Crea tu propio tema

**Datos primordiales.**

Para que los comandos descritos a continuación funcionen recuerda haber generado el proyecto contenedor de widget con la **collection-retail-banking-2.10.0**, por los pasos a seguir es duplicar el tema existente y poder modificarlo a tu parecer.

**Instalar Sass Plugin y bb-customize**

El tema de Backbase contiene archivos Sass. Por lo tanto, debemos instalar el complemento Sass para compilarlo durante el **bb-package**.

Como vamos a crear nuestro tema basado en el tema Backbase, **bb-customize** ejecutar el comando para duplicar el tema.

npm install -g @bb-cli/bb-build-plugin-sass @bb-cli/bb-customize

**1.- Duplicar el tema existente usando bb-customize**

1. Acceder a la carpeta **statics**
2. Copie el tema predeterminado de la carpeta de origen para crear su propio tema.

bb-customize item theme-bb-retail collection-training/src --new-name theme-training-default

**Establezca el título y la versión en model.xml**

1. Abrir la carpeta **statics/collection-training/src/ theme-training-default**
2. Edite el **archivo model.xml**
3. Reemplazar **Theme for Retail Banking Collection** por **Theme Training Default**
4. Reemplazar el valor de la versión por 0.0.1

**Importar a CXP Manager con bb-package y bb-import**

Use los comandos definidos en su archive **package.json**

**npm run package-ng**

**npm run import-ng**

**2.-Crear un tema desde cero**

Para crear un nuevo tema ejecutar:

bb-customize theme-bb-default --new-name **theme-custom-name-theme** **folder-name you-collection**/src

Se creara una carpeta en la siguiente ruta: **statics\collection-myportal\src\theme-custom-name-theme** con la siguiente estructura:

└─ styles  
 ├── vendor  
  │   ├── font-awesome  
  │   └── bootstrap-sass  
 ├── \_bootstrap-custom.scss

 ├── \_font-awesome-custom.scss

 ├── index.scss  
 └── model.xml

En la cual podemos agregar dos carpetas para organizar nuestro styles

Agregamos los folders **components** y **widgets**, que nos serán útil para agregar los stylos de acuerdo a lo que vallamos desarrollando de acuerdo a diseños de widgets o componentes.

Estos archivos se crearan con la siguiente estructura: **\_name[components|widgets]-ng.scss**

**components/\_ui-training-dropdown-ng.scss**

**widgets /\_ext-training-appointments-ng.scss**

En raíz agregamos dos archivos, que serán encargados de cargar los stylos de los components y widgets que existan en cada carpeta.

\_components.scss

\_widgets.scss

Los archivos contendrán los **import** correspondientes:

\_components.scss

@import "components/ui-training-dropdown-ng";

\_widgets.scss

@import "widgets/ext-training-appointments-ng";

En el archivo **index.scss** agregar

@import "components";

@import "widgets";

**Nota:** Lo que se acaba de agregar en el **index.scss** son los nombres de los archivos que se agregaron en raíz, sin **\_** y sin extensión **.scss**, de igual forma los archivos **\_widgets.scss** y **\_components.scss**

El contenido del archivo: **components/\_ui-training-dropdown-ng.scss**

ui-training-dropdown-ng {

    background:#999;

    .f11 {

        font-size: 15px;

        color: #333;

    }

}

Observar que no lleva **. #** porque hace referencia a un componente escrito dentro del template:

<section class="ext-training-appointments-ng" ng-controller="AppointmentsController as $ctrl">

<div class="container-fluid">

<div class="row">

<!-- DROPDOWNS -->

**<ui-training-dropdown-ng**

class="col-sm-6" items="$ctrl.customers"

item-type="$ctrl.customersHeader"

on-select="$ctrl.onSelectCustomer(item)"**></ui-training-dropdown-ng>**

**<ui-training-dropdown-ng**

class="col-sm-6" items="$ctrl.advisors"

item-type="$ctrl.advisorsHeader"

on-select="$ctrl.onSelectAdvisor(item)"**></ui-training-dropdown-ng>**

</div>

<br>

<div class="row">

<!-- LIST COMPONENT -->

<ui-training-appointments-list-ng items="$ctrl.filteredAppointments"

on-select="$ctrl.onSelectAppointments(item)"></ui-training-appointments-list-ng>

<h4 ng-if="$ctrl.filteredAppointments.length === 0">No appointments available.</h4>

</div>

</div>

</section>

Para el caso de widgets el contenido del archivo **widgets /\_ext-training-appointments-ng.scss**

.ext-training-appointments-ng .right-border {

border-right: 1px solid grey;

}

.ext-training-appointments-ng .urgent {

background-color: #fff470;

}

A diferencia de los componentes el widgets si tiene una **class="ext-training-appointments-ng"** que hace referencia al widgets del template:

<section class="**ext-training-appointments-ng**" ng-controller="AppointmentsController as $ctrl">

