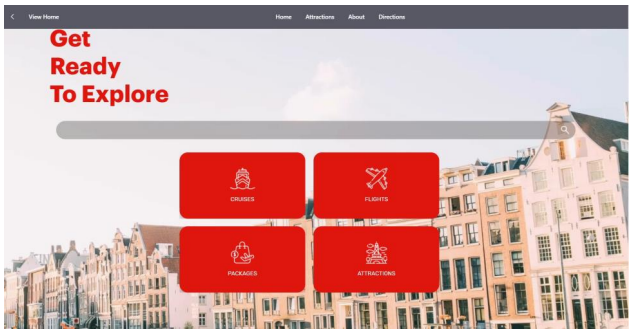
Diseño de una aplicación Angular

Introducción al Design System Object

GeneXus™

En otros videos mencionamos que una mejora que se incorporó a GeneXus es el Design System Object, como evolución del objeto Theme y de las clases. En este video veremos como podemos utilizar este objeto para realizar definiciones de diseño en nuestra aplicación.

Concepto de Design System



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

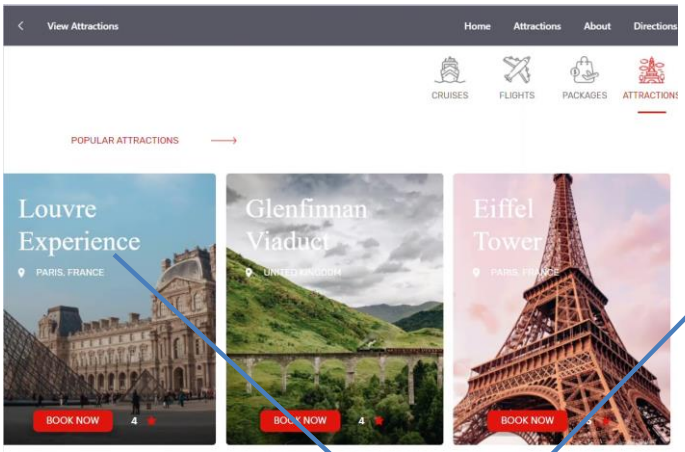
Si analizamos la pantalla inicial de la aplicación final, vemos que hay determinadas decisiones de diseño que se mantienen en varias pantallas, como por ejemplo el color de los botones que coincide con el del título, o el hecho de que el título esté sobre una imagen de fondo, cosa que se repite en la segunda pantalla e incluso el tipo y apariencia de los textos, que se mantiene en las distintas pantallas.

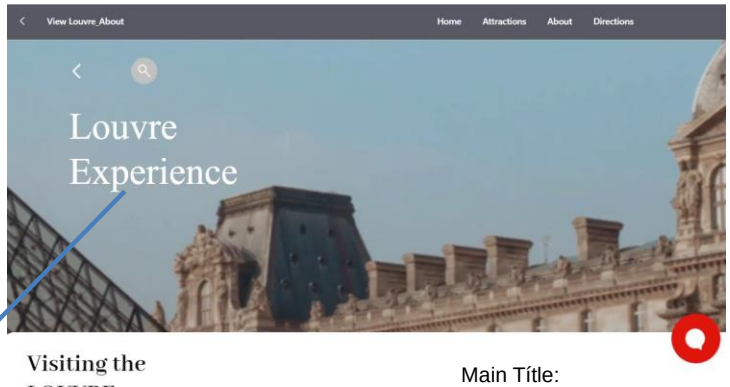
Esa regularidad en la que prevalece un estilo, una forma uniforme de presentar la información y estándares adoptados para toda la aplicación es un Design System en funcionamiento. Para abstraer estas definiciones, utilizamos una serie de objetos como los Master Panels, Panels, Components, Stencils, Controls y Patterns.

En particular, toma mucha importancia la definición de los controles en pantalla como elementos principales de la interfaz de usuario, independientemente de la pantalla en la que se encuentren. Esas definiciones se realizaban hasta ahora mediante el objeto Theme que agrupaba las clases en las que se definía la apariencia de los controles de la pantalla.

