Curso GeneXus Core

Considerações para aplicações nativas

Versão: GeneXus 18

Considerações para aplicações nativas

Recomendamos realizar o curso GeneXus Core tal como está montado, ou seja, com foco para Front-end Web (.Net).

No entanto, para aqueles que querem ir mapeando com o desenvolvimento Web Angular à medida que vão vendo os vídeos, deixamos aqui as considerações pertinentes para cada vídeo, mas com o aviso de que talvez isso possa confundi-lo e seja melhor que você espere terminar todo o curso tal como está.

Este material é complementar, não tem nenhum efeito sobre o exame do Curso Core nem sobre nenhuma outra parte do curso.

Conteúdo

Primeiros passos

"Criação da base de conhecimento"
Se você deseja construir também um front-end em Android ou Apple, na janela de criação selecionar o check box correspondente -além do Web (.NET) que vem por default-.

Se você não fizer isso neste momento de criação da KB, de qualquer maneira poderá fazê-lo mais adiante.

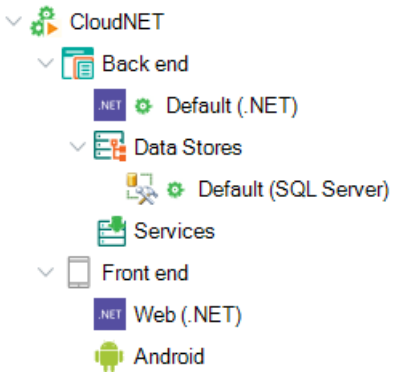
Transações

"Desenho da primeira transação"

O objeto transação possui uma parte que é front-end (ou seja, que é executada no cliente, como sua tela) e uma parte que é back-end (a que será executada no servidor, como o acesso à base de dados, por exemplo). O objeto transação só é executado integralmente (front-end e back-end) quando se trata do front-end Web (.Net, Java).

O front-end "Web (Angular)" não executa o objeto transação, mas poderá utilizar a parte back-end desse objeto. Será entendido mais adiante no curso, quando forem estudados os Business Components, que poderão funcionar como serviços expostos no servidor. Também suportará as telas do padrão Work With, embora em geral não sejam utilizados neste tipo de aplicações, pois o Backoffice normalmente é apenas web (seja .Net/Java ou Angular).

Como é a partir das transações que GeneXus cria automaticamente a base de dados da aplicação, mesmo que estejamos desenvolvendo uma aplicação para Angular, a parte do back-end será implementada em alguma das linguagens padrão (.NET, Java). É por isso que veremos separado o que é back-end do que é front-end assim:



Em outras palavras: quando se desenvolve para Android ou Apple obrigatoriamente é necessário utilizar .NET/Java para o back-end.

Por tudo o que foi dito, para carregar dados na transação Customer precisaremos ter um Backoffice .NET/Java (como o do vídeo do curso) -e é por isso que vemos o "Front end Web (.NET)- ou… usar o padrão Work With que será estudado vários vídeos mais adiante, embora não seja muito comum ser utilizado em aplicações nativas. Mas será comum ter que realizar adições/exclusões/modificações na base de dados, para o qual normalmente é utilizado o

objeto transação como Business Component (será estudado mais adiante), razão pela qual é importante aprender bem a lógica do objeto transação de qualquer maneira.

"Execução da aplicação pela primeira vez"

Vale apenas para o Front-end web em Java/.NET, como se explica no final do vídeo.

"Atributos e domínios", "Transações relacionadas", "Transações com mais de um nível", "Nomenclatura de atributos", "Definição de regras".

Tudo isso é importante entender porque diz respeito à normalização da base de dados, à integridade referencial, às regras que serão aplicadas sobre os dados a serem inseridos e aos dados em si e suas tabelas.

"Uso de padrões"

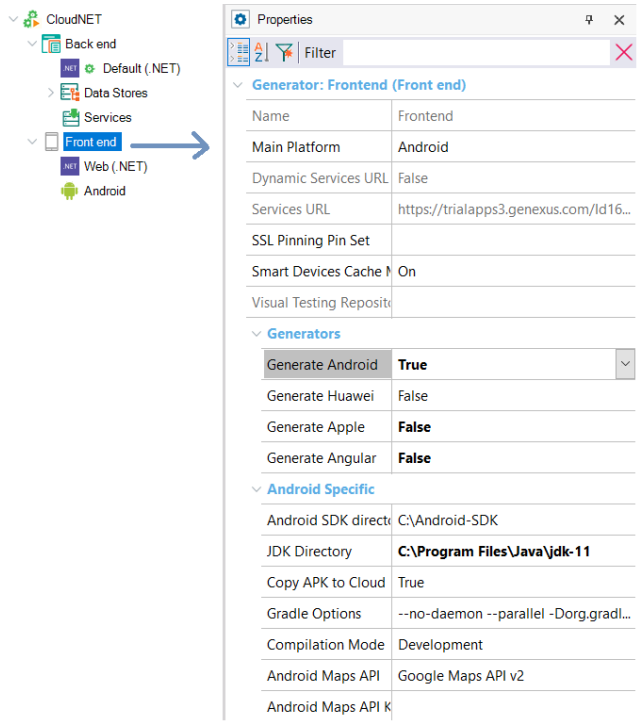
As aplicações móveis normalmente não utilizam as telas criadas pelo padrão Work With, pois normalmente não implementam Backoffice. De qualquer maneira, pode ser utilizada.

O padrão para aplicações Nativas não é o que é mostrado neste vídeo (o "Work With for Web"), mas o "Work With" geral (o que em versões anteriores de GeneXus era denominado "Work With for Smart Devices"), que vale também para aplicações Angular.

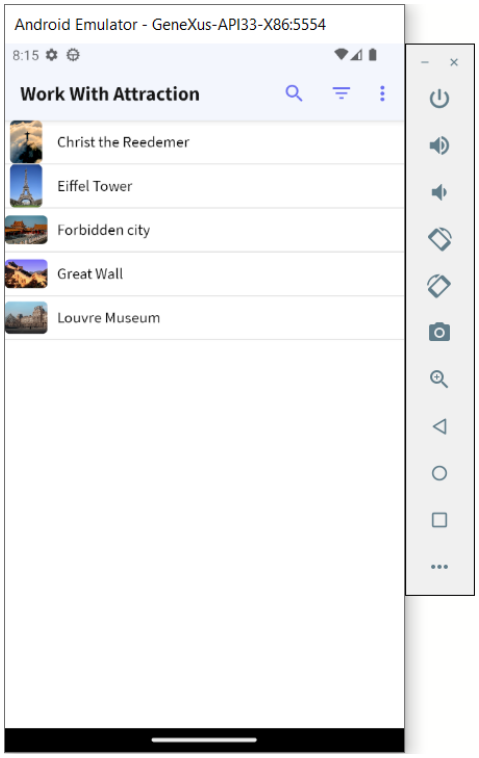
Anexamos este vídeo específico para Angular, para que o veja depois deste, porque basicamente o que é mostrado lá é idêntico ao que acontecerá para aplicações nativas.

Se você aplicar o padrão "Work With" a Attraction, para que possa executá-lo em Android ou Apple:

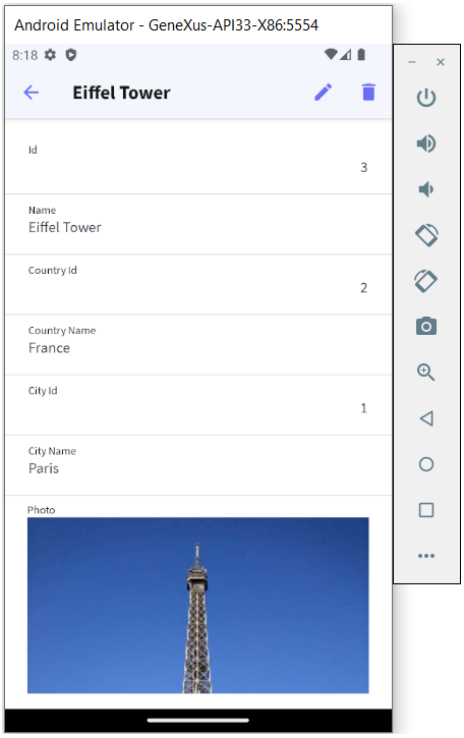
1. Certifique-se de ter ativado o front-end Android ou Apple. Aqui mostraremos o Android para poder utilizar seu emulador:



2. Deverá acessar as propriedades do objeto **WorkWithAttraction** criado pelo padrão e colocar valor True para a propriedade “Main program”. A seguir, posicionado sobre esse objeto, pressione o botão direito do mouse e selecione Run.
3. Você verá que será aberto o emulador de Android e dentro dele o panel List do Work With:



E se você fizer “tap” (“click”) sobre uma das atrações, será aberto o Detail da atração, em modo View:



Observações sobre Work With:
Enquanto para Web (.NET/Java) não existe como objeto executável, para aplicaciones móveis e Angular sí.

- Em Web (.NET/Java) tudo é configurado a partir da árvore mostrada no vídeo.
- En aplicações nativas e Web Angular tudo será programado como se fosse um objeto independente.

Isso faz com que a maneira de implementar as telas do Work With em um caso e outro sejam diferentes.
Você observará que uma vez aplicado o pattern “Work With” ao objeto transação (neste caso, Attraction) automaticamente é ativada a propriedade **Business Component**. Mais adiante você entenderá o porquê, agora só observe.

“Tabela base e estendida”, “Definição de subtipos”

Valem exatamente

“Definição de atributos como fórmulas”

Embora possamos pensar que isto não valerá para aplicações nativas , já que ali não é gerado objeto transação (mas que esta será executada em modo silencioso como Business Component) e, portanto, nada do que é mostrado em execução neste vídeo poderá ser visto

em aplicações nativas, na verdade a definição de atributos fórmula se aplica no nível da estrutura da transação, o que significa que se aplica no nível do modelo de dados. Portanto, as fórmulas serão válidas em todas as plataformas. A diferença será dada pelo momento em que serão calculadas. Mas a lógica é exatamente a apresentada aqui, em todas as plataformas de desenvolvimento. Na verdade, quando mais adiante vermos como percorrer uma tabela e listar sua informação, veremos como ali a fórmula será disparada automaticamente para cada registro sem que tenhamos que fazer nada. A definição de um atributo fórmula é uma definição lógica que tem repercussões na base de dados e em toda a KB.

"Eventos de disparo de regras em transações"

Embora possa parecer à primeira vista que isto não se aplicará a Android e Apple, na verdade as regras serão executadas também quando a transação for executada sem sua tela, ou seja, quando manipula sua informação através do Business Component.

"Índices", "Normalização de Tabelas: Um Estudo de Caso", "Relações entre atores da realidade", "Relações 1 para 1 entre atores da realidade", "Exportar e importar objetos GeneXus", "Análise do modelo de desenho de transações".

Valem exatamente

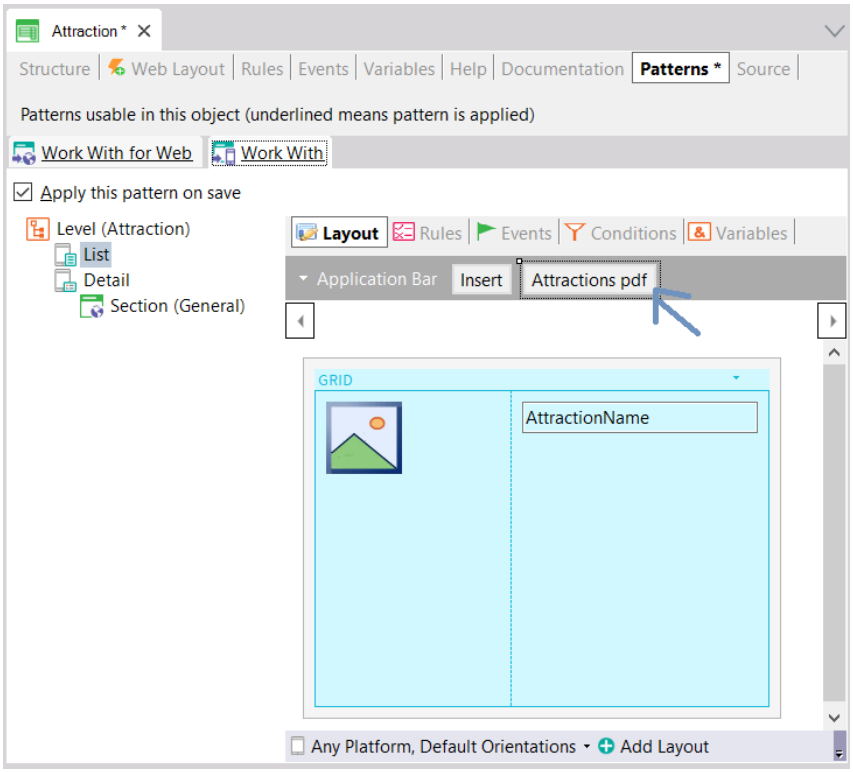
Listagens e acesso aos dados por código

"Listagens e comando For Each para consultar a base de dados"

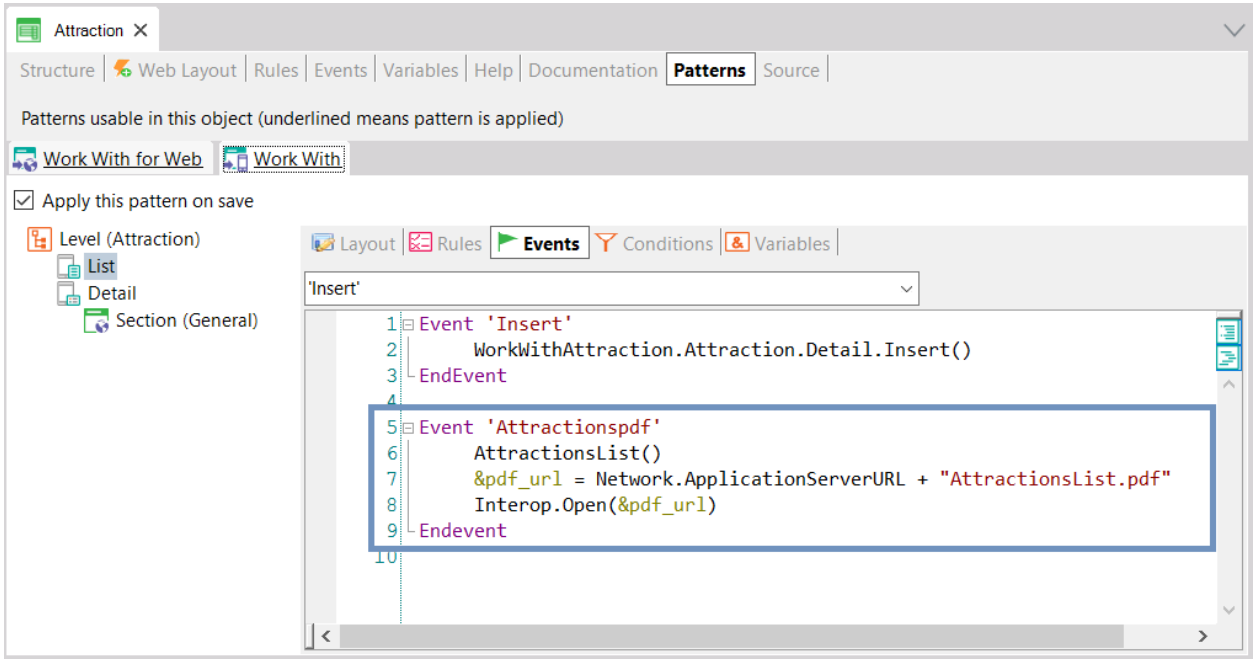
Se você seguir o vídeo como está, verá que a listagem será aberta na url do front-end .NET

Para que a lista seja executada e seja possível abrir o pdf a partir da aplicação nativa, você deve fazer o seguinte:

- Deixar propriedade "Call protocol" com o valor default, Internal.
- A partir de um nativo (pode ser o WorkWithAttraction) adicionar um botão para invocar o procedimento.



E na aba Events adicionar o que é mostrado:

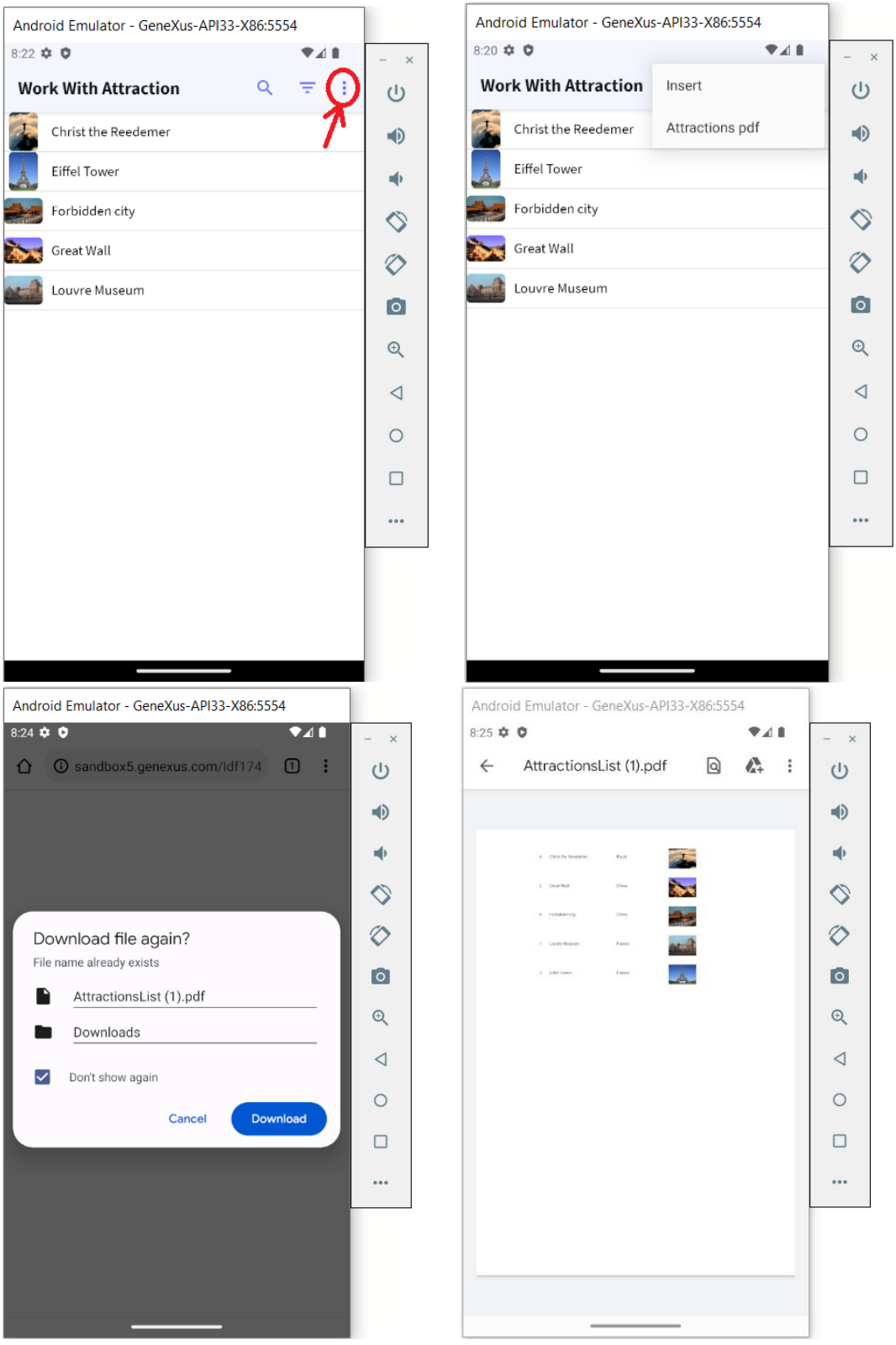


Onde:

- A variável &pdf_url deve ser do **domínio Url** (url, GeneXus:Domain)
- "AttractionsList.pdf" é o nome que foi dado ao pdf gerado pelo procedimento AttractionsList, pois ali foi especificada a regra:

Output_file(“AttractionsList.pdf”, “pdf”);

- Executar este objeto panel -WorkWithAttraction- (que deverá ter a propriedade “Main program” em True) -com botão direito / Run- e a partir dali quando for carregado o panel no emulador de Android abrir o menu e pressionar o botão para imprimir o pdf. Será aberto o browser para baixar o pdf. Uma vez baixado poderá abri-lo (no emulador será aberto no Drive).



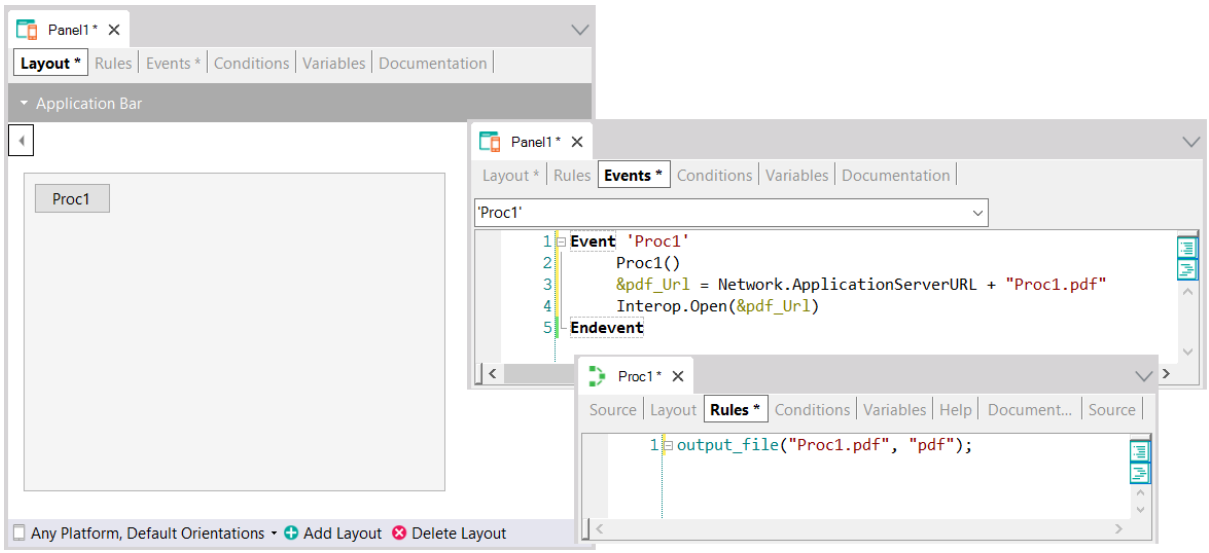
- Explicação para aqueles que estão mais avançados:

- A invocação do procedimento, AttractionsList(), solicita que o procedimento seja executado (e isso será feito no server). GeneXus terá definido a propriedade “Expose as Web Service” do procedimento para expô-lo como serviço Rest, de modo que possa ser invocado a partir do cliente. Será compreendido quando for estudada a arquitetura das aplicações nativas.
- Como o procedimento possui a regra **Output_file**, o resultado será salvo no servidor como um arquivo chamado AttractionsList.pdf (aquele indicado na regra output_file).
- Como queremos que este arquivo seja aberto pelo front-end Android, end Angular, então devemos pedir ao front-end que abra o recurso (no nosso caso o pdf), para o qual devemos primeiro indicar a url neste servidor, que é o que montamos na segunda sentença do evento (a propriedade ApplicationServerURL do objeto externo Network retornará a url onde se encontram os serviços Rest).
- Para abrir um recurso existe o método Interop.Open(...) -Interop é um objeto externo que vem no módulo GeneXus (procure-o no nó References).

“Como processar informação relacionada”, “Como processar informação agrupada”, “Fórmulas inline”

As mesmas considerações: para executar estes procedimentos deverá fazer o mesmo que no caso anterior. Como aqui o importante não é a execução nativa, mas sim o que é próprio da lógica do procedimento, sugerimos executar conforme mostrado nos vídeos (front-end Web .Net) e não no front-end da aplicação nativa.

De qualquer forma, se você também quiser fazer isso em Android/Apple você pode criar um **objeto Panel** e no Layout colocar tantos botões quantos Procedimentos necessite invocar. A variável que conterá a url (&pdf_Url) deve ser definida com a propriedade **Based on** com o domínio “Url”.
Você deve alterar a propriedade do panel **Main Program** para True, e então simplesmente pressionando **botão direito/Run** sobre o panel já irá executá-lo.

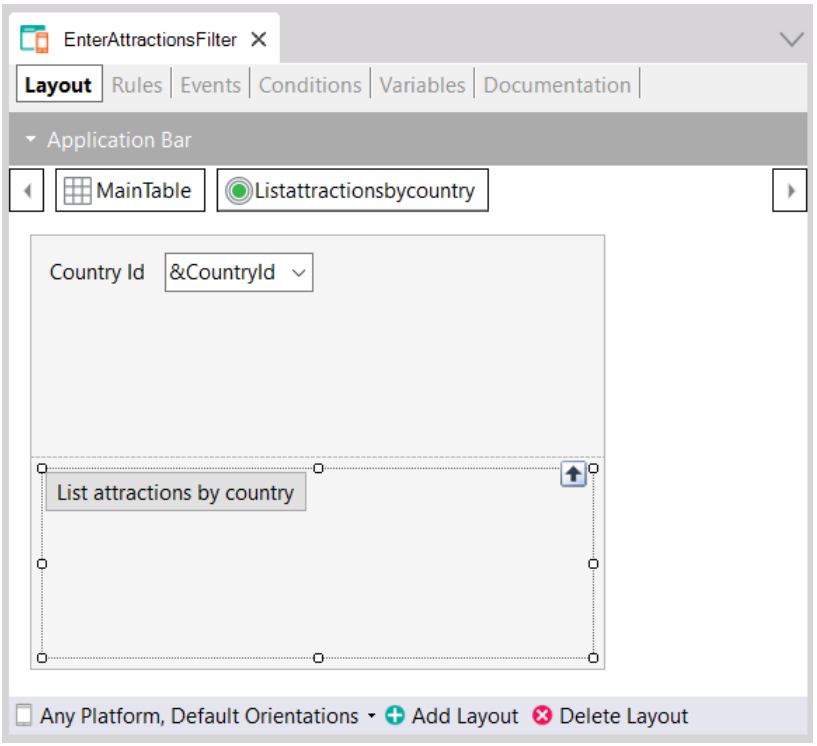


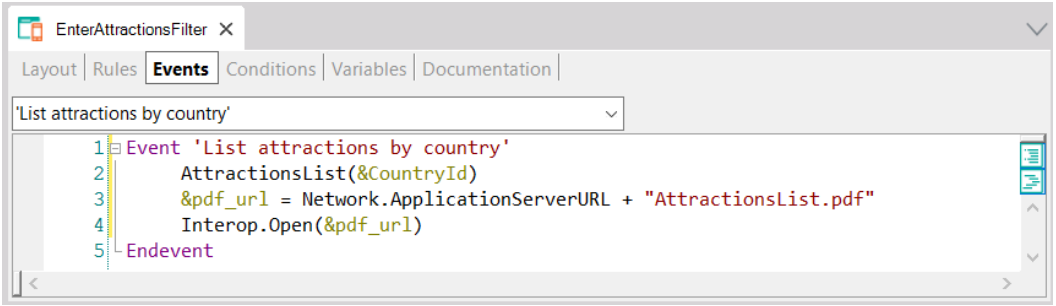
Comunicação entre objetos

"Invocações entre objetos", "Invocações entre objetos (cont.)"

O análogo ao Web Panel para aplicação nativa será o Panel. Recomendamos, para simplificar, seguir os vídeos conforme são mostrados aqui, para Web Panels e não para Panels.

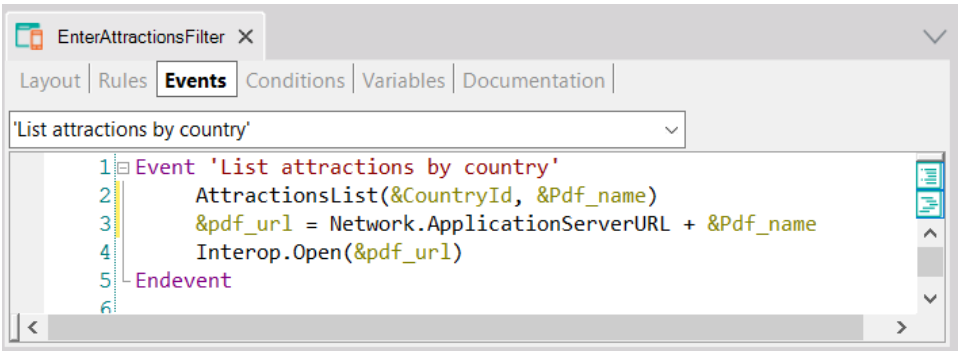
Uma opção para executar em Android/Apple o primeiro panel do primeiro vídeo, EnterAttractionsFilters:



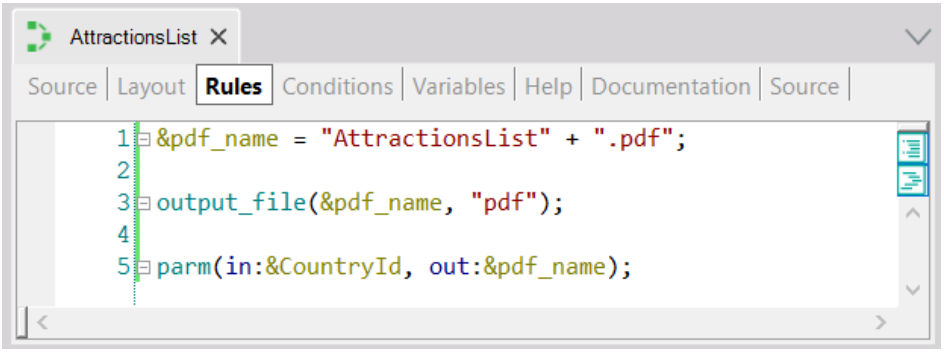


Colocando como "Main program" o panel e nos certificando de que a proc AtraçõesList tenha o valor "Expose as web service" em True, "Rest Protocol" True e "Call protocol" Internal.

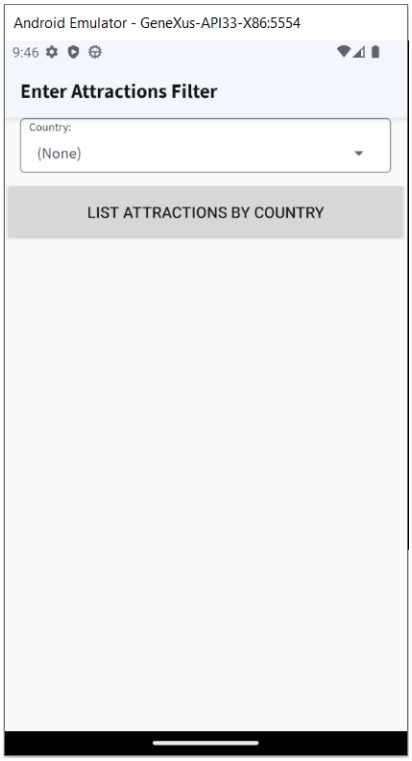
Uma maneira mais simples de implementar isso para não ter que lembrar o nome que foi dado ao pdf dentro do procedimento, é passá-lo também por parâmetro:

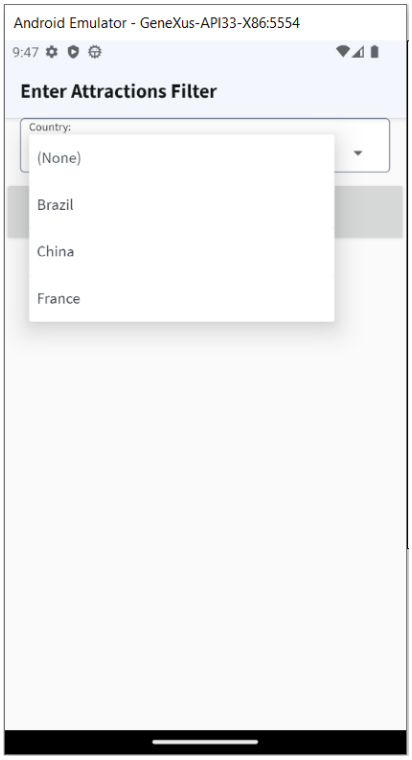


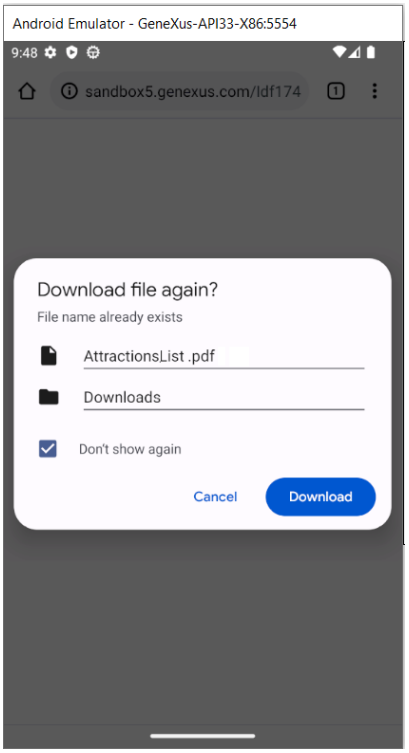
Assim, no procedimento AttractionsList:

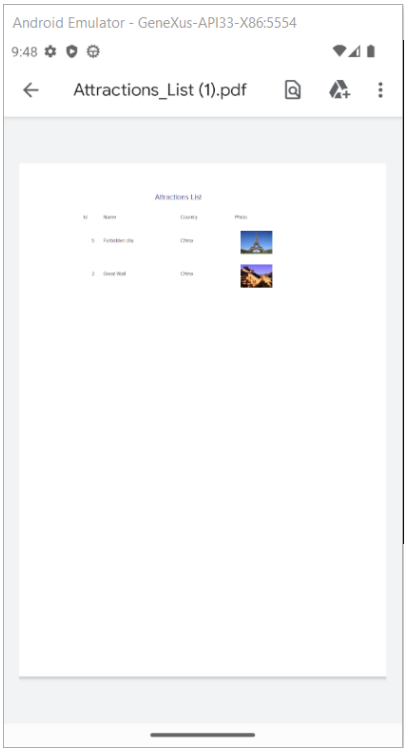


Em execução você verá:





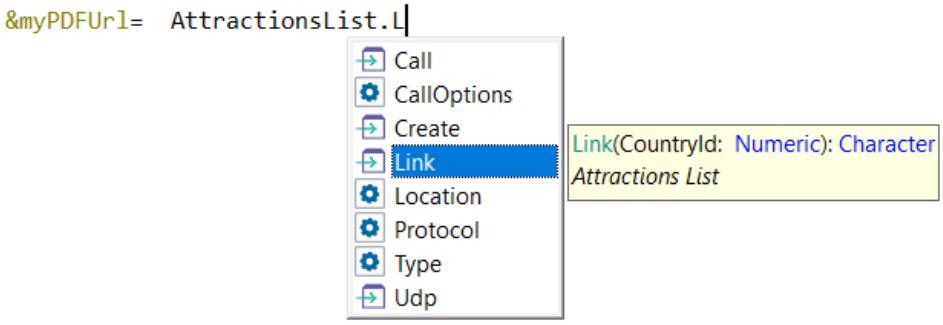




Observações extras:

Quando você chama diretamente um objeto pelo seu nome e entre parênteses a lista de parâmetros, é como se estivesse escrevendo o método Call (ou Udp se o objeto retorna um

parâmetro). Mas existem outras formas de invocação, por exemplo com o método Link. Agora não vamos nos alongar em suas semelhanças e diferenças, apenas iremos nomeá-lo.



Tipos de dados estruturados e Data Providers

"Tipos de Dados Estruturados", "Variáveis que armazenam coleções de dados em memória", "Carga de Tipos de Dados Estruturados (SDT) através de Data Providers"

Tudo isso é muito importante para todas as plataformas e é válido exatamente da mesma forma. A consideração é a mesma que vimos nas listas anteriormente (como poder visualizar uma lista pdf em front-end nativo). Sugerimos, como antes, seguir os vídeos tal como estão, em front-end Web (.NET).

Atualização da Base de Dados

Esses vídeos são fundamentais, ainda mais em aplicações nativas, onde não utilizamos a transação como objeto de UI.

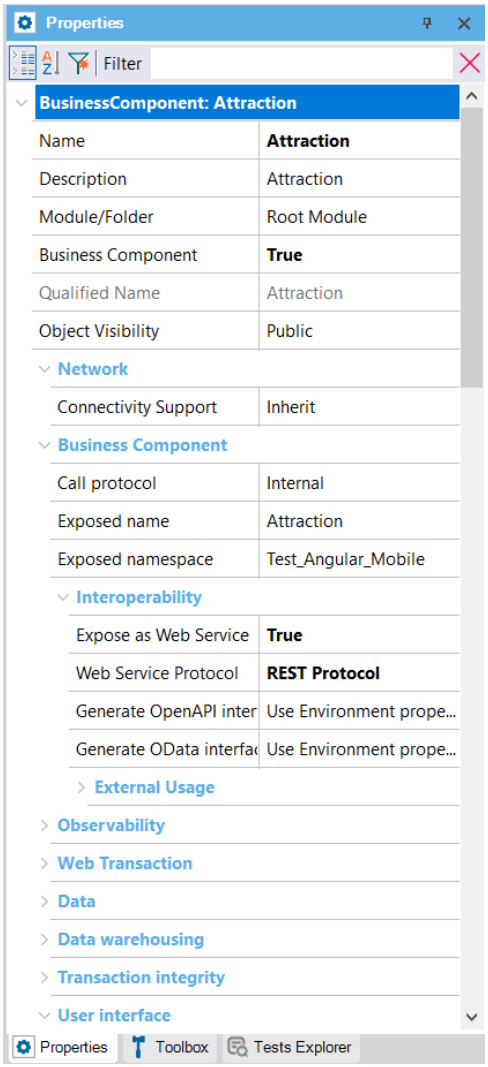
"Atualização de Business Components. Justificativa"

Quando mencionamos as regras que o pattern Work With adicionou à transação Attraction, estamos nos referindo ao pattern for Web. Não ao pattern Work With genérico, que é aquele que vale para Angular ou aplicações nativas. É que, lembre-se, em Angular e em aplicações nativas não será executado o objeto transação. Na verdade, o que será utilizado é o que está começando a ser apresentado neste vídeo, que é o Business Component que é criado a partir da transação.

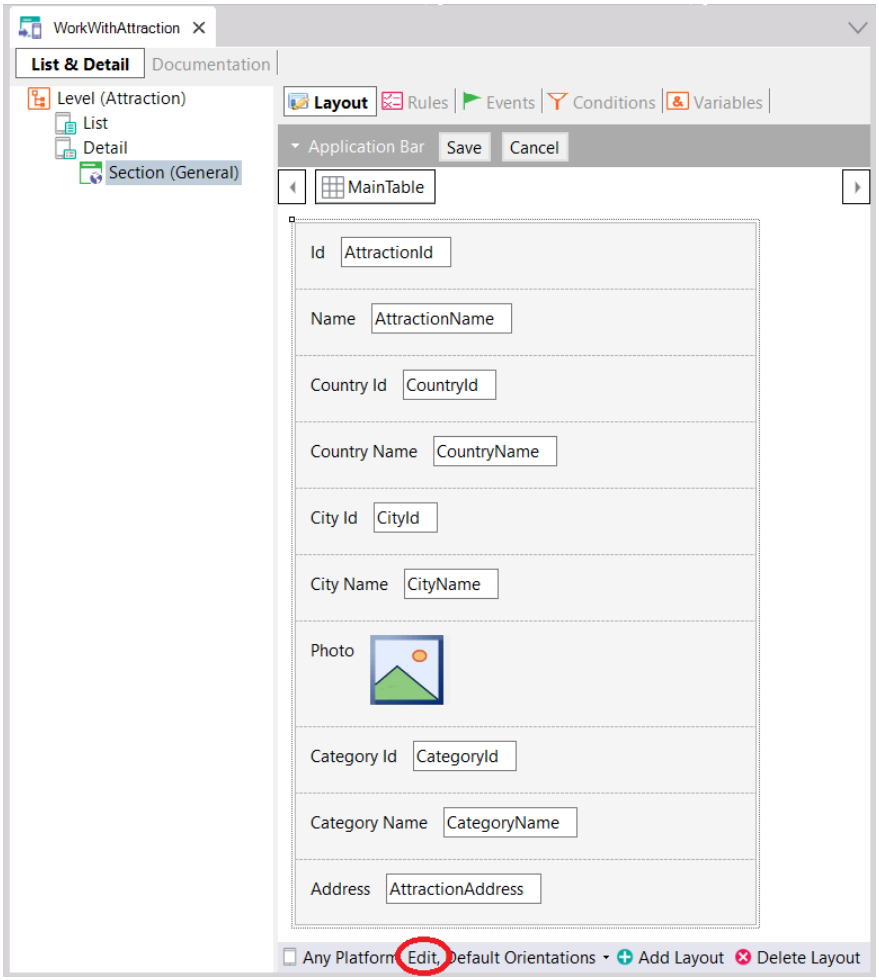
"Atualização com Business Component"

Primeiro assegure-se de entender o vídeo tal como está, e somente depois leia o que segue.

Se quando estudamos o padrão Work With você criou o pattern para Android/Apple, então terá observado que automaticamente GeneXus ativou a propriedade Business Component da transação e as propriedades de interoperabilidade mostradas na imagem:



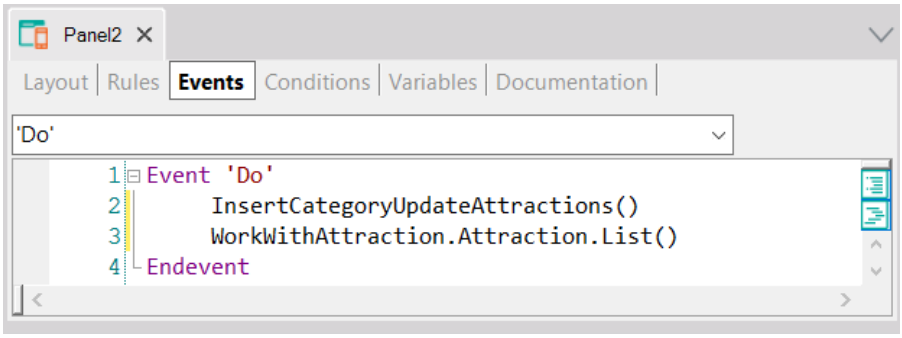
Isso aconteceu porque para poder inserir atrações, bem como modificar ou eliminar através das telas do Work With, em nas aplicações nativas não é chamada a transação Attraction e sim a tela correspondente à seção General, layout Edit.



E esses “atributos” que vemos na tela, na verdade não são eles. São os elementos do Business Component. Ao pressionar o botão **Save**, internamente o que será feito é chamar o serviço Rest correspondente ao Business Component, publicado no server, para que realize a operação correspondente sobre a base de dados. Resumindo, a partir de um evento do cliente, será chamado um serviço Rest do server, que é o que efetivamente tentará a atualização na base de dados através do business component, e caso tenha sucesso, realizará o Commit. Este Commit o serviço Rest tenta realizar automaticamente.

Diferente é o caso se quiser utilizar o BC a partir de um procedimento, pois o procedimento é executado no back-end e ali tudo é idêntico ao que foi visto no vídeo.

Para executar o procedimento InsertCategoryUpdateAttraction, criar um Panel com um botão Do no layout e no evento associado escrever:



Outra vez: o procedimento com “Call Protocol” Internal. Observar que ao invocar o procedimento a partir do evento, automaticamente GeneXus ativa as propriedades “Expose as Web Service” e “REST protocol” para o procedimento. Aproveitamos e invocamos a List do Work With para ver como aparece listada a nova atração.

Quando o Business Component é utilizado dentro de um procedimento, ali não é necessário executá-lo como serviço Rest (o business component, não a proc), pois o procedimento será executado no server, e é lá que utilizará o Business Component. É por isso que ali podem ser encadeados os commits como o desenvolvedor desejar, tornando-os explícitos. Tudo como visto no vídeo.

A parte final do vídeo, onde é falado que podem ser utilizados os BCs não só em Procs mas também em eventos de Web panels, é aquela que não será tão idêntica em aplicações nativas, onde terá limitações.

“Atualização com Business Components. Um exemplo”

Para realizar isso em aplicação nativa, em vez do Web Panel “CategoriesAndAttractions”, criar o Panel análogo (e torná-lo main para poder executá-lo com Run).

Para a primeira parte, quando é invocada a partir do evento 'Do' do botão do panel a proc, será tudo idêntico (embora observar que a proc automaticamente estará exposta como serviço Rest).

MAS: quando no meio do vídeo for copiado o código do Source do procedimento para o evento 'Do' do Web panel, as coisas não funcionarão da mesma forma aqui. Acontece que ao fazer isso no Web Panel, embora seja um evento no cliente, GeneXus internamente move esse código para o servidor, e o executa lá.

Para **Panels** isto não será igual. No Panel esse código será executado no cliente, não no servidor. E, logicamente, no cliente não há acesso à base de dados. Sim, funcionará atribuir valor aos elementos do BC e até a operação de Insert ou Update, pois para isso é invocado o BC como serviço Rest. Mas não funcionará a consulta do resultado com If, nem mesmo If &category.Success(), e muito menos o Commit ou Rollback.

Para realizar o Commit ou um Rollback não haverá uma alternativa senão invocar um procedimento que o realize (o que sempre será executado do lado do Server). Então acaba sendo conveniente fazer tudo nesse procedimento.

Também não poderão ser utilizados em eventos do cliente (executados no front-end) nem comandos for eachs, nem fórmulas que acessam a base de dados, como por exemplo find(CategoryId, CategoryName = “Monument”).

Portanto, essas implementações em um Panel deverão ser feitas invocando procs nos eventos (que, por executarem no server, têm à disposição todos os comandos de acesso à base de dados).

Você entenderá melhor tudo isto assistindo três vídeos mais adiante, na seção de Arquitetura, aquele chamado “Aplicações GeneXus e sua arquitetura”.

“Preenchimento de dados com Business Component e Data Provider”

Não funcionará o método Insert para coleção de BCs no nível de evento do cliente, portanto deverá incluir todo o código do evento em um procedimento.

“Preenchimento automático de dados”

O Data Provider que GeneXus cria para preencher com dados a tabela Category (e o mesmo para Attraction) será executado no back-end, ou seja, será gerado em .NET no nosso caso. Portanto, é independente do front-end. O front-end .NET no vídeo está sendo utilizado apenas para visualizar os dados inseridos.

“Atualização com comandos específicos de procedimentos. Introdução”

É válido como está, com a consideração de que estes comandos que acessam à base de dados, logicamente, só podem ser executados do lado do Server, ou seja, no back-end.

Arquitetura

“Aplicações GeneXus e sua arquitetura”

É importante para entender as diferenças na arquitetura de uma aplicação com front-end que utiliza Web panels (.NET ou Java) versus uma que utiliza Panels (Angular ou nativas).

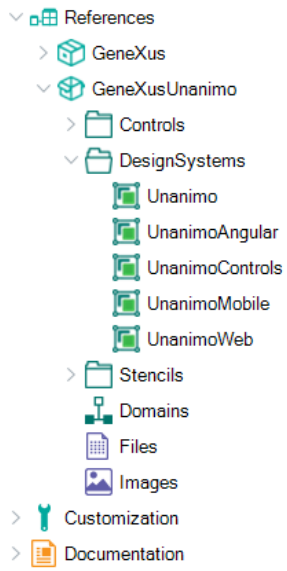
Telas Web com foco em Back-office

Todos os vídeos aqui incluídos são para front-end Web (.Net ou Java). Embora o objeto Web Panel não seja utilizado para front-end Angular ou nativo, parte da lógica será válida, com algumas diferenças, para os Panels, e aqui é explicado do zero, por isso sugerimos visualizar estes vídeos com a mesma atenção que os anteriores.
Depois, no desenvolvimento de aplicações nativas você estudará as particularidades dos Panels.

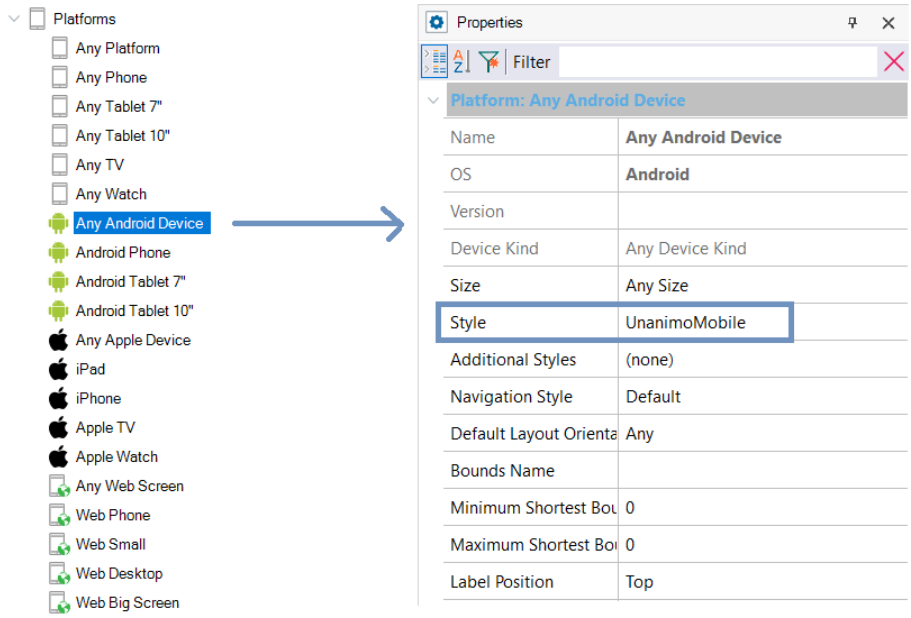
Desenho e modelagem de telas

UX Design. Introdução.
O que mostramos no vídeo é para font-end Web (.Net/Java), mas conceitualmente vale praticamente o mesmo para qualquer outro. Algumas diferenças significativas:

- Para front-end Web (.Net/Java), é definido o Design System Object que será aplicado por padrão a todas as telas através de uma propriedade da versão, chamada Default Style. Por padrão, quando é criada uma KB, é criado um DSO com o mesmo nome, e este DSO importará por padrão tudo o que venha do predefinido chamado UnanimoWeb:



- Para front-end nativo, a forma de indicar qual será o DSO que irá comandar o desenho das telas é nas propriedades da plataforma. Aqui é possível ver que o valor default é o DSO UnanimoAngular, que vem predefinido no módulo GeneXusUnanimo.



Recomendamos que você siga este vídeo como está. Você pode aproveitar e analisar o DSO UnanimoMobile e testar com o WorkWithAttractions que aplicou para a transação Attraction para Android.

Mais adiante (não agora) você pode estar interessado em assistir a este webinar onde explicamos os primeiros passos para desenhar do zero algumas telas para uma aplicação customer-facing Angular, sem depender de nenhum DSO preexistente, mas criando um do zero. Lá também é mencionado como seria para aplicações nativas.

Todos os demais vídeos do curso Core são válidos como estão.