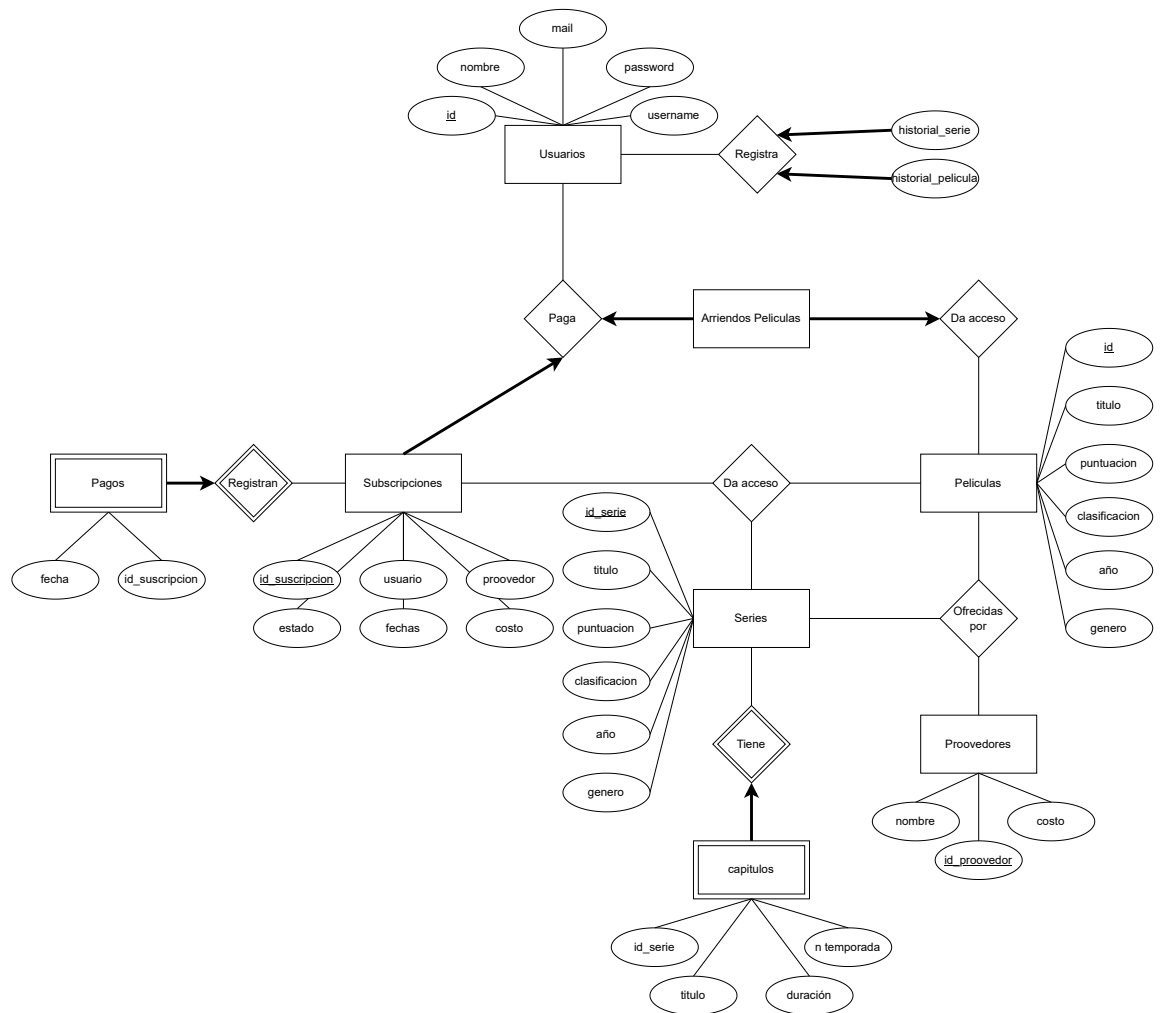


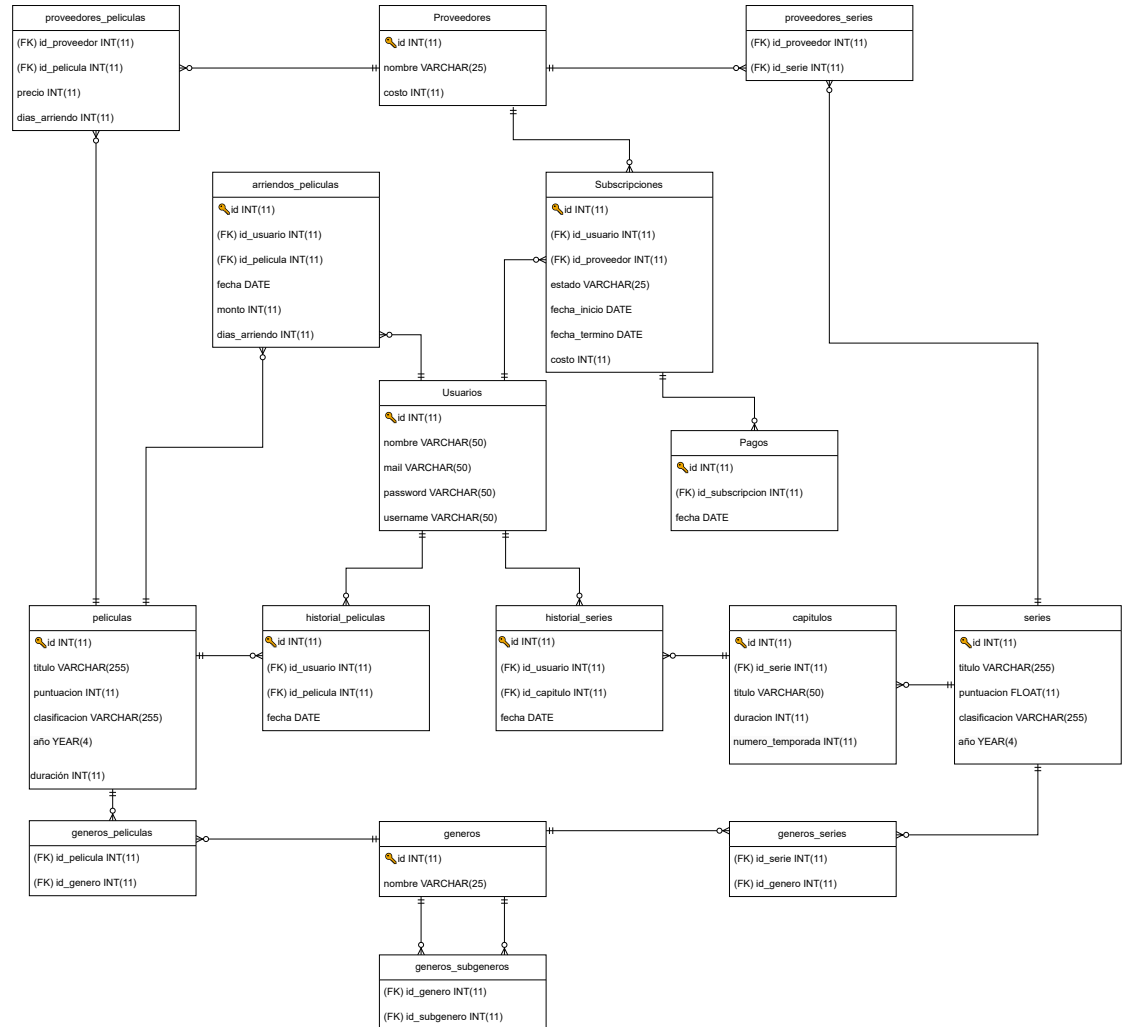
## Entrega 2

Grupo 131  
Leopoldo Farr  
Benjamin Huenchuñir  
20 de octubre de 2023

## 1. Diagrama E/R



## 2. Esquema relacional



## 3. Dependencias funcionales y justificación

### 1. proveedores\_peliculas

Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínima de la siguiente forma:

$id\_proveedor, id\_pelicula \rightarrow precio, dias\_arriendo$

2. Proveedores  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $id \rightarrow nombre, costo$
3. proveedores\_series  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna es independiente, por lo que no existen dependencias que transgredan BCNF.
4. arriendos\_peliculas  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $id \rightarrow id\_usuario, id\_pelicula, fecha, monto, dias\_arriendo$
5. Subscripciones  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $id \rightarrow id\_usuario, id\_proveedor, estado, fecha\_inicio, fecha\_termino, costo$
6. Usuarios  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $id \rightarrow nombre, mail, password, username$
7. Pagos  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $id \rightarrow id\_subscripcion, fecha$
8. peliculas  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $id \rightarrow titulo, puntuacion, clasificacion, año, duracion$

9. historial\_peliculas  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $\text{id} \rightarrow \text{id\_usuario}, \text{id\_pelicula}, \text{fecha}$
10. historial\_series  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $\text{id} \rightarrow \text{id\_usuario}, \text{id\_capitulo}, \text{fecha}$
11. capitulos  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $\text{id} \rightarrow \text{id\_serie}, \text{titulo}, \text{duracion}, \text{numero\_temporada}$
12. series  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $\text{id} \rightarrow \text{titulo}, \text{puntuacion}, \text{clasificacion}, \text{año}$
13. generos\_peliculas  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna es independiente, por lo que no existen dependencias que transgredan BCNF.
14. generos  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna depende unicamente de la llave mínimal de la siguiente forma:  
 $\text{id} \rightarrow \text{nombre}$
15. generos\_series  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna es independiente, por lo que no existen dependencias que transgredan BCNF.

16. `generos.subgeneros`  
Las dependencias de esta tabla están en BCNF dado que no existen repeticiones de datos y cada columna es independiente, por lo que no existen dependencias que transgredan BCNF.

## 4. Consultas

1. **Muestre todas las películas junto con sus proveedores, siempre y cuando el proveedor las ofrezca de manera gratuita**  
`SELECT peliculas.titulo AS pelicula, proveedores.nombre AS proveedor  
FROM peliculas INNER JOIN proveedores_peliculas ON peliculas.id =  
proveedores_peliculas.id_pelicula INNER JOIN proveedores ON proveedo-  
res_peliculas.id_proveedor = proveedores.id WHERE proveedores_peliculas.precio  
IS NULL;`
2. **Dado un numero n ingresado por el usuario, muestre todas las series que tengan al menos n temporadas**  
`SELECT series.titulo AS serie, COUNT(DISTINCT capitulos.numero_temporada)  
AS cantidad_temporadas FROM series INNER JOIN capitulos ON se-  
ries.id = capitulos.idSerie GROUP BY series.id, series.titulo HAVING  
COUNT(DISTINCT capitulos.numero_temporada) >= numero_temporadas;`
3. **Dado un titulo ingresado por el usuario, muestre todas las películas/series con ese título y los proveedores que las ofrecen**  
`SELECT * FROM ( SELECT peliculas.titulo AS titulo, proveedores.nombre  
AS proveedor FROM peliculas INNER JOIN proveedores_peliculas ON  
peliculas.id = proveedores_peliculas.id_pelicula INNER JOIN proveedores  
ON proveedores_peliculas.id_proveedor = proveedores.id UNION SELECT  
series.titulo AS titulo, proveedores.nombre AS proveedor FROM series IN-  
NER JOIN proveedores_series ON series.id = proveedores_series.idSerie  
INNER JOIN proveedores ON proveedores_series.id_proveedor = provee-  
dores.id ) AS U WHERE UPPER(titulo) = UPPER(titulo_pelicula);`
4. **Dado un genero seleccionado por el usuario, muestre todas las películas que pertenezcan a ese género, o que pertenezcan a alguno de sus subgéneros inmediatos**  
`SELECT DISTINCT peliculas.titulo AS titulo, generos.nombre AS genero  
FROM peliculas INNER JOIN generos_peliculas ON peliculas.id = gene-  
ros_peliculas.id_pelicula INNER JOIN generos ON generos_peliculas.id_genero  
= generos.id WHERE generos_peliculas.id_genero = id_genero OR gene-  
ros_peliculas.id_genero IN ( SELECT id_subgenero FROM genero_subgeneros  
WHERE id_genero = id_genero );`

