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

UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

PROCESO:

SISTEMA DE MUDANZAS DE LA EMPRESA “TRANSMARC”.

CURSO:

BASE DE DATOS.

DOCENTE:

JORGE ALVARADO TABACCHI.

INTEGRANTES:

- GODOS VIERA, ANTHONY GERARDO (100%)
- PARIAHUACHE PEÑA, JUANA BISNEY (100%)
- HUANCA FLORES, SEGUNDO ELVIS (100%)

PIURA – PERÚ

2023

Contenido

I. MARCO REFERENCIAL SOBRE LA EMPRESA

TRANSMARC es una pequeña empresa de la ciudad de Lima y a nivel nacional que se dedica al servicio de mudanza, tanto a clientes comunes como a empresas y pymes, y cuenta con una flota de 4 camiones. El servicio de mudanza consta de transporte y embalaje.

Esta empresa cuenta con una cartera de clientes (los cuales son almacenados en una agenda) conformada por personas comunes y también empresas.

Esta empresa cuenta con una página de Facebook, donde los clientes pueden acceder a la información necesaria y solicita el pedido de servicio.

Por cada servicio, la empresa tiene que pasar una aduana o por control cuando los viajes son largos.

Cuando el cliente desea contratar el servicio se le pregunta qué tipo de servicio desea (transporte, embalaje y mudanza), así mismo se le pregunta el punto de partida y punto de llegada.

El cliente efectúa el pago una vez culminado el servicio, así mismo la empresa emite comprobantes y el trato se cierra mediante un contrato con el fin de evitar inconvenientes con los productos.

Cuando se cierra un contrato con el cliente, a este se le llama antes para hacerle recordar del servicio pendiente que tiene y pronto se le prestara, para ello se le pide datos personales como dirección, número de teléfono, correo electrónico (esto a elección) y datos adicionales para el registro del servicio.

Los pagos se realizan de manera electrónica por billetera digital, transferencias bancarias o en efectivo, si es un cliente recurrente tiene acceso a ciertos beneficios como descuentos, etc.

Transmarc tiene a su disposición empleados que realizan la función de cargadores para movilizar los productos al camión y también para descargarlos, también cuenta con conductores encargados de movilizar los camiones hasta el punto de destino.

Con cada servicio de transporte se necesita de una hoja guía donde se especifica todo lo que se está trasladando, datos del conductor, lugar de destino, y motivo de transporte, esto con el fin de que al momento del traslado y pasen por una revisión (controles de carretera) o si se desea entrar a otra empresa, se tenga un registro de lo que se está transportando.

Así mismo si hay un gasto adicional, en los peajes la empresa se encargara de aquello, pero este gasto va incluido al momento de hacer la cotización del servicio al cliente.

- ✓ ***¿Qué tipos de pago se aceptan?***

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

- ✓ ***¿Se entregan facturas?***

  Emite facturas cuando es necesario, es decir si es un servicio local no se emiten factura; por ejemplo, si quieres mover una cama, ropero, etc. pero en el mismo distrito y si es un cliente común.

- ✓ ***¿Qué datos necesitas para realizar la mudanza?***

  Todos los datos del cliente, guía de remisión, punto de destino, punto de partida, fecha estimada para el servicio.

  Anota los servicios en agendas, el cliente firma un contrato de por medio; para clientes en general se pide la ubicación y distancia para la realización de la propuesta, y se requiere de una hoja guía que consiste en los datos a donde va las cosas y un permiso por parte del cliente para movilizarlas para evitar cualquier inconveniente al momento del traslado.

- ✓ ***¿Se firma un contrato?***

  Sí, para la confirmación del servicio.

- ✓ ***¿Qué pasa si el cliente cancela la mudanza habiendo pagado una parte del servicio o el servicio completo?***

  Se le reembolsa el dinero, pero aplicando un descuento como sanción según las políticas de la empresa.

- ✓ ***¿El material de embalaje, peajes, combustible, etc. supone un gasto adicional para el cliente?***

  No, estos costos adicionales ya están incluido al momento de hacer la cotización y especificados en el contrato.

II. REQUERIMIENTOS

2.1. 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.

