**Curso: Desarrollo Web - 12025-9989-031-A**

**LABORATORIO #5**



**Walter Rene Rosales Hurtarte   
Carné:** 9989-08-7438

# ¿Por qué usamos Vite + React en el desarrollo?

1. **React: Biblioteca para construir interfaces de usuario**
   * React es una de las tecnologías más populares para crear aplicaciones web modernas.
   * Permite trabajar con **componentes reutilizables**, lo que facilita mantener y escalar proyectos grandes.
   * Su enfoque declarativo hace que sea más sencillo describir “qué queremos ver en pantalla” y dejar que React gestione las actualizaciones de la interfaz.
   * Se integra fácilmente con librerías externas como **Chakra UI** para estilos, **React Router** para rutas, y herramientas de estado global (Redux, Context API).
2. **Vite: Entorno de desarrollo moderno**
   * Vite es un **bundler** y **servidor de desarrollo rápido** que reemplaza a herramientas más pesadas como Webpack.
   * Beneficios principales:
     + **Inicio rápido**: al correr npm run dev levanta el servidor casi instantáneamente.
     + **Hot Module Replacement (HMR)**: los cambios en el código se reflejan en el navegador sin recargar toda la aplicación.
     + **Optimización de build**: usa **esbuild** y **Rollup**, que generan aplicaciones rápidas y optimizadas para producción.
3. **Razones pedagógicas (nivel universitario)**
   * **Facilidad de aprendizaje**: React tiene una curva de aprendizaje amigable y mucha documentación.
   * **Adopción en la industria**: empresas reales lo usan masivamente, lo cual prepara a los estudiantes para el campo laboral.
   * **Proyectos modulares**: React fomenta la separación del código en pequeños componentes fáciles de probar.
   * **Eficiencia de desarrollo**: Vite permite que los estudiantes se concentren en aprender React, sin perder tiempo configurando entornos complejos.
4. **Ejemplo práctico en el laboratorio**
   * Usamos React para construir **pantallas dinámicas** (Login, Home, Productos).
   * Vite hace que el entorno se configure con un simple npm create vite@latest.

Repositorio Git  
[wallas2022/laboratorio06: Proyecto en React acceso a token y login](https://github.com/wallas2022/laboratorio06)

Estructura general del proyecto:

A screenshot of a computer

AI-generated content may be incorrect.

