# ****Дорожная карта RAG-микросервиса (0 → MVP → Production)****

## ****Этап 1. Анализ и архитектура (Неделя 1)****

### **1.1. Архитектура системы**

* Спроектировать **многоарендную архитектуру** (multi-tenant):
  + Каждый пользователь (или проект) видит только свои данные.
  + **Модель Project**: у пользователя может быть несколько проектов, у проекта свои документы и свой RAG-пайплайн.
  + Возможность гибко подключать **разные LLM для разных проектов** (например, Groq, OpenAI, LLaMA).
* Архитектура микросервисов:
  + **Auth & User Management** — на базе Django.
  + **RAG Core** — загрузка документов, построение эмбеддингов, поиск.
  + **Vector DB** — Qdrant (отдельный контейнер).
  + **LLM Gateway** — сервис для работы с облачными моделями (Groq API).
  + **Future Bots API** — точка подключения других систем.

### **1.2. Проектирование API**

* **API пользователей и проектов:**
  + POST /auth/register — регистрация.
  + POST /projects/ — создание проекта.
* **Документы:**
  + POST /projects/{id}/documents/ — загрузка PDF/DOCX.
  + GET /projects/{id}/documents/ — список документов.
* **RAG:**
  + POST /projects/{id}/query/ — задать вопрос.
* **Логирование:**
  + GET /projects/{id}/logs/ — запросы и ответы.

**Результат:** диаграмма архитектуры и OpenAPI-спецификация.

## ****Этап 2. Базовая инфраструктура (Неделя 2)****

### **2.1. Настройка окружения**

* Django + DRF проект (rag\_service).
* Docker + docker-compose (сервис web + qdrant + redis + postgres).
* Подключение Qdrant (для каждого проекта создается своя коллекция).

### **2.2. Модели Django**

* User (с JWT авторизацией).
* Project (связь с User).
* Document (файл + ссылка на проект).
* Chunk (текст + embedding + принадлежность к Document).

## ****Этап 3. Документы и Embeddings (Неделя 3)****

### **3.1. Обработка файлов**

* Поддержка **PDF/DOCX** (через pypdf + docx).
* Разделение на **чанки** (например, по 500 токенов).
* Асинхронная обработка (Celery + Redis):
  + Задача на генерацию эмбеддингов.
  + Сохранение эмбеддингов в Qdrant.

### **3.2. Выбор модели Embeddings**

* Использовать text-embedding-3-small (OpenAI) или bge-m3 локально для скорости.

## ****Этап 4. Поиск и Retrieval (Неделя 4)****

* Реализовать поиск **по проекту**:
  + Каждому проекту соответствует коллекция в Qdrant.
  + POST /projects/{id}/query/:
    - Генерация embedding запроса.
    - Поиск **top-k** релевантных чанков.
* Добавить **логирование запросов**.

## ****Этап 5. Generator (LLM) (Неделя 5)****

* Подключить облачную модель **Groq API**.
* Реализовать построение **prompt** с найденным контекстом.
* Мультиязычная поддержка (входной язык детектируется автоматически).

## ****Этап 6. Авторизация и безопасность (Неделя 6)****

* JWT (djangorestframework-simplejwt).
* Роли: **User / Project Owner / Admin**.
* Проверки: проект принадлежит пользователю.

## ****Этап 7. Логирование и аудит (Неделя 7)****

* Логи всех запросов (project\_id, user\_id, query, response).
* Возможность выгружать логи по проектам.
* Rate limiting (DRF throttling).

## ****Этап 8. Оптимизация (Неделя 8)****

* Кэширование популярных ответов (Redis).
* Улучшение поиска через **reranking** (например, Cohere Reranker).
* Подготовка **тестов** (Pytest + DRF).

## ****Этап 9. Деплой (Неделя 9)****

* Gunicorn + Nginx.
* Настройка на локальный сервер.
* CI/CD через GitHub Actions.

## ****Этап 10. Масштабирование (После MVP)****

* Возможность добавлять **ботов для разных проектов** (API).
* Подключение **разных моделей** (например, LLaMA для оффлайна).
* Переход к Kubernetes (если будет много пользователей).