



Instituto Tecnológico de Tepic
Ing. Sistemas Computacionales



Interfaces Web

Introducción a:

REACT

Acosta Carrillo Yvan Fernando
Ramírez Velázquez Lia Rebeca
Topete Arvizu Roman
Vargas Partida Jorge Luis



¿QUÉ ES React ?

Es una biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario interactivas y eficientes.



Se basa en un enfoque de componentes, lo que permite reutilizar código y mejorar la organización del desarrollo web.

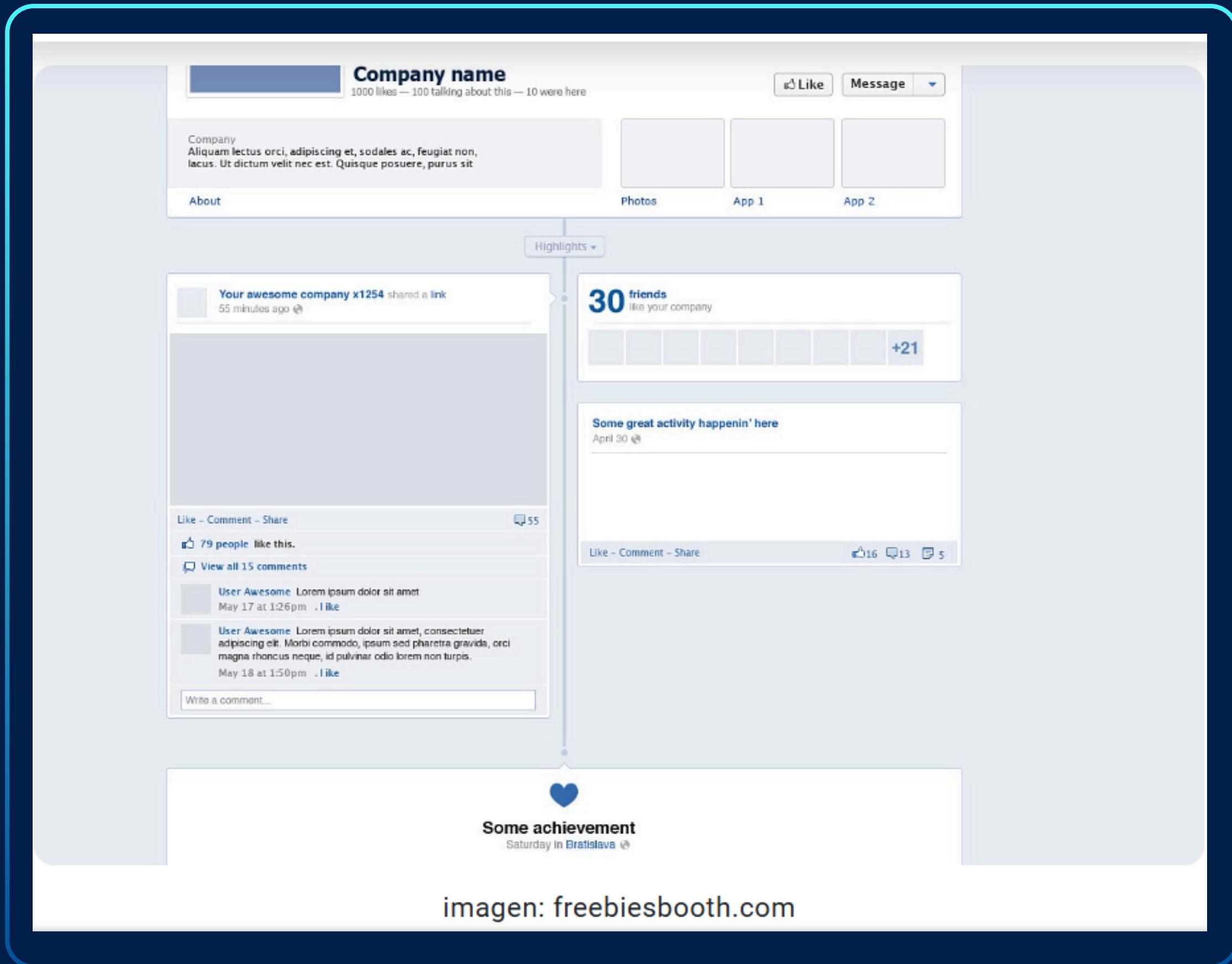
¿QUIÉN LO CREÓ Y POR QUÉ?

