



Universidad Nacional Autónoma De México

Facultad de Ingeniería

Ingeniería en Computación

Bases de Datos

Profesor: Fernando Arreola Franco
G01

Proyecto Final: El restaurante

Alcocer Velasco Abigail
Genis Cruz Lourdes Victoria
González Cuellar Arturo
Maya Herrera Alexis Daniel
Roldán Sánchez Alexis

Seshat Dev

27 de mayo de 2022

Índice

1. Introducción	3
2. Plan de Trabajo	3
3. Diseño	4
4. Implementación	13
5. Aplicación Web	24
6. Conclusiones	38

Proyecto Final Base de Datos

Alcocer Velasco Abigail, Genis Cruz Lourdes Victoria, González Cuellar Arturo,
Maya Herrera Alexis Daniel, Roldán Sánchez Alexis

May 2022

1. Introducción

Toda empresa u organización que maneja grandes cantidades de información requiere de un sistema de gestión de base de datos, ante esa necesidad, en el entorno del mercado actual, el implementar estas herramientas son imprescindibles para el éxito de la empresa.

Siendo el objetivo de esto automatizar tareas relacionadas con la inserción, sustracción o borrado de datos para obtener información estructurada que ayude a la mejora y la integridad de los datos así como el esfuerzo por mejorar la seguridad de los datos confidenciales que se puedan contener dentro de la organización.

Dado este entorno, existe una mayor demanda de estos datos, y por lo tanto, más necesidad de gestionarlos. Es ahí donde entran las organizaciones dedicadas a cumplir estas demandas al diseñar, administrar, supervisar y asegurar el uso adecuado de los datos dentro de una base de datos, dichas empresas deben garantizar que se manejen estas grandes cantidades de información de manera eficiente en la recopilación y análisis dependiendo de las necesidades del cliente.

Todo el código, archivos, diagramas y demás, puede encontrarse en el siguiente repositorio:
Repositorio

2. Plan de Trabajo

Recientemente la necesidad por optimizar los procesos relacionados con las diferentes áreas del restaurante han llevado a esta organización a diseñar una base de datos que vuelva eficiente la consulta y el almacenamiento de la información para garantizar el proceso de atención al cliente.

Siendo el objetivo que los clientes sean atendidos con un excelente servicio para mantener el buen posicionamiento en el mercado y aumentar el nivel de ventas.

Esta organización tiene la responsabilidad de cumplir los objetivos y las necesidades del cliente siendo la fecha límite de lanzamiento la última semana de mayo correspondiente a los días 25-27 del mes mencionado.

Este proyecto consta de distintas etapas que se irán sumando en el diseño de la base de datos, por lo cual presentamos el siguiente plan de trabajo definiendo además el tiempo estimado a cada una de ellas:

1. Comprensión, análisis y designación de requerimientos. En este primer punto los integrantes del equipo buscan analizar de forma grupal los requerimientos para desarrollar este proyecto, plantear dudas, hacer propuestas e ir tratando de definir un camino para realizar el proyecto.

2. Designación de requerimientos y definición de medios de comunicación y colaboración. Una vez entendido cada punto se busca elaborar un plan de trabajo para designar roles y actividades a cada integrante del equipo. También es de vital importancia definir los medios de comunicación y de colaboración para el equipo, usando así principalmente WhatsApp, GitHub y Google Drive.
3. Realizar el modelo entidad relación que es prácticamente el parteaguas para desarrollar la base de datos, ver las relaciones, atributos, cardinalidades y demás.
4. Creación de la base de datos en base al modelo relacional, realizar tablas, relaciones y restricciones dentro del manejador correspondiente (PostgreSQL).
5. Diseño y programación del FrontEnd de la página web una vez que se analice el modelo relacional en el cual se contemplan los puntos necesarios para la interacción con el usuario y el desarrollo de las funcionalidades.
6. Diseño, programación e implementación del BackEnd, en el cual se programará todo lo necesario para que funcione el servidor de la pagina web y se logre la conexión con la base de datos.
7. Se realizarán pruebas y verificaciones del correcto uso de la aplicación web.
8. Elaboración de la documentación necesaria para la entrega del proyecto, remarcando objetivos y planes; explicación de las fases, diagramas y de la implementación de la aplicación web; y finalmente las conclusiones para el cierre de la actividad.
9. Preparación y desarrollo de la exposición del proyecto.

La organización puede verse mejor reflejada dentro del siguiente diagrama de Gantt:

Nº Actividad	Actividad	Inicio	Final	23/04	24/04	25/04	30/04	01/05	03/05	05/05	08/05	12/05	16/05	19/05	20/05	21/05	23/05	25/05	30/05
1	Análisis de los requerimientos	23/04/2022	24/04/2022																
2	Plantamiento de los objetivos del proyecto	24/04/2022	25/04/2022																
3	Designación de actividades	25/04/2022	25/04/2022																
4	Elaboración del MER	25/04/2022	30/04/2022																
5	Elaboración del MR	01/05/2022	03/05/2022																
6	DDL de nuestra BD	05/05/2022	12/05/2022																
7	DML de nuestra BD	08/05/2022	16/05/2022																
8	Desarrollo FrontEnd	08/05/2022	19/05/2022																
9	Desarrollo Backend	15/05/2022	20/05/2022																
10	Conexión BD con aplicación web	19/05/2022	21/05/2022																
11	Pruebas	21/05/2022	23/05/2022																
12	Documentación en Latex	21/05/2022	23/05/2022																
13	Realizar presentación	23/05/2022	25/05/2022																
14	Entrega proyecto		25/05/2022																
15	Presentación del proyecto		30/05/2022																

3. Diseño

1. Modelo Entidad - Relación (MER)

El modelo entidad relación es una herramienta que nos permite representar de manera simplificada los componentes que participan en el proceso de negocio; con esto, creamos herramientas especialmente útiles para el diseño conceptual del requerimiento, así como para que una vez implementado, se tenga una forma intuitiva de los elementos clave con los que se está trabajando y cómo se relacionan entre ellos.

Una vez dado el contexto, esta organización identificó los requerimientos principales que formarán parte de este modelo creando así el concepto de la base de datos basados en el modelo de negocio con el que cuenta el restaurante y cumpliendo con las necesidades demandadas para la digitalización de operación de dicho restaurante, por lo cual se definen las siguientes entidades de diseño:

- Administrativos: Personal fundamental para la administración y el manejo del restaurante, nos interesa tener registro de su puesto.

- Cocineros: Personal encargado de la preparación de los alimentos por lo cual nos interesa conocer su especialidad.

- Meseros: Personal encargado de la atención al cliente por lo cual nos interesa conocer su horario para crear los planes de atención dependiendo el día y la demanda de su servicio.

Por su parte, debemos almacenar información básica de cada uno de los empleados en general sin importar su puesto. De esta información, que es de gran relevancia para el manejo administrativo y de recursos humanos, se debe almacenar: RFC, número de empleado asignado, nombre, fecha de nacimiento, teléfonos de contacto, edad, domicilio, así como su sueldo.

Adicionalmente, es de gran interés para el restaurante tener registro de los dependientes de los empleados, esto con el fin de garantizar los diferentes apoyos con los que se cuentan y que puedan ser brindados a los familiares. De esta manera, se solicitan los siguientes datos: CURP, nombre y parentesco.

Una vez identificado al personal, es de gran importancia almacenar los datos referentes a los recursos materiales y a los productos que ofrece este restaurante:

Inicialmente se debe tener disponible la información de los platillos y bebidas que el restaurante ofrece, de estos productos nos interesa almacenar una descripción, nombre, su receta para consulta de cocineros, precio y un indicador de disponibilidad, así como una categoría asignada de la cual debemos almacenar el nombre y su descripción.

Es importante tener un control del manejo administrativo y operativo del restaurante, por lo cual para generar este control referente a la atención de los clientes, control de personal y contabilidad de materia prima ocupada en día de operación, es necesario almacenar las órdenes generadas con un folio identificador, la fecha, cantidad total a pagar y registro de mesero que levantó la orden. Además, la cantidad de cada platillo/bebida y precio total a pagar.

Por su parte, el restaurante ofrece la facturación de la cuenta, para lo cual debemos tener registro de datos personales del cliente, los cuales son necesarios para realizar dicha factura, estos datos son los siguientes: RFC, nombre, domicilio, razón social, email y fecha de nacimiento.

Ante la definición de los roles y partes fundamentales para el funcionamiento del restaurante, surgen ciertas restricciones que se deben tomar en cuenta, esto con el fin de garantizar la integridad de la información y la parte administrativa del restaurante, dichas restricciones son las siguientes:

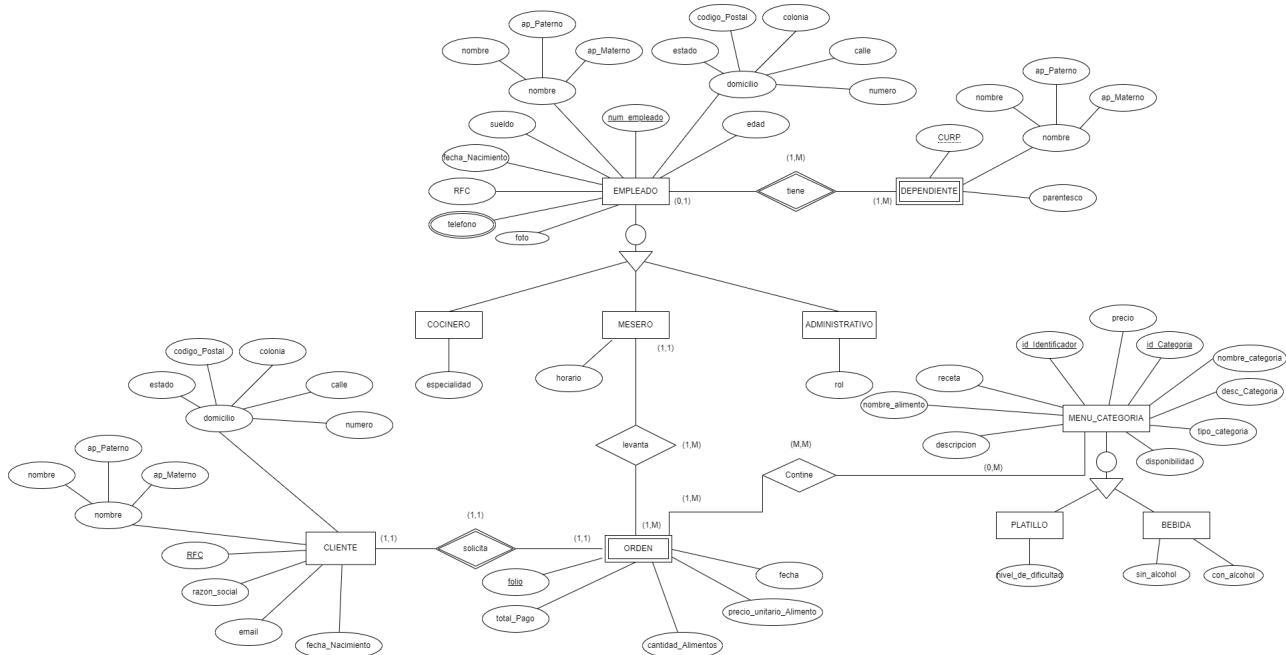
- Cada empleado puede no tener dependientes o puede tener más de uno, sin embargo, cada dependiente solo puede estar asociado a un empleado. Esto quiere decir que no se pueden brindar más de una vez los beneficios que brinda el restaurante a dichos dependientes.

