



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Proyecto final

PRESENTA

Hernández Hernández Cristian
Laparra Miranda Sandra
López Zugasti Christian

PROFESOR

Ing. Fernando Arreola Franco

ASIGNATURA

Bases de datos

27 de mayo

Índice

1. Introducción	3
1.1. Descripción del análisis del problema	3
1.2. Objetivos	3
1.3. Propuesta de solución	3
2. Plan de trabajo	4
3. Diseño	6
3.1. Modelo Entidad-Relación	8
3.2. Representación intermedia modelo relacional	9
3.3. Modelo Relacional	13
3.4. Normalización	14
4. Implementación	20
4.1. Creación de tablas (DDL)	20
4.2. Funciones	22
4.3. Vistas	28
5. Presentación	34
6. Conclusiones	39

1. Introducción

En este proyecto final se presentará el desarrollo, el análisis, diseño así como la propuesta de implementación de una base de datos para el almacenamiento de información de un restaurante, por lo que de esta forma se ayudará a la empresa que lo solicite a organizar, controlar y administrar los productos y las ventas. Se realizará la implementación de la interfaz para integrar la base de datos a un entorno gráfico de acuerdo con los requerimientos del proyecto haciendo posible que la interacción entre los clientes y los servicios del restaurante sea la mejor.

1.1. Descripción del análisis del problema

Para llevar a cabo la administración del restaurante, el proyecto realizó un sistema en el que los datos cambien en tiempo real y por otra parte se mantengan consistentes siguiendo ciertas reglas de negocio, por lo que para hacerlo se tomaron en cuenta las siguientes consideraciones:

- Consultar la información general de los empleados, incluyendo su fotografía.
- Ingresar una orden, de hasta 3 productos, los cuales podrán seleccionarse de una lista de opciones, permitir ingresar la cantidad, calcular el costo total de cada artículo y el costo total de toda la orden. Ingresar dicha información en la base de datos, respetando todas las restricciones de integridad.

1.2. Objetivos

Actualmente más empresas están optando por implementar sistemas de gestión de ventas al emplear las tecnologías de información con el propósito de incrementar los clientes así como las ventas, ya que estas herramientas mejoran la administración de los productos y ventas en tiempo real.

En el caso de un restaurante que será implementado en este proyecto, se desarrollará un sistema de restaurante virtual para la gestión de ventas, de tal forma que permita consultar la información general de los empleados e ingresar información en la base de datos como: una orden del cliente, calcular el costo total de cada artículo y el costo total de toda la orden.

Otro de los objetivos de este proyecto es implementar un sistema de restaurante virtual que permita: manejar un registro de empleados y manejar un registro de ventas, de tal forma que sirva de herramienta a la empresa para tomar mejores decisiones a corto y a largo plazo en cuanto a la organización de la misma.

1.3. Propuesta de solución

Para la base de datos a desarrollar comenzaremos con el análisis de la descripción del problema, así como todos sus requisitos incluyendo las reglas de negocio que implica para llegar a su solución. Se

tomará la decisión de realizar la implementación gráfica en una página web y no en una app móvil debido a que se tiene insuficiente conocimiento sobre la creación de apps. Utilizaremos distintos recursos y herramientas que se explicarán a continuación para lograr cumplir con lo requerido.

Para lograrlo primero se crearán borradores del modelo entidad relación para posteriormente ser comparado con cada borrador propuesto. Se evaluarán las similitudes que existen, así como la mejor solución para almacenar los datos. Para que no existan mayores dificultades, el modelo se realizará a través de la herramienta draw.io

Después del modelo entidad relación se creará la creación del modelo relacional. Para lograrlo será necesario describir la representación intermedia donde se incluirán todas las características de los datos. La creación del modelo relacional se realizará con la herramienta ERStudio. Después se realizará el análisis de la normalización de las tablas obtenidas.

Cuando se tenga listo la parte conceptual de la base de datos, la creación de la misma se llevará a cabo en PostgreSQL. Se definirán todos los objetos requeridos para después agregar la parte de manipulación de datos. En cada modificación se realizan varias pruebas en busca de errores y fallos si es que existen al momento que se requiera ingresar o consultar un dato. Con ayuda de la interfaz de pgAdmin4 lograremos realizar esas actividades.

Luego de crear la base de datos, ahora se implementará en una interfaz gráfica. En nuestro caso se implementará mediante una página web, para ello nos apoyaremos de XAMPP CONTROL, para trabajar con un servidor apache para lograr visualizar la página. Dicha página contendrá un menú simple que permitirá al usuario interactuar sin mayor complejidad, en el cual podrá consultar la información general de los empleados, así como el pedido de una orden.

Finalmente se realizarán las pruebas necesarias para verificar que todo esté en orden y además se cumpla con todos los requerimientos y puntos proporcionados en la primera y segunda parte de la descripción del problema. En caso de haber fallas en el desarrollo se tomarán las debidas medidas para dar solución a esos problemas.

2. Plan de trabajo

El plan de trabajo se llevó a cabo de acuerdo a la organización previa que se tuvo para que cada integrante desarrollará las partes del proyecto respectivamente, dichas partes se dividieron de esta manera:

Entorno de trabajo: Antes de comenzar con las actividades necesitamos de tener instaladas o poder acceder a algunas herramientas de trabajo como lo es PostgreSQL, ERStudio, XAMPP Control, Draw.io, Editor de código.

Diseño conceptual de la base de datos: En cual estará constituido por la creación del modelo entidad-relación, representación intermedia relacional y el modelo relacional así como la normalización de las tablas, respetando los requerimientos y reglas de negocio.

Diseño físico de la base de datos: En esta parte se creará la base de datos del restaurante y

dentro de esta base se crearán todos los objetos y se definirá su estructura para permitir almacenar la información correctamente (DDL), además se creará el código para realizar las consultas y la manipulación de los datos (DML).

Creación de la interfaz gráfica: Implementación de una interfaz web que permitirá conectarse con la base de datos para poder acceder a los datos de los empleados e ingresar órdenes.

A continuación se presenta la distribución del trabajo que realizó cada integrante del equipo:

- Hernández Hernández Cristian se encargó del diseño de modelos entidad-relación y modelo relacional. Normalización de la base de datos. Implementación en lenguaje SQL de base de datos y tablas. Participación en agregado de información para realizar pruebas así como en la implementación de consultas y funciones.
- Laparra Miranda Sandra se encargó del diseño de modelos entidad-relación y modelo relacional. Normalización de la base de datos. Implementación en lenguaje SQL de base de datos y tablas. Participación en agregado de información para realizar pruebas así como en la implementación de consultas y funciones.
- López Zugasti Christian se encargó del diseño de modelos entidad-relación y modelo relacional. Normalización de la base de datos. Implementación en lenguaje SQL de base de datos y tablas. Participación en agregado de información para realizar pruebas así como en la implementación de consultas y funciones.
Implementación de la interfaz gráfica en una página web. Adaptación de la interfaz de usuario usando html y php a la base de datos.

3. Diseño

Descripción del problema

El problema se divide en dos partes:

Parte uno:

Consiste en el diseño de una base de datos. Un restaurante desea digitalizar su forma de operación, para ello se desarrollará un sistema informático que constará 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.
- 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.

Parte dos:

Una vez diseñada y lista la base de datos, se debe crear una interfaz gráfica vía app móvil o web, que permita:

- Consultar la información general de los empleados, incluyendo su fotografía.
- Ingresar una orden, de hasta 3 productos, los cuales podrán seleccionarse de una lista de opciones, permitir ingresar la cantidad, calcular el costo total de cada artículo y el costo total de toda la orden. Ingresar dicha información en la base de datos, respetando todas las restricciones de integridad.

Descripción de lo realizado en las correspondientes fases de diseño de las bases de datos, agregando los resultados de cada una de ellas.

Para comenzar el diseño de la base de datos necesitamos obtener una lista de las entidades candidatas así como sus respectivos atributos. Del enunciado obtenemos:

EMPLEADO: RFC, número de empleado, nombre completo(nombre, apellido paterno, apellido materno), fecha de nacimiento, teléfonos, edad, domicilio(calle, número, colonia, estado, código postal), sueldo, y foto.

COCINERO: especialidad.

MESEROS: horario.

ADMINISTRATIVO: rol.

DEPENDIENTE: CURP, nombre completo(nombre, apellido paterno, apellido materno), parentesco.

PRODUCTO: descripción, nombre, receta, precio, disponibilidad y ventas.

CATEGORÍA: nombre y descripción

ORDEN: folio, fecha, monto final.

CLIENTE: RFC, nombre completo(nombre, apellido paterno, apellido materno), domicilio(calle, número, colonia, estado, código postal), razón social, fecha de nacimiento, email.

Otros atributos: Cantidad del platillo/bebida y precio total por cada platillo/bebida.

Con esto podemos realizar la creación del modelo entidad-relación teniendo en consideración las reglas de negocio

3.1. Modelo Entidad-Relación

Para definir al empleado dentro del modelo, consideramos al atributo teléfono como multivaluado y la edad como un atributo calculado. Por otra parte escogimos al número de empleado como clave principal ya que no se puede repetir además de que es un valor de tipo numérico y como llave candidata al RFC puesto que a diferencia del número de empleado, este tiene un valor alfanumérico.

Se realizó una especialización puesto que del atributo empleado surgen otras entidades con una característica en particular. En esta especialización el tipo de relación es traslape-total ya que un empleado debe tener por lo menos un puesto.

