Centro Universitario de Occidente

Análisis y Diseño 1

Ing. Daniel González

Segundo Semestre 2024

“Manual Técnico
E-Commerce”

201930693 - Leví Isaac Hernández Sapón
201930699 - Erick Daniel Morales Xicará
201832069 - Marcos Andrés Aguare Bravo
201830498 - Oscar Antonio de León Urizar

MANUAL TÉCNICO **2**
Requisitos del Sistema 2
Descripción del Proyecto 2
Tecnologías Utilizadas 3
Funcionalidades Clave 3
Instalación 4
Frontend (Angular) 4
Backend (Express) 4
Base de Datos (MariaDB) 5
Estructura del Proyecto 6
Backend 6
1. msAdmin 6
2. msCustomer 7
3. msEmail 9
4. msImg 9
5. msProduct 10
Frontend 12
Componentes 12
Servicios (services/) 13
Vistas (Rutas) 14
Base de Datos (MariaDB) 16
Descripción General 16
Tablas Principales 16
1. user 16
2. product 17
3. orders 17
4. inventory 18
5. branch 18
Tablas Relacionales 18
1. order_has_product 18
2. product_has_attribute 19
3. product_has_category 19
Tablas Auxiliares 19
1. role 19
2. attribute 19
3. brand 20
4. category 20
Relaciones Clave 20
Ejecución 21

MANUAL TÉCNICO

Requisitos del Sistema

El software puede ser utilizado en cualquier sistema operativo, ya que es un proyecto web. Está desarrollado utilizando **TypeScript** tanto en el frontend como en el backend. A continuación, se listan los requisitos principales:

- **Frontend:** Angular 17.x+
- **Backend:** Node.js 16.19.0+ con Express
- **Base de datos:** MariaDB 10.x+
- **Herramientas de desarrollo:** IntelliJ, Visual Studio Code
- **Navegadores soportados:** Google Chrome, Mozilla Firefox, Microsoft Edge (últimas versiones)

Descripción del Proyecto

Este software es un **E-commerce** diseñado para una empresa, permitiendo la **compra y venta de productos**. Además, el sistema permite el control de **stock de productos** en diversas sucursales. Entre sus principales características se incluyen:

- **Catálogo de productos:** Los usuarios pueden explorar productos y realizar búsquedas avanzadas.
- **Carrito de compras:** Funcionalidad para agregar, eliminar productos del carrito y modificar cantidades.
- **Gestión de stock:** Módulo de administración de inventario y sucursales.
- **Gestión de usuarios y roles:** Roles diferenciados para clientes y administradores, con acceso a las funcionalidades correspondientes.

- **Pasarelas de pago:** Integración con servicios de pago para procesar transacciones en línea.

Tecnologías Utilizadas

- **Frontend:** Angular 17.x, utilizando TypeScript.
- **Backend:** Express con Node.js.
- **Base de datos:** MariaDB, con consultas y migraciones para el manejo eficiente de datos.
- **Control de versiones:** Git para el control de versiones y la colaboración.

Funcionalidades Clave

1. **Catálogo de Productos:** Navegación por categorías, búsqueda avanzada y visualización de detalles de productos.
2. **Carrito de Compras:** Los usuarios pueden agregar productos al carrito, modificar las cantidades, y proceder con la compra.
3. **Gestión de Inventarios y Stock por Sucursales:** Administración del stock en tiempo real y en diferentes ubicaciones.
4. **Procesamiento de Pagos:** Integración con servicios de pago para gestionar transacciones de forma segura.
5. **Administración de Usuarios:** Gestión de cuentas y roles de usuario.
6. **Notificaciones:** Actualizaciones de estado de las órdenes de compra y cambios en el stock.

Instalación

Frontend (Angular)

- Clonar el repositorio del proyecto:

```
git clone https://github.com/Aguare/Ecommerce-AyD1
```

- Instalar las dependencias:

```
cd E-Commerce/e-frontend
npm install
```

- Levantar el servidor de desarrollo:

```
ng serve
```

Backend (Express)

- Clonar el repositorio del backend:

```
git clone git clone https://github.com/Aguare/Ecommerce-AyD1
```

- Instalar las dependencias:

```
cd E-Commerce/e-backend
npm install
```

1. Configurar las variables de entorno en un archivo .env para conectar la base de datos MariaDB, configurando parámetros como host, usuario y contraseña.

