**“AÑO DE LA UNIDAD, LA PAZ Y EL DESARROLLO”**

**UNIVERSIDAD NACIONAL DE PIURA**

FACULTAD DE 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
* PEÑA CORDOVA, BREYNER ANTONIO
* PASACHE PIZARRO, CRISTIAN JOEL

**PIURA – PERÚ**

2023

**Contenido**

[I. MARCO REFERENCIAL SOBRE LA EMPRESA 3](#_Toc153669723)

[II. REQUERIMIENTOS 5](#_Toc153669724)

[2.1. Requerimientos Funcionales 5](#_Toc153669725)

[2.2. Requerimientos No Funcionales 6](#_Toc153669726)

[III. DISEÑO DEL SISTEMA 7](#_Toc153669727)

[3.1. Reconocimiento de entidades 7](#_Toc153669728)

[3.2. Descripción de Entidades 8](#_Toc153669729)

[3.3. Modelo Relacional 16](#_Toc153669730)

[IV. Roles y Funciones de cada Integrante 18](#_Toc153669731)

[V. ANEXOS 19](#_Toc153669732)

[5.1. Creación de Tablas 19](#_Toc153669733)

[5.2. Creación de Claves Foráneas 23](#_Toc153669734)

[5.3. Creación de Restricciones 26](#_Toc153669735)

[5.4. Creación de Índices 27](#_Toc153669736)

[5.5. Creación de Procedimientos Almacenados 29](#_Toc153669737)

[5.6. Creación de Funciones Escalares 40](#_Toc153669738)

[5.7. Creación de Vistas 42](#_Toc153669739)

[5.8. Creación de Auditorías y Triggers 44](#_Toc153669740)

[5.9. Creación de Transacción 46](#_Toc153669741)

[5.10. Creación de Cursor 47](#_Toc153669742)

# MARCO REFERENCIAL SOBRE LA EMPRESA

“CORAZÓN DE MARIA” 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 consta 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 llamado 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 entrega comprobante.

### ¿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 incluido al momento de hacer la cotización .

# 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** |
| **RF1** | 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. |
| **RF2** | 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. |
| **RF3** | 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. |
| **RF4** | 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. |
| **RF5** | 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. |
| **RF6** | 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** |
| **RNF1** | Usabilidad | El sistema debe ser fácil de usar tanto para los clientes como para los administradores de la empresa de confección. |
| **RNF2** | 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. |
| **RNF3** | 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. |
| **RNF4** | 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. |
| **RNF5** | 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. |

# DISEÑO DEL SISTEMA

DE AQUÍ EN ADELANTE NO ES PARTE DEL TRABAJO… :D

## Reconocimiento de entidades

|  |  |
| --- | --- |
| **ENTIDAD** | **ATRIBUTOS** |
| PersonaNatural | idPersNatural(PK), nombre, apPat, apMat, dni(U), idCliente(FK) |
| PersonjaJuridica | idPersJuridica(PK), razonSocial, ruc(U), idCliente(FK) |
| Cliente | idCliente(PK), idCategoria(FK), dirección, celular, correo |
| Solicitud | idSolicitud(PK), tipoServicio, fecha, hora, idpuntoPartida(FK),  idpuntoLlegada(FK), idCliente(FK) |
| Categoría | idCategoria(PK), nombre, descripción |
| Propuesta | idPropuesta(PK) , idSolicitud(FK), fecha, hora, costo |
| Contrato | idContrato(PK) , idPropuesta(FK), fecha, hora |
| Comprobante | idComprobante(PK), serieComprobante(U), numComprobante(U),  idTipoComprobante(FK), fecha, hora, igv, subtotal, total, idContrato(FK) |
| TipoComprobante | idTipoComprobante(PK), nombre |
| Servicio | idServicio(PK) , idContrato(FK), idAdmin(FK), fecha, hora |
| DetalleServicio | idDetalleServicio(PK), descripción, cantidad, tipo, estado,  idServicio(FK), idGuía(FK) |
| Anulación | idAnulacion(PK) , idContrato(FK), idAdministrador(FK),  montoDescuento, fecha, hora |
| Administrador | idAdmin(PK), idPersona(FK) |
| Guía | idGuia(PK), serieGuia(U), numGuia(U), fechaTraslado,  motivoTraslado, direccionPartida, direccionLlegada |
| Camión | idCamion(PK), descripción, pesoNeto, pesoBruto, capacidad\_TN,  numPlaca(U), modelo |
| EstibadorServicio | idEstServ(PK), idServicio(FK), idEstibador(FK) |
| Estibador | idEstibador(PK), idPersona(FK) |
| Conductor | idConductor(PK), idPersona(FK), numLicencia(U) |
| Persona | idPersona(PK), nombre, apPat, apMat, dni(U), celular, correo,  dirección |
| CamionServicio | idCamServ(PK), idServicio(FK), idCamion(FK), idConductor(FK) |

|  |  |
| --- | --- |
| Departamento | idDepartamento(PK), nombre, distancia(km) |
| Provincia | idProvincia(PK), nombre, idDepartamento(FK) |
| Distrito | idDistrito(PK), nombre, idProvincia(FK) |

