Script Demo

GeneXus™ 16

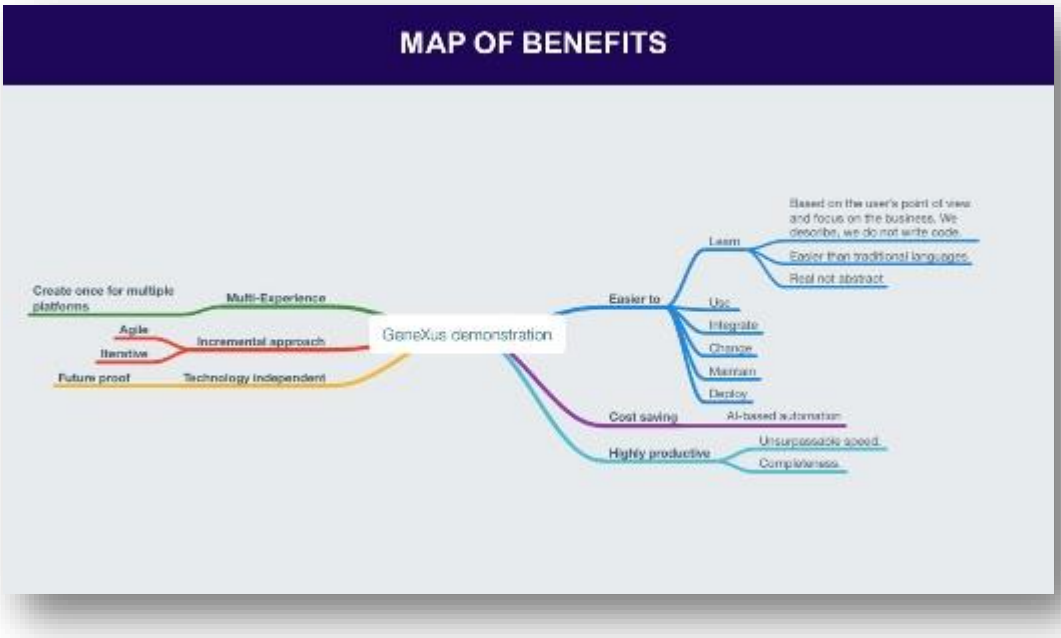
Guia passo a passo para a apresentação de GeneXus™ 16.

TABLE OF CONTENTS

OBJETIVOS

O objetivo deste vídeo é descrever os passos que devem ser seguidos para fazer uma demonstração GeneXus. O roteiro descreve todos os pontos importantes que devem ser transmitidos em sua apresentação.

O objetivo da demonstração é apresentar ao cliente os principais benefícios obtidos ao adquirir GeneXus. Esses benefícios têm suporte nos pilares de GeneXus e iremos demonstrá-los com fatos durante a apresentação, como por exemplo, a necessidade de escrever apenas uma vez para múltiplas plataformas, a abordagem incremental em relação à solução final, a independência da tecnologia, a facilidade de aprender, de usar, de integrar, de modificar, de manter e alta produtividade.



GENEXUS CHALLENGE

Para envolver os espectadores, proporemos a demonstração como um desafio, convidando-os a participar do *GeneXus Challenge*. Para isso, explicaremos aos participantes que, em nossa demonstração, criaremos um sistema para a empresa GeneXus que lhes permitirá gerenciar os pedidos de seus produtos de merchandising e seus clientes. *A seleção da empresa GeneXus é a título de exemplo e, de preferência, recomendamos adaptar esse mesmo exemplo ao seu cliente em potencial.*

O desafio é que os espectadores terão que estimar quanto tempo seria necessário para construir do zero um sistema com esses recursos para a plataforma web e também uma aplicação para

dispositivos móveis. Assim que tivermos essa estimativa primária, seria bom iniciar um cronômetro e começar a demonstração.

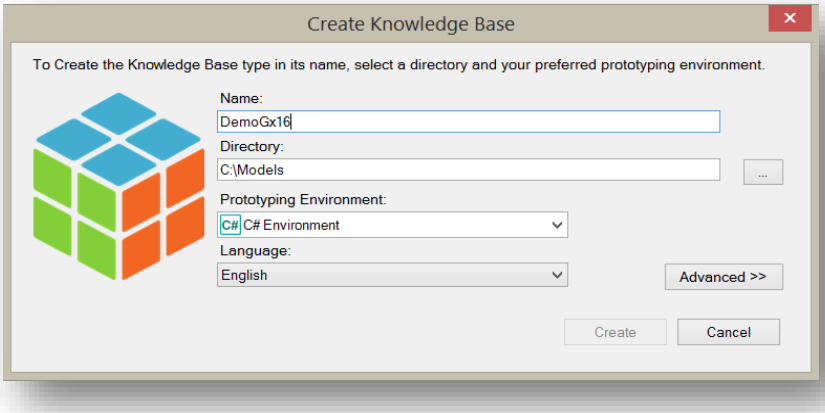
DEMO

1. NOVA BASE DE CONHECIMENTO

Começaremos criando uma *Knowledge Base*:

Para isso vamos à opção *File/New/Knowledge Base.*

E Propomos um nome para esta Base de Conhecimento, por exemplo DemoGX16.

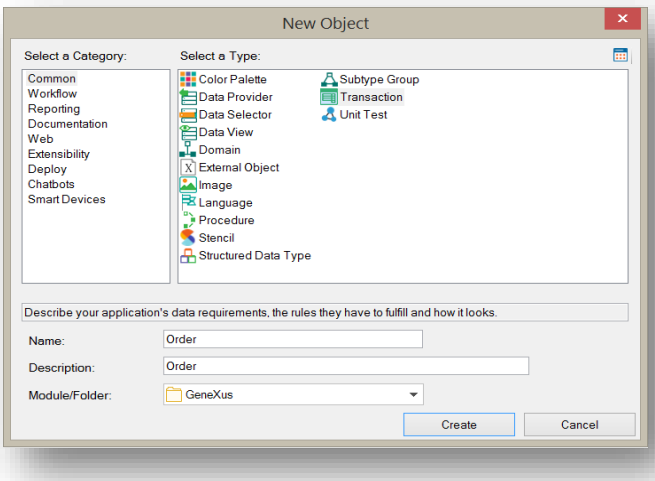


Enquanto é criada a KB é importante mencionar que os projetos GeneXus são chamados de "Knowledge Base", pois no GeneXus as aplicações são descritas em vez de serem programadas. São projetadas com base na realidade do negócio que estamos modelando e, portanto, o projeto acaba sendo uma base do conhecimento do negócio.

2. CRIAR TRANSAÇÃO ORDER

Vamos começar agora a descrever e modelar a realidade.

Então vamos para a opção ***File / New / Object*** *[ou pressionamos CTRL + N].*



E vemos que no GeneXus existem vários tipos de objetos que, juntos, nos permitem modelar qualquer aplicação.

Em particular, para representar as entidades da nossa realidade, utilizamos o objeto Transaction.

Das Transações definidas, o modelo de dados da aplicação é inferido na terceira forma normal e GeneXus também utiliza este objeto para gerar o programa da aplicação que permitirá ao usuário final inserir, apagar e atualizar registros da base de dados, de forma interativa.

Criamos então a Transação Order.

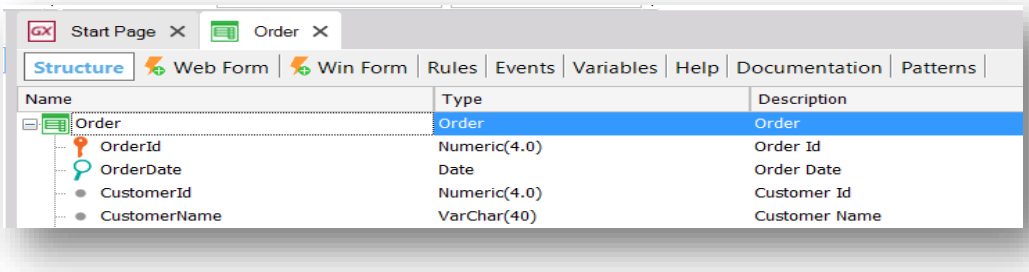
O que nos interessará registrar em um pedido de compra?

- Certamente um número que o identifica.
- A data.
- Os dados do Cliente.
- O conjunto de produtos com seu preço, quantidade e o total da linha.
- E o total do pedido.

Vamos então começar a definir esses dados e, para isto, utilizaremos uma nomenclatura na qual cada campo, ou atributo, é nomeado com um prefixo da transação à qual pertence.
Para isto, teremos que indicar o atributo, por exemplo, OrderId. E então, pressionar a tecla ".", para que já pré-carregue o nome da transação no momento de nomear cada atributo.

Damos Enter para inserir o próximo atributo que, neste caso, será a data do pedido, por isso, colocamos OrderDate.
Agora, definimos CustomerId e CustomerName para registrar o cliente do pedido e pressionamos tab para indicar o tipo de dados correspondente.

Até aqui inserimos os campos do cabeçalho do pedido de compra. Agora incluiremos as linhas.



Para isso, pressionamos Ctrl + a seta direita do teclado e inserimos os campos do segundo nível. Estes atributos do segundo nível são aqueles que correspondem aos produtos.

Então inserimos o Id, pressionamos Enter, inserimos o nome, tab para escolher o tipo de dados, Enter novamente, o preço.

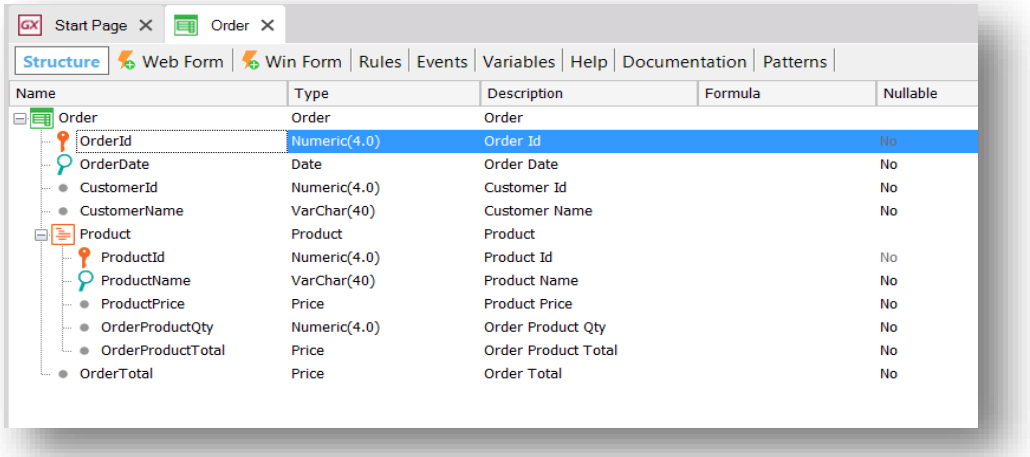
E chegando a este ponto vamos definir o domínio Price.
Os domínios nos permitem definir tipos de dados de forma centralizada para que eles possam ser reutilizados. Com isso, ganhamos em nível de abstração e em facilidade para nos adaptarmos à mudança. Por exemplo: lembre-se que no ano 2000 as datas passaram a de ter o ano modelado com 4 dígitos em vez de 2.

Então, vamos criar o domínio Price como um numérico de 6 dígitos com duas casas decimais.

Escrevemos o nome Price, sinal de "=" e o tipo de dados que serão armazenados. No nosso caso, numérico, parêntese e os 6 dígitos com 2 casas decimais.

Agora, inserimos a quantidade de unidades do produto, do tipo numérico e o total da linha baseado no domínio Price.

Depois que os atributos do segundo nível foram inseridos, pressione Ctrl e a seta para a esquerda para retornar ao nível do cabeçalho e inserir os dados do rodapé do pedido de compra, também do tipo Price.



Agora nos posicionamos no atributo OrderId e, em suas propriedades, indicamos que queremos que seja autonumerado. Então, indicamos o valor True na propriedade Autonumber

Salvamos a Transação.

Bom. Acabamos de criar a estrutura de uma Transação composta de dois níveis:

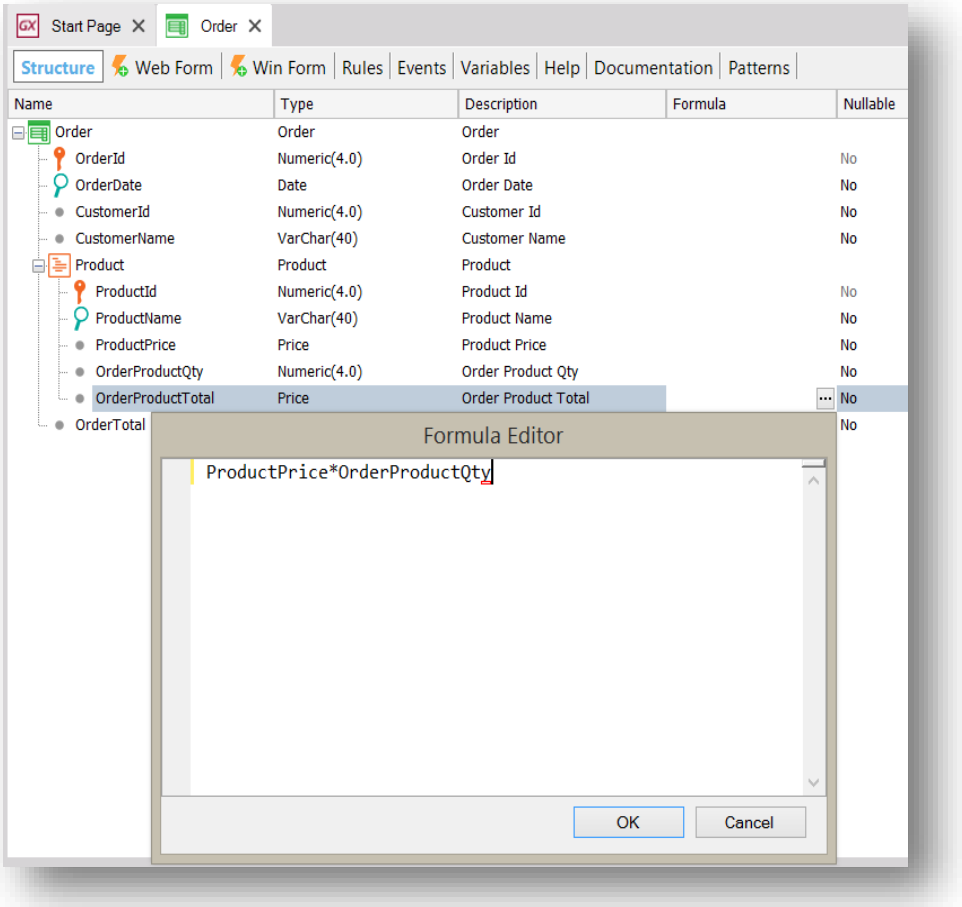
1. Um nível básico (Order), onde é especificada toda a informação necessária para o cabeçalho do pedido de compra.

2. E um nível aninhado correspondente aos produtos, onde se especifica a informação dos itens.

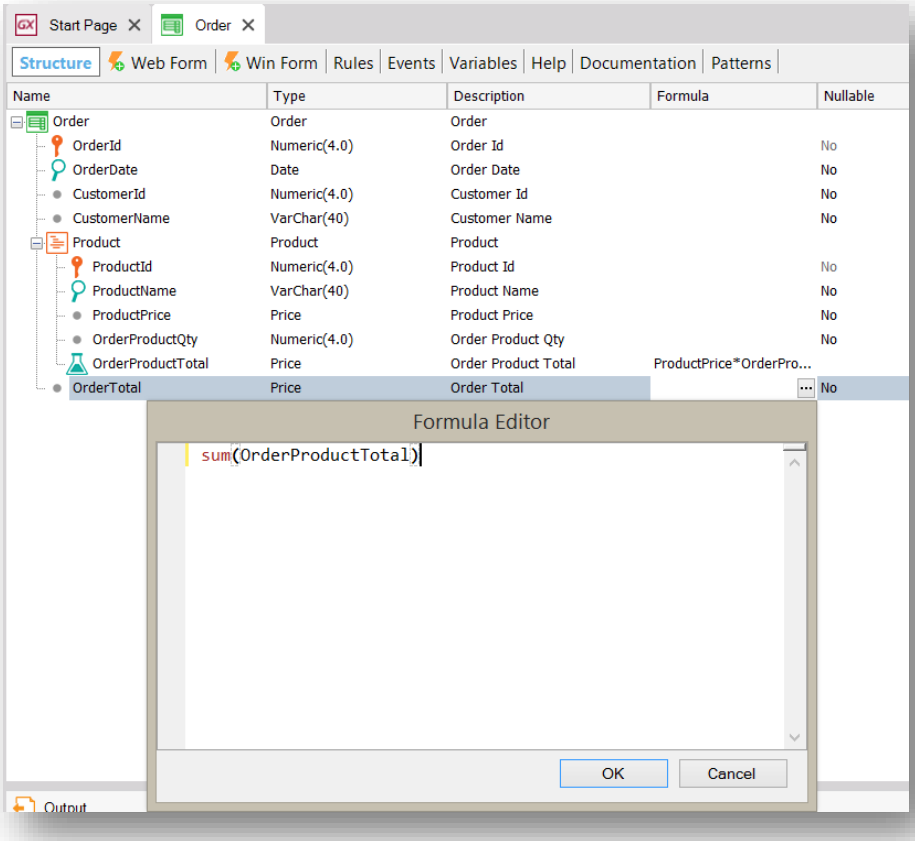
Observemos que faz sentido que determinados campos sejam calculados automaticamente, como o total da linha e o total do pedido.

