Facultad de Ingeniería

02/11/2024

Manual Técnico D&P Petshop

Didvin Nohel Estrada Pineda

Pablo Rodolfo Alexander Flores Mollinedo

About Us

Descripción General

El componente About Us es una funcionalidad React que administra y muestra información relacionada con la sección "Acerca de Nosotros" en la aplicación. Tiene capacidades de edición condicionadas y adapta su interfaz a dispositivos móviles y de escritorio.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para gestionar estados y contextos.
    - Button y Form de react-bootstrap para componentes de interfaz.
    - WindowWidthContext, EditModeContext, y NotificationContext para manejo de contexto global.
    - Link de react-router-dom para navegación de enlaces.
    - Funciones getFooter y putFooter para operaciones de datos con el footer.
2. **Variables y Estados**
    - footer: Estado que almacena la información del footer.
    - edited: Booleano que indica si el contenido ha sido modificado.
    - Contextos: setNotifications para mensajes de notificación y editMode para alternar entre modo de visualización y edición.
3. **Efectos**
    - useEffect para cargar el contenido inicial del footer.
    - Otro useEffect para guardar cambios en el footer si se ha editado y el modo de edición está desactivado.
4. **Funciones Internas**
    - handleInput: Actualiza los valores del footer y marca el contenido como editado.
5. **Interfaz de Usuario**
    - **DesktopUI**: Versión de la interfaz para pantallas grandes, utilizando una cuadrícula de 4 columnas para mostrar títulos y líneas de contenido. Las secciones se alternan entre etiquetas de texto o entradas editables según el estado editMode.
    - **MobileUI**: Versión de la interfaz para dispositivos móviles con un diseño más compacto y simplificado.

Best Sellers

Descripción General

El componente **BestSellers** se encarga de mostrar una lista de productos más vendidos. Esta lista se carga dinámicamente en función de la categoría y subcategoría seleccionadas, obtenidas de los parámetros de la URL. El componente también maneja el estado de carga y notificaciones globales.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados y efectos.
    - useParams de react-router para obtener los parámetros de la URL.
    - Card, Container de react-bootstrap para el diseño y estructura de la interfaz.
    - LoadingState para mostrar un indicador de carga.
    - getBestSellers para realizar la llamada a la API y obtener los productos.
    - NotificationContext para gestionar las notificaciones.
2. **Variables y Estados**
    - bsProducts: Estado que almacena la lista de productos más vendidos.
    - params: Obtenidos de la URL usando useParams para filtrar los productos según la categoría y subcategoría.
    - loadingProducts: Estado booleano para manejar la visualización del indicador de carga.
    - setNotifications: Función del contexto NotificationContext para mostrar notificaciones al usuario.
3. **Efectos**
    - useEffect: Ejecuta la función getBestSellers cada vez que cambian los parámetros de la URL (params). También se encarga de actualizar el estado de loadingProducts y bsProducts.
4. **Renderización**
    - Se muestra un contenedor con un diseño horizontal que puede desplazarse si hay más productos de los que caben en la pantalla.
    - Muestra un título "Best Sellers" seguido de los productos en tarjetas (Card). Cada tarjeta muestra el nombre del producto y una imagen.
    - Si los productos están cargando, se muestra el componente LoadingState.

Cart

Descripción General

El componente **Cart** se encarga de gestionar y mostrar el contenido del carrito de compras del usuario. Implementa una funcionalidad de menú desplegable que muestra los productos en el carrito, permite eliminar productos, y proporciona una opción para proceder al pago. El componente también realiza solicitudes HTTP para obtener y eliminar productos del carrito.

Estructura del Componente

1. **Importaciones**
    - useState, useContext de React para manejar el estado y contextos.
    - Button, Dropdown, Image de react-bootstrap para el diseño de la interfaz.
    - Iconos como FaShoppingCart, IoCloseSharp, y LiaWalletSolid de react-icons para mejorar la experiencia del usuario.
    - axios para realizar solicitudes HTTP.
    - getCartItems para obtener los artículos del carrito.
    - UserProfileContext y NotificationContext para manejar información del usuario y notificaciones.
2. **Variables y Estados**
    - cartItems: Estado que almacena los artículos del carrito.
    - userProfile: Información del usuario obtenida del contexto UserProfileContext.
    - setNotifications: Función del contexto NotificationContext para manejar notificaciones.
    - loadingProducts: Estado booleano que indica si los productos se están cargando.
    - localIp: Dirección IP local, obtenida de las variables de entorno.
3. **Funciones Internas**
    - handleDelete: Función asíncrona para eliminar un producto del carrito. Realiza una solicitud DELETE con axios y recarga la página si la operación es exitosa.
    - CustomToggle: Componente personalizado que funciona como un botón para mostrar el menú desplegable y actualizar los artículos del carrito.
4. **Renderización**

- El componente usa un Dropdown de react-bootstrap con un botón personalizado (CustomToggle) que carga los artículos del carrito al hacer clic.
- Muestra una lista de productos con imágenes, nombres, descripciones, cantidades, y precios.
- Incluye un botón de eliminación para cada producto, que ejecuta handleDelete.
- Un botón final permite al usuario proceder al pago, redirigiéndolo a la lista de pedidos.

Featured Products

Descripción General

El componente **FeaturedProducts** es un carrusel que muestra productos destacados, cargados dinámicamente en función de los parámetros de la categoría y subcategoría obtenidos de la URL. Incluye un estado de carga que se muestra mientras se obtienen los datos.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados y efectos.
    - Carousel de react-bootstrap para mostrar los productos en un carrusel deslizante.
    - useParams de react-router para acceder a los parámetros de la URL.
    - LoadingState para mostrar un estado de carga mientras se obtienen los productos.
    - getFeaturedProducts para realizar la llamada a la API y obtener los productos destacados.
    - NotificationContext para manejar notificaciones globales.
2. **Variables y Estados**
    - featuredProducts: Estado que almacena los productos destacados.
    - params: Obtenido de useParams para filtrar productos por categoría y subcategoría.
    - loadingProducts: Estado booleano que indica si los productos están en proceso de carga.
    - setNotifications: Función del contexto NotificationContext para manejar las notificaciones.
3. **Efectos**
    - useEffect: Ejecuta la función getFeaturedProducts cuando cambian los parámetros de la URL (params). También actualiza los estados featuredProducts y loadingProducts.
4. **Renderización**
    - Se utiliza el componente Carousel de react-bootstrap para mostrar los productos destacados con deslizamiento automático.

- Cada ítem del carrusel muestra una imagen del producto y una descripción, si los datos están disponibles.
- Muestra LoadingState si los productos están en proceso de carga.

Loading State

Descripción General

El componente **LoadingState** es un indicador de carga que se utiliza para informar al usuario que se están obteniendo datos o procesando una acción. Utiliza el componente Spinner de react-bootstrap para mostrar un icono de carga animado y un texto que indica que la operación está en curso.

Estructura del Componente

1. **Importaciones**
    - Spinner de react-bootstrap para mostrar una animación de carga.
    - React para la creación del componente funcional.
2. **Renderización**
    - El componente devuelve un div que utiliza clases de react-bootstrap para centrar el contenido horizontal y verticalmente.
    - Muestra el Spinner junto con el texto "loading ..." para indicar que se está realizando una operación

Low Stock Products

Descripción General

El componente **LowStockProducts** se encarga de mostrar productos con bajo stock, permitiendo al usuario navegar por las categorías y subcategorías. También implementa un sistema de paginación y utiliza un menú desplegable para seleccionar categorías. Se realizan solicitudes HTTP para obtener la lista de productos y categorías.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados y contextos.
    - Componentes de react-bootstrap como Col, Pagination, Row, Button, Card, y Dropdown para la interfaz de usuario.
    - useParams y useLocation de react-router para gestionar la navegación y los parámetros de la URL.
    - axios para realizar solicitudes HTTP a la API.
    - getLowStockProducts para obtener la lista de productos con bajo stock.
    - WindowWidthContext, NotificationContext, y EditModeContext para manejar información global y funcionalidades del estado.
2. **Variables y Estados**
    - lowProducts: Estado que almacena los productos con bajo stock.
    - params: Parámetros de la URL obtenidos con useParams para filtrar productos.
    - loadingProducts: Estado booleano que indica si los productos están en proceso de carga.
    - setNotifications: Función del contexto NotificationContext para mostrar notificaciones.
    - editMode: Indica si el modo de edición está activo, obtenido del EditModeContext.
    - selectedCategories y category: Estados para gestionar las categorías seleccionadas y la lista de categorías.
    - localIp: Dirección IP local del entorno, obtenida de las variables de entorno.
3. **Efectos**
    - **Primer useEffect**: Se ejecuta al montar el componente para obtener la lista de categorías mediante una solicitud GET a la API.

- **Segundo useEffect**: Se ejecuta cuando cambian los params para obtener la lista de productos con bajo stock, actualizando lowProducts y loadingProducts.

4. **Renderización**
    - El componente se estructura en un div contenedor que cambia de diseño según la ruta actual.
    - Muestra un título "Low Stock Items" y un sistema de paginación.
    - Renderiza una fila (Row) con tarjetas (Card) para cada producto, mostrando la imagen, el nombre, el precio, y un menú desplegable de categorías.
    - Incluye un indicador de carga (LoadingState) mientras se obtienen los productos.

Navigation Desktop

Descripción General

El componente **NavigationDesktop** implementa la barra de navegación de la aplicación, diseñada para funcionar en pantallas de escritorio. Proporciona una interfaz completa que incluye opciones de búsqueda, un menú de usuario, categorías destacadas, y funcionalidades para el carrito de compras y el perfil del usuario. También maneja el modo de edición para empleados y administradores.

Estructura del Componente

1. **Importaciones**
    - useContext, useState, useEffect, y useRef de React para manejar estados, contextos, y referencias.
    - Componentes de react-bootstrap como Button, Collapse, Dropdown, Form, Navbar para construir la interfaz de usuario.
    - Íconos de react-icons para mejorar la experiencia del usuario con iconografía personalizada.
    - Módulos personalizados como Search, Cart, Qr, y AllCategories para funcionalidades específicas.
    - axios para realizar solicitudes HTTP.
    - useParams y useLocation de react-router para gestionar la navegación y obtener información de la URL.
    - UserProfileContext, EditModeContext, y NotificationContext para manejar información global del usuario, el modo de edición, y las notificaciones.
2. **Variables y Estados**
    - localIp: Dirección IP local del entorno, obtenida de las variables de entorno.
    - userProfile, setUserProfile, guestProfile: Variables del contexto UserProfileContext para manejar la información del usuario.
    - editMode, setEditMode: Variables del contexto EditModeContext para manejar el modo de edición.
    - setNotifications: Función del contexto NotificationContext para mostrar notificaciones.
    - showCategories, setShowCategories: Estado para mostrar/ocultar las categorías.
    - categories: Lista de categorías obtenida de la API.

- showOffCanvas, setShowOffCanvas: Estado para manejar la visibilidad del panel de categorías.
- navigationData: Datos de navegación obtenidos de la API.
- categoriesClicks: Estado del contexto ClicksNumberContext para manejar los clics en las categorías.

3. **Efectos**
   - **Al Montar el Componente**: Guarda el perfil del usuario en localStorage y carga las categorías y datos de navegación.
   - **useEffect**: Actualiza la altura de la barra de navegación si es necesario.
4. **Funciones Internas**
   - getSubcategories: Genera una lista de subcategorías para una categoría específica.
   - handleLogOut: Función para cerrar sesión, desactiva el modo de edición, y muestra una notificación de éxito.
5. **Renderización**
   - **Barra de Navegación**: Incluye un logotipo, opciones de búsqueda, y botones de navegación.
   - **Categorías Destacadas**: Se muestran en un menú desplegable, y el usuario puede seleccionar categorías y subcategorías.
   - **Menú de Usuario**: Incluye opciones como "Users List", "Orders", "Categories", y "Profile", dependiendo del rol del usuario.
   - **Opciones de Invitado**: Se muestran botones para registrarse o iniciar sesión si el usuario es un invitado.
   - **Modo de Edición**: Solo visible para empleados o administradores, permite alternar el estado de edición.