Nº 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 mudanza a través de la página de Facebook de la empresa o por teléfono, requieren detalles como tipo de servicio (local o nacional), lugar de partida, lugar de destino, fecha y hora deseada.
RF3	Programación y viabilidad	El sistema debe permitir a los administradores programar los servicios de mudanza considerando la viabilidad, como la zona de destino, facilidad de llegada y capacidad de los recursos.
RF4	Programación y asignación	El sistema debe permitir a los administradores programar y asignar los servicios de mudanza a los equipos correspondientes, considerando la disponibilidad de los camiones y empleados.
RF5	Seguimiento del estado de la mudanza	Los clientes deben poder rastrear el estado de su mudanza a través del sistema, desde la confirmación de la reserva hasta la finalización del servicio.
RF6	Generación de comprobantes	El sistema debe generar comprobantes (boletas o facturas) para los servicios de mudanza cuando sea necesario.
RF6	Gestión de inventario	El sistema debe permitir a la empresa gestionar el inventario de los artículos a ser trasladados, incluyendo una lista detallada de los elementos, su estado y cualquier instrucción especial.
RF7	Reembolso de pagos	El sistema debe ser capaz de procesar reembolsos en caso de que un cliente cancele la mudanza después de haber realizado un pago parcial, deduciendo los gastos incurridos por la empresa.

2.2. 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.

Nº REQUERIMIENTO	**NOMBRE**	**DESCRIPCIÓN**
RNF1	Usabilidad	El sistema debe ser fácil de usar tanto para los clientes como para los administradores de la empresa de mudanzas.
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 mudanza cuando sea necesario.
RNF6	Integración con servicios de terceros	El sistema debe ser capaz de integrarse con servicios externos, como sistemas de control de aduanas o herramientas de seguimiento de paquetes, para facilitar los procesos de mudanza a nivel nacional.

III. DISEÑO DEL SISTEMA

3.1. 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)

3.2. Descripción de Entidades

1. PersonaNatural:
    a. *Descripción:* Persona natural que requiere un servicio
    b. Atributos:
        i. idPersNatural (PK): Identificador unívoco de la persona Natural
        ii. nombre: Nombre de la persona Natural
        iii. apPat: Apellido materno de la persona Natural
        iv. apMat: Apellido paterno de la persona Natural (opcional)
        v. dni (U): Documento Nacional de Identidad
        vi. idCliente (FK): Identificador del Cliente asociada la Persona Natural
    c. *Propósito:* Almacena información personal de una persona natural que requiera solicitar un servicio.
2. PersonaJuridica:
    a. *Descripción:* Persona jurídica que requiere un servicio
    b. Atributos:
        i. idPersJuridica (PK): Identificador unívoco de persona Jurídica
        ii. razonSocial: Razón social de la persona Jurídica
        iii. ruc (U): Número de Registro Único de Contribuyente
        iv. idCliente (FK): Identificador del Cliente asociada la Persona Jurídica
    c. *Propósito:* Almacena información personal de una persona jurídica que requiera solicitar un servicio.
3. Cliente:
    a. *Descripción:* Representa a un cliente que utiliza los servicios de la empresa de mudanzas.
    b. Atributos:
        i. idCliente (PK): Identificador unívoco del cliente
        ii. idCategoria (FK): Identificador de la Categoría asociada al cliente
        iii. dirección: Dirección del cliente
        iv. celular: Número de celular del cliente

v. correo: Correo electrónico del cliente (opcional)

c. *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:
    a. *Descripción:* Representa una solicitud para el pedido de servicio por un cliente
    b. Atributos:
        i. idSolicitud (PK): Identificador unívoco de la solicitud
        ii. tipoServicio: Tipo de servicio que desea el cliente (transporte y/o embalaje)
        iii. fecha: fecha de la solicitud
        iv. hora: hora de la solicitud
        v. idpuntoPartida (FK): Identificador del departamento asociado al punto de partida de donde se hará el servicio.
        vi. idpuntoLlegada (FK): Identificador del departamento asociado al punto de llegada hasta donde se trasladará el servicio.
        vii. idCliente (FK): Identificador del cliente que ha realizado la solicitud
    c. *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:
    a. *Descripción:* Representa la categoría del cliente (cliente recurrente o cliente nuevo)
    b. Atributos:
        i. idCategoria (PK): Identificador unívoco de la categoría
        ii. nombre: Nombre de la categoría
        iii. descripción: Describe el tipo de beneficios que tiene cada categoría.
    c. *Propósito:* Almacena información relacionada de la categoría que se encuentra el cliente.
6. Propuesta:
    a. *Descripción:* Representa una propuesta de la empresa al cliente.
    b. Atributos:
        i. idPropuesta (PK): Identificador unívoco de la propuesta
        ii. idSolicitud (FK): Identificador de la solicitud asociada a la propuesta

iii. fecha: Fecha de la propuesta
iv. hora: Hora de la propuesta
v. costo: Costo del servicio

