Proyecto Integrado

Realizado por:

Juan Ignacio Velasco Sánchez

Curso:

2016-2017

Diciembre de 2016

Índice

1. Presentación

Este proyecto consta de 3 partes principales:

Base de datos: en la cual se diseña la base de datos con la que va a funcionar la aplicación web, en nuestro caso, una aplicación web sobre valoraciones y comentarios sobre películas.

Aplicación Web: en esta parte se muestra la distribución de los archivos y el código fuente de la aplicación, y el manual para usuario normal y para administradores.

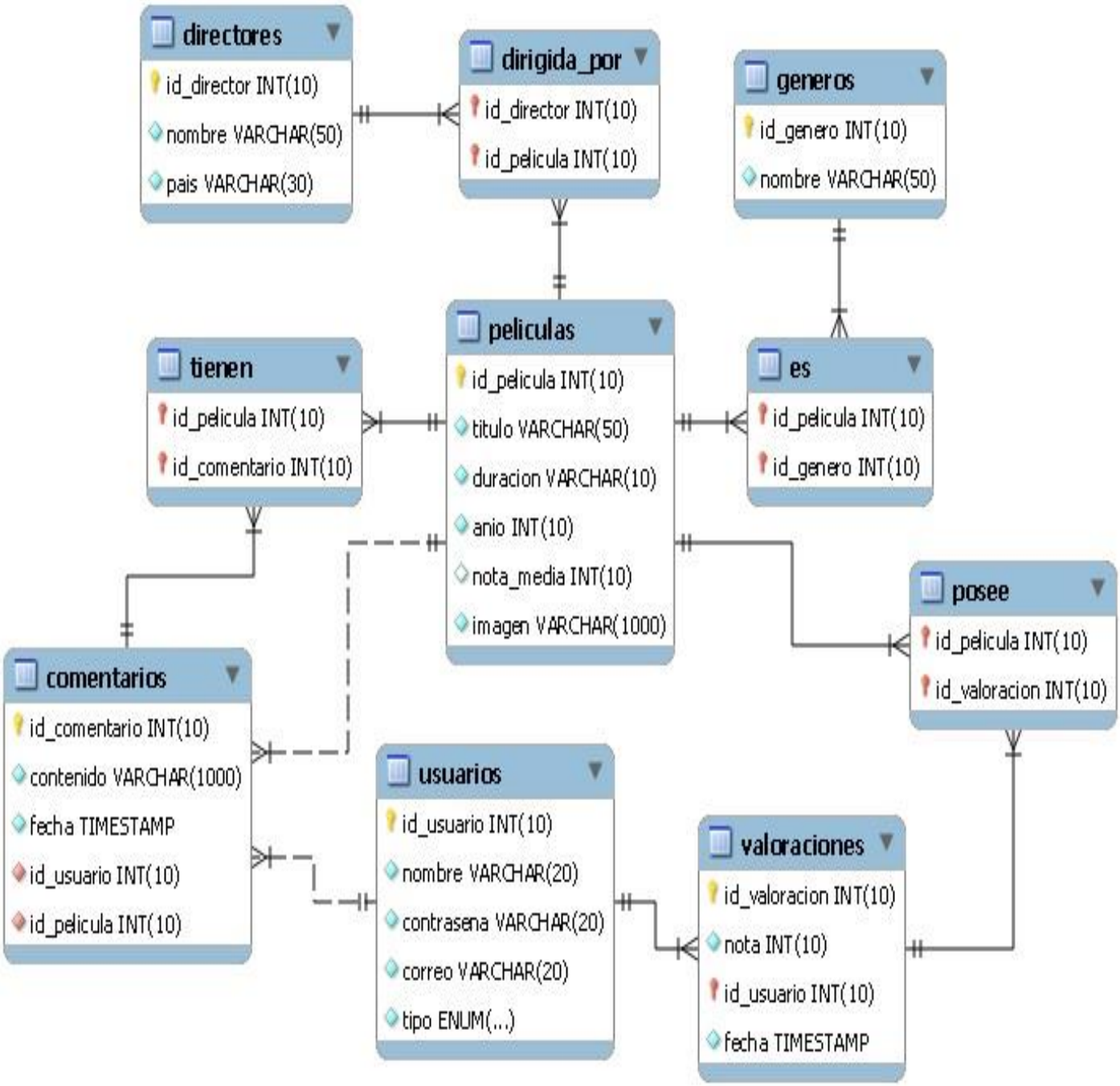
Redes: en esta parte se aborda una situación real, en la cual debemos realizar una instalación de red de una empresa desde los planos del edificio hasta las configuraciones de los dispositivos que intervienen en la red, subredes y otras características cumpliendo unos requisitos de seguridad y sin sobrepasar un presupuesto fijado como límite.