Se creó en 2013 por el ingeniero de software de Facebook llamado Jordan Walke.

React nació de una necesidad por ajustar con mayor facilidad y rapidez los timelines de Facebook

Al ser utilizado para la vista Frontend, se puede integrar con cualquier entorno de lado del servidor Backend como Node JS o Frameworks como Django, Ruby on Rails, Spring.

ORIGEN



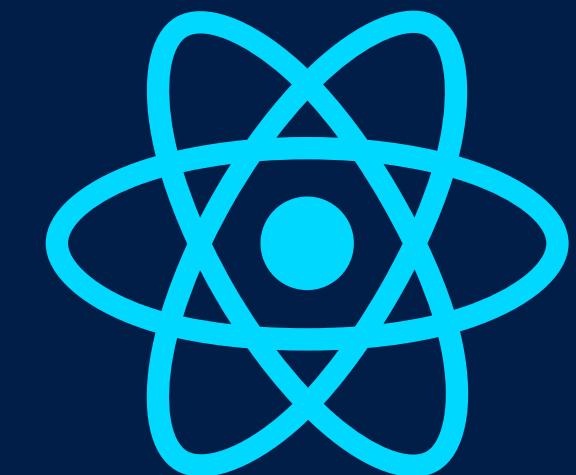
¿POR QUÉ ES TAN POPULAR?

Facilidad de uso:
Su sintaxis basada en JSX permite escribir código más intuitivo.

Alto rendimiento:
Gracias al DOM virtual, React actualiza sólo los elementos necesarios en la página.

Componentes reutilizables:
Facilita la creación de interfaces modulares.

Gran comunidad y soporte:
Al ser ampliamente adoptado, cuenta con una comunidad activa y numerosos recursos de aprendizaje.



VENTAJAS PRINCIPALES DE REACT

- Es fácil de aprender y no es necesario aprender TypeScript o Angular.
- Alto nivel de flexibilidad y máxima capacidad de respuesta.
- DOM virtual (Document Object Model)
- Biblioteca JavaScript 100% de código abierto con frecuentes actualizaciones

¿CÓMO FUNCIONA REACT?

React es una biblioteca de JavaScript desarrollada por Facebook que permite construir interfaces de usuario de manera eficiente y modular. Su enfoque principal es facilitar la creación de aplicaciones web interactivas y dinámicas mediante la reutilización de componentes.

Una de las mayores ventajas de usar React es que se puede infundir código HTML con JavaScript. Los usuarios pueden crear una representación de un nodo DOM declarando la función Element en React.





COMPONENTES EN REACT: LA BASE DE TODO

✓ ¿QUÉ SON?

Fragmentos de código reutilizables e independientes que combinan lógica e interfaz.

✓ TIPOS

Componentes Funcionales y de Clase con diferentes estilos y complejidades.

✓ IMPORTANCIA

Fundamento para construir aplicaciones escalables y mantenibles.

COMPONENTES FUNCIONALES: SIMPLICIDAD Y HOOKS

1

DEFINICIÓN

Funciones JavaScript que retornan JSX para representar UI.

2

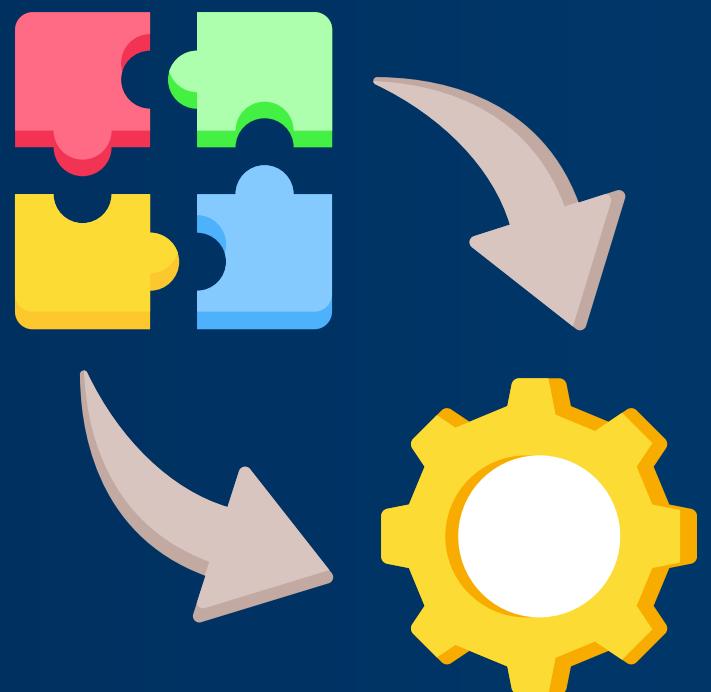
EJEMPLO BÁSICO

```
const MiComponente = () => {  
  return <h1>Hola Mundo</h1>;  
};
```

3

HOOKS

Permiten agregar estado y efectos: useState, useEffect, useContext.



