

Universidad Rafael Landívar

Facultad de Ingeniería

Base de Datos 2

CINE GT

FASE 2

Rodrigo Pereira 1269521

Yaxkem Lol 1136721

Luis Peralta 1231721

Brayan Leal 1089117

Guatemala, 5 de noviembre de 2024

Flujo del Negocio

Flujo de Negocio de CineGT

Gestión de Sesiones y Películas:

1. Crear o actualizar la programación de películas (sesiones) en cada sala.
2. Controlar el estado de cada sesión (activa/inactiva) y evitar traslapes de horarios.

Venta de Boletos:

1. Los clientes pueden comprar asientos, seleccionando entre:
 1. Asignación Automática: el sistema elige asientos libres.
 2. Asignación Manual: el cliente selecciona asientos específicos.
2. Confirmar la transacción de venta y generar un registro en Log_Transaccion.

Cambio de Asientos:

1. El cliente puede cambiar sus asientos para otra sesión o sala disponible.
2. Validar la disponibilidad en la nueva sesión antes de completar el cambio.

Anulación de Transacción:

1. Permitir la cancelación de boletos antes del inicio de la sesión, liberando los asientos.
2. Registrar esta acción en el log de transacciones.

Generación de Reportes:

1. Reportar datos como transacciones realizadas, ocupación de salas, y promedio de asistencia por sesión.

Seguridad y Auditoría:

1. Controlar el acceso a la base de datos con usuarios específicos.
2. Registrar toda acción relevante en Log_Transaccion para seguimiento.

Dominios y restricciones

Tabla Usuario

Campo	Dominio	Restricción
ID_Usuario	INT, clave primaria, auto-incremental.	Llave primaria
Nombre	VARCHAR(100), no nulo.	no puede exceder 100 caracteres.
Rol	BIT, 0 para usuarios estándar y 1 para administradores.	Solo puede ser 0 o 1

Tabla Sala

Campo	Dominio	Restricción
ID_Sala	INT, clave primaria, auto-incremental.	Llave primaria
Capacidad	INT, no nulo, representa el número de asientos de la sala	Valor obligatorio y tiene que ser mayor a cero.

Tabla Clasificacion

Campo	Dominio	Restricción
ID_Clasificacion	INT, clave primaria, auto-incremental.	Llave primaria
Tipo_Clasificacion	VARCHAR(100), no nulo	no puede exceder 100 caracteres

Tabla Película

Campo	Dominio	Restricción
ID_Pelicula	INT, clave primaria, auto-incremental	Llave primaria
Nombre	VARCHAR(100), no nulo	no puede exceder 100 caracteres
Duracion	TIME, no nulo, representa la duración de la película	Valor obligatorio y su duración no puede ser “cero”

Descripcion	VARCHAR(100)	no puede exceder 100 caracteres
ID_Clasificacion	INT, clave foránea	Llave foránea

Tabla Sesión

Campo	Dominio	Restricción
ID_Sesion	INT, clave primaria, auto-incremental	Llave primaria
ID_Sala	INT, clave foránea	Llave foránea
ID_Pelicula	INT, clave foránea	Llave foránea
Fecha_Inicio	DATETIME, no nulo	Fecha_Fin debe ser mayor que Fecha_Inicio
Fecha_Fin	DATETIME, no nulo	
Estado	BIT	Solo valor 0 o 1

Tabla Asiento

Campo	Dominio	Restricción
ID_Asiento	INT, clave primaria, auto-incremental.	Llave primaria
Fila	Varchar(1), representa la fila de una sala de cine.	Debe pertenecer a las letras de la A a la F.
Numero	INT, no nulo, representa el número de asiento en la fila	Debe ser valor mayor que 0.
Restricciones Unificadas entre campos <ol style="list-style-type: none"> Restricción de valor único, donde solo puede existir un ID_Sala, Fila, Numero, evitando venta de un mismo asiento en una sala. Restricción si Fila tiene el valor de 'A', numero puede estar del 1 al 11. Restricción si Fila tiene valor de la B a la F, numero puede estar del 1 al 9 		

Tabla Venta_Asiento

Campo	Dominio	Restricción
ID_Venta	INT, clave primaria, auto-incremental	Llave primaria
ID_Usuario	INT, clave foránea	Llave foránea
ID_Sesion	INT, clave foránea	Llave foránea
Cant_Asientos	INT, no nulo	Debe ser mayor que 0
Monto	Money, no nulo	Debe ser mayor que 0

Tabla Detalle_Asiento

Campo	Dominio	Restricción
ID_Detalle_Asiento	INT, clave primaria, auto-incremental	Llave primaria
ID_Sala_Asiento	INT, clave foránea	Llave foránea
ID_Venta	INT, clave foránea	Llave foránea
Estado	VARCHAR(8)	no puede exceder 8 caracteres, y ser 'Ocupado' y 'Libre'

Tabla Log_Transaccion

Campo	Dominio	Restricción
ID_Log	INT, clave primaria, auto-incremental	Llave primaria
ID_Venta	INT	Solo acepta valores enteros
Fecha	DATETIME	
ID_Usuario	INT	Solo acepta valores enteros
ID_Accion	INT, clave foránea. No nulo.	

Tabla Tipo_Accion

Campo	Dominio	Restricción
ID_Accion	INT, clave primaria.	Llave primaria
Descripcion	Varchar(50), No nulos, para creación de acciones "Modificación", "Eliminación", "Creación".	

Listado de SPs

1. Anular la venta de un asiento sin que haya iniciado la sesion.

```
CREATE PROCEDURE SP_Anular_Venta(  
1      @ID_Venta int,  
2      @ID_Usuario int)  
3  AS  
4  BEGIN  
5      BEGIN TRANSACTION  
6      SET TRANSACTION ISOLATION LEVEL SERIALIZABLE  
7  
8      DECLARE @ID_Sesion int,  
9              @SesionIniciada BIT;  
10  
11     Select @ID_Sesion = ID_Sesion From Venta_Asiento  
12     where ID_Venta = @ID_Venta;  
13  
14     Select @SesionIniciada = CASE WHEN GETDATE() >= Fecha_Inicio  
15 THEN 1 ELSE 0 END  
16     From Sesion  
17     Where ID_Sesion = @ID_Sesion  
18  
19     IF (@SesionIniciada = 1)  
20     BEGIN  
21         RAISERROR('La sesión ya ha iniciado y no se puede  
22 anular', 16, 1);  
23         ROLLBACK  
24         RETURN;  
25     END  
26  
27     UPDATE Detalle_Asiento  
28     SET Estado = 'Libre'  
29     WHERE ID_Venta = @ID_Venta;  
30     UPDATE Venta_Asiento  
31     SET Monto = 0,  
32         Cant_Asientos = 0  
33     Where ID_Venta = @ID_Venta;  
34  
35     INSERT INTO Log_Transaccion(ID_Venta, Fecha, ID_Usuario,  
36 ID_Accion)  
37     VALUES (@ID_Venta, GETDATE(), @ID_Usuario, 3)  
38     COMMIT  
39 END
```

2. Procedimiento que inserta sesiones por medio de un CSV

```
1 CREATE PROCEDURE SP_InsertarSesion_CSV
2     @RutaArchivo NVARCHAR(MAX)
3 AS
4 BEGIN
5     CREATE TABLE #SesionesTemp (
6         ID_Sala INT,
7         ID_Pelicula INT,
8         Fecha_Inicio DATETIME,
9         Estado BIT
10    );
11    BEGIN TRY
12        -- Cargar el archivo CSV
13        DECLARE @SQL NVARCHAR(MAX);
14        SET @SQL = '
15            BULK INSERT #SesionesTemp
16            FROM ''' + @RutaArchivo + '''
17            WITH (
18                FIELDTERMINATOR = ',', -- Delimitador de campo
19                ROWTERMINATOR = ''\\n'', -- Delimitador de fila
20                FIRSTROW = 2           -- Omite el encabezado si
21 existe
22            );
23        ';
24        EXEC sp_executesql @SQL;
25
26        DECLARE @Fecha_Fin DATETIME, @Duracion Time;
27        BEGIN TRANSACTION
28
29        DECLARE @ID_Sala INT, @ID_Pelicula INT, @Fecha_Inicio
30 DATETIME, @Estado BIT;
31
32        DECLARE ses_cursor CURSOR FOR
33        SELECT ID_Sala, ID_Pelicula, Fecha_Inicio, Estado FROM
34 #SesionesTemp;
35
36        OPEN ses_cursor;
37        FETCH NEXT FROM ses_cursor INTO @ID_Sala, @ID_Pelicula,
38 @Fecha_Inicio, @Estado;
39
40        WHILE @@FETCH_STATUS = 0
41        BEGIN
42            SELECT @Duracion = Duracion
43            FROM Pelicula
44            WHERE ID_Pelicula = @ID_Pelicula
45
46            SET @Fecha_Fin = DATEADD(MINUTE,
47 DATEPART(MINUTE, @Duracion), DATEADD(HOUR, DATEPART(HOUR, @Duracion),
48 @Fecha_Inicio))
49            IF EXISTS (
50                SELECT 1 FROM Sesion
51                WHERE ID_Sala = @ID_Sala
```

```

52         AND Fecha_Inicio = @Fecha_Inicio
53         AND Fecha_Fin = @Fecha_Fin
54         AND Estado = 1
55     )
56     BEGIN
57         print 'Error: Sesión duplicada.';
58         ROLLBACK TRANSACTION;
59         PRINT 'Se han revertido todas las
60 inserciones debido a errores.';
61         CLOSE ses_cursor;
62         DEALLOCATE ses_cursor;
63         RETURN;
64     END
65     ELSE
66     BEGIN
67
68         INSERT INTO Sesion (ID_Sala, ID_Pelicula,
69 Fecha_Inicio, Fecha_Fin, Estado)
70         VALUES (@ID_Sala, @ID_Pelicula, @Fecha_Inicio,
71 @Fecha_Fin, @Estado);
72         INSERT INTO
73 Log_Transaccion(ID_Accion, Fecha, ID_Usuario)
74         Values(4, GETDATE(), 1);
75
76     END
77         FETCH NEXT FROM ses_cursor INTO @ID_Sala,
78 @ID_Pelicula, @Fecha_Inicio, @Estado;
79     END
80
81         CLOSE ses_cursor;
82         DEALLOCATE ses_cursor;
83         DROP TABLE #SesionesTemp;
84         COMMIT TRANSACTION;
85         PRINT 'Se han insertado las sesiones válidas.';
86     END TRY
87     BEGIN CATCH
88         DECLARE @ErrorMessage NVARCHAR(4000) =
89 ERROR_MESSAGE();
90         PRINT 'Error inesperado: ' + @ErrorMessage;
91     END CATCH
92
93 END

