

# Curso prático de atualização para

# GeneXus™ 16

## PARTE 1

Abril 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.*

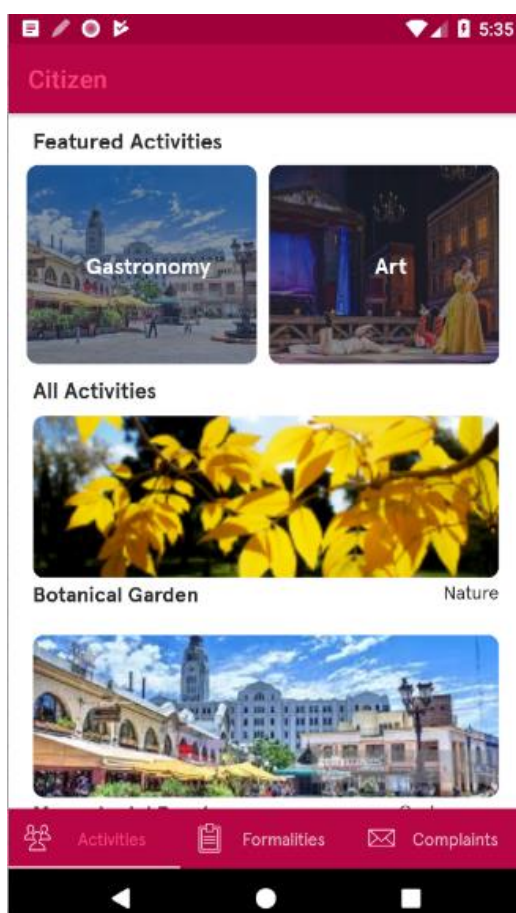
## CONTEÚDO

|   |           |
|---|-----------|
| CONTEÚDO.....                                   | 2         |
| OBJETIVO.....                                   | 3         |
| VAMOS COMEÇAR.....                              | 4         |
| FAMILIARIZAÇÃO COM A KB.....                    | 4         |
| STENCILS .....                                  | 6         |
| <b>Justificativa do uso de stencils .....</b>   | <b>6</b>  |
| Solução:.....                                   | 10        |
| <b>Criar um stencil e usá-lo .....</b>          | <b>11</b> |
| Solução Possível:.....                          | 11        |
| SMART GRID .....                                | 14        |
| Solução:.....                                   | 15        |
| FLEX GRID .....                                 | 16        |
| Solução:.....                                   | 18        |
| BASE STYLE.....                                 | 18        |
| USER CONTROL.....                               | 19        |
| Solução:.....                                   | 22        |
| <b>Adicionar uma ação ao User control .....</b> | <b>27</b> |
| KB SOLUÇÃO .....                                | 29        |

## OBJETIVO

Para alcançar o objetivo de desenvolver aplicações multi-experiência e omnichannel, melhorando a integração com a equipe de projeto, neste prático trataremos os temas Design Systems e experiência do usuário relacionado às formas de fazer com que as informações repetitivas fluam de forma mais contínua e adaptável na UI - tudo o que faz para os novos tipos de grid-.

A aplicação em que vamos trabalhar é uma simplificação de uma app para o município de uma cidade, que oferece um frontend Web e um SD para que os cidadãos, através de seu identificador de usuário, possam fazer reclamações (por exemplo, árvores caídas, semáforos que não funcionam, carros mal estacionados, etc.), possam realizar procedimentos (por exemplo, para obter uma carteira de motorista, refinanciar uma dívida com o município, instalar elementos publicitários, etc.), reservando um turno para ser atendido pela equipe do município. No frontend também são mostradas as diversas atividades culturais oferecidas pela cidade:



E há também um backoffice web para que certos funcionários do município lidem com os dados e vejam estatísticas.

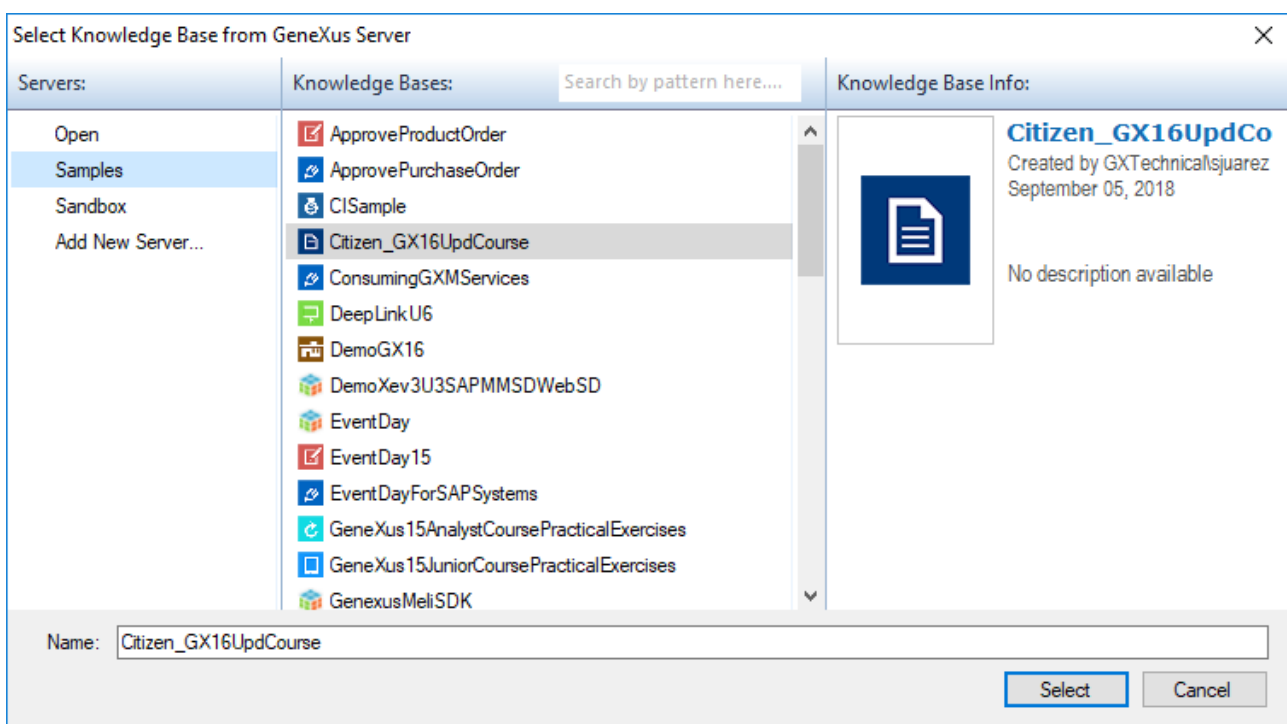
## VAMOS COMEÇAR

Utilizaremos a versão **GeneXus 16** e o Emulador do **Android SDK**. A carta deste prático foi atualizada de acordo com o upgrade 9 do GeneXus 16. Se você realizá-lo com uma versão posterior, considere que pode haver alterações entre o que é mostrado aqui e o que você vê.

A menos que os instrutores lhes digam que a KB já está criada na sua máquina, faça em GeneXus: File/New Knowledge Base from GeneXus Server, escolhendo como Server KB URL:

<http://samples.genexusserver.com/v16/>.

Em Select Server KB, escolha a de nome “Citizen\_GX16UpdCourse”:

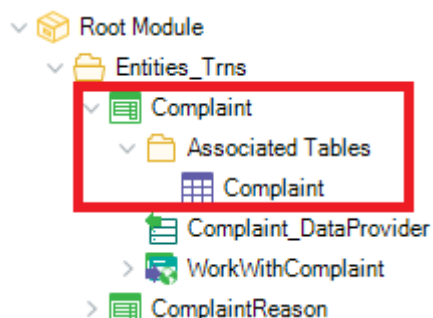


E selecione a versão trunk.

Execute um Rebuild All e Create. Vamos prototipar localmente e não na nuvem. Certifique-se de que já estejam configurados nome do server, e de base de dados, ou configure-os com a instância do SQLServer da máquina e nome do DB que você desejar. Consulte o instrutor.

## FAMILIARIZAÇÃO COM A KB

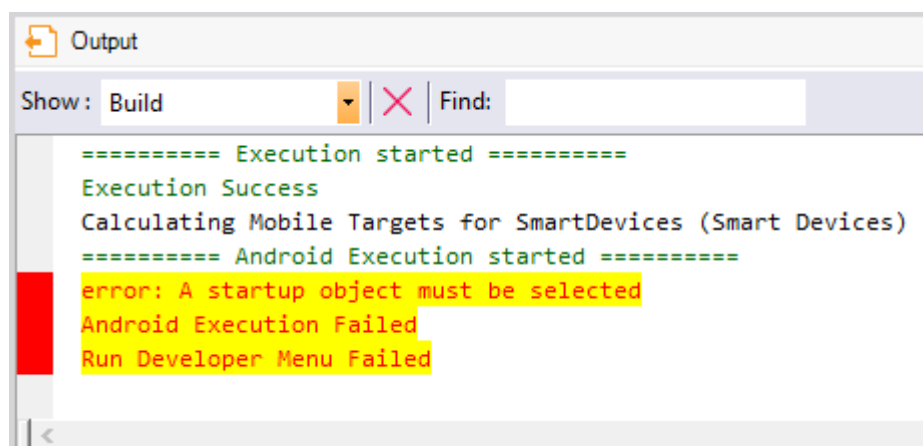
Sob a pasta Entities\_Trns, estão as entidades que modelam a realidade da app. Observe que agora as tabelas são exibidas sob cada transação no KB Explorer.



Da mesma forma observe que foram definidos os Data Providers para popular as tabelas associadas às transações (definindo a propriedade Data Provider da transação). Serão invocados automaticamente na primeira execução.

Então, há duas pastas FrontendWeb e FrontendSD que implementam ambos os frontends, com os mains da app que os cidadãos usarão.