Cada empleado puede tener algún dependiente puesto que no puede existir por sí solo la entidad, entonces se vuelve una entidad débil. El tipo de relación entre el empleado y el dependiente es 1:M ya que un empleado puede tener uno o varios dependientes. Cabe mencionar que el discriminante de dependiente es la CURP.

Después hay una relación entre mesero y orden. La función del mesero será la de levantar la orden de algún cliente. El tipo de relación es 1:M por que un mesero puede atender muchas órdenes. En dicha orden, el folio será la clave principal.

Luego de haber definido la orden, un cliente puede generar una o varias órdenes pero una orden no puede estar asociado a más de un cliente, entonces el tipo de relación entre estas dos entidades es 1:M. La entidad cliente tiene como clave principal el atributo RFC.

Como el producto está contenido dentro de una orden, entonces existirá una relación de tipo M:N por que un producto puede estar contenido en varias órdenes y una orden puede contener varios productos. Cabe resaltar que en esta relación existen dos atributos que son el precio total por producto y la cantidad ya que no pueden pertenecer particularmente de ninguna de las dos entidades mencionadas por que si estuvieran como atributos de orden, no se sabría de cual de todos los productos nos estamos refiriendo al igual que si estuvieran en el atributo producto, aquí no supiéramos a cual de todas las órdenes nos estamos refiriendo. El monto total, el precio total por producto y la disponibilidad pueden ser calculados.

Se escogió colocar una clave artificial a la entidad producto puesto que ningún atributo nos asegura que no se pueda repetir.

Por último tenemos a la entidad categoría. Cada producto debe estar dentro de una sola categoría pero una categoría puede contener varios productos, por lo tanto el tipo de relación de la categoría con producto es 1:M. También se decidió colocar una clave principal a la categoría porque ninguno de sus atributos nos asegura que no se pueda repetir.

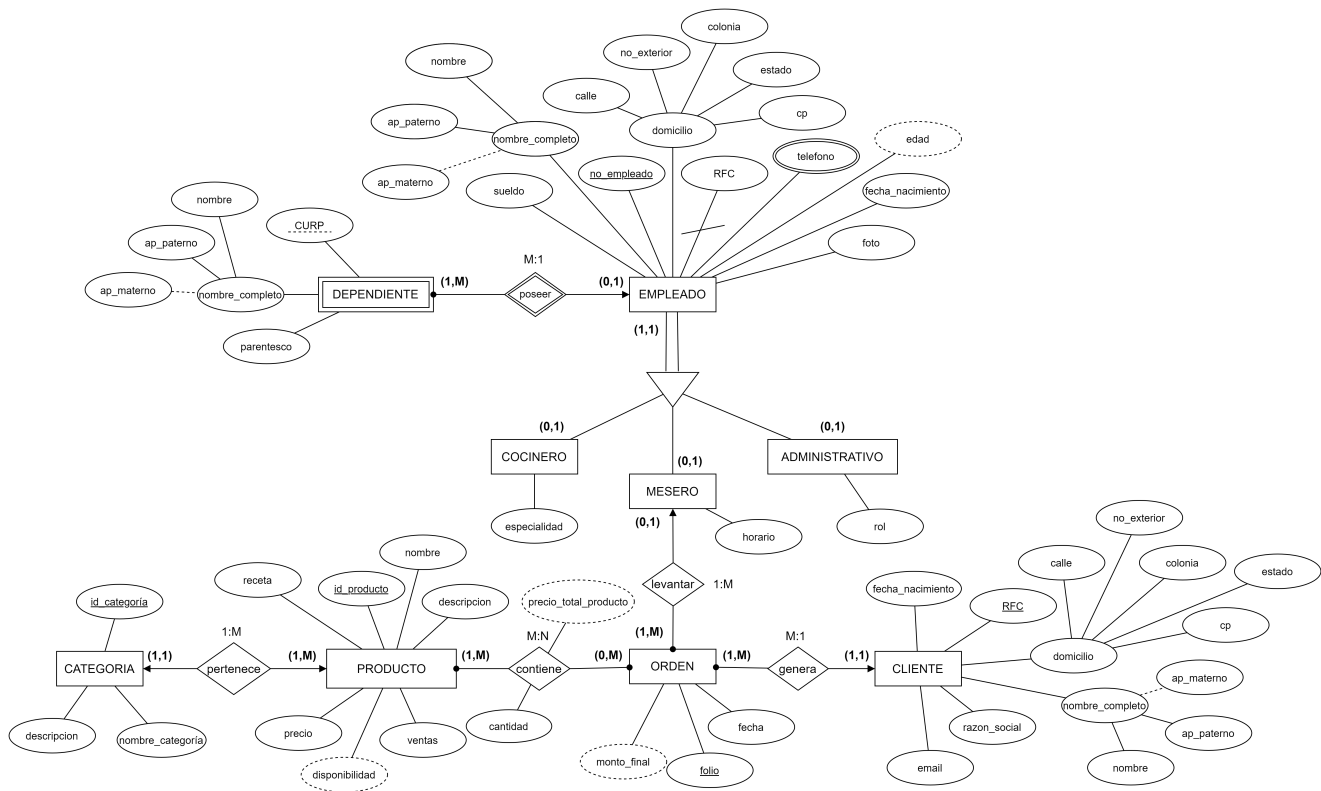


Figura 1: Modelo Entidad-Relación

3.2. Representación intermedia modelo relacional

Antes de crear el modelo relacional, vamos a realizar su representación intermedia en donde incluimos el tipo de dato para cada uno de los atributos, también si son llaves primarias (PK), atributos únicos (U), atributos opcionales (N), si son llaves foráneas (FK) y si son atributos calculados (C).

EMPLEADO: Primero definimos al empleado, tomando en cuenta la estructura del modelo ER. Nos damos cuenta que no existe el atributo teléfono puesto que es un atributo multivaluado y es necesario crear otra tabla que contenga ese atributo.

DEPENDIENTE: Como dependiente es una entidad débil, entonces tendrá una llave primaria compuesta por la CURP del dependiente y el número del empleado al que hace referencia.

ORDEN: La llave primaria de la tabla empleado y la llave primaria del cliente, número de empleado y RFC respectivamente, son mapeados como llaves foráneas en la tabla ORDEN debido al tipo de relación que existen entre las entidades 1:M.

PRODUCTO: Tiene incluida la llave artificial que nosotros propusimos, además solo tiene mapeado la llave primaria de la tabla categoría como llave foránea debido a la relación 1:M.

CATEGORIA: En esta tabla sólo se indico su PK y tipos de datos correspondientes.

CLIENTE: En la tabla cliente también sólo se indicó el tipo de datos así como la definición que

tendrán dentro de la tabla.

TELEFONO: Es una nueva tabla creada a partir del empleado ya que es multivaluado, es decir, pueden ver asociados varios teléfonos a un sólo empleado, por lo tanto se mapea la llave primaria del empleado como foránea en esta tabla.

ORDEN_PRODUCTO: Esta es una tabla creada a partir de la relación M:N que tienen las entidades orden y producto, por lo tanto, la llave primaria estará compuesta por las llaves foráneas correspondientes a las llaves primarias de estas dos entidades.

```

EMPLEADO {
    no_empleado int (PK),
    rfc varchar(13) (U),
    sueldo int,
    foto bytea,
    calle varchar(30),
    no_exterior varchar(9),
    colonia varchar(30),
    estado varchar(20),
    cp int,
    nombre varchar(20),
    ap_paterno varchar(25),
    ap_materno varchar(25) (N),
    edad smallint (C),
    fecha_nacimiento date,
    especialidad varchar(25) (N),
    rol varchar(25) (N),
    horario varchar(12) (N)
}

DEPENDIENTE {
    [curp varchar(18) (U), no_empleado int (FK)] (PK),
    parentesco varchar(14),
    nombre varchar(20),
    ap_paterno varchar(25),
    ap_materno varchar(25) (N),
}

ORDEN {
    folio varchar(7) (PK),
    fecha date,
    monto_final int (C),
    no_empleado int (FK),
    rfc_cliente varchar(13) (FK),
}

PRODUCTO {
    Id_producto int (PK),
    nombre varchar(50),
    receta varchar(140),
    descripcion varchar(140),
    precio int,
    disponibilidad boolean (C),
    ventas int,
    id_categoria (FK)
}

```

```

CATEGORIA {
    id_categoria int (PK),
    descripcion varchar(100),
    nombre_categoria varchar(8)
}

CLIENTE {
    rfc varchar(13) (PK),
    razon_social varchar(35),
    fecha_nacimiento date,
    email varchar(50),
    calle varchar(35),
    no_exterior varchar(10),
    colonia varchar(35),
    estado varchar(18),
    cp int,
    nombre varchar(35),
    ap_paterno varchar(25),
    ap_materno varchar(25) (N)
}

TELEFONO {
    telefono varchar(10) (PK),
    no_empleado int (FK)
}

ORDEN_PRODUCTO {
    [id_producto int (FK), folio varchar(7) (FK)] (PK),
    cantidad smallint,
    precio_total_producto int (C)
}

```

3.3. Modelo Relacional

Una vez que ya realizamos la representación a modelo relacional, sólo queda realizarlo de forma gráfica. En este caso utilizaremos la herramienta ERStudio para realizar el modelo relacional con la notación crow's foot.

