# User Controls in Angular

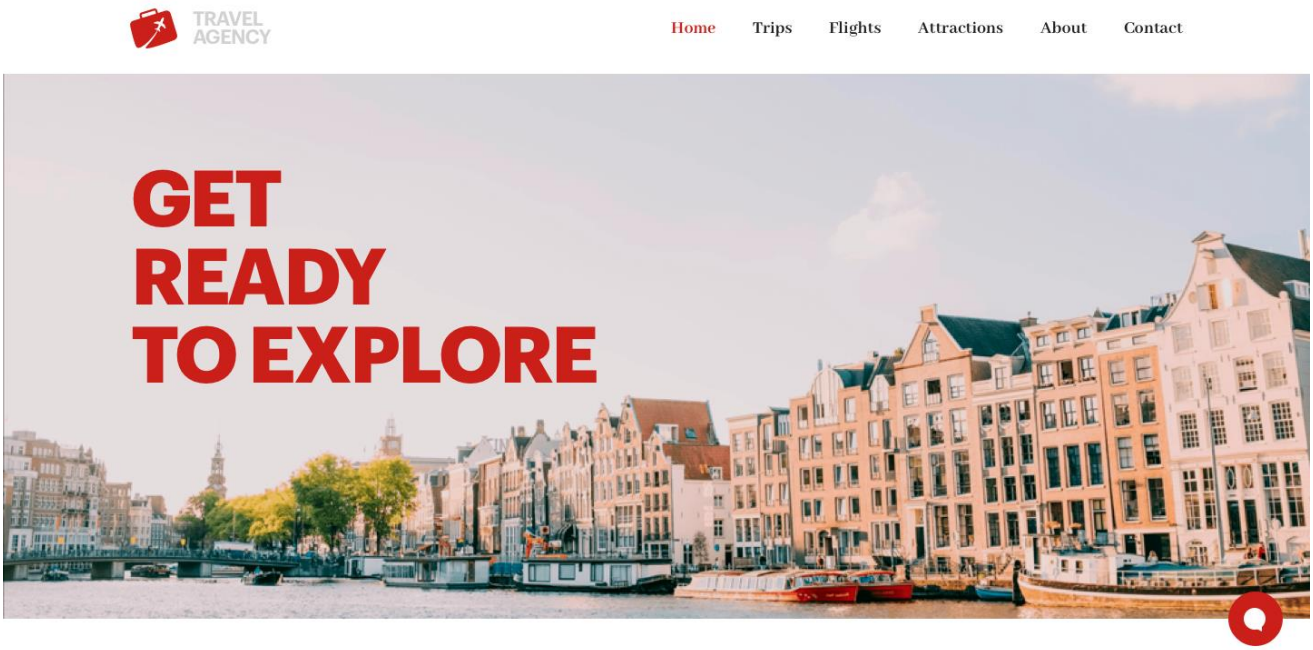
GeneXus™

Previous videos showed several screen controls that help in building the user interface, and the way to enhance the app's design with definitions in a Design System Object, as well as how to import a design already made by a designer in Sketch.

This video shows that, in addition to the predefined screen controls available in the toolbar, it is also possible to create your own controls to enrich the user experience even further.

