

My React App - CRUD con Autenticación

Gonzalo Rodríguez de Dios Cabrera

Introducción

Este proyecto implementa una aplicación React moderna con Vite que ofrece operaciones CRUD completas, autenticación JWT segura, y una interfaz de usuario profesional usando styled-components. La aplicación cuenta con un sistema de notificaciones avanzado, tema claro/oscuro personalizable, y gestión completa del perfil de usuario.



Tabla de Contenidos

- [Requisitos Implementados](#)
- [Características Principales](#)
- [Estructura del Proyecto](#)
- [Instalación y Ejecución](#)
- [Credenciales de Prueba](#)
- [Rutas y Endpoints](#)
- [Componentes y Arquitectura](#)
- [Guía de Uso](#)
- [Tecnologías Utilizadas](#)
- [Construcción y Despliegue](#)



Requisitos Implementados

Requisitos Básicos (5/10)

- ☒ CRUD completo usando json-server con búsqueda y filtrado avanzado.
- ☒ Autenticación JWT robusta con validación y gestión de errores.
- ☒ Control avanzado de errores en API con feedback visual.
- ☒ Uso de Styled Components para estilos modernos y responsivos.
- ☒ Apariencia profesional con diseño moderno y consistente.
- ☒ Uso completo de PropTypes en todos los componentes.
- ☒ Estructura modular con componentes React, hooks, context API, etc.

Requisitos Avanzados (5/10)

- ☒ Implementación avanzada de json-server-auth con endpoints personalizados.
- ☒ Interfaz moderna con transiciones, animaciones y feedback visual.
- ☒ CRUD completo con validación, búsqueda, y manejo de estados.
- ☒ Código limpio y modular con convenciones modernas de React.

-  Uso extensivo de contextos, custom hooks y componentes reutilizables.
-  LocalStorage para sesión, preferencias de tema y datos del usuario.

Características Principales

Autenticación JWT Mejorada

- Login con validación completa y feedback visual.
- Persistencia segura de sesión en localStorage.
- Verificación robusta de token en rutas protegidas.
- Interceptores avanzados para gestión de JWT.
- Manejo de expiración de sesión y renovación de tokens.

CRUD Completo y Avanzado

- Listado de registros con búsqueda en tiempo real y filtrado.
- Creación de registros con validación completa y feedback.
- Edición de registros con validación y persistencia de cambios.
- Eliminación de registros con confirmación de seguridad.
- Feedback visual en todas las operaciones mediante notificaciones.

UI/UX Profesional

- Sistema de notificaciones avanzado con desaparición automática y barra de progreso.
- Sistema de tema claro/oscuro totalmente personalizable.
- Transiciones y animaciones optimizadas para mejor experiencia.
- Diseño completamente responsivo y moderno.
- Formularios con validación en tiempo real y feedback visual.
- Navegación intuitiva con indicadores visuales.

Gestión de Perfil de Usuario

- Sección de datos personales con actualización en tiempo real.
- Gestión de preferencias de tema.
- Cambio de contraseña con validación de seguridad.
- Persistencia de preferencias de usuario entre sesiones.

Seguridad Reforzada

- Protección avanzada de rutas privadas.
- Manejo sofisticado de errores HTTP y de autenticación.
- Validación completa de datos en formularios.
- Feedback de seguridad para el usuario.
- Limpieza de entradas de usuario.

