Tecnicatura Universitaria en Programación
Base de Datos II

Equipo # 60

Integrantes:

Carabajal, Leandro - 31967
Ontivero, Camila - 31729
Rodriguez, Florencia - 31730

Explicación del Sistema

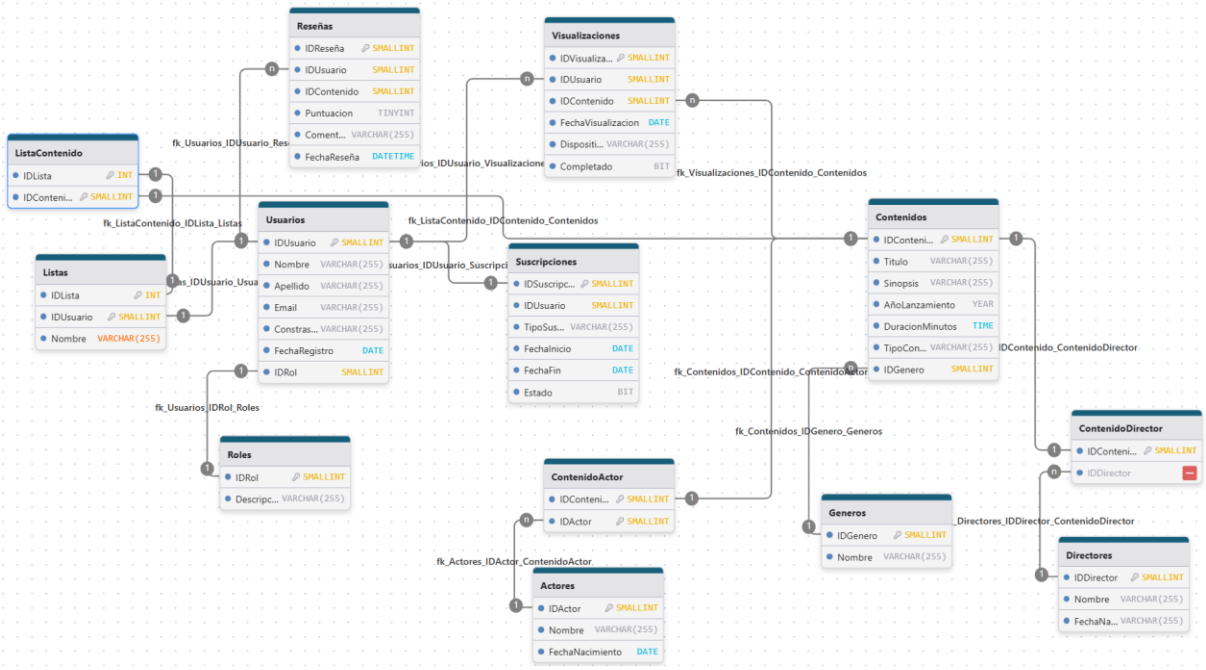
El sistema desarrollado permite gestionar de manera eficiente una plataforma de streaming de películas y series similar a Netflix. Está diseñado para administrar usuarios, contenidos audiovisuales, suscripciones, visualizaciones, reseñas y estadísticas, garantizando una experiencia personalizada y fluida para cada usuario. El sistema incluye login y roles diferenciados (administrador, usuario estándar, premium), habilitando funcionalidades específicas según el tipo de cuenta.

Funcionalidades principales

- **Gestión de usuarios**: Registro y administración de usuarios con datos personales, tipo de suscripción, historial de visualizaciones y preferencias.
- **Catálogo de contenidos**: Administración de películas y series, incluyendo géneros, duración, año de lanzamiento, sinopsis, actores y directores.
- **Sistema de suscripciones**: Los usuarios pueden acceder a distintos planes (gratuito, estándar, premium), que habilitan o restringen funcionalidades como calidad de video, cantidad de dispositivos o acceso a contenido exclusivo.
- **Visualizaciones**: Se registra cada vez que un usuario ve un contenido, incluyendo fecha, duración, dispositivo y si lo completó o no.
- **Reseñas y calificaciones**: Los usuarios pueden dejar reseñas y puntuar los contenidos que han visto. Se calcula el promedio de calificaciones por título.
- **Estadísticas y rankings**: Se generan vistas con los contenidos más vistos, mejor puntuados, y usuarios más activos.
- **Moderación de contenido**: Los administradores pueden agregar, editar o eliminar títulos del catálogo, así como gestionar reseñas inapropiadas.

