NOVI Hogeschool — AI Leerlijn

Les 8 — Lesopdracht

Van In-Memory naar Supabase

Werk deze opdracht stap voor stap door.
Alle UI-code staat er al in — jij schrijft de Supabase queries.

Les	8 van 18
Duur	3 uur
Focus	Supabase koppelen + /create pagina
Jij doet	Supabase queries schrijven (TODO blokken)
Vereisten	Werkend QuickPoll + Supabase project

Legenda:
Grijze blokken = code die je kunt copy-pasten
Gele blokken = TODO — dit moet jij zelf schrijven!

Deel 1: Setup (samen met docent)

Stap 1.1 — Installeer Supabase

Open je terminal in het QuickPoll project en voer uit:

```
npm install @supabase/supabase-js
```

Wacht tot het klaar is. Herstart daarna je dev server!

Stap 1.2 — Environment variables

Maak **.env.local** aan in de root van je project:

```
NEXT_PUBLIC_SUPABASE_URL=https://jouw-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=jouw-anon-key-hier
```

Vind je URL en key in Supabase dashboard: Settings > API. Na .env wijzigen: herstart dev server!

Stap 1.3 — Supabase client

Maak **lib/supabase.ts** aan:

```
import { createClient } from "@supabase/supabase-js";

export const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
);
```

Stap 1.4 — Types

Update **types/index.ts** zodat het matcht met je database:

```
export interface Poll {
  id: number;
  question: string;
  created_at: string;
  options: Option[];
}

export interface Option {
  id: number;
  poll_id: number;
  text: string;
  votes: number;
  created_at: string;
}
```

Klaar met de setup? Ga door naar Deel 2!

Deel 2: Data laag — Supabase queries

Nu ga je **lib/data.ts** herschrijven. De oude versie gebruikt in-memory arrays. Jij schrijft Supabase queries.

Stap 2.1 — Kopieer dit bestand

Vervang de inhoud van **lib/data.ts** met onderstaande code. De TODO blokken moet jij invullen!

```
import { supabase } from "./supabase";
import { Poll } from "@/types";
```

getPolls() — Haal alle polls op

```
export async function getPolls(): Promise<Poll[]> {
// TODO: Schrijf een Supabase query die alle polls ophaalt
// inclusief hun options (relatie).
//
// Hints:
// supabase.from("polls").select("*, options(*)")
// Bij error: console.error en return []
// Bij succes: return data || []
}
```

getPollById() — Haal 1 poll op

```
export async function getPollById(id: number): Promise<Poll | null> {
// TODO: Haal 1 specifieke poll op aan de hand van id.
//
// Hints:
// .from("polls").select("*, options(*)")
// .eq("id", id) <-- filter op id
// .single() <-- verwacht 1 resultaat
// Bij error: return null
// Bij succes: return data
}
```

votePoll() — Stem op een optie

```
export async function votePoll(optionId: number): Promise<boolean> {
// TODO: Roep de Supabase RPC functie aan om te stemmen.
//
// Hints:
// supabase.rpc("vote_option", { option_id: optionId })
// Bij error: console.error en return false
// Bij succes: return true
}
```

Klaar? Test door naar http://localhost:3000 te gaan. Je zou polls uit Supabase moeten zien!

Deel 3: Componenten updaten

Nu je data uit Supabase komt, moeten de componenten aangepast worden. Kopieer onderstaande code.

Stap 3.1 — app/page.tsx (Server Component)

Dit is nu een **async** Server Component die data ophaalt met **await**:

```
import { getPolls } from "@/lib/data";
import Link from "next/link";
import PollItem from "@/components/PollItem";

export default async function HomePage() {
  const polls = await getPolls();

  return (
    <div className="w-full max-w-2xl mx-auto p-6">
      <h1 className="text-3xl font-bold mb-6">Huidige Polls</h1>
      <Link href="/create"
        className="text-blue-600 hover:underline mb-6 block">
        + Nieuwe Poll
      </Link>
      <div className="space-y-4">
        {polls.map((poll) => (
          <PollItem key={poll.id} poll={poll} />
        ))}
      </div>
    </div>
  );
}
```

Stap 3.2 — components/PollItem.tsx

Client component met percentage bars:

```
'use client'

import Link from "next/link";
import { Option } from "@/types";

interface PollItemProps {
  poll: {
    id: number;
    question: string;
    options: Option[];
  };
}

export default function PollItem({ poll }: PollItemProps) {
  const totalVotes = poll.options.reduce(
    (sum, opt) => sum + opt.votes, 0
  );

  return (
```

```
    <div className="border rounded p-4">
      <h2 className="text-xl font-semibold mb-4">
        {poll.question}
      </h2>
      <div className="space-y-2">
        {poll.options.map((option) => {
          const pct = totalVotes > 0
            ? (option.votes / totalVotes) * 100 : 0;
          return (
            <Link key={option.id}
              href={`/poll/${poll.id}`}>
              <div className="flex items-center
                gap-2 cursor-pointer
                hover:opacity-80">
                <div className="flex-1 bg-gray-200
                  rounded h-8 overflow-hidden">
                  <div className="bg-blue-600
                    h-full transition-all"
                    style={{ width: `${pct}%` }}
                  />
                </div>
                <span className="text-sm
                  font-medium w-20">
                  {option.text} ({option.votes})
                </span>
              </div>
            </Link>
          );
        })}
      </div>
    </div>
  );
}
```