<div class="container-fluid">

<div class="row">

<!-- DROPDOWNS -->

<ui-training-dropdown-ng

class="col-sm-6" items="$ctrl.customers"

item-type="$ctrl.customersHeader"

on-select="$ctrl.onSelectCustomer(item)"></ui-training-dropdown-ng>

<ui-training-dropdown-ng

class="col-sm-6" items="$ctrl.advisors"

item-type="$ctrl.advisorsHeader"

on-select="$ctrl.onSelectAdvisor(item)"></ui-training-dropdown-ng>

</div>

<br>

<div class="row">

<!-- LIST COMPONENT -->

<ui-training-appointments-list-ng items="$ctrl.filteredAppointments"

on-select="$ctrl.onSelectAppointments(item)"></ui-training-appointments-list-ng>

<h4 ng-if="$ctrl.filteredAppointments.length === 0">No appointments available.</h4>

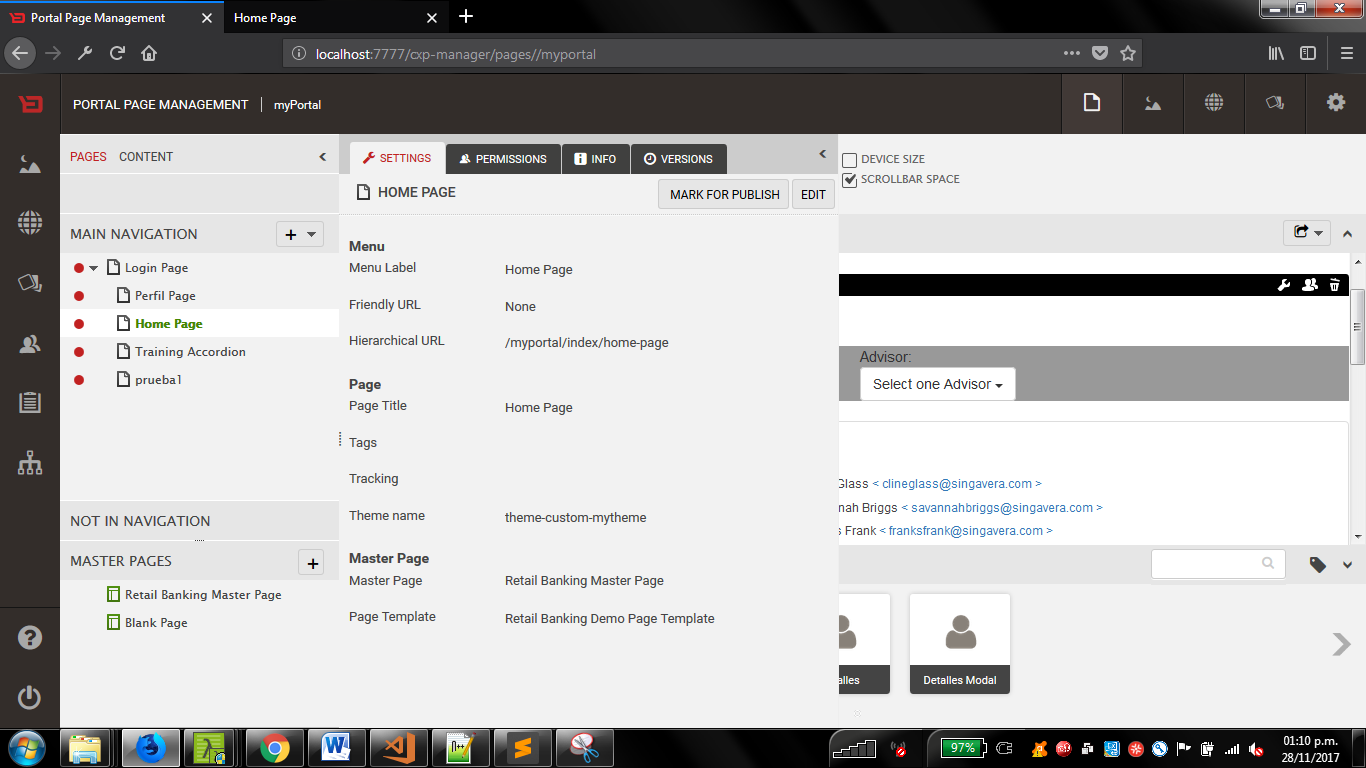
</div>

</div>

</section>

Para visualizar nuestro theme en el **CXP-Manager**

En las propiedades de la página en el **Theme name** agregar el nombre del theme que generamos, también es necesario tener una Master Page, con una área editable para poder agregar nuestra widgets.



Para mayor información consultar:

[https://my.backbase.com/docs/product-documentation/documentation//DevGuides/latest/how-to-change-theming.html](https://www.google.com/url?q=https%3A%2F%2Fmy.backbase.com%2Fdocs%2Fproduct-documentation%2Fdocumentation%2F%2FDevGuides%2Flatest%2Fhow-to-change-theming.html&sa=D&sntz=1&usg=AFQjCNHGm9_15U-TPmTbQsiblDEHheya5g)