COMPONENTES DE CLASE: ESTADO Y CICLO DE VIDA

¿QUÉ SON?

Clases que extienden React.Component y manejan estado y ciclo de vida.

CICLO DE VIDA

Métodos como componentDidMount para gestión y actualizaciones.

ESTADO LOCAL

Almacenado en this.state para controlar datos internos.



JSX JAVASCRIPT XML

Es una extensión de la sintaxis de JavaScript que permite escribir código similar a HTML dentro de archivos JavaScript.

Aunque parece HTML, JSX es transformado por herramientas como Babel en llamadas a `React.createElement` para crear los elementos que React renderiza.



PROPS Y STATE

PROPS

Las props son un mecanismo para pasar datos desde un componente padre a un componente hijo. Son de solo lectura y permiten que los componentes sean reutilizables y dinámicos al recibir diferentes valores.

```
function Saludo(props) {  
  return <h1>Hola, {props.nombre}!</h1>;  
}
```

STATE

El state (estado) es un objeto que representa los datos internos de un componente que pueden cambiar a lo largo del tiempo. A diferencia de las props, el state es mutable y se utiliza para manejar información que puede cambiar en respuesta a acciones del usuario o eventos del sistema.

```
import React, { useState } from 'react';  
  
function Contador() {  
  const [contador, setContador] = useState(0);  
  
  return (  
    <div>  
      <p>Has hecho clic {contador} veces</p>  
      <button onClick={() => setContador(contador + 1)}>  
        Haz clic  
      </button>  
    </div>  
  );  
}
```



¿QUÉ ES UN HOOK?

Los hooks son funciones especiales introducidas en React 16.8 que **permiten usar estado y otras características de React en componentes funcionales**.

Son una herramienta clave en React para **crear componentes funcionales con estado y ciclo de vida, simplificando el código y facilitando la reutilización de lógica**.



HOOKS MÁS USADOS



useState

Permite declarar y actualizar valores reactivos dentro de un componente.

Ejemplo: un contador que incrementa al hacer clic.

```
const [count, setCount] =  
  useState(0);
```

useEffect

Ejecuta código cuando el componente se monta o se actualiza. Ideal para llamar APIs o realizar tareas externas.

Ejemplo: hacer una petición a una API cuando se carga la página.

```
useEffect(() => {  
  fetchData();  
}, []);
```

useContext

Sirve para compartir datos globales entre componentes sin tener que pasar props manualmente por cada nivel.

Ejemplo: tema oscuro, idioma, datos del usuario logueado.

```
const theme =  
  useContext(ThemeContext);
```

ESTRUCTURA GENERAL DE UNA APP REACT



Una aplicación hecha con React tiene una **estructura de carpetas y archivos** que ayuda a mantener el código organizado, sobre todo cuando el proyecto crece.

El archivo **index.js** es el punto de entrada de la aplicación, donde React **renderiza el componente principal**, llamado **App.jsx**.

Carpetas y Archivos	Descripción
/mi-app	# Directorio raíz del proyecto
public/	# Archivos estáticos (favicon, index.html)
src/	# Código fuente principal
index.js	# Punto de entrada
App.jsx	# Componente principal
/components/	# Componentes reutilizables (Botón, Card, etc.)
/pages/	# Vistas de la app (Home, Login, Dashboard)
/hooks/	# Hooks personalizados
/context/	# Contextos globales
/services/	# Lógica para llamadas a APIs
/assets/	# Imágenes, íconos, fuentes
/styles/	# Archivos CSS o Tailwind config
package.json	# Dependencias y scripts



ESTILIZADO

1 CSS TRADICIONAL

La forma más básica es usar archivos CSS tradicionales, importándolos en los componentes. Sin embargo, esto puede causar conflictos cuando muchos componentes usan clases con los mismos nombres.

