Carrera:

Ingeniería en Sistemas de Computación

Materia:

SC-404 Fundamentos de Diseño de Base de Datos Relacionales

Profesor:

Prof. Lic. Max J. Bermúdez León

I CUATRIMESTRE 2022 - PROYECTO FINAL

Miembros:

Kenneth Arroyo Vargas

Josué David Inces Cascante

Randall José Portuguez Hernández

Abril del 2022

ÍNDICE

INTRODUCCIÓN

En el presente trabajo se realizará la implementación de una base de datos según las especificaciones del enunciado del proyecto. En el cual se contemplan las buenas prácticas para la construcción de una base de datos optima, la misma será programa en lenguaje de mysql con la administración de la aplicación Workbench.

La creación de la base de datos se realizó para que cumpla con la tercera regla de normalización, para esto se tomó en cuenta las tablas presentadas por enunciado y se aplicaran los cambios necesarios para cumplir con la regla de normalización, además de esto se confecciono un estándar para el nombramiento de tablas y datos, lo cual se explicara más afondo en el desarrollo del proyecto.

Contemplando las tablas y los datos que se relacionan entre sí, se confecciono un modelo lógico de la base datos con las relaciones según las llaves foráneas, para esto se utilizó la funcionalidad que nos brinda Workbench que nos permite crear el modelo a partir del script de creación de la base de datos.

Se realizó la creación de procedimientos almacenados para la creación, eliminación y modificación de datos de cada una de las tablas, exceptuando las que por su funcionalidad no lo requieran, además de esto se agregaron disparadores para las tablas de auditoría de las tablas Paquetería, Facturación, Empleados y Clientes.

DESCRIPCIÓN DEL TRABAJO

Una empresa que se llama FlashBox, la cual dedica a traer paquetes desde Estados Unidos contrato a una empresa para que les desarrollara un sistema informático para la gestión del negocio.

Al parecer la empresa que contrataron se declaró en quiebra y no puedo continuar con el desarrollo del sistema.

La empresa FlashBox decidió buscar otra empresa y contrato al grupo formado por estudiantes de la materia de Fundamentos de Diseños de Bases de Datos de la Universidad Fidélitas para se hicieran cargo del diseño de la base de datos, ya que el desarrollo los hará otra empresa.

Como parte de la documentación, se entregó un documento con las relaciones identificadas, pero las mismas al parecer no están Normalizadas.

Existe otro documento con algunos detalles que deben ser tomados en cuenta en el proceso de la Normalización, al parecer este documento fue el insumo para realizar la identificación de las Entidades y algunos de sus atributos, las relaciones generadas son las siguientes,

El documento detalla lo siguiente,

Detalle de Relaciones

EMPLEADO (IDENTIFICACION, NOMBRE_COMPLETO, FECHA_NACIMIENTO, PUESTO, SALARIO_EMPLEADO, OFICINA_ASIGNADA, UBICACION_OFICINA, DIRECCION_EMPLEADO, TELEFONO_1, TELEFONO_2, CORREOS_ELECTRONICOS, ESTADO, FECHA_REGISTRO, NOMBRE_USUARIO_REGISTRO)

Empleado, se quiere registrar su número de identificación, nombre completo (Nombre, Apellido Paterno y Apellido Materno concatenado), fecha de nacimiento del empleado, puesto del empleado, salario del empleado, importante indicar que el salario depende del puesto, oficina que esta asignado, ubicación de la oficina, así mismo se ocupa el número de teléfono de la oficina, dirección de residencia del empleado, concatenando Provincia, Cantón, Distrito y otras señas, así mismo la opción registrar como mínimo 2 o más teléfonos, así como los correos electrónicos que tenga el empleado, su estado, este estado es Activo o Inactivo, la fecha que se registró y el nombre del usuario que lo registro.

Se puede explorar la opción de separar el nombre completo en Nombre – Apellido Paterno y Apellido Materno.

En el caso de la dirección del empleado se puede separar en provincia, cantón, distrito y otras señas.

En lo que se refiere al correo electrónico pueden varios correos asignados al empleado.

INVENTARIO_VEHICULOS (NUM_INTERNO, NUM_PLACA, MARCA, MODELO, STOCK, FECHA_REVISION_MANTENIMIENTO, DESCRIPCION_MANTENIMIENTO, TIPO_MANTENIMIENTO, TIPO_VEHICULO, ESTADO_VEHICULO, FECHA_REGISTRO, NOMBRE_USUARIO_REGISTRO)

Inventario_Vehiculos, se quiere registrar un código o número interno de control, el número de placa del vehículo, la marca, modelo, cantidad en inventario.

Las revisiones se hacen al menos 3 por año, por esto se registra la fecha de la revisión de mantenimiento, descripción del tipo de mantenimiento, tipo de manteniendo (Revisión General, Cambio de aceite, cambio de llantas).

Se registra el tipo de vehículo (Moto, Todo Terreno, Sedan), la fecha que se registró el vehículo y el nombre del usuario que registro el vehículo.

PAQUETERIA (NUM_PAQUETE, DESCRIPCION_PAQUETE, OFICINA_UBICACION, EMPLEADO_PROCESO, NOMBRE_CLIENTE, IDENTIFICACION_CLIENTE, TIPO_ENVIO, TELEFONO_CLIENTE_1, TELEFONO_CLIENTE_2, DIRECCIONES_CLIENTE, CORREOS_CLIENTE, TIPO_CLIENTE, ESTADO_CLIENTE, ESTADO_PAQUETE, FECHA_REGISTRO, NOMBRE_USUARIO_REGISTRO)

Se necesita llevar el control del número de paquete, descripción del paquete, la oficina que está ubicado el paquete, nombre del empleado que proceso el paquete, nombre completo del cliente, identificación del cliente, tipo de envió (Prioritario, Normal o económico), al menos 2 teléfonos del cliente, las posibles direcciones de ubicación del cliente, los correos electrónicos del cliente, tipo de cliente (VIP o Normal), estado del paquete (En Transito, En Oficina, Entregado al cliente), fecha de registro del paquete y el nombre del usuario que registro el paquete

FACTURACION_PAQUETERIA (NUM_FACTURA, NOMBRE_CLIENTE, FECHA_FACTURA, NUMERO_PAQUETE, COSTO_IMPUESTOS, COSTO_FLETE, COSTO_IV, TOTAL_POR_ENVIO, TOTAL_ANTES_IVA, TOTAL_FACTURA, DIRECCION_CLIENTE, TELEFONO_CLIENTE_1, TELEFONO_CLIENTE_2, EMAIL_CLIENTE_FE, EMAIL_CLIENTE, FECHA_REGISTRO, NOMBRE_USUARIO_REGISTRO, ESTADO)

En el caso de la facturación se necesita un encabezado y detalle.

Los datos de facturación es el Numero de factura, nombre del cliente, fecha de la factura, numero del paquete, a cada paquete se le detalla el costo de los impuestos, costo del flete.

- El total por envió se calcula sumando Impuestos + Flete.
- El total de antes IVA es la suma de los totales total por envió.
- El total de factura es el cálculo de total antes iva * total iva.

- Así mismo se ocupa la dirección del cliente o si se genera una relación de clientes se puede establecer una integridad referencia con el código del cliente, así no se ocuparían los datos de teléfonos del cliente y la dirección.
- En el caso del correo electrónico de facturación electrónica se necesita para tramitar la misma, fecha de registro de la factura y el nombre del usuario que registro la factura y el estado de la factura (Anulada, Activa).
- Es importante agregar que se pueden facturar varios paquetes en una misma factura.

Basado en el anterior enunciado, se necesita desarrollar lo siguiente:

- Análisis de las relaciones y aplicar las 3 reglar de normalización y generar las nuevas relaciones que sean necesarias. Identificar las llaves primarias y foráneas.
- Identificación de los tipos de datos por cada uno de los atributos de las relaciones resultantes del proceso de normalización. Poner atención a los tipos de datos de las llaves primarias identificadas, así como las llaves foráneas, los tipos de datos deben coincidir.
- Generación del modelo lógico de la base datos con las relaciones según las llaves foráneas.
- Generar los scripts respectivos de la creación de
- Desarrollo de los procedimientos almacenados (Store Procedures) para incluir, borrar o modificar datos en las relaciones definidas.
- Desarrollo de disparadores (Triggers) de auditoria para generar las pistas sobre las modificaciones en las tablas generadas. Dichos disparadores se ejecutan en el borrado y modificación de datos.

Normas a seguir

Se debe presentar un documento con lo siguiente.

• Portada (Nombre Universidad, Nombre de Carrera, nombre de la materia, nombre del profesor, nombre completo de los miembros del equipo de trabajo, año).

