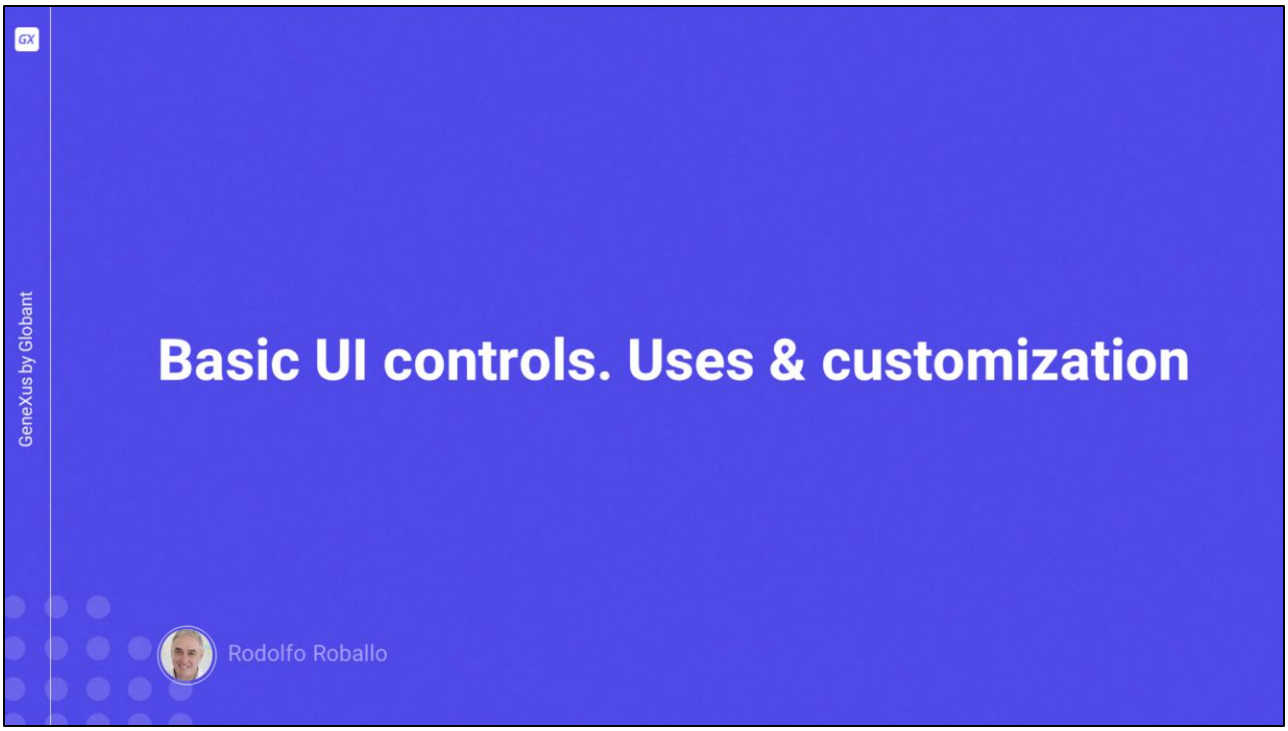
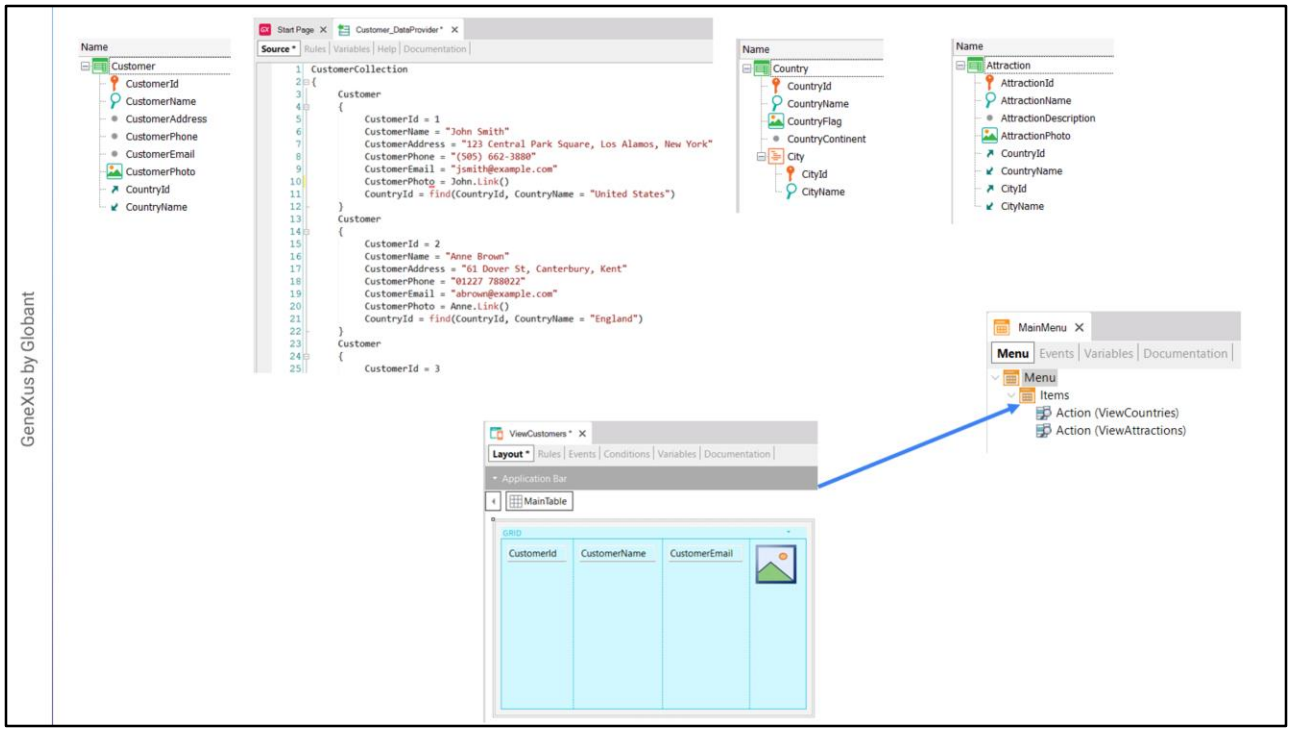


Algo que é fundamental em qualquer aplicação informática é o uso correto dos controles de tela disponíveis, para proporcionar a melhor experiência de usuário.

Neste vídeo veremos os controles mais utilizados, como podemos organizá-los no layout do panel e personalizar sua aparência.