Importing resources from UI providers

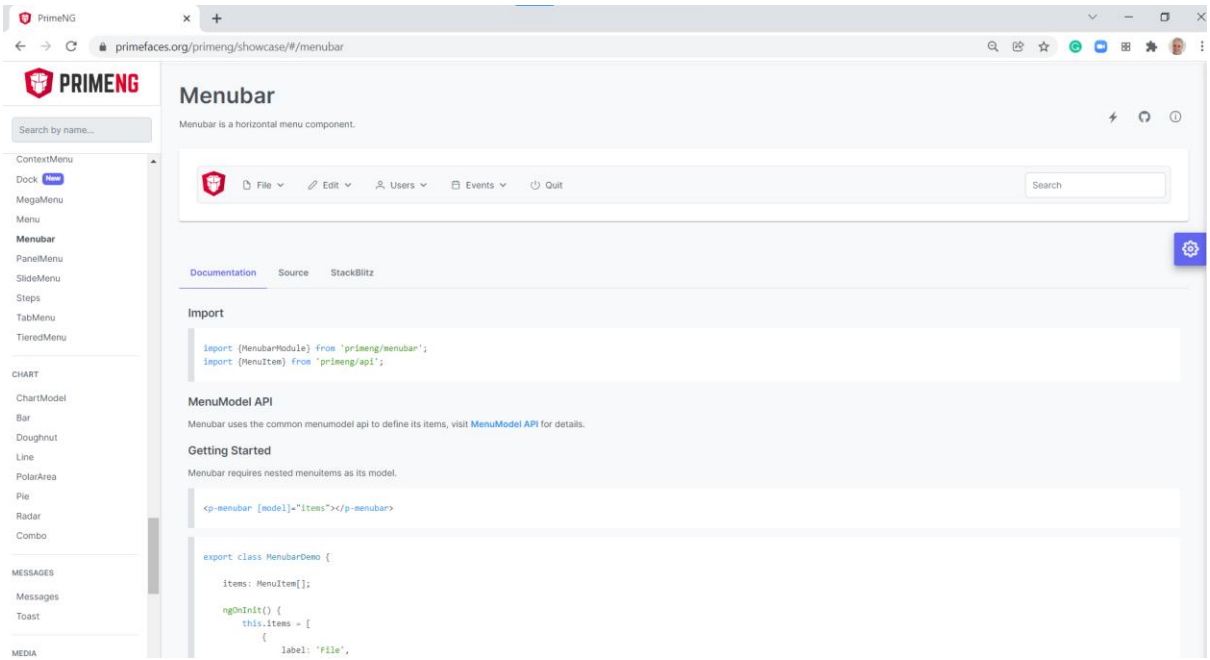

GeneXus allows you to create user controls from controls built by designers, or controls available on platforms of providers of User Interface resources. These may be either native components of the Angular framework –such as PrimeNG, AngularMaterial or Material-UI-, or HTML, CSS and JavaScript resources from generic providers like SemanticUI, VanillaFramework, or Bootstrap component libraries.




In the design received from the designer, there is an option menu in the upper right.

You will now build the menu with a user control provided by PrimeNG.

Defining the User Control to be created

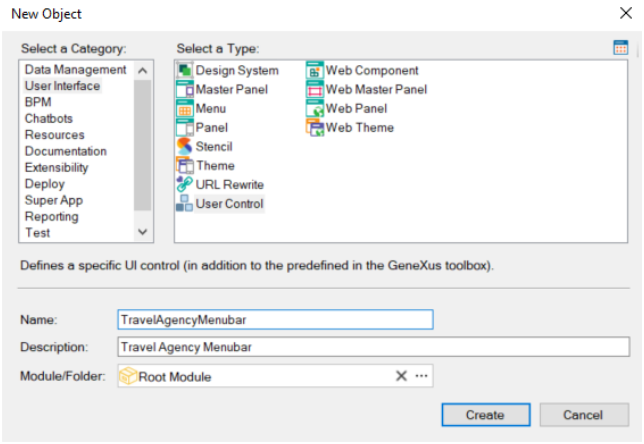


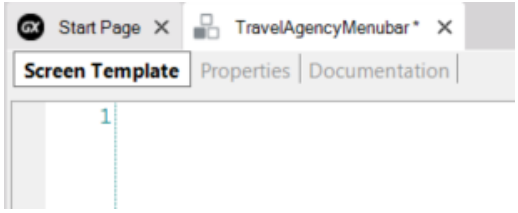

In the provider page you will find several menu examples, where the most suitable in your case is the Menubar. Here you can see the page with the data relative to that control.

On top you can see what the menu would look like, and further below you will see 3 tabs: Documentation, Source, and StackBlitz. Documentation includes general information about the control, Source has the HTML and examples, and StackBlitz opens up a screen where you may execute the code on a testing window.

The procedure for creating the user control in GeneXus that will be included in panel objects is similar to that of a web app built with web panels, as it was shown in other videos. You will obtain the control's HTML and add it in the user control to be created. Then you import other resources such as CSS libraries and so on.

