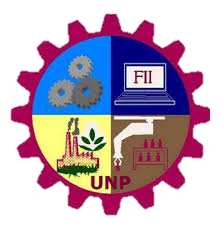
**UNIVERSIDAD NACIONAL DE PIURA**

**FACULTAD INGENIERÍA INDUSTRIAL**

**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA**

**PROCESO:**

Propuesta de sistema de base de datos para la empresa “Corazón de María”.

**CURSO:**

Base de datos.

**DOCENTE:**

Jorge Alvarado Tabacchi.

**INTEGRANTES:**

Aranda Zapata, Madeley Mariella

Huanca Flores, Segundo Elvis

Morales Rios, Marlon Josue

Pasache Pizarro, Cristian Joel

Peña Cordova, Breyner Antonio

**PIURA – PERÚ**

**2024**

ÍNDICE

[I. MARCO REFERENCIAL SOBRE LA EMPRESA 1](#_Toc160207296)

[II. REQUERIMIENTOS 3](#_Toc160207297)

[Requerimientos Funcionales 3](#_Toc160207298)

[Requerimientos No Funcionales 4](#_Toc160207299)

[III. DISEÑO DEL SISTEMA 5](#_Toc160207300)

[Reconocimiento de entidades 5](#_Toc160207301)

[Descripción de Entidades 7](#_Toc160207302)

[Modelo Relacional 18](#_Toc160207303)

[IV. Roles y Funciones de cada Integrante 19](#_Toc160207304)

[V. ANEXOS 20](#_Toc160207305)

[Creación de Tablas 20](#_Toc160207306)

[Creación de Claves Foráneas 26](#_Toc160207307)

[Creación de Restricciones 32](#_Toc160207308)

[Creación de Índices 35](#_Toc160207309)

[Índice de la tabla PersonaNatural 35](#_Toc160207310)

[Índice de la tabla Cliente 35](#_Toc160207311)

[Índice de la tabla Solicitud 35](#_Toc160207312)

[Índice de la tabla Contrato 36](#_Toc160207313)

# MARCO REFERENCIAL SOBRE LA EMPRESA

“CORAZÓN DE MARÍA” es una pequeña empresa de la ciudad de Lima que se dedica al servicio de confección de prendas de vestir y de orfebrería, tanto para clientes comunes, así como a entidades religiosas, esta empresa cuenta con empleados los cuales cumplen con roles específicos para su funcionamiento. El servicio consiste en brindar al cliente la comodidad de personalizar el diseño que crea apropiado para su estilo con las indicaciones necesarias que el mismo da.

Esta empresa cuenta con una cartera de clientes y con una página de Facebook, donde los clientes solicitan contratar los servicios de la empresa.

Por cada servicio, el encargado de ventas de la empresa entrevista al cliente en una sencilla conversación sobre su pedido mediante una llamada o de manera presencial para tomar los detalles del servicio que requiere el cliente, luego de dicha solicitud se le realizan propuestas con el tipo de materiales a realizar donde se le brinda el precio estimado.

El cliente efectúa una parte del pago una vez acordado el contrato y la parte faltante una vez culminado, así mismo la empresa emite boletas como comprobante con el fin de evitar inconvenientes con los pedidos. Cuando el cliente realiza su pedido, se le pide datos personales como dirección, número de teléfono, correo electrónico (esto a elección) y datos adicionales para poder comunicarnos con el cliente. Los pagos se realizan de manera electrónica por billetera digital, transferencias bancarias o en efectivo lo cual es más recurrente, además los clientes recurrentes tienen acceso a ciertos beneficios como descuentos o regalos de cortesía por parte de la empresa.

Corazón de María, tiene a su disposición empleados que realizan la función de confeccionar las prendas como costureros y orfebres, así mismo tiene proveedores que son los abastecedores de insumos, y una secretaria la cual se encarga de atender a los clientes.

* **¿Qué tipos de pago se aceptan?**

Billeteras digitales (Yape, Plin), transferencias (todos los bancos) y efectivo.

* **¿Se entregan comprobantes?**

Si, en nuestro caso emitimos boletas.

* **¿Qué datos necesitas para realizar el servicio?**

El tipo de materiales a utilizar los cuales los listamos según las indicaciones del cliente o los cuales le podemos recomendar según su criterio. Así mismo que nos indique sus datos como DNI, nombres y apellidos, número de celular, correo electrónico.

* **¿Se realiza un contrato?**

No, solo se entregan boletas.

* **¿Qué pasa si el cliente cancela el pedido habiendo pagado una parte del servicio o el servicio completo?**

Se le reembolsa el dinero, pero aplicando un descuento como sanción.

* **¿Los materiales suponen un gasto adicional para el cliente?**

No, estos costos adicionales ya están incluidos al momento de hacer la cotización.

# II. REQUERIMIENTOS

## Requerimientos Funcionales

Son las especificaciones que describen las funciones y tareas que el sistema debe ser capaz de realizar. Estos requisitos se centran en el "qué" debe hacer el sistema y generalmente se expresan en forma de casos de uso, escenarios o descripciones detalladas de las funcionalidades.

|  |  |  |
| --- | --- | --- |
| **REQUERIMIENTO** | **NOMBRE** | **DESCRIPCIÓN** |
| **I** | Registro de clientes | El sistema debe permitir al administrador registrar a los clientes, necesita la información necesaria como nombre, dirección, número de celular, correo electrónico, tipo de documento (RUC o DNI) y número de documento. |
| **II** | Solicitud de servicio | Los clientes deben poder solicitar un servicio de confección de prendas o de orfebrería a través de la página de Facebook de la empresa o por teléfono, requieren detalles como tipo de servicio, diseño, materiales, fecha deseada. |
| **III** | Cotización y Programación | El sistema debe permitir a los administradores generar cotizaciones para los servicios solicitados, considerando el tipo de servicio, los materiales, la mano de obra y el tiempo estimado de realización. El sistema también debe permitir a los administradores programar los servicios solicitados, teniendo en cuenta la disponibilidad de los empleados y los materiales. |
| **IV** | Pagos | El sistema debe permitir a los clientes realizar pagos por los servicios solicitados, admitiendo métodos de pago como transferencias bancarias, billeteras digitales (por ejemplo, Yape, plin) y pagos en efectivo. |
| **V** | Gestión de inventario | El sistema debe permitir a la empresa gestionar el inventario de los materiales utilizados en la confección de prendas y de orfebrería, incluyendo una lista detallada de los materiales y su cantidad. |
| **VI** | Gestión de contratos | El sistema debe permitir a la empresa gestionar los contratos, incluyendo un seguimiento del estado de los pedidos, desde la solicitud hasta la entrega. |

## Requerimientos No Funcionales

Son los requisitos que no están directamente relacionados con las funcionalidades específicas del sistema, pero que son igualmente importantes para su correcto funcionamiento. Estos pueden incluir aspectos como rendimiento, seguridad, usabilidad, escalabilidad, disponibilidad, compatibilidad, entre otros.

|  |  |  |
| --- | --- | --- |
| **REQUERIMIENTO** | **NOMBRE** | **DESCRIPCIÓN** |
| **I** | Usabilidad | El sistema debe ser fácil de usar tanto para los clientes como para los administradores de la empresa de confección. |
| **II** | Rendimiento | El sistema debe ser capaz de manejar un alto volumen de solicitudes de servicio y mantener tiempos de respuesta rápidos, especialmente durante los días de eventos y fines de semana. |
| **III** | Seguridad | El sistema debe garantizar la seguridad de la información personal y financiera de los clientes, así como cumplir con las normas de protección de datos. |
| **IV** | Integración de métodos de pago | El sistema debe admitir métodos de pago como transferencias bancarias, billeteras digitales (por ejemplo, Yape) y pagos en efectivo para acomodar las preferencias de los clientes. |
| **V** | Gestión de eventos y contratos | El sistema debe permitir a la empresa gestionar eventos y contratos especiales, incluyendo la firma de contratos y la obtención de permisos de confección cuando sea necesario. |

# III. DISEÑO DEL SISTEMA

## Reconocimiento de entidades

|  |  |
| --- | --- |
| **ENTIDAD** | **ATRIBUTOS** |
| PersonaNatural | idPersonaNatural(PK), idPersona(FK),nombre, apPat, apMat, dni(U) |
| PersonaJuridica | idPersJuridica(PK),idPersona(FK), razonSocial |
| Cliente | idCliente(PK), idCategoria(FK),idPersona(FK) |
| Persona | idPersona(PK), idTelefono(FK), email(U), dirección, ruc |
| Solicitud | idSolicitud(PK), idCliente(FK), idEmpleado(FK), fechaHora, descripcion |
| Categoría | idCategoria(PK), nombre, descripción |
| Propuesta | idPropuesta(PK) , idSolicitud(FK), fecha, hora, costo |
| Contrato | idContrato(PK) , idPropuesta(FK), idTipoContrato(FK),idEmpleado(FK), fechaHora, tiempoRealizar, descripcion , montoContrato,montoAdelanto, estado |
| TipoContrato | idTipoContrato(PK), nombre |
| Empleado | idEmpleado(PK),idCargo(FK),idPersona(FK),fechaIngreso, fechaCese, salario, estado |
|  | idPago(PK), idEmpleado(PK), monto, fecha |
| Pagos |
| Cargo | idCargo(PK), nombre |
| ContratoEmpleado | idContratoEmpleado(PK), idContrato(FK),idEmpleado(FK) |
| Anulación | idAnulacion(PK) , idContrato(FK), idEmpleado(FK), fecha, hora |
| SolicitudServicio | idSolicitudServicio(PK), idSolicitud(FK), idTipoServicio(FK) |
| TipoServicio | idTipoServicio(PK), nombre |
| Proveedor | idProveedor(PK), idPersona(FK), idTipoProveedor(FK) |
| TipoProveedor | idTipoProveedor(PK), nombre |
| ProveedorMateriaPrima | idProveedorMateriaPrima(PK), idProveedor(FK), idMateriaPrima(FK), precioMateriaPrima, fecha, cantidad, unidadMedida |
| MateriaPrima | idMateriaPrima(PK), idCategoriaMateriaPrima(FK), nombre |
| CategoriaMateriaPrima | idCategoriaMateriaPrima(PK), nombre |
|  |
| Comprobante | idComprobante(PK), idContrato(FK), idTipoComprobante(FK), nSerie, nComrpobante, fecha, hora, subTotal, igv, total |  |
| DetalleComprobanteTipoPago | idDetalleComprobanteTipoPago, idComprobante, idTipoPago, monto |  |
| TipoPago | idTipoPago(PK), nombre |  |
| TipoComprobante | idTipoComprobante(PK), nombre |  |
| DetalleComprobante | idDetalleComprobante(PK), idArticulo(FK), idComprobante(FK), cantidad, precioVenta, subtotal |  |
| Artículo | idArticulo(PK), idTipo(FK), nombre, precioUnitario |  |
| TipoArticulo | idTipo(PK), nombre |  |
| ArticuloMateriaPrima | idArticuloMateriaPrima(PK), idArticulo(FK),idMateriaPrima(FK), cantidadMateriaPrima |  |
| InventarioMateriaPrima | idInventarioMateriaPrima(PK), idMateriaPrima(FK), salidaMateria, entradaMateria |  |
| InventarioProducto | idInventarioProducto(PK), idArticulo(FK), salida, entrada |  |
| Telefono | idTelefono(PK), numeroTelefono |  |
| DetalleArticuloPropuesta | idDetalleArticulo(PK), idPropuesta(FK), idArticulo(FK), dimensiones, cantidad, precioVenta. |  |

