UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN, CONTROL Y EVALUACIÓN DE RECURSOS INFORMÁTICOS

LICENCIATURA EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN
SISTEMAS DE BASES DE DATOS II

PROYECTO FINAL

INTEGRANTES EQUIPO Nº6:

SAMUDIO, NEDITH 8-968-1471

SÁNCHEZ, ANA 8-967-832

SOLIS, MICHAEL 8-958-1219

TEJADA, ÁNGEL 8-969-974

URRIOLA, VICENTE 8-892-2296

DOCENTE: ING. HENRY J. LEZCANO P.

GRUPO: 1IF131

II SEMESTRE 2022

ÍNDICE

INTRODUCCIÓN 3

I PARTE – PROYECTO BASE DE DATOS (DISEÑO) 3

NOMBRE DEL PROYECTO DE BASE DE DATOS 4

MISIÓN DE LA BASE DE DATOS 4

OBJETIVOS DE LA MISIÓN DE LA BASE DE DATOS 4

DEFINICIÓN DEL PROYECTO 5

- **ÁMBITO** 5
- **ALCANCE O LÍMITES** 5

ANÁLISIS DE REQUERIMIENTOS 6

MODELADO DE LA BASE DE DATOS 7

- **MODELO CONCEPTUAL E/R** 7
- **MODELO LÓGICO RELACIÓN** 8
- **MODELO LÓGICO RELACIONAL NORMALIZADO** 9
- **MODELO FÍSICO SEGÚN SISTEMA DE GESTIÓN (Oracle)** 9

II PARTE – IMPLEMENTACIÓN DE BASE DE DATOS II 12

IMPLEMENTACIÓN DE PROCEDIMIENTOS, FUNCIONES, TRIGGERS DE LA BASE DE DATOS 12

ANEXO 29

CONCLUSIÓN 31

REFERENCIAS 32

INTRODUCCIÓN

Hoy en día, los modelos de bases de datos representan un estándar en la administración de la información referente al negocio, con esto en mente se propone desarrollar una solución tecnológica que pueda resolver una problemática emergente en cuanto al servicio brindado en la cafetería de la universidad tecnológica. La cafetería proporciona un servicio importante para el bienestar estudiantil, siendo esta la principal fuente de alimentación para el estudiante promedio dentro del campus universitario, de igual forma el personal docente y administrativo se beneficia de este servicio, con este enfoque en mente podemos determinar que el segmento analizado en esta investigación representa un servicio primordial para la gran mayoría de usuarios en la UTP (estudiantes, profesores y administrativos). Identificando el escenario anterior, la propuesta consiste en la creación de un sistema de base de datos que pueda proveer de una solución tecnológica a la cafetería universitaria, que a su vez pueda optimizar los procesos de generación de órdenes y reservas, otro aspecto valioso del uso de la base de datos es la posibilidad de gestionar con mayor detalle el inventario y los recursos utilizados por la propia cafetería, de esta manera se logra optimizar los recursos y la generación de informes de satisfacción de clientes con los que se pueden obtener los productos más consumidos y en su contratarte los menos gastados por los clientes, para una toma de decisiones asertivas en cuanto al negocio. Es importante destacar que la implementación del sistema de base de datos no se realiza de la noche a la mañana, hay que realizar la carga de los datos, conocer como es el ambiente en el que se desenvuelven las entidades y como pueden optimizarse los procesos dentro de la cafetería, es recomendable como en muchos casos empezar con un modelo de práctica que nos pueda brindar una visión general de como podrá comportarse el sistema. Considerar el modelo de base de datos como una herramienta medidora de desempeño es una de las muchas posibilidades que brinda la gestión de bases de datos, no sólo conocer cómo se relacionan las entidades del negocio, sino también conocer como es el desarrollo y desempeño de este, a lo largo del ciclo de vida de la base de datos. La gestión de recursos y optimización son las bases primordiales de este proyecto.

I PARTE – PROYECTO BASE DE DATOS (DISEÑO)

NOMBRE DEL PROYECTO DE BASE DE DATOS

El nombre de este proyecto es Base de Datos para Cafetería de la Universidad Tecnológica de Panamá, en específico la del Edificio Nº3, la idea ha surgido ya que nosotros como estudiantes hemos presenciado que hay falta de organización en las horas de comida, supongamos que es una cafetería nueva, más pequeña y que dispone de alimentos diferentes a la que ya está disponible en el edificio.

MISIÓN DE LA BASE DE DATOS

Su misión es poder llevar un control como en muchos restaurantes populares, en estos los meseros tienden a perder mesas o llamadas de los clientes durante las horas pico potencialmente disminuir la clientela, así como el método tradicional de agitar la mano para llamar a los servicios es ineficiente que a menudo conduce a muchas quejas. Si bien este es un problema continuo, todavía no hay ningún producto que mejora drásticamente la comunicación entre los servidores y los clientes en el mercado actual. Ya que se cuenta con muchos estudiantes, administrativos y profesores, tener una cafetería nueva no sería mala idea que esta se pueda manejar diferente y así la que ya está en el edificio pueda aplicar el mismo servicio.

OBJETIVOS DE LA MISIÓN DE LA BASE DE DATOS

El objetivo del proyecto es mejorar la eficiencia y administrar la cafetería nueva de una manera mejor y más fácil. Es mucho más eficiente disponer de una base de datos donde el gerente pueda obtener los datos generales de los clientes, varios administrativos cuentan con un correo donde pueden hacer su pedido y solo pasar

a recogerlo en la cafetería, varios de los clientes no pueden contar con un tiempo para comer en la mesa, pero piden su comida para llevar. Se espera tener un conteo de los alimentos para que todos los clientes puedan satisfacer su pedido. También se podremos ver las reservas que se hagan, y se busca que todos los empleados cumplan su labor para que la labor sea más rápido y organizado.

DEFINICIÓN DEL PROYECTO

ÁMBITO

Referente a la misión de este proyecto, queremos lograr el objetivo de diseñar un sistema en el que los clientes puedan llamar a sus meseros fácilmente y ayudar a la cafetería a aumentar la eficiencia general. Es fácil y efectivo administrar una cafetería utilizando una base de datos. Una base de datos de restaurante contiene información de clientes, información de empleados, detalles de alimentos, detalles de pedidos y datos esenciales, así mismo lo haremos en base a la cafetería, ya que sus clientes somo nosotros los estudiantes, profesores, administrativos, entre otros. Mediante el uso de una base de datos, el gerente de la cafetería puede analizar fácilmente los datos anteriores y tomar una decisión en unos segundos que garantice una gestión adecuada. Esta base de datos para la gestión de restaurantes, cafeterías, hasta una dulcería puede monitorear el flujo de trabajo de cualquier empleado del lugar, lo que garantiza la correcta gestión de la cafetería.

ALCANCE O LÍMITES

Estos son los alcances que podemos mencionar:

- Ver los pedidos de la cafetería y ayuda a tener los pedidos a tiempo.
- Mediante esta base de datos podemos supervisar la actividad de los empleados.

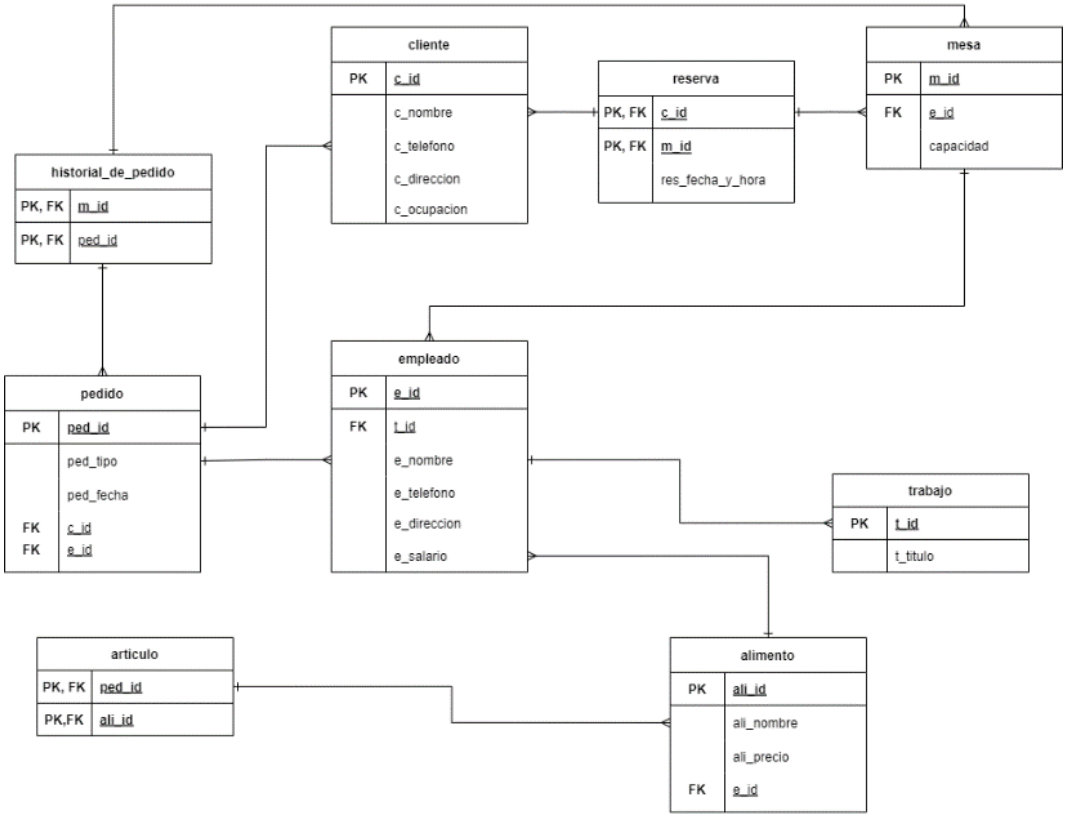
- Podemos determinar cuáles son los alimentos que más se utilizan y los que menos se utilizan.
- Esta base de datos puede ayudar al gerente a analizar los datos anteriores rápidamente y tomar decisiones eficaces.

ANÁLISIS DE REQUERIMIENTOS

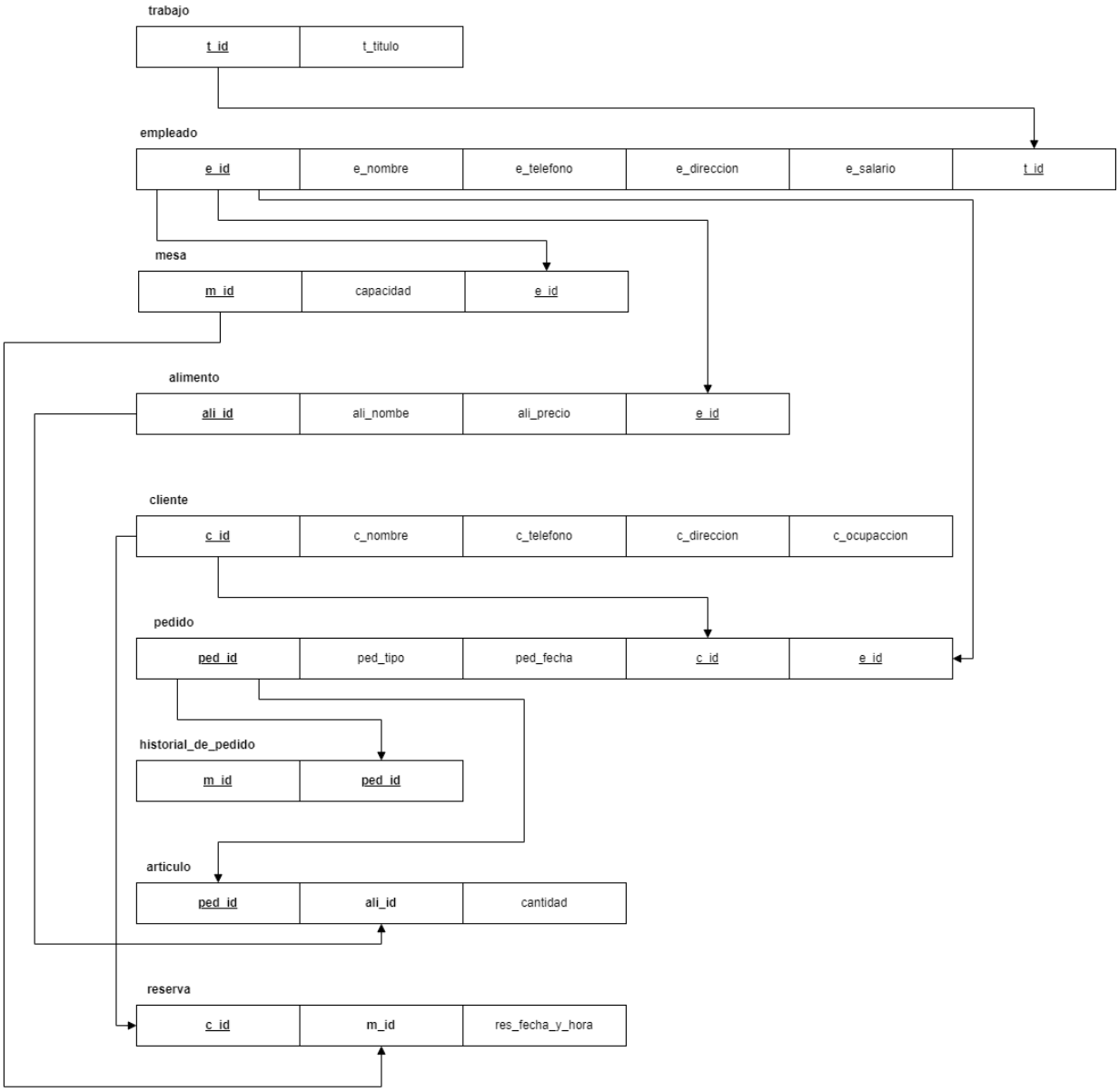
- Se debe tener una tabla empleados que almacene el id, el nombre, el teléfono, la dirección y salario.
- Una tabla cliente que contenga los atributos id del cliente, nombre, teléfono, dirección y ocupación.
- Una tabla de alimentos que contenga el id del alimento, el nombre y el precio.
- Una tabla pedidos que guarden el id del pedido, el tipo y la fecha.
- Para gestionar las mesas se debe almacenar el id de la mesa y la capacidad.
- Un empleado puede atender varios pedidos y un cliente puede hacer varios pedidos.
- Se debe tener un historial de las mesas y los pedidos realizado.

MODELADO DE LA BASE DE DATOS

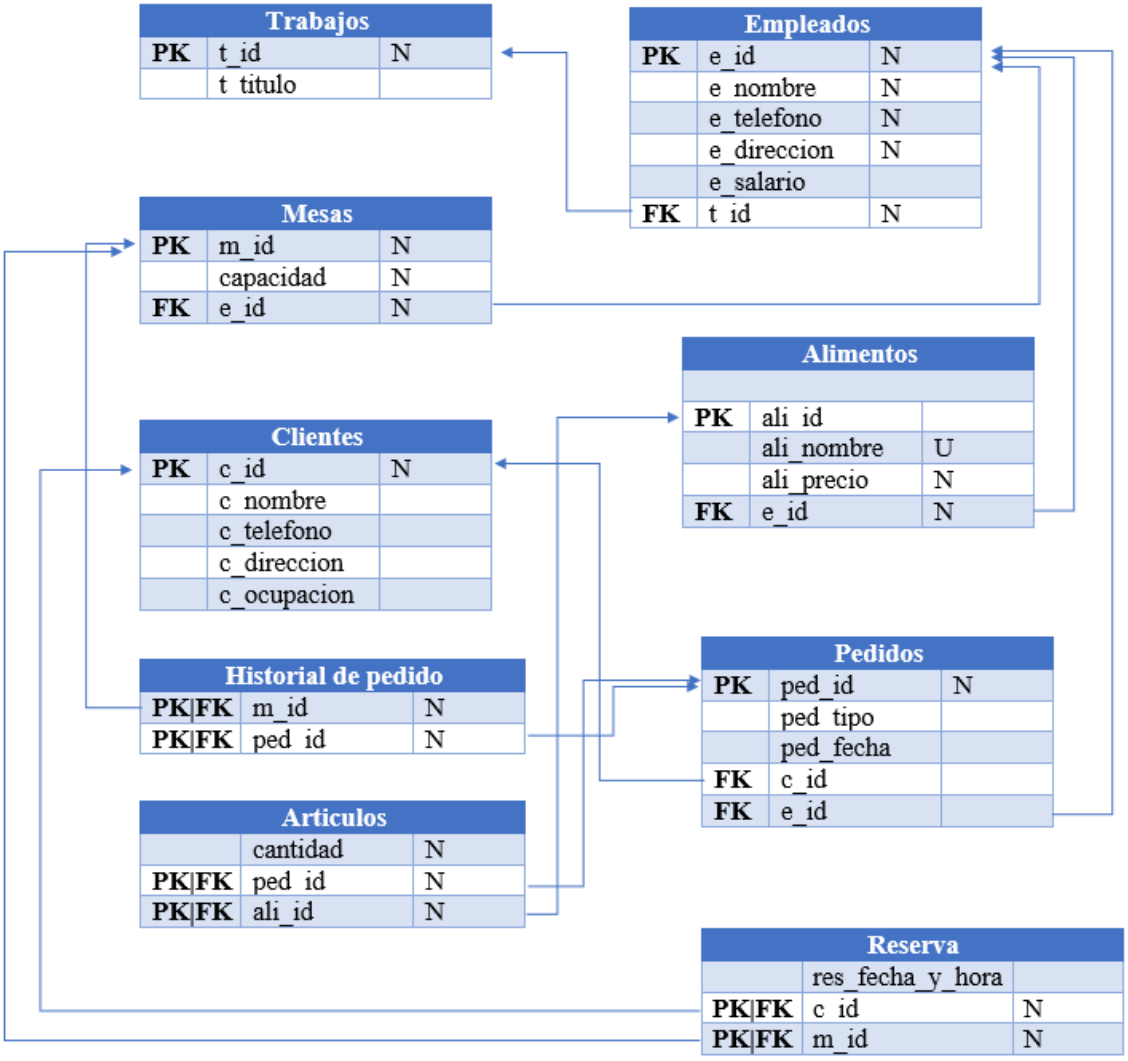
MODELO CONCEPTUAL E/R



MODELO LÓGICO RELACIONAL



MODELO LÓGICO RELACIONAL NORMALIZADO



MODELO FÍSICO SEGÚN SISTEMA DE GESTIÓN (Oracle)

```
-- set autocommit on;


/*****************************************************************************/
/* 1.- Creación de tablas                                                    */
/*****************************************************************************/


create table trabajo(
t_id number (5) not null,
t_titulo varchar2 (20),
```

```
constraint trabajo_pk primary key (t_id)
);

create table empleado(
e_id number (7) not null,
e_nombre varchar2 (20) not null,
e_telefono number (11)not null,
e_direccion varchar2 (30) not null,
e_salario number (6,2),
t_id number (5) not null,
constraint e_id_pk primary key (e_id),
constraint e_t_id_fk foreign key (t_id) references trabajo (t_id)
);

create table mesa(
m_id number (5) not null,
capacidad number (2) not null,
e_id number (7) not null,
constraint mesa_pk primary key (m_id),
constraint mesa_fk foreign key (e_id) references empleado (e_id)
);

create table alimento(
ali_id number (7) not null,
ali_nombre varchar2 (15) unique,
ali_precio number (5,2) not null,
e_id number (7) not null,
constraint alimento_pk primary key (ali_id),
constraint alimento_fk foreign key (e_id) references empleado (e_id)
);

create table cliente(
```

```
c_id number (7) not null,
c_nombre varchar2 (25),
c_telefono number (11),
c_direccion varchar2 (32),
c_ocupacion varchar2 (16),
constraint cliente_pk primary key (c_id)
);

create table pedido(
ped_id number (7) not null,
ped_tipo varchar2 (20),
ped_fecha date default sysdate,
c_id number (7),
e_id number (7),
constraint pedido_pk primary key (ped_id),
constraint clientes_ped_fk foreign key (c_id) references cliente (c_id),
constraint pedidos_fk foreign key (e_id) references empleado (e_id)
);

create table historial_de_pedido(
m_id number (5) not null,
ped_id number (7),
constraint historial_de_pedido_pk primary key (m_id, ped_id),
constraint historial_de_pedido_fk1 foreign key (m_id) references mesa
(m_id),
constraint historial_de_pedido_fk2 foreign key (ped_id) references pedido
(ped_id)
);

create table articulo(
cantidad number (4) not null,
ped_id number (7),
ali_id number (7),
```

```
constraint articulos_pk primary key (ped_id, ali_id),
constraint articulos_fk1 foreign key (ped_id) references pedido (ped_id),
constraint articulos_fk2 foreign key (ali_id) references alimento
(ali_id)
);


create table reserva(
res_fecha_y_hora timestamp,
c_id number (7),
m_id number (5),
constraint reserva_pk primary key (c_id, m_id),
constraint reserva_fk1 foreign key (c_id) references cliente(c_id),
constraint reserva_fk2 foreign key (m_id) references mesa (m_id)
);

-- Tabla de auditoria empleado
create table aud_empleado (
emp_id number,
trig varchar(50),
nombre varchar(20),
telefono number,
salario number(6,2)
);
```

II PARTE – IMPLEMENTACIÓN DE BASE DE DATOS II

IMPLEMENTACIÓN DE PROCEDIMIENTOS, FUNCIONES, TRIGGERS DE LA BASE DE DATOS

```
/*****************************************************/
/* 2.- Crear secuencias                              */
```

```
/*******************************************************/


create sequence sec_trabajo start with 1 increment by 1 maxvalue 5
minvalue 1 nocycle;

create sequence sec_cliente start with 1 increment by 1;

create sequence sec_empleado start with 1 increment by 1 maxvalue 8
minvalue 1 nocycle;

create sequence sec_alimento start with 1 increment by 1 maxvalue 4
minvalue 1 nocycle;

create sequence sec_pedido start with 1 increment by 1;

create sequence sec_mesa start with 1 increment by 1 maxvalue 6 minvalue
1 nocycle;

create sequence sec_aud_tot_pagar start with 1 increment by 1;




/*******************************************************/

/* 3.- Procedimientos para la inserción de datos PL/SQL */

/*******************************************************/

-- Tabla trabajo


-- Tenemos 4 tipos de trabajos, los cuales son:

-- 1.despachador

-- 2.cocinero

-- 3.limpiador

-- 4. cajero

create or replace procedure agregar_trabajo (

p_t_titulo trabajo.t_titulo%type

)

as

begin

insert into trabajo (t_id,t_titulo)

values (sec_trabajo.nextval,p_t_titulo);

exception

when dup_val_on_index then
```

```
dbms_output.put_line('Datos repetidos');

when value_error then

dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');

when others then

dbms_output.put_line('Error, ha ocurrido algo inesperado');

end agregar_trabajo;

/

-- Tabla clientes

create or replace procedure agregar_clientes(

p_c_nombre cliente.c_nombre%type,

p_c_telefono cliente.c_telefono%type,

p_c_direccion cliente.c_direccion%type,

p_c_ocupacion cliente.c_ocupacion%type

)

as

begin

insert into cliente (c_id,c_nombre,c_telefono,c_direccion,c_ocupacion)

values(sec_cliente.nextval,p_c_nombre,p_c_telefono,p_c_direccion,p_c_ocup
acion);

exception

when dup_val_on_index then

dbms_output.put_line('Datos repetidos');

when value_error then

dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');

when others then

dbms_output.put_line('Error, ha ocurrido algo inesperado');

end agregar_clientes;

/

-- Tabla empleados

create or replace procedure agregar_empleado(

p_e_nombre empleado.e_nombre%type,
```

```
p_e_telefono empleado.e_telefono%type,
p_e_direccion empleado.e_direccion%type,
p_e_salario empleado.e_salario%type,
p_t_id empleado.t_id%type
)
as
begin
insert into empleado(e_id,e_nombre,e_telefono,e_direccion,e_salario,t_id)
values(sec_empleado.nextval,p_e_nombre,p_e_telefono,p_e_direccion,p_e_sal
ario,p_t_id);
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end agregar_empleado;
/
-- Tabla alimentos
create or replace procedure agregar_alimentos (
p_ali_nombre alimento.ali_nombre%type,
p_ali_precio alimento.ali_precio%type,
p_e_id alimento.e_id%type
)
as
begin
insert into alimento(ali_id,ali_nombre,ali_precio,e_id)
values(sec_alimento.nextval,p_ali_nombre,p_ali_precio,p_e_id);
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
```

```
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end agregar_alimentos;
/
-- Tabla mesas

-- No son mesas infinitas.
create or replace procedure agregar_mesas(
p_capacidad mesa.capacidad%type,
p_e_id mesa.e_id%type
)
as
begin
insert into mesa (m_id,capacidad,e_id)
values(sec_mesa.nextval,p_capacidad,p_e_id);
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end agregar_mesas;
/
-- Tabla reserva
create or replace procedure agregar_reserva (
p_res_fecha_y_hora reserva.res_fecha_y_hora%type,
p_c_id reserva.c_id%type,
p_m_id reserva.m_id%type
```

```
)
as
begin
insert into reserva (res_fecha_y_hora,c_id,m_id)
values(p_res_fecha_y_hora,p_c_id,p_m_id);
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end agregar_reserva;
/


-- Tabla pedidos
create or replace procedure agregar_pedido (
p_ped_tipo pedido.ped_tipo%type,
p_ped_fecha pedido.ped_fecha%type,
p_c_id pedido.c_id%type,
p_e_id pedido.e_id%type
)
as
begin
insert into pedido (ped_id,ped_tipo,ped_fecha,c_id,e_id)
values (sec_pedido.nextval,p_ped_tipo,p_ped_fecha,p_c_id,p_e_id);
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
```

```
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end agregar_pedido;
/
-- Tabla articulos
create or replace procedure agregar_articulos (
p_cantidad articulo.cantidad%type,
p_ped_id articulo.ped_id%type,
p_ali_id articulo.ali_id%type
)
as
begin
insert into articulo (cantidad,ped_id,ali_id)
values (p_cantidad,p_ped_id,p_ali_id);
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end agregar_articulos;
/

-- Tabla historial_de_pedidos
create or replace procedure agregar_historial_de_pedidos (
p_m_id historial_de_pedido.m_id%type,
p_ped_id historial_de_pedido.ped_id%type
)
as
begin
```

```
insert into historial_de_pedido(m_id,ped_id)

values(p_m_id,p_ped_id);

exception

when dup_val_on_index then

dbms_output.put_line('Datos repetidos');

when value_error then

dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');

when others then

dbms_output.put_line('Error, ha ocurrido algo inesperado');

end agregar_historial_de_pedidos;

/

/*************************************************************/

/* 4.- Llamada a los procedimientos para inserción de datos */

/*************************************************************/

-- Trabajos

begin

agregar_trabajo('despachador');

agregar_trabajo('cocinero');

agregar_trabajo('limpiador');

agregar_trabajo('cajero');

end;

/

-- ******************************************************

-- Clientes

begin

agregar_clientes('Daniel',62514878,'Tocumen,La siesta', 'estudiante');

agregar_clientes('Kevin',65544879,'San Miguelito,Brisas del golf',
'profesores');

agregar_clientes('Mario',62555888,'San Miguelito,Villa Lucre',
'administrativos');

end;

/
```

```
-- ******************************************************

-- Empleados

-- la última columna es el tipo de trabajo [1.Despachador, 2.cocinero,
3.limpiador ,4. cajero]

begin

agregar_empleado('Gabriel',65458798,'Las colinas', 400.00,1); --
Despachador

agregar_empleado('Johel',62457532,'Paraiso', 650.00,2); -- cocinero

agregar_empleado('Ana',62457532,'Paraiso', 1050.00,3); -- limpiador

agregar_empleado('Natalia',62457532,'Villa lucre', 950.00,4); -- Cajero

agregar_empleado('Dany',61254385,'Villa lucre', 650.00,2); -- cocinero

agregar_empleado('David',62574893,'Cerro viento', 400.00,1); --
Despechador

agregar_empleado('Jose',62554894,'Cincuentenario', 950.00,4); -- Cajero

agregar_empleado('Juan',62554841,'Chorrillo', 1050.00,3); -- Limpiador

end;

/


-- ******************************************************

-- Alimentos a consumir


-- agregar_alimentos(nombre del alimento, precio del alimento,
ID_empleado de la tabla alimentos)

begin

agregar_alimentos('Hamburguesa',2.00,2);

agregar_alimentos('Emparedado',2.00,5);

agregar_alimentos('Pizza',2.85,2);

agregar_alimentos('Hojaldre',0.85,5);

end;

/


-- ******************************************************

-- Son 6 mesas enumeradas del 1 al 6 con secuencia.
```

```
-- agregar_mesas(numero de mesa automático,cantidad de personas, número
de empleado)

begin

agregar_mesas(10,3); -- la mesa 1 nada más puede tener 10 personas como
máximo y es atendido por el empleado 3

agregar_mesas(10,3); -- la mesa 2 nada más puede tener 10 personas como
máximo y es atendido por el empleado 3

agregar_mesas(10,3); -- la mesa 3 nada más puede tener 10 personas como
máximo y es atendido por el empleado 3

agregar_mesas(10,8); -- la mesa 4 nada más puede tener 10 personas como
máximo y es atendido por el empleado 8

agregar_mesas(10,8); -- la mesa 5 nada más puede tener 10 personas como
máximo y es atendido por el empleado 8

agregar_mesas(10,8); -- la mesa 6 nada más puede tener 10 personas como
máximo y es atendido por el empleado 8

end;

/


-- ******************************************************

-- reservar

-- agregar_reserva(fecha y hora, número de cliente, número de mesa)

begin

agregar_reserva('17-NOV-2022 10:00:00 AM',1,1);

agregar_reserva('17-NOV-2022 2:00:00 pm',2,2);

agregar_reserva('17-NOV-2022 4:00:00 pm',3,3);

end;

/


-- ******************************************************

-- pedidos

-- Para llevar: se refiere a ir a la cafetería y pedir empaquen la comida
para llevar.

-- para recoger: Administrativos ordena la comida previamente por corre y
pasa a recoger la comida cuando ya está lista.
```

```
-- nombre del tipo de pedido(Para llevar,para recoger), fecha de
pedido,número de cliente,id de empleado

begin

agregar_pedido('para recoger','17-NOV-2022',1,7);

agregar_pedido('para llevar','17-NOV-2022',2,4);

end;

/


-- ********************************************************

--artículos

-- agregar_articulos(cantidad del artículo o producto, número de pedido,
id de alimento)

begin

agregar_articulos(2,1,3);

agregar_articulos(3,2,3);

end;

/

/************************************************************/

/* 5.- Funciones                                            */

/************************************************************/


-- Función para calcular el total a pagar del cliente por su compra en la
cafetería.

create or replace function fn_total_a_pagar (

p_ali_precio alimento.ali_precio%type,

p_cantidad articulo.cantidad%type

)

return number

as

v_total number;

begin

v_total := p_ali_precio * p_cantidad;

return v_total;
```

```
end fn_total_a_pagar;
/
-- llenar estas 2 tablas para realizar el total a pagar.
-- select * from pedidos;
-- select * from articulos;

alter table pedido add (fn_total_a_pagar number(5,2),
dinero_entregado_cliente number(5,2),
cambio number(5,2)
);

-- Función para el cambio a devolver al cliente.
create or replace function fn_cambio (
p_random alimento.ali_precio%type,
p_total alimento.ali_precio%type
)
return number
as
v_cam alimento.ali_precio%type;
begin
v_cam := p_random - p_total;
return v_cam;
end fn_cambio;
/

/*************************************************************/
/* 6.- Triggers                                              */
/*************************************************************/

alter table historial_de_pedido add total_a_pagar number;
alter table historial_de_pedido add dinero_cliente number;
alter table historial_de_pedido add cambio number;
```

```
create or replace view vista_pedidos_antes as select * from pedido;
select * from vista_pedidos_antes;

create or replace trigger tr_aud_historial_ped
after update on pedido
for each row
declare
cursor c_cursor3 is select m_id from mesa;
begin
for v_cursor in c_cursor3 loop

insert into
historial_de_pedido(m_id,ped_id,total_a_pagar,dinero_cliente,cambio)
values(v_cursor.m_id,:new.ped_id,:new.FN_TOTAL_A_PAGAR,:new.DINERO_ENTREG
ADO_CLIENTE, :new.CAMBIO);
if (v_cursor.m_id <= :old.ped_id) then
exit;
end if;
end loop;

exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error en el tamaño de datos ingresados');
when others then
dbms_output.put_line('Error, ha ocurrido algo inesperado');
end;
/

-- Trigger para Actualizar, insertar, eliminar empleados
create or replace trigger tr_historial_empleados
```

```
after insert or delete or update on empleado

for each row

declare

begin

dbms_output.put_line('TRIGGER HISTORIAL_EMPLEADOS EJECUTADO CON ÉXITO');

if inserting then

insert into aud_empleado values (:new.e_id,'DATOS
INSERTADOS',:new.E_NOMBRE,:new.E_TELEFONO,:new.E_SALARIO);

elsif updating then


insert into aud_empleado values (:old.e_id,'DATOS ACTUALIZADOS -
ANTIGUOS',:old.E_NOMBRE,:old.E_TELEFONO,:old.E_SALARIO);

insert into aud_empleado values (:old.e_id,'DATOS ACTUALIZADOS -
NUEVOS',:new.E_NOMBRE,:new.E_TELEFONO,:new.E_SALARIO);


elsif deleting then

insert into aud_empleado values (:old.e_id,'DATOS
ELIMINADOS',:old.E_NOMBRE,:old.E_TELEFONO,:old.E_SALARIO);

end if;

end tr_historial_empleados;

/


/*********************************************************/

/* 7.- Bloque anónimo                                    */

/*********************************************************/


-- Bloque anónimo para calcular el total a pagar de pedido

declare

-- Declaración de variables con sus tipos de datos

v_ali_precio alimento.ali_precio%type;

v_ali_id alimento.ali_id%type;

v_cantidad articulo.cantidad%type;

v_total alimento.ali_precio%type; -- variable para calcular el total a
pagar
```

```
v_random alimento.ali_precio%type; -- variable que trata sobre el dinero
que ofrece el cliente para pagar su comida

v_cambio alimento.ali_precio%type; -- variable de cambio que se le
devuelve al cliente

-- Declaración del cursor para acceder a las columnas de las 3 tablas
(alimentos,articulos,pedidos)

cursor c_cursor is select pe.ped_id,ax.ali_id,ax.ali_nombre,

ax.ali_precio,aa.cantidad from alimento ax

inner join articulo aa on ax.ali_id = aa.ali_id

inner join pedido pe on pe.ped_id = aa.ped_id;

-- Sección ejecutable

begin


for v_cursor in c_cursor loop -- accedo a las columnas con la variable
v_cursor

dbms_output.put_line(chr(10)||'Número de pedido: '||v_cursor.ped_id||

chr(10) ||'Alimento ID: '||v_cursor.ali_id||chr(10)||'Precio del
alimento: '

||v_cursor.ali_precio||' dolares'||chr(10)||'Cantidad y nombre del
alimento: '

||v_cursor.cantidad||' -> '||v_cursor.ali_nombre);


-- Variable para calcular el total a pagar

v_total := fn_total_a_pagar(v_cursor.ali_precio,v_cursor.cantidad); --
Llamada a la función que calcula el total a pagar

v_random := dbms_random.value(0,20); -- función de número aleatorios

while (v_random < v_total) loop -- un condicional mientras para obligar a
v_random tomar un valor mayor al total a pagar.

v_random := dbms_random.value(0,20);

exit when v_random > v_total;

end loop;

v_cambio := fn_cambio(v_random,v_total); -- Llamada a la función que
calcula el cambio

-- a entregar al cliente una vez pagó

dbms_output.put_line(chr(10)||'El total a pagar es de: '||v_total||'
dolares');
```

```
dbms_output.put_line('************* Pagado ***************');
dbms_output.put_line('Dinero entregado por el cliente: '||v_random);
dbms_output.put_line('Cambio devuelto: '||v_cambio);

-- actualizar los valores de pedidos con el uso de cursor
update pedido
set fn_total_a_pagar = v_total,
dinero_entregado_cliente = v_random,
cambio = v_cambio
where ped_id = v_cursor.ped_id;

end loop;
-- Sección de errores
exception
when dup_val_on_index then
dbms_output.put_line('Datos repetidos');
when value_error then
dbms_output.put_line('Error causado por el tamaño de los datos
ingresados');
when others then
dbms_output.put_line('Error en la inserción de datos');
end;
/

/*************************************************************/
/* 8.- Vistas                                                */
/*************************************************************/

-- Vista para ver los pedidos antes de insertar pedidos
create or replace view vista_pedidos as select * from pedido;
select * from vista_pedidos;
```

```
-- Vista para ver el trigger de la tabla historial_pedidos

create or replace view trigger_historial_pedido as select * from
historial_de_pedido;

select * from trigger_historial_pedido;


-- Vista de cursor implementado en el bloque anónimo

create or replace view vista_cursor_bloque as select
pe.ped_id,ax.ali_id,ax.ali_nombre,

ax.ali_precio,aa.cantidad from alimento ax

inner join articulo aa on ax.ali_id = aa.ali_id

inner join pedido pe on pe.ped_id = aa.ped_id;


select * from vista_cursor_bloque;


-- vista antes de cambios en la tabla empleados

create or replace view vista_empleados as select * from empleado;

select * from vista_empleados;


delete from empleado where e_id = 1;


update empleado

set E_SALARIO = 1000

where e_id = 2;


-- Vista trigger de cambios hechos en la tabla empleados

create or replace view vista_aud_empleado as select * from aud_empleado;

select * from vista_aud_empleado;
```