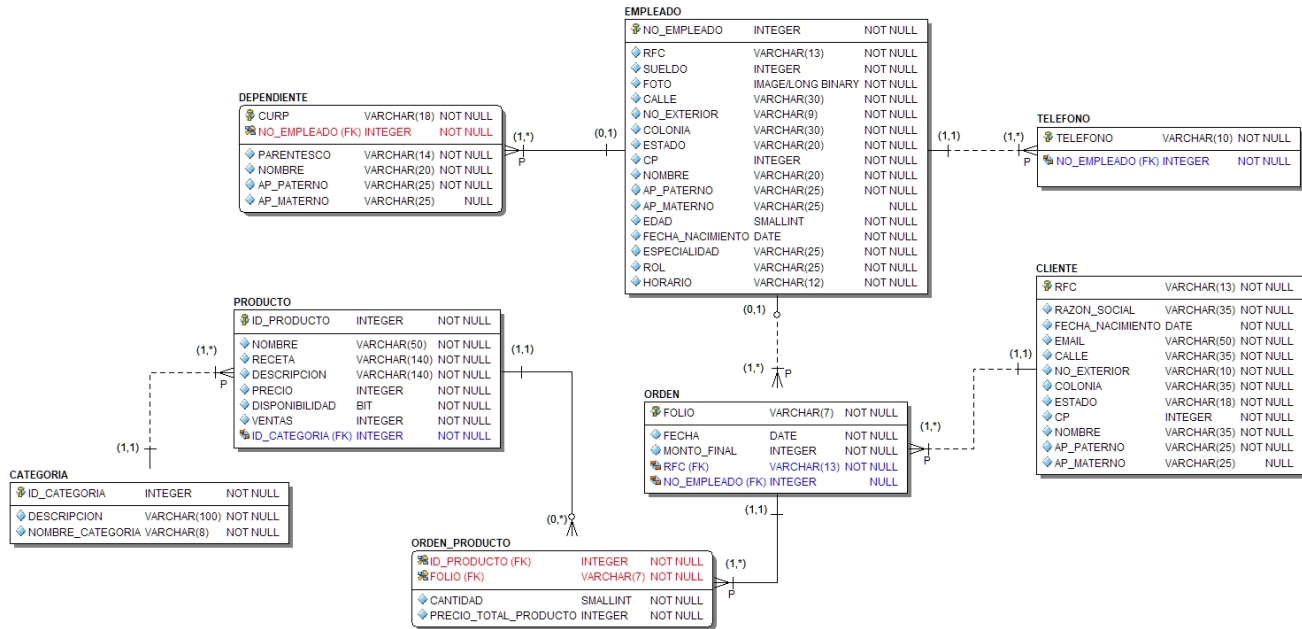


Figura 2: Modelo Relacional

3.4. Normalización

Ahora pasaremos a realizar la normalización de las tablas para buscar reducir redundancias e inconsistencias en los datos si es que se presenta el caso. **Normalización 1 forma normal**

Tabla EMPLEADO:

A: no_empleado (PK)
B: rfc (U)
C: sueldo
D: foto
E: calle
F: no_exterior
G: colonia
H: estado
I: cp
J: nombre
K: ap_paterno
L: ap_materno (N)
M: edad
N: fecha_nacimiento
O: especialidad (N)
P: rol (N)
Q: horario (N)

Identificando todas las dependencias:

$$\{A\} \rightarrow \{B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla CLIENTE si cumple con la 1 FN

Tabla DEPENDIENTE:

A y B: curp, no_empleado (FK) (PK)
C: parentesco
D: nombre
E: ap_paterno
F: ap_materno (N)

Identificando todas las dependencias:

$$\{A, B\} \rightarrow \{C, D, E, F\}$$

$$\{A\} \rightarrow \{D, E, F\}$$

$$\{A, B\} \rightarrow \{C\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla DEPENDIENTE si cumple con la 1 FN

Tabla ORDEN:

A: folio (PK)

B: fecha

C: monto_final (C),

D: no_empleado (FK),

E: rfc (FK)

Identificando todas las dependencias:

$$\{A\} \rightarrow \{B, C, D, E\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla DEPENDIENTE si cumple con la 1 FN

Tabla PRODUCTO:

A: id_producto (PK)

B: nombre

C: receta

D: descripción

E: precio

F: disponibilidad (C)

G: id_categoria (FK)

Identificando todas las dependencias:

$$\{A\} \rightarrow \{B, C, D, E, F, G\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla PRODUCTO si cumple con la 1 FN

Tabla CATEGORIA:

A: id_categoria (PK)

B: descripcion

C: nombre_categoria

Identificando todas las dependencias:

$$\{A\} \rightarrow \{B, C\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla PRODUCTO si cumple con la 1 FN

Tabla CLIENTE:

A: rfc (PK)

B: razón_solcial

C: fecha_nacimiento

D: email

E: calle

F: no_exterior

G: colonia

H: estado

I: cp

J: nombre

K: ap_paterno

L: ap_materno (N)

Identificando todas las dependencias:

$$\{A\} \rightarrow \{B, C, D, E, F, G, H, I, J, K, L\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla CLIENTE si cumple con la 1 FN

Tabla TELEFONO:

A: telefonno (PK)

B: no_empleado (FK)

Identificando todas las dependencias:

$$\{A\} \rightarrow \{B\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla TELEFONO si cumple con la 1 FN

Tabla ORDEN_PRODUCTO:

A y B: id_producto (FK), folio (FK) (PK)

C: cantidad

D: precio_total_producto (C)

Identificando todas las dependencias:

$$\{A, B\} \rightarrow \{C, D\}$$

$$\{A\} \rightarrow \{C\}$$

$$\{B\} \rightarrow \{D\}$$

¿Existe PK? : Si

¿Hay atributos multivaluados? : No

¿Hay grupos de repetición? : No

Por lo tanto, la tabla ORDEN_PRODUCTO si cumple con la 1 FN

Normalización 2 forma normal

Análisis tabla EMPLEADO

¿La PK es simple? : Si

Por lo tanto, la tabla EMPLEADO si cumple con la 2FN

Análisis tabla DEPENDIENTE

¿La PK es simple? : NO

Como la tabla no tiene PK simple, entonces analizamos si hay dependencias parciales

$$\{A, B\} \rightarrow \{C, D, E, F\}$$

$$\{A\} \rightarrow \{D, E, F\}$$

$$\{A, B\} \rightarrow \{C\}$$

Normalizamos la tabla DEPENDIENTE

Tabla DATOS_DEPENDIENTE

A: curp (PK)

D: nombre

E: ap_paterno

F: ap_materno (N)

Tabla PARENTESCO

A y B: curp (FK), no_employado (FK) (PK)

C: parentesco

Sin embargo, si se implementan estas tablas dentro de la base de datos entonces estaríamos permitiendo que existan varios dependientes sin tener asociado algún empleado previamente. Por cuestiones de rendimiento y la dependencia que existe entre el dependiente y el empleado, es correcto utilizarlo como si todos los atributos dependieran de la CURP del dependiente y el número del empleado.

Análisis tabla **ORDEN**
¿La PK es simple? : Si
Por lo tanto, la tabla ORDEN si cumple con la 2FN
Análisis tabla **PRODUCTO**
¿La PK es simple? : Si
Por lo tanto, la tabla PRODUCTO si cumple con la 2FN
Análisis tabla **CATEGORÍA**
¿La PK es simple? : Si
Por lo tanto, la tabla CATEGORÍA si cumple con la 2FN
Análisis tabla **CLIENTE**
¿La PK es simple? : Si
Por lo tanto, la tabla CLIENTE si cumple con la 2FN
Análisis tabla **TELEFONO**
¿La PK es simple? : Si
Por lo tanto, la tabla TELEFONO si cumple con la 2FN
Análisis tabla **ORDEN_PRODUCTO**
¿La PK es simple? : No

Como la tabla no tiene PK simple, entonces analizamos si hay dependencias parciales

$$\{A, B\} \rightarrow \{C, D\}$$

$$\{A\} \rightarrow \{C\}$$

$$\{B\} \rightarrow \{D\}$$

Como no tiene dependencias parciales entonces, la tabla ORDEN_PRODUCTO si cumple con la 2FN

Normalización 3 forma normal

Análisis tabla **EMPLEADO**
¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, la tabla EMPLEADO cumple con 3 FN
Análisis tabla **DEPENDIENTE**
¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN
Análisis tabla **ORDEN**
¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN
Análisis tabla **PRODUCTO**

¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN
Análisis tabla **CATEGORIA**

¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN
Análisis tabla **CLIENTE**

¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN
Análisis tabla **TELEFONO**

¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN
Análisis tabla **ORDEN_PRODUCTO**

¿Existe transitividad entre los atributos no principales? : No
Por lo tanto, cumple con 3 FN

4. Implementación

A continuación se presenta la implementación que se llevó a cabo para realizar la implementación de la base de datos para el restaurante:

4.1. Creación de tablas (DDL)

```
CREATE TABLE public.categoria (  
    id_categoria integer NOT NULL,  
    nombre_categoria character varying(8) NOT NULL,  
    descripcion character varying(100) NOT NULL  
);
```

```
CREATE TABLE public.cliente (  
    rfc character varying(13) NOT NULL,  
    razon_social character varying(55) NOT NULL,  
    nombre character varying(50),  
    ap_paterno character varying(30),  
    ap_materno character varying(30),  
    calle character varying(45),  
    no_exterior character varying(10),  
    colonia character varying(45),  
    estado character varying(18),  
    cp character varying(5),  
    fecha_nacimiento date,  
    email character varying(50)  
);
```

```
CREATE TABLE public.dependiente (  
    curp character varying(18) NOT NULL,  
    no_empleado text NOT NULL,  
    nombre character varying(30) NOT NULL,  
    ap_paterno character varying(25) NOT NULL,  
    ap_materno character varying(25),  
    parentesco character varying(14) NOT NULL  
);
```

```
CREATE TABLE public.empleado (  
    no_empleado text DEFAULT ( 'EMP-'::text || lpad((nextval('public.  
        emp_seq'::regclass))::text, 3, '0'::text)) NOT NULL,  
    rfc character varying(13) NOT NULL,  
    nombre character varying(20) NOT NULL,  
    ap_paterno character varying(25) NOT NULL,  
    ap_materno character varying(25),  
    calle character varying(30) NOT NULL,
```

```

    no_exterior character varying(9) NOT NULL,
    colonia character varying(30) NOT NULL,
    estado character varying(20) NOT NULL,
    cp character varying(5) NOT NULL,
    fecha_nacimiento date NOT NULL,
    edad integer,
    sueldo integer NOT NULL,
    especialidad character varying(25),
    rol character varying(25),
    horario character varying(12),
    foto character varying(20) NOT NULL
);

CREATE TABLE public.orden (
    folio text DEFAULT ('ORD-'::text || lpad((nextval('public.folio_seq'::regclass))::text, 3, '0')::text)) NOT NULL,
    fecha date DEFAULT (now())::date NOT NULL,
    monto_final integer DEFAULT 0,
    no_empleado text NOT NULL,
    rfc_cte character varying(13)
);

CREATE TABLE public.orden_producto (
    id_producto integer NOT NULL,
    folio text NOT NULL,
    cantidad smallint NOT NULL,
    precio_total_por_producto integer DEFAULT 0
);

CREATE TABLE public.producto (
    id_producto integer DEFAULT nextval('public.producto_id_producto_seq'::regclass) NOT NULL,
    nombre character varying(50) NOT NULL,
    descripcion character varying(350) NOT NULL,
    receta character varying(350) NOT NULL,
    disponibilidad boolean NOT NULL,
    id_categoria integer NOT NULL,
    precio integer NOT NULL,
    ventas integer DEFAULT 0
);

CREATE TABLE public.telefono (
    telefono character varying(18) NOT NULL,
    no_empleado text NOT NULL
);

```

4.2. Funciones

```
CREATE OR REPLACE FUNCTION public.empleado_ordenes(  
    num_empleado text)  
    RETURNS TABLE(folio_ord text, monto_orden integer)  
    LANGUAGE 'plpgsql'  
    COST 100  
    VOLATILE PARALLEL UNSAFE  
    ROWS 1000  
  
AS $BODY$  
DECLARE  
    mesero char varying;  
BEGIN  
    select empleado.horario into mesero from empleado where empleado.no_empleado = num_empleado ;  
    if mesero is null then  
        raise exception 'El empleado seleccionado NO es un mesero.';  
    else  
        RETURN QUERY  
        select orden.folio,orden.monto_final from empleado  
        join orden on empleado.no_empleado = orden.no_empleado  
        WHERE orden.fecha = now()::date;  
    END IF;  
  
END;  
$BODY$;
```

Dado un numero de empleado, regresa una tabla con todas las ordenes que ha trabajado, al igual que el monto por cada una, primero verifica que sea un mesero, guardamos en una variable, el atributo del horario, que es exclusivo de los meseros, si este atributo es nulo, quiere decir que no es mesero y lanza mensaje de error, Por otro lado, si el empleado es mesero, devuelve la tabla, seleccionamos el folio y el monto donde el no_empleado sea igual al ingresado, por medio de joins.

(Hay varios registros con \$0 de monto final, por múltiples pruebas realizadas)

21	ORD-028	50
22	ORD-029	0
23	ORD-030	0
24	ORD-031	0
25	ORD-032	0
26	ORD-033	0
27	ORD-034	0
28	ORD-036	25
29	ORD-053	286
30	ORD-037	129
31	ORD-038	179
32	ORD-054	215
33	ORD-056	339
34	ORD-057	355
35	ORD-045	251
36	ORD-046	251
37	ORD-047	251
38	ORD-048	303

	folio_ord text	monto_orden integer
1	ORD-049	303
2	ORD-050	615
3	ORD-007	0
4	ORD-008	0
5	ORD-009	0
6	ORD-010	0
7	ORD-011	0
8	ORD-012	0
9	ORD-013	0
10	ORD-014	0
11	ORD-015	0
12	ORD-016	0
13	ORD-017	0
14	ORD-018	0
15	ORD-019	0
16	ORD-020	0
17	ORD-023	52
18	ORD-051	511

```

5 CREATE OR REPLACE FUNCTION public.factura(
6     folio_orden text)
7     RETURNS text
8     LANGUAGE 'plpgsql'
9     COST 100
10    VOLATILE PARALLEL UNSAFE
11 AS $BODY$
12 DECLARE
13 fol text;
14 rfc_cliente char varying;
15 razon_cliente char varying;
16 fecha_orden date;
17 monto_orden integer;
18 registro Record;
19 productos CURSOR for select producto.nombre, producto.precio, cantidad,precio_total_por_producto
20 from orden_producto
21 join producto on orden_producto.id_producto = producto.id_producto
22 WHERE folio = folio_orden;
23 BEGIN
24
25 select folio,cliente.rfc,razon_social,orden.fecha,orden.monto_final
26 INTO fol,rfc_cliente,razon_cliente,fecha_orden,monto_orden from orden
27 JOIN cliente on rfc_cte = cliente.rfc WHERE folio = folio_orden;
28
29 if fol is not null then
30     RAISE NOTICE '-----';
31     RAISE NOTICE '-----FOLIO: %-----FECHA: %-----',fol,fecha_orden;
32     RAISE NOTICE '-----RFC EMISOR: CHE110527DQ3-----Nombre: Restaurante Los excentos-----';
33     RAISE NOTICE '-----REGIMEN FISCAL: ACTIVIDADES EMPRESARIALES-----EFECTO: INGRESO-----';
34     RAISE NOTICE '-----USO DE CFDI: GASTOS EN GENERAL-----FORMA DE PAGO:EFFECTIVO-----';
35     RAISE NOTICE '-----';
36     RAISE NOTICE '';
37     RAISE NOTICE ' RFC RECEPTOR: %',rfc_cliente;
38     RAISE NOTICE ' NOMBRE O RAZON SOCIAL: %',razon_cliente;

```

```

select folio,cliente.rfc,razon_social,orden.fecha,orden.monto_final
INTO fol,rfc_cliente,razon_cliente,fecha_orden,monto_orden from orden
JOIN cliente on rfc_cte = cliente.rfc WHERE folio = folio_orden;

if fol is not null then
    RAISE NOTICE '-----FOLIO: %-----FECHA: %-----',fol,fecha_orden;
    RAISE NOTICE '-----RFC EMISOR: CHE110527DQ3-----Nombre: Restaurante Los excentos-----';
    RAISE NOTICE '-----REGIMEN FISCAL: ACTIVIDADES EMPRESARIALES-----EFECTO: INGRESO-----';
    RAISE NOTICE '-----USO DE CFDI: GASTOS EN GENERAL-----FORMA DE PAGO:EFFECTIVO-----';
    RAISE NOTICE '-----';
    RAISE NOTICE ' RFC RECEPTOR: %',rfc_cliente;
    RAISE NOTICE ' NOMBRE O RAZON SOCIAL: %',razon_cliente;
    RAISE NOTICE '-----';
    for registro in productos loop
        RAISE NOTICE 'PRODUCTO: %',registro.nombre;
        RAISE NOTICE ' VALOR UNITARIO: $% CANTIDAD: % IMPORTE: %',registro.precio,registro.cantidad,registro.precio_total_por_l;
        RAISE NOTICE '-----';
    end loop;
    RAISE NOTICE '-----';
    RAISE NOTICE ' TOTAL: $%',monto_orden;
else
    RAISE EXCEPTION 'La orden ingresada NO existe O NO se ha ingresado el RFC del cliente. ';
END IF;

RETURN 'Su factura se ha realizado correctamente';
END;
$BODY$;

```

```

Proyecto=# factura('ORD-001');
ERROR: error de sintaxis en o cerca de «factura»
LINEA 1: factura('ORD-001');
      ^
Proyecto=# select * from factura('ORD-001');
NOTICE: -----
NOTICE: -----FOLIO: ORD-001-----FECHA: 2022-05-25-----
NOTICE: -----RFC EMISOR: CHE110527DQ3-----Nombre: Restaurante Los excentos-----
NOTICE: -----REGIMEN FISCAL: ACTIVIDADES EMPRESARIALES-----EFECTO: INGRESO-----
NOTICE: -----USO DE CFDI: GASTOS EN GENERAL-----FORMA DE PAGO:EFFECTIVO-----
NOTICE: -----
NOTICE:
NOTICE: RFC RECEPTOR: 10ZCO27J3
NOTICE: NOMBRE O RAZON SOCIAL: christian Inc
NOTICE:
NOTICE: PRODUCTO: sandwich
NOTICE: VALOR UNITARIO: $25 CANTIDAD: 3 IMPORTE: $75
NOTICE:
NOTICE: PRODUCTO: enchiladas suizas
NOTICE: VALOR UNITARIO: $52 CANTIDAD: 2 IMPORTE: $104
NOTICE:
NOTICE: PRODUCTO: limonada
NOTICE: VALOR UNITARIO: $25 CANTIDAD: 1 IMPORTE: $25
NOTICE:
NOTICE: -----
NOTICE: TOTAL: $204
NOTICE:
factura
-----
Su factura se ha realizado correctamente
(1 fila)