## Descripción de Entidades

1. PersonaNatural:
   1. *Descripción:* Persona natural que requiere un servicio
   2. *Atributos:*
      1. idPersNatural (PK): Identificador unívoco de la persona Natural
      2. nombre: Nombre de la persona Natural
      3. apPat: Apellido materno de la persona Natural
      4. apMat: Apellido paterno de la persona Natural (opcional)
      5. dni (U): Documento Nacional de Identidad
      6. idCliente (FK): Identificador del Cliente asociada la Persona Natural
   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. razonSocial: Razón social de la persona Jurídica
      3. ruc (U): Número de Registro Único de Contribuyente
      4. idCliente (FK): Identificador del Cliente asociada 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 de mudanzas.
   2. *Atributos:*
      1. idCliente (PK): Identificador unívoco del cliente
      2. idCategoria (FK): Identificador de la Categoría asociada al cliente
      3. dirección: Dirección del cliente
      4. celular: Número de celular del cliente
      5. correo: Correo electrónico del cliente (opcional)
   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. 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. tipoServicio: Tipo de servicio que desea el cliente (transporte y/o embalaje)
      3. fecha: fecha de la solicitud
      4. hora: hora de la solicitud
      5. idpuntoPartida (FK): Identificador del departamento asociado al punto de partida de donde se hará el servicio.
      6. idpuntoLlegada (FK): Identificador del departamento asociado al punto de llegada hasta donde se trasladará el servicio.
      7. idCliente (FK): Identificador del cliente que ha realizado la solicitud
   3. *Propósito:* almacena información relacionada con las solicitudes realizadas por los clientes a la empresa, incluyendo tipo de servicio, fecha, hora, punto de partida, punto de llegada y la referencia del cliente que solicita.
5. 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.
6. 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.
7. Contrato:
   1. *Descripción:* Representa un contrato entre la empresa de mudanzas y un cliente.
   2. *Atributos:*
      1. idContrato (PK): Identificador unívoco del contrato.
      2. fecha: Fecha del contrato.
      3. hora: Hora del contrato.
      4. IdPropuesta (FK): Identificador de la propuesta asociada al contrato.
   3. *Propósito:* Almacena la información relacionada con los contratos realizados entre la empresa de mudanzas y los clientes, incluyendo la fecha, hora y la referencia a la propuesta aceptada.
8. Comprobante:
   1. *Descripción:* Representa un comprobante emitido para un contrato.
   2. *Atributos:*
      1. IdComprobante (PK): Identificador unívoco del Comprobante
      2. serieComprobante (U): Serie del comprobante.
      3. numComprobante (U): Número del comprobante.
      4. fecha: Fecha del comprobante.
      5. hora: Hora del comprobante.
      6. IGV: Impuesto General a las Ventas (18%) aplicado al total.
      7. subtotal: Subtotal del comprobante.
      8. total: Total del comprobante.
      9. idTipoComprobante (FK): Identificador de tipo de comprobante asociado a un comprobante.
      10. idContrato (FK): Identificador del contrato asociado al comprobante.
   3. *Propósito:* Almacena la información detallada de los comprobantes emitidos para los contratos, incluyendo la serie, número, fecha, hora, IGV, subtotal, total y la referencia al contrato y tipo de comprobante correspondiente.
9. 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:* Almacena los diferentes tipos de comprobante utilizados en la emisión de los comprobantes para los contratos.
10. Servicio:
    1. *Descripción:* Representa un servicio de mudanza asociado a un contrato.
    2. *Atributos:*
       1. idServicio (PK): Identificador unívoco del servicio.
       2. fecha: Fecha del servicio.
       3. hora: Hora del servicio.
       4. idContrato (FK): Identificador del contrato asociado al servicio.
       5. idAdministrador (FK): Identificador del administrador responsable del servicio.
    3. *Propósito:* Registra la información detallada de los servicios de mudanza asociados a los contratos, incluyendo la fecha, hora y las referencias a los elementos relacionados, como el camión, conductor y administrador asignados.
11. DetalleServicio:
    1. Descripción: Representa el detalle de cada servicio que se realiza donde se especifica el tipo de servicio, cantidad de productos transportadores y detalles de cada producto trasportado.
    2. Atributos:
       1. idDetalleServicio (PK): Identificador unívoco de cada línea del detalle del servicio
       2. descripción: Registra los detalles y características de cada producto
       3. cantidad: Indica la cantidad de productos que se están trasladando
       4. tipo: Indica el tipo de producto que se traslada
       5. estado: Descripción del estado en el cual se traslada el artículo.
       6. idServicio (FK): Identificador del servicio asociado al detalle del servicio
       7. idGuia (FK): Identificador de la guía asociada al detalle del servicio
    3. Propósito: Registrar el detalle de cada producto que se está trasladando en un servicio.
12. Anulación:
    1. *Descripción:* Representa una anulación o cancelación de un contrato.
    2. *Atributos:*
       1. idAnulación (PK): Identificador unívoco de la anulación.
       2. montoDescontado: Monto descontado por la anulación.
       3. idContrato (FK): Identificador del contrato anulado.
       4. idAdministrador (FK): Identificador del administrador responsable de la anulación.
       5. Fecha: Fecha en la que se realiza la anulación de un servicio.
       6. Hora: Hora en la que se realiza la anulación de un servicio.
    3. *Propósito:* Registra la información relacionada con las anulaciones o cancelaciones de contratos, incluyendo el monto descontado, el contrato anulado y el administrador responsable.
13. Administrador
    1. *Descripción:* Representa a los administradores encargados en la empresa.
    2. *Atributos:*
       1. idAministrador (PK): Identificador unívoco del administrador
       2. idPersona (FK): Identificador de los datos de la persona asociada a administrador
    3. *Propósito:* Registrar a los administradores que trabajan en la empresa, asignándoles su identificador y heredando sus datos desde la entidad persona de acuerdo a su id de Persona.
14. Guía
    1. *Descripción:* Representa a las guías de remisión generadas por cada servicio de mudanza que se realice.
    2. *Atributos:*
       1. IdGuia (PK): Identificador unívoco de la guía de remisión
       2. serieGuia (U): Representa la serie de la guía de remisión
       3. numGuia (U): Representa el número de la guía de remisión
       4. fechaTraslado: Indica la fecha en la que se realizará el servicio
       5. motivoTraslado: Representa el motivo o razón del traslado de los bienes
       6. direccionPartida: Indica la dirección desde donde partirá el servicio
       7. direccionLLegada: Indica la dirección hasta donde tendrá que llegar el servicio.
    3. *Propósito:* Registrar las guías de todos los servicios que se realizan para poder tener los datos del servicio, datos del conductor, del camión y otros detalles al momento del traslado.
15. Camión
    1. *Descripción:* Representa a los camiones con los que cuenta la empresa
    2. *Atributos:*
       1. idCamion (PK): Identificador unívoco de cada camión
       2. descripción: Detalla las características del camión (opcional)
       3. Modelo: Representa el modelo del camión
       4. pesoNeto\_KG: Representa el peso del contenedor del camión
       5. pesoBruto\_KG: Representa el peso total del camión
       6. capacidad\_TN: Indica cual es la carga máxima que puede transportar el camión
       7. numPlaca (U): Identifica el número de placa único de cada camión
    3. *Propósito:* Registrar todos los camiones con todos sus datos, numero de placa de cada uno, sus características y capacidad de cada camión.
16. EstibadorServicio
    1. *Descripción:* Representa a un estibador que participa en cada servicio
    2. *Atributos:*
       1. idEstServ (PK): Identificador unívoco de cada estibador correspondiente a un servicio
       2. idEstibador (FK): Identificador del estibador que participará en dicho servicio
       3. idServicio (FK): Identificador del servicio en el cual participó dicho estibador
    3. *Propósito:* Registrar que estibadores participan en los diferentes servicios que se realizan
17. Estibador
    1. *Descripción:* Representa a todos los estibadores que trabajan en la empresa
    2. *Atributos:*
       1. idEstibador (PK): Identificador unívoco de un estibador
       2. idPersona (FK): Identificador de los datos de la persona asociada a estibador
    3. *Propósito:* Registrar a los estibadores que trabajan en la empresa, asignándoles su identificador y heredando sus datos desde la entidad persona de acuerdo a su id de Persona.
18. Conductor
    1. *Descripción:* Representa a todos los conductores que trabajan en la empresa
    2. *Atributos:*
       1. idConductor (PK): Identificador unívoco de un conductor
       2. idPersona (FK): Identificador de los datos de la persona asociada a conductor
       3. numLicencia (U): Representa el número de licencia único de cada conductor
    3. *Propósito:* Registrar a los conductores que trabajan en la empresa, asignándoles su identificador y heredando sus datos desde la entidad persona de acuerdo a su id de Persona.
19. Persona
    1. Descripción: Representa todos los datos de una persona
    2. Atributos:
       1. idPersona (PK): Identificador unívoco de una persona
       2. nombre: Nombre de la persona
       3. apPat: Apellido paterno de la persona
       4. apMat: Apellido materno de la persona (opcional)
       5. dni (U): Número de Documento Nacional de Identidad
       6. celular: Número de celular de la persona
       7. correo: Correo electrónico de la persona (opcional)
       8. dirección: Dirección de una persona
    3. Propósito: Registra todos los datos de las personas que pertenecen a la empresa.
20. CamiónServicio:
    1. *Descripción:* Representa a un camión que participa en cada servicio
    2. *Atributos:*
       1. idCamServ (PK): Identificador unívoco de cada camión correspondiente a un servicio
       2. idCamión (FK): Identificador del camión que participará en dicho servicio
       3. idServicio (FK): Identificador del servicio en el cual participó dicho estibador
       4. idConductor (FK): Identificador del conductor encargado de trasladar el camión correspondiente a un servicio.
    3. *Propósito:* Registrar que camiones participan en los diferentes servicios que se realizan y los conductores que trasladarán dicho camión.
21. Departamento
    1. *Descripción:* Representa los departamentos del Perú en los cuales pueden realizar el servicio de mudanza.
    2. *Atributos:*
       1. idDepartamento (PK): Identificador unívoco de cada departamento.
       2. Nombre: representa el nombre del departamento.
       3. Distancia(km): representa la distancia en kilómetros desde Lima hasta otro departamento.
    3. *Propósito:* Registrar todos los departamentos a los cuales la empresa puede acceder para realizar el servicio de mudanza.
22. Provincia
    1. *Descripción:* Representa las provincias correspondientes a cada departamento
    2. *Atributos:*
       1. idProvincia (PK): Identificador unívoco de cada provincia
       2. Nombre: representa el nombre de la provincia
       3. idDepartamento (FK): Identificador del departamento asociado a una provincia
    3. *Propósito:* Registrar todos los departamentos a los cuales la empresa puede acceder para realizar el servicio de mudanza.
23. Distrito
    1. *Descripción:* Representa los distritos correspondientes a cada provincia
    2. *Atributos:*
       1. idDistrito (PK): Identificador unívoco de cada departamento
       2. Nombre: representa el nombre del departamento:



|  |  |
| --- | --- |
| **comprobante** | |
|  | idComprobante |
|  | seriecomprobante |
|  | numcomprobante |
|  | fecha |
|  | hora |
|  | igv |
|  | subtotal |
|  | total |
|  | idContrato |
|  | idTipoComprobante |
|  | |

|  |  |
| --- | --- |
| **anulacion** | |
|  | idAnulacion |
|  | montoDescuento |
|  | idContrato |
|  | idAdmin |
|  | fecha |
|  | hora |
|  | |

|  |  |
| --- | --- |
| **servicio** | |
|  | idServicio |
|  | fecha |
|  | hora |
|  | idContrato |
|  | idAdmin |
|  | |

|  |  |
| --- | --- |
| **estibador\_Servicio** | |
|  | idEstServ |
|  | idServicio |
|  | idEstibador |
|  | |

|  |  |
| --- | --- |
| **tipoComprobante** | |
|  | idTipoComprobante |
|  | nombre |
|  | |

|  |  |
| --- | --- |
| **detalleServicio** | |
|  | idDetalleServicio |
|  | descripcion |
|  | cantidad |
|  | tipo |
|  | estado |
|  | idServicio |
|  | idGuia |
|  | |

|  |  |
| --- | --- |
| **camion** | |
|  | idCamion |
|  | descripcion |
|  | pesoNeto\_KG |
|  | pesoBruto\_KG |
|  | capacidad\_TN |
|  | numPlaca |
|  | modelo |
|  | |

|  |  |
| --- | --- |
| **guia** | |
|  | idGuia |
|  | serieGuia |
|  | numGuia |
|  | fechaTraslado |
|  | motivoTraslado |
|  | direccionPartida |
|  | direccionLlegada |
|  | |

|  |  |
| --- | --- |
| **conductor** | |
|  | idConductor |
|  | idPersona |
|  | numLicencia |
|  | |

|  |  |
| --- | --- |
| **persona** | |
|  | idPersona |
|  | nombre |
|  | apPat |
|  | apMat |
|  | dni |
|  | celular |
|  | correo |
|  | direccion |
|  | |

* + 1. idProvincia (FK): Identificador de la provincia asociada a un distrito.
  1. *Propósito:* Registrar todos los departamentos a los cuales la empresa puede acceder para realizar el servicio de mudanza.