ANEXO

Vista del trigger historial_de_pedido

```
View created.
```

M_ID	PED_ID	TOTAL_A_PAGAR	DINERO_CLIENTE	CAMBIO
1	1	5.7	15.69	9.99
1	2	8.55	14.06	5.51

Vista del cursor en el bloque anónimo

```
View created.
```

PED_ID	ALI_ID	ALI_NOMBRE	ALI_PRECIO	CANTIDAD
1	3	Pizza	2.85	2
2	3	Pizza	2.85	3

Vista empleados

View created.

E_ID	E_NOMBRE	E_TELEFONO	E_DIRECCION	E_SALARIO	T_ID
1	Gabriel	65458798	Las colinas	400	1
2	Johel	62457532	Paraiso	650	2
3	Ana	62457532	Paraiso	1050	3
4	Natalia	62457532	Villa lucre	950	4
5	Dany	61254385	Villa lucre	650	2
6	David	62574893	Cerro viento	400	1
7	Jose	62554894	Cincuentenario	950	4
8	Juan	62554841	Chorrillo	1050	3

Download CSV

8 rows selected.

Vista auditoria de empleado

View created.

EMP_ID	TRIG	NOMBRE	TELEFONO	SALARIO
1	DATOS ELIMINADOS	Gabriel	65458798	400
2	DATOS ACTUALIZADOS - ANTIGUOS	Johel	62457532	650
2	DATOS ACTUALIZADOS - NUEVOS	Johel	62457532	1000

CONCLUSIÓN

Las bases de datos son herramientas básicas en las empresas u organizaciones y son de vital importancia para estas entidades, el almacenamiento y gestión de los datos que manejan. Muchas veces con una finalidad más allá del almacenamiento como tal, sino que son imprescindibles para el funcionamiento, desarrollo y crecimiento de la empresa. Esto a través de la transformación de los datos en información importante para la organización y aplicar técnicas y estrategias que lleven al crecimiento o mejoramiento de esta. En el desarrollo de este proyecto se puede evidenciar la cantidad de datos que puede manejar un lugar en específico como lo es una cafetería universitaria. A medida que crece la cantidad de datos se pueden desarrollar técnicas para brindar un mejor servicio a los clientes y optimizar el funcionamiento de la cafetería. Como ya se ha especificado anteriormente la implementación de una base de datos no es inmediata, ya que primeramente se deben tener claros varios aspectos para que la implementación sea correcta y acertada. Iniciando por la misión de la base de datos, el objetivo, los alcances y los requerimientos que debe tener la base de datos y a partir de ellas llevar a cabo el desarrollo de la esta, como tal previamente transformando ese requerimiento en un modelo conceptual con el cual se le puede hacer una primera presentación del proyecto y luego entrando más en la parte especialista de base de datos convirtiéndolo a un modelo relacional que dicho sea de paso debe ser normalizado para una implementación (modelo físico) correcta y que permita gestionar los datos de manera eficaz y eficiente a través de instrucciones directas, pero principalmente a través de programación almacenada que permite la automatización de la gestión de los datos mediante procedimientos y funciones. Además, otros elementos como triggers que pueden ser acoplados a las tablas para dar un seguimiento a los datos que se almacenan. Finalmente, mediante el desarrollo de este proyecto se pudo demostrar como la aplicación de todos estos procesos ayudan y facilitan el tratamiento de los datos tanto para la organización, en este caso la cafetería universitaria, como para los clientes, debido a que se puede hacer una mejora constante utilizando los datos que se almacenan.

REFERENCIAS

- *Database PL/SQL Language Reference - Contents*. (s. f.). https://docs.oracle.com/database/121/LNPLS/toc.htm

- *Normalización de bases de datos*. (s. f.). CTI Soluciones. https://www.ctisoluciones.com/blog/normalizacion-base-de-datos

- *Jorge Sánchez. Manual de Gestión de Bases de Datos. Diseño lógico de bases de datos relacionales*. (s. f.). https://jorgesanchez.net/manuales/gbd/diseno-logico-relacional.html

- *Ejercicios PL/SQL #4 - Funciones y procedimientos*. (2018, agosto 23). https://youtu.be/lRp2ReMONx0