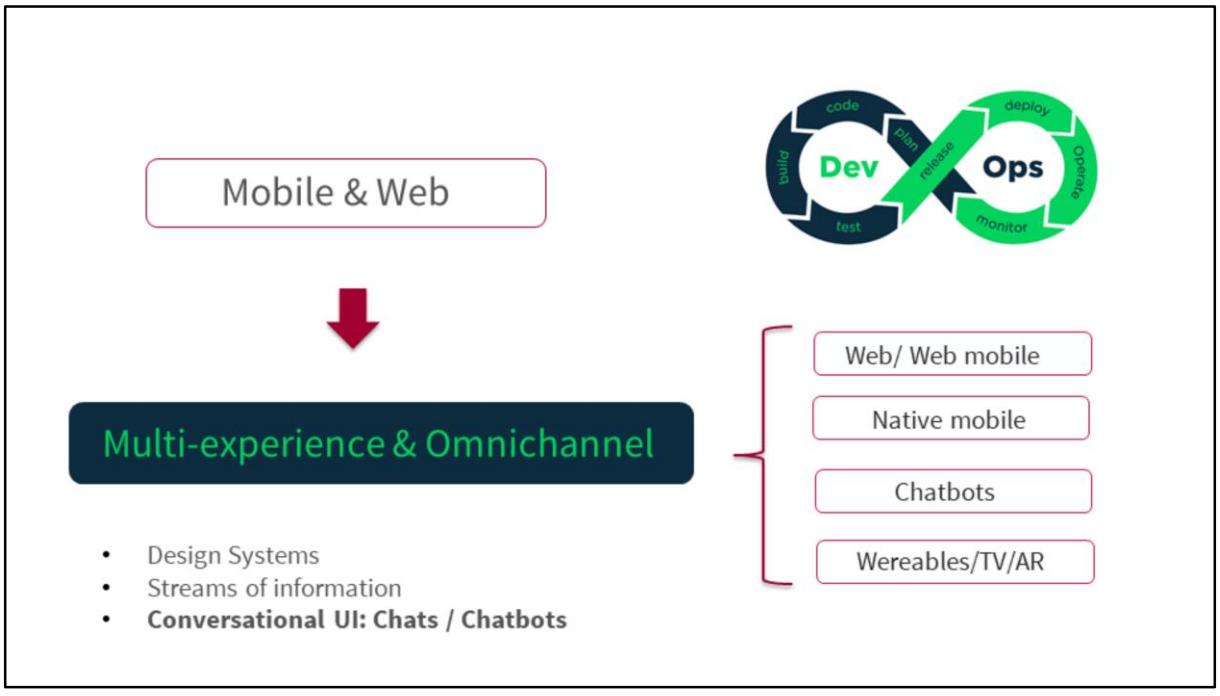
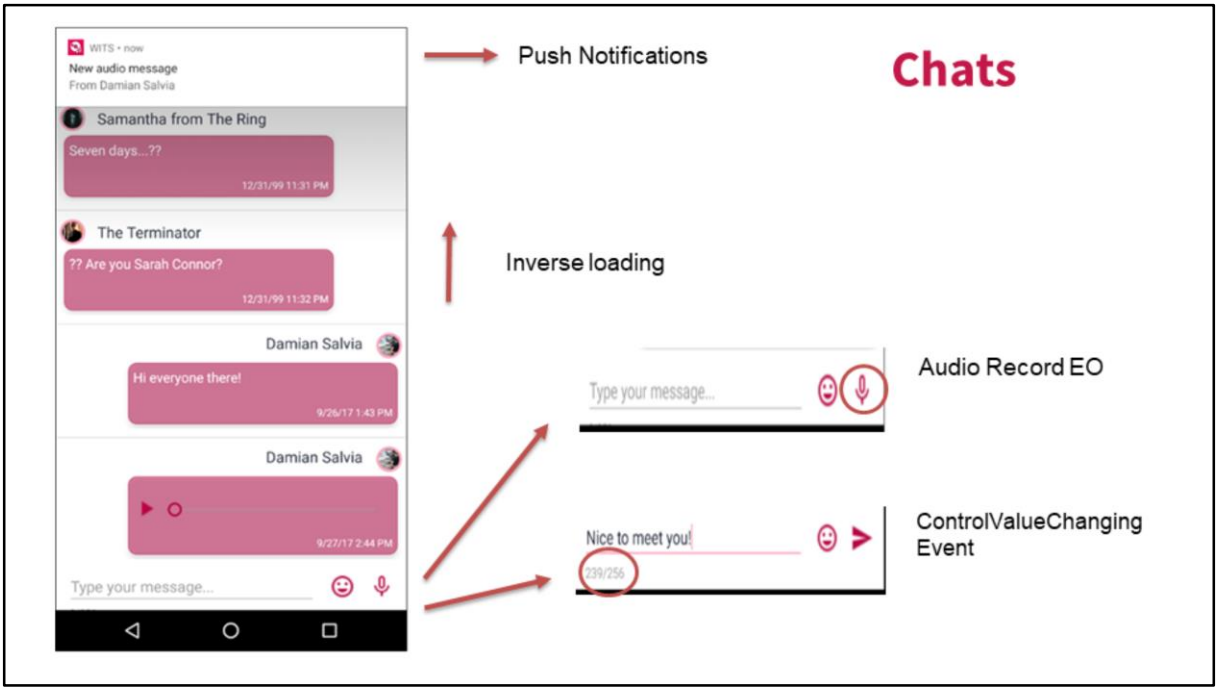

GeneXus™
The power of doing

Conversational UI:
Chats / Chatbots

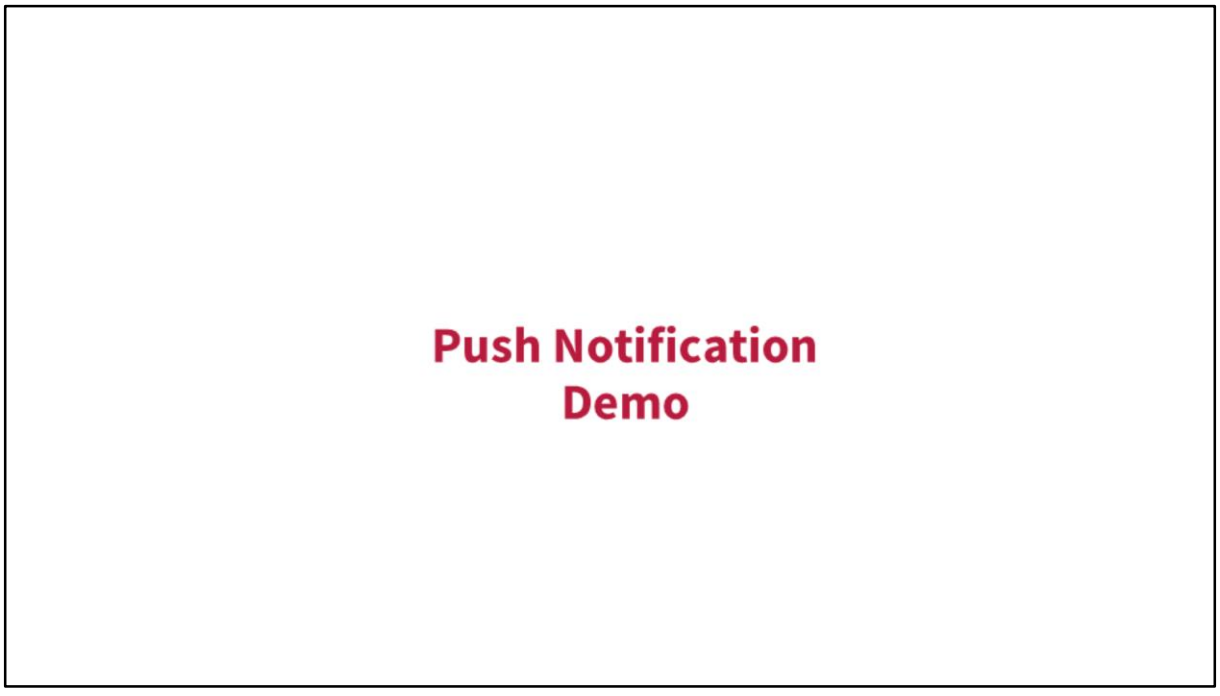
GeneXus 16



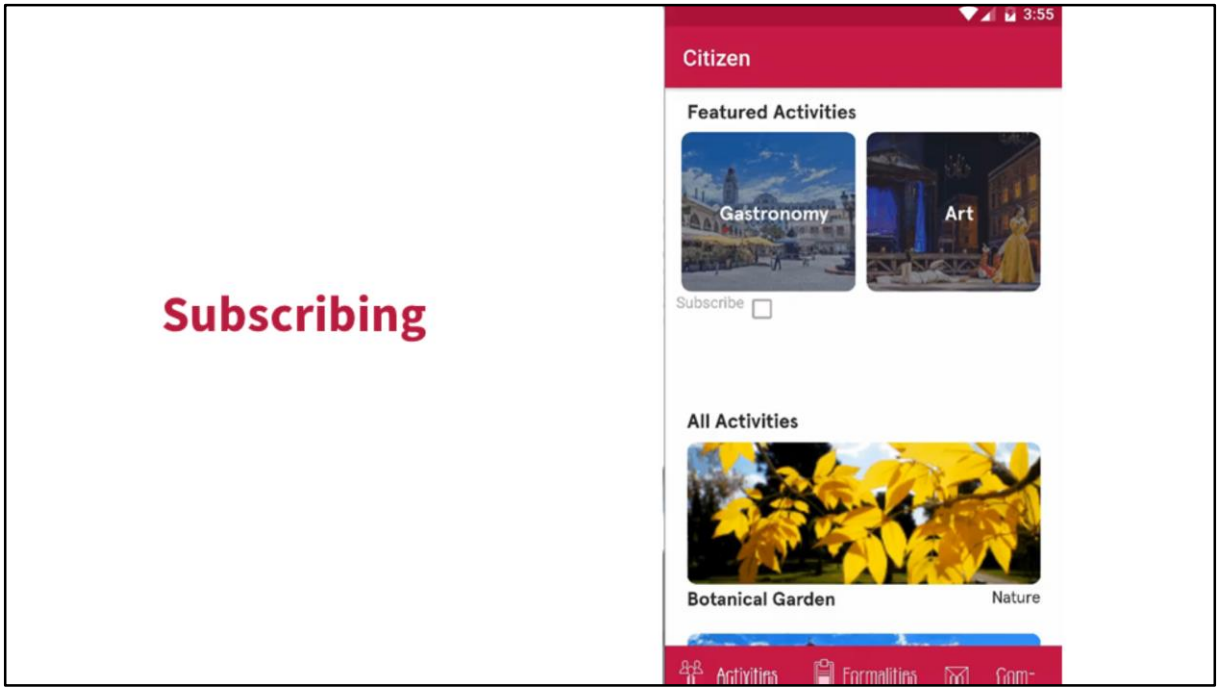
Now we'll see the next point: the features to implement Simple Chats and the facilities to implement Chatbots.



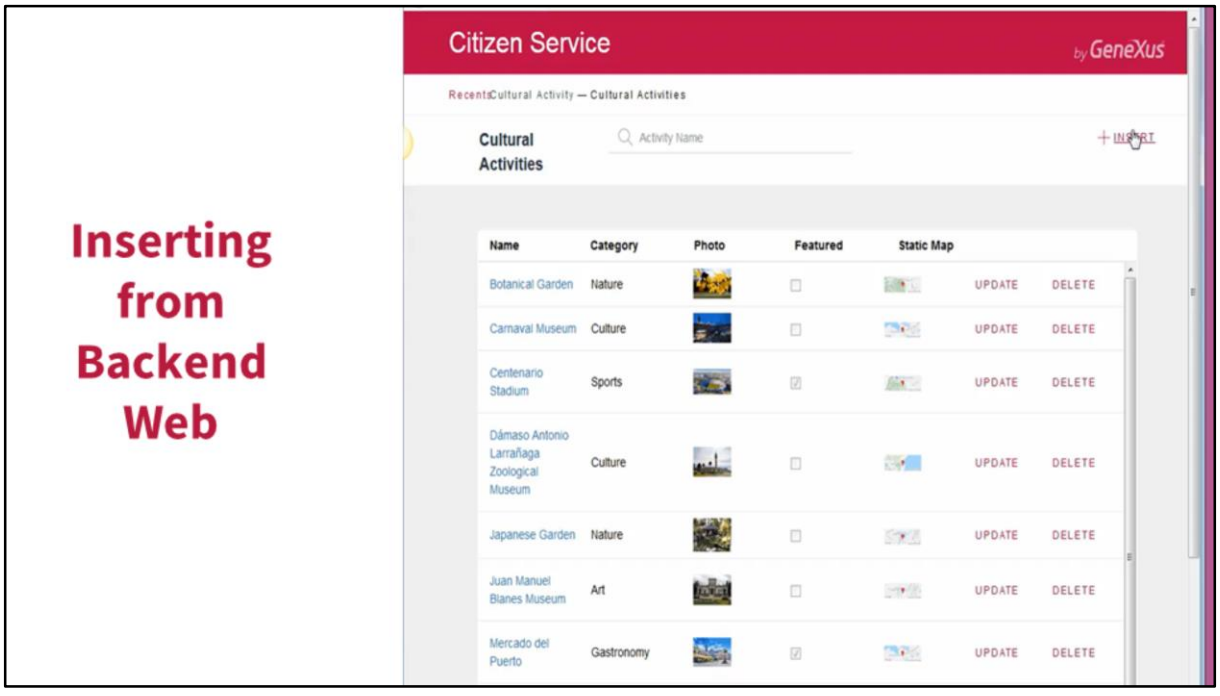
To implement a simple chat we have:
- Inverse Loading, which is a grid property that allows you to view the latest information from the bottom up.
- Audio Record, an external object that allows you to record an audio
    - Control Value Changing is an event that lets you know if the user has changed the value of an editable field (numeric or character), in this example it can be used to count the characters of the message to be sent.
- Push Notifications, that allow you to inform the user about some event by means of an alert, in this context, we inform that there is a new activity of interest.



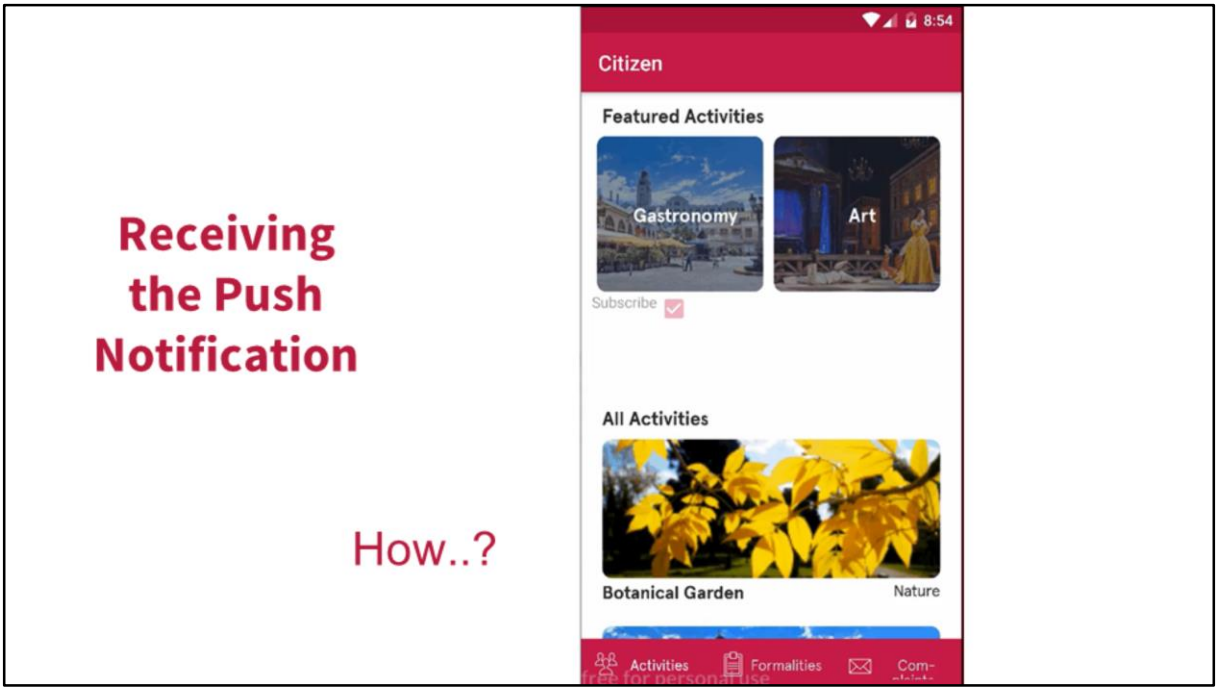
Now, we'll focus on Push Notifications and the implementation of the Socket API; later on we'll talk about the new features of version 16.
Let's see an example in execution.