Detalles Técnicos

1. **Solicitudes HTTP**: Se utiliza axios para obtener categorías y datos de navegación desde el backend.
2. **Gestión de Categorías**: Las categorías y subcategorías se manejan dinámicamente, y se pueden expandir o colapsar con un botón.
3. **Opciones Condicionales**: Las opciones del menú varían en función del rol del usuario (guest, employee, admin).

Navigation Mobile:

Descripción General

El componente **NavigationMobile** gestiona la barra de navegación para dispositivos móviles. Incluye funcionalidades como un botón de menú desplegable, un modo de edición para empleados y administradores, un campo de búsqueda, y un botón de carrito de compras. También proporciona la capacidad de subir archivos mediante un componente de entrada de archivos oculto.

Estructura del Componente

1. **Importaciones**
    - useContext, useState, useEffect, y useRef de React para manejar el estado, contextos, y referencias.
    - Componentes de react-bootstrap como Button, Form, FormCheck, y Navbar para construir la barra de navegación.
    - GiHamburgerMenu de react-icons para el icono del menú.
    - Componentes personalizados como Search, Cart, y AllCategories para funcionalidades adicionales.
    - Link de react-router-dom para la navegación.
    - UserProfileContext, EditModeContext, y NotificationContext para gestionar el perfil del usuario, el modo de edición, y las notificaciones.
    - getCategories para obtener las categorías de la API.
2. **Variables y Estados**
    - userProfile, setUserProfile, guestProfile: Variables del contexto UserProfileContext para manejar la información del usuario.
    - editMode, setEditMode: Variables del contexto EditModeContext para alternar el modo de edición.
    - setNotifications: Función del contexto NotificationContext para manejar notificaciones.
    - showOffCanvas, setShowOffCanvas: Estado para controlar la visibilidad del menú de categorías desplegable.
    - categories: Estado que almacena las categorías obtenidas de la API.
    - fileName, setFileName: Estado para gestionar el nombre del archivo seleccionado.
    - fileInputRef: Referencia al elemento de entrada de archivo oculto.

3. **Funciones Internas**
    - handleToggle: Alterna la visibilidad del menú desplegable.
    - handleButtonClick: Simula un clic en el elemento de entrada de archivo oculto.
    - handleFileChange: Actualiza el estado fileName con el nombre del archivo seleccionado.
    - **Lógica de Roles**: Determina si el usuario es un invitado, empleado o administrador y muestra elementos condicionales en consecuencia.
4. **Efectos**
    - **useEffect para Categorías**: Llama a getCategories para obtener y establecer las categorías.
    - **useEffect para Altura de la Barra de Navegación**: Ajusta la altura de la barra de navegación si es necesario.
5. **Renderización**
    - **Barra de Navegación**: Incluye un logotipo, un botón de menú, un campo de búsqueda, y un icono de carrito de compras.
    - **Modo de Edición**: Un interruptor de FormCheck permite alternar el modo de edición, visible solo para empleados y administradores.
    - **Menú Desplegable**: Botón de menú de hamburguesa que abre un menú offcanvas para mostrar todas las categorías.

New Product

Descripción General

El componente **NewProduct** permite a los usuarios agregar nuevos productos al sistema. Incluye un formulario con campos para ingresar información del producto, como título, descripción, precio, stock, y categorías. También permite seleccionar si un producto es destacado y si está habilitado. El componente realiza solicitudes HTTP para obtener las categorías y registrar el nuevo producto.

Estructura del Componente

1. **Importaciones**
    - useContext, useState, useEffect de React para gestionar estados y contextos.
    - Componentes de react-bootstrap como Button, Card, Form, FormControl, Dropdown para construir el formulario y la interfaz.
    - FaSave y FaListAlt de react-icons para iconos de guardar y lista de categorías.
    - WindowWidthContext para ajustar el diseño de la interfaz en función del ancho de la ventana.
    - axios para realizar solicitudes HTTP a la API.
2. **Variables y Estados**
    - name, desc, price, stock: Estados para almacenar la información ingresada del producto.
    - category: Lista de categorías obtenida de la API.
    - selectedCategories: Categorías seleccionadas por el usuario.
    - featuredItem, enableItem: Estados booleanos que determinan si el producto es destacado y si está habilitado.
    - localIp: Dirección IP local del entorno, obtenida de las variables de entorno.
3. **Funciones Internas**
    - handleCategories: Realiza una solicitud GET para obtener las categorías y almacenarlas en el estado category.
    - handleSubmit: Maneja el envío del formulario, realiza una solicitud POST para registrar el producto, y ejecuta handleData al finalizar.
    - handleCheckboxChange: Gestiona la selección de categorías, actualizando selectedCategories cuando un checkbox cambia de estado.
4. **Efectos**

- **useEffect para Categorías**: Llama a handleCategories al montar el componente para cargar las categorías desde la API.

5. **Renderización**
    - **Formulario**: Incluye campos para imagen, título, descripción, precio, stock, y checkboxes para las opciones de producto destacado y habilitado.
    - **Botón de Guardar**: Botón de envío que registra el producto al hacer clic.
    - **Dropdown de Categorías**: Muestra una lista desplegable con checkboxes para seleccionar múltiples categorías.

Detalles Técnicos

1. **Solicitudes HTTP**: Se realizan solicitudes GET para obtener categorías y POST para registrar el producto.
2. **Selección de Categorías**: Los usuarios pueden seleccionar múltiples categorías usando checkboxes en un menú desplegable.
3. **Gestión de Productos**: El formulario incluye validación básica, como el manejo de entradas vacías y la selección de categorías.

New Products

Descripción General

El componente **NewProducts** muestra una lista de productos nuevos en un diseño horizontal y desplazable. Utiliza una llamada a la API para obtener la lista de productos nuevos en función de la categoría y subcategoría proporcionadas a través de los parámetros de la URL. También maneja el estado de carga y muestra un componente de carga hasta que los productos estén disponibles.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados y efectos.
    - Componentes de react-bootstrap como Card, Container, Image para construir la interfaz.
    - useParams de react-router para obtener los parámetros de la URL.
    - LoadingState para mostrar un indicador de carga mientras se obtienen los productos.
    - getNewProducts para realizar la solicitud a la API y obtener los productos nuevos.
    - NotificationContext para manejar notificaciones globales.
    - Link de react-router-dom para la navegación.
2. **Variables y Estados**
    - newProducts: Estado que almacena la lista de productos nuevos.
    - params: Parámetros de la URL obtenidos con useParams para filtrar los productos por categoría y subcategoría.
    - loadingProducts: Estado booleano que indica si los productos se están cargando.
    - setNotifications: Función del contexto NotificationContext para mostrar notificaciones al usuario.
3. **Efectos**
    - **useEffect**: Llama a getNewProducts al montar el componente o cuando cambian los params para obtener los productos nuevos y actualizar los estados newProducts y loadingProducts.
4. **Renderización**

- Se usa un contenedor Container con estilo desplazable horizontalmente para mostrar los productos.
- Cada producto se renderiza dentro de un Card redondeado, que muestra el nombre del producto y su imagen.
- Si los productos se están cargando, se muestra el componente LoadingState.

Detalles Técnicos

1. **Llamada a la API**: Usa getNewProducts para obtener los productos nuevos basados en los parámetros de categoría y subcategoría.
2. **Renderización Condicional**: Muestra un componente de carga mientras los productos se están obteniendo y renderiza los productos solo si hay datos disponibles.
3. **Interfaz de Usuario**:
   - Los productos se muestran como tarjetas (Card) con un diseño redondeado (rounded-circle) y centrado.
   - Las imágenes se renderizan como círculos (roundedCircle) con tamaño fijo.

Notifications

Descripción General

El componente **Notifications** se encarga de mostrar notificaciones al usuario en forma de toasts. Las notificaciones pueden ser de diferentes tipos, como loading, error, success, o danger, y se posicionan dinámicamente en función del ancho de la ventana. Las notificaciones se ocultan automáticamente después de un tiempo establecido o se pueden cerrar manualmente.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados y efectos.
    - Spinner, Toast, ToastContainer de react-bootstrap para construir la interfaz de las notificaciones.
    - FaCheckCircle de react-icons para mostrar un icono de éxito.
    - WindowWidthContext para ajustar la posición de las notificaciones en función del ancho de la ventana.
2. **Variables y Estados**
    - position: Estado que almacena la posición de las notificaciones, que puede ser 'top-start' o 'bottom-center' según el ancho de la ventana.
    - windowWidth: Ancho de la ventana obtenido del WindowWidthContext.
3. **Efectos**
    - **useEffect**: Cambia la posición de las notificaciones cuando windowWidth cambia. Si el ancho de la ventana es menor a 1000px, las notificaciones se muestran en la parte inferior; de lo contrario, se muestran en la parte superior.
4. **Renderización**
    - Se usa un contenedor ToastContainer para organizar las notificaciones.
    - Cada notificación se renderiza como un Toast, con un encabezado (Toast.Header) y un cuerpo (Toast.Body).
    - Se muestra un Spinner si el tipo de notificación es loading o un icono de FaCheckCircle si es success.
    - Las notificaciones tienen colores de fondo específicos según su tipo:
        - **loading**: bg-primary
        - **error**: bg-secondary

- **success**: bg-success
- **default (opcional)**: bg-danger

Detalles Técnicos

1. **Posición Dinámica**: La posición de las notificaciones cambia automáticamente dependiendo del ancho de la ventana.
2. **Autocierre**: Las notificaciones se ocultan automáticamente después de 4 segundos (autohide: true, delay: 4000).
3. **Tipos de Notificación**:
    - **loading**: Muestra un Spinner y un mensaje de "Loading, Please Wait ..."
    - **success**: Muestra un icono de FaCheckCircle y un mensaje personalizado.
    - **error** y otros: Muestra un mensaje personalizado con un color de fondo específico.

Order List

Descripción General

El componente **OrderList** muestra una lista de productos en el carrito de compras del usuario actual, permite modificar la cantidad de productos, y ofrece la opción de eliminar productos del carrito. También muestra las órdenes pagadas del usuario y proporciona una funcionalidad de pago que incluye la verificación de la tarjeta y el envío de un correo de confirmación tras un pago exitoso.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados y efectos.
    - Spinner, Toast, Modal, Alert, y otros componentes de react-bootstrap para la interfaz de usuario.
    - useNavigate, useParams de react-router para la navegación y obtención de parámetros de la URL.
    - getCartItems para obtener los productos del carrito.
    - axios para realizar solicitudes HTTP.
    - IoCloseSharp y LiaWalletSolid de react-icons para iconos.
    - NotificationContext y UserProfileContext para manejar la información del usuario y las notificaciones globales.
2. **Variables y Estados**
    - cartItems: Estado que almacena los productos del carrito.
    - total, shipping, minimum, orderId: Estados para almacenar los totales, el costo de envío, el pedido mínimo y el ID del pedido.
    - paid: Estado que almacena las órdenes pagadas.
    - cardInput: Número de la tarjeta ingresada por el usuario.
    - error: Estado para manejar errores en el pago.
    - showModal: Estado para controlar la visibilidad del modal de pago.
    - loadingProducts: Indica si los productos se están cargando.
3. **Funciones Internas**
    - handlePayment: Maneja el proceso de pago, incluyendo la verificación del número de tarjeta y el envío de un correo.
    - handlePaid: Obtiene las órdenes pagadas del usuario.

- handleDelete: Elimina un producto del carrito y actualiza la interfaz.
- handleData: Obtiene el total, el mínimo de envío, y el costo de envío para actualizar los datos del carrito.
- handleUpdateAmount: Actualiza la cantidad de un producto en el carrito y verifica que no sea menor a 1 ni mayor al stock disponible.
- handleEmailSend: Envía un correo electrónico con los detalles del pedido.

4. **Efectos**
    - **useEffect**: Obtiene los productos del carrito y las órdenes pagadas al montar el componente o al cambiar el usuario.
    - **useEffect para Actualizar Datos**: Llama a handleData y handlePaid al cambiar el usuario.
5. **Renderización**
    - **Carrito de Compras**: Muestra una tabla con los productos, permitiendo editar la cantidad y eliminar productos.
    - **Información del Usuario**: Muestra los detalles de la tarjeta y la dirección del usuario.
    - **Opciones de Pago**: Botón que abre un modal para ingresar el número de tarjeta y confirmar el pago.
    - **Órdenes Pagadas**: Muestra una tabla con las órdenes pagadas, incluyendo el estado y comentarios.