El Design System Object es una evolución del objeto Theme, que reduce la brecha entre diseñadores y desarrolladores, pero manteniendo el principio general de GeneXus de expresar lo máximo, escribiendo lo mínimo.

Definiciones del Design System





Visiting the LOUVRE MUSEUM

Main Título:

H1_Black

Color: Black
Font-Family
Font-Size
Etc.

Main Título:

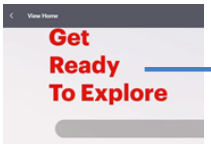
H1_White

Color: White
Font-Family
Font-Size
Etc.

Por ejemplo consideremos el título blanco que aparece arriba de la foto de cada atracción turística en todas las pantallas. Podemos abstraer ese estilo común, asignarle al mismo tamaño, tipo de letra y color y darle un nombre, por ejemplo: H1_White.
Le ponemos H1 justamente porque es un título principal y White porque es el color que utilizaremos cuando hay una imagen de fondo, que es el caso.

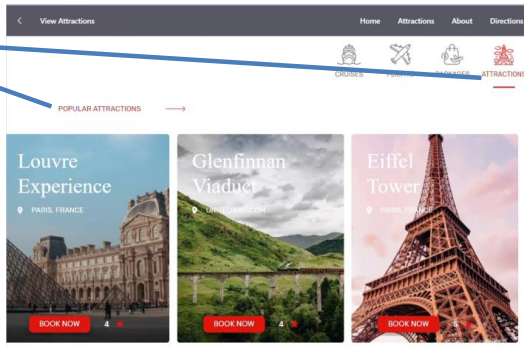
De la misma manera definimos este título de aquí como H1_Black.

