TAREA 1

CREACIÓN DE UNA BASE DE DATOS SQL

FACULTAD DE ESTUDIOS ESTADÍSTICOS

MASTER BIG DATA & BUSINESS ANALYTICS

AUTORA : ESTHER FERNÁNDEZ BARCHÍN

FECHA DE ENTREGA: 10/03/2022

1. DISEÑO CONCEPTUAL

1.1. Especificación de requisitos

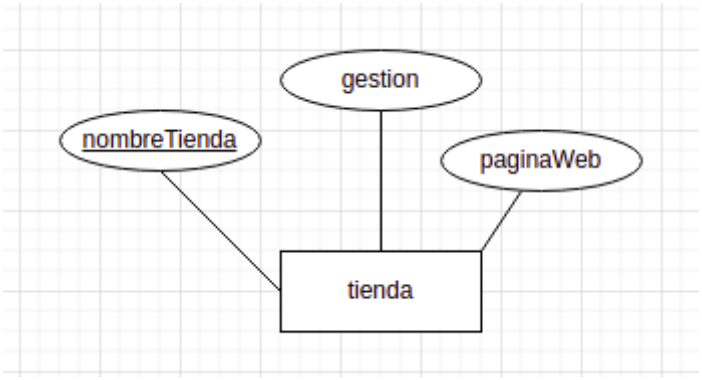
Esta base de datos que crearemos tiene como foco las aplicaciones de móvil, en el cual vemos actores de la vida real como las tiendas de aplicaciones, las empresas de servicios y los usuarios, que conformarían algunas de las entidades que tendremos que diferenciar para poder crear la base de datos con la que trabajaremos.

En primer lugar para permitir la creación de la base de datos especificada, hay que identificar las entidades que lo configuran.

1.2. Entidades y sus atributos

En este apartado se diferenciará las entidades que configuran la base de datos junto con sus atributos y las características propias de estas entidades tal cual se supone del enunciado.

- Tiendas de aplicaciones que se identificará como la entidad *tienda*

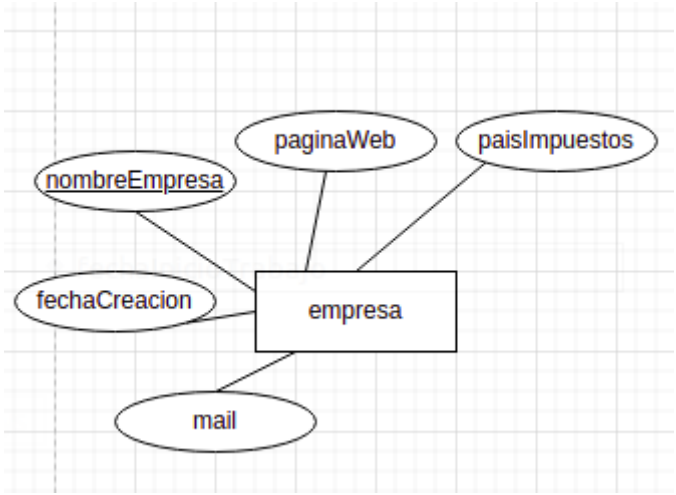


Sus atributos tal y como salen representados son :

- Nombre de la tienda como *nombreTienda* (para que sea mucho más fácil su lectura y diferenciación en la base de datos) y está se clasificaría como su clave primaria, ya que según el texto el nombre de la tienda es distinto para cada una.
- Nombre de quién gestiona la tienda como *gestion.*
- Dirección web como *paginaWeb.*

En este caso, los atributos definidos son simples y no muestran a priori ninguna característica especial.

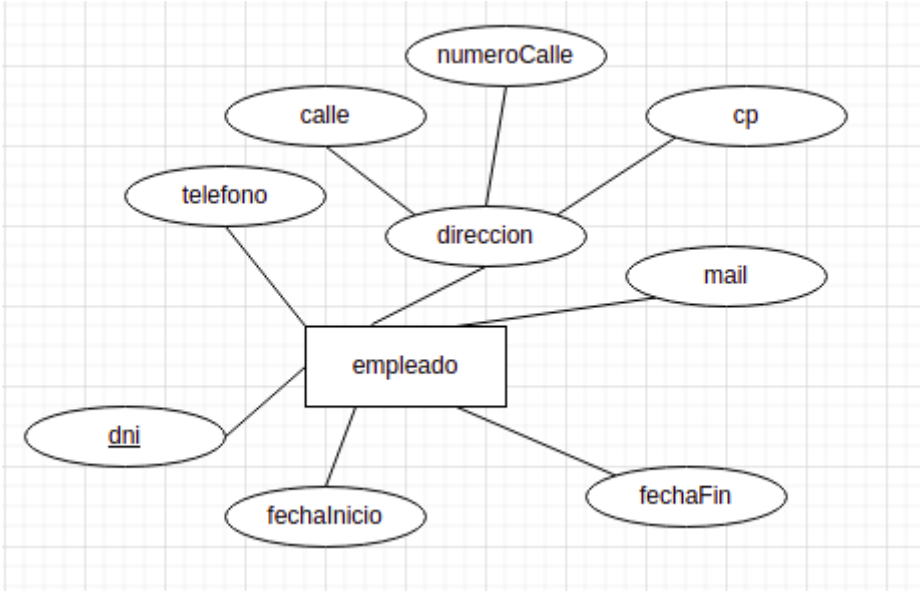
- Empresas de servicios que las identificaremos como la entidad *empresa:*



Sus atributos son:

- Nombre de la empresa definido como *nombreEmpresa,* esta tendrá que ser la clave primaria ya que en principio suponemos que dos empresas no tendrán el mismo nombre.
- País en el que la empresa paga sus impuestos, suponemos que lo paga solo en un país y por ello no tiene ninguna característica en especial. Lo definimos como *paisImpuestos.*
- Año de creación de la empresa como fecha*Creacion.*
- Correo electrónico como *mail,* aquí suponemos que cada empresa tiene un solo correo electrónico, en la realidad cada empresa tendrá varios según el departamento.
- Página web de la empresa como *paginaWeb.*

- Los empleados de la empresa los identificaremos como la entidad *empleado*.