# Capturas de pantallas de consumo de apis

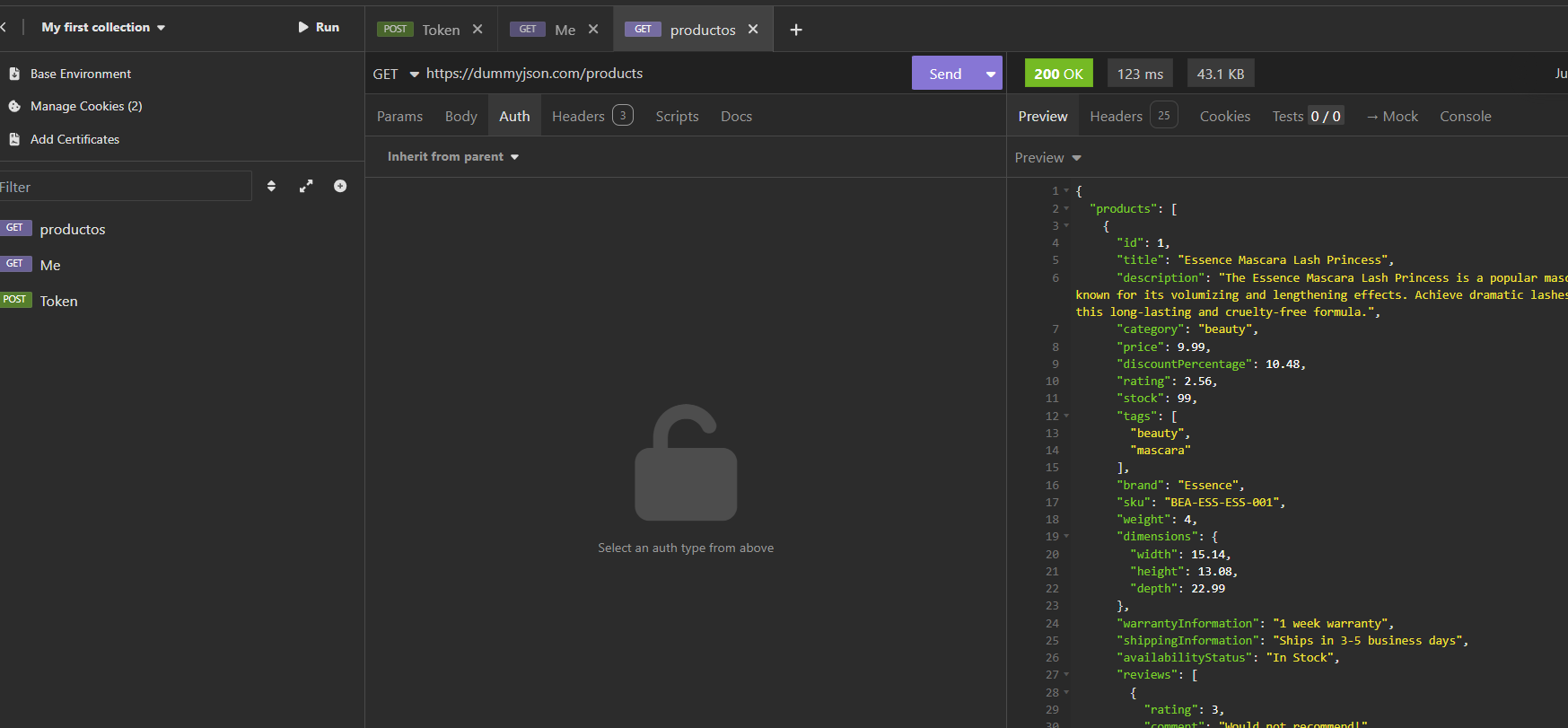
Captura de prueba de tocken

A screenshot of a computer

AI-generated content may be incorrect.

Captrua de api me  
  
A screenshot of a computer

AI-generated content may be incorrect.

Lista de productos  
  


Crear  
  
A screenshot of a computer

AI-generated content may be incorrect.

Actualizar  
  
A screenshot of a computer program

AI-generated content may be incorrect.

Eliminar  
  
A screenshot of a computer

AI-generated content may be incorrect.

Buscar con paginado y filtros

# A screenshot of a computer AI-generated content may be incorrect. 2.- Explicacion del código y capturas

// src/Login.jsx

import React, { useState } from 'react';

import {

  Box, Button, Flex, FormControl, FormLabel, Input,

  Heading, Text, Image, VStack

} from '@chakra-ui/react';

import { useNavigate } from 'react-router-dom';

const Login = () => {

  const [username, setUsername] = useState('emilys');      // demo

  const [password, setPassword] = useState('emilyspass');  // demo

  const [error, setError] = useState('');

  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleSubmit = async (e) => {

    e.preventDefault();

    setError('');

    setLoading(true);

    try {

      const r = await fetch('https://dummyjson.com/auth/login', {

        method: 'POST',

        headers: { 'Content-Type': 'application/json' },

        body: JSON.stringify({ username, password, expiresInMins: 30 })

      });

      if (!r.ok) {

        let msg = 'Error de autenticación';

        try {

          const err = await r.json();

          msg = err?.message || msg;

        } catch (\_) {}

        throw new Error(msg);

      }

      const data = await r.json();

      const { accessToken } = data || {};

      if (!accessToken) throw new Error('No se recibió accessToken');

      localStorage.setItem('token', accessToken);

      // Validar token

      const me = await fetch('https://dummyjson.com/auth/me', {

        headers: { Authorization: `Bearer ${accessToken}` },

      });

      if (!me.ok) throw new Error('No se pudo cargar el perfil');

      navigate('/home');

    } catch (err) {

      setError(err.message || 'Hubo un problema al iniciar sesión');

    } finally {

      setLoading(false);

    }

  };

  return (

    <Flex minH="100vh" align="center" justify="center" bg="gray.50">

      <Box

        as="form"

        onSubmit={handleSubmit}

        bg="white"

        p={8}

        rounded="xl"

        shadow="md"

        minW={{ base: '90%', sm: '400px' }}

        textAlign="center"

      >

        <VStack spacing={4} mb={6}>

          {/\* 🔹 Logo UMG (puedes reemplazar por tu archivo local en /assets) \*/}

          <Image

            src="https://play-lh.googleusercontent.com/PAgEDMao5gLi5N-9x-EdPIihJHe0CRqscma-BQPunQoV887HW58Wi8ccdAtU2UwBnwo=w480-h960-rw"

            alt="Logo UMG"

            boxSize="80px"

            objectFit="contain"

