Controles de Usuario en Angular

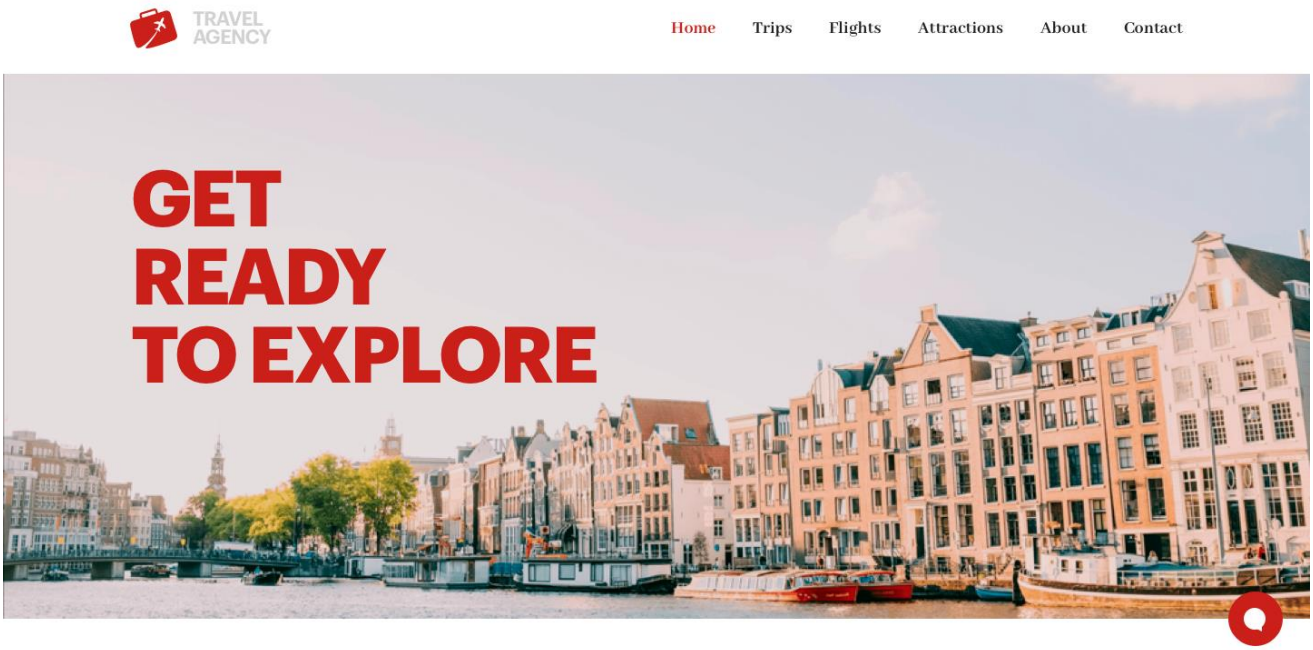
GeneXus™

En otros videos vimos varios controles de pantalla que nos ayudan a construir la interfaz de usuario y también vimos cómo mejorar el diseño de la aplicación realizando definiciones en un objeto Design System y cómo importar un diseño pronto hecho por un diseñador en Sketch.

En este video veremos cómo además de los controles de pantalla predefinidos que tenemos disponibles en la barra de herramientas, podemos crear nuestros propios controles para enriquecer aún más la experiencia de usuario.

Importación de recursos a partir de proveedores de UI

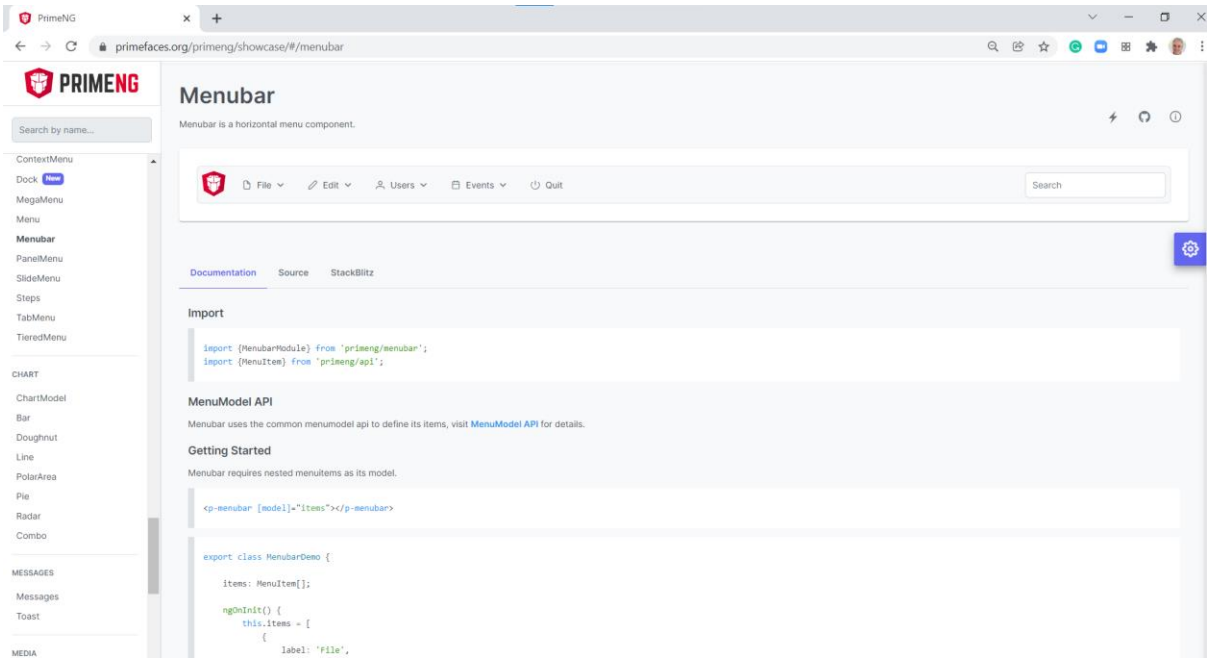
GeneXus nos permite crear controles de usuario a partir de controles construidos por diseñadores o disponibles en plataformas de proveedores de recursos de User Interface, tanto sean componentes nativos del framework Angular como por ejemplo PrimeNG, AngularMaterial o Material-UI, como recursos HTML, CSS y JavaScript de proveedores genéricos como SemanticUI, VanillaFramework, o bibliotecas de componentes Bootstrap.