## Descripción de Entidades

1. PersonaNatural:
   1. *Descripción:* Persona natural que requiere un servicio
   2. *Atributos:*
      1. idPersonaNatural (PK): Identificador unívoco de la persona Natural
      2. idPersona: Identificador de la persona asociada a la persona natural.
      3. nombre: Nombre de la persona natural.
      4. apPat: Apellido materno de la persona Natural
      5. apMat: Apellido paterno de la persona Natural (opcional)
      6. dni (U): Documento Nacional de Identidad
   3. *Propósito:* Almacena información personal de una persona natural que requiera solicitar un servicio.
2. PersonaJuridica:
   1. *Descripción:* Persona jurídica que requiere un servicio
   2. *Atributos:*
      1. idPersJuridica (PK): Identificador unívoco de persona Jurídica
      2. idPersona(FK): Identificador de la persona asociada a la persona juridica.
      3. razonSocial: Razón social de la persona Jurídica
   3. *Propósito:* Almacena información personal de una persona jurídica que requiera solicitar un servicio.
3. Cliente:
   1. *Descripción:* Representa a un cliente que utiliza los servicios de la empresa.
   2. *Atributos:*
      1. idCliente (PK): Identificador unívoco del cliente
      2. idCategoria (FK): Identificador de la Categoría asociada al cliente
      3. idPesona (FK): Identificador de la persona asociada al cliente
   3. *Propósito:* Almacena la información personal y de contacto de los clientes para realizar las operaciones de la empresa, como solicitar servicios y mantener un registro de los clientes recurrentes.
4. Persona
   1. Descripción: Representa todos los datos de una persona
   2. Atributos:
      1. idPersona (PK): Identificador unívoco de una persona
      2. idTelefono: Identificador unívoco de un teléfono asociado a la persona.
      3. email(U): Correo electrónico de la persona (opcional)
      4. dirección: Dirección de una persona
      5. ruc: Ruc de la persona
   3. Propósito: Registrar todos los datos de las personas que pertenecen a la empresa.
5. Solicitud:
   1. *Descripción:* Representa una solicitud para el pedido de servicio por un cliente
   2. *Atributos:*
      1. idSolicitud (PK): Identificador unívoco de la solicitud.
      2. idCliente (FK): Identificador del cliente que ha realizado la solicitud.
      3. idEmpleado(FK): Identificador del empleado quien recibe la solicitud.
      4. fechaHora: fecha y hora de la solicitud
      5. descripcion: Describe lo que el cliente solicita
   3. *Propósito:* almacena información relacionada con las solicitudes realizadas por los clientes a la empresa.
6. Categoría:
   1. *Descripción:* Representa la categoría del cliente (cliente recurrente o cliente nuevo)
   2. *Atributos:*
      1. idCategoria (PK): Identificador unívoco de la categoría
      2. nombre: Nombre de la categoría
      3. descripción: Describe el tipo de beneficios que tiene cada categoría.
   3. *Propósito:* Almacena información relacionada de la categoría que se encuentra el cliente.
7. Propuesta:
   1. *Descripción:* Representa una propuesta de la empresa al cliente.
   2. *Atributos:*
      1. idPropuesta (PK): Identificador unívoco de la propuesta
      2. idSolicitud (FK): Identificador de la solicitud asociada a la propuesta
      3. fecha: Fecha de la propuesta
      4. hora: Hora de la propuesta
      5. costo: Costo del servicio
   3. *Propósito:* Registra información detallada de las propuestas realizadas a las solicitudes realizadas por los clientes, incluyendo la referencia de la solicitud, la fecha, hora y costo.
8. Contrato:
   1. *Descripción:* Representa un contrato entre la empresa de orfebrería y costurera a un cliente.
   2. *Atributos:*
      1. idContrato (PK): Identificador unívoco del contrato.
      2. IdPropuesta (FK): Identificador de la propuesta asociada al contrato.
      3. idTipoContrato(FK): Identificador del Tipo Contrato asociado a contrato.
      4. idEmpleado(FK): Identificador del empleado asociado al contrato.
      5. fechaHora: Fecha y hora del contrato.
      6. montoTotal: describe el monto del servicio a realizar.
      7. tiempoRealizar: Tiempo aproximado que se realizará el servicio
      8. descripción: Descripción detallada del servicio a realizar.
      9. estado: estado en que se encuentra el contrato(A:activo,I:inactivo)
   3. *Propósito:* Almacena la información relacionada con los contratos realizados entre la empresa y los clientes.
9. TipoContrato
   1. *Descripción:* Representa los tipos de contrato.
   2. *Atributos:*
      1. idTipoContrato (PK): Identificador unívoco del tipo de pago.
      2. nombre: Nombre del tipo de contrato(mayorista,minorista).
   3. *Propósito:* Describir el tipo de contrato.
10. TipoServicio
    1. *Descripción:* Representa los tipos de servicios que brinda la empresa.
    2. *Atributos:*
       1. idTipoServicio(PK): Identificador unívoco del tipo de servicio.
       2. nombre: Nombre del tipo de servicio(confección,orfebrería).
    3. *Propósito:* Describir el tipo de servicio.
11. Empleado:
    1. *Descripción:* Representa a un trabajador que participa en la realización de los proyectos en la empresa.
    2. *Atributos:*
       1. idEmpleado(PK): Identificador unívoco del empleado.
       2. idCargo(FK): Identificador unívoco del cargo que posee el empleado.
       3. idPersona(FK): Identificador unívoco de la persona que se relaciona con empleado.
       4. fechaInicio: Fecha de inicio de las labores del empleado en la empresa.
       5. fechaCese: Fecha en la que el empleado dejó de trabajar en la empresa.
       6. salario(FK): salario del empleado.
       7. estado: Estado actual del empleado en la empresa.
    3. *Propósito:* Registrar qué empleados participan en los diferentes servicios que se realizan.
12. Cargo:
    1. *Descripción:* Representa el cargo que posee el empleado.
    2. *Atributos:*
       1. idCargo(PK): Identificador unívoco del cargo.
       2. nombre: nombre del cargo del empleado.
    3. *Propósito:* Registrar los cargos del empleado.
13. ContratoEmpleado:
    1. Descripción: Representa la relación entre un contrato y un empleado que participa en la realización del servicio.
    2. Atributos:
       1. idContratoEmpleado(PK): Identificador unívoco de la relación entre contrato y empleado.
       2. idContrato (FK): Identificador del contrato asociado.
       3. idEmpleado(FK): Identificador del empleado asociado.
    3. Propósito: Registrar qué empleados están asociados a cada contrato y participan en la realización de los servicios.
14. Anulación
    1. *Descripción:* Representa la Anulación de un contrato
    2. *Atributos:*
       1. idAnulacion(PK): Identificador unívoco de cada anulacion
       2. idContrato(FK): Identificador de Contrato que representa la anulación
       3. idEmpleado(FK): Identificador del empleado asociado a la anulación de algún contrato.
       4. fecha: Representa la fecha de anulación de contrato
       5. hora: Detalla la hora de anulación.
    3. *Propósito:* Registrar todas las anulaciones con todos sus datos del contrato realizado y el administrador encargado de anulación.
15. SolicitudServicio
    1. *Descripción:* Representa la solicitud del servicio que se va a realizar.
    2. *Atributos:*
       1. IdSolicitudServicio (PK): Identificador único de la solicitud de servicio
       2. idSolicitud(FK): Identificador de la solicitud.
       3. idTipoServicio(FK): Identificador del tipo de servicio solicitado
    3. *Propósito:* Registrar una petición de un cliente para recibir un servicio*.*
16. Proveedor
17. *Descripción:* Representa al proveedor que abastece a la empresa
18. *Atributos:*
    * 1. idProveedor(PK): Identificador unívoco del proveedor.
      2. idPersona: Identificador de la persona que se asocia a proveedor.
      3. idTipoProveedor: Identificador del tipo de proveedor relacionado con este proveedor.

*c. Propósito:* Registrar y gestionar la información relacionada con el proveedor.

1. TipoProveedor:
   1. *Descripción:* Representa el tipo del proveedor.
   2. *Atributos:*
      1. idTipoProveedor(PK): Identificador unívoco del tipo de proveedor.
      2. Nombre: Nombre del tipo del proveedor.
   3. *Propósito:* Registrar los datos del tipo de proveedor.
2. ProveedorMateriaPrima
3. *Descripción:* Representa el detalle del proveedor con la materia prima que brinda .
4. *Atributos:*
   * 1. idProveedorMateriaPrima(PK): Identificador unívoco del ProveedorMateriaPrima.
     2. idProveedor: identificador del proveedor que se relaciona con ProveedorMateriaPrima.
     3. idMateriaPrima: identificador de la materia prima que ha brindado el proveedor.
     4. precioMateriaPrima: el precio de la materia prima que se registra.
     5. fecha: fecha de la compra de materia prima.
     6. cantidad: cantidad de la materia prima registrada.
     7. unidadMedida: unidad de medida de la materia prima.

*c. Propósito:* Registrar y gestionar la información relacionada con el Proveedor y Materia prima que nos va a proporcionar.

1. InventarioMateriaPrima
2. *Descripción:* Representa al inventario de la materia prima.
3. *Atributos:*
   * 1. idInventarioMateriaPrima(PK): Identificador unívoco del inventario que corresponde a la materia prima.
     2. idMateriaPrima: identificador de la materia prima.
     3. salidaMateria: cantidad de la materia prima que sale del inventario
     4. entradaMateria: cantidad de la materia prima que entra del inventario.

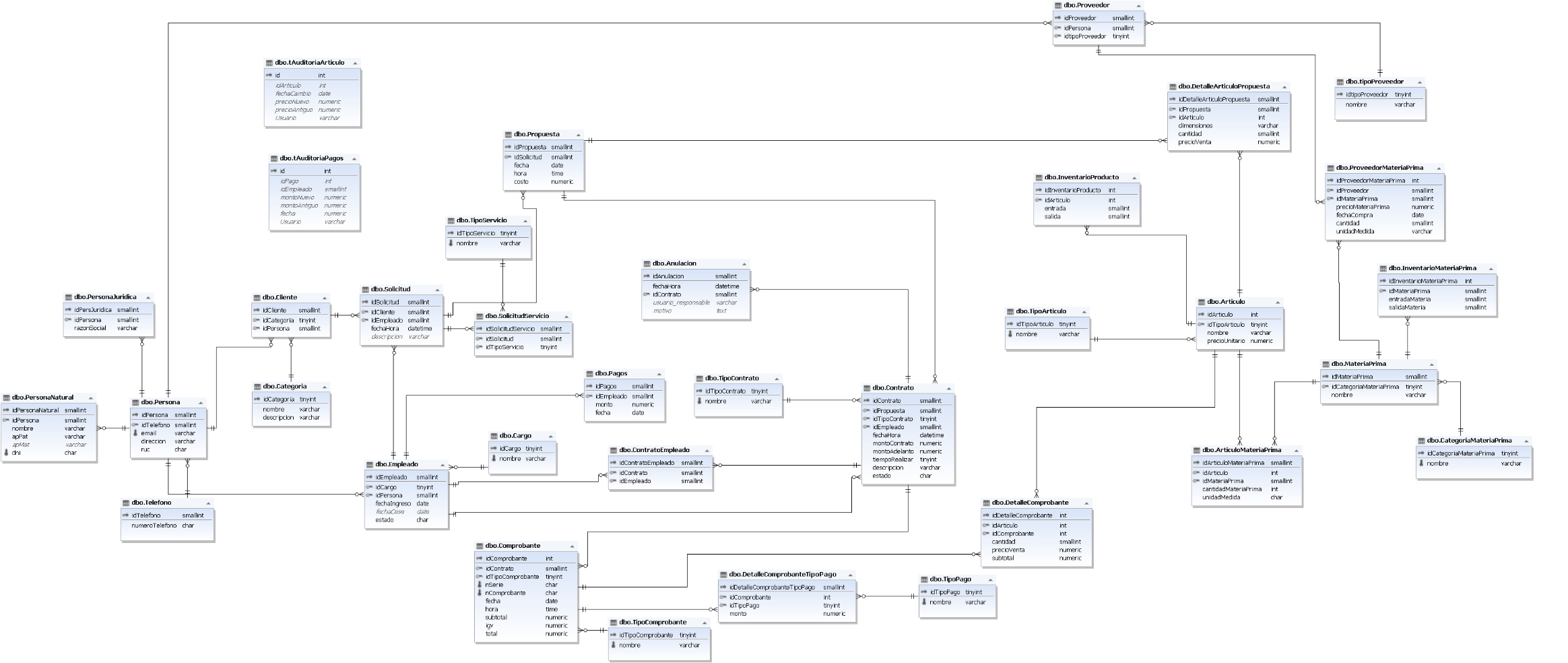
*c. Propósito:* Registrar y gestionar la información relacionada con el inventario de la materia prima.

1. MateriaPrima
   1. *Descripción:* Representa la materia prima.
   2. *Atributos:*
      1. idMateriaPrima(PK): Identificador unívoco de la materia prima.
      2. idCategoriaMateriaPrima(FK): Identificador de la categoría de la materia prima que se relaciona con la materia prima.
      3. nombre: nombre de la materia prima.
   3. *Propósito:* Registrar y gestionar la información relacionada con materia prima.
2. CategoriaMateriaPrima
3. *Descripción:* Representa la categoría de la materia prima.
4. *Atributos:*
   * 1. idCategoriaMateriaPrima(PK): Identificador unívoco de la categoría materia prima.
     2. nombre(FK): identificador de la categoría de la materia prima que se relaciona con la materia prima.

*c. Propósito:* Registrar y gestionar la información relacionada con la categoría a la que pertenece la materia prima.

1. Pagos
   1. *Descripción:* Representa el pago del empleado.
   2. *Atributos:*
      1. idPago (PK): Identificador unívoco del tipo de pago.
      2. idEmpleado: identificador del empleado que se relaciona con el pago.
      3. monto: monto del pago realizado.
      4. fecha: indica la fecha en la que se realiza el pago.
   3. *Propósito:* Registrar y gestionar la información relacionada con los pagos realizados a los empleados por los servicios.
2. TipoPago
   1. *Descripción:* Representa los tipos de pago disponibles.
   2. *Atributos:*
      1. idTipoPago (PK): Identificador unívoco del tipo de pago.
      2. nombre: Nombre del tipo de pago.
   3. *Propósito:* Proporcionar los diferentes métodos o tipos de pago.
3. DetalleComprobanteTipoPago
   1. *Descripción:* Representa el detalle del pago.
   2. *Atributos:*
      1. idDetalleComprobanteTipoPago(PK): Identificador unívoco del detalle de comprobante tipo de pago.
      2. idComprobante: Identificador del comprobante.
      3. idTipoPago: Identificador del tipo de pago.
      4. monto: Monto a pagar.
   3. *Propósito:* Proporcionar los diferentes métodos o tipos de pago.
4. TipoComprobante:
   1. *Descripción:* Representa los tipos de comprobante disponibles.
   2. *Atributos:*
      1. idTipoComprobante (PK): Identificador unívoco del tipo de comprobante.
      2. nombre: Nombre del tipo de comprobante.
   3. *Propósito:* Registrar el tipo de comprobante por el servicio brindado.
5. Articulo:
   1. *Descripción:* Representa el artículo que brindará la empresa.
   2. *Atributos:*
      1. idArticulo(PK): Identificador unívoco del artículo.
      2. idTipo(FK): Identificador del tipo de artículo.
      3. nombre: Nombre del artículo.
      4. preciounitario: Precio unitario del artículo.
   3. *Propósito:* Registrar los datos relacionados al artículo.
6. DetalleArticuloPropuesta:
   1. *Descripción:* Representa el detalle de los artículos que se relacionan a la propuesta.
   2. *Atributos:*
      1. idDetalleArticuloPropuesta(PK): Identificador unívoco del artículo.
      2. idPropuesta (FK): Identificador de la propuesta.
      3. idArticulo: Identificador del artículo.
      4. dimensiones: Dimensiones del artículo.
      5. cantidad: Cantidad del artículo.
      6. precioVenta: Precio de venta del artículo.
   3. *Propósito:* Registrar los datos relacionados al artículo que se encuentran en cada propuesta.
7. TipoArticulo:
   1. *Descripción:* Representa el tipo de artículo.
   2. *Atributos:*
      1. idTipoArticulo(PK): Identificador unívoco del tipo de artículo..
      2. Nombre: Nombre del tipo del artículo.
   3. *Propósito:* Registrar los datos del tipo de artículo.
8. InventarioProducto:
   1. *Descripción:* Representa el inventario del producto, para llevar un control de los productos terminados.
   2. *Atributos:*
      1. idInventarioProducti(PK): Identificador unívoco del inventario del producto.
      2. idArticulo: Identificador del artículo.
      3. salida: salida del articulo.
      4. entrada: entrada del artículo.
   3. *Propósito:* Registrar los datos del inventario del producto.
9. Comprobante:
   1. *Descripción:* Representa a la venta realizada.
   2. *Atributos:*
      1. idComprobante(PK): Identificador unívoco del comprobante(venta).
      2. idContrato: Identificador del contrato.
      3. idTipoComprobante: Identificador del tipo de comprobante.
      4. nSerie: número de serie del comprobante.
      5. nComprobante: número del comprobante.
      6. fecha: fecha en que se realizó la venta.
      7. hora: hora en la que realizó la venta.
      8. subtotal: subtotal de la venta.
      9. igv: igv de la venta.
      10. total: total de la venta, incluyendo el igv.
   3. *Propósito:* Registrar los datos de la venta y del comprobante.
10. TipoComprobante:
    1. *Descripción:* Representa los tipos de comprobante disponibles.
    2. *Atributos:*
       1. idTipoComprobante (PK): Identificador unívoco del tipo de comprobante.
       2. Nombre: Nombre del tipo de comprobante.
    3. *Propósito:* Registrar el tipo de comprobante por el servicio brindado.
11. DetalleComprobante:
    1. *Descripción:* Representa el detalle del comprobante (es decir de la venta).
    2. *Atributos:*
       1. idDetalleComprobante(PK): Identificador unívoco del tipo de comprobante.
       2. idArticulo: Nombre del tipo de comprobante.
       3. idComprobante: Identificador del comprobante.
       4. cantidad: cantidad del artículo vendido.
       5. precioVenta: precio de venta del artículo vendido.
       6. subtotal: subtotal, que vendría a ser la multiplicación de la cantidad y precioVenta de cada artículo.
    3. *Propósito:* Registrar el detalle del comprobante, que funciona como el detalle de venta.
12. Telefono:
    1. *Descripción:* Representa el telefono de la persona.
    2. *Atributos:*
       1. idTelefono(PK): Identificador unívoco del telefono.
       2. numeroTelefono: Numero del telefono.
    3. *Propósito:* Registrar los datos del telefono.

