

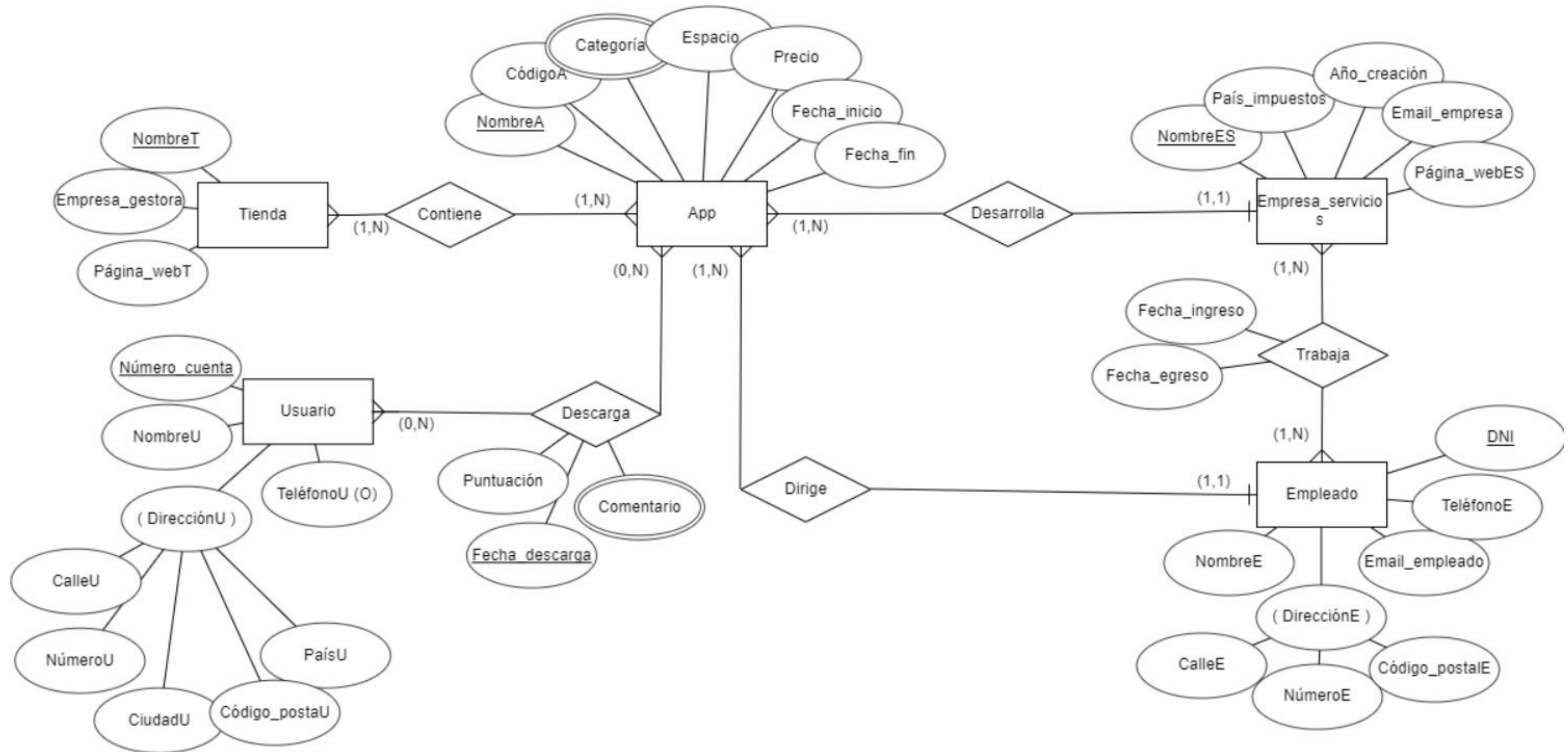


U N I V E R S I D A D  
COMPLUTENSE  
M A D R I D

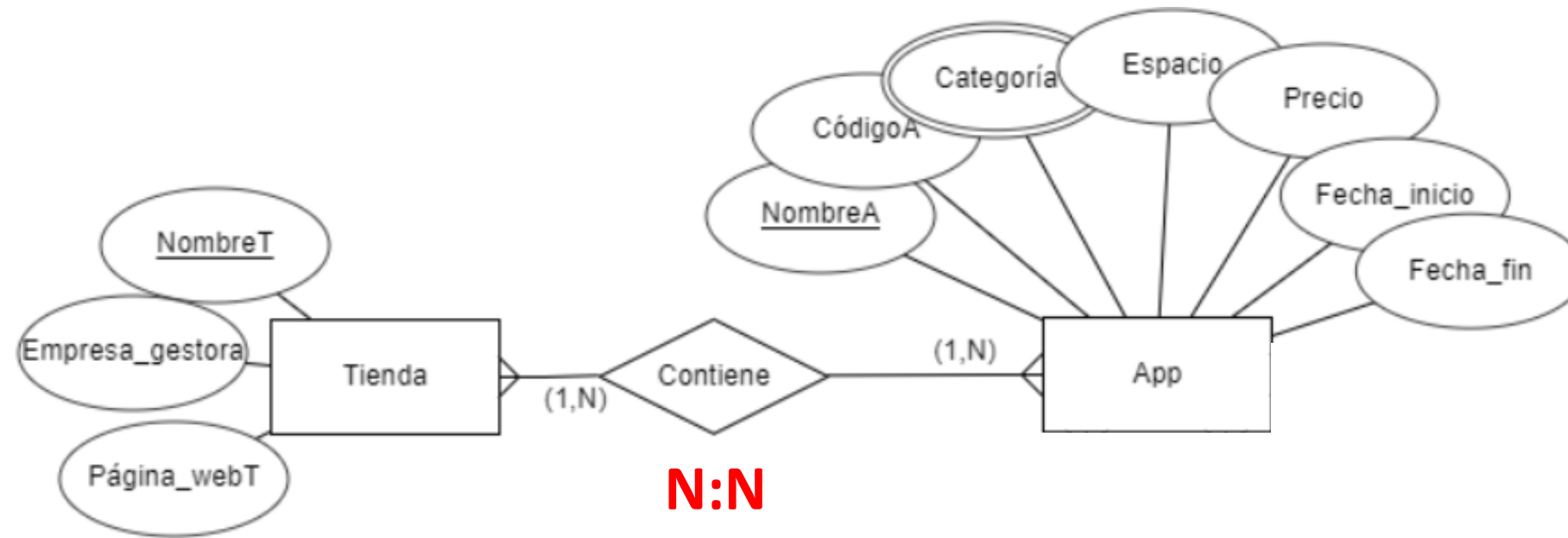
## Ejercicios para el proyecto final

### ‘Creación de una Base de Datos’

# Diseño conceptual

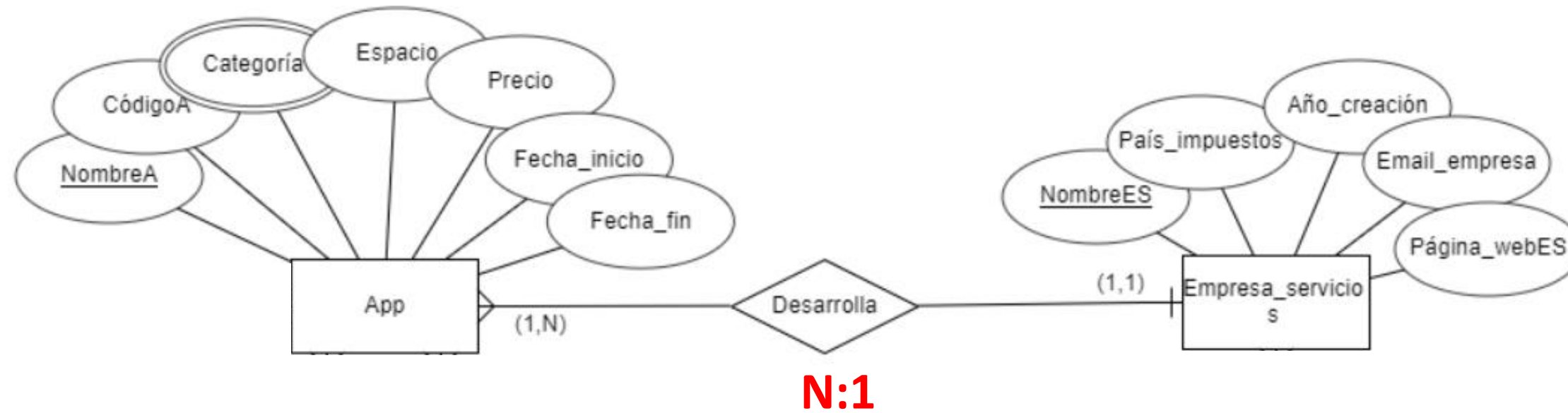


# Diseño conceptual



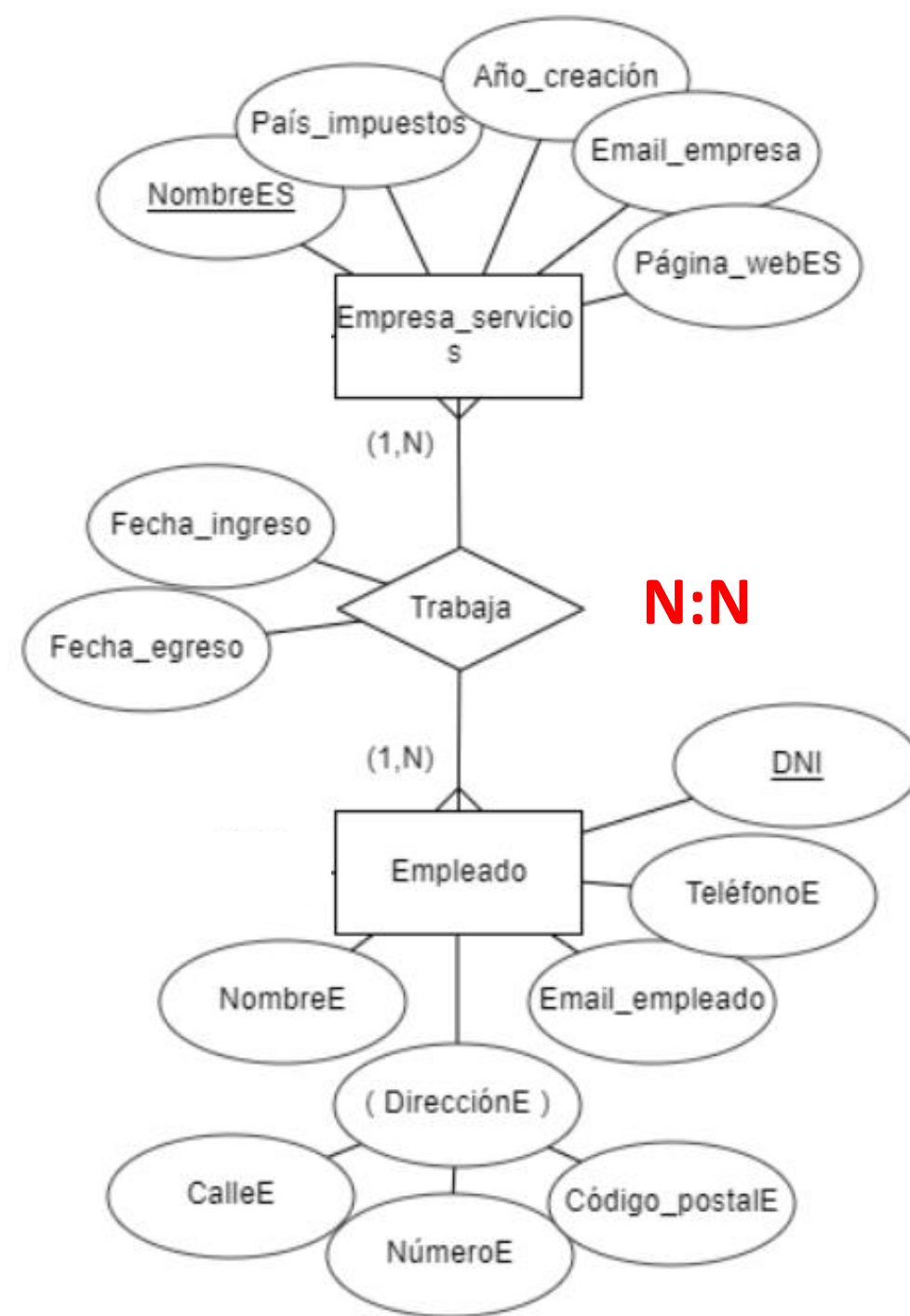
Entidad	Atributos relacionados	Tipo de atributo	Dominio	Cardinalidades
Tienda	NombreT	Clave	Nombres	(1,N)
	Empresa_gestora	Simple	Empresas	
	Página_webT	Simple	URLs	
