

Planificación de base de datos

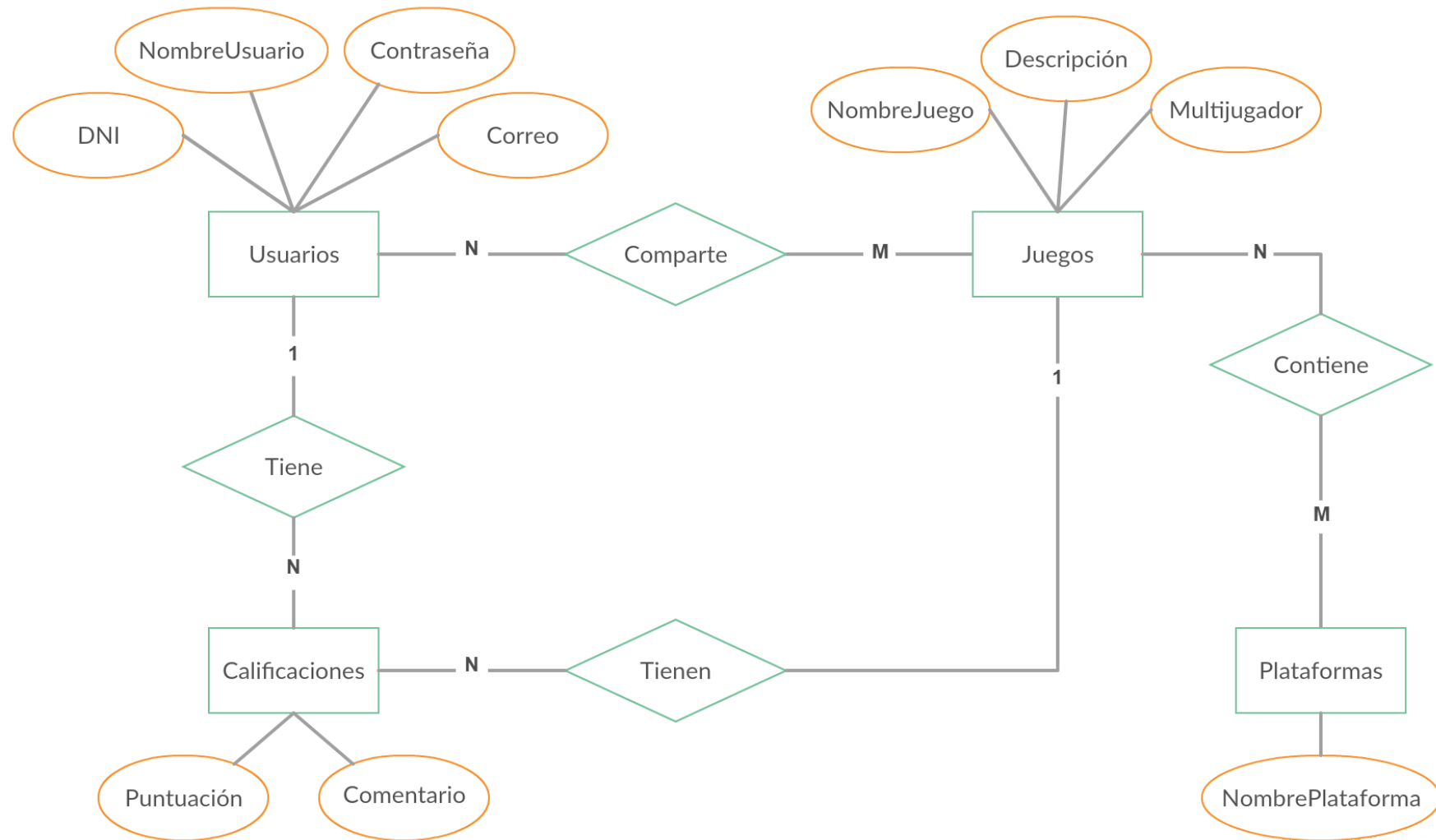
Proyecto: 4Games

Vitaly Sastre  
Toni Rosselló  
Tomeu Llompart  
Miquel Àngel González

## Índex

Mapa completo del Modelo E-R .....	Página 2
Atributos y dominios .....	Página 3
Súper llaves, llaves candidato y llave primaria .....	Página 4
Modelo relacional .....	Página 5
Mapa completo del Modelo E-R optimizado .....	Página 8
Reglas de integridad referencial .....	Página 9
Formas normales sobre base de datos relacionales .....	Página 10
Traducción del modelo relacional al modelo físico .....	Página 11
Atributos y dominios de forma final .....	Página 14
Modelo E-R en forma final .....	Página 15
Código consulta SQL creación de la base de datos .....	Página 16

## Modelo Entidad – Relación



## Atributos y dominios

Usuarios	
<i>Atributos</i>	<i>Dominios</i>
DNI	Ax9 a Zx9
NombreUsuario	Ax45 a Zx45
Contraseña	Ax45 a Zx45
Correo	Ax100 a Zx100

Juegos	
<i>Atributos</i>	<i>Dominios</i>
NombreJuego	Ax45 a Zx45
Descripción	Ax100 a Zx100
Multijugador	Si o No

Calificaciones	
<i>Atributos</i>	<i>Dominios</i>
Puntuación	Entre 0 y 5
Comentario	Ax100 a Zx100

Plataformas	
<i>Atributos</i>	<i>Dominios</i>
NombrePlataforma	Ax45 a Zx45

## Súper llaves

### Definición

La súper llave es un conjunto de uno o más atributos que, juntos, permiten identificar en forma única a una entidad dentro del conjunto de entidades.

