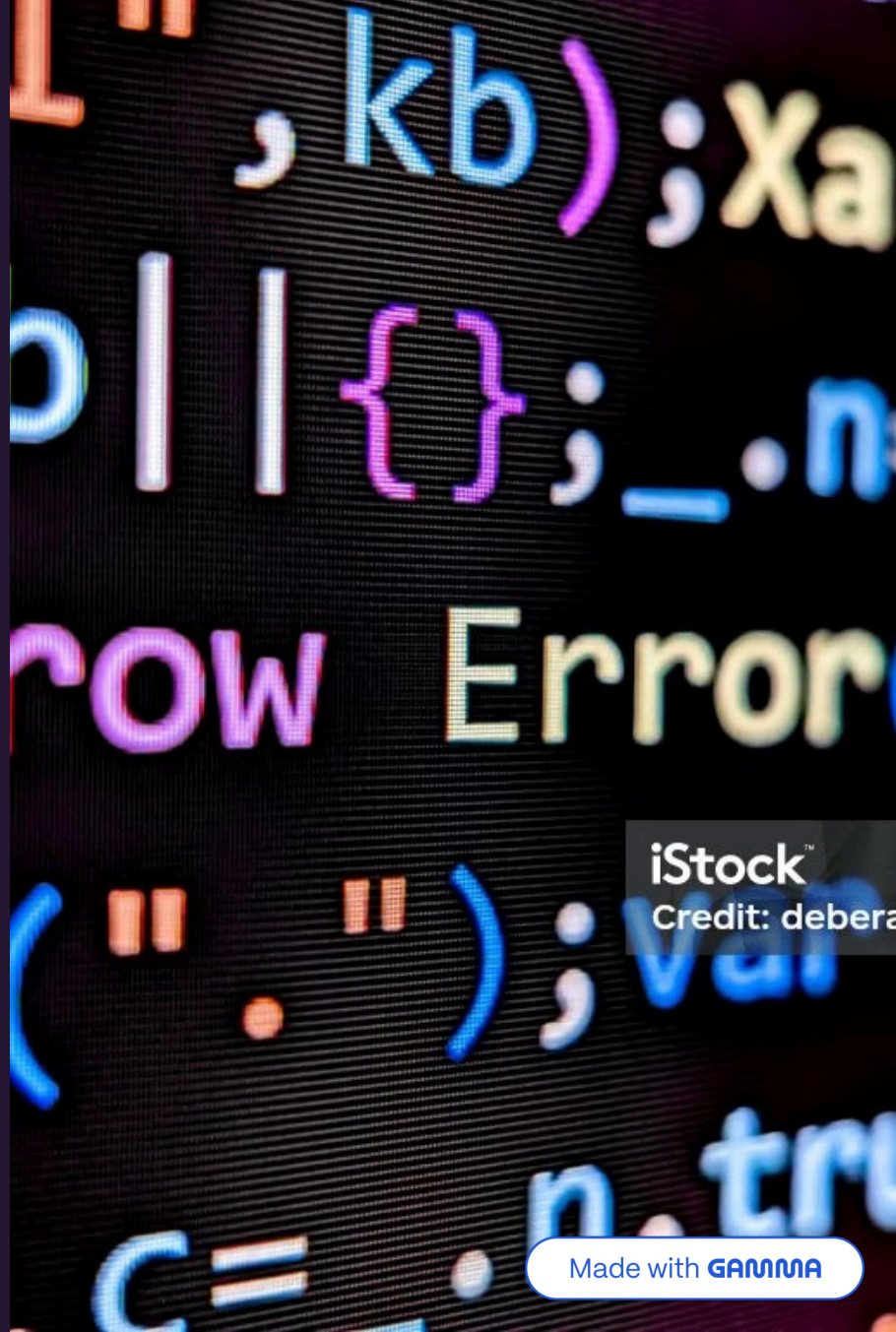
