Curso práctico de actualización a

PARTE 1

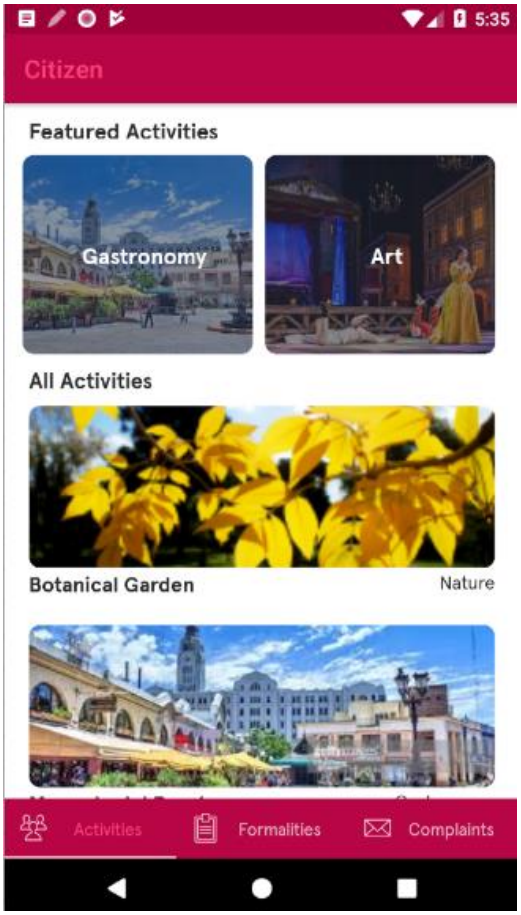
Abril 2020

CONTENIDO

OBJETIVO

Para cumplir el objetivo de desarrollar aplicaciones multi-experiencia y omnichannel, mejorando la integración con el equipo de diseño, en este práctico trataremos los temas Design Systems y experiencia de usuario relacionada con las maneras de lograr que la información repetitiva fluya de manera más continua y adaptable en la UI –todo lo que hace a los nuevos tipos de grid–.

La aplicación sobre la que trabajaremos es una simplificación de una app para la municipalidad de una ciudad, que ofrece un frontend Web y uno SD para que los ciudadanos mediante su identificador de usuario puedan realizar reclamos (por ejemplo por árboles caídos, semáforos que no funcionan, coches mal estacionados, etc.), puedan realizar trámites (por ejemplo para obtener licencia de conducir, refinanciar una deuda con el municipio, instalar elementos de publicidad, etc), reservando un turno para ser atendidos por el personal del municipio. En el frontend se les muestran también las diversas actividades culturales que ofrece la ciudad:



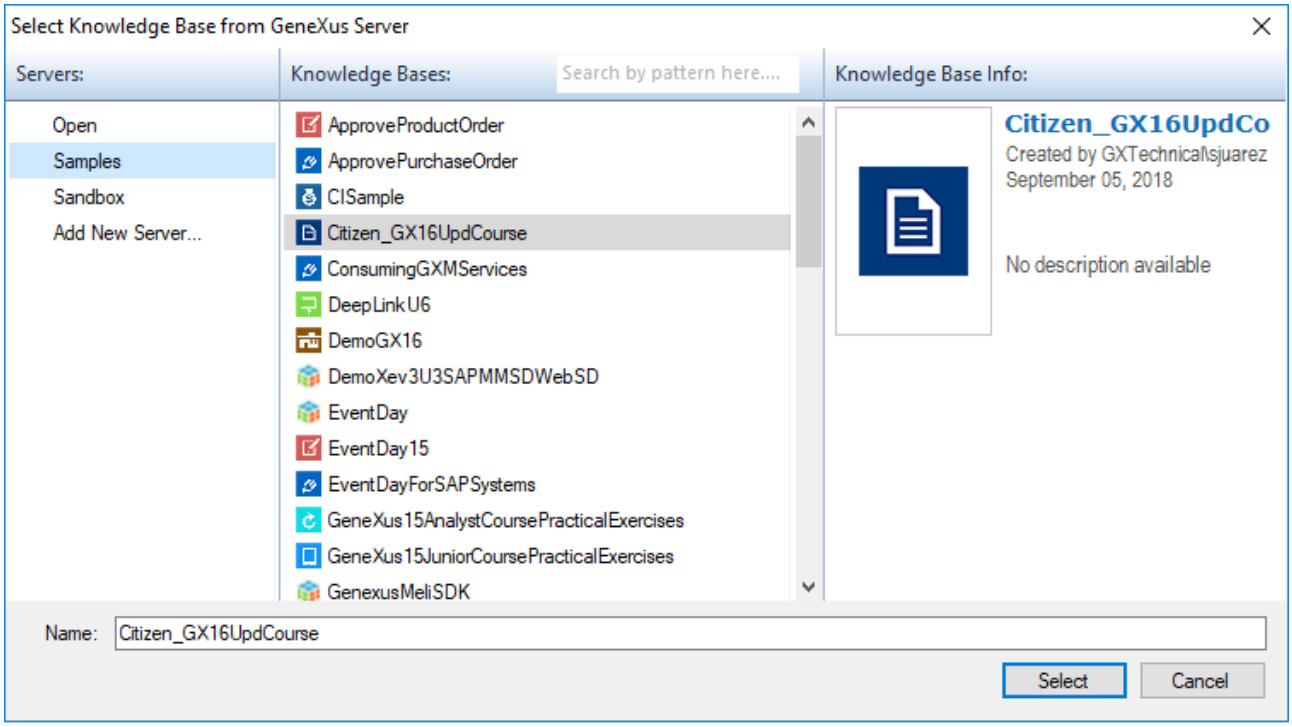
Y también se cuenta con un backoffice web para que ciertos funcionarios de la municipalidad manejen los datos y vean estadísticas.

COMENCEMOS

Utilizaremos la versión GeneXus 16 y el Emulador de Android SDK. La letra de este práctico se ha actualizado de acuerdo al upgrade 9 de GeneXus 16. Si lo realiza con una versión posterior, considere que puede haber cambios entre lo que aquí se le muestra y lo que usted ve.

A menos que los instructores le digan que la KB ya está creada en su máquina, haga en GeneXus: File/New Knwoledge Base from GeneXus Server, eligiendo como Server KB URL: http://samples.genexusserver.com/v16/.

En Select Server KB elija la de nombre “Citizen_GX16UpdCourse”:

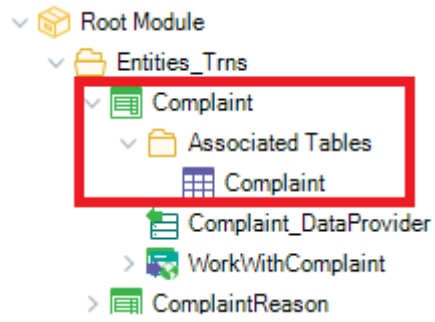


Y seleccione la versión trunk.

Realice un Rebuild All y Create. Prototiparemos localmente y no en la nube. Asegúrese de que ya estén configurados nombre de server, y de base de datos, o configúrelos con la instancia del SQLServer de la máquina y nombre de DB la que usted desee. Consulte al instructor.

FAMILIARIZACIÓN CON LA KB

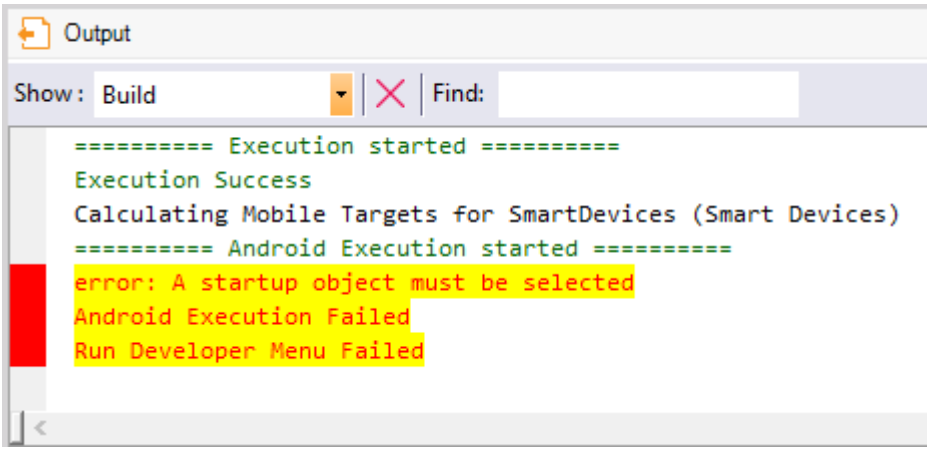
Bajo el folder Entities_Trns se encuentran las entidades que modelan la realidad de la app. Observe que ahora las tablas se muestran bajo cada transacción en el KB Explorer.



Asimismo observe que se han definido los Data Providers para poblar las tablas asociadas a las transacciones (prendiendo la propiedad Data Provider de la transacción). Serán invocados automáticamente en la primera ejecución.

Luego, se cuenta con dos folders FrontendWeb y FrontendSD que implementan ambos frontends, con los mains de la app que usarán los ciudadanos.

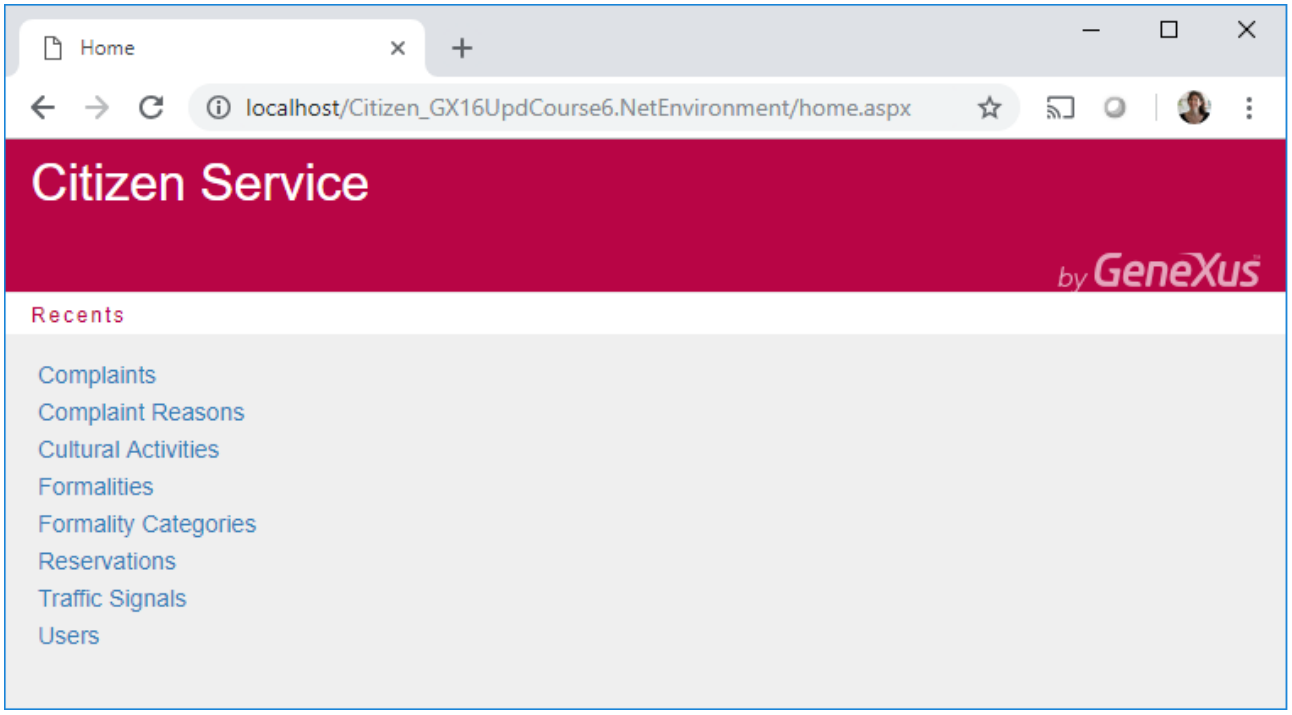
El Web panel Home dentro de GeneXus/Web contiene el main del backoffice (es el default), que será utilizado por los funcionarios del municipio para la administración de la info del sitio. Será el que se le desplegará como resultado del F5, que a su vez desplegará el siguiente error:



Esto es porque no configuramos Startup object, debido a que vamos a estar alternando entre backoffcie y frontend Web y SD. Puede especificar al Home del backoffice como Startup Object o no. Como desee, ya que de todas maneras deberá hacer Run sobre el main que desee ejecutar en cada oportunidad (CitizenMenu para frontend SD, CitizenWeb para frontend Web).

El folder Queries cuenta con unas consultas que se utilizarán en otro práctico.

Si ejecuta el web panel Home:



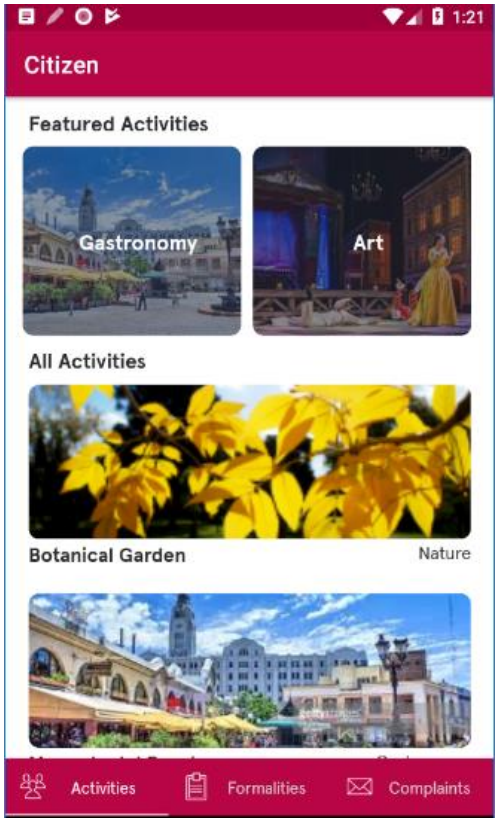
Podrá ver los work withs de las entidades, con los datos cargados. Déjelo abierto.