Si vemos el diseño que nos dio el diseñador, vemos que arriba a la derecha hay un menú de opciones.

Vamos a construir el menú utilizando un user control provisto por PrimeNG.

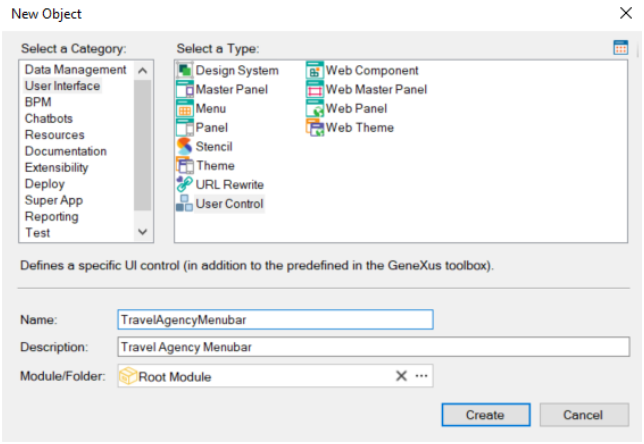
Definiendo el User Control a crear

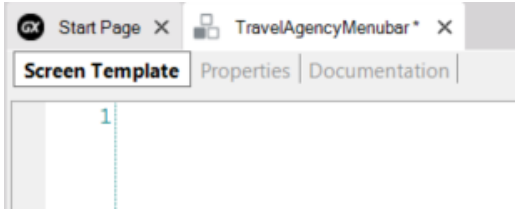


En la página del proveedor, encontramos varios ejemplos de menúes, y el que más nos sirve es el Menubar. Aquí vemos la página con los datos de ese control.
Arriba vemos cómo se vería el menú y más abajo vemos 3 solapas: Documentation, Source y StackBlitz. En Documentation encontramos información general del control, en Source está el HTML y ejemplos y el StackBlitz se abre una pantalla donde podemos ejecutar el código en una ventana de prueba.

El procedimiento de creación del user control en GeneXus para ser incluido en objetos panel, es similar al de una aplicación web construida con web panels que vimos en otros videos, vamos a obtener el HTML del control y agregarlo en el user control que vamos a crear. Luego imporaremos otros recursos, como bibliotecas CSS, etc.