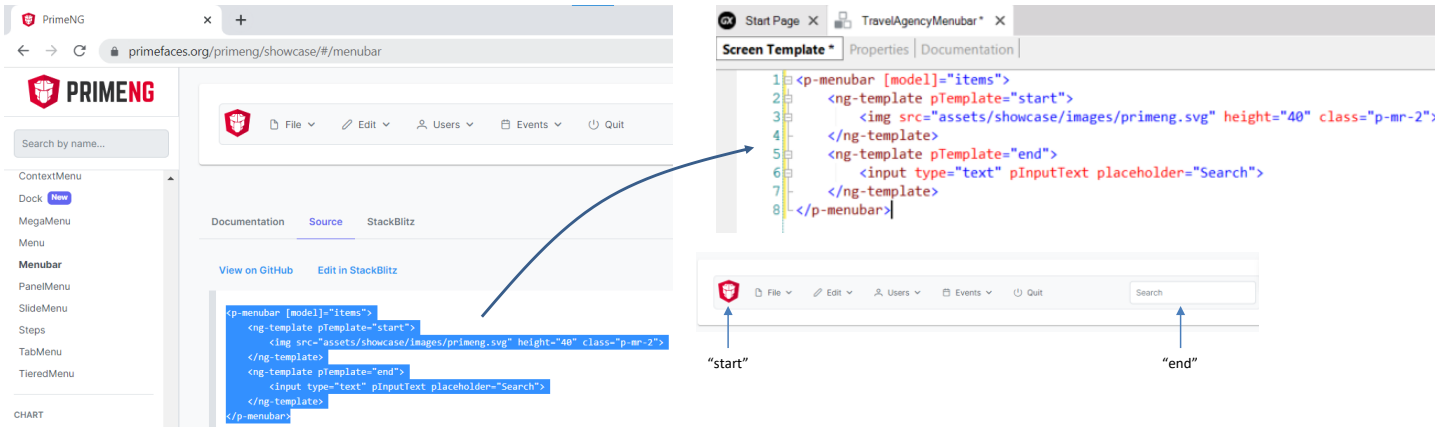
# Creating the User Control object




Create an object of the User Control type and call it TravelAgencyMenubar. The object has 2 sections with which you will be working, namely: Screen Template, where you will define the control's html code, and Properties, where you will assign values to some of the elements in the HTML.

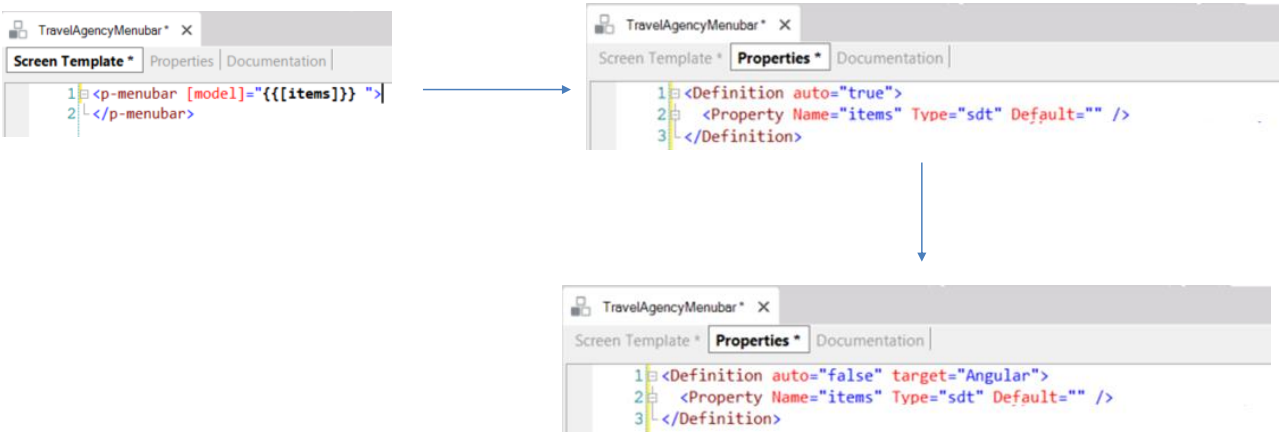
Copy and paste HTML code  on the control




Going back to the PrimeNG page, in Source you will see the HTML of the Menubar control. Copy it and paste it in the Screen Template section of your control.

Note that there are two ng-template blocks, one called "start", which defines the area of the icon and the other called "end", which defines the text to enter a search. Neither of these are of interest for your menu, so just delete those blocks and continue.

Substitute fixed data with variable elements




Now, substitute the fixed data included in the HTML with elements that will allow you to load them dynamically with your own data, either fixed or from the database.

