Controles de Usuário em Angular

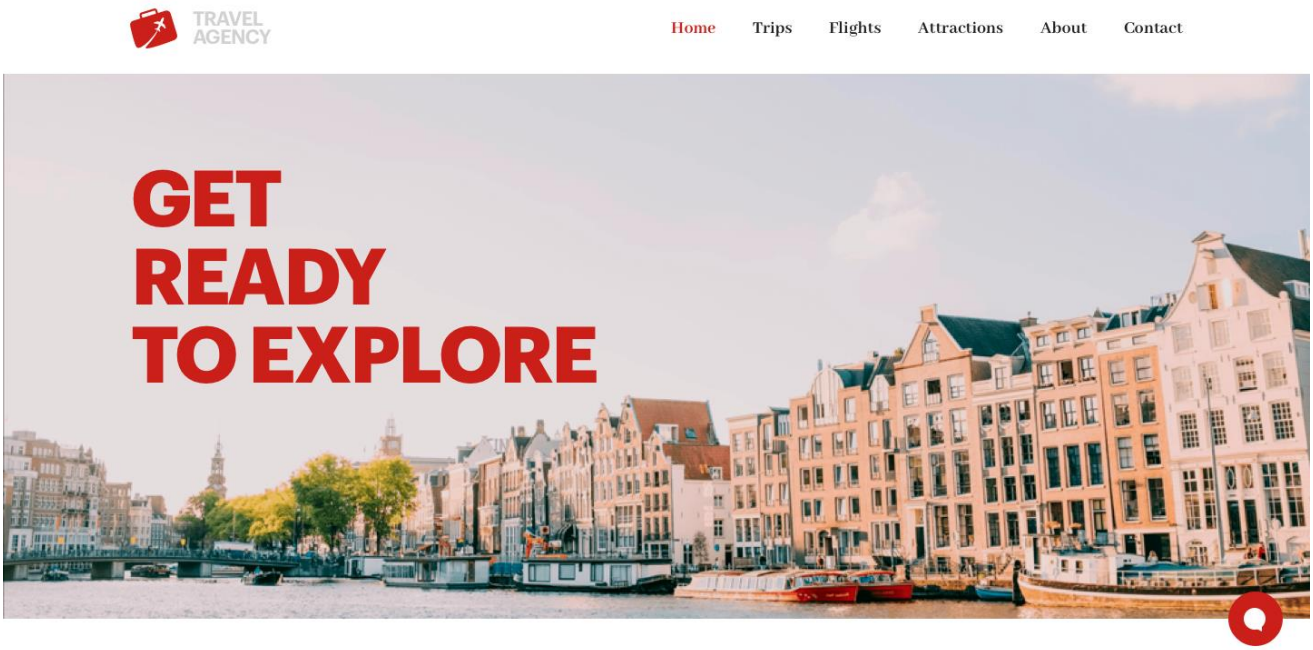
GeneXus™

Em outros vídeos vimos vários controles de tela que nos ajudam a construir a interface de usuário e também vimos como melhorar o design da aplicação realizando definições em um objeto Design System e como importar um design pronto feito por um designer em Sketch.

Neste vídeo veremos como, além dos controles de tela predefinidos que temos disponíveis na barra de ferramentas, podemos criar nossos próprios controles para enriquecer ainda mais a experiência de usuário.

Importação de recursos a partir de provedores de UI

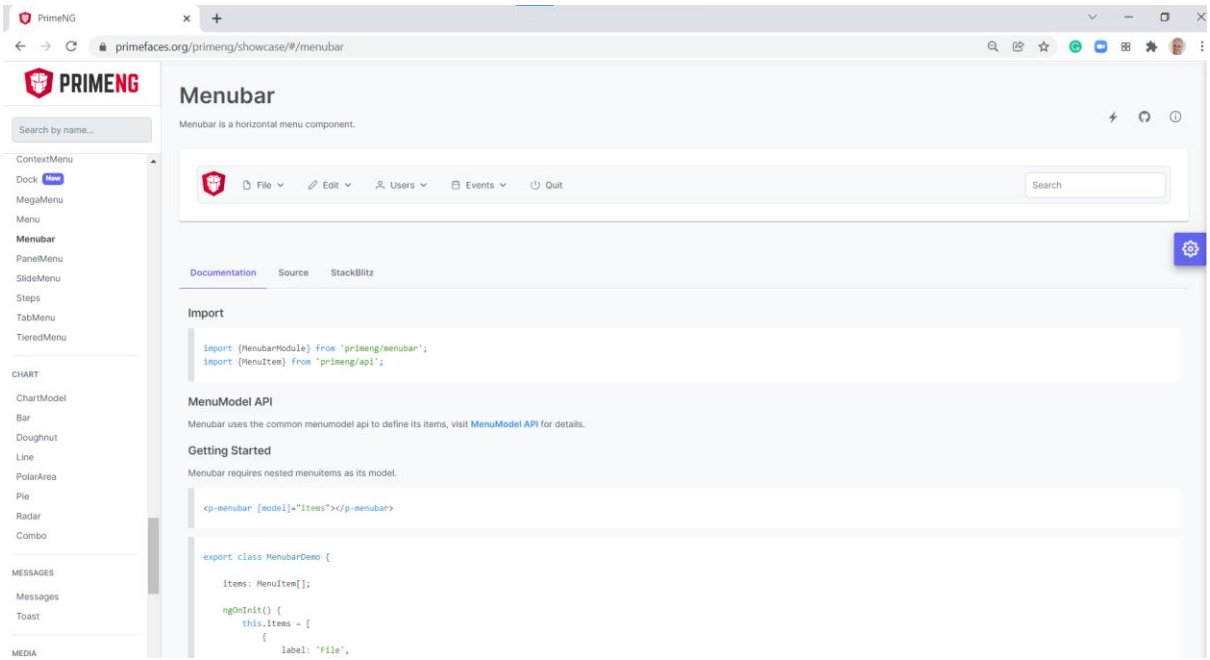
GeneXus nos permite criar controles de usuário a partir de controles construídos por designers ou disponíveis em plataformas de provedores de recursos de User Interface, sejam eles componentes nativos do framework Angular como por exemplo PrimeNG, AngularMaterial ou Material-UI, como recursos HTML, CSS e JavaScript de provedores genéricos como SemanticUI, VanillaFramework, ou bibliotecas de componentes Bootstrap.