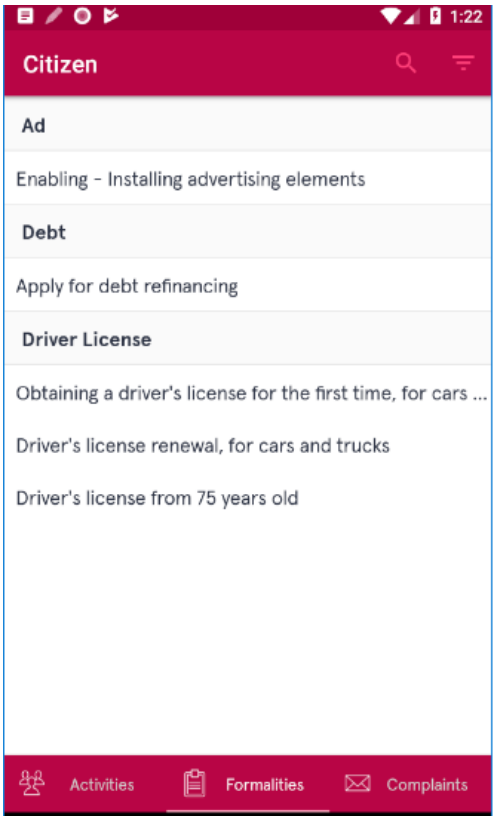
STENCILS

Ejecute el main de la app SD (objeto CitizenMenu dentro del folder FrontendSD).

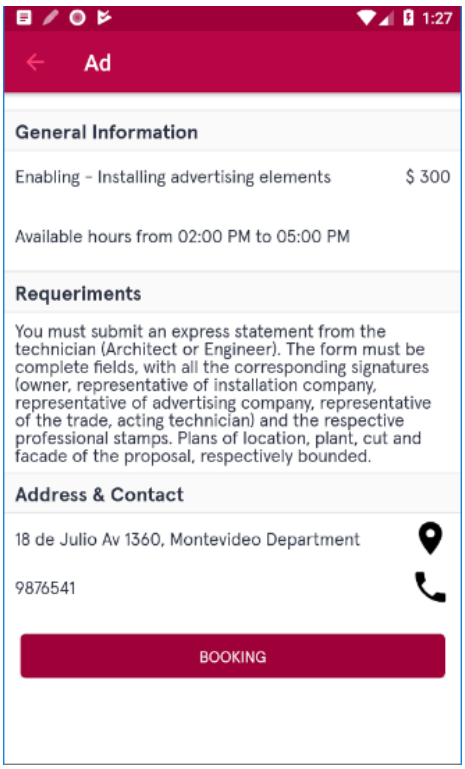
JUSTIFICACIÓN DEL USO DE STENCILS

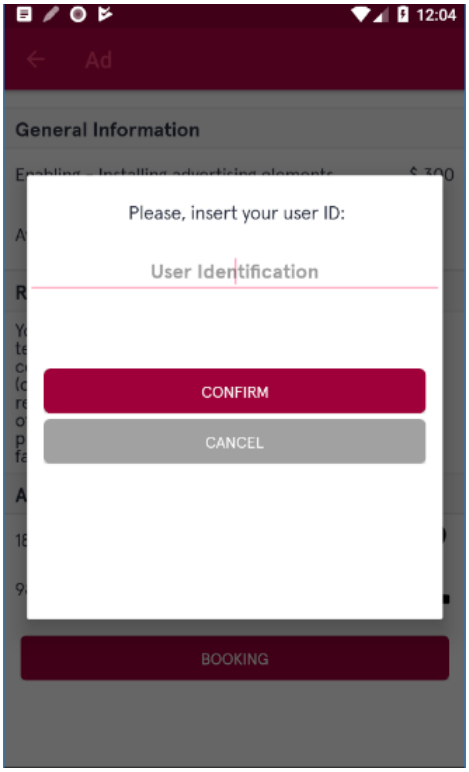
Verá una primera pantalla que muestra las actividades culturales (Activities). Y en el segundo tab del menú que se encuentra debajo podrá ver los trámites que el usuario puede realizar (instalar publicidad, refinanciar una deuda, obtener licencia de conducir):



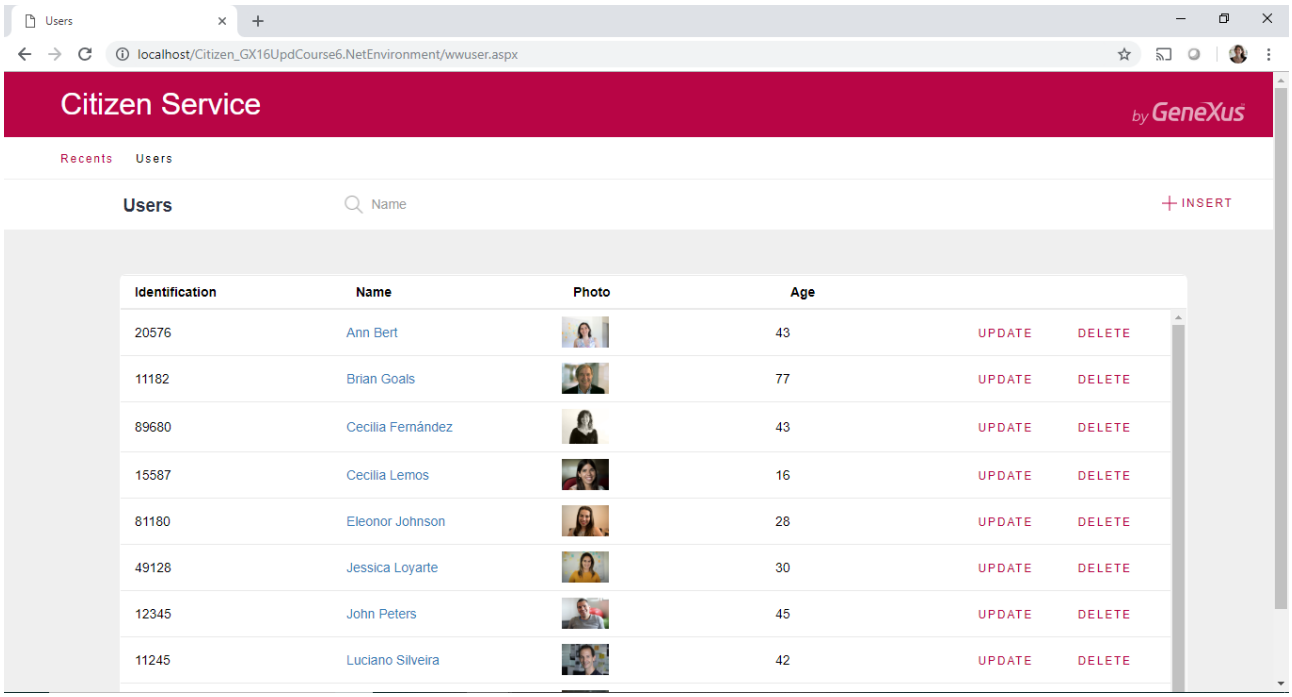


Si el usuario elige un trámite, por ejemplo, habilitar publicidad, se le muestra la información general del trámite, y se le permite al usuario reservar un turno para ser atendido para realizar ese trámite. Y al presionar el “BOOKING” se le pide el ID de usuario:

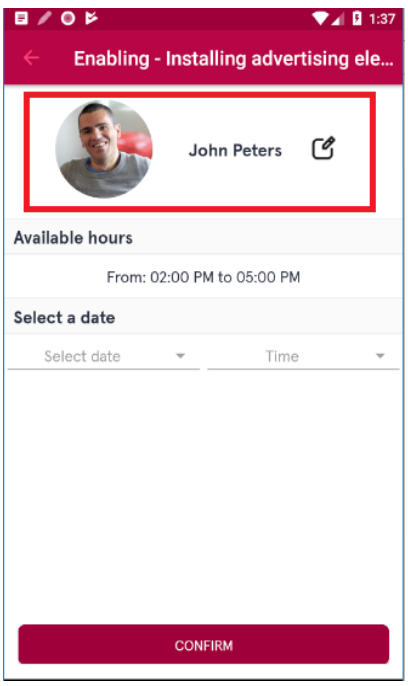


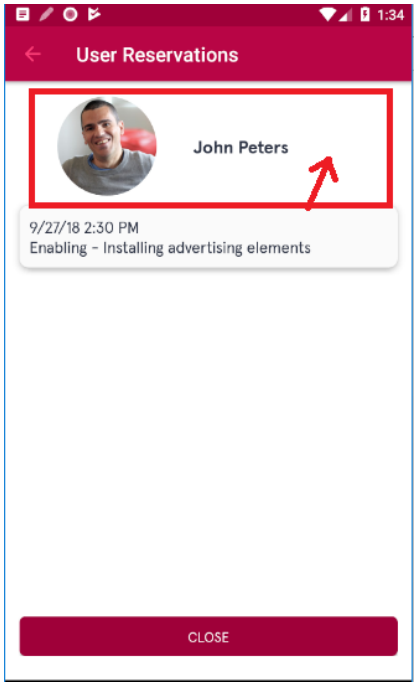


Le recomendamos abrir el Backoffice para ver los usuarios registrados y sus IDs:



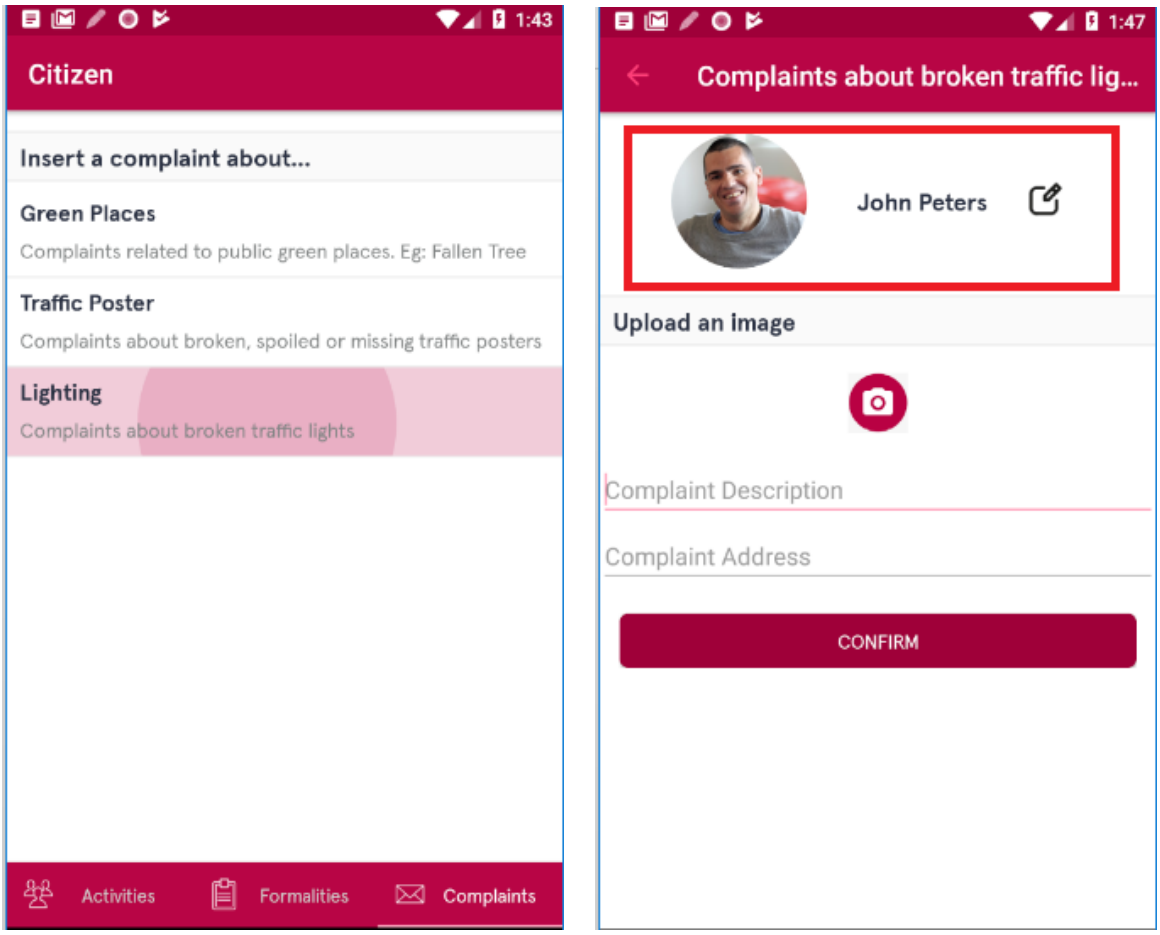
Al ingresar uno y confirmar (por ejemplo el de ID 12345), se le presenta una pantalla donde se muestra foto y nombre del usuario ingresado, y se le permite seleccionar día y hora, dentro de los horarios habilitados del trámite, para reservarle un turno para ser atendido. Al hacerlo, si todo fue bien, se le presentará otra pantalla que le muestra toda la información del trámite, y se le ofrece la opción de ver todas las reservas que tiene hasta el momento:





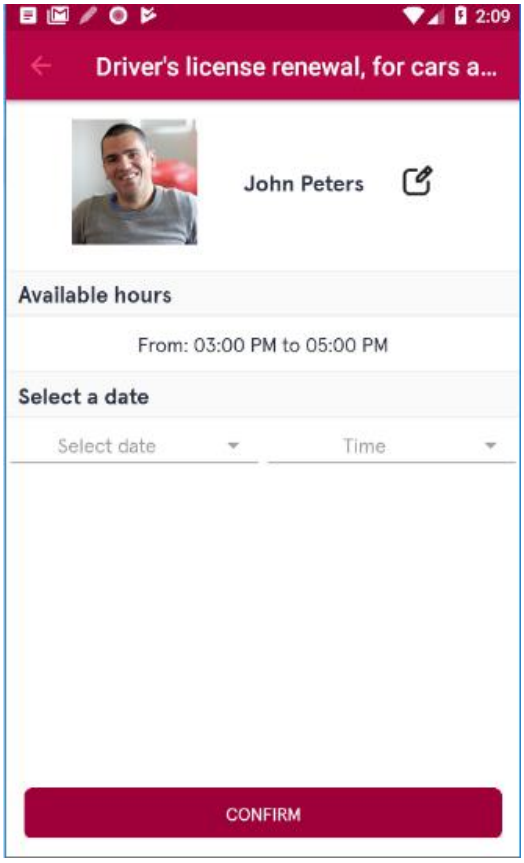
En esta última pantalla también se visualiza foto y nombre del usuario, con exactamente el mismo diseño que antes, aunque donde se ha ocultado la posibilidad de cambiar de usuario.