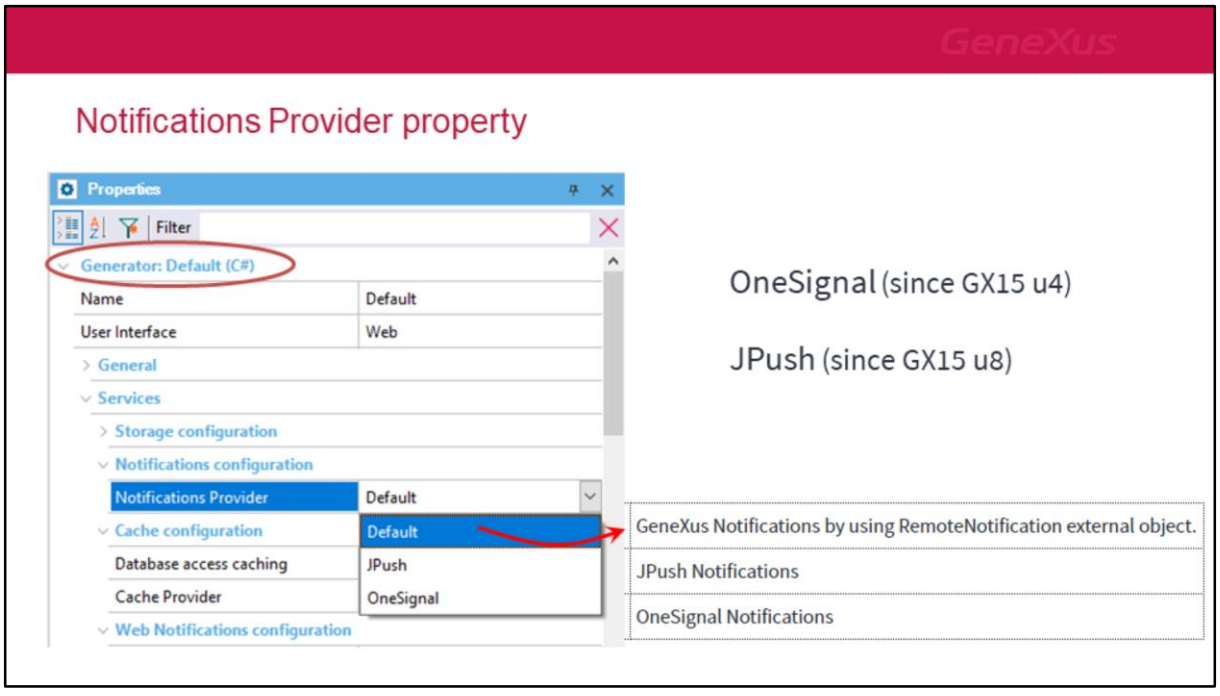
In the Citizen application, users can subscribe to receive notifications of new activities.



Afterwards, someone inserts a new activity from the web backend and after saving, the notification is sent to the subscribed users.

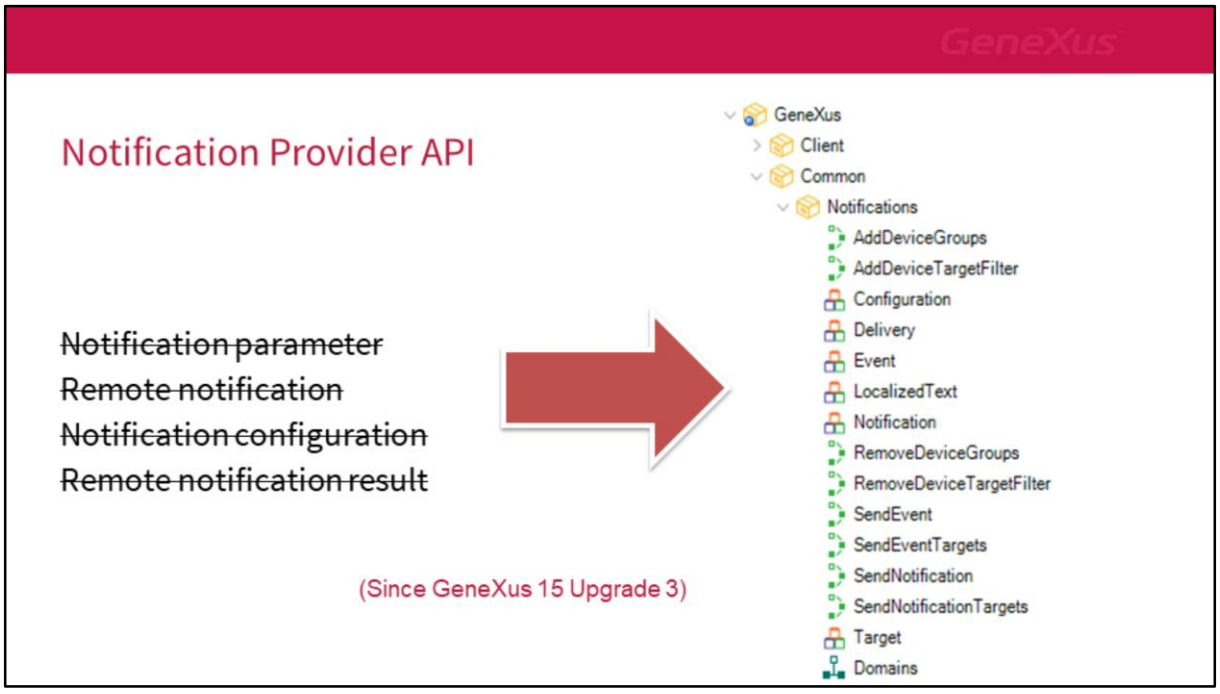


The user receives the notification in his/her device and when tapping on it, he can see the detail of the new activity.
And how is this done?

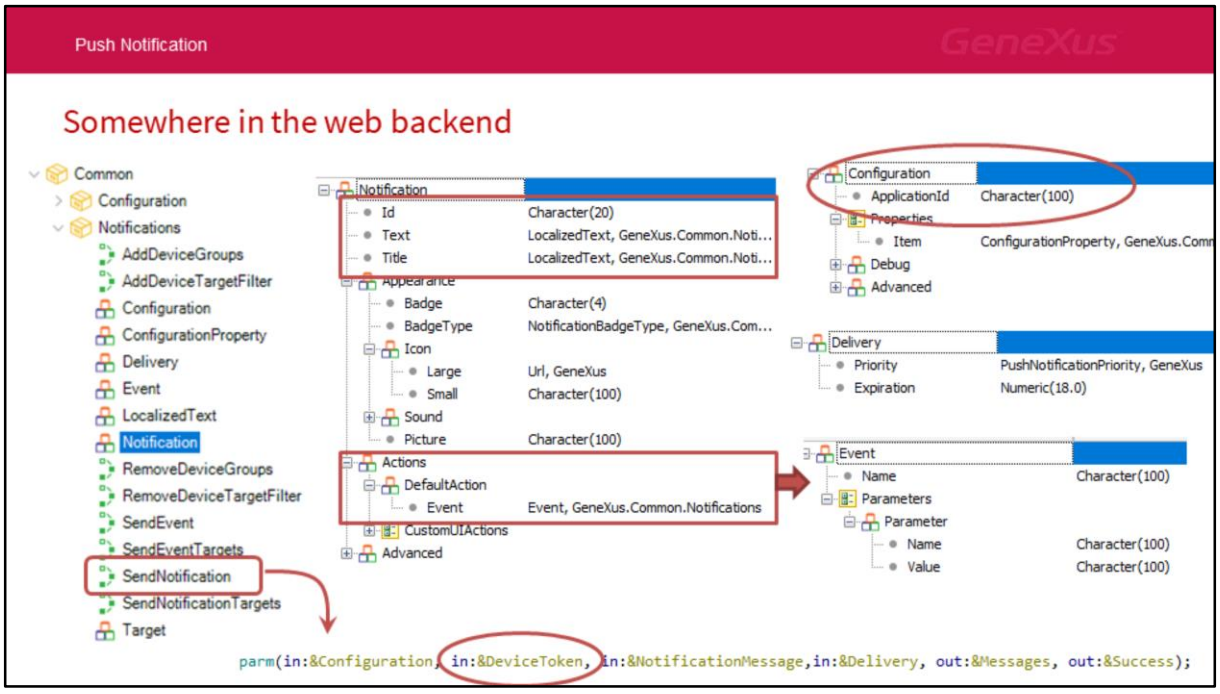


For this we have the property Notification Provider (Gx15 U4) at the web generator level, allowing us to choose the external provider with which we will handle the notifications. When you choose one provider or another, several properties will be displayed for its configuration.

Note: The main difference between OneSignal and Jpush is that the latter is used for sites where Google service is not supported, mainly used in app for China.



Previously we used several External Objects to handle Notifications; now we unify them into a single API called Notification Provider, which can be found in the module GeneXus-Common-Notifications.



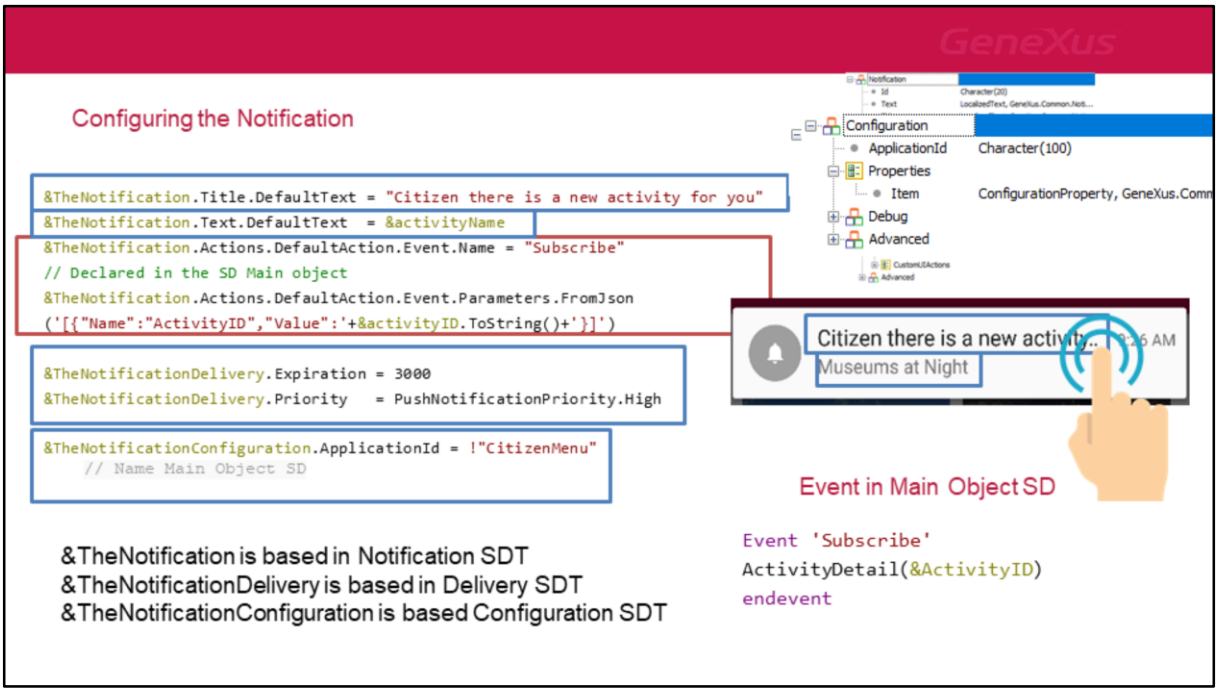
With this API we can configure the notification using variables of SDT type and sending it from the web backend.
Let's stop to analyze the SDT that will be used:

-The Notification SDT allows configuring the content of the notification as well as the action that will trigger some event, such as opening an sdpanel sending parameters.

- Configuration allows you to indicate the name of the main SD object where the notification is executed in its Application ID item.

-Delivery allows you to set the notification priority and expiration priority.

Finally, there are some procedures to be able to send the notification, for example Send Notification, which is in charge of sending the notification to the device with the configurations made.



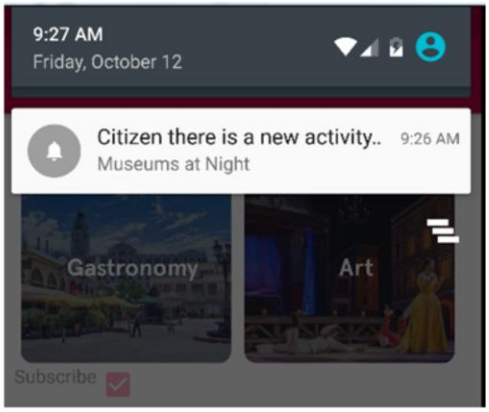
Within the notification you can configure:
-Title
-Text
-The event that was defined in the main sd object and the one in charge of opening the Activity Detail once the user tapped on the notification.

Next, set the priority and expiration of the notification.

Lastly, configure the application ID which is the name of the main object.

Sending the Notification

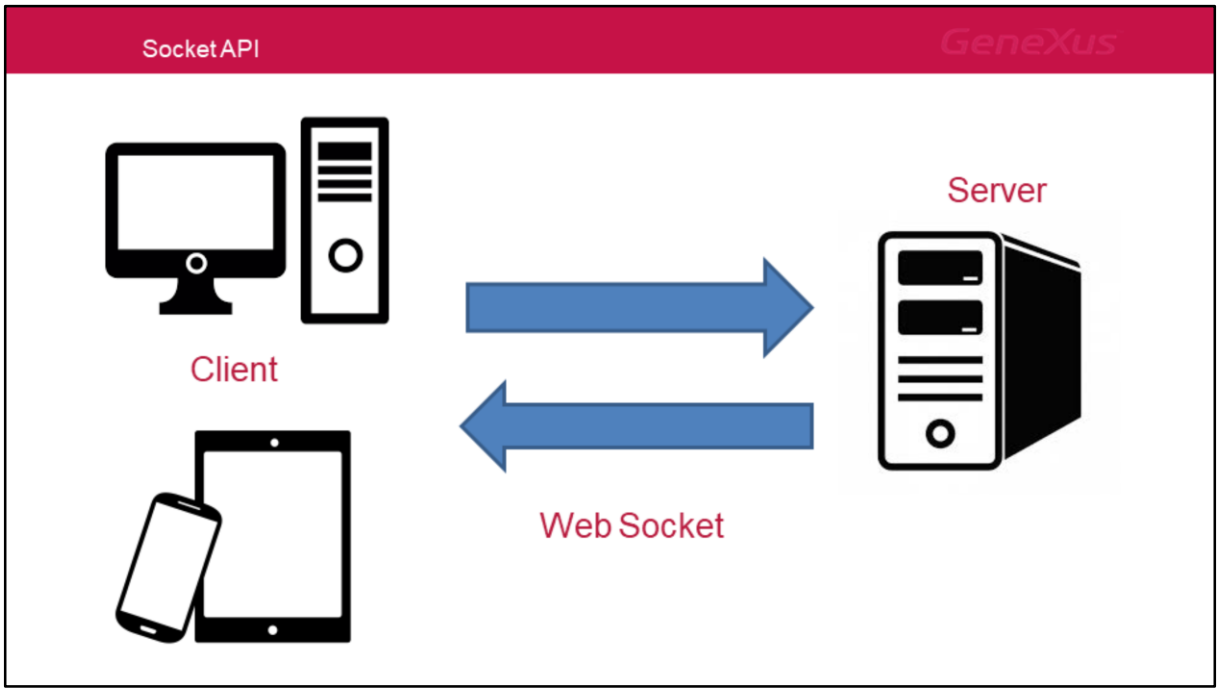
```
For each Device
   Where DeviceSubscribe= true

  GeneXus.Common.Notifications.SendNotification
  (
       &TheNotificationConfiguration,
       DeviceToken, // Target device token
       &TheNotification,
       &TheNotificationDelivery,
       &OutMessages,
       &IsSuccessful
  )
endfor
```



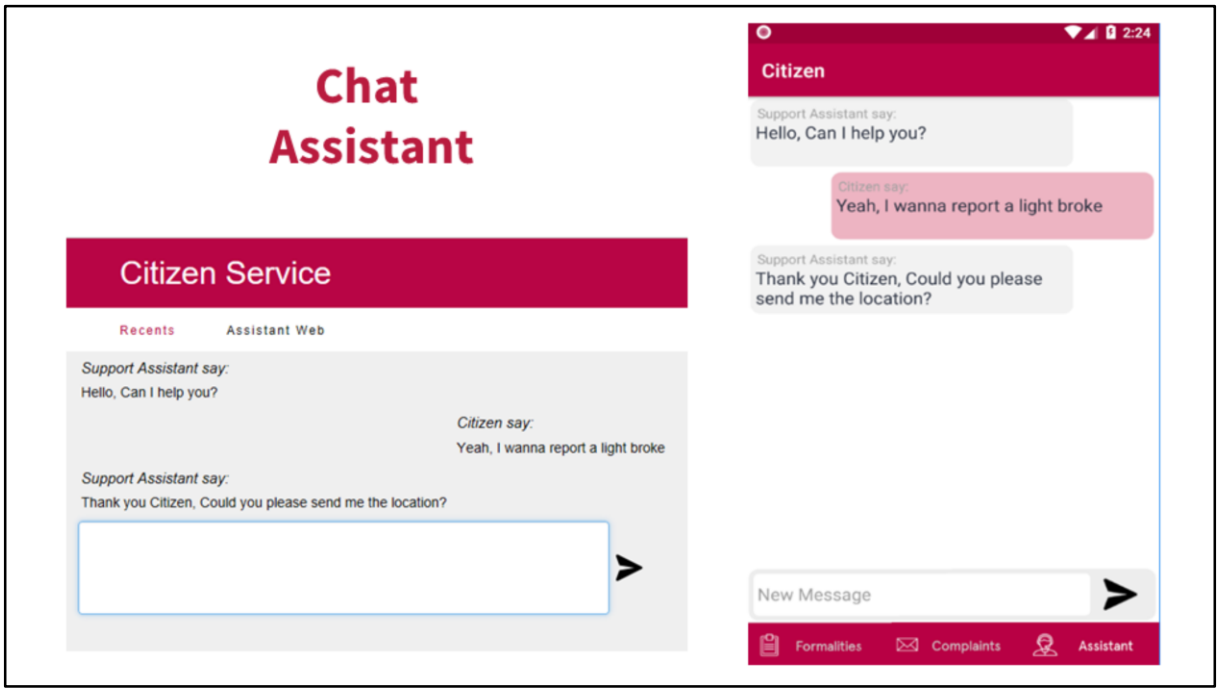
Lastly, you must call the SendNotification procedure passing the variables that you configured and indicating the device that will receive this notification; in this example, the users that subscribed.