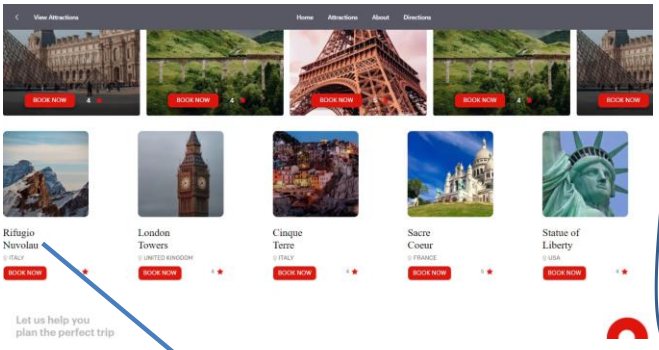
Definiciones del Design System

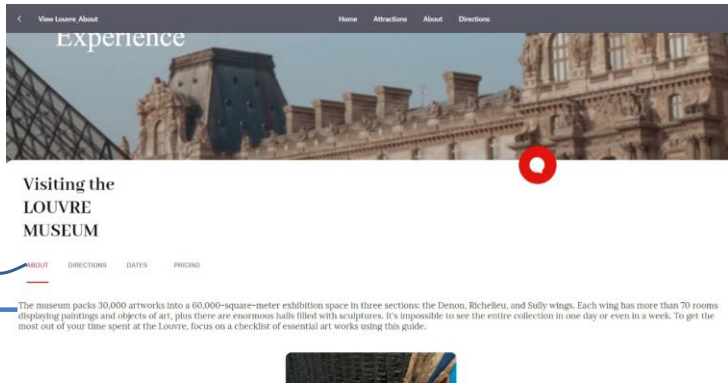


H1_Red

MainActionText







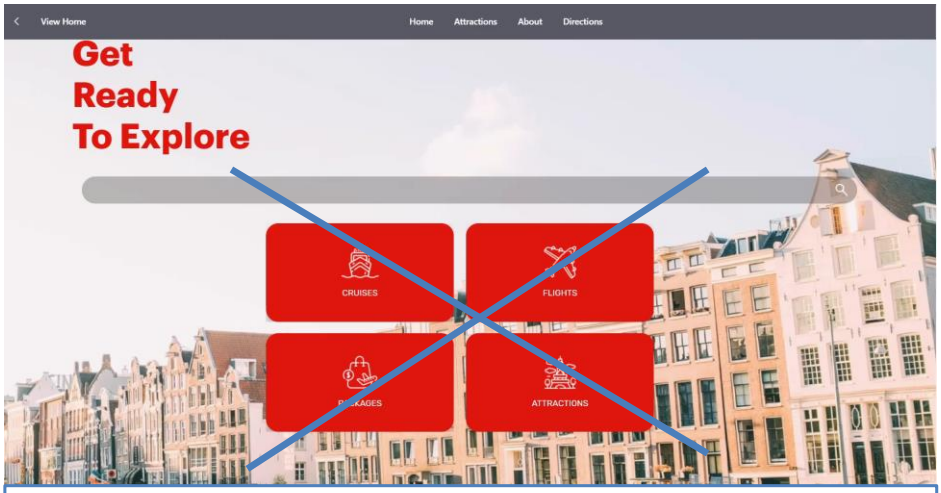
NormalText

Secundary Título: H2

Podríamos seguir haciendo abstracciones con otros textos, por ejemplo definir el título grande de la primera página como H1_Red, este texto como título secundario: H2, estos de aquí que están en rojo como textos que cumplen una acción: MainActionText y este texto aquí como texto regular o normal: NormalText.

Esos nombres que di a mis abstracciones serían entonces clases, en los que podré basar otros textos de la aplicación. La identificación de estas abstracciones ya es estar modelando el Design System de la aplicación.

El Design System Object nos permite concentrar todas esas abstracciones en un solo lugar, donde puedo definir las propiedades de cada una de las clases.

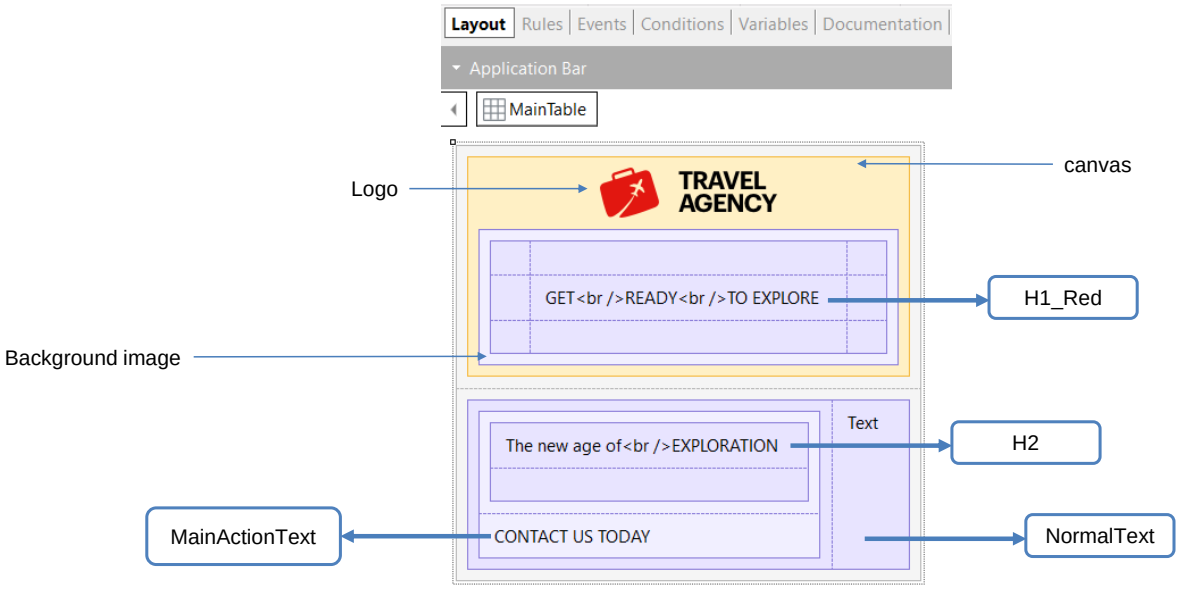


Informative text related to the Travel Agency

Vamos a comenzar a llevar estas clases a un objeto DSO pero antes vamos a crear un panel home, que sea el objeto de startup de nuestra aplicación, es decir el que se ejecute primero y sirva de punto de acceso al resto de las pantallas de la aplicación.

