



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

BASES DE DATOS Grupo 1

Profesor: Fernando Arreola Franco

Fecha de entrega: 18 de MAYO de 2024

PROYECTO FINAL

INTEGRANTES

Apellido paterno	Apellido materno	Nombre(s)
Barragán	Pilar	Diana
López	Ramírez	Monserrat
Martínez	Bravo	Daniela
Suarez	Velasco	Gabriela
Aguirre	Córdova	Omar Gabriel

Índice

Introducción	3
Objetivo	4
Plan de trabajo	4
Diseño	7
Modelo Conceptual	7
Modelo Relacional Intermedio	11
Implementación	16
Creación de tablas	16
Creación de Funciones y Triggers	20
Creación de un índice	26
Consultas y Vistas de la base de datos	26
Presentación	31
Conclusiones:	40
Referencias	42

Introducción

La descripción del problema plantea el desafío de diseñar una base de datos para un restaurante con el objetivo de digitalizar sus operaciones. Este sistema informático debe abordar diversos aspectos, desde la gestión de empleados hasta el registro de órdenes y la generación de facturas. Aquí se resumen los requisitos clave:

En cuanto a la gestión de empleados, se deben almacenar detalles como RFC, número de empleado, nombre completo, fecha de nacimiento, contacto, dirección, sueldo y roles específicos según el tipo de empleado (cocinero, mesero, administrativo). Además, se requiere mantener un registro de dependientes de empleados.

La gestión de productos implica incluir información detallada sobre los platillos y bebidas ofrecidos por el restaurante, como descripción, receta, precio y disponibilidad, junto con su categorización.

Para la gestión de órdenes, se necesita un sistema que registre órdenes, incluyendo el folio de la orden, fecha y hora, total a pagar, mesero responsable y detalles de los productos incluidos en la orden. Además, se debe permitir la solicitud de factura por parte de los clientes.

Además de estas funcionalidades principales, se requiere implementar ciertas funcionalidades adicionales, como la actualización de totales al agregar productos a una orden, visualización de la cantidad de órdenes registradas por un mesero en un día y detalles del platillo más vendido, entre otros.

Para abordar estos requisitos, se utilizará PostgreSQL como base de datos, implementando los elementos necesarios para cumplir con las especificaciones. Se seguirá un enfoque de diseño que garantice la integridad de los datos, la eficiencia en las consultas y una estructura flexible para futuras expansiones.

En cuanto a la parte de implementación del problema, se propone el desarrollo de una página web que permita consultar la información general de los empleados, incluyendo sus fotografías. Esto implica la creación de una interfaz web intuitiva y segura que acceda a la base de datos para mostrar la información requerida.

Objetivo

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

Plan de trabajo

Para la realización de este proyecto fue fundamental que desde un primer momento clasificáramos y dividiéramos las distintas etapas del proyecto para ir delegando las obligaciones y los tiempos en que debían estar listas las fases del proyecto para concluir con este en tiempo y forma.

Después de leer todos los requerimientos y las necesidades del proyecto, así como las reglas de negocio decidimos como era que íbamos a organizarnos para terminarlo progresivamente.

En nuestro caso utilizamos la herramienta llamada Notion para tener un lugar en donde todos fuéramos capaces de ver la planeación siempre que quisiéramos.

Como podemos ver en la Figura 1 en Notion dividimos el proyecto en tres secciones la primera de ellas con todo lo que concierne al diseño lógico de la base de datos, mientras que la segunda parte implica el desarrollo del modelo físico por ultimo la tercera parte es la realización de una implementación, en nuestro caso elegimos la creación de una pagina web que conectaremos a la base de datos.

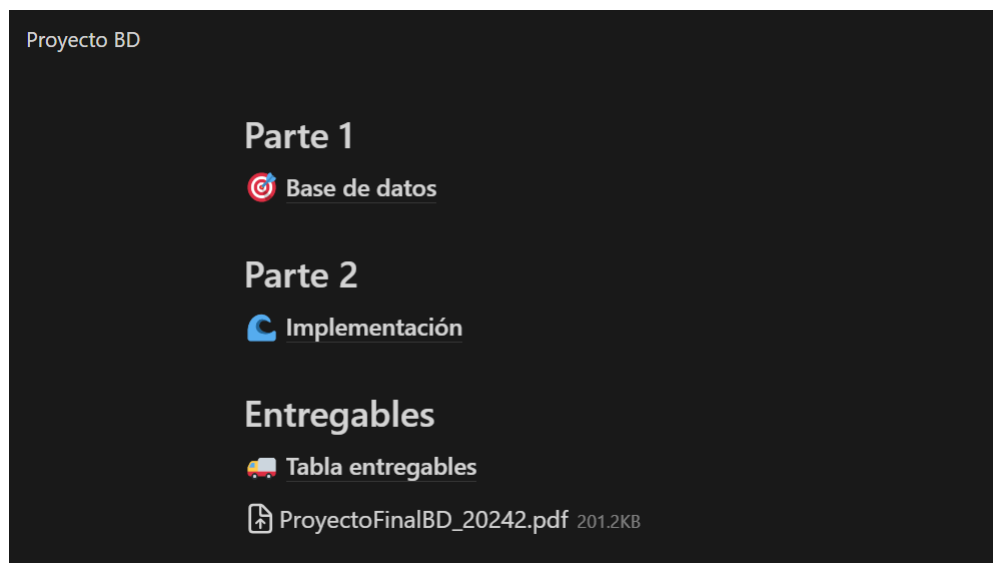


Figura 1: División del proyecto en Notion

Primera Parte (Diseño lógico)

Para el diseño lógico localizamos las actividades que tenemos que realizar, en este caso como vemos en la Figura 2 decidimos que serían cinco actividades las cuales son; el modelo entidad-relación, Modelo Relacional Intermedio, Modelo Relacional, dependencias y normalización.

Posteriormente cada uno de nosotros decidió en cual de las actividades antes mencionadas participaría, finalmente están las fechas de inicio y fin de las actividades.

Tabla 1 - Diseño				
Aa Actividad	Personas	Estado	Fecha inicio	fecha entrega
Modelo entidad relacion	MARTÍNEZ BRAVO DANIELA	Listo	25 de marzo de 2024	8 de abril de 2024
	Aguirre Córdova Omar Gabriel			
Modelo relacional intermedio	Gabriela Suarez Velasco	Listo	25 de marzo de 2024	22 de marzo de 2024
Modelo relacional	Gabriela Suarez Velasco	Listo	25 de marzo de 2024	22 de abril de 2024
	Aguirre Córdova Omar Gabriel			
Dependencias	Diana Pilar	Listo	25 de marzo de 2024	22 de abril de 2024
	Lopez Ramirez Monserrat			
Normalización	MARTÍNEZ BRAVO DANIELA	Listo	25 de marzo de 2024	22 de abril de 2024
	Diana Pilar			
	Lopez Ramirez Monserrat			

Figura 2: Diseño conceptual de la Base de Datos.

Segunda Parte (Modelo físico)

Para el modelo físico las actividades esta vez las propusimos de acuerdo a lo que se nos pide del proyecto como lo es la creación de las tablas de la base de datos del restaurante, cada que se agregue un producto a la orden se actualice los totales (por producto y venta), así como validar que el producto esté disponible, creación de un índice justificando el porque, dado un número de empleado, se muestre la cantidad de órdenes que ha registrado en el día así como el total que se ha pagado por dichas órdenes, si no es un mesero mandar un error, vista que muestre el platillo más vendido, nombre de los productos que están disponibles, vista que contenga la información de una factura de orden, dado una fecha de inicio y fin mostrar el total de número de ventas y el monto total en ese periodo de tiempo. De la misma forma que la anterior tabla cada uno eligió la actividad en la que participaría además de la fecha de inicio y fin de cada actividad.

TABLA 2 - Creación				
Aa Nombre	Personas	Estado	Fecha inicio	Fecha entrega
Crear la BD a partir del modelo relacional.	D Diana Pilar	Listo	22 de abril de 2024	29 de abril de 2024
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.	L Lopez Ramirez Monserrat D Diana Pilar	Listo	29 de abril de 2024	5 de mayo de 2024
Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.	G Gabriela Suarez Velasco M MARTÍNEZ BRAVO DANIEL	Listo	29 de abril de 2024	5 de mayo de 2024
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.	A Aguirre Córdova Omar Gat G Gabriela Suarez Velasco	Listo	29 de abril de 2024	5 de mayo de 2024
Vista que muestre todos los detalles del platillo más vendido.	D Diana Pilar	Listo	6 de mayo de 2024	12 de mayo de 2024
Permitir obtener el nombre de aquellos productos que no estén disponibles.	D Diana Pilar	Listo	6 de mayo de 2024	12 de mayo de 2024
De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una orden.	L Lopez Ramirez Monserrat M MARTÍNEZ BRAVO DANIEL	Listo	6 de mayo de 2024	12 de mayo de 2024
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 período de tiempo. Nota: Con producto se hace referencia a los alimentos y bebidas.	A Aguirre Córdova Omar Gat	Listo	6 de mayo de 2024	12 de mayo de 2024

Figura 3: Modelo físico de la Base de Datos.

De la misma manera dividimos las actividades para realizar la implementación. Entre todos llegamos al acuerdo de realizar la implementación de una pagina web de un restaurante de mariscos que estará conectada a la base de datos, tomando en cuenta el diseño de la pagina como la paleta de colores, las vistas que tendría, así como realizar la conexión pertinente, registrando también los días en que se termino cada actividad.

Tabla 3 - Implementación				
≡ Opcion	Aa Nombre	Estado	Fecha inicio	Fecha entrega
A	Consultar la información general de los empleados, incluyendo su fotografía por medio de una app móvil o web.	Listo	13 de mayo de 2024	17 de mayo de 2024
Front End	Diseño main	Listo	13 de mayo de 2024	17 de mayo de 2024
Front End	Elección de paleta de colores	Listo	13 de mayo de 2024	17 de mayo de 2024
Front End	Diseño vistas	Listo	13 de mayo de 2024	17 de mayo de 2024
Back end	Conexión de la BD a la página web.	Listo	13 de mayo de 2024	17 de mayo de 2024

Figura 4: Implementación de la base de datos.

Finalmente conforme avanzamos en el proyecto fuimos teniendo distintas reuniones ya sean presenciales o por computadora, todas ellas se fueron poniendo en una tabla en donde se explica un poco de lo que se reviso en la reunión al igual que el día que se realizo como se puede ver en la Figura 5.

Tabla 4 - Reuniones		
Aa Reunión	Comentarios	Fecha de la Reunión
Modelo conceptual	Después de terminar el modelo entidad conceptual, revisarlo entre todos los miembros del equipo y hacer ajustes.	8 de abril de 2024
Modelo Relacional e intermedio	Al tener el modelo conceptual, se pasa a revisar el intermedio y finalmente creamos el modelo relacional, revisando que se haya implementado de manera correcta	22 de abril de 2024
Creación de tablas	De tener el modelo relacional creamos las tablas ahora con la sintaxis de sql en postgres, revisando los atributos y las relaciones entre tablas, al igual que las funciones y triggers para atributos calculados.	29 de abril de 2024
Consultas	Reunión en donde a partir de la BD que ya tenemos y los requerimientos que se nos piden, comenzamos a crear índices, consultas y vistas para la información de la BD.	12 de mayo de 2024
Diseño Página web	Entre todos los miembros del equipo escogimos un diseño y que tipo de comida se servirá en el restaurante en este caso de mariscos.	14 de mayo de 2024
Conexión de BD a la página web	Ya que tenemos la página web y la base de datos realizamos la conexión, revisando que todo se visualice de la mejor manera.	17 de mayo de 2024

Figura 5: Reuniones para revisión del avance.

Diseño

Modelo Conceptual

Dado el siguiente caso de estudio proporcionado por el profesor sobre un restaurante:

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 (con hora), 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.

A partir de todo lo descrito anteriormente fue que desarrollamos el modelo conceptual que se puede observar en la Figura 6.

