

# Curso prático de atualização para

# GeneXus™ 16

## PARTE 2

Abril 2020

*Copyright © GeneXus S.A. 1988-2020.*

*All rights reserved. This document may not be reproduced by any means without the express permission of GeneXus S.A. The information contained herein is intended for personal use only.*

**Registered Trademarks:**

*GeneXus is trademark or registered trademark of GeneXus S.A. All other trademarks mentioned herein are the property of their respective owners.*

## CONTEÚDO

CONTEÚDO.....	2
OBJETIVO.....	3
ALGUM AJUSTE PRÉVIO .....	3
VAMOS COMEÇAR.....	4
<b>Quais são os conceitos básicos por trás de um chatbot?.....</b>	<b>4</b>
O CHATBOT EM AÇÃO.....	11
INTRODUZINDO MELHORAS NO CHATBOT .....	15
<b>Fazer uma reclamação.....</b>	<b>16</b>
Resumo .....	21
<b>Opcional: Ver as atividades culturais .....</b>	<b>22</b>
ANEXO: TRY LIVE .....	30
Lembre-se.....	31
DOCUMENTAÇÃO .....	32
KB SOLUÇÃO .....	32

## OBJETIVO

Veremos o poder do gerador de chatbots introduzido na versão GeneXus 16.

Criaremos um chatbot para a realidade de uma aplicação que fornece serviços ao cidadão. Será um chatbot tanto para a aplicação WEB quanto para a aplicação SD desta realidade (faremos isso no Android pela facilidade que oferece na prototipação).

Usaremos Watson como Natural Language Processing (NLP) Provider. Mas poderia muito bem ser o Dialogflow, sem nenhum problema.

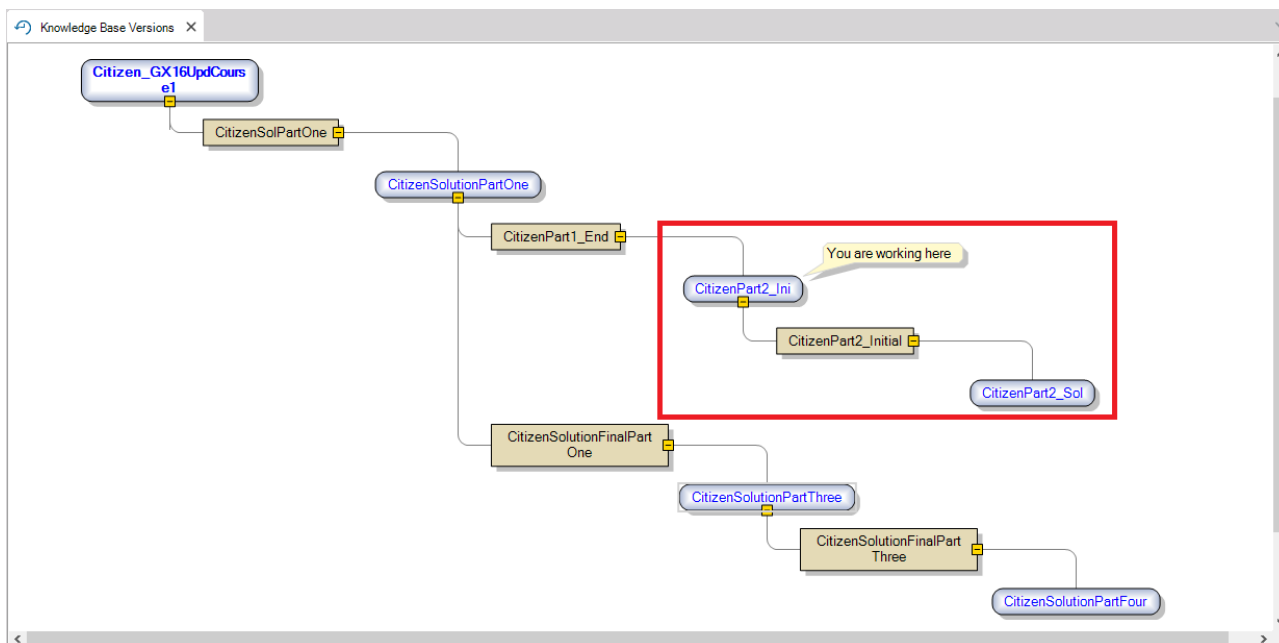
Através do nosso chatbot os usuários serão capazes de:

- Fazer uma reclamação por qualquer problema ocorrido na rua; pode ser uma árvore caída, um sinal de trânsito que falta ou deteriorado, um carro mal estacionado, etc.
- Obter informações sobre atividades culturais desenvolvidas, ou dos requisitos para procedimentos administrativos.

## ALGUM AJUSTE PRÉVIO

Ignoramos a configuração da conta no NLP provider, dado que para este prático já foi feito. O instrutor fornecerá as informações necessárias quando precisar.

Será baseado na versão **CitizenPart2\_Ini** da KB **Citizen\_GX16Course** em <http://samples.genexusserver.com/v16>.



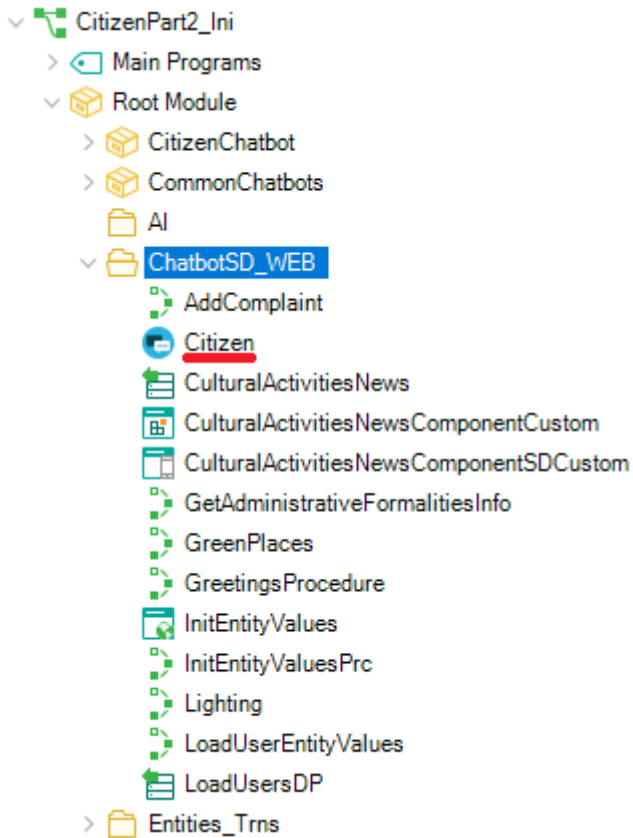
Esta versão da KB vem com pastas, objetos e módulos já criados para não perder tempo.

## VAMOS COMEÇAR

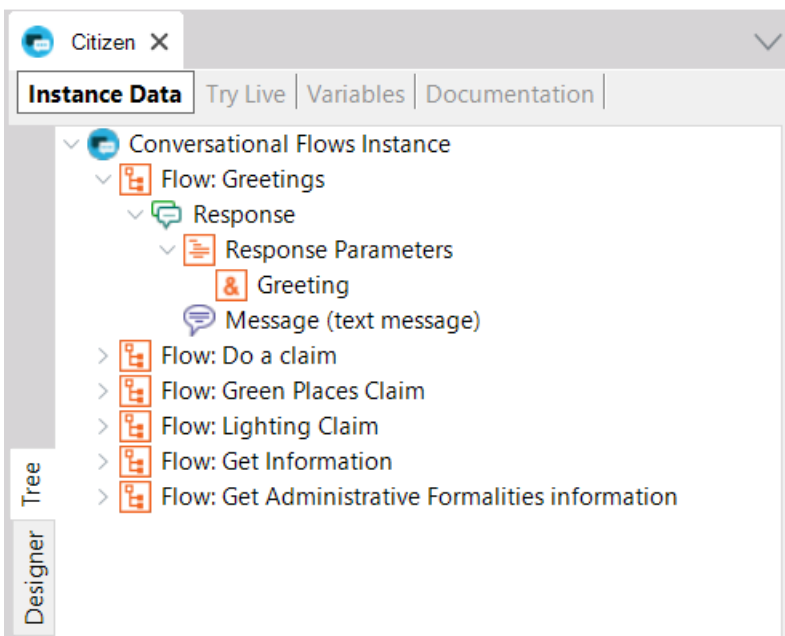
### QUAIS SÃO OS CONCEITOS BÁSICOS POR TRÁS DE UM CHATBOT?

Um chatbot é representado por um Objeto da KB, novo no **GeneXus 16**, chamado **Conversational Flows**.

Na KB, dentro da pasta ChatbotSD\_WEB, já criamos um objeto Conversational Flows, que nos servirá de base para construir novas funcionalidades para nossa solução de chatbots.

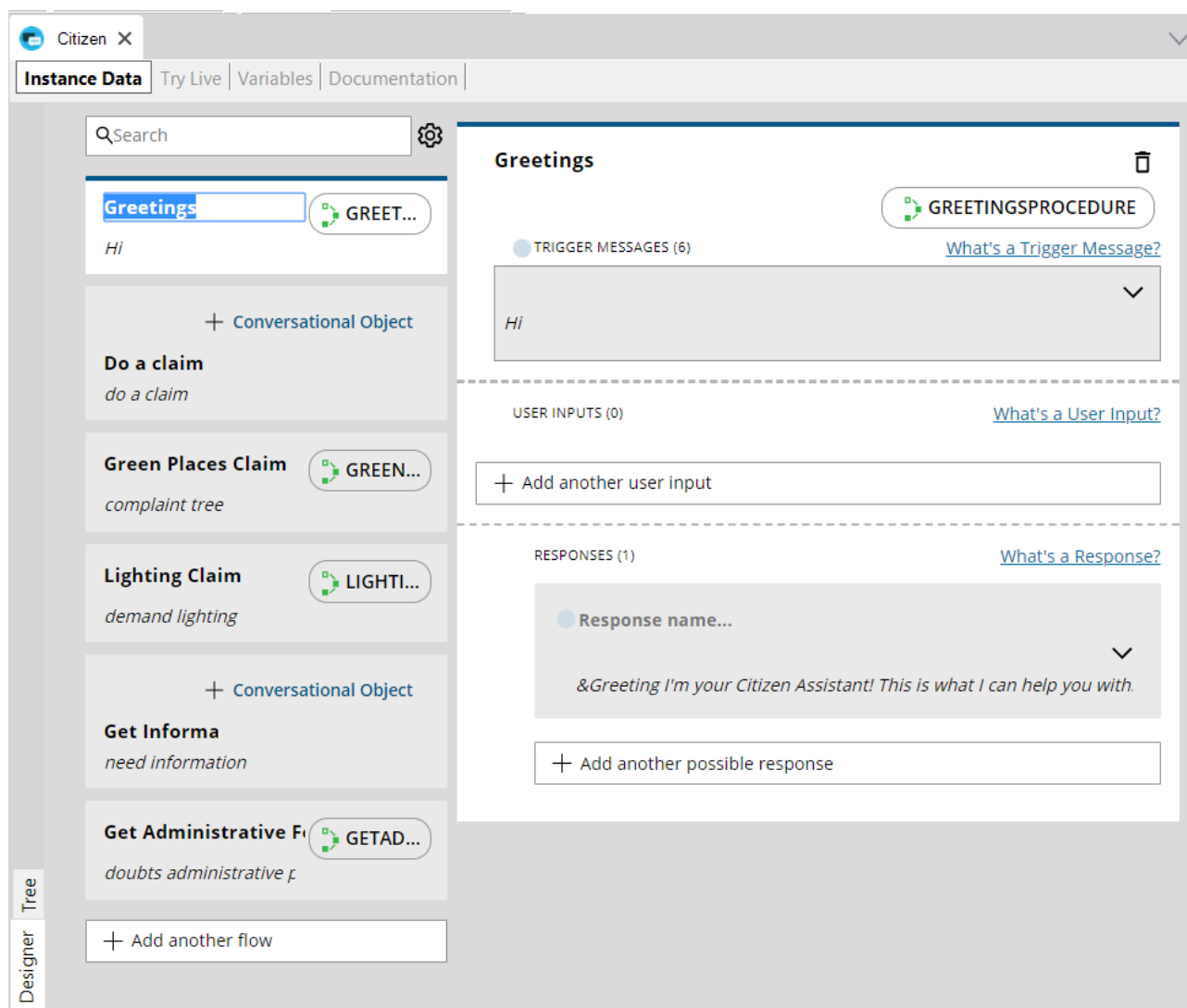


Abra-o e verá em que consiste o desenho de um chatbot no GeneXus: Por enquanto, possui apenas a visualização Tree:



Tem uma estrutura de árvore, onde cada nó principal, “Flow”, representa uma intenção do usuário final, que nosso chatbot saberá como responder adequadamente.

Na versão beta, já temos uma visualização Designer, que estará disponível em breve:



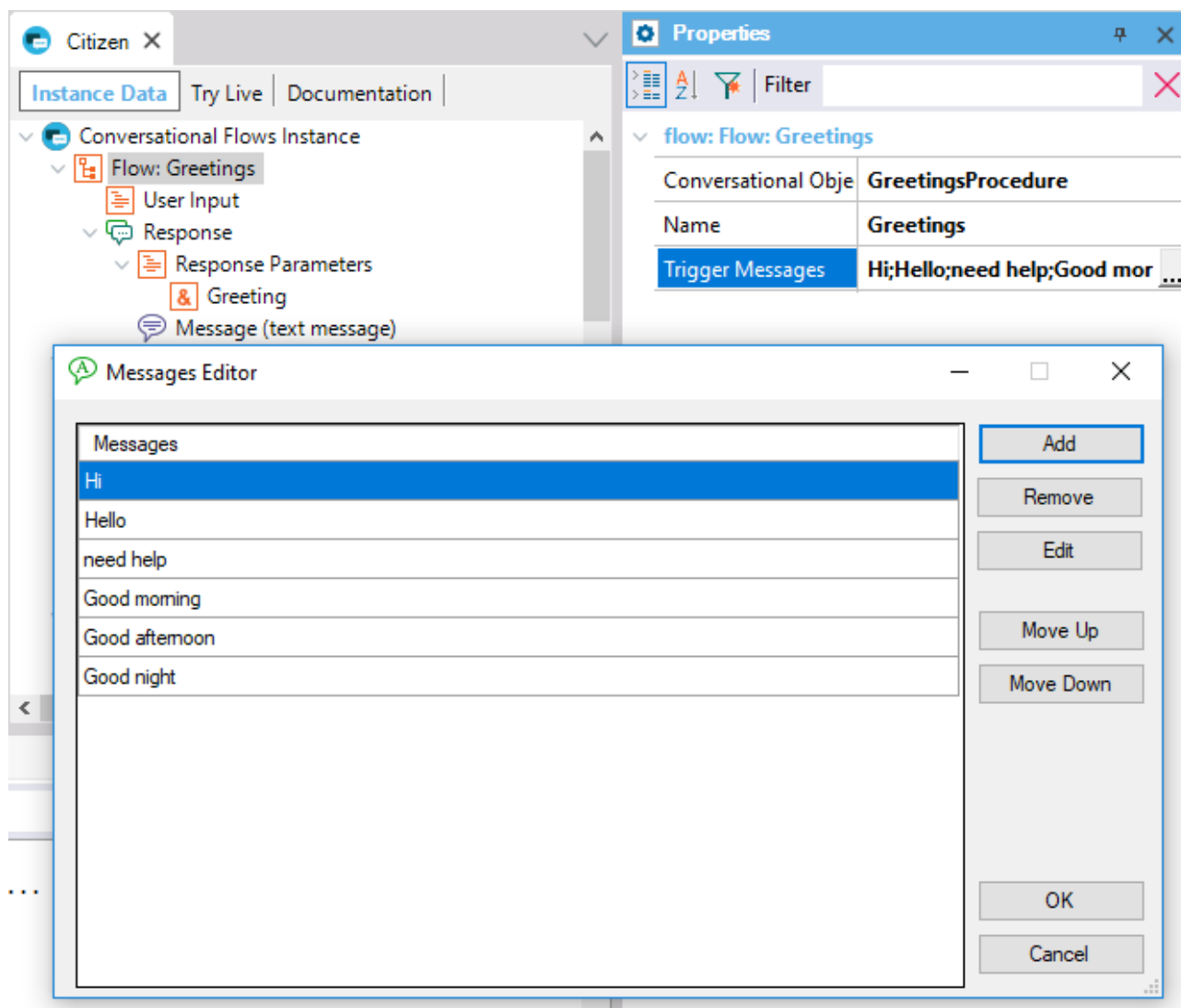
O que é uma *intent*? Reflete a vontade do usuário ao fazer a consulta, por exemplo “Fazer uma reclamação por um carro mal estacionado”.

No diagrama da Conversational Flows Instance, o Flow tem uma relação direta com a Intent.

Como o NLP provider interpreta a intenção do usuário?

Faz isso através das **Trigger messages** que adicionamos no Flow; quanto mais completas, melhor poderá fazer a inferência.

Observe no “Flow: Greetings” a propriedade **Trigger Messages**, que possui várias opções de saudação:

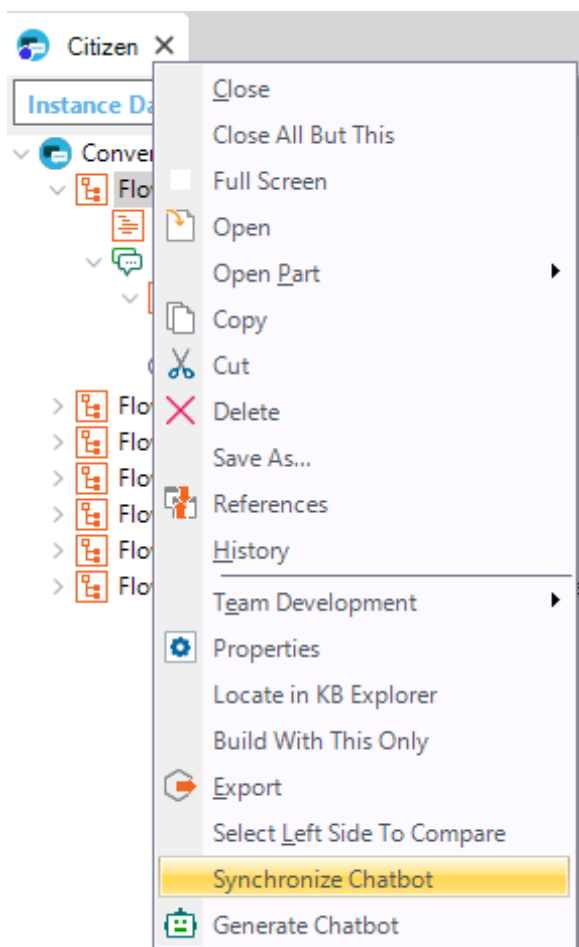


Este chatbot que temos na KB ainda não foi sincronizado com o Provider.

No objeto Citizen, no nó pai “Conversational Flows Instance”, configuraremos nas propriedades User name e User password os valores que o instrutor deve lhe fornecer.

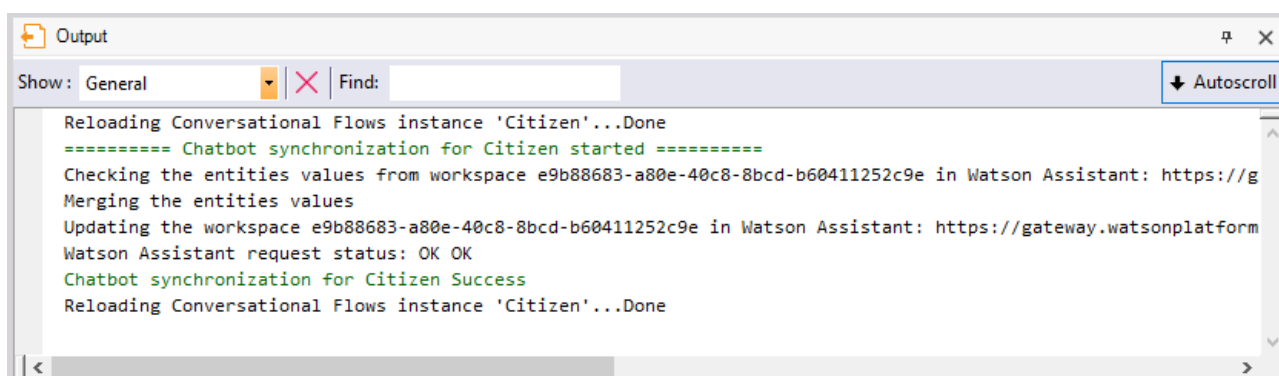
Salvemos o objeto e vamos ver como o diálogo é criado, com suas intents e Trigger messages no Watson.

Clicando com o botão direito sobre o objeto, podemos selecionar a opção Synchronize Chatbot:



Esta ação gera um arquivo de metadados que se impacta no NLP provider, para gerar o diálogo com suas intents e entidades.

Podemos ver na aba General de output do GeneXus, a saída da execução da opção Synchronize.



O que acontece se adicionarmos uma Trigger message ao Flow “Greetings”?

Faz uma alteração ou insere uma nova Trigger Message no Flow Greetings. Só com salvar o objeto Conversational Flows se impacta a alteração no Watson.



Dada uma intenção do usuário se podem distinguir diferentes objetos ou tópicos dentro dessa intenção, que formalmente chamamos Entidades.

Por exemplo, dada a intenção de agendar data e hora, uma entidade dessa intenção seria o **motivo** para agendar data, e o valor dessa entidade, por exemplo, pode ser o **processo de renovação** do documento de habilitação.

Assim, chamamos de entidade um objeto da consulta, que agrupa vários valores, e um ou vários deles serão instanciados na query.

As entidades são armazenadas no NLP provider, com seus possíveis valores e sinônimos.