Se observamos o design que o designer nos deu, vemos que acima à direita há um menu de opções.

Vamos construir o menu utilizando um user control fornecido por PrimeNG.

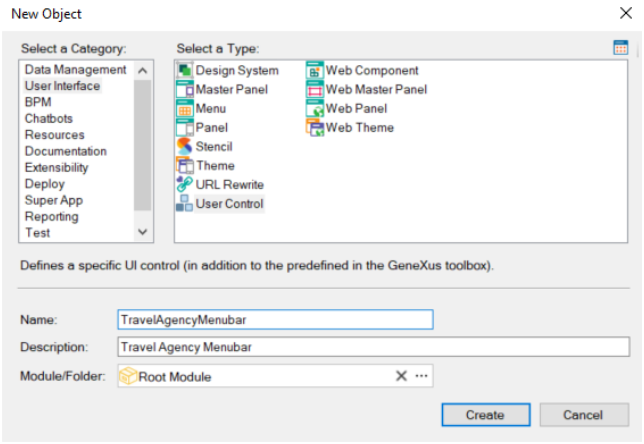
Definindo o User Control a criar

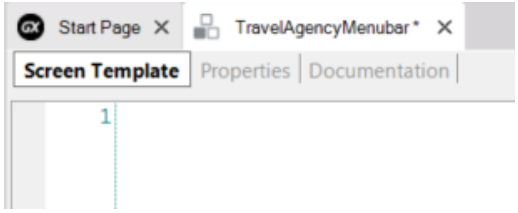


Na página do provedor, encontramos vários exemplos de menus, e o que mais nos atende é o Menubar. Aqui vemos a página com os dados desse controle.
Acima vemos como seria visto o menu e mais abaixo vemos 3 abas: Documentation, Source e StackBlitz. Em Documentation encontramos informações gerais do controle, em Source está o HTML e exemplos, e em StackBlitz é aberta uma tela onde podemos executar o código em uma janela de teste.

O procedimento de criação do user control em GeneXus para ser incluído em objetos panel é semelhante ao de uma aplicação web construída com web panels que vimos em outros vídeos, vamos obter o HTML do controle e adicioná-lo no user control que vamos criar. Em seguida, importaremos outros recursos, como bibliotecas CSS, etc.