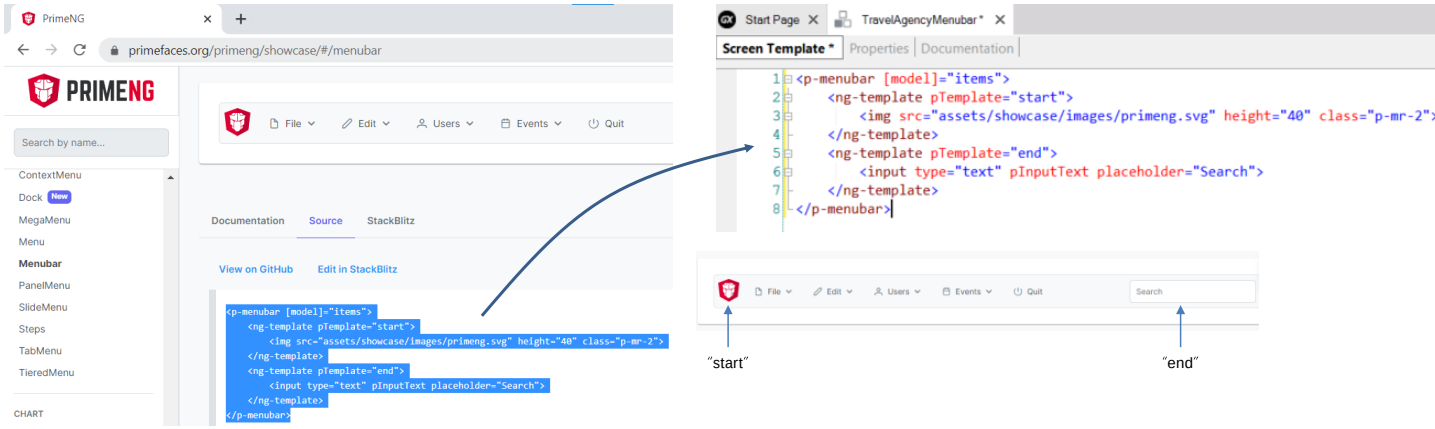
Creación del objeto User Control





Creamos un objeto del tipo User Control y le ponemos de nombre TravelAgencyMenubar. El objeto tiene 2 secciones con la que vamos a trabajar : Screen Template, donde definimos el código html del control y Properties, donde asignaremos valores a algunos de los elementos del HTML.

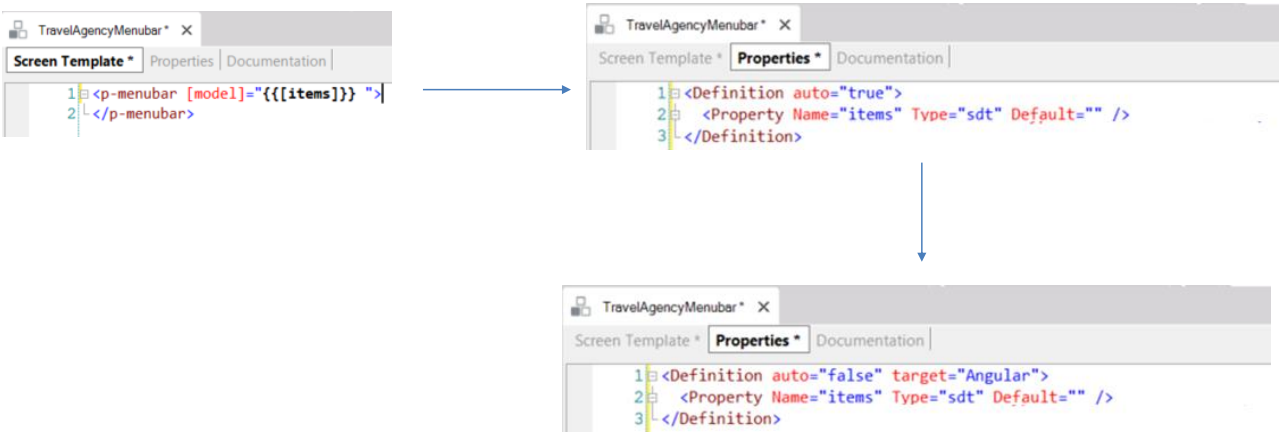
Copiamos y pegamos código HTML en nuestro control



Si volvemos a la página PrimeNG y vamos a Source, podemos ver el HTML del control Menubar, así que lo copiamos y lo pegamos en la sección Screen Template de nuestro control.

Observamos que tenemos dos bloques ng-template, uno con el nombre "start" que define a la zona del icono y otro con el nombre "end" que define al texto para entrar una búsqueda. Ninguna de las dos cosas nos interesa en nuestro menú, así que eliminamos estos bloques.

Sustituimos datos fijos por elementos variables



Ahora vamos a cambiar los datos fijos que vinieron en el HTML, por elementos que nos permitirán poder cargarlos dinámicamente con datos nuestros, ya sean fijos o de la base de datos.

