First Steps with GeneXus™

Create your first Application without knowing how to code.

GeneXus™ by Globant

El Table of Contents

INTRODUCTION

GeneXus is a Low-Code Development Suite that enables the quick generation of software applications in multiple languages and platforms. GeneXus offers several advantages: It's easy to learn, highly productive, cross-platform and future-proof, in a way that both protects your digital assets and simplifies new technology adoption.

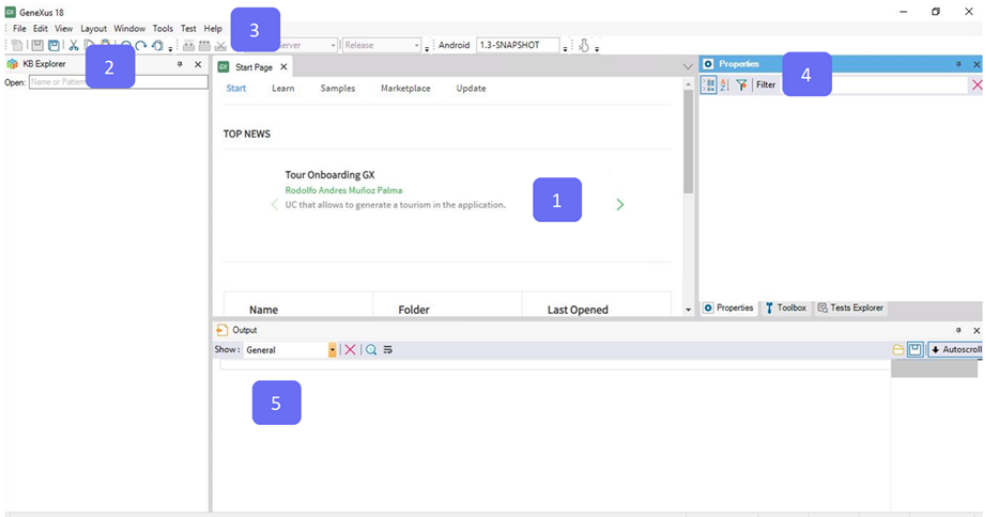
For example, GeneXus generates applications for the Web and/or Mobile devices (from a watch, cell phone, tablet or TV), for the target platform selected by the developer (certain language, database, environment, platform, with web responsive design, etc.).

This document is a beginners' guide for developing applications with GeneXus.

GETTING STARTED WITH GENEXUS

Upon opening GeneXus, you will see an interface known as IDE (Integrated Development Environment), similar to the one below. This interface is easy to use and may be parameterized by each developer.

It consists of different windows:



1. **Main Window (Start Page)**: It dynamically displays technical information about the tool and the GeneXus community (news, as well as solutions posted by other developers). It also shows recently used projects available to be opened and it offers the chance to create a new project.

2. **KB Explorer:** Displays objects and settings of the currently open project.

3. **Toolbar:** Displays an easy-to-use interface for commonly used functions in GeneXus.

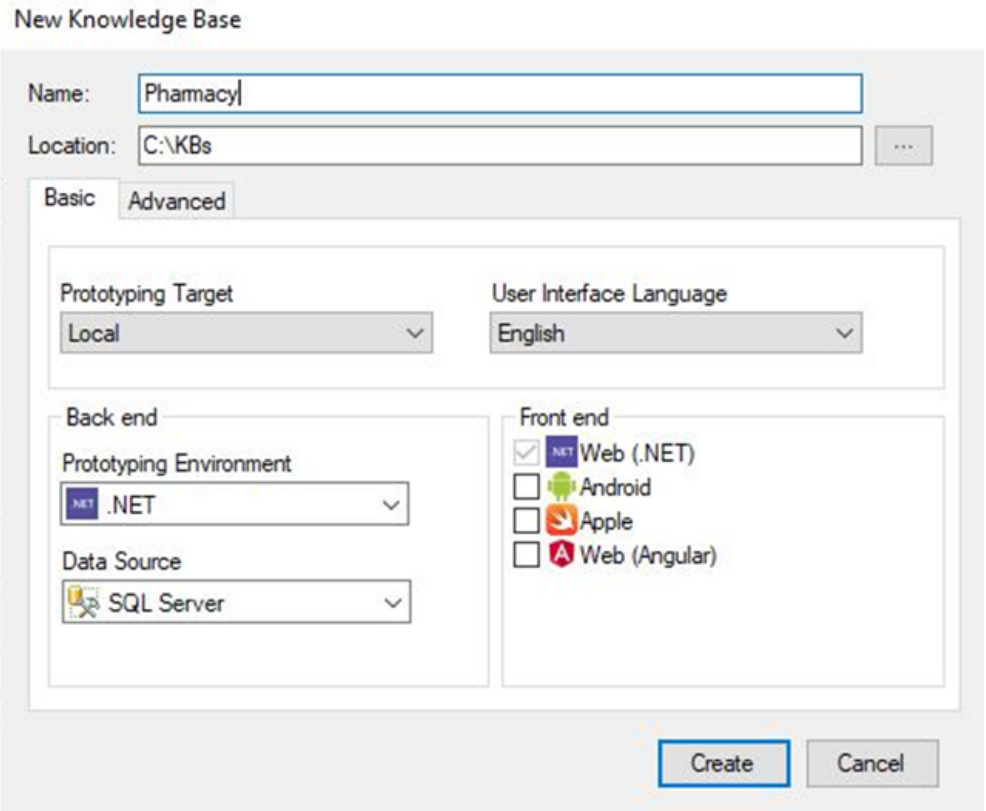
4. **Properties window:** Displays properties associated with the context in which the developer is positioned (like a selected object, attribute, variable, control, etc.).

5. **Output:** Displays the output of the actions performed.

CREATING A NEW GENEXUS APPLICATION

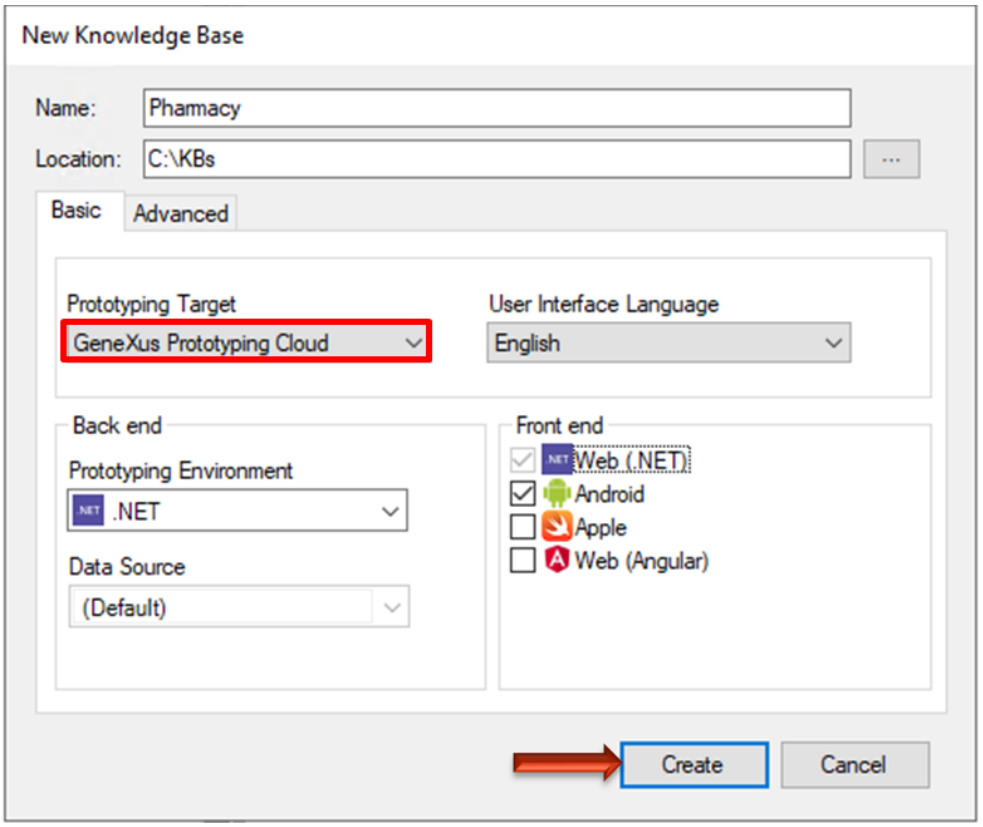
To start developing a new GeneXus application, you have to create a new **Knowledge Base** (a Knowledge Base is a GeneXus project).

By selecting **File > New > Knowledge Base** in the Toolbar, the following dialog box will be displayed:



The sample application that will be defined throughout this document is a real, but simplified application for a pharmacy. So, it makes sense to call the Knowledge Base “Pharmacy” (or “PharmacySystem”, among other options). Also, the location where you want to create the Knowledge Base must be entered.

The next step consists of selecting, if you want to, the programs to be generated by GeneXus in your local prototyping machine or in the GeneXus Prototyping Cloud. Select “GeneXus Prototyping Cloud” in the **Prototyping Target** combo box.



The **User Interface Language** combo box enables you to select the language in which you want GeneXus to generate automatic button captions, labels, messages for the users, etc. The default language is English.

The **Back end box** allows you to indicate:

- One of the programming languages available in the Prototyping Environment combo box. GeneXus will use the selected language to generate the back end application programs, as well as the necessary programs to create and maintain the database. By default, the selected language is .NET. Keep it.
- The DBMS over which the database will be created, accessed, and maintained. The Data Source combo box offers SQL Server by default. Keep it. Further ahead, you will have to enter the database details.

The **Front end box** allows you to indicate the programming languages you want GeneXus to generate the front end applications programs with. .NET is the language offered by default to generate front end applications for the Web. You can select others to generate Web and/or Native Mobile applications, too. In this case, .NET will be used.

By pressing the Create Button, GeneXus starts the Knowledge Base creation process.

DEFINING THE FIRST OBJECTS

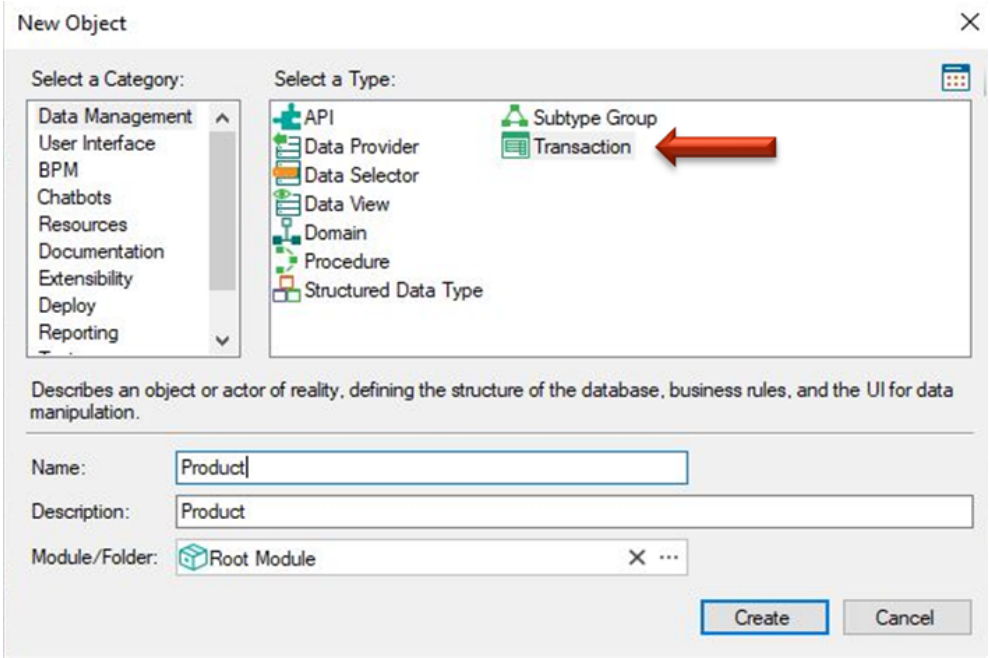
Once a new Knowledge Base is created, the next step is to describe the users' visions . In order to do so, it is necessary to identify real-life objects (we recommend paying attention to the nouns that users mention in their descriptions, such as: products, invoices, customers, etc.) and start defining them by using GeneXus **objects**.

GeneXus developers don't work on low-level tasks such as defining tables, normalizing, designing programs, programming, etc. Instead, their work is a higher-level activity that implies describing the users' reality. After that, GeneXus analyzes the defined objects and goes on to design the database and the application programs for the selected platform in a totally **automatic manner**.

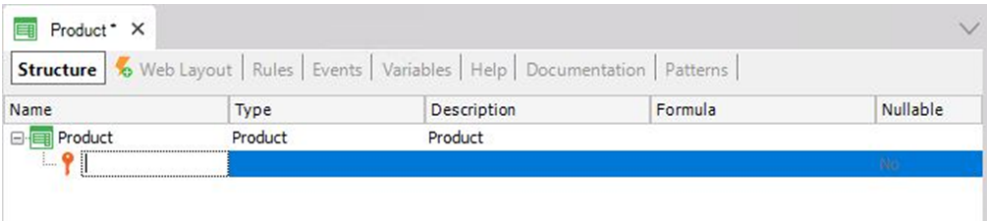
Consider the case where the pharmacy requesting the application asks to be able to record the products they have on sale.

To describe each identified real-life object, you have to create a GeneXus object of the Transaction type (not related to Database Transactions). So, let's see how to create a Transaction object to describe the Products.

By selecting **File > New > Object** in the Toolbar, the following dialog box will be displayed to allow you to select the type of object you want to create and enter a name for it. You have to select the Transaction type and you can call it: *Product*.



By clicking on the Create button, the *Product* Transaction is created and it keeps open ready for you to start defining its structure:



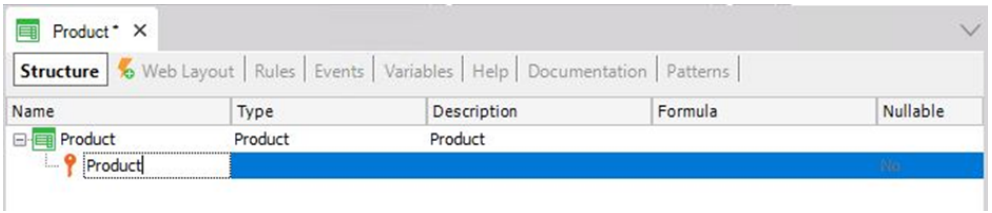
Each Transaction has some sections that will gradually be explained. Specifically, the Transaction structure enables the definition of the attributes or fields that describe a real-life object.

Suppose that at the pharmacy you were told that they need to keep a record of every product's code, name, sale price, stock and its type (medicine, cosmetic, etc.). Therefore, this data that must be recorded for each product matches the attributes that have to be created for this Transaction.

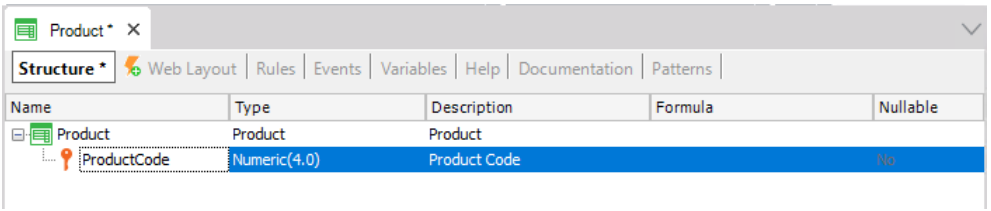
Note that in the image above, the first line in the Transaction structure is created ready to enter the first attribute. Also note that an icon key is associated with this line. The reason for this is that in every Transaction, an attribute – or set of attributes – must be set with an identifier or key role.

The concept of identifier or key attribute is aimed at uniquely identifying each product (or any object). In other words, the users will not be able to enter two products with the same identifier value. Clearly, the key attribute of the Product Transaction is the product code. So, let's see how to define it.

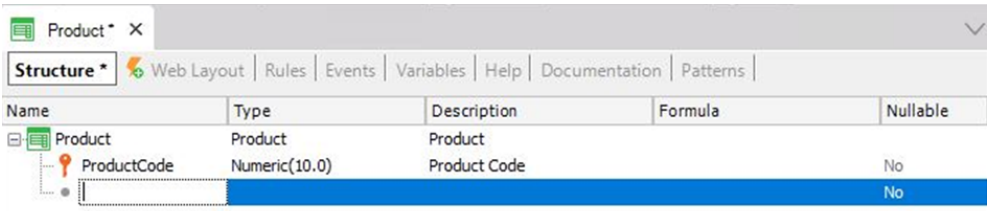
By pressing the dot key on the keyboard, GeneXus will automatically show the Transaction name as prefix in the attribute name:



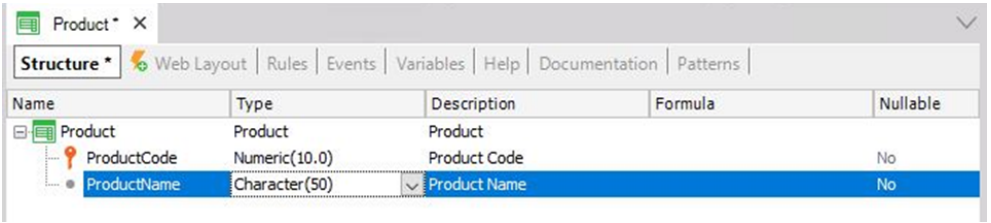
and you only have to type *Code* after the *Product* prefix:



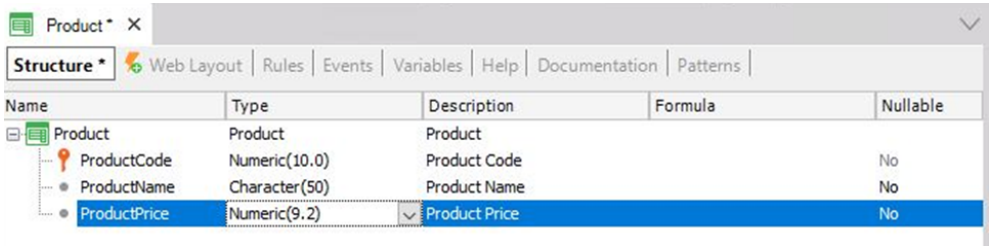
Then, you can choose the data type for this attribute. The default data type is: Numeric of 4 digits with no decimals. However the pharmacy requested that the product code always be a numeric value of up to 10 digits, so you have to change its length to 10:



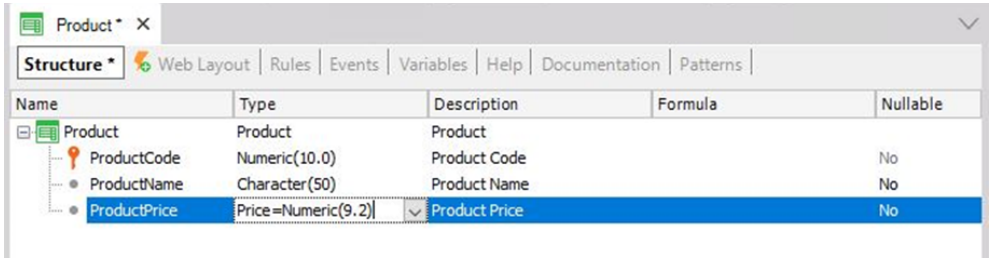
By pressing Enter, a new line is opened where you can start defining the second attribute. Again, you have to type the dot key on the keyboard and complete the attribute name with *Name*, that is, *ProductName* (of the Character type, and length of 50):



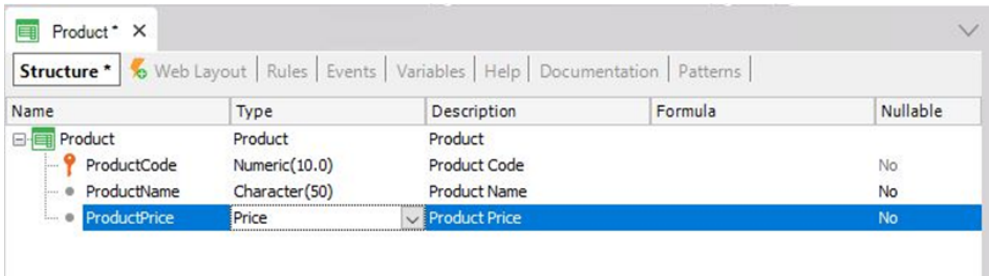
Now you must add the *ProductPrice* attribute (of the Numeric type, with 9 digits and 2 decimals):



As you will probably need to create more attributes to define prices or amounts (i.e. when the pharmacy buys or sells products), it might be a good idea to create a generic definition type for all prices. To do this, you just have to write: "Price=" in the Type column, before the type you have recently selected:



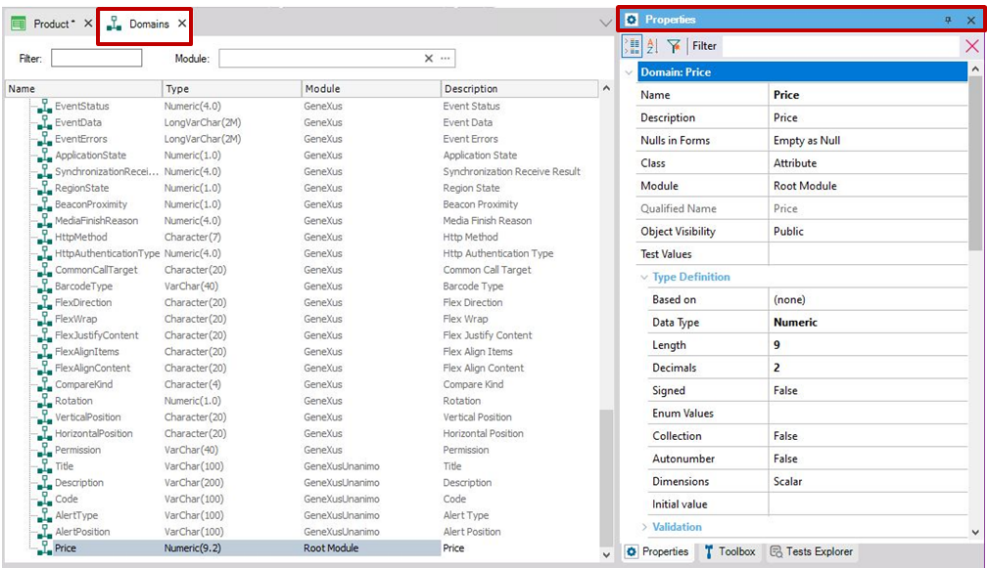
Press Enter, and you will see that the *ProductPrice* attribute has been set as *Price* type:



Your *Price* definition with Numeric type (9 digits with 2 decimals) is called **Domain**.

Domains aim at making generic definitions possible. **One of the advantages the domains provide is that, if later on you need prices to be Numeric of a different length, changing the domain definition will be enough to update all the attributes based on that domain in a single step.**

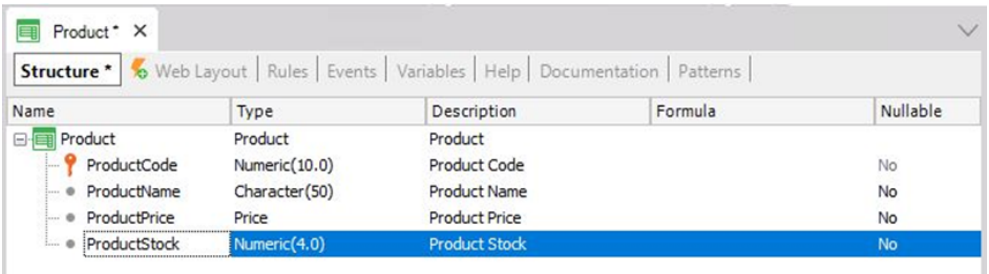
You can see the created domains in the Knowledge Base, by selecting **View > Domains** in the Toolbar:



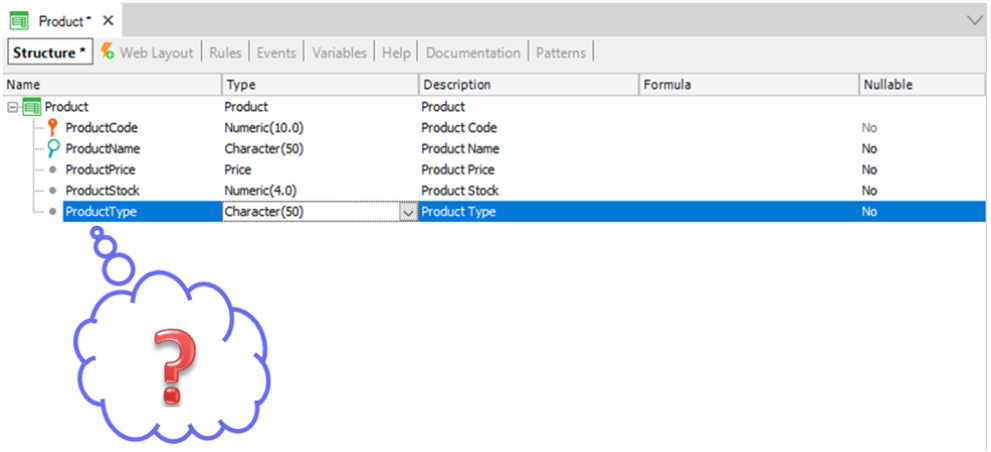
As shown in the image, GeneXus automatically creates some domains. When you click on a given domain, the Properties Window is refreshed with its properties. Note that, in addition to setting the data type for a domain, you can also define other interesting properties for it.