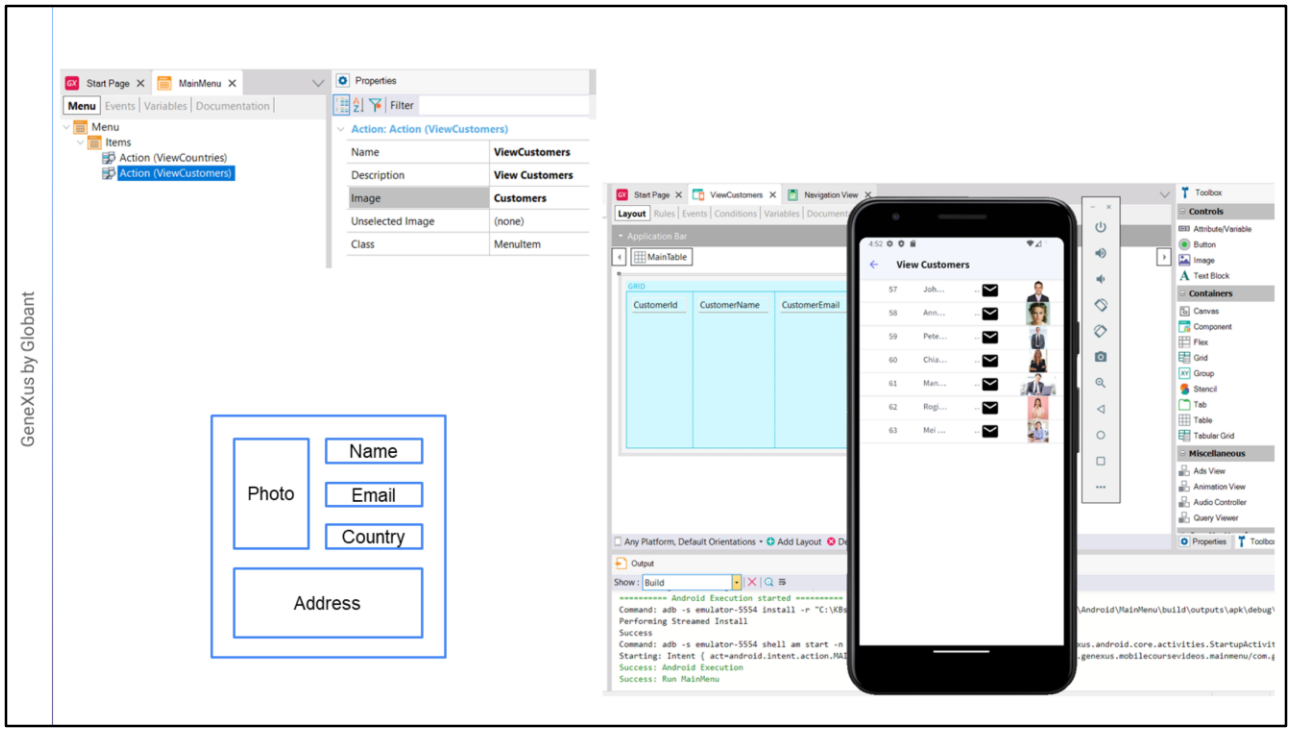


Vamos criar um panel para visualizar os dados dos clientes da agência de viagens.

Antes de mais nada, importamos para nossa KB o arquivo Customers.xpz que encontramos na seção Materiais da página do curso.

Vemos que foi importada a transação Customer com atributos para os dados principais como o identificador, nome, endereço, telefone, e-mail e sua foto. Também vemos que foi importado o data provider Customer_DataProvider, que inclui os dados dos clientes e também foram importados mais outros objetos.

Para ser possível ver os dados dos clientes, criamos o panel ViewCustomers, arrastamos um controle grid com os atributos CustomerId, CustomerName, CustomerEmail, CustomerPhoto e CountryName... e salvamos.



Agora abrimos o objeto MainMenu e arrastamos o panel ViewCustomers para o nó Items e na propriedade Image da action criada atribuímos a imagem Customers.

Observação: para ver o menu como é mostrado aqui, crie o Panel chamado ViewAttractions, arraste-o para o nó Items do objeto MainMenu e atribua em sua propriedade Image a imagem Attractions. Trabalharemos com esse Panel em um vídeo futuro.

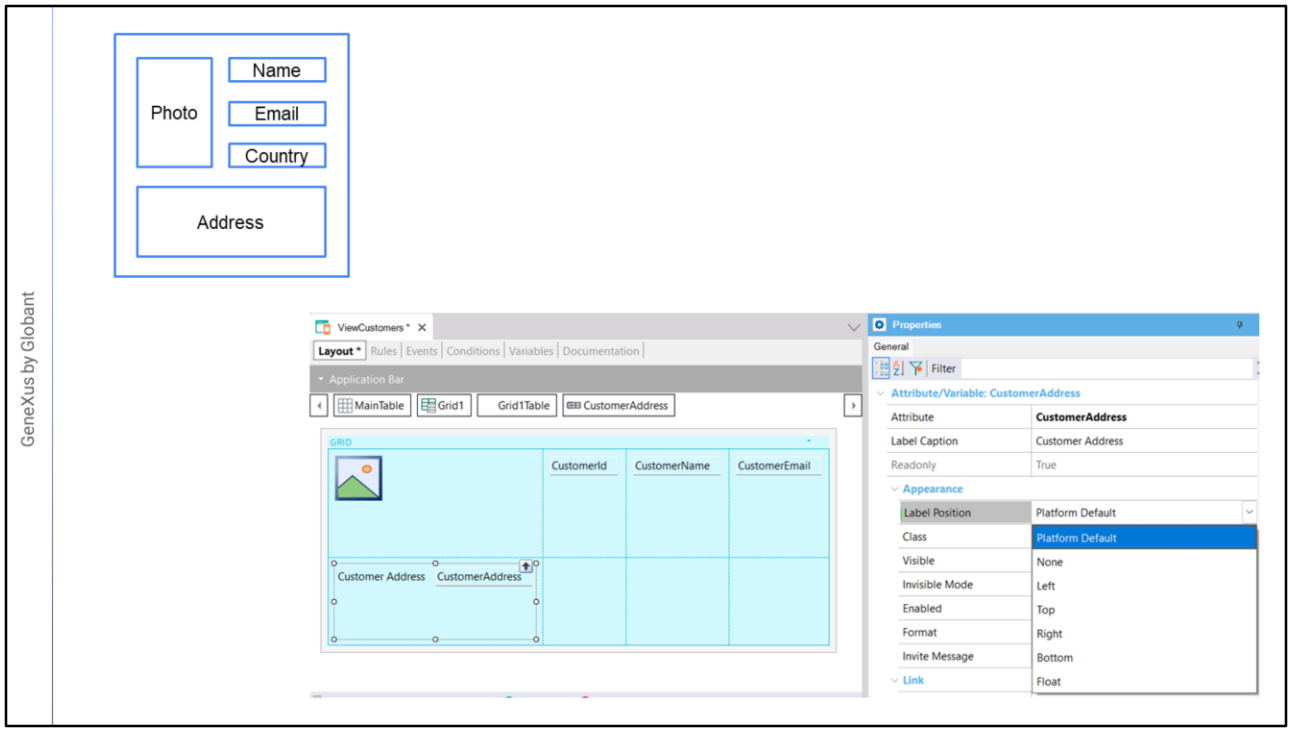
Recomendamos sempre executar a aplicação com o emulador previamente aberto, para otimizar os tempos de prototipagem. Isto significará que quando GeneXus quiser instanciar um emulador para executar a aplicação mobile, encontrará esse emulador em execução e o utilize, economizando assim o tempo de instanciação do dispositivo virtual e a execução será muito mais rápida. Agora sim, pressionamos F5.

Vemos que a lista dos clientes aparece sem nenhum design, inclusive há alguns dados que não conseguimos ver por falta de espaço.

Tentaremos chegar ao seguinte design tipo ficha, que nos parece o mais adequado. Cada linha do grid de clientes será exibida com este desenho, um cliente por linha.

Notemos que o Id não está incluído neste design e que é desejado mostrar o endereço

do cliente que não tínhamos incluído no grid original. Vamos adicionar o endereço clicando com o botão direito sobre o grid e escolhendo “Adicionar atributo ou variável” e Customer Address.