c. *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:
   a. *Descripción:* Representa un contrato entre la empresa de mudanzas y un cliente.
   b. Atributos:
      i. idContrato (PK): Identificador unívoco del contrato.
      ii. fecha: Fecha del contrato.
      iii. hora: Hora del contrato.
      iv. IdPropuesta (FK): Identificador de la propuesta asociada al contrato.
   c. *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:
   a. *Descripción:* Representa un comprobante emitido para un contrato.
   b. Atributos:
      i. IdComprobante (PK): Identificador unívoco del Comprobante
      ii. serieComprobante (U): Serie del comprobante.
      iii. numComprobante (U): Número del comprobante.
      iv. fecha: Fecha del comprobante.
      v. hora: Hora del comprobante.
      vi. IGV: Impuesto General a las Ventas (18%) aplicado al total.
      vii. subtotal: Subtotal del comprobante.
      viii. total: Total del comprobante.
      ix. idTipoComprobante (FK): Identificador de tipo de comprobante asociado a un comprobante.
      x. idContrato (FK): Identificador del contrato asociado al comprobante.
   c. *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:
    a. *Descripción:* Representa los tipos de comprobante disponibles.
    b. Atributos:
        i. idTipoComprobante (PK): Identificador unívoco del tipo de comprobante.
        ii. nombre: Nombre del tipo de comprobante.
    c. *Propósito:* Almacena los diferentes tipos de comprobante utilizados en la emisión de los comprobantes para los contratos.
10. Servicio:
    a. *Descripción:* Representa un servicio de mudanza asociado a un contrato.
    b. Atributos:
        i. idServicio (PK): Identificador unívoco del servicio.
        ii. fecha: Fecha del servicio.
        iii. hora: Hora del servicio.
        iv. idContrato (FK): Identificador del contrato asociado al servicio.
        v. idAdministrador (FK): Identificador del administrador responsable del servicio.
    c. *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:
    a. 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.
    b. Atributos:
        i. idDetalleServicio (PK): Identificador unívoco de cada línea del detalle del servicio
        ii. descripción: Registra los detalles y características de cada producto
        iii. cantidad: Indica la cantidad de productos que se están trasladando
        iv. tipo: Indica el tipo de producto que se traslada
        v. estado: Descripción del estado en el cual se traslada el artículo.

vi. idServicio (FK): Identificador del servicio asociado al detalle del servicio
vii. idGuia (FK): Identificador de la guía asociada al detalle del servicio

c. Propósito: Registrar el detalle de cada producto que se está trasladando en un servicio.

12. Anulación:
    a. *Descripción:* Representa una anulación o cancelación de un contrato.
    b. Atributos:
        i. idAnulación (PK): Identificador unívoco de la anulación.
        ii. montoDescontado: Monto descontado por la anulación.
        iii. idContrato (FK): Identificador del contrato anulado.
        iv. idAdministrador (FK): Identificador del administrador responsable de la anulación.
        v. Fecha: Fecha en la que se realiza la anulación de un servicio.
        vi. Hora: Hora en la que se realiza la anulación de un servicio.
    c. *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
    a. *Descripción:* Representa a los administradores encargados en la empresa.
    b. Atributos:
        i. idAministrador (PK): Identificador unívoco del administrador
        ii. idPersona (FK): Identificador de los datos de la persona asociada a administrador
    c. *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
    a. *Descripción:* Representa a las guías de remisión generadas por cada servicio de mudanza que se realice.
    b. Atributos:
        i. IdGuia (PK): Identificador unívoco de la guía de remisión
        ii. serieGuia (U): Representa la serie de la guía de remisión
        iii. numGuia (U): Representa el número de la guía de remisión

iv. fechaTraslado: Indica la fecha en la que se realizará el servicio
v. motivoTraslado: Representa el motivo o razón del traslado de los bienes
vi. direccionPartida: Indica la dirección desde donde partirá el servicio
vii. direccionLLegada: Indica la dirección hasta donde tendrá que llegar el servicio.

c. *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
    a. *Descripción:* Representa a los camiones con los que cuenta la empresa
    b. Atributos:
        i. idCamion (PK): Identificador unívoco de cada camión
        ii. descripción: Detalla las características del camión (opcional)
        iii. Modelo: Representa el modelo del camión
        iv. pesoNeto_KG: Representa el peso del contenedor del camión
        v. pesoBruto_KG: Representa el peso total del camión
        vi. capacidad_TN: Indica cual es la carga máxima que puede transportar el camión
        vii. numPlaca (U): Identifica el número de placa único de cada camión
    c. *Propósito:* Registrar todos los camiones con todos sus datos, numero de placa de cada uno, sus características y capacidad de cada camión.
16. EstibadorServicio
    a. Descripción: Representa a un estibador que participa en cada servicio
    b. Atributos:
        i. idEstServ (PK): Identificador unívoco de cada estibador correspondiente a un servicio
        ii. idEstibador (FK): Identificador del estibador que participará en dicho servicio
        iii. idServicio (FK): Identificador del servicio en el cual participó dicho estibador
    c. *Propósito:* Registrar que estibadores participan en los diferentes servicios que se realizan
17. Estibador