Para isto indicaremos a GeneXus a fórmula para o cálculo de cada um deles.

Então, nos posicionamos na coluna fórmula do atributo que corresponde ao total da linha e indicamos que seu cálculo corresponde ao preço do produto pela quantidade de unidades.



Da mesma forma, no atributo que representa o total do pedido, indicamos que seu cálculo corresponde à soma dos totais das linhas.

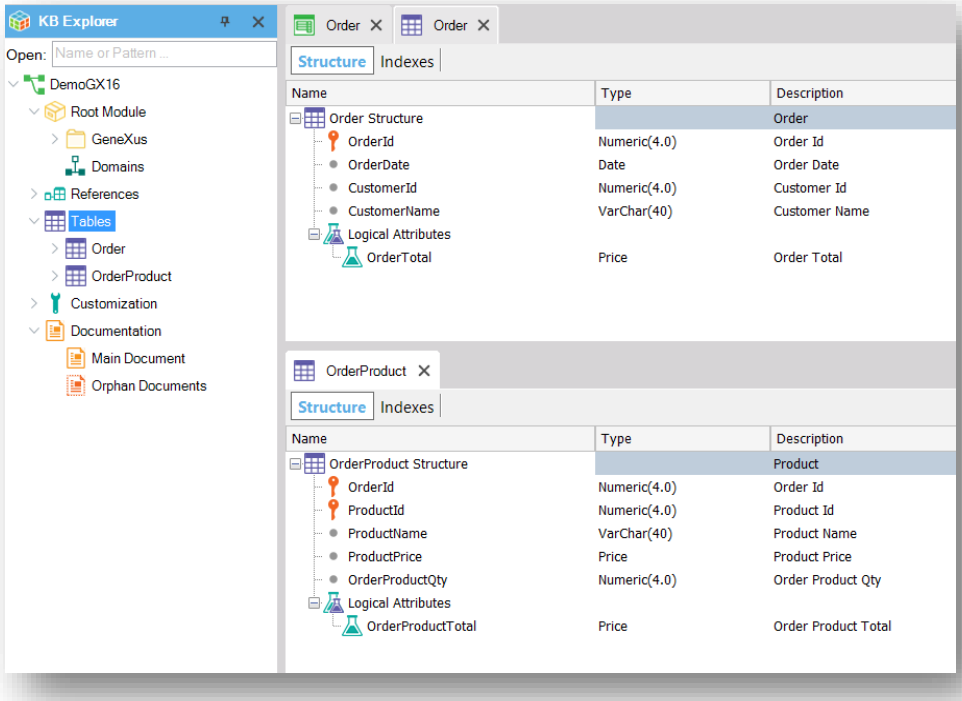


3. VER MODELO DE DADOS

GeneXus irá gerar automaticamente o melhor modelo de dados que representa as Transações definidas pelo Analista GeneXus.

Sempre que for clicado no botão Save ou Run, GeneXus irá inferir o modelo de dados otimizado (3ª forma normal sem redundâncias) que suporte as entidades do usuário final representadas pelos objetos Transação. Com base neste modelo, GeneXus irá gerar uma base de dados física para o DBMS definido em seu Environment.

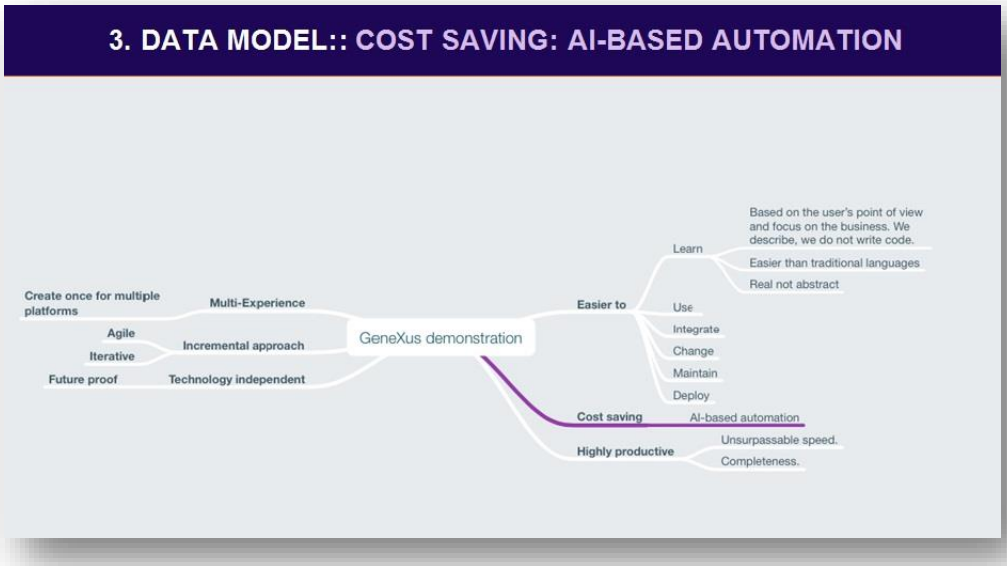
Se quisermos ver o modelo de dados criado pelo GeneXus, vamos ao menu superior View e escolhemos Tables.



Podemos ver que GeneXus criou automaticamente um modelo de dados normalizado, criando duas tabelas para suportar o objeto Transação Order:

- Order, que corresponde ao cabeçalho do pedido
- E OrderProduct que corresponde às linhas do pedido

Isso representa um benefício em tempo e em custos graças à automação através da inteligência artificial.

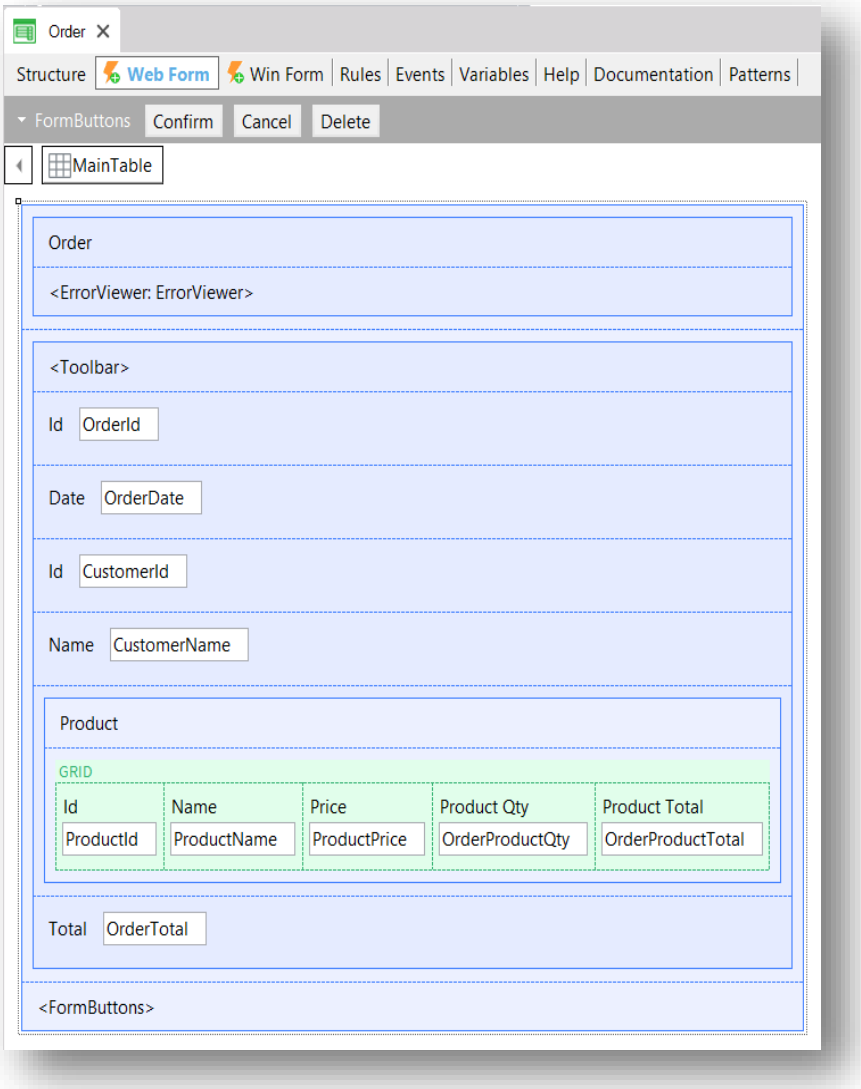


4. VER WEB FORM

Antes de executar a aplicação, vejamos o formulário web que foi criado automaticamente por GeneXus a partir da estrutura da transação. O referido web form será aquele que o usuário final visualizará quando executar a aplicação em uma plataforma web e através dele poderá inserir, modificar e eliminar registros como veremos em alguns minutos.

Estes formulários podem ser personalizados pelo analista de negócios a qualquer momento.

Os web forms são criados em um editor web abstrato que nos permitirá criar Responsive Web Applications.



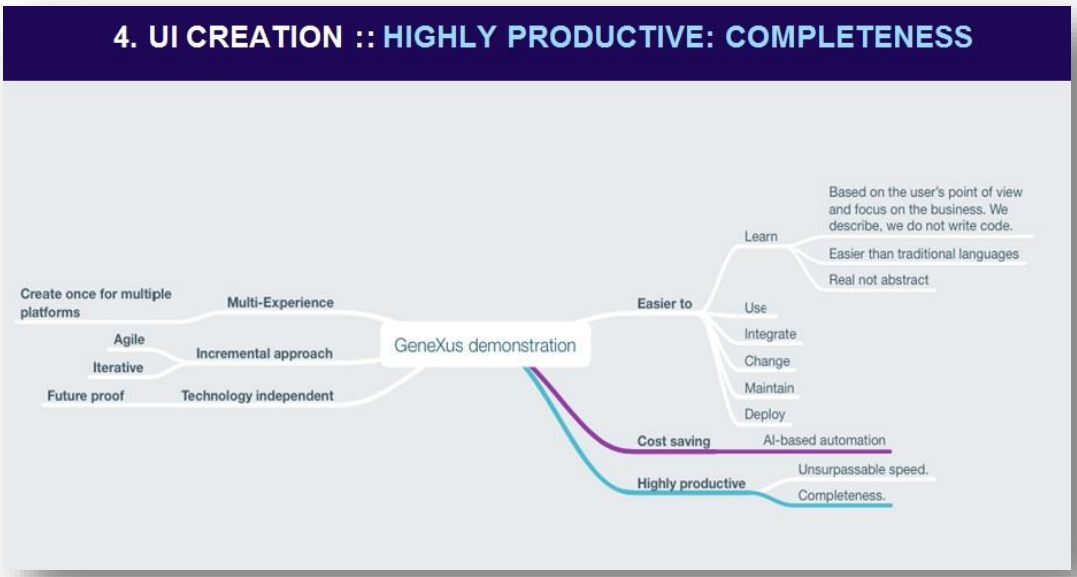
O Analista GeneXus não precisa programar nenhuma destas ações porque estão implícitas na lógica da Transação. O GeneXus irá gerar automaticamente o código nativo correspondente à plataforma escolhida.

Levar em conta que ao definir Transações GeneXus, você está:

Explicitamente: descrevendo a interface do usuário para a apresentação e captura de dados.

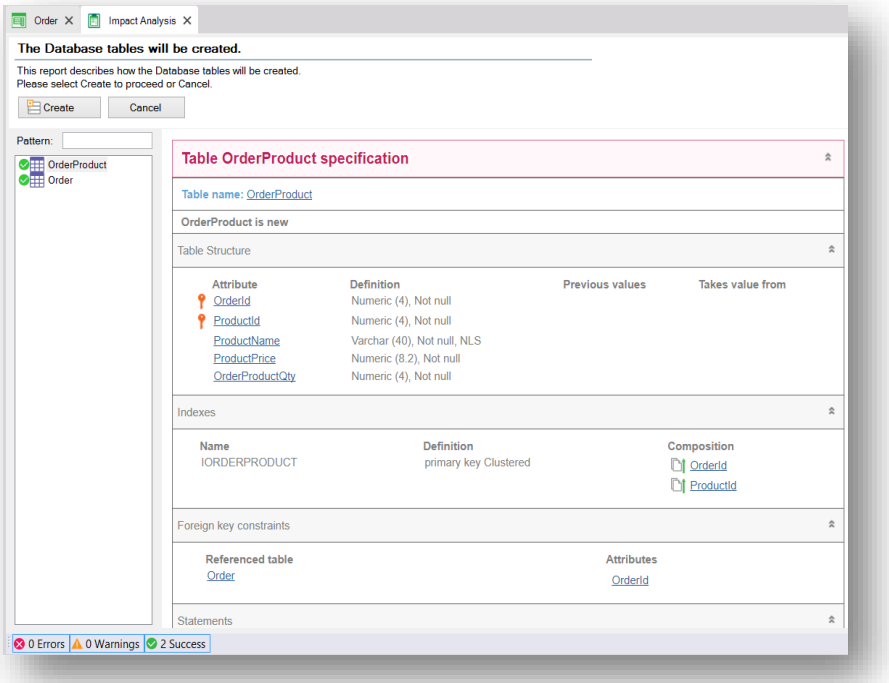
Implicitamente: projetando o modelo de dados da aplicação (tabelas, índices, etc.)

O fato de GeneXus ser responsável por gerar tudo isso automaticamente contribui para sua alta produtividade.



5. EJECUTAR LA APLICACIÓN POR PRIMERA VEZ

Para executar a aplicação, pressionamos F5 ou RUN a partir da Toolbar .



Essa tela que aparece aqui é o relatório de criação da base de dados. GeneXus nos declara quais scripts serão executados na Base de Dados.

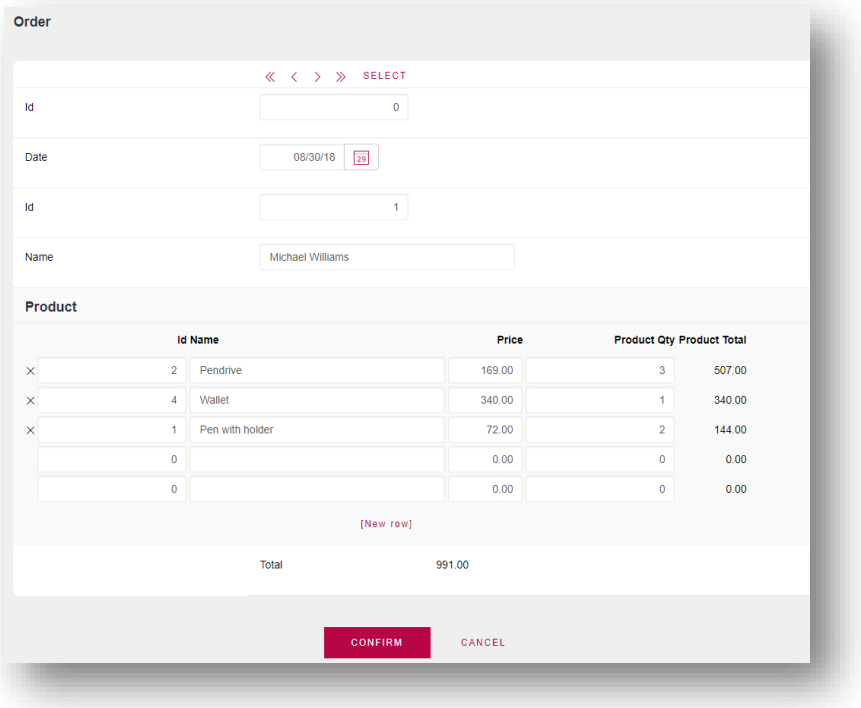
Clicamos “Create”.