Por otro lado, si vamos a la opción Complaints del menú, seleccionando el tipo de reclamo, el usuario podrá ingresar uno de ese tipo al sistema:



Observe que en esta pantalla, UserComplaint, también se muestra la información del usuario de la misma exacta manera que en la del trámite.

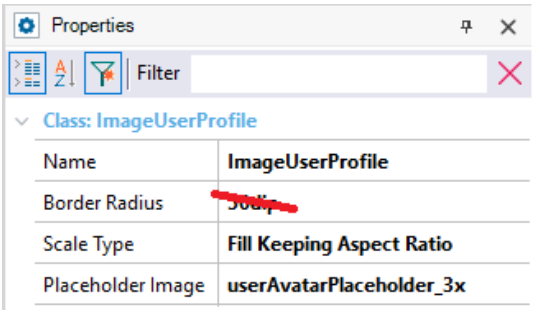
Si quiere modificar la manera en la que se muestra la foto (en las tres pantallas en las que aparece) para que no salga más redondeada, tal como se ve aquí:



¿Cómo lo implementa?

SOLUCIÓN:

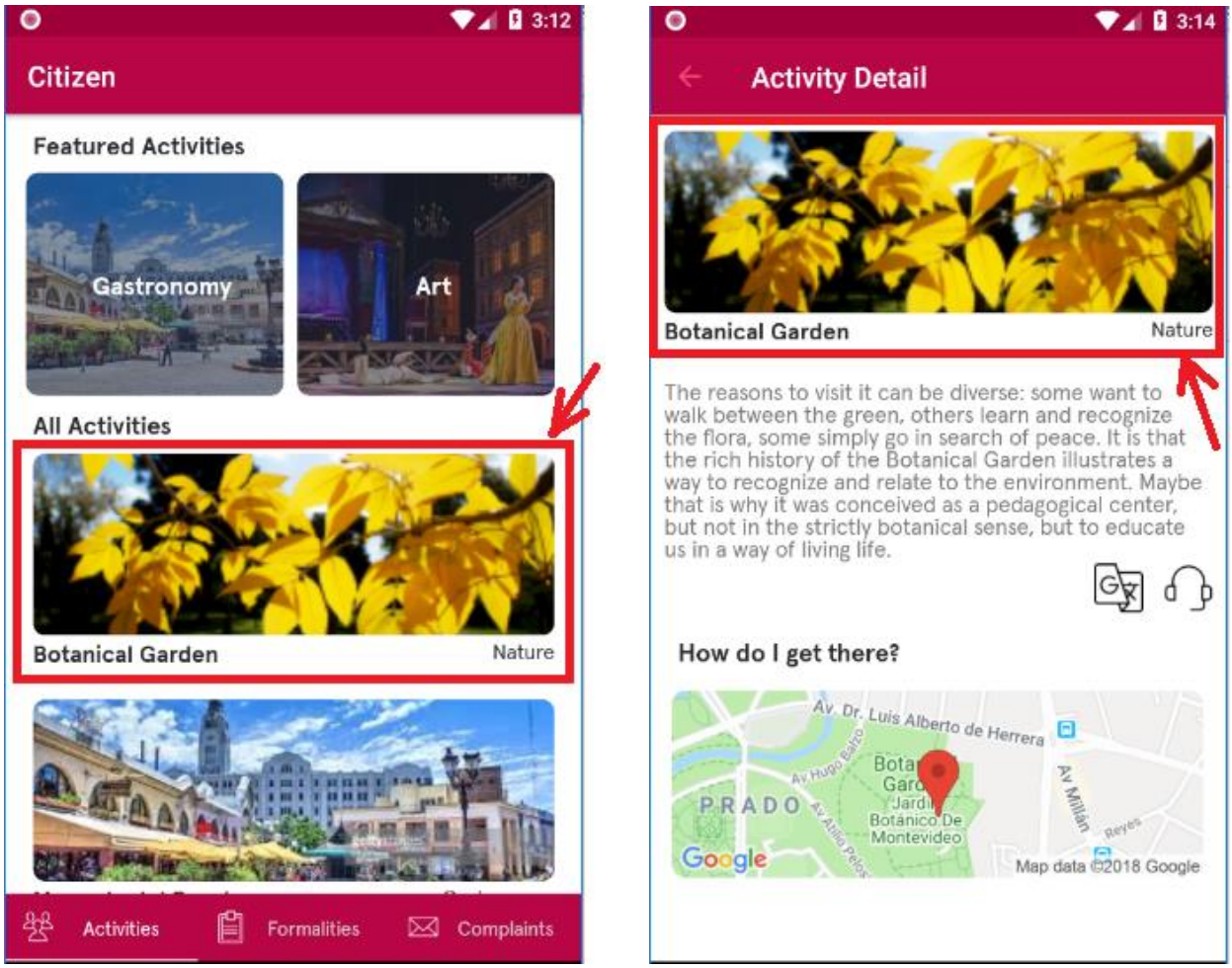
Abra el stencil UserProfile y vea que la foto tiene asociada la clase ImageUserProfile del theme CarmineSDCourse. Cámbiele la clase Border Radius, dejándola vacía.



Ejecute CitizenMenu y observe el cambio en las tres pantallas.

CREAR UN STENCIL Y USARLO

Ahora deseamos que la pantalla que muestra las actividades culturales (Activities) muestre arriba las actividades destacadas, y abajo todas las actividades. Cuando se selecciona una actividad en cualquiera de ambos grids, se visualiza una pantalla con su detalle:

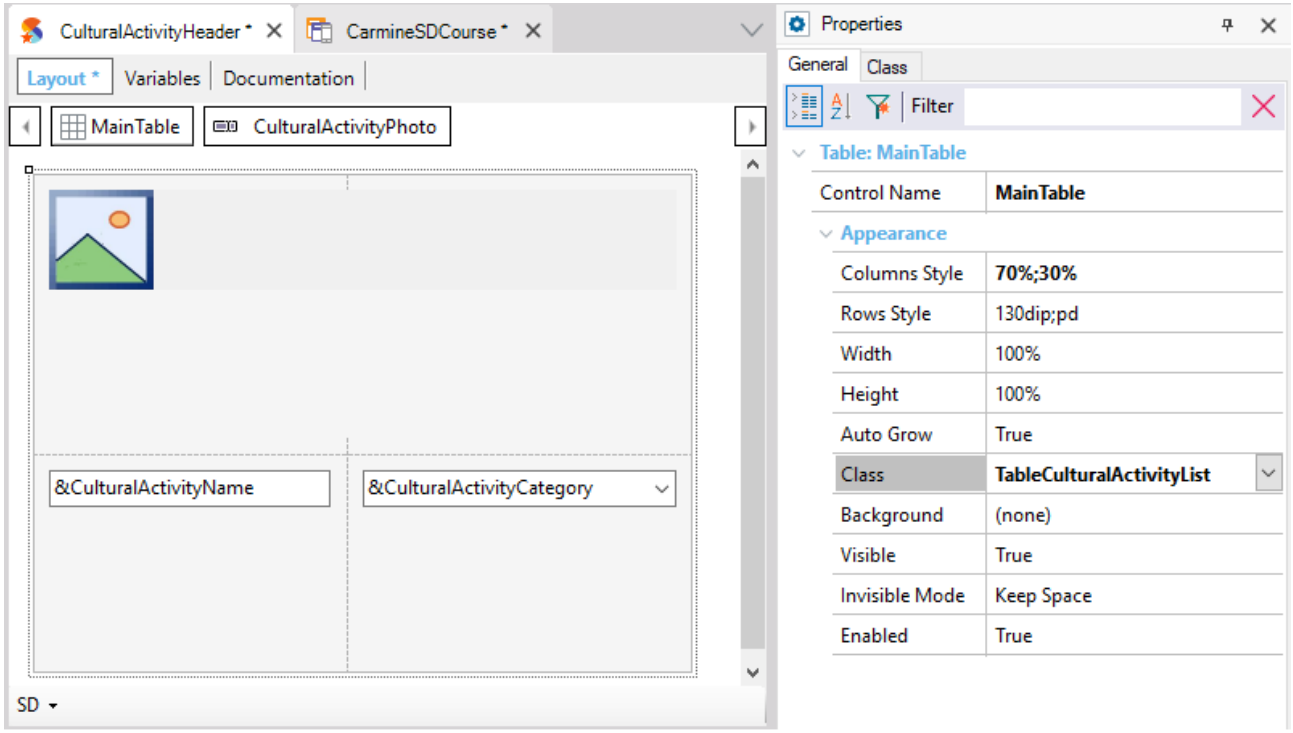


Observe que la imagen de la actividad, su nombre y categoría se muestran de manera idéntica en el grid que muestra All Activities en el primer panel y en el segundo panel. Para evitar repetir el diseño es que utilizamos stencils.

Implemente un stencil para mostrar esta información y utilícelo en ambos paneles: Activities y ActivityDetail.

POSIBLE SOLUCIÓN:

Cree el stencil de nombre CulturalActivityHeader (puede hacerlo a mano, o puede posicionarse sobre la tabla que ya contiene la foto y los dos atributos en el panel Activities, y con botón derecho elegir Wrap as new stencil):



Si lo hizo a mano: a la variable con la imagen configúrele las propiedades:

- Label Position: None
- Class: ImageCulturalActivityList
- Col Span: 2

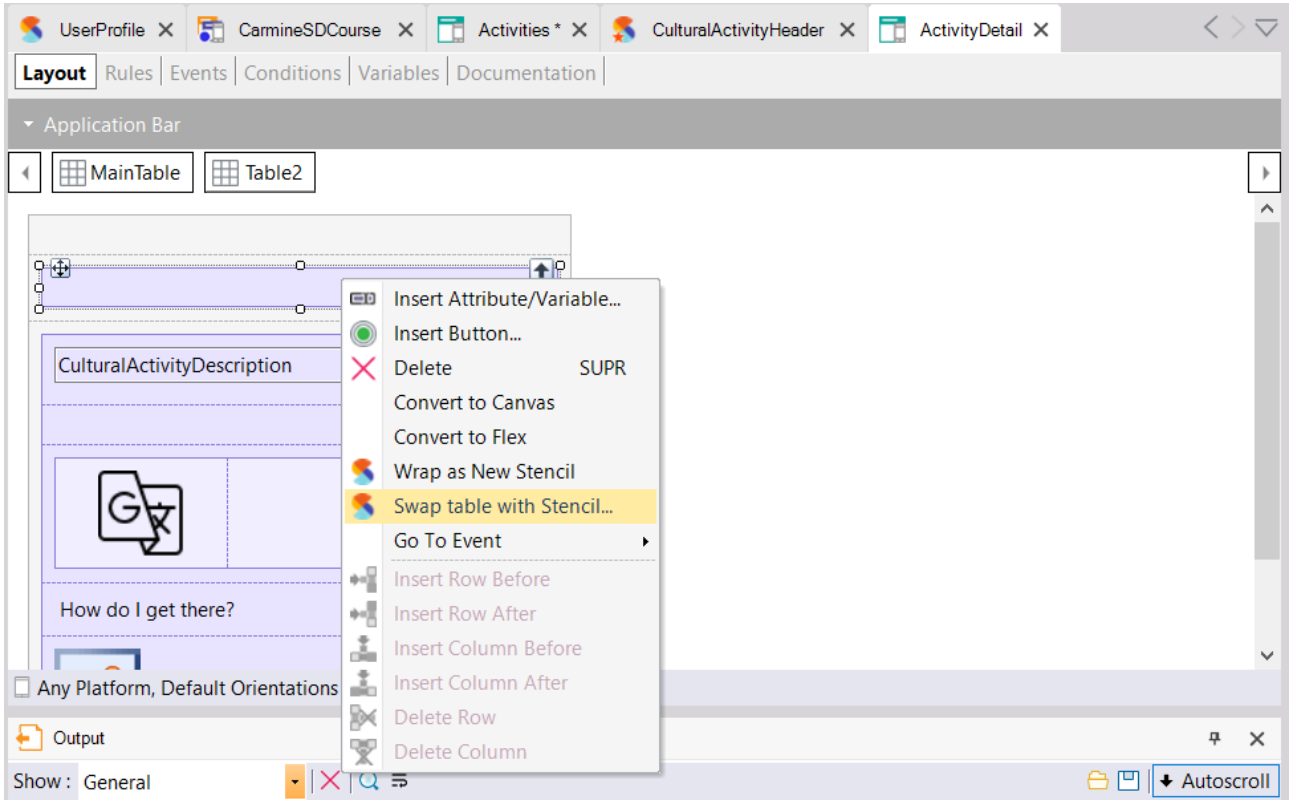
A la variable que muestra el nombre a la izquierda configúrele las propiedades:

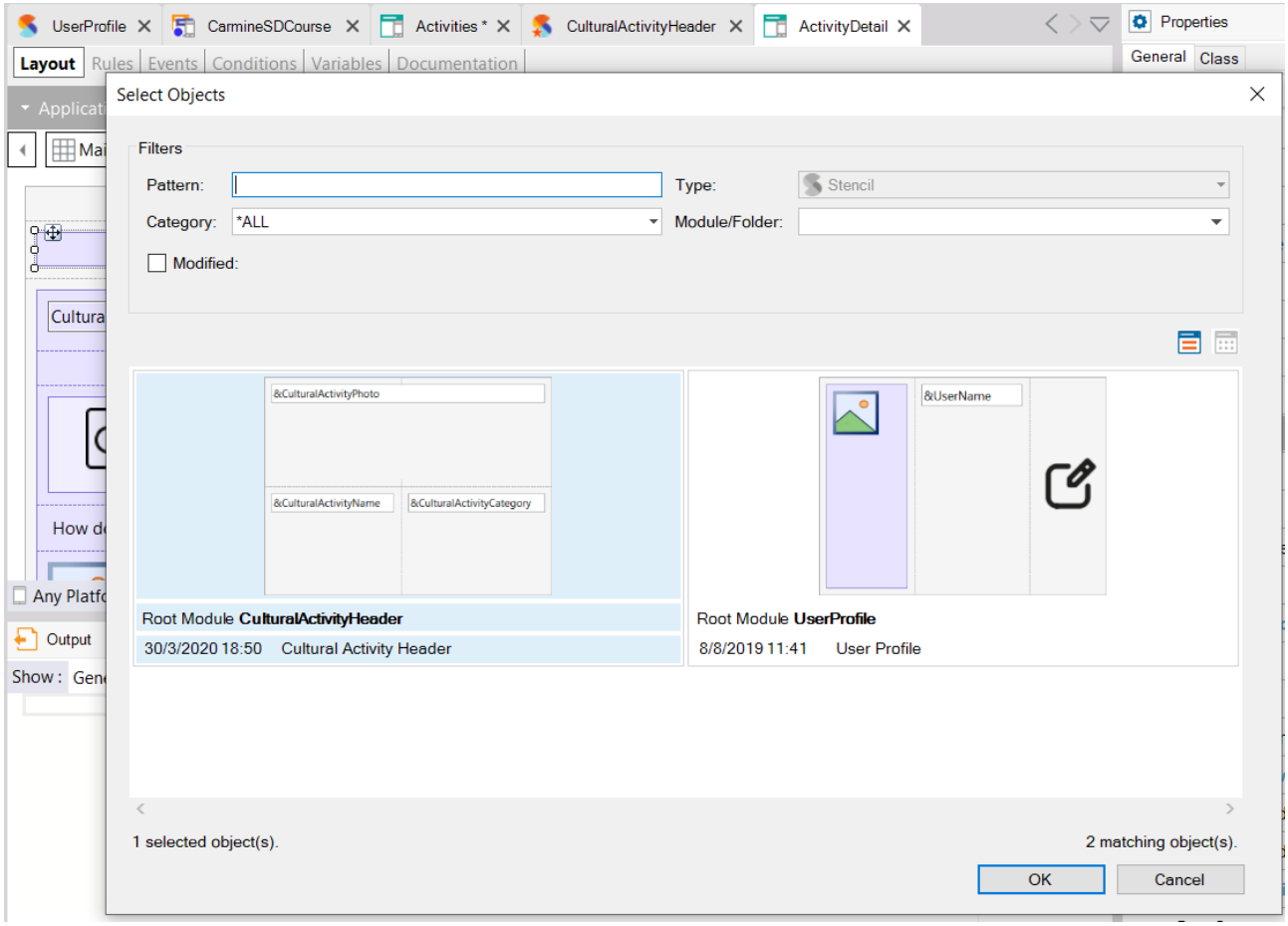
- Label Position: None
- Class: AttributeActivityNameList

A la variable que muestra la categoría a la derecha configúrele las propiedades:

- Label Position: None
- Class: AttributeActivityCategoryList
- Horizontal Aligment: Right

En el panel ActivityDetail, en la tabla donde insertaría los atributos CulturalActivityPhoto, CulturalActivityName y CulturalActivityCategory, que está vacía, busque el Stencil que acaba de crear, e insértelo.

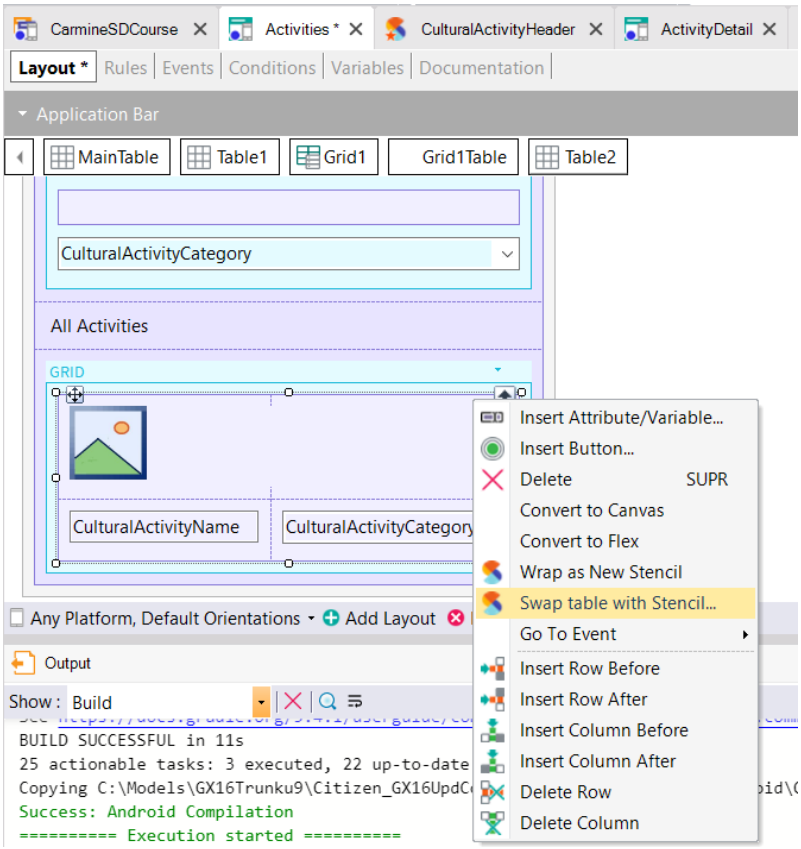

UserProfile
CarmineSDCourse
Activities *
CulturalActivityHeader
ActivityDetail
Layout
Rules
Events
Conditions
Variables
Documentation
Application Bar
MainTable
Table2
CulturalActivityDescription
How do I get there?
Insert Attribute/Variable...
Insert Button...
Delete SUPR
Convert to Canvas
Convert to Flex
Wrap as New Stencil
Swap table with Stencil...
Go To Event
Insert Row Before
Insert Row After
Insert Column Before
Insert Column After
Delete Row
Delete Column
Any Platform, Default Orientations
Output
Show : General
Autoscroll


Properties
General
Class
Select Objects
Filters
Pattern:
Type:
Stencil
Category:
*ALL
Module/Folder:
Modified:
&CulturalActivityPhoto
&CulturalActivityName
&CulturalActivityCategory
&UserName
Root Module CulturalActivityHeader
30/3/2020 18:50 Cultural Activity Header
Root Module UserProfile
8/8/2019 11:41 User Profile
1 selected object(s).
2 matching object(s).
OK
Cancel

Nota:

- Deberá reemplazar las variables por los atributos CulturalActivityPhoto, CulturalActivityName y CulturalActivityCategory.

Si creó el stencil a mano, entonces en el panel Activities, posiciónese sobre la tabla del grid que muestra todas las actividades, y presionando botón derecho, observe que al hacer “Swap table with stencil”:

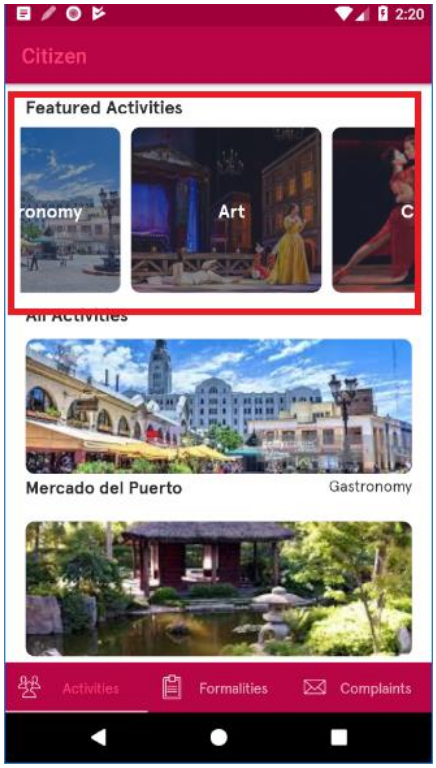


y elegirlo, ya sustituirá la tabla por el stencil, respetando los atributos (no los cambia por variables).

Pruebe en ejecución (Run sobre el objeto CitizenMenu).

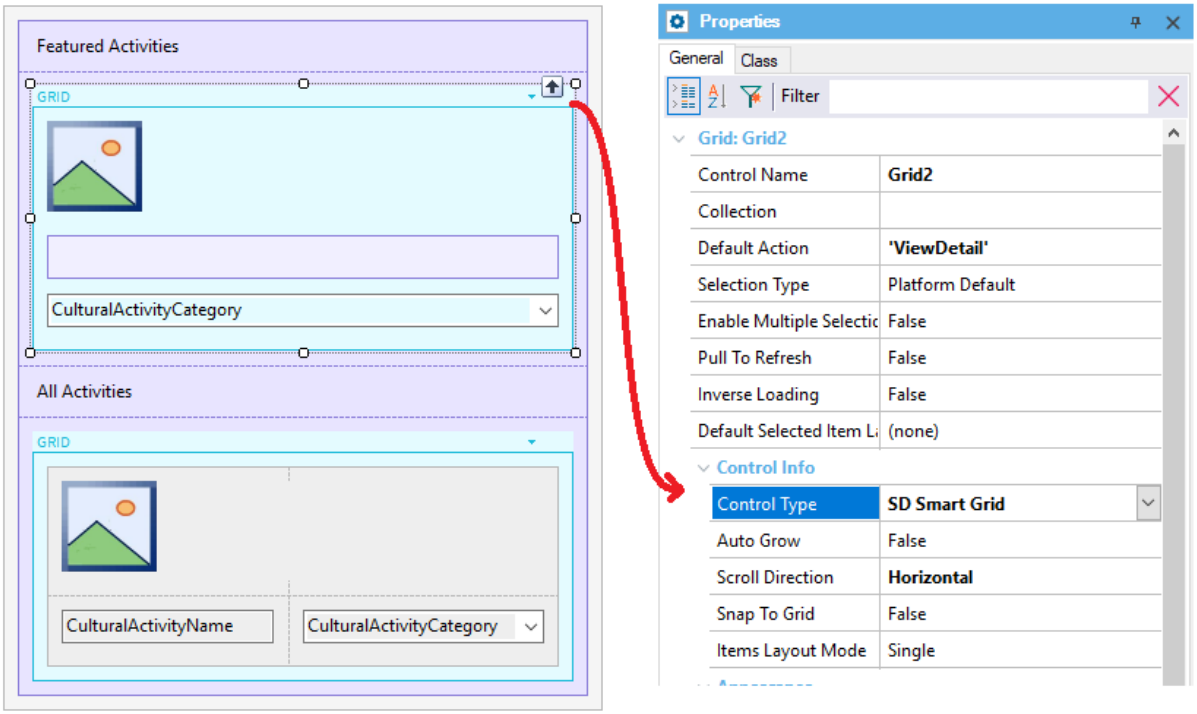
SMART GRID

El panel de actividades culturales muestra arriba un grid con las actividades marcadas como destacadas (ver atributo CulturalActivityFeatured de la transacción CulturalActivity). Se desea que esas actividades se vean en un formato carrusel, como se muestra en la imagen:

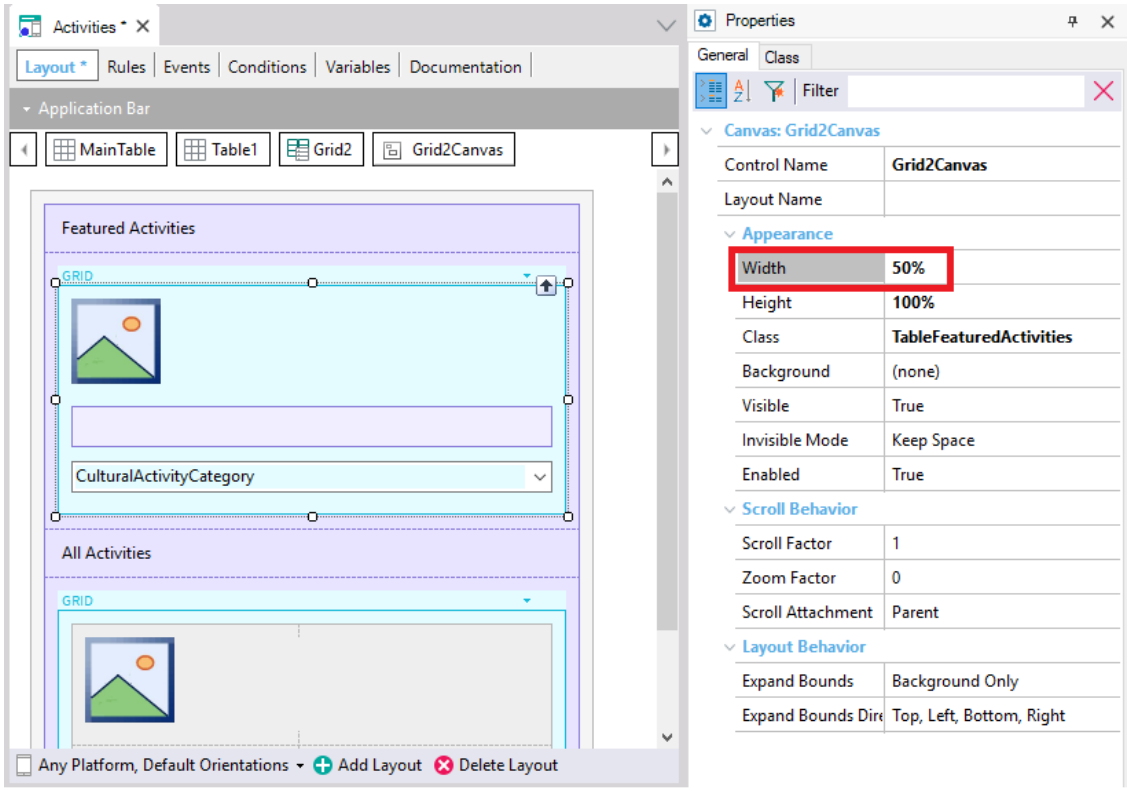


¿Cómo lo implementa?