- Índice
- Introducción
- Descripción del trabajo (Todo este enunciado)
- Desarrollo del trabajo
    - Análisis del caso, se debe detallar el análisis realizado a las tablas para determinar si las tablas cumplen con las Formas Normales (hasta la 3era) o su defecto cuales fueron los pasos para lograr cumplir con las 3 Formas Normales.

    - Detalle de tablas creadas, imágenes o insertos de las tablas que resultaron del proceso de análisis de las 3FN. Importante indicar tipos de datos y restricciones.
    - Detalle de Scripts de Creación de la BD.
    - Detalle de Scripts de Creación de las tablas con sus tipos de dato y restricciones y relaciones (integridad referencial).
    - Detalle de scripts de Inserts, Delete, Update solicitados.
    - Detalle de Procedimientos Almacenados.
    - Detalle de triggers solicitados.
- Conclusiones por cada miembro del grupo.

Se debe hacer una presentación en formato power point, con el nombre completo y el detalle del análisis del caso y el detalle de las tablas que se deben crear.

DESARROLLO DEL PROYECTO

Se debe crear un archivo .sql en el cual se detalle:

- Script de creación de la bd
- Script de Creación de las tablas (Especificar restricciones, tipos de dato)
- Script de Creación de las relaciones entre tablas
- Script de creación de Procedimientos almacenados
- Script de creación de Triggers.

Se deben crear procedimiento almacenados:

- Para insertar, modificar y eliminar registros de las tablas resultantes.
- Es importante indicar que todos los procedimientos deben especificar el control de errores.

Triggers o disparadores para las tablas de auditoría de las tablas Paquetería, Facturación, Empleados y Clientes.

Tabla de Calificación

- Presentación: 5%
- Desarrollo: 10%
- Pruebas de integridad: 10%
- Defensa: 5%
- Porcentaje total: 30%

DESARROLLO

Análisis

Para la creación de la base de datos del proyecto se aplicó la tercera regla de normalización a las tablas de Empleado, Inventario_Vehiculos, Paqueteria Y Facturacion_Paqueteria, al realizar el estudio de cada una de las tablas se logró identificar cambios importantes en la estructura de los datos que contenida cada una de ellas, los cuales fueron los siguientes:

Tabla de Empleado

Empleado (Identificacion, Nombre_Completo, Fecha_Nacimiento, Puesto, Salario_Empleado, Oficina_Asignada, Ubicacion_Oficina, Direccion_Empleado, Telefono_1, Telefono_2, Correos_Electronicos, Estado, Fecha_Registro, Nombre_Usuario_Registro)

Al realizar un análisis a la tabla de empleado se logran encontrar los siguientes puntos:

- Se logró identificar datos que no tenían relación con la tabla como puestos y oficina.
- Se logró identificar datos que por su funcionalidad se debe crear una tabla para que contenga dicha información como dirección, teléfono y correo.
- Se crean nuevas tablas y se realiza las relaciones con la tabla de empleados

Contemplando los puntos anteriores se restructura la tabla dando como resultado la siguiente normalización con la tercera forma normal

- Empleado (Id_Empleado, Identificacion, Apellido_1, Apellido_2, Nombre, Fecha_Nacimiento, Puesto, Oficina_Asignada, Direccion_Empleado, Estado, Fecha_Registro, Nombre_Usuario_Registro)
- Puesto (Id_Puesto, Nom_Puesto, Salario_Puesto)
- Oficina (Id_Oficina, Nom_Oficina, Ubicacion_Oficina, Tel_Oficina)
- Direccion (Id_Direccion, Id_Empleado, Provincia, Canton, Distrito, Otras_Senas)
- Telefono (Id_Telefonos, Id_Empleado, Tel)
- Correo (Id_Correo, Id_Empleado, Correo)

Tabla De Inventario_Vehiculos

Inventario_Vehiculos (Num_Interno, Num_Placa, Marca, Modelo, Stock, Fecha_Revision_Mantenimiento, Descripcion_Mantenimiento, Tipo_Mantenimiento, Tipo_Vehiculo, Estado_Vehiculo, Fecha_Registro, Nombre_Usuario_Registro)

Al realizar un análisis a la tabla de inventario vehículo se logran encontrar los siguientes puntos:

- Se logró identificar datos que no tenían relación con la tabla como los de mantenimiento

- Se logró identificar datos que por su funcionalidad se debe crear una tabla para que contenga dicha información como marca, modelo, tipo de vehículos entre otros.
- Se crean nuevas tablas y se realiza las relaciones con la tabla de mantenimiento, tipo de vehículo entre otros.

Contemplando los puntos anteriores se restructura la tabla dando como resultado la siguiente normalización con la tercera forma normal

- Inventario_Vehiculos (Num_Interno, Id_Tipo_Vehiculo, Num_Placa, Marca, Modelo, Stock, Estado_Vehiculo, Fecha_Registro, Nombre_Usuario_Registro)
- Revision(Id_Revision, Id_Vehiculo, Fecha_Revision_Mantenimiento, Descripcion_Mantenimiento, Tipo_Mantenimiento)
- Tiposvehiculos(Id_Tipo_Vehiculo, Tipo_Vehiculo)

Tabla De Paqueteria

Paqueteria (Num_Paquete, Descripcion_Paquete, Oficina_Ubicacion, Empleado_Proceso, Nombre_Cliente, Identificacion_Cliente, Tipo_Envio, Telefono_Cliente_1, Telefono_Cliente_2, Direcciones_Cliente, Correos_Cliente, Tipo_Cliente, Estado_Cliente, Estado_Paquete, Fecha_Registro, Nombre_Usuario_Registro)

Al realizar un análisis a la tabla de paquetería se logran encontrar los siguientes puntos:

- Se logró identificar datos que no tenían relación con la tabla como la información del cliente y de empleado
- Se logró identificar datos que por su funcionalidad se debe crear una tabla para que contenga dicha información como estado paquete.
- Se crean nuevas tablas y se realiza las relaciones con la tabla de empleados, clientes entre otras.

Contemplando los puntos anteriores se restructura la tabla dando como resultado la siguiente normalización con la tercera forma normal

- Tab_Paqueteria (Num_Paquete, Descripcion_Paquete, Fecha_Registro, Identificacion_Cliente, Identificacion_Empleado, Id_Oficina, Tipo_Envio, Id_Estadopaquete, Id_Usuario)
- Tab_Oficina (Id_Oficina, Nombre, Oficina_Ubicacion)
- Tab_Cliente(Identificacion_Cliente, Nombre_Cliente, Id_Tipo)
- Tab_Telefonos_Cliente(Id_Telefono, Telefonos_Cliente, Identificacion_Cliente)
- Tab_Direcciones_Clientes(Id_Dirreciones, Dirección, Identificacion_Cliente)
- Tab_Correos_Cliente (Id_Correocliente, Correo, Identificacion_Cliente)
- Tab_Tipo_Cliente (Id_Tipo, Nombre)
- Tab_Estado_Paquete (Id_Estadopaquete, Nombre)

- Tab_Tipo_Envio(Id_Tipoenvio, Nombre)
- Tab_Empleado_Proceso(Identificacion_Empleado, Nombre, Proceso)
- Tab_Usuario_Registro(Id_Usuario, Nombre)

Tabla De Facturacion Paqueteria

Facturacion_Paqueteria(Num_Factura, Nombre_Cliente, Fecha_Factura, Numero_Paquete, Costo_Impuestos, Costo_Flete, Costo_Iv, Total_Por_Envio, Total_Antes_Iva, Total_Factura, Direccion_Cliente, Telefono_Cliente_1, Telefono_Cliente_2, Email_Cliente_Fe, Email_Cliente, Fecha_Registro, Nombre_Usuario_Registro, Estado)

Al realizar un análisis a la tabla de Facturacion Paqueteria se logran encontrar los siguientes puntos:

- Se logro identificar datos que no tenían relación con la tabla como la información del cliente y de empleado según lo solicitado en el enunciado del proyecto
- Se logro identificar datos que por su funcionalidad se debe crear una tabla para que contenga dicha información como estado.
- Se crean nuevas tablas y se realiza las relaciones con la tabla de Facturacion Paqueteria.

Contemplando los puntos anteriores se restructura la tabla de Facturacion Paqueteria dando como resultado la siguiente normalización con la tercera forma normal

- Facturacion_paqueteria(Num_Factura, Fecha_Factura, Numero_Paquete, Costo_Impuestos, Costo_Flete, Costo_IV, Total_Por_Envio, Total_Antes_IVA, Total_Factura, Email_Cliente_FE, Estado)
- Datos_Cliente(Nombre_Cliente, Nombre_Usuario_Registro)
- Telefonos_Cliente(id_Telefonos, Telefono_Cliente_1, Telefono_Cliente_2, Nombre_Cliente)
- Direcciones_Cliente(Direccion_Cliente, Nombre_Cliente)
- Estado_factura(id_EstadoFactura, Nombre_Cliente, Nombre_Usuario_Registro)
- Correos_Cliente(Email_Cliente_FE, Email_Cliente, Nombre_Cliente, Nombre_Usuario_Registro)

Análisis final

Al contemplar cada una de las tablas en conjunto se logró de terminar que varios de los datos se requerían en más de una tabla por lo que nos dio como resultado la siguiente lista de tablas para crear en la base de datos.

