



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

---

---

FACULTAD DE INGENIERÍA

BASES DE DATOS

## **PROYECTO FINAL**

BASES DE DATOS

CORONA NAVA PEDRO JAIR

FERNÁNDEZ ROSALES SEBASTIAN

MARTÍNEZ GARCÍA GABRIELA

REYES MENDOZA MIRIAM GUADALUPE

ZARATE DÍAZ SOFÍA VIRIDIANA

**Asignatura:** Bases de datos

**Grupo:** 01

**Equipo:** Bases primaverales

**Semestre:** 2022-2



## INTRODUCCIÓN

Una base de datos es una colección de datos que se organizan de modo que se pueda acceder fácilmente a esta información, además de permitir un almacenamiento más preciso, confiable y de fácil acceso, ya que es una forma estructurada de almacenamiento. De hecho, en la actualidad, las bases de datos son esenciales, ya que se implementan desde un software bancario hasta una investigación científica o registros gubernamentales, pero sobre todo en los sitios web o aplicaciones móviles usadas de forma cotidiana.

Pero, las bases de datos no necesariamente deben tener un uso masivo, también pueden ser utilizadas por pequeñas empresas. La principal razón para esto se debe a que las bases de datos facilitan mucho, mucho el acceso a la información mediante el uso de un ordenador. Si trabajamos con información o datos todos los días, vale la pena entender qué son las bases de datos y como es que estas funcionan, razón por la cual a lo largo de este proyecto, se desarrolla una base de datos funcional y sencilla paso a paso para un buen desarrollo e implementación.

Para lo anterior, se debe comprender que una computadora es un dispositivo que permite manipular información, ya sea que la infor-

mación tome la forma de palabras, números, imágenes o videos. Sin embargo, una computadora necesita almacenar la información antes de que esa información pueda ser referenciada o cambiada, y también necesita asegurarse de que pueda encontrar la información correcta en el momento que se solicite. Las bases de datos son la forma en se resuelven estos dos problemas.

El término modelo de datos describe la estructura lógica de una base de datos, que determina las reglas sobre cómo se puede organizar y manipular la información que contiene. Razón por la cual se opta por usar bases de datos, ya que cuando los datos están estructurados, eso significa que están formateados y se pueden buscar. Mientras que, los datos no estructurados carecen tanto de formato como de accesibilidad y, por lo tanto, son mucho más difíciles de analizar.

Sea cual sea el tipo de datos con los que se está trabajando, estos deben organizarse de acuerdo con un modelo, que describe los detalles sobre cómo se desea implementar la base, como los tipos de datos requeridos u otras restricciones. Por último, necesitamos alguna forma de interactuar con la base de datos para realizar las acciones deseadas, para esto se hace uso de un sistema de administración de bases de datos que es el software que hace posible que los usuarios finales creen, modifiquen y administren bases de datos, así como también definan, almacenen, manipulen y recuperen los datos.

## 1.1. OBJETIVO

Analizar una serie de requerimientos y propondrá una solución que atienda a los mismos, aplicando los conceptos vistos en el curso.

## 1.2. PLANTEAMIENTO

Un restaurante desea digitalizar su forma de operación, para ello se desarrollará un sistema informático que constar a de varios módulos. El que corresponde a la implementación de la base de datos deberá atender el siguiente requerimiento:

Se debe almacenar el RFC, número de empleado, nombre, fecha de nacimiento, teléfonos, edad, domicilio, sueldo; de los cocineros su especialidad, de los meseros su horario y de los administrativos su rol, así como una foto de los empleados y considerar que un empleado puede tener varios puestos. Es necesario tener registro de los dependientes de los empleados, su curp, nombre y parentesco. Se debe tener disponible la información de los platillos y bebidas que el restaurante ofrece, una descripción, nombre, su receta, precio y un indicador de disponibilidad, así como el nombre y descripción de la categoría a la que pertenecen (considerar que un platillo o bebida sólo pertenece a una categoría).

Debe tenerse registro del folio de la orden, fecha, la cantidad total a pagar por la orden y registro del mesero que levantó la orden, así como la cantidad de cada platillo/bebida y precio total a pagar por platillo/bebida contenidos en cada orden. Considerar que es posible que los clientes soliciten factura de su consumo, por lo que debe almacenarse su RFC, nombre, domicilio, razón social, email y fecha de nacimiento.

Adicional al almacenamiento de información, la base de datos debe atender los siguientes puntos:

- Cada que se agregue un producto a la orden, debe actualizarse los totales (por producto y venta), así como validar que el producto esté disponible.
- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.
- Dado un número de empleado, mostrar la cantidad de órdenes que ha registrado en el día así como el total que se ha pagado por dichas órdenes. Si no se trata de un mesero, mostrar un mensaje de error.
- Vista que muestre todos los detalles del platillo más vendido.
- Permitir obtener el nombre de aquellos productos que no estén disponibles.

- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una orden.
- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una orden.

## **CONSIDERACIONES**

- El folio de la orden debe tener un formato similar a ORD-001, prefijo ORD, seguido de un guión y un número secuencial.
- El atributo domicilio, está compuesto por estado, código postal, colonia, calle y número.
- El atributo nombre, está compuesto por nombre, apellido paterno y materno.

## **PLAN DE TRABAJO**

El plan de trabajo fue dividido en 6 secciones para un buen desarrollo, resaltando que para las primeras 2 etapas se trabajó de forma colaborativa, mientras que para las siguientes etapas se hizo mediante parejas o de forma individual. Es importante hacer notar que, a pesar de trabajar algunas etapas de esta forma, en caso de surgir dudas durante el desarrollo se proporcionó ayuda de formal grupal.

## **2.1. PARTES A REALIZAR**

### **ANÁLISIS**

Esta etapa se llevó a cabo de forma colaborativa, para entender cuáles iban a ser los puntos de partida para el desarrollo del proyecto, aquí se analizó de una forma muy concreta cómo íbamos a trabajar y cuál iba a ser nuestra organización. Esto fue de suma importancia, ya que nos permitió realizar el proyecto de una forma ordenada, Para que toda persona trabajará en el punto donde tuviera mayor habilidad y de esta forma obtener un mejor resultado.

Durante esta parte nosotros nos dividimos los puntos, mencionados posteriormente, resaltando que si surgía alguna duda íbamos a trabajar en equipo para poder sacarla lo más adecuadamente posible, además organizamos una serie de reuniones que nos permitieran ver el avance de cada uno y también de forma general.

### **MODELADO Y DISEÑO**

Se empezó analizando la información o los datos que debemos almacenar, de esta forma nosotros pudimos empezar a hacer el modelo entidad – relación, mediante trabajo en equipo para obtener un resultado completo y que satisfacer a posteriormente las necesidades o los requerimientos pedidos para el desarrollo.

Esta parte es esencial puesto que de aquí se parte para obtener una buena base de datos, que sea óptima para lo que se está buscando, además posteriormente pudimos crear el modelo relacional, para observar si podíamos hacer mejoras sobre lo planteado con anterioridad, este fue el punto de partida para poder realizar los puntos a seguir para la manipulación de nuestros datos, de acuerdo con lo que hicimos en esta etapa.

## **REQUERIMIENTOS**

Esta parte nos permitió ver cuáles eran las restricciones, consultas y operaciones necesarias para la manipulación de los datos dentro de la base, esta parte al igual fue desarrollada en colaboración con todos los integrantes de equipo al igual que las etapas anteriores, sin embargo, fueron implementadas, corregidas y aplicadas dentro de la base por Pedro Corona y Sebastián Fernández.

Aquí se realizó de forma ordenada cada punto dentro de los requerimientos solicitados que debía cumplir nuestra base de forma lógica en la base de datos, es decir, la administración y sobre todo manipulación y consultas que estos debían tener. Estos llevaron un tiempo de elaboración bastante amplio ya que se necesitó un análisis profundo de los datos, las tablas y los modelos que fueron realizados con anterioridad.

## CREACIÓN DE TABLAS E INSERCIÓN DE DATOS

Esta fue un parte bastante sencilla de realizar, por lo que, para hacer la creación de tablas de una forma más rápida y eficiente para su posterior manipulación, 3 integrantes del equipo realizaron esta parte, estos fueron Gabriela Martínez, Sofía Zarate y Sebastián Fernández. Se optó por hacer de esta forma, ya que, a pesar de no ser algo tan complicado el número de tablas era bastante extenso.

Una vez con las tablas creadas nosotros pudimos pasar a las siguientes etapas, pero dentro de esto mismo es importante recargar otra de las cosas que se hizo durante esta etapa.

Una de las cosas que sí hizo posteriormente a la creación de tablas, fue insertar los datos que tendrían nuestras tablas para su uso estos datos, no sirvieron principalmente para aplicar y ver si los requerimientos están funcionando de una forma adecuada.

Este fue un listado bastante largo ya que se necesita relacionar las tablas de acuerdo con los atributos que tienen en común, pero una vez teniendo en cuenta cómo se están ingresado y con qué atributo en común trabajan, esto se vuelve más óptimo, por lo que esta actividad fue realizada por Miriam Reyes.

## **INTERFAZ GRÁFICA**

Esta es una sección, bastante importante en el desarrollo ya que nos permite visualizar los datos de una forma más amigable para el usuario, por lo que a pesar de no ser difícil necesita tiempo para poder hacer un diseño que sea óptimo para los requerimientos realizados, en este caso se optó porque después de hacer todas las etapas anteriores, se hicieron una interfaz que se adaptará al diseño que nosotros hicimos.

Esta interfaz fue realizada por Gabriela Martínez y Sofía Zarate mediante el análisis de los datos que ya teníamos para generar una interfaz óptima que cubriera todos los requerimientos pedidos y donde se pudiera observar de forma clara lo realizado durante el desarrollo de la base de datos.

## **DOCUMENTACIÓN**

La documentación es un proceso bastante extenso ya que se tiene que justificar parte por parte lo que se realizó durante el desarrollo, sin embargo, sí se entiende de forma clara qué es lo que se realizó, se vuelve más sencilla. La documentación fue realizada por Miriam Reyes y nos permitió, recopilar de forma clara cómo fue la elaboración de cada etapa del proyecto, además de una justificación de ciertos puntos.

Es importante ver detalladamente cada paso que se siguió para la creación de la base de datos es por eso por lo que, de forma resumida se presentan en este documento, además se presenta lo esencial como son el análisis, el modelado, la creación de tablas, la inserción de datos y la interfaz gráfica. Describiendo cada punto con la mayor claridad posible para que sea entendible qué fue lo que se pretendió realizar y cómo fue este proceso.

## 2.2. HERRAMIENTAS DE TRABAJO

- **GitHub:** Esta fue la plataforma utilizada para subir nuestro trabajo, este es un recurso de programación que se utiliza para compartir código y alojar proyectos.
  
- **Postgres-SQL:** Esta fue la plataforma más escencial para el desarrollo de nuestro proyecto, ya que al ser un gestor de base de datos es un software diseñado para almacenar, recuperar, definir y administrar los datos de nuestra base de datos y donde pudimos realizar consultas SQL.
  
- **Discord:** Esta es una aplicación de chat de texto, voz y video grupal, que nos permitió organizarnos de forma más eficiente y

óptima para la realización del proyecto, al igual que para ayudarnos con las dudas que surgieron a lo largo de todo el proceso.

- **Drive:** Este servicio de almacenamiento nos permitió trabajar las diferentes versiones y modificaciones en tiempo real que realizamos, en este se puede almacenar y acceder a archivos en línea ya que el servicio sincroniza los documentos cargados.

### 2.3. AVANCE

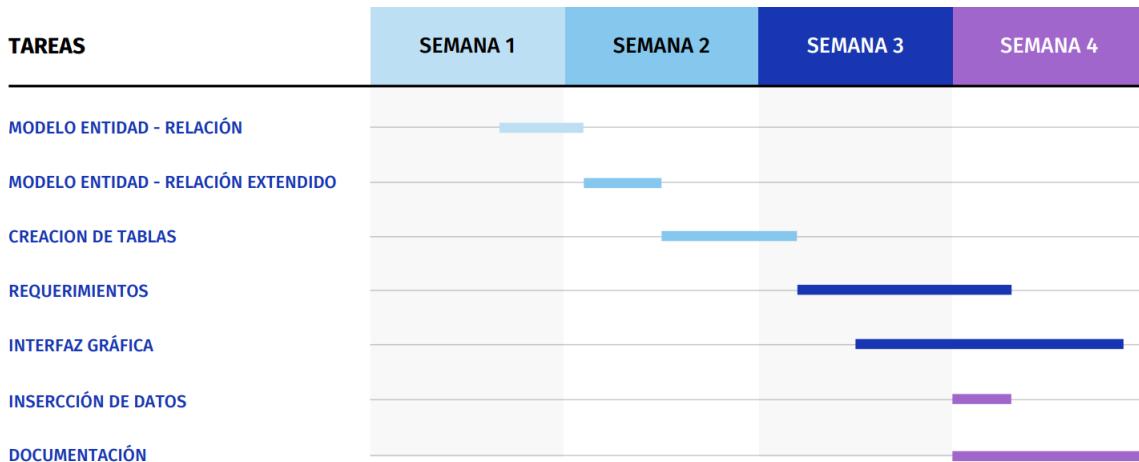


Figura 1: Cronograma de actividades

## DISEÑO

Para un buen diseño del modelo se tuvo que analizar a profundidad lo que se pedía, en primer lugar, las entidades y atributos que las conformaban, esto para que no nos faltara ningún dato dentro de nuestra base y cubriera todos los datos que se nos pedían. Por esto fue importante hacer una buena lectura y entendimiento de lo que se estaba solicitando.

Al igual que un buen diseño de nuestra base de datos para que ésta pudiera trabajar de forma adecuada de acuerdo con los requerimientos, por lo que se buscó hacer un diseño de forma detallada y analizada parte por parte.

### 3.1. MODELO ENTIDAD - RELACIÓN

Sabemos que un modelo entidad - relación describe la estructura de una base de datos con la ayuda de un diagrama. Un modelo de este tipo es un diseño de una base de datos que luego se puede implementar, ya que su diagrama tiene tres componentes principales, entidades que son la representación de un objeto, atributos que hacen referencia las propiedades de una entidad y relaciones las cuales muestran el vínculo que tienen las entidades.

En este se muestra la relación entre conjuntos de entidades. Un conjunto de entidades es un grupo de entidades similares y estas entidades pueden tener atributos. En términos de DBMS, una entidad se puede definir como una tabla o atributo de una tabla en la base de datos, por lo que al mostrar la relación entre las tablas y sus atributos, el diagrama muestra la estructura lógica completa de una base de datos.

A continuación, se presenta el modelo entidad - relación realizado de forma colaborativa como planteamiento de lo pedido:

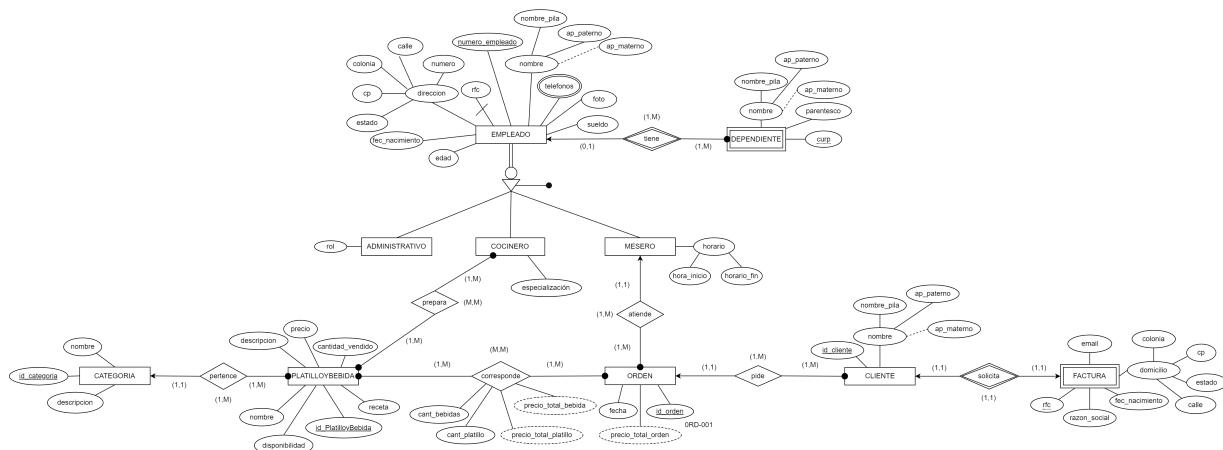


Figura 2: Modelo entidad - relación

Para explicar de forma más concreta cómo se hizo el análisis para el desarrollo del modelo, a continuación se explicarán cada una de las entidades y sus atributos y las relaciones que existen entre ellas.

### 3.1.1. EMPLEADO

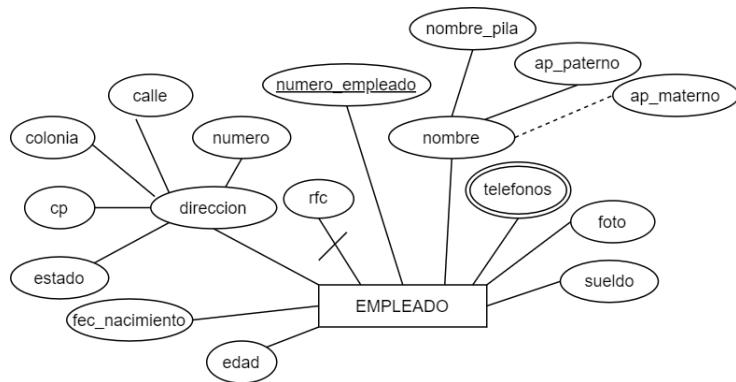


Figura 3: Entidad empleado

Del empleado, buscamos almacenar su edad, fecha de nacimiento, dirección la cual se con forma de estado, colonia, calle y número, además, de su rfc que lo consideramos como clave candidata, no lo escogimos como llave primaria o como llave principal debido a que no sería óptimo utilizarla de esta manera puesto que contiene muchos caracteres.

Además, su nombre que decidimos hacerlo como un atributo compuesto, es decir, guardar por separado nombre, apellido paterno y apellido materno, para poder almacenar los valores de forma separada y poder así identificarlos y un número de empleado que nos va a permitir identificarlo por lo cual será un atributo de valor único y será considerado la llave primaria. Sus teléfonos, como puede tener más de uno este debe de ser un atributo multivaluado, un folio y su sueldo.

### 3.1.2. DEPENDIENTE

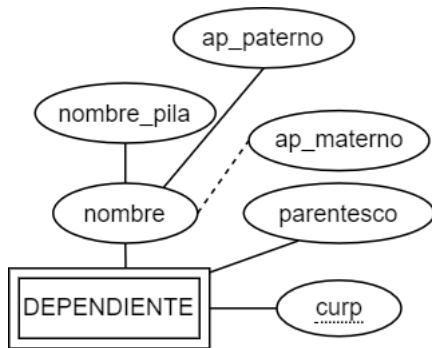


Figura 4: Entidad dependiente

La entidad dependiente la planteamos como una entidad débil ya que no existiría sí no hubiera empleado, de modo que el empleado ayuda a identificar esta entidad. Es importante recalcar que para esta entidad débil no hay dependencia de identidad.

Para esto, nosotros lo que hicimos fue guardar su nombre de la misma forma que lo hicimos para empleado, aparte de almacenar el parentesco que tiene con el empleado y su curp el cual nos sirve como llave primaria de esta entidad donde entendemos que cada uno tiene su clave, pero los dependientes forzosamente están relacionados con un empleado.

### 3.1.3. ADMINISTRATIVO



Figura 5: Entidad administrativo

Para la entrada administrativa podemos ver que solamente se ocupa un atributo llamado rol, este para identificar qué puesto está ejerciendo el administrativo que estamos registrando.

### 3.1.4. COCINERO

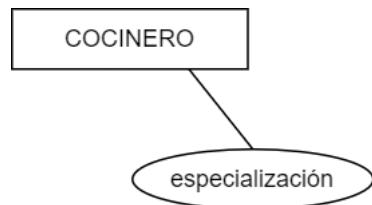


Figura 6: Entidad cocinero

Al igual que para administrativo, podemos ver que para cocinero también contamos solamente con un atributo, que nos permite identificar de la misma forma cuál es el papel que está desempeñando.

### 3.1.5. MESERO

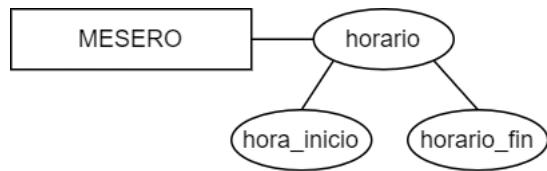


Figura 7: Entidad mesero

Mesero es igual otra entidad con solamente un atributo, pero este atributo es compuesto, que se almacena un horario que nos permite ver la hora de inicio y de fin que trabajo este empleado.

### 3.1.6. CATEGORÍA

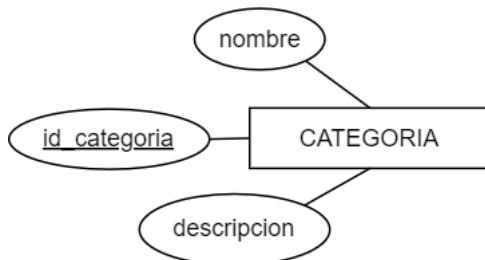


Figura 8: Entidad categoría

La categoría es una entidad que nos permite clasificar a los productos, está confirmada por 3 atributos uno de los cuales funciona como clave principal para este, en este caso un ID que nos permita hacer identificaciones de la categoría de una forma más fácil.