The Properties window will be refreshed every time you select another attribute, domain, object, etc., with the corresponding properties available for configuration in each case. If it is not visible, you can press F4 to open it.

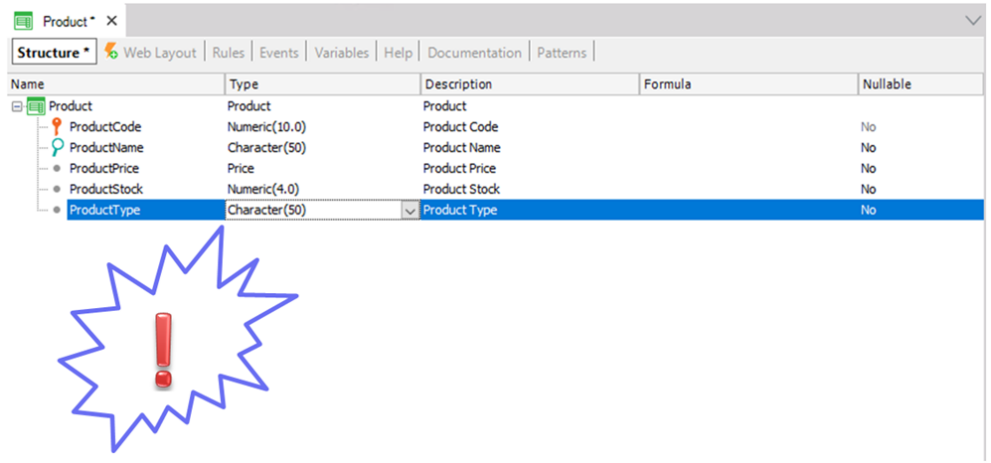
Let's go back to the *Product* Transaction, where the next attribute you have to define is *ProductStock* of the Numeric type, and length 4:



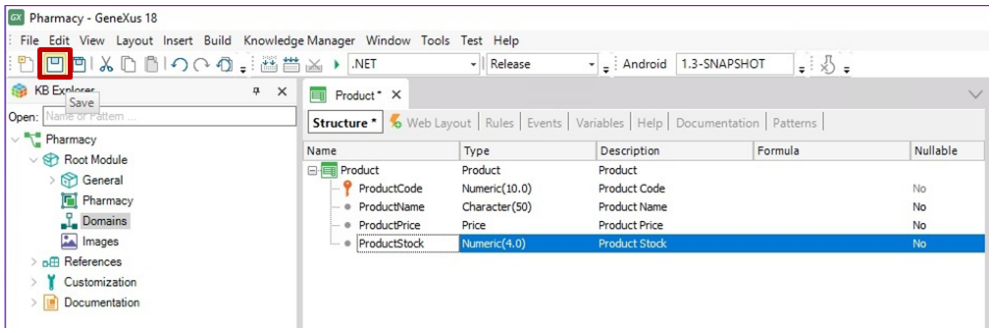
Now it is necessary to record the product type. You could create an attribute called *ProductType* as Character(50)…



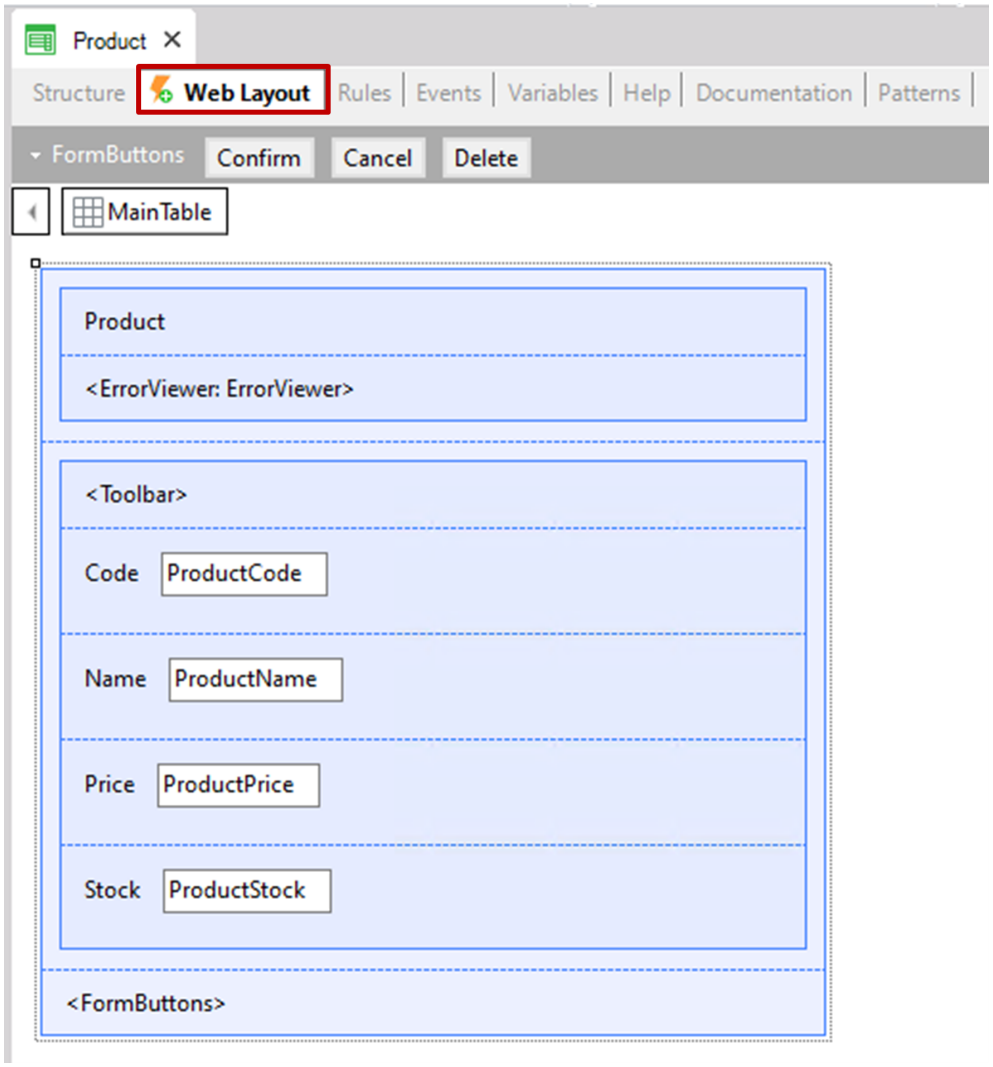
But what happens if the users want to enter two products of the same type? They would have to enter the same type name twice, being careful to type it exactly the same! Later on, they might need to search for all the products of a certain type, and to get them, the name must have been typed exactly the same.



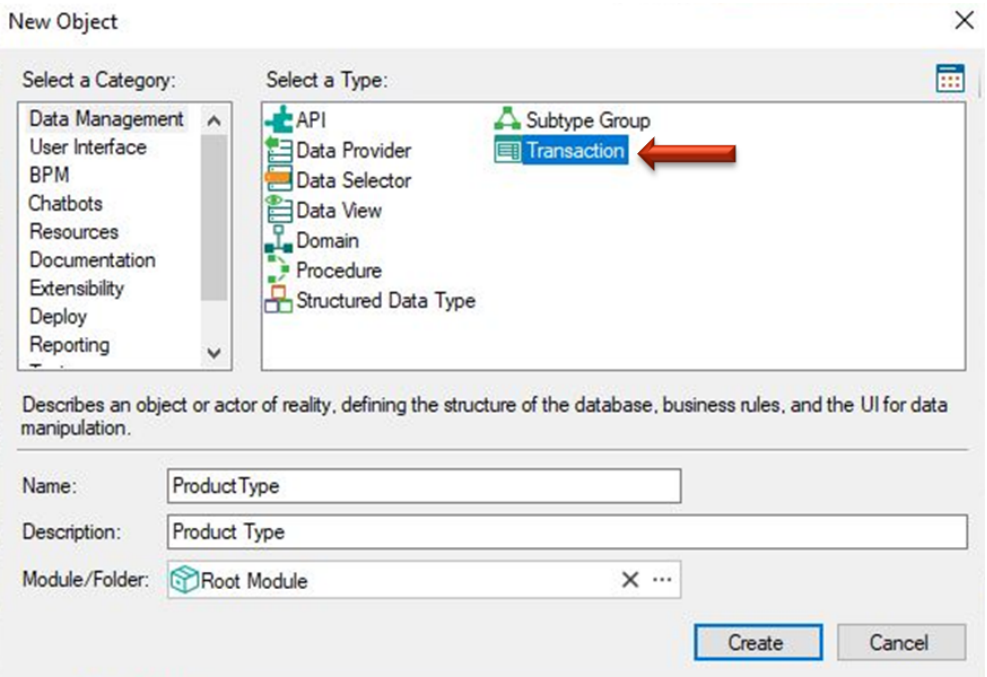
It seems more reasonable to enter the type only once, in a single location, and then, for each product, to reference the corresponding product type. So, let's delete the *ProductType* attribute from the structure and save the *Product* Transaction as shown:

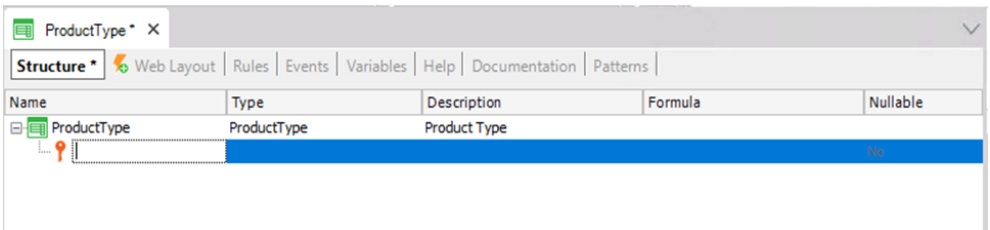


GeneXus has automatically designed a Web Layout according to the defined structure. This form will enable users to add, update and delete products in runtime:

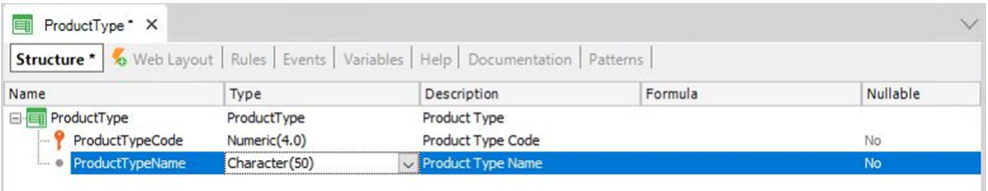


Now let's create another Transaction to record the product types, and after that, let's assign a product type to each product. To do so, select again **File > New > Object** and complete the following:



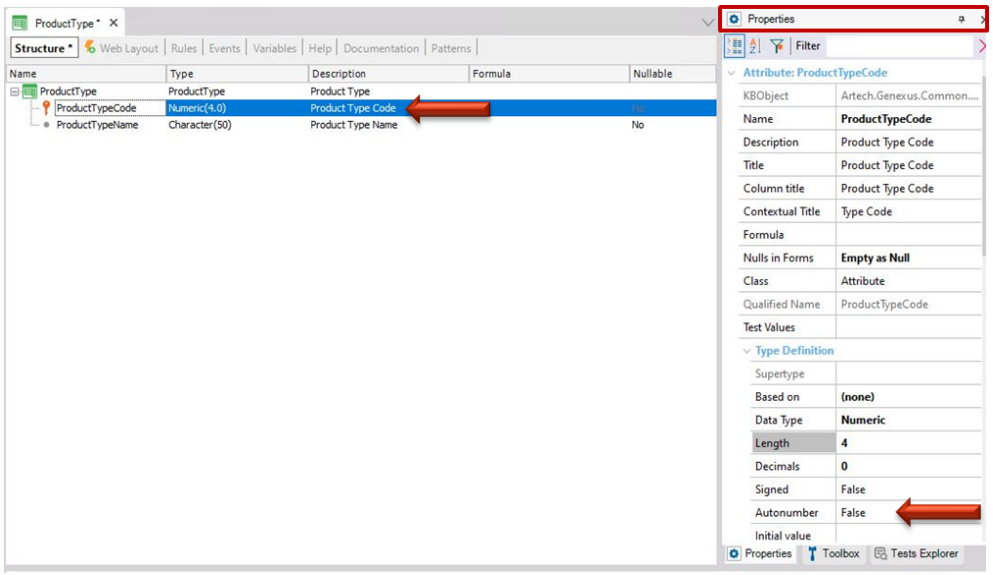


Let's store the code and name for each product type:



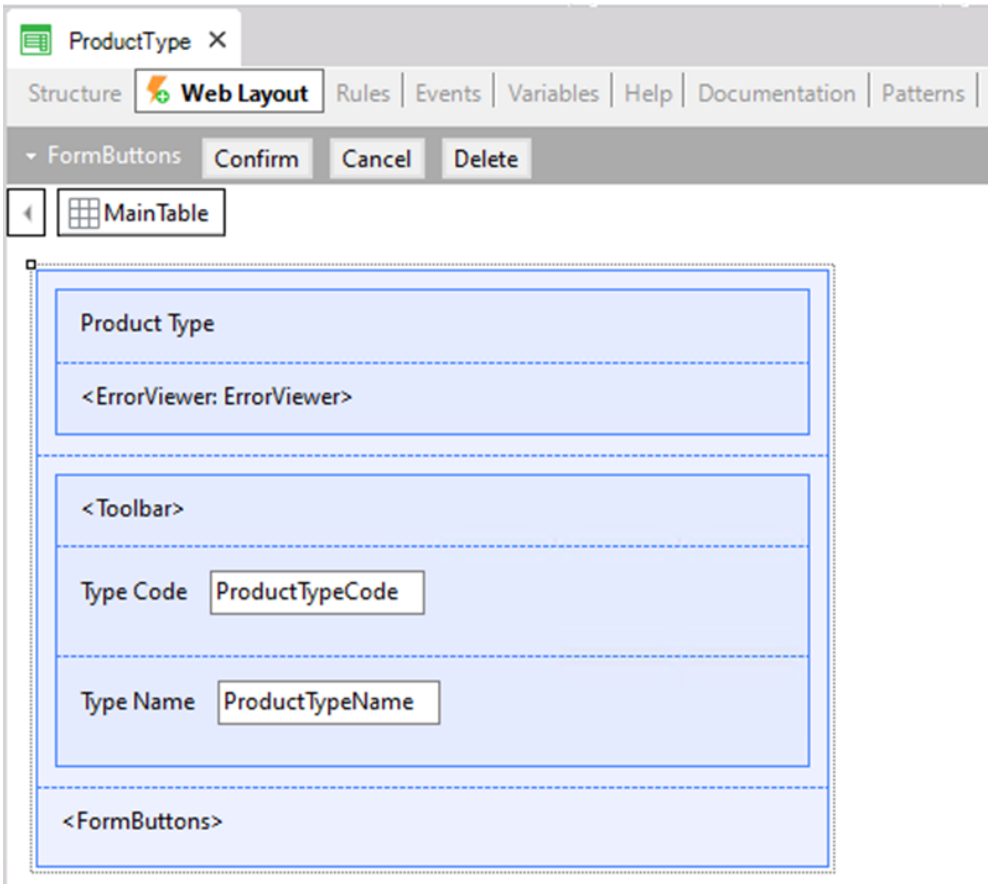
Remember the recommendation of typing the dot key on the keyboard when defining attributes, **so that GeneXus will automatically write the Transaction name as prefix, and all you will need to do is to complete the end of the attribute names.**
Naming attributes with the Transaction name as prefix not only makes defining attributes easier and faster, but is also a GeneXus community convention **for an easier comprehension when reading an attribute name wherever it may be, as to know to which object it is describing.**

Look at the properties of the *ProductTypeCode* attribute:

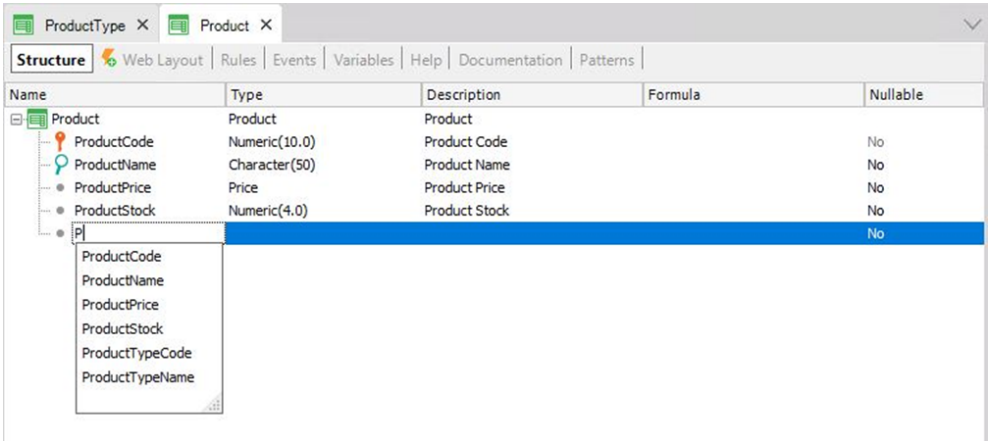


The Autonumber property is set to False by default. By changing it to True, all the new product types entered by the end user will be automatically numbered in sequence. So, set the Autonumber property to True for this identifier attribute and save the *ProductType* Transaction.

As explained before, each Transaction has a Web Layout automatically designed by GeneXus according to its structure. The following image shows the *ProductType* Web Layout:



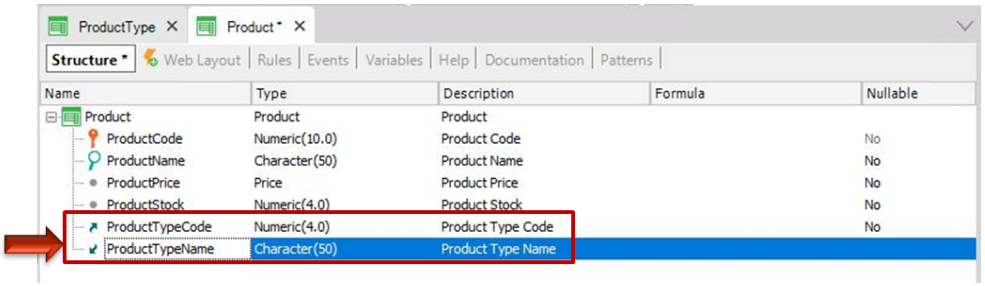
Now let's assign a product type to each product. To do so, go back to the *Product* Transaction and type "P" in a new line of the structure. The list of attributes existent in the Knowledge Base that begin with that letter is displayed:



Select *ProductTypeCode* and its entire definition is displayed.

Let's also include the ProductTypeName attribute in this Transaction, as users may want to see the corresponding product type name in the form after selecting a product type code while executing this Transaction.

Let's focus on these two attributes, included in more than one Transaction:



ProductTypeCode is the identifier attribute on the *ProductType* Transaction (more specifically, it is the primary key of that Transaction). So, when a primary key is included in another Transaction, GeneXus understands that the attribute has the role of a foreign key there.

Including an attribute that is a Transaction primary key in another Transaction allows you to relate both Transactions.

GeneXus establishes relations through attribute names, so when it finds attributes with the same name in different Transactions, it assumes that they refer to the same concept.

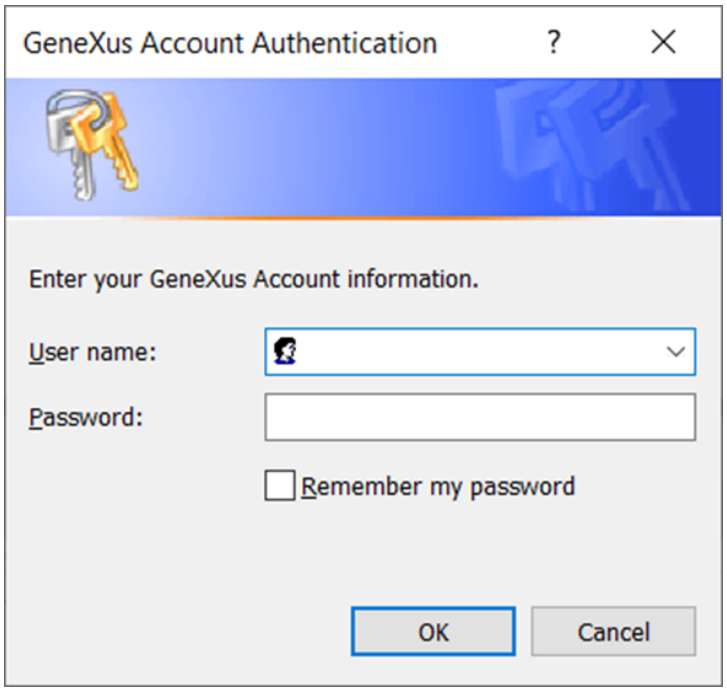
The *ProductTypeName* attribute is also present in both Transactions. However, it is not marked as the identifier of any of the defined Transactions. Therefore, GeneXus takes it as a **secondary** attribute. GeneXus will then include *ProductTypeName* in the *ProductType* physical table that it will create in the database and not in the *Product* physical table.

The concepts of Transaction and physical table are not the same. Keep in mind that Transaction is the GeneXus object that you create in the Knowledge Base to represent an object of reality. Upon considering its structure and the rest of the Transaction structures defined in the Knowledge Base (and also taking into account some properties), GeneXus will determine the physical tables that must be created in the database, as well as the attributes that must be stored in each table.

When executing the *Product* Transaction form at runtime, the user must enter for the *ProductTypeCode* attribute (which is a foreign key attribute there) a value that has been previously registered through the *ProductType* Transaction. Otherwise, an error will be displayed. After entering a valid *ProductTypeCode*, its *ProductTypeName* value will be obtained and shown on screen.

GENERATING AND RUNNING THE APPLICATION FOR THE FIRST TIME

If you want to generate and execute the application for the first time, you only have to press F5 and the following dialog box will be displayed:



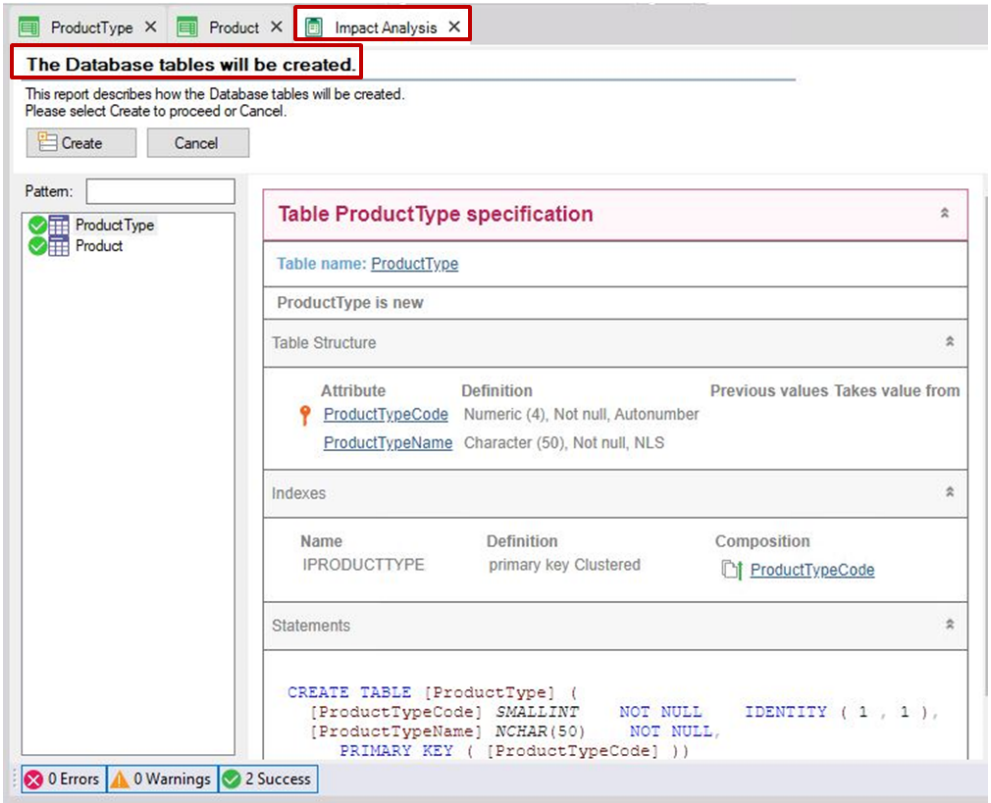
Use your GeneXus Account credentials to complete the following fields:

- *User name* or *email* corresponding to your GeneXus Account.
- *Password* corresponding to your GeneXus Account.