## Modelo Relacional



# IV. Roles y Funciones de cada Integrante

ARANDA ZAPATA, MADELEY MARIELLA

* + Descripción de entidades
  + Elaboración de diagrama relacional
  + Creación de tablas en la BD

HUANCAS FLORES, SEGUNDO ELVIS

* + Descripción de entidades
  + Reconocimiento de entidades
  + Creación de tablas, índices y restricciones en la BD

MORALES RIOS, MARLON JOSUE

* + Listado de requerimientos
  + Descripción de entidades
  + Creación de tablas en la BD

PASACHE PIZARRO, CRISTHIAN JOEL

* + Listado de requerimientos
  + Descripción de entidades
  + Creación de tablas en la BD

PEÑA CORDOVA, BREYNER ANTONIO

* + Listado de requerimientos
  + Descripción de entidades
  + Creación de tablas en la BD

# V. ANEXOS

## Creación de Tablas

CREATE TABLE PersonaNatural(

idPersonaNatural SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idPersona SMALLINT NOT NULL,

nombre VARCHAR(100) NOT NULL,

apPat VARCHAR(50) NOT NULL,

apMat VARCHAR(50) NULL,

dni CHAR(8) UNIQUE NOT NULL

)

CREATE TABLE PersonaJuridica(

idPersJuridica SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idPersona SMALLINT NOT NULL,

razonSocial VARCHAR(150) NOT NULL

)

CREATE TABLE Telefono(

idTelefono SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

numeroTelefono CHAR(9) NOT NULL

)

CREATE TABLE Persona(

idPersona SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idTelefono SMALLINT NOT NULL,

email VARCHAR(50) UNIQUE NOT NULL,

direccion VARCHAR(150) NOT NULL,

ruc CHAR(11) not null

)

CREATE TABLE tipoProveedor(

idtipoProveedor TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(40) NOT NULL

)

CREATE TABLE Proveedor(

idProveedor SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idPersona SMALLINT NOT NULL,

idtipoProveedor TINYINT NOT NULL

)

CREATE TABLE Cargo(

idCargo TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(50) NOT NULL

)

CREATE TABLE Empleado(

idEmpleado SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idCargo TINYINT NOT NULL,

idPersona SMALLINT NOT NULL,

fechaIngreso DATE NOT NULL,

fechaCese DATE NULL,

estado CHAR(1) NOT NULL

)

CREATE TABLE Pagos(

idPagos SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idEmpleado SMALLINT NOT NULL,

monto NUMERIC(7,2) NOT NULL,

fecha DATE NOT NULL

)

CREATE TABLE CategoriaMateriaPrima(

idCategoriaMateriaPrima TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(150) NOT NULL

)

CREATE TABLE MateriaPrima(

idMateriaPrima SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idCategoriaMateriaPrima TINYINT NOT NULL,

nombre VARCHAR(150) NOT NULL

)

CREATE TABLE TipoArticulo(

idTipoArticulo TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(40) NOT NULL

)

CREATE TABLE Articulo(

idArticulo INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idTipoArticulo TINYINT NOT NULL,

nombre VARCHAR(40) NOT NULL,

precioUnitario NUMERIC(7,2) NOT NULL

)

CREATE TABLE InventarioMateriaPrima(

idInventarioMateriaPrima INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idMateriaPrima SMALLINT NOT NULL,

idPersona SMALLINT NOT NULL,

entradaMateria SMALLINT NOT NULL,

salidaMateria SMALLINT NOT NULL

)

CREATE TABLE InventarioProducto(

idInventarioProducto INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idArticulo INT NOT NULL,

entrada SMALLINT NOT NULL,

salida SMALLINT NOT NULL

)

CREATE TABLE Categoria(

idCategoria TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(50) NOT NULL,

descripcion VARCHAR(100) NOT NULL

)

CREATE TABLE Cliente(

idCliente SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idCategoria TINYINT NOT NULL,

idPersona SMALLINT NOT NULL

)

CREATE TABLE Solicitud(

idSolicitud SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idCliente SMALLINT NOT NULL,

idEmpleado SMALLINT NOT NULL,

fechaHora DATETIME NOT NULL,

descripcion VARCHAR(250) NOT NULL

)

CREATE TABLE TipoServicio(

idTipoServicio TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(100) NOT NULL

)

CREATE TABLE SolicitudServicio(

idSolicitudServicio SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idSolicitud SMALLINT NOT NULL,

idTipoServicio TINYINT NOT NULL

)

CREATE TABLE Propuesta(

idPropuesta SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idSolicitud SMALLINT NOT NULL,

fecha DATE NOT NULL,

hora TIME NOT NULL,

costo NUMERIC(7,2) NOT NULL

)

CREATE TABLE DetalleArticuloPropuesta(

idDetalleArticuloPropuesta SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idPropuesta SMALLINT NOT NULL,

idArticulo INT NOT NULL,

dimensiones VARCHAR(50) NOT NULL,

cantidad SMALLINT NOT NULL,

precioVenta NUMERIC(7,2) NOT NULL

)

CREATE TABLE ArticuloMateriaPrima(

idArticuloMateriaPrima SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idArticulo INT NOT NULL,

idMateriaPrima SMALLINT NOT NULL,

cantidadMateriaPrima INT NOT NULL,

unidadMedida char(10) NOT NULL

)

CREATE TABLE TipoComprobante(

idTipoComprobante TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(50) NOT NULL

)

CREATE TABLE TipoContrato(

idTipoContrato TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(30) NOT NULL

)

CREATE TABLE Contrato(

idContrato SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idPropuesta SMALLINT NOT NULL,

idTipoContrato TINYINT NOT NULL,

idEmpleado SMALLINT NOT NULL,

fechaHora DATETIME NOT NULL,

montoContrato NUMERIC(5,2) NOT NULL,

montoAdelanto NUMERIC(5,2) NOT NULL,

tiempoRealizar TINYINT NOT NULL,

descripcion VARCHAR(250) NOT NULL,

estado CHAR(1) NOT NULL

)

CREATE TABLE Comprobante(

idComprobante INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idContrato SMALLINT NOT NULL,

idTipoComprobante TINYINT NOT NULL,

nSerie CHAR(4) NOT NULL,

nComprobante CHAR(8) NOT NULL,

fecha DATE NOT NULL,

hota TIME NOT NULL,

subtotal NUMERIC(7,2) NOT NULL,

igv NUMERIC(7,2) NOT NULL,

total NUMERIC(7,2) NOT NULL

)

CREATE TABLE DetalleComprobante(

idDetalleComprobante INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idArticulo INT NOT NULL,

idComprobante INT NOT NULL,

cantidad SMALLINT NOT NULL,

precioVenta NUMERIC(7,2) NOT NULL,

subtotal NUMERIC(7,2) NOT NULL

)

CREATE TABLE TipoPago(

idTipoPago TINYINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

nombre VARCHAR(50) NOT NULL

)

CREATE TABLE DetalleComprobanteTipoPago(

idDetalleComprobanteTipoPago SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idComprobante INT NOT NULL,

idTipoPago TINYINT NOT NULL,

monto NUMERIC(7,2) NOT NULL

)

CREATE TABLE ContratoEmpleado(

idContratoEmpleado SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idContrato SMALLINT NOT NULL,

idEmpleado SMALLINT NOT NULL

)

CREATE TABLE Anulacion(

idAnulacion SMALLINT IDENTITY(1,1) PRIMARY KEY NOT NULL,

fecha DATE NOT NULL,

hora TIME NOT NULL,

idContrato SMALLINT NOT NULL,

idEmpleado SMALLINT NOT NULL

)

CREATE TABLE ProveedorMateriaPrima(

idProveedorMateriaPrima INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

idProveedor SMALLINT NOT NULL,

idMateriaPrima SMALLINT NOT NULL,

precioMateriaPrima NUMERIC(5,2) NOT NULL,

fechaCompra DATE NOT NULL,

cantidad SMALLINT NOT NULL,

unidadMedida VARCHAR(20) NOT NULL

)

## Creación de Claves Foráneas

PersonaNatural

ALTER TABLE PersonaNatural

ADD CONSTRAINT FK\_PersonaNatural\_Persona

FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona);

-- PersonaJuridica

ALTER TABLE PersonaJuridica

ADD CONSTRAINT FK\_PersonaJuridica\_Persona

FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona);

-- Telefono

ALTER TABLE Persona

ADD CONSTRAINT FK\_Persona\_Telefono

FOREIGN KEY (idTelefono)

REFERENCES Telefono(idTelefono);

-- Proveedor

ALTER TABLE Proveedor

ADD CONSTRAINT FK\_Proveedor\_Persona

FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona),

CONSTRAINT FK\_Proveedor\_TipoProveedor

FOREIGN KEY (idtipoProveedor)

REFERENCES tipoProveedor(idtipoProveedor);

-- Empleado

ALTER TABLE Empleado

ADD CONSTRAINT FK\_Empleado\_Cargo

FOREIGN KEY (idCargo)

REFERENCES Cargo(idCargo),

CONSTRAINT FK\_Empleado\_Persona

FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona);

-- Pagos

ALTER TABLE Pagos

ADD CONSTRAINT FK\_Pagos\_Empleado

FOREIGN KEY (idEmpleado)

REFERENCES Empleado(idEmpleado);

-- MateriaPrima

ALTER TABLE MateriaPrima

ADD CONSTRAINT FK\_CategoriaMateriaPrima\_MateriaPrima

FOREIGN KEY (idCategoriaMateriaPrima)

REFERENCES CategoriaMateriaPrima(idCategoriaMateriaPrima)

-- Articulo

ALTER TABLE Articulo

ADD CONSTRAINT FK\_TipoArticulo\_Articulo

FOREIGN KEY (idTipoArticulo)

REFERENCES TipoArticulo(idTipoArticulo)

-- InventarioMateriaPrima

ALTER TABLE InventarioMateriaPrima

ADD CONSTRAINT FK\_InventarioMateriaPrima\_MateriaPrima

FOREIGN KEY (idMateriaPrima)

REFERENCES MateriaPrima(idMateriaPrima),

CONSTRAINT FK\_InventarioMateriaPrima\_Persona

FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona);

-- InventarioProducto

ALTER TABLE InventarioProducto

ADD CONSTRAINT FK\_InventarioProducto\_Articulo

FOREIGN KEY (idArticulo)

REFERENCES Articulo(idArticulo);

-- Cliente

ALTER TABLE Cliente

ADD CONSTRAINT FK\_Cliente\_Categoria

FOREIGN KEY (idCategoria)

REFERENCES Categoria(idCategoria),

CONSTRAINT FK\_Cliente\_Persona

FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona);

-- Solicitud

ALTER TABLE Solicitud

ADD CONSTRAINT FK\_Solicitud\_Cliente

FOREIGN KEY (idCliente)

REFERENCES Cliente(idCliente),

CONSTRAINT FK\_Solicitud\_Empleado

FOREIGN KEY (idEmpleado)

REFERENCES Empleado(idEmpleado);

-- SolicitudServicio

ALTER TABLE SolicitudServicio

ADD CONSTRAINT FK\_SolicitudServicio\_Solicitud

FOREIGN KEY (idSolicitud)

REFERENCES Solicitud(idSolicitud),

CONSTRAINT FK\_SolicitudServicio\_TipoServicio

FOREIGN KEY (idTipoServicio)

REFERENCES TipoServicio(idTipoServicio);

-- Propuesta

ALTER TABLE Propuesta

ADD CONSTRAINT FK\_Propuesta\_Solicitud

FOREIGN KEY (idSolicitud)

REFERENCES Solicitud(idSolicitud);

-- DetalleArticuloPropuesta

ALTER TABLE DetalleArticuloPropuesta

ADD CONSTRAINT FK\_DetalleArticuloPropuesta\_Propuesta

FOREIGN KEY (idPropuesta)

REFERENCES Propuesta(idPropuesta),

CONSTRAINT FK\_DetalleArticuloPropuesta\_Articulo

FOREIGN KEY (idArticulo)

REFERENCES Articulo(idArticulo);

-- ArticuloMateriaPrima

ALTER TABLE ArticuloMateriaPrima

ADD CONSTRAINT FK\_ArticuloMateriaPrima\_Articulo

FOREIGN KEY (idArticulo)

REFERENCES Articulo(idArticulo),

CONSTRAINT FK\_ArticuloMateriaPrima\_MateriaPrima

FOREIGN KEY (idMateriaPrima)

REFERENCES MateriaPrima(idMateriaPrima);

-- Contrato

ALTER TABLE Contrato

ADD CONSTRAINT FK\_Contrato\_Propuesta

FOREIGN KEY (idPropuesta)

REFERENCES Propuesta(idPropuesta),

CONSTRAINT FK\_Contrato\_TipoContrato

FOREIGN KEY (idTipoContrato)

REFERENCES TipoContrato(idTipoContrato),

CONSTRAINT FK\_Contrato\_Empleado

FOREIGN KEY (idEmpleado)

REFERENCES Empleado(idEmpleado);

-- Comprobante

ALTER TABLE Comprobante

ADD CONSTRAINT FK\_Comprobante\_Contrato

FOREIGN KEY (idContrato)

REFERENCES Contrato(idContrato),

CONSTRAINT FK\_Comprobante\_TipoComprobante

FOREIGN KEY (idTipoComprobante)

REFERENCES TipoComprobante(idTipoComprobante);

-- DetalleComprobante

ALTER TABLE DetalleComprobante

ADD CONSTRAINT FK\_DetalleComprobante\_Articulo

FOREIGN KEY (idArticulo)

REFERENCES Articulo(idArticulo),

CONSTRAINT FK\_DetalleComprobante\_Comprobante

FOREIGN KEY (idComprobante)

REFERENCES Comprobante(idComprobante);

-- DetalleComprobanteTipoPago

ALTER TABLE DetalleComprobanteTipoPago

ADD CONSTRAINT FK\_DetalleComprobanteTipoPago\_Comprobante

FOREIGN KEY (idComprobante)

REFERENCES Comprobante(idComprobante),

CONSTRAINT FK\_DetalleComprobanteTipoPago\_TipoPago

FOREIGN KEY (idTipoPago)

REFERENCES TipoPago(idTipoPago);

-- ContratoEmpleado

ALTER TABLE ContratoEmpleado

ADD CONSTRAINT FK\_ContratoEmpleado\_Contrato

FOREIGN KEY (idContrato)

REFERENCES Contrato(idContrato),

CONSTRAINT FK\_ContratoEmpleado\_Empleado

FOREIGN KEY (idEmpleado)

REFERENCES Empleado(idEmpleado);

-- Anulacion

ALTER TABLE Anulacion

ADD CONSTRAINT FK\_Anulacion\_Contrato

FOREIGN KEY (idContrato)

REFERENCES Contrato(idContrato),

CONSTRAINT FK\_Anulacion\_Empleado

FOREIGN KEY (idEmpleado)

