Carta de Productos - Proyecto React

Autor: Salvatore De Rosa Vega
Fecha: 4/11/2025

1. Introducción

Este documento describe el funcionamiento del proyecto "Carta-Cafetería", desarrollado en React con Vite. Implementa componentes modulares, estados locales y comunicación mediante props. Utiliza Node.js con PDFKit para generar documentación técnica automática.

2. Estructura del Script

El script generate-pdf.cjs produce un PDF técnico uniforme. Importa fs, path y pdfkit, crea un flujo de escritura hacia un archivo físico y documenta los componentes React, incrustando el código fuente y su explicación.

3. Función addComponent()

Automatiza la inserción de secciones. Lee un archivo fuente, lo inserta con fuente Courier y agrega una descripción técnica. Si el archivo no existe, incluye una nota de error.

4. Rutas base

Se definen rutas mediante path.join, apuntando a la estructura del proyecto en src/App. Desde ahí se derivan subrutas a Conteiner, Category y Product.

5. Componente App.jsx

```
import React, { useState, useEffect } from "react";
import "./App.css";
import Conteiner from "./Conteiner/Conteiner";

const API_BASE = "https://jlorenzo.ddns.net/carta_restaurante";
const USER_ID = 1;

export default function App() {
  const [categorias, setCategorias] = useState([]);

  async function fetchCategorias() {
    try {
      const res = await fetch(`${API_BASE}/categorias/?usuario_id=${USER_ID}`);
      if (!res.ok) throw new Error(`HTTP ${res.status}`);
      const json = await res.json();

      // Extraer data correctamente
      const arr = json.data;
```

```
    if (!Array.isArray(arr)) {
      console.error("Formato inesperado en /categorias:", json);
      setCategorias([]);
      return;
    }

    const categoriasNormalizadas = arr.map((c) => ({
      id: c.id,
      nombre: c.nombre,
      imagen: "",
      product: [],
    }));

    setCategorias(categoriasNormalizadas);

    for (const cat of categoriasNormalizadas) {
      await fetchProductos(cat.id);
    }
  } catch (err) {
    console.error("Error fetchCategorias:", err);
    setCategorias([]);
  }
}

async function fetchProductos(catId) {
  try {
    const res = await fetch(
      `${API_BASE}/productos/${catId}?usuario_id=${USER_ID}`
    );
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    const json = await res.json();

    const arr = Array.isArray(json.data)
      ? json.data
      : Array.isArray(json)
      ? json
      : [];

    if (!Array.isArray(arr)) {
      console.error("Formato inesperado en /productos:", json);
      return;
    }

    const productosNormalizados = arr.map((p) => ({
      id: p.id,
      name: p.nombre,
      price: parseFloat(p.precio),
    }));

    setCategorias((prev) =>
      prev.map((c) =>
        c.id === catId ? { ...c, product: productosNormalizados } : c
      )
    );
  } catch (err) {
```

```
    console.error("Error fetchProductos:", err);
  }
}

// ---------- CATEGORÍAS ----------
async function addCategoria({ category }) {
  const body = { usuario_id: USER_ID, nombre: category };
  try {
    const res = await fetch(`${API_BASE}/categorias/`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    await fetchCategorias();
  } catch (err) {
    console.error("Error addCategoria:", err);
  }
}

async function editCategoria(id, newName) {
  const body = { usuario_id: USER_ID, nombre: newName, orden: 1 };
  try {
    const res = await fetch(`${API_BASE}/categorias/${id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    await fetchCategorias();
  } catch (err) {
    console.error("Error editCategoria:", err);
  }
}

async function deleteCategoria(id) {
  const body = { usuario_id: USER_ID };
  try {
    const res = await fetch(`${API_BASE}/categorias/${id}`, {
      method: "DELETE",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    await fetchCategorias();
  } catch (err) {
    console.error("Error deleteCategoria:", err);
  }
}

// ---------- PRODUCTOS ----------
async function addProducto(catId, { name, price }) {
  const body = {
    usuario_id: USER_ID,
    nombre: name,
```

```
    precio: parseFloat(price),
    orden: 1,
  };
  try {
    const res = await fetch(`${API_BASE}/productos/${catId}`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    await fetchProductos(catId);
  } catch (err) {
    console.error("Error addProducto:", err);
  }
}

async function editProducto(catId, prodId, { name, price }) {
  const body = {
    usuario_id: USER_ID,
    nombre: name,
    precio: parseFloat(price),
  };
  try {
    const res = await fetch(`${API_BASE}/productos/${prodId}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    await fetchProductos(catId);
  } catch (err) {
    console.error("Error editProducto:", err);
  }
}

async function deleteProducto(catId, prodId) {
  const body = { usuario_id: USER_ID };
  try {
    const res = await fetch(`${API_BASE}/productos/${prodId}`, {
      method: "DELETE",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(body),
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    await fetchProductos(catId);
  } catch (err) {
    console.error("Error deleteProducto:", err);
  }
}

useEffect(() => {
  fetchCategorias();
}, []);

useEffect(() => {
```

```
    console.log("Categorías cargadas:", categorias);
  }, [categorias]);

  return (
    <div className="contenedor">
      <Conteiner
        categorias={categorias}
        onAddCategoria={addCategoria}
        onEditCategoria={editCategoria}
        onDeleteCategoria={deleteCategoria}
        onAddProducto={addProducto}
        onEditProducto={editProducto}
        onDeleteProducto={deleteProducto}
      />
    </div>
  );
}
```

