

Les 10 — Lesstof

Auth-flow + Row Level Security verdieping

Vak:	AI-Assisted Development
Klas:	Klas A (3 uur fysiek)
Vervolg op:	Les 9 — Supabase Auth: eerste login flow
Volgende les:	Les 11 — Vercel AI SDK

Vervolg op Les 9. We bouwen de auth-flow uit tot productie-kwaliteit en zetten RLS goed in. Aan het einde is je QuickPoll-app multi-user veilig: elke gebruiker kan alleen zijn eigen polls bewerken en verwijderen, anderen kunnen meelezen en stemmen.

1. Recap Les 9

Aan het einde van Les 9 had je:

- `signUp`, `signInWithPassword`, `signOut` server actions
- Navbar met sessie-status
- Middleware die de sessie ververs
- Eén basale RLS-policy op `polls`

Vandaag gaan we daar veel verder mee. Vier onderwerpen: de drie auth-methodes in detail, sessies onder de motorkap, RLS in detail, en beschermde routes via twee patronen.

2. De drie auth-methodes

2.1 E-mail + wachtwoord

Wat we in Les 9 gebruikten. Vereist e-mailverificatie in productie. Code is simpel, UX is klassiek.

2.2 Magic link

Geen wachtwoord, gebruiker krijgt een link in zijn e-mail.

```
await supabase.auth.signInWithOtp({
  email: "user@example.com",
  options: { emailRedirectTo: "https://app.com/auth/callback" }
});
```

Voordeel: gebruiker hoeft geen wachtwoord te onthouden, geen "wachtwoord vergeten"-flow. Nadeel: gebruiker moet altijd zijn e-mail openen.

2.3 OAuth (Google, GitHub, Discord, etc.)

```
await supabase.auth.signInWithOAuth({
  provider: "google",
  options: { redirectTo: "https://app.com/auth/callback" }
});
```

Voordeel: één klik. Nadeel: gebruiker moet account hebben bij die provider.

Wanneer welke?

- B2B SaaS → e-mail + wachtwoord of magic link
- Consumer app → OAuth (Google/Apple)
- Internal tools → magic link
- High-security → e-mail + wachtwoord + MFA

3. Hoe een sessie écht werkt

Een Supabase-sessie bestaat uit twee tokens:

Token	Levensduur	Gebruik
Access token (JWT)	1 uur	Elke API-call
Refresh token	30 dagen	Nieuwe access token aanvragen

De flow:

1. Login → server stuurt beide tokens terug, opgeslagen in cookies
2. Elke request stuurt het access token mee
3. Na 1 uur: access token verlopen, refresh token genereert een nieuwe
4. Na 30 dagen of na logout: opnieuw inloggen

@supabase/ssr regelt stap 3 automatisch in de middleware.

Wat zit er in een JWT?

```
{
  "sub": "abc-uuid-van-gebruiker",
  "email": "user@example.com",
  "role": "authenticated",
  "exp": 1734567890
}
```

sub is de user-ID. Postgres leest dit uit voor `auth.uid()` in je RLS-polities.

4. Server- vs. client-componenten

In Next.js 16 App Router heb je twee plekken waar je auth gebruikt:

Server component (default)

```
import { createSupabaseServerClient } from "@lib/supabase-server";

export default async function Page() {
  const supabase = await createSupabaseServerClient();
  const { data: { user } } = await supabase.auth.getUser();
  if (!user) redirect("/login");
  return <div>Hallo {user.email}</div>;
}
```

Client component ("use client")

```
"use client";
import { createBrowserClient } from "@supabase/ssr";

const supabase = createBrowserClient(URL, KEY);

export function LogoutButton() {
  return <button onClick={() => supabase.auth.signOut()}>Logout</button>;
}
```

Vuistregel: doe alles in server components tenzij je interactie nodig hebt (een knop, een form-event).