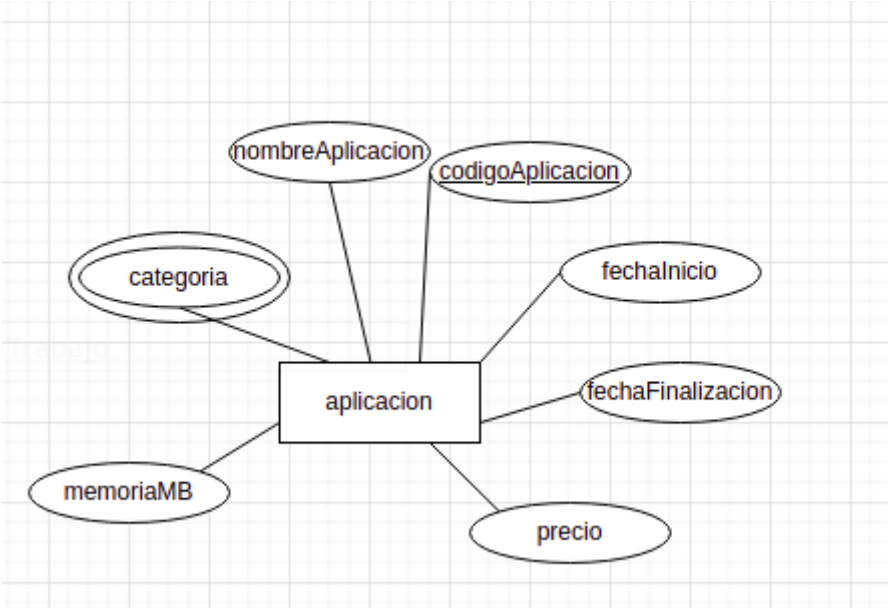


Los atributos de esta entidad los podemos enumerar de la siguiente manera:

- DNI del empleado será identificado como *dni* y al tener un valor único por empleado será la clave primaria.
- La dirección como *direccion* será un atributo compuesto que contendrá los siguientes atributos: *calle* , número como *numeroCalle* y código postal como *cp*.
- El correo electrónico estará definido como *mail.*
- El teléfono estará definido como *telefono.*

- También en el texto comenta que el empleado puede haber trabajado en varias empresas del sector e incluso puede trabajar en la misma empresa en distintos periodos de tiempo, esto lo representaremos con la fecha de Inicio en la que entró en la empresa como *fechaInicio* y la fecha de cuando termino, como *fechaFin*. Estos dos atributos pasará a estar en el nexo y tabla t*rabajoEmpresa.*

- Las aplicaciones se reconocerán como la entidad aplicacion.

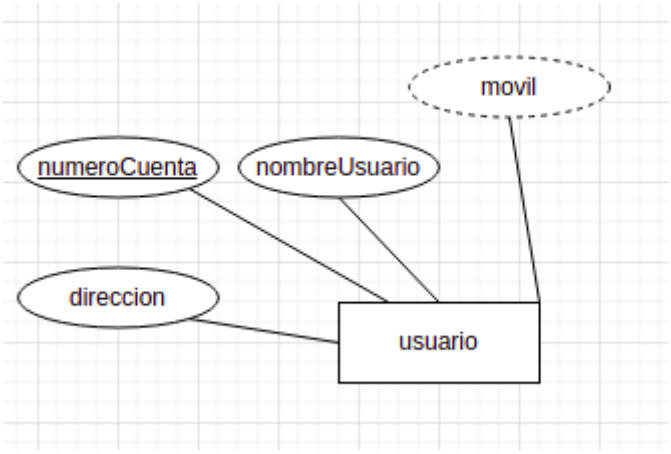


Los atributos serían:

- *nombreAlicacion* como el nombre de la Aplicación y como en el texto indica que es único, se le considerará UNIQUE.
- *codigoAplicacion* se refiere al código de aplicación y en el texto afirma que es distinto para cada aplicación, lo consideraremos la clave primaria.
- *memoriaMB* como el espacio de memoria que ocupa la aplicación.
- El precio como *precio*.

- La fecha en la que se comenzó a realizar la aplicación como *fechaInicio* y la fecha de terminación como *fechaFinalizacion.*
- La categoría de la aplicación como *categoria*, en este caso la consideramos como un atributo aunque en el texto se defina la relación con aplicaciones como : “ Una aplicación puede tener varias categorías y por supuesto una categoría puede ser de varias aplicaciones”. Esta relación necesitaría un atributo, como por ejemplo popularidad. Podríamos considerar este atributo como multivalorado ya que puede tener más de un valor simultáneamente. En este caso solo hemos considerado en una sola columna, aunque podría ponerse otra columna con el nombre *categoria2* o *subcategoria*.

- La entidad que comprende a los usuarios que se descargan las aplicaciones sería *usuario*



Los atributos a conocer serían:

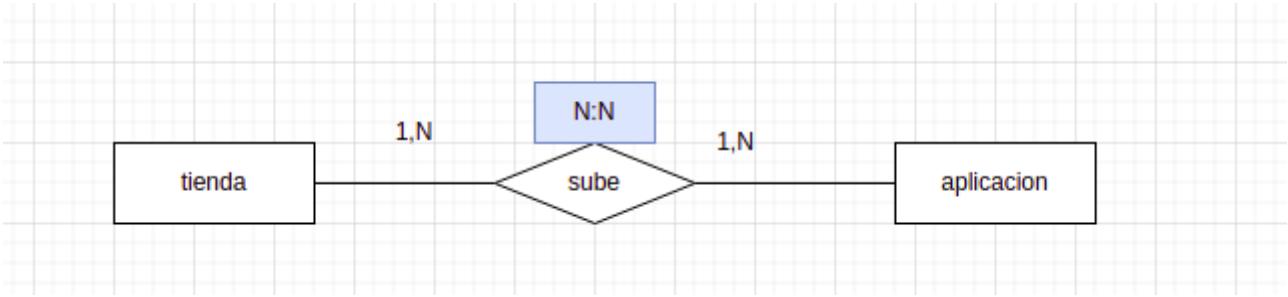
- El número de la cuenta de usuario como *numCuenta*, al ser único lo consideramos la clave primaria.
- El nombre del usuario como *nombreUsuario*.
- La dirección como *direccion es un atributo*
- El número de móvil,*movil*, si se descarga la aplicación en el teléfono, por lo tanto este atributo es opcional ya que esté valor puede ser null

1.3. Explicación de las relaciones entre entidades

En este apartado se establecerán las relaciones entre entidades y se explicarán para que en el posterior apartado podamos establecer el Modelo de Entidad-Relación completo.

