

# Practice course for upgrade to

# GeneXus™ 16

## PART 1

April 2020

*Copyright © GeneXus S.A. 1988-2020.*

*All rights reserved. This document may not be reproduced by any means without the express permission of GeneXus S.A. The information contained herein is intended for personal use only.*

**Registered Trademarks:**

*GeneXus is trademark or registered trademark of GeneXus S.A. All other trademarks mentioned herein are the property of their respective owners.*

**CONTENTS**

CONTENTS..... 2

OBJECTIVE..... 3

GETTING STARTED..... 4

BECOMING FAMILIAR WITH THE KB ..... 4

STENCILS ..... 6

**Justification for using stencils..... 6**

Solution: ..... 11

**Creating and using a stencil..... 12**

Possible solution: ..... 12

SMART GRID ..... 15

Solution: ..... 16

FLEX GRID ..... 17

Solution: ..... 19

BASE STYLE..... 19

USER CONTROL..... 20

Solution: ..... 23

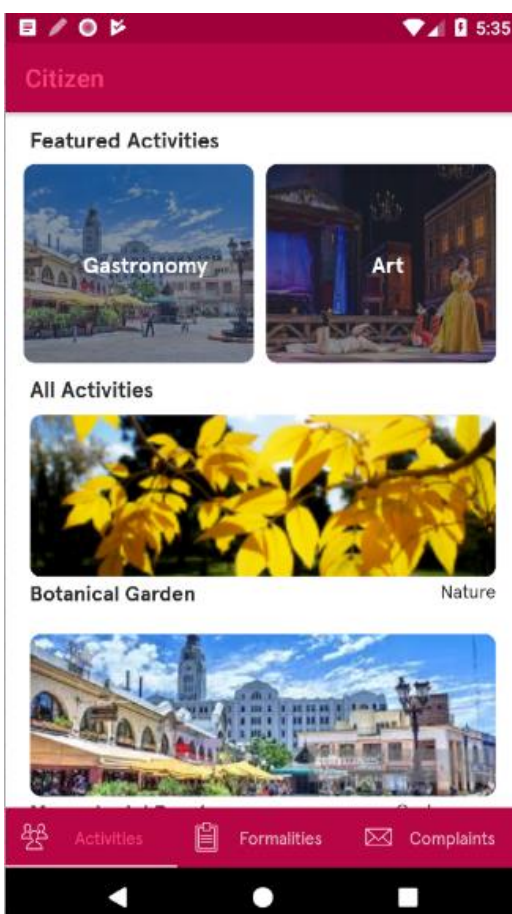
**Adding an action to the user control ..... 27**

KB SOLUTION..... 30

## OBJECTIVE

In order to achieve the objective of developing multi-experience and omni-channel applications while enhancing our integration with the design team, this practice session will deal with the topics of Design Systems and user experience related to the ways in which we can make repetitive information to flow more smoothly and in a manner more adaptable in the UI, all of which relates to the new grid types.

We will be working on an application made upon a simplification of an app for a city's municipality that has a Web frontend and an SD frontend for the citizens of that city to use a user identifier for filing claims (such as fallen trees, traffic lights out of order, wrongly parked vehicles and so on), as well as other procedures like obtaining a driver's license, making arrangements for financing amounts due, installing public advertisements, etc.). With the app, they may schedule a date for making their claims or procedures with the city's personnel. The frontend also shows the various cultural activities that the city has to offer:



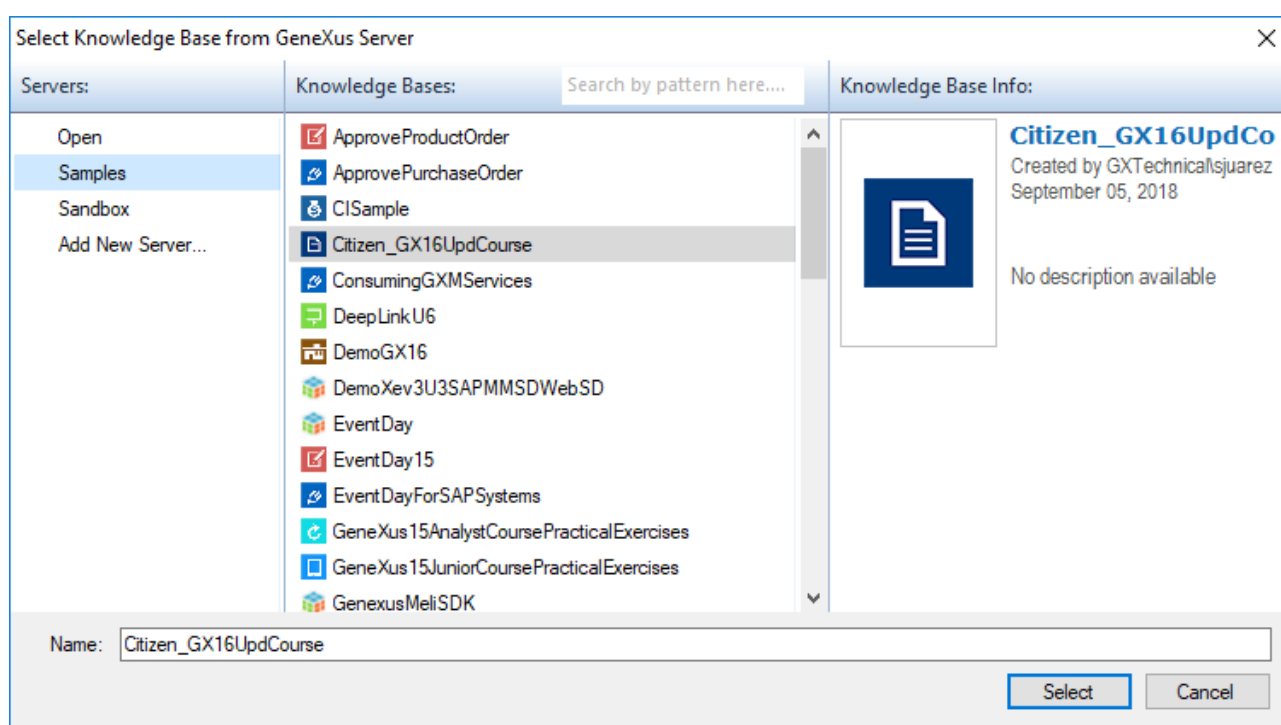
There is also a web backoffice for some employees to manage data and check statistics.

## GETTING STARTED

We will be using the **GeneXus 16** version and the **Android SDK** emulator. The instructions of these practice exercises have been updated according to upgrade 9 of GeneXus 16. If you do them with a later version, remember that there may be differences between what is shown here and what you see.

Unless the instructor tells you that the KB has already been created in your computer, in GeneXus do the following: File/New Knowledge Base from GeneXus Server, selecting Server KB URL: <http://samples.genexusserver.com/v16/>.

In Select Server KB, select the name “Citizen\_GX16UpdCourse”:

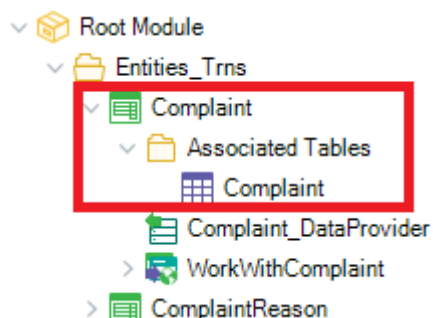


And select the trunk version.

Do a Rebuild All and Create. We will be doing local prototyping and not in the cloud. Make sure that the names of server and database are already set up, or set them up with the instance of the computer’s SQLServer and DB name as you wish. Ask your instructor.

## BECOMING FAMILIAR WITH THE KB

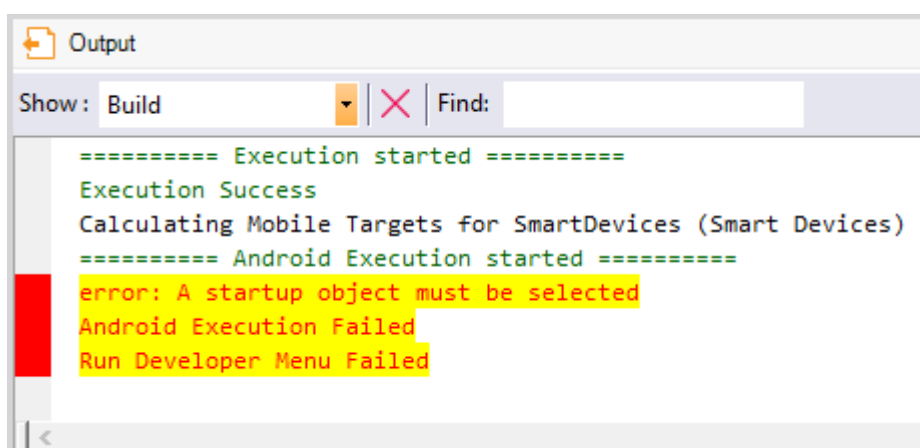
The entities that model the application’s reality are in the Entities\_Trns folder. Note that now, tables are shown under each transaction in the KB Explorer.



Also note that the Data Providers have been defined to populate the tables associated with the transactions (linking the Data Provider property of the transaction). They will be invoked automatically upon the first execution.

There are also two folders, FrontendWeb and FrontendSD, that implement the two frontends, with the main objects of the app that the citizens will use.

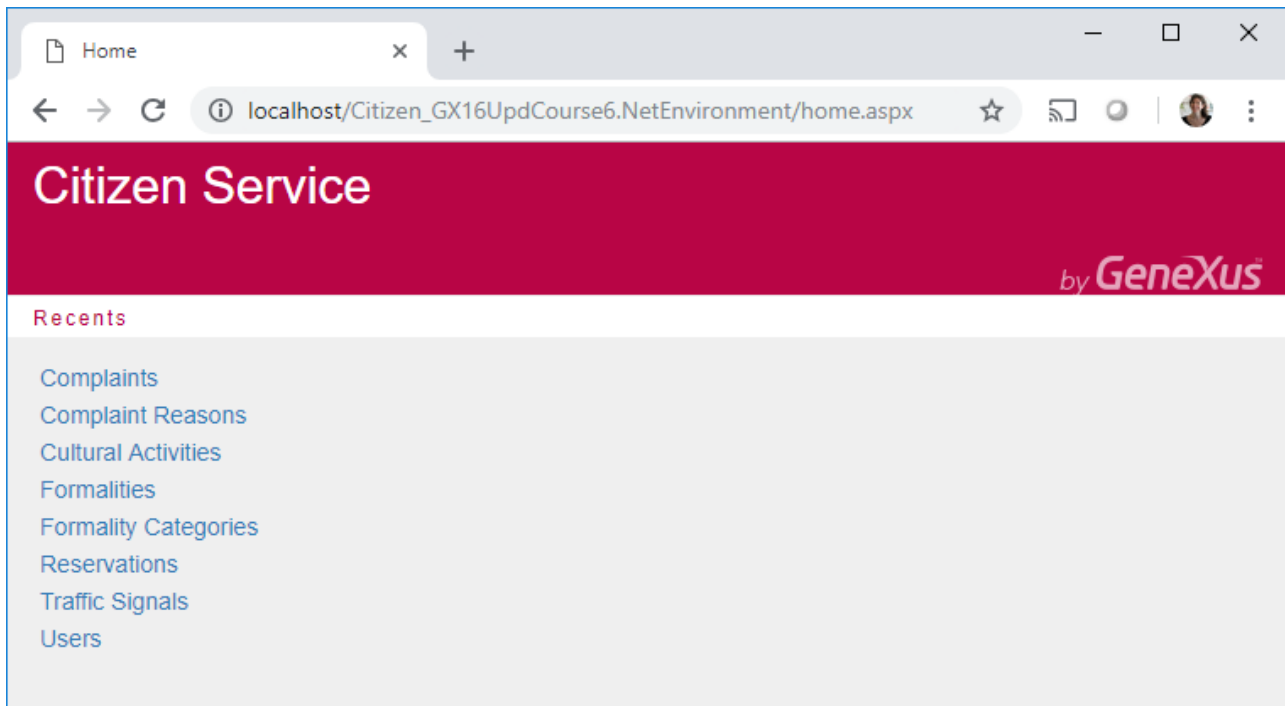
Web panel Home inside GeneXus/Web contains the backoffice's main object (it is default) that will be used by the city's staff for managing information and the site. It will be deployed as the outcome of the F5, which in turn will display the following error:



This is due to the fact that we have not set up a Startup object, because we will be moving between Web and SD frontend and backoffice. We can decide to specify the Home of the backoffice as Startup Object, or not, because we will nevertheless have to do Run on the main object we want execute every time (CitizenMenu for SD frontend, and CitizenWeb for Web frontend).