[Workspaces](#) / CitizenChatbot / Build

Intents **Entities** Dialog Content Catalog

**My entities** System entities

[Add entity](#) ↑ ↓ 🗑️

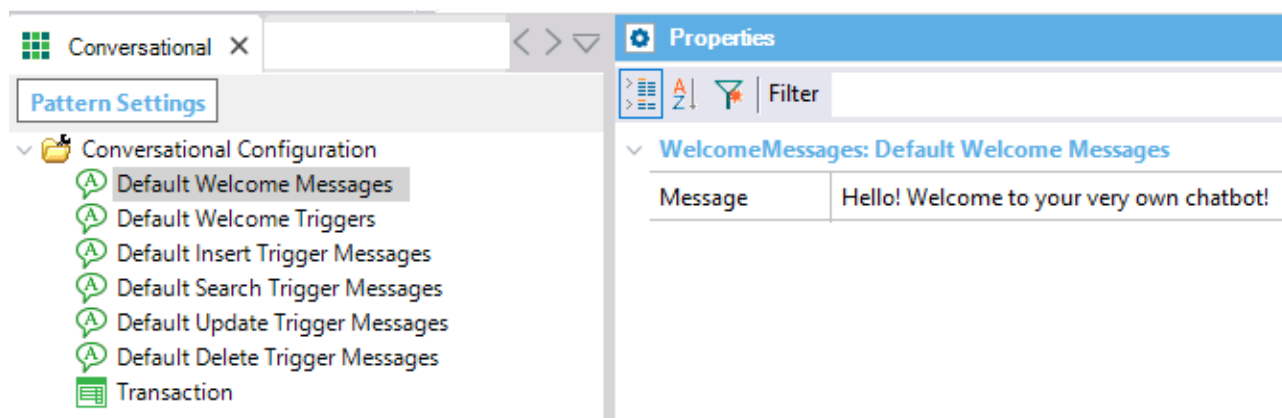
<input type="checkbox"/> Entity (6) ▼	Values
<input type="checkbox"/> @Activities	Art, Nature, Culture
<input type="checkbox"/> @AdmProcessType	Driver License first time, Enable advertisements, Debt Refinancing, Driver License Renewal
<input type="checkbox"/> @Claim_Type	Sanitation, Traffic, Lighting
<input type="checkbox"/> @GreenPlaces	tree, sanitation, street, cleaning
<input type="checkbox"/> @InformationType	Activities, Administrative Process
<input type="checkbox"/> @UserIdentification	67890, 12345

Para criar um chatbot, é necessário modelar seu comportamento (definir intents, entidades) e a resposta que é mostrada ao usuário para cada uma dessas intents.

Esse modelo é feito no objeto conversational flows (no nosso caso, Citizen).

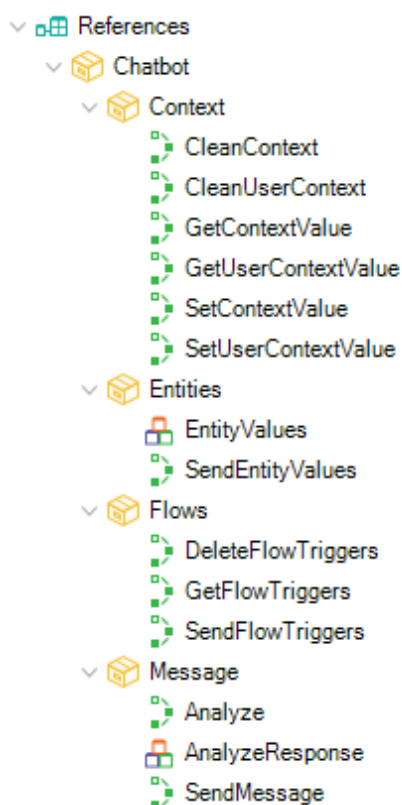
Nota: O gerador de chatbots funciona como um pattern. A partir do modelo, o pattern gera automaticamente todos os objetos necessários para resolver a conversa entre o usuário e o NLP provider.

Vejamos nas Preferences, abaixo de Patterns, os Conversational Flows e suas propriedades.

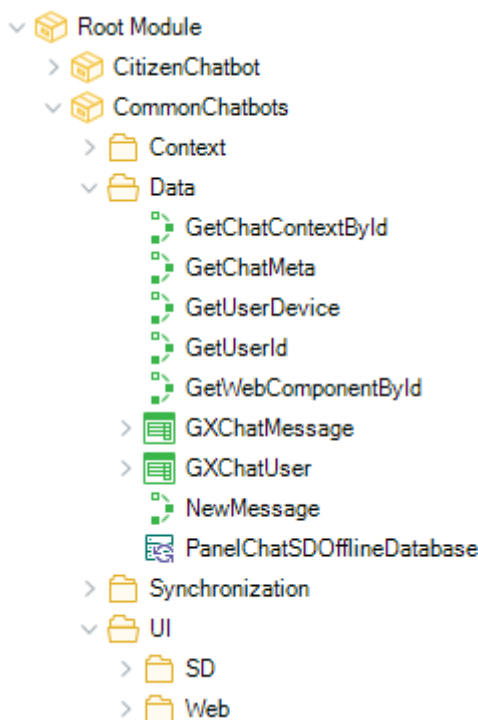


Os componentes dentro da KB são os seguintes:

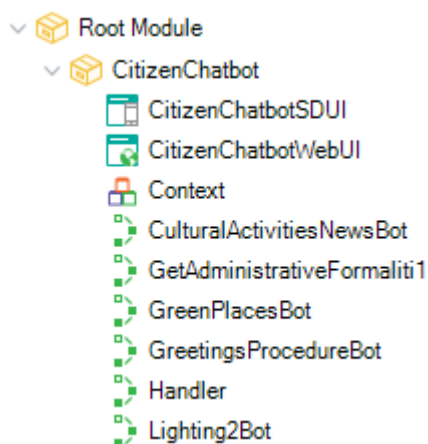
(1) Módulo Chatbots. É um módulo externo que se instala na KB e resolve a comunicação com o Provider. Verifique com o instrutor se é conveniente ou não atualizar para a versão mais recente (indo ao Knowledge Manager/Manage Module References):



(2) Recursos do pattern. São objetos de exemplo para a criação do nosso chatbot. Podemos modificá-los a gosto. Se encontram no módulo CommonChatbots.



<sup>(3)</sup> Objetos gerados. A partir de nosso modelo, estes objetos são gerados em um módulo que contém o nome do objeto Conversational Flows (CitizenChatbot), e servem de ligação com os objetos de nossa KB e o servidor.



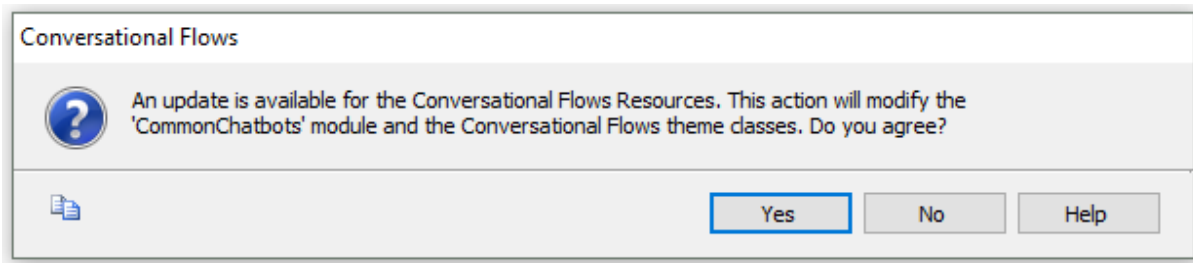
Já entendemos os principais conceitos, e o que é o Chatbot Generator, agora vamos começar a construir!

## O CHATBOT EM AÇÃO

Clicamos com o botão direito do mouse sobre o objeto Citizen e, em seguida, em Generate Chatbot.

**Nota:** A opção Generate Chatbot faz com que sejam calculados os objetos que gera o pattern <sup>(3)</sup>, e se importem na KB.

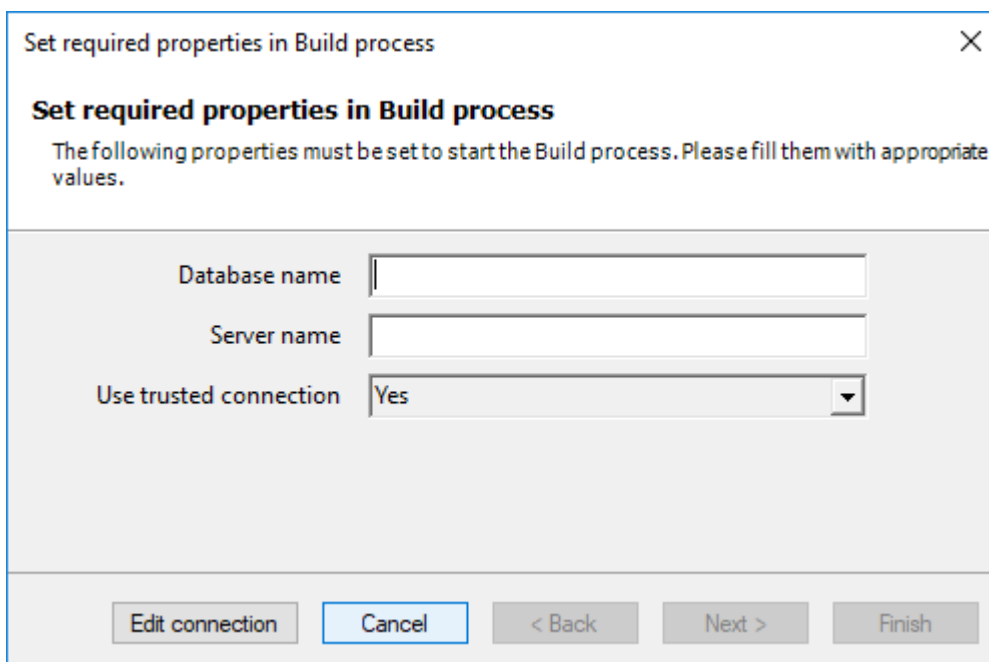
Veja a seguinte mensagem, que indica que se atualizarão os recursos<sup>(2)</sup> do gerador de chatbots.



