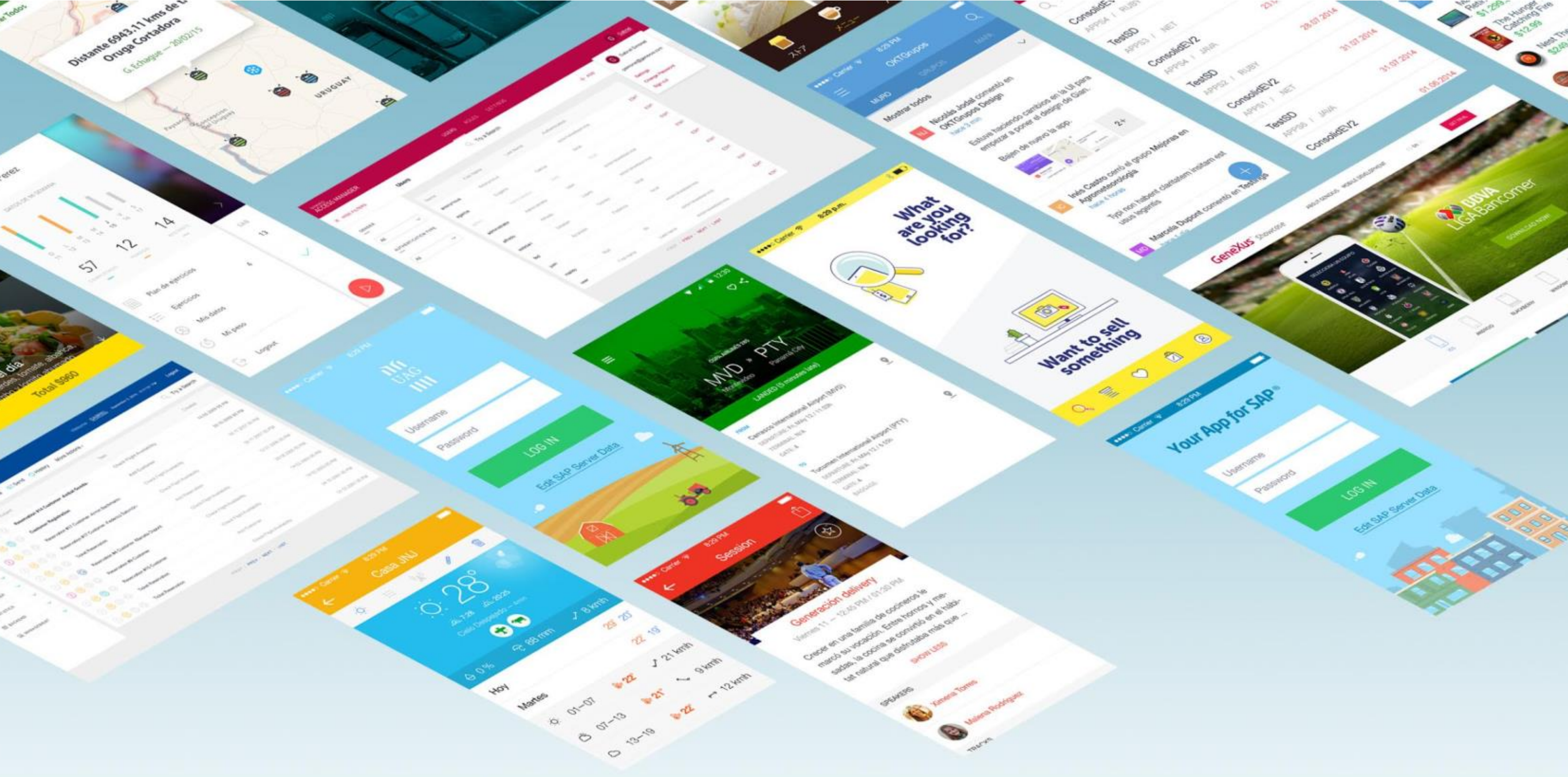


First Steps with GeneXus™

Create your first Application without knowing how to code.

TABLE OF CONTENTS

INTRODUCTION 3

GETTING STARTED WITH GENEXUS 3

CREATING A NEW GENEXUS APPLICATION 4

DEFINING THE FIRST OBJECTS 5

GENERATING AND RUNNING THE APPLICATION FOR THE FIRST TIME 18

THE APPLICATION GROWS UP 27

ADDING BUSINESS RULES 31

DEFINING CALCULATIONS: FORMULAS 33

USING PATTERNS (FOR WEB AND FOR MOBILE DEVICES) 36

WHAT IF YOU WANT TO GENERATE WHAT YOU HAVE DEFINED THIS FAR IN ANOTHER LANGUAGE AND/OR FOR A DIFFERENT DATA BASE? 59

WHAT ELSE DOES GENEXUS OFFER? 61

NEXT STEPS 64

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.

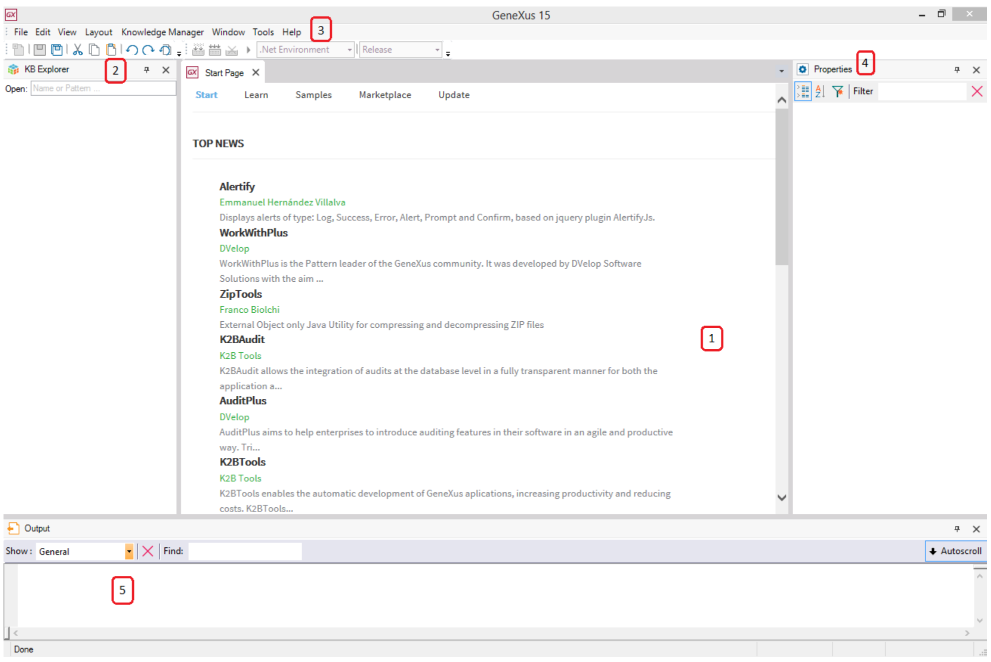
For example, GeneXus generates applications for the Web and/or Mobile and smart devices (from a watch, cell phone, tablet or TV), for the target platform selected by the developer (certain language, database, environment, platform, with web responsive design, etc.).

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

GETTING STARTED WITH GENEXUS

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

It consists of different windows:



TOP NEWS

Alertify
Emmanuel Hernández Villalva
Displays alerts of type: Log, Success, Error, Alert, Prompt and Confirm, based on jquery plugin AlertifyJs.

WorkWithPlus
DVelop
WorkWithPlus is the Pattern leader of the GeneXus community. It was developed by DVelop Software Solutions with the aim ...

ZipTools
Franco Biolchi
External Object only Java Utility for compressing and decompressing ZIP files

K2BAudit
K2B Tools
K2BAudit allows the integration of audits at the database level in a fully transparent manner for both the application a...

AuditPlus
DVelop
AuditPlus aims to help enterprises to introduce auditing features in their software in an agile and productive way. Tri...

K2BTools
K2B Tools
K2BTools enables the automatic development of GeneXus aplications, increasing productivity and reducing costs. K2BTools...

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.

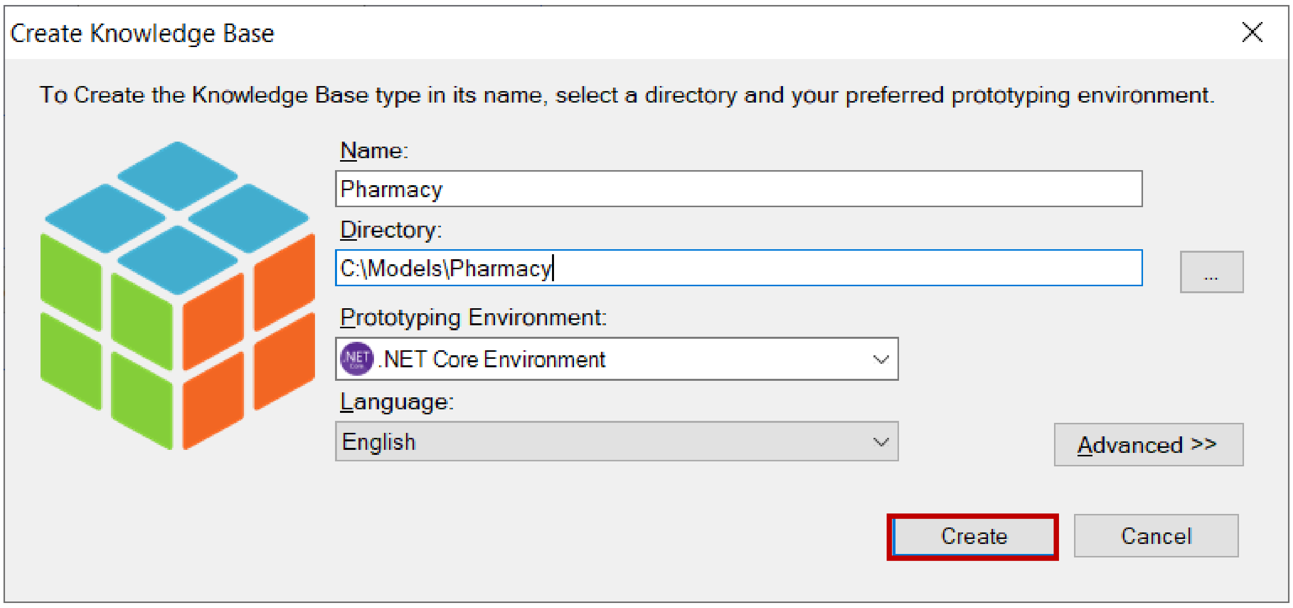
4. **Properties window:** Displays properties associated with the context in which the developer is positioned (like a selected object, attribute, variable, control, etc.).

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:


Create Knowledge Base

To Create the Knowledge Base type in its name, select a directory and your preferred prototyping environment.

Name: Pharmacy

Directory: C:\Models\Pharmacy

Prototyping Environment: .NET Core Environment

Language: English

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). Then, the path where you want to create the Knowledge Base must be entered.

The next step consists of selecting one of the programming languages available in the **Prototyping Environment** combo box. GeneXus will use the selected language to generate the application programs, as well as the necessary programs to create and maintain the database. The selected language in the image above is **.NET Core**. Further ahead, you will have to enter the database details.

The 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.

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

DEFINING THE FIRST OBJECTS

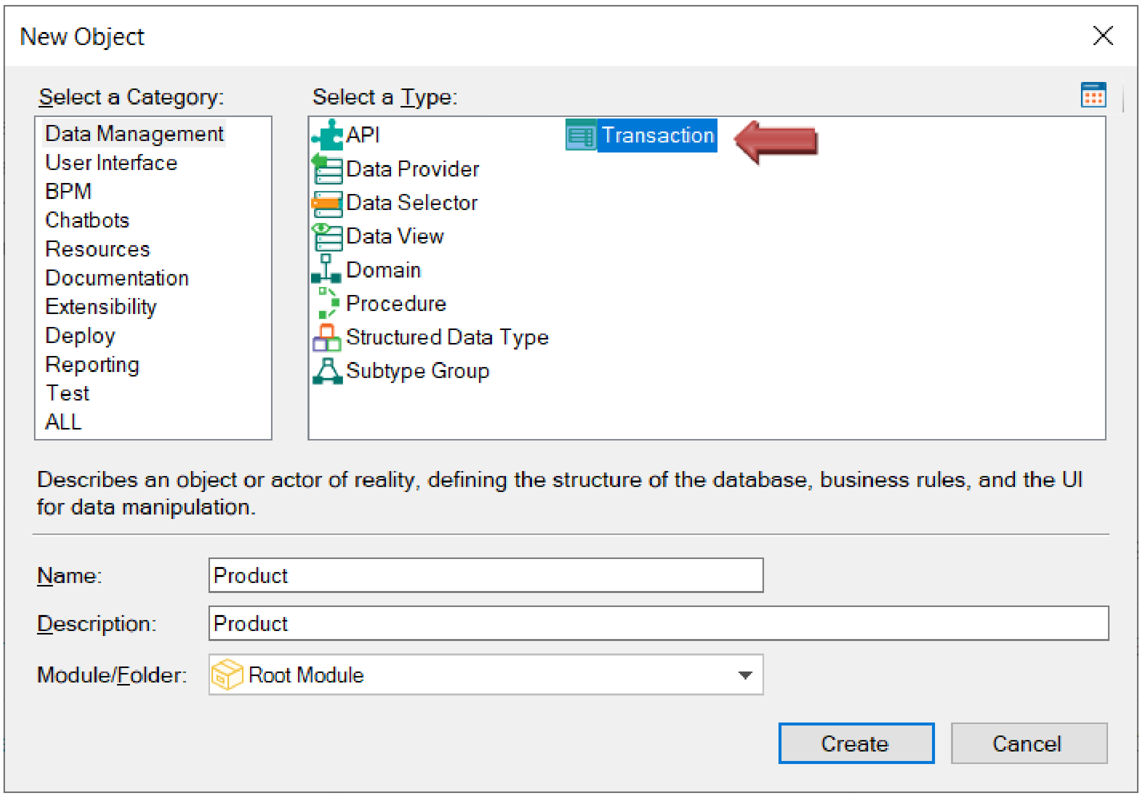
Once a new Knowledge Base is created, the next step is to describe the users' visions. In order to do so, it is necessary to identify real-life objects (we recommend paying attention to the nouns that users mention in their descriptions, such as: products, invoices, customers, etc.) and start defining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

GeneXus developers don't work on low-level tasks such as defining tables, normalizing, designing programs, programming, and the like. Instead, their work is a higher-level activity that implies describing the reality of users. 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 in order 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*



New Object

Select a Category:

- Data Management
- User Interface
- BPM
- Chatbots
- Resources
- Documentation
- Extensibility
- Deploy
- Reporting
- Test
- ALL

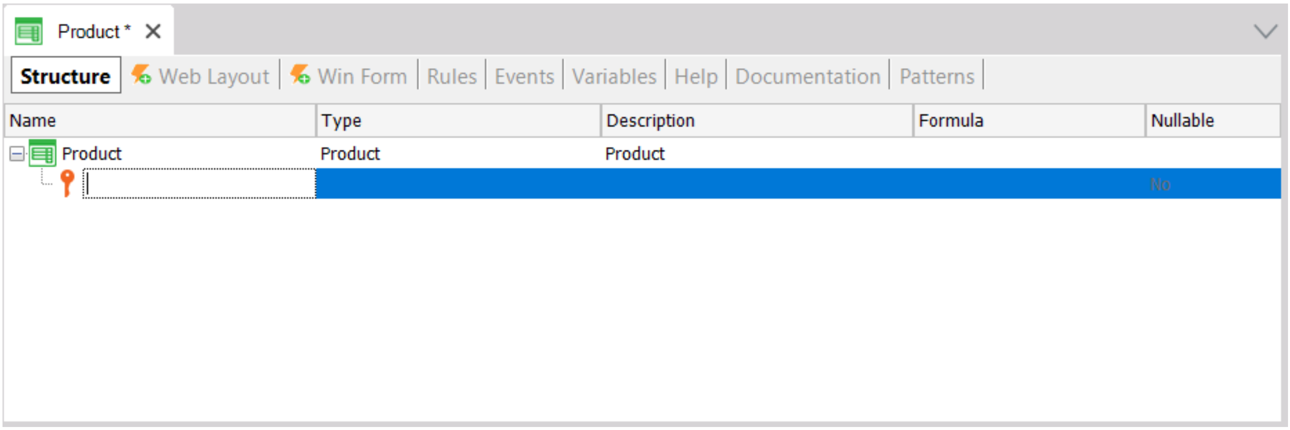
Select a Type:

- API
- Data Provider
- Data Selector
- Data View
- Domain
- Procedure
- Structured Data Type
- Subtype Group
- Transaction

Describes an object or actor of reality, defining the structure of the database, business rules, and the UI for data manipulation.

Name:	Product
Description:	Product
Module/Folder:	Root Module

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



Name	Type	Description	Formula	Nullable
Product	Product	Product		
				No

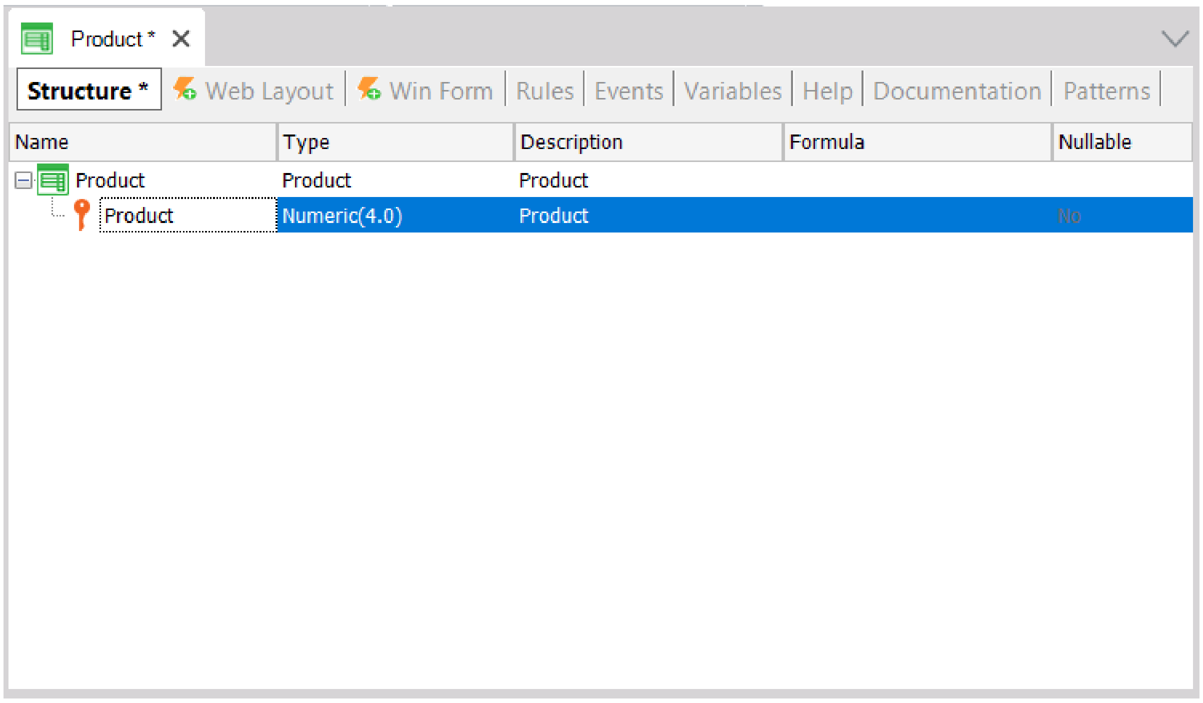
Each Transaction has some sections that will be explained gradually. Specifically, the Transaction structure enables the definition of the attributes or fields that describe a real-life object.

Suppose that at the pharmacy you were told that they need to keep 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 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.

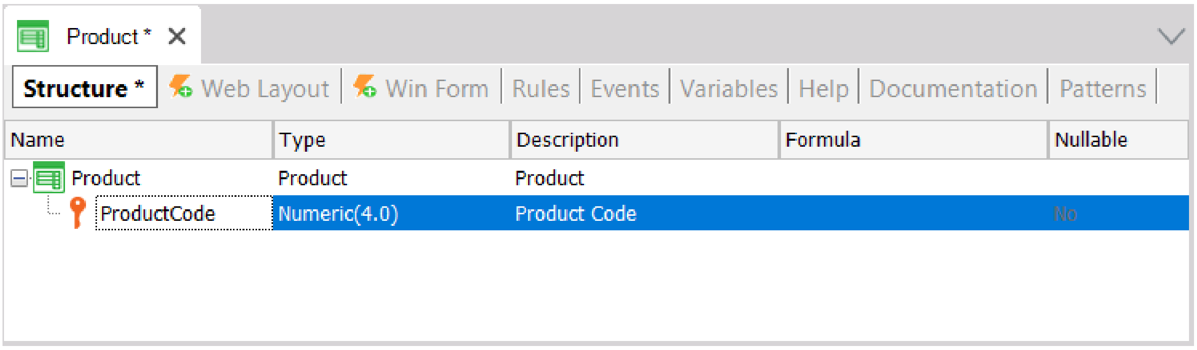
By pressing the dot key on the keyboard, GeneXus will automatically show the Transaction name as prefix in the attribute name:


Product * ×

Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
Product	Product	Product		
Product	Numeric(4.0)	Product		No

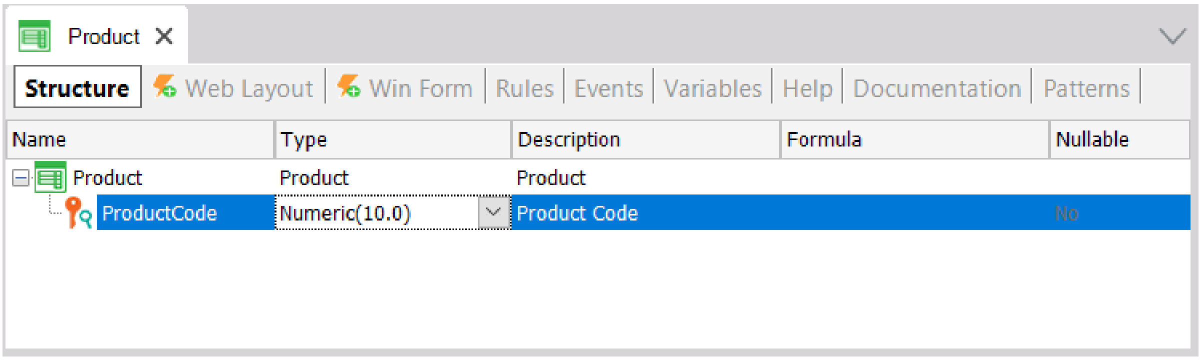
and you only have to type *Code* after the *Product* prefix:


Product * ×

Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

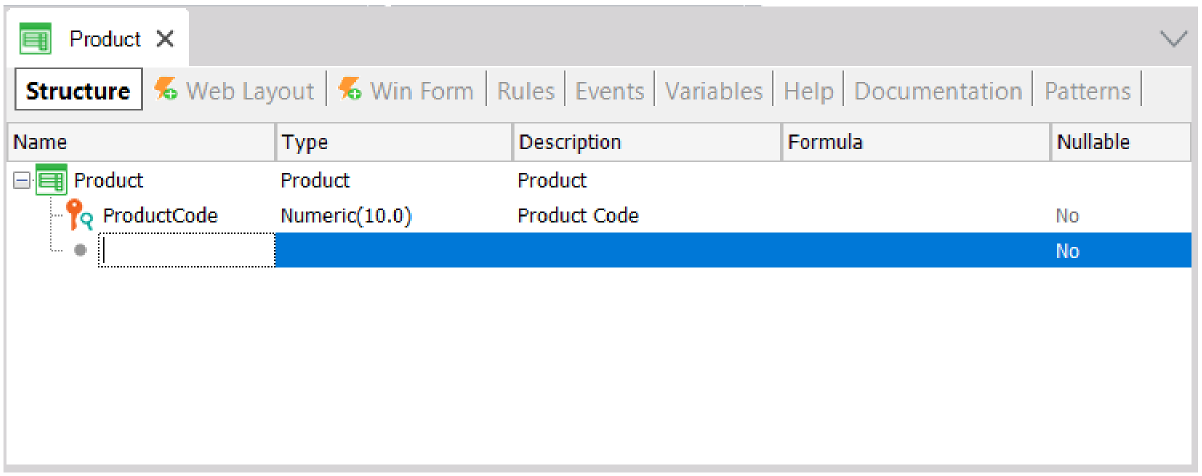
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(4.0)	Product Code		No

Then, by pressing the Tab key, you can choose the data type to be stored 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:



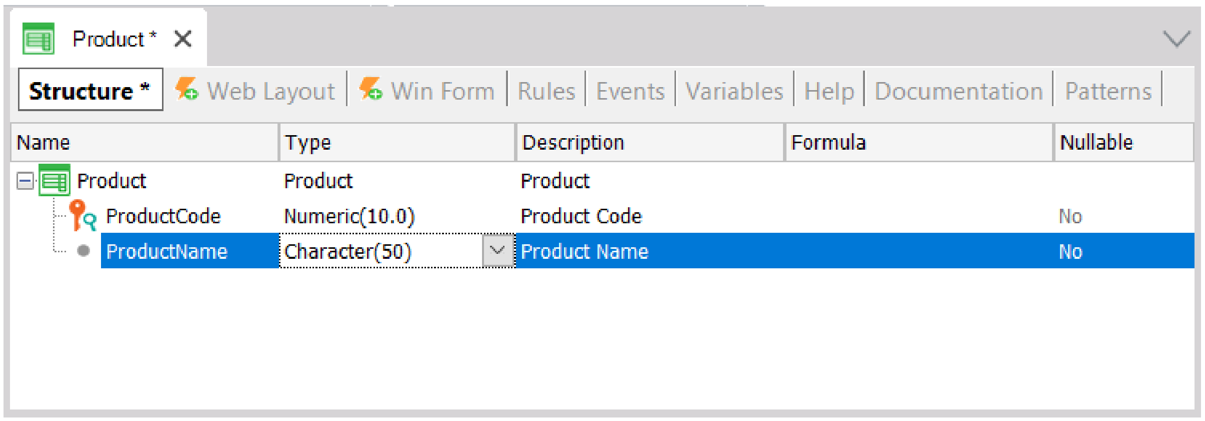
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No

By pressing Enter, a new line is opened where you can start defining the second attribute:



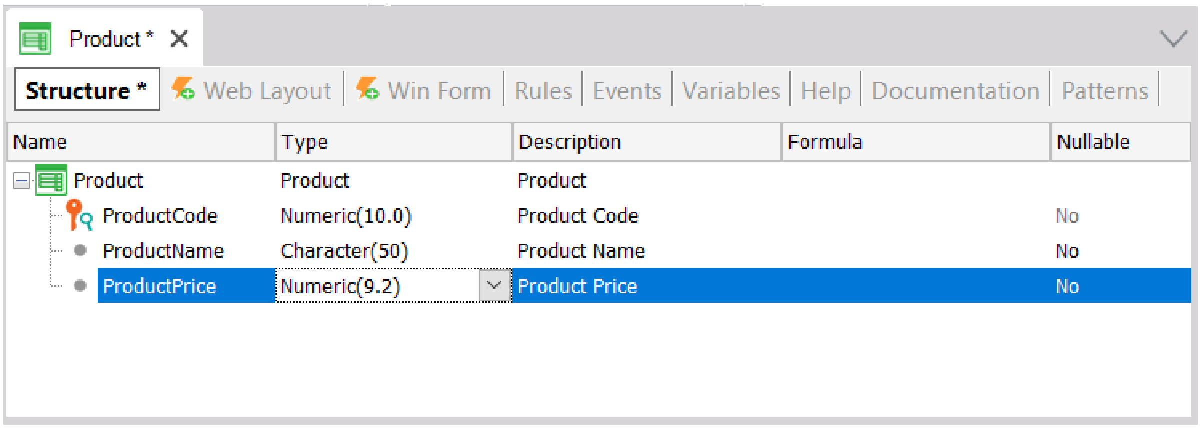
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
				No

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):



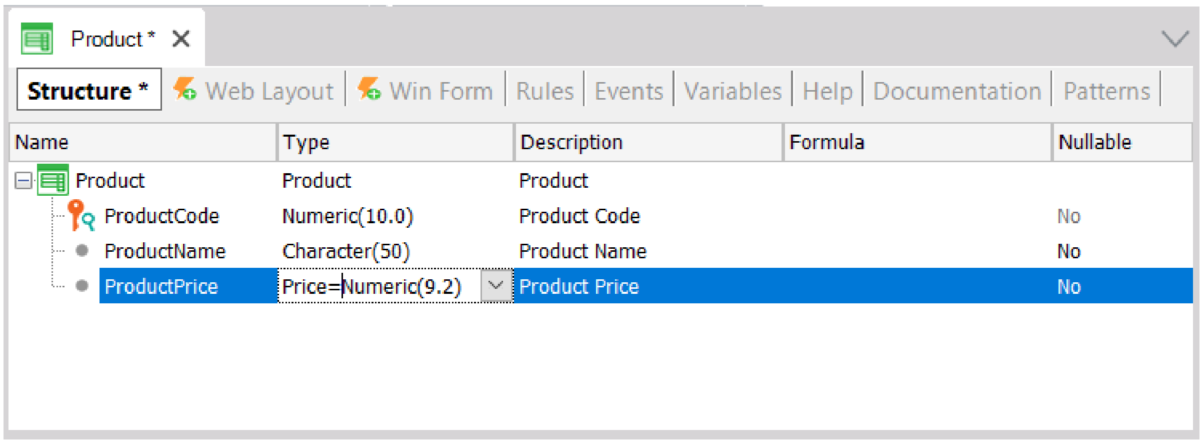
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No

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


Product *
Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

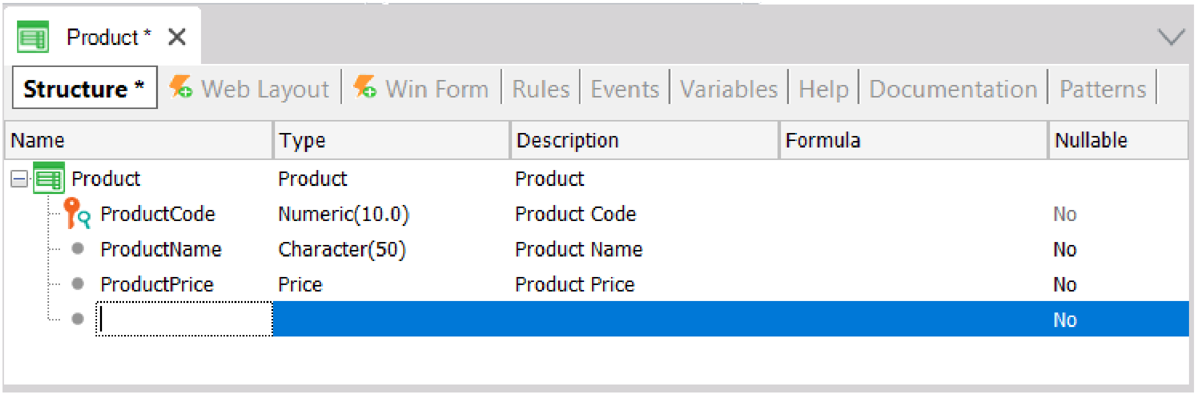
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Numeric(9.2)	Product Price		No