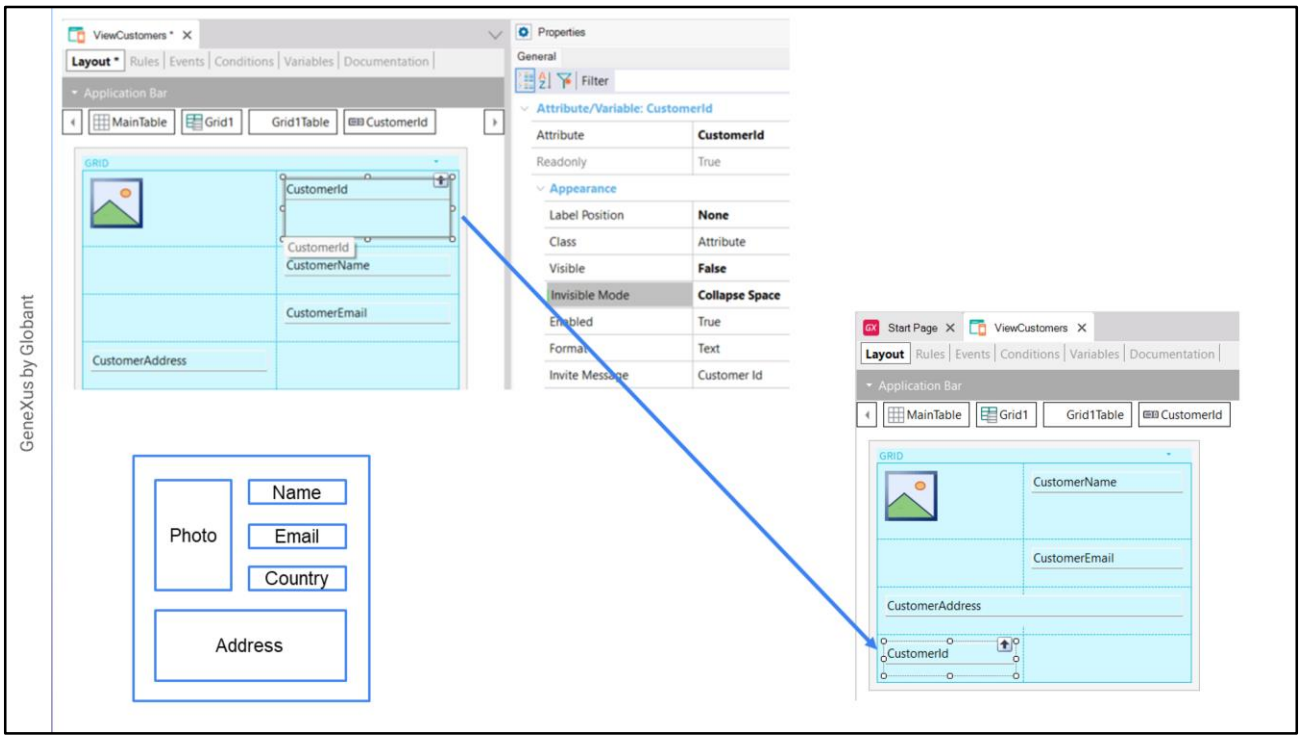
Vamos começar movendo o atributo da foto para o lado esquerdo do grid e arrastamos o campo de endereço para debaixo da foto.

Vemos que CustomerAddress tem um rótulo à esquerda. Se formos às propriedades do atributo, vemos que na seção Appearence, a propriedade Label Position tem o valor Platform Default. Para Android é o valor Top, acima do conteúdo e para iOS é à esquerda.

Se abrirmos o combo vemos uma série de valores que alteram o local padrão onde será exibido o rótulo associado ao atributo.

O valor None significa que não será exibido o rótulo. Os valores Left, Top, Right e Bottom determinam que o rótulo será exibido à esquerda, acima, à direita ou abaixo do campo de conteúdo.

Com o valor Float o rótulo será exibido na posição do conteúdo do atributo ou variável e quando o usuário começa a digitar no campo, o rótulo se moverá para cima dele como se flutuasse. Isto só tem efeito em campos editáveis e em dados do tipo numérico ou do tipo caractere; nos demais tipos será adotado o valor Top.



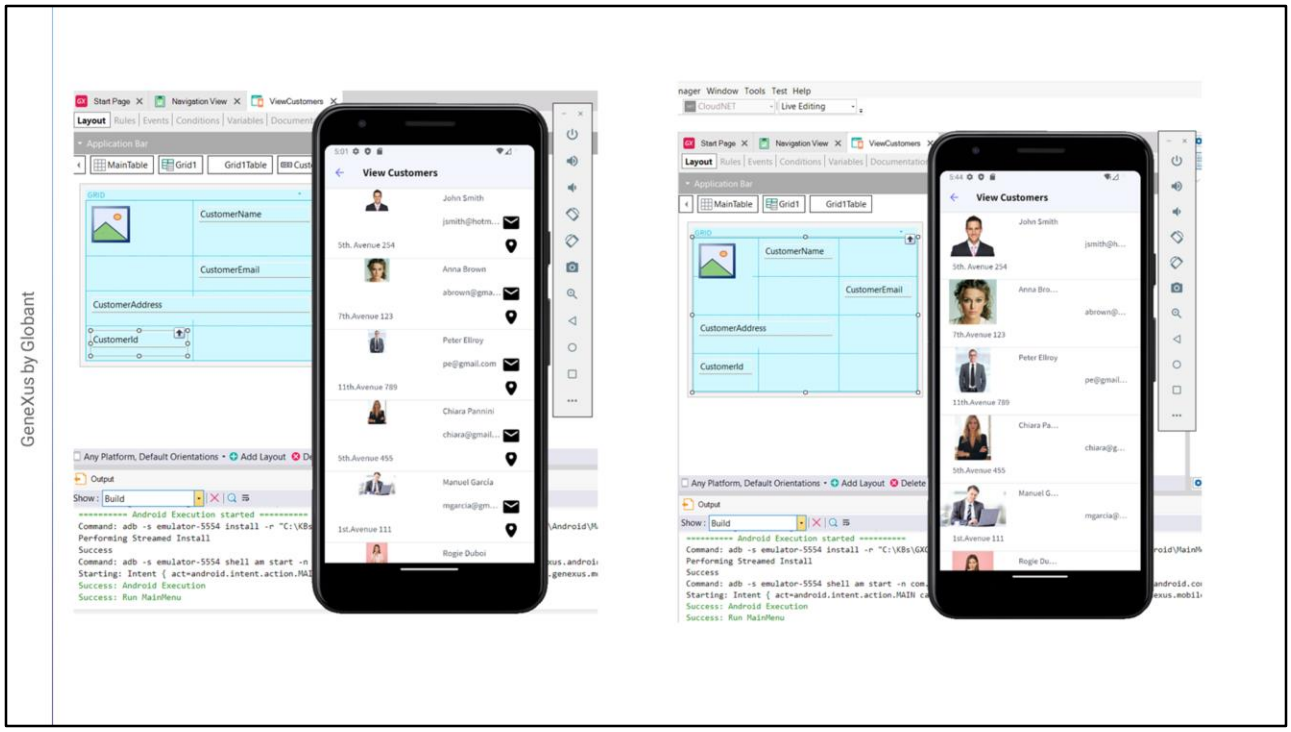
Para CustomerAddress vamos escolher o valor None, com o qual o rótulo desaparece.

Movemos os atributos de nome, e-mail e país para baixo do CustomerId. Como não queremos mostrar o CustomerId, podemos removê-lo ou definir sua propriedade Visible como False. Caso necessitemos obter outros dados de um cliente selecionado, devemos passar por parâmetro o valor de seu identificador, portanto devemos ocultar o atributo usando a propriedade Visible em False, em vez de excluí-lo.

Abaixo da propriedade Visible está a propriedade Invisible Mode com o valor padrão Keep Space e isso significa que mesmo que ocultemos o atributo como fizemos, ainda será mantido o espaço que ele ocupa quando visualizado.

Se abrirmos o combo vemos que também temos o valor Collapse Space, que significa que o atributo ficará oculto e será recolhido o espaço, permitindo que os demais controles ocupem o espaço disponível. No entanto, esta possibilidade só funciona para recolher informação de linhas (recolher horizontal), mas não de colunas (recolher vertical), portanto, no nosso caso, como o CustomerId está na mesma linha que CustomerPhoto, devemos remover o CustomerId de lá e se o movemos para a última linha, lá sim ele será recolhido.

