Comunicación entre Widgets

### Desarrollar la comunicación del widget Appointments con Details Widget (Modal Dialog)

Dentro de la siguiente ruta crear un nuevo widget **statics/collection-training/src**

Correr:

**bb-generate** widget ng

Responder las instrucciones del generador del widget

**Name** == > widget-training-appointments-details-ng

**Title** - --- Accept default value ---

**Description** - --- Accept default value ---

**Version** - --- Accept default value ---

**Do you want to create an extension now?** – --- Accept default value ---

**Do you want to create a model now,** == > No to connect the widget with API’s?

### Usando la librería de Modal de Vendor en el widget

Abrir la carpeta **statics/collection-training/src/widget-training-appointments-details-ng/scripts/index.js**

Buscar:

**import** angular **from** 'vendor-bb-angular';

Agregar la siguiente línea:

**import** uibModalKey **from** 'vendor-bb-uib-modal';

Buscar:

**export** **default** angular.module(moduleKey, [

widgetModuleKey,

eventBusModuleKey,

Reemplazar por:

**export** **default** angular.module(moduleKey, [

widgetModuleKey,

eventBusModuleKey,

uibModalKey,

## Nota: Para mayor información consultar: <http://docs.backbase.com/extranet/technical-docs/collection-retail/2.8.0/vendor-bb-uib-modal.html>

### Disparar la Modal dentro del Widget Appointments

Para comunicar dos widgets se usa el evento BUS para publicar: **publish**, para recuperar la respuesta: **subscribe** en el controlador de cada widgets.

### Evento Publish del Widgets Appointments

### Abrir el archive: ****statics/collection-training/src/widget-training-appointments-ng/scripts/controller.js****

Buscar:

**const** onSelectCustomer **=** (customer) => {

Despues agregar la siguiente función que publicara la respuesta para que otro widget la recupere:

**const** onSelectAppointments **=** (appointment) => {

bus.publish('appointment:showDetails', appointment);

};

Buscar:

onSelectAdvisor,

Debajo de la línea **agregar:**

onSelectAppointments,

### Nota: Todas las funciones se declaran como const que son agregados al método principal para poderlas usar en la vista:

**Object.assign**($ctrl, {

$onInit,

onSelectCustomer,

onSelectAdvisor,

onSelectAppointments,

### Evento Subscribe del widget Appointments Details

Abrir el archive: **statics/collection-training/src/widget-training-appointments-details-ng/scripts/index.js**

Buscar:

eventBusKey,

hooksKey,

Debajo de la línea **agregar**:

'$scope',

Abrir el archive: **statics/collection-training/src/widget-training-appointments-details-ng/scripts/controller.js**

Buscar:

AppointmentsDetailsController(bus, hooks) {

**Reemplazar** por:

AppointmentsDetailsController(bus, hooks, $scope) {

Buscar:

**const** $ctrl **=** **this**;

Debajo de la línea **agregar**:

$ctrl.appointment **=** **null**;

Buscar:

**const** $onInit **=** () => {

Debajo de la línea **agregar**:

bus.subscribe('appointment:showDetails', **function** (appointment) {

$scope.$evalAsync( **function** () {

$ctrl.appointment **=** appointment;

});

});

### Crear la vista de la Modal

Abrir el template: **statics/collection-training/src/ext-training-appointments-details-ng/templates/template.ng.html**

Se pude usar template de **Bootstrap modal** para darle el diseño apropiado.

<div class**=**"ext-training-appointments-details-ng modal fade" tabindex**=**"-1" role**=**"dialog"

ng-class**=**"{'in': $ctrl.appointment, 'show': $ctrl.appointment}"

ng-controller**=**"AppointmentsDetailsController as $ctrl">

<div class**=**"modal-dialog" role**=**"document">

<div class**=**"modal-content">

<div class**=**"modal-header">

<button type**=**"button" class**=**"close" data-dismiss**=**"modal" aria-label**=**"Close"

ng-click**=**"$ctrl.appointment = null;">

<span aria-hidden**=**"true">&times;</span>

</button>

<h4 class**=**"modal-title">Appointment Details</h4></div>

<div class**=**"modal-body">

<p>

{{$ctrl.appointment |json}}

</p>

</div>

<div class**=**"modal-footer">

<button type**=**"button" class**=**"btn btn-default" data-dismiss**=**"modal"