## Modelo Relacional



idCliente

dni

apMat

apPat

nombre

idPersNatural

**personaNatural**

|  |  |
| --- | --- |
| **personaJuridica** | |
|  | idPersJuridica |
|  | razonSocial |
|  | ruc |
|  | idCliente |
|  | |

|  |  |
| --- | --- |
| **departamento** | |
|  | idDepartamento |
|  | nombre |
|  | |

|  |  |
| --- | --- |
| **provincia** | |
|  | idProvincia |
|  | nombre |
|  | idDepartamento |
|  | |

|  |  |
| --- | --- |
| **categoria** | |
|  | idCategoria |
|  | nombre |
|  | descripcion |
|  | |

|  |  |
| --- | --- |
| **cliente** | |
|  | idCliente |
|  | direccion |
|  | celular |
|  | correo |
|  | idCategoria |
|  | |

|  |  |
| --- | --- |
| **solicitud** | |
|  | idSolicitud |
|  | tipoServicio |
|  | fecha |
|  | hora |
|  | idpuntoPartida |
|  | idpuntoLlegada |
|  | idCliente |
|  | |

|  |  |
| --- | --- |
| **distrito** | |
|  | idDistrito |
|  | nombre |
|  | idProvincia |
|  | |

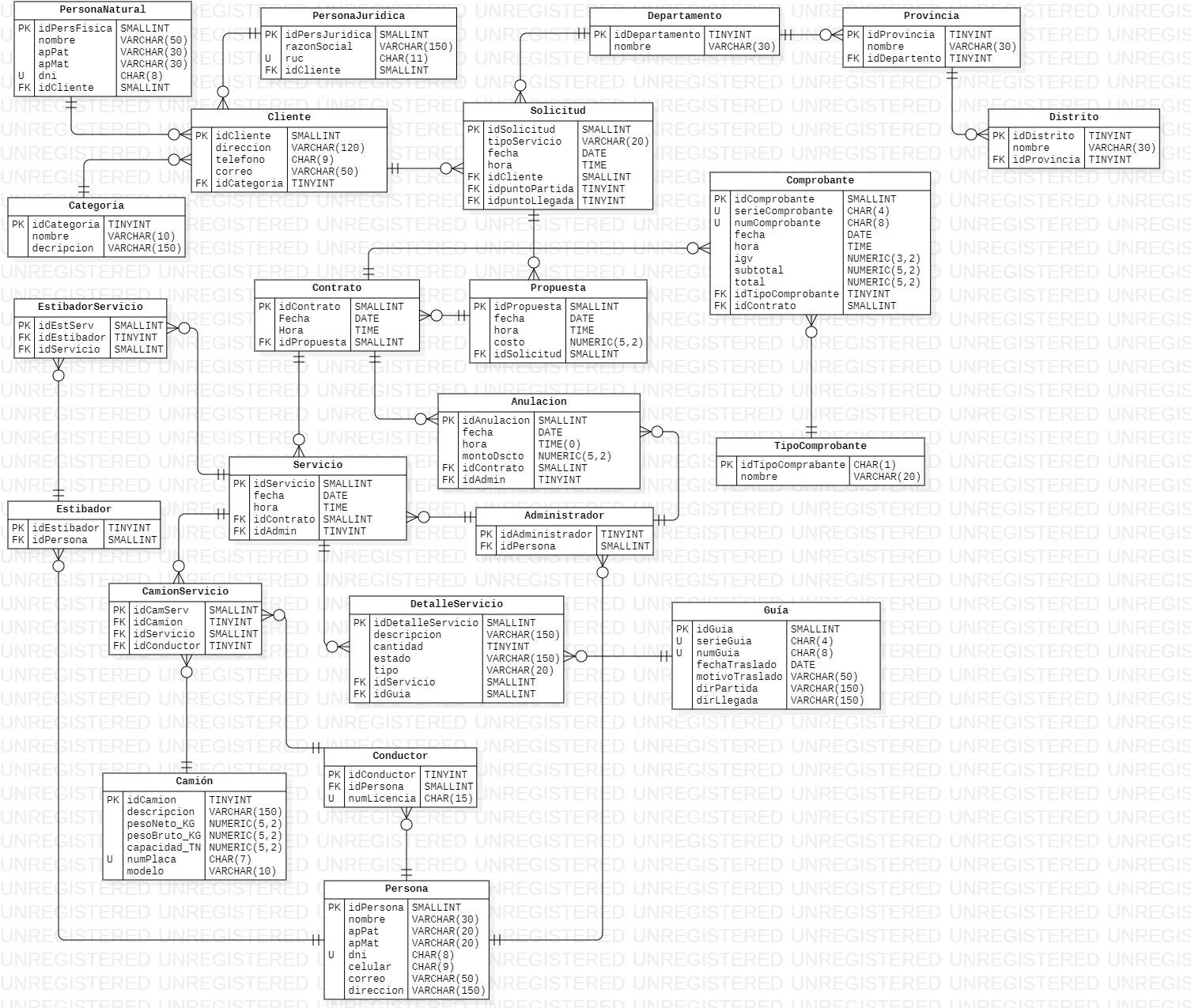
|  |  |
| --- | --- |
| **contrato** | |
|  | idContrato |
|  | fecha |
|  | hora |
|  | idPropuesta |
|  | |

|  |  |
| --- | --- |
| **propuesta** | |
|  | idPropuesta |
|  | fecha |
|  | hora |
|  | costo |
|  | idSolicitud |
|  | |

|  |  |
| --- | --- |
| **estibador** | |
|  | idEstibador |
|  | idPersona |
|  | |

|  |  |
| --- | --- |
| **camion\_Servicio** | |
|  | idCamServ |
|  | idCamion |
|  | idServicio |
|  | idConductor |
|  | |

|  |  |
| --- | --- |
| **administrador** | |
|  | idAdmin |
|  | idPersona |
|  | |



# Roles y Funciones de cada Integrante

HUANCA FLORES, SEGUNDO ELVIS

* + Descripción de entidades
  + Elaboración de diagrama relacional
  + Creación de tablas en la BD
  + Creación de procedimientos almacenados
  + Creación de vistas, triggers, funciones y transacciones GODOS VIERA, ANTHONY GERARDO
  + Descripción de entidades
  + Reconocimiento de entidades
  + Creación de tablas, índices y restricciones en la BD
  + Creación de procedimientos almacenados
  + Creación de vistas, triggers, funciones y transacciones PARIAHUACHE PEÑA, JUANA BISNEY
  + Listado de requerimientos
  + Descripción de entidades
  + Creación de tablas en la BD
  + Creación de procedimientos almacenados
  + Creación de vistas, triggers, funciones y transacciones

# ANEXOS

## Creación de Tablas

CREATE TABLE personaNatural(

idPersNatural SMALLINT IDENTITY(1,1) PRIMARY KEY, nombre VARCHAR(50) NOT NULL,

apPat VARCHAR(20) NOT NULL, apMat VARCHAR(20),

dni char(8) UNIQUE NOT NULL,

idCliente SMALLINT NOT NULL

)

CREATE TABLE personaJuridica(

idPersJuridica SMALLINT IDENTITY(1,1) PRIMARY KEY,

razonSocial VARCHAR(150) NOT NULL, ruc char(11) UNIQUE NOT NULL,

idCliente SMALLINT NOT NULL

)

CREATE TABLE cliente(

idCliente SMALLINT IDENTITY(1,1) PRIMARY KEY,

direccion VARCHAR(150) NOT NULL, celular CHAR(9) NOT NULL, correo VARCHAR(50),

idCategoria TINYINT NOT NULL

)

CREATE TABLE categoria(

idCategoria TINYINT IDENTITY(1,1) PRIMARY KEY, nombre VARCHAR(10) NOT NULL,

descripcion VARCHAR(150)

)

CREATE TABLE solicitud(

idSolicitud SMALLINT IDENTITY(1,1) PRIMARY KEY,

tipoServicio VARCHAR(20) NOT NULL, fecha DATE NOT NULL,

hora TIME(0) NOT NULL,

idpuntoPartida TINYINT NOT NULL, idpuntoLlegada TINYINT NOT NULL, idCliente SMALLINT NOT NULL

)

CREATE TABLE propuesta(

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

hora TIME(0) NOT NULL, costo NUMERIC(5,2),

idSolicitud SMALLINT NOT NULL

)

CREATE TABLE contrato(

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

hora TIME(0) NOT NULL,

idPropuesta SMALLINT NOT NULL

)

CREATE TABLE servicio(

idServicio SMALLINT IDENTITY(1,1) PRIMARY KEY, fecha DATE NOT NULL,

hora TIME(0) NOT NULL,

idContrato SMALLINT NOT NULL, idAdmin TINYINT NOT NULL,

)

CREATE TABLE detalleServicio(

idDetalleServicio SMALLINT IDENTITY(1,1) PRIMARY KEY, descripcion VARCHAR(150) NOT NULL,

cantidad DECIMAL(3,2) NOT NULL, tipo VARCHAR(10) NOT NULL, estado VARCHAR(150) NOT NULL,

idServicio SMALLINT NOT NULL, idGuia SMALLINT NOT NULL

)

CREATE TABLE administrador(

idAdmin TINYINT IDENTITY(1,1) PRIMARY KEY,

idPersona SMALLINT NOT NULL

)

CREATE TABLE conductor(

idConductor TINYINT IDENTITY(1,1) PRIMARY KEY,

idPersona SMALLINT NOT NULL, numLicencia CHAR(15) UNIQUE NOT NULL

)

CREATE TABLE estibador(

idEstibador TINYINT IDENTITY(1,1) PRIMARY KEY,

idPersona SMALLINT NOT NULL

)

CREATE TABLE persona(

idPersona SMALLINT IDENTITY(1,1) PRIMARY KEY, nombre VARCHAR(30) NOT NULL,

apPat VARCHAR(20) NOT NULL, apMat VARCHAR(20),

dni CHAR(8) UNIQUE NOT NULL,

celular CHAR(9) NOT NULL, correo VARCHAR(50),

direccion VARCHAR(150) NOT NULL,

)

CREATE TABLE camion(

idCamion TINYINT IDENTITY(1,1) PRIMARY KEY,

descripcion VARCHAR(150), pesoNeto\_KG DECIMAL(5,2), pesoBruto\_KG DECIMAL(5,2), capacidad\_TN DECIMAL(5,2), modelo VARCHAR(10),

numPlaca CHAR(7) UNIQUE NOT NULL

)

CREATE TABLE guia(

idGuia SMALLINT IDENTITY(1,1) PRIMARY KEY,

serieGuia CHAR(4) NOT NULL, numGuia CHAR(8) NOT NULL,

fechaTraslado DATETIME NOT NULL, motivoTraslado VARCHAR(50) NOT NULL, direccionPartida VARCHAR(20) NOT NULL, direccionLlegada VARCHAR(20) NOT NULL

)

CREATE TABLE estibador\_Servicio(

idEstServ SMALLINT IDENTITY(1,1) PRIMARY KEY,

idEstibador TINYINT NOT NULL, idServicio SMALLINT NOT NULL

)