2 ESTILOS EN LÍNEA

Definidos como objetos JavaScript. Aunque son rápidos para casos simples o dinámicos, pueden volverse difíciles de manejar en proyectos grandes.

3 ESTILOS EN LÍNEA

Definidos como objetos JavaScript. Aunque son rápidos para casos simples o dinámicos, pueden volverse difíciles de manejar en proyectos grandes.

4 CSS MODULES

Permiten crear estilos aislados por componente. Usan archivos .module.css y evitan que las clases se mezclen.



ROUTING, CÓMO FUNCIONA EL ENRUTAMIENTO EN REACT

En aplicaciones con mas de una página, al hacer click en un enlace, se hace una solicitud al servidor para cargar completamente la nueva pagina HTML

En React el contenido se carga a través de los componentes creados. Por medio **React Router**, se intercepta la solicitud del cliente y se carga el contenido dinámicamente, haciendo que la pagina funcione como un SPA, mejorando la UX.





¿QUÉ ES REACT ROUTER?

Se trata de una colección de componentes de navegación, con esta librería vamos a obtener un enrutamiento dinámico gracias a los componentes, en otras palabras tenemos unas rutas que renderizan un componente.

En el archivo `index.html` se carga todo el código que se escriba en '`App.jsx`', siendo este nuestro componente principal, así tenemos solo un archivo HTML donde se renderiza el contenido de nuestra aplicación. De esta manera, no se crea un archivo HTML nuevo, sino que React Router nos ayuda a reescribir `index.html` de manera lógica.

EJEMPLO DE ROUTER

```
import { Routes, Route } from "react-router-dom"
import Inicio from "./Inicio"
import SobreNosotros from "./SobreNosotros"
import Contacto from "./Contacto"

function Aplicacion() {
  return (
    <div className="Aplicacion">
      <Routes>
        <Route path="/" element={ <Inicio /> } />
        <Route path="/sobre-nosotros" element={ <SobreNosotros /> } />
        <Route path="/contacto" element={ <Contacto /> } />
      </Routes>
    </div>
  );
}

export default Aplicacion;
```

CONSUMO DE APIS

¿QUE ES UNA API REST?

Por sus siglas en inglés, API significa “Interfaz de Programación de Aplicación”. Se trata de un medio que permite la comunicación programática entre diferentes aplicaciones y que devuelve una respuesta en tiempo real.

Cuando se hace una petición a través de una API REST, ésta envía a un endpoint, una representación del estado actual del recurso. Esta representación del estado puede tomar la forma de archivo JSON (JavaScript Object Notation), XML, o HTML.



COMO CONSUMIR APIs EN REACT

Es posible consumir APIs de distintas maneras en React, pero vamos a ver únicamente las dos formas más populares, por medio de **Axios** y **Fetch API**. Cada uno cuenta con características que pueden resultar positivas o negativas según el proyecto y las necesidades de nuestro proyecto.



FETCH API

La Fetch API es un método de JavaScript para obtener recursos de un servidor o de un endpoint de una API. No requiere instalación extra

AXIOS

Axios es un cliente HTTP basado en promesas que hace más simple el envío de peticiones HTTP asíncronas a endpoints REST. Si requiere instalación adicional



FETCH API

El método `fetch()` requiere de un argumento obligatorio que es la URL o dirección al recurso que queremos obtener. Luego, este se convierte en una promesa para que podamos manejar su éxito o error a través de los métodos `then()` y `catch()`. La respuesta por defecto es una respuesta HTTP estándar más que un JSON, pero podemos ver la información como un objeto JSON al usar el método `json()` incluida en la respuesta.

```
fetch('https://jsonplaceholder.typicode.com/posts?_limit=10')
  .then(response => response.json())
  .then(data => console.log(data));
```

FETCH API CON USE EFFECT

```
useEffect(() => {
  const fetchPost = async () => {
    const response = await fetch(
      'https://jsonplaceholder.typicode.com/posts?_limit=10'
    );
    const data = await response.json();
    console.log(data);
    setPosts(data);
  };
  fetchPost();
}, []);
```

AXIOS

A diferencia de Fetch, Axios no viene incluido en el navegador por lo que necesitamos instalarlo en nuestro proyecto para poder utilizarlo.

