Practice course for upgrade to

PART 2

July 2019

CONTENTS

OBJECTIVE

We'll see the power of the chatbot generator introduced in GeneXus 16.

We will create a chatbot for the reality of an application that provides customer services. The chatbot will be for both the web application and the SD application of that reality (we will do it on Android for easier prototyping).

We will use Watson as our Natural Language Processing (NLP) Provider. However, we may also use Dialogflow.

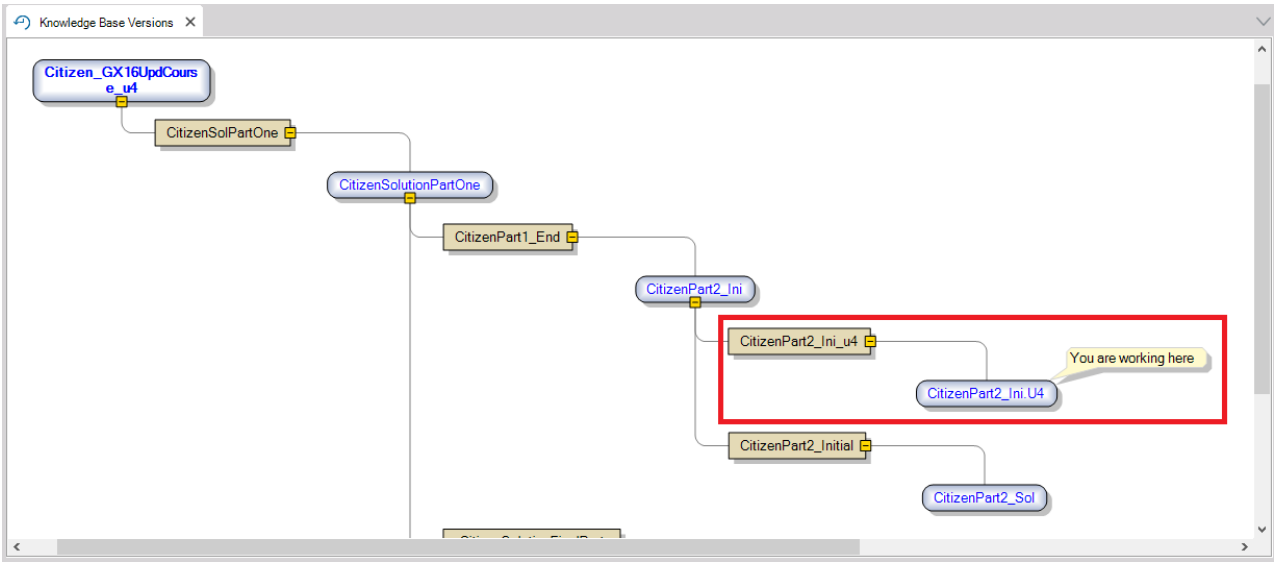
With our chatbot, users will be able to:

- Make a complaint for any issue arisen on the street: a fallen tree, a missing or damaged traffic sign, a car parked in the wrong spot, and that sort of things.
- Obtain information relative to cultural activities taking place, or to requirements for administrative procedures.

SOME PRELIMINARY ADJUSTMENTS

We'll skip the account configuration in the NLP provider, because it has already been done for this practice exercise. The instructor will provide you with the required information as needed.

We will start from version CitizenPart2_Ini.u4 of the KB Citizen_GX16UpdCourse at http://samples.genexusserver.com/v16. Make sure to activate that version and not another.



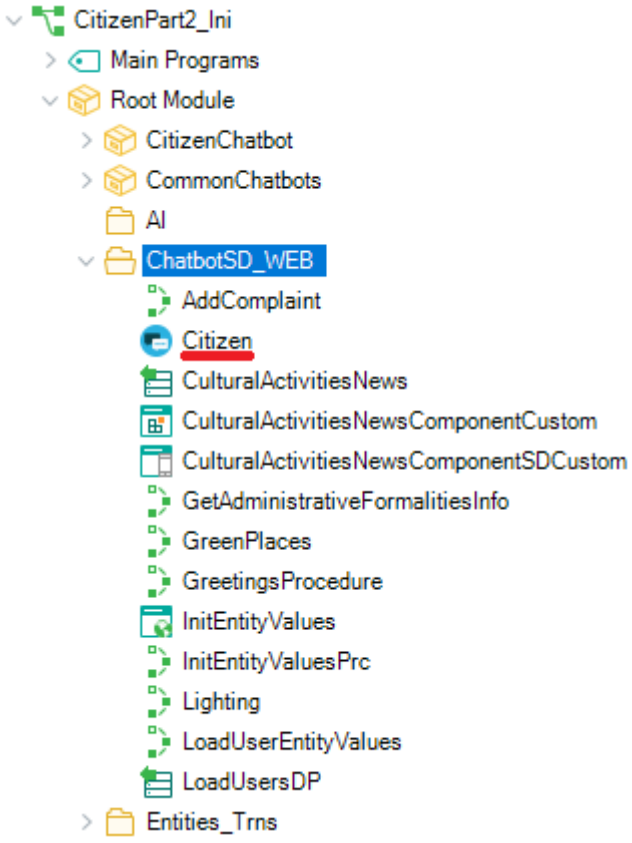
This version of the KB comes with folders, objects and modules already created so as not to waste time.

GETTING STARTED

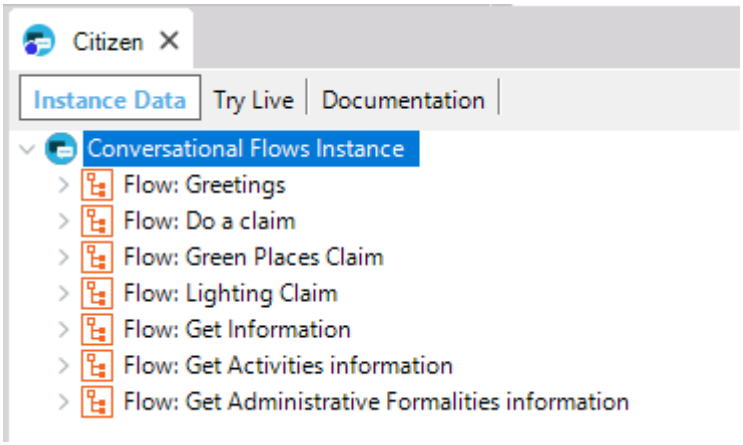
WHAT ARE THE BASIC CONCEPTS BEHIND A CHATBOT?

A chatbot is represented by an Object from the KB –something new in GeneXus 16– called Conversational Flows.

In the KB, in the ChatbotSD_WEB folder, we have already created a Conversational Flows object that will act as base for building new functionalities in the chatbots solution.



Please open it, …and you will see the design of a chatbot in GeneXus:



It has a tree structure, where each main node ("Flow") represents an intention by the end user, which the chatbot will be capable of responding properly.

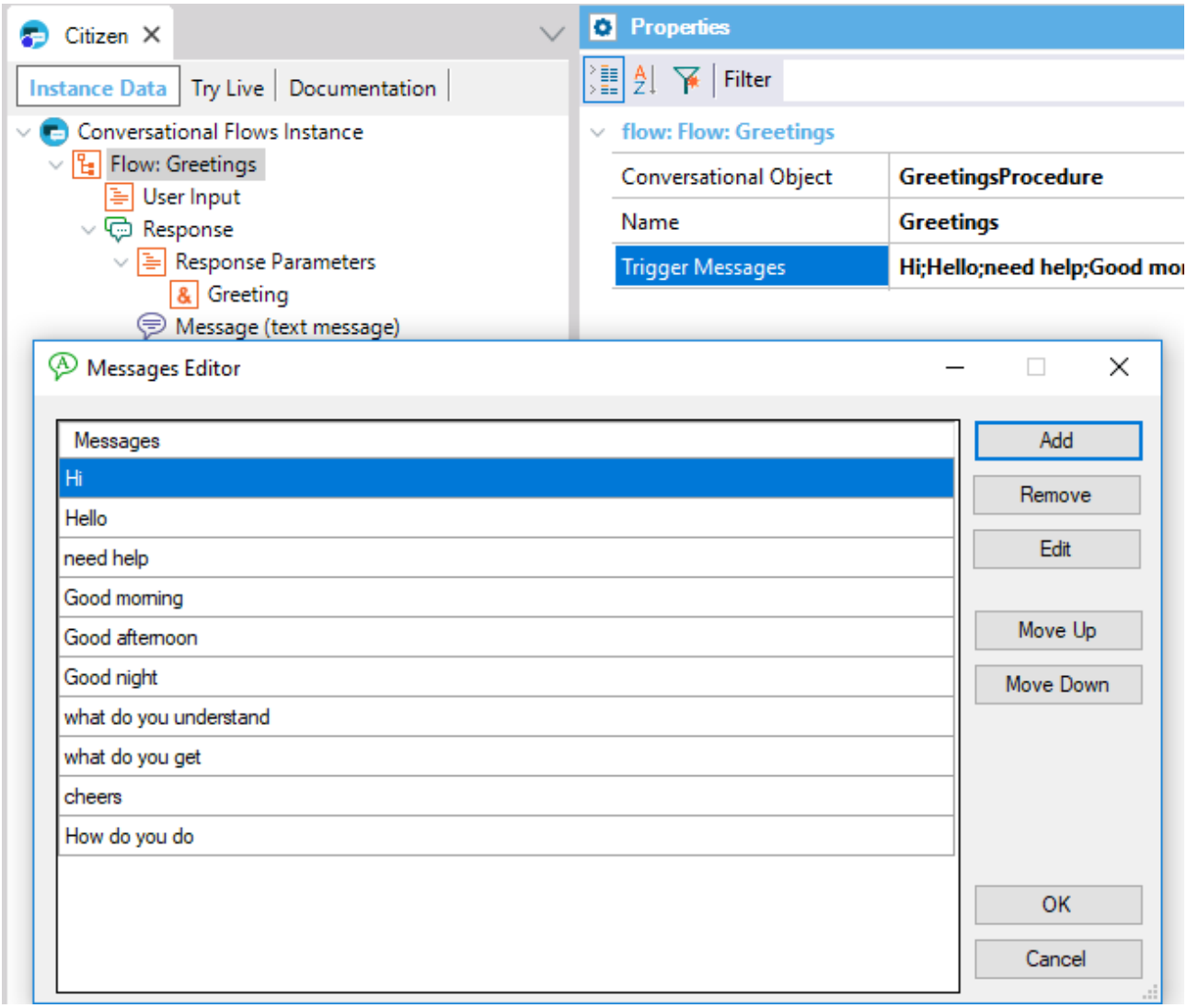
What is an intent? Something that reflects the user's will at the time of making a query, such as, for instance "Report a car that is not parked correctly."

In the diagram corresponding to the Conversational Flows Instance, the Flow is directly related to the Intent.

How does the NLP provider interpret the user's intention?

It does it through the Trigger messages that we add in the Flow. More complete messages will yield better inferences.

In "Flow: Greetings," note the Trigger Messages property that includes several greeting options:

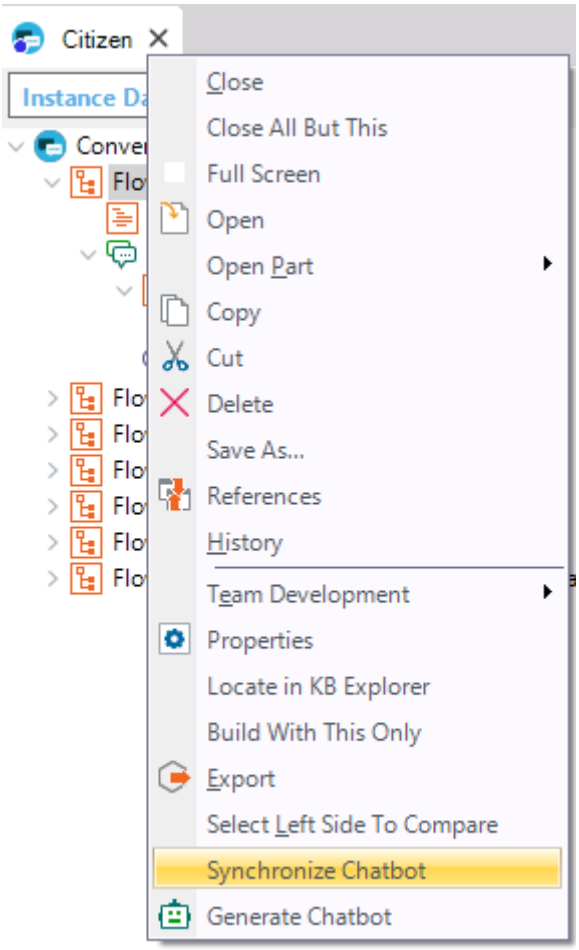


This chatbot that we have in the KB has not been synchronized with the Provider yet.