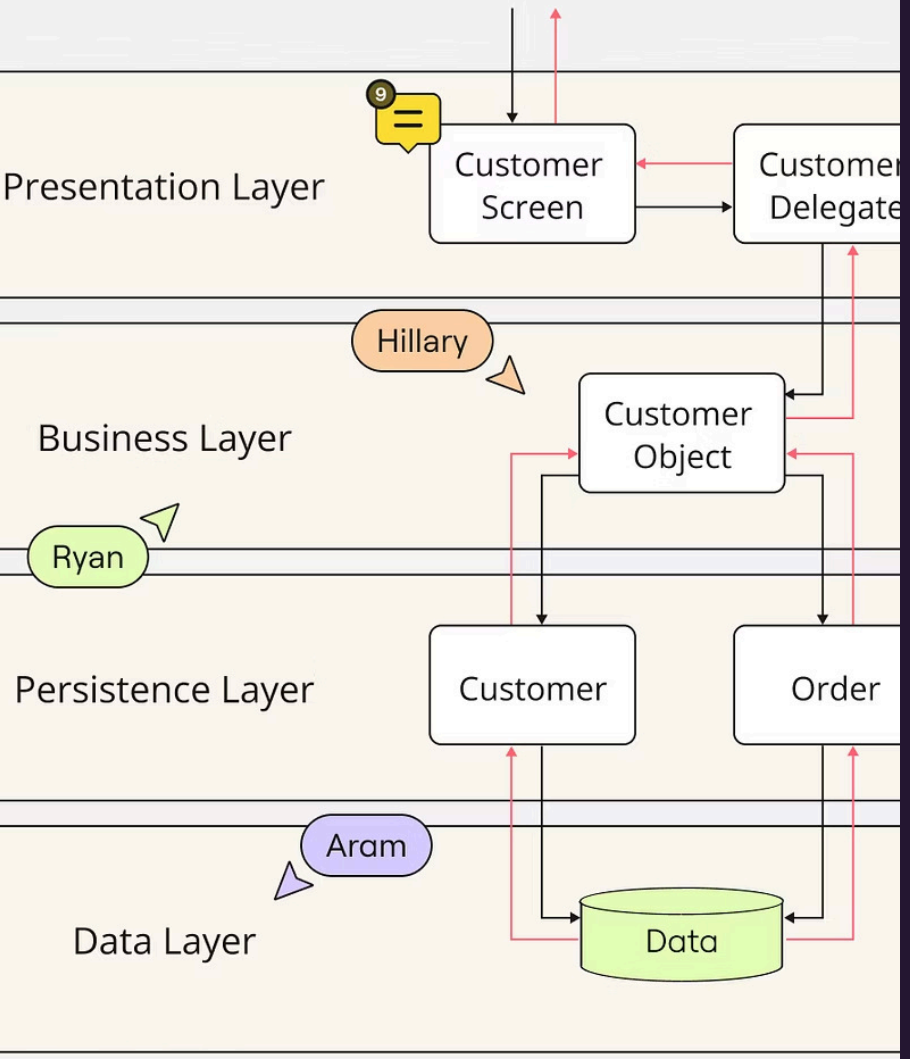
Introducción a React

