// React Login Form — Tutorial comentado y visual
// Público: Estudiantes del Propedéutico del Doctorado en Gobierno Electrónico
// Objetivo: implementar un formulario de inicio de sesión atractivo, accesible y listo para conectar a una API.
// Tecnologías: React + Tailwind (asumido disponible en el proyecto). Si no usan Tailwind, pueden reemplazar las clases por CSS propio.

```
import React, { useState } from "react";
import { motion } from "framer-motion"; // Animaciones suaves
import { Eye, EyeOff, Lock, Mail, Loader2, ShieldCheck } from "lucide-react";
// Iconos

// ============================
// 1) Componente principal
// ============================
// - Contiene el layout de la página, la tarjeta del login y el formulario.
// - Incluye ejemplos de validación básica en el cliente.
// - Muestra feedback visual: loading, errores, éxito simulado.
export default function LoginTutorial() {
  // ----------------------
  // 2) Estados del formulario
  // ----------------------
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [showPassword, setShowPassword] = useState(false);
  const [remember, setRemember] = useState(true);

  // Estados de UI
  const [errors, setErrors] = useState({ email: "", password: "" });
  const [isLoading, setIsLoading] = useState(false);
  const [status, setStatus] = useState("idle"); // idle | success | error
  const [message, setMessage] = useState("");

  // ----------------------
  // 3) Validación de campos
  // ----------------------
  const validate = () => {
    const newErrors = { email: "", password: "" };
    // Regla simple para email (solo a modo educativo, en producción validen también en el servidor)
    if (!email) newErrors.email = "Ingresa tu correo institucional";
    else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) newErrors.email = "Formato de correo no válido";

    if (!password) newErrors.password = "Ingresa tu contraseña";
    else if (password.length < 6) newErrors.password = "Mínimo 6 caracteres";
```

```
  setErrors(newErrors);
  return !newErrors.email && !newErrors.password;
 };

 // ----------------------
 // 4) Simulación de envío
 // ----------------------
 // Aquí conectarías tu API (fetch/axios). En este ejemplo sólo simulamos.
 const handleSubmit = async (e) => {
  e.preventDefault();
  if (!validate()) return;

  setIsLoading(true);
  setStatus("idle");
  setMessage("");

  try {
   // Simular petición a servidor
   await new Promise((res) => setTimeout(res, 1200));

   // Ejemplo de credenciales válidas de prueba
   const demoEmail = "demo@universidad.mx";
   const demoPass = "123456";

   if (email === demoEmail && password === demoPass) {
    setStatus("success");
    setMessage("¡Acceso concedido! Redirigiendo al panel…");
    // Ejemplo: remember me (persistencia simple)
    if (remember) localStorage.setItem("rememberEmail", email);
    // Aquí podrías usar navigate('/panel') o similar
   } else {
    setStatus("error");
    setMessage("Credenciales incorrectas. Revisa tu correo o contraseña.");
   }
  } catch (err) {
   setStatus("error");
   setMessage("Ocurrió un problema de red. Intenta nuevamente.");
  } finally {
   setIsLoading(false);
  }
 };

 // Cargar email recordado (sólo la primera vez)
 React.useEffect(() => {
  const saved = localStorage.getItem("rememberEmail");
  if (saved) setEmail(saved);
 }, []);
```

```
 return (
  <div className="min-h-screen bg-gradient-to-br from-slate-900 via-slate-800 to-slate-900 flex items-center justify-center p-4">
   {/* Fondo decorativo con blur */}
   <div className="absolute inset-0 -z-10 overflow-hidden">
    <div className="absolute top-[-10%] right-[-10%] h-72 w-72 rounded-full bg-indigo-500/20 blur-3xl" />
    <div className="absolute bottom-[-10%] left-[-10%] h-72 w-72 rounded-full bg-cyan-500/20 blur-3xl" />
   </div>

   {/* Tarjeta de login */}
   <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ duration: 0.6 }}
    className="w-full max-w-md"
   >
    <div className="bg-white/10 backdrop-blur-xl border border-white/20 rounded-3xl shadow-2xl p-8">
     {/* Encabezado */}
     <div className="flex items-center gap-3 mb-6">
      <div className="h-10 w-10 rounded-2xl bg-white/20 flex items-center justify-center">
       <ShieldCheck className="h-5 w-5 text-white" />
      </div>
      <div>
       <h1 className="text-white text-2xl font-semibold leading-tight">Acceso — Gobierno Electrónico</h1>
       <p className="text-white/70 text-sm">Propedéutico — Inicia sesión con tu correo institucional</p>
      </div>
     </div>

     {/* Mensajes de estado */}
     {status !== "idle" && (
      <div
       className={
        "mb-4 rounded-xl px-4 py-3 text-sm " +
        (status === "success"
         ? "bg-emerald-500/15 text-emerald-200 border border-emerald-400/30"
         : "bg-rose-500/15 text-rose-200 border border-rose-400/30")
       }
      >
       {message}
      </div>
```

```
      )}

      {/* Formulario */}
      <form onSubmit={handleSubmit} className="space-y-4">
        {/* Email */}
        <div>
          <label htmlFor="email" className="block text-sm text-white/80 mb-
1">Correo electrónico</label>
          <div className={`relative flex items-center rounded-2xl border $
{errors.email ? "border-rose-400/50" : "border-white/20"} bg-white/5 focus-
within:border-indigo-400/60` }>
            <Mail className="h-5 w-5 text-white/60 absolute left-3" />
            <input
              id="email"
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              placeholder="tu.nombre@universidad.mx"
              className="w-full pl-10 pr-4 py-3 bg-transparent text-white