El sistema está construido sobre una *base de datos relacional*, con procedimientos almacenados, vistas y triggers que automatizan reglas de negocio y mejoran la integridad de los datos.

Diagrama de Entidad Relación



El modelo relacional está compuesto por entidades como Usuarios, Contenidos, Suscripciones, Visualizaciones, Reseñas, Roles, Actores, Directores, Listas y sus respectivas relaciones. Se incluyen relaciones uno a muchos (1:N) como Usuarios → Visualizaciones, y relaciones muchos a muchos (N:N) como Contenidos ↔ Actores, resueltas mediante tablas intermedias (ContenidoActor, ContenidoDirector, ListaContenido).

Objetos de Base de Datos clave en el sistema

Vista 1

- vw_ContenidosMasVistos

Muestra los contenidos con mayor cantidad de visualizaciones.

```
    CREATE VIEW V_ContenidosMasVistos AS
    SELECT C.IDContenido, C.Titulo, COUNT(V.IDVisualizacion) AS TotalVisualizaciones
    FROM Contenidos C
    JOIN Visualizaciones V ON C.IDContenido = V.IDContenido
GROUP BY C.IDContenido, C.Titulo;
```

Vista 2

- vw_ContenidosMejorPuntuados

Calcula el promedio de puntuaciones por contenido.

```
CREATE VIEW vw_ContenidosMejorPuntuados AS
SELECT C.IDContenido, C.Titulo, AVG(R.Puntuacion) AS PromedioPuntuacion
FROM Contenidos C
JOIN Reseñas R ON C.IDContenido = R.IDContenido
GROUP BY C.IDContenido, C.Titulo
```

Vista 3

- Vista 3: vw_UsuariosMasActivos

Identifica a los usuarios con más visualizaciones

```
CREATE VIEW vw_UsuariosMasActivos AS
SELECT U.IDUsuario, U.Nombre, U.Apellido, COUNT(V.IDVisualizacion) AS
TotalVisualizaciones
FROM Usuarios U
JOIN Visualizaciones V ON U.IDUsuario = V.IDUsuario
GROUP BY U.IDUsuario, U.Nombre, U.Apellido
```

Procedimiento Almacenado 1

- sp_RegistrarVisualizacion

Registra una visualización de contenido por parte de un usuario.

```
CREATE PROCEDURE sp_RegistrarVisualizacion
  @IDUsuario INT,
  @IDContenido INT,
  @Dispositivo NVARCHAR(50),
  @Completado BIT
AS
BEGIN
  INSERT INTO Visualizaciones (IDUsuario, IDContenido, FechaVisualizacion, Dispositivo, Completado)
  VALUES (@IDUsuario, @IDContenido, GETDATE(), @Dispositivo, @Completado);
END;
```

Procedimiento Almacenado 2

- sp_RegistrarReseña

Registra una reseña solo si el usuario visualizó el contenido.

```
    CREATE PROCEDURE sp_RegistrarReseña
      @IDUsuario INT,
      @IDContenido INT,
      @Puntuacion INT,
      @Comentario NVARCHAR(500)
    AS
    BEGIN
      IF EXISTS (
        SELECT 1 FROM Visualizaciones
        WHERE IDUsuario = @IDUsuario AND IDContenido = @IDContenido
      )
      BEGIN
        INSERT INTO Reseñas (IDUsuario, IDContenido, Puntuacion, Comentario, FechaReseña)
        VALUES (@IDUsuario, @IDContenido, @Puntuacion, @Comentario, GETDATE());
      END
      ELSE
      BEGIN
        RAISERROR('El usuario no ha visualizado este contenido.', 16, 1);
      END
    END;
```

Procedimiento Almacenado 3 (que usa la función)

