Curso prático de atualização para

PARTE 3

Setembro 2018

CONTEÚDO

OBJETIVO

Para alcançar o objetivo de desenvolver aplicações modernas, melhorando a integração com serviços externos, neste prático trataremos os temas de OData e Inteligência Artificial (AI).

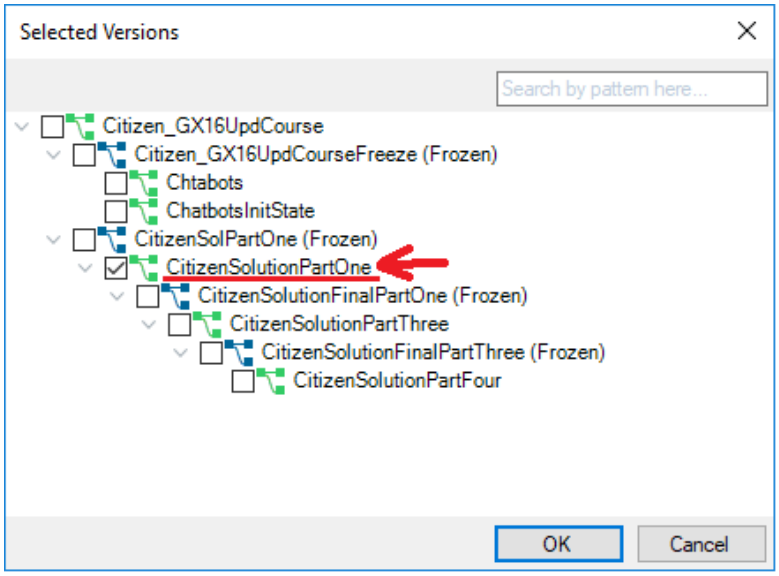
A aplicação em que vamos trabalhar é uma simplificação de uma app para o município de uma cidade, que oferece um frontend (Web/SD) para que os cidadãos, através de seu identificador de usuário, possam fazer reclamações (por exemplo, árvores caídas, semáforos que não funcionam, carros mal estacionados,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

Para melhorar o turismo da cidade, será feita uma integração com os serviços TripPin, que permitirão manter um registro dos voos feitos pelos usuários para diferentes destinos.

Além disso, será oferecido traduzir a descrição de atividades culturais ao idioma do dispositivo, e para melhorar a acessibilidade, também será oferecido reproduzir essa descrição como áudio.

VAMOS COMEÇAR

Será baseado na versão da KB com a qual você trabalhou no prático de Design Systems e UX, no estado que havia ficado ao concluir o prático. Se por algum motivo você a perdeu ou não a encontra, crie outra a partir http://samples.genexusserver.com/v16, (KB chamada Citizen_GX16Course) escolhendo a versão chamada CitizenSolutionPartOne.



Observe no KB Explorer que nós adicionamos a pasta GeneXus/Web/Backend_ODATA para que lá coloque os objetos que vá criando para esta parte do prático. E a pasta AI para criar lá os procedimentos necessários para se conectar ao provedor de AI que será indicado.

ODATA

Deseja-se adicionar duas novas funcionalidades para o backend da aplicação, consumindo OData a partir do TripPin. O que se quer é:

1. Manter um registro de voos de saída, gerenciando as informações dos voos, aeroportos e companhias aéreas.
2. Listar todas as informações relacionadas aos voos com destino a determinado aeroporto.

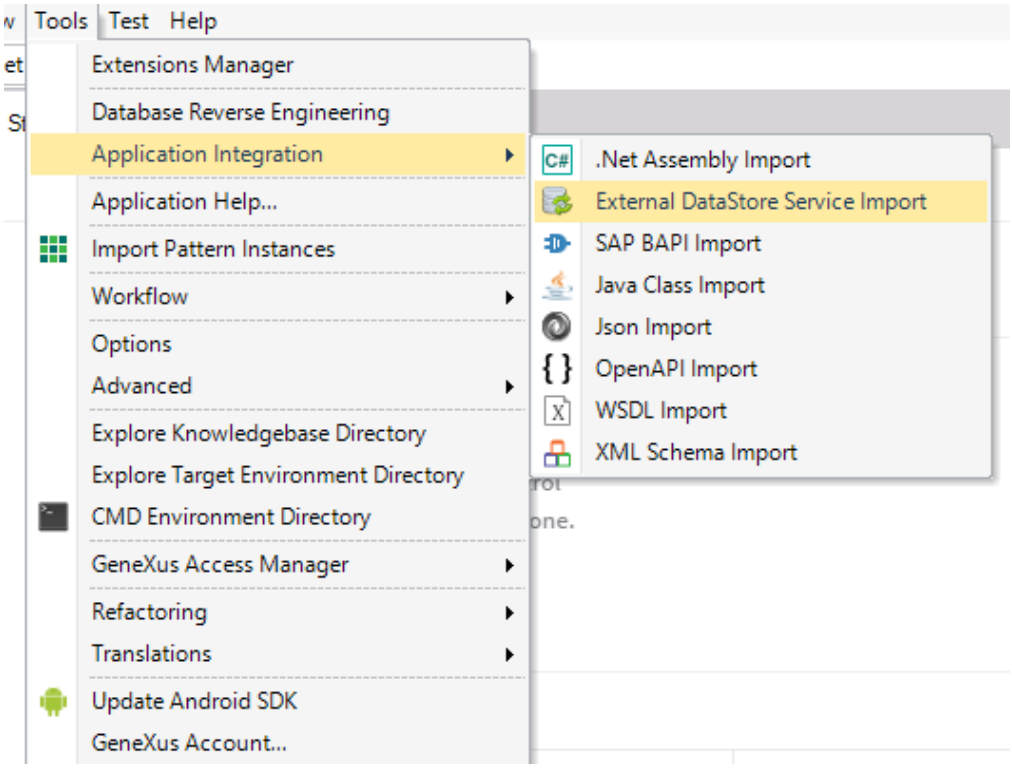
IMPORTAÇÃO DE SERVIÇOS

Para implementar as novas funcionalidades, precisaremos importar alguns serviços a partir do TripPin.

As entidades das quais queremos usar a implementação são a de Airlines e Airports. Os voos serão uma entidade da nossa própria KB, que teremos que criar mais tarde.