App	NombreA	Clave	Nombres	(1,N)
	CódigoA	Simple	Códigos alfanuméricos	
	Categoría	Multivariado	Categorías	
	Espacio	Simple	Números	
	Precio	Simple	Decimales	
	Fecha_inicio	Simple	Fecha	
	Fecha_fin	Simple	Fecha	

# Diseño conceptual



Entidad	Atributos relacionados	Tipo de atributo	Dominio	Cardinalidades
App	Detalle en página anterior			(1,N)
Empresa_servicios	NombreES	Clave	Nombres	(1,1)
	País_impuestos	Simple	Países	
	Año_creación	Simple	Años	
	Email_empresa	Simple	Empresas	
	Página_webES	Simple	URLs	

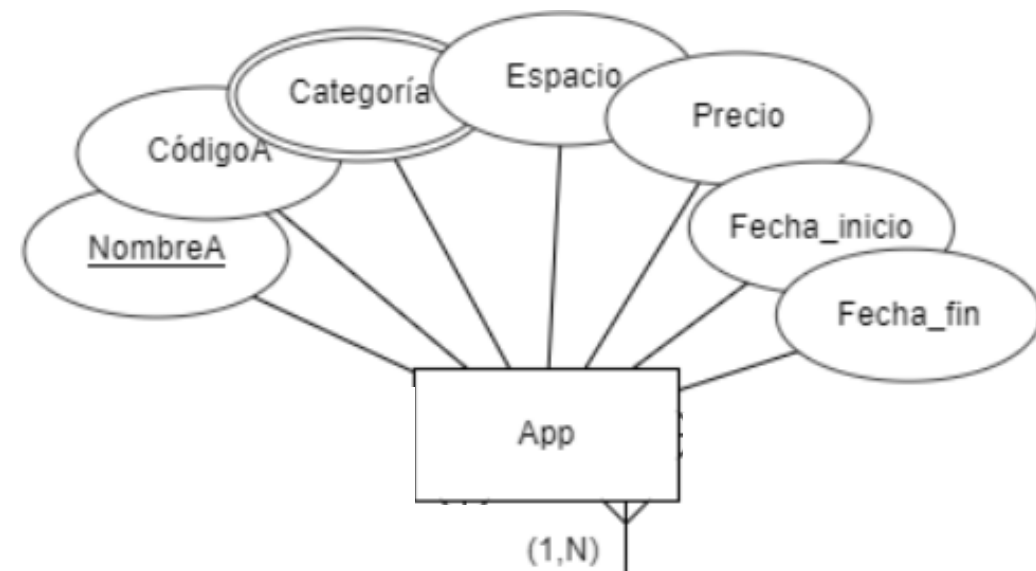
# Diseño conceptual



Entidad	Atributos relacionados	Tipo de atributo	Dominio	Cardinalidades
Empresa_servicios	Detalle en página anterior			(1,N)
Empleado	DNI	Clave	Números enteros	(1,N)
	NombreE	Simple	Nombres	
	<b>DirecciónE:</b> <ul style="list-style-type: none"><li>CalleE</li><li>NúmeroE</li><li>Código_postalE</li></ul>	Compuesto	<ul style="list-style-type: none"><li>Calles</li><li>Números enteros</li><li>Números enteros</li></ul>	

