

Les 15 — Lesopdracht

PDF Q&A app opzetten + eerste embeddings

Vak: AI-Assisted Development
Duur: 30 minuten in-class
Status: Tijdens de les afronden — daarna huiswerk

Doel

Werkende PDF Q&A app: upload PDF, chunk + embed + store in Supabase pgvector, stel vragen die accuraat beantwoord worden op basis van de PDF.

1

Vereisten

- Node 20+, pnpm, git
- Supabase account
- OpenAI API key

2

Stap 1 — Nieuwe Next.js app

```
pnpm create next-app@latest pdf-qa \
  --typescript --tailwind --app --no-src-dir --import-alias "@/*"
cd pdf-qa
pnpm add ai @ai-sdk/openai zod @supabase/supabase-js unpdf
```

.env.local:

```
OPENAI_API_KEY=sk-...
NEXT_PUBLIC_SUPABASE_URL=https://...supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
```

3

Stap 2 — Supabase pgvector + schema

```
create extension if not exists vector;

create table chunks (
  id bigserial primary key,
  source text not null, page int,
  content text not null,
  embedding vector(1536),
  created_at timestamp default now()
);

create index on chunks using hnsw (embedding vector_cosine_ops);
alter table chunks enable row level security;
create policy "demo open" on chunks
  for all to anon using (true) with check (true);

create or replace function match_chunks(
  query_embedding vector(1536), match_count int default 5
) returns table (id bigint, content text, source text, similarity float)
language sql stable as $$
  select chunks.id, chunks.content, chunks.source,
    1 - (chunks.embedding <=> query_embedding) as similarity
  from chunks
  order by chunks.embedding <=> query_embedding
  limit match_count;
$$;
```

4 Stap 3 — lib/embeddings.ts

```
import { embed, embedMany } from "ai";
import { openai } from "@ai-sdk/openai";

export const embedModel = openai.textEmbeddingModel("text-embedding-3-small");

export async function embedOne(text: string) {
  const { embedding } = await embed({ model: embedModel, value: text });
  return embedding;
}

export async function embedBatch(texts: string[]) {
  const { embeddings } = await embedMany({ model: embedModel, values: texts });
  return embeddings;
}

export function chunkText(text: string, size = 500, overlap = 50) {
  const chunks: string[] = [];
  let i = 0;
  while (i < text.length) {
    chunks.push(text.slice(i, i + size).trim());
    i += size - overlap;
  }
  return chunks.filter((c) => c.length > 0);
}
```

5 Stap 4 — Index endpoint

```
// app/api/index/route.ts
import { extractText } from "unpdf";
import { embedBatch, chunkText } from "@lib/embeddings";
import { supabase } from "@lib/supabase";

export async function POST(req: Request) {
  const formData = await req.formData();
  const file = formData.get("file") as File;
  const buffer = new Uint8Array(await file.arrayBuffer());
  const { text } = await extractText(buffer, { mergePages: true });

  const chunks = chunkText(text, 500, 50);
  const embeddings = await embedBatch(chunks);

  await supabase.from("chunks").insert(
    chunks.map((content, i) => ({
      source: file.name, content,
      embedding: embeddings[i],
    }))
  );
  return Response.json({ chunks: chunks.length });
}
```

6 Stap 5 — Ask endpoint

```
// app/api/ask/route.ts
export async function POST(req: Request) {
  const { question } = await req.json();
  const embedding = await embedOne(question);

  const { data: chunks } = await supabase.rpc("match_chunks", {
    query_embedding: embedding, match_count: 5,
  });

  const context = chunks.map((c) =>
    `[${c.source}]\n${c.content}`
  ).join("\n\n---\n\n");

  const { text } = await generateText({
    model: openai("gpt-4o-mini"),
    prompt: `Beantwoord op basis van context:\n${context}\n\nVraag: ${question}`,
  });
  return Response.json({ answer: text, sources: chunks });
}
```

7 Stap 6 — Test in terminal

```
# Terminal 1
pnpm dev

# Terminal 2
curl -X POST http://localhost:3000/api/index \
  -F "file=@./polderfest-lineup.pdf"
# -> {"chunks": 30}

curl -X POST http://localhost:3000/api/ask \
  -H "Content-Type: application/json" \
  -d '{"question": "Wie zijn de headliners?"}' \
  | jq
```

8 Eisen

- pgvector extension actief in Supabase
- chunks tabel + HNSW index + match_chunks function
- 1 PDF succesvol geïndexeerd
- 3 vragen werken met correcte antwoorden
- Chunks zichtbaar in Supabase Table Editor

9

Veelvoorkomende problemen

Symptoom	Oplossing
extension 'vector' does not exist	Supabase Database -> Extensions -> enable
dimension mismatch	Embed model en vector(N) moeten matchen
match_chunks does not exist	SQL function niet aangemaakt — stap 3
RLS error op insert	Open policy nog niet uitgevoerd
PDF parsing leeg	Scan-PDF (image-based)? unpdf werkt op text-PDFs

Klaar? Verder met huiswerk

Eigen PDF + UI + RAG-tool in agent + RAG.md analyse. Zie Les15-Huiswerk.pdf.