El diseño es un poco más sencillo que el que vimos, así que no incluimos a los botones por ahora y sí unos textos descriptivos.

Construyendo la pantalla inicial de la aplicación



Aquí yo tengo ya creado al panel View_home, que dentro tiene un control canvas, que es como una tabla, que nos va a permitir contener objetos que se ejecuten en distintas capas, uno encima de otro. Es el efecto que queremos dar con el logo y los textos viéndose por encima de la imagen de fondo.

Si vamos a los controles que están en el canvas vemos que ahora todos tienen una propiedad Zorder que es el valor numérico que dice en qué capa se encuentran. Cuanto más alto el número están más arriba.

Dentro del canvas hay una tabla que tendrá asociada en su propiedad background image a la imagen de fondo que queremos. Esta tabla tiene la propiedad Zorder con el valor 0, por lo que su imagen de fondo estará abajo del todo del canvas. La imagen del logo tiene Zorder = 1, por lo que la veremos encima de la imagen de fondo.
El textblock con el título principal está en una tabla que está dentro de la primera, por lo que saldrá sobre la imagen de fondo también.

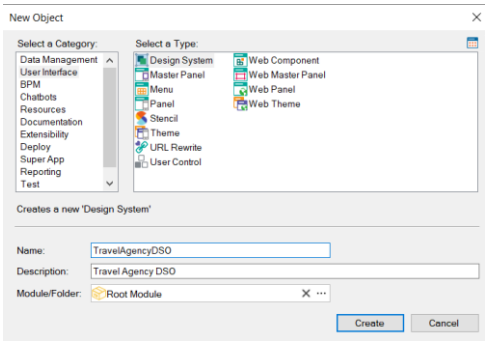
Abajo del canvas tenemos una tabla que contiene a la izquierda otra tabla con 2 textblocks, uno con el subtítulo "The new age of exploration" y debajo de éste, un título de acción "Contact us today". Y a la derecha de esa tabla un textblock con un texto descriptivo relativo a la agencia de viajes.

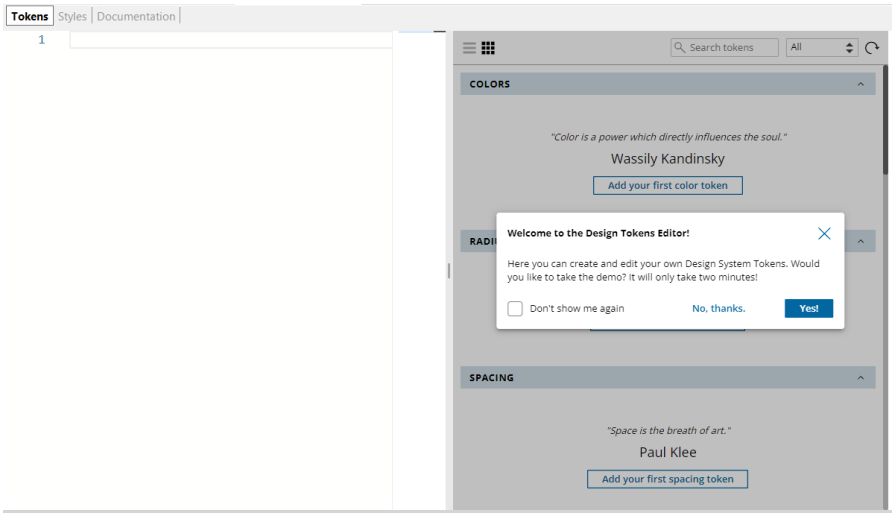
El título "Get ready to Explore" será rojo con la clase H1_Red, el subtítulo "The new age of Exploration" tendrá una clase H2, el link de "Contact Us

Today″ será del tipo MainActionText y el texto descriptivo será NormalText.

