000:8000 --env-file .env rag-service
```

4. 유지보수 및 데이터 관리

데이터 인덱싱 및 공지사항 업데이트를 위한 스크립트가 `scripts/` 폴더에 준비되어 있습니다.

- **주의:** 스크립트 실행 전 반드시 가상환경이 활성화되어 있어야 하며, `RAG/` 디렉토리(루트)에서 실행해야 경로 문제가 발생하지 않습니다.

4.1. 전체 인덱스 빌드

CSV 데이터(`data/` 폴더 내)를 기반으로 ChromaDB 인덱스를 처음부터 생성하거나 재구축할 때 사용합니다.

```
python scripts/build_indices.py
```

- `data/` 폴더에 있는 `notices`, `rules`, `schedule`, `courses` 관련 CSV 파일들을 읽어 임베딩하고 벡터 DB에 저장합니다.

4.2. 공지사항 업데이트

동국대학교 공지사항 웹사이트를 크롤링하여 최신 공지를 가져오고, 인덱스에 추가합니다.

- **기본 실행 (최신 5페이지만 확인):**

```
python scripts/update_notices.py
```

- **옵션 설명:**
- `--max-pages N`: 각 게시판별로 최신 N페이지만 크롤링 (기본값: 5)

- `--full`: 모든 페이지 크롤링
- `--boards BOARD_NAME`: 특정 게시판만 크롤링
- `--interval N`: N분마다 반복 실행
- **예시:**

```
# 30분마다 반복해서 공지사항 업데이트
python scripts/update_notices.py --interval 30

# 모든 게시판의 전체 페이지 긁어오기 (초기 구축 시)
python scripts/update_notices.py --full
```

5. 문제 해결 (Troubleshooting)

- **ModuleNotFoundError**: `PYTHONPATH` 문제일 수 있습니다. 스크립트 실행 시 반드시 `RAG/` 폴더 안에서 `python scripts/...` 형태로 실행하세요.
- **Redis Connection Error**: `.env` 파일의 `REDIS_URL`이 올바른지 확인하세요. 로컬에서 실행 중이라면 로컬 Redis 서버가 켜져 있어야 합니다.

6. 전체 백엔드 통합 실행 (RenuxServer + RAG + DB)

`src/RenuxServer` 디렉토리에는 메인 백엔드 서버(ASP.NET Core), 데이터베이스(PostgreSQL), Redis, 그리고 RAG 서비스를 통합하여 실행할 수 있는 `docker-compose.yml` 파일이 포함되어 있습니다.

6.1. 디렉토리 이동

`src/RenuxServer` 디렉토리로 이동합니다. (현재 위치가 `src/RAG`라면 `cd ../RenuxServer`)

```
cd ../RenuxServer
```

6.2. 환경 변수 설정 (.env)

`src/RenuxServer` 디렉토리 내에 `.env` 파일을 생성하고 다음 변수들을 설정해야 합니다.

- **`.env` 파일 예시:**

```
CONNECTIONSTRING="Host=db; Port=5432; Username=postgres; Passw
ord=password; Database=renux_db"
```

```
POSTGRES_USER=postgres
POSTGRES_PASSWORD=password
POSTGRES_DB=renux_db

JWT_KEY={JWT 인증키}

RAG_BASE_URL=http://host.docker.internal:8000
OPENAI_API_KEY=sk-proj-xxxxxxxxxxxxxxxxxxxxxxxx
REDIS_URL=redis://localhost:6379/0
```

6.3. 통합 서비스 실행

Docker Compose를 사용하여 모든 서비스를 한 번에 빌드하고 실행합니다.

```
docker-compose up -d --build
```

실행되는 서비스 목록:

- **db**: PostgreSQL 데이터베이스 (포트 5678:5432 매핑됨)
- **redis**: RAG 서비스용 Redis (포트 6380:6379 매핑됨)
- **rag-service**: Python RAG API (포트 8000:8000 매핑됨)
- **api**: RenuxServer (.NET Backend, 내부 포트 8080)
- **nginx**: 리버스 프록시 (http://localhost:8080 으로 접근 가능)

6.4. 실행 확인

브라우저에서 `http://localhost:8080` 으로 접속하면 Renux 서비스의 메인 페이지(Frontend)를 확인할 수 있습니다.