Socket API

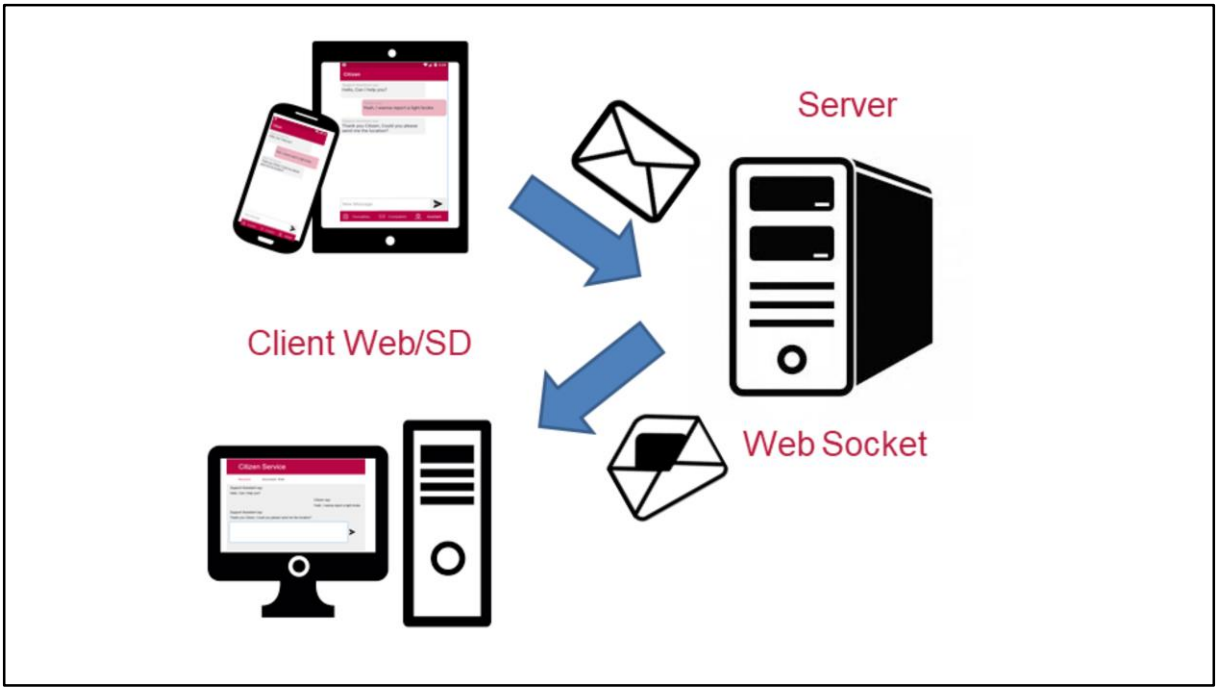
Let’s move on to the following topic…



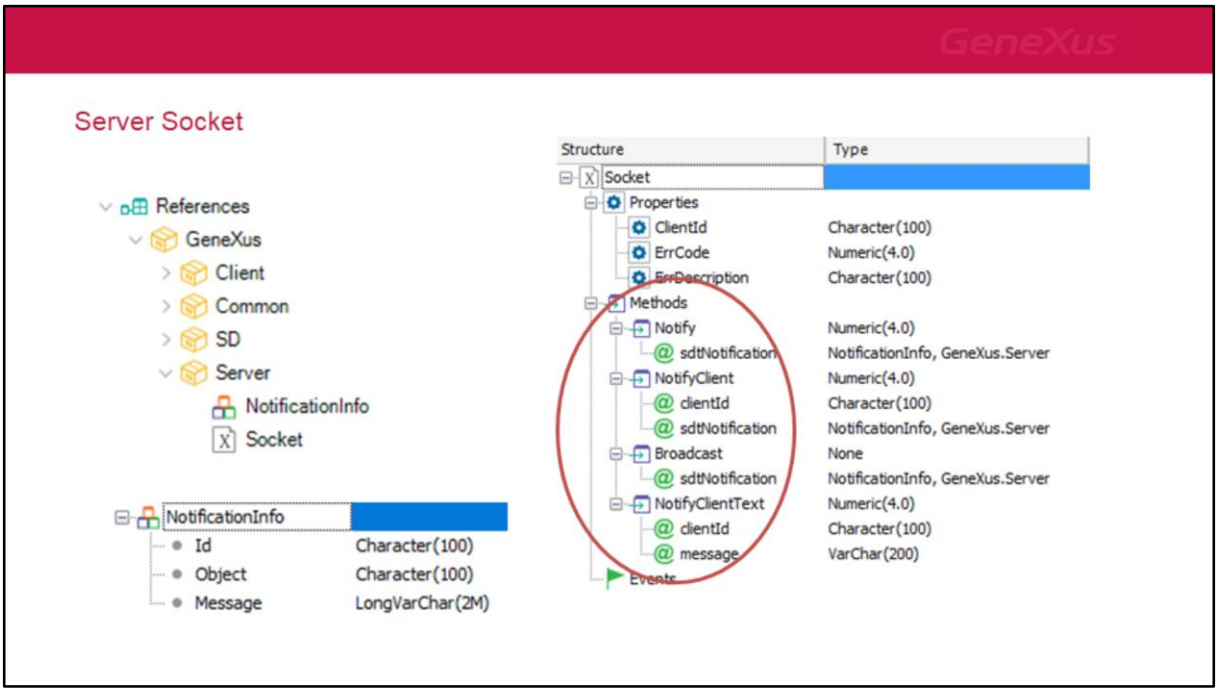
Regarding the Socket API, note that it allows establishing a two-way communication between the client and server through web sockets, which are a communications protocol that provides a communication channel over a tcp connection.



An example of use of the Socket API is an Assistance chat, which in this example allows the user to make a report in real time from his mobile and a wizard will support him from the web part.



It works as follows:
-The SD client sends the message to the Server
-The server sends it to the WEB client
- The web client will refresh its screen to show the new message
- And vice versa.

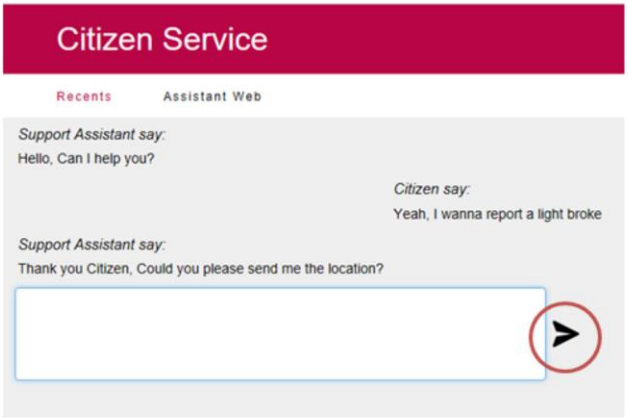


For this we have a Server Socket, which allows you to establish the communication from the server to the clients, sending a notification using some methods, such as:
-Notify allows sending a notification to the user who originated an action.
-Notify client; this method allows you to send a notification to a specific user.
-Broadcast allows you to send notifications to all users.
-NotifyClientText allows you to send plain text to a specific client.

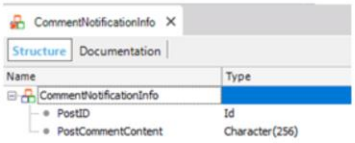
NotificationInfo allows you to specify the notification information that will be sent as notification ID, the Object that will receive that notification, and the message.

Optional
There are also some properties: clientID, which allows obtaining the client ID of the current session, error code and error description which return information in the event of an error.

Sending Message from Web

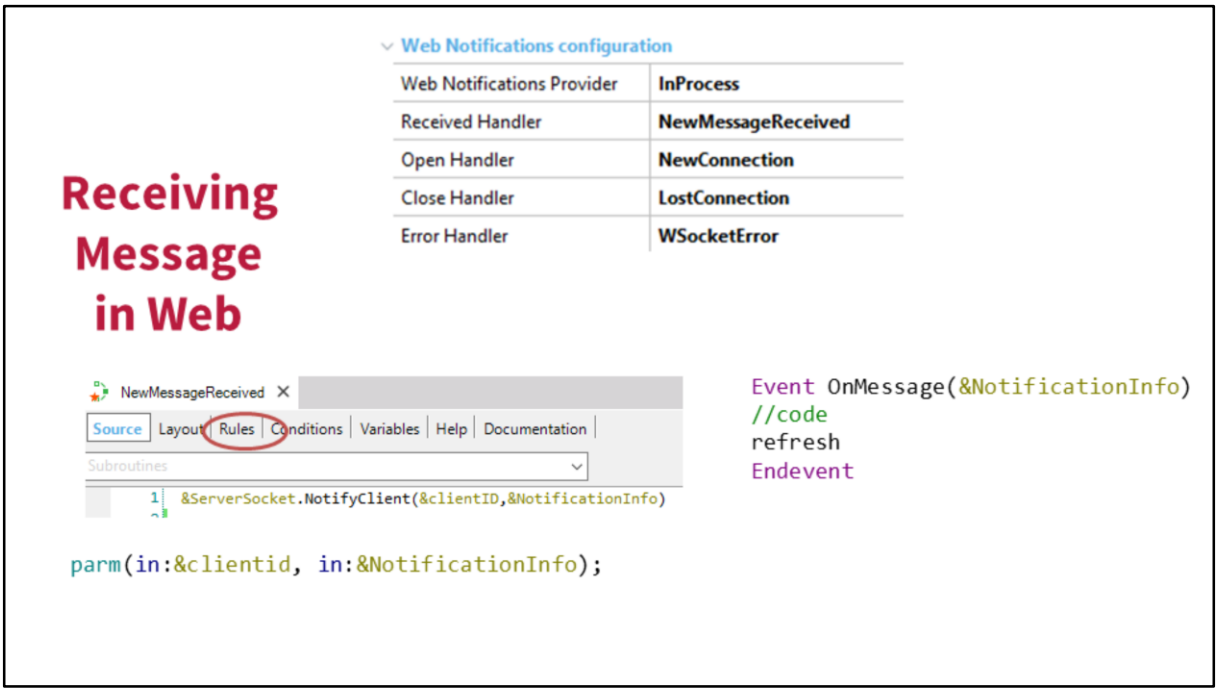



```
Event 'Send'
&commentNotificationInfo.PostId= random()
&commentNotificationInfo.PostCommentContent= &NewMessage
&NotificationInfo.Message= &commentNotificationInfo.ToJson()
&ServerSocket.NotifyClient(&clientID,&NotificationInfo)
Endevent
```

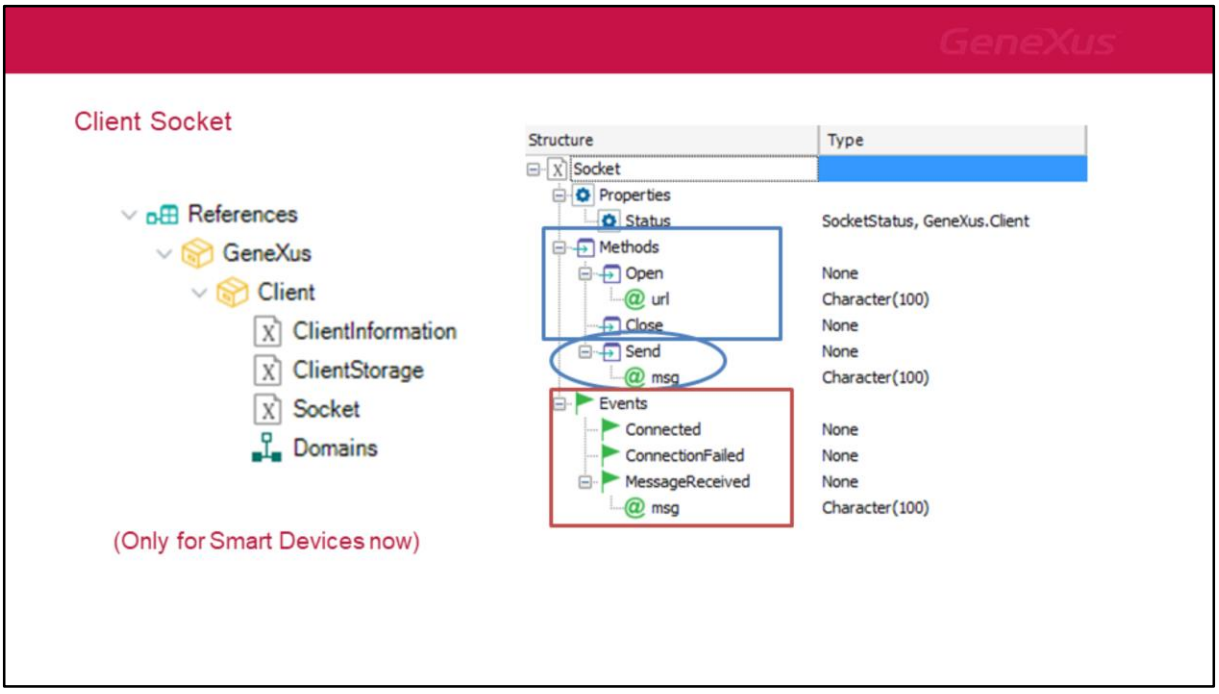


In this example, we send a new message from the Web Chat (a web panel) to the user engaged in the conversation (a Panel for Smart Devices).

Note that the SDT CommentNotificationInfo was created to send the message in json format.



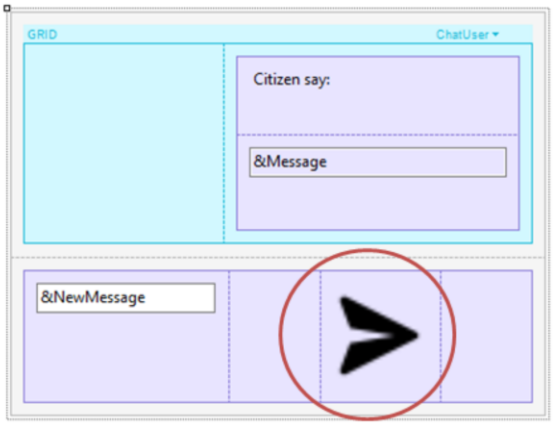
Subsequently, to receive the answer you need to configure at the KB level the procedures that will handle the connection and the message received. Note that in this case you have the procedure NewMessageReceived.
Finally to show it use the OnMessage event in the WebPanel and make a Refresh to load the new message that will be displayed.



On the other hand, for the communication between the client and the server there is a Client Socket which has methods for establishing the connection to the Web Socket and a method for sending a message.
In addition, there are events which are executed when establishing the connection or not and when receiving a message.

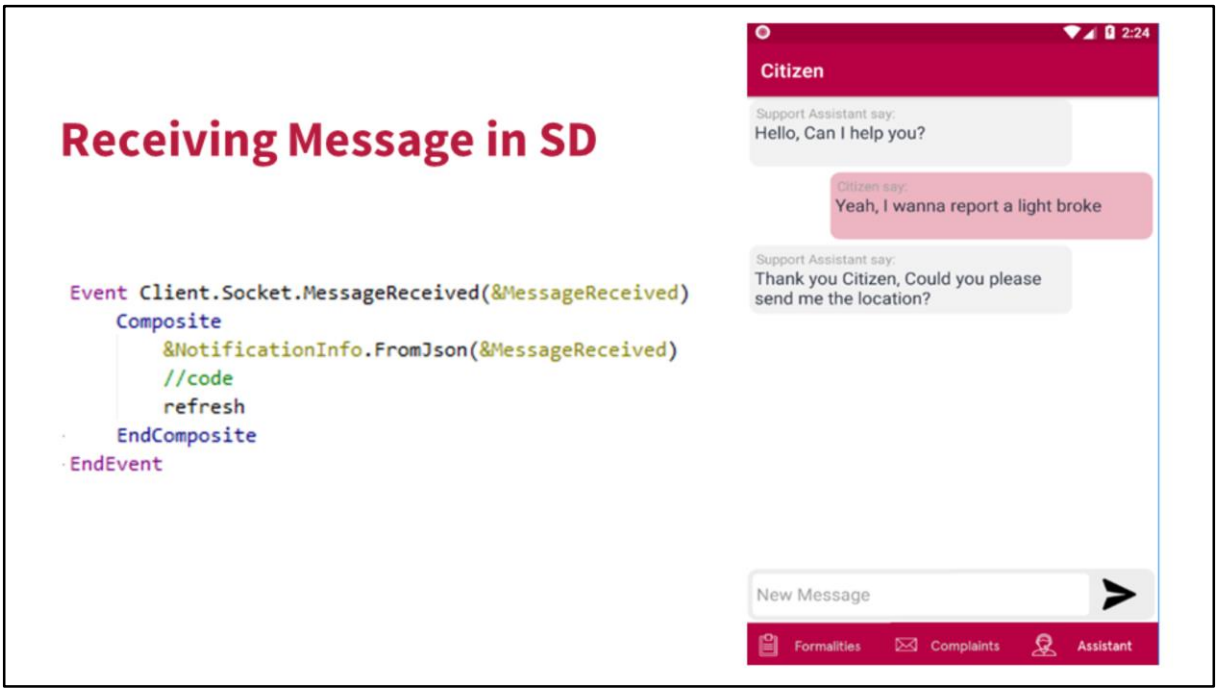
Note that Client Socket is currently available only for Smart Device Generators.