```

Dado un folio, se muestra el formato de una factura, para ello vamos a utilizar un cursor y una variable de tipo record, con el select vamos a guardar los datos de los productos que tengan el mismo folio, hacemos otro select para guardar los datos de la orden y del cliente, los guardamos en las variables correspondientes, primero con el if verifica que haya un cliente al cual facturar, si el cliente dio su RFC, se imprime como encabezado, el folio de la orden, y la fecha de esta, para asemejarse a una factura se incluyen datos adicionales, como el nombre del emisor, el régimen o el uso de CFDI, después se imprimen los datos del cliente, se omite la dirección, por no ser obligatoria en una factura real. Mediante un for recorremos el cursor, y en cada iteración, imprimimos los datos de los productos, al final se muestra el monto total.


```

1  -- FUNCTION: public.ingresar_productos(character varying, integer, character varying, integer, character varying, integer)
2
3  -- DROP FUNCTION IF EXISTS public.ingresar_productos(character varying, integer, character varying, integer, character varying, integer);
4
5  CREATE OR REPLACE FUNCTION public.ingresar_productos(
6      prod1 character varying,
7      cant1 integer,
8      prod2 character varying,
9      cant2 integer,
10     prod3 character varying,
11     cant3 integer)
12     RETURNS text
13     LANGUAGE 'plpgsql'
14     COST 100
15     VOLATILE PARALLEL UNSAFE
16 AS $BODY$
17 DECLARE
18     id_1 integer ;
19     id_2 integer;
20     id_3 integer;
21 BEGIN
22     SELECT id_producto into id_1 FROM producto WHERE nombre = prod1;
23     SELECT id_producto into id_2 FROM producto WHERE nombre = prod2;
24     SELECT id_producto into id_3 FROM producto WHERE nombre = prod3;
25
26     if (prod1 = '0' and prod2 = '0' and prod3 = '0') then
27         RAISE EXCEPTION 'No se ha ingresado ningun producto';
28     else
29         INSERT INTO public.orden(no_empleado)VALUES ('EMP-001');
30     end if;
31
32
33     if prod1 != '0' then |
34         INSERT INTO public.orden_producto(id_producto, folio, cantidad) VALUES (id_1, orden_actual(), cant1);
35     end if;

```

```

19 id_2 integer;
20 id_3 integer;
21 BEGIN
22     SELECT id_producto into id_1 FROM producto WHERE nombre = prod1;
23     SELECT id_producto into id_2 FROM producto WHERE nombre = prod2;
24     SELECT id_producto into id_3 FROM producto WHERE nombre = prod3;
25
26     if (prod1 = '0' and prod2 = '0' and prod3 = '0') then
27         RAISE EXCEPTION 'No se ha ingresado ningun producto';
28     else
29         INSERT INTO public.orden(no_empleado)VALUES ('EMP-001');
30     end if;
31
32
33     if prod1 != '0' then |
34         INSERT INTO public.orden_producto(id_producto, folio, cantidad) VALUES (id_1, orden_actual(), cant1);
35     end if;
36
37     if prod2 != '0' then
38         INSERT INTO public.orden_producto(id_producto, folio, cantidad) VALUES (id_2, orden_actual(), cant2);
39     end if;
40
41     if prod3 != '0' then
42         INSERT INTO public.orden_producto(id_producto, folio, cantidad) VALUES (id_3, orden_actual(), cant3);
43     end if;
44
45
46     RETURN id_1;
47 END;
48 $BODY$;
49

```

Funcion para la interfaz grafica, Recibe los datos obtenidos en el formulario, permite ingresar 3 o menos productos al registro, crea una nueva orden, y con otra funcion obtenemos el folio de esa orden.

```

4
5 CREATE OR REPLACE FUNCTION public.orden_actual(
6 )
7 RETURNS text
8 LANGUAGE 'plpgsql'
9 COST 100
10 VOLATILE PARALLEL UNSAFE
11 AS $BODY$
12 DECLARE fol text;
13 BEGIN
14 select folio into fol from orden order by folio desc
15 limit 1;
16
17 RETURN fol;
18 END;
19 $BODY$;
20
21 ALTER FUNCTION public.orden_actual()
22 OWNER TO postgres;
23

```

Funcion para la interfaz grafica, como se pueden crear n numero de ordenes, en la funcion necesitamos el folio de dicha orden, como los folios son numerados, la ultima orden, es la ultima en ser creada por la funcion anterior, solo seleccionamos el folio, lo ordenamos de manera descendente, y lo limitamos en 1 registro.

```

--
5 CREATE OR REPLACE FUNCTION public.no_ventas(
6     fecha_ini date,
7     fecha_fin date)
8 RETURNS TABLE(no_ventas bigint, monto_total bigint)
9 LANGUAGE 'plpgsql'
10 COST 100
11 VOLATILE PARALLEL UNSAFE
12 ROWS 1000
13
14 AS $BODY$
15 BEGIN
16 RETURN QUERY
17 SELECT count(folio),sum(monto_final) from orden where fecha between fecha_ini and fecha_fin;
18
19 END;
20 $BODY$;
--

```

Dada una fecha de inicio y una de fin, la funcion regresa una tabla con el numero de ventas y el monto total por todas las ordenes, simplemente retornamos el query, donde seleccionamos el numero de ordenes, y la suma de los montos, donde la fecha se encuentre entre esos dos rangos.

```
1 select * from no_ventas('2022-05-10',now())::date);
```

Data Output Explain Messages Notifications

	no_ventas bigint	monito_total bigint	
1	42	4977	

```
5 CREATE OR REPLACE FUNCTION public.telefonos(
6     emp text)
7     RETURNS text
8     LANGUAGE 'plpgsql'
9     COST 100
10    VOLATILE PARALLEL UNSAFE
11 AS $BODY$
12 DECLARE
13 numero text;
14 pivote text = '';
15 registro Record;
16 telefonos CURSOR FOR select telefono from telefono
17 join empleado on empleado.no_empleado = telefono.no_empleado
18 WHERE telefono.no_empleado = emp order by telefono;
19
20 BEGIN
21 for registro in telefonos loop
22     pivote = numero;
23     --numero = concat(pivote,' ', registro.telefono);
24     numero = concat(registro.telefono,' ',pivote);
25     end loop;
26 RETURN numero;
27 END;
28 $BODY$;
```

Funcion para interfaz grafica, que permite concatenar todos los teléfonos de los empleados en una sola línea de texto, se utiliza en el select para juntar los números en un solo registro, usamos un cursor, seleccionamos los números de teléfono de un solo empleado, usamos un for para recorrer el cursor, usamos una variable pivote, para ir almacenando todas las concatenaciones anteriores, y esa variable la volvemos a concatenar con el teléfono siguiente, hasta terminar.

4.3. Vistas

platillo_mas_vendido

General Definition Code Security SQL

```
1 SELECT p.id_producto,  
2     p.nombre,  
3     p.descripcion AS desc_plat,  
4     p.precio,  
5     p.ventas,  
6     p.disponibilidad,  
7     p.receta,  
8     c.nombre_categoria,  
9     c.descripcion AS desc_catego  
10 FROM producto p  
11     JOIN categoria c ON c.id_categoria = p.id_categoria  
12 WHERE c.id_categoria = 1  
13 ORDER BY p.ventas DESC  
14 LIMIT 1;
```

Una vista que nos muestra la información del platillo mas vendido, hacemos un join para obtener solo los platillos y descartar las bebidas, lo ordenamos de manera descendente por medio del atributo ventas, y limitamos los registros a 1.

```
1 select * from platillo_mas_vendido;
```

Data Output									
id_producto	nombre	desc_plat	precio	ventas	disponibilidad	receta	nombre_categoria	desc_catego	
integer	character varying (50)	character varying (350)	integer	integer	boolean	character varying (350)	character varying (8)	character varying (100)	
1	enchiladas suizas	Platillo elaborado con tortillas de maíz rellenas con pollo, bañadas c...		52	56	true	Hervir durante 15 min los tom...	platillo	Cada platillo se sirve con una p...

productos_no_disponibles

General Definition Code Security SQL

```
1 SELECT p.nombre,  
2     p.precio,  
3     p.disponibilidad,  
4     p.ventas,  
5     c.nombre_categoria  
6 FROM producto p  
7     JOIN categoria c ON c.id_categoria = p.id_categoria  
8 WHERE p.disponibilidad = false;
```

Vista que muestra el nombre, no_ventas y la categoría de aquellos productos que no estén disponibles, se hace un join con la tabla categoría, para obtener la categoría del producto.

```
1 select * from productos_no_disponibles;
```

