

Ejercicios prácticos

Curso GeneXus Advanced

GeneXus™ 18

Diciembre 2022

© GeneXus. Todos los derechos reservados. Este documento no puede ser reproducido por ningún medio sin la autorización expresa de GeneXus™. La información contenida en este documento es para uso personal únicamente.

Marca registrada:

GeneXus es una marca comercial o una marca registrada de GeneXus S.A. Todas las demás marcas comerciales mencionadas en este documento son propiedad de sus respectivos dueños

CONTENIDO

CONTENIDO.....	2
EL PROBLEMA.....	4
NUEVO PROYECTO, NUEVA BASE DE CONOCIMIENTO.....	4
MODELO DE DATOS: TRANSACCIONES Y SU DISEÑO DE PARTIDA.....	5
INGRESA DATOS EN EJECUCIÓN	7
REGLAS.....	8
Client Side Validation Behaviour [sugerido]	10
Transacción “Schedule”	10
MÁS REGLAS PARA DEFINIR COMPORTAMIENTO.....	11
opcional.....	16
EVENTOS DE DISPARO DE REGLAS EN TRANSACCIONES.....	16
Eventos de disparo de reglas en transacciones (continuación)	16
Reglas y eventos a nivel de las líneas.....	18
Opcional.....	19
Árbol de evaluación de disparo de reglas y fórmulas.....	20
Nuevo requerimiento:	23
Nuevo requerimiento:	25
opcional (controles sobre líneas en modo update).....	26
opcional: Ejercicio mental sobre árbol de evaluación y eventos.....	27
EVENTOS DE DISPARO DE REGLAS EN TRANSACCIONES (FINAL).....	28
Permitir valores nulos en parte de una clave foránea compuesta.....	30
Consecuencias de agregar un atributo a una tabla que ya tiene datos [SUGERIDO]	30

Transacciones Paralelas [SUGERIDO]	31
SUBTIPOS.....	32
Subtipos: Referencia Múltiple y Especialización	32
Más casos de uso de subtipos.....	33
FÓRMULAS.....	34
Fórmulas Versus Reglas	34
Fórmulas horizontales (opcional)	35
Fórmulas aggregate.....	36
Transacciones Dinámicas.....	38
Eventos en Transacciones.....	39
Más sobre For eachs anidados. Casos y navegación.....	39
Subrutinas.....	41
Cláusula Unique.....	42
Data Selector.....	44
CLÁUSULAS DE PAGINADO	46
Paginado en For each, Pattern y grid (OPCIONAL)	46
BUSINESS COMPONENT.....	47
Actualización de Encargados.....	47
Trabajando con las Películas y sus Actores	48
ACTUALIZACIÓN DE LA BASE DE DATOS CON PROCEDIMIENTO.....	48
PANTALLAS WEB: WEB PANELS	49
Variables en grid y acciones en el pattern Work With (OPCIONAL)	49
Creando pantallas interactivas.....	50

Múltiples Grid.....	51
Grid Sin tabla base.....	53
Cargando Colecciones.....	53
Guardar información de contexto (OPCIONAL)	54
Pantalla especial para la venta de tickets.....	54
PANTALLAS MULTI-CHANNEL: PANELS.....	57
Probando la aplicación en smart devices.....	59
APLICACIONES WEB PROGRESIVAS [SUGERIDO]	60
WEB SERVICES [SUGERIDO].....	64

EL PROBLEMA

Una empresa encargada de gestionar un Cinema te contrata para que desarrolles un sistema con la finalidad de almacenar, manipular y mostrar la información con la que se trabaja. Imagina que el sistema se compone de estos módulos:

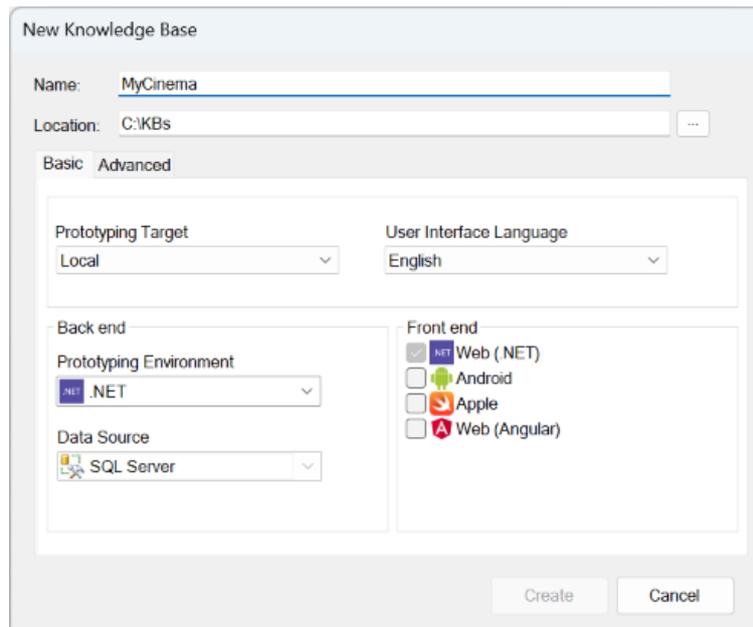
- **Backoffice:** parte de la aplicación que deberá correr en un servidor web, de manera tal que los empleados de la empresa puedan manipular la información desde cualquier lugar con conexión a internet.
- **Frontoffice:** parte de la aplicación que deberá correr en un servidor web, de manera tal que los clientes puedan ver la información con una estética cuidada.

NUEVO PROYECTO, NUEVA BASE DE CONOCIMIENTO

Entra a GeneXus y crea una base de conocimiento de nombre *MyCinema* para comenzar el desarrollo de la aplicación.

Sugerimos:

- Elegir como ambiente de desarrollo Net Core (.Net). Asegúrate de tener instalado todo lo necesario (incluyendo SQL Server). Si usas GeneXus Trial, el ambiente de generación con Net Core y SQL Server ya es predefinido, prototipando en la nube de Amazon.
- No crear la base de conocimiento en la carpeta “Mis Documentos” o cualquier otra carpeta que quede bajo “Documents and Settings”, debido a que estas carpetas tienen permisos especiales otorgados por Windows.



NOTA: Este práctico contiene dos tipos de ejercicios:

1. Ejercicios que se deberán hacer en el IDE de GeneXus
2. Ejercicios que pueden ser omitidos en la aplicación, pero hay un espacio en el práctico para que el alumno escriba las respuestas. (Estos ejercicios también los puedes replicar en el IDE)

MODELO DE DATOS: TRANSACCIONES Y SU DISEÑO DE PARTIDA

El cinema maneja las carteleras de películas que se exhiben en una serie de salas distribuidas en la ciudad. Las personas podrán sacar boletos (tickets) para una función determinada de una película que se exhibe en alguna de las salas, de acuerdo a la cartelera.

Interesa manejar información de las películas, su director, los actores principales, la categoría de la película, su país, etc.

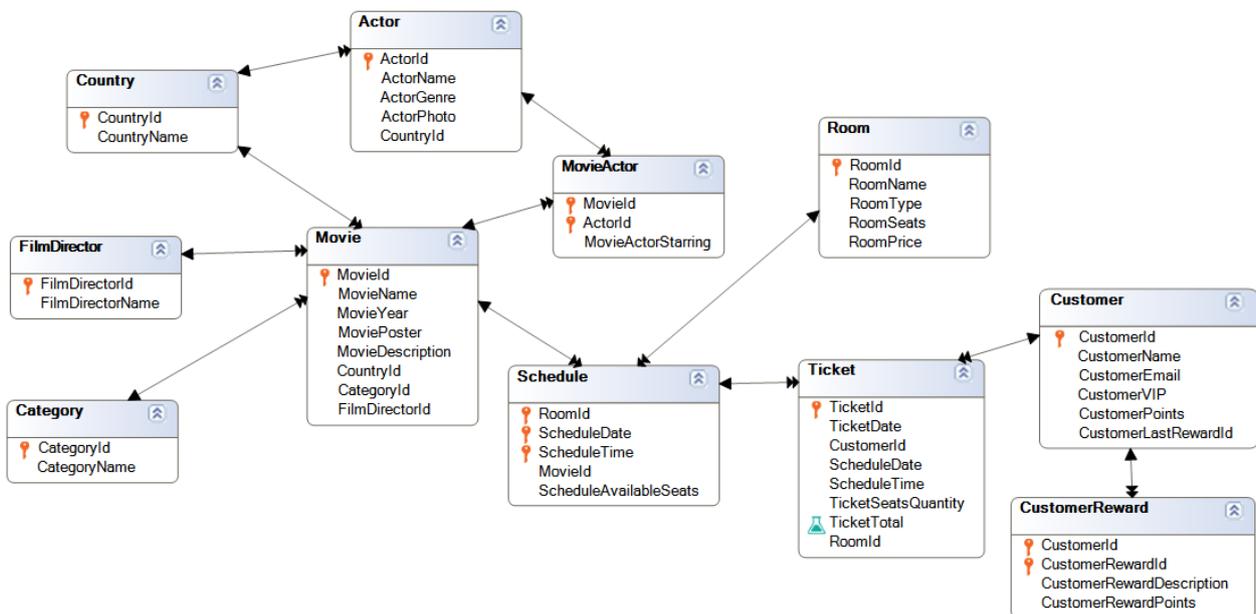
Y también de los clientes del cinema, que podrán tener beneficios de acuerdo al uso que hagan de los servicios.

Para inicializar la KB con las transacciones básicas para empezar a modelar esta realidad, debes importar en la KB el archivo MyCinema.xpz que se adjunta. Bajo el *Folder Trn* verás, entre otras cosas, las transacciones:

- Actor
- Category
- Country
- Customer
- FilmDirector
- Movie
- Room

- Schedule
- Ticket

Analiza detenidamente su estructura e infiere las relaciones entre las entidades de la realidad que modelan. Para ello te recomendamos armar un diagrama de transacciones. También puedes armar uno de tablas (recuerda que con View > Tables puedes obtener el nodo Tables en el KB Explorer con todas las tablas juntas, para arrastrarlas de una sola vez al objeto de tipo Diagram). Si no quieres invertir tiempo en hacerlo, ya brindamos hecho el de tablas (objeto Diagram1).



La transacción Movie es la que registra la información relevante de una película: su nombre, año, el afiche o póster que la representa, una descripción o sinopsis, el país de la película, su categoría (comedia, drama, thriller, etc.) y su director, además de la lista de actores (identificando con el atributo booleano MovieActorStarring si el actor es protagonista o de reparto).

Observa que los actores podrán, por supuesto, ser de países diferentes al país de la película.

Por otro lado, tenemos las salas (Room) en las que las películas se proyectarán. Cada sala además de un nombre y un tipo (si es sala normal, VIP, o 3D), registra la cantidad de asientos que posee, y el precio de la entrada de cine para esa sala. Aquí, evidentemente, se está considerando que los precios de la función dependen de la sala.

Por otro lado tenemos la “función” o “exhibición” en sí, que viene dada por la cartelera, Schedule, donde se registra para cada sala, día, y hora de función, la película que se proyectará y la cantidad de asientos todavía disponibles (no vendidos) para esa función. Observa el identificador compuesto por: sala, fecha y hora de función (las funciones se dan en 4 horarios fijos, y por eso ScheduleTime es

de tipo enumerado). De acuerdo a este modelado una misma película podrá exhibirse en dos salas distintas el mismo día en el mismo horario.

Cuando una persona compra entradas para una función se registra un Ticket, donde se almacenan los datos de la función y la cantidad de asientos comprados, y se calcula el precio total del ticket por ese concepto. Si la persona existe como cliente (customer), se le asocia al ticket.

Además, el cliente genera puntos por compra de tickets, que luego podrá canjear por premios particulares, que también se registrarán.

Observa que se han creado índices unique sobre la mayoría de las tablas, para establecer claves candidatas por nombre.

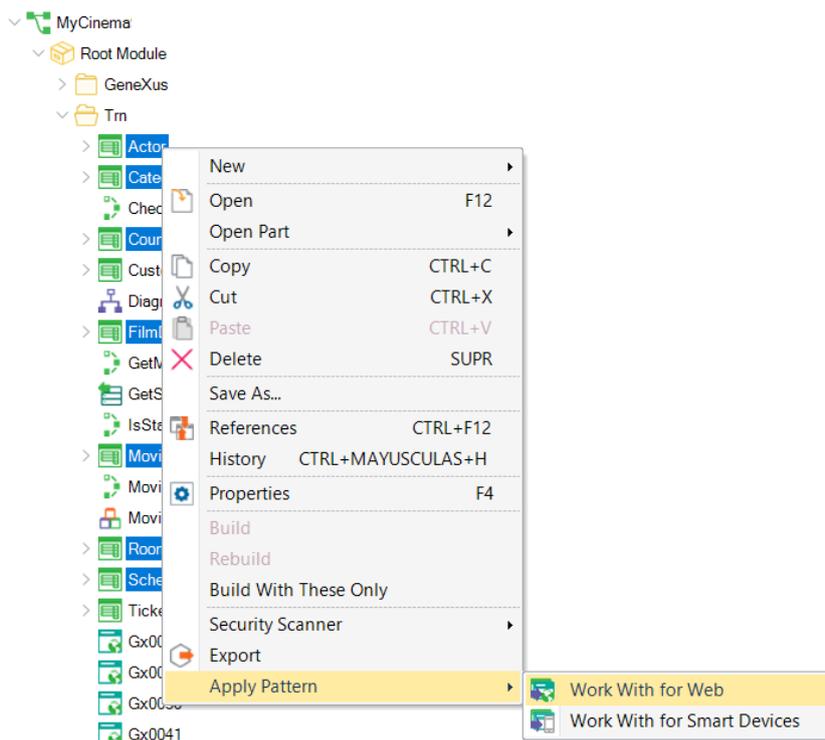
INGRESA DATOS EN EJECUCIÓN

Haz Run Developer Menu. Presiona Create en el listado Impact Analysis, y mientras GeneXus crea las tablas, especifica y genera, observa lo siguiente, teniendo en cuenta que el dominio Id es autonumerado:

1. ¿Qué significa la advertencia sobre las tablas CustomerReward y Customer en el listado de análisis de impacto? Recuerda que solamente pueden autonumerarse atributos que sean, por sí mismos, PK de la tabla. Si son parte de una PK, o no son PK en absoluto, no pueden autonumerarse. Simplemente se ignora su propiedad AutoNumber.
2. Observa que para algunas tablas (Room, FilmDirector, CustomerReward, Customer, Actor, MovieActor, Movie, Country y Carteogry) se informa que si bien la tabla “is new” se va a inicializar con un Data Provider. Si observas bajo esas transacciones encontrarás esos Data Providers. Se utilizarán para cargar automáticamente datos en las tablas en la ejecución siguiente. Sobre este asunto entraremos cuando estudiemos este tema más adelante. Por ahora solamente repara en su existencia.

Navega por los datos cargados hasta estar familiarizado con las transacciones. Observa que las transacciones Schedule y Ticket no cuentan con datos precargados.

Aplica el patern Work With for Web a todas las transacciones excepto a Ticket y Customer. Nota: Recuerda que puedes seleccionarlas todas desde el KB Explorer y desde allí botón derecho:



Ejecuta, y mientras esperas que GeneXus termine, continúa leyendo.

REGLAS

Observa la transacción Room y sus reglas (las no generadas por el pattern):

```

1  Msg("Room name is expected")
2  L   if RoomName.IsEmpty();
3
4  Error("Room seats must be greater than 0")
5  L   if Insert and RoomSeats = 0;
6
7  Msg("Room price not specified")
8  L   if RoomPrice.IsEmpty();
9

```

¿Implementan adecuadamente los requerimientos que se piden a continuación?:

- Se quiere advertir al usuario si el nombre de la sala está vacío, si el precio de la entrada a la sala quedó vacío y si el tipo de sala quedó sin especificar (observa el tipo de datos del atributo RoomType).
- Se debe impedir que se inserte una sala con cero asientos.

Prueba en ejecución:

- Dejar vacío el nombre de una sala previamente cargada. Verás que te permite hacerlo pero mostrándote el mensaje. Es que la regla no está condicionada a un modo en particular. ¿Podrás grabar la sala con nombre vacío?

- Intentar dejar vacío “valor (None)” el combo para RoomType. Observa que sin haber programado ninguna regla se está desplegando el mensaje “Field Room Type is out of range” y no te permite grabar.

Observa que al atributo RoomType, de tipo enumerado, le modificamos la propiedad del grupo Control Info para que permita el valor vacío. De lo contrario ni siquiera verías en el combo la opción “(None)”.

▼ Type Definition	
Supertype	
Based on	RoomType:Domain
Data Type	Numeric
Length	1
Decimals	0
Signed	False
Initial value	
> Validation	
▼ Control Info	
Control Type	Combo Box
Values	Normal:1,VIP:2,3D:3,(None):0
Empty Item	True
Empty Item Text	GX_EmptyItemText
Notify Context Char	False

Pero ese valor no es un valor válido del dominio. Lo colocamos así a nivel de atributo para que automáticamente en todos los Web Panels donde vaya a utilizarse una variable basada en el atributo permita elegir ese “(none)”, vacío. Pero no tiene sentido a nivel del atributo en la transacción, porque ese valor no es un valor válido para el atributo. Si quisieras que GeneXus te permita ingresar un valor vacío en el atributo, es decir, que el usuario pueda no especificarlo, modifica el dominio agregándolo:

Name	Description	Value
Normal	Normal	1
VIP	VIP	2
in3D	3D	3
None	(None)	0

Buttons: Add, Remove, Edit, Move Up, Move Down, OK, Cancel

Observa que es el valor 0 el que hará que la pregunta por “IsEmptyy()” resulte True. Entonces devuelve el valor False para la propiedad Empty Item del atributo RoomType y ejecuta ahora nuevamente la transacción con este cambio, dejando en (None) el valor del atributo. Ahora

no tienes ningún problema dejando el tipo de sala sin especificar. Inténtalo. Deberías agregar una **regla Msg** para advertirle al usuario que dejó el valor sin especificar, por si no era lo que quería.

- Agrega una nueva sala con cero asientos. No te permitirá hacerlo. Pero, ¿y si a una que ya existía le cambias el valor por 0? No deberíamos permitirlo. Modifica la regla para que **nunca** sea posible dejar en cero ese valor cuando se utiliza la transacción.

Observación: ¿Se puede repetir el nombre de una sala? Observa que hemos creado un índice único por RoomName y por ello se está realizando el control de duplicados.

CLIENT SIDE VALIDATION BEHAVIOUR [SUGERIDO]

Antes de continuar probando, interesa que en todas las transacciones no se permita continuar al siguiente campo cuando se dispare una regla error.