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, in the Type column, you just have to write: "Price=", before the type you have recently selected:


Product *
Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price=Numeric(9.2)	Product Price		No

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

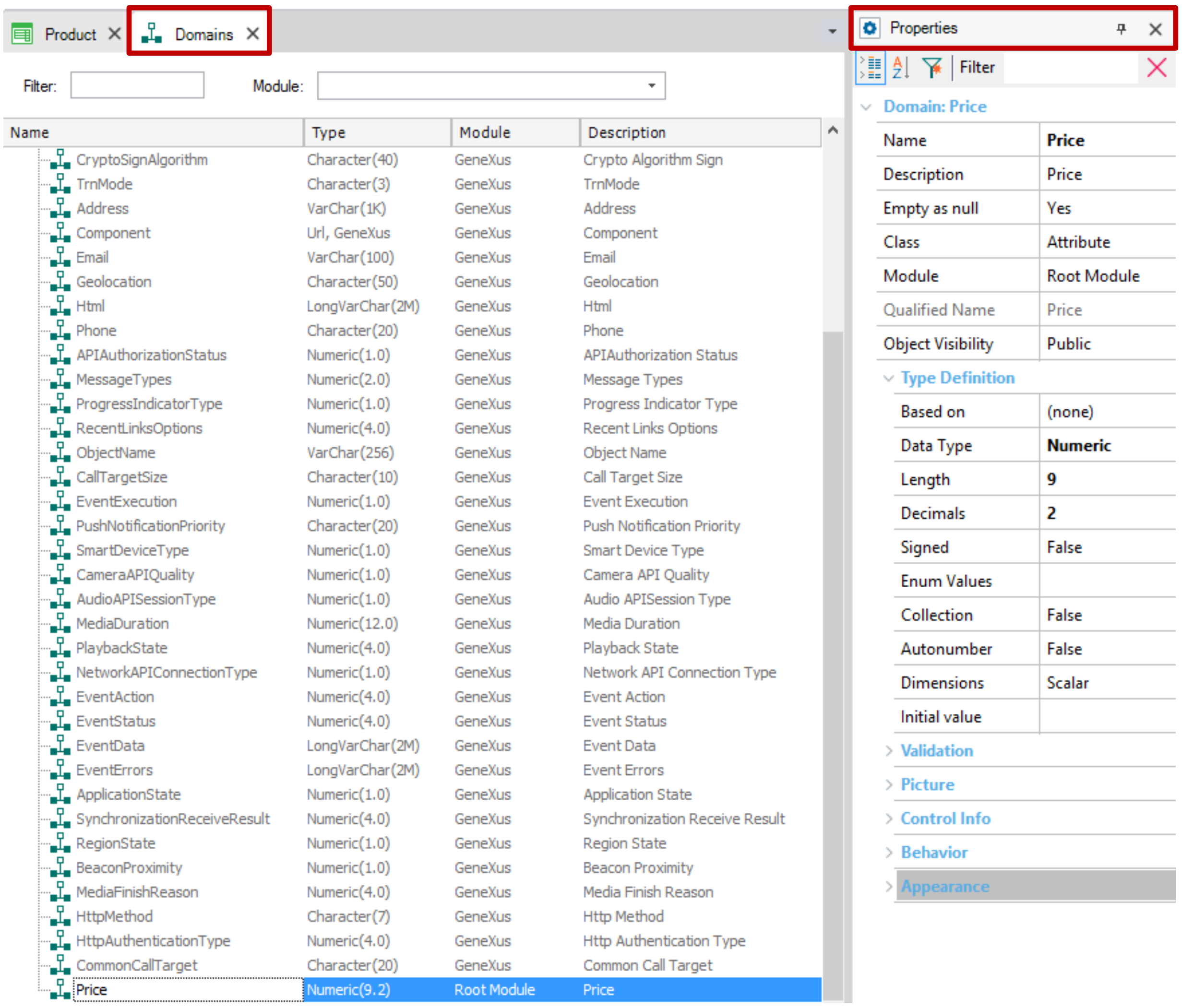

Product *
Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
				No

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 may view the created domains in the Knowledge Base, by selecting View > Domains in the Toolbar:


Product | Domains

Filter: Module:

Name	Type	Module	Description
CryptoSignAlgorithm	Character(40)	GeneXus	Crypto Algorithm Sign
TrnMode	Character(3)	GeneXus	TrnMode
Address	VarChar(1K)	GeneXus	Address
Component	Url, GeneXus	GeneXus	Component
Email	VarChar(100)	GeneXus	Email
Geolocation	Character(50)	GeneXus	Geolocation
Html	LongVarChar(2M)	GeneXus	Html
Phone	Character(20)	GeneXus	Phone
APIAuthorizationStatus	Numeric(1.0)	GeneXus	APIAuthorization Status
MessageTypes	Numeric(2.0)	GeneXus	Message Types
ProgressIndicatorType	Numeric(1.0)	GeneXus	Progress Indicator Type
RecentLinksOptions	Numeric(4.0)	GeneXus	Recent Links Options
ObjectName	VarChar(256)	GeneXus	Object Name
CallTargetSize	Character(10)	GeneXus	Call Target Size
EventExecution	Numeric(1.0)	GeneXus	Event Execution
PushNotificationPriority	Character(20)	GeneXus	Push Notification Priority
SmartDeviceType	Numeric(1.0)	GeneXus	Smart Device Type
CameraAPIQuality	Numeric(1.0)	GeneXus	Camera API Quality
AudioAPISessionType	Numeric(1.0)	GeneXus	Audio APISession Type
MediaDuration	Numeric(12.0)	GeneXus	Media Duration
PlaybackState	Numeric(4.0)	GeneXus	Playback State
NetworkAPIConnectionType	Numeric(1.0)	GeneXus	Network API Connection Type
EventAction	Numeric(4.0)	GeneXus	Event Action
EventStatus	Numeric(4.0)	GeneXus	Event Status
EventData	LongVarChar(2M)	GeneXus	Event Data
EventErrors	LongVarChar(2M)	GeneXus	Event Errors
ApplicationState	Numeric(1.0)	GeneXus	Application State
SynchronizationReceiveResult	Numeric(4.0)	GeneXus	Synchronization Receive Result
RegionState	Numeric(1.0)	GeneXus	Region State
BeaconProximity	Numeric(1.0)	GeneXus	Beacon Proximity
MediaFinishReason	Numeric(4.0)	GeneXus	Media Finish Reason
HttpMethod	Character(7)	GeneXus	Http Method
HttpAuthenticationType	Numeric(4.0)	GeneXus	Http Authentication Type
CommonCallTarget	Character(20)	GeneXus	Common Call Target
Price	Numeric(9.2)	Root Module	Price

Properties

Filter

Domain: Price

Name	**Price**
Description	Price
Empty as null	Yes
Class	Attribute
Module	Root Module
Qualified Name	Price
Object Visibility	Public
Type Definition	
Based on	(none)
Data Type	**Numeric**
Length	**9**
Decimals	**2**
Signed	False
Enum Values	
Collection	False
Autonumber	False
Dimensions	Scalar
Initial value	

Validation

Picture

Control Info

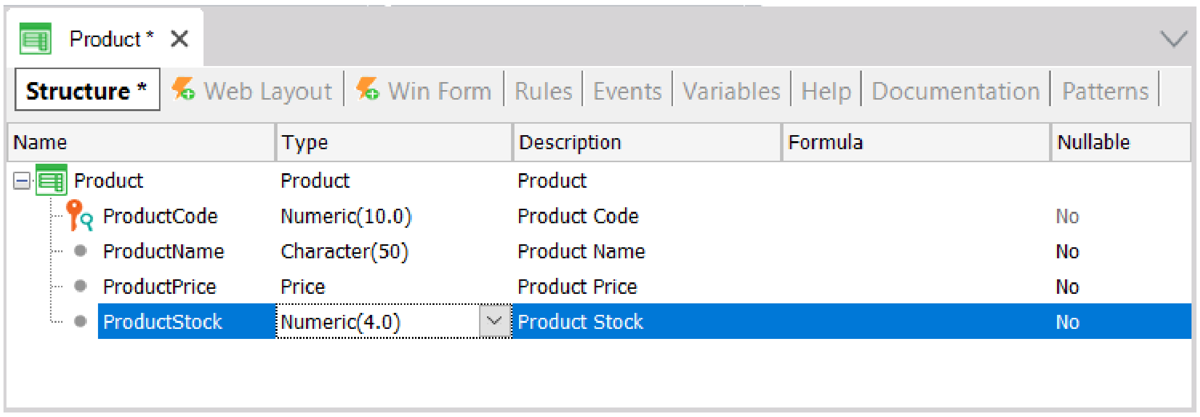
Behavior

Appearance

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.

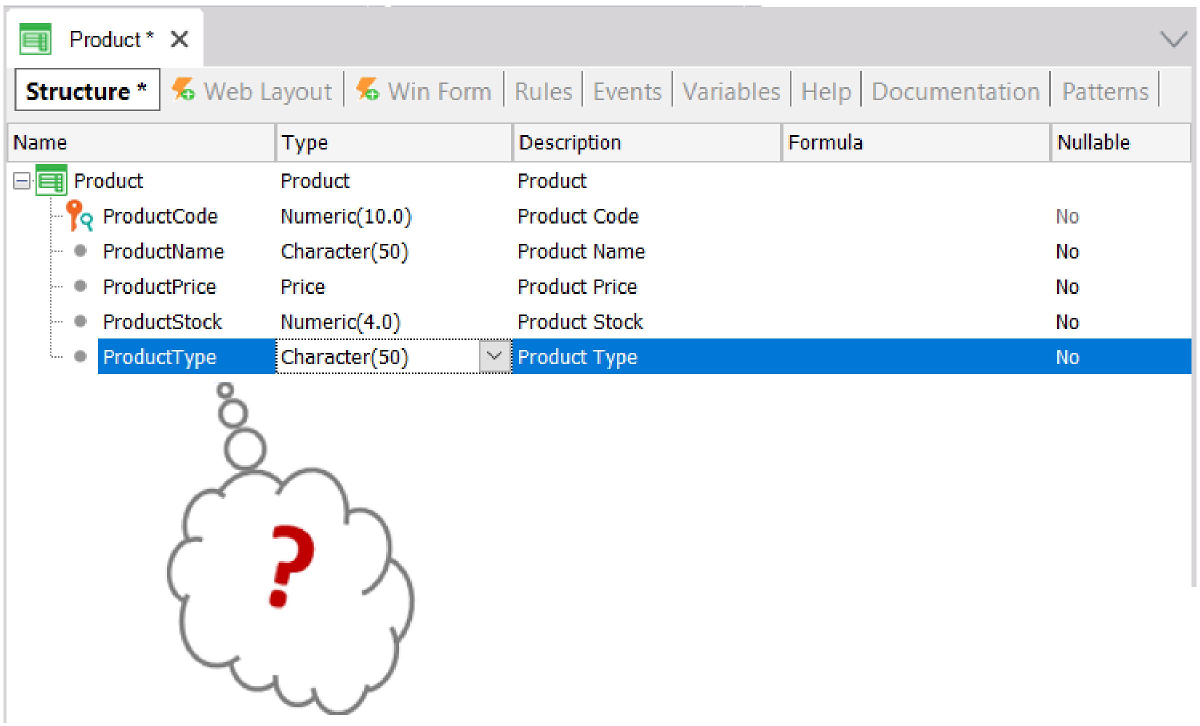
The Properties window will be refreshed every time you select another attribute, domain, object, etc., with the corresponding properties available for configuration in each case. If it is not visible, you can press F4 to open it.

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:



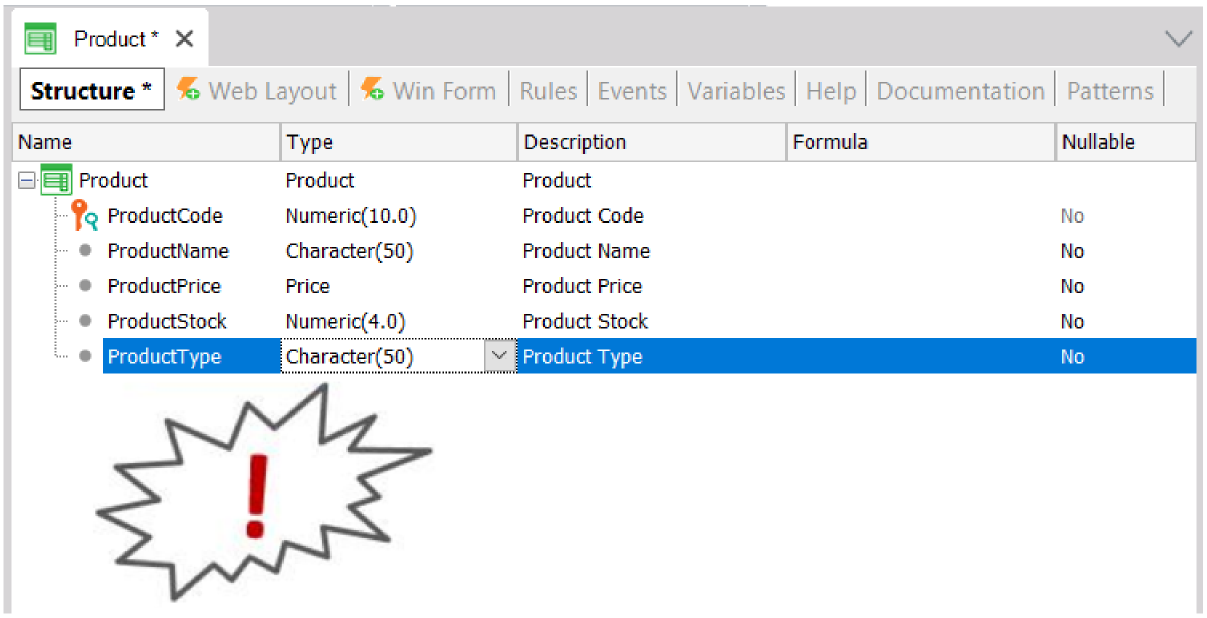
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No

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



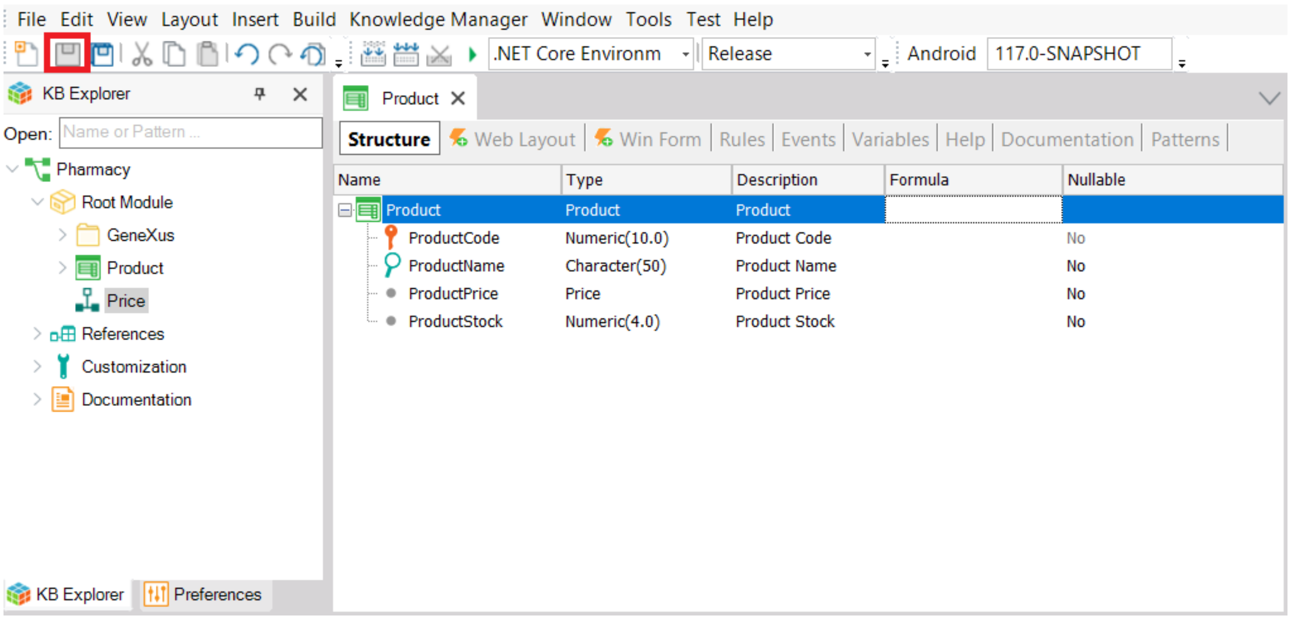
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No
ProductType	Character(50)	Product Type		No

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 type must have been typed exactly the same.


Product * ✕
Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns
!

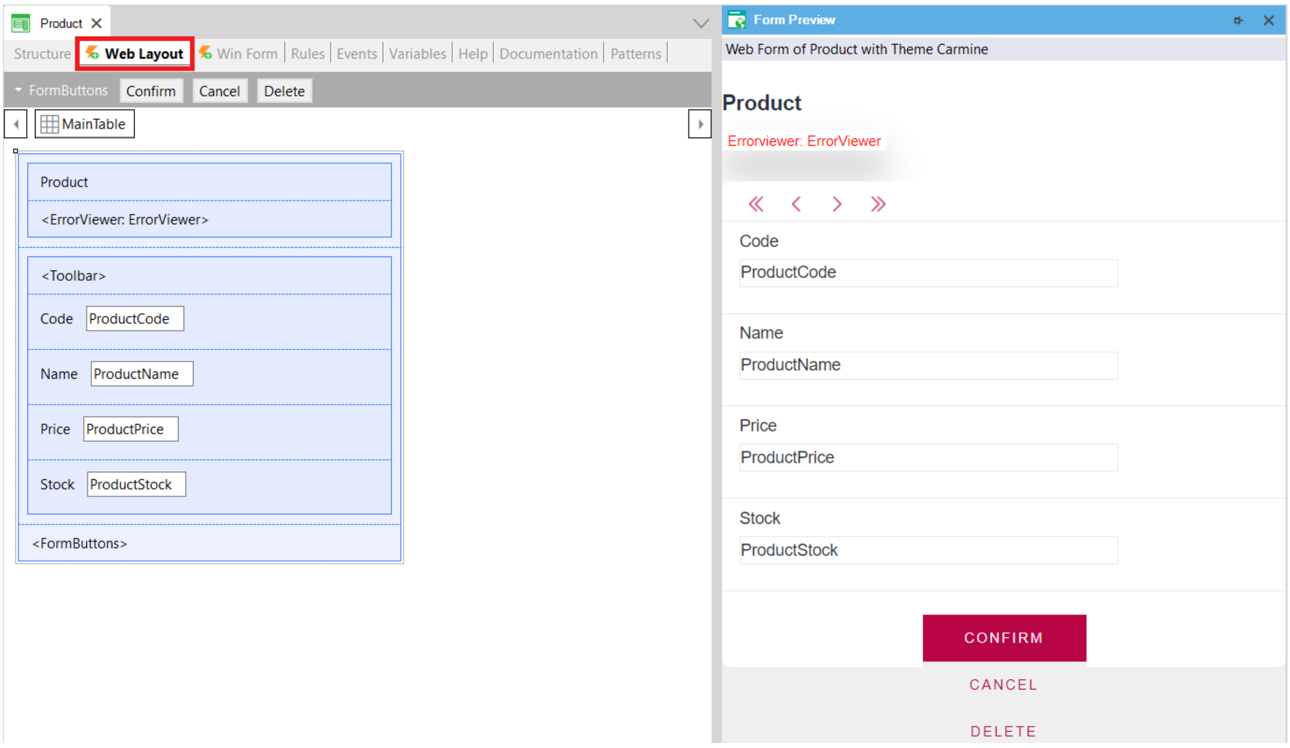
Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No
ProductType	Character(50)	Product Type		No

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:

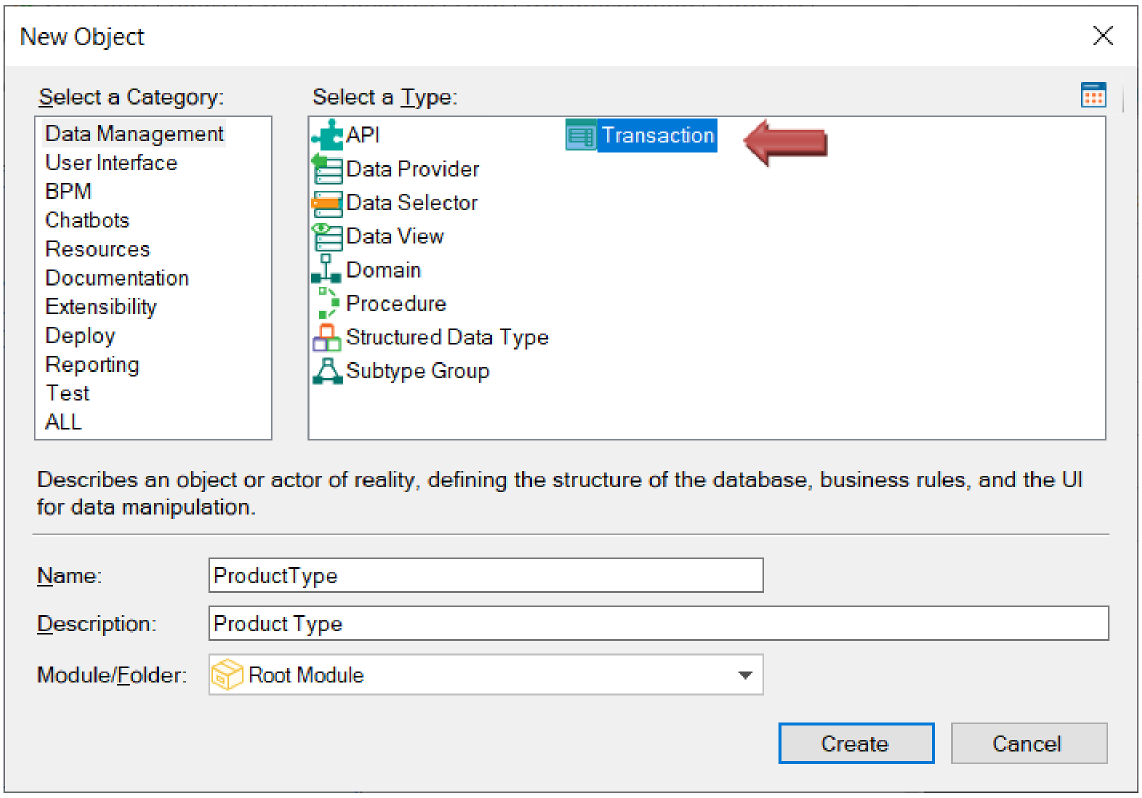

File Edit View Layout Insert Build Knowledge Manager Window Tools Test Help
.NET Core Environm
Release
Android
117.0-SNAPSHOT
KB Explorer
Open: Name or Pattern ...
Pharmacy
Root Module
GeneXus
Product
Price
References
Customization
Documentation
KB Explorer
Preferences
Product ✕
Structure | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No

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:

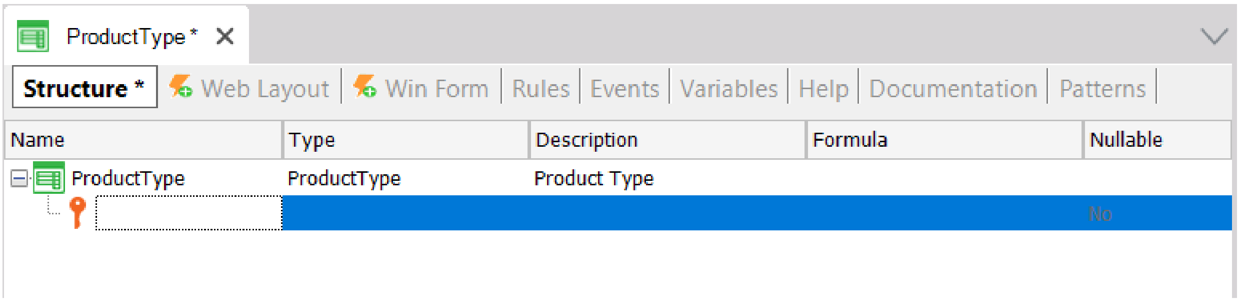


Now let's create another Transaction to record the product types, and after that, let's assign a product type to each product:



Describes an object or actor of reality, defining the structure of the database, business rules, and the UI for data manipulation.

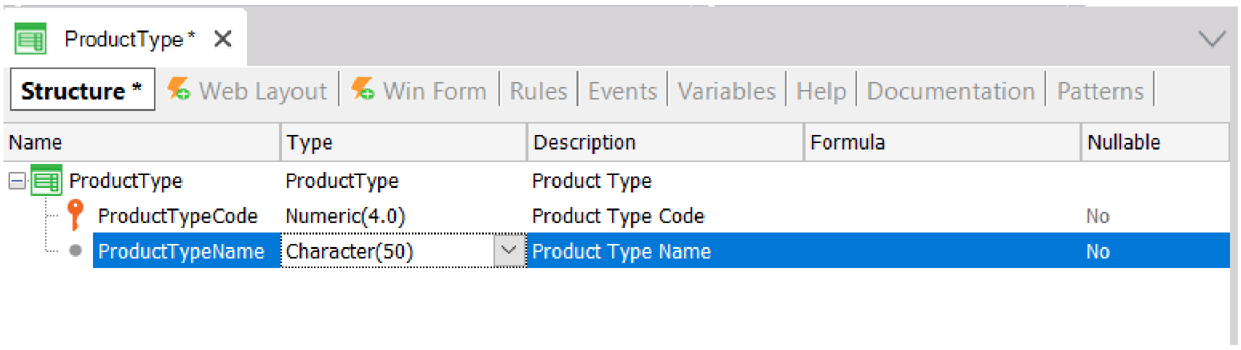
Name:	ProductType
Description:	Product Type
Module/Folder:	Root Module



ProductType * | Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
ProductType	ProductType	Product Type		
				No

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



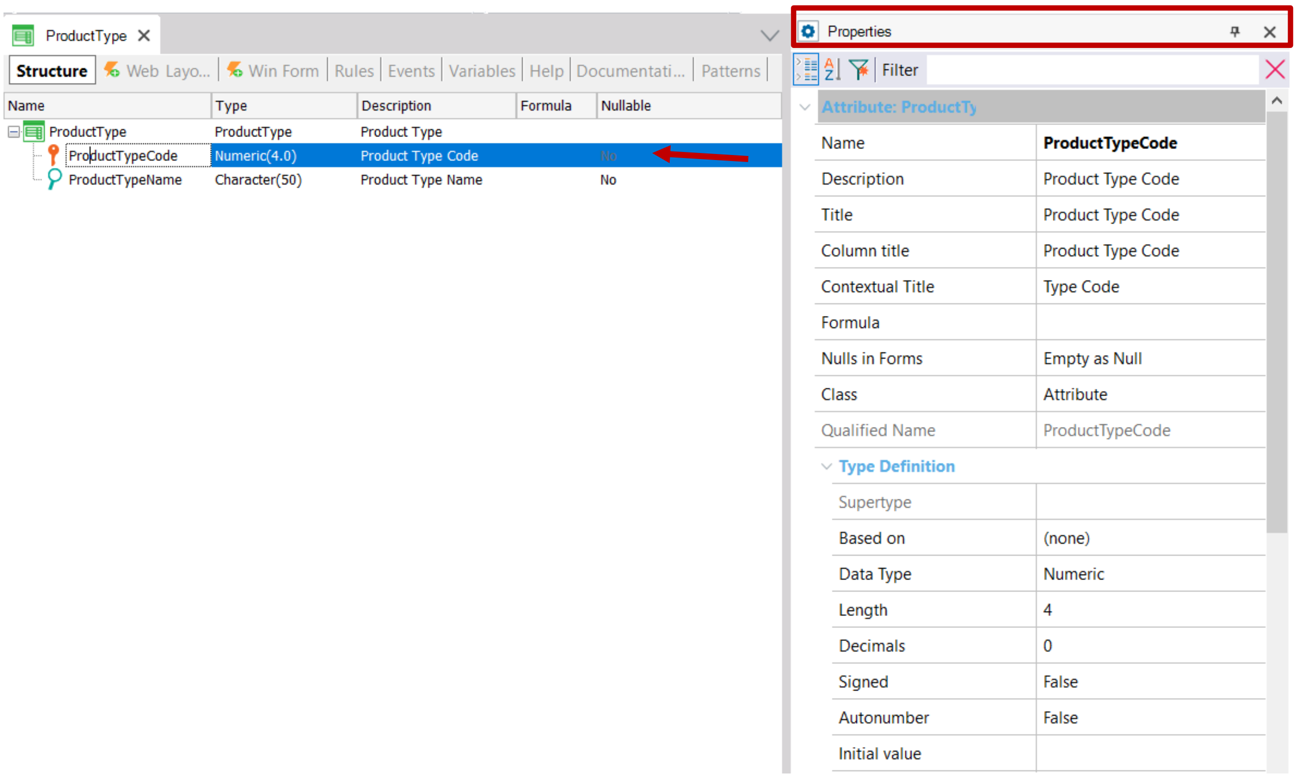
ProductType * | Structure * | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
ProductType	ProductType	Product Type		
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	Product Type Name		No

Remember our 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 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:

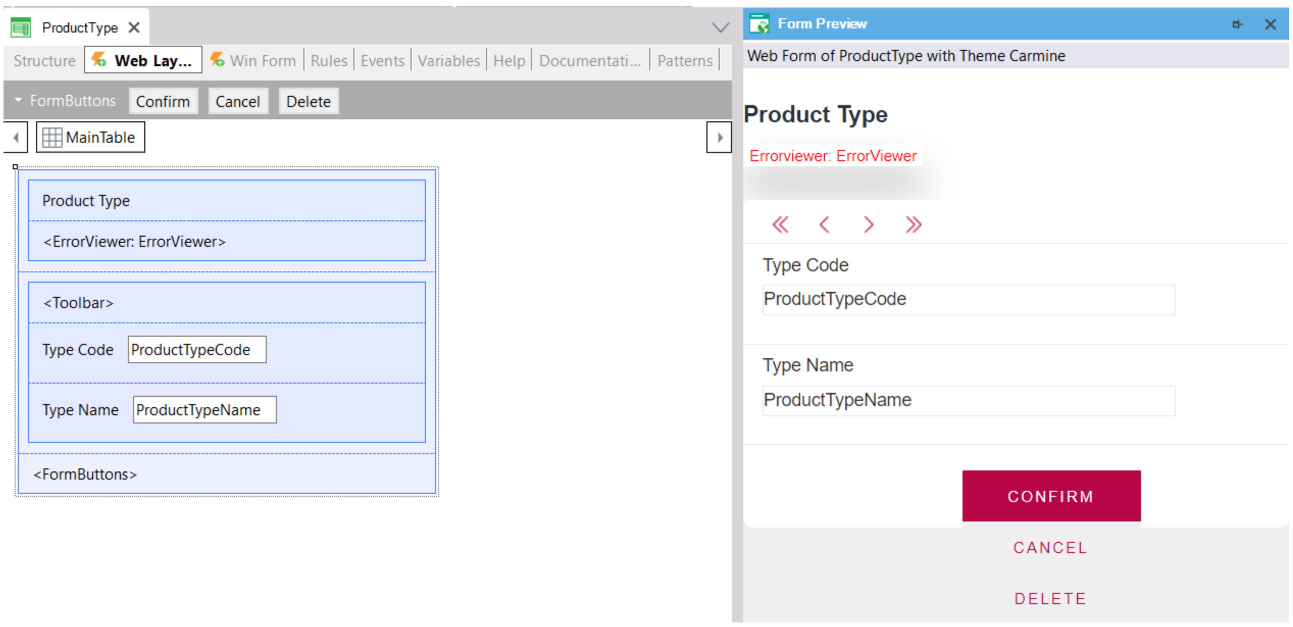


Name	Type	Description	Formula	Nullable
ProductType	ProductType	Product Type		
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	Product Type Name		No