Los demás atributos son la descripción para decir en qué consiste la categoría o decir un poco que tienen en común los productos que

se encuentran dentro de esta y por último un nombre que nos diga el tipo de platillo o bebida de forma muy específica.

### 3.1.7. PLATILLO Y BEBIDA



Figura 9: Entidad platillo y bebida

Para esta entidad es necesario mencionar que, es una de las más importantes ya que como veremos posteriormente es una de las entidades que cuenta con un mayor número de relaciones, para este entidad observaremos que cuenta con 7 atributos, esenciales para que está funcione bien, más aún, su llave principal que es un ID para identidad cada platillo.

Además, contamos con atributos descriptivos para los platillos y bebidas, en este caso, atributos característicos como lo son una descripción para ver la cantidad de producto o alimento, un precio que nos permita conocer el valor del producto, una cantidad de productos

vendidos que nos servirá para llevar un stock, una receta para poder decirle al cliente las características o ingredientes del plato o bebida, su disponibilidad para saber si podemos ofrecerlo o no y por último su nombre que hace referencia al producto ya sea tal cual o de forma característica.

### 3.1.8. ORDEN

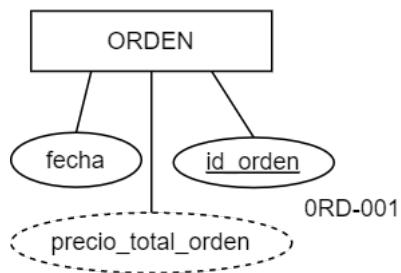


Figura 10: Entidad orden

En orden tenemos la fecha en que se realizó, un atributo calculado el cual es el precio total de la orden, que va de acuerdo a todos los productos que consumió el cliente y por último un ID que nos permite identificar cual fue la orden por si queremos consultarla nuevamente.

Cómo observamos tenemos una nota, esto es para hacer referencia a qué el ID de la orden debe tener el formato presentado para poder identificarlo de una forma adecuada y que sea formal.

### 3.1.9. CLIENTE

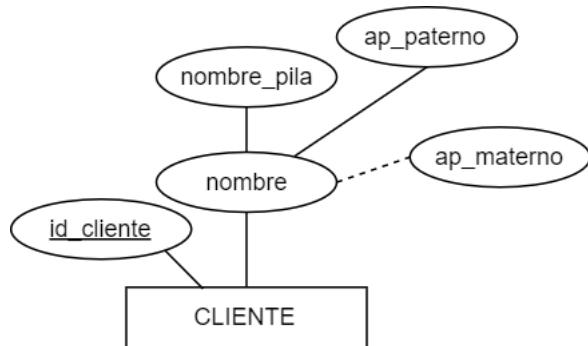


Figura 11: Entidad cliente

Del cliente nos interesa su nombre y crear un ID para modificarlo, esto es importante ya que haremos la unión de cuantos clientes a atendido un mesero y los productos que ha consumido, razón por la cual no interesa utilizar un ID que nos permita identificarlo y conocer también cual fue su consumo y por quien fue atendido.

### 3.1.10. FACTURA

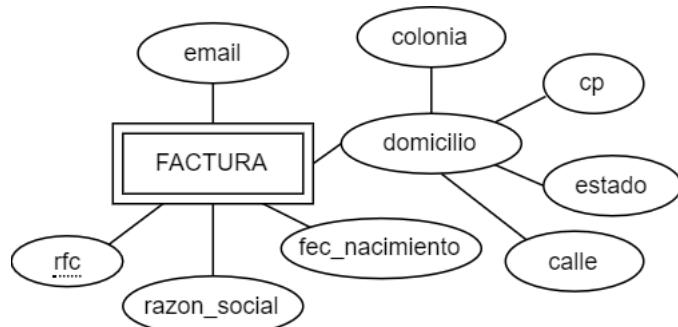


Figura 12: Entidad factura

Al igual que la entidad débil anterior, esta no tiene dependencia de existencia, pero es necesario que un cliente solicite factura para que esta se haga, es decir, es necesario que este lo solicite para hacerlo. En factura necesitamos atributos que nos permitan generar la factura, por ejemplo, un rfc que es el discriminante que permite hacer la distinción del cliente con factura y el que no lo solicita.

Es importante conocer de forma clara que para generar la factura también tener la razón social para generar de forma adecuada y con los datos necesarios esta parte, también la fecha de nacimiento, un email para generar la factura y enviarla a esta y, por último, un atributo compuesto que es el domicilio.

### 3.1.11. RELACIÓN TIENE

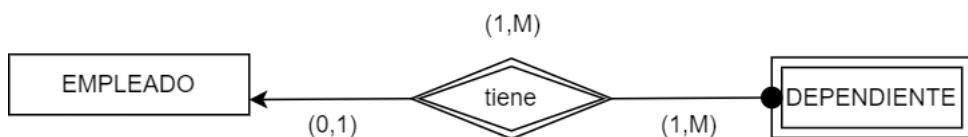


Figura 13: Relación empleado - dependiente

La relación tiene enlaza a dos entidades, al ser débil, esta participación puede ser opcional, por eso tenemos que un empleado puede tener muchos dependientes, mientras que un dependiente solo puede tener 1 empleado relacionado con él pero también, un empleado no puede tener dependientes. Por eso tenemos una relación 1 a muchos,

es decir, 1 empleado puede tener muchos dependientes.

### 3.1.12. RELACIÓN PREPARA

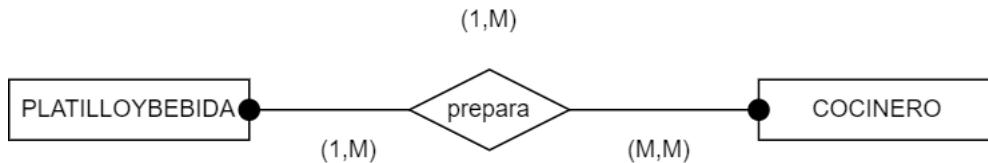


Figura 14: Relación platillo y bebida - cocinero

La relación prepara une el producto con el cocinero, un platillo o bebida puede ser preparado por muchos cocineros ya que si dos clientes piden lo mismo es más eficiente que existan dos cocineros para sacar más rápido el alimento. Mientras que un cocinero puedes preparar muchos platillos o bebidas, puesto que si solo hace uno la atención no sería eficiente y los cocineros serían muchos. Por esto, la relación es un a muchos, ya que un platillo o bebida puede ser preparado por muchos cocineros.

### 3.1.13. RELACIÓN PERTENECE

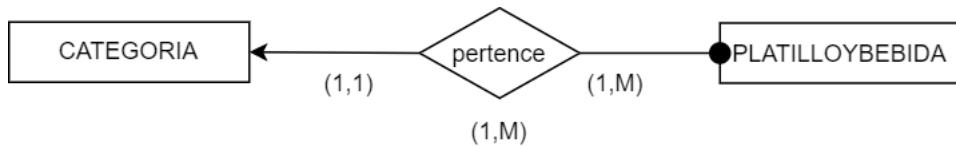


Figura 15: Relación categoría - platillo y bebida

Un platillo pertenece a una categoría específica como puede ser una bebida dulce, o una bebida refrescante o de la misma forma para los alimentos que tienen características en común. Por esto podemos decir, que una categoría tiene muchos platillos en ella que se relacionan por alguna razón o característica, mientras que un platillo solo se puede clasificar en una categoría. Por esta razón, la relación tiene una relación uno a muchos, es decir, a una categoría pueden pertenecer varios platillos o bebidas.

### 3.1.14. RELACIÓN CORRESPONDE

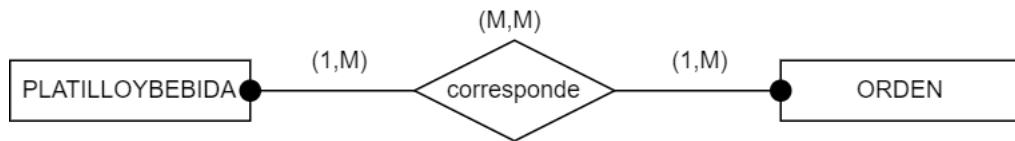


Figura 16: Relación platillo y bebida - orden

Corresponde, relaciona platillos y bebidas con orden, de este modo, tenemos de forma concreta que un cliente puede realizar muchas órdenes, mientras que un platillo o bebida puede ser pedido por muchos clientes, esta es la relación muchos a muchos donde muchos clientes que llegan al restaurante pueden pedir muchos platillos del menú.

### 3.1.15. RELACIÓN PIDE

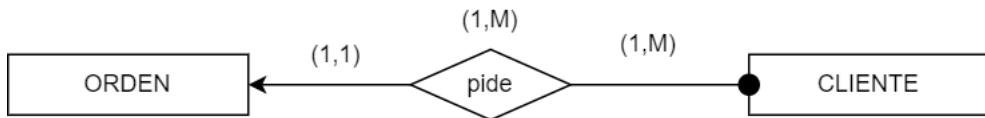


Figura 17: Relación orden - cliente

La relación pide lo que hace es relacional orden y cliente, donde sabemos que como se dijo anteriormente un cliente hace una orden que incluye todos los productos que consumo, mientras que una orden puede ser hecha por muchos clientes ya que son todos los que llegan al restaurante, entonces podemos decir que una orden puede ser realizada por muchos clientes.

### 3.1.16. RELACIÓN SOLICITA

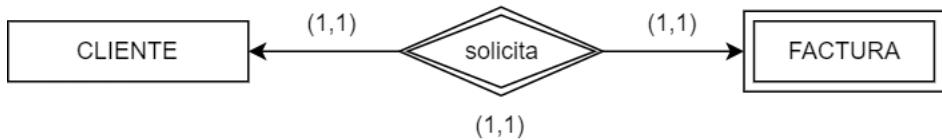


Figura 18: Relación cliente - factura

Solicita es nuestra única relación uno a uno, en esta lo que se hace al ser una entidad débil es que el cliente que pago puede generar su factura, por eso es una relación uno a uno, donde un cliente puede solicitar una factura mientras que una factura puede ser solicitada por un cliente que fue el que pago y puede generarla con sus datos.

### 3.1.17. GENERALIZACIÓN

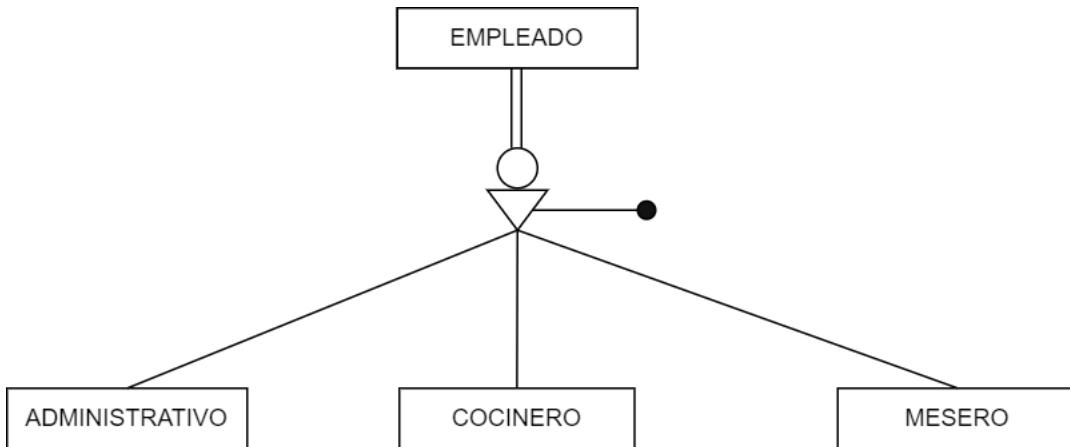


Figura 19: Generalización empleado

Se consideró como generalización debido a que partimos de múltiples entidades las cuales tienen una cosa en común, en este caso atributos, como lo son, nombre, apellido paterno, apellido materno, sueldo, dirección, edad, rfc, entre otros. Por lo tanto se puede realizar un proceso de generalización creando una entidad de nivel superior “EMPLEADO” la cual ya contiene esos atributos en común.

La generalización que se está manejando es una excluyente total, gráficamente podemos apreciar esto porque se representa por una doble raya entre la relación padre y los subtipos y el círculo que nos va a representar la generalización.

Es excluyente total debido a que forzosamente un empleado en

nuestro caso de estudio va a pertenecer a alguno de los subtipos (Administrativo, cocinero, mesero) y la totalidad es vista porque ese el único tipo de empleado que podrá ser. La representación gráfica de la bolita color negro nos representa un discriminante entre cada uno, es decir, si es mesero, administrativo o cocinero.

### **3.2. MODELO RELACIONAL**

Basándonos en el modelo entidad – relación, pudimos realizar nuestro modelo relacional, aquí podemos ver como serán nuestras tablas y el tipo de datos que utilizamos para cada una de ellas, en esta etapa de diseño también se buscaron corregir algunos puntos como lo son los datos creados y ver si nos convenia hacerlo de esta forma, también realizamos la interpretación intermedia para ver de forma concreta como iban a quedar en su versión final las tablas.

El modelo relacional representa la base de datos como una colección de relaciones. Una relación no es más que una tabla de valores. Cada fila de la tabla representa una colección de valores de datos relacionados.

El nombre de la tabla y los nombres de las columnas son útiles para interpretar el significado de los valores en cada fila. Los datos se representan como un conjunto de relaciones. En el modelo relacional,

los datos se almacenan como tablas. Sin embargo, el almacenamiento físico de los datos es independiente de la forma en que los datos están organizados lógicamente.

A continuación, se presentara su representación intermedia junto con el modelo relacional, para cada una de las tablas creadas.

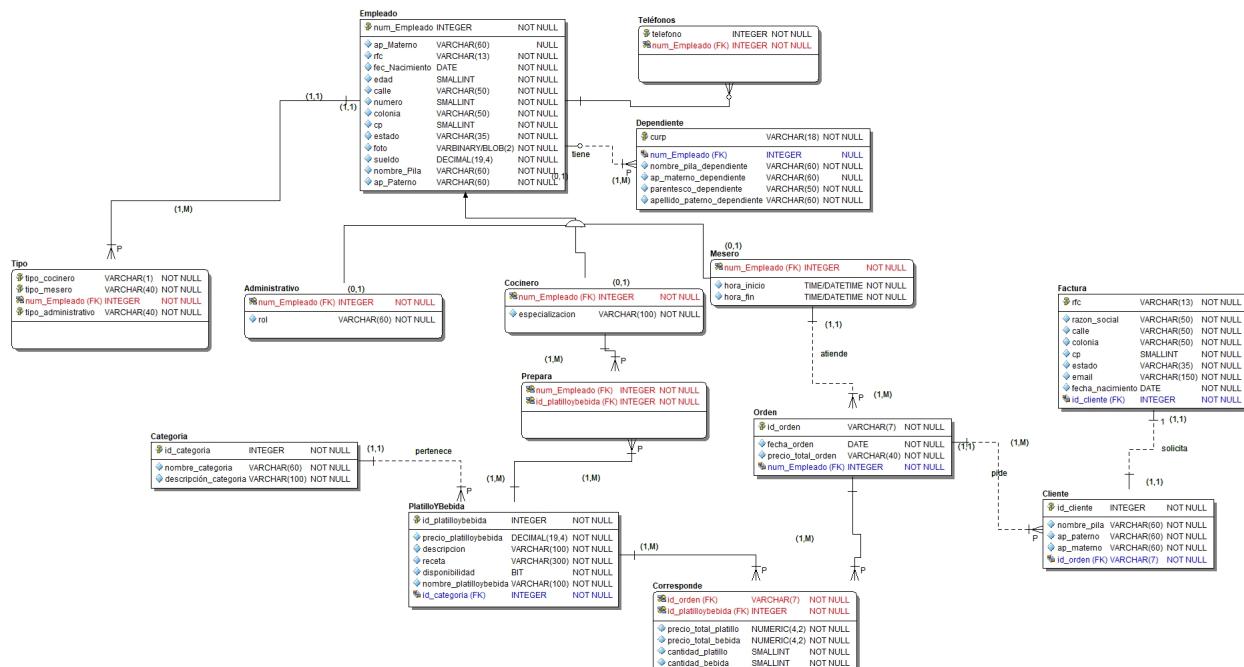


Figura 20: Modelo relacional

### 3.2.1. EMPLEADO

Empleado		
\$ num_Empleado	INTEGER	NOT NULL
ap_Materno	VARCHAR(60)	NULL
rfc	VARCHAR(13)	NOT NULL
fec_Nacimiento	DATE	NOT NULL
edad	SMALLINT	NOT NULL
calle	VARCHAR(50)	NOT NULL
numero	SMALLINT	NOT NULL
colonia	VARCHAR(50)	NOT NULL
cp	SMALLINT	NOT NULL
estado	VARCHAR(35)	NOT NULL
foto	VARBINARY/BLOB(2)	NOT NULL
sueldo	DECIMAL(19,4)	NOT NULL
nombre_Pila	VARCHAR(60)	NOT NULL
ap_Paterno	VARCHAR(60)	NOT NULL

Figura 21: Modelo relacional empleado

EMPLEADO: {num\_empleado int (PK), nombre\_pila varchar(60), ap\_paterno varchar(60), ap\_materno varchar(60)(N), rfc varchar(13)(C), fec\_nacimiento date, edad smallint, calle varchar(50), numero smallint, colonia varchar(50), cp smallint, estado varchar(35), foto by-  
tea,sueldo money}

### 3.2.2. TÉLEFONO

Teléfonos		
\$ telefono	INTEGER	NOT NULL
\$ num_Empleado (FK)	INTEGER	NOT NULL

Figura 22: Modelo relacional teléfono

TELÉFONOS: {teléfono smallint (PK), num\_empleado int (FK)}

### 3.2.3. DEPENDIENTE

Dependiente		
curp	VARCHAR(18)	NOT NULL
num_Emppleado (FK)	INTEGER	NULL
nombre_pila_dependiente	VARCHAR(60)	NOT NULL
ap_materno_dependiente	VARCHAR(60)	NULL
parentesco_dependiente	VARCHAR(50)	NOT NULL
apellido_paterno_dependiente	VARCHAR(60)	NOT NULL

Figura 23: Modelo relacional dependiente

DEPENDIENTE:{num\_empleado int (FK), curp varchar(18)(D)(PK), nombre\_pila\_dependiente varchar(60), apellido\_paterno\_dependiente varchar(60), apellido\_materno\_dependiente varchar(60)(N), parentesco\_dependiente varchar(50)}

### 3.2.4. TIPO

Tipo		
tipo_cocinero	VARCHAR(1)	NOT NULL
tipo_mesero	VARCHAR(40)	NOT NULL
num_Emppleado (FK)	INTEGER	NOT NULL
tipo_administrativo	VARCHAR(40)	NOT NULL

Figura 24: Modelo relacional tipo

TIPO: {[tipo\_cocinero boolean, tipo\_administrativo boolean, tipo\_mesero boolean](PK), num\_empleado integer(FK)}

### 3.2.5. ADMINISTRATIVO

Administrativo		
(0,1)		
num_Emppleado (FK)	INTEGER	NOT NULL
rol	VARCHAR(60)	NOT NULL

Figura 25: Modelo relacional administrativo

**ADMINISTRATIVO:** {num\_empleado\_administrativo int (FK), rol varchar(60)}

### 3.2.6. COCINERO



Figura 26: Modelo relacional cocinero

**COCINERO:** {num\_empleado\_cocinero int (FK), especialización varchar (100)}

### 3.2.7. MESERO



Figura 27: Modelo relacional mesero

**MESERO:** {num\_empleado\_mesero int (FK), hora\_inicio time, hora\_fin time}

### 3.2.8. CATEGORÍA



Figura 28: Modelo relacional categoría

CATEGORIA: {id\_categoria int (PK), nombre\_categoria varchar(60), descripción\_categoria varchar (100)}

### 3.2.9. PLATILLO Y BEBIDA

PlatilloYBebida		
	id_platilloybebida	INTEGER
◆	precio_platilloybebida	DECIMAL(19,4) NOT NULL
◆	descripción	VARCHAR(100) NOT NULL
◆	receta	VARCHAR(300) NOT NULL
◆	disponibilidad	BIT NOT NULL
◆	nombre_platilloybebida	VARCHAR(100) NOT NULL
◆	id_categoria (FK)	INTEGER NOT NULL

Figura 29: Modelo relacional platillo y bebida

PLATILLOYBEBIDA: {id\_platilloybebida int (PK), id\_categoria int (FK), precio\_platilloybebida money, descripción varchar(100), receta varchar (300), disponibilidad bool, nombre\_platilloybebida varchar(100)}

### 3.2.10. CORRESPONDE

Corresponde		
	id_orden (FK)	VARCHAR(7) NOT NULL
◆	id_platilloybebida (FK)	INTEGER NOT NULL
◆	precio_total_platillo	NUMERIC(4,2) NOT NULL
◆	precio_total_bebida	NUMERIC(4,2) NOT NULL
◆	cantidad_platillo	SMALLINT NOT NULL
◆	cantidad_bebida	SMALLINT NOT NULL