5. Row Level Security in detail

RLS draait in Postgres zelf. Je server action vraagt data op, en Postgres beslist wat je terugkrijgt — niet je code.

`using` vs. `with check`

Clausive	Gebruikt bij	Wat het doet
using	SELECT, UPDATE, DELETE	Welke rijen zie/raak je?
with check	INSERT, UPDATE	Mag de NIEUWE waarde bestaan?

Voorbeeld: voorkomen dat een gebruiker een poll claimt die van iemand anders is:

```
create policy "owner can update"
on polls for update
using (auth.uid() = user_id)
with check (auth.uid() = user_id);
```

using zegt: je mag de rij bewerken als je de eigenaar bent. with check zegt: na de update moet je nog steeds de eigenaar zijn (anders kun je user_id overschrijven).

Vier basis-polities voor je `polls` tabel

```
-- 1. Iedereen mag polls lezen
create policy "polls public select"
on polls for select
using (true);

-- 2. Ingelogde gebruikers mogen polls aanmaken
create policy "auth users can insert"
on polls for insert
with check (auth.uid() is not null and auth.uid() = user_id);

-- 3. Alleen eigenaar mag updaten
create policy "owner can update"
on polls for update
using (auth.uid() = user_id)
with check (auth.uid() = user_id);

-- 4. Alleen eigenaar mag verwijderen
create policy "owner can delete"
on polls for delete
using (auth.uid() = user_id);
```

6. Beschermd routes: twee patronen

Patroon 1 — middleware redirect

Voor hele route-groepen (bijv. alle /app/*):

```
// middleware.ts
const { data: { user } } = await supabase.auth.getUser();
if (!user && request.nextUrl.pathname.startsWith("/create")) {
  return NextResponse.redirect(new URL("/login", request.url));
}
```

Snel, gecentraliseerd. Nadeel: je kunt geen page-specifieke UX tonen ("log in om dit te zien" met preview).

Patroon 2 — server component check

Voor pagina-specifieke logica:

```
export default async function MyPollsPage() {
  const supabase = await createSupabaseServerClient();
  const { data: { user } } = await supabase.auth.getUser();
  if (!user) return <LoginPrompt />;

  const { data: polls } = await supabase
    .from("polls")
    .select("*")
    .eq("user_id", user.id);

  return <PollList polls={polls} />;
}
```

Gebruik **patroon 1** voor "deze hele sectie is privé", **patroon 2** voor "deze pagina werkt anders per user".

7. Belangrijk: dubbele veiligheid

Best practice = check op twee niveaus:

1. **Middleware** zegt: anonieme bezoeker mag deze route niet zien
2. **RLS** zegt: zelfs als de bezoeker er komt, ziet/wijzigt hij niks dat niet van hem is

Waarom? Als ooit één laag faalt (bug, vergeten check, security update), is de andere er nog. Diepte-verdediging.

8. Veelvoorkomende fouten

Fout	Oplossing
Policy werkt niet	RLS aan? alter table X enable row level security;
auth.uid() is null in policy	Cookies komen niet door — middleware check
Anoniem account kan ineens INSERTen	with check ontbreekt op insert-policy
User kan andermans rij stelen via update	with check op update vergeten

Fout	Oplossing
Service-role-key in browser	NOOIT. Gebruik anon-key in browser, service-role alleen op server

9. Checklist einde les

- 4 RLS-policies actief op polls (select, insert, update, delete)
- 2 RLS-policies actief op options (select-iedereen, insert-eigenaar-van-poll)
- "My Polls" pagina toont alleen polls van ingelogde gebruiker
- Edit + Delete knoppen werken alleen voor eigenaar
- Middleware-bescherming op /create en /my-polls
- Werkende auth-flow getest met 2 verschillende accounts

10. Vooruitblik

In Les 11 verlaten we QuickPoll en starten met de Vercel AI SDK. Je app moet op dat moment auth + RLS op productie-niveau hebben — dat is de basis waar je AI-features veilig op kunt bouwen.