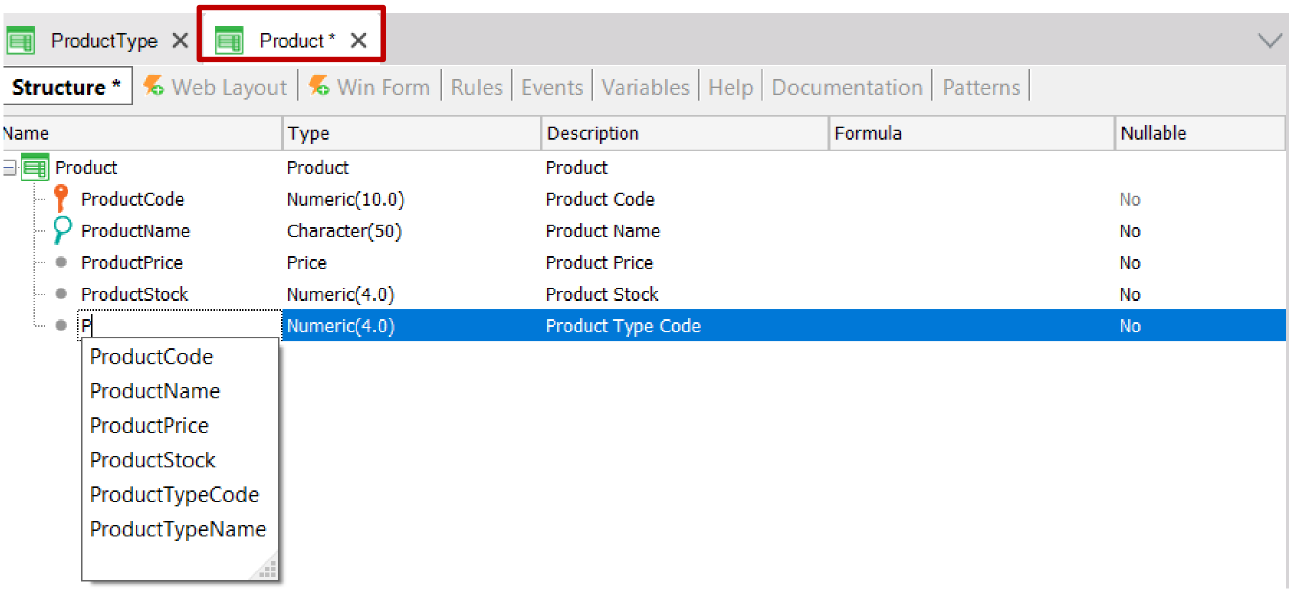
Name	**ProductTypeCode**
Description	Product Type Code
Title	Product Type Code
Column title	Product Type Code
Contextual Title	Type Code
Formula	
Nulls in Forms	Empty as Null
Class	Attribute
Qualified Name	ProductTypeCode
Type Definition	
Supertype	
Based on	(none)
Data Type	Numeric
Length	4
Decimals	0
Signed	False
Autonumber	False
Initial value	

Look at the Autonumber property. It 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, let's set the Autonumber property to True for this identifier attribute and let's 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:



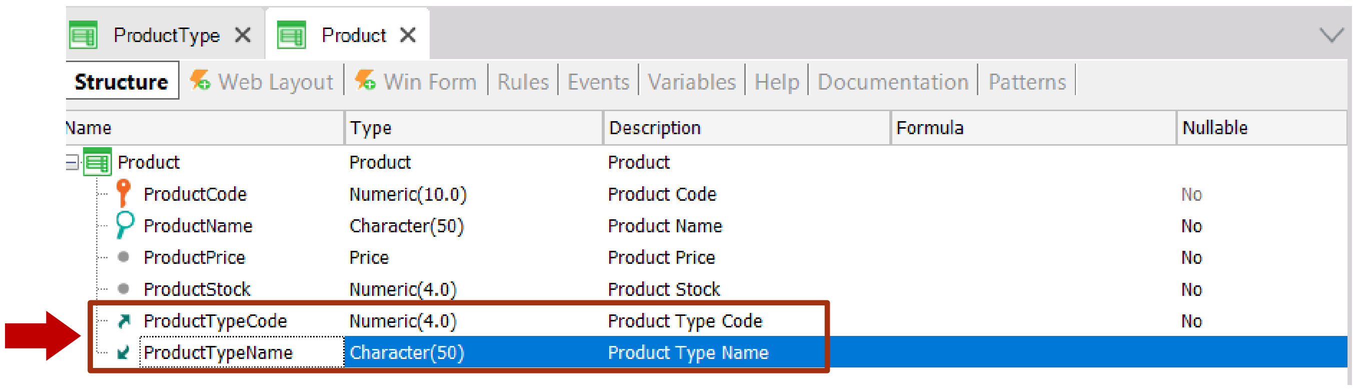
Now let's assign a product type to each product. So, let's go back to the *Product* Transaction and begin in a new line of its structure by typing the letter "P", so that the list of attributes existent in the Knowledge Base that begin with that letter is displayed:



Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No
P	Numeric(4.0)	Product Type Code		No

By selecting *ProductTypeCode,* its entire definition is displayed.

In this Transaction let's also include the *ProductTypeName* attribute, because when the users execute this Transaction and select a product type code, they will want to see the corresponding product type name in the form. Let's focus on these two attributes included in more than one Transaction:



ProductType | Product

Structure | Web Layout | Win Form | Rules | Events | Variables | Help | Documentation | Patterns

Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	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		

ProductTypeCode is the identifier attribute in 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 there, the attribute has the role of foreign key.

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.

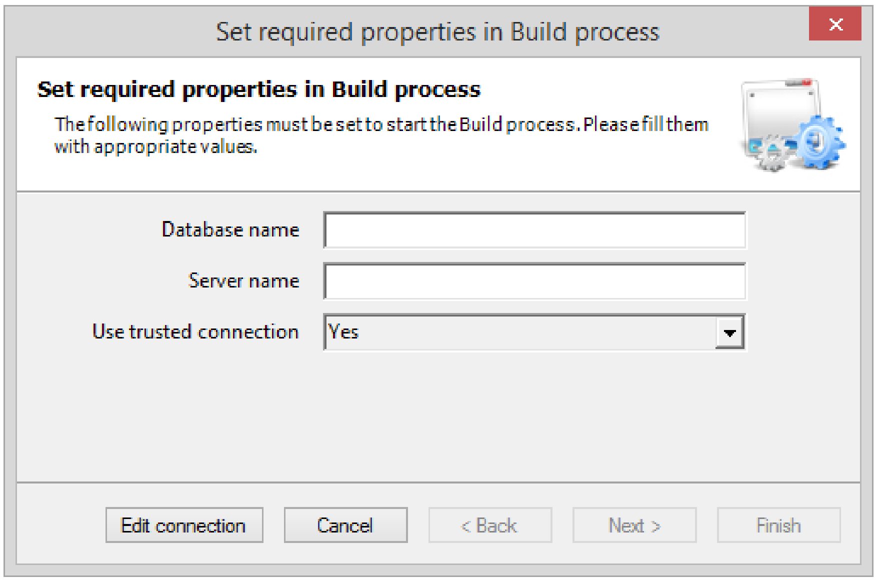
The *ProductTypeName* attribute is also present in both Transactions. However, it is not marked as the identifier of any of the defined Transactions. Therefore, GeneXus will take it as a secondary attribute. GeneXus will then include *ProductTypeName* in the *ProductType* physical table that it will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

The Transaction concept and the physical table concept 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 it must create in the database, as well as the attributes that it must store in each table.

In runtime, when executing the *Product* Transaction form, the user must enter for the *ProductTypeCode* attribute (which is a foreign key attribute there), a value that has been previously recorded 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:

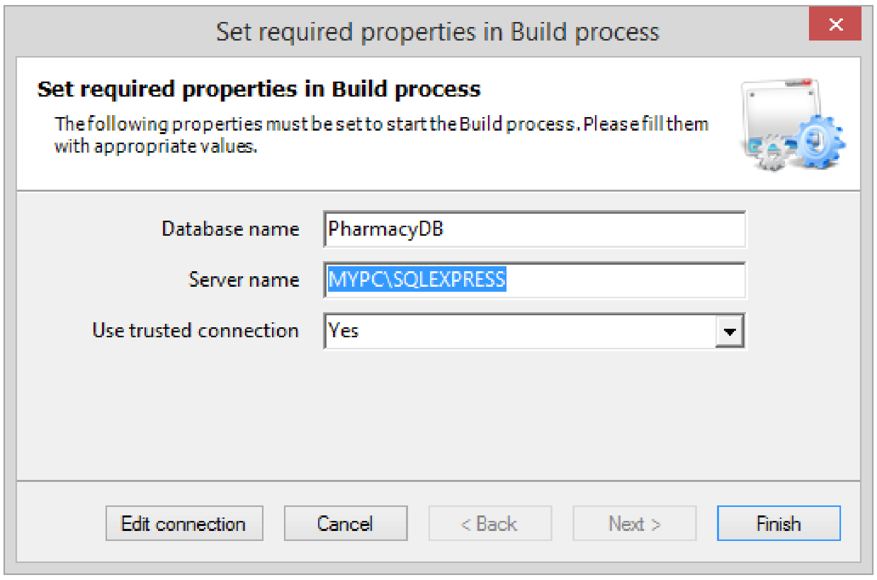

Set required properties in Build process

Set required properties in Build process

The following properties must be set to start the Build process. Please fill them with appropriate values.

Database name	
Server name	
Use trusted connection	Yes

GeneXus offers the possibility to prototype the application locally or in the cloud. There is a property to set this, and by default the property is set to prototype locally. So, you have to enter the name you want for your application's database and the name of the database server you have installed:



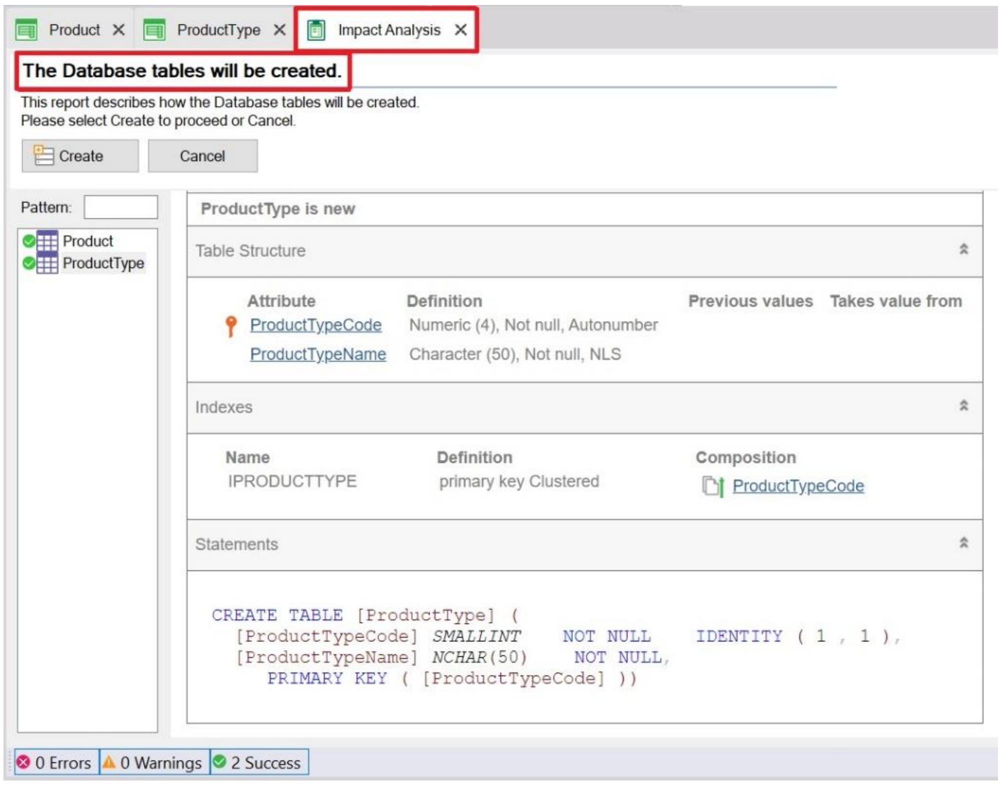
Set required properties in Build process

Set required properties in Build process

The following properties must be set to start the Build process. Please fill them with appropriate values.

Database name	PharmacyDB
Server name	MYPC\SQLEXPRESS
Use trusted connection	Yes

When you click on **Finish**, 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**:



The Database tables will be created.

This report describes how the Database tables will be created.
Please select Create to proceed or Cancel.

ProductType is new

Table Structure

Attribute	Definition	Previous values	Takes value from
ProductTypeCode	Numeric (4), Not null, Autonumber		
ProductTypeName	Character (50), Not null, NLS		

Indexes

Name	Definition	Composition
IPRODUCTTYPE	primary key Clustered	ProductTypeCode

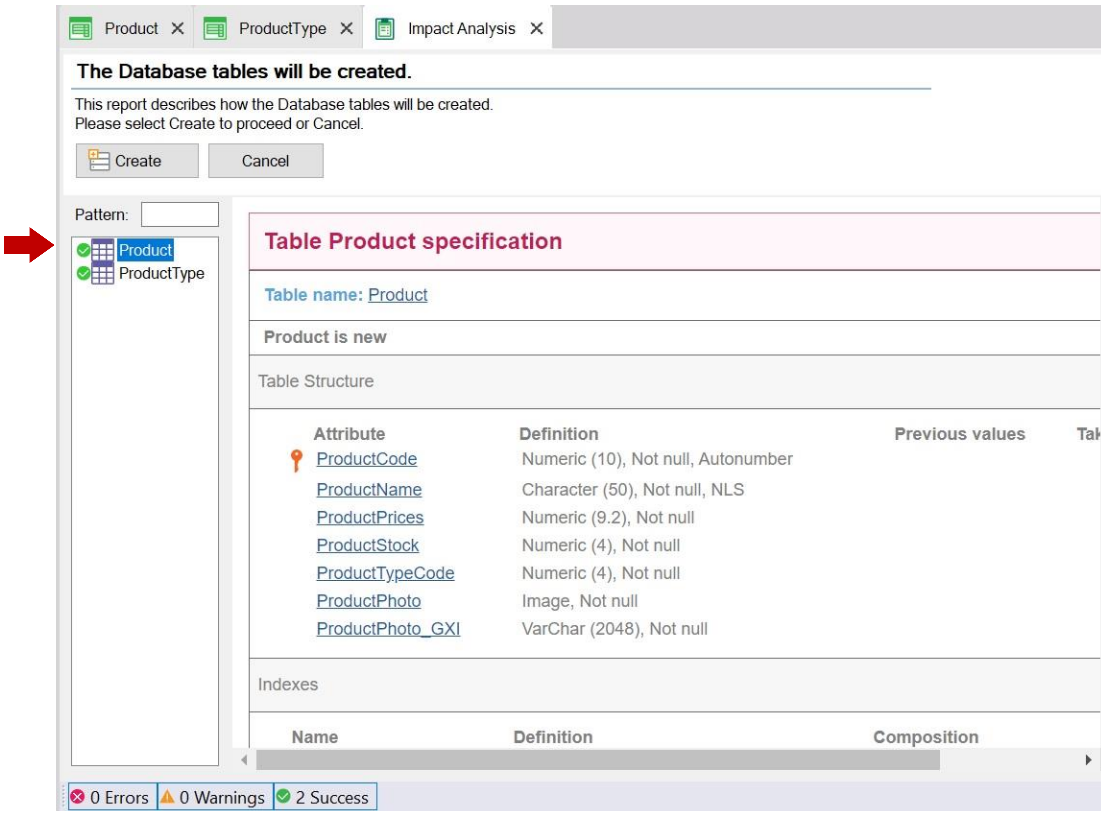
Statements

```
CREATE TABLE [ProductType] (
  [ProductTypeCode] SMALLINT    NOT NULL    IDENTITY ( 1 , 1 ),
  [ProductTypeName] NCHAR(50)    NOT NULL,
     PRIMARY KEY ( [ProductTypeCode] ))
```

This report shows which structural changes must be made in the database.

When you read the report, you will see that, in this case, the main title informs that **"The Database tables will be created".**

By clicking on each table (*ProductType* and *Product* in the left window), you will see at the right window the attributes that will be included in them:


Product ✕ | ProductType ✕ | Impact Analysis ✕

The Database tables will be created.

This report describes how the Database tables will be created.
Please select Create to proceed or Cancel.

Create | Cancel

Pattern:

- Product
- ProductType

Table Product specification

Table name: Product

Product is new

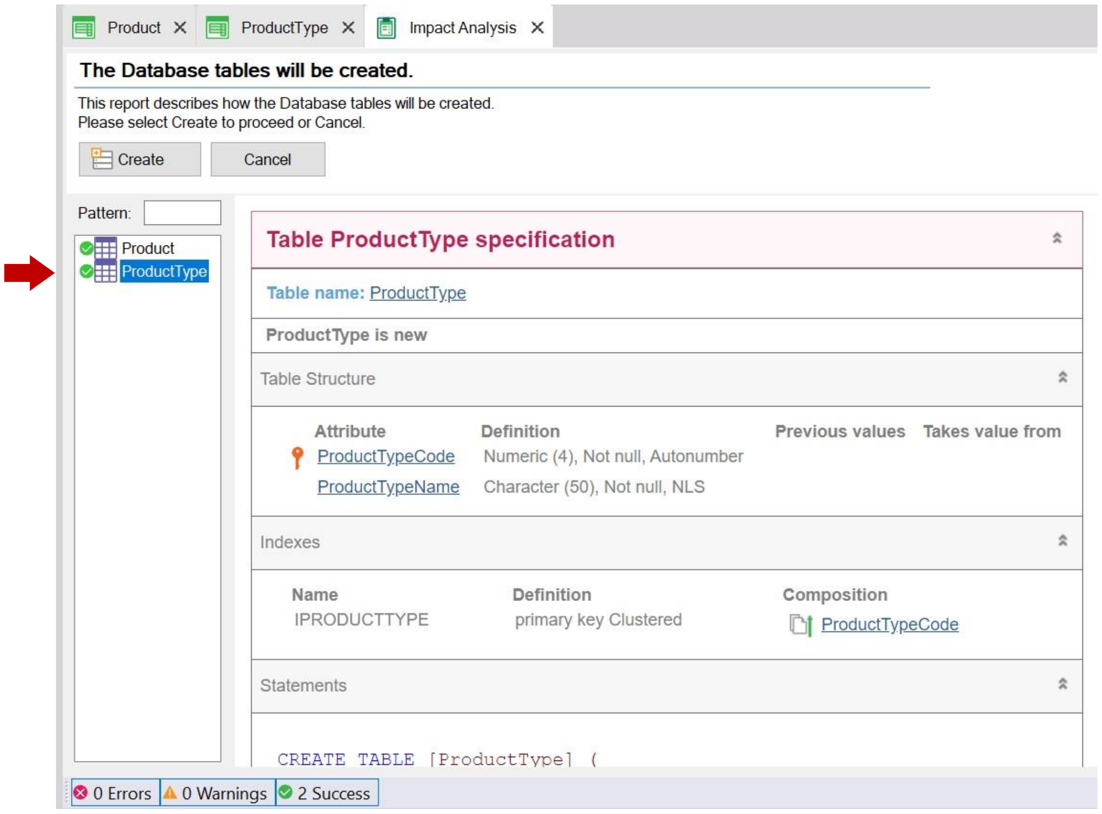
Table Structure

	Attribute	Definition	Previous values	Tak
🔑	ProductCode	Numeric (10), Not null, Autonumber		
	ProductName	Character (50), Not null, NLS		
	ProductPrices	Numeric (9.2), Not null		
	ProductStock	Numeric (4), Not null		
	ProductTypeCode	Numeric (4), Not null		
	ProductPhoto	Image, Not null		
	ProductPhoto_GXI	VarChar (2048), Not null		

Indexes

Name	Definition	Composition

0 Errors | 0 Warnings | 2 Success


Product ✕ | ProductType ✕ | Impact Analysis ✕

The Database tables will be created.

This report describes how the Database tables will be created.
Please select Create to proceed or Cancel.

Create | Cancel

Pattern:

- Product
- ProductType

Table ProductType specification

Table name: ProductType

ProductType is new

Table Structure

	Attribute	Definition	Previous values	Takes value from
🔑	ProductTypeCode	Numeric (4), Not null, Autonumber		
	ProductTypeName	Character (50), Not null, NLS		

Indexes

Name	Definition	Composition
IPRODUCTTYPE	primary key Clustered	ProductTypeCode

Statements

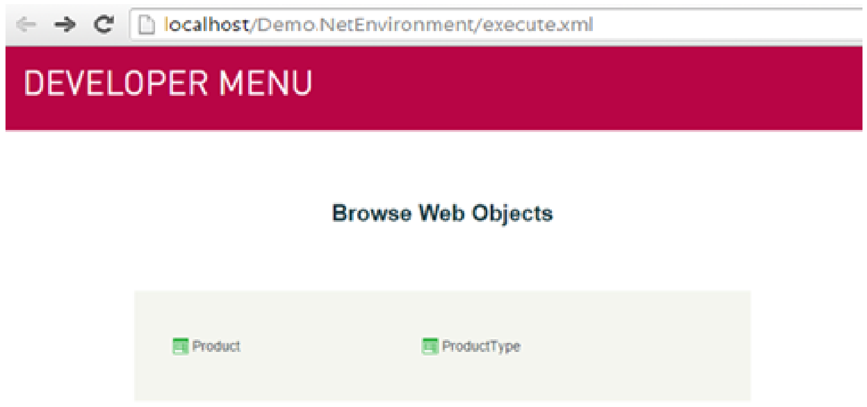
```
CREATE TABLE [ProductType] (
```

0 Errors | 0 Warnings | 2 Success

Note that as it was 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 form).

If you agree with the Impact Analysis proposal, you can click on the Create button, and GeneXus will start creating the programs needed to create the database (still inexistent), as well as the tables with their structures in that database. Next, GeneXus executes those 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.

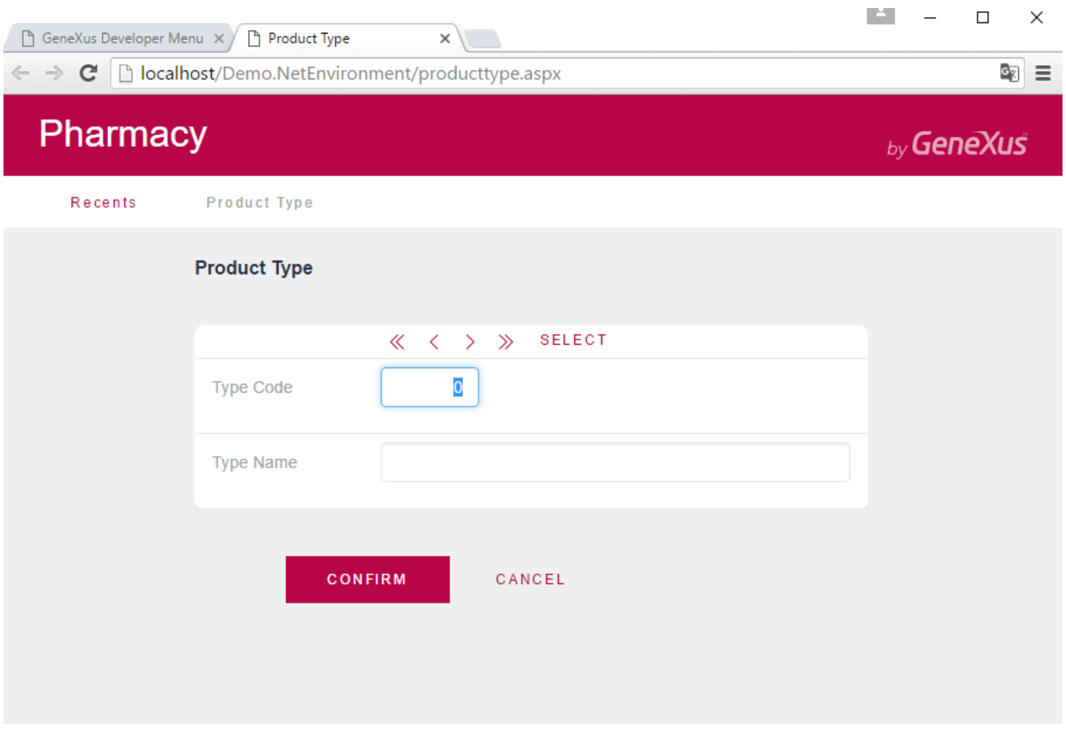
You are then informed if the result was successful or if there were any errors or warnings, and continuously you will see the application running:



The Web browser is opened by default, showing a simple page that offers a quick way to execute the defined objects.

This simple page, called Developer menu, is for developers, as indicated by its name. Of course, it is not what users will view on screen.

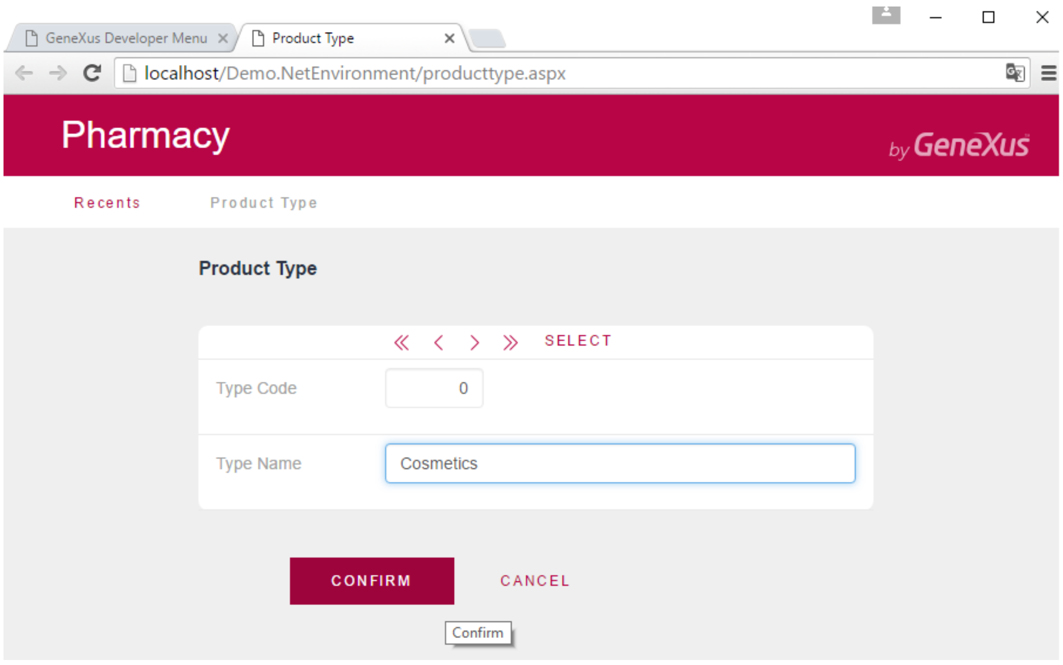
Let’s right-click on the *ProductType* link and choose to open it in a new tab:



Type Code	0
Type Name	

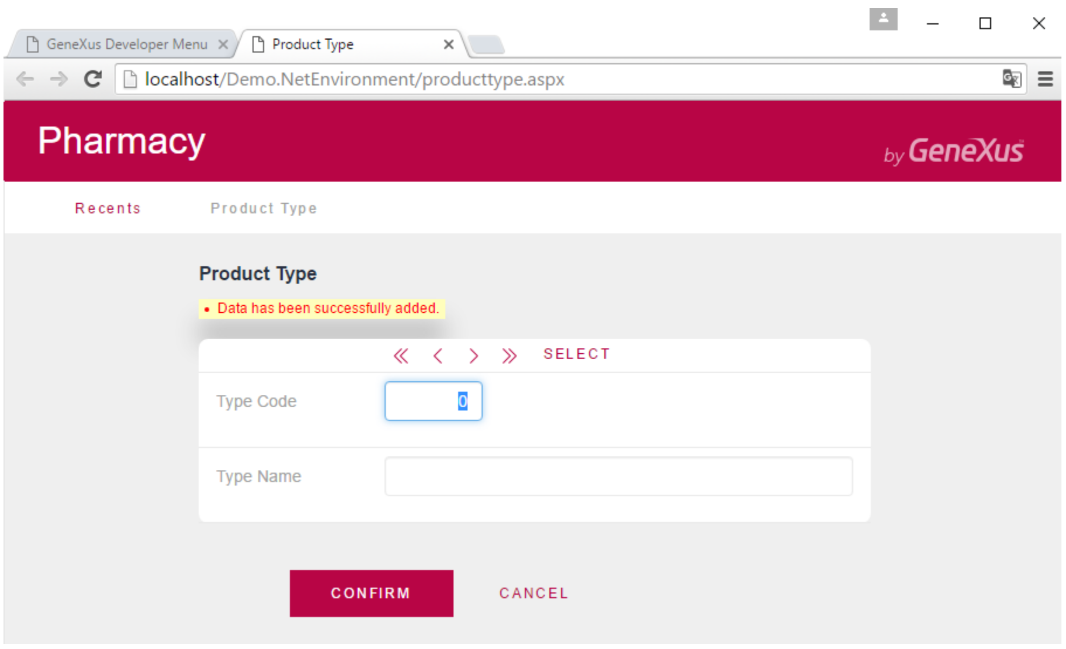
The above page allows the user to add, update and delete *product types*. Let's enter the first product type.