- Levantar el servidor:

```
npm start
```

Base de Datos (MariaDB)

- Instalar MariaDB:

```
sudo apt install mariadb-server
sudo systemctl start mariadb
```

1. La base de datos se encuentra en la nube, solamente es acceder mediante las credenciales

Estructura del Proyecto

Backend

El backend maneja la lógica del negocio y la interacción con la base de datos, implementado con **Express**. Algunas de las principales carpetas son:

1. msAdmin

- **Controllers:**
  - admin.controller.js: Controla la lógica para la administración del sistema.
  - settings.controller.js: Maneja la lógica de ajustes del sistema.
  - users.controller.js: Administra la lógica de gestión de usuarios (posiblemente roles, creación, actualización).
- **Routes:**
  - admin.routes.js: Rutas para gestionar la administración del sistema.
    - **GET /getPages/:** Obtener una página específica según el ID.
    - **POST /addEmployee**: Agregar un nuevo empleado.
    - **GET /getRoles**: Obtener la lista de roles disponibles.
  - settings.routes.js: Rutas para ajustes del sistema.
    - **GET /find/** : Obtener configuraciones del sistema según el nombre especificado.
    - **PUT /update**: Actualizar configuraciones del sistema.
    - **GET /tabs**: Obtener las pestañas de configuración disponibles..
  - users.routes.js: Rutas para gestión de usuarios.
    - **GET /user/image/**: Obtener la imagen de perfil de un usuario por su ID.

        - **GET /user/information/**: Obtener la información de perfil de un usuario por su nombre de usuario.
        - **PUT /user/information/**: Actualizar la información de perfil de un usuario por su ID.

2. msCustomer

- **Controllers:**
    - customer.controller.js: Controla la gestión de clientes, operaciones como agregar, editar, eliminar clientes.
    - order.controller.js: Gestiona las órdenes de compra para los clientes.
- **Routes:**
    - customer.routes.js: Aquí se define todas las rutas relacionadas con clientes.
        - **POST /myCart**: Agregar al carrito.
        - **PUT /updateCart**: Actualizar el carrito.
        - **DELETE /deleteCart/**: Eliminar el carrito por ID.
        - **GET /getStores**: Obtener las tiendas.
        - **POST /validateStock**: Validar stock.
        - **GET /validateStockOnline/**: Validar stock online por ID de producto.
        - **DELETE /deleteProductCart/**: Eliminar producto del carrito por ID de producto.
        - **POST /addProductCart**: Agregar producto al carrito.
        - **GET /getDeliveryCost**: Obtener el costo de envío.
        - **GET /getDataForCheckout/**: Obtener los datos de checkout por ID.
    - order.routes.js: Aquí se define las rutas para las órdenes.
        - **GET /getAllOrders**: Obtener todas las órdenes.

        - **GET /getProductsByOrderId/**: Obtener productos por ID de orden con límite y offset.
        - **GET /getOrderStatus**: Obtener el estado de la orden.
        - **GET /getOrdersByUserId/**: Obtener órdenes por ID de usuario.
        - **PUT /updateOrderStatus/**: Actualizar el estado de la orden por ID.
        - **POST /saveOrder**: Guardar una nueva orden.
    - routes.js: Aquí se define rutas del customer
        - **POST /customer/sign-up**: Registrar un nuevo cliente.
        - **GET /customer/getCurrency**: Obtener la moneda.
        - **GET /customer/getNumberInCart/**: Obtener el número de artículos en el carrito por ID de usuario.
        - **POST /customer/verifyResetPassword**: Verificar el restablecimiento de la contraseña.
        - **USE /customer**: Todas las rutas de gestión de clientes (requiere verificación de token).
        - **USE /order**: Todas las rutas de gestión de órdenes (requiere verificación de token)..

3. msEmail

- **Controllers:**
  - email.controller.js: Controla las operaciones relacionadas con el envío de correos electrónicos.
- **Routes:**
  - email.routes.js: Define las rutas para envío de correos.
    - **POST /sendVerificationEmail**: Enviar un correo electrónico de verificación.
    - **POST /verify-email**: Verificar un correo electrónico.
    - **POST /sendRecoveryPasswordEmail**: Enviar un correo electrónico de recuperación de contraseña.
    - **POST /validateEmail**: Validar si el correo electrónico ya está en uso.
  - routes.js: Aquí se define rutas para los correos electrónicos
    - **USE /email**: Incluye todas las rutas relacionadas con el envío y verificación de correos electrónicos (correspondiente a email.routes.js).