Figura 30: Modelo relacional corresponde

CORRESPONDE: {d\_orden character (FK), id\_platilloybebida integer, cant\_platillo smallint, cant\_bebida smallint, precio\_total\_bebida numeric(10,2), precio\_total\_platillo numeric(10,2), [id\_platilloybebida, id\_orden](PK)}

### 3.2.11. PREPARA



Figura 31: Modelo relacional prepara

PREPARA: {[id\_PlatilloyBebida (FK), num\_empleado\_cocinero int (FK)](PK)}

### 3.2.12. ORDEN



Figura 32: Cronograma de actividades

: {id\_Orden varchar(7)(PK), cantidad\_Platillo smallint, cantidad\_Bebida smallint, fecha Date, precio\_Total\_Bebida int (C), precio\_Total\_Platillo int (C), precio\_Total\_Orden int (C) ,num\_empleado\_mesero int (FK)}

### 3.2.13. CLIENTE



Figura 33: Modelo relacional cliente

**CLIENTE:** {id\_Cliente int (PK), nombre\_Pila varchar (60), ap\_Paterno varchar (60), ap\_Materno varchar (60), id\_Orden int (FK)}

### 3.2.14. FACTURA

Factura		
rfc	VARCHAR(13)	NOT NULL
razon_social	VARCHAR(50)	NOT NULL
calle	VARCHAR(50)	NOT NULL
colonia	VARCHAR(50)	NOT NULL
cp	SMALLINT	NOT NULL
estado	VARCHAR(35)	NOT NULL
email	VARCHAR(150)	NOT NULL
fecha_nacimiento	DATE	NOT NULL
id_cliente (FK)	INTEGER	NOT NULL

Figura 34: Modelo relacional factura

**FACTURA:** {rfc varchar(13)(D) (PK), razón\_social varchar(50), calle varchar(50), colonia varchar(50), codigo\_postal smallint, estado varchar(35), email varchar(150), fecha\_nacimiento date, id\_cliente (FK)}

## IMPLEMENTACIÓN

### 4.1. CREACIÓN DE TABLAS

- Creación de tablas
  - Tabla empleado
- ```
CREATE TABLE public."empleado" (
    num_empleado integer NOT NULL,
    nombre character varying(60) NOT NULL,
```

```
ap_paterno character varying(60) NOT NULL,  
ap_materno character varying(60),  
rfc character varying(13) NOT NULL,  
fec_nacimiento date NOT NULL,  
edad smallint NOT NULL, calle character varying(50) NOT NULL,  
numero smallint NOT NULL,  
colonia character varying(50) NOT NULL,  
cp integer NOT NULL,  
estado character varying(35) NOT NULL,  
foto bytea NULL,  
sueldo numeric(10,2) NOT NULL,  
CONSTRAINT empleado_num_empleado_pk PRIMARY KEY (num_empleado));
```

–Tabla tipo empleado

```
CREATE TABLE public."tipo_empleado" (  
tipo_cocinero boolean NOT NULL,  
tipo_administrativo boolean NOT NULL,  
tipo_mesero boolean NOT NULL,  
num_empleado integer NOT NULL);
```

–Tabla telefono

```
CREATE TABLE public."telefono" (  
telefono numeric(10,0) NOT NULL,  
num_empleado integer NOT NULL);
```

–Tabla dependiente

```
CREATE TABLE public.“dependiente” (
curp character varying(18) NOT NULL,
nombre_dep character varying(60) NOT NULL,
ap_paterno_dep character varying(60) NOT NULL,
ap_materno character varying(60),
parentesco_dep character varying(50) NOT NULL,
num_empleado integer NOT NULL);
```

—Tabla administrativo

```
CREATE TABLE public.“administrativo” (
rol character varying(60) NOT NULL,
num_empleado integer NOT NULL);
```

–Tabla cocinero

```
CREATE TABLE public.“cocinero” (
especializacion character varying(100) NOT NULL,
num_empleado integer NOT NULL);
```

–Tabla mesero

```
CREATE TABLE public.“mesero” (
hr_inicio time without time zone NOT NULL,
hr_fin time without time zone NOT NULL,
```

num\_empleado integer NOT NULL);

—tabla categoria

```
CREATE TABLE public.“categoria” (
    id_categoria integer NOT NULL,
    nombre_categoria character varying(60) NOT NULL,
    descripcion_categoria character varying(100) NOT NULL,
    PRIMARY KEY(id_categoria));
```

—tabla platillo y bebida

```
CREATE TABLE public.“platilloybebida” (
    id_platilloybebida integer NOT NULL,
    nombre_platilloybebida character varying(100) NOT NULL,
    descripcion character varying(100) NOT NULL,
    precio_platilloybebida numeric(10,2) NOT NULL,
    receta character varying(300) NOT NULL,
    disponibilidad boolean NOT NULL,
    cantidad_vendido smallint NOT NULL,
    es_platillo boolean NOT NULL,
    es_bebida boolean NOT NULL,
    id_categoria integer NOT NULL,
    PRIMARY KEY (id_platilloybebida));
```

—Tabla prepara

```
CREATE TABLE public.“prepara” (
    num_empleado integer NOT NULL,
    id_platilloybebida INTEGER NOT NULL);
```

—tabla orden

```
CREATE TABLE public.“orden” (
    id_orden character varying(7) NOT NULL,
    fec_orden date NOT NULL,
    precio_total_orden numeric(7, 2) NOT NULL,
    num_empleado integer NOT NULL,
    PRIMARY KEY (id_orden));
```

—tabla cliente

```
CREATE TABLE public.“cliente” (
    id_cliente character varying(10) NOT NULL,
    nombre_cliente character varying(60) NOT NULL,
    ap_paterno character varying(60) NOT NULL,
    ap_materno character varying(60),
    id_orden character varying(7) NOT NULL,
    PRIMARY KEY (id_cliente));
```

—tabla factura

```
CREATE TABLE public.“factura” (
    rfc character varying(13) NOT NULL,
```

```
razon_social character varying(50) NOT NULL,  
calle character varying(50) NOT NULL,  
colonia character varying(50) NOT NULL,  
cp smallint NOT NULL,  
estado character varying(35) NOT NULL,  
email character varying(150) NOT NULL,  
fec_nacimiento date NOT NULL,  
id_cliente character varying(10) NOT NULL,  
PRIMARY KEY (rfc);
```

—tabla corresponde

```
CREATE TABLE public.“corresponde” (  
id_platilloybebida integer NOT NULL,  
id_orden character varying(7) NOT NULL,  
cant_platillo smallint NOT NULL,  
cant_bebida smallint NOT NULL,  
precio_total_bebida numeric(10,2) NOT NULL,  
precio_total_platillo numeric(10, 2) NOT NULL,  
PRIMARY KEY(id_platilloybebida, id_orden));
```

## 4.2. CREACIÓN DE CONSTRAINTS

—Agregando restricciones a las tablas —  
—Restricción orden ALTER TABLE orden ADD CONSTRAINT

```
orden_num_empleado_fk FOREIGN KEY (num_empleado)
REFERENCES public.“empleado” (num_empleado)
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;
```

–Restricción telefono

```
ALTER TABLE telefono
ADD CONSTRAINT telefono_num_empleado_fk
FOREIGN KEY (num_empleado)
REFERENCES public.“empleado” (num_empleado)
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;
```

```
ALTER TABLE telefono
ADD CONSTRAINT telefono_tel_num_empleado_pk
PRIMARY KEY (telefono, num_empleado);
```

–Restricción tipo\_empleado

```
ALTER TABLE tipo_empleado
ADD CONSTRAINT tipo_empleado_num_empleado_fk
FOREIGN KEY (num_empleado)
REFERENCES public.“empleado” (num_empleado)
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;
ALTER TABLE tipo_empleado
ADD CONSTRAINT tipo_empleado_id_pk
PRIMARY KEY (tipo_cocinero, tipo_administrativo,
```

tipo\_mesero, num\_empleado);

–Restricción dependiente

ALTER TABLE dependiente  
ADD CONSTRAINT dependiente\_num\_empleado\_fk  
FOREIGN KEY (num\_empleado)  
REFERENCES public..`empleado"(num\_empleado)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

—Restricción administrativo

ALTER TABLE administrativo  
ADD CONSTRAINT administrativo\_num\_empleado\_fk  
FOREIGN KEY (num\_empleado)  
REFERENCES public.“empleado” (num\_empleado)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;  
ALTER TABLE administrativo  
ADD CONSTRAINT administrativo\_id\_pk  
PRIMARY KEY (num\_empleado);

–Restricción cocinero ALTER TABLE cocinero

ADD CONSTRAINT cocinero\_num\_empleado\_fk  
FOREIGN KEY (num\_empleado)  
REFERENCES public.“empleado” (num\_empleado)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

```
ALTER TABLE cocinero  
ADD CONSTRAINT cocinero_id_pk  
PRIMARY KEY (num_empleado);
```

–Restricción mesero

```
ALTER TABLE mesero  
ADD CONSTRAINT mesero_num_empleado_fk  
FOREIGN KEY (num_empleado)  
REFERENCES public.“empleado” (num_empleado)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;
```

```
ALTER TABLE mesero  
ADD CONSTRAINT mesero_id_pk  
PRIMARY KEY (num_empleado);
```

–Restricción corresponde

```
ALTER TABLE corresponde  
ADD CONSTRAINT corresponde_id_platilloybebida_fk  
FOREIGN KEY (id_platilloybebida)  
REFERENCES public.“platilloybebida” (id_platilloybebida)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;
```

```
ALTER TABLE corresponde  
ADD CONSTRAINT corresponde_id_orden_fk
```

FOREIGN KEY (id\_orden)

REFERENCES public.“orden” (id\_orden) ON UPDATE CASCADE ON  
DELETE CASCADE NOT VALID;

—Restricción prepara

ALTER TABLE prepara  
ADD CONSTRAINT prepara\_num\_empleado\_fk  
FOREIGN KEY (num\_empleado)  
REFERENCES public.“empleado” (num\_empleado)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

ALTER TABLE prepara

ADD CONSTRAINT prepara\_id\_platilloybebida\_fk  
FOREIGN KEY (id\_platilloybebida)  
REFERENCES public.“platilloybebida” (id\_platilloybebida)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

ALTER TABLE prepara

ADD CONSTRAINT prepara\_id\_pk  
PRIMARY KEY (num\_empleado, id\_platilloybebida);

—Restricción platilloybebida

ALTER TABLE platilloybebida  
ADD CONSTRAINT platilloybebida\_id\_categoria\_fk  
FOREIGN KEY (id\_categoria)

REFERENCES public.“categoria” (id\_categoria)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

–Agregando precio platillo y bebida  
–ALTER TABLE platilloybebida  
add column precio\_platilloybebida numeric(10,2);

–Restricción cliente  
ALTER TABLE cliente ADD CONSTRAINT cliente\_id\_orden\_fk  
FOREIGN KEY (id\_orden)  
REFERENCES public.“orden” (id\_orden)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

–Restricción factura  
ALTER TABLE factura  
ADD CONSTRAINT factura\_id\_cliente\_fk  
FOREIGN KEY (id\_cliente)  
REFERENCES public.“cliente” (id\_cliente)  
ON UPDATE CASCADE ON DELETE CASCADE NOT VALID;

#### **4.3. INSERCCIÓN DE DATOS**

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,

foto,sueldo)

```
values (1, 'gabriela','ramirez','ruiz','grrcdmx1584nr','(1980-01-01'),  
42,'bucareli',2888,'gustavo madero',09999,'cdmx',", 5000);
```

```
insert into empleado (num_empleado,nombre,ap_paterno  
,ap_materno,rfc,fec_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)
```

```
values (2, 'ivan','zamorano','pedor','izpcdmx1258nr','(1990-01-01'),  
32,'chapultepec',452,'alvaro obregon',04020,'cdmx',", 4000);
```

```
insert into empleado (num_empleado,nombre,ap_paterno  
,ap_materno,rfc,fec_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)
```

```
values (3, 'aura','perez','campos','apccdmx5145nr','(1995-03-13'),  
27,'roma',158,'coyoacan',04558,'cdmx',", 1000);
```

```
insert into empleado (num_empleado,nombre,ap_paterno  
,ap_materno,rfc,fec_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)
```

```
values (4, 'raul','vazquez','gallo','rvgcdmx2685nr','(1995-02-03'),  
27,'insurgentes',158,'coyoacan',03265,'cdmx',", 2500);
```

```
insert into empleado (num_empleado,nombre,ap_paterno  
,ap_materno,rfc,fec_nacimiento,edad,calle,numero,colonia,cp,estado,
```

foto,sueldo)

values (5, 'alexis','hernandez','romero','ahrcdmx8564nr','(1999-05-12'),  
23,'revolucion',44,'coyoacan',08954,'cdmx',", 3000);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (6, 'alan','becerra','castillo','abccdmx7845nr','(2001-09-15'),  
20,'misterios',1258,'gustavo madero',02584,'cdmx',", 2500);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (7, 'rodrigo','mendoza','cruz','rmccdmx5894nr','(1995-05-03'),  
27,'churubusco',589,'alvaro obregon',01881,'cdmx',", 1000);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (8, 'maria','lopez','sandoval','mlscdmx4587nr','(1991-02-13'),  
31,'chimalistac',4518,'benito juarez',02849,'cdmx',", 3000);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,

foto,sueldo)

values (9, 'jose','perez','montero',jpmcdmx5896nr,'(1999-12-02'),  
21,'amsterdam',5894,'gustavo madero',04497,'cdmx',", 1500);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (10, 'fernando','cruz','marquez',fcmcdmx4589nr,'(2002-10-08'),  
19,'milpa',254,'alvaro obregon',01881,'cdmx',", 1000);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (11, 'eder','avila','sanchez',eascdmx5849nr,'(1950-12-05'),  
71,'juarez',32,'coyoacan',0188,'cdmx',", 1000);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (12, 'carlos','rivera','dias',crdcdmx6954nr,'(1985-04-12'),  
37,'genova',587,'alvaro obregon',07815,'cdmx',", 1500);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,

foto,sueldo)

values (13, 'axel','mendez','pacheco','ampcdmx4587nr','(1984-02-13'),  
38,'regina',125,'benito juarez',0485,'cdmx',", 1800);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (14, 'ana','ramos','solis','arscdmx4587nr','(1962-12-03'),  
59,'valle',1145,'alvaro obregon',04180,'cdmx',", 1500);

insert into empleado (num\_empleado,nombre,ap\_paterno  
,ap\_materno,rfc,fec\_nacimiento,edad,calle,numero,colonia,cp,estado,  
foto,sueldo)

values (15, 'norma','quiroz','estrada','nqecdmx2457nr','(1991-09-02'),  
30,'boulevard',1246,'benito juarez',01808,'cdmx',", 4500);

---

insert into tipo\_empleado (tipo\_cocinero,tipo\_administrativo  
,tipo\_mesero,num\_empleado) values(FALSE,TRUE,FALSE,1);  
-administrativo

insert into tipo\_empleado (tipo\_cocinero,tipo\_administrativo  
,tipo\_mesero,num\_empleado) values(FALSE,TRUE,FALSE,2);

–administrativo

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(FALSE, FALSE, TRUE, 3);
```

–mesero

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(TRUE, FALSE, FALSE, 4);  
–cocinero postres y entradas
```

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(TRUE, FALSE, FALSE, 5);  
–cocinero platos fuertes
```

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(TRUE, FALSE, FALSE, 6);  
–cocinero sopas y ensaladas
```

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(FALSE, FALSE, TRUE, 7);  
–mesero
```

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(TRUE, FALSE, FALSE, 8);
```

–cocinero platos fuertes

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(FALSE, FALSE, TRUE, 9);
```

–mesero

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(FALSE, FALSE, TRUE, 10);
```

–administrativo

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(FALSE, FALSE, TRUE, 11);
```

–mesero

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(FALSE, FALSE, TRUE, 12);
```

–mesero

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(TRUE, FALSE, FALSE, 13);
```

–cocinero bebidas

```
insert into tipo_empleado (tipo_cocinero, tipo_administrativo  
, tipo_mesero, num_empleado) values(TRUE, FALSE, FALSE, 14);
```

-cocinero aperitivos

```
insert into tipo_empleado (tipo_cocinero,tipo_administrativo  
,tipo_mesero,num_empleado) values(FALSE,TRUE,FALSE,15);  
-administrativo
```

---

```
insert into telefono (telefono,num_empleado)  
values(5544667799,1);
```

```
insert into telefono (telefono,num_empleado)  
values(5566442366,2);
```

```
insert into telefono (telefono,num_empleado)  
values(5598756487,3);
```

```
insert into telefono (telefono,num_empleado)  
values(5589563124,4);
```

```
insert into telefono (telefono,num_empleado)  
values(5558795621,5);
```

```
insert into telefono (telefono,num_empleado)
```

```
values(5531254789,6);
```

```
insert into telefono (telefono,num_empleado)  
values(5589563214,7);
```

```
insert into telefono (telefono,num_empleado)  
values(5578945602,8);
```

```
insert into telefono (telefono,num_empleado)  
values(5503210258,9);
```

```
insert into telefono (telefono,num_empleado)  
values(5506320124,10);
```

```
insert into telefono (telefono,num_empleado)  
values(5547895421,11);
```

```
insert into telefono (telefono,num_empleado)  
values(5569998741,12);
```

```
insert into telefono (telefono,num_empleado)  
values(5574856854,13);
```

```
insert into telefono (telefono,num_empleado)
```

```
values(5568489587,14);
```

```
insert into telefono (telefono,num_empleado)  
values(5578989210,15);  
  
-----
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)  
values('ieieiejdhfebrsba3','maria','gutierrez','ruiz','sobrina',1);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)  
values('ters88374hdfretb4','rodolfo','zamorano','leon','hijo',2);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)  
values('resr010273hntrej9','jorge','campos','reyes','abuelo',3);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)  
values('hngbfsdsvrgg71v8e','sofia','perez','campos','hermana',3);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
```

```
parentesco_dep,num_empleado)
```

```
values('t4rng48t1g8rtg1vv','valeria','gallo','nuñez','hija',4);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)
```

```
values('hytgr87t1g8v1e1e8','carolina','flores','romero','hija',5);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)
```

```
values('verGtynr7yoii04i8','mauricio','becerra','ramirez','padre',6);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)
```

```
values('hv2s4ftyvy9h89j98','alberto','mendoza','gonzalez','padre',7);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)
```

```
values('wxctrv54676vf6787','jorge','lopez','sandoval','hermano',8);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,  
parentesco_dep,num_empleado)
```

```
values('mimjuih7ybbh76thj','oscar','campos','montero','tio',9);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
```

```
parentesco_dep,num_empleado)
values('358jfi03m4t04w677','carlos','gutierrez','montero','sobrino',9);

insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('uvebs5n548u9jh4t','felipe','cruz','ruiz','primo',10);

insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('lvdfingu45ht89544','miriam','leon','avila','hijo',11);

insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('vbearuhf8439045wn','pedro','garcia','martinez','primo',12);

insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('23vrtc678ij88iju8','gabriela','rivera','juarez','sobrina',13);

insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('gbnyug5rg789889yh','laura','mendez','pacheco','hermana',13);

insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
```

```
parentesco_dep,num_empleado)
values('vrtdezw3er67y8h90','rocio','pacheco','leon','hijo',13);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('rex434609i9inu00', 'esther','solis','gomez', 'madre',14);
```

```
insert into dependiente (curp,nombre_dep,ap_paterno_dep,ap_materno,
parentesco_dep,num_empleado)
values('q124frr5uuui889km','jaime','estrada','perales', 'abuelo',15);
```

---

```
insert into administrativo(rol, num_empleado)
values ('jefe', 1);
```

```
insert into administrativo(rol, num_empleado)
values ('contador', 2);
```

```
insert into mesero(hr_inicio, hr_fin, num_empleado)
values('09:01:01','15:00:00', 3);
```

```
insert into cocinero(especializacion, num_empleado)
values ('postres', 4);
```

```
insert into cocinero(especializacion, num_empleado)
values ('platos fuertes', 5);
```

```
insert into cocinero(especializacion, num_empleado)
values ('sopas y ensaldas', 6);
```

```
insert into mesero(hr_inicio, hr_fin, num_empleado)
values('09:01:01','15:00:00', 7);
```

```
insert into cocinero(especializacion, num_empleado)
values ('proteinas', 8);
```

```
insert into mesero(hr_inicio, hr_fin, num_empleado)
values('12:01:01','18:00:00', 9);
```

```
insert into administrativo(rol, num_empleado)
values ('supervisor', 10);
```

```
insert into mesero(hr_inicio, hr_fin, num_empleado)
values('15:01:01','21:00:00', 11);
```

```
insert into mesero(hr_inicio, hr_fin, num_empleado)
values('15:01:01','21:00:00', 12);
```

```
insert into cocinero(especializacion, num_empleado)
values ('bebidas', 13);
```

```
insert into cocinero(especializacion, num_empleado)
values ('aperitivos', 14);
```

```
insert into administrativo(rol, num_empleado)
values ('coordinador', 15);
```

#### – CATEGORIAS –