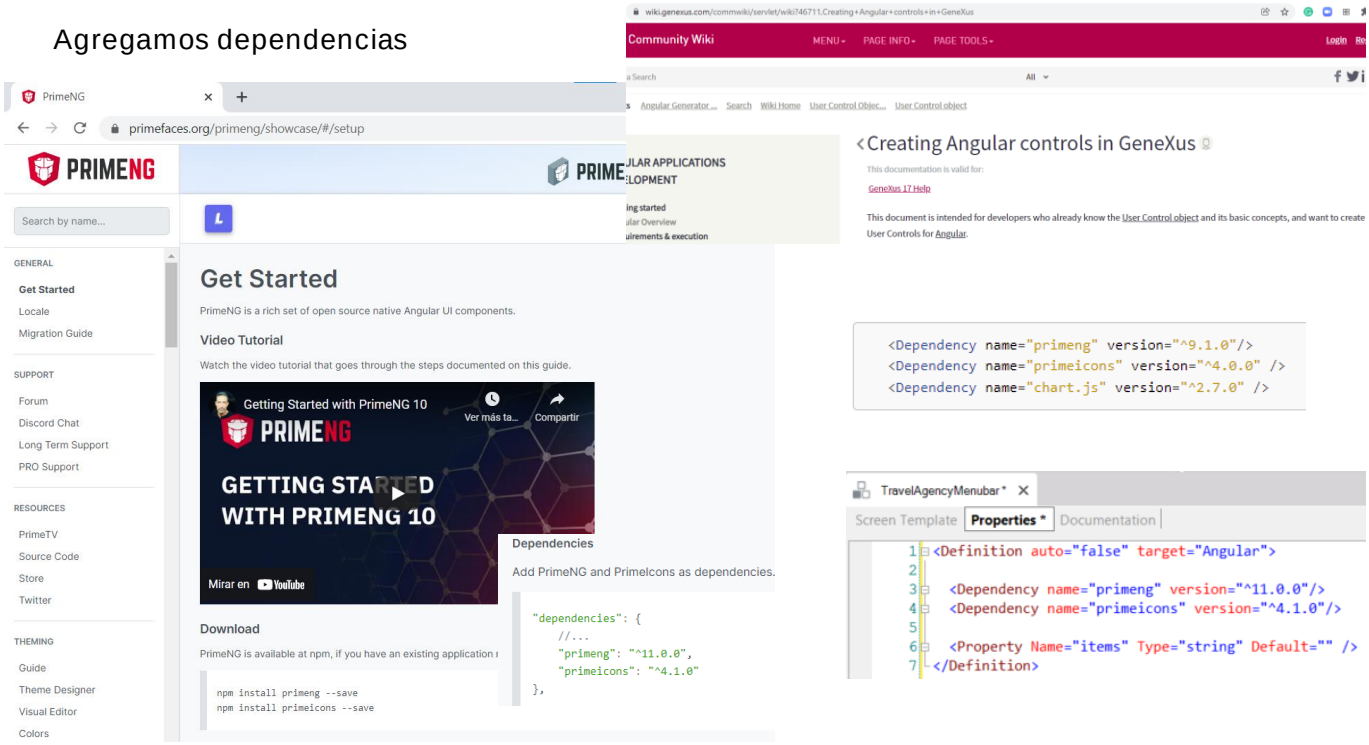
Sustituimos el texto "ítems" en la primera línea por el elemento {{[items]}} encerrado entre doble llaves. Observemos que además de las dobles llaves, el texto ítems está entre paréntesis rectos, que indica que lo que irá allí es una colección con los ítems del menú. Esto lo cargaremos luego con los datos de nuestro menú.

Esta sintaxis con doble llave se conoce como "mostacho" y nos permite sustituir datos fijos por datos variables.

Salvamos y vamos a la pestaña Properties. Vemos que allí aparece una property por la variable que agregamos. Notamos que la property "ítems" tiene tipo de datos "string". Como vamos a cargarla con un SDT, le asignamos el tipo de datos "sdt".
Los tipos de datos que podemos usar los encontramos en la documentación general de los User Controls del Wiki.

Vamos a cambiar la cláusula Definition, poniendole auto="false" y agregamos target = "Angular". Esto es fundamental para que el UC quede visible para ser agregado en un objeto panel.

Agregamos dependencias



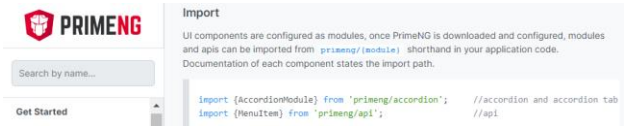
Ahora debemos agregar varias líneas para importar componentes del proveedor, necesarios para que pueda interpretarse adecuadamente el control.

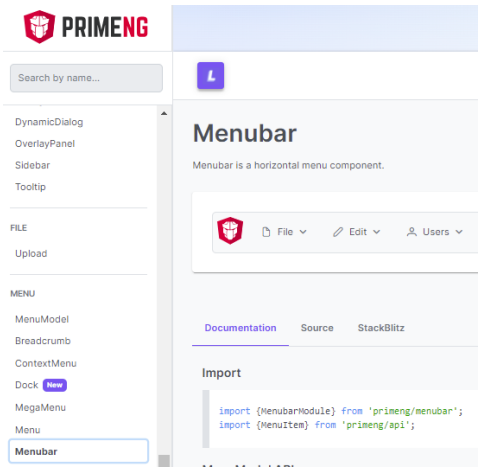
Lo primero que debemos agregar son las dependencias de los paquetes a importar. Si vamos al sitio de PrimeNG: **primefaces.org/primeng** y hacemos clic en Get Started se abre una página de guía del uso de los controles de la biblioteca. Esta guía depende de cada proveedor, pero todos tienen documentación que nos muestra cómo obtener sus componentes.

