

Les 13 — Lesopdracht

Setup research-agent + eerste run

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

Doel

Aan het einde heb je een werkende research-agent in een eigen Next.js repo, draait er één test-query, en kun je in de console alle stappen zien.

1 Vereisten

- Node 20+, pnpm, git
- Supabase account (mag dezelfde zijn als Les 11/12)
- OpenAI API key (zit waarschijnlijk al in je .env)
- **Tavily API key** — gratis tier op tavily.com (1000 calls/maand)
- VS Code / Cursor

2 Stap 1 — Nieuwe Next.js app

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

3 Stap 2 — Environment variabelen

.env.local (in .gitignore!):

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

Let op

OPENAI_API_KEY is GEEN NEXT_PUBLIC_* — moet server-side blijven.

4 Stap 3 — Supabase tabel

```
create table research_reports (  
  id          bigserial primary key,  
  query       text not null,  
  summary     text not null,  
  sources     jsonb default '[]'::jsonb,  
  created_at  timestamp default now()  
);  
  
alter table research_reports enable row level security;  
create policy "demo open" on research_reports  
  for all to anon using (true) with check (true);
```

5 Stap 4 — Tools schrijven

lib/tools.ts — webSearch met Tavily:

```
import { tool } from "ai";  
import { z } from "zod";  
import { createClient } from "@supabase/supabase-js";  
  
const supabase = createClient(  
  process.env.NEXT_PUBLIC_SUPABASE_URL!,  
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,  
);  
  
export const webSearch = tool({  
  description: "Zoek op het web. Top 5 resultaten met url + snippet.",  
  inputSchema: z.object({ query: z.string() }),  
  execute: async ({ query }) => {  
    const res = await fetch("https://api.tavily.com/search", {  
      method: "POST",  
      headers: { "Content-Type": "application/json" },  
      body: JSON.stringify({  
        api_key: process.env.TAVILY_API_KEY,  
        query, max_results: 5,  
      }),  
    });  
    const data = await res.json();  
    return data.results?.map((r: any) => ({  
      url: r.url, title: r.title, snippet: r.content,  
    })) ?? [];  
  },  
});
```

readPage tool:

```
export const readPage = tool({
  description: "Haal de volledige tekst van een URL op.",
  inputSchema: z.object({ url: z.string().url() }),
  execute: async ({ url }) => {
    const res = await fetch("https://api.tavily.com/extract", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        api_key: process.env.TAVILY_API_KEY,
        urls: [url],
      }),
    });
    const data = await res.json();
    const content = data.results?.[0]?.raw_content ?? "";
    return { url, content: content.slice(0, 5000) };
  },
});
```

saveReport tool:

```
export const saveReport = tool({
  description:
    "Sla een onderzoeksrapport op. Alleen als compleet.",
  inputSchema: z.object({
    query: z.string(),
    summary: z.string().min(100),
    sources: z.array(z.object({
      url: z.string().url(),
      title: z.string(),
    })),
  }),
  execute: async ({ query, summary, sources }) => {
    const { data, error } = await supabase
      .from("research_reports")
      .insert({ query, summary, sources })
      .select().single();
    if (error) return { error: error.message };
    return { reportId: data.id, success: true };
  },
});
```

6

Stap 5 — Agent definiëren

lib/agent.ts:

```
import { ToolLoopAgent, stepCountIs } from "ai";
import { openai } from "@ai-sdk/openai";
import { webSearch, readPage, saveReport } from "./tools";

export const researchAgent = new ToolLoopAgent({
  model: openai("gpt-4o-mini"),
  system: `Je bent een research-assistent. Werk als volgt:
1. Begin met een kort plan (3-5 bullets).
2. Gebruik webSearch voor orientatie.
3. Gebruik readPage voor de 2-3 meest relevante URLs.
4. Schrijf een samenvatting van min 200 woorden.
5. Sla op met saveReport. Inclusief alle gebruikte sources.

Verzin niets - gebruik alleen wat je via tools hebt opgehaald.`,
  tools: { webSearch, readPage, saveReport },
  stopWhen: stepCountIs(10),
});
```

7

Stap 6 — API route

app/api/research/route.ts:

```
import { researchAgent } from "@lib/agent";

export async function POST(req: Request) {
  const { query } = await req.json();
  const result = await researchAgent.generate({ prompt: query });

  return Response.json({
    text: result.text,
    steps: result.steps.map((s, i) => ({
      step: i,
      toolCalls: s.toolCalls?.map((tc) => ({
        tool: tc.toolName, input: tc.input,
      })),
      hasText: !!s.text,
    })),
  });
}
```

8 Stap 7 — Test in terminal

Terminal 1:

```
pnpm dev
```

Terminal 2:

```
curl -X POST http://localhost:3000/api/research \
-H "Content-Type: application/json" \
-d '{"query": "Wat zijn de belangrijkste AI-trends in 2026?"}' \
| jq
```

Verwacht: JSON met text (eindrapport) en steps (welke tools werden aangeroepen in welke volgorde).

9 Eisen

- Repo werkt — pnpm dev zonder errors
- .env.local met alle 4 keys + in .gitignore
- Supabase tabel aangemaakt
- Alle 3 tools werken
- Agent draait 1 succesvolle query
- Je kunt in steps zien welke tools werden aangeroepen
- Eindrapport staat in Supabase

10 Veelvoorkomende problemen

Symptoom	Oplossing
TAVILY_API_KEY undefined	Restart pnpm dev na env-edit
Tavily 401	Key fout — kopieer opnieuw
Supabase insert faalt	RLS policy check
Agent stopt te vroeg	stepCountIs(10) -> stepCountIs(20)
Steps array leeg	Check je result.steps-mapping
gpt-4o-mini does not exist	Check OpenAI key GPT-4 access

Klaar? Verder met huiswerk

Het huiswerk bouwt voort op deze setup: extra tools (listReports), prepareStep toevoegen, custom stop-conditie, en AGENT.md documentatie. Zie Les13-Huiswerk.pdf.