

Les 09 — Lesstof

signUp, signIn, signOut, Navbar, RLS intro

Vak:	AI-Assisted Development
Klas:	Klas A (3 uur fysiek)
Vervolg op:	Les 8 — Van In-Memory naar Supabase
Volgende les:	Les 10 — Supabase Auth & RLS verdieping

Eerste echte authenticatie voor QuickPoll. Aan het einde van deze les heeft je app signup, login, logout én een werkende Navbar die de sessie-status toont. We sluiten af met een eerste blik op Row Level Security (RLS) als opmaat voor Les 10.

1. Waarom Supabase Auth?

Je hebt drie opties als je een webapp wil beveiligen:

- Zelf bouwen** — password-hashing, sessie-cookies, e-mailverificatie, OAuth-callbacks, password-reset, rate-limiting. Een fulltime baan.
- Auth0 / Clerk / Cognito** — externe service, prima maar duur en je data ligt elders.
- Supabase Auth** — zit naast je database, gratis voor onze schaal, werkt out-of-the-box met RLS.

Voor deze cursus kiezen we Supabase Auth omdat het een directe lijn heeft naar je Postgres-database. Dat maakt RLS (autorisatie op database-niveau) mogelijk.

Authenticatie vs. Autorisatie

- Authenticatie:** wie ben je? (login + wachtwoord, magic link, OAuth)
- Autorisatie:** wat mag je? (RLS-policies, role checks)

Vandaag focussen we op authenticatie. Autorisatie (RLS) gaan we in detail behandelen in Les 10.

2. Supabase Auth in vogelvlucht

Supabase Auth ondersteunt veel methodes. De drie meest gebruikte:

Methode	UX	Geschikt voor
E-mail + wachtwoord	Klassiek	Apps met geregistreerde gebruikers

Methode	UX	Geschikt voor
Magic link	Klik op link in e-mail	Lage drempel, geen wachtwoord onthouden
OAuth (Google/GitHub)	Eén klik	Snelle signup voor consumer apps

Vandaag gebruiken we e-mail + wachtwoord. Magic link en OAuth zijn beide twee config-regels extra.

De vier kern-functies

```
// 1. signUp – nieuw account
await supabase.auth.signUp({ email, password });

// 2. signInWithPassword – inloggen
await supabase.auth.signInWithPassword({ email, password });

// 3. signOut – uitloggen
await supabase.auth.signOut();

// 4. getUser – wie is ingelogd?
const { data: { user } } = await supabase.auth.getUser();
```

Dat is het. Geen complexer API. De rest doet Supabase voor je: cookies, JWT-refresh, e-mailverificatie.

3. Sessies en `@supabase/ssr`

Een sessie is een geldig JWT-token. Supabase bewaart dit in cookies. Het pakket `@supabase/ssr` zorgt dat zowel je server-components als je middleware automatisch toegang hebben tot deze cookie.

Server-client setup

```
// lib/supabase-server.ts
import { createServerClient } from "@supabase/ssr";
import { cookies } from "next/headers";

export async function createSupabaseServerClient() {
  const cookieStore = await cookies();
  return createServerClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
    {
      cookies: {
        getAll() { return cookieStore.getAll(); },
        setAll(cookiesToSet) {
          try {
            cookiesToSet.forEach(({ name, value, options }) =>
              cookieStore.set(name, value, options)
            );
          } catch { /* server component, geen mutation */ }
        },
      },
    }
  );
}
```

Belangrijk: in een server component **mag** je geen cookies muteren. Daarom de try/catch. De middleware doet de cookie-refresh.

4. Server actions voor signUp en signIn

```
// app/signup/actions.ts
"use server";
import { createSupabaseServerClient } from "@lib/supabase-server";
import { redirect } from "next/navigation";

export async function signUpAction(formData: FormData) {
  const supabase = await createSupabaseServerClient();
  const { error } = await supabase.auth.signUp({
    email: formData.get("email") as string,
    password: formData.get("password") as string,
  });
  if (error) return { error: error.message };
  redirect("/");
}
```

Dezelfde structuur voor signInAction met signInWithPassword. Server actions houden de wachtwoord-flow uit de browser.

5. Navbar met sessie-status

De Navbar is een server component. Hij vraagt de huidige user op en toont "Inloggen" of "Uitloggen" op basis daarvan.

```
// components/Navbar.tsx
import { createSupabaseServerClient } from "@lib/supabase-server";
import Link from "next/link";
import { signOutAction } from "@app/signout/actions";

export async function Navbar() {
  const supabase = await createSupabaseServerClient();
  const { data: { user } } = await supabase.auth.getUser();

  return (
    <nav className="flex gap-4 p-4 border-b">
      <Link href="/">QuickPoll</Link>
      <div className="ml-auto">
        {user ? (
          <form action={signOutAction}>
            <span className="mr-3">{user.email}</span>
            <button type="submit">Uitloggen</button>
          </form>
        ) : (
          <Link href="/login">Inloggen</Link>
        )}
      </div>
    </nav>
  );
}
```

6. Middleware voor sessie-refresh

Zonder middleware verloopt je sessie na 1 uur. Met deze middleware blijft je token automatisch actueel.

```
// middleware.ts
import { type NextRequest } from "next/server";
import { updateSession } from "@lib/supabase-middleware";

export async function middleware(request: NextRequest) {
  return await updateSession(request);
}

export const config = {
  matcher: ["/(?!_next/static|_next/image|favicon.ico).*"],
};
```

De updateSession-helper (template uit Supabase docs) leest de cookie, verlengt het token en blokkeert optioneel routes voor anonieme bezoekers.

7. Eerste blik op Row Level Security (RLS)

Tot nu toe was je polls-tabel "open": iedereen kan alles lezen en schrijven. Dat moet beter. RLS = autorisatie op database-niveau, ingebakken in Postgres.

```
-- Zet RLS aan op polls
alter table polls enable row level security;

-- Iedereen mag alle polls lezen
create policy "polls zijn publiek leesbaar"
  on polls for select
  using (true);

-- Alleen ingelogde gebruikers mogen polls maken
create policy "ingelogde gebruikers mogen polls maken"
  on polls for insert
  with check (auth.uid() is not null);
```

Belangrijk: `auth.uid()` is een Postgres-functie die Supabase invult op basis van het JWT van de huidige sessie. Geen JWT? Dan is `auth.uid()` `null`.

In Les 10 gaan we dit veel verder uitbouwen.

8. Checklist einde les

Aan het einde van deze les heb je:

- `@supabase/ssr` geïnstalleerd en geconfigureerd
- Server-client helper (`lib/supabase-server.ts`)
- `/signup` route met server action
- `/login` route met server action
- `/signout` server action
- Navbar met sessie-status
- Middleware die routes beschermt
- RLS aan voor polls (twee policies)

Als één van deze ontbreekt: pak de Live-Coding-Guide erbij en loop het stappenplan door.

9. Veelgemaakte fouten

Fout	Oplossing
<code>auth.uid()</code> is null ondanks login	Middleware niet geconfigureerd — sessie wordt niet ververs
"Email not confirmed"	Zet "Disable Email Confirmations" aan in dashboard voor lokaal
Navbar updatet niet na login	Vergeten <code>revalidatePath("/")</code> of <code>redirect()</code> aan eind van action
<code>cookies()</code> errors in server component	Niet proberen te <code>set()</code> in een server component zelf

10. Volgende les

Les 10 gaat veel dieper op RLS in: meerdere policies, `auth.uid()` overal, en het verschil tussen `using` (read) en `with check` (write). We voegen ook eigenaarschap toe: "alleen ik mag mijn polls bewerken".