Pregunta:

Por último, ¿podrías detallarme el paso 6?

Respuesta:

¡Claro que sí! Llegamos al último paso, que consiste en unir todo para crear una experiencia de usuario coherente y segura. Este punto se encarga de que la aplicación se comporte como se espera: que pida iniciar sesión primero y que, una vez dentro, presente un entorno de trabajo organizado.

Aquí te detallo cómo implementar el flujo de navegación, las rutas protegidas y la página de inicio. 🚪

### **Paso 6.1: Crear una Página Principal o "Dashboard"**

Después de iniciar sesión, el usuario no debería aterrizar directamente en la tabla de usuarios, sino en una página principal que actúe como centro de operaciones. Desde aquí, podrá navegar a las distintas secciones.

Primero, vamos a organizar las rutas. Crearemos una carpeta dashboard dentro de src/app. Todas las páginas que requieran autenticación vivirán aquí.

1. Crea la carpeta: mi-app-gestion/src/app/dashboard.
2. Crea la página principal del dashboard: mi-app-gestion/src/app/dashboard/page.tsx.

#### mi-app-gestion/src/app/dashboard/page.tsx

TypeScript

'use client';  
  
import { Typography, Container, Paper, Box } from '@mui/material';  
import { useAuth } from '@/context/AuthContext';  
  
export default function DashboardPage() {  
 const { user } = useAuth();  
  
 return (  
 <Container maxWidth="lg">  
 <Box sx={{ my: 4 }}>  
 <Typography variant="h4" component="h1" gutterBottom>  
 ¡Bienvenido, {user?.username}!  
 </Typography>  
 <Paper sx={{ p: 2 }}>  
 <Typography variant="body1">  
 Has iniciado sesión correctamente en el panel de gestión.  
 Utiliza el menú de navegación para acceder a las diferentes secciones de la aplicación.  
 </Typography>  
 </Paper>  
 </Box>  
 </Container>  
 );  
}

### **Paso 6.2: Crear un Layout Protegido con Menú de Navegación**

Este es el paso más importante. Crearemos un layout que envolverá a *todas* las páginas dentro de /dashboard. Este layout hará dos cosas:

1. **Proteger la ruta**: Verificará si el usuario está autenticado. Si no lo está, lo redirigirá a la página de /login.
2. **Mostrar un menú persistente**: Tendrá una barra superior y/o un menú lateral para navegar entre secciones.

Crea el fichero de layout para el dashboard: mi-app-gestion/src/app/dashboard/layout.tsx.

#### mi-app-gestion/src/app/dashboard/layout.tsx

TypeScript

'use client';  
  
import { useAuth } from '@/context/AuthContext';  
import { useRouter } from 'next/navigation';  
import { useEffect } from 'react';  
import {   
 Box,   
 AppBar,   
 Toolbar,   
 Typography,   
 Drawer,   
 List,   
 ListItem,   
 ListItemButton,   
 ListItemIcon,   
 ListItemText,   
 CircularProgress,  
 Button  
} from '@mui/material';  
import PeopleIcon from '@mui/icons-material/People';  
import Link from 'next/link';  
  
const drawerWidth = 240;  
  
export default function DashboardLayout({  
 children,  
}: {  
 children: React.ReactNode;  
}) {  
 const { isAuthenticated, isLoading, logout } = useAuth();  
 const router = useRouter();  
  
 useEffect(() => {  
 // Si no está cargando y el usuario no está autenticado, lo echamos al login.  
 if (!isLoading && !isAuthenticated) {  
 router.push('/login');  
 }  
 }, [isLoading, isAuthenticated, router]);  
  
 // Mientras se verifica el estado de autenticación, mostramos un spinner.  
 if (isLoading || !isAuthenticated) {  
 return (  
 <Box sx={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height: '100vh' }}>  
 <CircularProgress />  
 </Box>  
 );  
 }  
   
 const handleLogout = () => {  
 logout();  
 router.push('/login');  
 };  
  
 return (  
 <Box sx={{ display: 'flex' }}>  
 {/\* Barra de Navegación Superior \*/}  
 <AppBar position="fixed" sx={{ zIndex: (theme) => theme.zIndex.drawer + 1 }}>  
 <Toolbar>  
 <Typography variant="h6" noWrap component="div" sx={{ flexGrow: 1 }}>  
 Panel de Gestión  
 </Typography>  
 <Button color="inherit" onClick={handleLogout}>  
 Cerrar Sesión  
 </Button>  
 </Toolbar>  
 </AppBar>  
  