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, let's enter the product type name:



Type Code	0
Type Name	Cosmetics

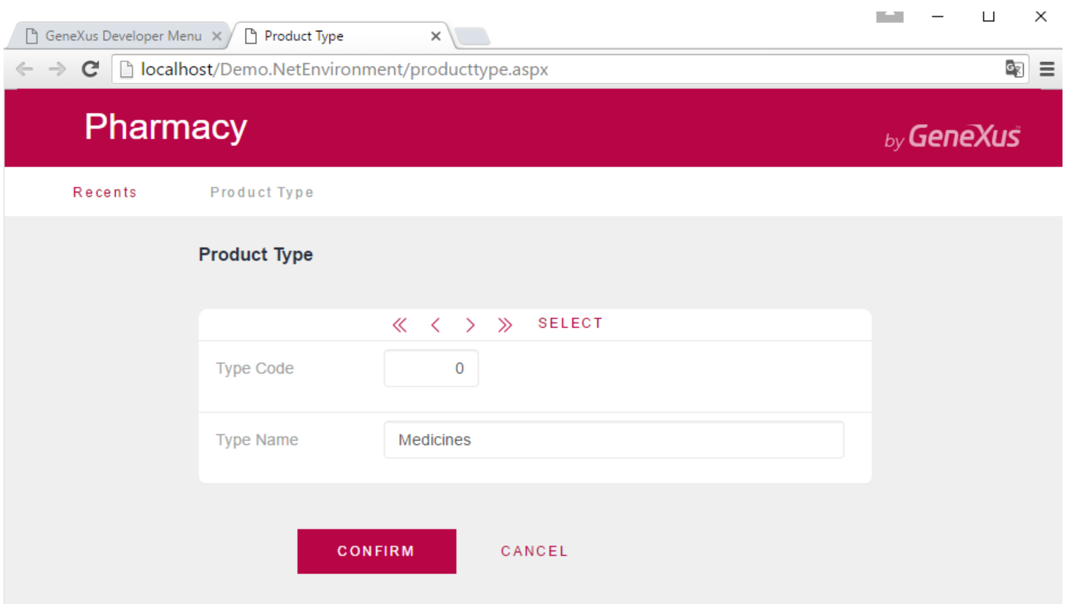
After entering the product type name and clicking on the Confirm button, a message will be displayed to inform that the data was added successfully; meanwhile, the form is cleared and is ready to enter another product type:



Data has been successfully added.

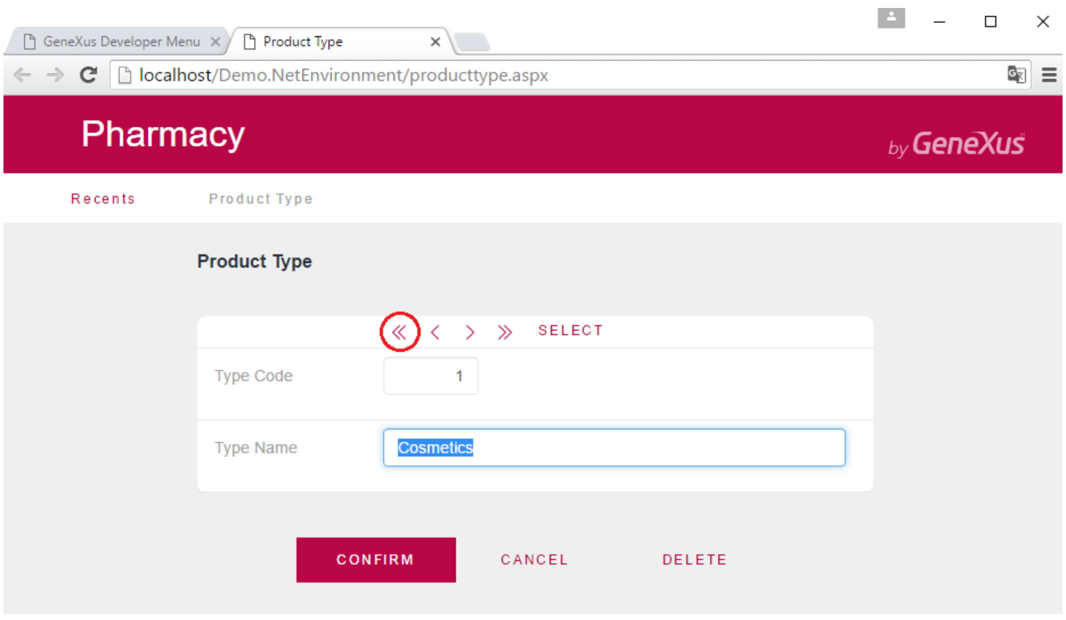
Type Code	0
Type Name	

Let's enter the second product type:

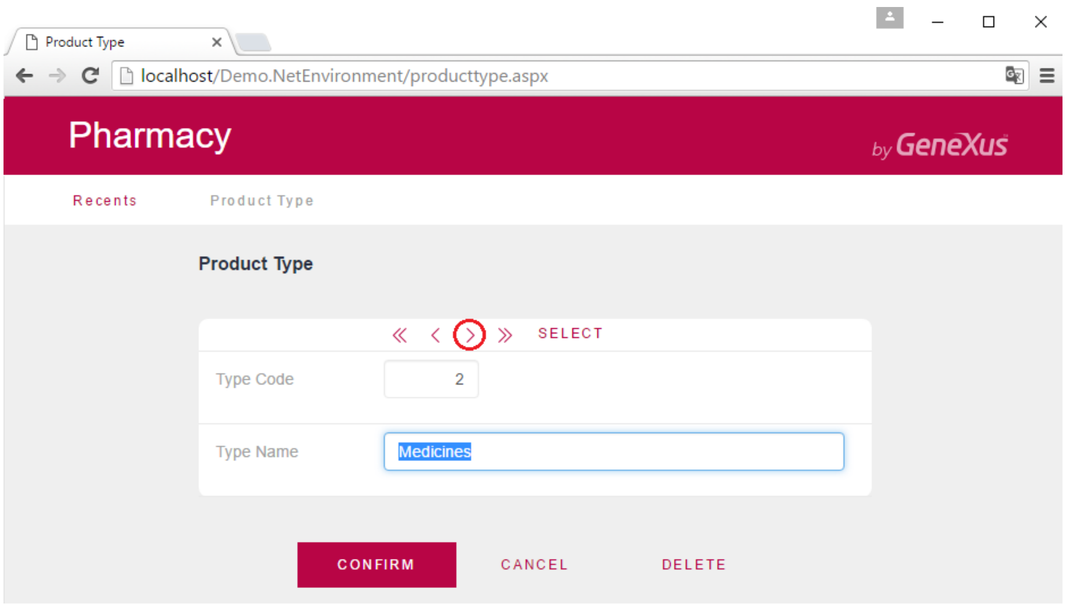


Type Code	0
Type Name	Medicines

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

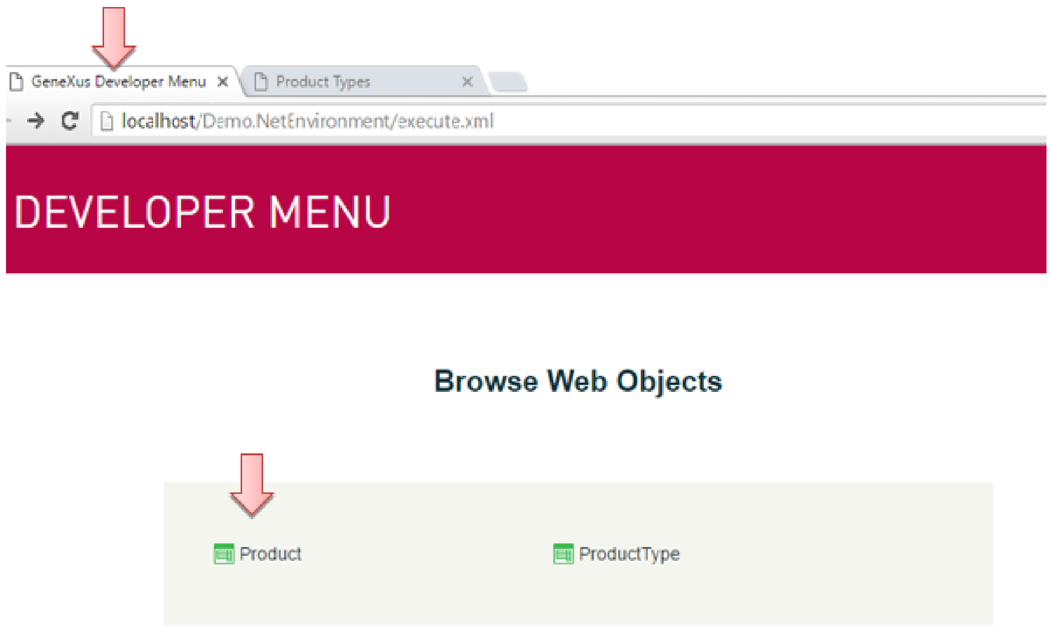


Type Code	1
Type Name	Cosmetics

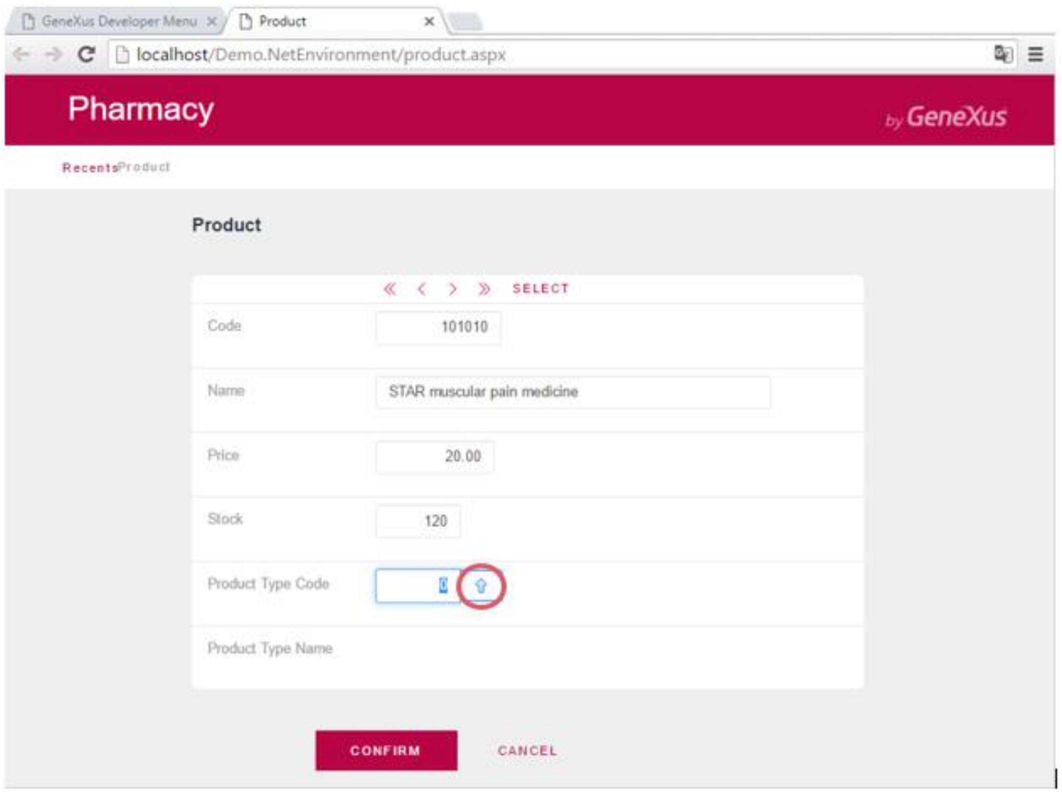


Type Code	2
Type Name	Medicines

Now let’s execute the *Product* Transaction. To do so, choose the navigator tab where the Developer Menu page is displayed, and then let’s execute the *Product* Transaction:

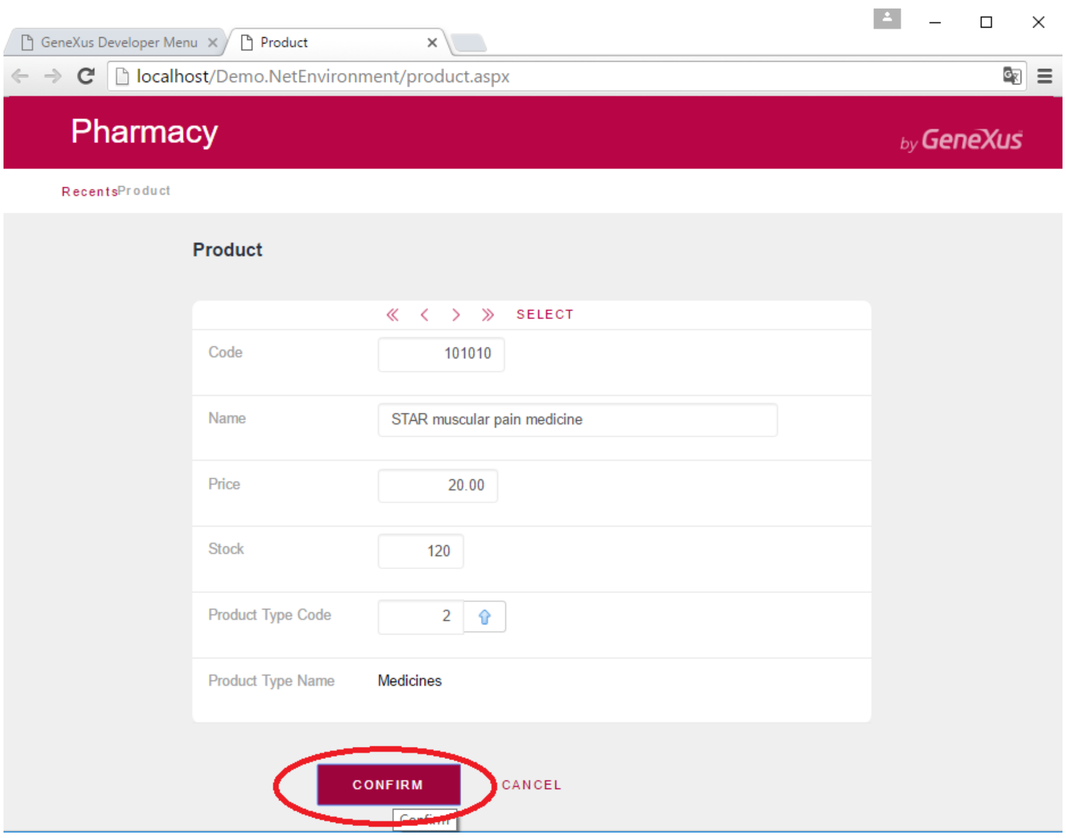


Try to add the first product:



Code	101010
Name	STAR muscular pain medicine
Price	20.00
Stock	120
Product Type Code	0
Product Type Name	

You must indicate the product type. If you remember the product type code you can enter it, and another option is to select it from a list by clicking on the arrow.

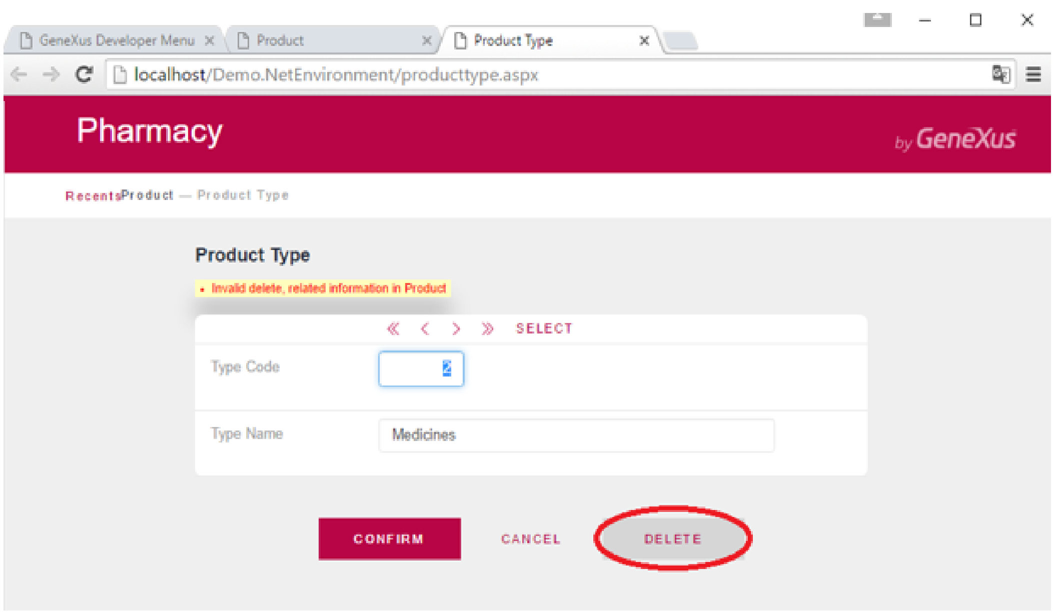

localhost/Demo.NetEnvironment/product.aspx

Pharmacy

Product

Code	101010
Name	STAR muscular pain medicine
Price	20.00
Stock	120
Product Type Code	2
Product Type Name	Medicines

Now, try to delete a product type:


localhost/Demo.NetEnvironment/producttype.aspx

Pharmacy

Product Type

• Invalid delete, related information in Product

Type Code	2
Type Name	Medicines

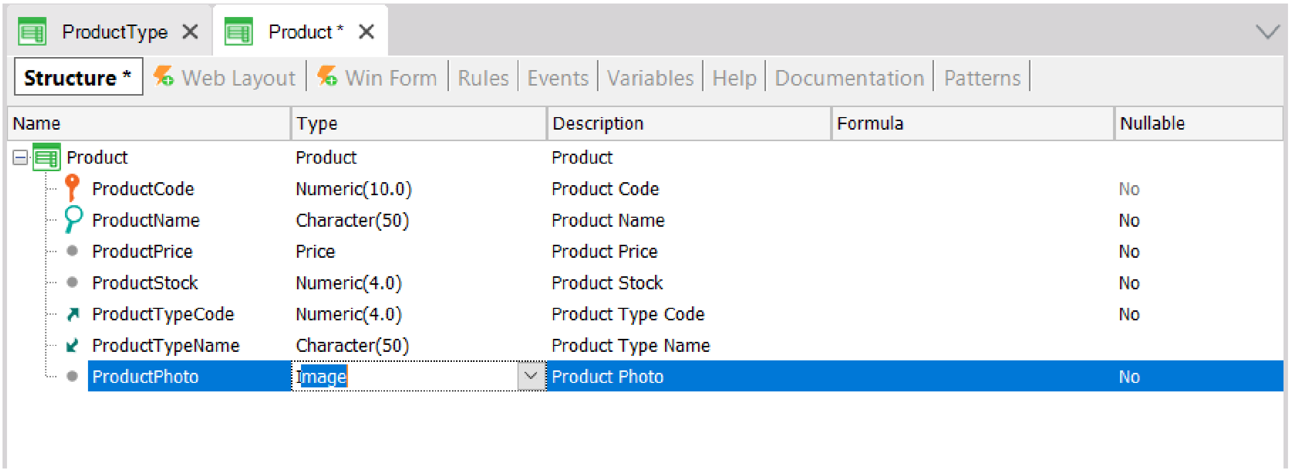
A message informs you that the deletion cannot be performed because related data exists in Product (the STAR muscular pain medicine is a product that belongs to this product type).

THE APPLICATION GROWS UP

You might have seen how much was automatically generated by GeneXus from the two objects you have defined.

Now suppose at the pharmacy you are told that, for each product, they want to record an image.

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

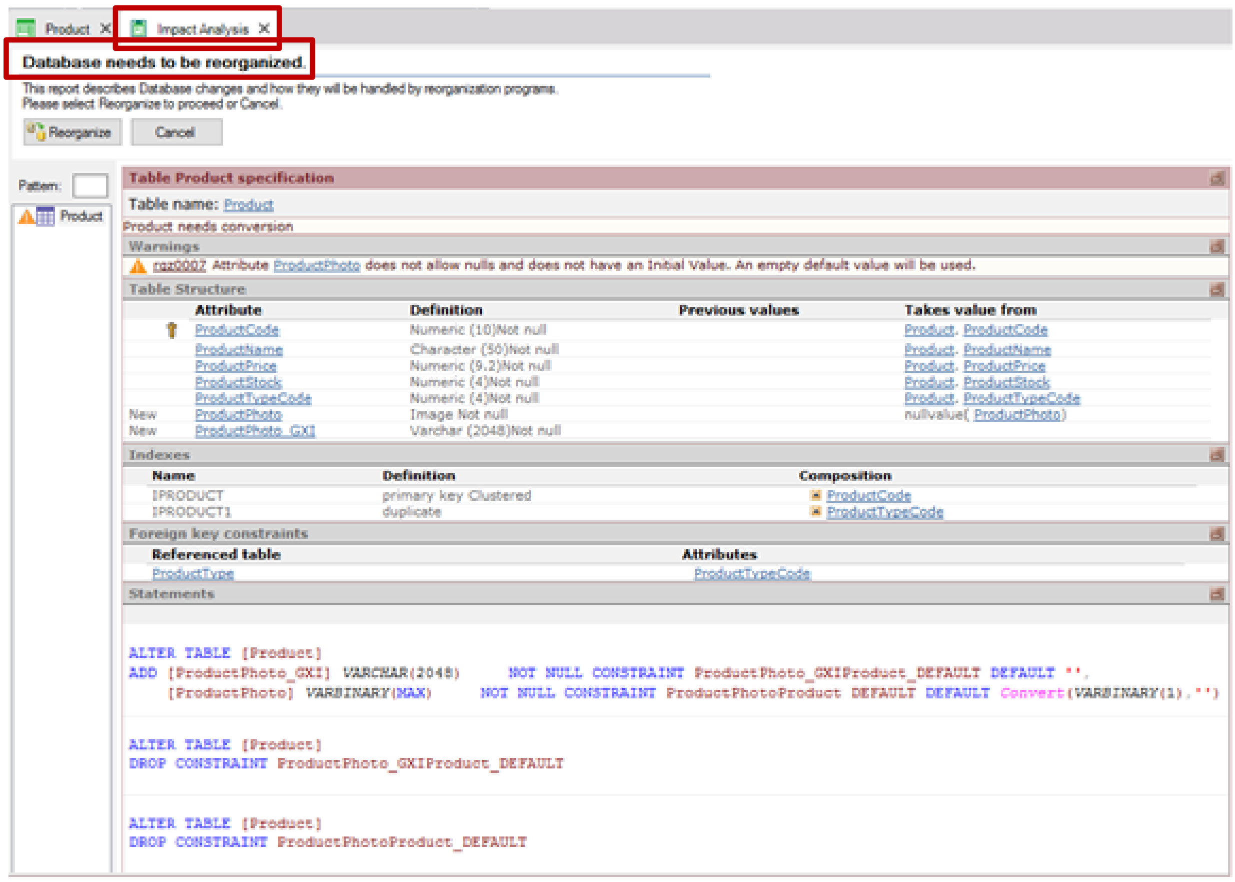


Name	Type	Description	Formula	Nullable
Product	Product	Product		
ProductCode	Numeric(10.0)	Product Code		No
ProductName	Character(50)	Product Name		No
ProductPrice	Price	Product Price		No
ProductStock	Numeric(4.0)	Product Stock		No
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	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:



Database needs to be reorganized.

This report describes Database changes and how they will be handled by reorganization programs.
Please select Reorganize to proceed or Cancel.

Table Product specification

Table name: Product

Product needs conversion

Warnings

⚠ rgz0007 Attribute ProductPhoto does not allow nulls and does not have an Initial Value. An empty default value will be used.

Table Structure

	Attribute	Definition	Previous values	Takes value from
🔑	ProductCode	Numeric (10)Not null		Product. ProductCode
	ProductName	Character (50)Not null		Product. ProductName
	ProductPrice	Numeric (9.2)Not null		Product. ProductPrice
	ProductStock	Numeric (4)Not null		Product. ProductStock
	ProductTypeCode	Numeric (4)Not null		Product. ProductTypeCode
New	ProductPhoto	Image Not null		nullvalue(ProductPhoto)
New	ProductPhoto_GXI	Varchar (2048)Not null		

Indexes

Name	Definition	Composition
IPRODUCT	primary key Clustered	ProductCode
IPRODUCT1	duplicate	ProductTypeCode

Foreign key constraints

Referenced table	Attributes
ProductType	ProductTypeCode

Statements

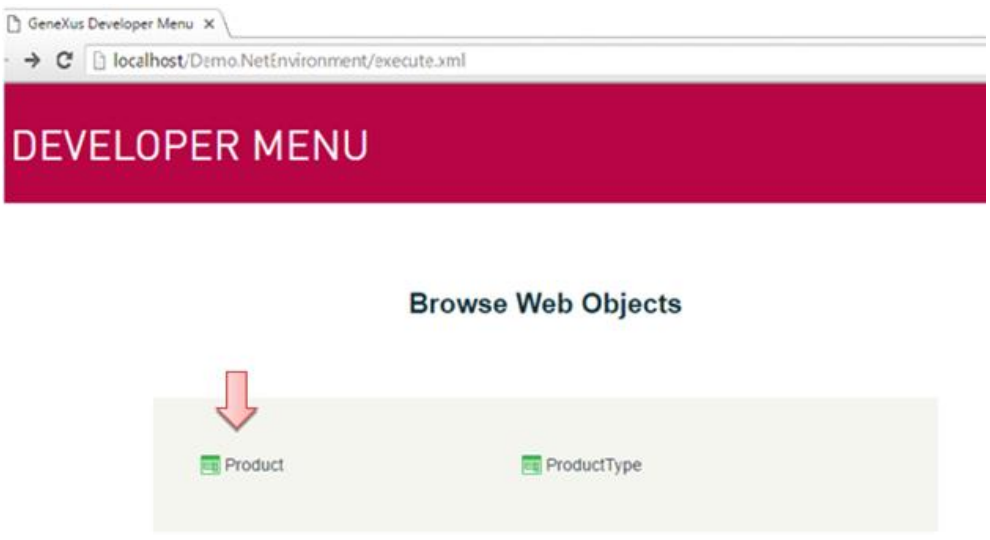
```
ALTER TABLE [Product]
ADD [ProductPhoto_GXI] VARCHAR(2048)      NOT NULL CONSTRAINT ProductPhoto_GXIProduct_DEFAULT DEFAULT '',
    [ProductPhoto] VARBINARY(MAX)      NOT NULL CONSTRAINT ProductPhotoProduct DEFAULT DEFAULT Convert(VARBINARY(1),'')

ALTER TABLE [Product]
DROP CONSTRAINT ProductPhoto_GXIProduct_DEFAULT

ALTER TABLE [Product]
DROP CONSTRAINT ProductPhotoProduct_DEFAULT
```

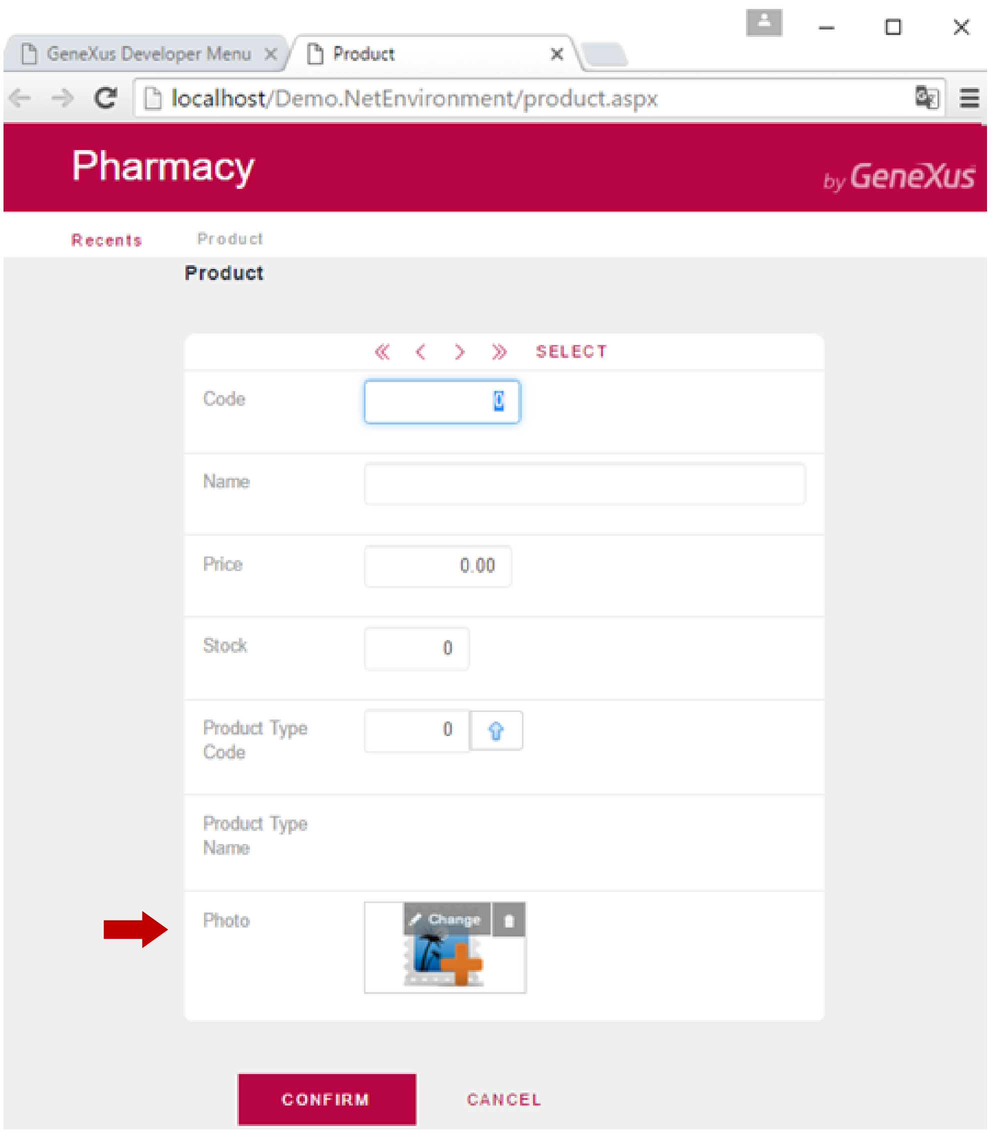
Remember: The Impact Analysis Report indicates the structural changes required in the database.

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 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 programs required that correspond to the application in itself.

