

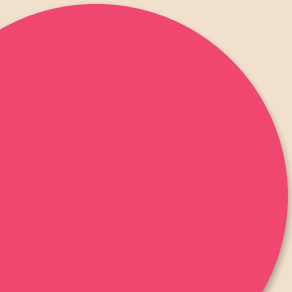


Les 18

Supabase Auth + RLS

Multi-user apps met per-user data isolation — productie ready

(de laatste les)



Terugblik

18 lessen — waar staan we?

Les 1-10

Foundations: Next.js, Supabase basics, Tailwind, Shadcn

Les 11

AI SDK basics + Polderfest

Les 12

Tool Calling

Les 13

Agents

Les 14

RAG + Embeddings

Les 15

Cursor + Vercel deploy + CI

Les 16

MCP servers

Les 17

Externe APIs in diepte

Les 18

Vandaag: Supabase Auth + RLS

Vandaag: alles wat je hebt geleerd, multi-user maken. Klaar voor eindopdracht.

Planning

Vandaag — 180 minuten

Terugblik + waarom auth	10 min
Theorie: Supabase Auth basics	20 min
Theorie: RLS — wat en waarom	20 min
Theorie: Session in Next.js	15 min
LIVE DEMO 1 — Auth + magic link	25 min
LIVE DEMO 2 — Middleware + protected	20 min
Pauze	15 min
LIVE DEMO 3 — RLS policies multi-user	30 min
LIVE DEMO 4 — Social login (GitHub)	15 min
Eindopdracht recap	5 min
Vragen + afsluiting cyclus	15 min

Theorie 1

Waarom auth + multi-user

Wat je niet kunt zonder auth:

- Per-user data (mijn tasks, jouw tasks)
- Privacy (alleen jij ziet je data)
- Audit trail (wie deed wat wanneer)
- Subscriptions (wie betaalde, wie niet)

Auth = 'wie ben jij?'

Authz = 'wat mag jij?'

In Supabase: Auth + RLS doen beide.

Geïntegreerd, simpel, productie-ready.

Voor je eindopdracht: must have als app meerdere users heeft.

Theorie 1

Supabase Auth — drie methoden

Email + password

Klassiek. Signup met email + wachtwoord.

Magic link

Passwordless. Link in mail → klik → ingelogd.

Social (OAuth)

Login met Google, GitHub, Discord, etc.

Voor demo: magic link. Snel, geen wachtwoord-hassle, productie-grade.

```
await supabase.auth.signInWithOtp({  
  email: "user@example.com",  
});
```

Theorie 2

RLS — Row Level Security

Zonder RLS:

```
SELECT * FROM tasks;  
-- Returns ALL tasks of ALL users  
-- → disaster
```

Met RLS:

```
SELECT * FROM tasks WHERE  
  user_id = auth.uid();  
-- Returns alleen JOUW tasks
```

RLS = filter dat Postgres ALTIJD toepast op queries, ongeacht waar query vandaan komt.

```
alter table tasks enable row level security;  
  
create policy "users see own tasks" on tasks  
  for select using (auth.uid() = user_id);  
  
create policy "users insert own tasks" on tasks  
  for insert with check (auth.uid() = user_id);
```

auth.uid() — Supabase helper, geeft user UUID of NULL.

Theorie 3

Session in Next.js App Router

```
pnpm add @supabase/supabase-js @supabase/ssr
```

Vier files:

- | | | |
|---|-------------------------------------|----------------------------------|
| 1 | utils/supabase/client.ts | voor Client Components |
| 2 | utils/supabase/server.ts | voor Server Components + actions |
| 3 | utils/supabase/middleware.ts | refresh sessie elke request |
| 4 | middleware.ts (root) | gebruikt #3 |

Supabase heeft een template voor deze setup.

Kopiëren werkt. Snappen waarom is bonus.

Waarom: cookies sync browser ↔ server, sessies verlopen, refresh nodig.

Vandaag bouwen we

Multi-user Tasks app

Doel: simpele tasks-app, ieder z'n eigen lijst, geen vermenging.