También interesa que todos los mensajes de las reglas aparezcan en la parte inferior del campo.

Realiza lo necesario para obtener este comportamiento y vuelve a probar con Room.

Ahora concéntrate en la siguiente transacción.

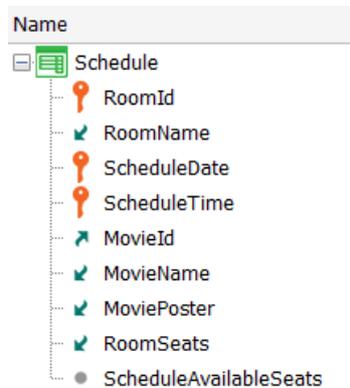
TRANSACCIÓN "SCHEDULE"

La cartelera se compone de la sala (Room), la fecha y el horario en el que se van a presentar las películas (solo en estos horarios: 11:00 AM, 2:00 PM, 5:00 PM y 8:00 PM). Para registrar esos cuatro horarios se ha creado un dominio enumerado, MovieTime.

Name	Description	Value
EarlyMorning	11AM	11
Afternoon	2PM	14
Evening	5PM	17
Night	8PM	20

Aquí no hemos agregado valor (None), pero sí hemos agregado la posibilidad del elemento vacío en el combo.

Observa que hemos optado por utilizar un identificador compuesto y no un autonumerado. Con este diseño, ¿podrá una misma película ser exhibida en dos salas distintas el mismo día y horario?



La respuesta es que sí y esto es correcto pues nos informan que es parte del requerimiento.

El atributo ScheduleAvailableSeats almacenará la cantidad de asientos disponibles (no vendidos aún), por lo que se inicializa con la capacidad de la sala (RoomSeats).

Observa las reglas declaradas en la transacción (fuera de las del pattern) y prueba en ejecución insertar una nueva función, donde:

- RoomId sea inexistente. ¿Qué mensaje se despliega? Coloca una existente.
- La fecha de la exhibición sea anterior a hoy. Coloca una posterior.
- Dejar sin especificar el horario. Sucede lo mismo que habíamos visto para el caso de RoomType. Aquí necesitamos sí o sí que el usuario ingrese un valor válido, pues el horario es parte de la clave. Ingresa uno de los cuatro.
- Dejar sin especificar el MovieId o asignar uno inexistente. Coloca uno válido.
- ¿Por qué RoomSeats y AvailableSeats tienen el mismo valor y son readonly? ¿En qué momento se vio el valor de RoomSeats y el de AvailableSeats? Si no alcanzaste a darte cuenta, confirma y vuelve a empezar ingresando otro Schedule. Presta atención a estos campos cuando sales de RoomId.

MÁS REGLAS PARA DEFINIR COMPORTAMIENTO

Observa la transacción Ticket, que corresponde a los boletos que se le venden a una persona para una función. Si la persona está ingresada en el sistema como cliente, entonces se lo asignará al ticket, y en caso contrario se dejará vacío el campo. La fecha del ticket será la fecha de emitido.

Para el ticket se registran los datos de la función (según Schedule) y la cantidad de asientos que se le están vendiendo a la persona, mostrándose el precio total (calculado de acuerdo a la cantidad de asientos y el costo de la función en esa sala).

Name	Type	Description	Formula	Nullable
 Ticket	Ticket	Ticket		
 TicketId	Id	Ticket Id		No
 TicketDate	Date	Ticket Date		No
 CustomerId	Id	Customer Id		Yes
 CustomerName	Name	Customer Name		
 RoomId	Id	Room Id		No
 RoomName	Name	Room Name		
 RoomType	RoomType	Room Type		
 ScheduleDate	Date	Schedule Date		No
 ScheduleTime	MovieTime	Schedule Time		No
 MovieId	Id	Movie Id		
 MovieName	Name	Movie Name		
 RoomSeats	Numeric(4.0)	Room Seats		
 RoomPrice	Price	Room Price		
 TicketSeatsQuantity	Numeric(4.0)	Ticket Seats Quantity		No
 TicketTotal	Price	Ticket Total	RoomPrice*TicketSeatsQuantity	

Observa las reglas declaradas y reflexiona sobre si implementan adecuadamente los requerimientos que se listan a continuación. Si no lo hacen, modifícalas para que así sea:

Ticket

- Por defecto se debe sugerir la fecha del día de hoy. No permitir modificarla cuando se está insertando. Pero sí en otro modo.
- La fecha de la película no debe ser menor a la fecha de compra. Esto solo en caso de inserción.

Ejecuta la aplicación y agrega un par de tickets de venta, probando las reglas. Por ejemplo:

- Ingresa un ticket y observa en qué momento se disparan las reglas que involucran a TicketDate.
- Edita el ticket recién ingresado y observa si ahora sí puedes modificar la fecha del ticket. Colócale una fecha posterior a la de la película. ¿Te permite continuar y hasta grabar? ¿Por qué no se está disparando la regla de error?
- Presta atención a lo siguiente: ¿Puedes modificar el cliente de un ticket previamente ingresado? ¿y la función (Schedule)? ¿Verdad que sí? No son campos readonly. A partir del agregado de las siguientes reglas verás que ya no podrás modificarlos.

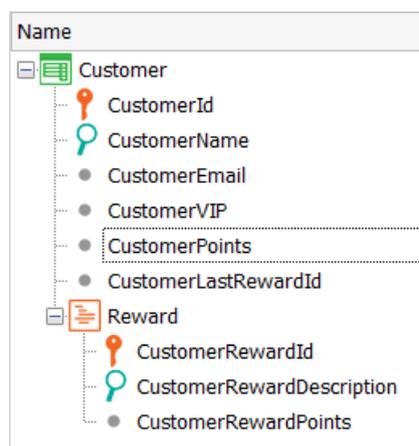
Se requiere agregar el siguiente comportamiento:

1. Verificar que la cantidad de asientos que se están queriendo comprar para la exhibición (Schedule) están disponibles. De lo contrario, que no se permita la venta.
2. Se decremente la cantidad de asientos disponibles para la exhibición (ScheduleAvailableSeats) de acuerdo a la cantidad de asientos que se están comprando.
3. En caso de que la venta esté asociada a un cliente, se deberán sumar a la cuenta del cliente los puntos generados por esta compra. Cada compra de boletos genera el 10% del total del ticket como puntos.

Agrega las reglas a la transacción Ticket que entiendas necesarias para lograr este comportamiento.

Nota: recuerda que si necesitas utilizar un atributo en las reglas de la transacción deberás tenerlo declarado en la estructura, así sea inferido.

Observa que en la transacción Customer tienes el atributo CustomerPoints para almacenar los puntos que el cliente ha generado hasta el momento.



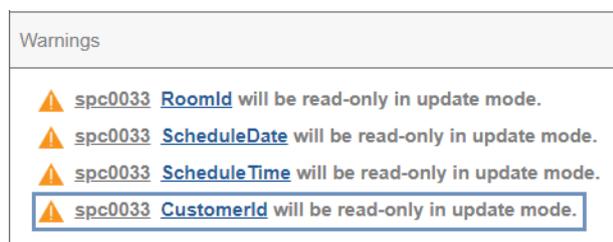
Con los cambios que has realizado, prueba en ejecución:

1. Insertar un nuevo Ticket sin asociarle cliente y llevando una cantidad de asientos que no supere los disponibles. Observa que quede actualizada la cantidad de asientos disponibles de la sala para la exhibición (Schedule).
2. Edita el ticket que acabas de ingresar. ¿Puedes modificar la sala, la fecha o el horario de la función? ¡Ya no! Vuelve a GeneXus y observa estos warnings que aparecieron en el listado de navegación (por ahora dejemos de lado el de CustomerId):



Aparecieron cuando agregaste la regla Subtract. Prueba comentarla y obtener de nuevo el listado de navegación (sin generar el objeto: posicionándote sobre su solapa y con botón derecho eligiendo View Navigation). Verás que ya no están los warnings. ¿Por qué aparecerá esta restricción? Vuelve a dejar la regla subtract descomentada.

3. Inserta un nuevo Ticket asociándole ahora un cliente, y llevando una cantidad de asientos que no supere los disponibles. Observa que se hayan actualizado la cantidad de asientos disponibles de la sala, y la cantidad de puntos del cliente.
4. Edita el ticket anterior. ¿Puedes cambiar el cliente del ticket? En el listado de navegación también encontrarás la advertencia, que se relaciona con la regla Add. ¿Por qué crees que a partir de la introducción de esa regla ya no te permite modificar el cliente?



5. Para el ticket que estás actualizando modifica la cantidad de asientos que se habían ingresado. Corroborar que se actualizan tanto la cantidad de asientos disponibles de la sala, como los puntos del cliente.
6. Edita o inserta un ticket y trata de llevar más asientos de los disponibles. No deberá permitirte grabar.
7. Elimina un ticket con cliente, y observa que los puntos de ese cliente se restituyen.

¿Te quedó así la estructura de la transacción?

Name	Type	Description	Formula	Nullable
Ticket	Ticket	Ticket		
TicketId	Id	Ticket Id		No
TicketDate	Date	Ticket Date		No
CustomerId	Id	Customer Id		Yes
CustomerName	Name	Customer Name		
CustomerPoints	Numeric(4,0)	Customer Points		
RoomId	Id	Room Id		No
RoomName	Name	Room Name		
RoomType	RoomType	Room Type		
ScheduleDate	Date	Schedule Date		No
ScheduleTime	MovieTime	Schedule Time		No
MovieId	Id	Movie Id		
MovieName	Name	Movie Name		
RoomSeats	Numeric(4,0)	Room Seats		
RoomPrice	Price	Room Price		
ScheduleAvailableSeats	Numeric(4,0)	Schedule Available Seats		
TicketSeatsQuantity	Numeric(4,0)	Ticket Seats Quantity		No
TicketTotal	Price	Ticket Total	RoomPrice*TicketSeatsQuantity	
TicketPoints	Numeric(4,0)	Ticket Points	TicketTotal*.10	

El tipo de sala no puede modificarse desde la transacción Ticket porque es inferido. ¿Puedes lograr que el usuario pueda modificarlo desde esta misma pantalla, sin necesidad de tener que hacerlo yendo a la transacción Room? Inténtalo.

Como {RoomId, ScheduleDate, ScheduleTime} conforman juntas una clave foránea a la tabla Schedule, hasta que el usuario no ingresa el último de los valores no se hace el chequeo de integridad sobre esa tabla y el mensaje puede resultar confuso para el usuario.

Si en vez de este mensaje:

Room Id	6
Room Name	Room 6
Room Type	VIP
Schedule Date	05/28/21
Schedule Time	8PM
Movie Id	0

No matching 'Schedule'.

queremos desplegar este otro:

Room Id	6
Room Name	Room 6
Room Type	VIP
Schedule Date	05/28/21
Schedule Time	8PM
Movie Id	0

No movie for Room: 6, Day:05/28/21, Time: 8PM

¿Cómo lo consigues? Si necesitas ver la sintaxis exacta de la regla cuando la clave foránea es compuesta, como en este caso, búscala en wiki.genexus.com.

Puedes armar el string para la salida de forma sencilla así:

```
"No movie for Room:" + RoomId.ToString() + ", Day:" + ScheduleDate.ToFormattedString() + ", Time: " + ScheduleTime.EnumerationDescription()
```

Ejecuta y prueba las reglas que has agregado.

OPCIONAL

Por otro lado, de cada cliente se registran las recompensas que éste puede canjear por sus puntos:

Name	Type	Description	Formula	Nullable
Customer	Customer	Customer		
CustomerId	Id	Customer Id		No
CustomerName	Name	Customer Name		No
CustomerEmail	Email, GeneXus	Customer Email		No
CustomerVIP	Boolean	Customer VIP		No
CustomerPoints	Numeric(4.0)	Customer Points		No
CustomerLastRewardId	Id	Customer Last Reward Id		No
Reward	Reward	Reward		
CustomerRewardId	Id	Customer Reward Id		No
CustomerRewardDescription	Numeric(4.0)	Customer Reward Description		No
CustomerRewardPoints	Numeric(4.0)	Customer Reward Points		No

Estas recompensas son exclusivas del cliente y es por ello que se modelaron como entidad débil. De cada recompensa se registra la cantidad de puntos que vale. Por tanto, cuando un cliente desea cambiar algunos de sus puntos generados por una recompensa, al ingresarse ésta como línea de la transacción deberán actualizarse los puntos del cliente (CustomerPoints).

Por supuesto, no debe permitirse una recompensa si el cliente no cuenta con los puntos que vale.

Implementa este comportamiento y pruébalo en ejecución.

EVENTOS DE DISPARO DE REGLAS EN TRANSACCIONES

Antes de comenzar esta serie de ejercicios, te recomendamos hacerte un esquema en un papel del orden de los eventos de disparo de reglas en transacciones que conoces hasta el momento, lo que a veces también denominamos “momentos de disparo”. Partir de ese esquema claro te será de gran utilidad para aprovechar al máximo los ejercicios. Proponemos ejercicios ligados a cada uno de los tres videos que trabajan este tema. Puedes mirar cada video en orden e ir haciendo los ejercicios relacionados, o puedes mirar los tres y luego hacer todo el bloque de práctico.

Te recomendamos hacer todas las pruebas sobre la transacción Movie con la película “Gone with the Wind”, del director VictorFleming, de United States, tipo Drama y actores “Clark Gable”, “Olivia de Havilland” y “Vivien Leigh”. También recomendamos poner invisible el atributo MovieDescription del archivo de instancia del pattern Work With Web.

EVENTOS DE DISPARO DE REGLAS EN TRANSACCIONES (CONTINUACIÓN)

Supongamos que contamos con un Web Service al que podemos pasarle el nombre de una película y nos devuelve su año. Para hacerlo más simple (porque aún no vimos nada de Web Services), en su lugar utilizaremos el procedimiento GetMovieYear que viene con el xpz.

Cuál será el comportamiento de la transacción Movie si agregamos la siguiente regla:

```
MovieYear = GetMovieYear(MovieName) if Insert;
```

Antes de probarlo en ejecución, piensa en qué momento deberá dispararse. Claramente será únicamente cuando se esté insertando una película nueva, y no en otro modo.

1. ¿Cuando se abre la transacción en modo Insert y antes de pasar por el campo MovieName?
2. ¿Cuando se sale del campo MovieName y se ingresa en el campo MovieYear?
3. ¿Después de salir del campo MovieYear?
4. ¿Recién cuando se confirma?

Antes de probarlo, mueve MoviePoster para que entre MovieName y MovieYear haya otro atributo.

Ahora sí, prueba insertando la película “Gone with the Wind” (escrita exactamente así). Busca en la carpeta Movies que brindamos con el material, la imagen para el póster. Luego de insertada cámbiale el nombre y observa cómo la fecha no se modifica.

Ahora cambia la regla anterior, condicionándola de esta otra manera:

```
MovieYear = GetMovieYear(MovieName) if MovieYear.IsEmpty() on BeforeInsert;
```

¿Cuándo se va a disparar?

Intenta insertar nuevamente la película “Gone with the Wind”, dejando el campo MovieYear vacío y observa que no se disparará la regla hasta que no confirmes. ¿Por qué?

Elimina la película y vuelve a insertarla, pero ahora dejando en el campo MovieYear un valor cualquiera distinto a 1939. ¿Se dispara?

¿Qué diferencia habrá entre condicionar la regla como la tenías y condicionarla de acuerdo a la opción 2 siguiente?

1. `MovieYear = GetMovieYear(MovieName) if MovieYear.IsEmpty() on BeforeInsert;`
2. `MovieYear = GetMovieYear(MovieName) if MovieYear.IsEmpty() and Insert on AfterValidate;`

Respuesta: el evento AfterValidate es inmediatamente anterior al BeforeInsert / BeforeUpdate / BeforeDelete, pero entre medio de ellos no ocurre ninguna acción, por lo que son casi equivalentes.

Prueba ahora cambiar el evento de disparo de la regla por AfterInsert, el momento inmediatamente posterior al BeforeInsert, e intenta insertar nuevamente la película “Gone with the Wind”, dejando vacío el valor MovieYear. ¿Le asigna el valor 1939 tras confirmar o queda vacío?

Vuelve a dejar la regla condicionada correctamente: “`if MovieYear.IsEmpty() on BeforeInsert;`”

Y elimina la película.

REGLAS Y EVENTOS A NIVEL DE LAS LÍNEAS

Observa que con el xpz también venía un procedimiento IsStarring, que utiliza el Data Provider GetStarrings y el SDT MovieStarringActors. La idea es que al procedimiento se le pasa un nombre de película y nombre de actor y devuelve true si ese actor es protagonista (starring) de la película. Otra vez, estaríamos emulando la utilización de un web service que nos brinde esa información.

Por tanto, podrías agregar la siguiente regla a la transacción Movie:

```
MovieActorStarring = IsStarring(MovieName, ActorName) if Insert;
```

Vuelve a ingresar la película “Gone with the Wind”, con los actores “Clark Gable”, “Olivia de Havilland” y “Vivien Leigh”.

La regla se estará disparando en el cliente (además de dispararse luego también en el servidor). ¿Cuándo se dispara en el cliente? Piensa el momento preciso para la ejecución mencionada en la que ingresas los tres actores. ¿Cuántas veces se habrá disparado antes de que el usuario presione el Confirm? ¿Y después del Confirm, en el servidor, cuándo se dispararán?

¿Y si en lugar de asignar directamente valor al atributo MovieActorStarring a partir del procedimiento, queremos dejar que el usuario ingrese el valor, pero chequear contra el procedimiento y preguntarle si está seguro de que desea dejar ese valor, en caso de que no coincida? Es decir, algo así:

```
msg("Are you sure about starring?") if MovieActorStarring <> IsStarring(MovieName,  
ActorName);
```

Otra posibilidad sería que en caso de que el usuario haya dejado en false el campo entonces allí se sobrescriba con el valor devuelto por el procedimiento:

```
MovieActorStarring = IsStarring(MovieName, ActorName) if Insert and MovieActorStarring  
= False;
```

Observa ingresando “Gone with the Wind” que es recién cuando sales del campo MovieActorStarring que la regla se dispara (pruébalo con Clark Gable, dejando en False el campo y con Olivia de Havilland (que no es protagonista) también en False).

Observa que en ese caso marcará el check box para Clark Gable, y si luego vas explícitamente y lo desmarcas, lo vuelve a marcar, es decir, vuelve a disparar la regla.