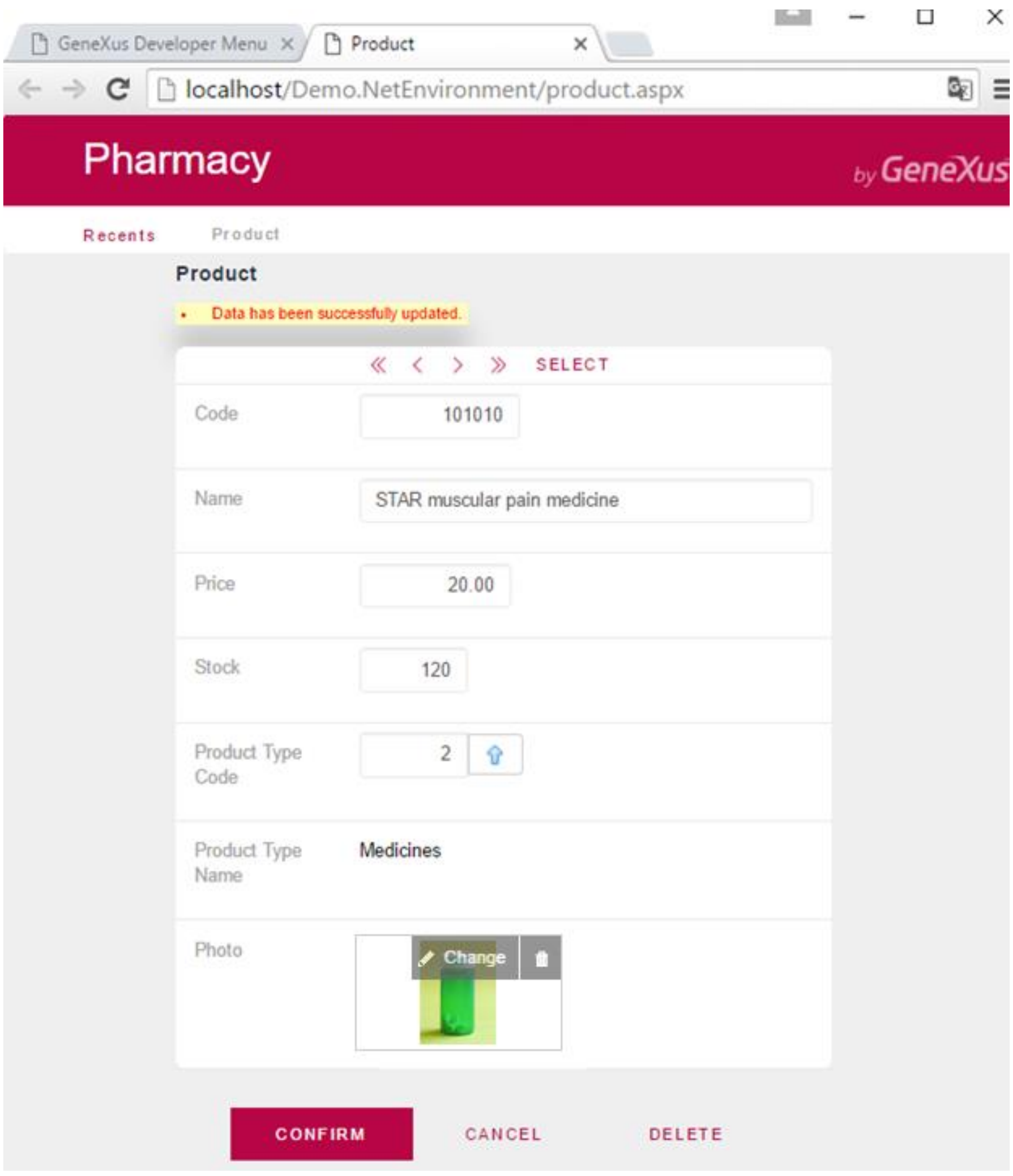
Note that immediately you will have the application running again in the web browser, with the new definition included:





Code	
Name	
Price	0.00
Stock	0
Product Type Code	0
Product Type Name	
Photo	

If you remember the product code you may enter it, and 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:



Product

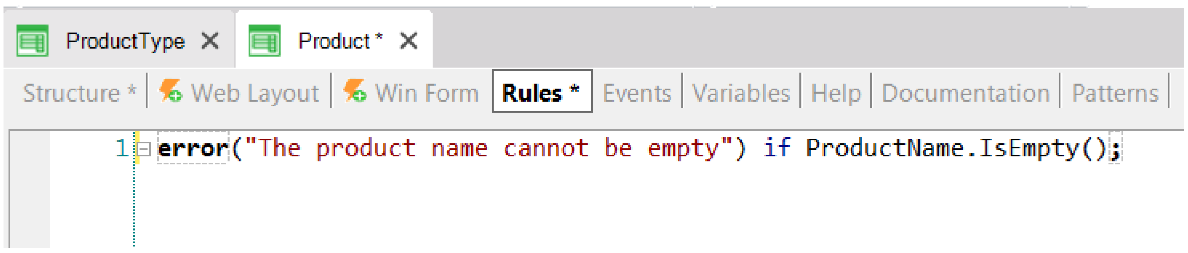
- Data has been successfully updated.

Code	101010
Name	STAR muscular pain medicine
Price	20.00
Stock	120
Product Type Code	2
Product Type Name	Medicines
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.

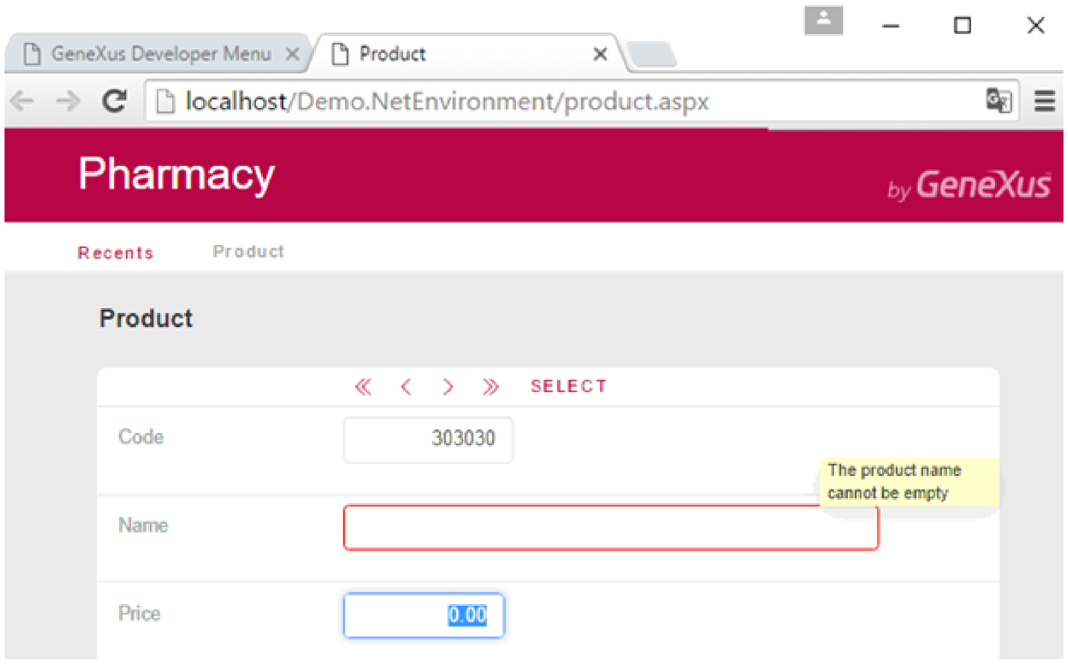
If, for example, a requirement is not to allow storing of products without a name, GeneXus offers a rule called Error that will enable you to avoid that:



```
error("The product name cannot be empty") if ProductName.IsEmpty();
```

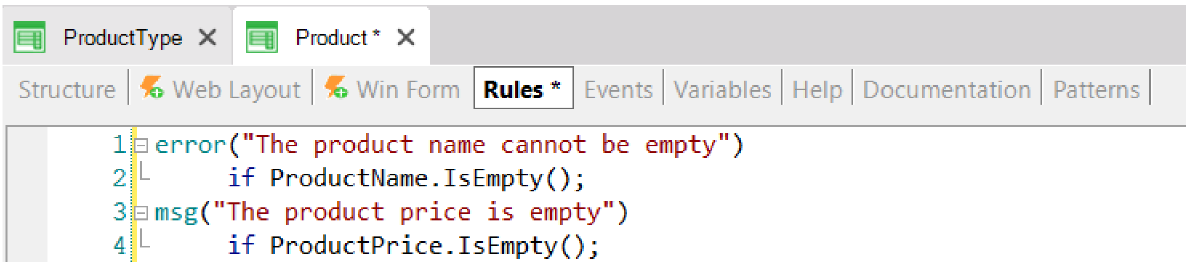
Press F5 and GeneXus will save and evaluate the new definitions included the Knowledge Base. In this case it will infer that it is not necessary to modify the database, so it will not show an Impact Analysis Report. GeneXus will generate the necessary code and after that, it will execute the application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



```
error("The product name cannot be empty")
      if ProductName.IsEmpty();
msg("The product price is empty")
      if ProductPrice.IsEmpty();
```

This set of rules could be written in any other order and the result at runtime would be exactly the same, because 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 kind of validations and actions.

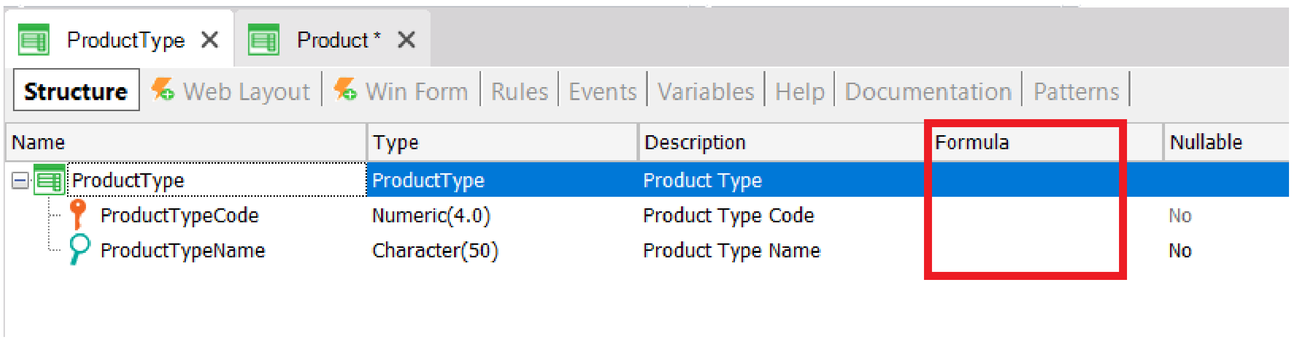
Each Transaction may need to have its own behavior rules defined.

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.

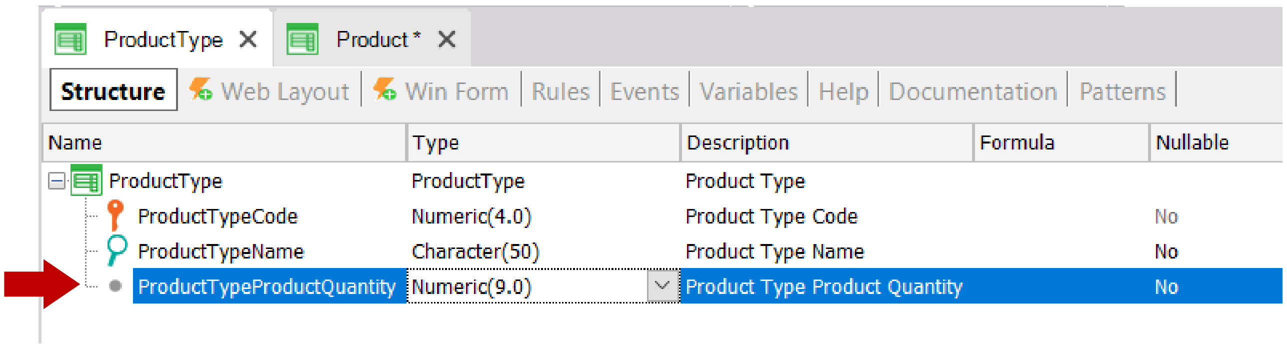
Let's start by learning what a **Global Formula** is. A global formula is a calculation you define associated with an attribute. Note that the Transaction structures contain a column labeled **Formula**:



Name	Type	Description	Formula	Nullable
ProductType	ProductType	Product Type		
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	Product Type Name		No

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

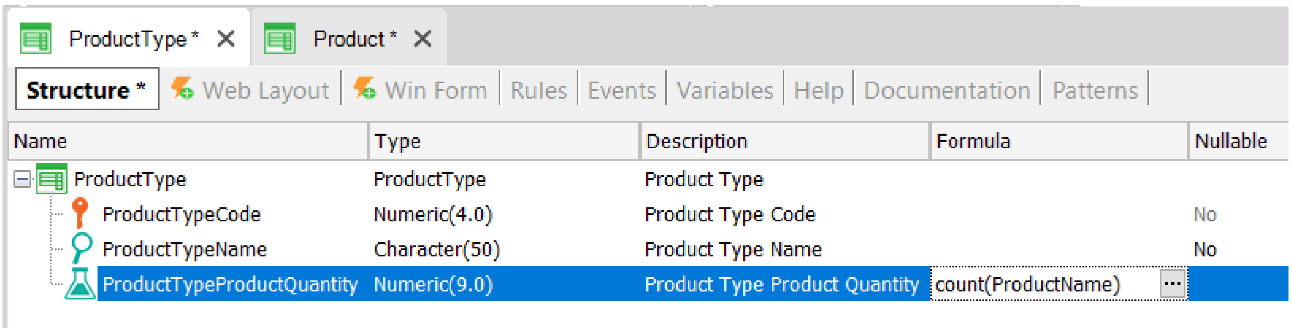
Let's see this with an example. Suppose the pharmacy needs to know at all times how many registered products there are of each product type. So, let's define a new attribute in the *ProductType* Transaction with the purpose of defining it as a global formula:



Name	Type	Description	Formula	Nullable
ProductType	ProductType	Product Type		
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	Product Type Name		No
ProductTypeProductQuantity	Numeric(9.0)	Product Type Product Quantity		No

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

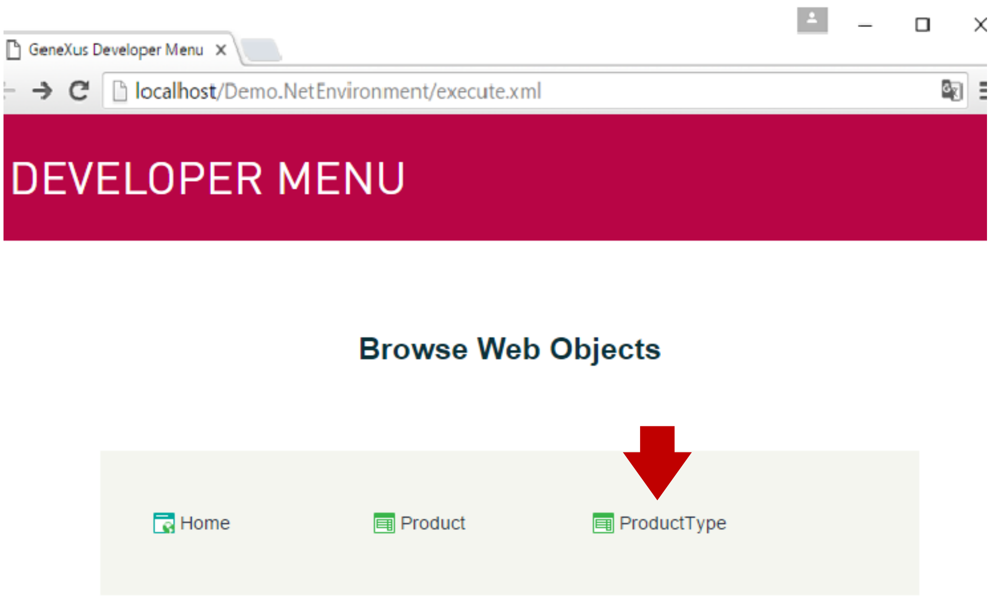
GeneXus offers a formula called Count to calculate the pharmacy need (there are many others, like Sum, Average, etc.).



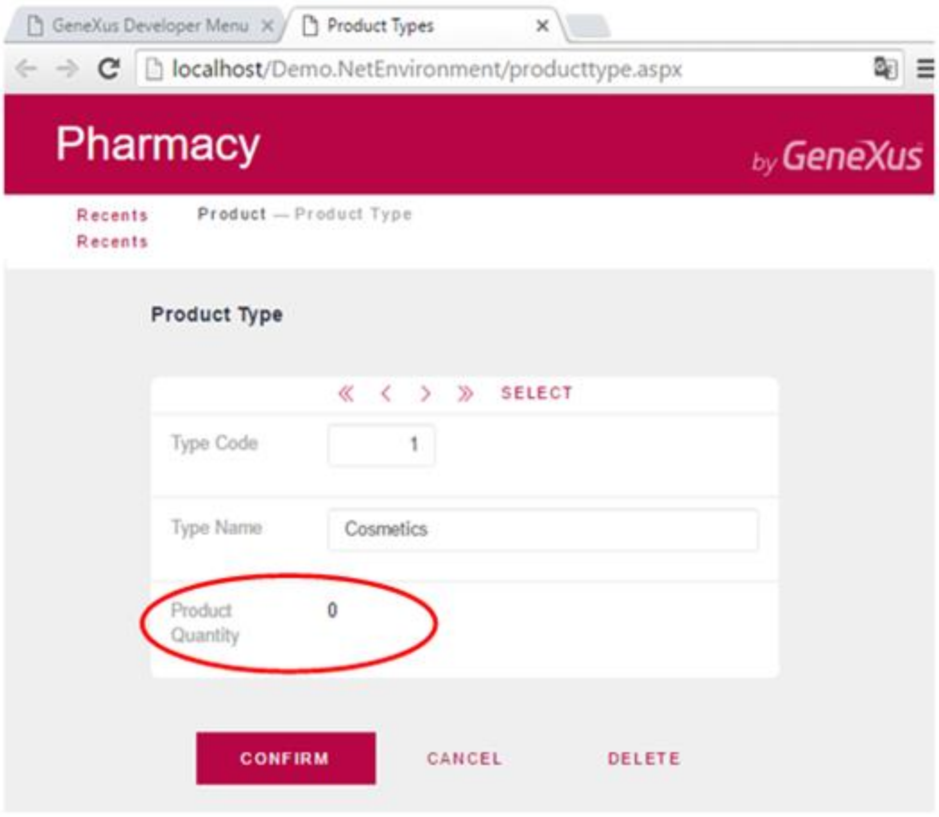
Name	Type	Description	Formula	Nullable
ProductType	ProductType	Product Type		
ProductTypeCode	Numeric(4.0)	Product Type Code		No
ProductTypeName	Character(50)	Product Type Name		No
ProductTypeProductQuantity	Numeric(9.0)	Product Type Product Quantity	count(ProductName)	

The attribute referenced inside the parenthesis 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, *ProductTyepId* 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 products of each product type are counted and not all 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 some programs and you will get the Developer Menu running again:

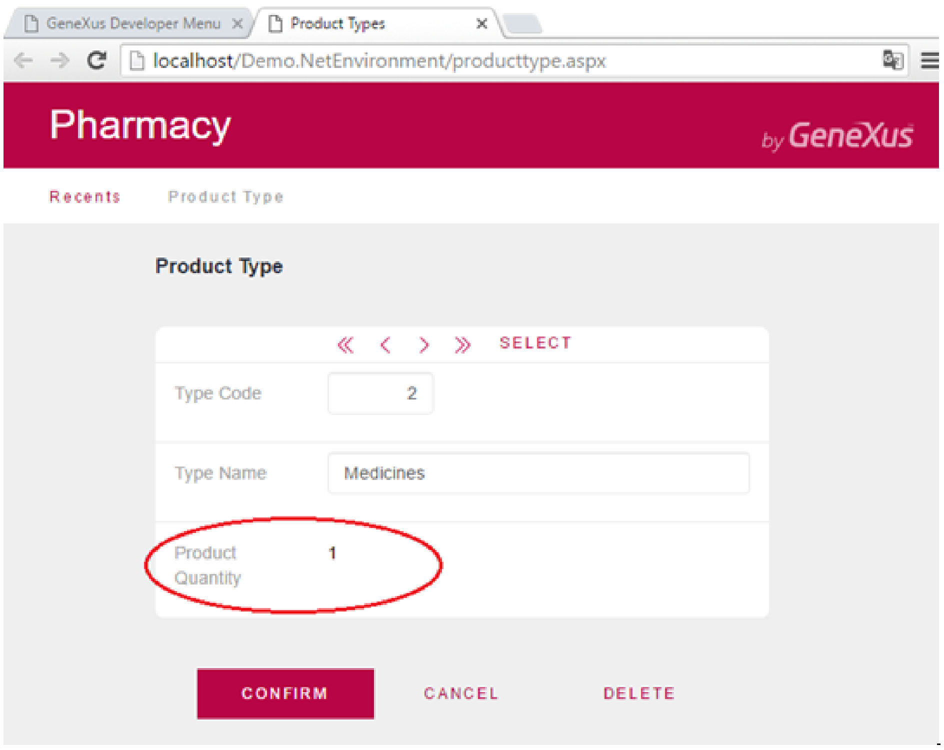


Execute the *ProductType* Transaction in order to see for each product type how the product quantity is always calculated at the time:



Product Type

Type Code	1
Type Name	Cosmetics
Product Quantity	0



Product Type

Type Code	2
Type Name	Medicines
Product Quantity	1

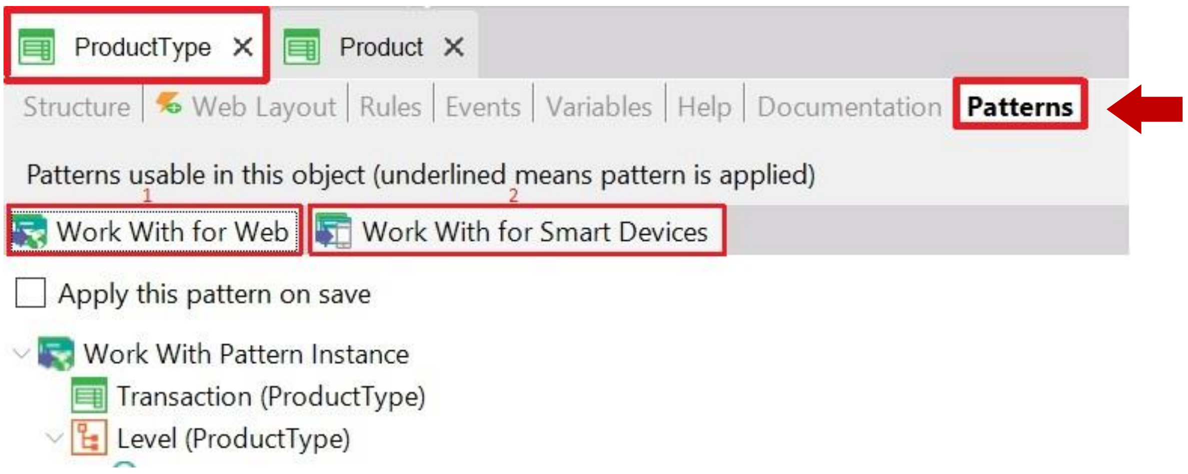
You can add more products in order to verify for each product type how the product quantity is always calculated at the time.

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.

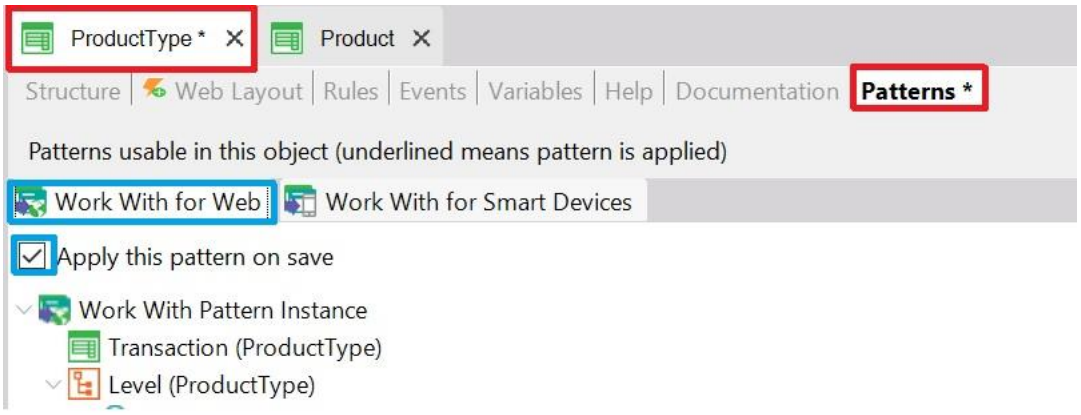
Look at the Patterns section of a Transaction. For example, in the *ProductType* Transaction, select the Patterns section:


ProductType
Product
Structure
Web Layout
Rules
Events
Variables
Help
Documentation
Patterns
Patterns usable in this object (underlined means pattern is applied)
1
2
Work With for Web
Work With for Smart Devices
Apply this pattern on save
Work With Pattern Instance
Transaction (ProductType)
Level (ProductType)

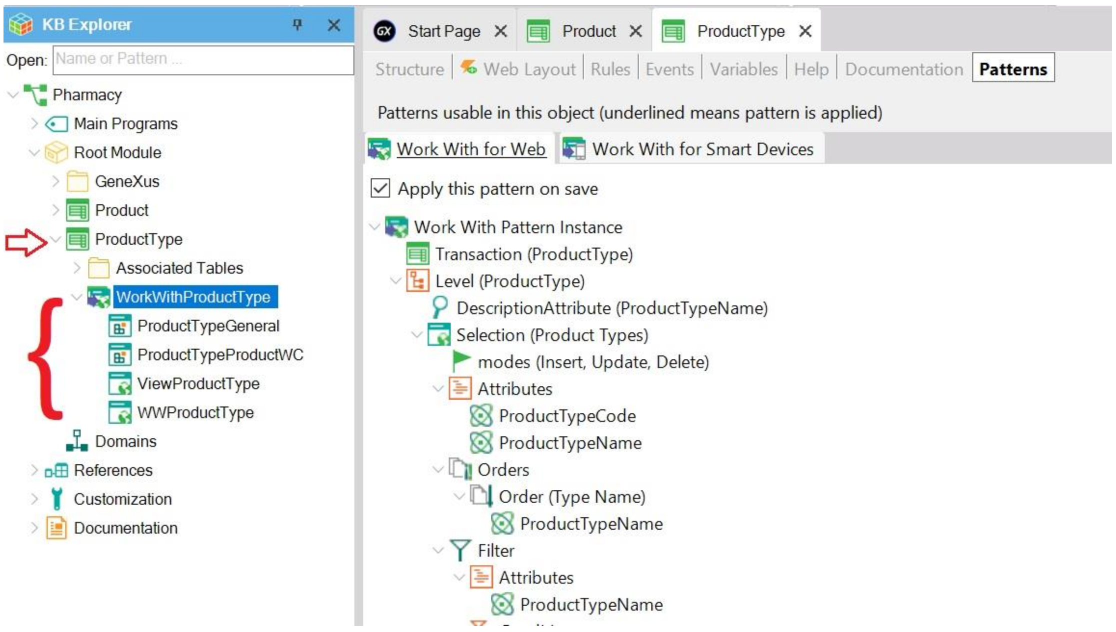
Note that two tabs are available, each one offering a different pattern to be applied to the same Transaction.

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 checkbox **Apply this pattern on save** and save ( ):

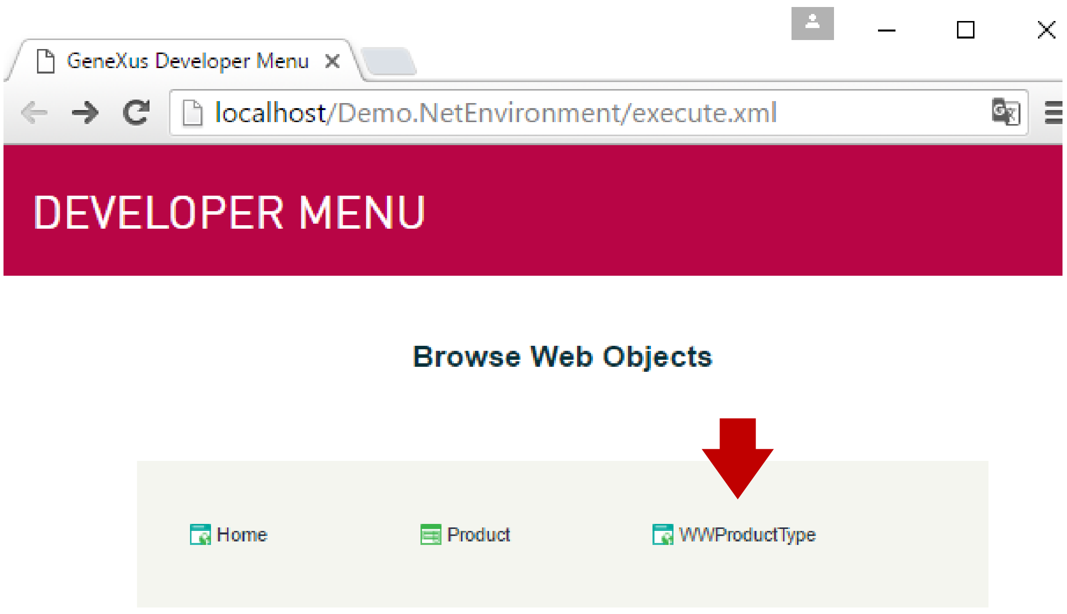

ProductType *
Product
Structure
Web Layout
Rules
Events
Variables
Help
Documentation
Patterns *
Patterns usable in this object (underlined means pattern is applied)
Work With for Web
Work With for Smart Devices
Apply this pattern on save
Work With Pattern Instance
Transaction (ProductType)
Level (ProductType)

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

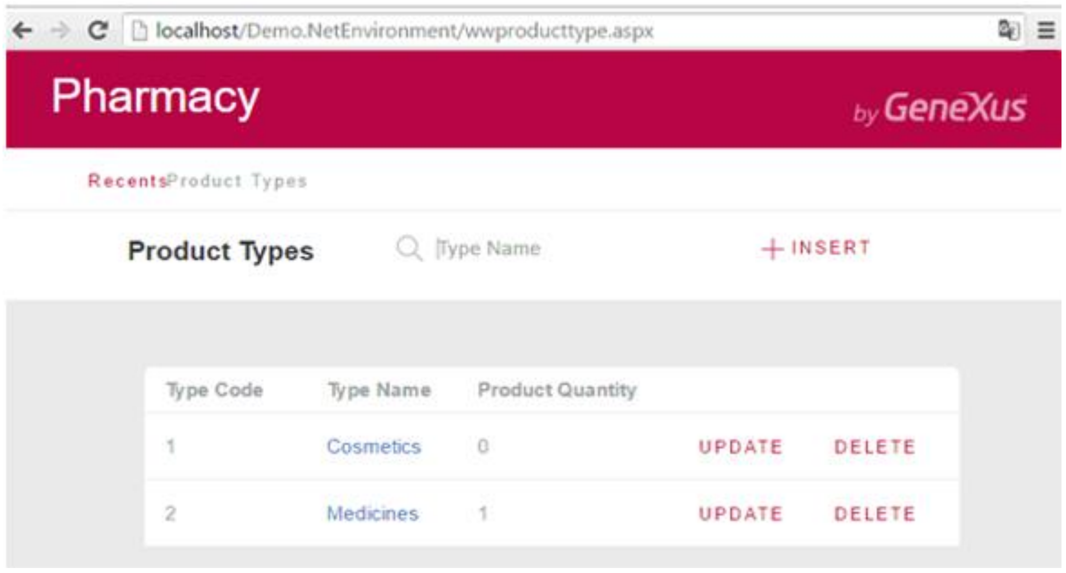


They were created by GeneXus when the *Work With for Web* pattern was applied.