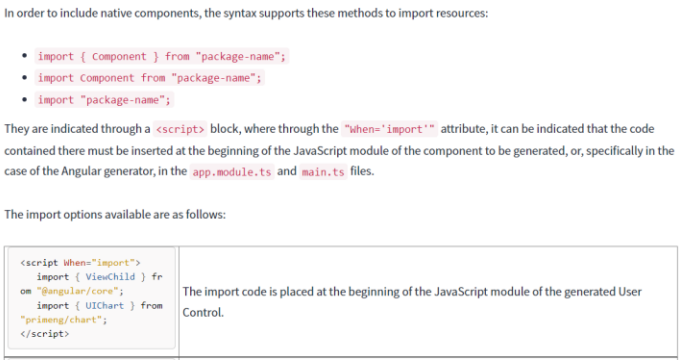
En la sección Download nos indica los paquetes que se deberían instalar con npm. Estos paquetes son los que necesitamos declarar en nuestro user control como dependencias. Si vamos a "Angular CLI Integration", vemos un ejemplo donde reconocemos los paquetes mencionados antes y nos da información de la versión de los paquetes.

Si vamos a la página del wiki que nos guía como utilizar un user control en Angular y vamos a la sección de Dependencias, vemos la sintaxis que debemos usar. Agregamos las dependencias a la sección de Properties de nuestro User Control con los datos de la página del proveedor.

Agregamos importación de recursos







Si volvemos a la página de Get Started de PrimeNG, vemos que sigue con la sección de Import, ya que es necesario importar módulos. Aquí vemos un ejemplo genérico, pero para saber cuáles son los módulos que precisamos para el control Menubar, vamos a la página de información del control Menubar, que la encontramos más abajo en el menú de la izquierda, bajo la sección Menu.

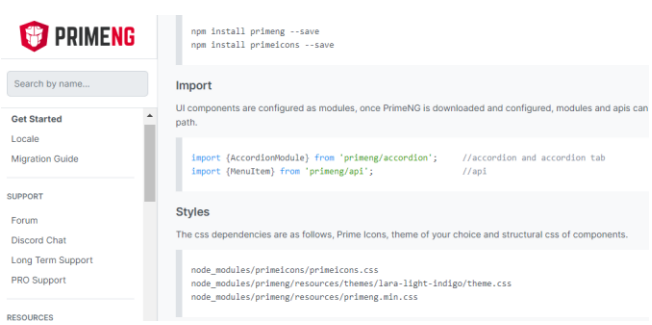
Vemos que los módulos que deberíamos importar son el MenubarModule y el MenuItem. Como no vamos a tener subítems en nuestro menú, solamente precisamos el módulo Menubar.

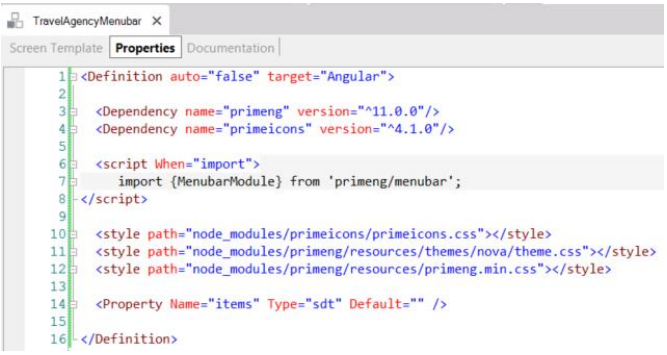
Los generadores de front-end como Angular agregan formas avanzadas de incluir recursos externos, utilizando la sentencia "**import**". Esta declaración se puede utilizar tanto para incluir módulos JavaScript como para incluir, a través de un paquete como webpack, otros tipos de recursos como imágenes, archivos SVG u hojas de estilo.

Si vamos al wiki, encontramos la sintaxis que debemos usar en GeneXus para el import. Primero tenemos una descripción de las distintas sintaxis del comando import y más abajo tenemos algunos ejemplos de uso. Podemos elegir cualquiera, ya que en nuestro caso no tenemos requerimientos especiales de donde debe quedar el código generado.

Adaptamos el ejemplo del wiki para importar el módulo MenubarModule desde primeng/menubar y lo agregamos en la ventana de Properties.

Agregamos estilos



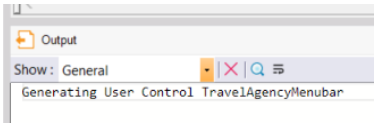


Style sheets

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.