- Se tiene que contar con un inventario digital, en el cual, cada que se agregue un producto a una orden, debe actualizarse el total (por producto y venta), así como validar que el producto este disponible.

- Se debe tener un reporte de la cantidad de órdenes que registra cada mesero, así como el total que se ha pagado por dicha orden.

- Generar un reporte de contabilidad, en el cual se pueda consultar el total del número de ventas y el monto por las ventas en un periodo de tiempo.

Finalmente, una vez identificados todos los requerimientos y las restricciones mencionadas anteriormente, presentamos el siguiente modelo gráfico, en el cual podemos identificar de una manera sencilla la relación y cómo es que van a interactuar los datos dentro de la base:



Modelo Entidad-Relación

2. Modelo Relacional (MR)

Una vez definido el entorno de desarrollo y planteando los requerimientos solicitados, se va a definir la estructura de datos para la implementación lógica de la información de nuestro cliente mediante una serie de tablas considerando las restricciones existentes en las relaciones para reflejar la semántica y mantener la integridad de los datos.

En este proceso, el corporativo tiene la tarea de definir la estructura de la base con las restricciones, llaves primarias, tipos de datos y relaciones que están asociadas a la base de datos solicitada sin perder de vista el objetivo de mantener la integridad y la calidad de los datos del restaurante.

A continuación se muestra la definición y la estructura de datos que se tomarán en cuenta para el diseño de la base de datos planteada en el modelo entidad-relación anterior.

Es importante conocer los métodos de transformación que se implementarán para la definición del modelo relacional, dichos métodos empleados con la mejor estrategia para garantizar la optimización y la totalidad de los datos, evitando así redundancias e inconsistencias de estos:

- Transformación de atributos

- En claves candidatas debe establecerse restricción de unicidad (U).
- Los atributos compuestos deben indicarse de forma individual.
- Para atributos multivaluados se crea una nueva relación y se propaga como llave foránea (FK) la PK de la relación base a la nueva relación (para los atributos multivaluados vamos a crear una nueva relación el cual lo vamos a denotar por una FK).
- Para atributos derivados se indica que son calculados (C).
- Finalmente, se deben indicar las restricciones que haya sobre los atributos, como check (CK) o NULL (N).

A su vez, es importante tener restricciones y métodos para la transformación de relaciones, esto con el fin de garantizar la conexión necesaria para consulta de cualquier registro sin perder la referencia, así como garantizar que se obtendrá el mejor rendimiento al realizar dicha consulta, dichos métodos son los siguientes:

- Transformación de Relaciones

- m:m Se crea una nueva relación que tendrá como PK's de la entidades que une (que a su vez son FK's), más los atributos (si hubiera) de la relación.
- 1:m o m:1 La llave primaria de la relación con cardinalidad 1 se propaga como llave foránea a la relación con cardinalidad m.
- 1:1 La clave primaria de una relación se propaga a la otra relación dependiendo de:
 1. La semántica.
 2. Considerar cuál relación será accedida más frecuentemente.

- Transformación de Entidades Débiles

- Se crea una nueva relación conservando todos sus atributos.
- Cuando hay dependencia de identificación se propaga la llave principal de la entidad fuerte hacia la débil (FK), ya que en conjunto con el discriminante, formará la llave primaria (PK) de la entidad débil.

Una vez definidas las transformaciones que se implementarán para el modelo relacional, comenzaremos a realizar dicha transformación de nuestro modelo iniciando con la entidad EMPLEADO, en donde vamos a destacar el número de empleado como nuestra llave primaria y al teléfono como atributo multivaluado. Esto en el diseño DDL se verá reflejado como una tabla independiente.



Entidad EMPLEADO

Ya que se identificaron los tipos de datos, entonces realizamos la transformación sin perder de vista la llave primaria, la cual aparte de servir como identificador único de los atributos, nos permitirá crear relaciones entre otras entidades, y nuestro atributo multivaluado, que posteriormente se convertirá en una tabla independiente junto con la llave primaria, para garantizar el rendimiento al momento de actualizar, insertar o borrar un registro:

Tipo de atributo	Nombre del atributo
int	num_empleado, codigo_Postal, telefono
smallint	edad
float	sueldo
date	fecha_nacimiento
varchar	RFC_EMPLEADO, nombre, ap_Paterno, ap_Materno, calle, numero, colonia, estado

Atributos entidad EMPLEADO

Finalmente se muestra la transformación de nuestra entidad EMPLEADO, recordando que es una generalización de nuestros diversos tipos de empleados, sin embargo, para diferenciarlos se implementarán estrategias las cuales serán mencionadas posteriormente, esto con el fin de asegurar la optimización y el mejoramiento del manejo de información:

```
EMPLEADO: {num_empleado int (PK),
edad smallint,
sueldo float,
fecha_nacimiento date,
RFC_EMPLEADO varchar(13),
nombre varchar(60),
ap_Paterno varchar(60),
ap_Materno varchar(60),
calle varchar(60),
numero smallint,
colonia varchar(60),
codigo_Postal int,
estado varchar(60)}
```

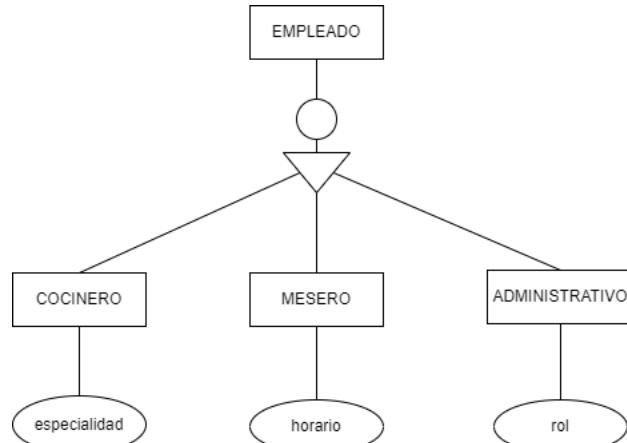
Transformación de entidad EMPLEADO

Como se mencionó anteriormente, el manejo y transformación de atributos multivaluados recibe otro tratamiento, esto con el fin de garantizar el buen manejo de los datos al poder actualizar, borrar e insertar distintos números asociados al mismo empleado sin la necesidad de actualizar otros datos personales, además de que se evita la duplicidad de números asociados al empleado registrado; por lo cual se considera una nueva tabla en la cual se tiene una llave primaria compuesta por el número de empleado en conjunto con el teléfono asociado a dicho empleado; la transformación queda de la siguiente manera:

```
telefono:{ No_telefono int (PK),
num_empleado int (FK)}
```

Transformación de atributo multivaluado

Para lograr el mantenimiento de los datos diferenciales asociados a cada empleado es necesario identificar la mejor estrategia de transformación; en este caso se busca ir de lo particular a lo general, es decir, partimos de la entidad Empleado con atributos definidos, a partir de esta, se realiza una especialización a los tres diferentes tipos de empleados con los que se cuenta, partiendo de que nos interesa almacenar información única para cada rol.



Entidad EMPLEADO con sus roles

Para realizar dicha transformación se seleccionó la siguiente estrategia al permitirnos una mejor identificación de cada rol disponible:

- Estrategia de Transformación
 - Crear una relación para el supertipo, incluyendo todos sus atributos.
 - Crear una relación independiente para cada subtipo, incluyendo la llave primaria del supertipo y los atributos del subtipo (si lo hay).

Una vez definida la estrategia a implementar, se presentan dichas transformaciones realizadas a cada uno de los roles con los que cuenta el restaurante:

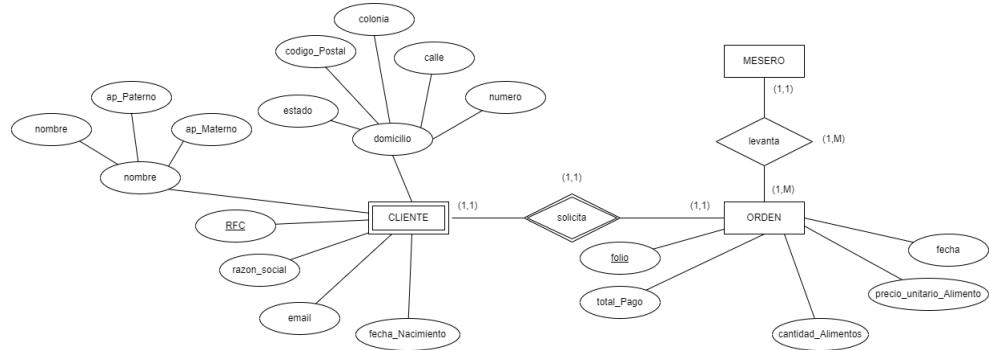
**COCINERO:{num_empleado int (PK)(FK),
especialidad varchar(60)}**

**ADMINISTRATIVO:{num_empleado int (PK)(FK),
rol varchar(25)}**

**MESERO:{num_empleado int (PK)(FK),
Horario date}**

Roles Restaurante

Continuando con nuestras transformaciones, analizamos la entidad débil CLIENTE. Sabemos que una entidad débil es aquella que no puede ser identificada sin una entidad fuerte. En este caso la entidad está asociada a la entidad ORDEN, si los clientes no llegan al restaurante, las órdenes no se solicitan ya que dependen de las mismas. En este caso no se tiene dependencia de identificación ya que se cuenta con una entidad débil que dentro de sus atributos tiene uno que permite identificar de forma clara y única cada registro de información que en este caso es el folio.