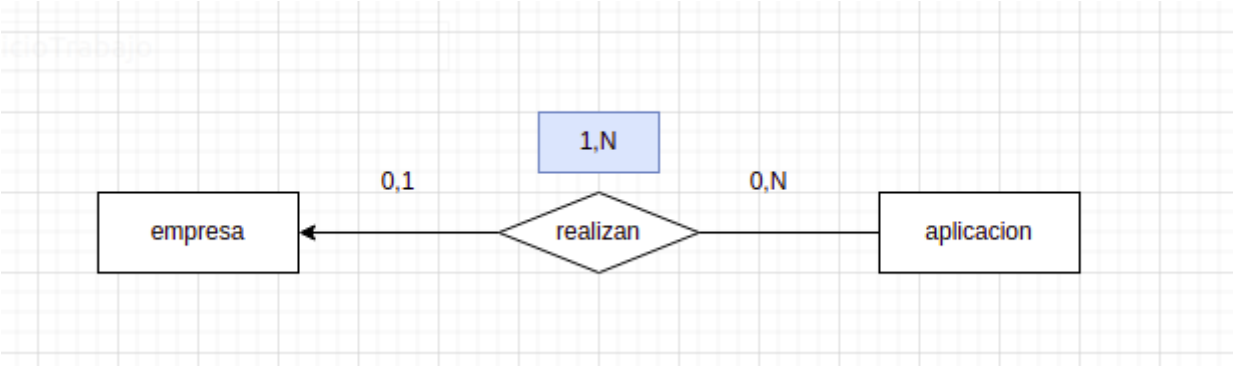
Relación de *tienda* y *aplicacion*

Las app se suben a las tiendas o plataformas de app, tiene como nexo la palabra *sube*.
En el texto se establece como: *"Una misma aplicación puede ser subida a varias tiendas, por supuesto una tienda tiene muchas aplicaciones"* que se definiría como una relación de muchas a muchas, debido a que responden a las siguientes preguntas:
- ¿Se puede subir una app en varias tiendas? Sí.
- ¿Varias tiendas pueden haber subido la misma aplicación? Sí.



Relación de *empresa* y *aplicacion*

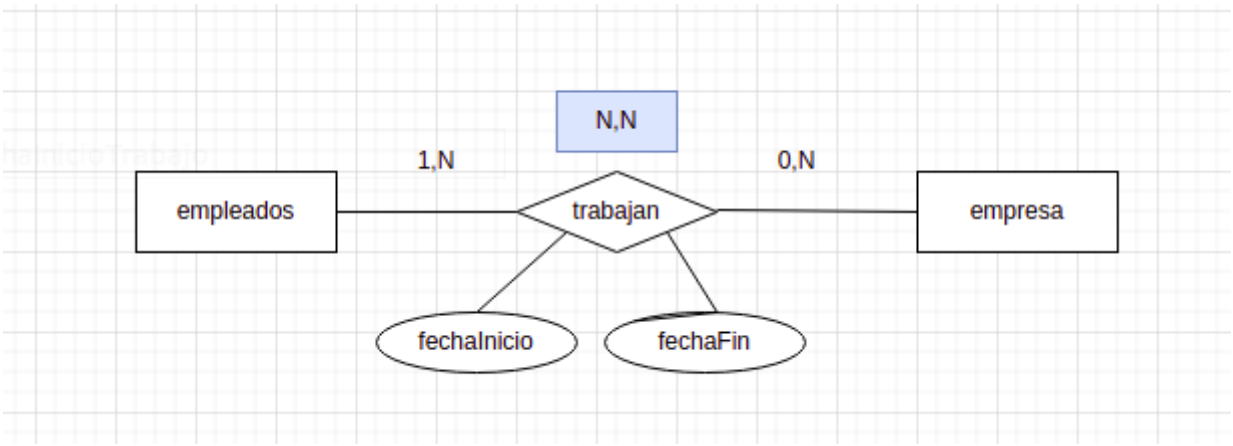
Esta relación está definida como *"Las empresas de servicio realizan las aplicaciones"*. Por tanto, el nexo de está relación es *realiza*. Esta relación se definiría como de uno a muchos ya que responden a las siguientes preguntas:
- ¿ Una empresa puede no haber realizado alguna app? Sí, puede que dicha aplicación esté en producción, por lo que no haya salido al mercado, pero en el contexto establecido se entiende que cada empresa realiza una aplicacion y que varias aplicaciones pueden haber sido realizadas por la misma empresa. Pero varias empresas no pueden haber realizado la misma aplicación. Esta relación la absorberá la tabla *aplicación*.



