Curso 2020/2021 Pablo L. Navarro

PROYECTO “FREE PELOS”

Contenido:

1. PRESENTACIÓN

El objetivo del proyecto es organizar el trabajo de una peluquera autónoma que ofrece gran cantidad de servicios en diferentes lugares.

Lo más importante que se necesita registrar es la información relacionada con el cliente al que asignaremos un código, nombre, sexo, preferencias de servicio y teléfono; luego guardaremos la información del servicio que va a recibir el cual identificaremos con un nombre, descripción y precio preestablecido.

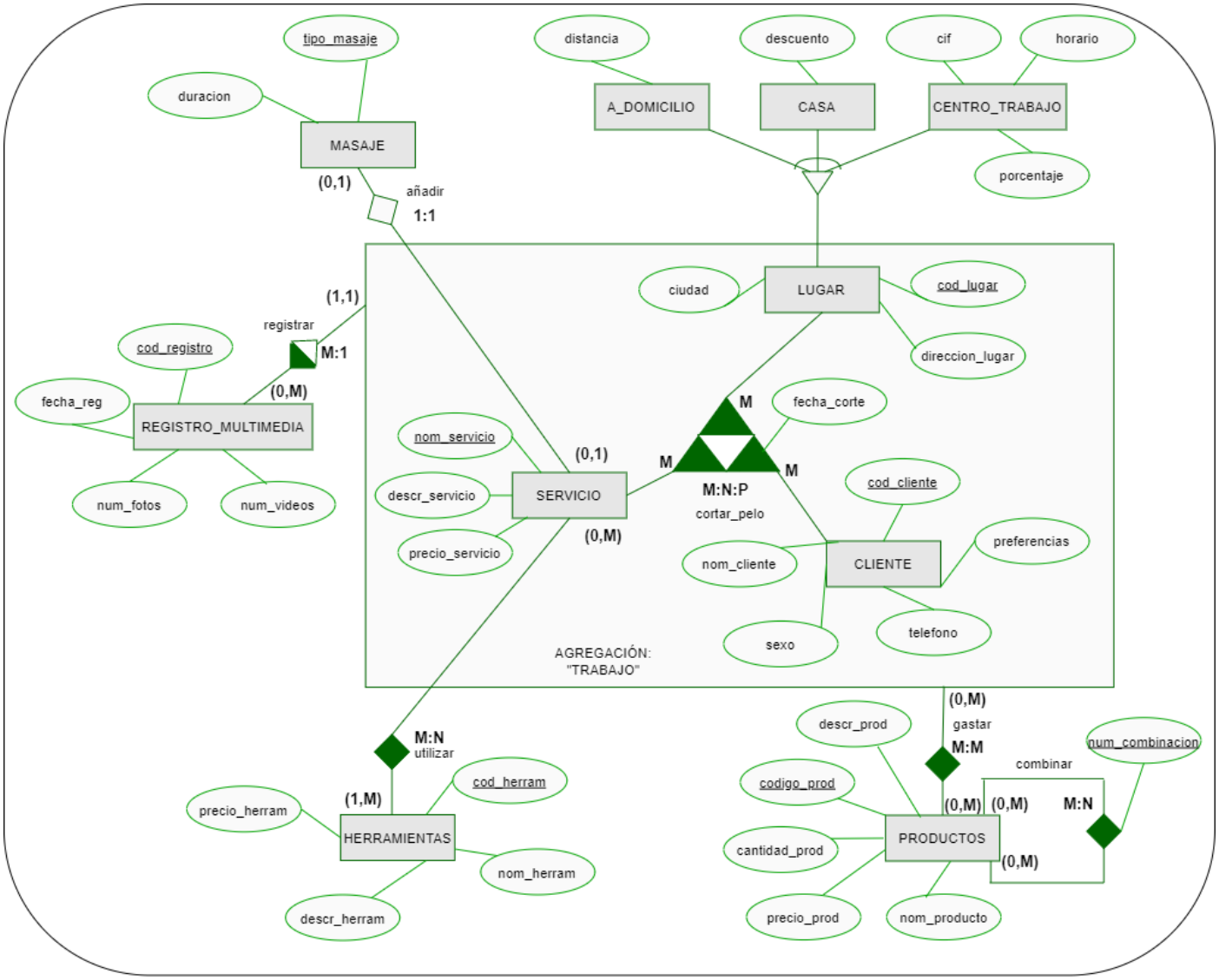
Lo siguiente a tener en cuenta es dónde se va dar el servicio con el cliente. Lo registraremos con un código, dirección, ciudad y si es a domicilio, en el hogar de la autónoma o en un centro de trabajo externo. En el caso de que el lugar del corte de pelo sea a domicilio registraremos la distancia recorrida desde el hogar, si el corte se produce en el hogar registraremos los descuentos que pueden conllevar y si es un centro de trabajo externo necesitaremos el cif de la empresa, el horario de apertura y el porcentaje con el que están negociados los servicios.

Un servicio concreto puede conllevar un masaje del que registraremos su tipo y duración, por tanto, un masaje concreto sólo puede estar vinculado a un servicio en especial probablemente más caro, dentro del servicio de corte va incluido el precio del masaje lo reciba o no el cliente puesto que el trabajador carece de titulación y necesita horas para practicar.

Casi siempre un trabajo o corte implica el gasto de uno o varios productos ya que a su vez pueden combinarse muchas veces entre si para conseguir las medidas deseadas. De cada producto nos interesa registrar su código, cantidad, descripción, precio y nombre genérico.

Por último, esporádicamente se hacen registros fotográficos de los trabajos para subir contenido a las redes sociales y publicitarse, cada registro llevará un código único ya que en un mismo corte se pueden hacer varios registros diferentes, guardaremos el número de fotos y vídeos que incluye cada uno.