Componente raíz. Gestiona el estado global "categorias" mediante useState, implementa las funciones CRUD con la API REST remota y pasa los métodos como props a Conteiner.

6. Componente Conteiner.jsx

```
import { useState } from "react";Ð
import "./Conteiner.css";Ð
import Header from "./Header/Header";Ð
import Spacer from "./Spacer/Spacer";Ð
import Category from "./Category/Category";Ð
import Footer from "./Footer/Footer";Ð
Ð
export default function Conteiner({Ð
  categorias,Ð
  onAddCategoria,Ð
  onEditCategoria,Ð
  onDeleteCategoria,Ð
  onAddProducto,Ð
  onEditProducto,Ð
  onDeleteProducto,Ð
}) {Ð
  const [newCatName, setNewCatName] = useState("");Ð
  const [newCatImagen, setNewCatImagen] = useState("");Ð
Ð
  function handleAddCategoria(e) {Ð
    e.preventDefault();Ð
    if (!newCatName.trim()) return;Ð
    onAddCategoria({Ð
      category: newCatName.trim(),Ð
      imagen: newCatImagen.trim() || undefined,Ð
    });Ð
    setNewCatName("");Ð
    setNewCatImagen("");Ð
  }Ð
Ð
```

```
  return (Đ
    <>Đ
      <Header />Đ
      <Spacer />Đ
Đ
      <section className="add-category-section">Đ
        <form onSubmit={handleAddCategoria}>Đ
          <inputĐ
            placeholder="Nombre categoría"Đ
            value={newCatName}Đ
            onChange={(e) => setNewCatName(e.target.value)}Đ
          />Đ
          <inputĐ
            placeholder="URL imagen (opcional)"Đ
            value={newCatImagen}Đ
            onChange={(e) => setNewCatImagen(e.target.value)}Đ
          />Đ
          <button type="submit">Agregar categoría</button>Đ
        </form>Đ
      </section>Đ
Đ
      <Spacer />Đ
      {categorias.map((cat) => (Đ
        <CategoryĐ
          key={cat.id}Đ
          cat={cat}Đ
          onEditCategoria={onEditCategoria}Đ
          onDeleteCategoria={onDeleteCategoria}Đ
          onAddProducto={onAddProducto}Đ
          onEditProducto={onEditProducto}Đ
          onDeleteProducto={onDeleteProducto}Đ
        />Đ
      ))}Đ
      <Spacer />Đ
      <Footer />Đ
    </>Đ
  );Đ
}Đ
```

Contenedor principal. Recibe categorías y métodos CRUD desde App.jsx.
Incluye el formulario para añadir categorías y renderiza Category.jsx por cada una.

7. Componente Header.jsx

```
import './Header.css'Đ
Đ
export default function Header (){Đ
    return(Đ
        <header>Đ
            <h1 className='h1-header'>CAMPER CAFE</h1>Đ
            <h2 className='h2-header'>Est. 2020</h2>Đ
        </header>Đ
    )Đ
}
```

Encabezado estático que muestra el nombre del establecimiento. Sin lógica funcional.