Relación de *empresa* y *empleado*

En el texto salen dos lineas de las cuales se pueden deducir la relación,

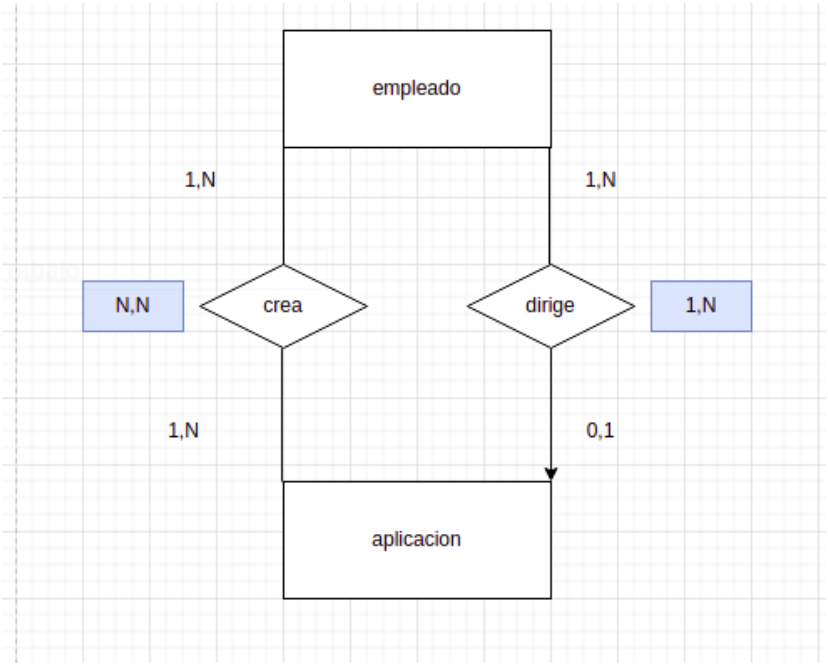
- *“ El empleado puede haber trabajado en varias empresas”* , esta relación habría que guardarla en un histórico de las veces que un empleado trabaja en una empresa , esto seria una relación de uno a muchos porque ¿un empleado puede haber trabajado en varias empresas? Si y en una empresa puede trabajar mas de un empleado si y en esa empresa puede volver a trabajar un empleado si, esto responde a la siguiente afirmación que establece esta relación
- *“Un empleado puede trabajar en la misma empresa en distintos periodos de tiempo”*, entonces tendríamos que establecer un nuevo atributo para contabilizar esto tal como fecha de inicio y fecha de fin o fecha de inicio e intervalo de tiempo en el cual el empleado ha trabajado en dicha empresa, aunque esto no nos asegura que el empleado pueda haber trabajado en un numero de empresas determinado a la vez, ya que esto no nos lo asegura el texto. En definitiva, esta relación se establecería de muchos a muchos, ya que un empleado puede haber trabajado en varias empresas y en una empresa pueden haber trabajado varios empleados.



Relación entre *aplicacion* y *empleado*

En esta relación tenemos que distinguir dos tipos de nexo:

- Para la frase *“ Cada aplicación está realizada por empleados”*, el nexo sería *está realizada.* ¿Varios empleados pueden realizar una aplicación? Sí. ¿Varias aplicaciones pueden ser realizadas por varios empleados?Sí. Entonces esto sería una relación de muchos a muchos.
- Para la frase *“Cada aplicación es dirigida por un solo empleado y un empleado puede dirigir varias aplicaciones”*, el nexo sería *es dirigida.*¿Cuantas aplicaciones puede dirigir un empleado? Varias. ¿Cuantos empleados pueden dirigir cada aplicación? Uno. Por lo que esta relación sería de uno a muchos. Esta última relación será absorbida por la tabla llamada *aplicacion* como *dniLider*.



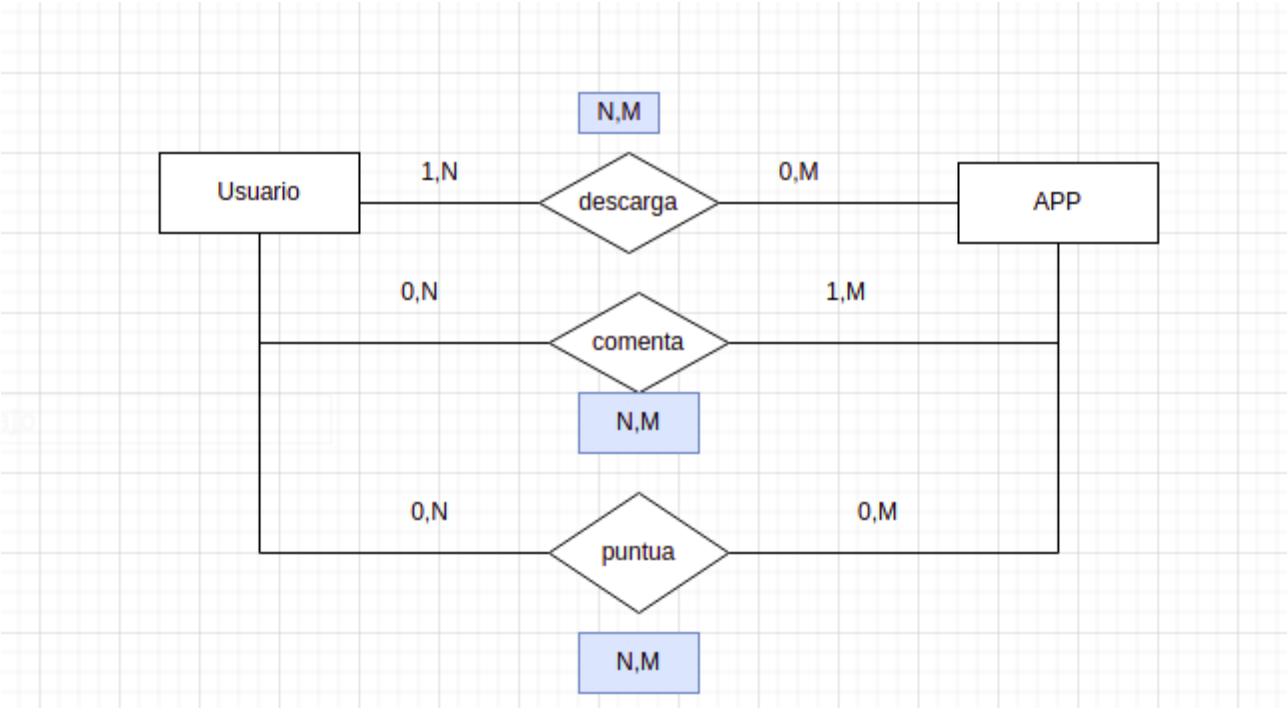
Relación entre *aplicacion* y *usuario*

Entre estas dos entidades en el texto podemos encontrar dos afirmaciones

- *“ Un usuario puede descargar o no aplicaciones, pero no puede descargar 2 veces la misma aplicación, la tiene que desinstalar y volverla a instalar”*. Este tipo de bases de datos lleva un registro actual, es decir, si tiene descargada la aplicación saldrá en la base de datos, en cambio si la desinstala, se borrará el registro de la base de datos. El nexo de esta relación sería *descarga*.La relación que se define sería de muchos a muchos ya que contesta afirmativamente a las siguientes

