/agileboard-frontend

├── public

├── src

│ ├── assets <- Imágenes o íconos si usas alguno

│ ├── components <- Componentes reutilizables

│ ├── pages <- Las vistas principales del sistema

│ ├── services <- Funciones para conectar al backend (fetch/axios)

│ ├── context <- (opcional) Para manejar estados globales

│ ├── App.jsx <- Rutas y navegación

│ └── main.jsx <- Entry point

└── package.json

**📌 1. HomePage.jsx**

* Muestra dos botones:
  + "Nuevo Proyecto"
  + "Proyectos"
* El ícono de inicio debería estar siempre visible (como botón o nav)

**📌 2. NuevoProyectoPage.jsx**

* Formulario para:
  + Escribir nombre
  + Seleccionar color (puede ser dropdown o selector visual)
* Al enviar:
  + Crea el proyecto (POST al backend)
  + Navega automáticamente al tablero recién creado

**📌 3. ListaProyectosPage.jsx**

* Llama al backend y lista todos los proyectos guardados
* Cada proyecto debe tener:
  + Su nombre
  + Color
  + Un botón para "ver tablero"

**📌 4. TableroPage.jsx**

* Muestra el **tablero Kanban de ese proyecto** específico
* Tiene:
  + Título del proyecto
  + Columnas: To Do | In Progress | Done
  + Botón [+] para crear nueva tarea
  + Tareas que se pueden mover (drag & drop)
  + Opciones para editar color, texto, etc.

**📌 5. TarjetaTarea.jsx**

* Componente visual de cada tarea
* Contiene:
  + Título
  + Color
  + Icono para editar (opcional)

**📌 6. ColumnaKanban.jsx**

* Recibe:
  + Nombre de columna
  + Lista de tareas de esa columna
  + Maneja los eventos de agregar o mover tareas

**🚀 Nueva guía desde cero (react + vite + tailwind v4)**

**Te recomiendo borrar la carpeta actual agileboard-frontend y empezar totalmente limpio**.

**🗑️ Paso 0: Empezar limpio (opcional pero recomendado)**

En tu terminal:

bash

CopiarEditar

cd C:\Users\romer\OneDrive\Documents\Proyectos\AgileBoard

rm -rf agileboard-frontend

O elimina manualmente usando el explorador de archivos.

**🧱 Paso 1: Crear proyecto React con Vite**

bash

CopiarEditar

npm create vite@latest

Cuando pregunte:

* **Nombre del proyecto:** agileboard-frontend
* **Framework:** React
* **Variante:** JavaScript

Luego:

bash

CopiarEditar

cd agileboard-frontend

npm install

Confirma que funciona ejecutando:

bash

CopiarEditar

npm run dev

**🎨 Paso 2: Instalar Tailwind CSS v4 + Vite**

Ejecuta esto desde tu carpeta agileboard-frontend:

bash

CopiarEditar

npm install -D tailwindcss postcss autoprefixer @tailwindcss/vite

**⚙️ Paso 3: Configurar Vite para usar Tailwind**

Edita el archivo vite.config.js que está en la raíz del proyecto, y déjalo así:

js

CopiarEditar

import { defineConfig } from 'vite'

import react from '@vitejs/plugin-react'

import tailwindcss from '@tailwindcss/vite'

export default defineConfig({

plugins: [react(), tailwindcss()],

})

**📝 Paso 4: Crear archivo tailwind.config.js (a mano)**

En la raíz del proyecto (agileboard-frontend) crea un archivo llamado:

arduino

CopiarEditar

tailwind.config.js

y ponle esto adentro:

js

CopiarEditar

/\*\* @type {import('tailwindcss').Config} \*/

export default {

content: ["./index.html", "./src/\*\*/\*.{js,jsx}"],

theme: {

extend: {},

},

plugins: [],

};

**🎨 Paso 5: Crear archivo src/index.css con Tailwind**

Edita o crea el archivo src/index.css así:

css

CopiarEditar

@import "tailwindcss";