Detalles Técnicos

1. **Procesamiento de Pago**: El pago se procesa al verificar que el número de tarjeta ingresado coincide con el del usuario. Luego se envía un correo de confirmación.
2. **Modificación de Cantidad**: Se previene el ingreso de valores no numéricos o negativos, y se ajusta automáticamente al máximo disponible si se excede el stock.
3. **Eliminación de Productos**: Los productos se pueden eliminar del carrito y la interfaz se actualiza automáticamente.
4. **Órdenes Pagadas**: Se muestran las órdenes pagadas en una tabla, con diferentes estados representados como "Confirmed", "Preparing", "In route", "Completed", o "Canceled".

Product

Descripción General

El componente **Product** permite visualizar y editar los detalles de un producto específico. Se utiliza para gestionar tanto la visualización estándar de un producto como su edición, lo que incluye actualizar la información del producto y cambiar su imagen mediante una herramienta de redimensionamiento. El modo de edición solo está disponible para usuarios con los permisos adecuados.

Estructura del Componente

1. **Importaciones**
    - useContext, useState, useEffect de React para gestionar estados, contextos y efectos.
    - Componentes de react-bootstrap como Button, Card, Dropdown, Form, FormControl para la interfaz de usuario.
    - FaListAlt, FaSave, MdDelete de react-icons para iconos.
    - useLocation, useNavigate, useParams de react-router para la navegación y gestión de parámetros de la URL.
    - WindowWidthContext, EditModeContext, UserProfileContext para manejar el estado global y el modo de edición.
    - Resizer de react-image-file-resizer para redimensionar imágenes antes de cargarlas.
    - axios para realizar solicitudes HTTP.
2. **Variables y Estados**
    - category: Lista de categorías obtenida de la API.
    - formData: Objeto que almacena los datos del producto, incluyendo nombre, descripción, precio, stock, categorías, etc.
    - selectedCategories: Categorías seleccionadas para el producto.
    - featuredItem, enableItem: Estados booleanos para gestionar si el producto es destacado o está habilitado.
    - localIp: Dirección IP local del entorno, obtenida de las variables de entorno.
3. **Funciones Internas**
    - handleImageChange: Redimensiona una imagen seleccionada y actualiza el estado formData con la imagen redimensionada en formato base64.

- handleCategories: Obtiene la lista de categorías de la API y actualiza el estado category.
- handleDownload: Descarga una imagen con un nombre específico.
- handleDelete: Elimina un producto de la base de datos y actualiza la lista de productos.
- handleSave: Envía los datos actualizados del producto al backend.
- handleInput: Actualiza formData en función de los cambios en los campos de entrada.
- handleCheckboxChange, handleCheckboxChangeFeatured, handleCheckboxChangeEnable: Gestionan los cambios en las categorías y opciones booleanas (producto destacado y habilitado).

4. **Efectos**
   - **useEffect para Categorías**: Llama a handleCategories al montar el componente para obtener las categorías.
   - **useEffect para Configurar Datos del Producto**: Configura formData y los estados booleanos (featuredItem y enableItem) al recibir los datos del producto.
5. **Renderización**
   - **Modo de Visualización**: Muestra el producto en una tarjeta con su imagen, nombre, precio y categorías.
   - **Modo de Edición**: Muestra un formulario que permite editar los detalles del producto, incluyendo la carga de imágenes y selección de categorías.
   - **Botones y Dropdowns**: Incluye un botón para guardar cambios y un dropdown para seleccionar categorías.

Detalles Técnicos

1. **Carga y Redimensionamiento de Imágenes**: Usa Resizer para redimensionar imágenes a un máximo de 300x300 píxeles antes de cargarlas como base64.
2. **Modo de Edición**: Solo disponible si editMode es true, muestra un formulario editable para actualizar el producto.
3. **Solicitudes HTTP**: Realiza llamadas GET, PUT, y DELETE para obtener categorías, actualizar productos y eliminar productos respectivamente.
4. **Validación de Datos**: Incluye lógica para manejar entradas válidas y actualiza formData de manera eficiente.

Qr

Descripción General

El componente **Qr** genera y muestra un código QR basado en la URL actual de la página. La visibilidad del código QR se controla mediante una propiedad show que se pasa al componente. Cuando show es true, el código QR se renderiza y se centra horizontalmente en la página.

Estructura del Componente

1. **Importaciones**
    - QRCode de react-qr-code para generar el código QR.
    - React para la creación del componente funcional.
2. **Variables y Estados**
    - currentUrl: Se obtiene utilizando window.location.href para capturar la URL actual del navegador.
3. **Renderización**
    - El componente solo renderiza el código QR si la propiedad show es true.
    - El código QR se centra utilizando las clases de Bootstrap my-2 y d-flex justify-content-center.

Search

Descripción General

El componente **Search** proporciona una barra de búsqueda para buscar productos en la aplicación. Permite a los usuarios buscar productos por nombre, descripción, categorías, y para administradores o empleados, filtrar productos deshabilitados. Incluye un dropdown de opciones de filtro y utiliza la biblioteca react-bootstrap para una interfaz elegante y responsiva.

Estructura del Componente

1. **Importaciones**
    - useContext, useState, useEffect de React para manejar estados, contextos, y efectos.
    - Componentes de react-bootstrap como Button, Dropdown, Form para construir la interfaz.
    - FaFilter, FaSearch de react-icons para los iconos de búsqueda y filtro.
    - useLocation, useNavigate de react-router para la navegación.
    - UserProfileContext, SearchProductsContext para manejar la información del usuario y el contexto de búsqueda.
    - postSearchProducts para realizar la búsqueda y actualizar el contexto de resultados.
    - useLocalStorage para almacenar y recuperar datos localmente en el navegador.
2. **Variables y Estados**
    - searchTerm: Término de búsqueda ingresado por el usuario, almacenado en localStorage.
    - checkedFields: Array de booleanos que indica qué filtros están activos, también almacenado en localStorage.
    - previousPath: Ruta anterior a la búsqueda, para facilitar la navegación.
    - userProfile: Información del perfil del usuario, incluyendo el rol.
    - employee y admin: Variables booleanas que indican si el usuario tiene permisos de empleado o administrador.
3. **Funciones Internas**
    - handleSearch: Realiza la búsqueda al presionar el botón de búsqueda o la tecla "Enter". Actualiza la ruta de búsqueda y realiza una solicitud POST para obtener los productos filtrados.

- CustomToggle: Componente personalizado para el botón del dropdown de filtros.
- Gestión de filtros: Actualiza checkedFields cuando se seleccionan o deseleccionan las opciones de filtro.

4. **Renderización**
    - **Formulario de Búsqueda**: Incluye un campo de búsqueda y un botón de filtro.
    - **Dropdown de Filtros**: Permite seleccionar filtros como "Name", "Description", "Categories", y "Disabled" (solo visible para empleados o administradores).
    - **Interfaz Condicional**: Cambia el diseño y las opciones de filtro según el rol del usuario.

Detalles Técnicos

1. **Gestión de Búsqueda**: Usa navigate para cambiar la URL y postSearchProducts para realizar la búsqueda y actualizar los productos en el contexto.
2. **Filtros Personalizados**: Los usuarios pueden filtrar los productos por diferentes campos y las opciones cambian dinámicamente según el rol del usuario.
3. **Almacenamiento Local**: Utiliza useLocalStorage para persistir el término de búsqueda y los filtros seleccionados, incluso después de actualizar la página.

All Categories

Descripción General

El componente **AllCategories** muestra un menú lateral (Offcanvas) que lista todas las categorías de productos, junto con las subcategorías y opciones para editar (solo en modo de edición). El menú puede ser mostrado o escondido según el estado del componente, y es adaptable para dispositivos móviles.

Estructura del Componente

1. **Importaciones**
    - useContext, useEffect, useState de React para manejar estados, contextos, y efectos.
    - Componentes de react-bootstrap como Button, Dropdown, Form, Nav, y Offcanvas para construir la interfaz.
    - Iconos de react-icons como CgUserList, IoMdPersonAdd, IoLogIn, RiLogoutBoxFill para la representación visual.
    - axios para realizar solicitudes HTTP.
    - getCategories para obtener las categorías desde la API.
    - Contextos personalizados como UserProfileContext, NotificationContext, EditModeContext, ClicksNumberContext para manejar información de usuario, notificaciones, modo de edición y clics.
2. **Variables y Estados**
    - categories: Lista de categorías obtenida de la API.
    - filteredCat: Lista de categorías filtradas según la búsqueda del usuario.
    - recentCat: Categorías recientemente visitadas por el usuario.
    - categoriesClicks: Número de clics en las categorías.
    - clickedCategories: Categorías más visitadas.
    - showOffCanvas: Estado que controla la visibilidad del menú Offcanvas.
    - mobile: Propiedad que indica si el componente se está utilizando en un dispositivo móvil.
3. **Funciones Internas**
    - fetchCategoriesWithNavigationData: Obtiene las categorías y los datos de navegación desde la API, y marca las subcategorías.
    - handleFormCheckChange: Gestiona los cambios en los checkboxes de subcategorías, enviando los datos actualizados al backend.

- handleSave: Actualiza las categorías en la base de datos.
- handleClicks: Gestiona el aumento del número de clics en una categoría.
- recentCategories: Gestiona las categorías recientes, asegurando que no haya duplicados.

4. **Renderización**
    - **Offcanvas**: Menú lateral que muestra las categorías, opciones de usuario (Sign Up, Log In, Profile, Log Out) y las categorías más visitadas.
    - **Recientes**: Sección que muestra las últimas categorías visitadas.
    - **Más Visitadas**: Sección que muestra las categorías con más clics.
    - **Búsqueda de Categorías**: Un campo de búsqueda que filtra las categorías en tiempo real.
    - **Modo de Edición**: Solo visible si editMode es true, permite marcar categorías como destacadas y gestionar subcategorías.

Detalles Técnicos

1. **Gestión de Categorías**: Obtiene las categorías y datos de navegación desde la API y actualiza los estados correspondientes.
2. **Modo de Edición**: Permite marcar categorías como destacadas y gestionar subcategorías.
3. **Búsqueda y Filtrado**: Incluye un campo de búsqueda que filtra las categorías en tiempo real.
4. **Recientes y Más Visitadas**: Muestra las categorías visitadas recientemente y las más populares según el número de clics.

Category Page

Descripción General

El componente **CategoryPage** es una página que muestra productos destacados y todos los productos de una categoría específica. Usa el contexto EditModeContext para determinar si debe mostrar los productos destacados o dejar espacio adicional en el diseño cuando el modo de edición está activo.

Estructura del Componente

1. **Importaciones**
    - useContext de React para usar el contexto EditModeContext.
    - FeaturedProducts desde ../components/FeaturedProducts para mostrar los productos destacados.
    - Products desde ./Products para listar todos los productos.
    - EditModeContext desde ../context/EditModeContext para manejar el modo de edición.
2. **Variables y Contextos**
    - editMode: Variable de contexto que indica si el modo de edición está activo (true) o no (false).
3. **Renderización**
    - **Condicional**: Muestra FeaturedProducts si editMode es false; de lo contrario, añade un margen superior adicional para ajustar el diseño.
    - Siempre renderiza el componente Products debajo de los productos destacados o del espacio de margen.

Edit Categories

Descripción General

El componente **EditCategories** proporciona una interfaz para que los administradores o empleados puedan gestionar (añadir, editar o eliminar) categorías de productos. Utiliza react-bootstrap para el diseño, axios para las solicitudes HTTP, y useContext para gestionar el estado y las notificaciones.

Estructura del Componente

1. **Importaciones**
    - react: Biblioteca principal de React.
    - useContext, useEffect, useState: Hooks para manejar el contexto, efectos y estado.
    - axios: Para realizar solicitudes HTTP al backend.
    - react-bootstrap: Para los elementos de la interfaz de usuario.
    - **Contextos Importados**:
        - UserProfileContext: Para verificar el rol del usuario y manejar el perfil.
        - NotificationContext: Para gestionar las notificaciones de éxito o error.
    - **React Router**:
        - useNavigate: Para la navegación en la aplicación.
2. **Contextos y Variables de Estado**
    - userProfile: Información del perfil del usuario, obtenida de UserProfileContext.
    - data: Almacena las categorías recuperadas del backend.
    - editRowId: Guarda el ID de la fila que se está editando.
    - formData: Almacena los datos del formulario para editar categorías.
    - name: Almacena el nombre de la nueva categoría que se está añadiendo.
    - showModal: Controla la visibilidad del modal para agregar categorías.
3. **Funciones Principales**
    - **handleData**: Recupera las categorías del backend.
    - **handleDelete**: Elimina una categoría específica y recarga los datos.
    - **handleEdit**: Activa el modo de edición para una categoría específica.