```
INSERT INTO categoria (id_categoria, nombre_categoria,
descripcion_categoria)
VALUES (1,'bebidas calientes','100 % mexicanos');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,
descripcion_categoria)
VALUES (2,'bebidas frias','refrescantes sabores');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,
descripcion_categoria)
VALUES (3,'jugos y frutas','ingredientes naturales');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,
```

descripcion\_categoria)

VALUES (4,'snacks','para iniciar la mañana');

INSERT INTO categoria (id\_categoria, nombre\_categoria,  
descripcion\_categoria)

VALUES (5,'huevos y omelettes','preparados con un toque especial');

INSERT INTO categoria (id\_categoria, nombre\_categoria,  
descripcion\_categoria)

VALUES (6,'almuezos','comidas ligeras');

INSERT INTO categoria (id\_categoria, nombre\_categoria,  
descripcion\_categoria)

VALUES (7,'chilaquiles','tradiccion y sazon mexicanos');

INSERT INTO categoria (id\_categoria, nombre\_categoria,  
descripcion\_categoria)

VALUES (8,'entradas','alimentos para empezar');

INSERT INTO categoria (id\_categoria, nombre\_categoria,  
descripcion\_categoria)

VALUES (9,'tacos','plato para compartir');

INSERT INTO categoria (id\_categoria, nombre\_categoria,

```
descripcion_categoria)  
VALUES (10,'sopas y caldos','con excelente sazon');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (11,'ensaladas','con el perfecto balance de ingredientes');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (12,'hamburguesas y sandwiches','autenticas combinaciones  
en nuestro pan casero');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (13,'carnes','cortes de alta calidad');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (14,'aves','una proteina de calidad');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (15,'pescados','proteinas del mar');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (16,'enchiladas','recetas tradicionales');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (17,'postres','dulce antojo');
```

```
INSERT INTO categoria (id_categoria, nombre_categoria,  
descripcion_categoria)  
VALUES (18,'bebidas refrescantes','frescas opciones para hidratarse');
```

---

### – PLATILLOS Y BEBIDAS –

```
INSERT INTO platilloybebida (id_platilloybebida,  
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,  
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)  
VALUES (1, 'capuchinno','240 ml,incluye refil',40,'cafe espumoso con  
leche entera',TRUE,0,FALSE,TRUE,1);
```

```
INSERT INTO platilloybebida (id_platilloybebida,  
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,  
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)  
VALUES (2,'chocolate caliente','240 ml',50,'chocolate oaxaqueño espu-
```

moso',TRUE,0,0,0,0,0);

INSERT INTO platilloybebida (id\_platilloybebida,  
nombre\_platilloybebida,descripcion, precio\_platilloybebida,receta,  
disponibilidad,cantidad\_vendido,es\_platillo,es\_bebida, id\_categoria)  
VALUES (3,'smoothie de mango','420 ml',40,'acompañado de chile en  
polvo y chamoy',TRUE,0,0,0,0,0);

INSERT INTO platilloybebida (id\_platilloybebida,  
nombre\_platilloybebida,descripcion, precio\_platilloybebida,receta,  
disponibilidad,cantidad\_vendido,es\_platillo,es\_bebida, id\_categoria)  
VALUES (4,'malteada de vainilla','473 ml',50,'acompañada con galle-  
ta de barquillo',TRUE,0,0,0,0,0);

INSERT INTO platilloybebida (id\_platilloybebida,  
nombre\_platilloybebida,descripcion, precio\_platilloybebida,receta,  
disponibilidad,cantidad\_vendido,es\_platillo,es\_bebida, id\_categoria)  
VALUES (5,'jugo de naranja','355 ml',30,'exprimido al momento',TRUE  
,0,0,0,0,0);

INSERT INTO platilloybebida (id\_platilloybebida,  
nombre\_platilloybebida,descripcion, precio\_platilloybebida,receta,  
disponibilidad,cantidad\_vendido,es\_platillo,es\_bebida, id\_categoria)  
VALUES (6,'jugo de piña','355 ml',30,'preparado al momento',TRUE,0,0,0,0,0);

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (7,'hot cakes','3 pzas',60,'con mantequilla y miel de abeja o
jarabe de maple',TRUE,0,TRUE,FALSE,4);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (8,'molletes','4 pzas',70,'dorados con jamon, tocino al gra-
tin',TRUE,0,TRUE,FALSE,4);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (9,'huevos con arrachera','3 pzas',100,'salsa verde al cilan-
tro, acompañados con frijoles',TRUE,0,TRUE,FALSE,5);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (10,'tarascos','2 pzas',60,'huevos fritos sobre tortilla con ja-
mon, salsa y queso',FALSE,0,TRUE,FALSE,5);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (11,'norteño','250 g',70,'machaca a la mexicana y tortillas de
harina',FALSE,0,TRUE,FALSE,6);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (12,'mañanera','120 g',150,'arrachera con chilaquiles rojos'
,TRUE,0,TRUE,FALSE,6);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (13,'enfrijoladas','3 pzas',70,'al gratin con pechuga de po-
llo',TRUE,0,TRUE,FALSE,7);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (14,'chilaquiles verdes','487 kcal',60,'acompañados con tiras
de pollo',TRUE,0,TRUE,FALSE,7);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (15,'queso fundido','150 g',40,'con salsa verda al cilantro y
tortillas de harina',TRUE,0,TRUE,FALSE,8);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (16,'doradas de chicharron','5 pzas',50,'con salsa habanero
y guacamole',TRUE,0,TRUE,FALSE,8);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (17,'taco de pastor','5 pzas',80,'acompañados con gauacamole
y frijoles refritos',TRUE,0,TRUE,FALSE,9);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (18,'tacos dorados','150 g',80,'con crema, queso y salsa',TRUE,
0,TRUE,FALSE,9);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (19,'consome de pollo','250 ml',40,'con arroz blanco y pechuga de pollo',TRUE,0,TRUE,FALSE,10);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (20,'caldo tlalpeño','250 ml',40,'con pechuga de pollo, queso manchego, aguacate y arroz',TRUE,0,TRUE,FALSE,10);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (21,'mediterranea','aderezo al gusto',30,'con pechuga de pollo, lechuga, espinaca y jitomate',TRUE,0,TRUE,FALSE,11);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (22,'san francisco','aderezo al gusto',30,'combinacion de lechugas con pechuga de pollo a la hierbas finas',TRUE,0,TRUE,FALSE,11);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (23,'club sandwich','4 pzas',50,'con pechuga de pollo, tocino,
jampon y queso americano',TRUE,0,TRUE, FALSE,12);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (24,'hamburguesa de pavo','120 g',70,'en pan de ajonjoli con
aguacate, cebolla, acompañada de papas fritas',TRUE,0,TRUE, FALSE,12);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (25,'arrachera','200 g',120,'jugosa y suave, cocinada a la pa-
rrilla, con papas fritas',TRUE,0,TRUE, FALSE,13);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (26,'asado mixto','180 g',150,'carne de res, pechuga de pollo,
pierna de cerdo, queso y nopales',TRUE,0,TRUE, FALSE,13);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (27,'pechuga spicy','150 g',100,'de pollo con ejotes al limon y
papas fritas',TRUE,0,TRUE,FALSE,14);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (28,'sabana','150 g',90,'pechuga de pollo con salsa verde al
cilantro y esquites',TRUE,0,TRUE,FALSE,14);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (29,'filete almendrado','160 g',80,'sobre cebolla francesa, pu-
re de papa y ensalada de lechugas',TRUE,0,TRUE,FALSE,15);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (30,'filete teriyaki','160 g',80,'sobre arroz con verduras y la-
minas de chile habanero',TRUE,0,TRUE,FALSE,15);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (31,'enchiladas originales','3 pzas',60,'de pechuga de pollo
con salsa al gratin',TRUE,0,TRUE,FALSE,16);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (32,'enchiladas de mole','3 pzas',60,'de pechuga de pollo y
mole artesanal',TRUE,0,TRUE,FALSE,16);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (33,'pastel de chocolate','3 capas',80,'pan y cobertura de cho-
colate',TRUE,0,TRUE,FALSE,17);
```

```
INSERT INTO platilloybebida (id_platilloybebida,
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)
VALUES (34,'crepas de cajeta','3 pzas',50,'con helado de vainilla, ca-
rاملizadas con nuez',TRUE,0,TRUE,FALSE,17);
```

```
INSERT INTO platilloybebida (id_platilloybebida,  
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,  
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)  
VALUES (35,'naranjada','355 ml',20,'bebida mineral con fruta natural',TRUE,0, FALSE, TRUE,18);
```

```
INSERT INTO platilloybebida (id_platilloybebida,  
nombre_platilloybebida,descripcion, precio_platilloybebida,receta,  
disponibilidad,cantidad_vendido,es_platillo,es_bebida, id_categoria)  
VALUES (36,'refresco','600 ml',30,'bebida endulzada',TRUE,0, FALSE,  
TRUE,18);
```

---

```
insert into prepara(num_empleado,id_platilloybebida)  
values(13,1);
```

```
insert into prepara(num_empleado,id_platilloybebida)  
values(13,2);
```

```
insert into prepara(num_empleado,id_platilloybebida)  
values(13,3);
```

```
insert into prepara(num_empleado,id_platilloybebida)
```

```
values(13,4);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(13,5);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(13,6);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,7);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,8);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,9);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,10);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,11);
```

```
insert into prepara(num_empleado,id_platilloybebida)
```

```
values(14,12);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(5,13);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(5,14);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,15);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(14,16);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(5,17);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(5,18);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(6,19);
```

```
insert into prepara(num_empleado,id_platilloybebida)
```

```
values(6,20);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,21);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,22);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,23);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,24);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,25);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,26);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,27);
```

```
insert into prepara(num_empleado,id_platilloybebida)
```

```
values(8,28);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,29);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(8,30);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(5,31);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(5,32);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(4,33);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(4,34);
```

```
insert into prepara(num_empleado,id_platilloybebida)
values(13,35);
```

```
insert into prepara(num_empleado,id_platilloybebida)
```

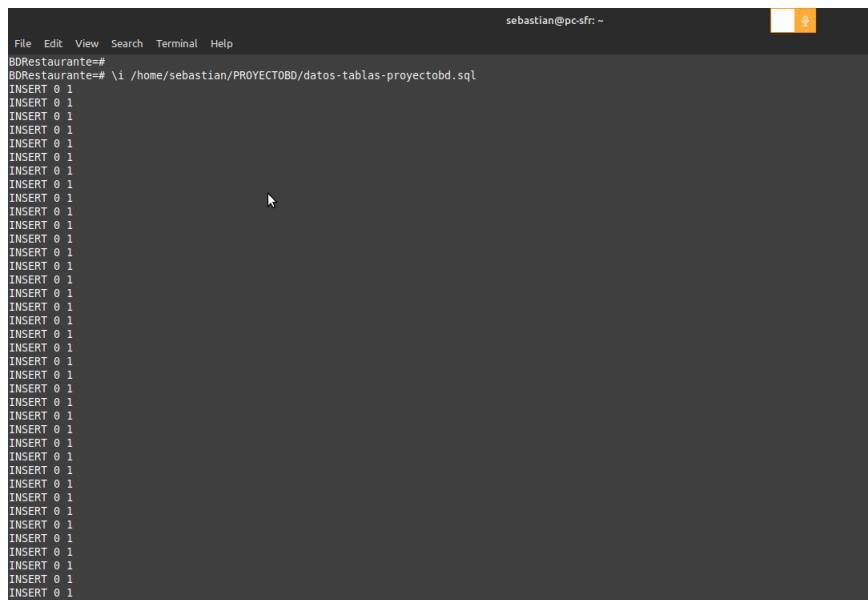
values(13,36);

#### 4.4. GENERACIÓN DE LAS TABLAS POR MEDIO DEL SCRIPT

```
File Edit View Search Terminal Help
postgres=# \c "BDRestaurante"
You are now connected to database "BDRestaurante" as user "postgres".
BDRestaurante=# \i /home/sebastian/PROYECTOBD/creacion-tablas-constraints.sql
CREATE TABLE
ALTER TABLE
BDRestaurante=#
BDRestaurante=#
```

Figura 35: Modelo relacional cliente

#### 4.5. INSERTANDO DATOS POR MEDIO DEL SCRIPT



```
File Edit View Search Terminal Help
BDRestaurante=# \i /home/sebastian/PROYECTOBD/datos-tablas-proyectobd.sql
INSERT 0 1
```

Figura 36: Ejecución del script para agregar los datos vistos en las tablas anteriores

## 4.6. IMPLEMENTACIÓN DE TABLAS EN POSTGRES - SQL

### 4.6.1. TABLA EMPLEADO Y TIPO EMPLEADO

```

File Edit View Search Terminal Help
BDRestaurante=# select * from empleado;
num_empleado | nombre | ap_paterno | ap_materno | rfc | fec_nacimiento | edad | calle | numero | colonia | cp | estado | foto | sueldo
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 | gabriela | ramirez | ruiz | grrcdmx1584nr | 1980-01-01 | 42 | bucareli | 2888 | gustavo madero | 9999 | cdmx | \x | 5000.00
2 | ivan | zamorano | pedor | izpcdmx1258nr | 1990-01-01 | 32 | chapultepec | 452 | alvaro obregon | 4020 | cdmx | \x | 4000.00
3 | aura | perez | campos | apcdmx5145nr | 1995-03-13 | 27 | roma | 158 | coyoacan | 4558 | cdmx | \x | 1000.00
4 | raul | vazquez | gallo | rvgcdmx2685nr | 1995-02-03 | 27 | insurgentes | 158 | coyoacan | 3265 | cdmx | \x | 2500.00
5 | alexis | hernandez | romero | ahrcdmx8564nr | 1999-05-12 | 23 | revolucion | 44 | coyoacan | 8954 | cdmx | \x | 3000.00
6 | alan | becerra | castillo | abcdmx7845nm | 2001-09-15 | 20 | misterios | 1258 | gustavo madero | 2584 | cdmx | \x | 2500.00
7 | rodrigo | mendoza | cruz | rmccdmx5894nr | 1995-05-03 | 27 | churubusco | 589 | alvaro obregon | 1881 | cdmx | \x | 1000.00
8 | maria | lopez | sandoval | mlscdmx4587nr | 1991-02-13 | 31 | chimalistac | 4518 | benito juarez | 2849 | cdmx | \x | 3000.00
9 | jose | perez | montero | jpmcdmx5896nr | 1999-12-02 | 21 | amsterdam | 5894 | gustavo madero | 4497 | cdmx | \x | 1500.00
10 | fernando | cruz | marquez | fmcdmx5896nr | 2002-10-08 | 19 | milpa | 254 | alvaro obregon | 1881 | cdmx | \x | 1000.00
11 | eder | avila | sanchez | eascdmx5849nr | 1950-12-05 | 71 | juarez | 32 | coyoacan | 188 | cdmx | \x | 1000.00
12 | carlos | rivera | dias | crcdmx6954nr | 1985-04-12 | 37 | genova | 587 | alvaro obregon | 7815 | cdmx | \x | 1500.00
13 | axel | mendez | pacheco | ampcdmx4587nr | 1984-02-13 | 38 | regina | 125 | benito juarez | 485 | cdmx | \x | 1800.00
14 | ana | ramos | solis | arscdmx4587nr | 1962-12-03 | 59 | valle | 1145 | alvaro obregon | 4188 | cdmx | \x | 1500.00
15 | norma | quiroz | estrada | nqecdmx2457nr | 1991-09-02 | 30 | boulevard | 1246 | benito juarez | 1888 | cdmx | \x | 4500.00
(15 rows)

BDRestaurante#
BDRestaurante# select * from tipo_empleado;
tipo_cocinero | tipo_administrativo | tipo_mesero | num_empleado
-----+-----+-----+-----+
f | t | f | 1
f | t | t | 2
f | f | f | 3
t | f | f | 4
t | f | f | 5
t | f | f | 6
t | f | t | 7
f | f | f | 8
f | f | t | 9
f | f | t | 10
f | f | t | 11
f | f | t | 12
t | f | f | 13
t | f | f | 14
f | t | f | 15
(15 rows)

```

Figura 37: Se muestran ambas tablas de ambas

### 4.6.2. TABLA DEPENDIENTE DE EMPLEADO

```

File Edit View Search Terminal Help
BDRestaurante# select * from dependiente;
curp | nombre dep | ap_paterno dep | ap_materno | parentesco_dep | num_empleado
-----+-----+-----+-----+-----+-----+
ieieie1dhfebrsa3 | maria | gutierrez | ruiz | sobrina | 1
ters88374hdffretb4 | rodolfo | zamorano | leon | hijo | 2
resr010273hntrej9 | jorge | campos | reyes | abuelo | 3
hngbfsdsvrg71v8e | sofia | perez | campos | hermana | 3
t4rng4stlg8rtglvv | valeria | gallo | nuñez | hija | 4
hytr87flg8vlele8 | carolina | flores | romero | hija | 5
vergtym7yo10418 | mauricio | becerra | ramirez | padre | 6
hvzsafyyu9h89j98 | alberto | mendoza | gonzalez | padre | 7
wxctrv54676vf76787 | jorge | lopez | sandoval | hermano | 8
nimjuir7ybbh76thj | oscar | campos | montero | tio | 9
358f163m4t04w677 | carlos | gutierrez | montero | sobrino | 9
uvebsgn54809j4t | felipe | cruz | ruiz | primo | 10
tvdfing5h5m9999m | miriam | leon | avila | hijo | 11
vbeamfaf8m9999m | petro | garcia | martinez | primo | 12
23vrcte78ij80ij08 | gabriela | rivera | juarez | sobrina | 13
gbywv789899yh | lauri | mendez | pacheco | hermana | 13
vttdczu2d67y8h98 | rocio | pacheco | leon | hijo | 13
rexed3469919inu00 | esther | solis | gomez | madre | 14
q124fr5uuu1869km | jaime | estrada | perales | abuelo | 15
(19 rows)

BDRestaurante#
BDRestaurante# select * from administrativo;
rol | num_empleado
-----+-----+
jefe | 1
contador | 2
supervisor | 10
coordinador | 15
(4 rows)

BDRestaurante#
BDRestaurante#
BDRestaurante#
BDRestaurante#
BDRestaurante#

```

Figura 38: Se muestra la tabla dependiente

#### 4.6.3. TABLA ADMINISTRATIVO, COCINERO, MESERO

```

sebastian@pc-sfr: ~
File Edit View Search Terminal Help
BDRestaurante=# 
BDRestaurante=# 
BDRestaurante=# 
BDRestaurante=#
select * from administrativo ;
+-----+-----+
| rol | num_empleado |
+-----+-----+
jefe	1
contador	2
supervisor	10
coordinador	15
+-----+-----+
(4 rows)

BDRestaurante=#
select * from cocinero;
+-----+-----+
| especializacion | num_empleado |
+-----+-----+
postres	4
platos fuertes	5
sopas y ensaladas	6
proteinas	8
bebidas	13
aperitivos	14
+-----+-----+
(6 rows)

BDRestaurante=#
select * from mesero;
+-----+-----+-----+
| hr_inicio | hr_fin | num_empleado |
+-----+-----+-----+
09:01:01	15:00:00	3
09:01:01	15:00:00	7
12:01:01	18:00:00	9
15:01:01	21:00:00	11
15:01:01	21:00:00	12
+-----+-----+-----+
(5 rows)

BDRestaurante=#
BDRestaurante=#
BDRestaurante=#
BDRestaurante=#
BDRestaurante=#
BDRestaurante=#
BDRestaurante=#

```

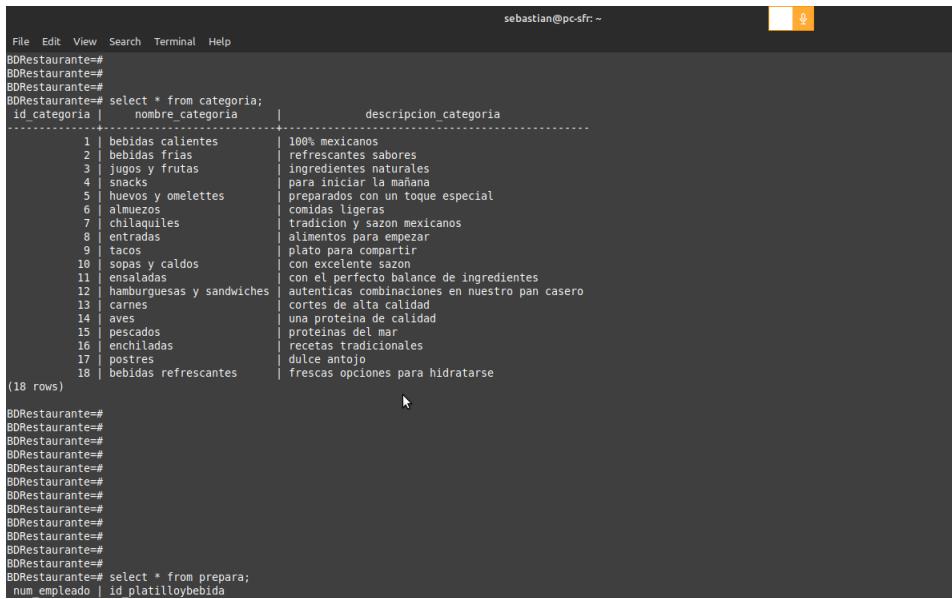
Figura 39: Se muestran las tablas de administrativo, cocinero y mesero