GeneXus gerará os programas necessários para executar a aplicação web, bem como todos os programas para criar e manter a base de dados normalizada.

Quando este processo terminar, um navegador web será aberto e um menu será carregado com os objetos criados no GeneXus.

O Developer Menu é um arquivo XML que inclui todos os objetos executáveis. É um menu auxiliar para que o Analista GeneXus possa testar a aplicação.

Em seguida, clicamos na opção Order para inserir alguns registros de pedidos de compra.



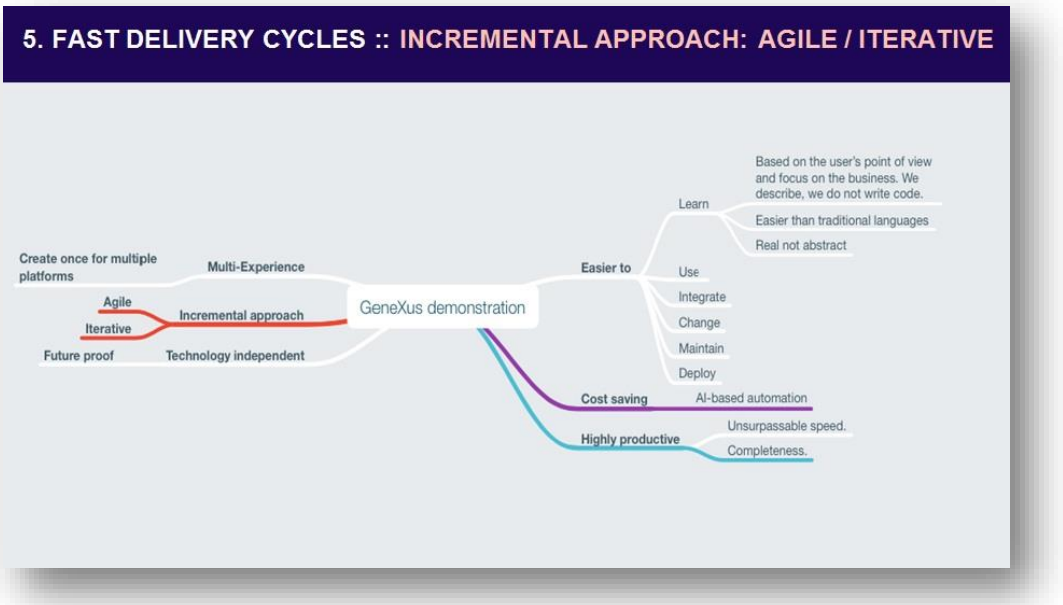
Indicamos a data e inserimos alguns produtos.

Uma vez concluída a entrada de todos os dados, pressionamos Confirm.

GeneXus gerou os programas com as mais recentes tecnologias em um ambiente web: Full Ajax, HTML5, CSS3, Bootstrap, Responsive web design, etc. Fica claro, então, uma das razões pelas quais GeneXus é altamente produtivo.

Como conclusão do que acabamos de ver, também podemos afirmar que GeneXus nos permite seguir tanto um processo de desenvolvimento ágil ou mais tradicional (como pode ser em cascata), se desejado.

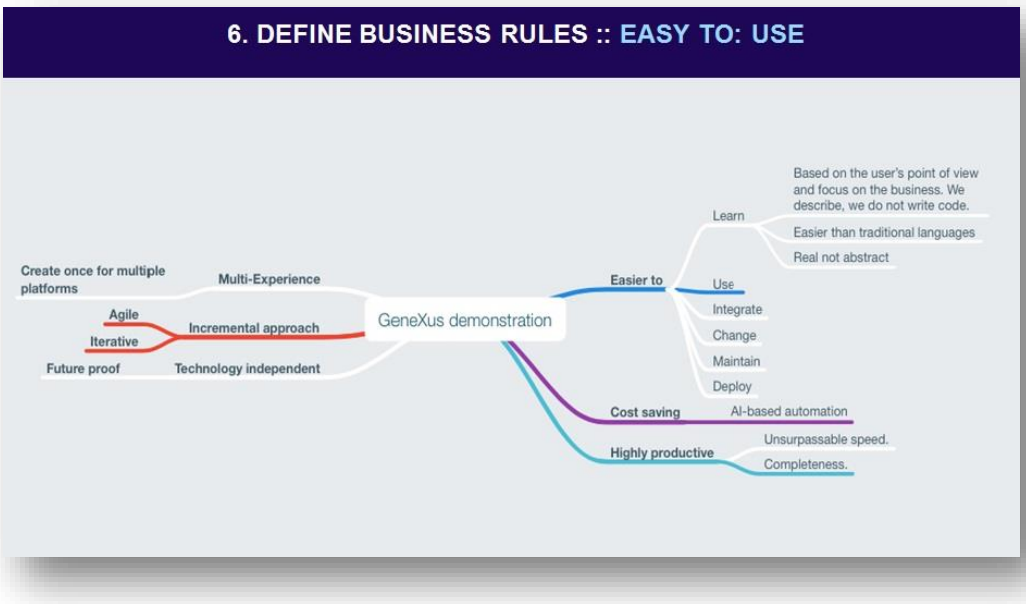
No caso de ter que entregar incremento do produto ao cliente em curtos períodos de tempo, como sugerem as metodologias ágeis, isto que acabamos de criar já pode ser colocado em produção e agregar valor ao cliente. É uma aplicação 100% funcional. Vale destacar que, sob nenhuma circunstância, é um produto descartável, ele será iterado sobre o mesmo e futuros incrementos serão entregues ao cliente, tendo esse desenvolvimento como base.



6. DEFINIR REGRAS DE NEGÓCIO

Até o momento, criamos uma aplicação com base no conhecimento disponível. Agora mostraremos como manter uma aplicação quando a realidade muda ou mais informações forem obtidas, simplesmente editando objetos existentes ou criando novos.

Em seguida, adicionaremos algumas regras de negócio.



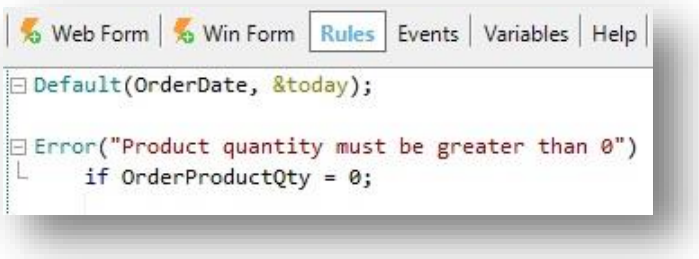
As Regras em GeneXus são os meios para definir a lógica do negócio associada a cada objeto. São escritas declarativamente e GeneXus decide de maneira inteligente que regra aplicar e quando aplicá-la.

Essas regras desempenham um papel muito importante nos objetos do tipo Transação, pois permitem descrever seu comportamento, por exemplo: atribuindo valores predeterminados, definindo controles de dados, etc.

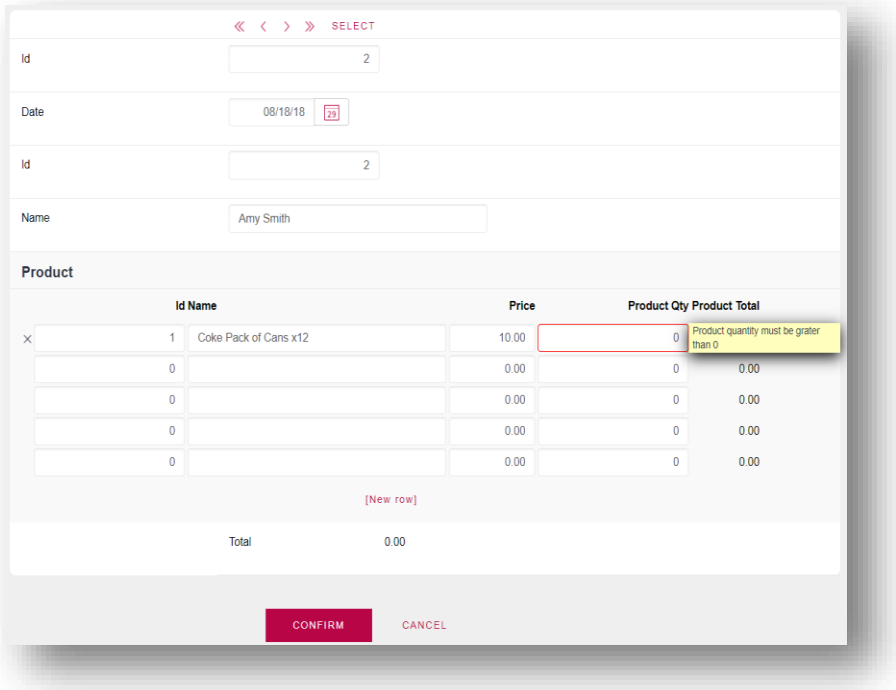
Suponhamos que nosso cliente solicite que a data do pedido de compra seja por padrão o valor da data do dia. Para conseguir isto, vamos ao setor de Rules da transação Order e adicionamos a regra Default a partir do menu superior Insert / Default

E concluímos a declaração desta regra da seguinte maneira…

Em seguida, adicionamos outra regra que estabelece um erro se a quantidade do produto inserido for 0.



Vamos salvar as alterações e vê-las em execução.



Desta forma, vimos como é fácil descrever as regras de negócio no GeneXus.

7. CRIAR TRANSAÇÃO PRODUCT

Nesta última interação com o formulário, achamos estranho precisar digitar o nome dos produtos novamente. Não seria melhor ter um catálogo de produtos onde esteja sua definição e seu preço? Esta observação é totalmente correta, portanto, mudaremos nosso sistema para que ele tenha um catálogo de produtos.

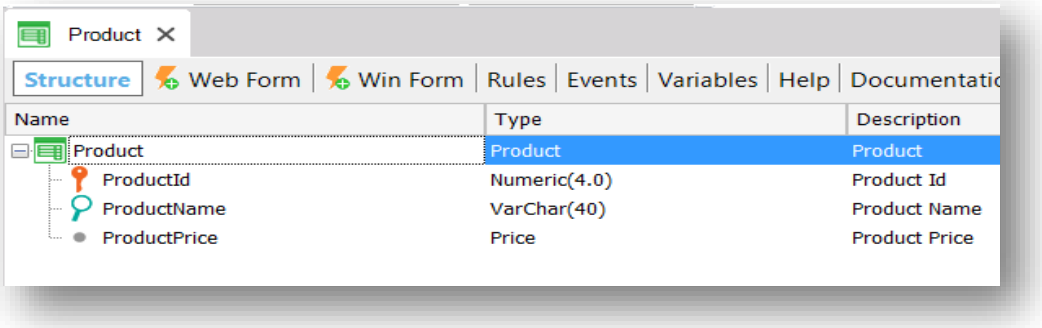
Na modelagem atual, o produto fazia parte da Transação Order, mas entendemos que deveria ser uma entidade independente do pedido, portanto, deve ser definido como uma Transação em si.

Não há nenhum problema com isto, pois GeneXus será responsável por manter a base de dados normalizada com base em nosso esquema de transações.

Então, continuamos a criar a Transação Produto.

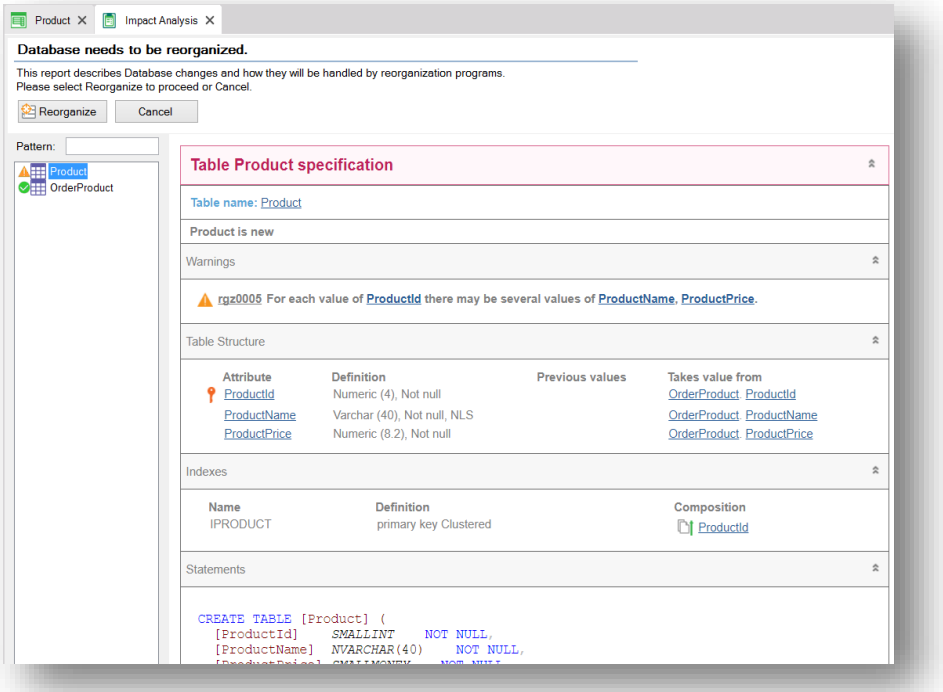
Os dados inerentes ao produto que já tínhamos na Transação Order são os seguintes:

- O identificador
- O nome
- E o preço



Observe que não foi necessário definir o tipo de dados de cada um dos atributos, pois GeneXus identificou que já haviam sido definidos anteriormente na transação Order.

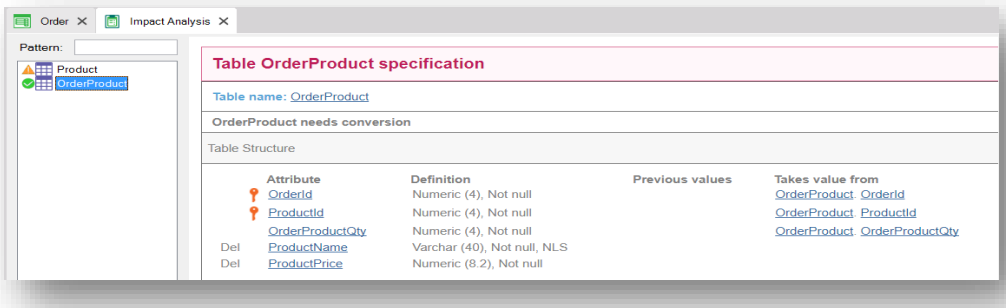
Salvamos e pressionamos F5.



GeneXus realiza a análise de impacto e analisa quais alterações devem ser feitas na base de dados com base nas alterações que fizemos nas transações e, em seguida, declara para cada objeto quais serão essas modificações.

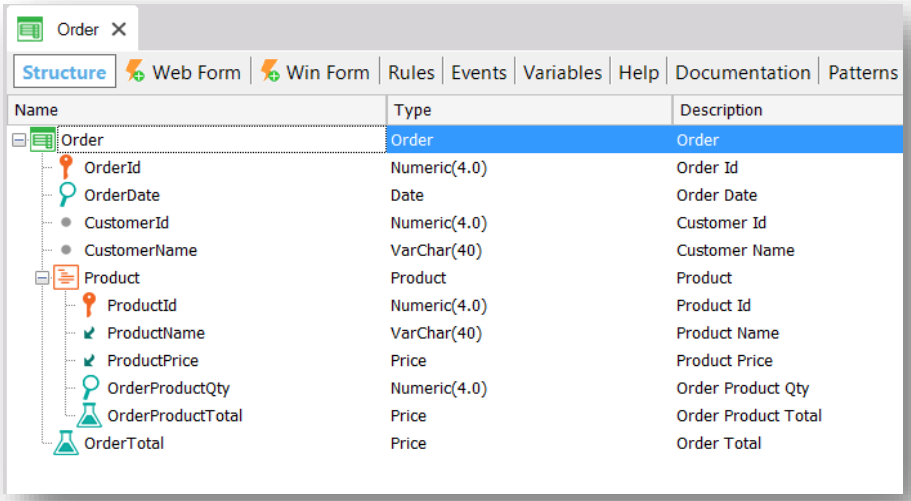
Vemos que GeneXus nos indica que deverá criar a tabela correspondente ao Produto e nos alerta em um warning de especificação, que para o mesmo valor de ProductId pode haver vários valores de ProductName e ProductPrice.