- **handleInput**: Actualiza los datos del formulario cuando el usuario cambia los campos.
- **handleSave**: Guarda los cambios realizados en una categoría y actualiza los datos.
- **handleCancel**: Cancela el modo de edición.
- **handleSubmit**: Añade una nueva categoría enviando los datos al backend.
- **handleOpenModal y handleCloseModal**: Abren y cierran el modal para añadir una categoría.

4. **Control de Acceso**
    - Se asegura de que solo los usuarios con el rol de administrador (rol igual a 3) o empleado (rol igual a 2) puedan acceder a la página. Si no, redirige al usuario a una página de error.

Detalles Técnicos

1. **Manejo de Datos**:
    - Los datos se obtienen del backend mediante solicitudes GET y se actualizan con solicitudes PUT y DELETE.
2. **Control de Acceso**:
    - Solo los usuarios con el rol adecuado pueden acceder a la página y realizar operaciones.
3. **Gestión de Modal**:
    - Se utiliza un Modal de Bootstrap para añadir nuevas categorías.

Error

Descripción General

El componente **Error** proporciona una página personalizada que se muestra cuando un usuario intenta acceder a una ruta no válida o no autorizada en la aplicación. Incluye un mensaje de error visualmente atractivo y un botón que redirige al usuario de vuelta a la página principal.

Estructura del Componente

1. **Importaciones**
    - react: Biblioteca principal de React.
    - useEffect: Hook para manejar los efectos secundarios.
    - useNavigate: De react-router para la navegación programática.
    - **Componentes de IU**: Button, Image de react-bootstrap.
    - **Iconos**: IoHomeSharp de react-icons/io5 para agregar un icono al botón de navegación.
2. **Funcionalidad Principal**
    - **useEffect**: Añade un listener para el evento popstate, que se dispara cuando el usuario utiliza los botones de navegación del navegador (atrás/adelante).
        - Si el usuario intenta retroceder a la ruta /error, se redirige automáticamente a la página principal (/).
        - Se limpia el evento cuando el componente se desmonta.
3. **Diseño de la Página de Error**
    - **Imagen**: Muestra una imagen de un perro confundido (ruta: ${process.env.PUBLIC_URL}/confusedDog.webp) que se adapta al tamaño de la pantalla (fluid).
    - **Mensaje de Error**: Un mensaje grande y centrado que dice "Sorry, page not found".
    - **Botón de Navegación**: Un botón estilizado que lleva al usuario de vuelta a la página de inicio.

Error Page

Descripción General

El componente **ErrorPage** proporciona una página de error que se muestra cuando un usuario intenta acceder a una ruta inexistente o no permitida. Incluye una imagen temática, un mensaje de error, la barra de navegación y el componente AboutUS para ofrecer contexto adicional.

Estructura del Componente

1. **Importaciones**
    - **React**: Biblioteca principal de React.
    - **useEffect**: Hook para manejar los efectos secundarios.
    - **useNavigate**: De react-router para la navegación programática.
    - **Components**: Navigation, AboutUS, Image de react-bootstrap para estilizar el contenido.
2. **Funcionalidad Principal**
    - **useEffect**: Configura un listener para el evento popstate del navegador.
        - Si el usuario intenta retroceder hasta la ruta /error, se redirige automáticamente a la página principal (/).
        - Se asegura de limpiar el listener cuando el componente se desmonta para evitar problemas de memoria.
3. **Diseño de la Página de Error**
    - **Navigation**: Incluye el componente de navegación en la parte superior.
    - **Imagen**: Muestra una imagen de un perro confundido (confusedDog.webp) que se adapta al tamaño de la pantalla con la clase fluid.
    - **Mensaje de Error**: Muestra un mensaje en texto grande y centrado que dice "Sorry, page not found."
    - **AboutUS**: Se incluye este componente para brindar más contexto o información útil a los visitantes.

Forgotten

Descripción General

El componente **Forgotten** se utiliza para manejar el proceso de cambio de contraseña de los usuarios, basado en un token de restablecimiento proporcionado en la URL. Si el token es válido y no ha sido utilizado previamente, se permite al usuario establecer una nueva contraseña. De lo contrario, se muestra una página de error.

Estructura del Componente

1. **Importaciones**
    - **React y Hooks**: useContext, useState, useEffect se utilizan para manejar el estado y los efectos secundarios.
    - **React-Bootstrap**: Componentes como Button, Card, Form para el diseño del formulario.
    - **Axios**: Para realizar solicitudes HTTP al backend.
    - **Contextos**: UserProfileContext, NotificationContext para manejar el perfil del usuario y las notificaciones.
    - **Error**: Componente que se muestra si el token es inválido o ya se ha utilizado.
2. **Manejo de Estados y Contextos**
    - userProfile: Contexto que proporciona información sobre el perfil del usuario.
    - loadingProducts: Estado que indica si los datos están cargando.
    - password y confirmpass: Estados para almacenar y comparar las contraseñas ingresadas por el usuario.
    - tokens: Lista de tokens de restablecimiento obtenidos del backend.
    - notifications: Contexto para mostrar notificaciones al usuario.
3. **Lógica Principal**
    - **Validación de Contraseñas**: Se verifica si la nueva contraseña y la confirmación coinciden.
    - **Token y Usuario**: Se busca el token en la lista obtenida del backend para verificar su validez y si ya ha sido utilizado.
    - **Solicitud de Cambio de Contraseña**: Se realiza una solicitud PUT al backend con el nuevo password si todas las validaciones pasan.
4. **Efectos Secundarios**
    - useEffect: Se utiliza para obtener la lista de tokens de restablecimiento del backend al cargar el componente.

- Se actualiza el estado tokens con la respuesta del backend.

5. **Renderizado Condicional**
    - Si el token es válido y no ha sido utilizado (used === 0), se muestra el formulario para cambiar la contraseña.
    - Si el token es inválido o ya ha sido utilizado, se muestra el componente Error.

Login

Descripción General

El componente **Login** maneja tanto el proceso de autenticación de usuarios como la solicitud de restablecimiento de contraseña. Incluye dos formularios modales, uno para iniciar sesión y otro para solicitar un correo de restablecimiento de contraseña.

Secciones Principales del Componente

1. **Importaciones y Dependencias**
    - useContext, useState, useEffect: Hooks de React para manejar el estado y los efectos secundarios.
    - Button, Form, Modal: Componentes de react-bootstrap para construir el formulario y los botones.
    - NotificationContext: Contexto personalizado para manejar notificaciones.
    - UserProfileContext: Contexto personalizado que contiene información del usuario y funciones para actualizarla.
    - axios: Biblioteca para realizar solicitudes HTTP.
2. **Estados y Variables**
    - validated: Estado para validar el formulario.
    - password y forgotemail: Estados para almacenar la contraseña ingresada y el correo electrónico para el restablecimiento.
    - showForgotten: Estado para mostrar u ocultar el modal de "Olvidé mi contraseña".
    - localIp: Dirección IP local para las solicitudes a la API.
3. **Funciones Principales**
    - **handleClose** y **handleCloseForgotten**: Cierran los modales correspondientes.
    - **handleSubmit**: Envía la solicitud de inicio de sesión.
        - Valida el formulario y verifica si las credenciales son correctas.
        - Actualiza el contexto UserProfileContext con la información del usuario si el inicio de sesión es exitoso.
        - Muestra notificaciones de éxito o error.
    - **handleSubmitForgot**: Envía la solicitud para restablecer la contraseña.

        - Realiza una solicitud POST a la API con el correo electrónico proporcionado.
        - Muestra notificaciones basadas en la respuesta de la API.

4. **Estructura del Modal de Inicio de Sesión**
    - **Formulario de Inicio de Sesión**
        - Campos para ingresar el correo electrónico y la contraseña.
        - Botón "Submit" para enviar el formulario.
        - Enlace para abrir el modal de restablecimiento de contraseña.
    - **Formulario de Restablecimiento de Contraseña**
        - Campo para ingresar el correo electrónico asociado a la cuenta.
        - Botón "Submit" para enviar la solicitud de restablecimiento.

Lógica de Autenticación

1. **Envío de la Solicitud de Inicio de Sesión**
    - Si el formulario es válido, se envía una solicitud POST a la API con el correo electrónico y la contraseña.
    - Si la respuesta es exitosa, se actualiza el perfil del usuario y se muestra una notificación de éxito.
    - Si el usuario está inactivo o no confirmado, se muestra una notificación de error.
2. **Restablecimiento de Contraseña**
    - Se envía una solicitud POST con el correo electrónico para recibir un correo de restablecimiento de contraseña.
    - Se manejan los casos en los que el correo electrónico no está asociado a ninguna cuenta.

Login Comfirm

Descripción General

LoginComfirm es un componente de React que permite a los usuarios confirmar su intención de iniciar sesión. Se utiliza un token de confirmación, recibido a través de los parámetros de la URL, para verificar la solicitud. Si el token es válido, el usuario puede introducir sus credenciales (correo electrónico y contraseña) para completar el proceso de inicio de sesión.

Estructura del Componente

1. **Importaciones y Dependencias**
    - useContext, useState, useEffect: Hooks de React para gestionar el estado y los efectos secundarios.
    - Button, Card, Form: Componentes de react-bootstrap para crear la interfaz del formulario.
    - axios: Biblioteca para hacer solicitudes HTTP.
    - NotificationContext y UserProfileContext: Contextos personalizados para manejar notificaciones y datos de usuario.
    - useParams: Hook de react-router para acceder a los parámetros de la URL.
    - Error: Componente que se muestra si el token no es válido.
2. **Estado y Variables**
    - email y password: Estados para almacenar las credenciales introducidas por el usuario.
    - tokens: Estado para almacenar la lista de tokens de confirmación obtenidos del servidor.
    - loadingProducts: Estado que podría usarse para manejar la carga de datos (aunque no se utiliza completamente en este código).
    - localIp: Variable que obtiene la dirección IP local del entorno.
3. **Funciones Principales**
    - **handleSubmit**: Se ejecuta al enviar el formulario.
        - Realiza una solicitud POST a la API con el correo electrónico, la contraseña y el token.
        - Muestra notificaciones de éxito o error en función de la respuesta del servidor.
    - **useEffect**: Se ejecuta al montar el componente.

        - Realiza una solicitud GET para obtener todos los tokens de confirmación del servidor.
        - Almacena los tokens en el estado tokens.

4. **Validación del Token**
    - Antes de mostrar el formulario, el componente verifica si el token recibido en los parámetros de la URL está presente en la lista de tokens obtenidos del servidor.
    - Si el token es válido, se muestra el formulario de confirmación. Si no, se muestra el componente Error.

Secciones del Formulario

- **Correo Electrónico**: Campo de entrada para que el usuario introduzca su correo electrónico.
- **Contraseña**: Campo de entrada para que el usuario introduzca su contraseña.
- **Botón de Envío**: Envía las credenciales y el token al servidor para confirmar el inicio de sesión.

New User Admin

Está diseñado para permitir la creación de nuevos usuarios administradores o empleados mediante un formulario de registro.

1. **Importaciones**
    - useState y useContext: Hooks de React para manejar el estado y contexto.
    - Button, Form, Modal: Componentes de react-bootstrap utilizados para la interfaz.
    - axios: Biblioteca para hacer solicitudes HTTP.
    - NotificationContext: Contexto personalizado para gestionar notificaciones.
2. **Estados del Componente**
    - validated: Estado booleano para validar el formulario.
    - firstname, lastname, email, password, birthdate, rol: Estados que almacenan los datos introducidos por el usuario.
    - confirm: Almacena la contraseña confirmada para verificar si coincide con la contraseña original.
    - notifications: Contexto que maneja las notificaciones en la aplicación.
3. **Funciones Principales**
    - **handleSubmit**:
        - Se encarga de enviar los datos del formulario.
        - Verifica si las contraseñas coinciden y envía los datos a la API mediante una solicitud PUT.
        - Muestra notificaciones de éxito o error según la respuesta.
    - **handleClose**:
        - Cierra el modal de registro.
4. **Validación del Formulario**
    - La función handleSubmit incluye una validación para verificar si las contraseñas coinciden antes de enviar los datos.
    - Se asegura de que los roles introducidos sean únicamente 2 o 3.
5. **Estructura del Formulario**
    - **Foto de Perfil**: Componente Form.Control para subir una imagen (actualmente no manejada).