React es una biblioteca de JavaScript diseñada para construir interfaces de usuario dinámicas y eficientes. Fue creada y es mantenida por Facebook, lo que garantiza constantes mejoras y una gran comunidad de soporte. React se enfoca en el desarrollo basado en componentes reutilizables, facilitando la creación de aplicaciones web modernas.

Su popularidad radica en su capacidad para desarrollar aplicaciones de página única (SPA), donde la navegación es rápida y fluida, mejorando la experiencia del usuario y la productividad del desarrollador.





¿Qué es React?

Biblioteca, no framework

React se enfoca únicamente en la capa de presentación (UI), a diferencia de frameworks completos que abarcan todo el stack.

Componentización

Permite dividir la interfaz en componentes independientes, facilitando mantenimiento y escalabilidad.

Gestión del estado

Facilita manejar el estado de la aplicación para renderizar solo lo necesario.

Características Clave de React

Componentes

Son los bloques reutilizables que forman la UI, permitiendo modularidad y orden.

JSX

Una sintaxis que combina HTML y JavaScript para escribir interfaces de forma intuitiva.

Virtual DOM

React usa una copia virtual del DOM para actualizar solo lo que cambia, mejorando la velocidad.

Flujo Unidireccional

Los datos fluyen de padres a hijos, facilitando la depuración y control del estado.

Componentes en React

Componentes Funcionales

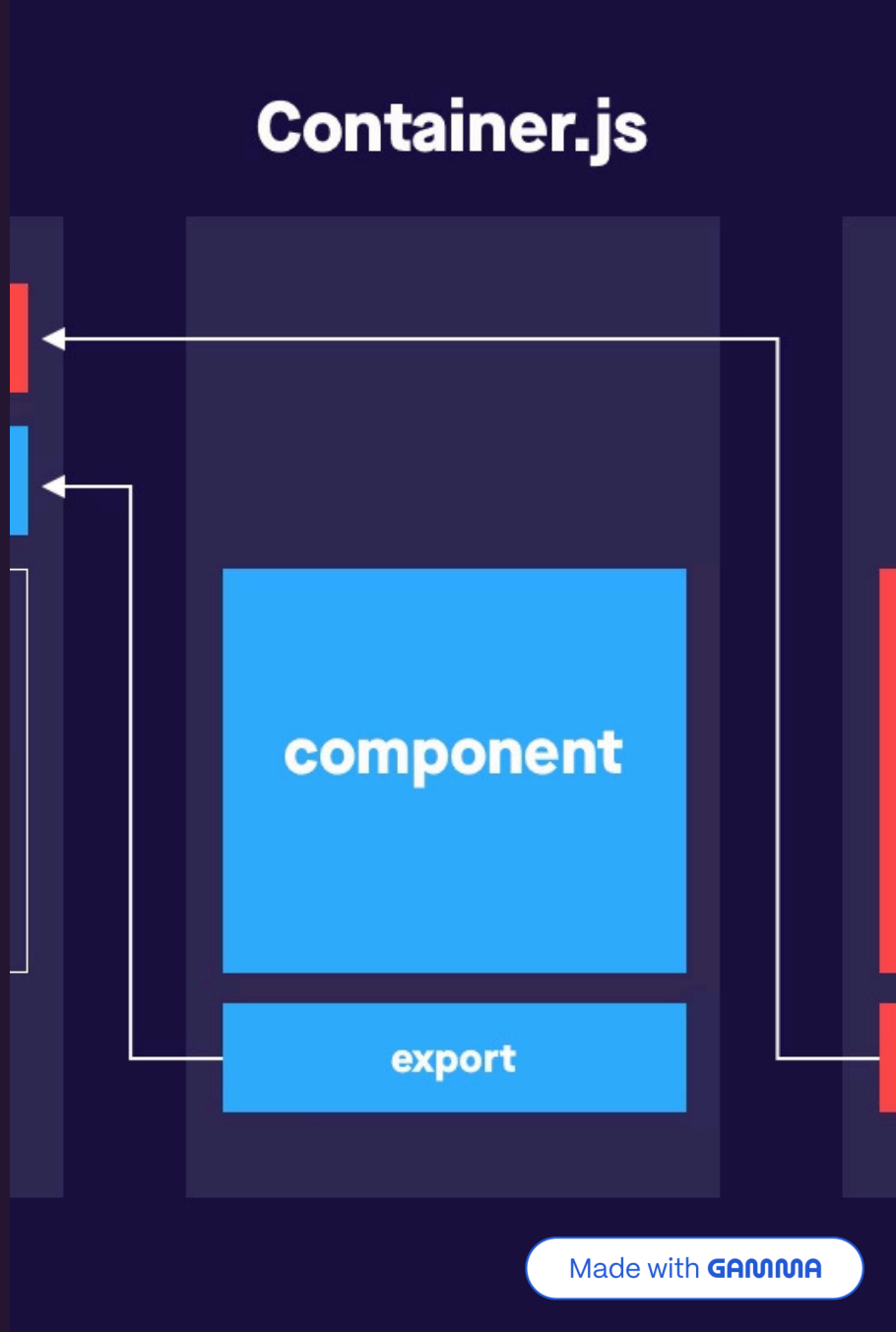
Introducen hooks para gestionar estado y efectos, son más simples y recomendados en la actualidad.