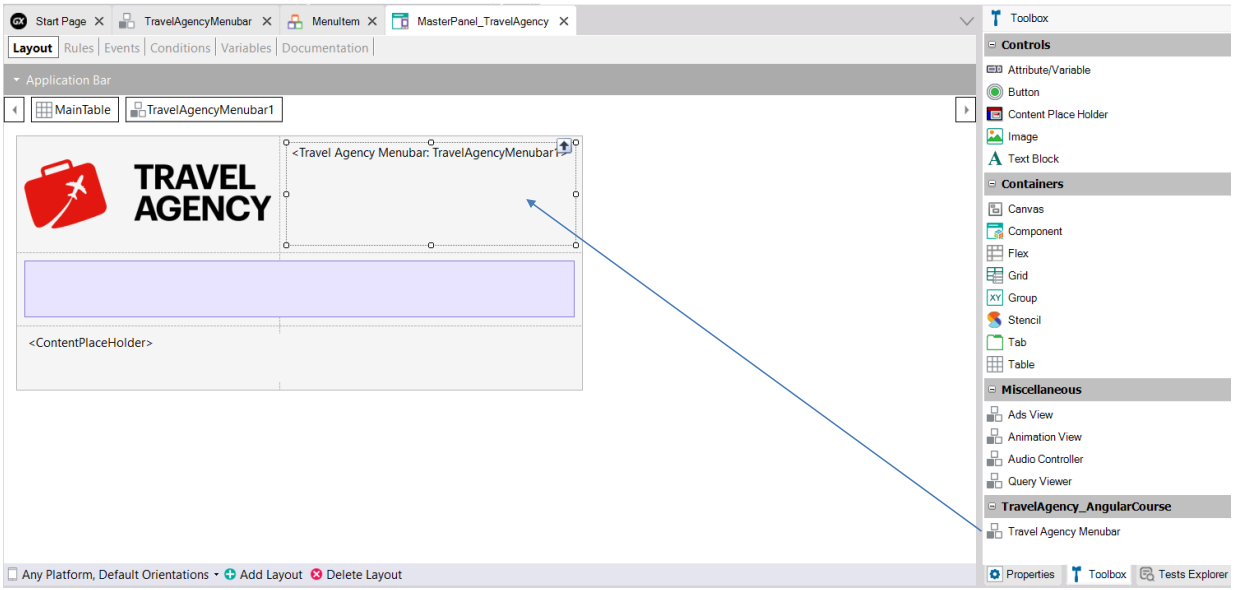


Volvemos una vez más a la página Get Started de PrimeNG y vemos que nos indica que debemos cargar iconos y código CSS.
Nuevamente vamos al wiki para ver cómo escribimos eso en GeneXus, usando la cláusula style y detallando el path donde estarán ubicados los CSS necesarios,.

Ahora sí nos quedó completa la definición del User Control, salvamos y vemos que se generó el user control TravelAgencyMenubar.

Ahora vamos a incorporarlo a un objeto panel.

Insertamos el User Control que creamos en el MasterPanel

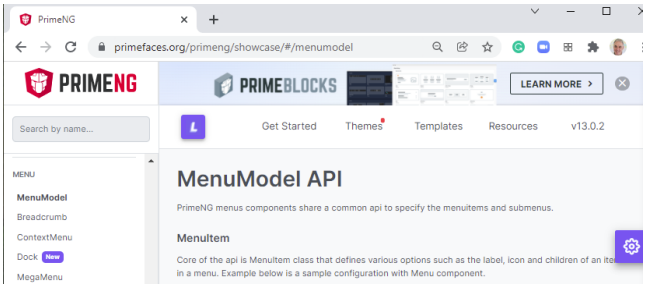


El lugar más lógico de incluir un menú es en un objeto Master Panel, así el menú está presente en todas las pantallas de la aplicación y nos asegura un acceso rápido a las distintas partes. Así que abrimos al objeto MasterPanel_TravelAgency que habíamos construido.

Vemos que en la barra de herramientas aparece el user control TravelAgencyMenubar que creamos, así que lo arrastramos al form.

Se crea un user control de nombre TravelAgencyMenubar1 y como no necesitamos la tabla a la derecha del logo, simplificamos un poco el diseño.

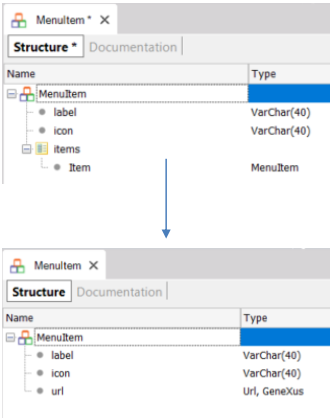
Creamos el SDT con la estructura necesaria




```
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'}
            ]
        }];
    }
}
```