Relación Solicitud

Por otro lado se puede identificar que la entidad ORDEN tiene una relación levanta con la entidad MESERO, por lo tanto seguimos las reglas de la transformación de relaciones que en este caso, al ser una relación una a muchos, la llave primaria de la relación con cardinalidad uno se propaga como llave foránea a la relación con cardinalidad m. Debemos recordar que la llave primaria de la entidad MESERO es la llave primaria del supertipo, es decir, el RFC:

* Para la entidad CLIENTE:

```

CLIENTE:{RFC_CLIENTE varchar(13) (PK),
nombre varchar(60),
ap_Paterno varchar(60),
ap_Materno varchar(60),
razon_social varchar(100),
email varchar(100),
fecha_nacimiento date,
calle varchar(60),
numero smallint,
colonia varchar(60),
codigo_Postal int,
estado varchar(60)}
  
```

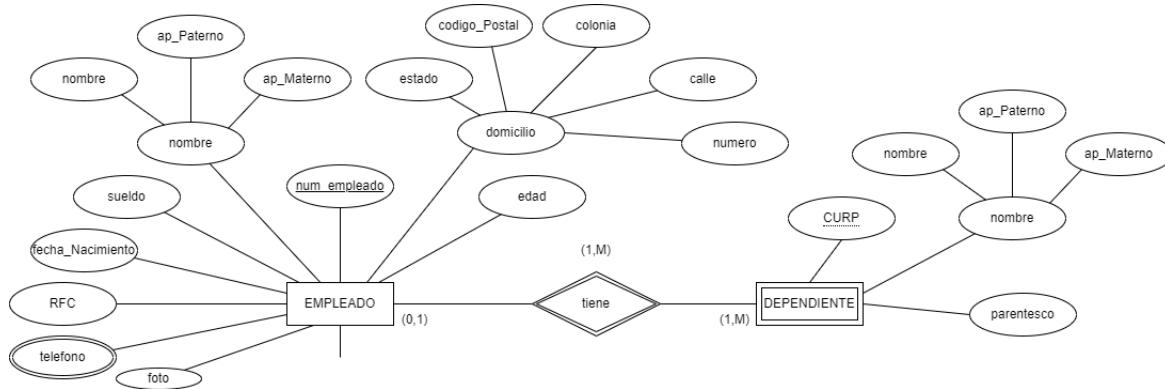
Entidad CLIENTE

* Para la entidad ORDEN:

```
ORDEN:{ folio int (PK),
fecha date,
total_Pago float,
cantidad_Alimentos int,
precio_unitario_Alimento float,
RFC_CLIENTE varchar(13) (FK)}
```

Entidad ORDEN

A continuación se realiza la transformación para otra entidad débil, la entidad EMPLEADO tiene relación con la entidad DEPENDIENTE, en este caso, se identifica que existe dependencia de identificación al deber tener asociado el número de empleado a cada uno de los dependientes para la unicidad del requerimiento:



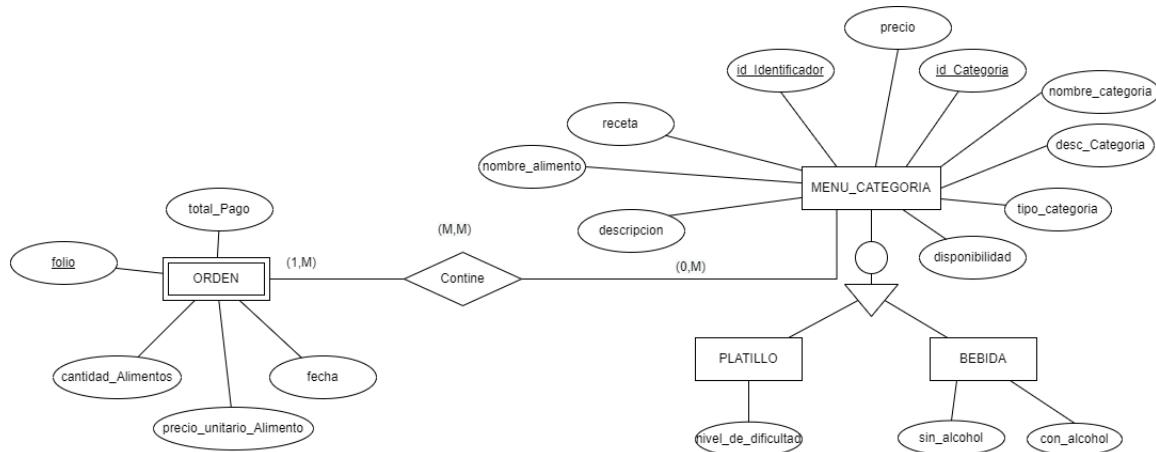
Relación Tiene

Finalmente aplicamos la regla de transformación de entidades débiles la cual nos dice que debemos crear una nueva relación conservando los atributos y propagamos la llave primaria de la relación, por lo cual se tiene dependencia a la nueva relación como llave foránea:

```
DEPENDIENTE: {CURP varchar(18)(PK),
nombre varchar(60),
ap_Paterno varchar(60),
ap_Materno varchar(60),
parentesco varchar(25),
num_empleado int (FK)}
```

Entidad DEPENDIENTE

Por último, se visualiza una relación Contiene entre las entidades ORDEN y CLIENTE, además que la entidad MENU CATEGORIA está generalizada, es decir, se tienen atributos generales para los platillos y bebidas ofrecidos por el restaurante:



Relación Contiene

Para realizar esta transformación se implementa la estrategia que se centra en los subtipos, los cuales tienen en su interior uno o a lo mucho dos atributos, así que para evitar el cruce entre tablas, se crea una sola relación para el supertipo, el cual va a incluir todos los atributos incluyendo los atributos de los subtipos:

```

MENU_CATEGORIA: {id_Identifier smallint (PK),
precio float,
receta varchar(100),
nombre_alimento varchar(60),
disponibilidad boolean,
nombre_categoria varchar(60),
descripcion varchar(60),
desc_Categoria varchar(60),
tipo_categoria varchar(20) null,
nivel_de_dificultad smallint null,
sin_alcohol boolean null,
con_alcohol boolean null}

```

contiene:{[folio int,id_Identifier smallint](PK)(FK)}

Entidad MENU CATEGORIA

4. Implementación

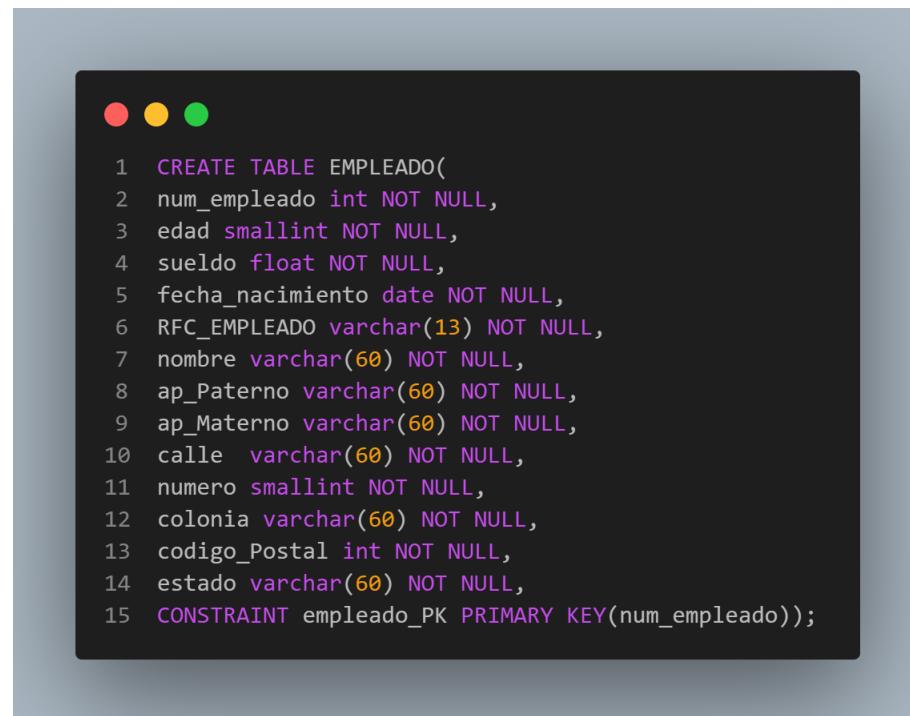
1. DDL

Una vez que se tienen definidas las transformaciones, entonces comenzamos a implementarlas y llevarlas al diseño final, este lenguaje de definición de datos nos ayudará a crear y manipular las estructuras de los datos, siendo útiles para crear, cambiar y eliminar objetos en los cuales están incluidos los índices, tablas y vistas. En este caso las declaraciones más importantes dentro de este lenguaje son las siguientes:

- CREATE (generar una nueva tabla).
- ALTER (modifica la tabla).
- DROP (elimina una tabla de la base de datos).

Por lo cual, con ayuda de nuestras transformaciones realizadas en el punto anterior, se crean las siguientes tablas:

* Tabla EMPLEADO: En esta tabla se definen los elementos de la tabla padre, siendo el número de empleado la llave primaria definida a través de un constraint a nivel de tabla:



```
1  CREATE TABLE EMPLEADO(
2    num_empleado int NOT NULL,
3    edad smallint NOT NULL,
4    sueldo float NOT NULL,
5    fecha_nacimiento date NOT NULL,
6    RFC_EMPLEADO varchar(13) NOT NULL,
7    nombre varchar(60) NOT NULL,
8    ap_Paterno varchar(60) NOT NULL,
9    ap_Materno varchar(60) NOT NULL,
10   calle varchar(60) NOT NULL,
11   numero smallint NOT NULL,
12   colonia varchar(60) NOT NULL,
13   codigo_Postal int NOT NULL,
14   estado varchar(60) NOT NULL,
15   CONSTRAINT empleado_PK PRIMARY KEY(num_empleado));
```

Tabla EMPLEADO

* Tabla telefono: Con el objetivo de normalizar los atributos multivaluados en la tabla EMPLEADO, se define una llave primaria compuesta por los dos atributos presentes:

```
● ● ●  
1 CREATE TABLE telefono(  
2 No_telefono int,  
3 num_empleado int,  
4 CONSTRAINT telefono_PK PRIMARY KEY(No_telefono),  
5 CONSTRAINT telefono_empleado_FK FOREIGN KEY(num_empleado)  
6 REFERENCES EMPLEADO(num_empleado));
```

Tabla telefono

* Tablas COCINERO, MESERO y ADMINISTRATIVO: Permiten almacenar los atributos únicos para cada rol existente dentro del restaurante, teniendo como llave primaria al número de empleado, misma que es llave foránea para no perder la relación y conexión con su tabla padre: EMPLEADO.

```
● ● ●  
1 CREATE TABLE COCINERO(  
2 num_empleado int NOT NULL,  
3 especialidad varchar(60) NOT NULL,  
4 CONSTRAINT cocinero_PK PRIMARY KEY(num_empleado),  
5 CONSTRAINT cocinero_empleado_FK FOREIGN KEY(num_empleado)  
6 REFERENCES empleado(num_empleado));
```