```


3. Procedimiento para vender asientos

```
1  Create PROCEDURE VentaDeAsientos
2  (
3      @ID_Usuario INT,
4      @ID_Sesion INT,
5      @Cantidad INT,
6      @AsignacionAutomatica BIT,
7      @AsientosManual NVARCHAR(MAX) = NULL
8  )
9  AS
10 BEGIN
11     SET NOCOUNT ON;
12
13     IF NOT EXISTS (SELECT 1 FROM Sesion WHERE ID_Sesion = @ID_Sesion)
14     BEGIN
15         RAISERROR('El ID de sesión especificado no existe.', 16, 1);
16         RETURN;
17     END;
18
19     SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
20     BEGIN TRANSACTION;
21
22     BEGIN TRY
23         DECLARE @VentaID INT;
24         DECLARE @FechaActual DATETIME = GETDATE();
25         DECLARE @AsientoID INT;
26         DECLARE @Contador INT;
27
28         INSERT INTO Venta_Asiento (ID_Usuario, ID_Sesion,
29 Cant_Asientos, Monto, Fecha_Venta)
30         VALUES (@ID_Usuario, @ID_Sesion, @Cantidad, @Cantidad * 10.0,
31 @FechaActual);
32
33         SET @VentaID = SCOPE_IDENTITY();
34
35         -- Asignat automatico
36         IF @AsignacionAutomatica = 1
37         BEGIN
38             SET @Contador = 0;
39
40             DECLARE AsientosCursor CURSOR FOR
41                 SELECT TOP (@Cantidad) ID_Asiento
42                 FROM Asiento
43                 WHERE ID_Asiento NOT IN (SELECT ID_Asiento FROM
44 Detalle_Asiento WHERE Estado = 'Ocupado' AND ID_Venta IN (SELECT
45 ID_Venta FROM Venta_Asiento WHERE ID_Sesion = @ID_Sesion))
46                 AND ID_Sala = (SELECT ID_Sala FROM Sesion WHERE
47 ID_Sesion = @ID_Sesion)
48                 ORDER BY Fila, Numero;
49
50             OPEN AsientosCursor;
51
```

```

52         FETCH NEXT FROM AsientosCursor INTO @AsientoID;
53     WHILE @@FETCH_STATUS = 0 AND @Contador < @Cantidad
54     BEGIN
55         INSERT INTO Detalle_Asiento (ID_Asiento, ID_Venta,
56 Estado)
57         VALUES (@AsientoID, @VentaID, 'Ocupado');
58
59         SET @Contador = @Contador + 1;
60         FETCH NEXT FROM AsientosCursor INTO @AsientoID;
61     END;
62
63     CLOSE AsientosCursor;
64     DEALLOCATE AsientosCursor;
65
66
67     IF @Contador < @Cantidad
68     BEGIN
69         RAISERROR('No hay suficientes asientos disponibles.',
70 16, 1);
71
72         ROLLBACK TRANSACTION;
73         RETURN;
74     END;
75
76     ELSE
77     BEGIN
78         DECLARE @Asiento NVARCHAR(10);
79         DECLARE @Pos INT;
80
81         WHILE CHARINDEX(',', @AsientosManual) > 0
82         BEGIN
83             SET @Pos = CHARINDEX(',', @AsientosManual);
84             SET @Asiento = LTRIM(RTRIM(SUBSTRING(@AsientosManual,
85 1, @Pos - 1)));
86             SET @AsientosManual = SUBSTRING(@AsientosManual, @Pos
87 + 1, LEN(@AsientosManual) - @Pos);
88
89             SELECT TOP 1 @AsientoID = ID_Asiento
90             FROM Asiento
91             WHERE CONCAT(Fila, Numero) = @Asiento
92             AND ID_Sala = (SELECT ID_Sala FROM Sesion WHERE
93 ID_Sesion = @ID_Sesion);
94
95             IF EXISTS (SELECT 1 FROM Detalle_Asiento WHERE
96 ID_Asiento = @AsientoID AND Estado = 'Ocupado' AND ID_Venta IN
97 (SELECT ID_Venta FROM Venta_Asiento WHERE ID_Sesion = @ID_Sesion))
98             BEGIN
99                 RAISERROR('El asiento %s ya está reservado.', 16,
100 1, @Asiento);
101                 ROLLBACK TRANSACTION;
102                 RETURN;
103             END;

```

```

104             INSERT INTO Detalle_Asiento (ID_Asiento, ID_Venta,
105 Estado)
106             VALUES (@AsientoID, @VentaID, 'Ocupado');
107         END;
108
109         SET @Asiento = LTRIM(RTRIM(@AsientosManual));
110         IF @Asiento <> ''
111         BEGIN
112             SELECT TOP 1 @AsientoID = ID_Asiento
113             FROM Asiento
114             WHERE CONCAT(Fila, Numero) = @Asiento
115             AND ID_Sala = (SELECT ID_Sala FROM Sesion WHERE
116 ID_Sesion = @ID_Sesion);
117
118             IF EXISTS (SELECT 1 FROM Detalle_Asiento WHERE
119 ID_Asiento = @AsientoID AND Estado = 'Ocupado' AND ID_Venta IN
120 (SELECT ID_Venta FROM Venta_Asiento WHERE ID_Sesion = @ID_Sesion))
121             BEGIN
122                 RAISERROR('El asiento %s ya está reservado.', 16,
123 1, @Asiento);
124                 ROLLBACK TRANSACTION;
125                 RETURN;
126             END;
127
128             INSERT INTO Detalle_Asiento (ID_Asiento, ID_Venta,
129 Estado)
130             VALUES (@AsientoID, @VentaID, 'Ocupado');
131         END;
132     END;
133
134     INSERT INTO Log_Transaccion (ID_Venta, Fecha, ID_Usuario,
135 ID_Accion)
136     VALUES (@VentaID, GETDATE(), @ID_Usuario, 1);
137
138     COMMIT TRANSACTION;
139     PRINT 'Venta completada exitosamente.';
140 END TRY
141 BEGIN CATCH
142     DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
143     DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
144     DECLARE @ErrorState INT = ERROR_STATE();
145
146     ROLLBACK TRANSACTION;
147     RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
148 END CATCH
END;

```

4. Procedimiento para hacer cambio de asientos

```
1  CREATE PROCEDURE CambioDeAsientos
2  (
3      @ID_Venta INT,
4      @NuevosAsientos NVARCHAR(MAX),
5      @ID_SesionDestino INT
6  )
7  AS
8  BEGIN
9      SET NOCOUNT ON;
10
11     SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
12     BEGIN TRANSACTION;
13     BEGIN TRY
14         DECLARE @FechaInicioSesion DATETIME;
15         DECLARE @Asiento NVARCHAR(10);
16         DECLARE @Fila NVARCHAR(1);
17         DECLARE @Numero INT;
18         DECLARE @AsientosTabla TABLE (Fila NVARCHAR(1), Numero INT);
19
20         SELECT @FechaInicioSesion = Fecha_Inicio
21         FROM Sesion
22         WHERE ID_Sesion = @ID_SesionDestino;
23
24         IF @FechaInicioSesion <= GETDATE()
25         BEGIN
26             RAISERROR('La sesión de destino ya ha comenzado. No se
27 pueden cambiar los asientos.', 16, 1);
28             ROLLBACK TRANSACTION;
29             RETURN;
30         END
31
32         DECLARE @Pos INT = 1;
33         DECLARE @AsientoActual NVARCHAR(10);
34         WHILE CHARINDEX(',', @NuevosAsientos, @Pos) > 0
35         BEGIN
36             SET @AsientoActual = SUBSTRING(@NuevosAsientos, @Pos,
37 CHARINDEX(',', @NuevosAsientos, @Pos) - @Pos);
38             SET @Pos = CHARINDEX(',', @NuevosAsientos, @Pos) + 1;
39
40             SET @Fila = LEFT(@AsientoActual, 1);
41             SET @Numero = CAST(SUBSTRING(@AsientoActual, 2,
42 LEN(@AsientoActual) - 1) AS INT);
43
44             INSERT INTO @AsientosTabla VALUES (@Fila, @Numero);
45         END;
46
47         SET @AsientoActual = SUBSTRING(@NuevosAsientos, @Pos,
48 LEN(@NuevosAsientos) - @Pos + 1);
49         SET @Fila = LEFT(@AsientoActual, 1);
50         SET @Numero = CAST(SUBSTRING(@AsientoActual, 2,
51 LEN(@AsientoActual) - 1) AS INT);
```

```

52     INSERT INTO @AsientosTabla VALUES (@Fila, @Numero);
53     DECLARE AsientosCursor CURSOR FOR
54         SELECT Fila, Numero FROM @AsientosTabla;
55
56     OPEN AsientosCursor;
57     FETCH NEXT FROM AsientosCursor INTO @Fila, @Numero;
58
59     WHILE @@FETCH_STATUS = 0
60     BEGIN
61
62         IF EXISTS (SELECT 1 FROM Detalle_Asiento da
63                     INNER JOIN Venta_Asiento va ON da.ID_Venta =
64 va.ID_Venta
65                     INNER JOIN Asiento a ON a.ID_Asiento =
66 da.ID_Asiento
67                     WHERE va.ID_Sesion = @ID_SesionDestino
68                     AND a.Fila = @Fila AND a.Numero = @Numero)
69             BEGIN
70                 RAISERROR('El asiento %s%d ya está reservado en la
71 sesión destino.', 16, 1, @Fila, @Numero);
72                 ROLLBACK TRANSACTION;
73                 RETURN;
74             END;
75
76         FETCH NEXT FROM AsientosCursor INTO @Fila, @Numero;
77     END;
78
79     CLOSE AsientosCursor;
80     DEALLOCATE AsientosCursor;
81
82     UPDATE Detalle_Asiento
83     SET Estado = 'Libre'
84     WHERE ID_Venta = @ID_Venta;
85
86         update Venta_Asiento
87         SET ID_Sesion = @ID_SesionDestino
88         WHERE ID_Venta = @ID_Venta
89
90     INSERT INTO Detalle_Asiento (ID_Venta, ID_Asiento, Estado)
91     SELECT @ID_Venta, a.ID_Asiento, 'Ocupado'
92     FROM @AsientosTabla at
93     INNER JOIN Asiento a ON a.Fila = at.Fila AND a.Numero =
94 at.Numero
95     WHERE a.ID_Sala = (SELECT ID_Sala FROM Sesion WHERE ID_Sesion
96 = @ID_SesionDestino);
97
98     INSERT INTO Log_Transaccion (ID_Venta, Fecha, ID_Usuario,
99 ID_Accion)
100     VALUES (@ID_Venta, GETDATE(), (SELECT ID_Usuario FROM
101 Venta_Asiento WHERE ID_Venta = @ID_Venta), 2);
102     COMMIT TRANSACTION;
103     PRINT 'Cambio de asientos completado exitosamente.';

```

```

104     END TRY
105     BEGIN CATCH
106         DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
107         DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
108         DECLARE @ErrorState INT = ERROR_STATE();
109
110         ROLLBACK TRANSACTION;
111         RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
112     END CATCH
113 END;
114
115

```

5. Procedimiento para obtener el promedio de asientos ocupados y la cantidad de sesiones

```

1  CREATE PROCEDURE ObtenerPromedioAsientosOcupadosPorSala
2  (
3      @ID_Sala INT
4  )
5  AS
6  BEGIN
7      SET NOCOUNT ON;
8      DECLARE @FechaInicio DATETIME = DATEADD(MONTH, -3, GETDATE());
9
10     SELECT
11         YEAR(s.Fecha_Inicio) AS Año,
12         MONTH(s.Fecha_Inicio) AS Mes,
13         COUNT(s.ID_Sesion) AS Cantidad_Sesiones,
14         COALESCE(AVG(CAST(da.AsientosOcupados AS DECIMAL) /
15 sa.Capacidad * 100), 0) AS Promedio_Ocupacion
16     FROM
17         Sesion s
18     INNER JOIN
19         Sala sa ON s.ID_Sala = sa.ID_Sala
20     LEFT JOIN
21         (SELECT va.ID_Sesion, COUNT(da.ID_Asiento) AS AsientosOcupados
22          FROM Venta_Asiento va
23          INNER JOIN Detalle_Asiento da ON va.ID_Venta = da.ID_Venta
24          WHERE da.Estado = 'Ocupado'
25          GROUP BY va.ID_Sesion) AS da ON s.ID_Sesion = da.ID_Sesion
26     WHERE
27         s.ID_Sala = @ID_Sala
28         AND s.Fecha_Inicio >= @FechaInicio
29     GROUP BY
30         YEAR(s.Fecha_Inicio),
31         MONTH(s.Fecha_Inicio)
32     ORDER BY
33         Año, Mes;
34 END;