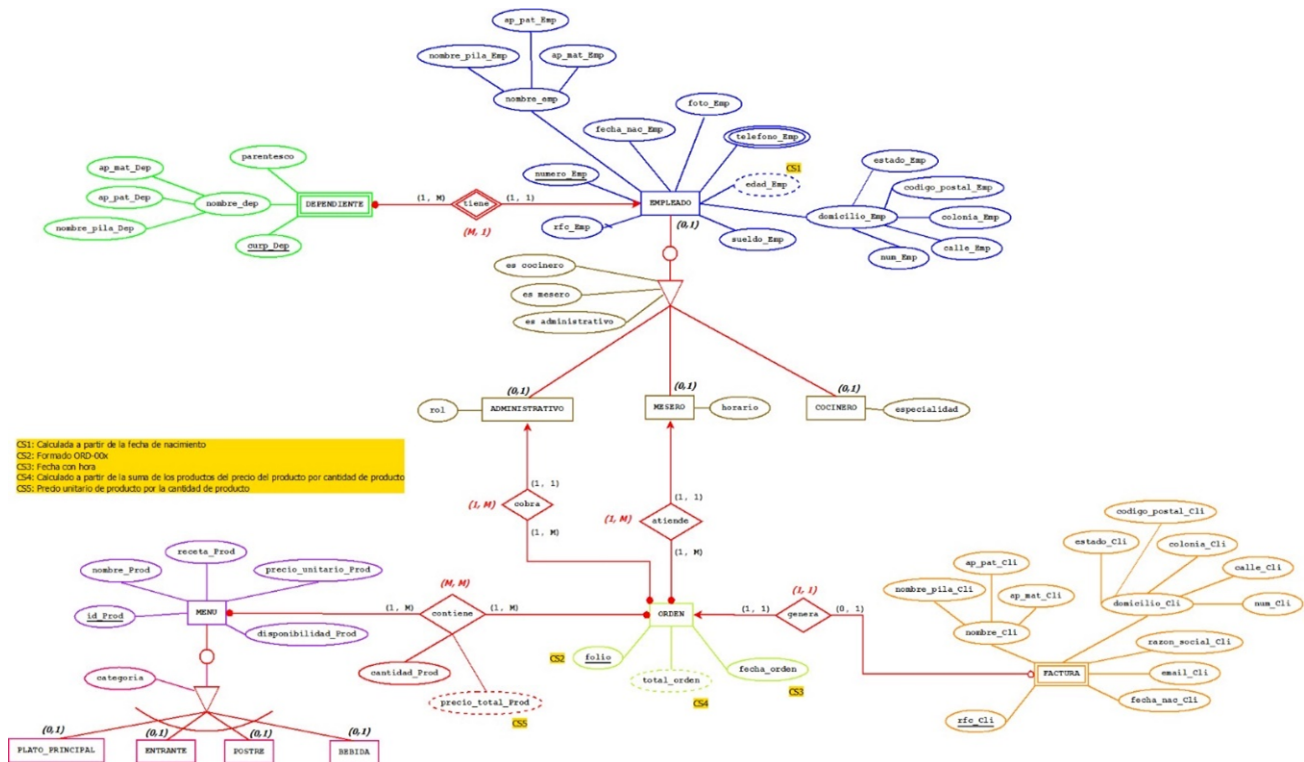


Figura 6: Modelo Conceptual de la Base de Datos de un restaurante

En el modelo conceptual primero analizamos el caso de estudio y obtuvimos las entidades que en este caso son; EMPLEADO con sus subtipos ADMINISTRATIVO, MESERO Y COCINERO, DEPENDIENTE, MENU, ORDEN, FACTURA.

- **EMPLEADO:** La entidad empleado tiene una relación de generalización puesto que como sabemos esta se da al tener atributos en común en distintas entidades, creamos una entidad superior. En este caso dado que las entidades ADMINISTRATIVO, MESERO y COCINERO contaban con los mismos atributos los cuales son; RFC (clave candidata), número de empleado (clave principal), nombre (atributo compuesto), fecha de nacimiento, teléfonos (atributo multivaluado), edad (atributo calculado a partir de la fecha de nacimiento), domicilio (atributo compuesto) y sueldo, decidimos crear el supertipo EMPLEADO con todos los atributos antes mencionados.

La relación con los subtipos es de tipo total, debido a que el empleado debe ser miembro de al menos una de las entidades del subtipo como ADMINISTRATIVO, MESERO o

COCINERO.

- **ADMINISTRATIVO:** Subtipo de la entidad EMPLEADO, con el atributo de rol.
- **MESERO:** Subtipo de la entidad EMPLEADO, con el atributo de horario.
- **COCINERO:** Subtipo de la entidad EMPLEADO, con el atributo de especialidad.
- **DEPENDIENTE:** Es una entidad débil ya que su existencia depende de que exista la entidad EMPLEADO esta tiene como atributos; parentesco, nombre del dependiente (atributo compuesto), curp del dependiente (en una entidad debil en vez de ser clave primaria lo llamamos discriminante).
- **MENU:** En la entidad menú nuevamente se utilizo una generalización, puesto que esta es el supertipo de las entidades PLATO PRINCIPAL, ENTRANTE, POSTRE y BEBIDA. A la relación del supertipo con los subtipos, se le agrega un tipo para la categoría además de agregar una restricción de exclusividad para que cada platillo de la entidad menú solo pertenezca a una de las categorías.

Los atributos de la entidad MENU son; id del producto (clave principal artificial), nombre del producto, receta del producto, precio unitario del producto, disponibilidad del producto.

- **PLATO PRINCIPAL:** Subtipo de la entidad MENU que es excluyente.
- **ENTRANTE:** Subtipo de la entidad MENU que es excluyente.
- **POSTRE:** Subtipo de la entidad MENU que es excluyente.
- **BEBIDA:** Subtipo de la entidad MENU que es excluyente.
- **ORDEN:** Al ser una relación muchos a muchos entre la entidad MENU y ORDEN podemos poner atributos en la relación puesto que al convertir a Modelo Relacional se crea una tabla intermedia por ello en la relación contiene tenemos los atributos; cantidad_prod y precio_total_prod (atributo calculado a partir del precio unitario por la cantidad de productos).

Los atributos de la entidad MENU son; folio (Clave principal), fecha_orden y total_orden (Atributo calculado a partir de la suma de los productos del precio del producto por la cantidad).

- **FACTURA:** La entidad FACTURA es una entidad débil puesto que depende de la existencia de una ORDEN.

Los atributos de esta entidad son; rfc_cli (clave principal), nombre_cli (atributo compuesto por nombre_pila_cli, ap_pat_cli, ap_mat_cli), domicilio_cli (atributo compuesto por estado_cli, codigo_postal_cli, colonia_cli, calle_cli, num_cli), razon_social_cli, email_cli, fecha_nac_cli.

Posteriormente asignamos las relaciones así como su cardinalidad las cuales son:

Entidades DEPENDIENTE, EMPLEADO:

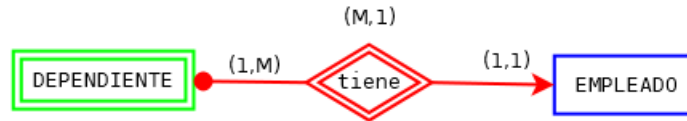


Figura 7: Relación entre las entidades DEPENDIENTE, EMPLEADO

- Un empleado tiene muchos dependientes.
- Un dependiente tiene solo un empleado.

Entidades ADMINISTRATIVO, ORDEN:

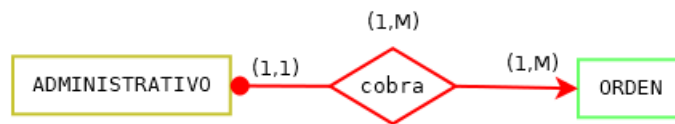


Figura 8: Relación entre las entidades ADMINISTRATIVO, ORDEN

- Una orden la cobra un administrativo.
- Un administrativo cobra muchas ordenes.

Entidades MESERO, ORDEN:

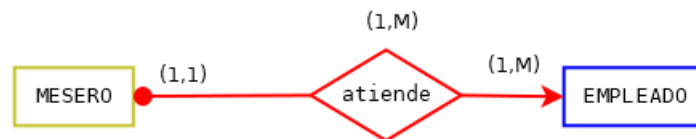


Figura 9: Relación entre las entidades MESERO, ORDEN

- Una orden la atiende un mesero.
- Un mesero atiende muchas ordenes.

Entidades MENU, ORDEN:

- Un producto del menú esta contenido en muchas ordenes.
- Una orden contiene muchos productos del menú.

Entidades ORDEN, FACTURA:

- Puede o no haber una factura que es generada por una orden.
- Una orden genera una factura.

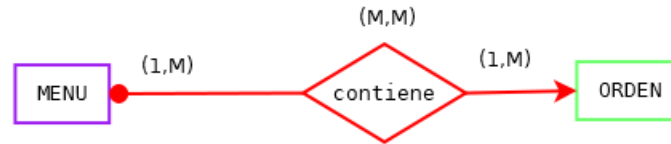


Figura 10: Relación entre las entidades MENU, ORDEN

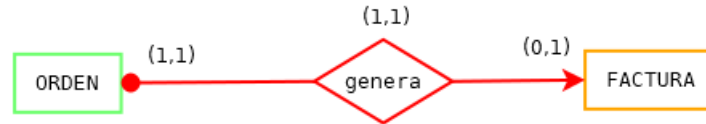


Figura 11: Relación entre las entidades ORDEN, FACTURA

Modelo Relacional Intermedio

Al concluir con el modelo conceptual, comenzamos a convertirlo al modelo Relacional Intermedio utilizando distintas reglas para ello

```
Empleado:{
  num_emp smallint PK not null,
  rfc_emp varchar(13) U not null,
  nombre_pila_emp varchar(150) not null,
  ap_pat_emp varchar(150) not null,
  ap_mat_emp varchar(150) not null,
  fecha_nac_emp date not null,
  foto_emp bytea not null,
  edad_emp smallint C not null,
  estado_emp varchar(50) not null,
  codigo_postal_emp varchar(5) not null,
  colonia_emp varchar(80) not null,
  calle_emp varchar(150) not null,
  num_dom_emp smallint not null,
  sueldo_emp decimal(8) not null,
  cocinero boolean not null,
  mesero boolean not null,
  administrativo boolean not null
}

Administrativo:{
  num_emp smallint FK PK not null,
  rol varchar(60) not null
}

Mesero:{
  num_emp smallint FK PK not null,
  horario varchar(40) not null
}

Cocinero:{
  num_emp smallint FK PK not null,
  especialidad varchar(40) not null
}
```

```
Telefono:{
  telefono varchar(8) PK not null,
  num_emp smallint FK not null,
}

Dependiente:{
  curp_dep varchar(18) D PK not null,
  num_emp smallint FK PK not null,
  nombre_pila_dep varchar(150) not null,
  ap_pat_dep varchar(150) not null,
  ap_mat_dep varchar(150) not null,
  parentesco varchar(20) not null
}

Menu:{
  id_prod smallint PK not null,
  nombre_prod varchar(80) not null,
  receta_prod text not null,
  precio_unitario_prod decimal(6, 2) not null,
  disponibilidad_prod boolean
  categoria char(15) not null
}

Contiene:{
  id_Prod smallint FK PK not null,
  folio_orden varchar(8) FK PK not null,
  cantidad_prod smallint not null,
  precio_total_prod decimal(8, 2) not null
}

Plato_Principal:{
  id_Prod smallint FK PK not null
}

Entrante:{
  id_Prod smallint FK PK not null
}

Postre:{
  id_Prod smallint FK PK not null
}

Bebida:{
  id_Prod smallint FK PK not null
}

Orden:{
  folio_orden varchar(8) PK not null,
  total_orden decimal(8,2) C not null,
  fecha_orden timestamp not null,
  num_Emp smallint FK not null
}
```

```
Factura:{
  rfc_cli varchar(13) D PK not null,
  folio_orden varchar(8) FK PK not null,
  nombre_pila_cli varchar(150) not null,
  ap_pat_cli varchar(150) not null,
  ap_mat_cli varchar(150) not null,
  estado_cli varchar(50) not null,
  codigo_postal_cli varchar(5) not null,
  colonia_cli varchar(80) not null,
  calle_cli varchar(150) not null,
  num_doc_cli smallint not null,
  razon_social_cli varchar(60) not null,
  email_cli varchar(100) not null,
  fecha_nac_cli date not null
}
```

Para este modelo hicimos las consideraciones para pasar del modelo conceptual al modelo relacional. Para ello primero creamos las entidades junto con sus atributos, para los atributos derivados pusimos en las tablas directamente los atributos asociados a ellos.