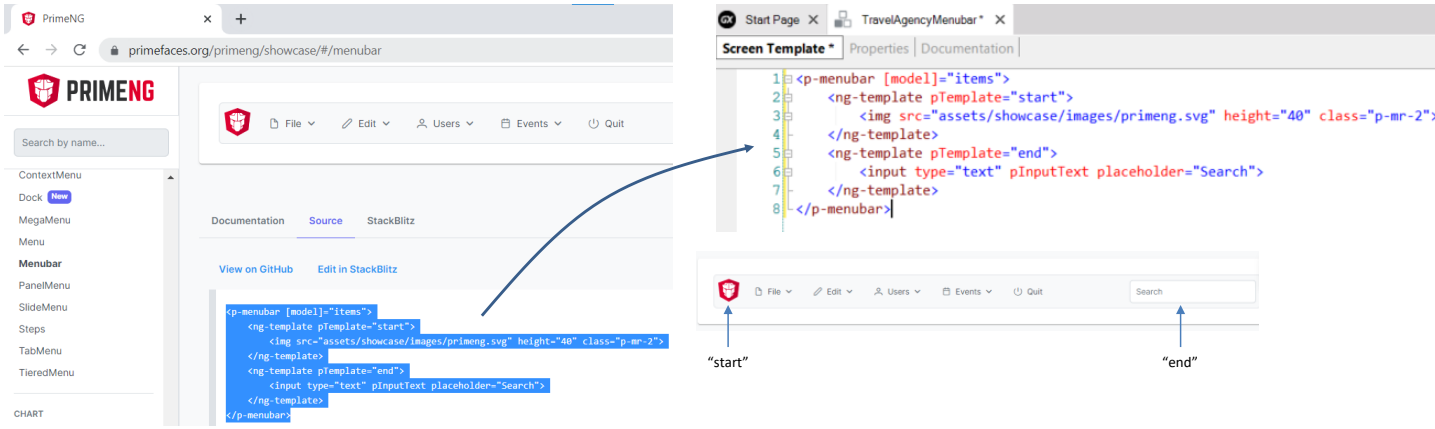
Criação do objeto User Control





Criamos um objeto do tipo User Control e o nomeamos TravelAgencyMenubar. O objeto possui 2 seções com as quais vamos trabalhar: Screen Template, onde definimos o código html do controle e Properties, onde atribuiremos valores a alguns dos elementos do HTML.

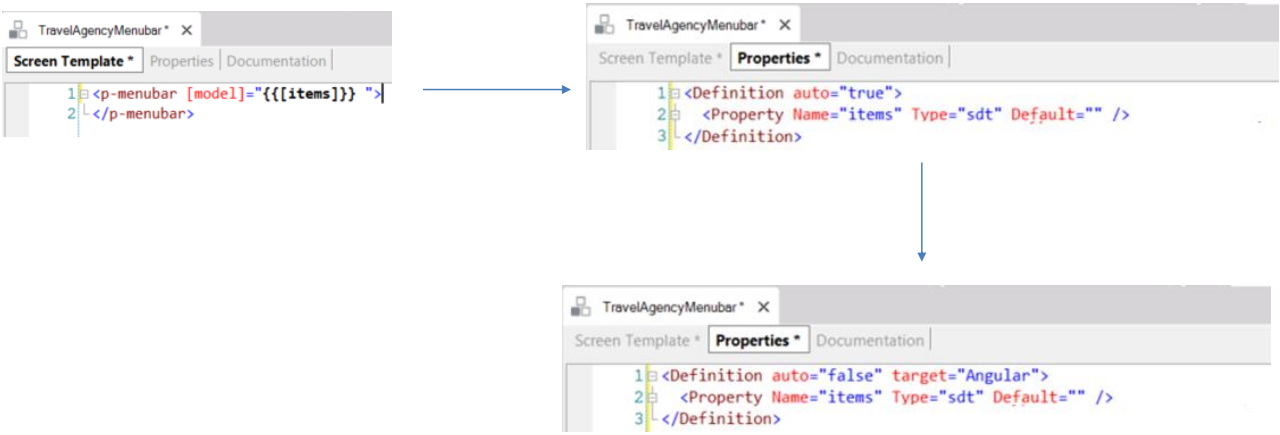
Copiamos e colamos o código HTML em nosso controle



Se voltamos para a página PrimeNG e vamos para Source, podemos ver o HTML do controle Menubar, então o copiamos e colamos na seção Screen Template do nosso controle.

Observamos que temos dois blocos ng-template, um com o nome "start" que define a área do ícone e outro com o nome "end" que define o texto para inserir uma pesquisa. Nenhuma das duas coisas nos interessa em nosso menu, então removemos estes blocos.

Substituímos dados fixos por elementos variáveis



Agora vamos trocar os dados fixos que vieram no HTML, por elementos que nos permitirão carregá-los dinamicamente com dados nossos, sejam eles fixos ou da base de dados.