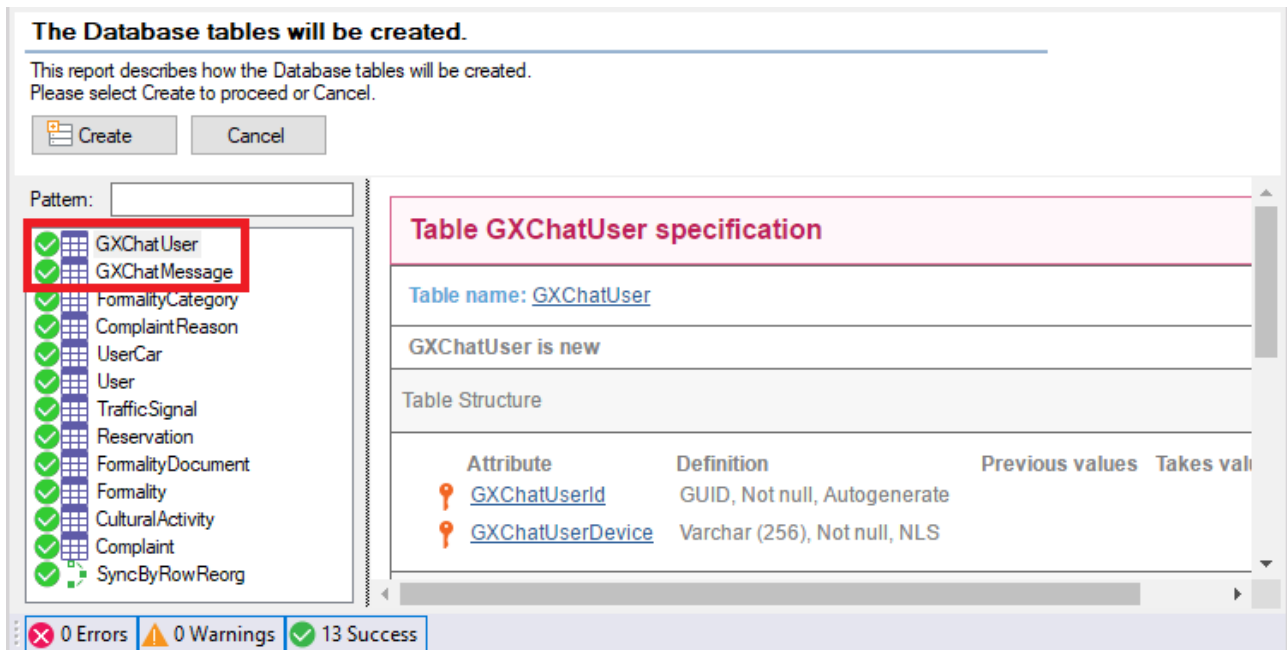
Observar na output que são importados os recursos para Web e SD do gerador de chatbots.

Façamos um **rebuild all**. O build também gera os objetos do Pattern quando é necessário (se forem detectadas alterações na instância).

Será solicitada a criação da BD. Digite o Server de BD correspondente ao SQLServer local (estamos prototipando com um ambiente .Net).



Note que existem duas tabelas, chamadas GXChatMessage e GXChatUser, que são usadas pelo chatbot.



Trabalharemos com as seguintes entidades, que você pode ver na Interface do usuário para IBM watson assistant”:

- Entity (6) ▼
- @Activities
- @AdmProcessType
- @Claim\_Type
- @GreenPlaces
- @InformationType
- @UserIdentification

Serão carregadas no NLP provider por meio de uma API fornecida pelo módulo chatbots.

Abra o objeto InitEntityValues para conhecer o método usado para carregar valores e entidades para o Watson (Chatbot.SendEntityValues).

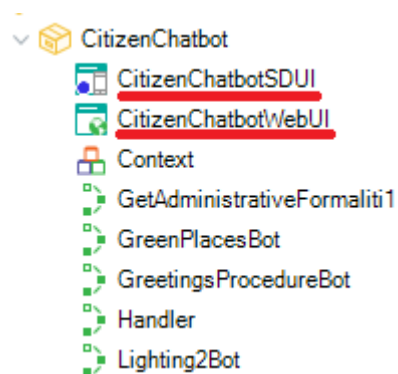
Isto sempre faz um merge com o que temos no NLP provider.

Em seguida, execute o webpanel InitEntityValues e pressione o botão Initialize Entity Values.

O resultado da execução será:



Agora executemos os seguintes objetos para ver o chatbot em runtime.



Aplicação WEB: Execute o objeto main CitizenChatbotWebUI

Aplicação SD: Execute o objeto main CitizenChatbotSDUI

**Nota:** Estes objetos chamam nosso painel principal de chatbots, passando como parâmetro o Provider e o nome da instância.

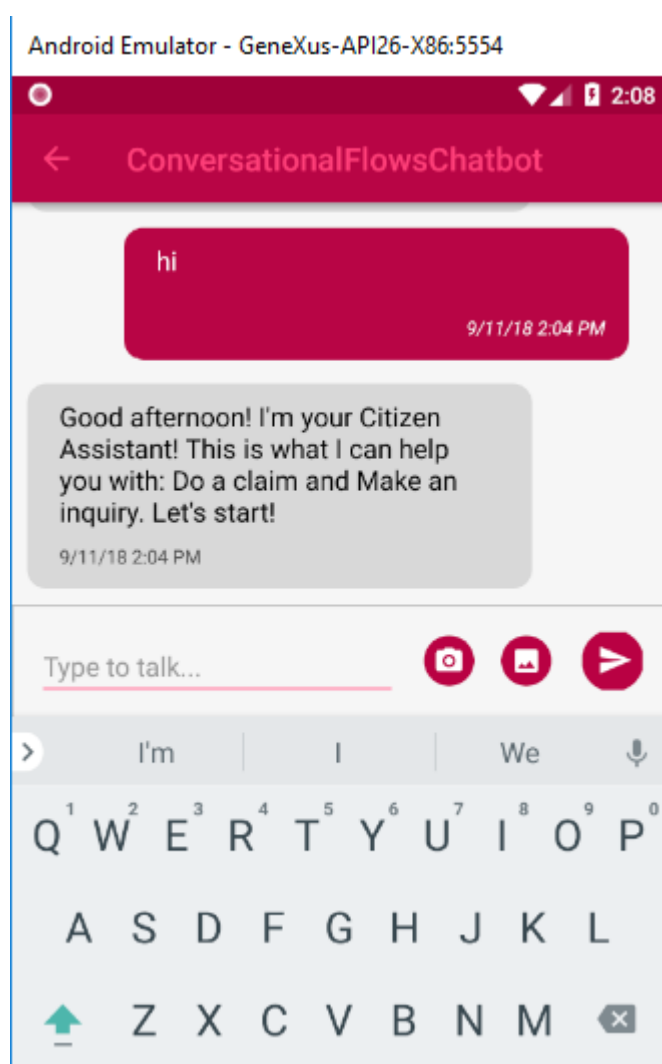
```
Event 'ClientStart'
```

```
CommonChatbots.PanelChatSD.Call(!"Citizen")
```

```
EndEvent
```

Já pode iniciar um diálogo com o chatbot Web e SD. Comece cumprimentando o chatbot.

Em SD por exemplo:



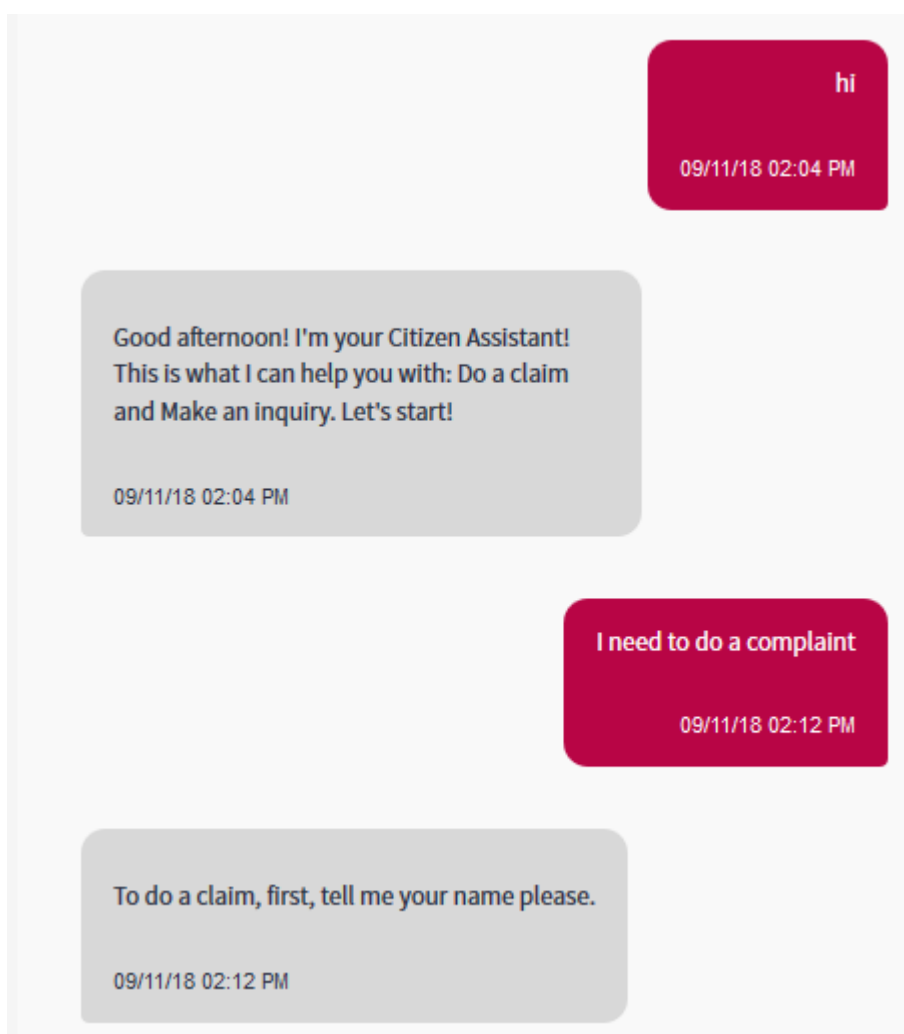
## INTRODUZINDO MELHORAS NO CHATBOT

Vamos adicionar intents e fazer algumas modificações no nosso chatbot (Citizen).

## FAZER UMA RECLAMAÇÃO

Execute o painel CitizenChatbotWebUI ou CitizenChatbotSDUI e cumprimente o bot. Você pode trabalhar indistintamente com Web ou SD.

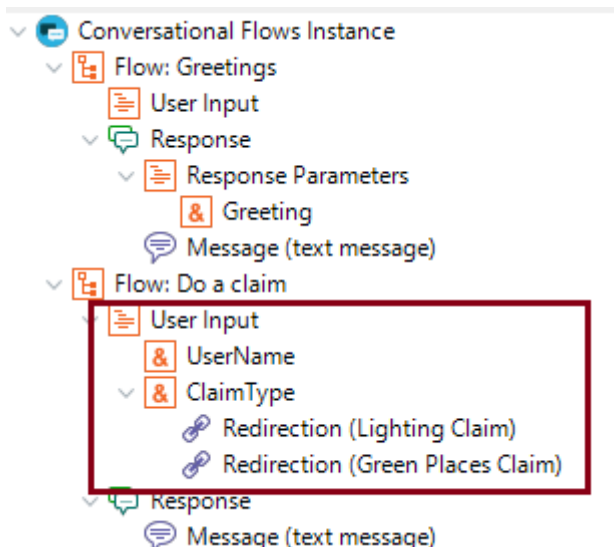
O bot perguntará por onde continuar. No momento, insira “Do a claim” ou um conceito semelhante, como na captura a seguir:



Imediatamente se chama o Flow “Do a claim”, que lhe solicitará as informações a serem fornecidas.

As informações solicitadas a serem inseridas são as User Inputs que vemos na definição do Flow.





Você verá que o primeiro é o &UserName, para o qual a propriedade Clean Context Value está desativada. Isso significa que o bot lembrará esse dado no contexto, para não solicitar novamente dentro da mesma sessão em qualquer um dos Flows da conversa.