Sending Message from SD




```
Event ImageSend.Tap
    GeneXus.Client.Socket.Send(&NewMessage)
Endevent
```

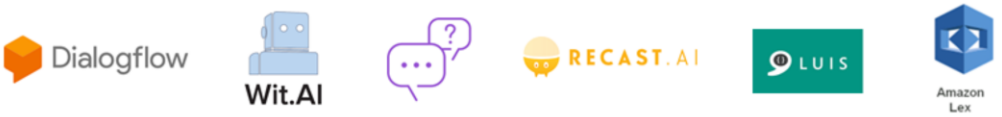
Continuing with the example, from the mobile part configure the delivery through the Send method passing the variable of the new message.



To receive messages, the MessageReceived event is used to then display them to the user.

Chatbot

CONVERSATIONAL USER INTERFACES



GeneXus

CHATBOT GENERATOR

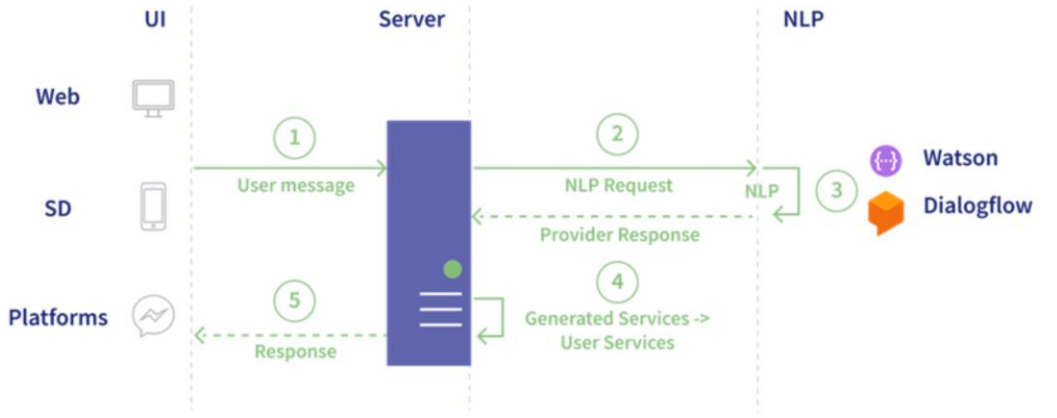
Cross-platform

Integrated to the solution

Hybrid UI

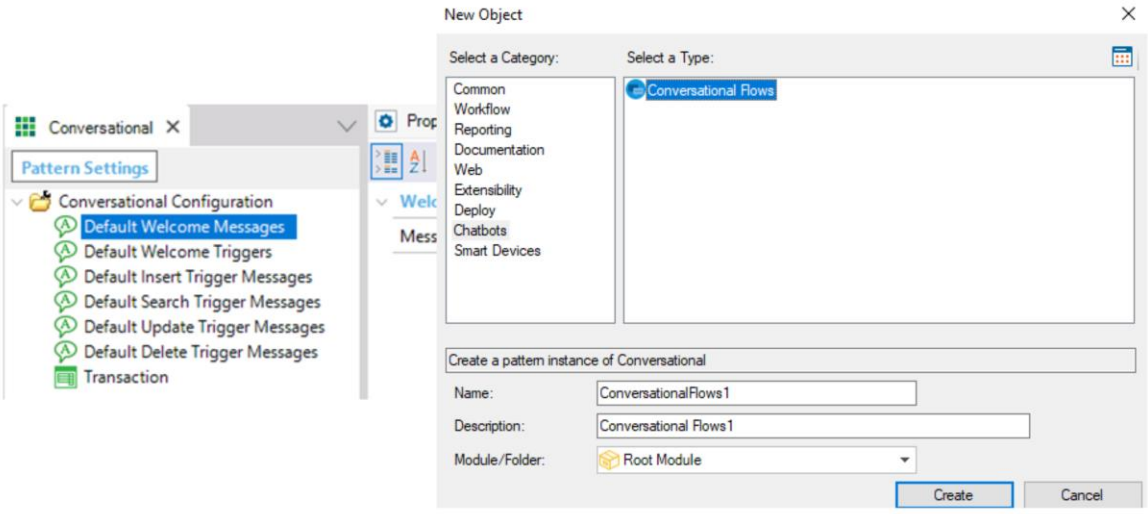
GeneXus

CHATBOT GENERATOR ARCHITECTURE



GeneXus

Chatbot Generator

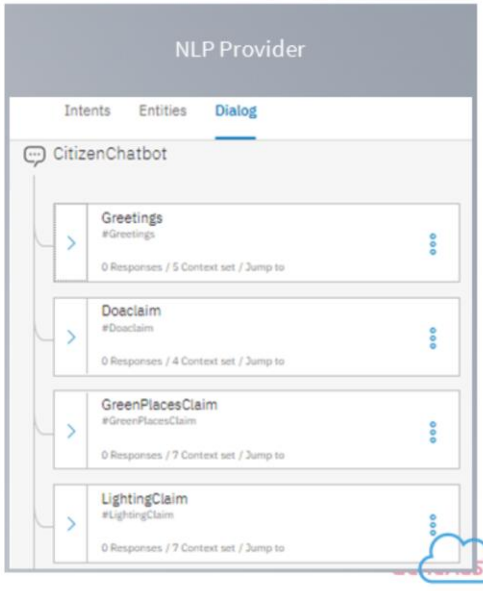


CHATBOT GENERATOR OVERVIEW

Conversational Flows

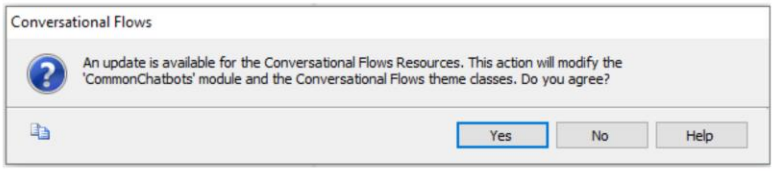
Chatbot Generator

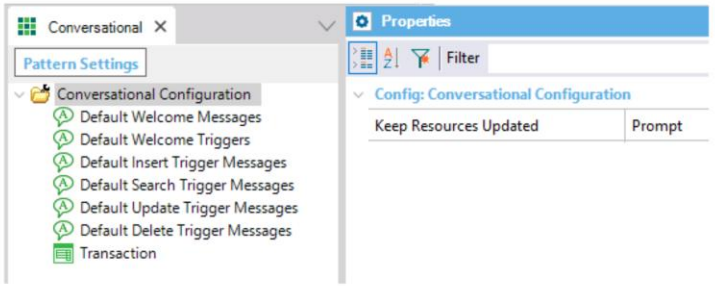
GeneXus objects generated

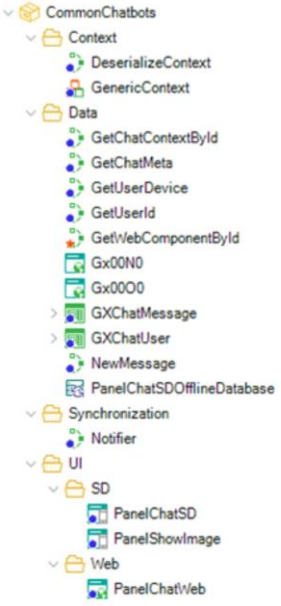


Resources update

CommonChatbots module update







Chatbot Generator Resources

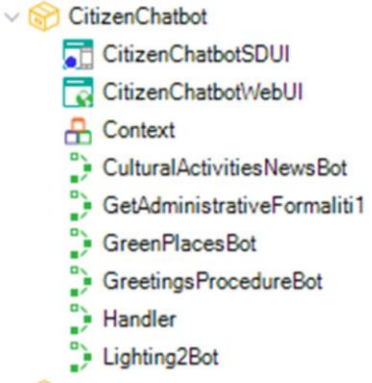
GXChatMessage, GXChatUser

Carmine, CarmineSD

PanelChatWeb, PanelChatSD

Generated Objects

Module containing generated objects



CitizenChatbotWebUI:

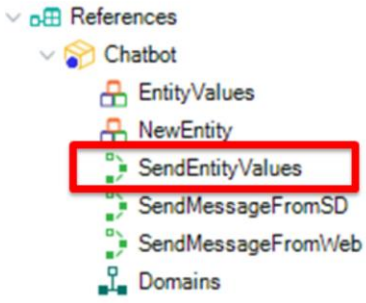
```
Event Start
    CommonChatbots.PanelChatWeb(Chatbot.Conversational.Watson, !"Citizen")
Endevent
```

https://wiki.genexus.com/commwiki/servlet/wiki?37102,Chatbot+generator,#+2+Generated+objects+of+the+Pattern

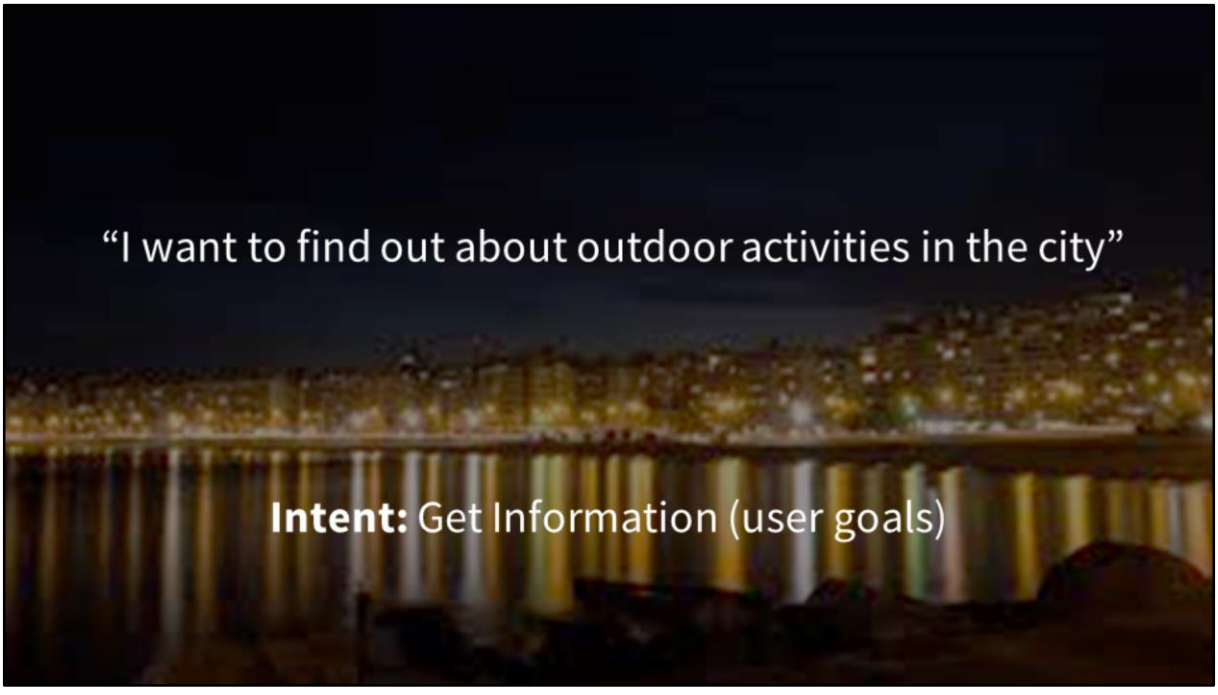
Chatbot External module

Communication with the NLP Providers

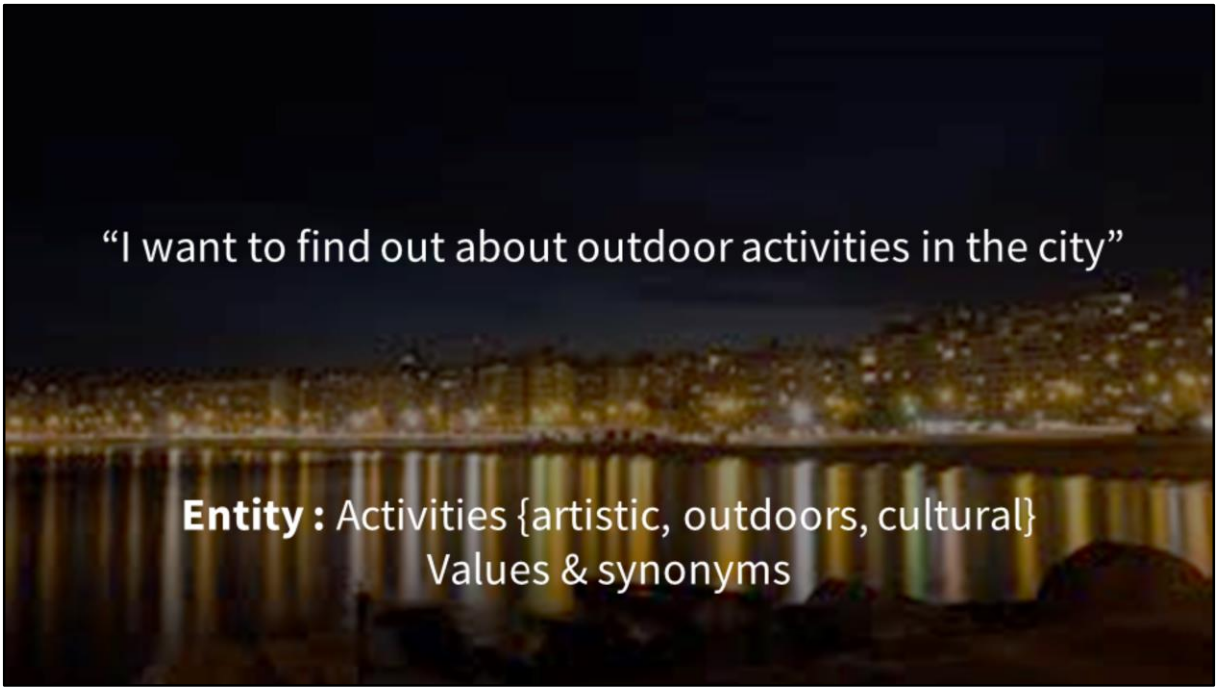
```
Chatbot.SendEntityValues(&Provider,&SDTEntityValues,
!"UserIdentification",&InstanceName,&messages)
```

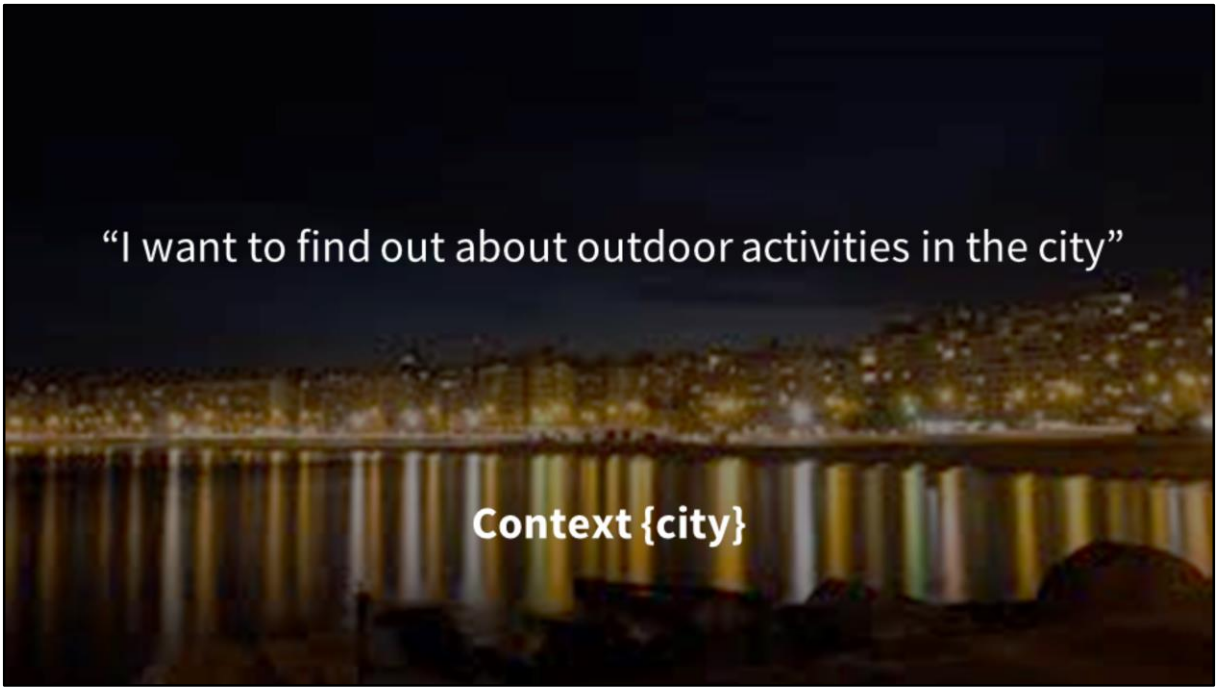


How to create a Chatbot using GeneXus



https://www.youtube.com/watch?v=wyWgsF9eYc8


“I want to find out about outdoor activities in the city”
Entity : Activities {artistic, outdoors, cultural}
Values & synonyms