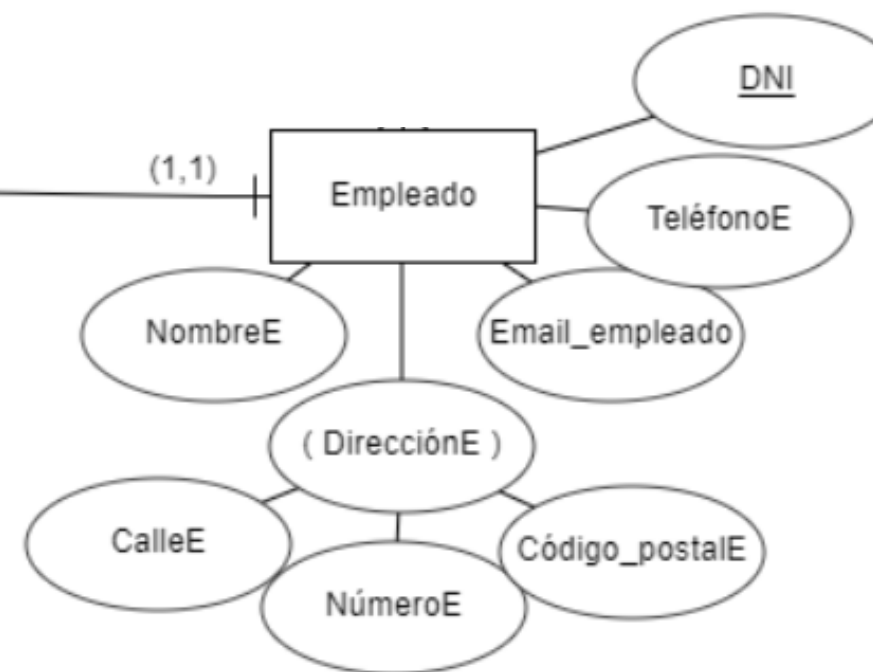
Relación	Atributos relacionados	Tipo de atributo	Dominio
Trabaja	Fecha_ingreso	Simple	Fecha
	Fecha_egreso	Simple	Fecha