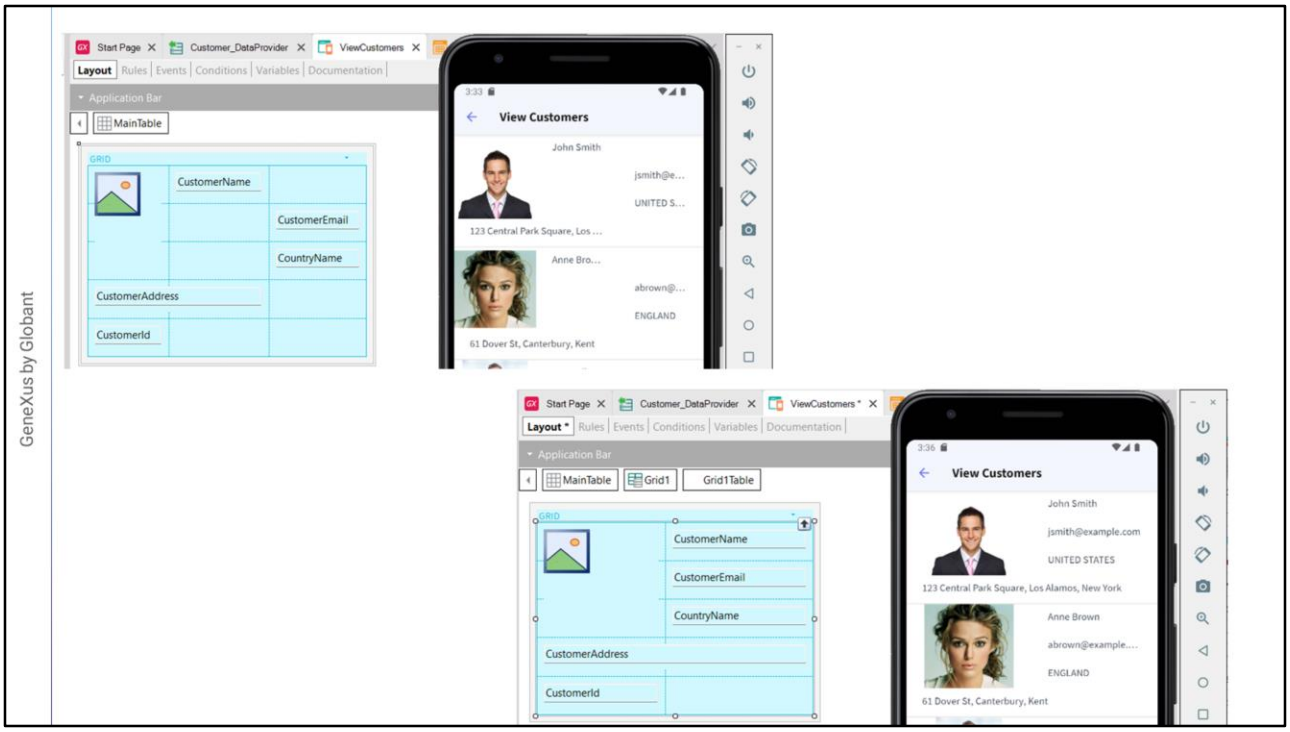
Vamos organizar um pouco os controles para respeitar o design que queríamos alcançar. Como precisamos que o endereço ocupe o espaço das duas colunas, vamos até a seção Cell Information e na propriedade Col Span definimos o valor 2.
E pressionamos F5.



Tocamos no ícone View Customers e vemos que é carregada a lista de clientes.
Embora se pareça com o que precisamos, não se vê muito bem. Notamos que efetivamente não aparece CustomerId e que foi recolhido o lugar que ele ocuparia.
Vamos tentar melhorar um pouco o tamanho da foto e fazer com que ocupe três linhas de altura, definindo a propriedade Row Span como 3.

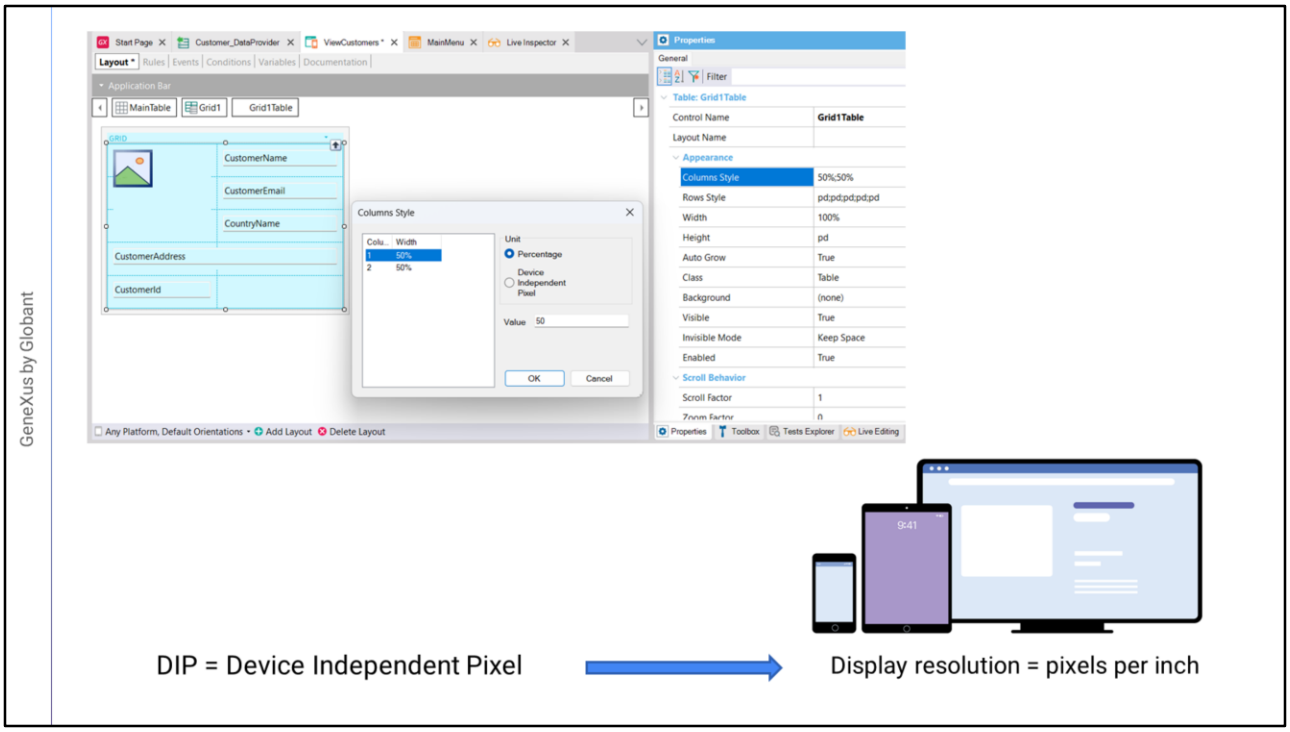
Antes de continuar com outras mudanças, vamos utilizar a funcionalidade do Live Editing fornecida pelo IDE de GeneXus.

Para ver as alterações com o Live Editing, alteramos neste combobox o valor Release para Live Editing e executamos novamente a aplicação.
Comprovamos que o Live Editing está em funcionamento vendo a aba Live Editing da janela de Properties, onde vemos que o IDE está conectado ao emulador que tínhamos em execução. Vemos também que foi aberta a janela do Live Inspector, onde podemos identificar os controles de tela e suas propriedades.



Voltando ao nosso design, vemos que ficaram o nome do cliente, seu e-mail e seu país em diferentes colunas. Vamos colocá-los na mesma coluna e vemos que graças ao Live Editing, a alteração foi refletida imediatamente no emulador, sem a necessidade de salvar ou executar novamente.

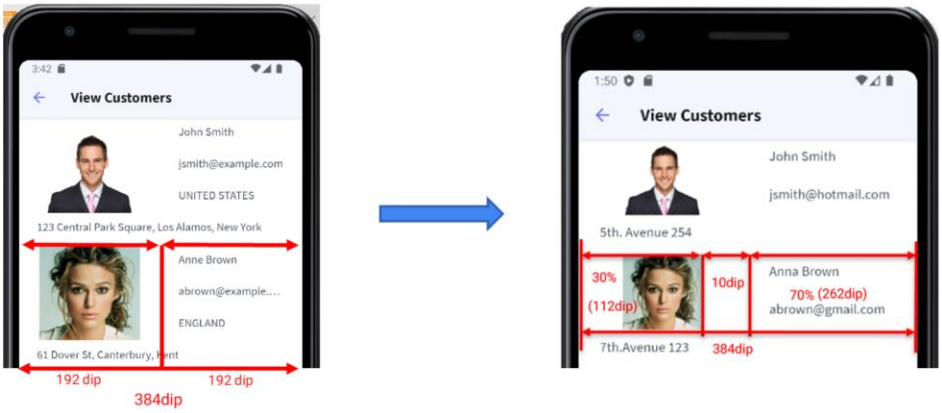
Vamos agora apagar a coluna que está vazia, então clicamos com o botão direito e escolhemos Delete Column. Vemos novamente que já vemos a mudança no emulador.