- **tab_empleados**
- **tab_puestos**
- **tab_oficinas**
- **tab_direcciones**
- **tab_telefonos**
- **tab_correos**
- **tab_inventario_vehiculos**
- **tab_tipo_vehiculo**
- **tab_tipo_mantenimiento_vehiculo**
- **tab_revision_vehiculos**
- **tab_paqueteria**
- **tab_tipo_envio_paqueteria**
- **tab_estado_paquete_paqueteria**
- **tab_clientes**
- **tab_tipo_cliente_paqueteria**
- **tab_facturacion_paqueteria**

Nota: Más adelante del documento se espaciará los datos y el tipo que contendrán cada una de las tablas mencionadas anteriormente.

Detalle de tablas creadas

Tabla de clientes

tab_clientes				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
cliente_identificacion	int	No	Si	No
id_tipo_cliente_paqueteria	int	No	No	Si
cliente_apellido_1	varchar(15)	No	No	No
cliente_apellido_2	varchar(15)	No	No	No
cliente_nombre	varchar(15)	No	No	No

Tabla de correos

tab_correos				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_correo	int (autoincrementable)	No	Si	No
identificacion_usuario_correo	varchar(15)	No	No	Si
correo	varchar(25)	No	No	No

Tabla de direcciones

tab_direcciones				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_direccion	int (autoincrementable)	No	Si	No
identificacion_usuario_direccion	int	No	No	Si
direccion_provincia	varchar(15)	No	No	No
direccion_canton	varchar(15)	No	No	No
direccion_distrito	varchar(15)	No	No	No
direccion_otras_senas	varchar(100)	No	No	No

Tabla de empleados

tab_empleados				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_empleado	int (autoincrementable)	No	Si	No
id_empleado_puesto	int	No	No	Si
id_empleado_oficina_asignada	int	No	No	Si
id_empleado_direccion	int	No	No	Si
id_empleado_correo	int	No	No	Si
empleado_estado	varchar(10)	No	No	No
empleado_fecha_registro	fecha	No	No	No
empleado_nombre_usuario_registro	varchar(15)	No	No	No
empleado_identificacion	varchar(15)	No	No	No
empleado_apellido_1	varchar(15)	No	No	No
empleado_apellido_2	varchar(15)	No	No	No
empleado_nombre	varchar(15)	No	No	No
empleado_fecha_nac	fecha	No	No	No

Tabla de estado de paquete

tab_estado_paquete_paqueteria				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_estado_paquete_paqueteria	int (autoincrementable)	No	Si	No
estado_paquete_paqueteria	varchar(25)	No	No	No

Tabla de oficinas

tab_oficinas				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_oficina	int (autoincrementable)	No	Si	No
oficina_nombre	varchar(15)	No	No	No
oficina_ubicacion	varchar(25)	No	No	No
oficina_telefono	int	No	No	No

Tabla de facturación

tab_facturacion_paqueteria				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_factura	int (autoincrementable)	No	Si	No
id_cliente_factura	int	No	No	Si
factura_numero	int	No	No	No
factura_fecha	date	No	No	No
factura_costo_impuestos	float	No	No	No
factura_costo_flete	float	No	No	No
factura_costo_iva	float	No	No	No
factura_total_por_envio	float	No	No	No
factura_total_antes_iva	float	No	No	No
factura_total	float	No	No	No
factura_estado	varchar(15)	No	No	No
factura_usuario_registro	varchar(15)	No	No	No

Tabla de inventario de vehículos

tab_inventario_vehiculos				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_inventario_vehiculo	int (autoincrementable)	No	Si	No
id_tipo_mantenimiento_vehiculo	int	Si	No	Si
id_tipo_vehiculo	int	No	No	Si
vehiculo_num_interno	int	No	No	No
vehiculo_num_placa	varchar(15)	No	No	No
vehiculo_marca	varchar(15)	No	No	No
vehiculo_modelo	varchar(15)	No	No	No
vehiculo_stock	varchar(15)	No	No	No
vehiculo_estado	varchar(15)	No	No	No
vehiculo_fecha_registro	fecha	No	No	No

Tabla de paquetería

tab_paqueteria				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_paqueteria	int (autoincrementable)	No	Si	No
id_cliente	int	No	No	Si
id_oficina	int	No	No	Si
id_empleado_proceso	int	No	No	Si
id_tipo_envio_paqueteria	int	No	No	Si
id_estado_paquete_paqueteria	int	No	No	Si
paqueteria_numero	int	No	No	No
paqueteria_usuario_registro	varchar(15)	No	No	No
paqueteria_descripcion	varchar(100)	No	No	No
paqueteria_fecha_registro	date	No	No	No

Tabla de puestos

tab_puestos				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_puesto	int (autoincrementable)	No	Si	No
puesto_nombre	varchar(15)	No	No	No
puesto_salario	float	No	No	No

Tabla de revisión de vehículos

tab_revision_vehiculos				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_revision_vehiculo	int (autoincrementable)	No	Si	No
id_inventario_vehiculo	int	No	No	Si
id_tipo_mantenimiento_vehiculo	int	No	No	Si
revision_fecha_mantenimiento	date	No	No	No
revision_descripcion_mantenimiento	varchar(15)	No	No	No

Tabla de teléfonos

tab_telefonos				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_telefono	int (autoincrementable)	No	Si	No
identificacion_usuario_telefono	varchar(15)	No	No	Si
telefono	int	No	No	No

Tabla de tipo de cliente

tab_tipo_cliente_paqueteria				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_tipo_cliente_paqueteria	int (autoincrementable)	No	Si	No
tipo_cliente_paqueteria	varchar(15)	No	No	No

Tabla de tipo de envió

tab_tipo_envio_paqueteria				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_tipo_envio_paqueteria	int (autoincrementable)	No	Si	No
tipo_envio_paqueteria	varchar(15)	No	No	No