# Diseño conceptual

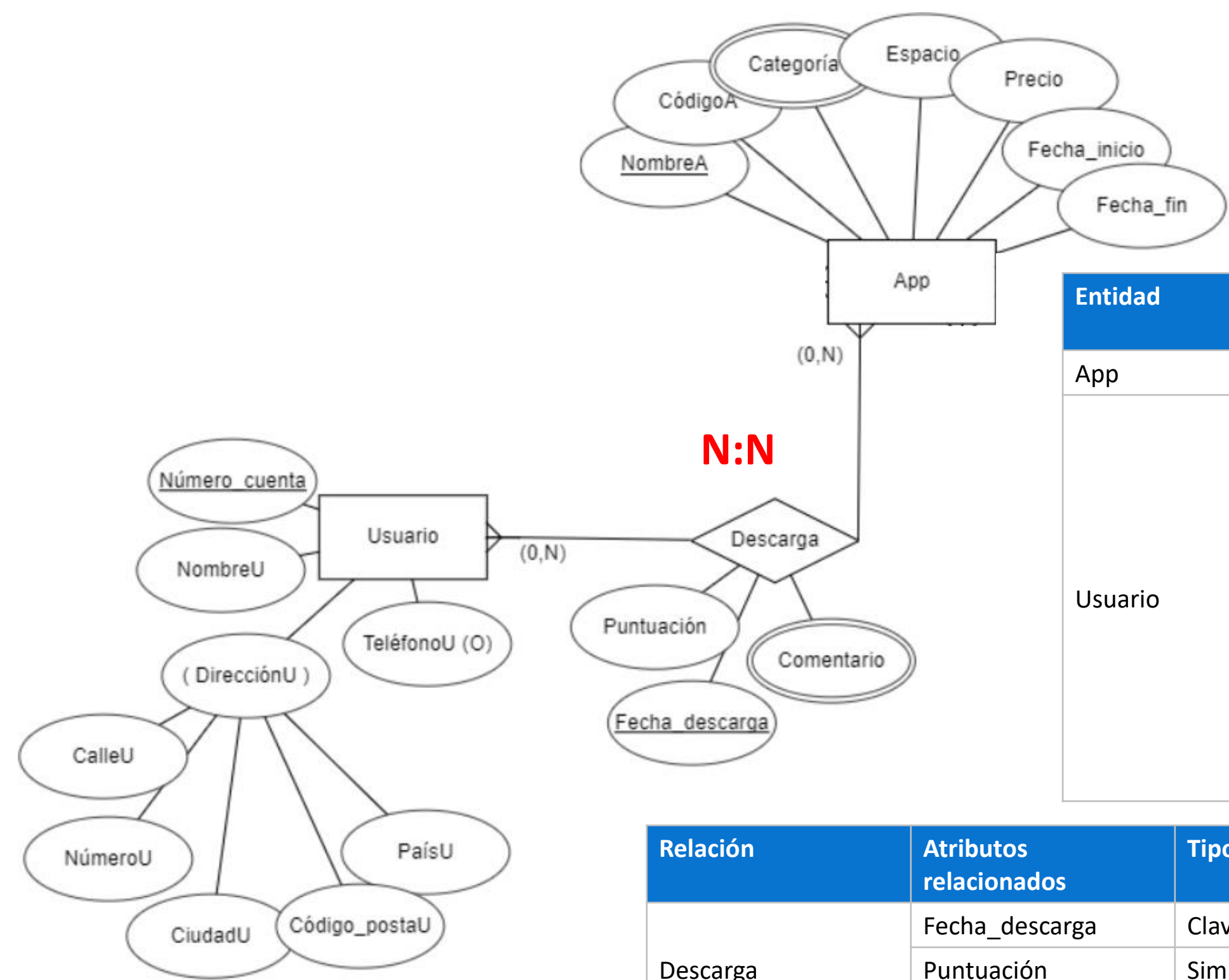


Entidad	Atributos relacionados	Tipo de atributo	Dominio	Cardinalidades
App	Detalle en página 3			(1,N)
Empleado	Detalle en página anterior			(1,1)

**N:1**



# Diseño conceptual



Entidad	Atributos relacionados	Tipo de atributo	Dominio	Cardinalidades
App	Detalle en página anterior			(0,N)
Usuario	Número_cuenta	Clave	Nombres	(0,N)
	NombreU	Simple	Países	
	<b>DirecciónU:</b> <ul style="list-style-type: none"><li>CalleU</li><li>NúmeroU</li><li>CiudadU</li><li>Código_postalU</li><li>PaísU</li></ul>	Compuesto	<ul style="list-style-type: none"><li>Calles</li><li>Números enteros</li><li>Ciudades</li><li>Números enteros</li><li>Países</li></ul>	
	TeléfonoU	Simple	Números telefónicos	

Relación	Atributos relacionados	Tipo de atributo	Dominio
Descarga	Fecha_descarga	Clave	Fecha
	Puntuación	Simple	Número entero
	Comentario	Multivariado	Texto



# Diseño lógico

## PASO A TABLAS DE LAS ENTIDADES:

- Tienda (NombreT, Empresa\_gestora, Página\_web)
- App (NombreA, CódigoA, Espacio\_memoria, Precio, fecha\_inicio, fecha\_fin, *NombreES\**, *DNI\**)
- Empresa\_servicios (NombreES, País\_impuestos, Año\_creación, EmailES, Página\_webES)
- Empleado (DNI, NombreE, CalleE, NúmeroE, Código\_postalE, EmailE, TeléfonoE)
- Usuario (Número\_cuenta, NombreU, CalleU, NúmeroU, CiudadU, Código\_postalU, PaísU, TeléfonoU)