#### 4.6.4. TABLA PLATILLO Y BEBIDA

| id | platilloybebida       | nombre               | descripcion | precio_platilloybebida | receta  |                                                       | disponibilidad |   |
|----|-----------------------|----------------------|-------------|------------------------|---------|-------------------------------------------------------|----------------|---|
|    |                       |                      |             |                        | vendido | platillo                                              | bebida         |   |
| 1  | capuchino             | 240 ml,incluye refil | t           | 1                      | 40.00   | cafe espumoso con leche entera                        | f              | t |
| 2  | chocolate caliente    | 240 ml               | t           | 1                      | 50.00   | chocolate oaxaqueño espumoso                          | f              | t |
| 3  | smoothie de mango     | 420 ml               | t           | 2                      | 40.00   | acompañado de chile en polvo y chamoy                 | f              | t |
| 4  | malteada de vainilla  | 473 ml               | t           | 2                      | 50.00   | acompañada con galleta de barquillo                   | f              | t |
| 5  | jugo de naranja       | 355 ml               | t           | 3                      | 30.00   | exprimido al momento                                  | f              | t |
| 6  | jugo de piña          | 355 ml               | t           | 3                      | 30.00   | preparado al momento                                  | f              | t |
| 7  | hot cakes             | 3 pzas               | t           | 4                      | 60.00   | con mantequilla y miel de abeja o jarabe de maple     | f              | t |
| 8  | moltedo               | 4 pzas               | t           | 4                      | 70.00   | dorados con jamon, tocino al gratin                   | f              | t |
| 9  | huevos con arrachera  | 3 pzas               | f           | 4                      | 100.00  | salsa verde al cilantro, acompañados con frijoles     | f              | t |
| 10 | tarascos              | 2 pzas               | f           | 5                      | 60.00   | huevos fritos sobre tortilla con jamon, salsa y queso | f              | f |
| 11 | norteño               | 250 g                | f           | 5                      | 70.00   | machaca a la mexicana y tortillas de harina           | f              | f |
| 12 | mañanera              | 120 g                | f           | 6                      | 150.00  | arrachera con chilaquiles rojos                       | f              | t |
| 13 | enfrijoladas          | 3 pzas               | f           | 6                      | 70.00   | al gratin con pechuga de pollo                        | f              | t |
| 14 | chilaquiles verdes    | 487 kcal             | f           | 7                      | 60.00   | acompañados con tiras de pollo                        | f              | t |
| 15 | queso fundido         | 150 g                | f           | 7                      | 40.00   | con salsa verda al cilantro y tortillas de harina     | f              | t |
| 16 | doradas de chicharrón | 5 pzas               | f           | 8                      | 50.00   | con salsa habanero y guacamole                        | f              | t |
| 17 | taco de pastor        | 5 pzas               | f           | 8                      | 80.00   | acompañados con guacamole y frijoles refritos         | f              | t |
| 18 | tacos dorados         | 150 g                | f           | 9                      | 80.00   | con crema, queso y salsa                              | f              | t |

Figura 40: Se muestra la tabla de platillo y bebida

#### 4.6.5. TABLA CATEGORÍA

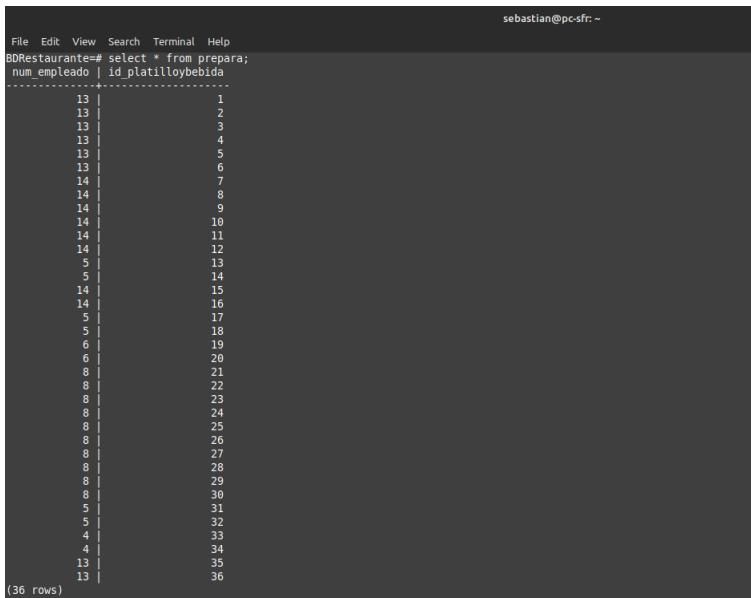


```
File Edit View Search Terminal Help
BDRestaurante=# 
BDRestaurante=# 
BDRestaurante=# 
BDRestaurante=# select * from categoria;
+-----+-----+
| id_categoria | nombre_categoria | descripcion_categoria |
+-----+-----+
1	bebidas calientes	100% mexicanos
2	bebidas frias	refrescantes sabores
3	jugos y frutas	ingredientes naturales
4	snacks	para iniciar la mañana
5	huevos y omelettes	preparados con un toque especial
6	almuezos	comidas ligeras
7	chilaquiles	tradicion y sazon mexicanos
8	entradas	alimentos para empezar
9	tacos	plato para compartir
10	sopas y caldos	con excelente sazon
11	ensaladas	con el perfecto balance de ingredientes
12	hamburguesas y sandwiches	autenticas combinaciones en nuestro pan casero
13	carnes	cortes de alta calidad
14	aves	una proteina de calidad
15	pescados	proteinas del mar
16	enchiladas	recetas tradicionales
17	postres	dulce antojo
18	bebidas refrescantes	frescas opciones para hidratarse
+-----+-----+
(18 rows)

BDRestaurante=# 
BDRestaurante=#
BDRestaurante=#
num_empleado | id_platilloybebida
```

Figura 41: Se muestra la tabla de categoría

#### 4.6.6. TABLA PREPARA



```
File Edit View Search Terminal Help
BDRestaurante=# select * from prepara;
+-----+-----+
| num_empleado | id_platilloybebida |
+-----+-----+
13	1
13	2
13	3
13	4
13	5
13	6
14	7
14	8
14	9
14	10
14	11
14	12
5	13
5	14
14	15
14	16
5	17
5	18
6	19
6	20
8	21
8	22
8	23
8	24
8	25
8	26
8	27
8	28
8	29
8	30
5	31
5	32
4	33
4	34
13	35
13	36
+-----+-----+
(36 rows)
```

Figura 42: se muestra la tabla prepara

Las demás tablas se llenaran en tiempo de ejecución.

## 4.7. FUNCIONES IMPLEMENTADAS

### 4.7.1. LISTA DE FUNCIONES, VISTAS, PROCEDIMIENTOS Y SECUENCIAS

Se muestra una lista de todas las operaciones implementadas para la generación de requerimientos del caso de estudio, en secciones anteriores se explicará más a detalle cada una de ellas. Se insertan en un script y al ejecutar creamos todas las siguientes:

\*Function concatena\_folio\_function(in\_idorden bigint) returns character varying(7)

\*SEQUENCE orden\_seq

\*function agregar\_orden(num\_e integer) returns void

\* function agregar\_orden(idorden character varying(7),num\_e integer) returns void

\*function disponible\_bool(in\_idplatilloybebida integer) returns bool

\*function agregar\_producto(id\_pb integer, cantidad integer) returns

void

\*function agregar\_producto(idordenn character varying(7),id\_pb integer, cantidad integer)

\* view orden\_view

\* index ind\_fecha on orden("fec\_orden");

\*function total\_dia( in num\_emp int, out Cantidad\_Ordenes int, out Total\_Ordenes int)

\*view mas\_vendido

\*view no\_disponible

\*view facturaaa

\*function ventas\_fecha(in inicio date, out Numero\_Ventas int, out Monto\_Total int)

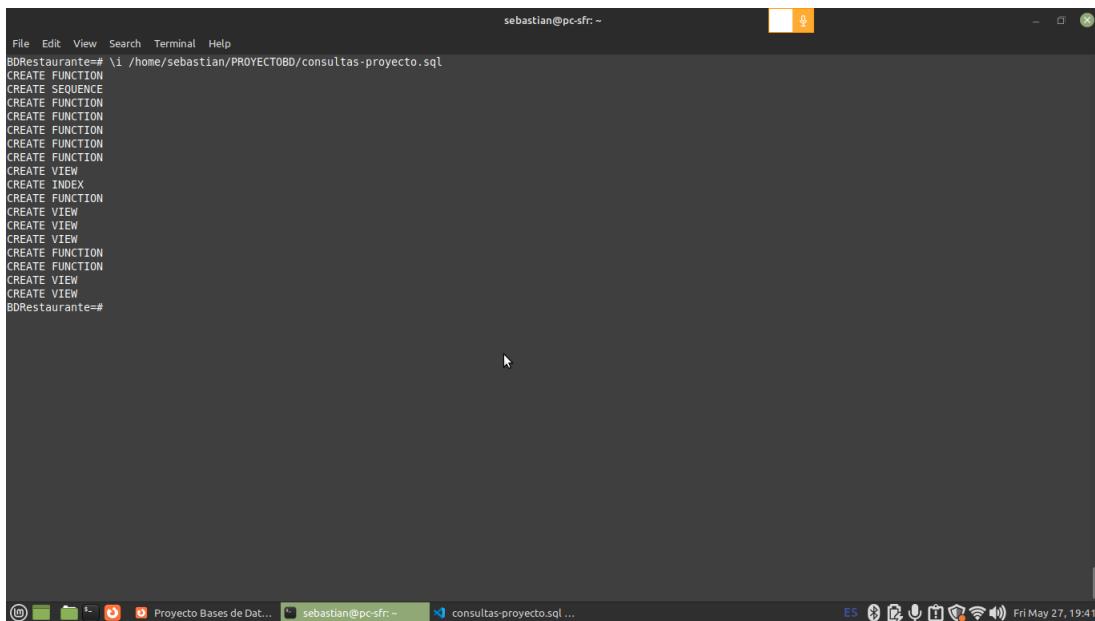
\*function ventas\_fecha(in inicio date, in fin date, out Numero\_Ventas int, out Monto\_Total int)

\*view datos\_cliente

\*view datos\_producto

Cada una tendrá su utilidad en la generación de la implementación para los requerimientos.

#### **4.7.2. CREACIÓN DE LAS FUNCIONES, VISTAS, SECUENCIAS, PROCEDIMIENTOS. EJECUTANDO EL SCRIPT DONDE SE ENCUENTRAN PARA SU CREACIÓN.**



The screenshot shows a terminal window titled 'sebastian@pc-sfr: ~'. The window displays the command 'BDRestaurante=# \i /home/sebastian/PROYECTOBD/consultas-proyecto.sql' followed by numerous 'CREATE' statements for functions, sequences, and views. The terminal is running on a Linux desktop environment, with the taskbar at the bottom showing icons for the terminal, file manager, and system status.

```
File Edit View Search Terminal Help
BDRestaurante=# \i /home/sebastian/PROYECTOBD/consultas-proyecto.sql
CREATE FUNCTION
CREATE SEQUENCE
CREATE FUNCTION
CREATE VIEW
CREATE INDEX
CREATE FUNCTION
CREATE VIEW
CREATE VIEW
CREATE VIEW
CREATE FUNCTION
CREATE FUNCTION
CREATE VIEW
CREATE VIEW
BDRestaurante=#
```

Figura 43: Se muestra la ejecución del script donde vienen todas las operaciones, código, consultas, funciones, vistas, procedimientos y secuencias

## **4.8. REQUERIMIENTOS E IMPLEMENTACIÓN DE LOS PUNTOS, CONSULTAS, FUNCIONES**

### **CONSIDERACIONES**

- Puede haber distintas soluciones al problema.
  
- Los requerimientos enlistados anteriormente, deben ser realizados por medio de PgSQL, con los elementos que se consideren adecuados para resolverlos.
  
- El folio de la orden debe tener un formato similar a ORD-001, prefijo ORD, seguido de un guión y un número secuencial.
  
- Donde este presente el atributo domicilio, está compuesto por estado, código postal, colonia, calle y número.
  
- Donde este presente el atributo nombre, está compuesto por nombre, apellido paterno y materno.
  
- El diseño debe satisfacer todos los principios de diseño, los requerimientos anteriores y un buen manejo de información.

## FUNCIONES GENERALES

Creamos una secuencia con un rango de 1-999 para generar de forma automática el número subsecuente al anterior para poder tener el orden especificado para el folio de la orden (ORD-001, ORD-002, etc).

```
-- secuencia id_orden
CREATE SEQUENCE orden_seq
START WITH 0
INCREMENT BY 1
MAXVALUE 999
MINVALUE 0
CYCLE;
```

Figura 44: Creación de secuencia

Una vez teniendo nuestra secuencia procedemos a crear una función la cual nos permite hacer la concatenación de la cadena “ORD-” con el número secuencial generado. Esta función recibe como parámetro una secuencia(bigint) y nos retorna un character varying el cual representa el folio de la orden con el formato indicado.

```
create or replace function concatena_folio_function(in_idorden bigint)
returns character varying(7) as
$$
declare
folio character varying(7);
idc character varying(3);
ord character varying(4);
cerocero character varying(2);
cero character varying(1);
declare
conteo smallint;
begin
idc = cast( in_idorden as character varying );
ORD = 'ORD-';
cerocero = '00';
cero = '0';
if (length(idc) = 1) then
folio= ORD||cerocero||idc;
return (folio);
elsif (length(idc) = 2) then
folio= ORD||cero||idc;
return (folio);
elsif (length(idc) = 3) then
folio:= ORD||idc;
return (folio);
else
RAISE NOTICE 'nada';
end if;
RAISE NOTICE 'Folio generado(%)', folio;
end
$$
LANGUAGE plpgsql;
```

Figura 45: Se muestra la función que concatena el folio

Es importante mencionar que al momento de guardar la información de las ordenes en la BD debemos hacer uso de la instrucción currval('orden\_seq') ya que nos permite recuperar el valor del id\_orden del cliente actual.

Se crearon dos vistas “datos\_cliente” y “datos\_producto” respectivamente con la finalidad de mostrar de una mejor manera los datos de la factura.

```
create view datos_cliente as
select orden.id_orden as folio_orden, orden.precio_total_orden,
cliente.nombre_cliente, cliente.ap_paterno, cliente.ap_materno ,
factura.rfc, factura.calle, factura.colonia, factura.estado,
factura.email, factura.fec_nacimiento from orden inner join cliente on
orden.id_orden=cliente.id_orden
      inner join factura on factura.id_cliente=cliente.id_cliente
where orden.id_orden = concatena_folio_function(currval('orden_seq'));
```

Figura 46: Vistas para mayor facilidad de captura de factura

```
create view datos_producto as
select platilloybebida.id_platilloybebida,
platilloybebida.nombre_platilloybebida, corresponde.cant_platillo,
corresponde.cant_bebida,platilloybebida.precio_platilloybebida as
precio_unitario, corresponde.precio_total_platillo,
corresponde.precio_total_bebida
from platilloybebida inner join corresponde on
platilloybebida.id_platilloybebida=corresponde.id_platilloybebida where
corresponde.id_orden = concatena_folio_function(currval('orden_seq'));
```

Figura 47: Vistas para mayor facilidad de captura

Vista para que quede más presentable la orden. Una vez guardados todos los datos de la orden se puede mostrar un desglose de la misma con la vista llamada “orden\_view” en la que se utiliza INNER JOIN para poder hacer la conexión entre las tablas que tienen algún campo de interés para este punto.

```
-- VISTA ADICIONAL PARA TENER MÁS PRESENTABLE LA ORDEN
create view orden_view as
select orden.id_orden as Folio_orden,orden.fec_orden as Fecha_orden,
orden.num_empleado, platilloybebida.id_platilloybebida as id_producto,
platilloybebida.nombre_platilloybebida as Nombre_producto,
platilloybebida.precio_platilloybebida as Precio_producto, |
corresponde.cant_platillo as Cantidad_platillo,
corresponde.precio_total_platillo, corresponde.cant_bebida as
Cantidad_bebida, corresponde.precio_total_bebida,
orden.precio_total_orden
from orden inner join corresponde on
orden.id_orden=corresponde.id_orden
      inner join platilloybebida on
platilloybebida.id_platilloybebida=corresponde.id_platilloybebida;
```

Figura 48: Vista orden\_view

## 4.9. REQUERIMIENTOS

**4.9.1. Cada que se agregue un producto a la orden, debe actualizarse los totales (por producto y venta), así como validar que el producto este disponible.**

Se implementa una función para verificar si un producto (platillo o bebida) existe y se encuentra o no disponible al momento de intentar insertarlo en una orden.

Para ello se utilizan sentencias ‘if’ las cuales nos permiten verificar en primera instancia si el producto está registrado en nuestra BD mostrando los mensajes correspondientes según sea el caso, si el producto se encuentra en la BD se procede a verificar su disponibilidad de igual forma se muestran los mensajes correspondientes dependiendo si el producto está disponible o no.

```
create or replace function disponible_bool(in_idplatilloybebida
integer) returns bool as
$$
declare
esta_disponible boolean;
begin
if in_idplatilloybebida = (select id_platilloybebida from
platilloybebida)
where in_idplatilloybebida = platilloybebida.id_platilloybebida )then
raise notice 'El platillo existe';
if (select disponibilidad from platilloybebida
where in_idplatilloybebida = platilloybebida.id_platilloybebida) is
TRUE then
raise notice 'Claro este producto esta disponible';
esta_disponible = TRUE;
return(esta_disponible);
elsif (select disponibilidad from platilloybebida
where in_idplatilloybebida = platilloybebida.id_platilloybebida ) is
FALSE then
esta_disponible = FALSE;
raise exception 'Lo sentimos, este producto ya no está disponible :(';
else
raise exception 'Este platillo no se encuentra en nuestra lista de
opciones';
end if;
else
raise exception 'Este platillo no se encuentra en nuestra lista de
opciones';
end if;
raise notice 'Claro este producto esta disponible';
return(esta_disponible );
end;

$$
LANGUAGE plpgsql;
```

Figura 49: Se muestra la función que verifica la disponibilidad de un platillo o bebida

Para recabar las órdenes de forma más eficiente se implementó una función que recibe como parámetro el id del mesero que capturó la orden , para ello primero se verifica que el id ingresado efectivamente sea de un mesero si no lo es se mostrará el mensaje “No es un mesero”, si se trata de un id válido de mesero se generan de forma automática los campos necesarios para guardar la información de la orden en la tabla “orden” recordando que el folio se genera con la función ‘concatena\_folio\_function’ y la fecha se obtiene con la sentencia ‘current\_date’

```
create or replace function agregar_orden(num_e integer) returns void as
-- solo se agrega numero de empleado;
$$
begin
if exists(select num_empleado from empleado where num_empleado= num_e)
then
if tipo_empleado.tipo_mesero='f' from tipo_empleado where
tipo_empleado.num_empleado=num_e then
    raise exception 'No es un mesero';
else
    insert into orden(id_orden, fec_orden, precio_total_orden,
num_empleado)
    values ( (concatena_folio_function(nextval('orden_seq'))), (select
current_date), 0, num_e);
    end if;
else
raise exception 'No existe este empleado';
end if;
end;
$$
LANGUAGE plpgsql;
```

Figura 50: Función para agregar orden y asignar folio

La función “agregar\_producto” como intuitivamente se puede notar nos permite agregar productos a la orden actual, recibe dos parámetros los cuales son el id del producto y la cantidad de unidades que se desean de dicho producto.

Para el funcionamiento de la función es necesario hacer uso algunas de las funciones previamente creadas, en específico de la función para obtener el folio de la orden y de la función para verificar que la existencia y disponibilidad del producto en cuestión.

De forma simple la función verifica que el producto exista y esté disponible, si es así se procede a agregar los detalles de dicho producto en la tabla “corresponde” ya que esta nos muestra los detalles de los productos y la orden en la que están registrados, posterior a esto

se actualiza el atributo cantidad vendida correspondiente al producto para poder hacer futuras consultas.