Componentes de Clase

Método tradicional que utiliza la función render y lifecycle methods para manejar el ciclo del componente.

Reutilización

Permite usar componentes en múltiples partes, ahorrando desarrollo y garantizando consistencia.



JSX: JavaScript XML

HTML en JavaScript

JSX combina la estructura de HTML con la lógica de JavaScript para una escritura más clara y legible.

Transpilado por Babel

JSX no es JavaScript puro, se convierte en llamadas a funciones usando Babel para ser entendido por el navegador.

Por ejemplo, escribir **<h1>Hola, {nombre}</h1>** genera un elemento que muestra un saludo personalizado según el valor de **nombre**.

Virtual DOM

1 **Representación en memoria**

Se crea una copia ligera del DOM real en memoria para gestiones rápidas.

2 **Comparación y diffing**

React compara la nueva versión del Virtual DOM con la anterior para detectar cambios precisos.

3 **Actualización selectiva**

Solo se modifican las partes necesarias del DOM real, evitando recargas completas.

Flujo de Datos Unidireccional

Dirección clara

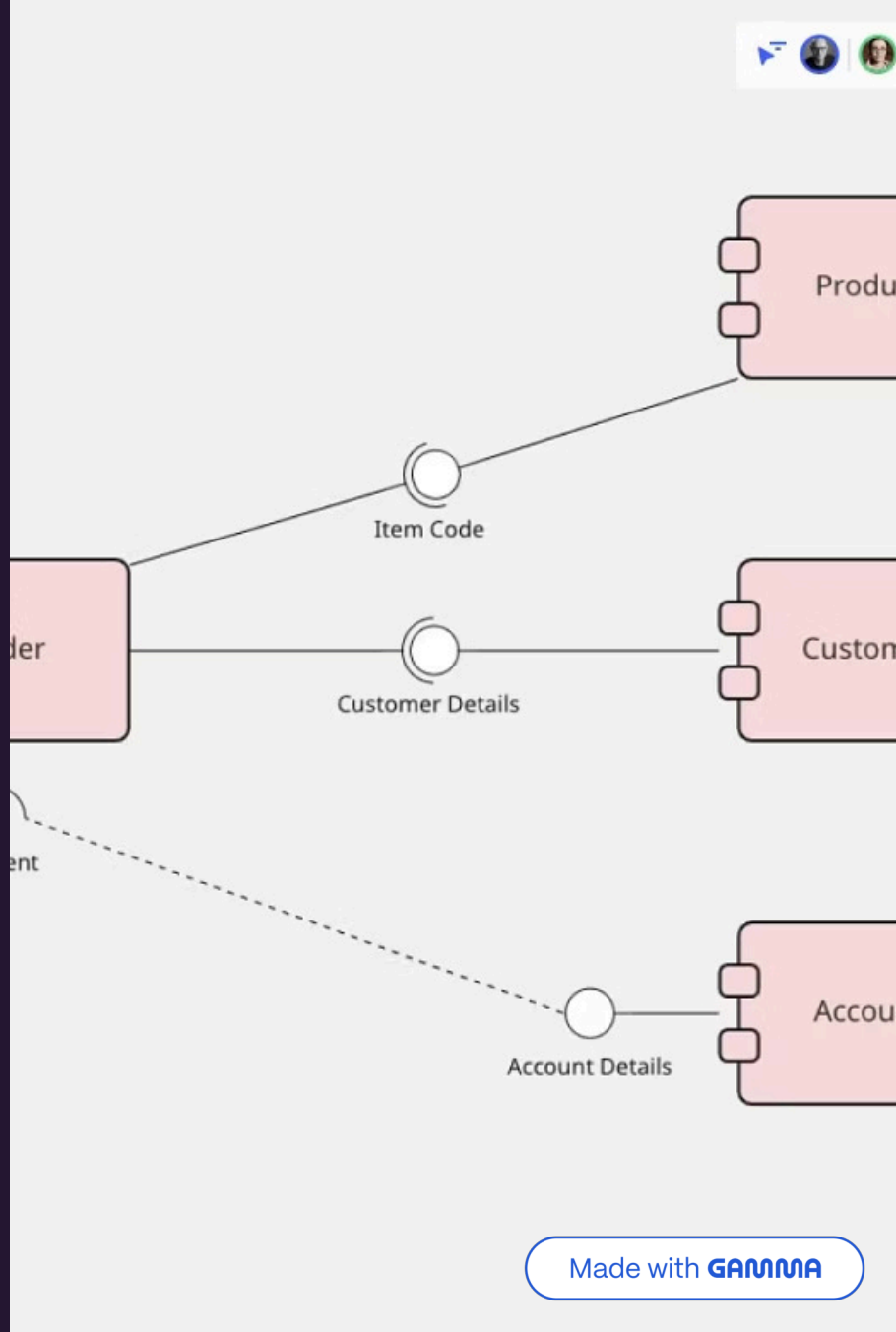
Los datos viajan de componentes padres a hijos, simplificando la lógica y la depuración.