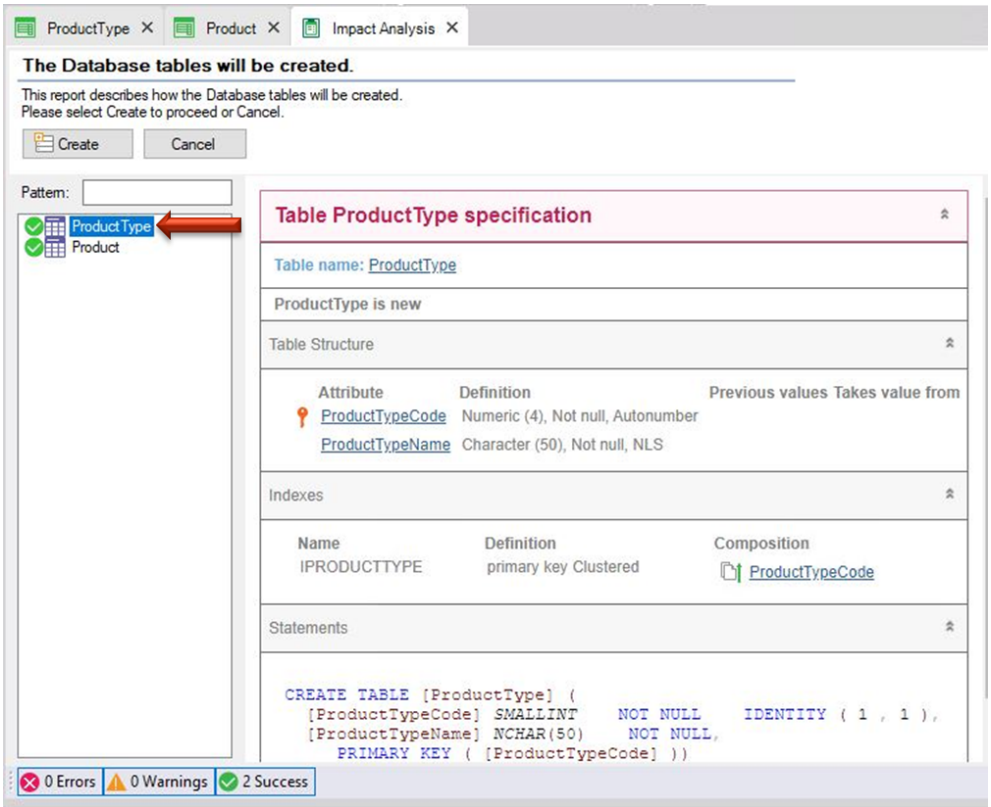
Next, press the OK button.

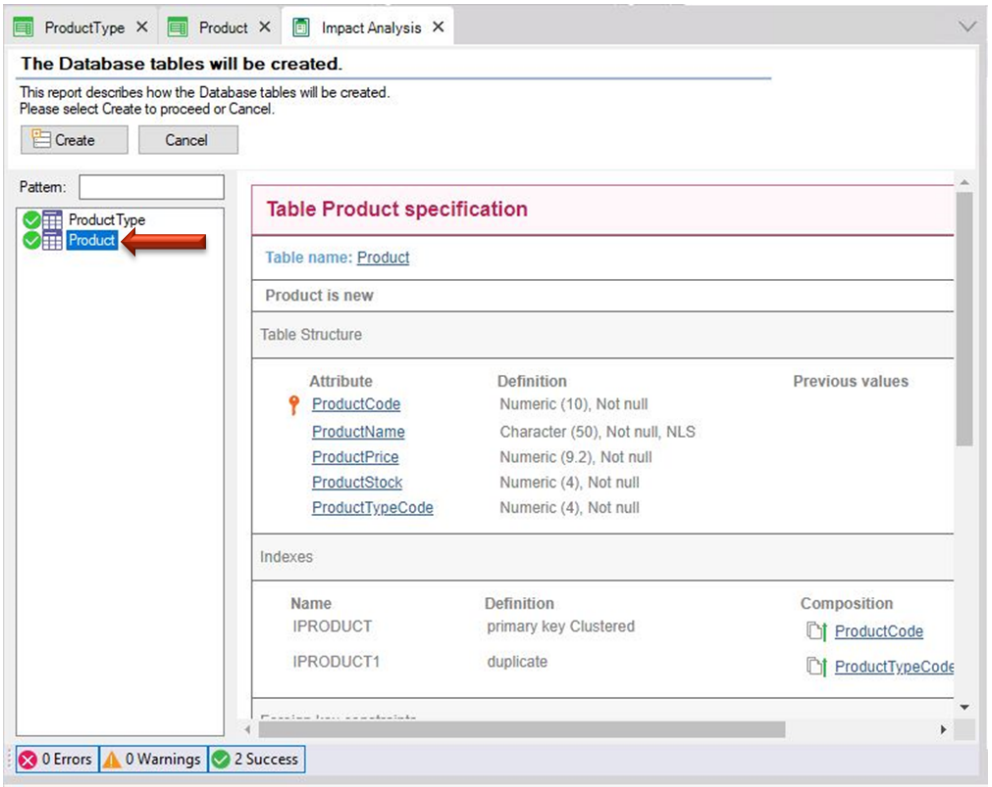
GeneXus will evaluate the impact caused by the new definitions in the Knowledge Base, and it will show a report under the name **Impact Analysis**.

By clicking on each table (*ProductType* and *Product* on the left) the attributes that will be included in them will be displayed on the right.



Observe each table structure proposed by GeneXus to create in the database:

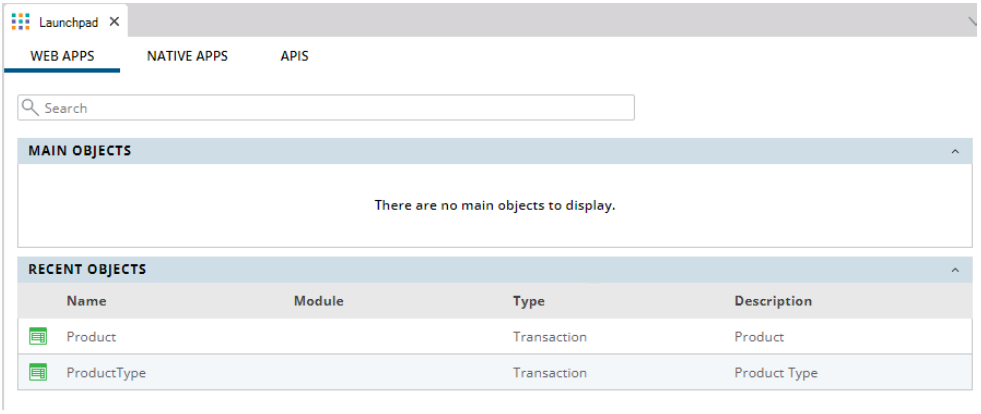




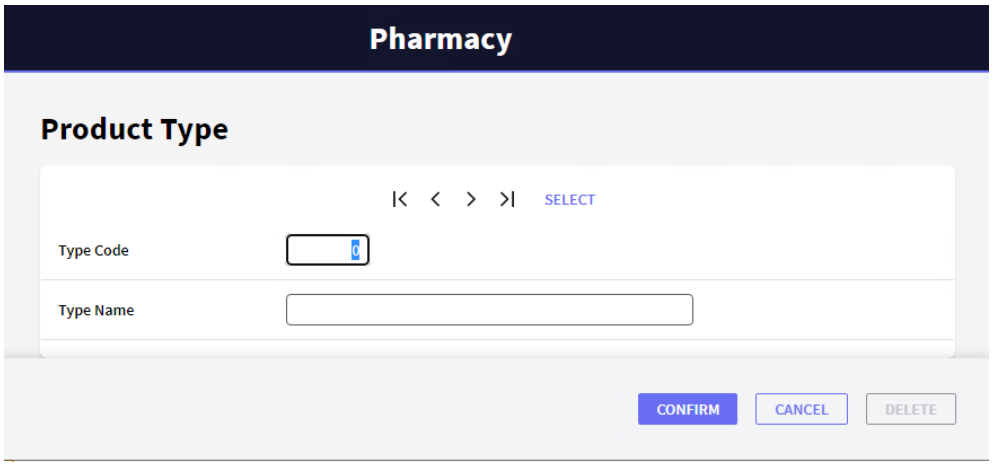
Note that, as it was previously explained, the *ProductTypeName* attribute is not included in the *Product* physical table that will be created, despite the fact that you included it in the *Product* Transaction structure (with the purpose of being shown in its layout).

If you agree with the Impact Analysis proposal, click on the Create button and GeneXus will start creating the programs needed to build the database (still inexistent), as well as the tables with their structures in that database. Next, GeneXus executes these programs and after creating the database and the tables, it will generate all the necessary lines of code - in the selected programming language - to obtain the application that will enable the users to insert, update and delete product types and products.

You are informed if the result was successful or if there were any errors or warnings. After that, you will be able to run and test the application. To do so, the Launchpad window will be opened offering a quick way to execute the defined objects:

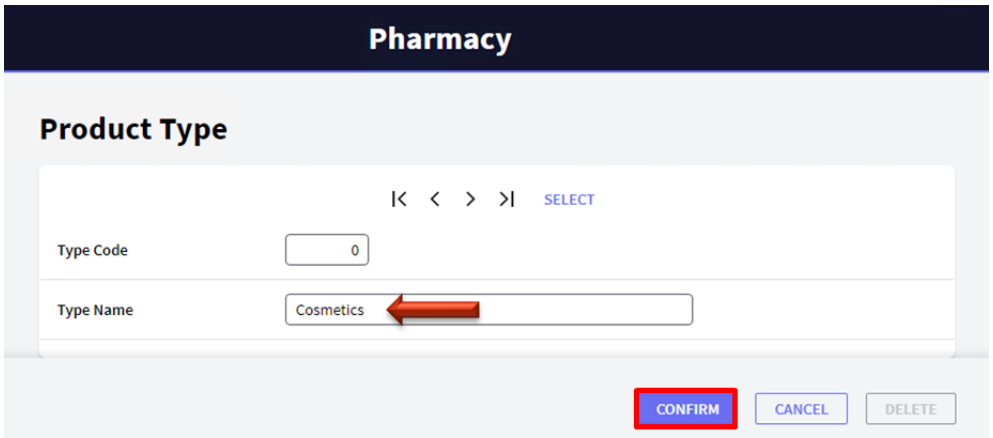


Click on the *ProductType* link:

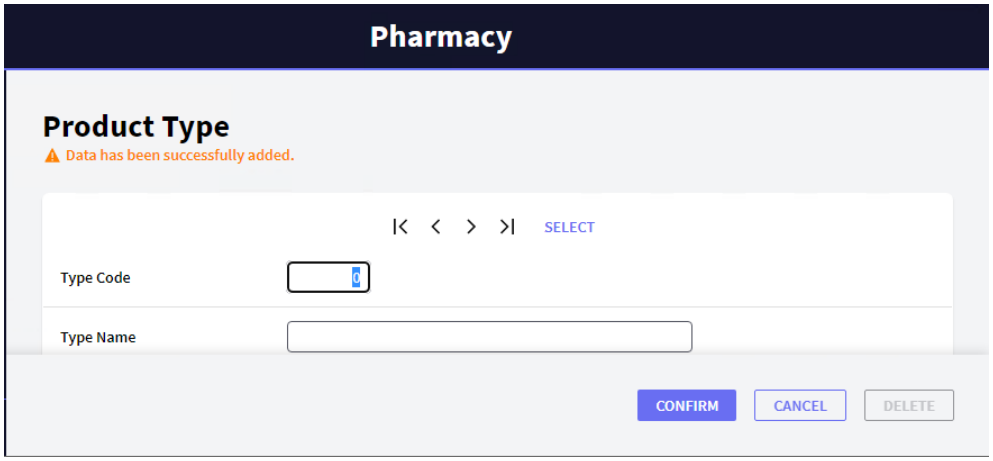


The Web browser is opened by default, and the page above is shown. It allows you to add, update and delete *product types*. Enter the first product type.

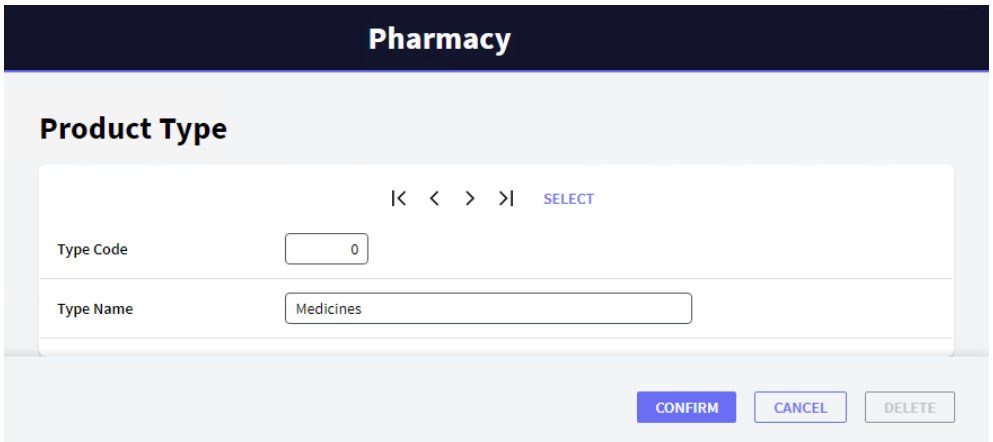
Since the *ProductTypeCode* attribute has the Autonumber property set to True, the users will not have to enter a value for the identifier because it will be numbered automatically. So, enter the product type name:



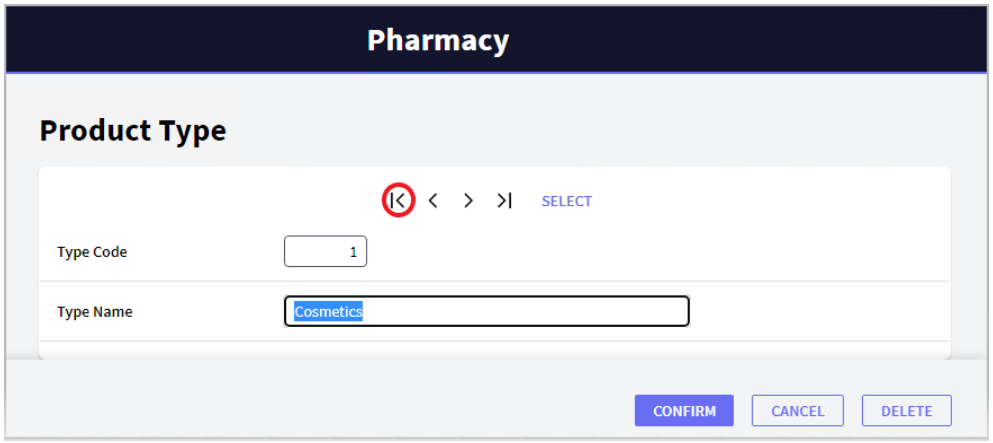
After entering the product type name and clicking on the Confirm button, a message will be displayed to inform that the data was added successfully; meanwhile, the form is cleared and get ready to enter another product type:

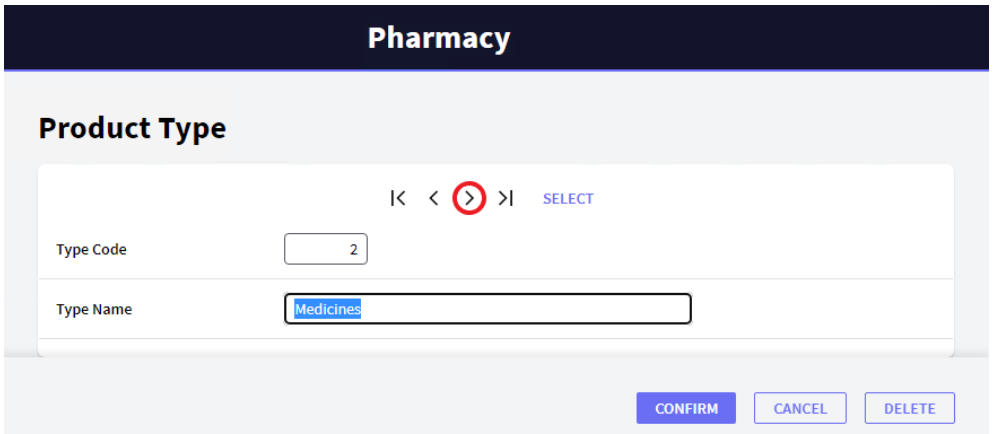


Enter the second product type:

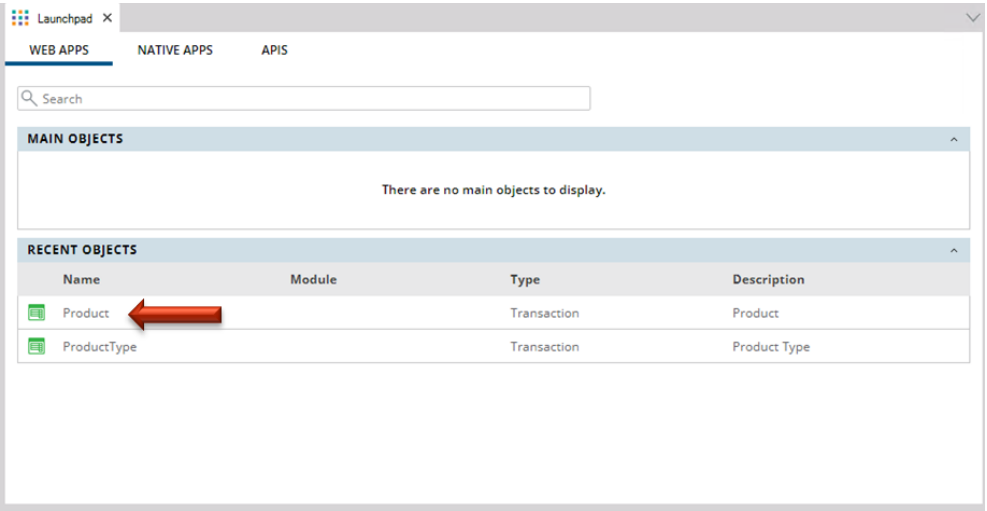


Click on the Confirm button and then you can browse the data to confirm that they were numbered:

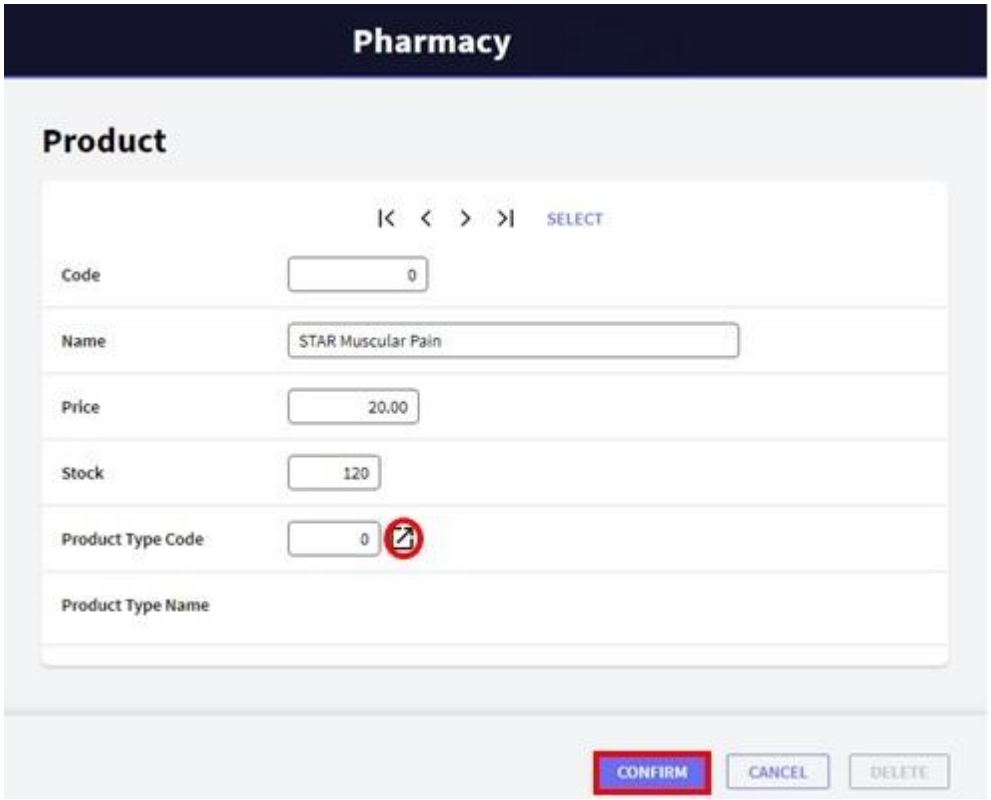


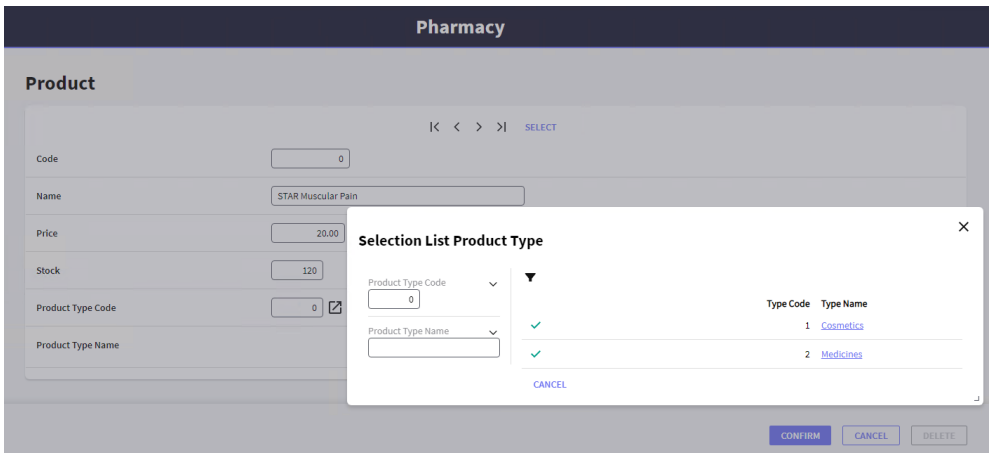


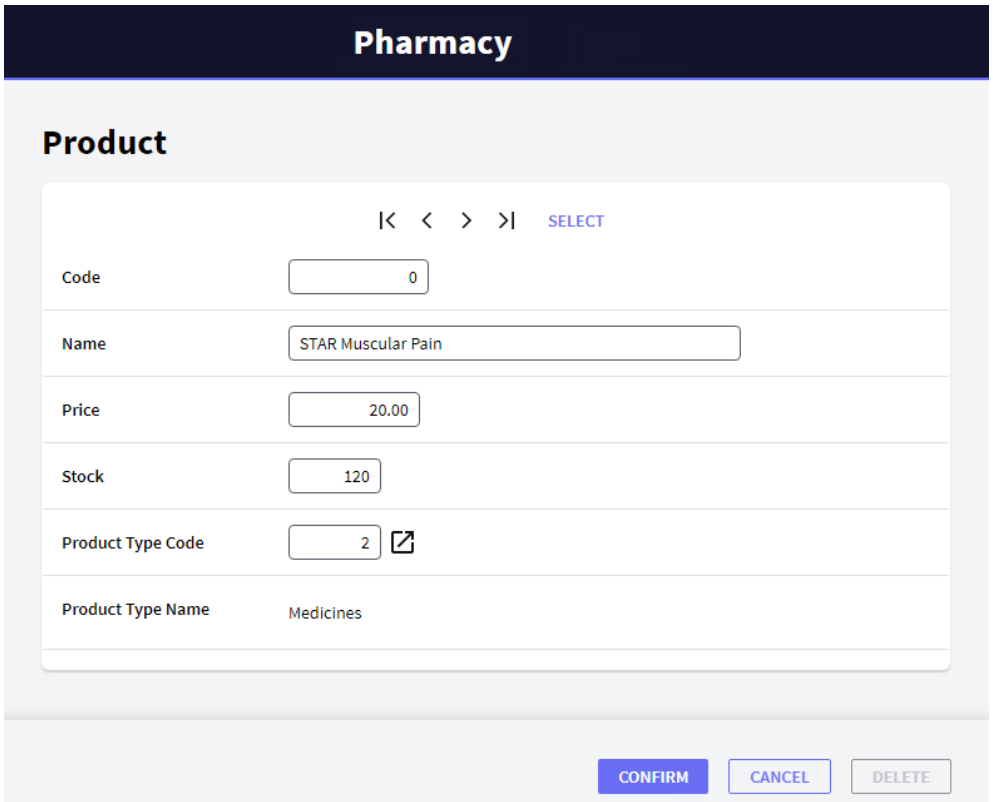
Now, execute the *Product* Transaction. To do so, go back to the Launchpad and execute the *Product* Transaction by clicking on its link:



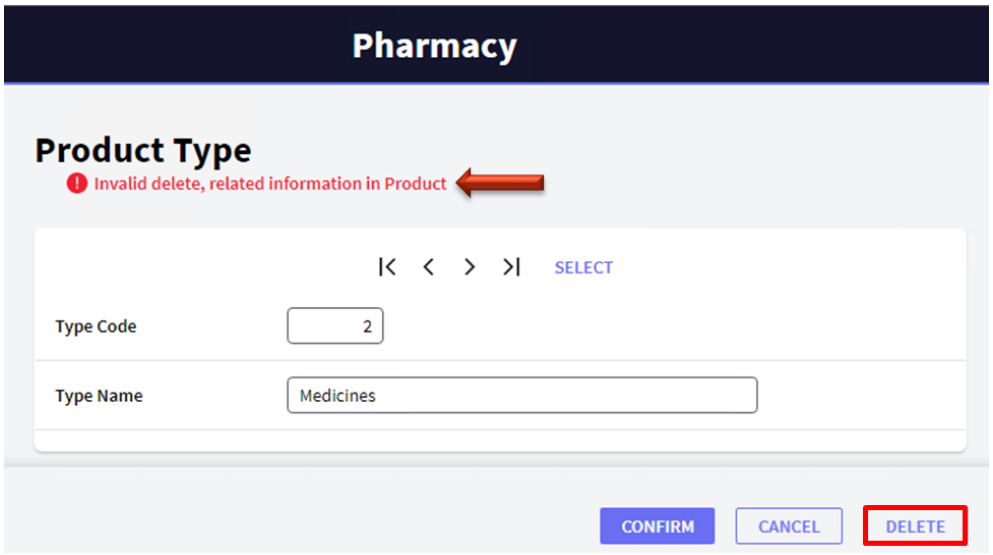
Try adding the first product. You must indicate the product type. If you remember the product type code you can enter it. Another option is to select it from a list by clicking on the arrow.