Também nos indica que deverá remover os atributos ProductName e ProductPrice da transação Order, pois eles se tornarão parte da transação Product e poderão ser inferidos pelo valor de ProductId. Lembremos que GeneXus nos garante que mantém a base de dados normalizada, portanto não podemos ter um atributo secundário em mais de uma tabela física.

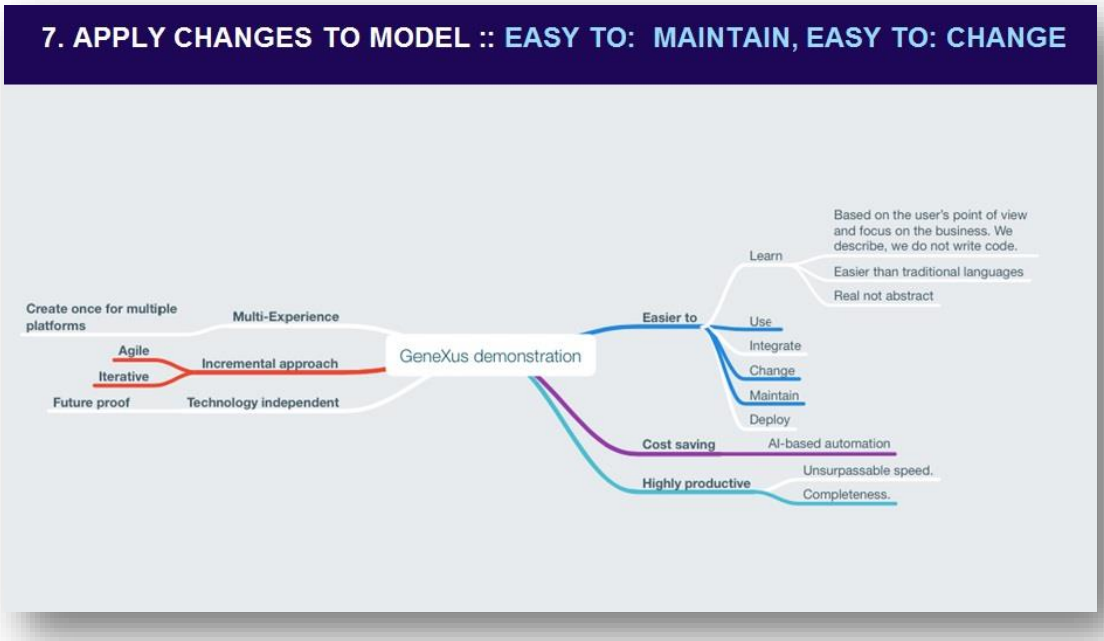


Quando a reorganização do modelo de dados é concluída, ao navegar pelos registros da transação Product, vemos que temos as informações dos produtos que inserimos através do Pedido. Essa automação de GeneXus é o que viabiliza a alteração no modelo de dados. GeneXus se encarregou de modificar a estrutura e da migração de dados de uma tabela para outra.

Mas ...o que aconteceu com as informações do Produto que estavam na Transação Order?



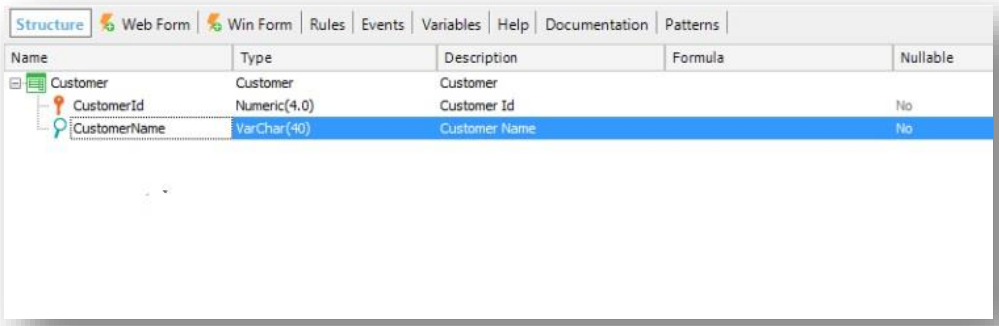
Vejamos que os atributos que agora pertencem fisicamente à tabela Produto aparecem com “uma seta para baixo” na Transação Order. Isto indica que estão sendo inferidos a partir do atributo ProductId. Isso mostra que GeneXus se encarrega de manter a base de dados normalizada.



8. CRIAR TRANSAÇÃO CUSTOMER

Avançando um pouco mais com a pesquisa, nos indicam que na empresa têm um mestre de clientes, portanto, não será correto ter os dados do cliente como parte da transação Order. Em vez disso, eles devem ser modelados em uma transação separada.

Então, criamos a transação Customer e a salvamos.

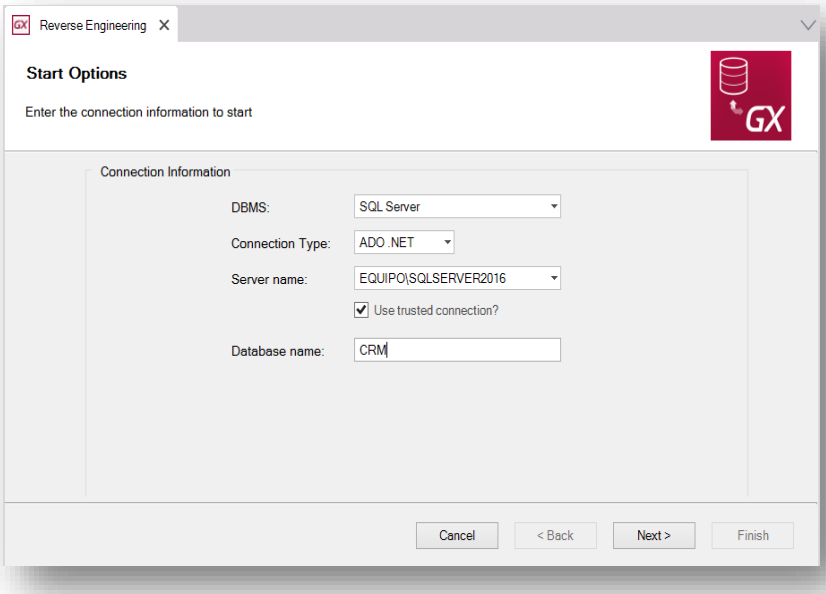


Os clientes estão registrados no CRM da empresa e é necessário que sejam obtidos de lá para registrar os pedidos de compra.

Generalizando isto, diremos que GeneXus nos permite integrar com qualquer sistema externo e extrair informações de sua base de dados para serem utilizadas no sistema a ser desenvolvido.

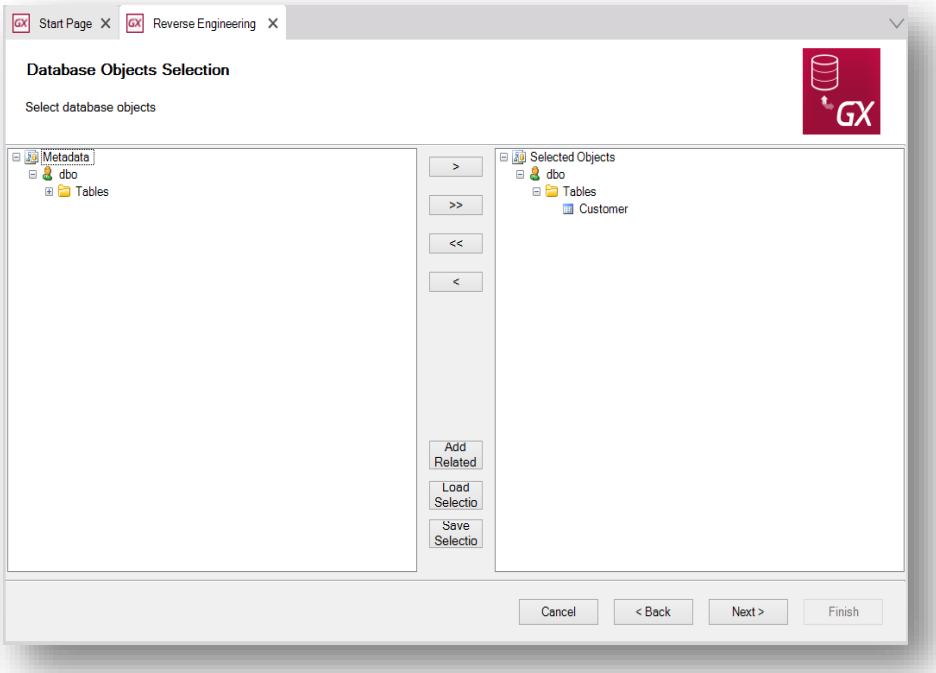
Para fazer isto, vamos ao menu *Tools* e selecionamos a opção *Database Reverse Engineering*

A partir daqui, indicamos o nome do servidor de base de dados e o nome da base de dados à qual desejamos nos integrar.

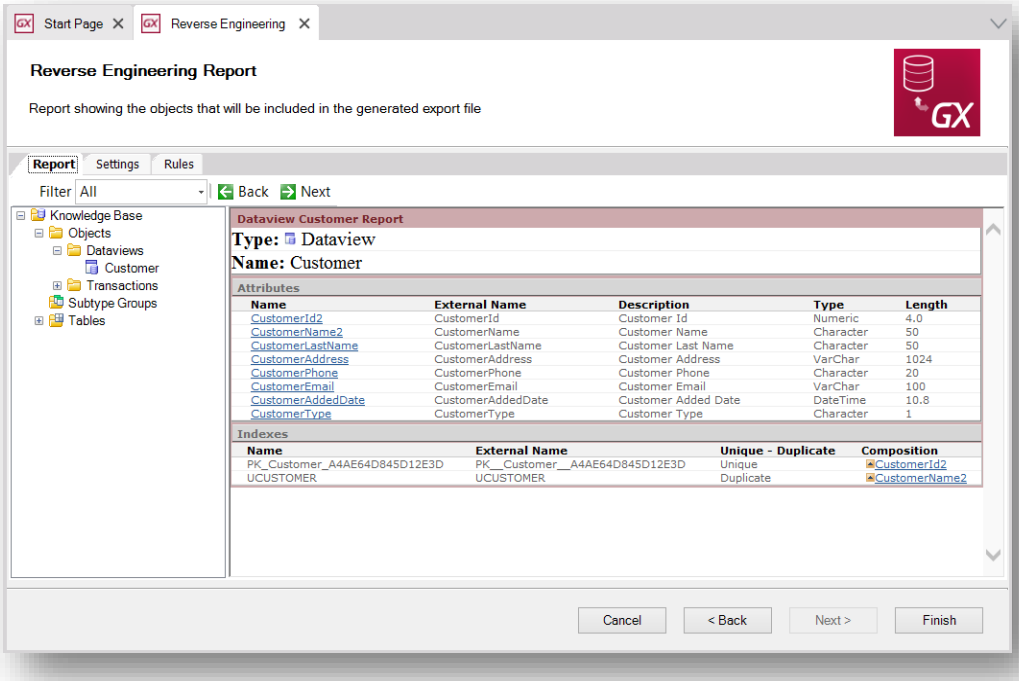


Ao clicar no botão *Next* nos é exibida uma lista de todas as tabelas da base de dados para que possamos selecionar as que precisamos.

Selecionamos a tabela Customer.



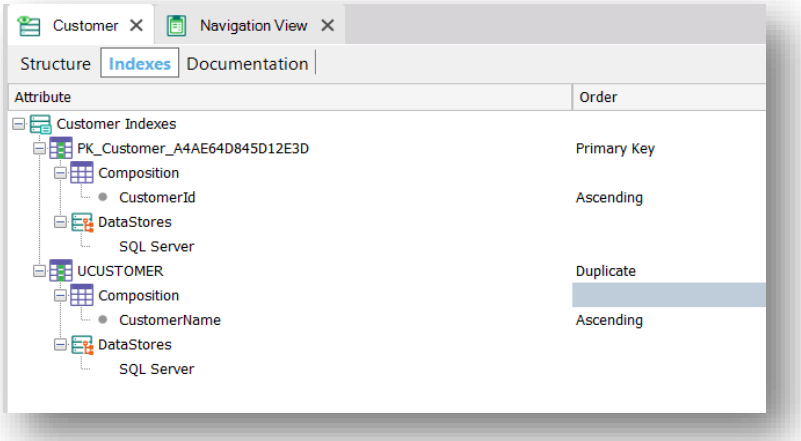
Ao clicar novamente no botão *Next* GeneXus nos indica que criará o Data View Customer

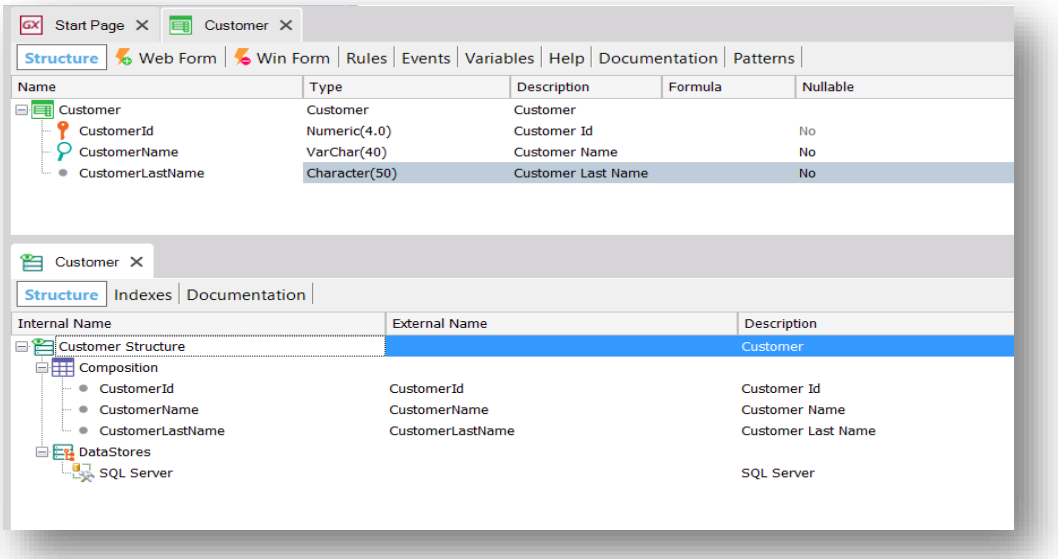


Vemos que, aos atributos CustomerId e CustomerName, foi adicionado o número 2, isso ocorre porque eles já existiam em nosso modelo. No caso de querer que sejam os mesmos, devemos explicitar isso. Para isto, vamos ao Data View Customer e alteramos o nome para estes atributos.

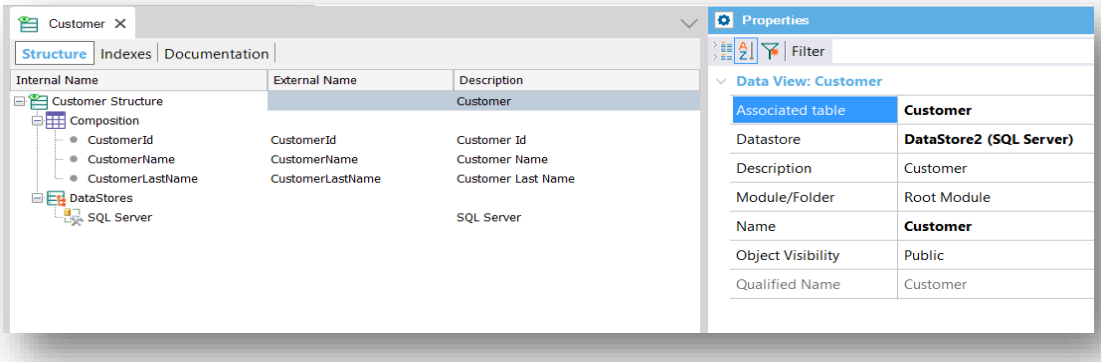
Também removemos os atributos que não usaremos. Deixamos apenas CustomerId, CustomerName e CustomerLastName. Portanto, excluímos CustomerAddress, CustomerPhone e CustomerEmail.

Também verificamos a definição dos índices para que o nome dos atributos sejam os corretos.