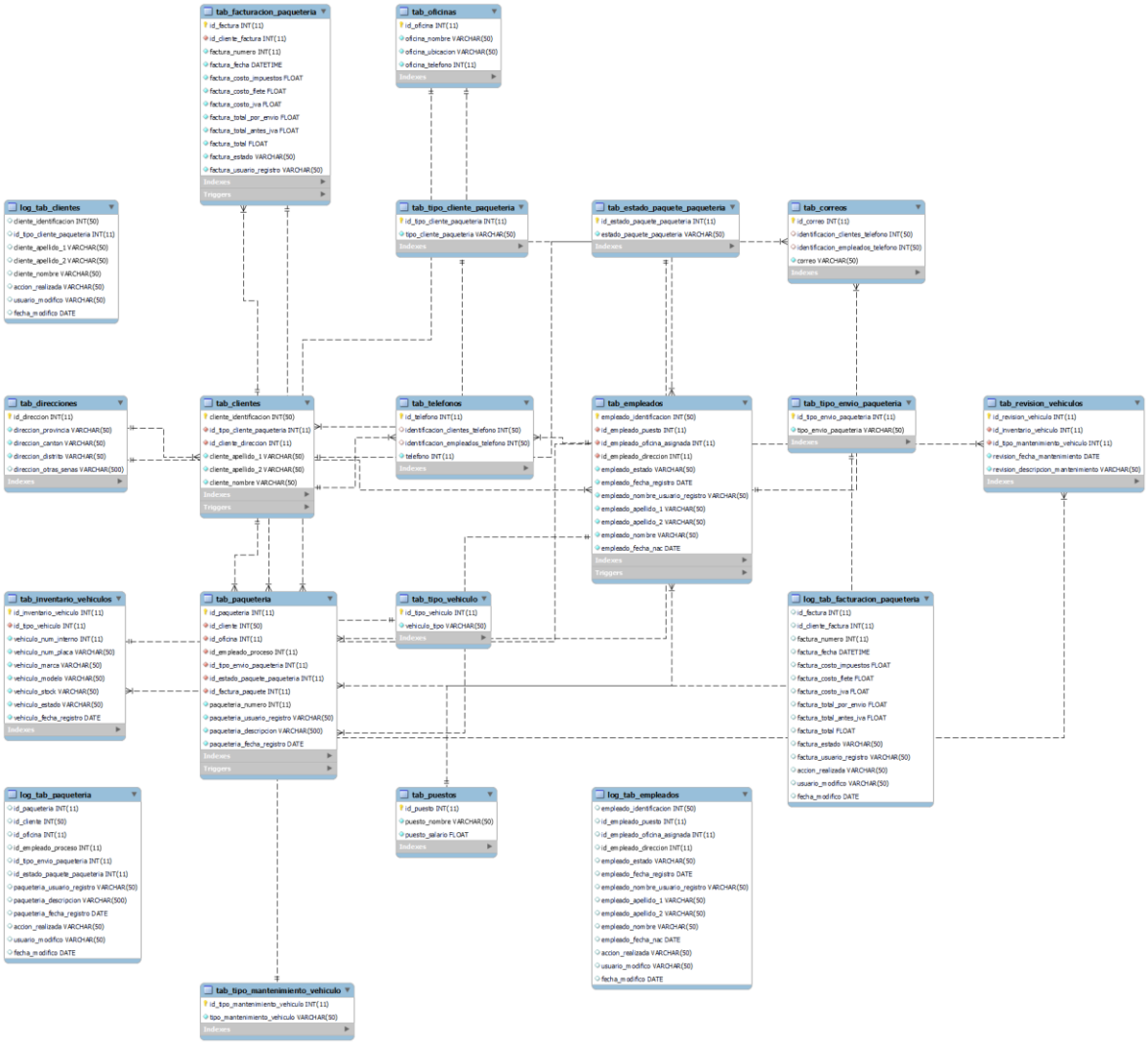
Tabla de tipo de mantenimiento

tab_tipo_mantenimiento_vehiculo				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_tipo_mantenimiento_vehiculo	int (autoincrementable)	No	Si	No
tipo_mantenimiento_vehiculo	varchar(15)	No	No	No

Tabla de tipo de vehículo

tab_tipo_vehiculo				
Campo	**Tipo de dato**	**Null**	**PK**	**FK**
id_tipo_vehiculo	int (autoincrementable)	No	Si	No
vehiculo_tipo	varchar(15)	No	No	No

Modelo de entidad relación



Scripts de Creación de la BD

En la siguiente imagen se muestran los comandos que se utilizaron para crear la base de datos de nuestro proyecto, tomando en cuenta las mejores prácticas a la hora de crear una base de datos funcional.

```
CREATE DATABASE IF NOT EXISTS BDProyecto;
USE BDProyecto;
```

Scripts de Creación de las tablas

Creación de tabla de tab_puestos

Funcionalidad: Almacenar la información de los puestos de la empresa de paquetería

```
CREATE TABLE IF NOT EXISTS tab_puestos(
        id_puesto int auto_increment primary key not null,
        puesto_nombre varchar(50) not null,
        puesto_salario float not null);
```

Creación de tabla tab_oficinas

Funcionalidad: Almacenar la información de las oficinas de la empresa de paquetería

```
CREATE TABLE IF NOT EXISTS tab_oficinas(
        id_oficina int auto_increment primary key not null,
        oficina_nombre varchar(50) not null,
        oficina_ubicacion varchar(50) not null,
        oficina_telefono int not null
);
```

Creación de tabla tab_direciones

Funcionalidad: Almacenar la información de las direcciones de los empleados de la empresa de paquetería

```
CREATE TABLE IF NOT EXISTS tab_direcciones(
        id_direccion int auto_increment primary key not null,
        direccion_provincia varchar(50) not null,
        direccion_canton varchar(50) not null,
        direccion_distrito varchar(50) not null,
        direccion_otras_senas varchar(500)
);
```

Creación de tabla tab_empleados

Funcionalidad: Almacena la información del empleado de la empresa de paquetería.

Relaciones con tablas: La tabla de empleados cuenta con una relación con las siguientes tablas:

- tab_puestos
- tab_oficinas
- tab_dirrecciones

```
CREATE TABLE IF NOT EXISTS tab_empleados(
        empleado_identificacion int(50) primary key not null,
    id_empleado_puesto int not null,
    id_empleado_oficina_asignada int not null,
    id_empleado_direccion int not null,
    empleado_estado varchar(50) not null,
    empleado_fecha_registro date not null,
    empleado_nombre_usuario_registro varchar(50) not null,
    empleado_apellido_1 varchar(50) not null,
        empleado_apellido_2 varchar(50) not null,
    empleado_nombre varchar(50) not null,
    empleado_fecha_nac date not null,
    foreign key(id_empleado_puesto) references tab_puestos(id_puesto),
    foreign key(id_empleado_oficina_asignada) references tab_oficinas(id_oficina),
    foreign key(id_empleado_direccion) references tab_direcciones(id_direccion)
);
```

Creación de tabla tab_correos

Funcionalidad: Almacena la información de los correos del empleado de la empresa de paquetería, se agrega la funcionalidad de ON DELETE CASCAE para que se elimine la información en conjunto con el empleado relacionado.

Relaciones con tablas: La tabla de correos cuenta con una relación con la siguiente tabla:

- tab_empleados

```
CREATE TABLE IF NOT EXISTS tab_correos(
        id_correo int auto_increment primary key not null,
        identificacion_usuario_correo int(50) not null,
        correo varchar(50) not null,
    foreign key (identificacion_usuario_correo) references tab_empleados(empleado_identificacion) ON DELETE CASCADE
);
```

Creación de tabla tab_telefonos

Funcionalidad: Almacena la información de los teléfonos del empleado de la empresa de paquetería, se agrega la funcionalidad de ON DELETE CASCAE para que se elimine la información en conjunto con el empleado relacionado.

Relaciones con tablas: La tabla de correos cuenta con una relación con la siguiente tabla:

- tab_empleados

```
CREATE TABLE IF NOT EXISTS tab_telefonos(
        id_telefono int auto_increment primary key not null,
        identificacion_usuario_telefono int(50) not null,
        telefono int not null,
    foreign key (identificacion_usuario_telefono) references tab_empleados(empleado_identificacion) ON DELETE CASCADE
);
```

Creación de tabla tab_tipo_vehiculo

Funcionalidad: Almacena la información de los tipos de vehículo con los que cuenta la empresa.

```
CREATE TABLE IF NOT EXISTS tab_tipo_vehiculo(
        id_tipo_vehiculo int auto_increment primary key not null,
        vehiculo_tipo varchar(50) not null
);
```

Creación de tabla tab_inventario_vehiculos

Funcionalidad: Almacena la información de los vehículos los que cuenta la empresa para crear un inventario de estos.

Relaciones con tablas: La tabla de inventarios de vehículos cuenta con una relación con la siguiente tabla:

- tab_tipo_vehiculo

```
CREATE TABLE IF NOT EXISTS tab_inventario_vehiculos(
        id_inventario_vehiculo int auto_increment primary key not null,
        id_tipo_vehiculo int not null,
        vehiculo_num_interno int not null,
        vehiculo_num_placa varchar(50) not null,
        vehiculo_marca varchar(50) not null,
        vehiculo_modelo varchar(50) not null,
        vehiculo_stock varchar(50) not null,
        vehiculo_estado varchar(50) not null,
        vehiculo_fecha_registro date not null,
    foreign key(id_tipo_vehiculo) references tab_tipo_vehiculo(id_tipo_vehiculo)
);
```

Creación de tabla tab_tipo_vehiculo

Funcionalidad: Almacenar los tipos de manteamientos que reciben los vehículos de la empresa.

```
CREATE TABLE IF NOT EXISTS tab_tipo_mantenimiento_vehiculo(
        id_tipo_mantenimiento_vehiculo int auto_increment primary key not null,
        tipo_mantenimiento_vehiculo varchar(50) not null
    );
```

Creación de tabla tab_revision_vehiculos

Funcionalidad: Almacenar las revisiones que reciben los vehículos de la empresa.

Relaciones con tablas: La tabla de revisión de vehículos cuenta con relación con las siguientes tablas:

- tab_tipo_vehiculo
- tab_inventario_vehiculos

```
CREATE TABLE IF NOT EXISTS tab_revision_vehiculos(
        id_revision_vehiculo int auto_increment primary key not null,
        id_inventario_vehiculo int not null,
        id_tipo_mantenimiento_vehiculo int not null,
        revision_fecha_mantenimiento date not null,
        revision_descripcion_mantenimiento varchar(50) not null,
    foreign key (id_inventario_vehiculo) references tab_inventario_vehiculos(id_inventario_vehiculo),
    foreign key(id_tipo_mantenimiento_vehiculo) references tab_tipo_mantenimiento_vehiculo(id_tipo_mantenimiento_vehiculo)
);
```

Creación de tabla tab_tipo_cliente_paqueteria

Funcionalidad: Almacenar los tipos de clientes de envíos de paquetes con los que cuenta la empresa.

```
CREATE TABLE IF NOT EXISTS tab_tipo_cliente_paqueteria(
        id_tipo_cliente_paqueteria int auto_increment primary key not null,
        tipo_cliente_paqueteria varchar(50) not null
    );
```

Creación de tabla tab_clientes

Funcionalidad: Almacena la información de los clientes con los que cuenta la empresa.

