Les 10 — Hands-on Werkboek

Supabase Auth & RLS stap voor stap

Vak: AI-Assisted Development | NOVI Hogeschool Utrecht | Docent: Tim

Wat ga je bouwen?

- Supabase Auth opzetten (client, server, middleware)
- Login & registratie pagina bouwen
- Sessie management & beschermde routes
- Row Level Security (RLS) policies schrijven

Benodigdheden

- Starter project: **poll-app-starter.zip**
- Supabase project (URL + anon key uit het dashboard)
- Cursor of andere code editor
- Terminal

Hands-on Blok 1: Auth Opzetten

Stap 1: Starter project openen

Pak de **poll-app-starter.zip** uit en open in Cursor:

```
cd poll-app-starter
npm install
```

Stap 2: Environment variabelen

Maak een **.env.local** bestand aan in de root van je project:

```
NEXT_PUBLIC_SUPABASE_URL=https://jouw-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJhbGci...
```

> **Tip:** Je vindt deze waarden in het Supabase Dashboard onder Settings > API.

Stap 3: Packages installeren

```
npm install @supabase/ssr @supabase/supabase-js
```

Stap 4: Browser client aanmaken

Maak het bestand **src/lib/supabase/client.ts**:

```
import { createBrowserClient } from '@supabase/ssr'

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

> **Info:** De browser client gebruik je in Client Components — interactieve componenten met useState en onClick.

Stap 5: Server client aanmaken

Maak het bestand **src/lib/supabase/server.ts**:

```
import { createServerClient } from '@supabase/ssr'
import { cookies } from 'next/headers'

export async function createClient() {
  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) {
          cookiesToSet.forEach(({ name, value, options }) =>
            cookieStore.set(name, value, options)
          )
        },
      },
    }
  )
}
```

Info: De server client gebruik je in Server Components en API routes. Het verschil: deze kan cookies lezen EN schrijven.

Stap 6: Middleware aanmaken

Maak het bestand **src/middleware.ts** (LET OP: in de src/ map, NIET in src/app/):

```
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)
          )
        },
      },
    }
  )

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

  if (
    !user &&
    !request.nextUrl.pathname.startsWith('/login') &&
    !request.nextUrl.pathname.startsWith('/auth')
  ) {
    const url = request.nextUrl.clone()
    url.pathname = '/login'
    return NextResponse.redirect(url)
  }

  return supabaseResponse
}

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

Let op: De middleware MOET in src/middleware.ts staan, NIET in src/app/middleware.ts!

Stap 7: Auth callback route aanmaken

Maak de map en het bestand **src/app/auth/callback/route.ts**:

```
mkdir -p src/app/auth/callback
```

```
import { createClient } from '@/lib/supabase/server'
import { NextResponse } from 'next/server'

export async function GET(request: Request) {
  const { searchParams, origin } = new URL(request.url)
  const code = searchParams.get('code')

  if (code) {
    const supabase = await createClient()
    await supabase.auth.exchangeCodeForSession(code)
  }

  return NextResponse.redirect(origin)
}
```

> **Info:** Deze route verwerkt magic link codes. Als een gebruiker op een magic link klikt, komt hij hier terecht.

Stap 8: Email provider aanzetten

Ga naar het Supabase Dashboard > **Authentication** > **Providers** > **Email**. Zorg dat Email aangevinkt is. Zet voor development **"Confirm email" UIT** (toggle).

> **Tip:** In productie laat je email bevestiging aan. Voor nu schakelen we het uit zodat je direct kunt registreren.

Hands-on Blok 2: Login & Registratie

Stap 9: Login pagina aanmaken

Maak **src/app/login/page.tsx**:

```
'use client'

import { createClient } from '@/lib/supabase/client'
import { useRouter } from 'next/navigation'
import { useState } from 'react'

export default function LoginPage() {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const [loading, setLoading] = useState(false)
  const [message, setMessage] = useState('')
  const [isSignUp, setIsSignUp] = useState(false)
  const router = useRouter()
  const supabase = createClient()

  const handleEmailLogin = async (e: React.FormEvent) => {
    e.preventDefault()
    setLoading(true)
    setMessage('')

    if (isSignUp) {
      const { error } = await supabase.auth.signUp({
        email,
        password,
      })
      if (error) {
        setMessage(error.message)
      } else {
        setMessage('Check je email voor een bevestigingslink!')
      }
    } else {
      const { error } = await supabase.auth.signInWithPassword({
        email,
        password,
      })
      if (error) {
        setMessage(error.message)
      } else {
        router.push('/')
        router.refresh()
      }
    }

    setLoading(false)
  }

  const handleMagicLink = async () => {
    if (!email) {
      setMessage('Vul eerst je email in')
      return
    }
    setLoading(true)
    setMessage('')

    const { error } = await supabase.auth.signInWithOtp({
      email,
    })

    if (error) {
      setMessage(error.message)
    } else {
      setMessage('Check je email voor een magic link!')
    }

    setLoading(false)
  }
```