4. msImg

- **Controllers:**
  - upload.controller.js: Maneja la lógica para la subida de imágenes.
- **Routes:**
  - upload.routes.js: Define las rutas para la gestión de imágenes.
    - **POST /upload/product**: Subir una imagen de producto.
    - **POST /upload/client**: Subir una imagen de cliente.
    - **POST /upload/admin**: Subir una imagen de administrador.
    - **GET /company/logo**: Obtener el logo de la compañía.

        - **DELETE /img/deleteImage/**: Eliminar una imagen por su ID.

5. msProduct

- **Controllers:**
    - brand.controller.js: Gestiona las marcas de productos.
    - category.controller.js: Gestiona las categorías de productos.
    - product.controller.js: Maneja la lógica CRUD para productos.
- **Routes:**
    - brand.routes.js: Define rutas para gestionar marcas.
        - **GET /getBrands**: Obtener la lista de marcas.
        - **POST /saveBrand**: Guardar una nueva marca.
        - **PUT /updateBrand**: Actualizar una marca existente.
        - **DELETE /deleteBrand**: Eliminar una marca.
    - categories.routes.js: Define rutas para gestionar categorías.
        - **GET /getCategories**: Obtener la lista de categorías.
    - product.routes.js: Define rutas para gestionar productos.
        - **POST /getProductsByCategory**: Obtener productos por categoría.
        - **GET /getProductsWithCategory/**: Obtener productos por ID de categoría.
        - **GET /getProductsForCart/**: Obtener productos para el carrito por ID.
        - **GET /getProductDetailById/**: Obtener el detalle de producto por ID.
        - **GET /getStockProductById/**: Obtener el stock del producto por ID.
        - **GET /getProductsByCategory/**: Obtener productos por categoría por ID.
        - **POST /saveProduct**: Guardar un nuevo producto.

- **GET /getProducts**: Obtener la lista de productos.
- **GET /getProductById**: Obtener producto por ID.
- **PUT /updateDataProduct**: Actualizar datos de un producto.
- **PUT /updateAttributesProduct**: Actualizar atributos de un producto.
- **GET /getBranchesWithProduct**: Obtener las sucursales que tienen el producto.

Frontend

El frontend está implementado con **Angular 17**, ofreciendo una interfaz visual y responsiva para los usuarios. Algunas de las principales carpetas son:

Componentes

Los componentes están organizados en carpetas dentro de src/app/components. Cada carpeta representa un componente individual o un grupo de componentes relacionados. Aquí algunos de los componentes más importantes:

- **admin/**
    - **add-helper**: Componente para añadir ayudantes en la sección de administración.
    - **company-settings**: Componente para la configuración de la empresa.
    - **dashboard**: Componente que sirve como el panel principal de administración.
    - **orders**: Componente para la gestión de órdenes en la vista de administrador.
    - **store-billing**: Componente relacionado con la facturación de la tienda.
    - **store-config**: Componente para la configuración de la tienda.
- **commons/**
    - **forgot-password**: Componente para recuperar la contraseña.
    - **login**: Componente para la autenticación de usuarios.
    - **register-modal**: Componente para el registro de usuarios, probablemente utilizando un modal.
    - **reset-password**: Componente para restablecer la contraseña de un usuario.
    - **verify-email**: Componente para la verificación de correos electrónicos.
- **home/**: Vistas y componentes relacionados con la página principal o "home".

    - **navbar**: Barra de navegación para usuarios autenticados.
    - **navbar-guest**: Barra de navegación para usuarios invitados o no autenticados.
    - **not-found**: Componente que muestra una página de error 404.
- **order-card/**: Componente que muestra un resumen de una orden en formato de tarjeta.
- **products/**
    - **add-product**: Componente para agregar productos.
    - **edit-product**: Componente para editar productos.
    - **view-products**: Vista o componente para mostrar productos disponibles.
- **carousel-related components**
    - **card-carrousel**: Carrusel de tarjetas.
    - **category-carrousel**: Carrusel de categorías.
    - **image-carrousel**: Carrusel de imágenes

Servicios (services/)

- **admin.service.ts**: Maneja la lógica de negocio y las interacciones relacionadas con la administración del sistema, como la gestión de usuarios, configuración de la tienda y otras funcionalidades administrativas.
- **email.service.ts**: Gestiona el envío y verificación de correos electrónicos, como el envío de correos de verificación de cuenta, recuperación de contraseña, etc.
- **image.service.ts**: Gestiona la subida y recuperación de imágenes desde el servidor, relacionado con la administración de imágenes de productos, usuarios, o logos de la empresa.