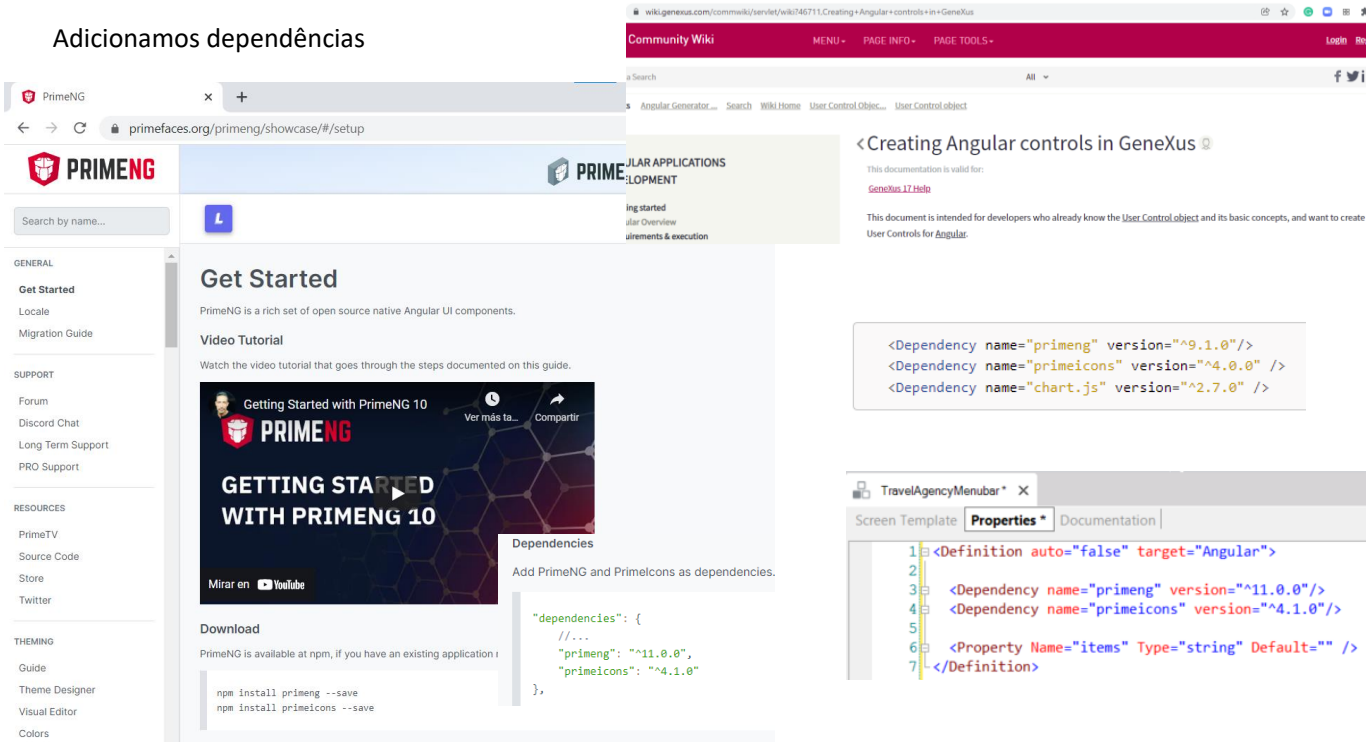
Substituímos o texto “items” na primeira linha pelo elemento {{[items]}} entre chaves duplas. Observemos que além das chaves duplas, o texto items está entre colchetes, que indica que o que irá ali é uma coleção com os itens do menu. Vamos carregar isto mais tarde com os dados do nosso menu.

Esta sintaxe com chave dupla é conhecida como “bigode” e nos permite substituir dados fixos por dados variáveis.

Salvamos e vamos para a guia Properties. Vemos que lá aparece uma property para a variável que adicionamos. Notamos que a property “items” tem tipo de dados “string”. Como vamos carregá-la com um SDT, atribuímos a ela o tipo de dados “sdt”.
Os tipos de dados que podemos usar, encontramos na documentação geral dos User Controls da Wiki.

Vamos alterar a cláusula Definition, colocando auto=“false” e adicionamos target = “Angular”. Isto é essencial para que o UC fique visível para ser adicionado em um objeto panel.

Adicionamos dependências



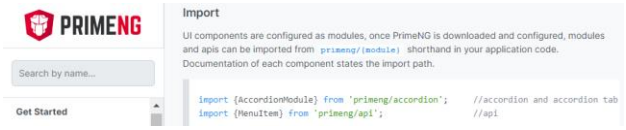
Agora devemos adicionar várias linhas para importar componentes do provedor, necessárias para que possa ser interpretado adequadamente o controle.

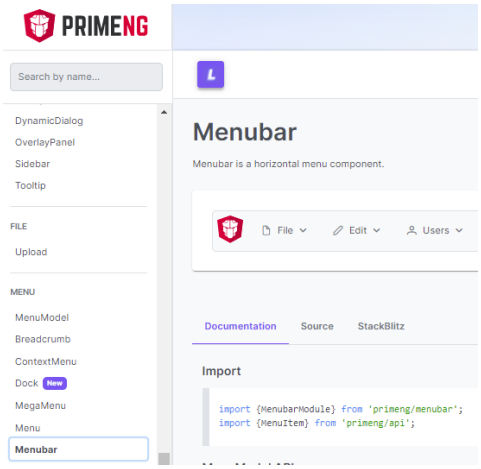
A primeira coisa que precisamos adicionar são as dependências dos pacotes a serem importados. Se vamos ao site de PrimeNG: **primefaces.org/primeng** e clicamos em Get Started é aberta uma página de guia do uso dos controles da biblioteca. Esta guia depende de cada fornecedor, mas todos eles possuem documentação que nos mostra como obter seus componentes.