In the Citizen object, in the parent node “Conversational Flows Instance”, in the User name and User password properties enter the values provided by the instructor.

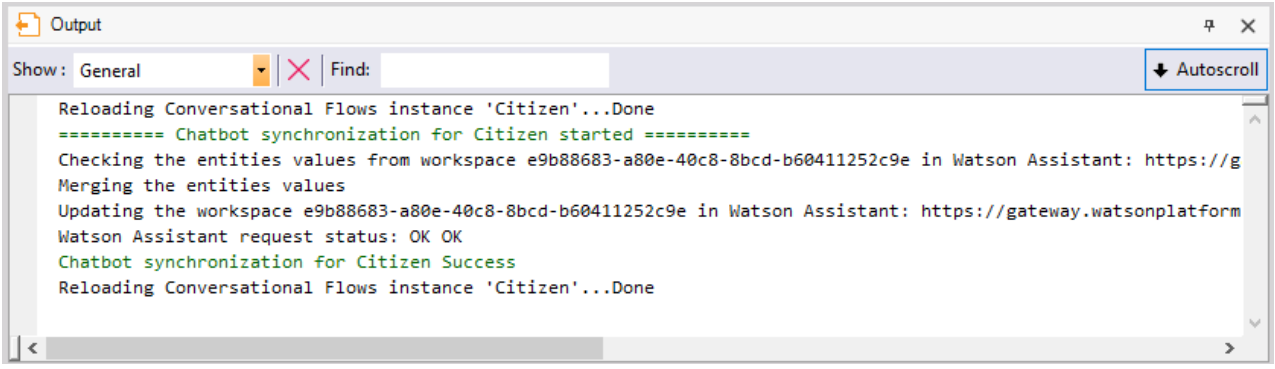
Save the object and watch how the dialog is created, with intents and Trigger messages in Watson.

By right-clicking on the object, we may select the Synchronize Chatbot option:



This action will generate a metadata file that is impacted on the NLP provider to generate the dialog with its intents and entities.

In the General tab of the GeneXus output, we can see the output of the execution corresponding to the Synchronize option.



What happens if we add a Trigger message for the “Greetings” Flow?

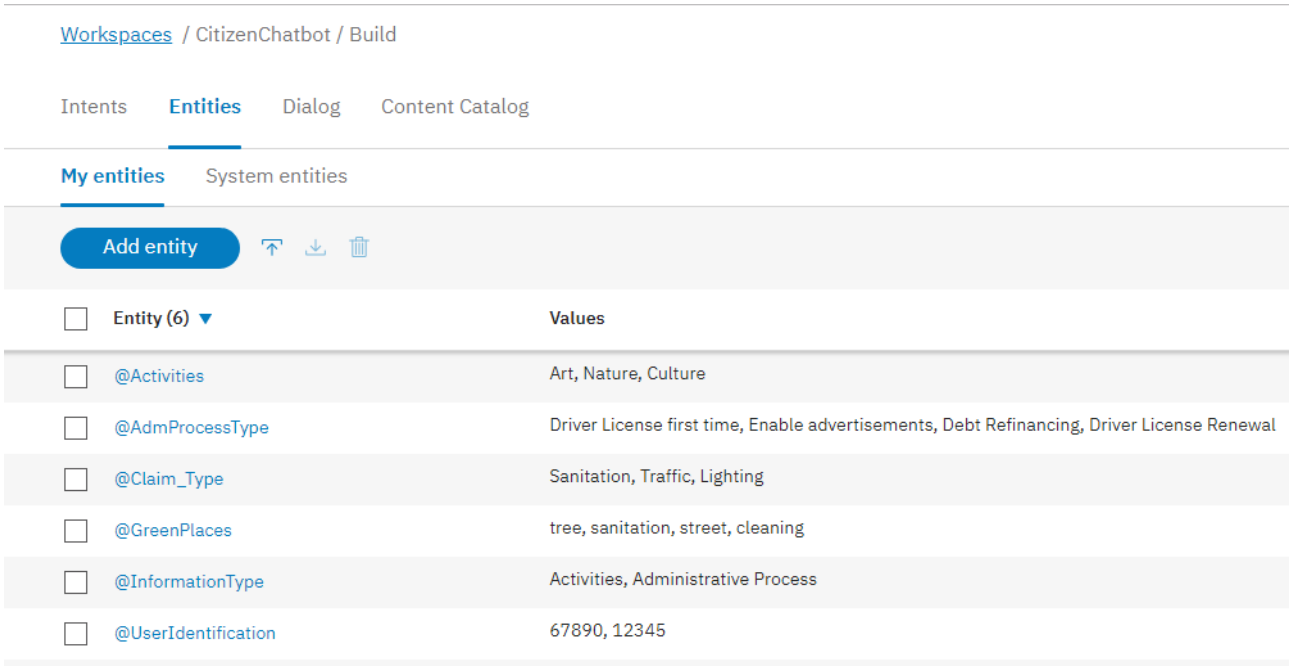
A change is made or a new Trigger Message is inserted in the Greetings Flow. The change is impacted in Watson by just saving the Conversational Flows object.

Upon a given intention by the user, you may distinguish different objects or topics inside the intention that we formally call Entities.

For example, upon an intention to set a date and time, an entity of that intention would be the reason for setting a date, and the value of that entity could be, for example, the renewal procedure for a driver's license.

So, we use the name entity to refer to an object in the query that groups several values, and one of them or several will be instanced in the query.

Entities are stored in the NLP provider, along with their possible values and synonyms.

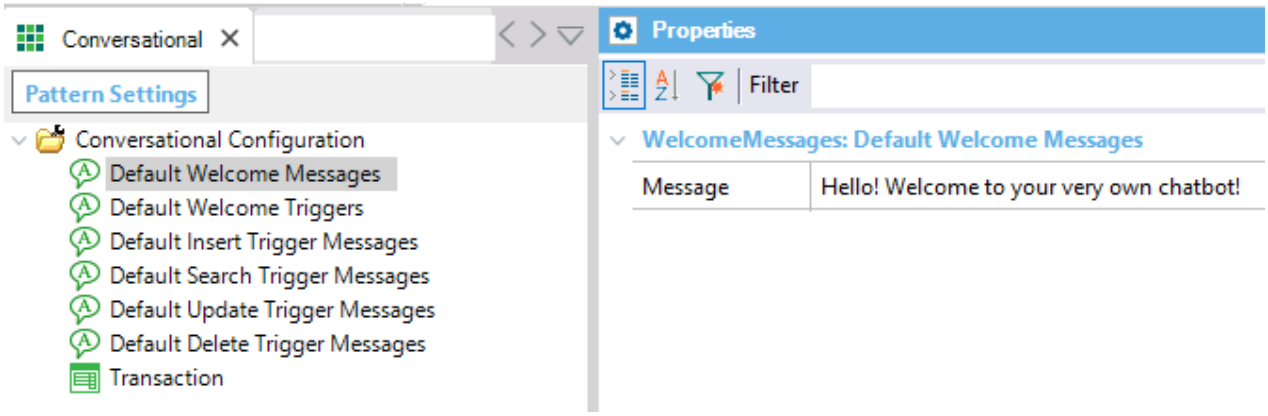


To create a chatbot, you must model its behavior (defining intents and entities) and the answers that are shown to the user for each intent.

That modeling is done in the conversational flows object (Citizen in our case).

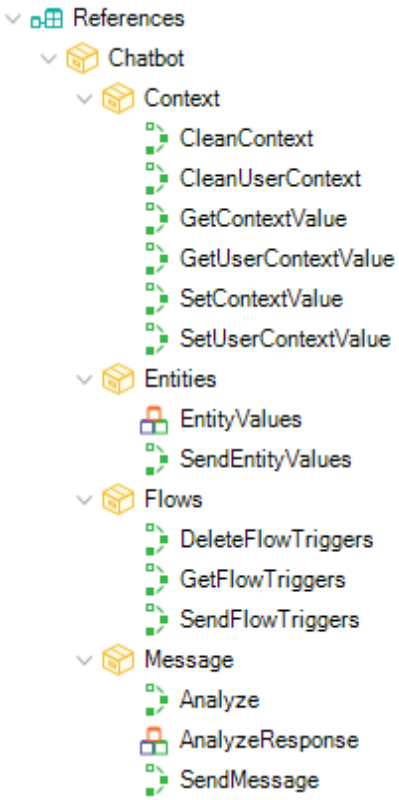
Note: The chatbots generator functions like a pattern. Based on the model, the pattern automatically generates all the objects necessary to solve the conversation between the user and the NLP provider.

These are the Preferences, under Patterns, the Conversational Flows and the properties.

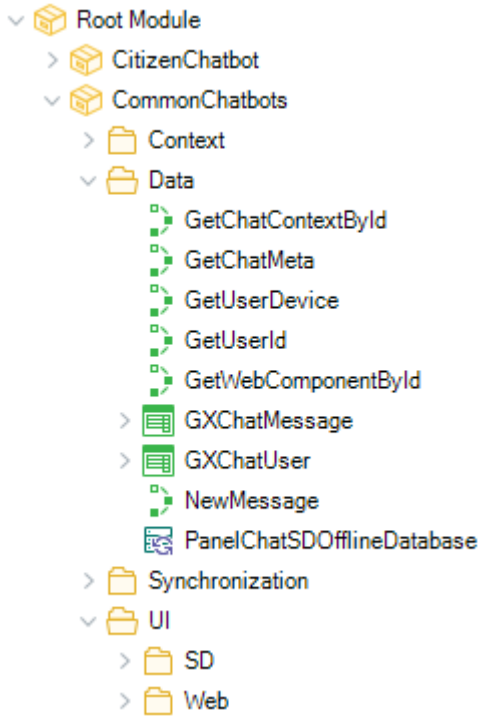


The components in the KB are as follows:

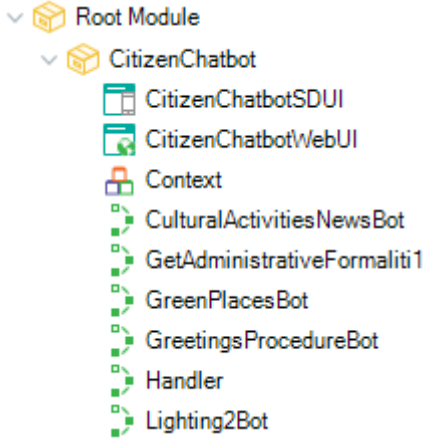
(1) Chatbots Module. This is an external module installed in the KB that solves the communication with the Provider.



(2) Pattern resources. These are example objects for creating a chatbot, and you may modify them at your discretion. They are in the CommonChatbots module.



(3) Objects generated. From the model, these objects are generated in a module containing the name of the Conversational Flows object (CitizenChatbot). They are a link between the objects in the KB and the server.