Cabe mencionar que todos los productos se agregarán a la orden actual hasta que se genere una nueva orden.

```
create or replace function agregar_producto(id_pb integer, cantidad
integer) returns void as
$$
declare
orden_new character varying(7);
cantidaddp integer;
cantidaddb integer;
begin
orden_new = concatena_folio_function(currval('orden_seq'));
if (select disponible_bool(id_pb)) is TRUE then
if (select es_platillo from platilloybebida where
platilloybebida.id_platilloybebida = id_pb ) is true then
cantidaddp = cantidad;
cantidaddb = 0;
insert into corresponde(id_platilloybebida, id_orden,
cant_platillo,cant_bebida, precio_total_bebida, precio_total_platillo)
values (id_pb, orden_new, cantidaddp, cantidaddb, 0, (select
precio_platilloybebida from platilloybebida where id_pb =
platilloybebida.id_platilloybebida) * cantidaddp);
-- se actualiza el atributo cantidad_vendido para una proxima consulta
update platilloybebida set cantidad_vendido = (select cantidad_vendido
from platilloybebida where platilloybebida.id_platilloybebida = id_pb)
+ cantidaddp where id_platilloybebida = id_pb;

elseif (select es_bebida from platilloybebida where
platilloybebida.id_platilloybebida = id_pb ) is true then
cantidaddb = cantidad;
cantidaddp = 0;
insert into corresponde(id_platilloybebida, id_orden,
cant_platillo,cant_bebida, precio_total_bebida, precio_total_platillo)
values ( id_pb, orden_new, cantidaddp, cantidaddb, (select
precio_platilloybebida from platilloybebida where id_pb =
platilloybebida.id_platilloybebida) * cantidaddb), 0);
-- se actualiza el atributo cantidad_vendido para una proxima consulta
update platilloybebida set cantidad_vendido = (select cantidad_vendido
from platilloybebida where platilloybebida.id_platilloybebida = id_pb)
+ cantidaddb where id_platilloybebida = id_pb;
```

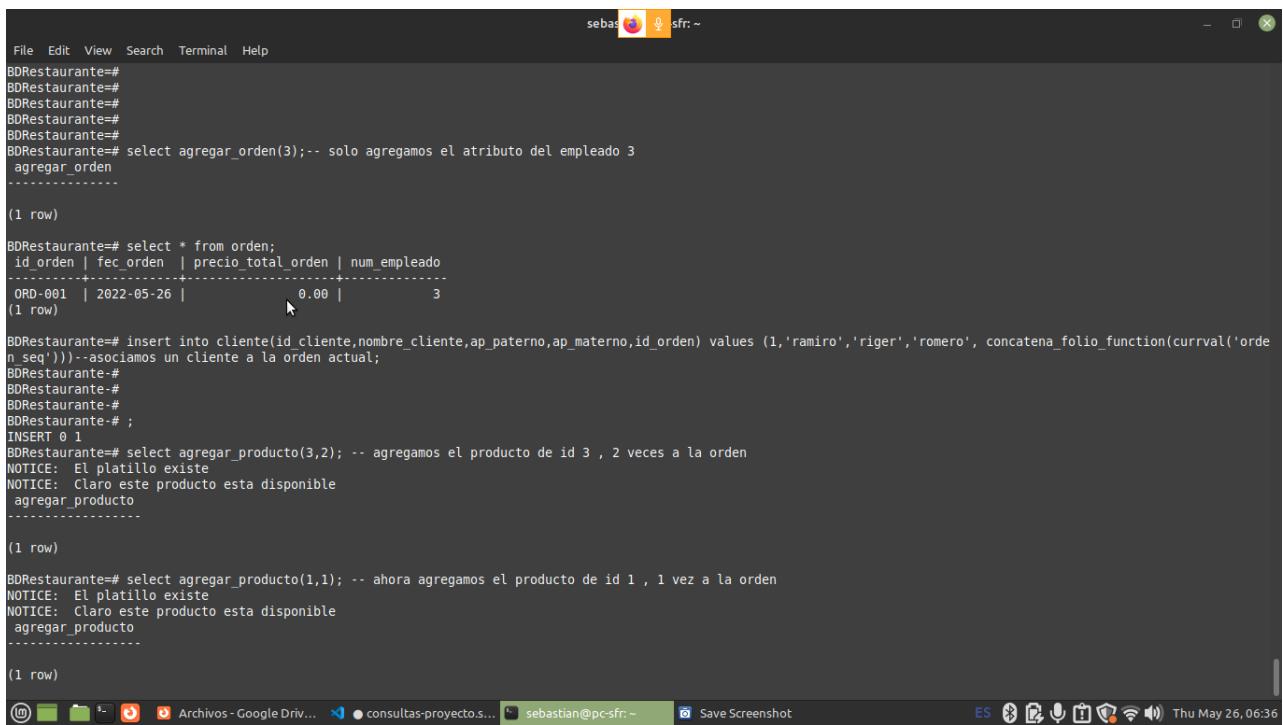
Figura 51: Funcion para agregar un producto y actualiza el precio total de la orden

```
--(select cantidad_vendido from platilloybebida pb where (select
id_platilloybebida from corresponde c ) = id_pb ) + cantidadb;
else
raise exception 'El platillo no está disponible';
end if;
-- se actualiza de manera correcta el precio total de la orden con el
ingreso de cada producto.
update orden set precio_total_orden = ((select
sum(precio_total_platillo) from corresponde c where c.id_orden =
orden_new)+ (select sum(precio_total_bebida)
from corresponde c where c.id_orden = orden_new))where id_orden =
orden_new;
else
raise exception 'El id del producto no existe en el menú o no se
encuentra disponible';
end if;
end;
$$
LANGUAGE plpgsql;
```

Figura 52: Funcion para agregar un producto y actualiza el precio total de la orden

## IMPLEMENTACIÓN EN POSTGRES

Se muestra el cumplimiento del punto en postgres-sql



```

File Edit View Search Terminal Help
BDRestaurante=>
BDRestaurante=>
BDRestaurante=>
BDRestaurante=>
BDRestaurante=>
BDRestaurante=> select agregar_orden(3);-- solo agregamos el atributo del empleado 3
agregar_orden
-----
(1 row)

BDRestaurante=> select * from orden;
+-----+-----+-----+
| id_orden | fec_orden | precio_total_orden | num_empleado |
+-----+-----+-----+
| ORD-001 | 2022-05-26 | 0.00 | 3 |
+-----+-----+-----+
(1 row)

BDRestaurante=> insert into cliente(id_cliente,nombre_cliente,ap_paterno,ap_materno,id_orden) values (1,'ramiro','riger','romero', concatena_folio_function(curval('orden_seq')))--asociamos un cliente a la orden actual;
BDRestaurante=>
BDRestaurante=>
BDRestaurante=> ;
INSERT 0 1
BDRestaurante=> select agregar_producto(3,2); -- agregamos el producto de id 3 , 2 veces a la orden
NOTICE: El platillo existe
NOTICE: Claro este producto esta disponible
agregar_producto
-----
(1 row)

BDRestaurante=> select agregar_producto(1,1); -- ahora agregamos el producto de id 1 , 1 vez a la orden
NOTICE: El platillo existe
NOTICE: Claro este producto esta disponible
agregar_producto
-----
(1 row)

```

The terminal window shows the MySQL command-line interface. The user is connected to a database named 'BDRestaurante'. They run several commands to add an order (with ID 3, date 2022-05-26, total price 0.00, and employee ID 3), insert a client record (ID 1, name 'ramiro', middle names 'riger' and 'romero', and associate it with the current order), and add two items (ID 3, quantity 2) and one item (ID 1, quantity 1) to the order. The output includes notices about existing products and their availability.

Figura 53: Se ingresa una orden y se genera automaticamente con el folio correspondiente



```

File Edit View Search Terminal Help
BDRestaurante=>
BDRestaurante=>
BDRestaurante=>
BDRestaurante=>
BDRestaurante=> select * from orden;
+-----+-----+-----+
| id_orden | fec_orden | precio_total_orden | num_empleado |
+-----+-----+-----+
| ORD-001 | 2022-05-26 | 0.00 | 3 |
+-----+-----+-----+
(1 row)

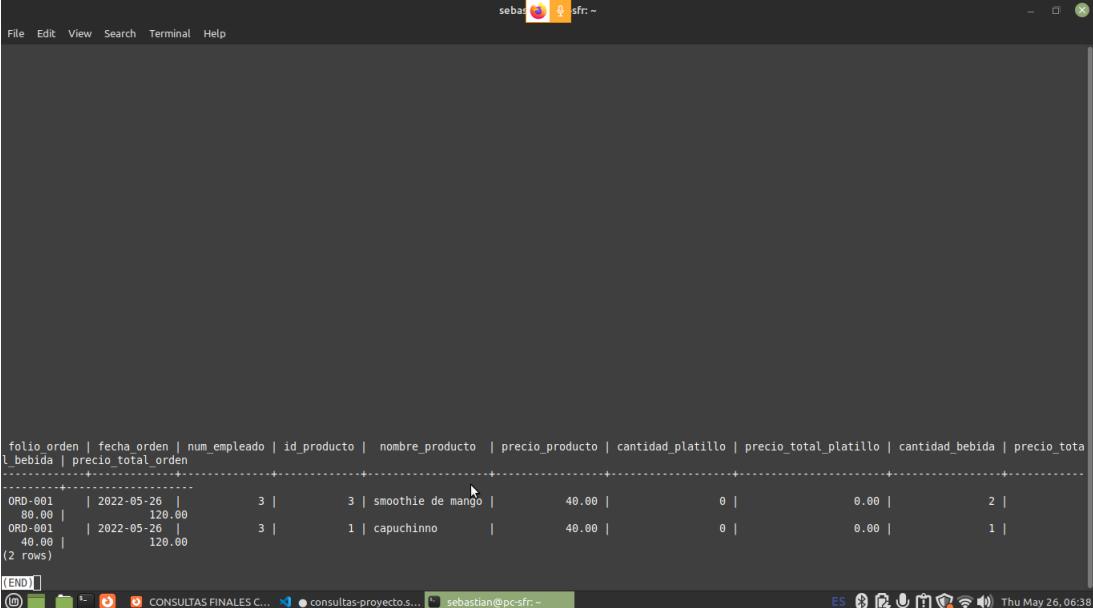
BDRestaurante=> insert into cliente(id_cliente,nombre_cliente,ap_paterno,ap_materno,id_orden) values (1,'ramiro','riger','romero', concatena_folio_function(curval('orden_seq')))--asociamos un cliente a la orden actual;
BDRestaurante=>
BDRestaurante=>
BDRestaurante=> ;
INSERT 0 1
BDRestaurante=> select agregar_producto(3,2); -- agregamos el producto de id 3 , 2 veces a la orden
NOTICE: El platillo existe
NOTICE: Claro este producto esta disponible
agregar_producto
-----
(1 row)

BDRestaurante=> select agregar_producto(1,1); -- ahora agregamos el producto de id 1 , 1 vez a la orden
NOTICE: El platillo existe
NOTICE: Claro este producto esta disponible
agregar_producto
-----
(1 row)

```

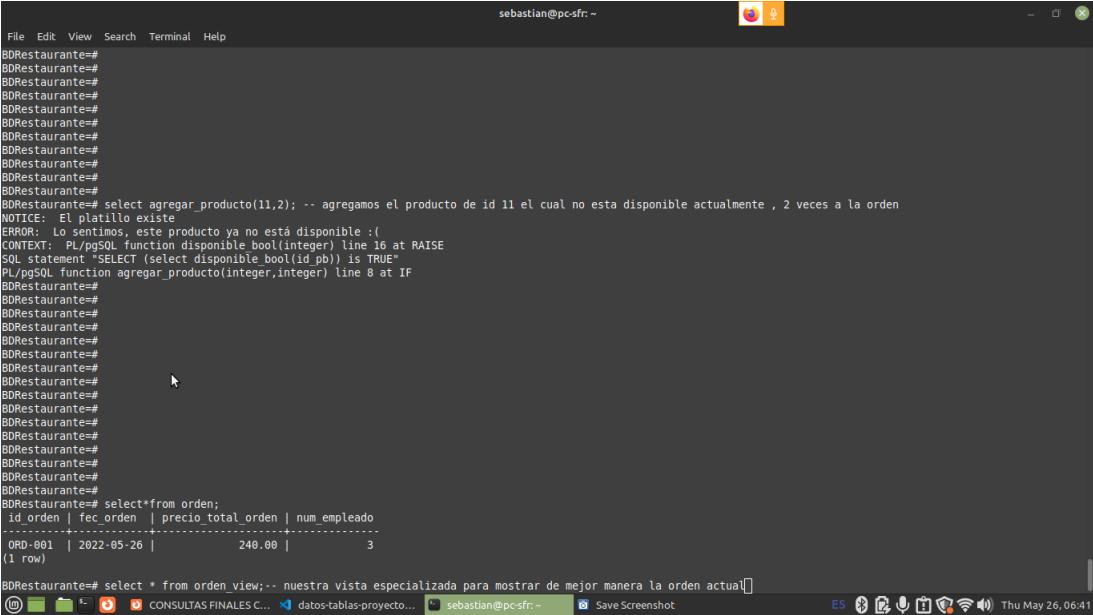
The terminal window shows the MySQL command-line interface. The user runs the same sequence of commands as in Figure 53: inserting an order, associating a client with it, and adding items to the order. The output shows the addition of two items (ID 3, quantity 2) and one item (ID 1, quantity 1) to the order, with notices about existing products and their availability.

Figura 54: Se agrega otro producto y se llama la vista orden



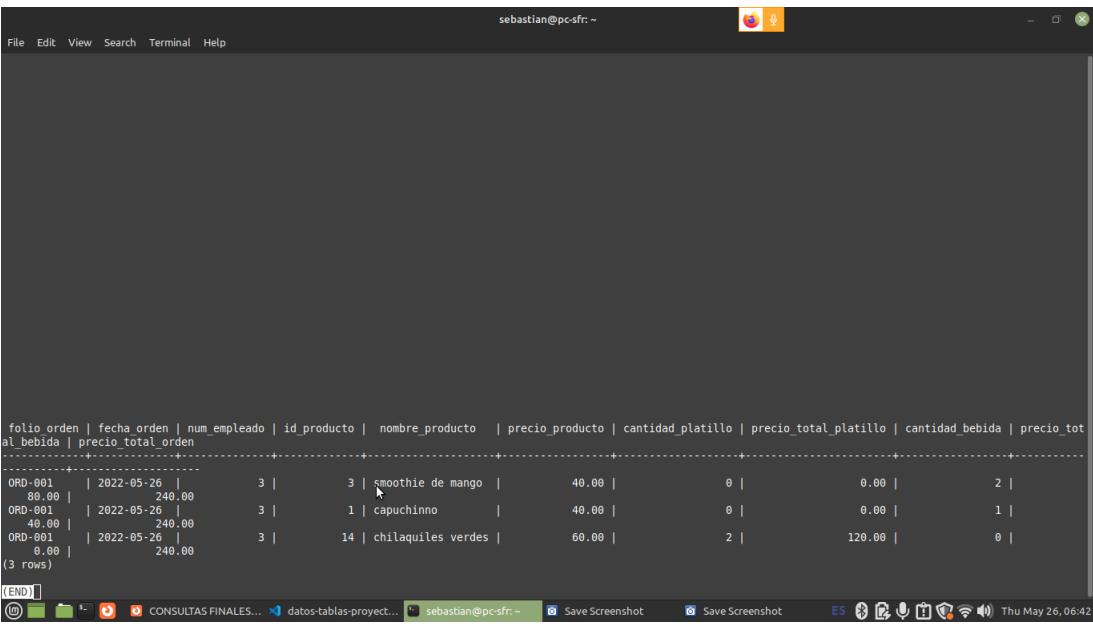
folio\_orden | fecha\_orden | num\_empleado | id\_producto | nombre\_producto | precio\_producto | cantidad\_platillo | precio\_total\_platillo | cantidad\_bebida | precio\_total\_bebida | precio\_total\_orden  
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----  
ORD-001 | 2022-05-26 | 3 | 3 | smoothie de mango | 40.00 | 0 | 0.00 | 2 | 0.00 | 2 |  
ORD-001 | 2022-05-26 | 3 | 1 | capuchinno | 40.00 | 0 | 0.00 | 1 | 0.00 | 1 |  
(2 rows)  
(END)

Figura 55: Se muestra la orden actual



```
sebastian@pc-sfr: ~  
File Edit View Search Terminal Help  
BDRestaurante=#  
BDRestaurante= select agregar_producto(11,2); -- agregamos el producto de id 11 el cual no esta disponible actualmente , 2 veces a la orden  
NOTICE: El platillo existe  
ERROR: Lo sentimos, este producto ya no est&aacute; disponible :(  
CONTEXT: PL/pgSQL function disponible_bool(integer) line 16 at RAISE  
SQL statement "SELECT (select disponible_bool(id_pbl)) is TRUE"  
PL/pgSQL function agregar_producto(integer,integer) line 8 at IF  
BDRestaurante=#  
BDRestaurante= select*from orden;  
-----+-----+-----+-----+-----  
ORD-001 | 2022-05-26 | 240.00 | 3  
(1 row)  
BDRestaurante=# select * from orden_view;-- muestra vista especializada para mostrar de mejor manera la orden actual
```

Figura 56: Se verifica la disponibilidad



The screenshot shows a terminal window with a dark background. At the top, it displays the user 'sebastian@pc-sfr: ~' and icons for a browser and a microphone. Below the header, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main area of the terminal shows a table output from a MySQL query. The columns are labeled: 'folio orden', 'fecha orden', 'num\_empleado', 'id\_producto', 'nombre\_producto', 'precio\_producto', 'cantidad\_platillo', 'precio\_total\_platillo', 'cantidad\_bebida', and 'precio\_total\_bebida'. The data shows three rows of orders:

| folio orden | fecha orden | num_empleado | id_producto | nombre_producto    | precio_producto | cantidad_platillo | precio_total_platillo | cantidad_bebida | precio_total_bebida |
|-------------|-------------|--------------|-------------|--------------------|-----------------|-------------------|-----------------------|-----------------|---------------------|
| ORD-001     | 2022-05-26  | 3            | 3           | smoothie de mango  | 40.00           | 0                 | 0.00                  | 2               |                     |
| ORD-001     | 2022-05-26  | 3            | 1           | capuchinno         | 40.00           | 0                 | 0.00                  | 1               |                     |
| ORD-001     | 2022-05-26  | 3            | 14          | chilaquiles verdes | 60.00           | 2                 | 120.00                | 0               |                     |

Below the table, it says '(3 rows)'. At the bottom of the terminal window, there are several icons: a file, a terminal, a browser, a database, a file manager, and a terminal. The status bar at the bottom right shows the date and time: 'Thu May 26, 06:42'.

Figura 57: Se muestra la orden actual y se verifica que no se agrego al total de la orden el producto 11

#### 4.9.2. Crear al menos, un índice, del tipo que se prefiera y donde se prefiere. Justificar el porqué de la elección en ambos aspectos.

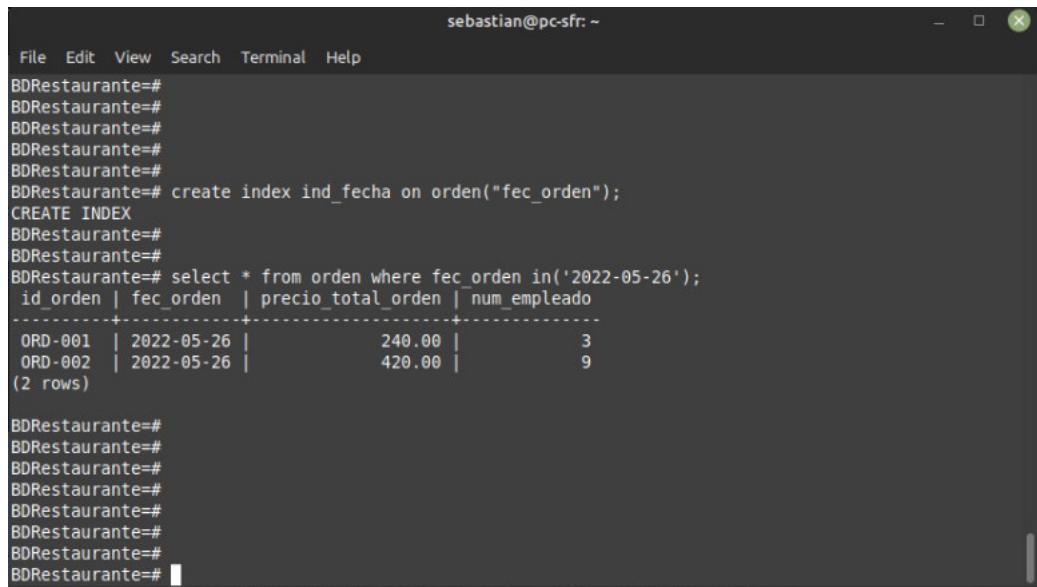
Se crea un índice haciendo referencia a la fecha de la orden.

Se eligió este índice debido a que existen muchas implementaciones reales en donde se emplean este tipo de índices para tener un registro de los productos que se están vendiendo y de esta manera tener un control sobre los productos y las ganancias que estos generan. Otro aspecto importante para elegir este índice es que se pueden observar todos los detalles de las órdenes que se hicieron en un día en específico algo conocido como corte en las empresas.

```
create index ind_fecha on orden("fec_orden");
```

Figura 58: Creacion del indice.