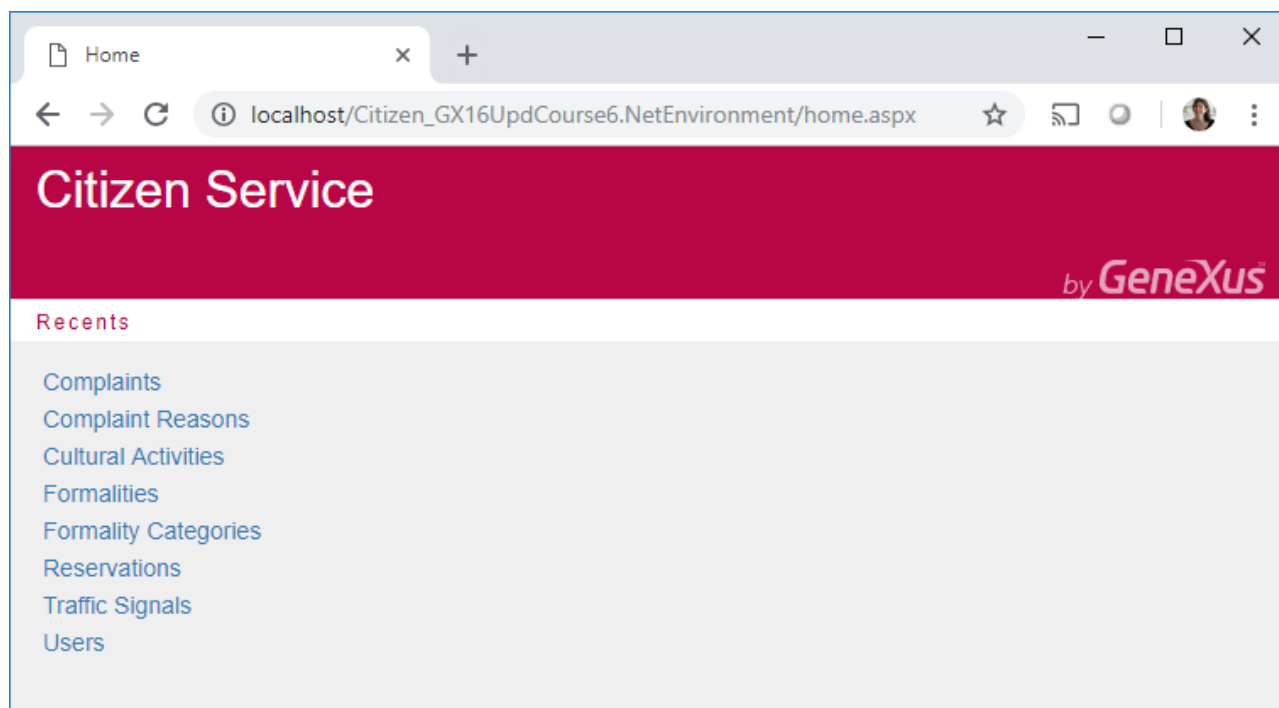
O Web panel Home dentro de GeneXus/Web contém o main do backoffice (é o default), que será utilizado pelos funcionários do município para a administração da informação do site. Será o que se exibirá como resultado do F5, que por sua vez exibirá o seguinte erro:



Isto ocorre porque não configuramos o Startup object pois estaremos alternando entre backoffice e frontend Web e SD. Você pode especificar o Home do backoffice como Startup Object ou não. Como desejo, já que de toda forma, deverá fazer Run sobre o main que deseja executar em cada oportunidade (CitizenMenu para frontend SD, CitizenWeb para frontend Web).

A pasta Queries possui algumas consultas que serão utilizadas em outro prático.

Se você executar o web panel Home:



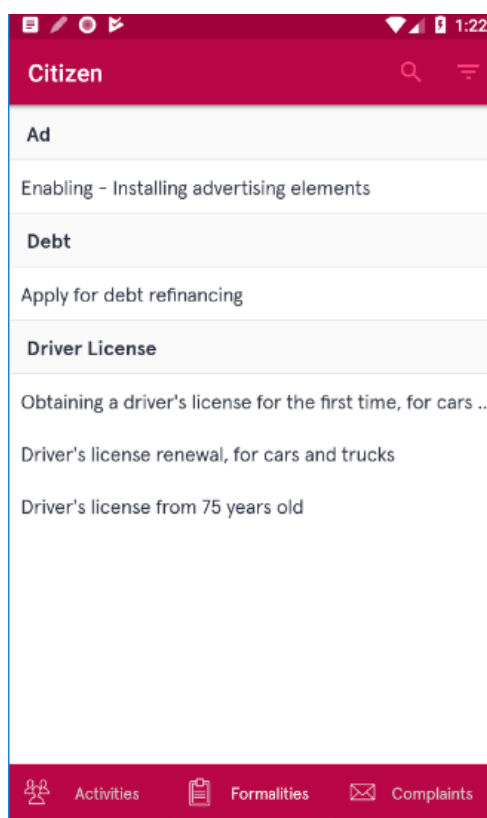
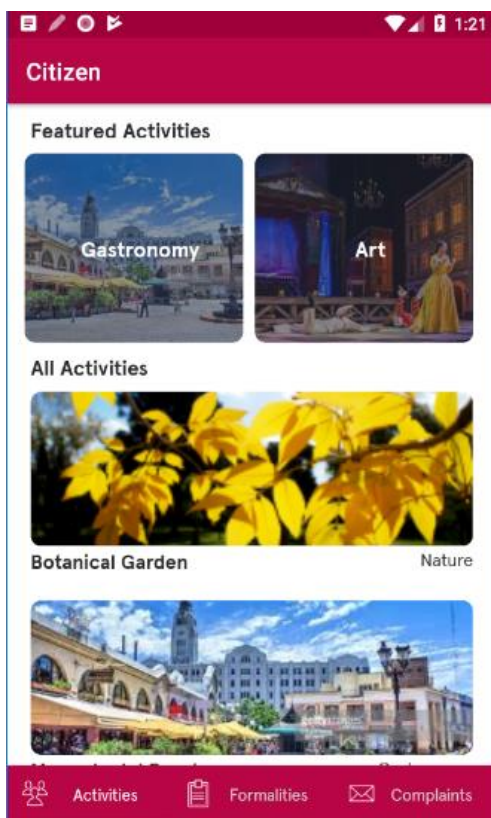
Poderá ver os work withs das entidades, com os dados carregados. Deixe-o aberto.

## STENCILS

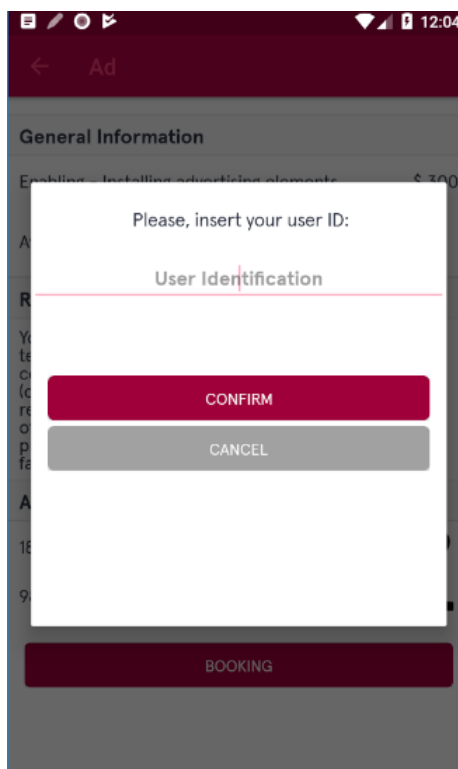
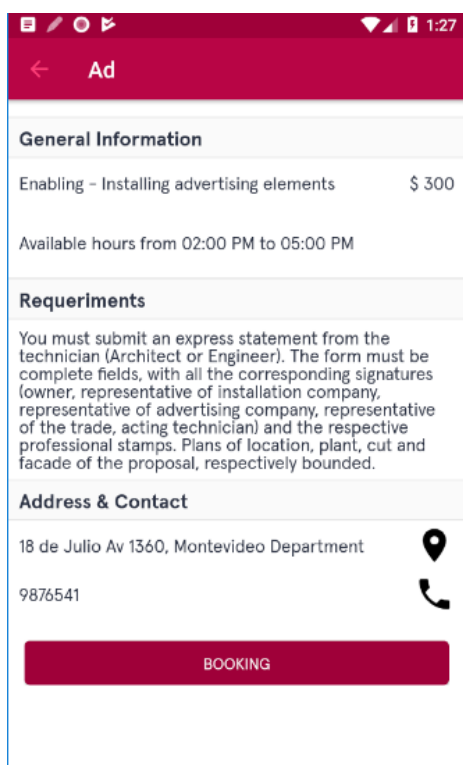
Execute o main da app SD (objeto CitizenMenu dentro da pasta FrontendSD).

## JUSTIFICATIVA DO USO DE STENCILS

Verá uma primeira tela que mostra as atividades culturais (Activities). E na segunda aba do menu abaixo poderá ver os procedimentos que o usuário pode realizar (instalar publicidade, refinar uma dívida, obter carteira de motorista):



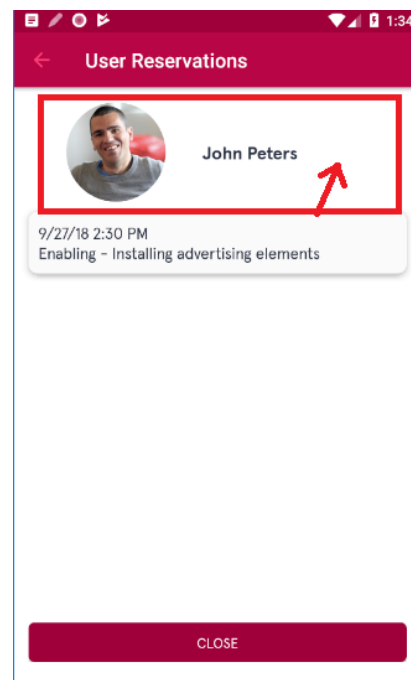
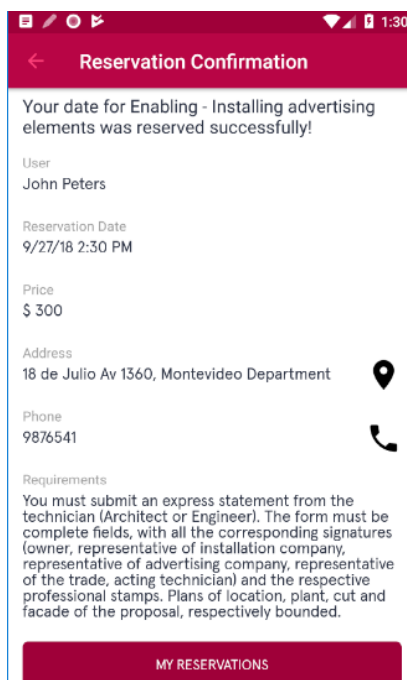
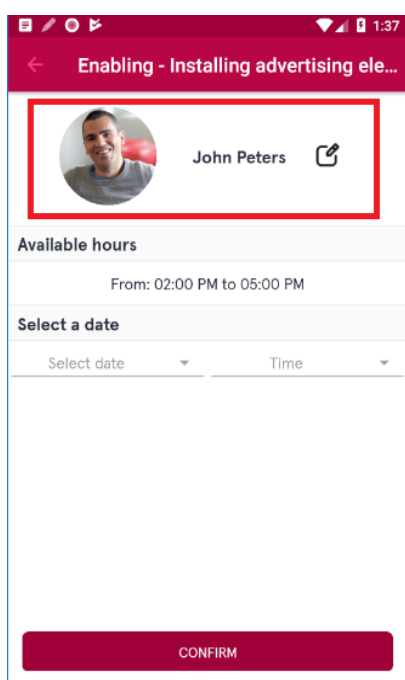
Se o usuário escolher um procedimento, por exemplo, permitir publicidade, as informações gerais do procedimento são mostradas e permite ao usuário reservar um turno para ser atendido para realizar esse procedimento. E ao pressionar o "BOOKING" é solicitado o ID de usuário:



Recomendamos abrir o Backoffice para ver os usuários registrados e seus IDs:

| Identification | Name              | Photo | Age | UPDATE | DELETE |
|----------------|-------------------|-------|-----|--------|--------|
| 20576          | Ann Bert          |       | 43  | UPDATE | DELETE |
| 11182          | Brian Goals       |       | 77  | UPDATE | DELETE |
| 89680          | Cecilia Fernández |       | 43  | UPDATE | DELETE |
| 15587          | Cecilia Lemos     |       | 16  | UPDATE | DELETE |
| 81180          | Eleonor Johnson   |       | 28  | UPDATE | DELETE |
| 49128          | Jessica Loyarte   |       | 30  | UPDATE | DELETE |
| 12345          | John Peters       |       | 45  | UPDATE | DELETE |
| 11245          | Luciano Silveira  |       | 42  | UPDATE | DELETE |

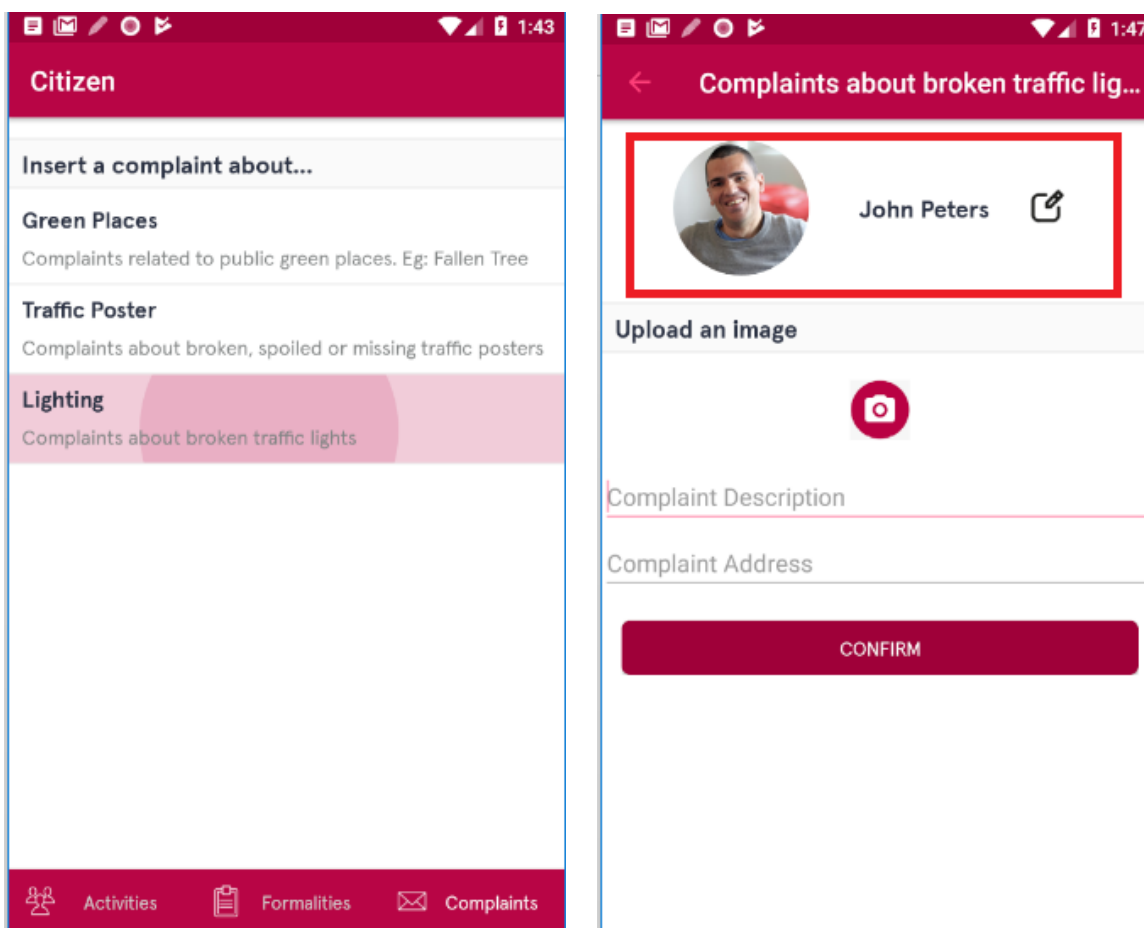
Ao inserir um e confirmar (por exemplo, ID 12345), será exibida uma tela mostrando a foto e o nome do usuário inserido, e permite selecionar o dia e a hora, dentro dos horários autorizados do procedimento, para reservar um turno para ser atendido. Ao fazer isso, se tudo correr bem, será apresentada outra tela que mostra todas as informações do procedimento, e é oferecida a opção de ver todas as reservas que tem até o momento:





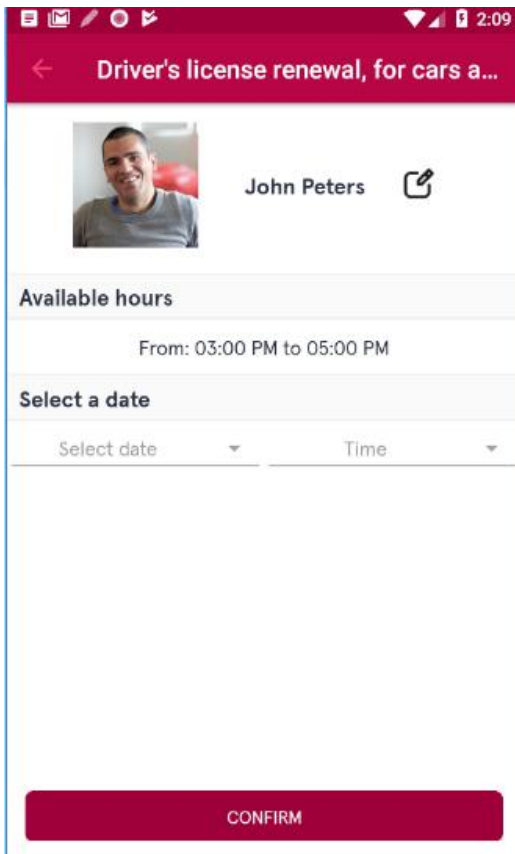
Nesta última tela também se visualiza foto e nome de usuário, com exatamente o mesmo design de antes, embora onde tenha sido ocultada a possibilidade de alterar o usuário.

Por outro lado, se formos para a opção Complaints do menu, selecionando o tipo de reclamação, o usuário poderá inserir uma desse tipo no sistema:



Observe que nesta tela, UserComplaint, também são exibidas as informações do usuário da mesma maneira que na do procedimento.

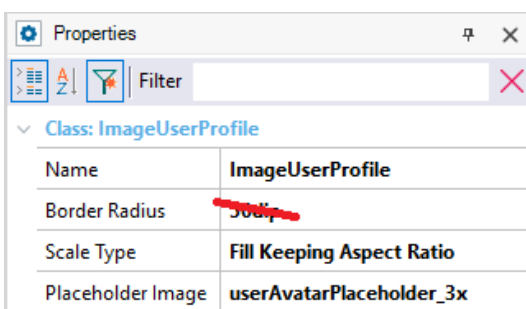
Se quiser modificar a maneira que a foto é exibida (nas três telas em que aparece) para que não saia mais arredondada, tal como se vê aqui:



Como você implementa isso?

SOLUÇÃO:

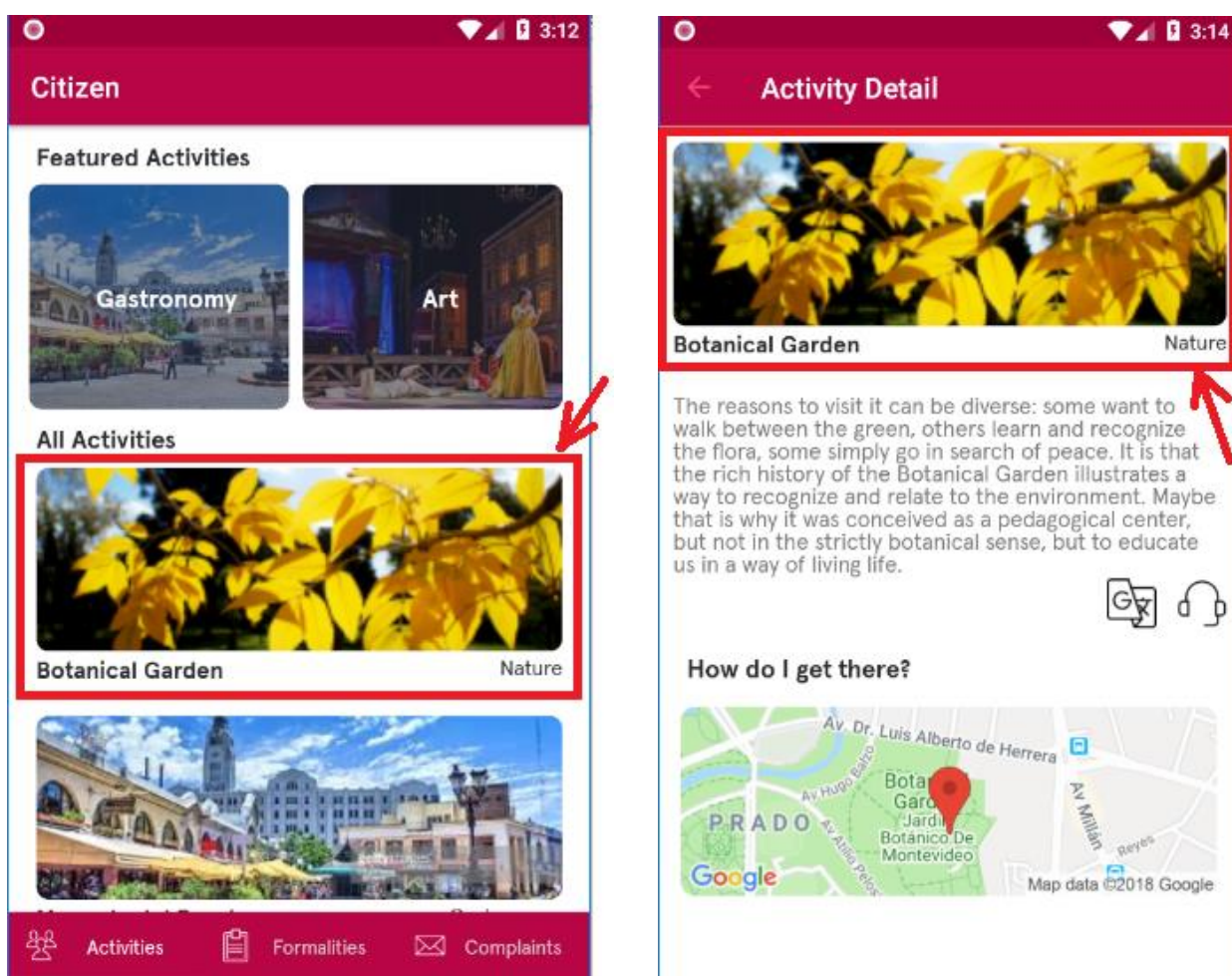
Abra o stencil UserProfile e veja que a foto tem associada a classe ImageUserProfile do theme CarmineSDCourse. Altere a classe Border Radius, deixando-a vazia.