- **local-storage.service.ts**: Maneja el almacenamiento local del navegador, probablemente para guardar tokens de sesión o información temporal del usuario entre sesiones de la aplicación.
- **product.service.ts**: Maneja la lógica de negocio relacionada con los productos, como la creación, edición, eliminación y listado de productos.
- **window-ref.service.ts**: Es utilizado para obtener referencias directas a la ventana o al DOM en Angular, lo que puede ser útil para realizar tareas que requieren acceso directo al navegador (como eventos o interacciones específicas del entorno del navegador).

Vistas (Rutas)

A partir del archivo de rutas (app-routing.module.ts), se puede observar que tienes varias rutas organizadas para diferentes vistas. A continuación te muestro un resumen de las vistas que manejas:

- **Rutas principales:**
  - /home: Componente HomeComponent, que representa la página principal.
  - /login: Componente LoginComponent para la autenticación de usuarios.
  - /store-config: Componente StoreConfigComponent para la configuración de la tienda.
  - /store-billing: Componente StoreBillingComponent para la facturación.
  - /company-settings/:name: Componente SettingsFormComponent para la configuración de la empresa.
  - /dashboard: Componente DashboardComponent, vista principal para los administradores.
  - /orders: Componente OrdersComponent, lista o gestión de órdenes.

- /purchases: Componente PurchasesComponent para visualizar las compras.
- /account/:username: Componente MyAccountComponent, vista de la cuenta de usuario.
- /shop/cart/:username: Componente ShopCartComponent, carrito de compras del usuario.
- /forgot-password: Componente ForgotPasswordComponent, para recuperar la contraseña.
- /verify-email/:token/:email: Componente VerifyEmailComponent para la verificación de correo.
- /reset-password/:token/:email: Componente ResetPasswordComponent para restablecer la contraseña.
- /add-helper: Componente AddHelperComponent, para agregar ayudantes en la sección de administración.

- **Rutas de edición de perfil:**
  - /edit/profile: Componente EditProfileComponent para la edición de información del perfil de usuario.
- **Rutas relacionadas con productos (products):**
  - /products/init: Componente ProductHomeComponent, página principal para la gestión de productos.
  - /products/view: Componente ViewProductsComponent para visualizar los productos.
  - /products/addProduct: Componente AddProductComponent para agregar nuevos productos.
  - /products/editProduct: Componente EditProductComponent para editar productos existentes.

- **Detalle de productos:**
    - /product-details/:id: Componente ProductDetailsComponent, que muestra los detalles de un producto específico.
- **Rutas de manejo de errores:**
    - /**: Componente NotFoundComponent, página de error 404 para rutas no encontradas.

Base de Datos (MariaDB)

Descripción General

El diagrama de la base de datos refleja las principales entidades del sistema, así como sus relaciones. El sistema sigue un modelo relacional, con varias tablas interconectadas que representan datos como productos, usuarios, roles, pedidos, y más. A continuación se detalla cada tabla y su función en el sistema.

Tablas Principales

1. user

- **Descripción:** Almacena la información de los usuarios registrados en el sistema.
- **Campos principales:**
    - idUser: Identificador único del usuario.
    - username: Nombre de usuario.
    - password: Contraseña encriptada.
    - email: Correo electrónico.

- **Relaciones:**
    - Relacionado con la tabla role a través de FK_RoleUser (clave foránea) para asignar roles a los usuarios.
    - Relacionado con user_information, user_2FA, shop_cart, y otras.

2. product

- **Descripción:** Almacena la información sobre los productos disponibles.
- **Campos principales:**
    - idProduct: Identificador único del producto.
    - name: Nombre del producto.
    - price: Precio base del producto.
    - stock: Cantidad disponible en inventario.
- **Relaciones:**
    - Relacionado con category, brand, product_has_attribute, y inventory.