Entidad Usuarios:

- DNI
- DNI + NombreUsuario
- DNI + NombreUsuario + Contraseña
- DNI + NombreUsuario + Contraseña + Correo

Entidad Juegos:

- NombreJuego
- NombreJuego + Descripción
- NombreJuego + Descripción + Multijugador

Entidad Calificaciones:

- No existe un conjunto de atributos suficientes para formar una llave única. Por lo tanto esta entidad se define como **Entidad débil** y se le asignará un discriminador llamado **IdCalificación**.

Entidad Plataformas:

- No existe un conjunto de atributos suficientes para formar una llave única. Por lo tanto esta entidad se define como **Entidad débil** y se le asignará un discriminador llamado **IdPlataforma**.

## Llaves candidato

### Definición

Lo que se busca es la súper llave más pequeña posible. Estas súper llaves mínimas se denominan llaves candidato.

Entidad Usuarios:

- DNI.

Entidad Juegos:

- NombreJuego.

Entidad Calificaciones:

- IdCalificación.

Entidad Plataformas:

- IdPlataforma.

## Llaves Primarias

### Definición

Se utilizará el término llave primaria para referirse a la llave candidato que elija el diseñador de la base de datos como la forma principal de identificar a las entidades dentro de un conjunto de éstas.

Entidad Usuarios:

- DNI.

Entidad Juegos:

- NombreJuego.

Entidad Calificaciones:

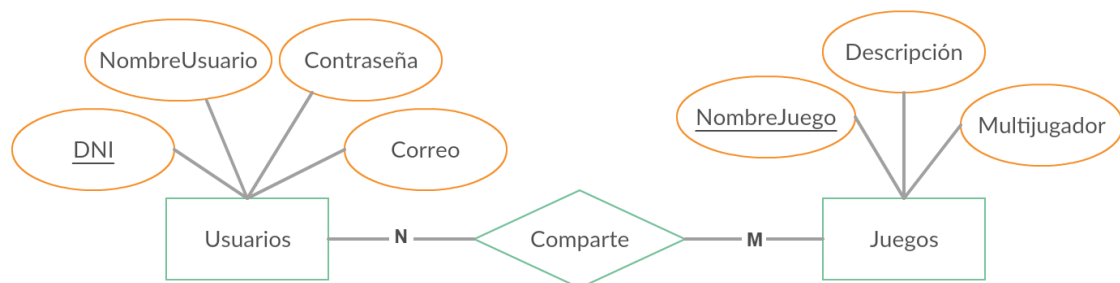
- IdCalificación.

Entidad Plataformas:

- IdPlataforma.

## Modelo Relacional

### Relación entre Usuarios y Juegos (N:M)



**Usuarios** = (DNI, NombreUsuario, Contraseña, Correo)

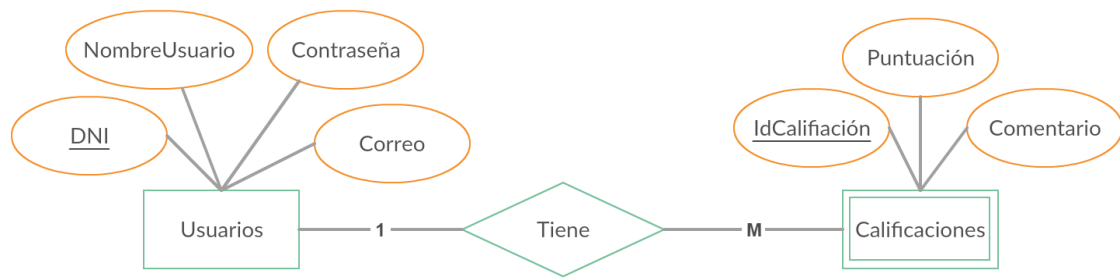
**Juegos** = (NombreJuego, Descripción, Multijugador)

**Comparte** = (DNI, NombreJuego)

===== OPTIMIZACIÓN =====

No tiene optimización ya que las dos llaves primarias se comparten. Por lo tanto se creará una nueva entidad con atributos de las llaves primarias de las otras dos entidades.