- **Nombre y Apellido**: Campos de entrada de texto.
- **Correo Electrónico**: Campo de entrada con validación de correo.
- **Contraseña y Confirmación de Contraseña**: Campos de entrada de contraseña con validación de coincidencia.
- **Rol**: Campo de entrada para el rol del usuario (solo acepta valores 2 o 3).
- **Fecha de Nacimiento**: Campo de entrada de tipo fecha.

Detalles Técnicos

- **Solicitudes HTTP**: Se utiliza axios para enviar una solicitud PUT a la API con los datos del usuario.
- **Validación de Contraseñas**: Se verifica que las contraseñas coincidan antes de enviar el formulario. Si no coinciden, se muestra una notificación de error.
- **Notificaciones**: Se utilizan notificaciones para informar al usuario sobre el estado del registro (éxito o error).

Orders

Está diseñado para mostrar una lista de pedidos en una tabla, permitiendo a los usuarios con roles específicos (2 o 3) administrar los pedidos mediante la actualización de comentarios y la transición de estados.

1. **Importaciones y Contextos**
    - useContext, useState, useEffect: Hooks de React para manejar estados y contextos.
    - NotificationContext, EditModeContext, UserProfileContext: Contextos personalizados para gestionar notificaciones, editar modos, y datos del usuario.
    - Componentes de react-bootstrap como Button, Table, Form.
2. **Estados del Componente**
    - orders: Almacena la lista de pedidos.
    - loading: Estado booleano para mostrar una animación de carga.
    - comments: Objeto que almacena los comentarios de los pedidos con el índice correspondiente.
    - userProfile: Información del perfil del usuario para controlar permisos de acceso.
    - setNotifications: Contexto para mostrar notificaciones.
3. **Funciones Principales**
    - **handleCommentChange**:
        - Actualiza el estado comments al escribir en el campo de texto.
    - **handleUpdateComment**:
        - Llama a updateComment para actualizar solo el comentario en el servidor.
        - Muestra notificaciones de éxito o error.
    - **handleAdvanceOrder**:
        - Llama a editOrders para avanzar el estado del pedido y actualizar el comentario.
        - Actualiza el estado local de orders y muestra notificaciones.
    - **useEffect**:
        - Se ejecuta al montar el componente para obtener la lista de pedidos.

4. **Restricción de Acceso**
    - Muestra la página de error ErrorPage si el rol del usuario no es 2 o 3.
5. **Interfaz de Usuario**
    - Se muestra una tabla con los pedidos.
    - Los usuarios en modo edición (editMode) pueden:
        - Agregar o actualizar comentarios.
        - Avanzar al siguiente estado del pedido, excepto cuando ya está marcado como "Delivered".
        - Actualizar comentarios para pedidos entregados.
6. **Manejo de Estados de Pedido**
    - Los estados se traducen a texto legible ("Order Accepted", "Preparing", "On its way", "Delivered").
    - El botón "Next State" avanza el estado y se desactiva cuando el pedido ya está entregado.
    - Un botón adicional "Update Comment" aparece cuando el pedido está en estado "Delivered".

Product Detail

Proporciona una experiencia completa para mostrar la información detallada de un producto, gestionar la adición de ese producto al carrito y manejar comentarios de los usuarios.

1. **Importaciones y Contextos**
    - useContext, useEffect, useState: Hooks de React para manejar estados, efectos y contextos.
    - Componentes de react-bootstrap como Button, Image, Card, Form, Container.
    - axios: Se utiliza para hacer solicitudes HTTP a la API.
    - Contextos personalizados: WindowWidthContext para manejar el ancho de la ventana y UserProfileContext para manejar los datos del usuario.
2. **Estados del Componente**
    - data: Guarda los datos del producto.
    - loadingProducts: Controla la carga de datos del producto.
    - amount: Guarda la cantidad seleccionada para añadir al carrito.
    - comments: Guarda los comentarios del producto.
    - loadingComments: Controla la carga de los comentarios.
    - newComment: Guarda el contenido del nuevo comentario que el usuario está escribiendo.
    - replyTo: Identifica si el usuario está respondiendo a un comentario específico.
3. **Funciones Principales**
    - **handleCreate**: Envía una solicitud para crear un pedido con los datos del usuario y el producto.
    - **handleData**: Obtiene los datos del producto desde la API utilizando params.id.
    - **handleGetComments**: Obtiene los comentarios del producto y los guarda en el estado comments.
    - **handleSubmitComment**: Envía un nuevo comentario o una respuesta a un comentario existente, actualizando la lista de comentarios tras el envío.
    - **renderComments**: Renderiza los comentarios y sus respuestas de forma jerárquica.
4. **Efectos**

- **useEffect**: Se ejecuta cuando params.id cambia, cargando los datos del producto y los comentarios.

5. **Renderizado del Componente**
    - **Imagen y Descripción del Producto**: Se muestra la imagen del producto, nombre, descripción, precio y stock.
    - **Formulario de Cantidad y Botón de Carrito**: Permite al usuario seleccionar la cantidad y añadir el producto al carrito.
    - **Sección de Comentarios**: Muestra los comentarios en una tarjeta con capacidad de desplazamiento y permite a los usuarios escribir nuevos comentarios o responder a comentarios existentes.
    - **Formulario de Comentarios**: Aparece solo si el usuario tiene un rol diferente de 0 (lo que significa que está autenticado y tiene permisos).

Detalles Técnicos

- **Validaciones en el Campo de Cantidad**: Se asegura de que solo se puedan introducir números positivos y se limita la cantidad al stock disponible.
- **Sistema de Comentarios**:
    - Los comentarios se muestran de forma jerárquica, con la posibilidad de responder a comentarios específicos.
    - Se muestra un botón para responder solo si el usuario tiene los permisos adecuados (rol diferente de 0).
- **Manejo de API**:
    - Se utilizan solicitudes axios.post y axios.get para interactuar con la API.
    - Las solicitudes se manejan de forma asíncrona, con manejo de errores.

Profile

Está diseñado para proporcionar a los usuarios una forma de actualizar su perfil, incluyendo detalles personales, dirección, número de teléfono, así como la capacidad de cambiar su contraseña y gestionar su perfil de facturación.

1. **Estados del Componente**
    - **activeBilling**: Controla si el perfil de facturación está activo.
    - **activePassword**: Controla si el cambio de contraseña está activo.
    - **newPass** y **confirmPass**: Guardan la nueva contraseña y su confirmación.
    - **validated**: Controla la validación del formulario.
    - **oldPassword**: Guarda la contraseña anterior del usuario.
    - **tempProfile**: Almacena temporalmente los datos del perfil mientras se realizan ediciones.
2. **Funciones Principales**
    - **handleClose**: Cierra el modal y reinicia los estados activeBilling y tempProfile.
    - **handleInput**: Actualiza los datos temporales del perfil en función de los cambios realizados en los campos de entrada.
    - **handleSubmit**: Valida y envía los cambios al servidor. Si activePassword está activado, verifica que las contraseñas coincidan y actualiza la contraseña en tempProfile antes de enviar.
        - Si las contraseñas no coinciden, se muestra una notificación de error.
        - Si la contraseña anterior no es correcta, se muestra un mensaje de error.
    - **updateProfile**: Se utiliza para enviar los datos actualizados del perfil al servidor (suponiendo que exista la función en ../conections/updateProfile).
3. **Formulario de Entrada**
    - Incluye campos para actualizar:
        - **Nombre** (First Name)
        - **Apellido** (Last Name)
        - **Correo Electrónico** (Email)
        - **Dirección** (Address)
        - **Fecha de Nacimiento** (BirthDate)

        - **Número de Teléfono** (Phone Number)
    - **Conmutadores de Cambio**: Se utilizan Form.Check para activar o desactivar el perfil de facturación y el cambio de contraseña.
        - Si el cambio de contraseña está activado, se muestran campos adicionales para ingresar la contraseña anterior y la nueva.
        - Si el perfil de facturación está activado, se muestran campos para ingresar el número de tarjeta y la fecha de vencimiento.

Sign Up

Proporciona una interfaz para registrar nuevos usuarios.

1. **Estados del Componente**
    - **user**: Un objeto que almacena la información del nuevo usuario, como nombre, correo electrónico, contraseña, etc.
    - **activeBilling**: Un booleano que controla si la sección de perfil de facturación está activa.
    - **confirm**: Un booleano que se usa para verificar si la confirmación de la contraseña coincide con la contraseña.
    - **setNotifications**: Se utiliza para manejar las notificaciones a través del NotificationContext.
2. **Funciones Principales**
    - **handleInput**: Actualiza el estado user cuando el usuario ingresa información en los campos del formulario.
    - **handleClose**: Cierra el modal y reinicia el estado activeBilling.
    - **onSubmit**: Llama a la función signUp para registrar al usuario, pasando la información del formulario.
3. **Formulario de Registro**
    - **Campos de Entrada**: El formulario incluye campos para:
        - **Nombre** (First Name)
        - **Apellido** (Last Name)
        - **Correo Electrónico** (E-Mail)
        - **Contraseña** (Password) y **Confirmación de Contraseña**
        - **Fecha de Nacimiento** (Birth Date)
        - **Dirección** (Address)
        - **Número de Teléfono** (Phone number)
    - **Verificación de Contraseñas**: Se verifica si la contraseña y su confirmación coinciden para actualizar el estado confirm.
    - **Perfil de Facturación**: Un conmutador que permite al usuario activar o desactivar la sección de perfil de facturación. Si está activado, se muestran los campos para ingresar el número de tarjeta y la fecha de vencimiento.

Users List

Proporciona una interfaz para la gestión de usuarios en tu aplicación.

Desglose del Componente UsersList

1. **Estados del Componente**
    - **users**: Almacena la lista de usuarios obtenida de la base de datos.
    - **loading**: Indica si los datos se están cargando (útil para mostrar un estado de carga).
    - **showNewUserAdmin**: Controla la visibilidad del modal para agregar un nuevo usuario.
2. **Contextos Utilizados**
    - **NotificationContext**: Utilizado para mostrar notificaciones de éxito o error.
    - **EditModeContext**: Determina si el modo de edición está activado.
3. **Funciones Principales**
    - **handleInput**: Se encarga de actualizar los valores de los usuarios cuando se editan.
    - **handleValue**: Cambia el valor de un campo específico en el objeto user de un usuario seleccionado.
    - **useEffect**: Se ejecuta al montar el componente para cargar la lista de usuarios llamando a getUsers.
4. **Renderización Condicional**
    - Si **loading** es true, se muestra un componente LoadingState.
    - Si **editMode** es true, se muestran formularios y controles interactivos para editar los datos del usuario, como campos de entrada y un menú desplegable para cambiar el rol.
    - Si **editMode** es false, se muestran los datos de los usuarios en un formato de solo lectura.
5. **Interacciones con el Servidor**
    - **getUsers**: Carga la lista de usuarios al inicio.
    - **editUser**: Se utiliza para guardar los cambios realizados en un usuario específico.
    - **deleteUser**: Elimina un usuario de la base de datos y actualiza la lista.
6. **Agregar Nuevo Usuario**

- Un botón con un ícono FaUserPlus abre el modal NewUserAdmin para agregar un nuevo usuario.

Ejemplo de Flujo de Trabajo

1. Al cargar el componente, se obtiene la lista de usuarios y se establece en el estado users.
2. Si el modo de edición está activado, los datos de los usuarios se pueden modificar y guardar mediante el botón "Save" o eliminar con "Delete".
3. Se puede agregar un nuevo usuario mediante el botón "Add New User", que abre el modal NewUserAdmin.

API’s

Add Comment

Descripción General

Este script PHP maneja la inserción de comentarios en una base de datos. Utiliza datos recibidos en formato JSON para agregar un comentario a un producto específico, ya sea como un comentario principal o una respuesta a otro comentario.