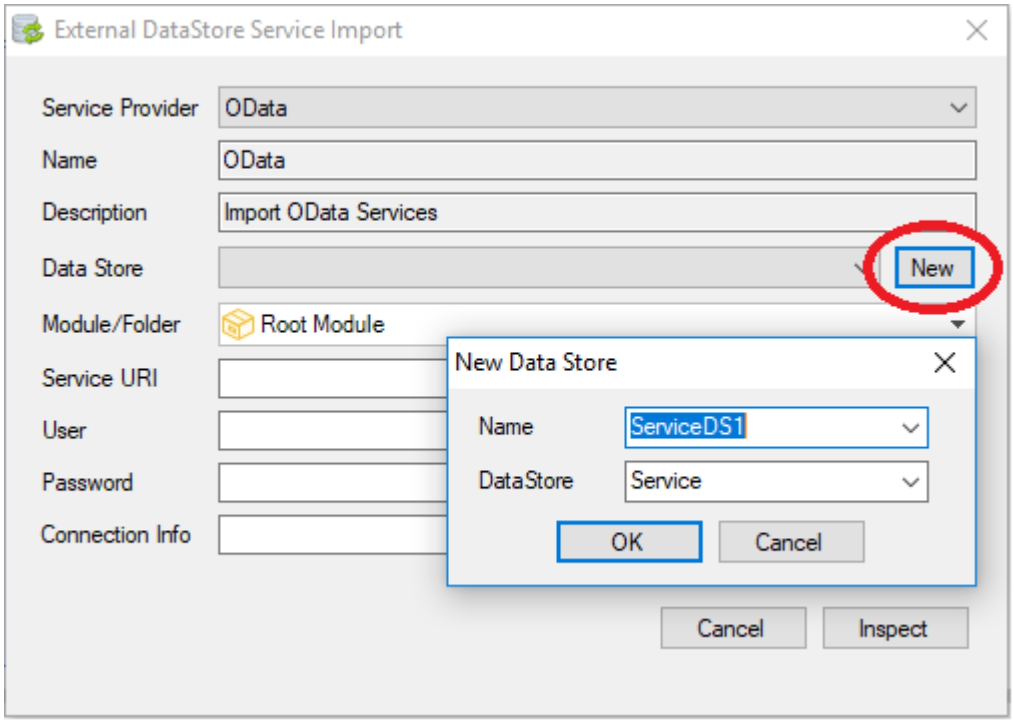
SOLUÇÃO POSSÍVEL

Importaremos a partir do TripPin o que é necessário para podermos implementar as novas funcionalidades. Para fazer isto:



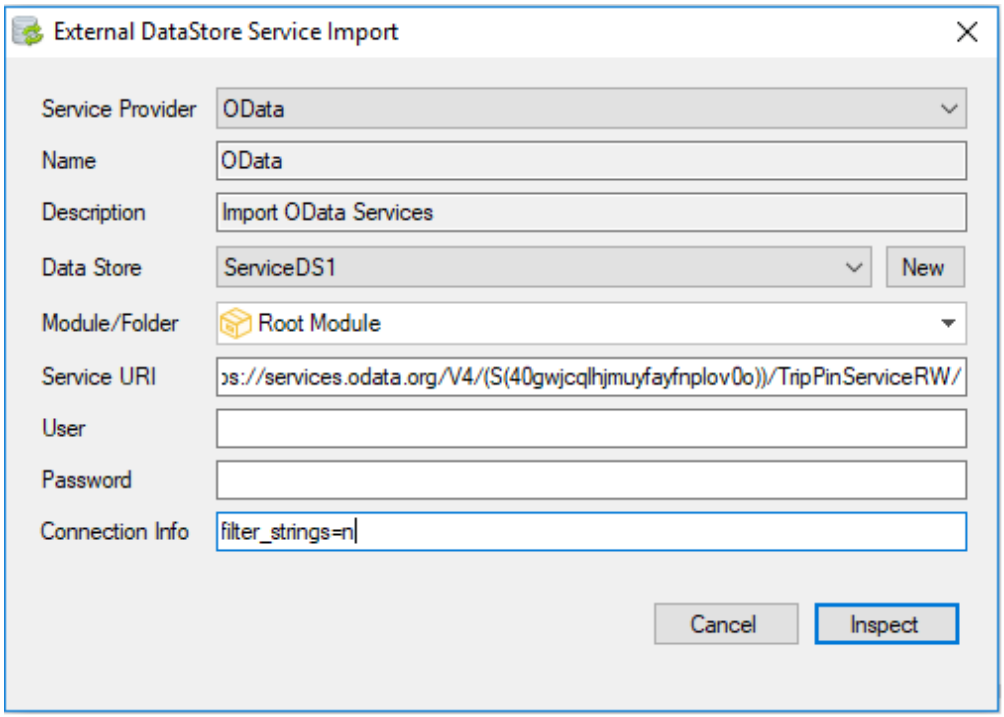
Será aberta uma nova janela onde inseriremos os dados para poder consumir o serviço.

A primeira coisa que devemos fazer é criar um novo Data Store de tipo Service, a partir do botão New localizado à direita da opção de DataStore:

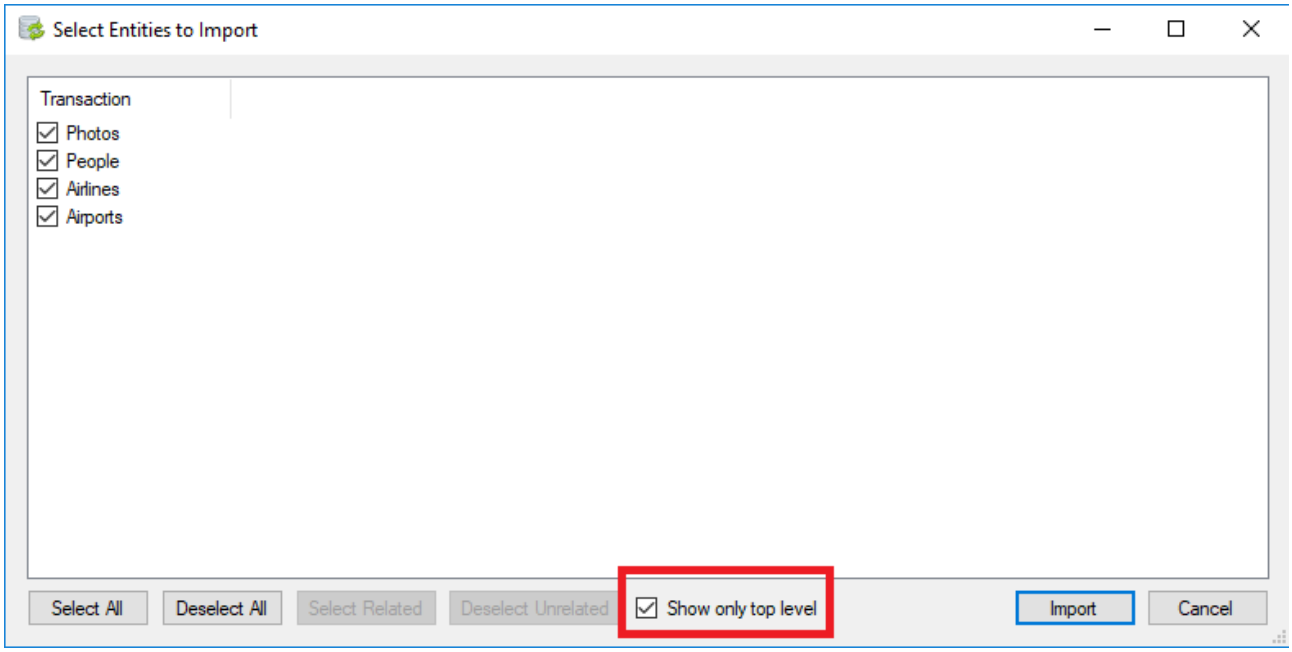


Então devemos preencher os campos:

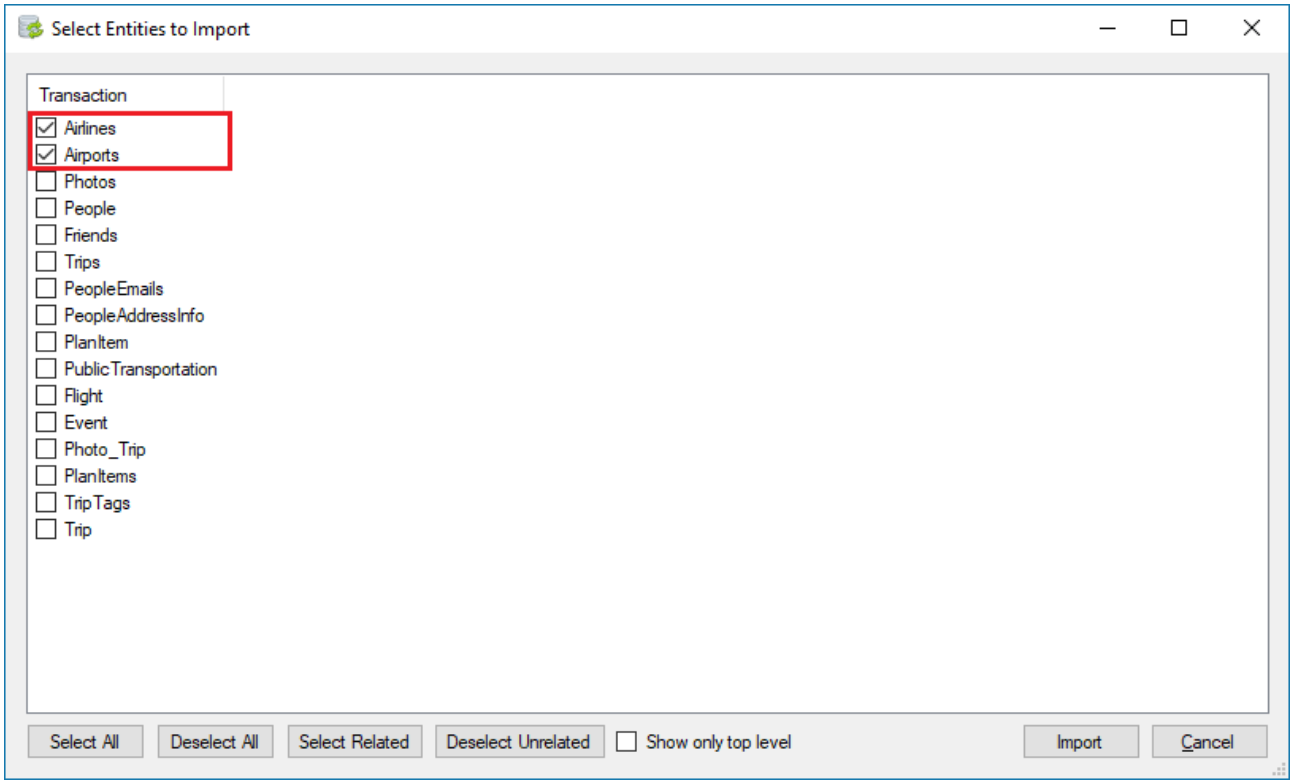
- Service URI: aqui devemos inserir o endereço do serviço a ser consumido. Neste caso, usaremos o serviço do TripPin: https://services.odata.org/V4/(S(40gwjcqlhjmuyfayfnplov0o))/TripPinServiceRW/
- User: aqui devemos inserir o nome de usuário, se necessário. Neste caso, não é necessário, portanto vamos deixá-lo vazio.
- Password: aqui devemos inserir a password do usuário. Neste caso, não é necessário, portanto vamos deixá-la vazia.
- Connection Info: aqui você insere informações extras para a conexão, neste caso é necessário que nós digitemos "filter_strings=n" para indicar que não aceita condições como <,>, <=,> = com dados do tipo String.



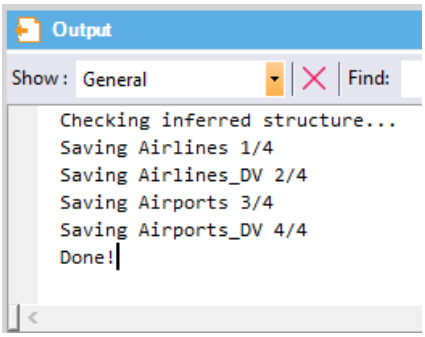
Depois de preencher os campos, clicamos em Inspect. Isso abrirá uma nova janela onde serão exibidas as entidades a serem importadas do serviço. Para este caso, precisaremos apenas de Airports e Airlines. Para isto, devemos desmarcar o checkbox "Show only top level":

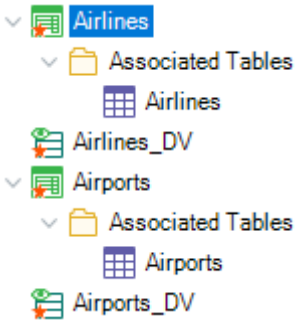


E deixar marcado apenas as opções de Airlines y Airports

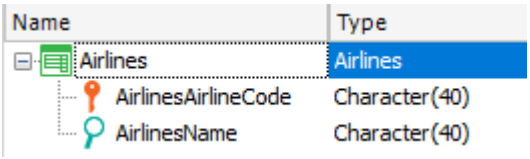


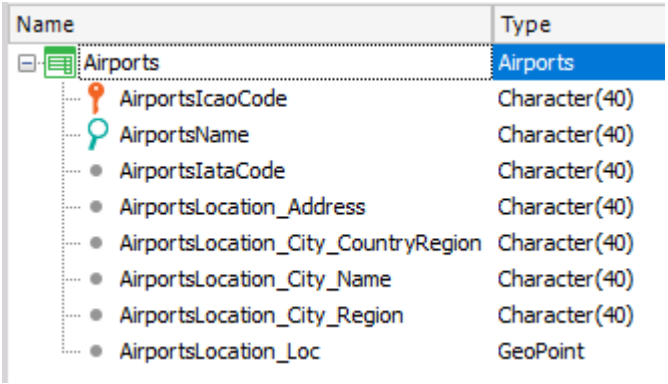
Feito isso, pressionaremos Import. Isto importará as entidades Airports e Airlines como transações, com os data views associados:





A estrutura das transações é a seguinte:



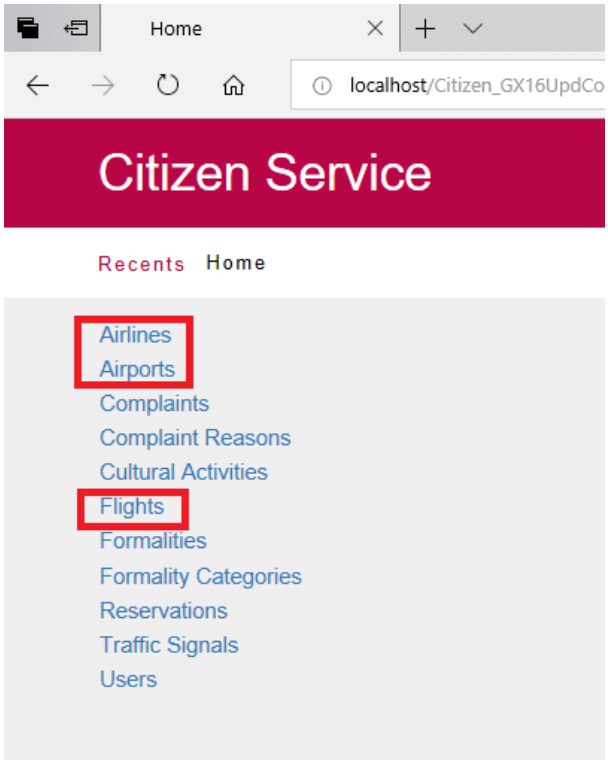


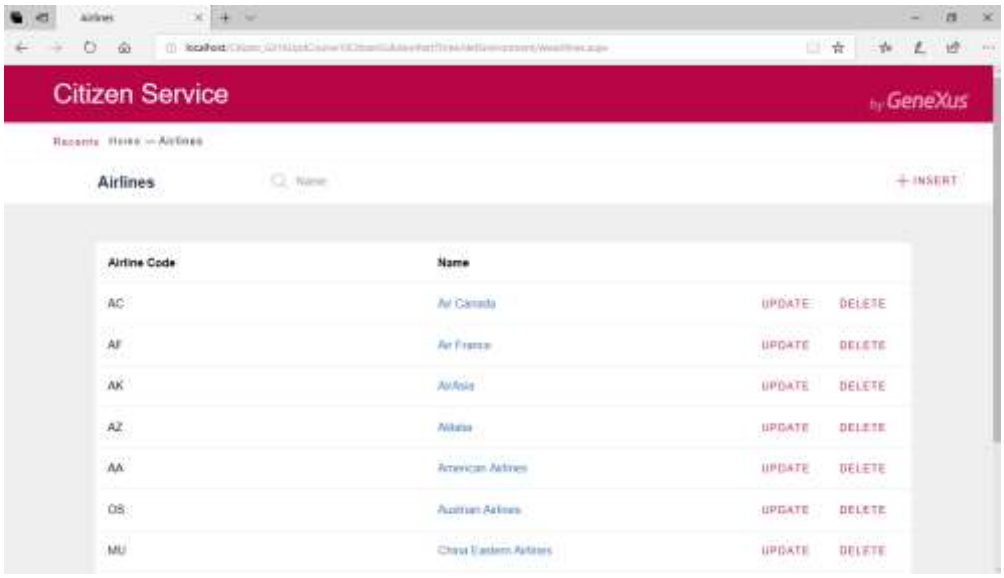
Recomendamos mover as entidades para a pasta GeneXus/Web/Backend_ODATA

UTILIZAR OS SERVIÇOS IMPORTADOS: ADMINISTRAÇÃO DE VOOS, AIRLINES, AIRPORTS

Depois de importar os dados, devemos usá-los para implementar as funcionalidades solicitadas.

Primeiro de tudo, queremos poder gerenciar (CRUD) os dados de linhas aéreas e de voos, adicionando-os ao backend.



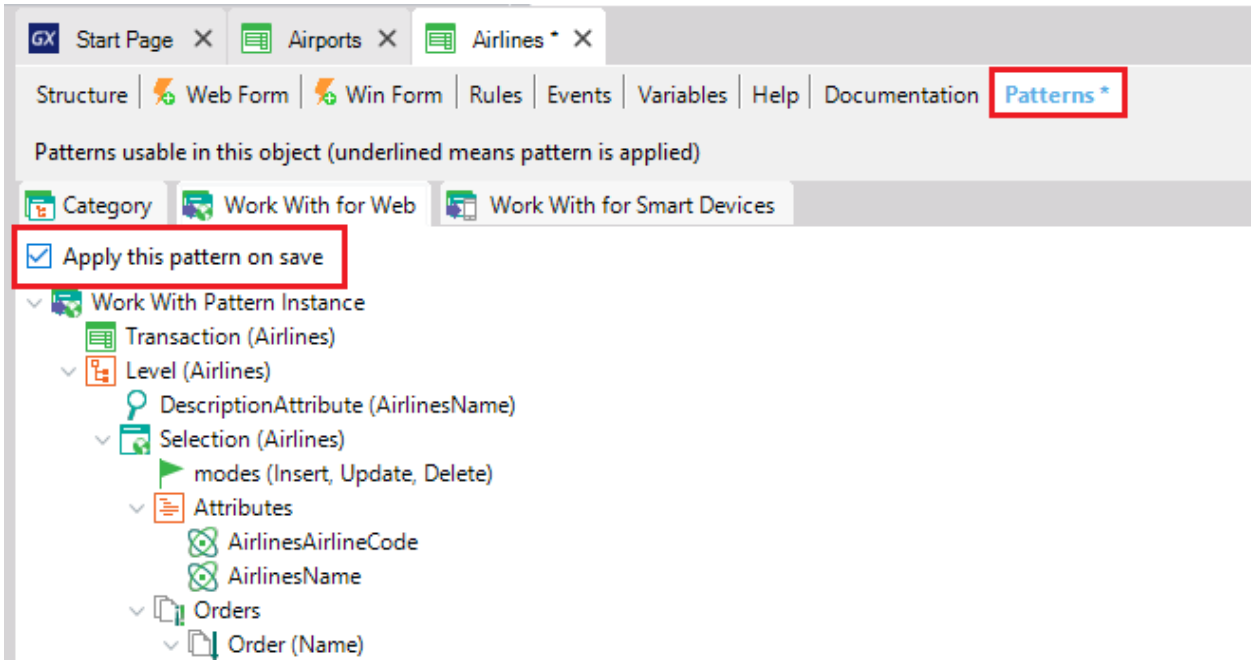


Nota: não podem ser inseridos ou excluídos aeroportos, pois o serviço prestado não permite.

Depois de implementado, tente alterar o nome de um aeroporto, por exemplo, e insira voos.