Pharmacy
Product
SELECT
Code
0
Name
STAR Muscular Pain
Price
20.00
Stock
120
Product Type Code
0
Product Type Name
Selection List Product Type
Product Type Code
0
Product Type Name
Type Code
Type Name
1
Cosmetics
2
Medicines
CANCEL
CONFIRM
CANCEL
DELETE


Pharmacy
Product
SELECT
Code
0
Name
STAR Muscular Pain
Price
20.00
Stock
120
Product Type Code
2
Product Type Name
Medicines
CONFIRM
CANCEL
DELETE

Now, try deleting the “Medicines” product type. To do so, select that product type and press the Delete button:

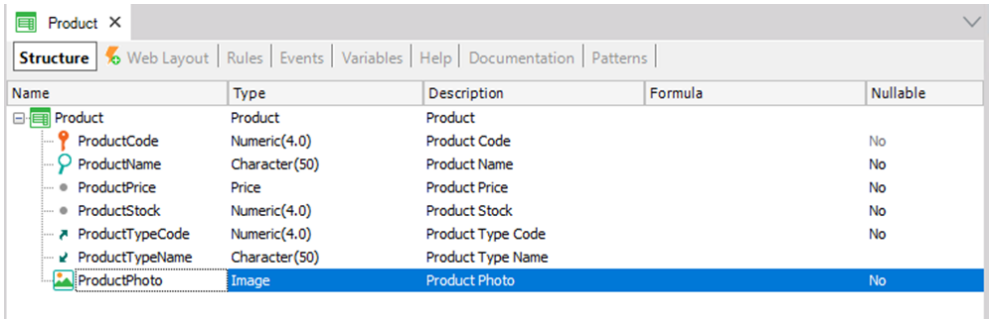


A message informs you that the deletion cannot be performed because related data exists in Product (the STAR Muscular Pain medicine is a product that belongs to this product type).

THE APPLICATION GROWS UP

Now suppose that one of the requirements you were asked for is that you need to have a picture of each one of the products.

To do that, go back to the *Product* Transaction, and you have just to enter a new attribute called *ProductPhoto*:

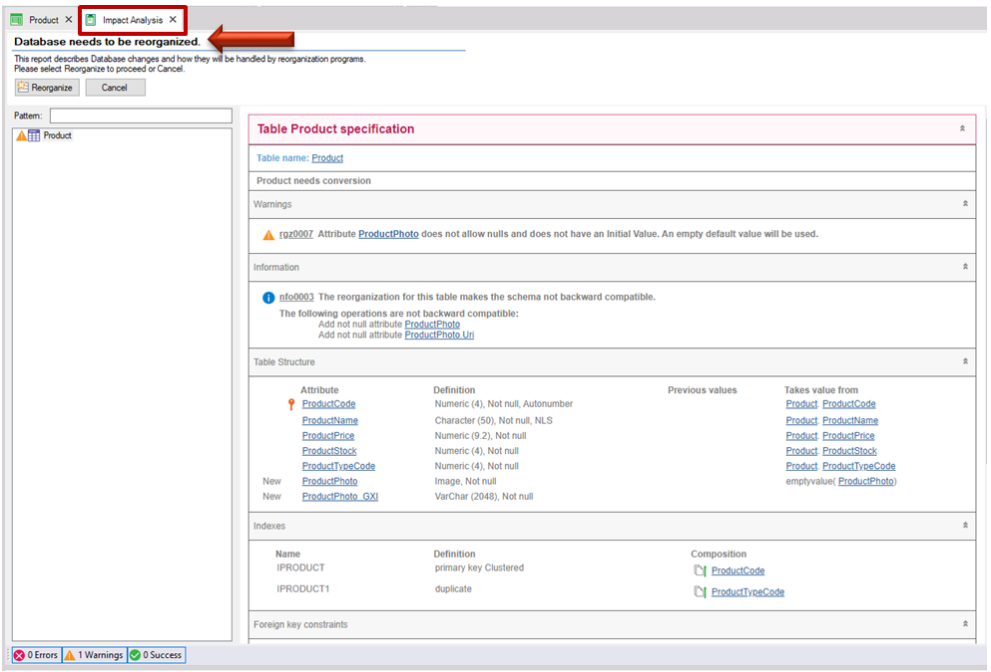


Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(4.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	Product Type Name		
ProductPhoto	Image	Product Photo		No

The Image type enables you to store images.

The Web Layout is automatically updated, including the *ProductPhoto* attribute.

Press F5 and GeneXus will evaluate the impact caused by the new definitions in the Knowledge Base, and it will show the **Impact Analysis Report**:

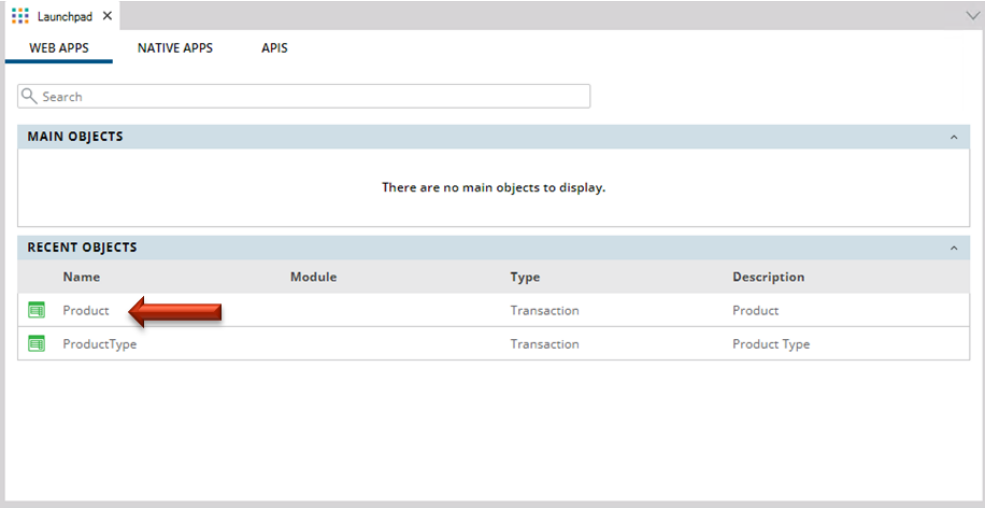


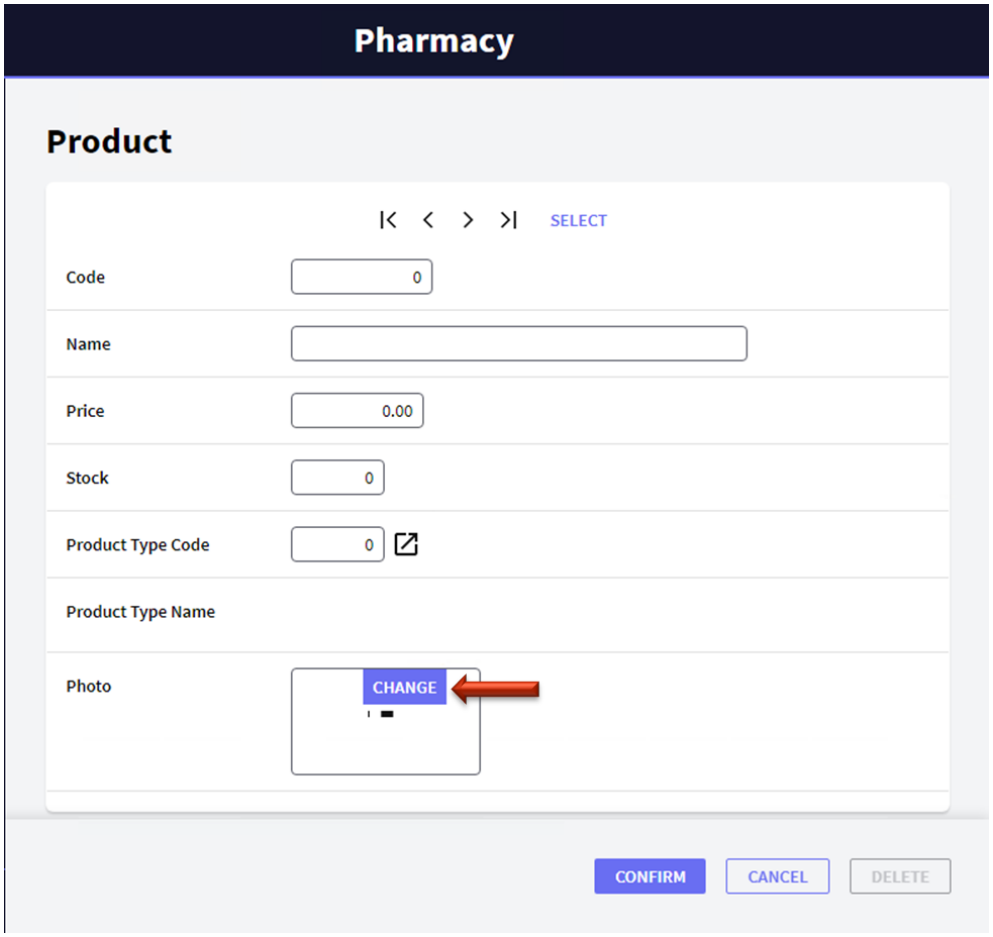
Remember: The *Impact Analysis Report* indicates the structural changes required in the database.

By reading the report, you will see that the main title, in this case, informs that **"The Database needs to be reorganized".** The term "Reorganize" implies the task of making changes to the database. In this particular case, the report indicates that the *Product* table must be updated.

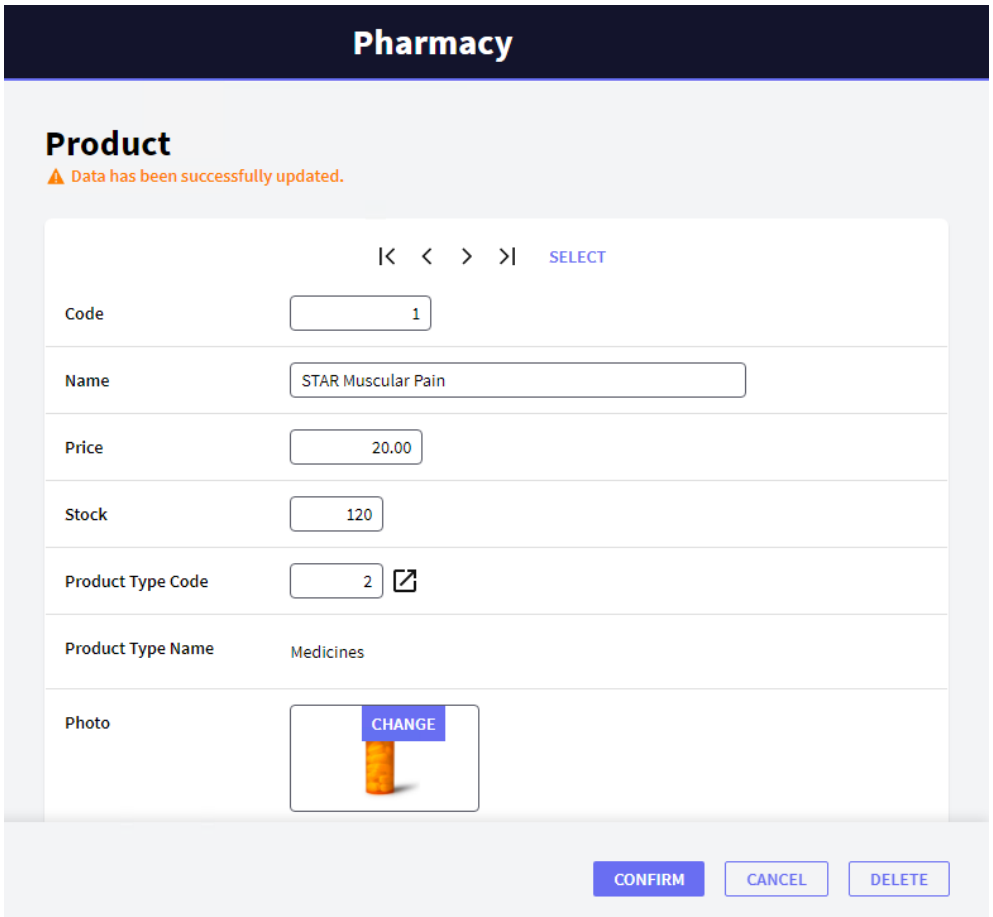
By clicking on the Reorganize button, GeneXus will create and execute the programs that will change the database. Then, it will generate the necessary programs that correspond to the application itself.

Note that you will be able to run the application again immediately, with the new definition included:





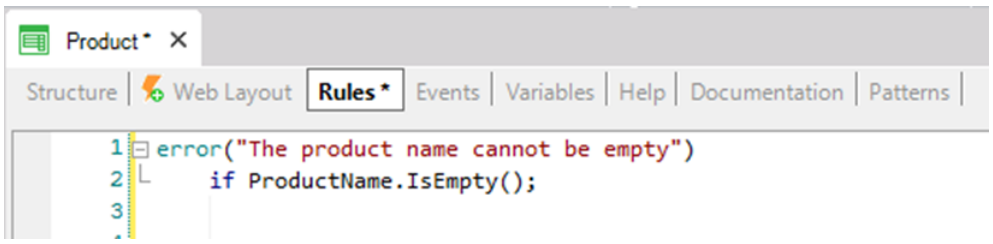
If you remember the product code you may enter it, or another option is to select it from a list by clicking on the SELECT button. From there, you can retrieve the “STAR muscular pain medicine” and upload its photo:



ADDING BUSINESS RULES

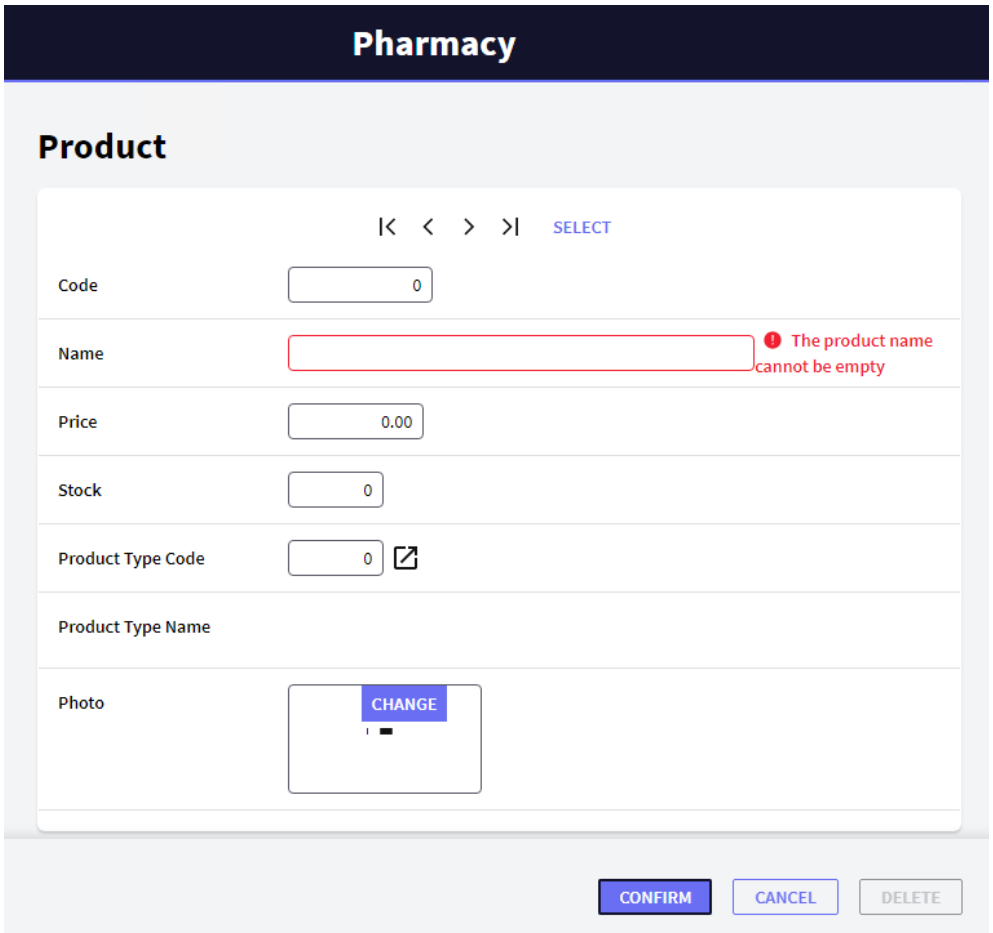
In addition to all the automatic controls included by GeneXus in the applications it generates, sometimes users request some specific controls. In Transactions, the rules that must be complied with, or the controls that you are asked to validate, are defined in the **Rules** section.

If, for example, a requirement is not to allow storing of products without a name, GeneXus offers a rule called **Error** that will enable you to avoid that:



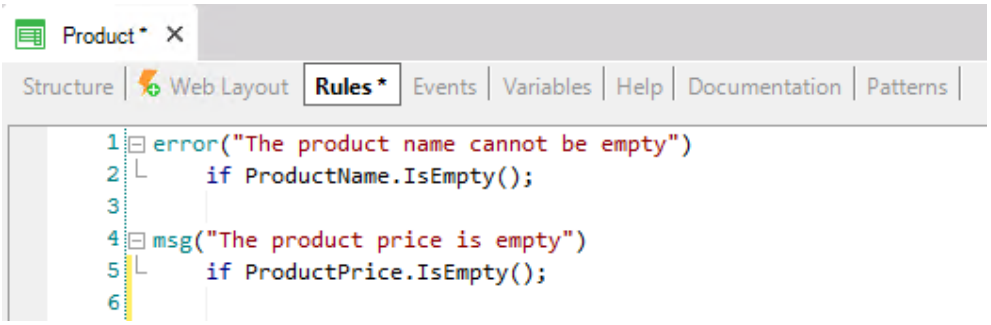
Press F5 and GeneXus will save and evaluate the new definitions included in the Knowledge Base. In this case, it will infer that it is not necessary to modify the database, so it will not show an Impact Analysis Report. GeneXus will generate the necessary code and after that, it will execute the application updated with the new definitions.

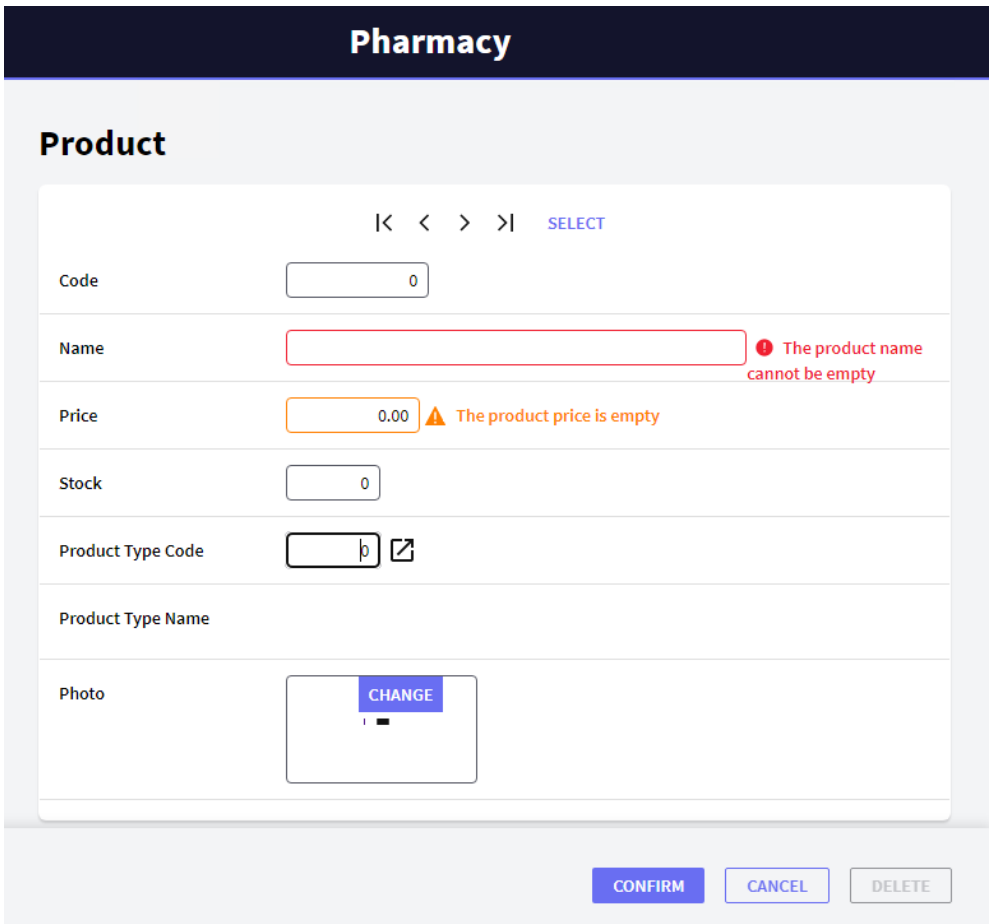
Run the *Product* Transaction. Note that if the product name is left blank, the rule you have defined is executed:



There is another rule whose syntax is very similar to the Error rule. It is called **Msg** and the only difference between them is that if the condition is met, in this case the message is displayed as a notice or warning, and the user can continue working.