          />

          <Heading size="md">Laboratorio 05 – APIs DummyJSON</Heading>

        </VStack>

        <FormControl mb={4}>

          <FormLabel>Usuario</FormLabel>

          <Input

            value={username}

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

            placeholder="usuario"

          />

        </FormControl>

        <FormControl mb={4}>

          <FormLabel>Contraseña</FormLabel>

          <Input

            type="password"

            value={password}

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

            placeholder="contraseña"

          />

        </FormControl>

        {error && <Text color="red.500" mb={4}>{error}</Text>}

        <Button type="submit" w="full" colorScheme="blue" isLoading={loading}>

          Entrar

        </Button>

      </Box>

    </Flex>

  );

};

export default Login;

A screenshot of a computer

AI-generated content may be incorrect.

Codigo:  
// src/Login.jsx

import React, { useState } from 'react';

import {

  Box, Button, Flex, FormControl, FormLabel, Input,

  Heading, Text, Image, VStack

} from '@chakra-ui/react';

import { useNavigate } from 'react-router-dom';

const Login = () => {

  const [username, setUsername] = useState('emilys');      // demo

  const [password, setPassword] = useState('emilyspass');  // demo

  const [error, setError] = useState('');

  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleSubmit = async (e) => {

    e.preventDefault();

    setError('');

    setLoading(true);

    try {

      const r = await fetch('https://dummyjson.com/auth/login', {

        method: 'POST',

        headers: { 'Content-Type': 'application/json' },

        body: JSON.stringify({ username, password, expiresInMins: 30 })

      });

      if (!r.ok) {

        let msg = 'Error de autenticación';

        try {

          const err = await r.json();

          msg = err?.message || msg;

        } catch (\_) {}

        throw new Error(msg);

      }

      const data = await r.json();

      const { accessToken } = data || {};

      if (!accessToken) throw new Error('No se recibió accessToken');

      localStorage.setItem('token', accessToken);

      // Validar token

      const me = await fetch('https://dummyjson.com/auth/me', {

        headers: { Authorization: `Bearer ${accessToken}` },

      });

      if (!me.ok) throw new Error('No se pudo cargar el perfil');

      navigate('/home');

    } catch (err) {

      setError(err.message || 'Hubo un problema al iniciar sesión');

    } finally {

      setLoading(false);

    }

  };

  return (

    <Flex minH="100vh" align="center" justify="center" bg="gray.50">

      <Box

        as="form"

        onSubmit={handleSubmit}

        bg="white"

        p={8}

        rounded="xl"

        shadow="md"

        minW={{ base: '90%', sm: '400px' }}

        textAlign="center"

      >

        <VStack spacing={4} mb={6}>

          {/\* 🔹 Logo UMG (puedes reemplazar por tu archivo local en /assets) \*/}

          <Image

            src="https://play-lh.googleusercontent.com/PAgEDMao5gLi5N-9x-EdPIihJHe0CRqscma-BQPunQoV887HW58Wi8ccdAtU2UwBnwo=w480-h960-rw"

            alt="Logo UMG"

            boxSize="80px"

            objectFit="contain"

          />

          <Heading size="md">Laboratorio 05 – APIs DummyJSON</Heading>

        </VStack>

        <FormControl mb={4}>

          <FormLabel>Usuario</FormLabel>

          <Input

            value={username}

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

            placeholder="usuario"

          />

        </FormControl>

        <FormControl mb={4}>

          <FormLabel>Contraseña</FormLabel>

          <Input

            type="password"

            value={password}

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

            placeholder="contraseña"

          />

        </FormControl>

        {error && <Text color="red.500" mb={4}>{error}</Text>}

        <Button type="submit" w="full" colorScheme="blue" isLoading={loading}>

          Entrar

        </Button>

      </Box>

    </Flex>

  );

};

export default Login;

A screenshot of a computer

AI-generated content may be incorrect.  
  
