Documentación Base de datos

Definiciones:

```
create database if not exists db_restaurante;
use db_restaurante;
```

Definición de la base de datos, crea la base de datos si no existe e inmediatamente se hace uso de la misma.

Table Usuarios:

```
create table usuarios (
id_usuario int auto_increment primary key,
nombre varchar(100) not null,
correo varchar(100) unique,
usuario varchar(50) not null unique,
contraseña varchar(255) not null unique,
rol Enum('administrador', 'cajero', 'mesero') not null
);
```

Se define o crea la tabla correspondiente para la información de los usuarios del sistema principal:

Un identificador el cual fungirá como clave primaria para el objeto.

Nombre el cual no puede estar vacío

Correo único

Usuario para autenticación no puede estar vació y es único.

Contraseña un en un carácter de máximo 255, no vacía y única.

Rol, según el rol tiene permisos diferentes.

Tabla Mesas:

```
create table mesas(
id_mesa int auto_increment primary key,
numero int not null,
tipo varchar(100) not null,   -- regular o grande(para poder unir solo dos mesas y sea mas sencillo
estado  Enum('libre', 'ocupada', 'reservada') default 'libre'
);
```

Gestiona las mesas del restaurante

La razón por la que una mesa tenga un identificador diferente al numero es por si se cambian los números de las mesas, de esta manera se conoce cual es la mesa original.

Tipo, se puede regular o grade por si es un grupo grande se unen dos mesas grandes.

Estado se refiere al estado actual de la mesa, esto con el fin de proporcionar información a las estadísticas y posteriormente para las reservas.

Tabla para las mesas unidas:

```
create table mesas_unidas (
id int auto_increment primary key,
mesa_principal int not null,
mesa_secundaria int not null,
estado enum('activa','cerrada') default 'activa',
foreign key (mesa_principal) references mesas(id_mesa),
foreign key (mesa_secundaria) references mesas(id_mesa)
);
```

Esta es una tabla adicional para el manejo de las mesas cuando están unidas consiste en una mesa principal unida a otra mesa secundaria.

Estado para manipular si las mesas están unidas o no, y por defecto está en activa.

Como clave foránea utiliza el id.

Tabla productos:

```
create table productos(
id_producto int auto_increment primary key,
nombre varchar(100) not null,
precio decimal(10,2) not null,
categoria enum('comida', 'bebida', 'postre', 'otro') not null,
disponible boolean default true
);
```

Esta tabla se crea con el fin de manejar el catálogo de los productos (menú).

Tiene un identificador que funge como clave primaria del producto.

Precio es el precio unitario del producto.

Categoría, es para clasificar los productos.

Disponible, indica si el producto esta en stock o no.

Tabla descuentos:

```sql
create table descuentos (
    id int auto_increment primary key,
    tipo enum('hora_feliz','cupon','convenio') not null,
    valor DECIMAL(10,2) NOT NULL,
    condiciones TEXT,
    fecha_inicio DATETIME,
    fecha_fin DATETIME
);
```

Aquí se registran las promociones y descuentos:

El tipo se refiere al tipo de descuento que se trata (ej: hora feliz).

Valor se refiere al monto o porcentaje que se va a descontar.

Tabla clientes:

```sql
create table clientes (
id_cliente int auto_increment primary key,
documento varchar(50) unique, -- cedula o din por si un cliente web o presencial se registra en la pagina web o en el restaurante encuentre la coincidencia y los relacione (hay que crear un campo en la pagina para registro)
nombre varchar(100) not null,
correo varchar(100) ,
unique (correo),
id_descuento int,
foreign key(id_descuento) references descuentos(id)

);
```

Para almacenar los datos de los clientes

Foreign key con descuentos: cada cliente puede estar asociado a un descuento activo.

Tabla pedidos:

```sql
create table pedidos(
id_pedido int auto_increment primary key,
id_mesa int not null,
fecha timestamp default current_timestamp,
id_usuario int not null, -- el del mesero
observaciones text,
estado enum('recibido-pendiente', 'servido', 'pagado') default 'recibido-pendiente',
foreign key (id_usuario) references usuarios(id_usuario),
foreign key (id_mesa) references mesas(id_mesa)
);
```

Representa los pedidos realizados por clientes en una mesa.

FK con usuarios → mesero que atendió.

FK con mesas → mesa correspondiente.

Tabla detalle_pedido:

```sql
create table detalle_pedido(
id_detalle int auto_increment primary key,
id_pedido int not null,
id_producto  int not null,
cantidad int not null default 1,
precio_unitario decimal(10,2) not null,
subtotal decimal(10,2) generated always as (cantidad * precio_unitario) stored,
foreign key (id_pedido)references pedidos(id_pedido),
foreign key (id_producto) references productos(id_producto)
);
```

Esta tabla es para los detalles del pedido el incluye el subtotal el cual se calcula automáticamente.

FK con pedidos.

FK con productos.

Tabla reservas:

```
create table reservas(
id_reserva int auto_increment primary key,
nombre_cliente varchar(100) not null,
documento varchar(50) ,                          -- Cualquiera puede hacer una reservacion, si es un cliente en resgistrado
telefono varchar(15),                                  -- bien sea en la web o presencial tiene prioriadad o descuentos? hay que hacer que coincida
fecha_reserva datetime not null,                       -- por si detecta que un cliente registrado quiere hacer una reservacion
id_mesa int not null,
estado enum('activa', 'cumplida', 'cancelada') default 'activa',
foreign key (id_mesa) references mesas(id_mesa)
);
```

Registro de reservas de clientes.

FK con mesas.

Tabla parqueadero:

```
create table parqueadero (
id int auto_increment primary key,
capacidad_total int not null,
capacidad_ocupada int not null default 0

);
```

Esta es para almacenar el estado actual o la capacidad actual del parqueadero para las estadísticas.

Tabla pagos:

```
create table pagos(
id int auto_increment primary key,
id_pedido int not null,
monto decimal(10,2) not null,  -- se calcula en la app
metodo_pago enum('efectivo', 'tajeta', 'trasferencia') not null,
fecha datetime default current_timestamp,
cliente varchar(100),
foreign key (id_pedido) references pedidos(id_pedido)

);
```

Esta es para registrar los pagos de los pedidos y los diferentes medios de pago y tiene FK con pedidos, la idea es que los cálculos se realicen en la parte de Python teniendo en cuenta los descuentos.

Tabla ventas:

```
create table ventas (
id int auto_increment primary key,
fecha date not null,
total_dia decimal(12,2) not null,
generado_por int,
foreign key (generado_por) references usuarios(id_usuario)
);
```

El objetivo es que almacene un consolidado de ventas diarias realizadas con el fin de proporcionar esta información a los administradores.

FK con usuarios, ósea el empleado que genero el reporte.

Relaciones:

usuarios → atienden pedidos, generan ventas.

mesas → pueden estar unidas (mesas_unidas), asociadas a pedidos y reservas.

productos → se incluyen en los pedidos a través de detalle_pedido.

clientes → pueden tener descuentos y hacer reservas.

pedidos → generan detalle_pedido y están asociados a pagos.

ventas → resumen de ingresos diarios.