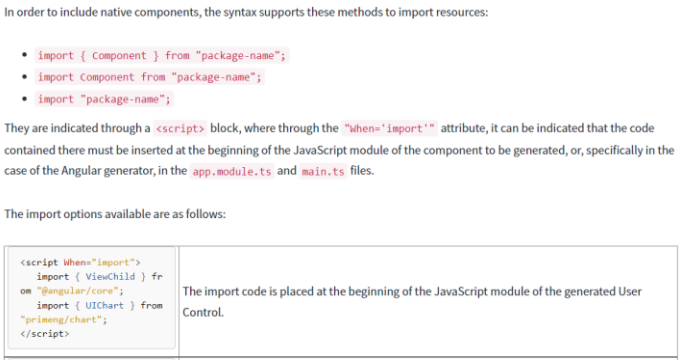
Na seção Download, indica os pacotes que devem ser instalados com npm. Estes pacotes são o que precisamos declarar em nosso user control como dependências. Se vamos para “Angular CLI Integration”, vemos um exemplo onde reconhecemos os pacotes mencionados anteriormente e nos fornece informações sobre a versão dos pacotes.

Se vamos para a página da wiki que nos orienta como utilizar um user control em Angular e vamos para a seção de Dependências, vemos a sintaxe que devemos usar. Adicionamos as dependências à seção de Properties de nosso User Control com os dados da página do provedor.

Adicionamos importação de recursos







Se voltamos à página de Get Started de PrimeNG, vemos que continua com a seção de Import, pois é necessário importar módulos. Aqui vemos um exemplo genérico, mas para saber quais são os módulos que precisamos para o controle Menubar, vamos para a página de informações do controle Menubar, que encontramos mais abaixo no menu da esquerda, abaixo da seção Menu.

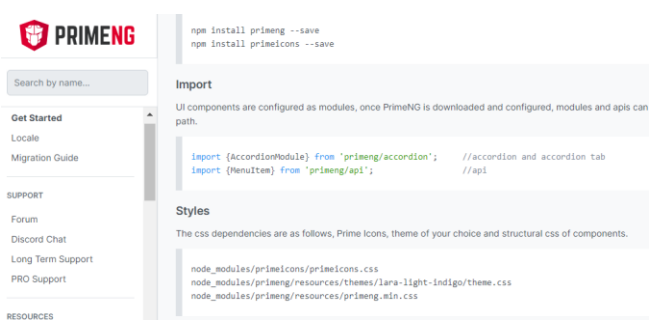
Vemos que os módulos que deveríamos importar são o MenubarModule e o MenuItem. Como não teremos subitens em nosso menu, somente precisamos do módulo Menubar.

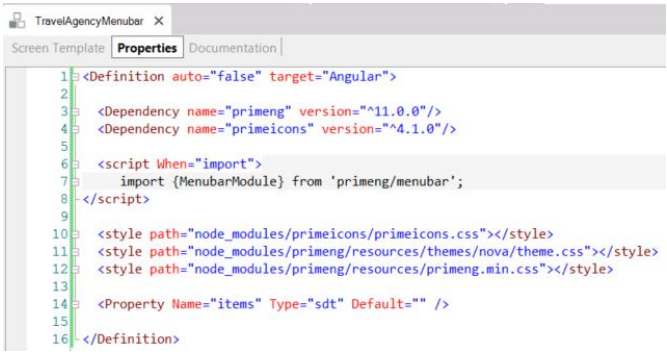
Os geradores de front-end como Angular adicionam maneiras avançadas de incluir recursos externos, utilizando a instrução “import”. Essa declaração pode ser utilizada tanto para incluir módulos JavaScript quanto para incluir, através de um pacote como webpack, outros tipos de recursos como imagens, arquivos SVG ou folhas de estilo.

Se formos à wiki, encontramos a sintaxe que devemos usar no GeneXus para o import. Primeiro temos uma descrição das diferentes sintaxes do comando import e mais abaixo temos alguns exemplos de uso. Podemos escolher qualquer um, pois no nosso caso não temos requisitos especiais de onde deve ficar o código gerado.

Adaptamos o exemplo da wiki para importar o módulo MenubarModule a partir de primeng/menubar e o adicionamos na janela de Properties.

Adicionamos estilos



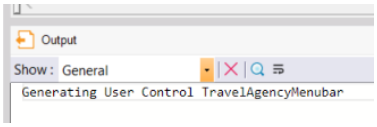


Style sheets

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.

