

Les 5 — Lesopdracht

QuickPoll Part 1 — bouw de basis

Vak: Technical Foundations
Cursus: AI Developer
Volgende les: Les 6 — QuickPoll Part 2

Duur: ~120 minuten klassikaal bouwen

Doel

Aan het einde heb je een werkende **QuickPoll Part 1**: lijst van polls op homepage, click → detail-page, voting met lokale state (data verdwijnt bij refresh — dat lossen we Les 7 op).

Stap 1 — Setup

```
pnpm create next-app@latest quickpoll \
  --typescript --tailwind --app --no-src-dir --import-alias "@/*"
cd quickpoll
pnpm dev
```

Check: localhost:3000 toont Next.js welcome.

Stap 2 — In-memory data

lib/data.ts:

```
export const polls = [
  {
    id: "1",
    title: "Beste taal?",
    options: [
      { id: "a", text: "TypeScript", votes: 0 },
      { id: "b", text: "Python", votes: 0 },
      { id: "c", text: "Go", votes: 0 },
    ],
  },
];

export function getPoll(id: string) {
  return polls.find(p => p.id === id);
}
```

Stap 3 — Homepage

app/page.tsx:

```
import { polls } from "@lib/data";
import Link from "next/link";

export default function Home() {
  return (
    <main className="max-w-2xl mx-auto p-8">
      <h1 className="text-3xl font-bold mb-6">QuickPoll</h1>
      <ul className="space-y-2">
        {polls.map(p => (
          <li key={p.id}>
            <Link href={`/${p.id}`} className="text-blue-600 hover:underline">
              {p.title}
            </Link>
          </li>
        ))}
      </ul>
    </main>
  );
}
```

Stap 4 — Detail-page

app/poll/[id]/page.tsx:

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

export default async function PollPage({
  params,
}: { params: Promise<{ id: string }> }) {
  const { id } = await params;
  const poll = getPoll(id);
  if (!poll) notFound();
  return (
    <main className="max-w-md mx-auto p-8">
      <h1 className="text-2xl font-bold mb-4">{poll.title}</h1>
      <VoteForm poll={poll} />
    </main>
  );
}
```

Stap 5 — VoteForm (Client Component)

components/VoteForm.tsx:

```

"use client";
import { useState } from "react";

export function VoteForm({ poll }) {
  const [votes, setVotes] = useState(poll.options);
  const [voted, setVoted] = useState(false);

  function vote(optionId: string) {
    setVotes(prev => prev.map(o =>
      o.id === optionId ? { ...o, votes: o.votes + 1 } : o
    ));
    setVoted(true);
  }

  return (
    <div className="space-y-2">
      {votes.map(o => (
        <button
          key={o.id}
          onClick={() => vote(o.id)}
          disabled={voted}
          className="w-full text-left p-3 border rounded hover:bg-gray-50 disabled:opacity-50">
            <span>{o.text}</span>
            <span className="ml-auto float-right text-sm text-gray-500">
              {o.votes}
            </span>
          </button>
        </div>
      ))}
      {voted && <p className="text-green-600 text-sm">Bedankt voor je stem!</p>}
    </div>
  );
}

```

Eisen

- ☐ Next.js project draait lokaal
- ☐ Homepage toont lijst polls
- ☐ Klik op poll → detail-pagina
- ☐ Voting werkt lokaal (visuele update)
- ☐ not-found.tsx voor onbekende poll IDs
- ☐ Tailwind styling — leesbaar

Bonus

- Voeg een 2e poll toe in lib/data.ts
- Maak een app/loading.tsx met een mooi skeleton

-
- Voeg basic styling polish toe (hover effects, transitions)

Klaar?

Volgende les: Server Actions + form handling + voorbereiding op Supabase. Daarna Les 7 = echte database.