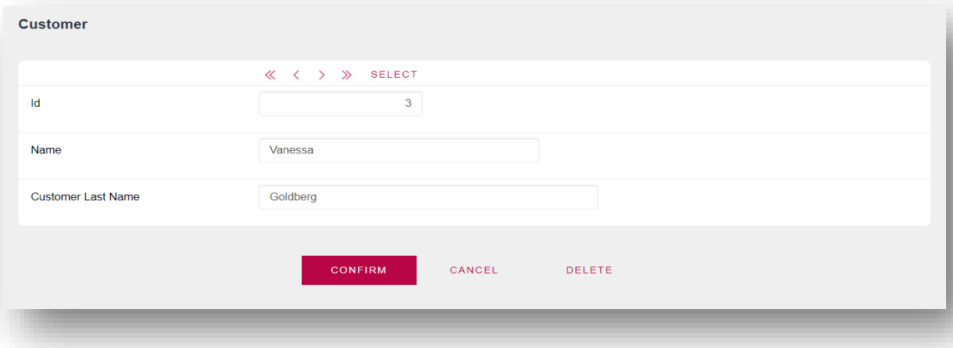


Depois de alterar os nomes, o Data View perde o conteúdo da propriedade *Associated table*, portanto atribuiremos o valor Customer.

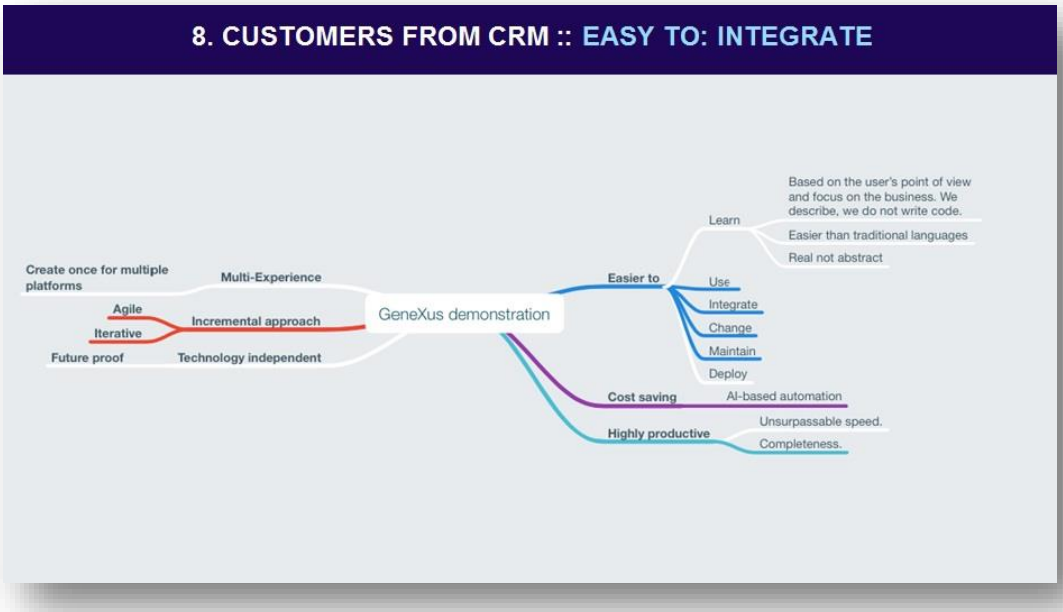


Pressionamos F5 e vemos que na Análise de Impacto indica que a transação Customer está associada ao Data View Customer e que não será reorganizada, ou seja, não criará uma tabela física associada.

Executamos e vemos que temos a transação Customer que obtém dados da tabela Customer do CRM.



De maneira muito simples, conseguimos nos integrar a um sistema externo, independentemente da tecnologia ou do gerenciador de base de dados destino.



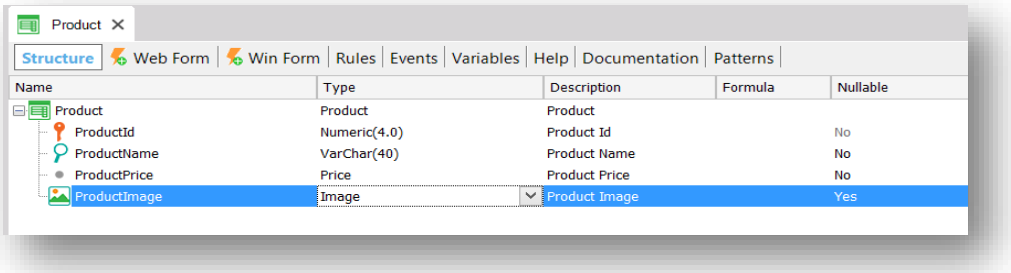
9. CATÁLOGO DE PRODUTOS

Agora mostraremos como podemos gerar aplicações mais avançadas no nível da usabilidade da interface do usuário. Aplicaremos o pattern Work With for Web e veremos em execução as facilidades que nos oferece.

Os patterns permitem definir e encapsular o comportamento de forma automática e replicá-lo em massa com base em um modelo de definição.

Vejamos um exemplo.

Para torná-lo mais real, adicionaremos a foto ao produto e em seguida aplicaremos o pattern Work With for Web.

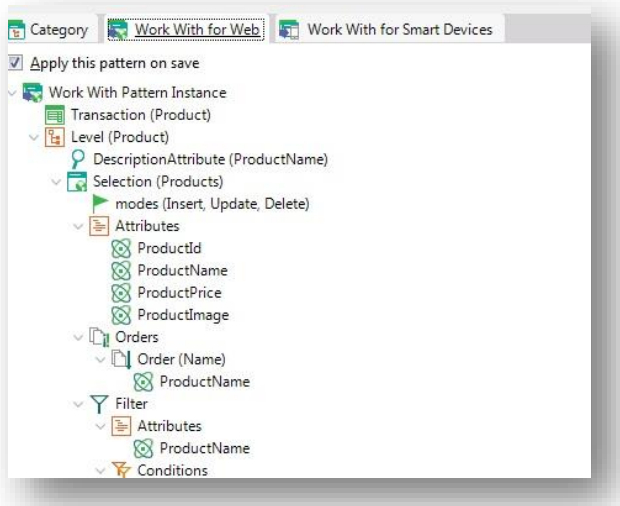


O tipo de dados Image é um dos chamados domínios semânticos, que em sua definição já tem implícito seu comportamento. Por exemplo, esse tipo de dados colocará um seletor de arquivos no formulário para fazer upload de uma imagem. Existem outros domínios semânticos, como Address, Phone, Email, etc.

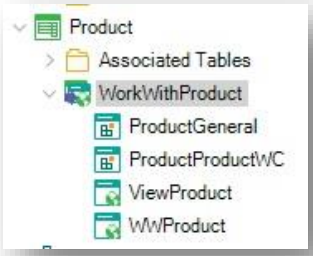
Para aplicar o pattern Work With for Web, vamos para a guia Patterns, selecionamos Work With for Web e clicamos nesta caixa que diz *Apply this pattern on save*.

Vale mencionar que o que estamos vendo é o que chamamos de instância do pattern aplicado à transação.

A partir daqui, podemos adicionar novas ordens, novos filtros, personalizar as colunas, adicionar ações etc. Salvamos.

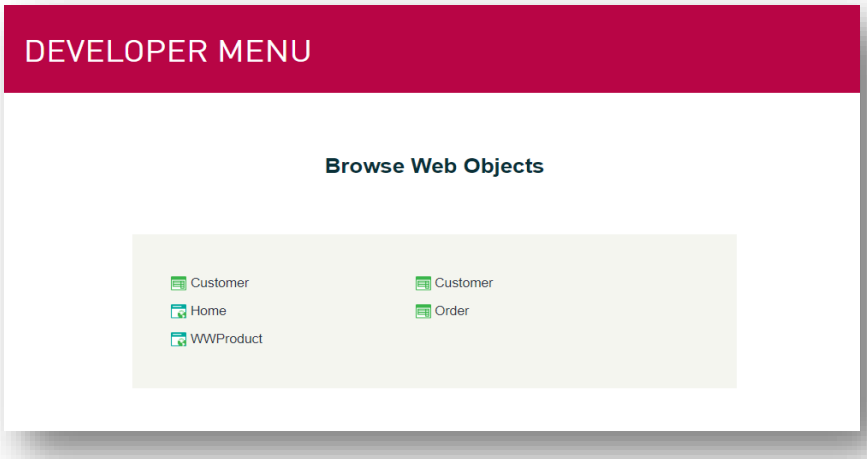


Se clicarmos duas vezes na transação Product, vemos que, automaticamente, aparece a lista de objetos criados automaticamente pela aplicação desse pattern.

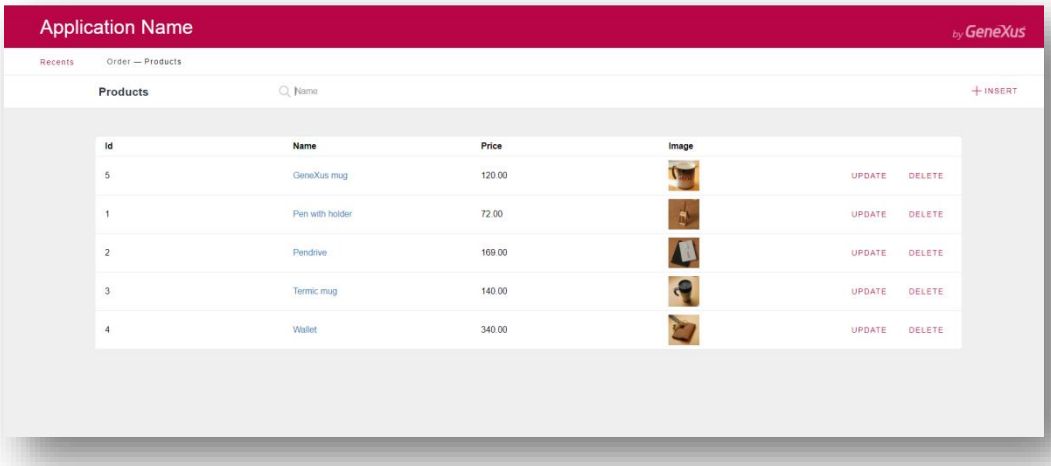


Vamos executar. Pressionamos F5.

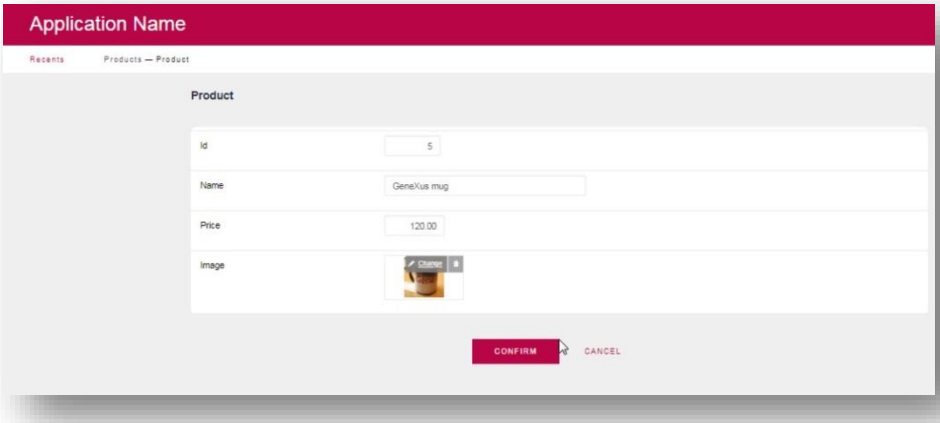
Ao executar, vemos que o acesso direto à transação Product mudou e que agora é acessado através do WWProduct.



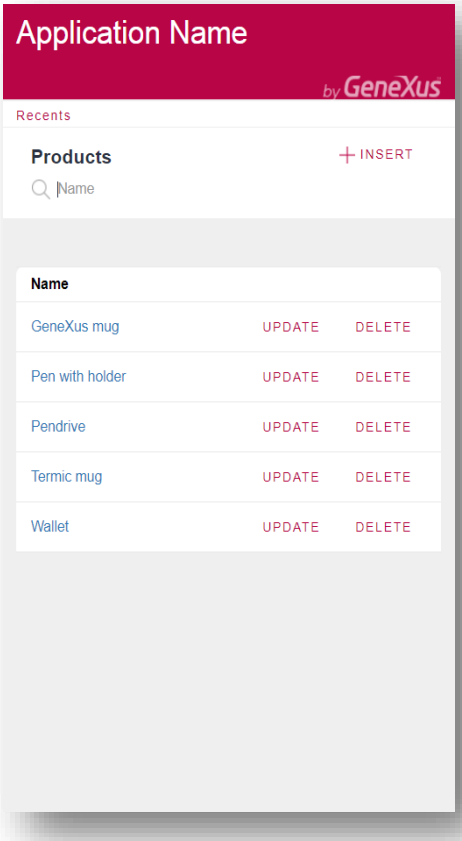
Vemos que GeneXus gerou automaticamente uma nova página web com a lista de todos os produtos, a partir dos quais podemos pesquisar, inserir, editar, excluir e exibir a ficha do produto.



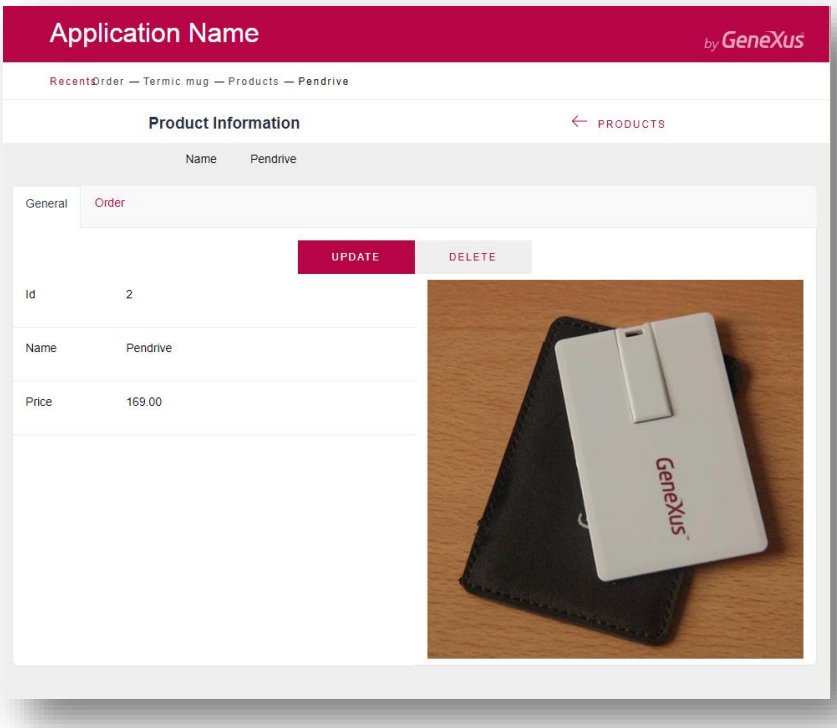
Como exemplo, inseriremos um novo produto. Pressionamos *Insert*.



Se encolhermos o navegador, vemos que a aplicação é responsiva, adaptando-se ao tamanho da tela disponível e priorizando a exibição das informações mais relevantes.



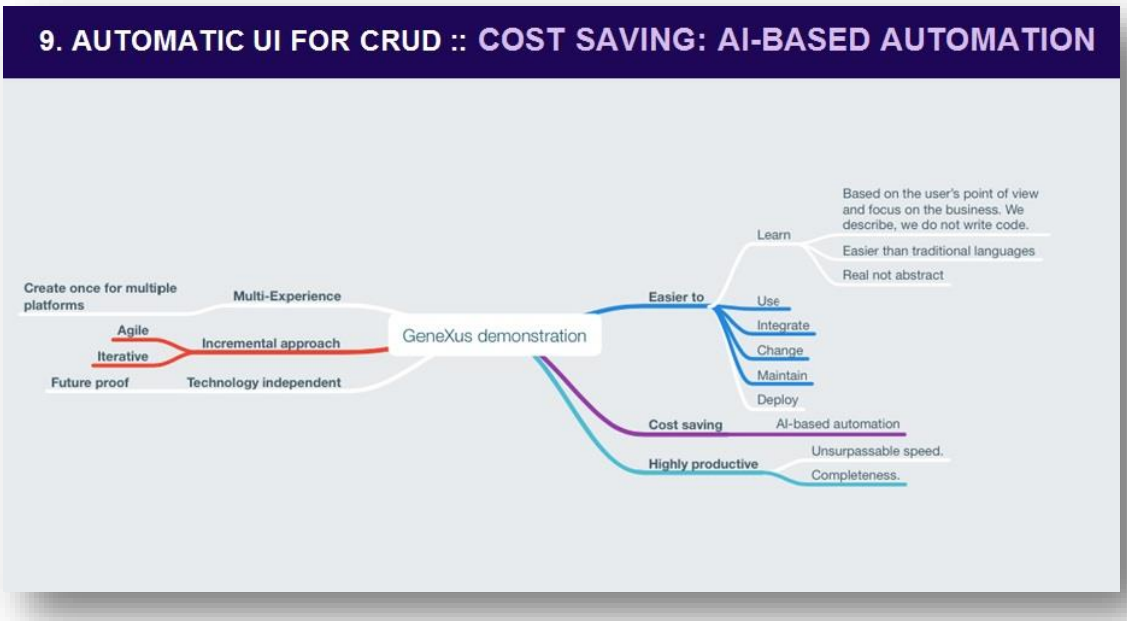
Se clicarmos no nome de um produto, acessamos a ficha do mesmo. Onde vamos ver todas as suas informações, bem como as informações das entidades relacionadas. Por exemplo, uma tab com os pedidos em que o produto foi incluído.