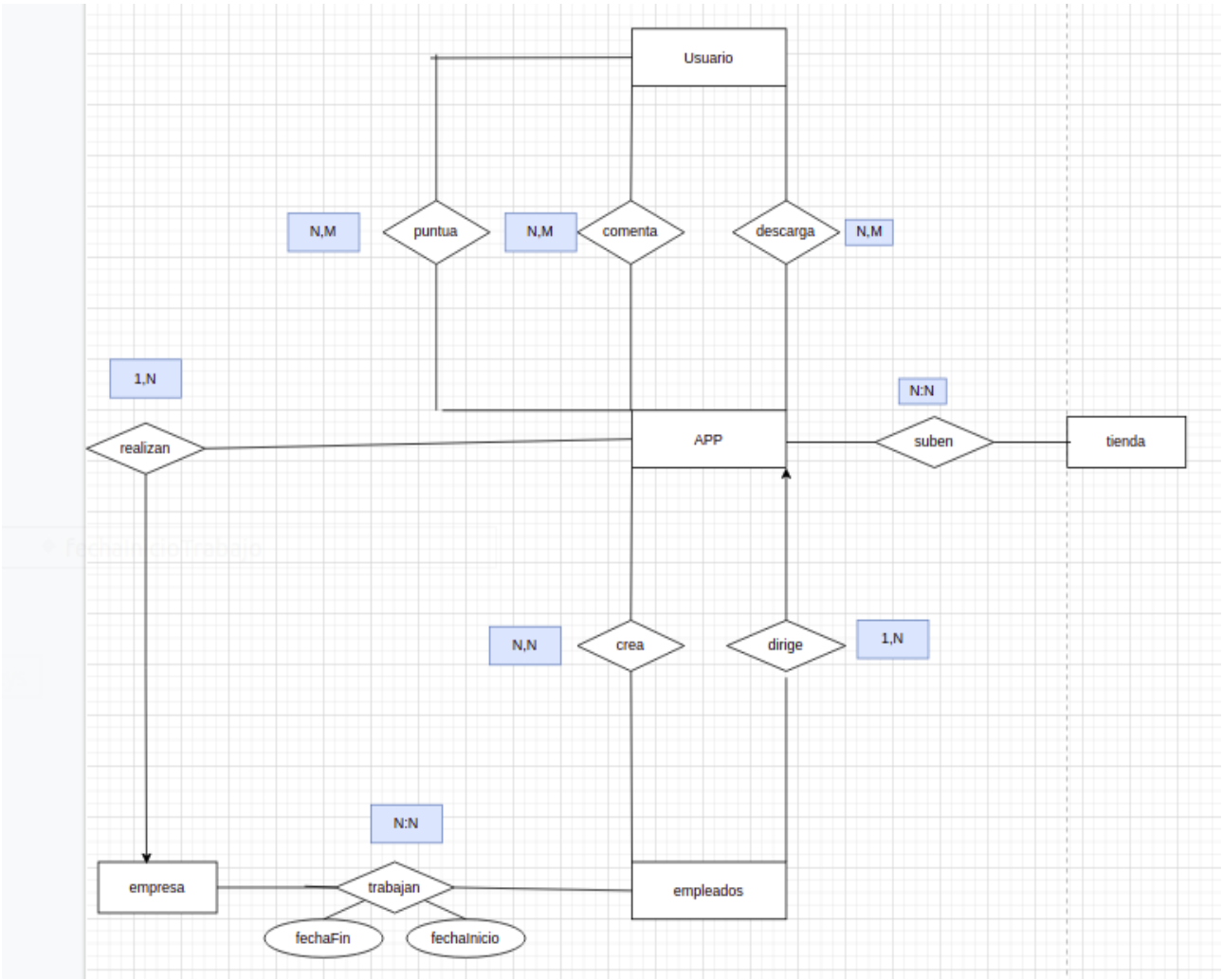
cuestiones.¿Un usuario puede descargar más de una app? Sí. ¿Una app puede ser descargada por más de un usuario? Sí.

- *"El usuario puede puntuar de 0 a 5 cada una de las apps que se descargan y hacer comentarios referentes a la misma"*. Se puede interpretar esta afirmación como que el usuario puede hacer más de un comentario sobre una app y varios comentarios pueden ser hechos por varios usuarios, se establecería como una relación de muchos a muchos. El nexo de estas relaciones sería *puntúa y comenta*. En cambio, el usuario al puntuar una aplicación una vez, para cambiarla ha de editar la puntuación que ya ha hecho. ¿ Cuántos usuarios pueden puntuar una aplicación? Varios. ¿Cuántas aplicaciones puede puntuar un usuario? Varias, se establecería una relación de muchos a muchos. Estas relaciones luego se dividirán en dos tablas.