Todas esas definiciones las haremos en un objeto Design System Object.

Creando un Design System Object



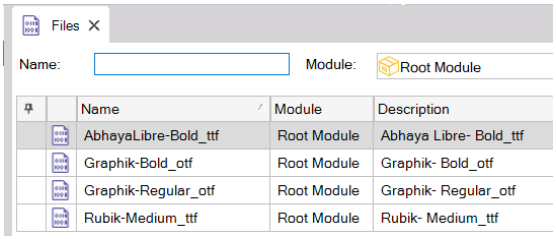


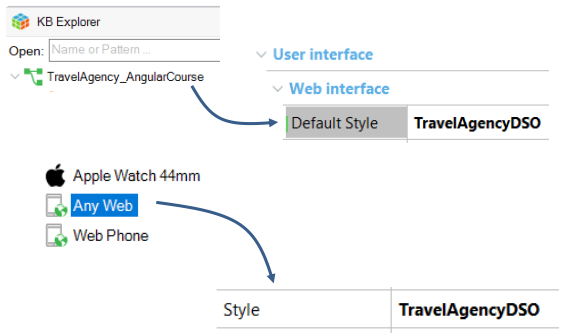
Ahora vamos a crear un Design System Object de nombre TravelAgencyDSO.

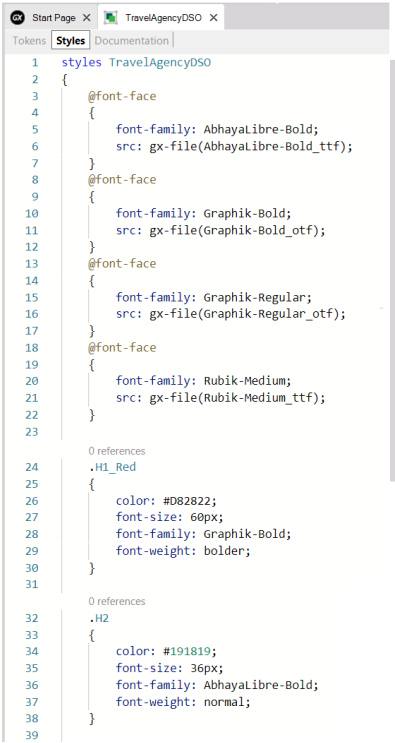
Vemos que se posiciona en una solapa llamada Tokens y nos da la bienvenida al Design Tokens Editor, invitándonos a ver una demo de ayuda.

También vemos que se crea una solapa Styles, donde vamos a definir las clases para el título principal, subtítulo, etc. que vimos antes.

Definiendo clases de estilo







Comenzamos escribiendo la estructura que va a contener los styles.

Vamos a crear a la clase H1_Red que usaremos para definir nuestro título principal.

Lo que hacemos es dar las características de estilo de la clase. Por ejemplo, al color le ponemos un color rojo especial, que lo escribimos como hexadecimal. Luego el tamaño de la fuente le decimos que sea 60 pixels, a la fuente lo definimos como Graphik-bold y al peso de la fuente como bolder, una negrita intensa.

Estas propiedades que estamos definiendo son idénticas a las de CSS, pero no son las de CSS, porque hay algunas que son propias de GeneXus y no están definidas en este estándar. Una cosa que podríamos pensar es que si definimos el tamaño en pixels va a ser fijo para una plataforma, pero estos pixels se convierten automáticamente a dips. En el caso de Angular, sí serán 60 pixels reales porque 1 dip es 1 pixel.

Si tuviéramos el archivo que nos dan los diseñadores, hecho en la herramienta Sketch, podríamos inspeccionar el archivo y obtener las propiedades exactas que deberíamos poner en nuestro Design System Object.

Si la fuente del texto no es estándar, tenemos que cargarla como archivo en files, y luego declararla en nuestro DSO.