Relaciones con tablas: La tabla de clientes cuenta con una relación con la siguiente tabla:

- tab_tipo_cliente_paqueteria

```
CREATE TABLE IF NOT EXISTS tab_tipo_envio_paqueteria(
        id_tipo_envio_paqueteria int auto_increment primary key not null,
        tipo_envio_paqueteria varchar(50) not null
);
```

Creación de tabla tab_tipo_envio_paqueteria

Funcionalidad: Almacena los tipos de envíos de paquetes con los que cuenta la empresa.

```
CREATE TABLE IF NOT EXISTS tab_tipo_envio_paqueteria(
        id_tipo_envio_paqueteria int auto_increment primary key not null,
        tipo_envio_paqueteria varchar(50) not null
);
```

Creación de tabla tab_estado_paquete_paqueteria

Funcionalidad: Almacena los tipos de estado en los que se puede encontrar un paquete

```
CREATE TABLE IF NOT EXISTS tab_estado_paquete_paqueteria(
        id_estado_paquete_paqueteria int auto_increment primary key not null,
        estado_paquete_paqueteria varchar(50) not null
);
```

Creación de tabla tab_facturacion_paqueteria

Funcionalidad: Almacena la información del paquete para crear la factura que se le debe entregar al cliente

Relaciones con tablas: La tabla de clientes cuenta con una relación con la siguiente tabla:

- tab_clientes

```
CREATE TABLE IF NOT EXISTS tab_facturacion_paqueteria(
        id_factura int auto_increment primary key not null,
        id_cliente_factura int not null,
        factura_numero int not null,
        factura_fecha datetime not null,
        factura_costo_impuestos float not null,
        factura_costo_flete float not null,
        factura_costo_iva float not null,
        factura_total_por_envio float not null,
        factura_total_antes_iva float not null,
        factura_total float not null,
        factura_estado varchar(50) not null,
        factura_usuario_registro varchar(50) not null,
    foreign key (id_cliente_factura) references tab_clientes(cliente_identificacion)
);
```

Creación de tabla tab_paqueteria

Funcionalidad: Almacena la información de los paquetes de la empresa

Relaciones con tablas: La tabla de paquetería cuenta con relación con las siguientes tablas:

- tab_clientes
- tab_oficinas
- tab_empleados
- tab_tipo_envio_paqueteria
- tab_estado_paquete_paqueteria
- tab_facturacion_paqueteria

```
CREATE TABLE IF NOT EXISTS tab_paqueteria(
        id_paqueteria int auto_increment primary key not null,
        id_cliente int(50) not null,
        id_oficina int not null,
        id_empleado_proceso int not null,
        id_tipo_envio_paqueteria int not null,
        id_estado_paquete_paqueteria int not null,
    id_factura_paquete int not null,
        paqueteria_numero int not null,
        paqueteria_usuario_registro varchar(50) not null,
        paqueteria_descripcion varchar(500) not null,
        paqueteria_fecha_registro date not null,
    foreign key (id_cliente) references tab_clientes(cliente_identificacion),
    foreign key (id_oficina) references tab_oficinas(id_oficina),
    foreign key (id_empleado_proceso) references tab_empleados(empleado_identificacion),
    foreign key (id_tipo_envio_paqueteria) references tab_tipo_envio_paqueteria(id_tipo_envio_paqueteria),
    foreign key (id_estado_paquete_paqueteria) references tab_estado_paquete_paqueteria(id_estado_paquete_paqueteria),
    foreign key (id_factura_paquete) references tab_facturacion_paqueteria(id_factura)
);
```

Procedimientos Almacenados.

A continuación de se detallan los procedimientos almacenados creados para la administración de las tablas creadas para la base de datos.

Procedimientos almacenados de la tabla tab_ tipo_mantenimiento_vehiculo

Insertar registro de tipo de manteamiento de vehículo

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_tipo_mantenimiento_vehiculo
(
in tipo_mantenimiento_vehiculo varchar(50)
)
BEGIN
insert into tab_tipo_mantenimiento_vehiculo(tipo_mantenimiento_vehiculo)
                        values(tipo_mantenimiento_vehiculo);
END$$
Delimiter ;
```

Actualizar registro de tipo de manteamiento de vehículo

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_tipo_mantenimiento_vehiculo
(
in id_tipo int,
in tipo_mantenimiento varchar(50)
)
BEGIN
UPDATE tab_tipo_mantenimiento_vehiculo SET tipo_mantenimiento_vehiculo=tipo_mantenimiento
WHERE id_tipo_mantenimiento_vehiculo=id_tipo;
END$$
Delimiter ;
```

Eliminar registro de tipo de manteamiento de vehículo

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_tipo_mantenimiento_vehiculo
(
in id_tip int
)
BEGIN
DELETE FROM tab_tipo_mantenimiento_vehiculo WHERE id_tipo_mantenimiento_vehiculo=id_tip;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_tipo_vehiculo

Insertar registro de tipo de vehículo

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_tipo_vehiculo
(
in vehiculo_tipo varchar(50)
)
BEGIN
insert into tab_tipo_vehiculo(vehiculo_tipo)
                        values(vehiculo_tipo);
END$$
Delimiter ;
```

Actualizar registro de tipo de vehículo

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_tipo_vehiculo
(
in id_tipo int,
in vehiculo varchar(50)
)
BEGIN
UPDATE tab_tipo_vehiculo SET vehiculo_tipo=vehiculo WHERE id_tipo_vehiculo=id_tipo;
END$$
Delimiter ;
```

Eliminar registro de tipo de vehículo

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_tipo_vehiculo
(
in id_tipo int
)
BEGIN
DELETE FROM tab_tipo_vehiculo WHERE id_tipo_vehiculo=id_tipo;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tap_tipo_envio_paqueteria

Insertar registro de tipo de envió de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_tipo_envio_paqueteria
(
in tipo_envio_paqueteria varchar(50)
)
BEGIN
insert into tab_tipo_envio_paqueteria(tipo_envio_paqueteria)
                        values(tipo_envio_paqueteria);
END$$
Delimiter ;
```

Modificar registro de tipo de envió de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_tipo_envio_paqueteria
(
in id_tipo_envioa int,
in tipo_envio varchar(50)
)
BEGIN
UPDATE tab_tipo_envio_paqueteria SET tipo_envio_paqueteria=tipo_envio
WHERE id_tipo_envio_paqueteria=id_tipo_envioa;
END$$
Delimiter ;
```

Eliminar registro de tipo de envió de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_tipo_envio_paqueteria
(
in id_tipo_envio int
)
BEGIN
DELETE FROM tab_tipo_envio_paqueteria WHERE id_tipo_envio_paqueteria=id_tipo_envio;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_tipo_cliente_paqueteria

Insertar registro de tipo de cliente de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_tipo_cliente_paqueteria
(
in tipo_cliente_paqueteria varchar(50)
)
BEGIN
insert into tab_tipo_cliente_paqueteria(tipo_cliente_paqueteria)
                        values(tipo_cliente_paqueteria);
END$$
Delimiter ;
```

Modificar registro de tipo de cliente de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_tipo_cliente_paqueteria
(
in id_tipo_cliente int,
in tipo_cliente varchar(50)
)
BEGIN
UPDATE tab_tipo_cliente_paqueteria SET tipo_cliente_paqueteria=tipo_cliente
WHERE id_tipo_cliente_paqueteria=id_tipo_cliente;
END$$
Delimiter ;
```

Eliminar registro de tipo de cliente de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_tipo_cliente_paqueteria
(
in id_tipo_cliente int
)
BEGIN
DELETE FROM tab_tipo_cliente_paqueteria WHERE id_tipo_cliente_paqueteria=id_tipo_cliente;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_estado_paquete_paqueteria

Insertar registro de estado del paquete en la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_estado_paquete_paqueteria
(
in estado_paquete_paqueteria varchar(50)
)
BEGIN
insert into tab_estado_paquete_paqueteria(estado_paquete_paqueteria)
                        values(estado_paquete_paqueteria);
END$$
Delimiter ;
```

Modificar registro de estado del paquete en la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_estado_paquete_paqueteria
(
in id_estado_paquete int,
in estado_paquete varchar(50)
)
BEGIN
UPDATE tab_estado_paquete_paqueteria SET estado_paquete_paqueteria=estado_paquete
WHERE id_estado_paquete_paqueteria=id_estado_paquete;
END$$
Delimiter ;
```

Eliminar registro de estado del paquete en la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_estado_paquete_paqueteria
(
in id_estado_paquete int
)
BEGIN
DELETE FROM tab_estado_paquete_paqueteria WHERE id_estado_paquete_paqueteria=id_estado_paquete;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_puestos