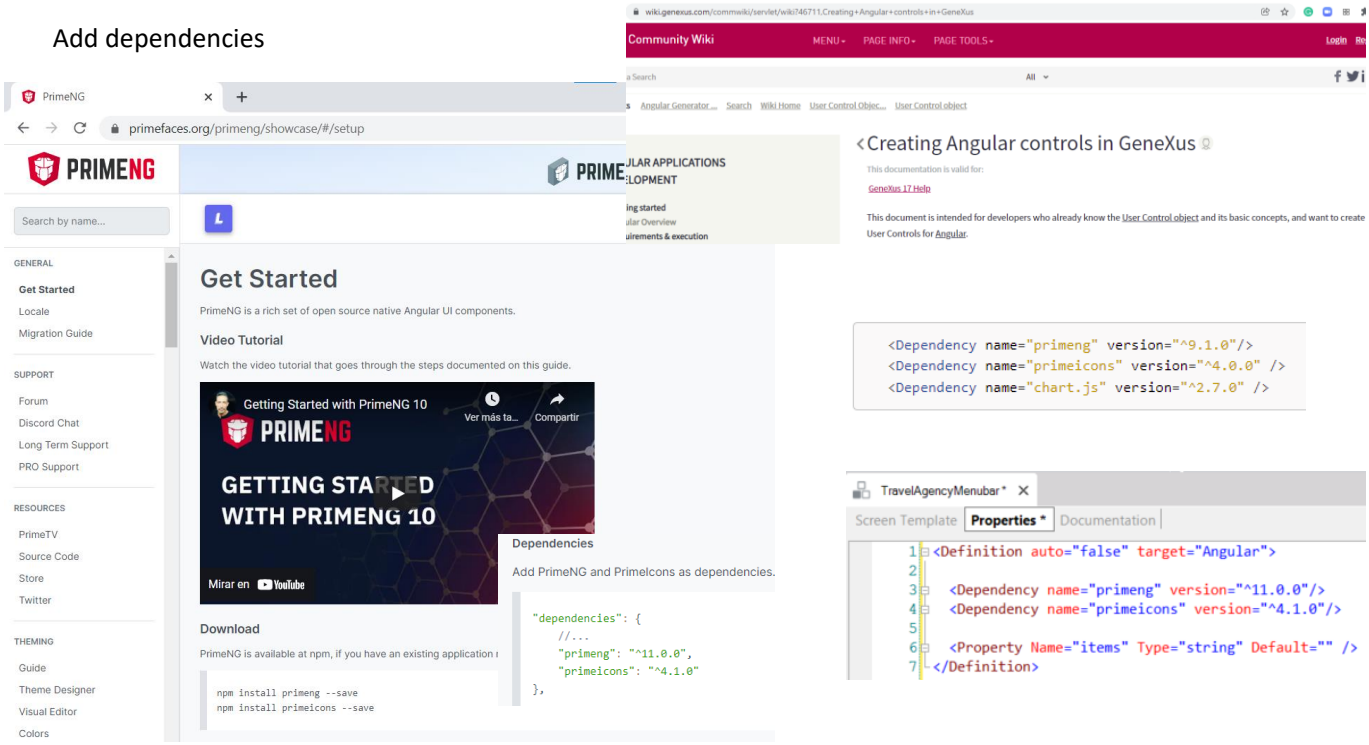
Substitute the text “items” in the first line with the element {{[items]}} between double sets of braces. The text “items” is between brackets, indicating that what will be there is a collection with the menu items. This should be loaded later, with the data from your menu.

This double braces syntax, known as “moustache”, enables you to replace fixed data with variable data.

Now you do Save and go to the Properties tab, where you will see that now a property has appeared as a consequence of the variable added. Note that the “items” property is of the “string” data type. Since it will be loaded with an SDT, then you will assign to it the “sdt” data type. The data types that you may use are shown in the general documentation of the Wiki User Controls.

Now change the Definition clause with auto=“false” and add target = “Angular”. This is required for the UC to remain visible in order to be added in a panel object.

## Add dependencies



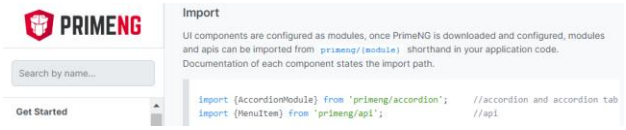

Now you must add several lines to import components from the provider that will be needed to properly interpret the control.

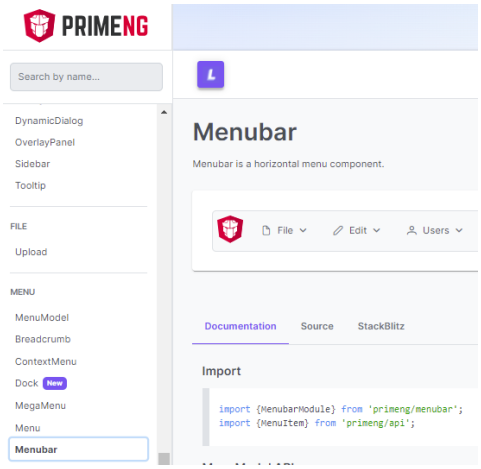
The first thing to add are the dependencies of the packages that will be imported. If you go to the PrimeNG site: **primefaces.org/primeng** and click on Get Started, you will be opening a page with a user guide on the library's controls. This guide depends on each provider, but they all have documentation showing how to obtain their components.