  
CODIGO:  
  
**Pantalla Home – Explicación de componentes**

1. **Contenedor principal**
   * **Box**: se usa como un “div” con padding (p={8}), que da espacio interno y organiza los elementos en bloque.
2. **Encabezado**
   * **Heading**: título grande que muestra “Home”. Sirve como cabecera de la página principal después de hacer login.
3. **Estado de sesión**
   * **Text**: muestra un mensaje indicando si existe o no un token en localStorage.
     + Ejemplo: “Token presente: Sí” si hay sesión activa.
     + “Token presente: No” si no hay sesión (token borrado o no autenticado).
4. **Acciones principales (Botones)**
   * **Flex con gap={4}**: organiza los botones en fila con espacio entre ellos.
   * **Button (colorScheme="red") → Cerrar sesión**
     + Al hacer clic, ejecuta:
     + localStorage.removeItem('token');
     + navigate('/');
     + Esto **borra la sesión** (token) y redirige automáticamente al **Login**.
   * **Button (colorScheme="blue") → Ir a Productos**
     + Usa Link de react-router-dom para navegar a /productos.
     + Permite entrar al listado con búsqueda, paginación y CRUD de productos.

1. Estados principales en el consumo de APIs

**Loading (Cargando):**

Se activa cuando se inicia la petición al endpoint de la API. Durante este estado normalmente se muestra un spinner o un mensaje indicando que los datos están cargando.

**Success** (Éxito):

Ocurre cuando la API responde correctamente (`status 200 OK`). En este punto se procesan los datos recibidos y se muestran en la interfaz (por ejemplo, la lista de productos).

**Error (Fallo):**

Sucede cuando la API devuelve un error (ej. `401 Unauthorized`, `404 Not Found`) o hay un problema de red. En la interfaz se despliega un mensaje de error al usuario.

# Pagina de Productos

A screenshot of a computer

AI-generated content may be incorrect.

## Explicación de los componentes en la pantalla Productos

**Encabezado**

Heading (Chakra UI): “Productos” en la parte superior izquierda.

Se usa para indicar el título de la sección.

**Barra superior de acciones**

InputGroup + Input: Campo de búsqueda con placeholder “Buscar por título (DummyJSON)”.

Incluye SearchIcon (cuando está vacío) o CloseIcon (cuando hay texto, para limpiar).

Select: Dropdown “Por página: 10” que permite seleccionar cuántos registros ver por página (5, 10, 15, 20).

Button (outline): “Volver a Home”, que navega a la pantalla Home.

Button (colorScheme="blue"): “Crear Producto”, abre el formulario de creación.

Indicadores de paginación

Badge (colorScheme="purple"): muestra el total de registros (TOTAL: 194).

Badge (colorScheme="teal"): muestra la página actual (Página 1 de 20).

Button: “Anterior” (deshabilitado en la primera página).

Button: “Siguiente” (activo si hay más páginas).

## Tabla de productos (Table)

**Encabezado (Thead y Th):**

ID

Nombre

Precio

Acciones

Cuerpo (Tbody y Tr con Td):

Cada fila representa un producto traído del API DummyJSON.

**Ejemplo:**

ID: 1

Nombre: Essence Mascara Lash Princess

Precio: $9.99

## Acciones en cada fila

Button (colorScheme="yellow"): “Editar” → lleva a /productos/:id/edit  
Button (colorScheme="red"): “Eliminar” → lleva a /productos/:id/delete.

Codigo:

// src/Productos.jsx

import React, { useEffect, useMemo, useState } from "react";

import {

  Box, Button, Flex, Heading, Spinner, Table, Tbody, Td, Th, Thead, Tr, Text,

  Input, InputGroup, InputRightElement, IconButton, Select, HStack, Spacer, Badge

} from "@chakra-ui/react";

import { Link } from "react-router-dom";

import { CloseIcon, SearchIcon } from "@chakra-ui/icons";

const Productos = () => {

  const [productos, setProductos] = useState([]);

  const [total, setTotal] = useState(0);

  const [loading, setLoading] = useState(false);

  const [error, setError] = useState("");

  // UI state

  const [query, setQuery] = useState("");

  const [debouncedQ, setDebouncedQ] = useState(""); // debounce para la búsqueda

  const [page, setPage] = useState(1);

  const [limit, setLimit] = useState(10);

  const pages = useMemo(() => Math.max(1, Math.ceil(total / limit || 1)), [total, limit]);

  const skip = useMemo(() => (page - 1) \* limit, [page, limit]);

  // Debounce de la búsqueda

  useEffect(() => {

    const t = setTimeout(() => {

      setPage(1);          // al cambiar búsqueda, volver a página 1

      setDebouncedQ(query.trim());

    }, 400);

    return () => clearTimeout(t);