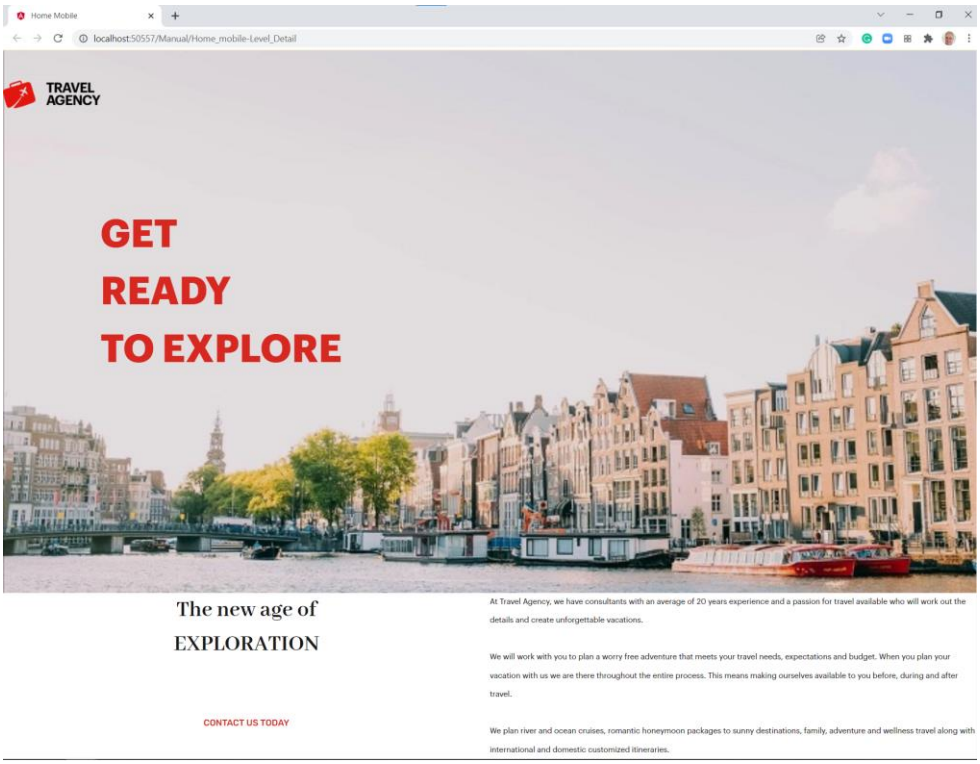
Vamos agregar las fuentes que tenemos en el archivo files, así que escribimos font-family : AbhayaLibre-Bold, punto y coma, y src: gx-file y entre paréntesis ponemos el nombre del archivo del fuente: AbhayaLibre-Bold_ttf y terminamos con punto y coma.
Hacemos lo mismo para los fuentes Graphic-Bold, Graphic-Regular y Rubik-Medium.

Ahora vamos a definir a resto de las clases que necesitamos: H2, Normal Text y MainActionText.

Por último, para la imagen de fondo debemos setear algunas propiedades, así que nos creamos una clase .BackgroundImage y ponemos la propiedad background-repeat en el valor: no-repeat y background-size en el valor: cover.

Ahora, para hacer que nuestra aplicación base su diseño en el Design System Object que creamos, vamos a las propiedades de la versión de la KB y en la propiedad Default Style le asignamos el design system TravelAgencyDSO.
Luego vamos al nodo Platform y como este diseño lo vamos a utilizar en nuestra aplicación generada en Angular, en la plataforma Any Web asignamos la propiedad Style en el DSO que creamos.

Ejecutemos el panel para ver lo que hicimos.



Con la aplicación en ejecución, observamos que se aplicaron los estilos correctamente, ya que cada texto salió con el estilo que queríamos.