a. *Descripción:* Representa a todos los estibadores que trabajan en la empresa
b. *Atributos:*
    i. idEstibador (PK): Identificador unívoco de un estibador
    ii. idPersona (FK): Identificador de los datos de la persona asociada a estibador
c. *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
    a. *Descripción:* Representa a todos los conductores que trabajan en la empresa
    b. *Atributos:*
        i. idConductor (PK): Identificador unívoco de un conductor
        ii. idPersona (FK): Identificador de los datos de la persona asociada a conductor
        iii. numLicencia (U): Representa el número de licencia único de cada conductor
    c. *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
    a. Descripción: Representa todos los datos de una persona
    b. Atributos:
        i. idPersona (PK): Identificador unívoco de una persona
        ii. nombre: Nombre de la persona
        iii. apPat: Apellido paterno de la persona
        iv. apMat: Apellido materno de la persona (opcional)
        v. dni (U): Número de Documento Nacional de Identidad
        vi. celular: Número de celular de la persona
        vii. correo: Correo electrónico de la persona (opcional)
        viii. dirección: Dirección de una persona
    c. Propósito: Registra todos los datos de las personas que pertenecen a la empresa.
20. CamiónServicio:
    a. *Descripción:* Representa a un camión que participa en cada servicio

b. *Atributos:*
    i. idCamServ (PK): Identificador unívoco de cada camión correspondiente a un servicio
    ii. idCamión (FK): Identificador del camión que participará en dicho servicio
    iii. idServicio (FK): Identificador del servicio en el cual participó dicho estibador
    iv. idConductor (FK): Identificador del conductor encargado de trasladar el camión correspondiente a un servicio.
c. *Propósito:* Registrar que camiones participan en los diferentes servicios que se realizan y los conductores que trasladarán dicho camión.

21. Departamento
    a. *Descripción:* Representa los departamentos del Perú en los cuales pueden realizar el servicio de mudanza.
    b. *Atributos:*
        i. idDepartamento (PK): Identificador unívoco de cada departamento.
        ii. Nombre: representa el nombre del departamento.
        iii. Distancia(km): representa la distancia en kilómetros desde Lima hasta otro departamento.
    c. *Propósito:* Registrar todos los departamentos a los cuales la empresa puede acceder para realizar el servicio de mudanza.

22. Provincia
    a. *Descripción:* Representa las provincias correspondientes a cada departamento
    b. *Atributos:*
        i. idProvincia (PK): Identificador unívoco de cada provincia
        ii. Nombre: representa el nombre de la provincia
        iii. idDepartamento (FK): Identificador del departamento asociado a una provincia
    c. *Propósito:* Registrar todos los departamentos a los cuales la empresa puede acceder para realizar el servicio de mudanza.

