Es el proyecto que más “contrata”, y que todo tech lead o recruiter puede entender de inmediato.

Y si después quieres subir el nivel técnico de tu portafolio, puedes agregar un sexto proyecto como el de búsqueda con Elastic o Algolia para mostrar algo más avanzado.

**🗂️ Project Idea: Personal Task Manager with Login**

Una app tipo “to-do list” con autenticación de usuarios y operaciones CRUD completas.

**🔧 Características principales:**

1. **Registro/Login de usuario**
   * Usa Firebase Auth, Supabase o Auth0 para no montar backend propio.
   * Email + contraseña.
2. **Dashboard personal de tareas**
   * Cada usuario solo ve sus tareas.
   * Tareas con: título, descripción, estado (pendiente/completa), fecha límite.
3. **Operaciones CRUD**
   * Crear, leer, editar y eliminar tareas.
4. **Protección de rutas**
   * Solo accede al dashboard quien ha iniciado sesión.
   * Redirección automática si no está logueado.
5. **Diseño responsive**
   * Que se vea bien en mobile.

**✨ Extras opcionales (si tienes tiempo):**

* Filtro por tareas completadas o pendientes.
* Notificaciones con toast.
* Animaciones suaves con Framer Motion.
* Guardado offline con localStorage o indexedDB (si no usas backend).

**🖼️ ¿Cómo se vería en el portafolio?**

**Título:** Taskify – Simple task manager with user login  
**Descripción:** A task-tracking app built with React and Firebase. Users can register, log in, and manage their personal task list with full CRUD operations. The app includes protected routes, mobile-friendly design, and real-time state updates.

**🖼️ Diseño rápido (wireframe mental)**

**🔒 Login / Register Page**

+--------------------------+

| [Logo] |

| Sign in / Register |

| -------------------- |

| [ Email ] |

| [ Password ] |

| [ Log In Button ] |

| -------------------- |

| Don't have an account? |

| [ Register here ] |

+--------------------------+

**✅ Dashboard (protegido)**

+------------------------------+

| Hello, [username] |

| [ Logout ] |

|------------------------------|

| [ Add New Task Form ] |

| - title |

| - description |

| - due date |

| - [ Save ] |

|------------------------------|

| Your Tasks: |

| - [ ] Task 1 [edit] [x] |

| - [✓] Task 2 [edit] [x] |

+------------------------------+

**📦 Tecnologías sugeridas:**

* **React** (con Hooks)
* **React Router Dom** (para rutas y protección)
* **Firebase** (Auth + Firestore para guardar tareas)
* **Chakra UI o TailwindCSS** (para diseño rápido)
* **Toastify** para notificaciones

|  |
| --- |
| **Paso a paso para crear un proyecto React + Vite + Tailwind CSS**  **Doc: https://tailwindcss.com/docs/installation/using-vite**  pnpm create vite@latest CRUD --template react  Enter para confirmar la carpeta Despues entro a la carpeta    **Instalar las dependencias**  Pnpm crea las dependencias  **Instalación de Tailwind CSS con Vite**   1. **Instalar Tailwind CSS y sus plugins**   **pnpm install -D tailwindcss @tailwindcss/vite** ****2. Configura Vite**** Edita tu vite.config.js  Se agrega:  import tailwindcss from '@tailwindcss/vite'  tailwindcss(), Configure the Vite plugin Add the @tailwindcss/vite plugin to your Vite configuration.  **3. @import "tailwindcss"; en**  **src/index.css** |