REFERENCES Empleado(idEmpleado);

-- ProveedorMateriaPrima

ALTER TABLE ProveedorMateriaPrima

ADD CONSTRAINT FK\_ProveedorMateriaPrima\_Proveedor

FOREIGN KEY (idProveedor)

REFERENCES Proveedor(idProveedor),

CONSTRAINT FK\_ProveedorMateriaPrima\_MateriaPrima

FOREIGN KEY (idMateriaPrima)

REFERENCES MateriaPrima(idMateriaPrima);

## Creación de Restricciones

--PersonaNatural

ALTER TABLE PersonaNatural

ADD CONSTRAINT CK\_dniNumeros CHECK(dni LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

CONSTRAINT CK\_dniOcho CHECK(LEN(dni) = 8 );

--PersonaJuridica

ALTER TABLE PersonaJuridica

ADD CONSTRAINT chk\_razonSocial\_no\_vacia

CHECK ( razonSocial <> '');

--Propuesta

ALTER TABLE Propuesta

ADD CONSTRAINT CK\_costoPropuesta CHECK(costo > 0);

--Contrato

ALTER TABLE Contrato

ADD CONSTRAINT CK\_montoAdelanto CHECK(montoAdelanto > 0),

CONSTRAINT CK\_montoContrato CHECK(montoContrato >= 0),

CONSTRAINT CK\_tiempoRealizar CHECK (tiempoRealizar > 0),

CONSTRAINT CK\_estadoContrato CHECK(estado IN ('A','I') );

--Comprobante

ALTER TABLE Comprobante

ADD CONSTRAINT CK\_serie CHECK (nSerie LIKE '[A-Z][0-9][0-9][0-9]'),

CONSTRAINT CK\_nComprobante CHECK (nComprobante LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

CONSTRAINT UQ\_comprobante UNIQUE(nSerie,nComprobante),

CONSTRAINT CK\_igvComprobante CHECK(igv > 0),--el igv sacado siempre es positivo

CONSTRAINT CK\_subtotalComprobante CHECK(subtotal > 0),--Siempre tendremos un monto positivo

CONSTRAINT CK\_totalComprobante CHECK(total = igv + subtotal),--Para comprobar el total y no hayan incoherencias

CONSTRAINT CK\_igv18 CHECK(igv = total - subtotal),--para verificar si el igv sacado coincide

CONSTRAINT CK\_subtotaligv CHECK(subtotal = total / 1.18);--verificamos que el subtotal sea el correcto

--Empleado

ALTER TABLE Empleado ADD

CONSTRAINT CK\_empleado CHECK( estado IN('A','I') ),--A:activo,I:inactivo

CONSTRAINT CK\_fechaIngreso CHECK(fechaIngreso <= GETDATE()), -- La fecha de ingreso no puede ser en el futuro

CONSTRAINT CK\_fechaCese CHECK(fechaCese >= fechaIngreso); -- La fecha de cese debe ser posterior o igual a la fecha de ingreso

--ProveedorMateriaPrima

ALTER TABLE ProveedorMateriaPrima

ADD CONSTRAINT CK\_CantidadNoNegativo\_ProveedorMateria\_Prima CHECK (cantidad > 0),

CONSTRAINT CK\_unidadMedida\_ProveedorMateria\_Prima CHECK( unidadMedida IN ('m','kg','kt'));

--InventarioMateriaPrima

ALTER TABLE InventarioMateriaPrima

ADD CONSTRAINT CK\_EntradaNoNegativoInventarioMateriaPrima CHECK (entradaMateria >= 0),

CONSTRAINT CK\_SalidaNonNegativeInventarioMateriaPrima CHECK (salidaMateria >= 0);

---CategoriaMateriaPrima

ALTER TABLE CategoriaMateriaPrima

ADD CONSTRAINT UQ\_nombre\_unico\_categoria UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_CategoriaMateriaPrima CHECK(nombre <> '');

--InventarioProducto

ALTER TABLE InventarioProducto

ADD CONSTRAINT CK\_EntradaNoNegativoInventarioProducto CHECK (entrada >= 0),

CONSTRAINT CK\_SalidaNonNegativeInventarioProducto CHECK (salida >= 0);

--TipoPago

ALTER TABLE TipoPago

ADD CONSTRAINT UQ\_nombre\_unico\_TipoPago UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_TipoPago CHECK(nombre <> '');

--TipoArticulo

ALTER TABLE TipoArticulo

ADD CONSTRAINT UQ\_nombre\_unico\_TipoArticulo UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_TipoArticulo CHECK(nombre <> '');

--TipoContrato

ALTER TABLE TipoContrato

ADD CONSTRAINT UQ\_nombre\_unico\_TipoContrato UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_TipoContrato CHECK(nombre <> '');

--TipoServicio

ALTER TABLE TipoServicio

ADD CONSTRAINT UQ\_nombre\_unico\_TipoServicio UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_TipoServicio CHECK(nombre <> '');

--Cargo

ALTER TABLE Cargo

ADD CONSTRAINT UQ\_nombre\_unico\_Cargo UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_Cargo CHECK(nombre <> '');

--TipoComprobante

ALTER TABLE TipoComprobante

ADD CONSTRAINT UQ\_nombre\_unico\_TipoComprobante UNIQUE(nombre),

CONSTRAINT CK\_nombre\_noVacio\_TipoComprobante CHECK(nombre <> '');

--Telefono

ALTER TABLE Telefono

ADD CONSTRAINT CK\_telefono\_formato CHECK(numeroTelefono LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'), -- Verifica que el teléfono comience con un dígito

CONSTRAINT CK\_longitud\_telefono CHECK(LEN(numeroTelefono) >= 6 AND LEN(numeroTelefono) <= 9); -- Si es telefono de casa son 6 a 7 dig, si es celular son 9 dig

--Persona

ALTER TABLE Persona

ADD CONSTRAINT CK\_ruc\_formato

CHECK ( (SUBSTRING(ruc, 1, 2) IN ('10', '20')) AND -- Verifica el formato del inicio de RUC 10 o RUC 20

LEN(ruc) = 11 AND -- Verifica la longitud del RUC

ruc LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'); -- Verifica que solo contiene dígitos numérico

## Creación de Índices

Los índices son creados con el fin de optimizar y facilitar el rendimiento al momento de hacer una consulta.

En esta base de datos hemos creído convenientes creas los siguientes índices:

Índice de la tabla PersonaNatural:

Este índice se crea en la columna DNI para facilitar la búsqueda rápida y eficiente de registros específicos de personas naturales por su DNI, dado que el DNI es único.

CREATE NONCLUSTERED INDEX IDX\_DNI\_PersonaNatural

ON PersonaNatural(dni);

Índice de la tabla Cliente:

Este índice no agrupado se crea para consultar clientes por categoría, mejorando el rendimiento de las consultas que filtran clientes según su categoría.

CREATE NONCLUSTERED INDEX IDX\_IDCategoria\_Cliente

ON Cliente(idCategoria);

Este índice no agrupado es creado para consultas que involucren clientes independientemente de si son personas jurídicas o naturales.Así Facilita la búsqueda rápida por idPersona.

CREATE NONCLUSTERED INDEX IDX\_IDPersona\_Cliente

ON Cliente(idPersona)

Índice de la tabla Solicitud:

Este índice no agrupado se crea para realizar consultas que involucren intervalos de tiempo específicos o para ordenar las solicitudes por fecha y hora.

CREATE NONCLUSTERED INDEX IDX\_FechaHora\_Solicitud

ON Solicitud(fechaHora)

Índice de la tabla Contrato:

Este índice no agrupado es creado para facilitar la búsqueda y recuperación eficiente de contratos basados en su idPropuesta asociada.

CREATE NONCLUSTERED INDEX IDX\_IDPropuesta\_Contrato

ON Contrato(idPropuesta)

Este índice no agrupado es creado para cuando se realicen consultas que filtren o ordene contratos por su estado.

Si las consultas con frecuencia filtran o ordenan contratos por su estado, este índice no agrupado mejorará el rendimiento.

CREATE NONCLUSTERED INDEX IDX\_Estado\_Contrato

ON Contrato(estado);

PROCEDIMIENTOS ALMACENADOS

Procedimientos almacenados de inserción y actualización

/\*

Nombre del procedimiento: paActualizaPersonaNatural

Propósito: Actualiza o inserta información de una persona natural en la base de datos.

Parámetros de entrada:

- @idPersonaNatural: Identificador único de la persona natural.

- @idPersona: Identificador único de la persona asociada.

- @nombre: Nombre de la persona natural.

- @apPat: Apellido paterno de la persona natural.

- @apMat: Apellido materno de la persona natural.

- @dni: DNI de la persona natural.

\*/

CREATE PROCEDURE paActualizaPersonaNatural

(

@idPersonaNatural SMALLINT,

@idPersona SMALLINT,

@nombre VARCHAR(100),

@apPat VARCHAR(50),

@apMat VARCHAR(50),

@dni CHAR(8)

)

AS

BEGIN

-- Comprueba si la persona asociada existe

IF EXISTS (SELECT 1 FROM Persona WHERE idPersona = @idPersona)

BEGIN

-- Comprueba si no existe una referencia de persona en persona natural

IF NOT EXISTS (SELECT 1 FROM PersonaNatural WHERE idPersona = @idPersona)

BEGIN

-- Comprueba si existe un registro con el ID de persona natural proporcionado

IF EXISTS (SELECT 1 FROM PersonaNatural WHERE idPersonaNatural = @idPersonaNatural)

BEGIN

-- Actualización de los datos existentes

UPDATE PersonaNatural

SET

idPersona = @idPersona,

nombre = @nombre,

apPat = @apPat,

apMat = @apMat,

dni = @dni

WHERE idPersonaNatural = @idPersonaNatural

END

ELSE

BEGIN

-- Inserción de un nuevo registro

INSERT INTO PersonaNatural(idPersona, nombre, apPat, apMat, dni)

VALUES (@idPersona, @nombre, @apPat, @apMat, @dni);

END

END

-- Mensaje si ya existe el ID de persona

ELSE

BEGIN

PRINT 'Ya se registró el idPersona proporcionado, utilice otro'

END

END

-- Mensaje si el ID de persona no se encuentra en la tabla Persona

PRINT 'El idPersona proporcionado no se encuentra en la tabla Persona'

END

/\*

Nombre del procedimiento: paActualizaPersonaJuridica

Propósito: Actualiza o inserta información de una persona jurídica en la base de datos.

Parámetros de entrada:

- @idPersJuridica: Identificador único de la persona jurídica.

- @idPersona: Identificador único de la persona asociada.

- @razonSocial: Razón social de la persona jurídica.

\*/

CREATE PROCEDURE paActualizaPersonaJuridica

(

@idPersJuridica SMALLINT,

@idPersona SMALLINT,

@razonSocial VARCHAR(150)

)

AS

BEGIN

-- Comprueba si existe la persona

IF EXISTS (SELECT 1 FROM Persona WHERE idPersona = @idPersona)

BEGIN

-- Comprueba si la persona no es persona jurídica

IF NOT EXISTS (SELECT 1 FROM PersonaJuridica WHERE idPersona = @idPersona)

BEGIN

-- Comprueba si ya existe la persona jurídica

IF EXISTS (SELECT 1 FROM PersonaJuridica WHERE idPersJuridica = @idPersJuridica)

BEGIN

-- Actualización de la persona jurídica existente

UPDATE PersonaJuridica

SET

idPersona = @idPersona,

razonSocial = @razonSocial

WHERE idPersJuridica = @idPersJuridica;

END

ELSE

BEGIN

-- Inserción de una nueva persona jurídica

INSERT INTO PersonaJuridica(idPersona, razonSocial)

VALUES (@idPersona, @razonSocial)

END

END

ELSE

BEGIN

PRINT 'El idPersona ya existe en la tabla'

END

END

ELSE

BEGIN

PRINT 'No existe idPersona en la tabla Persona'

END

END

/\*

Nombre del procedimiento: paActualizarTelefono

Propósito: Actualiza o inserta un número de teléfono en la base de datos.

Parámetros de entrada:

- @id: Identificador único del teléfono.

- @numTelefono: Número de teléfono a actualizar o insertar.

\*/

CREATE PROCEDURE paActualizarTelefono

(

@id SMALLINT,

@numTelefono CHAR(9)

)

AS

BEGIN

DECLARE @numTelefonoA VARCHAR(40)

-- Comprueba si el teléfono existe para realizar una actualización

IF EXISTS (SELECT 1 FROM Telefono WHERE idTelefono = @id)

BEGIN

SELECT @numTelefonoA = numeroTelefono

FROM Telefono

WHERE idTelefono = @id

-- Realiza la actualización si el número es diferente

IF @numTelefono <> @numTelefonoA

BEGIN

-- Actualiza el número de teléfono

UPDATE Telefono

SET numeroTelefono = @numTelefono

WHERE idTelefono = @id

END

END

ELSE

BEGIN

-- Inserta un nuevo número de teléfono si no existe

INSERT INTO Telefono(numeroTelefono)

VALUES (@numTelefono)

END

END

/\*

Nombre del procedimiento: paActualizaPersona

Propósito: Actualiza o inserta información de una persona en la base de datos.

Parámetros de entrada:

- @idPersona: Identificador único de la persona.

- @idTelefono: Identificador único del teléfono asociado.

- @email: Correo electrónico de la persona.

- @direccion: Dirección de la persona.

- @ruc: RUC de la persona.

\*/

CREATE PROCEDURE paActualizaPersona

(

@idPersona SMALLINT,

@idTelefono SMALLINT,

@email VARCHAR(50),

@direccion VARCHAR(150),

@ruc CHAR(11)

)

AS

BEGIN

-- Verifica si el teléfono asociado existe

IF EXISTS(SELECT 1 FROM Telefono WHERE idTelefono = @idTelefono)

BEGIN

-- Si no existe teléfono asignado a la persona

IF NOT EXISTS(SELECT 1 FROM Persona WHERE idTelefono = @idTelefono)

BEGIN

-- Si existe la persona

IF EXISTS(SELECT 1 FROM Persona WHERE idPersona = @idPersona)

BEGIN

-- Actualiza la información de la persona y asigna el teléfono

UPDATE Persona

SET idTelefono = @idTelefono,

email = @email,

direccion = @direccion,

ruc = @ruc

WHERE idPersona = @idPersona

END

-- Si no existe la persona, la inserta y asigna el teléfono

ELSE

BEGIN

-- Inserta una nueva persona y asigna el teléfono

INSERT INTO Persona

VALUES(@idTelefono, @email, @direccion, @ruc)

END

END

-- Si ya existe el teléfono asignado a la persona

ELSE BEGIN

PRINT 'La persona ya existe'

END

END

-- Si no existe el registro de teléfono

ELSE BEGIN

PRINT 'ERROR, VERIFIQUE LA EXISTENCIA DE REGISTRO DE TELEFONO'

END

END

/\*

Nombre del procedimiento: paActualizaTipoProveedor

Propósito: Actualiza o inserta un tipo de proveedor en la base de datos.

Parámetros de entrada:

- @id: Identificador único del tipo de proveedor.