Execute o CitizenMenu e observe a alteração nas três telas.

## CRIAR UM STENCIL E USÁ-LO

Agora queremos que a tela que mostra as atividades culturais (Activities) mostre acima as atividades destacadas, e abaixo todas as atividades. Quando uma atividade é selecionada em qualquer um dos dois grids, é exibida uma tela com seus detalhes:

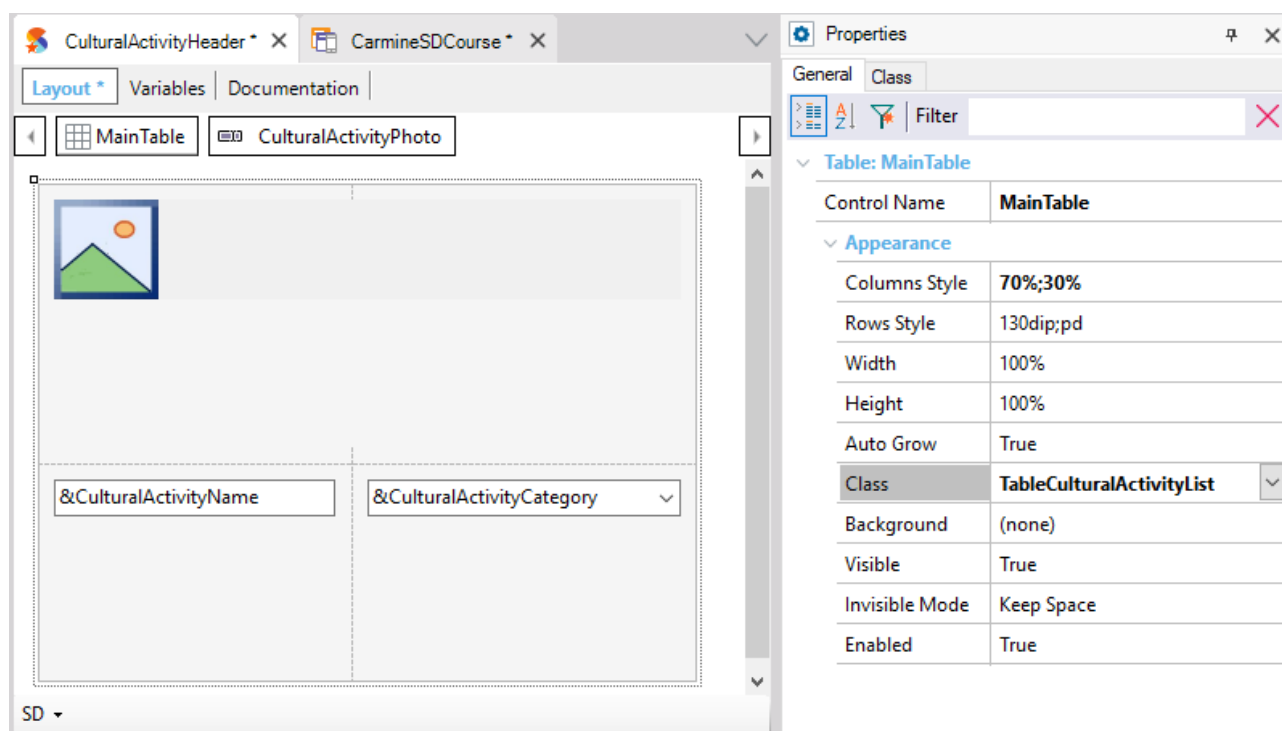


Observe que a imagem da atividade, seu nome e categoria são exibidos de forma idêntica no grid que exibe All Activities no primeiro painel e no segundo painel. Para evitar a repetição do design, é que usamos stencils.

Implemente um stencil para mostrar esta informação e use-o nos dois painéis: Activities e ActivityDetail.

SOLUÇÃO POSSÍVEL:

Crie o stencil de nome CulturalActivityHeader (você pode fazer isso manualmente ou pode posicionar-se sobre a tabela que já contém a foto e os dois atributos no painel Activities e, com o botão direito, escolher Wrap as new stencil):



Se você fez isso manualmente: para a variável com a imagem, configure as propriedades:

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

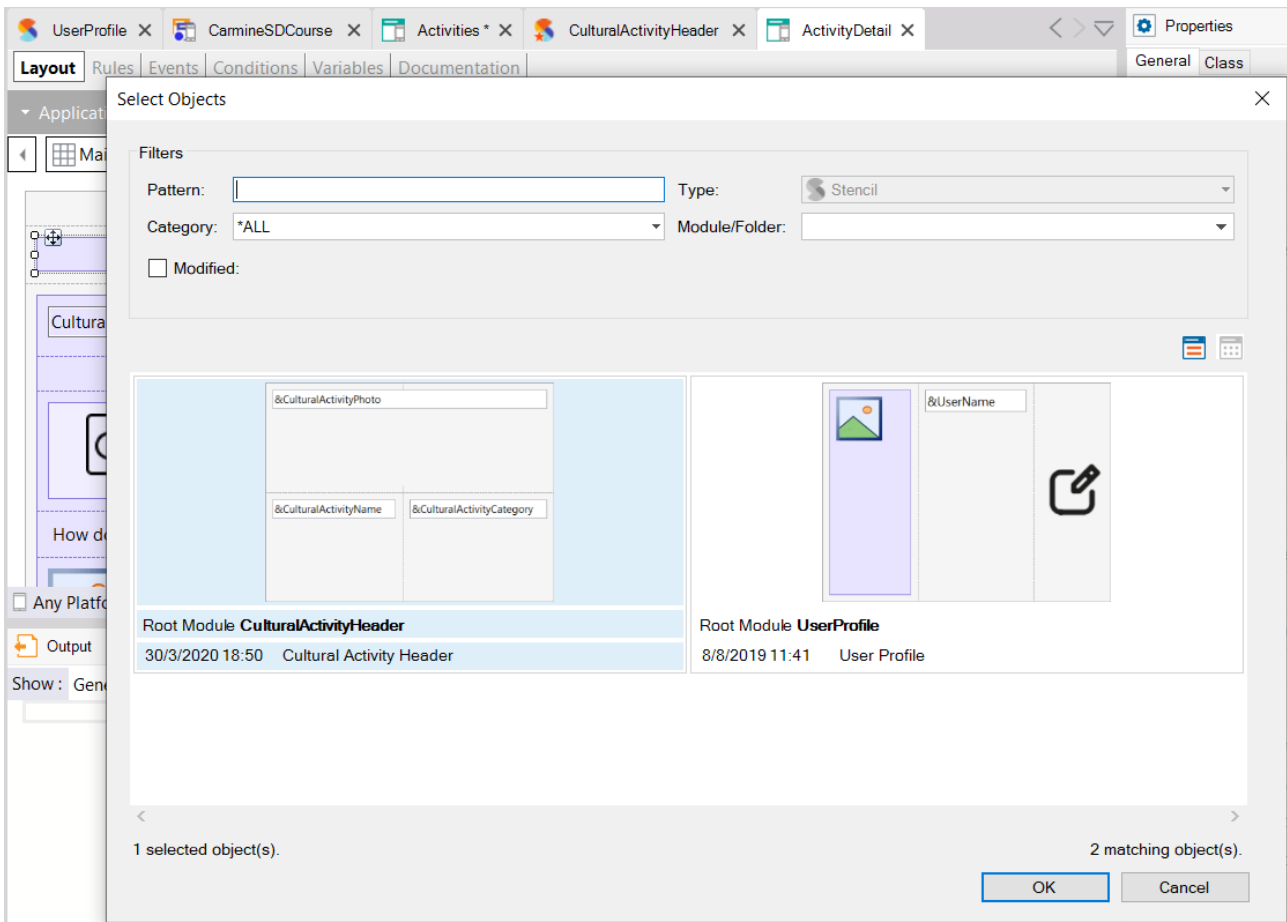
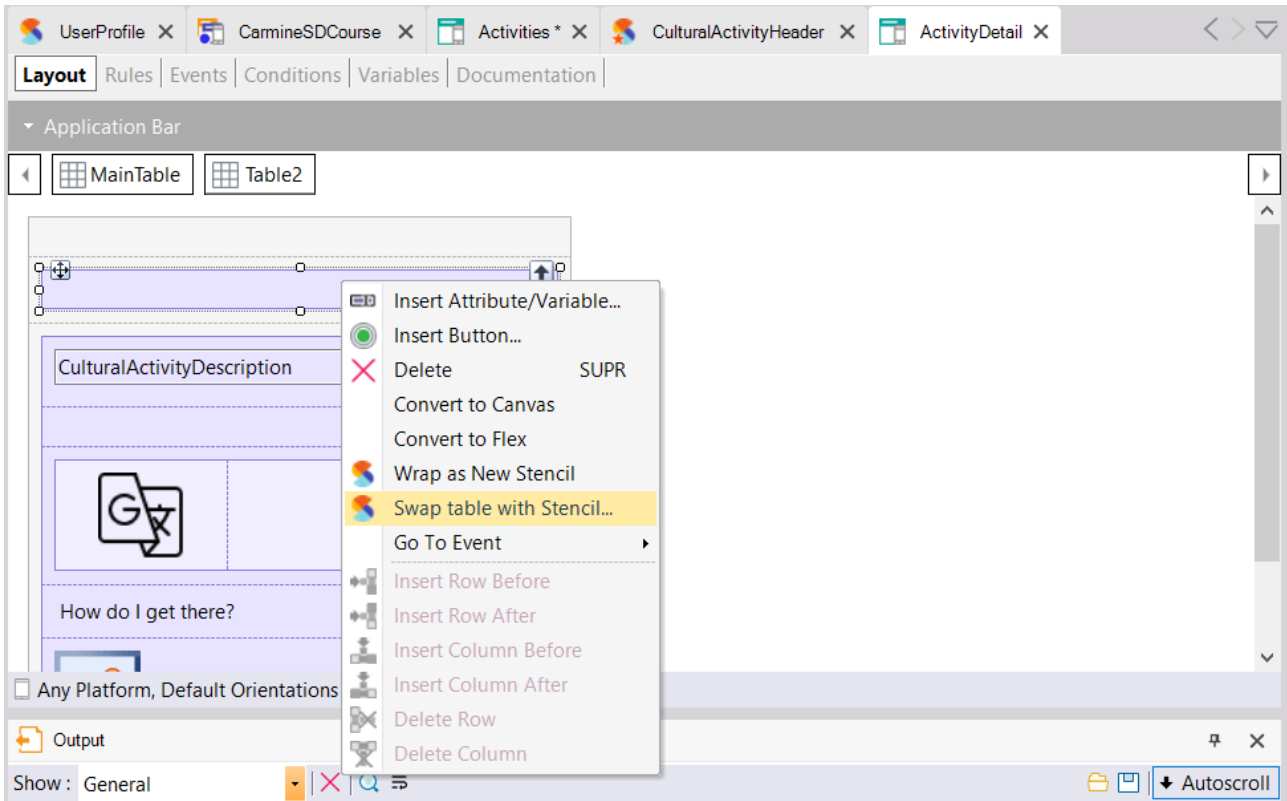
Para a variável que mostra o nome à esquerda, configure as propriedades:

- Label Position: None
- Class: AttributeActivityNameList

Para a variável que mostra a categoria à direita, configure as propriedades:

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

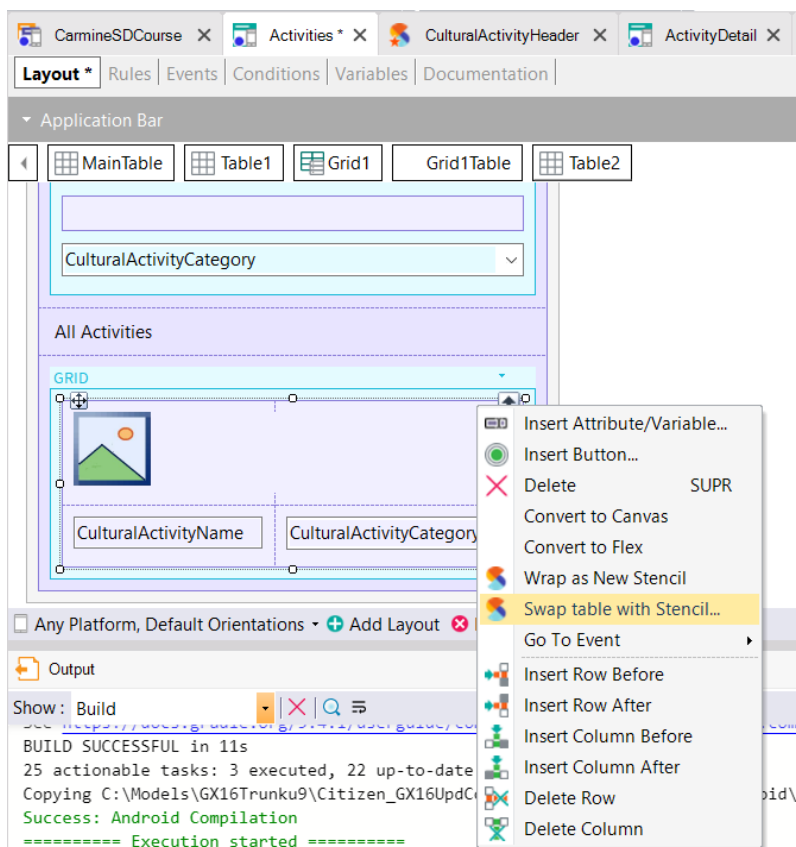
No painel ActivityDetail, na tabela onde inseriria os atributos CulturalActivityPhoto, CulturalActivityName e CulturalActivityCategory, que está vazia, procure o Stencil recém-criado e insira-o.



Nota:

- Deverá substituir as variáveis pelos atributos CulturalActivityPhoto, CulturalActivityName e CulturalActivityCategory.

Se criou o stencil manualmente, então no painel Activities, posicione-se na tabela do grid que mostra todas as atividades e, pressionando o botão direito, observe que ao fazer “Swap table with stencil”:

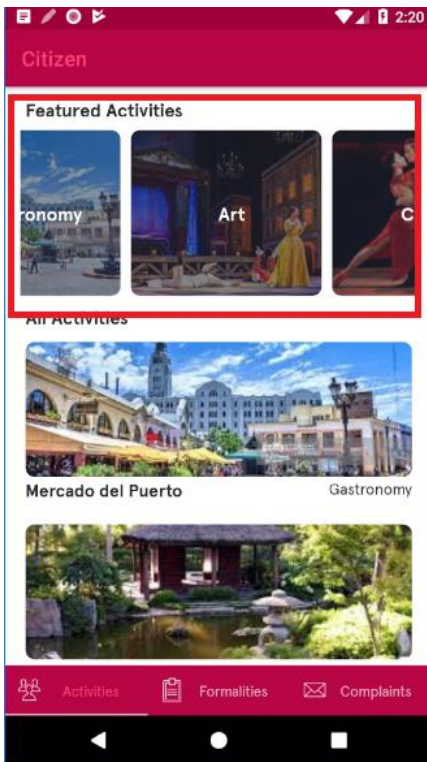


e o escolher, já substituirá a tabela pelo stencil, respeitando os atributos (não os alterará por variáveis).

Teste em execução (Run sobre o objeto CitizenMenu).

## SMART GRID

O painel de atividades culturais mostra acima um grid com as atividades marcadas como destacadas (ver atributo CulturalActivityFeatured da transação CulturalActivity). Se deseja que essas atividades sejam vistas em um formato de carrossel, como mostrado na imagem:

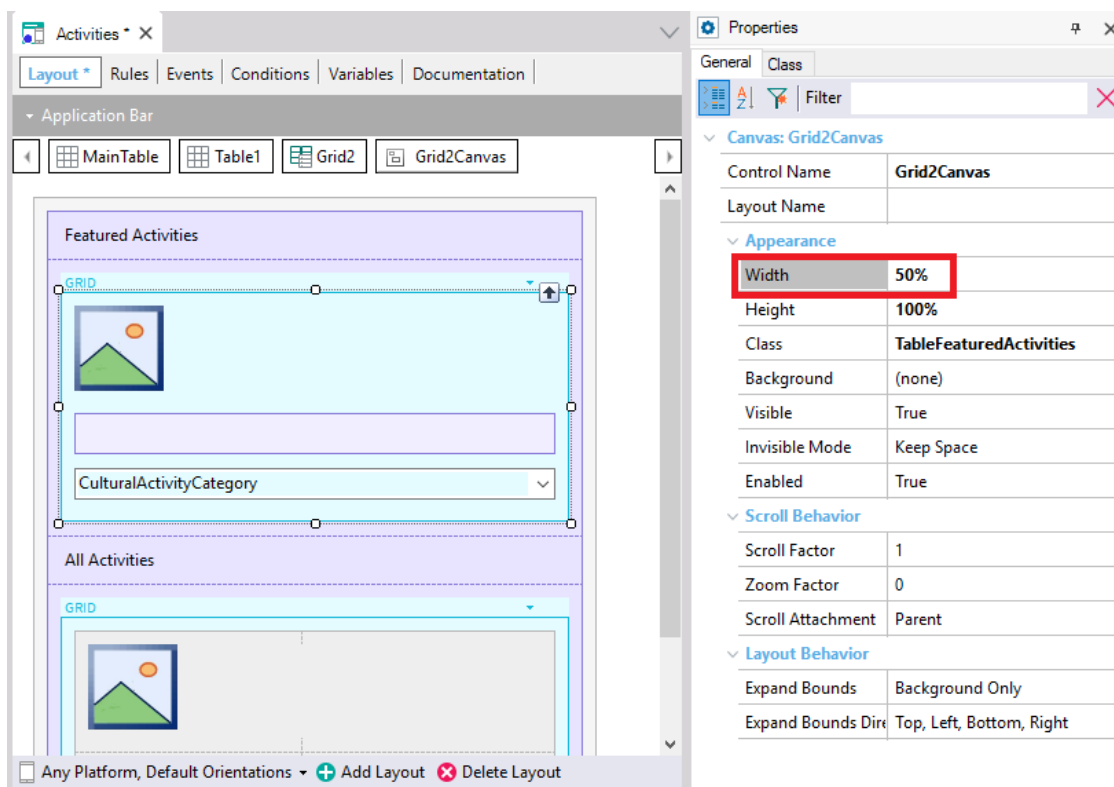


Como você implementa isso?

SOLUÇÃO:

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

Altere, além disso, a Width Canvas para que ocupe 50% e não 100:

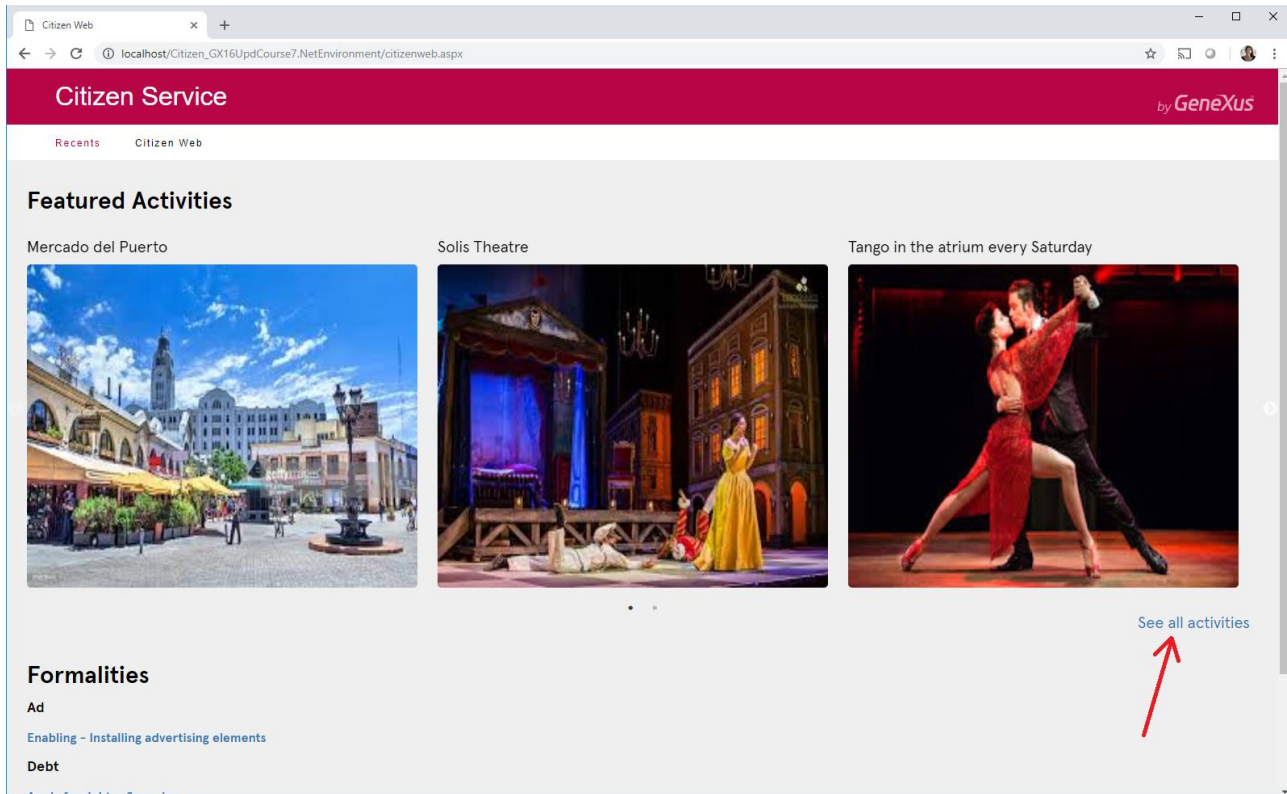


Observe em execução a diferença com o horizontal grid.

