Documentación “Tecnolibros”

Biblioteca de libros sobre tecnología

Sandra Catchot Sancho

DWEC - 2ºDAW

1. Introducción

Tecnolibros es una aplicación web desarrollada con React y Firebase que permite gestionar una biblioteca de libros sobre tecnología. El propósito principal es proporcionar una plataforma intuitiva y eficiente donde los usuarios puedan añadir, visualizar, buscar y gestionar libros relacionados con la tecnología. Esta aplicación facilita operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre los libros almacenados en la base de datos Firestore de Firebase.

La elección de React como framework frontend se debe a su capacidad para crear interfaces de usuario dinámicas y reactivas, mientras que Firebase se encarga de la persistencia de datos y ofrece un backend en tiempo real.

2. Instalación y configuración

Requisitos previos

Antes de comenzar con la instalación y configuración del proyecto, me aseguro de tener los siguientes requisitos instalados en mi entorno de desarrollo:

- **Node.js** y **npm**.
- **Vite** para la creación del entorno de desarrollo React.
- Una cuenta en **Firebase** para gestionar la base de datos.

Pasos de instalación:

- **Crear repositorio e instalar dependencias:**
    - Instalo dependencias mediante Vite
- **Configurar Firebase:**
    - Creo un proyecto en Firebase.
    - Habilito Firestore Database
    - Obtengo las credenciales y las añado al proyecto en un archivo concreto (firebaseConfig.js)
- **Ejecutar la aplicación:**
  npm run dev

3. Estructura del código

Components: Contiene los componentes reutilizables que forman la estructura general de la aplicación.

- Navbar.jsx
    - ○ Barra de navegación presente en toda la aplicación.
    - ○ Incluye un título enlazado a Inicio.
    - ○ Contiene un enlace que abre la la barra de búsqueda y un botón para agregar un nuevo libro.

- List.jsx
    - ○ Muestra el listado de libros obtenidos desde Firebase.
    - ○ Cada libro tiene un enlace a su detalle (Libro.jsx).
- Searchbar.jsx
    - ○ Barra de búsqueda basada en parámetros tipo query.
    - ○ Muestra resultados en tiempo real en la página de búsqueda.

Hooks: Carpeta para los hooks.

- useFirestore.js
    - ○ Hook personalizado que centraliza todas las operaciones CRUD utilizando métodos como collection, addDoc, doc, deleteDoc, y onSnapshot para interactuar con Firestore.

Pages: Carpeta para los componentes enrutados. Cada componente estará, a su vez, dentro de una carpeta propia. Se utilice o no, cada componente estará acompañado de su "css".

- inicio/Inicio.jsx
    - ○ Muestra todos los libros almacenados en Firebase, mediante List.jsx
- libro/Libro.jsx
    - ○ Muestra la información detallada de un libro.
    - ○ Incluye botones para editar y eliminar un libro.
- formulario/Formulario.jsx
    - ○ Contiene un formulario para agregar un nuevo libro.
- busqueda/Busqueda.jsx

○ Muestra los resultados de la búsqueda de libros por título y/o autor en tiempo real.

4. Funciones implementadas

Operaciones CRUD

La aplicación permite realizar operaciones CRUD completas sobre los libros almacenados en Firestore:

- Crear:
  - Añadir nuevos libros desde Formulario.jsx.
  - Los datos se almacenan en Firestore y se actualizan en tiempo real.

- Leer:
  - Listar todos los libros en Inicio.jsx.
  - Visualizar los detalles de un libro específico en Libro.jsx.

- Eliminar:
  - Borrar libros existentes directamente desde la página de detalles de cada libro.
- Editar:
  - Editar libros existentes directamente desde la página de detalles de cada libro.

Enrutamiento:

Implementado utilizando React Router.

Algunos ejemplos de rutas utilizadas son:

➔ /: Página de inicio.
➔ /libro/:id: Página de detalles del libro.
➔ /agregar: Formulario para añadir un nuevo libro.
➔ /buscar: Página de búsqueda.
➔ “*” Redirección a la página de inicio para rutas no encontradas.

Búsqueda en tiempo real:

- Implementada a través de la Searchbar.jsx.
- Filtra los resultados dinámicamente conforme el usuario introduce texto.

- Utiliza consultas de Firestore para optimizar los resultados.

Manejo de errores:

- Gestión de errores al interactuar con Firebase.
- Validaciones básicas en el formulario para asegurar que los campos obligatorios estén completos.
- Mensajes de error amigables y alertas para guiar al usuario.

Estilizado:

- Uso de CSS personalizado para asegurar una interfaz atractiva.

5. Mejoras y futuras implementaciones

Autenticación de usuarios:

- Implementar un sistema de registro y login utilizando **Firebase Authentication**.

Validación avanzada de formularios:

- Validaciones más estrictas utilizando librerías como **Formik** o **React Hook Form**.

Filtros avanzados en la búsqueda:

- Permitir filtrar por múltiples criterios como autor, año de publicación, o categorías.

Interfaz mejorada:

- Integración completa de **Material-UI** o **Tailwind CSS** para componentes estilizados.