Control del estado

El estado se centraliza y mantiene organizado, evitando problemas difíciles de rastrear.

Evita efectos secundarios

Minimiza errores y comportamientos inesperados al tener un flujo predecible de datos.



¿Por qué usar React?

Reutilización de componentes

Ahorra tiempo y esfuerzo al construir interfaces consistentes y modulares.

Alto rendimiento

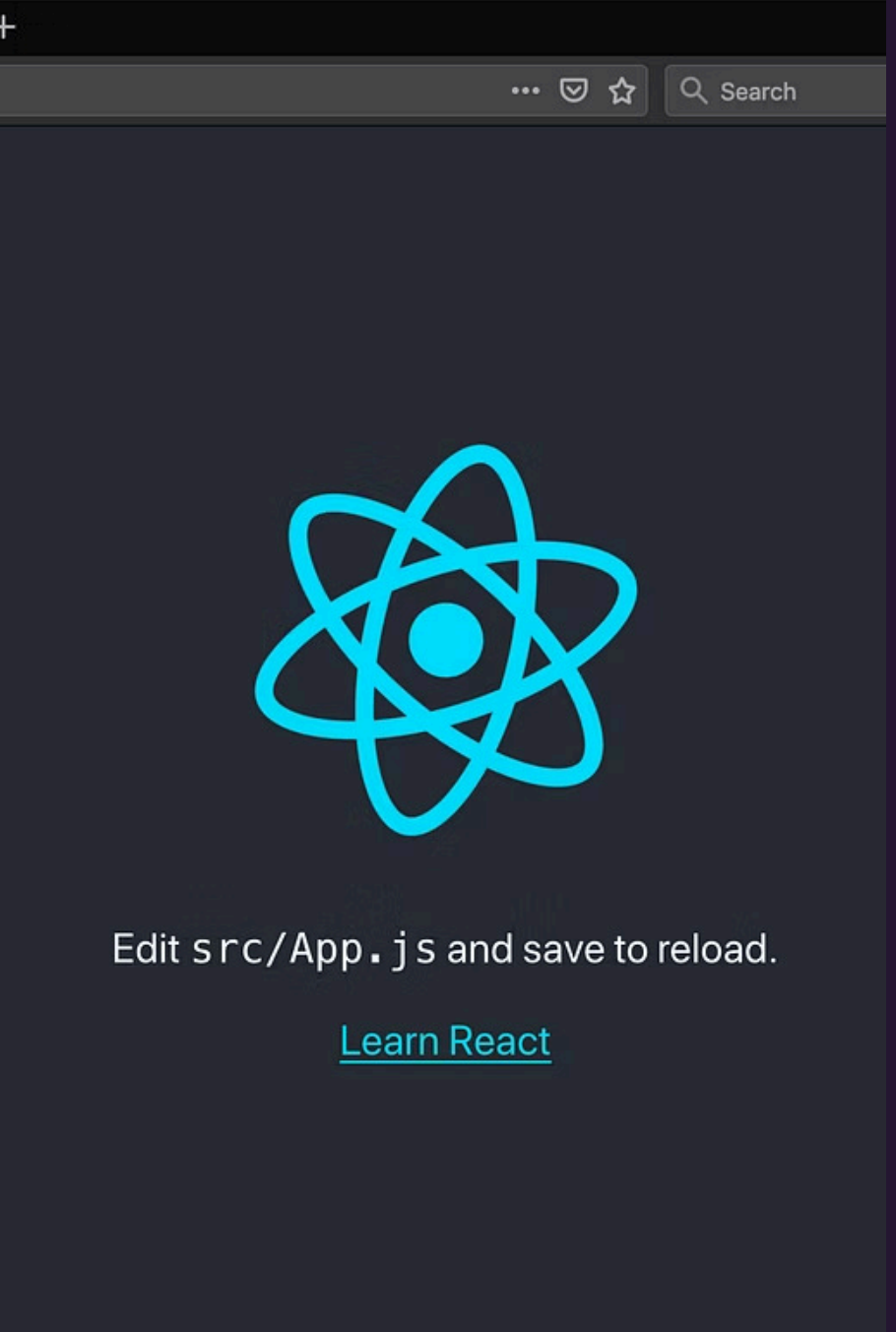
El Virtual DOM optimiza las actualizaciones, haciendo apps rápidas y responsivas.

Gran comunidad

Acceso a recursos, bibliotecas y soporte activo de desarrolladores alrededor del mundo.

Curva de aprendizaje suave

Facilidad para comenzar y escalar en proyectos complejos gracias a su diseño claro y documentado.



Primeros Pasos con React

Instalar Node.js y npm/yarn

Herramientas básicas para gestionar paquetes y ejecutar proyectos React.