Isso torna a aplicação muito amigável de navegar, tendo ao alcance de um clique todas as informações necessárias sobre a entidade com a qual estamos trabalhando.

Reserve um segundo para perceber que tudo isto foi gerado pelo GeneXus com apenas 1 clique.

Vimos então como GeneXus nos ajuda a criar aplicações ricas em experiência do usuário, aplicando inteligência artificial e automação, com custo insignificante para o desenvolvedor.



10. MULTIPLATAFORMA

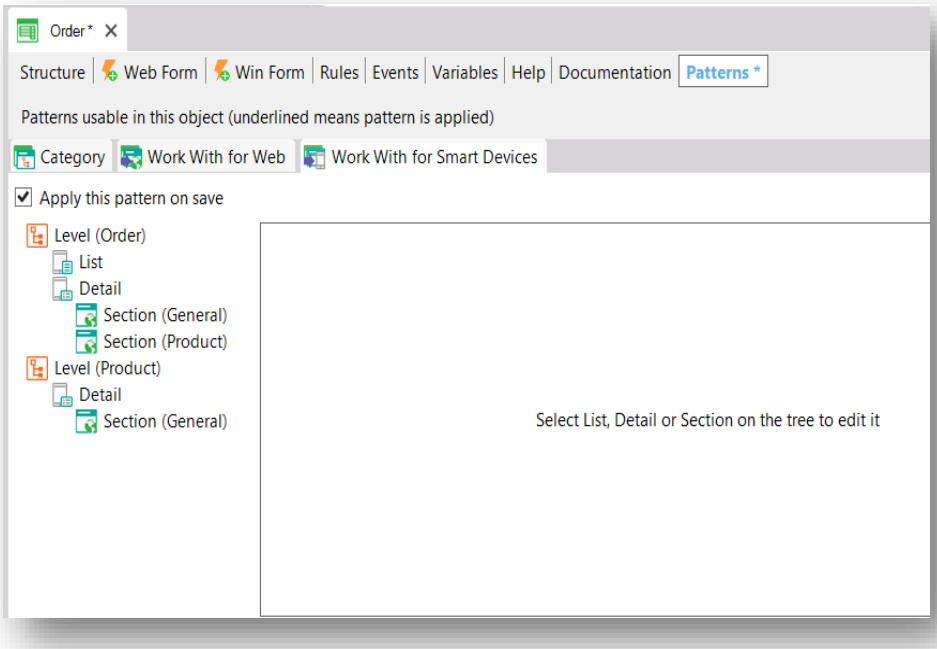
Vamos agora gerar uma aplicação para dispositivos móveis com o sistema operacional Android. A aplicação será gerada usando código Android nativo.

Para esta aplicação móvel, estaremos interessados em ter as seguintes telas:

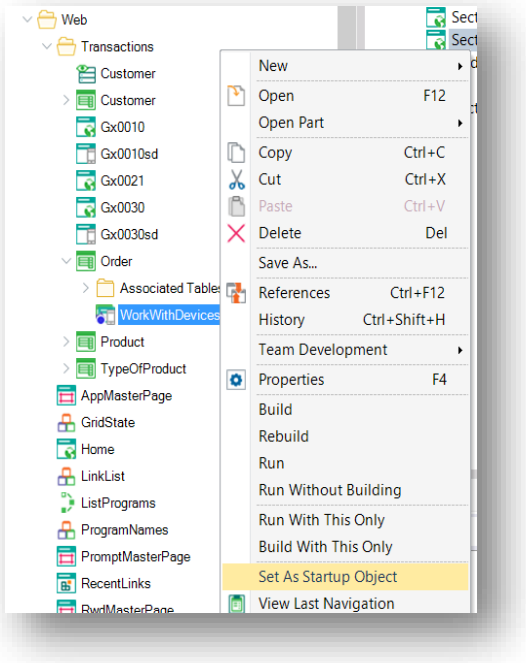
- Um menu de acesso a opções
- A lista de pedidos de compra classificadas por data em ordem decrescente.
- Para cada pedido de compra, a lista dos produtos incluídos com os totais e subtotais.
- Um catálogo de produtos, com o detalhe de cada produto.

Vamos começar com a lista de pedidos de compra.

Para isto, vamos para a guia Patterns da transação Order e aplicamos o pattern Work With for Smart Devices.



Salvamos e precisamos definir um objeto de smart devices como start up object. Portanto, enquanto estivermos prototipando, ao clicar em RUN, automaticamente será executada a aplicação no emulador. Localizamos a transação Order na janela de navegação de objetos e clicamos com o botão direito do mouse no objeto WorkWithDevicesOrder. O definimos como Start Up Object



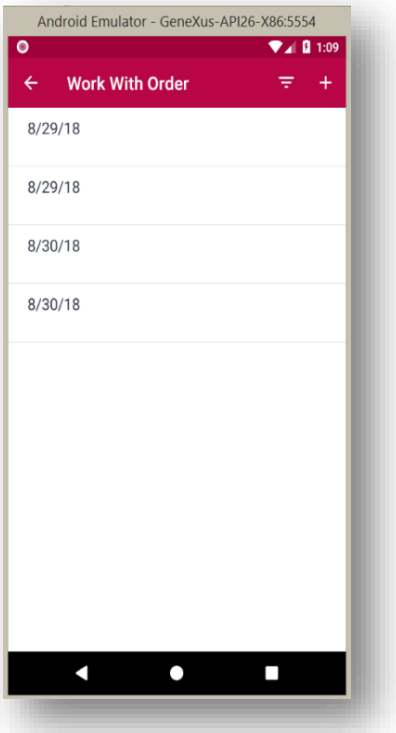
Vale ressaltar que se você tiver um dispositivo Android conectado por usb ao equipamento, com a configuração de depuração usb ativada, ao clicar em RUN, a aplicação será instalada automaticamente no dispositivo e poderá ser testada ali. Isto é de grande valor para tornar o teste mais real.

Pressionamos RUN para que GeneXus gere a aplicação para Android.

GeneXus está gerando todos os programas, telas e serviços web necessários para criar a aplicação móvel utilizando código nativo, neste caso Android.

Como GeneXus é quem gera os programas, nos torna independentes da tecnologia. Indo a um caso específico, quando iOS mudou sua linguagem de geração de Objective-C para Swift, enquanto muitos tiveram que reescrever suas aplicações, o usuário GeneXus não precisou fazer absolutamente nada, pois o gerador mudou para que agora sejam geradas as aplicações utilizando Swift.

Vemos que GeneXus criou uma tela com a lista de pedidos. Para cada um deles, é mostrado por padrão sua data.



Seria desejável que:

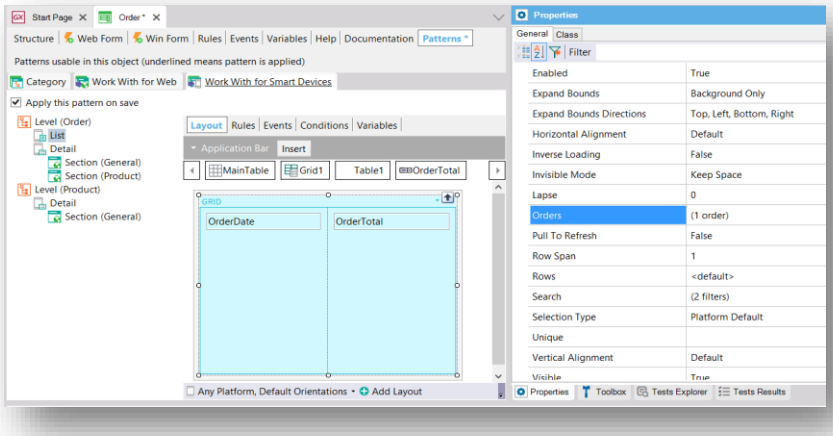
- O título da tela seja *Orders*.
- Que para cada pedido seja exibido também o total.
- Que sejam listados em ordem de data decrescente.
- E que o conteúdo da linha apareça centralizado verticalmente.

Vamos fazer estes agregados.

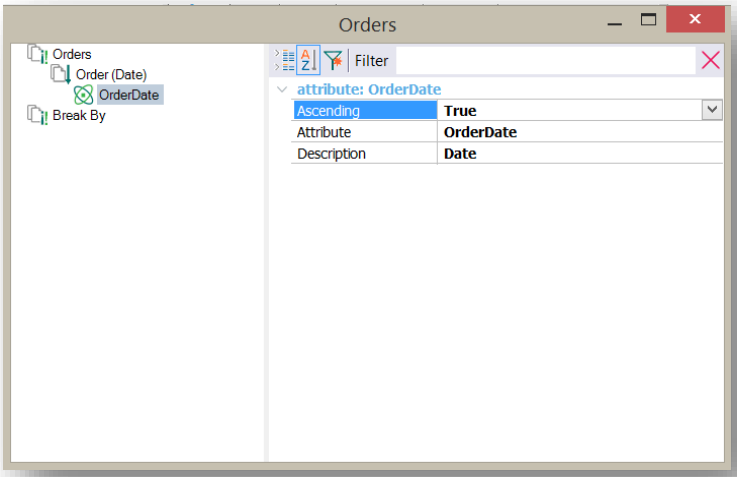
Nos posicionamos no nó List da instância Work With for Smart Devices da transação Order e modificamos o valor da propriedade *Caption* para colocar *Orders.*

A partir da Toolbox arrastamos o atributo OrderTotal para o grid. E modificamos a propriedade *LabelPosition* com o valor *None* para que não seja exibida a descrição.

Em seguida, clicamos no Grid e nos posicionamos na propriedade *Orders*



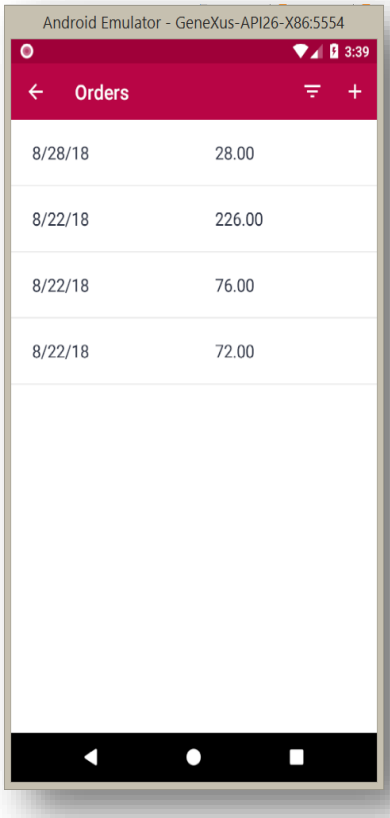
Ao clicar nos exibe uma tela para alterar as ordens especificadas.



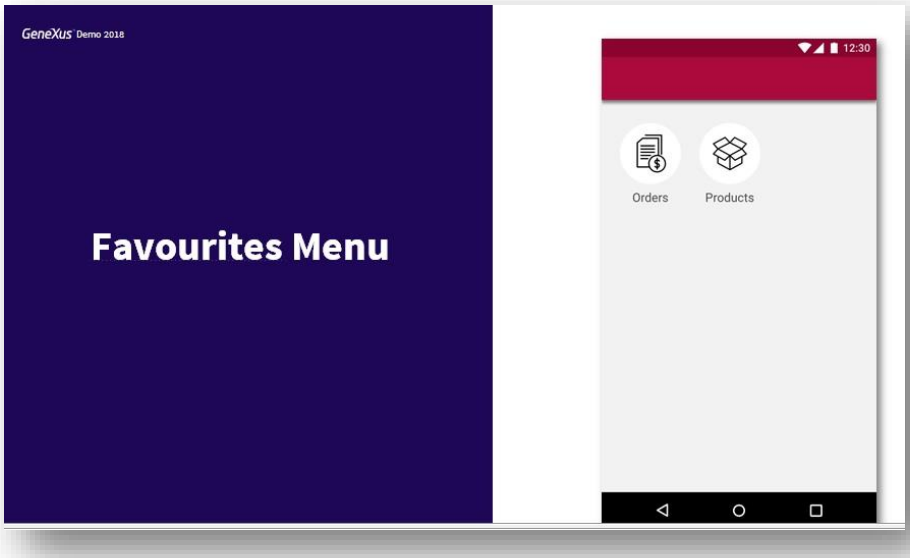
Selecionamos OrderDate e, para atender nosso requisito, alteramos o valor de *Ascending* para *False*.

Por último, alteramos o valor da propriedade *Vertical Alignment* de ambos os atributos para *Middle.*

Pressionamos RUN novamente para ver as alterações em execução.



Já vimos como é fácil modificar o conteúdo das telas. Agora vamos avançar um pouco mais rápido e aplicaremos o design que nos enviou o designer.



1. **Menu**

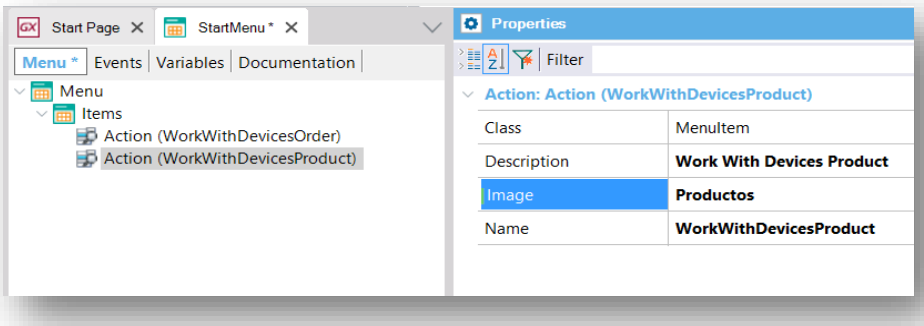
   Vemos que a primeira tela é um menu de acesso com opções.

   Para isto, criaremos no GeneXus um novo objeto do tipo Menu for Smart Devices.

   Já adicionamos o pattern Work With for Smart Devices à transação correspondente ao Produto.

   Vamos agora criar o menu de acesso.

   Adicionaremos uma ação ao menu para cada acesso que desejamos ter.



   Assim carregamos os acessos aos pedidos e aos produtos.

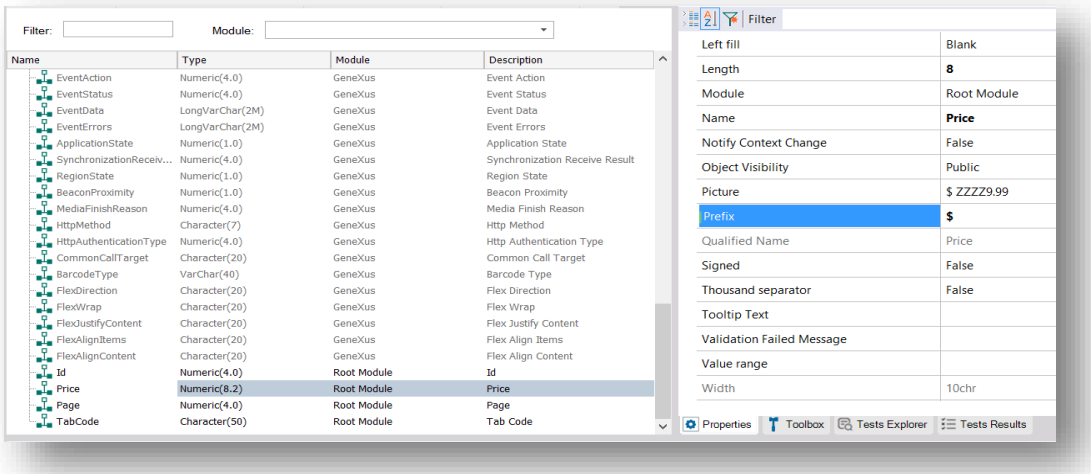
   Também podemos carregar uma imagem na propriedade Image para incluir os ícones que o designer nos enviou.