(vervolg login pagina — return statement):

```
  return (
    <div className="min-h-screen flex items-center
         justify-center bg-gray-50">
      <div className="max-w-md w-full space-y-8 p-8
           bg-white rounded-xl shadow-lg">
        <div>
          <h2 className="text-center text-3xl font-bold
              text-gray-900">
            {isSignUp ? 'Account aanmaken' : 'Inloggen'}
          </h2>
          <p className="mt-2 text-center text-sm
             text-gray-600">
            Poll App
          </p>
        </div>

        <form onSubmit={handleEmailLogin}
              className="mt-8 space-y-6">
          <div className="space-y-4">
            <div>
              <label htmlFor="email"
                className="block text-sm font-medium
                  text-gray-700">Email</label>
              <input id="email" type="email" required
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                className="mt-1 block w-full px-3 py-2
                  border border-gray-300 rounded-md"
                placeholder="jouw@email.nl" />
            </div>
            <div>
              <label htmlFor="password"
                className="block text-sm font-medium
                  text-gray-700">Wachtwoord</label>
              <input id="password" type="password" required
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                className="mt-1 block w-full px-3 py-2
                  border border-gray-300 rounded-md"
                placeholder="Minimaal 6 tekens" />
            </div>
          </div>
```

(vervolg login pagina — buttons & toggle):

```
          {message && (
            <div className="text-sm text-center
                 text-red-600">{message}</div>
          )}

          <div className="space-y-3">
            <button type="submit" disabled={loading}
              className="w-full flex justify-center py-2 px-4
                border border-transparent rounded-md text-sm
                font-medium text-white bg-blue-600
                hover:bg-blue-700 disabled:opacity-50">
              {loading ? 'Laden...'
                : isSignUp ? 'Registreren' : 'Inloggen'}
            </button>

            <button type="button" onClick={handleMagicLink}
              disabled={loading}
              className="w-full flex justify-center py-2 px-4
                border border-gray-300 rounded-md text-sm
                font-medium text-gray-700 bg-white
                hover:bg-gray-50 disabled:opacity-50">
              Stuur Magic Link
            </button>
          </div>
        </form>

        <div className="text-center">
          <button onClick={() => setIsSignUp(!isSignUp)}
            className="text-sm text-blue-600
              hover:text-blue-500">
            {isSignUp
              ? 'Heb je al een account? Inloggen'
              : 'Nog geen account? Registreren'}
          </button>
        </div>
      </div>
    </div>
  )
}
```

Stap 10: Testen — Registreren & Inloggen

Volg deze stappen om te testen:

1. Start je dev server: **npm run dev**
2. Ga naar http://localhost:3000 — je wordt naar /login gestuurd
3. Klik op "Nog geen account? Registreren"
4. Vul een email en wachtwoord in (minimaal 6 tekens)
5. Klik "Registreren" — je zou nu ingelogd moeten zijn
6. Check in Supabase Dashboard > Authentication > Users of je user verschijnt

Troubleshooting

"Invalid login credentials"	Wachtwoord te kort (min 6 tekens)
"User already registered"	Gebruik een ander emailadres
Redirect loop	Check de matcher config in middleware.ts
404 op /login	Bestand staat op de verkeerde plek

Hands-on Blok 3: Sessie & Beschermde Routes

Stap 11: Navbar component aanmaken

Maak **src/components/Navbar.tsx**:

```
'use client'

import { createClient } from '@/lib/supabase/client'
import { useRouter } from 'next/navigation'
import { useEffect, useState } from 'react'
import type { User } from '@supabase/supabase-js'

export default function Navbar() {
  const [user, setUser] = useState<User | null>(null)
  const router = useRouter()
  const supabase = createClient()

  useEffect(() => {
    const getUser = async () => {
      const { data: { user } } = await supabase.auth.getUser()
      setUser(user)
    }
    getUser()
  }, [])

  const handleSignOut = async () => {
    await supabase.auth.signOut()
    router.push('/login')
    router.refresh()
  }

  return (
    <nav className="bg-white shadow-sm border-b">
      <div className="max-w-7xl mx-auto px-4 sm:px-6
           lg:px-8">
        <div className="flex justify-between h-16
             items-center">
          <div className="flex-shrink-0">
            <h1 className="text-xl font-bold
                text-gray-900">Poll App</h1>
          </div>
          {user && (
            <div className="flex items-center gap-4">
              <span className="text-sm text-gray-600">
                {user.email}
              </span>
              <button onClick={handleSignOut}
                className="text-sm text-red-600
                  hover:text-red-800 font-medium">
                Uitloggen
              </button>
            </div>
          )}
        </div>
      </div>
    </nav>
  )
}
```