Si lo que se desea es que el usuario ingrese en su pantalla todos los actores marcando el check box como le parezca y recién cuando confirme para aquellos actores que estaban en False se asigne el valor de acuerdo al procedimiento, ¿cómo debes modificar la regla? Te proponemos 4 opciones:

1. `MovieActorStarring = IsStarring(MovieName, ActorName) if MovieActorStarring = False on BeforeInsert;`
2. `MovieActorStarring = IsStarring(MovieName, ActorName) if MovieActorStarring = False on AfterInsert;`
3. `MovieActorStarring = IsStarring(MovieName, ActorName) if MovieActorStarring = False on AfterLevel Level ActorId;`
4. `MovieActorStarring = IsStarring(MovieName, ActorName) if MovieActorStarring = False on Aftercomplete;`

Si ahora observas estas dos reglas:

1. `MovieYear = GetMovieYear(MovieName) if MovieYear.IsEmpty() on BeforeInsert;`
2. `MovieActorStarring = IsStarring(MovieName, ActorName) if MovieActorStarring = False on BeforeInsert;`

¿Cómo sabe GenXus que la primera se disparará inmediatamente antes de insertar el cabezal en la tabla Movie, y en cambio la segunda se disparará inmediatamente antes de insertar la línea en la tabla MovieActor (para cada línea)?

OPCIONAL

Si tienes tiempo y ganas, puedes probar además lo siguiente: haz un “Save as” de la transacción Movie, MovieCopy1, quitándole todas las reglas que agregó el pattern y todos los eventos (de la solapa Events), así como colocando en False la propiedad Data Provider. De esta forma podrás trabajar directamente sobre ella (más adelante verás el tema Transacciones Paralelas donde verás que trabajará sobre la misma tabla que Movie) invocándola desde el Developer Menu.

Y agrégale las siguientes reglas, definiendo la variable &ActorLine de tipo Numeric:

```
&ActorLine = &ActorLine + 1
    on AfterInsert
    Level ActorId;

Msg(&ActorLine.ToString() + " actors. Directed by: "+ FilmDirectorName)
    If Insert
    on AfterLevel
    Level ActorId;
```

La idea es sumar 1 después de insertar cada línea en la tabla MovieActor, y disparar el mensaje luego de que se hayan insertado todas las líneas. Prueba en ejecución.

Luego cambia la cláusula Level de la primera regla, para que ahora sea el level de FilmDirectorId y observa qué sucede.

ÁRBOL DE EVALUACIÓN DE DISPARO DE REGLAS Y FÓRMULAS

En la transacción Movie ya nos habíamos preguntado cuándo se iba a disparar esta regla:

```
MovieYear = GetMovieYear(MovieName) if Insert;
```

Si en el layout web los controles se encuentran en este order:

Si pensamos en que cada regla se disparará “lo antes posible”, y que “lo posible” tiene que ver con qué información necesita contar para ello, será claro que la asignación a MovieYear va a depender del modo de la transacción (Insert), y de contar con el valor que el usuario le haya dado al atributo MovieName.

- Si la transacción se ejecuta recibiendo por parámetro el modo (como cuando es invocada por el Work With) entonces ni bien sea abierta ya sabrá si el modo es o no Insert. Pero como aún necesita saber el valor de MovieName, no podrá disparar la regla hasta que el usuario abandone ese campo. Si, en cambio, la regla se hubiese escrito:

```
MovieYear = GetMovieYear() if Insert;
```

Es decir, sin necesitar de ninguna otra cosa que el modo para dispararse, entonces ni bien se abre la transacción en Insert ya se disparará la regla, y veremos el cursor posicionado sobre el campo MovieName vacío, pero con un valor ya cargado en MovieYear.

- Si, en cambio, la transacción se ejecuta sin regla parm (como sería el caso de la transacción paralela MovieCopy1 con la que terminamos el ejercicio anterior), para saber el modo deberá esperarse a que el usuario abandone el campo MovieId. Pero todavía no podrá dispararse, porque aún no pasamos por el campo MovieName, que es necesario pues la regla depende de ese valor. Si la regla se hubiese escrito como:

```
MovieYear = GetMovieYear() if Insert;
```

Entonces ni bien el usuario abandona MovieId la transacción ya disparará la regla.

Y si en esa transacción tenemos escritas en este orden las siguientes reglas, ¿cuándo se disparará cada una?

```
MovieYear = GetMovieYear(MovieName) if Insert;  
error("The name is empty")if MovieName.IsEmpty();
```

Para cualquiera de las dos, es necesario que el usuario haya salido del campo MovieName. Por lo que se evaluarán y dispararán en el orden en el que están definidas. Por supuesto, la segunda se disparará tanto en modo Insert como en los demás.

Si ahora agregamos la regla Msg siguiente antes que las otras dos, ¿cuándo se va a disparar?

```
msg("The movie belongs to the classic film era")  
    if MovieYear < 1950;  
  
MovieYear = GetMovieYear(MovieName)  
    if Insert;  
  
error("The name is empty")  
    if MovieName.IsEmpty();
```

Observa que, por depender del atributo MovieYear será después no solo de que se dispare la segunda regla, la que le asigna valor a MovieYear, sino que además será después de salir del campo MovieYear, porque el usuario podría querer modificar el valor que se le asignó en la segunda regla. Aquí es donde queda clara la importancia del ordenamiento de las reglas según el árbol de dependencias, pero también el lugar de los controles en la pantalla y el flujo de interacciones del usuario.

Piensa en el siguiente flujo:

1. Ingresas en modo Insert
2. Escribes algo en el campo MovieName o lo dejas vacío y sales de él. Se evaluará la regla error y también se disparará la regla de asignación, por lo que verás en pantalla para MovieYear el valor devuelto por el proc, aunque aún no hayas pasado por ese campo.
3. Ingresas una foto en MoviePoster.

4. Pasas al campo MovieYear. Puedes dejar el valor que asignó la regla, o modificarlo. En cualquier caso, recién cuando salgas del campo se entenderá que debe evaluarse la condición de la regla msg.
5. Decides volver al campo MovieName a cambiar el nombre de la película. ¿Qué se re-disparará cuando salgas del campo?

Observa el listado de navegación de la transacción. ¿Te informa del orden de ejecución de las reglas? Por defecto no lo hace. Si quieres ver ese informe detallado, tendrás que ir a Tools > Options y en la opción Build cambiar por True el valor de la propiedad Detailed Navigation.

Ahora vuelve a obtener el listado de navegación. Aquí mostramos el de la trn. MovieCopy1 (solo con las reglas anteriores; todas las demás comentadas):

```

Level Movie
READ Movie
WHERE
  Movie.MovieId = MovieId
INTO MovieYear MovieName MoviePoster MoviePoster.Uri MovieDescription CountryId CategoryId FilmDirectorId

READ FilmDirector
WHERE
  FilmDirector.FilmDirectorId = Movie.FilmDirectorId
INTO FilmDirectorName

READ Category
WHERE
  Category.CategoryId = Movie.CategoryId
INTO CategoryName

READ Country
WHERE
  Country.CountryId = Movie.CountryId
INTO CountryName

READ Movie UNIQUE
WHERE
  Movie.MovieName = MovieName

MovieYear = GetMovieYear ( MovieName ) IF insert,
Error( "The name is empty" ) IF MovieName isempty()
Msg( "The movie belongs to the classic film era" ) IF MovieYear < 1950

INSERT INTO Movie (MovieYear, MovieName, MoviePoster, MoviePoster.Uri, MovieDescription, CountryId, CategoryId, FilmDirectorId)
UPDATE Movie (MovieYear, MovieName, MoviePoster, MoviePoster.Uri, MovieDescription, CountryId, CategoryId, FilmDirectorId)
DELETE FROM Movie

Referential integrity controls on delete:
  • Schedule ( MovieId )
Level MovieActor

```

Por último, ¿qué sucederá si a alguna de las reglas anteriores le agregas un evento de disparo? Por ejemplo:

```
msg("The movie belongs to the classic film era")
  if MovieYear < 1950;
```

```
MovieYear = GetMovieYear(MovieName)
  on BeforeInsert;
```

```
error("The name is empty")
  if MovieName.IsEmpty();
```

A partir de aquí, la regla de asignación se independiza del árbol de dependencias que se va disparando a medida que se va pasando por los campos (tanto interactivamente en el browser, como en el Server, antes de validarse el cabezal e insertarse). La regla se disparará solo en el Server, y será mucho después de que en el server se vaya pasando por cada campo y disparando sus reglas. Será, de hecho, cuando se haya terminado de pasar por todos los campos del cabezal, y se haya validado todo lo referente a ese cabezal (la integridad referencial, que los valores estén dentro de los rangos permitidos, etc.) y un instante antes de insertar ese cabezal en la tabla Movie.

Observa ahora el listado de navegación de la transacción y compáralo con el anterior.

Level Movie ⤴

```

READ Movie
WHERE
  Movie. MovieId = MovieId
INTO MovieYear MovieName MoviePoster MoviePoster.Uri MovieDescription CountryId CategoryId FilmDirectorId

READ FilmDirector
WHERE
  FilmDirector. FilmDirectorId = Movie. FilmDirectorId
INTO FilmDirectorName

READ Category
WHERE
  Category. CategoryId = Movie. CategoryId
INTO CategoryName

READ Country
WHERE
  Country. CountryId = Movie. CountryId
INTO CountryName

READ Movie UNIQUE
WHERE
  Movie. MovieName = MovieName

Error( "The name is empty" ) IF MovieName. isempty()
Msg( "The movie belongs to the classic film era" ) IF MovieYear < 1950
Before Insert Rules
  MovieYear = GetMovieYear ( MovieName );

INSERT INTO Movie (MovieYear, MovieName, MoviePoster, MoviePoster.Uri, MovieDescription, CountryId, CategoryId, FilmDirectorId)
UPDATE Movie (MovieYear, MovieName, MoviePoster, MoviePoster.Uri, MovieDescription, CountryId, CategoryId, FilmDirectorId)
DELETE FROM Movie

Referential integrity controls on delete:
  • Schedule ( MovieId )

```

Level MovieActor ⤴

NUEVO REQUERIMIENTO:

Nos piden controlar que al agregar una película por lo menos se registren dos actores, de lo contrario la aplicación no nos deberá permitir guardar la película.

Movie

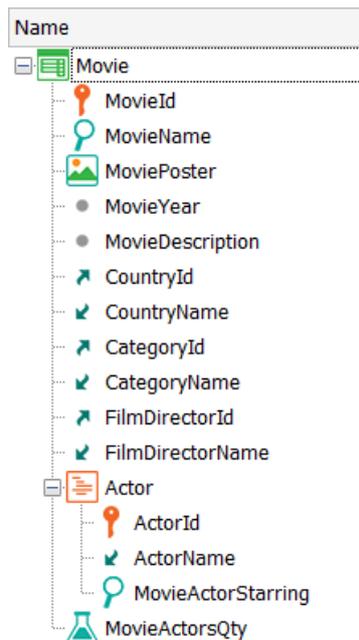
❗ Select at least two actors

Id	1
Name	<input type="text" value="Avatar 2: The way of water"/>
Year	<input type="text" value="2022"/>
Poster	
Description	Jake Sully and Ney'tiri have formed a family and are doing everything to stay together. However, they must leave their home and explore the regions of Pandora. When an ancient threat resurfaces, Jake must fight a difficult war against the humans.

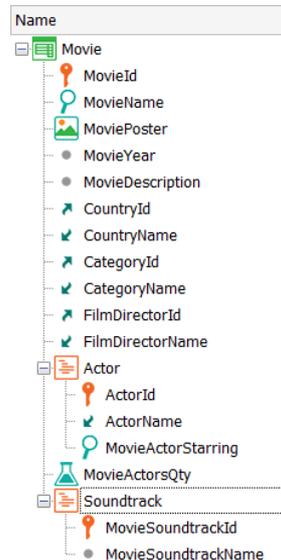
¿Cómo condicionas la regla de error?

```
Error("Select at least two actors")
  If <condition>
    on <Event>;
```

¿La estructura de la transacción te quedó ahora así?



¿Hay alguna diferencia en este caso entre elegir el evento “AfterLevel Level MovieActorStarring” y el “BeforeComplete”? ¿Y si hubiera otro nivel paralelo a Actor y posterior a él, por ejemplo, Soundtracks, habría diferencia?



NUEVO REQUERIMIENTO:

Cada tanto se realizan festivales de cine, para los cuales las distintas productoras envían en préstamo copias de sus películas para ser exhibidas, y no es raro que una productora tome la decisión de enviar la película a último momento.

Por este motivo, queremos dar la oportunidad al usuario de ingresar los datos de una película: cabecal y líneas y a lo último, antes del Commit, recién allí chequear si se recibió el permiso con la copia para exhibición o no. Si no se recibió, entonces no debe ingresarse la película al sistema.

Tenemos en la KB el procedimiento MovieInStock que lo que hace es verificar si ha llegado al Cinema ese permiso. No importa su código (el que brindamos en la KB produce un verdadero o falso randómico, solo para poder probar).

Y lo invocamos en las rules de la transacción Movie:

```
&IsInStock = MovieInStock(MovieId) If Insert;
```

Hacemos esto suponiendo que en el procedimiento se va a acceder a la tabla Movie para poder recuperar el MovieName, por ejemplo. Es decir, necesitamos que el registro del cabecal, por lo menos, esté insertado.

Pero, en verdad, ten en cuenta de que es preferible que ya se haya realizado todo lo posible, para dar más tiempo. Es decir, queremos demorar el chequeo hasta último momento.

¿Qué evento de disparo deberíamos agregarle? ¿Y qué más debemos hacer para cumplir con el requerimiento de que no quede la película grabada si no llegó el permiso y la copia?

Para saber qué te devolvió el proc (dado se programó como randómico) puedes modificarlo para que te devuelva False en un caso y True en otro y así puedes controlar el comportamiento.

OPCIONAL (CONTROLES SOBRE LÍNEAS EN MODO UPDATE)

Ahora cambiamos un poco las cosas, y agregamos un atributo MovieStatus de un dominio enumerado de igual nombre (venía creado en el xpz), que permite dos valores Pending y Completed. De esta manera permitiremos ir ingresando una película en más de una instancia, y recién la damos por buena cuando el usuario pasa a Completed el valor de ese campo.

Dejaremos la corroboración de que estén correctos los datos de “starring” de los actores para cuando el usuario dé por completado el ingreso de los datos, marcando para MovieStatus el valor Completed.

La estructura de la transacción, entonces, te quedará:

Name	Type
Movie	Movie
MovieId	Id
MovieName	Name
MoviePoster	Image
MovieYear	Numeric(4.0)
MovieDescription	LongVarChar(1M)
CountryId	Id
CountryName	Name
CategoryId	Id
CategoryName	Name
FilmDirectorId	Id
FilmDirectorName	Name
MovieStatus	MovieStatus
Actor	Actor
ActorId	Id
ActorName	Name
MovieActorStarring	Boolean
MovieActorsQty	Numeric(4.0)

Quita, claro, la regla que asignaba valor al atributo MovieActorStarring, y las que no quieras que se disparen para probar este nuevo requerimiento -por ejemplo, los mensajes-. Deberás agregar las nuevas reglas (definiendo para &ok el tipo Boolean):

```
Default(MovieStatus, MovieStatus.Pending);

&ok = CheckMovieActors(MovieId)
    if MovieStatus = MovieStatus.Completed
    on <event>;
```

```

Error("Problem with starrings")
  if not &ok and MovieStatus = MovieStatus.Completed
  on <event>;

```

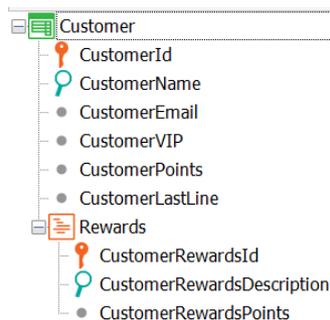
Observa que el procedimiento CheckMovieActors viene dentro del xpz que importaste al principio. ¿Qué evento elegiste?

Antes de probar, asegúrate de que se haya agregado al Web Layout el atributo MovieStatus. Si no lo hizo, ve al menú principal, y elige Edit > Apply Default (Web Layout).

Prueba (tendrás que reorganizar) ingresando nuevamente la película “Gone with the Wind”. Recuerda que es el Data Provider GetStarrings el que determina cuáles son y cuáles no actores protagonistas. Juega con esos datos, dejando primero la película en status Pending pero con algún dato de starring mal. Verás que te permite grabar. Luego ingresa en modo Update, cambia el estado a Completed, y verás desplegarse el mensaje de error sobre el campo MovieStatus. Modifica los errores en los starrings de los actores hasta que te permita grabar con éxito.

OPCIONAL: EJERCICIO MENTAL SOBRE ÁRBOL DE EVALUACIÓN Y EVENTOS

Tomando en cuenta la siguiente estructura para la transacción Customer:



A continuación, se muestra una lista de las reglas programadas en dicha transacción.

Coloca en el paréntesis el correcto orden de ejecución (no tienes que probarlo):

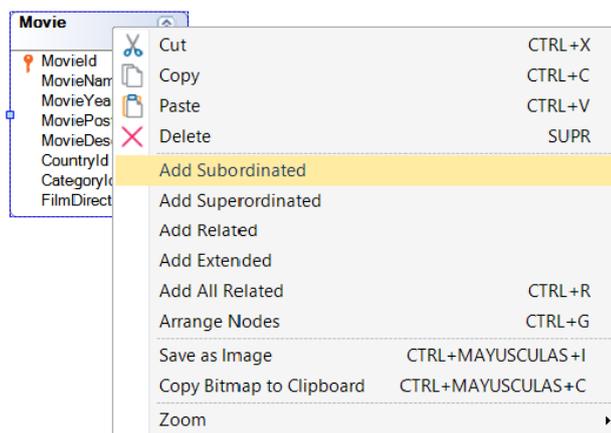
```

( )error('The name is empty') if CustomerName.IsEmpty();
( )msg('The email is empty') if CustomerEmail.IsEmpty();
( )PrintCustomerList() on afterComplete;
( )Subtract(CustomerRewardsPoints, CustomerPoints);
( )error("Not enough points") if CustomerPoints < 0;
( )PrintRewards(CustomerId, CustomerRewardsId) on AfterInsert;
( )serial(CustomerRewardsId, CustomerLastLine, 1);
( )noaccept(CustomerRewardsId);
( )Checkpoints(CustomerId) if update on AfterInsert;

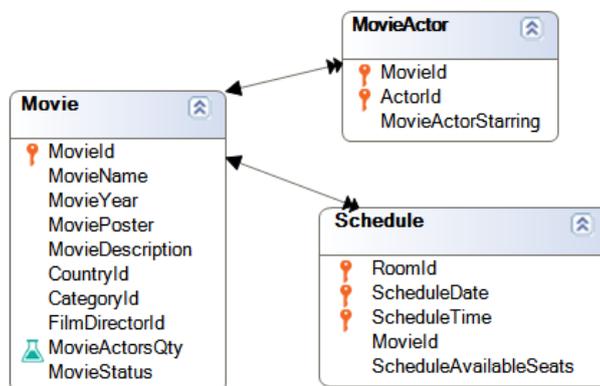
```

EVENTOS DE DISPARO DE REGLAS EN TRANSACCIONES (FINAL)

Crea un nuevo objeto diagrama, y arrastra allí dentro la tabla Movie. Luego posicionado sobre la tabla Botón derecho > Add Subordinated, para obtener todas las tablas que tienen claves foráneas referenciando a Movie.