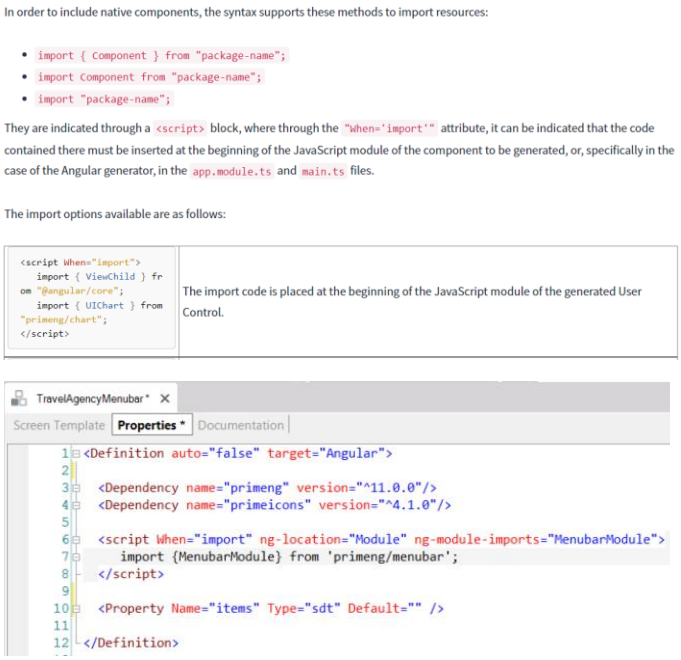
Section Download indicates the packages that should be installed with npm. These are the packages you must declare in your user control as dependencies. If you go to "Angular CLI Integration" you will see an example where the packages referred before are acknowledged to obtain information on the version of the packages.

If you go to the wiki page that guides you on how to use a user control in Angular and then go to the Dependencies section, you will see the syntax that should be used. Add the dependencies to the Properties section in your User Control with the data from the provider page.

## Add resource imports




Going back to the Get Started page of PrimeNG, you will see that the Import section remains because it will be necessary to import modules. Here you have a generic example, but to know which modules you will need for the Menubar control you must go to the information page of the Menubar control that is located further down in the menu, to the left, under the Menu section.

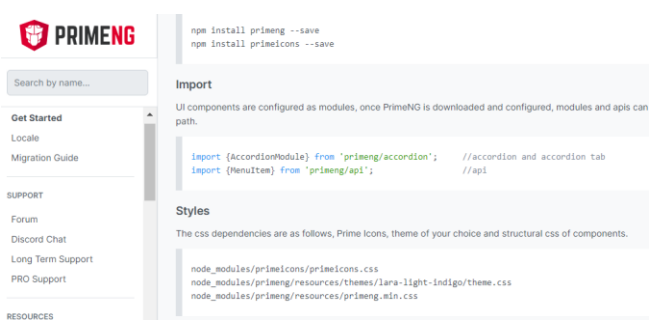
You will see that the modules to be imported are MenubarModule and MenuItem, but considering that your menu will not have subitems, then you will only need the Menubar module.

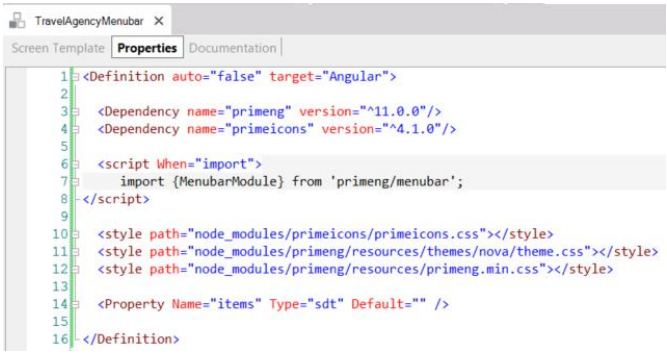
Front-end generators, such as Angular, contribute with advanced ways of including external resources using the “import” sentence. This declaration may be used both for including JavaScript modules and for including other types of resources, like images, SVG files or stylesheets, through a package as webpack.

In the wiki you will find the syntax to use in GeneXus for the import. First there is a description of the various syntaxes of the import command and below are some examples of their use. You may choose any because in this case there are no special requirements as to where the code generated must be.

Adapt the wiki example to import the MenubarModule module from primeng/menubar and add it to the Properties window.

## Add styles



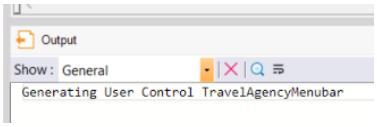




### Style sheets

The possibility to declare style sheets to be included is added, using the <style> tag and the path attribute:

```
<style path="node_modules/primeicons/primeicons.css" />
<style path="node_modules/primeng/resources/themes/nova-light/theme.css" />
<style path="node_modules/primeng/resources/primeng.min.css" />
```

The Angular generator incorporates these styles in the style property of the angular.json file.