- sp_ReporteConsumoUsuario

Crea un reporte de total de minutos y horas por usuario.

```
CREATE PROCEDURE sp_ReporteConsumoUsuario
   @IDUsuario INT
AS
BEGIN
   DECLARE @TotalMinutos INT;
   DECLARE @NombreCompleto NVARCHAR(100);
   DECLARE @TotalHoras DECIMAL(10,2);
   SELECT @NombreCompleto = Nombre + ' ' + Apellido
   FROM Usuarios
   WHERE IDUsuario = @IDUsuario;

   SET @TotalMinutos = dbo.fn_totalMinutosVistosPorUsuario(@IDUsuario);
   SET @TotalHoras = ROUND(@TotalMinutos * 1.0 / 60, 2);

   SELECT
      @IDUsuario AS IDUsuario,
      @NombreCompleto AS Usuario,
      @TotalMinutos AS TotalMinutosVistos,
      @TotalHoras AS TotalHorasVistas;
END;
GO
```

Trigger 1

- trg_ValidarPuntuacionReseña

Evita puntuaciones fuera del rango 1–5.

```
CREATE TRIGGER trg_ValidarPuntuacionReseña
ON Reseñas
FOR INSERT
AS
BEGIN
  IF EXISTS (
    SELECT * FROM inserted WHERE Puntuacion < 1 OR Puntuacion > 5
  )
  BEGIN
    RAISERROR('La puntuación debe estar entre 1 y 5.', 16, 1);
    ROLLBACK;
  END
END;
```

Trigger 2

- trg_ActualizarEstadoSuscripcion

Actualiza automáticamente el campo Estado de la tabla Suscripciones cuando se inserta o actualiza una suscripción cuya FechaFin ya venció.

```
CREATE TRIGGER trg_ActualizarEstadoSuscripcion
ON Suscripciones
AFTER INSERT, UPDATE
AS
BEGIN
  UPDATE S
  SET Estado = 'Vencida'
  FROM Suscripciones S
  WHERE S.FechaFin < GETDATE() AND S.Estado <> 'Vencida';
END;
```

Trigger 3

- trg_BloquearReseñasDuplicadas

evita que un usuario deje más de una reseña para el mismo contenido.

```
CREATE TRIGGER trg_BloquearReseñasDuplicadas
ON Reseñas
INSTEAD OF INSERT
AS
BEGIN
  IF EXISTS (
    SELECT 1
    FROM Reseñas R
    JOIN inserted I ON R.IDUsuario = I.IDUsuario AND R.IDContenido = I.IDContenido
  )
  BEGIN
    RAISERROR('Ya existe una reseña para este contenido por este usuario.', 16, 1);
    RETURN;
  END

  INSERT INTO Reseñas (IDUsuario, IDContenido, Puntuacion, Comentario, FechaReseña)
  SELECT IDUsuario, IDContenido, Puntuacion, Comentario, FechaReseña
  FROM inserted;
END;
```

Trigger 4

- trg_BloquearEliminacionUsuarioConSuscripcion

Evita eliminar usuarios con suscripciones activas.

```
CREATE TRIGGER trg_BloquearEliminacionUsuarioConSuscripcion
ON Usuarios
INSTEAD OF DELETE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM deleted D
        JOIN Suscripciones S ON D.IDUsuario = S.IDUsuario
        WHERE S.Estado = 'Activa'
    )
    BEGIN
        RAISERROR('No se puede eliminar un usuario con una suscripción activa.', 16, 1);
        RETURN;
    END

    DELETE FROM Usuarios
    WHERE IDUsuario IN (SELECT IDUsuario FROM deleted);
END;
GO
```

Función 1:

- fn_totalMinutosVistosPorUsuario

Devuelve la suma total de minutos vistos por un usuario.

```
CREATE FUNCTION fn_totalMinutosVistosPorUsuario (@IDUsuario INT)
RETURNS INT
AS
BEGIN
  DECLARE @Total INT;
  SELECT @Total = SUM(C.DuracionMinutos)
  FROM Visualizaciones V
  JOIN Contenidos C ON V.IDContenido = C.IDContenido
  WHERE V.IDUsuario = @IDUsuario;
  RETURN ISNULL(@Total, 0);
END;
```

Links a los recursos

Script de creación de base de datos con datos

https://github.com/leandrocarabajal1988/TpFinal_baseDatos_Streaming.git

Video demo del sistema (hasta 25 minutos)

https://youtube.com/usuario/VideoDemo