E fazemos o mesmo para selecionar o ícone correspondente aos Produtos.

O salvamos e o definimos como objeto Start up. Pressionamos RUN.

2. Lista de Pedidos

É desejável que o total seja mostrado com o sinal $ e faz sentido que esse mesmo critério seja usado para todas os valores no sistema. Então, vamos para a definição do domínio Price e vamos adicionar um prefixo usando a propriedade *Prefix*.



También vemos que se deseja que o total do pedido seja mostrado em negrito.

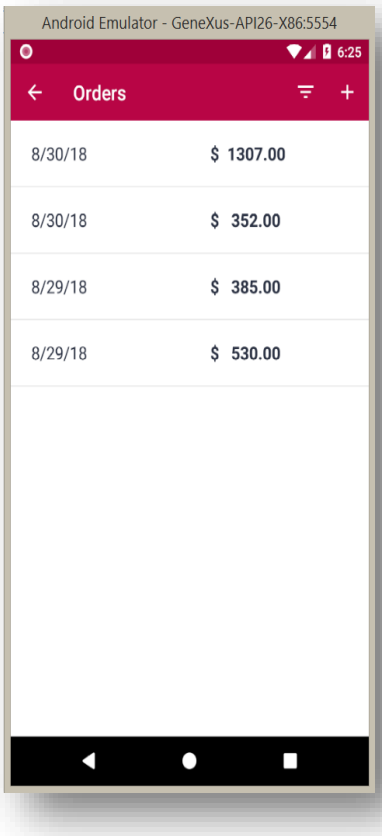
Para conseguir isso, abrimos o objeto chamado CarmineAndroid (do tipo Theme). E mencionamos que os Themes encapsulam a definição da estética dos componentes. Observemos que, para cada tipo de componente (por exemplo, Atributos, Tabelas, Grids, Botões), temos várias classes definidas. Em seguida, criaremos uma nova classe de atributo com fonte em negrito para poder usá-la no total dos pedidos.

Então, vamos ao nó de Atributos, clicamos com o botão direito do mouse e selecionamos a opção *Add class*. A chamaremos de AttributeBold.

GeneXus copia a definição da classe pai e nos permite personalizar sua estética através das propriedades. Agora vamos para a propriedade *Font Weight* e escolhemos *bold*. Salvamos.

Por último, vamos ao objeto OrderList para alterar a propriedade correspondente ao atributo do total do pedido, e colocamos a nova classe definida.

Selecionamos RUN para ver as alterações.



Vemos que efetivamente foram aplicadas as alterações e esta tela já atende aos requisitos estéticos solicitados.

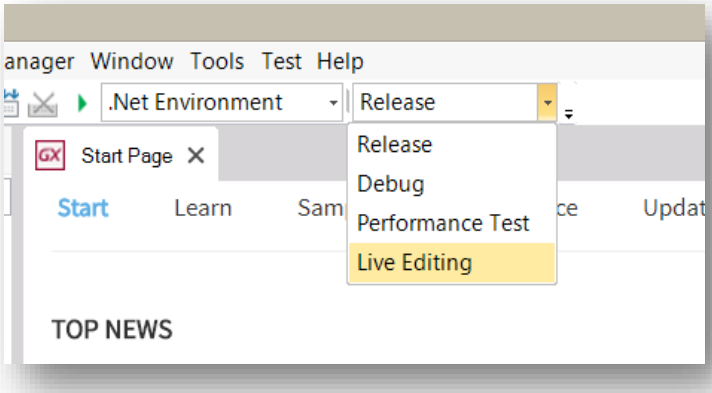
Mas ainda temos uma maneira mais dinâmica de trabalhar para aplicar o design às telas.

GeneXus possui um mecanismo chamado Live Editing que nos permite ver em tempo real as mudanças que estamos aplicando no design.

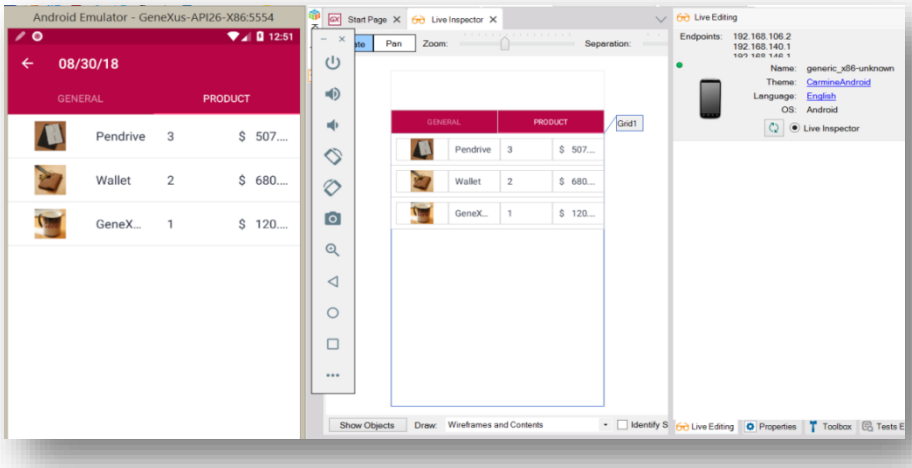
Vamos testá-lo com a tela correspondente à lista dos produtos de um Pedido

3. Lista de Produtos de um Pedido

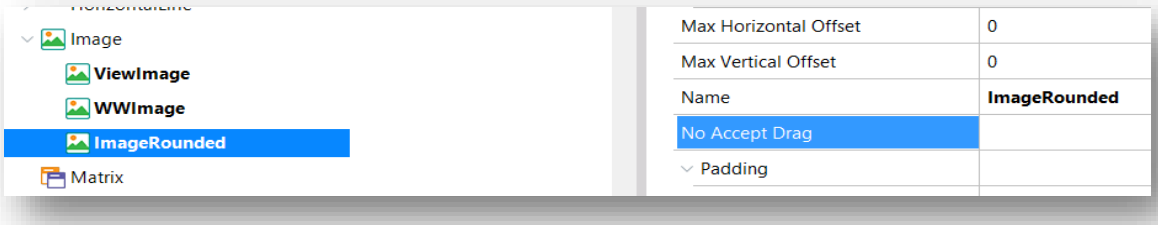
No combo box que se encontra na barra superior da IDE, selecionamos Live Editing.



Ao fazer isto, será aberta uma janela chamada Live Inspector, onde se verá o design com o qual está trabalhando com as alterações aplicadas em tempo real.

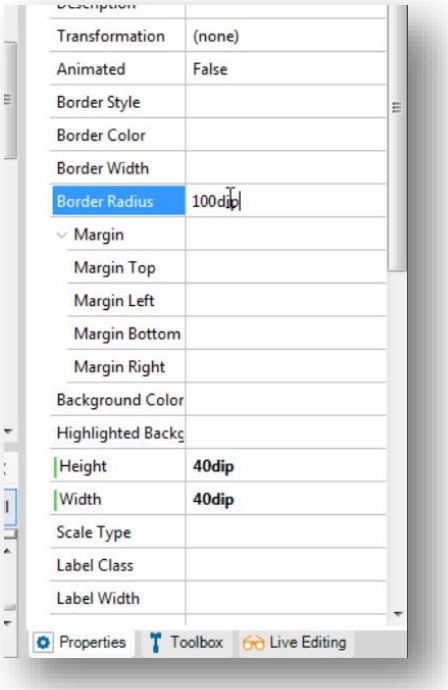


A mudança mais importante no nível estético para cumprir com o design desta tela é que temos que fazer com que as imagens apareçam arredondadas. Para isto, criaremos uma nova classe do theme dentro do nó Image.



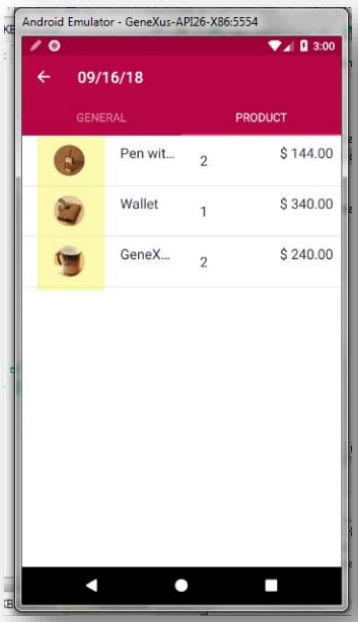
Criamos uma nova classe. Colocamos como nome ImageRounded

e vamos definir as seguintes propriedades:



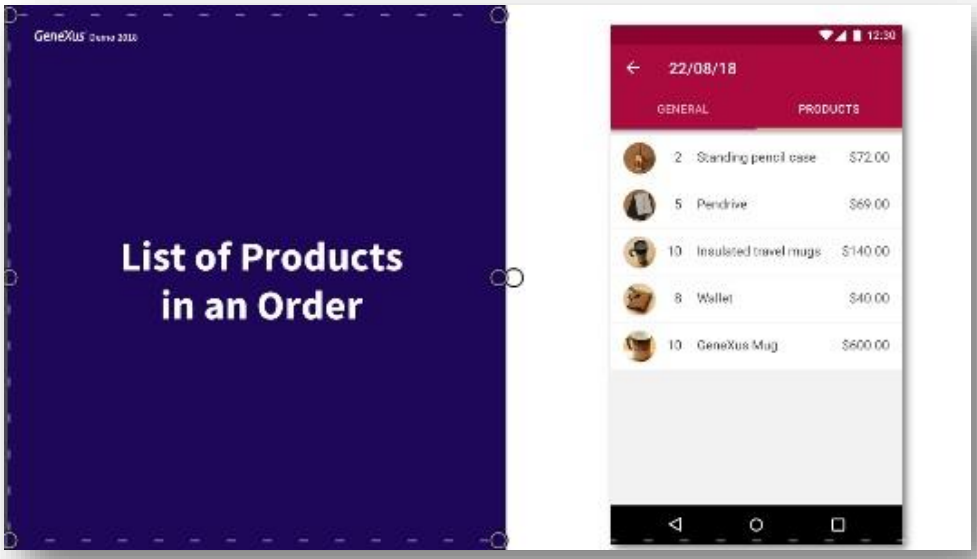
E salvamos as mudanças feitas no Theme.

E atribuímos a nova classe ao atributo da imagem... e vemos as alterações aplicadas



Vale a pena destacar que o Live Editing estabelece uma conexão bidirecional entre a KB e a aplicação em execução.

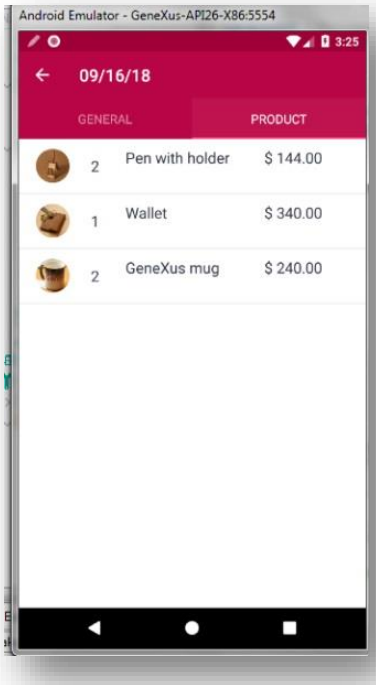
Ainda temos um detalhe para conseguir o design esperado. Temos que ajustar a largura das colunas para que o nome do produto apareça completo e o preço não seja visto cortado.



Para conseguir isto, nos posicionamos na tabela que contém as colunas do grid (Table1) e alteramos os valores da propriedade *Columns Style* para distribuir de melhor maneira a porcentagem para a largura das colunas.

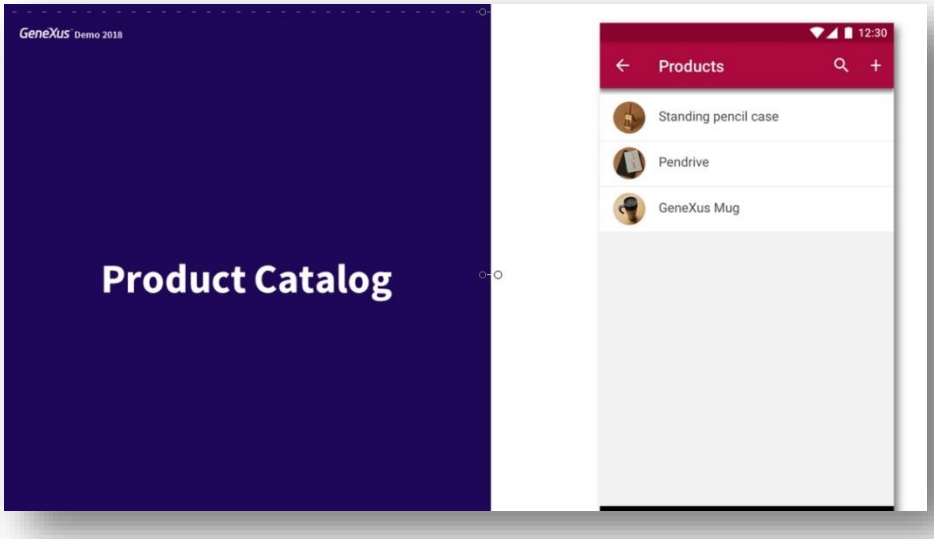
Definimos as porcentagens respectivamente: 15%, 10%, 40%, 35%

E vemos as mudanças no emulador.

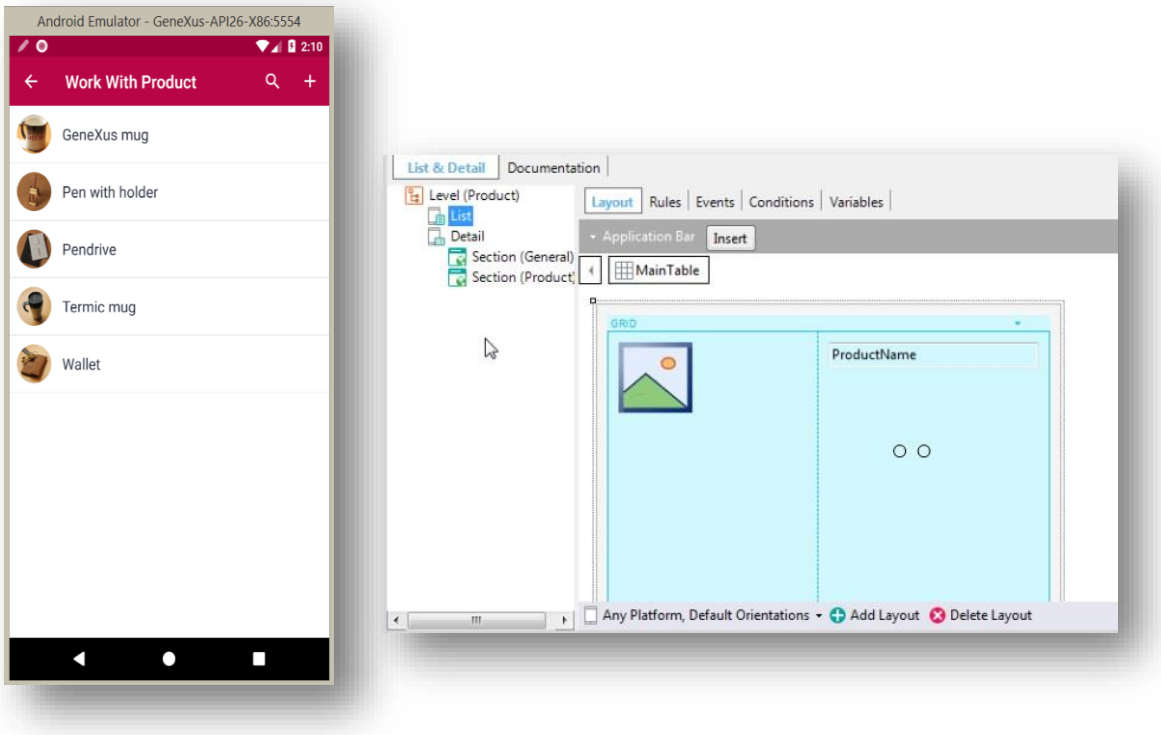


4. Lista de Produtos

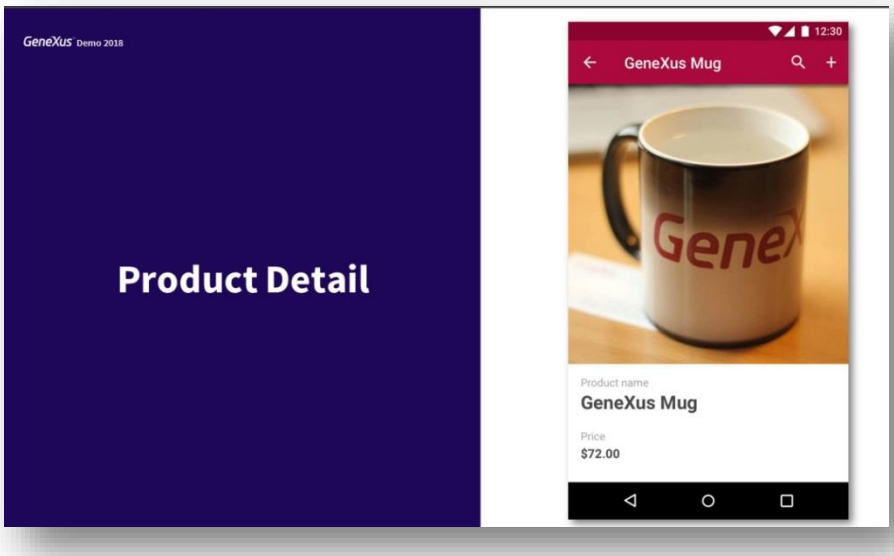
Vamos ver agora a que distância estamos do design desejado na tela da lista de produtos.