CREATE TABLE comprobante(

idComprobante SMALLINT IDENTITY(1,1) PRIMARY KEY,

seriecomprobante CHAR(4) NOT NULL, numcomprobante CHAR(8) NOT NULL, idTipoComprobante CHAR(1) NOT NULL, fecha DATE NOT NULL,

hora TIME(0) NOT NULL, igv NUMERIC(5,2) NOT NULL,

subtotal NUMERIC(5,2) NOT NULL, total NUMERIC(5,2) NOT NULL,

idContrato SMALLINT NOT NULL

)

CREATE TABLE tipoComprobante( idTipoComprobante CHAR(1) PRIMARY KEY, nombre VARCHAR(20) NOT NULL

)

CREATE TABLE anulacion(

idAnulacion SMALLINT IDENTITY(1,1) PRIMARY KEY,

montoDescuento NUMERIC(5,2) NOT NULL, idContrato SMALLINT NOT NULL,

idAdmin SMALLINT NOT NULL, fecha DATE NOT NULL,

hora TIME(0) NOT NULL

)

CREATE TABLE camion\_Servicio(

idCamServ SMALLINT IDENTITY(1,1) PRIMARY KEY,

idCamion TINYINT NOT NULL, idServicio SMALLINT NOT NULL, idConductor TINYINT NOT NULL

)

CREATE TABLE departamento(

idDepartamento SMALLINT IDENTITY(1,1) PRIMARY KEY, nombre VARCHAR(50) NOT NULL

distancia(km) DDECIMAL(5,2) NOT NULL

)

CREATE TABLE provincia(

idProvincia SMALLINT IDENTITY(1,1) PRIMARY KEY,

nombre VARCHAR(50) NOT NULL,

idDepartamento SMALLINT NOT NULL) CREATE TABLE distrito(

idDistrito SMALLINT IDENTITY(1,1) PRIMARY KEY, nombre VARCHAR(50) NOT NULL,

idProvincia SMALLINT NOT NULL

)

## Creación de Claves Foráneas

* + 1. PersonaNatural

ALTER TABLE personaNatural

ADD CONSTRAINT FK\_Cliente\_PersNatural FOREIGN KEY (idCliente)

REFERENCES cliente(idCliente)

* + 1. PersonaJuridica

ALTER TABLE personaJuridica

ADD CONSTRAINT FK\_Cliente\_PersJuridica FOREIGN KEY (idCliente)

REFERENCES cliente(idCliente)

* + 1. Cliente

ALTER TABLE cliente

CONSTRAINT FK\_Categoria\_Cliente FOREIGN KEY (idCategoria) REFERENCES Categoria(idCategoria)

* + 1. Solicitud

ALTER TABLE solicitud

ADD CONSTRAINT FK\_Cliente\_Solicitud FOREIGN KEY (idCliente)

REFERENCES cliente(idCliente), CONSTRAINT FK\_Soli\_Partida FOREIGN KEY (idpuntoPartida)

REFERENCES departamento(idDepartamento), CONSTRAINT FK\_Soli\_Llegada

FOREIGN KEY (idpuntoLlegada)

REFERENCES departamento(idDepartamento)

* + 1. Propuesta

ALTER TABLE propuesta

ADD CONSTRAINT FK\_Solicitud\_Propuesta FOREIGN KEY (idSolicitud)

REFERENCES Solicitud(idSolicitud)

* + 1. Contrato

ALTER TABLE contrato

ADD CONSTRAINT FK\_Propuesta\_Contrato FOREIGN KEY (idPropuesta)

REFERENCES propuesta(idPropuesta)

* + 1. Comprobante

ALTER TABLE comprobante

ADD CONSTRAINT FK\_Contrato\_Comprobante FOREIGN KEY (idContrato)

REFERENCES contrato(idContrato), CONSTRAINT FK\_TipoComp\_Comprobante FOREIGN KEY (idTipoComprobante)

REFERENCES tipoComprobante(idTipoComprobante)

* + 1. Anulación

ALTER TABLE anulacion

ADD CONSTRAINT FK\_Contrato\_Anulacion FOREIGN KEY (idContrato)

REFERENCES contrato(idContrato), CONSTRAINT FK\_Admin\_Anulacion FOREIGN KEY (idAdmin)

REFERENCES administrador(idAdmin)

* + 1. Servicio

ALTER TABLE servicio

ADD CONSTRAINT FK\_Contrato\_Servicio FOREIGN KEY (idContrato)

REFERENCES contrato(idContrato), CONSTRAINT FK\_Admin\_Servicio FOREIGN KEY (idAdmin)

REFERENCES administrador(idAdmin)

* + 1. DetalleServicio

ALTER TABLE detalleServicio

ADD CONSTRAINT FK\_Guia\_DetalleServicio FOREIGN KEY (idGuia)

REFERENCES guia(idGuia),

CONSTRAINT FK\_Servicio\_DetalleServicio FOREIGN KEY (idServicio)

REFERENCES servicio(idServicio)

* + 1. Administrador

ALTER TABLE administrador

ADD CONSTRAINT FK\_Persona\_Admin FOREIGN KEY (idPersona) REFERENCES Persona(idPersona)

* + 1. Estibador

ALTER TABLE estibador

ADD CONSTRAINT FK\_Persona\_Estibador FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona)

* + 1. Conductor

ALTER TABLE conductor ADD CONSTRAINT FK\_Persona\_Conductor FOREIGN KEY (idPersona)

REFERENCES Persona(idPersona)

* + 1. CamionServicio

ALTER TABLE camion\_Servicio

ADD CONSTRAINT FK\_CamServ\_Camion FOREIGN KEY (idCamion) REFERENCES camion(idCamion), CONSTRAINT FK\_CamServ\_Servicio FOREIGN KEY (idServicio) REFERENCES servicio(idServicio), CONSTRAINT FK\_Conductor\_CamServ FOREIGN KEY (idConductor)

REFERENCES conductor(idConductor)

* + 1. Provincia

ALTER TABLE provincia ADD CONSTRAINT FK\_Dep\_Prov FOREIGN KEY (idDepartamento)

REFERENCES departamento(idDepartamento)

* + 1. Distrito

ALTER TABLE distrito ADD CONSTRAINT FK\_Prov\_Dist FOREIGN KEY (idProvincia)

REFERENCES provincia(idProvincia)

## Creación de Restricciones

* + 1. PersonaNatural

ALTER TABLE personaNatural

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

CONSTRAINT CK\_dniOcho CHECK(len(dni) = 8)

* + 1. PersonaJuridica

ALTER TABLE personaJuridica

ADD CONSTRAINT CK\_rucNumeros CHECK(ruc LIKE '%[0-9]'),

CONSTRAINT CK\_rucOnce CHECK(len(ruc) = 11)

* + 1. Cliente

ALTER TABLE cliente

ADD CONSTRAINT CK\_telCliente CHECK(telefono LIKE '%[0-9]')

* + 1. Propuesta

ALTER TABLE propuesta

ADD CONSTRAINT CK\_costoPropuesta CHECK(costo > 0)

* + 1. Persona

ALTER TABLE persona

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

CONSTRAINT CK\_celPersona CHECK(celular LIKE '%[0-9]'), CONSTRAINT CK\_dniPersonaOcho CHECK(len(dni) = 8)

* + 1. Camión

ALTER TABLE camion

ADD CONSTRAINT CK\_pesoNetoCamion CHECK(pesoNeto\_KG > 0), CONSTRAINT CK\_pesoBrutoCamion CHECK(pesoBruto\_KG > 0), CONSTRAINT CK\_capacidadCamion CHECK(capacidad\_TN > 0)

* + 1. Guía

ALTER TABLE guia

ADD CONSTRAINT UQ\_guia UNIQUE(serieGuia,numGuia)

* + 1. Comprobante

ALTER TABLE comprobante

ADD CONSTRAINT UQ\_comprobante UNIQUE(serieComprobante,numComprobante), CONSTRAINT CK\_igvComprobante CHECK(igv > 0),

CONSTRAINT CK\_subtotalComprobante CHECK(subtotal > 0), CONSTRAINT CK\_totalComprobante CHECK(total > igv + subtotal), CONSTRAINT CK\_igv18 CHECK(igv = total - subtotal),

CONSTRAINT CK\_subtotaligv CHECK(subtotal = total / 1.18)

## 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:

* + 1. Índice de la tabla personaFísica:

Creado en la columna de apellido paterno (apPat) para que los apellidos siempre se muestren en orden desde la A hasta la Z.

CREATE NONCLUSTERED INDEX IDX\_Apellido

ON personaFisica(apPat)

* + 1. Índice de la tabla personaJuridica:

Creado en la columna Razón Social (razonSocial) para que los nombres siempre se muestren en orden desde la A hasta la Z.

CREATE NONCLUSTERED INDEX IDX\_RazonSocial

ON personaJuridica(razonSocial)

* + 1. Índice de la tabla persona:

Creado en la columna de apellido paterno (apPat) para que los apellidos siempre se muestren en orden desde la A hasta la Z.

CREATE NONCLUSTERED INDEX IDX\_ApellidoPersona ON persona(apPat)

* + 1. Índice de la tabla pago:

Creado en la columna Fecha (fecha) para que los pagos se muestren en forma descendente desde el último pago realizado hasta el primero.

CREATE NONCLUSTERED INDEX IDX\_FechaPago

ON pago(fecha DESC)

* + 1. Índice de la tabla anulación:

Creado en la columna Fecha (fecha) para que las anulaciones se muestren en forma descendente desde la última anulación realizada hasta la primera.

CREATE NONCLUSTERED INDEX IDX\_FechaAnulacion ON anulacion(fecha DESC)

* + 1. Índice de la tabla comprobante:

Creado en la columna Fecha (fecha) para que los comprobantes se muestren en forma descendente desde el último comprobante emitido hasta el primero.