  }, [query]);

  const fetchProductos = async () => {

    setLoading(true);

    setError("");

    try {

      const base = "https://dummyjson.com";

      const url = debouncedQ

        ? `${base}/products/search?q=${encodeURIComponent(debouncedQ)}&limit=${limit}&skip=${skip}`

        : `${base}/products?limit=${limit}&skip=${skip}`;

      const res = await fetch(url);

      if (!res.ok) throw new Error("No se pudieron cargar los productos");

      const data = await res.json();

      // DummyJSON retorna { products, total, skip, limit }

      setProductos(data.products || []);

      setTotal(Number(data.total ?? 0));

    } catch (err) {

      setError(err.message || "Error al cargar productos");

      setProductos([]);

      setTotal(0);

    } finally {

      setLoading(false);

    }

  };

  useEffect(() => {

    fetchProductos();

    // eslint-disable-next-line react-hooks/exhaustive-deps

  }, [debouncedQ, page, limit]);

  const canPrev = page > 1;

  const canNext = page < pages;

  const resetSearch = () => {

    setQuery("");

    setDebouncedQ("");

    setPage(1);

  };

  return (

    <Box p={6}>

      <Flex direction={{ base: "column", md: "row" }} gap={3} align={{ md: "center" }} mb={4}>

        <Heading size="lg">Productos</Heading>

        <Spacer />

        <HStack spacing={3}>

          <Button as={Link} to="/home" variant="outline">Volver a Home</Button>

          <Button as={Link} to="/productos/create" colorScheme="blue">Crear Producto</Button>

        </HStack>

      </Flex>

      <Flex direction={{ base: "column", md: "row" }} gap={3} mb={4} align={{ md: "center" }}>

        <InputGroup maxW="420px">

          <Input

            placeholder="Buscar por título (DummyJSON)"

            value={query}

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

          />

          <InputRightElement>

            {query ? (

              <IconButton

                aria-label="Limpiar búsqueda"

                size="sm"

                variant="ghost"

                icon={<CloseIcon boxSize={2.5} />}

                onClick={resetSearch}

              />

            ) : (

              <SearchIcon opacity={0.6} />

            )}

          </InputRightElement>

        </InputGroup>

        <HStack>

          <Text>Por página:</Text>

          <Select

            value={limit}

            onChange={(e) => { setLimit(Number(e.target.value)); setPage(1); }}

            width="90px"

          >

            {[5,10,15,20].map(n => <option key={n} value={n}>{n}</option>)}

          </Select>

        </HStack>

        <Spacer />

        <HStack>

          <Badge colorScheme="purple">Total: {total}</Badge>

          <Badge colorScheme="teal">Página {page} de {pages}</Badge>

          <Button onClick={() => setPage(p => p - 1)} isDisabled={!canPrev}>Anterior</Button>

          <Button onClick={() => setPage(p => p + 1)} isDisabled={!canNext}>Siguiente</Button>

        </HStack>

      </Flex>

      {loading && <Spinner />}

      {error && <Text color="red.500" mb={3}>{error}</Text>}

      {!loading && !error && productos.length === 0 && (

        <Text>No hay productos para mostrar.</Text>

      )}

      {productos.length > 0 && (

        <Table variant="striped" colorScheme="gray">

          <Thead>

            <Tr>

              <Th>ID</Th>

              <Th>Nombre</Th>

              <Th isNumeric>Precio</Th>

              <Th>Acciones</Th>

            </Tr>

          </Thead>

          <Tbody>

            {productos.map((p) => (

              <Tr key={p.id}>

                <Td>{p.id}</Td>

                <Td>{p.title}</Td>

                <Td isNumeric>{p.price != null ? `$${p.price}` : "-"}</Td>

                <Td>

                  <Button

                    as={Link}

                    to={`/productos/${p.id}/edit`}

                    size="sm"

                    mr={2}

                    colorScheme="yellow"

                  >

                    Editar

                  </Button>

                  <Button

                    as={Link}

                    to={`/productos/${p.id}/delete`}

                    size="sm"

                    colorScheme="red"

                  >

                    Eliminar

                  </Button>

                </Td>

              </Tr>

            ))}

          </Tbody>

        </Table>

      )}

    </Box>

  );

};

export default Productos;

CREAR NUEVO PRODUCTO

A screenshot of a computer

AI-generated content may be incorrect.