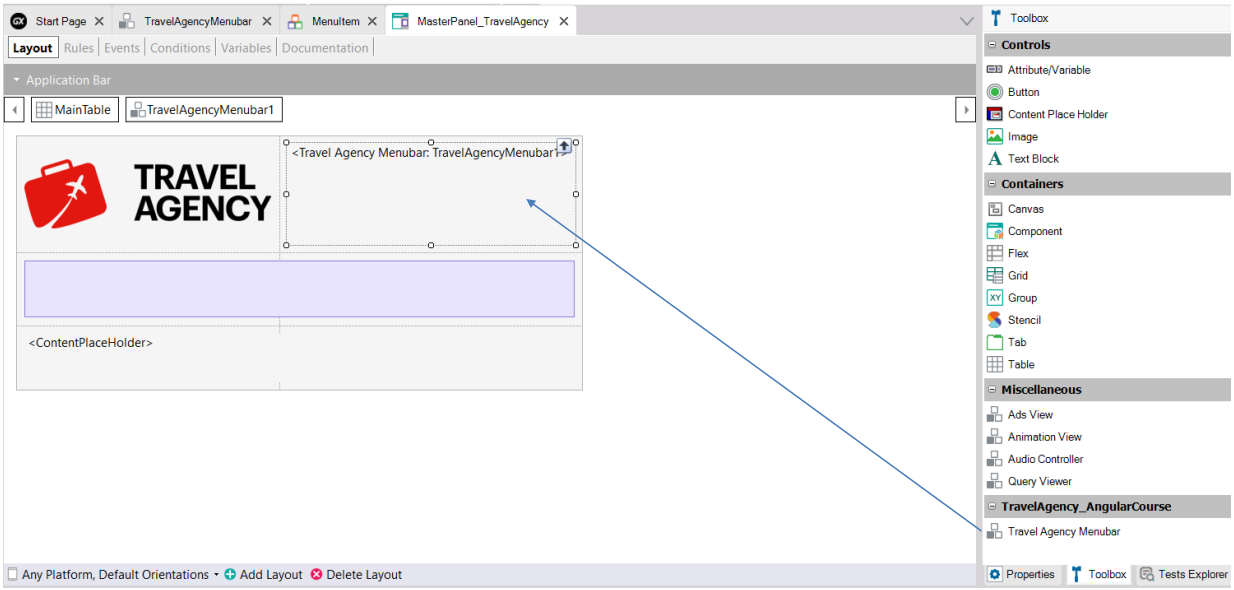


Voltamos mais uma vez à página Get Started de PrimeNG e vemos que nos indica que devemos carregar ícones e código CSS.
Novamente vamos à wiki para ver como escrevemos isso em GeneXus, usando a cláusula style e detalhando o path onde estarão localizados os CSS necessários,.

Agora sim ficou completa a definição do User Control, salvamos e vemos que foi gerado o user control TravelAgencyMenubar.

Agora vamos incorporá-lo em um objeto panel.

Inserimos o User Control que criamos no MasterPanel

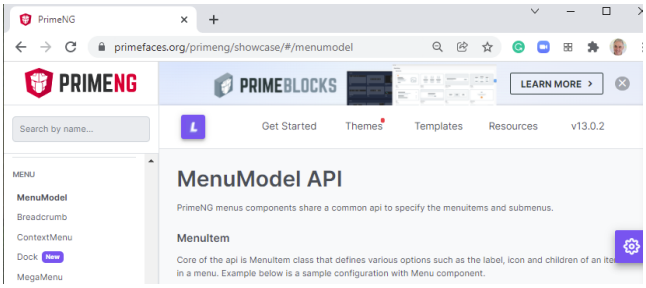


O local mais lógico para incluir um menu é em um objeto Master Panel, assim o menu está presente em todas as telas da aplicação e nos garante acesso rápido às diferentes partes. Então abrimos o objeto MasterPanel_TravelAgency que construímos.

Vemos que na barra de ferramentas aparece o user control TravelAgencyMenubar que criamos, então o arrastamos para o form.

É criado um user control chamado TravelAgencyMenubar1 e, como não precisamos da tabela à direita do logotipo, simplificamos um pouco o design.

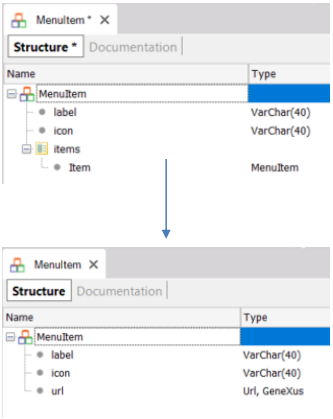
Criamos o SDT con la estructura necesaria