Serán solo 2:



Una es la obvia, la que corresponde al segundo nivel de la transacción. La única otra que tenemos es la de la transacción Schedule.

Por tanto, ¿qué sucederá si intentas desde la transacción Movie eliminar una película para la que hay registros en Schedule relacionados?

Inténtalo. Controlará la integridad referencial y no permitirá eliminar la película. Pero... ¿en qué momento exacto de la ejecución de la transacción Movie se realizará este control?

Si queremos cambiar este comportamiento de manera tal de que en caso de que existan “funciones” relacionadas (registros de Schedule) éstas se eliminen inmediatamente antes de que el control de

integridad referencial se realice en la transacción Movie, y por tanto no falle y por tanto nos permita eliminar la película, ¿cómo lo conseguimos?

Observa que cuentas con el procedimiento DeleteSchedules ya implementado, que elimina todos los registros de Schedule del MovieId recibido por parámetro.

Como pista, deberás invocarlo desde las reglas de la transacción Movie, condicionado al modo, pero... ¿en qué evento de disparo?

```
DeleteSchedules(MovieId)
  If Delete
    on .....;
```

Recuerda: debe ser inmediatamente antes de que se realice el control de integridad referencial relativo a la eliminación del registro del cabezal de la tabla Movie. ¿Cuál es ese momento? Elígelo entre los que siguen:

1. BeforeValidate
2. AfterValidate
3. BeforeDelete
4. AfterDelete
5. BeforeComplete
6. AfterComplete

Y prueba en ejecución eliminar una película que tenga schedules asociados (sin tickets).

Si había tickets para el Schedule que estás queriendo eliminar dentro del procedimiento, tendrás una excepción de base de datos. El por qué lo entenderás más adelante. ¿Y si intentas la eliminación a través del Business component Schedule (tendrás que prender la propiedad Business Component de la transacción para poder utilizar variable de tipo Schedule) en el procedimiento DeleteSchedules? No te permitirá eliminar, pero al menos no se caerá el programa.

```
parm(in:MovieId);

For each Schedule
  &schedule.Load(RoomId, ScheduleDate, ScheduleTime)
  &schedule.Delete()

Endfor
```

Elimina los tickets de la película y vuelve a probar.

PERMITIR VALORES NULOS EN PARTE DE UNA CLAVE FORÁNEA COMPUESTA

En ocasiones se puede vender boletos de cine “abiertos” para que el cliente asista a la función en el horario que más le convenga, es decir, se puede vender un boleto para una determinada sala el día de hoy pero el cliente puede llegar al Cinema a la hora que desee.

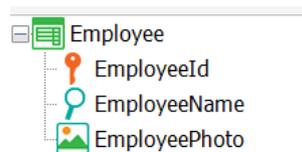
¿Qué cambio se debe realizar en la transacción Ticket para que no se tome en cuenta el control de la integridad referencial en la tabla Schedule entre RoomId, ScheduleDate y ScheduleTime?

Realiza los cambios necesarios e intenta ingresar un ticket sin asignarle una sala, una fecha ni un horario de la función. Debería permitirte grabar sin problemas.

Ahora revierte los cambios hechos para dejarlo como estaba anteriormente.

CONSECUENCIAS DE AGREGAR UN ATRIBUTO A UNA TABLA QUE YA TIENE DATOS [SUGERIDO]

Otro requisito es agregar un catálogo de empleados. Agrega la siguiente transacción a la KB.

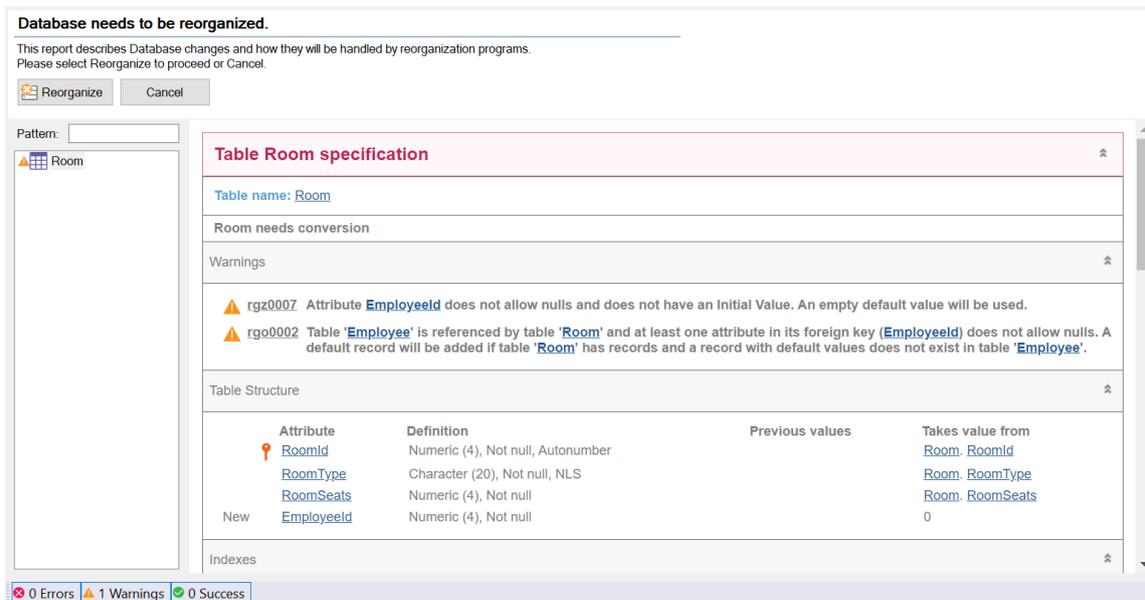


Ejecuta la aplicación y agrega un par de registros para el empleado.

Nos solicitan el siguiente cambio: se requiere que para cada sala haya un empleado encargado y cada empleado puede estar a cargo de una o más salas. **¿Cómo modelarías esto en GeneXus teniendo en cuenta las transacciones Employee y Room?**

Replica en GeneXus lo que acabas de analizar.

El análisis de impacto nos indica lo siguiente. **¿Qué pasará en la tabla de Empleado si presionamos Reorganize?**



Si no queremos que suceda esto que nos informarn los warnings, ¿qué podemos hacer?

TRANSACCIONES PARALELAS [SUGERIDO]

Las compañías cinematográficas comparten cierta información de los pre-estrenos con todos los Cinemas. Se comparte: El nombre de la película, el año, un póster, director, país y su categoría; los datos de la descripción y los actores son confidenciales hasta cierto tiempo después, por lo cual en el sistema no se colocan hasta que la película se exhiba en Cartelera.

Para registrar los datos iniciales de la película y no ver los otros campos, se requiere de una transacción paralela donde solamente se registre la información con la que se cuenta.

Realiza esta implementación.

Recuerda que ya habíamos definido una transacción paralela antes: era MovieCopy1, paralela de Movie. Lo habíamos hecho para tener una sin el pattern (sin parámetros ni todo lo que le agregaba el pattern a Movie).

SUBTIPOS

SUBTIPOS: REFERENCIA MÚLTIPLE Y ESPECIALIZACIÓN

Se requieren las siguientes mejoras en la aplicación:

Recordemos que una película tiene un país, y que puede tener 1 o más actores asociados a ella. Se requiere agregar, también para sus actores, la información de su país.

¿Qué debemos cambiar en nuestra estructura de transacciones para guardar el País de la Película y también hacer referencia al País del Actor? Piensa en 3 diseños diferentes que resuelvan este requerimiento. Escribe para cada uno sus ventajas y desventajas (si las tienen), y escoge el que consideres más adecuado para implementarlo en GeneXus.

Luego del análisis realiza la implementación en la aplicación que venimos desarrollando.

Ten en cuenta, que con la implementación más conveniente deberás también realizar cambios en el DP de la transacción Movie. Así cómo también en las transacciones MovieCopy1 y MoviePreview.

DISEÑO 1:

VENTAJAS	DESVENTAJAS

DISEÑO 2:

VENTAJAS	DESVENTAJAS

DISEÑO 3:

VENTAJAS	DESVENTAJAS

Los Empleados del Cinema pueden ser categorizados en:

- **Administrativos:** se encargan de la venta de entradas y atención al público en general. De este tipo de empleados interesa conocer si hablan algún idioma adicional.
- **Operarios:** se encargan de programar las películas en las diferentes salas en los días y horarios en las que se proyectan. Estos empleados deben ser mayores de 22 años, por lo cual interesa conocer su edad.
- **Técnicos:** se encargan del mantenimiento del Cinema en lo que tiene que ver con reparaciones de maquinaria, instalaciones, etc. De este tipo de empleados interesa conocer su formación técnica relevante al cargo.

¿Cómo podemos hacer esta distinción en GeneXus? Impleméntalo.

MÁS CASOS DE USO DE SUBTIPOS

Se solicita una transacción donde podamos ver la información de los Empleados del cine y su jefe a cargo. ¿Cómo debe modelarse si se toma en cuenta que el jefe también es un empleado?

En la transacción de Empleado se deberá mostrar un error si intentamos asignar como jefe a un empleado que no tiene ese rango, y tampoco debe permitir guardar la información si el jefe y el empleado son la misma persona.

¿Cómo se deben programar esas reglas?

Realiza la implementación en GeneXus y escriba también las reglas para probar el comportamiento.

```
error("The employee and the Manager can't be the same")  
  if _____ = _____;
```

```
error("The employee is not a Manager")  
  if _____ and _____;
```

The screenshot shows a form titled "Employee" with the following fields and values:

Id	0
Name	John Smith
Photo	[Empty box with edit and delete icons]
Is Manager	<input type="checkbox"/>
Employee Id	[Input field with value 2 and a red error message: "The employee is not a Manager"]
Employee Name	Billie Robin
Is Manager	<input type="checkbox"/>

FÓRMULAS

FÓRMULAS VERSUS REGLAS

Vuelve sobre el control de los puntos que el cliente dispone para canjear por premios.

Name	Type	Description	Formula	Nullable
Customer	Customer	Customer		
CustomerId	Id	Customer Id		No
CustomerName	Name	Customer Name		No
CustomerEmail	Email, GeneXus	Customer Email		No
CustomerVIP	Boolean	Customer VIP		No
CustomerPoints	Numeric(4.0)	Customer Points		No
CustomerLastRewardId	Id	Customer Last Reward Id		No
Reward	Reward	Reward		
CustomerRewardId	Id	Customer Reward Id		No
CustomerRewardDescription	Numeric(4.0)	Customer Reward Description		No
CustomerRewardPoints	Numeric(4.0)	Customer Reward Points		No

¿Es posible resolverlo utilizando fórmulas globales en lugar de las reglas add y subtract?

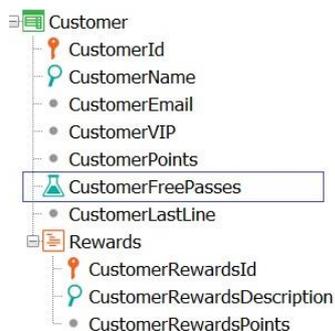
¿Obtenemos el mismo comportamiento?

FÓRMULAS HORIZONTALES (OPCIONAL)

Antes de continuar probando se solicita un nuevo requerimiento:

Los clientes pueden obtener pases libres (CustomerFreePasses) en el mes actual, dependiendo de la cantidad de tickets registrados:

- Si registra en el mes más de 2 y menos de 6 tickets, genera 1 pase libre.
- Si registra en el mes más de 5 y menos de 9 tickets, genera 2 pases libres.
- Si registra en el mes más de 8 tickets, genera 3 pases libres.



¿Qué definición harías para el atributo fórmula CustomerFreePasses?

Una opción puede ser la siguiente:

```

Formula Editor
1 IF count(TicketDate, TicketDate.month()=month(today()))>2 and count(TicketDate, TicketDate.month()=month(today())) < 6;
2 IF count(TicketDate, TicketDate.month()=month(today()))>5 and count(TicketDate, TicketDate.month()=month(today())) < 9;
3 IF count(TicketDate, TicketDate.month()=month(today())) > 8;
    
```

Otra posibilidad es definir un procedimiento FreePasses que devuelva el valor correspondiente. Implementa esta opción.

¿Cómo será su regla Parm?

¿Cómo será su invocación para que el atributo CustomerFreePasses tome el valor devuelto por el procedimiento?

FÓRMULAS AGGREGATE

Anteriormente el precio de los boletos era definido en el momento de la compra, sin embargo, ahora el precio será definido de acuerdo a la sala que se esté seleccionando. Para esto, se requiere guardar un historial de precios para la sala y la fecha en la que se guardó dicho precio (NOTA: la fecha no debe poder repetirse) y el valor que será utilizado en el cálculo del ticket será el de la fecha más actualizada.

De acuerdo a lo anterior, ¿cuál sería la opción correcta para modelar el requisito?



¿Qué fórmula de agregación usarías para retornar el valor del precio de acuerdo a la fecha más actualizada?

RoomUpdatedPrice = _____

Una vez actualizada la transacción Room, ¿Qué cambios debes realizar en Ticket? ¿Ticket Price sigue siendo válido?

Realiza los cambios necesarios (en la trn Room y en la trn Ticket) y pruébalo en ejecución.

Dentro de la transacción Movie se desea agregar un campo que permita saber si la película es Premium o no.

En caso de que la película sea Premium, se deberá agregar al momento de la compra un recargo del 7% de su valor y si la película no tiene este campo especificado, entonces deberá tomar el precio normal.

Room Type	VIP
Movie Id	1
Movie Name	Avengers End Game
Seats Quantity	<input type="text" value="1"/>
Room Updated Price	135.00
Total Price	144.45
Points	14

Se desea conocer para cada registro de Cartelera los boletos vendidos y los asientos disponibles.

UNANIMO

- Actors
- Billboards
- Categories
- Countries
- Customers
- Film Directors
- Candy Stores
- Cinema Products
- Employees
- Movie

Billboards

Room Id	Movie Date	Movie Schedule	Movie Id	Movie Name	Movie Poster	Room Seats	Seats Available	Seats Sell		
1	12/15/22	11 AM	2	Avengers: Endgame		50	50	0	UPDATE	DELETE
4	12/15/22	5 PM	4	Avatar: Way to water		30	2	28	UPDATE	DELETE
6	12/15/22	8 PM	8	Unofficial Mission		50	50	0	UPDATE	DELETE
5	12/19/22	5 PM	4	Avatar: Way to water		40	37	3	UPDATE	DELETE
1	12/19/22	11 AM	5	Forest Gump		50	50	0	UPDATE	DELETE

En caso de que ya se hayan vendido todos los boletos para dicha sala, el sistema no debe permitir guardar la compra.

TRANSACCIONES DINÁMICAS

Otro de los requerimientos del sistema de Cinemas es unir la información y tener un catálogo de las películas y los productos que se venden en dulcería. Se guardará: el identificador del producto, su nombre, una imagen y su precio.

A fin de conocer todos los productos ofrecidos por el Cinema (películas y golosinas), se requiere de un reporte donde se listen todas las películas y los productos de dulcería. Se necesita ver: su identificador, su nombre, una foto y el tipo (si es película o artículo de dulcería).

Sugerencia

Utiliza una transacción dinámica para recuperar la información de ambos catálogos y poder listarla en el reporte anteriormente solicitado.



All Products

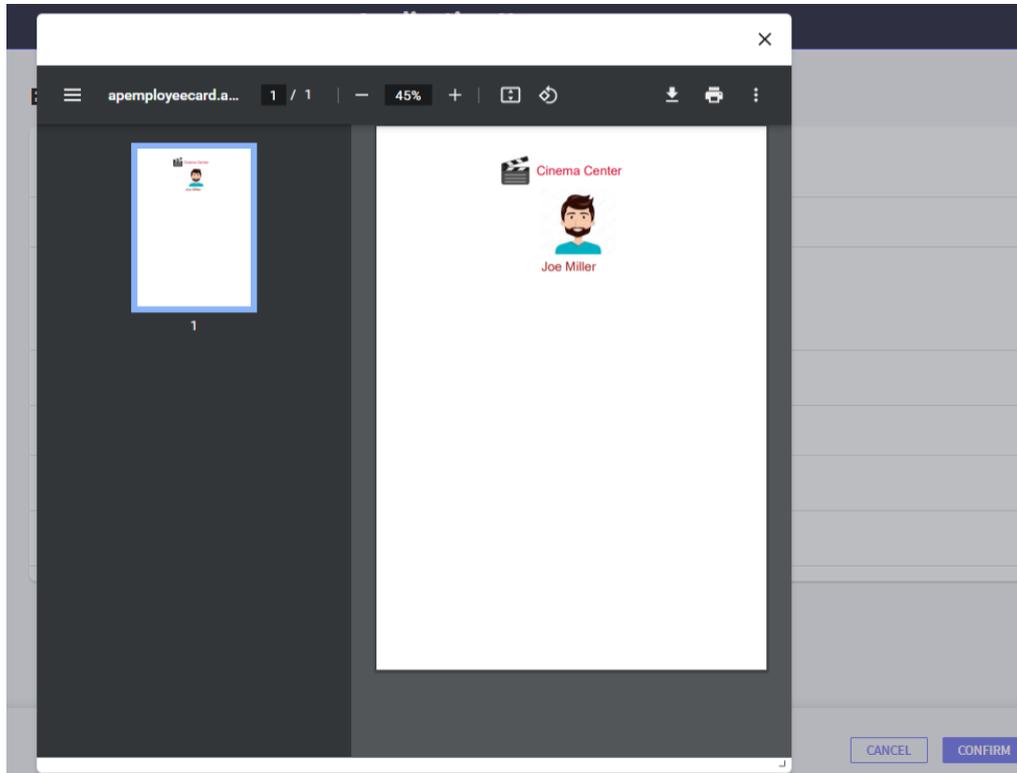
Id	Type	Product Name	Photo
1	Candy	Popconm	
1	Movie	Avengers End Game	
2	Candy	Soda	
2	Movie	Catch me if you can	
3	Candy	Gummies	
3	Movie	Forrest Gump	
4	Candy	Hot Dog	

Primero que nada, se deberá crear una transacción (CandyStoreProduct) donde se pueda ingresar, modificar y eliminar las golosinas que se venderán. De estas nos interesa registrar su nombre, una foto y su precio.