Para adequar o tamanho das colunas e linhas, selecionamos a tabela do grid, chamada Grid1Table e na seção Appearance, temos as propriedades Columns Style, Rows Style, Width, Height e AutoGrow que nos permitem alterar o tamanho e o comportamento dos controles.

Em Columns Style diz 50%;50% o que significa que a tabela tem duas colunas e que cada uma está ocupando 50% da largura disponível. Abrimos esta propriedade e vemos as duas colunas identificadas e que cada uma tem a unidade selecionada em Porcentagem com um valor de 50.

Vemos que a outra unidade de medida é DIPS: Device Independent Pixels. O Device Independent Pixel corresponde a uma abstração de um pixel que em seguida uma aplicação converte em pixels físicos, o que permite dimensionar para diferentes tamanhos de tela. O dip para cada plataforma possui um número diferente de pixels.

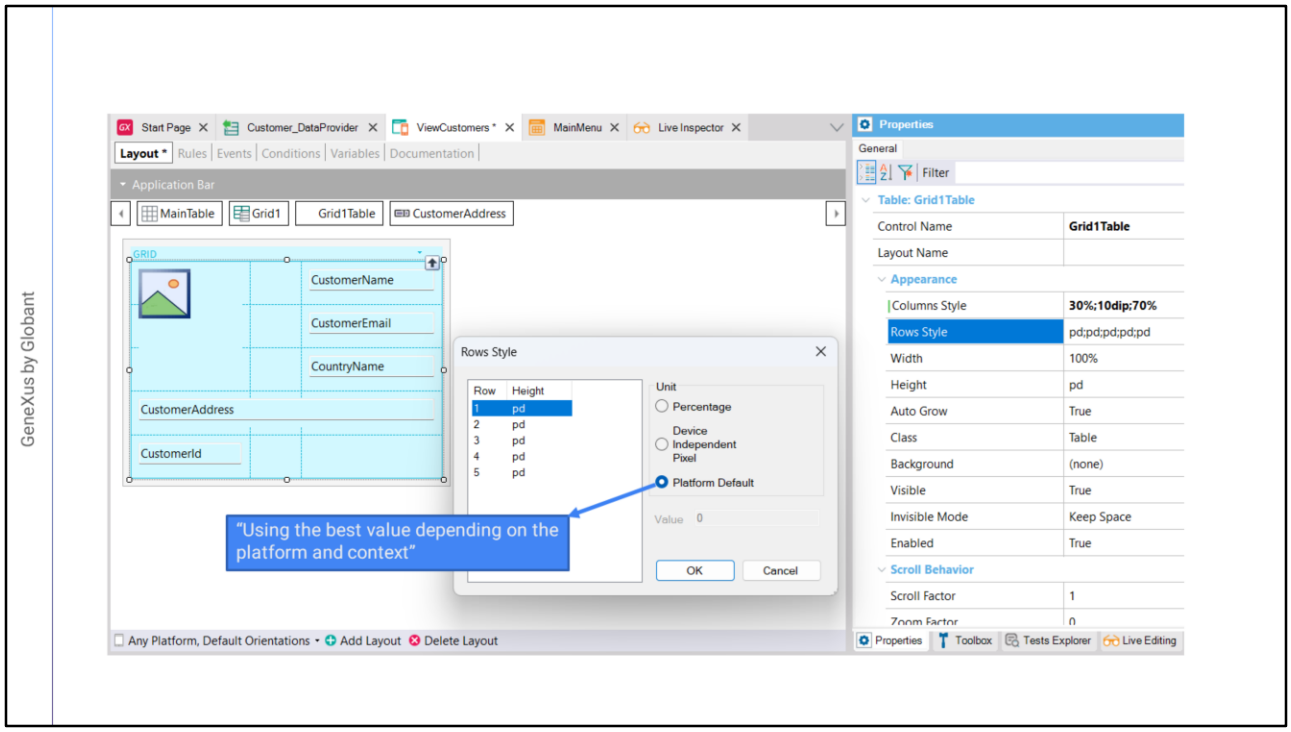


A largura total é fixa e é de 384 dips, portanto com este desenho cada coluna ocupa 50%, ou seja 192 dips.

Mas se existir alguma coluna definida em dips, os valores percentuais são relativos ao valor que resulta da subtração da largura total, os valores fixos (em dips). Por exemplo, se quisermos separar a coluna da foto da coluna dos textos, podemos adicionar uma coluna como separador.

Se tivéssemos três colunas, a primeira com 30%, a segunda com valor fixo de 10 dips e a terceira com 70%, os valores que irão assumir a primeira e a terceira são obtidos aplicando essas percentagens ao valor resultante de subtrair a soma dos valores fixos (aqui apenas um, 10 dips da coluna separadora) da largura da tabela.

Devido a isso, a quantidade de espaço disponível para a primeira e terceira colunas é de: 384 – 10 = 374 dips, portanto a primeira coluna terá 30% de 374 o que equivale a 112 dips e a terceira os 70% restantes, que são 262 dips.

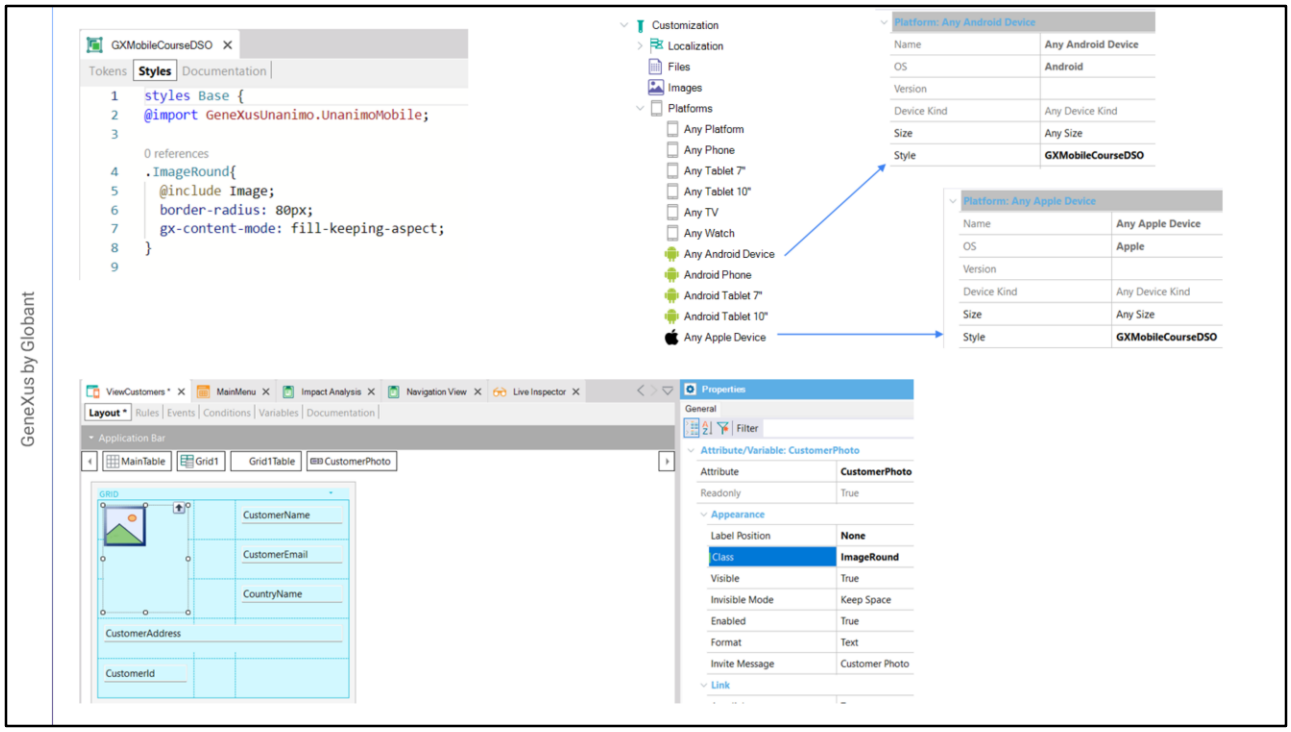


Vamos nos posicionar na foto, clicamos com o botão direito e escolhemos Insert column after. Vamos especificar os valores do Columns Style como vimos, colocando 30% para a primeira coluna, 10 dips para a segunda e 70% para a terceira, assim vemos o resultado e graças ao fato de estarmos usando Live Editing vemos que mudou e agora temos um espaço entre o nome e a imagem.