## Relación entre Usuarios y Calificaciones (1:M)



**Usuarios** = (DNI, NombreUsuario, Contraseña, Correo)

**Calificaciones** = (IdCalificación, Puntuación)

**Tiene** = (DNI, IdCalificación)

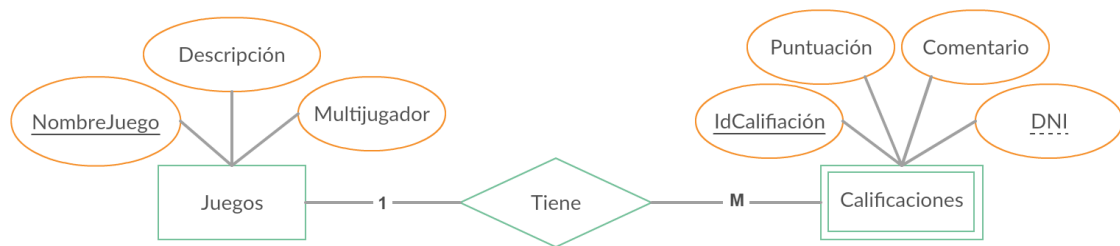
===== OPTIMIZACIÓN =====

**Usuarios** = (DNI, NombreUsuario, Contraseña, Correo)

**Calificaciones** = (IdCalificación, Puntuación, DNI)

~~**Tiene**~~ = (DNI, IdCalificación)

## Relación entre Juegos y Calificaciones (1:M)



**Juegos** = (NombreJuego, Descripción, Multijugador)

**Calificaciones** = (IdCalificación, Puntuación, DNI)

**Tienen** = (NombreJuego, IdCalificación)

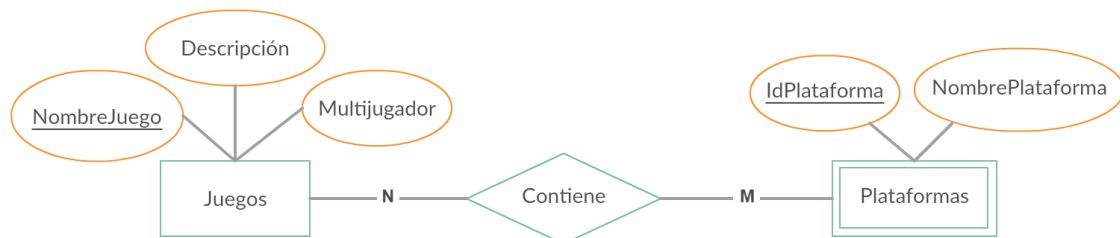
===== OPTIMIZACIÓN =====

**Juegos** = (NombreJuego, Descripción, Multijugador)

**Calificaciones** = (IdCalificación, Puntuación, DNI, NombreJuego)

~~**Tienen** = (NombreJuego, IdCalificación)~~

## Relación entre Juegos y Plataformas (N:M)



**Juegos** = (NombreJuego, Descripción, Multijugador)

**Plataformas** = (IdPlataforma, NombrePlataforma)

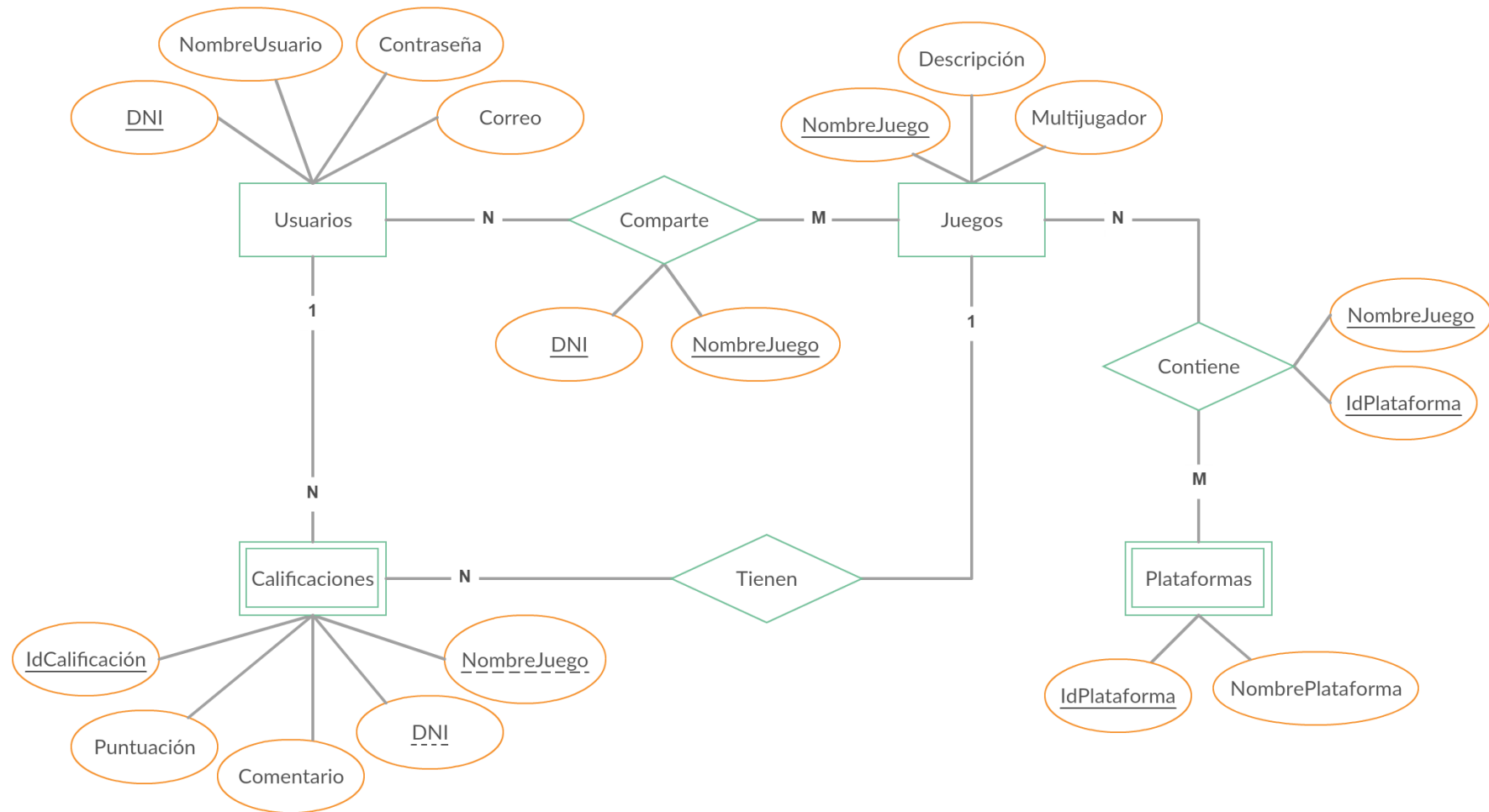
**Contienen** = (NombreJuego, IdPlataforma)

