



UNIVERSIDAD ESTATAL DE MILAGRO

**FACULTAD DE CIENCIAS E INGENIERÍA CARRERA DE INGENIERÍA  
DE SOFTWARE**

**TEMA:**

TRABAJO PRACTICO DE INVESTIGACION EN GENEXUS

**AUTORES:**

IVAN DARIO SUAREZ TOAPANTA

EZEQUIEL CRISTOBAL JARAMILLO YEPEZ

CHRISTIAN STALIN MONCAYO NARANJO

MAITE NAOMI MERA FIGUEROA

HAROLD JOSUEA SANTANA CORREA

**ASIGNATURA:**

ARQUITECTURA Y DISEÑO DE SOFTWARE

**DOCENTE:**

MGS. RICAUTER MOISES LOPEZ BERMUDEZ

**FECHA DE ENTREGA:**

10/11/2024

**PERIODO:**

Abril 2024 a Agosto 2025

**MILAGRO-ECUADOR**

## Introducción

En el trabajo en equipo, la colaboración y el compromiso son esenciales para lograr los objetivos establecidos. En este caso, el grupo se organizó de manera eficiente, distribuyendo las tareas entre sus miembros para optimizar la productividad. Iván Darío Suárez Toapanta y Ezequiel Cristóbal Jaramillo Yepez se encargaron de la elaboración de los diagramas y de realizar las transacciones necesarias, mientras que Christian Stalin Moncayo Naranjo, Maite Naomi Mera Figueroa y Harold Josué Santana Correa encargó de coordinar y supervisar el progreso del equipo. Esta división de responsabilidades permitió que cada miembro del equipo aportara de manera significativa, asegurando el cumplimiento de los plazos y manteniendo la calidad del trabajo. La participación activa de todos los integrantes del grupo resultó en un proceso fluido y sin inconvenientes, lo que refleja el valor de la cooperación en proyectos colaborativos.

## Caso de Estudio a resolver

Se requiere desarrollar una aplicación Web para la gestión de una tienda en línea.

La información obtenida del análisis es la siguiente:

### 1 País (Country)

Cada país se registrará con un identificador único, su nombre y su bandera.

El sistema deberá controlar que no se repita el nombre del país (que no se puedan ingresar dos países con el mismo nombre). Indicar un error si un usuario intenta dejar vacío el nombre del país.

### 2 Categoría (Category)

Existen diferentes categorías que serán asociadas a los productos comercializados (Tecnología, Entretenimiento, Ropa, Joyería, etc.). Cada Categoría se registrará con un identificador único y su nombre. El sistema deberá controlar que no se repita el nombre de la categoría (que no se puedan ingresar dos categorías con el mismo nombre). También deberá desplegarse un error si un usuario intenta dejar vacío el nombre de la categoría.

### 3 Cliente (Customer)

Cada cliente se registrará con un identificador único, su nombre, el país donde vive, la dirección donde serán enviados los productos, un email y su teléfono de contacto.

Deberá desplegarse un error si el usuario deja vacío el nombre del cliente o su dirección, y se deberá desplegar un mensaje de advertencia si no se ingresa el teléfono de contacto.

#### **4 Producto (Product)**

Para cada Producto se deben registrar un identificador único, su nombre (no se puede tener dentro de la tienda dos productos que se llamen igual), una breve descripción del producto, su precio (deberá aceptar dos decimales), una imagen del producto, el país donde fue manufacturado el producto, y la categoría a la que pertenece.

El sistema no debe permitir ingresar un producto sin nombre y sin precio.

#### **5 Carrito de compra (Shopping Cart, factura en línea)**

El cliente ingresa a la tienda a comprar, agrega los productos a un carrito de compra. Este carrito tendrá un identificador único, la fecha (debe sugerir la fecha del día), los datos del cliente que realiza la compra (nombre, país y dirección de envío) y el costo total de la compra. Debemos REGISTRAR EL PAIS DONDE SE REALIZA LA FACTURA.

Se detalla por cada producto adquirido, la correspondiente cantidad de unidades, el precio unitario y el costo total por producto.

Los productos de categoría Joyería tienen un recargo del 5% por costo de envío, mientras que los productos de categoría Entretenimiento tienen un 10% de descuento.

En el carrito de compra se debe registrar también la fecha de entrega, que será 5 días después de la fecha de registro de la compra.

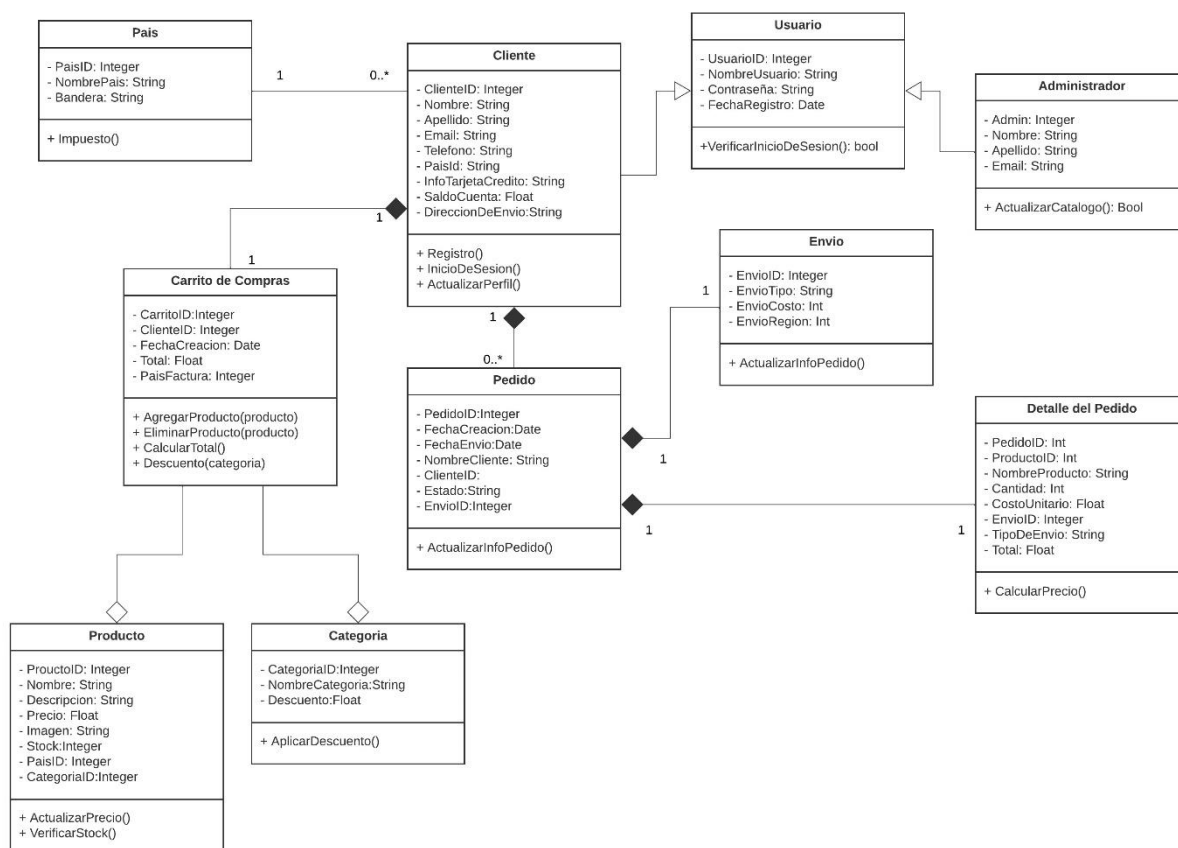
El sistema deberá controlar que no se puedan eliminar carritos de compra registrados en el día de la fecha actual.

### **PARTE 1 diseño Orientado a Objetos tradicional con UML**

**Trabajo Práctico: Diseño de Software y Arquitectura Orientada a Objetos para un Carrito de Compra**

## Requerimientos Específicos:

### 1. Diagrama de Clases:



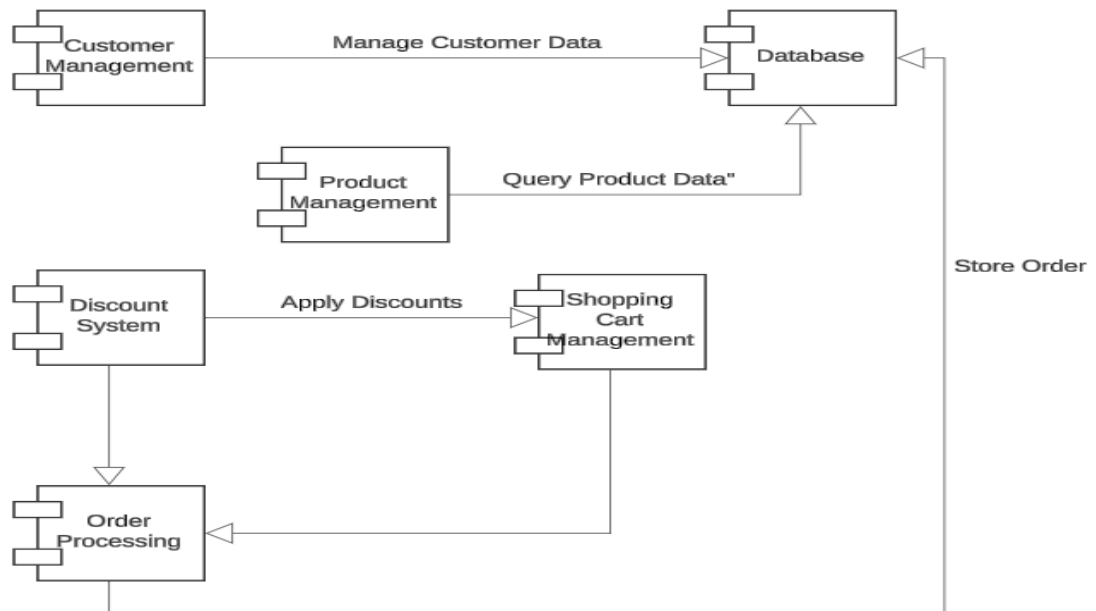
#### • Justificación:

- **País y Cliente:** Un país puede tener muchos clientes (0..\*), pero un cliente solo puede pertenecer a un país (1). Esto significa que al registrar un nuevo cliente, se debe asociar a un país específico.
- **Cliente y Carrito de Compras:** Un cliente puede tener muchos carritos de compras (0..\*), pero un carrito de compras solo puede pertenecer a un cliente (1). Esto es lógico, ya que cada cliente puede realizar múltiples compras a lo largo del tiempo.
- **Carrito de Compras y Pedido:** Un carrito de compras puede generar un pedido (1), pero un pedido solo puede estar asociado a un carrito de compras (1). Es decir, al finalizar una compra, se crea un pedido a partir del carrito de compras actual del cliente.
- **Pedido y Detalle del Pedido:** Un pedido puede tener muchos detalles de pedido (1..\*), pero un detalle de pedido solo puede pertenecer a un pedido (1).

(1). Cada detalle del pedido representa un producto específico dentro de un pedido, con su cantidad, precio unitario y otros datos relevantes.

- **Producto y Categoría:** Un producto solo puede pertenecer a una categoría (1), pero una categoría puede tener muchos productos (1..\*). Esto permite organizar los productos en diferentes categorías para facilitar la búsqueda y la gestión.
- **Usuario y Cliente:** Un usuario puede ser un cliente (1), pero un cliente solo puede ser un usuario (1). Esto indica que para realizar una compra, el usuario debe tener una cuenta de cliente asociada.
- **Usuario y Administrador:** Un usuario puede ser un administrador (0..1), pero un administrador siempre es un usuario (1). Los administradores tienen permisos especiales para gestionar el sistema.

## 2. Diagrama de Componentes:



### • Justificación:

#### Modularidad:

- **Independencia:** Cada componente (Customer Management, Product Management, etc.) representa una unidad de funcionalidad bien definida. Esto significa que puedes desarrollar, probar y modificar un componente de forma independiente sin afectar significativamente a otros componentes. Por ejemplo, puedes mejorar el sistema de descuentos sin tener que tocar la parte de gestión de productos.

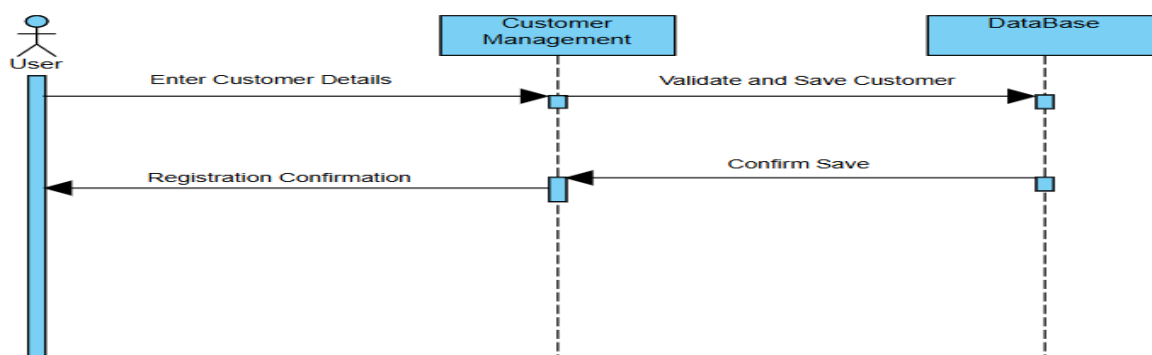
- **Reutilización:** Los componentes pueden ser reutilizados en diferentes partes de tu sistema o incluso en otros proyectos. Imagina que tienes otro sistema que necesita gestionar clientes. El componente "Customer Management" podría ser reutilizado en ese nuevo sistema sin necesidad de reescribir el código.
- **Mantenimiento:** Al aislar las funcionalidades en componentes, es más fácil localizar y solucionar problemas. Si hay un error en un componente, puedes corregirlo sin tener que revisar todo el sistema.
- **Escalabilidad:** Puedes agregar o modificar componentes de forma más sencilla, lo que facilita la evolución del sistema a medida que crecen tus necesidades. Por ejemplo, si quieres agregar un nuevo módulo de gestión de devoluciones, puedes crear un nuevo componente y conectarlo a los componentes existentes.

### Comprensión:

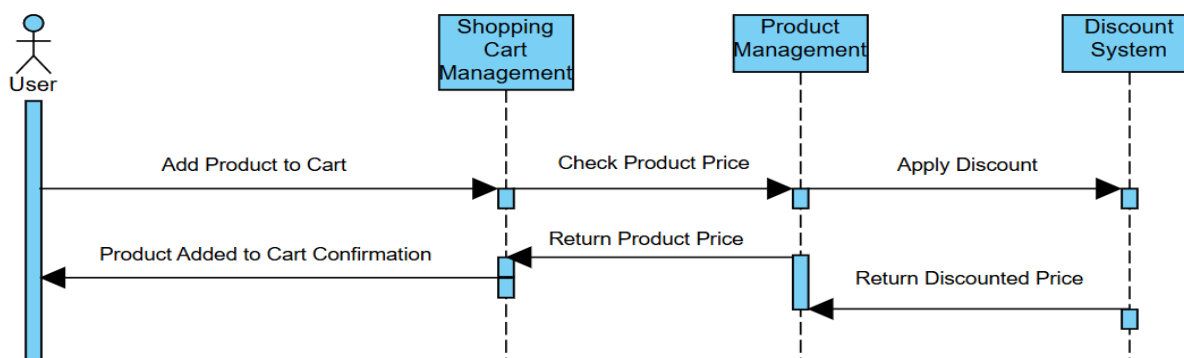
- **Abstracción:** Cada componente representa una abstracción de una parte del sistema, lo que simplifica la comprensión del sistema en su conjunto. Los desarrolladores pueden centrarse en un componente específico sin tener que entender todos los detalles de implementación.
- **Visualización:** El diagrama de componentes proporciona una vista de alto nivel del sistema, lo que facilita la comunicación entre los miembros del equipo y la documentación del sistema.
- **Onboarding:** Los nuevos miembros del equipo pueden entender rápidamente la estructura del sistema y su flujo de trabajo al analizar el diagrama.
- **Reducción de la complejidad:** Al dividir el sistema en componentes más pequeños, se reduce la complejidad general del sistema, lo que facilita su comprensión y mantenimiento.

## 3. Diagrama de Secuencia:

### Diagrama de Secuencia 1



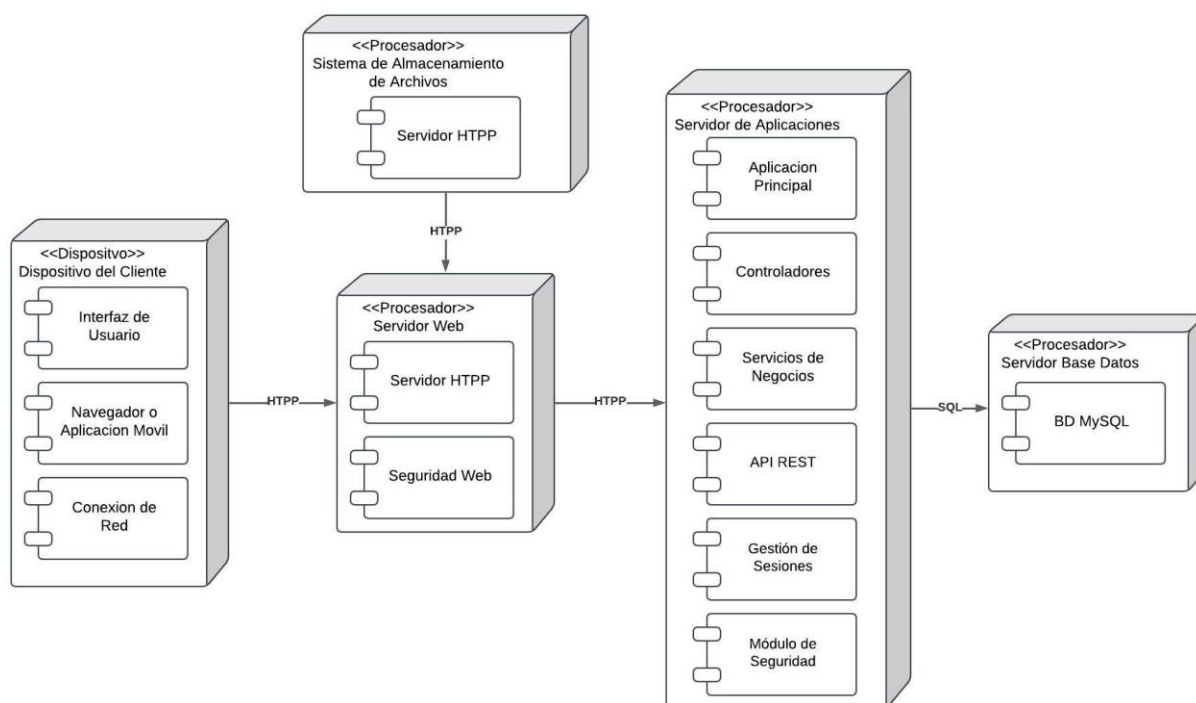
## Diagrama de Secuencia 2



### • Justificación:

- **Modularidad y Encapsulación:** Cada componente (Usuario, Customer Management, Database, Product Catalog, Discount Engine, Shopping Cart) tiene una responsabilidad bien definida, lo que promueve la modularidad y encapsulación. Esto significa que cada componente se encarga de una tarea específica, lo que facilita el desarrollo, mantenimiento y comprensión del sistema.
- **Flujo Claro y Lógico:** La secuencia de mensajes entre los componentes es clara y lógica, lo que facilita el seguimiento del flujo de datos y la identificación de posibles puntos de error. Por ejemplo, al agregar un producto al carrito, se verifica la disponibilidad con el Product Catalog antes de calcular los descuentos.
- **Reutilización de Componentes:** Componentes como el "Customer Management" y el "Product Catalog" pueden ser reutilizados en diferentes partes del sistema, lo que reduce la duplicación de código y mejora la eficiencia.
- **Gestión de Descuentos Eficiente:** El componente "Discount Engine" se encarga específicamente de calcular los descuentos, lo que permite aplicar reglas de descuento complejas y personalizadas sin afectar a otros componentes.
- **Verificación de Disponibilidad:** El componente "Product Catalog" verifica la disponibilidad de los productos en tiempo real, lo que garantiza que los clientes no puedan agregar productos agotados a sus carritos.
- **Personalización:** Al almacenar la información de los clientes en la base de datos, se pueden utilizar estos datos para ofrecer descuentos personalizados y recomendaciones de productos.
- **Escalabilidad:** La estructura modular del sistema facilita la adición de nuevas funcionalidades o la modificación de las existentes sin afectar a todo el sistema. Por ejemplo, se puede agregar un nuevo método de pago o un nuevo tipo de descuento sin necesidad de realizar cambios significativos en otros componentes.

#### 4. Diagrama de Despliegue:



#### • Justificación:

##### Componentes clave identificados:

- **Cliente:** El dispositivo del usuario final (computadora, teléfono móvil) que interactúa con la aplicación a través de un navegador o una aplicación móvil.
- **Servidor Web:** Maneja las solicitudes HTTP del cliente y sirve las páginas web.
- **Servidor de Aplicaciones:** Ejecuta la lógica de negocio de la aplicación, interactúa con la base de datos y proporciona servicios a través de una API REST.
- **Base de Datos:** Almacena los datos persistentes de la aplicación.
- **Sistema de Almacenamiento de Archivos:** Almacena archivos estáticos como imágenes, videos, etc.

#### Cómo esta Estructura Soporta la Confiabilidad, Disponibilidad y Seguridad

##### Confiabilidad

- **Separación de responsabilidades:** Cada componente tiene una función específica, lo que facilita el aislamiento de problemas y la identificación de la raíz de las fallas.



- **Redundancia:** La implementación de múltiples servidores (web, aplicaciones, base de datos) permite la alta disponibilidad. Si un servidor falla, otros pueden asumir la carga.
- **Balanceo de carga:** Distribuye las solicitudes entre múltiples servidores para prevenir sobrecargas y mejorar el rendimiento.
- **Copias de seguridad:** Las copias de seguridad regulares de la base de datos y los archivos aseguran que los datos puedan ser restaurados en caso de pérdida.

### Disponibilidad

- **Arquitectura cliente-servidor:** Permite que múltiples usuarios accedan a la aplicación simultáneamente.
- **Protocolo HTTP:** Es un protocolo estándar y ampliamente utilizado, lo que garantiza la interoperabilidad entre diferentes sistemas.
- **Servicios web:** La exposición de la lógica de negocio a través de una API REST permite la integración con otras aplicaciones y servicios.

### Seguridad

- **Seguridad web:** Implementa medidas para proteger la aplicación contra ataques comunes como inyección SQL, XSS, CSRF.
- **Módulo de seguridad:** Centraliza las políticas de seguridad y controla el acceso a los recursos.
- **Cifrado:** Protege los datos en tránsito y en reposo.
- **Gestión de sesiones:** Controla la autenticación y autorización de los usuarios.
- **Firewall:** Protege la red interna de ataques externos.

## 5. Evaluación de Decisiones de Diseño:

- **Mantenibilidad:** El diseño modular permite que cada componente tenga una función específica, facilitando ajustes o añadidos sin afectar el resto del sistema. Por ejemplo, al mejorar el cálculo de descuentos en el módulo de descuentos, no se requiere modificar otros módulos como el de productos, lo cual mejora la mantenibilidad y permite corregir errores con un impacto mínimo en el sistema.
- **Escalabilidad:** La arquitectura modular permite que el sistema gestione un mayor número de usuarios, productos o regiones sin afectar el rendimiento. Los componentes autónomos, como la gestión de clientes y productos, dividen los procesos, lo que facilita la adición de nuevos módulos, como uno para devoluciones o más países, sin necesidad de rediseñar los módulos existentes.

## **Diagrama de Clases: Justificación de Asociaciones**

### **1. País y Cliente:**

- Cada cliente se asocia con un país específico, asegurando que los registros mantengan la consistencia geográfica. Esto permite, por ejemplo, la personalización de la oferta de productos o la aplicación de regulaciones específicas por región.

### **2. Cliente y Carrito de Compras:**

- Cada cliente puede tener múltiples carritos, permitiendo que se acumulen registros de compras a lo largo del tiempo, pero cada carrito pertenece exclusivamente a un cliente para preservar el historial de compras y mantener la privacidad.

### **3. Carrito de Compras y Pedido:**

- Una vez que el cliente decide finalizar una compra, el carrito se convierte en un pedido. Esto permite al sistema rastrear y organizar los detalles de compra de cada cliente de manera lógica.

### **4. Pedido y Detalle del Pedido:**

- Un pedido puede contener múltiples detalles de productos, manteniendo datos como cantidad, precio, y especificaciones, facilitando la gestión de inventario y el análisis de ventas.

### **5. Producto y Categoría:**

- Esta relación permite organizar productos en categorías específicas, facilitando la navegación del usuario y mejorando la administración del inventario.

### **6. Usuario y Cliente:**

- Requiere que cada usuario esté vinculado a un cliente para realizar compras, garantizando la trazabilidad y simplificación de la autenticación.

### **7. Usuario y Administrador:**

- Los administradores tienen permisos especiales, lo que proporciona control de acceso y seguridad en la gestión del sistema.

## **Diagrama de Componentes: Justificación de Modularidad**

### **1. Independencia:**

- Cada componente, como `Customer Management` o `Product Management`, funciona de forma autónoma. Esto asegura que los desarrollos y actualizaciones en un módulo no impacten otros componentes. La independencia permite escalar cada módulo de forma aislada según sea necesario.

**2. Reutilización:**

- Componentes reutilizables como `Customer Management` podrían ser empleados en otros proyectos o en diferentes partes del mismo sistema, optimizando recursos y reduciendo la duplicación de código.

**3. Mantenimiento:**

- Al segmentar las funcionalidades en módulos, es fácil localizar problemas en componentes específicos y solucionarlos sin comprometer el sistema entero. La estructura modular permite identificar rápidamente errores y aplicar soluciones sin extender las pruebas a todas las áreas del sistema.

**4. Escalabilidad:**

- La adición de nuevos módulos o la expansión de los existentes es simple y no afecta la funcionalidad del sistema. Este diseño escalable permite que el sistema evolucione sin rediseños significativos.

**5. Comprensión y Onboarding:**

- La abstracción de componentes en un diagrama facilita la comprensión para los desarrolladores, especialmente aquellos nuevos en el equipo. Esto ayuda en el proceso de integración de nuevos miembros y simplifica la documentación y comunicación del diseño.

**Diagrama de Secuencia: Justificación de Diseño Modular y Encapsulación****1. Modularidad y Encapsulación:**

- Cada componente como `Discount Engine`, `Shopping Cart`, y `Product Catalog` tiene una responsabilidad específica y es independiente, facilitando el mantenimiento y la modificación de su lógica sin afectar el flujo general.

**2. Flujo Claro y Lógico:**

- La secuencia de mensajes entre componentes sigue un flujo natural que permite verificar la disponibilidad de productos y calcular descuentos antes de finalizar la compra, proporcionando robustez al sistema.

**3. Reutilización de Componentes:**

- Módulos reutilizables, como `Customer Management`, son eficaces para evitar redundancia y mejorar la eficiencia.

**4. Gestión de Descuentos y Verificación de Disponibilidad:**

- El sistema gestiona descuentos y disponibilidad de productos eficientemente, evitando duplicación de lógica y asegurando precisión en los cálculos.

**5. Escalabilidad:**

- La estructura modular permite la integración de nuevos métodos de pago o tipos de

descuento sin alterar el núcleo del sistema.

## Diagrama de Despliegue: Justificación de Fiabilidad, Disponibilidad y Seguridad

### 1. Confiabilidad:

- La separación de responsabilidades y redundancia asegura que, en caso de falla de un servidor, otros puedan cubrir la carga, manteniendo el sistema operativo.

### 2. Disponibilidad:

- La arquitectura cliente-servidor soporta múltiples conexiones, garantizando el acceso de varios usuarios sin interrupciones. Además, el uso de APIs REST para el backend permite interoperabilidad y futuras integraciones.

### 3. Seguridad:

- La implementación de medidas de seguridad como cifrado de datos, gestión de sesiones y cortafuegos garantiza que el sistema esté protegido contra vulnerabilidades y ataques externos, salvaguardando tanto los datos del cliente como la infraestructura del sistema.

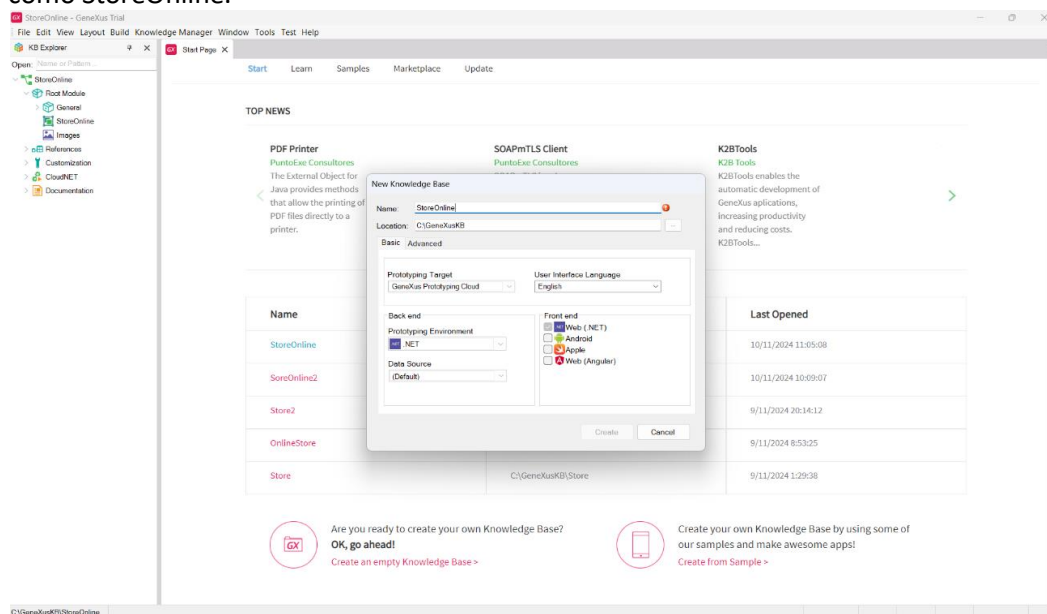
## PARTE 2 Diseño Avanzado con Genexus

### Trabajo Práctico: Diseño de Carrito de Compras en GeneXus

#### Actividades

#### 1. Introducción al Caso de Estudio

- **Primer paso:** Creamos nuestra base de conocimiento en GeneXus y la denominamos como StoreOnline.



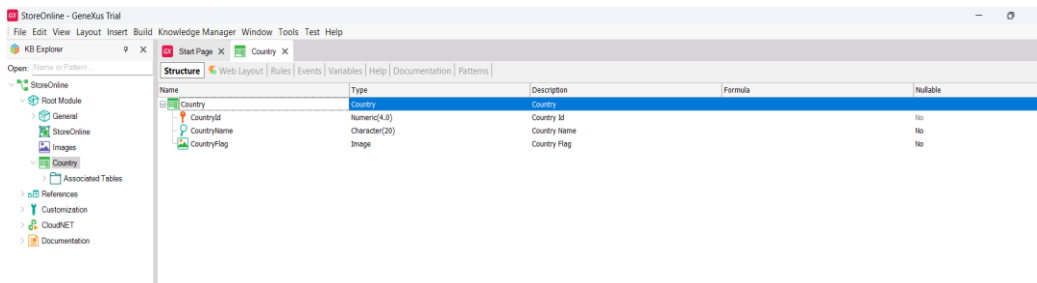
## 2. Creación de Transacciones y Configuración de Atributos

- Creamos cada transacción con los atributos identificados de caso de estudio que se nos presentó.

- **Transaction 1: Country**

- Atributos:

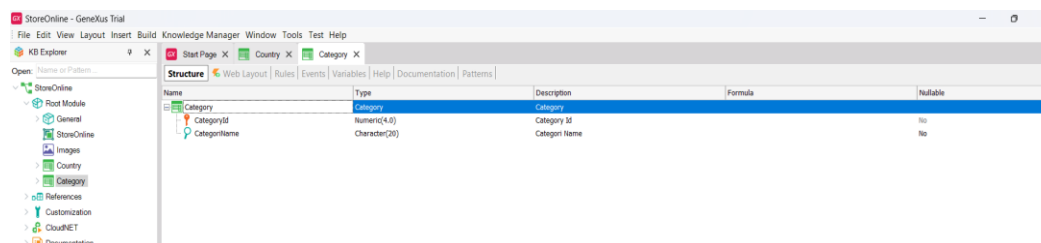
- CountryId: Tipo Numérico, AutoNumerado
    - CountryName: Tipo Carácter, longitud 20
    - CountryFlag: Tipo Imagen



- **Transaction 2: Category**

- Atributos:

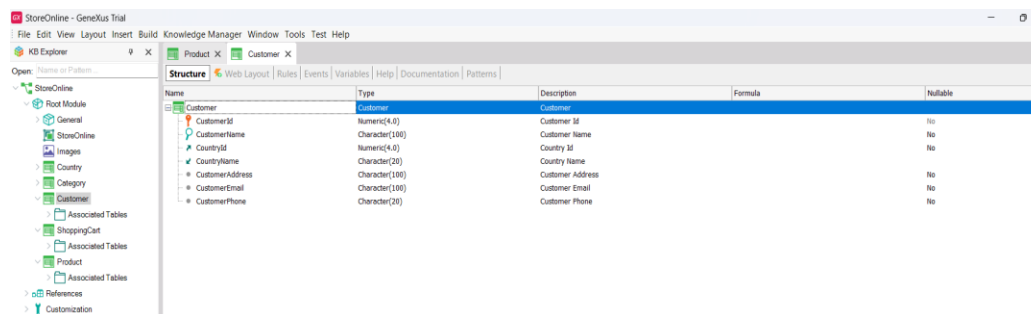
- CategoryId: Tipo Numérico, AutoNumerado
    - CategoryName: Tipo Carácter, longitud 20



- **Transaction 3: Customer**

- Atributos:

- CustomerId: Tipo Numérico, AutoNumerado
    - CustomerName: Tipo Carácter, longitud 100
    - CountryId: Referencia a Country
    - CustomerAddress: Tipo Carácter, longitud 150
    - CustomerEmail: Tipo Carácter, longitud 100
    - CustomerPhone: Tipo Carácter, longitud 20



- **Transaction 4: Product**

- Atributos:

- ProductId: Tipo Numérico, AutoNumerado
    - ProductName: Tipo Carácter, longitud 100
    - ProductPrice: Tipo Numérico, longitud 10.2
    - CountryId: Referencia a Country
    - CategoryId: Referencia a Category

The screenshot shows the 'Structure' tab in the GeneXus IDE. The 'Product' transaction is selected, and its attributes are listed in a table:

Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductId	Numeric(4,0)	Product ID		No
ProductName	Character(100)	Product Name		No
ProductPrice	Numeric(10,2)	Product Price		No
CountryId	Numeric(4,0)	Country ID		No
CountryName	Character(20)	Country Name		No
CategoryId	Numeric(4,0)	Category ID		No
CategoryName	Numeric(4,0)	Category Name		No

- **Transaction 5: ShoppingCart**

- Atributos:

- CartId: Tipo Numérico
    - CartDate: Tipo Fecha
    - CustomerId: Referencia a Customer
    - CartTotalCost: Tipo Numérico, longitud 10.2
    - DeliveryDate: Tipo Fecha

The screenshot shows the 'Structure' tab in the GeneXus IDE. The 'ShoppingCart' transaction is selected, and its attributes are listed in a table:

Name	Type	Description	Formula	Nullable
ShoppingCart	ShoppingCart	Shopping Cart		
CartId	Numeric(4,0)	Cart ID		No
CartDate	Date	Cart Date		No
CustomerId	Numeric(4,0)	Customer ID		No
CustomerName	Character(100)	Customer Name		No
CartTotalCost	Numeric(10,2)	Cart Total Cost		No
DeliveryDate	Date	Delivery Date		No

### Relaciones entre Transacciones:

A continuación, tenemos las relaciones entre las transacciones que tenemos de nuestra StoreOnline:

- **Transacción "Country" y "Product":** La transacción "Product" tiene una referencia a "Country", indicando que cada producto pertenece a un país específico.
- **Transacción "Category" y "Product":** La transacción "Product" también tiene una referencia a "Category", lo que indica que cada producto pertenece a una categoría específica.
- **Transacción "Customer" y "Country":** La transacción "Customer" tiene una referencia a "Country", indicando que cada cliente reside en un país determinado.
- **Transacción "ShoppingCart" y "Customer":** La transacción "ShoppingCart" tiene una referencia a "Customer", lo que indica que cada carrito de compras pertenece a un cliente específico.

### Como las Transacciones Representan Objetos en un Modelo de Datos Relacional:

En GeneXus, cada transacción se convierte en una tabla en la base de datos relacional. Los atributos de la transacción se traducen en columnas de la tabla correspondiente, y las referencias entre transacciones

(claves foráneas) se reflejan como relaciones entre tablas en el modelo de datos relacional. Por ejemplo, la transacción Product tiene los siguientes atributos:

- ProductId, ProductName, ProductPrice, CountryId, CategoryId

En el modelo relacional, la tabla Product tendría una estructura como la siguiente:

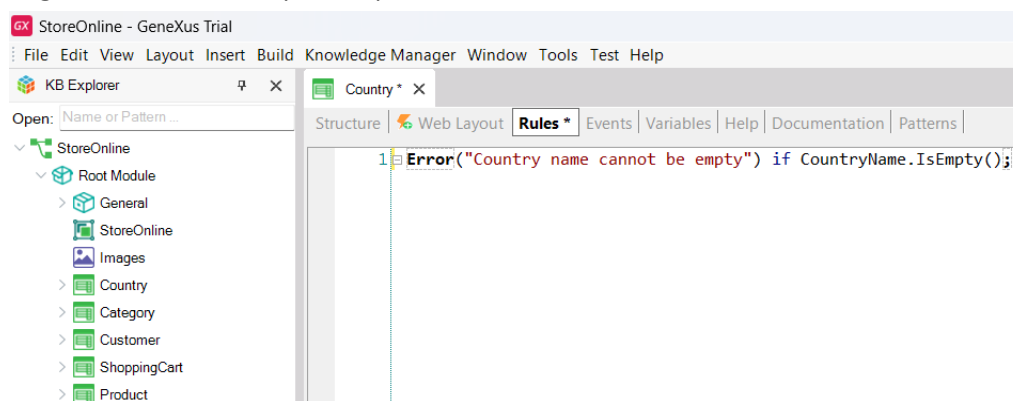
Column (Columna)	Data Type (Tipo de dato)	Foreign Key (Clave Foranea)
ProductId	INT	
ProductName	VARCHAR(100)	
ProductPrice	DECIMAL(10,2)	
CountryId	INT	Country.CountryId
CategoryId	IN	Category.CategoryId

En este caso, CountryId y CategoryId son claves foráneas que refieren a sus respectivas tablas que son: Country y Category.

### 3. Configuración de Reglas de Negocio

#### 1. Transacción "Country":

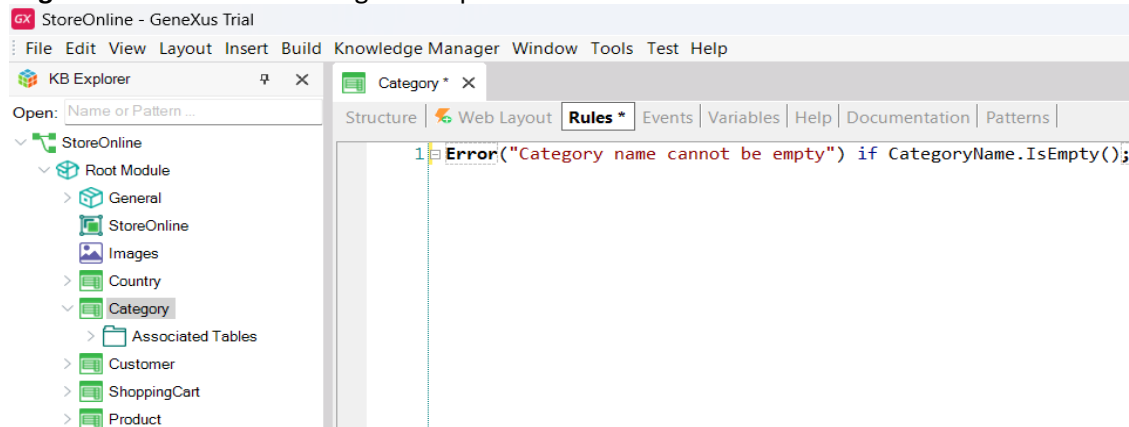
- **Regla 1:** El nombre del país no puede estar vacío.



- Esta regla garantiza que se ingrese un nombre de país válido, evitando la creación de registros sin un nombre.

#### 2. Transacción "Category":

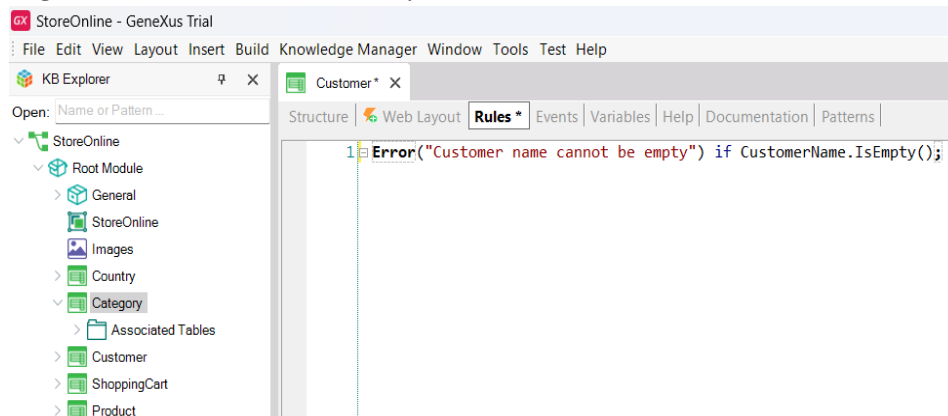
- **Regla 1:** El nombre de la categoría no puede estar vacío.



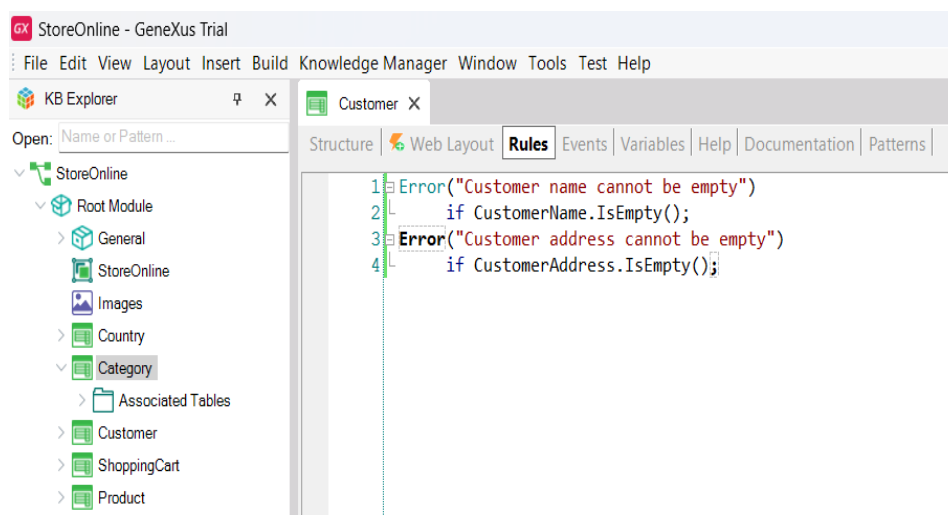
- La validación asegura que cada categoría tiene un nombre asociado, evitando la creación de categorías sin nombre.

### 3. Transacción "Customer":

- Regla 1:** El nombre del cliente no puede estar vacío.

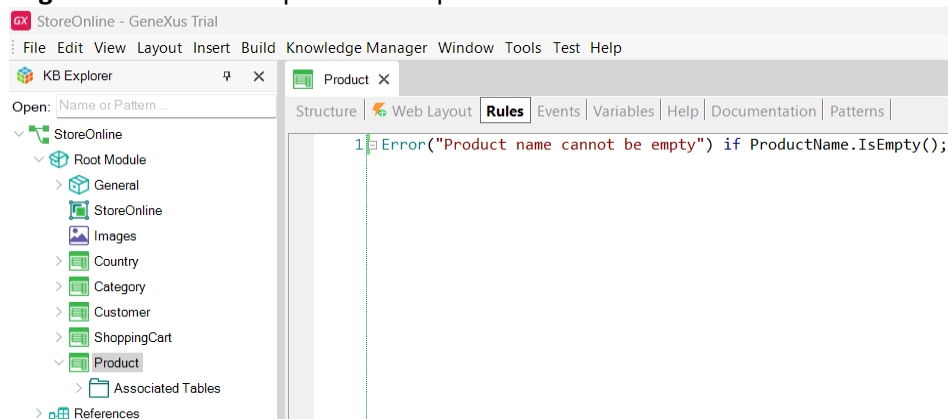


- Regla 2:** La dirección del cliente no puede estar vacía.



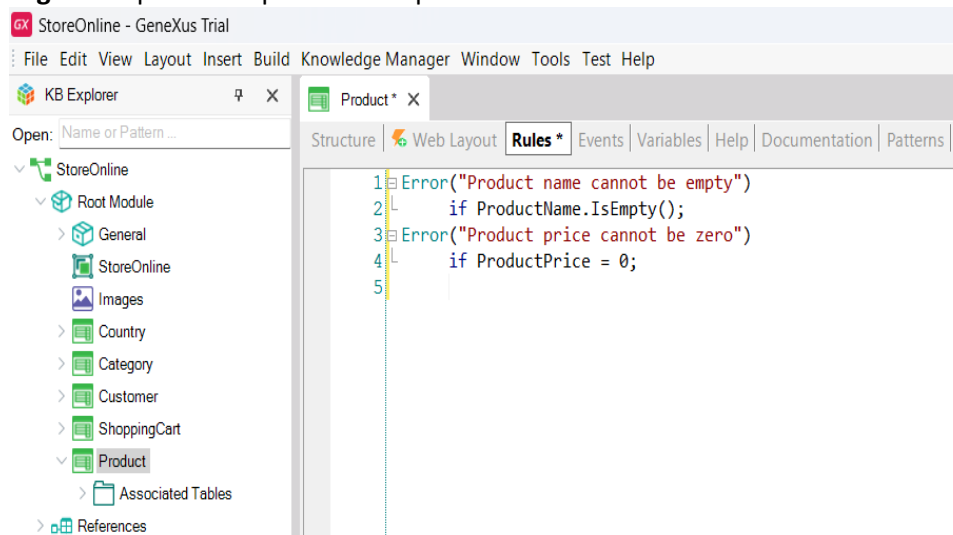
### 4. Transacción "Product":

- Regla 1:** El nombre del producto no puede estar vacío.





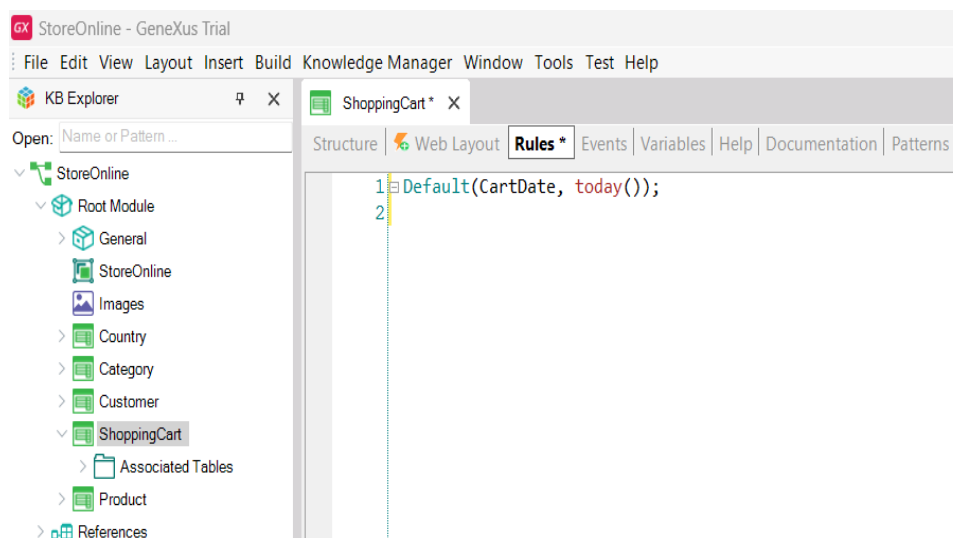
- **Regla 2:** El precio del producto no puede ser cero.



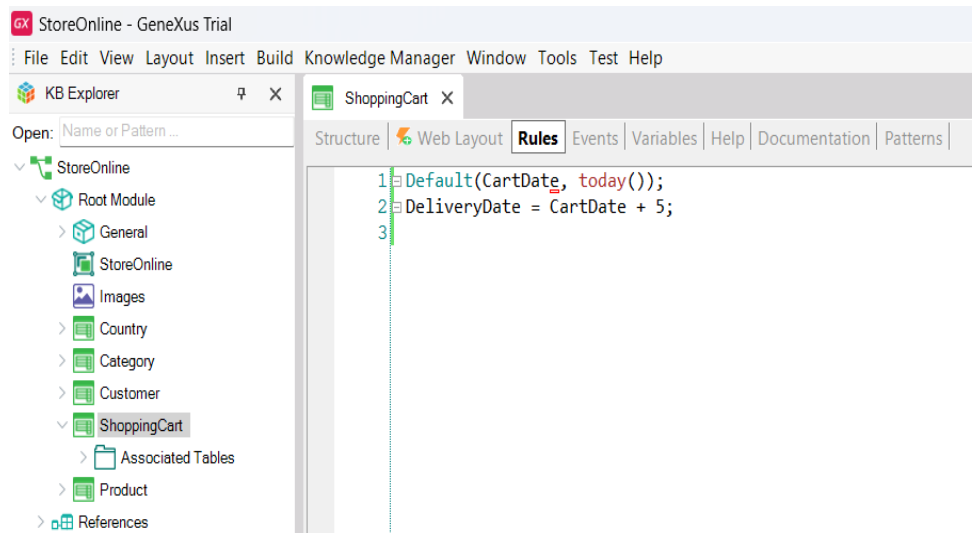
- Estas reglas aseguran que los productos tengan un nombre y un precio válido, evitando errores en el sistema de inventario.

## 5. Transacción "ShoppingCart":

- **Regla 1:** La fecha de creación del carrito debe ser la fecha actual por defecto.



- **Regla 2:** La fecha de entrega se calcula automáticamente como 5 días después de la fecha de creación del carrito.



- Estas reglas de negocio aseguran que la fecha de creación y la fecha de entrega se gestionen automáticamente, evitando errores manuales en el proceso.

#### 4. Creación de Subtipos

- En este caso de estudio no se necesita la creación de subtipos por ahora, pero identificando en algún futuro si, en la parte de customer ya que puede haber varios tipos de customer como por ejemplo CustomerVip o CustomerNormal, pero por el momento no se ve necesario la creación de subtipos.
- Instrucciones:
  - Identificar y documentar las relaciones entre transacciones
- Documentar: Foto sin genexus aun que sirva
  - Adjuntar explicaciones de programa que no sabemos como usar ni instalar

#### 5. Implementación de Fórmulas y Cálculos en GeneXus

```

1 // Cálculo de descuento para categoría "Entretenimiento"
2 if CategoryName = "Entertainment"
3   &Discount = &Subtotal * 0.10
4 else
5   &Discount = 0
6
7 // Cálculo de recargo para categoría "Joyería"
8 if CategoryName = "Jewelry"
9   &Surcharge = &Subtotal * 0.05
10 else
11   &Surcharge = 0
12
13 // Cálculo del costo total
14 &TotalCost = &Subtotal - &Discount + &Surcharge
  
```

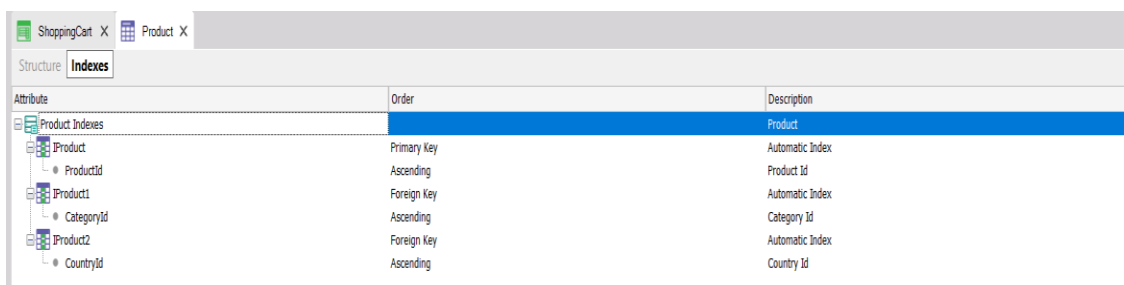
##### Explicación:

- Esta fórmula calcula el descuento del 10% en el Subtotal de un carrito de compras si el producto pertenece a la categoría "Entretenimiento".
- Si el producto pertenece a esa categoría, se calcula el 10% del Subtotal y se asigna a la variable & Discount. Si no, el descuento se establece en 0.

### Explicación:

- Similar al cálculo del descuento, aquí se calcula un recargo del 5% sobre el Subtotal si el producto es de la categoría "Joyería".
- Si el producto pertenece a esta categoría, se multiplica el Subtotal por 0.05 (5%) y se asigna el valor a la variable & Surchage. Si no, el recargo es 0.

## 6. Configuración de Índices para no permitir ingresos duplicados



Attribute	Order	Description
Product Indexes		Product
Product	Primary Key	Automatic Index
ProductId	Ascending	Product Id
Product1	Foreign Key	Automatic Index
CategoryId	Ascending	Category Id
Product2	Foreign Key	Automatic Index
CountryId	Ascending	Country Id

Los índices en GeneXus no solo permiten la creación de búsquedas rápidas y eficientes, sino que también garantizan la integridad de los datos al evitar duplicados. Configurar correctamente los índices en las transacciones es una práctica recomendada que optimiza tanto el rendimiento del sistema como la calidad de la base de datos.

## 7. Pruebas y Evaluación del Sistema

Para registrar nuevos clientes y asociarlos a países, el sistema debe asignar correctamente el país, evitando valores nulos o duplicados en campos clave como el `CountryId`, y asegurar que las transacciones de cliente y país funcionen correctamente. En cuanto al carrito, se deben aplicar los descuentos correctos según la categoría del producto y el país del cliente. Las búsquedas de productos por categoría deben validar que los índices y las relaciones entre productos y categorías están bien configurados.

### Optimización con GeneXus:

Las transacciones en GeneXus organizan el sistema eficientemente al definir relaciones entre entidades (clientes, productos, países), mejorando la integración entre componentes. Las configuraciones automáticas para subtotales, descuentos e impuestos agilizan las operaciones y reducen errores, mejorando la experiencia del usuario. Además, los índices optimizan consultas y mantienen la integridad de datos. En conclusión, las transacciones y configuraciones en GeneXus optimizan la estructura y el rendimiento, garantizando una experiencia de usuario eficiente y sin errores.