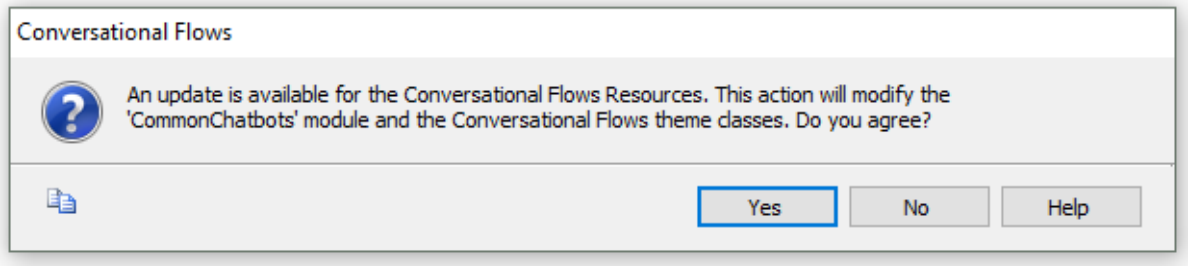
So far, these are the main concepts… and what the Chatbot Generator is about. Now you can start building!

THE CHATBOT IN ACTION

Right-click on the Citizen object and then on Generate ChatBot.

Note: the Generate Chatbot option causes the calculation of objects generated by the pattern (3) and their import into the KB.

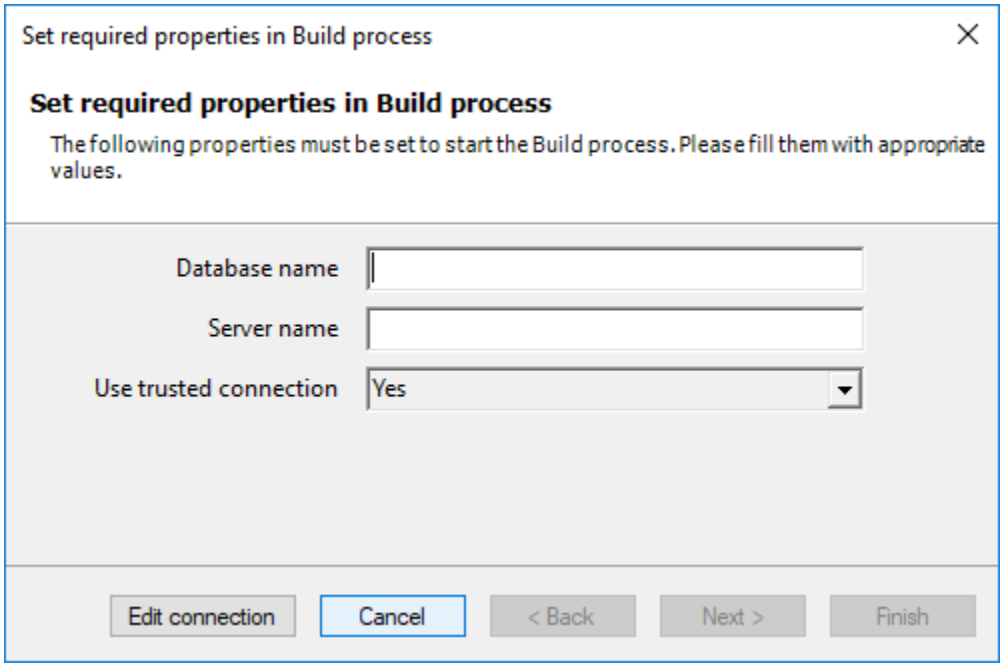
The following message indicates that the resources[2] of the chatbots generator will be updated.



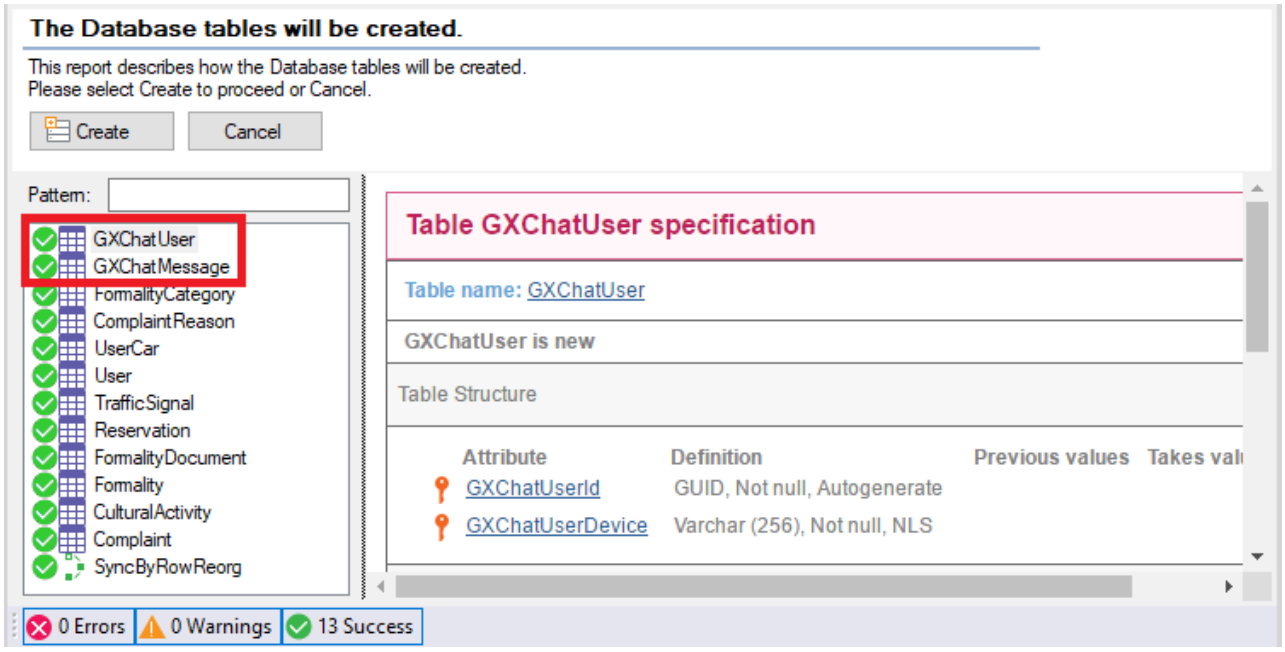
Note, in the output, the import of the resources for Web and SD of the chatbots generator.

Now do a Rebuild all. The build also generates the Pattern objects when necessary (if changes are detected in the instance).

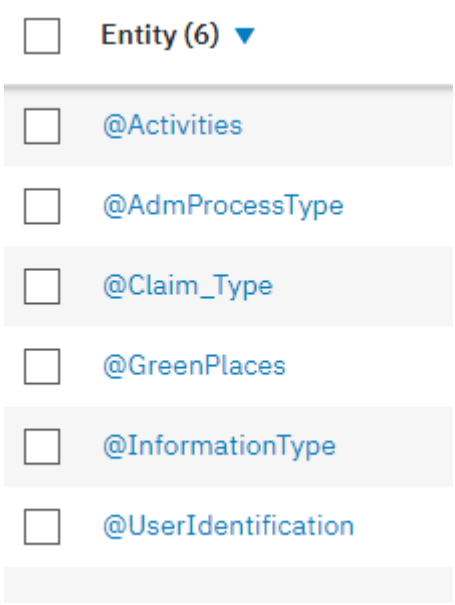
The DB Create action will be requested. Enter the DB Server corresponding to the local SQLServer (we're prototyping with a .Net environment).



Note that there are two tables called GXChatMessage and GXChatUser that are used by the chatbot.



You will be working with the following entities, which you may see in the user interface for IBM Watson assistant:



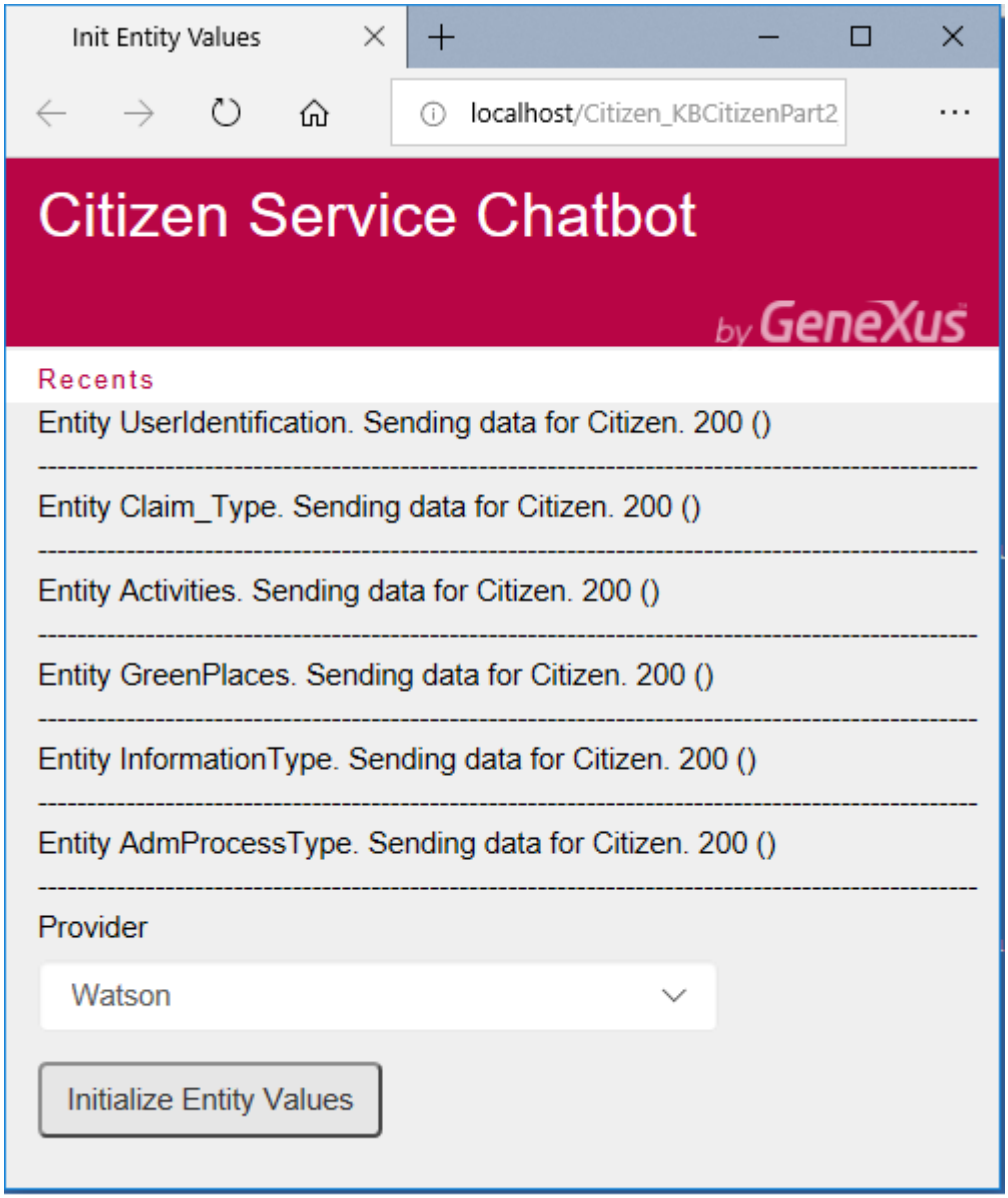
They will be loaded on the NLP provider by means of an API provided by the chatbots module.

Open the InitEntityValues object to learn about the method applied in uploading values and entities to Watson (Chatbot.SendEntityValues).

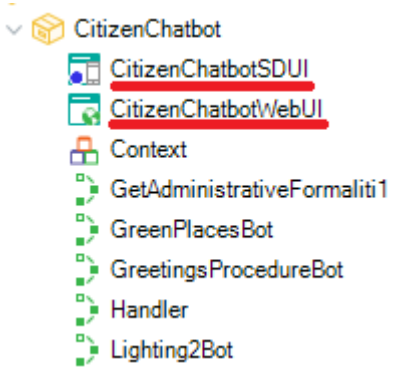
This always causes a merge with what you have in the NLP provider.

So, execute the InitEntityValues webpanel and press the Initialize Entity Values button.

The outcome of the execution will be:



Now execute the following objects to view the chatbot at runtime.