===== OPTIMIZACIÓN =====

No tiene optimización ya que las dos llaves primarias se comparten. Por lo tanto se creará una nueva entidad con atributos de las llaves primarias de las otras dos entidades.



## Modelo Entidad - Relación optimizado



## Reglas de integridad referencial

### Regla de integridad de entidades

Entidad Usuarios:

- Su llave primaria es **DNI** y no puede tener valor nulo y tampoco puede repetirse.

Entidad Juegos:

- Su llave primaria es **NombreJuego** y no puede tener valor nulo y tampoco puede repetirse.

Entidad Calificaciones:

- Su llave primaria es **IdCalificación** y no puede tener valor nulo y tampoco puede repetirse.

Entidad Plataformas:

- Su llave primaria es **IdPlataforma** y no puede tener valor nulo y tampoco puede repetirse.

Relación Comparte:

- Sus llaves primarias son **DNI** y **NombreJuego** y no pueden tener valor nulo y tampoco pueden repetirse.

Relación Contiene:

- Sus llaves primarias son **NombreJuego** e **IdPlataforma** y no pueden tener valor nulo y tampoco pueden repetirse.

### Regla de integridad referencial

Entidad Calificaciones:

- Sus llaves extranjeras son **DNI** y **NombreJuego** y ambas son consistentes.

### Reglas de integridad para llaves extranjeras

Entidad Calificaciones:

- Sus llaves extranjeras son **DNI** y **NombreJuego**.
  - **DNI:**
    - Sí puede ser nulo.
    - Si se intenta eliminar la llave primaria referenciada por una llave extranjera: **No hacer nada**.
    - Si se intenta modificar la llave primaria referenciada por una llave extranjera: **No hacer nada**.
  - **NombreJuego:**
    - No puede ser nulo.
    - Si se intenta eliminar la llave primaria referenciada por una llave extranjera: **Eliminar en cascada**.
    - Si se intenta modificar la llave primaria referenciada por una llave extranjera: **Modificar en cascada**.

## Formas normales sobre base de datos relacionales

### Dependencia funcional o dependencia funcional plena

#### Concepto

Sea R una relación, y X e Y dos atributos de R.

Se dice que X determina funcionalmente a Y o Y depende funcionalmente de X si y solo si: Para cada valor de X hay una y solo una imagen de Y.

Usuarios = (DNI, NombreUsuario, Contraseña, Correo)



Juegos = (NombreJuego, Descripción, Multijugador)



Calificaciones = (IdCalificación, Puntuación, DNI, NombreJuego)



Plataformas = (IdPlataforma, NombrePlataforma)



#### Primera forma normal (1FN)

Todas las entidades sí están en 1FN por que todos los dominios sobre los que están definidos los atributos son atómicos.

#### Segunda forma normal (2FN)

Todas las entidades sí están en 2FN por que cumplen la norma de 1FN y además todo atributo que no es llave primaria depende funcionalmente de toda la llave primaria.

#### Tercera forma normal (3FN)

Todas las entidades sí están en 3FN por que cumple la norma de 2FN y además todo atributo que no es llave primaria no depende funcionalmente de ningún otro atributo que no sea llave primaria.