“I want to find out about outdoor activities in the city”
Context {city}

Citizen service chatbot

Do a traffic claim

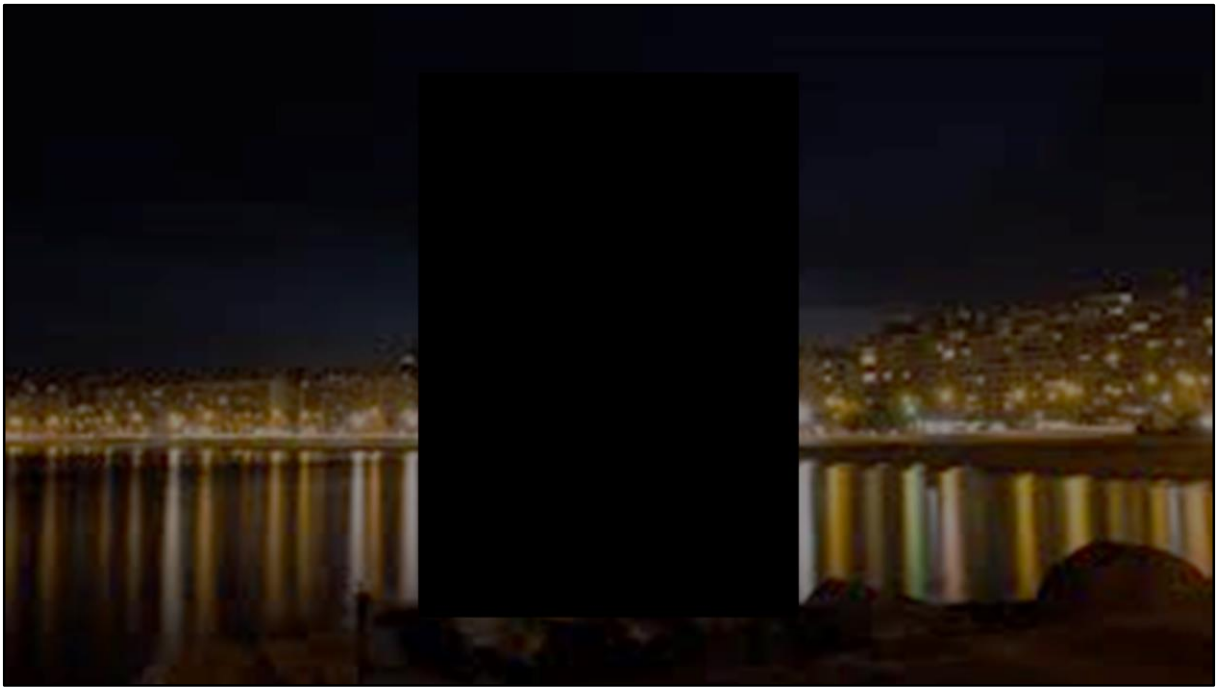
Find out about activities

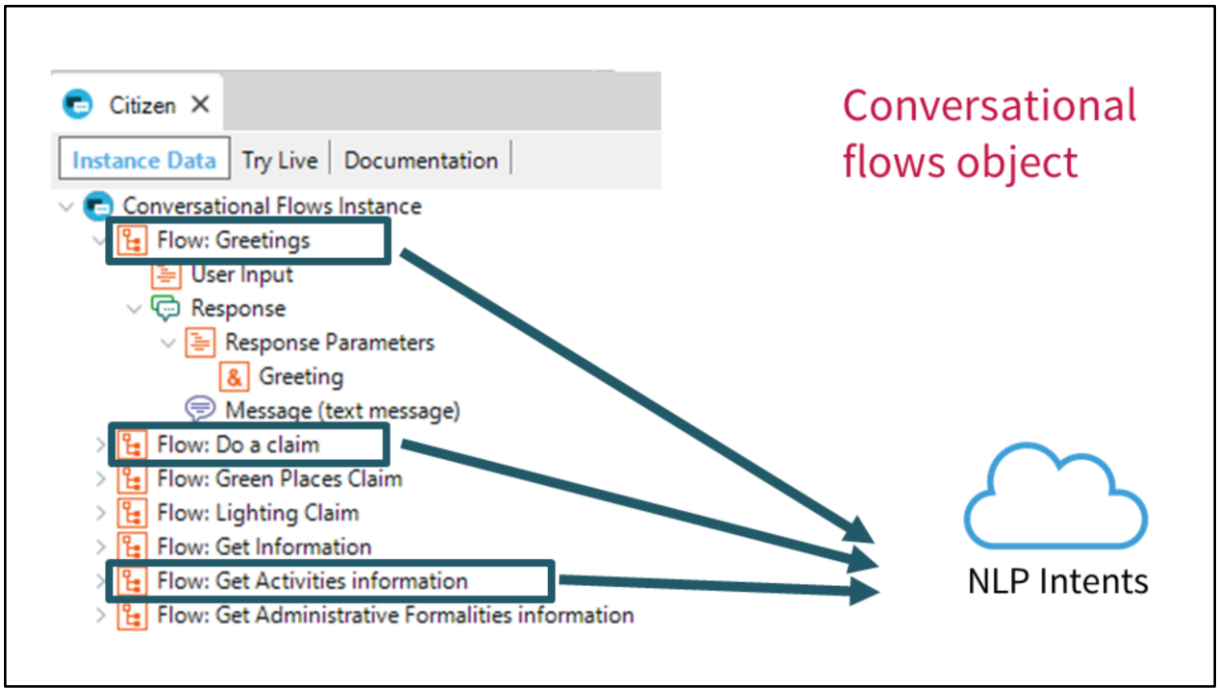
Get debt refinancing info

Setup an appointment

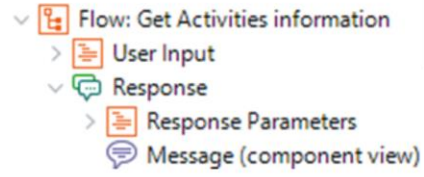
NLP response

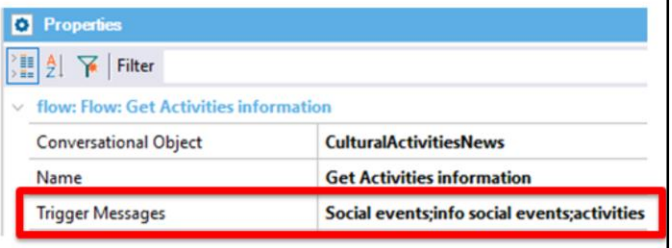

```
{
 "intents": [
  {
   "intent": "Doaclaim",
   "confidence": 1.0
  }
 ],
 "entities": [],
 "input": {
  "text": "do a claim"
 },
```


Citizen
Instance Data
Try Live
Documentation
Conversational Flows Instance
Flow: Greetings
User Input
Response
Response Parameters
Greeting
Message (text message)
Flow: Do a claim
Flow: Green Places Claim
Flow: Lighting Claim
Flow: Get Information
Flow: Get Activities information
Flow: Get Administrative Formalities information
Conversational flows object
NLP Intents

Trigger messages





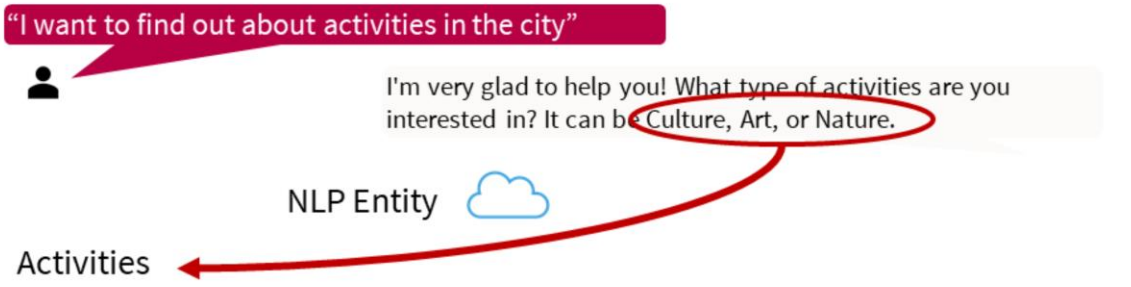
NLP Messages

Know about activities

Find out about leisure

Social events information

User input



NLP Entity

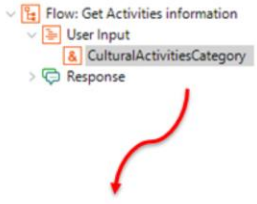
Activities

{artistic (art, music, theatre),
outdoors (nature, fresh air),
culture (cultural, museum)}

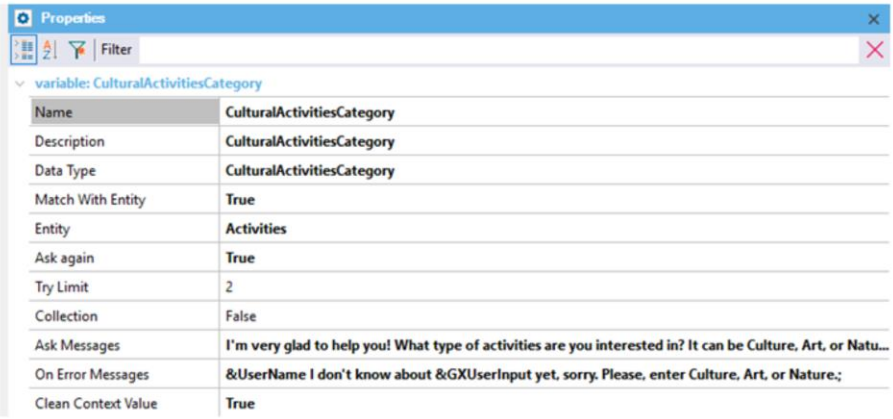
SendEntityValues(in:&Provider,
in:&EntityValues, in:&Entity,
in:&ChatbotInstance,
out:&Messages);

User input

"Outdoors activities! Thanks!"

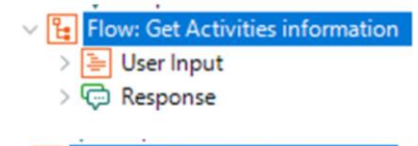


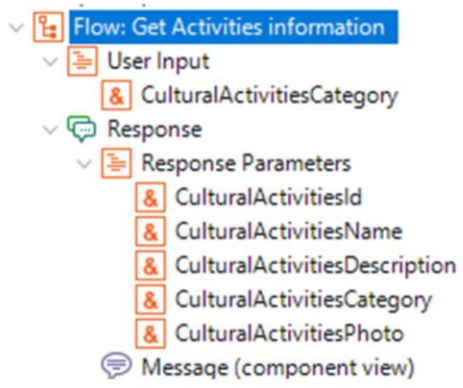
"outdoors"

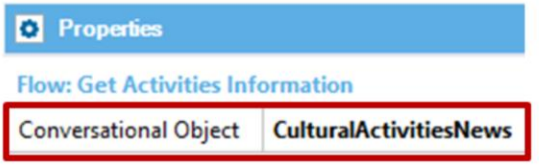


Name	CulturalActivitiesCategory
Description	CulturalActivitiesCategory
Data Type	CulturalActivitiesCategory
Match With Entity	True
Entity	Activities
Ask again	True
Try Limit	2
Collection	False
Ask Messages	I'm very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Natu...
On Error Messages	&UserName I don't know about &GXUserInput yet, sorry. Please, enter Culture, Art, or Nature.;
Clean Context Value	True

Response

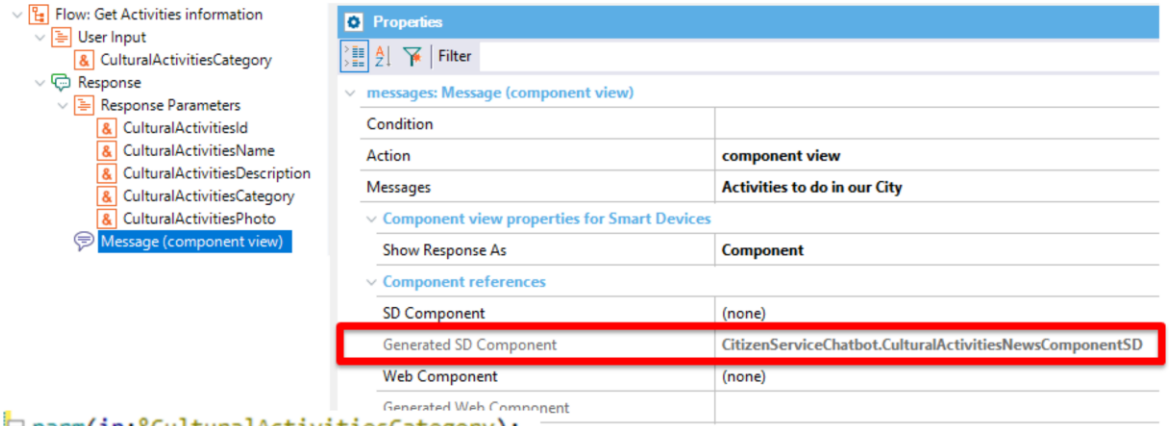






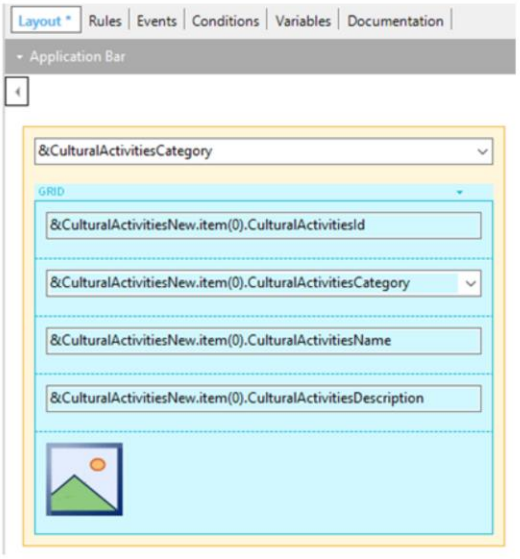

```
parm(in:&CulturalActivitiesCategory);

CulturalActivitiesNew
where CulturalActivitiesCategory = &CulturalActivitiesCategory
{
    CulturalActivitiesId = CulturalActivitiesId
    CulturalActivitiesName = CulturalActivitiesName
    CulturalActivitiesDescription = CulturalActivitiesDescription
    CulturalActivitiesCategory = CulturalActivitiesCategory
    CulturalActivitiesPhoto = CulturalActivitiesPhoto
}
```

Response




```
parm(in:&CulturalActivitiesCategory);

Event Start
    &CulturalActivitiesNew = CulturalActivitiesNews(&CulturalActivitiesCategory)
Endevent
```

Response

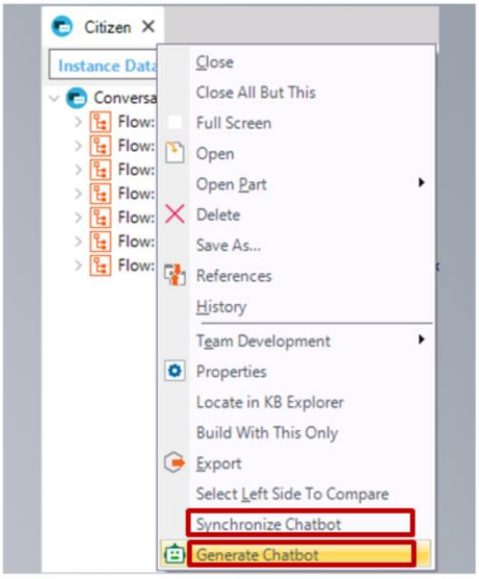

Layout *
Rules
Events
Conditions
Variables
Documentation
Application Bar
&CulturalActivitiesCategory
GRID
&CulturalActivitiesNew.item(0).CulturalActivitiesId
&CulturalActivitiesNew.item(0).CulturalActivitiesCategory
&CulturalActivitiesNew.item(0).CulturalActivitiesName
&CulturalActivitiesNew.item(0).CulturalActivitiesDescription


Android Emulator - GeneXus-API26-X86:5554
6:50
ConversationalFlowsChatbot
BACK
Japanese Garden
Behind the emblematic Municipal Museum of Fine Arts Juan Manuel Blanes is the distinguished Japanese Garden of Montevideo, a symbol of friendship between the Japanese people and the Uruguayan people. The garden is a work of art of nearly two thousand square meters that represents the beauty of nature and the spiritual values of Japanese culture.
Type to talk...