8. Componente Footer.jsx

```
import './Footer.css'Ð
Ð
export default function Footer(){Ð
    return(Ð
        <footer>Ð
            <a className='a-footer'href="www.ejemplo.com">Visit our website</a>Ð
            <p className="p-footer">123 Free Code Camp Drive</p>Ð
        </footer>Ð
    )Ð
}
```

Pie de página estático con contacto o información adicional.

9. Componente Spacer.jsx

```
import './Spacer.css'Ð
Ð
export default function Spacer(){Ð
    return(Ð
        <hr />Ð
    )Ð
}
```

Elemento de separación visual representado mediante <hr />.

10. Componente Category.jsx

```
import React from "react";Ð
import "./Category.css";Ð
import CategoryHeader from "./CategoryHeader";Ð
import AddProductForm from "./AddProductForm";Ð
import ProductList from "./ProductList";Ð
Ð
export default function Category({Ð
  cat,Ð
  onEditCategoria,Ð
  onDeleteCategoria,Ð
  onAddProducto,Ð
  onEditProducto,Ð
  onDeleteProducto,Ð
}) {Ð
  return (Ð
    <main className="main-category">Ð
      <CategoryHeaderÐ
        cat={cat}Ð
        onEditCategoria={onEditCategoria}Ð
        onDeleteCategoria={onDeleteCategoria}Ð
      />Ð
Ð
```

```
      {cat.imagen && <img src={cat.imagen} alt={cat.nombre} width="120" />}Ð
Ð
      <AddProductForm catId={cat.id} onAddProducto={onAddProducto} />Ð
Ð
      <ProductListÐ
        productos={Array.isArray(cat.product) ? cat.product : []}Ð
        catId={cat.id}Ð
        onEditProducto={onEditProducto}Ð
        onDeleteProducto={onDeleteProducto}Ð
      />Ð
    </main>Ð
  );Ð
}Ð
```

Representa una categoría de productos. Permite editar o eliminar la categoría y añadir productos nuevos. Contiene CategoryHeader, AddProductForm y ProductList.

11. Componente CategoryHeader.jsx

```
import React, { useState } from "react";Ð
Ð
export default function CategoryHeader({Ð
  cat,Ð
  onEditCategoria,Ð
  onDeleteCategoria,Ð
}) {Ð
  const [editing, setEditing] = useState(false);Ð
  const [newName, setNewName] = useState(cat.nombre);Ð
Ð
  function handleSave() {Ð
    if (!newName.trim()) return;Ð
    onEditCategoria(cat.id, newName.trim());Ð
    setEditing(false);Ð
  }Ð
Ð
  function handleDelete() {Ð
    const res = onDeleteCategoria(cat.id);Ð
    if (res && res.blocked) {Ð
      const confirmForced = window.confirm(Ð
        `La categoría tiene ${res.count} producto(s). ¿Eliminar igualmente?`Ð
      );Ð
      if (confirmForced) onDeleteCategoria(cat.id, { force: true });Ð
    }Ð
  }Ð
Ð
  return (Ð
    <div className="category-header">Ð
      {!editing ? (Ð
        <>Ð
          <h2 className="h2-category">{cat.nombre.toUpperCase()}</h2>Ð
          <div className="category-actions">Ð
            <button onClick={() => setEditing(true)}>Editar</button>Ð
            <button onClick={handleDelete}>Borrar</button>Ð
          </div>Ð
```

```
        </>Đ
      ) : (Đ
        <div className="edit-name">Đ
          <input value={newName} onChange={(e) => setNewName(e.target.value)} />Đ
          <button onClick={handleSave}>Guardar</button>Đ
          <button onClick={() => setEditing(false)}>Cancelar</button>Đ
        </div>Đ
      )}Đ
    </div>Đ
  );Đ
}Đ
```

Encargado de mostrar el encabezado de una categoría.
Permite editar el nombre de la categoría o eliminarla. Implementa confirmación si la categoría contiene productos antes de su eliminación.