```

6. Obtener el top 5 películas mayores con promedio de asientos vendidos

```
1 CREATE PROCEDURE TopPelículasMayorPromedioAsientosVendidos
2 AS
3 BEGIN
4     SET NOCOUNT ON;
5
6     DECLARE @FechaInicio DATETIME = DATEADD(MONTH, -3, GETDATE());
7
8     SELECT TOP 5
9         p.Nombre AS Película,
10        AVG(CAST(va.Cant_Asientos AS DECIMAL)) AS
11 Promedio_Asientos_Vendidos
12 FROM
13     Película p
14 INNER JOIN
15     Sesión s ON p.ID_Película = s.ID_Película
16 INNER JOIN
17     Venta_Asiento va ON s.ID_Sesión = va.ID_Sesión
18 WHERE
19     s.Fecha_Inicio >= @FechaInicio
20 GROUP BY
21     p.Nombre
22 ORDER BY
23     Promedio_Asientos_Vendidos DESC;
24 END;
```

7. Obtener log de transacciones por medio de un rango de fechas

```
1 CREATE PROCEDURE ObtenerLogTransaccionesRango
2 (
3     @FechaInicio DATETIME,
4     @FechaFin DATETIME
5 )
6 AS
7 BEGIN
8     SET NOCOUNT ON;
9     SELECT
10         lt.ID_Log,
11         lt.Fecha AS Fecha_Log,
12         lt.ID_Venta,
13         va.ID_Usuario,
14         u.Nombre AS Nombre_Usuario,
15         va.Monto,
16         va.Cant_Asientos,
17         s.ID_Sesión,
18         s.Fecha_Inicio AS Fecha_Sesión,
19         s.Fecha_Fin AS Fecha_Fin_Sesión,
20         p.Nombre AS Nombre_Película,
21         ta.Descripción AS Tipo_Acción
```

```

22     FROM
23         Log_Transaccion lt
24     INNER JOIN
25         Venta_Asiento va ON lt.ID_Venta = va.ID_Venta
26     INNER JOIN
27         Usuario u ON va.ID_Usuario = u.ID_Usuario
28     INNER JOIN
29         Sesion s ON va.ID_Sesion = s.ID_Sesion
30     INNER JOIN
31         Pelicula p ON s.ID_Pelicula = p.ID_Pelicula
32     INNER JOIN
33         Tipo_Accion ta ON lt.ID_Accion = ta.ID_Accion
34     WHERE
35         lt.Fecha BETWEEN @FechaInicio AND @FechaFin
36     ORDER BY
37         lt.Fecha;
38 END;

```

8. Obtener sesiones y películas en un rango de fechas

```

1 CREATE PROCEDURE ObtenerSesionYPeliculaRango
2 (
3     @FechaInicio DATETIME,
4     @FechaFin DATETIME
5 )
6 AS
7 BEGIN
8     SET NOCOUNT ON;
9
10    SELECT
11        s.ID_Sesion,
12        s.ID_Sala,
13        s.Fecha_Inicio,
14        s.Fecha_Fin,
15        s.Estado,
16        p.ID_Pelicula,
17        p.Nombre AS Nombre_Pelicula,
18        p.Duracion,
19        p.Descripcion,
20        p.ID_Clasificacion
21    FROM
22        Sesion s
23    INNER JOIN
24        Pelicula p ON s.ID_Pelicula = p.ID_Pelicula
25    WHERE
26        s.Fecha_Inicio BETWEEN @FechaInicio AND @FechaFin
27    ORDER BY
28        s.Fecha_Inicio;
29 END;

```


9. Obtener cantidad de asientos ocupados menor a un porcentaje dado en los últimos 3 meses

```
CREATE PROCEDURE SesionesConOcupacionBaja
1 (
2     @Porcentaje DECIMAL(5, 2)
3 )
4 AS
5 BEGIN
6     SET NOCOUNT ON;
7     DECLARE @FechaInicio DATETIME = DATEADD(MONTH, -3, GETDATE());
8
9     SELECT
10         s.ID_Sesion,
11         s.ID_Sala,
12         s.Fecha_Inicio,
13         s.Fecha_Fin,
14         sa.Capacidad AS Capacidad_Sala,
15         COUNT(da.ID_Asiento) AS AsientosOcupados,
16         CAST(COUNT(da.ID_Asiento) AS DECIMAL(5, 2)) / sa.Capacidad *
17 100 AS Porcentaje_Ocupacion
18     FROM
19         Sesion s
20     INNER JOIN
21         Sala sa ON s.ID_Sala = sa.ID_Sala
22     LEFT JOIN
23         Venta_Asiento va ON s.ID_Sesion = va.ID_Sesion
24     LEFT JOIN
25         Detalle_Asiento da ON va.ID_Venta = da.ID_Venta AND da.Estado
26 = 'Ocupado'
27     WHERE
28         s.Fecha_Inicio >= @FechaInicio
29     GROUP BY
30         s.ID_Sesion, s.ID_Sala, s.Fecha_Inicio, s.Fecha_Fin,
31 sa.Capacidad
32     HAVING
33         CAST(COUNT(da.ID_Asiento) AS DECIMAL(5, 2)) / sa.Capacidad *
34 100 < @Porcentaje
35     ORDER BY
36         s.Fecha_Inicio;
37 END;
```

10. Creación de una sesión sin CSV

```
1 CREATE PROCEDURE SP_CrearSesion
2     @ID_Sala INT,
3     @ID_Pelicula INT,
4     @Fecha_Inicio DATETIME
5 AS
6 BEGIN
7     DECLARE @Duracion TIME
8     DECLARE @Fecha_Fin DATETIME
9     BEGIN TRANSACTION
10    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
11    SELECT @Duracion = Duracion
12    FROM Pelicula
13    WHERE ID_Pelicula = @ID_Pelicula
14
15    SET @Fecha_Fin = DATEADD(MINUTE, DATEPART(MINUTE, @Duracion),
16    DATEADD(HOUR, DATEPART(HOUR, @Duracion), @Fecha_Inicio))
17
18    IF EXISTS (
19        SELECT 1 FROM Sesion
20        WHERE ID_Sala = @ID_Sala
21        AND Fecha_Inicio = @Fecha_Inicio
22        AND Fecha_Fin = @Fecha_Fin
23    )
24    BEGIN
25        RAISERROR('La sala ya tiene una sesión programada en el
26 intervalo de tiempo seleccionado.', 16, 1);
27        ROLLBACK TRANSACTION;
28        RETURN;
29    END
30
31    INSERT INTO Sesion (ID_Sala, ID_Pelicula, Fecha_Inicio, Fecha_Fin,
32 Estado)
33    VALUES (@ID_Sala, @ID_Pelicula, @Fecha_Inicio, @Fecha_Fin, 1)
34    INSERT INTO Log_Transaccion(ID_Usuario, Fecha, ID_Accion)
35    VALUES (1, GETDATE(), 4);
36    PRINT 'Sesión creada exitosamente.'
37    COMMIT TRANSACTION;
38 END
39
```

11. Creación de una nueva película.

```
CREATE PROCEDURE SP_CrearPelicula(  
1  @Nombre VARCHAR(100),  
2  @Duracion TIME,  
3  @Descripcion VARCHAR(100),  
4  @ID_Clasificacion INT)  
5 AS  
6 BEGIN  
7     BEGIN TRANSACTION;  
8     SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
9  
10    IF EXISTS (  
11        SELECT *  
12        FROM Pelicula  
13        WHERE Nombre = @Nombre AND ID_Clasificacion =  
14 @ID_Clasificacion  
15    )  
16    BEGIN  
17        PRINT 'Error: Ya existe una película con el mismo nombre y  
18 clasificación.';  
19        ROLLBACK TRANSACTION;  
20        RETURN;  
21    END  
22  
23    INSERT INTO Pelicula (Nombre, Duracion, Descripcion,  
24 ID_Clasificacion)  
25    VALUES (@Nombre, @Duracion, @Descripcion, @ID_Clasificacion);  
26    INSERT INTO Log_Transaccion(ID_Usuario, Fecha, ID_Accion)  
27    VALUES (1, GETDATE(), 6);  
28    PRINT 'Película creada exitosamente.';  
29    COMMIT TRANSACTION;  
END;
```

Funciones

1. Validar la disponibilidad que hay de asientos retorna los asientos que no están disponibles.

```
1 CREATE FUNCTION fn_ValidarDisponibilidadAsientos
2 (
3     @ID_Sesion INT,
4     @ListadoAsientos VARCHAR(MAX)
5 )
6 RETURNS @AsientosNoDisponibles TABLE (
7     Asiento VARCHAR(10)
8 )
9 AS
10 BEGIN
11     INSERT INTO @AsientosNoDisponibles (Asiento)
12     SELECT CONCAT(a.Fila, CAST(a.Numero AS VARCHAR)) AS Asiento
13     FROM Detalle_Asiento AS da
14     INNER JOIN Asiento AS a ON da.ID_Asiento = a.ID_Asiento
15     WHERE da.ID_Venta IN (
16         SELECT ID_Venta FROM Venta_Asiento WHERE ID_Sesion =
17 @ID_Sesion
18     )
19     AND CONCAT(a.Fila, CAST(a.Numero AS VARCHAR)) IN (
20         SELECT value FROM STRING_SPLIT(@ListadoAsientos, ',')
21     )
22     AND da.Estado = 'Ocupado';
23
24     RETURN;
25 END;
```

2. Valida el estado de una sesion si esta activa o inactiva.

```
1 CREATE FUNCTION fn_ValidarEstadoSesion
2 (
3     @ID_Sesion INT
4 )
5 RETURNS VARCHAR(10)
6 AS
7 BEGIN
8     DECLARE @Estado VARCHAR(10);
9     DECLARE @FechaHoraActual DATETIME = GETDATE();
10    DECLARE @FechaHoraInicio DATETIME;
11    DECLARE @FechaHoraFin DATETIME;
12
13    SELECT
14        @FechaHoraInicio = Fecha_Inicio,
15        @FechaHoraFin = Fecha_Fin
16    FROM Sesion
```

```
17     WHERE ID_Sesion = @ID_Sesion;
18
19     IF @FechaHoraInicio IS NULL OR @FechaHoraFin IS NULL
20     BEGIN
21         SET @Estado = 'Sesion no encontrada';
22     END
23     ELSE
24     BEGIN
25         SET @Estado = CASE
26             WHEN @FechaHoraActual <= @FechaHoraInicio
27 THEN 'Activa'
28             ELSE 'Inactiva'
29         END;
30     END;
31
32     RETURN @Estado;
33 END;
34
```