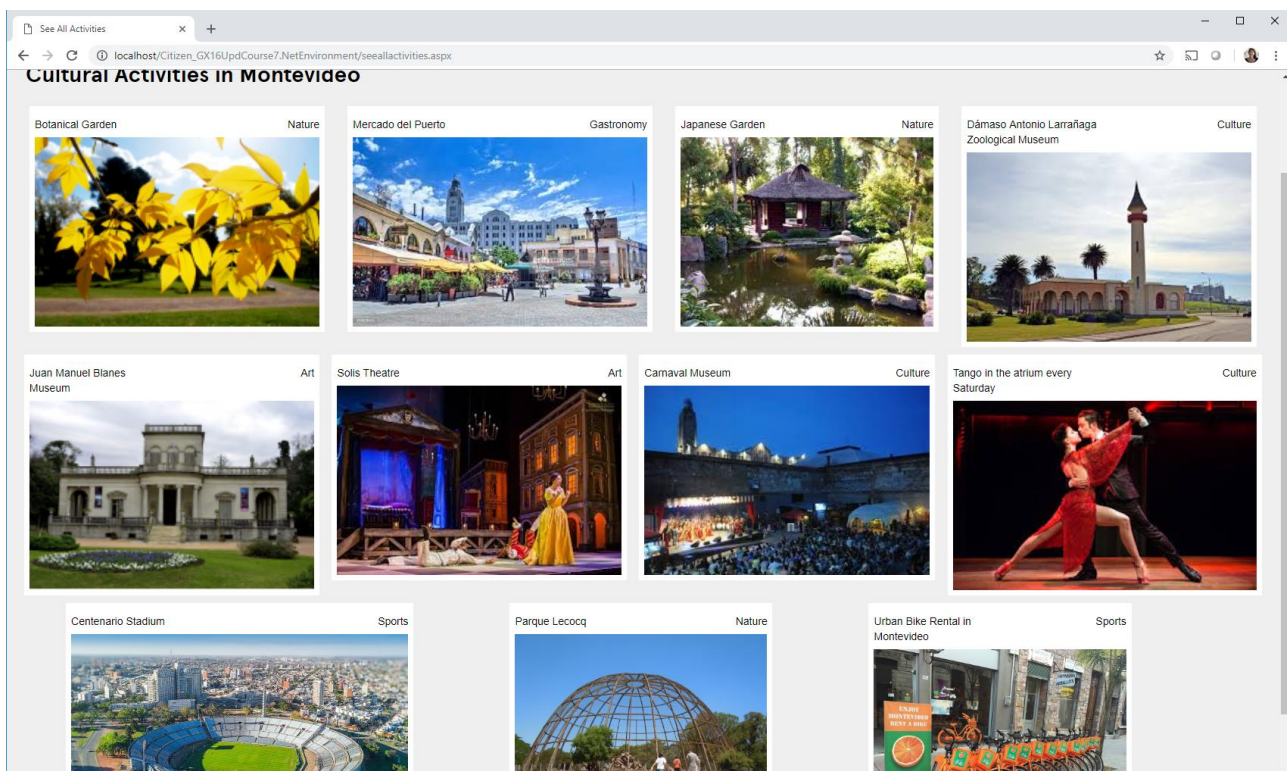
## FLEX GRID

Se você observar o painel main do frontend web (chamado CitizenWeb), além de ver as atividades culturais destacadas, é oferecida uma opção “See all Activities” para ver todas as atividades:



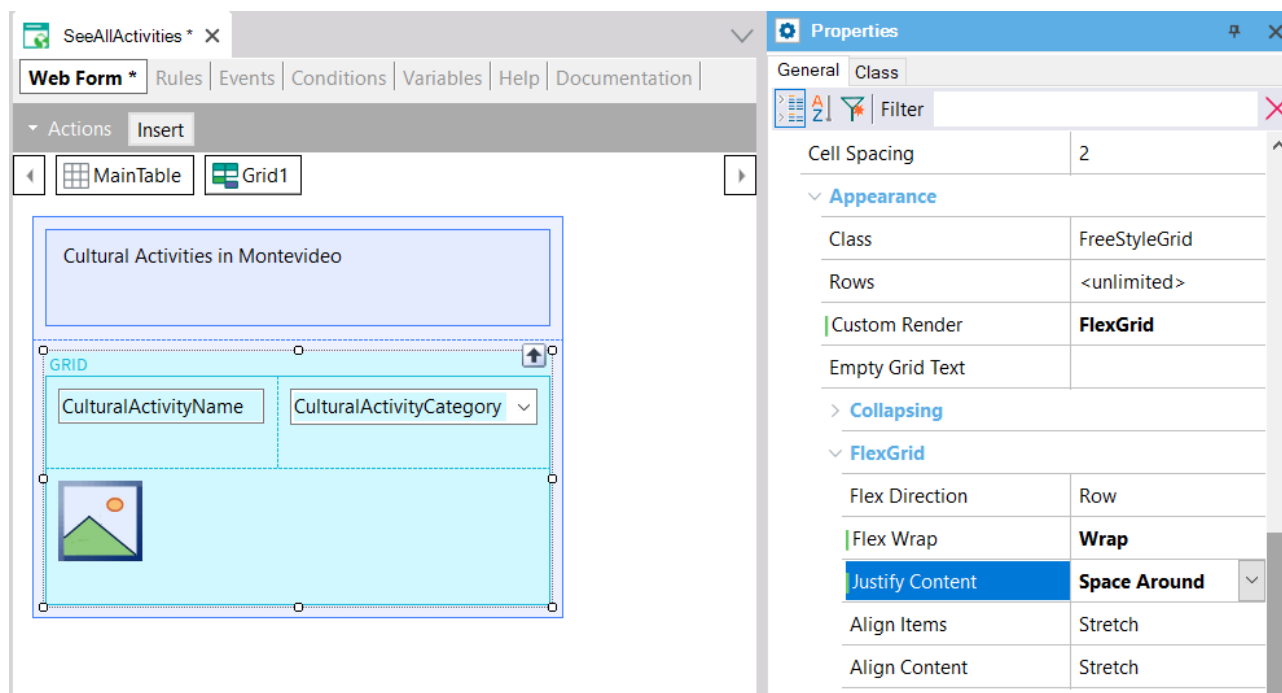


Nessa outra tela queremos que o grid seja capaz de mostrá-las de forma variável, de acordo com seu tamanho:



Como você implementa isso?

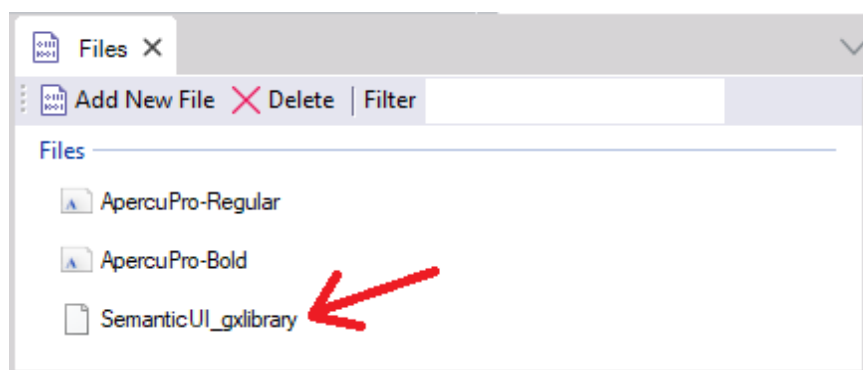
SOLUÇÃO:



Observe as diferenças com Horizontal Grid que tinha antes.

## BASE STYLE

Se você editar os files da KB, descobrirá que existe:



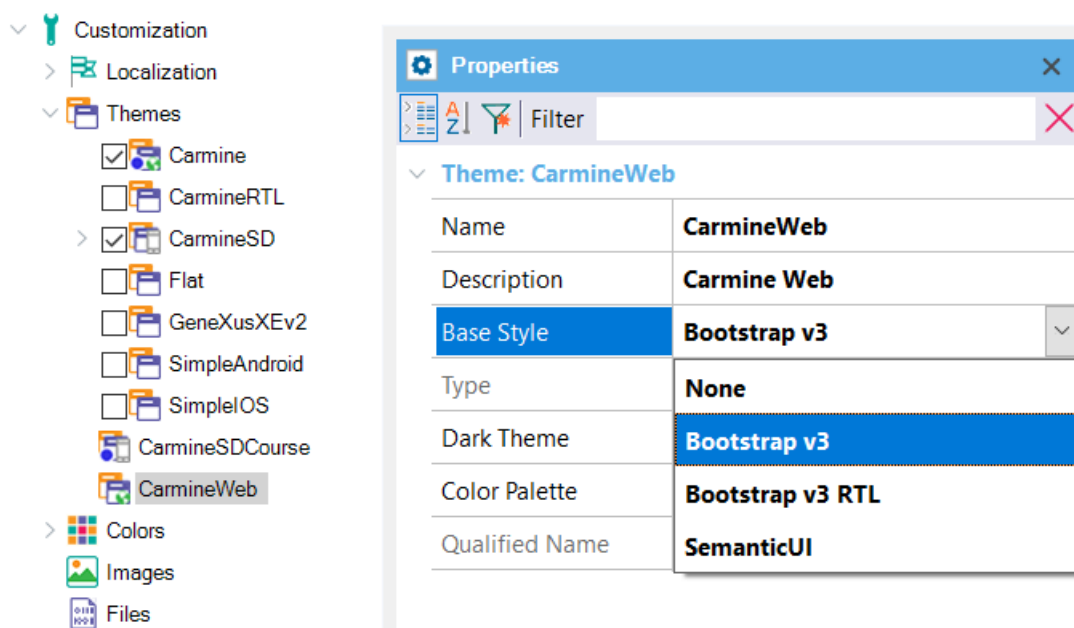
Também encontrará uma nova propriedade "Base Style" nos objetos:

- Theme
- User Control

Esta propriedade permite atribuir um estilo base a esses objetos.