SOLUCIÓN:



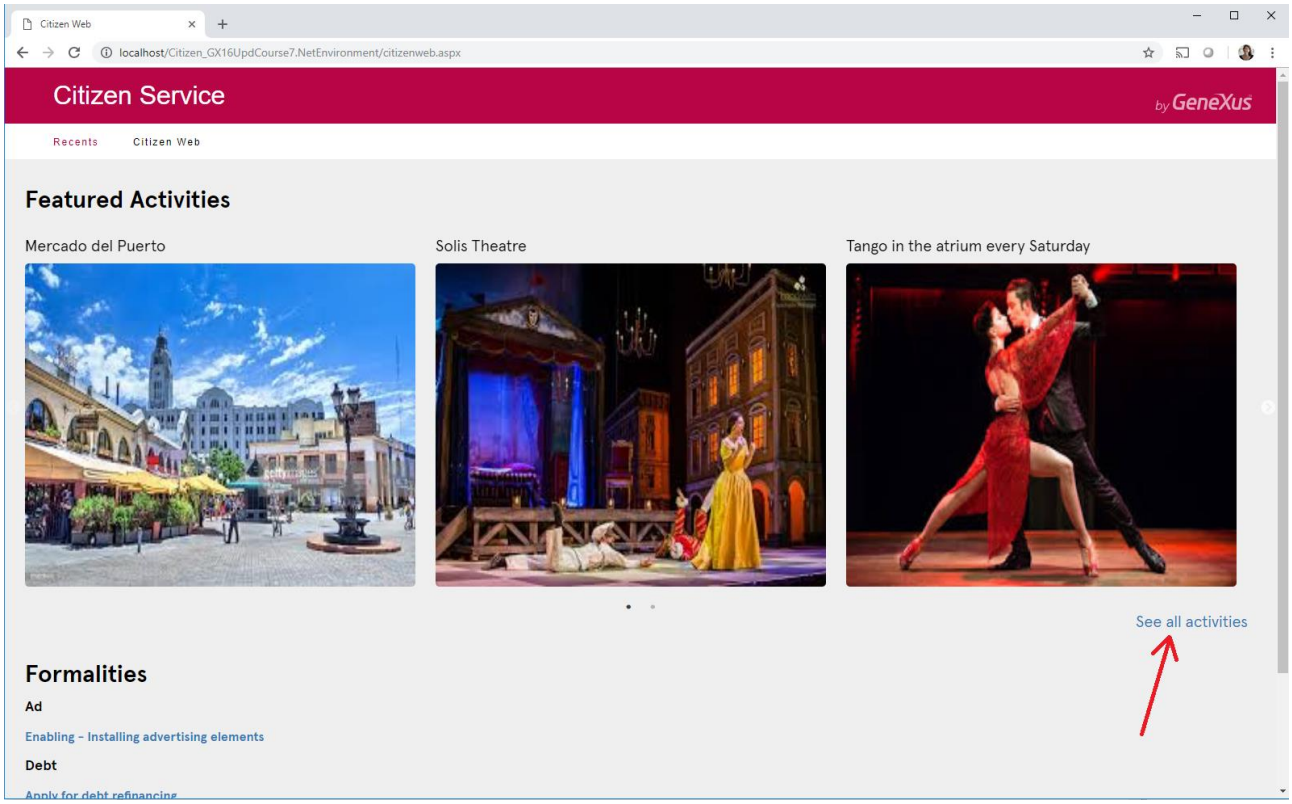
Cámbiele, también, el Width al Canvas para que ocupe el 50% y no el 100:



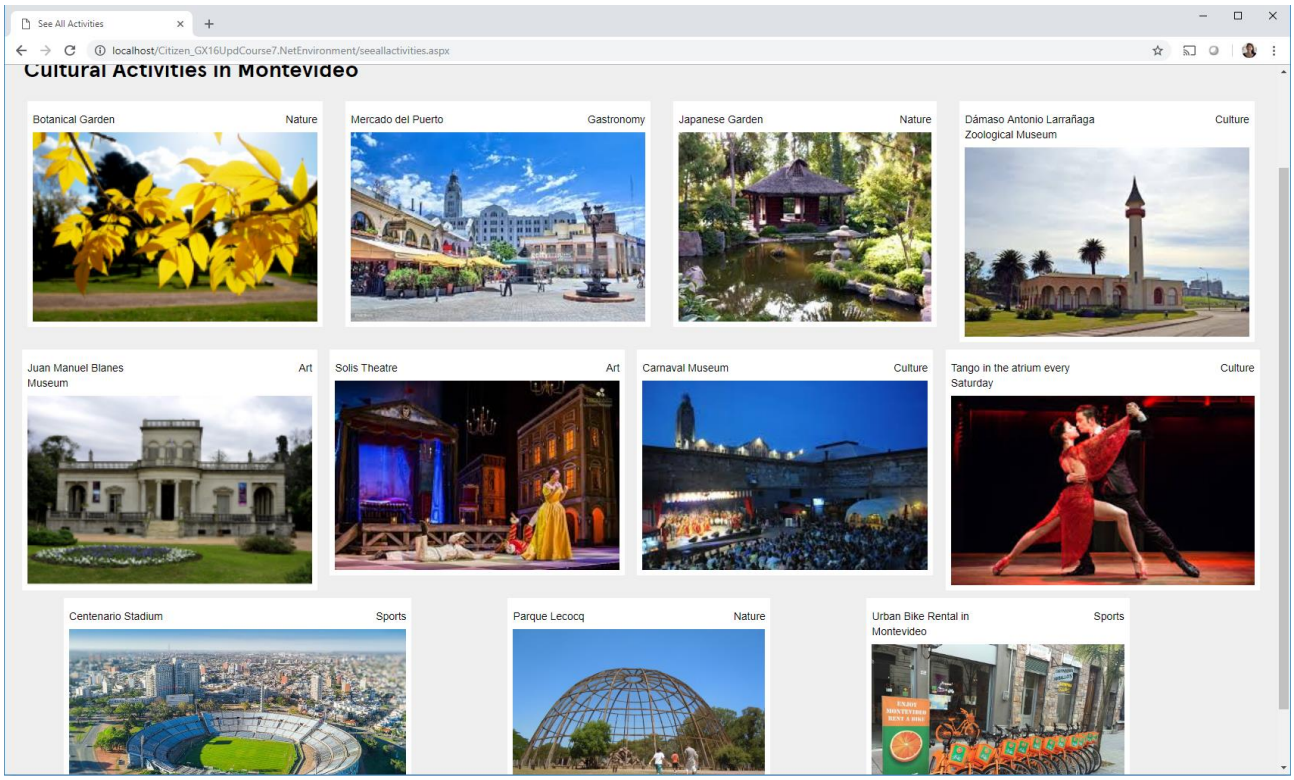
Observe en ejecución la diferencia con el horizontal grid.

FLEX GRID

Si observa el panel main del frontend web (de nombre CitizenWeb), además de ver las actividades culturales destacadas se ofrece una opción “See all Activities” para ver todas las actividades:

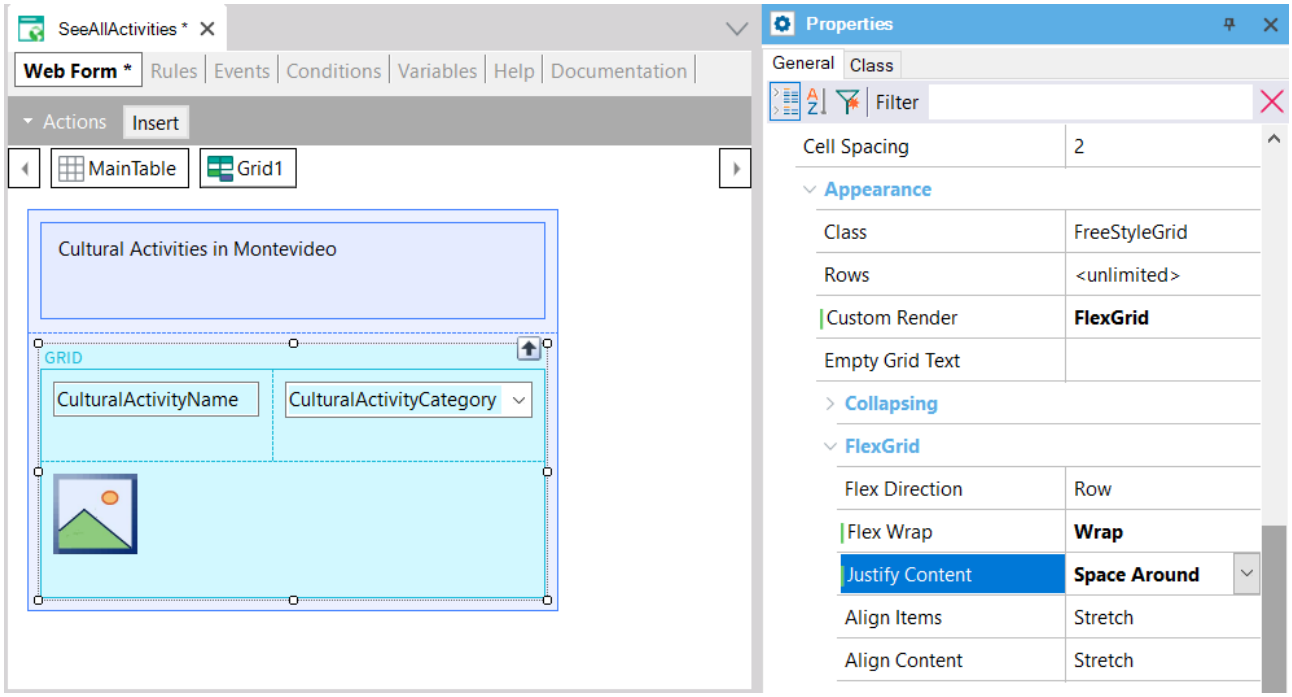


En esa otra pantalla queremos que el grid permita mostrarlas en forma variable, de acuerdo a su tamaño:



¿Cómo lo implementa?

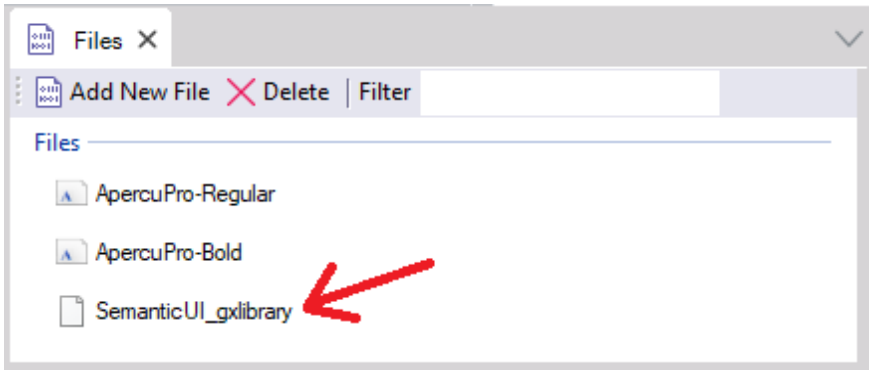
SOLUCIÓN:



Observe las diferencias con el Horizontal Grid que tenía antes.

BASE STYLE

Si edita los files de la KB encontrará que existe:



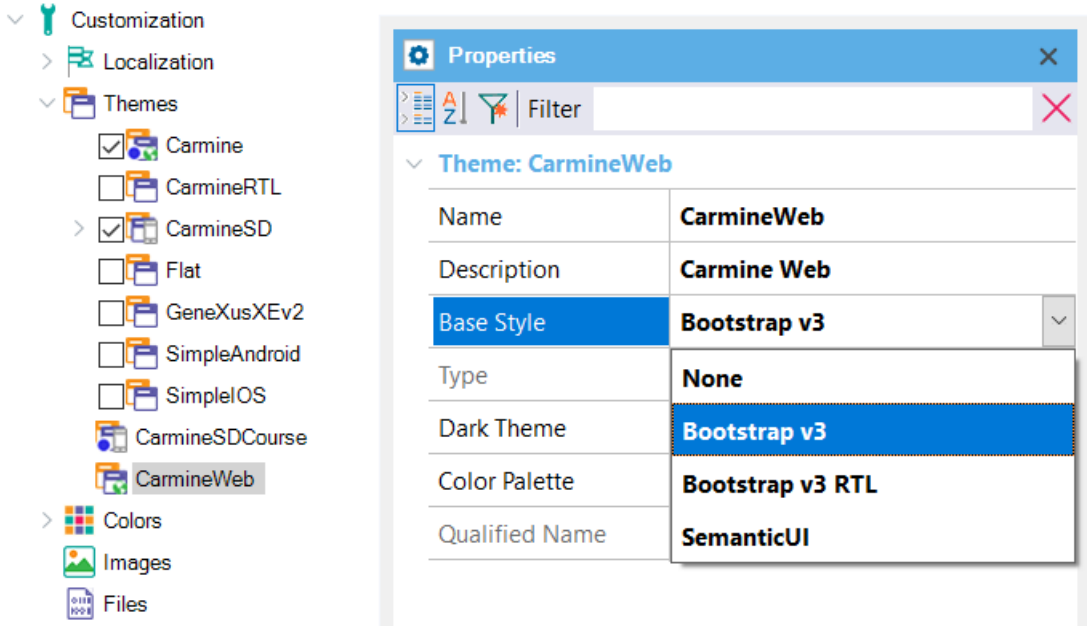
También encontrará una nueva propiedad “Base Style” en los objetos:

- Theme
- User Control

Esta propiedad permite asignar un estilo base a esos objetos.