Stap 3.3 — components/VoteForm.tsx (Client Component)

Dit component laat gebruikers stemmen. De UI staat er al in. Bekijk hoe **votePoll()** wordt aangeroepen — die functie heb jij in Deel 2 geschreven!

```
'use client'

import { useState } from "react";
import { votePoll } from "@/lib/data";
import { Option } from "@/types";

interface VoteFormProps {
  options: Option[];
}

export default function VoteForm({ options }: VoteFormProps) {
  const [loading, setLoading] = useState(false);
  const [voted, setVoted] = useState(false);

  const handleVote = async (optionId: number) => {
    setLoading(true);
    const success = await votePoll(optionId);
    if (success) {
      setVoted(true);
    }
    setLoading(false);
  };

  if (voted) {
    return <p className="text-green-600">
      Dank je voor je stem!</p>;
  }

  return (
    <div className="space-y-2">
      {options.map((option) => (
        <button key={option.id}
          onClick={() => handleVote(option.id)}
          disabled={loading}
          className="w-full bg-blue-600 text-white
            p-2 rounded hover:bg-blue-700
            disabled:opacity-50">
          {option.text}
        </button>
      ))}
    </div>
  );
}
```

Stap 3.4 — app/poll/[id]/page.tsx

Server Component die een enkele poll toont + VoteForm:

```
import { getPollById } from "@/lib/data";
import VoteForm from "@/components/VoteForm";
import { notFound } from "next/navigation";

export default async function PollPage(
```

```
  { params }: { params: { id: string } }
) {
  const poll = await getPollById(
    parseInt(params.id)
  );

  if (!poll) {
    notFound();
  }

  return (
    <div className="w-full max-w-md mx-auto p-6">
      <h1 className="text-2xl font-bold mb-6">
        {poll.question}
      </h1>
      <VoteForm options={poll.options} />
    </div>
  );
}
```

Test: ga naar http://localhost:3000. Polls laden? Stemmen werkt? Check Supabase dashboard!

Deel 4: /create pagina bouwen

Nu het spannende deel: een pagina waarmee je **nieuwe polls kunt aanmaken**. De volledige UI staat hieronder. Jij schrijft alleen de Supabase INSERT logica.

Stap 4.1 — RLS policy toevoegen

Zonder dit blokkeert Supabase je INSERT. Open **Supabase dashboard > SQL Editor** en voer uit:

```
-- INSERT policy voor polls
CREATE POLICY "Allow public insert on polls"
ON polls FOR INSERT
TO anon
WITH CHECK (true);

-- INSERT policy voor options
CREATE POLICY "Allow public insert on options"
ON options FOR INSERT
TO anon
WITH CHECK (true);
```

Dit is tijdelijk — volgende les beperken we dit met Auth.

Stap 4.2 — INSERT queries (theorie)

Zo werkt INSERT in Supabase:

```
// 1. Insert poll (krijg id terug)
const { data: poll, error } = await supabase
  .from("polls")
  .insert({ question: "Mijn vraag" })
  .select()
  .single();

// poll.id is nu beschikbaar!

// 2. Insert meerdere options tegelijk
await supabase.from("options").insert([
  { poll_id: poll.id, text: "Optie A", votes: 0 },
  { poll_id: poll.id, text: "Optie B", votes: 0 },
]);
```

.insert() = INSERT statement. .select().single() = geef het inserted record terug. poll.id gebruik je voor de options.

Stap 4.3 — app/create/page.tsx

Maak **app/create/page.tsx** aan. Kopieer onderstaande code. Het formulier is compleet — jij vult alleen **handleSubmit** in!

```
'use client'

import { useState } from "react";
import { useRouter } from "next/navigation";
import { supabase } from "@/lib/supabase";

export default function CreatePollPage() {
  const [question, setQuestion] = useState("");
  const [options, setOptions] = useState(["", ""]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const router = useRouter();

  const addOption = () =>
    setOptions([...options, ""]);

  const updateOption = (
    index: number, value: string
  ) => {
    const newOptions = [...options];
    newOptions[index] = value;
    setOptions(newOptions);
  };

  const removeOption = (index: number) => {
    if (options.length <= 2) return;
    setOptions(
      options.filter((_, i) => i !== index)
    );
  };

  const handleSubmit = async (
    e: React.FormEvent
  ) => {
    e.preventDefault();
    setLoading(true);
    setError("");

    // TODO: Implementeer Supabase INSERT
    //
    // Stap 1: Insert poll in "polls" tabel
    // const { data: poll, error: pollError } =
    // await supabase.from("polls")
    // .insert({ question })
    // .select().single();
    //
    // Stap 2: Check of pollError niet null is
    // Als error: setError(pollError.message)
    // setLoading(false) en return
    //
    // Stap 3: Insert options met poll.id
    // Filter lege strings uit options array
```

```
    // Map naar objects: { poll_id, text, votes: 0 }
    // await supabase.from("options").insert(...)
    //
    // Stap 4: router.push("/") om terug te gaan

    setLoading(false);
  };
```