Tabla COCINERO

```
1 CREATE TABLE MESERO(
2 num_empleado int NOT NULL,
3 horario date NOT NULL,
4 CONSTRAINT mesero_PK PRIMARY KEY(num_empleado),
5 CONSTRAINT mesero_empleado_FK FOREIGN KEY(num_empleado)
6 REFERENCES empleado(num_empleado));
```

Tabla MESERO

```
1 CREATE TABLE ADMINISTRATIVO(
2 num_empleado int NOT NULL,
3 rol varchar(25) NOT NULL,
4 CONSTRAINT administrativo_PK PRIMARY KEY(num_empleado),
5 CONSTRAINT administrativo_empleado_FK FOREIGN KEY(num_empleado)
6 REFERENCES empleado(num_empleado));
```

Tabla ADMINISTRATIVO

* Tabla DEPENDIENTE: Cuenta con una llave primaria identificada por el CURP de la persona. Asimismo, se tiene una relación con la tabla EMPLEADO al contar con la referencia del número de empleado asociado a dicho dependiente como llave foránea, esto con el fin de garantizar que un dependiente solo puede estar asociado a un empleado:

```
● ● ●  
1 CREATE TABLE DEPENDIENTE(  
2 CURP varchar(18) NOT NULL,  
3 nombre varchar(60) NOT NULL,  
4 ap_Paterno varchar(60) NOT NULL,  
5 ap_Materno varchar(60) NOT NULL,  
6 parentesco varchar(25),  
7 num_empleado int NOT NULL,  
8 CONSTRAINT dependiente_PK PRIMARY KEY(CURP),  
9 CONSTRAINT dependiente_empleado_FK FOREIGN KEY(num_empleado)  
10 REFERENCES EMPLEADO(num_empleado);
```

Tabla DEPENDIENTE

* Tabla CLIENTE: Se definen los datos necesarios para emitir la factura por parte del restaurante, mismos que podrán ser consultados posteriormente a través de la llave primaria que es el RFC del cliente compuesta por 13 dígitos:

* Tabla Contiene: Nos permite relacionar la orden con los diferentes platillos que se pueden solicitar dentro de una misma orden, por lo cual, hacemos referencia como llave primaria al folio de la orden y al id identificador de cada uno de los platillos solicitados:

```
● ● ●  
1 CREATE TABLE CLIENTE(  
2 RFC_CLIENTE varchar(13) NOT NULL,  
3 nombre varchar(60) NOT NULL,  
4 ap_Paterno varchar(60) NOT NULL,  
5 ap_Materno varchar(60) NOT NULL,  
6 razon_social varchar(100) NOT NULL,  
7 email varchar(100) NOT NULL,  
8 fecha_nacimiento date NOT NULL,  
9 calle varchar(60) ,  
10 numero smallint,  
11 colonia varchar(60),  
12 codigo_Postal int,  
13 estado varchar(60),  
14 CONSTRAINT cliente_PK PRIMARY KEY(RFC_CLIENTE));
```

Tabla CLIENTE

* Tabla ORDEN: Se definen sus atributos únicos referenciados a los precios y alimentos solicitados por el cliente. Definimos al folio como llave primaria ya que se genera uno distinto por orden solicitada, por su parte, se hace referencia al RFC del cliente como llave foránea, la cual nos permitirá enlazar los datos del cliente para la facturación de su cuenta:

* Tabla MENU CATEGORIA: En esta tabla se enuncian las características de los platillos, su categoría y el tipo de platillo o bebida que representa, cada uno de ellos tiene un id identificador el cual nos proporciona información clave para los cocineros como lo son su receta, disponibilidad y descripción:

```
1 CREATE TABLE contiene(
2 folio int NOT NULL,
3 id_identificador smallint NOT NULL,
4 CONSTRAINT contiene_menu_orden_PK PRIMARY KEY(folio,id_identificador),
5 CONSTRAINT contiene_menu_FK FOREIGN KEY(id_identificador)
6 REFERENCES MENU_CATEGORIA(id_identificador),
7 CONSTRAINT contiene_orden_FK FOREIGN KEY(folio)
8 REFERENCES ORDEN(folio));
```

Tabla Contiene

```
1 CREATE TABLE ORDEN(
2 folio int NOT NULL,
3 fecha date DEFAULT now() NOT NULL, --now()
4 total_Pago float NOT NULL,
5 cantidad_Alimentos int NOT NULL,
6 precio_unitario_Alimento float NOT NULL,
7 RFC_CLIENTE varchar(13) ,
8 CONSTRAINT orden_PK PRIMARY KEY(folio),
9 CONSTRAINT orden_cliente_FK FOREIGN KEY(RFC_CLIENTE)
10 REFERENCES CLIENTE(RFC_CLIENTE));
```

Tabla ORDEN

Finalmente, se define la secuencia que nos permite generar el número de empleado de forma automática, con esto, cada que se registre a un empleado, la base de datos le asignará un número de empleado definido conforme al orden con el cual se fueron registrando, por lo cual, hacemos referencia a la tabla EMPLEADO recordando que en ella se encuentra la generalización de todos los roles existentes dentro del restaurante:

2. DML

El lenguaje de manipulación de datos nos proporciona realizar la gestión de base de datos que concede a los usuarios y administrativos realizar tareas como consulta o manipulación de datos aprobados por el modelo definido anteriormente, dentro de este lenguaje tenemos las siguientes acciones permitidas:

- Insert: Agrega uno o más registros a una tabla de la base de datos
- Update: Es utilizada para modificar los valores de un conjunto de registros existentes.

```

1 CREATE TABLE MENU_CATEGORIA(
2   id_identificador smallint NOT NULL,
3   precio float NOT NULL,
4   receta varchar(100) NOT NULL,
5   nombre_alimento varchar(60) NOT NULL,
6   disponibilidad boolean NOT NULL,
7   nombre_categoria varchar(60) NOT NULL,
8   descripcion varchar(60) NOT NULL,
9   desc_Categoría varchar(60) NOT NULL,
10  tipo_categoria varchar(20) null,
11  nivel_de_dificultad smallint null,
12  sin_alcohol boolean null,
13  con_alcohol boolean null,
14  CONSTRAINT menu_PK PRIMARY KEY(id_identificador));

```

Tabla MENU CATEGORIA

```

1 CREATE SEQUENCE NUM_EMPLEADO
2 START WITH 1
3 INCREMENT BY 1;
4
5 ALTER TABLE EMPLEADO ALTER COLUMN num_empleado SET DEFAULT nextval('NUM_EMPLEADO');

```

Indice Número Empleado

- Delete: Borra uno o más registros existentes en una tabla.

A continuación realizamos algunas alteraciones dentro de las tablas, especialmente en las llaves foráneas, para poner los RESTRICT correspondientes. Iniciamos con la tabla DEPENDIENTE en la cual agregamos la llave foránea haciendo referencia a la tabla EMPLEADO con el atributo número de empleado. Además, se definen las restricciones de borrado como Restrict y la actualización de tipo Cascade.

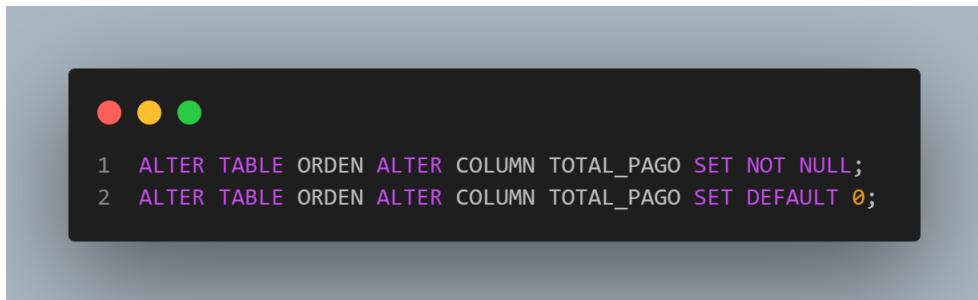
Posteriormente creamos otra referencia de llave foránea a la tabla ORDEN haciendo mención a la tabla CLIENTE a través del atributo RFC del cliente, definiendo de la misma forma el tipo de borrado como Restrict y la actualización de tipo Cascade.

Además se crea otro Constraint de tipo llave foránea en la tabla MESERO, creando la referencia a partir de la tabla EMPLEADO con el atributo número de empleado, de igual forma se define el tipo de borrado como Set Null y la actualización de datos de tipo Cascade.

```
● ● ●  
1 ALTER TABLE DEPENDIENTE DROP CONSTRAINT dependiente_empleado_FK;  
2 ALTER TABLE DEPENDIENTE ADD CONSTRAINT  
3 dependiente_empleado_FK FOREIGN KEY(num_empleado)  
4 REFERENCES EMPLEADO(num_empleado) ON DELETE RESTRICT ON UPDATE CASCADE;  
5  
6  
7 ALTER TABLE ORDEN DROP CONSTRAINT orden_cliente_FK;  
8 ALTER TABLE ORDEN ADD CONSTRAINT  
9 orden_cliente_FK FOREIGN KEY(RFC_CLIENTE)  
10 REFERENCES CLIENTE(RFC_CLIENTE) ON DELETE RESTRICT ON UPDATE CASCADE;  
11  
12 ALTER TABLE MESERO DROP CONSTRAINT mesero_empleado_FK;  
13 ALTER TABLE MESERO ADD CONSTRAINT  
14 mesero_empleado_FK FOREIGN KEY(num_empleado)  
15 REFERENCES empleado(num_empleado) ON DELETE SET NULL ON UPDATE CASCADE;
```

ALTER

Finalmente realizamos modificaciones en la tabla ORDEN con el objetivo de poder ir sumando a nuestra orden cada producto para realizar el total y generar cuentas independientes asociadas a cada orden:



```
1 ALTER TABLE ORDEN ALTER COLUMN TOTAL_PAGO SET NOT NULL;
2 ALTER TABLE ORDEN ALTER COLUMN TOTAL_PAGO SET DEFAULT 0;
```

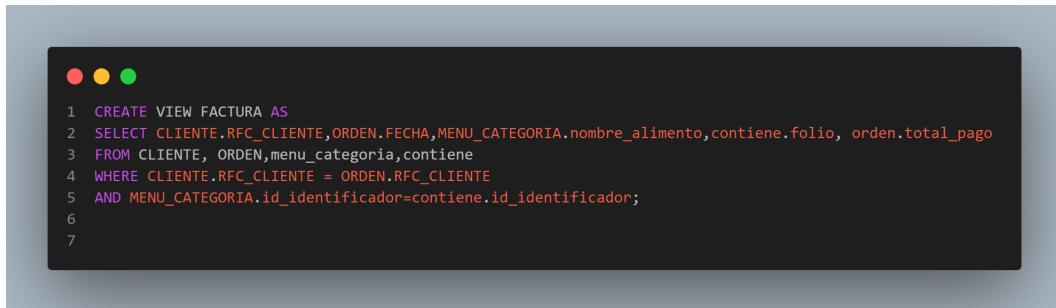
ALTER

3. Programación SQL

El lenguaje de consulta estructurada es un gestor para el manejo de la información en nuestra base de datos, nos permite comunicarnos con la base de datos, y también, realizar operaciones de acceso y manipulación de la información almacenada. Con este lenguaje podemos controlar todas las funciones que la base de datos ofrece a los usuarios brindando la seguridad de que se mantendrá la integridad de los datos.

