Les 6: QuickPoll

Van Scratch tot Werkend

De hele app, stap voor stap

NOVI Hogeschool

Les 6 van 18

Overzicht van de Les

Stap 0: Project aanmaken (recap)

Stap 1: Types & Data (recap)

Stap 2: Layout & Homepage (recap)

Stap 3: GET API Route (recap)

Stap 4: POST Vote Route **(nieuw)**

Stap 5: Poll Detail Pagina **(nieuw)**

Stap 6: VoteForm Component **(nieuw)**

Stap 7: Loading, Error & Not-Found **(nieuw)**

Stap 8: Middleware **(nieuw)**

Bonus: Nieuwe Poll Aanmaken

Stap 0: Project Aanmaken

Maak een nieuw Next.js project aan met de volgende commando's:

```
npx create-next-app@latest quickpoll
```

Kies deze opties:

- TypeScript: **Yes**
- ESLint: **Yes**
- Tailwind CSS: **Yes**
- src/ directory: **Yes**
- App Router: **Yes**
- Import alias (@/*): **Yes**

Start de development server:

```
cd quickpoll && npm run dev
```

De app is nu beschikbaar op **http://localhost:3000**

Stap 1: Types & Data

Maak twee bestanden aan voor types en mock-data:

src/types/index.ts

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

src/lib/data.ts

```
import type { Poll } from "@/types";

let polls: Poll[] = [
{
id: "1",
question: "Wat is je favoriete programmeer taal?",
options: ["JavaScript", "Python", "TypeScript", "Rust"],
votes: [45, 32, 28, 15],
},
{
id: "2",
question: "Hoe veel uur slaap krijg je per nacht?",
options: ["< 6 uur", "6-8 uur", "8+ uur"],
votes: [12, 68, 35],
},
{
id: "3",
question: "Welke framework gebruik je het meest?",
options: ["React", "Vue", "Svelte", "Angular"],
votes: [89, 34, 12, 8],
},
];

let nextId = 4;

export function getPolls(): Poll[] {
return polls;
}

export function getPollById(id: string): Poll | undefined {
return polls.find((poll) => poll.id === id);
}

export function votePoll(
pollId: string,
optionIndex: number
): Poll | undefined {
const poll = polls.find((p) => p.id === pollId);
if (!poll || optionIndex < 0 ||
optionIndex >= poll.options.length)
```

```
return undefined;
poll.votes[optionIndex]++;
return poll;
}
```

Stap 2: Layout & Homepage

src/app/layout.tsx — Root layout met navbar:

```
import type { Metadata } from "next";
import Link from "next/link";
import "./globals.css";

export const metadata: Metadata = {
title: "QuickPoll",
description: "Een snelle polling app met Next.js",
};

export default function RootLayout({
children,
}: Readonly<{ children: React.ReactNode }>) {
return (
<html lang="nl">
<body className="bg-gray-50 min-h-screen flex flex-col">
<nav className="bg-white shadow-sm border-b">
<div className="container mx-auto px-4 py-4">
<Link href="/" className="text-2xl font-bold
text-purple-600">QuickPoll</Link>
</div>
</nav>
<main className="flex-1">{children}</main>
<footer className="bg-white border-t mt-12">
<div className="container mx-auto px-4 py-6">
© 2026 QuickPoll
</div>
</footer>
</body>
</html>
);
}
```

src/app/page.tsx — Homepage met poll kaarten:

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

export default function Home() {
const polls = getPolls();
return (
<div className="container mx-auto py-12 px-4">
<h1 className="text-4xl font-bold">QuickPoll</h1>
<div className="grid grid-cols-1 md:grid-cols-3 gap-6">
{polls.map((poll) => (
<Link key={poll.id} href={`/poll/${poll.id}`}>
<div className="bg-white rounded-lg border p-6">
<h2 className="text-lg font-semibold">
{poll.question}
</h2>
</div>
</Link>
```

```
))}
</div>
</div>
);
}
```

Stap 3: GET API Route

Maak een API route om polls op te halen:

src/app/api/polls/[id]/route.ts

```
import { NextResponse } from "next/server";
import { getPollById } from "@/lib/data";

export async function GET(
request: Request,
{ params }: { params: Promise<{ id: string }> }
) {
const { id } = await params;
const poll = getPollById(id);

if (!poll) {
return NextResponse.json(
{ error: "Poll not found" },
{ status: 404 }
);
}

return NextResponse.json(poll);
}
```

Testen:

Open in je browser: **http://localhost:3000/api/polls/1**

Je ziet de JSON van poll 1.

Stap 4: POST Vote Route

Maak een POST route om stemmen toe te voegen:

src/app/api/polls/[id]/vote/route.ts