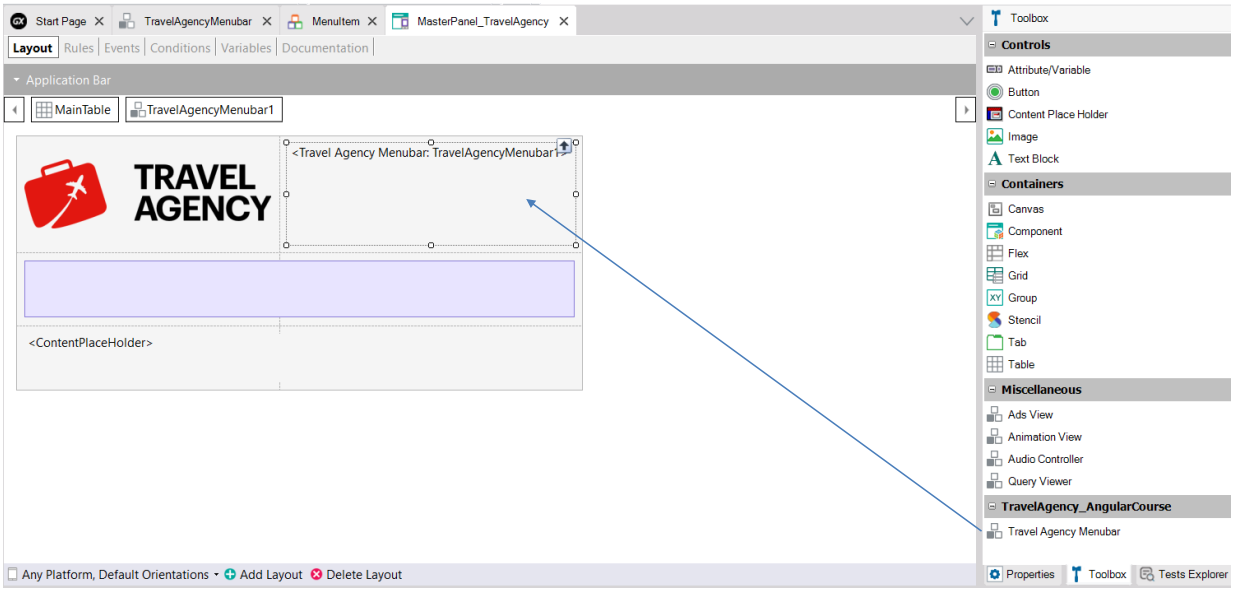



Back to the Get Started page of PrimeNG you will see an indication that you must load icons and CSS code.
Go to the wiki again to see how to write that in GeneXus using the style clause and detailing the path where the CSS necessary will be located.

Now that the definition of the User Control is complete, do Save and you will see that the TravelAgencyMenubar user control has been generated.

Now proceed to include it in a panel object.

Insert the User Control created in the MasterPanel

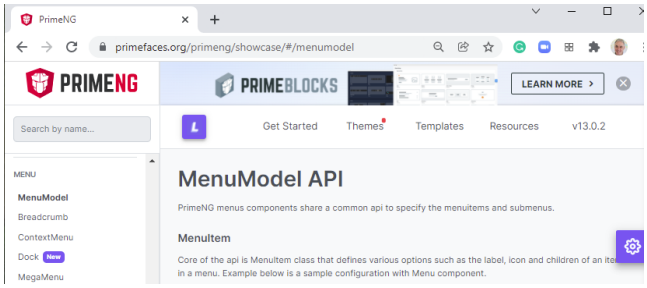



The best place to include a menu is in a Master Panel object so that the menu is present in all of the app's screens, ensuring quick access to the various parts. So, open the MasterPanel_TravelAgency object you had built.

You will see that the TravelAgencyMenubar user control you created appears on the toolbar, so now you drag it to the form.

A user control called TravelAgencyMenubar1 is created, and since you don't need the table to the right of the logo, you just simplify the design a bit.

Create the SDT with the structure required

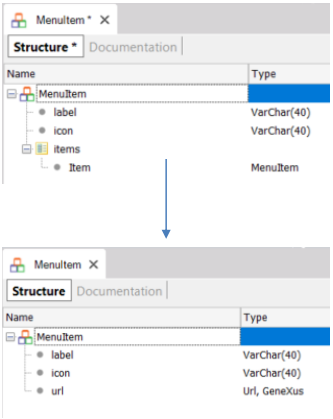




```
export class MenuDemo {

    private items: MenuItem[];

    ngOnInit() {
        this.items = [{
            label: 'File',
            items: [
                {label: 'New', icon: 'pi pi-plus'},
                {label: 'Open', icon: 'pi pi-download'}
            ]
        },
        {
            label: 'Edit',
            items: [
                {label: 'Undo', icon: 'pi pi-refresh'},
                {label: 'Redo', icon: 'pi pi-repeat'}
            ]
        }];
    }
}
```




As mentioned, the menu items are loaded using an SDT. To know which structure you must have for the SDT return to the documentation of the Menubar control, and in the MenuModel API section see the documentation of MenuItem with its properties.