En la propiedad se tomarán los valores de los Files de la KB que tengan extensión “gxlibrary” (son un zip que contiene todos los archivos css, js, assets, etc. provistos por los diseñadores o el Design System utilizado)

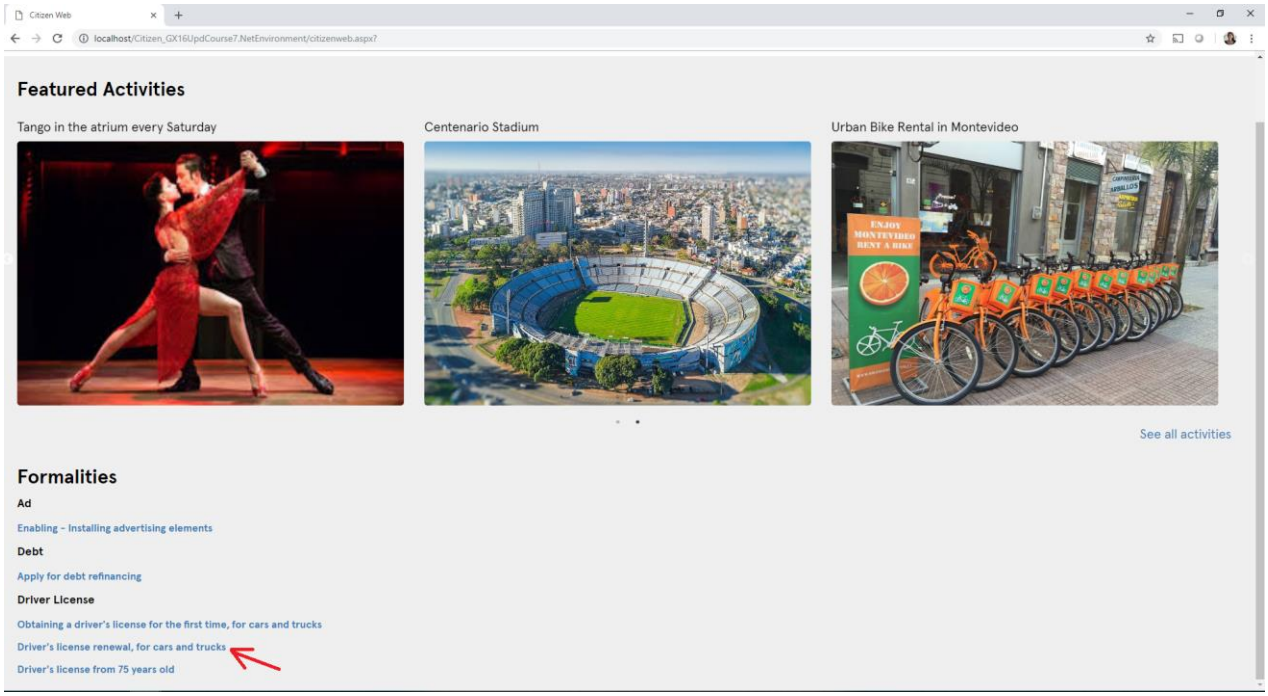
A nivel de theme encontrará la propiedad:



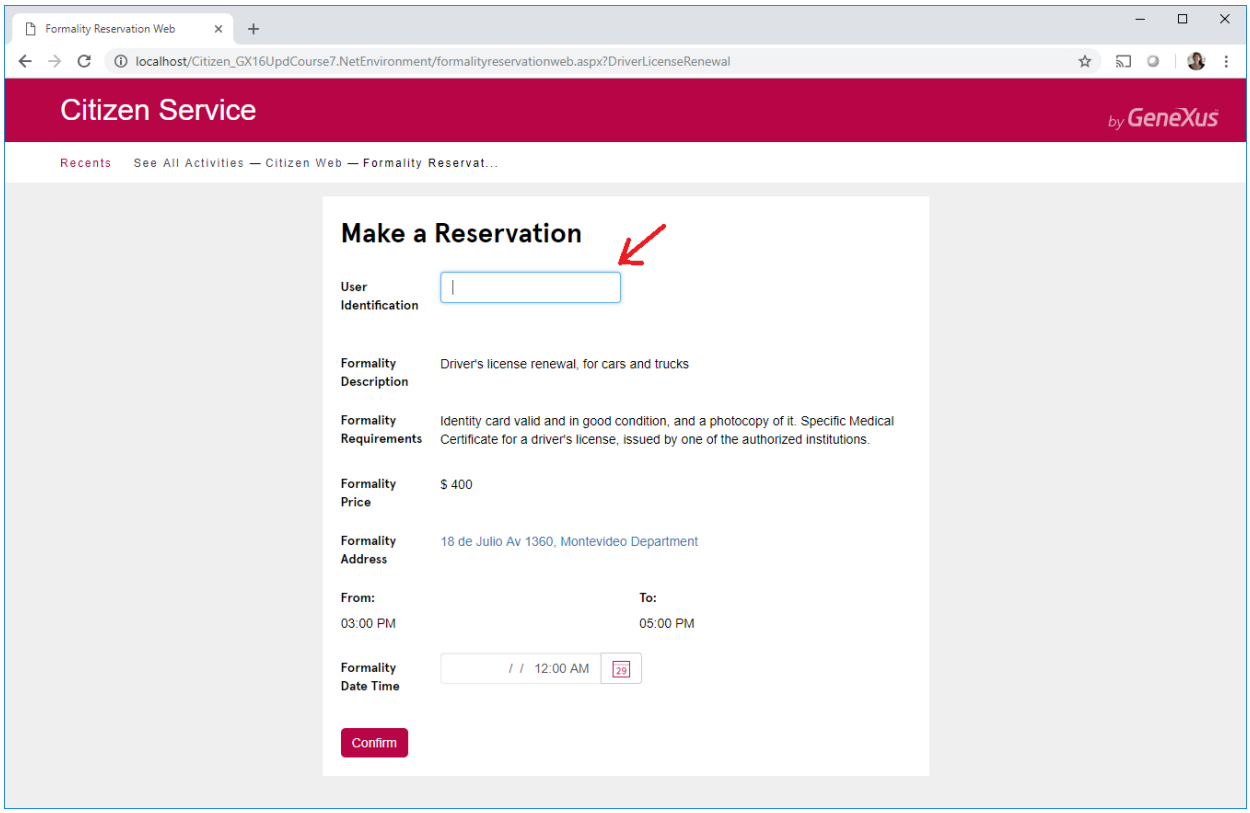
Y a nivel de objeto la veremos en seguida cuando creemos un User control.

USER CONTROL

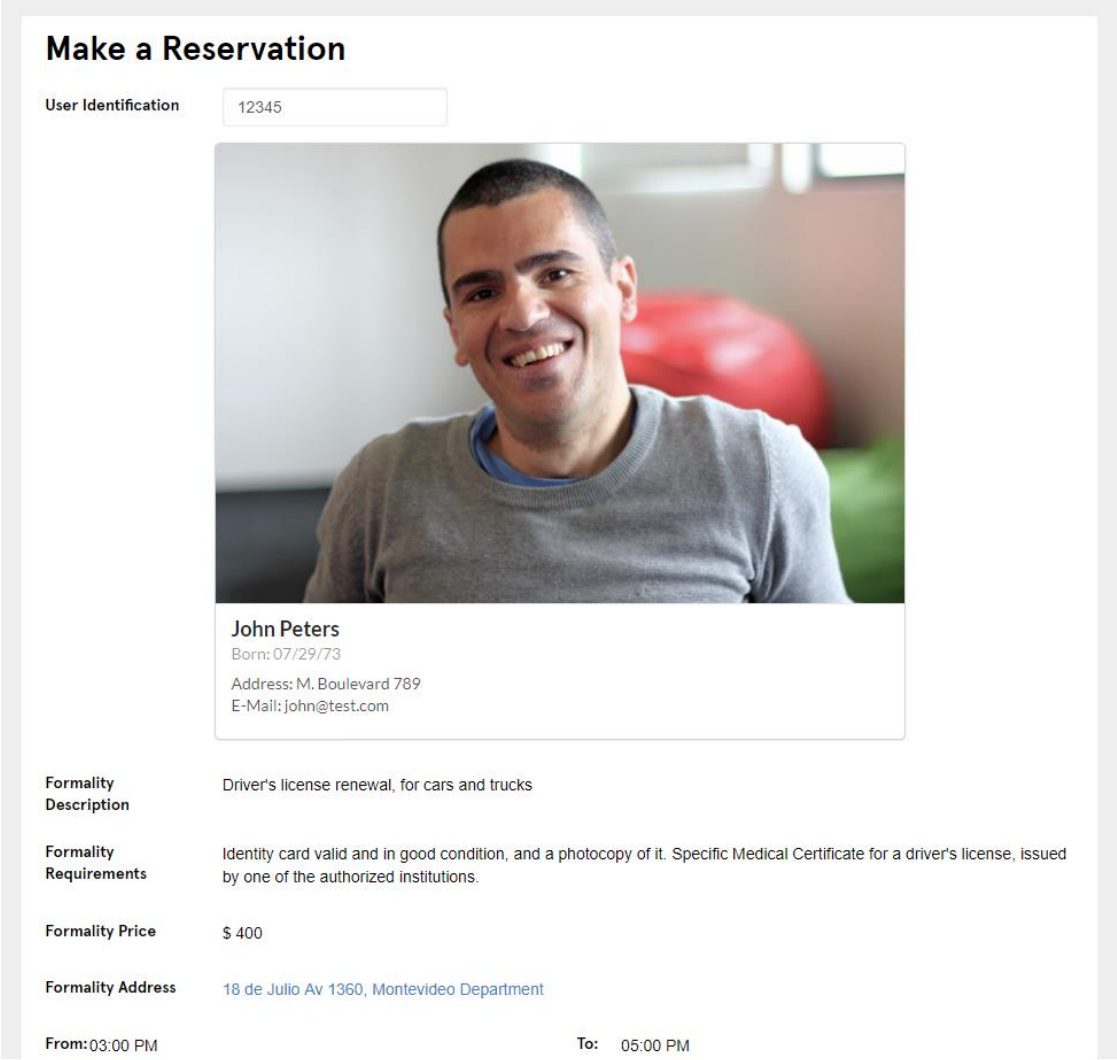
Si ahora desde la página principal del frontend web elegimos un trámite:



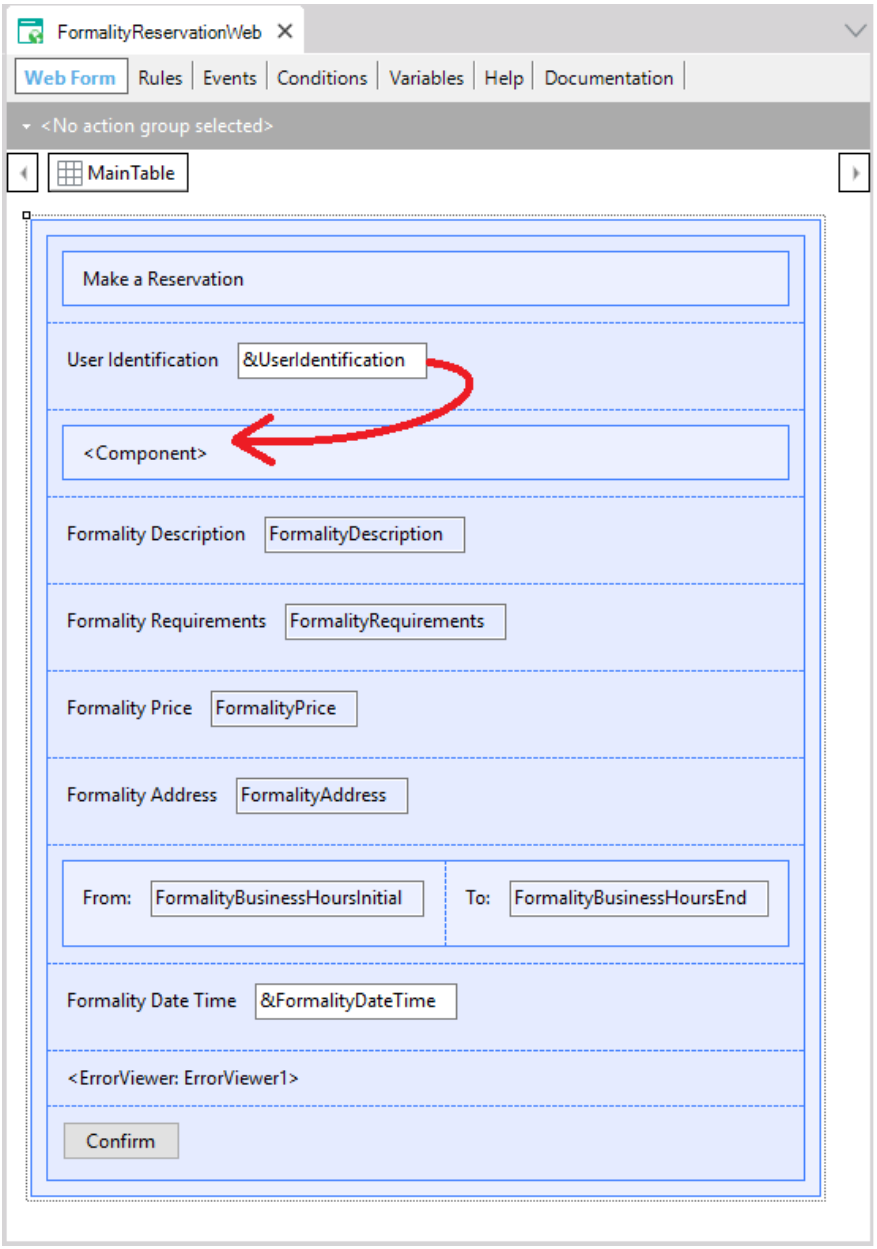
De manera similar a la app mobile nativa, nos pide identificador de usuario para poder reservar hora para ese trámite:



Al ingresar un ID de usuario válido y salir del campo, queremos ver la info del usuario de la siguiente manera:



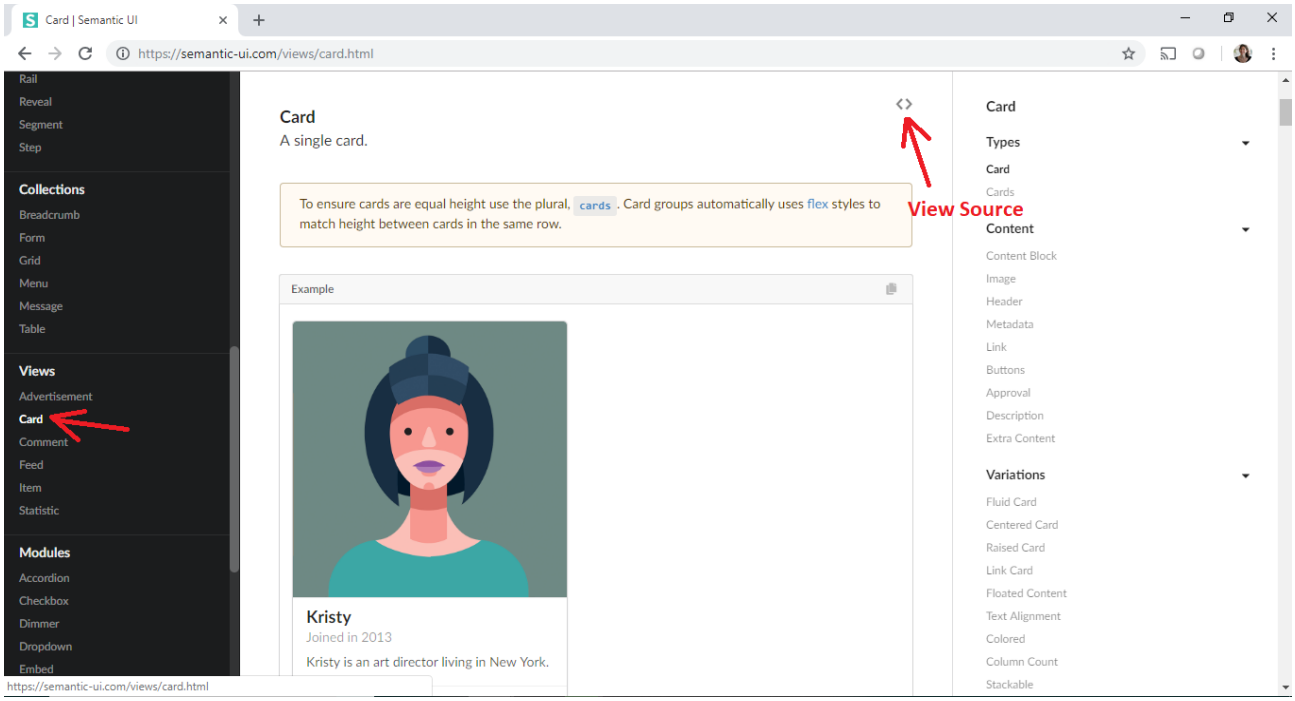
Observe que esa Card se carga en el componente de nombre WCUserInformation que se carga en el web form:



¿Cómo implementa ese componente (WCUserInformation) que en su KB está vacío?

SOLUCIÓN:

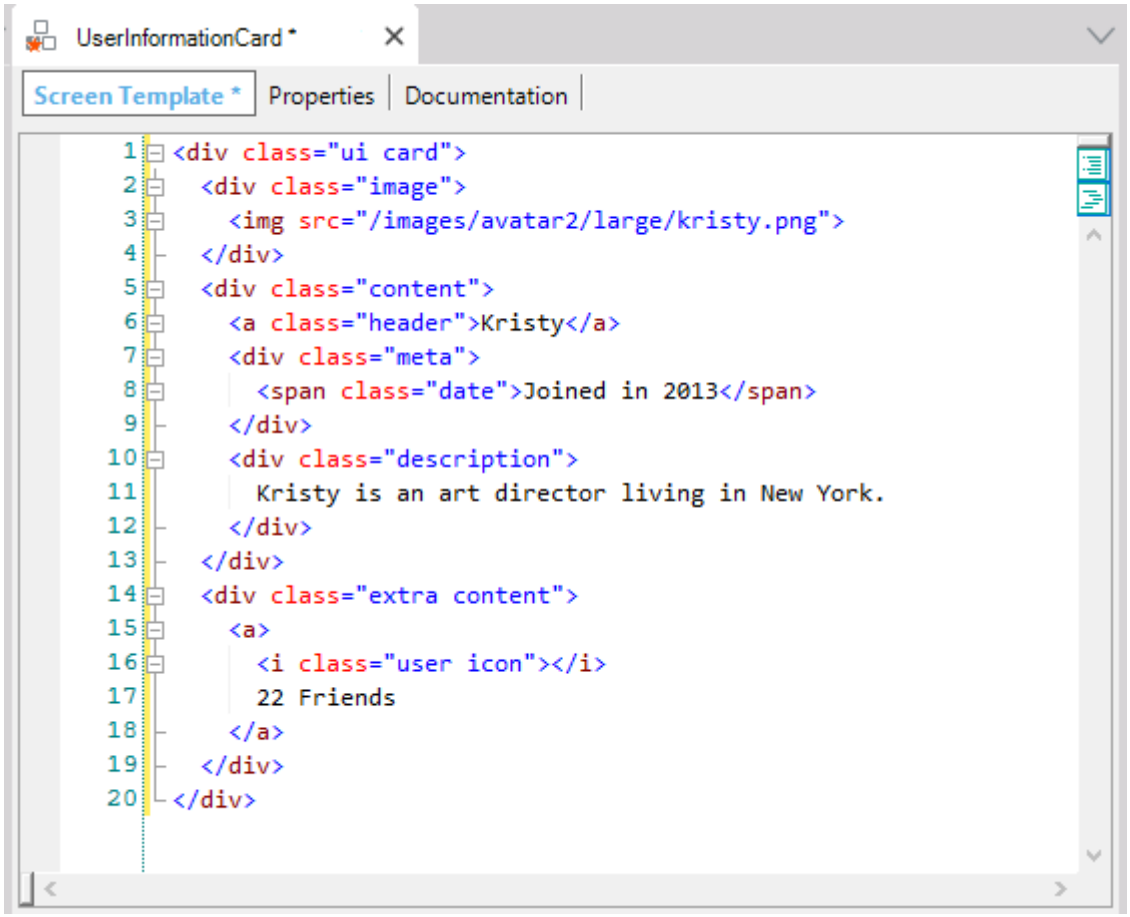
Se trata de un control Card de SemanticUI. Si va al sitio de SemanticUI, y busca por Card, encontrará distintos tipos de estos controles. Elija por ejemplo la Card más simple y visualice el html, para poder copiarlo:



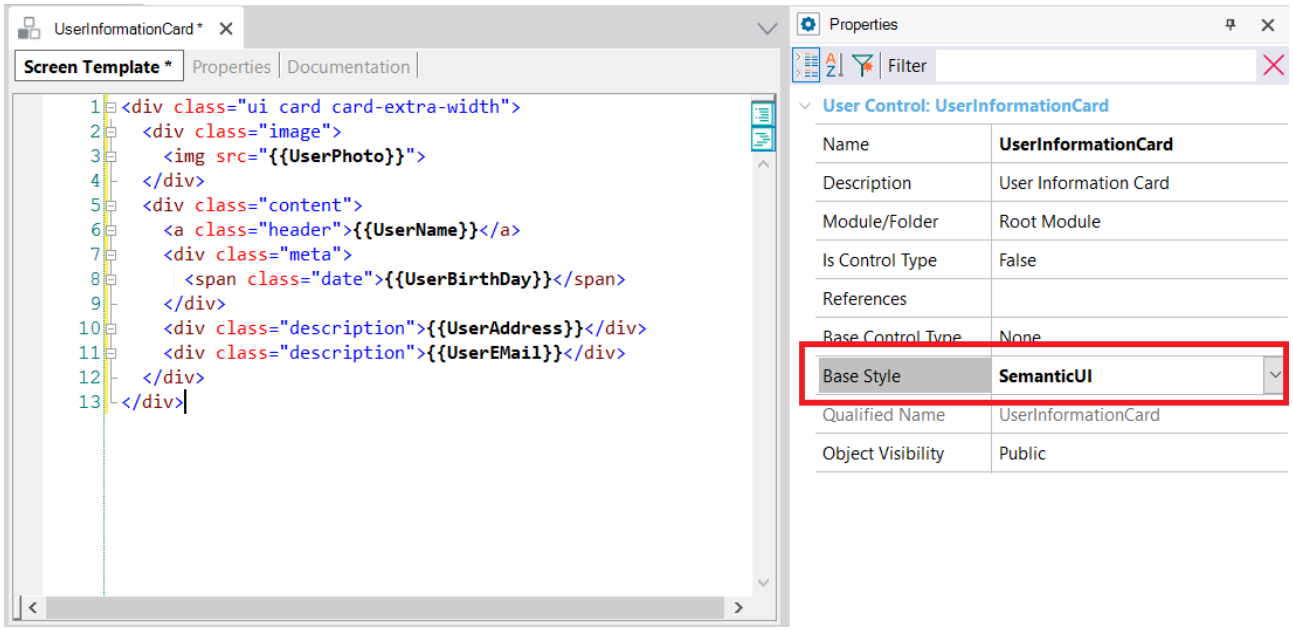
Aquí se lo damos escrito para que pueda copiarlo luego y no perder tiempo:

```
<div class="ui card">
  <div class="image">
    <img src="/images/avatar2/large/kristy.png">
  </div>
  <div class="content">
    <a class="header">Kristy</a>
    <div class="meta">
      <span class="date">Joined in 2013</span>
    </div>
    <div class="description">
      Kristy is an art director living in New York.
    </div>
  </div>
  <div class="extra content">
    <a>
      <i class="user icon"></i>
      22 Friends
    </a>
  </div>
</div>
```

Cree un objeto User Control en GeneXus y copie en su “Screen Template” el html anterior:



Parametrice las propiedades, de acuerdo a lo que desea ver. En nuestro caso nos quedará así:

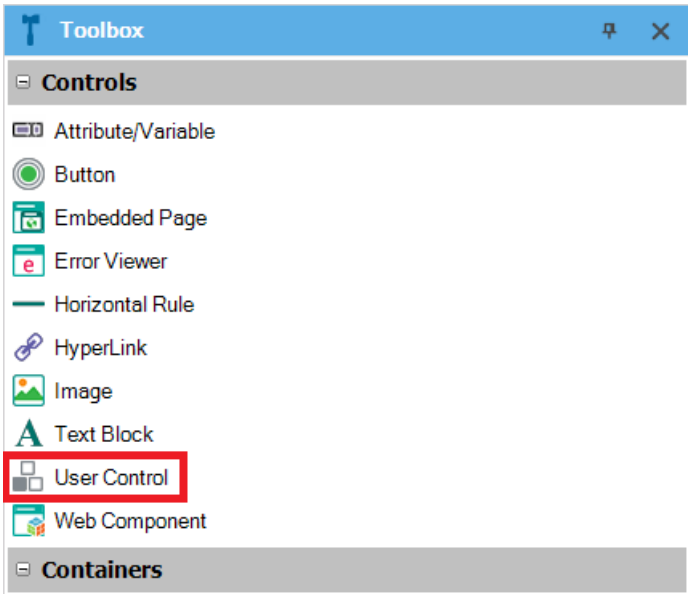


Aquí le damos escrito el código anterior para que pueda copiarlo:

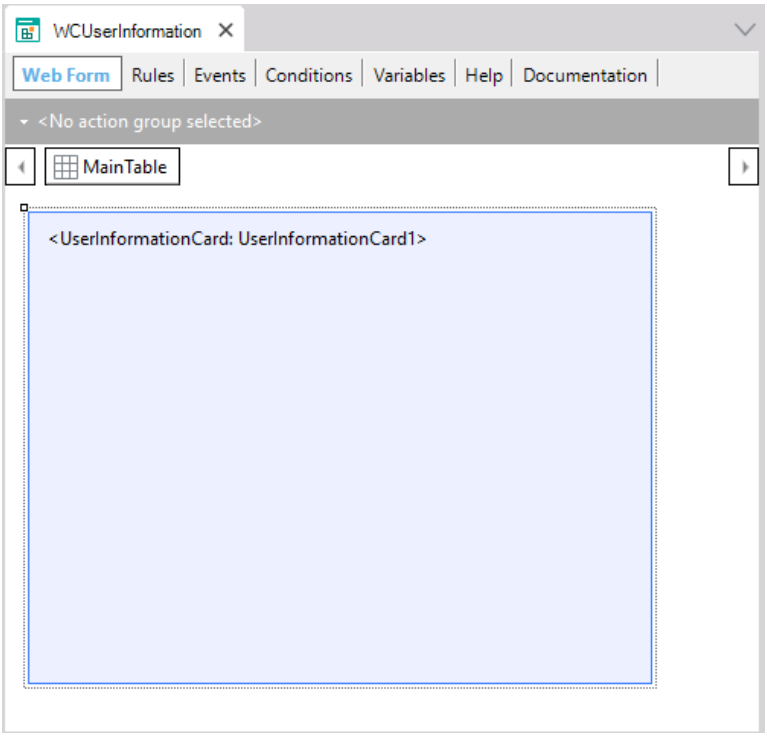
```
<div class="ui card card-extra-width">
  <div class="image">
    <img src="{{UserPhoto}}">
  </div>
  <div class="content">
    <a class="header">{{UserName}}</a>
    <div class="meta">
      <span class="date">{{UserBirthDay}}</span>
    </div>
    <div class="description">{{UserAddress}}</div>
      <div class="description">{{UserEMail}}</div>
  </div>
</div>
```

Revise que la propiedad Base Style del objeto User Control tenga el valor “SemanticUI” definido. De lo contrario, seleccione ese valor.