Stap 4.3 (vervolg) — return JSX

Dit is het formulier. Kopieer dit onder de handleSubmit functie:

```
return (
  <div className="w-full max-w-md mx-auto p-6">
    <h1 className="text-2xl font-bold mb-6">
      Nieuwe Poll
    </h1>

    {error && (
      <p className="text-red-600 mb-4">
        {error}
      </p>
    )}

    <form onSubmit={handleSubmit}
      className="space-y-4">
      <div>
        <label className="block text-sm
          font-medium mb-1">
          Vraag
        </label>
        <input type="text"
          value={question}
          onChange={(e) =>
            setQuestion(e.target.value)}
          className="w-full p-2 border rounded"
          placeholder="Stel je vraag..."
          required />
      </div>

      <div>
        <label className="block text-sm
          font-medium mb-2">
          Opties
        </label>
        {options.map((option, index) => (
          <div key={index}
            className="flex gap-2 mb-2">
            <input type="text"
              value={option}
              onChange={(e) =>
                updateOption(index,
                  e.target.value)}
              className="flex-1 p-2
                border rounded"
              placeholder={`Optie ${index+1}`}
              required />
            {options.length > 2 && (
              <button type="button"
                onClick={() =>
                  removeOption(index)}
                className="text-red-500
                  hover:text-red-700 px-2">
                X
              </button>
            )}
          </div>
```

```
          ))}
          <button type="button"
            onClick={addOption}
            className="text-blue-600
              hover:underline text-sm">
            + Optie toevoegen
          </button>
        </div>

        <button type="submit"
          disabled={loading}
          className="w-full bg-blue-600
            text-white p-2 rounded
            hover:bg-blue-700
            disabled:opacity-50">
          {loading ? "Bezig..." : "Poll aanmaken"}
        </button>
      </form>
    </div>
  );
}
```

Deel 5: Testen

Checklist

Loop deze checks af. Alles werkt? Dan ben je klaar!

- ■ Homepage laadt polls uit Supabase (niet meer in-memory)
- ■ Elke poll toont opties met percentage bars
- ■ Klik op poll -> detail pagina met stemknoppen
- ■ Stemmen werkt -> check in Supabase dashboard dat votes stijgt
- ■ /create pagina toont formulier met vraag + opties
- ■ Poll aanmaken -> verschijnt op homepage
- ■ Check in Supabase dashboard: nieuwe poll + options staan erin

Troubleshooting

Probleem	Oplossing
"Cannot find module @supabase/supabase-js"	npm install @supabase/supabase-js
Env vars undefined	NEXT_PUBLIC_ prefix? Dev server herstart na .env wijziging?
"RLS policy violation" bij INSERT	Stap 4.1 gedaan? SQL in dashboard uitgevoerd?
poll is undefined na insert	.select().single() toegevoegd aan je insert query?
Form submit refresht de pagina	e.preventDefault() in handleSubmit?
Redirect werkt niet	useRouter van "next/navigation" (niet "next/router"!)
Opties array gaat fout	Gebruik [...options] spread voor immutable updates
getPolls() returns []	Check Supabase SQL Editor: SELECT * FROM polls;

Huiswerk

Verplicht:

1. **/create pagina afmaken** als je niet klaar was
2. **Validatie toevoegen:**
   - Vraag mag niet leeg zijn
   - Minimaal 2 niet-lege opties
   - Toon foutmelding bij ongeldige input

Extra (voor snelle studenten):

1. **Delete functionaliteit** — delete knop bij poll, verwijder uit Supabase
2. **Styling** — maak de app mooier met Tailwind
3. **SQL queries** — test direct in Supabase SQL Editor

Volgende les: Supabase Auth — inloggen, registreren, en bepalen wie wat mag!

Concept overzicht

Concept	Wat het doet	Voorbeeld
supabase.from()	Selecteer een tabel	`.from("polls")`
.select()	Kies kolommen/relaties	`.select("*, options(*)")`
.eq()	Filter op waarde	`.eq("id", 5)`
.single()	Verwacht 1 resultaat	`Voor getPollById`
.insert()	Voeg rijen toe	`.insert({ question })`
.select().single()	Return inserted row	`Na .insert()`
.rpc()	Roep DB functie aan	`.rpc("vote_option")`
async/await	Wacht op Supabase	`await supabase...`
'use client'	Client Component	`Forms, useState`
async function	Server Component	`Data fetching`