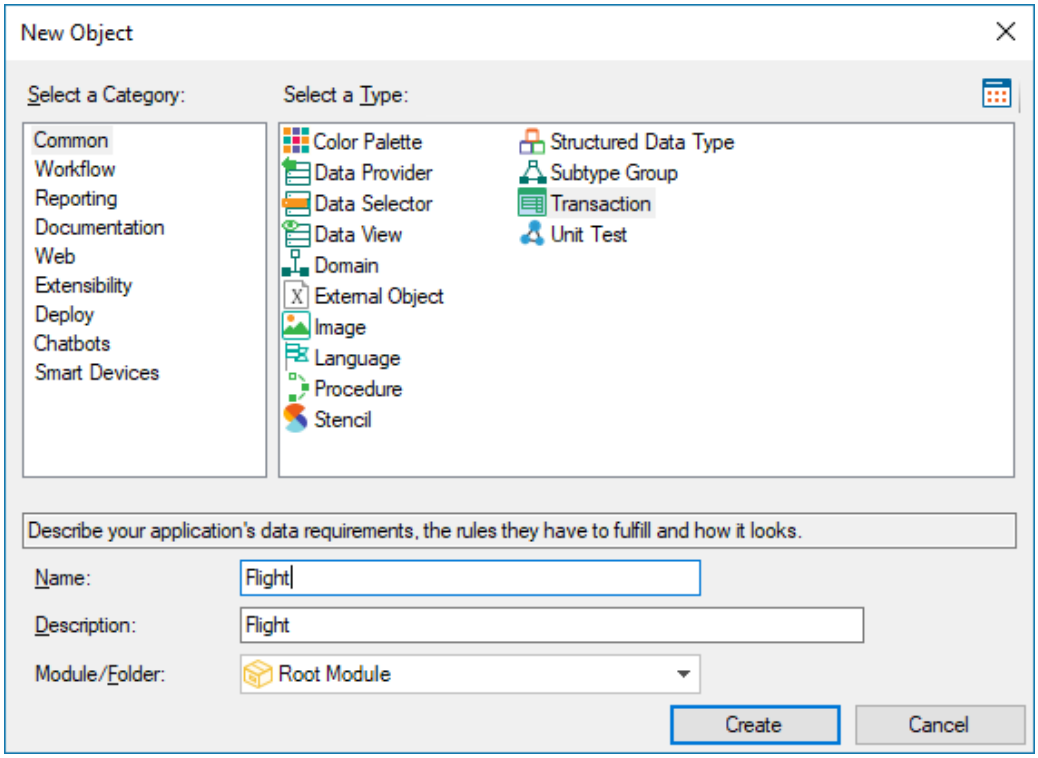
SOLUÇÃO POSSÍVEL

Aplicar o pattern Work With for Web às transações Airports e Airlines.

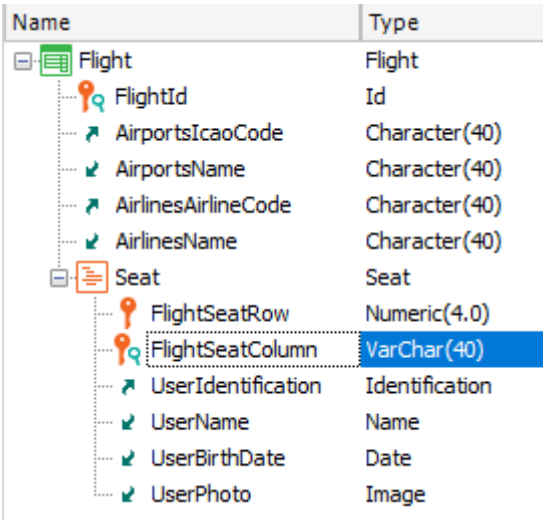


Com isso, serão automaticamente adicionadas ao backend.

Em seguida, criará uma nova transação Flight, que será responsável por manter o registro dos voos e se relacionará com Airports e Airlines::



A estrutura desta transação será:



A esta transação também aplicaremos o pattern Work With Web.

Com isto já conseguimos implementar o gerenciamento de voos.

Execute e visualize os dados, e insira alguns. Se desejar, você pode usar os seguintes dados de teste.

Voo 1:

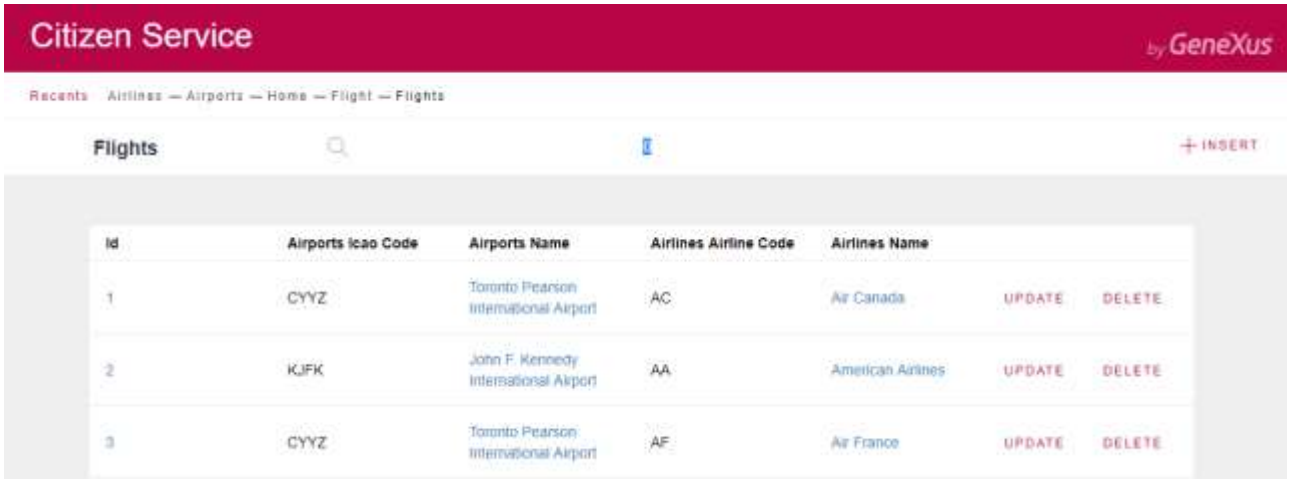
FlightId	1
Airport	Toronto Pearson International Airport
Airline	Air Canada
Seat Row 1 Column A	User Luciano Silveira
Seat Row 1 Column B	User Mary Smith
Seat Row 2 Column A	User Martin Torad
Seat Row 2 Column B	User Ann Bert

Voo 2:

FlightId	2
Airport	John F. Kennedy International Airport
Airline	American Airlines
Seat Row 1 Column A	User Rudolph Roball
Seat Row 1 Column B	User Cecilia Lemos
Seat Row 2 Column A	User Jessica Loyarte

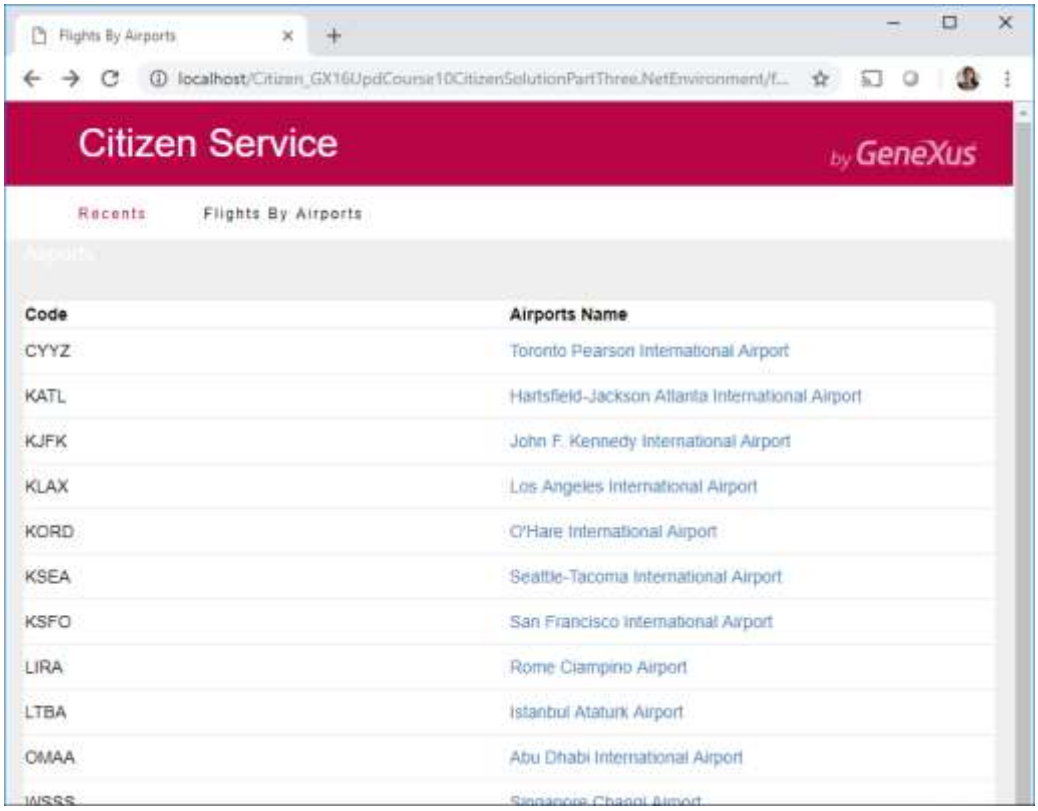
Voo 3

FlightId	3
Airport	Toronto Pearson International Airport
Airline	Air France
Seat Row 1 Column A	User Brian Goals
Seat Row 1 Column B	User John Peters
Seat Row 2 Column A	User Eleonor Johnson
Seat Row 2 Column B	User Valerie Molins