Triggers

1. Validar sesión para que se respete los 15 minutos entre sesiones y que no haya 2 sesiones en una misma sala a la misma hora.

```
1 CREATE TRIGGER TR_Validar_Sesion
2 ON Sesion
3 INSTEAD OF INSERT
4 AS
5 BEGIN
6     DECLARE @ID_Sala int, @Hora_Inicio DATETIME, @Hora_Final
7     DATETIME;
8     Select @ID_Sala = ID_Sala, @Hora_Inicio = Fecha_Inicio,
9     @Hora_Final = Fecha_Fin
10    FROM inserted
11    DECLARE @Fecha_Inicio DATETIME, @Fecha_Fin DATETIME;
12    DECLARE CURSOR_SESSION CURSOR FOR
13    Select Fecha_Inicio, Fecha_Fin From Sesion
14    where @ID_Sala = ID_Sala
15           AND MONTH(@Hora_Inicio) = MONTH(Fecha_Inicio)
16           AND DAY(@Hora_Inicio) = DAY(Fecha_Inicio);
17
18    OPEN CURSOR_SESSION;
19    FETCH NEXT FROM CURSOR_SESSION INTO @Fecha_Inicio, @Fecha_Fin;
20
21    WHILE @@FETCH_STATUS = 0
22    BEGIN
23        IF(@Hora_Inicio BETWEEN @Fecha_Fin AND
24        DATEADD(MINUTE,14,@Fecha_Fin))
25        BEGIN
26            RAISERROR('No se puede programar la sesión, ya
27 que no respeta el intervalo mínimo de 15 minutos.', 16, 1);
28            ROLLBACK TRANSACTION;
29            RETURN;
30        END
31        IF(@Fecha_Inicio BETWEEN @Hora_Final AND
32        DATEADD(MINUTE,14, @Hora_Final))
33        BEGIN
34            RAISERROR('No se puede programar la sesión, ya
35 que no respeta el intervalo mínimo de 15 minutos.', 16, 1);
36            ROLLBACK TRANSACTION;
37            RETURN;
38        END
39        IF(@Hora_Inicio BETWEEN @Fecha_Inicio AND @Fecha_Fin OR
40 @Hora_Final BETWEEN @Fecha_Inicio AND @Fecha_Fin)
41        BEGIN
42            RAISERROR('No se puede programar la sesión ya
43 hay una a esa hora',16,1);
44            ROLLBACK TRANSACTION;
45            RETURN;
46        END
47    END
```

```

48             FETCH NEXT FROM CURSOR_SESSION INTO @Fecha_Inicio,
49 @Fecha_Fin;
            END

            CLOSE CURSOR_SESSION;
            DEALLOCATE CURSOR_SESSION;
            INSERT INTO sesion (ID_Sala, ID_Pelicula, Fecha_Inicio,
Fecha_Fin, estado)
            SELECT ID_Sala, ID_Pelicula, Fecha_Inicio, Fecha_Fin, estado
            FROM inserted;
END

```

2. Insertar en el Log el registro que se insertó una nueva película

```

CREATE TRIGGER TR_InsertarLog_Pelicula
ON Pelicula
1 AFTER INSERT
2 AS
3 BEGIN
4     DECLARE @Nombre varchar(100), @Duracion TIME, @Descripcion
5     varchar(100), @ID_Clasificacion INT;
6
7     Select @Nombre = Nombre, @Duracion = Duracion, @Descripcion =
8     Descripcion, @ID_Clasificacion = ID_Clasificacion
9     from inserted
10    IF @Nombre is null OR @Duracion IS NULL OR @Descripcion IS
11    NULL OR @ID_Clasificacion IS NULL
12    BEGIN
13        RAISERROR ('Ningún campo puede ser nulo', 16, 1);
14        ROLLBACK TRANSACTION;
15    RETURN;
16    END
17
18    INSERT INTO Log_Transaccion(ID_Usuario, Fecha, ID_Accion)
19    values(1,GETDATE(),6);
END

```

3. Insertar en el Log el registro que se insertó una nueva sesión

```

1 CREATE TRIGGER TR_InsertarLog_Sesion
2 ON Sesion
3 AFTER INSERT
4 AS
5 BEGIN
6     DECLARE @ID_Sala int, @ID_Pelicula int, @Fecha_Inicio
7     DATETIME, @Fecha_Fin DATETIME, @Estado BIT;
8
9     Select @ID_Sala = ID_Sala, @ID_Pelicula = ID_Pelicula,
10 @Fecha_Inicio = Fecha_Inicio, @Fecha_Fin = Fecha_Fin, @Estado = Estado
11    From inserted

```

```
12
13     IF @ID_Sala IS NULL OR @ID_Pelicula IS NULL OR @Fecha_Inicio
14 IS NULL OR @Fecha_Fin IS NULL OR @Estado != 1
15     BEGIN
16         RAISERROR ('Ningún campo puede ser nulo', 16, 1);
17         ROLLBACK TRANSACTION;
18     RETURN;
19     END
20
    INSERT INTO Log_Transaccion(ID_Usuario, Fecha, ID_Accion)
    VALUES(1, GETDATE(), 4)
END
```


Manual de usuario

Requisitos del sistema

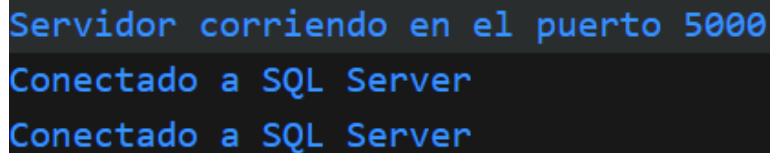
- Node Js
- sql server management studio
- Visual Studio Code (recomendado para el backend)

Preparacion del entorno

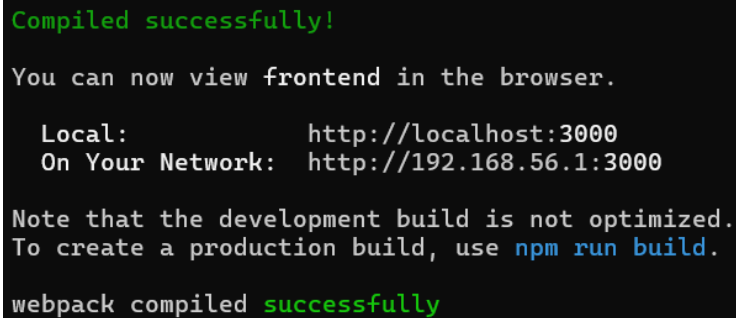
- BACKEND
 - Abrir la terminal en la carpeta backend para las instalaciones
 - `npm install express mssql cors dotenv multer axios`
- FRONTEND
 - abrir la terminal en la carpeta frontend (segunda) para las instalaciones
 - `npm install axios react-router-dom`
- SERVER
 - Editar db.js en ruta `\CINEGT\backend\config`
- Base de Dato
 - Hacer backup a ProyectoBasesII.bak

Ejecución

- BACKEND
 - Abrir y ejecutar index.js (de backend)
 - En ruta `\CINEGT\backend\routes`
- FRONTEND
 - abrir la terminal en la carpeta frontend (segunda)
 - ruta `\CINEGT\frontend\frontend`
 - `npm start`



Servidor corriendo en el puerto 5000
Conectado a SQL Server
Conectado a SQL Server

A terminal window with a dark background and blue text. It displays three lines of status information: 'Servidor corriendo en el puerto 5000', 'Conectado a SQL Server', and 'Conectado a SQL Server'.

Compiled successfully!
You can now view frontend in the browser.

Local: http://localhost:3000
On Your Network: http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use `npm run build`.

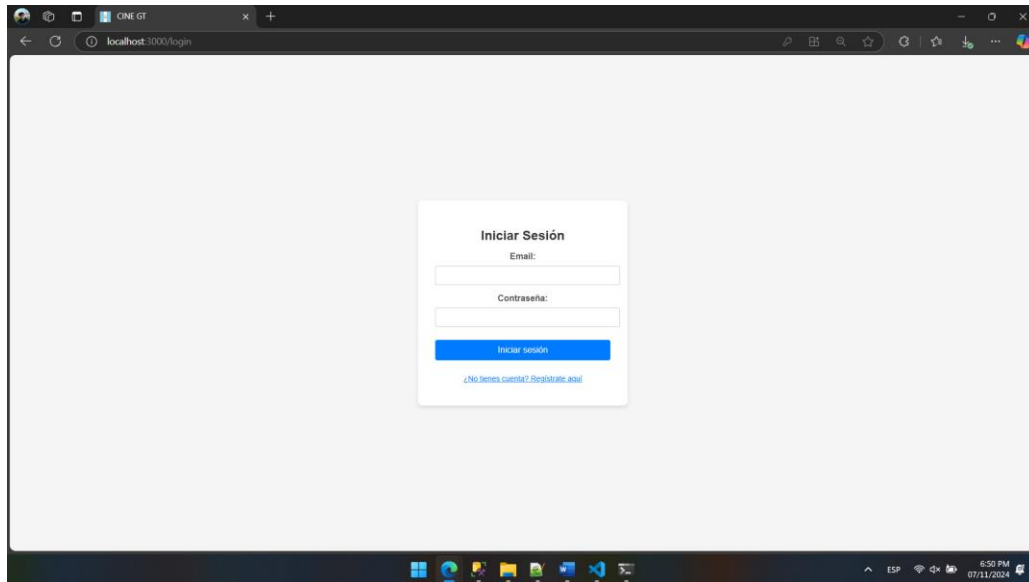
webpack compiled successfully

A terminal window with a dark background and green and white text. It displays a series of messages: 'Compiled successfully!', 'You can now view frontend in the browser.', a blank line, 'Local: http://localhost:3000', 'On Your Network: http://192.168.56.1:3000', a blank line, 'Note that the development build is not optimized.', 'To create a production build, use npm run build.', a blank line, and 'webpack compiled successfully'.

LOGIN

Ingresar:

- Correo
- Contraseña

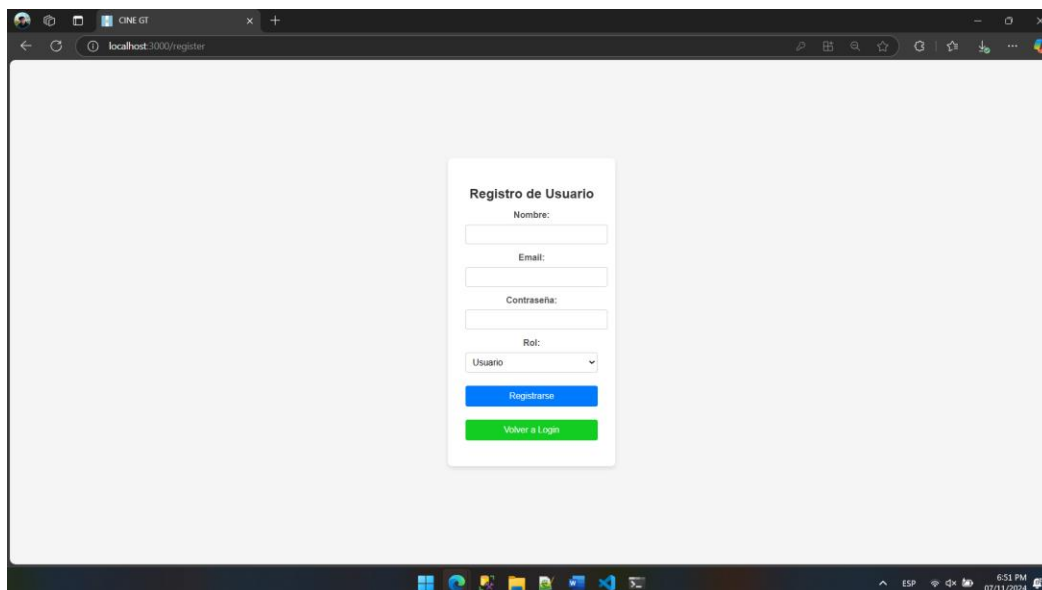


The screenshot shows a web browser window with the address bar displaying 'localhost:3000/login'. The page features a central white card with the title 'Iniciar Sesión'. Inside the card, there are two input fields labeled 'Email:' and 'Contraseña:'. Below these fields is a blue button labeled 'Iniciar sesión'. At the bottom of the card, there is a link that reads '¿No tienes cuenta? Regístrate aquí!'. The browser's taskbar at the bottom shows the time as 6:50 PM on 07/11/2024.