Selecionamos Work With Devices Product

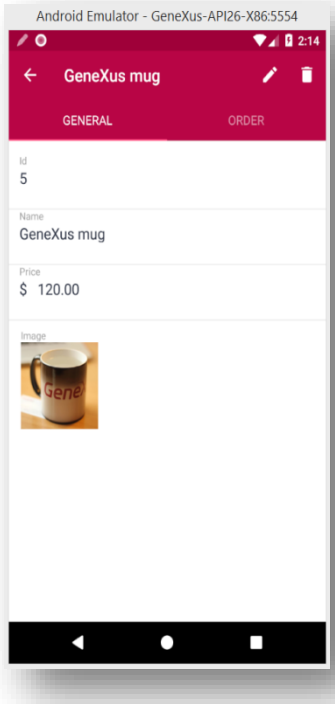


Alcançaríamos com distribuir um pouco melhor a largura das colunas para dar mais espaço à imagem e alterar o título da tela. Como são coisas que já fizemos nas telas anteriores, vamos pular essa personalização e vamos para nossa última tela.

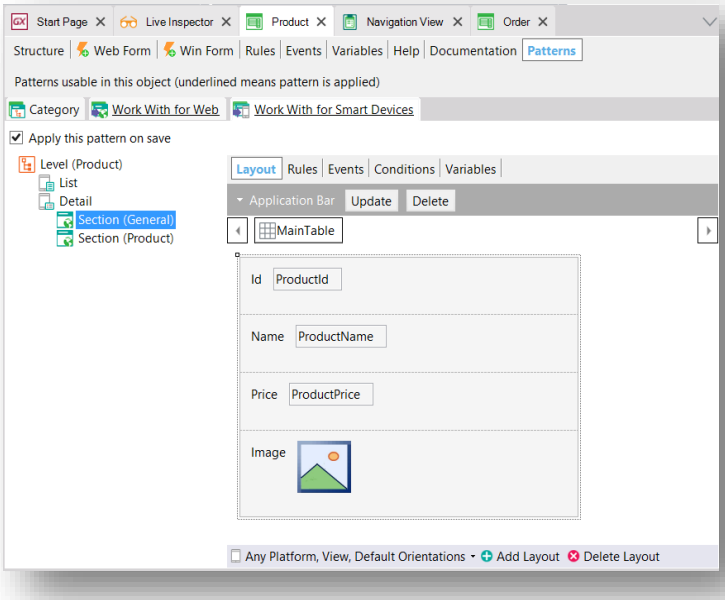


5. Ficha de um Produto

Até agora, a tela criada por GeneXus por padrão para mostrar as informações gerais do produto é assim:



A partir do objeto Work With devices Product, vamos à Seção general para editar esta tela:



Faremos as seguintes alterações:

Para o atributo da imagem, definimos a propriedade *Label Position* com o valor *None* e o posicionamos na parte superior da tela, simplesmente fazendo drag and drop.

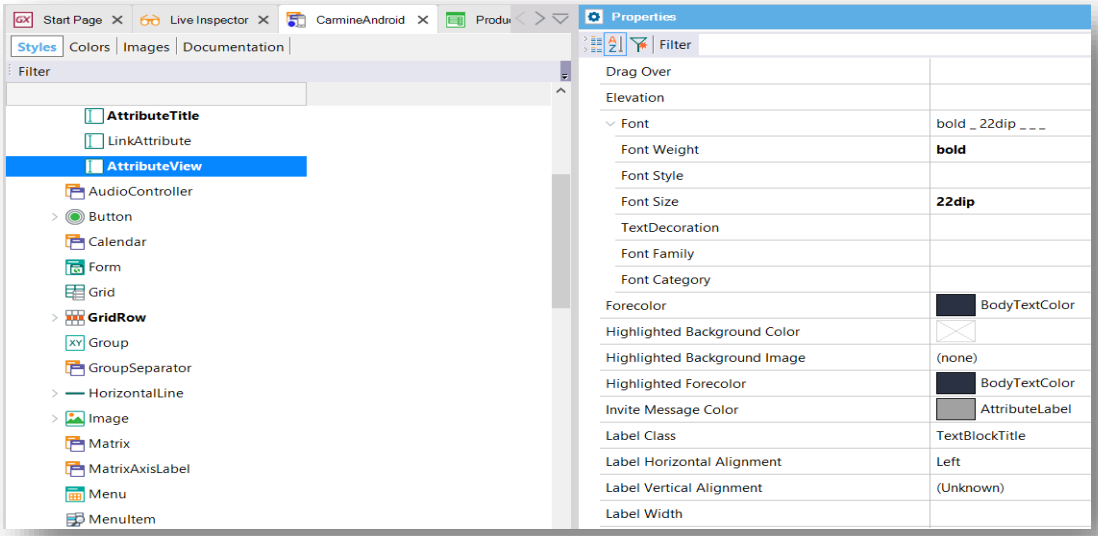
Em seguida, teremos que editar as propriedades da classe do theme que está utilizando a imagem. Vemos que a classe utilizada é ViewImage.

Então, vamos ao theme e a esta classe ViewImage, vamos alterar as seguintes propriedades:

- Width: (Use default, sem especificar)
- Height: (Use default, sem especificar)
- Margin: (Use default, sem especificar)

Salvamos. Em seguida, na tabela MainTable, que é a que contém propriamente a ficha do produto, alteraremos a classe para que não sejam exibidas as linhas separadoras. Então, definimos a classe Table.

Por último, criaremos uma nova classe de Atributo para poder destacar um pouco mais o nome do mesmo e seu preço. Portanto, criamos a classe de nome AttributeView e definimos as seguintes propriedades: *FontSize = 22 dip e Font Weight = bold*.

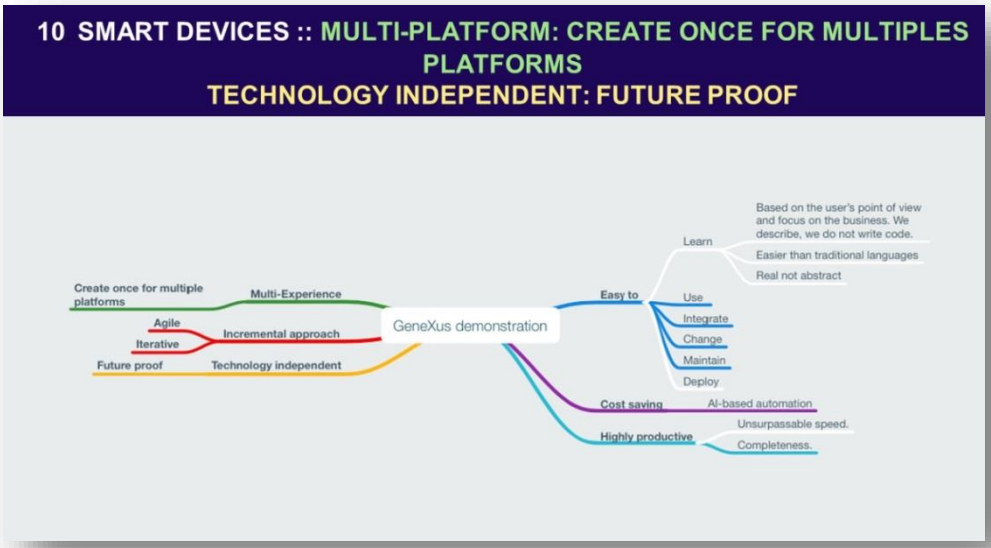


Salvamos e aplicamos esta classe aos atributos de nome e preço. Também modificamos a propriedade Label position para que a descrição não fique visível.

E vemos as mudanças no emulador.

Em resumo, vimos que, a partir do que já definimos para o sistema web (estrutura e comportamento), podemos gerar aplicações para dispositivos móveis reutilizando o conhecimento do negócio que encapsula a Knowledge Base.

Isto se traduz no benefício da multiexperiência e a necessidade de criar apenas uma vez para múltiplas plataformas.



Vimos também que GeneXus nos permite tornar-nos independentes da tecnologia, uma vez que é o gerador correspondente que se encarrega de escrever a aplicação com base no estado da arte atual. Isto se traduz no benefício de ser independente da tecnologia, ou seja, à prova de futuro..

Lembre-se também de que GeneXus se encarregou de criar todas as telas móveis e web services necessários para que a aplicação interaja com o backend, economizando muito tempo. É por isso que dizemos que GeneXus nos ajuda a ser altamente produtivos, pois se encarrega da integridade necessária para criar uma aplicação para dispositivos móveis.

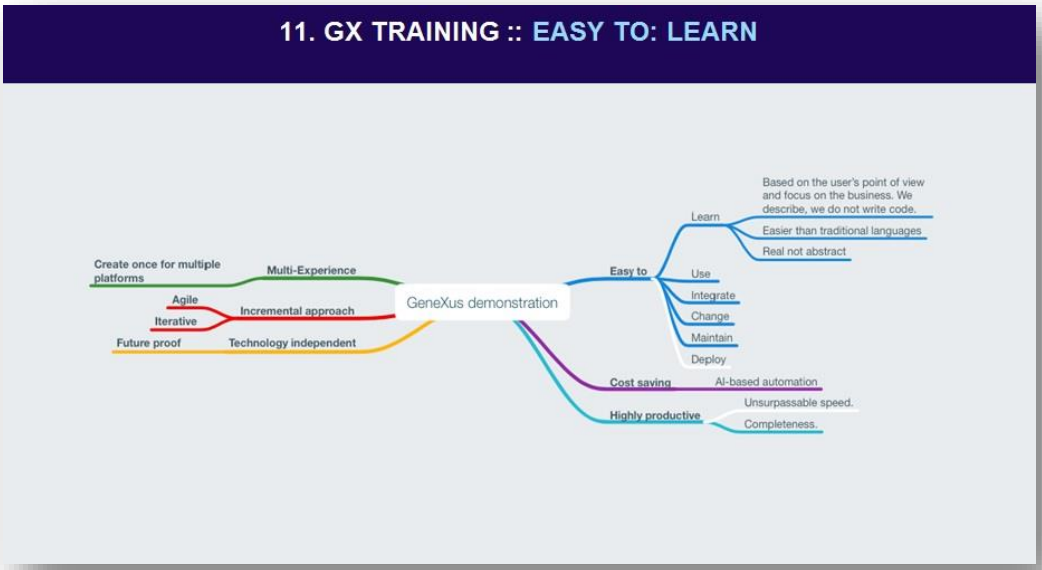
Com o pattern Work With for Smart Devices, vimos novamente o poder de aplicar padrões, gerenciando a partir da definição da instância e da definição das configurações gerais do pattern, a criação e manutenção de todas as telas móveis.

Isto foi conseguido aplicando inteligência artificial e automação.

Por último, gostaríamos de encerrar a demonstração com um conceito importante e, na realidade, que é cada vez mais complexo criar sistemas e aplicações, pois há cada vez mais tecnologias que precisam ser usadas em conjunto e mais sistemas com os quais temos que nos integrar.

GeneXus então consegue ser muito simples por fora, sendo complexo por dentro. Muitas coisas são resolvidas interna e automaticamente e isso nos livra, os Analistas GeneXus, de ter que fazê-las. É por isso que dizemos que GeneXus é fácil de usar: simples por fora e complexo por dentro.

Se você deseja capacitar-se e começar a utilizar GeneXus, o curso no modo de autoestudo consiste em vídeos com uma duração total de 7 horas e meia.



GeneXus é muito fácil de aprender.

11. RESUMO

Até agora, vimos como os recursos do GeneXus se traduzem em benefícios concretos para o usuário da ferramenta.

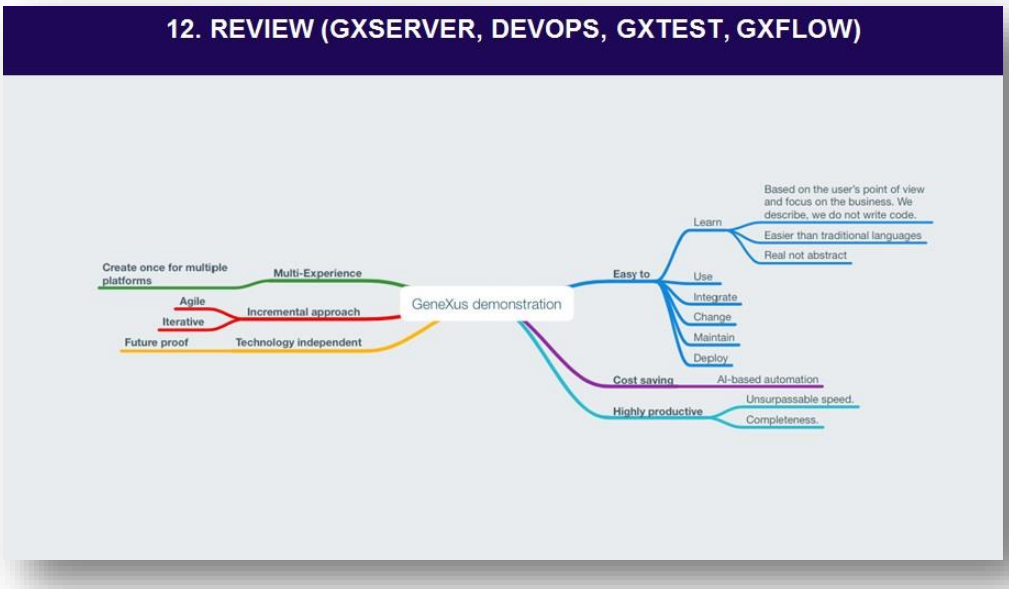
Mas existem vários outros recursos e produtos que completam a suíte GeneXus. Por exemplo:

GXserver

DevOps

GXtest

GXflow



12. GX SERVER

GeneXus Server é um produto que automatiza a integração do conhecimento. Permite o desenvolvimento distribuído como se estivesse centralizado, pois trata da consolidação do projeto de forma automática e eficiente.

13. GX TEST – TESTES UNITÁRIOS

O GXtest é um produto integrado à IDE do GeneXus e permite definir e salvar testes, para que seja possível executá-los novamente quando acharmos conveniente, por exemplo, depois de ter impactado alterações na linha de desenvolvimento.

14. GX FLOW – BUSINESS PROCESS MODEL

GXflow é uma ferramenta de workflow integrada ao GeneXus que permite modelar, gerenciar e otimizar os processos de negócio de uma empresa para criar aplicações de maneira simples e eficiente.

15. DEPLOY TO CLOUD F6

Quanto à funcionalidade de deploy to cloud, nos permite fazer o deploy da aplicação na nuvem apenas pressionando uma tecla.

16. ENCERRAMENTO

Então, cumprimos o desafio? Poderia ter feito tudo isso em outra tecnologia em menos de 1 hora?

The GeneXus Team