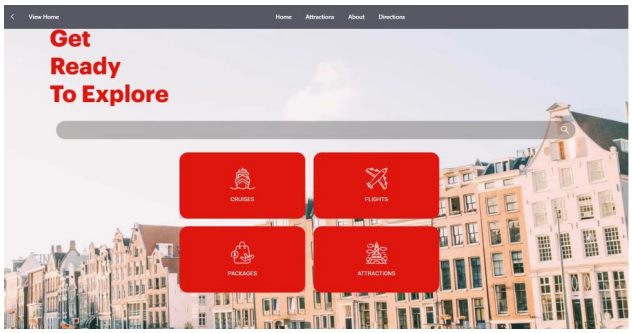
Design de uma aplicação Angular

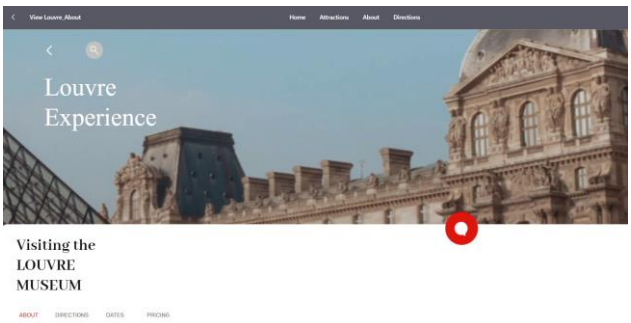
Introdução ao Design System Object

GeneXus™

Em outros vídeos mencionamos que uma melhoria que foi incorporada ao GeneXus é o Design System Object, como evolução do objeto Theme e das classes. Neste vídeo veremos como podemos utilizar este objeto para realizar definições de design em nossa aplicação.

Conceito de Design System





- Master Panel
- Panels
- Components
- Stencils
- Controls
- Patterns

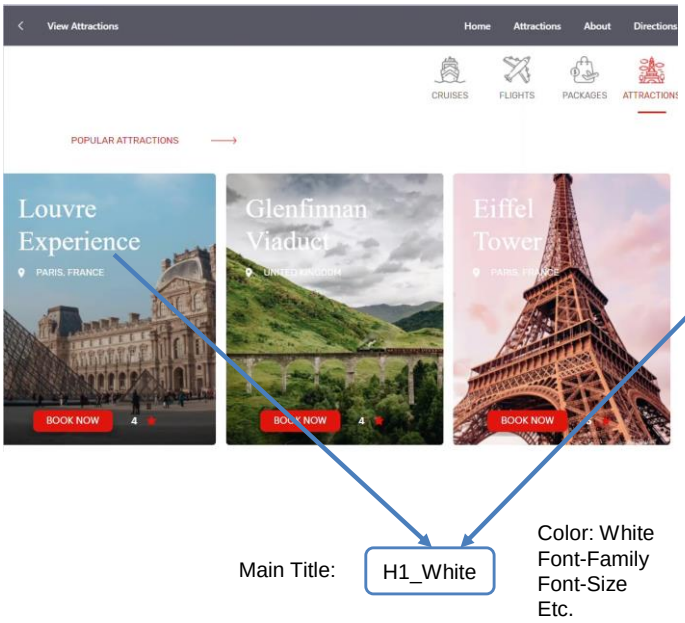
Se analisamos a tela inicial da aplicação final, vemos que existem certas escolhas de design que são mantidas em várias telas como, por exemplo, a cor dos botões que coincide com a do título, ou o fato de o título estar sobre uma imagem fundo, o que se repete na segunda tela e até o tipo e aparência dos textos, que é mantido nas diferentes telas.

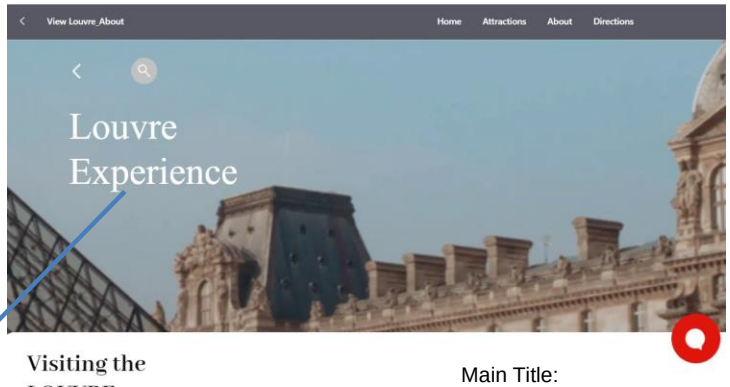
Esta regularidade em que prevalece um estilo, uma forma uniforme de apresentar as informações e padrões adotados para toda a aplicação é um Design System em funcionamento. Para abstrair estas definições, utilizamos uma série de objetos como os Master Panels, Panels, Components, Stencils, Controls e Patterns.

Em particular, torna-se muito importante a definição dos controles em tela como elementos principais da interface de usuário, independentemente da tela em que estejam. Essas definições eram realizadas até agora através do objeto Theme que agrupava as classes nas quais era definida a aparência dos controles da tela.

O Design System Object é uma evolução do objeto Theme, que reduz a distância entre designers e desenvolvedores, mas, mantendo o princípio geral de GeneXus de expressar o máximo, escrevendo o mínimo.

Definições do Design System





Visiting the LOUVRE MUSEUM

Main Title: H1_Black

Color: Black
Font-Family
Font-Size
Etc.

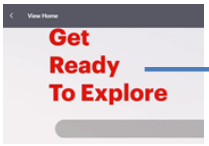
Main Title: H1_White

Color: White
Font-Family
Font-Size
Etc.

Por exemplo, consideremos o título branco que aparece em cima da foto de cada atração turística em todas as telas. Podemos abstrair esse estilo comum, atribuir-lhe o mesmo tamanho, fonte e cor e dar-lhe um nome, por exemplo: H1_White. Colocamos H1 justamente por ser um título principal e White por ser a cor que utilizaremos quando houver uma imagem de fundo, que é o caso.

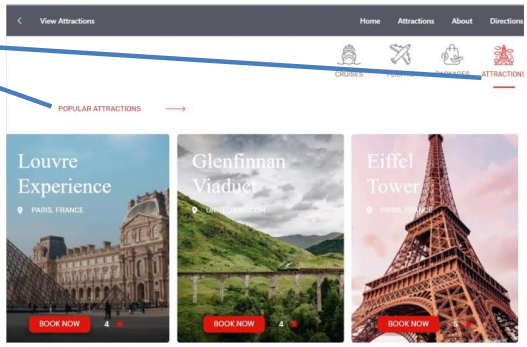
Da mesma forma, definimos este título aqui como H1_Black.

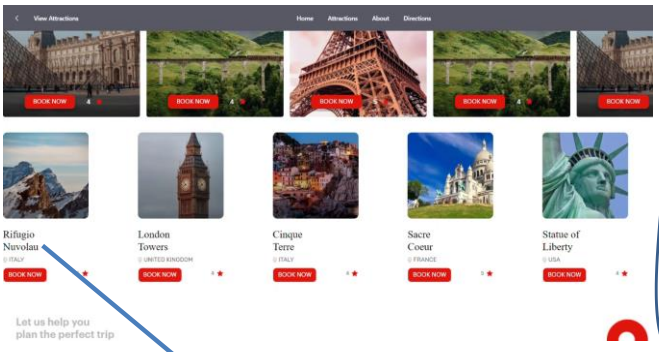
Definições do Design System

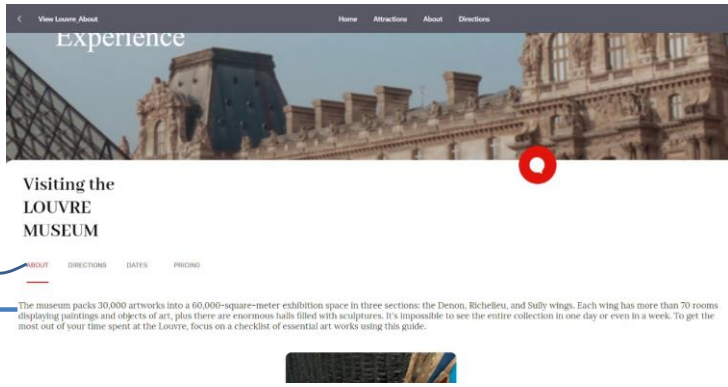


H1_Red

MainActionText







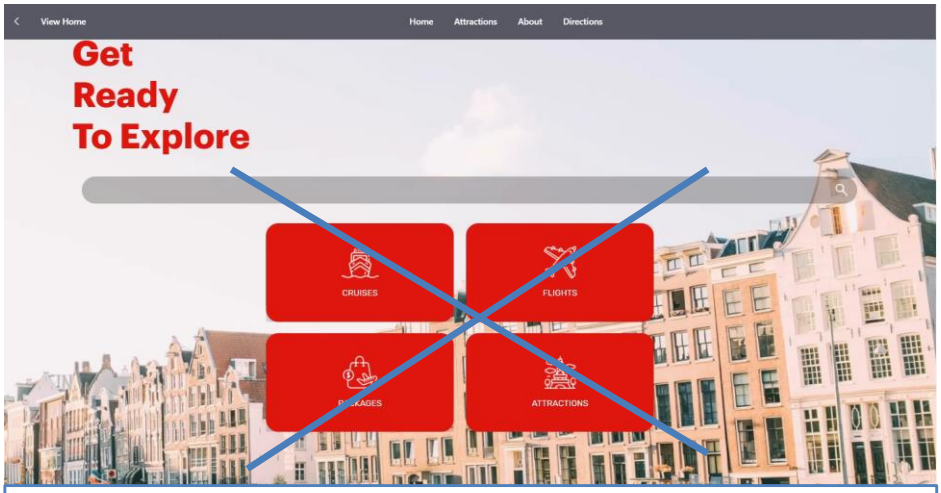
NormalText

Secundary Title: H2

Poderíamos continuar fazendo abstrações com outros textos, por exemplo definir o título grande da primeira página como H1_Red, este texto como título secundário: H2, estes aqui que estão em vermelho como textos que cumprem uma ação: MainActionText e este texto aqui como texto regular ou normal: NormalText.

Esses nomes que dei às minhas abstrações seriam então classes, nas quais poderei basear outros textos da aplicação. A identificação destas abstrações já significa estar modelando o Design System da aplicação.

O Design System Object nos permite concentrar todas essas abstrações em um só lugar, onde posso definir as propriedades de cada uma das classes.

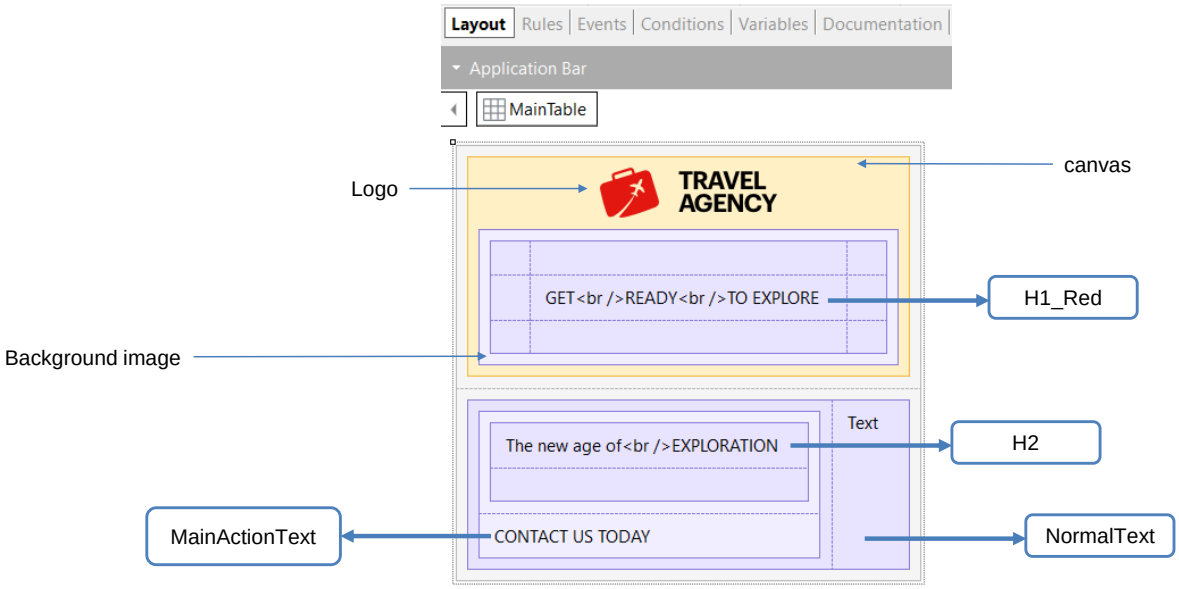


Informative text related to the Travel Agency

Vamos começar a levar estas classes para um objeto DSO mas primeiro vamos criar um panel home, que seja o objeto de startup de nossa aplicação, ou seja, aquele que é executado primeiro e serve como ponto de acesso para o restante das telas da aplicação.

O design é um pouco mais simples do que o que vimos, então não incluímos os botões por enquanto e incluímos alguns textos descritivos.

Construindo a tela inicial da aplicação



Aqui já criei o panel View_home, que dentro possui um controle canvas, que é como uma tabela, que nos permitirá conter objetos que são executados em diferentes camadas, um em cima do outro. É o efeito que queremos dar com o logotipo e os textos sendo vistos em cima da imagem de fundo.

Se formos aos controles que estão no canvas, vemos que agora todos possuem uma propriedade Zorder, que é o valor numérico que informa em qual camada se encontram. Quanto maior o número, estão mais acima.

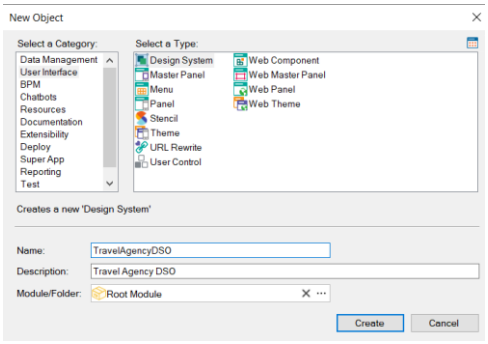
Dentro do canvas existe uma tabela que terá associada em sua propriedade background image a imagem de fundo que queremos. Esta tabela tem a propriedade Zorder com o valor 0, portanto, sua imagem de fundo estará abaixo de tudo do canvas. A imagem do logotipo tem Zorder = 1, então a veremos em cima da imagem de fundo.
O textblock com o título principal está em uma tabela que está dentro da primeira, então aparecerá sobre a imagem de fundo também.

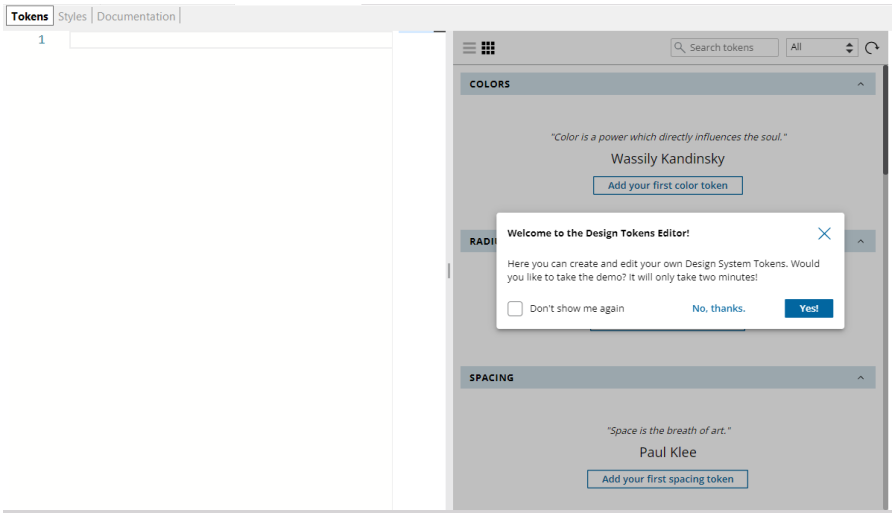
Abaixo do canvas temos uma tabela que contém à esquerda outra tabela com 2 textblocks, um com o subtítulo “The new age of exploration” e abaixo dele, um título de ação “Contact us today”. E à direita dessa tabela um textblock com um texto descritivo relativo à agência de viagens.

O título “Get ready to Explore” será vermelho com a classe H1_Red, o subtítulo “The new age of Exploration” terá uma classe H2, o link “Contact Us Today” será do tipo MainActionText e o texto descritivo será NormalText.

Todas essas definições faremos em um objeto Design System Object.

Criando um Design System Object



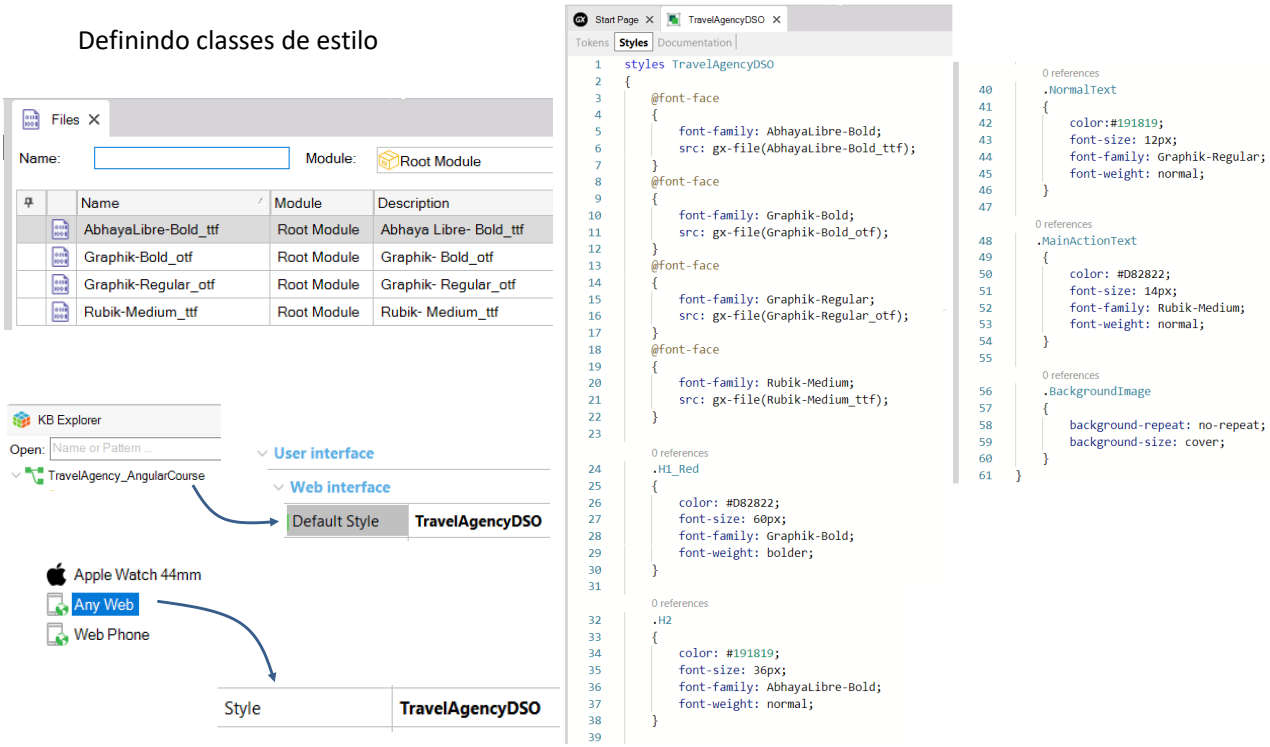


Agora vamos criar um Design System Object chamado TravelAgencyDSO.

Vemos que se posiciona em uma aba chamada Tokens e nos dá as boas-vindas ao Design Tokens Editor, convidando-nos a ver uma demonstração de ajuda.

Também vemos que é criada uma aba Styles, onde vamos definir as classes para o título principal, subtítulo, etc. que vimos antes.

Definindo classes de estilo



Começamos escrevendo a estrutura que conterá os styles.

Vamos criar a classe H1_Red que usaremos para definir nosso título principal.

O que fazemos é dar as características de estilo da classe. Por exemplo, para a cor, colocamos uma cor vermelha especial, que escrevemos como hexadecimal. Então dizemos que o tamanho da fonte é 60 pixels, a fonte definimos como Graphik-bold e o peso da fonte como bolder, um negrito intenso.

Estas propriedades que estamos definindo são idênticas às do CSS, mas não são as do CSS, pois existem algumas que são específicas de GeneXus e não estão definidas neste padrão. Uma coisa em que poderíamos pensar é que, se definimos o tamanho em pixels, será fixo para uma plataforma, mas estes pixels são convertidos automaticamente em dips. No caso de Angular, serão 60 pixels reais porque 1 dip é 1 pixel.

Se tivéssemos o arquivo que nos dão os designers, feito na ferramenta Sketch, poderíamos inspecionar o arquivo e obter as propriedades exatas que deveríamos colocar em nosso Design System Object.

Se a fonte do texto não for padrão, temos que carregá-la como arquivo em files e então declará-la em nosso DSO. Vamos adicionar as fontes que temos no arquivo files, então escrevemos font-family : AbhayaLibre-Bold, ponto e vírgula e

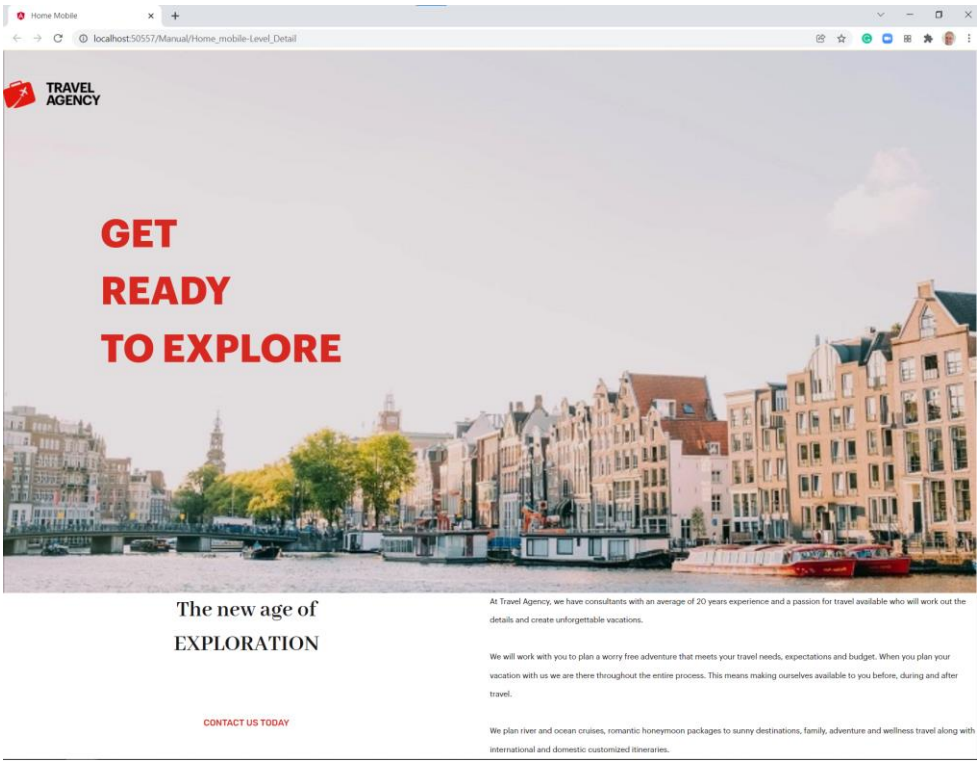
src: gx-file e entre parênteses colocamos o nome do arquivo de fonte: AbhayaLibre-Bold_ttf e terminamos com ponto e vírgula.
Fazemos o mesmo para as fontes Graphic-Bold, Graphic-Regular e Rubik-Medium.

Agora vamos definir o restante das classes que precisamos: H2, Normal Text e MainActionText.

Por último, para a imagem de fundo precisamos definir algumas propriedades, então criamos uma classe .BackgroundImage e definimos a propriedade background-repeat com o valor: no-repeat e background-size com o valor: cover.

Agora, para fazer com que nossa aplicação baseie seu design no Design System Object que criamos, vamos às propriedades da versão da KB e na propriedade Default Style atribuímos o design system TravelAgencyDSO.
Então vamos para o nó Platform e como este design vamos utilizar em nossa aplicação gerada em Angular, na plataforma Any Web atribuímos a propriedade Style no DSO que criamos.

Vamos executar o panel para ver o que fizemos.



Com a aplicação em execução, observamos que foram aplicados os estilos corretamente, pois cada texto saiu com o estilo que queríamos.

Definindo tokens

Tokens | Styles * | Documentation

```
tokens TravelAgencyDSO {

    #colors
    {
        OnSurface: #191819;
        Surface: #FFFFFF;
        Highlight: #D82822;
    }

    // #fontSizes
    // {
    //  MainTitle: 60px;
    //      SubTitle: 36px;
    //      NormalText: 12px;
    //      LinkText: 14px;
    // }
}
```

Start Page | TravelAgencyDSO

Tokens | Styles | Documentation

```
styles TravelAgencyDSO
{
    @font-face
    {
        font-family: AbhayaLibre-Bold;
        src: gx-file(AbhayaLibre-Bold_ttf);
    }
    @font-face
    {
        font-family: Graphik-Bold;
        src: gx-file(Graphik-Bold_otf);
    }
    @font-face
    {
        font-family: Graphik-Regular;
        src: gx-file(Graphik-Regular_otf);
    }
    @font-face
    {
        font-family: Rubik-Medium;
        src: gx-file(Rubik-Medium_ttf);
    }

    0 references
    .H1_Red
    {
        color: $colors.Highlight;
        font-size: 60px;
        font-family: Graphik-Bold;
        font-weight: bolder;
    }

    0 references
    .H2
    {
        color: $colors.OnSurface;
        font-size: 36px;
        font-family: AbhayaLibre-Bold;
        font-weight: normal;
    }

    0 references
    .NormalText
    {
        color:$colors.OnSurface;
        font-size: 12px;
        font-family: Graphik-Regular;
        font-weight: normal;
    }

    0 references
    .MainActionText
    {
        color: $colors.Highlight;
        font-size: 14px;
        font-family: Rubik-Medium;
        font-weight: normal;
    }

    0 references
    .BackgroundImage
    {
        background-repeat: no-repeat;
        background-size: cover;
    }
}
```

Se voltamos ao DSO, vemos que existem determinados valores, como as cores ou os tamanhos de texto, que poderíamos abstrair e dar um nome a eles.
Os elementos deste nível de abstração mais alto, chamaremos de tokens.

Vamos para a seção de tokens e definiremos um token como constante de cor para os textos que devem ir sobre uma superfície de fundo, como é o caso quando estavam sobre uma imagem. Para fazer isso, na seção de Colors, pressionamos o botão "Add your first color token" e vemos que se abre uma estrutura para começar a escrever.

O nomeamos: OnSurface e copiamos o valor hexadecimal do valor da cor que havíamos definido no estilo.
Também podemos definir uma cor para o fundo que é um tipo de branco, por exemplo, Surface, e uma cor Highlight para os textos que são vermelhos, como o título H1_red e para o texto que realiza uma ação.

Agora podemos ir aos estilos e naquelas classes correspondentes modificamos o valor fixo da cor, pelo token correspondente.
Se escrevermos o sinal de $ podemos escolher o token correspondente em cada caso. Por exemplo na classe H1_Red em color escrevemos $ e escolhemos o nome do token, colors e depois escolhemos o valor OnSurface.

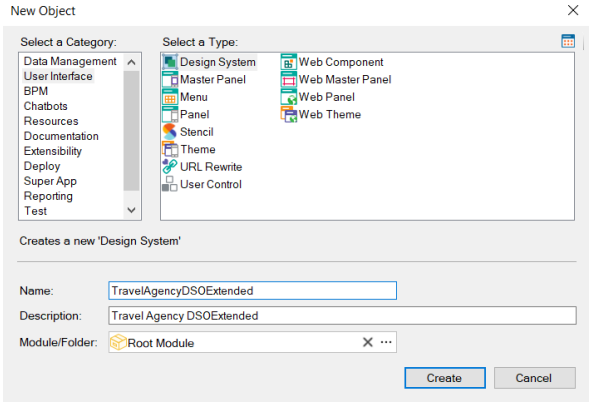
E fazemos o mesmo para todas as cores que estavam com valores fixos…

Desta forma, se eu alterar o token, todos os estilos que são baseados no mesmo token serão alterados ao mesmo tempo, o que me dá uma maior flexibilidade e

menor possibilidade de erros.

Também poderíamos definir outros tipos de tokens como, por exemplo, para o tamanho da fonte. Se filtrarmos aqui por categoria e escolhermos fontSizes vemos que é adicionado o token para começarmos a escrever. Podemos definir um token para o tamanho da fonte do título principal, outro para o subtítulo, etc. e então, como fizemos antes, vamos para os estilos e mudaríamos para o valor do token correspondente.

Estendendo o DSO com outro DSO



Tokens | **Styles *** | Documentation

```
1  styles TravelAgencyDSOExtended
2  {
3      @import TravelAgencyDSO;
4
5      //      .H1_Red
6      // {
7      //      color: $colors.Highlight;
8      //      font-size: 60px;
9      //      font-family: Graphik-Bold;
10     //      font-weight: bolder;
11     // }
12
13     .H1_Blue
14     {
15         @include H1_Red;
16         color: blue;
17     }
18
```

Uma coisa que podemos fazer em um DSO é importar as definições realizadas em outro DSO.

Por exemplo, podemos criar um DSO chamado TravelAgencyDSOExtended e importar do DSO que construímos anteriormente.

Para isso usamos a regra @import e o nome do DSO que queremos importar. Neste caso, podemos reutilizar classes ou tokens definidos no DSO que importamos.

Também podemos sobrescrever do DSO principal, alguma propriedade de uma classe ou token importado e, nesse caso, será tomada aquela que definimos no DSO destino. Ou podemos herdar de uma classe do DSO que importamos e criar outra a partir da anterior.

Para fazer isso usamos a regra @include com o nome da classe a sobrescrever e adicionamos apenas aquela propriedade que queremos alterar, o restante permanecerá com o valor do DSO importado.

Se precisássemos executar a aplicação em outra plataforma, por exemplo, em um dispositivo móvel, poderíamos importar o DSO e alterar apenas aquelas coisas que precisamos adaptar para a nova plataforma e reutilizar todo o resto.

Para saber mais sobre este tema:

https://wiki.genexus.com/commwiki/servlet/wiki?47375

Neste vídeo, resumimos como criar e modificar um Design System Object e estendê-lo importando definições de outro DSO.

Outro caminho que podemos realizar é importar um design completo feito pelos designers, a partir do Sketch e isso veremos em outro vídeo.

Se quiser saber mais sobre o Design System Object, sugerimos que visite a seguinte página da wiki:
https://wiki.genexus.com/commwiki/servlet/wiki?47375

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