Insertar registro de puesto en la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_puestos
(
in puesto_nombre varchar(50),
in      puesto_salario float
)
BEGIN
insert into tab_puestos(puesto_nombre, puesto_salario)
                        values(puesto_nombre, puesto_salario);
END$$
Delimiter ;
```

Actualizar registro de puesto en la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_puestos
(
in id int,
nombre varchar(50),
salario float
)
BEGIN
UPDATE tab_puestos SET puesto_nombre=nombre, puesto_salario=salario WHERE id_puesto=id;
END$$
Delimiter ;
```

Eliminar registro de puesto en la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_puestos
(
in id int
)
BEGIN
DELETE FROM tab_puestos WHERE id_puesto=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_oficinas

Insertar registro de oficinas de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_oficinas
(
in oficina_nombre varchar(50),
in      oficina_ubicacion varchar(50),
in      oficina_telefono int
)
BEGIN
insert into tab_oficinas(oficina_nombre, oficina_ubicacion, oficina_telefono)
                        values(oficina_nombre, oficina_ubicacion, oficina_telefono);
END$$
Delimiter ;
```

Modificar registro de oficinas de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_oficinas
(
in id int,
nombre varchar(50),
ubicacion varchar(50),
telefono int(11)
)
BEGIN
UPDATE tab_oficinas SET oficina_nombre=nombre, oficina_ubicacion=ubicacion, oficina_telefono=telefono
WHERE id_oficina=id;
END$$
Delimiter ;
```

Eliminar registro de oficinas de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_oficinas
(
in id int
)
BEGIN
DELETE FROM tab_oficinas WHERE id_oficina=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_direcciones

Insertar registro de direcciones

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_direcciones
(
in direccion_provincia varchar(50),
in      direccion_canton varchar(50),
in      direccion_distrito varchar(50),
in      direccion_otras_senas varchar(500)
)
BEGIN
insert into tab_direcciones(direccion_provincia, direccion_canton, direccion_distrito, direccion_otras_senas)
                        values(direccion_provincia, direccion_canton, direccion_distrito, direccion_otras_senas);
END$$
Delimiter ;
```

Modificar registro de direcciones

```
CREATE PROCEDURE sp_actualizar_direcciones
(
in id int,
provincia varchar(50),
canton varchar(50),
distrito varchar(50),
otras_senas varchar(500)
)
BEGIN
UPDATE tab_direcciones SET direccion_provincia=provincia, direccion_canton=canton,
direccion_distrito=distrito, direccion_otras_senas=otras_senas WHERE id_direccion=id;
END$$
Delimiter ;
```

Eliminar registro de direcciones

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_direcciones
(
in id int
)
BEGIN
DELETE FROM tab_direcciones WHERE id_direccion=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_empleados

Insertar registro de empleados de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_empleados
(
in empleado_identificacion int(50),
in id_empleado_puesto int,
in   id_empleado_oficina_asignada int ,
in  id_empleado_direccion int ,
in  empleado_estado varchar(50),
in  empleado_fecha_registro date ,
in  empleado_nombre_usuario_registro varchar(50),
in  empleado_apellido_1 varchar(50) ,
in empleado_apellido_2 varchar(50) ,
in   empleado_nombre varchar(50) ,
 in  empleado_fecha_nac date
)
BEGIN
insert into tab_empleados(empleado_identificacion, id_empleado_puesto, id_empleado_oficina_asignada, id_empleado_direccion, empleado_estado,
                      empleado_fecha_registro, empleado_nombre_usuario_registro, empleado_apellido_1, empleado_apellido_2, empleado_nombre, empleado_fecha_nac)

              values(empleado_identificacion,id_empleado_puesto, id_empleado_oficina_asignada, id_empleado_direccion, empleado_estado,
                     empleado_fecha_registro, empleado_nombre_usuario_registro, empleado_apellido_1, empleado_apellido_2, empleado_nombre, empleado_fecha_nac);
END$$
Delimiter ;
```

Modificar registro de empleados de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_empleados
(
in identificacion int,
puesto int(11),
oficina_asignada int(11),
direccion int(11),
estado varchar(50),
fecha_registro date,
usuario_registro varchar(50),
apellido_1 varchar(50),
apellido_2 varchar(50),
nombre varchar(50),
fecha_nac date
)
BEGIN
UPDATE tab_empleados SET id_empleado_puesto=puesto, id_empleado_oficina_asignada=oficina_asignada,
id_empleado_direccion=direccion, empleado_estado=estado, empleado_fecha_registro=fecha_registro,
empleado_nombre_usuario_registro=usuario_registro, empleado_apellido_1=apellido_1,
empleado_apellido_2=apellido_2, empleado_nombre=nombre, empleado_fecha_nac=fecha_nac
WHERE empleado_identificacion=identificacion;
END$$
Delimiter ;
```

Eliminar registro de empleados de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_empleados
(
in identificacion int
)
BEGIN
DELETE FROM tab_empleados WHERE empleado_identificacion=identificacion;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_clientes

Insertar registro de clientes de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_clientes
(
        in identificacion int(50),
        in id_tipo_paqueteria int(11),
        in id_direccion int(11),
        in apellido_1 varchar(50),
        in apellido_2 varchar(50),
        in nombre varchar(50)
)

BEGIN
insert into tab_clientes(cliente_identificacion, id_tipo_cliente_paqueteria, id_cliente_direccion,cliente_apellido_1,cliente_apellido_2,cliente_nombre)
                        values(identificacion, id_tipo_paqueteria,id_direccion,apellido_1,apellido_2,nombre);
END$$
Delimiter ;
```

Modificar registro de clientes de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_clientes
(
        in identificacion int(50),
        in id_tipo_paqueteria int(11),
        in id_direccion int(11),
        in apellido_1 varchar(50),
        in apellido_2 varchar(50),
        in nombre varchar(50)
)
BEGIN
UPDATE tab_clientes SET id_tipo_cliente_paqueteria=id_tipo_paqueteria, id_cliente_direccion=id_direccion,
cliente_apellido_1=apellido_1, cliente_apellido_2=apellido_2, cliente_nombre=nombre
Where cliente_identificacion=identificacion;
END$$
Delimiter ;
```

Eliminar registro de clientes de la paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_clientes
(
in id int
)
BEGIN
DELETE FROM tab_clientes WHERE cliente_identificacion=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_telefonos

Insertar registro de teléfonos de personas

```
/* Procedimientos almacenados para insertar, modificar y eliminar telefonos */
DELIMITER $$
CREATE PROCEDURE sp_insertar_telefonos_empleado
(
in      identificacion_empleados_telefono int,
in      telefono int
)

BEGIN
insert into tab_telefonos(identificacion_empleados_telefono, telefono)
                        values(identificacion_empleados_telefono, telefono);
END$$
Delimiter ;

DELIMITER $$
CREATE PROCEDURE sp_insertar_telefonos_cliente
(
in      identificacion_clientes_telefono int,
in      telefono int
)
```

Modificar registro de teléfonos de personas

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_telefonos
(
in id int,
in telefono int
)
BEGIN
UPDATE tab_telefonos SET telefono=telefono Where id_telefono=id;
END$$
Delimiter ;
```

Eliminar registro de teléfonos de personas

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_telefonos
(
in id int
)
BEGIN
DELETE FROM tab_telefonos WHERE id_telefono=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_correos

Insertar registro de correos de personas

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_correos_empleado
(
in      identificacion_empleados_telefono int,
in      correo varchar(50)
)

BEGIN
insert into tab_correos(identificacion_empleados_telefono, correo)
                        values(identificacion_empleados_telefono, correo);
END$$
Delimiter ;

DELIMITER $$
CREATE PROCEDURE sp_insertar_correos_cliente
(
in      identificacion_clientes_telefono int,
in      correo varchar(50)
)

BEGIN
insert into tab_correos(identificacion_clientes_telefono, correo)
                        values(identificacion_clientes_telefono, correo);
END$$
Delimiter ;
```

Modificar registro de correos de personas

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_correos
(
in id int,
in correo varchar(50)
)
BEGIN
UPDATE tab_correos SET correo=correo Where id_correo=id;
END$$
Delimiter ;
```

Eliminar registro de correos de personas

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_correos
(
in id int
)
BEGIN
DELETE FROM tab_correos WHERE id_correo=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_inventario_vehiculos

Insertar registro de vehículos

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_vehiculos
(
in      id_tipo_vehiculo int,
in      vehiculo_num_interno int,
in      vehiculo_num_placa varchar(50),
in      vehiculo_marca varchar(50),
in      vehiculo_modelo varchar(50),
in      vehiculo_stock varchar(50),
in      vehiculo_estado varchar(50),
in      vehiculo_fecha_registro date
)

