BASE DE DATOS NAILA´S BEAUTY

1. Introducción y Descripción General

Propósito de la Base de Datos:

La base de datos de Naila's Beauty tiene como objetivo gestionar y almacenar la información relacionada con los productos, clientes y ventas de la tienda de belleza.

Alcance del Proyecto:

Este proyecto abarca el diseño, implementación y mantenimiento de la base de datos que soporta las operaciones diarias de Naila's Beauty, incluyendo la gestión de inventarios, procesamiento de ventas, seguimiento de clientes y reportes financieros.

Descripción General del Sistema y su Contexto:

El sistema está diseñado para integrarse con el software de gestión de la tienda, facilitando el acceso a datos precisos y actualizados. Proporciona interfaces para usuarios internos (administradores) y externos (clientes) mediante aplicaciones web y móviles.

2. Requisitos Funcionales y No Funcionales

Requisitos Funcionales:

- Registro y gestión de productos, categorías y subcategorías.

- Gestión de inventarios.

- Control de ventas.

- Registro y seguimiento de usuarios.

Requisitos No Funcionales:

- Rendimiento: La base de datos debe manejar hasta 1000 transacciones por minuto.

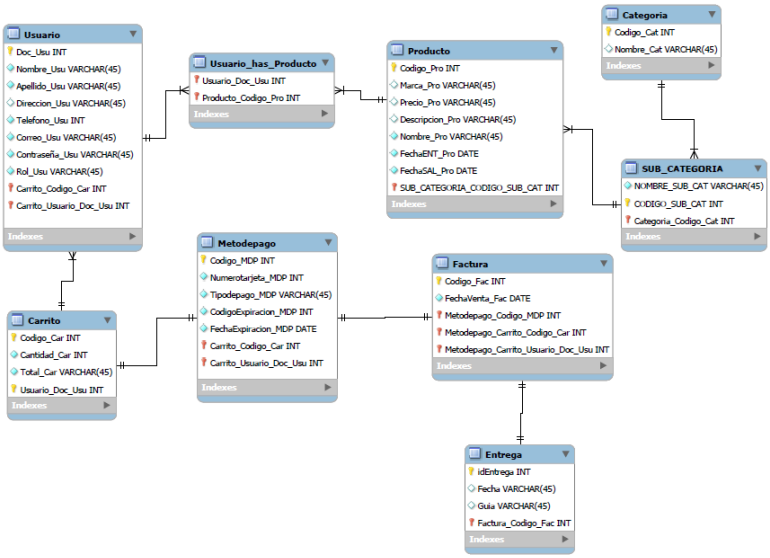
- Seguridad: Protección de datos sensibles mediante encriptación y políticas de acceso.

- Disponibilidad: Debe estar disponible el 99.9% del tiempo.

- Escalabilidad: Debe permitir la expansión conforme crece el negocio.

3. Modelo de Datos

Diagrama de Entidad-Relación (ERD):



Descripción de Entidades y Relaciones:

-**Productos:** Información sobre los productos (Codigo_Pro,Marca_Pro,Precio_pro,Descripcion_Pro,Nombre_Pro,FechaENT_Pro,Fecha Sal_Pro,).

Categoría: Información sobre las categorías (Codigo_Cat, Nombre_Cat).

Subcategorías: Información sobre las subcategorías (Nombre_Sub_Cat, Codigo_Sub_CAT).

-Usuarios: Detalles de los Usuario (Doc_Usu,Nombre_Usu,Apellido_Usu,Direccion_Usu, Telefono_Usu,correo_Usu,Contraseña_Usu,Rol_Usu).

-Carrito: Detalles de los productos seleccionados (Cantidad_Car,Total_Car).

-Método de pago: Metodologías de pago(Numerotarjeta_MDP, Tipodepago_MDP, Codigoexpiracion_MDP, FechaExpiracion_MDP).

-Factura: Información del Pedido (FechaVenta_Fac).

4. Esquema de la Base de Datos

Tablas y sus Descripciones:

- Productos:

-Codigo_Pro (PK)

-Marca_Pro

-Precio_pro

-Descripcion_Pro

-Nombre_Pro

-FechaENT_Pro

-FechaSal_Pro

- CATEGORIAS:

-Codigo_Cat (PK)

-Nombre_Cat

-SUBCATEGORIAS:

-Nombre_Sub_Cat (PK)

-Codigo_Sub_CAT

-USUARIOS:

Doc_Usu (PK)

-Nombre_Usu

-Apellido_Usu

-Direccion_Usu

-Telefono_Usu

-correo_Usu

-Contraseña_Usu

-Rol_Usu

-CARRITO:

-Cantidad_Car (PK)

-Total_Car

-Metododepago

-Numerotarjeta_MDP (PK)

-Tipodepago_MDP

-Codigoexpiracion_MDP

-FechaExpiracion_MDP

Restricciones:

- PK: Claves primarias.

- FK: Claves foráneas.

- UNIQ: Restricciones de unicidad.

- CHECK: Restricciones de validación.

Relaciones entre Tablas:

- Doc_Usu en la tabla Ventas es una FK que referencia ID en la tabla Usuarios.