Now press F5 to view in runtime the results:



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


localhost/Demo.NetEnvironment/wwproducttype.aspx

Pharmacy by GeneXus

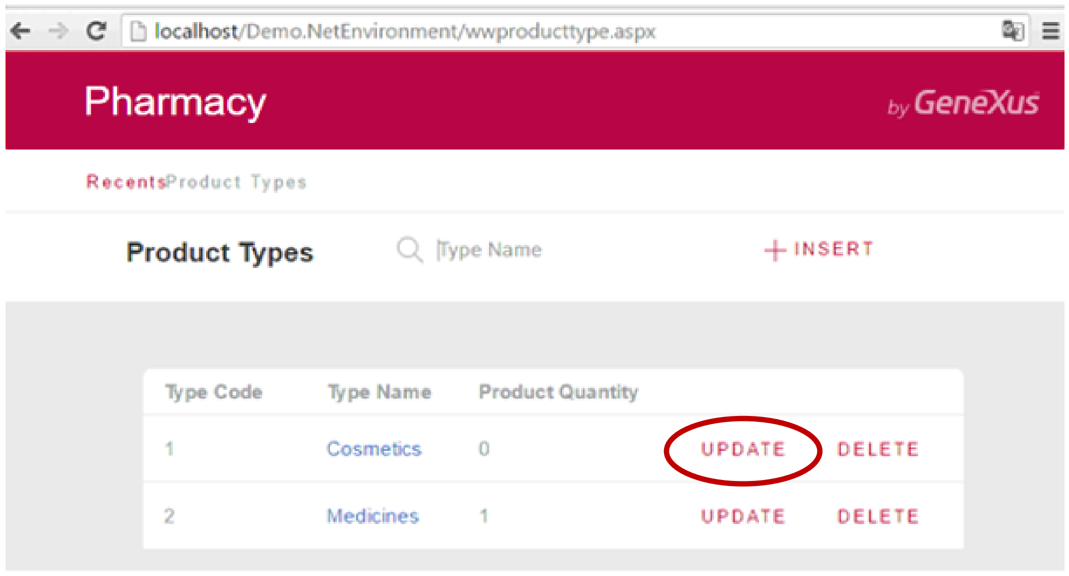
Recents Product Types

Product Types | Type Name | + INSERT

Type Code	Type Name	Product Quantity		
1	Cosmetics	0	UPDATE	DELETE
2	Medicines	1	UPDATE	DELETE

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

For example, click UPDATE for the first line:

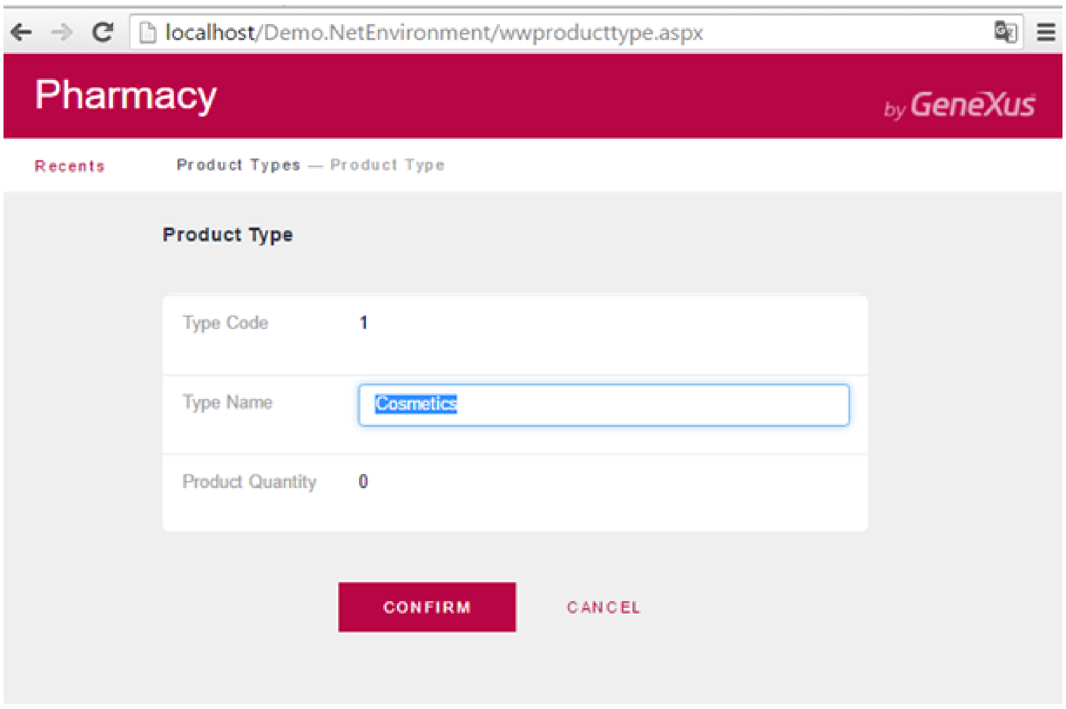

localhost/Demo.NetEnvironment/wwproducttype.aspx

Pharmacy by GeneXus

Recents Product Types

Product Types | Type Name | + INSERT

Type Code	Type Name	Product Quantity		
1	Cosmetics	0	UPDATE	DELETE
2	Medicines	1	UPDATE	DELETE


localhost/Demo.NetEnvironment/wwproducttype.aspx

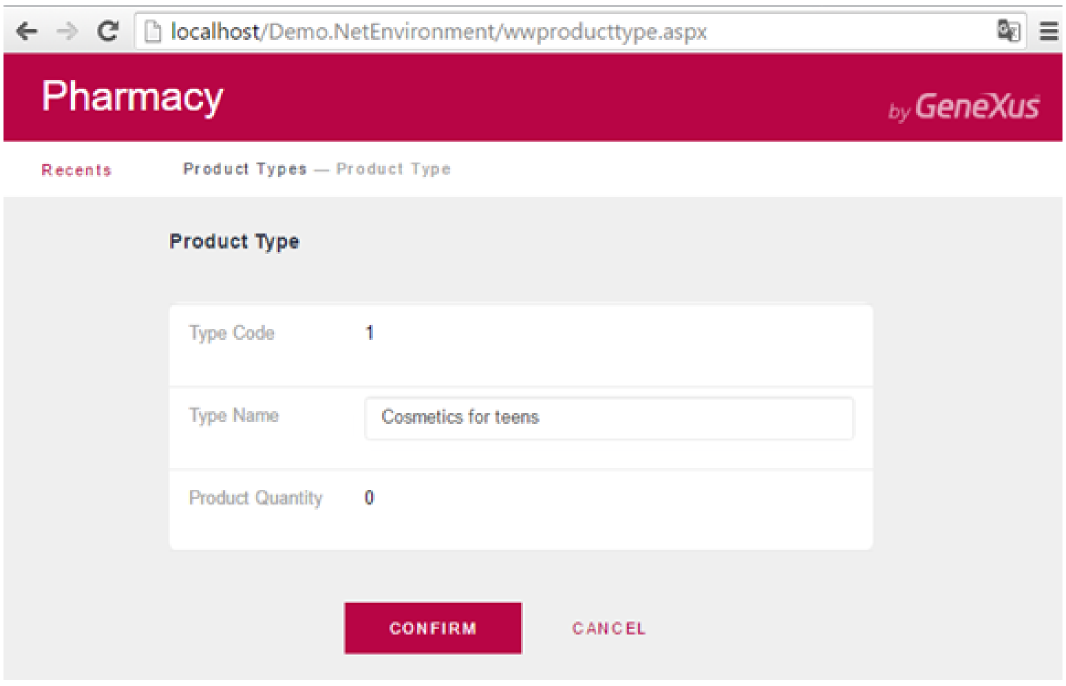
Pharmacy by GeneXus

Recents | Product Types — Product Type

Product Type

Type Code	1
Type Name	Cosmetics
Product Quantity	0

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:


localhost/Demo.NetEnvironment/wwproducttype.aspx

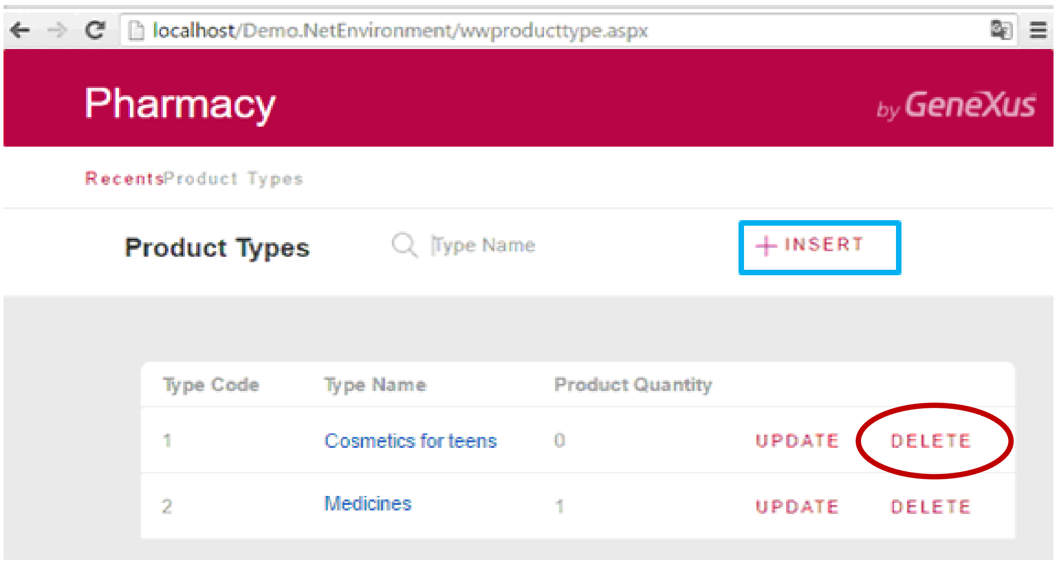
Pharmacy by GeneXus

Recents | Product Types — Product Type

Product Type

Type Code	1
Type Name	Cosmetics for teens
Product Quantity	0

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


localhost/Demo.NetEnvironment/wwproducttype.aspx

Pharmacy by GeneXus

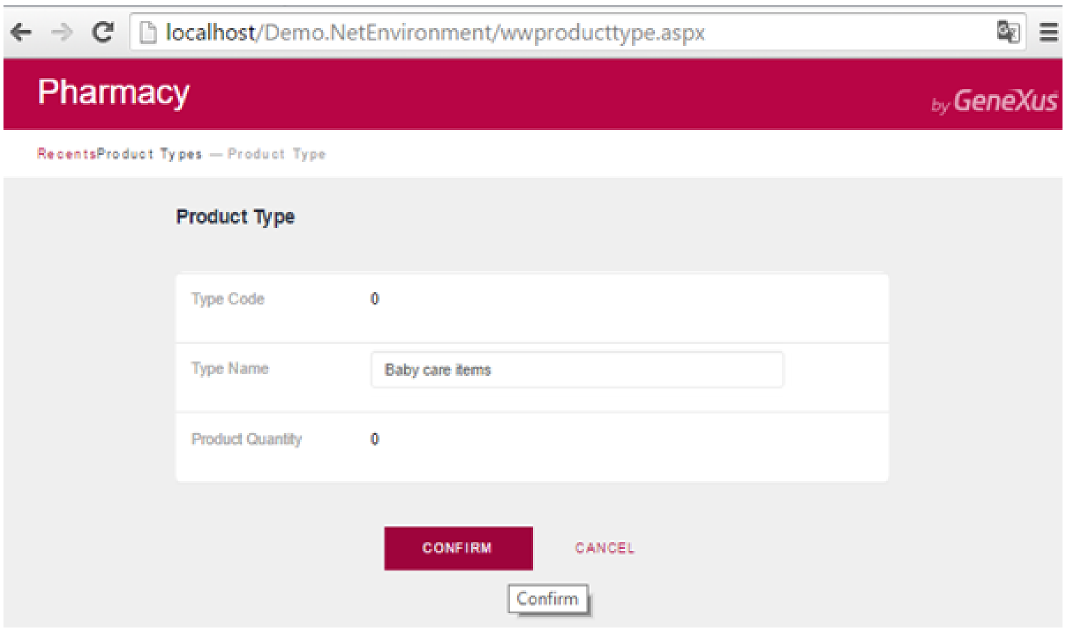
RecentsProduct Types

Product Types — Type Name — + INSERT

Type Code	Type Name	Product Quantity		
1	Cosmetics for teens	0	UPDATE	DELETE
2	Medicines	1	UPDATE	DELETE

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 it, 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):


localhost/Demo.NetEnvironment/wwproducttype.aspx

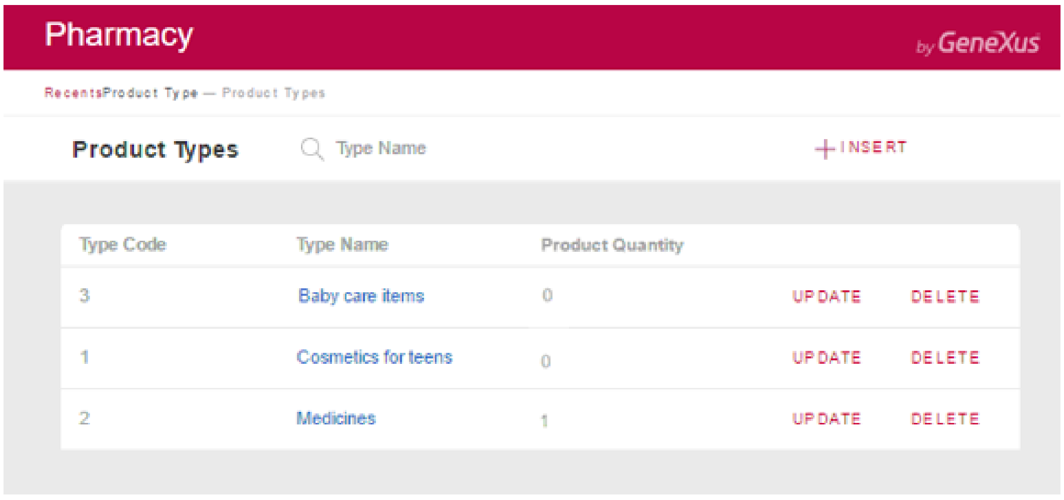
Pharmacy by GeneXus

RecentsProduct Types — Product Type

Product Type

Type Code	0
Type Name	Baby care items
Product Quantity	0

Once again, after the insertion, the application returns to the *Work With Product Types* page:



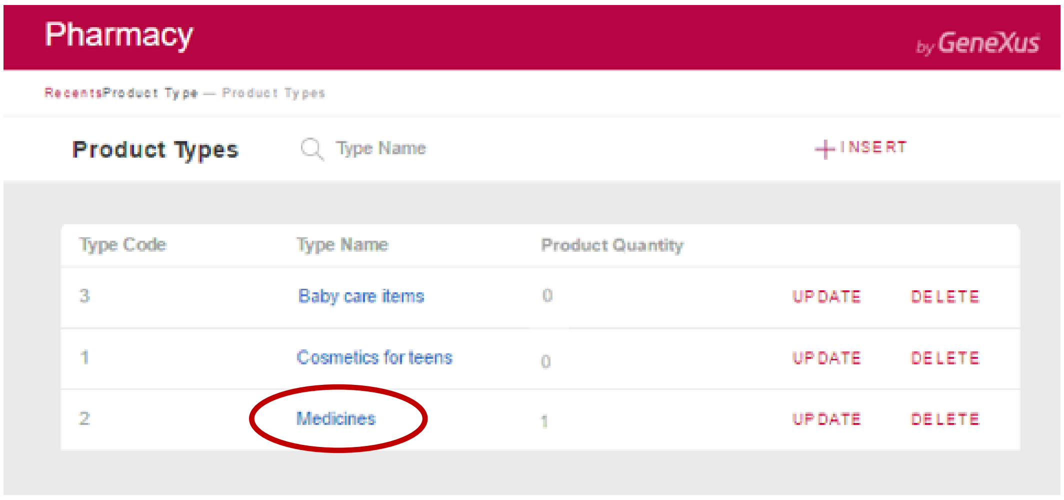
Pharmacy by GeneXus

RecentsProduct Type — Product Types

Product Types   Type Name   + INSERT

Type Code	Type Name	Product Quantity		
3	Baby care items	0	UPDATE	DELETE
1	Cosmetics for teens	0	UPDATE	DELETE
2	Medicines	1	UPDATE	DELETE

Now note that each product type name has a link. Click on the product type: Medicines.



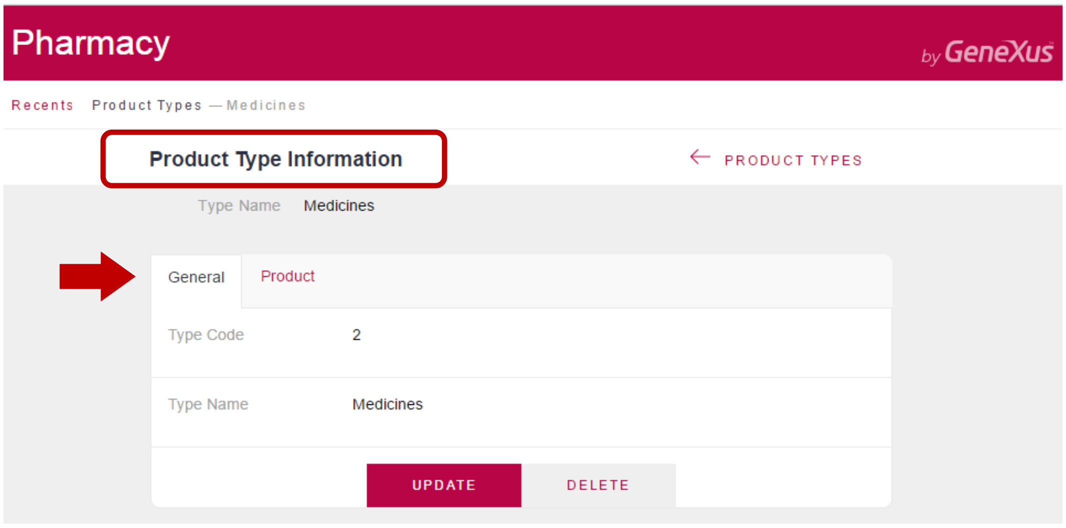
Pharmacy by GeneXus

RecentsProduct Type — Product Types

Product Types   Type Name   + INSERT

Type Code	Type Name	Product Quantity		
3	Baby care items	0	UPDATE	DELETE
1	Cosmetics for teens	0	UPDATE	DELETE
2	Medicines	1	UPDATE	DELETE

As you can see below, all the details of the selected product type are displayed in a first tab, and another tab shows the list of products that belong to that product type.


Pharmacy by GeneXus

Recents Product Types — Medicines

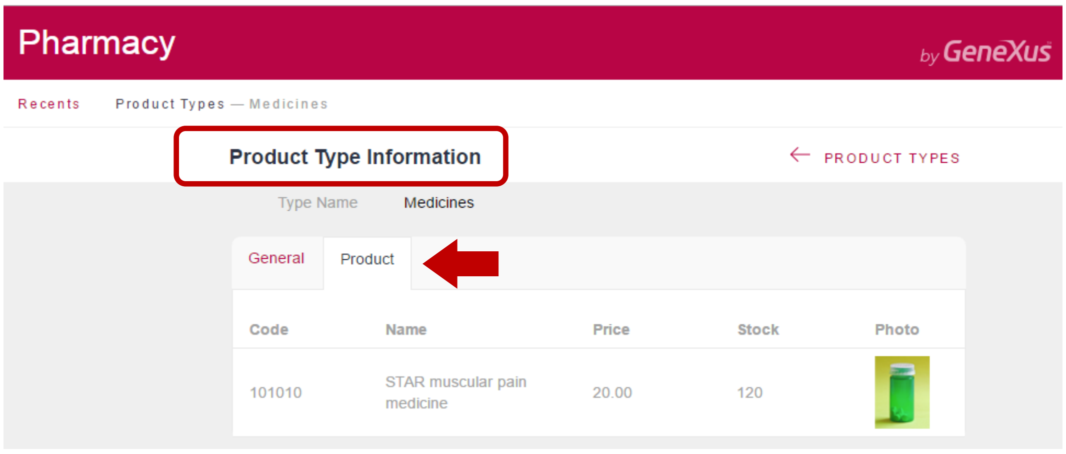
Product Type Information ← PRODUCT TYPES

Type Name Medicines

General | Product

Type Code	2
Type Name	Medicines

UPDATE DELETE


Pharmacy by GeneXus

Recents Product Types — Medicines

Product Type Information ← PRODUCT TYPES

Type Name Medicines

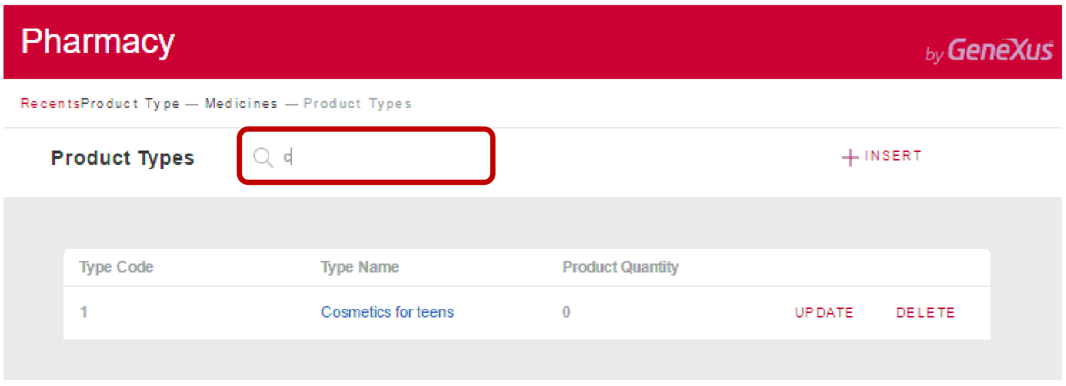
General | Product

Code	Name	Price	Stock	Photo
101010	STAR muscular pain medicine	20.00	120	

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 kind, more tabs would have been generated in order to show each list of data related to the product type.

Now, go back to the *Work With Product Types* page, by clicking on the corresponding link in the Recents section provided at the top left of the page.

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:


Pharmacy by GeneXus

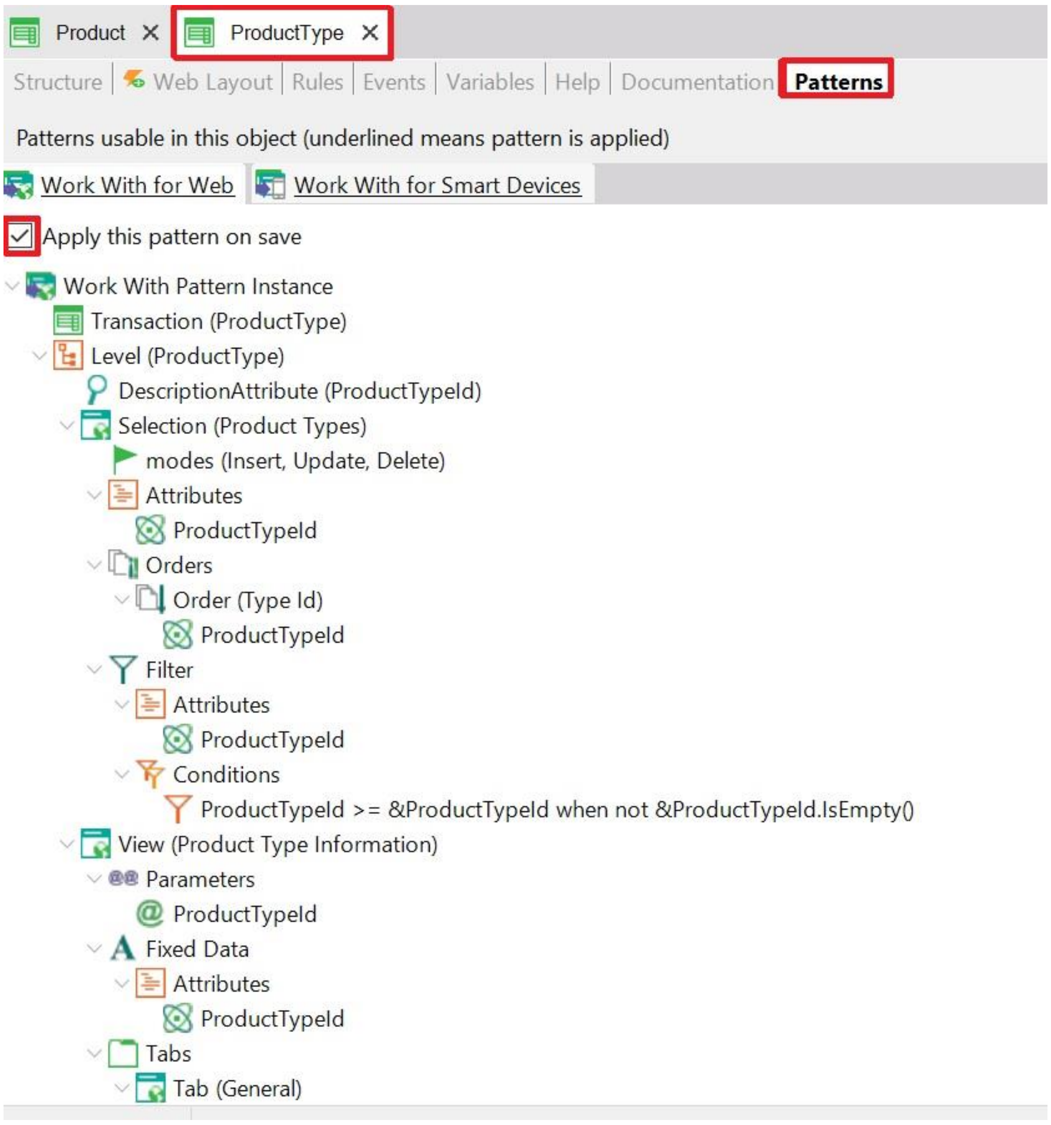
RecentsProduct Type — Medicines — Product Types

Product Types c + INSERT

Type Code	Type Name	Product Quantity		
1	Cosmetics for teens	0	UPDATE	DELETE

Go back to GeneXus. 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:



Patterns usable in this object (underlined means pattern is applied)

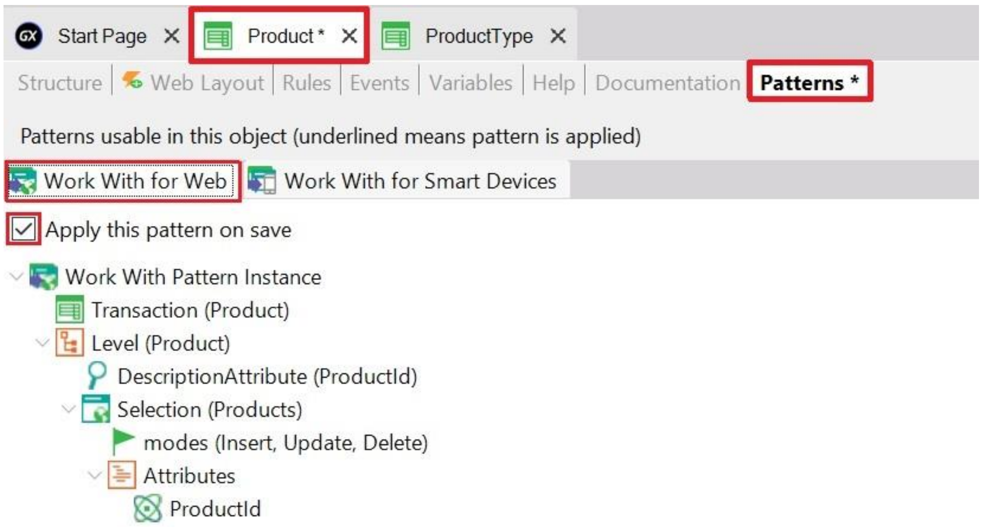
Work With for Web | Work With for Smart Devices

☑ Apply this pattern on save

```
Work With Pattern Instance
  Transaction (ProductType)
  Level (ProductType)
    DescriptionAttribute (ProductTypeId)
    Selection (Product Types)
      modes (Insert, Update, Delete)
      Attributes
        ProductTypeId
      Orders
        Order (Type Id)
          ProductTypeId
      Filter
        Attributes
          ProductTypeId
        Conditions
          ProductTypeId >= &ProductTypeId when not &ProductTypeId.IsEmpty()
    View (Product Type Information)
      Parameters
        ProductTypeId
      Fixed Data
        Attributes
          ProductTypeId
      Tabs
        Tab (General)
```

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 for Web* tab; then, you have to check the option **Apply this pattern on save** and save:


Start Page | Product * | ProductType

Structure | Web Layout | Rules | Events | Variables | Help | Documentation | **Patterns ***

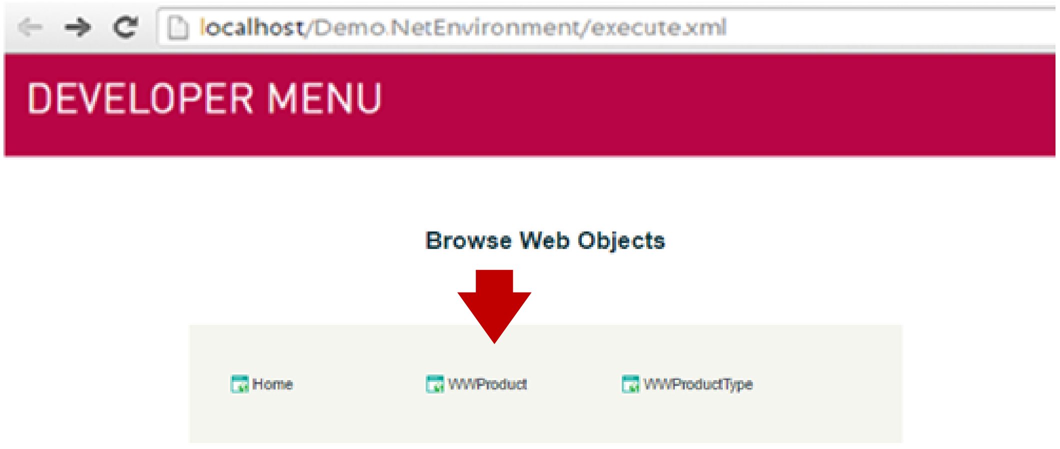
Patterns usable in this object (underlined means pattern is applied)

Work With for Web | Work With for Smart Devices

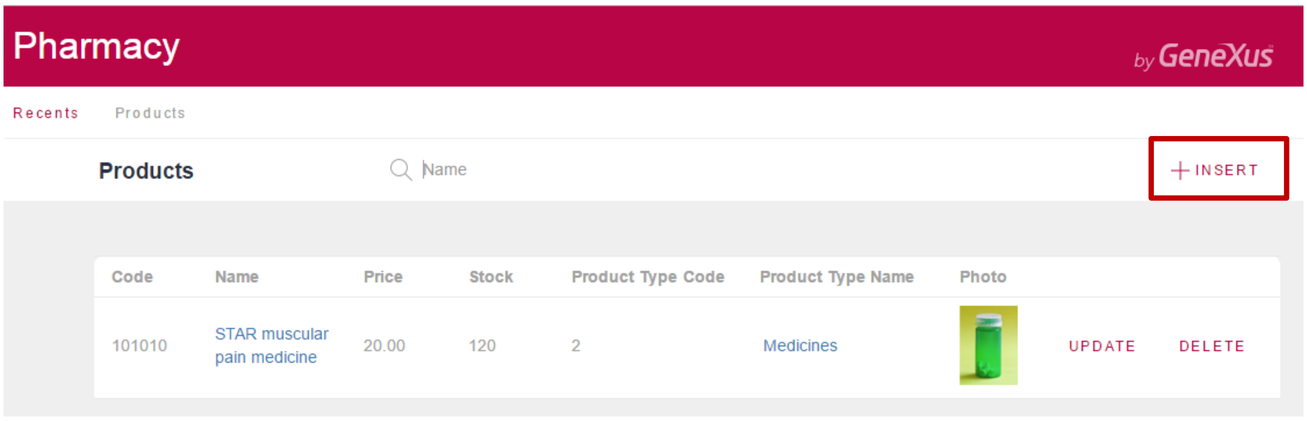
✓ Apply this pattern on save

```
Work With Pattern Instance
    Transaction (Product)
    Level (Product)
        DescriptionAttribute (ProductId)
        Selection (Products)
            modes (Insert, Update, Delete)
            Attributes
                ProductId
```

