Сделаем полноценный Header.tsx, который сам понимает:

* если пользователь **не вошёл** → показывает кнопку **Войти**;
* если **вошёл** → показывает **Выйти** и ссылки на **Главная** и **Админка**.

**📂 src/components/Header.tsx**

// src/components/Header.tsx

"use client"

import Link from "next/link"

import { useRouter } from "next/navigation"

import { useEffect, useState } from "react"

import { supabase } from "@/lib/supabaseClient"

export default function Header() {

const router = useRouter()

const [user, setUser] = useState<any>(null)

useEffect(() => {

supabase.auth.getUser().then(({ data }) => {

setUser(data?.user ?? null)

})

const { data: listener } = supabase.auth.onAuthStateChange(

(\_event, session) => {

setUser(session?.user ?? null)

}

)

return () => {

listener.subscription.unsubscribe()

}

}, [])

const handleLogout = async () => {

await supabase.auth.signOut()

router.push("/")

}

return (

<header className="bg-gray-900 text-white shadow-md">

<div className="max-w-6xl mx-auto px-4 py-3 flex items-center justify-between">

{/\* Лого / Название \*/}

<div className="text-xl font-bold">

<Link href="/">Блог</Link>

</div>

{/\* Меню \*/}

<nav className="flex space-x-6 text-sm font-medium">

<Link

href="/"

className="hover:text-yellow-400 transition-colors"

>

Главная

</Link>

<Link

href="/admin"

className="hover:text-yellow-400 transition-colors"

>

Админка

</Link>

</nav>

{/\* Войти/Выйти \*/}

<div>

{user ? (

<button

onClick={handleLogout}

className="bg-red-600 hover:bg-red-700 text-white px-4 py-1 rounded-lg transition-colors"

>

Выйти

</button>

) : (

<Link

href="/auth"

className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-1 rounded-lg transition-colors"

>

Войти

</Link>

)}

</div>

</div>

</header>

)

}

**✅ Как работает:**

* При загрузке проверяет supabase.auth.getSession().
* Если пользователь есть → показывается **Админка** + кнопка **Выйти**.
* Если нет → показывается только **Главная** и кнопка **Войти**.

**1. Создаём AuthContext**

src/context/AuthContext.tsx

"use client"

import { createContext, useContext, useEffect, useState } from "react"

import { supabase } from "@/lib/supabase"

import { Session, User } from "@supabase/supabase-js"

type AuthContextType = {

session: Session | null

user: User | null

loading: boolean

}

const AuthContext = createContext<AuthContextType>({

session: null,

user: null,

loading: true,

})

export function AuthProvider({ children }: { children: React.ReactNode }) {

const [session, setSession] = useState<Session | null>(null)

const [loading, setLoading] = useState(true)

useEffect(() => {

// Первичная загрузка сессии

supabase.auth.getSession().then(({ data }) => {

setSession(data.session)

setLoading(false)

})

// Подписка на изменения сессии

const { data: subscription } = supabase.auth.onAuthStateChange(

(\_event, newSession) => {

setSession(newSession)

}

)

return () => {

subscription.subscription.unsubscribe()

}

}, [])

return (

<AuthContext.Provider

value={{

session,

user: session?.user ?? null,

loading,

}}

>

{children}

</AuthContext.Provider>

)

}

export function useAuth() {

return useContext(AuthContext)

}

**2. Подключаем в layout.tsx**

src/app/layout.tsx

import "./globals.css"

import { AuthProvider } from "@/context/AuthContext"

export default function RootLayout({

children,

}: {

children: React.ReactNode

}) {

return (

<html lang="ru">

<body>

<AuthProvider>

{children}

</AuthProvider>

</body>

</html>

)

}

**3. Используем в Header**

Теперь Header может легко брать данные о пользователе:

"use client"

import Link from "next/link"

import { useRouter } from "next/navigation"

import { supabase } from "@/lib/supabase"

import { useAuth } from "@/context/AuthContext"

export default function Header() {

const router = useRouter()

const { user, loading } = useAuth()

const handleLogout = async () => {

await supabase.auth.signOut()

router.push("/")

}

return (

<header className="bg-gray-900 text-white shadow-md">

<div className="max-w-6xl mx-auto px-4 py-3 flex items-center justify-between">

<div className="text-xl font-bold">

<Link href="/">Блог</Link>

</div>

<nav className="flex space-x-6 text-sm font-medium">

<Link href="/" className="hover:text-yellow-400">

Главная

</Link>

<Link href="/admin" className="hover:text-yellow-400">

Админка

</Link>

</nav>

{!loading && (

<div>

{user ? (

<button

onClick={handleLogout}

className="bg-red-600 hover:bg-red-700 text-white px-4 py-1 rounded-lg"

>

Выйти

</button>

) : (

<Link

href="/auth"

className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-1 rounded-lg"

>

Войти

</Link>

)}

</div>

)}

</div>

</header>

)

}

Теперь у тебя:

* AuthProvider → глобальное состояние.
* useAuth() → доступ к user, session, loading в любом компоненте.
* Header стал намного чище.

Заполняем таблицу

insert into posts (title, content, image\_url, author)

values

('Первая запись', 'Это тестовый контент для первого поста.', null, 'admin@example.com'),

('Новости проекта', 'Мы начинаем разработку нового блога на Next.js и Supabase.', null, 'admin@example.com'),

('Обновление системы', 'Добавлена авторизация через Google. Теперь входить стало проще.', null, 'editor@example.com'),

('Советы по Tailwind', 'Tailwind помогает быстро создавать аккуратный UI.', null, 'user1@example.com'),

('Статья о Supabase', 'Supabase — отличная альтернатива Firebase с Postgres внутри.', null, 'admin@example.com'),

('Next.js 14', 'В новой версии Next.js улучшили App Router и серверные компоненты.', null, 'user2@example.com'),

('TypeScript в проектах', 'TypeScript позволяет писать более надежный код.', null, 'editor@example.com'),

('Роутинг в Next.js', 'App Router делает код чище и упрощает навигацию.', null, 'user3@example.com'),

('Почему стоит учить React', 'React — это база для современного фронтенда.', null, 'admin@example.com'),

('Финальный пост', 'Это заключительная запись для теста.', null, 'user4@example.com');