CREATE NONCLUSTERED INDEX IDX\_FechaComprobante

ON comprobante(fecha DESC)

* + 1. Índice de la tabla contrato:

Creado en la columna Fecha (fecha) para que los contratos se muestren en forma descendente desde el último contrato realizado hasta el primero.

CREATE NONCLUSTERED INDEX IDX\_FechaContrato ON contrato(fecha DESC)

* + 1. Índice de la tabla detalleServicio:

Creado en la columna Identificador de Servicio (idServicio) para que los detalles se muestren ordenados de acuerdo al identificador de servicio correspondiente.

CREATE NONCLUSTERED INDEX IDX\_DetalleServicio ON detalleServicio(idServicio DESC)

* + 1. Índice de la tabla pago:

Creado en la columna Fecha (fecha) para que las propuestas se muestren en forma descendente desde la última propuesta realizada hasta la primera.

CREATE NONCLUSTERED INDEX IDX\_FechaPropuesta ON propuesta(fecha DESC)

* + 1. Índice de la tabla servicio:

Creado en la columna Fecha (fecha) para que los servicios se muestren en forma descendente desde el último servicio realizado hasta el primero.

CREATE NONCLUSTERED INDEX IDX\_FechaServicio ON servicio(fecha DESC)

* + 1. Índice de la tabla solicitud:

Creado en la columna Fecha (fecha) para que las solicitudes se muestren en forma descendente desde la última solicitud realizada hasta la primera.

CREATE NONCLUSTERED INDEX IDX\_FechaSolicitud ON solicitud(fecha DESC)

## Creación de Procedimientos Almacenados

* + 1. Administrador

CREATE PROCEDURE paActualizaAdministrador

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idAdmin TINYINT = NULL, @idPersona SMALLINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I'

BEGIN

END

INSERT INTO administrador (idPersona) VALUES (@idPersona);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ADMINISTRADOR'

ELSE IF @accion = 'U'

BEGIN

UPDATE administrador

SET idPersona = @idPersona WHERE idAdmin = @idAdmin;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idAdmin) +' EN LA TABLA ADMINISTRADOR'

END

ELSE

END;

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) --> ACTUALIZAR'

* + 1. Anulación

CREATE PROCEDURE paActualizaAnulacion

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idAnulacion SMALLINT = NULL,

@montoDescuento NUMERIC(5, 2), @idContrato SMALLINT,

@idAdmin TINYINT, @fecha DATE, @hora TIME(0)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

hora)

END

INSERT INTO anulacion (montoDescuento, idContrato, idAdmin, fecha,

VALUES (@montoDescuento, @idContrato, @idAdmin, @fecha, @hora); PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ANULACIÓN'

ELSE IF @accion = 'U' BEGIN

UPDATE anulacion

SET montoDescuento = @montoDescuento, idContrato = @idContrato, idAdmin = @idAdmin,

fecha = @fecha, hora = @hora

WHERE idAnulacion = @idAnulacion; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idAnulacion) +' EN LA TABLA ANULACIÓN' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) --> ACTUALIZAR'

END;

* + 1. Camión

CREATE PROCEDURE paActualizaCamion

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idCamion TINYINT = NULL,

@descripcion VARCHAR(150), @pesoNeto\_KG NUMERIC(5, 2),

@pesoBruto\_KG NUMERIC(5, 2),

@capacidad\_TN NUMERIC(5, 2), @numPlaca CHAR(7),

@modelo VARCHAR(10)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

INSERT INTO camion (descripcion, pesoNeto\_KG, pesoBruto\_KG, capacidad\_TN, numPlaca, modelo)

VALUES (@descripcion, @pesoNeto\_KG, @pesoBruto\_KG, @capacidad\_TN, @numPlaca, @modelo);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CAMIÓN'

END

ELSE IF @accion = 'U' BEGIN

UPDATE camion

SET descripcion = @descripcion, pesoNeto\_KG = @pesoNeto\_KG, pesoBruto\_KG = @pesoBruto\_KG, capacidad\_TN = @capacidad\_TN, numPlaca = @numPlaca,

modelo = @modelo WHERE idCamion = @idCamion;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idCamion)

+' EN LA TABLA CAMIÓN' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Camión – Servicio

CREATE PROCEDURE paActualizaCamionServicio

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idCamServ SMALLINT = NULL,

@idCamion TINYINT, @idServicio SMALLINT, @idConductor TINYINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO camion\_Servicio (idCamion, idServicio, idConductor) VALUES (@idCamion, @idServicio, @idConductor);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CAMIÓN-SERVICIO'

ELSE IF @accion = 'U' BEGIN

UPDATE camion\_Servicio

SET idCamion = @idCamion, idServicio = @idServicio, idConductor = @idConductor

WHERE idCamServ = @idCamServ;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idCamServ) +' EN LA TABLA CAMIÓN-SERVICIO'

END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Categoría

CREATE PROCEDURE paActualizaCategoria

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idCategoria TINYINT = NULL,

@nombre VARCHAR(10), @descripcion VARCHAR(150)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO categoria (nombre, descripcion) VALUES (@nombre, @descripcion);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CATEGORÍA'

ELSE IF @accion = 'U' BEGIN

UPDATE categoria

SET nombre = @nombre,

descripcion = @descripcion WHERE idCategoria = @idCategoria; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idCategoria) +' EN LA TABLA CATEGORÍA' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Cliente

CREATE PROCEDURE paActualizaCliente

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idCliente SMALLINT = NULL,

@direccion VARCHAR(150), @celular CHAR(9), @correo VARCHAR(50), @idCategoria TINYINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO cliente (direccion, celular, correo, idCategoria) VALUES (@direccion, @celular, @correo, @idCategoria);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CLIENTE'

ELSE IF @accion = 'U' BEGIN

UPDATE cliente

SET direccion = @direccion,

celular = @celular, correo = @correo,

idCategoria = @idCategoria WHERE idCliente = @idCliente;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idCliente) +' EN LA TABLA CLIENTE'

END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Comprobante

CREATE PROCEDURE paActualizaComprobante

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idComprobante SMALLINT = NULL,

@seriecomprobante CHAR(4), @numcomprobante CHAR(8), @fecha DATE,

@hora TIME(0), @igv NUMERIC(5, 2),

@subtotal NUMERIC(5, 2),

@total NUMERIC(5, 2), @idContrato SMALLINT, @idTipoComprobante CHAR(1)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

INSERT INTO comprobante (seriecomprobante, numcomprobante, fecha, hora, igv, subtotal, total, idContrato, idTipoComprobante)

VALUES (@seriecomprobante, @numcomprobante, @fecha, @hora, @igv, @subtotal, @total, @idContrato, @idTipoComprobante);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA COMPROBANTE'

END

ELSE IF @accion = 'U' BEGIN

UPDATE comprobante

SET seriecomprobante = @seriecomprobante, numcomprobante = @numcomprobante, fecha = @fecha,

hora = @hora, igv = @igv,

subtotal = @subtotal, total = @total, idContrato = @idContrato,

idTipoComprobante = @idTipoComprobante WHERE idComprobante = @idComprobante;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idComprobante) +' EN LA TABLA COMPROBANTE'

END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Conductor

CREATE PROCEDURE paActualizaConductor

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idConductor TINYINT = NULL,

@idPersona SMALLINT,

@numLicencia CHAR(15)

AS

BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO conductor (idPersona, numLicencia) VALUES (@idPersona, @numLicencia);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CONDUCTOR'

ELSE IF @accion = 'U' BEGIN

UPDATE conductor

SET idPersona = @idPersona, numLicencia = @numLicencia

WHERE idConductor = @idConductor; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idConductor) +' EN LA TABLA CONDUCTOR' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Contrato

CREATE PROCEDURE paActualizaContrato

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idContrato SMALLINT = NULL,

@fecha DATE, @hora TIME(0),

@idPropuesta SMALLINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO contrato (fecha, hora, idPropuesta) VALUES (@fecha, @hora, @idPropuesta);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CONTRATO'

ELSE IF @accion = 'U' BEGIN

UPDATE contrato

SET fecha = @fecha,

hora = @hora,

idPropuesta = @idPropuesta WHERE idContrato = @idContrato; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idContrato) +' EN LA TABLA CONTRATO' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Detalle Servicio

CREATE PROCEDURE paActualizaDetalleServicio

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idDetalleServicio SMALLINT = NULL,

@descripcion VARCHAR(150), @cantidad DECIMAL(3, 2), @tipo VARCHAR(10),

@estado VARCHAR(150), @idServicio SMALLINT, @idGuia SMALLINT

AS

BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

INSERT INTO detalleServicio (descripcion, cantidad, tipo, estado, idServicio, idGuia)

VALUES (@descripcion, @cantidad, @tipo, @estado, @idServicio,

@idGuia);

END

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA DETALLESERVICIO'

ELSE IF @accion = 'U' BEGIN

UPDATE detalleServicio

SET descripcion = @descripcion, cantidad = @cantidad, tipo = @tipo,

estado = @estado, idServicio = @idServicio, idGuia = @idGuia

WHERE idDetalleServicio = @idDetalleServicio; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idDetalleServicio) +' EN LA TABLA DETALLESERVICIO' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Estibador

CREATE PROCEDURE paActulizaEstibador

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idEstibador TINYINT = NULL,

@idPersona SMALLINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO estibador (idPersona) VALUES (@idPersona);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ESTIBADOR'

ELSE IF @accion = 'U' BEGIN

UPDATE estibador

SET idPersona = @idPersona

WHERE idEstibador = @idEstibador; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idEstibador) +' EN LA TABLA ESTIBADOR' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Estibador – Servicio

CREATE PROCEDURE paActualizaEstibadorServicio

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar

@idEstServ SMALLINT, @idServicio SMALLINT, @idEstibador TINYINT

AS

BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

idEstibador)