Press F5. GeneXus proceeds generating the necessary programs and executing the application with the changes. Then, run the *Work With Product* page:

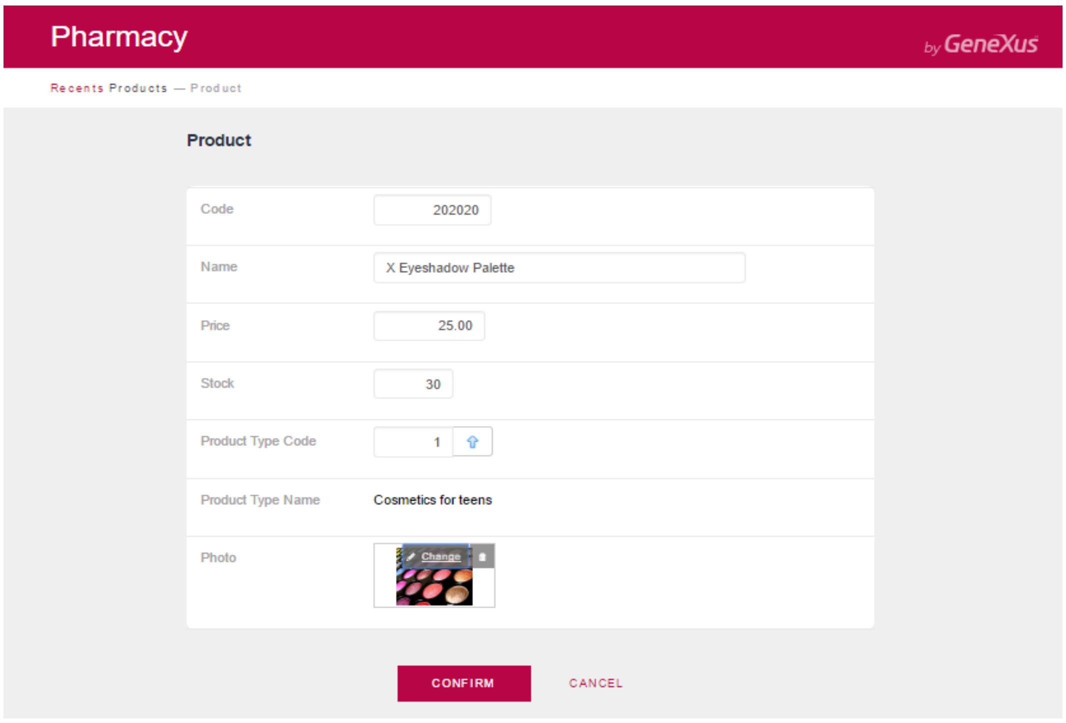


You can see the same query features that you already saw for the *Work With Product Types* page:



Code	Name	Price	Stock	Product Type Code	Product Type Name	Photo		
101010	STAR muscular pain medicine	20.00	120	2	Medicines		UPDATE	DELETE

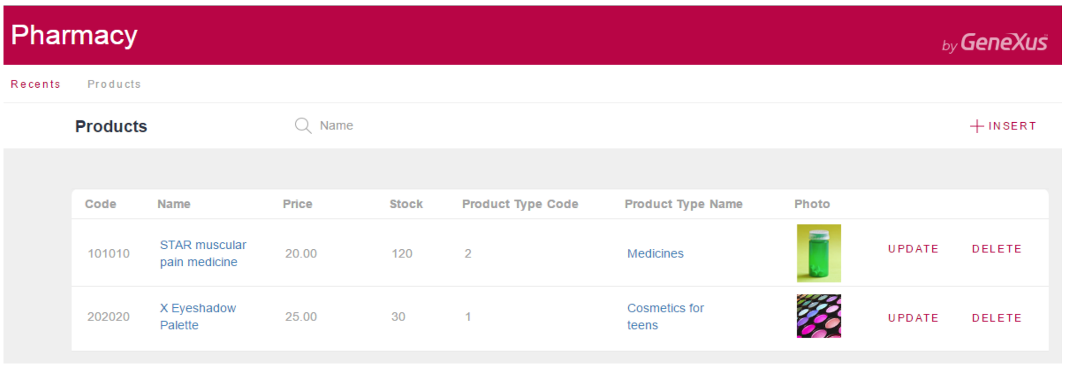
Let's insert a new product:


Pharmacy
by GeneXus
Recents Products — Product

Product

Code	202020
Name	X Eyeshadow Palette
Price	25.00
Stock	30
Product Type Code	1
Product Type Name	Cosmetics for teens
Photo	

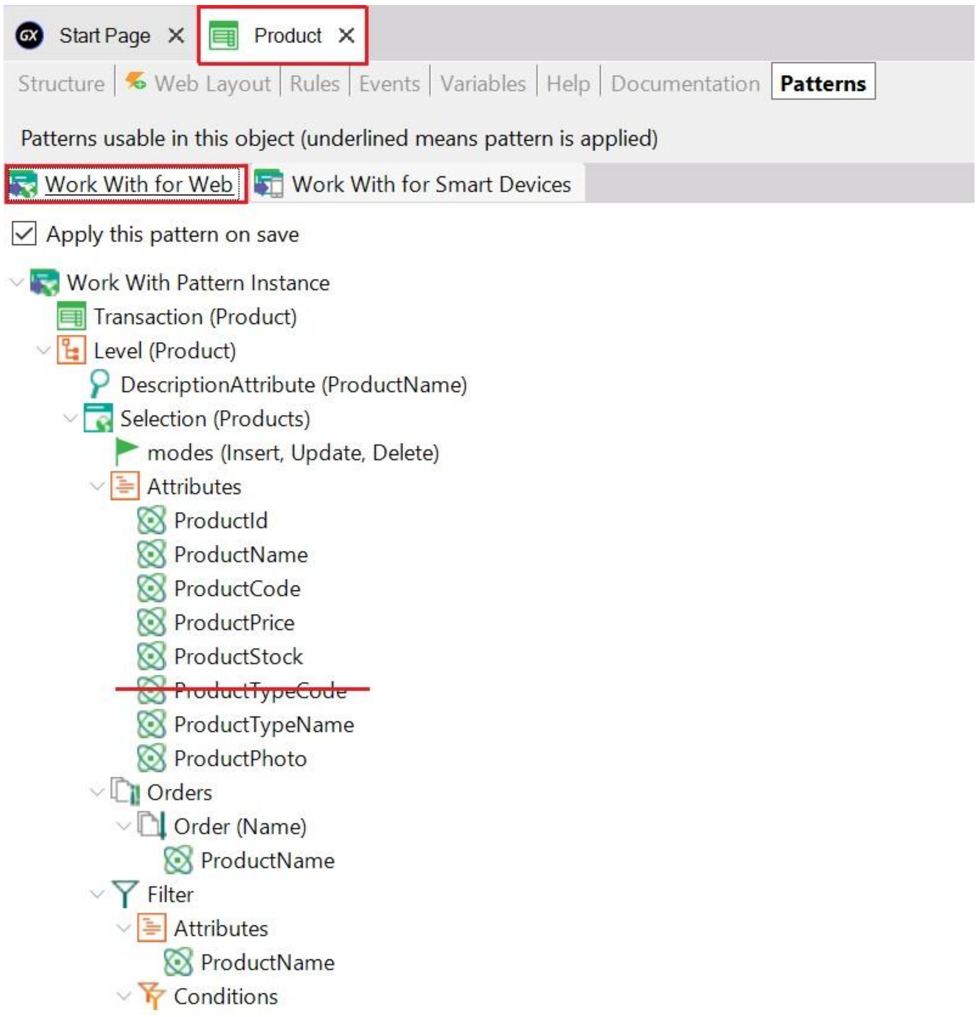
After the confirmation, the application returns to the *Work With Product* page:


Pharmacy
by GeneXus
Recents Products

Products Name + INSERT

Code	Name	Price	Stock	Product Type Code	Product Type Name	Photo		
101010	STAR muscular pain medicine	20.00	120	2	Medicines		UPDATE	DELETE
202020	X Eyeshadow Palette	25.00	30	1	Cosmetics for teens		UPDATE	DELETE

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*:



Start Page ✕ | Product ✕

Structure | Web Layout | Rules | Events | Variables | Help | Documentation | Patterns

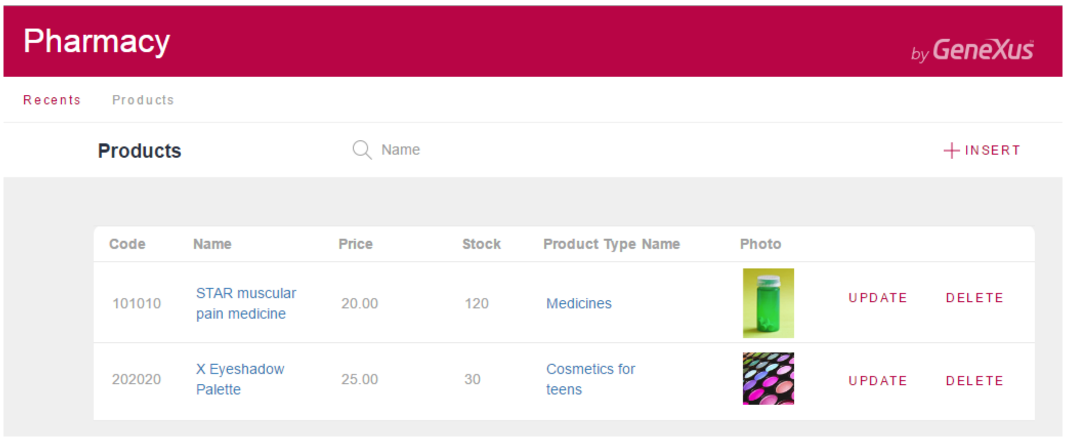
Patterns usable in this object (underlined means pattern is applied)

Work With for Web | Work With for Smart Devices

☑ Apply this pattern on save

```
Work With Pattern Instance
  Transaction (Product)
  Level (Product)
    DescriptionAttribute (ProductName)
    Selection (Products)
      modes (Insert, Update, Delete)
      Attributes
        ProductId
        ProductName
        ProductCode
        ProductPrice
        ProductStock
        ~~ProductTypeCode~~
        ProductTypeName
        ProductPhoto
      Orders
        Order (Name)
          ProductName
      Filter
        Attributes
          ProductName
        Conditions
```

Press F5 and GeneXus proceeds saving, generating only the necessary programs and executing the application with the changes:



Pharmacy — by GeneXus

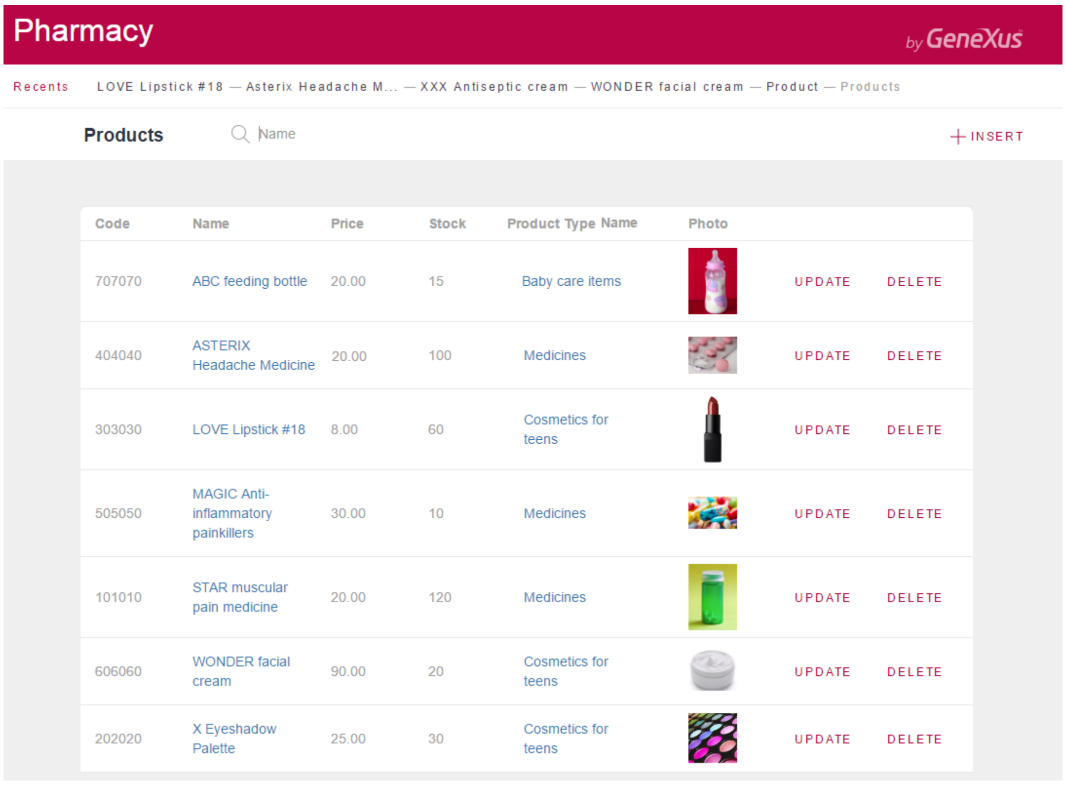
Recents | Products

Products | Name | + INSERT

Code	Name	Price	Stock	Product Type Name	Photo		
101010	STAR muscular pain medicine	20.00	120	Medicines		UPDATE	DELETE
202020	X Eyeshadow Palette	25.00	30	Cosmetics for teens		UPDATE	DELETE

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

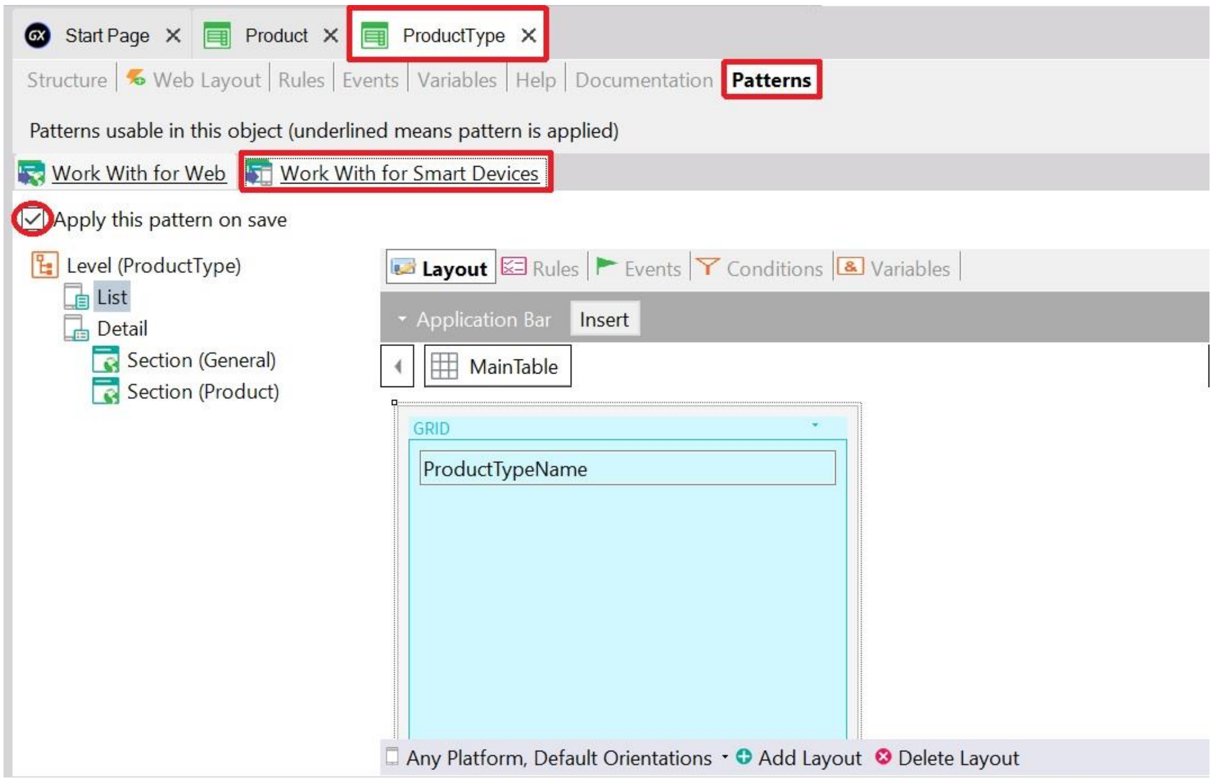
Below is the Work With Products dialog that lists all the products that have been added:


Pharmacy
by GeneXus
Recents
LOVE Lipstick #18 — Asterix Headache M... — XXX Antiseptic cream — WONDER facial cream — Product — Products
Products
Name
INSERT

Code	Name	Price	Stock	Product Type Name	Photo		
707070	ABC feeding bottle	20.00	15	Baby care items		UPDATE	DELETE
404040	ASTERIX Headache Medicine	20.00	100	Medicines		UPDATE	DELETE
303030	LOVE Lipstick #18	8.00	60	Cosmetics for teens		UPDATE	DELETE
505050	MAGIC Anti-inflammatory painkillers	30.00	10	Medicines		UPDATE	DELETE
101010	STAR muscular pain medicine	20.00	120	Medicines		UPDATE	DELETE
606060	WONDER facial cream	90.00	20	Cosmetics for teens		UPDATE	DELETE
202020	X Eyeshadow Palette	25.00	30	Cosmetics for teens		UPDATE	DELETE

Now let's pay attention to the ***Work With for Smart Devices*** tab offered for each Transaction.

Let's apply it to the *ProductType* Transaction:



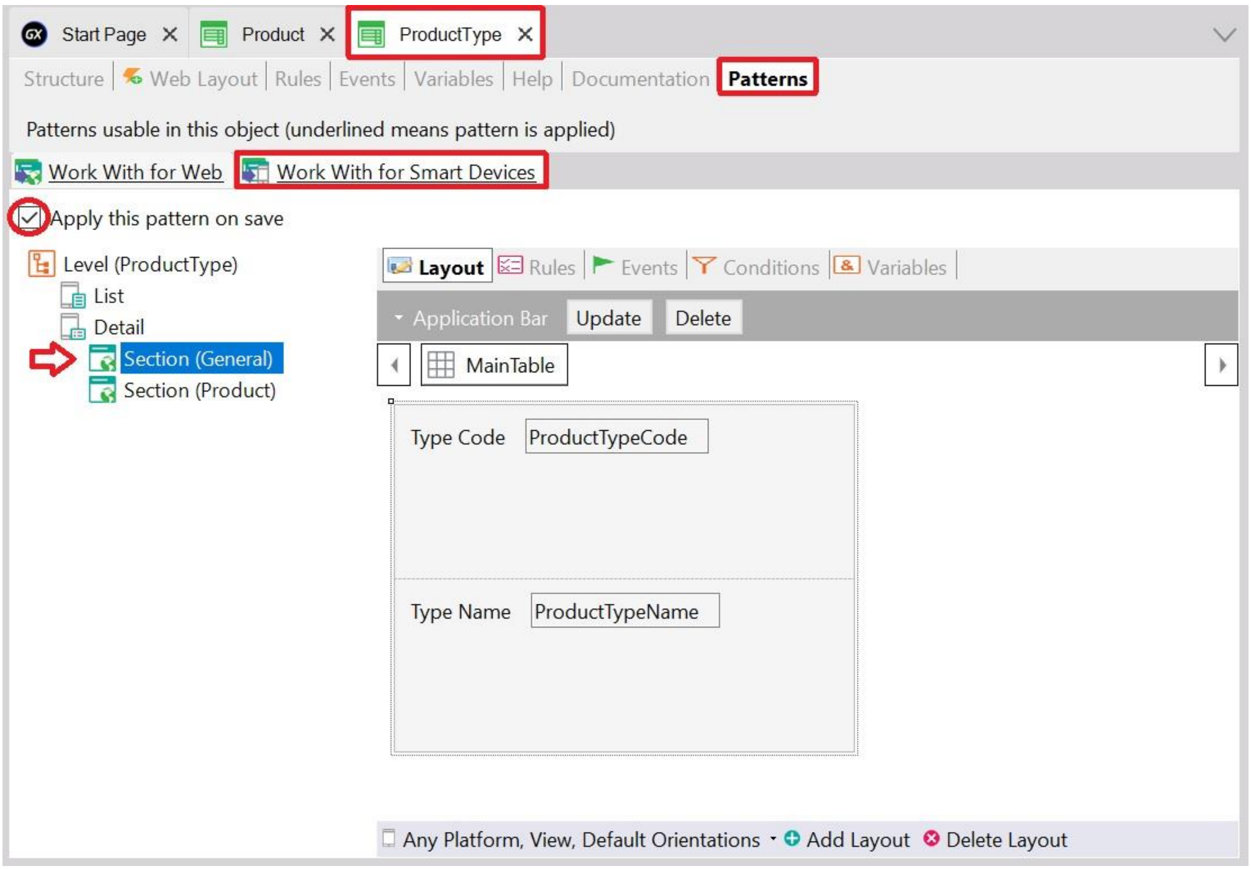
Note that under the main node (*ProductType*), there is the **List** node. If you click on it, you can see at the right window, a grid that has the *ProductTypeName* attribute inserted in it.

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

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

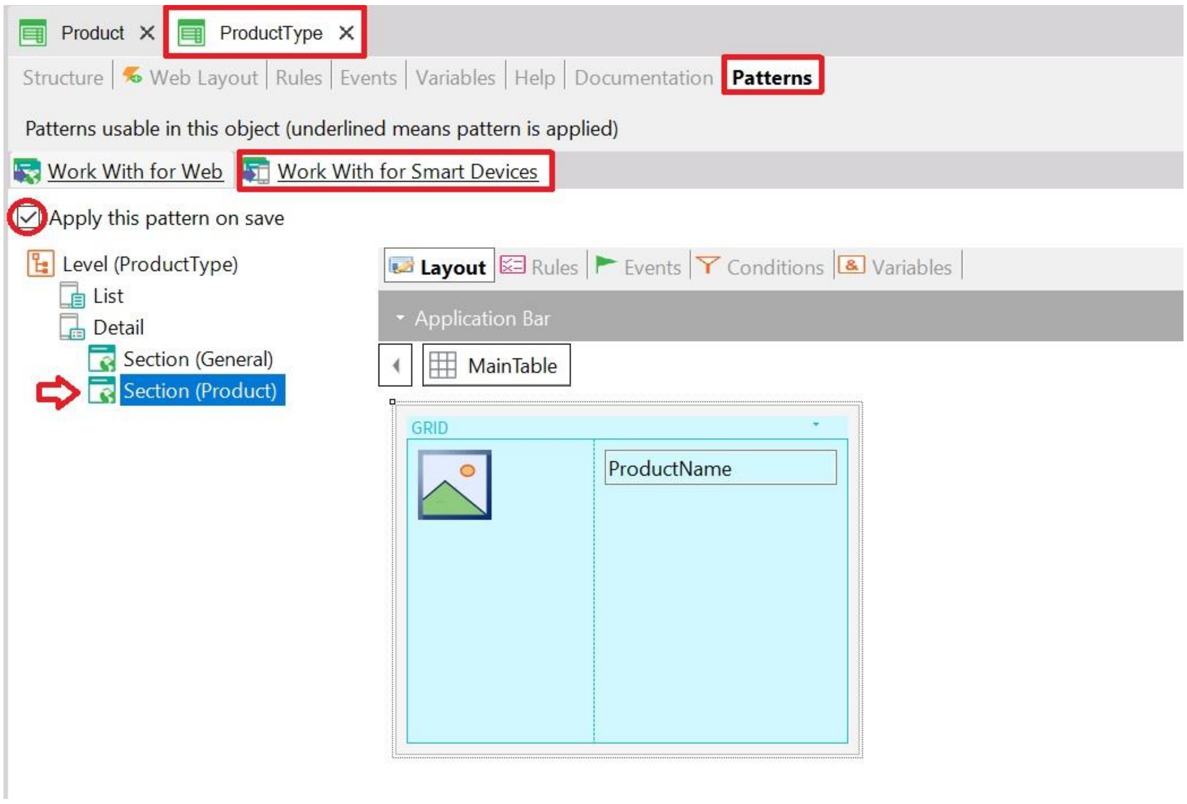
The **Detail** node is composed of two sections: **General** and **Product**.

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 inside a grid all the products that belong to the product type.



Patterns usable in this object (underlined means pattern is applied)

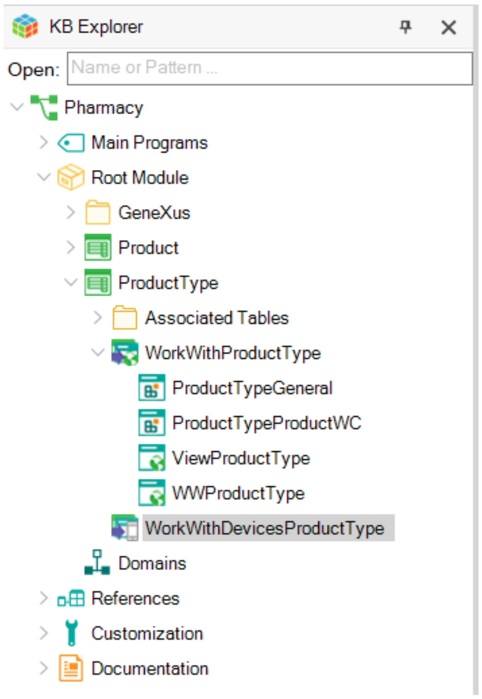
[✓] Apply this pattern on save



Patterns usable in this object (underlined means pattern is applied)

[✓] Apply this pattern on save

After applying this pattern and saving, if you look at the *ProductType* Transaction in the KB Explorer, you can see now a new object called *WorkWithDevicesProductType* under the *ProductType* Transaction:



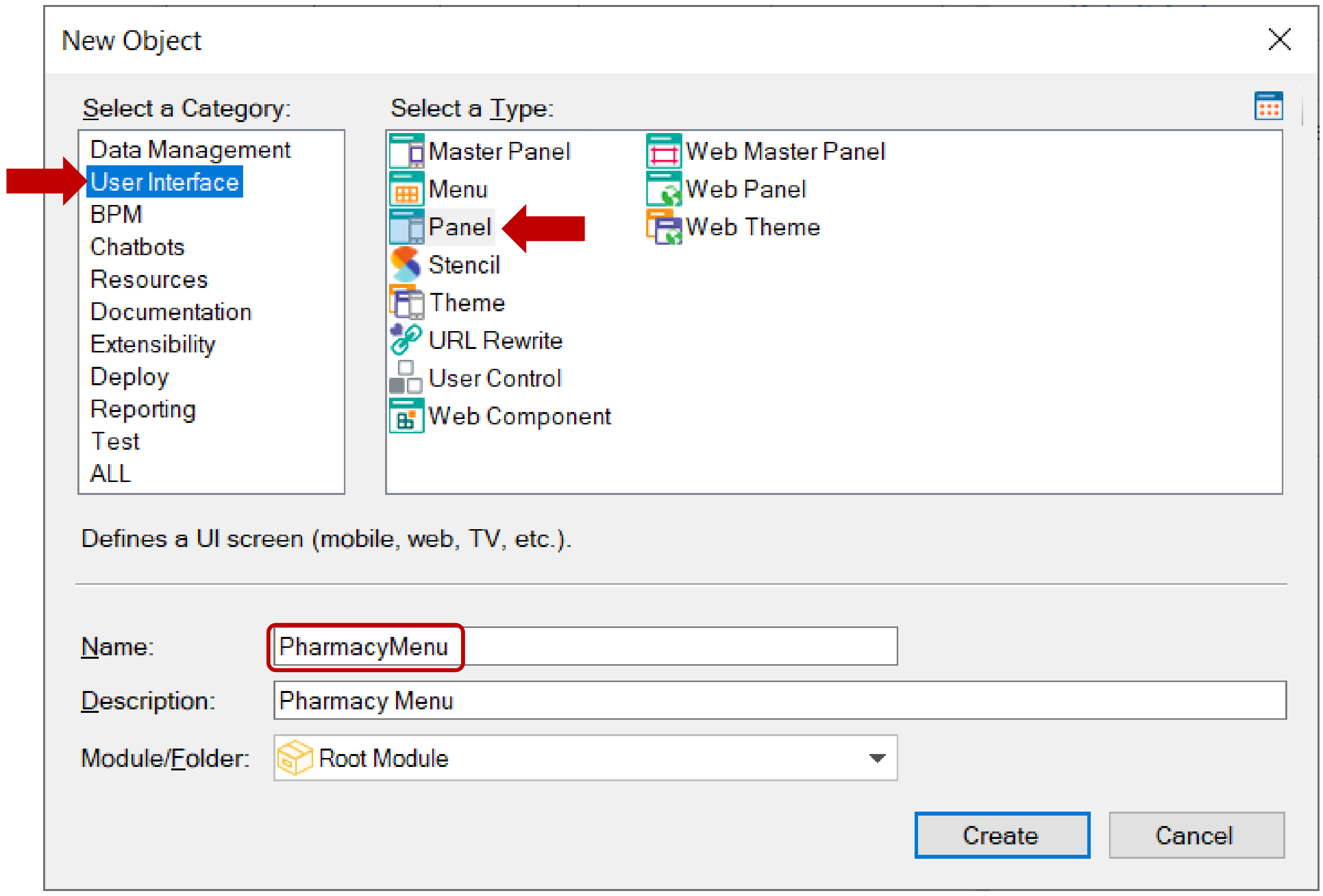
If you look at both objects generated under the *ProductType* Transaction, you can notice that the *WorkWithDevicesProductType* hasn't got other objects under it (because it includes different sections to define the entire implementation inside it).

On the other hand, the *WorkWithProductType*, as you have already seen, 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 seen.

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

The proposal now is to create a Panel object, in order to be the first object of the application to be executed; and it will show an image, so that when the user taps it, the object *WorkWithDevicesProductType* is executed.