Na propriedade serão tomados os valores dos Files da KB que tenham extensão "gxlibrary" (são um zip que contém todos os arquivos css, js, assets, etc. fornecidos pelos designers ou o Design System utilizado)

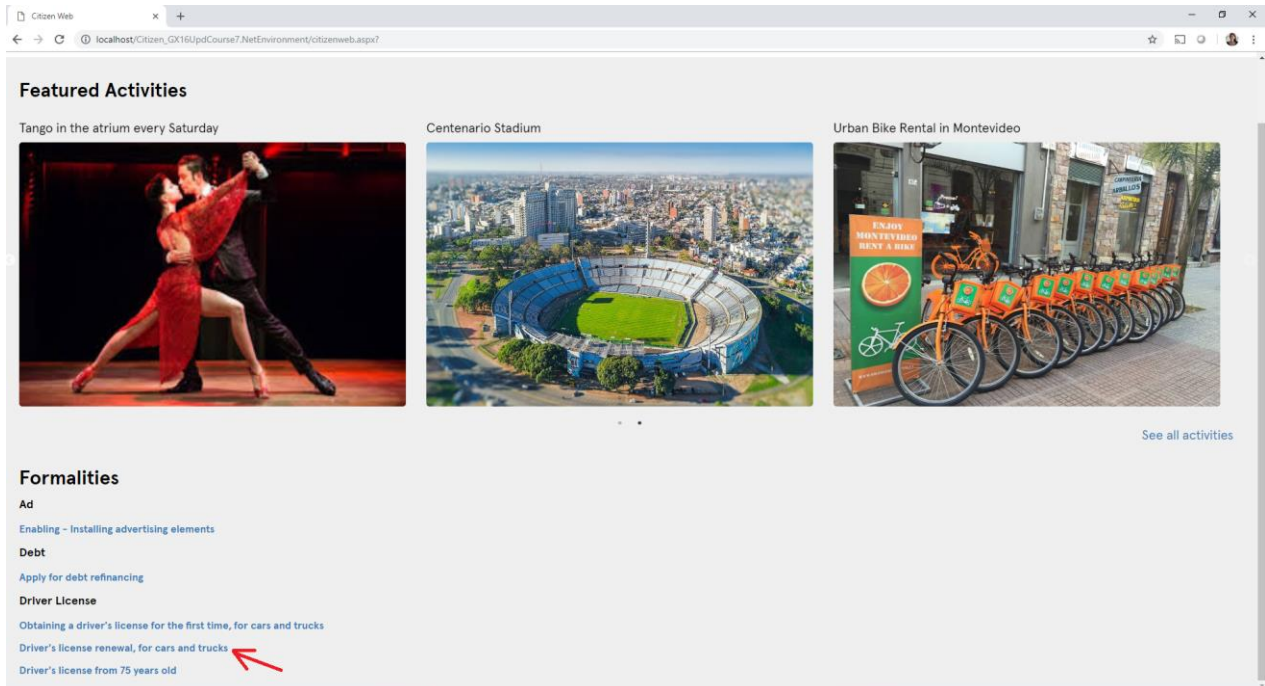
No nível do theme, encontrará a propriedade:



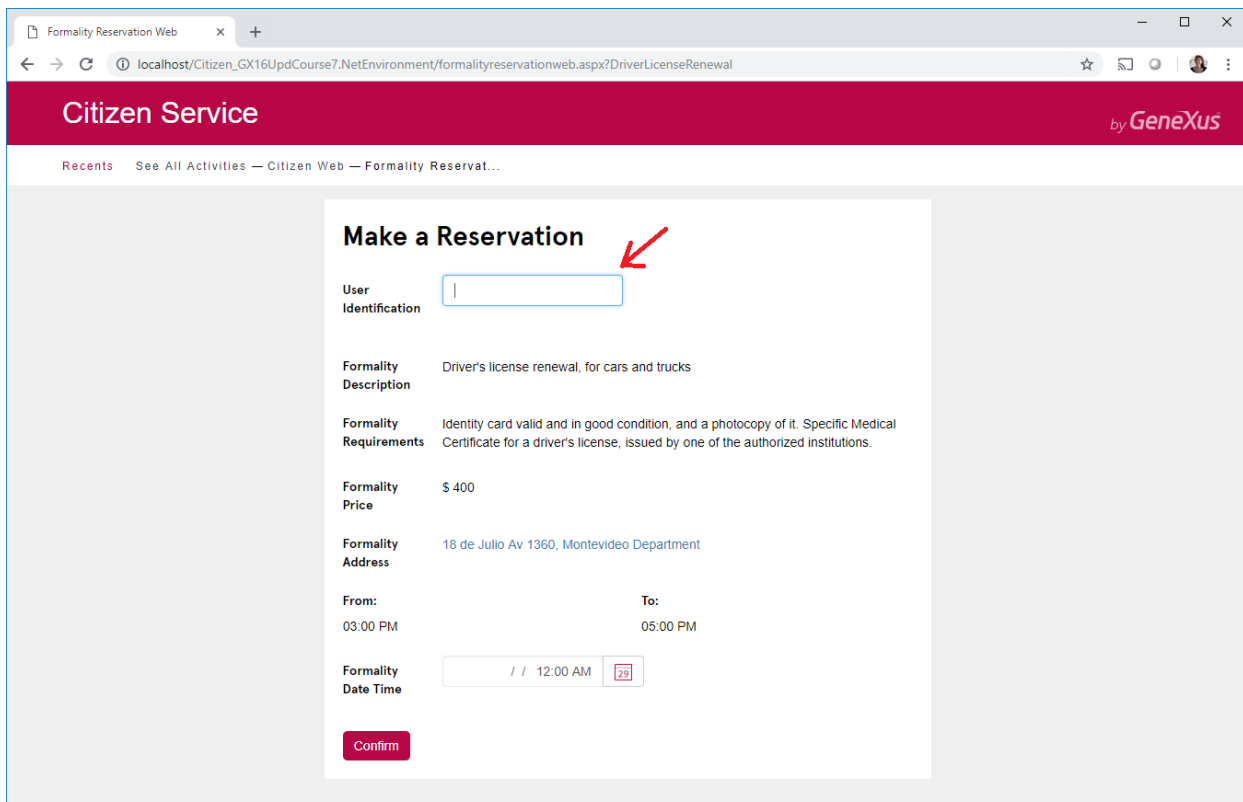
E no nível do objeto, a veremos imediatamente quando criamos um User control.

## USER CONTROL

Se agora a partir da página principal do frontend web, escolhermos um procedimento:




De maneira semelhante à app móvel nativa, nos solicita identificador de usuário para poder reservar horário para este procedimento:



Ao inserir um ID de usuário válido e deixar o campo, queremos ver as informações do usuário da seguinte forma:

## Make a Reservation

User Identification



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

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

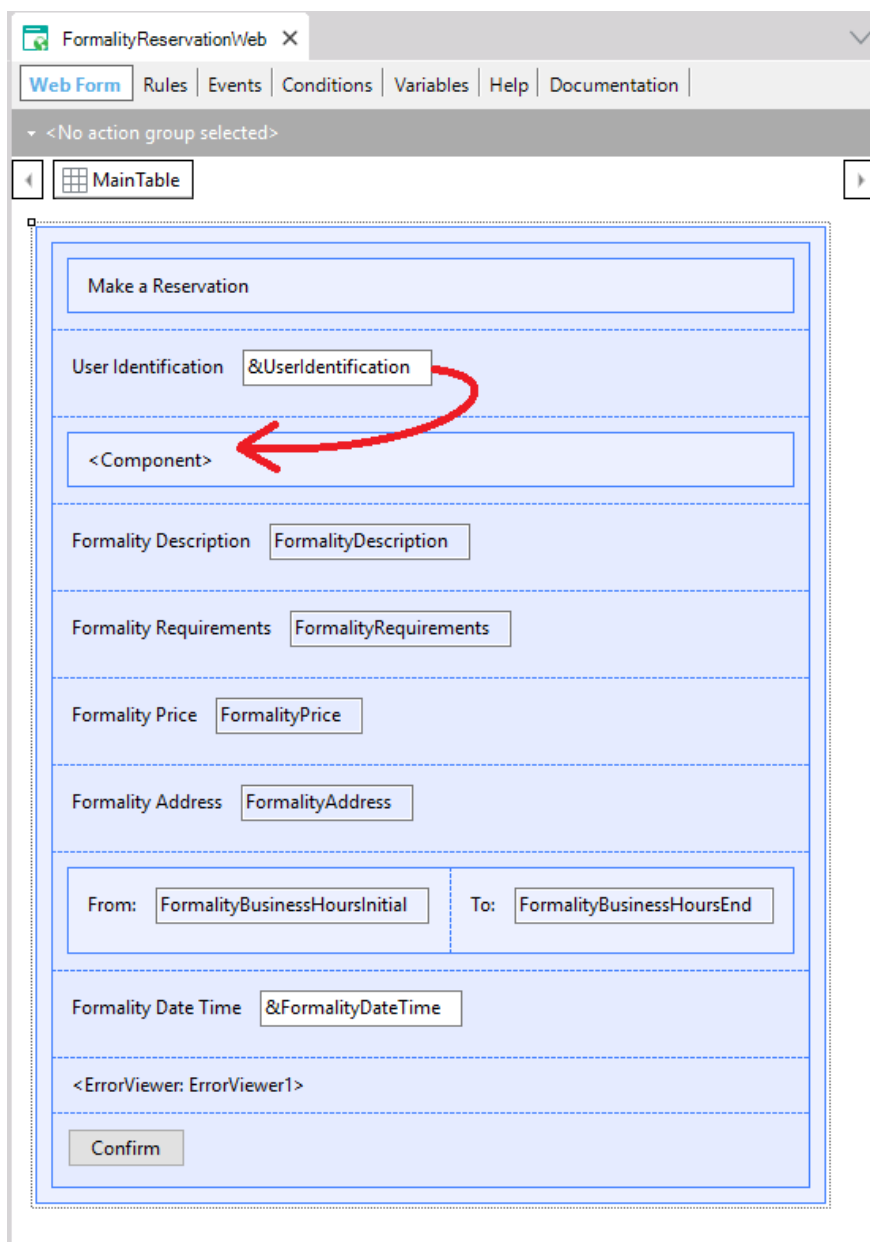
**Formality Requirements** Identity card valid and in good condition, and a photocopy of it. Specific Medical Certificate for a driver's license, issued by one of the authorized institutions.