WEB application: Execute the CitizenChatbotWebUI main object

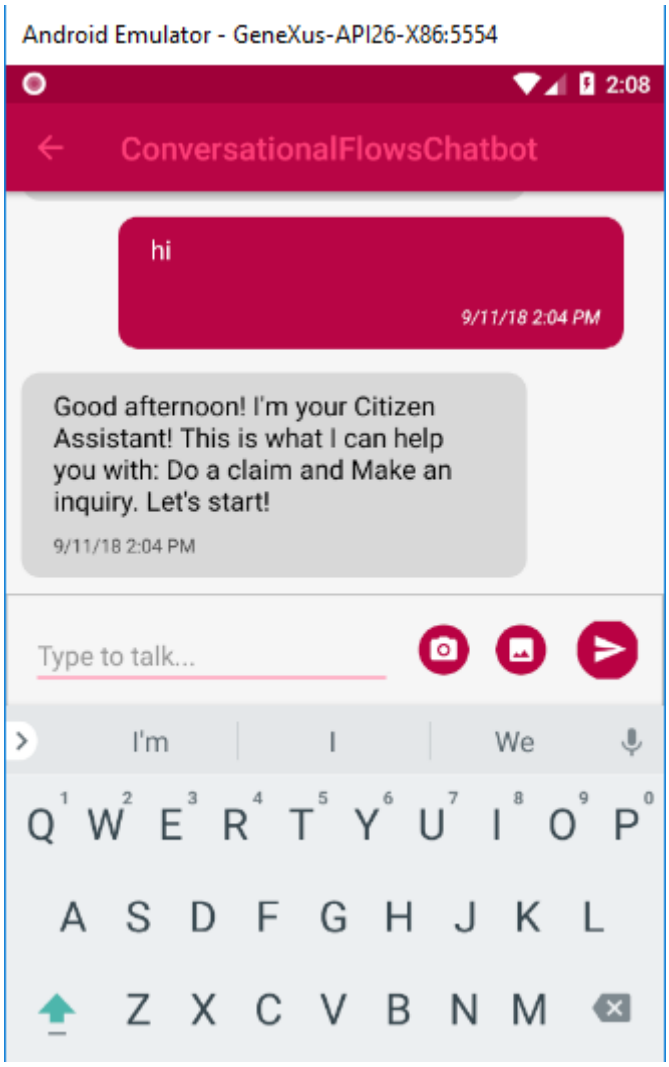
SD application: Execute the CitizenChatbotSDUI main object

Note: These objects call your chatbots main panel, passing the name of the instance and the Provider as parameter.

```
Event 'ClientStart'
      CommonChatbots.PanelChatSD.Call(!"Citizen")
EndEvent
```

You can already start dialoging with the Web and SD chatbot. Start by greeting the chatbot!

For example, in SD:



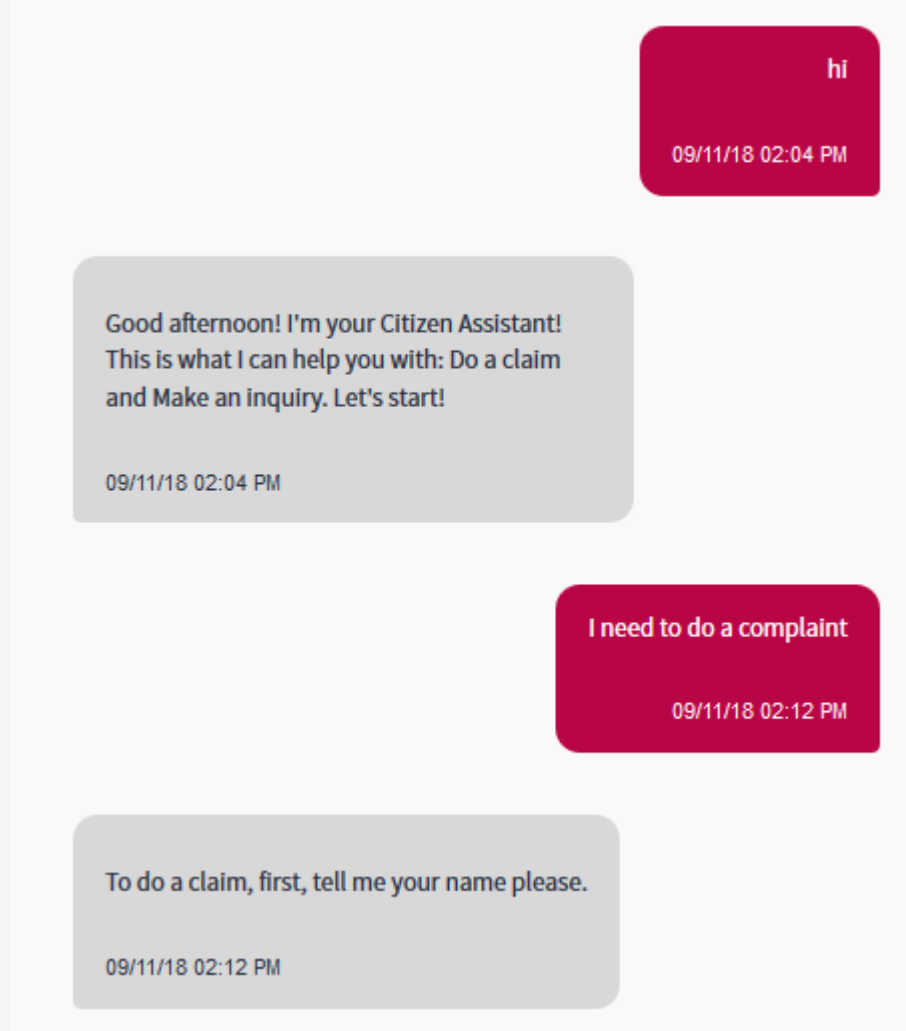
INTRODUCING ENHANCEMENTS IN THE CHATBOT

Now add intents and make some changes to your chatbot (Citizen).

MAKING A COMPLAINT

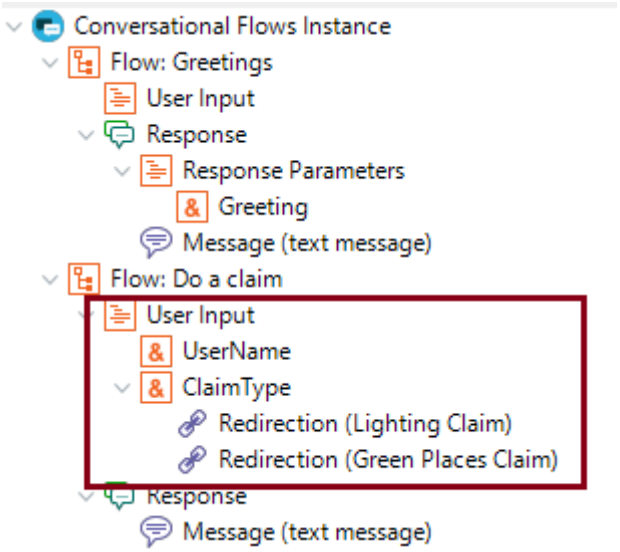
Execute the CitizenChatbotWebUI or the CitizenChatbotSDUI panel and greet the bot. You may use either Web or SD.

The bot will ask you where to go next. For the time being, enter “Make a complaint” or a similar concept, like in the capture below:



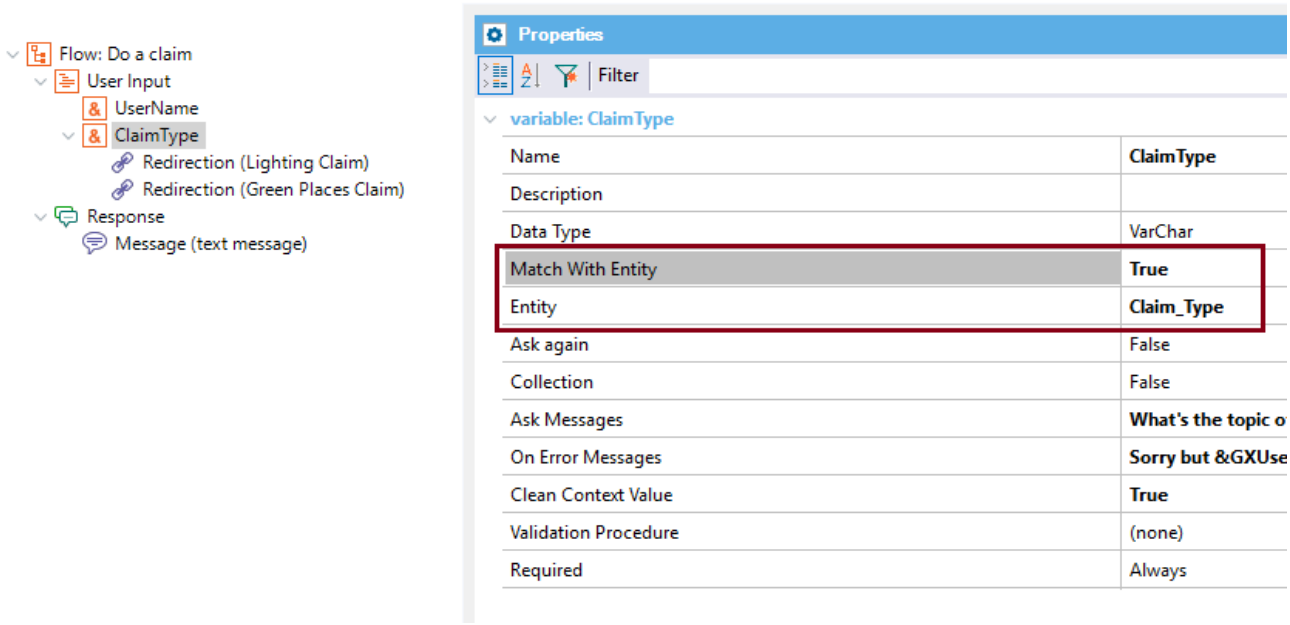
The “Make a complaint” Flow is called immediately, which will be asking you information for it to be completed.

The information you are asked to enter are the User Inputs displayed in the Flow definition.