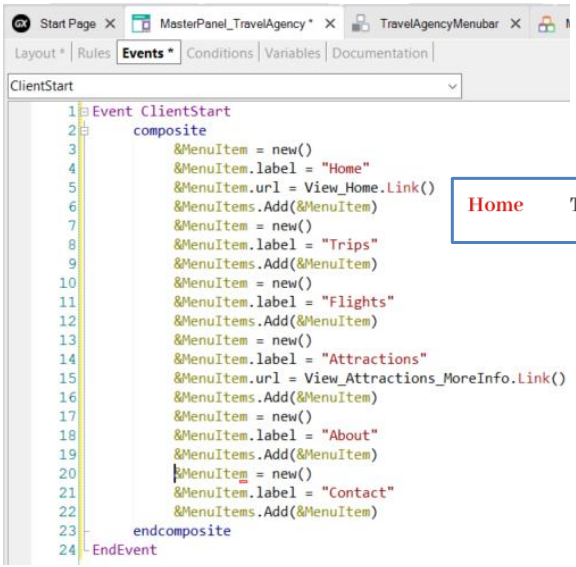


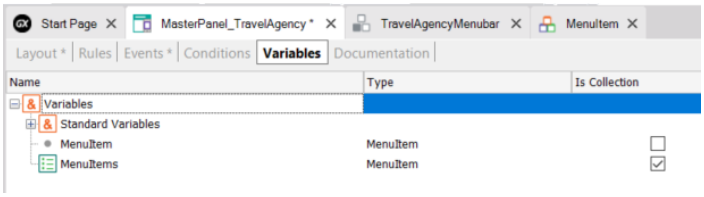
Como dijimos antes, los ítems del menú los vamos a cargar usando un SDT. Para saber la estructura que tenemos que darle al SDT volvemos a la documentación del control Menubar y la sección MenuModel API vemos la documentación del MenuItem con sus propiedades.
Allí identificamos que un item del menú tiene un id, un label, un icono, un comando, una url, etc. Más arriba vemos un ejemplo muy simple que usa únicamente la propiedad label y nos muestra que un item puede tener una colección de subitems y que los subitems tienen la misma estructura de los ítems.

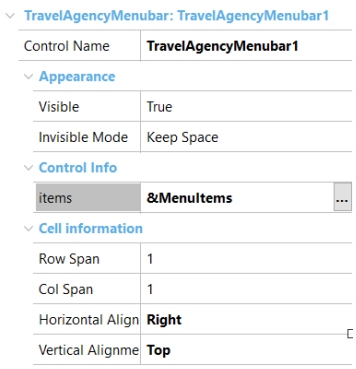
Así que podemos crear un SDT de nombre MenuItem con una estructura similar que tenga un label del tipo character, un icon también del tipo carácter y una colección de ítems del mismo tipo que estamos creando: MenuItem.
Como en nuestro caso no vamos a usar subitems en el menú, no le agregamos la colección. Y sí agregamos un elemento url que contendrá la url del objeto al que invocaremos cuando presionemos el menú.

Cada API de cada proveedor es diferente, así que debemos remitirnos a la documentación del proveedor para ver cómo armamos nuestro User Control y qué estructuras debemos usar para cargar los datos que necesita.

Cargamos el SDT en los eventos del MasterPanel







Ahora vamos a la sección de variables del MasterPanel_TravelAgency. Creamos una variable MenuItem que automáticamente nos queda del tipo MenuItem. La renombramos MenuItems y la marcamos como colección. Luego creamos otra variable MenuItem que usaremos para cargar cada item del menú para luego agregarlo a la colección de ítems.

Abrimos la sección de eventos del panel y agregamos un evento ClientStart. Escribimos composite. Ahora insertamos a la variable &MenuItem y le asignamos un comando new().

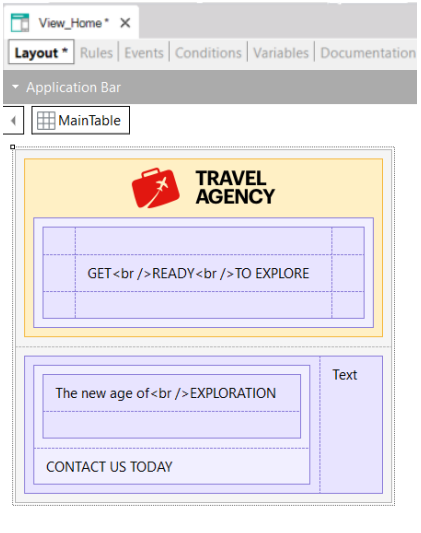
Luego asignamos al miembro label el valor "Home" y agregamos el item del menú a la colección.

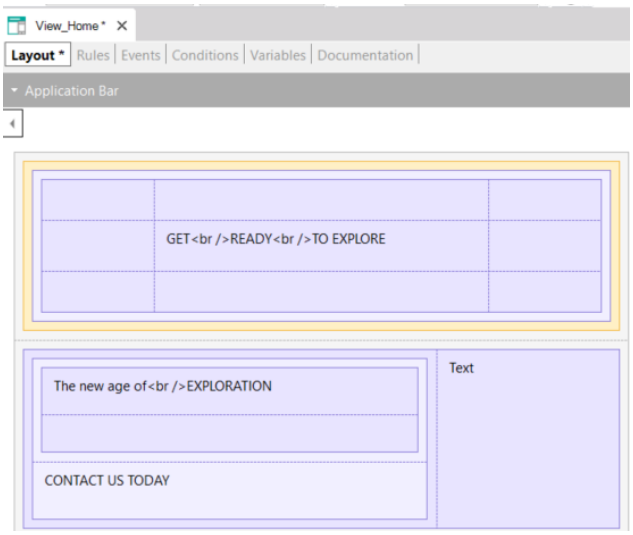
Hacemos lo mismo para los otros ítems del menú.