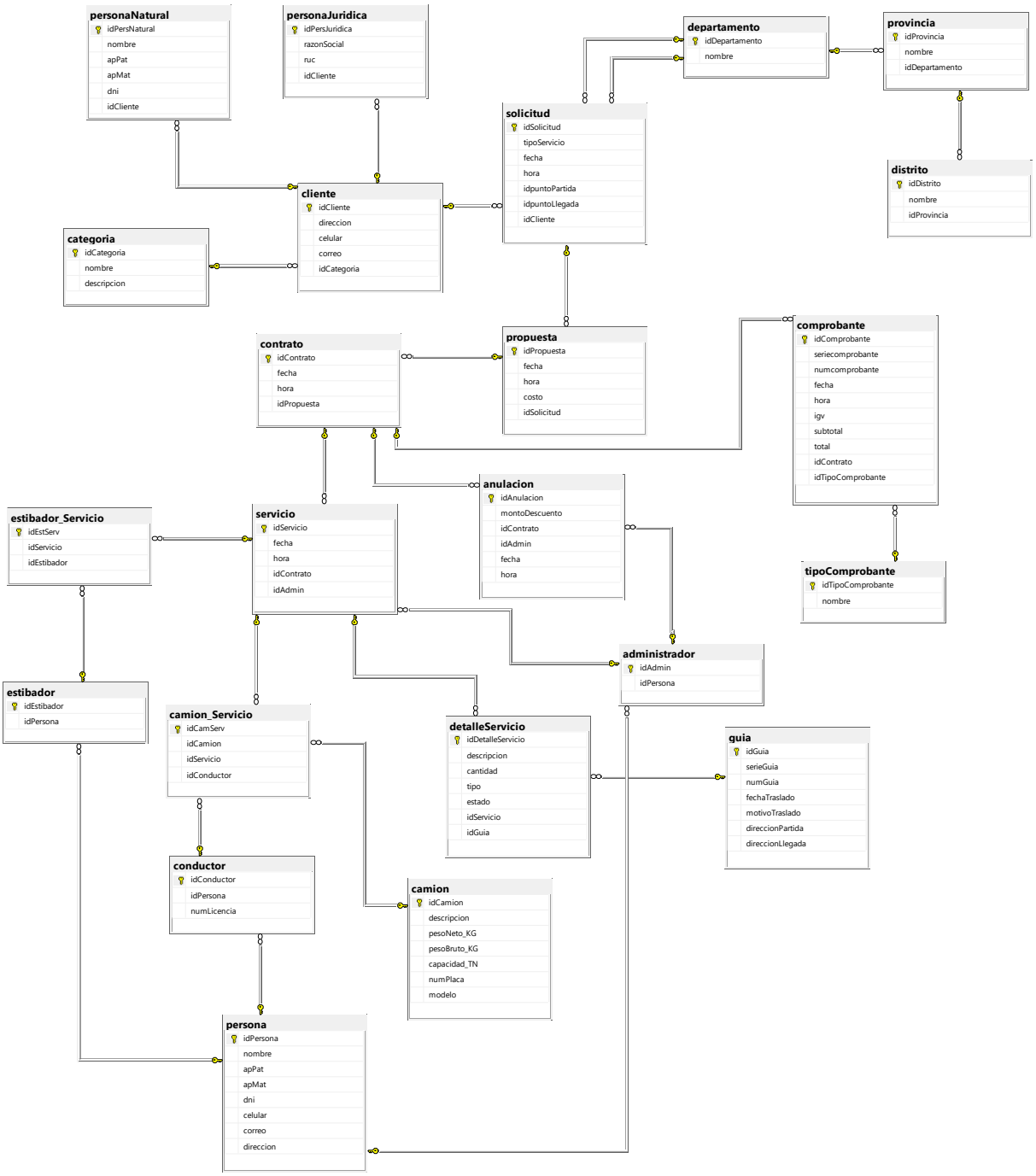
23. Distrito
    a. *Descripción:* Representa los distritos correspondientes a cada provincia
    b. *Atributos:*
        i. idDistrito (PK): Identificador unívoco de cada departamento

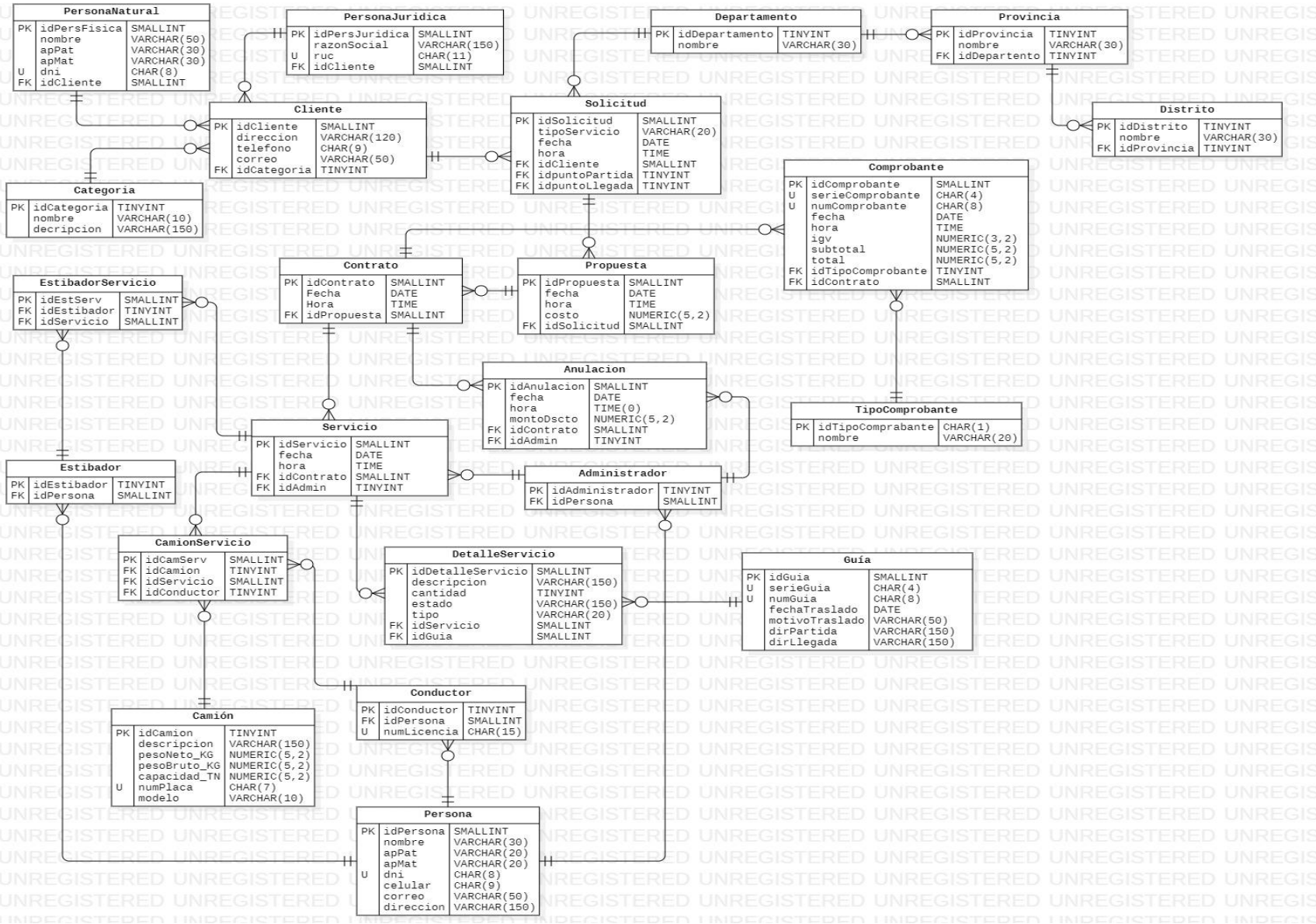
ii. Nombre: representa el nombre del departamento:

iii. idProvincia (FK): Identificador de la provincia asociada a un distrito.

c. *Propósito:* Registrar todos los departamentos a los cuales la empresa puede acceder para realizar el servicio de mudanza.

3.3. Modelo Relacional




PersonaNatural
PK idPersFisica SMALLINT
nombre VARCHAR(50)
apPat VARCHAR(30)
apMat VARCHAR(30)
U dni CHAR(8)
FK idCliente SMALLINT
PersonaJuridica
PK idPersJuridica SMALLINT
razonSocial VARCHAR(150)
U ruc CHAR(11)
FK idCliente SMALLINT
Departamento
PK idDepartamento TINYINT
nombre VARCHAR(30)
Provincia
PK idProvincia TINYINT
nombre VARCHAR(30)
FK idDepartento TINYINT
Distrito
PK idDistrito TINYINT
nombre VARCHAR(30)
FK idProvincia TINYINT
Cliente
PK idCliente SMALLINT
direccion VARCHAR(120)
telefono CHAR(9)
correo VARCHAR(50)
FK idCategoria TINYINT
Solicitud
PK idSolicitud SMALLINT
tipoServicio VARCHAR(20)
fecha DATE
hora TIME
FK idCliente SMALLINT
FK idpuntoPartida TINYINT
FK idpuntoLlegada TINYINT
Categoria
PK idCategoria TINYINT
nombre VARCHAR(10)
decripcion VARCHAR(150)
Comprobante
PK idComprobante SMALLINT
U serieComprobante CHAR(4)
U numComprobante CHAR(8)
fecha DATE
hora TIME
igv NUMERIC(3,2)
subtotal NUMERIC(5,2)
total NUMERIC(5,2)
FK idTipoComprobante TINYINT
FK idContrato SMALLINT
Contrato
PK idContrato SMALLINT
Fecha DATE
Hora TIME
FK idPropuesta SMALLINT
Propuesta
PK idPropuesta SMALLINT
fecha DATE
hora TIME
costo NUMERIC(5,2)
FK idSolicitud SMALLINT
EstibadorServicio
PK idEstServ SMALLINT
FK idEstibador TINYINT
FK idServicio SMALLINT
Anulacion
PK idAnulacion SMALLINT
fecha DATE
hora TIME(0)
montoDscto NUMERIC(5,2)
FK idContrato SMALLINT
FK idAdmin TINYINT
TipoComprobante
PK idTipoComprabante CHAR(1)
nombre VARCHAR(20)
Servicio
PK idServicio SMALLINT
fecha DATE
hora TIME
FK idContrato SMALLINT
FK idAdmin TINYINT
Estibador
PK idEstibador TINYINT
FK idPersona SMALLINT
Administrador
PK idAdministrador TINYINT
FK idPersona SMALLINT
CamionServicio
PK idCamServ SMALLINT
FK idCamion TINYINT
FK idServicio SMALLINT
FK idConductor TINYINT
DetalleServicio
PK idDetalleServicio SMALLINT
descripcion VARCHAR(150)
cantidad TINYINT
estado VARCHAR(150)
tipo VARCHAR(20)
FK idServicio SMALLINT
FK idGuia SMALLINT
Guía
PK idGuia SMALLINT
U serieGuia CHAR(4)
U numGuia CHAR(8)
fechaTraslado DATE
motivoTraslado VARCHAR(50)
dirPartida VARCHAR(150)
dirLlegada VARCHAR(150)
Conductor
PK idConductor TINYINT
FK idPersona SMALLINT
U numLicencia CHAR(15)
Camión
PK idCamion TINYINT
descripcion VARCHAR(150)
pesoNeto_KG NUMERIC(5,2)
pesoBruto_KG NUMERIC(5,2)
capacidad_TN NUMERIC(5,2)
U numPlaca CHAR(7)
modelo VARCHAR(10)
Persona
PK idPersona SMALLINT
nombre VARCHAR(30)
apPat VARCHAR(20)
apMat VARCHAR(20)
U dni CHAR(8)
celular CHAR(9)
correo VARCHAR(50)
direccion VARCHAR(150)