- @nombre: Nombre del tipo de proveedor.

\*/

CREATE PROCEDURE paActualizaTipoProveedor

(

@id TINYINT,

@nombre VARCHAR(40)

)

AS

BEGIN

DECLARE @nombreA VARCHAR(40)

-- Comprueba si el tipo de proveedor existe

IF EXISTS (SELECT 1 FROM tipoProveedor WHERE idtipoProveedor = @id)

BEGIN

SELECT @nombreA = nombre

FROM tipoProveedor

WHERE idtipoProveedor = @id

-- Actualiza el nombre si es diferente

IF @nombre <> @nombreA

BEGIN

UPDATE tipoProveedor

SET nombre = @nombre

WHERE idtipoProveedor = @id

END

ELSE

BEGIN

-- Inserta un nuevo tipo de proveedor si no existe

IF NOT EXISTS (SELECT 1 FROM tipoProveedor WHERE nombre = @nombre)

BEGIN

INSERT INTO tipoProveedor (nombre)

VALUES(@nombre)

END

ELSE

BEGIN

PRINT 'Ya se ha registrado ese tipo de proveedor'

END

END

END

/\*

Nombre del procedimiento: paActualizaProveedor

Propósito: Actualiza o inserta información de un proveedor en la base de datos.

Parámetros de entrada:

- @idProveedor: Identificador único del proveedor.

- @idPersona: Identificador único de la persona asociada al proveedor.

- @idtipoProveedor: Identificador único del tipo de proveedor asociado.

\*/

CREATE PROCEDURE paActualizaProveedor

(

@idProveedor SMALLINT,

@idPersona SMALLINT,

@idtipoProveedor TINYINT

)

AS

BEGIN

DECLARE @idPersonaA SMALLINT, @idtipoProveedorA TINYINT

-- Verifica la existencia de la persona en la tabla Persona

IF EXISTS (SELECT 1 FROM Persona WHERE idPersona = @idPersona )

BEGIN

-- Verifica la existencia del tipo de proveedor en la tabla tipoProveedor

IF EXISTS (SELECT 1 FROM tipoProveedor WHERE idtipoProveedor = @idtipoProveedor)

BEGIN

-- Verifica si el idPersona ya se encuentra registrado en la tabla Proveedor

IF NOT EXISTS (SELECT 1 FROM PROVEEDOR WHERE idPersona = @idPersona)

BEGIN

-- Comprueba si existe el idProveedor, si existe, se trata de una actualización

IF EXISTS (SELECT 1 FROM Proveedor WHERE idProveedor = @idProveedor)

BEGIN

-- Realiza la actualización del proveedor

UPDATE Proveedor

SET idPersona = @idPersona,

idtipoProveedor = @idtipoProveedor

WHERE idProveedor = @idProveedor

END

ELSE

BEGIN

-- Inserta un nuevo proveedor

INSERT INTO Proveedor (idPersona, idtipoProveedor)

VALUES (@idPersona, @idtipoProveedor)

END

END

ELSE

BEGIN

PRINT 'El idPersona ya se encuentra registrado'

END

END

ELSE

BEGIN

PRINT 'El idtipoProveedor pasado como parámetro no se encuentra registrado en tipoProveedor'

END

END

ELSE

BEGIN

PRINT 'El idPersona pasado como parámetro no se encuentra registrado en Persona'

END

END

/\*

Nombre del procedimiento: paActualizaCargo

Propósito: Actualiza o inserta un cargo en la base de datos.

Parámetros de entrada:

- @idCargo: Identificador único del cargo.

- @nombre: Nombre del cargo.

\*/

CREATE PROCEDURE paActualizaCargo

(

@idCargo TINYINT,

@nombre VARCHAR(50)

)

AS

BEGIN

IF EXISTS (SELECT 1 FROM Cargo WHERE idCargo = @idCargo)

-- Actualiza el cargo si existe

UPDATE Cargo

SET nombre = @nombre

WHERE idCargo = @idCargo

ELSE

IF NOT EXISTS(SELECT 1 FROM Cargo WHERE nombre = @nombre)

BEGIN

-- Inserta un nuevo cargo si el nombre no está registrado

INSERT INTO Cargo (nombre)

VALUES(@nombre)

END

ELSE

BEGIN

PRINT 'El nombre del cargo ya se encuentra registrado'

END

END

/\*

Nombre del procedimiento: paActualizaEmpleado

Propósito: Actualiza o inserta información de un empleado en la base de datos.

Parámetros de entrada:

- @idEmpleado: Identificador único del empleado.

- @idCargo: Identificador único del cargo del empleado.

- @idPersona: Identificador único de la persona asociada al empleado.

- @fechaIngreso: Fecha de ingreso del empleado.

- @fechaCese: Fecha de cese del empleado.

- @estado: Estado del empleado.

\*/

CREATE PROCEDURE paActualizaEmpleado

(

@idEmpleado SMALLINT,

@idCargo TINYINT,

@idPersona SMALLINT,

@fechaIngreso DATE,

@fechaCese DATE,

@estado CHAR(1)

)

AS

BEGIN

-- Verifica si el cargo existe en la tabla Cargo

IF EXISTS(SELECT 1 FROM Cargo WHERE idCargo = @idCargo )

BEGIN

-- Verifica si la persona existe en la tabla Persona

IF EXISTS(SELECT 1 FROM Persona WHERE idPersona = @idPersona)

BEGIN

-- Verifica si el idPersona no está registrado como empleado

IF NOT EXISTS(SELECT 1 FROM Empleado WHERE idPersona = @idPersona)

BEGIN

-- Verifica si el idEmpleado existe, en caso afirmativo, se trata de una actualización

IF EXISTS(SELECT 1 FROM Empleado WHERE idEmpleado = @idEmpleado)

BEGIN

-- Realiza la actualización del empleado

UPDATE Empleado

SET idCargo = @idCargo,

idPersona = @idPersona,

fechaIngreso = @fechaIngreso,

fechaCese = @fechaCese,

estado = @estado

WHERE idEmpleado = @idEmpleado

END

ELSE

BEGIN

-- Inserta un nuevo empleado

INSERT INTO Empleado(idCargo,idPersona,fechaIngreso,fechaCese,estado)

VALUES(@idCargo,@idPersona,@fechaIngreso,@fechaCese,@estado)

END

END

ELSE

BEGIN

PRINT 'El idPersona pasado como parámetro ya está registrado en Empleado'

END

END

ELSE

BEGIN

PRINT 'El idPersona no está registrado en Persona'

END

END

ELSE

BEGIN

PRINT 'El idCargo no está registrado en Cargo'

END

END

/\*

Nombre del procedimiento: paActualizaPago

Propósito: Actualiza o inserta información de un pago en la base de datos.

Parámetros de entrada:

- @idPagos: Identificador único del pago.

- @idEmpleado: Identificador único del empleado asociado al pago.

- @monto: Monto del pago.

- @fecha: Fecha del pago.

\*/

CREATE PROCEDURE paActualizaPago

(

@idPagos SMALLINT,

@idEmpleado SMALLINT,

@monto NUMERIC(7,2),

@fecha DATE

)

AS

BEGIN

-- Verifica si el empleado existe en la tabla Empleado

IF EXISTS(SELECT 1 FROM Empleado WHERE idEmpleado = @idEmpleado)

BEGIN

-- Verifica si el pago existe en la tabla Pagos

IF EXISTS(SELECT 1 FROM Pagos WHERE idPagos = @idPagos)

BEGIN

-- Realiza la actualización del pago

UPDATE Pagos

SET idEmpleado = @idEmpleado,

monto = @monto,

fecha = @fecha

WHERE idPagos = @idPagos

END

ELSE

BEGIN

-- Inserta un nuevo pago

INSERT INTO Pagos(idEmpleado, monto, fecha)

VALUES(@idEmpleado, @monto,@fecha)

END

END

ELSE

BEGIN

PRINT 'El idEmpleado pasado como parámetro no está registrado en Empleado'

END

END

/\*

Nombre del procedimiento: paActualizaCategoriaMP

Propósito: Actualiza o inserta una categoría de materia prima en la base de datos.

Parámetros de entrada:

- @idCategoriaMateriaPrima: Identificador único de la categoría de materia prima.

- @nombre: Nombre de la categoría de materia prima.

\*/

CREATE PROCEDURE paActualizaCategoriaMP

(

@idCategoriaMateriaPrima TINYINT,

@nombre VARCHAR(50)

)

AS

BEGIN

IF EXISTS (SELECT 1 FROM CategoriaMateriaPrima WHERE idCategoriaMateriaPrima=@idCategoriaMateriaPrima)

BEGIN

-- Actualiza la categoría si existe

UPDATE CategoriaMateriaPrima

SET nombre = @nombre

WHERE idCategoriaMateriaPrima=@idCategoriaMateriaPrima

END

ELSE

BEGIN

-- Inserta una nueva categoría si el nombre no está registrado

IF NOT EXISTS (SELECT 1 FROM CategoriaMateriaPrima WHERE nombre = @nombre)

BEGIN

INSERT INTO CategoriaMateriaPrima(nombre)

VALUES (@nombre)

END

ELSE

BEGIN

PRINT 'El nombre de la categoría ya se encuentra registrado'

END

END

END

/\*

Nombre del procedimiento: paActualizaMateriaPrima

Propósito: Actualiza o inserta información de una materia prima en la base de datos.

Parámetros de entrada:

- @idMateriaPrima: Identificador único de la materia prima.

- @idCategoriaMateriaPrima: Identificador único de la categoría de materia prima asociada.

- @nombre: Nombre de la materia prima.

\*/

CREATE PROCEDURE paActualizaMateriaPrima

(

@idMateriaPrima SMALLINT,

@idCategoriaMateriaPrima TINYINT,

@nombre VARCHAR(150)

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM CategoriaMateriaPrima WHERE idCategoriaMateriaPrima = @idCategoriaMateriaPrima)

BEGIN

IF EXISTS(SELECT 1 FROM MateriaPrima WHERE idMateriaPrima = @idMateriaPrima)

BEGIN

IF NOT EXISTS(SELECT 1 FROM MateriaPrima WHERE nombre = @nombre)

BEGIN

-- Actualización de la materia prima

UPDATE MateriaPrima

SET idCategoriaMateriaPrima = @idCategoriaMateriaPrima,

nombre = @nombre

WHERE idMateriaPrima = @idMateriaPrima

END

ELSE

BEGIN

PRINT 'La materia prima ya está registrada'

END

END

ELSE

BEGIN

-- Inserción de una nueva materia prima

INSERT INTO MateriaPrima(idCategoriaMateriaPrima, nombre)

VALUES(@idCategoriaMateriaPrima, @nombre)

END

END

ELSE

BEGIN

PRINT 'La idCategoriaMateriaPrima no está registrada en CategoriaMateriaPrima'

END

END

/\*

Nombre del procedimiento: paActualizaTipoArticulo

Propósito: Actualiza o inserta información de un tipo de artículo en la base de datos.

Parámetros de entrada:

- @idTipoArticulo: Identificador único del tipo de artículo.

- @nombre: Nombre del tipo de artículo.

\*/

CREATE PROCEDURE paActualizaTipoArticulo

(

@idTipoArticulo TINYINT,

@nombre VARCHAR(40)

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM TipoArticulo WHERE idTipoArticulo=@idTipoArticulo)

BEGIN

-- Actualización del tipo de artículo

UPDATE TipoArticulo

SET nombre = @nombre

WHERE idTipoArticulo=@idTipoArticulo

END

ELSE

BEGIN

IF NOT EXISTS (SELECT 1 FROM TipoArticulo WHERE nombre = @nombre)

BEGIN

-- Inserción de un nuevo tipo de artículo

INSERT INTO TipoArticulo(nombre)

VALUES (@nombre)

END

ELSE

BEGIN

PRINT 'El tipo de artículo ya está registrado'

END

END

END

/\*

Nombre del procedimiento: paActualizaArticulo

Propósito: Actualiza o inserta información de un artículo en la base de datos.

Parámetros de entrada:

- @idArticulo: Identificador único del artículo.

- @idTipoArticulo: Identificador único del tipo de artículo asociado.

- @nombre: Nombre del artículo.

- @precioUnitario: Precio unitario del artículo.

\*/

CREATE PROCEDURE paActualizaArticulo

(

@idArticulo INT,

@idTipoArticulo TINYINT,

@nombre VARCHAR(40),

@precioUnitario NUMERIC(7,2)

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM TipoArticulo WHERE idTipoArticulo = @idTipoArticulo)

BEGIN

IF EXISTS(SELECT 1 FROM Articulo WHERE idArticulo = @idArticulo)

BEGIN

-- Actualización del artículo

UPDATE Articulo

SET idTipoArticulo = @idTipoArticulo,

nombre = @nombre,

precioUnitario = @precioUnitario

WHERE idArticulo = @idArticulo

END

ELSE

BEGIN

-- Inserción de un nuevo artículo

INSERT INTO Articulo(idTipoArticulo, nombre, precioUnitario)

VALUES (@idTipoArticulo, @nombre, @precioUnitario)

END

END

ELSE

BEGIN

PRINT 'El idTipoArticulo no está registrado en TipoArticulo'

END

END

/\*

Nombre del procedimiento: paActualizaInventarioMateriaPrima

Propósito: Actualiza o inserta información del inventario de materia prima en la base de datos.

Parámetros de entrada:

- @idInventarioMateriaPrima: Identificador único del inventario de materia prima.

- @idMateriaPrima: Identificador único de la materia prima asociada.

- @entradaMateria: Cantidad de entrada de materia prima.

- @salidaMateria: Cantidad de salida de materia prima.

\*/

CREATE PROCEDURE paActualizaInventarioMateriaPrima

(

@idInventarioMateriaPrima INT,

@idMateriaPrima SMALLINT,

@entradaMateria SMALLINT,

@salidaMateria SMALLINT

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM MateriaPrima WHERE idMateriaPrima = @idMateriaPrima)

BEGIN

IF EXISTS(SELECT 1 FROM InventarioMateriaPrima WHERE idInventarioMateriaPrima = @idInventarioMateriaPrima)

BEGIN

-- Actualización del inventario de materia prima

UPDATE InventarioMateriaPrima

SET idMateriaPrima = @idMateriaPrima,

entradaMateria = @entradaMateria,

salidaMateria = @salidaMateria

WHERE idInventarioMateriaPrima = @idInventarioMateriaPrima

END

ELSE

BEGIN

-- Inserción de un nuevo registro de inventario de materia prima

INSERT INTO InventarioMateriaPrima(idMateriaPrima, entradaMateria, salidaMateria)

VALUES (@idMateriaPrima, @entradaMateria, @salidaMateria)

END

END

ELSE

BEGIN

PRINT 'El idMateriaPrima no está registrado en MateriaPrima'

END

END

/\*

Nombre del procedimiento: paActualizaInventarioProducto

Propósito: Actualiza o inserta información del inventario de producto en la base de datos.

Parámetros de entrada:

- @idInventarioProducto: Identificador único del inventario de producto.

- @idArticulo: Identificador único del artículo asociado.

- @entrada: Cantidad de entrada de producto.