If, for example, you want to inform that the product's price has been left blank without forcing the user to enter it, you can add the following rule in the *Product* Transaction:





This set of rules could be written in any other order and the result at runtime would be exactly the same, as GeneXus decides when each one of the defined rules should be triggered (when the user leaves each involved field, if the condition is true, etc.).

Of course, GeneXus offers more available rules to define different kinds of validations and actions.

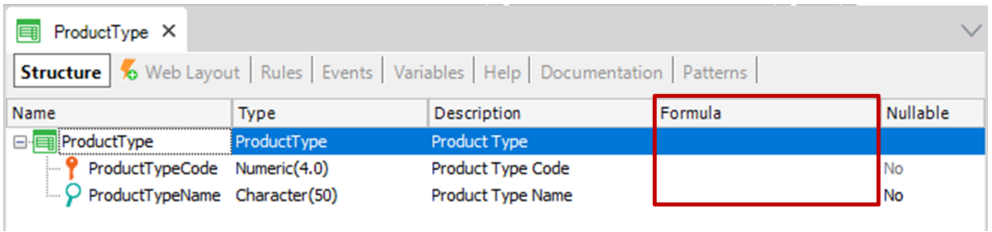
Each Transaction may need to have its own behavior rules defined.

DEFINING CALCULATIONS: FORMULAS

Applications are often required to make calculations that involve the values of specific attributes, constants and/or functions. For all these cases, GeneXus offers **Formulas**.

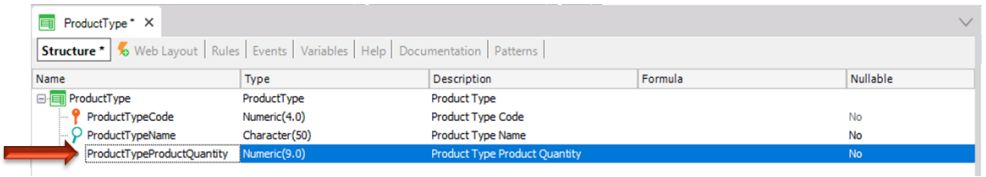
There are different possible ways to define formulas.

Let's start by learning what a **Global Formula** is. A global formula is a calculation you define associated with an attribute. Note that the Transaction structures contain a column labeled **Formula**:



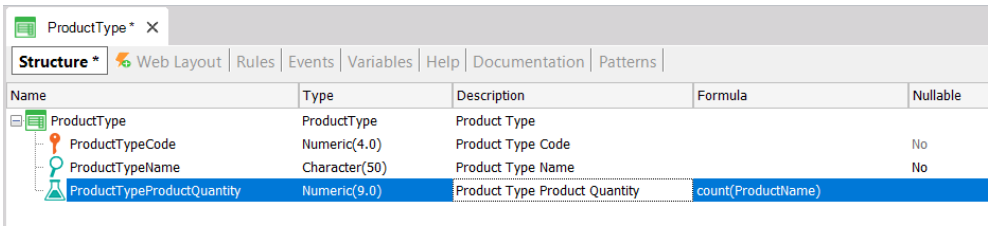
When a calculation is defined for an attribute in this column, this means that the attribute is virtual. In other words, it will not be physically created as a field in a table, because the value of the attribute will be obtained by doing the calculation every time it is needed.

Let's see this with an example. Suppose the pharmacy needs to know how many registered products there are of each product type at all times. So, let's define a new attribute in the *ProductType* Transaction with the purpose of defining it as a global formula:



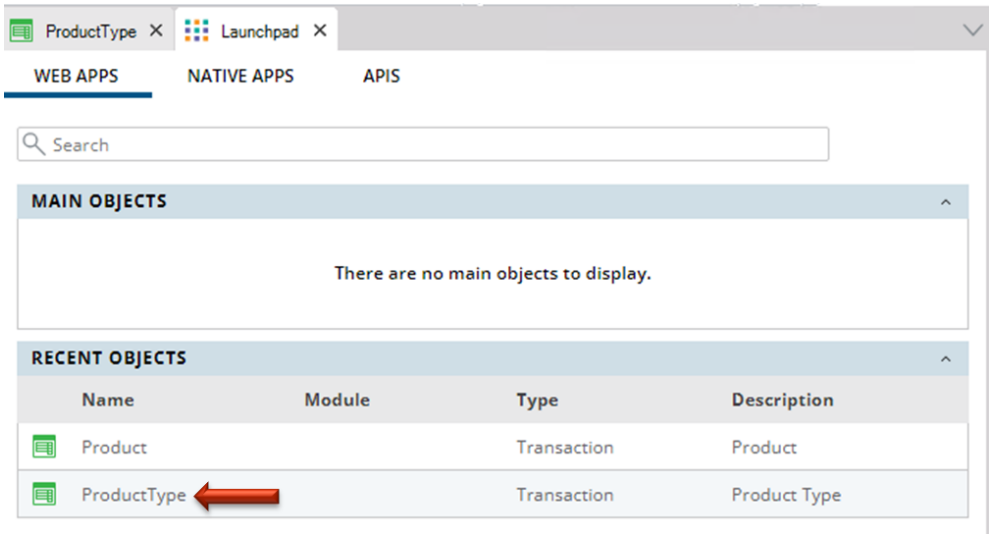
Let's now define the calculation associated with the *ProductTypeProductQuantity* attribute.

GeneXus offers a formula called Count to calculate the pharmacy's need (there are many others, like Sum, Average, etc.).

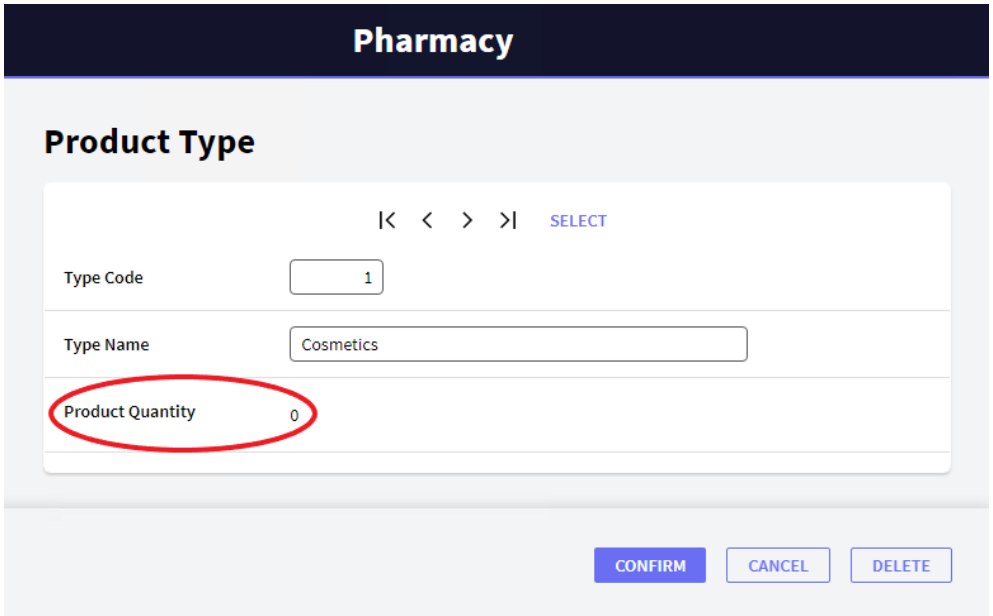


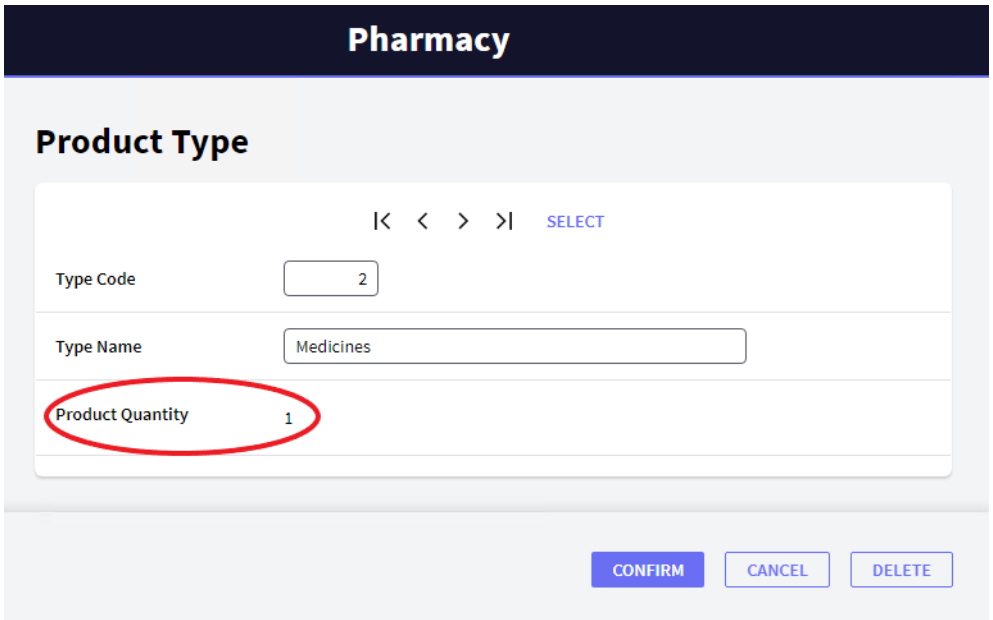
The attribute referenced inside the parentheses of the formula provides GeneXus with the information of the table to be navigated to do the calculation (in the definition above, GeneXus knows that it has to count in the Product table). Then, if GeneXus detects a relation between the table it will navigate (Product) and the context where the formula attribute is defined (ProductType), it will only consider the related records for the calculation. In this example, *ProductTypeCode* is present in both contexts: where the formula is defined and in the table to be navigated for doing the calculation of the formula. So, only the products of each product type are counted and not all the products recorded in the navigated table, will be considered. If no relation is found, then GeneXus will do the calculation considering all records in the navigated table.

Press F5. You can see that no physical changes will be made to the database. GeneXus will only generate the necessary programs and you will get the Launchpad again:



Execute the *ProductType* Transaction in order to see how the product quantity is always calculated for each product type at the time:





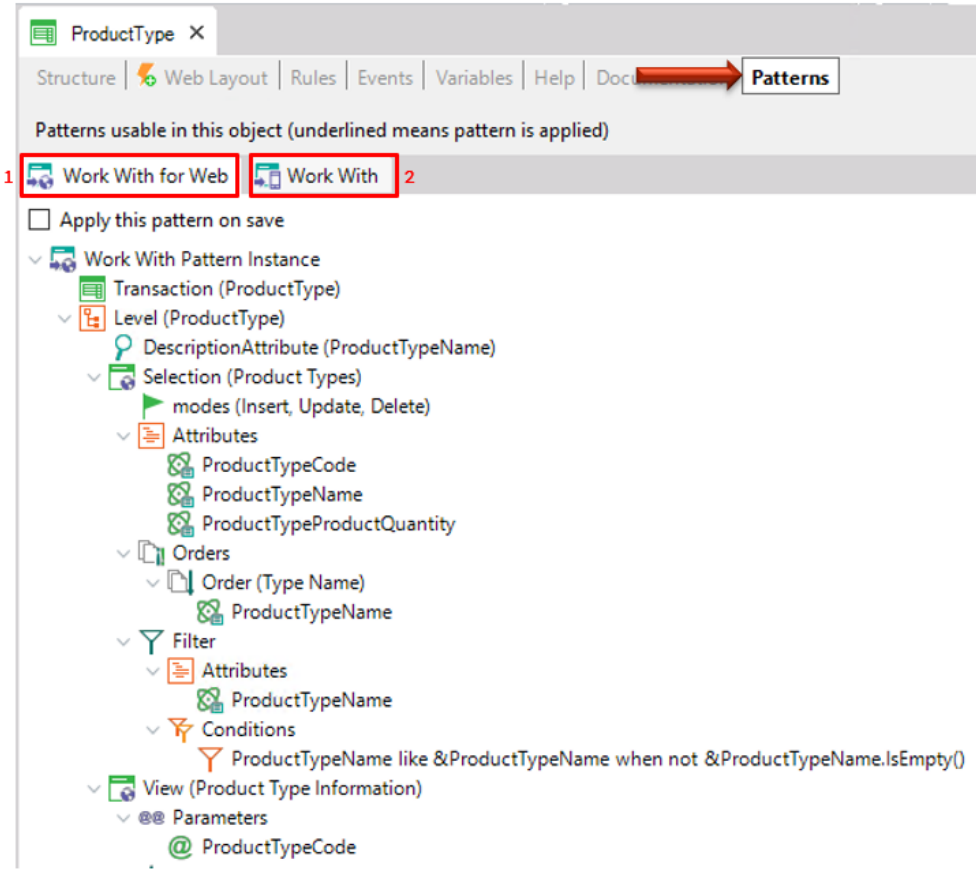
You can add more products in order to verify how the product quantity is always calculated at the time for each product type.

USING PATTERNS (FOR WEB AND FOR MOBILE DEVICES)

Patterns allow you to empower your applications even more, automatically.

Applying a pattern is really easy, and as soon as you do it, GeneXus creates objects, codes and settings to provide interesting behaviors without the need for us to program them.

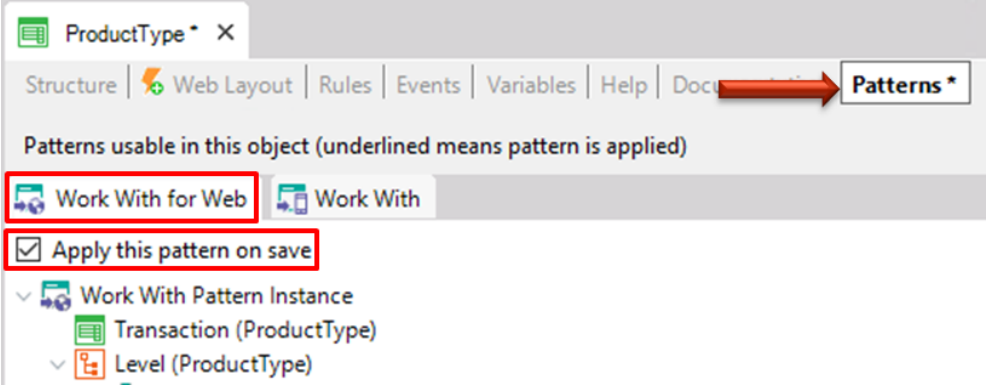
Look at the Patterns section of a Transaction. For example, in the *ProductType* Transaction, select the Patterns section:



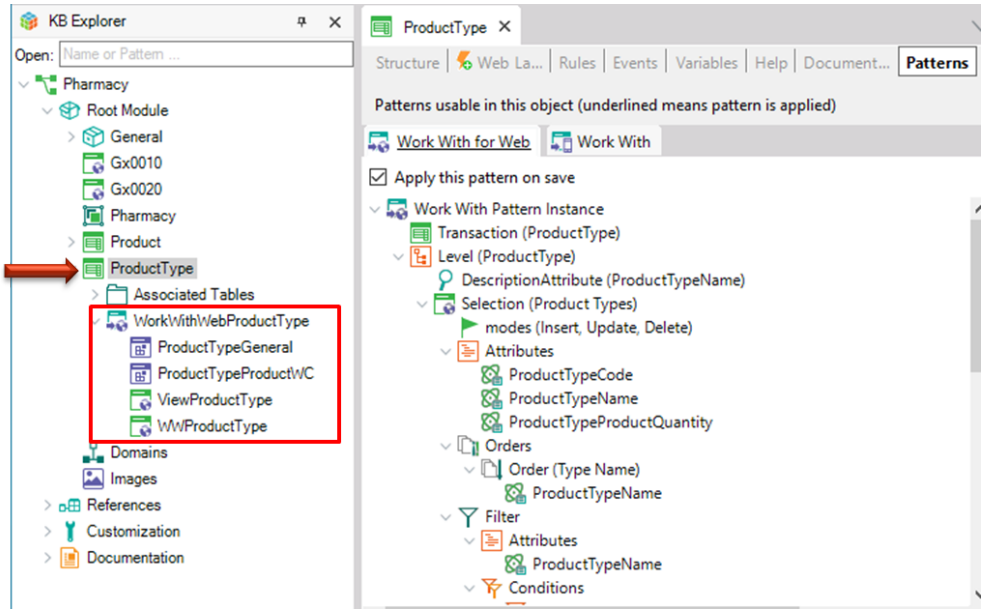
Note that two tabs are available, each one of them offering a different pattern to be applied to the same Transaction.

First of all, choose the *Work With for Web* tab, in order to see how simple applying a pattern is, and how quickly you obtain interesting results.

You only have to click on the **Apply this pattern on save** checkbox and save ( ):

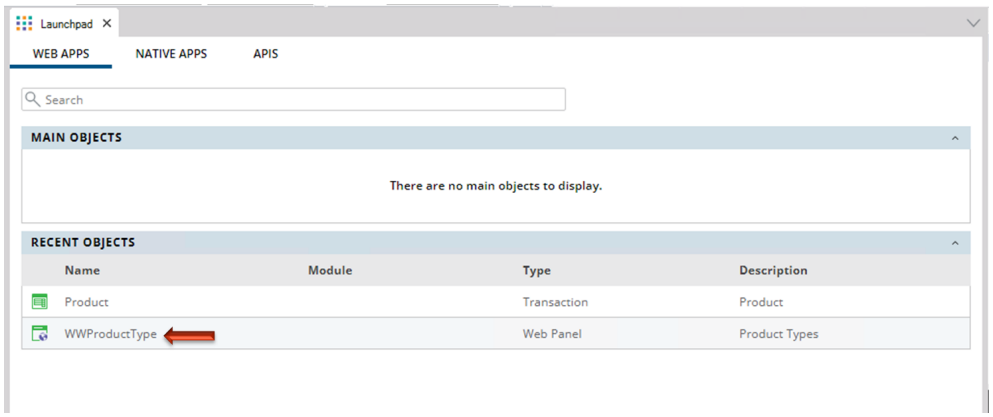


After that, if you look for the *ProductType* Transaction in the **KB Explorer**, you can see that several objects are located below the Transaction:

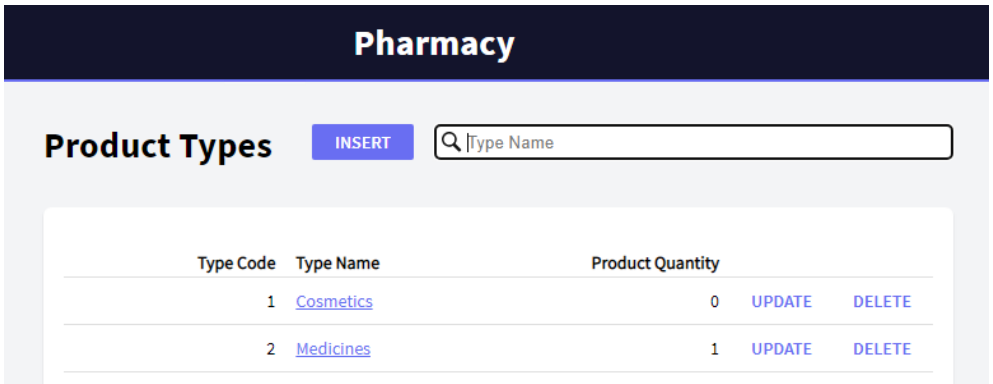


They were created by GeneXus when the *Work With for Web* pattern was applied.

Press F5 to view the results in runtime:

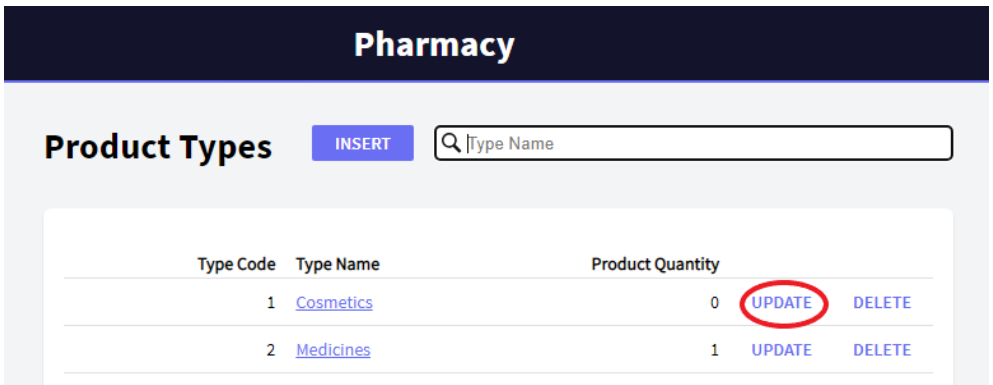


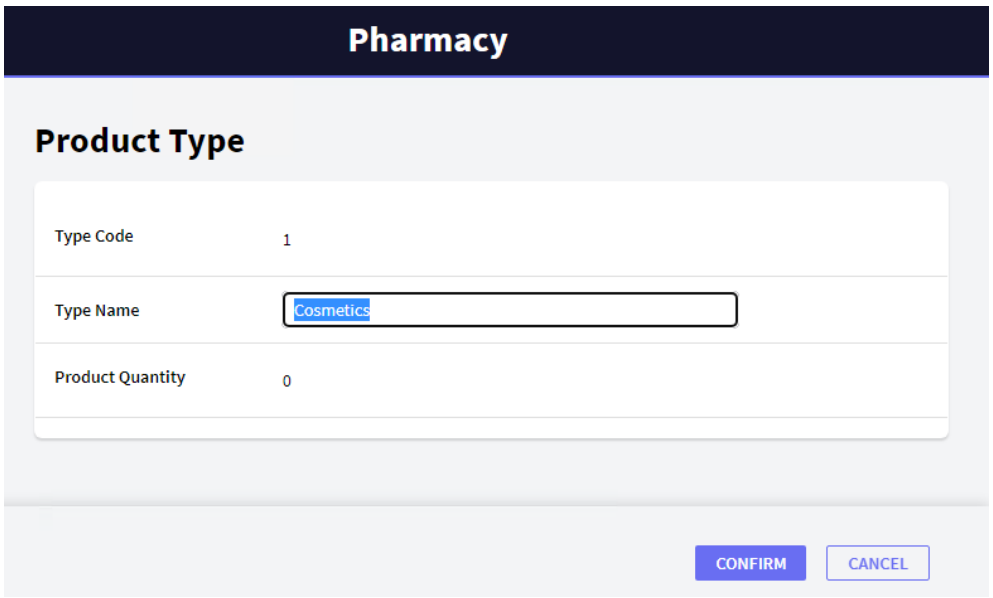
Look at the last link. You're offered to “work with Product Type” and the *ProductType* Transaction will be called from there. Click on that link.



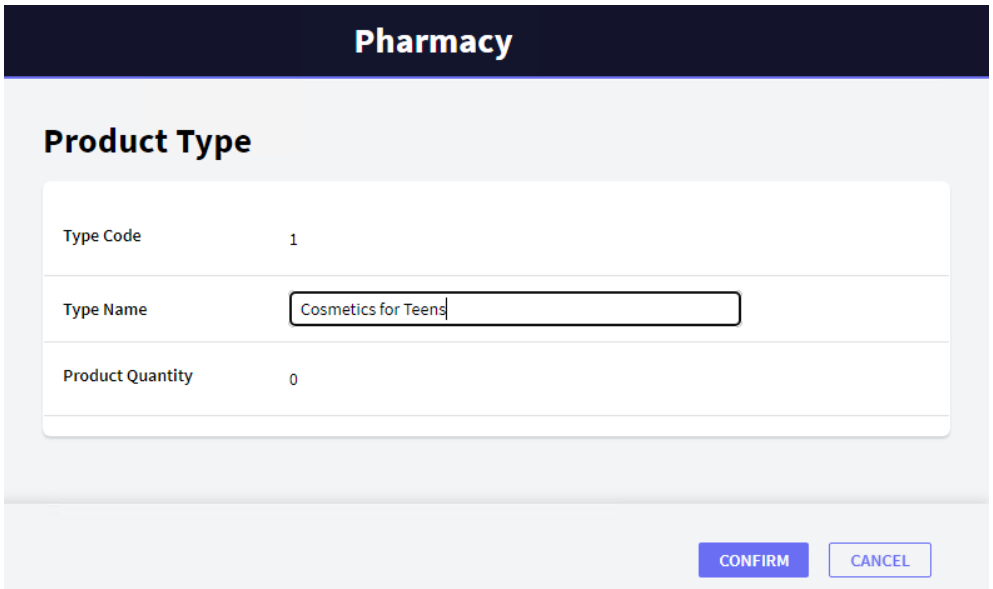
You can see that a page showing all the product types is opened. This page lets the users work with the product types with a wider range of features.