2. Base de datos

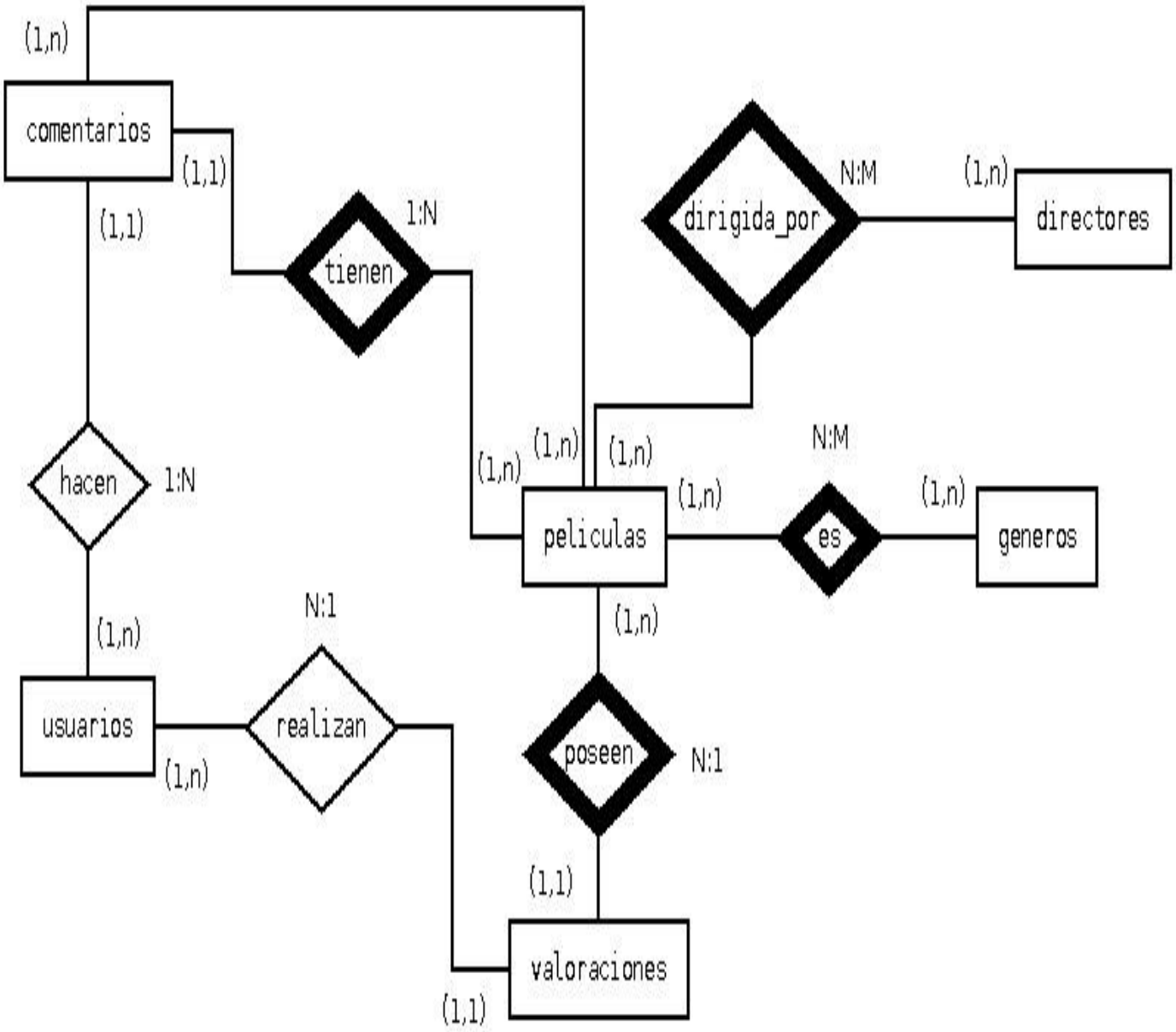
Para diseñar la BD de la aplicación web, se ha tenido en cuenta la mayor casuística posible dentro del contexto de la aplicación web, que se va a llamar ***FilmReview***, y que ha sido diseñada con el objetivo de que los usuarios puedan valorar películas, dejar valoraciones sobre las mismas, y ver en tiempo real notas medias de películas gracias a las valoraciones de otros usuarios. Veamos los modelos para entender la distribución de la información que se va a almacenar en la base de datos:

2.1. Modelo Chen(A3)

2.2. Modelo Martin



2.3. Grafo Relacional



2.4. DLL generado por Visio

-- This SQL DDL script was generated by Visio Enterprise (Release Date: LOCAL BUILD).

