📗 Manual Técnico – Frontend (React + Tailwind CSS)

---

1. Introducción

Este documento técnico describe la estructura, dependencias, funcionamiento y recomendaciones para el mantenimiento del **frontend** de la aplicación web de gestión de inventario, desarrollado con **React** y **Tailwind CSS**. Está dirigido a desarrolladores encargados del soporte, evolución y despliegue del cliente web.

---

2. Tecnologías Utilizadas

- **React 18+**: Librería principal para la construcción de interfaces.
- **Tailwind CSS**: Framework utility-first para estilos rápidos y responsivos.
- **Vite**: Herramienta para desarrollo y build (rápido y moderno).
- **React Router DOM**: Navegación entre vistas.
- **Axios/Fetch**: Consumo de APIs REST.
- **Heroicons / Lucide / React Icons**: Íconos.
- **Context API / Zustand / Redux (si aplica)**: Para manejo de estado (según implementación).

---

3. Instalación y Ejecución Local

Requisitos:

- Node.js 18+
- npm o yarn

Clonar e iniciar proyecto:

```
git clone https://github.com/SantanaDV/proyecto-frontend-gestor-inventario
cd frontend-inventario
npm install
npm run dev
```

Esto levantará el proyecto en `http://localhost:5173` por defecto (si usas Vite).

---

4. Estructura del Proyecto

```
proyecto-frontend/
├── excel_bd/
├── node_modules/
├── public/
├── src/
│   ├── components/
│   │   ├── layout/
│   │   │   ├── AppLayout.jsx
│   │   │   └── ProtectedRoute.jsx
│   │   └── ui/
│   │       ├── Badge.jsx
│   │       ├── Button.jsx
│   │       ├── Card.jsx
│   │       ├── ConfirmDialog.jsx
│   │       ├── Dialog.jsx
│   │       ├── DropdownMenu.jsx
│   │       ├── ErrorMessage.jsx
│   │       ├── Input.jsx
│   │       ├── Label.jsx
│   │       ├── LoadingSpinner.jsx
│   │       ├── Select.jsx
```

```
|   |       ├── Separator.jsx
|   |       ├── Switch.jsx
|   |       ├── Table.jsx
|   |       ├── Tabs.jsx
|   |       ├── Textarea.jsx
|   |       └── Tooltip.jsx
|   ├── context/
|   |   └── AuthContext.jsx
|   ├── hooks/
|   |   ├── useApi.js
|   |   ├── useAuth.js
|   |   └── useLocalStorage.js
|   ├── lib/
|   |   └── utils.js
|   ├── pages/
|   |   ├── Almacen.jsx
|   |   ├── Calendar.jsx
|   |   ├── Home.jsx
|   |   ├── Inventario.jsx
|   |   ├── Login.jsx
|   |   └── Tareas.jsx
|   ├── services/
|   |   └── api.js
|   ├── utilities/
|   |   ├── apiComunicator.jsx
|   |   ├── auth.jsx
|   |   └── storage.jsx
|   ├── App.jsx
|   ├── index.css
|   └── main.jsx
├── .gitignore
├── components.json
├── eslint.config.js
├── index.html
├── jsconfig.json
├── package-lock.json
├── package.json
├── README.md
└── vite.config.js
```

---

5. Configuración de Tailwind CSS

index.css

```
@import "tailwindcss";
@import "tw-animate-css";
```

vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import path from 'path'
import tailwindcss from '@tailwindcss/vite'
// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src'),
    },
  },
})
```

---

6. Ruteo

El código se encuentra documentado en la carpeta del proyecto.Se utiliza `react-router-dom`:

```
import { BrowserRouter as Router, Route, Routes } from
"react-router-dom"
import { AuthProvider } from "./context/AuthContext"
import AppLayout from "./components/layout/AppLayout"
import ProtectedRoute from "./components/layout/ProtectedRoute"

import Home from "./pages/Home"
import Login from "./pages/Login"
import Almacen from "./pages/Almacen"
import Estanteria from "./components/Estanteria"
import Inventario from "./pages/Inventario"
import Tareas from "./pages/Tareas"
```

```
import CalendarioTareas from "./pages/Calendar"
import LogoutButton from "./utilities/auth"

function App() {
 return (
   <Router>
     <AuthProvider>
       <AppLayout>
         <Routes>
           <Route path="/" element={<Login />} />

           <Route
             path="/home"
             element={
               <ProtectedRoute>
                 <Home />
               </ProtectedRoute>
             }
           />

           <Route
             path="/almacen"
             element={
               <ProtectedRoute>
                 <Almacen />
               </ProtectedRoute>
             }
           />

           <Route
             path="/estanteria/:id"
             element={
               <ProtectedRoute>
                 <Estanteria />
               </ProtectedRoute>
             }
           />

           <Route
             path="/inventario"
             element={
```

```
              <ProtectedRoute>
                <Inventario />
              </ProtectedRoute>
            }
          />
          <Route
            path="/tareas"
            element={
              <ProtectedRoute>
                <Tareas />
              </ProtectedRoute>
            }
          />

          <Route
            path="/calendario"
            element={
              <ProtectedRoute>
                <CalendarioTareas />
              </ProtectedRoute>
            }
          />

          <Route
            path="/logout"
            element={
              <ProtectedRoute>
                <LogoutButton />
              </ProtectedRoute>
            }
          />
        </Routes>
      </AppLayout>
    </AuthProvider>
  </Router>
 )}

export default App
```

---

7. Conexión con Backend

Se utiliza **Axios** centralizado en `services/api.js`:

```
El código se encuentra perfectamente documentado en la carpeta del
proyecto en la ruta que se menciona arriba.
```

---

8. Autenticación

El código se encuentra documentado en la carpeta del proyecto. Implementada con Context:

```
"use client"

import { createContext, useContext } from "react"
import { useAuth } from "../hooks/useAuth"

const AuthContext = createContext(null)

export function AuthProvider({ children }) {
 const auth = useAuth()

return <AuthContext.Provider
value={auth}>{children}</AuthContext.Provider>
}

export function useAuthContext() {
 const context = useContext(AuthContext)


 if (!context) {
   throw new Error("useAuthContext debe usarse dentro de un
AuthProvider")
 }return context}
```

---

9. Componentes Comunes

- `Modal.jsx` – Reutilizable para formularios
- `Button.jsx` – Estilizado con clases Tailwind
- `Input.jsx`, `Select.jsx` – Con validación integrada
- `LoadingSpinner.jsx` – Indicadores de carga con spinner
- `ConfirmDialog.jsx` – Confirmaciones reutilizadas para Eliminación,Guardado,…
- `ErrorMessage.jsx` – Mensaje de error reutilizados

Los modales se usan para crear/editar productos, tareas, categorías, etc.

---

10. Buenas Prácticas

- Componentes **pequeños, reutilizables** y **sin lógica de negocio interna**.
- Separar **presentación (UI)** de la **lógica de datos (servicios)**.
- Usar **Tailwind** con `@apply` en componentes base si hay repetición.
- Validaciones con **React Hook**.
- Usar `use Effect` correctamente para evitar renders innecesarios.

---

11. Despliegue

Vite Build:

```
npm run build
```

Esto genera la carpeta `/dist`.

---

12. Mantenimiento

- Actualizar dependencias con `npm outdated` y `npm update`.
- Limpiar componentes no usados.
- Optimizar la carga de imágenes.
- Revisar errores de consola en desarrollo.

---

13. Información del proyecto

El proyecto frontend está completamente documentado y estructurado para facilitar su mantenimiento, colaboración y escalabilidad. Toda la documentación técnica, así como el historial de cambios, se gestiona mediante control de versiones utilizando **Git** y está alojado en **GitHub**.

La documentación incluye:

- Guía de instalación y despliegue.
- Descripción de la arquitectura y estructura de carpetas.
- Detalles de componentes reutilizables.
- Uso de librerías y dependencias externas.
- Convenciones de desarrollo y estilos.
- Registro de issues, mejoras y cambios a través de *commits* y *pull requests*.

Esto garantiza la trazabilidad del desarrollo, la colaboración entre distintos miembros del equipo y una base sólida para futuras actualizaciones.

---

14. Contacto Técnico (Frontend)

- Email: info@qualicard.eu

- Repositorio: https://github.com/SantanaDV/proyecto-frontend-gestor-inventario