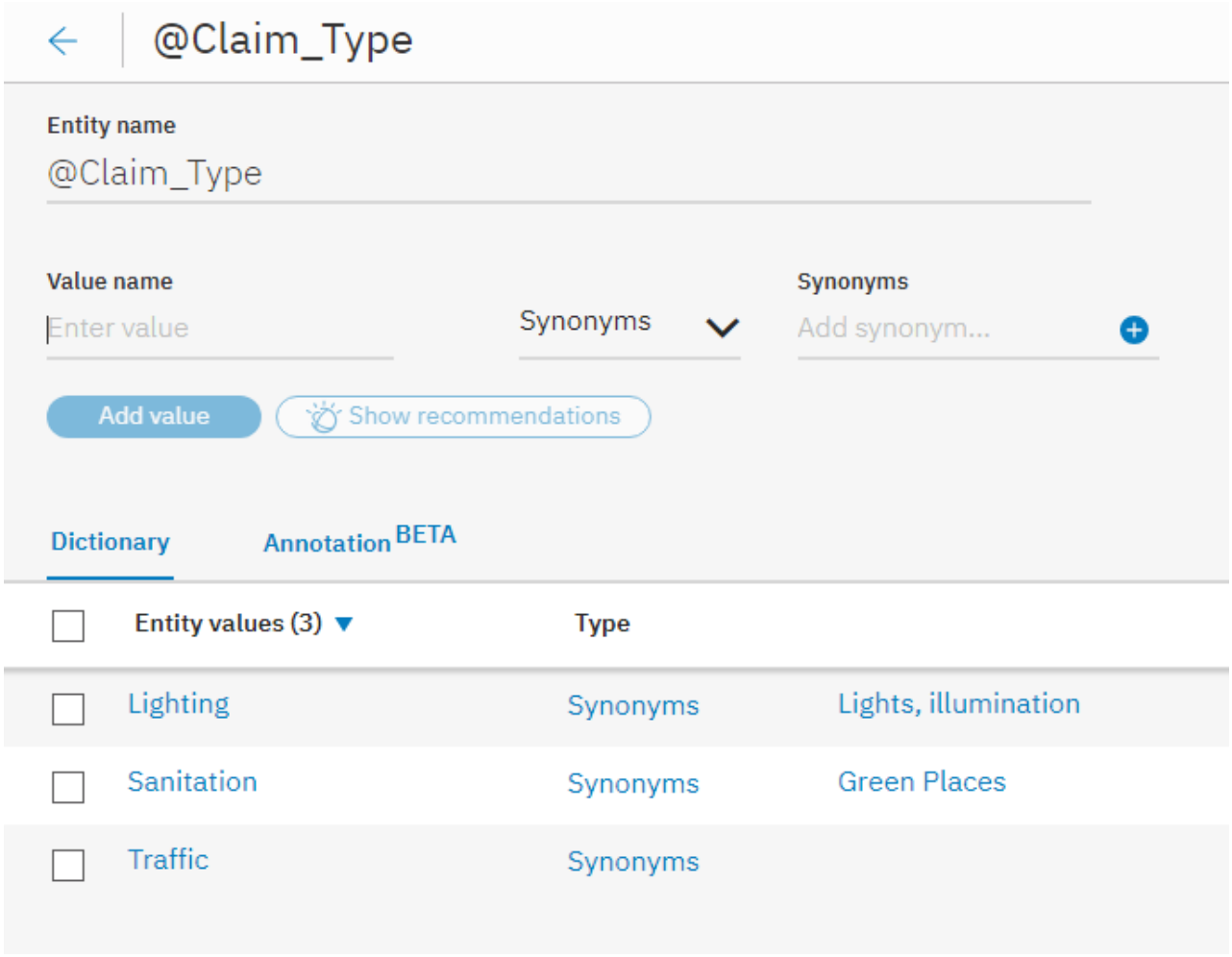
You will see that the first one is &UserName, for which the Clean Context Value property is off. This means that the bot will remember that data in the context, to not request it again within the same session in any of the Flows in the conversation.

Note that the other data requested is ClaimType. This is specific in that it is mapped with an entity in Watson that has the same name. You can see this by viewing the Match With Entity property in this Input.

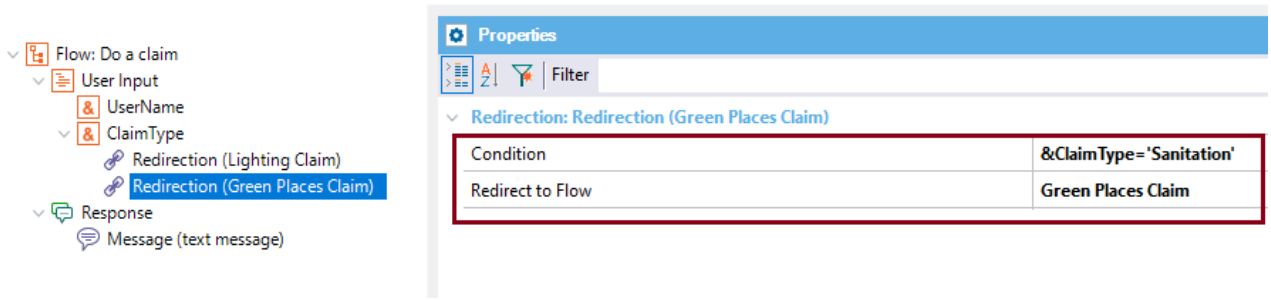


What does "being mapped with an entity in Watson" mean? That there will be a control for the user not to enter data that is outside the values (and their synonyms) of this entity. It's like an integrity check, against the Provider.

In Watson, you will see that the possible values for this entity are as follows:

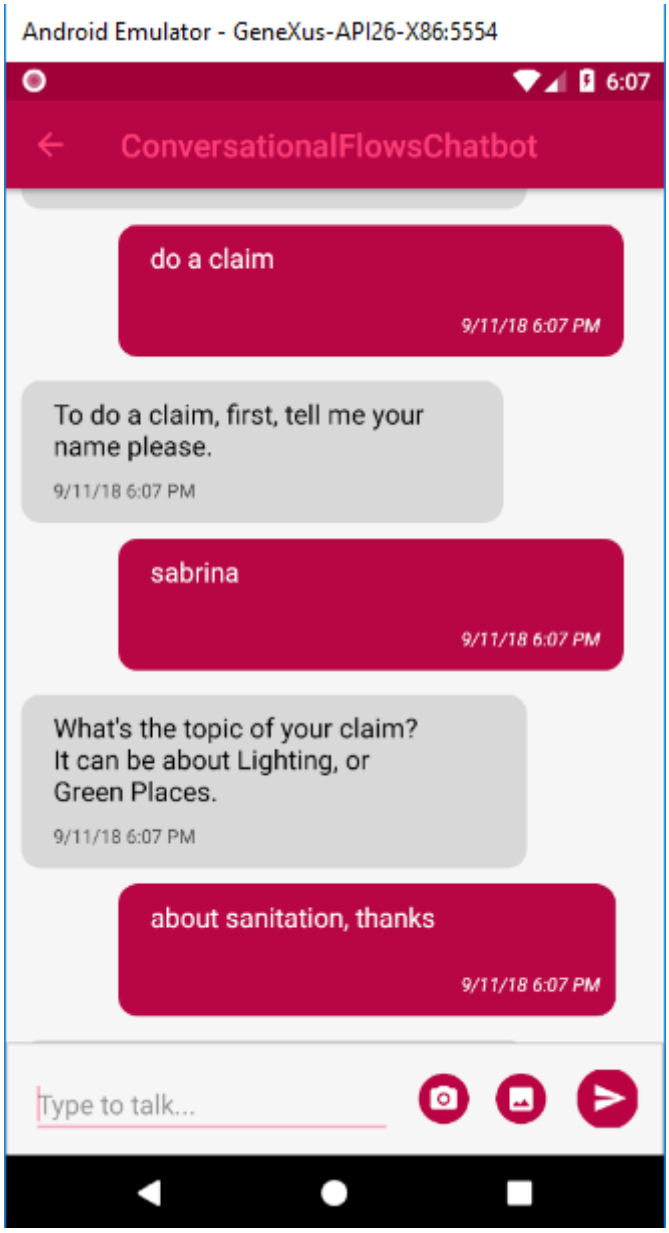


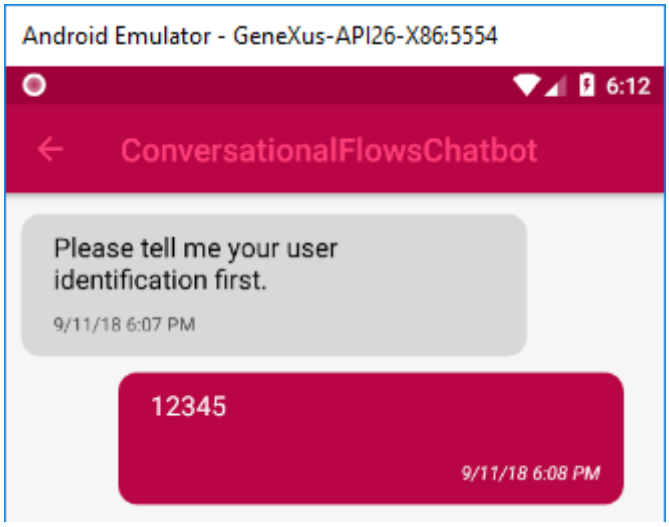
Note that each of the possible input values will redirect to another Flow to that the conversation may continue. This is due to the Redirect to Flow property, under a Redirection node.



You can try executing any of them and see how the dialog progresses.

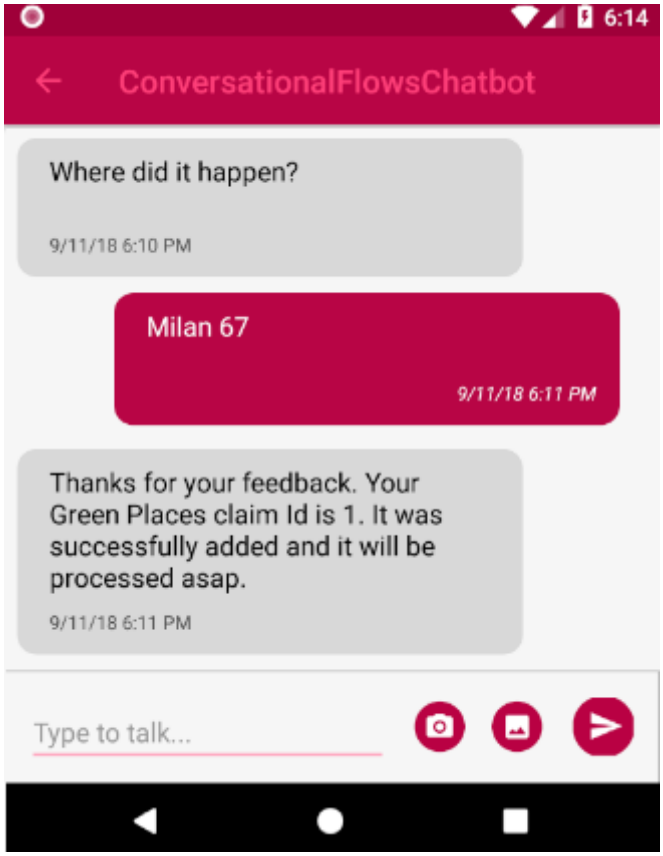
For example, the following is a possible conversation:


Android Emulator - GeneXus-API26-X86:5554
6:07
ConversationalFlowsChatbot
do a claim
9/11/18 6:07 PM
To do a claim, first, tell me your name please.
9/11/18 6:07 PM
sabrina
9/11/18 6:07 PM
What's the topic of your claim? It can be about Lighting, or Green Places.
9/11/18 6:07 PM
about sanitation, thanks
9/11/18 6:07 PM
Type to talk...

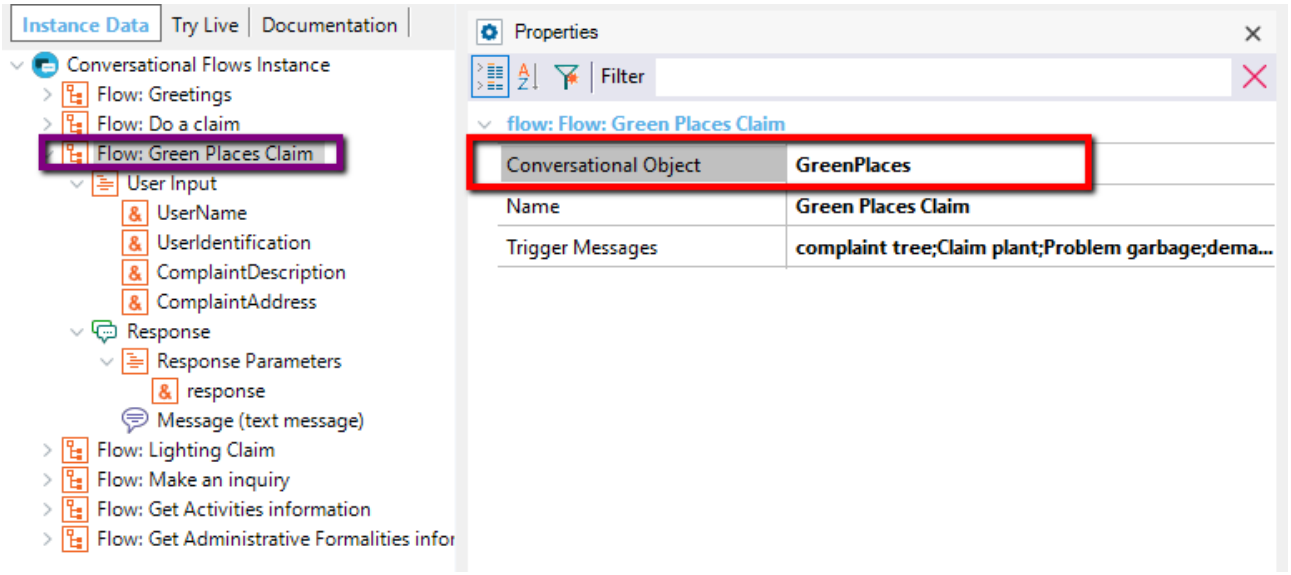

Android Emulator - GeneXus-API26-X86:5554
6:12
ConversationalFlowsChatbot
Please tell me your user identification first.
9/11/18 6:07 PM
12345
9/11/18 6:08 PM

Note: You can search for Citizen users by running the Backend Home panel.