Observe que o outro dado solicitado é ClaimType. Este é particular, no sentido de que é mapeado com uma entidade em Watson, que leva o mesmo nome. Isso, notamos vendo a propriedade Match With Entity deste Input.

variable: ClaimType	
Variable	ClaimType
Description	
Match With Entity	True
Entity	Claim_Type
Ask again	False
Ask Messages	What's the topic of your claim? It can be about Lighting, o...
On Error Messages	Sorry but &GXUserinput isn't a valid Claim type. It should...
Clean Context Value	True
Validation Procedure	(none)
Required	Always

O que significa mapear com uma entidade no Watson? Que será controlado que o usuário não insira um dado que não esteja dentro dos valores (e seus sinônimos) desta entidade. É como uma verificação de integridade contra o Provider.

Pode ver no Watson, que os valores possíveis desta entidade são os seguintes:

← | @Claim\_Type

Entity name  
@Claim\_Type

Value name  
[Enter value]

Synonyms  
Synonyms ▼ Add synonym... +

Add value Show recommendations

Dictionary Annotation <sup>BETA</sup>

<input type="checkbox"/>	Entity values (3) ▼	Type	
<input type="checkbox"/>	Lighting	Synonyms	Lights, illumination
<input type="checkbox"/>	Sanitation	Synonyms	Green Places
<input type="checkbox"/>	Traffic	Synonyms	

Observe que cada um dos possíveis valores de entrada redireciona para outro Flow, para continuar a conversa. Isso é dado pela propriedade Redirect to Flow, sob um nó [Redirection](#).

- Flow: Do a claim
  - User Input
    - UserName
    - ClaimType
      - Redirection (Lighting Claim)
      - Redirection (Green Places Claim)
  - Response
    - Message (text message)

**Properties**

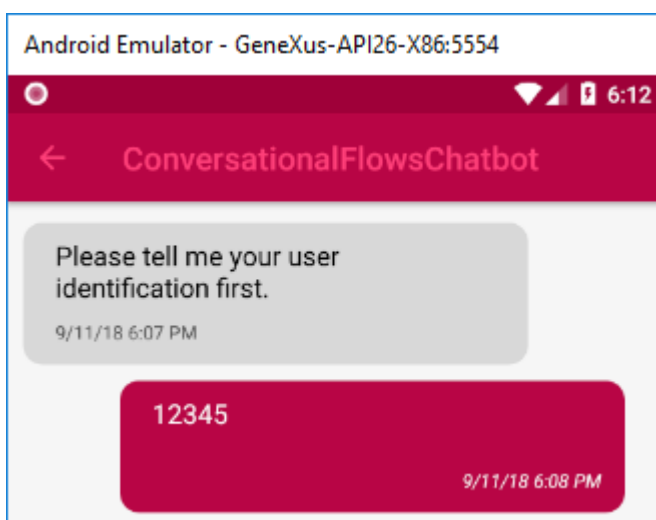
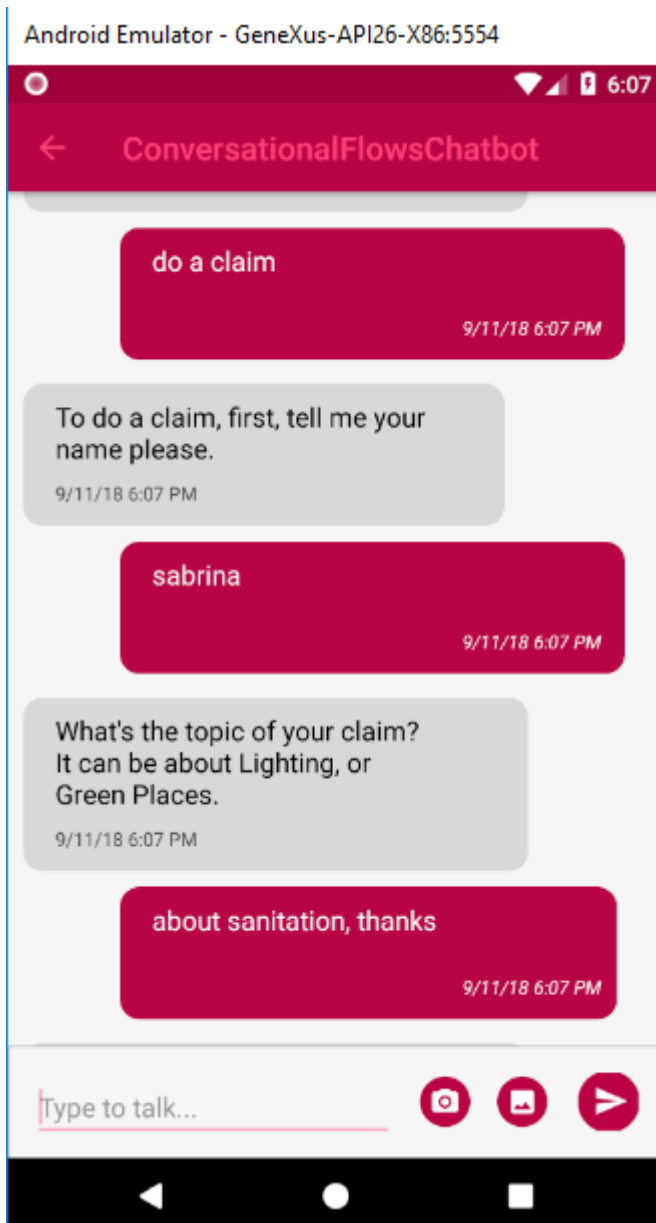
Filter

Redirection: Redirection (Green Places Claim)

Condition	&ClaimType='Sanitation'
Redirect to Flow	Green Places Claim

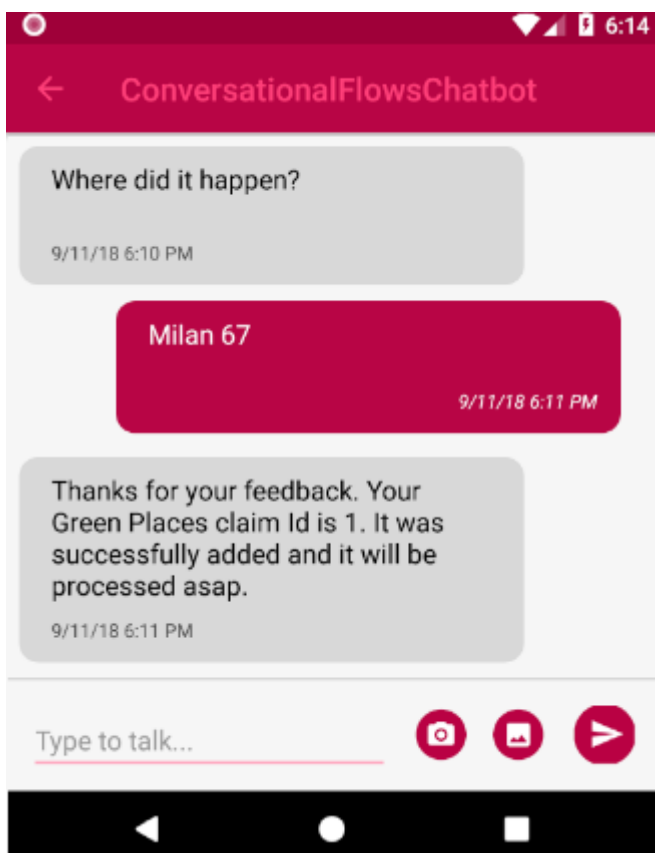
Você pode tentar executar qualquer um deles e ver como o diálogo é acionado.

Por exemplo, uma conversa possível é esta:



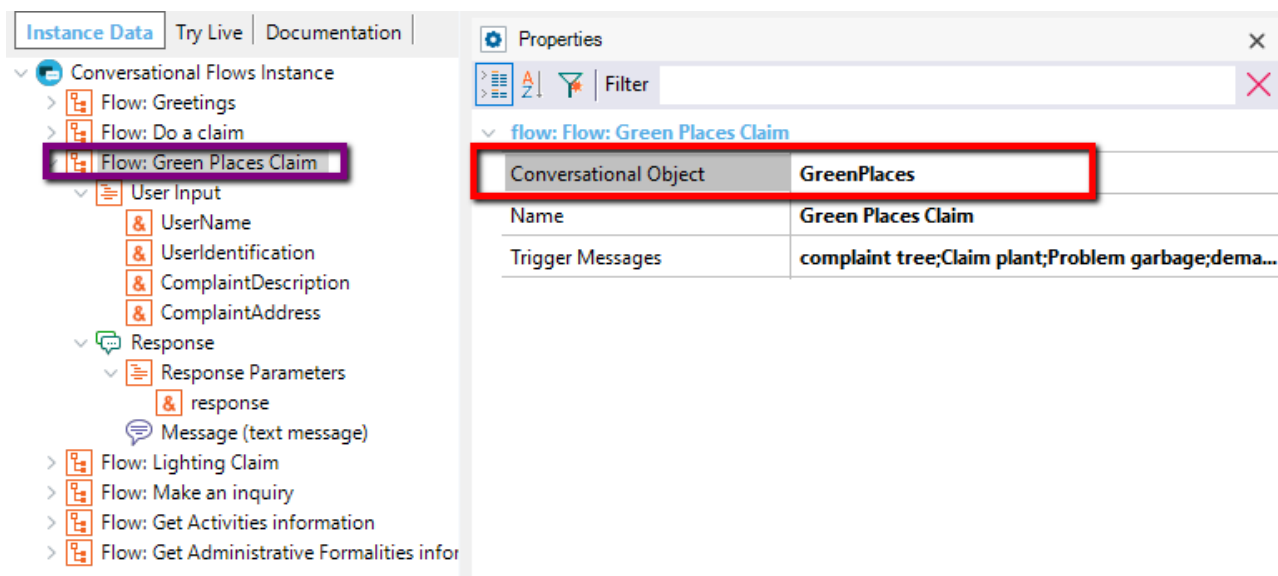
Nota: Você pode procurar os usuários de Citizen executando o panel Home do backend.

Escreva por exemplo “There is a broken pipe” e o bot lhe responderá:



Vejamos o Flow “Green Places Claim”.

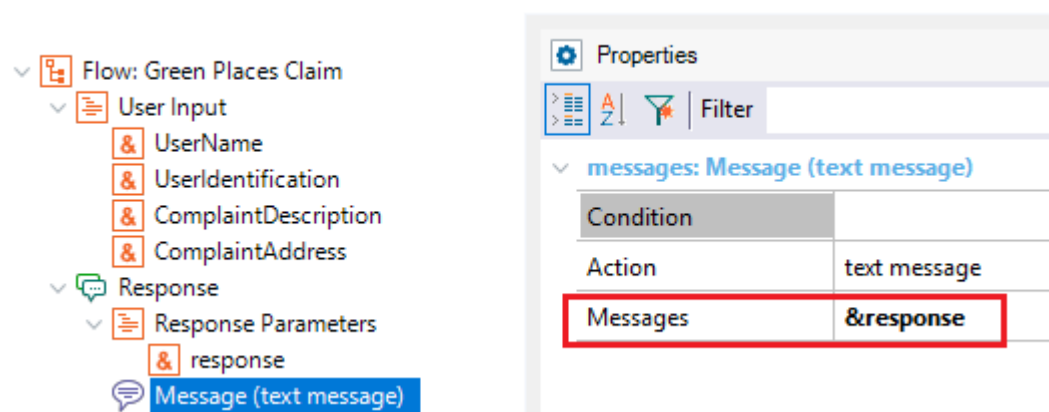
É um fluxo simples, em que, quando os dados são inseridos, o resultado da resposta ocorre por meio da execução de um procedimento, que é fornecido na propriedade “Conversational Object” do Flow. Trata-se do objeto GreenPlaces.