Una vez que lo instalado, podemos crear una instancia, esto no es necesario pero si recomendable ya que nos evita repeticiones innecesarias. Para crear una instancia, usamos el método **.create()**, que podemos usarlo para especificar información como la **URL** y posibles encabezados similar al siguiente ejemplo:

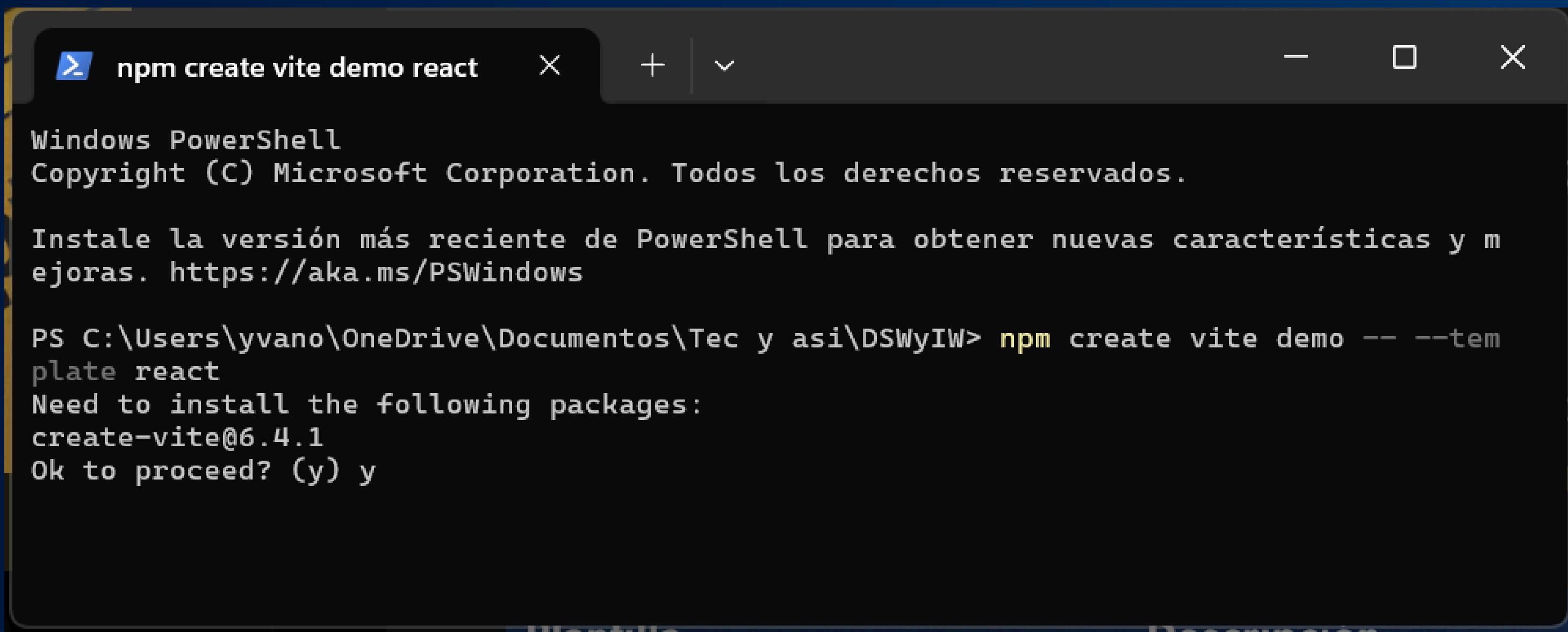
```
import axios from "axios";

const client = axios.create({
  baseURL: "https://jsonplaceholder.typicode.com/posts"
});
```

AXIOS CON USE EFFECT

```
useEffect(() => {  
  const fetchPost = async () => {  
    let response = await client.get('?_limit=10');  
    setPosts(response.data);  
  };  
  fetchPost();  
}, []);
```