Los tipos de datos que utilizamos en los atributos son los siguientes:

- **SMALLINT**: Este es un entero de rango pequeño que tiene un tamaño de almacenamiento de 2 bytes con un rango de -32768 a +32767.
- **DECIMAL**: Son aquellos que tienen una posición especificada por el usuario, por ejemplo DECIMAL(8,2) dice que tendrá ocho dígitos de los cuales dos dígitos se encontrarán después del punto decimal. El tamaño de almacenamiento es variable y su rango es de hasta 131072 dígitos antes del punto decimal; hasta 16383 dígitos después del punto decimal.
- **BOOLEAN**: Puede tener los estados falso y verdadero. Los valores validos son TRUE, 't', 'true', 'y', 'yes', '1', FALSE, 'f', 'false', 'n', 'no', '0'.
- **VARCHAR(n)**: Este tipo de dato es utilizado para cadenas y tiene una longitud variable con limite es decir que si VARCHAR(80) y se almacenan cadenas más pequeñas solo se ocupará el espacio necesario para la cadena. La n representa un entero y es el número de caracteres que tendrá la cadena.
- **CHAR(n)**: La diferencia con respecto a varchar es que char es de longitud fija, es decir, si CHAR(8) el espacio que se reservará será el que se especifico aun cuando la cadena que ingrese sea más pequeña.
- **TEXT**: Es aquel que puede guardar una longitud ilimitada variable de caracteres.

Las restricciones que se tomaron en cuenta son las siguientes:

- **PK**: Hace referencia al atributo que será una llave primaria.
- **FK**: Hace referencia al atributo que será una llave foránea.
- **D**: Es el atributo determinante de una entidad débil.

- U: Son los atributos que pudieron ser llaves primarias, llamadas claves candidatas. Es una restricción de unicidad para que no pueda haber duplicados.
- C: Es para los atributos que son calculados a partir de otros atributos.
- NOT NULL: Atributos que no pueden ser nulos.

Para las llaves foráneas seguimos las siguientes reglas a la hora de propagarlas:

- Relación "tiene.^{en}tre entidades DEPENDIENTE y EMPLEADO: Para esta relación dado que DEPENDIENTE es una entidad débil se propaga la llave primaria de la entidad fuerte hacia la débil como FK que también será PK junto con el discriminante.
- Relación cobra.^{en}tre entidades ADMINISTRATIVO y ORDEN: Al ser una relación uno a uno nosotros decidimos que es más conveniente propagar la PK de la entidad ADMINISTRATIVO a ORDEN como una FK.
- Relación "atiende.^{en}tre entidades MESERO y ORDEN: Al ser también una relación uno a uno nosotros decidimos que es más conveniente propagar la PK de la entidad MESERO a ORDEN como una FK.
- Relación contiene.^{en}tre entidades MENU y ORDEN: como es una relación muchos a muchos creamos una nueva tabla llamada CONTIENE en la cual propagamos las PK's de MENU Y ORDEN como FK y PK en la tabla CONTIENE además de agregar los atributos de la relación.
- Relación "genera.^{en}tre entidades FACTURA y ORDEN: Dado que FACTURA es una entidad débil se propaga la PK de la entidad fuerte hacia la débil como FK que también será PK junto con el discriminante.

Para la parte de los Supertipos y los subtipos realizamos lo siguiente:

- Supertipo EMPLEADO subtipos ADMINISTRATIVO, MESERO, COCINERO: Ya que estamos hablando de una generalización para el modelo relacional creamos las tablas ADMINISTRATIVO, MESERO y COCINERO a parte propagando la PK de EMPLEADO como FK y PK en las nuevas tablas con sus respectivos atributos. En la tabla EMPLEADO del supertipo agregamos los atributos administrativo, mesero y cocinero que del tipo boolean que nos indicarán si un empleado tiene más de un puesto.
- Supertipo MENU subtipos ENTRANTE, PLATO_PRINCIPAL, POSTRE, BEBIDA: Para este caso también es una generalización, sin embargo al tener una exclusión solo puede ser parte de una categoría, por ello en la tabla MENU agregamos el atributo de categoría que nos dice a cual de ella pertenece, además de propagar la PK de MENU a los subtipos ENTRANTE, PLATO_PRINCIPAL, POSTRE Y BEBIDA como PK y FK.

Finalmente en la herramienta de ERStudio realizamos el Modelo Relacional con las tablas a partir del Modelo Relacional Intermedio.

Después de ello realizamos la normalización y obtención de dependencia sin embargo, no fue necesario puesto que ya se encontraban normalizadas todas nuestras tablas.

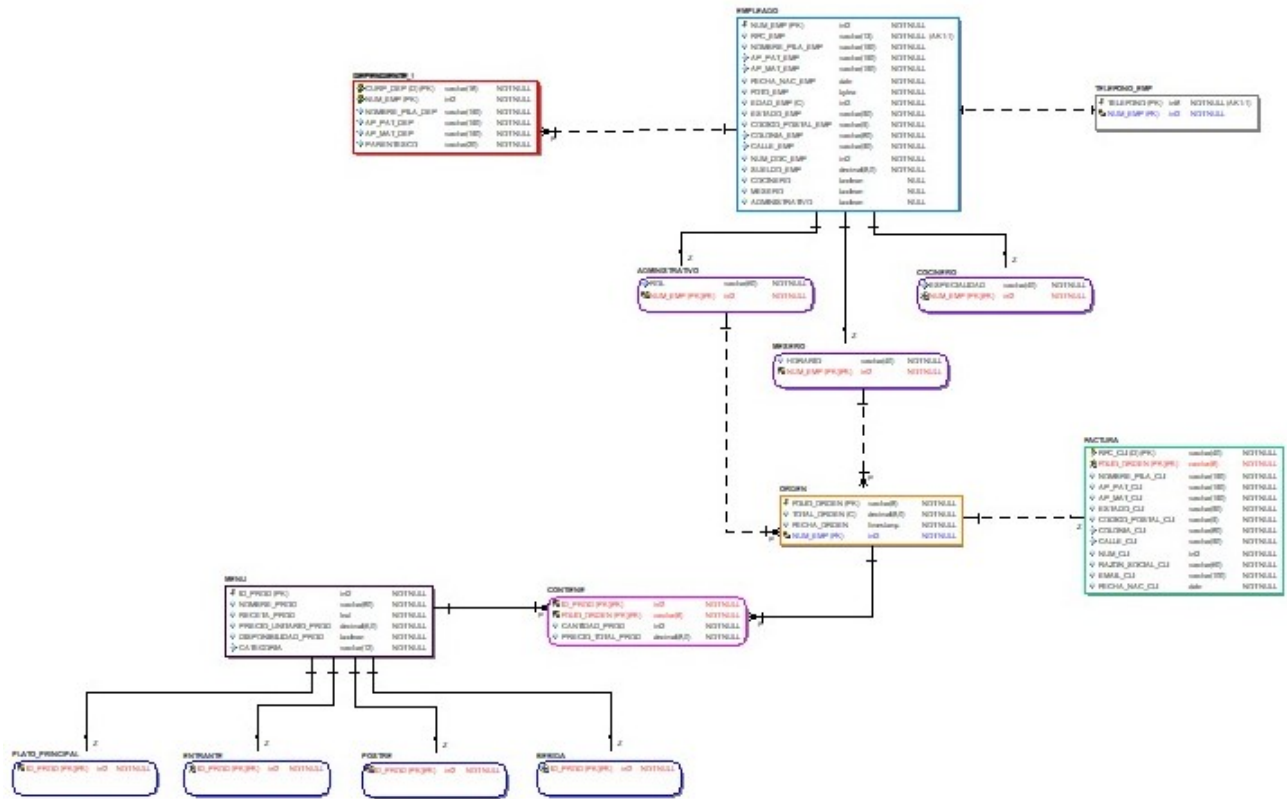


Figura 12: Modelo Relacional.

Dependencias:

Empleado
A→A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R.

Administrativo
A→A,B

Mesero
A→A,B

Cocinero
A→A,B

Teléfono
A→A,B

Dependiente
A→A,B,C,D,E,F,

Menu
A→A,B,C,D,E,F,G,H,I,

Contiene
A→A,B,C,D,E

Plato Principal
A→A

Entrada
A→A

Postre
A→A

```

Bebida
A->A

Orden
A->A,B,C,D

Factura
A->A,B,C,D,E,F,G,H,I,J,K,L,M

Normalización:
1FN: Eliminar Atributos multivalor, PK, Eliminar grupos de repetición.
2FN: Eliminar dependencias parciales
3FN: Transitividad

Las tablas ya están normalizadas.

```

Implementación

Creación de Tablas

A partir del Modelo Relacional que obtuvimos fue que creamos las tablas haciendo uso del DDL (Data Definition Language). Primero creamos la base de datos en postgres con CREATE DATABASE, luego con la sentencia CREATE TABLE creamos las tablas agregando los atributos con sus respectivos tipos de datos y sus restricciones:

Creación de la base de datos llamada 'restaurante' y de la tabla 'empleado'

```

1  CREATE DATABASE restaurante;
2  -- TABLA EMPLEADO
3
4  CREATE TABLE empleado(
5      num_emp          SMALLINT          NOT NULL,
6      rfc_emp          VARCHAR(13)       NOT NULL,
7      nombre_pila_emp  VARCHAR(150)      NOT NULL,
8      ap_pat_emp       VARCHAR(150)      NOT NULL,
9      ap_mat_emp       VARCHAR(150)      NOT NULL,
10     fecha_nac_emp     DATE              NOT NULL,
11     foto_emp          BYTEA            NULL,
12     edad_emp          INT              NULL,
13     estado_emp        VARCHAR(50)       NOT NULL,
14     codigo_postal_emp VARCHAR(5)        NOT NULL,
15     colonia_emp       VARCHAR(80)       NOT NULL,
16     calle_emp         VARCHAR(150)      NOT NULL,
17     num_dom_emp       SMALLINT          NOT NULL,
18     sueldo_emp        DECIMAL(8, 2)     NOT NULL,
19     cocinero          BOOLEAN          NOT NULL,
20     mesero            BOOLEAN          NOT NULL,
21     administrativo    BOOLEAN          NOT NULL,
22     CONSTRAINT empleado_PK PRIMARY KEY (num_emp),
23     CONSTRAINT rfc_emp_U UNIQUE (rfc_emp)
24 );

```

Podemos ver que en este caso solo hay dos CONSTRAINT la primera es la restricción de llave primaria aplicada al num_emp y la segunda es de UNIQUE para que no haya duplicados aplicada a rfc_emp. Por ultimo el ON DELETE RESTRICT y ON UPDATE CASCADE, el primero de ellos se refiere a que si se quiere borrar un registro de una tabla que tiene esta asociada a una llave foránea no se va a permitir, la segunda es para que al actualizar un

registro de una tabla se actualice en todas las demás tablas que ocupen ese registro. El ON DELETE RESTRICT y ON UPDATE CASCADE se lo aplicamos a todas las tablas.

Creación de la tabla 'telefono_emp'

```
1  -- TABLA TELEFONO
2
3  CREATE TABLE telefono_emp(
4  telefono VARCHAR(8) NOT NULL,
5  num_emp SMALLINT NOT NULL,
6  CONSTRAINT telefono_emp_PK PRIMARY KEY (telefono),
7  CONSTRAINT telefono_empleado_FK FOREIGN KEY (num_emp)
8  REFERENCES empleado(num_emp) ON DELETE RESTRICT
9  ON UPDATE CASCADE
10 );
```

Creación de la tabla 'administrativo'

```
1  -- TABLA ADMINISTRADOR
2
3  CREATE TABLE administrativo(
4  rol VARCHAR(60) NOT NULL,
5  num_emp_administrativo SMALLINT NOT NULL,
6  CONSTRAINT administrativo_PK PRIMARY KEY (num_emp_administrativo),
7  CONSTRAINT administrativo_empleado_FK FOREIGN KEY (
8  num_emp_administrativo)
9  REFERENCES empleado(num_emp) ON DELETE RESTRICT
10 ON UPDATE CASCADE
);
```

Creación de la tabla 'mesero'

```
1  -- TABLA MESERO
2
3  CREATE TABLE mesero(
4  horario VARCHAR(40) NOT NULL,
5  num_emp_mesero SMALLINT NOT NULL,
6  CONSTRAINT mesero_PK PRIMARY KEY (num_emp_mesero),
7  CONSTRAINT mesero_empleado_FK FOREIGN KEY (num_emp_mesero)
8  REFERENCES empleado(num_emp) ON DELETE RESTRICT
9  ON UPDATE CASCADE
10 );
```

Creación de la tabla 'cocinero'

```
1  -- TABLA COCINERO
2
3  CREATE TABLE cocinero(
4  especialidad VARCHAR(40) NOT NULL,
5  num_emp_cocinero SMALLINT NOT NULL,
6  CONSTRAINT cocinero_PK PRIMARY KEY (num_emp_cocinero),
7  CONSTRAINT cocinero_empleado_FK FOREIGN KEY (num_emp_cocinero)
8  REFERENCES empleado(num_emp) ON DELETE RESTRICT
9  ON UPDATE CASCADE
10 );
```