END

INSERT INTO estibador\_Servicio (idEstServ, idServicio,

VALUES (@idEstServ, @idServicio, @idEstibador);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ESTIBADOR-SERVICIO'

ELSE IF @accion = 'U' BEGIN

UPDATE estibador\_Servicio

SET idServicio = @idServicio, idEstibador = @idEstibador

WHERE idEstServ = @idEstServ;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idEstServ) +' EN LA TABLA ESTIBADOR-SERVICIO'

END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Guía

CREATE PROCEDURE paActualizaGuia

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idGuia SMALLINT,

@serieGuia CHAR(4), @numGuia CHAR(8), @fechaTraslado DATETIME,

@motivoTraslado VARCHAR(50),

@direccionPartida VARCHAR(150), @direccionLlegada VARCHAR(150)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

INSERT INTO guia (idGuia, serieGuia, numGuia, fechaTraslado, motivoTraslado, direccionPartida, direccionLlegada)

VALUES (@idGuia, @serieGuia, @numGuia, @fechaTraslado, @motivoTraslado, @direccionPartida, @direccionLlegada);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA GUÍA'

END

ELSE IF @accion = 'U' BEGIN

UPDATE guia

SET serieGuia = @serieGuia, numGuia = @numGuia,

fechaTraslado = @fechaTraslado, motivoTraslado = @motivoTraslado, direccionPartida = @direccionPartida, direccionLlegada = @direccionLlegada

WHERE idGuia = @idGuia;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idGuia)

+' EN LA TABLA GUÍA'

END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) --> ACTUALIZAR'

END;

* + 1. Persona

CREATE PROCEDURE paActualizaPersona

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idPersona SMALLINT = NULL,

@nombre VARCHAR(30), @apPat VARCHAR(20),

@apMat VARCHAR(20) = NULL,

@dni CHAR(8), @celular CHAR(9),

@correo VARCHAR(50) = NULL,

@direccion VARCHAR(150)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

direccion) @direccion);

END

INSERT INTO persona (nombre, apPat, apMat, dni, celular, correo, VALUES (@nombre, @apPat, @apMat, @dni, @celular, @correo,

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PERSONA'

ELSE IF @accion = 'U' BEGIN

UPDATE persona

SET nombre = @nombre,

apPat = @apPat, apMat = @apMat, dni = @dni,

celular = @celular, correo = @correo, direccion = @direccion

WHERE idPersona = @idPersona;

PRINT 'SE ACTUALIZÓ EL REGISTRO ' + CONVERT(VARCHAR(10),@idPersona) +' EN LA TABLA PERSONA'

END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Persona Jurídica

CREATE PROCEDURE paActualizaPersonaJuridica

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idPersJuridica SMALLINT = NULL,

@razonSocial VARCHAR(150), @ruc CHAR(11),

@idCliente SMALLINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO personaJuridica (razonSocial, ruc, idCliente) VALUES (@razonSocial, @ruc, @idCliente);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PERSONA JURIDICA'

ELSE IF @accion = 'U'

BEGIN

UPDATE personaJuridica

SET razonSocial = @razonSocial, ruc = @ruc,

idCliente = @idCliente

WHERE idPersJuridica = @idPersJuridica; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idPersJuridica) +' EN LA TABLA PERSONA JURIDICA' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Persona Natural

CREATE PROCEDURE paActualizaNatural

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idPersNatural SMALLINT = NULL,

@nombre VARCHAR(50), @apPat VARCHAR(20),

@apMat VARCHAR(20) = NULL,

@dni CHAR(8), @idCliente SMALLINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO personaNatural (nombre, apPat, apMat, dni, idCliente) VALUES (@nombre, @apPat, @apMat, @dni, @idCliente);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PERSONA NATURAL'

ELSE IF @accion = 'U' BEGIN

UPDATE personaNatural SET nombre = @nombre,

apPat = @apPat, apMat = @apMat, dni = @dni,

idCliente = @idCliente

WHERE idPersNatural = @idPersNatural; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idPersNatural) +' EN LA TABLA PERSONA NATURAL' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Propuesta

CREATE PROCEDURE paActualizaPropuesta

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idPropuesta SMALLINT = NULL,

@fecha DATE, @hora TIME(0),

@costo NUMERIC(5, 2), @idSolicitud SMALLINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO propuesta (fecha, hora, costo, idSolicitud) VALUES (@fecha, @hora, @costo, @idSolicitud);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PROPUESTA'

ELSE IF @accion = 'U' BEGIN

UPDATE propuesta SET fecha = @fecha,

hora = @hora, costo = @costo,

idSolicitud = @idSolicitud WHERE idPropuesta = @idPropuesta; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idPropuesta) +' EN LA TABLA PROPUESTA' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Servicio

CREATE PROCEDURE paActualizaServicio

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idServicio SMALLINT = NULL,

@fecha DATE, @hora TIME(0),

@idContrato SMALLINT, @idAdmin TINYINT

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO servicio (fecha, hora, idContrato, idAdmin) VALUES (@fecha, @hora, @idContrato, @idAdmin);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA SERVICIO'

ELSE IF @accion = 'U' BEGIN

UPDATE servicio

SET fecha = @fecha,

hora = @hora,

idContrato = @idContrato, idAdmin = @idAdmin

WHERE idServicio = @idServicio; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idServicio) +' EN LA TABLA SERVICIO' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Solicitud

CREATE PROCEDURE paActualizaSolicitud

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idSolicitud SMALLINT = NULL,

@tipoServicio VARCHAR(20), @fecha DATE,

@hora TIME(0), @idpuntoPartida TINYINT, @idpuntoLlegada TINYINT,

@idCliente SMALLINT

AS

BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

INSERT INTO solicitud (tipoServicio, fecha, hora, idpuntoPartida, idpuntoLlegada, idCliente)

VALUES (@tipoServicio, @fecha, @hora, @idpuntoPartida, @idpuntoLlegada, @idCliente);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA SOLICITUD'

END

ELSE IF @accion = 'U' BEGIN

UPDATE solicitud

SET tipoServicio = @tipoServicio, fecha = @fecha,

hora = @hora,

idpuntoPartida = @idpuntoPartida, idpuntoLlegada = @idpuntoLlegada, idCliente = @idCliente

WHERE idSolicitud = @idSolicitud; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idSolicitud) +' EN LA TABLA SOLICITUD' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

* + 1. Tipo Comprobante

CREATE PROCEDURE paActualizaTipoComprobante

@accion CHAR(1), -- 'I' para insertar, 'U' para actualizar @idTipoComprobante CHAR(1),

@nombre VARCHAR(20)

AS BEGIN

SET NOCOUNT ON;

IF @accion = 'I' BEGIN

END

INSERT INTO tipoComprobante (idTipoComprobante, nombre) VALUES (@idTipoComprobante, @nombre);

PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA TIPO COMPROBANTE'

ELSE IF @accion = 'U' BEGIN

UPDATE tipoComprobante SET nombre = @nombre

WHERE idTipoComprobante = @idTipoComprobante; PRINT 'SE ACTUALIZÓ EL REGISTRO ' +

CONVERT(VARCHAR(10),@idTipoComprobante) +' EN LA TABLA TIPO COMPROBANTE' END

ELSE

PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) -->

ACTUALIZAR' END;

## Creación de Funciones Escalares

* + 1. Calcular el monto total de Ventas

Es una función que nos permite calcular el total de ventas por cada tipo de comprobante, según lo indiquemos con el id que referencia a un tipo de comprobante

CREATE FUNCTION dbo.fnc\_obtenerMontoTotalComprobantes

(

@idTipoComprobante varchar(10), --INDICAMOS EL ID DEL TIPO DE COMPROBANTE @fecha date --Y LA FECHA POR LA QUE QUEREMOS FILTRAR EL TOTAL DE VENTAS

)

RETURNS NUMERIC(10, 2) --SE RETORNA UN VALOR NUMÉRICO PORQUE CALCULAREMOS

--EL TOTAL DE VENTAS

AS BEGIN

DECLARE @montoTotal NUMERIC(10, 2); --DECLARAMOS UNA VARIABLE DONDE

--ALMACENAREMOS EL TOTAL

IF @idTipoComprobante = '1' --'1' ES EL ID DEL TIPO DE COMPROBANTE BOLETA BEGIN

SELECT @montoTotal = SUM(total) --SUMARÁ TODOS LOS VALOS DE

--LA COLUMNA TOTAL

FROM dbo.comprobante

--CUANDO LA COLUMNA IDTIPOCOMPROBANTE SEA IGUAL AL ID INGRESADO Y LA FECHA SEA IGUAL A LA FECHA INGRESADA

WHERE idTipoComprobante = @idTipoComprobante and fecha = @fecha;

END

ELSE IF @idTipoComprobante = '2' --'2' ES EL ID DEL TIPO DE COMPROBANTE FACTURA BEGIN

SELECT @montoTotal = SUM(total) --SUMARÁ TODOS LOS VALOS DE

--LA COLUMNA TOTAL

ELSE

END BEGIN END

FROM dbo.comprobante

--CUANDO LA COLUMNA IDTIPOCOMPROBANTE SEA IGUAL AL ID INGRESADO Y LA FECHA SEA IGUAL A LA FECHA INGRESADA

WHERE idTipoComprobante = @idTipoComprobante and fecha = @fecha;

SET @montoTotal = NULL; -- Tipo de comprobante no válido

RETURN @montoTotal; --RETORNAMOS EL TOTAL DE LAS VENTAS PARA ESE TIPO

--DE COMPROBANTE

END;

* + 1. Calcular Tarifa de transporte

Esta función calculará la distancia entre el punto de partida, en este caso siempre partirá desde Lima, hasta el punto de llegada, que puede ser otro departamento; para esto indicamos el id del departamento hasta el cual se hará la mudanza. Cabe resaltar que esta tarifa es únicamente por trasporte, no incluye el resto de tarifas.