5. Índices y Claves

Índices Creados y su Propósito:

- Índice en la columna nombre de la tabla Productos para acelerar las búsquedas.

Claves Primarias y Foráneas:

PK en cada tabla.

FK en Ventas referenciando Clientes.

6. Vistas

Descripción de Vistas:

- Vista_Productos_Disponibles: Muestra productos con cantidad en stock mayor a cero.

Propósito y Uso de Cada Vista:

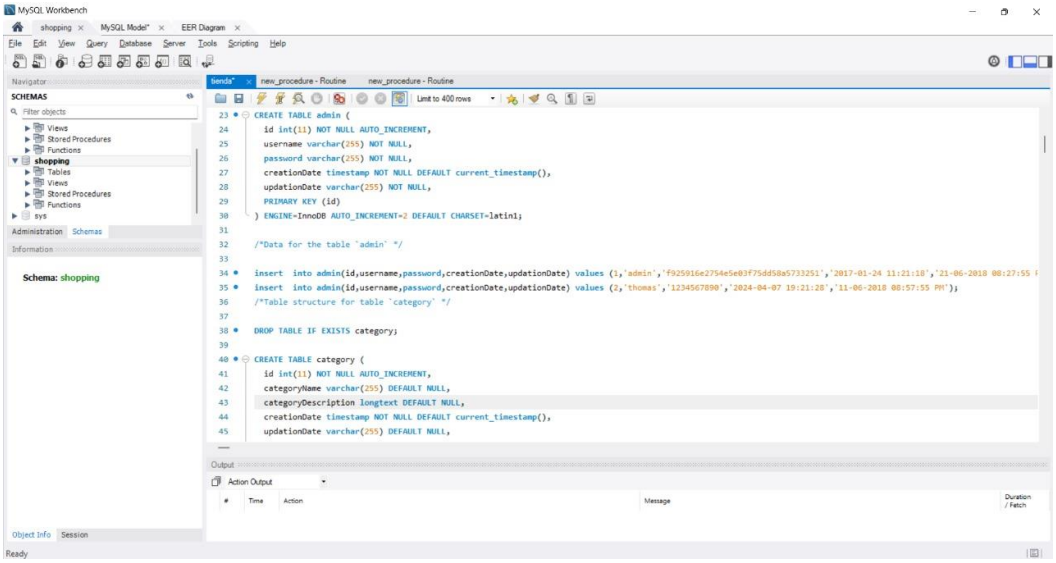
- Facilita la consulta de productos disponibles para la venta.

7. Procedimientos Almacenados y Funciones

Descripción y Propósito de Procedimientos Almacenados:

- Factura: Procedimiento para registrar una nueva venta.

Código de los Procedimientos y Funciones:



8. Triggers

Codigo:

/TRIGGER USUARIO/

create table datos_anteriores(id int auto_increment, name varchar(255),email varchar(255), contactno bigint,

password varchar(255), primary key (id),updationDate varchar(255));

drop table datos_anteriores;

select * from datos_anteriores;

select * from users;

delimiter //

create trigger before_usuario_update before update on users for each row begin insert into datos_anteriores(name,email,contactno,password,updationDate)

values (old.name,old.email,old.contactno,old.password,old.updationDate); end //delimiter

update users set email='felipesierrayy1420@gmarril.com' where name='Thomas';

drop trigger if exists before_usuario_update

/TRIGGER PRODUCTO/

create table update_producto(id int auto_increment,productName varchar(255), productCompany varchar(255),

productPrice int, productPriceBeforeDiscount int,category int, primary key (id));

drop table update_producto;

```
delimiter //

create trigger before_producto_update before update on products for each row begin
insert into update_producto(productName, productCompany, productPrice,
productPriceBeforeDiscount, category)

values
(old.productName,old.productCompany,old.productPrice,old.productPriceBeforeDiscount,o
ld.shippingCharge); end //delimiter ;

update products set productCompany='0987' where productPrice='20000';

drop trigger if exists  before_producto_update;

select * from update_producto;

select * from products;

 /TRIGGER CATEGORIA/

 create table datos_categoria( id int auto_increment,categoryName varchar(255),
updationDate varchar(255),primary key (id));

drop table  datos_categoria;

delimiter //

create trigger before_category_update before update on category for each row begin
insert into datos_categoria(categoryName, updationDate)

values (old.categoryName,old.updationDate); end //delimiter ;

drop trigger if exists  before_category_update;

update category set updationDate='25-03-2024 04:30:17 AM' where
categoryName='ACCESORIOS';

select * from datos_categoria;

select * from category;
```

9. Consultas Frecuentes

Consultas SQL Comunes y su Propósito:

- Consultar productos por categoría.

```
SELECT p.id, p.productName, p.productPrice, c.categoryName

FROM products p

JOIN category c ON p.category = c.id

WHERE c.categoryName = 'Electrónica' AND p.subCategory = 2; -- Suponiendo que 2 es
el ID de una subcategoría
```

- Obtener historial de compras de un cliente.

```
SELECT
  o.id AS order_id, o.orderDate, o.totalAmount, oi.product_id, p.productName, oi.quantity, oi.price
FROM
  orders o
JOIN
  order_items oi ON o.id = oi.order_id
JOIN
  products p ON oi.product_id = p.id
WHERE
  o.customer_id = 1;  -- Reemplaza 1 con el ID del cliente específico
```

10. Seguridad

Usuarios y Roles:

Admin: Acceso total.

Cliente: Acceso a sus datos y compras.

Permisos y Políticas de Acceso:

- Permisos específicos para cada rol.

```
GRANT SELECT, INSERT ON my_database.sales TO 'SalesManager';
GRANT UPDATE (price, quantity) ON my_database.sales TO 'SalesManager';
GRANT DELETE ON my_database.sales TO 'SalesManager';
```

Medidas de Seguridad Implementadas:

- Encriptación de datos sensibles.
- Encriptar y desencriptar usando AES

```
SET @key = 'mi_clave_secreta';

SELECT AES_ENCRYPT('Texto a encriptar', @key) AS texto_encriptado;

SELECT AES_DECRYPT(texto_encriptado, @key) AS texto_desencriptado;
```

- Autenticación de usuarios.

-- Crear un usuario con un nombre y una contraseña segura

CREATE USER 'mi_usuario'@'localhost' IDENTIFIED BY 'ContraseñaSegura2024!';

-- Asignar privilegios de lectura y escritura en una base de datos

GRANT SELECT, INSERT, UPDATE, DELETE ON mi_base_datos.* TO 'mi_usuario'@'localhost';

-- Aplicar los cambios

FLUSH PRIVILEGES;

11. Mantenimiento y Operaciones

Estrategias de Respaldo y Recuperación:

- Respaldos diarios y semanales.

Tareas de Mantenimiento Programadas:

- Limpieza de datos antiguos.

- Optimización de índices.

Monitoreo y Herramientas Utilizadas:

- Monitoreo de rendimiento con herramientas como Nagios.

Instalar Plugins de Nagios para MySQL: Nagios no viene con plugins para MySQL por defecto, pero puedes usar plugins de terceros como check_mysql o check_mysql_health para monitorear la base de datos MySQL.

- **check_mysql**: Un plugin básico para verificar la conexión y la disponibilidad de MySQL.
- **check_mysql_health**: Un plugin más avanzado que ofrece una variedad de chequeos sobre la salud del servidor MySQL, como el uso de CPU, memoria, y métricas específicas de MySQL.

12. Plan de Migración

Pasos para la Migración de Datos:

- Exportación de datos actuales.

- Transformación de datos según el nuevo esquema.

- Importación a la nueva base de datos.

Consideraciones y Precauciones:

- Verificación de la integridad de los datos después de la migración.

13. Anexos y Recursos Adicionales

Glosario de Términos:

- (Incluye términos técnicos y sus definiciones).

Referencias a Documentación Adicional:

- Manuales de usuario.

- Documentación técnica.

Información de Contacto para Soporte:

- Correo electrónico y teléfono del soporte técnico.

14. Historial de Cambios

Registro de Modificaciones en la Base de Datos:

- (Incluye un registro detallado de los cambios realizados).

Ejemplo de Trigger para INSERT en la Tabla products:

```
DELIMITER //

CREATE TRIGGER after_product_insert
AFTER INSERT ON products
FOR EACH ROW
BEGIN
    INSERT INTO audit_log (table_name, operation_type, old_data, new_data, user, justification, impact_expected)
    VALUES ('products', 'INSERT',
            'N/A',  -- No hay datos antiguos en un INSERT
```

```
        CONCAT('ID: ', NEW.id, ', Name: ', NEW.name, ', Price: ', NEW.price),
        USER(),  -- Captura el usuario actual
        'Añadido nuevo producto al inventario',  -- Justificación del cambio
        'Añadido para ampliar el catálogo de productos');  -- Impacto esperado
END;
//

DELIMITER ;
```

Razones y Descripciones de los Cambios:

- (Incluye justificación y descripción de cada cambio).

Uso de Triggers (Desencadenadores):

triggers para registrar cambios en las tablas. Por ejemplo, puedes crear un trigger que se active antes o después de una operación INSERT, UPDATE o DELETE y registre estos cambios en una tabla de auditoría.

Identificación del Cambio:

- **ID del Cambio:** Un identificador único para cada cambio registrado.
- **Fecha y Hora:** Fecha y hora en que se realizó el cambio.
- **Usuario:** El usuario que realizó el cambio.

Descripción del Cambio:

- **Tipo de Cambio:** INSERT, UPDATE, DELETE, etc.
- **Tabla Afectada:** La tabla en la base de datos que fue modificada.
- **Datos Anteriores (si aplica):** El estado de los datos antes del cambio (para UPDATE y DELETE).
- **Datos Nuevos (si aplica):** El estado de los datos después del cambio (para INSERT y UPDATE).

Esta estructura proporciona una guía completa para documentar la base de datos del proyecto Naila's Beauty, asegurando que todos los aspectos relevantes sean cubiertos.