¿Qué modificaciones deberías realizar a la implementación propuesta si el usuario final deseara poder ingresar, modificar o eliminar películas y golosinas a través del form de una nueva transacción la cual será dinámica?

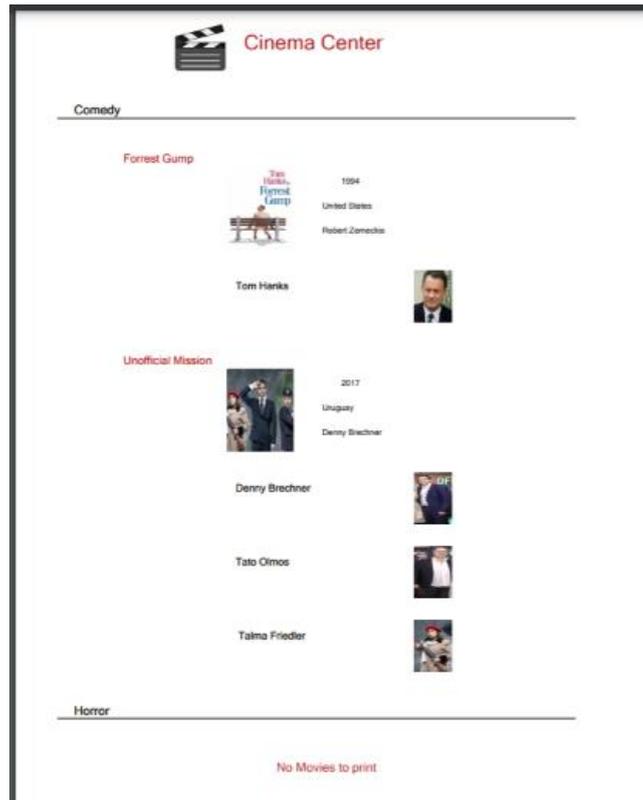
EVENTOS EN TRANSACCIONES

El sistema de cines requiere que al momento de guardar un empleado nuevo se vea en pantalla un Popup con su información a estilo de credencial:



MÁS SOBRE FOR EACHS ANIDADOS. CASOS Y NAVEGACIÓN

Nos solicitan un reporte que cuente con el siguiente formato. Colocar las categorías y sus películas relacionadas, así como los actores que participan en dichas películas. En caso de que no haya películas asociadas a la categoría deberá desplegar un mensaje informando que no hay información:

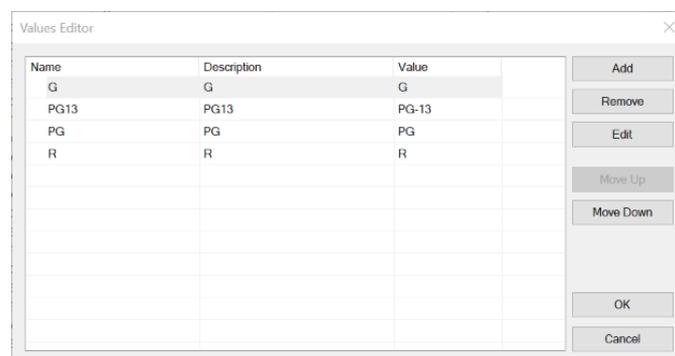


Se nos solicita un listado con el formato del anterior con la única restricción de que solamente se listen las categorías que cuentan con películas asociadas. ¿Cómo lograrías lo anterior? Escribe el código a continuación con los cambios que debes hacer.

Realiza un Save As del primer procedimiento y haz las modificaciones adecuadas para poder recibir como parámetro una Categoría desde un web panel. El procedimiento debe mostrar la información de la categoría seleccionada y en caso de que no se seleccione ninguna categoría, se debe mostrar toda la información de las Categorías con sus Películas y Actores correspondientes.

SUBRUTINAS

En la transacción Movie agrega un nuevo atributo llamado MovieRating donde guardaremos la clasificación de edades para cada película (G, PG, PG-13, R). Para esto haz un dominio enumerado llamado Rating que contendrá los siguientes datos:

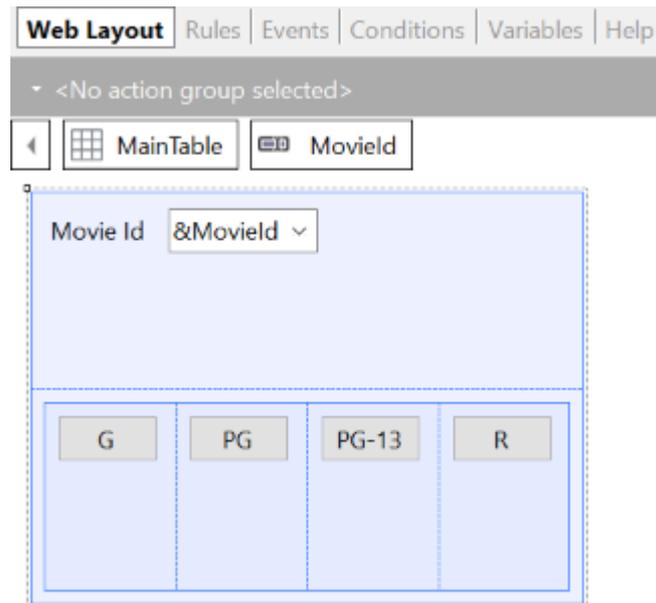


Name	Description	Value
G	G	G
PG13	PG13	PG-13
PG	PG	PG
R	R	R

Sobre los dominios enumerados:

En las columnas "Name" y "Description" no se permiten nombres que contengan guiones u otros caracteres especiales. Para colocar el nombre de la clasificación PG-13, hacerlo sin el guion.

A continuación, elabora el siguiente web panel donde seleccionaremos una película y dando clic a un botón actualizaremos su clasificación de acuerdo al valor seleccionado.



Observa que cierto código deberá repetirse para cada botón, por lo tanto podemos hacer uso de las subrutinas para usar el mismo código pero actualizar según lo seleccionado.

```
1 Event 'G'  
2     &Rating = Rating.G  
3     do "UpdateRating"  
4 Endevent
```

Sugerencia

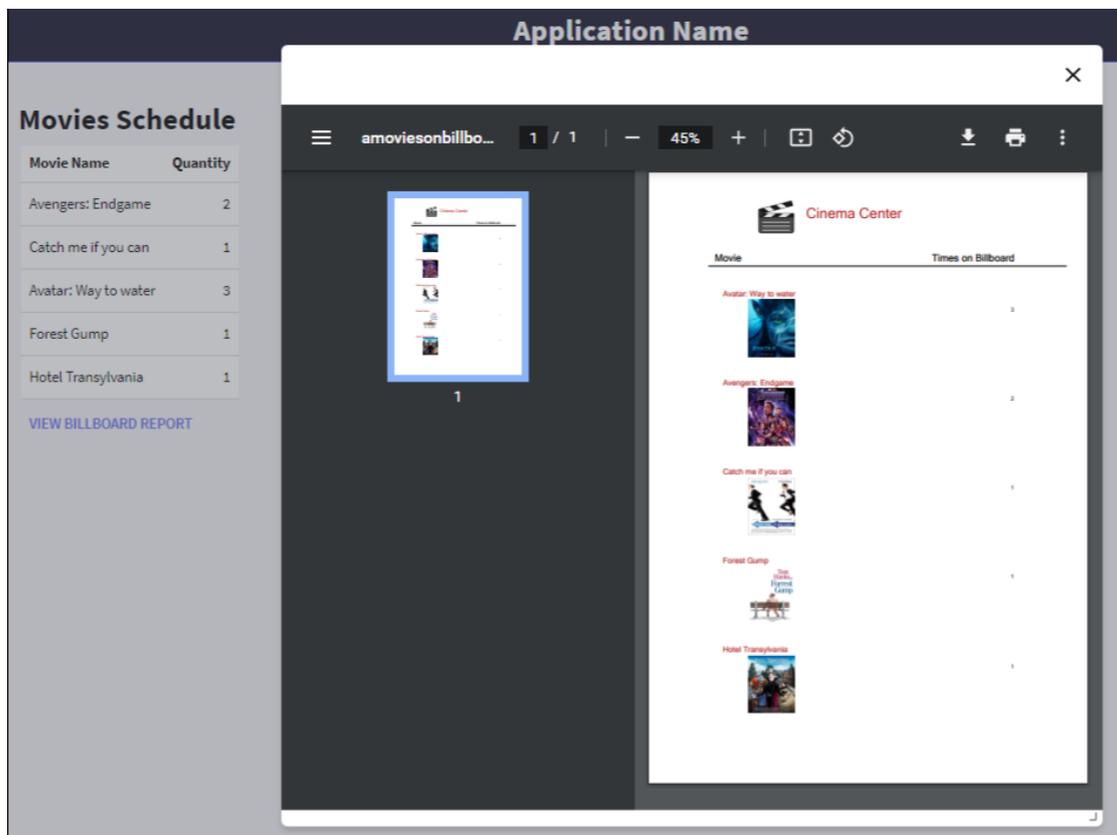
Utiliza una tabla responsiva para colocar los botones y cambia la clase de los botones por una que vaya con la estética del programa, ejemplo: BTN Cancel.

CLÁUSULA UNIQUE

Otro requerimiento del Cinema es poder visualizar un reporte de las películas que aparecen en cartelera, Schedule, sin repeticiones y con la cantidad de apariciones que han tenido en cartelera.



Un nuevo requerimiento es poder visualizar la información anterior en un web panel. Coloca un botón donde puedas consultar el listado anterior como PopUp.



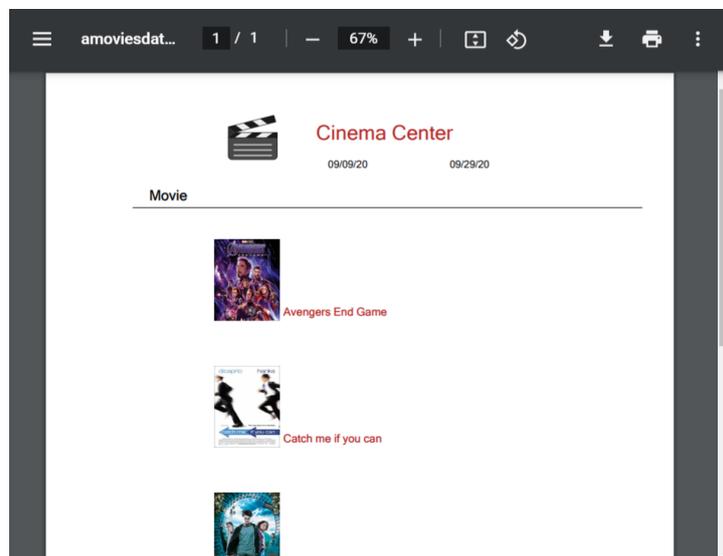
Sugerencia

Cambia las clases del Grid por WorkWith y la del Botón por Btn Cancel para dar más estética a la aplicación.

DATA SELECTOR

Se requiere de una pantalla web que permita seleccionar un rango de fechas y que con un botón se muestre como popup el listado de todas las Películas que aparecen en cartelera, pero solamente las películas que estén en el rango de fecha seleccionado.

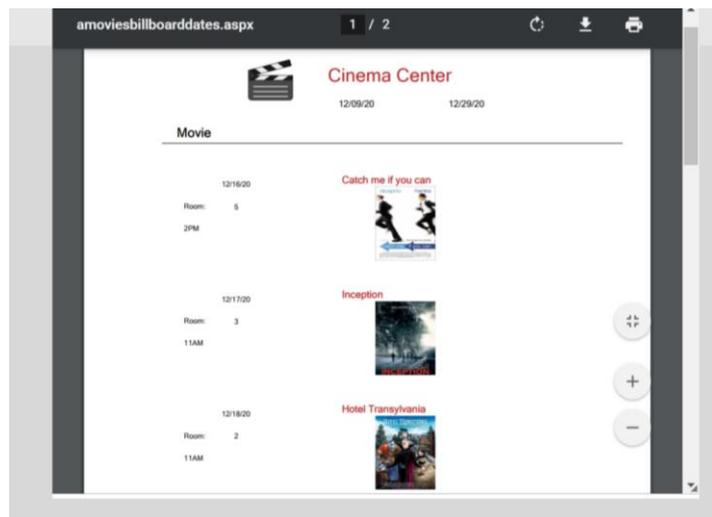
¿Cuál es la navegación resultante? Explícalo...



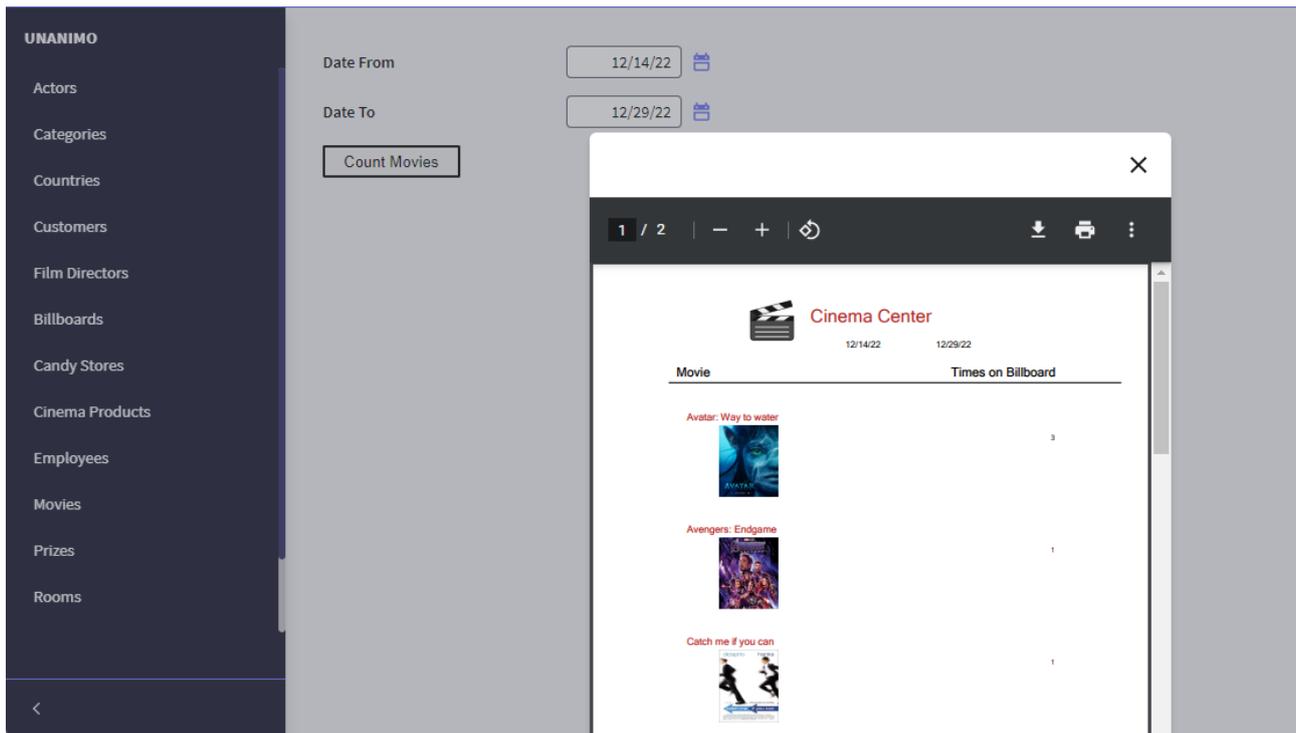
NOTA: Como es una consulta que recurrentemente se va a realizar dentro del sistema de Cinemas, genera un nuevo objeto data selector para poder usarlo después.

En la misma pantalla web se desea generar otro listado popup donde se consulte la información de las películas que se van a presentar, su horario, la sala y la fecha, en un rango determinado de fechas.

¿Cuál es la diferencia entre esta navegación resultante y la anterior? Explícalo...



Adicional, debes permitir visualizar un listado Pop up con las películas que aparecen en cartelera, sin repeticiones y contando solamente la cantidad de apariciones que han tenido en cartelera en dicho rango de fechas.



¿Cuál es la diferencia entre esta navegación resultante y las anteriores? Explícalo...

CLÁUSULAS DE PAGINADO

En la pantalla trabajar con asociada a la Cartelera me interesa solamente ver 5 registros a la vez, además de que el formato de salida sea un scroll infinito. Aplica el cambio solicitado.

PAGINADO EN FOR EACH, PATTERN Y GRID (OPCIONAL)

Se requiere una pantalla donde el usuario pueda llamar un listado con la información de los actores. En particular, en la pantalla debe seleccionar cuántos registros desea omitir y cuántos pueda visualizar después. Ejemplo: “deseo omitir los primeros 3 registros y poder trabajar con los siguientes 5 para trabajar puntualmente con esos datos.”



List of Actors

4	Daisy Ridley	United Kingdom	
5	Emma Watson	United Kingdom	
6	Robert Downey Jr	United States	
7	Chris Hemsworth	Australia	
8	Adam Sadler	United States	

BUSINESS COMPONENT

Realizaremos una serie de operaciones sobre la Base de Datos, a través de Business Component.

ACTUALIZACIÓN DE ENCARGADOS

Se requiere de una pantalla especializada que permita seleccionar un tipo de sala (VIP, 3D, etc) y seleccionar un empleado. El empleado se hará cargo de todas las salas que sean de ese tipo.

Room Type

Employee Id

TRABAJANDO CON LAS PELÍCULAS Y SUS ACTORES

Se solicita una pantalla con formato especial que nos permita seleccionar una película para poder trabajar con sus actores. Se ofrecerán dos botones:

- Uno que nos permita modificar el rol del actor a estelar o quitar el rol estelar.
- Otro que permita eliminar al actor de la película seleccionada.

Sugerencia

Utiliza la variable &Messages de GeneXus para programar los mensajes del sistema en caso de que la actualización falle o sea exitosa.

¿Qué podrías hacer para que en la variable asociada al actor solamente se vean los actores que participan en la película seleccionada?

Application Name

▲ Data has been successfully updated.