Creación de la tabla 'dependiente'

```
1  -- TABLA DEPENDIENTE
2
3  CREATE TABLE dependiente(
4  curp_dep      VARCHAR(18)      NOT NULL,
5  num_emp       SMALLINT         NOT NULL,
6  nombre_pila_dep VARCHAR(150)   NOT NULL,
7  ap_pat_dep    VARCHAR(150)     NOT NULL,
8  ap_mat_dep    VARCHAR(150)     NOT NULL,
9  parentesco    VARCHAR(20)      NOT NULL,
10 CONSTRAINT dependiente_PK PRIMARY KEY (curp_dep, num_emp),
11 CONSTRAINT dependiente_empleado_FK FOREIGN KEY (num_emp)
12 REFERENCES empleado(num_emp) ON DELETE RESTRICT
13 ON UPDATE CASCADE
14 );
```

Creación de la tabla 'menu'

```
1  -- TABLA MENU
2
3  CREATE TABLE menu(
4  id_prod       SMALLINT         NOT NULL,
5  nombre_prod   VARCHAR(80)      NOT NULL,
6  receta_prod   TEXT             NOT NULL,
7  precio_unitario_prod DECIMAL(6, 2) NOT NULL,
8  disponibilidad_prod BOOLEAN     NOT NULL,
9  categoria     CHAR(15)         NOT NULL,
10 CONSTRAINT menu_PK PRIMARY KEY (id_prod)
11 );
```

Creación de la tabla 'entrante'

```
1  -- TABLA ENTRANTE
2
3  CREATE TABLE entrante(
4  id_prod SMALLINT NOT NULL,
5  CONSTRAINT entrante_PK PRIMARY KEY (id_prod),
6  CONSTRAINT entrante_menu_FK FOREIGN KEY (id_prod)
7  REFERENCES menu(id_prod) ON DELETE RESTRICT
8  ON UPDATE CASCADE
9  );
```

Creación de la tabla 'plato_principal'

```
1  -- TABLA PLATO PRINCIPAL
2
3  CREATE TABLE plato_principal(
4  id_prod SMALLINT NOT NULL,
5  CONSTRAINT plato_principal_PK PRIMARY KEY (id_prod),
6  CONSTRAINT plato_principal_menu_FK FOREIGN KEY (id_prod)
7  REFERENCES menu(id_prod) ON DELETE RESTRICT
8  ON UPDATE CASCADE
9  );
```

Creación de la tabla 'postre'

```
1  -- TABLA POSTRE
2
3  CREATE TABLE postre(
4  id_prod SMALLINT NOT NULL,
5  CONSTRAINT postre_PK PRIMARY KEY (id_prod),
6  CONSTRAINT postre_menu_FK FOREIGN KEY (id_prod)
7  REFERENCES menu(id_prod) ON DELETE RESTRICT
8  ON UPDATE CASCADE
9  );
```

Creación de la tabla 'bebida'

```
1  -- TABLA BEBIDA
2
3  CREATE TABLE bebida(
4  id_prod SMALLINT NOT NULL,
5  CONSTRAINT bebida_PK PRIMARY KEY (id_prod),
6  CONSTRAINT bebida_menu_FK FOREIGN KEY (id_prod)
7  REFERENCES menu(id_prod) ON DELETE RESTRICT
8  ON UPDATE CASCADE
9  );
```

Creación de la tabla 'orden'

```
1  -- TABLA ORDEN
2
3  CREATE TABLE orden(
4  folio_orden VARCHAR(8) NOT NULL,
5  total_orden DECIMAL(8, 2) NULL,
6  fecha_orden TIMESTAMP NOT NULL,
7  num_emp_administrativo SMALLINT NOT NULL,
8  num_emp_mesero SMALLINT NOT NULL,
9  CONSTRAINT orden_PK PRIMARY KEY (folio_orden),
10 CONSTRAINT orden_administrativo_FK FOREIGN KEY (
11   num_emp_administrativo)
12 REFERENCES administrativo(num_emp_administrativo) ON DELETE
13   RESTRICT
14   ON UPDATE CASCADE,
15 CONSTRAINT orden_mesero_FK FOREIGN KEY (num_emp_mesero)
16 REFERENCES mesero(num_emp_mesero) ON DELETE RESTRICT
17   ON UPDATE CASCADE
18   );
```

Creación de la tabla 'contiene'

```

1  -- TABLA CONTIENE
2
3  CREATE TABLE contiene(
4  id_prod          SMALLINT          NOT NULL,
5  folio_orden      VARCHAR(8)        NOT NULL,
6  cantidad_prod    SMALLINT          NOT NULL,
7  precio_total_prod DECIMAL(8, 2)     NULL,
8  CONSTRAINT contiene_PK PRIMARY KEY (id_prod, folio_orden),
9  CONSTRAINT contiene_menu_FK FOREIGN KEY (id_prod)
10 REFERENCES menu(id_prod) ON DELETE RESTRICT
11 ON UPDATE CASCADE,
12 CONSTRAINT contiene_orden_FK FOREIGN KEY (folio_orden)
13 REFERENCES orden(folio_orden) ON DELETE RESTRICT
14 ON UPDATE CASCADE
15 );

```

Creación de la tabla 'factura'

```

1  -- TABLA FACTURA
2
3  CREATE TABLE factura(
4  rfc_cli          VARCHAR(13)        NOT NULL,
5  folio_orden      VARCHAR(8)         NOT NULL,
6  nombre_pila_cli  VARCHAR(150)       NOT NULL,
7  ap_pat_cli       VARCHAR(150)       NOT NULL,
8  ap_mat_cli       VARCHAR(150)       NOT NULL,
9  estado_cli       VARCHAR(50)        NOT NULL,
10 codigo_postal_cli VARCHAR(5)         NOT NULL,
11 colonia_cli      VARCHAR(80)        NOT NULL,
12 calle_cli        VARCHAR(50)        NOT NULL,
13 num_dom_cli      SMALLINT           NOT NULL,
14 razon_social_cli  VARCHAR(60)       NOT NULL,
15 email_cli        VARCHAR(100)       NOT NULL,
16 fecha_nac_cli    DATE               NOT NULL,
17 CONSTRAINT factura_PK PRIMARY KEY (rfc_cli, folio_orden),
18 CONSTRAINT factura_orden_FK FOREIGN KEY (folio_orden)
19 REFERENCES orden(folio_orden) ON DELETE RESTRICT
20 ON UPDATE CASCADE
21 );

```

Creación de Funciones y Triggers

Para los atributos calculados creamos distintas funciones en postgres y para la actualización de estos atributos a la hora de insertar valores en las tablas creamos triggers.

La primera función que creamos es la de calcular edad”, en ella de entrada tomamos la fecha_nac_emp del tipo DATE, luego RETURNS INT indica que se devolverá un número entero y \$\$ señala que se comenzara con un bloque de código. Con DECLARE declaramos las variables que utilizaremos dentro de la función que en este caso es edad_emp del tipo INT, después con el BEGIN después ponemos el código que ejecutaremos en la función en el que con SELECT DATE_PART seleccionamos una parte específica de la fecha que en este caso sería el año, con age() es que obtenemos la diferencia entre la fecha que se nos dio y la actual, como resultado se devolverá la edad_emp, al final solo especifica que se utilizo el lenguaje plpgsql.

Función para calcular la edad

```
1  -- FUNCION PARA CALCULAR LA EDAD
2
3  CREATE OR REPLACE FUNCTION calcular_edad(fecha_nac_emp DATE)
4  RETURNS INT AS $$
5  DECLARE
6  edad_emp INT;
7  BEGIN
8  SELECT DATE_PART('year', age(fecha_nac_emp)) INTO edad_emp;
9  RETURN edad_emp;
10 END;
11 $$ LANGUAGE plpgsql;
```

Con la siguiente función asignamos (con :=) a la nueva fila que vamos a insertar el valor que calculamos de la función calcular_edad() le mandamos el valor fecha_nac_emp que tenemos en la nueva fila y esta función nos devuelve la fila ya con el valor actualizado.

Función para actualizar la edad

```
1  -- FUNCION PARA ACTUALIZAR LA EDAD
2
3  CREATE OR REPLACE FUNCTION actualizar_edad()
4  RETURNS TRIGGER AS $$
5  BEGIN
6  NEW.edad_emp := calcular_edad(NEW.fecha_nac_emp);
7  RETURN NEW;
8  END;
9  $$ LANGUAGE plpgsql;
```

Por ultimo creamos un trigger para que la edad_emp se calcule y actualice cada vez que insertamos o actualizamos un registro de la tabla EMPLEADO.

Trigger para actualizar la edad

```
1  -- TRIGGER PARA ACTUALIZAR LA EDAD AL INSERTAR UN EMPLEADO
2
3  CREATE TRIGGER trigger_actualizar_edad
4  BEFORE INSERT OR UPDATE ON empleado
5  FOR EACH ROW EXECUTE FUNCTION actualizar_edad();
```

Insertión de algunos datos en la tabla empleado para verificar que se calcula correctamente el atributo edad_emp.

num_emp	rfc_emp	nombre_pila_emp	ap_pat_emp	ap_mat_emp	fecha_nac_emp	foto_emp	edad_emp
1	ABC123456789	Juan	Pérez	González	1990-05-15		34
2	DEF987654321	María	López	Martínez	1985-10-20		38
3	GHI567890123	Pedro	Ramírez	Hernández	1988-03-10		36
4	JKL321098765	Ana	García	Pérez	1992-07-25		31
5	MNO543216789	Luis	Fernández	Sánchez	1995-01-30		29
6	PQR987654321	Laura	Díaz	Gómez	1991-09-05		32
7	STU543216789	Carlos	Martínez	Rodríguez	1989-12-12		34
8	VWX123456789	Paula	Hernández	López	1993-04-18		31
9	YZA789012345	Javier	Pérez	Fernández	1994-08-20		29
10	BCD345678901	Elena	Gómez	Sánchez	1996-02-28		28
11	EFG678901234	Ricardo	Martínez	García	1987-11-15		36
12	HIJ456789012	Mónica	López	Ramírez	1990-06-10		33
13	KLM789012345	Fernando	González	Hernández	1986-01-05		38

Figura 13: Datos de la tabla empleado.

En la función declaramos la variable `total_orde_actual` del tipo `DECIMAL(8,2)` que será la variable en la que guardaremos el calculo que hicimos con `COALESCE(SUM(precio_total_prod), 0)` lo que hace es regresar el primer valor no nulo de sumar el `precio_total_prod`, si resulta ser 0 devuelve el cero en vez de devolver nulo, esto donde en la tabla contiene el `folio_orden` y el `folio_orden` que se quiere modificar son iguales. Por ultimo actualizamos la tabla orden para que el atributo `total_orden` tenga el resultado del calculo que realizamos.

Función para calcular el total_orden

```

1  -- FUNCION PARA CALCULAR EL ATRIBUTO TOTAL_ORDEN
2
3  CREATE OR REPLACE FUNCTION actualizar_total_orden() RETURNS
4  TRIGGER AS $$
5  DECLARE
6  total_orden_actual DECIMAL(8, 2);
7  BEGIN
8  -- Calcular el nuevo total de la orden
9  SELECT COALESCE(SUM(precio_total_prod), 0) INTO
10  total_orden_actual FROM contiene WHERE folio_orden = NEW.
11  folio_orden;
12
13  -- Actualizar el total de la orden en la tabla orden
14  UPDATE orden SET total_orden = total_orden_actual WHERE
15  folio_orden = NEW.folio_orden;
16
17  RETURN NEW;
18  END;
19  $$ LANGUAGE plpgsql;

```

Trigger para actualizar total orden

```

1  -- TRIGGER PARA ACTUALIZAR DESPUES DE INSERTAR EN LA TABLA
2  CONTIENE
3
4  CREATE TRIGGER trigger_actualizar_total_orden
5  AFTER INSERT OR UPDATE ON contiene
6  FOR EACH ROW
7  EXECUTE FUNCTION actualizar_total_orden();

```

Inserción de algunos datos en la tabla orden para verificar que se calcula correctamente el

atributo total_orden.

restaurant=#	SELECT *FROM orden;				
folio_orden	total_orden	fecha_orden	num_emp_administrativo	num_emp_mesero	
ORD-001	240.00	2024-05-18 17:24:11.906598	1	6	
ORD-002	180.00	2024-05-18 17:24:11.906598	2	7	
ORD-003	600.00	2024-05-18 17:24:11.906598	3	8	
ORD-004	500.00	2024-05-18 17:24:11.906598	4	9	
ORD-005	150.00	2024-05-18 17:24:11.906598	5	10	

Figura 14: Datos de la tabla orden.