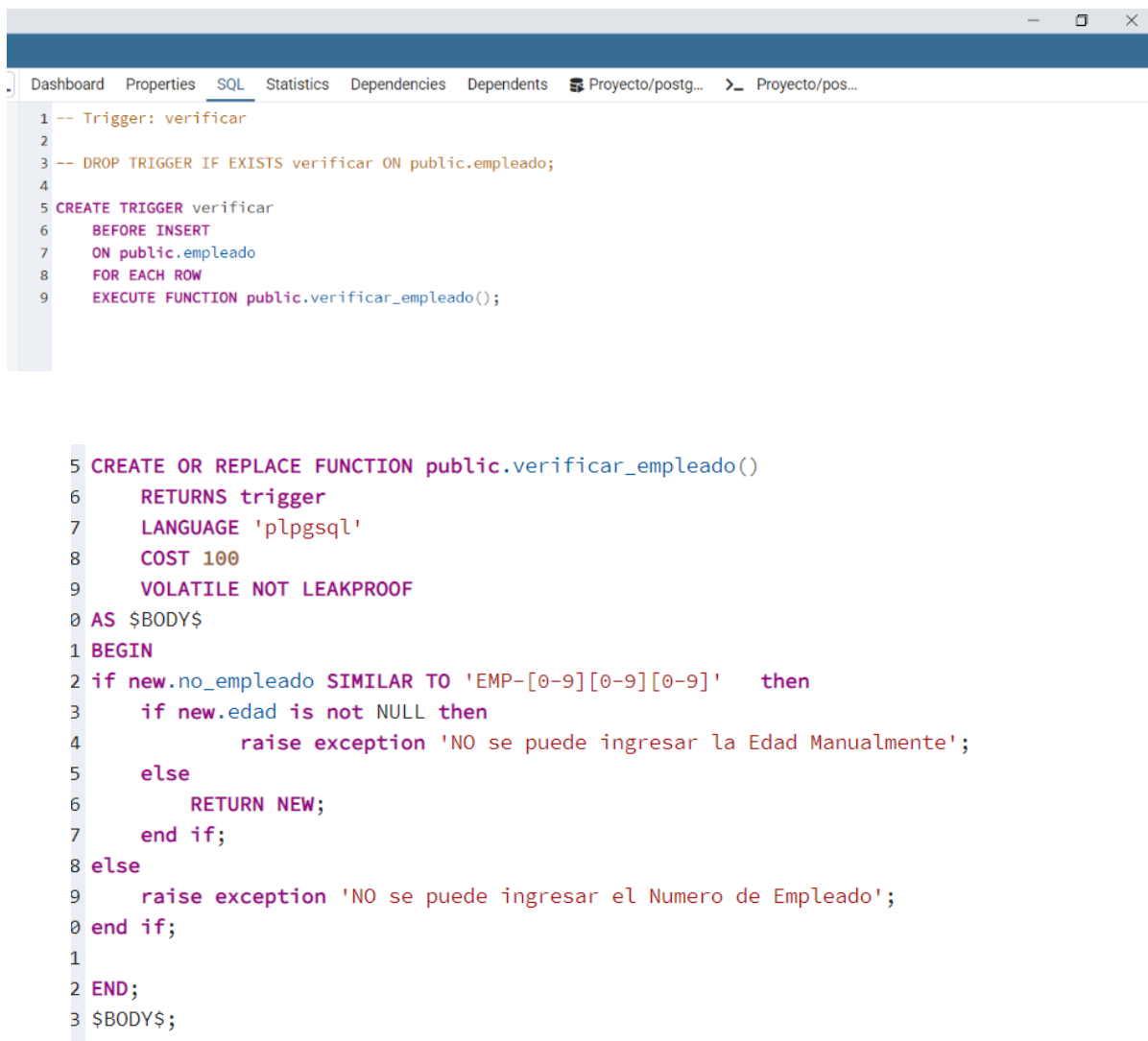
	nombre	precio	disponibilidad	ventas	nombre_categoria	
	character varying (50)	integer	boolean	integer	character varying (8)	
1	hot dog	15	false	0	platillo	
2	coca cola	10	false	0	bebida	

TRIGGERS: TODAS LAS FUNCIONES SON VOLATILES, PUEDEN DEVOLVER UN RESULTADO DIFERENTE DEPENDIENDO DE LAS ENTRADAS O POR EL MISMO DISEÑO DE LA MISMA.

```
CREATE OR REPLACE FUNCTION public.calcula_edad()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
edadcal integer;
BEGIN
edadcal = (extract(year from now()) - extract(year from NEW.fecha_nacimiento))::integer;
UPDATE empleado SET edad = edadcal WHERE no_empleado = NEW.no_empleado;
RETURN NEW;
END;
$BODY$;
```

```
CREATE TRIGGER calcula_edad
    AFTER INSERT
    ON public.empleado
    FOR EACH ROW
    EXECUTE FUNCTION public.calcula_edad();
```

Después de que se agregue un registro en la tabla empleado, se ejecuta la función `calcula_edad()`, calculamos la edad restando el año del día de hoy – el año de la fecha de nacimiento ingresado, se guarda el valor, se hace un update del atributo edad en la tabla alumno, donde el `no_empleado` sea igual al del agregado recientemente.



```
1 -- Trigger: verificar
2
3 -- DROP TRIGGER IF EXISTS verificar ON public.empleado;
4
5 CREATE TRIGGER verificar
6 BEFORE INSERT
7 ON public.empleado
8 FOR EACH ROW
9 EXECUTE FUNCTION public.verificar_empleado();

5 CREATE OR REPLACE FUNCTION public.verificar_empleado()
6 RETURNS trigger
7 LANGUAGE 'plpgsql'
8 COST 100
9 VOLATILE NOT LEAKPROOF
0 AS $BODY$
1 BEGIN
2 if new.no_empleado SIMILAR TO 'EMP-[0-9][0-9][0-9]' then
3     if new.edad is not NULL then
4         raise exception 'NO se puede ingresar la Edad Manualmente';
5     else
6         RETURN NEW;
7     end if;
8 else
9     raise exception 'NO se puede ingresar el Numero de Empleado';
0 end if;
1
2 END;
3 $BODY$;
```

Cada que se ingrese un registro, antes se ejecuta la funcion verificar_empleado, donde primero verifica que el no_empleado tenga el formato correspondiente, si esta escrito bien o se pone el default, verifica que en el atributo edad, no se ingrese ningún valor, ya que es calculado, si se cumplen las condiciones se ingresa el registro.

```
CREATE TRIGGER revisa_orden
    BEFORE INSERT
    ON public.orden
    FOR EACH ROW
    EXECUTE FUNCTION public.revisa_orden();
```

```

5 CREATE OR REPLACE FUNCTION public.revisa_orden()
6     RETURNS trigger
7     LANGUAGE 'plpgsql'
8     COST 100
9     VOLATILE NOT LEAKPROOF
10 AS $BODY$
11 DECLARE
12 horario_mesero char varying;
13 BEGIN
14 select horario into horario_mesero from empleado where empleado.no_empleado = new.no_empleado;
15 if new.folio SIMILAR TO 'ORD-[0-9][0-9][0-9]' then
16     if new.monto_final = 0 then
17         if horario_mesero is null then
18             raise exception 'El empleado ingresado No es un mesero que pueda levantar ordenes';
19         else
20             return new;
21         end if;
22     else
23         raise exception 'No se puede ingresar manualmente el monto total de la orden';
24     end if;
25 else
26     raise exception 'El Numero de folio no tiene el formato correcto';
27 end if;
28 end;
29 $BODY$;
30

```

El trigger antes de insertar en la tabla orden, verifica que el folio tenga el formato correspondiente, después verifica que el monto de la orden sea cero(Default) y después verifica que el empleado sea un mesero, si las tres condiciones se cumplen se inserta el registro, si no manda un mensaje correspondiente al error.

```

4
5 CREATE TRIGGER actualiza_monto_total_prod
6     AFTER INSERT
7     ON public.orden_producto
8     FOR EACH ROW
9     EXECUTE FUNCTION public.actualiza_monto_total_prod();

```

```

1 CREATE OR REPLACE FUNCTION public.actualiza_monto_total_prod()
2     RETURNS trigger
3     LANGUAGE 'plpgsql'
4     COST 100
5     VOLATILE NOT LEAKPROOF
6 AS $BODY$
7 DECLARE
8 monto_finali integer;
9 monto_total_producto integer;
10 no_ventas integer;
11 precio_producto integer;
12 BEGIN
13 select ventas,precio into no_ventas,precio_producto from producto where producto.id_producto = new.id_producto;
14 select monto_final into monto_finali from orden where orden.folio = new.folio;
15
16 no_ventas = no_ventas + new.cantidad;
17 UPDATE producto SET ventas = no_ventas WHERE id_producto = new.id_producto; --actualiza el numero de ventas, sumando el no que tenia mas la cantidad nueva
18
19 monto_total_producto = new.cantidad * precio_producto;
20 UPDATE orden_producto SET precio_total_por_producto = monto_total_producto WHERE id_producto = new.id_producto and folio=new.folio;
21 -- actualiza el monto total de un producto y su cantidad
22
23 monto_finali = monto_finali + monto_total_producto;
24 UPDATE orden SET monto_final = monto_finali WHERE folio = new.folio;
25 --se actualiza el monto total de la orden, sumando todos los montos por producto que pueda existir.
26 return new;
27 END;
28 $BODY$;