Estructura del Script

1. **Importación de Dependencias**
    - connection.php: Se importa para establecer la conexión con la base de datos.
2. **Recepción y Validación de Datos**
    - Los datos se obtienen desde la solicitud HTTP y se decodifican desde formato JSON.
    - Se validan los datos obligatorios: id_user, product_name y comment. Si alguno está ausente, se envía un mensaje de error y se detiene el script.
3. **Consulta del Producto**
    - Se ejecuta una consulta SQL para obtener el id_products basado en el product_name.
    - Si se encuentra el producto, se procede con la inserción del comentario. Si no, se devuelve un error.
4. **Inserción del Comentario**
    - Se construye una consulta SQL para insertar el comentario en la tabla comment.
    - Si id_prevcomment es NULL, el comentario se inserta como principal. Si tiene un valor, se inserta como respuesta a un comentario anterior.
    - Se ejecuta la consulta y se devuelve un mensaje indicando el éxito o el error de la operación.
5. **Cierre de la Conexión**
    - Se cierra la conexión a la base de datos.

Bproducts

Descripción General

Este script PHP gestiona la obtención de productos más vendidos desde la base de datos. Permite filtrar productos en función de las categorías proporcionadas a través de los parámetros de la URL (category y subCategory). Utiliza consultas SQL para obtener la información y transforma las imágenes en un formato base64 para enviarlas en la respuesta JSON.

Estructura del Script

1. **Importación de Dependencias**
    - connection.php: Importa la conexión a la base de datos.
2. **Configuración Inicial**
    - Se configura el encabezado para definir el método HTTP permitido y el tipo de contenido de la respuesta.
    - Se captura el método de la solicitud y se redirige a la función getBestProducts si el método es GET.
3. **Funciones Principales**
    - **getBestProducts**: Se encarga de:
        - Obtener la configuración de la base de datos, como el minimum_stock.
        - Filtrar productos según las categorías definidas (o no) en los parámetros de la URL.
        - Construir y ejecutar consultas SQL para recuperar los productos.
        - Procesar las imágenes en formato base64 y agregar información de las categorías relacionadas.
        - Filtrar y devolver los productos en formato JSON.
4. **Consulta y Filtrado de Productos**
    - Si no se especifica ninguna categoría, se devuelven los productos más vendidos, limitados a 5.
    - Si se define solo una categoría, se realiza una búsqueda específica de productos relacionados con esa categoría.
    - Si se definen ambas categorías, la consulta busca productos que pertenezcan a ambas.
5. **Formato de la Respuesta**

- Los productos se devuelven como un array JSON, incluyendo campos como id_products, product_name, description, price, image (codificada en base64), stock, important, enabled, y date.
- Cada producto también incluye una lista de categories asociadas.

Categories

Descripción General

Este script PHP es un servicio API que gestiona las categorías y productos en una base de datos. Permite obtener categorías, así como realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en productos mediante diferentes métodos HTTP.

Estructura del Script

1. **Importación de Dependencias**
    - connection.php: Se incluye para establecer la conexión con la base de datos.
2. **Configuración Inicial**
    - Se definen los encabezados HTTP para permitir las solicitudes CORS y especificar el tipo de contenido JSON.
    - Se captura el método de la solicitud (GET, POST, PUT, DELETE) y se redirige a la función correspondiente.
3. **Funciones CRUD**
    - **getCategories**:
        - Realiza una consulta SQL para obtener las categorías de la base de datos.
        - Si hay resultados, se envían como un array JSON.
        - Si no hay resultados, devuelve un mensaje indicando que no se encontraron productos.
    -

Categories2

Descripción General

Este script PHP define un servicio API RESTful para gestionar categorías en una base de datos. Las operaciones permitidas son la creación, lectura, actualización y eliminación (CRUD) de categorías mediante diferentes métodos HTTP. El código se estructura utilizando funciones específicas para cada operación.

Funciones CRUD

1. **createCategory**
    - **Descripción**: Inserta una nueva categoría en la tabla category.
    - **Parámetros**:
        - name: Nombre de la categoría, recibido en el cuerpo de la solicitud.
    - **Operación**:
        - Inserta la nueva categoría con el valor isfeatured predeterminado como 0.
        - Retorna un mensaje de éxito o un error detallado si la operación falla.
2. **getCategories**
    - **Descripción**: Recupera todas las categorías de la tabla category.
    - **Operación**:
        - Ejecuta una consulta SELECT * FROM category.
        - Si se encuentran resultados, los convierte a un array y los envía como JSON.
        - Si no se encuentran resultados, devuelve un mensaje indicando que no hay categorías disponibles.
3. **updateCategory**
    - **Descripción**: Actualiza el nombre y el estado isfeatured de una categoría.
    - **Parámetros**:
        - id: Nombre actual de la categoría (usado como identificador).
        - name: Nuevo nombre de la categoría.
        - isfeatured: Estado de si es una categoría destacada.
    - **Operación**:
        - Actualiza los campos name y isfeatured en la tabla category.

        - Retorna un mensaje de éxito o un error detallado si la operación falla.

4. **deleteCategory**
    - **Descripción**: Elimina una categoría y sus referencias en la tabla product_category.
    - **Parámetros**:
        - id: ID de la categoría a eliminar, pasado como parámetro en la URL.
    - **Operación**:
        - Elimina primero las referencias de product_category usando el id_category.
        - Luego, elimina la categoría de la tabla category.
        - Retorna un mensaje de éxito o un error detallado si la operación falla.

Checkout

Descripción General

Este script PHP gestiona la actualización del estado de un pedido y recalcula el total de la orden en una transacción segura. Utiliza el método POST para recibir el id_order y realiza las siguientes acciones en la base de datos:

1. Actualiza el estado del pedido en order_dp.
2. Inserta un nuevo registro en orders_states con un comentario sobre el estado.
3. Calcula el total del pedido usando una función de base de datos.
4. Obtiene las configuraciones de envío y calcula el total final, teniendo en cuenta los costos de envío.
5. Actualiza el total en la tabla order_dp.
6. Asegura que todas las operaciones se realicen correctamente usando una transacción para mantener la integridad de los datos.

Estructura del Código

1. **Iniciar una Transacción**
    - begin_transaction(): Se inicia la transacción para garantizar que todas las operaciones se ejecuten juntas. Si una de ellas falla, se revertirán todas las operaciones.
2. **Actualizar el Estado del Pedido**
    - UPDATE order_dp SET state = 2: Cambia el estado del pedido a "2", lo que indica que el pedido ha sido recibido y está siendo procesado.
3. **Registrar el Estado en orders_states**
    - Inserta un nuevo estado con la fecha actual y un comentario informativo: "Your order has been received and is being processed".
4. **Recalcular el Total del Pedido**
    - SELECT order_total($id_order): Llama a una función en la base de datos para calcular el total de la orden.
    - SELECT minimum, shipping FROM ordersettings: Obtiene los valores de envío y mínimo desde la tabla de configuración.
    - Calcula el total final con o sin envío, dependiendo de si el total supera el mínimo requerido.
5. **Actualizar el Total en order_dp**

- UPDATE order_dp SET total = $final_total: Actualiza el total del pedido con el valor calculado.

6. **Confirmar o Revertir la Transacción**
    - commit(): Confirma la transacción si todo es exitoso.
    - rollback(): Revertir la transacción si ocurre algún error durante el proceso.

Confirm

Descripción General

Este script PHP obtiene una lista de tokens de todos los usuarios en la base de datos. Se conecta a la base de datos, ejecuta una consulta para seleccionar los tokens y devuelve los resultados en formato JSON.

Estructura del Código

1. **Incluir la Conexión a la Base de Datos**
    - include 'connection.php';: Se importa el archivo connection.php para establecer la conexión con la base de datos.
2. **Consulta SQL para Obtener Tokens**
    - SELECT token FROM users: Se ejecuta una consulta SQL para seleccionar la columna token de la tabla users.
3. **Procesar los Resultados**
    - Si hay filas en el resultado ($result->num_rows > 0), se recorren todas las filas con fetch_assoc() y se agregan a un array $tokens.
    - Los tokens se codifican en formato JSON y se envían como respuesta.

Disable user

Descripción General

Este script PHP utiliza PHPMailer para enviar notificaciones por correo electrónico a los usuarios que han sido deshabilitados y marcados como notificados. El script selecciona a estos usuarios desde la base de datos y les envía un correo electrónico informándoles sobre el estado de su cuenta.

Estructura del Código

1. **Importar Dependencias y Configurar PHPMailer**
    - require 'Exception.php', require 'PHPMailer.php', require 'SMTP.php': Se cargan las clases necesarias de PHPMailer.
    - use PHPMailer\PHPMailer\PHPMailer;: Se usa el espacio de nombres para PHPMailer.
2. **Configuración de PHPMailer**
    - Configura PHPMailer con los datos del servidor SMTP, el correo electrónico de origen y las credenciales.
3. **Seleccionar Usuarios Deshabilitados**
    - SELECT email, first_name FROM users WHERE active = 0 AND notified = 1: Se buscan los usuarios deshabilitados que aún no han sido notificados.
    - Si se encuentran usuarios, se envían correos electrónicos.
4. **Enviar Correo Electrónico a Cada Usuario**
    - Se configura y envía un correo electrónico personalizado a cada usuario.
    - Tras enviar el correo, se actualiza la base de datos para marcar al usuario como notificado: UPDATE users SET notified = 2.

Detail products

Descripción General

Este script PHP maneja solicitudes GET para devolver información detallada sobre un producto específico.

Funcionamiento Principal

1. **Recepción de Solicitud**: Se verifica si la solicitud es GET.
2. **Extracción de Parámetro**: Se toma el nombre del producto desde la URL ($_GET['product']).
3. **Consulta y Respuesta**:
    - Se busca el producto en la base de datos con ese nombre.
    - Si se encuentra, se devuelve la información del producto (incluyendo imagen codificada en Base64) en formato JSON.

Forgot

Descripción General

Este script PHP gestiona solicitudes POST, GET y PUT para el sistema de restablecimiento de contraseñas de usuarios, usando PHPMailer para el envío de correos electrónicos.

Funcionamiento

1. **Carga de Variables de Entorno**:
    - Se carga un archivo .env.local con variables de configuración como la dirección IP.
2. **Manejo de Solicitudes**:
    - **POST**:
        - Genera un token único y envía un correo electrónico con un enlace de restablecimiento de contraseña al usuario.
        - El correo incluye un enlace que redirige a una URL de restablecimiento.
        - Inserta el token en la base de datos.
    - **GET**:
        - Recupera todos los tokens de restablecimiento de la base de datos.
    - **PUT**:
        - Actualiza la contraseña del usuario y marca el token como usado en la base de datos.

Featured Products

Descripción General

Este script maneja una solicitud GET para recuperar productos destacados, permitiendo filtrar por categorías y subcategorías.

Funcionamiento del Código

1. **Recepción de Categorías**:
    - Se obtienen category y subCategory de los parámetros de la URL para definir la lógica de filtrado.
2. **Consulta Principal**:
    - Recupera los productos marcados como importantes (important = 1) y limita los resultados a 5 productos.
3. **Relación Producto-Categoría**:
    - Se realiza una consulta secundaria para obtener las categorías asociadas a cada producto.
4. **Filtrado de Resultados**:
    - Se filtran productos asegurando que tengan los datos clave completos: id_products, product_name, description, price, y stock.
5. **Manejo de Escenarios**:
    - **Sin Categorías**: Si no se especifica ninguna categoría, devuelve todos los productos importantes.
    - **Con Categoría Única**: Si se especifica solo category, devuelve productos que pertenezcan a dicha categoría.
    - **Con Categoría y Subcategoría**: Filtra productos que pertenezcan a ambas categorías y subcategorías.
6. **Codificación de Imágenes**:
    - Las imágenes de los productos se codifican en base64 antes de enviarlas.

Ejemplo de Uso

- Si category o subCategory están definidos en la URL, el script los usa para realizar consultas específicas.

Get Comments

Descripción General

Este script PHP se encarga de manejar una solicitud para obtener comentarios de un producto específico de una base de datos.