BEGIN
insert into tab_inventario_vehiculos(id_tipo_vehiculo,vehiculo_num_interno, vehiculo_num_placa, vehiculo_marca, vehiculo_modelo, vehiculo_stock,
vehiculo_estado, vehiculo_fecha_registro)
                        values(id_tipo_vehiculo, vehiculo_num_interno, vehiculo_num_placa, vehiculo_marca, vehiculo_modelo, vehiculo_stock,
vehiculo_estado, vehiculo_fecha_registro);
END$$
Delimiter ;
```

Modificar registro de vehículos

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_vehiculos
(
in id int,
in id_tipo_vehiculo int,
in      vehiculo_num_interno int,
in      vehiculo_num_placa varchar(50),
in      vehiculo_marca varchar(50),
in      vehiculo_modelo varchar(50),
in      vehiculo_stock varchar(50),
in      vehiculo_estado varchar(50),
in      vehiculo_fecha_registro date
)
BEGIN
UPDATE tab_inventario_vehiculos SET id_tipo_vehiculo=id_tipo_vehiculo, vehiculo_num_interno=vehiculo_num_interno,  vehiculo_num_placa=vehiculo_num_placa,
vehiculo_marca=vehiculo_marca, vehiculo_modelo=vehiculo_modelo, vehiculo_modelo=vehiculo_modelo, vehiculo_stock=vehiculo_stock, vehiculo_estado=vehiculo_estado,
vehiculo_fecha_registro=vehiculo_fecha_registro
Where id_inventario_vehiculo=id;
END$$
Delimiter ;
```

Eliminar registro de vehículos

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_vehiculos
(
in id int
)
BEGIN
DELETE FROM tab_inventario_vehiculos WHERE id_inventario_vehiculo=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_facturacion_paqueteria

Insertar registro de facturación de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_facturas_paqueteria
(
in      id_cliente_factura int,
in      factura_numero int,
in      factura_fecha datetime,
in      factura_costo_impuestos float,
in      factura_costo_flete float,
in      factura_costo_iva float,
in      factura_total_por_envio float,
in      factura_total_antes_iva float,
in      factura_total float,
in      factura_estado varchar(50),
in      factura_usuario_registro varchar(50)
)

BEGIN
insert into tab_facturacion_paqueteria(id_cliente_factura, factura_numero, factura_fecha, factura_costo_impuestos, factura_costo_flete, factura_costo_iva, factura_total_por_er
factura_total_antes_iva, factura_total, factura_estado, factura_usuario_registro)
                    values(id_cliente_factura, factura_numero, factura_fecha, factura_costo_impuestos, factura_costo_flete, factura_costo_iva, factura_total_por_envio,
factura_total_antes_iva, factura_total, factura_estado, factura_usuario_registro);
END$$
Delimiter ;
```

Modificar registro de facturación de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_facturas_paqueteria
(
in id int,
in      factura_numero int,
in      factura_fecha datetime,
in      factura_costo_impuestos float,
in      factura_costo_flete float,
in      factura_costo_iva float,
in      factura_total_por_envio float,
in      factura_total_antes_iva float,
in      factura_total float,
in      factura_estado varchar(50),
in      factura_usuario_registro varchar(50)
)
BEGIN
UPDATE tab_facturacion_paqueteria SET factura_numero=factura_numero, factura_fecha=factura_fecha, factura_costo_impuestos=factura_costo_impuestos,
factura_costo_flete=factura_costo_flete, factura_costo_iva=factura_costo_iva, factura_total_por_envio=factura_total_por_envio, factura_total_antes_iva=factura_total_antes_iva,
factura_total=factura_total, factura_estado=factura_estado, factura_usuario_registro =factura_usuario_registro
Where id_factura=id;
END$$
Delimiter ;
```

Eliminar registro de facturación de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_facturas_paqueteria
(
in id int
)
BEGIN
DELETE FROM tab_facturacion_paqueteria WHERE id_factura=id;
END$$
Delimiter ;
```

Procedimientos almacenados de la tabla tab_paqueteria

Insertar registro de paquete de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_insertar_paquetes
(
in      id_cliente int(50),
in      id_oficina int,
in      id_empleado_proceso int,
in      id_tipo_envio_paqueteria int,
in      id_estado_paquete_paqueteria int,
in  id_factura_paquete int,
in      paqueteria_numero int,
in      paqueteria_usuario_registro varchar(50),
in      paqueteria_descripcion varchar(500),
in      paqueteria_fecha_registro date
)

BEGIN
insert into tab_paqueteria(id_cliente, id_oficina, id_empleado_proceso, id_tipo_envio_paqueteria, id_estado_paquete_paqueteria, id_factura_paquete, paqueteria_numero,
paqueteria_usuario_registro, paqueteria_descripcion, paqueteria_fecha_registro)
                        values(id_cliente, id_oficina, id_empleado_proceso, id_tipo_envio_paqueteria, id_estado_paquete_paqueteria, id_factura_paquete, paqueteria_numero,
paqueteria_usuario_registro, paqueteria_descripcion, paqueteria_fecha_registro);
END$$
Delimiter ;
```

Modificar registro de paquete de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_actualizar_paquetes
(
in id int,
in      id_oficina int,
in      id_empleado_proceso int,
in      id_tipo_envio_paqueteria int,
in      id_estado_paquete_paqueteria int,
in  id_factura_paquete int,
in      paqueteria_numero int,
in      paqueteria_usuario_registro varchar(50),
in      paqueteria_descripcion varchar(500),
in      paqueteria_fecha_registro date
)
BEGIN
UPDATE tab_paqueteria SET id_oficina=id_oficina, id_empleado_proceso=id_empleado_proceso, id_tipo_envio_paqueteria=id_tipo_envio_paqueteria,
id_estado_paquete_paqueteria=id_estado_paquete_paqueteria, id_factura_paquete=id_factura_paquete, paqueteria_numero=paqueteria_numero, paqueteria_usuario_registro=paqueteria_u
paqueteria_descripcion=paqueteria_descripcion, paqueteria_fecha_registro=paqueteria_fecha_registro
Where id_paqueteria=id;
END$$
Delimiter ;
```

Eliminar registro de paquete de paquetería

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_paquetes
(
in id int
)
BEGIN
DELETE FROM tab_paqueteria WHERE id_paqueteria=id;
END$$
Delimiter ;
```

Triggers

A continuación, se detalla la elaboración de los triggers o disparadores de auditoría para las tablas de Paqueteria, Facturacion, Empleados y Clientes. Para la creación de estos triggers de auditoria fue necesaria la creación de tablas adicionales a manera de log con referencia a las anteriormente mencionadas, esto con el fin de replicar la información modificada o eliminada de forma que se puedan consultar los cambios realizados, así como el usuario y la hora en que se realizaron dichos cambios.

Creación del trigger o disparador de auditoría para tab_paqueteria

```
CREATE TABLE IF NOT EXISTS log_tab_paqueteria(
    id_paqueteria int,
    id_cliente int(50),
    id_oficina int,
    id_empleado_proceso int,
    id_tipo_envio_paqueteria int,
    id_estado_paquete_paqueteria int,
    paqueteria_usuario_registro varchar(50),
    paqueteria_descripcion varchar(500),
    paqueteria_fecha_registro date,
    accion_realizada varchar(50),
    usuario_modifico varchar (50),
    fecha_modifico date
);
```

```
DELIMITER //

CREATE TRIGGER trg_borrar_paqueteria BEFORE DELETE ON tab_paqueteria FOR EACH ROW
    begin
            insert into log_tab_paqueteria (id_paqueteria, id_cliente, id_oficina, id_empleado_proceso, id_tipo_envio_paqueteria,
            id_estado_paquete_paqueteria, accion_realizada, paqueteria_usuario_registro, paqueteria_descripcion,
            paqueteria_fecha_registro, usuario_modifico, fecha_modifico)
                values (old.id_paqueteria, old.id_cliente, old.id_oficina, old.id_empleado_proceso, old.id_tipo_envio_paqueteria,
                old.id_estado_paquete_paqueteria, old.paqueteria_usuario_registro, old.paqueteria_descripcion,
            old.paqueteria_fecha_registro, 'Se actualizó el registro de paquetería', current_user(), now());
    end //

DELIMITER //

CREATE TRIGGER trg_actualizar_paqueteria AFTER UPDATE ON tab_paqueteria FOR EACH ROW
    begin
            insert into log_tab_paqueteria (id_paqueteria, id_cliente, id_oficina, id_empleado_proceso, id_tipo_envio_paqueteria,
            id_estado_paquete_paqueteria, accion_realizada, paqueteria_usuario_registro, paqueteria_descripcion,
            paqueteria_fecha_registro, usuario_modifico, fecha_modifico)
                values (new.id_paqueteria, new.id_cliente, new.id_oficina, new.id_empleado_proceso, new.id_tipo_envio_paqueteria,
                new.id_estado_paquete_paqueteria, 'Se eliminó el registro de paquetería', new.paqueteria_usuario_registro, new.paqueteria_descripcion,
            new.paqueteria_fecha_registro, current_user(), now());
  end //

Delimiter ;
```

