

Les 17 — Externe APIs in diepte

OAuth, webhooks, paid APIs — productie-patronen

Vak: AI-Assisted Development
Vervolg op: Les 16 — MCP
Volgende les: Les 18 — Supabase Auth + RLS

Leerdoelen

- Drie soorten externe API authenticatie kennen
- OAuth flow begrijpen + Auth.js inzetten
- Webhooks ontvangen + signature verifiëren
- Stripe Checkout productie-pattern implementeren
- Resend transactional email integreren
- Retry-logic met exponential backoff
- Productie checklist voor externe API integratie

Inhoud

1. Beyond simple fetch
2. Drie soorten authenticatie
3. OAuth in Next.js
4. Webhooks — wat en waarom
5. Signature verification
6. Stripe Checkout integratie
7. Resend transactional email
8. Rate limiting + retry
9. Productie checklist
10. Bronnen

1 Beyond simple fetch

In Les 15 deden we PokeAPI — simpel. Productie is anders:

- **Authentication** — key in header of OAuth token
- **Webhooks** — externe service belt JOUW app terug
- **Idempotency** — zelfde event 2x = OK
- **Rate limiting** — max X calls per minuut
- **Error handling** — retry of fail fast?
- **Cost monitoring** — paid APIs lopen op
- **Security** — secrets, signatures, validation

2 Drie soorten authenticatie

API key in header

```
const res = await fetch("https://api.openai.com/v1/...", {
  headers: {
    "Authorization": `Bearer ${process.env.OPENAI_API_KEY}`,
  },
});
```

OAuth flow

Voor user-authenticatie. Gebruiker logt in via externe service, jouw server krijgt access token.

Webhook signatures

Verificatie van inkomende calls. Voorbeelden: Stripe, GitHub, Resend.

3

OAuth in Next.js

De flow (7 stappen):

1. User klikt "Login met GitHub"
2. Redirect naar `github.com/login/oauth/authorize?client_id=...`
3. User logt in, geeft toestemming
4. GitHub redirect terug met `?code=xyz`
5. JOUW server: POST naar token endpoint met code
6. Krijgt `access_token` terug
7. JOUW server zet sessie cookie

Auth.js (NextAuth) setup

```
pnpm add next-auth@beta

// auth.ts
import NextAuth from "next-auth";
import GitHub from "next-auth/providers/github";

export const { handlers, auth, signIn, signOut } = NextAuth({
  providers: [GitHub],
});
```

Sessie ophalen

```
import { auth } from "@/auth";
const session = await auth();
if (!session?.user) return <p>Niet ingelogd</p>;
```

4 Webhooks — wat en waarom

Webhook = HTTP endpoint dat door externe service wordt aangeroepen wanneer er iets gebeurt.

```
Stripe processes payment
-> Stripe sends POST to your-app.com/api/webhook/stripe
    with JSON body: { type: 'checkout.session.completed', data: {...} }
-> Your server: update database, send email, etc.
```

Lokaal testen met Stripe CLI

```
stripe listen --forward-to localhost:3000/api/webhook/stripe
```

5 Signature verification

Waarom

Zonder verify kan iedereen fake events sturen. Een fraudeur stuurt fake 'betaling completed' — jouw server geeft premium aan iemand die niets heeft betaald.

```
import { headers } from "next/headers";
import Stripe from "stripe";

export async function POST(req: Request) {
  const body = await req.text(); // RAW body!
  const sig = (await headers()).get("stripe-signature")!;

  let event;
  try {
    event = stripe.webhooks.constructEvent(
      body, sig, process.env.STRIPE_WEBHOOK_SECRET!
    );
  } catch (err) {
    return new Response("Bad signature", { status: 400 });
  }

  // event is verified, veilig om te gebruiken
  return Response.json({ received: true });
}
```

Cruciaal: req.text(), niet req.json(). Signature wordt over raw bytes berekend.