Type for example "There is a broken pipe" and the bot will respond:



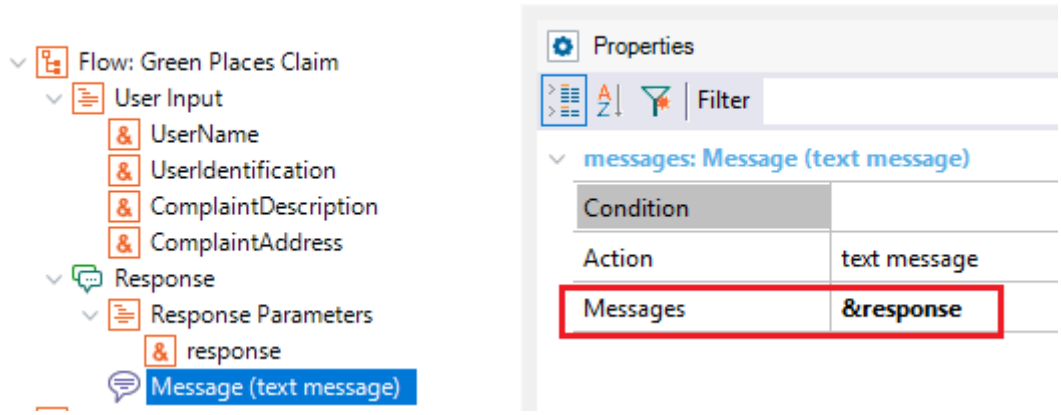
The “Green Places Complaint” Flow is a simple flow where, once the data has been entered, the progress of the response takes place by way of the execution of a procedure given in the “Conversational Object” property of the Flow. It is the GreenPlaces object.



Open the GreenPlaces object and note that it receives as parameter a subset of the inputs requested to the user which are used to record the user's complaint.

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

You can see the output parameter, &response, in the Message property of the Response node. This means that the output of this Flow is simply the response returned by the GreenPlaces procedure.



It's that simple!

|Summary: How is the response implemented? When the user is done entering the required data, the object given in the Conversational Object property is executed.

Note and test:

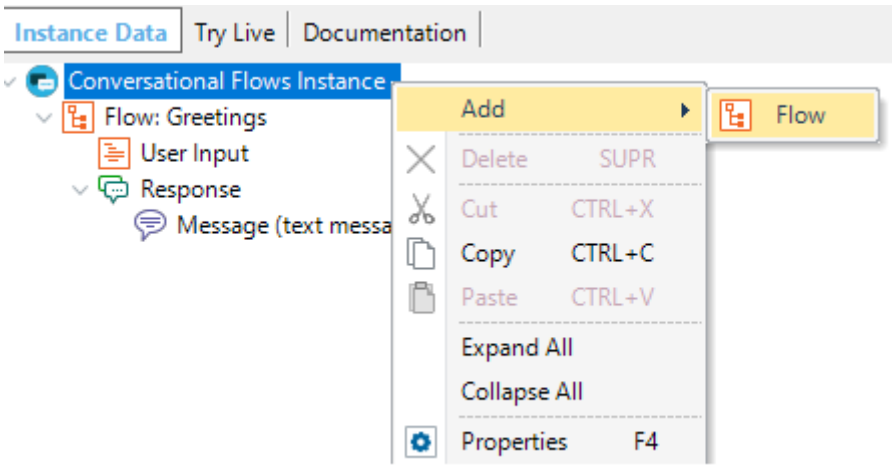
The user may enter directly to the “Green Places complaint” flow or to any of the claim flows. This means that the user need not be requested to enter the reason for the complaint. This may be included directly in the query. Try entering “I want to make a complaint about a lighting issue.”

In the query, you may include any of the inputs that map with an entity, and you will not be asked again. For example: “I’m user 12345 and I want to make a complaint about missing lights on the street.”

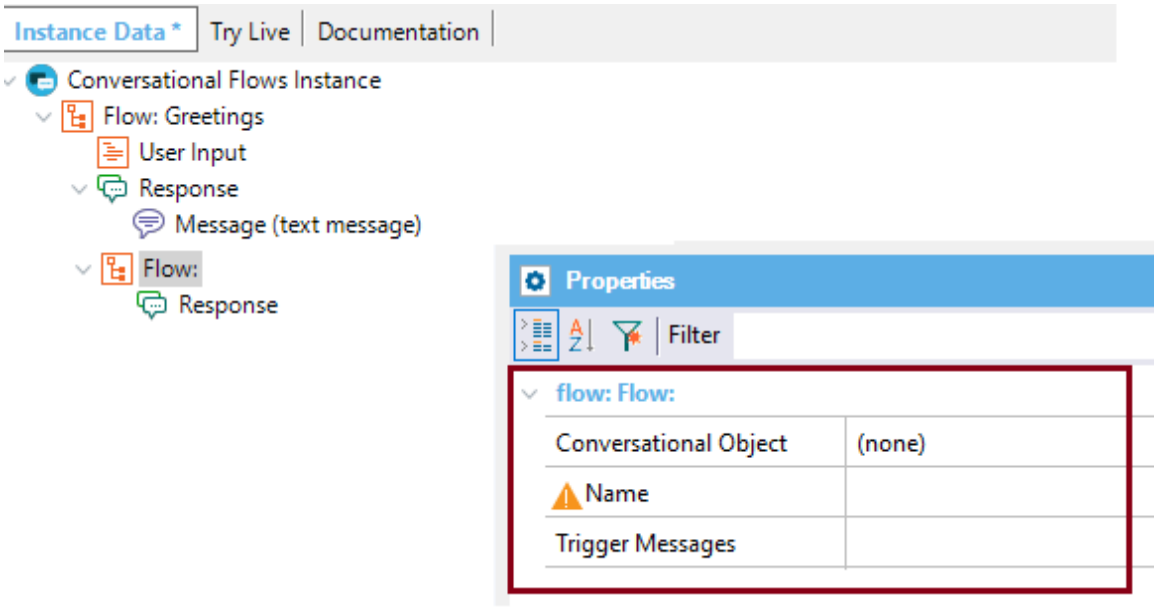
OPTIONAL: VIEW CULTURAL ACTIVITIES

Now you will add a new intent to the chatbot, so that the user can view the cultural activities where it is possible to take part.

Add a Flow called “Get Activities information.” To do this, right-click on the parent node of the “Conversational Flows Instance” instance and then Add -> Flow.

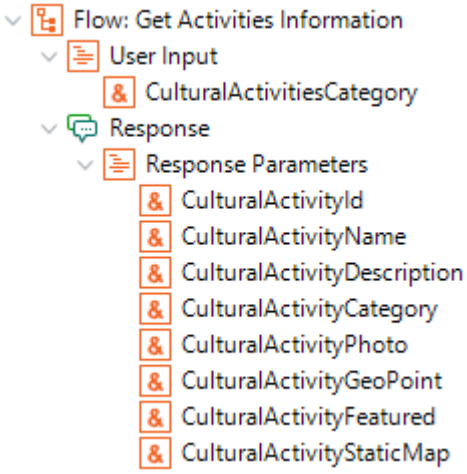


There you will have to fill in the Name del Flow (“Get Activities information”).



In the **Conversational Object** property, set up Data Provider "CulturalActivitiesNews."

You will see that the **User Input** and the **Response parameters** are automatically completed. This is done by GeneXus, based on the data of the **Conversational Object** selected.



Open the CulturalActivitiesNews Data Provider and you will see what it consists of and the parameters that it receives and returns. Basically, it receives a category of activity (art, culture, nature), and it returns the activities according to that filter.

