Next.js with Supabase Project Structure

Supabase integration with auth, database, and real-time subscriptions.

#nextjs #react #typescript #supabase #postgres

PNG PDF Copy Prompt

Project Directory

```
my-app/
  app/
    layout.tsx
    page.tsx
    globals.css
    (auth)/ Auth pages
      login/
        page.tsx
      signup/
        page.tsx
      callback/ OAuth callback
        route.ts
    (protected)/
      layout.tsx
      dashboard/
        page.tsx
      account/
        page.tsx
  lib/
    supabase/ Supabase clients
      client.ts Browser client
      server.ts Server Componen…
      middleware.ts Middleware clie…
  components/
    auth/
      login-form.tsx
      signup-form.tsx
      oauth-buttons.tsx
    ui/
      button.tsx
      input.tsx
  types/
    database.types.ts Generated from …
  middleware.ts Refresh session…
  next.config.js
  package.json
  tsconfig.json
  .env.local SUPABASE_URL, S…
  .env.example
  .gitignore
```

Why This Structure?

Supabase provides auth, PostgreSQL database, and real-time out of the box. This structure uses SSR-compatible clients: browser client for interactivity, server client for Server Components, and middleware client for session refresh. Types are auto-generated from your database schema.

Key Directories

lib/supabase/ - Three clients for different contexts

lib/supabase/client.ts - Browser— `createBrowserClient()`

lib/supabase/server.ts - RSC— `createServerClient()`

app/(auth)/callback/ - OAuth code exchange route

types/database.types.ts - Generated DB types

Server Client

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

export function createClient() {
  const cookieStore = cookies()
  return createServerClient(url, key, { cookies: ... })
}
```

Getting Started

```
npm install @supabase/supabase-js @supabase/ssr
npx supabase gen types typescript > types/database.types.t
npm run dev
```

When To Use This

- Rapid prototyping with hosted backend
- Apps needing auth + database quickly
- Projects using PostgreSQL with RLS
- Real-time features (chat, live updates)

Trade-offs

Vendor lock-in - Supabase-specific APIs and hosting

Three client files - SSR requires separate clients per context

Row Level Security - Must configure RLS policies in Supabase