RectaCard/ReceraCard.js

```
import { useState } from "react";

function TituloEImagen({ titulo }) {
  return <h2>{titulo}</h2>;
}

function BotonFavorito({ favorito, onToggle }) {
  return (
    <button onClick={onToggle}>
      {favorito ? "💖 Quitar" : "□ Favorito"}
    </button>
  );
}

export default function RecetaCard() {
  const [favorito, setFavorito] = useState(false);
  const handleToggle = () => setFavorito(!favorito);
  return (
    <div>
      <TituloEImagen titulo="Mi Receta" />
      <BotonFavorito favorito={favorito} onToggle={handleToggle} />
    </div>
  );
}
```

Notificaciones/NoficacionesContext.js

```
import { createContext, useContext, useState } from "react";
const NotiContext = createContext();
export function useNoti() { return useContext(NotiContext); }
export function NotiProvider({ children }) {
  const [activadas, setActivadas] = useState(false);
  return (
    <NotiContext.Provider value={{ activadas, setActivadas }}>
      {children}
    </NotiContext.Provider>
  );
}
```

Notificaciones/PaginaConfiguracion.js

```
import { useNoti } from "./NotificacionesContext";
export default function PaginaConfiguracion() {
  const { activadas, setActivadas } = useNoti();
  return (
    <div>
      <h2>Configuración</h2>
      <button onClick={() => setActivadas(!activadas)}>
        {activadas ? "Desactivar" : "Activar"} notificaciones
      </button>
    </div>
  );
}
```

Notificaciones/PaginaPerfil.js

```
import { useNoti } from "./NotificacionesContext";
export default function PaginaPerfil() {
  const { activadas } = useNoti();
  return (
    <h2>
      {activadas
        ? "¡Tienes notificaciones activadas!"
        : "Notificaciones desactivadas."}
    </h2>
  );
}
```

NotificacionesCentral/BotonCargaInicial.js

```
import { useNotif } from "./NotificationContext";
export default function BotonCargaInicial() {
  const { cargar, loading } = useNotif();
  return <button onClick={cargar} disabled={loading}>
    {loading ? "Cargando..." : "Cargar alertas"}
  </button>;
}
```

NotificacionesCentral/NotificacionContext.js

```
import { createContext, useContext, useState } from "react";
const Ctx = createContext();
export const useNotif = () => useContext(Ctx);
export function NotificationProvider({ children }) {
  const [lista, setLista] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const cargar = async () => {
    try {
      setLoading(true);
      const datos = await new Promise(res =>
        setTimeout(() => res(["Alerta 1", "Alerta 2"]), 1000)
      );
      setLista(datos);
    } catch (e) { setError("Error"); }
    finally { setLoading(false); }
  };
  return (
    <Ctx.Provider value={{ lista, loading, error, cargar }}>
      {children}
    </Ctx.Provider>
  );
}
```

NotificacionesCentral/PanelDeAlertas.js

```
import { useNotif } from "./NotificationContext";
```

```
export default function PanelDeAlertas() {
  const { lista, loading, error } = useNotif();
  if (loading) return <p>Cargando...</p>;
  if (error) return <p>{error}</p>;
  return <ul>{lista.map((a,i)=><li key={i}>{a}</li>)}</ul>;
}
```

RutasProtegidas/AuthContext.js

```
// src/RutasProtegidas/AuthContext.jsx
import React, { createContext, useContext, useState, useEffect } from "react";

const AuthContext = createContext();
export const useAuth = () => useContext(AuthContext);

export function AuthProvider({ children }) {
  // Valores iniciales seguros para evitar "undefined"
  const [isAuthenticated, setIsAuthenticated] = useState(() => {
    try {
      const raw = localStorage.getItem("auth");
      return raw ? JSON.parse(raw).isAuthenticated : false;
    } catch {
      return false;
    }
  });
```

```
const [user, setUser] = useState(() => {
  try {
    const raw = localStorage.getItem("auth");
    return raw ? JSON.parse(raw).user : { role: "publico" };
  } catch {
    return { role: "publico" };
  }
});

// Mantener en localStorage para facilitar pruebas y persistencia simple
useEffect(() => {
  localStorage.setItem("auth", JSON.stringify({ isAuthenticated, user }));
}, [isAuthenticated, user]);

const loginAs = (role = "publico") => {
  setIsAuthenticated(true);
  setUser({ role });
};

const logout = () => {
  setIsAuthenticated(false);
  setUser({ role: "publico" });
};

// Helpers
const canEdit = () => user?.role === "admin" || user?.role === "editor";
```

```
  // hasRole acepta string o array
  const hasRole = (roles) => {
    if (!roles) return true;
    if (Array.isArray(roles)) return roles.includes(user?.role);
    return user?.role === roles;
  };

  return (
    <AuthContext.Provider
      value={{ isAuthenticated, user, loginAs, logout, canEdit, hasRole }}
    >
      {children}
    </AuthContext.Provider>
  );
}
```