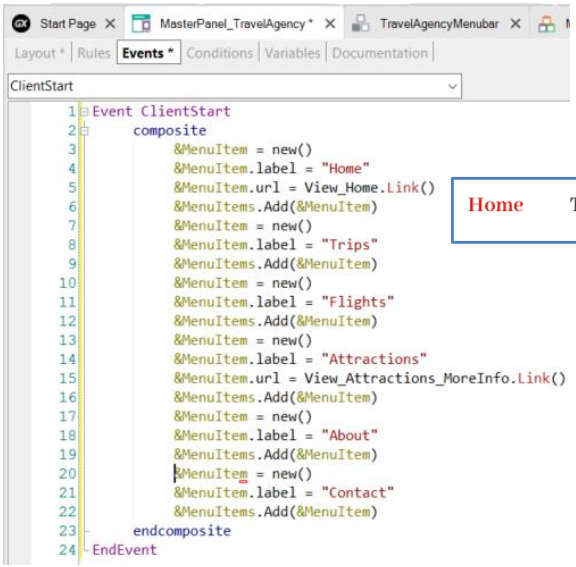
There you will find that an item in the menu has an id, a label, an icon, a command, a url, and so on.
Further up you will see a very simple example that only uses the label property, showing that an item may have a collection of subitems with the same structure of items.

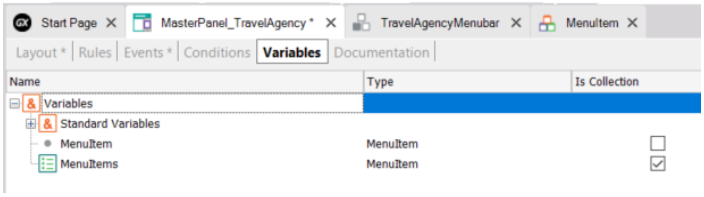
So, you can create an SDT called MenuItem with a similar structure and with a label of the character type, in addition to an icon of the character type as well, and a collection of items of the same type you are creating: MenuItem.

Since you will not use subitems in the menu for this case, you will not add the collection. But do add a url element that will contain the url of the object that you will invoke when you press the menu.

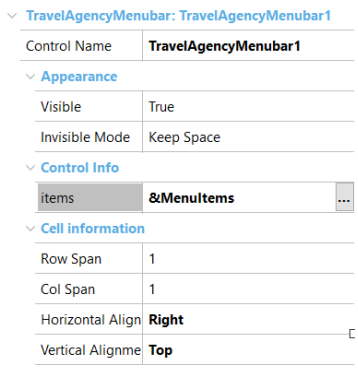
Every API of each provider is different, so you must refer to the provider documentation to see how you set up your User Control and which structures you must use to load the data needed.

## Load the SDT in the MasterPanel events








Now go to the variables section of MasterPanel_TravelAgency. Create the MenuItem variable that will automatically be of the MenuItem type. Then re-name it MenuItems and mark it as collection.
Then create another MenuItem variable to be used in loading each item in the menu to later add it to the collection of items.

Open the panel's event section and add a ClientStart event. Write composite. And insert the &MenuItem variable assigning to it a new() command.

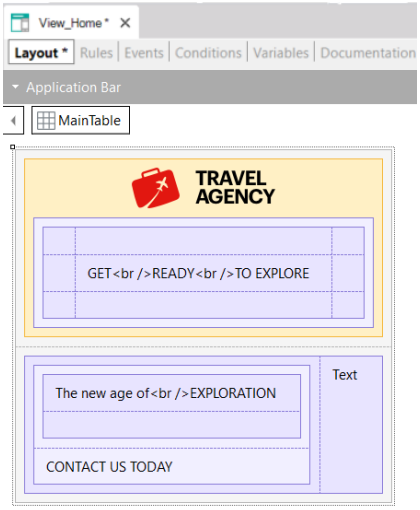
Then assign value "Home" to the label member and add the menu item to the collection.

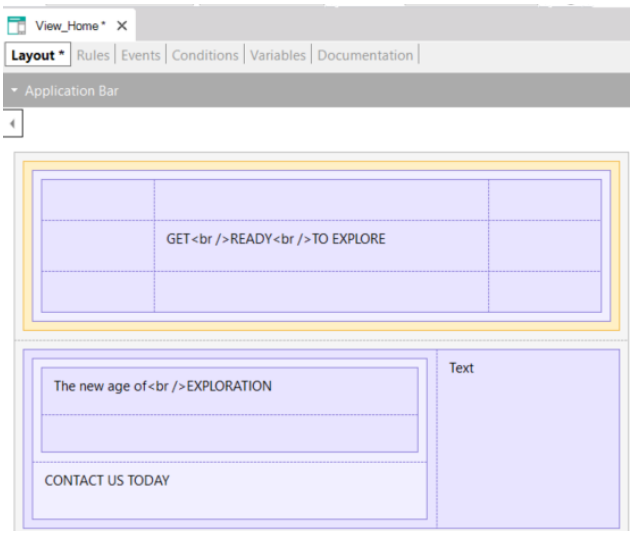
Then do the same for the other items in the menu.