2. MODELO ENTIDAD – RELACIÓN



3. RELACIONAL

CLIENTE: (cod_cliente, preferencias, telefono, sexo, nom_cliente)
PK: (cod_cliente)

LUGAR: (cod_lugar, direccion_lugar, ciudad)
PK: (cod_lugar)

SERVICIO: (nom_servicio, descr_servicio, precio_servicio)
PK: (nom_servicio)

MASAJE: (tipo_masaje, duracion)
PK: (tipo_masaje)

AÑADIR_MASAJE: (tipo_masaje, nom_servicio)
PK: (nom_servicio,tipo_masaje)
FK: (nom_servicio) → SERVICIO
FK: (tipo_masaje) → MASAJE

A_DOMICILIO: (cod_lugar, distancia)
PK: (cod_lugar)
FK: (cod_lugar) → LUGAR

CASA: (cod_lugar, descuento)
PK: (cod_lugar)
FK: (cod_lugar) → LUGAR

CENTRO_TRABAJO: (cod_lugar, cif, horario, porcentaje)
PK: (cod_lugar)
FK: (cod_lugar) → LUGAR

HERRAMIENTAS: (cod_herram, nom_herram, descr_herram, precio_herram)
PK: (cod_herram)

UTILIZACION: (nom_servicio, cod_herramienta)
PK: (nom_servicio, cod_herramienta)
FK: (nom_servicio) → SERVICIO
FK: (cod_herramienta) → HERRAMIENTAS

PRODUCTOS: (cod_prod, descr_prod, cantidad_prod, precio_prod, nom_prod)
PK: (cod_prod)

COMBINAR: (num_combinacion, cod_prod1, cod_prod2)
PK: (num_combinacion, cod_prod1, cod_prod2)
FK: (cod_producto1, cod_producto2) --> PRODUCTOS

CORTE_PELO (fecha_corte, cod_cliente, cod_lugar, nom_servicio, cod_producto, cod_reg)
PK: (fecha_corte, cod_cliente, cod_lugar, nom_servicio)
FK: (cod_cliente) → CLIENTE
FK: (cod_lugar) → LUGAR
FK: (nom_servicio) → SERVICIO
FK: (cod_producto) → PRODUCTOS
FK: (cod_reg) → REGISTRO_MULTIMEDIA

GASTAR: (fecha_corte, cod_cliente, cod_lugar, nom_servicio, cod_producto)
PK: (fecha_corte, cod_cliente, cod_lugar, nom_servicio, cod_producto)
FK: (fecha_corte, cod_cliente, cod_lugar, nom_servicio, cod_producto) → CORTE_PELO

REGISTRO_MULTIMEDIA: (cod_registro, num_fotos, num_videos, fecha_registro, cod_cliente, cod_lugar, nom_servicio)
PK: (cod_registro, fecha_registro)
FK: (fecha_registro, cod_cliente, cod_lugar, nom_servicio) → CORTE_PELO
VNN: (fecha_registro, cod_cliente, cod_lugar, nom_servicio)

4. DDL

```
CREATE TABLE cliente(
cod_cliente SERIAL NOT NULL,
nombre_cliente varchar (50),
telefono numeric (10),
preferencias varchar (100) DEFAULT 'Sin preferencias',
sexo varchar (10),
CONSTRAINT PK_CLIENTE PRIMARY KEY (cod_cliente)
);

CREATE TABLE lugar(
cod_lugar SERIAL NOT NULL,
direccion_lugar varchar (100),
ciudad varchar (50),
CONSTRAINT PK_LUGAR PRIMARY KEY (cod_lugar)
);

CREATE TABLE servicio(
nom_servicio varchar (50) NOT NULL,
descr_servicio varchar (100) DEFAULT 'Sin descripción',
precio_servicio numeric (5) CHECK (precio_servicio >= 0),
CONSTRAINT PK_SERVICIO PRIMARY KEY (nom_servicio)
);

CREATE TABLE masaje (
tipo_masaje varchar (50) NOT NULL,
duracion int CHECK (duracion >= 0), -- MINUTOS
CONSTRAINT PK_MASAJE PRIMARY KEY (tipo_masaje)
);
```

```
CREATE TABLE anyadir_masaje (
tipo_masaje varchar (50) NOT NULL,
nom_servicio varchar (50) NOT NULL,
CONSTRAINT PK_ANYADIR_MASAJE PRIMARY KEY (tipo_masaje, nom_servicio),
CONSTRAINT FK_ANYADIR_TIPO FOREIGN KEY (tipo_masaje)
        REFERENCES masaje (tipo_masaje),
CONSTRAINT FK_ANYADIR_SERVICIO FOREIGN KEY (nom_servicio)
        REFERENCES servicio (nom_servicio)
);

CREATE TABLE a_domicilio(
cod_lugar int NOT NULL,
distancia numeric(5) CHECK (distancia >= 0),
CONSTRAINT PK_DOMICILIO PRIMARY KEY (cod_lugar),
CONSTRAINT FK_DOMICILIO FOREIGN KEY (cod_lugar)
        REFERENCES lugar (cod_lugar)
);

CREATE TABLE casa(
cod_lugar int NOT NULL,
descuento numeric (5) CHECK (descuento >= 0),
CONSTRAINT PK_CASA PRIMARY KEY (cod_lugar),
CONSTRAINT FK_CASA FOREIGN KEY (cod_lugar)
        REFERENCES lugar(cod_lugar)
);
```

```
CREATE TABLE centro_trabajo(
cod_lugar int NOT NULL,
cif varchar(20) NOT NULL,
horario varchar (50) DEFAULT '09:00 - 21:00',
porcentaje numeric (5) CHECK (porcentaje >= 0),
CONSTRAINT PK_CENTRO PRIMARY KEY (cod_lugar),
CONSTRAINT FK_CENTRO FOREIGN KEY (cod_lugar)
        REFERENCES lugar(cod_lugar)
);

CREATE TABLE herramientas(
cod_herram SERIAL NOT NULL,
nom_herram varchar (50),
descr_herram varchar (100) DEFAULT 'Sin descripción',
precio_herram numeric (10) CHECK (precio_herram >= 0),
CONSTRAINT PK_HERRAMIENTAS PRIMARY KEY (cod_herram)
);

CREATE TABLE utilizar(
nom_servicio varchar(50) NOT NULL,
cod_herram int NOT NULL DEFAULT 000,
CONSTRAINT PK_UTILIZAR PRIMARY KEY (nom_servicio, cod_herram),
CONSTRAINT FK_UTILIZAR_SERVICIO FOREIGN KEY (nom_servicio)
        REFERENCES servicio (nom_servicio),
CONSTRAINT FK_UTILIZAR_HERRAMIENTA FOREIGN KEY (cod_herram)
        REFERENCES herramientas (cod_herram)
        ON UPDATE CASCADE
        ON DELETE SET DEFAULT
);
```

```
CREATE TABLE productos (
cod_prod SERIAL NOT NULL,
nom_prod varchar (150),
descr_prod varchar (100) DEFAULT 'Sin descripción',
cantidad_prod numeric (5) CHECK (cantidad_prod >= 0),
precio_prod numeric (5) CHECK (precio_prod >= 0),
CONSTRAINT PK_PRODUCTOS PRIMARY KEY (cod_prod)
);

CREATE TABLE combinar (
num_combinacion SERIAL NOT NULL, --Tengo que crear un pk adicional
cod_prod1 int NOT NULL DEFAULT 0, --para poder repetir las combinaciones
cod_prod2 int NOT NULL DEFAULT 0,
CONSTRAINT PK_COMBINAR PRIMARY KEY (num_combinacion, cod_prod1, cod_prod2),
CONSTRAINT FK_COMBINAR1 FOREIGN KEY (cod_prod1)
        REFERENCES productos (cod_prod)
                ON UPDATE CASCADE
                ON DELETE SET DEFAULT,
CONSTRAINT FK_COMBINAR2 FOREIGN KEY (cod_prod2)
        REFERENCES productos (cod_prod)
                ON UPDATE CASCADE
                ON DELETE SET DEFAULT
);
```

```
CREATE TABLE corte_pelo(
fecha_corte date CHECK(to_char(fecha_corte, 'YYYY') > '2017'),
cod_cliente int NOT NULL,
cod_lugar int NOT NULL,
nom_servicio varchar (50) NOT NULL,
cod_producto int DEFAULT '000',
CONSTRAINT PK_CORTE_PELO PRIMARY KEY (fecha_corte, cod_cliente, cod_lugar, nom_servicio),
CONSTRAINT FK_CORTE_CLIENTE FOREIGN KEY (cod_cliente)
        REFERENCES cliente (cod_cliente),
CONSTRAINT FK_CORTE_LUGAR FOREIGN KEY (cod_lugar)
        REFERENCES lugar (cod_lugar),
CONSTRAINT FK_CORTE_SERVICIO FOREIGN KEY (nom_servicio)
        REFERENCES servicio (nom_servicio),
CONSTRAINT FK_CORTE_PRODUCTO FOREIGN KEY (cod_producto)
        REFERENCES productos (cod_prod)
        ON UPDATE CASCADE
        ON DELETE SET DEFAULT
);
```

```
CREATE TABLE registro_multimedia(

cod_registro SERIAL NOT NULL,

num_fotos numeric(5) CHECK(num_fotos >= 0),

num_videos numeric(5) CHECK(num_videos >= 0),

fecha_registro date CHECK(to_char(fecha_registro, 'YYYY') > '2017'),

cod_cliente int,

cod_lugar int,

nom_servicio varchar(50),

CONSTRAINT PK_REGISTRO_MULTIMEDIA PRIMARY KEY (cod_registro),

CONSTRAINT FK_REGISTRO_MULTIMEDIA FOREIGN KEY (fecha_registro, cod_cliente, cod_lugar,
nom_servicio)

        REFERENCES corte_pelo (fecha_corte, cod_cliente, cod_lugar, nom_servicio)

        ON UPDATE CASCADE

        ON DELETE SET NULL

);



CREATE TABLE gastar (

fecha_corte date CHECK(to_char(fecha_corte, 'YYYY') > '2017'),

cod_cliente int,

cod_lugar int,

cod_servicio varchar (50),

cod_producto int DEFAULT '000',

CONSTRAINT PK_GASTAR PRIMARY KEY (fecha_corte, cod_cliente, cod_lugar, cod_servicio, cod_producto),

CONSTRAINT FK_GASTAR_CORTE FOREIGN KEY (fecha_corte, cod_cliente, cod_lugar, cod_servicio)

        REFERENCES corte_pelo (fecha_corte, cod_cliente, cod_lugar, nom_servicio)

        ON DELETE SET NULL

        ON UPDATE CASCADE,

CONSTRAINT FK_GASTAR_PRODUCTO FOREIGN KEY (cod_producto)

        REFERENCES productos (cod_prod)

        ON DELETE SET DEFAULT   ON UPDATE CASCADE  );
```

5. DML

--------TABLA CLIENTE--------

```
INSERT INTO cliente (nombre_cliente, telefono, preferencias, sexo)
VALUES
    ('Misis One', 1234567, 'Corte y Tinte', 'M'),
    ('Don Two', 4567473, 'Rapado al 0', 'H'),
    ('Lady Three', 45645645, 'Corte a la antigua', 'M'),
    ('Mister Four', 45688942, 'Rapado Demoníaco', 'H'),
    ('Sir Five', 23476794, 'Corte Militar', 'H');

INSERT INTO cliente (nombre_cliente, telefono)
VALUES
    ('Queen Six', 389432),
    ('Maese Seven', 795950),
    ('Duquesa Eight', 079067456),
    ('Nine-Chan', 067453623),
    ('Hokage Ten', 0674567);
```

--------TABLA SERVICIO--------

```
INSERT INTO servicio
VALUES
    ('Corte Simple', 'Lavar y cortar', 20),
    ('Corte Doble', 'Lavar y cortar 2 pax', 30),
    ('Corte Especial', 'Secado especial', 40),
    ('Corte y Tinte', 'Pr de gama alta', 50),
    ('Corte Eco', 'poco pelo',10);
```

```
INSERT INTO servicio (nom_servicio, precio_servicio)
VALUES
    ('Corte Samurai', 100),
    ('Permanente', 200),
    ('Corte Laser', 300),
    ('Japonés', 150),
    ('Ochentero', 60);
```

--------TABLA LUGAR--------

```
INSERT INTO lugar
VALUES (000, 'Direccion Casa', 'Benidorm');

INSERT INTO lugar (direccion_lugar, ciudad)
VALUES
    ('Calle Stephen King 19', 'Misery City'),
    ('Calle Brandom Sanderson 14', 'Ciudad de las Brumas'),
    ('Calle Patrick Rothfuss 33', 'Ciudad del Silencio'),
    ('Calle Shinigami', 'Tokio'),
    ('Complejo Umbrella', 'Raccon City'),
    ('Mansion Hellsing', 'New York'),
    ('Marine Ford', 'Grand Line'),
    ('Calle Ichigo', 'Karakura Town'),
    ('Calle Songbird 3', 'Columbia');
```

--------TABLA MASAJE--------

```
INSERT INTO masaje
VALUES
    ('Completo', 60),
    ('Medio', 30),
    ('Piernas', 20),
    ('Brazos', 20),
    ('Cabeza', 15),
    ('Vietnamita', 90),
    ('Turco', 45),
    ('Sueco', 35),
    ('Shiatsu', 60),
    ('Lumbar', 15);
```

--------TABLA AGREGAR_MASAJE--------

```
INSERT INTO anyadir_masaje
VALUES
    ('Completo', 'Corte Especial'),
    ('Cabeza', 'Corte Simple'),
    ('Piernas', 'Corte Doble'),
    ('Brazos', 'Corte y Tinte'),
    ('Medio', 'Corte Eco'),
    ('Shiatsu', 'Japonés'),
    ('Cabeza', 'Corte Especial'),
    ('Sueco','Ochentero'),
    ('Lumbar','Permanente'),
    ('Vietnamita', 'Corte Laser');
```

--------TABLA CASA-------

```
INSERT INTO casa
VALUES  (0, 15);
        --Si el lugar es casa habra 15%
```

--------TABLA A_DOMICILIO-------

```
INSERT INTO a_domicilio
VALUES
        (1, 50),
        (2, 100),
        (3, 300),
        (4, 900),
        (8, 500),
        (9, 800);
```

--------TABLA CENTRO_TRABAJO-------

```
INSERT INTO centro_trabajo
VALUES
        (5, 'TVIRUSZ00MB1', '00:00 - 08:00', 60),
        (7, 'ECH1R00DA', '02:00 - 22:00', 40);

INSERT INTO centro_trabajo (cod_lugar, cif, porcentaje)
VALUES (6, '41UC4RD', 50);
```

--------TABLA HERRAMIENTAS--------

```
INSERT INTO herramientas
VALUES (000, 'Borrado u Obsoleto', 'Herramienta no disponible, borrada u obsoleta', '0');
```

```
INSERT INTO herramientas (nom_herram, precio_herram)
VALUES
('Tijeras Schvarosky', 400),
('Harusame', 900),
('Longclaw', 300),
('Masamune', 1000),
('X-Gun', 200),
('Secador Solar', 100),
('Andúril', 3000),
('Oathkeeper', 200),
('Rizador de Saúco', 900);
```

--------TABLA UTILIZAR--------

```
INSERT INTO utilizar
VALUES
('Corte Eco', 1),
('Corte Eco', 9),
('Corte Especial', 3),
('Corte Doble', 7),
('Permanente', 7),
('Japonés', 4),
('Japonés', 5),
('Ochentero', 2),
('Corte Samurai', 4),
('Corte y Tinte', 8);
```

--------TABLA PRODUCTOS--------

```
INSERT INTO productos(cod_prod, nom_prod, descr_prod)
VALUES (000, 'Producto Obsoleto', 'El producto ya no existe o no se utiliza');


INSERT INTO productos (nom_prod, cantidad_prod, precio_prod)
VALUES
    ('Tinte Revlon', 5, 20),
    ('Decolorador Natural', 2, 30),
    ('Tinte Schwarzkopzt', 10, 40),
    ('Tinte Loreal', 20, 15),
    ('Reparador Loreal', 50, 10),
    ('Mascarilla Natural', 10, 30),
    ('Eliminador de Residuios Schwarzkopzt', 5, 25),
    ('Champu Anticaida Revlon', 8, 8),
    ('Antigrasa Loreal', 5, 14);

    --En la tabla combinar y gastar no hago inserts aun, hare un trigger
    --para que cada insert disminuya la cantidad de los productos;
```

--------TABLA CORTE_PELO--------

```
INSERT INTO corte_pelo
VALUES
('05-04-2021', 1, 1, 'Corte Eco', 1),
('19-07-2019', 2, 2, 'Corte Simple', 2),
('11-09-2020', 3, 3, 'Corte Simple', 3),
('18-11-2019', 4, 4, 'Corte Especial', 4),
('01-02-2019', 5, 5, 'Permanente', 5),
('02-02-2021', 6, 6, 'Corte Doble', 6),
('08-12-2019', 7, 7, 'Ochentero', 7),
('10-07-2018', 8, 8, 'Corte Samurai', 8),
('26-01-2019', 9, 9, 'Japonés', 9),
('15-11-2020', 1, 2, 'Japonés', 3),
('22-09-2018', 4, 5, 'Corte y Tinte', 6),
('09-07-2020', 7, 8, 'Corte Especial', 9);
```

--------TABLA REGISTRO_MULTIMEDIA--------

INSERT INTO registro_multimedia (num_fotos, num_videos, fecha_registro, cod_cliente, cod_lugar, nom_servicio)

VALUES

(20,5,'19-07-2019', 2, 2, 'Corte Simple'),

(2,0,'11-09-2020', 3, 3, 'Corte Simple'),

(50,18,'18-11-2019', 4, 4, 'Corte Especial'),

(10,1,'01-02-2019', 5, 5, 'Permanente'),

(14,12,'02-02-2021', 6, 6, 'Corte Doble'),

(30,6,'10-07-2018', 8, 8, 'Corte Samurai'),

(20,1,'26-01-2019', 9, 9, 'Japonés'),

(0,10,'15-11-2020', 1, 2, 'Japonés'),

(0,2,'22-09-2018', 4, 5, 'Corte y Tinte'),

(15,3,'09-07-2020', 7, 8, 'Corte Especial');

6. SQL

```
------ (A) 5 CONSULTAS SIMPLES DE UNA SOLA TABLA ------


-- Muestra todo el contenido de la tabla mas importante sin ninguna modificacion,
    -- quiero mostrarla asi de primeras para luego ir ensenyando lo que voy haciendo.
    SELECT * FROM corte_pelo;


-- Muestra el codigo 0, esta fila no se podra eliminar (por triggers), de esta manera
    -- podremos eliminar cualquier producto y los FK de otras tablas pasaran a referenciar al 0
    -- evitando asi posibles errores o perdida de informacion.
    SELECT cod_prod, nom_prod, descr_prod
        FROM productos
        WHERE cod_prod = 0;


-- Como en el caso anterior, ocurre lo mismo con las herramientas
    SELECT cod_herram, nom_herram, descr_herram
        FROM herramientas
        WHERE nom_herram LIKE 'B%';


-- Acota los registros con un determinado numero de fotos o videos
    SELECT num_fotos, num_videos, fecha_registro, nom_servicio
        FROM registro_multimedia
        WHERE num_fotos > 20
            OR num_videos > 10;


-- Muestra los diferentes tipos de sexo que se tienen en cuenta
    SELECT DISTINCT sexo FROM cliente;
```

```
-------------------------------------------------------------------------------------------------------------------------------------------------------

        ------ (B) 2 UPDATES Y 2 DELETES EN CUALQUIER TABLA  ------


    UPDATE cliente SET sexo = 'X' WHERE sexo IS NULL; -- La mitad de los inserts estan sin sexo


    UPDATE herramientas SET cod_herram = 333 WHERE cod_herram = 7;

        -- Desencadena UPDATE CASCADE en UTILIZAR


    DELETE FROM productos WHERE cod_prod = 1;

        -- Desencadena DELETE SET DEFAULT en CORTE_PELO y GASTAR


    DELETE FROM corte_pelo WHERE fecha_corte BETWEEN '2019-06-19' AND '2019-08-19';

        -- Desencadena DELETE SET NULL en REGISTRO_MULTIMEDIA y GASTAR


-------------------------------------------------------------------------------------------------------------------------------------------------------
        ------ (C) 3 CONSULTAS CON MAS DE UNA TABLA ------


-- Muestra el nombre de las herramientas, el precio y el servicio donde se usan ordenadas por precio
    SELECT h.nom_herram AS herramienta, h.precio_herram AS precio_euros, u.nom_servicio AS servicio_donde_se_usa
        FROM herramientas h, utilizar u
        WHERE h.cod_herram = u.cod_herram
            AND h.precio_herram > 800
        ORDER BY h.precio_herram DESC;


-- Muestra la distancia desde casa a las direcciones y ciudades mas lejanas, ordenado por distancia
    SELECT d.distancia AS km, l.ciudad, l.direccion_lugar AS direccion
        FROM a_domicilio d, lugar l
        WHERE d.cod_lugar = l.cod_lugar
            AND d.distancia > 50
        ORDER BY km DESC;
```

```
-- Muestra los cortes a partir del 2020 ordenados por fecha, aparece nombre del cliente,
    -- su direccion y ciudad, el producto utilizado y el servicio que consumio.
    SELECT cp.fecha_corte AS fecha, cl.nombre_cliente AS cliente, lu.ciudad,
    lu.direccion_lugar AS direccion, pr.nom_prod AS producto_usado, cp.nom_servicio AS servicio
        FROM corte_pelo cp, cliente cl, lugar lu, productos pr
        WHERE cp.cod_cliente = cl.cod_cliente
            AND cp.cod_lugar = lu.cod_lugar
            AND cp.cod_producto = pr.cod_prod
            AND TO_CHAR(cp.fecha_corte, 'YYYY') >= '2020'
        ORDER BY fecha;

-------------------------------------------------------------------------------------------------------------------------------------------
        ------ (D) 3 CONSULTAS USANDO FUNCIONES ------

-- Media de todos los masajes
    SELECT AVG (duracion) AS media_minutos_masajes
        FROM masaje;

-- Total de clientes registrados
    SELECT COUNT(*) as clientes_registrados
        FROM cliente;

-- Muestra el coste del servicio y el producto de los anyos seleccionados
    SELECT SUM(se.precio_servicio + pr.precio_prod) AS ganancia_bruto
        FROM corte_pelo cp, servicio se, productos pr
        WHERE cp.cod_producto = pr.cod_prod
            AND cp.nom_servicio = se.nom_servicio
            AND TO_CHAR(cp.fecha_corte, 'YYYY') IN ('2018','2020');
```

------ (E) 2 CONSULTAS USANDO GROUP BY ------

-- Muestra el total obtenido por servicio y el numero de veces que se ha realizado

```
SELECT cp.nom_servicio, SUM(se.precio_servicio) AS total_por_servicio, COUNT(cp.nom_servicio) AS veces
    FROM corte_pelo cp, servicio se
    WHERE cp.nom_servicio = se.nom_servicio
    GROUP BY cp.nom_servicio
    ORDER BY total_por_servicio DESC;
```

-- Muestra el total obtenido por cada cliente por servicio contar el gasto de productos

```
SELECT c.nombre_cliente, s.precio_servicio, s.nom_servicio
    FROM cliente c, servicio s, corte_pelo p
    WHERE s.precio_servicio > 10
        AND c.cod_cliente = p.cod_cliente
        AND s.nom_servicio = p.nom_servicio
    GROUP BY c.nombre_cliente, s.precio_servicio, s.nom_servicio
    ORDER BY s.precio_servicio DESC;
```

------ (F) 2 CONSULTAS UTILIZANDO SUBCONSULTAS ------

-- Muestra qué tipo de masaje incluyó cada servicio realizado

```
SELECT *
    FROM anyadir_masaje
    WHERE nom_servicio IN (SELECT nom_servicio FROM corte_pelo);
```

```
-- Muestra el nombre del cliente que tuvo el servicio más barato (3 subconsultas anidadas)

        SELECT nombre_cliente AS cliente_mas_tacanyo

                FROM cliente

                WHERE cod_cliente =

                        (SELECT cod_cliente FROM corte_pelo  WHERE nom_servicio =

                                (SELECT nom_servicio FROM servicio WHERE precio_servicio =

                                        (SELECT MIN(precio_servicio) FROM servicio)));
```

```
                ------ (G) 2 CONSULTAS UTILIZANDO GROUP BY y HAVING ------


-- Muestra el total por servicio de los servicios que al menos se hayan realizado 2 veces

        SELECT cp.nom_servicio, SUM(se.precio_servicio) AS total_por_servicio, COUNT(cp.nom_servicio) AS veces

                FROM corte_pelo cp, servicio se

                WHERE cp.nom_servicio = se.nom_servicio

                GROUP BY cp.nom_servicio

                HAVING COUNT(cp.nom_servicio) >= 2;


-- Muestra el total de gasto por productos de los productos que al menos se hayan gastado 2 veces

        SELECT pr.nom_prod AS producto, SUM(pr.precio_prod) AS total_gasto, COUNT(cp.cod_producto) AS veces_usado

                FROM productos pr, corte_pelo cp

                WHERE pr.cod_prod = cp.cod_producto

                GROUP BY pr.nom_prod

                HAVING COUNT(cp.cod_producto) >= 2;
```

------ (H) 3 UPDATES USANDO SUBCONSULTAS EN WHERE Y SET ------

```
-- Copia las preferencias del cliente con pk 5 en el 9
	UPDATE cliente
		SET preferencias = (
			SELECT preferencias
			FROM cliente
			WHERE cod_cliente = 5
		)
		WHERE nombre_cliente = (
			SELECT nombre_cliente
			FROM cliente
			WHERE cod_cliente = 9
		);

-- Cambia el pk de herramienta 4 por 60; Desencadena UPDATE CASCADE en UTILIZAR
	UPDATE herramientas
		SET cod_herram = (
			SELECT (cod_prod * 10)
			FROM productos
			WHERE nom_prod = 'Mascarilla Natural'
		)
		WHERE cod_herram = (
			SELECT cod_cliente
			FROM corte_pelo
			WHERE fecha_corte = '2019-11-18');
```

```
-- Cambia el cod_prod de 5 a 80; Desencadena UPDATE CASCADE en CORTE_PELO

        UPDATE productos

                SET cod_prod = (

                        SELECT (cod_cliente * 20)

                        FROM corte_pelo

                        WHERE fecha_corte = '2019-11-18'

                )

                WHERE cod_prod = (

                        SELECT cod_lugar

                        FROM corte_pelo

                        WHERE nom_servicio = 'Permanente'

                                AND cod_cliente = 5

                );
```

7. PLPG SQL

---------- 2 VISTAS ----------

```
-- Esta primera vista es una mejora de la tabla corte_pelo donde sustitituimos los pk's que protegen los datos por
        -- otros de interes directo, referenciamos los vinculos con el resto de tablas y anyadimos el calculo del total
        -- bruto por servicio de manera que para obtener el neto de cada uno de los servicios completos solo faltaria
        -- aplicar descuentos o incrementos dependiendo de si el lugar es a domicilio, en un centro de trabajo o en
        -- la casa del trabajador, pero para ello necesitaremos utilizar funciones;

        CREATE VIEW precio_corte_bruto (fecha, codigo, nombre_cliente, direccion, ciudad, servicio, precio_servicio,
                                        producto, precio_producto, total_bruto ) AS
                SELECT cp.fecha_corte, cl.cod_cliente, cl.nombre_cliente, lu.direccion_lugar, lu.ciudad, se.nom_servicio,
                        se.precio_servicio, pr.nom_prod, pr.precio_prod, (pr.precio_prod + se.precio_servicio) AS total_bruto
                        FROM corte_pelo cp, cliente cl, lugar lu, productos pr, servicio se
                        WHERE cp.cod_cliente = cl.cod_cliente
                                AND cp.cod_lugar = lu.cod_lugar
                                AND cp.nom_servicio = se.nom_servicio
                                AND cp.cod_producto = pr.cod_prod;

        SELECT * FROM precio_corte_bruto;


--En esta segunda vista mostramos el masaje que puede incluir gratuitamente cada servicio con su duracion;

        CREATE VIEW masaje_precio (servicio, masaje, duracion, precio) AS
                SELECT se.nom_servicio, ma.tipo_masaje, ma.duracion, se.precio_servicio
                        FROM masaje ma, servicio se, anyadir_masaje am
                        WHERE se.nom_servicio = am.nom_servicio
                                AND ma.tipo_masaje = am.tipo_masaje;

        SELECT * FROM masaje_precio;
```

---------- 2 FUNCIONES ----------

-- A la primera funcion le pasaremos como parametro el id de un cliente y la fecha donde se realizo el corte y nos devolvera

-- el total del precio con los descuentos o incrementos aplicados dependiendo del lugar donde se realizo; Codigo comentado;

```
CREATE OR REPLACE FUNCTION precio_neto (cliente_id int, fecha_servicio date)
    RETURNS numeric AS $$
-- GUARDAMOS EL BRUTO Y COD_LUGAR DE LA VISTA 1 Y CORTE_PELO PARA EL CLIENTE Y FECHA ELEGIDOS;
        DECLARE bruto numeric := (SELECT total_bruto FROM precio_corte_bruto
                WHERE cliente_id = codigo AND fecha_servicio = fecha);
        DECLARE id_lugar int := (SELECT cod_lugar FROM corte_pelo
                WHERE cliente_id = cod_cliente AND fecha_servicio = fecha_corte);
    -- PREPARAMOS VARIABLES PARA GUARDAR LOS RESULTADOS;
        DECLARE tasas numeric;
        DECLARE neto numeric;
    BEGIN
    -- IF 1º COMPRUEBA SI EL SERVICIO FUE EN CASA Y APLICA DESCUENTO;
        IF id_lugar = 0 THEN
            tasas := (bruto * 0.15);
            neto := (bruto - tasas);
    RAISE INFO 'EL SERVICIO DE CLIENTE % Y FECHA % FUE EN CASA',cliente_id, fecha_servicio;
    -- IF 2º COMPRUEBA SI EL SERVICIO FUE A DOMICILIO Y APLICA INCREMENTO POR KM;
        ELSIF EXISTS ( SELECT cod_lugar FROM a_domicilio
                WHERE cod_lugar = id_lugar ) THEN
            tasas := ((SELECT distancia FROM a_domicilio WHERE cod_lugar = id_lugar) * 0.20);
            neto := (bruto + tasas);
    RAISE INFO 'EL SERVICIO DE CLIENTE % Y FECHA % FUE A DOMICILIO',cliente_id, fecha_servicio;
```

```
        -- IF 3º COMPRUEBA SI EL SERVICIO FUE EN CENTRO DE TRABAJO Y APLICA PORCENTAJES;
                ELSIF EXISTS ( SELECT cod_lugar FROM centro_trabajo
                                        WHERE cod_lugar = id_lugar ) THEN
        tasas := (bruto * ((100 - (SELECT porcentaje FROM centro_trabajo WHERE cod_lugar = id_lugar))/100));
        neto := TRUNC((bruto - tasas),2);
RAISE INFO 'EL SERVICIO DE CLIENTE % Y FECHA % FUE EN UN CENTRO DE TRABAJO',cliente_id, fecha_servicio;
    -- SI NO EXISTE EN NINGUN LUGAR LOS PARAMETROS NO SON VALIDOS, INFORMAMOS Y DEVOLVEMOS 0.0
                ELSE
    RAISE INFO 'NO EXISTE REGISTRO CON CODIGO DE CLIENTE % Y FECHA %',cliente_id, fecha_servicio;
                        RETURN 0.0;
                END IF;
-- SI EXISTE PERO EL CALCULO ES NULL SE ELIMINO EL PRODUCTO O SERVICIO DEL REGISTRO, INFORMAMOS Y DEVOLVEMOS 0.0
                IF neto IS NULL THEN
RAISE INFO 'EL SERVICIO O PRODUCTO FUERON ELIMINADOS PARA EL CLIENTE % Y FECHA %',cliente_id, fecha_servicio;
                        RETURN 0.0;
                END IF;
        RETURN neto;
        END;
        $$ LANGUAGE PLPGSQL;


    SELECT precio_neto (8, '2018-07-10');       -- DEVUELVE DATO (A_DOMICILIO);
    SELECT precio_neto (5, '2019-02-01');       -- DEVUELVE DATO (CENTRO_TRABAJO);
    SELECT precio_neto (1, '2018-04-05');   -- DEVUELVE 0.0 Y MENSAJE DE ELIMINADO;
    SELECT precio_neto (4, '2010-01-01');       -- DEVUELVE 0.0 INDICANDO QUE NO EXISTE REGISTRO;
```

```
-- Ahora aprovechamos la funcion para crear otra y recorrer con un cursor la vista entre los anyos deseados obteniendo
    -- todos los beneficios netos y la suma total de estos los acumulamos en otra variable que agregamos tras el bucle;

CREATE OR REPLACE FUNCTION get_registros (anyo_inicio varchar, anyo_fin varchar)
    RETURNS text AS $$
        DECLARE registros_salida text DEFAULT '';
        DECLARE total_beneficio numeric DEFAULT 0;
        DECLARE record_registros RECORD;
        DECLARE cursor_registros CURSOR
                FOR SELECT fecha, nombre_cliente, codigo, servicio, total_bruto
                FROM precio_corte_bruto
                WHERE TO_CHAR(fecha, 'YYYY') BETWEEN anyo_inicio AND anyo_fin;
    BEGIN
        OPEN cursor_registros;
    RAISE WARNING 'Como invocamos dos veces la funcion precio_neto, todas las INFO estan duplicadas';
        LOOP
            FETCH cursor_registros INTO record_registros;
            EXIT WHEN NOT FOUND;

            registros_salida := registros_salida || '  ----  ' || record_registros.fecha || ';  '
            || record_registros.nombre_cliente || ';  ' || record_registros.servicio || ';  TOTAL NETO => '
            || (SELECT precio_neto ((SELECT cod_cliente FROM corte_pelo
                                        WHERE record_registros.codigo = cod_cliente
                                        AND record_registros.fecha = fecha_corte),
                                            record_registros.fecha)) || '€';
        total_beneficio := total_beneficio + (SELECT precio_neto ((SELECT cod_cliente FROM corte_pelo
                                                WHERE record_registros.codigo = cod_cliente
                                                AND record_registros.fecha = fecha_corte),
                                                    record_registros.fecha));
        END LOOP;
```

```
                CLOSE cursor_registros;
        registros_salida := registros_salida || '  >>>>>>>>>>>>  TOTAL DE GANANCIAS => ' || total_beneficio || '€';
                RETURN registros_salida;
        END;
        $$ LANGUAGE PLPGSQL;

SELECT get_registros ('2018', '2021');
SELECT get_registros ('2020', '2020');
SELECT get_registros ('2021', '2021');
```