-- Driver Used : Visio Enterprise - ODBC Generic Driver Driver.

-- Document : c:\documents and settings\admin\escritorio\dll_final.vsd.

-- Time Created: 27 de November de 2016 18:02.

-- Operation : From Visio Generate Wizard.

-- Connected data source : origen

-- Connected server : ASI

-- Connected database :

-- Create new table posee.

-- posee : Table of posee

-- id_pelicula : id_pelicula identifies posee

-- id_valoracion : id_valoracion partly identifies posee

create table posee (

id_pelicula INTEGER not null constraint poseeid_pelicula_Chk check (id_pelicula between 0 and 9999999999) ,

id_valoracion INTEGER not null constraint poseeid_valoracion_Chk check (id_valoracion between 0 and 9999999999) , constraint posee_PK primary key (id_pelicula, id_valoracion))

-- Create new table valoraciones.

-- valoraciones : Table of valoraciones

-- id_valoracion : id_valoracion identifies valoraciones

-- nota : nota is of valoraciones

-- id_usuario : id_usuario is of valoraciones

-- fecha : fecha is of valoraciones

create table valoraciones (

id_valoracion INTEGER not null constraint valoracionesid_valoracion_Chk check (id_valoracion between 0 and 9999999999) ,

nota INTEGER null constraint valoracionesnota_Chk check (nota between 0 and 9999999999) ,

id_usuario INTEGER null constraint valoracionesid_usuario_Chk check (id_usuario between 0 and 9999999999) ,