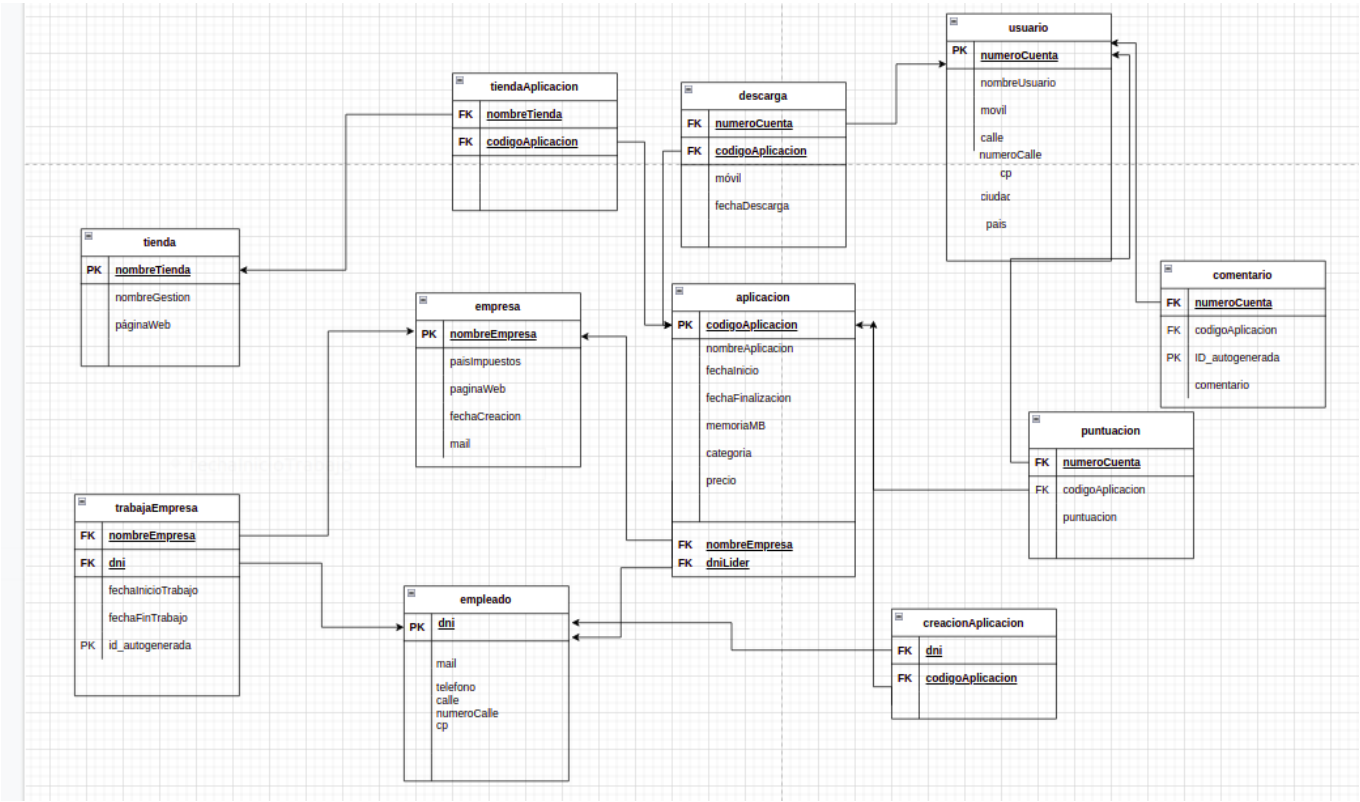
1.4. Modelo de Entidad-Relación

Al haber establecido ya todas las relaciones el modelo de E-R resultante sería el siguiente:


Usuario
N,M
puntua
N,M
comenta
descarga
N,M
1,N
N:N
realizan
APP
suben
tienda
N,N
crea
dirige
1,N
N:N
empresa
trabajan
empleados
fechaFin
fechaInicio

2. DISEÑO LÓGICO

Este es el paso anterior a la implantación de la base de datos en SQL, en el cual, tendremos que transformar el diagrama entidad-relación obtenido previamente para pasarlo al modelo relacional. Para ello, las relaciones establecidas hay que pasarlas a tablas, es decir, diferenciando la clave primaria de las claves ajenas.



Este sería el paso a tablas completo pero se explicarán las relaciones más relevantes.

En primer lugar, la entidad aplicación se ha transformado en una tabla con sus correspondiente clave primaria que es el *codigoAplicacion. Aunque hay que* destacar las dos claves foráneas con el *nombreEmpresa* y el *dniLider* que provienen de las tablas empresa y empleado.

Esto es debido a que tanto la relación de la empresa realiza aplicación y la aplicación es dirigida por un empleado, tienen relación uno a muchos, por lo que la tabla de aplicación absorbe ambas relaciones.

Las relaciones a comentar serían:

- trabajaEmpresa : las claves foráneas son *nombreEmpresa* y el *dni* ya que sirven para unir la relación entre trabajador y empresa. Además esta *relación* tiene una clave primaria auto generada, esto sirve para dictaminar que un empleado puede trabajar para una empresa más de una vez. Este tipo de clave sirve para controlar el intervalo de tiempo de un trabajador en cada empresa. La experiencia del empleado se mide con la *fechaInicio* y *fechaFinalizacion*.

- *descarga* : la relación entre usuario y aplicación tiene como claves foráneas tanto al *numeroCuenta* como *codigoAplicacion* para unir ambas entidades,además se ha puesto el móvil del usuario ya que cuando esté se descarga una aplicación sale el numero de móvil. Hemos supuesto que todos los usuarios de la base de datos se han descargado alguna aplicación pero podría darse el caso de que un usuario no esté en dicha tabla.

- *puntuacion* y *comentario*: un usuario en teoría al descargar una aplicación puede comentarla y puntuarla, hemos supuesto una vez que el usuario se ha descargado la aplicación ya puede hacer alguna de las dos opciones o las dos. En el caso de *puntuacion* unimos el *codigoAplicacion* y *numeroCuenta* como clave foránea además de la puntuación que va de 0 a 5. Además el comentario, tiene como clave foráneas las mismas que la puntuación, pero en este caso tenemos una clave primaria autogenerada, ya que suponemos que el usuario puede hacer más de un comentario a lo largo del tiempo.

Para finalizar, se comentará la dimensión de algunos de los atributos.
- Algunas dimensiones que pueden tener varios caracteres como *nombreTienda*, *gestion*, *paginaWeb*, *mail*, *nombreUsuario*, entre otros. Las hemos especificado como VARCHAR(40) ya que dichos nombres pueden tener varios caracteres,espacios, símbolos...
- *dni* lo hemos especificado como CHAR(9) ya que el Documento de Identidad tendrá obligatoriamente 9 caracteres. El teléfono lo hemos especificado como CHAR(12) ya que hemos contabilizado los espacios entre conjunto de números. EL código postal también está restringido a 5 dígitos.
- El numero de cuenta del usuario lo hemos especificado como NUMERIC (8), ya que puede haber varios usuarios en el mundo y así se podría cubrir gran número de ellos.