|  |
| --- |
| **🔧 ¿Qué ofrece Firebase?**  Aquí lo más importante para ti como desarrollador frontend:  **1. Autenticación (Firebase Auth)**  Permite agregar **login y registro de usuarios** con:   * Correo y contraseña * Google, Facebook, GitHub, etc. * Sin tener que manejar contraseñas en tu base de datos   ✅ Ideal para proyectos como *Taskify*.  **2. Base de datos (Firestore o Realtime Database)**   * Firestore es una base de datos **NoSQL basada en documentos**. * Puedes **leer, escribir y actualizar datos en tiempo real** desde el navegador.   ✅ Perfecto para guardar tareas, productos, mensajes, etc. 3. **Hosting**  * Puedes **subir tu proyecto de React/Vue/Angular** directamente desde la terminal con un solo comando.   bash  CopyEdit  firebase deploy 4. **Storage**  * Para guardar **imágenes, archivos, videos**, etc. (útil si tienes perfiles con fotos o productos con imágenes).  5. **Functions (Opcional)**  * Escribes funciones en JavaScript/TypeScript que se ejecutan en la nube. * Como un pequeño backend (por ejemplo, para enviar emails, validar pagos, etc.). |

### ¿Por qué usar Firebase en proyectos personales o portafolio?

* Porque **te ahorra tener que crear un servidor completo**.
* Porque **es gratis** para pequeños proyectos.
* Porque te permite enfocarte en la lógica del frontend.

### npm install firebase 📦 ¿Qué hace esto?

* Te permite importar y usar los servicios como:
  + getAuth() para autenticación
  + getFirestore() si usas base de datos
  + initializeApp() para conectar con tu proyecto en Firebase Console

### Después de instalar:

1. Crea un archivo firebase.js (por ejemplo, en /src/services/firebase.js).
2. Copia ahí tu configuración de Firebase (te la da la consola de Firebase).
3. Luego puedes importar auth o db desde ese archivo.

|  |
| --- |
| 📌 ¿Dónde obtener el firebaseConfig?  1. Ve a https://console.firebase.google.com 2. Crea un proyecto nuevo (o entra a uno existente) 3. Haz clic en **</> Add web app** 4. **Haz clic en el ícono web </>**.   En la parte central de la pantalla verás el mensaje **“Comienza por agregar Firebase a tu app”** con varios íconos (iOS, Android, **web </>**, Unity, Flutter).  Ahí verás el código con tu apiKey, projectId, etc.  **ID del Proyecto taskify-55e73** |

**Siguientes pasos:**

**- Manejar el estado de autenticación de forma global**

Crea un **AuthContext** (o un hook useAuth) para exponer user, login, logout, register y loading, de modo que no dependas del efecto en App.jsx en cada componente.

// src/context/AuthContext.jsx

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **onAuthStateChanged que es ?**   * + Es un listener (observador) de Firebase Auth que **detecta–en tiempo real–cada cambio en el estado de autenticación** del usuario.   + ts onAuthStateChanged(auth, callback, [errorCallback]) → unsubscribeFn  |  | | --- | |  |  |  | | --- | | * + 1. **auth** – objeto Auth que creaste con getAuth(app). 2. **callback(user)** – función que recibirá:  • user (objeto FirebaseUser) si hay sesión activa.  • null si el usuario cerró sesión o aún no se ha autenticado. 3. *(opcional)* errorCallback – para manejar errores inespera |   **🔁 ¿Cuándo dispara el callback?**   | **Momento** | **Ejemplo práctico** | | --- | --- | | Al **montar** tu app por primera vez | Comprueba si existe un token guardado en el navegador y devuelve el usuario o null. | | Tras **login** exitoso | Llamas a signInWithEmailAndPassword; al completarse, onAuthStateChanged se dispara con el usuario autenticado. | | Después de **logout** (signOut) | El listener se dispara con null. | | Cuando expira/renueva el token en segundo plano | El usuario sigue autenticado, pero el listener se vuelve a ejecutar con el mismo user (token refrescado). | |

Crear el hook personalizado useAuth

Utilizarlo en el **main.js**

**-Completar** Login.jsx **y** Register.jsx

Register.jsx es análogo usando register

|  |
| --- |
| **Icono trash** Se tuvo que importar un icono para que funcionara el hover de color  **npm i @heroicons/react**  **import { TrashIcon } from "@heroicons/react/24/solid";**  <button  aria-label="Delete task"  onClick={() => deleteTask(t.id)}  className="text-gray-500 hover:text-red-600 transition-colors"  >  <TrashIcon className="h-5 w-5" />  </button> |