Crear app con create-react-app

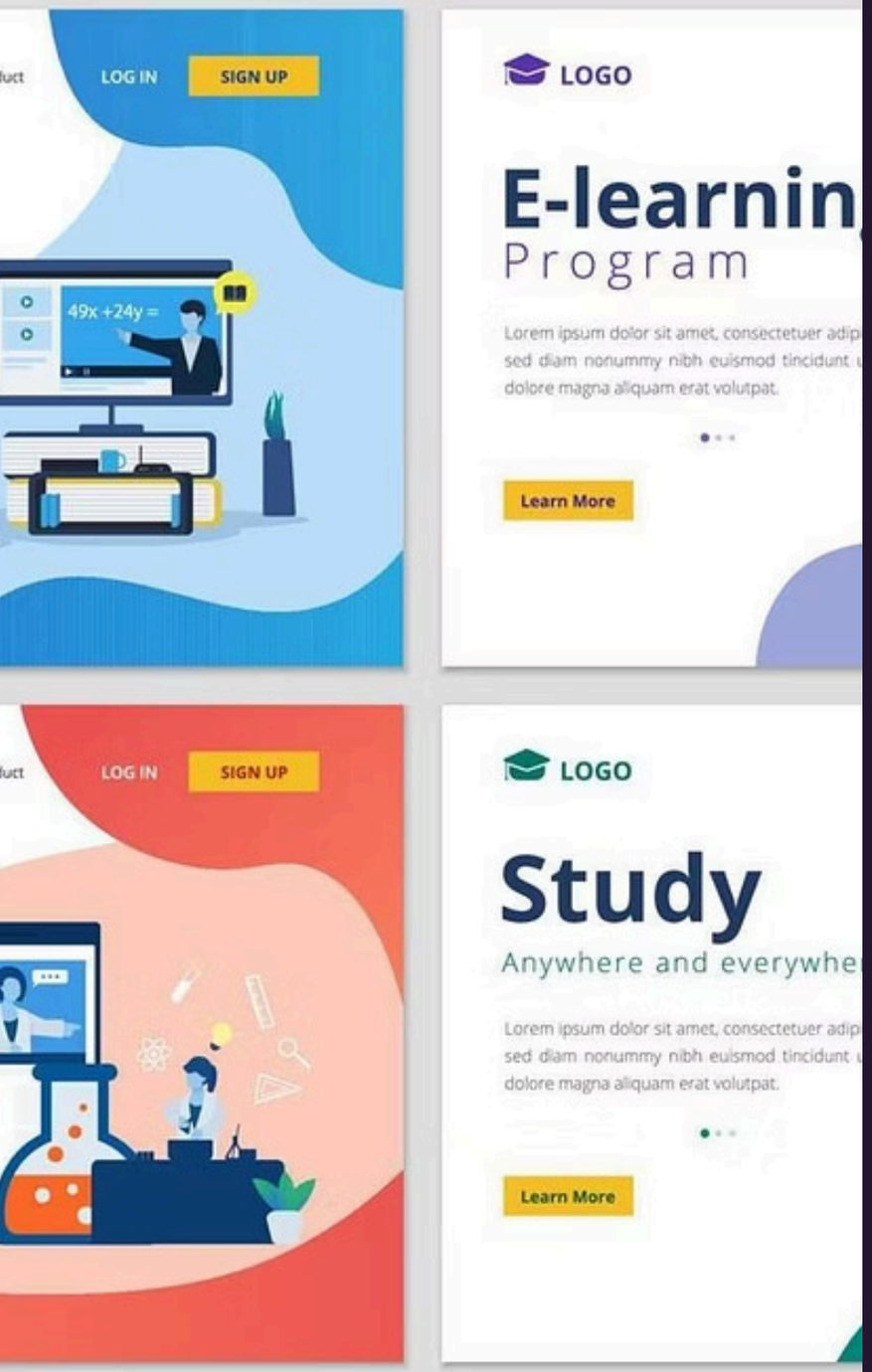
Generador de plantilla que configura automáticamente un proyecto listo para desarrollar.

Estructura del proyecto

Carpetas organizadas para componentes, estilos y lógica, facilitando el orden y mantenimiento.

Ejecutar en navegador

Con un servidor local, se puede probar la aplicación y ver cambios en tiempo real.



Recursos para Aprender React

Documentación oficial

La mejor fuente para entender funciones y buenas prácticas: **react.dev**.

Cursos online

Plataformas como Udemy, Coursera y freeCodeCamp ofrecen guías completas para todos los niveles.

Tutoriales y blogs

Sitios como dev.to y Medium comparten trucos, novedades y casos prácticos de React.