```
import { NextResponse } from "next/server";
import { votePoll } from "@/lib/data";

interface RouteParams {
params: Promise<{ id: string }>;
}

interface VoteBody {
optionIndex: number;
}

export async function POST(
request: Request,
{ params }: RouteParams
): Promise<NextResponse> {
const { id } = await params;
const body: VoteBody = await request.json();

if (typeof body.optionIndex !== "number") {
return NextResponse.json(
{ error: "optionIndex is verplicht" },
{ status: 400 }
);
}

const updatedPoll = votePoll(id, body.optionIndex);
if (!updatedPoll) {
return NextResponse.json(
{ error: "Poll niet gevonden" },
{ status: 404 }
);
}

return NextResponse.json(updatedPoll);
}
```

Testen met console:

```
fetch('/api/polls/1/vote', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ optionIndex: 0 })
}).then(r => r.json()).then(console.log);
```

Stap 5: Poll Detail Pagina

Maak de detailpagina met generateMetadata en notFound():

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

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

interface PageProps { params: Promise<{ id: string }>; }

export async function generateMetadata(
{ params }: PageProps
): Promise<Metadata> {
const { id } = await params;
const poll = getPollById(id);
if (!poll) return { title: "Poll niet gevonden" };
return { title: `${poll.question} — QuickPoll` };
}

export default async function PollPage({ params }: PageProps) {
const { id } = await params;
const poll = getPollById(id);
if (!poll) notFound();

return (
<div className="max-w-2xl mx-auto py-12 px-4">
<h1 className="text-2xl font-bold mb-6">
{poll.question}
</h1>
<VoteForm poll={poll} />
</div>
);
}
```

Stap 6: VoteForm Component — State & Logic

Maak een interactieve stemcomponent met client-side logica:

src/components/VoteForm.tsx — Deel 1: Imports en State

```
"use client";
import { useState } from "react";
import type { Poll } from "@/types";

interface VoteFormProps {
poll: Poll;
}

export default function VoteForm({ poll }: VoteFormProps) {
const [selectedOption, setSelectedOption] = useState<number | null>(null);
const [hasVoted, setHasVoted] = useState<boolean>(false);
const [isSubmitting, setIsSubmitting] = useState<boolean>(false);
const [currentPoll, setCurrentPoll] = useState<Poll>(poll);

const totalVotes = currentPoll.votes.reduce(
(sum, v) => sum + v, 0
);

function getPercentage(votes: number): number {
if (totalVotes === 0) return 0;
return Math.round((votes / totalVotes) * 100);
}
```

Handler: async handleVote()

```
async function handleVote(): Promise<void> {
if (selectedOption === null || isSubmitting) return;
setIsSubmitting(true);

const response = await fetch(
`/api/polls/${currentPoll.id}/vote`,
{
method: "POST",
headers: { "Content-Type": "application/json" },
body: JSON.stringify({ optionIndex: selectedOption }),
}
);

if (response.ok) {
const updatedPoll: Poll = await response.json();
setCurrentPoll(updatedPoll);
setHasVoted(true);
}
setIsSubmitting(false);
}
```

Stap 6: VoteForm Component — UI & JSX

Return JSX: Rendering options en buttons

```
return (
<div className="space-y-3">
{currentPoll.options.map((option, index) => {
const percentage = getPercentage(
currentPoll.votes[index]
);
const isSelected = selectedOption === index;
return (
<button
key={index}
onClick={() => !hasVoted &&
setSelectedOption(index)}
disabled={hasVoted}
className={`w-full text-left p-4 rounded-lg
border-2 transition-all relative overflow-hidden
${hasVoted ? "border-gray-200 cursor-default"
: isSelected ? "border-purple-500
bg-purple-50"
: "border-gray-200 hover:border-purple-300
cursor-pointer"
}`}>
{hasVoted && (
<div
className="absolute inset-0 bg-purple-100
transition-all duration-500"
style={{ width: `${percentage}%` }}>
</div>
)}
<div className="relative flex justify-between">
<span className="font-medium">{option}</span>
{hasVoted && (
<span className="text-purple-700">
{percentage}%
</span>
)}
</div>
</button>
);
})}
```

```
{!hasVoted && (
<button onClick={handleVote}
disabled={selectedOption === null}
className="w-full bg-purple-600 text-white py-3">
{isSubmitting ? "Bezig..." : "Stem!"}</button>
)}
```

```
{hasVoted && (
<p className="text-center text-green-600 mt-4">
Bedankt! Totaal: {totalVotes} stemmen
```

```
</p>
)}
</div>
);
}
```