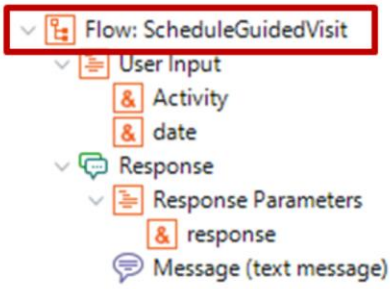
Impacting model changes

Synchronize Chatbot

Generate Chatbot



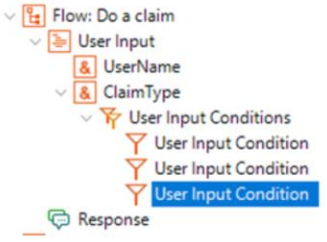
Another example: scheduling

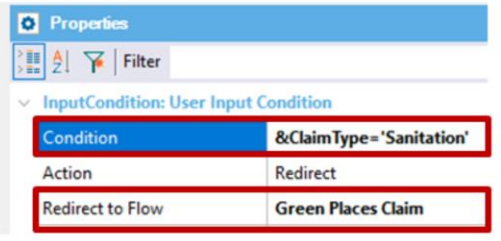


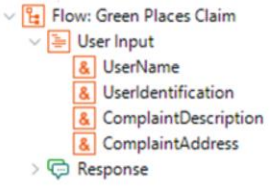
Context: Activity

System Entity: date

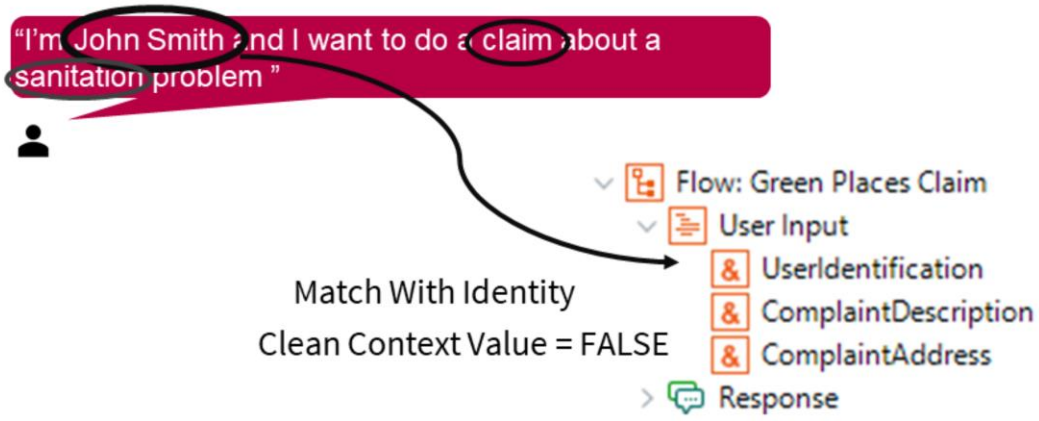
Redirect to another flow







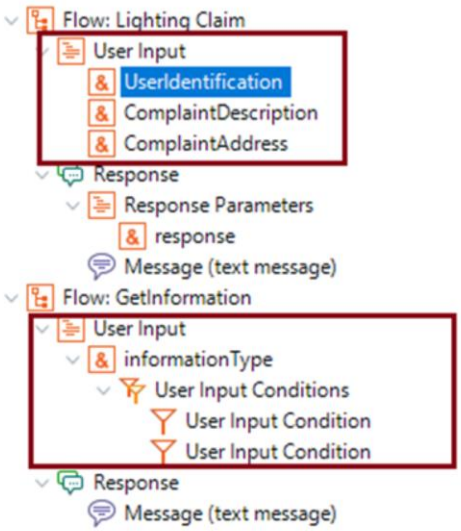
Entity inference from the query



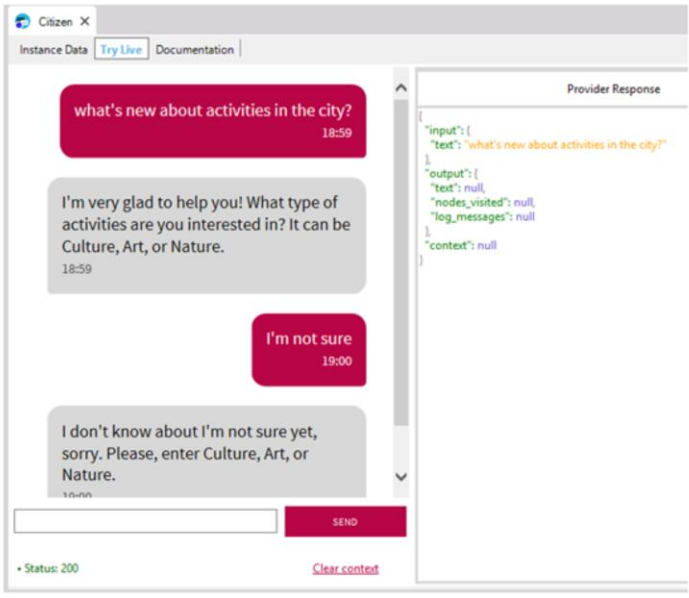
Context

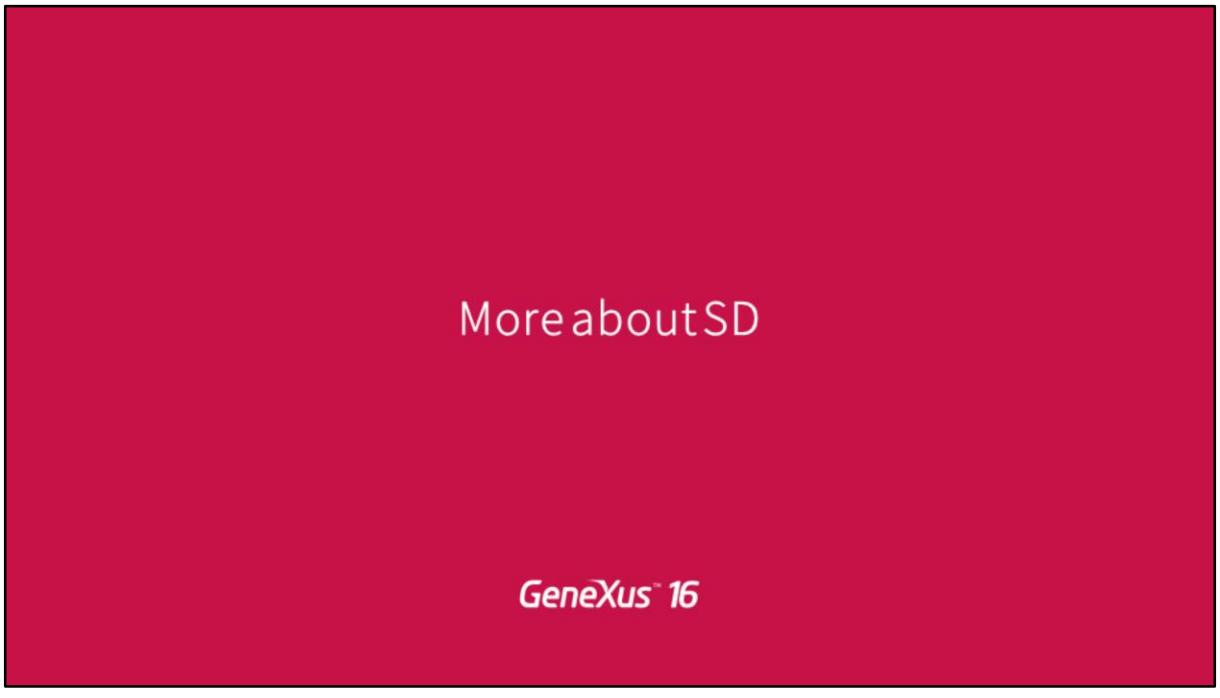
- &GXUserInput
- User Inputs

Clean Context Value = TRUE/FALSE

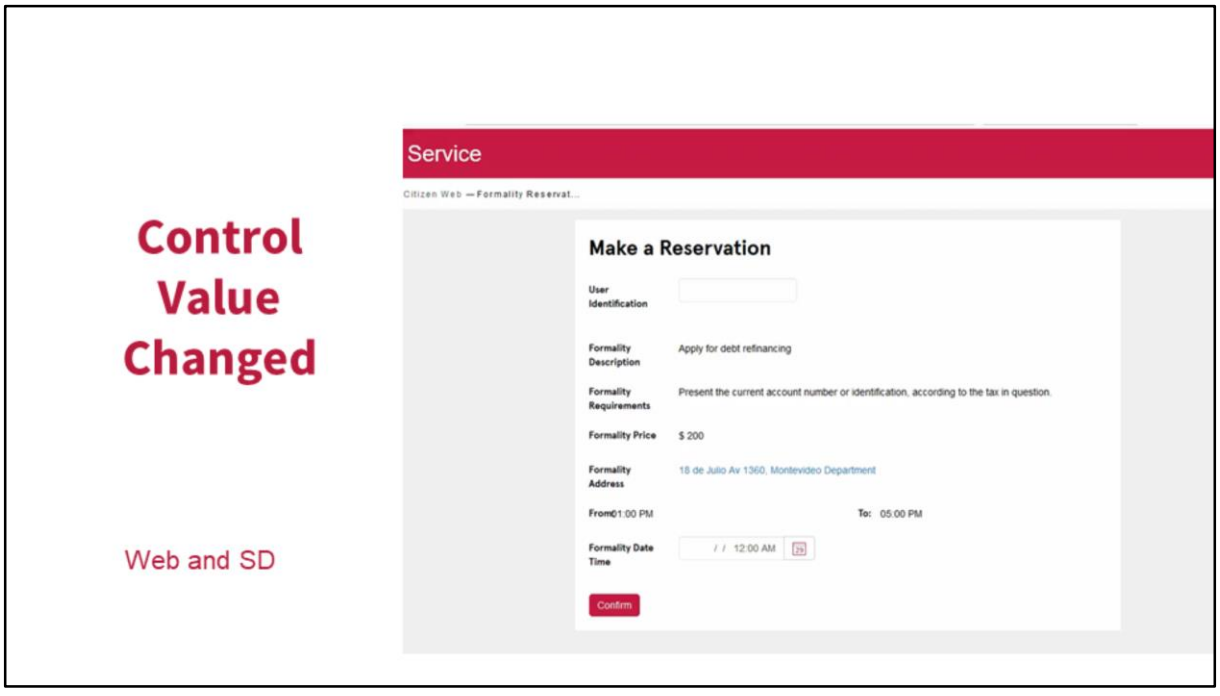


Try live


Citizen
Instance Data
Try Live
Documentation
what's new about activities in the city?
18:59
I'm very glad to help you! What type of activities are you interested in? It can be Culture, Art, or Nature.
18:59
I'm not sure
19:00
I don't know about I'm not sure yet, sorry. Please, enter Culture, Art, or Nature.
SEND
• Status: 200
Clear context
Provider Response
{
"input": {
"text": "what's new about activities in the city?"
},
"output": {
"text": null,
"nodes_visited": null,
"log_messages": null
},
"context": null
}



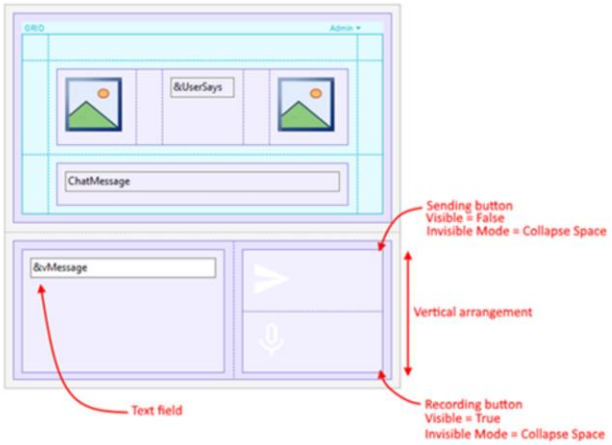
Let’s see some new developments in Smart Devices.



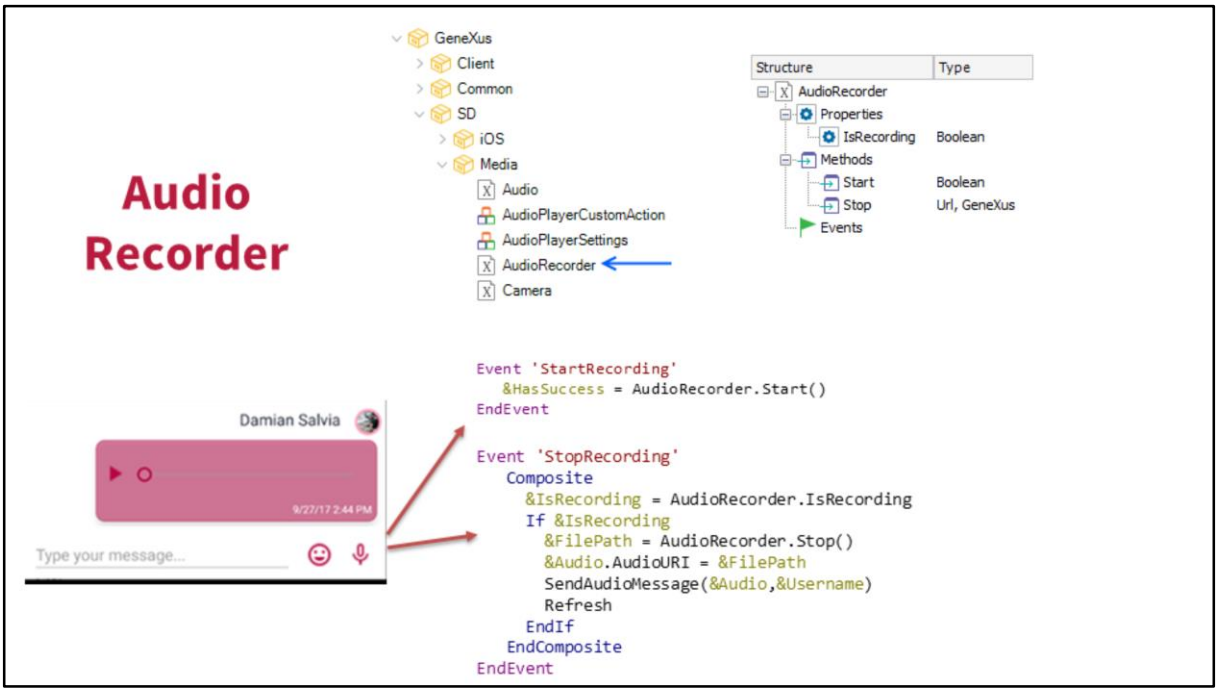
Control Value Changed. It’s an event that allows executing a certain code when the user finishes entering a value.
It is available for Web and SD.

ControlValue Changing

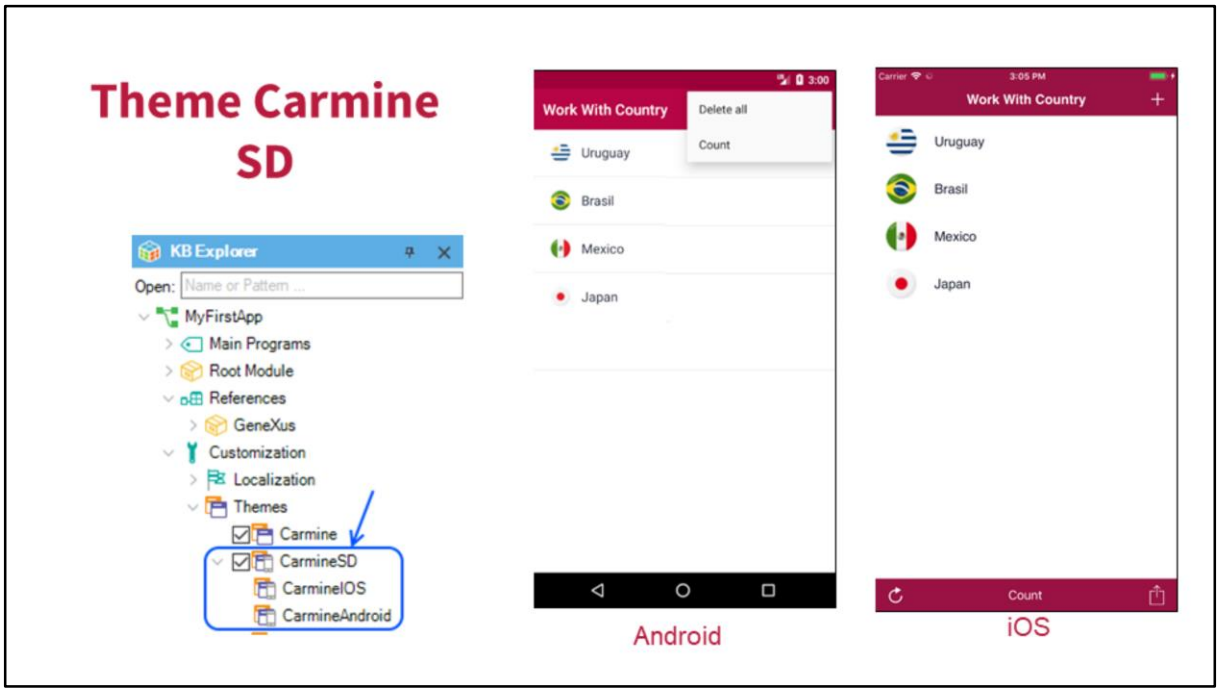

```
Event &vMessage.ControlValueChanging(&vMessageNew)
    Composite
        &MessageLen = &vMessageNew.Length()
        SendButton.Visible = &MessageLen > 0
        RecButton.Visible  = &MessageLen = 0
    EndComposite
Endevent
```



