# Zero@Design — FAZ 1.1: Ingestion & Semantic Query Integration

Bu doküman, FAZ 1 kapsamında tamamlanan Supabase şemasının ardından, veri yükleme (Ingestion) ve semantik sorgulama (RAG Query) entegrasyonlarını açıklamaktadır. Amaç, n8n üzerinde metin parçalama (chunking) + embedding akışını kurmak ve FastAPI üzerinden Supabase RPC çağrısı ile bağlama dayalı yanıt üretebilen bir endpoint oluşturmaktır.

## A) n8n Function Node — Chunker

Bu node, gelen metni 1500 kelimelik parçalara ayırır ve her parçayı embedding işlemi için ayrı bir öğe olarak iletir. Normalize Columns (Function) node’undan sonra yerleştirilmelidir.

// === Chunk Text Function (Zero@Design FAZ1) ===  
const CHUNK\_SIZE = 1500; // kelime  
const results = [];  
  
for (const item of items) {  
 const text = item.json.content || "";  
 const words = text.split(/\s+/);  
 for (let i = 0; i < words.length; i += CHUNK\_SIZE) {  
 const chunk = words.slice(i, i + CHUNK\_SIZE).join(" ");  
 const newItem = { ...item, json: { ...item.json, content: chunk } };  
 results.push(newItem);  
 }  
}  
  
return results;

Flow zinciri: HTTP In → Normalize Columns → Chunk Text → OpenAI Embedding → Supabase Insert → HTTP Response

## B) FastAPI Endpoint — Supabase RPC Çağrısı

Bu endpoint, gelen kullanıcı sorgusunun embedding’ini oluşturur, Supabase üzerindeki rag\_search\_cosine RPC fonksiyonunu çağırır ve elde edilen bağlamı kullanarak GPT-5 modeli ile semantik yanıt üretir.

from fastapi import FastAPI  
from pydantic import BaseModel  
import requests, os  
from openai import OpenAI  
  
app = FastAPI()  
oai = OpenAI(api\_key=os.getenv("OPENAI\_API\_KEY"))  
SUPABASE\_URL = os.getenv("SUPABASE\_URL")  
SUPABASE\_KEY = os.getenv("SUPABASE\_SERVICE\_ROLE\_KEY")  
  
class Question(BaseModel):  
 q: str  
 top\_k: int = 5  
  
@app.post("/ai/query")  
def query(body: Question):  
 emb = oai.embeddings.create(model="text-embedding-3-large", input=body.q).data[0].embedding  
 rpc\_url = f"{SUPABASE\_URL}/rest/v1/rpc/rag\_search\_cosine"  
 headers = {  
 "apikey": SUPABASE\_KEY,  
 "Authorization": f"Bearer {SUPABASE\_KEY}",  
 "Content-Type": "application/json"  
 }  
 data = {"query\_embedding": emb, "match\_count": body.top\_k}  
 r = requests.post(rpc\_url, headers=headers, json=data)  
 ctx = r.json() if r.ok else []  
  
 context = "\n\n".join([c["content"] for c in ctx])  
 prompt = f"CONTEXT:\n{context}\n\nQUESTION: {body.q}\n\nAnswer in Turkish with a short summary and bullet points."  
 chat = oai.chat.completions.create(  
 model="gpt-5.1-mini",  
 messages=[  
 {"role": "system", "content": "You are Zero@Design AI Core. Answer precisely using context."},  
 {"role": "user", "content": prompt}  
 ]  
 )  
  
 return {  
 "answer": chat.choices[0].message.content,  
 "context\_count": len(ctx),  
 "sources": [c.get("source") for c in ctx]  
 }

Çalıştırma:

uvicorn app.main:app --host 0.0.0.0 --port 5000 --reload

Test:

curl -X POST http://localhost:5000/ai/query -H "Content-Type: application/json" -d '{"q":"What are the main sustainability differences between organic cotton and rPET?"}'

## Environment Variables

OPENAI\_API\_KEY=sk-...  
SUPABASE\_URL=https://<your-project>.supabase.co  
SUPABASE\_SERVICE\_ROLE\_KEY=eyJhbGciOi...

## Example Output

{  
 "answer": "• Organic cotton has high water use but biodegradable\n• rPET lowers CO₂ and water impact\n→ Choose blend 60% rPET + 40% organic cotton",  
 "context\_count": 5,  
 "sources": ["manual\_upload", "csv\_seed"]  
}

## Flow Summary

1. n8n, gelen metinleri parçalayıp (chunk) OpenAI üzerinden embedding oluşturur.  
2. Embedding verileri Supabase’e kaydedilir (knowledge\_chunks tablosu).  
3. FastAPI, gelen sorgunun embedding’ini oluşturur ve Supabase RPC ile semantik arama yapar.  
4. LLM, bulunan içerik bağlamını kullanarak açıklamalı yanıt üretir.