fecha TIMESTAMP null, constraint valoraciones_PK primary key (id_valoracion))

```
-- Create new table usuarios.
-- usuarios : Table of usuarios
--    id_usuario : id_usuario identifies usuarios
--    nombre : nombre is of usuarios
--    contrasena : contrasena is of usuarios
--    correo : correo is of usuarios
--    tipo : tipo is of usuarios
create table usuarios (
      id_usuario INTEGER not null constraint
usuariosid_usuario_Chk check (id_usuario between 0 and
9999999999) ,
      nombre VARCHAR(20) null,
      contrasena VARCHAR(20) null,
      correo VARCHAR(20) null,
      tipo VARCHAR(8) null, constraint usuarios_PK primary key
(id_usuario) )

-- Create new table comentarios.
-- comentarios : Table of comentarios
--    id_comentario : id_comentario identifies comentarios
--    contenido : contenido is of comentarios
--    fecha : fecha is of comentarios
--    id_usuario : id_usuario is of comentarios
```

```
--    id_pelicula : id_pelicula is of comentarios

create table comentarios (

      id_comentario INTEGER not null constraint
comentariosid_comentario_Chk check (id_comentario between 0
and 9999999999) ,

      contenido VARCHAR(250) null,

      fecha TIMESTAMP null,

      id_usuario INTEGER null constraint
comentariosid_usuario_Chk check (id_usuario between 0 and
9999999999) ,

      id_pelicula INTEGER null constraint
comentariosid_pelicula_Chk check (id_pelicula between 0 and
9999999999) , constraint comentarios_PK primary key
(id_comentario) )


-- Create new table generos.

-- generos : Table of generos

--    id_genero : id_genero identifies generos

--    nombre : nombre is of generos

create table generos (

      id_genero INTEGER not null constraint
generosid_genero_Chk check (id_genero between 0 and
9999999999) ,

      nombre VARCHAR(50) null, constraint generos_PK primary
key (id_genero) )
```

```
-- Create new table es.
-- es : Table of es
--    id_genero : id_genero identifies es
--    id_pelicula : id_pelicula partly identifies es
create table es (
      id_genero INTEGER not null constraint esid_genero_Chk check (id_genero between 0 and 9999999999) ,
      id_pelicula INTEGER not null constraint esid_pelicula_Chk check (id_pelicula between 0 and 9999999999) , constraint es_PK primary key (id_genero, id_pelicula) )

-- Create new table tienen.
-- tienen : Table of tienen
--    id_comentario : id_comentario identifies tienen
--    id_pelicula : id_pelicula partly identifies tienen
create table tienen (
      id_comentario INTEGER not null constraint tienenid_comentario_Chk check (id_comentario between 0 and 9999999999) ,
      id_pelicula INTEGER not null constraint tienenid_pelicula_Chk check (id_pelicula between 0 and 9999999999) , constraint tienen_PK primary key (id_comentario, id_pelicula) )

-- Create new table directores.
```

```
-- directores : Table of directores
--    id_director : id_director identifies directores
--    nombre : nombre is of directores
--    pais : pais is of directores
create table directores (
      id_director INTEGER not null constraint
directoresid_director_Chk check (id_director between 0 and
9999999999) ,
      nombre VARCHAR(50) null,
      pais VARCHAR(30) null, constraint directores_PK primary
key (id_director) )


-- Create new table dirigida_por.
-- dirigida_por : Table of dirigida_por
--    id_pelicula : id_pelicula partly identifies dirigida_por
--    id_director : id_director identifies dirigida_por
create table dirigida_por (
      id_pelicula INTEGER not null constraint
dirigida_porid_pelicula_Chk check (id_pelicula between 0 and
9999999999) ,
      id_director INTEGER not null constraint
dirigida_porid_director_Chk check (id_director between 0 and
9999999999) , constraint dirigida_por_PK primary key
(id_director, id_pelicula) )
```

```
-- Create new table peliculas.
-- peliculas : Table of peliculas
--    id_pelicula : id_pelicula identifies peliculas
--    titulo : titulo is of peliculas
--    duracion : duracion is of peliculas
--    anio : anio is of peliculas
--    nota_media : nota_media is of peliculas
--    imagen : imagen is of peliculas
--    id_director : id_director is of peliculas
create table peliculas (
      id_pelicula INTEGER not null constraint peliculasid_pelicula_Chk check (id_pelicula between 0 and 9999999999) ,
      titulo VARCHAR(50) not null,
      duracion VARCHAR(10) not null,
      anio INTEGER not null constraint peliculasanio_Chk check (anio between 0 and 9999999999) ,
      nota_media INTEGER null constraint peliculasnota_media_Chk check (nota_media between 0 and 9999999999) ,
      imagen VARCHAR(250) not null,
      id_director INTEGER null constraint peliculasid_director_Chk check (id_director between 0 and 9999999999) , constraint peliculas_PK primary key (id_pelicula) )
```

```
-- Add foreign key constraints to table posee.
alter table posee
	add constraint peliculas_posee_FK1 foreign key (
		id_pelicula)
	 references peliculas (
		id_pelicula)

alter table posee
	add constraint valoraciones_posee_FK1 foreign key (
		id_valoracion)
	 references valoraciones (
		id_valoracion)

-- Add foreign key constraints to table valoraciones.
alter table valoraciones
	add constraint usuarios_valoraciones_FK1 foreign key (
		id_usuario)
	 references usuarios (
		id_usuario)

-- Add foreign key constraints to table comentarios.
alter table comentarios
	add constraint peliculas_comentarios_FK1 foreign key (
```

```
        id_pelicula)
    references peliculas (
        id_pelicula)

alter table comentarios
    add constraint usuarios_comentarios_FK1 foreign key (
        id_usuario)
    references usuarios (
        id_usuario)

-- Add foreign key constraints to table es.
alter table es
    add constraint generos_es_FK1 foreign key (
        id_genero)
    references generos (
        id_genero)

alter table es
    add constraint peliculas_es_FK1 foreign key (
        id_pelicula)
    references peliculas (
        id_pelicula)
```

```
-- Add foreign key constraints to table tienen.
alter table tienen
	add constraint peliculas_tienen_FK1 foreign key (
		id_pelicula)
	 references peliculas (
		id_pelicula)

alter table tienen
	add constraint comentarios_tienen_FK1 foreign key (
		id_comentario)
	 references comentarios (
		id_comentario)

-- Add foreign key constraints to table dirigida_por.
alter table dirigida_por
	add constraint directores_dirigida_por_FK1 foreign key (
		id_director)
	 references directores (
		id_director)

alter table dirigida_por
	add constraint peliculas_dirigida_por_FK1 foreign key (
		id_pelicula)
```

```
        references peliculas (
            id_pelicula)


-- Add foreign key constraints to table peliculas.
alter table peliculas
        add constraint dirigida_por_peliculas_FK1 foreign key (
            id_director,
            id_pelicula)
         references dirigida_por (
            id_director,
            id_pelicula)


-- This is the end of the Visio Enterprise generated SQL DDL script.
```

En resumen, se han contemplado casuísticas tales como por ejemplo que un usuario pueda dar una única valoración o nota por película, para que no sean alterables las notas medias por un mismo usuario, y que puedan comentar una película tantas veces como se desee. Los usuarios están clasificados según el campo “tipo” de la tabla usuarios, donde pueden ser usuario “estándar” o “admin”.

3. Aplicación Web

En esta parte, veremos la distribución de los archivos y carpetas de la aplicación web, proceso de instalación de la aplicación , librerías, funciones principales, interfaz principal y manual de funcionalidades de los usuarios estándar y los administradores.