Features:

/login

magic link form + GitHub button

/

task list (alleen ingelogd)

Add/complete/delete

RLS bepaalt scope

/logout

logout button

Schema:

```
create table tasks (  
  id bigserial primary key,  
  user_id uuid not null  
    references auth.users(id) default auth.uid(),  
  text text not null,  
  done boolean default false,  
  created_at timestamp default now()  
);
```


LIVE DEMO 1

Auth setup + magic link — ~25 min

- 1 pnpm create + add @supabase/ssr
- 2 Supabase: Auth → Email → enable Magic Links
- 3 utils/supabase/server.ts + client.ts (template)
- 4 app/login/page.tsx — magic link form
- 5 app/auth/callback/route.ts — exchange code voor sessie
- 6 Test: email → mail komt → klik → ingelogd

Magic link = passwordless login

```
await supabase.auth.signInWithOtp({  
  email, options: { emailRedirectTo: `${origin}/auth/callback` }  
});
```

LIVE DEMO 2

Middleware + protected routes — ~20 min

```
// middleware.ts (root)
import { type NextRequest } from "next/server";
import { updateSession } from "@utils/supabase/middleware";

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

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

```
// app/page.tsx – Server Component check
import { createClient } from "@utils/supabase/server";
import { redirect } from "next/navigation";

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

Test: niet ingelogd → /login. Ingelogd → home.



Pauze

15 minuten

(laatste pauze van de cyclus)

LIVE DEMO 3

RLS policies multi-user — ~30 min

```
create table tasks (  
  id bigserial primary key,  
  user_id uuid not null references auth.users(id) default auth.uid(),  
  text text not null, done boolean default false  
);  
  
alter table tasks enable row level security;  
  
create policy "select own" on tasks for select using (auth.uid() = user_id);  
create policy "insert own" on tasks for insert with check (auth.uid() = user_id);  
create policy "update own" on tasks for update using (auth.uid() = user_id);  
create policy "delete own" on tasks for delete using (auth.uid() = user_id);
```

Demo:

- Add task — werkt voor mij, niet zichtbaar voor ander account
- Tweede browser/incognito met ander account — andere data
- Probeer alle tasks ophalen — RLS stopt het

LIVE DEMO 4

Social login (GitHub) — ~15 min

- 1 GitHub OAuth App aanmaken (Settings → Developer settings)
- 2 Callback URL: `https://[project].supabase.co/auth/v1/callback`
- 3 Supabase Dashboard → Auth → Providers → GitHub → enable
- 4 Plak Client ID + Client Secret
- 5 Login button: `signInWithOAuth({ provider: 'github' })`
- 6 Test: klik → GitHub → authorize → ingelogd
- 7 Database: zelfde RLS werkt — GitHub user heeft eigen `auth.uid()`

Eindopdracht

Hoe combineer je alles?

Checklist:

- ✓ Next.js + TypeScript + Tailwind (Les 1-10)
- ✓ Supabase Postgres + Auth + RLS (Les 10, 18)
- ✓ Vercel AI SDK met Tool Calling of Agent (Les 11-13)
- ✓ RAG (Les 14, optioneel)
- ✓ Cursor + GitHub + Vercel deploy (Les 15)
- ✓ Externe API met patterns uit Les 17
- ✓ Deployed met productie URL

Ideale eindopdracht:

- Multi-user app waar elke user eigen data heeft
- AI feature met tool calling of agent (kern, geen afterthought)
- Minstens 1 externe API integratie
- Tijd: ~40-60 uur thuis, verspreid over 4-6 weken

Afsluiting

Bedankt en succes!

Wat we samen gedaan hebben:

- ✓ 18 lessen, ~54 uur klassikaal
- ✓ Van localhost naar productie
- ✓ Van simple fetch naar agents + RAG
- ✓ Van mock-data naar multi-user productie-apps

Wat je nu zelfstandig kunt:

- Next.js app van scratch naar productie
- AI SDK + tool calling + agents
- Externe APIs met OAuth + webhooks
- Multi-user apps met auth + RLS
- Code reviews + CI/CD pipelines

Bedankt voor jullie inzet. Veel succes met de eindopdracht.

Vragen?