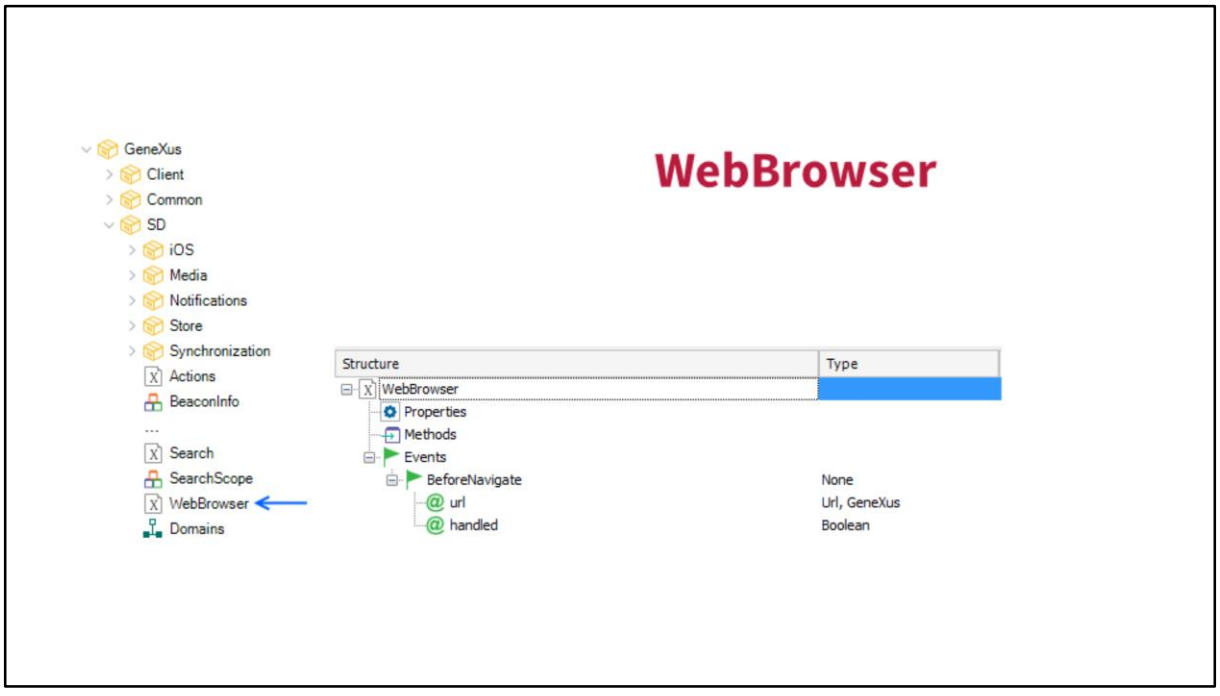
The ControlValue Changing event allows validating if the user is changing the value of an editable field and execute a code.
For example, when the user hasn't started entering text, the button to record an audio is enabled, and when he begins to write the text the button to send it is displayed.



Audio recorder is an external Object that allows recording an audio track using the Start and Stop methods.



Since upgrade 6 of GeneXus 15 the Carmine theme is available. It allows obtaining a modern and sophisticated look & feel.



Web Browser is an External Object that allows managing the events triggered from a URL embedded in a Smart Device application.

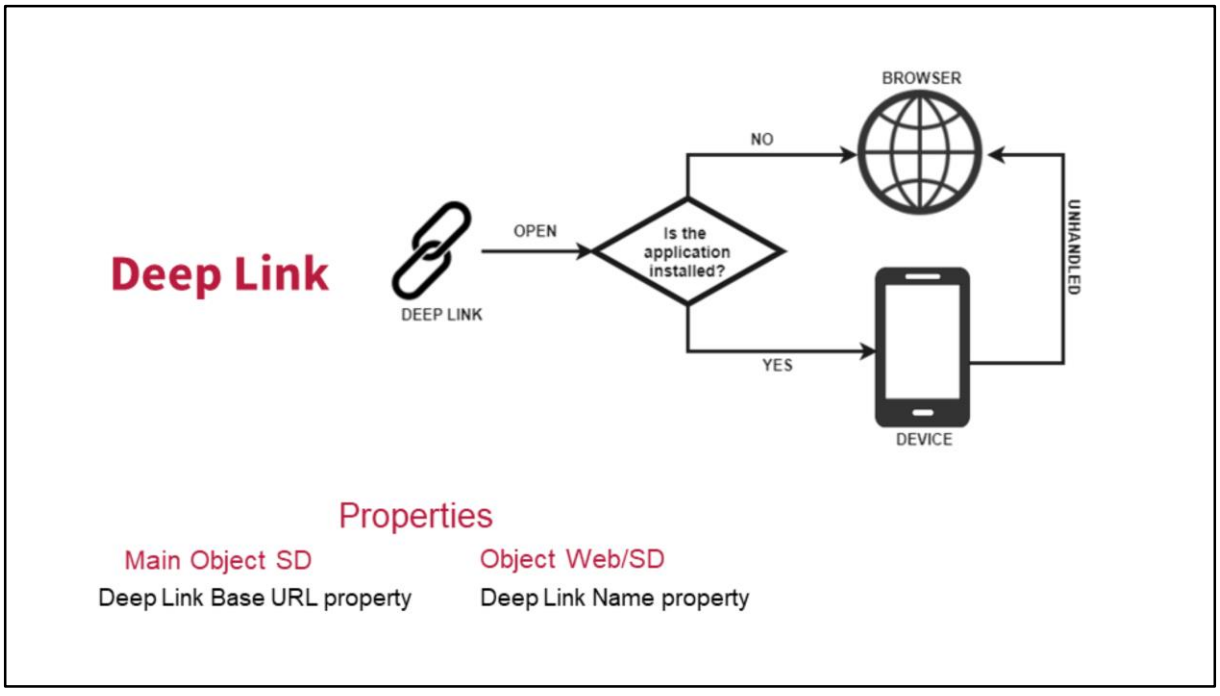
Sample: WebBrowser

Before Navigate to URL or Web app Embedded

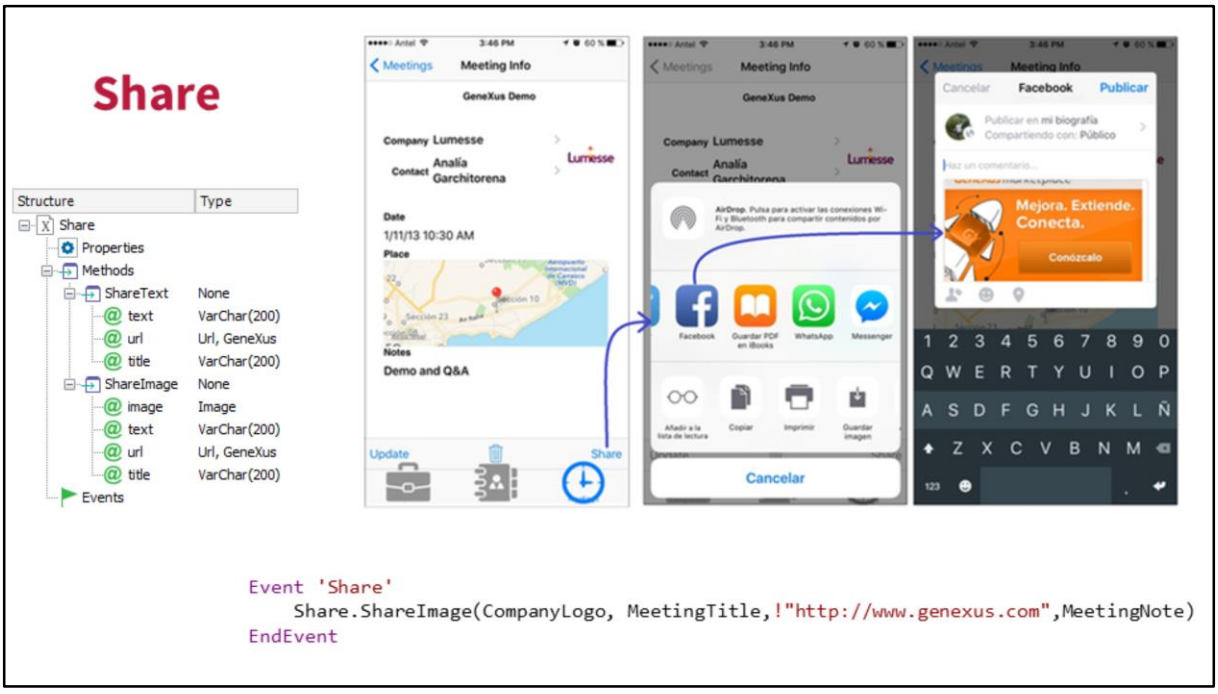
```
Event GeneXus.SD.WebBrowser.BeforeNavigate(&Url, &Handled)
   composite
     if &Url = //something
        //<your_code>
       &Handled = true
       return
     endif
   endcomposite
EndEvent
```

Web app Embedded in SD app

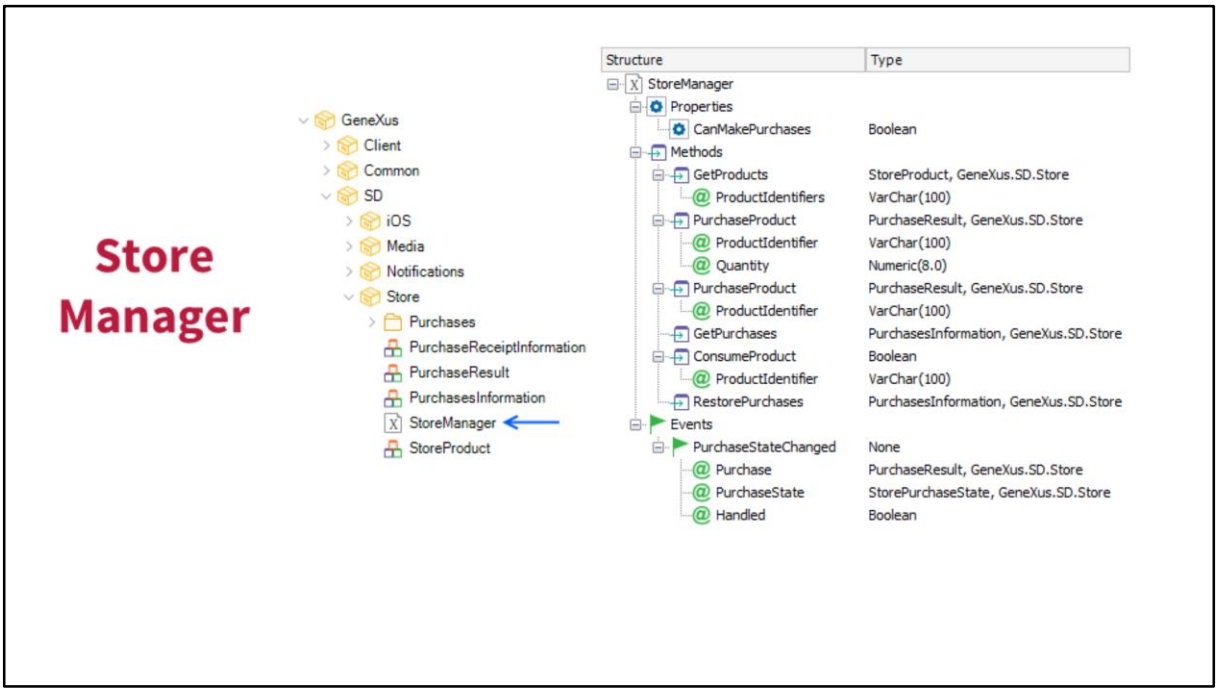
To manage navigation, use the event Before Navigate, which is triggered before browsing a URL or embedded web application.
In this way you can control if the user is accessing an external URL to your application.



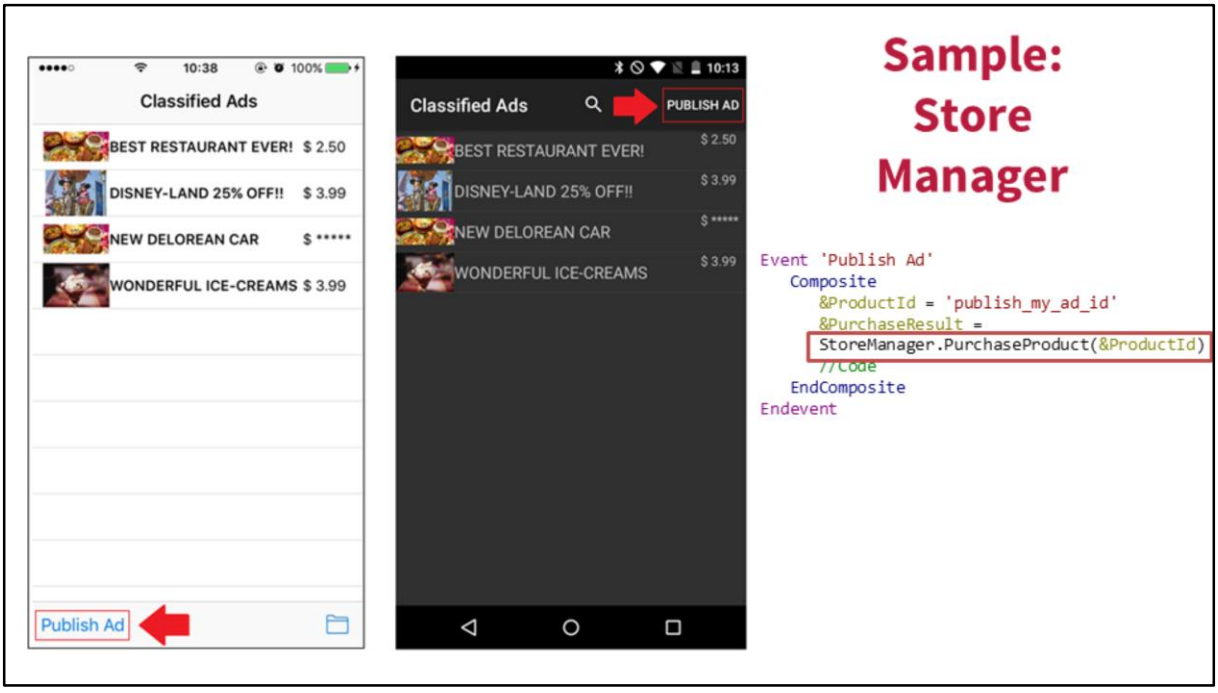
Deep Link allows choosing how to open a URL. If the SD app is installed it will open it there; otherwise, it will open the corresponding object in the Web browser.



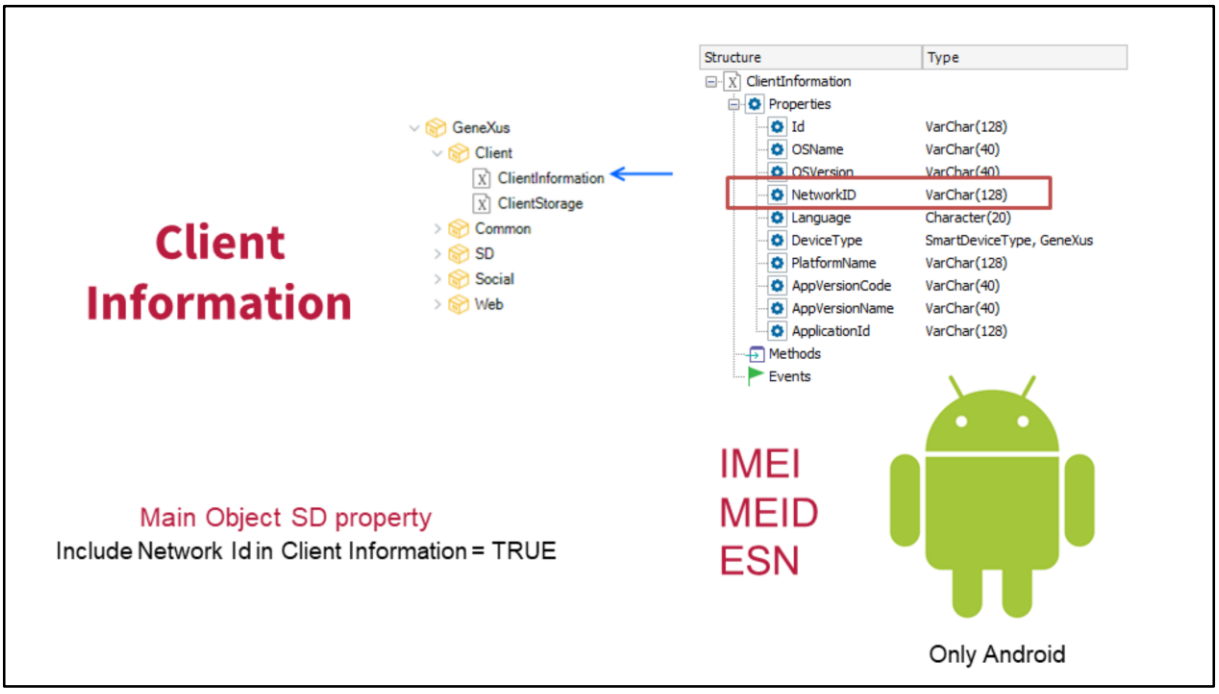
Share is an external Object that allows sharing Texts or Images with an external application that makes it possible to do so.