**Formality Price** \$ 400

**Formality Address** 18 de Julio Av 1360, Montevideo Department

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

Observe que esse Card é carregado no componente de nome WCUserInformation que é carregado no web form:

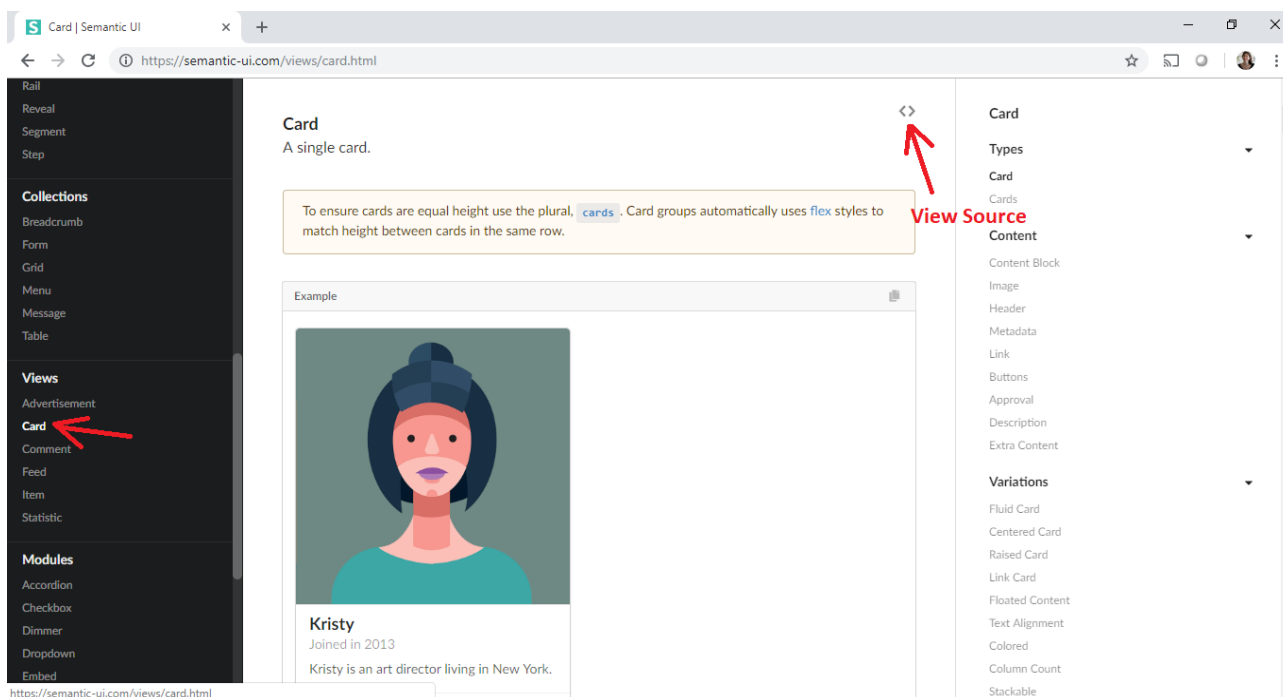


Como você implementa esse componente (WCUserInformation) que em sua KB está vazio?

SOLUÇÃO:

Trata-se de um controle Card de SemanticUI. Se você for ao site de SemanticUI e procurar por Card, encontrará diferentes tipos destes controles. Escolha por exemplo o Card mais simples e visualize o html, para copiá-lo:





Aqui nós damos escrito para que você possa copiá-lo depois e não perder tempo:

```
<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>
```

Crie um objeto User Control em GeneXus e copie em seu "Screen Template" o html anterior:

Parametrize as propriedades, de acordo com o que você deseja ver. No nosso caso, ficará assim:

The code editor shows the following parametrized HTML structure:

```

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>

```

The Properties window displays the following information for the **User Control: UserInformationCard**:

|                   |                       |
|-------------------|-----------------------|
| Name              | UserInformationCard   |
| Description       | User Information Card |
| Module/Folder     | Root Module           |
| Is Control Type   | False                 |
| References        |                       |
| Base Control Type | None                  |
| <b>Base Style</b> | <b>SemanticUI</b>     |
| Qualified Name    | UserInformationCard   |
| Object Visibility | Public                |

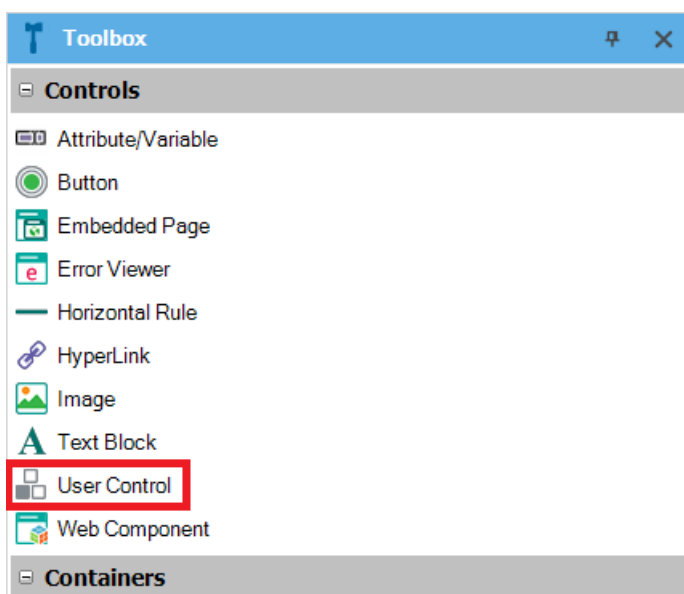


Aqui nós damos escrito a você o código anterior para copiá-lo:

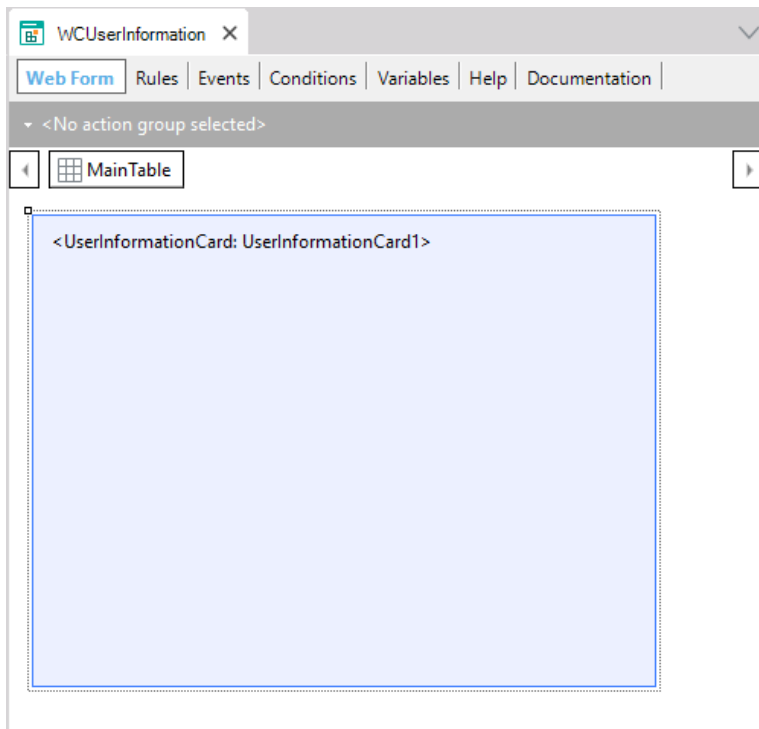
```
<div class="ui card card-extra-width">
  <div class="image">
    
  </div>
  <div class="content">
    <a class="header">{{UserName}}</a>
    <div class="meta">
      <span class="date">{{UserBirthDay}}</span>
    </div>
    <div class="description">{{UserAddress}}</div>
    <div class="description">{{UserEMail}}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

Verifique se a propriedade Base Style do objeto User Control tem o valor "SemanticUI" definido. Caso contrário, selecione esse valor.

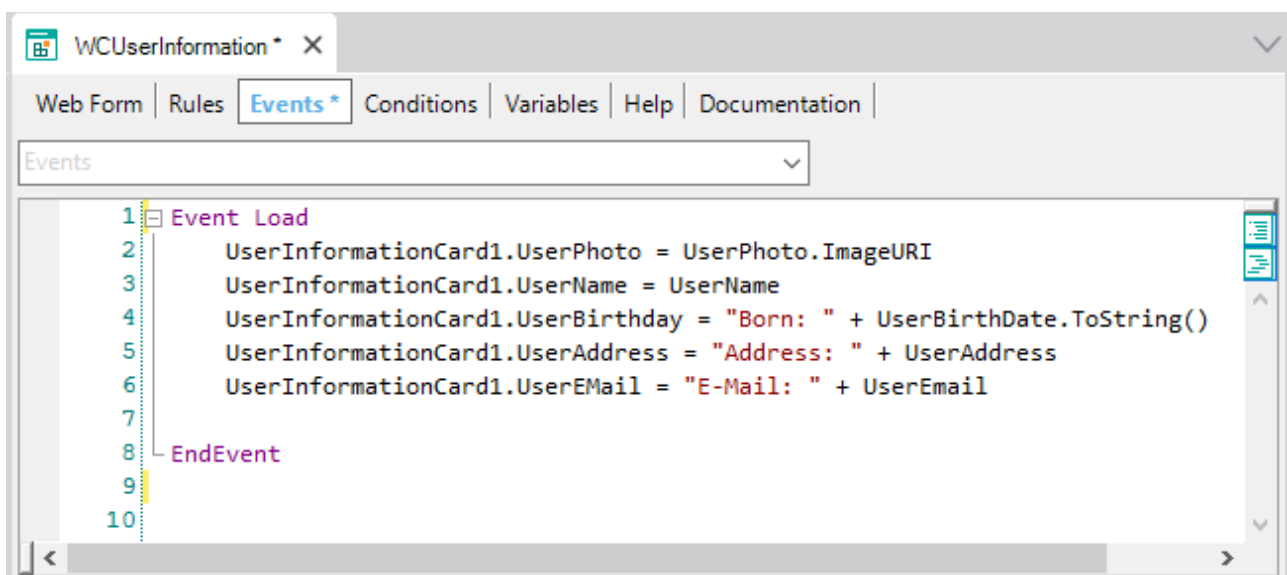
No componente WCUserInformation, você deve inserir o user control que acabou de criar. Na Toolbox, encontrará:



Arraste-o para o Web form e selecione o User Control que criou anteriormente:



Em seguida, inicialize as propriedades do User Control com base nos atributos:



**Nota:** Para confirmar que foi carregada corretamente a Base Library (neste caso, SemanticUI), verifique o diretório web da KB (Tools → Explore Target Environment Directory) e veja se existe a pasta "SemanticUI". Se não existir, sugerimos que você faça um Rebuild do objeto CitizenWeb.


Execute.

## ADICIONAR UMA AÇÃO AO USER CONTROL

Opcionalmente, você pode querer permitir a partir do Card editar informações do usuário.

### 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

Isso não veio exatamente assim no html source do controle de SemanticUI. Você pode personalizar seu objeto User Control adicionando:

```

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>

```

E então no 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

```

Teste em execução.

Falta programar o evento. Para fazer isto:

```

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
    
```

Teste novamente.

## KB SOLUÇÃO

Você pode baixar do GeneXus Server a KB solução deste prático para comparar resultados. É a versão da KB chamada CitizenSolutionPartOne.