- @salida: Cantidad de salida de producto.

\*/

CREATE PROCEDURE

paActualizaInventarioProducto

(

@idInventarioProducto INT,

@idArticulo INT,

@entrada SMALLINT,

@salida SMALLINT

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Articulo WHERE idArticulo = @idArticulo)

BEGIN

IF EXISTS(SELECT 1 FROM InventarioProducto WHERE idInventarioProducto = @idInventarioProducto)

BEGIN

-- Actualización del inventario de producto

UPDATE InventarioProducto

SET idArticulo = @idArticulo,

entrada = @entrada,

salida = @salida

WHERE idInventarioProducto = @idInventarioProducto

END

ELSE

BEGIN

-- Inserción de un nuevo registro de inventario de producto

INSERT INTO InventarioProducto(idArticulo, entrada, salida)

VALUES(@idArticulo, @entrada, @salida)

END

END

ELSE

BEGIN

PRINT 'El idArticulo no está registrado en Articulo'

END

END

/\*

Nombre del procedimiento: paActualizaCategoria

Propósito: Actualiza o inserta información de una categoría en la base de datos.

Parámetros de entrada:

- @idCategoria: Identificador único de la categoría.

- @nombre: Nombre de la categoría.

- @descripcion: Descripción de la categoría.

\*/

CREATE PROCEDURE paActualizaCategoria

(

@idCategoria TINYINT,

@nombre VARCHAR(50),

@descripcion VARCHAR(100)

)

AS

BEGIN

IF NOT EXISTS(SELECT 1 FROM Categoria WHERE nombre = @nombre)

BEGIN

IF EXISTS(SELECT 1 FROM Categoria WHERE idCategoria = @idCategoria)

BEGIN

-- Actualización de la categoría

UPDATE Categoria

SET nombre = @nombre,

descripcion = @descripcion

WHERE idCategoria = @idCategoria

END

ELSE

BEGIN

-- Inserción de una nueva categoría

INSERT INTO Categoria(nombre, descripcion)

VALUES (@nombre, @descripcion)

END

END

ELSE

BEGIN

PRINT 'La categoría ya está registrada'

END

END

/\*

Nombre del procedimiento: paActualizaCliente

Propósito: Actualiza o inserta información de un cliente en la base de datos.

Parámetros de entrada:

- @idCliente: Identificador único del cliente.

- @idCategoria: Identificador único de la categoría asociada al cliente.

- @idPersona: Identificador único de la persona asociada al cliente.

\*/

CREATE PROCEDURE paActualizaCliente

(

@idCliente SMALLINT,

@idCategoria TINYINT,

@idPersona SMALLINT

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Categoria WHERE idCategoria = @idCategoria)

BEGIN

IF EXISTS(SELECT 1 FROM Persona WHERE idPersona = @idPersona)

BEGIN

-- Cada cliente está relacionado con un idPersona

IF NOT EXISTS (SELECT 1 FROM Cliente WHERE idPersona = @idPersona)

BEGIN

IF EXISTS(SELECT 1 FROM Cliente WHERE idCliente = @idCliente)

BEGIN

-- Actualización del cliente

UPDATE Cliente

SET idCategoria = @idCategoria,

idPersona = @idPersona

WHERE idCliente = @idCliente

END

ELSE

BEGIN

-- Inserción de un nuevo cliente

INSERT INTO Cliente(idCategoria, idPersona)

VALUES(@idCategoria, @idCliente)

END

END

END

ELSE

BEGIN

PRINT 'El idPersona no está registrado en Persona'

END

END

ELSE

BEGIN

PRINT 'El idCategoria no está registrado en Categoria'

END

END

/\*

Nombre del procedimiento: paActualizaSolicitud

Propósito: Actualiza o inserta información de una solicitud en la base de datos.

Parámetros de entrada:

- @idSolicitud: Identificador único de la solicitud.

- @idCliente: Identificador único del cliente asociado a la solicitud.

- @idEmpleado: Identificador único del empleado asociado a la solicitud.

- @fechaHora: Fecha y hora de la solicitud.

- @descripcion: Descripción de la solicitud.

\*/

CREATE PROCEDURE paActualizaSolicitud

(

@idSolicitud SMALLINT,

@idCliente SMALLINT,

@idEmpleado SMALLINT,

@fechaHora DATETIME,

@descripcion VARCHAR(250)

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Cliente WHERE idCliente = @idCliente)

BEGIN

IF EXISTS(SELECT 1 FROM Empleado WHERE idEmpleado = @idEmpleado)

BEGIN

IF EXISTS(SELECT 1 FROM Solicitud WHERE idSolicitud = @idSolicitud)

BEGIN

-- Actualización de la solicitud

UPDATE Solicitud

SET idCliente = @idCliente,

idEmpleado = @idEmpleado,

fechaHora = @fechaHora,

descripcion = @descripcion

WHERE idSolicitud = @idSolicitud

END

ELSE

BEGIN

-- Inserción de una nueva solicitud

INSERT INTO Solicitud(idCliente, idEmpleado, fechaHora, descripcion)

VALUES(@idCliente, @idEmpleado, @fechaHora, @descripcion)

END

END

ELSE

BEGIN

PRINT 'El idEmpleado no está registrado en Empleado'

END

END

ELSE

BEGIN

PRINT 'El idCliente no está registrado en Cliente'

END

END

/\*

Nombre del procedimiento: paAcualizaTipoServicio

Propósito: Actualiza o inserta información de un tipo de servicio en la base de datos.

Parámetros de entrada:

- @idTipoServicio: Identificador único del tipo de servicio.

- @nombre: Nombre del tipo de servicio.

\*/

CREATE PROCEDURE paAcualizaTipoServicio

(

@idTipoServicio tinyint,

@nombre varchar(100)

)

AS

BEGIN

IF NOT EXISTS ( SELECT 1 FROM TipoServicio WHERE nombre = @nombre)

BEGIN

IF EXISTS(SELECT 1 FROM TipoServicio WHERE idTipoServicio=@idTipoServicio)

BEGIN

-- Actualización del tipo de servicio

UPDATE TipoServicio

SET nombre = @nombre

WHERE idTipoServicio=@idTipoServicio

END

ELSE

BEGIN

-- Inserción de un nuevo tipo de servicio

INSERT INTO TipoServicio(nombre)

VALUES(@nombre)

END

END

ELSE

BEGIN

PRINT 'El tipo de servicio ya está registrado'

END

END

/\*

Nombre del procedimiento: paActualizaSolicitudServicio

Propósito: Actualiza o inserta información de una solicitud de servicio en la base de datos.

Parámetros de entrada:

- @idSolicitudServicio: Identificador único de la solicitud de servicio.

- @idSolicitud: Identificador único de la solicitud asociada.

- @idTipoServicio: Identificador único del tipo de servicio asociado.

\*/

CREATE PROCEDURE paActualizaSolicitudServicio

(

@idSolicitudServicio SMALLINT,

@idSolicitud SMALLINT,

@idTipoServicio TINYINT

)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Solicitud WHERE idSolicitud = @idSolicitud)

BEGIN

IF EXISTS(SELECT 1 FROM TipoServicio WHERE idTipoServicio = @idTipoServicio)

BEGIN

IF EXISTS(SELECT 1 FROM SolicitudServicio WHERE idSolicitudServicio = @idSolicitudServicio)

BEGIN

-- Actualización de la solicitud de servicio

UPDATE SolicitudServicio

SET idSolicitud = @idSolicitud,

idTipoServicio = @idTipoServicio

WHERE idSolicitudServicio = @idSolicitudServicio

END

ELSE

BEGIN

-- Inserción de una nueva solicitud de servicio

INSERT INTO SolicitudServicio(idSolicitud, idTipoServicio)

VALUES(@idSolicitud, @idSolicitudServicio)

END

END

ELSE

BEGIN

PRINT 'El idTipoServicio no está registrado en TipoServicio'

END

END

ELSE

BEGIN

PRINT 'El idSolicitud no está registrado en Solicitud'

END

END

/\*

Nombre del procedimiento: paActualizaPropuesta

Propósito: Actualiza o inserta información de una propuesta en la base de datos.

Parámetros de entrada:

- @idPropuesta: Identificador único de la propuesta.

- @idSolicitud: Identificador único de la solicitud asociada a la propuesta.

- @fecha: Fecha de la propuesta.

- @hora: Hora de la propuesta.

- @costo: Costo de la propuesta.

\*/

CREATE PROCEDURE paActualizaPropuesta

@idPropuesta SMALLINT,

@idSolicitud SMALLINT,

@fecha DATE,

@hora TIME,

@costo NUMERIC(7,2)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Solicitud WHERE idSolicitud = @idSolicitud)

BEGIN

IF EXISTS(SELECT 1 FROM Propuesta WHERE idPropuesta=@idPropuesta)

BEGIN

-- Actualización de la propuesta

UPDATE Propuesta

SET idSolicitud = @idSolicitud,

fecha = @fecha,

hora = @hora,

costo = @costo

WHERE idPropuesta=@idPropuesta

END

ELSE

BEGIN

-- Inserción de una nueva propuesta

INSERT INTO Propuesta(idSolicitud,fecha,hora,costo)

VALUES(@idSolicitud,@fecha,@hora,@costo)

END

END

ELSE

BEGIN

PRINT 'idSolicitud NO REGISTRADA EN Solicitud'

END

END

/\*

Nombre del procedimiento: paActualizaDAP

Propósito: Actualiza o inserta información de un detalle de artículo de una propuesta en la base de datos.

Parámetros de entrada:

- @idDetalleArticuloPropuesta: Identificador único del detalle de artículo de la propuesta.

- @idPropuesta: Identificador único de la propuesta asociada al detalle de artículo.

- @idArticulo: Identificador único del artículo asociado al detalle de artículo.

- @dimensiones: Dimensiones del artículo.

- @cantidad: Cantidad del artículo.

- @precioVenta: Precio de venta del artículo.

\*/

CREATE PROCEDURE paActualizaDAP

@idDetalleArticuloPropuesta SMALLINT,

@idPropuesta SMALLINT,

@idArticulo INT,

@dimensiones VARCHAR(50),

@cantidad SMALLINT,

@precioVenta NUMERIC(7,2)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Propuesta WHERE idPropuesta = @idPropuesta)

BEGIN

IF EXISTS(SELECT 1 FROM Articulo WHERE idArticulo = @idArticulo)

BEGIN

IF EXISTS(SELECT 1 FROM DetalleArticuloPropuesta WHERE idDetalleArticuloPropuesta = @idDetalleArticuloPropuesta)

BEGIN

-- Actualización del detalle de artículo de la propuesta

UPDATE DetalleArticuloPropuesta

SET idPropuesta = @idPropuesta,

idArticulo = @idArticulo,

dimensiones = @dimensiones,

cantidad = @cantidad,

precioVenta = @precioVenta

WHERE idDetalleArticuloPropuesta = @idDetalleArticuloPropuesta

END

ELSE

BEGIN

-- Inserción de un nuevo detalle de artículo de la propuesta

INSERT INTO DetalleArticuloPropuesta(idPropuesta,idArticulo,dimensiones,cantidad,precioVenta)

VALUES(@idPropuesta,@idArticulo,@dimensiones,@cantidad,@precioVenta)

END

END

ELSE

BEGIN

PRINT 'idArticulo NO REGISTRADO en ARTICULO'

END

END

ELSE

BEGIN

PRINT 'idPropuesta NO REGISTRADO en Propuesta'

END

END

/\*

Nombre del procedimiento: paActualizaArticuloMP

Propósito: Actualiza o inserta información de un artículo de materia prima en la base de datos.

Parámetros de entrada:

- @idArticuloMateriaPrima: Identificador único del artículo de materia prima.

- @idArticulo: Identificador único del artículo asociado.

- @idMateriaPrima: Identificador único de la materia prima asociada.

- @cantidadMateriaPrima: Cantidad de la materia prima.

- @unidadMedida: Unidad de medida de la materia prima.

\*/

CREATE PROCEDURE paActualizaArticuloMP

@idArticuloMateriaPrima SMALLINT,

@idArticulo INT,

@idMateriaPrima SMALLINT,

@cantidadMateriaPrima INT,

@unidadMedida CHAR(10)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Articulo WHERE idArticulo = @idArticulo)

BEGIN

IF EXISTS(SELECT 1 FROM MateriaPrima WHERE idMateriaPrima = @idMateriaPrima)

BEGIN

IF EXISTS (SELECT 1 FROM ArticuloMateriaPrima

WHERE idArticuloMateriaPrima = @idArticuloMateriaPrima )

BEGIN

-- Actualización del artículo de materia prima

UPDATE ArticuloMateriaPrima

SET idArticulo = @idArticulo,

idMateriaPrima = @idMateriaPrima,

cantidadMateriaPrima = @cantidadMateriaPrima,

unidadMedida = @unidadMedida

WHERE idArticuloMateriaPrima = @idArticuloMateriaPrima

END

ELSE

BEGIN

-- Inserción de un nuevo artículo de materia prima

INSERT INTO ArticuloMateriaPrima(idArticulo, idMateriaPrima,

cantidadMateriaPrima, unidadMedida)

VALUES(@idArticulo,@idMateriaPrima,@cantidadMateriaPrima,@unidadMedida)

END

END

ELSE

BEGIN

PRINT 'idMateriaPrima NO REGISTRADA en MateriaPrima'

END

END

ELSE

BEGIN

PRINT 'idArticulo NO REGISTRA EN Articulo'

END

END

/\*

Nombre del procedimiento: paActualizaTipoComprobante

Propósito: Actualiza o inserta información de un tipo de comprobante en la base de datos.

Parámetros de entrada:

- @idTipoComprobante: Identificador único del tipo de comprobante.

- @nombre: Nombre del tipo de comprobante.

\*/

CREATE PROCEDURE paActualizaTipoComprobante

@idTipoComprobante TINYINT,

@nombre VARCHAR(50)

AS

BEGIN

IF NOT EXISTS(SELECT 1 FROM TipoComprobante WHERE nombre = @nombre)

BEGIN

IF EXISTS(SELECT 1 FROM TipoComprobante

WHERE idTipoComprobante = @idTipoComprobante )

BEGIN

-- Actualización del tipo de comprobante

UPDATE TipoComprobante

SET nombre = @nombre

WHERE idTipoComprobante = @idTipoComprobante

END

ELSE

BEGIN

-- Inserción de un nuevo tipo de comprobante

INSERT INTO TipoComprobante(nombre)

VALUES(@nombre)

END

END

END

/\*

Nombre del procedimiento: paActualizaTipoContrato

Propósito: Actualiza o inserta información de un tipo de contrato en la base de datos.

Parámetros de entrada:

- @idTipoContrato: Identificador único del tipo de contrato.

- @nombre: Nombre del tipo de contrato.