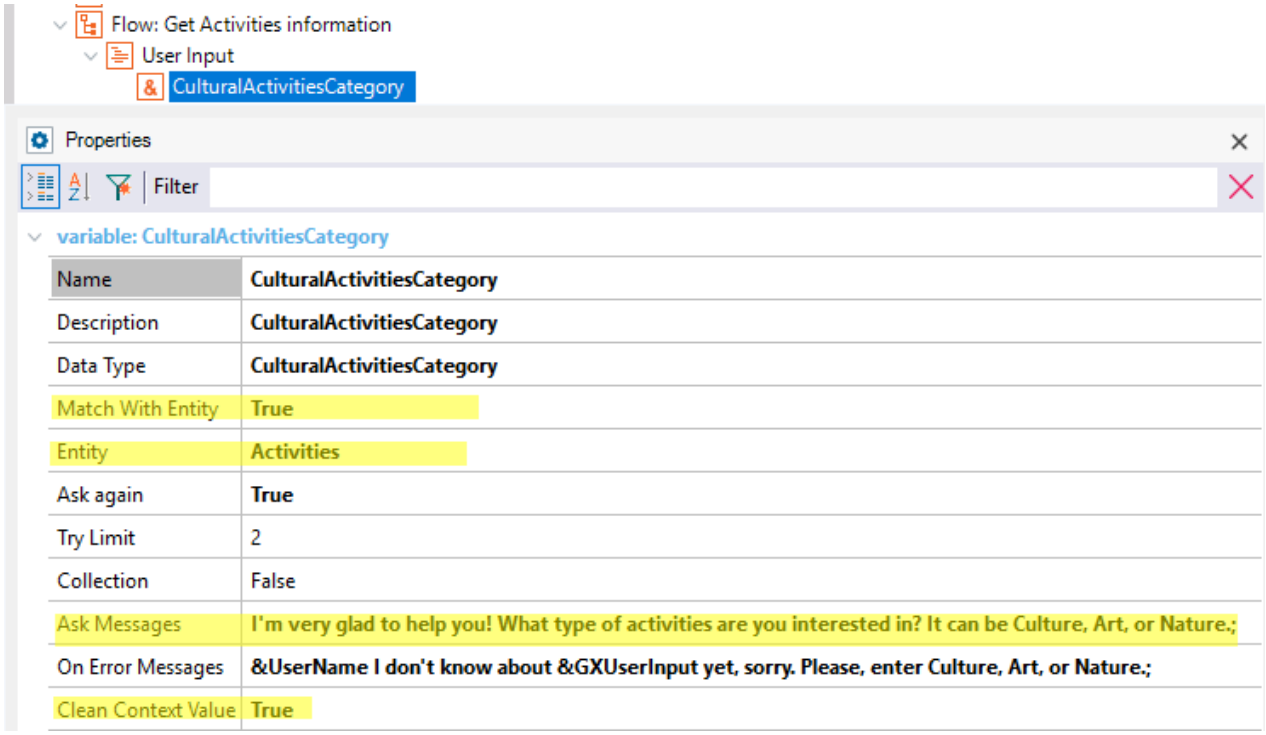
```
parm(in:&CulturalActivitiesCategory);

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

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

Now, back in the Flow, select **User Input** "CulturalActivitiesCategory."

Set up the following properties:

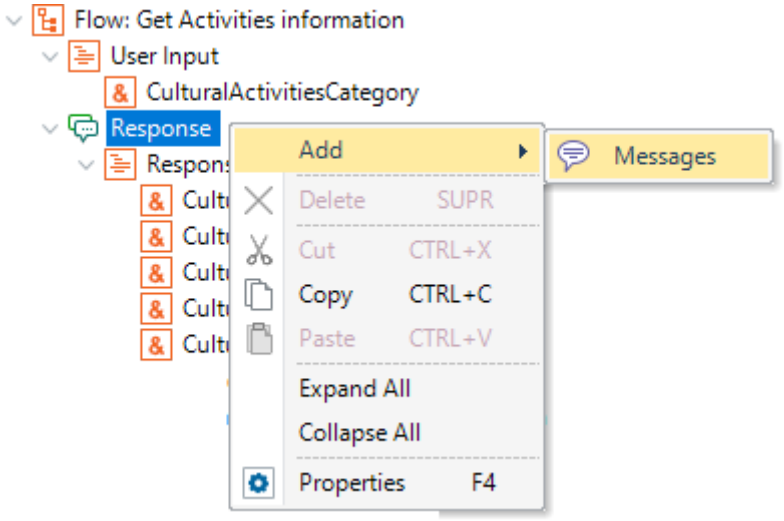
- Match with Entity = TRUE
- Entity = Activities
- Ask Messages = I'm glad to help you! What type of activities are you interested in? It can be Culture, Art, or Nature.
- Clean Context Value = TRUE



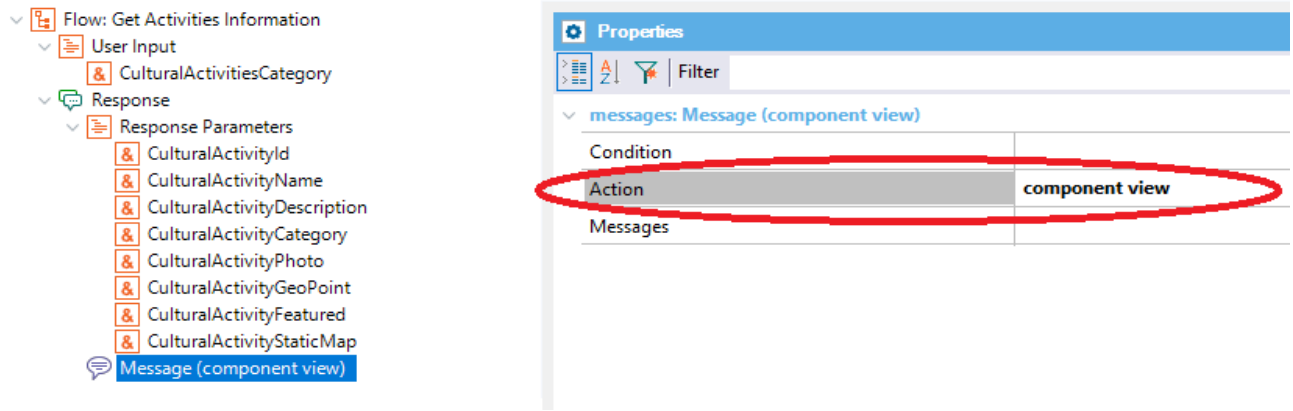
As result for this Flow, you will show the user a horizontal grid with the activities.

Add a **Message** whose **Action** property is "Component View," so that the output of the response is a component.

To add the Message, right-click on the Response node and then Add -> Messages.



In the recently created Message node, edit the Action property and change its value to “component view.”

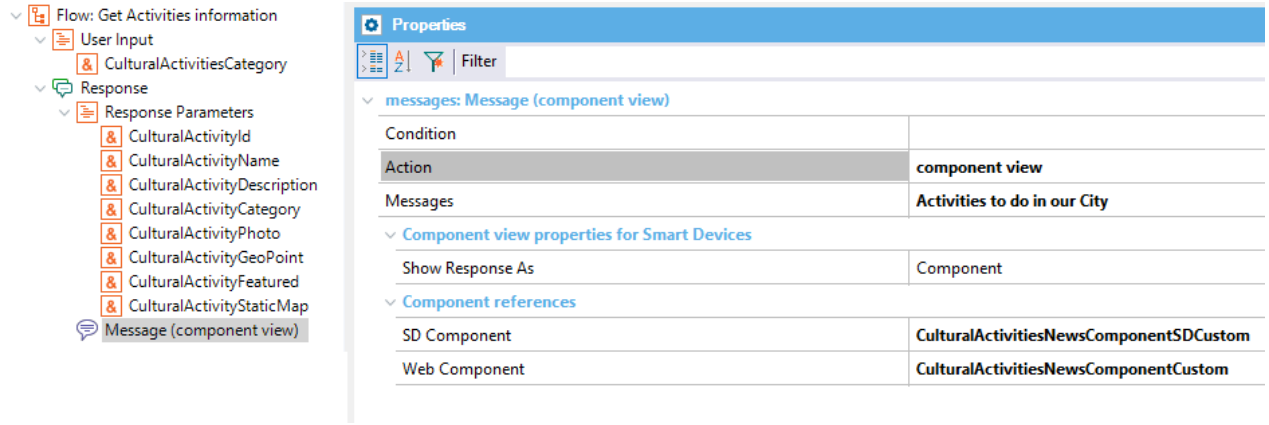


Edit the messages property and type "Activities to do in our City."

What is left to do now is indicate which component should be shown as result for this Flow, where we will have the horizontal grid that deploys the data.

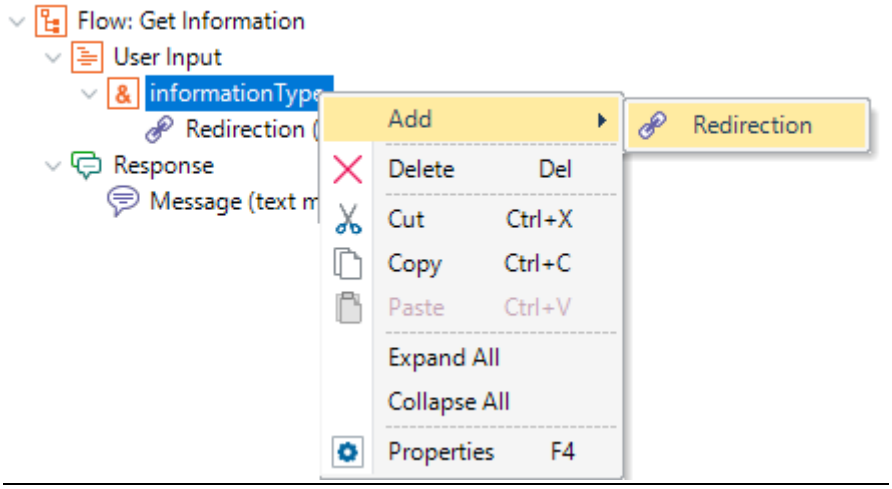
You could use the object generated automatically for this purpose (indicated in the Generated web Component property) but you will use your own.

Set up the **Web Component** property with value “CulturalActivitiesNewsComponentCustom”, and the SD components property with value “CulturalActivitiesNewsComponentSDCustom”, as shown in the figure below:

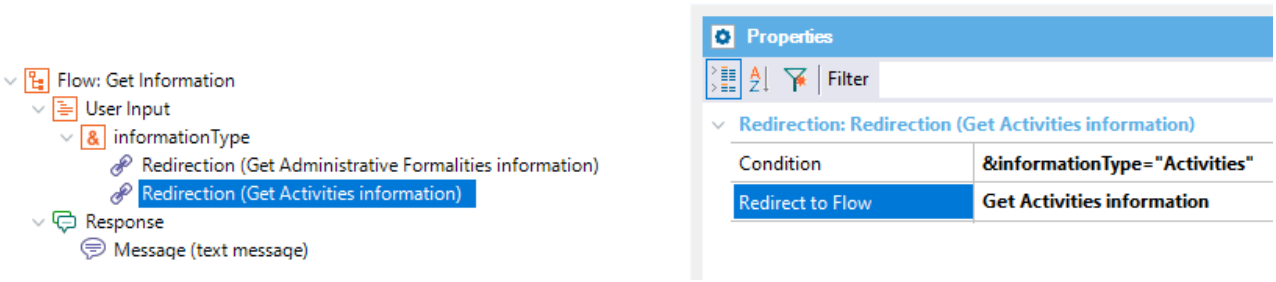


To complete the exercise, edit the “Get Information” Flow and add a Redirection node to redirect the Flow you've just created.

You have to right-click on the InformationType node of the “Get Information” Flow and then Add -> Redirection.



Then, complete the following (keep the casing when entering text in the Condition):



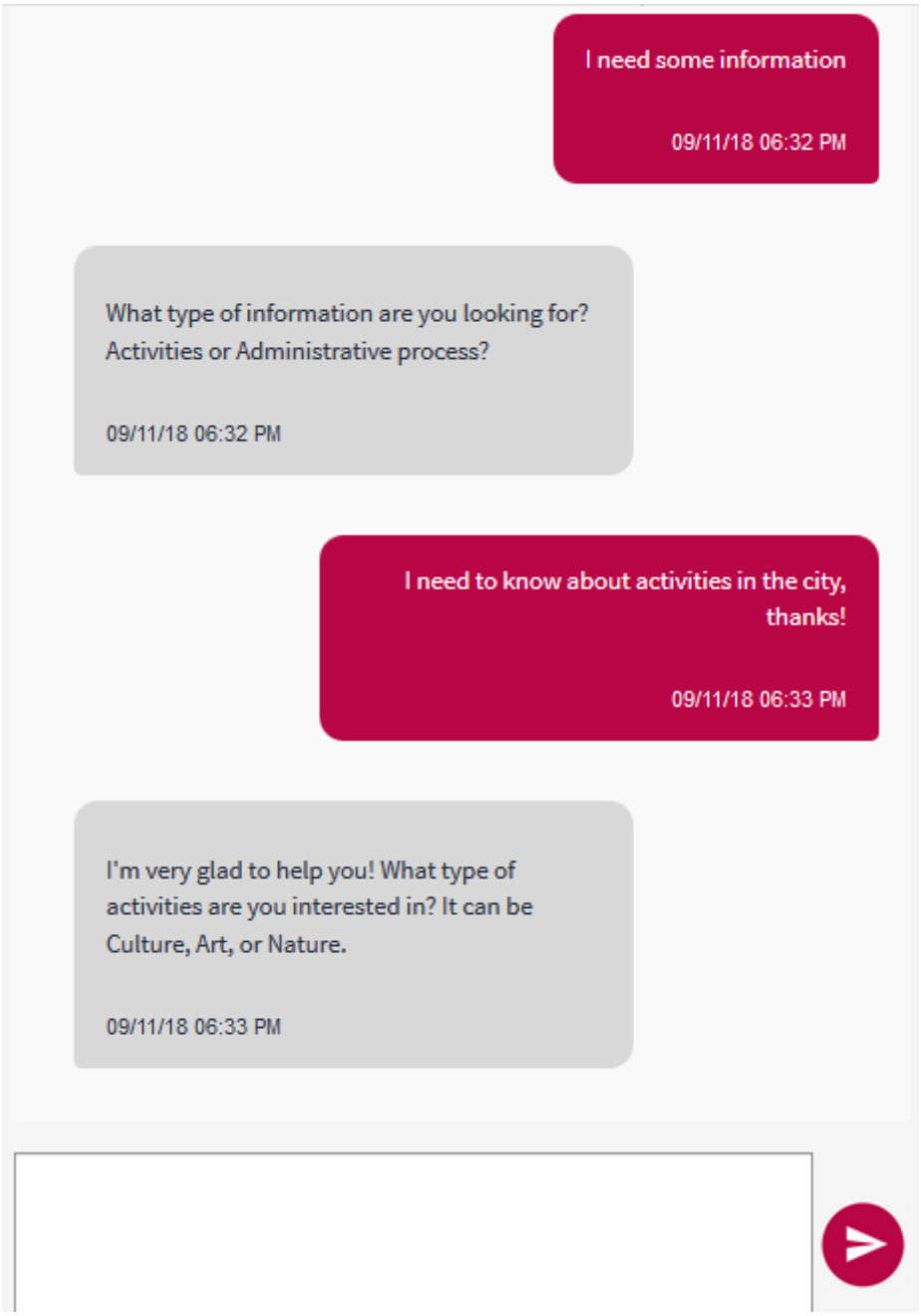
So, if the user enters through the “Get Information” Flow, he/she will be able to go, automatically, to the “Get Activities Information” Flow after requesting the desired data and indicating that it is about “Activities.”

Ready!

To execute you must do the following:

1. Right-click on the instance Citizen -> Generate Chatbot
2. Run the webpanel CitizenChatbotWebUI and the panel CitizenChatbotSDUI to see the result.