For example, click UPDATE for the first line:

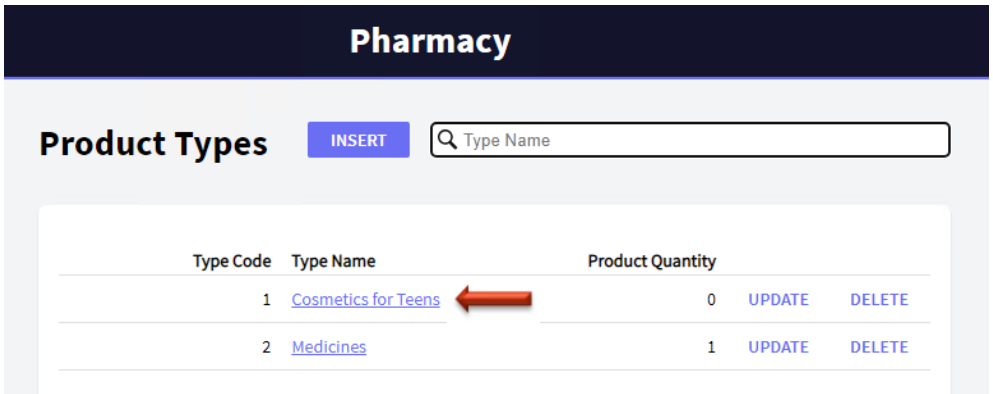




You can see that the *ProductType* Transaction is opened offering to edit the details of the product type in that line. Let's edit the type name and confirm:

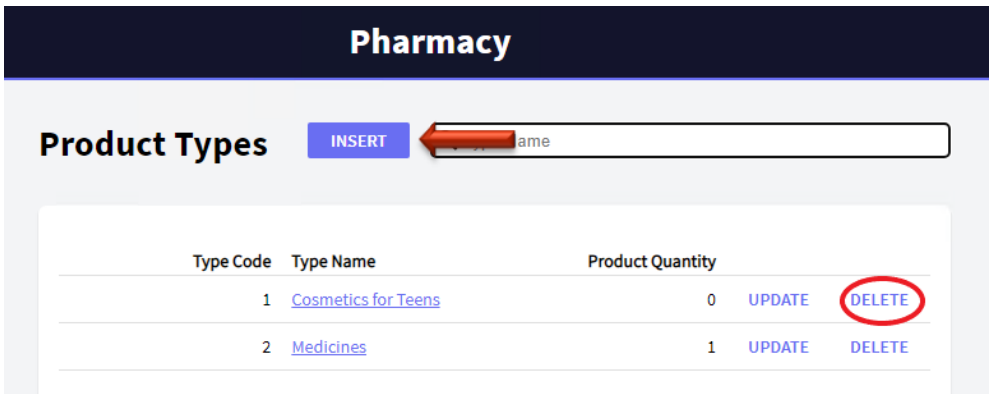


After the edition and confirmation, the application returns to the *Work With Product Types* page and you can see the change:

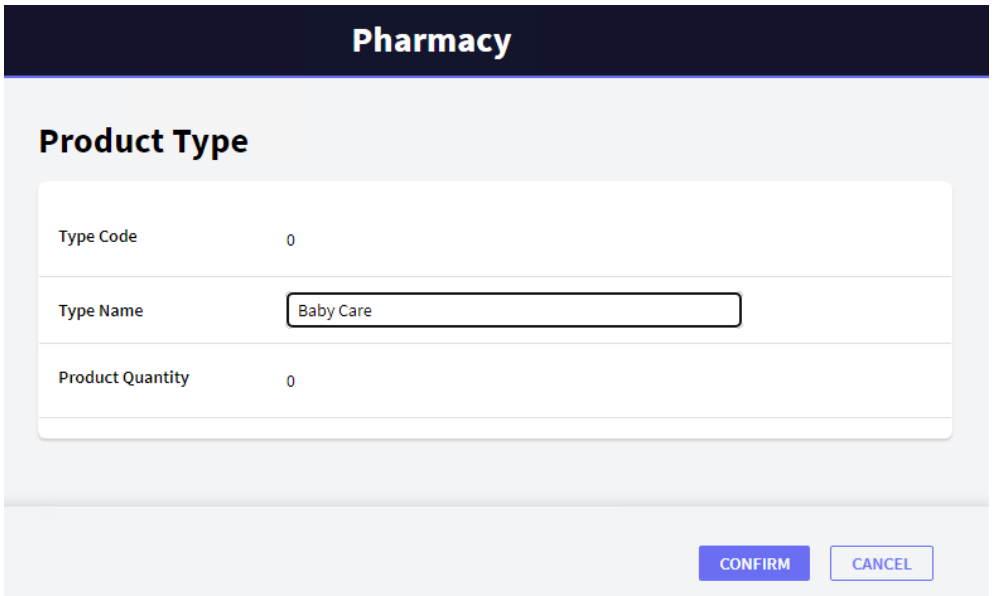


The DELETE link offers the users to delete the product type in the line.

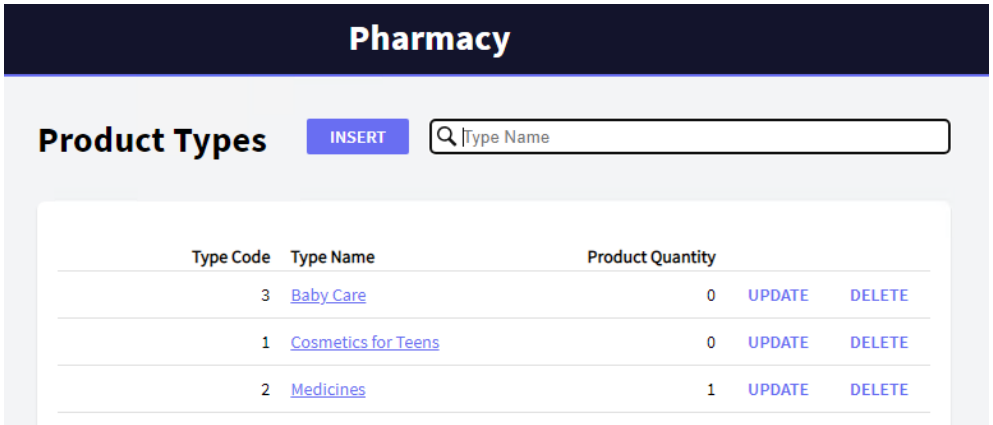
Meanwhile, the INSERT button located outside the grid, allows the users to add new product types.



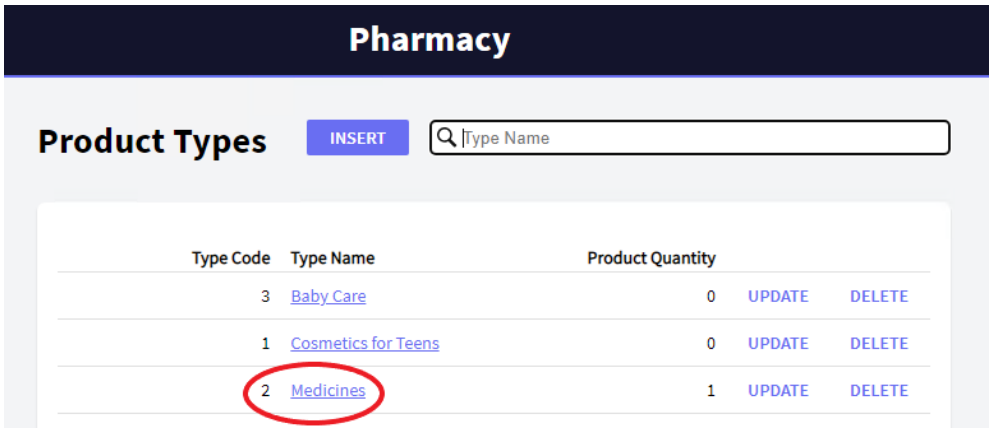
By clicking on the INSERT button, the *ProductType* Transaction is opened, ready for adding a new product type. Press it in order to enter a new product type (remember in this case it's only necessary to enter the product type name because you have set the key attribute's Autonumber property = True):



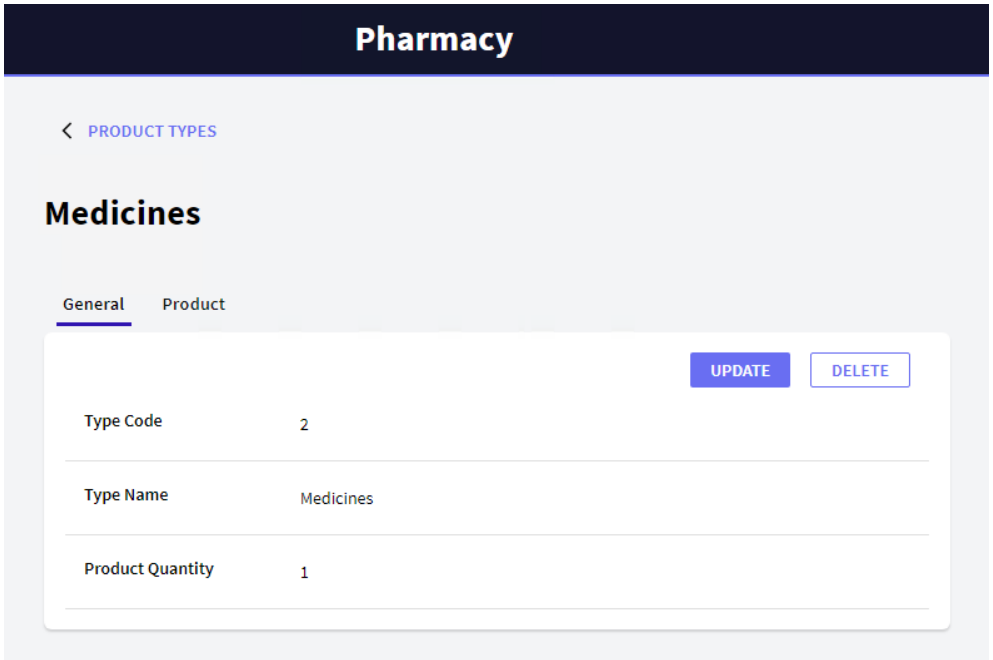
After the insertion, the application returns once again to the *Work With Product Types* page:



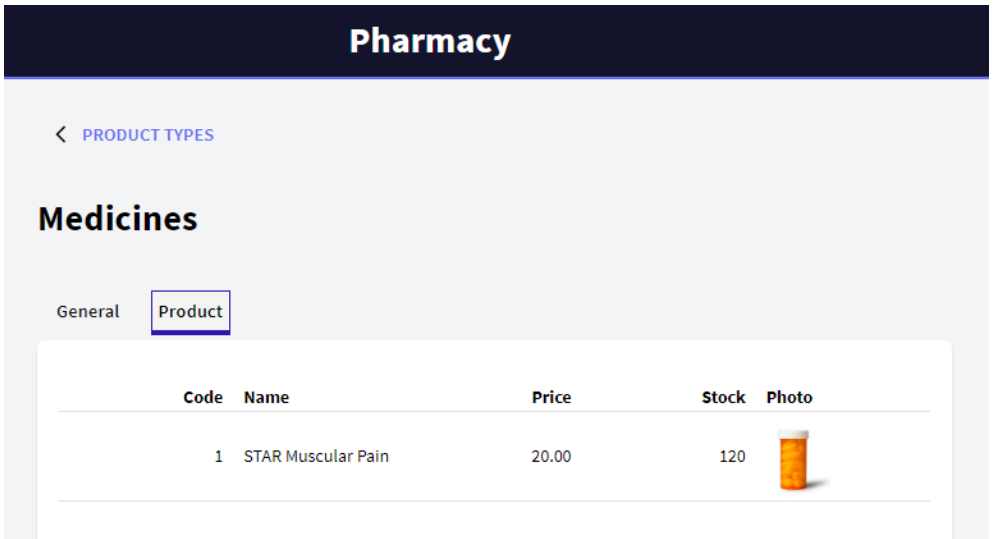
Note that each product type name has a link. Click on the product type: *Medicines*.



As you can see below, all the details of the selected product type are displayed in a first tab (*General tab*):

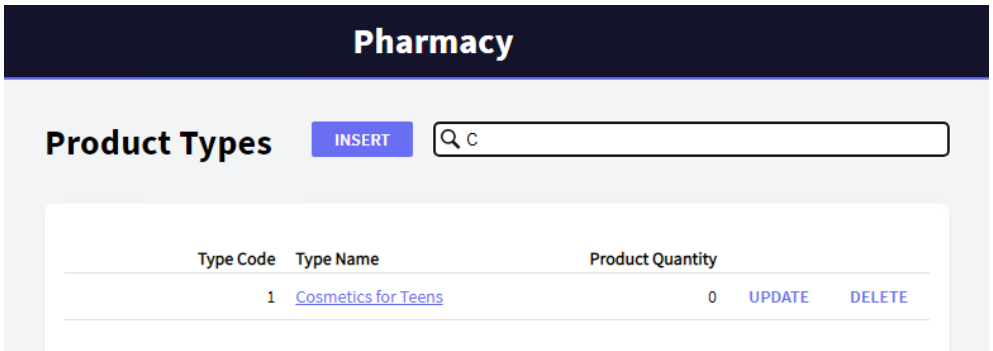


Another tab (*Product tab*) shows the list of products that belong to that product type:



The ***Product tab*** was automatically generated because **each product type has several related products**. If each product type had also several related data of other kinds, more tabs would have been generated in order to show each list of data related to the product type.

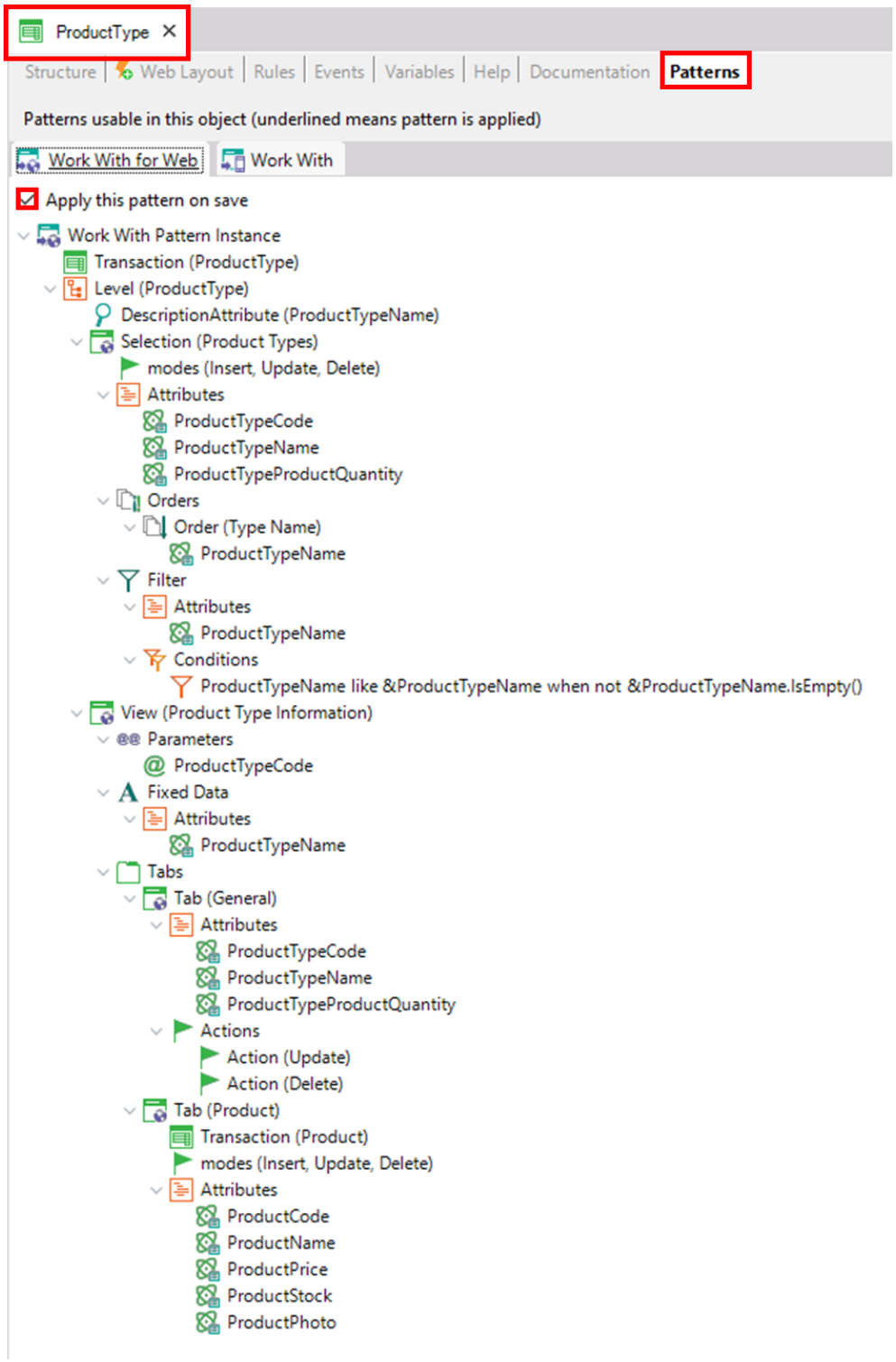
Now, let's go back to the *Work With Product Types* page by clicking on the link offered in the top left corner.



Note that it's possible to search by name. This means that if, for example, the user types "C", only the product types that begin with this letter will be displayed:

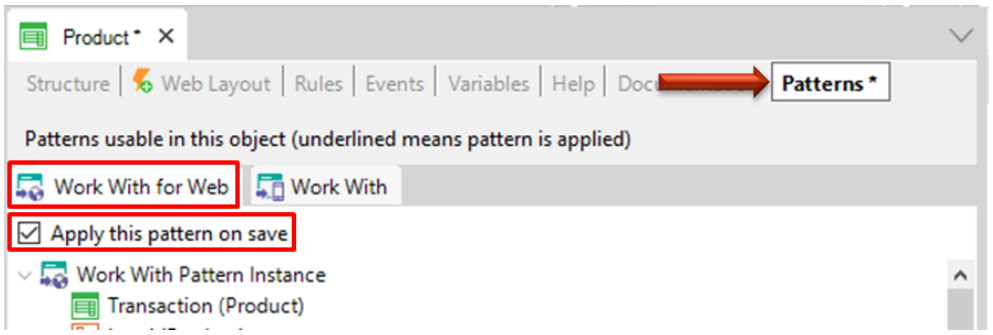
Go back to GeneXus again. So far, you have only selected **Apply this pattern on save** in the *Work With for Web* tab of the *ProductType* Transaction, and after saving you have seen all the features that are automatically generated.

What you may not have noticed is this configurable tree:

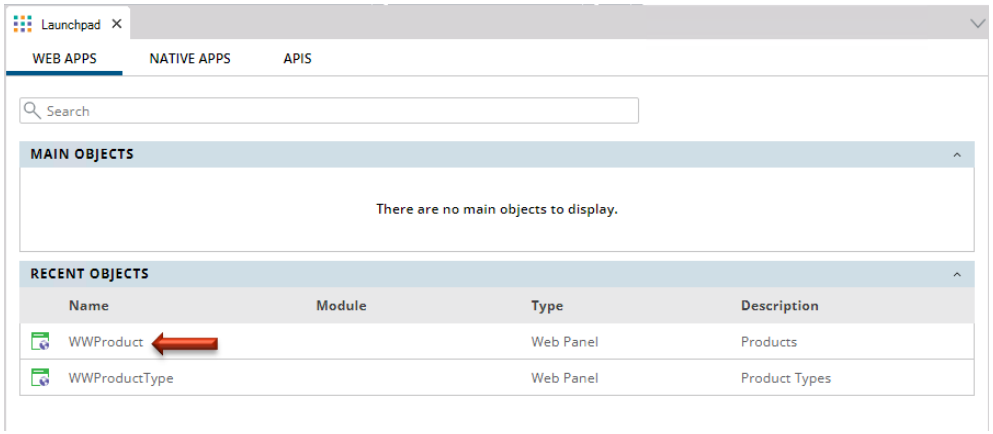


It has configurable nodes, sub-nodes and elements, so that you can customize the behaviors to be generated (i.e., change the search criteria).

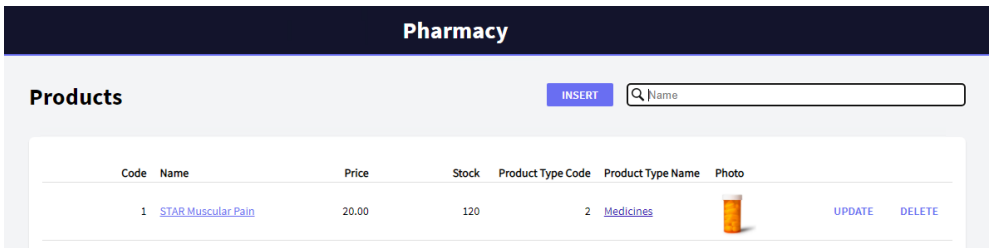
Now let's apply the *Work With for Web* pattern to the *Product* Transaction, too. As explained before, you only have to open the *Product* Transaction and in its Patterns section, you have to select the *Work With* tab. Then, check the **Apply this pattern on save** option and save:



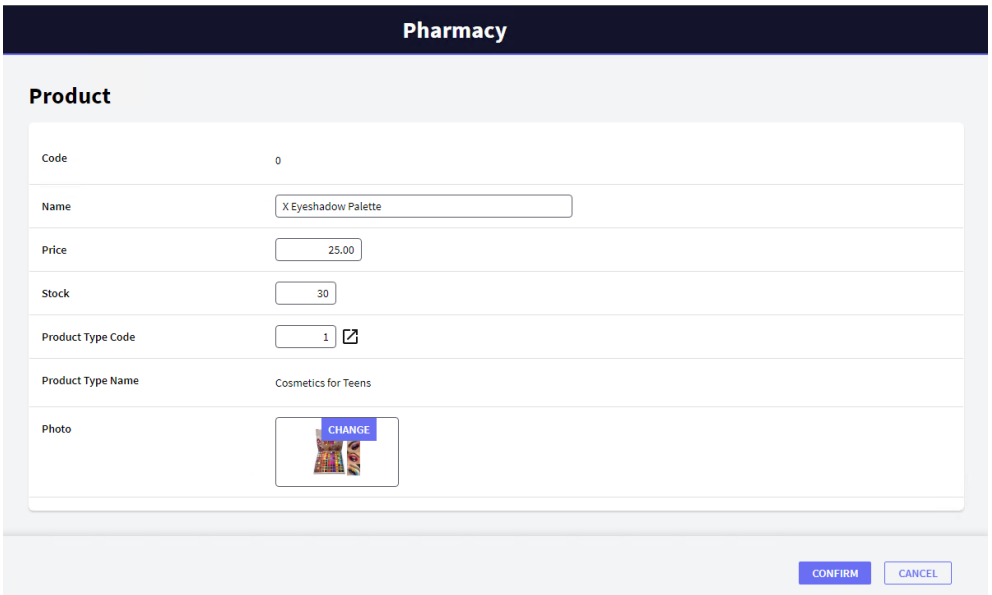
Press F5. GeneXus proceeds to generate the necessary programs and execute the application with the changes. Click on the shown link:



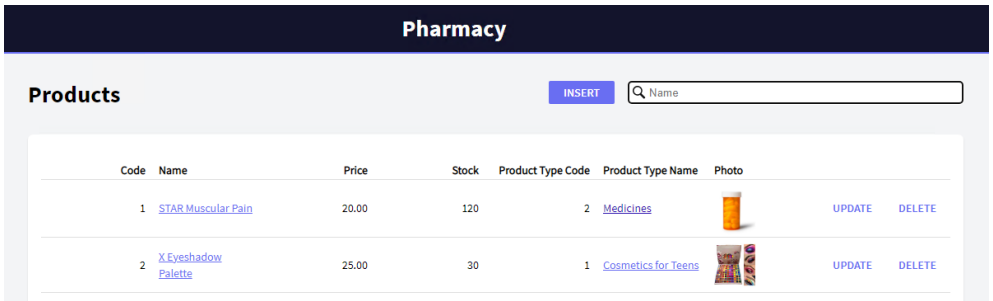
You can see the same query features that you already saw for the *Work With Product Types* page:



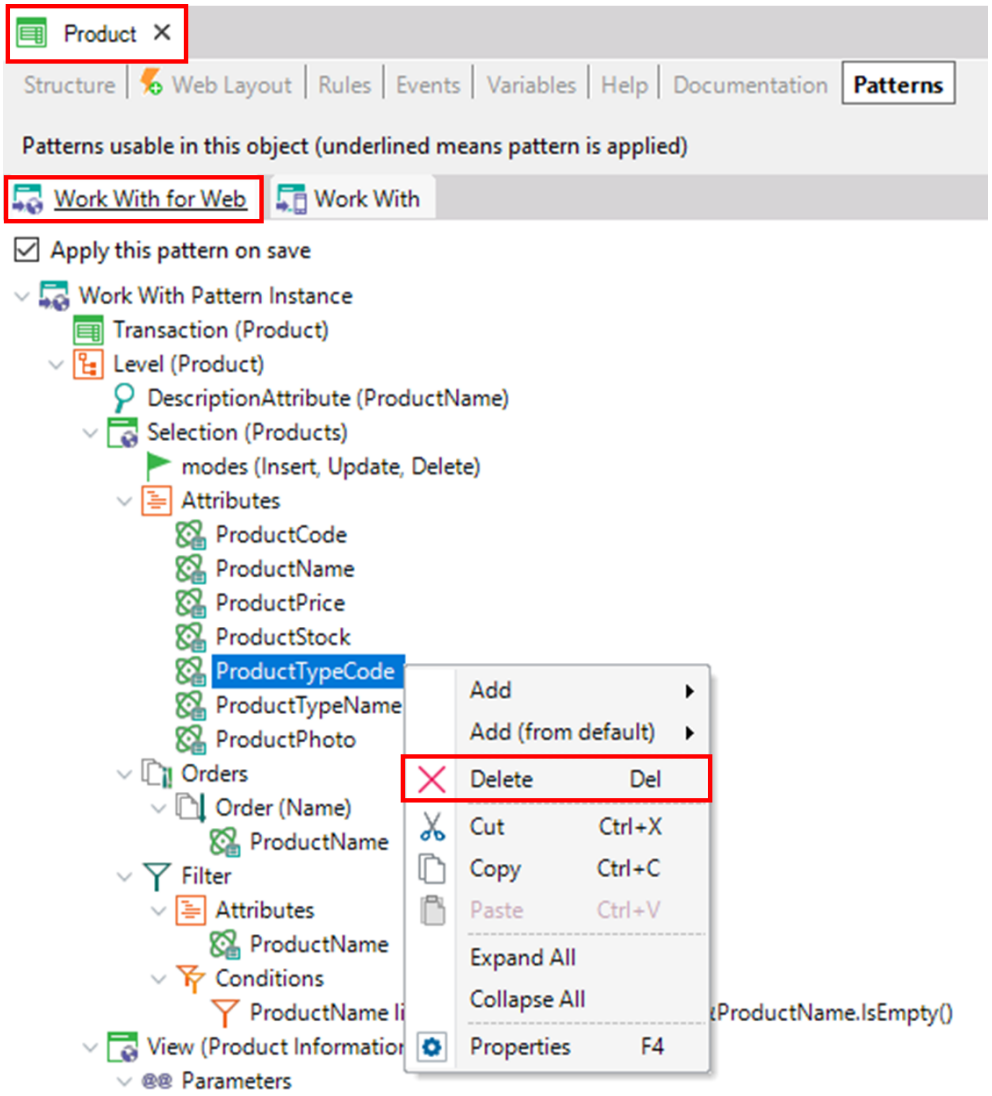
Let’s insert a new product:



After the confirmation, the application returns to the *Work With Product* page:



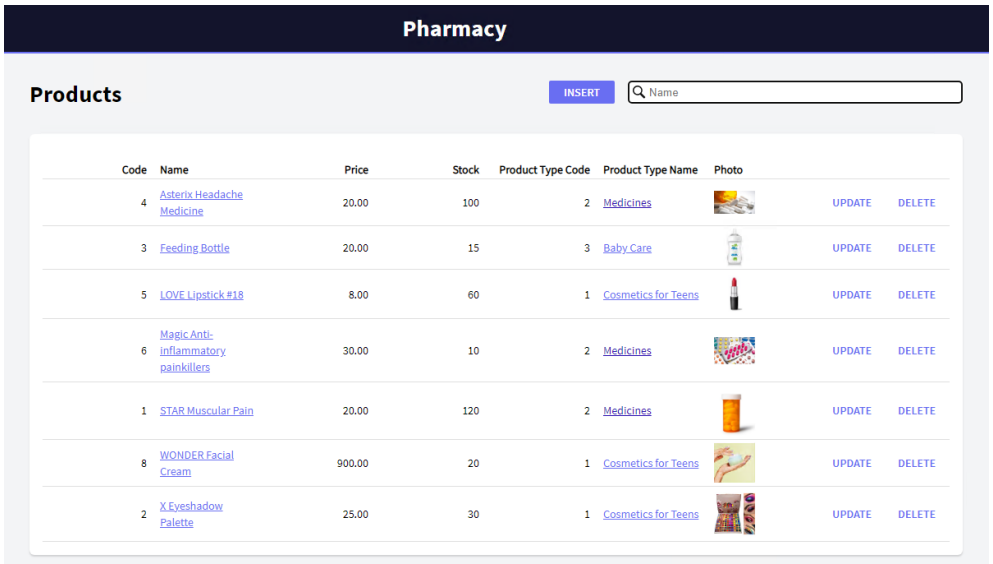
Since it's irrelevant to display the Product Type Code in the grid, let's remove it from the configurable tree which is taken into account to generate this *Work With for Web*:



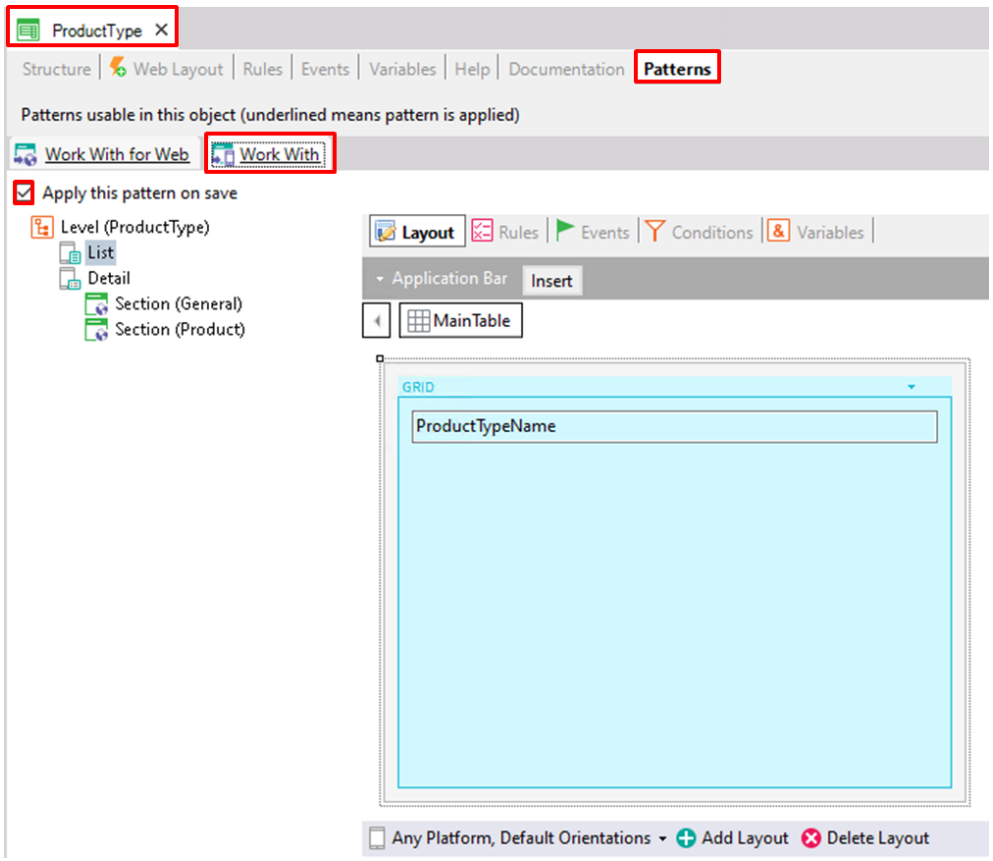
Press F5 and GeneXus proceeds to save, generating only the necessary programs and executing the application with the changes.

Now, insert some products in the same manner as shown before (by pressing the INSERT button that invokes the *Product* Transaction).

The Work With Products lists all the products that have been added:



Now, pay attention to the ***Work With*** tab offered for each Transaction. Let's apply it to the *ProductType* Transaction.

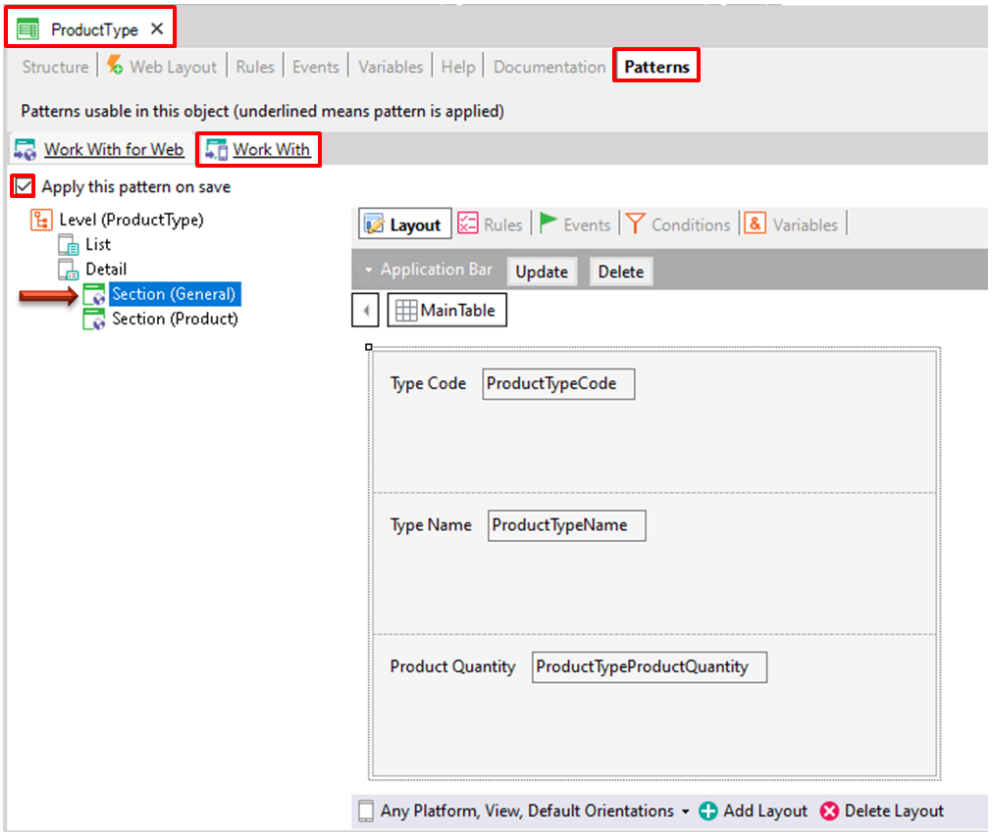


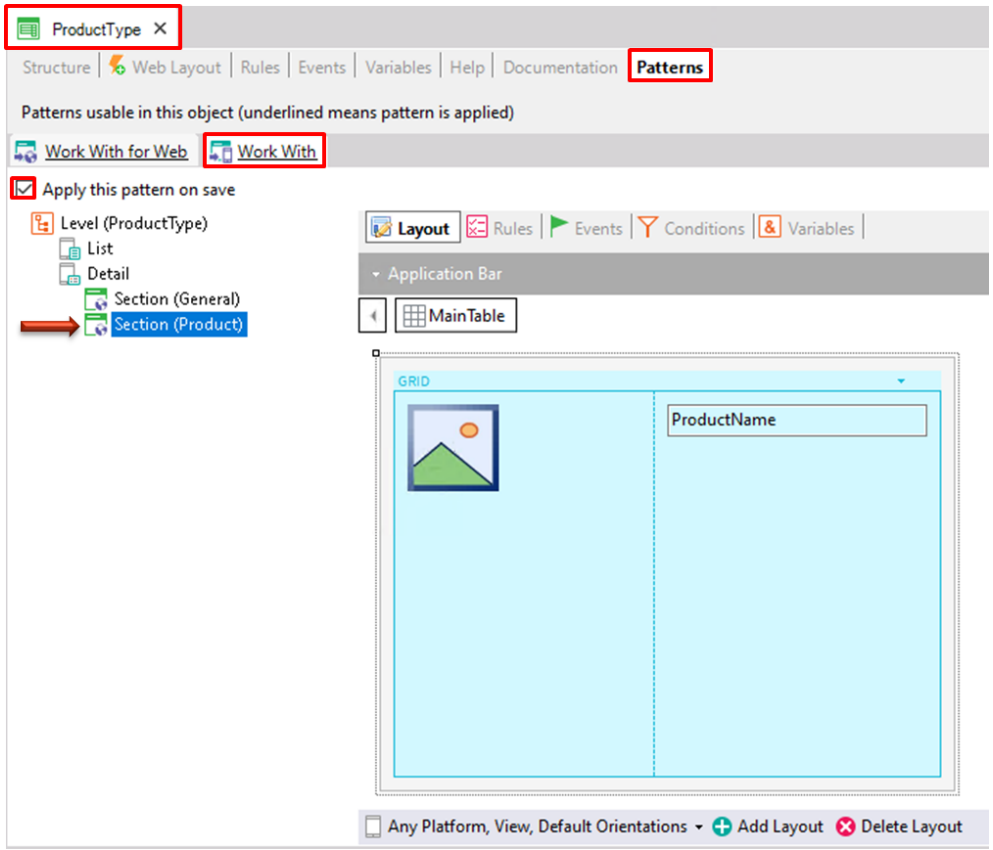
In contrast to the *Work With for Web* pattern, instead of seeing a list of attributes to be included in the grid under the node, the Layout is already shown in this case.

Now, look at the **Detail** node. You can associate the term Detail with seeing the details of a particular line in the list.

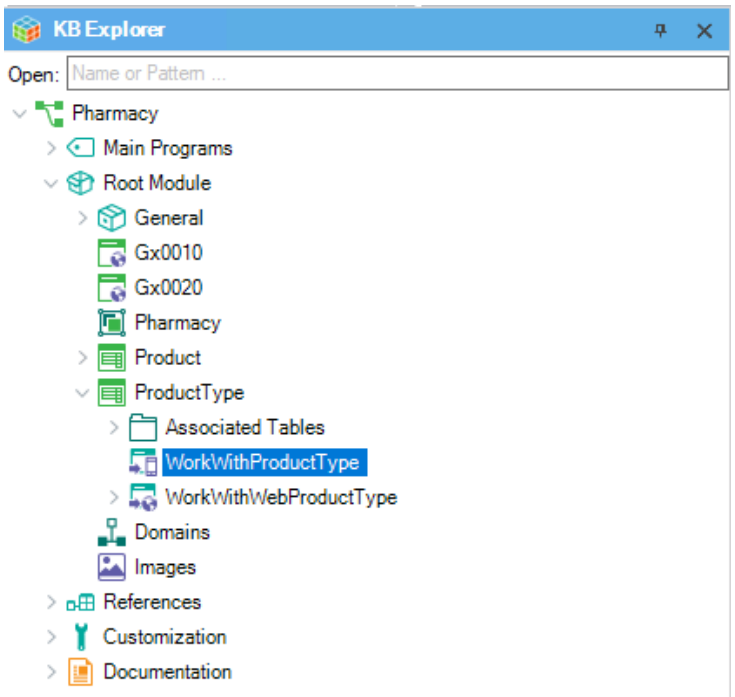
The **Detail** node is composed of two sections: **General** and **Product**.

Like the functionality implemented by the *Work With for Web* pattern, the **General** section displays the data associated with the selected product type and the **Product** section displays all the products that belong to the product type inside a grid.





After applying this pattern and saving, note that there is a new object called *WorkWithProductType* under the *ProductType* Transaction in the KB Explorer:



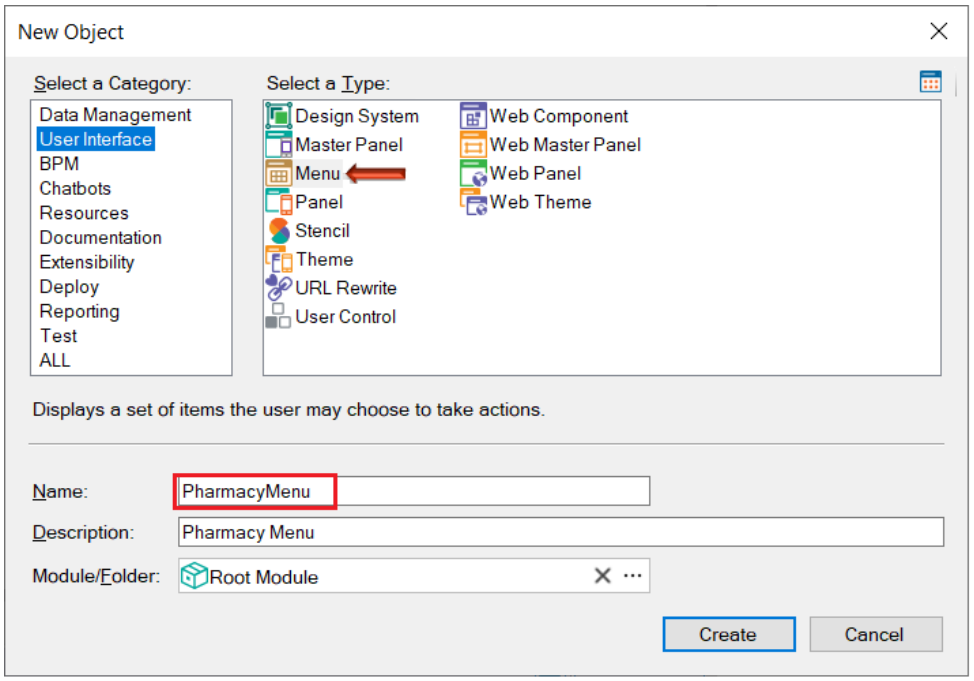
If you look at both objects generated under the *ProductType* Transaction, you can notice that the *WorkWithProductType* doesn't have other objects under it (because it includes different sections to define the entire implementation inside it).

On the other hand, the *WorkWithWebProductType*, is a configurable instance; so, you can set and save that instance object and GeneXus generates other objects under it to provide the useful behaviors you have already seen.

Soon, you will see the *WorkWithProductType* in action.

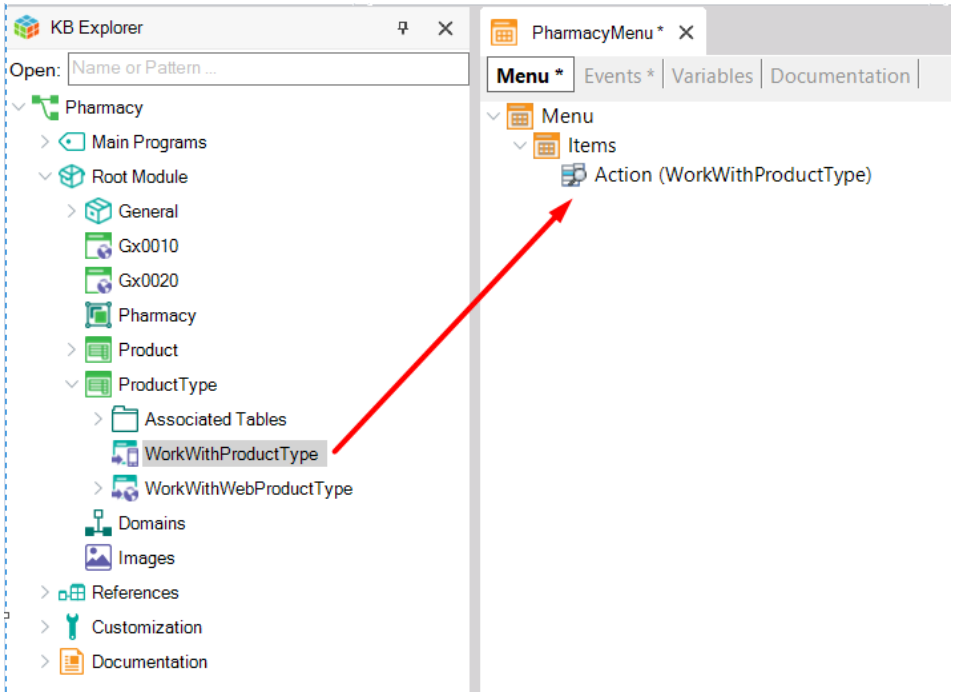
The proposal now is to create a **Menu** object, which will be the first one the application will execute. It will show an icon, so that when the user taps on it, the object *WorkWithProductType* is executed.

As explained before, to create an object you only have to select **File > New > Object**. The following window is then opened and you must choose the category User Interface, so that the objects that GeneXus offers which belong in this category are shown:

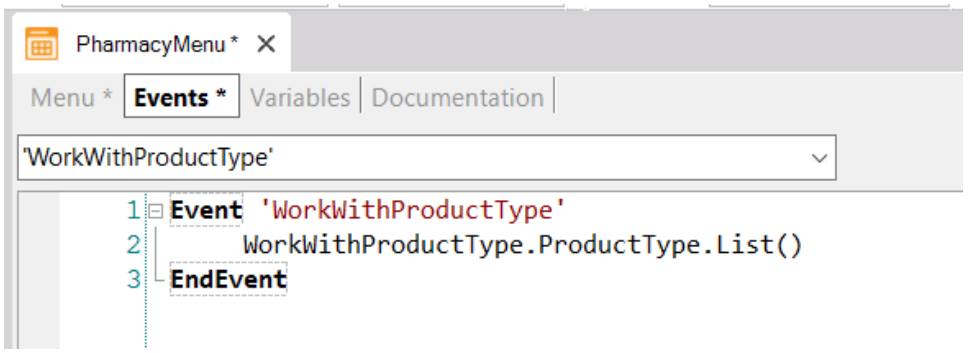


By default, every **Menu** is created with its property **Main program = True,** so that this object becomes an executable object (that is to say, it is compilable and executable on its own).