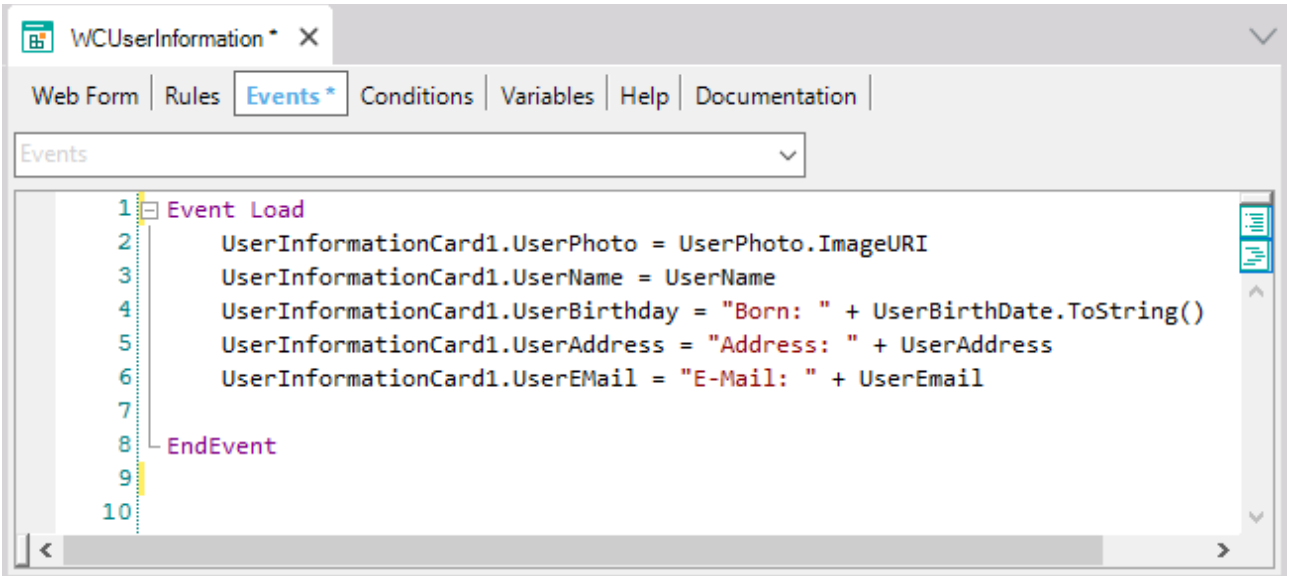
En el componente WCUserInformation deberá insertar el user control que acaba de crear. En la Toolbox encontrará:



Arrástrelo al Web form y seleccione el User Control que creó antes:



Luego, inicialice las propiedades del User Control en base a los atributos:

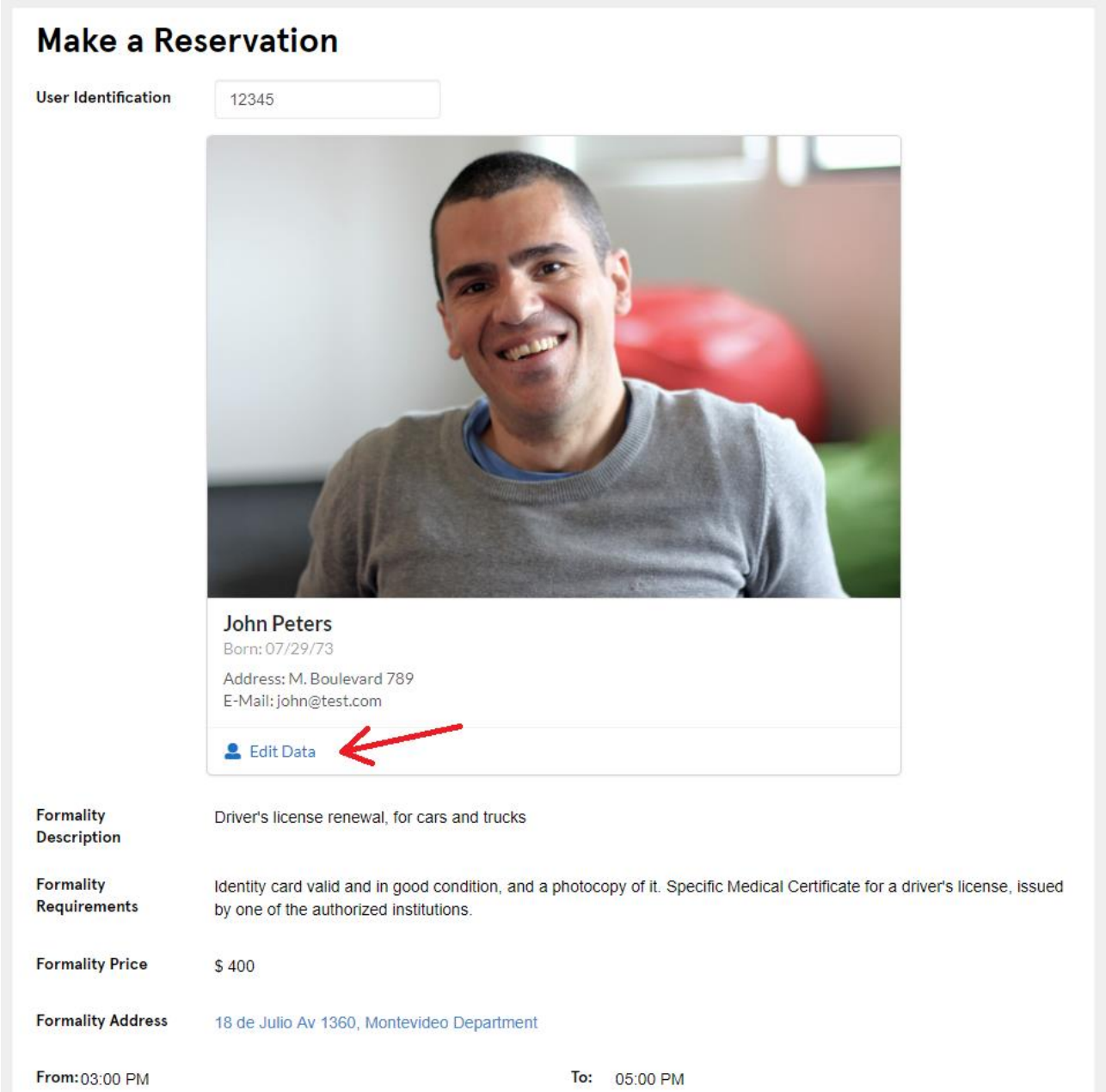


Nota: Para corroborar que se haya cargado correctamente la Base Library (en este caso SemanticUI), revisar el directorio web de la KB (Tools → Explore Target Environment Directory) y ver si existe el folder "SemanticUI". Si no existe, sugerimos realizar un Rebuild del objeto CitizenWeb.

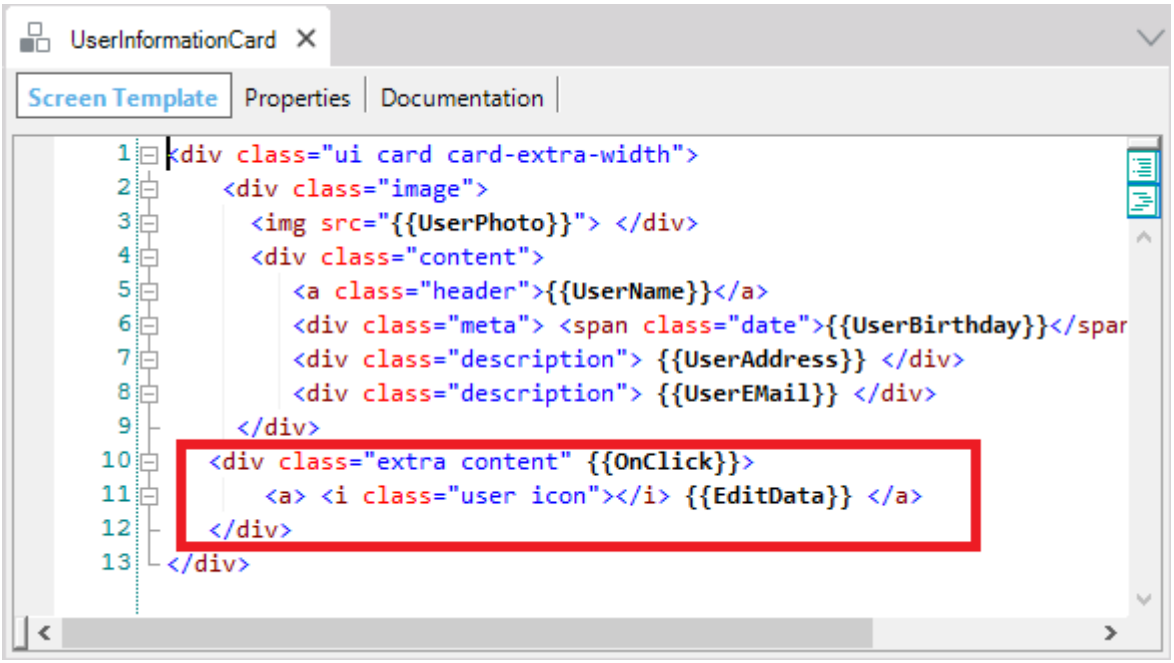
Ejecute.

AGREGAR UNA ACCIÓN AL USER CONTROL

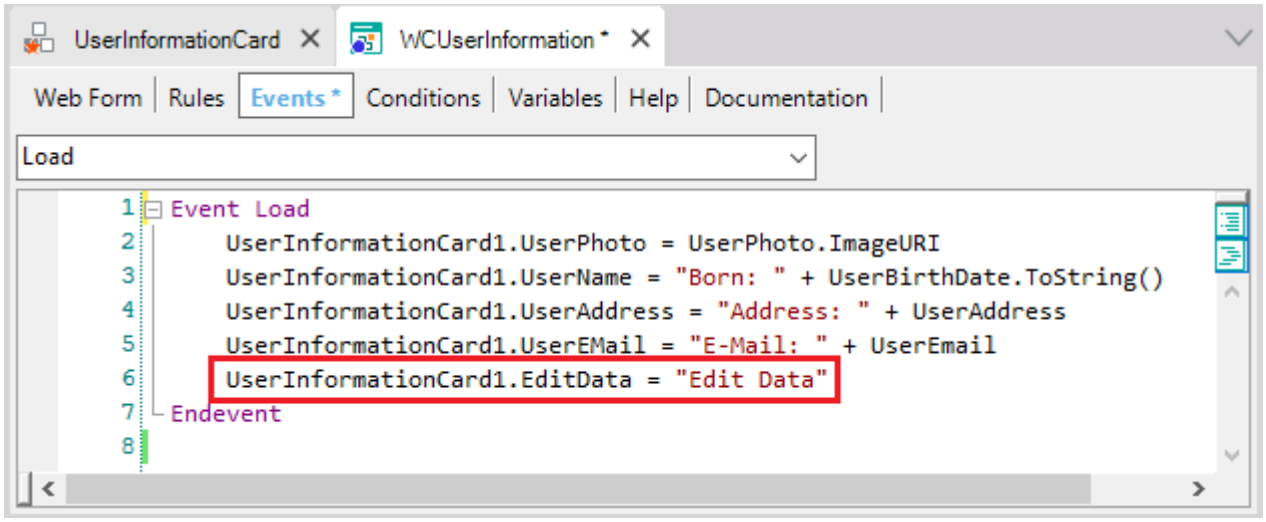
Opcionalmente, puede querer que se permita desde la Card editar la información del usuario.



Eso no venía exactamente así en el html source del control de SemanticUI. Puede personalizar su objeto User Control, agregándole:

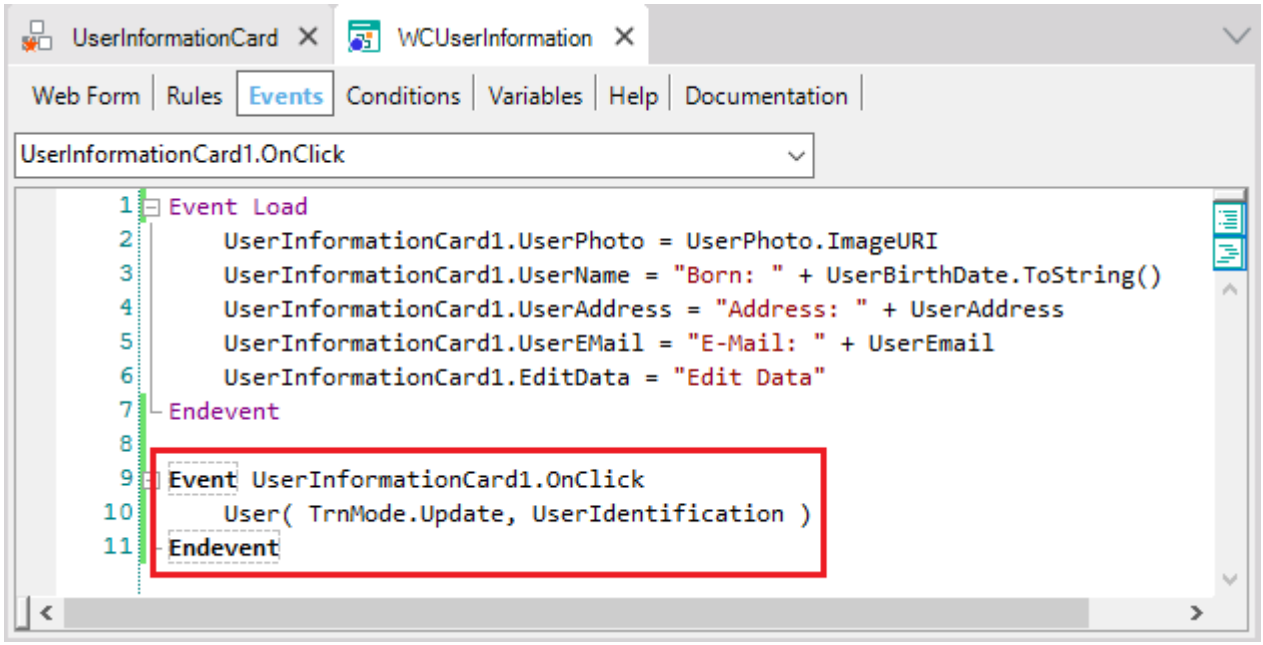


Y luego en el Web component:



Pruébelo en ejecución.

Falta programar el evento. Para ello:



Vuelva a probar.

KB SOLUCIÓN

Podrá descargar de GeneXus Server la KB solución de este práctico para comparar resultados. Es la versión de la KB de nombre CitizenSolutionPartOne.