## IMPLEMENTACIÓN



The screenshot shows a terminal window titled "sebastian@pc-sfr: ~". The window contains the following text:

```
File Edit View Search Terminal Help
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
CREATE INDEX
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
select * from orden where fec_orden in('2022-05-26');
id_orden | fec_orden | precio_total_orden | num_empleado
-----+-----+-----+
ORD-001 | 2022-05-26 | 240.00 | 3
ORD-002 | 2022-05-26 | 420.00 | 9
(2 rows)

BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#
BDRestaurante=&#</pre></div><div data-bbox="285 540 669 559" data-label="Caption">Figura 59: Uso del indice en el manejador</div><div data-bbox="101 625 857 725" data-label="Text"><p><b>4.9.3. Dado un número de empleado, mostrar la cantidad de órdenes que ha registrado en el día así como el total que se ha pagado por dichas órdenes. Si no se trata de un mesero, mostrar un mensaje de error.</b></p></div><div data-bbox="101 750 857 907" data-label="Text"><p>Hablando de las órdenes generadas por día empleamos una función que recibe como parámetro un id de mesero y nos permite observar la cantidad de órdenes que capturó dicho mesero en específico y en base a eso calcular el total por esas órdenes. Para esto primero se verifica que el id ingresado sea de un mesero, si no lo es se muestra el mensaje</p></div>
```

“No es un mesero”, si si lo es entonces se hace un conteo con la función count() de todas las órdenes que tengan asociado el id del mesero ingresado y sumamos el total de las órdenes que cumplieron con la condición anterior con la función sum, además de la condición de que las órdenes deben estar asociadas al id ingresado también tenemos que verificar que la fecha en la que se capturó la orden sea la misma a la fecha actual (current\_date). Si ambas condiciones se cumplen entonces se muestra el número de órdenes y el total de estas.

```
create or replace function total_dia( in num_emp int, out
Cantidad_Ordenes int, out Total_Ordenes int)
language plpgsql
as $$
begin
    if tipo_empleado.tipo_mesero='f' from tipo_empleado where
    tipo_empleado.num_empleado=num_emp then
        raise exception 'No es un mesero';
    end if;
    select count(*), sum(precio_total_orden)
    into Cantidad_Ordenes, Total_Ordenes
    from orden
    where num_empleado = num_emp
    and
    fec_orden = current_date;
end; $$;
```

Figura 60: Funcion total<sub>dia</sub>

## IMPLEMENTACIÓN

Observamos todas las órdenes y el tipo de empleado. Probaremos con el empleado que tenga el número de empleado 3 ya que es mesero y tiene órdenes registradas, es importante mencionar que el día actual es (2022-05-25). Nuestro resultado esperado es que este día ha registrado dos órdenes con un total de \$85.

```
postgres=# select * from tipo_empleado;
 tipo_cocinero | tipo_administrativo | tipo_mesero | num_empleado
-----+-----+-----+-----
 f          | t          | f          | 1
 t          | f          | f          | 2
 f          | f          | t          | 3
 f          | t          | f          | 2
 t          | f          | f          | 4
 t          | f          | f          | 5
 t          | f          | f          | 6
 f          | f          | t          | 7
 t          | f          | f          | 8
 f          | f          | t          | 9
 f          | f          | t          | 10
 f          | f          | t          | 11
 f          | f          | t          | 12
 t          | f          | f          | 13
 t          | f          | f          | 14
 f          | t          | f          | 15
(16 filas)

postgres=# select * from orden;
 id_orden | fec_orden | precio_total_orden | num_empleado
-----+-----+-----+-----
 1 | 2022-05-22 |      20.00 | 1
 2 | 2022-05-22 |      10.00 | 1
 3 | 2022-08-22 |      60.00 | 1
 4 | 2022-08-22 |      60.00 | 2
 5 | 2022-05-25 |      50.00 | 3
 6 | 2022-05-25 |      35.00 | 3
 7 | 2022-05-27 |      30.00 | 3
(7 filas)
```

Figura 61: Se muestran todos los empleados para ver cuales son meseros

```
postgres=# select * from total_dia(3);
 cantidad_ordenes | total_ordenes
-----+-----
 2 | 85
(1 fila)
```

Figura 62: Se obtiene la funcion de este requerimiento

```
postgres=# select * from total_dia(2);
NOTICE:  No es un mesero
 cantidad_ordenes | total_ordenes
-----+-----
 0 | 
(1 fila)
```

Figura 63: Ahora probaremos con el numero de empleado 2 ya que se trata de un cocinero y no de un mesero.

#### 4.9.4. Vista que muestre todos los detalles del platillo más vendido.

Para obtener los detalles del producto más vendido se creó una vista la cual ordena todos los productos en base a la cantidad de vendidos (campo que se actualiza al agregar un producto) de forma descendente, de esta manera tendremos el producto más vendido en la cima de la consulta y utilizamos LIMIT 1 para solo mostrar los detalles de este producto.

```
create view mas_vendido as
select
pb.id_platilloybebida as id_producto, pb.nombre_platilloybebida as
nombre_producto, pb.receta, c.id_categoria, c.nombre_categoria,
c.descripcion_categoria
from platilloybebida pb inner join categoria c on pb.id_categoria
=c.id_categoria
order by pb.cantidad_vendido desc limit 1;
```

Figura 64: Vista que retorna el platillo más vendido por medio de el atributo *cantidad\_vendido*.

## IMPLEMENTACIÓN

Observando los registros de la tabla “corresponde” podemos encontrar que el producto más vendido es el que tiene asignado el id 2, ahora faremos esta evaluación dentro de nuestra base utilizando vistas y consultas recordando que nuestro resultado deben ser los detalles del producto con ID 2.

```
postgres=# select * from mas_vendido;
 id_producto | nombre_producto | receta | id_categoria | nombre_categoria | descripcion_categoria
-----+-----+-----+-----+-----+-----+
 2 | chocolate caliente | chocolate oaxaque|o espumoso | 1 | bebidas calientes | 100% mexicanos
(1 fila)
```

Figura 65: Vista del platillo más vendido

#### 4.9.5. Permitir obtener el nombre de aquellos productos que no estén disponibles.

Se implementa una vista para mostrar los nombres de aquellos productos que no estén disponibles. Para esto simplemente se verifica que el campo ‘disponibilidad’ de la tabla “platilloybebida” sea falso lo que nos indica que el producto en cuestión no tiene stock.

```
create view no_disponible as
select nombre_platilloybebida as nombre_producto from platilloybebida
where disponibilidad = FALSE;
```

Figura 66: Vista de productos no disponibles

## IMPLEMENTACIÓN

Los registros mostrados en la imagen nos indican los productos y sus características, de ahí podemos encontrar que los únicos que no están disponibles son los tarascos y el norteño. Ahora verificaremos esta información mediante programación en la BD.

```

postgres=# select * from orden;
 id_orden | fec_orden | precio_total_orden | num_empleado
-----+-----+-----+-----
 ORD-001 | 2022-05-26 | 240.00 | 3
 ORD-002 | 2022-05-26 | 100.00 | 7
 ORD-003 | 2022-05-26 | 650.00 | 10
 ORD-004 | 2022-05-26 | 130.00 | 7
 ORD-005 | 2022-05-26 | 120.00 | 12
 ORD-006 | 2022-05-26 | 480.00 | 3
(6 filas)

postgres=# select * from corresponde;
 id_platilloybebida | id_orden | cant_platillo | cant_bebida | precio_total_bebida | precio_total_platillo
-----+-----+-----+-----+-----+-----
 3 | ORD-001 | 0 | 4 | 160.00 | 0.00
 1 | ORD-001 | 0 | 2 | 80.00 | 0.00
 9 | ORD-002 | 1 | 0 | 0.00 | 100.00
 2 | ORD-003 | 0 | 10 | 500.00 | 0.00
 6 | ORD-003 | 0 | 2 | 60.00 | 0.00
 5 | ORD-003 | 0 | 3 | 90.00 | 0.00
 5 | ORD-004 | 0 | 3 | 90.00 | 0.00
 3 | ORD-004 | 0 | 1 | 40.00 | 0.00
 1 | ORD-005 | 0 | 3 | 120.00 | 0.00
 7 | ORD-006 | 8 | 0 | 0.00 | 480.00
(10 filas)

postgres=# select * from cliente;
 id_cliente | nombre_cliente | ap_paterno | ap_materno | id_orden
-----+-----+-----+-----+-----
 1 | Jose | Lopez | Mora | ORD-001
 2 | Nicole | Sanchez | Ortega | ORD-002
 3 | Juan | Marquez | Olivas | ORD-003
 4 | Mario | Morales | Medina | ORD-004
 5 | Emmanuel | Martinez | Gutierrez | ORD-005
 6 | Katya | Navarrete | Gomez | ORD-006
(6 filas)

```

Figura 67: Se muestran todos los platillos para identificar cuales no estaban disponibles

```

postgres=# select * from no_disponible;
 nombre_producto
-----
 tarascos
 norte
(2 filas)

```

Figura 68: Se muestran los no disponibles

#### 4.9.6. De manera automatica se genere una vista que contenga informacion necesaria para asemejarse a una factura de una orden.

Una vez concluida la orden se genera la factura (si el cliente así lo solicita). Por ello se crea la vista “facturaaa” que basicamente es una conexion entre las tablas “orden”, “platilloybebida”, “factura” y

“corresponde” mediante INNER JOIN

```
create view facturaaa as
select orden.id_orden as folio_orden, orden.precio_total_orden,
cliente.nombre_cliente, cliente.ap_paterno, cliente.ap_materno ,
factura.rfc, factura.calle, factura.colonia, factura.estado,
factura.email, factura.fec_nacimiento,
platilloybebida.id_platilloybebida,
platilloybebida.nombre_platilloybebida, corresponde.cant_platillo,
corresponde.cant_bebida,platilloybebida.precio_platilloybebida
as precio_unitario, corresponde.precio_total_platillo,
corresponde.precio_total_bebida
from platilloybebida inner join corresponde on
platilloybebida.id_platilloybebida=corresponde.id_platilloybebida
inner join orden on corresponde.id_orden = orden.id_orden inner join
cliente on orden.id_orden=cliente.id_orden
inner join factura on factura.id_cliente=cliente.id_cliente where
orden.id_orden = concatena_folio_function(curval('orden_seq'));
```

Figura 69: Se muestra la vista que genera los datos de la factura

## IMPLEMENTACIÓN

```
psql:sql> select * from facturaaa;
 folio_orden | precio_total_orden | nombre_cliente | ap_paterno | ap_materno | rfc      | calle    | colonia   | estado   | email     | fec_nacimiento | id_platilloybebida | nombre_platilloybebida
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
ORD-006 |          480.00 | Katya       | Gomez      | Gomez     | dewedwed23d | sur 24  | Jardines  | COMX    | exwced@xde | 2000-08-23  |           7 | hot cakes
(1 fila)
```

Figura 70: Se invoca la vista desde el manejador

**4.9.7. Dada una fecha, o una fecha de inicio y fecha de fin, regresar el total del número de ventas y el monto total por las ventas en ese periodo de tiempo. Nota: Con producto se hace referencia a los alimentos y bebidas.**

Para regresar el total del número de ventas y el monto total por las ventas en una fecha o rango de fechas se creó la función ‘ventas\_fecha’ y como se reciben distintos parámetros dependiendo si solo se desea una fecha o un rango de fechas se tuvo que sobrecargar dicha función, recibiendo así un solo parámetro que es la fecha en específico o recibir

dos parámetros que hacen referencia a la fecha de inicio y a la fecha de fin del rango seleccionado.

Si se trata de una sola fecha hacemos el conteo y la suma del total por las órdenes registradas ese día siempre, si se trata de un rango de fechas al igual se realiza el conteo y la suma del total por las órdenes que se encuentren entre el parámetro 1 y el parámetro 2, para ello utilizamos la sentencia BETWEEN que nos permite encontrar los registros en un rango.

```
--funcion con un solo parametro de entrada
create or replace function ventas_fecha(in inicio date, out
Numero_Ventas int, out Monto_Total int)
language plpgsql
as $$
begin
    select count (*), sum(precio_total_orden)
    into Numero_Ventas, Monto_Total
    from orden
    where fec_orden=inicio;
end; $$;

--funcion con dos parametros de entrada sobre carga y regresa entre
un rango de fechas
create or replace function ventas_fecha(in inicio date, in fin date,
out Numero_Ventas int, out Monto_Total int)
language plpgsql
as $$
begin
    select count (*), sum(precio_total_orden)
    into Numero_Ventas, Monto_Total
    from orden
    where fec_orden between inicio and fin;
end; $$;
```

Figura 71: Funciones para mostrar el número de ventas de una fecha o de un rango de fechas (sobre carga de funciones)

## IMPLEMENTACIÓN

Se pasan como parámetros las fechas a las funciones y como está

sobrecargada podemos pasarle una o dos fechas y esta sabra que ha-cer.

```
postgres=# select * from orden;
 id_orden | fec_orden | precio_total_orden | num_empleado
-----+-----+-----+-----+
 ORD-001 | 2022-05-26 |      240.00 |      3
 ORD-002 | 2022-05-26 |      100.00 |      7
 ORD-003 | 2022-05-26 |      650.00 |     10
 ORD-004 | 2022-05-26 |      130.00 |      7
 ORD-005 | 2022-05-26 |      120.00 |     12
 ORD-006 | 2022-05-26 |      480.00 |      3
 ORD-007 | 2019-11-12 |      200.00 |      3
 ORD-008 | 2020-10-12 |      200.00 |      7
 ORD-009 | 2022-05-27 |      200.00 |      7
(9 filas)

postgres=# select * from ventas_fecha('2022-05-26');
 numero_ventas | monto_total
-----+-----+
 6 |      1720
(1 fila)

postgres=# select * from ventas_fecha('2019-10-10','2022-05-26');
 numero_ventas | monto_total
-----+-----+
 8 |      2120
(1 fila)
```

Figura 72: Dado una serie de ordenes se muestra el numero de ventas y el monto total

#### 4.9.8. Tablas finales de orden, corresponde y cliente

Para finalizar con la implementación del proyecto se muestra como es que terminan llenas las tablas de orden, corresponde y cliente, las cuales son las que se llenan por medio de todas las funciones y reque-rimientos que se mencionaron, mostrando la buena ejecución de codigo en postgres-sql.

```

postgres=# select * from orden;
 id_orden | fec_orden | precio_total_orden | num_empleado
-----+-----+-----+-----
 ORD-001 | 2022-05-26 | 240.00 | 3
 ORD-002 | 2022-05-26 | 100.00 | 7
 ORD-003 | 2022-05-26 | 650.00 | 10
 ORD-004 | 2022-05-26 | 130.00 | 7
 ORD-005 | 2022-05-26 | 120.00 | 12
 ORD-006 | 2022-05-26 | 480.00 | 3
(6 filas)

postgres=# select * from corresponde;
 id_platilloybebida | id_orden | cant_platillo | cant_bebida | precio_total_bebida | precio_total_platillo
-----+-----+-----+-----+-----+-----
 3 | ORD-001 | 0 | 4 | 160.00 | 0.00
 1 | ORD-001 | 0 | 2 | 80.00 | 0.00
 9 | ORD-002 | 1 | 0 | 0.00 | 100.00
 2 | ORD-003 | 0 | 10 | 500.00 | 0.00
 6 | ORD-003 | 0 | 2 | 60.00 | 0.00
 5 | ORD-003 | 0 | 3 | 90.00 | 0.00
 5 | ORD-004 | 0 | 3 | 90.00 | 0.00
 3 | ORD-004 | 0 | 1 | 40.00 | 0.00
 1 | ORD-005 | 0 | 3 | 120.00 | 0.00
 7 | ORD-006 | 8 | 0 | 0.00 | 480.00
(10 filas)

postgres=# select * from cliente;
 id_cliente | nombre_cliente | ap_paterno | ap_materno | id_orden
-----+-----+-----+-----+-----
 1 | Jose | Lopez | Mora | ORD-001
 2 | Nicole | Sanchez | Ortega | ORD-002
 3 | Juan | Marquez | Olivas | ORD-003
 4 | Mario | Morales | Medina | ORD-004
 5 | Emmanuel | Martinez | Gutierrez | ORD-005
 6 | Katya | Navarrete | Gomez | ORD-006
(6 filas)

```

Figura 73: Dado una serie de ordenes se muestra el numero de ventas y el monto total

## PRESENTACIÓN

### 5.1. POSTGRES - SQL

El uso de POSTGRES-SQL fue de gran ayuda para este proyecto, todas las manipulaciones internas fueron realizadas con postgres-sql, y la gran parte de información teórica se vio explicada en puntos anteriores en donde se explicaron todas y cada una de las funciones que se implementaron en este documento.

## 5.2. PYTHON

La interfaz se realizó por medio de python, debido a su gran facilidad para generar interfaces gráficas y su alta compatibilidad con estas mismas, se muestra a continuación capturas de nuestra interfaz

## 5.3. CONEXIÓN

Para lograr la conexión de nuestra base de datos con la interfaz realizada en python fue necesario la importación de la biblioteca fundamental psycopg2, el cual es un adaptador de base de datos con postgres.

Inicialmente fue necesario instalar el paquete en linea de comando desde la terminal con la instrucción:

<pip install psycopg2>

Posteriormente en el desarrollo del código se utilizó la función connect( ) del módulo psycopg2, especificando los parámetros de nuestra base de datos en postgres con variables previamente definidas, creando una lista de argumentos como se muestra en la siguiente figura, donde:

- host: Es la dirección del servidor de la base de datos, por ejemplo, localhost o una dirección IP. En nuestro caso localhost

- port: Es el número de puerto que por defecto es 5432 si no se proporciona.
- user: Es el nombre de usuario utilizado para la autenticación
- password: Es la contraseña utilizada para la autenticación.
- database: Es el nombre de la base de datos que deseamos conectar.

Se hizo uso de la función cursor() la cual permite que el código de python ejecute comandos PostgreSQL durante la sesión de la base de datos. Todo esto implementado con excepciones en dado caso en el que por algún error la base de datos no se pueda conectar satisfactoriamente.

```
#-----#
#      Disconnection to database
#-----
try:
    if connection:
        cursor.close()
        connection.close()
    print("Conexión a la base de datos restaurante FINALIZADA")
except Exception as ex:
    print(ex)
```

Figura 74: Forma de conectarse

```
#-----#
#      Connection to database
#-----
try:
    connection = psycopg2.connect(user=PSQL_USER,
                                    password=PSQL_PASS,
                                    host=PSQL_HOST,
                                    port=PSQL_PORT,
                                    database=PSL_BD)
    cursor = connection.cursor()
    print("Conexión a la base de datos restaurante EXITOSA")
except Exception as ex:
    print(ex)
```

Figura 75: Forma de conectarse en python

## 5.4. INTERFAZ

### 5.4.1. PANTALLA DE INICIO

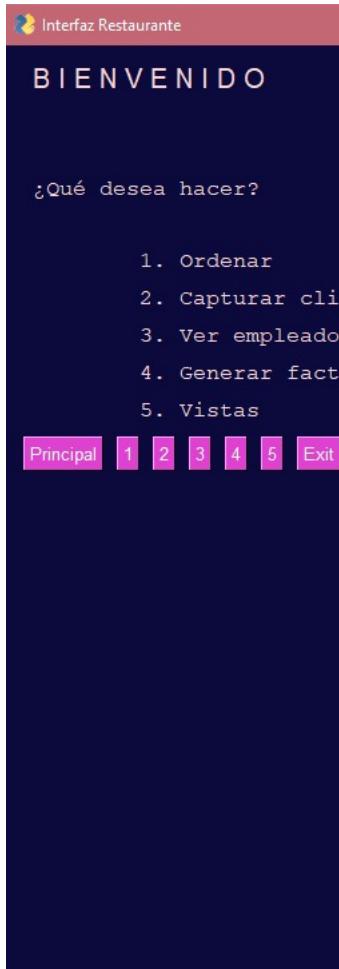


Figura 76: Interfaz de inicio

Tenemos la pantalla de inicio la cual es de gran utilidad para mostrar la vista principal de nuestra interfaz.

### 5.4.2. MENÚ

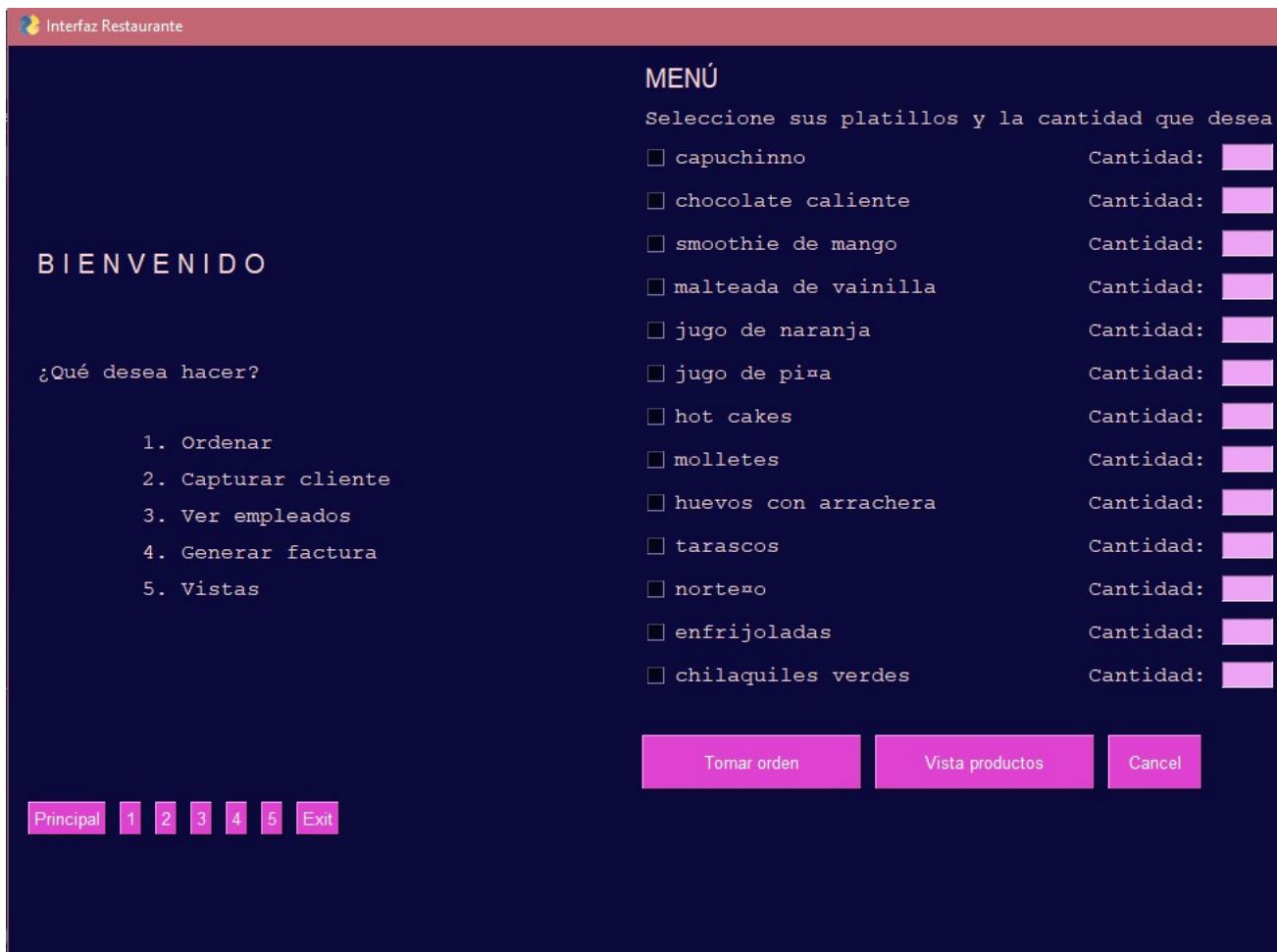


Figura 77: Interfaz del menú

Mostrando el menú podemos hacer alusión a cada detalle que se puede realizar o seleccionar dentro de nuestra interfaz gráfica.