Se agora olharmos para a propriedade Rows Style, vemos que existem 5 linhas com o valor pd: Platform Default.

Este valor difere de plataforma para plataforma, e para uma mesma plataforma também depende do conteúdo da célula e se o campo possui label ou não e, caso possua, se o rótulo será exibido acima ou à esquerda. O valor pd corresponde a: "Usar o Melhor Valor Dependendo da Plataforma e do Contexto".

Por exemplo, para Android, com Label Position = Top, corresponde a 64 dips, enquanto no iOS a 53 dips.



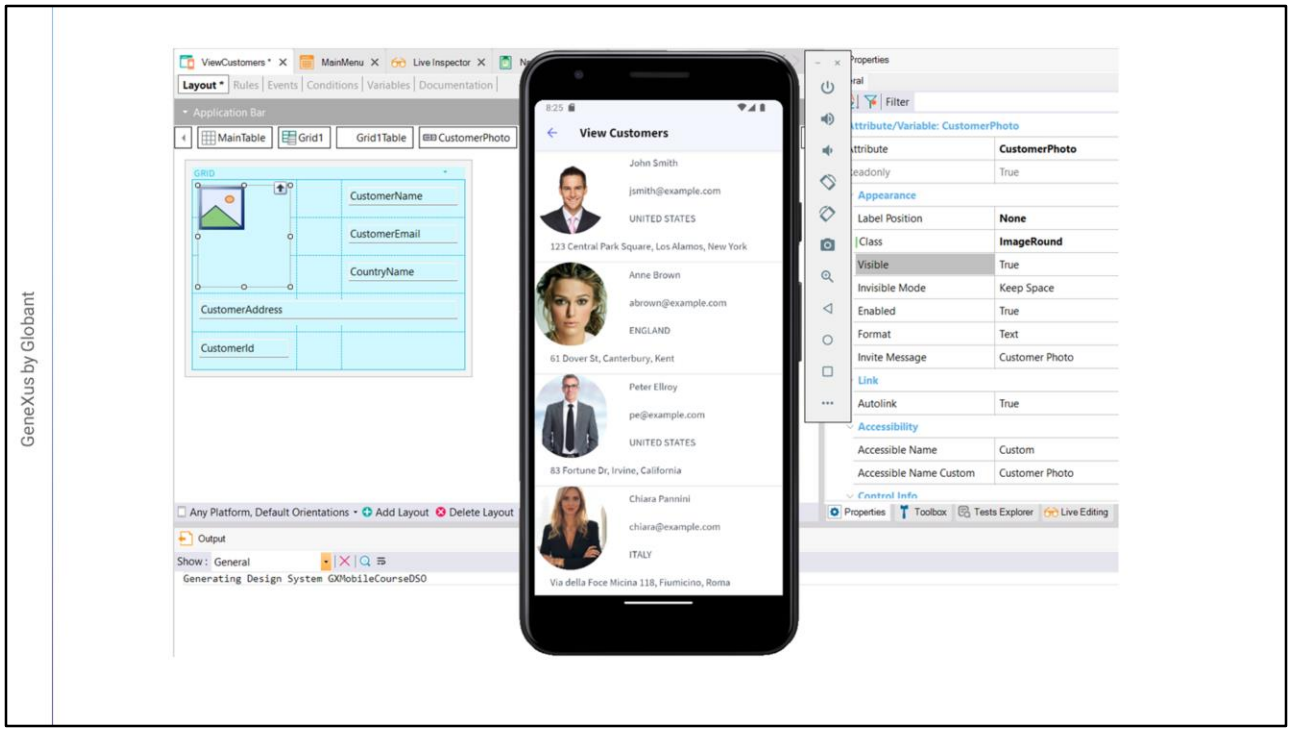
Revisamos a execução e as linhas aparecem muito bem, então deixamos estes valores.

Agora vamos mudar a aparência da foto para que seja visualizada redonda. Para isso vamos atribuir ao controle da foto uma classe com propriedades que permitem isso.

Abrimos o objeto design system GXMobileCourseDSO que foi importado com o xpz e em sua aba Style criamos uma classe chamada ImageRound. Atribuímos à propriedade border-radius um valor de 80 pixels, o que fará com que a imagem seja visualizada redonda, e à propriedade gx-content-mode o valor fill-keeping-aspect, com o que a imagem será ajustada automaticamente ao tamanho disponível sem alterar sua relação de aparência, ou seja, a proporção entre a largura e a altura.

Para que este Design System Object seja o que é levado em consideração em nossos objetos mobile, devemos atribuí-lo às plataformas nas quais iremos gerar. Abrimos Customization e então selecionamos AnyAndroidDevice e em sua propriedade Style atribuímos o design system GXMobileCourseDSO.
O mesmo fazemos para a plataforma AnyAppleDevice.

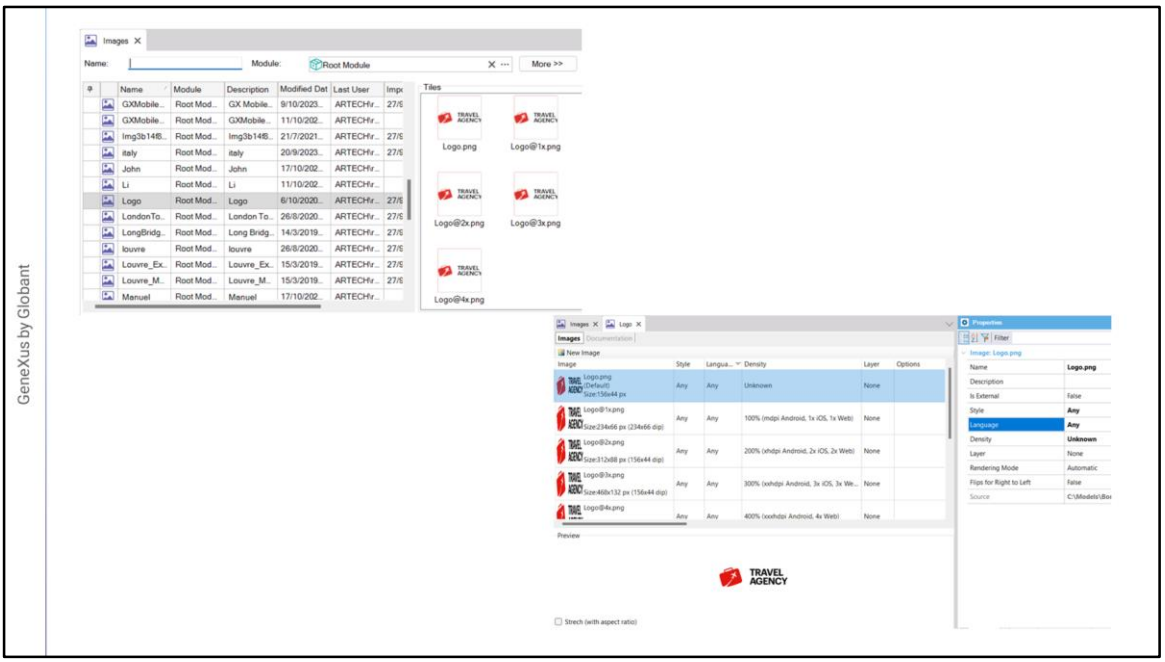
Agora selecionamos a foto e em sua propriedade Class escrevemos ImageRound.