Movie Id: Avengers: Endgame

Actor Id: Robert Downey Jr

Is Starring:

UPDATE ACTOR DELETE ACTOR

ACTUALIZACIÓN DE LA BASE DE DATOS CON PROCEDIMIENTO

En la misma pantalla donde se hizo la actualización de los empleados encargados de las salas de acuerdo a su tipo, implementa un botón que realice la misma actualización antes mencionada pero ahora usando un procedimiento.

¿Cómo podría borrar las películas y sus actores asociados a través de un procedimiento?

Inserta una película nueva (la que se desees) con sus respectivos actores.

Programa un botón en la pantalla que creaste anteriormente para trabajar con las películas y todos sus actores, dicho botón debe eliminar una película seleccionada y eliminar los actores asociados. Elimina la Película que acaba de crear.

¿Qué sucede cuando intenta eliminar Películas que tienen relación con otras tablas?

PANTALLAS WEB: WEB PANELS

A continuación, comenzaremos a generar pantallas interactivas personalizadas que nos han sido solicitadas para el sistema de Cinemas.

VARIABLES EN GRID Y ACCIONES EN EL PATTERN WORK WITH (OPCIONAL)

En la pantalla que nos permite trabajar con las películas nos solicitan los siguientes cambios:

- No deben visualizarse los Id
- Un campo calculado que nos muestre cuántos actores estelares tiene la película.
- Un botón que nos muestre el listado de la información de las películas anteriormente creado donde se visualiza la Categoría, la película y los actores que la conforman.

GeneXus Application Name

INCRCT Info Movies

Q Name

Movies

Name	Year	Poster	Description	Film Director Name	Country Name	Category Name	Starring Actors	Premium Movie	Rating	
Avatar: Way to water	2022		Jelle Sully and Neytiri have formed a family and are doing everything to stay together. However, they must leave their home and explore the regions of Pandora. When an ancient threat resurfaces, Jake must fight a difficult war against the humans.	James Cameron	United States	Fantasy	0	<input checked="" type="checkbox"/>	PG13	UPDATE DELETE
Avengers: Endgame	2019		After Thanos, an intergalactic warlord, disintegrates half of the universe, the survivors must reunite and assemble again to reimagine their fabled allies and restore balance.	Joe Russo	United States	Action	2	<input checked="" type="checkbox"/>	PG	UPDATE DELETE
Catch me if you can	2002		Notorious con artist Frank Abagnate has duped people worth millions of dollars with his masterful art of deception. With his seams getting tighter, he is soon pursued by FBI agent Carl Hanratty.	Steven Spielberg	United States	Comedy	2	<input type="checkbox"/>	PG13	UPDATE DELETE
Forrest Gump	1994		Forrest, a man with low IQ, recounts the early years of his life when he found himself in the middle of key historical events. All he wants now is to be reunited with his childhood sweetheart, Jenny.	Robert Zemeckis	United States	Drama	1	<input type="checkbox"/>	PG	UPDATE DELETE
Hotel Transylvania	2012		Count Dracula runs a high-end resort for monsters and is overprotective of his daughter, Mavis. When a human named Johnny enters the hotel, Dracula tries to prevent Mavis from falling for him.	Genndy Tartakovsky	Australia	Children	1	<input type="checkbox"/>	G	UPDATE DELETE

CREANDO PANTALLAS INTERACTIVAS

- Se solicita una pantalla donde se vea la información de la Cartelera ordenada por la fecha de exhibición. Se debe permitir filtrar por:
 - La fecha, que sea mayor o igual al día de hoy. (El sistema debe sugerir en automático tomar la fecha de hoy).
 - Por horario.
 - Por tipo de Sala.
 - Por nombre de la película.

Si los filtros se encuentran vacíos, se deberá ver toda la información de la cartelera.

Para cada registro se requiere de un botón que al darle clic nos muestre el detalle de la película seleccionada. En la pantalla del detalle, implementa un botón que nos permita regresar al web panel anterior.

GeneXus™ **Application Name**

Room Movie Date 

Room Movie Schedule 

Room Type 

Movie Name

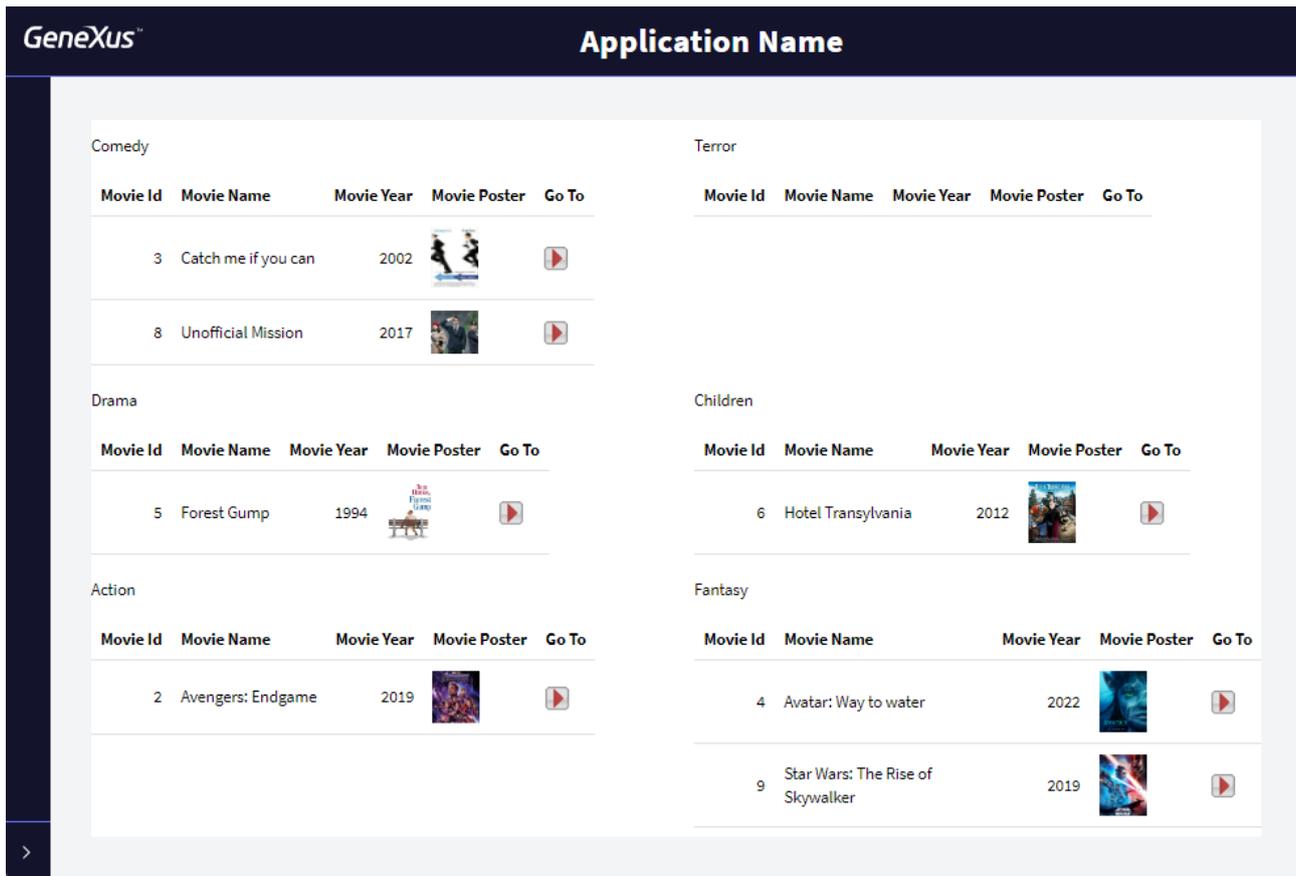
Room Type	Room Movie Date	Room Movie Schedule	Movie Name	Movie Poster	Goto
VIP	12/15/22	5 PM	Avatar: Way to water		
VIP	12/19/22	5 PM	Avatar: Way to water		
Normal	12/21/22	5 PM	Hotel Transylvania		

MÚLTIPLES GRID

Se solicita un Web Panel que permita ver todas las Categorías y por cada una de ellas sus películas asociadas.

Para cada película se requiere de un botón que al darle clic nos muestre el detalle de la película seleccionada. En la pantalla del detalle, implementa un botón que nos permita regresar al web panel anterior.

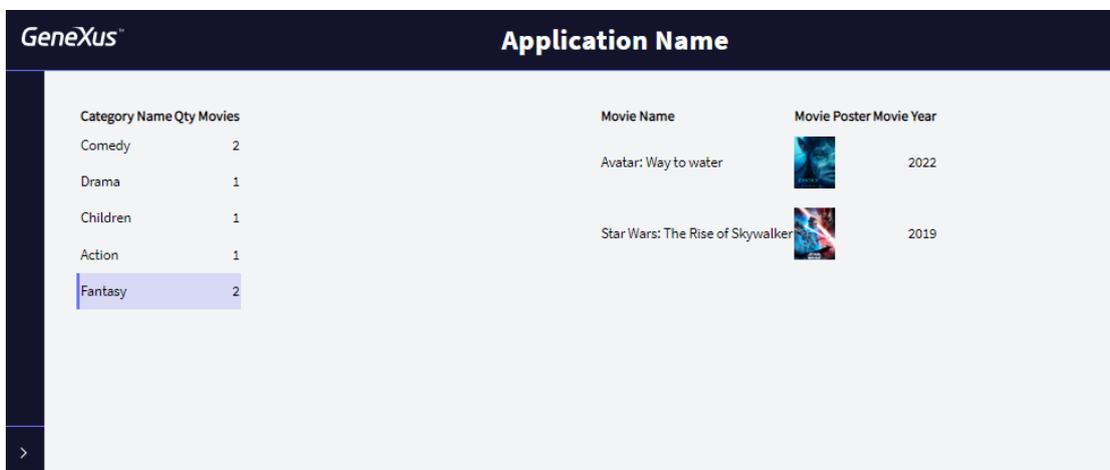
Nota: Se puede reutilizar el web panel del detalle de la película.



¿Cómo podrías asegurar que solamente se vieran las categorías que tienen películas asociadas?

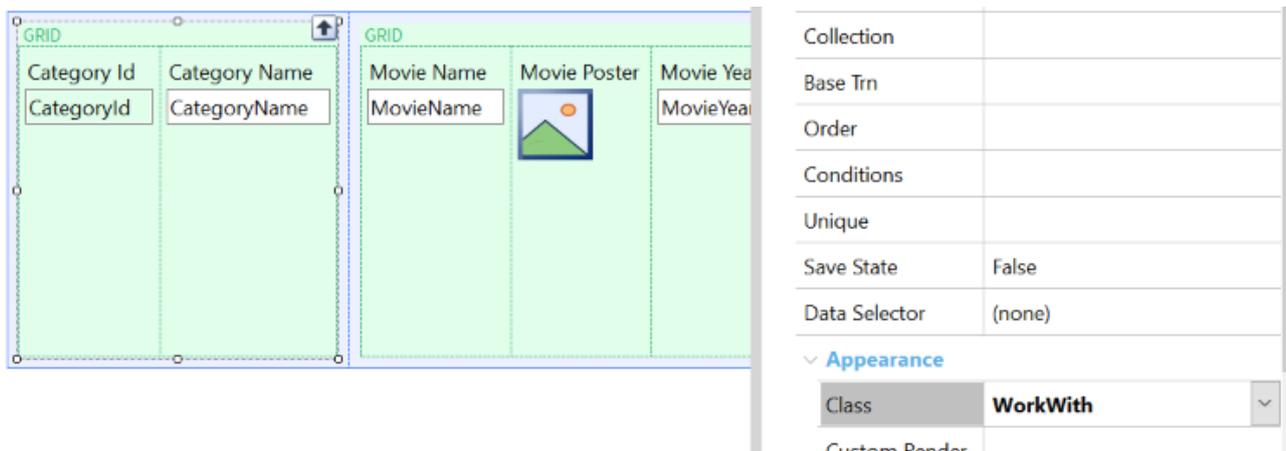
Implementa ese cambio.

Se requiere de una pantalla donde se muestre la lista de las categorías que tienen películas asociadas (sin repeticiones) y su cantidad de películas, a su vez se podrá seleccionar una categoría y a continuación se visualizarán las películas asociadas en la misma pantalla.



¿Cuál debe ser la tabla base del Grid que muestra la información de las Categorías?

Nota: Se recomienda que se aplique la clase WorkWith al Grid para mejorar la estética



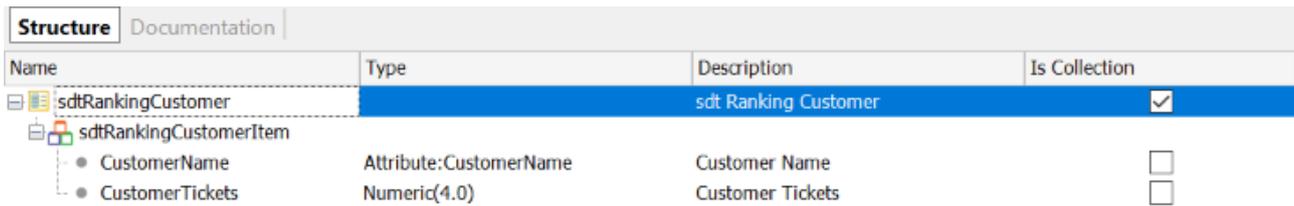
GRID SIN TABLA BASE

Se requiere de una pantalla especial que nos muestre por cada fecha las películas a presentar. El formato es el siguiente:



CARGANDO COLECCIONES

Se requiere de un ranking de Clientes de acuerdo a los asientos de cada ticket que hayan comprado. Para resolver lo anterior utiliza una estructura.



Name	Type	Description	Is Collection
sdtRankingCustomer		sdt Ranking Customer	<input checked="" type="checkbox"/>
sdtRankingCustomerItem			
CustomerName	Attribute:CustomerName	Customer Name	<input type="checkbox"/>
CustomerTickets	Numeric(4.0)	Customer Tickets	<input type="checkbox"/>

Además de lo anterior, la pantalla debe permitir que al seleccionar un cliente muestre la información en formato PDF del cliente con su respectiva cantidad de tickets.

Sugerencia

Utiliza la propiedad AllowSellection del Grid y el evento OnLineActivate.

GUARDAR INFORMACIÓN DE CONTEXTO (OPCIONAL)

Abre el web panel que contiene la información completa de la Cartelera. Se requiere que al momento de regresar a dicha pantalla después de ver el detalle de la película se guarden los valores de los filtros.

PANTALLA ESPECIAL PARA LA VENTA DE TICKETS

Realiza una copia de la pantalla anterior y modifica las clases del web panel para que tenga una estética parecida a la siguiente:

Room Type	Movie Date	Movie Schedule	Movie Name	Movie Poster
Normal	12/15/22 11 AM		Avengers: Endgame	
VIP	12/15/22 5 PM		Avatar: Way to water	
3D	12/15/22 8 PM		Unofficial Mission	
VIP	12/19/22 5 PM		Avatar: Way to water	
Normal	12/19/22 11 AM		Forest Gump	
Normal	12/20/22 2 PM		Catch me if you can	
Normal	12/21/22 5 PM		Hotel Transylvania	

Sugerencia

Utiliza la clase `WorkWith` del Grid y las clases `WWColumn` y `WWOptionalColumn`.

Al seleccionar una línea de la Cartelera se debe desplegar un pop up que nos permita ver los datos de la película seleccionada. El Empleado podrá seleccionar un cliente o no y el número de asientos a comprar; al dar clic en el botón Buy se deberá realizar la compra. Nota: hay ocasiones en las que no se podrá realizar la compra debido a algún control que tengamos programado, así que se deberán ver los mensajes del sistema en pantalla:

GeneXus Application Name

Date: 12/15/22
Schedule: 5 PM
Room Type: Select Room Type
Movie Name: [input field]

Room Type	Movie	Date	Movie Schedule	Movie Name	Movie Poster
VIP	12/15/22 5 PM	Avatar: Way to water	[Poster]		
VIP	12/19/22 5 PM	Avatar: Way to water	[Poster]		
Normal	12/21/22 5 PM	Hotel Transylvania	[Poster]		
VIP	12/31/22 5 PM	Avatar: Way to water	[Poster]		

Buy Tickets

⚠ Not enough seats

Room	Date	Schedule	Movie Name	Seats Available
4	12/15/22	5 PM	Avatar: Way to water	2

Customer Id: Anne Parker

Ticket Seats Quantity: [1, 2, 3, 4, 5, 6]

BUY

GeneXus Application Name

Date: 12/15/22
Schedule: 5 PM
Room Type: Select Room Type
Movie Name: [input field]

Room Type	Movie	Date	Movie Schedule	Movie Name	Movie Poster
VIP	12/15/22 5 PM	Avatar: Way to water	[Poster]		
VIP	12/19/22 5 PM	Avatar: Way to water	[Poster]		
Normal	12/21/22 5 PM	Hotel Transylvania	[Poster]		
VIP	12/31/22 5 PM	Avatar: Way to water	[Poster]		

Buy Tickets

⚠ Data has been successfully added.

Room	Date	Schedule	Movie Name	Seats Available
5	12/19/22	5 PM	Avatar: Way to water	40

Customer Id: Anne Parker

Ticket Seats Quantity: [1, 2, 3, 4, 5, 6]

BUY

PANTALLAS MULTI-CHANNEL: PANELS

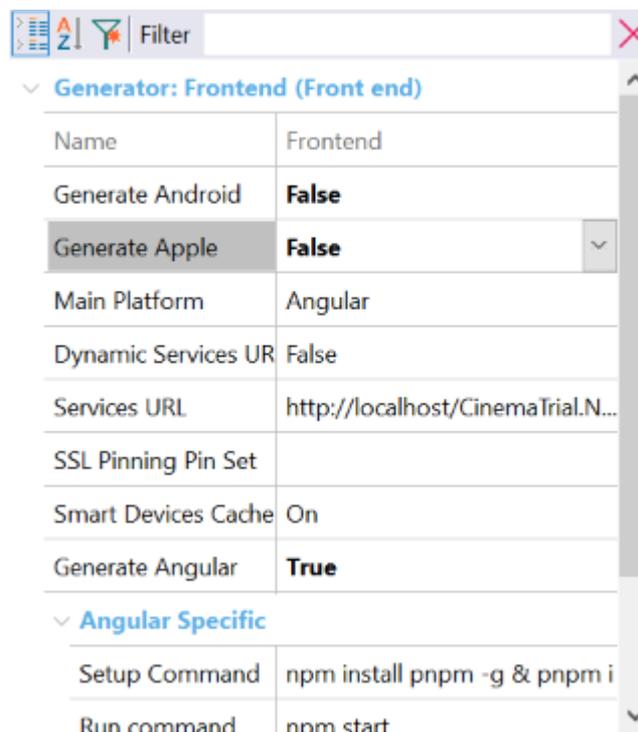
Para realizar el siguiente ejercicio necesitará verificar que cuenta con los [requisitos necesarios para generar en Angular](#).