### 5.4.3. USO DEL MENÚ

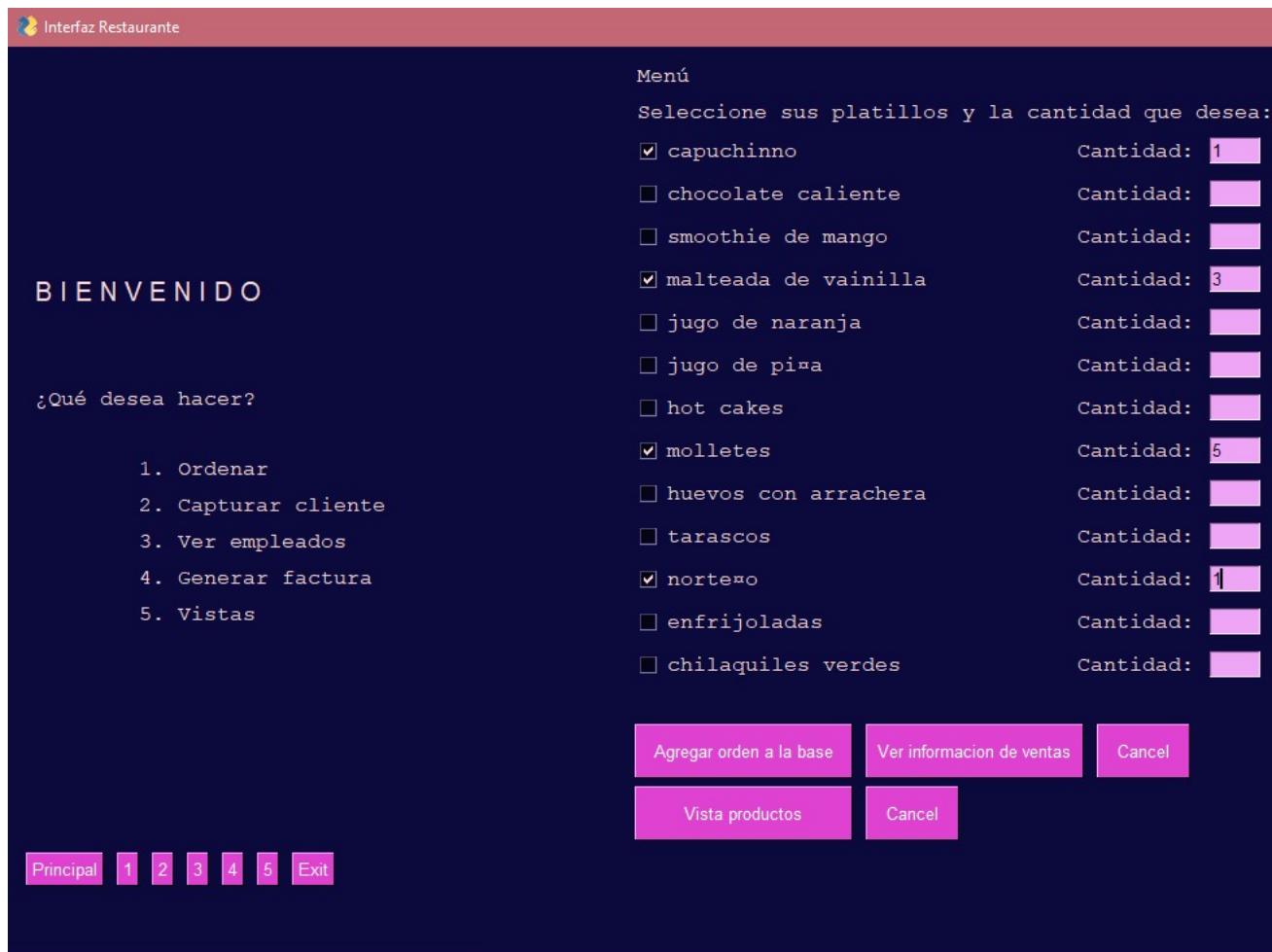


Figura 78: Uso de la interfaz del menú del alimentos

Mostramos una sección de los platillos y bebidas que pueden agregarse a las ordenes de nuestro menú.

A continuación se muestran las interfaces de los clientes, empleados, registros de empleados, como se pide en los requerimientos.

#### 5.4.4. INTERFAZ CLIENTE

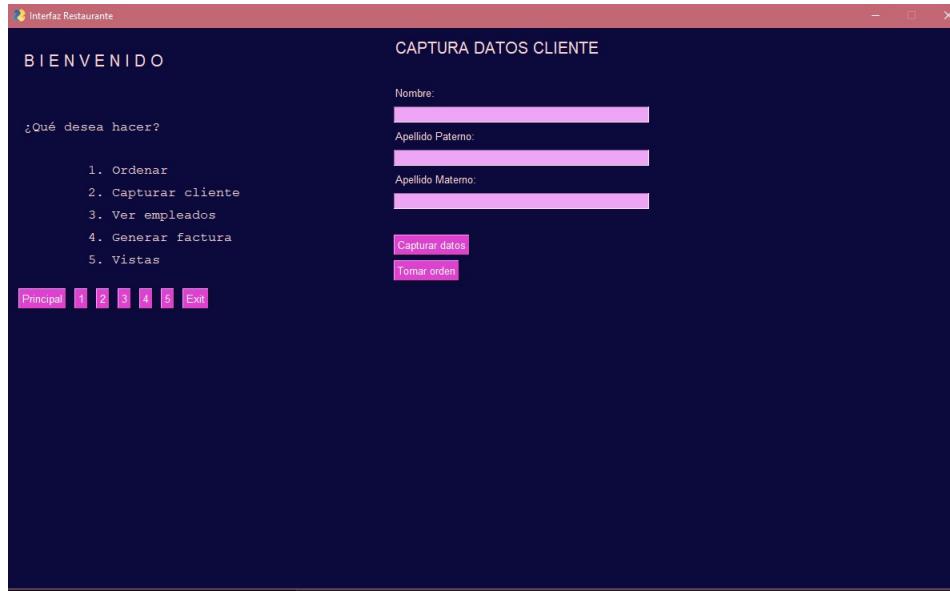


Figura 79: Interfaz de cliente

#### 5.4.5. INTERFAZ EMPLEADO

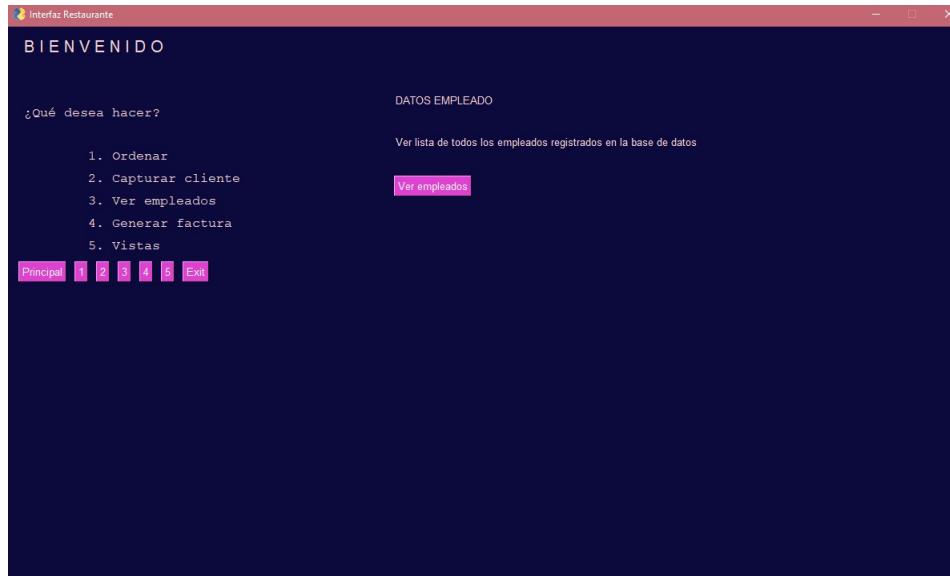


Figura 80: Interfaz de empleados

#### 5.4.6. Menú para vistas

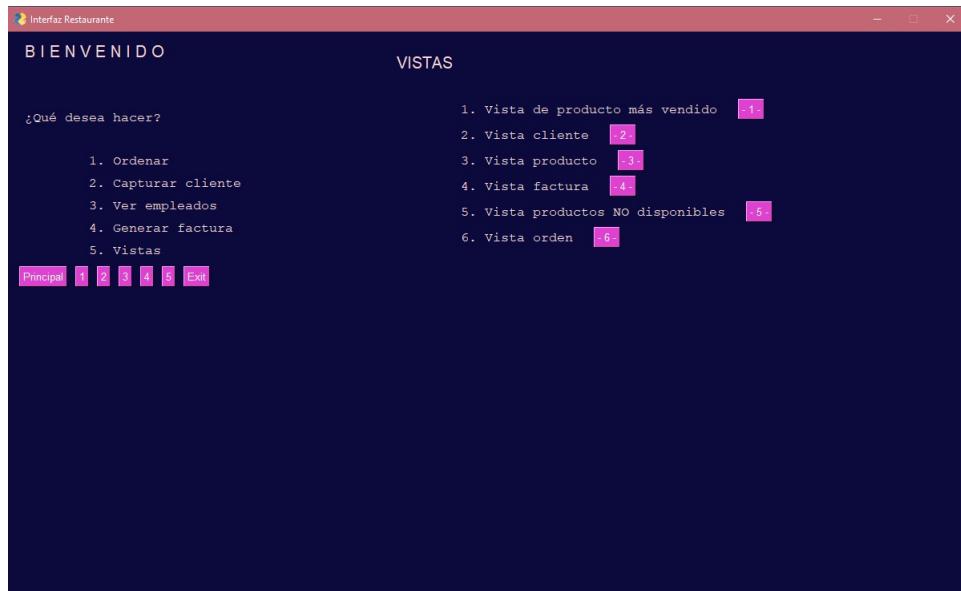


Figura 81: Menú para vistas

#### 5.4.7. INTERFAZ FACTURA

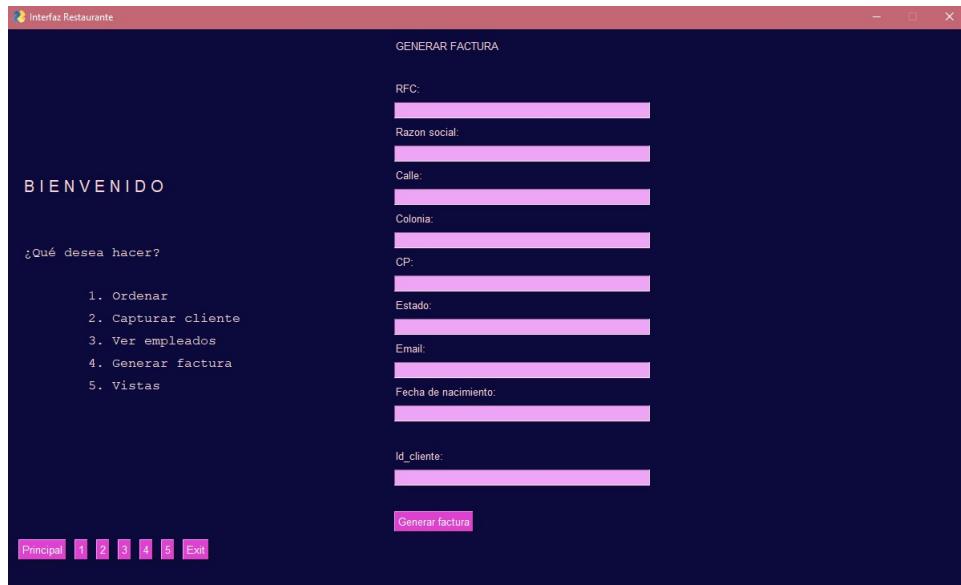


Figura 82: Interfaz de factura

Cuando mostramos la interfaz factura recibimos los datos de factura que se requieren del cliente.

## CONCLUSIONES

### **Corona Nava Pedro Jair**

En La realización del proyecto me pareció desafiante y muy interesante ya que se asemeja mucho a una situación que se nos puede presentar en nuestra vida laboral, abordar la solución desde esta etapa de nuestra formación es muy útil ya que es una forma de poner en práctica y ‘pulir’ nuestros conocimientos por decirlo de alguna manera. Al realizar este tipo de proyectos podemos desarrollar habilidades para encontrar la solución de una forma más simple y eficaz ya que aprendemos a realizar un buen análisis de los requerimientos del problema lo cual nos permite tener un buen planteamiento de los diversos modelos (MER,MR) y con ellos se vuelve más fácil la conexión entre tablas al momento de crear nuestra base de datos debido a que encontramos las conexiones referentes a las llaves primarias, las llaves secundarias, la propagación de estas dependiendo de la relación, etc. Lo cual hace que nuestras tablas tengan una buena estructura y con esto se simplifican los trabajos de consultas y manipulación de datos.

Un punto muy importante en el desarrollo del proyecto fue la manipulación de la información recabada, es decir, la creación de consultas y vistas para obtener un resultado en específico a partir de los campos de una o varias tablas. Otro aspecto que me pareció muy importante fue que el proyecto abarca temas de diversos campos del conocimiento ya que se tenían que desarrollar interfaces gráficas, manejo de BD en postgresql, entre otros, por lo que fue de gran ayuda delegar responsabilidades de acuerdo a las capacidades y conocimientos de cada uno de nosotros. Por todos estos motivos puede afirmar que este proyecto cumplió con los objetivos que se tenían desde el planteamiento del programa, los cuales son implementar la mayor cantidad de temas vistos durante el semestre en nuestra clase de bases de datos tanto teoría como laboratorio para encontrar la solución a un problema en específico.

### **Fernández Rosales Sebastian**

A decir verdad la realización de este proyecto fue de las actividades más retroalimentadoras que he tenido, considero que la mayoría del proyecto fue de gran ayuda para terminar de comprender como funcionan las bases de datos, son pocos los temas que se dejaron a un lado, ya que casi todo lo que vimos durante el curso se terminó implementando aquí. Lo que más me llamó la atención fue hacer que

este tipo de códigos no solo nos sirvan para hacer funciones o distintas actividades, si no, el almacenado de datos que se está haciendo, en algún momento de la carrera se vió la materia de programación orientada a objetos, una materia que su principal utilidad era darnos a entender que son los objetos en el mundo de la programación, lo cual a mi se me hizo fascinante, y en algún momento se nos hizo saber que ese tipo de programación iba a evolucionar a las bases de datos, y hoy me doy cuenta que es totalmente cierto. A mi parecer este proyecto se basó totalmente en el funcionamiento de los objetos con las bases de datos, un claro ejemplo es la relación que se crea entre ellos, una relación invulnerable, como en el ejemplo de los clientes y las ordenes, los clientes no podrían existir sin que se les asigne una orden de por medio, y el hecho de que se quisiera ingresar un cliente a la base sin antes haberle asignado una orden se me hizo algo muy óptimo, y como mencionábamos en equipo, la mayoría de las tablas se fueron enlazando solas y solo fue necesario aplicar todos los conocimientos de postgres-sql para poder hacer los requerimientos de este proyecto. A decir verdad uno de los proyectos más completos que he tenido.

### **Martínez García Gabriela**

El desarrollo del presente proyecto hace una recapitulación de los temas vistos durante el curso tanto en teoría como en laboratorio; en el cual se trabaja en cada una de las fases que corresponden al di-

seño y creación de una base datos, desde su diseño a partir de los requerimientos/reglas de negocio del cliente hasta su levantamiento y conexión a una interfaz que le permita ser más intuitiva y agradable a vista y uso del usuario.

El proceso secuencial fue el análisis de los requerimientos, posterior el diseño del modelo conceptual trabajando con el MER, creando de acuerdo al análisis; las entidades, atributos y relaciones correspondientes de todo el planteamiento. Seguido, el modelo lógico con la creación del MR; en el que se realizan las reglas de Codd en la propagación de atributos con llave principal para mantener relaciones entre las tablas que se conforman y a su vez, identificando los tipos de datos que referencian cada atributo, garantizando con esto la integridad referencial. Finalmente terminando con el modelo físico, utilizando los lenguajes de datos DDL y DML. Algo importante en el proceso de diseño, fue que debido al correcto diseño inicial en el MER, no fue necesario aplicar el proceso de normalización.

El resultado de nuestro trabajo considero fue satisfactorio, se hizo gran retroalimentación de todo el curso, se aplicaron conocimientos extras que permitieron crear un proyecto que sale del esquema de solo cumplir con un proyecto a nivel académico sino que se visiona con un impacto más acertado al desarrollo de proyectos en el campo profesional. Lo cual como futuros profesionistas nos ayuda a nues-

tra formación, sobre todo al buscar mecanismo de trabajo colaborativo, herramientas, aplicaciones y programas de trabajo. Una de las principales dificultades que se presentaron fueron la inserción de una imagen en una base de datos y el desarrollo a nivel de código para la creación de la interfaz. Este último aspecto fue muy debatido en el equipo pues el conocimiento en el desarrollo de la interfaz era muy deficiente en el equipo.

### **Reyes Mendoza Miriam Guadalupe**

Este proyecto nos sirvió para implementar todos los conceptos vistos en clase para la creación de una buena base de datos, desde el modelo entidad - relación, el modelo relacional, la representación intermedia, las creación de tablas e inserción de datos en el manejador hasta la creación de una interfaz amigable con el usuario para que observé cómo está trabajando nuestra base.

Sin duda alguna es un proyecto bastante completo puesto que si abarca todos los temas vistos a lo largo del curso y podemos ver qué en este, se nos proporcionaron las bases adecuadas para una buena creación y sobre todo implementación de una base da datos utilizando varias herramientas útiles para el desarrollo de la misma.

Es interesante ver cómo todos lo temas que abarcó el curso nos per-

mitieron hacer este pequeño sistema que se puede llevar más allá de solo un proyecto escolar y que nos hizo ver también como es el trabajo y la organización del equipo, pues muchas veces llegábamos a tener dudas en los que cada quien estaba desarrollando y ahí entraba todo el equipo para resolver los problemas que se presentaban.

### **Zarate Diaz Sofía Viridiana**

En este proyecto se aprendieron a manejar distintas herramientas para su realización, así como el uso de lenguajes de programación para la realización de la interfaz. Se aplicaron todos los temas vistos en el curso, ya que se tuvo que plantear el problema y encontrar sus respectivas entidades, atributos, restricciones y relaciones que se tenían, también se tuvo que aplicar los modelos vistos en clase los cuales fueron el modelo relacional extendido, así como el modelo relacional esto para poder crear las tablas y sus restricciones. El diseño se fue analizando mejor al momento de examinar los requerimientos y su implementación.

Uno de los retos que se tuvo fue en el desarrollo de la interfaz, ya que no se tenía mucha experiencia sobre la conexión que se hace entre la interfaz y una base de datos, al hacerlo en Python se tuvieron algunas dificultades en la tabla orden y en la tabla cliente ya que se hizo complicado poder insertar los datos. Se dificultó la manera de in-

sertar la foto en la interfaz, ya que en PostgreSQL generaba errores al enviarle la ruta de la foto insertada, además de que fue difícil conseguir información de cómo lograrlo en Python.

Considero que en este proyecto se aprendieron muchos aspectos importantes de la materia, en mi caso me ayudó a comprender mejor la manera en que funcionan las vistas y las consultas. Los temas vistos sirvieron para poder aplicarlos a un problema que puede ser aplicado en un restaurante e incluso se puede adaptar a otros ambientes. La manera en que desarrollamos el proyecto fue óptima, ya que ayudó a que conociéramos el funcionamiento de una base de datos con una interfaz gráfica.

## **REFERENCIAS**

[ 1 ] Psycopg 2.9.3. Psycopg - PostgreSQL database adapter for Python. Recuperado de: <https://www.psycopg.org/docs/>

[ 2 ] PySimpleGUI. Documentation. Recuperado de: <https://pysimplegui.readthedocs.io/en/latest/>