The Queries folder includes queries that will be used in a different practice session.

When you execute the Home web panel:



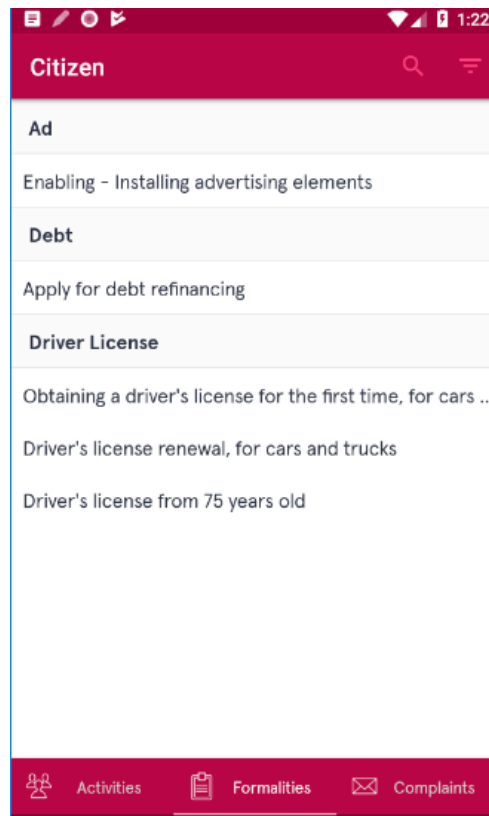
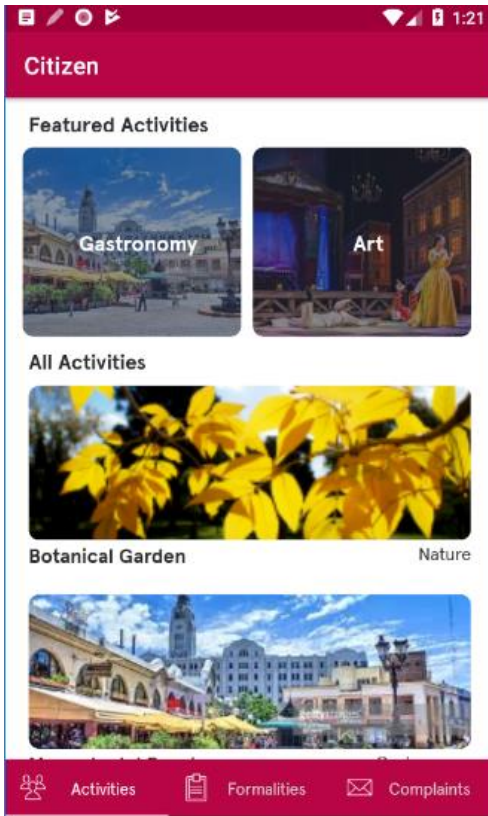
You will see the entities' WorkWiths with the data loaded. Leave that open.

## STENCILS

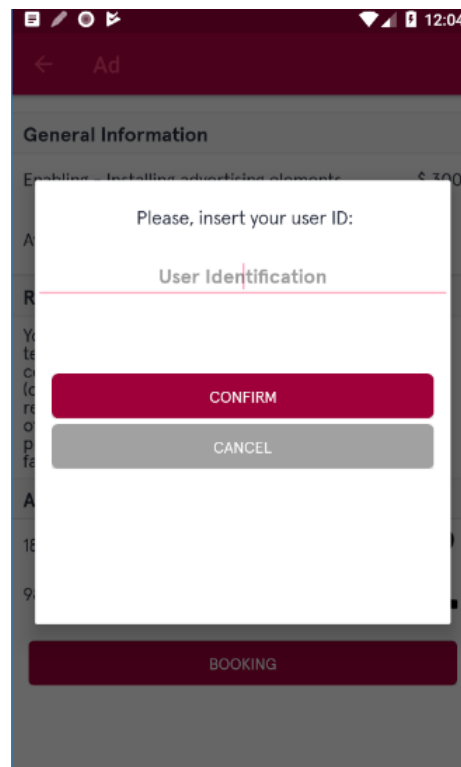
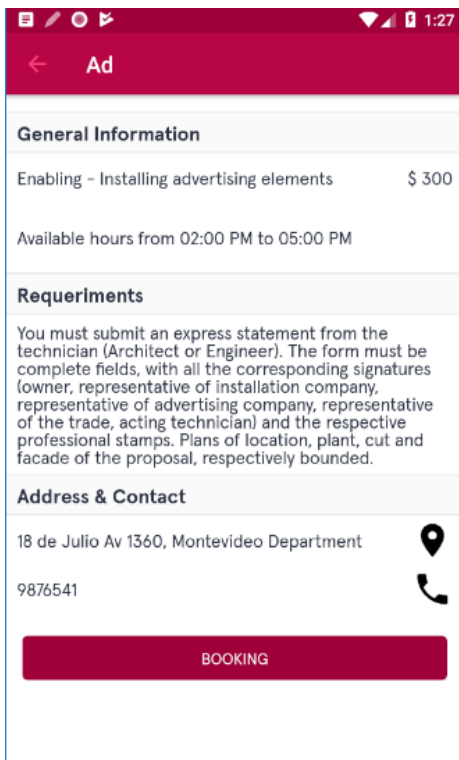
Execute the main object of the SD app (CitizenMenu object in the FrontendSD folder).

## JUSTIFICATION FOR USING STENCILS

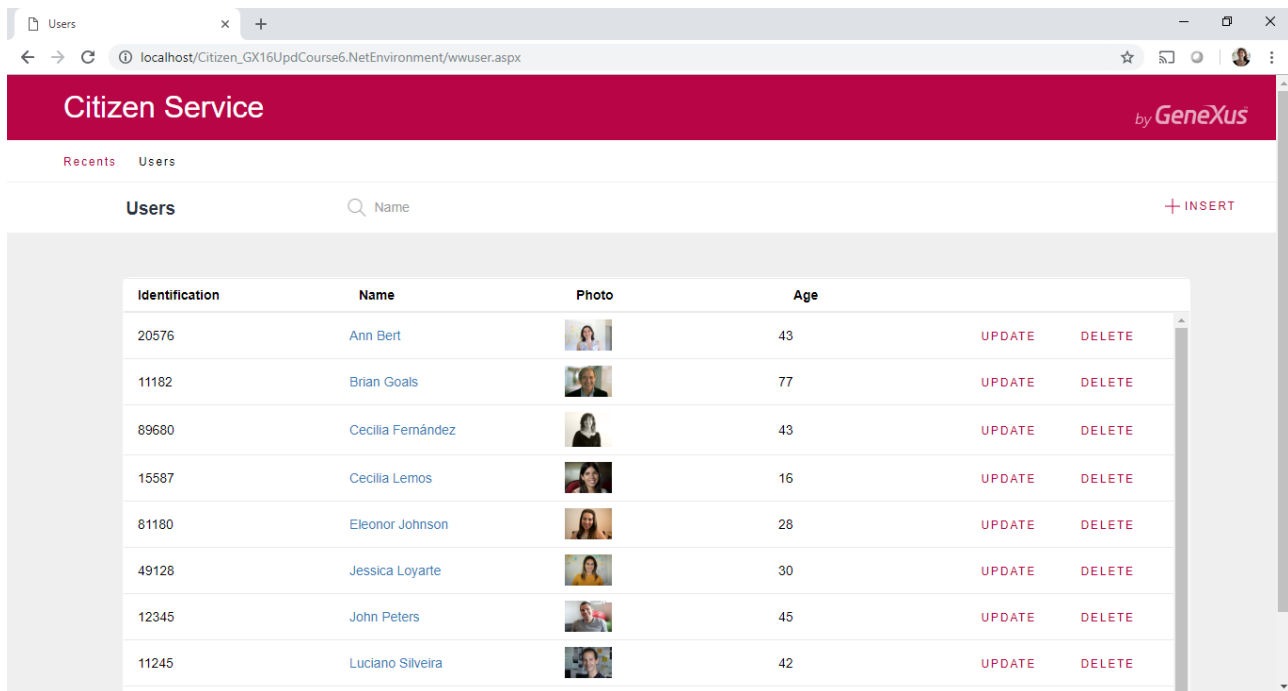
The first screen you will see shows the cultural activities (Activities). And in the menu's second tab underneath you will see the different claims and procedures available to citizens (installing public advertisements, arrangements for the financing of amounts due, obtaining a driver's license):



When the user selects a procedure, such as an advertisement authorization, the user is shown general information on the procedure and is allowed to book a reservation where he will be carrying out that procedure. When the user presses “BOOKING” he will be asked for the user ID:

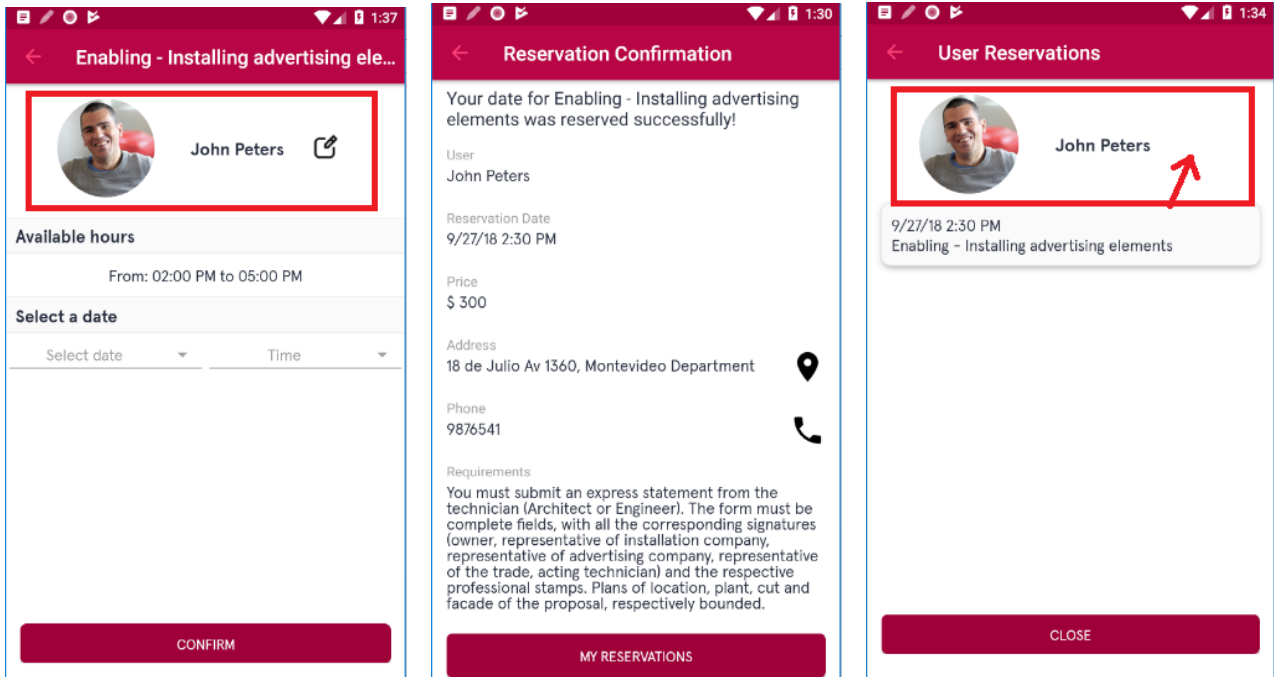


We recommend that you open the Backoffice to view the users registered and their IDs:



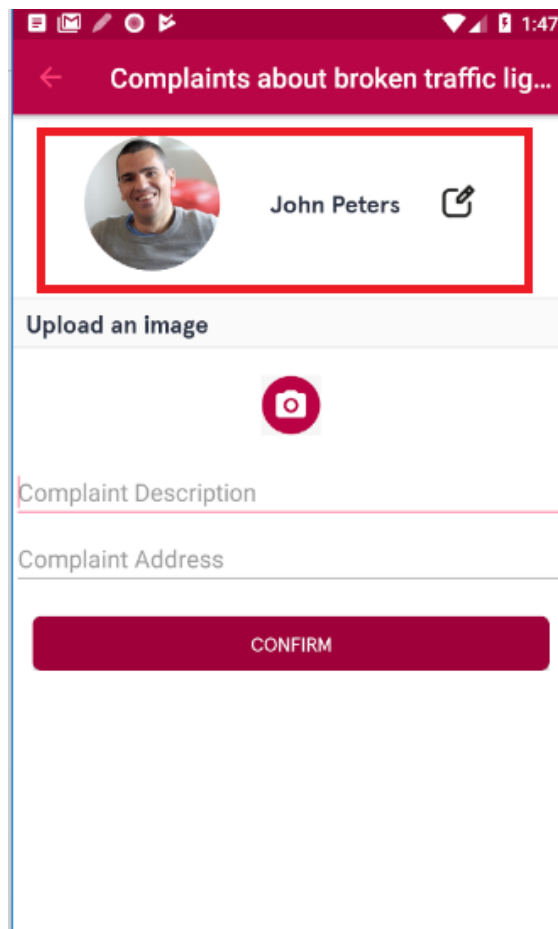
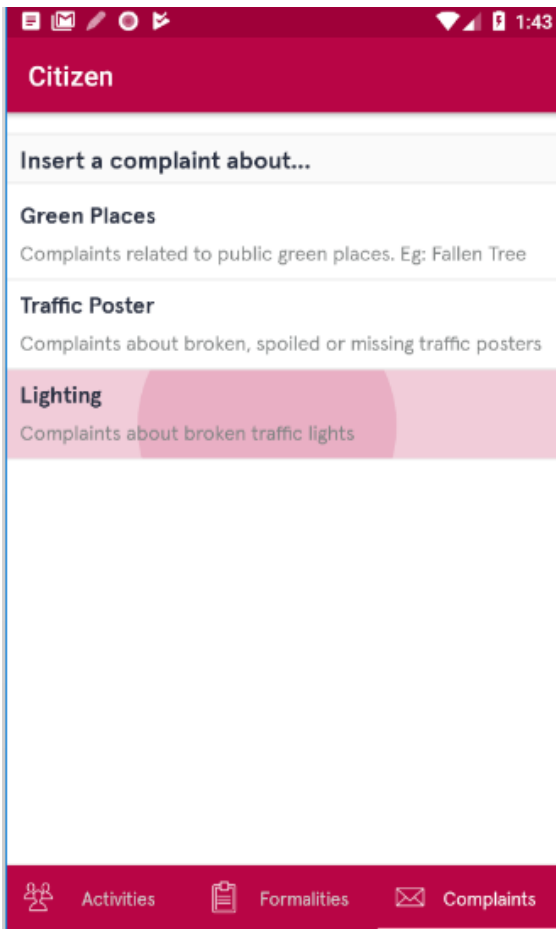
When you enter one and confirm (such as ID 12345), you get a screen showing a photo and the name of the user entered, with the possibility of booking a day and time, within the office hours allowed for the procedure, so as to book a reservation for carrying out the procedure. When you do this, if it all works out fine, you will get another screen showing all the information regarding that procedure, and you will be offered the option of viewing all the bookings you have made so far:





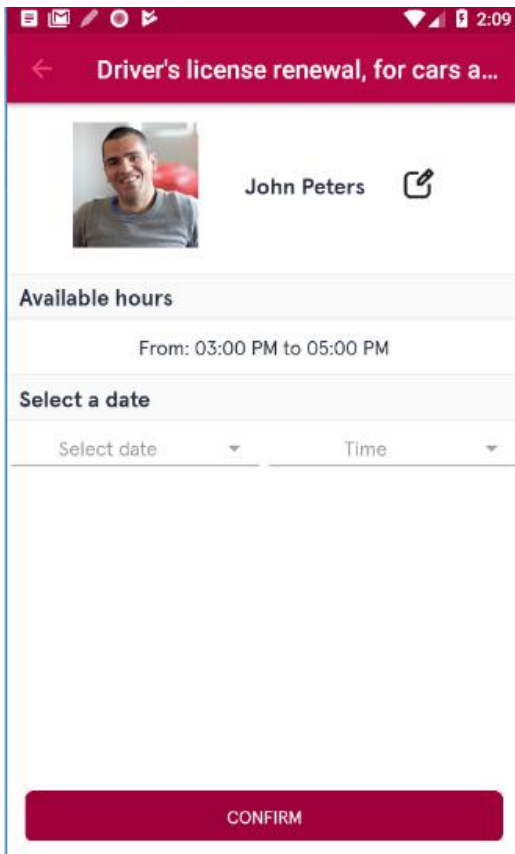
In this last screen, you will also view the user's photo and name, with the same design as previously, but with the possibility of changing users now hidden.

By going to the Complaints menu option and selecting the claim type, the user will be able to enter one of that type into the system:



Note that, on this screen, UserComplaint, the user information is also shown in the same manner as in the procedure.

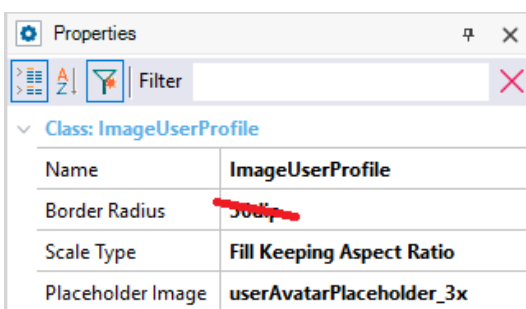
To modify the manner in which the photo is shown (on the three screens on which it appears) so that it will not appear with a round shape anymore, as shown here:



How could this be implemented?

SOLUTION:

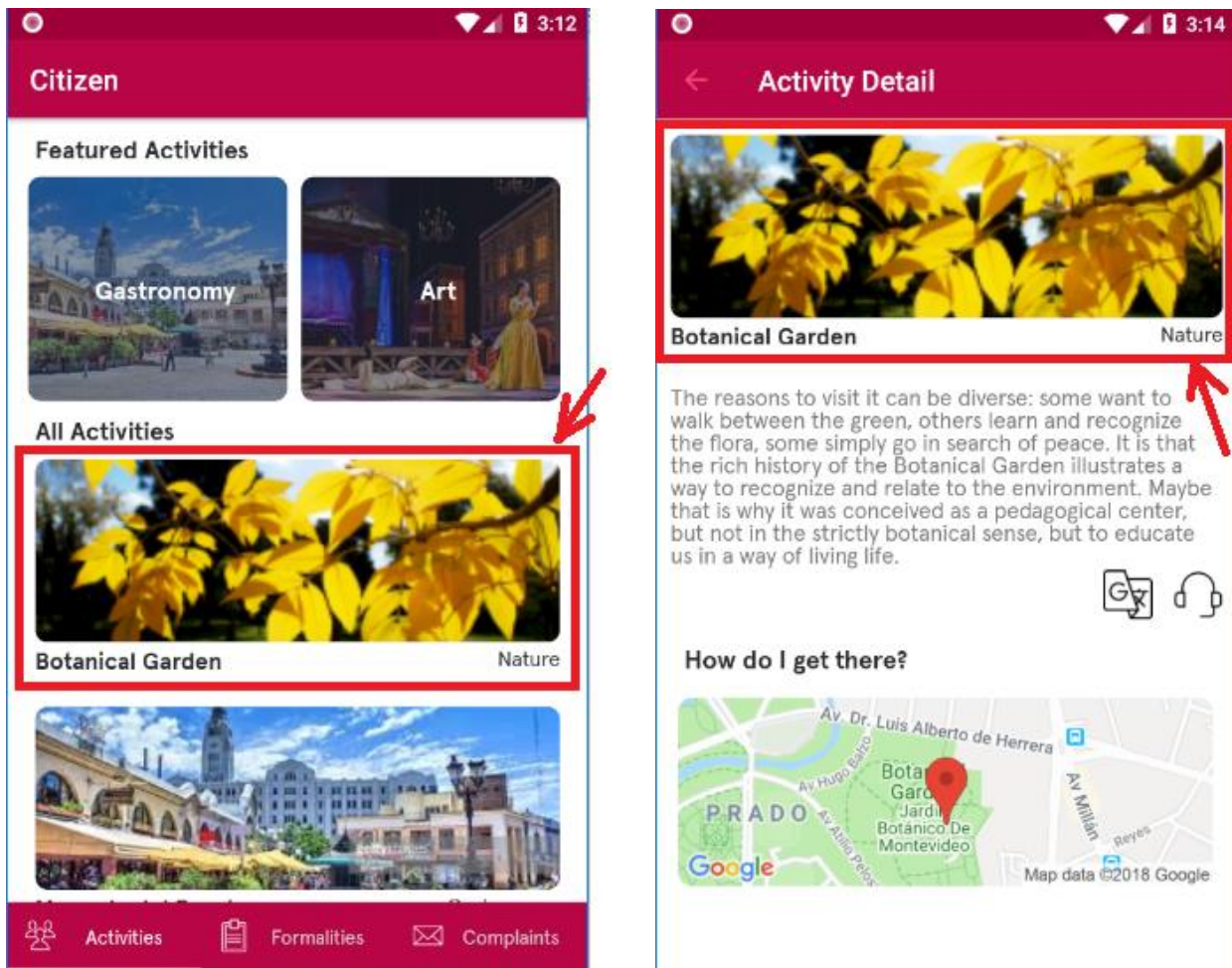
Open the UserProfile stencil and see that the photo is associated with the ImageUserProfile class of theme CarmineSDCourse. Change the Border Radius class and leave it empty.