Con esto, se definen las siguientes consultas y vistas, las cuales tienen un papel muy importante en el manejo de las vistas y la página web con la cual van a interactuar los usuarios finales:

Para la emisión de la factura, creamos una vista con las diferentes consultas y atributos necesarios para los datos de dicha factura, para esto se eligen ciertos datos específicos en los cuales se muestra la información necesaria ocultando la información personal y confidencial, lo anterior no puede ser consultado por todos los usuarios:



```
1 CREATE VIEW FACTURA AS
2 SELECT CLIENTE.RFC_CLIENTE,ORDEN.FECHA,MENU_CATEGORIA.nombre_alimento,contiene.folio, orden.total_pago
3 FROM CLIENTE, ORDEN,menu_categoria,contiene
4 WHERE CLIENTE.RFC_CLIENTE = ORDEN.RFC_CLIENTE
5 AND MENU_CATEGORIA.id_identificador=contiene.id_identificador;
6
7
```

Vista FACTURA

Se crea la siguiente función que cumple con el objetivo de sumar el total de una orden cada vez que se agregue un platillo a esta. Funciona con ayuda del Trigeer, en el cual se selecciona el precio y se actualiza con las nuevas órdenes solicitadas, revisando en todo momento que se seleccione el folio correspondiente al cliente asociado:

La función que se muestra a continuación nos permite actualizar el total de platillos dentro de la tabla ORDEN. Primero se convierte en Default el valor del total y cada vez que se haga una inserción

```
● ● ●
1 CREATE OR REPLACE FUNCTION sum_total()
2 RETURNS TRIGGER AS
3 $sum_total$
4 begin
5   UPDATE orden
6   SET TOTAL_PAGO=
7     SELECT SUM(PRECIO) FROM
8       menu_categoria JOIN CONTIENE ON menu_categoria.ID_IDENTIFICADOR=NEW.ID_IDENTIFICADOR
9     )
10    WHERE ORDEN.FOLIO=NEW.FOLIO;
11    RETURN NEW;
12 END;
13 $sum_total$ LANGUAGE plpgsql;
14
15
16
17 CREATE TRIGGER SUMA_ORDEN
18   AFTER INSERT
19   ON CONTIENE
20   FOR EACH ROW
21   EXECUTE PROCEDURE sum_total();
22
```

Función Suma

en “contiene” se actualiza el total existente para consultar la disponibilidad con la que cuenta el restaurante, misma que podrá confirmar si se puede realizar otra orden para un distinto cliente:

```

1 ALTER TABLE ORDEN ALTER COLUMN cantidad_alimentos SET NOT NULL;
2 ALTER TABLE ORDEN ALTER COLUMN cantidad_alimentos SET DEFAULT 0;
3
4 CREATE OR REPLACE FUNCTION sum_cantidad()
5 RETURNS TRIGGER AS
6 $sum_cantidad$
7 begin
8     UPDATE orden
9        SET CANTIDAD_ALIMENTOS=CANTIDAD_ALIMENTOS +1
10       WHERE ORDEN.FOLIO=NEW.FOLIO;
11   RETURN NEW;
12 END;
13 $sum_cantidad$ LANGUAGE plpgsql;
14
15 CREATE TRIGGER SUMA_CANTIDAD
16    AFTER INSERT
17    ON CONTIENE
18    FOR EACH ROW
19    EXECUTE PROCEDURE SUM_CANTIDAD();

```

Actualización de ORDEN

La siguiente función nos permite sumar el total de todos los pedidos consultados en un intervalo de tiempo, para esto se considera el id identificador de cada platillo, el cual se encuentra dentro de la tabla Contiene y se extraen los datos que coincidan en el intervalo de tiempo introducido por número de mes:

```

1 SELECT sum(precio) FROM
2 menu_categoria inner JOIN CONTIENE ON menu_categoria.ID_IDENTIFICADOR=CONTIENE.ID_IDENTIFICADOR
3 inner JOIN ORDEN ON CONTIENE.FOLIO=ORDEN.FOLIO
4 WHERE extract(month from orden.fecha)>=# and extract(month from orden.fecha)<#;
5

```

Consulta en intervalo de tiempo

La siguiente vista nos permite conocer cuál es el platillo que ha sido pedido más veces. En la vista se muestra el id identificador de dicho platillo así como su nombre asociado, para lograr esta consulta se suman los id identificadores diferenciando así cuál es el que más se solicita dentro de las órdenes:

```
● ● ●  
1 CREATE VIEW PLATILLOMAIN AS  
2 SELECT C.ID_IDENTIFICADOR,M.NOMBRE_ALIMENTO  
3 FROM CONTIENE as C  
4 INNER JOIN MENU_CATEGORIA AS M ON C.ID_IDENTIFICADOR=M.ID_IDENTIFICADOR  
5 GROUP BY C.ID_IDENTIFICADOR, M.ID_IDENTIFICADOR HAVING COUNT(C.ID_IDENTIFICADOR)=MAX(C.ID_IDENTIFICADOR);  
6
```

Platillo más vendido

Finalmente se crea una consulta con el fin de verificar si alguno de los platos no está disponible, en este caso nos mostrará aquellos que no tengan ninguna unidad almacenada registrada dentro de la tabla MENU CATEGORIA:

```
● ● ●  
1 SELECT*FROM MENU_CATEGORIA WHERE DISPONIBILIDAD='0';
```

Disponibilidad de unidades

5. Aplicación Web

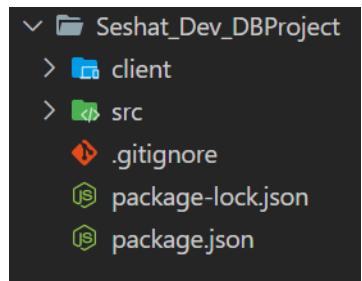
Existen diferentes niveles de automatización que nos ayudan a visualizar e interactuar con los datos, esto puede ser la implementación de un sitio web dinámico que incluya las facilidades de procesamiento de transacciones de los datos almacenados por alguna organización.

Para que nuestro cliente cumpla con las actividades cotidianas en el desempeño de sus funciones laborales, se crea la siguiente página web encargada de la interacción con el cliente para facilidad y eficiencia de procesamiento:

1. Estructura

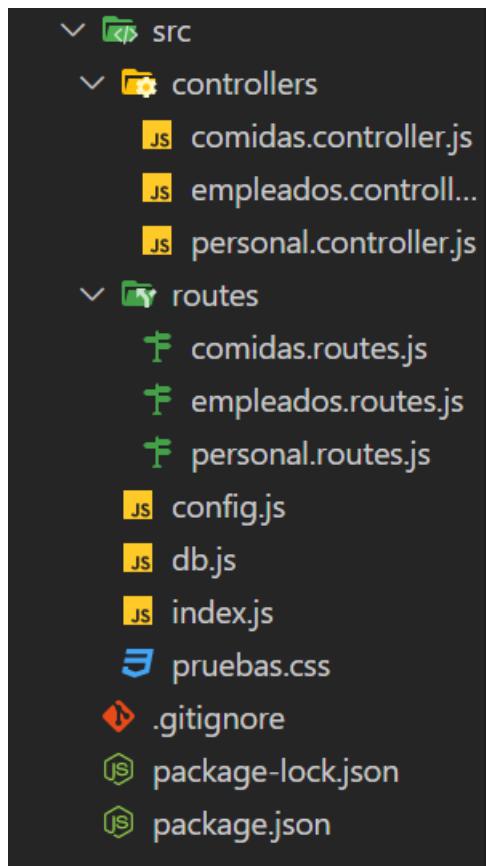
La aplicación fue realizada principalmente con NodeJs para el BackEnd y las conexiones; también se usó ReactJs para el FrontEnd.

La estructura principal de la pagina es la siguiente:



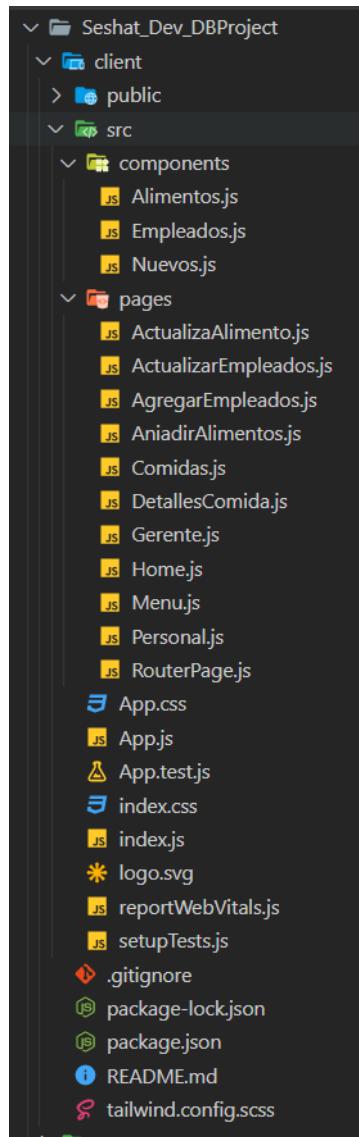
Estructura Global

En donde en la parte de SRC está todo el código del Backend. Este sistema recoge información de los usuarios u otros sistemas de tratamiento de datos en la compañía. Es el encargado de gestionar la información que proporciona el usuario recogida por el sitio web:



BackEnd

Y en client está todo lo del FrontEnd. Es la parte de una web que conecta e interactúa con los usuarios que la visitan. Es la parte visible, la que muestra el diseño, los contenidos y la que permite a los visitantes navegar por las diferentes páginas mientras lo deseen:



FrontEnd

2. BackEnd

En los package se crean y enrutan las dependencias y demás para nuestra conexión con Node. Estos archivos se encuentran tanto en BackEnd como en el FrontEnd.