3. orders

- **Descripción:** Contiene información sobre las órdenes realizadas por los usuarios.
- **Campos principales:**
    - idOrder: Identificador único de la orden.
    - total: Total de la compra.
    - delivery: Método de entrega.
    - FK_User: Clave foránea que conecta con la tabla user.
- **Relaciones:**
    - Relacionado con order_has_product para vincular productos a las órdenes.

4. inventory

- **Descripción:** Registra el inventario de productos disponibles por sucursal.
- **Campos principales:**
    - idInventory: Identificador del inventario.
    - inStock: Cantidad disponible de productos.
    - FK_Branch: Clave foránea que conecta con la tabla branch (sucursal).
    - FK_Product: Clave foránea que conecta con la tabla product.

5. branch

- **Descripción:** Almacena la información de las sucursales donde se venden los productos.
- **Campos principales:**
    - idBranch: Identificador de la sucursal.
    - name: Nombre de la sucursal.
    - location: Ubicación de la sucursal.

Tablas Relacionales

1. order_has_product

- **Descripción:** Relaciona los productos con las órdenes realizadas.
- **Campos principales:**
    - FK_Order: Clave foránea que conecta con la tabla orders.
    - FK_Product: Clave foránea que conecta con la tabla product.
    - quantity: Cantidad de productos en la orden.

2. product_has_attribute

- **Descripción:** Relaciona productos con atributos específicos (como color, tamaño, etc.).
- **Campos principales:**
    - FK_Product: Clave foránea que conecta con la tabla product.
    - FK_Attribute: Clave foránea que conecta con la tabla attribute.

3. product_has_category

- **Descripción:** Relaciona productos con categorías específicas.
- **Campos principales:**
    - FK_Product: Clave foránea que conecta con la tabla product.
    - FK_Category: Clave foránea que conecta con la tabla category.

Tablas Auxiliares

1. role

- **Descripción:** Define los roles de los usuarios en el sistema.
- **Campos principales:**
    - idRole: Identificador único del rol.
    - name: Nombre del rol (por ejemplo, administrador, cliente).

2. attribute

- **Descripción:** Almacena los atributos disponibles para los productos.
- **Campos principales:**
    - idAttribute: Identificador único del atributo.
    - name: Nombre del atributo (por ejemplo, color, tamaño).

3. brand

- **Descripción:** Almacena las marcas de los productos.
- **Campos principales:**
    - idBrand: Identificador único de la marca.
    - name: Nombre de la marca.

4. category

- **Descripción:** Almacena las categorías de productos.
- **Campos principales:**
    - idCategory: Identificador único de la categoría.
    - name: Nombre de la categoría.

Relaciones Clave

- **user - role:** Un usuario puede tener un rol en el sistema, como administrador o cliente.
- **orders - user:** Las órdenes están asociadas con usuarios.
- **product - inventory:** Los productos están almacenados en inventarios que pertenecen a sucursales específicas.
- **product - category:** Un producto puede estar asociado con una o más categorías.
- **orders - order_has_product:** Cada orden puede contener múltiples productos

Ejecución

- **Frontend**:

  bash

  ng serve

- **Backend**:

  bash

  pm2 restart ecosystem.config.js --env production

- **Base de Datos**: Asegurarse de que el servidor MariaDB esté corriendo:

  bash

  sudo systemctl start mariadb

Créditos

Este documento ha sido elaborado con el propósito de proporcionar una guía técnica detallada para la implementación y uso del software de E-commerce desarrollado por Error Encoders. El contenido de este manual abarca la configuración del sistema, las especificaciones técnicas, la arquitectura del software, los procesos de instalación, mantenimiento y solución de problemas, así como las mejores prácticas para optimizar su desempeño.

Elaborado por los miembros del equipo de desarrollo de Error Encoders:

- 201930693 - Leví Isaac Hernández Sapón - Estudiante Ing. en Sistemas
- 201930699 - Erick Daniel Morales Xicará - Estudiante Ing. en Sistemas
- 201832069 - Marcos Andrés Aguare Bravo - Estudiante Ing. en Sistemas
- 201830498 - Oscar Antonio de León Urizar- Estudiante Ing. en Sistemas

Agradecemos su confianza en nuestro equipo para la implementación de esta solución de E-commerce. Para cualquier consulta técnica o soporte adicional, no dude en ponerse en contacto con nuestro equipo de atención técnica.