Execute CitizenMenu and note the change on the three screens.

## CREATING AND USING A STENCIL

Now we want the screen that shows the cultural activities (Activities) to show, on top, the outstanding activities that are highlighted, with the remaining activities underneath. When you select an activity in either grid, you will view a screen with a detail:

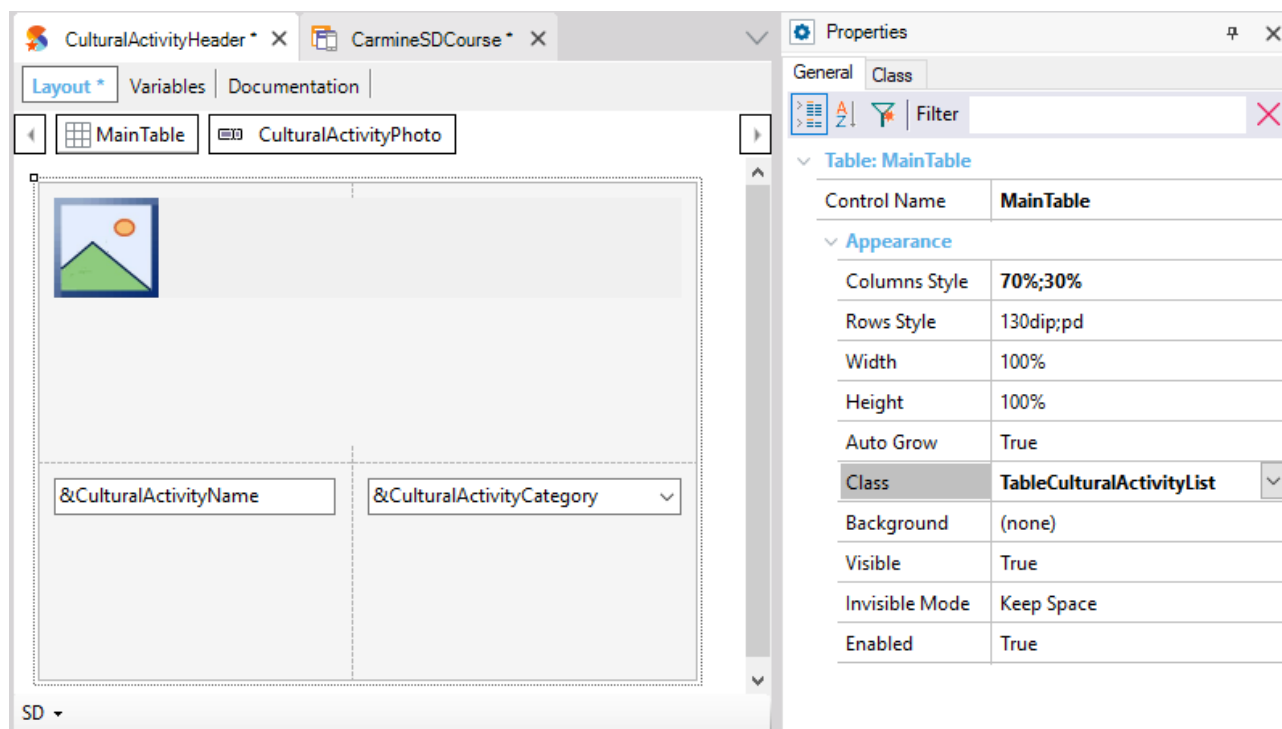


Note that the activity, as well as its name and category, are shown in a manner identical as in the grid that shows All Activities in the first panel and in the second panel. To avoid repeated designs we use stencils.

Implement a stencil to show this information and use it in both panels: Activities and ActivityDetail.

POSSIBLE SOLUTION:

Create the stencil called CulturalActivityHeader (you can do this manually, or you can click on the table that already contains the photo and both attributes in the Activities pane, and right-click to select Wrap as new stencil):



If you did it manually, set the properties for the variable with the image as follows:

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

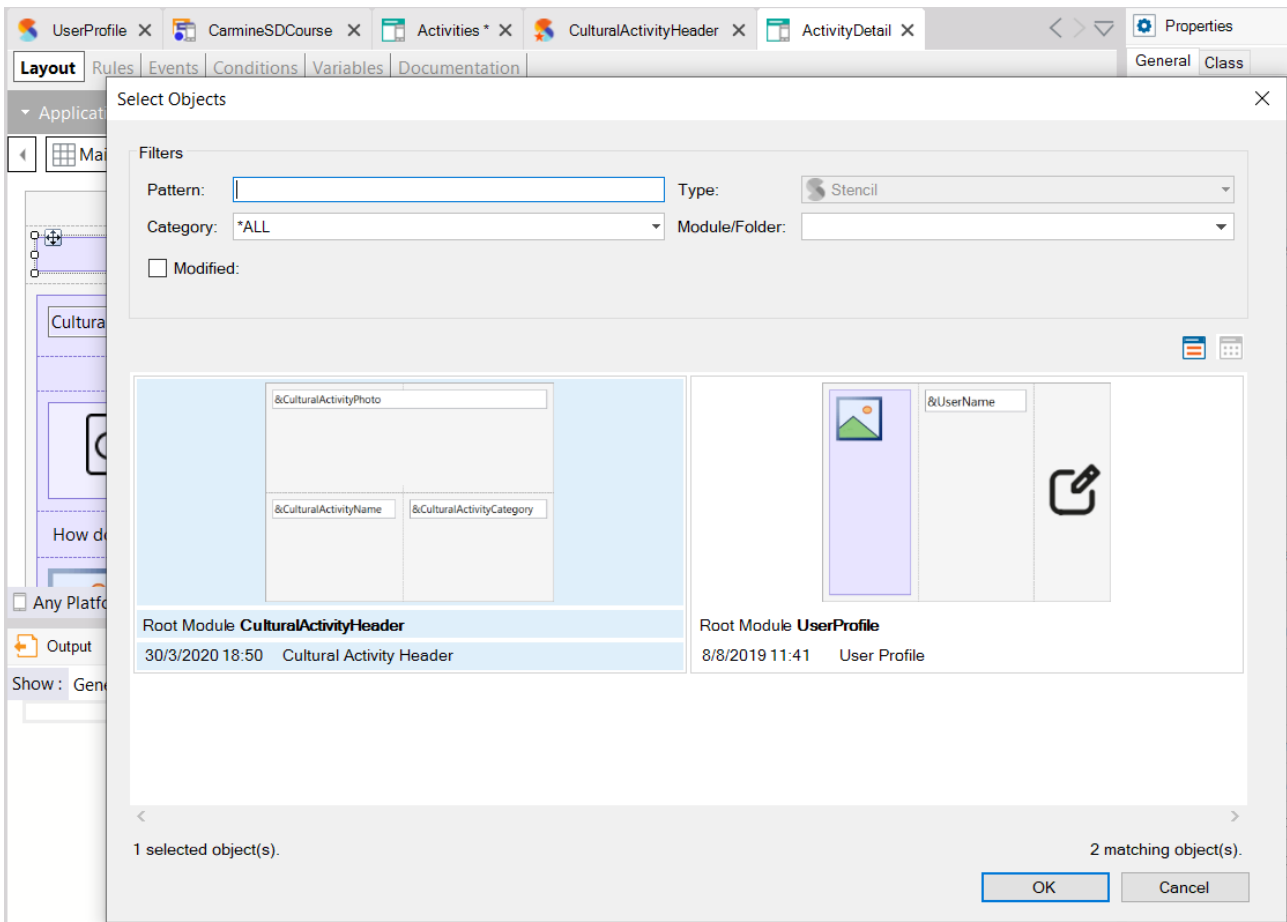
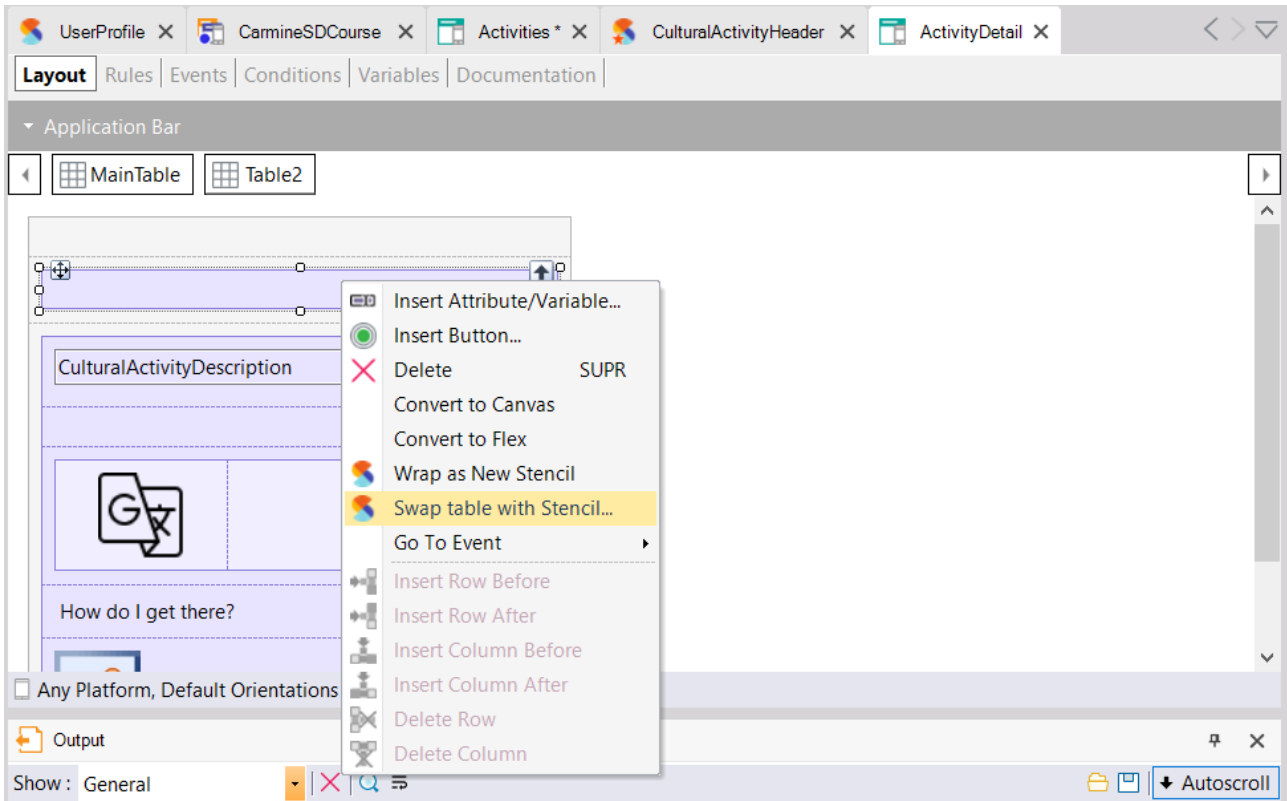
Set up the properties for the variable that shows the name on the left, as follows:

- Label Position: None
- Class: AttributeActivityNameList

Set up the properties for the variable that shows the category on the right, as follows:

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

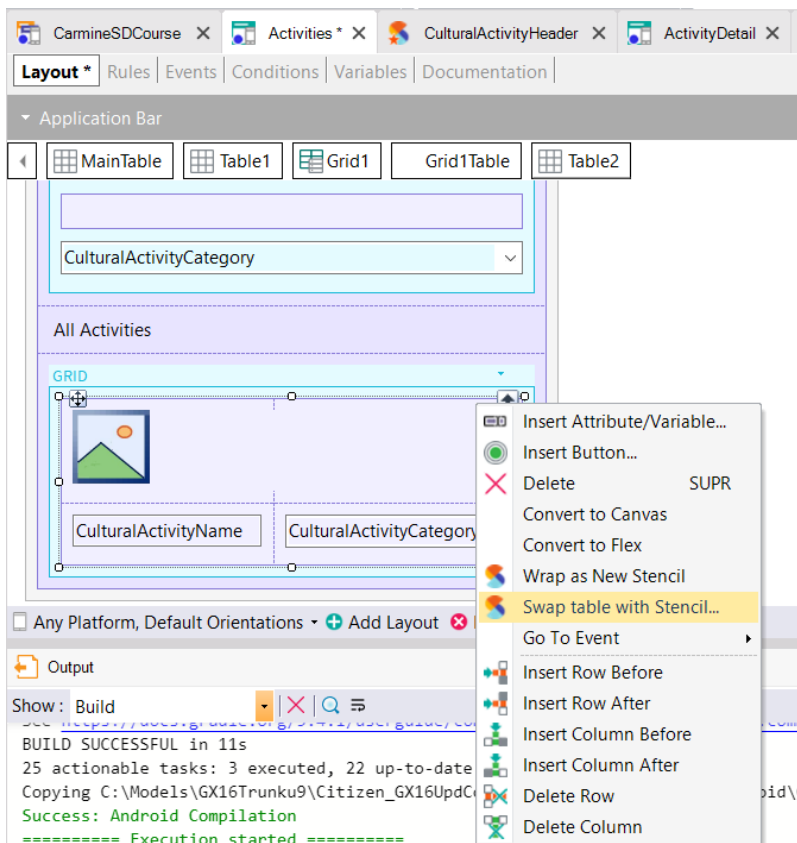
In the ActivityDetail panel, and in the table where you would insert the CulturalActivityPhoto, CulturalActivityName and CulturalActivityCategory attributes, now empty, search for the Stencil you have just created and insert it.



Note:

- You must replace the variables with the CulturalActivityPhoto, CulturalActivityName and CulturalActivityCategory attributes.

If you created the stencil manually, in the Activities panel go to the grid table that shows all the activities, right-click and note that, when you select “Swap table with stencil”:

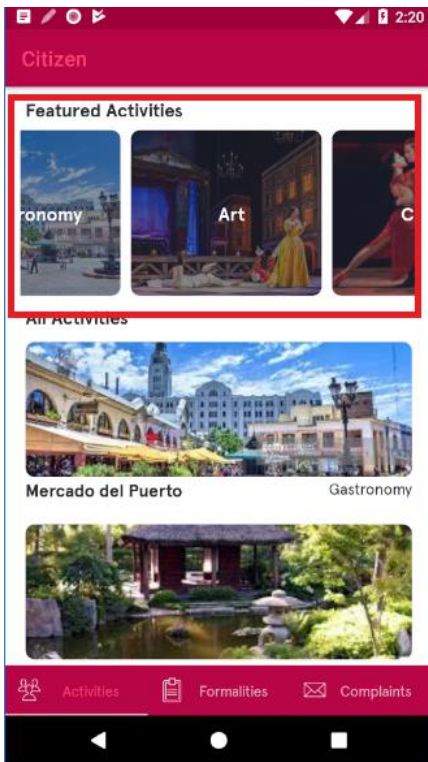


The table will be replaced with the stencil, keeping the attributes (they are not replaced with variables).

Test at runtime (Run on the CitizenMenu object).

## SMART GRID

The cultural activities panel shows, on top, a grid with the activities marked and highlighted (see CulturalActivityFeatured attribute of transaction CulturalActivity). Those activities should be viewed in a carousel formal, as shown in the image:



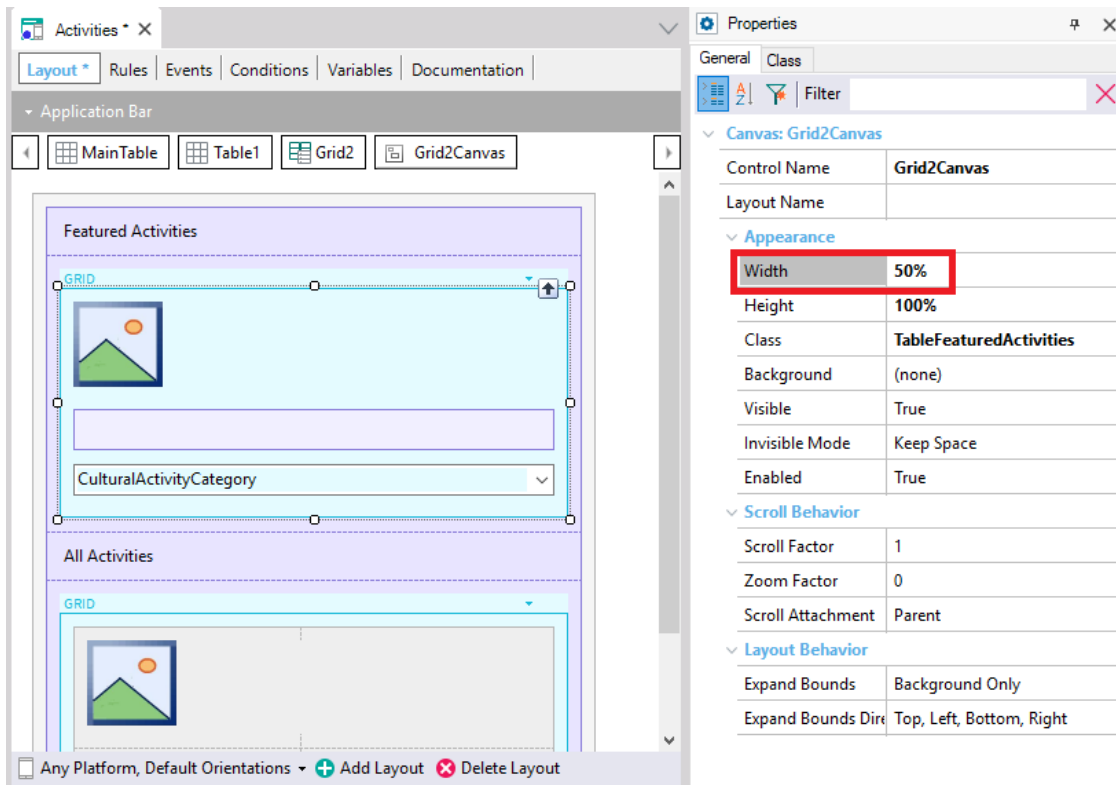
How is this implemented?

SOLUTION:

Properties	
General Class	
Control Name	Grid2
Collection	
Default Action	'ViewDetail'
Selection Type	Platform Default
Enable Multiple Selection	False
Pull To Refresh	False
Inverse Loading	False
Default Selected Item Label	(none)
Control Info	
Control Type	SD Smart Grid
Auto Grow	False
Scroll Direction	Horizontal
Snap To Grid	False
Items Layout Mode	Single

Change the Canvas Width as well, so that it occupies 50% instead of 100%:

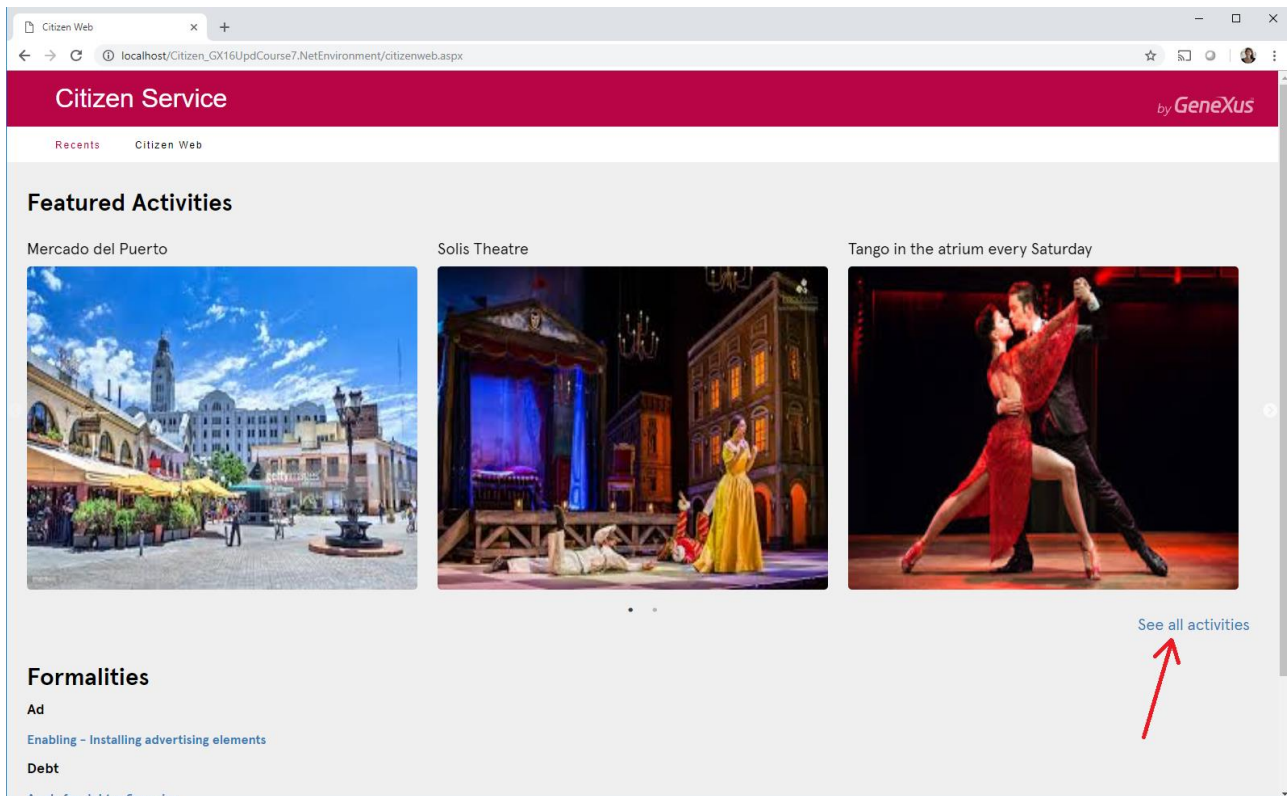




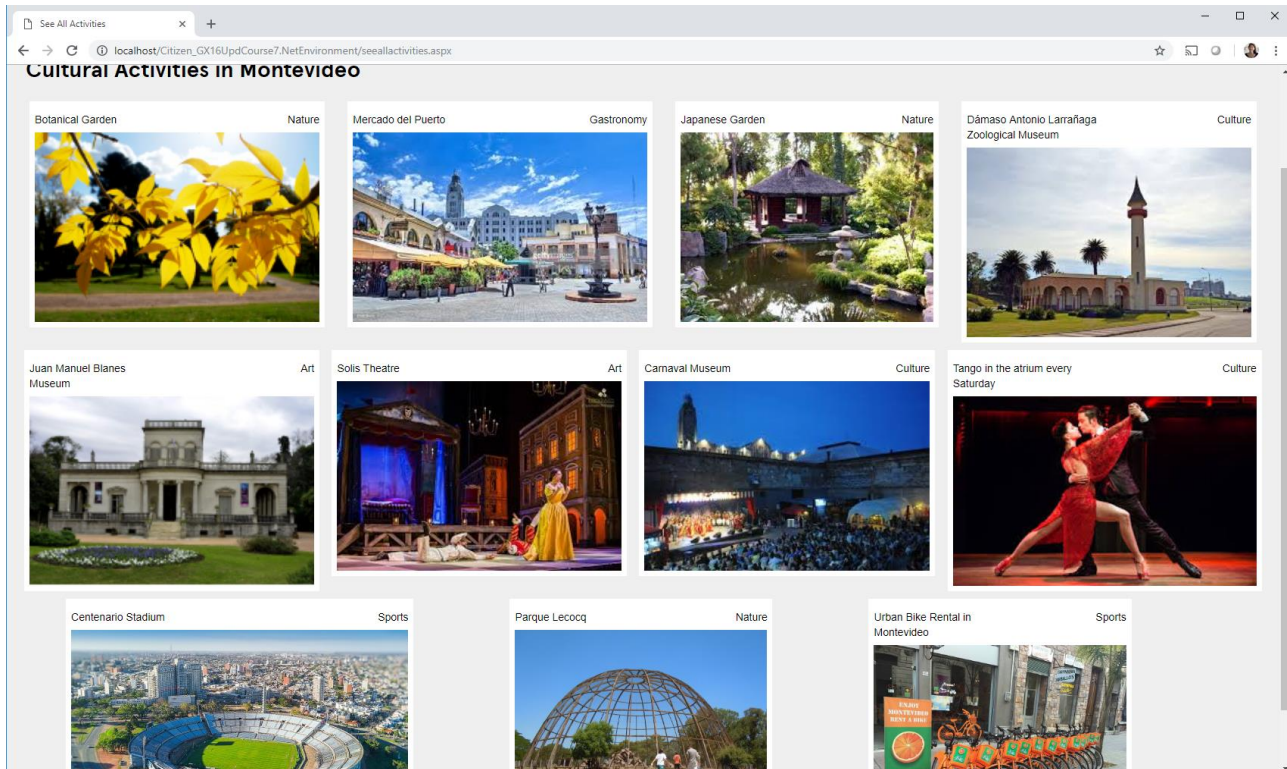
At runtime, note the difference with the horizontal grid.

## FLEX GRID

Notice that, in the main panel of the web frontend (called CitizenWeb), apart from viewing the highlighted cultural activities, there is an option called “See all Activities”:

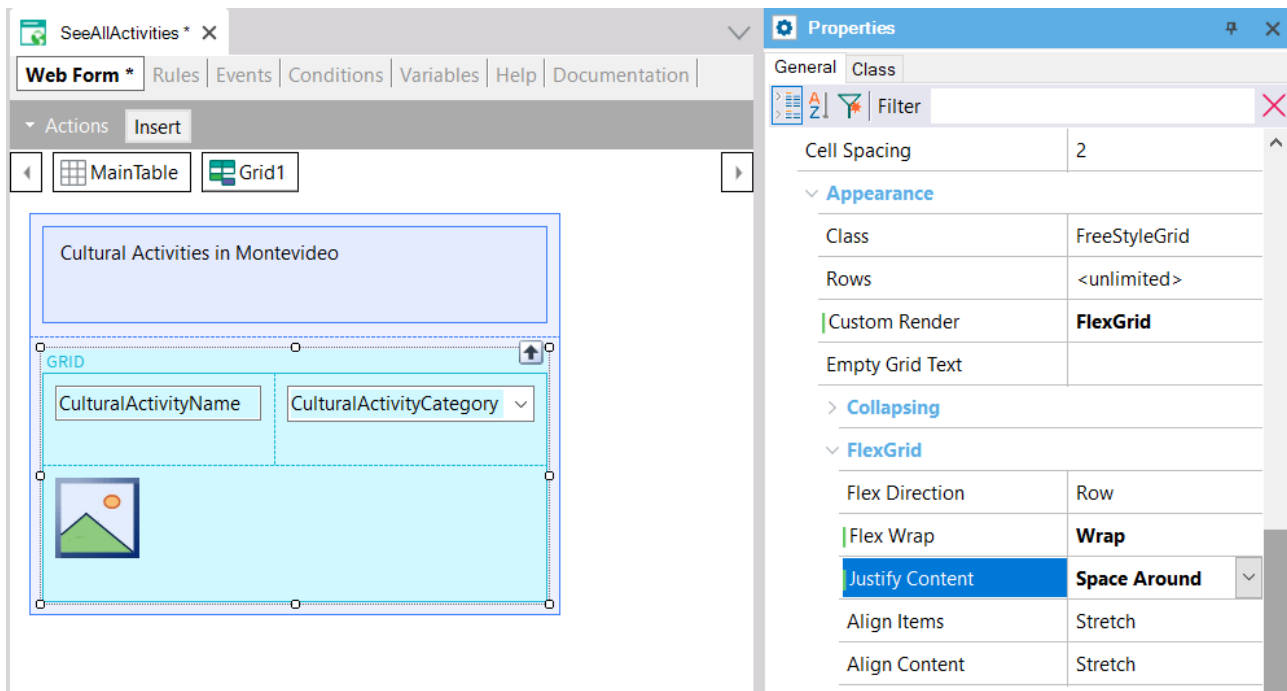


In this other screen we want the grid to allow showing them in a variable manner, according to size:



How is this implemented?

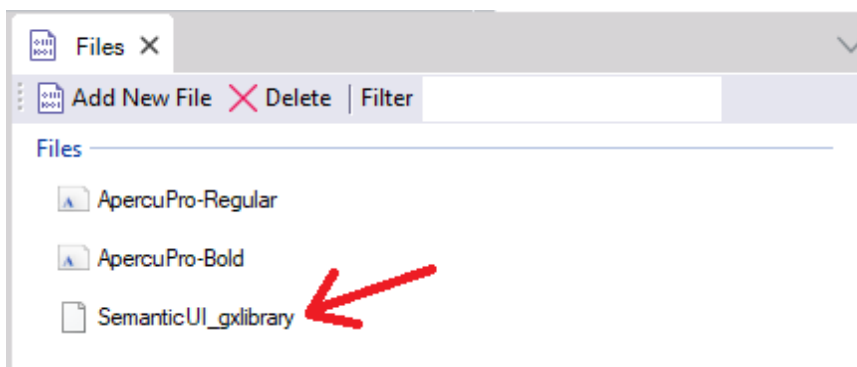
SOLUTION:



Note the differences with the Horizontal Grid you had before.

## BASE STYLE

When you edit the KB's files you will find the following:



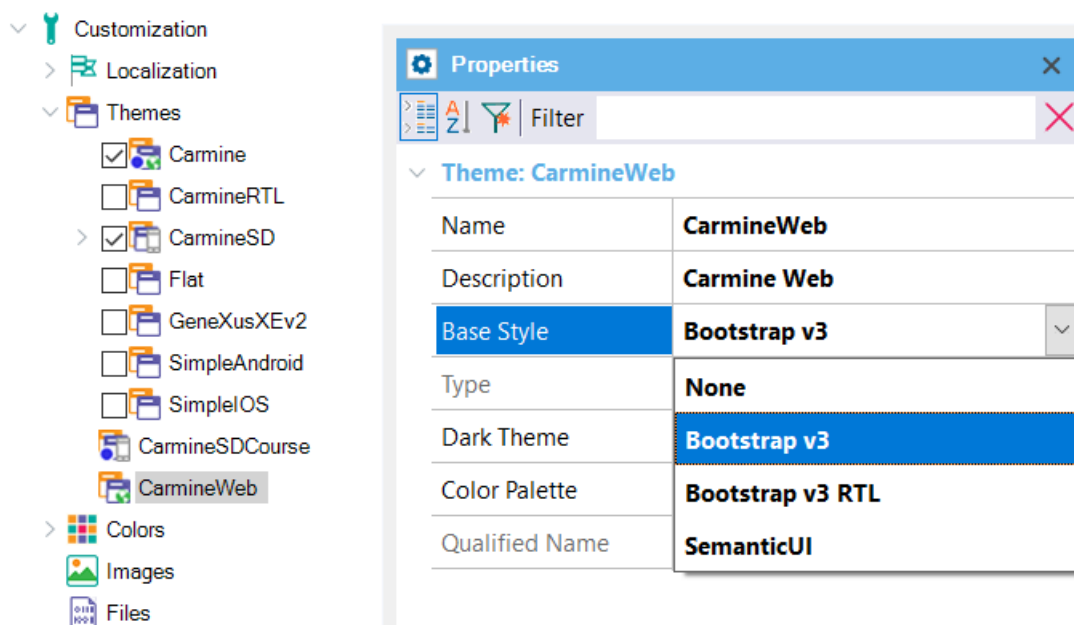
You will also find a new “Base Style” property in the objects:

- Theme
- User Control

This property enables you to assign a base style to these objects.

The property will take the values of the KB's files with extension "gxlbrary" (they are a zip containing all files css, js, assets, etc. that are provided by the designers or the Design System used).