CREATE FUNCTION dbo.fnc\_calcularTarifaPorRecorrido (

@iddestino tinyint --INDICAMOS EL ID DEL DEPARTAMENTO DESTINO HASTA DONDE SE REALIZARÁ LA MUDANZA

)

RETURNS DECIMAL(10,2) --SE RETORNA UN VALOR DECIMAL PORQUE CALCULAREMOS EL PRECIO TOTAL POR RECORRIDO

AS BEGIN

DECLARE @distanciaOrigen DECIMAL(8,2); --VARIABLE PARA GUARDAR LA DISTANCIA EN KM DE ORIGEN

DECLARE @distanciaDestino DECIMAL(8,2); --VARIABLE PARA GUARDAR LA DISTANCIA EN KM DE DESTINO

DECLARE @precioPorKm DECIMAL(5,2) = 2.5; --DEFINIMOS UNA TARIFA DE S/2.5 POR CADA KM RECORRIDO

SELECT @distanciaOrigen = [distancia(km)] --ASIGANOS LOS KM DE DISTANCIA A LA

--VARIABLE DE ORIGEN

FROM departamento

WHERE nombre = 'LIMA';--EN ESTE CASO LAS MUDANZAS SOLO SE REALIZAN DESDE

--LA CIUDAD DE LIMA HACIA OTRO DEPARTAMENTO, ES DECIR,

--EL ORIGEN ES SIEMPRE DESDE LIMA

SELECT @distanciaDestino = [distancia(km)]----ASIGANOS LOS KM DE DISTANCIA A LA VARIABLE DE DESTINO

FROM departamento

WHERE idDepartamento = @iddestino;--OBTENEMOS LOS KM DE DISTANCIA DESDE

--LA CIUDAD DE LIMA HASTA EL DEPARTAMENTO

--INDICADO EN EL PARÁMETRO

DECLARE @distanciaRecorrida DECIMAL(8,2);--DECLARAMOS UNA VARIABLE PARA GUARDAR LA DISTANCIA QUE SE RECORRERÁ

--ENTRE EL

DEPARTAMENTO DE ORIGEN Y DEPARTAMENTO DE DESTINO

SET @distanciaRecorrida = ABS(@distanciaDestino - @distanciaOrigen);

--LE ASIGNAMOS EL VALOR A LA VARIABLE

--RESTANTO LAS DISTANCIAS Y USAMOS LA

--FUNCION ABS PARA EVITAR VALORES

--NEGATIVOS AL EFECTUAR LA RESTA

DECLARE @monto DECIMAL(10,2);--DECLARAMOS UNA VARIABLE PARA GUARDAR EL MONTO TOTAL

SET @monto = @distanciaRecorrida \* @precioPorKm; --LE ASIGNAMOS EL VALOR A LA VARIABLE MULTIPLICANDO LOS KM QUE SE RECORRERÁN POR LA TARIFA DE PRECIO POR KM

RETURN @monto; --RETORNAMOS EL TOTAL EN SOLES DE LA TARIFA QUE SE COBRARÁ POR EL SERVICIO DE TRANSPORTE

END;

## Creación de Vistas

* + 1. Vista 1

CREATE VIEW vServicios AS

-- Este script crea una vista llamada vServicios que proporciona información detallada sobre los servicios y sus detalles relacionados.

-- Creación de la vista vServicios. SELECT

-- Selecciona el ID del servicio. s.idServicio IDServicio,

-- Selecciona la fecha del servicio. s.fecha Fecha,

-- Selecciona la hora del servicio. s.hora Hora,

-- Concatena y muestra el nombre completo del administrador. pr.apPat + ' ' + pr.apMat + ', ' + pr.nombre Administrador,

-- Cuenta el número de estibadores asociados al servicio. COUNT(es.idEstServ) Estibador,

-- Selecciona el ID del cliente. c.idCliente Cliente,

-- Selecciona el nombre del punto de partida. dp.nombre Partida,

-- Selecciona el nombre del destino. dl.nombre Destino

FROM servicio s

-- Realiza una unión interna entre servicio y administrador. INNER JOIN administrador a ON s.idAdmin = a.idAdmin

-- Realiza una unión interna entre administrador y persona. INNER JOIN persona pr ON a.idPersona = pr.idPersona

-- Realiza una unión interna entre servicio y estibador\_Servicio. INNER JOIN estibador\_Servicio es ON s.idServicio = es.idServicio

-- Realiza una unión interna entre servicio y contrato. INNER JOIN contrato ct ON s.idContrato = ct.idContrato

-- Realiza una unión interna entre contrato y propuesta. INNER JOIN propuesta pt ON ct.idPropuesta = pt.idPropuesta

-- Realiza una unión interna entre propuesta y solicitud. INNER JOIN solicitud st ON pt.idSolicitud = st.idSolicitud

-- Realiza una unión interna entre solicitud y cliente. INNER JOIN cliente c ON c.idCliente = st.idCliente

-- Realiza una unión interna entre solicitud y departamento (punto de partida). INNER JOIN departamento dp ON st.idpuntoPartida = dp.idDepartamento

-- Realiza una unión interna entre solicitud y departamento (punto de llegada). INNER JOIN departamento dl ON st.idpuntoLlegada = dl.idDepartamento

-- Agrupa los resultados por ID del servicio, hora, fecha, nombre completo del administrador, ID del cliente, nombre del punto de partida y nombre del destino.

GROUP BY s.idServicio, s.hora, s.fecha, pr.apPat + ' ' + pr.apMat + ', ' + pr.nombre, c.idCliente, dp.nombre, dl.nombre;

* + 1. Vista 2

CREATE VIEW vClientes AS

--Este script realiza una consulta que muestras todos los datos de un clientes y decide si mostrar el nombre completo o la razón social en función de si el cliente es una persona natural o jurídica.

--Utiliza uniones izquierdas para combinar la información de las tablas relacionadas.

-- Selecciona el ID de cliente y utiliza una expresión CASE para decidir qué información mostrar.

SELECT

c.idCliente AS ID, CASE

-- Si el campo idPersNatural en la tabla personaNatural no es nulo, concatena los campos apPat, apMat (si no es nulo) y nombre.

-- Esto forma el nombre completo de la persona natural.

WHEN pn.idPersNatural IS NOT NULL THEN pn.apPat + ' ' + ISNULL(pn.apMat, '') + ', ' + pn.nombre

-- Si el campo idPersJuridica en la tabla personaJuridica no es nulo, selecciona la razonSocial.

-- Esto muestra la razón social de la persona jurídica. WHEN pj.idPersJuridica IS NOT NULL THEN pj.razonSocial

-- Si ninguna de las condiciones anteriores es verdadera, muestra una cadena vacía.

ELSE ''

END AS Cliente, CASE

-- Si el campo idPersNatural en la tabla personaNatural no es nulo, muestra que

--el tipo de documento es DNI

WHEN pn.idPersNatural IS NOT NULL THEN 'DNI'

-- Si el campo idPersJuridica en la tabla personaNatural no es nulo, muestra que

--el tipo de documento es RUC

WHEN pj.idPersJuridica IS NOT NULL THEN 'RUC'

-- Si ninguna de las condiciones anteriores es verdadera, muestra una cadena vacía.

ELSE ''

END AS TipoDoc, CASE

muestra

muestra

-- Si el campo idPersNatural en la tabla personaNatural no es nulo,

--el número de documento (DNI)

WHEN pn.idPersNatural IS NOT NULL THEN pn.dni

-- Si el campo idPersJuridica en la tabla personaNatural no es nulo,

--el número de documento (RUC)

WHEN pj.idPersJuridica IS NOT NULL THEN pj.ruc

-- Si ninguna de las condiciones anteriores es verdadera, muestra una

cadena vacía.

ELSE ''

END AS NumDoc,

--tambien muestra el celular, dirección y correo del cliente c.celular AS Celular,

c.direccion AS Dirección, c.correo AS Correo

FROM cliente c

-- Realiza una unión izquierda (LEFT JOIN) entre la tabla cliente y la tabla personaNatural

-- para combinar la información de ambos tipos de clientes. LEFT JOIN personaNatural pn ON c.idCliente = pn.idCliente

-- Realiza una unión izquierda (LEFT JOIN) entre la tabla cliente y la tabla personaJuridica

-- para combinar la información de ambos tipos de clientes. LEFT JOIN personaJuridica pj ON c.idCliente = pj.idCliente;

## Creación de Auditorías y Triggers

* + 1. Auditoría para la tabla Propuesta

Esta tabla de auditoría la hemos considerado por si llega a haber algún cambio en la propuesta que se le hizo a un cliente, ya sea que hubo un error al momento de generar la propuesta o si el cliente decide hacer algún cambio en el servicio; por lo que almacenaremos los cambios que haya en el costo de dicha propuesta.