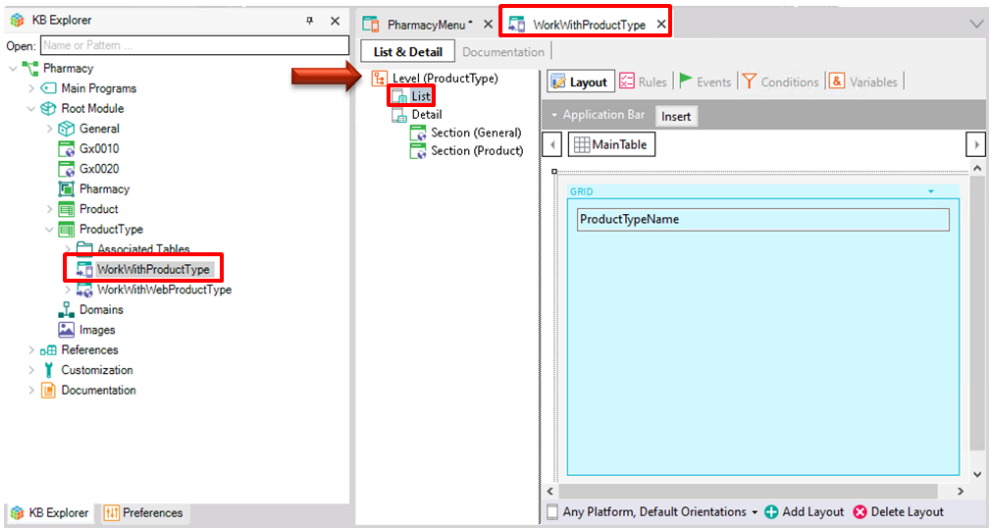
Now, drag and drop the WorkWithProductType object from the KB Explorer to the Items node:



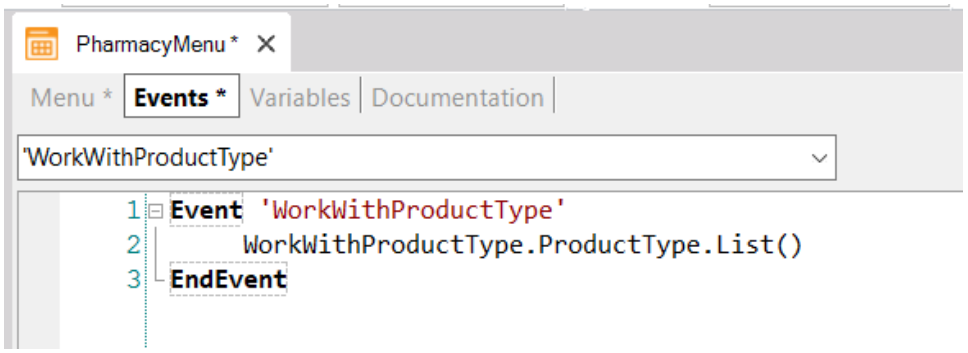
The shown Action is added and the following Event associated with the Action is automatically created:



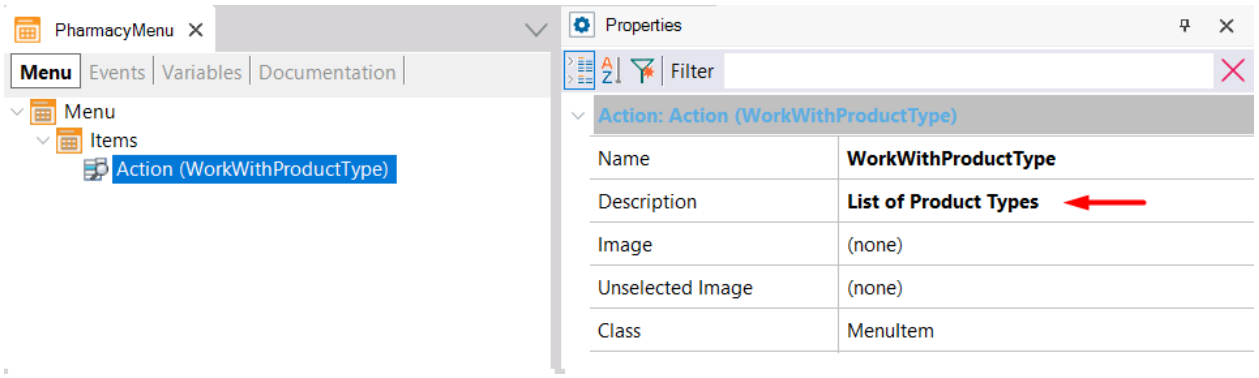
To understand the code line inside the Event, let's review the node tree contained inside the *WorkWithProductType* object:



The object has a main node, *ProductType,* and under it you can find the nodes List and Detail respectively, so to call the List node, the complete syntax is:

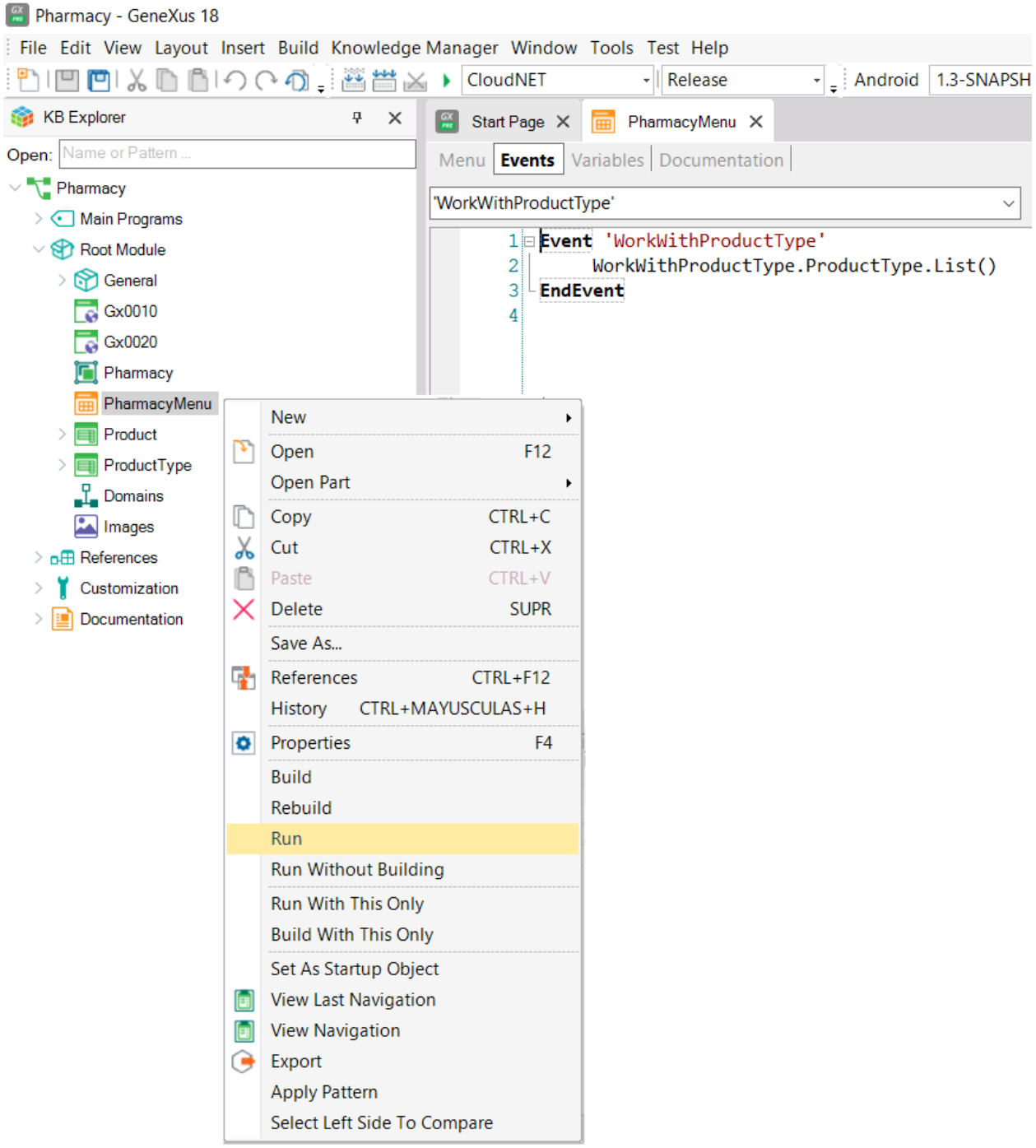


Change the Description property of the Action node in the Menu to "List of Product Types":



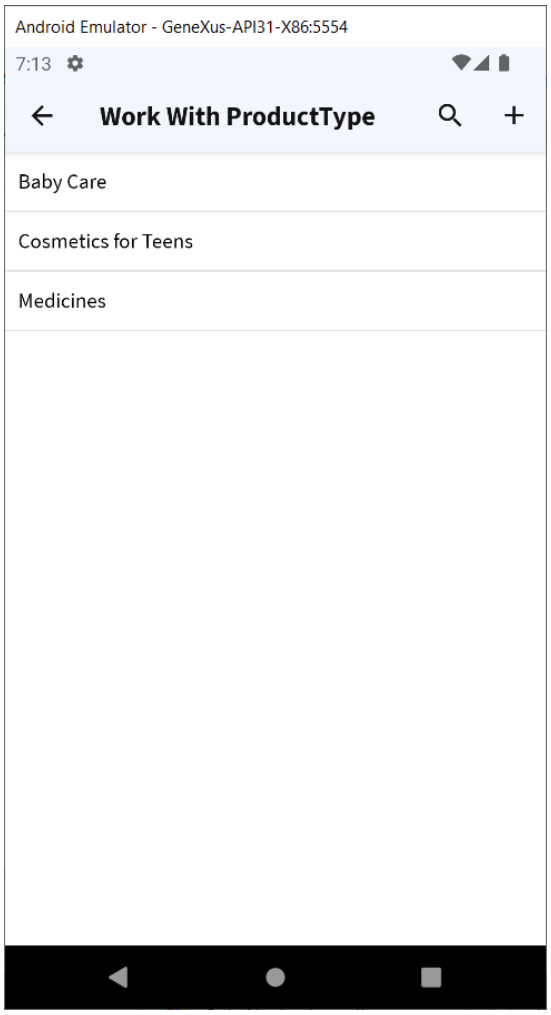
Now everything is defined and ready to run the Mobile application.

Since the *PharmacyMenu* object has its Main Program property set to True, you can execute it independently from the KB Explorer, by right-clicking the object and selecting **Run:**



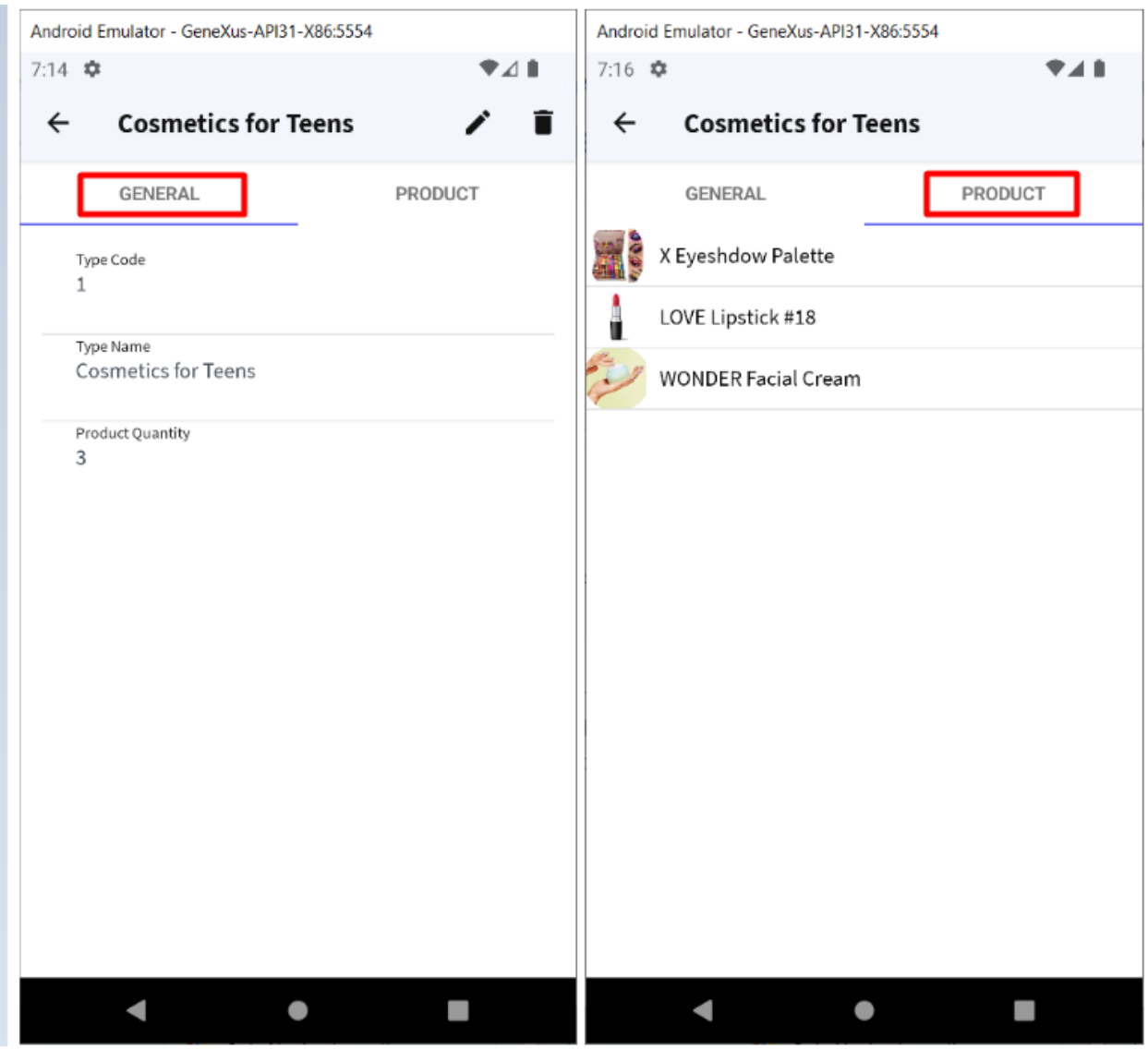
The execution may be performed either in a Mobile device connected to your computer or in an emulator which will open directly in the computer in which you are working:

When you tap the image, the list of product types the pharmacy offers is shown:



Note that the insert button on the top-right corner can be easily removed, as this application is for end-users, and they should only be able to view the different product types, neither edit them nor insert new products.

When you tap each product type (for example on “Cosmetics for teens”), the **Detail** is shown, along with its two sections GENERAL and PRODUCT:



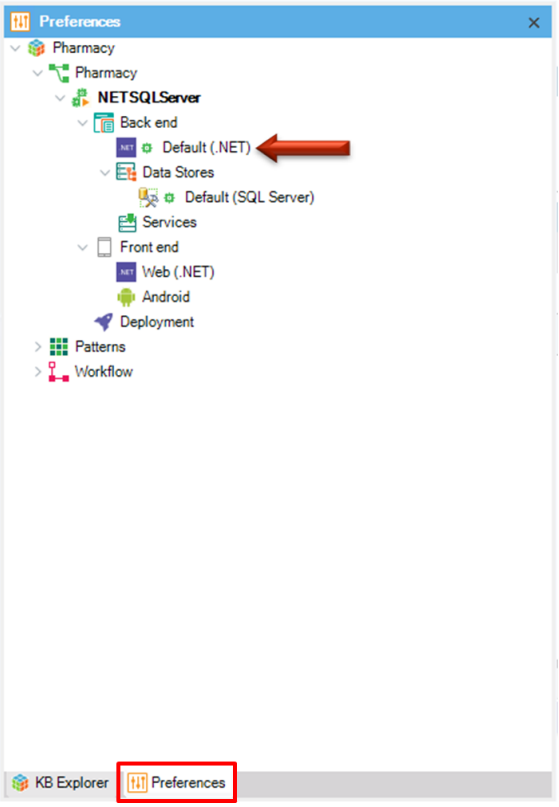
Obviously, this is just a very simple demonstration and you can achieve much more sophisticated applications.

GENERATING WHAT YOU HAVE DEFINED SO FAR IN ANOTHER LANGUAGE AND/OR FOR A DIFFERENT DATABASE

As it has been mentioned before, one of GeneXus' great advantages is that it allows you to generate the same application for different platforms, generating code in different programming languages and/or storing the application data in different databases. All this information is defined in an **Environment.**

An **Environment** allows you to configure and store all the information **related to a specific implementation of your application** (the generators that you want to use to generate the Back-end of your application, the generators that will be used to generate the Front-end, the information of the database, etc.).

Select the Preferences window by clicking on the tab next to the KB Explorer tab:

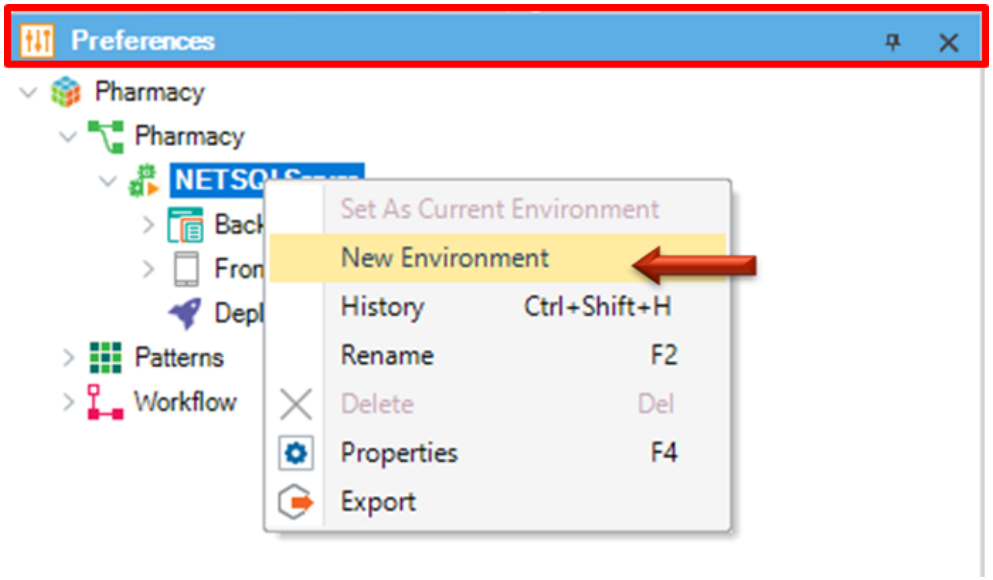


There is only one Environment defined **(.NET Environment)**. It was created automatically at Knowledge Base creation time when you chose .NET as the generation language. After that, it was completed when you pressed F5 for the first time and you entered the database name to be created and the database server. The sub nodes of the Environment have configurable properties.

It is possible to create more than one execution Environment for the same Knowledge Base. For example, it is common to create an Environment for development, where you connect to a database with test data and another Environment for production, where you define the server and database that you will use for the finished system.

You may also want to create, in the same Knowledge Base, a new Environment to generate everything for a different platform (for example, you may want to generate the code in Java and use another DBMS such as Oracle, MySQL, PostgreSQL, DB2 or others. For the generation of the Front end you could use Apple or Angular).

As the following image shows, to create a new Environment, you have to right-click on your node Environment, choose **New Environment**, and then choose the language, database, and configure the necessary properties.



To work with the new environment once it was created, you have to right-click on it and choose **Set As Current Environment**. It is possible to identify the active environment by the PLAY symbol.

[Learn more about how to create a new Environment.]

WHAT ELSE DOES GENEXUS OFFER?

- Accessing External Databases
    - You may need to access external databases from GeneXus applications. For example, in order to make an initial load, you may need to get data from an external database to the tables of the database associated with the Knowledge Base. After that, you may not need to stay connected to that external database, or, you may need to connect and always stay connected to a certain table or tables of one or more external databases (not just to read them, but also to access and change the data in them). GeneXus offers a "reverse engineering process" to connect to tables of external databases in order to achieve the needs described above.

- Collaborative Development Support
    - GeneXus Server offers the option to upload a Knowledge Base to a server. After that, new developers can create a local copy from the Knowledge Base in the server, from any geographical point, when the need arises. They can work (always locally) and upload their changes to the server. Of course, there is a mechanism for conflict resolution. This solution offers a lot of advantages, including version control of the models in the Knowledge Base.

- Consuming and defining web services
    - It is possible to consume web services developed by third parties from a GeneXus application, as well as to develop your own web services with GeneXus.

- Defining massive updates to the database and freely defining other types of processes.

- Defining interactive and personalized panels for both Web apps and Mobile apps.

- Designing & fine tuning UIs (User Interfaces)
    - As the user experience is extremely important, GeneXus offers the power to customize the user interface through specific generators for native apps, apps with web responsive design, web mobile, etc. It also offers a Cross-Platform Live Editing feature, which simplifies the process of applying Design to your application and Live Prototyping it.

- Deploying your app to production in Local Servers or Cloud Providers
    - You can deploy your app to production just by clicking on one button.

- Documenting within the Knowledge Base
    - GeneXus provides a Wiki-style Documentation editor, so that you can easily describe the Knowledge Base's purpose (in an object of the Documentation type - called Main - which every knowledge base has automatically created).
      Moreover, all GeneXus objects have a Documentation tab, where you can describe the object's purpose as well.
      When writing the documentation, you may include texts, images, links to attributes, objects, etc. Files can also be stored in the Knowledge Base as part of your documentation.

- Artificial Intelligence
    - GeneXus provides capabilities for easily integrate Artificial Intelligence (IA).

- Chatbot generator
    - GeneXus includes a Chatbot generator to automatically build and deploy a chatbot to any of the supported Chatbot providers.

- Extensibility
    - GeneXus allows the creation of specific extensions that let developers take advantage of different platform languages to create specific solutions and extend the capabilities of the GeneXus core.
        - https://training.genexus.com/en/learning/courses/genexus/v18/core

- Integrating external Systems and Data Sources into a GeneXus application
    - GeneXus ERP Connector for SAP makes the development of applications integrated to the SAP ERP possible, allowing you to complement the functionalities it offers.

- Managing Security
    - GeneXus offers a fully integrated security module called GeneXus Access Manager (GAM). By just enabling it, it solves authentication and authorization functionalities for both Web apps and Mobile apps.

- Modeling and automating business processes
    - GeneXus has a suite of tools that allow modeling and automation of business processes, as well as an execution environment to manage them. The modeling tool GeneXus Business Process Modeler is based on the BPMN 2.0 standard, and it is aimed at those users whose objective is to model business processes. These diagrams can be integrated or created in the GeneXus developing environment to implement the automation stage in which, by using GeneXus, we associate the different objects in each task modeled in the processes. GXflow offers the execution, management, and monitoring tools for the end users. In this way, GeneXus offers what we know as GeneXus BPM Suite, which is the set of tools which enable the development of systems based on Business Process Management; that is to say, business process-oriented systems.

- Reporting
    - Defining static reports (typical reports which can be printed, saved or just viewed on screen).
    - Defining visual and dynamic queries.
        - You can create queries to the database, group data according to one or several criteria, make calculations, and finally show the result in different types of graphs, Pivot tables, and tables. To carry out these kinds of queries, GeneXus offers the Query object and the Query Viewer control.
        - Moreover, the GXquery product allows end users to dynamically carry out queries, based on the same data model of the Knowledge Base. This tool focuses on enabling data access and analysis on the system's actual operational database, and gives the user an intuitive interface from which he can create his own queries and later see them through the web interface and the mobile application, or integrated into Microsoft Office Excel.

- Sharing Development and Marketplace
    - GeneXus Marketplace allows developers to share their User Controls, Extensions, Patterns, External Tools and External Objects created for and with GeneXus.

- Testing applications with GXtest
    - When new functionalities or variations are implemented, it's necessary to check that what already worked (before the changes) continues to behave properly. This kind of task can become very tedious if the application grows a lot, as the number of things to test will increase each time, etc. GeneXus helps to automatize these tests through its **GXtest** software, which allows you to save sequences of operations to test. Then, the tests are reproduced automatically, verifying that the system still works properly.

NEXT STEPS

You have come so far knowing GeneXus, so the natural question is "What's Next?"

- **First things first:**
  - Access the following online course in order to continue learning:
    - If you haven't tried GeneXus yet, you can do that for free, following this link: http://genexus.com/trial

- **Dig Deeper:** GeneXus is a very comprehensive development platform, and there is so much for you to read and learn. You could start digging deeper both in:
  - The GeneXus Training site: http://training.genexus.com/
  - The GeneXus Wiki: http://wiki.genexus.com/

- **Get GeneXus!** We live to provide the best tool that simplifies software development, so we thank every new client as the first one. Please, get in contact with us through info@genexus.com or check the http://genexus.com/plans to see which one fits you the best.

- **Be part of our Community:** Once you are ready, you can join our ever-growing community through a great array of possibilities
  - Publish your work in our marketplace: http://marketplace.genexus.com/
  - Jobs in GeneXus: http://genexus.com/company/work-with-us?en
  - Opportunities in our Partners: http://genexus.com/jobs/Opportunities?en
  - Be part of the GeneXus Alliance: http://genexus.com/partners
  - Come to the next GeneXus Meeting near you: http://genexus.com/meetings

We really hope we hear from you soon!!

The GeneXus Team

GeneXus™ by Globant
GeneXus™ by Globant

MONTEVIDEO - URUGUAY	Av. Italia 6201- Edif. Los Pinos, P1	(598) 2601 2082
CIUDAD DE MÉXICO - MÉXICO	Hegel N° 221, Piso 2, Polanco V Secc.	(52) 55 5255 4733
MIAMI - USA	8950 SW 74th Ct., Suite 1406	(1) 201-603-2022
SÃO PAULO - BRASIL	Rua Samuel Morse 120 Conj. 141	(55) 11 4858 0300
TOKYO - JAPAN	2-27-3, Nishi-Gotanda Shinagawa-ku, Tokyo, 141-0031	(81) 3 6303 9381 (81) 3 6303 9980