Les 9
Supabase Authentication

signUp, signIn, signOut, Middleware, Navbar, RLS

Generated: 31 March 2026

DEEL 1: Uitleg Auth Concepten

Wat is Auth?

Authenticatie (WHO) antwoord op: wie ben jij? Email + password, Supabase verifies, JWT token given, user object (email, id, created_at).

Autorisatie (WHAT) antwoord op: wat mag je doen? Wie mag polls maken? Dit regelen we met RLS policies.

Supabase Auth Features:

• Email/password signup & signin

• Session management (cookies)

• JWT tokens

• Password reset

• Multi-factor auth

• OAuth (Google, GitHub, etc.)

Vier Core Functies

signUp

```
const { error } = await supabase.auth.signUp({ email, password });
```

signInWithPassword

```
const { error } = await supabase.auth.signInWithPassword({ email, password });
```

signOut

```
await supabase.auth.signOut();
```

getUser

```
const { data: { user } } = await supabase.auth.getUser();
```

Server vs Browser Client

Server Client (@supabase/ssr): Secure, via cookies, Node.js, used in middleware/Navbar/SSR.

Browser Client (@supabase/ssr): Less secure, via localStorage, React/client-side, used in login forms.

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 { }
                },
            },
        }
    );
}
```

lib/supabase-browser.ts

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

export function createSupabaseBrowserClient() {
    return createBrowserClient(
        process.env.NEXT_PUBLIC_SUPABASE_URL!,
        process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
    );
}
```

Middleware & Auth Callback

Middleware runs on every request. It refreshes the session token so you don't get logged out.

middleware.ts

```
import { createServerClient } from "@supabase/ssr";
import { NextResponse, type NextRequest } from "next/server";

export async function middleware(request: NextRequest) {
    let supabaseResponse = NextResponse.next({ request });
    const supabase = createServerClient(
        process.env.NEXT_PUBLIC_SUPABASE_URL!,
        process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
        {
            cookies: {
                getAll() { return request.cookies.getAll(); },
                setAll(cookiesToSet) {
                    cookiesToSet.forEach(({ name, value }) => request.cookies.set(name, value));
                    supabaseResponse = NextResponse.next({ request });
                    cookiesToSet.forEach(({ name, value, options }) =>
                        supabaseResponse.cookies.set(name, value, options)
                    );
                },
            },
        }
    );
    await supabase.auth.getUser();
    return supabaseResponse;
}

export const config = {
    matcher: ["/((?!_next/static|_next/image|favicon.ico|.*\.(?:svg|png|jpg|jpeg|gif|webp)$).*)"],
};
```

app/auth/callback/route.ts

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

export async function GET(request: Request) {
    const { searchParams, origin } = new URL(request.url);
    const code = searchParams.get("code");
    if (code) {
        const supabase = await createSupabaseServerClient();
        await supabase.auth.exchangeCodeForSession(code);
    }
    return NextResponse.redirect(origin);
}
```

DEEL 2: Reference Code — Zelf Doen

app/signup/page.tsx

```
'use client'
import { createSupabaseBrowserClient } from "@/lib/supabase-browser";
import { useRouter } from "next/navigation";
import { useState } from "react";
import Link from "next/link";

export default function SignUp() {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [message, setMessage] = useState("");
    const [loading, setLoading] = useState(false);
    const router = useRouter();
    const supabase = createSupabaseBrowserClient();

    const handleSignUp = async (e: React.FormEvent) => {
        e.preventDefault();
        setLoading(true);
        setMessage("");
        const { error } = await supabase.auth.signUp({ email, password });
        if (error) { setMessage(error.message); }
        else { setMessage("Account aangemaakt!"); router.push("/login"); }
        setLoading(false);
    };

    return (
        <div className="w-full max-w-md mx-auto p-6">
            <h1 className="text-2xl font-bold mb-6">Registreren</h1>
            <form onSubmit={handleSignUp} className="space-y-4">
                <div>
                    <label className="block text-sm font-medium mb-1">Email</label>
                    <input type="email" value={email} onChange={(e) => setEmail(e.target.value)}
                        className="w-full p-2 border rounded" required />
                </div>
                <div>
                    <label className="block text-sm font-medium mb-1">Wachtwoord</label>
                    <input type="password" value={password} onChange={(e) => setPassword(e.target.value)}
                        className="w-full p-2 border rounded" minLength={6} required />
                </div>
                <button type="submit" disabled={loading}
                    className="w-full bg-blue-600 text-white p-2 rounded hover:bg-blue-700 disabled:opacity-50">
                    {loading ? "Bezig..." : "Registreren"}
                </button>
            </form>
            {message && <p className="mt-4 text-sm text-center">{message}</p>}
            <p className="mt-4 text-sm text-center">
                Al een account? <Link href="/login" className="text-blue-600 hover:underline">Inloggen</Link>
            </p>
        </div>
    );
}
```

app/login/page.tsx

```
'use client'
import { createSupabaseBrowserClient } from "@/lib/supabase-browser";
import { useRouter } from "next/navigation";
import { useState } from "react";
```

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

export default function Login() {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [message, setMessage] = useState("");
    const [loading, setLoading] = useState(false);
    const router = useRouter();
    const supabase = createSupabaseBrowserClient();

    const handleLogin = async (e: React.FormEvent) => {
        e.preventDefault();
        setLoading(true);
        setMessage("");
        const { error } = await supabase.auth.signInWithPassword({ email, password });
        if (error) { setMessage(error.message); }
        else { router.push("/"); router.refresh(); }
        setLoading(false);
    };

    return (
        <div className="w-full max-w-md mx-auto p-6">
            <h1 className="text-2xl font-bold mb-6">Inloggen</h1>
            <form onSubmit={handleLogin} className="space-y-4">
                <div>
                    <label className="block text-sm font-medium mb-1">Email</label>
                    <input type="email" value={email} onChange={(e) => setEmail(e.target.value)}
                        className="w-full p-2 border rounded" required />
                </div>
                <div>
                    <label className="block text-sm font-medium mb-1">Wachtwoord</label>
                    <input type="password" value={password} onChange={(e) => setPassword(e.target.value)}
                        className="w-full p-2 border rounded" required />
                </div>
                <button type="submit" disabled={loading}
                    className="w-full bg-blue-600 text-white p-2 rounded hover:bg-blue-700 disabled:opacity-50">
                    {loading ? "Bezig..." : "Inloggen"}
                </button>
            </form>
            {message && <p className="mt-4 text-sm text-red-600 text-center">{message}</p>}
            <p className="mt-4 text-sm text-center">
                Nog geen account? <Link href="/signup" className="text-blue-600 hover:underline">Registreren</Link>
            </p>
        </div>
    );
}
```

components/LogoutButton.tsx

```
'use client'
import { createSupabaseBrowserClient } from "@/lib/supabase-browser";
import { useRouter } from "next/navigation";

export function LogoutButton() {
    const router = useRouter();
    const supabase = createSupabaseBrowserClient();
    const handleLogout = async () => {
        await supabase.auth.signOut();
        router.push("/");
        router.refresh();
    };
    return (
        <button onClick={handleLogout} className="text-sm text-gray-600 hover:text-gray-900">
            Uitloggen
        </button>
    );
}
```

components/Navbar.tsx

```
import Link from "next/link";
import { createSupabaseServerClient } from "@/lib/supabase-server";
import { LogoutButton } from "./LogoutButton";

export async function Navbar() {
    const supabase = await createSupabaseServerClient();
    const { data: { user } } = await supabase.auth.getUser();
    return (
        <nav className="w-full border-b p-4 flex justify-between items-center">
            <Link href="/" className="text-xl font-bold">QuickPoll</Link>
            <div className="flex items-center gap-4">
                {user ? (
                    <>
                        <span className="text-sm text-gray-600">{user.email}</span>
                        <LogoutButton />
                    </>
                ) : (
                    <>
                        <Link href="/login" className="text-sm hover:underline">Inloggen</Link>
                        <Link href="/signup" className="text-sm bg-blue-600 text-white px-3 py-1 rounded hover:bg-b
                    </>
                )}
            </div>
        </nav>
    );
}
```

app/layout.tsx (updated)

```
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import "./globals.css";
import { Navbar } from "@/components/Navbar";

const geistSans = Geist({ variable: "--font-geist-sans", subsets: ["latin"] });
const geistMono = Geist_Mono({ variable: "--font-geist-mono", subsets: ["latin"] });

export const metadata: Metadata = { title: "QuickPoll", description: "Stem op je favoriete opties" };

export default function RootLayout({ children }: Readonly<{ children: React.ReactNode }>) {
```

```
    return (
        <html lang="nl">
            <body className={`${geistSans.variable} ${geistMono.variable} antialiased`}>
                <Navbar />
                {children}
            </body>
        </html>
    );
}
```

Row Level Security (RLS)

RLS policies enforce who can do what on the database. Run this SQL in Supabase dashboard → SQL Editor.

```
ALTER TABLE polls ENABLE ROW LEVEL SECURITY;

CREATE POLICY "polls_select_all" ON polls
    FOR SELECT USING (true);

CREATE POLICY "polls_insert_authenticated" ON polls
    FOR INSERT WITH CHECK (auth.uid() IS NOT NULL);

CREATE POLICY "polls_update_owner" ON polls
    FOR UPDATE USING (auth.uid() = created_by);
```

Wat dit doet:

- Iedereen kan polls zien (SELECT)
- Alleen ingelogde users kunnen polls maken (INSERT)
- Alleen de maker kan hun poll updaten (UPDATE)

Huiswerk

Verplicht (Les 10):

1. Profiel pagina — app/profile/page.tsx

Toon user.email, user.id, user.created_at

2. Maker tonen bij poll

Voeg created_by uuid kolom toe polls tabel

Update INSERT in /create: created_by: user.id

Toon 'Gemaakt door: [email]' op homepage

Bonus (optioneel):

• Google OAuth — Supabase dashboard → Auth → Providers → Google

• Password reset — signInWithPassword flow

Preview volgende les:

Les 10: Profiel pagina, Google OAuth, Vercel deployment