Idempotency

Wat als Stripe een event 2x stuurt? Sla event.id op in DB, check voor handle.

6

Stripe Checkout integratie

```
pnpm add stripe
```

```
// .env.local
STRIPE_SECRET_KEY=sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...
```

```
// app/api/checkout/route.ts
import Stripe from "stripe";
const stripe = new Stripe(process.env.STRIPE_SECRET_KEY!);

export async function POST(req: Request) {
  const { priceId, email } = await req.json();
  const session = await stripe.checkout.sessions.create({
    mode: "subscription",
    line_items: [{ price: priceId, quantity: 1 }],
    customer_email: email,
    success_url: `${process.env.APP_URL}/success?session_id={CHECKOUT_SESSION_ID}`,
    cancel_url: `${process.env.APP_URL}/pricing`,
  });
  return Response.json({ url: session.url });
}
```

Test cards

- 4242 4242 4242 4242 — succesvolle betaling
- 4000 0000 0000 0002 — geweigerd
- 4000 0025 0000 3155 — 3D Secure

7 Resend transactional email

```
pnpm add resend react-email
```

```
// .env.local  
RESEND_API_KEY=re_...
```

Email als React component

```
// emails/WelcomeEmail.tsx  
import { Html, Heading, Text } from "@react-email/components";  
  
export function WelcomeEmail({ name }: { name: string }) {  
  return (  
    <Html>  
      <Heading>Welkom {name}!</Heading>  
      <Text>Bedankt voor je aanmelding.</Text>  
    </Html>  
  );  
}
```

Verzenden

```
import { Resend } from "resend";  
const resend = new Resend(process.env.RESEND_API_KEY!);  
  
await resend.emails.send({  
  from: "Acme <onboarding@resend.dev>",  
  to: "user@example.com",  
  subject: "Welkom!",  
  react: WelcomeEmail({ name: "Tim" }),  
});
```

Demo: onboarding@resend.dev. Productie: eigen domain.

8 Rate limiting + retry patterns

Exponential backoff retry

```
async function retry<T>(  
  fn: () => Promise<T>,  
  attempts = 3,  
  baseDelayMs = 1000  
) : Promise<T> {  
  for (let i = 0; i < attempts; i++) {  
    try { return await fn(); }  
    catch (err) {  
      if (i === attempts - 1) throw err;  
      const delay = baseDelayMs * Math.pow(2, i); // 1s, 2s, 4s  
      await new Promise(r => setTimeout(r, delay));  
    }  
  }  
  throw new Error("unreachable");  
}
```

Wanneer NIET retryen

- 4xx errors — jouw fout, geen retry helpt
- POST endpoints zonder idempotency
- Vereist user-input correctie

Rate limit eigen endpoints (Upstash)

```
import { Ratelimit } from "@upstash/ratelimit";  
import { Redis } from "@upstash/redis";  
  
const ratelimit = new Ratelimit({  
  redis: Redis.fromEnv(),  
  limiter: Ratelimit.slidingWindow(10, "10s"),  
});  
  
const { success } = await ratelimit.limit(ip);  
if (!success) return new Response("Too many", { status: 429 });
```

9 Productie checklist

Voor je live gaat:

- **API keys:** test-keys voor preview, prod-keys voor prod, scoping per env
- **Webhooks:** signature verify altijd, idempotency check, log payloads
- **Errors:** exponential backoff, rate limit eigen endpoints, monitor

-
- **Security:** HTTPS only, geen secrets in client, Zod validation
 - **Cost:** dashboards monitoren, budget alerts, cache responses

10 Bronnen

- Stripe Checkout — docs.stripe.com/checkout
- Stripe webhooks — docs.stripe.com/webhooks
- Stripe CLI — docs.stripe.com/stripe-cli
- Resend — resend.com/docs
- React Email — react.email
- Auth.js — authjs.dev
- Upstash Ratelimit — upstash.com/docs/oss/sdks/ts/ratelimit