En esta función lo que hacemos es revisar si el producto esta disponible con el atributo disponibilidad_prod en la tabla MENU si lo esta asignamos a la nueva fila de precio_total_prod el resultado de multiplicar el precio_unitario_prod (como no se encuentra en la misma tabla hacemos una consulta a la tabla menu) por la cantidad_prod y devolvemos este nuevo valor que calculamos, si el producto no esta disponible mandamos un mensaje de error.

Función para calcular y actualizar el precio_total_prod.

```

1  -- FUNCION PARA ACTUALIZAR EL PRECIO_TOTAL_PROD
2
3  CREATE OR REPLACE FUNCTION calcular_precio_total() RETURNS
4  TRIGGER AS $$
5  BEGIN
6  IF EXISTS (SELECT 1 FROM menu WHERE id_prod = NEW.id_prod AND
7             disponibilidad_prod) THEN
8  NEW.precio_total_prod = NEW.cantidad_prod * (SELECT
9             precio_unitario_prod FROM menu WHERE id_prod = NEW.id_prod);
10 RETURN NEW;
11 ELSE
12 RAISE EXCEPTION 'El producto no esta disponible en el menu.';
13 END IF;
14 END;
15 $$ LANGUAGE plpgsql;

```

Trigger para actualizar el precio_total_prod.

```

1  -- TRIGGER PARA ACTUALIZAR CADA QUE SE INSERTE UN REGISTRO
2
3  CREATE TRIGGER trigger_actualizar_precio_total
4  BEFORE INSERT OR UPDATE ON contiene
5  FOR EACH ROW
6  EXECUTE FUNCTION calcular_precio_total();

```

Insertión de algunos datos en la tabla contiene para verificar que se calcula correctamente el atributo precio_total_prod.

```
restaurante=# SELECT *FROM contiene;
```

id_prod	folio_orden	cantidad_prod	precio_total_prod
1	ORD-001	2	240.00
2	ORD-002	1	180.00
3	ORD-003	3	600.00
4	ORD-004	2	500.00
5	ORD-005	1	150.00

Figura 15: Datos de la tabla contiene.

Ya que se nos pide que 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.

Creamos una función para esta consulta primero debe recibir el `num_emp` que vamos a verificar en este caso le llamamos `emp_id` que será del tipo `SMALLINT` luego declaramos las variables `id_empleado` `SMALLINT`, `cantidad_ordenes` `INT`, `total_pagado` `NUMERIC` que son los atributos que se nos pide mostrar del empleado, luego verificamos si es un mesero puesto que en caso de no serlo mandará un error, si el empleado es un mesero hacemos una consulta para obtener el `num_emp` de la tabla empleado que le dimos el alias `e`, el número de ordenes que ha atendido con el código `CAST(COUNT(o.folio_orden) AS INT)` y el total de todas las ordenes que tomo con el código `COALESCE(SUM(o.total_orden), 0)` esto lo obtiene la tabla que resulta después de hacer un `left join` entre la tabla empleado y la tabla orden donde `num_emp` sea igual y la `fecha_orden` del tipo `DATE` que sea igual a la fecha actual con `CURRENT_DATE`, todo esto donde el `num_emp` sea igual al `emp_id` que queremos consultar, el resultado se va a agrupar de acuerdo a el `num_emp`.

Función para obtener las ordenes y total de un mesero.

```

1  -- FUNCION PARA OBTENER LAS ORDENES Y TOTAL ME UN MESERO DADO UN
2  ID DE UN EMPLEADO
3  CREATE OR REPLACE FUNCTION obtener_ordenes_y_total(emp_id
4  SMALLINT)
5  RETURNS TABLE (
6  id_empleado SMALLINT,
7  cantidad_ordenes INT,
8  total_pagado NUMERIC
9  ) AS $$
10 BEGIN
11 -- Verificar si el empleado es un mesero
12 IF EXISTS (
13 SELECT 1
14 FROM empleado e
15 WHERE e.num_emp = emp_id AND e.mesero = TRUE
16 ) THEN
17 -- Si es mesero, devolver la cantidad de ordenes y el total
18 -- pagado
19 RETURN QUERY
20 SELECT
21 e.num_emp AS id_empleado,
22 CAST(COUNT(o.folio_orden) AS INT) AS cantidad_ordenes,
23 COALESCE(SUM(o.total_orden), 0) AS total_pagado
24 FROM
25 empleado e
26 LEFT JOIN
27 orden o ON e.num_emp = o.num_emp_mesero AND o.fecha_orden::DATE =
28 CURRENT_DATE
29 WHERE
30 e.num_emp = emp_id
31 GROUP BY
32 e.num_emp;
33 ELSE
34 -- Si no es mesero, levantar una excepcion
35 RAISE EXCEPTION 'El empleado con ID % no es un mesero.', emp_id;
36 END IF;
37 END;
38 $$ LANGUAGE plpgsql;

```

Ya que se nos pide que se genere una vista de manera automática que contenga información necesaria para asemejarse a una factura de una orden.

Para que la vista se actualice constantemente debemos crear una función que en este caso de va a llamar `refresh_factura_orden_view()` con ella refrescamos la vista para que cambie conforme se van modificando los datos, para ello también hacemos uso de un trigger que se ejecutará después de haber insertado, borrado o actualizado algún registro en la tabla orden, entonces llamará a la función `refresh_factura_orden_view()` para actualizar a la vista materializada que se explicará más adelante como fue creada.

Función para actualizar la factura_orden.

```

1  -- FUNCION PARA LA VISTA CON EL DETALLE DE LA ORDEN
2
3  CREATE OR REPLACE FUNCTION refresh_factura_orden_view()
4  RETURNS TRIGGER AS $$
5  BEGIN
6  REFRESH MATERIALIZED VIEW factura_orden;
7  RETURN NULL;
8  END;
9  $$ LANGUAGE plpgsql;

```

Trigger para refrescar la vista materializada factura_orden.

```

1  -- TRIGGER PARA REFRESCAR LA VISTA DE FACTURA_ORDEN
2
3  CREATE TRIGGER trg_refresh_factura_orden_view
4  AFTER INSERT OR UPDATE OR DELETE
5  ON orden
6  FOR EACH STATEMENT
7  EXECUTE FUNCTION refresh_factura_orden_view();

```

Creación de un Índice

Para la creación del índice nosotros elegimos crear un índice sencillo para un solo campo de la tabla MENU, que en este caso es el de disponibilidad_pro, esto ya que en diversas consultas como la de obtener el nombre de aquellos productos que no estén disponibles o la función calcular_precio_total() que valida la disponibilidad del producto sino manda un mensaje de error es más rápida la forma en que se hará la búsqueda y consulta. Este índice es de tipo B-Tree ya que además de ser uno de los mejores para optimizar y eficientar la base de datos es el que postgres generará por defecto al crear un índice.

Creación del índice en disponibilidad_prod.

```

1  -- INDICE EN LA TABLA MENU EN LA COLUMNA DISPONIBILIDAD_PROD
2
3  CREATE INDEX disponible
4  ON menu (disponibilidad_Prod);

```

Consultas y Vistas de la base de datos

- 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.

Para esta consulta se creo la función obtener_ordenes_y_total(emp_id SMALLINT) de la que ya se explico su funcionamiento anteriormente ahora para poder visualizar los datos que se nos están pidiendo debemos de mandar a llamar a la función pasandole el num_emp que vamos a evaluar obtener_ordenes_y_total(6::SMALLINT) le ponemos el ::SMALLINT para convertir el tipo de dato explícitamente a un tipo SMALLINT.

Consulta a una función para obtener la cantidad y el total de las ordenes de un mesero.

```

1      -- CONSULTA PARA OBTENER LA CANTIDAD Y EL TOTAL DE ORDENES
2
3      SELECT * FROM obtener_ordenes_y_total(6::SMALLINT);

```

Implementación en la base de datos cuando el num_emp si es un mesero.

```

restaurante=# SELECT * FROM obtener_ordenes_y_total(6::SMALLINT);
 id_empleado | cantidad_ordenes | total_pagado
-----+-----+-----
      6 |          1 |      240.00
(1 fila)

```

Figura 16: Consulta para el total y ordenes de un mesero.

Implementación en la base de datos cuando el num_emp no es un mesero.

```

restaurante=# SELECT * FROM obtener_ordenes_y_total(1::SMALLINT);
ERROR:  El empleado con ID 1 no es un mesero.
CONTEXTO:  función PL/pgSQL obtener_ordenes_y_total(smallint) en la línea 25 en RAISE

```

Figura 17: Consulta para el total y ordenes cuando no es un mesero.

- Vista que muestre todos los detalles del platillo más vendido.

Creamos una vista llamada platillo_mas_vendido que tendrá la consulta de seleccionar todos los datos de la tabla MENU donde el id sea igual a el resultado de otra consulta, en esta subconsulta seleccionamos el atributo id_prod de la tabla contiene y los agrupamos por el id_prod para tener un producto diferente por fila, después se van a ordenar de forma descendente para que el de mayor valor sea el primero y los ordenamos a partir de la suma de cantidad de productos vendidos, por ultimo con LIMIT 1 seleccionamos el producto que se encuentra hasta arriba de la tabla que será el más vendido.

Vista del platillo mas vendido.

```

1      -- CREACION DE VISTA PARA EL PLATILLO MAS VENDIDO
2
3      CREATE VIEW platillo_mas_vendido AS
4      SELECT *
5      FROM menu
6      WHERE id_prod = (
7      SELECT id_prod
8      FROM contiene
9      GROUP BY id_prod
10     ORDER BY SUM(cantidad_prod) DESC
11     LIMIT 1
12 );

```

Vista del platillo mas vendido.

```

1      -- PARA OBTENER LOS DATOS DE LA VISTA
2
3      SELECT * FROM platillo_mas_vendido;

```

Implementación de la vista del platillo mas vendido

```

restaurante=# SELECT * FROM platillo_mas_vendido;
 id_prod | nombre_prod | receta_prod | precio_unitario_prod | disponibilidad_prod | categoria
-----
      3 | Pulpo a la Gallega | Pulpo cocido y aderezado con aceite de oliva, pimentón y sal. |      200.00 | t
| Plato principal
(1 fila)

```

Figura 18: Platillo más vendido.

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

Para obtener el nombre de los productos que no están disponibles seleccionamos el nombre_prod de la tabla MENU que cumplan con la condición de que en el atributo disponibilidad_prod = false.

Nombre de los productos que no están disponibles.

```

1      -- CONSULTA PARA LOS PRODUCTOS QUE NO ESTEN DISPONIBLES
2
3      SELECT nombre_prod
4      FROM menu
5      WHERE disponibilidad_prod = false;

```

Implementación cuando todos son disponibles.

```

restaurante=# SELECT nombre_prod
restaurante=# FROM menu
restaurante=# WHERE disponibilidad_prod = false;
 nombre_prod
-----
(0 filas)

```

Figura 19: Productos disponibles.

Implementación cuando un producto no esta disponible.

```

restaurante=# SELECT nombre_prod
restaurante=# FROM menu
restaurante=# WHERE disponibilidad_prod = false;
 nombre_prod
-----
Ceviche de Camarón
(1 fila)

```

Figura 20: Productos no disponibles.

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

Para esto primero creamos una vista materializada para que los datos y la vista se guarden en disco. Para la vista materializada se seleccionan las columnas folio_orden, rfc_cli, nombre_pila_cli, ap_pat_cli, ap_mat_cli, estado_cli, codigo_postal_cli, colonia_cli, calle_cli, num_dom_cli, razon_social_cli, email_cli, fecha_nac_cli, fecha_orden, nombre_prod, cantidad_prod, precio_unitario_prod, precio_total_prod y total_orden que pertenecen a las tablas orden, contiene, factura y menu, por ello realizamos un join de la tabla orden con la tabla factura y la tabla contiene y de la tabla contiene con la tabla menu.

Posteriormente para que se actualice lo hacemos con la función refresh_factura_orden_view() que ejecuta la sentencia para REFRESH esta se ejecutará cada vez que alguien actualice, borre o inserte un registro con ayuda del trigger que manda a llamar a la función.