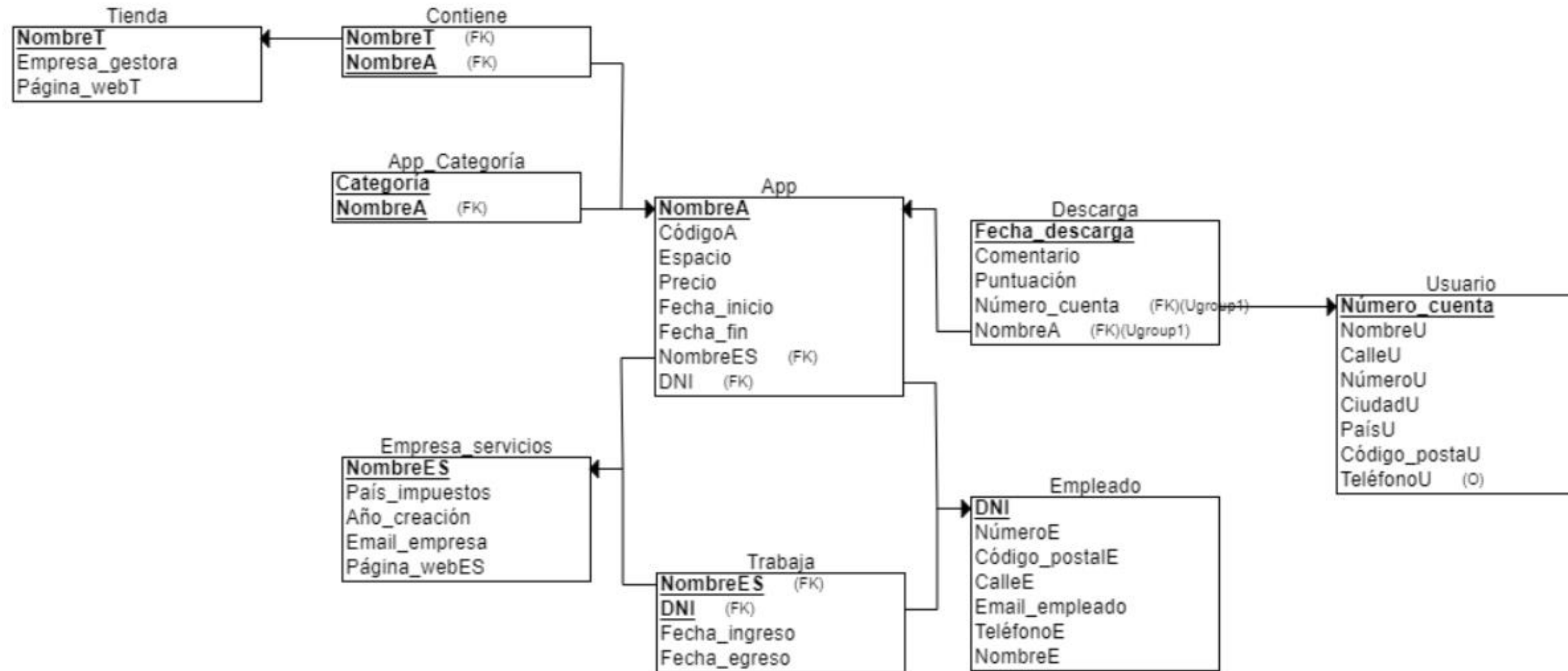
\* Indica foreign key

## PASO A TABLAS DE LAS RELACIONES:

- Contiene (NombreT, NombreA)  
/\*Debido a la relación N:N de las entidades “tienda” y “app”, se genera la tabla “contiene” para almacenar las tiendas en las que las distintas apps figuran\*/
- Trabaja (NombreES, DNI, fecha\_ingreso, fecha\_egreso)  
  
/\*Debido a la relación N:N de las entidades “empresa\_servicios” y “empleado”, se genera la tabla “trabaja” para almacenar las fechas de ingreso y egreso de los empleados en distintas empresa. Se asume que empleados trabajan solo en una empresa a la vez\*/
- Descarga (NombreA, Número\_cuenta, Fecha\_descarga, Puntuación, Comentario)  
/\*Debido a la relación N:N de las entidades “app” y “usuario”, se genera la tabla “descarga” para almacenar las múltiples apps que son descargadas por distintos usuarios\*/
- App\_categoria (categoria, nombreA)  
/\*Debido a que apps pueden tener múltiples categorías, se genera la tabla “app\_categoria” para almacenar las apps y sus categorías correspondientes\*/



# Diseño lógico



# Implementación SQL

**Se adjunta los siguientes archivos para la 4ta parte del proyecto:**



- Archivo SQL que cuenta con las secciones:
  - Creación de base de datos y tablas
  - Carga de datos manuales usando INSERT INTO
  - 15 consultas sobre los datos en la base de datos creada
- Ficheros con datos cargados a la base de datos usando “Tabla Data Import Wizard”

A continuación se detallan las 15 consultas creadas para el proyecto

# Implementación SQL

-- Consulta #1: Busca identificar cuantas descargas se realizaron por mes y ordenar los resultados por orden descendiente

```
select year(fecha_descarga), month(fecha_descarga), count(*)  
from joseobrien.descarga  
group by year(fecha_descarga), month(fecha_descarga)  
order by count(*) desc;
```

Result Grid    Filter Rows: <input type="text"/>   Export: 			
	year(fecha_descarga)	month(fecha_descarga)	count(*)
▶	2020	8	4
	2020	10	3
	2020	3	3
	2019	10	3
	2020	4	3
	2019	7	2
	2020	5	2
	2019	11	2
	2020	6	2
	2020	7	2
	2019	5	1
	2019	6	1
	2020	1	1
	2020	2	1
	2020	11	1
	2019	9	1
	2020	12	1

# Implementación SQL

-- Consulta #2: Busca identificar el país de los usuarios que más aplicaciones se han descargado y ordenar los resultados en orden descendiente

```
select paisU as pais, count(paisU) as cantidadDescargas
from joseobrien.usuario as u inner join joseobrien.descarga as d on u.numero_cuenta=d.idUsuario
group by país
order by count(paisU) desc;
```

	pais	cantidadDescargas
▶	Peru	11
	España	7
	USA	5
	Chile	4
	Brasil	4
	Argentina	2

-- Consulta #3: Busca identificar la puntuación promedio de todas las apps y ordenar los resultados por orden descendiente

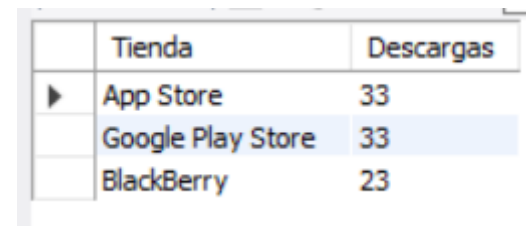
```
select idNombreApp as App, avg (puntuacion) as Puntuación
from joseobrien.descarga
group by idNombreApp
order by avg (puntuacion) desc;
```

	App	Puntuacion
▶	Splitwise	3.7500
	Disney+	3.2000
	LinkedIn	3.0000
	Moodle	3.0000
	BlockFi	2.8571
	YouTube	2.8000
	Duolingo	1.0000
	MyFitnessPal	1.0000
	Rextie	1.0000
	Sunat	1.0000

# Implementación SQL

-- Consulta #4: Busca identificar las tiendas de apps que tienen apps con más de 10 descargas y ordenar los resultados por orden descendiente

```
select idTienda as Tienda, count(idTienda) as Descargas
from joseobrien.contiene as c inner join joseobrien.descarga as d on c.idApp=d.idNombreApp
group by idTienda having count(idTienda) > 10
order by count(idTienda) desc;
```

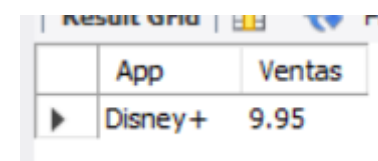


	Tienda	Descargas
▶	App Store	33
	Google Play Store	33
	BlackBerry	23

-- Consulta #5: Busca identificar el app con mayor nivel de ventas (número de descargas multiplicado por precio unitario)

```
create view ventasPorApp as select idNombreApp as App, (count(idNombreApp) * precio) Ventas
from joseobrien.descarga as d inner join joseobrien.app as a on d.idNombreApp=a.nombreA
group by idNombreApp;

select * from ventasPorApp order by Ventas desc limit 1;
```

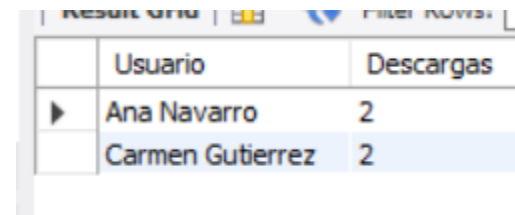


	App	Ventas
▶	Disney+	9.95

# Implementación SQL

-- Consulta #6: Busca identificar los usuarios que descargaron apps en dispositivos que no sean móviles dado que no tienen registrados sus teléfonos, así como el número de descargas correspondientes

```
select nombreU as Usuario, count(idNombreApp) as Descargas
from joseobrien.usuario as u inner join joseobrien.descarga as d on u.numero_cuenta=d.idUsuario
where telefonoU is null
group by nombreU
order by Usuario;
```

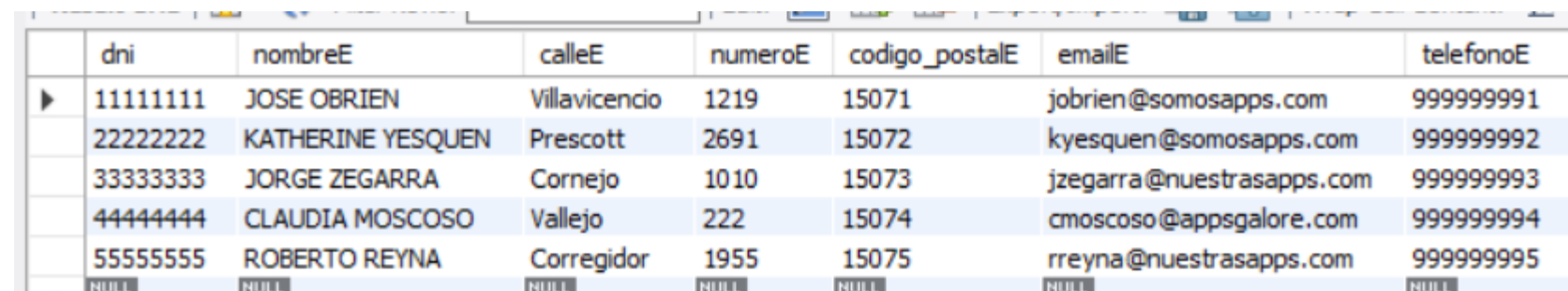


Usuario	Descargas
Ana Navarro	2
Carmen Gutierrez	2

-- Consulta #7: Busca generar un trigger que ponga en mayúscula el nombre y apellido de empleados en la tabla "empleado"

```
drop trigger if exists nuevo_empleado;
create trigger nuevo_empleadobefore insert on joseobrien.empleado
for each rowset
new.nombreE = upper (new.nombreE);
```

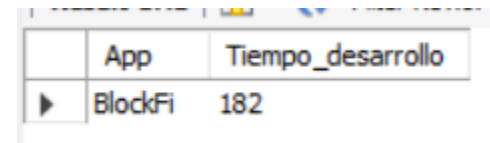
```
INSERT INTO empleado VALUES ('11111111', 'jose obrien', 'Villavicencio', '1219', '15071', 'jobrien@somosapps.com', 999999991);
```



dni	nombreE	calleE	numeroE	codigo_postalE	emailE	telefonoE
11111111	JOSE OBRIEN	Villavicencio	1219	15071	jobrien@somosapps.com	999999991
22222222	KATHERINE YESQUEN	Prescott	2691	15072	kyesquen@somosapps.com	999999992
33333333	JORGE ZEGARRA	Cornejo	1010	15073	jzegarra@nuestrasapps.com	999999993
44444444	CLAUDIA MOSCOSO	Vallejo	222	15074	cmoscoso@appsgalore.com	999999994
55555555	ROBERTO REYNA	Corregidor	1955	15075	rreyna@nuestrasapps.com	999999995
NULL	NULL	NULL	NULL	NULL	NULL	NULL