Estructura del Proyecto

```

src/
├── assets/                # Recursos estáticos y servidor
│   ├── db.json           # Base de datos para json-server con usuarios y regist
│   └── server.js         # Configuración avanzada de json-server-auth con endpo
├── components/           # Componentes de la aplicación
│   ├── common/           # Componentes reutilizables
│   │   ├── FormComponents.jsx # Componentes de formulario con validación
│   │   └── Notification.jsx   # Sistema de notificaciones avanzado con au
│   ├── Auth.jsx          # Autenticación con validación y manejo de errores
│   ├── Create.jsx        # Creación de registros con validación completa
│   ├── Read.jsx          # Listado de registros con búsqueda y filtrado
│   ├── Update.jsx        # Actualización de registros con validación
│   └── Profile.jsx       # Perfil de usuario con secciones de datos personales
├── contexts/             # Contextos de React
│   ├── AuthContext.jsx   # Gestión avanzada de autenticación con JWT
│   └── ThemeContext.jsx  # Sistema completo de temas claro/oscuro
├── styles/               # Estilos globales
│   └── GlobalStyles.jsx  # Estilos globales con soporte completo de temas
├── utils/                # Utilidades
│   └── api.js            # Cliente API con Axios, interceptores y manejo de err
├── App.jsx               # Componente raíz con rutas protegidas
└── main.jsx              # Punto de entrada de la aplicación

```

Instalación y Ejecución

Requisitos Previos

- Node.js 16.x o superior
- npm 7.x o superior

Instalación

1. Clona el repositorio:

```

git clone <URL_DEL_REPOSITORIO>
cd my-react-app

```

2. Instala las dependencias:

```

npm install

```

Ejecución del Proyecto

La aplicación requiere **dos terminales abiertas simultáneamente**:

1. **Terminal 1 (Backend):**

```
npm run server
```

El servidor estará disponible en `http://localhost:3001`

2. **Terminal 2 (Frontend):**

```
npm run dev
```

La aplicación estará disponible en `http://localhost:5173`

Credenciales de Prueba

Para acceder a la aplicación, utiliza:

- **Email:** nuevo@test.com
- **Contraseña:** password123

Rutas y Endpoints

Rutas de la Aplicación

Ruta	Componente	Acceso	Descripción
/	Auth.jsx	Público	Página de inicio/login para autenticación
/read	Read.jsx	Privado	Lista de registros con opciones de búsqueda, edición y eliminación
/create	Create.jsx	Privado	Formulario para crear nuevos registros
/update/:id	Update.jsx	Privado	Formulario para actualizar un registro existente
/profile	Profile.jsx	Privado	Gestión del perfil de usuario con datos personales y preferencias

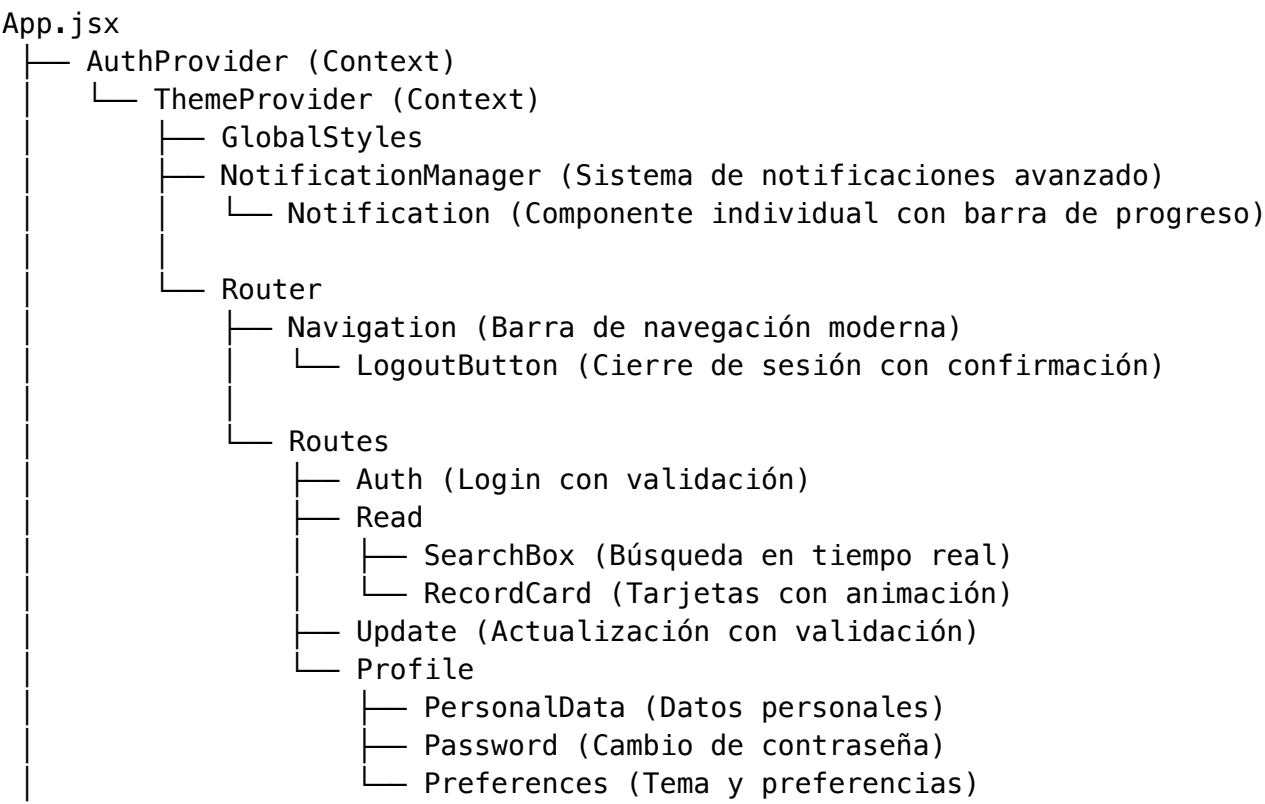
Endpoints de la API

Endpoint	Método	Autenticación	Descripción
/login	POST	No	Autenticación de usuarios y obtención de token JWT
/users/:id	GET/PUT	Sí	Obtención y actualización de datos de usuario
/records	GET/POST	Sí	Obtención de todos los registros y creación de nuevos

/records/:id	GET/PUT/DELETE	Sí	Operaciones sobre un registro específico
/profile	GET	Sí	Obtención de datos del perfil del usuario autenticado
/profile	PUT	Sí	Actualización de datos del perfil del usuario autenticado

Componentes y Arquitectura

Árbol de Componentes



Componentes Principales

Auth.jsx

- Gestiona la autenticación mediante formulario de login.
- Validación completa de campos de email y contraseña.
- Manejo detallado de errores con mensajes específicos.
- Redirección automática a área privada tras login exitoso.

Read.jsx

- Muestra la lista de registros con diseño de tarjetas moderno.
- Búsqueda en tiempo real por título y descripción.
- Acciones rápidas (editar/eliminar) por cada registro.
- Estados de carga y mensajes de error claros.

Create.jsx

- Formulario para creación de nuevos registros.
- Validación en tiempo real (título mín. 3 caracteres, descripción mín. 10).
- Feedback visual durante validación y envío.
- Notificaciones de éxito/error tras la operación.

Update.jsx

- Formulario pre-rellenado para actualización de registros.
- Carga automática de datos del registro seleccionado.
- Mismas validaciones que el formulario de creación.
- Botón para cancelar y volver a la lista.

Profile.jsx

- Gestión completa del perfil de usuario con pestañas.
- Secciones para datos personales, cambio de contraseña y preferencias.
- Actualización en tiempo real de la información de perfil.
- Persistencia de preferencias entre sesiones.

Componentes Comunes

1. FormComponents.jsx

- Componentes reutilizables para formularios con estados visuales.
- Manejo unificado de errores y validación.
- Optimizados para funcionar con ambos temas (claro/oscuro).

2. Notification.jsx

- Sistema de notificaciones con desaparición automática (5 segundos).
- Diferentes tipos: success, error, warning, info.
- Barra de progreso visual para tiempo restante.
- Colores optimizados para ambos temas.