Creación del trigger o disparador de auditoría para tab_facturacion_paqueteria

```
CREATE TABLE IF NOT EXISTS log_tab_facturacion_paqueteria(
id_factura int ,
id_cliente_factura int,
factura_numero int,
factura_fecha datetime,
factura_costo_impuestos float,
factura_costo_flete float,
factura_costo_iva float,
factura_total_por_envio float,
factura_total_antes_iva float,
factura_total float,
factura_estado varchar(50),
factura_usuario_registro varchar(50),
accion_realizada varchar(50),
usuario_modifico varchar (50),
fecha_modifico date
);
```

```
DELIMITER //

CREATE TRIGGER trg_actualizar_facturacion AFTER UPDATE ON tab_facturacion_paqueteria FOR EACH ROW
    begin
        insert into log_tab_facturacion_paqueteria (id_factura, id_cliente_factura, factura_numero, factura_fecha, factura_costo_impuestos,
                                                    factura_costo_flete, factura_costo_iva, factura_total_por_envio, factura_total_antes_iva, factura_total, factura_estado,
                                                    factura_usuario_registro, accion_realizada, usuario_modifico, fecha_modifico)
            values (new.id_factura, new.id_cliente_factura, new.factura_numero, new.factura_fecha, new.factura_costo_impuestos,
                    new.factura_costo_flete, new.factura_costo_iva, new.factura_total_por_envio, new.factura_total_antes_iva, new.factura_total, new.factura_estado,
                    new.factura_usuario_registro,'Se actualizó un registro de facturación', current_user(), now());
    end //

    DELIMITER //

CREATE TRIGGER trg_borrar_facturacion BEFORE DELETE ON tab_facturacion_paqueteria FOR EACH ROW
    begin
        insert into log_tab_facturacion_paqueteria (id_factura, id_cliente_factura, factura_numero, factura_fecha, factura_costo_impuestos,
                                                    factura_costo_flete, factura_costo_iva, factura_total_por_envio, factura_total_antes_iva, factura_total, factura_estado,
                                                    factura_usuario_registro, accion_realizada, usuario_modifico, fecha_modifico)
            values (old.id_factura, old.id_cliente_factura, old.factura_numero, old.factura_fecha, old.factura_costo_impuestos,
                    old.factura_costo_flete, old.factura_costo_iva, old.factura_total_por_envio, old.factura_total_antes_iva, old.factura_total, old.factura_estado,
                    old.factura_usuario_registro,'Se eliminó un registro de facturación', current_user(), now());
    end //

    Delimiter ;
```

Creación del trigger o disparador de auditoría para tab_empleados

```
CREATE TABLE IF NOT EXISTS log_tab_empleados(
    empleado_identificacion int(50),
    id_empleado_puesto int,
    id_empleado_oficina_asignada int,
    id_empleado_direccion int,
    empleado_estado varchar(50),
    empleado_fecha_registro date,
    empleado_nombre_usuario_registro varchar(50),
    empleado_apellido_1 varchar(50),
    empleado_apellido_2 varchar(50),
    empleado_nombre varchar(50),
    empleado_fecha_nac date,
    accion_realizada varchar(50),
    usuario_modifico varchar (50),
    fecha_modifico date
);
```

```
DELIMITER //

CREATE TRIGGER trg_actualizar_empleados AFTER UPDATE ON tab_empleados FOR EACH ROW
    begin
        insert into log_tab_empleados (empleado_identificacion, id_empleado_puesto, id_empleado_oficina_asignada, id_empleado_direccion,
                                       empleado_estado, empleado_fecha_registro, empleado_nombre_usuario_registro, empleado_apellido_1, empleado_apellido_2,
                                       empleado_nombre, empleado_fecha_nac, accion_realizada, usuario_modifico, fecha_modifico)
            values (new.empleado_identificacion, new.id_empleado_puesto, new.id_empleado_oficina_asignada, new.id_empleado_direccion,
                    new.empleado_estado, new.empleado_fecha_registro, new.empleado_nombre_usuario_registro, new.empleado_apellido_1, new.empleado_apellido_2,
                    new.empleado_nombre, new.empleado_fecha_nac, 'Se actualizó un registro de empleado', current_user(), now());
    end //

DELIMITER //

CREATE TRIGGER trg_borrar_empleados BEFORE DELETE ON tab_empleados FOR EACH ROW
    begin
        insert into log_tab_empleados (empleado_identificacion, id_empleado_puesto, id_empleado_oficina_asignada, id_empleado_direccion,
                                       empleado_estado, empleado_fecha_registro, empleado_nombre_usuario_registro, empleado_apellido_1, empleado_apellido_2,
                                       empleado_nombre, empleado_fecha_nac, accion_realizada, usuario_modifico, fecha_modifico)
            values (old.empleado_identificacion, old.id_empleado_puesto, old.id_empleado_oficina_asignada, old.id_empleado_direccion,
                    old.empleado_estado, old.empleado_fecha_registro, old.empleado_nombre_usuario_registro, old.empleado_apellido_1, old.empleado_apellido_2,
                    old.empleado_nombre, old.empleado_fecha_nac, 'Se eliminó un registro de empleado', current_user(), now());
    end //

Delimiter ;
```

Creación del trigger o disparador de auditoría para tab_clientes

```
CREATE TABLE IF NOT EXISTS log_tab_clientes(
    cliente_identificacion int(50),
    id_tipo_cliente_paqueteria int,
    cliente_apellido_1 varchar(50),
    cliente_apellido_2 varchar(50),
    cliente_nombre varchar(50),
    accion_realizada varchar(50),
  usuario_modifico varchar (50),
    fecha_modifico date
);
```

```
    DELIMITER //

CREATE TRIGGER trg_actualizar_clientes AFTER UPDATE ON tab_clientes FOR EACH ROW
    begin
        insert into log_tab_clientes (cliente_identificacion, id_tipo_cliente_paqueteria, cliente_apellido_1, cliente_apellido_2, cliente_nombre,
                                      accion_realizada, usuario_modifico, fecha_modifico)
            values (new.cliente_identificacion, new.id_tipo_cliente_paqueteria, new.cliente_apellido_1, new.cliente_apellido_2, new.cliente_nombre,
                    'Se actualizó un registro de clientes', current_user(), now());
    end //

    DELIMITER //

CREATE TRIGGER trg_borrar_clientes BEFORE DELETE ON tab_clientes FOR EACH ROW
    begin
        insert into log_tab_clientes (cliente_identificacion, id_tipo_cliente_paqueteria, cliente_apellido_1, cliente_apellido_2, cliente_nombre,
                                      accion_realizada, usuario_modifico, fecha_modifico)
            values (old.cliente_identificacion, old.id_tipo_cliente_paqueteria, old.cliente_apellido_1, old.cliente_apellido_2, old.cliente_nombre,
                    'Se eliminó un registro de clientes', current_user(), now());
    end //

    Delimiter ;
```

CONCLUSIONES

Al realizar este proyecto hemos podido conocer la importancia de establecer los parámetros, datos y relaciones de una tabla para contar con una funcionalidad optima de una base de datos, ya que si no se cuenta con una buena construcción de la misma puede llegar a generar muchos errores a futuro. Además, se lo reconocer que es de suma importancia manejar los INSERT, DELETE Y UPDATE de datos mediante procesos almacenados ya que esto nos permite aumentar la seguridad de nuestra base de datos y también manejar un registro de estos mismos para tener un control en caso de algún problema.

A partir de este proyecto y su elaboración hemos podido conocer las ventajas de las bases de datos que se superponen a lo que antes teníamos disponible como sistemas para el manejo de archivos como lo es el modelo relacional de SQL siendo un excelente aliado en la administración, seguridad y fiabilidad de los datos. Personalmente llegué a comprender que el lenguaje no es innecesariamente complejo o detallado, pero si requiere una planeación anticipada de cómo será todo el modelo de trabajo y las características que esté ofrecerá una vez esté implementado, las mismas deben de facilitar el orden de los datos, la efectividad y el control administrativo. Otro punto que en lo personal destaco por su eficacia es la utilización de procedimientos almacenados los cuales cuentan con la ventaja de mejorar de manera exponencial el rendimiento al ser estos capaces de poder ejecutar código compilado directamente desde la base de datos aprovechando caché y demás recursos del servidor propiamente.

El Proyecto realizado ha aportado de manera muy importante para determinar y considerar los aspectos que hay que abarcar y tomar en cuenta para llevar a cabo una implementación exitosa de una base de datos, donde se debe conocer y ejecutar desde temas teóricos como la normalización de tablas, hasta la ejecución practica para llevar a cabo el código fuente, por lo cual pienso que el aporte es muy enriquecedor en cuanto a conocimiento adquirido.