Ahora, edita src/main.jsx y asegúrate de importar este CSS así:

jsx

CopiarEditar

import React from 'react'

import ReactDOM from 'react-dom/client'

import App from './App.jsx'

import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>,

)

**🚀 Paso 6: ¡Probar que todo funciona!**

Pon una clase Tailwind (bg-red-500) en el archivo App.jsx para comprobar que todo está OK:

jsx

CopiarEditar

function App() {

return (

<div className="min-h-screen bg-red-500 text-white">

Hola AgileBoard con Tailwind v4

</div>

)

}

export default App

Ejecuta:

bash

CopiarEditar

npm run dev

Si ves un fondo rojo brillante, ¡quedó perfecto!

**🐙 Paso 7: Subir todo al repositorio GitHub**

Ejecuta estos comandos (dentro de la carpeta agileboard-frontend):

bash

CopiarEditar

git init

git remote add origin https://github.com/Adal666/AgileBoard.git

git add .

git commit -m "Proyecto inicial React con Vite y Tailwind v4"

git push -u origin master

Si aparece un error porque ya tenías archivos anteriores, avísame para ayudarte a resolverlo fácilmente creando una rama nueva.

**🎯 Conclusión rápida de qué hicimos:**

| **Paso** | **Descripción** | **Estado** |
| --- | --- | --- |
| 0 | Limpiar y empezar desde cero | ✅ Opcional |
| 1 | Crear proyecto React con Vite | ✅ Obligatorio |
| 2 | Instalar Tailwind v4 | ✅ Obligatorio |
| 3 | Configurar Tailwind con Vite | ✅ Obligatorio |
| 4 | Crear configuración manual Tailwind | ✅ Obligatorio |
| 5 | Importar Tailwind en CSS | ✅ Obligatorio |
| 6 | Prueba visual básica | ✅ Recomendado |
| 7 | Subir a GitHub | ✅ Recomendado |

Ahora sí debe quedar todo perfecto, portable, y listo para continuar desarrollando.

¿Avanzamos así? ¡Quedo pendiente para cualquier detalle!

**Tú dijiste:**

aguarda, en el paso 5: 🎨 Paso 5: Crear archivo src/index.css con Tailwind Edita o crea el archivo src/index.css así: css Copiar Editar @import "tailwindcss"; Ahora, edita src/main.jsx y asegúrate de importar este CSS así: jsx Copiar Editar import React from 'react' import ReactDOM from 'react-dom/client' import App from './App.jsx' import './index.css' ReactDOM.createRoot(document.getElementById('root')).render( <React.StrictMode> <App /> </React.StrictMode>, ) a que te refieres? osea, el archivo existe, y tiene lo siguiente: src/index.css: :root { font-family: system-ui, Avenir, Helvetica, Arial, sans-serif; line-height: 1.5; font-weight: 400; color-scheme: light dark; color: rgba(255, 255, 255, 0.87); background-color: #242424; font-synthesis: none; text-rendering: optimizeLegibility; -webkit-font-smoothing: antialiased; -moz-osx-font-smoothing: grayscale; } a { font-weight: 500; color: #646cff; text-decoration: inherit; } a:hover { color: #535bf2; } body { margin: 0; display: flex; place-items: center; min-width: 320px; min-height: 100vh; } h1 { font-size: 3.2em; line-height: 1.1; } button { border-radius: 8px; border: 1px solid transparent; padding: 0.6em 1.2em; font-size: 1em; font-weight: 500; font-family: inherit; background-color: #1a1a1a; cursor: pointer; transition: border-color 0.25s; } button:hover { border-color: #646cff; } button:focus, button:focus-visible { outline: 4px auto -webkit-focus-ring-color; } @media (prefers-color-scheme: light) { :root { color: #213547; background-color: #ffffff; } a:hover { color: #747bff; } button { background-color: #f9f9f9; } }

**ChatGPT Plus**

¡Buen punto! Me refiero a que ahora que instalaste Tailwind v4 con Vite, la forma correcta y recomendada por la documentación oficial para importar Tailwind al CSS principal cambió.

