React + Tailwind

Diego Muñoz

09 de noviembre de 2025

Introducción

- Objetivo: montar **React + Vite + Tailwind** y entender su lógica **utility-first**.
- Practicar componentes simples y responsivos.
- Implementar **dark mode**.
- Mantener un código limpio y componible.

Creación del proyecto

```
yarn create vite react-tailwind-demo --template react-swc
cd react-tailwind-demo && yarn
yarn add tailwindcss @tailwindcss/vite
```

Setup mínimo (Vite + Tailwind)

Editamos `vite.config.js`:

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react-swc";
import tailwindcss from "@tailwindcss/vite";

// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
});
```

Añadimos a `src/index.css` lo siguiente:

```
@import "tailwindcss";
```

Arranque básico

index.html:

```
<body class="bg-white text-gray-900 dark:bg-gray-950 dark:text-gray-100">
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
```

src/main.jsx:

```
import React from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App.jsx";

createRoot(document.getElementById("root")).render(<App />);
```

¿Qué significa “utility-first”?

- Tailwind no entrega componentes prefabricados.
- En vez de eso, usa **clases pequeñas y descriptivas** (utilities) que se combinan.

```
<button className="px-3 py-1 rounded bg-indigo-600 text-white">
  Guardar
</button>
```

Diseño responsivo (ejemplo)

```jsx
export default function Hero(){
  return (
    <section className="mx-auto max-w-2xl px-4 py-6">
      <h1 className="text-3xl sm:text-5xl font-bold">React + Tailwind</h1>
      <p className="mt-2 text-gray-600">UI rápida y consistente.</p>
      <div className="mt-4 flex gap-2">
        <button className="px-3 py-1 rounded bg-indigo-600 text-white">Start</button>
        <button className="px-3 py-1 rounded border">Docs</button>
      </div>
    </section>
  );
}
```

Componente Button

```jsx
// src/components/Button.jsx
export default function Button({className="",...p}){
  return <button className={"px-3 py-1 rounded "+className} {...p}/>;
}

<Button className="bg-indigo-600 text-white">Save</Button>
<Button className="bg-emerald-600 text-white">Confirm</Button>
```

Componente `Card` + grid

```jsx
// src/components/Card.jsx
export default function Card({title,children,footer}){
  return (
    <div className="rounded-xl border bg-white dark:bg-gray-900">
      <div className="p-3">
        {title && <h3 className="text-lg font-semibold mb-2">{title}</h3>}
        {children}
      </div>
      {footer && <div className="p-2 border-t">{footer}</div>}
    </div>
  );
}

<div className="grid grid-cols-1 sm:grid-cols-2 gap-3">
  <Card title="A">…</Card>
  <Card title="B">…</Card>
</div>
```

Navbar pegada

```jsx
// src/components/Navbar.jsx
export default function Navbar({onToggleTheme}){
  return (
    <header className="sticky top-0 bg-white/80 backdrop-blur border-b">
      <div className="max-w-4xl mx-auto h-10 px-3 flex items-center justify-between">
        <b>React + Tailwind</b>
        <nav className="flex gap-2 text-sm">
          <a href="#">Home</a><a href="#">Docs</a>
          <button onClick={onToggleTheme} className="px-2 py-1 rounded border">Theme</button>
        </nav>
      </div>
    </header>
  );
}
```

Composición en App.jsx

```
import Navbar from "./components/Navbar";
import Button from "./components/Button";
import Card from "./components/Card";
export default function App(){
  const t=()=>document.documentElement.classList.toggle("dark");
  return (
    <>
      <Navbar onToggleTheme={t}/>
      <main className="max-w-4xl mx-auto px-3 py-4 grid gap-3 sm:grid-cols-2">
        <Card title="A"
          footer={<Button className="bg-indigo-600 text-white">Go</Button>}>…</Card>
        <Card title="B"
          footer={<Button className="bg-emerald-600 text-white">Go</Button>}>…</Card>
      </main>
    </>
  );
}
```

Proyecto de ejemplo

Referir a ejemplo en GitHub.

Buenas prácticas

- Repite lo menos posible los mismos `className`.
- Si algo se usa mucho, hazlo componente (`Button`, `Card`, `Container`).
- Usa `clsx` para clases que dependen de una condición.
- Mira los ejemplos oficiales de Tailwind para inspirarte.