Contextos

1. AuthContext.jsx

- Gestión centralizada del estado de autenticación.
- Almacenamiento seguro de tokens JWT en localStorage.
- Interceptores para peticiones autenticadas con Axios.
- Manejo de errores de autenticación y expiración de sesión.

2. ThemeContext.jsx

- Gestión del tema claro/oscuro con cambio instantáneo.
- Persistencia de preferencias en localStorage.
- Variables CSS globales para consistencia visual.

Sistema de Notificaciones

- **Desaparición automática:** Cada notificación se cierra tras 5 segundos.
- **Barra de progreso visual:** Indica gráficamente el tiempo restante.
- **Tipos de notificaciones:**
 - Success (verde): Operaciones correctas.
 - Error (rojo): Fallos y errores.
 - Warning (amarillo): Advertencias.
 - Info (azul): Información general.
- **Compatibilidad con temas:** Colores optimizados para tema claro y oscuro.

Uso desde cualquier componente:

```
window.notifications.success("Operación completada");  
window.notifications.error("Ha ocurrido un error");
```

Autenticación y Seguridad

- **JWT (JSON Web Token):** Autenticación basada en tokens.
- **Persistencia segura:** Almacenamiento en localStorage con manejo de expiración.
- **Interceptores Axios:** Inyección automática del token en cabeceras.
- **Rutas protegidas:** Redirección automática para usuarios no autenticados.
- **Validación robusta:** Validación en frontend y backend.
- **Gestión de errores HTTP:** Manejo centralizado de errores 401, 403, etc.

Guía de Uso

CRUD Avanzado

1. Navega a /read para ver la lista de registros.
2. Utiliza la barra de búsqueda para filtrar registros en tiempo real.
3. Crea un nuevo registro con el botón "Crear".
4. Edita un registro existente haciendo clic en "Editar".
5. Elimina un registro con el botón "Eliminar" (incluye confirmación).

Autenticación JWT

1. Inicia sesión con nuevo@test.com / password123
2. Verifica en DevTools (Application > Local Storage) el token JWT.
3. Prueba cerrar y reabrir el navegador para comprobar la persistencia.
4. Intenta acceder a una ruta protegida sin autenticación (deberías ser redirigido).

Uso de Notificaciones

1. Realiza acciones como crear, editar o eliminar registros.
2. Observa las notificaciones que aparecen con la barra de progreso.
3. Espera 5 segundos para ver cómo desaparecen automáticamente.
4. Cierra una notificación manualmente haciendo clic en la X.

Tema Claro/Oscuro

1. Navega a "Perfil" > "Preferencias".
2. Cambia entre temas claro y oscuro.
3. Comprueba que toda la interfaz se actualiza instantáneamente.
4. Cierra sesión y vuelve a entrar para verificar la persistencia.

Gestión de Perfil

1. Accede a la sección de "Perfil".
2. Actualiza tu nombre de usuario en "Datos Personales".
3. Cambia tu contraseña en la sección correspondiente.
4. Modifica tus preferencias de tema en la sección correspondiente.

Tecnologías Utilizadas

- **Frontend**
 - React 19 con Vite 6.
 - React Router v7.
 - Styled Components v6.
 - PropTypes para validación de tipos.
 - Axios con interceptores.
- **Backend**
 - JSON Server.
 - JSON Server Auth.
 - JWT para autenticación.
- **Herramientas y Utilidades**
 - LocalStorage para persistencia.
 - Variables CSS para temas.
 - Interceptores para gestión de JWT.

Construcción y Despliegue

Build de Producción

1. Genera la build optimizada:

```
npm run build
```

2. Visualiza la build localmente:

```
npm run preview
```

La previsualización estará disponible en <http://localhost:4173>

Despliegue

La aplicación está optimizada para despliegue en:

- Netlify.
- Vercel.
- GitHub Pages.
- Cualquier hosting estático.

Para el backend, considera migrar json-server a una solución más robusta como:

- Express.js + MongoDB.
- Firebase.
- Supabase.