Se requiere de un Panel generado con Angular que contenga la información de la cartelera

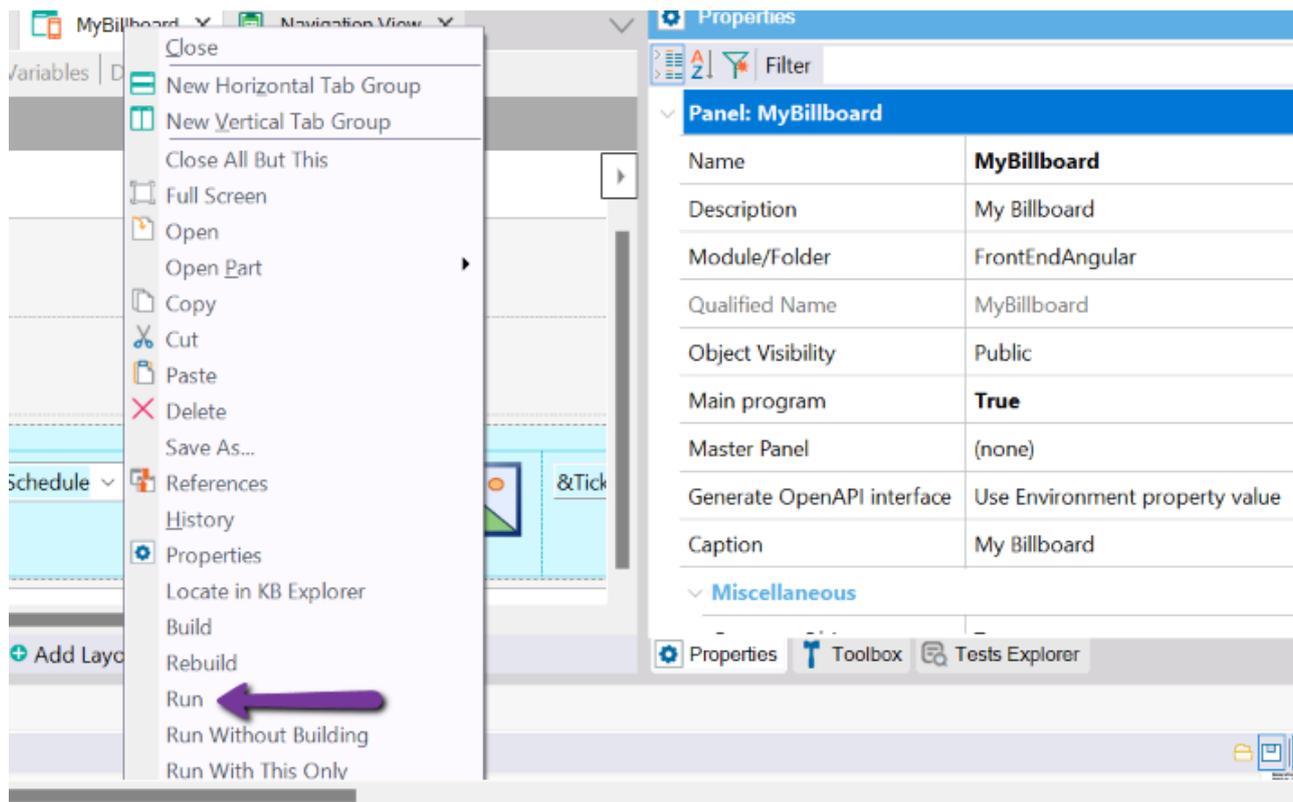
Para solucionar lo anterior, de Ctrl+n y seleccione el objeto Panel. En el panel arrastra un Grid y selecciona la fecha de la película, el horario, la sala, el nombre de la película y el poster.

Coloca la propiedad Autogrow del Grid en True.

Una vez guardado el objeto panel, para comenzar a ejecutar en Angular, vaya a *FrontEnd* en el *KBExplorer*. Cambia la propiedad *Generate Angular* en *True* y como por el momento no vamos a desarrollar para aplicaciones móviles, cambia las propiedades *Generate Android* y *Generate Apple* a *False*.



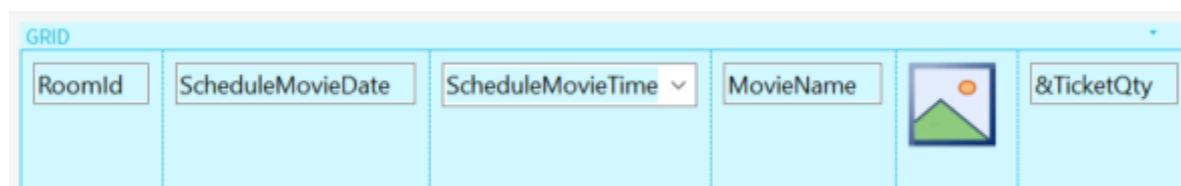
En las propiedades del panel cambia la propiedad Main Program por True, haz clic derecho en el nombre del objeto y clic en Run para ejecutar la aplicación.



Verifica que la aplicación se vea de esta manera.

3	7/5/2021	5 PM	Hotel Transylvania	
4	6/5/2021	11 AM	Avengers End Game	
6	30/4/2021	8 PM	Inception	

Ahora agrega una variable que calcule el número de ventas emitidas por Cartelera. *Nota: tenemos un atributo que calcula el número de asientos vendidos, no queremos mostrar esa información, sino el número de tickets o ventas.*



Programa lo necesario en la sección de eventos y prueba el funcionamiento de la aplicación.

Nos interesa ver que la información aparezca ordenada por fecha y programar filtros por el nombre de la película y que muestre las exhibiciones a partir de una fecha

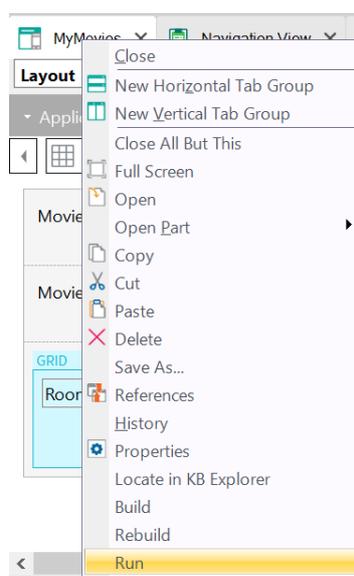
Movie Date	04/05/2021							
Movie Name	av							
6/5/2021	11 AM	4	Avengers End Game		3			

PROBANDO LA APLICACIÓN EN SMART DEVICES

Cambia la propiedad Generate Android en True para comenzar a probar la aplicación en Smart Devices y verifica que la propiedad Main Platform se cambie a Android.

Generators	
Generate Android	True
Generate Huawei	False
Generate Apple	False
Generate Angular	False

Ahora da clic derecho en el nombre del panel y ejecuta la aplicación. Deberá desplegarse una pantalla en Android que será la misma generada anteriormente para web.



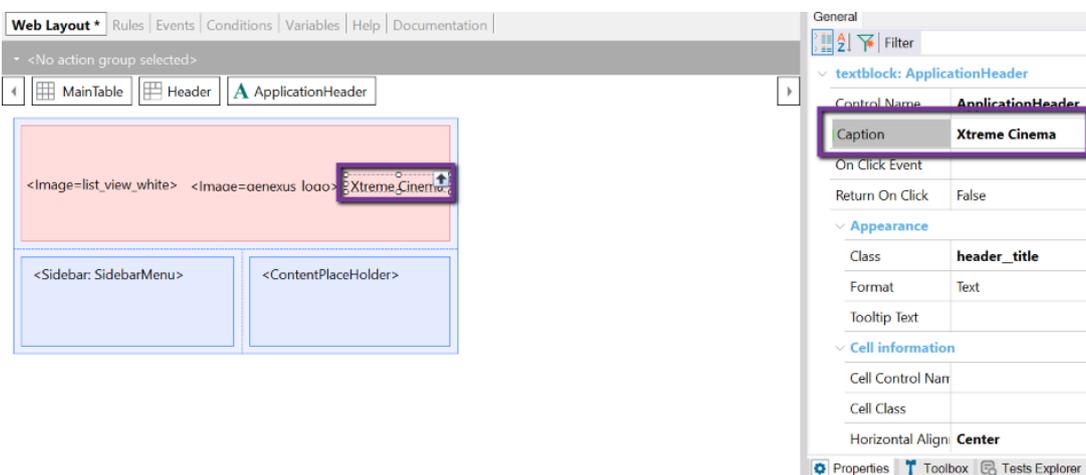
APLICACIONES WEB PROGRESIVAS [SUGERIDO]

Vamos a cambiar un poco la estética de la aplicación para que nuestra PWA se vea mejor diseñada. Dirígete al KB Explorer y busca en el filtro “RWD Master Page”. Esta es la Master Page asociada a todos los objetos GeneXus de nuestra KB.

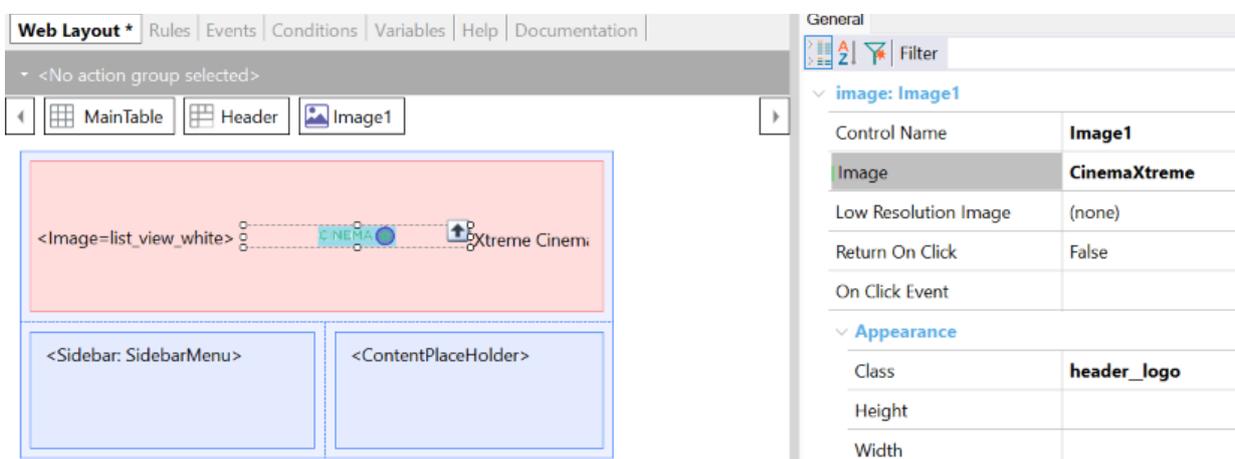
Observa que cuenta con tres secciones:

- Un Header donde podemos localizar el nombre de la aplicación y el logotipo.
- Un objeto web component que es el encargado de ofrecernos el menú de los links recientes.
- Un content place holder donde se despliegan los objetos de nuestra aplicación.

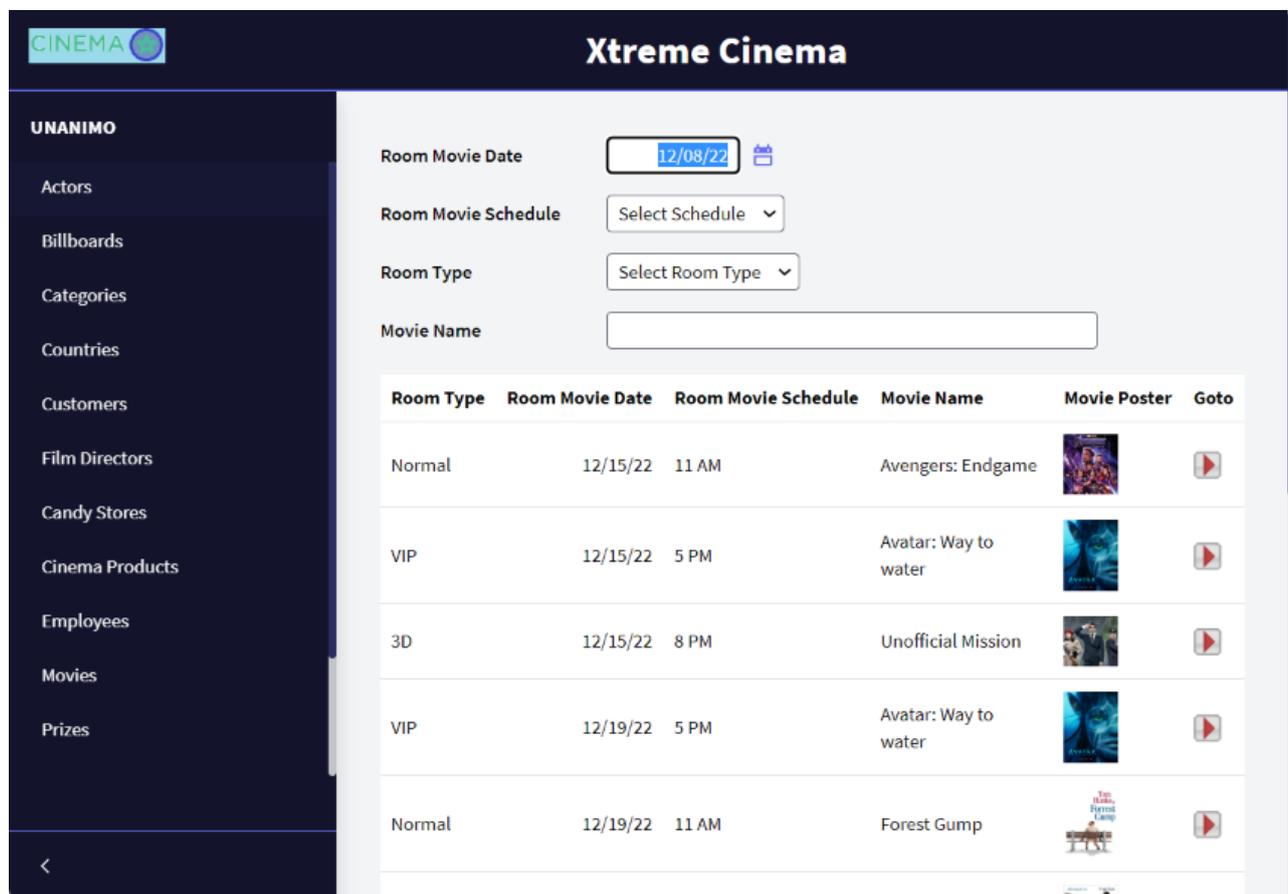
En el *Header*, selecciona el control *ApplicationHeader* y cambia la propiedad *caption* por el nombre de la aplicación: My Cinema.



Selecciona el control *Image1* y en la propiedad *Image* selecciona una imagen como logotipo de la empresa, puedes auxiliarte de las imágenes destinadas al práctico. Verifica que tenga la clase *PoweredBy* aplicada.



Realiza una copia del web panel donde se muestra toda la información de la cartelera:



The screenshot shows a web panel for 'Xtreme Cinema'. On the left is a dark sidebar with a 'UNANIMO' header and a list of menu items: Actors, Billboards, Categories, Countries, Customers, Film Directors, Candy Stores, Cinema Products, Employees, Movies, and Prizes. The main content area has a header with the 'CINEMA' logo and the title 'Xtreme Cinema'. Below the header are four filter fields: 'Room Movie Date' (with a date picker set to 12/08/22), 'Room Movie Schedule' (a dropdown menu), 'Room Type' (a dropdown menu), and 'Movie Name' (a text input field). Below these filters is a table with the following data:

Room Type	Room Movie Date	Room Movie Schedule	Movie Name	Movie Poster	Goto
Normal	12/15/22	11 AM	Avengers: Endgame		
VIP	12/15/22	5 PM	Avatar: Way to water		
3D	12/15/22	8 PM	Unofficial Mission		
VIP	12/19/22	5 PM	Avatar: Way to water		
Normal	12/19/22	11 AM	Forest Gump		

Dentro del nuevo web panel, selecciona el Grid y en su propiedad class cambia a WorkWith.

También oculta el Id de la Película, elimina la variable que nos permite ver el detalle de la película y su evento asociado.

Selecciona los atributos RoomId, ScheduleMovieDate, ScheduleMovieTime y MovieName; en la propiedad class coloca: *WWColumn*. Y para cada columna en la propiedad Title cambia sus nombres a unos más cortos.

Selecciona los atributos RoomType, MovieId, y MoviePoster; en la propiedad class coloque: *WWColumn WWOOptionalColumn*. Igualmente coloque nombres más cortos para cada columna.

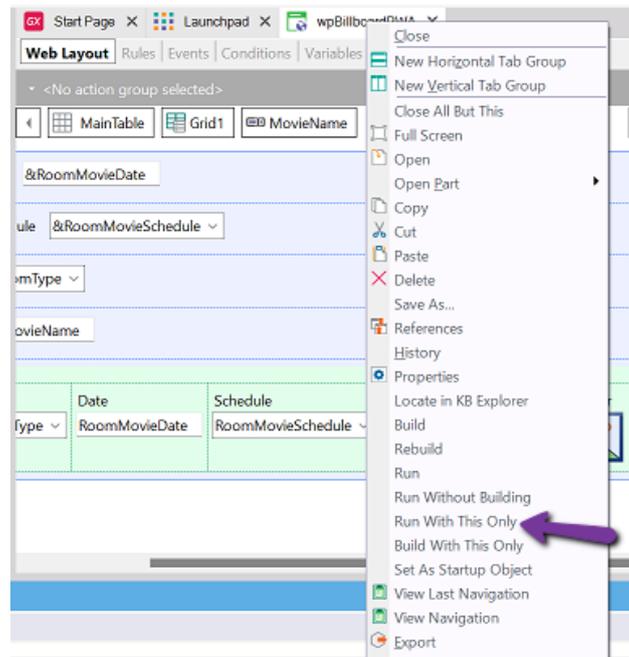
Dentro de las propiedades del web panel donde se tiene programado el Grid anidado, cambia *Main Program* por *True*. Observa que se han activado un nuevo grupo de propiedades llamado *Main object properties*. Dentro de ese bloque cambia la propiedad *Web Application* por *Progressive*.