3.IMPLEMENTACIÓN

```
/*----------------------------------------------------------------------------
Esther Fernández Barchín
Nombre de la base de datos: tiendaMovil
----------------------------------------------------------------------------*/


/*----------------------------------------------------------------------------
Esther Fernández Barchín
Nombre de la base de datos: tiendaMovil
----------------------------------------------------------------------------*/


/*----------------------------------------------------------------------------
 Definición de la estructura de la base de datos
 ----------------------------------------------------------------------------*/

-- Creación de la base de datos --

drop DATABASE IF EXISTS tiendaMovil;
CREATE database tiendaMovil;
use tiendaMovil;

-- Tablas de entidades

CREATE TABLE tienda(
   nombreTienda VARCHAR(40) PRIMARY KEY,
   gestion VARCHAR(40) NOT NULL,
   paginaWeb VARCHAR(40)
);

CREATE TABLE empresa(
          nombreEmpresa VARCHAR(40) PRIMARY KEY,
    paisImpuestos CHAR(40) NOT NULL,
```

```
    paginaWeb VARCHAR(50),
    fechaCreacion DATE NOT NULL,
    mail VARCHAR(40)
);

CREATE TABLE empleado(
        dni CHAR(9),
    nombre VARCHAR(40),
    primerApellido VARCHAR(40),
    mail VARCHAR(40),
    telefono CHAR(12),
    calle VARCHAR(20),
    numeroCalle NUMERIC(4),
    cp CHAR(5),
    PRIMARY KEY (dni)
);

CREATE TABLE usuario(
        numeroCuenta NUMERIC(8),
    nombreUsuario VARCHAR(40),
    movil CHAR(15) NOT NULL,
    calle VARCHAR(40),
    numeroCalle NUMERIC(4),
    cp CHAR(5),
    ciudad VARCHAR(20),
    pais VARCHAR(20) NOT NULL,
    PRIMARY KEY (numeroCuenta)
);

CREATE TABLE aplicacion(
    codigoAplicacion CHAR(10) PRIMARY KEY,
        nombreAplicacion VARCHAR(40) UNIQUE,
    fechaInicio DATE,
    fechaFinalizacion DATE,
    memoriaMB DECIMAL(4,2),
```

```
    categoria CHAR(20),
    precio DECIMAL(4,2),
    nombreEmpresa VARCHAR(40),
    dniLider CHAR(9),
        FOREIGN KEY (nombreEmpresa) REFERENCES empresa(nombreEmpresa)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (dniLider) REFERENCES empleado(dni)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- Tablas de relaciones --

CREATE TABLE descarga(
        numeroCuenta NUMERIC(8),
    codigoAplicacion CHAR(10),
    movil CHAR(15) NOT NULL,
    fechaDescarga DATE,
    PRIMARY KEY (numeroCuenta,codigoAplicacion),
    FOREIGN KEY (numeroCuenta) REFERENCES usuario(numeroCuenta)
    ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (codigoAplicacion) REFERENCES aplicacion(codigoAplicacion)
    ON DELETE CASCADE
    ON UPDATE CASCADE);

CREATE TABLE tiendaAplicacion(
        nombreTienda VARCHAR(40),
    codigoAplicacion CHAR(10),
    PRIMARY KEY (nombreTienda,codigoAplicacion),
    FOREIGN KEY (nombreTienda) REFERENCES tienda(nombreTienda)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (codigoAplicacion) REFERENCES aplicacion(codigoAplicacion)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE puntuacion(
        numeroCuenta NUMERIC(8),
        codigoAplicacion CHAR(10),
    puntuacion DECIMAL(4,2),
    fecha DATE,
    PRIMARY KEY(numeroCuenta,codigoAplicacion),
    FOREIGN KEY(numeroCuenta) REFERENCES usuario(numeroCuenta)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY(codigoAplicacion) REFERENCES aplicacion(codigoAplicacion)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE comentario(
    comentarioId int AUTO_INCREMENT PRIMARY KEY,
    numeroCuenta NUMERIC(8),
    codigoAplicacion CHAR(10),
    comentario TEXT,
    fecha DATE,
    FOREIGN KEY (numeroCuenta) REFERENCES usuario(numeroCuenta)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (codigoAplicacion) REFERENCES aplicacion(codigoAplicacion)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE creacionAplicacion(
        dni CHAR(9),
    codigoAplicacion CHAR(10),
```

```
    PRIMARY KEY (dni,codigoAplicacion),
    FOREIGN KEY (dni) REFERENCES empleado(dni)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY(codigoAplicacion) REFERENCES aplicacion(codigoAplicacion)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE trabajoEmpresa(
    trabajoEmpresaId int AUTO_INCREMENT PRIMARY KEY,
    dni CHAR(9),
        nombreEmpresa VARCHAR(40),
    fechaInicioTrabajo DATE,
    fechaFinTrabajo DATE,
    FOREIGN KEY (nombreEmpresa) REFERENCES empresa(nombreEmpresa)
        ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (dni) REFERENCES empleado(dni)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

/* ----------------------------------------------------------------------------------
Inserción de datos
----------------------------------------------------------------------------------*/
-- Cargar datos en worbench --
-- Hay tres formas de hacerlo, la primera es mediante LOAD DATA ON FILE pero ha estado dando
-- varios errores por lo que hemos cargado la tabla a traves del Import Table Wizard
-- También se puede añadir datos a traves de INSERT INTO, que lo haremos más adelante

--
/*----------------------------------------------------------------------------------
Consultas,modificaciones,borrados y vistas con enunciado
----------------------------------------------------------------------------------*/
```

-- Consulta 1: Veces en la que se ha descargado cada aplicación y fecha --

```
SELECT codigoAplicacion, count(*) AS VecesDescargadas, fechaDescarga FROM descarga
GROUP BY codigoAplicacion
HAVING count(*) > 1;
```

-- Consulta 2: Fecha en la que se descarga mayor número de Aplicaciones --

```
SELECT fechaDescarga, count(*) AS NumeroDescargas FROM descarga
GROUP BY codigoAplicacion
HAVING count(*) > 1
LIMIT 1;
```

-- Consulta 3. Los países de los usuarios que más aplicaciones han descargado --

```
SELECT  u.pais, count(*) AS VecesDescargadas
FROM usuario AS u INNER JOIN descarga AS des ON u.numeroCuenta = des.numeroCuenta
GROUP BY pais
HAVING count(*) > 1
ORDER BY  VecesDescargadas DESC;
```