---------- 2 TRIGGERS ----------

--Trigger para asegurar que los deletes en productos no puedan afectar al codigo 0;

```
CREATE OR REPLACE FUNCTION eliminar_producto()
        RETURNS TRIGGER AS $$
                DECLARE codigo int := OLD.cod_prod;
        BEGIN
                IF codigo = 0 THEN
                        RAISE EXCEPTION 'TRIGGER: La entrada con codigo 0 no se puede borrar';
                ELSE
                        DELETE FROM productos WHERE cod_prod = NEW.cod_prod;
                END IF;
                RETURN OLD;
        END;
        $$ LANGUAGE PLPGSQL;

CREATE TRIGGER protege_productos_delete BEFORE DELETE
        ON productos FOR EACH ROW
        EXECUTE PROCEDURE eliminar_producto();
```

```
DELETE FROM productos WHERE cod_prod = 0;

DELETE FROM productos WHERE cod_prod = 7;

DELETE FROM productos WHERE nom_prod = 'Producto Obsoleto';

SELECT * FROM productos;


-- Esta funcion y trigger se disparan al introducir dos codigos de producto dentro de la tabla combinar,

-- se comprueba si el producto existe, en caso afirmativo se suma cantidad al resultado y se resta de

-- los sumandos, en caso contrario se crea el nuevo producto y se anyade a la tabla productos;

CREATE OR REPLACE FUNCTION combinar_productos()

RETURNS TRIGGER AS $$

DECLARE cod1 int := NEW.cod_prod1;

DECLARE cod2 int := NEW.cod_prod2;

DECLARE nom1 varchar := (SELECT nom_prod FROM productos WHERE cod_prod = cod1);

DECLARE nom2 varchar := (SELECT nom_prod FROM productos WHERE cod_prod = cod2);

DECLARE cant1 int := (SELECT cantidad_prod FROM productos WHERE cod_prod = cod1);

DECLARE cant2 int := (SELECT cantidad_prod FROM productos WHERE cod_prod = cod2);

DECLARE prec1 numeric := (SELECT precio_prod FROM productos WHERE cod_prod = cod1);

DECLARE prec2 numeric := (SELECT precio_prod FROM productos WHERE cod_prod = cod2);

DECLARE nom_combinado varchar := nom1 || ' - ' || nom2;

DECLARE precio_combinado numeric := prec1 + prec2;

BEGIN

IF cant1 > 0 AND cant2 > 0 THEN

IF EXISTS (SELECT nom_prod FROM productos WHERE nom_prod = nom_combinado) THEN

RAISE INFO 'Trigger de combinacion que actualiza las cantidades en productos';

UPDATE productos SET cantidad_prod = cantidad_prod -1 WHERE cod_prod = cod1;

UPDATE productos SET cantidad_prod = cantidad_prod -1 WHERE cod_prod = cod2;

UPDATE productos SET cantidad_prod = cantidad_prod +1 WHERE nom_prod = nom_combinado;

ELSE

RAISE INFO 'Trigger de combinacion que anyade un nuevo producto y resta cantidades';

UPDATE productos SET cantidad_prod = cantidad_prod -1 WHERE cod_prod = cod1;
```

```
                UPDATE productos SET cantidad_prod = cantidad_prod -1 WHERE cod_prod = cod2;
                INSERT INTO productos (nom_prod, descr_prod, cantidad_prod, precio_prod)
                        VALUES (nom_combinado, 'Combinacion', 1, precio_combinado);
                END IF;
        ELSE
                RAISE INFO 'Combinacion fallida, no hay suficientes productos para % ', nom_combinado;
        END IF;
        RETURN NEW;
    END;
    $$ LANGUAGE PLPGSQL;

CREATE TRIGGER actualiza_cantidades BEFORE INSERT
    ON combinar FOR EACH ROW
    EXECUTE PROCEDURE combinar_productos();

INSERT INTO combinar (cod_prod1, cod_prod2) VALUES (0,5);
INSERT INTO combinar (cod_prod1, cod_prod2) VALUES (6,5);
INSERT INTO combinar (cod_prod1, cod_prod2) VALUES (5,4);

SELECT * FROM productos;
SELECT * FROM combinar;
```

8. CONCLUSIONES

Para empezar diré que he tenido muchos problemas para desarrollar el proyecto ya que el tema no es mi especialidad, cuando empecé no tenía la confianza suficiente como para desarrollar algo que controlara demasiado, pensé en hacer algo simple sin saber que la complejidad exigía que supiera mucho del tema. Como justificación diré que la base de datos era para otra persona muy cercana a mi al inicio del curso pero que en la tercera evaluación ya no forma parte de mi vida, pensé en empezar de cero pero acabé descartando la idea ya que tenía el modelo entidad relación y relacional ya formado, por tanto, esta base de datos ha terminado siendo algo útil solo a nivel técnico, que me ha enseñado practicando y probando a enrevesar las tablas cada vez más, y dejándome una gran sensación de querer empezar mi propio sistema sin depender de nadie.

En cuanto al apartado técnico, veo que falta mucho para que la base sea verdaderamente funcional, habría que añadir más triggers que sirvieran de restricciones para que no se actualicen los “códigos 0” que menciono a menudo, más funciones que permitieran extraer los datos de una manera más óptima y por supuesto utilizar el resto de tablas como gastar y herramientas que no he tenido espacio para hacer pero me he negado a eliminar, aunque a cada trimestre he tenido que revistar todo el contenido casi desde el principio, modificando relaciones, añadiendo nuevas pk´s para que no obtener problemas e incluso borrando atributos que en un principio consideraba indispensables.

Finalmente, aunque la base no sea lo más útil del mundo, funciona perfectamente y creo que he conseguido aplicar todos los conocimientos que he ido aprendiendo a lo largo del curso lo que me ha hecho asentar cada uno de los conceptos para empezar ahora mi propio proyecto.