Filter	
Type	Web Page
Master Page	MasterUnanimoSidebar
Show Master Page when Pop-u	False
Main program	True
On session timeout	Ignore
Focus control	Use Environment property value
Cache expiration lapse	
Automatic refresh	Yes
Auto compress http traffic	Use Environment property value
Qualified Name	wpBillboardPWA
Object Visibility	Public
Main object properties	
Web Application	Progressive
Location	
Generator	Default (.NET)

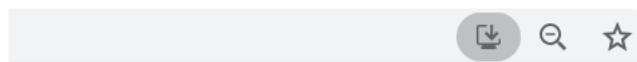
Al cambiar la propiedad default a Progressive, se han activado un nuevo grupo de propiedades llamado Web Application que configuraremos de esta manera:

Filter	
Web Application	
Web Application Name	Xtreme Cinema
Web Application Short Name	XC
Web Application Description	Cinema Progressive Web App description
Primary Text Direction	Auto
Display	Fullscreen
Background Color	 #D4D8D0
Theme Color	 #0000CD
Prefer Related Applications	False
Web Application Icon	movies 
Android Alternative App	
Android Alternative App Identifier	
iOS Alternative App	
Offline Object	WWBillboard
Compatibility	

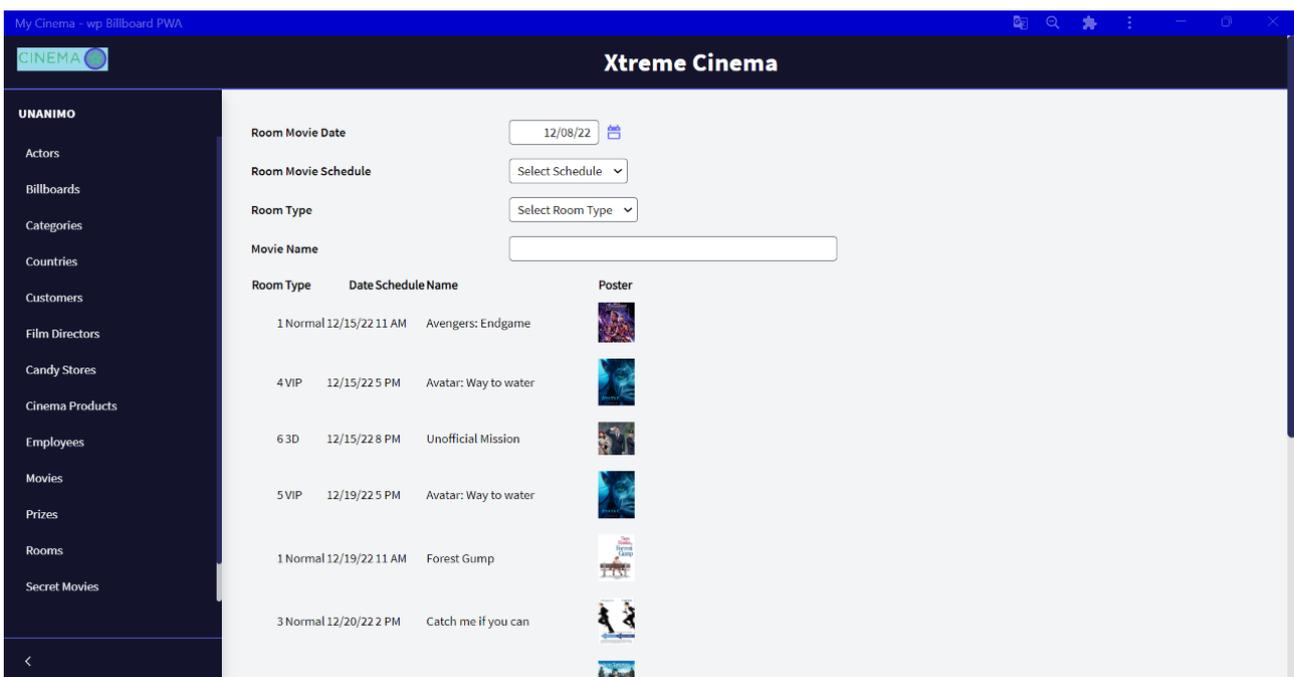
Ahora, da clic derecho en el web panel y selecciona la opción Run with this only.



En el navegador, observa la opción de instalar la aplicación, da clic sobre ella y ve al acceso directo que se ha creado en el escritorio.



Abre la aplicación dando doble clic. Observa que los colores, el icono y que al cambiar el tamaño de la pantalla la aplicación responde a la configuración que hemos realizado.



The screenshot shows a web application interface for 'Xtreme Cinema'. The browser title is 'My Cinema - wp Billboard PWA'. The application has a dark blue header with a hamburger menu icon and the text 'CINEMA' and 'Xtreme Cinema'. Below the header, there are several form fields:

- Room Movie Date:** A date input field with the value '12/08/22' and a calendar icon.
- Room Movie Schedule:** A dropdown menu with the text 'Select Schedule'.
- Room Type:** A dropdown menu with the text 'Select Room Type'.
- Movie Name:** A text input field.

Below the form fields is a table with the following data:

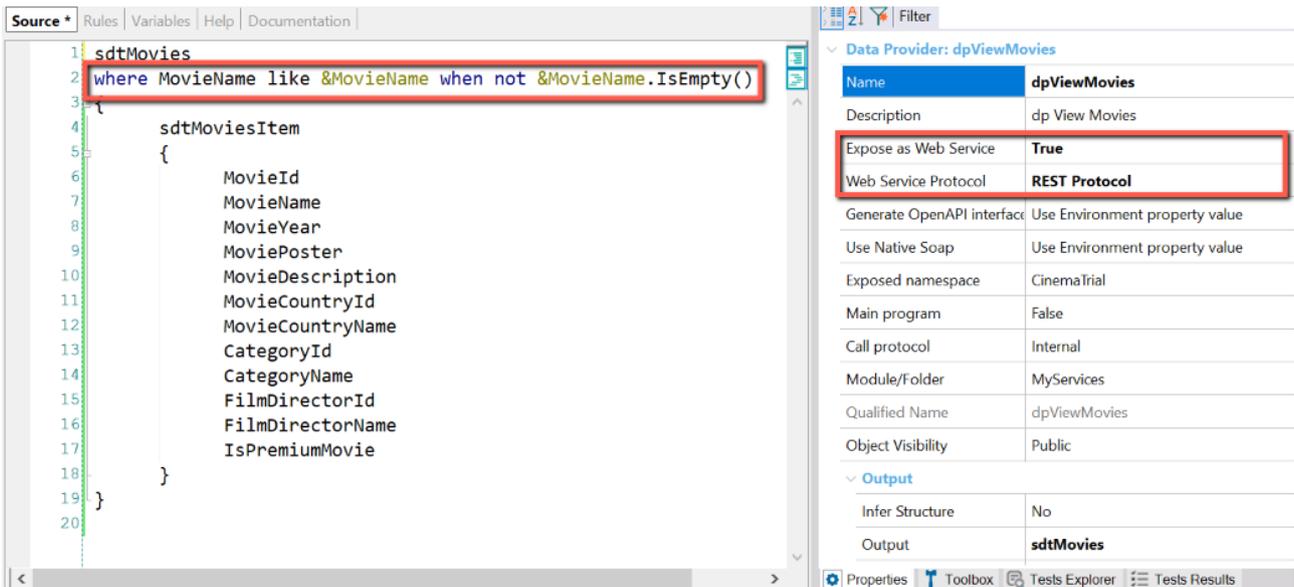
Room	Date	Schedule Name
1	12/15/22	11 AM Avengers: Endgame
4	12/15/22	5 PM Avatar: Way to water
6	12/15/22	8 PM Unofficial Mission
5	12/19/22	5 PM Avatar: Way to water
1	12/19/22	11 AM Forest Gump
3	12/20/22	2 PM Catch me if you can
2	12/21/22	5 PM Hotel Transylvania
3	12/22/22	2 PM Star Wars: The Rise of Skywalker
6	12/22/22	8 PM Avatar: Way to water

WEB SERVICES [SUGERIDO]

Se nos solicita poder publicar un servicio web que permita consultar la información de cierta película consultando por su nombre.

Para implementar lo anterior vamos a crear una colección SDT basada en la transacción Movie. Puedes arrastrar la transacción hacia la estructura o crear el SDT manualmente.

Posteriormente arrastra la estructura a un data provider y configura lo siguiente:



La condición previamente programada nos permitirá recuperar la lista de todas las películas aunque el parámetro esté vacío. Expondremos el servicio como Rest.

Coloca una regla que reciba el atributo por el que filtraremos: `parm(in:&MovieName);`

Ahora da clic derecho en el nombre del dp y selecciona en la opción “Build with this only” para exponer el servicio. Para corroborar que todo esté correcto coloca la siguiente dirección en el navegador:

<http://server/baseURL/rest/nombreDelDataProvider>

Deberá aparecer lo siguiente:

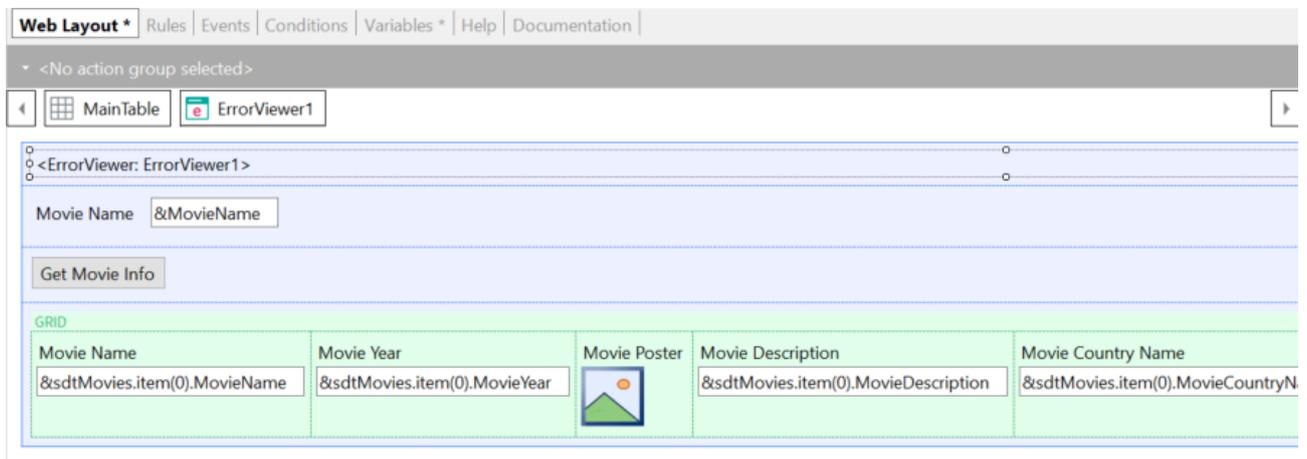
```
[{"MovieId":1,"MovieName":"Avengers End Game","MovieYear":2019,"MoviePoster":"PublicTempStorage\\multimedia\\AvengersEnd_4f30d21695f74e7fb87112f132ab9770.jpg","MovieDescription":"Avengers: Endgame is an American superhero film based on the Marvel Comics superhero team the Avengers.", "MovieCountryId":2,"MovieCountryName":"United States","CategoryId":5,"CategoryName":"Action","FilmDirectorId":1,"FilmDirectorName":"Joe Russo","IsPremiumMovie":true}, {"MovieId":2,"MovieName":"Catch me if you can","MovieYear":2002,"MoviePoster":"PublicTempStorage\\multimedia\\CatchMe_c115ac88cf1742529610664b27a9f24.jpg","MovieDescription":"The film is based on the life of Frank Abagnale, who, before his 19th birthday, successfully performed cons worth millions of dollars by posing as a Pan American World Airways pilot, a Georgia doctor and a Louisiana parish prosecutor.", "MovieCountryId":2,"MovieCountryName":"United States","CategoryId":5,"CategoryName":"Action","FilmDirectorId":3,"FilmDirectorName":"Steven Spielberg","IsPremiumMovie":false}, {"MovieId":3,"MovieName":"Forrest Gump","MovieYear":1994,"MoviePoster":"PublicTempStorage\\multimedia\\ForrestGump_1293eaa799bd4485abd10a7ff00d11c.jpg","MovieDescription":"The story depicts several decades in the life of Forrest Gump (Hanks), a slow-witted but kind-hearted man from Alabama who witnesses and unwittingly influences several defining historical events in the 20th century United States.", "MovieCountryId":2,"MovieCountryName":"United States","CategoryId":1,"CategoryName":"Comedy","FilmDirectorId":7,"FilmDirectorName":"Robert Zemeckis","IsPremiumMovie":false}, {"MovieId":4,"MovieName":"Hotel Transylvania","MovieYear":2012,"MoviePoster":"PublicTempStorage\\multimedia\\HotelTransylvania_36a5b8a603c549b0b209c0db3a7d925a5.jpg","MovieDescription":"The film tells the story of Count Dracula, the owner of a hotel called Hotel Transylvania where the world's monsters can take a rest from human civilization.", "MovieCountryId":2,"MovieCountryName":"United States","CategoryId":4,"CategoryName":"Cartoon","FilmDirectorId":8,"FilmDirectorName":"Genndy Tartakovsky","IsPremiumMovie":false}, {"MovieId":5,"MovieName":"The family Man","MovieYear":2000,"MoviePoster":"PublicTempStorage\\multimedia\\FamilyMan_ce19e3c9ddf64a45b7bb9758fb6ce33b.jpg","MovieDescription":"The film centers on a man who experiences what his life might have been if he had made a different decision earlier in his life.", "MovieCountryId":2,"MovieCountryName":"United States","CategoryId":3,"CategoryName":"Drama","FilmDirectorId":4,"FilmDirectorName":"Brett Ratner","IsPremiumMovie":false}, {"MovieId":6,"MovieName":"Harry Potter and the Prisoner of Azkaban","MovieYear":2004,"MoviePoster":"PublicTempStorage\\multimedia\\HPPrisoner_7c5f2c408a194c64b08984814713a385.jpg","MovieDescription":"The film stars Daniel Radcliffe as Harry Potter, with Rupert Grint as Ron Weasley, and Emma Watson as Hermione Granger. Its story follows Harry Potter's third year at Hogwarts as he is informed that a prisoner named Sirius Black has escaped from Azkaban and intends to kill him.", "MovieCountryId":4,"MovieCountryName":"United Kingdom","CategoryId":6,"CategoryName":"Fantasy","FilmDirectorId":2,"FilmDirectorName":"Alfonso Cuaron","IsPremiumMovie":true}, {"MovieId":7,"MovieName":"Unofficial Mission","MovieYear":2017,"MoviePoster":"PublicTempStorage\\multimedia\\misionNoOficial_372f49892b604d39b1252d62e431613f.jpg","MovieDescription":"\u000aUnofficial Mission is a documentary comedy, directed by Alfonso Guerrero, Denny Brechner and Marcos Hecht.", "MovieCountryId":1,"MovieCountryName":"Uruguay","CategoryId":1,"CategoryName":"Comedy","FilmDirectorId":5,"FilmDirectorName":"Denny Brechner","IsPremiumMovie":false}, {"MovieId":8,"MovieName":"Inception","MovieYear":2010,"MoviePoster":"PublicTempStorage\\multimedia\\Inception_4199f1d287e3419f80146fdb645c3e02.jpg","MovieDescription":"The film stars Leonardo DiCaprio as a professional thief who steals information by infiltrating the subconscious of his targets.", "MovieCountryId":2,"MovieCountryName":"United States","CategoryId":5,"CategoryName":"Action","FilmDirectorId":6,"FilmDirectorName":"Christopher Nolan","IsPremiumMovie":false}]
```

Ahora construiremos un web panel que nos permitirá consumir el servicio Rest con el tipo de dato HttpClient. Puede ser dentro de la misma KB o en una diferente. El web panel deberá tener la siguiente estructura:

Variables:

Variable	Tipo de dato
HttpClient	HttpClient
MovieName	Character(50)
SdtMovies	Basado en el SDT previamente creado
StringParameter	Character(200)

Web Layout



Eventos

```

1 Event 'Get Movie Info'
2     &httpClient.Host= "localhost"
3     &httpClient.Port = 8082
4     &httpClient.BaseUrl = "/MyCinemaNETSQLServer/rest/"
5     &httpClient.secure = 0
6     &StringParameter = '<DP_NAME>?' + 'MovieName=' + &MovieName
7     &httpClient.Execute('GET', &StringParameter)
8
9     if &httpClient.StatusCode = 200
10        msg("Everything Ok!")
11        &sdtMovies.FromJson(&httpClient.ToString())
12
13    else
14        msg("Error: " + &httpClient.StatusCode.ToString())
15    endif
16 Endevent

```

Ejecuta la aplicación y verifica que se pueda consumir el servicio. Escribe el nombre de una película o deja vacío el campo, en ambos casos se debería poder consumir el servicio.

NOTA: Habilita la propiedad *read only* en true para cada campo del SDT que se está imprimiendo en pantalla.

Xtreme Cinema

Everything Ok!

Movie Name:

[GET MOVIE INFO](#)

Movie Name	Movie Year	Movie Poster	Movie Description	Movie Country Name	Category Name	Film Director Name
Avatar: Way to water	2022		Jake Sully and Ney'tiri have formed a family and are doing everything to stay together. However, they must leave their home and explore the regions of Pandora. When an ancient threat resurfaces, Jake must fight a difficult war against the humans.	United States	Fantasy	James Cameron

Xtreme Cinema

Everything Ok!

Movie Name:

[GET MOVIE INFO](#)

Movie Name	Movie Year	Movie Poster	Movie Description	Movie Country Name	Category Name	Film Director Name
Avengers: Endgame	2019		After Thanos, an intergalactic warlord, disintegrates half of the universe, the Avengers must reunite and assemble again to reinvigorate their trounced allies and restore balance.	United States	Action	Joe Russo
Catch me if you can	2002		Notorious con artist Frank Abagnale has duped people worth millions of dollars with his masterful art of deception. With his scams getting bolder, he is soon pursued by FBI agent Carl Hanratty.	United States	Comedy	Steven Spielberg
Avatar: Way to water	2022		Jake Sully and Ney'tiri have formed a family and are doing everything to stay together. However, they must leave their home and explore the regions of Pandora. When an ancient threat resurfaces, Jake must fight a difficult war against the humans.	United States	Fantasy	James Cameron
Forest Gump	1994		Forrest, a man with low IQ, recounts the early years of his life when he found himself in the middle of key historical events. All he wants now is to be reunited with his childhood sweetheart, Jenny.	United States	Drama	Robert Zemeckis