Abra o objeto GreenPlaces e observe que ele recebe como parâmetro um subconjunto dos inputs solicitados ao usuário e são usados para registrar a demanda do usuário.

```
parm(in:&UserIdentification, in:&ComplaintDescription,
in:&ComplaintAddress, out:&response);
```

O parâmetro de saída, &response, pode vê-lo na propriedade Message do nó Response. Isso significa que a saída deste Flow é, simplesmente, a resposta retornada pelo procedimento GreenPlaces.



Simple assim!

Resumo: Como se resolve a resposta? Quando o usuário termina de inserir os dados solicitados, se executa o objeto dado na propriedade Conversational Object.

Observe e tente:

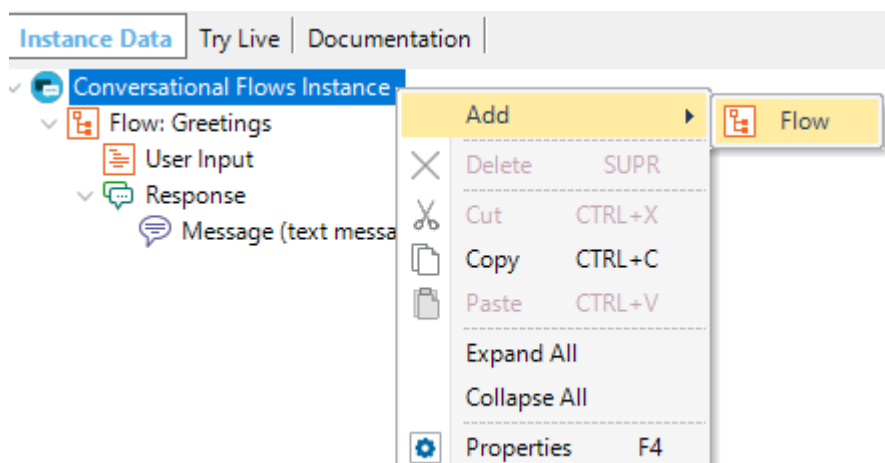
O usuário pode entrar diretamente no Flow “Green Places claim”, ou qualquer um dos Flows de reclamação, ou seja, não tem porquê ser solicitado que insira o motivo da reclamação, mas pode incluir diretamente em sua consulta. Tente inserir “I want to do a demand about a lighting issue”.

Na consulta pode incluir qualquer um dos inputs que mapeiam com uma entidade, e não será solicitado novamente. Exemplo “I’m user 12345 and I want to do a claim about missing lights on the street.”

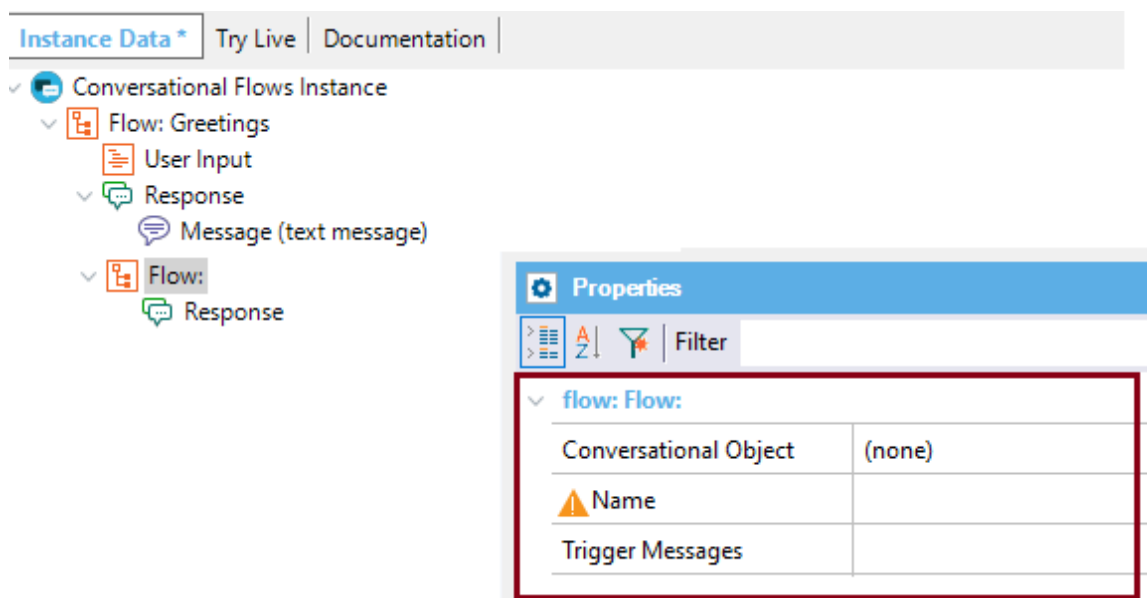
#### OPCIONAL: VER AS ATIVIDADES CULTURAIS

Neste ponto, vamos adicionar uma nova intent ao nosso chatbot, para que o usuário possa ver as atividades culturais onde possa participar.

Adicione um Flow chamado “Get Activities information”. Para isso, precisa clicar com o botão direito do mouse no nó pai da instância "Conversational Flows Instance" e depois em Add -> Flow.

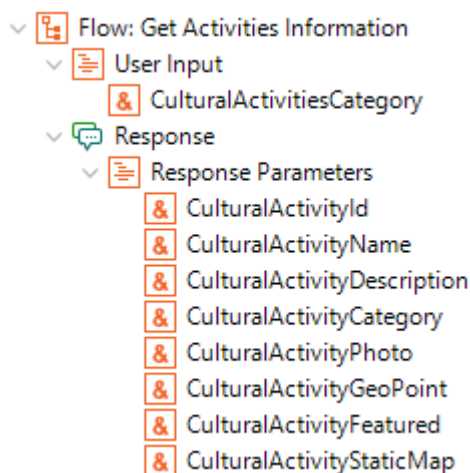


Lá você terá que completar o Name do Flow (“Get Activities information”).



Na propriedade **Conversational Object**, configure o Data Provider “CulturalActivitiesNews”.

Verá que automaticamente se completam a **User Input** e os **Response parameters**. Isto é feito pelo GeneXus a partir dos dados do **Conversational Object** selecionado.



Abra o Data Provider CulturalActivitiesNews e verá em que consiste e os parâmetros que recebe e retorna. Basicamente, recebe uma categoria de atividade (Arte, cultura, natureza) e retorna as atividades conforme esse filtro.

```
parm(in:&CulturalActivitiesCategory);
```

```
CulturalActivity
where CulturalActivityCategory = &CulturalActivitiesCategory
{
    CulturalActivityId = CulturalActivityId
```

```
CulturalActivityName = CulturalActivityName
CulturalActivityDescription = CulturalActivityDescription
CulturalActivityCategory = CulturalActivityCategory
CulturalActivityPhoto = CulturalActivityPhoto
}
```

Voltando ao Flow, selecione a **User Input** “CulturalActivitiesCategory”.

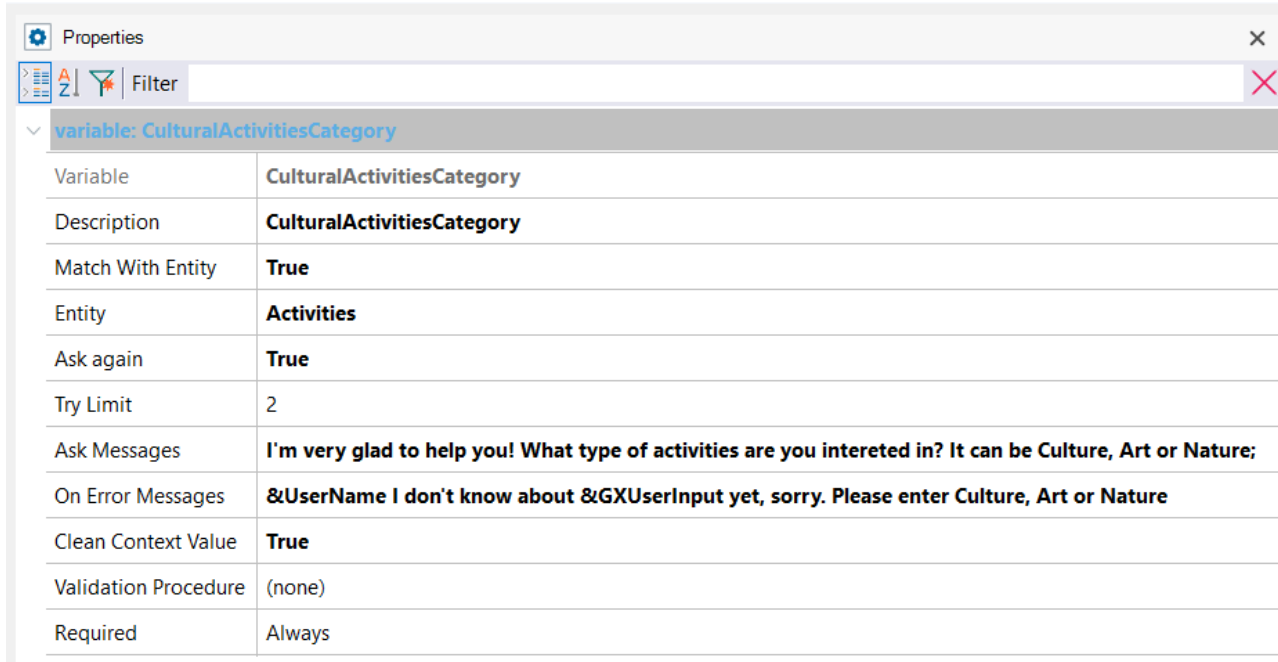
Configure as seguintes propriedades:

- Match with Entity = TRUE
- Entity = Activities
- Ask Messages = I’m very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Nature.
- Clean Context Value = TRUE

Flow: Get activities information

User Input

CulturalActivitiesCategory



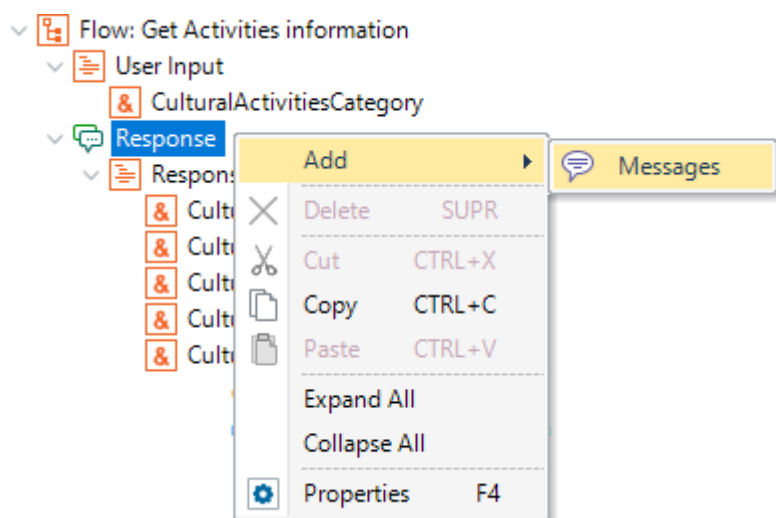
variable: CulturalActivitiesCategory	
Variable	CulturalActivitiesCategory
Description	CulturalActivitiesCategory
Match With Entity	True
Entity	Activities
Ask again	True
Try Limit	2
Ask Messages	I'm very glad to help you! What type of activities are you intereted in? It can be Culture, Art or Nature;
On Error Messages	&UserName I don't know about &GXUserInput yet, sorry. Please enter Culture, Art or Nature
Clean Context Value	True
Validation Procedure	(none)
Required	Always

Como resultado deste Flow, mostraremos ao usuário um grid horizontal com as atividades.