\*/

CREATE PROCEDURE paActualizaTipoContrato

@idTipoContrato TINYINT,

@nombre VARCHAR(30)

AS

BEGIN

IF NOT EXISTS(SELECT 1 FROM TipoContrato WHERE nombre = @nombre)

BEGIN

IF EXISTS(SELECT 1 FROM TipoContrato WHERE idTipoContrato = @idTipoContrato )

BEGIN

-- Actualización del tipo de contrato

UPDATE TipoContrato

SET nombre = @nombre

WHERE idTipoContrato = @idTipoContrato

END

ELSE

BEGIN

-- Inserción de un nuevo tipo de contrato

INSERT INTO TipoContrato(nombre)

VALUES(@nombre)

END

END

END

/\*

Nombre del procedimiento: paActualizaContrato

Propósito: Actualiza o inserta información de un contrato en la base de datos.

Parámetros de entrada:

- @idContrato: Identificador único del contrato.

- @idPropuesta: Identificador único de la propuesta asociada al contrato.

- @idTipoContrato: Identificador único del tipo de contrato asociado.

- @idEmpleado: Identificador único del empleado asociado al contrato.

- @fechaHora: Fecha y hora del contrato.

- @montoContrato: Monto del contrato.

- @montoAdelanto: Monto del adelanto del contrato.

- @tiempoRealizar: Tiempo estimado para realizar el contrato.

- @descripcion: Descripción del contrato.

- @estado: Estado del contrato.

\*/

CREATE PROCEDURE paActualizaContrato

@idContrato SMALLINT,

@idPropuesta SMALLINT,

@idTipoContrato TINYINT,

@idEmpleado SMALLINT,

@fechaHora DATETIME,

@montoContrato NUMERIC(5,2),

@montoAdelanto NUMERIC(5,2),

@tiempoRealizar TINYINT,

@descripcion VARCHAR(250),

@estado CHAR(1)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Propuesta WHERE idPropuesta=@idPropuesta)

BEGIN

IF EXISTS(SELECT 1 FROM TipoContrato WHERE idTipoContrato=@idTipoContrato)

BEGIN

IF EXISTS(SELECT 1 FROM Empleado WHERE idEmpleado=@idEmpleado)

BEGIN

IF NOT EXISTS(SELECT 1 FROM Contrato WHERE idPropuesta = @idPropuesta)

BEGIN

IF EXISTS(SELECT 1 FROM Contrato WHERE idContrato=@idContrato)

BEGIN

-- Actualización del contrato

UPDATE Contrato

SET idPropuesta = @idPropuesta,

idTipoContrato=@idTipoContrato,

idEmpleado=@idEmpleado,

fechaHora=@fechaHora,

montoContrato=@montoContrato,

montoAdelanto=@montoAdelanto,

tiempoRealizar=@tiempoRealizar,

descripcion=@descripcion,

estado=@estado

WHERE idContrato=@idContrato

END

ELSE

BEGIN

-- Inserción de un nuevo contrato

INSERT INTO Contrato

VALUES(@idPropuesta,@idTipoContrato,@idEmpleado,@fechaHora,@montoContrato,@montoAdelanto,@tiempoRealizar,@descripcion,'A')

END

END

ELSE

BEGIN

PRINT 'idPropuesta NO REGISTRADO EN LA TABALA PROPUESTA'

END

END

ELSE

BEGIN

PRINT 'idEmpleado NO REGISTRADO EN LA TABLA EMPLEADO'

END

END

ELSE

BEGIN

PRINT 'idContrato NO REGISTRADO EN LA TABLA CONTRATO'

END

END

ELSE

BEGIN

PRINT 'idPropuesta NO REGISTRADA EN LA TABLA PROPUESTA'

END

END

\*/

/\*

Nombre del procedimiento: paActualizaComprobante

Propósito: Actualiza o inserta información de un comprobante en la base de datos.

Parámetros de entrada:

- @idComprobante: Identificador único del comprobante.

- @idContrato: Identificador único del contrato asociado al comprobante.

- @idTipoComprobante: Identificador único del tipo de comprobante asociado.

- @nSerie: Número de serie del comprobante.

- @nComprobante: Número del comprobante.

- @fecha: Fecha del comprobante.

- @hora: Hora del comprobante.

- @subtotal: Subtotal del comprobante.

- @igv: IGV del comprobante.

- @total: Total del comprobante.

\*/

CREATE PROCEDURE paActualizaComprobante

@idComprobante INT,

@idContrato SMALLINT,

@idTipoComprobante TINYINT,

@nSerie CHAR(4),

@nComprobante CHAR(8),

@fecha DATE,

@hora TIME,

@subtotal NUMERIC(7,2),

@igv NUMERIC(7,2),

@total NUMERIC(7,2)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Contrato WHERE idContrato=@idContrato)

BEGIN

IF EXISTS(SELECT 1 FROM TipoComprobante WHERE idTipoComprobante=@idTipoComprobante)

BEGIN

IF NOT EXISTS(SELECT 1 FROM Comprobante WHERE idContrato=@idContrato)

BEGIN

IF EXISTS(SELECT 1 FROM Comprobante WHERE idComprobante=@idComprobante)

BEGIN

-- Actualización del comprobante

UPDATE Comprobante

SET idContrato=@idContrato,

idTipoComprobante=@idTipoComprobante,

nSerie=@nSerie,

nComprobante=@nComprobante,

fecha=@fecha,

hora=@hora,

subtotal=@subtotal,

igv=@igv,

total=@total

WHERE idComprobante=@idComprobante

END

ELSE

BEGIN

-- Inserción de un nuevo comprobante

INSERT INTO Comprobante

VALUES(@idContrato,@idTipoComprobante,@nSerie,@nComprobante,@fecha,@hora,@subtotal,@igv,@total)

END

END

ELSE

BEGIN

PRINT 'Ya se registro el comprobante de ese contrato'

END

END

ELSE

BEGIN

PRINT 'idTipoComprobante NO REGISTRADO EN TipoComprobante'

END

END

ELSE

BEGIN

PRINT 'idContrato NO REGISTRADO EN Contrato(NO EXISTE ESE CONTRATO'

END

END

\*/

/\*

Nombre del procedimiento: paActualizaDetalleComprobante

Propósito: Actualiza o inserta información de un detalle de comprobante en la base de datos.

Parámetros de entrada:

- @idDetalleComprobante: Identificador único del detalle de comprobante.

- @idArticulo: Identificador único del artículo asociado al detalle de comprobante.

- @idComprobante: Identificador único del comprobante asociado al detalle de comprobante.

- @cantidad: Cantidad del artículo.

- @precioVenta: Precio de venta del artículo.

- @subtotal: Subtotal del detalle de comprobante.

\*/

CREATE PROCEDURE paActualizaDetalleComprobante

@idDetalleComprobante INT,

@idArticulo INT,

@idComprobante INT,

@cantidad SMALLINT,

@precioVenta NUMERIC(7,2),

@subtotal NUMERIC(7,2)

AS

BEGIN

IF EXISTS(SELECT 1 FROM DetalleComprobante WHERE idDetalleComprobante=@idDetalleComprobante)

BEGIN

-- Actualización del detalle de comprobante

UPDATE DetalleComprobante

SET idArticulo=@idArticulo,

idComprobante=@idComprobante,

cantidad=@cantidad,

precioVenta=@precioVenta,

subtotal=@subtotal

WHERE idDetalleComprobante=@idDetalleComprobante

END

ELSE

BEGIN

-- Inserción de un nuevo detalle de comprobante

INSERT INTO DetalleComprobante

VALUES(@idArticulo,@idComprobante,@cantidad,@precioVenta,@subtotal,@idDetalleComprobante)

END

END

\*/

/\*

Nombre del procedimiento: paActualizaTipoPago

Propósito: Actualiza o inserta información de un tipo de pago en la base de datos.

Parámetros de entrada:

- @idTipoPago: Identificador único del tipo de pago.

- @nombre: Nombre del tipo de pago.

\*/

CREATE PROCEDURE paActualizaTipoPago

@idTipoPago TINYINT,

@nombre VARCHAR(50)

AS

BEGIN

IF NOT EXISTS(SELECT 1 FROM TipoPago WHERE nombre=@nombre)

BEGIN

IF EXISTS(SELECT 1 FROM TipoPago WHERE idTipoPago=@idTipoPago)

BEGIN

-- Actualización del tipo de pago

UPDATE TipoPago

SET nombre = @nombre

WHERE idTipoPago=@idTipoPago

END

ELSE

BEGIN

INSERT INTO TipoPago

VALUES(@nombre)

END

END

ELSE

BEGIN

PRINT 'Ya se encuentra registrado dicho TipoPago'

END

END

\*/

/\*

Nombre del procedimiento: paActualizaDetalleComprobanteTipoPago

Propósito: Actualiza o inserta información de un detalle de comprobante tipo pago en la base de datos.

Parámetros de entrada:

- @idDetalleComprobanteTipoPago: Identificador único del detalle de comprobante tipo pago.

- @idComprobante: Identificador único del comprobante asociado.

- @idTipoPago: Identificador único del tipo de pago asociado.

- @monto: Monto del detalle de comprobante tipo pago.

\*/

CREATE PROCEDURE paActualizaDetalleComprobanteTipoPago

@idDetalleComprobanteTipoPago SMALLINT,

@idComprobante INT,

@idTipoPago TINYINT,

@monto NUMERIC(7,2)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Comprobante WHERE idComprobante = @idComprobante)

BEGIN

IF EXISTS(SELECT 1 FROM TipoPago WHERE idTipoPago=@idTipoPago)

BEGIN

IF EXISTS(SELECT 1 FROM DetalleComprobanteTipoPago WHERE idDetalleComprobanteTipoPago=@idDetalleComprobanteTipoPago)

BEGIN

-- Actualización del detalle de comprobante tipo pago

UPDATE DetalleComprobanteTipoPago

SET idComprobante=@idComprobante,

idTipoPago=@idTipoPago,

monto=@monto

WHERE idDetalleComprobanteTipoPago=@idDetalleComprobanteTipoPago

END

ELSE

BEGIN

-- Inserción de un nuevo detalle de comprobante tipo pago

INSERT INTO DetalleComprobanteTipoPago

VALUES(@idComprobante,@idTipoPago,@monto)

END

END

ELSE

BEGIN

PRINT 'Tipo de pago -NO EXISTE'

END

END

ELSE

BEGIN

PRINT 'Comprobante -NO EXISTE'