CREAR USUARIO

Ingresar:

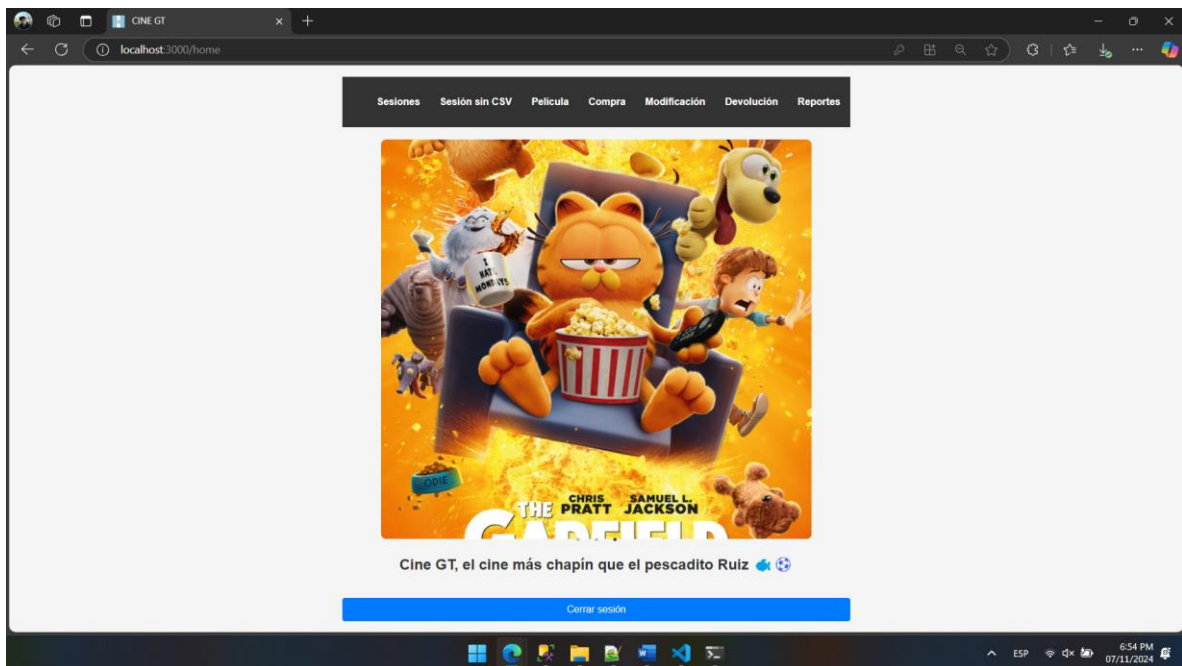
- Nombre
- Correo
- Contraseña
- Seleccionar rol



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/register'. The page features a central white card with the title 'Registro de Usuario'. Inside the card, there are four input fields labeled 'Nombre:', 'Email:', 'Contraseña:', and 'Rol:'. The 'Rol:' field is a dropdown menu with 'Usuario' selected. Below these fields are two buttons: a blue 'Registrarse' button and a green 'Volver a Login' button. The browser's taskbar at the bottom shows the time as 6:51 PM on 07/11/2024.

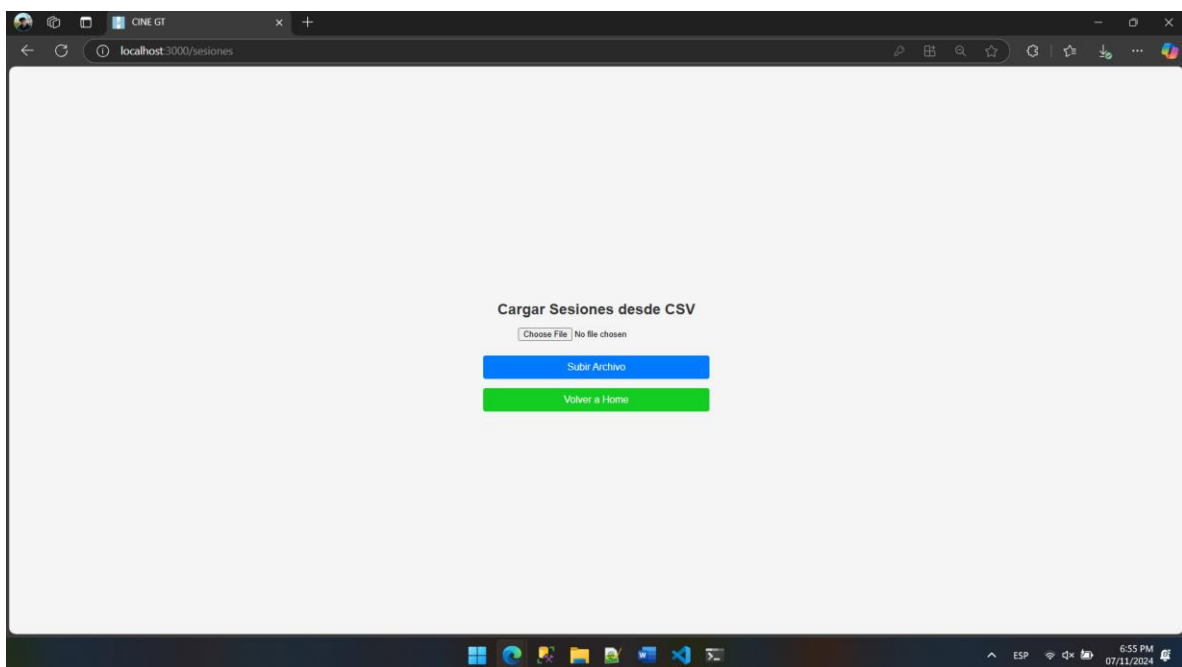
HOME

Seleccionar opción a trabajar



SESIONES

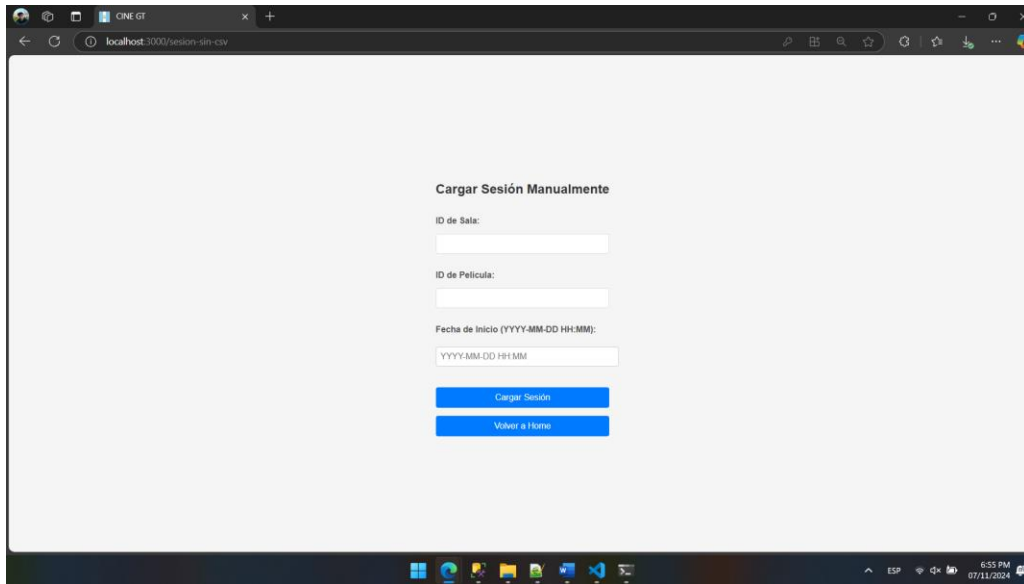
Se selecciona un archivo .csv y se carga



SESIONES SIN CSV

Ingresar:

- Id de la sala
- Id de la película
- Fecha de inicio (AAA-MM-DD HH:MM)

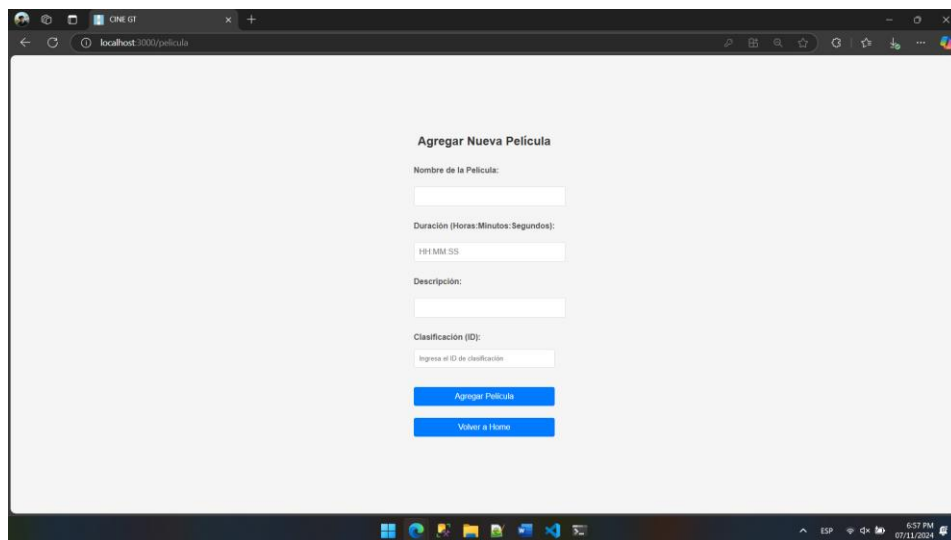


A screenshot of a web browser window showing a form titled "Cargar Sesión Manualmente". The form is centered on a light gray background. It contains three input fields: "ID de Sala:", "ID de Película:", and "Fecha de Inicio (YYYY-MM-DD HH:MM):". Below the input fields are two blue buttons: "Cargar Sesión" and "Volver a Inicio". The browser's address bar shows "localhost:3000/sesion-sin-csv". The Windows taskbar is visible at the bottom of the screen.

PELICULA

Ingresar:

- Nombre de la película
- Duración de la película (HH:MM:SS)
- Descripción
- Id de la clasificación

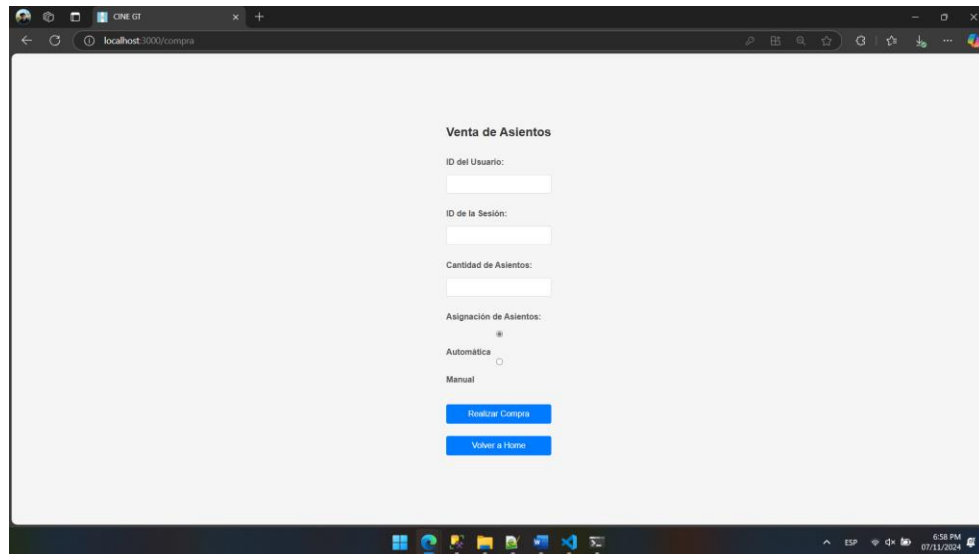


A screenshot of a web browser window showing a form titled "Agregar Nueva Película". The form is centered on a light gray background. It contains four input fields: "Nombre de la Película:", "Duración (Horas:Minutos:Segundos):", "Descripción:", and "Clasificación (ID):". Below the input fields are two blue buttons: "Agregar Película" and "Volver a Inicio". The browser's address bar shows "localhost:3000/pelicula". The Windows taskbar is visible at the bottom of the screen.