Store Manager is an External Object for managing purchase applications.



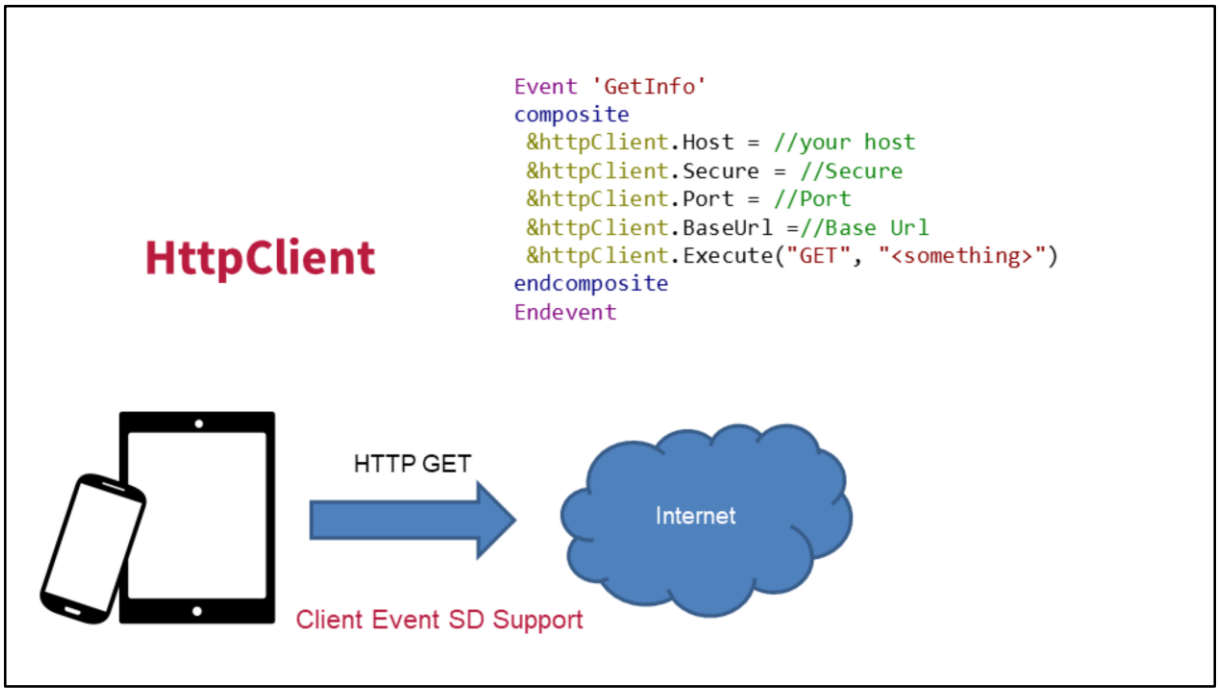
In this example we can see that when you tap the Publish Ad button, the option to complete the purchase is displayed.



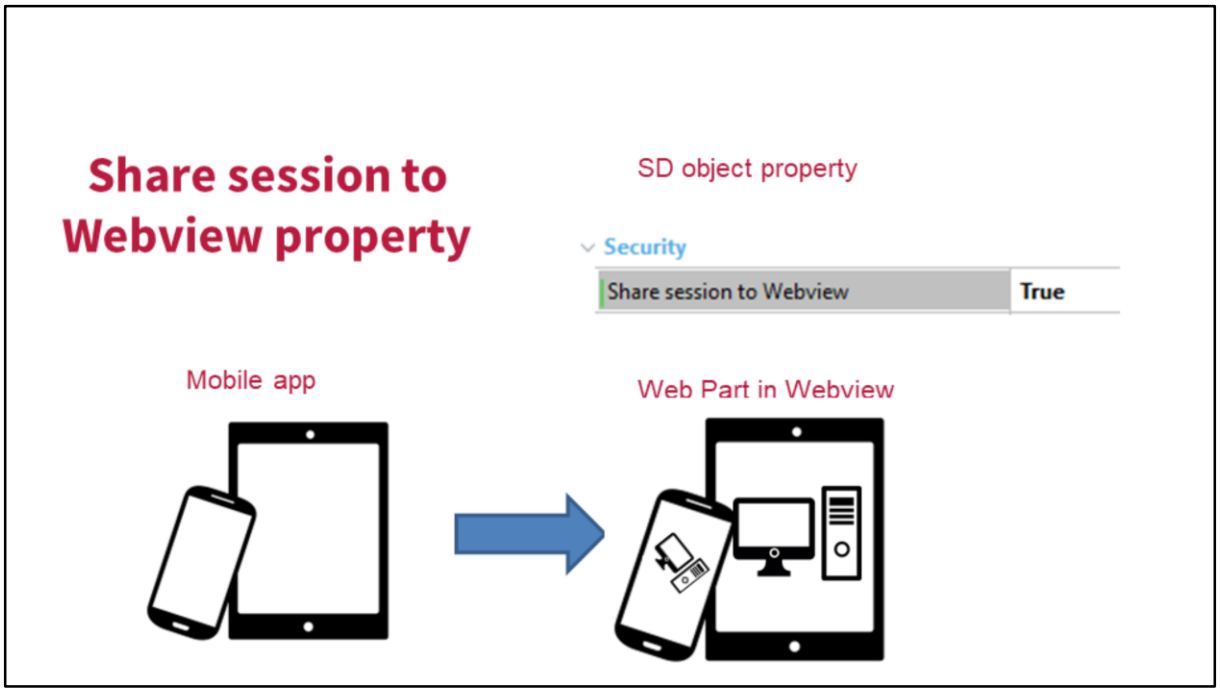
Client Information is an External Object for retrieving the device information. One of its main features is the Network ID property, which together with the Include Network ID property of the main object allows retrieving the IMEI of the device.

SDGauge

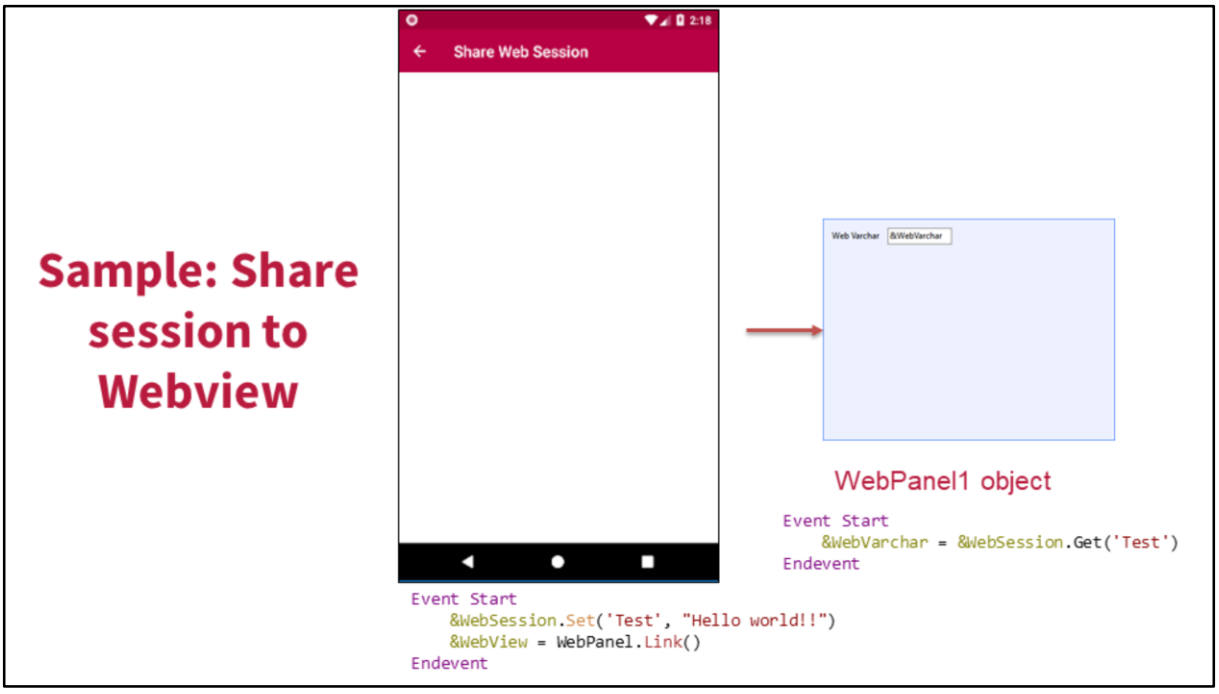
SDGauge is a control that allows displaying information as ranges, in linear or circular mode. This version of the control offers animation features to improve the user's experience.



As from version 15, the HttpClient data type can be used in Smart Device applications, and from upgrade 12, client-side event management is available.

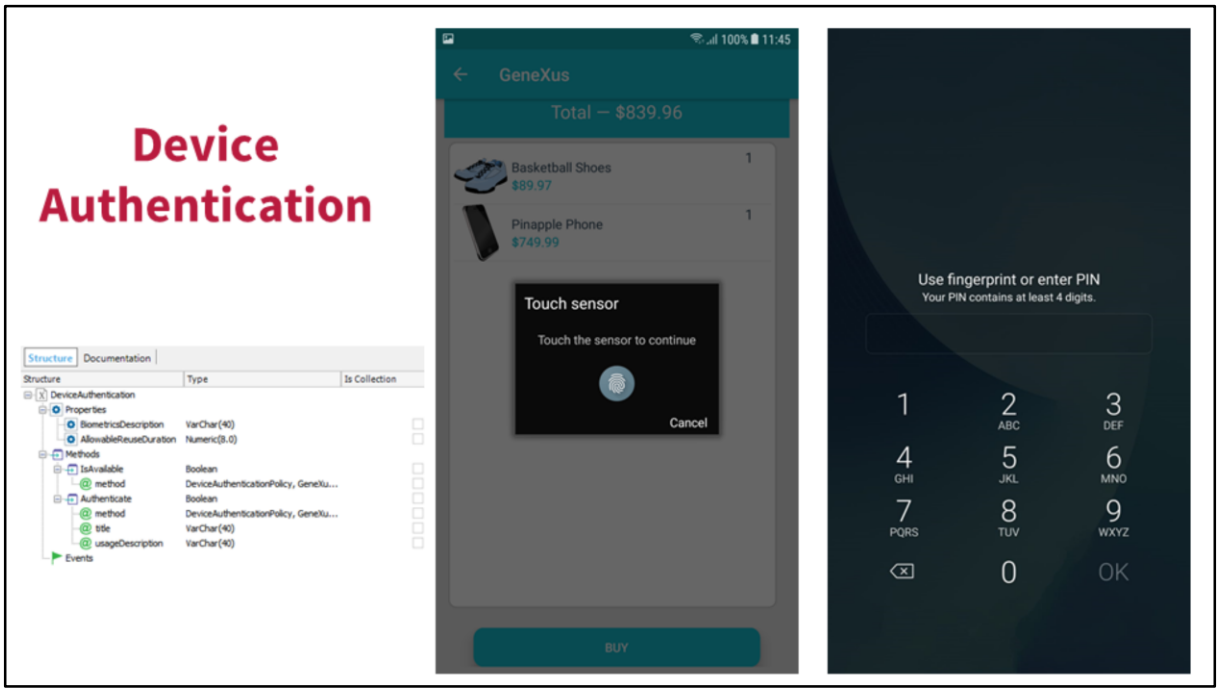


Share session to Webview property allows sharing a session between the mobile app and its web part.



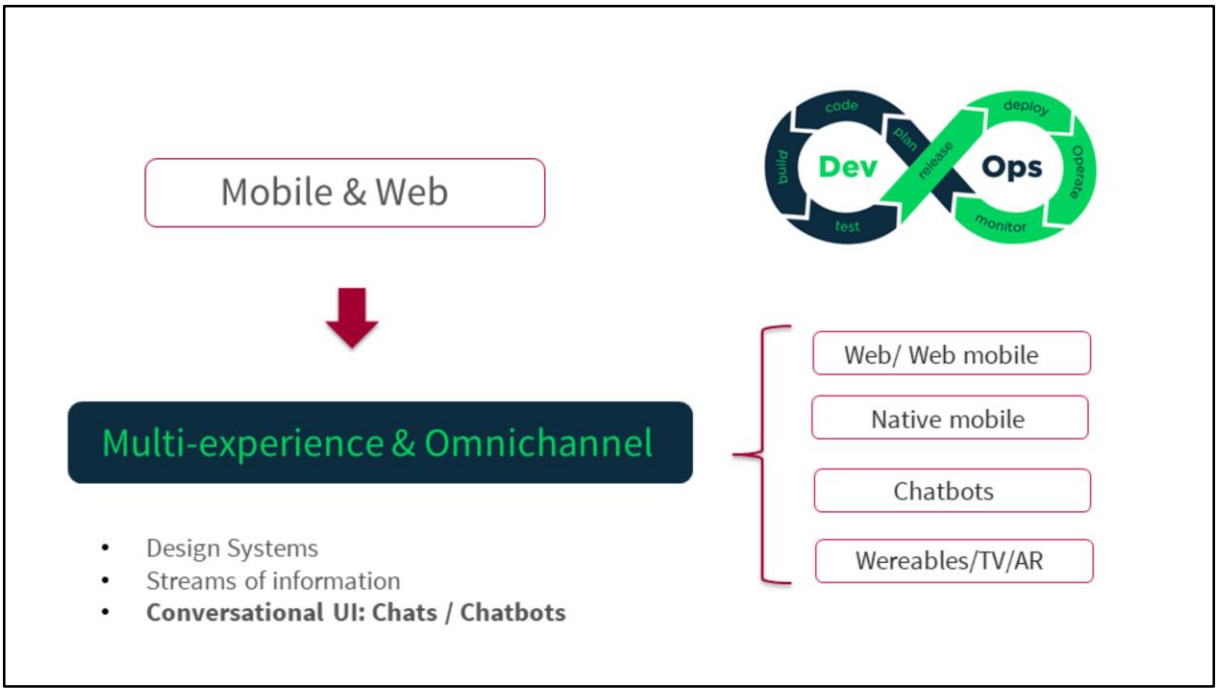
For example, from the SD object the web session is set.

In a web panel we obtain that web session value and open the web panel in a web view.

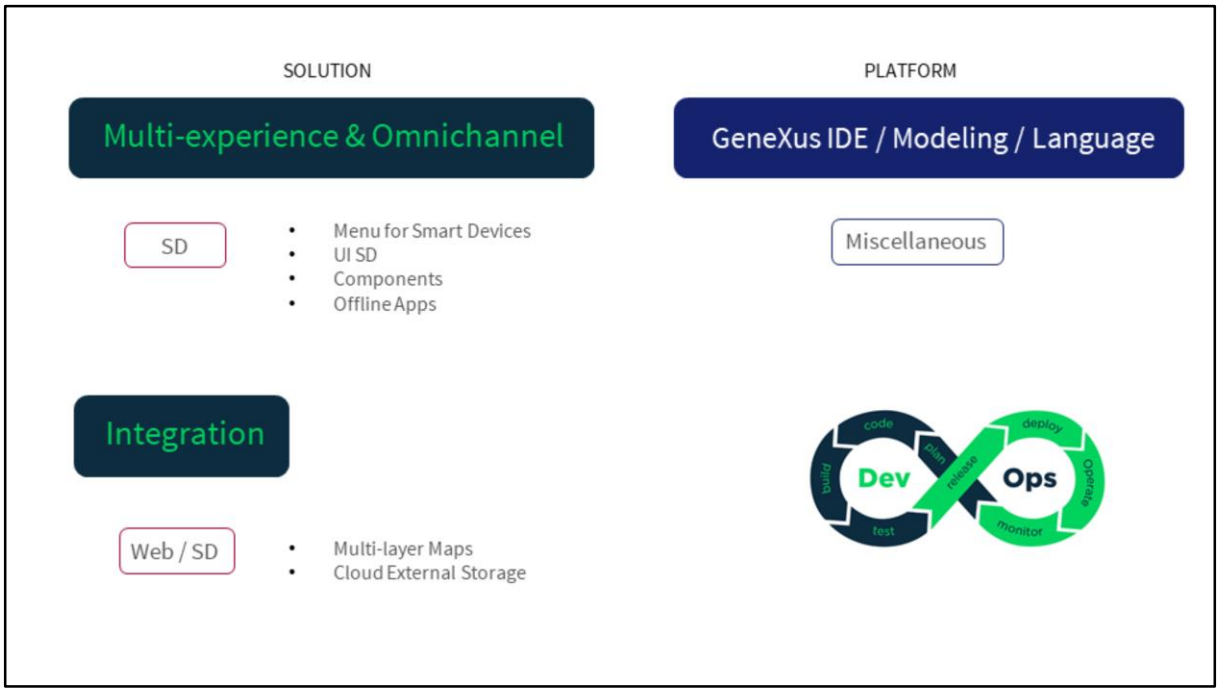


Device Authentication allows the developer to access the local authentication methods of the device.

Each platform manages its own.



We've seen the features available to develop a simple chat and a chatbot, and some aspects about SD.



Next, we'll see more features that have been incorporated in the different upgrades of GeneXus 15 up to version 16.

We'll study the new possibilities offered by maps, and some improvements regarding the external storage of multimedia files in the clouds supported.

Also, we'll see some aspects related to the GeneXus platform.


GeneXus™
The power of doing