# Traducción del modelo relacional de datos al modelo físico de datos

## Reglas de traducción de relaciones a tablas

### 1ª Regla

Toda relación del modelo relacional se convierte en una tabla del modelo físico.

### 2ª Regla

Cada atributo de una relación del modelo relacional se convierte en un campo de una tabla del modelo físico.

### 3ª Regla

Cada dominio de los atributos de una relación del modelo relacional, definen el tipo de dato del campo de una tabla.

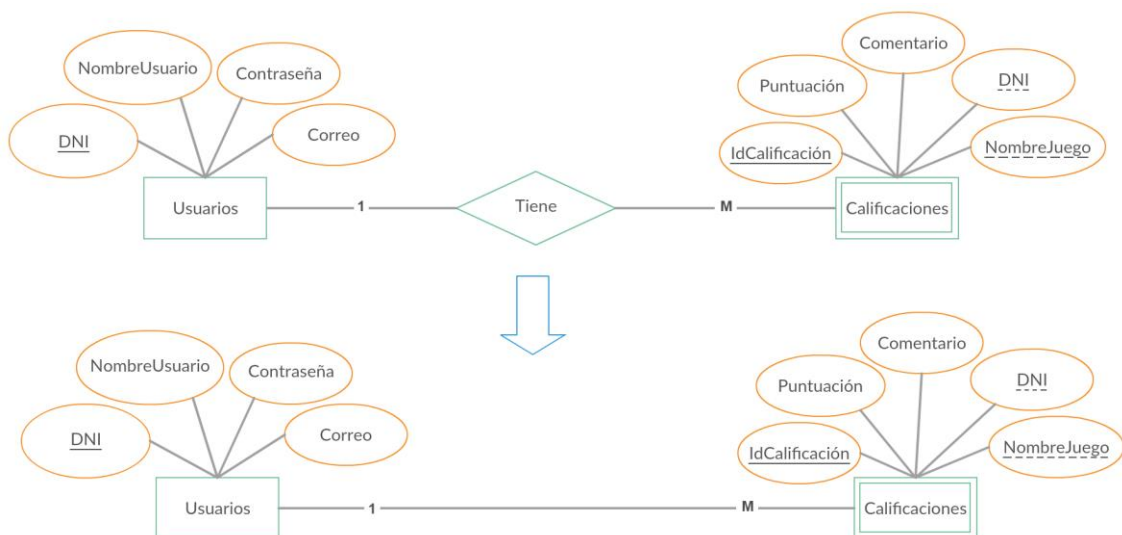
## Reglas de cardinalidad

### 4ª Regla

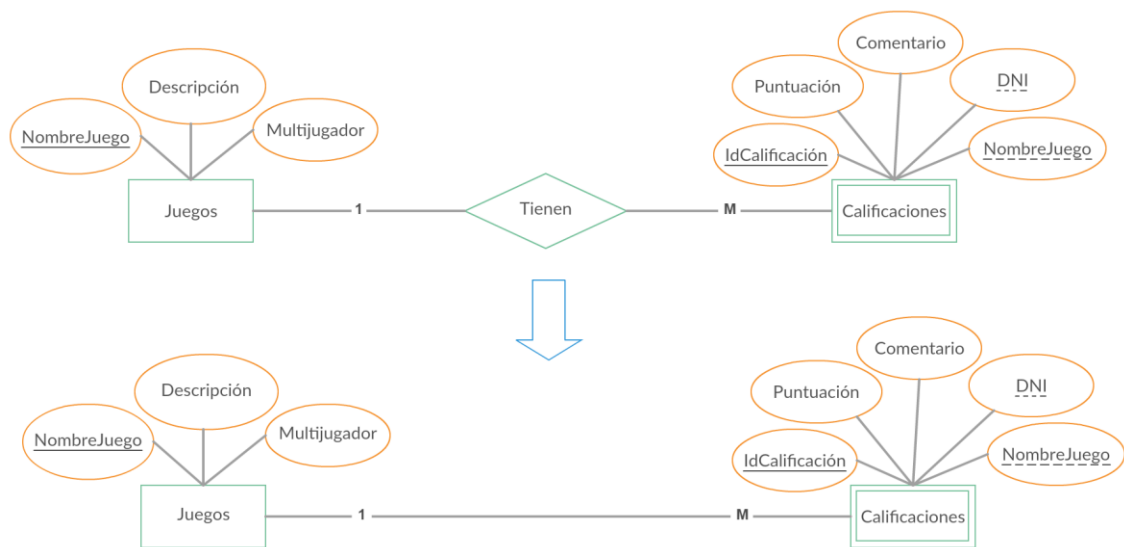
Las relaciones son de tipo 1:1, en este caso no existe ese tipo de relación.