Creación de la vista materializada para el detalle de la orden.

```
1  -- VISTA MATERIALIZADA PARA EL DETALLE DE LA ORDEN
2
3  CREATE MATERIALIZED VIEW factura_orden AS
4  SELECT
5  o.folio_orden,
6  f.rfc_cli,
7  f.nombre_pila_cli || ' ' || f.ap_pat_cli || ' ' || f.ap_mat_cli
   AS nombre_cliente,
8  f.estado_cli,
9  f.codigo_postal_cli,
10 f.colonia_cli,
11 f.calle_cli,
12 f.num_dom_cli,
13 f.razon_social_cli,
14 f.email_cli,
15 f.fecha_nac_cli,
16 o.fecha_orden AS fecha_orden,
17 m.nombre_prod,
18 ctd.cantidad_prod,
19 m.precio_unitario_prod,
20 ctd.precio_total_prod AS subtotal_producto,
21 o.total_orden AS total_factura
22 FROM
23 orden o
24 JOIN
25 factura f ON o.folio_orden = f.folio_orden
26 JOIN
27 contiene ctd ON o.folio_orden = ctd.folio_orden
28 JOIN
29 menu m ON ctd.id_prod = m.id_prod;
```

Consulta la vista materializada.

```
1  -- CONSULTA DE LA VISTA MATERIALIZADA QUE CREAMOS
2
3  SELECT * FROM factura_orden;
```

Implementación de la instrucción para visualizar los datos de la vista materializada factura_orden.

```

restaurantes=# SELECT * FROM factura_orden;
folio_orden | rfc_cli | nombre_cliente | estado_cli | codigo_postal_cli | colonia_cli | calle_cli | num_dom_cli | razon_social_cli
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
ORD-001 | ABC123456789 | Juan González Pérez | Ciudad de México | 12345 | Centro | Calle 123 | 456 | Juan González S.A. de C.V.
.00 | 240.00 | 240.00
ORD-002 | XYZ987654321 | María López Martínez | Guadalajara | 54321 | Reforma | Avenida Principal | 789 | López Martínez Hermanos
.00 | 180.00 | 180.00
ORD-003 | DEF456789123 | Pedro Sánchez García | Monterrey | 67890 | Independencia | Calle Libertad | 1011 | Pedro Sánchez Construcciones
.00 | 600.00 | 600.00
ORD-004 | GHI654321987 | Ana Martínez Rodríguez | Puebla | 98765 | Juárez | Calle 456 | 1213 | Ana Rodríguez Arquitectura
.00 | 500.00 | 500.00
ORD-005 | JKL321987654 | Luis Hernández Fernández | Toluca | 23456 | San Miguel | Calle Flores | 1415 | Luis Fernández Abogados
.00 | 150.00 | 150.00
(5 filas)

```

Figura 21: Vista factura_orden materializada.

```

| email_cli | fecha_nac_cli | fecha_orden
|-----|-----|-----|
| juan@example.com | 1990-05-15 | 2024-05-18 18:21:51.075215 | Ceviche de Camarón | 2 | 120
| maria@example.com | 1988-12-10 | 2024-05-18 18:21:51.075215 | Filete de Pescado a la Veracruzana | 1 | 180
| pedro@example.com | 1995-08-20 | 2024-05-18 18:21:51.075215 | Pulpo a la Gallega | 3 | 200
| ana@example.com | 1985-03-25 | 2024-05-18 18:21:51.075215 | Paella Marinera | 2 | 250
| luis@example.com | 1992-10-05 | 2024-05-18 18:21:51.075215 | Ensalada de Mariscos | 1 | 150

```

Figura 22: Vista factura_orden materializada.

- 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.

Para esta consulta obtenemos lo que resulte de contar el numero de ordenes con COUNT() y el total de las ordenes con SUM() de los atributos folio_orden y total_orden de la tabla orden, pero solo se mostrarán los que cumplan con que en la fecha_orden sean mayores a la fecha de inicio del periodo y menores a la fecha fin del periodo.

A partir de dos fechas obtener el numero total y monto de las ventas.

```

1      -- DADAS DOS FECHAS SE REGRESA EL NUMERO TOTAL Y MONTO DE LAS
2      VENTAS
3      SELECT
4      COUNT(o.folio_orden) AS total_ventas,
5      SUM(o.total_orden) AS monto_total_ventas
6      FROM
7      orden o
8      WHERE
9      o.fecha_orden >= '2024-01-01' -- fecha de inicio del periodo
10     AND o.fecha_orden <= '2024-12-31'; -- fecha de fin del periodo

```

Implementación al dar dos fechas una de inicio y otra de fin para ver el numero y monto de ventas en ese periodo de tiempo.

```

restaurante=# SELECT
restaurante=#     COUNT(o.folio_orden) AS total_ventas,
restaurante=#     SUM(o.total_orden) AS monto_total_ventas
restaurante=# FROM
restaurante=#     orden o
restaurante=# WHERE
restaurante=#     o.fecha_orden >= '2024-01-01'
restaurante=#     AND o.fecha_orden <= '2024-12-31';
total_ventas | monto_total_ventas
-----+-----
              5 |              1670.00
(1 fila)

```

Figura 23: Número y monto de ventas en un periodo de tiempo.

Presentación

Para la implementación del proyecto, optamos por desarrollar una página web con PHP, HTML y CSS para permitir a los usuarios interactuar de manera intuitiva con el sistema. A continuación, describimos el proceso de desarrollo seguido para la creación de las interfaces:

Para comenzar con el proceso de diseño de la página web, se tomó como punto de partida el análisis realizado previamente para la creación de la base de datos desde el diagrama de entidad relación, el modelo relacional intermedio hasta su implementación en postgres y sql.

Para ejecutar nuestro código en un entorno de desarrollo web local seleccionamos MAMP, herramienta que funciona para convertir una computadora en un servidor web local, permitiéndonos así desarrollar y probar sitios web directamente en nuestra computadora.

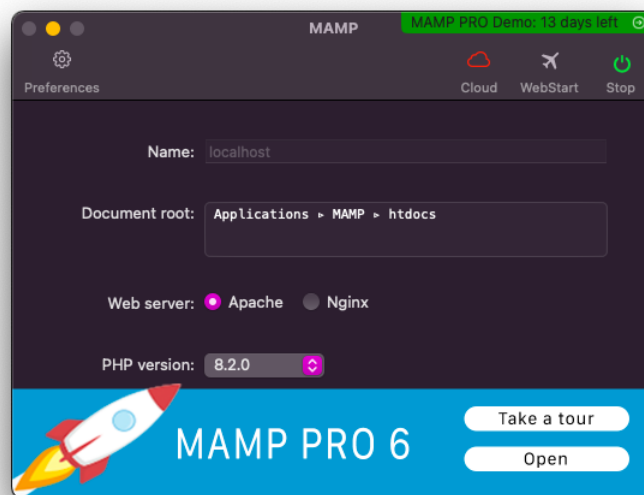
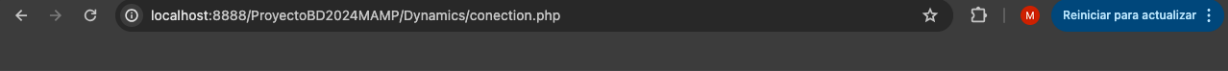


Figura 24: MAMP.

Con ayuda de php conectamos nuestra base de datos y realizamos una consulta de prueba sobre la tabla empleado.



Conexion de la base de datos en postgres con PHP

Conexión exitosa
Consulta exitosa

num_emp	rfc_emp	nombre_pila_emp	ap_pat_emp	ap_mat_emp	fecha_nac_emp	foto_emp	edad_emp	estado_emp	codigo_postal_emp	colonia_emp	calle_emp	num_dom_emp
1	ABC123456789	Juan	Pérez	González	1990-05-15			Ciudad de México	12345	Centro	Calle 1	123
2	DEF987654321	María	López	Martínez	1985-10-20			Ciudad de México	54321	Colonia Norte	Calle 2	456
3	GHI567890123	Pedro	Ramírez	Hernández	1988-03-10			Ciudad de México	67890	Colonia Sur	Calle 3	789
4	JKL321098765	Ana	García	Pérez	1992-07-25			Ciudad de México	98765	Colonia Este	Calle 4	1011
5	MNO543216789	Luis	Fernández	Sánchez	1995-01-30			Ciudad de México	13579	Colonia Oeste	Calle 5	1213
6	PQR987654321	Laura	Díaz	Gómez	1991-09-05			Ciudad de México	24680	Centro	Calle 6	1415
7	STU543216789	Carlos	Martínez	Rodríguez	1989-12-12			Ciudad de México	36924	Colonia Norte	Calle 7	1617
8	VWX123456789	Paula	Hernández	López	1993-04-18			Ciudad de México	48215	Colonia Sur	Calle 8	1819
9	YZA789012345	Javier	Pérez	Fernández	1994-08-20			Ciudad de México	59124	Colonia Este	Calle 9	2021
10	BCD345678901	Elena	Gómez	Sánchez	1996-02-28			Ciudad de México	63179	Colonia Oeste	Calle 10	2223
11	EFG678901234	Ricardo	Martínez	García	1987-11-15			Ciudad de México	74821	Centro	Calle 11	2425
12	HIJ456789012	Mónica	López	Ramírez	1990-06-10			Ciudad de México	89214	Colonia Norte	Calle 12	2627
13	KLM789012345	Fernando	González	Hernández	1986-01-05			Ciudad de México	95123	Colonia Sur	Calle 13	2829

Figura 25: Conexión a la base de datos.

Posteriormente hicimos las vistas necesarias para las consultas más importantes de nuestro proyecto, para facilitar el proceso se hicieron bocetos en Figma. Esta herramienta nos permitió crear prototipos visuales, definiendo la disposición de los elementos, los colores y la tipografía. El diseño en Figma facilitó la planificación y visualización de cómo se verían las páginas finales, asegurando una experiencia de usuario coherente y agradable.

Al finalizar el diseño en Figma, procedimos a la implementación en HTML y CSS. Comenzamos estructurando los documentos HTML, asegurándonos de incluir todos los metadatos necesarios para una correcta configuración de las páginas. Utilizamos Google Fonts para importar la fuentes, que se emplean en diversos elementos de texto en todas las páginas.

El cuerpo del HTML incluye un contenedor principal que agrupa todos los elementos visuales. En la sección del encabezado, colocamos el logotipo del restaurante, el nombre 'Mar de Sabores' y los títulos de las paginas. Estos elementos están diseñados para captar la atención del usuario y proporcionar accesos rápidos y directos. La estructura general de las páginas es consistente, con un encabezado fijo y secciones claramente definidas para diferentes contenidos, como imágenes de platillos, botones, formularios y tablas.

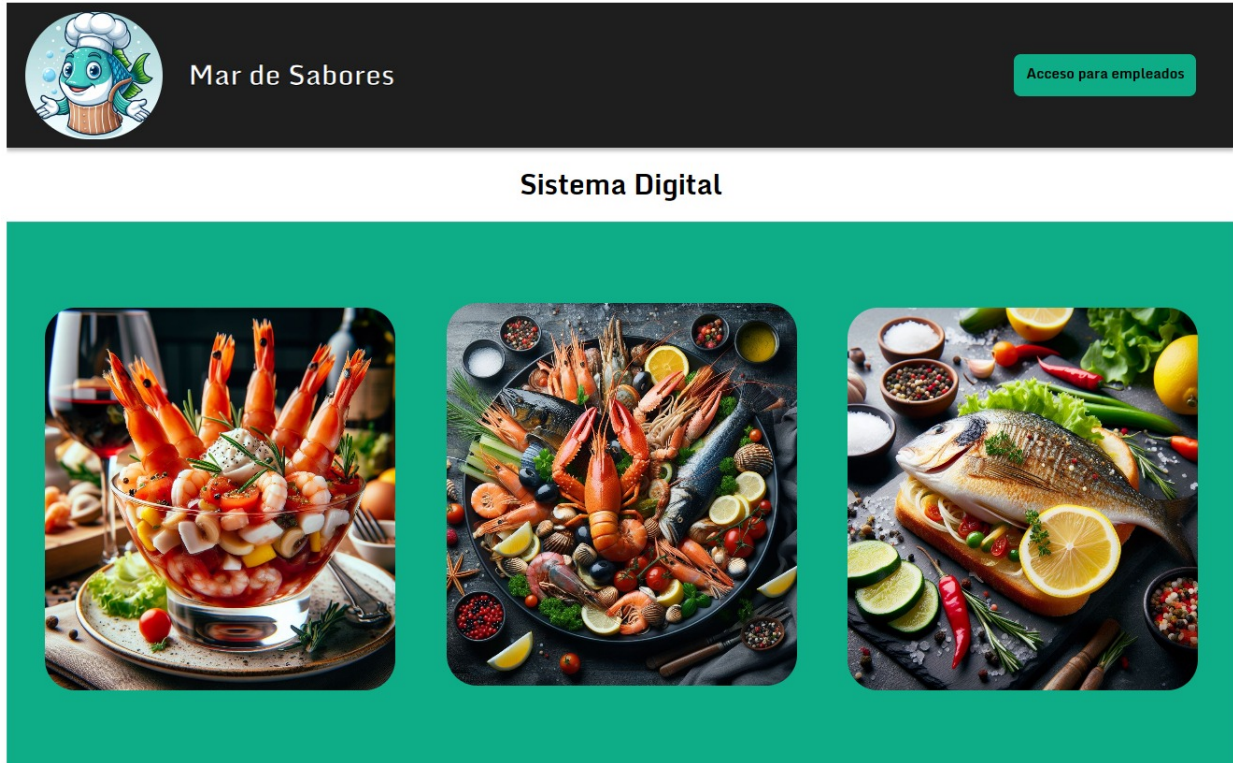


Figura 26: Diseño de la página web.

El archivo CSS contiene estilos detallados para cada elemento. Utilizamos variables CSS para definir las familias de fuentes y asegurar consistencia en todo el diseño. Estilizamos el contenedor principal y sus elementos hijos, como el encabezado, el logotipo y los textos, para que se alineen con el diseño inicial creado en Figma. Además, añadimos sombras, bordes redondeados y un esquema de colores consistente con la identidad del restaurante.

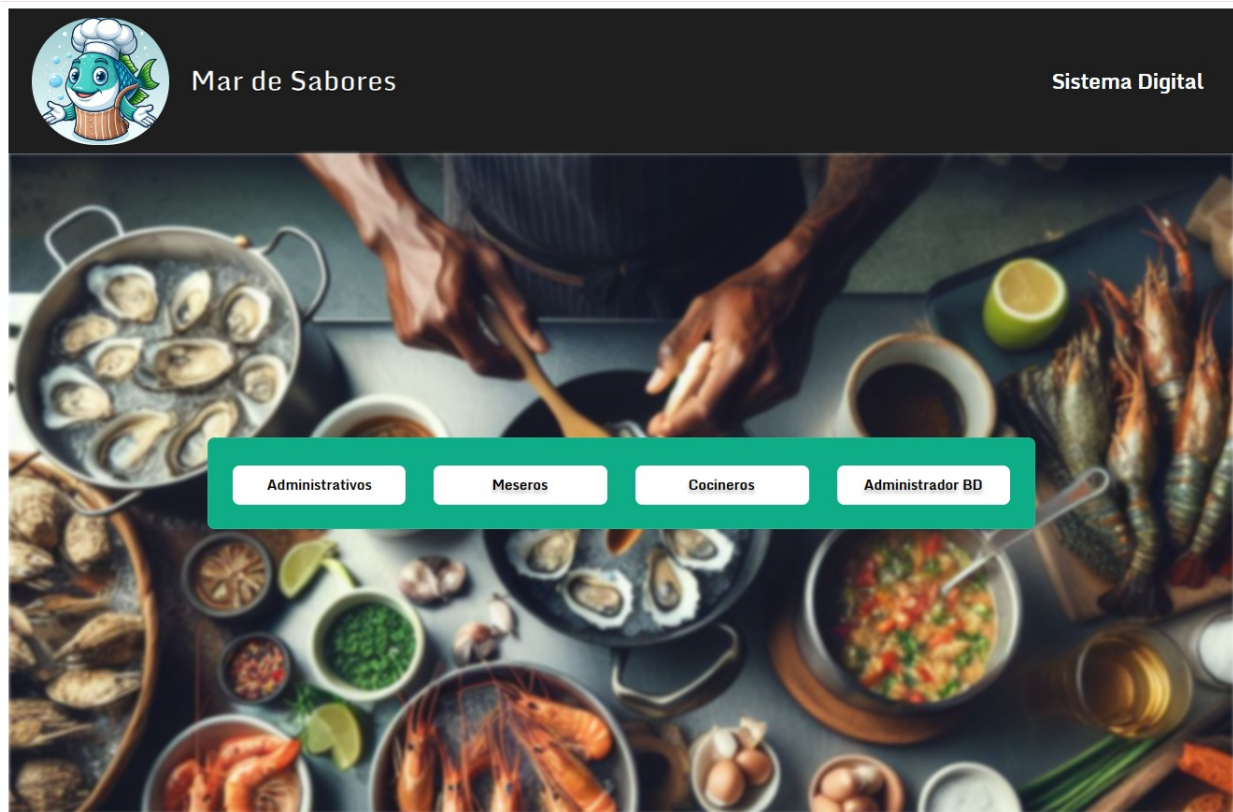
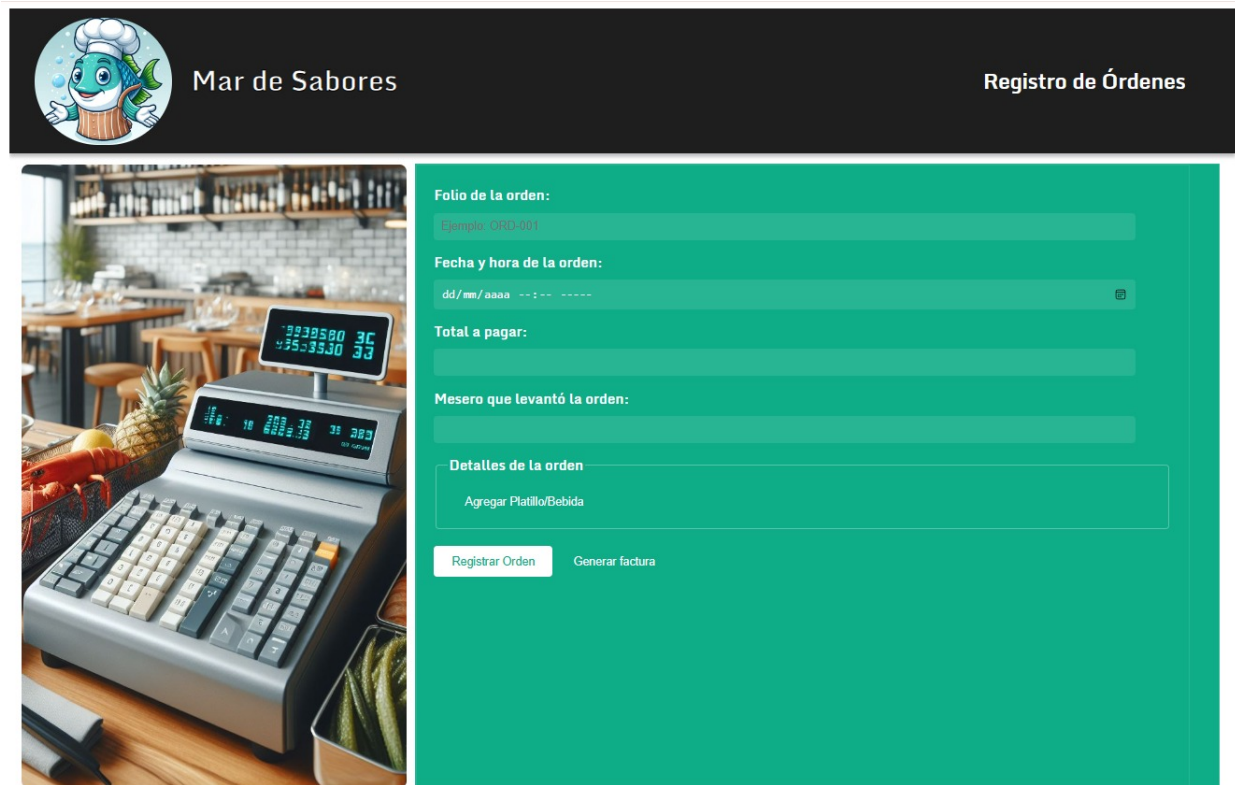


Figura 27: Diseño de la página web para los distintos usuarios.

Aunque los elementos específicos como botones, formularios y tablas varían entre las diferentes interfaces, la estructura general, los colores y el estilo permanecen uniformes, proporcionando una experiencia de usuario cohesiva y profesional.



Mar de Sabores Registro de Órdenes

Folio de la orden:
Ejemplo: ORD-001

Fecha y hora de la orden:
dd/mm/aaaa --:--:--

Total a pagar:

Mesero que levantó la orden:

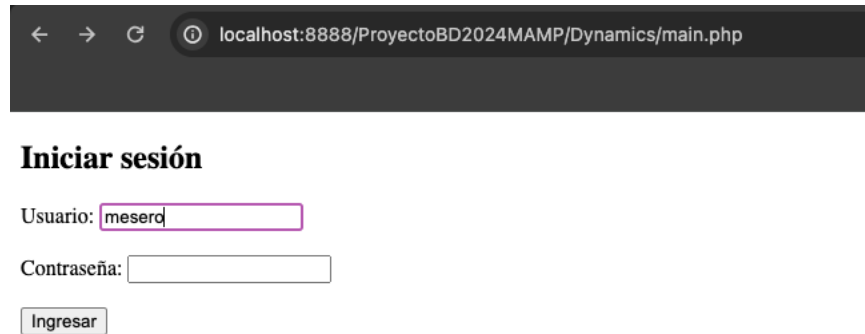
Detalles de la orden
Agregar Plato/Bebida

Registrar Orden Generar factura

Figura 28: Diseño de la página web para insertar los datos de las Ordenes.

La implementación de estas tecnologías resultó en una página de inicio y otras interfaces visualmente atractivas y funcionales, y también reforzó nuestras habilidades en diseño de interfaces y desarrollo web. Aprendimos a integrar herramientas de diseño como Figma con lenguajes de programación web, asegurando una transición fluida del diseño conceptual a la implementación práctica. Esta experiencia nos permitió comprender mejor cómo diseñar y desarrollar interfaces de usuario que no solo cumplen con los requisitos técnicos, sino que también proporcionan una experiencia de usuario óptima.

Una vez que terminamos nuestros archivos html y los unimos con el diseño realizado en figma, comenzamos a programar con PHP la consultas de nuestra pagina web. Para decidir que tipo de vistas se iban a presentar al usuario se programó el formulario para los diferentes tipos de usuario:



← → ↻ ⓘ localhost:8888/ProyectoBD2024MAMP/Dynamics/main.php

Iniciar sesión

Usuario:

Contraseña:

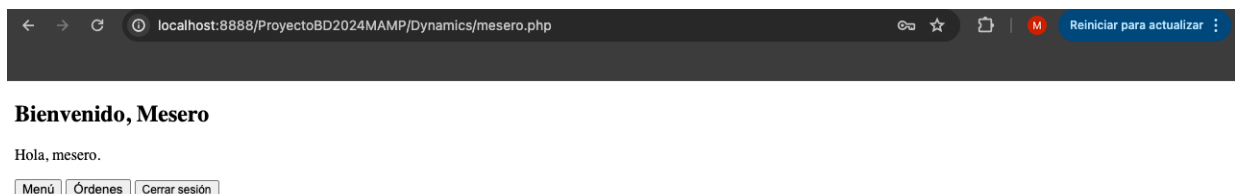
Figura 29: Inicio de sesión.

El objetivo de este inicio de sesión fue delimitar los permisos que cada usuario tendría sobre las tablas para evitar posibles conflictos con la integridad de la base de datos:

```
~
7 CREATE USER bdadmin WITH PASSWORD 'bdadmin123' SUPERUSER;
8 CREATE USER administrativo WITH PASSWORD 'administrativo123';
9 CREATE USER mesero WITH PASSWORD 'mesero123';
10
11 -- Otorgar permisos al usuario administrativo
12 GRANT CREATE ON DATABASE restaurante_db TO administrativo;
13 GRANT CONNECT ON DATABASE restaurante_db TO administrativo;
14 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrativo;
15
16 -- Otorgar permisos al usuario empleado (mesero)
17 GRANT CONNECT ON DATABASE restaurante_db TO mesero;
18 GRANT SELECT, INSERT, UPDATE ON TABLE public.ordenes TO mesero;
```

Figura 30: Permisos.

En nuestro archivo de PHP lo manejamos de la siguiente forma:



← → ↻ ⓘ localhost:8888/ProyectoBD2024MAMP/Dynamics/mesero.php

Bienvenido, Mesero

Hola, mesero.

Reiniciar para actualizar

Figura 31: Vista mesero.

Como podemos observar, se tienen dos vistas posibles. Si el mesero ingresa al menu puede observar todos los productos dependiendo de la categoría seleccionada:

Independientemente de la categoría se hizo una consulta que nos permite ver los productos que no están disponibles:

```

    $query = "SELECT id_prod,nombre_prod,precio_unitario_prod
    FROM menu
    WHERE disponibilidad_prod = false;
    ";
    echo "<h2> Productos no disponibles :</h2>";
    $result = pg_query($conexion, $query);
    muestraTab($result, pg_num_fields($result));
    // Cerrar conexión
    pg_close($conexion);

```

Figura 32: Consulta en PHP.