Comunidades activas

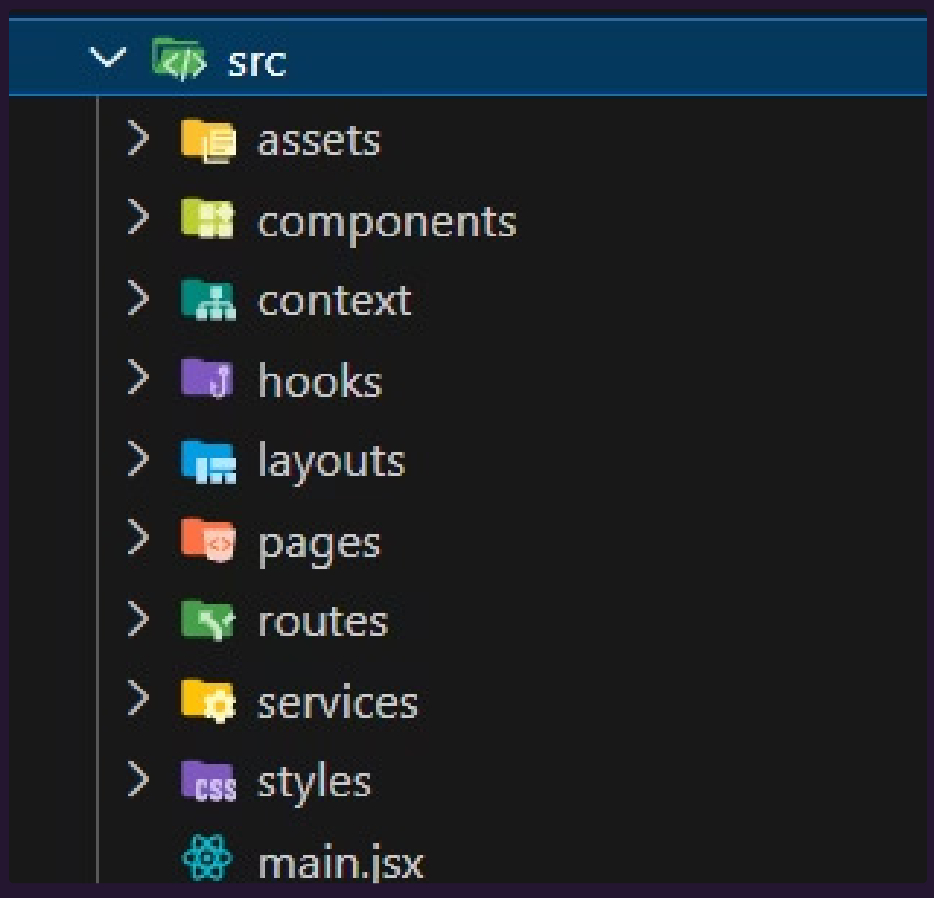
Participa en Stack Overflow y Discord para resolver dudas y aprender de otros desarrolladores.

Creación de Proyecto en React con Vite

✅ PASO 1: Crear el proyecto con Vite

```
npm create vite@latest my-react-app -- --template react
cd my-react-app
npm install
```

✅ PASO 2: Estructura básica de carpetas




```
├── assets/ # Imágenes, íconos, etc.
├── components/ # Componentes reutilizables
├── pages/ # Vistas o páginas completas
├── layouts/ # Layouts generales (Header, Footer)
├── routes/ # Definición de rutas
├── services/ # Funciones para acceder a APIs
├── hooks/ # Custom Hooks
├── context/ # React Context API
└── styles/ # Archivos CSS o tailwind.config
```

✅ PASO 3: Archivos básicos con comentarios

📄 src/main.jsx

```jsx
// Importa React y ReactDOM
import React from 'react';
import ReactDOM from 'react-dom/client';
// Importa el componente principal de la app
import App from './App.jsx';

// Renderiza la aplicación en el elemento con id 'root'
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

📄 src/App.jsx

```jsx
// Componente principal de la app
export default function App() {
  return (
    <div>
      <h1>Hola Mundo con React</h1>
      <p>Este es el inicio de tu proyecto.</p>
    </div>
  );
}
```

✅ PASO 4: Ejecutar la app

```
npm run dev
```