-- Crear tabla de auditoría para guardar los cambios en propuesta CREATE TABLE auditoriapropuesta(

-- Columna para identificación única de auditoría idAuditoria INT IDENTITY(1, 1) PRIMARY KEY,

-- ID del cliente que se modificó idCliente SMALLINT,

-- Columna que se modificó (direccion, celular, correo) columnaModificada VARCHAR(50),

-- Valor antiguo de la columna modificada valorAntiguo VARCHAR(150),

-- Valor nuevo de la columna modificada valorNuevo VARCHAR(150),

-- Fecha de modificación fechaModificacion date NOT NULL,

-- Hora de modificación horaModificacion time(0) NOT NULL,

-- Usuario que realizó la modificación usuarioModificacion varchar(50) NOT NULL

);

-- Crear trigger para auditar cambios en la tabla propuesta CREATE TRIGGER tr\_AuditarCambiosPropuesta

ON [dbo].[propuesta] AFTER UPDATE

AS BEGIN

-- Variables para almacenar datos DECLARE @idPropuesta smallint;

DECLARE @oldCosto decimal(10, 2), @newCosto decimal(10, 2); DECLARE @usuario varchar(50);

-- Obtener valores de la tabla deleted (antes de la actualización) SELECT @idPropuesta = [idPropuesta], @oldCosto = [costo], @usuario =

SYSTEM\_USER

FROM deleted;

-- Obtener valores de la tabla inserted (después de la actualización) SELECT @newCosto = [costo] FROM inserted;

-- Verificar si el valor de "costo" cambió IF @oldCosto <> @newCosto

BEGIN

-- Insertar en la tabla de auditoría

INSERT INTO auditoriapropuesta ([idPropuesta], [columnaModificada], [valorAntiguo], [valorNuevo], [fechaModificacion], [horaModificacion], [usuarioModificacion])

VALUES (@idPropuesta, 'costo', @oldCosto, @newCosto, CONVERT(DATE, GETDATE()), CONVERT(TIME(0), GETDATE()), @usuario);

END

END;

* + 1. Auditoría para la tabla Cliente

Esta tabla de auditoría la hemos considerado por si llega a haber algún cambio datos de un cliente, ya sea que el cliente cambio de dirección, celular o correo, almacenaremos estos cambios para tener un registro o historial de los datos de los clientes registrados.

-- Crear tabla de auditoría para guardar los cambios en la tabla cliente CREATE TABLE auditoriaCliente (

-- Columna para identificación única de auditoría idAuditoria INT IDENTITY(1, 1) PRIMARY KEY,

-- ID del cliente que se modificó idCliente SMALLINT,

-- Columna que se modificó (direccion, celular, correo) columnaModificada VARCHAR(50),

-- Valor antiguo de la columna modificada valorAntiguo VARCHAR(150),

-- Valor nuevo de la columna modificada valorNuevo VARCHAR(150),

-- Fecha de modificación fechaModificacion date NOT NULL,

-- Hora de modificación horaModificacion time(0) NOT NULL,

-- Usuario que realizó la modificación usuarioModificacion varchar(50) NOT NULL

);

-- Crear trigger para auditar cambios en la tabla cliente CREATE TRIGGER tr\_AuditarCambiosCliente

ON cliente AFTER UPDATE AS

BEGIN

-- Verificar si hay cambios en las columnas relevantes

IF NOT UPDATE(direccion) AND NOT UPDATE(celular) AND NOT UPDATE(correo) RETURN; -- No hay cambios en las columnas relevantes

-- Variables para almacenar datos DECLARE @idCliente smallint;

DECLARE @columnaModificada VARCHAR(50); DECLARE @valorAntiguo VARCHAR(150); DECLARE @valorNuevo VARCHAR(150);

DECLARE @usuario varchar(50);

-- Obtener el usuario que realizó la modificación SELECT @usuario = SYSTEM\_USER;

-- Obtener datos de las tablas inserted y deleted (antes y después de la actualización)

SELECT @idCliente = i.idCliente,

@columnaModificada = CASE

WHEN i.direccion <> d.direccion THEN 'direccion' WHEN i.celular <> d.celular THEN 'celular'

WHEN i.correo <> d.correo THEN 'correo'

END,

@valorAntiguo = CASE

WHEN @columnaModificada = 'direccion' THEN d.direccion

WHEN @columnaModificada = 'celular' THEN d.celular WHEN @columnaModificada = 'correo' THEN d.correo

END,

@valorNuevo = CASE

WHEN @columnaModificada = 'direccion' THEN i.direccion

WHEN @columnaModificada = 'celular' THEN i.celular WHEN @columnaModificada = 'correo' THEN i.correo

END

FROM deleted d

INNER JOIN inserted i ON d.idCliente = i.idCliente;

-- Insertar en la tabla de auditoría si hubo una modificación relevante IF @columnaModificada IS NOT NULL

BEGIN

INSERT INTO auditoriaCliente (idCliente, columnaModificada, valorAntiguo, valorNuevo, fechaModificacion, horaModificacion, usuarioModificacion)

VALUES (@idCliente, @columnaModificada, @valorAntiguo, @valorNuevo, CONVERT(DATE, GETDATE()), CONVERT(TIME(0), GETDATE()), @usuario);

END

END;

## Creación de Transacción

En este ejemplo, el procedimiento ***<<paRegistrarCliente>>*** recibe los datos del cliente, incluido el tipo de persona ("N" para persona natural, "J" para persona jurídica) y los datos específicos según el tipo de persona. Luego, se inserta en la tabla "cliente", se obtiene el ID del cliente recién insertado y se intenta insertar en la tabla correspondiente según el tipo de persona.

Si ocurre algún error durante el proceso, se realizará un **rollback** para asegurar que no haya datos inconsistentes. Si el proceso es exitoso, se confirmará la transacción. Es importante proporcionar los valores adecuados para los parámetros de acuerdo con el tipo de persona que estás registrando.

CREATE PROCEDURE paRegistrarCliente @direccion varchar(150), @celular char(9),

@correo varchar(50), @idCategoria tinyint,

@tipoPersona char(1), -- 'N' para persona natural, 'J' para persona jurídica

@apPat varchar(50) = NULL, @apMat varchar(50) = NULL, @nombre varchar(50) = NULL, @dni varchar(8) = NULL,

@razonSocial varchar(100) = NULL, @ruc char(11) = NULL

AS BEGIN

-- Iniciar la transacción BEGIN TRANSACTION;

DECLARE @idCliente smallint;

-- Insertar en la tabla cliente

INSERT INTO [dbo].[cliente] ([direccion], [celular], [correo], [idCategoria])

VALUES (@direccion, @celular, @correo, @idCategoria);

-- Obtener el ID del cliente recién insertado SET @idCliente = SCOPE\_IDENTITY();

-- Intentar insertar en la tabla correspondiente BEGIN TRY

IF @tipoPersona = 'N'

BEGIN

INSERT INTO [dbo].[personaNatural] ([apPat], [apMat], [nombre], [dni], [idCliente])

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

END

ELSE IF @tipoPersona = 'J'

BEGIN

[ruc], [idCliente])

END

INSERT INTO [dbo].[personaJuridica] ([razonSocial], VALUES (@razonSocial, @ruc, @idCliente);

-- Si todo ha sido exitoso, confirmar la transacción COMMIT TRANSACTION;

END TRY BEGIN CATCH

-- Si ocurre un error, revertir la transacción ROLLBACK TRANSACTION;

-- Manejar el error aquí (puedes imprimir un mensaje de error o realizar otras acciones)

PRINT 'Error durante el registro: ' + ERROR\_MESSAGE(); END CATCH;

END;

## Creación de Cursor

Justificación para usar un cursor:

El uso de un cursor es apropiado en situaciones en las que necesitas procesar fila por fila y realizar operaciones específicas en cada una de ellas. En este caso, el cursor se utiliza para recorrer la tabla "solicitud" fila por fila y realizar operaciones con los valores de cada fila.

Razones para usar un cursor en lugar de un único comando Transact-SQL:

* ***Procesamiento por fila:*** Cuando necesitas aplicar lógica o cálculos específicos en cada fila de la tabla, un cursor te permite acceder a los valores individualmente y realizar las operaciones necesarias en cada paso del bucle.
* ***Lógica condicional compleja:*** Si la lógica que deseas aplicar en cada fila es compleja y varía según ciertas condiciones, un cursor te permite manejar estas condiciones de manera más granular y flexible.
* ***Interacción con otras tablas o procedimientos:*** Si necesitas interactuar con otras tablas o procedimientos almacenados en función de los valores de cada fila, un cursor te brinda el control necesario para coordinar estas interacciones.
* ***Visualización de procesamiento:*** Usar un cursor te permite imprimir o registrar información detallada sobre el procesamiento de cada fila, lo que puede ser útil para fines de seguimiento y depuración.

Sin embargo, es importante tener en cuenta que el uso de cursores puede tener un impacto en el rendimiento, especialmente en tablas grandes. Siempre es recomendable evaluar alternativas, como consultas en conjunto, que son más eficientes en términos de rendimiento cuando sea posible.

-- Crear el procedimiento almacenado CREATE PROCEDURE paRecorrerSolicitud AS

BEGIN

-- Declarar variables para almacenar los valores de las filas DECLARE @idSolicitud smallint,

@tipoServicio varchar(20), @fecha date,

@hora time(0), @idpuntoPartida tinyint, @idpuntoLlegada tinyint, @idCliente smallint;

-- Declarar el cursor

DECLARE cursorSolicitud CURSOR FOR

SELECT [idSolicitud], [tipoServicio], [fecha], [hora], [idpuntoPartida], [idpuntoLlegada], [idCliente]

FROM [dbo].[solicitud];

-- Abrir el cursor OPEN cursorSolicitud;

-- Inicializar variables

FETCH NEXT FROM cursorSolicitud INTO @idSolicitud, @tipoServicio, @fecha, @hora, @idpuntoPartida, @idpuntoLlegada, @idCliente;

-- Recorrer la tabla usando el cursor WHILE @@FETCH\_STATUS = 0

BEGIN

-- Realizar operaciones con los valores de la fila actual

-- Aquí puedes realizar las operaciones que necesites con los valores de la fila actual

-- Por ejemplo, podrías imprimir los valores o realizar cálculos

-- Imprimir los valores de la fila actual

PRINT 'idSolicitud: ' + CAST(@idSolicitud AS varchar(10)) + ', tipoServicio: ' + @tipoServicio + ', idCliente: ' + CAST(@idCliente as varchar(10));

-- Obtener la siguiente fila

FETCH NEXT FROM cursorSolicitud INTO @idSolicitud, @tipoServicio, @fecha, @hora, @idpuntoPartida, @idpuntoLlegada, @idCliente;

END;

END;

-- Cerrar y desalojar el cursor CLOSE cursorSolicitud; DEALLOCATE cursorSolicitud;

exec paRecorrerSolicitud