UTILIZAR OS SERVIÇOS IMPORTADOS: LISTA DE VOOS

Queremos mostrar em um painel todos os aeroportos de destino possíveis:



E escolhendo um, emitir um pdf com todos os voos com destino a esse aeroporto. Para cada voo, desejam listar as informações dos passageiros (users de Citizen):

Já lhe damos criado o web panel FlightsByAirports e o procedimento ListAirportFlight, para que você modifique o que precisa e não perca tempo. Eles estão localizados na pasta GeneXus/Web/Backend_ODATA.

SOLUÇÃO:

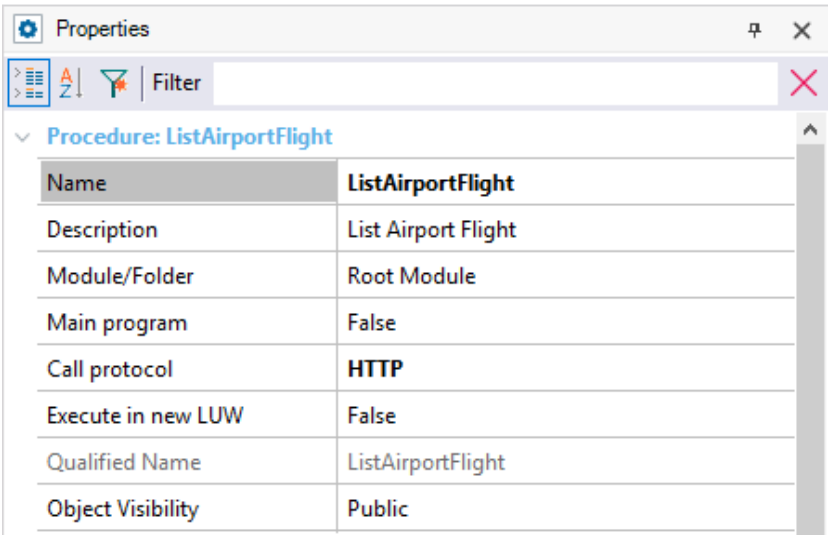
Utilizaremos uma procedure (já a criamos, é a de nome ListAirportFlight) para implementar a lista, que receberá por parâmetro o identificador do aeroporto, a fim de listar os voos que chegam a ele.

Na seção Rules tem:

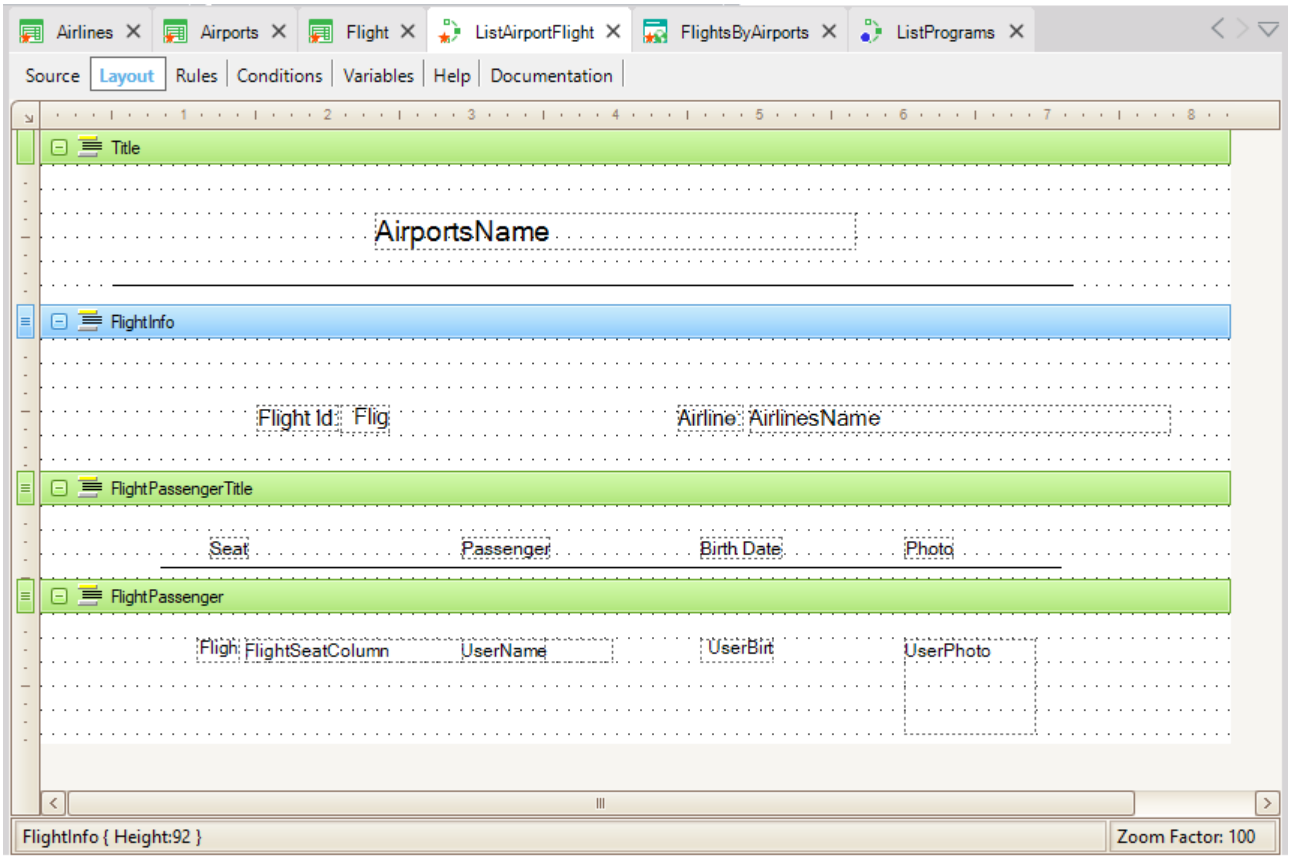
```
parm(in: &AirportCode);

Output_file("AirportFlightsList.pdf","pdf");
```

Onde `&AirportCode` é uma variável baseada no tipo de dados do atributo AirlinesAirlineCode (que é Character(40)). Lembre-se de que `Output_file` se utiliza para definir a saída da lista (em nosso caso um documento PDF).