En SRC se encuentra todo el código necesario. Principalmente tenemos los archivos fuera de las rutas y los controladores, aquí es dónde se hacen las conexiones.

- config.js



```
1 const { config } = require('dotenv')
2 config()
3
4 module.exports = {
5     bd: {
6         user: process.env.DB_USER,
7         password: process.env.DB_PASSWORD,
8         host: process.env.DB_HOST,
9         port: process.env.DB_PORT,
10        database: process.env.DB_NAME
11    }
12 }
```

SRC

Enlazamos y exportamos la variable bd, para hacer nuestra conexión después con la base de datos, hacemos este procedimiento de un process principalmente por seguridad.

- db.js

```
● ○ ●
1 const { Pool } = require('pg') // Se importa el modulo pg de node para comunicar el back end
2                                         // y la base de datos
3
4 const {bd} = require('./config') // Y se importa el archivo .config que contiene el
5                                         // acceso a las variables de entorno que dan acceso a la base
6
7 const pool = new Pool({ // Se establecen los parametros para la conexión de la base
8     user: bd.user,           // Por seguridad se añadieron los parametros en las variables de entorno
9     password: bd.password,
10    host: bd.host,
11    port: bd.port,
12    database: bd.database,
13    ssl: {
14        rejectUnauthorized: false,
15    }
16 })
17
18 module.exports = pool; // se exporta el modulo configurado
```

Enlace y Exportación

Creamos la conexión de la base de datos con nuestra app.
Conectamos todas las rutas e iniciamos el servidor.

- index.js



```
1  /* Modulos */
2
3  const express = require('express');
4  const morgan = require('morgan');
5  const cors = require('cors')
6  const fileUpload = require('express-fileupload')
7
8  /* Endpoints */
9
10 const personalRoutes = require('./routes/personal.routes');
11 const comidasRoutes = require('./routes/comidas.routes');
12 const empleadosRoutes = require('./routes/empleados.routes');
13 const ordenesRoutes = require('./routes/ordenes.routes');
14 const datosRoutes = require('./routes/datos.routes')
15 const imagesRoutes = require('./routes/imagenes.routes')
16
17 const app = express();
18
19 app.use(morgan('dev'))
20 app.use(express.json())
21 app.use(cors());
22 app.use(fileUpload());
23
24 app.use(personalRoutes)
25 app.use(comidasRoutes)
26 app.use(empleadosRoutes)
27 app.use(ordenesRoutes)
28 app.use(datosRoutes)
29 app.use(imagesRoutes)
30
31 app.listen(4000)
32 console.log('Server on port 4000')
```

Conexión

Controladores y rutas

Los controladores nos permiten realizar las operaciones de inserción, consulta, eliminación y actualización de datos para nuestra aplicación. Ahora, dentro de las rutas especificamos qué acción hacer con ese controlador cuando deseemos hacer alguna acción en la página. Este ejemplo lo veremos con nuestro controlador de empleado.

Como se puede observar cuando hacemos una consulta por ejemplo de empleados, hacemos una sentencia de SQL para mostrar a todos estos, y así, cuando los creamos, eliminamos y/o editamos.

Al final en rutas especificamos cada acción.

```

1 const pool = require('../db')
2
3 const getEmpleados = async (req, res) => {
4   try {
5     const empleados = await pool.query('SELECT * FROM empleado')
6     console.log(empleados)
7     res.json(empleados.rows)
8   } catch (error) {
9     console.log(error.message)
10   }
11 }
12
13 const getEmpleado = async (res, req) => {
14   try {
15     const { id } = req.params
16     const empleado = await pool.query('SELECT * FROM empleado WHERE num_empleado=$1', [id])
17     res.json(empleado.rows[0])
18   } catch (error) {
19     console.log(error.message)
20   }
21 }
22
23 const createEmpleado = async (req, res) => {
24   try {
25     const { edad, sueldo, fecha_nacimiento, rfc_empleado, nombre, ap_paterno, ap_materno, calle, numero, colonia, codigo_postal, estado } = req.body
26     const result = await pool.query(`INSERT INTO empleado (edad, sueldo, fecha_nacimiento, rfc_empleado, nombre, ap_paterno, ap_materno, calle, numero, colonia, codigo_postal, estado) VALUES ($1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12)`, [
27       edad, sueldo, fecha_nacimiento, rfc_empleado, nombre, ap_paterno, ap_materno, calle, numero, colonia, codigo_postal, estado
28     ])
29   } catch (error) {
30     console.log(error.message)
31   }
32 }
33
34 const updateEmpleado = async (req, res) => {
35   try {
36     const { id } = req.params
37     const { edad, sueldo, fecha_nacimiento, rfc_empleado, nombre, ap_paterno, ap_materno, calle, numero, colonia, codigo_postal, estado } = req.body
38     const result = await pool.query(`UPDATE empleado SET edad=$1, sueldo=$2, fecha_nacimiento=$3, rfc_empleado=$4, nombre=$5, ap_paterno=$6, ap_materno=$7, calle=$8, numero=$9, colonia=$10, codigo_postal=$11, estado=$12 WHERE num_empleado=$13`, [
39       id, edad, sueldo, fecha_nacimiento, rfc_empleado, nombre, ap_paterno, ap_materno, calle, numero, colonia, codigo_postal, estado, num_empleado
40     ])
41   } catch (error) {
42     console.log(error.message)
43   }
44 }
45
46 const deleteEmpleado = async (req, res) => {
47   try {
48     const { id } = req.params
49     const result = await pool.query(`DELETE FROM empleado WHERE num_empleado=$1`, [id])
50     res.sendStatus(204) // sin devolucion para todo en orden
51   } catch (error) {
52     console.log(error.message)
53   }
54 }
55
56
57 }
58
59 const deleteEmpleado = async (req, res) => {
60   try {
61     const { id } = req.params
62     const result = await pool.query(`DELETE FROM empleado WHERE num_empleado=$1`, [id])
63     res.sendStatus(204) // sin devolucion para todo en orden
64   } catch (error) {
65     console.log(error.message)
66   }
67 }
68
69 module.exports = {
70   getEmpleados,
71   getEmpleado,
72   createEmpleado,
73   updateEmpleado,
74   deleteEmpleado,
75 }
76 }

```

Controladores

3. FrontEnd

Como se mencionó anteriormente, la vista del cliente está realizada principalmente con React. Y en las diferentes carpetas que están dentro del directorio client, se puede observar de mejor manera la estructura de la aplicación.

Components

Aquí es un caso similar a la parte del BackEnd, pues nos relaciona la pagina del cliente con nuestro servidor, para saber qué debemos hacer específicamente con cada acción. Por ejemplo, en nuestro componente de empleados realizamos todas nuestras redirecciones y estructuramos la vista.

```
● ● ●
1 const { Router } = require('express')
2 const { getEmpleados, getEmpleado, createEmpleado, updateEmpleado, deleteEmpleado } = require('../controllers/empleados.controller')
3
4 const router = Router()
5
6 router.get('/Empleados', getEmpleados)
7
8 router.get('/Empleados/:name', getEmpleado)
9
10 router.post('/Empleados', createEmpleado)
11
12 router.put('/Empleado/:name', updateEmpleado)
13
14 router.delete('/Empleado/:name', deleteEmpleado)
15
16 module.exports = router;
```

Rutas

Y en pages tenemos de igual forma las vistas para cada vez que se haga una consulta o se busque dentro de nuestro servidor.

```

1 import React from 'react'
2
3 import { useEffect, useState } from 'react'
4 import { Splide, SplideSlide } from '@splidejs/react-splide'
5 import '@splidejs/splide/dist/css/splide.min.css'
6 import { Link } from 'react-router-dom'
7
8 function Alimentos() {
9
10   /* Es necesario utilizar los hooks useState y useEffect de react para mejorar la comunicación */
11
12   /* Se crean hooks para almacenar los json que va a mandar el backend y la base de datos */
13
14   const [listaComidas, setListaComidas] = useState([])
15
16   const [comidas, setComidas] = useState([
17     {
18       id_identificador: '',
19       precio: '',
20       receta: '',
21       nombre_alimento: '',
22       disponibilidad: '',
23       nombre_categoria: '',
24       descripcion: '',
25       desc_categoria: '',
26       tipo: '',
27       nivel_dificultad: '',
28       sin_alcohol: '',
29       con_alcohol: '',
30     }
31   ]);
32
33   /* uso de la base de datos en la página web */
34
35   const loadComidas = async () => { // esta función envía una solicitud GET al backend y el back end manda
36     // la respectiva query a la base de datos
37
38     const response = await fetch(`http://localhost:4000/comidas`)
39     const data = await response.json()
40     setListaComidas(data) // así ya cargo los datos recibidos
41   }
42
43   useEffect(() => {
44     loadComidas() // esta función es necesaria para llenar los hooks correctamente
45   }, [])
46
47   /* Cuando se quiera realizar una creación, actualización o eliminación en la base
48   es necesario añadir el método que se quiere realizar
49   DELETE ----> para DELETE
50   POST -----> para CREATE
51   PUT -----> para UPDATE */
52
53   const handleDelete = async (id) => {
54     const res = await fetch(`http://localhost:4000/comidas/${id}`, {
55       method: 'DELETE',
56     })
57     setListaComidas(comidas.filter(comida => comidas.id_comida !== id))
58   }
59
60   const handleUpdate = (id) => {
61     window.location.href = `/ActualizarAlimento/${id}`
62   }
63
64   const initAnadirAlimento = e => {
65     e.preventDefault()
66     window.location.href = '/AnadirAlimento'
67   }
68
69
70   return (
71     <div>
72       <div className="comidas_container">
73         <h3>Lista de alimentos</h3>
74
75         <div>
76           <input type="button" className="Agregar_btn" value="Aregar" onClick={initAnadirAlimento} />
77         </div>
78
79         <Splide options={{>
80           perPage: 4,
81           arrows: false,
82           dots: false,          /* Se puede ver como la información proporcionada */
83           drag: 'free',          /* viene directamente de los datos que arrastra la base */
84           gap: '5rem',
85         }}>
86           {
87             listaComidas.map((comidas) => (
88               <React.Fragment key={comidas.id_identificador}>
89                 <SplideSlide key={comidas.id_identificador}>
90                   <div className="alimento-card">
91                     <div><comidas.nombre_alimento/>
92                     <div><comidas.descripcion/>
93                     <div><comidas.nivel_dificultad/>
94                     <div><comidas.con_alcohol/>
95                     <div><comidas.sin_alcohol/>
96                     <div><comidas.precio/>
97                     <div><comidas.id_comida/>
98                     <div><comidas.id_identificador/>
99                     <div><comidas.receta/>
100                     <div><comidas.nombre_categoria/>
101                     <div><comidas.descripcion/>
102                     <div><comidas.nivel_dificultad/>
103                     <div><comidas.con_alcohol/>
104                     <div><comidas.sin_alcohol/>
105                     <div><comidas.precio/>
106                     <div><comidas.id_comida/>
107                     <div><comidas.id_identificador/>
108                     <div><comidas.receta/>
109                     <div><comidas.nombre_categoria/>
110                     <div><comidas.descripcion/>
111                     <div><comidas.nivel_dificultad/>
112                     <div><comidas.con_alcohol/>
113                     <div><comidas.sin_alcohol/>
114                   </div>
115                 </SplideSlide>
116               </React.Fragment>
117             ))
118           )
119         </Splide>
120       </div>
121     </div>
122   )
123 }
124
125 export default Alimentos

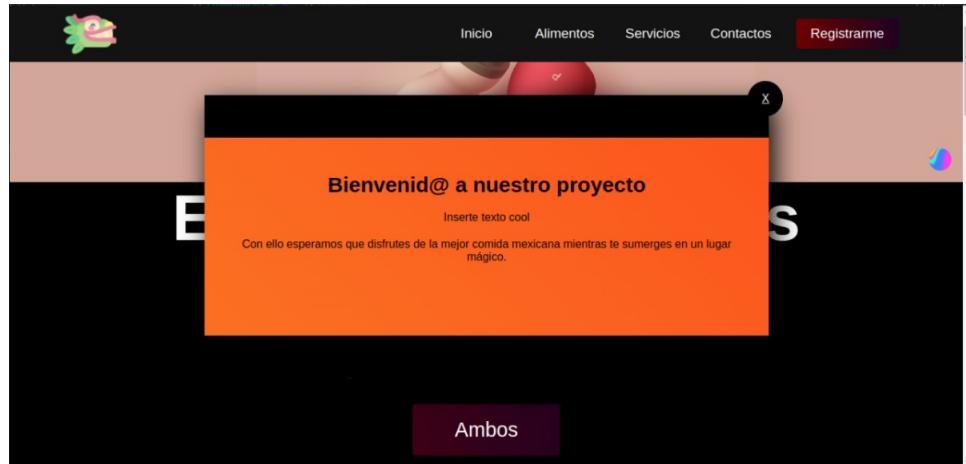
```