COMPRAR

Ingresar:

- Id del usuario
- Id de la sesión
- Cantidad de Asientos
- Asignación Automática o manual
 - Si es manual deberá ingresar los asientos



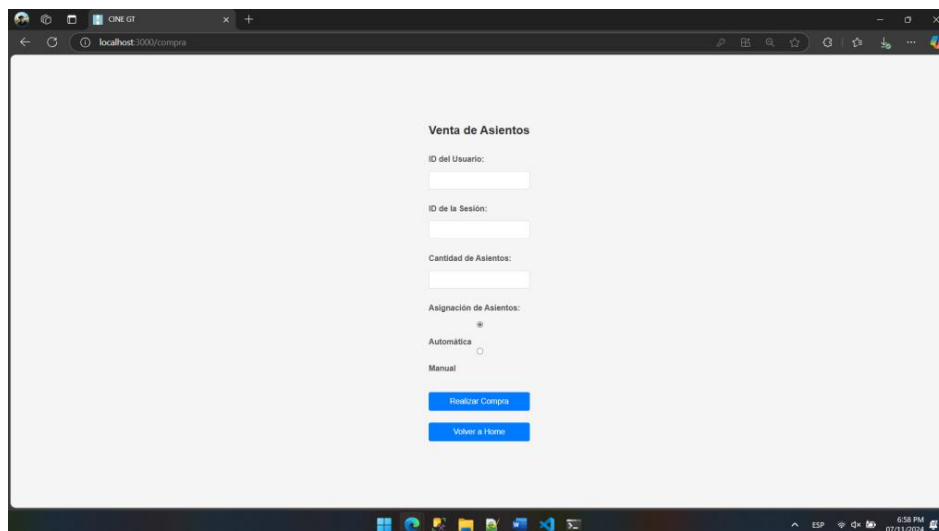
The screenshot shows a web browser window with the URL 'localhost:3000/compra'. The page title is 'Venta de Asientos'. The form contains the following fields and controls:

- ID del Usuario:
- ID de la Sesión:
- Cantidad de Asientos:
- Asignación de Asientos: ☐ Automática ☐ Manual
- Realizar Compra (blue button)
- Volver a Home (blue button)

MODIFICACION

Ingresar:

- Id de la venta
- Id de la nueva sesión
- Nuevos asientos



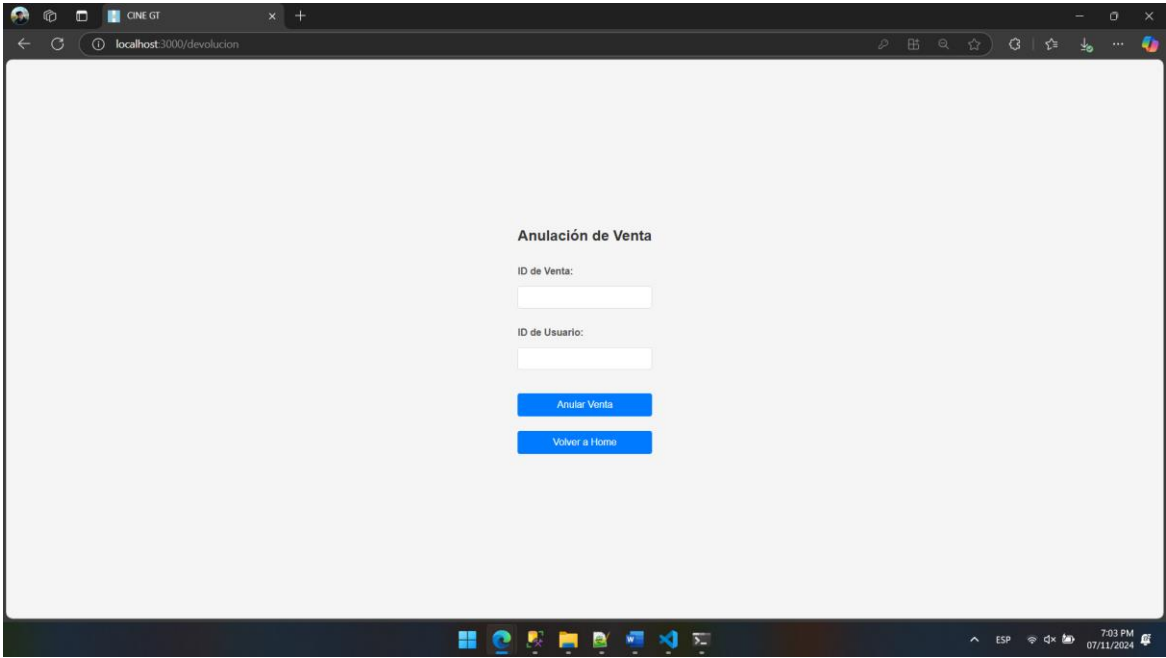
The screenshot shows a web browser window with the URL 'localhost:3000/compra'. The page title is 'Venta de Asientos'. The form contains the following fields and controls:

- ID del Usuario:
- ID de la Sesión:
- Cantidad de Asientos:
- Asignación de Asientos: ☐ Automática ☐ Manual
- Realizar Compra (blue button)
- Volver a Home (blue button)

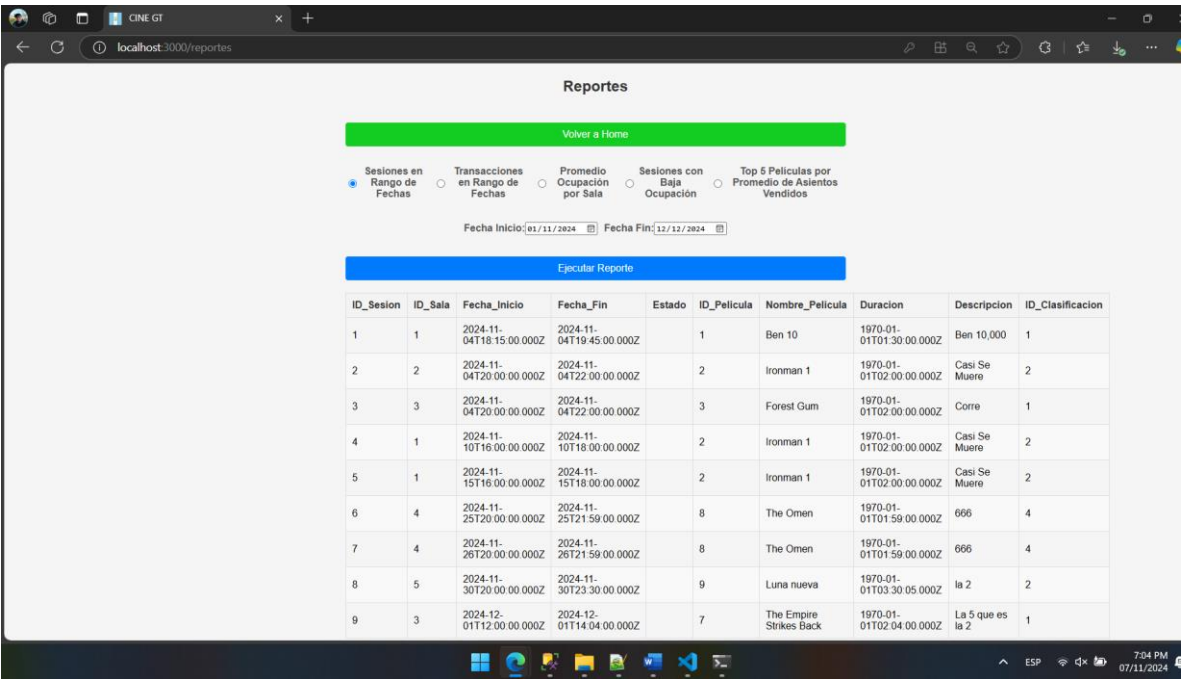
DEVOLUCION/ANULACION

Ingresar:

- Id de venta
- Id usuario



REPORTES



CINE GT

localhost:3000/reportes

Reportes

Volver a Home

Sesiones en Rango de Fechas

Transacciones en Rango de Fechas

Promedio Ocupación por Sala

Sesiones con Baja Ocupación

Top 5 Películas por Promedio de Asientos Vendidos

Fecha Inicio01/11/2024Fecha Fin12/12/2024

Ejecutar Reporte

ID_Log	Fecha_Log	ID_Venta	ID_Usuario	Nombre_Usuario	Monto	Cant_Asientos	ID_Sesion	Fecha_Sesion	Fecha_Fin_Sesion	Nombre_Pelicula	Tipo_Accion
1	2024-11-05T12:48:49.310Z	1	1	Admin	10	1	1	2024-11-04T18:15:00.000Z	2024-11-04T19:45:00.000Z	Ben 10	Venta
2	2024-11-05T18:03:25.730Z	2	1	Admin	10	1	1	2024-11-04T18:15:00.000Z	2024-11-04T19:45:00.000Z	Ben 10	Venta
3	2024-11-05T19:49:29.560Z	3	3	Rodrigo	10	1	2	2024-11-04T20:00:00.000Z	2024-11-04T22:00:00.000Z	Ironman 1	Venta
4	2024-11-05T19:49:41.250Z	4	1	Admin	10	1	1	2024-11-04T18:15:00.000Z	2024-11-04T19:45:00.000Z	Ben 10	Venta
9	2024-11-07T11:01:44.730Z	5	4	Brayan	0	0	7	2024-11-26T20:00:00.000Z	2024-11-26T21:59:00.000Z	The Omen	Venta
11	2024-11-07T11:03:21.727Z	5	4	Brayan	0	0	7	2024-11-26T20:00:00.000Z	2024-11-26T21:59:00.000Z	The Omen	Cambio
12	2024-11-07T11:04:12.357Z	5	4	Brayan	0	0	7	2024-11-26T20:00:00.000Z	2024-11-26T21:59:00.000Z	The Omen	Anulación
13	2024-11-07T11:13:05.873Z	6	4	Brayan	0	0	7	2024-11-26T20:00:00.000Z	2024-11-26T21:59:00.000Z	The Omen	Venta
14	2024-11-07T11:13:59.160Z	6	4	Brayan	0	0	7	2024-11-26T20:00:00.000Z	2024-11-26T21:59:00.000Z	The Omen	Anulación

CINE GT

localhost:3000/reportes

Reportes

Volver a Home

Sesiones en Rango de Fechas

Transacciones en Rango de Fechas

Promedio Ocupación por Sala

Sesiones con Baja Ocupación

Top 5 Películas por Promedio de Asientos Vendidos

ID Sala:4

Ejecutar Reporte

Año	Mes	Cantidad_Sesiones	Promedio_Ocupacion
2024	11	2	8.928571428

CINE GT

localhost:3000/reportes

Reportes

Volver a Home

☐ Sesiones en Rango de Fechas

☐ Transacciones en Rango de Fechas

☐ Promedio Ocupación por Sala

☒ Sesiones con Baja Ocupación

☐ Top 5 Películas por Promedio de Asientos Vendidos

Porcentaje de Ocupación:5

Ejecutar Reporte

ID_Sesion	ID_Sala	Fecha_Inicio	Fecha_Fin	Capacidad_Sala	AsientosOcupados	Porcentaje_Ocupacion
2	2	2024-11-04T20:00:00.000Z	2024-11-04T22:00:00.000Z	56	1	1.78571428571
3	3	2024-11-04T20:00:00.000Z	2024-11-04T22:00:00.000Z	56	0	0
4	1	2024-11-10T16:00:00.000Z	2024-11-10T18:00:00.000Z	56	0	0
5	1	2024-11-15T16:00:00.000Z	2024-11-15T18:00:00.000Z	56	0	0
6	4	2024-11-25T20:00:00.000Z	2024-11-25T21:59:00.000Z	56	0	0
8	5	2024-11-30T20:00:00.000Z	2024-11-30T23:30:00.000Z	56	0	0
9	3	2024-12-01T12:00:00.000Z	2024-12-01T14:04:00.000Z	56	0	0