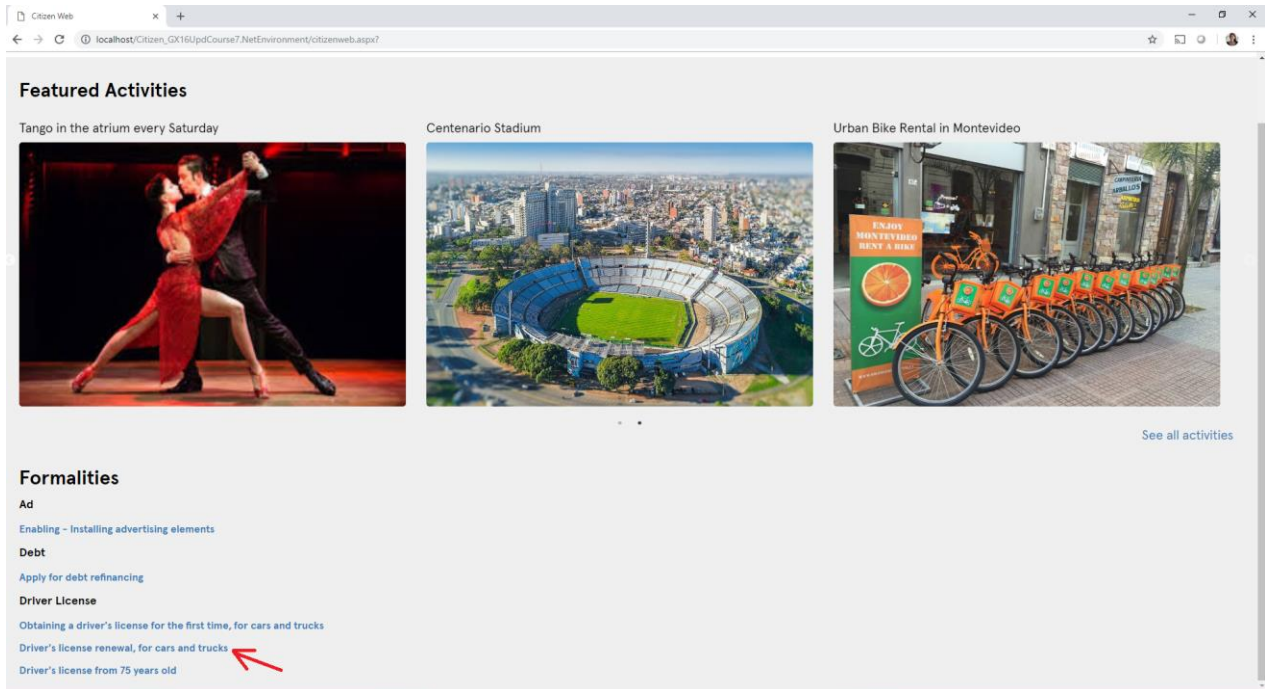
At the theme level you will find the following property:



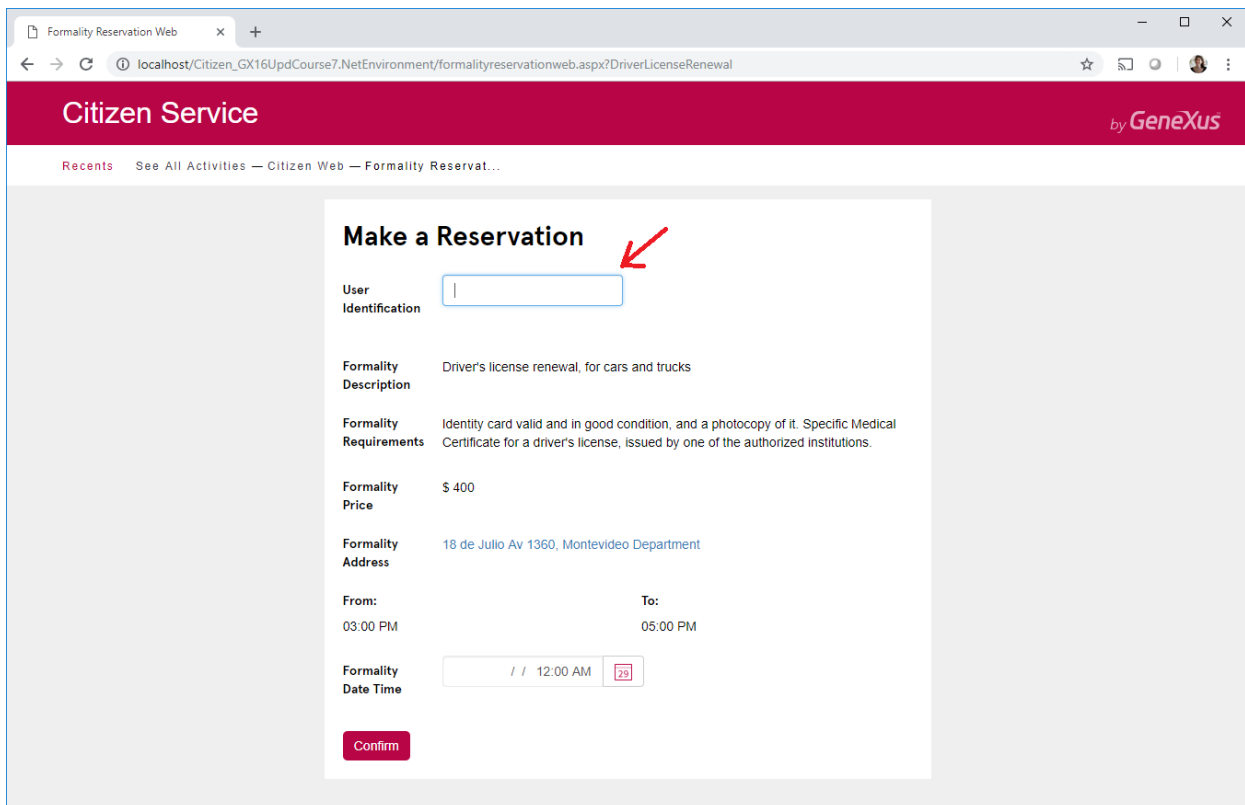
And at the object level we will see it as soon as we create a User control.

## USER CONTROL

If we now select a procedure from the main page of the web frontend:




As it was the case with the native mobile app, we will be required to provide a user identifier in order to book a reservation for that procedure:



Upon entering a valid user ID and exiting the field, we will want to view the user information as follows:

## Make a Reservation

User Identification

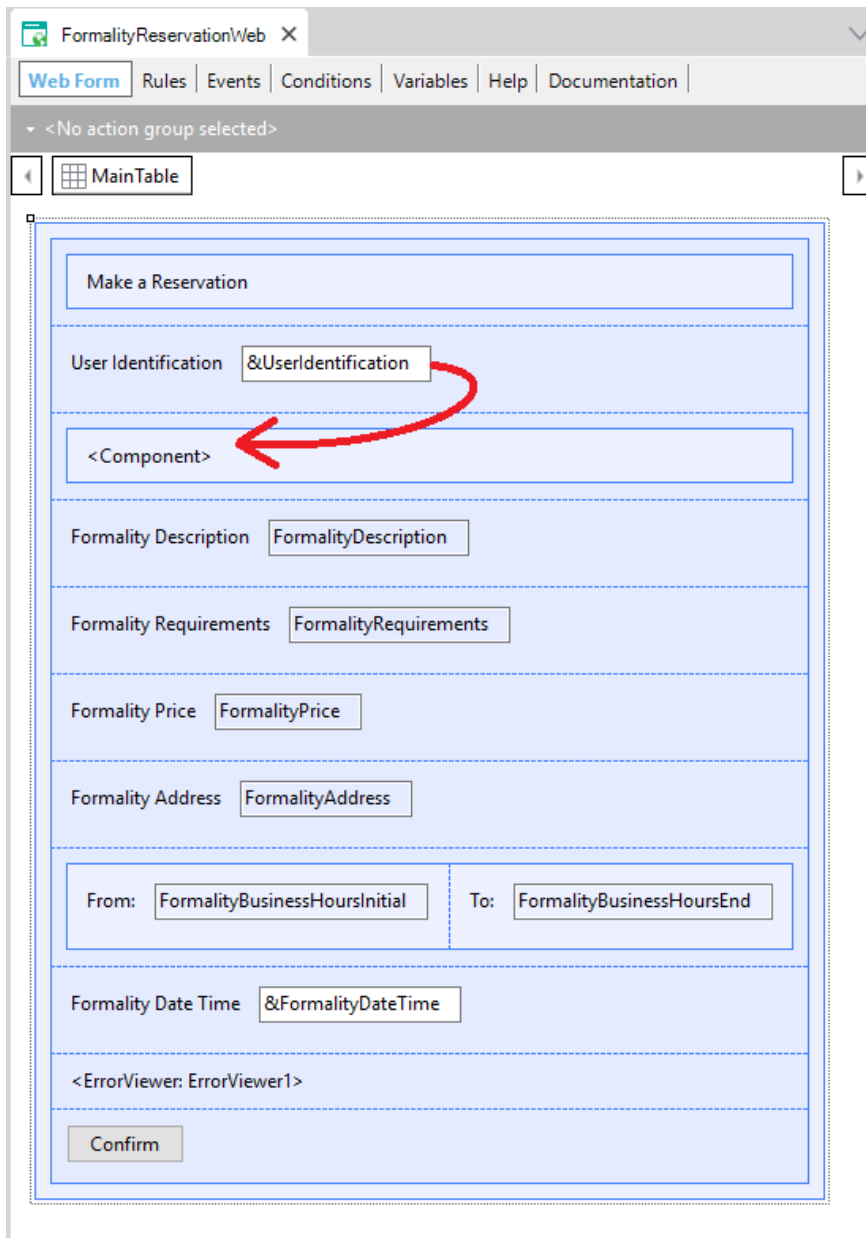


**John Peters**  
Born: 07/29/73  
Address: M. Boulevard 789  
E-Mail: john@test.com

<b>Formality Description</b>	Driver's license renewal, for cars and trucks
<b>Formality Requirements</b>	Identity card valid and in good condition, and a photocopy of it. Specific Medical Certificate for a driver's license, issued by one of the authorized institutions.
<b>Formality Price</b>	\$ 400
<b>Formality Address</b>	18 de Julio Av 1360, Montevideo Department

From: 03:00 PM To: 05:00 PM

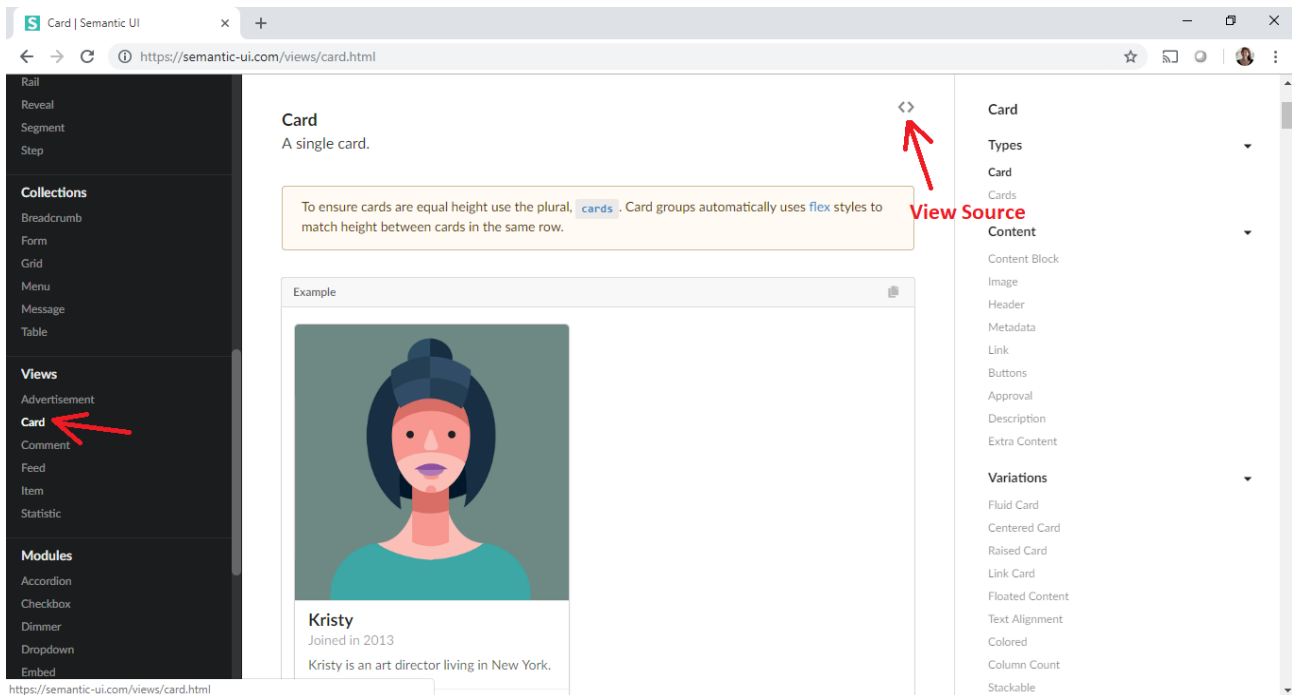
Note that the Card is loaded onto the component called WCUserInformation loaded on the web form:



How would you implement that component (WCUserInformation) that is empty in the KB?