4. Funcionamiento

A continuación se explicarán las vistas que tiene la aplicación.

Inicio

Al entrar al sitio web tenemos una pequeña presentación, informando que es nuestro proyecto final. Despues hay una vista principal de la página, donde se habla un poco del concepto del restaurante, entre otras cosas.



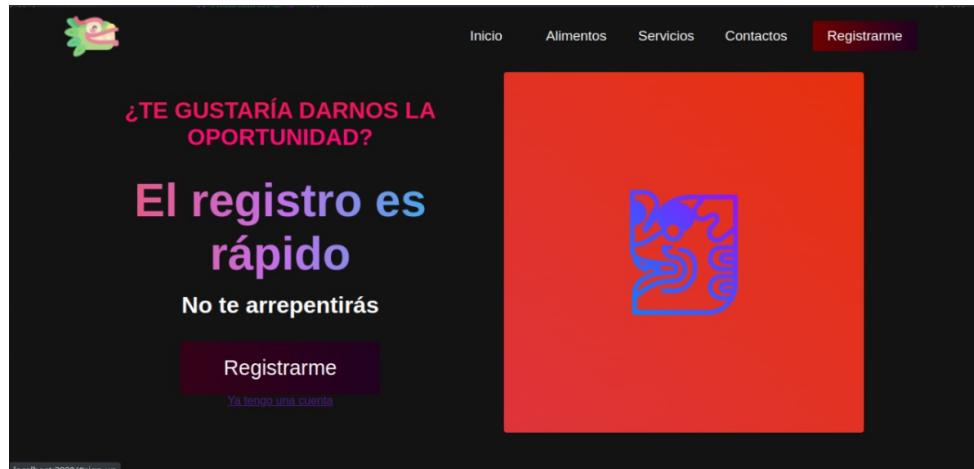
Vista Principal



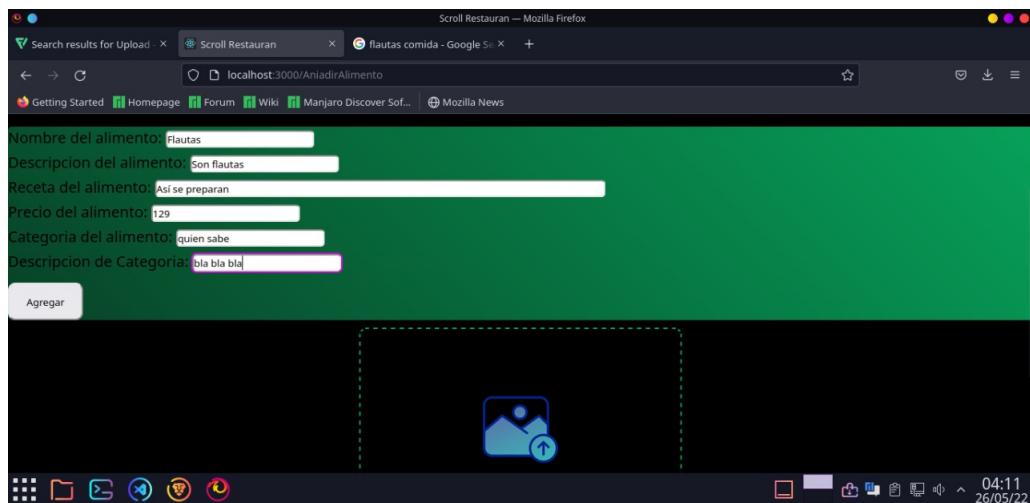
Página de inicio

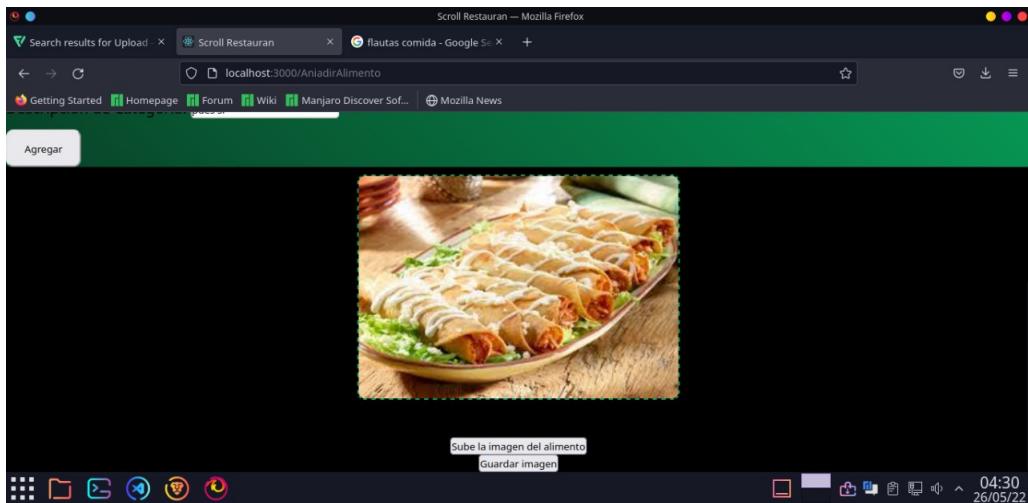
Registro

Se permite registrar dentro de la página a un usuario nuevo para poder ordenar, si se trata de un administrador, se permite registrar alimentos y usuarios.



Página de opción de registro





Página de registro

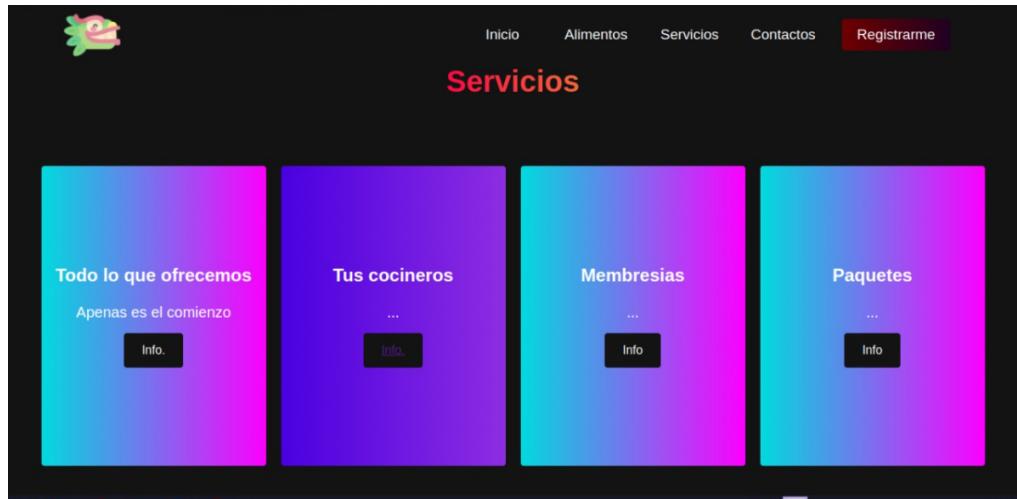
A screenshot of a registration form on a website. The form is titled "Gracias por tu elección" and contains fields for Name(s), Surname(s), Email, and Password. It also includes a checkbox for accepting terms and conditions and a "Registrar" button. A red sidebar on the right features a logo and text.