CODIGO:

// src/ProductosCreate.jsx

import React, { useState } from "react";

import {

Box, Button, FormControl, FormLabel, Input, Heading, Textarea,

Text, Flex, SimpleGrid, NumberInput, NumberInputField, useToast

} from "@chakra-ui/react";

import { useNavigate, Link } from "react-router-dom";

const ProductosCreate = () => {

const [title, setTitle] = useState("");

const [price, setPrice] = useState("");

const [brand, setBrand] = useState("");

const [category, setCategory] = useState("");

const [description, setDescription] = useState("");

const [saving, setSaving] = useState(false);

const [error, setError] = useState("");

const toast = useToast();

const navigate = useNavigate();

const handleSubmit = async (e) => {

e.preventDefault();

setError("");

if (!title.trim()) {

setError("El nombre (title) es obligatorio");

return;

}

setSaving(true);

try {

const payload = {

title: title.trim(),

// DummyJSON acepta campos extra; price opcional

...(price !== "" ? { price: Number(price) } : {}),

...(brand.trim() ? { brand: brand.trim() } : {}),

...(category.trim() ? { category: category.trim() } : {}),

...(description.trim() ? { description: description.trim() } : {}),

};

const res = await fetch("https://dummyjson.com/products/add", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(payload),

});

if (!res.ok) throw new Error("No se pudo crear el producto");

await res.json(); // respuesta simulada

toast({

title: "Producto creado (simulado).",

status: "success",

duration: 2000,

isClosable: true,

});

navigate("/productos");

} catch (e) {

setError(e.message || "Error al crear el producto");

} finally {

setSaving(false);

}

};

return (

<Box p={6}>

<Flex justify="space-between" align="center" mb={4}>

<Heading size="lg">Crear Producto</Heading>

<Button as={Link} to="/productos" variant="outline">

Volver

</Button>

</Flex>

{error && <Text color="red.500" mb={4}>{error}</Text>}

<Box as="form" onSubmit={handleSubmit} maxW="720px">

<SimpleGrid columns={{ base: 1, md: 2 }} spacing={4}>

<FormControl isRequired>

<FormLabel>Nombre (title)</FormLabel>

<Input

value={title}

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

placeholder="Ej: TESLA X"

/>

</FormControl>

<FormControl>

<FormLabel>Precio</FormLabel>

<NumberInput min={0} value={price} onChange={(v) => setPrice(v)}>

<NumberInputField placeholder="Ej: 999" />

</NumberInput>

</FormControl>

<FormControl>

<FormLabel>Marca</FormLabel>

<Input

value={brand}

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

placeholder="Ej: Tesla"

/>

</FormControl>

<FormControl>

<FormLabel>Categoría</FormLabel>

<Input

value={category}

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

placeholder="Ej: autos"

/>

</FormControl>

</SimpleGrid>

<FormControl mt={4}>

<FormLabel>Descripción</FormLabel>

<Textarea

value={description}

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

placeholder="Detalles del producto"

/>

</FormControl>

<Button

type="submit"

colorScheme="blue"

mt={6}

isLoading={saving}

isDisabled={!title.trim()}

>

Guardar

</Button>

</Box>

</Box>

);

};

export default ProductosCreate;

EDITAR

A screenshot of a computer

AI-generated content may be incorrect.

CODIGO:

// src/ProductosEdit.jsx

import React, { useEffect, useState } from "react";

import {

  Box, Button, FormControl, FormLabel, Input, Heading, Text, Spinner, Flex

} from "@chakra-ui/react";

import { useNavigate, useParams, Link } from "react-router-dom";

const ProductosEditar = () => {

  const { id } = useParams();

  const navigate = useNavigate();

  const [title, setTitle] = useState("");

  const [price, setPrice] = useState("");

  const [error, setError] = useState("");

  const [loading, setLoading] = useState(true);

  const [saving, setSaving] = useState(false);

  // Cargar datos actuales

  useEffect(() => {

    const load = async () => {

      setError("");

      try {

        const res = await fetch(`https://dummyjson.com/products/${id}`);

        if (!res.ok) throw new Error("No se pudo cargar el producto");

        const data = await res.json();

        setTitle(data.title ?? "");

        setPrice(data.price ?? "");

      } catch (e) {

        setError(e.message || "Error cargando producto");

      } finally {

        setLoading(false);

      }

    };

    load();

  }, [id]);

  const handleSubmit = async (e) => {

    e.preventDefault();

    setSaving(true);

    setError("");

    try {

      const res = await fetch(`https://dummyjson.com/products/${id}`, {

        method: "PUT",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify({

          title,

          price: price === "" ? undefined : Number(price),

        }),

      });

      if (!res.ok) throw new Error("No se pudo actualizar el producto");

      await res.json();

      navigate("/productos");