Funcionamiento Específico

1. **Obtener y Validar el Nombre del Producto**:
    - El script toma el nombre del producto desde una solicitud GET.
    - Se utiliza una función de seguridad para escapar caracteres especiales, previniendo ataques de inyección SQL.
2. **Consulta para Obtener el ID del Producto**:
    - Se ejecuta una consulta a la base de datos para encontrar el id del producto basado en su nombre.
3. **Obtener Comentarios Asociados**:
    - Si el producto existe, el script realiza una segunda consulta para recuperar todos los comentarios asociados.
    - Los comentarios incluyen información como el texto del comentario y los nombres del usuario que hizo el comentario.
4. **Organizar y Devolver los Datos**:
    - Los comentarios se almacenan en un array y se envían de vuelta al cliente en formato JSON.
    - Si no se encuentran productos o comentarios, el script devuelve un mensaje de error apropiado.

Login

Descripción General

Este código PHP maneja las solicitudes de autenticación de usuario, verificando si los credenciales proporcionados (correo electrónico y contraseña) son correctos.

Funcionamiento Específico

1. **Recibir y Procesar Datos de la Solicitud**:
    - El script recibe datos en formato JSON, que incluye el correo electrónico y la contraseña del usuario.
    - Los datos se decodifican y se asignan a las variables $email y $password.
2. **Consulta de Autenticación**:
    - Se realiza una consulta a la base de datos para buscar un usuario cuyo correo y contraseña coincidan con los proporcionados.
    - Si se encuentra un usuario, se recupera toda la información relevante: email, rol, nombre, fecha de nacimiento, dirección, estado de cuenta (activo/inactivo), número de tarjeta, fecha de expiración, última conexión, número de teléfono, ID de usuario, contraseña, y token.
3. **Respuesta Exitosa o Error**:
    - Si las credenciales son correctas ($result->num_rows > 0), se devuelve una respuesta JSON con todos los detalles del usuario, indicando que la autenticación fue exitosa.

Login confirm

Descripción General

Este script PHP maneja la verificación y activación de la cuenta de usuario basada en un **token** y las credenciales proporcionadas.

Funcionamiento del Código

1. **Recepción de Datos**:
    - El código recibe datos en formato JSON, incluyendo el correo electrónico ($email), la contraseña ($password), y el token ($token).
    - Estos datos se decodifican desde el cuerpo de la solicitud y se asignan a las variables correspondientes.
2. **Consulta de Usuario en la Base de Datos**:
    - Se ejecuta una consulta para verificar si existe un usuario con el email y la contraseña proporcionados.
    - Si se encuentra un usuario ($result->num_rows > 0):
        - **Verificación del Token y Estado de Activación**:
            - Si el **token coincide** y el usuario ya está activo (active == 1), se envía un mensaje JSON indicando que el usuario ya está confirmado (status: "already").
            - Si el **token coincide** y el usuario está inactivo (active == 0), se actualiza el estado a **activo** (active = 1) en la base de datos, y se devuelve un mensaje de éxito (status: "successful").
            - Si el **token no coincide**, se envía un mensaje de error indicando la discrepancia del token.

Low Products

Descripción General

Este script PHP se encarga de gestionar solicitudes para obtener productos con bajo stock. Se divide en dos partes principales: la gestión de la solicitud y la lógica para filtrar los productos en función de las categorías y el umbral de stock mínimo.

Cómo Funciona

1. **Gestión de Solicitud HTTP**:
    - El script verifica el método de la solicitud. Si es GET, se llama a la función getLowStockProducts(). De lo contrario, se devuelve un mensaje de error.
2. **Obtener Configuración de Stock Mínimo**:
    - Se realiza una consulta para obtener el stock mínimo permitido desde una tabla de configuración (ordersettings). Si no se encuentra un valor, el mínimo se establece en 0.
3. **Filtrar Productos por Stock**:
    - Se ejecutan diferentes consultas SQL dependiendo de si se proporcionaron una o dos categorías:
        - **Sin Categoría**: Se obtienen todos los productos con stock por debajo del umbral y con el campo enabled establecido en 1.
        - **Con Una Categoría**: Se filtran los productos que pertenecen a la categoría proporcionada.
        - **Con Dos Categorías**: Se obtienen productos que pertenecen a ambas categorías.
4. **Asociación de Categorías**:
    - Se consulta una tabla de relación (product_category) para vincular productos con sus categorías y se añade esta información a cada producto.
5. **Codificación de Imágenes y Formato de Salida**:
    - Las imágenes de los productos se codifican en Base64.
    - Los productos se filtran para asegurarse de que tengan los campos necesarios antes de devolverlos en formato JSON.

Navigation

Descripción General

Este script PHP gestiona la interacción con la tabla navigation. Se encarga de procesar las solicitudes POST y GET para marcar y desmarcar categorías y recuperar las categorías marcadas. La funcionalidad es útil para sistemas que permiten a los usuarios seleccionar o deseleccionar categorías y guardar esas selecciones en una base de datos.

Cómo Funciona

1. **Métodos HTTP Soportados**:
    - **POST**: Se utiliza para insertar o eliminar relaciones entre id_category e id_sub en la tabla navigation dependiendo de si el campo is_checked es true o false.
    - **GET**: Se usa para obtener todas las categorías marcadas de la tabla navigation.
2. **Procesamiento de la Solicitud POST**:
    - Se extraen y decodifican los datos enviados en el cuerpo de la solicitud.
    - Los valores (id_category, id_sub, y is_checked) se validan y se escapan para proteger contra inyecciones SQL.
    - Dependiendo de is_checked:
        - Si es true, se ejecuta una consulta INSERT IGNORE para agregar la relación si no existe.
        - Si es false, se ejecuta una consulta DELETE para eliminar la relación.
    - La respuesta se devuelve en formato JSON con un mensaje de éxito o de error.
3. **Procesamiento de la Solicitud GET**:
    - Se consulta la tabla navigation para obtener todas las relaciones id_category y id_sub.

Navigation 2

Descripción General

Este script PHP gestiona solicitudes para marcar y desmarcar categorías y sus subcategorías en la base de datos, además de recuperar las categorías y subcategorías marcadas. La lógica está diseñada para manejar tanto la inserción como la eliminación de relaciones entre categorías y subcategorías, así como para devolver estas relaciones de una manera estructurada.

Cómo Funciona

1. **Métodos HTTP Soportados**:
    - **POST**: Inserta o elimina una relación entre id_category e id_sub en la tabla navigation dependiendo del estado is_checked proporcionado.
    - **GET**: Recupera todas las relaciones de categorías y subcategorías de la tabla navigation y las estructura en un formato fácil de manejar.
2. **Solicitud POST**:
    - **Entrada**: JSON con id_category, id_sub, y is_checked.
    - **Acciones**:
        - Escapa los valores para proteger contra inyecciones SQL.
        - Si is_checked es true, inserta la relación usando INSERT IGNORE para evitar duplicados.
        - Si is_checked es false, elimina la relación correspondiente.
    - **Respuesta**: JSON indicando éxito o error, con detalles en caso de fallo.
3. **Solicitud GET**:
    - **Acciones**: Ejecuta una consulta para obtener todas las categorías y subcategorías marcadas, uniendo las tablas category y navigation para recuperar los nombres de las subcategorías.
    - **Estructuración**: Organiza las subcategorías bajo cada categoría principal en un array.
    - **Respuesta**: JSON con la estructura de categorías y sus subcategorías marcadas.

New Products

Descripción General

Este código PHP maneja solicitudes GET para obtener productos con fechas recientes desde la base de datos, usando un intervalo basado en la configuración mínima de días. Se estructura en un formato JSON, organizando productos por categorías.

Cómo Funciona

1. **Configuración**:
    - El script acepta solicitudes GET y devuelve los productos en JSON.
    - Configura los encabezados para permitir solicitudes desde cualquier origen y establece el contenido como JSON.
2. **Consulta de Configuración**:
    - Recupera el valor minimum_days de la tabla ordersettings, que se usa para filtrar productos recientes con respecto a la fecha actual.
3. **Filtrado de Productos**:
    - Si no se especifica ninguna categoría, selecciona hasta 5 productos habilitados con fechas recientes.
    - Si se define category, filtra los productos en esa categoría, limitando la cantidad a 5.
    - Si category y subCategory están presentes, recupera productos que pertenecen a ambas categorías, ordenándolos y limitando la cantidad.
4. **Procesamiento de Categorías**:
    - Une los productos con sus respectivas categorías mediante una consulta a la tabla product_category.
    - Filtra y organiza productos con todos los campos necesarios, como id_products, product_name, description, price, y stock.

Order List

Descripción General

Este código PHP gestiona solicitudes GET para interactuar con órdenes en la base de datos.

Resumen de Funcionalidades

1. **GET - getOrders()**:
    - Recupera las órdenes con sus estados de la base de datos y devuelve un array de objetos JSON.
    - Utiliza una consulta simple para seleccionar id_order y state de la tabla order_dp.

Orders

Este código PHP maneja solicitudes para gestionar órdenes y detalles de carrito en la base de datos. Aquí se detallan las funcionalidades implementadas:

Desglose de Funcionalidades

1. **GET - getOrder()**:
    - Recupera los productos en el carrito para un usuario específico.
    - Requiere id_user en la solicitud GET.
    - Realiza una consulta para obtener productos con su id, nombre, imagen, cantidad y precio total en la tabla order_detail mientras la orden esté en estado 1 (activo).
    - Codifica las imágenes en base64 antes de devolverlas como un array JSON.
    - Si no se proporcionan productos, devuelve un mensaje indicando que el carrito está vacío.
2. **POST - createOrder()**:
    - Crea una nueva orden o actualiza una existente para un usuario específico.
    - Lee id_user, id_product y amount del cuerpo de la solicitud.
    - Verifica si ya existe una orden activa (estado = 1) para el usuario:
        - **Si existe**: Utiliza id_order de esa orden.
        - **Si no existe**: Crea una nueva orden en order_dp y usa su id_order.
    - Comprueba si el producto ya está en la orden:
        - **Si está**: Actualiza la cantidad del producto.
        - **Si no está**: Inserta el producto como un nuevo detalle en order_detail.
    - Calcula el total de la orden llamando a una función order_total() y actualiza el total en order_dp.
    - Devuelve mensajes JSON indicando el éxito o errores específicos en las operaciones.

Paid Orders

Este código maneja solicitudes HTTP para recuperar órdenes de la base de datos para un usuario específico.

Desglose de Funcionalidad

1. **GET - getOrder()**:
    - Recupera las órdenes para un usuario con id_user proporcionado en la solicitud GET.
    - Consulta las órdenes desde orders_states y order_dp donde el estado de la orden (state) es mayor que 1 (probablemente indicando que la orden ha avanzado en el proceso).
    - Las órdenes se ordenan por id_order de forma ascendente.
    - Si se encuentran órdenes, las devuelve como un array JSON.
    - Si no se encuentran órdenes o si no se proporciona el id_user, devuelve mensajes de error o información.

Product Order

Este código maneja las solicitudes de actualización y eliminación de detalles de productos en órdenes de la base de datos. Se enfoca en modificar la cantidad de un producto en la orden o eliminar un producto específico de una orden de un usuario y actualizar el total de la orden en consecuencia.

Desglose de Funcionalidad

1. **DELETE - deletePOrder()**:
    - **Propósito**: Eliminar un producto específico de la orden de un usuario.
    - **Entradas**: id (ID del usuario) y product (ID del producto) pasados como parámetros en la URL.
    - **Proceso**:
        - Elimina el producto de la tabla order_detail si la orden pertenece al usuario y está en estado "1" (probablemente indicando una orden activa o en proceso).
        - Si el producto se elimina correctamente, se obtiene el id_order correspondiente al usuario.
        - Se recalcula el total de la orden usando una función llamada order_total(), y se actualiza el total en la tabla order_dp.
    - **Respuesta**: Devuelve un mensaje de éxito o un error en caso de fallo.
2. **PUT - updatePOrder()**:
    - **Propósito**: Actualizar la cantidad de un producto en la orden de un usuario.
    - **Entradas**: id (ID del usuario), product (ID del producto), y amount (nueva cantidad) pasados como parámetros en la URL.
    - **Proceso**:
        - Actualiza la cantidad del producto en order_detail y establece total_product_price como NULL, para ser recalculado más tarde.
        - Recupera el id_order correspondiente al usuario y recalcula el total de la orden usando order_total().
        - Actualiza el total de la orden en order_dp.
    - **Respuesta**: Devuelve un mensaje de éxito o un error si ocurre un problema.

Products

Gestiona datos de productos incluyendo la creación, actualización, eliminación y obtención de detalles de productos y sus categorías asociadas.

