Les 13 — Stap voor stap

Polderfest → Polderfest-met-agent

Vak: AI-Assisted Development
Vervolg op: Les 12 — Tool Calling
Zip: polderfest-with-agent.zip — referentie

Dit document beschrijft welke stappen je doet om van de Polderfest-app (Les 11/12) toe te werken naar een echte agent met vijf tools, een eind- tool, model-switching en een write-tool.

Startpunt: je eigen Polderfest-app uit Les 12, of `polderfest-original.zip` uitpakken. **Eindpunt:** vergelijkbaar met `polderfest-with-agent.zip` — open die als spiekbriefje.

Wat is AL bekend (uit Les 11/12)

Dit doen we vandaag **niet opnieuw** — het zit al in je Polderfest-app:

- `tool({ description, inputSchema, execute })`
- `stopWhen: stepCountIs(N)` voor multi-step
- Supabase-client opzetten
- `searchBands` en `getDaySchedule` als read-tools
- Bestaande tabellen `bands` en `user_favorites`

Wat is NIEUW vandaag

Zes echt nieuwe features. Allemaal in één nieuwe route + bijbehorende UI.

#	Stap	STEP-anchor (Cmd+F in code)
1	Project klaarzetten	—
2	Nieuwe route app/api/agent/route.ts	—
3	addToFavorites write-tool	STEP 8 — addToFavorites
4	getWeather externe API tool	STEP 7 — getWeather
5	finishPlan structured-output tool	STEP 9 — finishPlan

#	Stap	STEP-anchor (Cmd+F in code)
6	hasToolCall extra stop-conditie	STEP 10 — hasToolCall
7	prepareStep voor model-switching	STEP 11 — prepareStep
8	UI-pagina /agent met tool-rendering	STEP 13 — Tool-call rendering
9	Testen + deploy	—

Stap 1 — Project klaarzetten

```
unzip polderfest-original.zip
cd polderfest-original
cp .env.example .env.local       # zelf invullen
npm install
npm run dev
```

Browser: `http://localhost:3000` — de chat uit Les 11/12 werkt.

Verifieer in Supabase dat de tabel `user_favorites` bestaat met kolommen `id`, `user_email`, `band_id`, `created_at`. Die hebben we nodig voor de write-tool.

Stap 2 — Nieuwe route `app/api/agent/route.ts`

Kopieer `app/api/chat/route.ts` naar `app/api/agent/route.ts` als startpunt. Verwijder daar voor nu eventuele bands-context — we gaan de agent puur via tools laten werken.

```
import {
  convertToModelMessages,
  streamText,
  stepCountIs,
  hasToolCall,           // ← nieuw vergeleken met Les 12
  tool,
  type UIMessage,
} from "ai";
import { openai } from "@ai-sdk/openai";
import { createClient } from "@supabase/supabase-js";
import { z } from "zod";

const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.SUPABASE_SERVICE_ROLE_KEY ??
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
);

export async function POST(req: Request) {
  const { messages }: { messages: UIMessage[] } = await req.json();
  const result = streamText({
    model: openai("gpt-5.2"),
    system: "Je bent een Polderfest-agent.",
    messages: await convertToModelMessages(messages),
    stopWhen: stepCountIs(8),
    tools: {
      // searchBands en getDaySchedule overnemen uit chat/route.ts
    },
  });
  return result.toUIMessageStreamResponse();
}
```

Test eerst dat de route bereikbaar is, bouw daarna stap-voor-stap uit.

Stap 3 — `addToFavorites` write-tool (NIEUW)

Eerste tool die de **wereld verandert** ipv. alleen leest. Anchor: `STEP 8 — addToFavorites`

```
addToFavorites: tool({
  description:
    "Voeg een band toe aan de favorieten van de gebruiker. " +
    "Gebruik dit als de gebruiker zegt: ik vind X mooi, voeg toe, etc.",
  inputSchema: z.object({
    bandName: z.string(),
  }),
  execute: async ({ bandName }) => {
    // 1. Zoek band op naam — we hebben band_id nodig (FK)
    const { data: band, error: findErr } = await supabase
      .from("bands")
      .select("id, name")
      .ilike("name", `%${bandName}%`)
      .limit(1)
      .single();
    if (findErr || !band) {
      return { error: `Band '${bandName}' niet gevonden` };
    }

    // 2. Insert in user_favorites
    const demoUser = "demo@polderwave.app";
    const { data, error } = await supabase
      .from("user_favorites")
      .insert({ user_email: demoUser, band_id: band.id })
      .select()
      .single();
    if (error) return { error: error.message };
    return { added: true, band: band.name, favoriteId: data.id };
  },
}),
```

Test: zeg tegen de agent *"Voeg Band 5 toe aan mijn favorieten."* en check in de Supabase Table Editor of er een rij in `user_favorites` verschijnt.

Stap 4 — `getWeather` externe API (NIEUW)

Geen account, geen API-key — Open-Meteo is gratis. Toont dat tools ook gewoon `fetch()` kunnen doen naar externe APIs. Anchor: `STEP 7 — getWeather`

```
getWeather: tool({
  description: "Weer-forecast voor een festivaldag.",
  inputSchema: z.object({
    date: z.string().describe("ISO datum, bv. '2027-07-10'"),
  }),
  execute: async ({ date }) => {
    const url = new URL("https://api.open-meteo.com/v1/forecast");
    url.searchParams.set("latitude", "52.0907");
    url.searchParams.set("longitude", "5.1214");
    url.searchParams.set(
      "daily",
      "temperature_2m_max,precipitation_probability_max",
    );
    url.searchParams.set("start_date", date);
    url.searchParams.set("end_date", date);
    url.searchParams.set("timezone", "Europe/Amsterdam");

    const res = await fetch(url);
    if (!res.ok) return { error: `Weather API faalde: ${res.status}` };
    const data = await res.json();
    const d = data?.daily;
    return {
      date,
      temperatureMaxC: d?.temperature_2m_max?.[0],
      rainChancePercent: d?.precipitation_probability_max?.[0],
    };
  },
}),
```

Festivaldata: vrijdag `2027-07-09`, zaterdag `2027-07-10`, zondag `2027-07-11`.