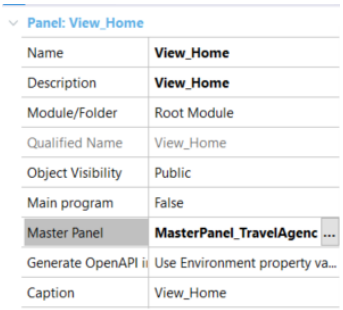
Cerramos el comando composite y el evento. Ya tenemos cargada la variable colección de ítems.

Ahora volvemos al layout del panel, seleccionamos al user control TravelAgencyMenubar1 y en su propiedad "ítems" seleccionamos a la variable &MenuItems. Con esto estamos diciéndole al user control adonde tiene que ir a buscar los datos para mostrar los ítems.

Modificamos el objeto panel de startup y le asignamos el Master Panel



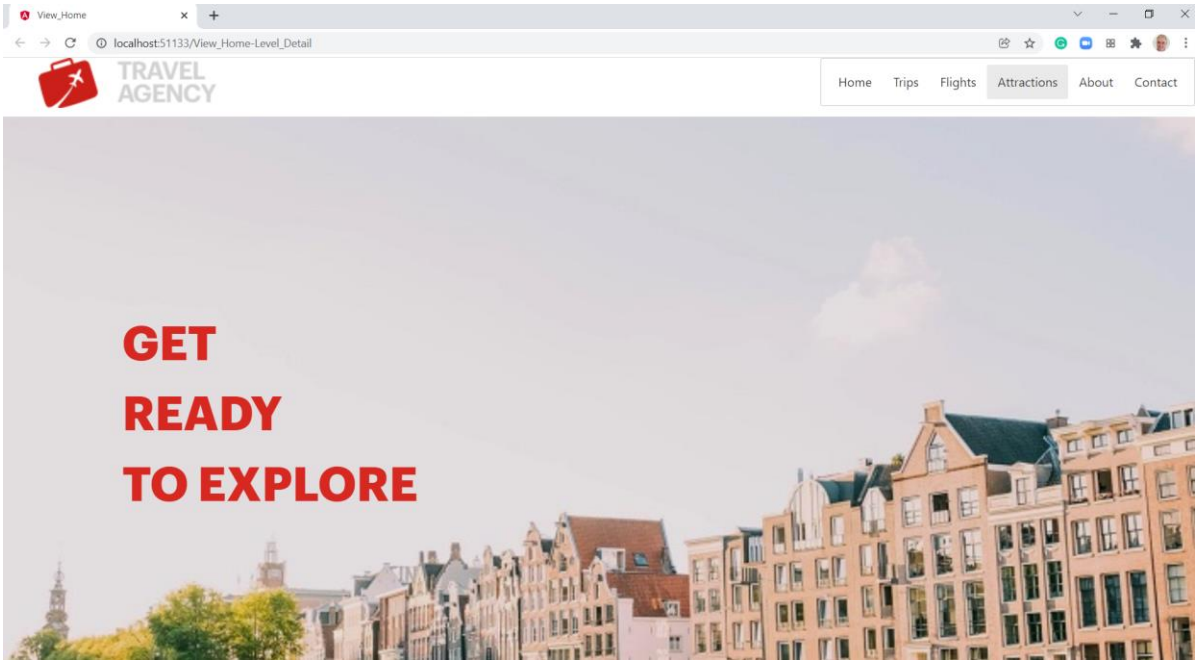




Para probar esto, abrimos el panel View_Home, que fue el objeto main que habíamos creado nosotros como objeto de startup antes de usar el diseño que nos enviara el diseñador desde Sketch, le sacamos el icono y le asignamos la propiedad Master Panel con el Master_Panel_TravelAgency, que ya incluye el ícono y el menú.

Damos botón derecho sobre el panel View_Home y seleccionamos Run.

Ejecución del User Control



Vemos que se abre el objeto View_Home, mostrándose el menú en la parte superior derecha de la pantalla.

Si desplazamos el mouse sobre los botones, vemos que cambian de color a medida que pasamos sobre ellos para indicarnos el que quedará seleccionado si damos clic.

Aquí no asignamos objetos al menú, ya que ya tenemos el menú que realmente vamos a usar, que fue el creado a partir de Sketch.

El uso de user controls en una aplicación generada en Angular, tiene algunas consideraciones que son propias de la arquitectura del framework, y por eso es que debemos remitirnos a la documentación de cada proveedor de user controls o si armamos uno nosotros, debemos considerar la forma de incluir los módulos, estilos y otros componentes necesarios.

Que podamos definir nuestros propios user controls en GeneXus y no quedarnos únicamente con los controles estándares , nos permite aumentar significativamente la experiencia de usuario de nuestras aplicaciones, ya que podemos utilizar creaciones de muchas fuentes e incorporarlas a nuestros desarrollos.

En otros videos veremos como podemos también incorporar otras funcionalidades externas, accediendo a API´s.

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