### 5ª Regla

Relación entre Usuarios y Calificaciones (1:N)

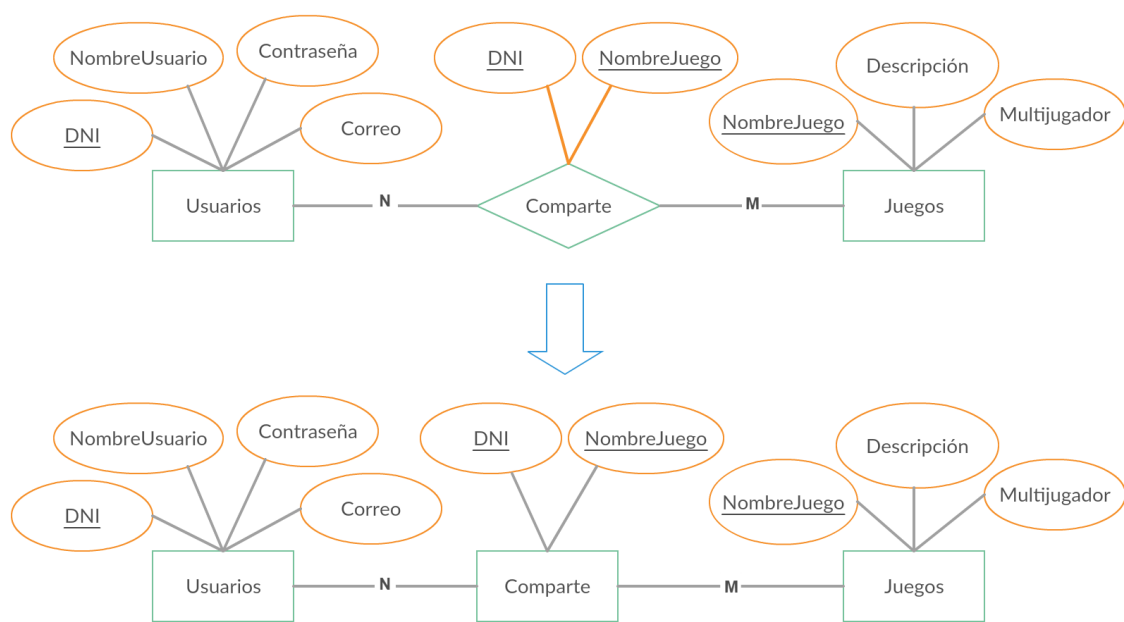


Relación entre Juegos y Calificaciones (1:N)

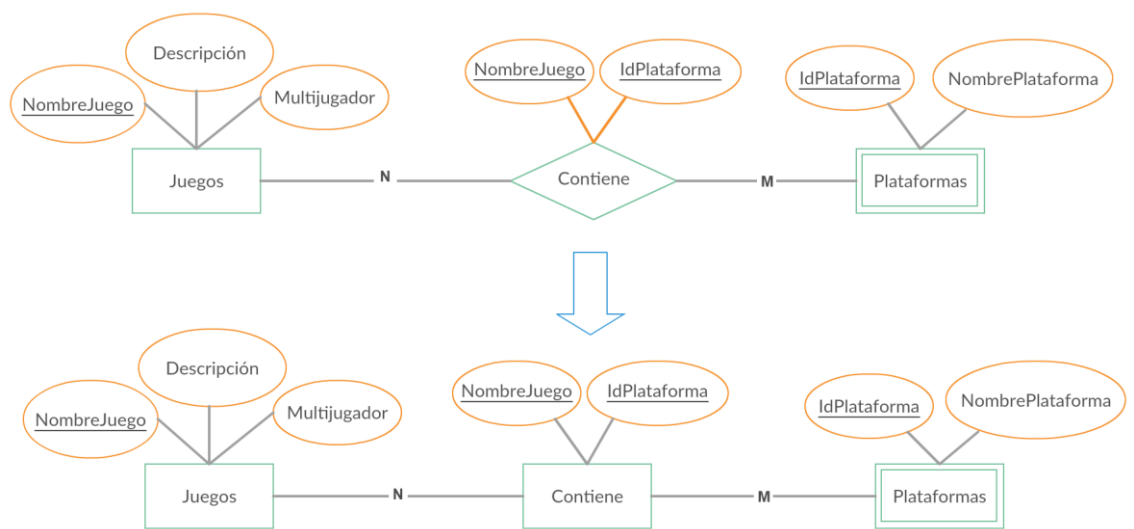


6ª Regla

Relación entre Usuarios y Juegos (N:M)



Relación entre Juegos y Plataformas (N:M)



## Atributos y dominios de forma final

Usuarios	
<i>Atributos</i>	<i>Dominios</i>
<u>DNI</u>	Ax9 a Zx9
NombreUsuario	Ax45 a Zx45
Contraseña	Ax45 a Zx45
Correo	Ax100 a Zx100

Comparte	
<i>Atributos</i>	<i>Dominios</i>
<u>DNI</u>	Ax9 a Zx9
<u>NombreJuego</u>	0x9 a 9x9