Stap 7: Loading, Error & Not-Found

Next.js biedt speciale bestanden voor UI-states:

src/app/loading.tsx — Skeleton loader:

```
export default function Loading() {
return (
<div className="max-w-2xl mx-auto py-12 px-4">
<div className="animate-pulse space-y-4">
<div className="h-8 bg-gray-200 rounded w-3/4">
</div>
{[...Array(3)].map((_, i) => (
<div key={i} className="h-12 bg-gray-200
rounded"></div>
))}
</div>
</div>
);
}
```

src/app/error.tsx — Error boundary (use client!):

```
"use client";

export default function Error({
error, reset
}: { error: Error; reset: () => void; }) {
return (
<div className="max-w-2xl mx-auto py-12 px-4">
<h1 className="text-2xl font-bold text-red-600">
Er ging iets mis!
</h1>
<p className="text-gray-600 mt-2">{error.message}</p>
<button onClick={reset}
className="bg-purple-600 text-white px-4 py-2 mt-4">
Probeer opnieuw
</button>
</div>
);
}
```

src/app/not-found.tsx — 404 pagina:

```
import Link from "next/link";

export default function NotFound() {
return (
<div className="max-w-2xl mx-auto py-12 px-4 text-center">
<h1 className="text-4xl font-bold text-gray-900">
404
</h1>
<p className="text-gray-600 mt-2">
```

```
Poll niet gevonden
</p>
<Link href="/" className="text-purple-600 mt-4 block">
Terug naar home
</Link>
</div>
);
}
```

Stap 8: Middleware

Middleware voert code uit **voordat** requests worden verwerkt. Perfect voor logging!

src/middleware.ts — In de src/ root, NIET in een folder!

```
import { NextResponse } from "next/server";
import type { NextRequest } from "next/server";

export function middleware(request: NextRequest)
: NextResponse {
const start = Date.now();
console.log(`[${request.method}]
${request.nextUrl.pathname}`);

const response = NextResponse.next();
response.headers.set(
"x-request-time",
String(Date.now() - start)
);
return response;
}

export const config = {
matcher: ["/api/:path*", "/poll/:path*"],
};
```

Wat gebeurt er?

- Logt alle requests naar /api en /poll naar de console
- Voegt een custom header toe met response time
- Check de server logs om de output te zien!

Bonus: Nieuwe Poll Aanmaken

Voeg functionaliteit toe om nieuwe polls te creëren:

Stappen:

1. POST Handler: Maak **src/app/api/polls/route.ts** met een POST functie die:

• Een JSON body (question, options) verwacht

• Een nieuwe Poll met unieke ID aanmaakt

• De poll toevoegt aan de polls array

• De nieuwe poll als JSON teruggeeft

2. Create Page: Maak **src/app/create/page.tsx** met:

• Een form met input fields voor question

• Een field voor opties (bijv. textarea of multiple inputs)

• Een submit button die naar de POST handler fetch't

• Na success: redirect naar de nieuwe poll met useRouter

3. Link naar create: Voeg een link toe in layout of homepage

• <Link href="/create">Nieuwe Poll</Link>

Tips:

- Zorg dat options een non-empty array is
- Initialiseer votes met [0, 0, 0, ...] (één per option)
- Gebruik UUID of incrementing ID voor unieke IDs
- Valideer inputs in je API!

Troubleshooting & Hulp

Veelgestelde Problemen:

"ReferenceError: fetch is not defined"

Vergeten client component? Zet "use client"; bovenaan.

"Cannot read property 'params' of undefined"

Vergeten: const { id } = await params; (Next.js 15!)

"Module not found: @/types"

Check je tsconfig.json alias: "@/*": ["./src/*"]

CSS werkt niet

Tailwind classes zijn case-sensitive, check speling.

API geeft 404

Controleer: folder structure, params destructuring, export statements.

Styles niet zichtbaar

Zorg dat imports in layout.tsx staan, en ./globals.css bevat Tailwind directives.

Debug-tips:

- ✓ Check de **terminal** voor TypeScript/build errors
- ✓ DevTools **Network tab**: See API response status
- ✓ DevTools **Console**: Look for JavaScript errors
- ✓ Zet **console.log()** in je routes/components
- ✓ Refresh browser en clear cache (Cmd+Shift+R / Ctrl+Shift+R)
- ✓ Restart dev server: Ctrl+C en npm run dev

HULP NODIG?

- • Controleer de **Next.js officiële docs**: https://nextjs.org/docs
- • Bekijk de **Tailwind CSS docs**: https://tailwindcss.com
- • Kijk je **bestandsstructuur** na (hoofdletters tellen!)
- • Vraag een **medestudent** om hulp
- • Neem contact op met de **docent**