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

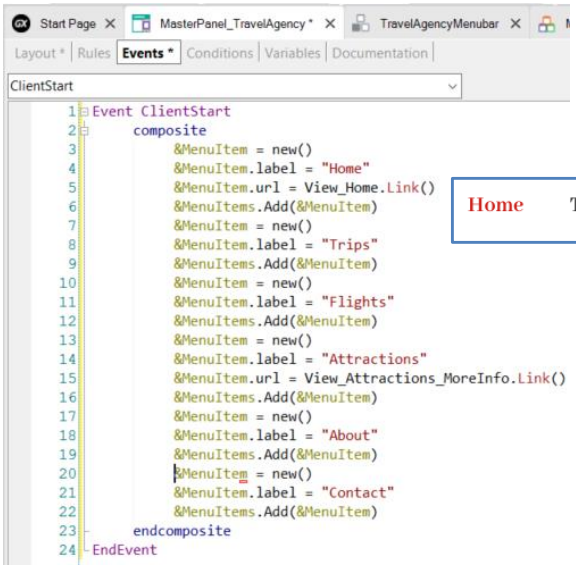


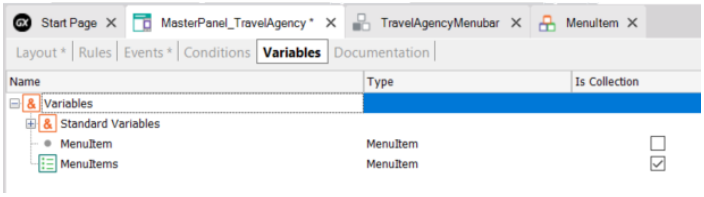
Como dissemos antes, os itens do menu vamos carregar usando um SDT. Para conhecer a estrutura que temos que dar ao SDT voltamos à documentação do controle Menubar e na seção MenuModel API vemos a documentação do MenuItem com suas propriedades.
Ali identificamos que um item do menu possui um id, um rótulo, um ícone, um comando, uma url, etc. Mais acima vemos um exemplo bem simples que usa unicamente a propriedade label e nos mostra que um item pode ter uma coleção de subitens e que os subitens possuem a mesma estrutura dos itens.

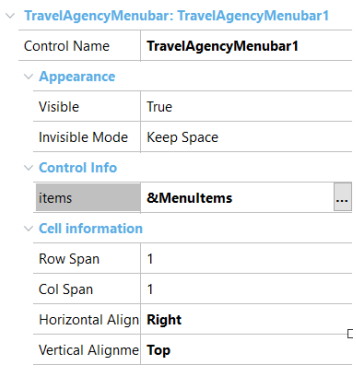
Assim podemos criar um SDT chamado MenuItem com uma estrutura semelhante que tenha um rótulo do tipo caractere, um ícone também do tipo caractere e uma coleção de itens do mesmo tipo que estamos criando: MenuItem.
Como em nosso caso não vamos usar subitens no menu, não adicionamos a coleção a ele. E adicionamos um elemento url que conterá a url do objeto que invocaremos quando pressionarmos o menu.

Cada API de cada provedor é diferente, portanto, devemos consultar a documentação do provedor para ver como montamos nosso User Control e quais estruturas devemos usar para carregar os dados necessários.

Carregamos o SDT nos eventos do MasterPanel







Agora vamos para a seção de variáveis do MasterPanel_TravelAgency. Criamos uma variável MenuItem que automaticamente fica do tipo MenuItem. A renomeamos MenuItems e a marcamos como coleção.
Em seguida, criamos outra variável MenuItem que usaremos para carregar cada item do menu para então adicioná-la à coleção de itens.

Abrimos a seção de eventos do panel e adicionamos um evento ClientStart. Escrevemos composite. Agora inserimos a variável &MenuItem e atribuímos um comando new() a ela.

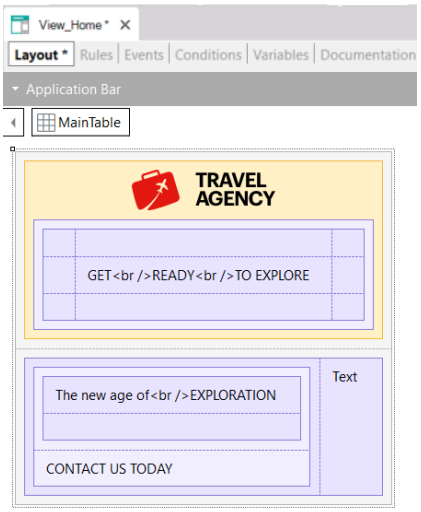
Em seguida, atribuímos ao membro label o valor “Home” e adicionamos o item de menu à coleção.

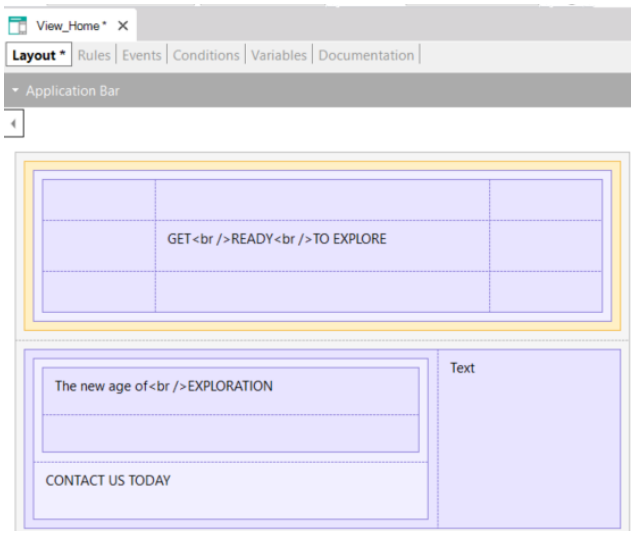
Fazemos o mesmo para os outros itens do menu.

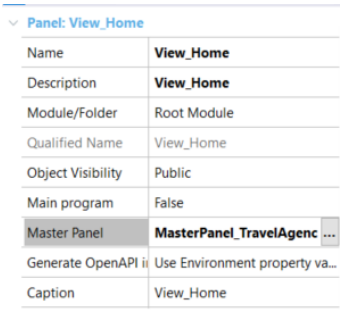
Fechamos o comando composite e o evento. Já temos carregada a variável de coleção de itens.

Agora voltamos ao layout do panel, selecionamos o user control TravelAgencyMenubar1 e em sua propriedade “items” selecionamos a variável &MenuItems. Com isto estamos dizendo ao user control onde ele deve buscar os dados para mostrar os itens.

Modificamos o objeto panel de startup e atribuímos a ele o Master Panel



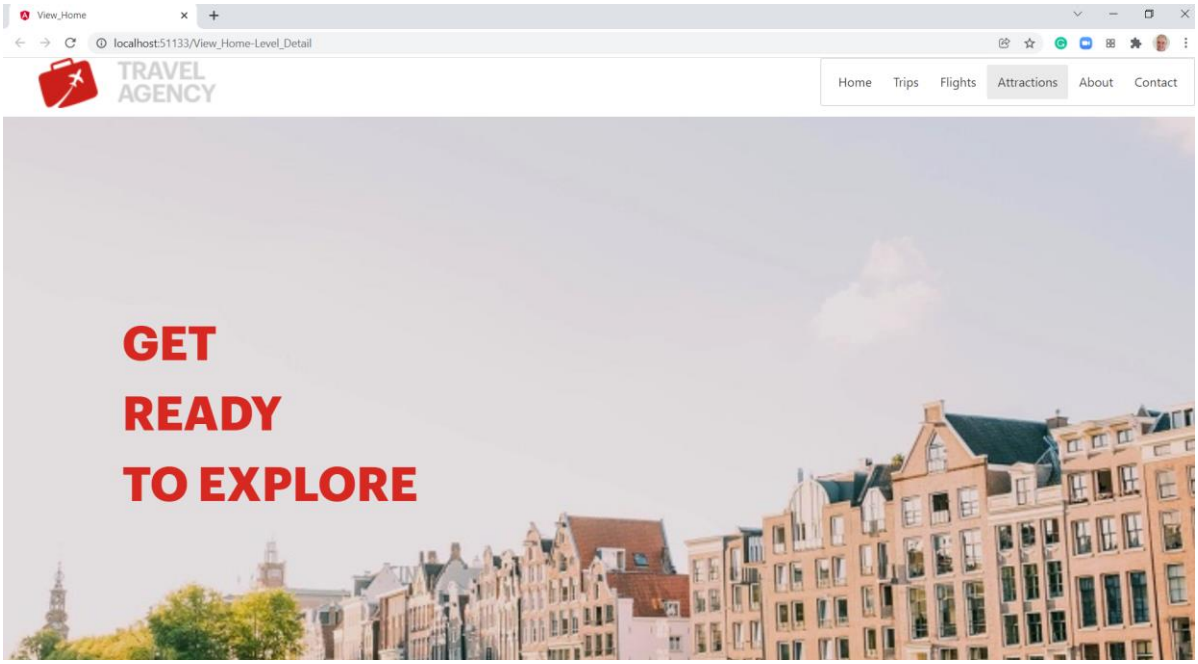




Para testar isto, abrimos o panel View_Home, que foi o objeto main que criamos como objeto de startup antes de usar o design que nos enviou o designer a partir do Sketch, removemos o ícone e atribuímos a propriedade Master Panel com o Master_Panel_TravelAgency, que já inclui o ícone e o menu.

Clicamos com o botão direito do mouse sobre o panel View_Home e selecionamos Run.

Execução do User Control



Vemos que é aberto o objeto View_Home, mostrando o menu na parte superior direita da tela.

Se passarmos o mouse sobre os botões, veremos que mudam de cor à medida que passamos sobre eles para indicar qual será selecionado se clicarmos.

Aqui não atribuímos objetos ao menu, pois já temos o menu que realmente vamos usar, que foi aquele criado a partir do Sketch.

O uso de user controls em uma aplicação gerada em Angular tem algumas considerações que são específicas da arquitetura do framework, e por isso devemos consultar a documentação de cada provedor de user controls ou se construirmos um por conta própria, devemos considerar a maneira de incluir os módulos, estilos e outros componentes necessários.

O fato de podermos definir nossos próprios user controls no GeneXus e não ficarmos apenas com os controles padrão, nos permite aumentar significativamente a experiência do usuário de nossas aplicações, já que podemos utilizar criações de muitas fontes e incorporá-las em nossos desenvolvimentos.

Em outros vídeos veremos como podemos também incorporar outras funcionalidades externas, acessando APIs.

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