    } catch (e) {

      setError(e.message || "Error al guardar");

    } finally {

      setSaving(false);

    }

  };

  if (loading) return <Spinner ml={6} mt={6} />;

  return (

    <Box p={6}>

      <Flex justify="space-between" align="center" mb={4}>

        <Heading size="lg">Editar Producto #{id}</Heading>

        <Button as={Link} to="/productos" variant="outline">Volver</Button>

      </Flex>

      {error && <Text color="red.500" mb={4}>{error}</Text>}

      <Box as="form" onSubmit={handleSubmit} maxW="520px">

        <FormControl mb={4} isRequired>

          <FormLabel>Nombre</FormLabel>

          <Input value={title} onChange={(e) => setTitle(e.target.value)} />

        </FormControl>

        <FormControl mb={6}>

          <FormLabel>Precio</FormLabel>

          <Input

            type="number"

            value={price}

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

            placeholder="Ej: 999"

          />

        </FormControl>

        <Button type="submit" colorScheme="blue" isLoading={saving}>

          Guardar cambios

        </Button>

      </Box>

    </Box>

  );

};

export default ProductosEditar;

ELIMINAR

A screenshot of a computer

AI-generated content may be incorrect.

// src/ProductosDelete.jsx

import React, { useEffect, useState } from "react";

import {

  Box, Button, Heading, Text, Spinner, Flex, Stack

} from "@chakra-ui/react";

import { useNavigate, useParams, Link } from "react-router-dom";

const ProductosDelete = () => {

  const { id } = useParams();

  const navigate = useNavigate();

  const [prod, setProd] = useState(null);

  const [loading, setLoading] = useState(true);

  const [deleting, setDeleting] = useState(false);

  const [error, setError] = useState("");

  useEffect(() => {

    const load = async () => {

      setError("");

      try {

        const res = await fetch(`https://dummyjson.com/products/${id}`);

        if (!res.ok) throw new Error("No se pudo cargar el producto");

        const data = await res.json();

        setProd(data);

      } catch (e) {

        setError(e.message || "Error cargando producto");

      } finally {

        setLoading(false);

      }

    };

    load();

  }, [id]);

  const handleDelete = async () => {

    setDeleting(true);

    setError("");

    try {

      const res = await fetch(`https://dummyjson.com/products/${id}`, {

        method: "DELETE",

      });

      if (!res.ok) throw new Error("No se pudo eliminar (simulado)");

      await res.json();

      navigate("/productos");

    } catch (e) {

      setError(e.message || "Error al eliminar");

    } finally {

      setDeleting(false);

    }

  };

  if (loading) return <Spinner ml={6} mt={6} />;

  return (

    <Box p={6}>

      <Flex justify="space-between" align="center" mb={4}>

        <Heading size="lg">Eliminar Producto #{id}</Heading>

        <Button as={Link} to="/productos" variant="outline">Cancelar</Button>

      </Flex>

      {error && <Text color="red.500" mb={4}>{error}</Text>}

      {prod ? (

        <Box borderWidth="1px" rounded="md" p={4} maxW="640px">

          <Stack spacing={2} mb={4}>

            <Text><b>Nombre:</b> {prod.title}</Text>

            {prod.price != null && <Text><b>Precio:</b> ${prod.price}</Text>}

            {prod.brand && <Text><b>Marca:</b> {prod.brand}</Text>}

            {prod.category && <Text><b>Categoría:</b> {prod.category}</Text>}

          </Stack>

          <Text mb={4} color="red.600">

            ¿Confirmas que deseas eliminar este producto? (Operación simulada)

          </Text>

          <Button colorScheme="red" onClick={handleDelete} isLoading={deleting}>

            Sí, eliminar

          </Button>

        </Box>

      ) : (

        <Text>No se encontró el producto.</Text>

      )}

    </Box>

  );

};

export default ProductosDelete;

# **Conclusión para el informe:**

Usamos **Vite + React** porque juntos brindan un **entorno de desarrollo rápido, modular y eficiente**, ideal tanto para fines educativos como para proyectos reales. React aporta la **estructura y modularidad** para la interfaz, mientras que Vite ofrece un **flujo de trabajo ágil y moderno** que optimiza la experiencia del desarrollador.