EJEMPLO BASICO DE PROYECTO DEMO



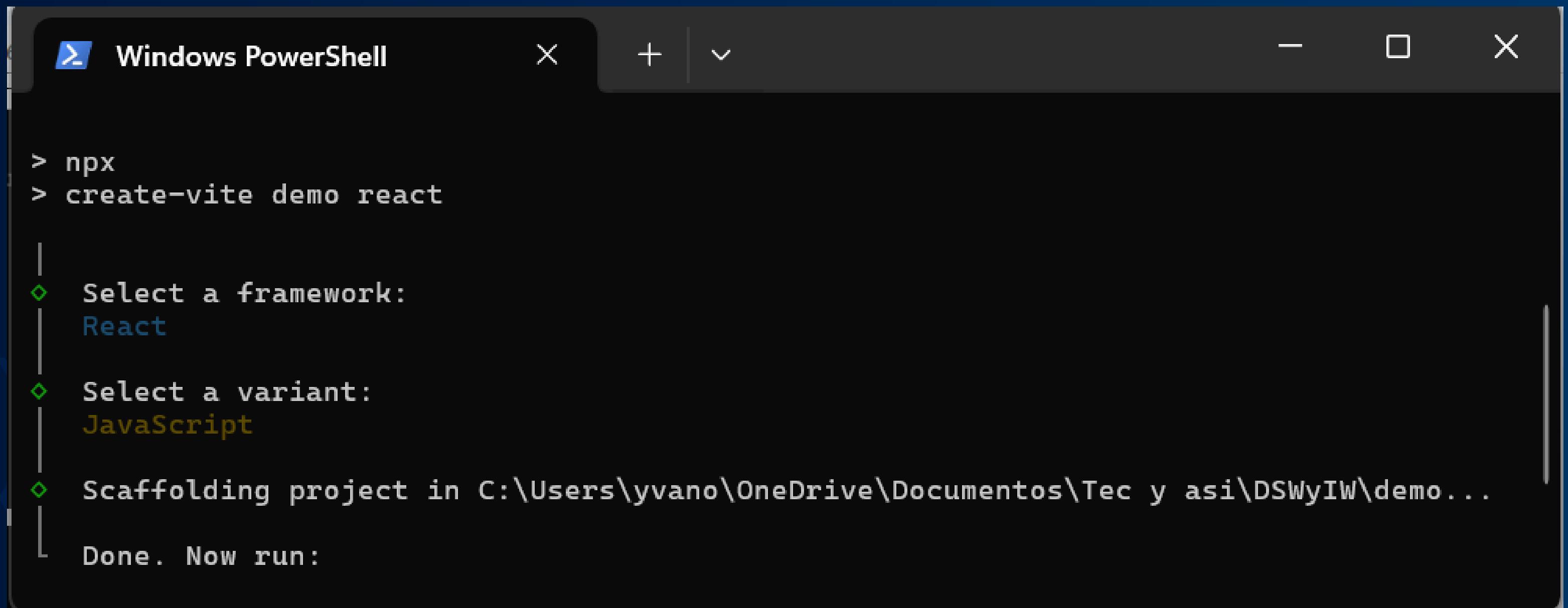
```
> npm create vite demo react
```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. <https://aka.ms/PSWindows>

```
PS C:\Users\yvano\OneDrive\Documentos\Tec y asi\DSWyIW> npm create vite demo -- --template react
Need to install the following packages:
create-vite@6.4.1
Ok to proceed? (y) y
```

EJEMPLO BASICO DE PROYECTO DEMO



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "npx create-vite demo react" being run, followed by a series of configuration steps for the new project:

- Select a framework: React
- Select a variant: JavaScript
- Scaffolding project in C:\Users\yvano\OneDrive\Documentos\Tec y asi\DSWyIW\demo...
- Done. Now run:

EJEMPLO BASICO DE PROYECTO DEMO

```
PS C:\Users\yvano\OneDrive\Documentos\Tec y asi\DSWyIW> cd demo
PS C:\Users\yvano\OneDrive\Documentos\Tec y asi\DSWyIW\demo> npm install
added 225 packages, and audited 226 packages in 37s
48 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\yvano\OneDrive\Documentos\Tec y asi\DSWyIW\demo> |
```

EJEMPLO BASICO DE PROYECTO DEMO

The image shows a split-screen view of a development environment. On the left, the VS Code Explorer sidebar displays the project structure for a 'DEMO' folder. The structure includes a 'node_modules' folder, a 'public' folder, and a 'src' folder containing 'assets', 'App.css', 'App.jsx', 'index.css', 'main.jsx', '.gitignore', 'eslint.config.js', 'index.html', 'package-lock.json', 'package.json', 'README.md', and 'vite.config.js'. The 'App.css' file is currently selected. On the right, a browser window shows the running application at 'localhost:5173'. The page features the Vite logo (a stylized lightning bolt) and the React logo (an atom symbol). The title 'Vite + React' is displayed. A button labeled 'count is 9' is visible. Below it, a message says 'Edit src/App.jsx and save to test HMR'. At the bottom, there is a link 'https://react.dev'.

THANK YOU
FOR YOUR ATTENTION