SOLUTION:

It is a SemanticUI Card control. If you go to the SemanticUI site and search for Card, you will find different types of these controls. Select, for example, the simplest Card and view the html, in order to copy it:



It is written below so you can copy it later and save time during the session:

```
<div class="ui card">
  <div class="image">
    
  </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>
```

Create a User Control object in GeneXus and copy the previous html in your "Screen Template":



```

1 <div class="ui card">
2   <div class="image">
3     
4   </div>
5   <div class="content">
6     <a class="header">Kristy</a>
7     <div class="meta">
8       <span class="date">Joined in 2013</span>
9     </div>
10    <div class="description">
11      Kristy is an art director living in New York.
12    </div>
13  </div>
14  <div class="extra content">
15    <a>
16      <i class="user icon"></i>
17      22 Friends
18    </a>
19  </div>
20 </div>

```

Parameterize the properties according to what you wish to view. In our case, it will be as follows:

```

1 <div class="ui card card-extra-width">
2   <div class="image">
3     
4   </div>
5   <div class="content">
6     <a class="header">{{UserName}}</a>
7     <div class="meta">
8       <span class="date">{{UserBirthDay}}</span>
9     </div>
10    <div class="description">{{UserAddress}}</div>
11    <div class="description">{{UserEMail}}</div>
12  </div>
13 </div>

```

User Control: UserInformationCard	
Name	UserInformationCard
Description	User Information Card
Module/Folder	Root Module
Is Control Type	False
References	
Base Control Type	None
Base Style	SemanticUI
Qualified Name	UserInformationCard
Object Visibility	Public

Below is the previous code written down for you to copy:

```
<div class="ui card card-extra-width">
```

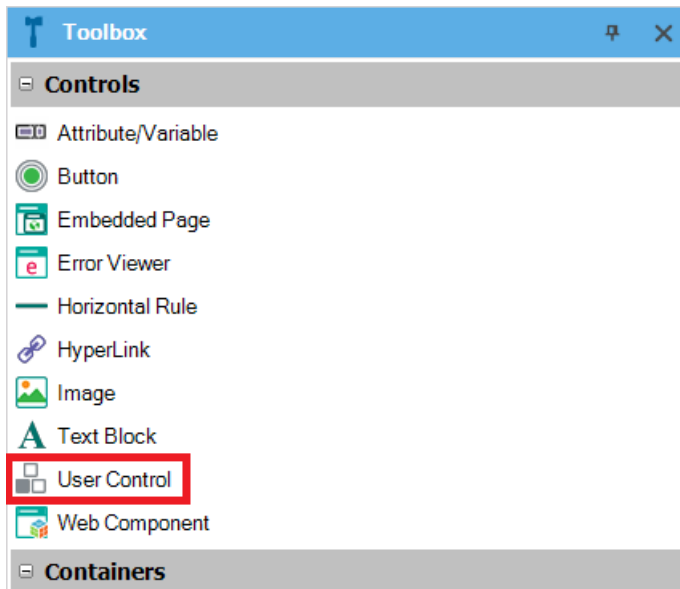
```

<div class="image">
  
</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>

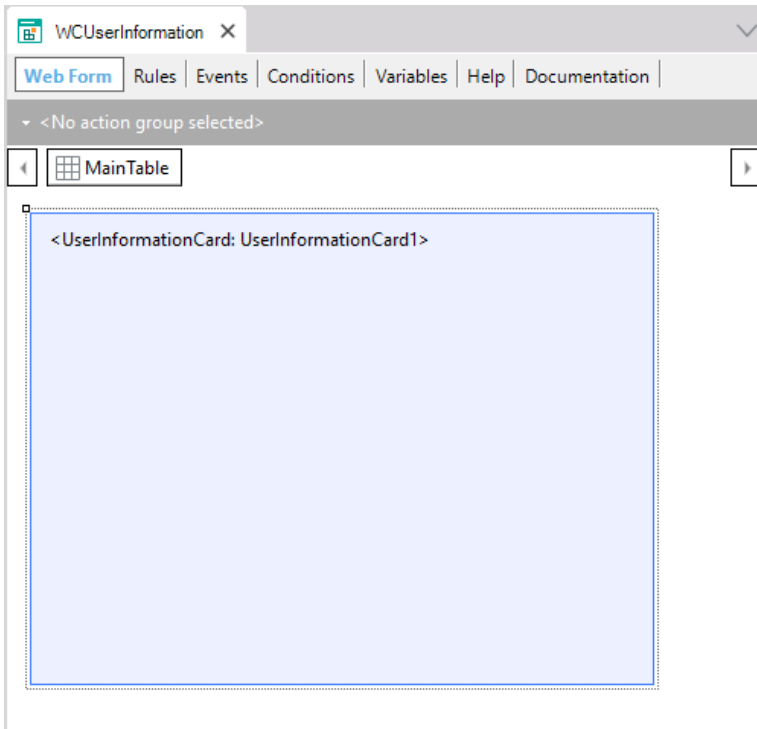
```

Verify that the value defined for the Base Style property of the User Control object is “SemanticUI”. Otherwise, select that value yourself.

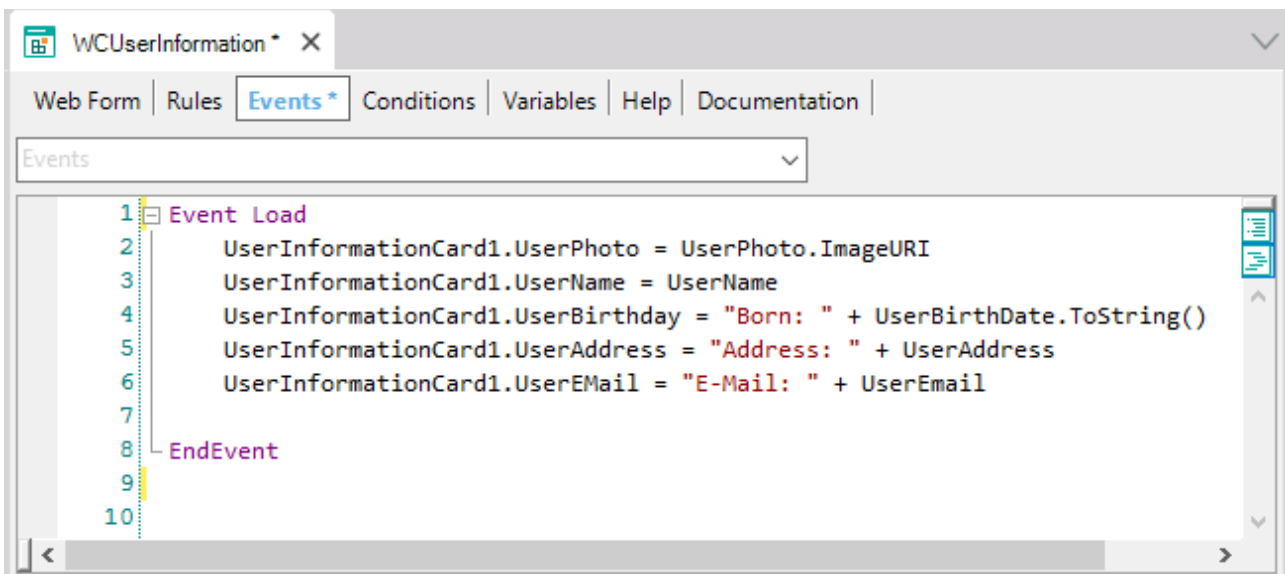
You will have to insert the user control you just created in the WCUserInformation component. In the Toolbox, you will find:



Drag it to the Web form and select the User Control you created before:



Then initialize the properties of the User Control based on the attributes:



**Note:** To verify that the Base Library (in this case SemanticUI) was correctly loaded, check the web directory of the KB (Tools → Explore Target Environment Directory) and see if the “SemanticUI” folder exists. If not, we suggest that you do a Rebuild of the CitizenWeb object.


Now execute.

## ADDING AN ACTION TO THE USER CONTROL

There is the option to allow that the user information be edited from the Card.

## Make a Reservation

User Identification



**John Peters**  
Born: 07/29/73  
Address: M. Boulevard 789  
E-Mail: john@test.com

[Edit Data](#) ←

**Formality Description** Driver's license renewal, for cars and trucks

**Formality Requirements** Identity card valid and in good condition, and a photocopy of it. Specific Medical Certificate for a driver's license, issued by one of the authorized institutions.

**Formality Price** \$ 400

**Formality Address** [18 de Julio Av 1360, Montevideo Department](#)

**From:** 03:00 PM **To:** 05:00 PM

It was not exactly like this in the source html of the SemanticUI control. You may customize its User Control object by adding:

```

1 <div class="ui card card-extra-width">
2   <div class="image">
3      </div>
4     <div class="content">
5       <a class="header">{{UserName}}</a>
6       <div class="meta"> <span class="date">{{UserBirthday}}</span>
7       <div class="description"> {{UserAddress}} </div>
8       <div class="description"> {{UserEMail}} </div>
9     </div>
10    <div class="extra content" {{OnClick}}>
11      <a <i class="user icon"></i> {{EditData}} </a>
12    </div>
13 </div>

```

And then, in the Web component:

```

1 Event Load
2   UserInformationCard1.UserPhoto = UserPhoto.ImageURI
3   UserInformationCard1.UserName = "Born: " + UserBirthDate.ToString()
4   UserInformationCard1.UserAddress = "Address: " + UserAddress
5   UserInformationCard1.UserEMail = "E-Mail: " + UserEMail
6   UserInformationCard1.EditData = "Edit Data"
7 Endevent
8

```

Test it at runtime.

And now the event must be programmed. To do that:

```

1  Event Load
2      UserInformationCard1.UserPhoto = UserPhoto.ImageURI
3      UserInformationCard1.UserName = "Born: " + UserBirthDate.ToString()
4      UserInformationCard1.UserAddress = "Address: " + UserAddress
5      UserInformationCard1.UserEMail = "E-Mail: " + UserEMail
6      UserInformationCard1.EditData = "Edit Data"
7  Endevent
8
9  Event UserInformationCard1.OnClick
10     User( TrnMode.Update, UserIdentification )
11  Endevent
    
```

Test again.

## KB SOLUTION

You may download the solution KB for this practice session from GeneXus Server in order to compare results. It is the version of the KB called CitizenSolutionPartOne.