IV. 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

V. ANEXOS

5.1. 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
)
```

5.2. Creación de Claves Foráneas

5.2.1. PersonaNatural

```
ALTER TABLE personaNatural
ADD CONSTRAINT FK_Cliente_PersNatural
FOREIGN KEY (idCliente)
REFERENCES cliente(idCliente)
```

5.2.2. PersonaJuridica

```
ALTER TABLE personaJuridica
ADD CONSTRAINT FK_Cliente_PersJuridica
FOREIGN KEY (idCliente)
REFERENCES cliente(idCliente)
```

5.2.3. Cliente

```
ALTER TABLE cliente
CONSTRAINT FK_Categoria_Cliente
FOREIGN KEY (idCategoria)
REFERENCES Categoria(idCategoria)
```

5.2.4. 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)
```

5.2.5. Propuesta

```
ALTER TABLE propuesta
ADD CONSTRAINT FK_Solicitud_Propuesta
FOREIGN KEY (idSolicitud)
REFERENCES Solicitud(idSolicitud)
```

5.2.6. Contrato

```
ALTER TABLE contrato
ADD CONSTRAINT FK_Propuesta_Contrato
FOREIGN KEY (idPropuesta)
REFERENCES propuesta(idPropuesta)
```

5.2.7. 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)
```

5.2.8. 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)
```

5.2.9. 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)
```

5.2.10. 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)
```

5.2.11. Administrador

```
ALTER TABLE administrador
ADD CONSTRAINT FK_Persona_Admin
FOREIGN KEY (idPersona)
REFERENCES Persona(idPersona)
```

5.2.12. Estibador

```
ALTER TABLE estibador
ADD CONSTRAINT FK_Persona_Estibador
FOREIGN KEY (idPersona)
REFERENCES Persona(idPersona)
```

5.2.13. Conductor

```
ALTER TABLE conductor ADD CONSTRAINT FK_Persona_Conductor
FOREIGN KEY (idPersona)
REFERENCES Persona(idPersona)
```

5.2.14. 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)
```

5.2.15. Provincia

```
ALTER TABLE provincia ADD CONSTRAINT FK_Dep_Prov
FOREIGN KEY (idDepartamento)
REFERENCES departamento(idDepartamento)
```

5.2.16. Distrito

```
ALTER TABLE distrito ADD CONSTRAINT FK_Prov_Dist
FOREIGN KEY (idProvincia)
REFERENCES provincia(idProvincia)
```

5.3. Creación de Restricciones

5.3.1. PersonaNatural

```
ALTER TABLE personaNatural
ADD CONSTRAINT CK_dniNumeros CHECK(dni LIKE '%[0-9]'),
CONSTRAINT CK_dniOcho CHECK(len(dni) = 8)
```

5.3.2. PersonaJuridica

```
ALTER TABLE personaJuridica
ADD CONSTRAINT CK_rucNumeros CHECK(ruc LIKE '%[0-9]'),
CONSTRAINT CK_rucOnce CHECK(len(ruc) = 11)
```

5.3.3. Cliente

```
ALTER TABLE cliente
ADD CONSTRAINT CK_telCliente CHECK(telefono LIKE '%[0-9]')
```

5.3.4. Propuesta

```
ALTER TABLE propuesta
ADD CONSTRAINT CK_costoPropuesta CHECK(costo > 0)
```

5.3.5. 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)
```

5.3.6. 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)
```

5.3.7. Guía

```
ALTER TABLE guia
ADD CONSTRAINT UQ_guia UNIQUE(serieGuia,numGuia)
```

5.3.8. 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)
```

5.4. 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:

5.4.1. 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)
```

5.4.2. 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)
```

5.4.3. 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)
```

5.4.4. 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)
```

5.4.5. 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)
```

5.4.6. 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)
```