Reportes

Volver a Home

☐ Sesiones en Rango de Fechas

☐ Transacciones en Rango de Fechas

☐ Promedio Ocupación por Sala

☐ Sesiones con Baja Ocupación

☒ Top 5 Películas por Promedio de Asientos Vendidos

Ejecutar Reporte

Pelicula	Promedio_Asientos_Vendidos
The Omen	1.666666
Ironman 1	1
Ben 10	1

Venta_Asiento		
ID_Venta	int	
ID_Usuario	int	NN
ID_Sesion	int	NN
Cant_Asientos	int	NN
Monto	money	NN
Fecha_Venta	datetime	NN

Usuario		
ID_Usuario	int	
Rol	bit	
Nombre	varchar(100)	NN
Email	varchar(100)	NN
Contraseña	varbinary(256)	NN

Detalle_Asiento		
ID_Detalle_Asiento	int	
ID_Asiento	int	NN
ID_Venta	int	NN
Estado	varchar(20)	NN

Asiento		
ID_Asiento	int	
Fila	varchar(1)	NN
Numero	int	NN
ID_Sala	int	NN

Sala		
ID_Sala	int	
Capacidad	int	NN

Sesion		
ID_Sesion	int	
ID_Sala	int	NN
ID_Pelicula	int	NN
Fecha_Inicio	datetime	NN
Fecha_Fin	datetime	NN
Estado	bit	NN

Pelicula		
ID_Pelicula	int	
Nombre	varchar(100)	NN
Duracion	time	NN
Descripcion	varchar(100)	NN
ID_Clasificacion	int	NN

Clasificacion		
ID_Clasificacion	int	
Tipo_Clasificacion	varchar(100)	NN

Tipo_Accion		
ID_Accion	int	
Descripcion	varchar(50)	NN

Log_Transaccion		
ID_Log	int	
ID_Venta	int	
Fecha	datetime	NN
ID_Usuario	int	NN
ID_Accion	int	NN

Venta_Asiento		
ID_Venta	int	1
ID_Usuario	int	NN
ID_Sesion	int	NN
Cant_Asientos	int	NN
Monto	money	NN
Fecha_Venta	datetime	NN

Usuario		
ID_Usuario	int	1
Rol	bit	
Nombre	varchar(100)	NN
Email	varchar(100)	NN
Contraseña	varbinary(256)	NN

Detalle_Asiento		
ID_Detalle_Asiento	int	
ID_Asiento	int	NN
ID_Venta	int	NN
Estado	varchar(20)	NN

Asiento		
ID_Asiento	int	1
Fila	varchar(1)	NN
Numero	int	NN
ID_Sala	int	NN

Sala		
ID_Sala	int	1
Capacidad	int	NN

Sesion		
ID_Sesion	int	1
ID_Sala	int	NN
ID_Pelicula	int	NN
Fecha_Inicio	datetime	NN
Fecha_Fin	datetime	NN
Estado	bit	NN

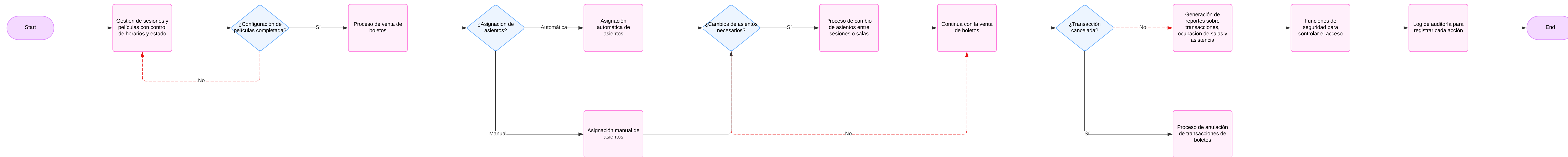
Pelicula		
ID_Pelicula	int	1
Nombre	varchar(100)	NN
Duracion	time	NN
Descripcion	varchar(100)	NN
ID_Clasificacion	int	NN

Clasificacion		
ID_Clasificacion	int	1
Tipo_Clasificacion	varchar(100)	NN

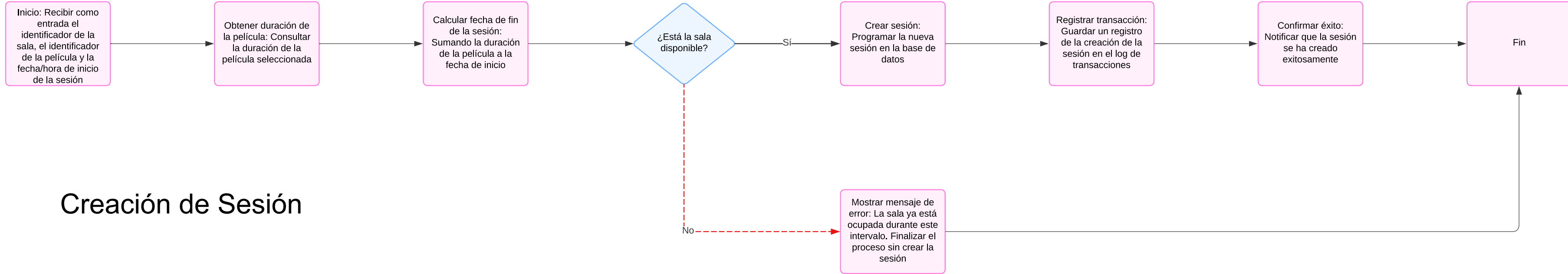
Tipo_Accion		
ID_Accion	int	1
Descripcion	varchar(50)	NN

Log_Transaccion		
ID_Log	int	
ID_Venta	int	
Fecha	datetime	NN
ID_Usuario	int	NN
ID_Accion	int	NN

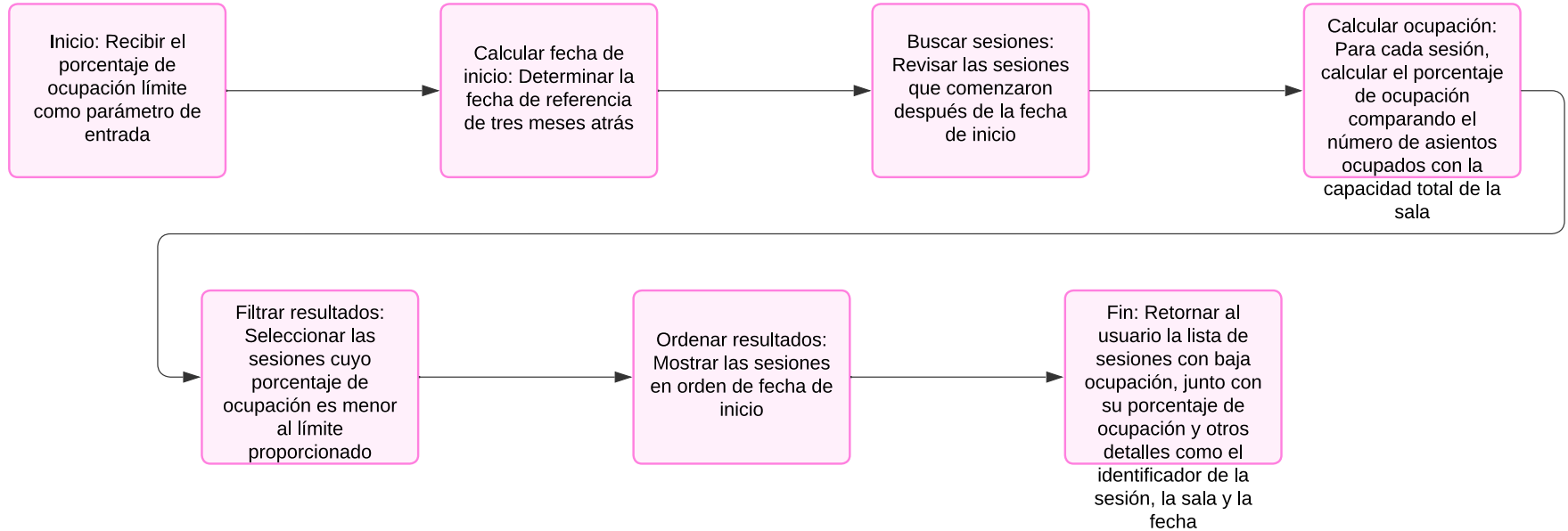
Flujo de acciones de negocio.



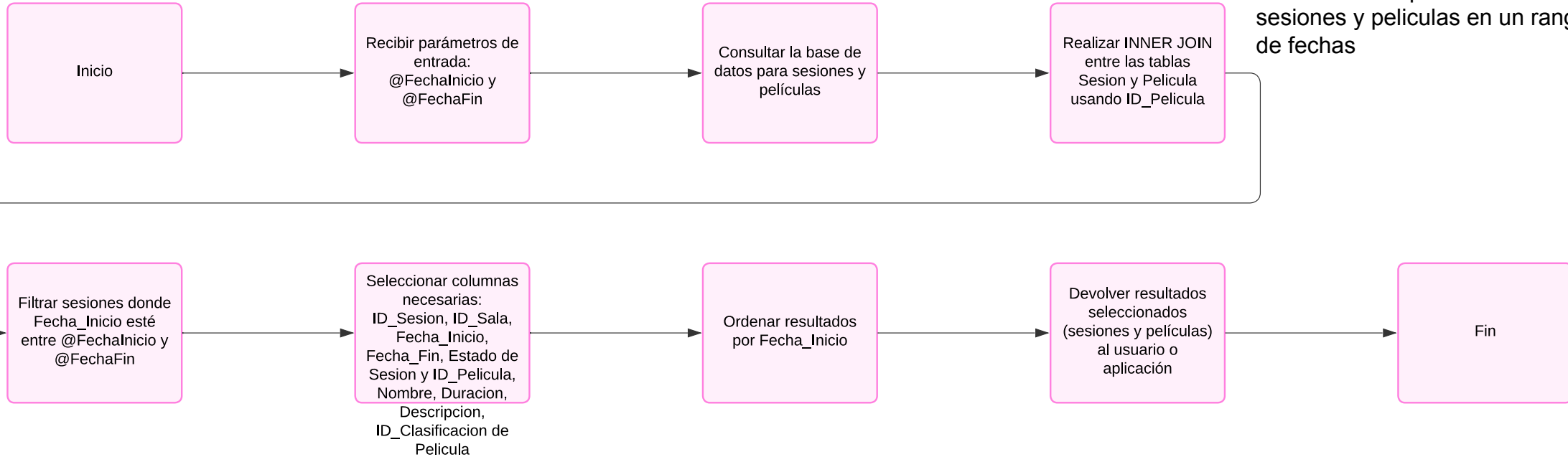
Creación de Sesión



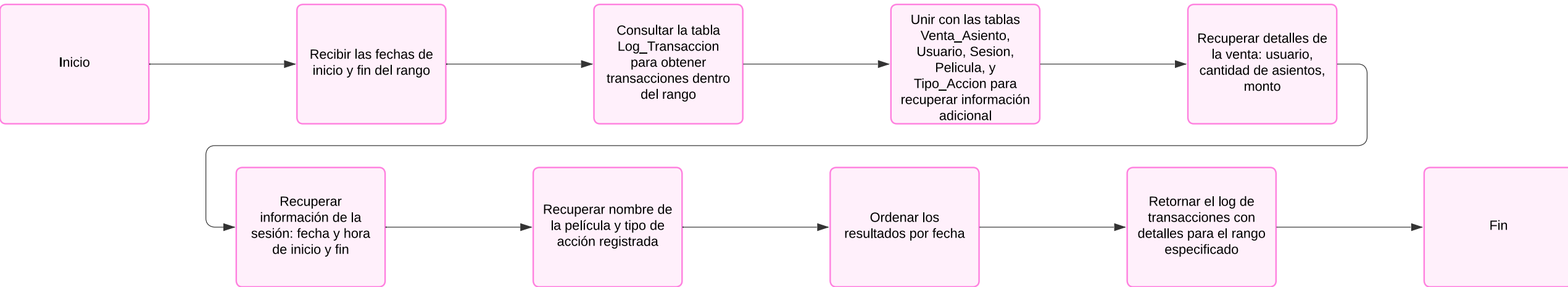
Procedimiento para cantidad de asientos ocupados menor a un porcentaje dado en los últimos 3 meses