Con **Tailwind v4**, en lugar de usar las líneas anteriores que configuraban Tailwind en CSS:

css

CopiarEditar

@tailwind base;

@tailwind components;

@tailwind utilities;

ahora debes usar únicamente la importación directa así:

css

CopiarEditar

@import "tailwindcss";

**🔨 ¿Qué debes hacer exactamente?**

Para que quede totalmente claro, sigue estos pasos:

**1. Edita tu archivo actual src/index.css:**

* **Borra todo lo que tienes ahora** (todas las líneas que me mostraste).
* Déjalo simplemente así:

css

CopiarEditar

@import "tailwindcss";

Esto le indica a Tailwind que importe automáticamente todas las clases necesarias.

**2. Guarda los cambios.**

**📌 ¿Por qué borrar todo lo anterior?**

* Porque todas esas reglas anteriores (body, button, etc.) ya no son necesarias, ahora Tailwind manejará todo el diseño.
* Esas reglas eran solamente ejemplos de Vite cuando creaste el proyecto.

**🧪 ¿Cómo validar que está correcto?**

* Después de hacer este cambio, prueba Tailwind rápidamente editando src/App.jsx así:

jsx

CopiarEditar

function App() {

return (

<div className="min-h-screen bg-green-500 text-white flex items-center justify-center">

¡Tailwind funcionando!

</div>

)

}

export default App

Si al ejecutar npm run dev ves el fondo verde con texto blanco centrado, todo está correcto ✅.

¡Ahora sí debería quedar perfecto! ¿Probamos esto rápidamente?

**Tú dijiste:**

C:\Users\romer\agileboard-frontend>npm run dev > agileboard-frontend@0.0.0 dev > vite 11:48:06 p.m. [vite] (client) Re-optimizing dependencies because lockfile has changed VITE v6.2.3 ready in 729 ms ➜ Local: http://localhost:5173/ ➜ Network: use --host to expose ➜ press h + enter to show help entonces todo deberia estar bien verdad?

**ChatGPT Plus**

¡Exactamente! 🎉  
Ese mensaje indica que **Vite está corriendo perfectamente**, y que tu entorno ya está **listo y funcional**.

**✅ ¿Cómo saber si Tailwind está funcionando?**

Haz esta mini prueba rápida:

1. Abre tu archivo src/App.jsx
2. Reemplázalo con esto:

jsx

CopiarEditar

function App() {

return (

<div className="min-h-screen bg-purple-600 text-white flex items-center justify-center">

<h1 className="text-4xl font-bold">¡Tailwind está funcionando! 🎉</h1>

</div>

)

}

export default App

1. Guarda y abre http://localhost:5173/ en tu navegador.

**🔍 Si ves:**

* Fondo **morado**
* Texto **blanco, centrado y grande**

➡️ **¡Todo está perfecto!** Ya tienes React + Vite + Tailwind v4 funcionando como debe ser.

¿Avanzamos con crear la estructura de carpetas y tu primer página (HomePage.jsx) con los botones “Nuevo Proyecto” y “Proyectos”?

**Tú dijiste:**

solo tengo una duda: function App() { return ( <div className="min-h-screen bg-red-500 text-white"> Hola AgileBoard con Tailwind v4 </div> ) } export default App lo cambie por esto el App.jsx pero aun aparece eso verde que te muestro en la imagen, esta bien? podemos continuar?

**🚀 SIGUIENTES PASOS DESPUÉS DEL SETUP**

**✅ 8. Crear estructura de carpetas base**

Organiza src/ para tener:

css

CopiarEditar

/src

├── components/ ← botones, tarjetas, columnas

├── pages/ ← pantallas completas (home, tablero, etc.)