5.4.7. 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)
```

5.4.8. 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)
```

5.4.9. 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)
```

5.4.10. 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)
```

5.4.11. 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)
```

5.5. Creación de Procedimientos Almacenados

5.5.1. 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
                  INSERT INTO administrador (idPersona)
                  VALUES (@idPersona);
                  PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ADMINISTRADOR'
           END
    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
           PRINT 'INGRESE UN PARÁMETRO CORRECTO: (I) --> INSERTAR Ó (U) --> ACTUALIZAR'
    END;
```

5.5.2. 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
                  INSERT INTO anulacion (montoDescuento, idContrato, idAdmin, fecha,
hora)
                  VALUES (@montoDescuento, @idContrato, @idAdmin, @fecha, @hora);
                  PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ANULACIÓN'
           END
    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;
```

5.5.3. 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;
```

5.5.4. 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
                INSERT INTO camion_Servicio (idCamion, idServicio, idConductor)
                VALUES (@idCamion, @idServicio, @idConductor);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CAMIÓN-SERVICIO'
          END
    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;
```

5.5.5. 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
            INSERT INTO categoria (nombre, descripcion)
            VALUES (@nombre, @descripcion);
            PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CATEGORÍA'
        END
    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;
```

5.5.6. 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
            INSERT INTO cliente (direccion, celular, correo, idCategoria)
            VALUES (@direccion, @celular, @correo, @idCategoria);
            PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CLIENTE'
        END
    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;
```

5.5.7. 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;
```

5.5.8. 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
                INSERT INTO conductor (idPersona, numLicencia)
                VALUES (@idPersona, @numLicencia);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CONDUCTOR'
          END
    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;
```

5.5.9. 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
                INSERT INTO contrato (fecha, hora, idPropuesta)
                VALUES (@fecha, @hora, @idPropuesta);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA CONTRATO'
          END
    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;
```

5.5.10. 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);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA DETALLESERVICIO'
          END
    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;
```

5.5.11. 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
                INSERT INTO estibador (idPersona)
                VALUES (@idPersona);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ESTIBADOR'
          END
    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;
```

5.5.12. 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
                INSERT INTO estibador_Servicio (idEstServ, idServicio,
idEstibador)
                VALUES (@idEstServ, @idServicio, @idEstibador);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA ESTIBADOR-SERVICIO'
          END
    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;
```

5.5.13. 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;
```

5.5.14. 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
			INSERT INTO persona (nombre, apPat, apMat, dni, celular, correo,
direccion)
			VALUES (@nombre, @apPat, @apMat, @dni, @celular, @correo,
@direccion);
			PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PERSONA'
		END
    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;
```

5.5.15. 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
			INSERT INTO personaJuridica (razonSocial, ruc, idCliente)
			VALUES (@razonSocial, @ruc, @idCliente);
			PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PERSONA JURIDICA'
		END
    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;
```

5.5.16. 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
            INSERT INTO personaNatural (nombre, apPat, apMat, dni, idCliente)
            VALUES (@nombre, @apPat, @apMat, @dni, @idCliente);
            PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PERSONA NATURAL'
        END
    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;
```

5.5.17. 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
```

```
                INSERT INTO propuesta (fecha, hora, costo, idSolicitud)
                VALUES (@fecha, @hora, @costo, @idSolicitud);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA PROPUESTA'
        END
    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;
```

5.5.18. 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
                INSERT INTO servicio (fecha, hora, idContrato, idAdmin)
                VALUES (@fecha, @hora, @idContrato, @idAdmin);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA SERVICIO'
        END
    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;
```

5.5.19. 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;
```

5.5.20. 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
                INSERT INTO tipoComprobante (idTipoComprobante, nombre)
                VALUES (@idTipoComprobante, @nombre);
                PRINT 'SE INSERTÓ UN REGISTRO EN LA TABLA TIPO COMPROBANTE'
          END
    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;
```

5.6. *Creación de Funciones Escalares*

5.6.1. 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
                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
          BEGIN
                SET @montoTotal = NULL; -- Tipo de comprobante no válido
          END

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

5.6.2. 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;
```

5.7. Creación de Vistas

5.7.1. 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;
```

5.7.2. 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.
```

```sql
--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
          -- Si el campo idPersNatural en la tabla personaNatural no es nulo,
muestra
          --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,
muestra
          --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;
```

5.8. Creación de Auditorías y Triggers

5.8.1. 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;
```

5.8.2. 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;
```

5.9. *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
                            INSERT INTO [dbo].[personaJuridica] ([razonSocial],
[ruc], [idCliente])
                            VALUES (@razonSocial, @ruc, @idCliente);
                    END

        -- 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;
```

5.10. 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;

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

exec paRecorrerSolicitud
```