```

El trigger se ejecuta después de insertar un registro en la tabla de orden_producto, la cual une la orden con los diferentes productos, se selecciona el monto por producto, el monto_final de la orden, y las ventas del producto ingresado, para ello hay 3 UPDATES que hacen lo mismo, para las ventas del producto se suma el valor que tenía mas las cantidades ingresadas, para el monto por producto multiplicamos la cantidad de producto por el precio, para el monto final sumamos el valor que tenía mas el obtenido en la operación anterior.

```

4
5 CREATE TRIGGER revisa_ingreso_productos_orden
6     BEFORE INSERT
7     ON public.orden_producto
8     FOR EACH ROW
9     EXECUTE FUNCTION public.revisa_ingreso_productos_orden();

--
10 CREATE OR REPLACE FUNCTION public.revisa_ingreso_productos_orden()
11     RETURNS trigger
12     LANGUAGE 'plpgsql'
13     COST 100
14     VOLATILE NOT LEAKPROOF
15 AS $BODY$
16 DECLARE
17 disponible bool;
18 nombre_prod char varying;
19 BEGIN
20 select disponibilidad,nombre into disponible,nombre_prod from producto where producto.id_producto = new.id_producto;
21 if (disponible) then
22     if new.precio_total_por_producto is not null then
23         if new.precio_total_por_producto = 0 then
24             return new;
25         else
26             raise exception 'No se puede ingresar manualmente el monto total por el producto';
27         end if;
28     else
29         return new;
30     end if;
31 else
32     raise exception 'El producto % NO esta disponible',nombre_prod;
33 end if;
34 end;
35 $BODY$;

```

El trigger se va a ejecutar antes de ingresar datos en la tabla orden_producto, donde se verifica que el producto este disponible, si cumple la condicion, se compara que el precio total por producto sea igual a cero(Default), no se puede ingresar el valor manualmente.

```

1
2 CREATE TRIGGER revisa_ingreso_productos
3     BEFORE INSERT
4     ON public.producto
5     FOR EACH ROW
6     EXECUTE FUNCTION public.revisa_ingreso_productos();

```



```

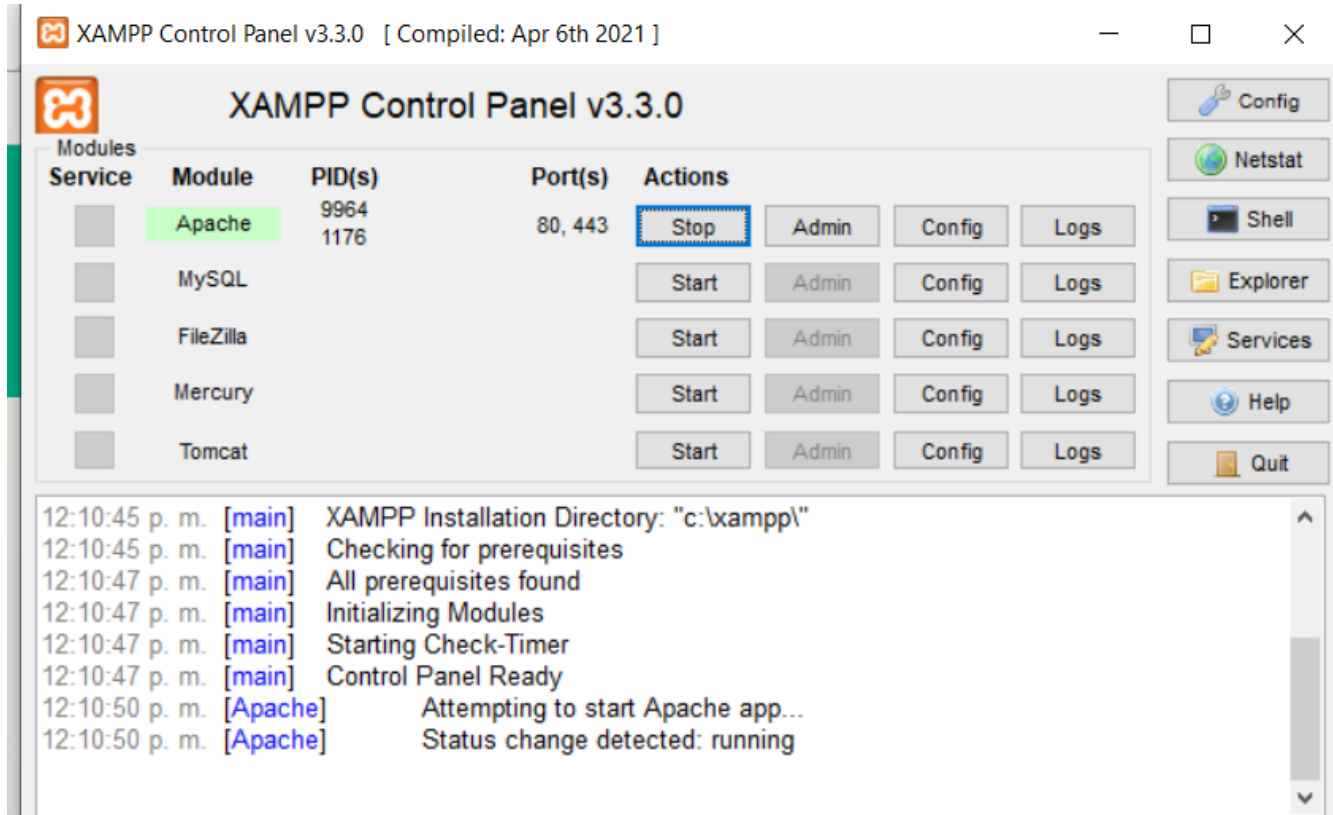
5 CREATE OR REPLACE FUNCTION public.revisa_ingreso_productos()
6     RETURNS trigger
7     LANGUAGE 'plpgsql'
8     COST 100
9     VOLATILE NOT LEAKPROOF
10 AS $BODY$
11
12 BEGIN
13     if new.ventas = 0 then
14         return new;
15     else
16         raise exception 'No se puede ingresar manualmente el no de ventas del producto';
17     end if;
18 end;
19 $BODY$;

```

El trigger se ejecuta antes de ingresar en la tabla productos, donde se verifica que el número de ventas sea cero, para que no se pueda ingresar un producto nuevo con un número de ventas que no generó.

5. Presentación

Se descargó XAMPP CONTROL, para trabajar con un servidor apache, y poder ver la pagina a desarrollar.



Comenzamos indicando la conexion a la base de datos en la cual vamos a trabajar



Index.php, es la pagina inicial donde se debe ingresar, y que va a servir para generar el formulario, para ingresar los 3 o menos productos en una orden, donde las opciones que se muestran son obtenidas directamente de las bases de datos, si se ingresan nuevos productos (que estén disponibles) o se eliminan las listas se van a actualizar automáticamente.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title> PRUEBA </title>
7 </head>
8 <body>
9
10 <h2> Ingreso de producto en la orden</h2>
11 <form action="alta.php" method = "POST"> <!-- Se abre un formulario que al mandarse nos direcciona a al archivo alta.php -->
12 <label> Seleccione los productos y cantidades que iran en la orden, si no necesita agregar mas productos, deje los recuadros tal cual estan</
13 <label for="producto">Ingrese el nombre del producto:</label> <!-- Etiqueta que se imprime en pantalla -->
14 <select name="producto1"> <!-- Se crea una lista desplegable -->
15 <option value="0">Selecione</option> <!-- Definimos el valor por defecto-->
16 <?php
17 require 'conexion.php';
18 $producto = "SELECT * FROM public.producto where disponibilidad = true order by nombre";
19 $resultado = pg_query($conexion,$producto);
20 while ($obj = pg_fetch_assoc($resultado)){
21     echo <option value="'. $obj["nombre"]. "'>'. $obj["nombre"]. '</option>';
22 }
23 >
24 <!-- Se abre un php donde consultamos el nombre de todos los productos disponibles en la base -->
25 <!-- con un ciclo while recorremos todos registros obtenidos en la consulta y el valor obtenido
26 lo ponemos como una opcion de la liga con ese valor para simplificar -->
27 </select><br><br><!-- Se cierra el registro de datos para la lista desplegable -->
28 <label for="nombre">Ingresa la cantidad:</label>
29 <input type="number" name="cantidad1"> <br><br>
30 <!-- Se crea un cuadro blanco donde admite solo numeros, que representa la cantidad del producto-->
31 <!-- Se REPITE EL MISMO PROC PARA LOS DEMAS PRODUCTOS -->
32
33 <label for="producto">Ingrese el nombre del producto:</label>
34 <select name="producto2">
35 <option value="0">Selecione</option>
36 <?php
37 require 'conexion.php';

```

```

57 }
58 >
59 </select><br><br>
60 <label for="nombre">Ingresa la cantidad:</label>
61 <input type="number" name="cantidad3"> <br><br>
62
63 <input type="submit" value = "Ingresar orden"> <label > </label> <input type="reset" name="Reset"><br><br>
64 <!-- Se crea un boton que es el evento para que la accion del formulario ocurra
65 y se crea un boton de reset para reiniciar las opciones por defecto -->
66 </form>
67
68
69
70
71
72
73 <form action="prueba.php" method="POST">
74 <input type="submit" value="VER informacion de los empleados.">
75 <!-- Se abre un forms que nos va a llevar a prueba.php, es la pagina donde esta la tabla
76 para ello creamos un boton para activar el evento -->
77 </form>

```

Al final se agregan dos botones que nos van a direccionar a paginas diferentes, el que dice Ingresar orden, es para ingresar los productos seleccionados en una orden, y el que dice “Ver información de los empleados” nos direcciona a la pagina donde se muestra toda la información de los empleados.

Como se ve el Formulario:

Ingreso de producto en la orden

Seleccione los productos y cantidades que iran en la orden, si no necesita agregar mas productos, deje los recuadros tal cual estan

Ingrese el nombre del producto:

Ingresar la cantidad:

Ingrese el nombre del producto:

Ingresar la cantidad:

Ingrese el nombre del producto:

Ingresar la cantidad:

Tabla de empleados:

```
1 <?php
2 require 'conexion.php';
3 $query = "SELECT no_empleado,foto,UPPER(concat(nombre,' ',ap_paterno)) as nombre,
4 rfc,UPPER(concat(calle,' #',no_exterior,' ',colonia,' ',estado,' ',cp)) as direccion,
5 fecha_nacimiento,edad, concat('$',sueldo) as sueldo, horario,
6 UPPER(especialidad) as especialidad ,UPPER(rol) as rol,telefonos(no_empleado) as telefono FROM empleado";
7
8 $consulta = pg_query($conexion,$query);
9
10 <!-- indicamos la consulta que se va a ejecutar($query), el resultado lo guardamos en la variable consulta, poniendo la conexion y el query -->
11 <!DOCTYPE html>
12 <html lang = "en">
13 <head>
14 <meta charset="UTF-8">
15 <meta name="viewport" content="width=device-width, initial-scale=1.0">
16 <h2> Tabla de Empleados</h2>
17 </head>
18 <body>
19
20 <table border = "1">
21 <thead>
22 <tr>
23
24 <!-- Creamos una tabla con las siguientes columnas y su nombre correspondiente-->
25 <th>No_empleado</th>
26 <th>Foto</th>
27 <th>Nombre</th>
28 <th>RFC</th>
29 <th>Direccion</th>
30 <th>Fecha_Nacimiento</th>
31 <th>Edad</th>
32 <th>Sueldo</th>
33 <th>Horario</th>
34 <th>Especialidad</th>
35 <th>Rol</th>
36 <th>Telefonos</th>
37 </tr>
```

Este es el archivo donde se encuentra la lógica de la tabla de los empleados, primero se hace una consulta SELECT con las columnas que vamos a mostrar, y generamos una tabla por medio de html con las mismas columnas que se solicitan en el query.

```

C:\xampp\htdocs\proyecto\prueba.php (Tasm) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

conexión.php  index.php  prueba.php  altap.php




39  </thead>
40  <tbody>
41
42  <!-- Abrimos y cerramos constantemente un php, para obtener y trabajar con los resultados del php
43  e insertarlos en la tabla del html.
44  Basicamente, recorremos los registros obtenido como si fuera una variable Record
45  en un ciclo while y en cada iteración se insertan los valores en su columna correspondiente con la variable
46  $obj->nombre donde obj es el objeto donde se guarda el registro actual y ->nombre es el nombre de la columna obtenida
47  en la base de datos. para la imagen indicamos que se imprima la imagen en su casilla correspondiente
48  ,con la dirección que guardamos en la base de datos -->
49  <?php
50  while($obj=pg_fetch_object($consulta)){ ?>
51  <tr>
52  <center>
53  <td><?php echo $obj->no_employado;?></td>
54  </center>
55  <td><?php echo "<img src='$obj->foto';?></td>
56  <td><?php echo $obj->nombre;?></td>
57  <td><?php echo $obj->rfc;?></td>
58  <td><?php echo $obj->direccion;?></td>
59  <td><?php echo $obj->fecha_nacimiento;?></td>
60  <td><?php echo $obj->edad;?></td>
61  <td><?php echo $obj->sueldo;?></td>
62  <td><?php echo $obj->horario;?></td>
63  <td><?php echo $obj->especialidad;?></td>
64  <td><?php echo $obj->rol;?></td>
65  <td><?php echo $obj->telefono;?></td>
66
67  </tr>
68  <?php
69  }
70  ?>
71  </table>
72  <br>
73  <form action="index.php" method="POST">
74  <input type="submit" value="Regresar">
75  </form>
76

```

Y las vamos imprimiendo en pantalla, con un código similar al utilizar una variable Record en postgres. Para la foto se indica a php que imprima la foto, en una determinada dirección, dicha dirección se encuentra en el directorio raíz del servidor, por lo que solo guardamos el nombre del archivo y su extensión.

Como se ve la Tabla de EMPLEADOS:

Tabla de Empleados

No_employado	Foto	Nombre	RFC	Direccion	Fecha_Nacimiento	Edad	Sueldo	Horario	Especialidad	Rol	Telefonos
EMP-001		CHRISTIAN LOPEZ	LOZC010528J73	BAJA CALIFORNIA NORTE #221 .PROVIDENCIA ,CDMX 07550	2001-05-28	21	\$2500	17:00- 18:00	CHEF		77210934, 654739054, 562343212, 55215057, 55214323, 07743554,
EMP-002		BENITO WEBER	BWM160310C75	AV DOCTORES #54 ,PROVIDENCIA .CHIHUAHUA 98765	1980-05-28	42	\$4000		MARISQUERO	GERENTE	77235442, 75553342, 34543124,
EMP-003		PEDRO CRUZ	CUCP930104NUA	5 DE FEBRERO #57 ,ZIHUATANEJO .MICHOACAN 58942	1980-03-12	42	\$500	14:00- 13:00	COCINERO		5590633114, 5590206633,

Regresar

```

conexion.php x index.php x prueba.php x alta.php x
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title></title>
</head>
<body>
  <?php
    require 'conexion.php';

    $query = "SELECT * FROM Ingresar_productos('$ _REQUEST[producto1]','$ _REQUEST[cantidad1]','$ _REQUEST[producto2]','$ _REQUEST[cantidad2]','$ _REQUEST[producto3]','$ _REQUEST[cantidad3]')";
    $consulta = pg_query($conexion,$query); "IF"
    pg_close();
    echo 'La orden se ha registrado correctamente';
  ?>

  <!-- Se ejecuta una funcion diseñada para este requerimiento, como parametros, pasamos los valores obtenidos en cada select, los valores son llamados por medio de "REQUEST", si se ejecuta el query al final se imprime un mensaje de que se ingresaron los productos en la orden . Al final se crea un boton para regresar a la pagina inicial, para ver los mensajes -->

  $consulta = pg_query($conexion,$query);
  ?>

  <form action="index.php" method="POST">
    <input type="submit" value="Regresar">
  </form>

</body>
</html>

```

Una vez ingresada la orden, nos dirigen a este archivo, donde se ejecuta una funcion, creada para esta parte, si se ejecuta correctamente al final sale un mensaje afirmando lo anterior, y por ultimo un boton para regresar a la pagina.

6. Conclusiones

Hernández Hernández Cristian: En cada etapa del desarrollo de este proyecto existieron situaciones en que era sencillo visualizar la solución de algún problema que se presentaba, mientras que en otros casos era más complicado. Esto se fue originando mientras se avanzaba con el proyecto, al principio no hubo muchas complicaciones, pero después comenzaron a aparecer, principalmente en el almacenado de la información y algunas funciones ya que se tienen que cumplir varias condiciones. También se presentaron complicaciones en la implementación de la página web ya que no tenía los conocimientos suficientes para realizar una, pero investigando se pudo lograr realizarlo. Para ello fue importante tener una comunicación con el equipo para dar soluciones más rápidas a estas complicaciones. En mi caso se hizo un poco complicado algunas cuestiones del manejador PostgreSQL puesto que en el laboratorio utilicé el manejador de Oracle y en algunos aspectos es diferente. En general el proyecto requirió el conocimiento de lo visto durante todo el curso desde el comienzo al analizar el problema planteado hasta su implementación final.

Laparra Miranda Sandra: En este proyecto se aplicaron todos los conceptos vistos durante el curso siendo estos, modelo entidad-relación, modelo relacional, normalización, mapeo de entidades y sobre todo conceptualizar los requerimientos y el funcionamiento más cercano a lo que requería nuestro caso de estudio. Utilizar el manejo de Postgres fue complicado al inicio, ya que en el laboratorio se utilizó Oracle por lo mismo fue un poco diferente migrar de un manejador a otro, sin embargo la adaptación al cambio fue sencilla porque se utilizó el mismo lenguaje así como los conceptos vistos en teoría, no obstante algunas funciones de ambos manejadores si pueden llegar a cambiar. Se logró implementar una base de datos para nuestro caso de estudio que fue el restaurante, y de esta forma se aprendió la aplicación tan importante que tienen las bases de datos para almacenar grandes cantidades de información que se requieren por las empresas para facilitar la administración y organización de sus productos y ventas, de tal forma que el acceso a la información de su negocio sea más accesible y así mejore su incremento de clientes.

López Zugasti Christian: Uno de los principales retos, al momento de programar la base fue generar los trigger y sus funciones para asegurar que se esta ingresando la información correctamente, que de primer instante parecía algo banal, pero resulta en muchas condiciones que se tienen que cumplir para asegurar el correcto registro de la información, y a pesar de cubrir varias columnas, nos faltaron varios aspectos, como el RFC, la principal dificultad fue en la parte dos, de la interfaz grafica, tuve que aprender un nuevo “lenguaje” para desarrollar la página y adicionalmente investigar la manera de conectarla con la base de datos.

Referencias

- [1] CHURCHER, C. , *Beginning database design. From novice to professional.*, Apress, Berkeley, California, 2007.
- [2] DE MIGUEL, A., PIATTINI, M. y MARCOS, E. *Diseño de bases de datos relacionales.*, Alfaomega/RA-MA, Madrid, 2000.
- [3] OPPEL, A. y SHELDON, R. *Fundamentos de SQL*, Tercera edición, Mc Graw Hill, México, 2010.
- [4] RAMEZ, E. y SHAMKANT, B., N *Fundamentos de Sistemas de Bases de Datos*, Pearson Educación, México, 2007.