Definiendo 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;
    }
}
```

Si volvemos al DSO, vemos que hay ciertos valores como los colores o los tamaños de texto, que podríamos abstraerlos y darles un nombre.
A los elementos de este nivel de abstracción más alto, le vamos a llamar tokens.

Vamos a la sección de tokens y definiremos un token como constante de color para los textos que deban ir sobre una superficie de fondo, como es el caso de cuando estaban sobre una imagen. Para hacerlo, en la sección de Colors, presionamos el botón "Add your first color token" y vemos que se abre una estructura para empezar a escribir.

Le ponemos de nombre: OnSurface y el copiamos el valor hexadecimal del valor del color que habíamos definido en el estilo.
También podemos definir un color para el fondo que es un tipo de blanco, por ejemplo Surface y un color Highlight para los textos que son rojos, como el título H1_red y para el texto que realiza una acción.

Ahora podemos ir a los estilos y en aquellas clases que corresponda modificamos el valor fijo del color, por el token correspondiente.
Si escribimos el signo de $ podemos elegir el token que corresponde en cada caso. Por ejemplo en la clase H1_Red en color escribimos $ y elegimos el nombre del token, colors y luego elegimos el valor OnSurface.

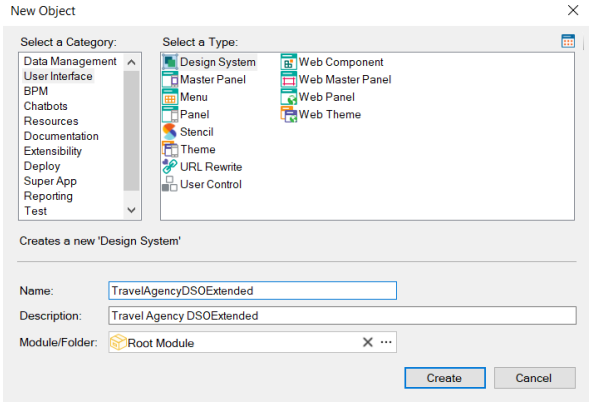
Y hacemos lo mismo para todos los colores que estaban con valores fijos...

De esta forma, si yo cambio el token, todos los estilos que están basados

en el mismo token cambiarán a la vez, lo que me da una mayor flexibilidad y menor posibilidad de errores.

También podríamos definir otros tipos de tokens, como por ejemplo para el tamaño de fuente. Si filtramos aquí por categoría y elegimos fontSizes vemos que se agrega el token para que empecemos a escribir. Podemos definir un token para el tamaño de fuente del título principal, otro para el subtítulo, etc. y luego como hicimos antes vamos a los estilos y cambiaríamos por el valor de token correspondiente.

Extendiendo el DSO con otro 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
```

Una cosa que podemos hacer en un DSO es importar las definiciones realizadas en otro DSO.

Por ejemplo podemos crear un DSO llamado TravelAgencyDSOExtended e importar del DSO que construimos antes.

Para eso usamos la regla @import y el nombre del DSO que queremos importar. En este caso, podemos reutilizar clases o tokens definidos en el DSO que importamos.

También podemos sobreescribir del DSO principal, alguna propiedad de una clase o token importado, y en ese caso se va a tomar la que definamos en el DSO destino. O podemos heredar de una clase del DSO que importamos y crear otra a partir de la anterior.

Para hacer eso usamos la regla @include con el nombre de la clase a sobreescribir y agregamos solamente a aquella propiedad que queremos cambiar, el resto permanecerán con el valor del DSO importado.

Si se necesitara ejecutar la aplicación en otra plataforma, por ejemplo en un dispositivo móvil, se podría importar el DSO y cambiar solamente aquellas cosas que necesitemos adaptar para la nueva plataforma y reutilizar todo el resto.

Para conocer más sobre este tema:

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

En este video hemos resumidos cómo crear y modificar un Design System Object y extenderlo importando definiciones de otro DSO.

Otro camino que podemos realizar es importar un diseño completo hecho por los diseñadores, desde Sketch y eso lo veremos en otro video.

Si desea conocer más sobre el Design System Object, le sugerimos visitar la siguiente página del wiki:
https://wiki.genexus.com/commwiki/servlet/wiki?47375

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