Vemos que apenas atribuímos a classe ao atributo, são vistas as alterações graças ao Live Editing. Poderíamos até alterar os valores na classe e também veríamos a alteração refletida automaticamente, mesmo sem salvar ou compilar o objeto ou executar a aplicação. Isto favorece o desenvolvimento incremental.

No nosso exemplo estamos vendo a foto do cliente, que é uma imagem armazenada na base de dados, mas é frequente termos que usar uma imagem fixa, como um logotipo ou uma imagem de fundo.

Neste caso, devemos levar em consideração que ao adicionar a imagem à KB devemos, na verdade, adicionar várias imagens com diferentes resoluções, pois os dispositivos que executam a aplicação podem ter mais ou menos densidade e resolução de tela.



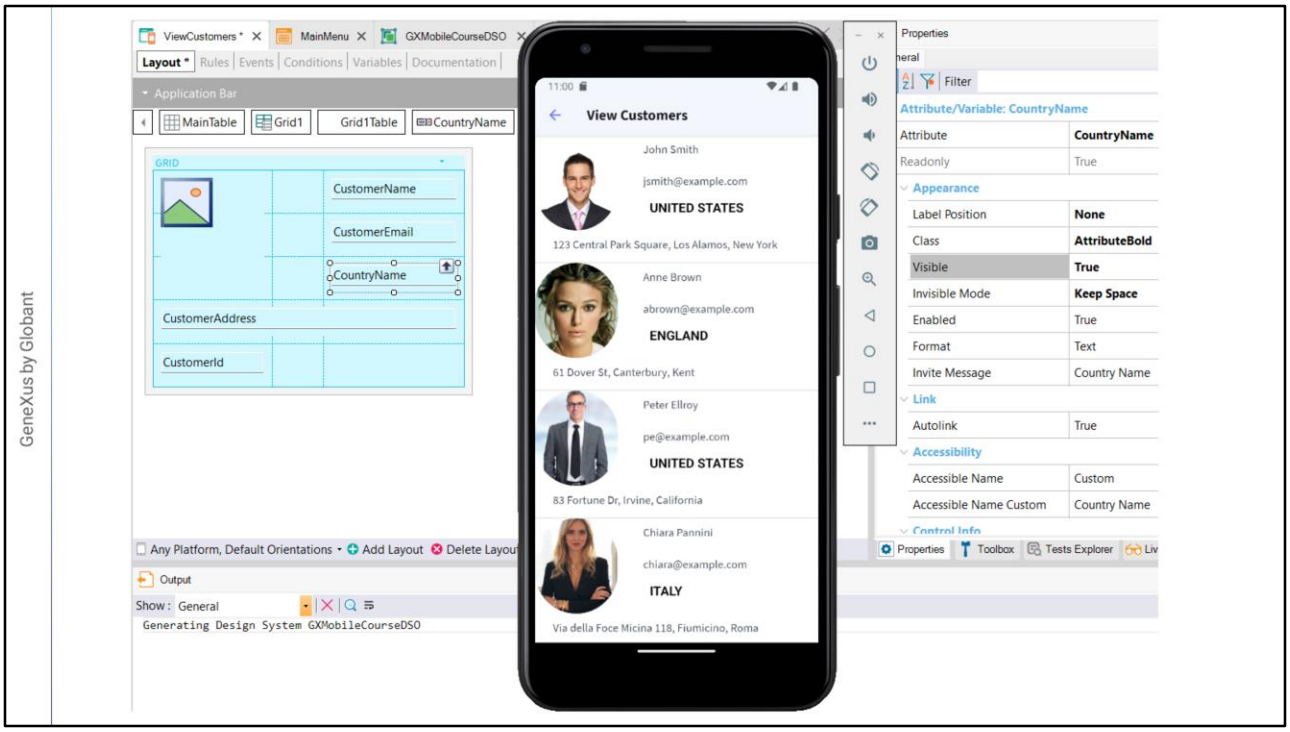
Se selecionarmos a imagem chamada Logo, vemos que na verdade existem várias imagens associadas. Se abrirmos a imagem, podemos ver que cada imagem tem uma resolução diferente e ao executarmos a aplicação será selecionada automaticamente a adequada dependendo da resolução de tela do dispositivo.

Vemos também que para cada imagem podemos especificar qual estilo usará, ou seja, qual é o Design System associado. Isto permite carregar imagens diferentes associadas a diferentes Design System Objects. No nosso exemplo, o valor Any significa que a mesma imagem será utilizada para todos os estilos definidos na aplicação.

Também podemos associar um idioma específico, por exemplo no caso de a imagem incluir um texto no seu gráfico e que este texto seja diferente por idioma, de forma que podemos carregar várias imagens associadas, uma para cada idioma definido em nossa aplicação.

Podemos fazer upload de imagens com diferentes densidades. Por exemplo, o valor de 100% corresponde à densidade média (aproximadamente 160 dips) que é tomada como linha de base tanto para Android quanto para Web e para iOS, neste último caso, o valor corresponde a telas que não são Retina Display. Lembremos que os dpi (device independent pixels) são então convertidos em pixels reais de acordo com a resolução de cada dispositivo.

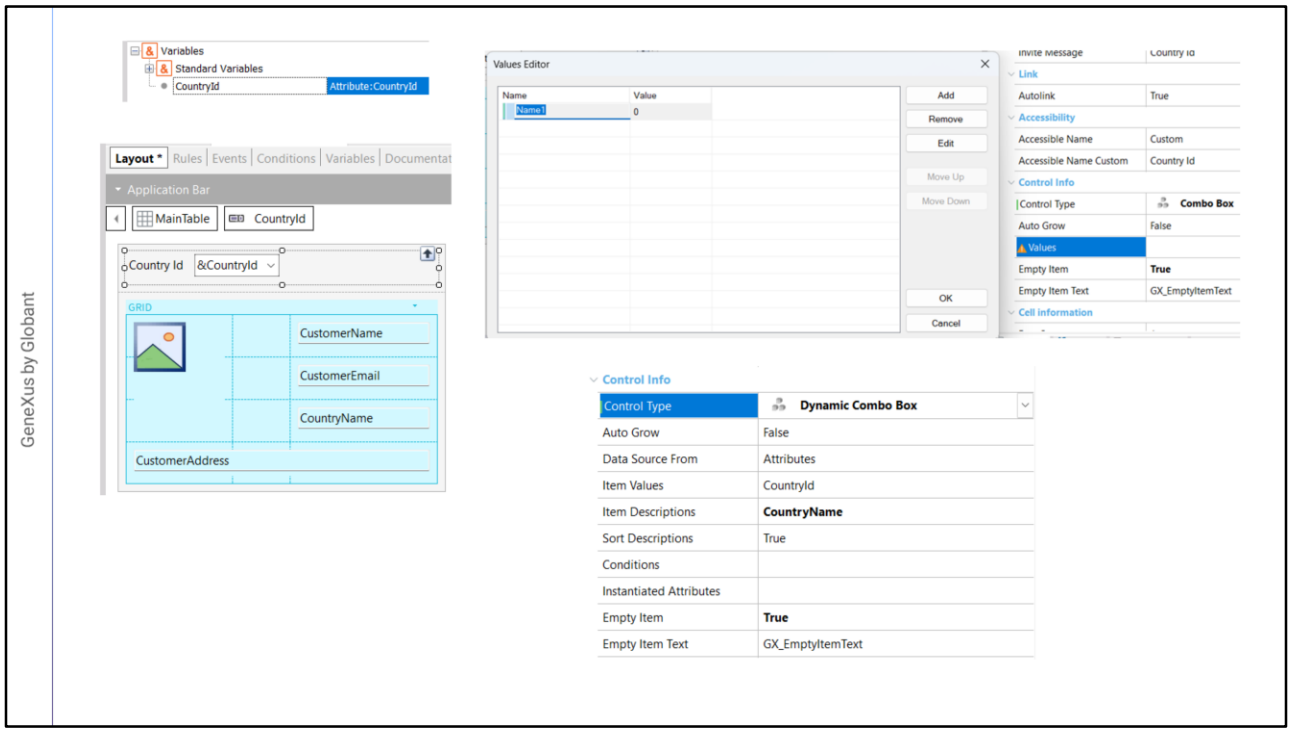
O valor 200% é para Extra-high-density (aproximadamente 320 dpis) correspondente a imagens Retina 2x para iOS ou 2x de Android. 300% corresponde a Extra-extra-high-density com cerca de 480 dpi, que são para Retina 3x de iOS e 3x de Android e 400% correspondem a dispositivos com 640 dpis. À medida que aparecerem dispositivos com valores maiores de resolução, este valor irá aumentando.



Voltemos ao desenvolvimento de nossa aplicação.

Agora vamos fazer com que o nome do país saia em negrito, então criamos uma classe AttributeBold que herda da classe Attribute, com as seguintes propriedades: font-weight em bold e a propriedade color no valor: $colors.AttributeBolded que definimos como token. Agora atribuímos a classe ao nome do país.

Aproveitamos e ajustamos todos os dados da segunda coluna com Horizontal Alignment em Left e Vertical Alignment em Middle. Para o endereço colocamos também alinhamento horizontal à esquerda e vertical em top.

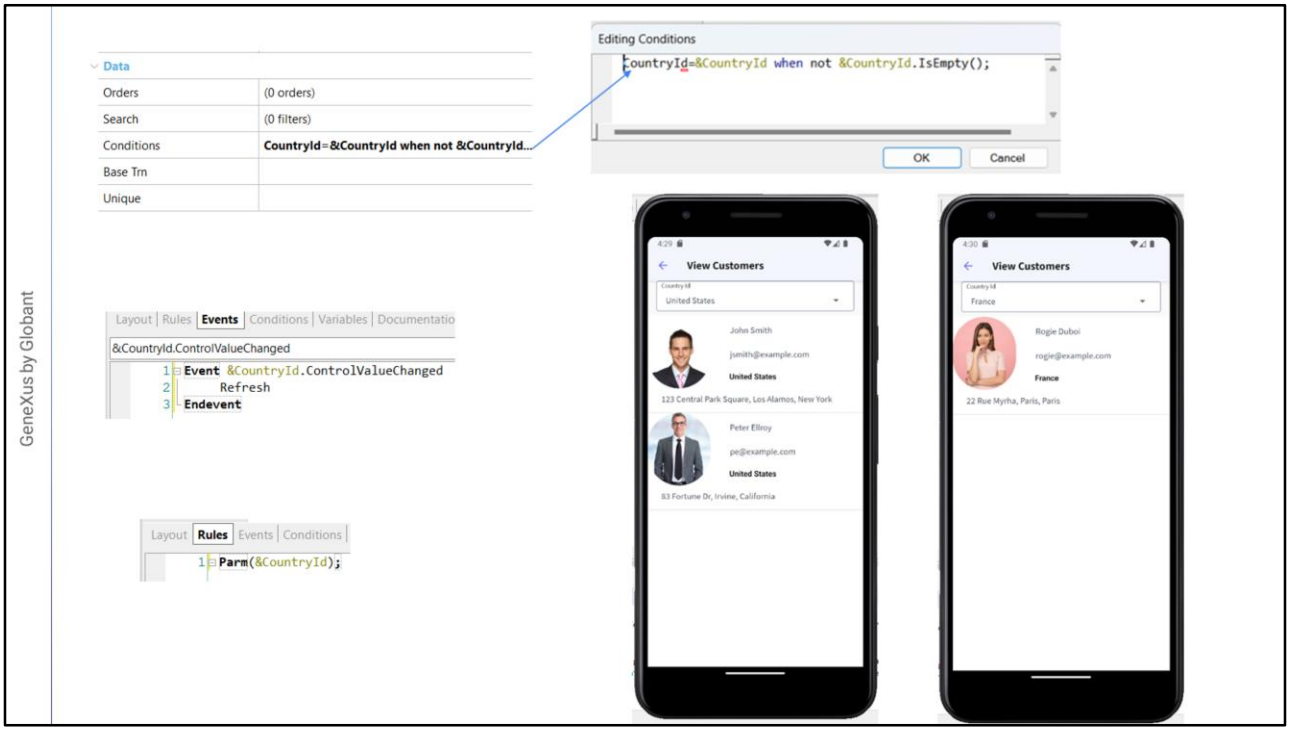


Muito bem. Para completar a funcionalidade desta tela, vamos adicionar um combo box que nos permita selecionar o país e filtrar os clientes pelo país selecionado. Para isso, primeiro criamos a variável &CountryId, que devido ao seu nome, fica automaticamente baseada no atributo CountryId e a adicionamos ao form do panel, acima do grid.

Se formos para as propriedades de &CountryId, veremos que em ControlType diz Edit, então por padrão esta variável será vista como um campo de edição. Para alterar sua aparência e comportamento, abrimos o combo desta propriedade e vemos que poderíamos atribuir a ela o tipo ComboBox, mas neste caso deveríamos inserir na propriedade Value, os valores à mão para preencher o combo.

Mas já temos a informação dos países na base de dados, então em vez desse tipo de combo escolhemos um Dynamic Combo Box e vemos que na propriedade Item Values fica atribuído automaticamente com o atributo CountryId pois a variável foi chamada assim, que será o valor retornado pelo combo ao selecionar um país.

A caixa de propriedades nos indica que devemos atribuir a propriedade Item Descriptions, que será o campo cujos valores aparecerão no combo ao abri-lo, por isso atribuímos o atributo CountryName a esta propriedade. E por último, como queremos que o combo seja iniciado sem nenhum país por padrão, configuramos a propriedade EmptyItem como True, deixando o valor do texto do item vazio pelo atribuído por padrão.



Para fazer com que o país selecionado no combo funcione como filtro no grid de clientes, vamos até as propriedades do Grid e em Conditions adicionamos: CountryId=&CountryId when not &CountryId.IsEmpty(); para que sejam mostrados apenas aqueles clientes cujo identificador de país coincida com o identificador do país selecionado no combo e, no caso de não ter selecionado nenhum, o filtro não agirá e serão mostrados todos os clientes.

Agora vamos para a aba de eventos e inserimos um evento da variável &CountryId, o ControlValueChanged e escrevemos o comando Refresh. Isso fará com que ao terminar de selecionar o combo seja atualizado o conteúdo do grid e sejam carregados os dados da base de dados novamente, filtrados pelo país selecionado. Veremos mais sobre eventos mais adiante.

Por último, vamos para as regras e adicionamos uma regra Parm com a variável &CountryId. Isto é necessário nesta arquitetura mobile pois, para minimizar as consultas ao servidor é priorizado o uso de dados em cache e para que o servidor entenda que queremos trazer novos dados adicionamos uma regra Parm com as variáveis que usamos nos filtros, para que quando alterem seu valor seja atualizada a página.

Como adicionamos filtros e eventos que envolvem o servidor, para ver as alterações temos que compilar novamente a aplicação, então fazemos F5.

Abrimos View Customers e vemos o combo na parte superior, que diz (None), para indicar que ainda não selecionamos um país. Escolhemos os Estados Unidos e vemos que o filtro funciona e só são mostrados os clientes desse país. Escolhemos outro país e verificamos que tudo corre como queríamos.

Neste vídeo vimos como trabalhar com controles de tela básicos, como melhorar sua aparência e distribuição em tela. Nos próximos vídeos continuaremos vendo outros controles de tela com funcionalidades interessantes.