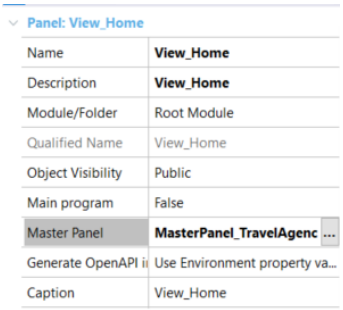
Close the composite command and the event and you will have the items collection variable loaded already.

Now go back to the panel's layout and select the TravelAgencyMenubar1 user control. In its "items" property select the &MenuItems variable. This is informing the user control where it must search for the data to show the items.

Modify the startup panel object and assign the Master Panel



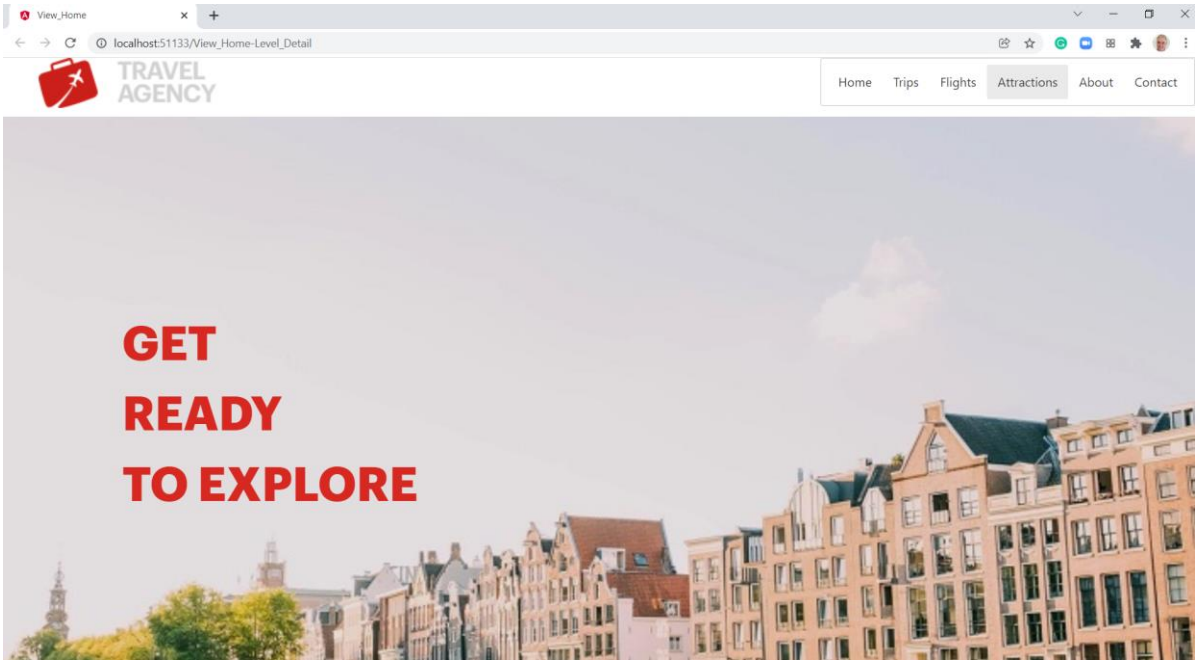




To try this, open the View_Home panel that was the main object you created as startup object prior to using the design sent by the designer from Sketch, remove the icon and assign the Master Panel property with Master_Panel_TravelAgency, which already includes the icon and the menu.

Right click on the View_Home panel and select Run.

## Executing the User Control




You will see the View_Home object open up, showing the menu on the screen's upper right.

When you drag the mouse over the buttons you will see them change in color as you go over them; this is to indicate the one that will be selected if you click.

You will not assign objects to the menu here because you already have the menu that you will actually use, that is, the one created from Sketch.

The use of user controls in an app generated in Angular implies certain considerations inherent to the framework's architecture, so you must refer to the documentation of each user control provider, or if you set up your own, then you should consider a way of including the modules, styles and other components necessary.

The possibility of defining your own user controls in GeneXus instead of having only the standard controls allows you to significantly increase the user experience of your apps, and makes it possible to use creations from a variety of sources and include them in your development.

In further videos you will also see how to include other external functionalities accessing APIs.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications