Medical RAG Project – Teknik Dökümantasyon

Versiyon: 1.0 • **Son güncelleme:** 21 Temmuz 2025
Yazar: Muhammed İkbal KARADELİ
İletişim: karadeli2001@hotmail.com

---

1 Teknik Dökümantasyona Giriş ve Amacı

Bu doküman, **Medical RAG Projesi**'nin kod tabanını **anlamak, kurmak, genişletmek ve denetlemek** isteyen geliştiriciler için ayrıntılı bir referans sunar. README.md dosyasını tamamlayıcı nitelikte olup mimari yapı, veri akışı, kurulum ayrıntıları, veritabanı şeması ve sorun giderme yönergelerini içerir.

---

2 Proje Genel Bakış

Medical RAG Projesi, modüler Python kodu üzerine inşa edilmiş ve tamamen Dockerize edilmiş uçtan uca bir sistemdir. **Retrieval-Augmented Generation (RAG)** yaklaşımıyla tıbbi sorulara yanıt üretir.

Ana bileşenler:

- **FastAPI** REST arka ucu (`/query`, `/status` endpoint'leri)
- PubMed özetlerinden **gerçek zamanlı semantik geri getirme** (retrieval)
- Yanıt üretimi için **LLM — BioGPT**
- Sağlam kayıt, metrik ve izlenebilirlik için **PostgreSQL** (`db_logger`)
- Yerel etkileşim için **Tkinter tabanlı GUI**
- Otomatik **değerlendirme metrikleri** (BLEU, ROUGE-L, BERTScore F1)
- **Çoklu sorgu** betiği ve toplu değerlendirme
- Tam **Docker Compose** tabanlı dağıtım

Tüm servisler Docker Compose ile orkestre edilen konteynerlerde çalışır; ağ iletişimi Docker'ın **dahili köprü ağı** (internal bridge) ile sağlanır.

---

3 Proje Dizin Yapısı

```
medical_rag_project/
├─ app/
│  ├─ adjunct.py
│  ├─ api_routes.py
│  ├─ config.py
```

```
│   ├─ db_logger.py
│   ├─ document_loader.py
│   ├─ generation.py
│   ├─ init_db.py
│   ├─ main.py
│   ├─ retrieval.py
│   └─ __init__.py
├─ gui/
│   └─ main.py
├─ scripts/
│   ├─ send_bulk_queries.py
│   └─ 100_question.json
├─ fast_api_test/
│   ├─ test_main.py
│   └─ pytest.ini
├─ test/
│   └─ test_query.py
├─ output/
│   ├─ query-documents.csv
│   └─ rag-log-queries.csv
├─ Dockerfile
├─ docker-compose.yml
├─ .gitignore
└─ README.md
```

---

4 Depo Düzeni

Yol	Amaç
app/	API ve temel iş mantığı için tüm Python modülleri
gui/	Bağımsız Tkinter masaüstü arayüzü
scripts/	Yardımcı/çoklu sorgu betikleri
fast_api_test/	Pytest ile otomatik endpoint testleri
test/	Manuel API test betiği
output/	Veritabanı tablolarının CSV dışa aktarımı
Dockerfile & **docker-compose.yml**	Konteyner tanımları
init_db.py	İlk çalıştırmada tabloları oluşturan başlatıcı

---

5 Backend Çekirdek Modülleri (`/app`)

a) adjunct.py

Yardımcı fonksiyonları barındırır.

- ` classify_query_length(query )` – Sorguyu kelime sayısına göre **Kısa / Orta / Uzun** olarak sınıflandırır.

b) config.py

Ortamdan bağımsız tüm ayarları merkezî olarak yönetir.

- Model konfigürasyonları
- PostgreSQL bağlantı dizeleri (yerel & Docker)
- Uygulama genelinde kullanılan sabitler

c) document_loader.py

PubMed özetleriyle çalışır.

1. PubMed API'inden makale özetlerini çeker.
2. Özetleri cümlelere böler.
3. Anahtar kelimelere göre filtreleme yapar.

d) retrieval.py

Yoğun (dense) anlamsal arama ile ilgili belgeleri seçer.

1. **Gömüleme:** Sentence-Transformers ile vektörleştirme.
2. **Arama:** FAISS ile en yakın vektörleri bulur.
3. **Çıktı:** En alakalı cümleleri ve ilgili PubMed ID'lerini döndürür.

e) generation.py

Dil modeli (BioGPT) üzerinden yanıt oluşturur.

- Girdi: "getirilen metin + kullanıcı sorusu".
- Serbest biçimde yanıt üretir.

f) api_routes.py

FastAPI endpoint'lerini tanımlar.

- **GET/status** – Servis durumu.
- **POST/query** – Sorgu akışı:
    1. `document_loader` → ilgili cümleleri getirir.
    2. `generation` → BioGPT ile yanıt üretir.
    3. BLEU, ROUGE-L, BERTScore F1 hesaplar.
    4. `db_logger` → tüm veriyi PostgreSQL'e kaydeder.
    5. JSON yanıt döner.

g) db_logger.py

Veritabanı günlük işlemlerini yönetir.

- ` rag_log_queries ` – Soru/yanıt, süreler, metrikler. RETURNING id kullanarak query_documents tabloladaki satırları birbirine bağlayarak, referans makalelrin PMID lerini belirtir.
- ` query_documents ` – Hangi sorgu için hangi PubMed cümleleri kullanıldı.

h) init_db.py

Idempotent biçimde tabloları oluşturur ve veritabanı hazır olana kadar bekler.Her servis ayağı kaldırıldığında ve tekrarda oluşturulduğunda oluşturulur ve ekstra database yapısı oluşturulmadan sistem hazır hale gelir. Servis konteyneri başlatıldığında, veritabanı hazır olana dek bekler.

i) main.py

FastAPI uygulamasının giriş noktasıdır; `api_routes`'u ekler.

---

6 Veritabanı Yapısı ve Günlükleme

a) `rag_log_queries` Tablosu

Sütun	Tür	Açıklama
id	SERIAL PK	
soru	TEXT	Kullanıcı sorgusu
sorgu_uzunluğu	TEXT	Kısa / Orta / Uzun
cevap	TEXT	Model çıktısı
çekme_süresi_ms	INT	Retrieval süresi
oluşturma_süresi_ms	INT	Generation süresi
toplam_süre_ms	INT	Tam gecikme
bleu_puanı	FLOAT	Opsiyonel
rouge_l_puanı	FLOAT	Opsiyonel
bertscore_f1	FLOAT	Opsiyonel
oluşturulma_zamanı	TIMESTAMP	Varsayılan NOW()

b) `query_documents` Tablosu

Sütun	Tür	Açıklama
id	SERIAL PK	
sorgu_id	INT FK	⇢ `rag_log_queries.id`

Sütun	Tür	Açıklama
pmid	VARCHAR(32)	PubMed ID
cümle_indeksi	INT	Abstract içi sıra
oluşturulma_zamanı	TIMESTAMP	

c) CSV Dışa Aktarımları (`/output`)

- `output/rag-log-queries.csv`
- `output/query-documents.csv`

---

7 API Akışı ve Endpoint'ler

a) GET `/status`

Sağlık kontrolü.

```
{"Message": "OK"}
```

b) POST `/query`

Girdi

```
{"question": "Anemi belirtileri nelerdir?"}
```

İş Akışı

1. Yoğun geri getirme ile PubMed cümlelerini alır.
2. BioGPT ile yanıt üretir.
3. BLEU, ROUGE-L, BERTScore F1 hesaplar.
4. Tüm veriyi veritabanına kaydeder.

Çıktı ()

```
{
  "question": "...",
  "top_documents": ["..."],
  "PubMed_ID": ["..."],
  "answer": "...",
  "Query_Length": "Short/Medium/Long",
  "Retrieval_Time_MS": 123,
  "Generation_Time_MS": 456,
  "Total_Time_MS": 579,
  "BLEU": 0.77,
  "ROUGE-L": 0.63,
  "BERTScore_F1": 0.87
}
```

---

8 Kullanıcı Arayüzü

a) Tkinter GUI (`/gui/main.py`)

- Soru girişi ve yanıt görüntüleme.
- API ile HTTP üzerinden iletişim.

b) Test & Değerlendirme Betikleri

- `test/test_query.py` – Tek sorguluk manuel test.
- `scripts/send_bulk_queries.py ` – Çoklu sorgu, JSON sonuç kaydı.

---

9 Dağıtım & Operasyon

a) Docker Compose

Hizmetler

- PostgreSQL veritabanı
- FastAPI uygulaması
- DB Init (`init_db.py`)

Erişim

- API → `http://localhost:8000`
- Veritabanı → `localhost:5434` (yerel) / `db:5432` (Docker ağı)

b) Çalıştırma Adımları

```
# 1) Depoyu klonlayın
git clone https://github.com/MuhammedIkbalKARADELI/medical_rag_project.git
cd medical_rag_project

# 2) Docker imajlarını oluşturun
docker compose build

# 3) Servisleri başlatın
docker compose up
```

c) Konfigürasyon

- Tüm ayarlar **config.py** içinde.
- Gizli bilgiler sabit kodlanmaz; Docker ortam değişkenleri kullanılır.

---

10 Otomatik Metrikler & Değerlendirme

- Her sorgu için BLEU, ROUGE-L ve BERTScore F1 hesaplanır.

- Metrikler veritabanında ve CSV dışa aktarımlarında saklanır.
- Tekrarlanabilir ve adil model karşılaştırması hedeflenmiştir.

---

11 Teknoloji Yığını & Bağımlılıklar

- Python 3.13
- FastAPI & Uvicorn
- SentenceTransformers
- FAISS
- HuggingFace Transformers (BioGPT)
- NLTK
- psycopg2
- PostgreSQL
- Docker

Detaylı liste için **requirements.txt** dosyasına bakınız.

---

12 Sınırlamalar & Notlar

- Bu proje **klinik tanı aracı değildir**.
- Model çıktıları tıbben kesin değildir; uzman onayı gerektirir.
- Veritabanı kayıt sistemi araştırma denetlenebilirliği için tasarlanmıştır; yüksek hacimli üretim yükleri hedeflenmemiştir.