 {/\* Menú Lateral Persistente \*/}  
 <Drawer  
 variant="permanent"  
 sx={{  
 width: drawerWidth,  
 flexShrink: 0,  
 [`& .MuiDrawer-paper`]: { width: drawerWidth, boxSizing: 'border-box' },  
 }}  
 >  
 <Toolbar />  
 <Box sx={{ overflow: 'auto' }}>  
 <List>  
 {/\* Enlace a la gestión de usuarios \*/}  
 <ListItem disablePadding>  
 <ListItemButton component={Link} href="/dashboard/users">  
 <ListItemIcon>  
 <PeopleIcon />  
 </ListItemIcon>  
 <ListItemText primary="Gestión de Usuarios" />  
 </ListItemButton>  
 </ListItem>  
 {/\* Aquí puedes añadir más enlaces a otras secciones \*/}  
 </List>  
 </Box>  
 </Drawer>  
  
 {/\* Contenido Principal de la Página \*/}  
 <Box component="main" sx={{ flexGrow: 1, p: 3 }}>  
 <Toolbar />  
 {children}  
 </Box>  
 </Box>  
 );  
}

### **Paso 6.3: Mover la Gestión de Usuarios a la Ruta Protegida**

Tu página de gestión de usuarios debe estar dentro del área protegida.

1. **Mueve el fichero**: Coge la carpeta users que está en mi-app-gestion/src/app/users y muévela dentro de mi-app-gestion/src/app/dashboard/.
2. La nueva ruta para la gestión de usuarios será ahora http://localhost:3000/dashboard/users.

El fichero page.tsx de los usuarios no necesita más cambios, ya que ahora estará automáticamente envuelto por el DashboardLayout, heredando la protección y el menú de navegación.

### **Paso 6.4: Modificar la Página de Inicio (/) para que actúe como un "Router"**

Finalmente, la página raíz de tu aplicación (/) debe decidir a dónde enviar al usuario.

Modifica el fichero mi-app-gestion/src/app/page.tsx. Este componente será muy simple, ya que su única misión es redirigir.

#### mi-app-gestion/src/app/page.tsx

TypeScript

'use client';  
  
import { useEffect } from 'react';  
import { useRouter } from 'next/navigation';  
import { useAuth } from '@/context/AuthContext';  
import { Box, CircularProgress } from '@mui/material';  
  
export default function HomePage() {  
 const { isAuthenticated, isLoading } = useAuth();  
 const router = useRouter();  
  
 useEffect(() => {  
 // No hagas nada hasta que sepamos si el usuario está logueado o no  
 if (isLoading) {  
 return;  
 }  
  
 if (isAuthenticated) {  
 router.replace('/dashboard'); // Si está logueado, llévalo al dashboard  
 } else {  
 router.replace('/login'); // Si no, a la página de login  
 }  
 }, [isLoading, isAuthenticated, router]);  
  
 // Muestra un spinner de carga a pantalla completa mientras se decide la redirección  
 return (  
 <Box  
 sx={{  
 display: 'flex',  
 justifyContent: 'center',  
 alignItems: 'center',  
 height: '100vh',  
 }}  
 >  
 <CircularProgress />  
 </Box>  
 );  
}

### **Resumen del Flujo Final**

Con estos cambios, has completado la estructura de navegación de una aplicación profesional:

1. Un usuario accede a http://localhost:3000/.
2. El componente HomePage se carga, ve que el usuario no está autenticado y lo redirige a /login.
3. El usuario introduce sus credenciales y hace clic en "Acceder".
4. Si el login es exitoso, el AuthContext se actualiza y el usuario es redirigido a /dashboard.
5. La página /dashboard se carga dentro del DashboardLayout. Como el usuario está autenticado, se le muestra la bienvenida y el menú de navegación.
6. Desde el menú, puede hacer clic en "Gestión de Usuarios" para ir a /dashboard/users y ver la tabla de usuarios.
7. Si en algún momento el usuario cierra la sesión o su token expira, cualquier intento de acceder a una ruta /dashboard/... lo redirigirá automáticamente de vuelta a /login.

¡Felicidades! Has unido todas las piezas para crear una aplicación web funcional, segura y con una estructura de navegación sólida. 🚀