RutasProtegidas/RutaPrivada.js

```
// src/RutasProtegidas/RutaPrivada.jsx
import React from "react";
import { Navigate, useLocation } from "react-router-dom";
import { useAuth } from "./AuthContext";
```

```
export default function RutaPrivada({ children, rolRequerido }) {
  const { isAuthenticated, hasRole } = useAuth();
  const location = useLocation();

  // 1) Si no está autenticado -> redirige a /login (y guarda desde dónde vino)
  if (!isAuthenticated) {
    return <Navigate to="/login" state={{ from: location }} replace />;
  }

  // 2) Si hay rol requerido y no coincide -> redirige al home (o a una página "sin permiso")
  if (rolRequerido && !hasRole(rolRequerido)) {
    return <Navigate to="/" replace />;
  }

  // 3) Si pasa las comprobaciones, renderiza children
  return children;
}
```

RutasProtegidas/PanelAdministrador.js

```
// src/RutasProtegidas/PanelAdministrador.jsx
import React from "react";
import BotonEditar from "./BotonEditar";
```

```
import { useAuth } from "./AuthContext";

export default function PanelAdministrador() {
  const { user } = useAuth();
  return (
    <div>
      <h2>Panel de Administración</h2>
      <p>Acceso concedido. Rol actual: <b>{user?.role}</b></p>
      <p>Aquí iría la UI administrativa (solo admin).</p>
      <BotonEditar />
    </div>
  );
}
```

RutasProtegidas/BotonEditar.js

```
// src/RutasProtegidas/BotonEditar.jsx
import React from "react";
import { useAuth } from "./AuthContext";

export default function BotonEditar() {
  const { user } = useAuth();

  // Visible solo para admin o editor
  const visible = user?.role === "admin" || user?.role === "editor";
  if (!visible) return null;

  return (
```

```
    <button onClick={() => alert("Abrir modal de edición (simulado).")}>
      Editar Contenido
    </button>
  );
}
```

RutasProtegidas/Login.js

```
// src/RutasProtegidas/Login.jsx
import React from "react";
import { useNavigate, useLocation } from "react-router-dom";
import { useAuth } from "./AuthContext";

export default function Login() {
  const { loginAs } = useAuth();
  const navigate = useNavigate();
  const location = useLocation();
  const from = location.state?.from?.pathname || "/";

  const doLogin = (role) => {
    loginAs(role);
    // regresar a la ruta que intentó acceder
    navigate(from, { replace: true });
  };
```

```
  return (
    <div>
      <h2>Login (simulado)</h2>
      <p>Selecciona un rol para iniciar sesión:</p>
      <div style={{ display: "flex", gap: 8 }}>
        <button onClick={() => doLogin("admin")}>Iniciar como Admin</button>
        <button onClick={() => doLogin("editor")}>Iniciar como Editor</button>
        <button onClick={() => doLogin("publico")}>Iniciar como
Público</button>
      </div>
    </div>
  );
}
```

PREGUNTAS

1. **Describa cómo implementaría la función (el handler) que se define en el padre (RecetaCard) y se pasa al hijo (BotonFavorito) para que, al hacer clic, el icono del corazón cambie su apariencia en el padre.**

En RecetaCard creo un estado const [favorito,setFavorito]=useState(false) y una función toggleFavorito que hace setFavorito(!favorito). Esa función se pasa como prop a BotonFavorito.

2. **Describa qué mecanismo de persistencia de estado (más allá del simple useState local del componente) necesitaría implementar para que la preferencia de notificaciones establecida en /configuracion sea recordada y accedida por el componente PaginaPerfil al cambiar de ruta. Pista: Considera una solución que haga que el estado "sobreviva" a la navegación.**

Uso un Context (NotiProvider) que guarda notificacionesActivadas y lo leo en PaginaPerfil. Así el estado está en un provider y no se pierde al cambiar de ruta, porque vive fuera del componente.

3. **Describa la secuencia de tres pasos (Acción, Disparo y Actualización) que ocurre desde el momento en que el usuario hace clic en el BotonCargaInicial hasta que la lista de alertas aparece en el PanelDeAlertas, usando la terminología de gestión de estado centralizado (Context/Provider o Dispatch/Action/Reducer).**

Acción: el usuario hace click en BotonCargaInicial.
Disparo: el botón despacha la acción: pone loading=true y llama al servicio API.
Actualización: cuando llega la respuesta, el provider actualiza listaDeAlertas y loading=false; el PanelDeAlertas re-renderiza mostrando las alertas.

4. **Describa cómo crearía un Custom Hook (ej. useAuthGuard) para la RutaPrivada que combine la verificación de autenticación y la verificación de Autorización (Rol). ¿Qué tres piezas de información (al menos) necesitaría este hook para tomar una decisión de redirección o acceso?**

El hook revisaría si el usuario está logueado y si su rol cumple con el requerido. Necesita al menos:

isAuthenticated del contexto,

user.role del usuario,

rolRequerido de la ruta.
Con eso decide si deja entrar o redirige (por ejemplo usando useNavigate).

LINK VIDEO

https://www.youtube.com/watch?v=UeRVe27JZ8U