Lembre-se também que teve que alterar a propriedade Call protocol para HTTP:



O layout deveria ficar mais ou menos assim:



E a seção Source:

```
For each Airports
	where AirportsIcaoCode = &AirportCode
	print Title
	For each Flight
		print FlightInfo
		print FlightPassengerTitle
		For each Flight.Seat
			print FlightPassenger
		endfor
	endfor

endfor
```

Invocamos este procedimento a partir do Web Panel FlightsByAirports, onde inserimos um Grid com os atributos AirportsIcaoCode e AirportsName.

Na seção de Events:

```
Event AirportsName.Click
	ListAirportFlight( AirportsIcaoCode )
endevent
```

Para ligar este painel ao backend (ao web panel Home), editamos o procedimento ListPrograms e adicionamos o seguinte código:

```
&name = !" FlightsByAirports"
&description = " Flights By Airports"
&link = FlightsByAirports.Link()

Do 'AddProgram'
```

Executamos a aplicação.

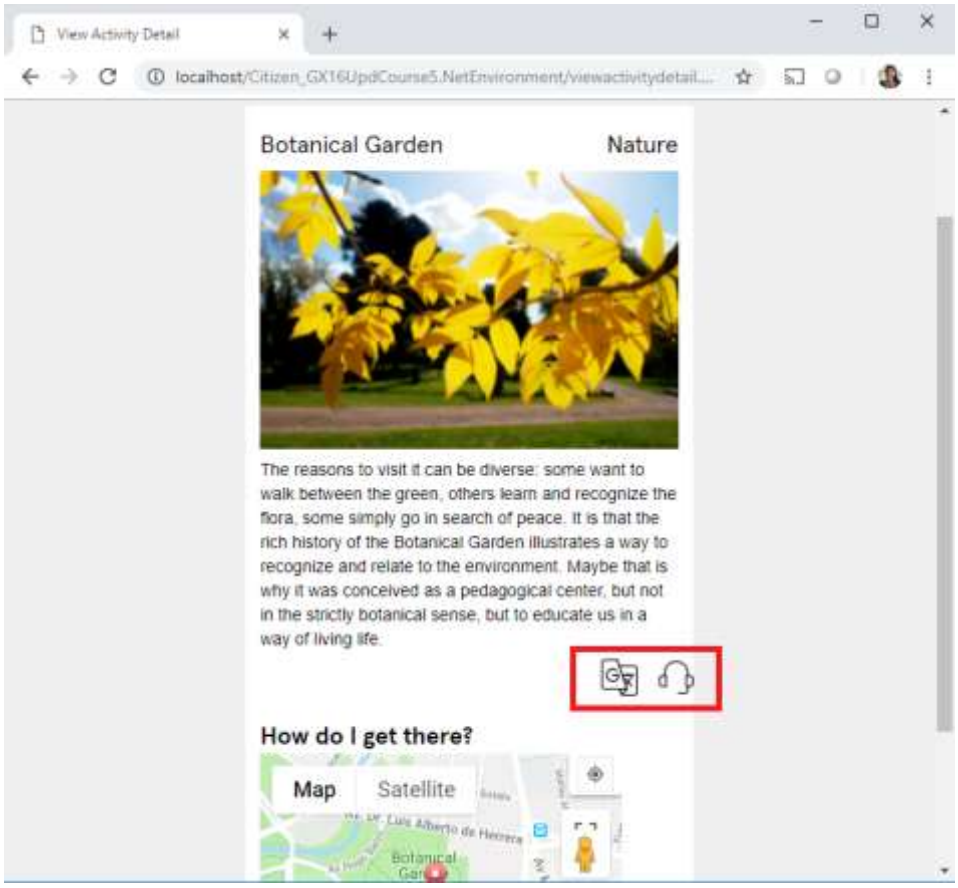
GENEXUS AI

Deseja-se adicionar duas novas funções à aplicação para poder:

1. traduzir a descrição de uma determinada atividade cultural para o idioma do dispositivo, e
2. permitir que a aplicação leia esta descrição em voz alta.

Para isso, no painel ActivityDetail do frontend SD, temos dois botões onde a funcionalidade será adicionada:

O mesmo queremos no painel ViewActivityDetail análogo, do frontend web:



PASSOS PRÉVIOS

ADICIONAR REFERÊNCIA AO MÓDULO GENEXUSAI

Antes de começarmos a adicionar a funcionalidade, devemos adicionar a referência ao módulo GeneXusAI.

Fazemos isto na opção Knowledge Manager > Manage Module References

MUDAR O IDIOMA DO SIMULADOR

Se o simulador estiver em inglês, mude para outro idioma para poder executar os testes.

Dica: para alterar o idioma do simulador Android, vá para Settings > System >Languages & input > toque na opção Languages > adicione o novo idioma, por exemplo, Español (Estados Unidos) e posicione-o no topo da lista.

TRADUÇÃO AUTOMÁTICA DE TEXTO

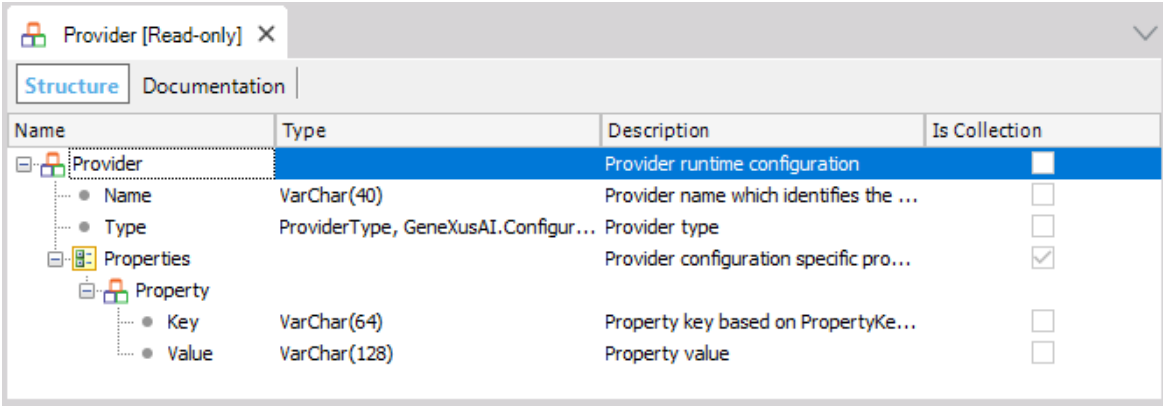
Implemente a tradução da descrição no painel para SD ActivityDetail (então você pode repeti-la no web panel análogo ViewActivityDetail).

Uma vez programado o evento, esperamos ver o texto traduzido como visto abaixo:

O serviço que vamos usar para tradução é o do GeneXusAI.Text.Translate. Para isto, devemos configurar o Provider com Type = ProviderType.IBM e o seguinte par de chave-valor:

- Key: PropertyKey.Key; Value: euY0twBoSB4qs8jDj4gvuxVlWt9soLHOKu5O8lYv5o3p

Dica: Lembre-se de que isto é feito em um procedimento. Uma pasta AI foi criada para você o crie lá. E isto é feito com base no SDT predefinido, que vem dentro do módulo GeneXusAI/Configuration:



Dica: o idioma de origem será sempre o inglês, mas se você quiser, você também pode incorporar a função de GeneXusAI.Text.DetectLanguage.