Note: if this doesn't work, as a workaround modify the handler object to be main and rebuild it.
Then remove the main property and run step 2 again.

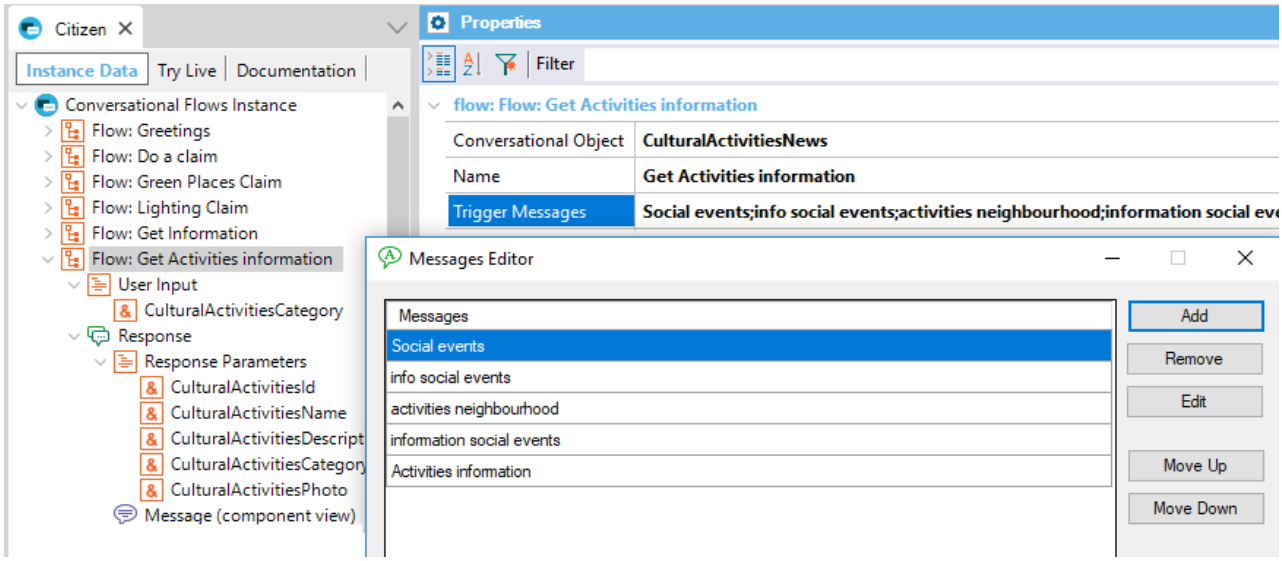


Choose one and you will see the panel with the horizontal grid.

It would be even better if the user were allowed to indicate directly the topics of his/her choice on which information is required, without the need to ask the user if it is not necessary.

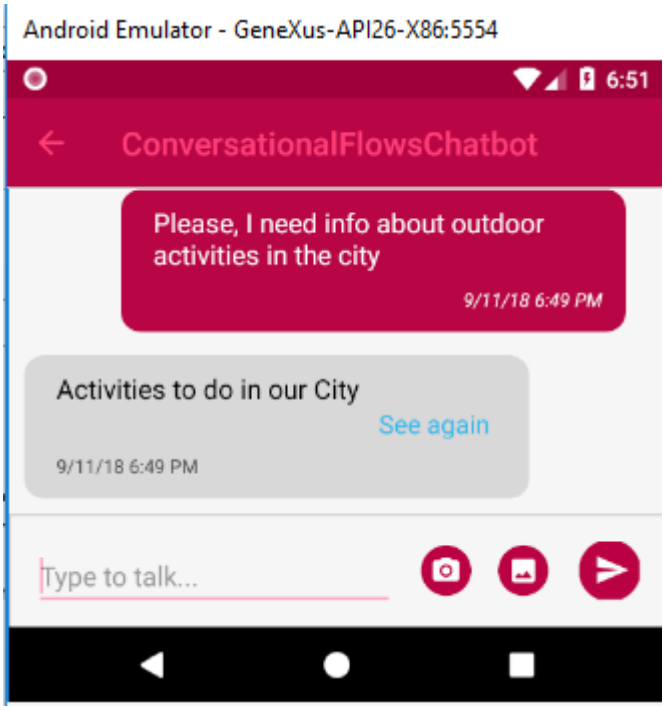
This means having to train the chatbot to allow direct entry to the “Get Activities Information” Flow. So, add a **Trigger Message** you consider appropriate so that Watson will learn how the user may enter in this flow directly without the need to arrive from another flow.

For example:



You can try this by entering the query “I need information about outdoor activities!” (this is an example, and you are free to use any other query that is semantically equivalent).

Note that, in this query, the activity type, “outdoor,” has been added as well.


Android Emulator - GeneXus-API26-X86:5554
6:51
ConversationalFlowsChatbot
Please, I need info about outdoor activities in the city
9/11/18 6:49 PM
Activities to do in our City
See again
9/11/18 6:49 PM
Type to talk...

ANNEX: TRY LIVE

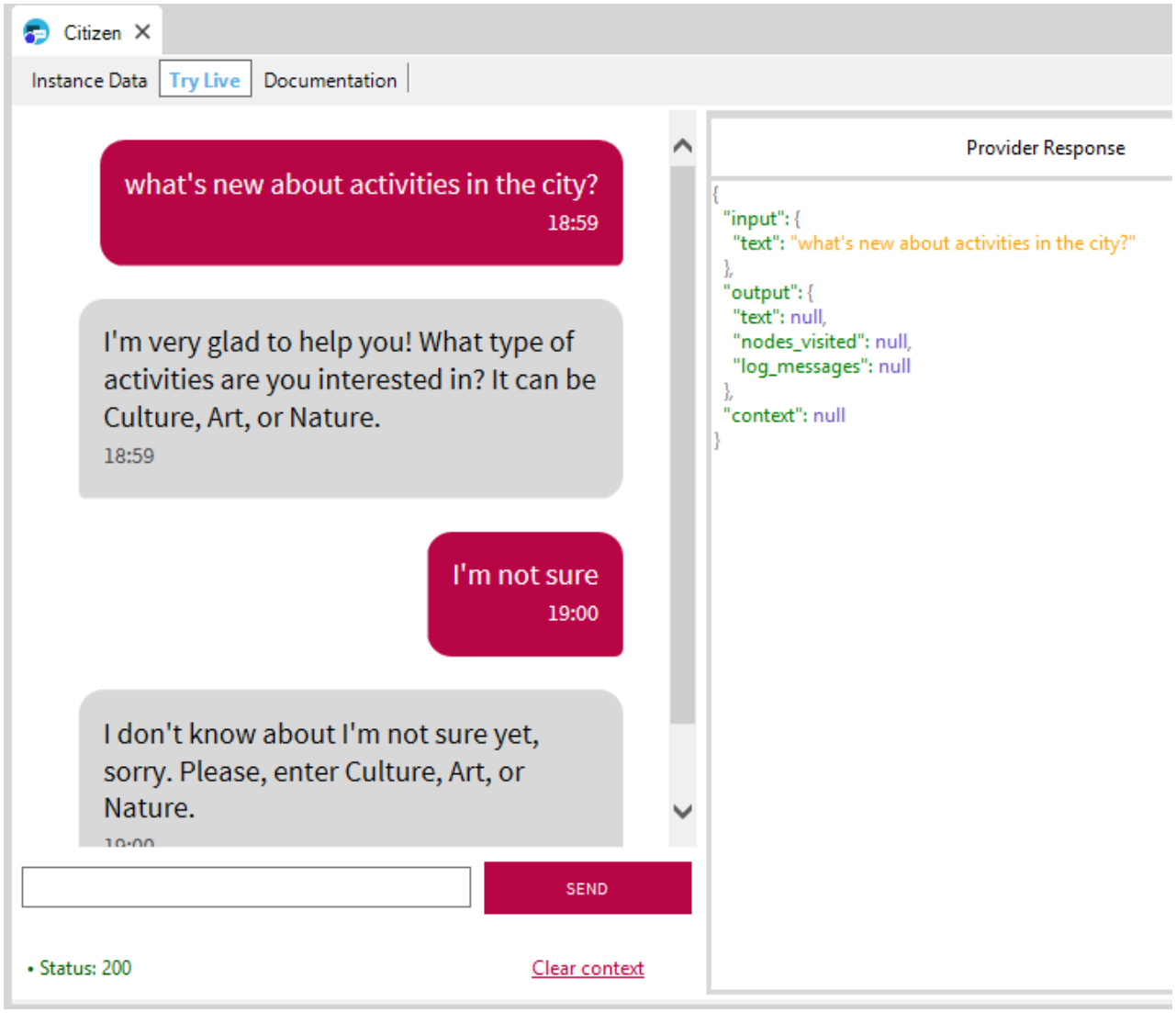
This section is optional.

In GeneXus, you may test your chatbot's dialog using the Try Live tool.

Try Live is a tab that is next to the Instance Data tab where you may try the direct response of the NLP provider.

You may try any query related to what you have been working on in your chatbot.

For example:



Note:

The Try live feature enables you to try the recognition of intents, and the execution of the flow, as far as there is no need to call a GeneXus object. In that case, the outcome that you will view on screen is the procedure's output variable.

REMEMBER THAT: CHATBOTS MUST BE TRAINED TO ACHIEVE AN OPTIMUM PERFORMANCE THAT WILL INCLUDE ALL POSSIBLE USER QUERIES.

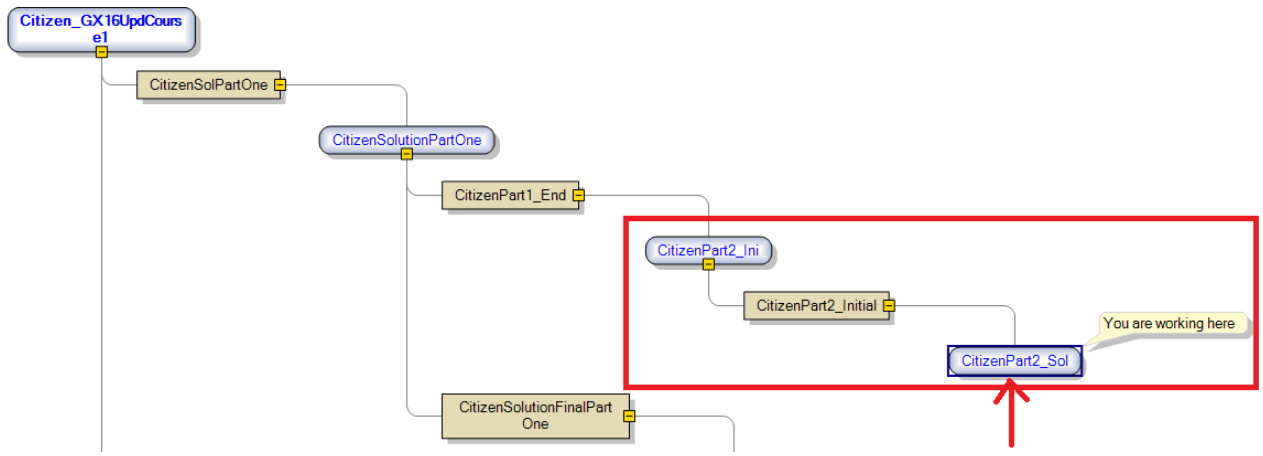
Training a chatbot implies adding Trigger messages to the intents so that these will be recognized, with lesser chances for ambiguities.

DOCUMENTS

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

SOLUTION KB

You may download the solution KB for this practice session from GeneXus Server in order to compare results. It is the version of the KB called CitizenPart2_Sol_u4.



We have removed the UserName and UserPassword. Configure the appropriate values to be able to execute it.