5. **Dado un username ingresado por el usuario, muestre todas las películas a las que tiene acceso dicho usuario**  
SELECT DISTINCT peliculas.titulo AS pelicula FROM peliculas INNER JOIN proveedores\_peliculas ON peliculas.id = proveedores\_peliculas.id\_pelicula INNER JOIN subscripciones ON proveedores\_peliculas.id\_proveedor = subscripciones.id\_proveedor INNER JOIN usuarios ON subscripciones.id\_usuario = usuarios.id LEFT JOIN arriendos\_peliculas ON peliculas.id = arriendos\_peliculas.id\_pelicula AND arriendos\_peliculas.id\_usuario = usuarios.id WHERE UPPER(usuarios.username) = UPPER(username) AND ( (precio IS NULL AND subscripciones.fecha\_termino IS NULL) OR (precio IS NOT NULL AND arriendos\_peliculas.id IS NOT NULL AND (CURRENT\_DATE <= arriendos\_peliculas.fecha + arriendos\_peliculas.dias\_arriendo)) );
6. **Dado un username ingresado por el usuario, muestre todas las series para las cuales el usuario ingresado ha visto mas de un capítulo en el último año**  
SELECT series.titulo AS titulo, COUNT(\*) AS cantidad\_capitulos FROM historial\_series INNER JOIN capitulos ON historial\_series.id\_capitulo = capitulos.id INNER JOIN series ON series.id = capitulos.id\_serie INNER JOIN usuarios ON historial\_series.id\_usuario = usuarios.id WHERE UPPER(usuarios.username) = UPPER(username) AND historial\_series.fecha >= CURRENT\_DATE - INTERVAL '4 year' GROUP BY series.id, series.titulo HAVING COUNT(\*) > 1;
7. **Muestre la suma de dinero gastada por cada usuario en películas no incluidas en planes de suscripción**  
SELECT usuarios.nombre AS usuario, SUM(arriendos\_peliculas.monto) AS total FROM arriendos\_peliculas INNER JOIN usuarios ON arriendos\_peliculas.id\_usuario = usuarios.id GROUP BY usuarios.id, usuarios.nombre;

## 5. Supuestos

1. Para la consulta 6, "Dado un username ingresado por el usuario, muestre todas las series para las cuales el usuario ha visto más de un capítulo durante el último año", el ver dos veces el mismo capítulo también cuenta (issue #72).
2. Para la consulta 6 se consideró un intervalo de tiempo mayor (4 años), ya que no hay datos del último año (issue #175)
3. Clasificación, puntuación y año son de la serie y no varían según capítulo (issue #171)
4. La serie Rick y Morty no tiene género en los datos entregados, se le asignó Comedia (issue #189)

5. Se eliminó el atributo estado de la tabla de subscripciones, ya que solo representa si la subscripción esta activa o no, y eso se puede inferir de la fecha de termino (si existe o es nula). Así se evita una dependencia transitiva.
6. Al igual que el costo de subscripción y el costo de arriendo de una película, los días de arriendo de la película (el atributo disponibilidad) también pueden variar en el tiempo, requiriendo ser guardados para cada arriendo.
7. Una serie que tiene capítulos pertenecientes a n temporadas distintas tiene n temporadas, sin importar el número de temporada de estas (issue #200)