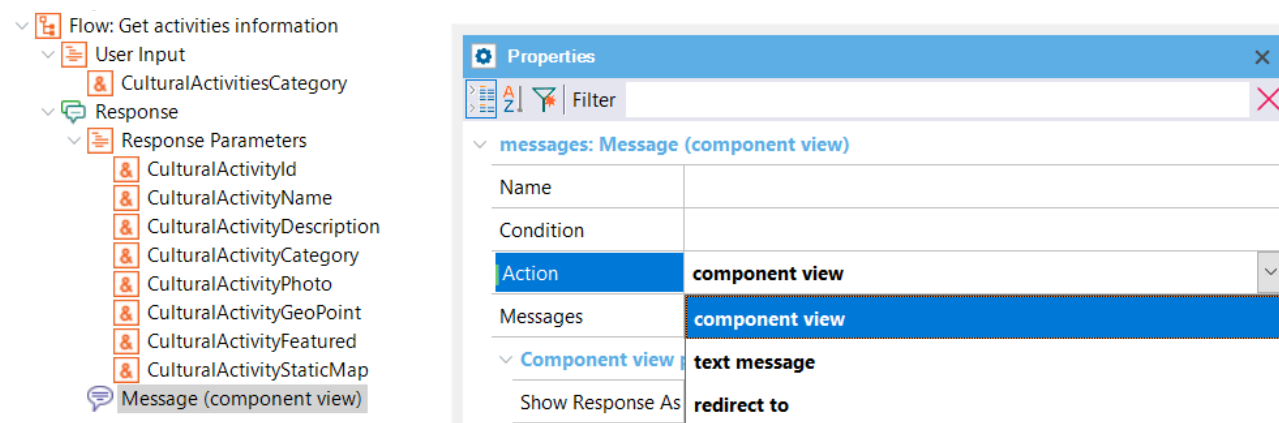
Adicione uma **Message**, cuja propriedade **Action** seja "Component View", para que a saída da resposta seja um componente.

Para adicionar a Message, deve clicar com o botão direito do mouse no nó Response e, em seguida, Add -> Messages.





No recém-criado nó Message, edite a propriedade Action e altere-a para o valor “component view”.

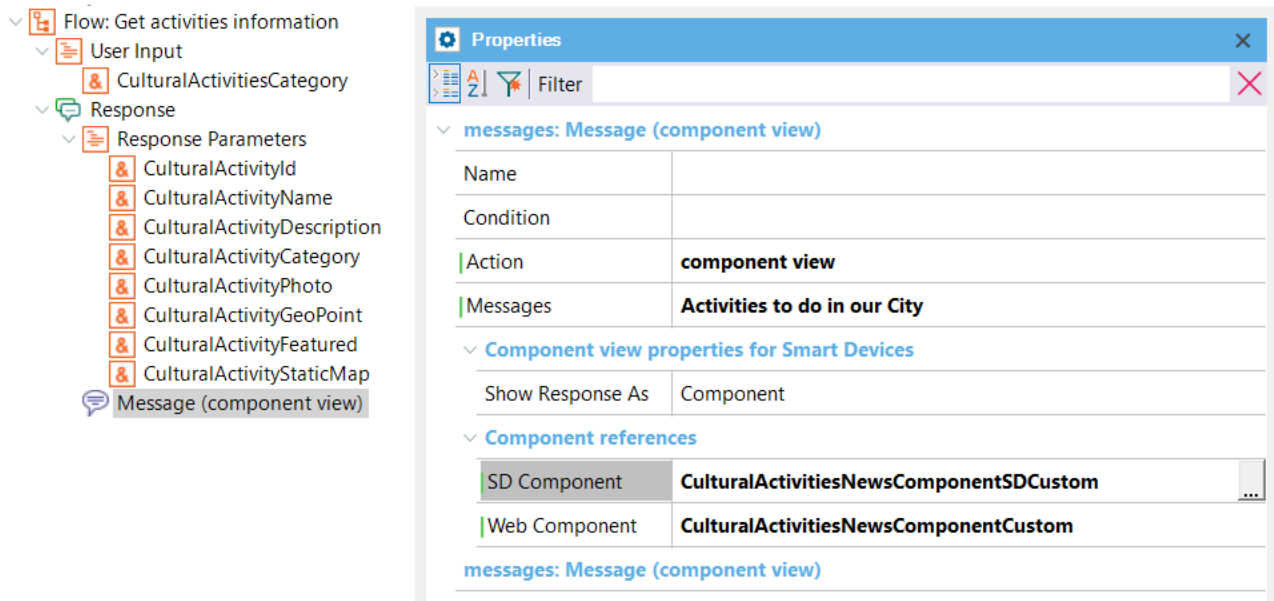


Edite a propriedade messages e digite “Activities to do in our City”.

Agora falta indicar qual é o componente a ser mostrado como resultado deste Flow, onde teremos o horizontal grid que exibe os dados.

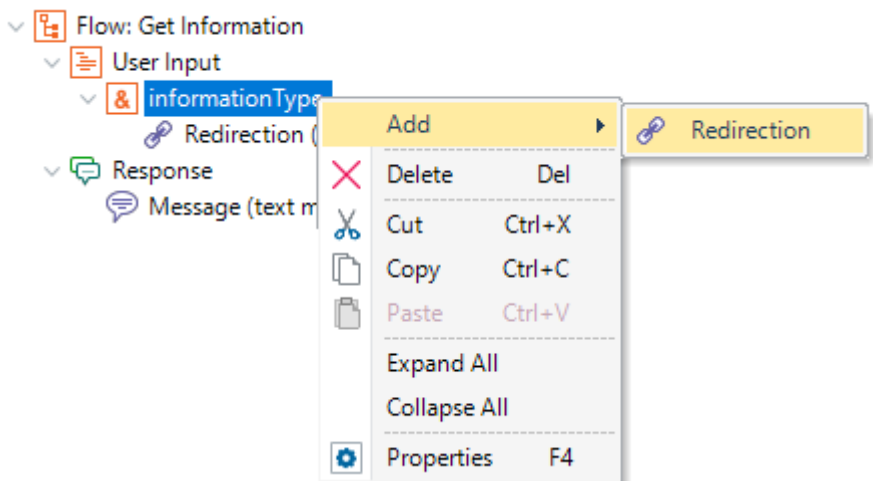
Poderíamos usar o objeto gerado automaticamente para este fim (indicado na propriedade Generated web Component), mas usaremos o nosso.

Configure a propriedade **Web Component** com o valor “CulturalActivitiesNewsComponentCustom”, e a propriedade SD components com o valor “CulturalActivitiesNewsComponentSDCustom”, conforme mostrado na figura:

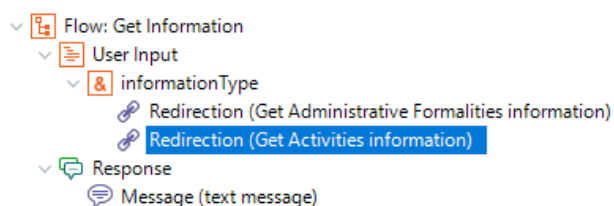


Para concluir o exercício, vamos editar o Flow “Get Information” e adicionar um nó Redirection que redireciona para o Flow que acabamos de criar.

Deve clicar com o botão direito do mouse no nó informationType, do Flow “Get Information” e, em seguida, Add -> Redirection.



Em seguida, preencha o seguinte (deve respeitar o casing no que foi inserido na Condition):



Properties	
Filter	
Redirection: Redirection (Get Activities information)	
Condition	&informationType="Activities"
Redirect to Flow	Get Activities information

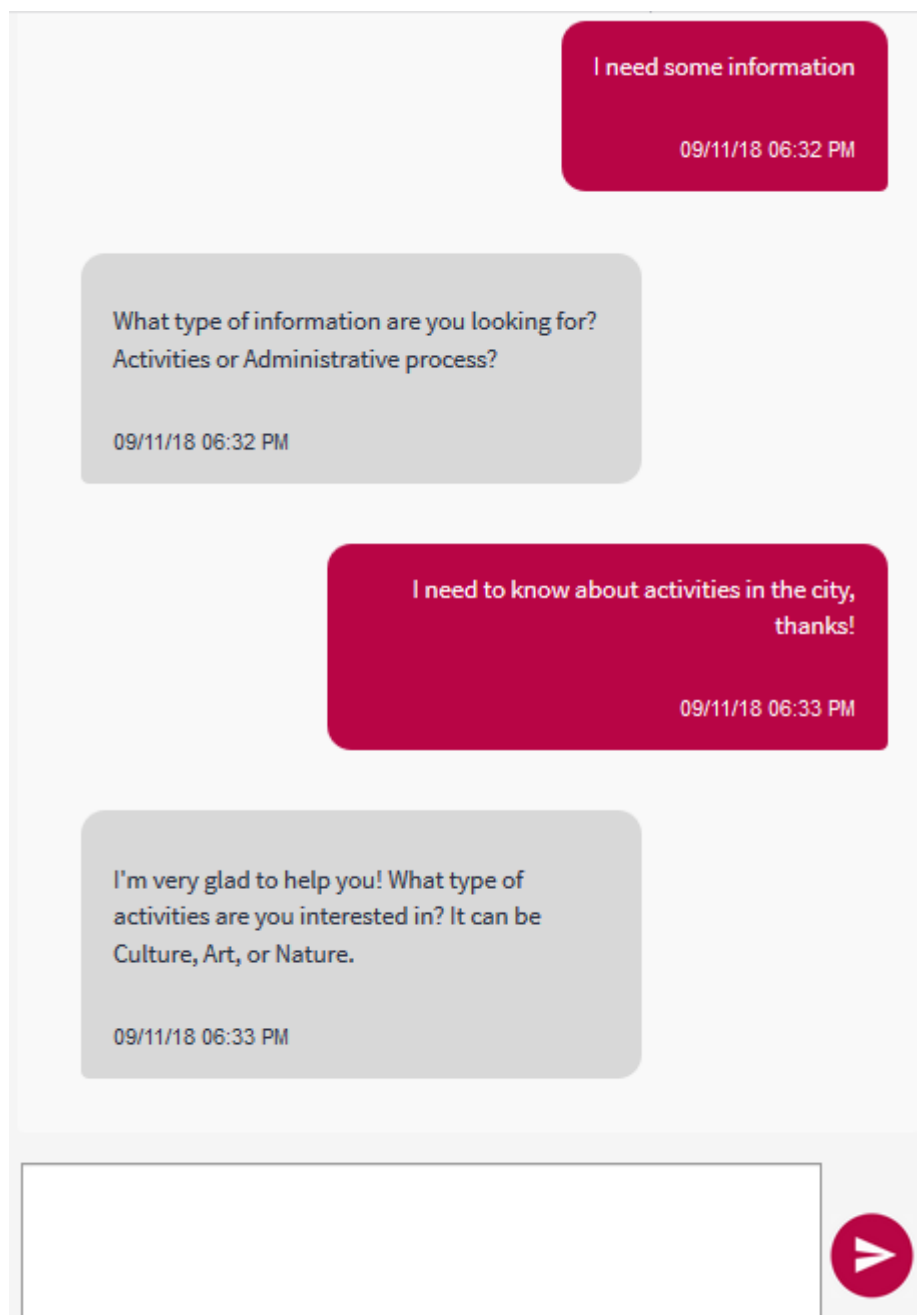
Dessa forma, se o usuário entrar pelo Flow “Get Information”, poderá ir ao Flow de “Get Activities Information” automaticamente, depois de ser solicitada a informação que deseja obter, e indique que se trata de “Activities”.