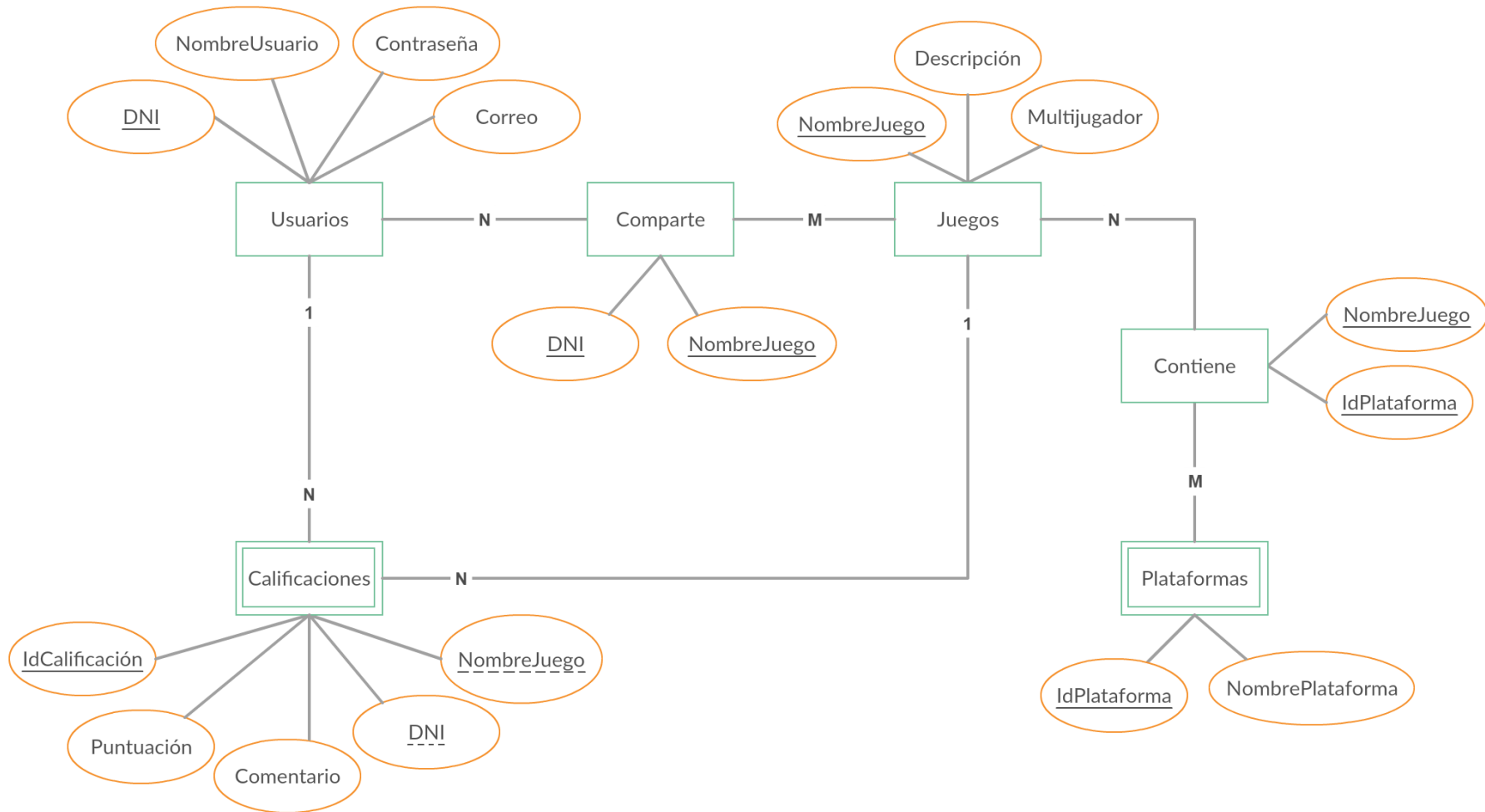
Juegos	
<i>Atributos</i>	<i>Dominios</i>
<u>NombreJuego</u>	Ax45 a Zx45
Descripción	Ax100 a Zx100
Multijugador	Si o No

Contiene	
<i>Atributos</i>	<i>Dominios</i>
<u>NombreJuego</u>	0x9 a 9x9
<u>IdPlataforma</u>	0x9 a 9x9

Plataformas	
<i>Atributos</i>	<i>Dominios</i>
<u>IdPlataforma</u>	0x9 a 9x9
NombrePlataforma	Ax45 a Zx45

Calificaciones	
<i>Atributos</i>	<i>Dominios</i>
<u>IdCalificación</u>	0x9 a 9x9
Puntuación	Entre 0 y 5
Comentario	Ax100 a Zx100
<u>DNI</u>	Ax9 a Zx9
<u>NombreJuego</u>	0x9 a 9x9

## Forma final





## Código consulta SQL creación de la base de datos

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,
ALLOW_INVALID_DATES';