Página de formulario

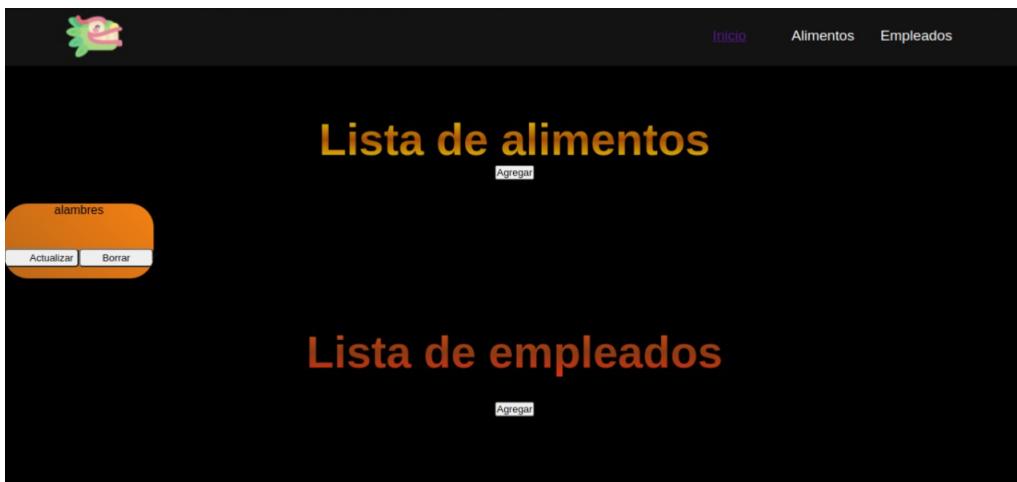
Además, la página ofrece la posibilidad de actualizar y modificar alimentos (solo a usuarios permitidos).

Listas y vistas

En la página, podemos listar todos los empleados, los platillos en general disponibles en la carta o realizar una selección específica de un platillo para observar su descripción general.



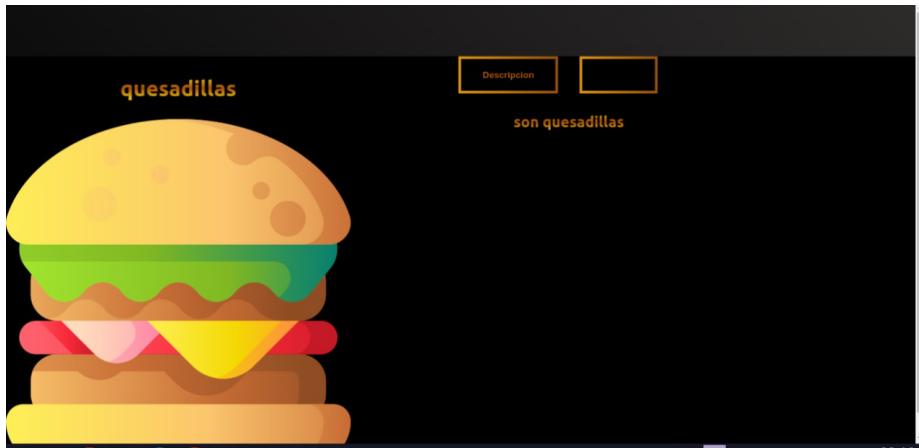
Servicios



Lista de alimentos

Carrito de compras

En nuestro carrito de compras vamos agregando nuestros platillo y nos aparece nuestro precio y fecha de la orden.



Descripción de alimento

Carrito de compras

La calidad de una base de datos es uno de los aspectos más importantes en el sector profesional, ya que de esta forma se puede asegurar la toma de decisiones y garantizar los mejores resultados de una empresa al contrastar, almacenar, verificar y actualizar datos.

Finalmente se entrega la base de datos con el objetivo de que, gestionada correctamente, el restaurante aumentará su eficacia, rapidez, agilidad y seguridad de los datos que se almacenen, y con todos estos factores, maximizar la productividad y funcionalidad del restaurante.

6. Conclusiones

Alcocer Velasco Abigail

Durante la elaboración de este proyecto se repasó la mayoría de los temas vistos en clase y por lo tanto, se aplicaron los conocimientos adquiridos para poder realizarlo. Para la construcción del proyecto fue necesario organizar el contenido que se iba a ejecutar y repartir el trabajo entre los miembros del equipo; a mi parecer el diseño más importante fue el del modelo entidad relación ya que a partir de este, lo demás se fue creando, afortunadamente se desarrolló correctamente. Asimismo, cada integrante hizo su parte y se colaboró entre todos para resolver dudas que habían. En cuanto al objetivo, desde mi punto de vista se cumplió ya que se creó una base de datos que pudiera ayudar al cliente a brindar un buen servicio a sus consumidores, y no solo eso, sino también para mantener su posición en el mercado e incrementar sus ventas. Por último, hablando generalmente de las bases de datos, queda mencionar que son muy importantes no solo a nivel personal sino ya a un nivel empresarial ya que mediante ellas se puede almacenar una cantidad grande de datos, ayudando a evitar la redundancia y mejorando la estructura de nuestras actividades a realizar.

Genis Cruz Lourdes Victoria

En este proyecto se analizaron una serie de requerimientos y mediante estos obtuvimos la solución aplicando los conceptos vistos en nuestro curso de Base de datos. Este proyecto se realizó con el objetivo de aprender más sobre las base de datos. Concluimos de forma práctica que una base de datos es un conjunto de información que está relacionada entre sí y que se encuentra agrupada, es decir, lleva una estructura, en este caso en particular visualizamos cómo es que se relacionaban las entidades que conforman nuestro restaurante (cliente, empleados, etc). Además podemos destacar algunos aspectos de gran peso como lo son el diseño y la creación de la base de datos, ya aquí es donde organizamos la información y representamos las relaciones que existen entre nuestros datos. Entonces las bases de datos son importantes en los negocios ya que nos ayudan a llevar el control de los movimientos, asimismo nos ayuda a ver el alcance del nuestro negocio. Ahora en cuanto a la organización y trabajo en equipo optamos por dividir el trabajo en áreas en donde cada integrante del equipo se dedicó al área en donde se sentía más fuerte.

González Cuellar Arturo

El desarrollo de este proyecto fue de gran importancia para un buen cierre del semestre ya que con este, se lograron aplicar todos los conceptos estudiados a lo largo del curso. Esta materia resulta tener muchos campos de aplicación y estudio por lo cual resulta sencillo visualizarlo en nuestra vida cotidiana, y el desarrollo de este proyecto solo fue una de esas múltiples aplicaciones. Me pareció interesante ver cómo todo está conectado y relacionado, si bien es cierto que puede parecer muy metódico, resulta sencillo aplicarlo y saber el procedimiento a seguir, es decir, que desde un principio logramos identificar los pasos que se tenían que seguir para el cumplimiento de los objetivos. Finalmente, puedo decir que se cumplieron los objetivos al realizar cada una de las actividades en tiempo y con una buena retroalimentación en la cual el equipo siempre estuvo con la disponibilidad y con la responsabilidad de trabajo para cumplir con los objetivos.

Maya Herrera Alexis Daniel

En todo el proceso de este proyecto se aprendió una gran cantidad de aspectos en lo relacionado a construir trabajos de programación más complejos. Sin duda implicó varios retos como el aprender a comunicar distintos lenguajes, entender la importancia de usar buenas prácticas para facilitar el trabajo en equipo, saber repartir la carga del trabajo no solo pensando en cantidades sino dandole prioridad a las habilidades individuales. En cuanto a la materia, resultó muy buen proyecto para fortalecer lo aprendido en clase y en equipo mantener un ambiente de apoyo en caso de que hubieran dudas en algún aspecto. En lo personal, la parte del MER y MR constituyen un porcentaje muy alto

en la definición del alcance del proyecto, en este caso resultó un muy buen diseño, prácticamente no hubo modificaciones en etapas posteriores lo cual nos brindó bastante tiempo. Al final se cumplió el objetivo del proyecto y a mi consideración es una propuesta con bastante potencial para seguir mejorando y reforzar más conocimientos.

Roldán Sánchez Alexis

Considero que, especialmente en nuestra área, no hay mejor manera de aprender y reforzar conocimientos que aplicando los temas aprendidos. En este caso la implementación de la base de datos en un proyecto es algo que debe de estar bien estructurado, planeado y trabajado para que en su implementación sea eficiente y se evite la mayor cantidad de inconvenientes. Me fue reconfortante el desarrollo de este proyecto, pues con un buen equipo y una buena retroalimentación se pudo ir trabajando cada una de las etapas del trabajo de manera eficiente, lo que evitó que se sintiera el tiempo encima, dando lo que considero yo, un muy buen resultado para la aplicación web. Me agradó que nos enfocáramos en las habilidades que mejor tuviéramos desarrolladas para el desarrollo del proyecto sin, claro, dejar de trabajar y aportar en otras áreas. Es un proyecto que todavía puede ser escalable, pues nos brinda esa posibilidad debido a sus grandes cimientos que tiene como base. Gracias a este proyecto, el reforzamiento de conceptos y habilidades incrementaron.

Concepto	Definición	Persona encargada
Modelo Entidad - Relación (MER)	Es una herramienta que sirve para diseñar y construir bases de datos.	González Cuellar Arturo y Roldán Sánchez Alexis
Modelo Relacional (MR)	Es un modelo basado en lógica de predicado y en teoría de conjuntos. Los datos se almacenan en tablas compuestas por filas, o tuplas, y columnas o campos.	Genis Cruz Lourdes Victoria y Alcocer Velasco Abigail
Lenguaje de definición de datos (DDL)	Es un lenguaje que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos.	Genis Cruz Lourdes Victoria y Alcocer Velasco Abigail
Lenguaje de manipulación de datos (DML)	Es un lenguaje por el cual se puede acceder y modificar los datos de la base de datos utilizando comandos como select, update, insert, delete, truncate, begin, commit y rollback.	Maya Herrera Alexis Daniel, Genis Cruz Lourdes Victoria y González Cuellar Arturo
Diseño de página web	En un desarrollo web, las bases de datos nos permiten actuar como almacén de datos, extraer el contenido de la web y disponerlo fácilmente, tal y como el desarrollador requiere en su proyecto web o el usuario necesita visualizarlo.	Maya Herrera Alexis Daniel
Implementación de la base de datos en la página	Una conexión a base de datos es un archivo de configuración donde se especifican los detalles físicos de una base de datos como por ejemplo el tipo de base de datos, la versión, y los parámetros que permiten una conexión.	Maya Herrera Alexis Daniel y Roldán Sánchez Alexis

Elaboración de la presentación	<p>Es una propuesta o actualización de un proyecto específico del profesional, ya sea que esté en proceso o en la etapa de planificación.</p> <p>Para que una presentación ejecutiva sea efectiva, la comunicación debe ser concisa y clara.</p>	Alcocer Velasco Abigail y Genis Cruz Lourdes Victoria
Documentación	<p>La documentación del proyecto incluye todos los documentos relevantes que se crean durante y para el propio proyecto. La documentación del proyecto actúa como contenedor de memoria para el curso del proyecto, las decisiones tomadas, los cambios de contenido y para gestionar el nivel de detalle del proyecto.</p>	Todos los integrantes del equipo
Exposición	<p>Es el producto presentado de un intenso trabajo de investigación y análisis metódico.</p>	Persona(as) elegida(as) por el profesor

Anexo: Tabla de responsabilidades