Stap 12: Navbar toevoegen aan layout

Open **src/app/layout.tsx** en voeg toe:

```
import Navbar from '@/components/Navbar'
```

En in de body:

```
<body>
  <Navbar />
  <main>{children}</main>
</body>
```

> **Info:** Pas alleen de import en het Navbar component toe. Laat je bestaande styling intact.

Stap 13: Testen

Controleer de volgende punten:

- [ ] Refresh de pagina — je ziet je email in de navbar
- [ ] Klik op "Uitloggen" — je gaat naar /login
- [ ] Probeer naar http://localhost:3000 te gaan — je wordt geredirect naar /login
- [ ] Log opnieuw in — alles werkt weer

Hands-on Blok 4: Row Level Security

Stap 14: RLS inschakelen

Ga naar Supabase Dashboard > **SQL Editor** > New query:

```
-- RLS inschakelen op beide tabellen
ALTER TABLE polls ENABLE ROW LEVEL SECURITY;
ALTER TABLE options ENABLE ROW LEVEL SECURITY;
```

> **Let op:** Na het uitvoeren van deze SQL zijn ALLE operaties geblokkeerd! Je app zal even niet werken tot je policies toevoegt.

Stap 15: Policies voor de polls tabel

```
-- Iedereen mag polls lezen (ook niet-ingelogd)
CREATE POLICY "Polls are viewable by everyone"
ON polls FOR SELECT
USING (true);

-- Alleen ingelogde users mogen polls aanmaken
CREATE POLICY "Authenticated users can create polls"
ON polls FOR INSERT
TO authenticated
WITH CHECK (true);
```

> **Info: TO authenticated** betekent: alleen gebruikers met een geldige sessie. Niet-ingelogde gebruikers hebben de **anon** rol.

Stap 16: Policies voor de options tabel

```
-- Iedereen mag options lezen
CREATE POLICY "Options are viewable by everyone"
ON options FOR SELECT
USING (true);

-- Ingelogde users mogen opties aanmaken
CREATE POLICY "Authenticated users can create options"
ON options FOR INSERT
TO authenticated
WITH CHECK (true);

-- Ingelogde users mogen stemmen (update)
CREATE POLICY "Authenticated users can vote"
ON options FOR UPDATE
TO authenticated
USING (true);
```

Stap 17: Testen — als ingelogde user

Controleer de volgende punten:

- [ ] Refresh je app — polls verschijnen weer
- [ ] Maak een nieuwe poll aan — werkt!
- [ ] Stem op een optie — werkt!

Stap 18: Testen — zonder login

Open een **incognito venster** en ga naar je app.

- [ ] Je wordt geredirect naar /login (middleware beschermt de routes)
- [ ] Als je de middleware tijdelijk uitzet: polls zijn zichtbaar (SELECT = public)
- [ ] Maar poll aanmaken faalt (INSERT = authenticated only)
- [ ] Stemmen faalt ook (UPDATE = authenticated only)

Gefeliciteerd! Je hebt een volledig werkend authenticatie- en autorisatiesysteem gebouwd!

Bestandsoverzicht & Checklist

Aangemaakte bestanden

Bestand	Doel
`src/lib/supabase/client.ts`	Browser client (Client Components)
`src/lib/supabase/server.ts`	Server client (Server Components)
`src/middleware.ts`	Sessie refresh + route bescherming
`src/app/auth/callback/route.ts`	Magic link / OAuth callback
`src/app/login/page.tsx`	Login & registratie pagina
`src/components/Navbar.tsx`	Navigatie met user info + logout

Eindchecklist

- [ ] Alle 6 bestanden aangemaakt
- [ ] .env.local met Supabase URL en anon key
- [ ] Registreren werkt
- [ ] Inloggen werkt
- [ ] Uitloggen werkt
- [ ] Beschermde routes werken (redirect naar /login)
- [ ] RLS ingeschakeld op polls en options
- [ ] 5 policies aangemaakt (2x SELECT, 2x INSERT, 1x UPDATE)
- [ ] App werkt als ingelogd, blokkeert als niet ingelogd

Dit document bevat alle stappen van de hands-on opdracht van Les 10. Gebruik het als werkboek tijdens de les en als naslagwerk bij het huiswerk.