Dica: o idioma de destino é aquele do dispositivo, que pode ser obtido usando ClientInformation.Language em SD (que deve então ser convertido para os códigos de idioma usados por GeneXusAI).

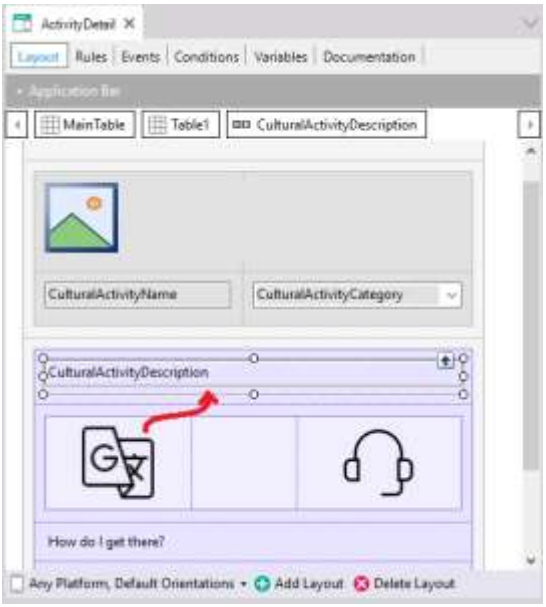
Nota: Em Web, é mais difícil obter o idioma do cliente, portanto o idioma de destino pode ser deixado fixo.

SOLUÇÃO

Precisaremos de um procedimento que chamaremos de TranslationProvider para configurar o provedor.

```
// Rules
parm(out:&provider);

// Source
&provider.Name = !'translate'
&provider.Type = ProviderType.IBM
&prop = new()
&prop.Key = PropertyKey.Key
&prop.Value = !'euY0twBoSB4qs8jDj4gvuxVlWt9soLHOKu5O8lYv5o3p'
&provider.Properties.Add(&prop)

// Variables
&provider : Provider data type
&prop: Provider.Property data type
```

Então vamos para o painel ActivityDetail:



Devemos primeiro alterar na tela o atributo CulturalActivityDescription por uma variável (para poder alterá-la pelo texto traduzido) que carregamos no evento `Refresh`.

```
Event Refresh
	&CulturalActivityDescription = CulturalActivityDescription
EndEvent
```

Para a tradução do texto, devemos programar o evento ImageTranslate.Tap::

```
Event ImageTranslate.Tap
	composite
		&provider = TranslationProvider()
		&languageStr = ClientInformation.Language
		&languageStr = substr(&languageStr, 1, 2)
		&language.FromString(&languageStr)
		&CulturalActivityDescription = Translate(&CulturalActivityDescription,
Language.English, &language, &provider, &messages)
	endcomposite

Endevent
```

Onde ClientInformation é o EO que vem por padrão com o módulo GeneXUs, submódulo Client. Se você olhar o tipo de dados de ClientInformation.Language, verá que é C(20), portanto, o tipo de dados da variável &languageStr terá que ser desse tipo.
E o tipo de dados retornado pela proc Translate é Text, e o de &language é Language e &messages é do tipo Messages.

LEITURA DE TEXTO EM VOZ ALTA

Agora queremos implementar a tradução da descrição da atividade cultural em áudio, e reproduzi-la no dispositivo no momento. Lembre-se que para reproduzir um áudio temos o EO Audio.

O serviço que vamos usar nesta parte é o do GeneXusAI.Audio.TextToSpeech. Para isto, devemos configurar o Provider com Type = ProviderType.IBM e os seguintes pares de chave-valor:

- Key: PropertyKey.Username; Value: 9e579369-9b47-4c8e-84ec-122affee5ae1
- Key: PropertyKey.Password; Value: 5BiwVDLGd0H4

SOLUÇÃO

Para usar a função text to speech, devemos implementar o evento `ImageSpeak.Tap`:

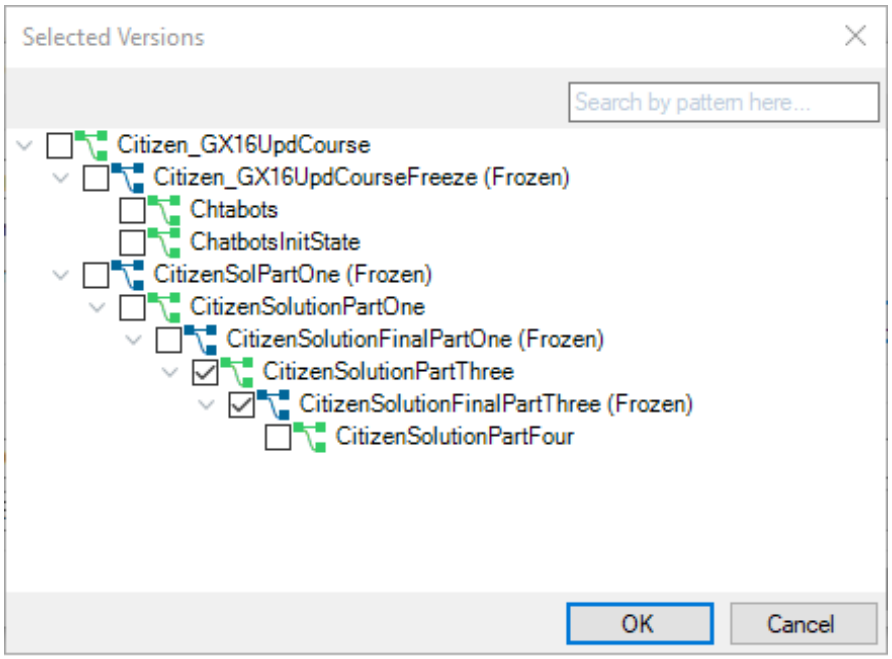
```
Event ImageSpeak.Tap
	composite
		&provider = TextToSpeechProvider()
		&localeStr = ClientInformation.Language
		&localeStr = substr(&localeStr, 1, 5)
		&locale.FromString(&localeStr)
		&audio = TextToSpeech(&CulturalActivityDescription, VoiceType.Female,
&locale, &provider, &messages)
		Audio.PlayBackground(&audio)
	endcomposite

Endevent
```

Onde `TextToSpeechProvider` é uma Procedure com o seguinte código:

```
// Rules
parm(out:&provider);


// Source
&provider.Name = !'textToSpeech'
&provider.Type = ProviderType.IBM

&prop = new()
&prop.Key = PropertyKey.Username
&prop.Value = !'9e579369-9b47-4c8e-84ec-122affee5ae1'
&provider.Properties.Add(&prop)

&prop = new()
&prop.Key = PropertyKey.Password
&prop.Value = !'5BiwVDLGd0H4'
&provider.Properties.Add(&prop)
```

KB SOLUÇÃO

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



Confirme que no Data Store “ServiceDS1 (Service)” a propriedade de conexão Server name tenha ficado assim: https://services.odata.org/V4/(S(40gwjcqlhjmuyfayfnplov0o))/TripPinServiceRW/

E não vazia. Se estiver vazia, copie o valor anterior.

Foi ativada a propriedade “Data provider” da transação "Flight" para popular a tabela com dados iniciais, assim não precisa fazer nada além de executar para testar as funcionalidades.