├── services/ ← lógica para manejar datos (crear proyecto, obtener tareas)

├── App.jsx

├── main.jsx

├── index.css

**✅ 9. Instalar React Router y configurar navegación**

bash

CopiarEditar

npm install react-router-dom

Cambia App.jsx así:

jsx

CopiarEditar

import { BrowserRouter, Routes, Route } from "react-router-dom"

import HomePage from "./pages/HomePage"

function App() {

return (

<BrowserRouter>

<Routes>

<Route path="/" element={<HomePage />} />

</Routes>

</BrowserRouter>

)

}

export default App

**✅ 10. Crear HomePage.jsx con los dos botones**

Crea archivo src/pages/HomePage.jsx:

jsx

CopiarEditar

import { useNavigate } from "react-router-dom"

function HomePage() {

const navigate = useNavigate()

return (

<div className="min-h-screen flex flex-col items-center justify-center bg-gray-100 text-center">

<h1 className="text-4xl font-bold mb-8">AgileBoard</h1>

<div className="space-x-4">

<button

className="bg-blue-500 hover:bg-blue-600 text-white px-6 py-3 rounded"

onClick={() => navigate("/nuevo-proyecto")}

>

Nuevo Proyecto

</button>

<button

className="bg-green-500 hover:bg-green-600 text-white px-6 py-3 rounded"

onClick={() => navigate("/proyectos")}

>

Proyectos

</button>

</div>

</div>

)

}

export default HomePage

**✅ 11. Crear página de Nuevo Proyecto**

Creamos src/pages/NuevoProyecto.jsx  
Formulario: nombre del proyecto + color + guardar.

**✅ 12. Crear página de Lista de Proyectos**

Mostrará los proyectos guardados con Lowdb.

**✅ 13. Configurar backend (Lowdb + Express)**

Creamos un backend local (otra carpeta agileboard-backend) con:

* Express
* Lowdb (.json como base de datos embebida)

**✅ 14. Conectar frontend con backend (fetch/Axios)**

El botón de “Guardar” proyecto mandará los datos al backend y este los guardará en el .json.

**✅ 15. Crear componente de tablero Kanban**

Dentro de un proyecto ya creado:

* Mostrar columnas To-Do / In Progress / Done
* Crear tarjetas
* Mover entre columnas (Drag & Drop)
* Guardar en base de datos

**✅ 16. Bonus: Notificaciones, dashboard, calendarios, filtros, etc.**

**🔁 Y luego: pruebas, despliegue, documentación, video demo... y a subir al GitHub.**

**🧱 PASO 1: Abre CMD y navega a tu proyecto frontend**

cmd

CopiarEditar

cd C:\Users\romer\agileboard-frontend\src

**📁 PASO 2: Crear carpetas components, pages y services**

cmd

CopiarEditar

mkdir components

mkdir pages

mkdir services

**🧹 PASO 3: Confirmar que todo quedó bien**

Tu carpeta src/ ahora debe verse así:

css

CopiarEditar

src/

├── components/

├── pages/

├── services/

├── App.jsx

├── main.jsx

├── index.css

**🧠 ¿Qué va a ir en cada carpeta?**

| **Carpeta** | **Contenido** |
| --- | --- |
| components | Componentes visuales: botón, tarjeta, columna Kanban, etc. |
| pages | Vistas completas: HomePage, TableroPage, NuevoProyecto, etc. |
| services | Lógica de conexión con backend y base de datos (guardar, obtener, etc) |

Si ya hiciste estos pasos, dime y avanzamos al siguiente:

**🚦 Paso 9: Instalar React Router y crear navegación**

**✅ 1. Instalar React Router**

**Desde la raíz del proyecto (donde está tu package.json):**

**cmd**

**CopiarEditar**

**npm install react-router-dom**

**✅ 2. Configurar rutas en App.jsx**

**Abre src/App.jsx y reemplázalo por esto:**

**jsx**

**CopiarEditar**

**import { BrowserRouter, Routes, Route } from "react-router-dom"**

**import HomePage from "./pages/HomePage"**

**function App() {**

**return (**

**<BrowserRouter>**

**<Routes>**

**<Route path="/" element={<HomePage />} />**

**</Routes>**

**</BrowserRouter>**

**)**

**}**

**export default App**

**✅ 3. Crear HomePage.jsx en src/pages**

**Crea el archivo src/pages/HomePage.jsx  
Copia y pega esto:**

**jsx**

**CopiarEditar**

**import { useNavigate } from "react-router-dom"**

**function HomePage() {**

**const navigate = useNavigate()**

**return (**

**<div className="min-h-screen flex flex-col items-center justify-center bg-gray-100 text-center">**

**<h1 className="text-4xl font-bold mb-8">AgileBoard</h1>**

**<div className="space-x-4">**

**<button**

**className="bg-blue-500 hover:bg-blue-600 text-white px-6 py-3 rounded"**

**onClick={() => navigate("/nuevo-proyecto")}**

**>**

**Nuevo Proyecto**

**</button>**

**<button**

**className="bg-green-500 hover:bg-green-600 text-white px-6 py-3 rounded"**

**onClick={() => navigate("/proyectos")}**

**>**

**Proyectos**

**</button>**

**</div>**

**</div>**

**)**

**}**

**export default HomePage**

**✅ 4. Crear rutas vacías para lo que sigue**

**Agrega estas rutas al App.jsx para dejar todo listo:**

**jsx**

**CopiarEditar**

**import NuevoProyecto from "./pages/NuevoProyecto"**

**import ListaProyectos from "./pages/ListaProyectos"**

**// dentro de <Routes>...**

**<Route path="/nuevo-proyecto" element={<NuevoProyecto />} />**

**<Route path="/proyectos" element={<ListaProyectos />} />**

**Y crea estos dos archivos vacíos:**

**bash**

**CopiarEditar**

**cd src/pages**

**echo // TODO: Crear pantalla para nuevo proyecto > NuevoProyecto.jsx**

**echo // TODO: Crear pantalla para mostrar proyectos > ListaProyectos.jsx**

**O escríbelos a mano como archivos .jsx vacíos con el siguiente contenido:**

**NuevoProyecto.jsx**

**jsx**

**CopiarEditar**

**function NuevoProyecto() {**

**return <h2>Pantalla: Nuevo Proyecto</h2>**

**}**

**export default NuevoProyecto**

**ListaProyectos.jsx**

**jsx**

**CopiarEditar**

**function ListaProyectos() {**

**return <h2>Pantalla: Lista de Proyectos</h2>**

**}**

**export default ListaProyectos**

**🧪 5. Prueba todo**

**Corre el servidor:**

**cmd**

**CopiarEditar**

**npm run dev**

* **Ve a http://localhost:5173**
* **Haz clic en "Nuevo Proyecto" → debe ir a /nuevo-proyecto**
* **Haz clic en "Proyectos" → debe ir a /proyectos**

**✅ Paso 10: Crear formulario para Nuevo Proyecto**

Este paso es clave porque empieza a darle vida real a tu app: el usuario podrá **crear un nuevo proyecto**, y luego guardarlo (más adelante lo conectamos con Lowdb).

**📦 ¿Qué vamos a hacer?**

1. Mostrar un formulario con:
   * Campo de nombre
   * Selector de color
   * Botón de guardar
2. Validar los datos localmente
3. Mostrar el resultado en consola por ahora (luego conectamos al backend)

**🛠️ Paso a paso**

**📄 1. Abre NuevoProyecto.jsx y reemplaza con este código:**

jsx

CopiarEditar

import { useState } from "react"

function NuevoProyecto() {

const [nombre, setNombre] = useState("")

const [color, setColor] = useState("#4f46e5") // valor por defecto

const handleSubmit = (e) => {

e.preventDefault()

if (!nombre.trim()) {

alert("El nombre del proyecto es obligatorio.")

return

}

const nuevoProyecto = {

id: Date.now(),

nombre,

color,

}

console.log("🟢 Proyecto creado:", nuevoProyecto)

alert("Proyecto creado (aún no se guarda en la base de datos).")