-- Consulta 4. La puntuación media de cada una de las aplicaciones --

-- Como no todas las apps están puntuadas vamos a hacer un INSERT INTO con nuevos datos
-- para las aplicaciones que estan sin puntuar

```
INSERT INTO puntuacion VALUES ('3577','102','4.0','2021-12-30');
INSERT INTO puntuacion VALUES ('1071','103','2.0','2021-12-30');
INSERT INTO puntuacion VALUES ('3601','104','4.1','2021-12-30');
INSERT INTO puntuacion VALUES ('3944','105','4.2','2021-12-30');
INSERT INTO puntuacion VALUES ('2792','106','5.0','2021-12-30');
INSERT INTO puntuacion VALUES ('2196','107','2.6','2021-12-30');
```

```
INSERT INTO puntuacion VALUES ('3613','108','4.7','2021-12-30');
INSERT INTO puntuacion VALUES ('3007','109','4.1','2021-12-30');
INSERT INTO puntuacion VALUES ('2137','110','4.2','2021-12-30');



SELECT  a.codigoAplicacion, a.nombreAplicacion, AVG(puntuacion)AS
MediaPuntuadaPorAplicacion
FROM aplicacion AS a INNER JOIN puntuacion AS pu ON a.codigoAplicacion =
pu.codigoAplicacion
GROUP BY nombreAplicacion
ORDER BY MediaPuntuadaPorAplicacion desc;

-- Vista 1 --
-- En esta vista queremos crear una nueva tabla para poder ver ver que empleados trabajan en cada
empresa y la experiencia que tienen en cada una

CREATE VIEW trabajoresEmpresa (nombreEmpresa,idTrabajador, nombre,
apellido,fechaInicioTrabajo,
fechaFinTrabajo) AS
SELECT  tra.nombreEmpresa,em.dni,
em.nombre,em.primerApellido,tra.fechaInicioTrabajo,tra.fechaFinTrabajo
FROM empleado AS em INNER JOIN trabajoEmpresa AS tra ON em.dni = tra.dni;

-- Consulta 5. Ver cuantos empleados trabajan en cada empresa --

SELECT nombreEmpresa,count(*) as trabajadoresxEmpresa
FROM tiendaMovil.trabajoresEmpresa
GROUP BY nombreEmpresa;

-- Consulta 6. ¿Quién trabaja en CDC? ¿Cúal es su experiencia? --

SELECT nombreEmpresa,idTrabajador, nombre,
apellido,TIMESTAMPDIFF(YEAR,fechaInicioTrabajo,fechaFinTrabajo) AS experiencia
FROM tiendaMovil.trabajoresEmpresa
```

```
WHERE nombreEmpresa = 'CDC'
ORDER BY experiencia DESC;

-- Consulta 7. ¿ Cuál es el nombre de quién dirige la creación de cada una de las aplicaciones? --

SELECT  nombreAplicacion,dniLider, nombre as nombreLider
FROM aplicacion AS a LEFT JOIN empleado AS e ON a.dniLider = e.dni;

-- Consulta 8. ¿ Cuál es la aplicación más cara? --

SELECT codigoAplicacion, nombreAplicacion,precio
FROM aplicacion
WHERE precio = (select max(precio) from aplicacion );

-- Consulta 9. ¿Cómo distribuyen las aplicaciones en categorias?--
SELECT categoria, count(*) as mismaCategoria
FROM aplicacion
GROUP BY categoria;

-- Consulta 10. ¿Cuáles son las empresas que crean menos aplicaciones?--
SELECT  nombreEmpresa, count(*) as creacionAplicaciones
FROM aplicacion
GROUP BY nombreEmpresa
HAVING creacionAplicaciones < 2;

-- Consulta 11. Nombre de las tiendas que suben las Aplicaciones --
SELECT ti.nombreTienda, a.nombreAplicacion
FROM tiendaAplicacion as ti RIGHT JOIN aplicacion as a
ON ti.codigoAplicacion = a.codigoAplicacion;

-- Consulta 12.Usuarios que NO han descargado una aplicación --

-- Para esta consulta tenemos que inventarnos un nuevo usuario pero que no va a estar
-- en la tabla descargas
```

```
INSERT INTO usuario VALUES('3894', 'efdezbarchin', '640 523 144', 'Calle Olimpia', '20', '28234', 'Madrid', 'España' );

SELECT u.numeroCuenta,u.nombreUsuario
FROM usuario as u LEFT JOIN descarga as des ON u.numeroCuenta = des.numeroCuenta
WHERE des.numeroCuenta IS NULL;

-- Consulta 13. De los que han creado Aplicaciones, ¿había alguno que dirige?

SELECT a.dniLider
FROM creacionAplicacion as cre LEFT JOIN aplicacion
ON cre.dni = a.dniLider ;

-- Consulta 14. Cuando los usuarios puntuan, ¿cuál es la puntuación más reciente a fecha de hoy?
-- (calculado en meses) --
-- hay una formula que no me ha salido en donde puedes comparar la fecha que está en base de datos
-- con now(), asi que como alternativa he escogido la forma fácil

SELECT numeroCuenta, puntuacion,fecha, TIMESTAMPDIFF(MONTH,fecha,CURDATE()) as meses
FROM puntuacion
HAVING meses < 7
ORDER BY fecha;

-- Consulta 15.Nombre de las empresas en las que trabajan  más de 3 empleados --


SELECT nombreEmpresa, count(*) as trabajanPorEmpresa
FROM trabajoEmpresa
GROUP BY nombreEmpresa
HAVING trabajanPorEmpresa > 3;

/*----------------------------------------------------------------------------------------
Trigger
```

El trigger será de update, ya que dos usuarios han comentado las aplicaciones recientemente por lo que necesitaremos actualizar la tabla llamada comentario */

```
delimiter //
DROP TRIGGER IF EXISTS nuevoComentario;
 CREATE TRIGGER  nuevoComentario BEFORE UPDATE ON comentario FOR EACH ROW
  BEGIN
    INSERT INTO  comentario VALUES
(new.comentarioId,new.numeroCuenta,new.codigoAplicacion,new.comentario,now());

END //
INSERT INTO comentario(numeroCuenta,codigoAplicacion,comentario) VALUES ('3601', '101', 'Good');
INSERT INTO comentario(numeroCuenta,codigoAplicacion,comentario) VALUES ('1071', '101', 'Bad');
```