Para la categoría 'Entrada':

Menú

Selecciona una categoría: Entradas ▼

Mostrar

Entradas:

id_prod	nombre_prod	receta_prod	precio_unitario_prod	disponibilidad_prod	categoria
7	Cóctel de Camarón	Camarones cocidos y enfriados, mezclados con salsa de tomate, cebolla, cilantro, aguacate y jugo de limón.	110.00	t	Entrante
8	Tostadas de Atún	Tostadas de maíz crujientes cubiertas con atún fresco marinado en salsa de soja, limón y chile.	160.00	t	Entrante
9	Gambas al Ajillo	Gambas, ajo, aceite de oliva, perejil, guindilla	170.00	t	Entrante

Productos no disponibles :

id_prod	nombre_prod	precio_unitario_prod
15	botella de vino	520.00

[Regresar a Mesero](#)

Figura 33: Menu Mesero.

Para la categoría "Plato principal":

Menú

Selecciona una categoría: Entradas ▼

Mostrar

Plato principal:

id_prod	nombre_prod	receta_prod	precio_unitario_prod	disponibilidad_prod	categoria
1	Ceviche de Camarón	Camarones frescos marinados en jugo de limón, con cebolla morada, chile serrano, aguacate y cilantro.	120.00	t	Plato principal
2	Filete de Pescado a la Veracruzana	Filete de pescado fresco cocinado en salsa de tomate, aceitunas, alcaparras, chiles y cebolla.	180.00	t	Plato principal
3	Pulpo a la Gallega	Pulpo cocido y aderezado con aceite de oliva, pimentón y sal.	200.00	t	Plato principal
4	Paella Marinera	Arroz con una variedad de mariscos como camarones, almejas, mejillones, calamares y pescado, cocinados con azafrán.	250.00	t	Plato principal
5	Ensalada de Mariscos	Mezcla de mariscos frescos como camarones, pulpo y calamares sobre una cama de lechuga fresca, tomate, pepino y aguacate.	150.00	t	Plato principal
6	Aguachile de Camarón	Camarones frescos marinados en una salsa de jugo de limón, chile serrano y pepino.	130.00	t	Plato principal

Productos no disponibles :

id_prod	nombre_prod	precio_unitario_prod
15	botella de vino	520.00

[Regresar a Mesero](#)

Figura 34: Plato principal.

Para la categoría "Postre":



Menú

Selecciona una categoría: Entradas ▼

Mostrar

Postres:

id_prod	nombre_prod	receta_prod	precio_unitario_prod	disponibilidad_prod	categoria
10	Pastel de Tres Leches	Pastel esponjoso bañado en una mezcla de tres tipos de leche y cubierto con crema batida.	90.00	t	Postre
11	Flan de Coco	Postre tradicional hecho con crema de coco, leche condensada y huevos, cubierto con caramelo líquido.	80.00	t	Postre
12	Cheesecake de Maracuyá	Cheesecake suave y cremoso con un toque refrescante de maracuyá, sobre una base de galleta.	120.00	t	Postre

Productos no disponibles :

id_prod	nombre_prod	precio_unitario_prod
15	botella de vino	520.00

[Regresar a Mesero](#)

Figura 35: Postres.

Para la categoría "Bebida":

Menú

Selecciona una categoría: Entradas ▼

Bebidas:

id_prod	nombre_prod	receta_prod	precio_unitario_prod	disponibilidad_prod	categoria
13	Mojito de Maracuyá	Refrescante cóctel hecho con ron, maracuyá, jugo de limón, hojas de menta y agua mineral.	100.00	t	Bebida
14	Margarita de Mango	Cóctel clásico con tequila, licor de naranja, mango fresco, jugo de limón y un borde de sal.	110.00	t	Bebida
15	botella de vino	Botella de vino tinto italiano	520.00	f	bebida

Productos no disponibles :

id_prod	nombre_prod	precio_unitario_prod
15	botella de vino	520.00

[Regresar a Mesero](#)

Figura 36: Bebidas.

Por otro lado, si en el apartado de mesero de la pagina ingresamos a orden tenemos lo siguiente:

← → ↻ localhost:8888/ProyectoBD2024MAMP/Dynamics/ordenes.php ☆ | M Reiniciar para actualizar ⋮

Mar de Marisco

Categoría: Entradas ▼

Detalles:

ID del Producto:

Cantidad:

Figura 37: Enter Caption.

El objetivo principal al realizar la pagina web era implementar para el usuario una página intuitiva para facilitar el manejo y uso de la base de datos, sin embargo, al realizar el código de PHP hubo complicaciones al traducir todas las consultas al lenguaje de programación puesto que se tiene que programar cada una de manera individual y posteriormente leer y darle formato a la información con el mismo lenguaje para visualizar los resultados.

Nosotros hicimos uso de los siguientes archivos php para realizar las consultas mostradas:

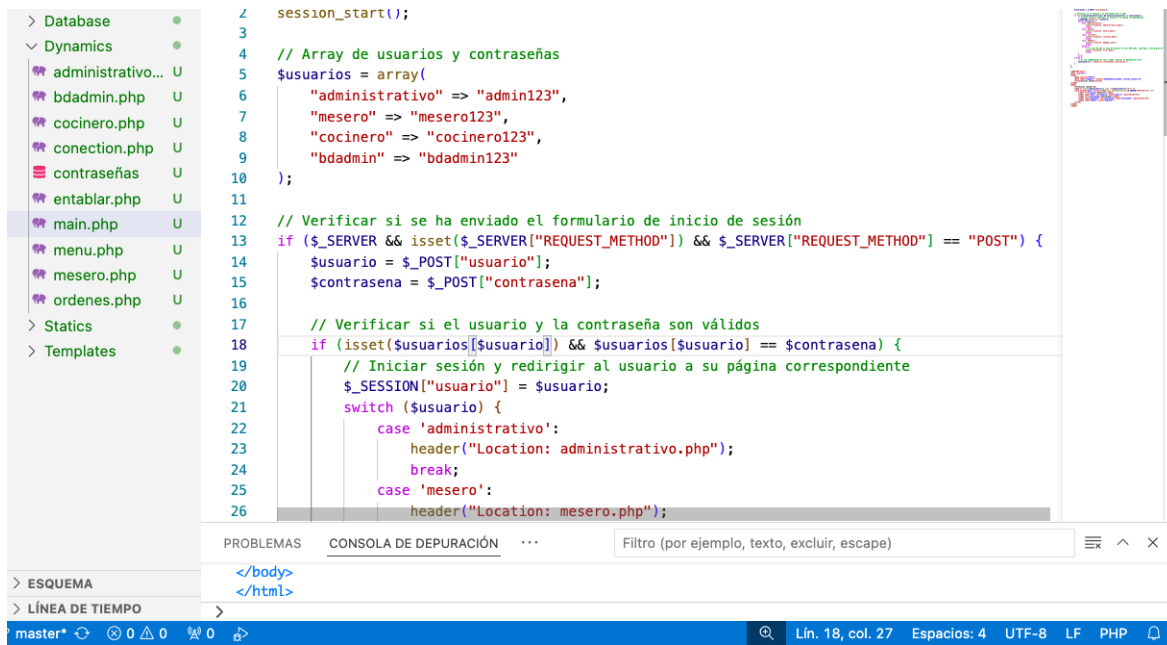


Figura 38: Enter Caption.

Conclusiones

Barragán Pilar Diana: Para poder realizar el proyecto primero identificamos las distintas reglas de negocio y los requerimientos que se nos piden para a partir de estos realizar el modelo conceptual como lo son; identificar las entidades que tendrá nuestro modelo en este caso fueron catorce entidades en todas, de la misma manera procedimos con los atributos a partir del texto que se nos presento, identificamos cuales de ellos serían claves principales, claves candidatas, atributos compuestos, calculados, entre muchos otros tipos de atributos. Para posteriormente hacer uso del modelo entidad relación extendido con los supertipos y subtipos, agregando a esto el uso de la generalización y restricciones como la exclusividad, para después asimilar que tipo de relaciones tendrían entre ellas ya sean de muchos a muchos, uno a uno o uno a muchos, sin dejar de lado el que existían entidades débiles que también debían de ser representadas en el modelo conceptual.

A partir del modelo entidad relación que implementamos, mapeamos esto que ya teníamos con la ayuda de distintos procedimientos a un modelo relacional intermedio para finalmente tener un modelo relacional, en el intermedio definimos los tipos de datos que utilizaríamos para cada atributo igual que las restricciones, siendo que estos podían ser tanto primary key PK, foreign key FK, unique U, calculado C, determinante D, no nulos. Con los atributos y sus definiciones pasamos a propagar las llaves primarias de acuerdo al tipo de relación por ejemplo si se trata de una entidad débil pasamos la PK de la entidad fuerte como FK y PK a la débil. Finalmente a partir de este modelo obtuvimos el modelo relacional con las tablas.

Teniendo todo esto en cuenta procedimos a pasarlo al lenguaje SQL haciendo uso de sus diversas categorías como DDL (Data Definition Language) para crear las tablas, DML (Data Manipulation Language) para insertar valores y DQL (Data Query Language) para las

consultas, por ultimo realizamos las consultas que se nos pidieron de la información de las tablas.

Para terminar este trabajos fue bastante completo y complejo puesto que tuvimos que aplicar todo lo adquirido en el curso a un caso real, para ello se presentaron dificultades puesto que en un primer momento nos tardamos generando el modelo conceptual pues este era la base de todo el proyecto, lo que sin duda más se nos complico fue la realización de la página web, específicamente a la hora de conectar la base de datos, sin embargo logramos implementar los requerimientos que se nos pidieron dado el caso de estudio.

Martínez Bravo Daniela: Primero, nos enfocamos en el diseño lógico y la creación del modelo conceptual, donde analizamos los requisitos y definimos entidades clave como Empleado, Dependiente, Menú, Orden y Factura. Este proceso nos permitió crear un modelo entidad-relación sólido y eficiente.

Luego, implementamos el modelo físico de la base de datos en SQL, asegurando la integridad referencial y el cumplimiento de las reglas de negocio. También desarrollamos triggers y funciones en PL/pgSQL para automatizar tareas críticas, como el cálculo de edad de empleados y la actualización de órdenes, mejorando la consistencia y el rendimiento del sistema.

Además, desarrollamos una página web conectada a la base de datos, permitiendo a los usuarios interactuar de manera intuitiva con formularios de inserción de datos y consultas dinámicas, proporcionando una experiencia de usuario eficiente.

Este proyecto reforzó varias habilidades clave. Profundicé en el diseño de bases de datos, comprendiendo mejor la importancia de una buena estructuración de datos. Aprendí a automatizar procesos con funciones y triggers en PL/pgSQL, mejorando mi capacidad para escribir scripts eficientes. La colaboración en equipo se facilitó con el uso de herramientas como Notion, asegurando la gestión efectiva de tareas. Finalmente, el desarrollo de una interfaz web conectada a la base de datos me permitió entender mejor la integración de tecnologías para ofrecer soluciones completas.

López Ramírez Monserrat: Con base en los resultados obtenidos se llego a diferentes conclusiones: En primer lugar, el uso de bases de datos como herramienta de manejo de información es muy útil, sin embargo es necesario plantear un correcto diseño desde un inicio para lograr que la aplicación sea funcional para los requerimientos del proyecto.

En segundo lugar, se comprobó que una correcta planeación de las actividades así como una buena colaboración y comunicación entre los miembros de un equipo es fundamental para completar un proyecto adecuadamente.

Suarez Velasco Gabriela: Para este proyecto en lo personal el caso de estudio fue accesible de entender ya que era todo muy especificado y por ende fue mas rápido el poder realizar el modelo relacional, claro que tuvimos algunos errores que se solucionaron con las observación del Ingeniero Arreola y el equipo, la colaboración con mis compañeros fue totalmente grata ya que con ayuda de las sesiones y la correcta distribución de tareas pudimos tener un mayor alcance en los puntos de este proyecto.

Por otro lado este proyecto tubo su complejidad, en lo personal tuve ciertas incertidumbres

al crear algunos índices ya que no sabía si eran los correctos, al igual la crear la tablas para la base tuvimos algunos pequeños tropiezos, y mas que un reto fue un desafío el hacer la presentación de este proyecto en ingles ya que para mí es la primera vez que entrego un trabajo en ese formato y al final de todo este proceso se logro el cometido de manera satisfactoria.

Aguirre Córdova Omar Gabriel: El desarrollo de este proyecto nos permitió aplicar todo el temario de forma integral, desde el diseño hasta la implementación aunque esta no se terminara por completo.

Referencias

postgresql. (2024, May 09). Chapter 8. Data Types. [Online]. Available: <https://goo.su/yqTF> [Accessed May. 03, 2024].

postgresql. (2024, May 09). PostgreSQL 16.3 Documentation. [Online]. Available: <https://goo.su/pHpDch3> [Accessed May. 13, 2024].