Procedimiento para obtener las
sesiones y películas en un rango
de fechas



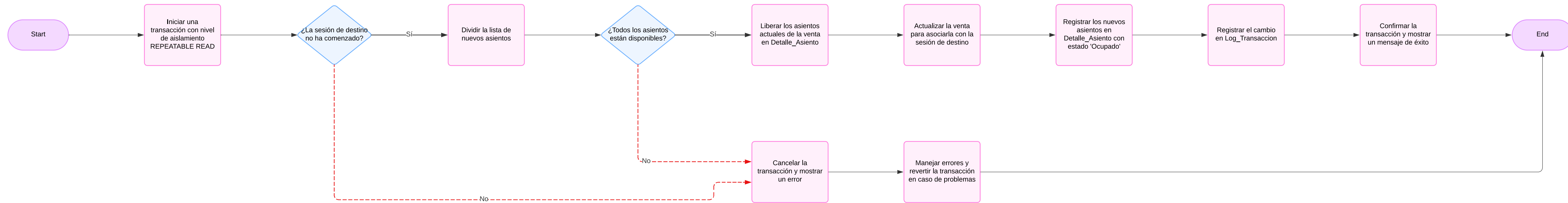
Log transacciones por medio de fechas



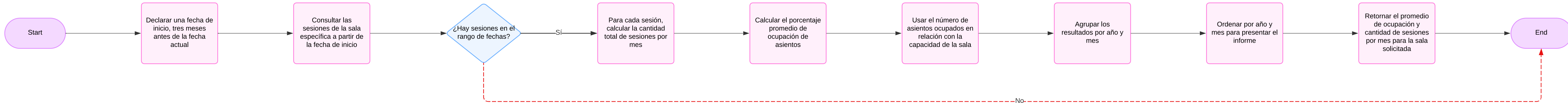
Top 5 películas mayores con promedio de asientos vendidos



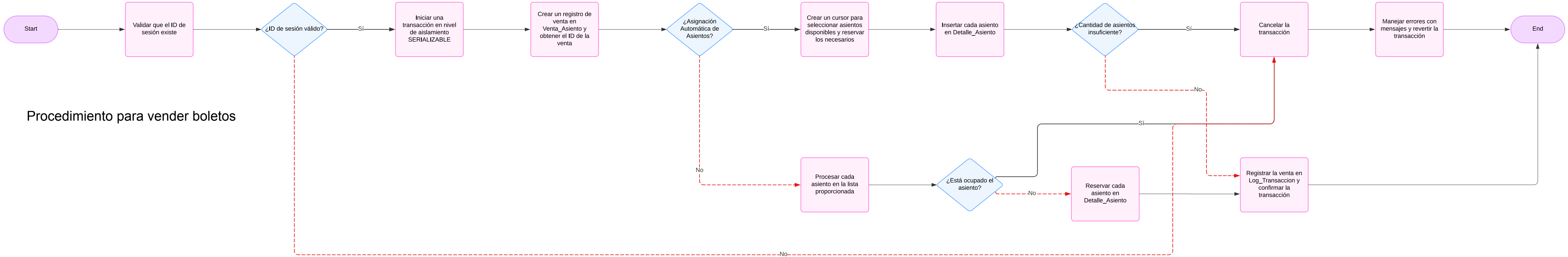
Procedimiento para cambiar un asiento



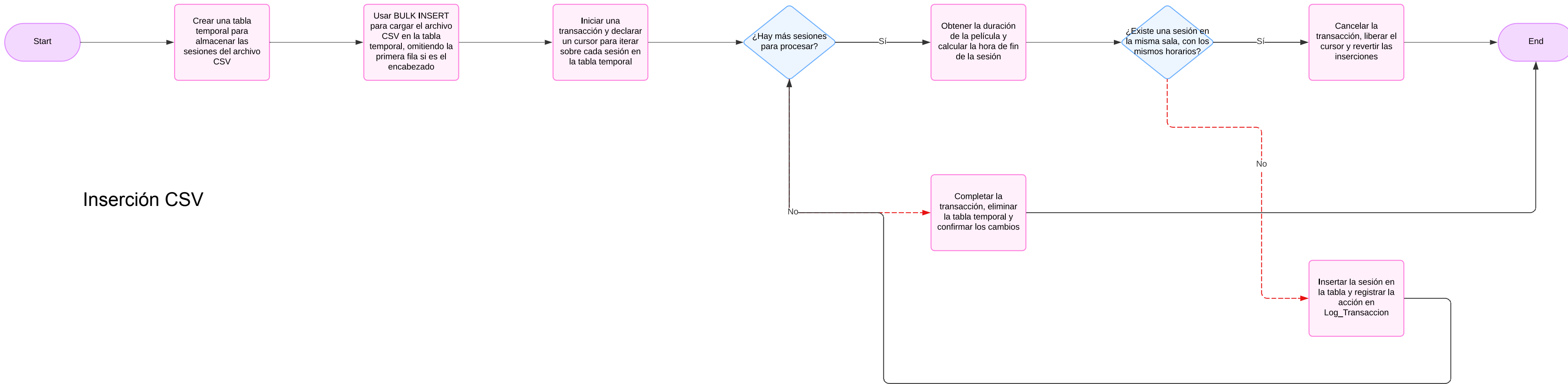
Promedio de Asientos ocupados y cantidad de sesiones



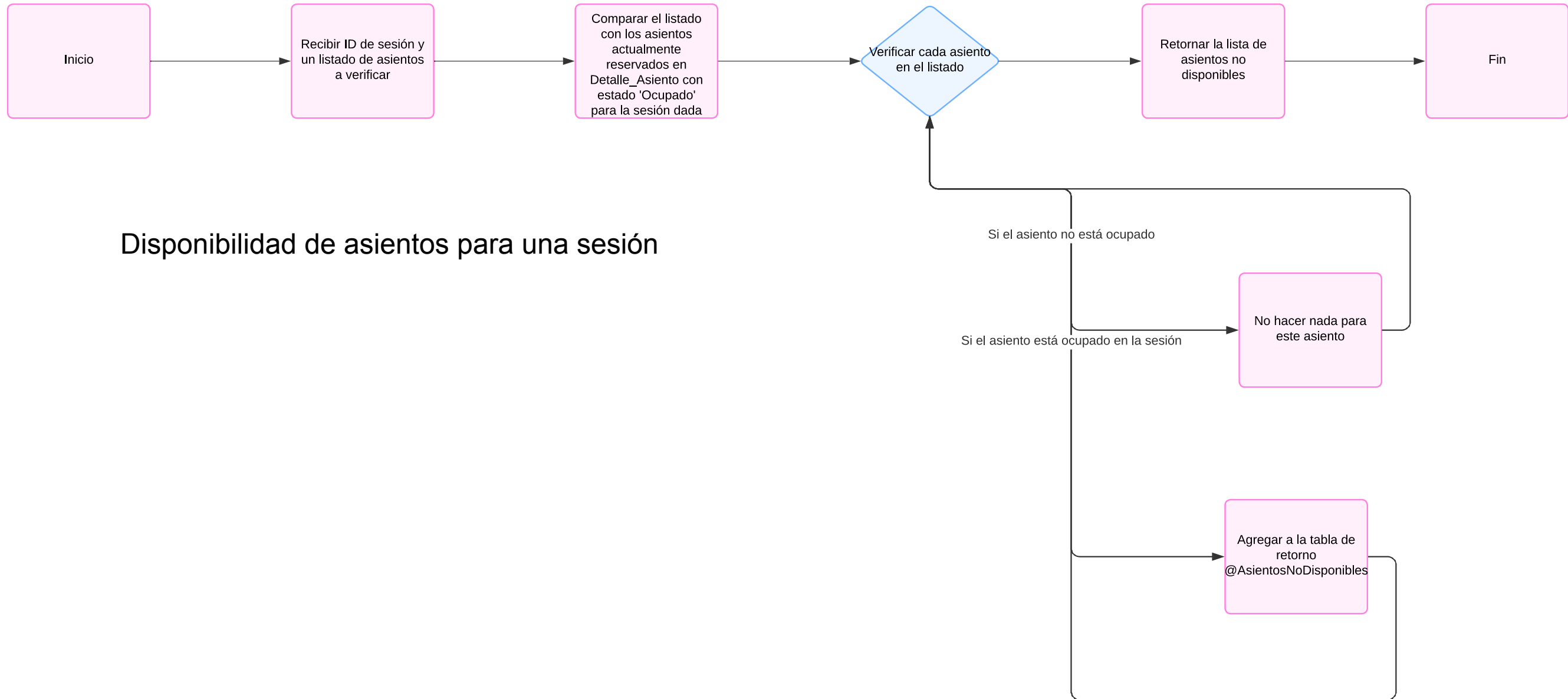
Procedimiento para vender boletos



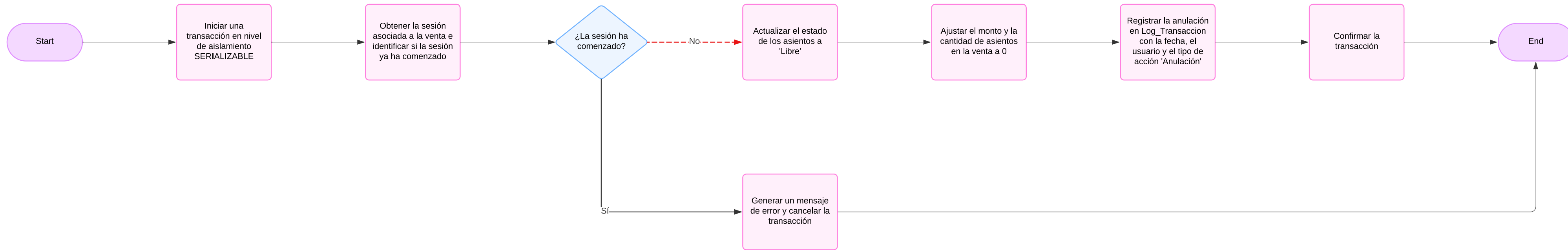
Inserción CSV



Disponibilidad de asientos para una sesión



Anulación de asientos



Validacion de sesiones que no se traslapen