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


New Object

Select a Category: Data Management, User Interface, BPM, Chatbots, Resources, Documentation, Extensibility, Deploy, Reporting, Test, ALL

Select a Type: Master Panel, Menu, Panel, Stencil, Theme, URL Rewrite, User Control, Web Component, Web Master Panel, Web Panel, Web Theme

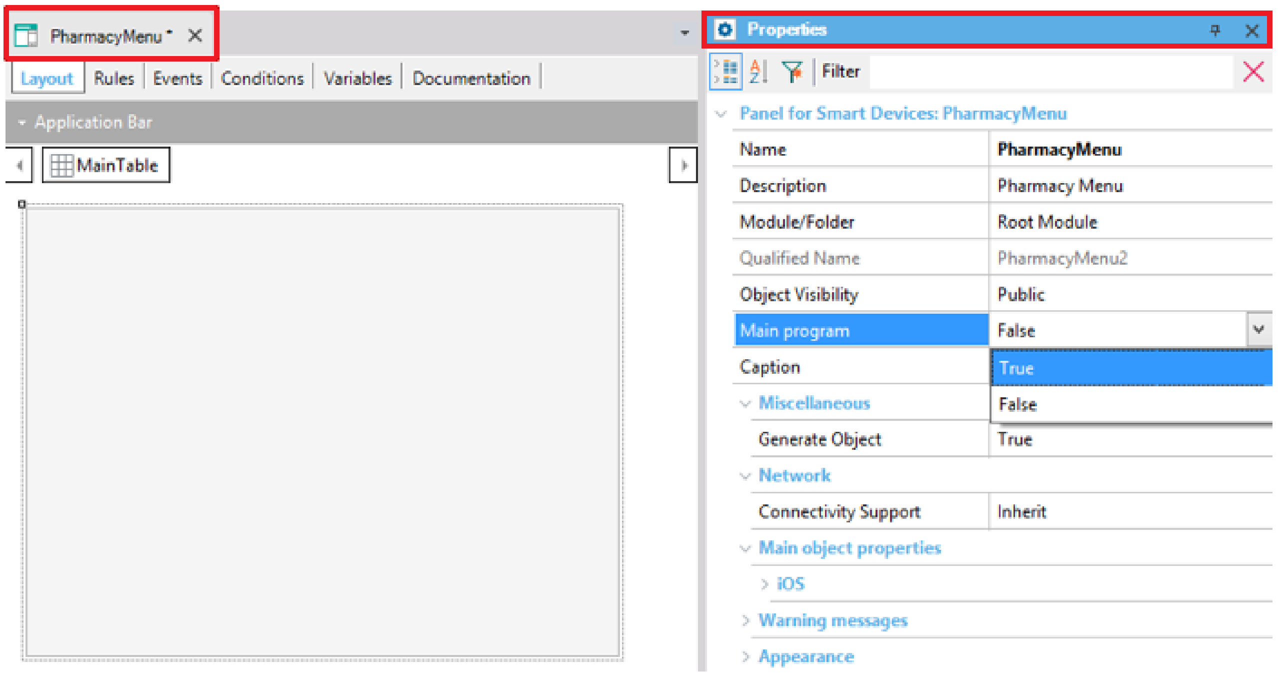
Defines a UI screen (mobile, web, TV, etc.).

Name: PharmacyMenu

Description: Pharmacy Menu

Module/Folder: Root Module

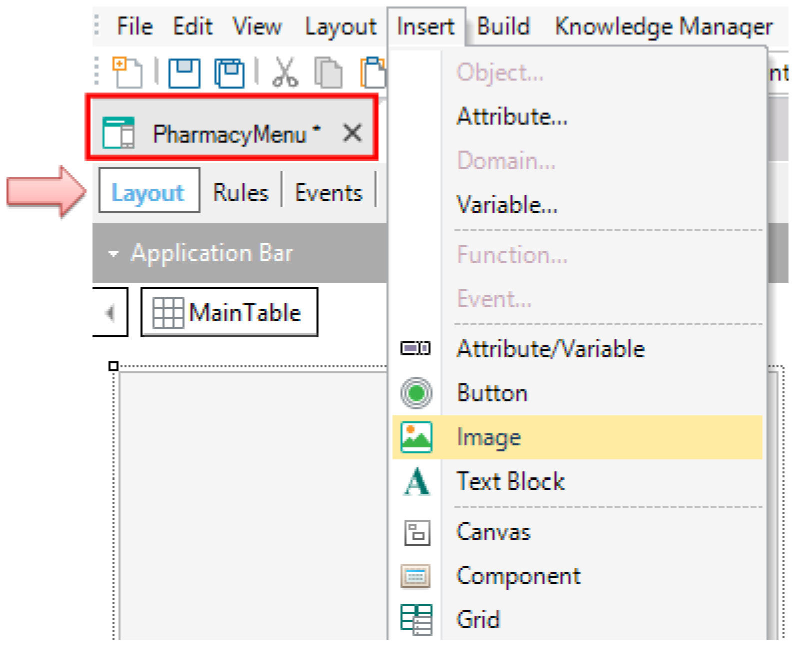
Once you have created the **Panel** (which in this example has been named *PharmacyMenu*), you must configure its property **Main program = True,** so that this object becomes an executable object independent of the Developer Menu (that is to say, it is compilable and executable on its own).



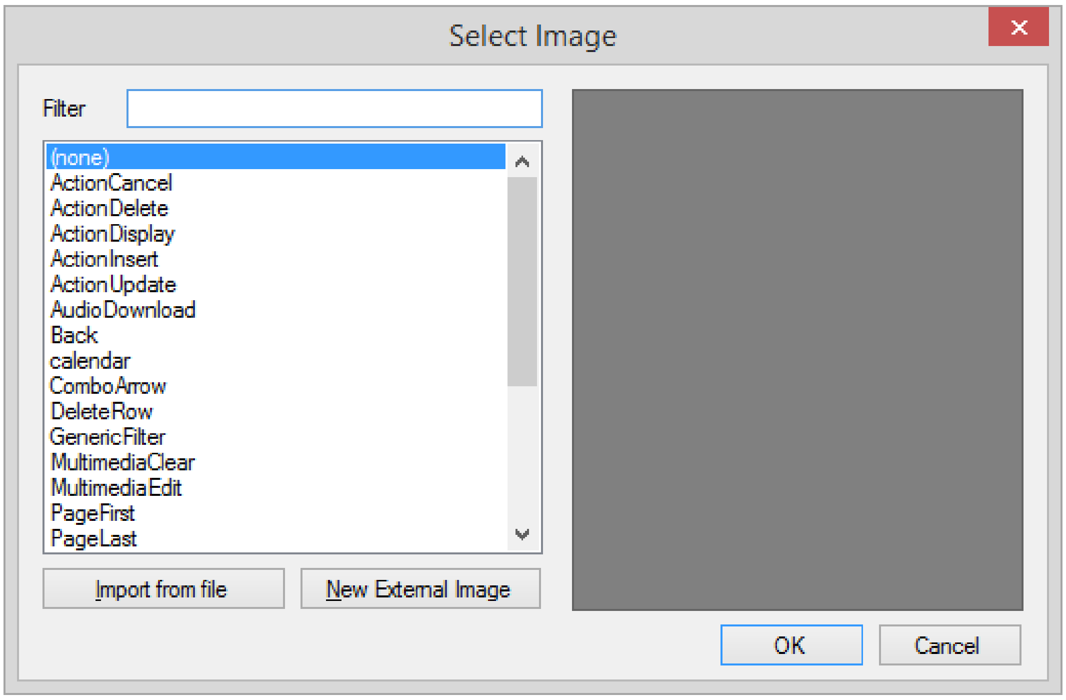
Panel for Smart Devices: PharmacyMenu

Name	PharmacyMenu
Description	Pharmacy Menu
Module/Folder	Root Module
Qualified Name	PharmacyMenu2
Object Visibility	Public
Main program	False (True / False)
Caption	
Miscellaneous	
Generate Object	True
Network	
Connectivity Support	Inherit
Main object properties	
iOS	
Warning messages	
Appearance	

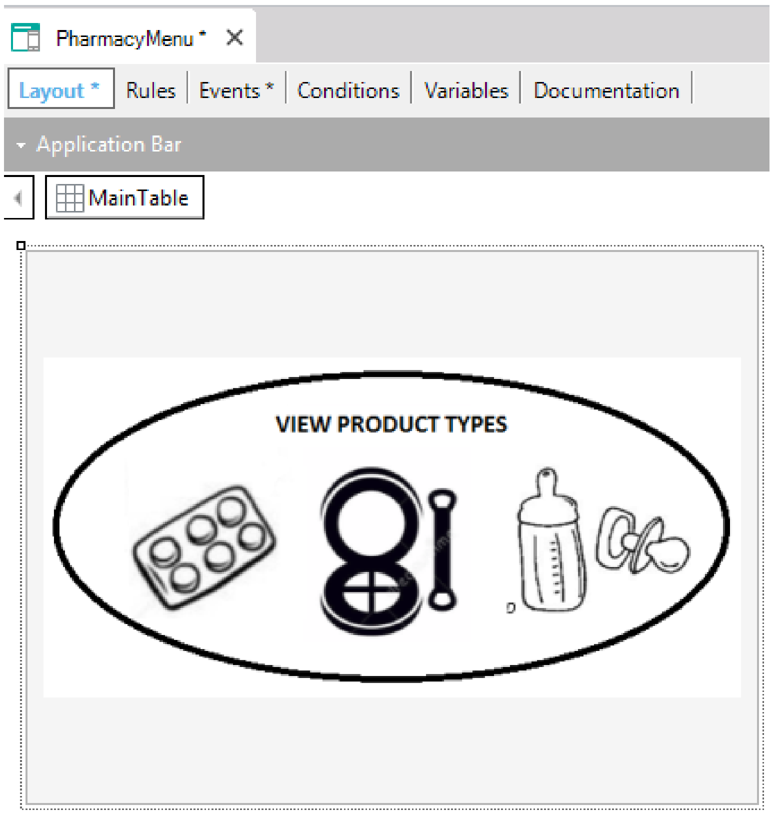
Now insert an image control inside its Layout:



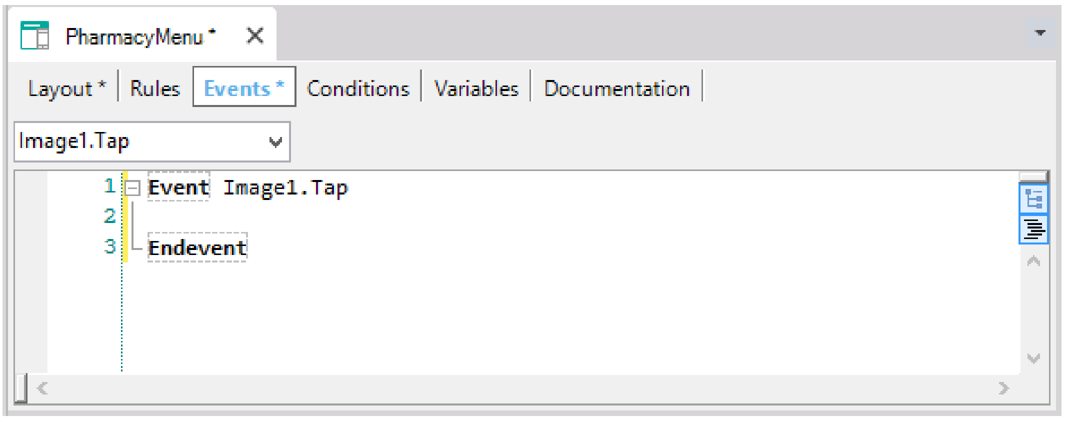
When you select **Insert/Image,** the following window is displayed; if your photo is stored in a file, you must press the button **Import from file**.



After importing the image to the Knowledge Base and inserting it in the Layout of the *PharmacyMenu* **Panel for Smart Devices,** your layout should look something like this:



By double-clicking on the image control, its default associated event is shown:

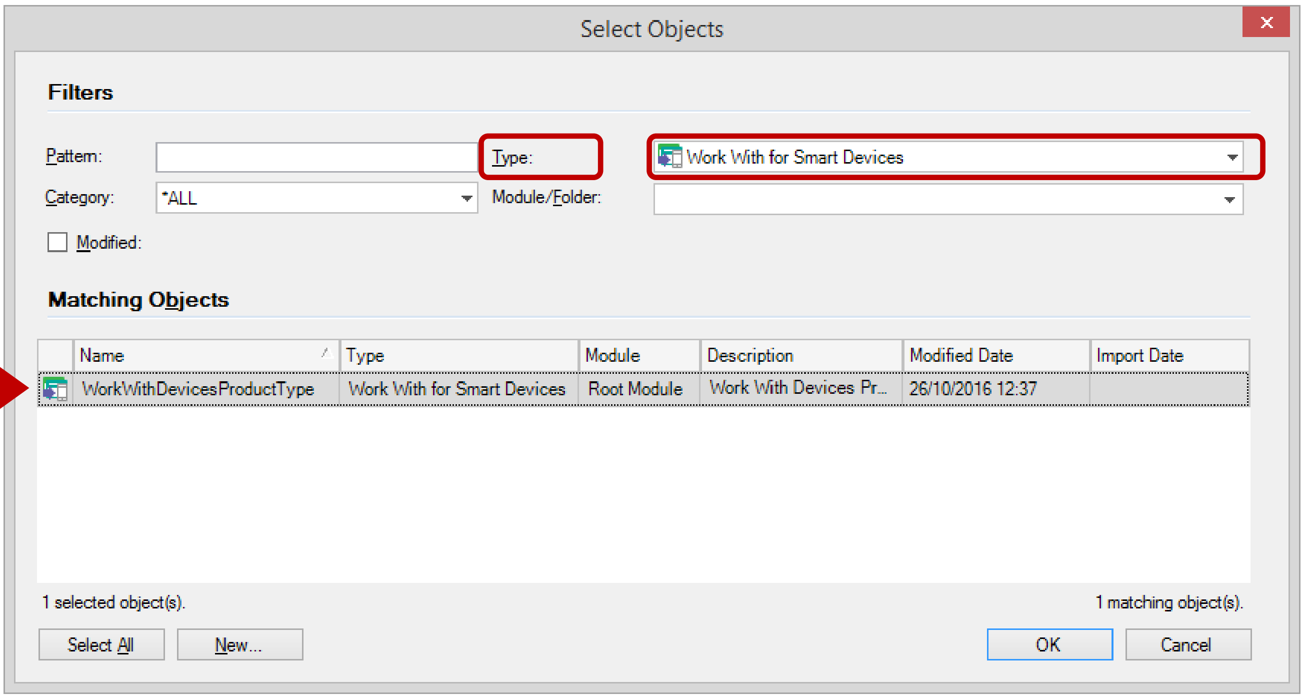


```
Event Image1.Tap

Endevent
```

Inside the Tap event associated to the image, the *WorkWithDevicesProductType* object must be called; but said object is composed of many parts (remember or look again at its nodes). In this particular case the List node of the object must be called, as our objective is to show the list with all the product types the pharmacy offers.

To carry out this, after locating the cursor inside the event, select Insert / Object.

The following dialog is displayed:


Select Objects

Filters

Pattern:		Type:	Work With for Smart Devices
Category:	*ALL	Module/Folder:	

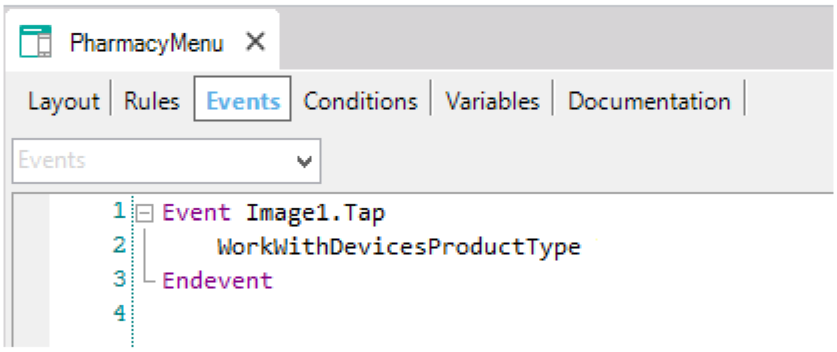
☐ Modified:

Matching Objects

Name	Type	Module	Description	Modified Date	Import Date
WorkWithDevicesProductType	Work With for Smart Devices	Root Module	Work With Devices Pr...	26/10/2016 12:37	

1 selected object(s). 1 matching object(s).

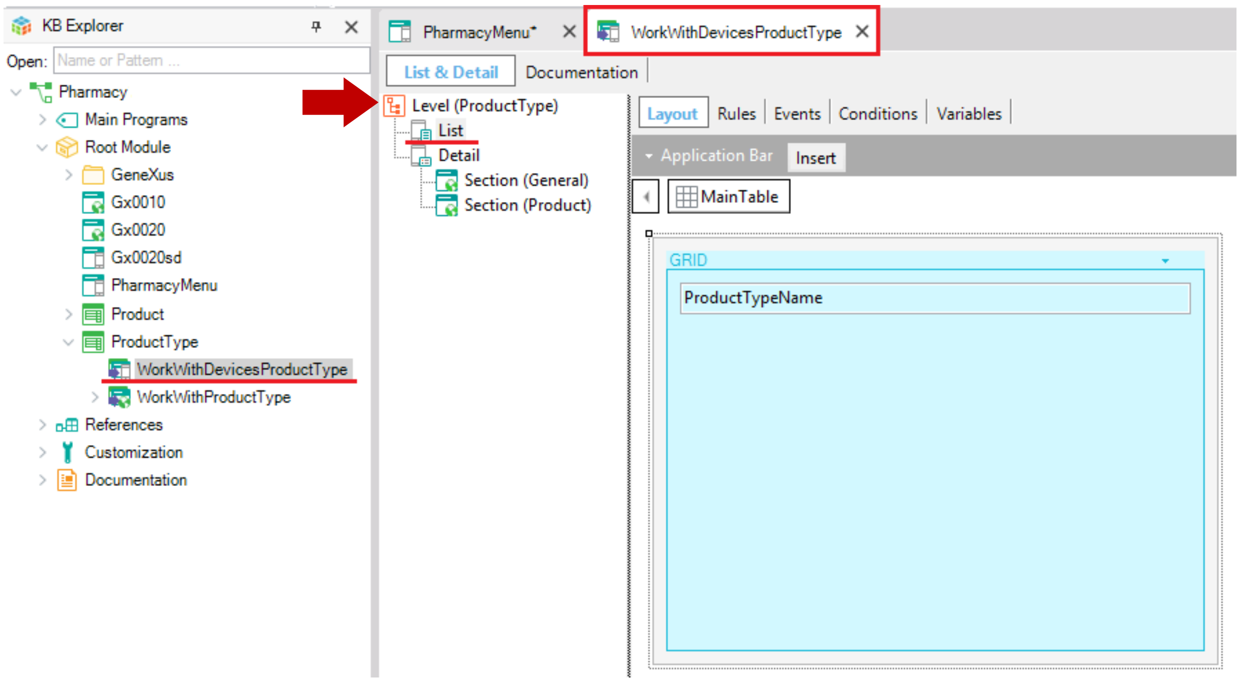
Let's filter as shown above and let's select the *WorkWithDevicesProductType* object:



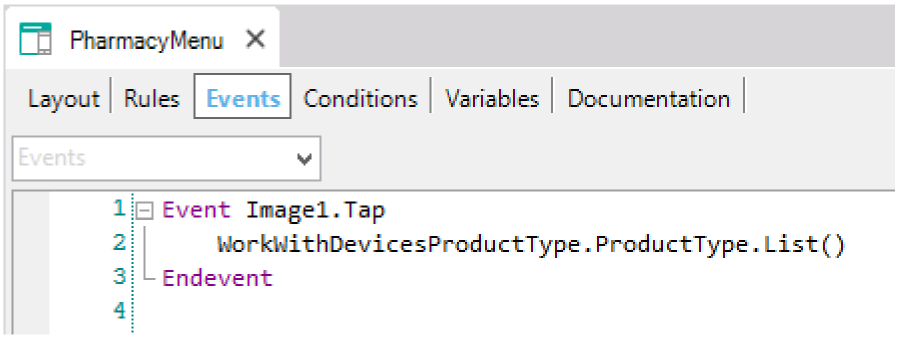
```
Event Image1.Tap
    WorkWithDevicesProductType
Endevent
```

The object's name has been inserted, so now let's complete which component of the object must be executed.

To understand what you have to complete in the code line, let's review the node tree contained inside the *WorkWithDevicesProductType* object:



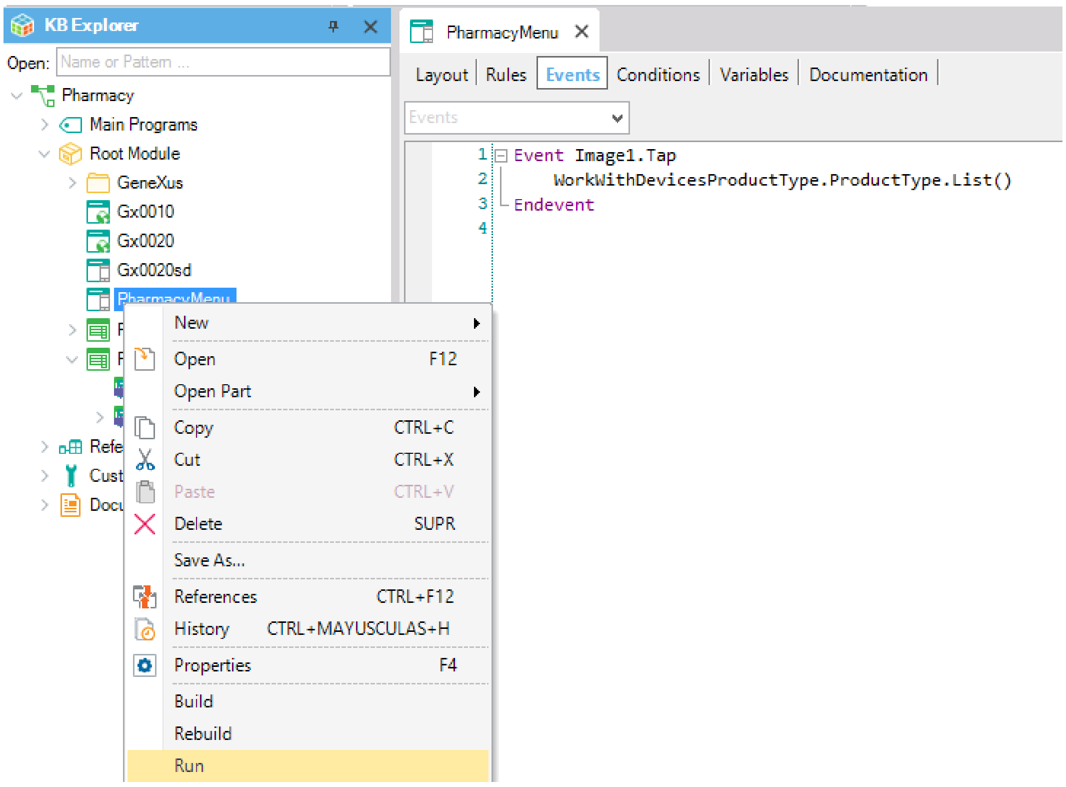
The object has a main node, *ProductType,* and under it you can find the nodes List and Detail respectively, so to call the List node, the complete syntax is:



```
Event Image1.Tap
    WorkWithDevicesProductType.ProductType.List()
Endevent
```

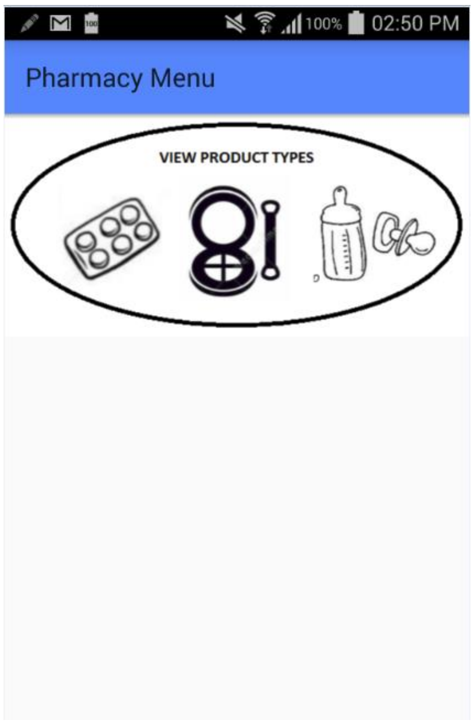
Now everything is defined and ready to run the Mobile and Smart Devices application.

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:**

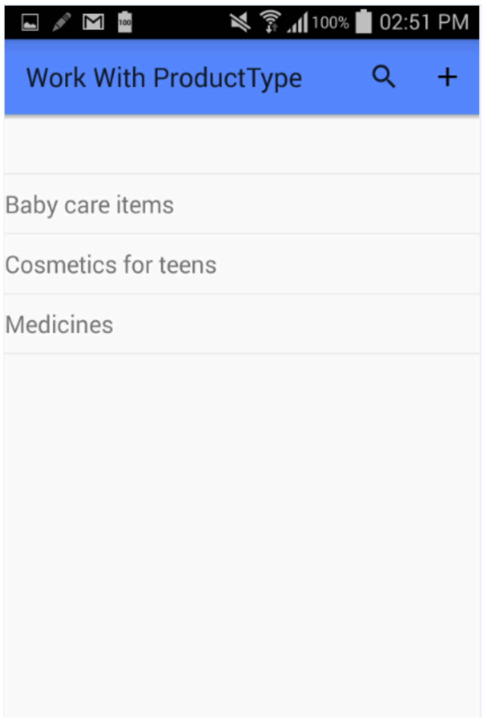


```
Event Image1.Tap
    WorkWithDevicesProductType.ProductType.List()
Endevent
```

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



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



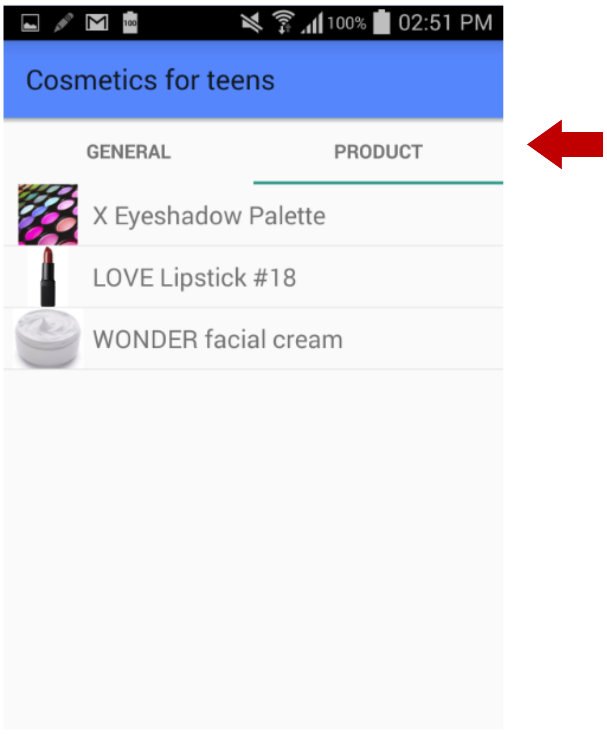
Baby care items

Cosmetics for teens

Medicines

Note that the insert button on the top-right corner can be easily removed, as this application is for end-users, and we do not want them to be able to insert new products; they should only be able to view the different product types, not edit them.

And when you tap each product type (for example on "Cosmetics for teens"), the Detail is shown, along with its two sections GENERAL and PRODUCT:



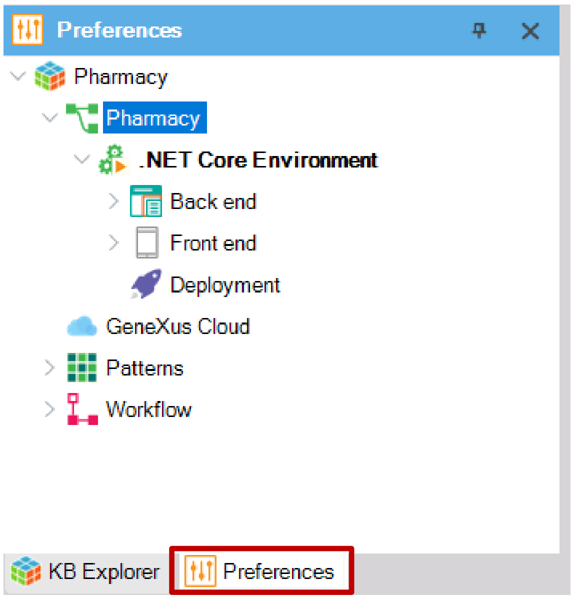
Obviously this is just a very simple demonstration and you can achieve much more sophisticated applications.

WHAT IF YOU WANT TO GENERATE WHAT YOU HAVE DEFINED THIS FAR IN ANOTHER LANGUAGE AND/OR FOR A DIFFERENT DATA BASE?

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 data bases. 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:

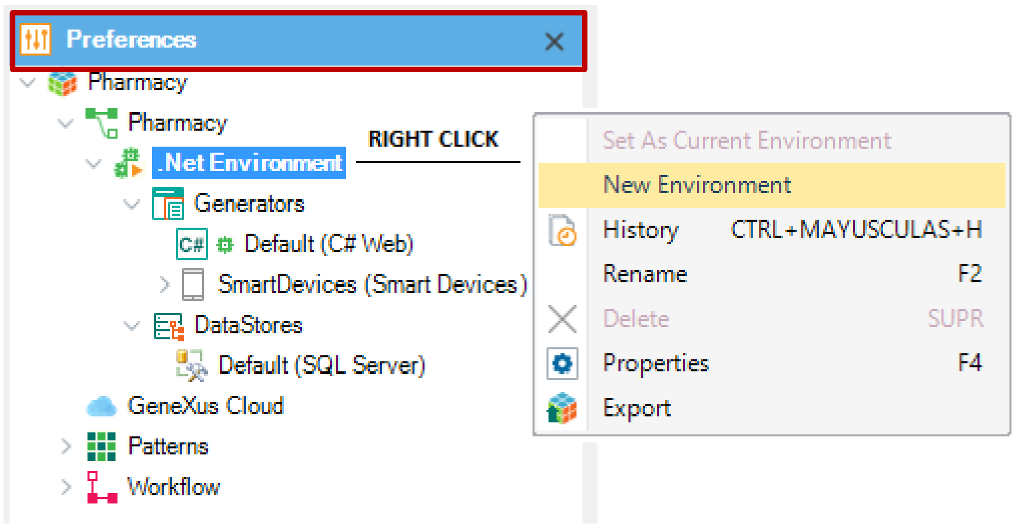


There is only one Environment defined **(.NET Core Environment)**. It was created automatically at Knowledge Base creation time when you chose .NET Core as the generation language. After that, it was completed when you pressed F5 for the first time and you entered the database name to be created and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

It is possible to create more than one execution Environment for the same Knowledge Base. For example, it's 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.

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.



Once the new environment is created, to work with it, you have to right-click 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, you may need to load data from an external database to tables of the database associated with the knowledge base to make an initial load. 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 and Smart Devices apps.

- Designing & fine tuning UIs (User Interfaces)
  - GeneXus offers power to customize the user interface and, because the user experience is extremely important, it offers 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
    - By clicking one button you can deploy your app to production.

- 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 Articial 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 to create specific extensions, that allow developers to leverage different platform languages to create specific solutions and extend GeneXus core capabilities.

- 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 security module (fully integrated into GeneXus), called GeneXus Access Manager (GAM). By just enabling it, it offers to solve authentication and authorization functionalities, for both Web apps and Mobile and Smart Devices apps.

- Modeling and automating business processes
    - GeneXus has a suite of tools that allow for the 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’s directed towards 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, where, using GeneXus, we associate the different objects in each task modelled 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 tool 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 data base, 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:
    - https://training.genexus.com/en/learning/courses/genexus/v17/core
    - If you didn't try 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 to 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