Stap 5 — `finishPlan` structured-output tool (NIEUW)

Dwingt een gestructureerd eindplan af via Zod-schema. De UI kan dit later als nette plan-kaart renderen.
Anchor: `STEP 9 — finishPlan`

```
finishPlan: tool({
  description:
    "Roep dit ALS LAATSTE aan. Geef het eindprogramma in gestructureerde vorm. " +
    "De agent stopt automatisch na deze aanroep.",
  inputSchema: z.object({
    summary: z.string().describe("Korte samenvatting"),
    items: z
      .array(
        z.object({
          time: z.string().describe("HH:MM"),
          band: z.string(),
          stage: z.string(),
          reason: z.string(),
        }),
      )
      .describe("Items in chronologische volgorde"),
  }),
  execute: async ({ summary, items }) => {
    return { summary, items, ok: true };
  },
}),
```

Stap 6 — `hasToolCall` extra stop-conditie (NIEUW)

Combineer `stepCountIs(8)` met `hasToolCall("finishPlan")`: Anchor: `STEP 10 — hasToolCall`

```
import { stepCountIs, hasToolCall } from "ai";

streamText({
  // ...
  stopWhen: [
    stepCountIs(8),                 // max 8 stappen (veiligheid)
    hasToolCall("finishPlan"),      // klaar-signaal van de agent
  ],
});
```

Effect: zodra de agent `finishPlan` aanroept stopt de loop direct. Gegarandeerde nette afsluiting, ongeacht resterende stappen.

Stap 7 — `prepareStep` voor model-switching (NIEUW)

Per stap een ander model. Stap 0 (planning) krijgt het krachtige model. Stap 1+ (uitvoering van tool-calls) gebruikt een lichter model. Anchor: `STEP 11 — prepareStep`

```
streamText({
  model: openai("gpt-5.2"),  // default

  prepareStep: ({ stepNumber }) => {
    if (stepNumber === 0) {
      return { model: openai("gpt-5.2") };       // planning
    }
    return { model: openai("gpt-5-mini") };       // uitvoering
  },

  // ...rest van je streamText config
});
```

Resultaat: significant lagere kosten en latency zonder kwaliteitsverlies op het denkwerk.

Stap 8 — UI-pagina `/agent` met tool-rendering (NIEUW)

Maak `app/agent/page.tsx`. Verschil met `/`: Anchor: `STEP 13 — Tool-call rendering`

```
const { messages, sendMessage, status } = useChat({
  api: "/api/agent",       // ← belangrijk: agent-route
});
```

Speciaal renderen voor `finishPlan` en `addToFavorites`:

```
{m.parts.map((p, i) => {
  if (p.type === "text") return <div>{p.text}</div>;

  if (p.type === "tool-finishPlan") {
    return <PlanCard result={p.result} />;       // nette timeline
  }
  if (p.type === "tool-addToFavorites") {
    return <FavoriteBadge name={p.input.bandName} />;   // ■ badge
  }
  if (p.type.startsWith("tool-")) {
    return <ToolDetails part={p} />;             // expandable JSON
  }
})}
```

De gestructureerde output uit `finishPlan` kun je nu prachtig renderen — geen vrije-vorm-tekst meer die je moet parsen.

Stap 9 — Testen in de browser

Open `localhost:3000/agent`. Probeer:

> "Plan een avond met indie en techno voor zaterdag."

→ meerdere tool-calls + `finishPlan` aan het eind. Plan-kaart in de UI.

> "Welke band moet ik NIET missen op vrijdag? Zet hem in mijn favorieten."

→ Tool `addToFavorites` aangeroepen, badge in UI, rij in `user_favorites`.

> "Wat speelt er op zondag en hoe is het weer?"

→ Externe weer-API call. Agent combineert weer + bands.

Stap 10 — Deploy

Push naar GitHub, importeer in Vercel, env vars:

- `NEXT_PUBLIC_SUPABASE_URL`
- `NEXT_PUBLIC_SUPABASE_ANON_KEY`
- `SUPABASE_SERVICE_ROLE_KEY`
- `OPENAI_API_KEY`

Samenvatting — wat je nu hebt

Zes nieuwe agent-features bovenop Les 12:

Feature	Voorbeeld in code
addToFavorites — write naar user_favorites	STEP 8
getWeather — externe Open-Meteo API	STEP 7
finishPlan — Zod-schema structured output	STEP 9
hasToolCall extra stop-conditie	STEP 10
prepareStep model per stap	STEP 11
UI met tool-call rendering	STEP 13

Plus de bestaande tools (`searchBands`, `getDaySchedule`) en patterns (`tool()`, `stopWhen`, Supabase) die je al uit Les 11/12 kende.

Vastgelopen?

`polderfest-with-agent.zip` is de complete referentie. Zoek op `// STEP N` om direct bij de juiste plek te komen. Reset `user_favorites` tussen demos met `supabase-reset-user-favorites.sql`.