# Implementación SQL

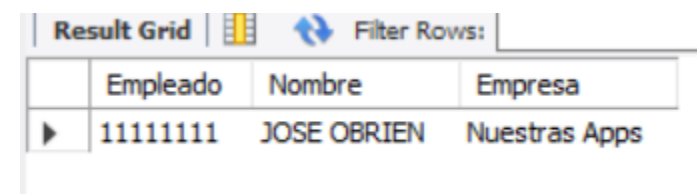
```
-- Consulta #8: Busca identificar el app con mayor tiempo de desarrollo
select nombreA as App, datediff (fecha_fin, fecha_inicio) + 1 as Tiempo_desarrollo
from joseobrien.app
order by Tiempo_desarrollo desc limit 1;
```



	App	Tiempo_desarrollo
▶	BlockFi	182

```
-- Consulta #9: Busca identificar empleados que hayan trabajado para más de 1 una empresa de servicios y el nombre de sus empresas anteriores

select idTrabajador as Empleado, nombreE as Nombre, idEmpleador as Empresa
from joseobrien.trabaja as t inner join joseobrien.empleado as e on t.idTrabajador=e.dni
where fecha_egreso is not null
order by Nombre;
```



	Empleado	Nombre	Empresa
▶	11111111	JOSE OBRIEN	Nuestras Apps



# Implementación SQL

-- Consulta #10: Busca identificar el promedio de apps que cada empleado dirige

```
select count(distinct(nombreA)) / count(distinct(idDirigente)) as promedioAppsDirigidas  
from joseobrien.app  
order by idDirigente;
```

	promedioAppsDirigidas
▶	2.0000

-- Consulta #11: Busca identificar el país con la mayor cantidad de usuarios que descargan apps

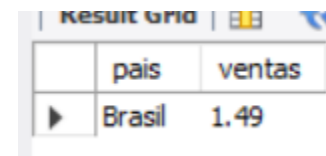
```
select paisU, cantidadDescargas from  
(  
    select count(*) as cantidadDescargas, paisU  
    from joseobrien.usuario  
    group by paisU  
) tmp  
order by cantidadDescargas desc limit 1;
```

	paisU	cantidadDescargas
▶	Peru	5

# Implementación SQL

-- Consulta #12: Busca identificar el país con el menor nivel de ventas de app

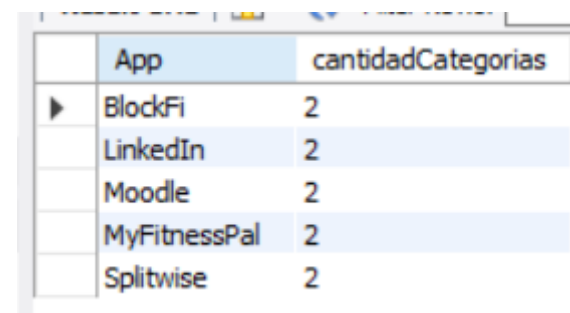
```
sselect paisU as pais, sum(precio) as ventas
from joseobrien.usuario as u inner join joseobrien.descarga as d on u.numero_cuenta=d.idUsuario
inner join joseobrien.app as a on d.idNombreApp=a.nombreA
group by paísU
order by ventas limit 1;
```



	pais	ventas
▶	Brasil	1.49

-- Consulta #13: Busca identificar las apps que tienen más de una categoría asignadas

```
select idNombreAppCategoría as App, count(idNombreAppCategoría) as cantidadCategorías
from joseobrien.app_categoría
group by idNombreAppCategoría
having cantidadCategorías > 1;
```

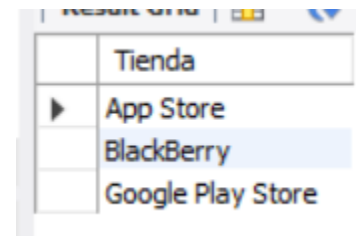


	App	cantidadCategorías
▶	BlockFi	2
	LinkedIn	2
	Moodle	2
	MyFitnessPal	2
	Splitwise	2

# Implementación SQL

-- Consulta #14: Busca identificar las tiendas que cuentan con la app con la mejor puntuación promedio

```
select idTienda as Tienda
from joseobrien.contiene
where idApp = (
    select idNombreApp as App
    from joseobrien.descarga
    group by App
    order by avg (puntuacion) desc limit 1);
```



	Tienda
▶	App Store
	BlackBerry
	Google Play Store

# Implementación SQL

-- Consulta #15: Busca identificar a los usuarios que tienen un promedio más de 1 y menos de 5 y ordenar los resultados por orden descendiente

```
select nombreU as Usuario, avg (puntuacion) as Promedio
from joseobrien.usuario as u inner join joseobrien.descarga as d on u.numero_cuenta=d.idUsuario
group by nombreU
having Promedio < 5 and Promedio > 1
order by Promedio desc;
```

	Usuario	Promedio
►	Isabel Dominguez	4.5000
	Marta Morales	4.0000
	Cristina Molina	3.5000
	Juan Jimenez	3.0000
	Manuel Gonzalez	3.0000
	Carmen Gutierrez	3.0000
	Antonio Garcia	2.5000
	Daniel Moreno	2.0000
	Francisco Sanchez	2.0000
	Laura Ramos	2.0000
	María Alvarez	1.5000
	Javier Ruiz	1.3333