Funcionalidad

GET (getProduct)

- **Propósito**: Obtener detalles de productos, incluyendo nombre, descripción, precio, stock y categorías asociadas.
- **Proceso**:
    - Si no se especifica ninguna categoría, se obtienen todos los productos y sus categorías.
    - Si se proporciona solo una categoría principal, se obtienen los productos asociados a esa categoría.
    - Si se proporcionan tanto la categoría principal como la secundaria, se obtienen los productos que pertenecen a ambas.
- **Salida**: Devuelve un array en formato JSON con los detalles de los productos, incluyendo imágenes codificadas y los IDs de las categorías.

POST (createProduct)

- **Propósito**: Crear un nuevo producto en la base de datos.
- **Proceso**:
    - Extrae datos del cuerpo de la solicitud, como nombre del producto, descripción, precio, stock y categorías.
    - Inserta el producto en la tabla products y asocia las categorías especificadas en product_category.
- **Salida**: Devuelve un mensaje de éxito o un error si la inserción falla.

PUT (updateProduct)

- **Propósito**: Actualizar los detalles de un producto existente.
- **Proceso**:
    - Actualiza los detalles del producto, como nombre, descripción, precio, stock e imagen (si se proporciona).
    - Elimina las asociaciones antiguas de categorías y agrega las nuevas.
- **Salida**: Devuelve un mensaje de éxito o un error si la actualización falla.

DELETE (deleteProduct)

- **Propósito**: Eliminar un producto y sus asociaciones de categorías.
- **Proceso**:
    - Elimina las asociaciones del producto en product_category y luego elimina el producto de la tabla products.
- **Salida**: Devuelve un mensaje de éxito o un error si la eliminación falla.

Profile

Es una API para actualizar la información de un usuario, como nombre, correo electrónico, fecha de nacimiento, dirección, número de teléfono y detalles de la tarjeta. También se puede actualizar la contraseña si se proporciona.

Funcionalidad

POST (Actualización de Usuario)

- **Propósito**: Actualizar los detalles del usuario, incluyendo nombre, apellido, correo electrónico, fecha de nacimiento, dirección, número de teléfono y detalles de la tarjeta.
- **Proceso**:
    - Se reciben los datos del usuario desde el cuerpo de la solicitud.
    - Si se proporciona una contraseña, se incluye en la actualización.
    - Se construye una consulta SQL para actualizar los campos proporcionados en la tabla de usuarios.
    - Se ejecuta la consulta y se verifica si se realizó correctamente.

Search Products

Este script en PHP es una API que busca y filtra productos según ciertos criterios utilizando un término de búsqueda y filtros específicos proporcionados en el cuerpo de la solicitud. La búsqueda se realiza en la tabla de productos, considerando campos como el nombre del producto, la descripción y la categoría asociada.

Funcionalidad del POST

- **Propósito**: Obtener una lista de productos que coincidan con ciertos filtros y un término de búsqueda.
- **Parámetros**:
    - filters: Un array de booleanos que indica si se debe aplicar un filtro específico para el nombre, descripción o categoría.
    - searchTerm: Un término de búsqueda utilizado para filtrar los productos.
- **Proceso**:

1. Se construye una consulta SQL base que une la tabla de productos con las categorías.

2. Se crean condiciones WHERE basadas en los filtros proporcionados.

3. Se utiliza prepared statements para prevenir inyecciones SQL, con parámetros vinculados dinámicamente.

4. Si la consulta se ejecuta con éxito, se devuelven los productos en formato JSON, con las imágenes codificadas en base64.

- **Salida**: Devuelve un array JSON con los productos filtrados o un mensaje de error si la consulta no se puede preparar o si hay parámetros inválidos.

Send email

Esta API se encarga de enviar un correo electrónico con los detalles de los productos en el carrito de compras.

Funcionalidad

Inclusion y Configuración:

- Se incluye PHPMailer y se carga la configuración del servidor de correo desde un archivo separado.

2. **Lectura y Validación de Datos**:
    - Se obtienen los datos de la solicitud (carrito de compras y correo electrónico del usuario).
    - Se validan los datos para asegurarse de que estén completos.
3. **Configuración del Correo**:
    - Se crea una instancia de PHPMailer y se configura el servidor SMTP.
    - Se especifican el remitente, el destinatario y los detalles del correo, incluyendo el asunto y el cuerpo en formato HTML.
4. **Cuerpo del Correo**:
    - Se genera una tabla HTML con los detalles de los productos, incluyendo nombre, cantidad y precio total.
    - Los datos se escapan para prevenir inyecciones XSS.
5. **Envío y Manejo de Errores**:
    - Se intenta enviar el correo. Si se envía correctamente, se devuelve un mensaje de éxito en formato JSON.

Sign Up

Este script en PHP se encarga de crear una cuenta de usuario para un sistema de tienda de mascotas y de enviar un correo de bienvenida con un enlace de confirmación.

Funcionalidad

1. **Carga de Variables de Entorno**
    - Se carga un archivo de configuración .env.local para establecer variables importantes, como la IP local y credenciales del correo, que se usan a lo largo del script.
    - Las variables de entorno se manejan para proporcionar seguridad y flexibilidad.
2. **Configuración del Servidor SMTP**
    - Se configura PHPMailer para conectarse a un servidor SMTP (en este caso, Gmail) con autenticación segura. Las credenciales de acceso provienen de un archivo separado.
3. **Recepción y Validación de Datos**
    - Se extraen los datos de la solicitud PUT, como el nombre, correo, contraseña, número de teléfono, fecha de nacimiento, dirección, y datos de la tarjeta.
    - Se genera un token único para el usuario, que se utiliza en un enlace de confirmación incluido en el correo.
4. **Envío de Correo de Bienvenida**
    - Se envía un correo HTML con un mensaje de bienvenida y un enlace de confirmación para que el usuario pueda iniciar sesión en la tienda de mascotas.
    - En caso de que falle el envío del correo, se muestra un mensaje de error.
5. **Inserción en la Base de Datos**
    - Se inserta la información del usuario en la tabla users con un token generado y otros detalles proporcionados.
    - Si la inserción es exitosa, se devuelve un mensaje JSON indicando éxito. De lo contrario, se devuelve un mensaje de error.

Sign Up admin

Se utiliza para crear un nuevo usuario y enviar un correo electrónico de bienvenida con la ayuda de la biblioteca PHPMailer. Se encarga de manejar la recepción de datos, la inserción en la tabla users y la notificación por correo electrónico.

Funcionalidad

1. **Configuración Inicial**
    - Se incluye un archivo de conexión connection.php para conectar con la base de datos.
    - Se importa PHPMailer y sus dependencias necesarias para enviar correos.
    - Se configuran los encabezados HTTP para manejar solicitudes desde cualquier origen.
2. **Recepción y Validación de Datos**
    - Los datos se extraen del cuerpo de la solicitud (método POST) en formato JSON, incluyendo el nombre, apellido, correo electrónico, contraseña, fecha de nacimiento y rol.
3. **Generación de Token**
    - Se genera un token único usando random_bytes que será almacenado en la base de datos y se podría utilizar para futuros procesos de autenticación o verificación.
4. **Envío de Correo Electrónico**
    - Se configura PHPMailer para enviar un correo HTML con un mensaje de bienvenida a la dirección proporcionada por el usuario.
    - Si el correo se envía correctamente, se continúa con el proceso. Si no, se muestra un mensaje de error.
5. **Inserción en la Tabla users**
    - Se construye y ejecuta una consulta SQL para insertar los datos del nuevo usuario en la tabla users.
    - Si la inserción es exitosa, se devuelve un mensaje JSON indicando éxito. Si falla, se devuelve un error con detalles del problema.

Total

Es una API sencilla para obtener el total de una orden específica junto con la configuración relacionada a la misma, como el mínimo de compra y el costo de envío. Se usa una solicitud GET para obtener los datos necesarios.

Funcionalidad

1. **Configuración Inicial**
    - Se incluye un archivo de conexión connection.php para establecer la conexión con la base de datos.
    - Se configuran los encabezados para permitir el acceso y manejar las solicitudes desde cualquier origen.
2. **Solicitud GET**
    - El script verifica el método de solicitud y, si es GET, ejecuta la función getTotal.
3. **Función getTotal**
    - **Entrada**: Recibe un id_user de los parámetros de la URL.
    - **Consulta 1**: Recupera el total de la orden activa (con estado 1) del usuario proporcionado.
    - **Consulta 2**: Recupera el mínimo de compra y el costo de envío desde la tabla ordersettings.
    - **Consulta 3**: Recupera el id_order de la orden activa del usuario.
    - **Salida**: Devuelve un objeto JSON que contiene el total de la orden, el mínimo de compra, el costo de envío y el id_order.

Update Comment

Proporciona una API para actualizar comentarios asociados a un pedido específico en el sistema. El objetivo principal es modificar solo el comentario en la tabla orders_states cuando el estado del pedido es 5.

Explicación de la Funcionalidad

1. **Configuración Inicial**
    - Se incluye el archivo connection.php para establecer la conexión con la base de datos.
    - Se configuran los encabezados HTTP para permitir solicitudes desde cualquier origen, así como las solicitudes POST, GET, DELETE y PUT.
2. **Datos de Entrada**
    - Se recibe un objeto JSON con id_order y comment a través del cuerpo de la solicitud.
    - El script decodifica estos datos y los asigna a las variables $id_order y $comment.
3. **Consulta SQL para Actualizar el Comentario**
    - Se construye y ejecuta una consulta SQL para actualizar el comentario en la tabla orders_states solo si el estado es 5.
    - La consulta usa los datos proporcionados (id_order y comment) para identificar y modificar el registro correcto.
4. **Respuesta**
    - Si la actualización es exitosa, se devuelve un JSON indicando el éxito con un mensaje.

Update Order State

Proporciona una API para actualizar el estado de un pedido y agregar un registro correspondiente con el nuevo estado y un comentario en las tablas order_dp y orders_states.

Explicación de la Funcionalidad

1. **Configuración Inicial**
    - Se incluye connection.php para establecer la conexión con la base de datos.
    - Se configuran los encabezados HTTP para permitir solicitudes de cualquier origen y los métodos POST, GET, DELETE y PUT.
2. **Datos de Entrada**
    - Se recibe un objeto JSON que contiene id_order, state, y comment.
    - Los datos son decodificados y asignados a variables.
3. **Actualizar el Estado del Pedido**
    - Se ejecuta una consulta SQL para actualizar el estado del pedido en la tabla order_dp usando id_order y new_state.
    - Si la actualización es exitosa, se procede al siguiente paso.
4. **Insertar Registro en orders_states**
    - Se inserta una nueva entrada en orders_states con id_order, el nuevo estado, la fecha actual (NOW()), y el comentario.
    - Si la inserción es exitosa, se devuelve un mensaje de éxito en formato JSON.
    - Si falla, se devuelve un mensaje de error con detalles de MySQL.

Users List

Proporciona una API para gestionar usuarios, permitiendo operaciones de obtención, creación, actualización y eliminación de usuarios a través de solicitudes HTTP. Soporta los métodos GET, POST, PUT y DELETE para interactuar con la tabla users.

Desglose de la Funcionalidad

1. GET (getUsers)

- **Propósito**: Obtener todos los usuarios.
- **Proceso**: Ejecuta una consulta SQL para seleccionar todos los registros de users. Si hay resultados, los guarda en un array y los devuelve en formato JSON.
- **Salida**: Array de usuarios en formato JSON.

2. POST (createUser)

- **Propósito**: Crear un nuevo usuario (sin implementación en el ejemplo).
- **Nota**: Deberías definir esta función si deseas permitir la creación de nuevos usuarios.

3. PUT (updateUser)

- **Propósito**: Actualizar los detalles de un usuario específico.
- **Proceso**:
    - Obtiene id del usuario y los datos actualizados del cuerpo de la solicitud.
    - Ejecuta una consulta SQL para actualizar los campos de users con los datos proporcionados.
    - Devuelve un mensaje de éxito o error.
- **Validaciones Necesarias**: Este método es susceptible a inyecciones SQL; sería más seguro usar consultas preparadas.

4. DELETE (deleteUser)

- **Propósito**: Eliminar un usuario por su id.
- **Proceso**: Obtiene id del usuario de la solicitud y ejecuta una consulta SQL para eliminarlo.
- **Salida**: Devuelve un mensaje de error si la eliminación falla.