-----

-- Schema Juegos
-----

CREATE SCHEMA IF NOT EXISTS `Juegos` DEFAULT CHARACTER SET utf8;
USE `Juegos`;

-----

-- Table `Juegos`.`Usuarios`
-----

DROP TABLE IF EXISTS `Juegos`.`Usuarios` ;
CREATE TABLE IF NOT EXISTS `Juegos`.`Usuarios` (
  `DniUsuarios` VARCHAR(9) NOT NULL,
  `NombreUsuario` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`DniUsuarios`))
ENGINE = InnoDB;
```

```
-----  
-- Table `Juegos`.`Juegos`  
-----
```

```
DROP TABLE IF EXISTS `Juegos`.`Juegos` ;  
CREATE TABLE IF NOT EXISTS `Juegos`.`Juegos` (  
  `Idjuego` MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `NombreJuego` VARCHAR(45) NOT NULL,  
  `DescripcionJuego` VARCHAR(300) NOT NULL,  
  `Multijugador` TINYINT(1) NOT NULL,  
  PRIMARY KEY (`Idjuego`))  
ENGINE = InnoDB;
```

```
-----  
-- Table `Juegos`.`Calificaciones`  
-----
```

```
DROP TABLE IF EXISTS `Juegos`.`Calificaciones` ;  
CREATE TABLE IF NOT EXISTS `Juegos`.`Calificaciones` (  
  `IdCalificacion` MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `Puntuacion` INT UNSIGNED NOT NULL,  
  `DniUsuario` VARCHAR(9) NOT NULL,  
  `IdJuego` MEDIUMINT UNSIGNED NOT NULL,  
  PRIMARY KEY (`IdCalificacion`),  
  INDEX `dni_idx` (`DniUsuario` ASC),  
  INDEX `idJuegos_idx` (`IdJuego` ASC),  
  CONSTRAINT `dni`  
    FOREIGN KEY (`DniUsuario`)  
    REFERENCES `Juegos`.`Usuarios` (`DniUsuarios`)  
    ON DELETE NO ACTION  
    ON UPDATE CASCADE,  
  CONSTRAINT `idJuegos`  
    FOREIGN KEY (`IdJuego`)  
    REFERENCES `Juegos`.`Juegos` (`Idjuego`)  
    ON DELETE CASCADE
```

```

        ON UPDATE CASCADE)

ENGINE = InnoDB;

-----

-- Table `Juegos`.`Plataformas`
-----

DROP TABLE IF EXISTS `Juegos`.`Plataformas` ;
CREATE TABLE IF NOT EXISTS `Juegos`.`Plataformas` (
  `IdPlataforma` MEDIUMINT UNSIGNED NOT NULL,
  `NombrePlataforma` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`IdPlataforma`))
ENGINE = InnoDB;

-----

-- Table `Juegos`.`Usuarios_has_Juegos`
-----

DROP TABLE IF EXISTS `Juegos`.`Usuarios_has_Juegos` ;
CREATE TABLE IF NOT EXISTS `Juegos`.`Usuarios_has_Juegos` (
  `DniUsuario` VARCHAR(9) NOT NULL,
  `IdJuego` MEDIUMINT UNSIGNED NOT NULL,
  PRIMARY KEY (`DniUsuario`, `IdJuego`),
  INDEX `fk_usuarios_has_juegos_juegos1_idx` (`IdJuego` ASC),
  INDEX `fk_usuarios_has_juegos_usuarios1_idx` (`DniUsuario` ASC),
  CONSTRAINT `fk_usuarios_has_juegos_usuarios1`
    FOREIGN KEY (`DniUsuario`)
      REFERENCES `Juegos`.`Usuarios` (`DniUsuarios`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_usuarios_has_juegos_juegos1`
    FOREIGN KEY (`IdJuego`)
      REFERENCES `Juegos`.`Juegos` (`Idjuego`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```
-----  
-- Table `Juegos`.`Juegos_has_Plataformas`  
-----
```

```
DROP TABLE IF EXISTS `Juegos`.`Juegos_has_Plataformas` ;  
CREATE TABLE IF NOT EXISTS `Juegos`.`Juegos_has_Plataformas` (  
  `IdJuegos` MEDIUMINT UNSIGNED NOT NULL,  
  `IdPlataformas` MEDIUMINT UNSIGNED NOT NULL,  
  PRIMARY KEY (`IdJuegos`, `IdPlataformas`),  
  INDEX `fk_juegos_has_plataformas_plataformas1_idx` (`IdPlataformas`  
ASC),  
  INDEX `fk_juegos_has_plataformas_juegos1_idx` (`IdJuegos` ASC),  
  CONSTRAINT `fk_juegos_has_plataformas_juegos1`  
    FOREIGN KEY (`IdJuegos`)  
    REFERENCES `Juegos`.`Juegos` (`Idjuego`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_juegos_has_plataformas_plataformas1`  
    FOREIGN KEY (`IdPlataformas`)  
    REFERENCES `Juegos`.`Plataformas` (`IdPlataforma`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;  
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```