END

END

Aquí tienes los procedimientos almacenados documentados y comentados:

```sql

-- ContratoEmpleado

/\*

Nombre del procedimiento: paActualizaContratoEmpleado

Propósito: Actualiza o inserta información de un contrato de empleado en la base de datos.

Parámetros de entrada:

- @idContratoEmpleado: Identificador único del contrato de empleado.

- @idContrato: Identificador único del contrato asociado.

- @idEmpleado: Identificador único del empleado asociado.

\*/

CREATE PROCEDURE paActualizaContratoEmpleado

@idContratoEmpleado SMALLINT,

@idContrato SMALLINT,

@idEmpleado SMALLINT

AS

BEGIN

IF EXISTS(SELECT 1 FROM Contrato WHERE idContrato = @idContrato)

BEGIN

IF EXISTS(SELECT 1 FROM Empleado WHERE idEmpleado = @idEmpleado)

BEGIN

IF EXISTS(SELECT 1 FROM ContratoEmpleado WHERE idContratoEmpleado = @idContratoEmpleado)

BEGIN

-- Actualización

UPDATE ContratoEmpleado

SET idContrato = @idContrato,

idEmpleado = @idEmpleado

WHERE idContratoEmpleado = @idContratoEmpleado

END

ELSE

BEGIN

-- Inserción

INSERT INTO ContratoEmpleado

VALUES(@idContrato, @idEmpleado)

END

END

ELSE

BEGIN

PRINT 'ERROR: El empleado especificado no existe.'

END

END

ELSE

BEGIN

PRINT 'ERROR: El contrato especificado no existe.'

END

END

-- Anulacion

/\*

Nombre del procedimiento: paActualizaAnulacion

Propósito: Actualiza o inserta información de una anulación en la base de datos.

Parámetros de entrada:

- @idAnulacion: Identificador único de la anulación.

- @fecha: Fecha de la anulación.

- @hora: Hora de la anulación.

- @idContrato: Identificador único del contrato asociado.

- @idEmpleado: Identificador único del empleado asociado.

\*/

CREATE PROCEDURE paActualizaAnulacion

@idAnulacion SMALLINT,

@fecha DATE,

@hora TIME,

@idContrato SMALLINT,

@idEmpleado SMALLINT

AS

BEGIN

IF EXISTS(SELECT 1 FROM Contrato WHERE idContrato = @idContrato)

BEGIN

IF EXISTS(SELECT 1 FROM Empleado WHERE idEmpleado = @idEmpleado)

BEGIN

IF NOT EXISTS(SELECT 1 FROM Anulacion WHERE idContrato = @idContrato)

BEGIN

IF EXISTS(SELECT 1 FROM Anulacion WHERE idAnulacion = @idAnulacion)

BEGIN

-- Actualización

UPDATE Anulacion

SET fecha = @fecha,

hora = @hora,

idContrato = @idContrato,

idEmpleado = @idEmpleado

WHERE idAnulacion = @idAnulacion

END

ELSE

BEGIN

-- Inserción

INSERT INTO Anulacion

VALUES(@fecha, @hora, @idContrato, @idEmpleado)

END

END

ELSE

BEGIN

PRINT 'ERROR: El contrato ya ha sido anulado.'

END

END

ELSE

BEGIN

PRINT 'ERROR: El empleado especificado no existe.'

END

END

ELSE

BEGIN

PRINT 'ERROR: El contrato especificado no existe.'

END

END

-- ProveedorMateriaPrima

/\*

Nombre del procedimiento: paActualizaProveedorMateriaPrima

Propósito: Actualiza o inserta información de un proveedor de materia prima en la base de datos.

Parámetros de entrada:

- @idProveedorMateriaPrima: Identificador único del proveedor de materia prima.

- @idProveedor: Identificador único del proveedor asociado.

- @idMateriaPrima: Identificador único de la materia prima asociada.

- @precioMateriaPrima: Precio de la materia prima.

- @fechaCompra: Fecha de compra de la materia prima.

- @cantidad: Cantidad de materia prima comprada.

- @unidadMedida: Unidad de medida de la materia prima.

\*/

CREATE PROCEDURE paActualizaProveedorMateriaPrima

@idProveedorMateriaPrima INT,

@idProveedor SMALLINT,

@idMateriaPrima SMALLINT,

@precioMateriaPrima NUMERIC(5,2),

@fechaCompra DATE,

@cantidad SMALLINT,

@unidadMedida VARCHAR(20)

AS

BEGIN

IF EXISTS(SELECT 1 FROM Proveedor WHERE idProveedor = @idProveedor)

BEGIN

IF EXISTS(SELECT 1 FROM MateriaPrima WHERE idMateriaPrima = @idMateriaPrima)

BEGIN

IF EXISTS(SELECT 1 FROM ProveedorMateriaPrima WHERE idProveedorMateriaPrima = @idProveedorMateriaPrima)

BEGIN

-- Actualización

UPDATE ProveedorMateriaPrima

SET idProveedor = @idProveedor,

idMateriaPrima = @idMateriaPrima,

precioMateriaPrima = @precioMateriaPrima,

fechaCompra = @fechaCompra,

cantidad = @cantidad,

unidadMedida = @unidadMedida

WHERE idProveedorMateriaPrima = @idProveedorMateriaPrima

END

ELSE

BEGIN

-- Inserción

INSERT INTO ProveedorMateriaPrima

VALUES(@idProveedor, @idMateriaPrima, @precioMateriaPrima, @fechaCompra, @cantidad, @unidadMedida)

END

END

ELSE

BEGIN

PRINT 'ERROR: La materia prima especificada no existe.'

END

END

ELSE

BEGIN

PRINT 'ERROR: El proveedor especificado no existe.'

END

END

PROCEDIMIENTO CON CURSORES

-- Procedimiento almacenado: pa\_muestra\_Articulo\_xComprobante

-- Propósito: Mostrar los detalles de los artículos vendidos en los comprobantes de una fecha específica.

-- Parámetros de entrada:

-- - @fecha: Fecha para la cual se desean mostrar los detalles de los artículos vendidos, especificamente por mes .

CREATE PROCEDURE pa\_muestra\_Articulo\_xComprobante

@fecha DATE

AS

BEGIN

-- Verificar si existen comprobantes para la fecha especificada

IF EXISTS (SELECT 1 FROM Comprobante WHERE MONTH(fecha) = MONTH(@fecha))

BEGIN

-- Declarar variables para almacenar información del artículo

DECLARE @nombre VARCHAR(100), @precio NUMERIC(7,2);

-- Declarar el cursor para obtener los detalles de los artículos vendidos en los comprobantes

DECLARE c\_articulo CURSOR FOR

SELECT a.nombre, a.precio

FROM DetalleComprobante c

JOIN Comprobante d ON d.idComprobante = c.idComprobante

JOIN Articulo a ON a.idArticulo = d.idArticulo

WHERE MONTH(d.fecha) = MONTH(@fecha);

-- Abrir el cursor

OPEN c\_articulo;

-- Recorrer fila por fila los detalles de los artículos vendidos

FETCH NEXT FROM c\_articulo INTO @nombre, @precio;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Mostrar la información del artículo

PRINT 'Articulo: ' + @nombre + ' Precio: ' + CAST(@precio AS VARCHAR(50));

FETCH NEXT FROM c\_articulo INTO @nombre, @precio;

END;

-- Cerrar y liberar el cursor

CLOSE c\_articulo;

DEALLOCATE c\_articulo;

END

ELSE

BEGIN

-- Mostrar mensaje si no se encontraron comprobantes para la fecha especificada

PRINT 'No se encontraron artículos vendidos para la fecha: ' + CAST(@fecha AS VARCHAR(10));

END;

END;

PROCEDIMIENTO CON TRANSACCIONES

---FALTA :C

FUNCIONES ESCALARES

-- Función: datosCliente

-- Descripción: Obtiene los datos completos de un cliente a partir de su ID, incluyendo nombre, apellidos y tipo de persona.

-- Parámetros de entrada:

-- @idCliente: ID del cliente del cual se desean obtener los datos.

-- Retorna: Una cadena de caracteres que contiene el mensaje de cliente encontrado o no encontrado junto con los datos del cliente si se encuentra.

CREATE FUNCTION dbo.datosCliente(@idCliente smallint)

RETURNS VARCHAR(MAX)

AS

BEGIN

-- Declaración de variables

DECLARE @nombre VARCHAR(50), @apellidoPaterno VARCHAR(50), @apellidoMaterno VARCHAR(50), @idPersona smallint, @mensaje varchar(50);

-- Verifica la existencia del cliente

IF EXISTS(SELECT 1 FROM Cliente WHERE idCliente = @idCliente) BEGIN

-- Se establece mensaje de cliente encontrado

SET @mensaje = 'CLIENTE ENCONTRADO';

-- Se guarda el idPersona del cliente que coincida con el idCliente pasado como parámetro

SELECT @idPersona=idPersona

FROM Cliente

WHERE idCliente = @idCliente;

-- En caso exista ese idPersona en la tabla PersonaNatural, se toman dichos datos

IF EXISTS(SELECT 1 FROM PersonaNatural WHERE idPersona = @idPersona) BEGIN

SELECT @nombre = nombre, @apellidoPaterno = apPat, @apellidoMaterno = apMat

FROM PersonaNatural

WHERE idPersona = @idPersona;

END

-- Caso contrario, no existe, quiere decir que es una persona jurídica, por tanto se toman dichos datos

ELSE BEGIN

SELECT @nombre = razonSocial

FROM PersonaJuridica

WHERE idPersona = @idPersona;

END

END

-- Caso contrario, el cliente no existe, se establece un mensaje para el usuario

ELSE BEGIN

SET @mensaje = 'CLIENTE NO ENCONTRADO';

END

-- Concatenar los datos para obtener los datos completos

RETURN CONCAT(@mensaje, ': ', @nombre, ' ', @apellidoPaterno, ' ', @apellidoMaterno);

END;

-- Función: TotalVentasPorDia

-- Descripción: Obtiene el total de ventas por día a partir de una fecha dada.

-- Parámetros de entrada:

-- @fecha: La fecha para la cual se desea obtener el total de ventas.

-- Retorna: Un valor decimal que representa el total de ventas para la fecha especificada.

CREATE FUNCTION dbo.TotalVentasPorDia(@fecha DATE)

RETURNS DECIMAL(10,2)

AS

BEGIN

-- Declarar una variable para almacenar el total

DECLARE @totalVentas DECIMAL(10,2);

-- Selecciona la suma del 'total' de la tabla Comprobante y guárdala en la variable

SELECT @totalVentas = SUM(c.total)

FROM Comprobante c

-- Filtrando para cuando las fechas coincidan

WHERE c.fecha = @fecha;

-- Retorna el valor almacenado en la variable, es decir, retorna el total de ventas

RETURN @totalVentas;

END;

VISTAS

-- Vista: vContratosVigentes

-- Descripción: Muestra los contratos que se encuentran vigentes (estado = 'A').

-- Función: Esta vista proporciona una manera conveniente de visualizar los contratos activos,

-- lo que ayuda en la gestión de la información al proporcionar una lista clara de los contratos en curso.

CREATE VIEW vContratosVigentes

AS

SELECT

-- Selecciona el idContrato, la fechaHora del contrato, el monto del contrato y la descripción

c.idContrato,

c.fechaHora,

c.montoContrato,

c.descripcion,

-- Además, muestra el nombre del cliente asociado al contrato

CASE

WHEN EXISTS(SELECT 1 FROM PersonaNatural pn WHERE idPersona = p.idPersona) THEN

(SELECT pn.nombre FROM PersonaNatural pn WHERE pn.idPersona = p.idPersona)

WHEN EXISTS(SELECT 1 FROM PersonaJuridica pj WHERE idPersona = p.idPersona) THEN

(SELECT pj.razonSocial FROM PersonaJuridica pj WHERE pj.idPersona = p.idPersona)

END AS 'Cliente'

FROM Contrato c

-- Realiza inner joins entre las tablas para relacionar los datos necesarios

INNER JOIN Empleado e ON e.idEmpleado = c.idEmpleado

INNER JOIN Propuesta pp ON pp.idPropuesta = c.idPropuesta

INNER JOIN Solicitud s ON pp.idSolicitud = s.idSolicitud

INNER JOIN Cliente cc ON cc.idCliente = s.idCliente

INNER JOIN Persona p ON p.idPersona = cc.idPersona

-- Filtra los contratos por el estado 'A' (activo)

WHERE c.estado = 'A';

-- Selecciona todos los datos de la vista vContratosVigentes

SELECT \* FROM vContratosVigentes;

-- Elimina la vista vContratosVigentes

DROP VIEW vContratosVigentes;

-- Vista: vDetalleArticulosPorContrato

-- Descripción: Muestra el detalle de artículos vendidos por contrato.

-- Función: Esta vista permite visualizar qué artículos se han vendido más, lo que facilita la identificación de los productos más populares.

CREATE VIEW vDetalleArticulosPorContrato

AS

SELECT

-- Selecciona el nombre del artículo y la cantidad vendida del mismo

a.nombre AS 'Nombre del artículo',

SUM(dc.cantidad) AS 'Cantidad vendida'

-- Los datos se obtienen de la tabla DetalleComprobante

FROM DetalleComprobante dc

-- Realiza inner joins con las tablas necesarias para relacionar los datos

INNER JOIN Comprobante c ON c.idComprobante = dc.idComprobante

INNER JOIN Articulo a ON a.idArticulo = dc.idArticulo

-- Agrupa los datos por el nombre del artículo

GROUP BY a.nombre;

-- Selecciona todos los datos de la vista vDetalleArticulosPorContrato y los ordena por la cantidad vendida en orden descendente

SELECT \*

FROM vDetalleArticulosPorContrato

ORDER BY "Cantidad vendida" DESC;

TRIGGERS

TRIGGERS AUDITORIA

-- Trigger para la tabla Articulo

/\*

Nombre del Trigger: tr\_Articulo

Propósito: Captura los cambios en la columna crítica 'precioUnitario' de la tabla Articulo y registra los detalles en la tabla de auditoría 'tAuditoriaArticulo'.

\*/

---Columna critica → precioUnitario del articulo

create table tAuditoriaArticulo(

id int,

idArticulo int,

fechaCambio date,

precioNuevo numeric,

precioAntiguo numeric,

Usuario varchar(50)

)

CREATE TRIGGER tr\_Articulo

ON Articulo

AFTER UPDATE

AS

BEGIN

-- Declarar variables para almacenar valores relevantes

DECLARE @idArticulo int, @precioNuevo numeric, @precioAntiguo numeric;

-- Obtener los valores actualizados y antiguos de precioUnitario

SELECT @idArticulo = i.idArticulo, @precioAntiguo = d.precioUnitario, @precioNuevo = i.precioUnitario

FROM inserted i

INNER JOIN deleted d ON i.idArticulo = d.idArticulo;

-- Verificar si el precioUnitario se ha actualizado

IF UPDATE(precioUnitario) BEGIN

-- Insertar los datos relevantes en la tabla de auditoría

INSERT INTO tAuditoriaArticulo

VALUES (@idArticulo, GETDATE(), @precioNuevo, @precioAntiguo, SYSTEM\_USER());

END;

END;

-- Trigger para la tabla Pagos

/\*

Nombre del Trigger: tr\_Pagos

Propósito: Captura los cambios en la columna crítica 'monto' de la tabla Pagos y registra los detalles en la tabla de auditoría 'tAuditoriaPagos'.

\*/

---Columna critica → monto de la tabla Pagos

create table tAuditoriaPagos(

id int,

idPago int,

idEmpleado smallint,

montoNuevo numeric,

montoAntiguo numeric,

fecha numeric,

Usuario varchar(50)

)

CREATE TRIGGER tr\_Pagos

ON Pagos

AFTER UPDATE

AS

BEGIN

-- Declarar variables para almacenar valores relevantes

DECLARE @idPago int, @idEmpleado numeric, @montoNuevo numeric, @montoAntiguo numeric;

-- Obtener los valores actualizados y antiguos de monto

SELECT @idPago = i.idPago, @montoNuevo = i.monto, @montoAntiguo = d.monto, @idEmpleado = i.idEmpleado

FROM inserted i

INNER JOIN deleted d ON i.idPago = d.idPago;

-- Verificar si el monto se ha actualizado

IF UPDATE(monto) BEGIN

-- Insertar los datos relevantes en la tabla de auditoría

INSERT INTO tAuditoriaPagos

VALUES (@idPago, @idEmpleado, @montoNuevo, @montoAntiguo, GETDATE(), SYSTEM\_USER());

END;

END;

TRIGGERS DE ACTUALIZACION

/\*

Nombre del Trigger: Actualizar\_entrada\_InventarioMateriaPrima

Propósito: Actualizar el inventario de materia prima cuando se inserta una nueva entrada en la tabla ProveedorMateriaPrima.

Explicación: Este trigger se dispara después de una inserción en la tabla ProveedorMateriaPrima. Su objetivo es actualizar el inventario de materia prima sumando la cantidad de materia prima que ha entrado en el almacén. Esto garantiza que el inventario se mantenga actualizado y se refleje correctamente la cantidad disponible de materia prima en el sistema.

\*/

CREATE TRIGGER Actualizar\_entrada\_InventarioMateriaPrima

ON [ProveedorMateriaPrima]

AFTER Insert

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted i)

BEGIN

-- El artículo no ha sido vendido, actualizamos el inventario sumando la cantidad de entrada

declare @cantidad smallint,@idMateriaPrima smallint

select @cantidad = i.cantidad,

@idMateriaPrima=i.idMateriaPrima

from inserted i

if exists ( select 1 from InventarioMateriaPrima where idMateriaPrima=@idMateriaPrima)--si existe entonces actualizacion

begin

UPDATE ip

SET ip.entradaMateria = ip.entradaMateria + @cantidad -- Corrección de la sintaxis, cambio de '+=' a '+'

FROM InventarioMateriaPrima ip

join inserted d on d.idInventarioMateriaPrima=ip.idInventarioMateriaPrima

end

else--sino existe ese idMateriaPrima en inventario entonces lo que hacemos es realizar insert

begin

insert into InventarioMateriaPrima(idMateriaPrima,entradaMateria)

values (@idMateriaPrima,@cantidad)

end

END;

END;

-- Trigger para actualizar la salida en el inventario de materia prima

/\*

Nombre del Trigger: tr\_DetalleComprobante

Propósito: Actualizar el inventario de materia prima cuando se inserta un nuevo detalle en la tabla DetalleComprobante.

Explicación: Este trigger se dispara después de una inserción en la tabla DetalleComprobante. Su objetivo es actualizar el inventario de materia prima restando la cantidad correspondiente a la salida de materia prima asociada a la venta de un artículo. Esto asegura que el inventario se mantenga actualizado y refleje correctamente la cantidad disponible de materia prima después de cada venta.

\*/

CREATE TRIGGER tr\_DetalleComprobante

ON DetalleComprobante

AFTER Insert

AS

BEGIN

declare @idArticulo int,@cantidad smallint,@cantidadMateriaPrima int,@idMateriaPrima smallint,

@cant int,@id smallint;

select @idArticulo=idArticulo,@cantidad=cantidad

from inserted

DECLARE cursor\_articulo CURSOR FOR

SELECT idMateriaPrima, cantidadMateriaPrima

FROM ArticuloMateriaPrima

WHERE idArticulo = @idArticulo

OPEN cursor\_articulo

FETCH NEXT FROM cursor\_articulo

INTO @idMateriaPrima, @cantidadMateriaPrima

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @id = @idMateriaPrima, @cant = @cantidadMateriaPrima

FETCH NEXT FROM cursor\_articulo

INTO @idMateriaPrima, @cantidadMateriaPrima

UPDATE InventarioMateriaPrima

set salida = salida + @cantidadMateriaPrima

where idMateriaPrima = @idMateriaPrima

FETCH NEXT FROM cursor\_articulo

INTO @idMateriaPrima, @cantidadMateriaPrima

END

CLOSE cursor\_articulo

DEALLOCATE cursor\_articulo

END