12. Componente AddProductForm.jsx

```
import React, { useState } from "react";Đ
Đ
export default function AddProductForm({ catId, onAddProducto }) {Đ
  const [name, setName] = useState("");Đ
  const [price, setPrice] = useState("");Đ
Đ
  function handleSubmit(e) {Đ
    e.preventDefault();Đ
    if (!name.trim() || price === "") return;Đ
    onAddProducto(catId, { name: name.trim(), price: parseFloat(price) });Đ
    setName("");Đ
    setPrice("");Đ
  }Đ
Đ
  return (Đ
    <section className="add-product">Đ
      <form onSubmit={handleSubmit}>Đ
        <inputĐ
          placeholder="Nombre producto"Đ
          value={name}Đ
          onChange={(e) => setName(e.target.value)}Đ
        />Đ
        <inputĐ
          placeholder="Precio"Đ
          type="number"Đ
          step="0.01"Đ
          value={price}Đ
          onChange={(e) => setPrice(e.target.value)}Đ
        />Đ
        <button type="submit">Agregar producto</button>Đ
      </form>Đ
    </section>Đ
  );Đ
}Đ
```

Formulario controlado para agregar nuevos productos.

Gestiona los estados "name" y "price" y llama a la función onAddProducto pasando los datos al componente padre Category.

13. Componente ProductList.jsx

```
import React from "react";Ð
import Product from "./Product/Product";Ð
Ð
export default function ProductList({Ð
  productos,Ð
  catId,Ð
  onEditProducto,Ð
  onDeleteProducto,Ð
}) {Ð
  if (!Array.isArray(productos)) productos = [];Ð
Ð
  return (Ð
    <ul className="product-list">Ð
      {productos.map((p) => (Ð
        <ProductÐ
          key={p.id}Ð
          prod={p}Ð
          catId={catId}Ð
          onEditProducto={onEditProducto}Ð
          onDeleteProducto={onDeleteProducto}Ð
        />Ð
      ))}Ð
    </ul>Ð
  );Ð
}Ð
```

Renderiza una lista de productos dentro de cada categoría.
Mapea cada elemento del array productos hacia el componente Product, pasando las funciones CRUD correspondientes como props.

14. Componente Product.jsx

```
import React, { useState } from "react";Ð
import "./Product.css";Ð
Ð
export default function Product({Ð
  prod,Ð
  catId,Ð
  onEditProducto,Ð
  onDeleteProducto,Ð
}) {Ð
  const [editing, setEditing] = useState(false);Ð
  const [name, setName] = useState(prod.name);Ð
  const [price, setPrice] = useState(prod.price);Ð
Ð
  function saveEdit() {Ð
    if (!name.trim() || price === "") return;Ð
    onEditProducto(catId, prod.id, { name: name.trim(), price: Number(price) });Ð
```

```
    setEditing(false);Đ
  }Đ
Đ
  function handleDelete() {Đ
    const ok = window.confirm(`Eliminar producto "${prod.name}"?`);Đ
    if (ok) onDeleteProducto(catId, prod.id);Đ
  }Đ
Đ
  return (Đ
    <li className="product-item">Đ
      {!editing ? (Đ
        <>Đ
          <span className="name">{prod.name}</span>Đ
          <span className="price">{Number(prod.price).toFixed(2)}</span>Đ
          <div className="actions">Đ
            <buttonĐ
              onClick={() => {Đ
                setEditing(true);Đ
                setName(prod.name);Đ
                setPrice(prod.price);Đ
              }}Đ
            >Đ
              EditarĐ
            </button>Đ
            <button onClick={handleDelete}>Borrar</button>Đ
          </div>Đ
        </>Đ
      ) : (Đ
        <div className="edit-product">Đ
          <input value={name} onChange={(e) => setName(e.target.value)} />Đ
          <inputĐ
            type="number"Đ
            step="0.01"Đ
            value={price}Đ
            onChange={(e) => setPrice(e.target.value)}Đ
          />Đ
          <button onClick={saveEdit}>Guardar</button>Đ
          <button onClick={() => setEditing(false)}>Cancelar</button>Đ
        </div>Đ
      )}Đ
    </li>Đ
  );Đ
}Đ
```

Representa un producto individual. Permite editar o eliminar el producto, usando estados internos para alternar entre modo edición y visualización.

15. Repositorio GitHub

https://github.com/S4lv4-code/Carta-cafeteria

16. Conclusión técnica

El proyecto combina React para la interfaz interactiva con PDFKit para la documentación automatizada.
La modularidad y el uso de API REST aseguran un mantenimiento limpio y escalabilidad del sistema.