Pronto!

Para executar você deve fazer o seguinte:

1. Botão direito na instância Citizen -> Generate Chatbot
2. Run do webpanel CitizenChatbotWebUI e o painel CitizenChatbotSDUI para ver o resultado.

**Nota:** se não funcionar com isto, como workaround modifique o objeto handler para que seja main e faça um rebuild nele. Em seguida remova a propriedade main e volte a executar o passo 2.

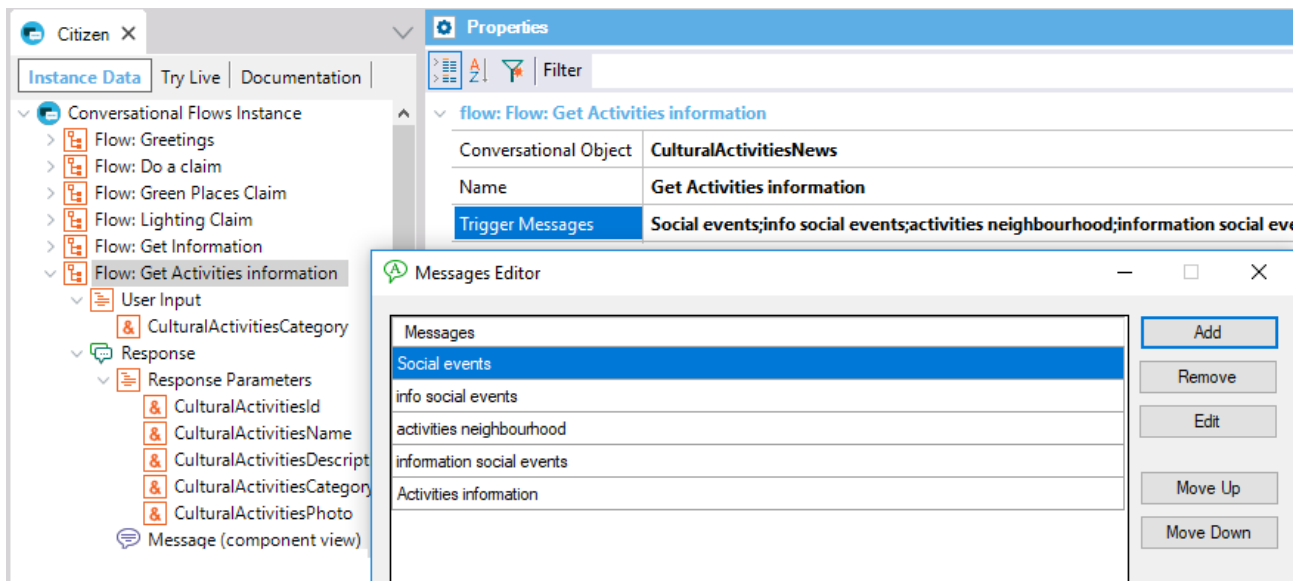


Escolha uma e verá o painel com o grid horizontal.

Melhor ainda, seria se fosse possível permitir que o usuário indicasse diretamente sobre quais temas deseja obter informações, e não precisar consultá-lo, caso não seja necessário.

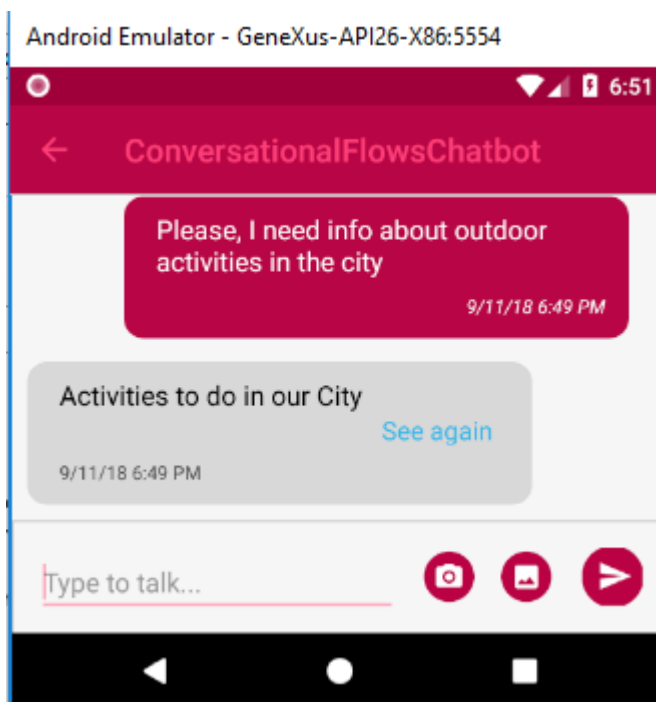
Isso significa treinar o chatbot para permitir ingressar no Flow “Get Activities Information” diretamente. Adicionemos algumas **Trigger Message** (o que lhe parecer apropriado), para que o Watson aprenda como pode ingressar o usuário neste fluxo diretamente, sem necessidade de vir de outro.

Por exemplo:



Pode tentar digitando a consulta “I need information about the outdoor activities!” (é um exemplo, tem liberdade para usar qualquer consulta semanticamente equivalente).

Observe que nessa consulta também adicionamos o tipo de atividade, que é “outdoor”.





## ANEXO: TRY LIVE

Esta seção é opcional.

Dentro do GeneXus, você pode testar o diálogo de seu chatbot usando a ferramenta Try Live.

**Try Live** é uma guia ao lado da guia Instance Data, onde pode testar como responde diretamente o NLP provider.

Pode testar qualquer consulta dentro do que estamos trabalhando em nosso chatbot.

Por exemplo:

The screenshot displays the 'Try Live' interface for a chatbot named 'Citizen'. The interface includes a header with 'Instance Data', 'Try Live', and 'Documentation'. The chat area shows a sequence of messages: a user query 'what's new about activities in the city?' (18:59), a bot response 'I'm very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Nature.' (18:59), a user response 'I'm not sure' (19:00), and a bot response 'I don't know about I'm not sure yet, sorry. Please, enter Culture, Art, or Nature.' (19:00). At the bottom, there is an input field, a 'SEND' button, and a 'Clear context' link. The status is shown as 'Status: 200'. On the right, the 'Provider Response' is displayed as a JSON object:

```

{
  "input": {
    "text": "what's new about activities in the city?"
  },
  "output": {
    "text": null,
    "nodes_visited": null,
    "log_messages": null
  },
  "context": null
}

```

**Nota:**

O Try live nos permite testar o reconhecimento das intents e a execução do fluxo, enquanto não há necessidade de chamar um objeto GeneXus. Quando este é o caso, o resultado que veremos na tela é a variável de saída do procedimento.

Lembre-se: Os chatbots devem ser treinados para ter um funcionamento ótimo com relação a todas as possibilidades de consulta que os usuários realizem.

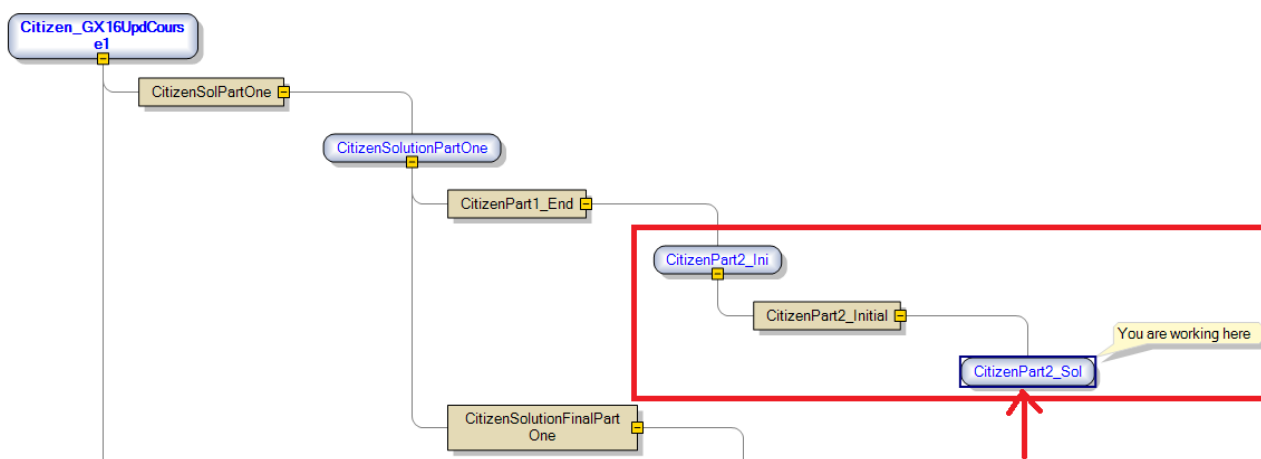
Treiná-lo significa adicionar **Trigger messages** às intents, para que estas sejam reconhecidas, reduzindo a possibilidade de ambiguidade.

## DOCUMENTAÇÃO

- [Chatbots in GeneXus](#)
- [Chatbot Generator](#)
- [Chatbots Architecture](#)
- [How to build a Chatbot using GeneXus](#)
- [Chatbot Generator Try Live](#)
- [IBM Watson Setup](#)

## KB SOLUÇÃO

Você pode baixar do GeneXus Server a KB solução deste prático para comparar resultados. É a versão da KB chamada CitizenPart2\_Sol.



Nós removemos o UserName e UserPassword. Configure os valores apropriados para poder executar.