placeholder-white/40 outline-none rounded-2xl"
              autoComplete="email"
            />
          </div>
          {errors.email && <p className="text-rose-300 text-xs mt-
1">{errors.email}</p>}
        </div>

        {/* Password */}
        <div>
          <label htmlFor="password" className="block text-sm text-white/80 mb-
1">Contraseña</label>
          <div className={`relative flex items-center rounded-2xl border $
{errors.password ? "border-rose-400/50" : "border-white/20"} bg-white/5 focus-
within:border-indigo-400/60` }>
            <Lock className="h-5 w-5 text-white/60 absolute left-3" />
            <input
              id="password"
              type={showPassword ? "text" : "password"}
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              placeholder="••••••••"
              className="w-full pl-10 pr-12 py-3 bg-transparent text-white
placeholder-white/40 outline-none rounded-2xl"
              autoComplete="current-password"
            />
            <button
              type="button"
```

```
          onClick={() => setShowPassword((v) => !v)}
          className="absolute right-3 p-1 rounded-lg text-white/70 hover:bg-
white/10 focus:outline-none focus-visible:ring-2 focus-visible:ring-white/40"
          aria-label={showPassword ? "Ocultar contraseña" : "Mostrar
contraseña"}
        >
          {showPassword ? <EyeOff className="h-5 w-5" /> : <Eye className="h-5
w-5" />}
        </button>
       </div>
       {errors.password && <p className="text-rose-300 text-xs mt-
1">{errors.password}</p>}
      </div>

      {/* Opciones */}
      <div className="flex items-center justify-between">
       <label className="flex items-center gap-2 text-white/80 text-sm select-
none">
        <input
         type="checkbox"
         checked={remember}
         onChange={(e) => setRemember(e.target.checked)}
         className="h-4 w-4 rounded accent-indigo-500"
        />
        Recordarme
       </label>
       <button type="button" className="text-indigo-300 hover:text-indigo-200
text-sm">
        ¿Olvidaste tu contraseña?
       </button>
      </div>

      {/* Botón principal */}
      <button
       type="submit"
       disabled={isLoading}
       className="w-full group relative overflow-hidden rounded-2xl bg-indigo-
500/90 hover:bg-indigo-500 px-4 py-3 text-white font-semibold transition
disabled:opacity-70 disabled:cursor-not-allowed"
      >
       <span className="relative z-10 inline-flex items-center justify-center
gap-2">
        {isLoading ? (
         <>
          <Loader2 className="h-5 w-5 animate-spin" /> Verificando…
         </>
        ) : (
```

```
          <>Ingresar</>
        )}
      </span>
      {/* Efecto brillo */}
      <span className="absolute inset-0 -translate-x-full group-hover:translate-x-0 transition will-change-transform bg-gradient-to-r from-transparent via-white/30 to-transparent" />
    </button>

    {/* Separador visual */}
    <div className="flex items-center gap-3">
      <div className="h-px flex-1 bg-white/15" />
      <span className="text-white/50 text-xs">o</span>
      <div className="h-px flex-1 bg-white/15" />
    </div>

    {/* Acceso alternativo (placeholders) */}
    <button
      type="button"
      onClick={() => alert("Conectar aquí proveedor SSO / Google / Microsoft")}
      className="w-full rounded-2xl border border-white/20 bg-white/5 hover:bg-white/10 px-4 py-3 text-white/90"
    >
      Continuar con Google (demo)
    </button>
  </form>

  {/* Nota de ayuda */}
  <p className="text-white/50 text-xs mt-6 leading-relaxed">
    <strong>Notas:</strong> Este formulario valida en el cliente y simula la comunicación con el servidor.
    Para producción, conecten su API (por ejemplo, <code>fetch('/api/login')</code>) y manejen tokens (JWT),
    control de sesiones y redirecciones seguras. Consideren usar HTTPS, manejo de errores del servidor y
    protección contra ataques de fuerza bruta (rate-limiting, captchas) si corresponde.
  </p>
</div>

{/* Pie de página pequeño */}
<p className="text-center text-white/50 text-xs mt-4">
  Tip: Intenten con <code>demo@universidad.mx</code> y contraseña <code>123456</code> para ver el flujo de éxito.
</p>
</motion.div>
```

```
  </div>
 );
}
```

Instala Tailwind y sus dependencias
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p

Configura Tailwind en `tailwind.config.js`

```
content: [
  "./src/**/*.{js,jsx,ts,tsx}",
],
```

src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

```
import './index.css';
// ==========================
// 5) Guía rápida para conectar a su API (pseudo-código)
// ==========================
// Reemplazar el bloque de simulación por:
//   const resp = await fetch("https://su-api.edu/auth/login", {
//     method: "POST",
//     headers: { "Content-Type": "application/json" },
//     body: JSON.stringify({ email, password })
//   });
//   if (!resp.ok) throw new Error("No autorizado");
//   const data = await resp.json(); // data.token, data.user
//   // Guardar token de manera segura (considerar HttpOnly cookies en el servidor)
//   // Redirigir a /panel o almacenar estado global (Context/Redux/Zustand).

// ==========================
// 6) Accesibilidad y buenas prácticas
// ==========================
// - Los labels están asociados a inputs con htmlFor/id.
// - Botones tienen aria-label cuando es acción icónica (mostrar/ocultar contraseña).
// - Estados de error visibles con texto de ayuda.
// - Controles con tamaños cómodos para táctil (padding y altura adecuados).
// - Contraste suficiente entre texto y fondo.
```