// limpiar formulario

setNombre("")

setColor("#4f46e5")

}

return (

<div className="min-h-screen bg-white flex flex-col items-center justify-center p-6">

<h2 className="text-3xl font-bold mb-6">Crear Nuevo Proyecto</h2>

<form

onSubmit={handleSubmit}

className="bg-gray-100 p-6 rounded-lg shadow-md w-full max-w-md space-y-4"

>

<div>

<label className="block text-sm font-semibold mb-1">Nombre del Proyecto</label>

<input

type="text"

value={nombre}

onChange={(e) => setNombre(e.target.value)}

className="w-full px-4 py-2 border rounded"

placeholder="Ej. Tablero Kanban"

/>

</div>

<div>

<label className="block text-sm font-semibold mb-1">Color</label>

<input

type="color"

value={color}

onChange={(e) => setColor(e.target.value)}

className="w-12 h-12 rounded-full border-none"

/>

</div>

<button

type="submit"

className="w-full bg-blue-500 text-white py-2 rounded hover:bg-blue-600"

>

Crear Proyecto

</button>

</form>

</div>

)

}

export default NuevoProyecto

**🔍 ¿Qué hace este código?**

* Guarda el **nombre** y el **color** con useState.
* Cuando haces submit:
  + Valida el nombre
  + Crea un objeto proyecto
  + Lo muestra en consola
  + Limpia el formulario

**🧪 Prueba ahora:**

bash

CopiarEditar

npm run dev

1. Ve a /nuevo-proyecto
2. Escribe un nombre
3. Elige un color
4. Clic en **Crear Proyecto**
5. Verás el resultado en consola del navegador (F12 → pestaña Console)

**✅ Resumen de Avance del Proyecto AgileBoard (hasta el Paso 12)**

**📦 Estructura del Proyecto**

**Frontend (React + Vite + Tailwind CSS v4)**

* Proyecto creado con Vite y React
* Tailwind CSS v4 integrado manualmente (sin npx tailwindcss init)
* Ruteo con react-router-dom
* Carpetas organizadas:

css

CopiarEditar

/src

├── components/

├── pages/

├── services/

├── App.jsx

├── main.jsx

├── index.css

**Backend (Express + Lowdb)**

* Backend creado con Node.js + Express
* Base de datos embebida con Lowdb (db.json)
* API básica:
  + GET /proyectos: listar todos los proyectos
  + POST /proyectos: crear un nuevo proyecto

**🧱 Funcionalidades ya implementadas**

**🖼️ Vistas React:**

* **HomePage.jsx**:  
  Botones para navegar a:
  + /nuevo-proyecto
  + /proyectos
* **NuevoProyecto.jsx**:
  + Formulario con:
    - Input de nombre
    - Selector de color
    - Botón para crear proyecto
  + Valida campos
  + Usa servicio crearProyecto() para enviar al backend

**📡 Backend funcionando:**

* Express + Lowdb conectado a db.json
* Maneja rutas para crear y listar proyectos
* Inicializa db.json automáticamente si no existe

**🔧 Correcciones importantes realizadas**

| **#** | **Problema detectado** | **Solución aplicada** |
| --- | --- | --- |
| 1 | npx tailwindcss init -p da error | Se siguió el enfoque moderno con Tailwind v4 usando plugin @tailwindcss/vite |
| 2 | git no era reconocido | Se instaló Git y se configuró correctamente |
| 3 | Error "lowdb: missing default data" | Se creó manualmente db.json y se agregó validación en código para inicializarlo |
| 4 | Vite no encontraba ../services/proyectosService | Se creó el archivo proyectosService.js dentro de src/services |

**📁 Archivos clave hasta ahora**

| **Archivo** | **Descripción** |
| --- | --- |
| NuevoProyecto.jsx | Formulario para crear proyectos |
| proyectosService.js | Funciones crearProyecto() y obtenerProyectos() |
| index.js (backend) | Express + Lowdb para manejar datos en db.json |

**🔜 ¿Qué sigue?**

**✅ Paso 13: Mostrar todos los proyectos en la vista /proyectos**

* Crear la pantalla ListaProyectos.jsx
* Llamar a obtenerProyectos() desde el servicio
* Renderizar tarjetas con el nombre y color de cada proyecto