ng-click**=**"$ctrl.appointment = null;">

Close

</button>

</div>

</div>*<!-- /.modal-content -->*

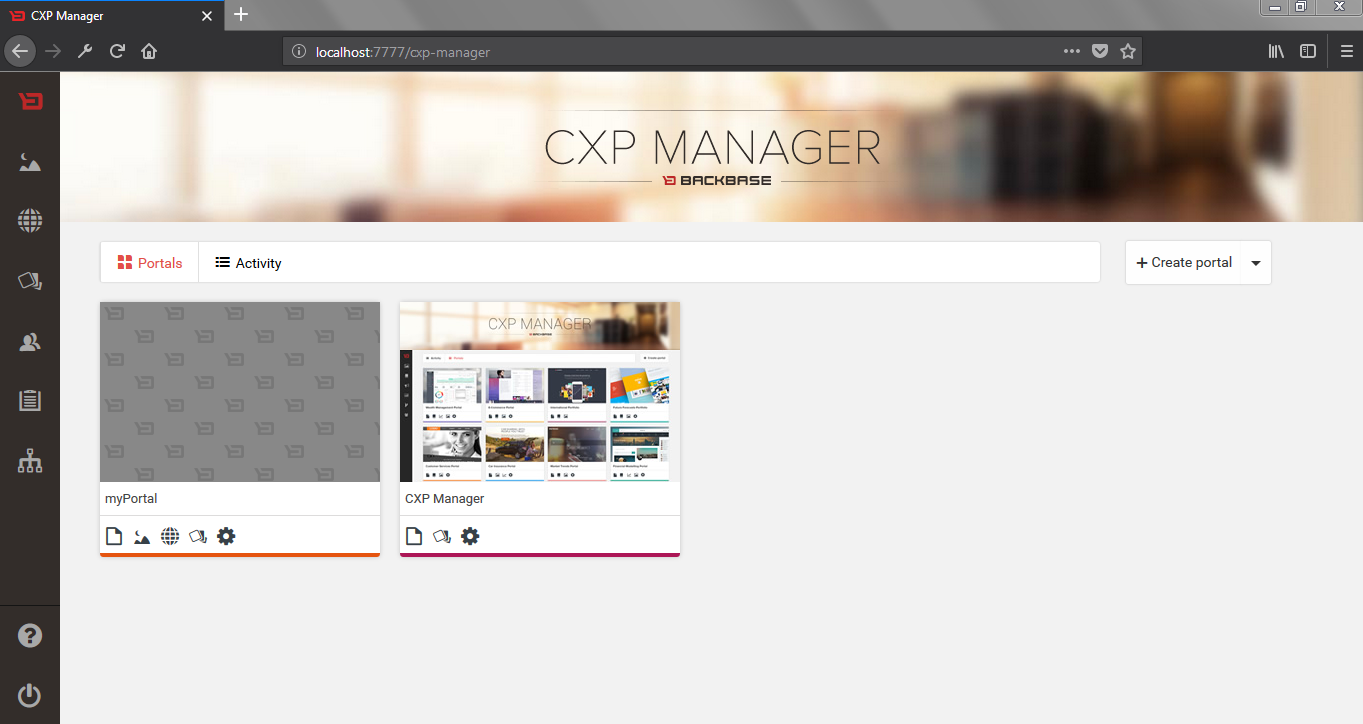
</div>*<!-- /.modal-dialog -->*

</div>*<!-- /.modal -->*

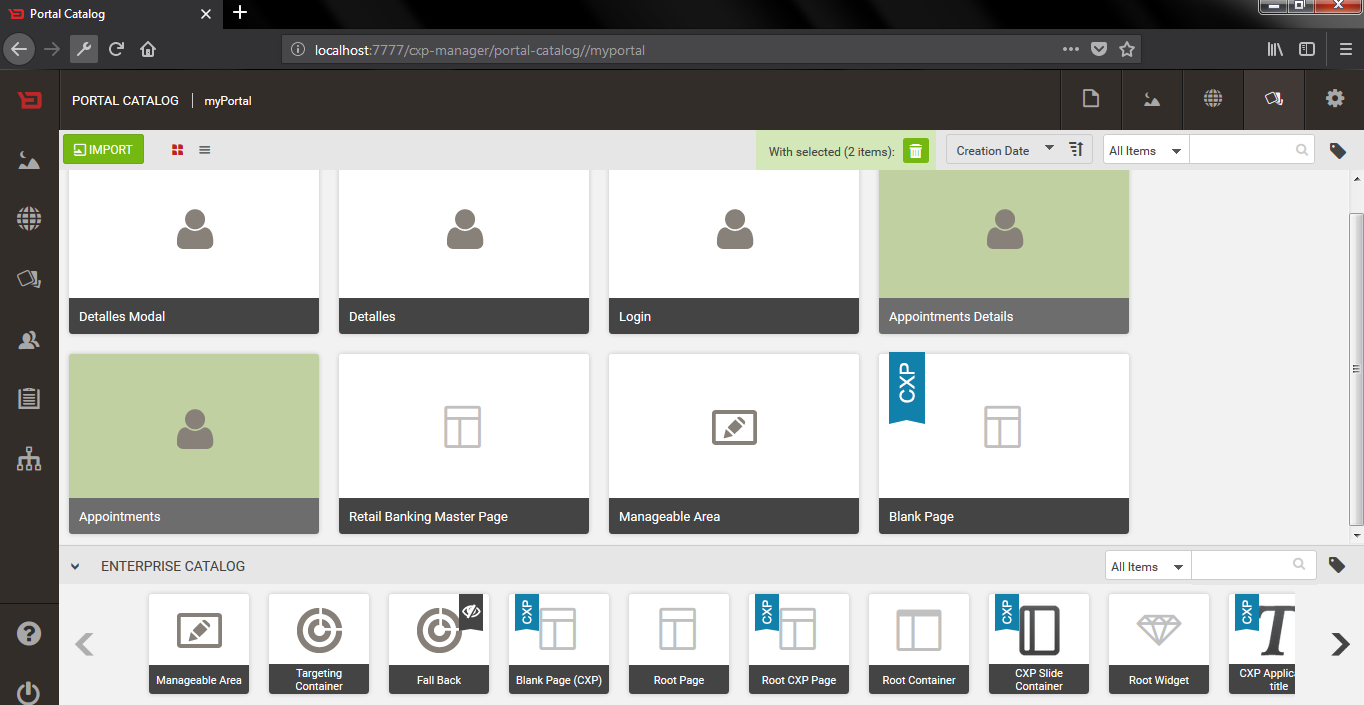
Terminando de realizar los cambios es necesario importa al CXP Manager usando **bb-package** y **bb-import**

### Mostrar los cambios en el CXP-Manager

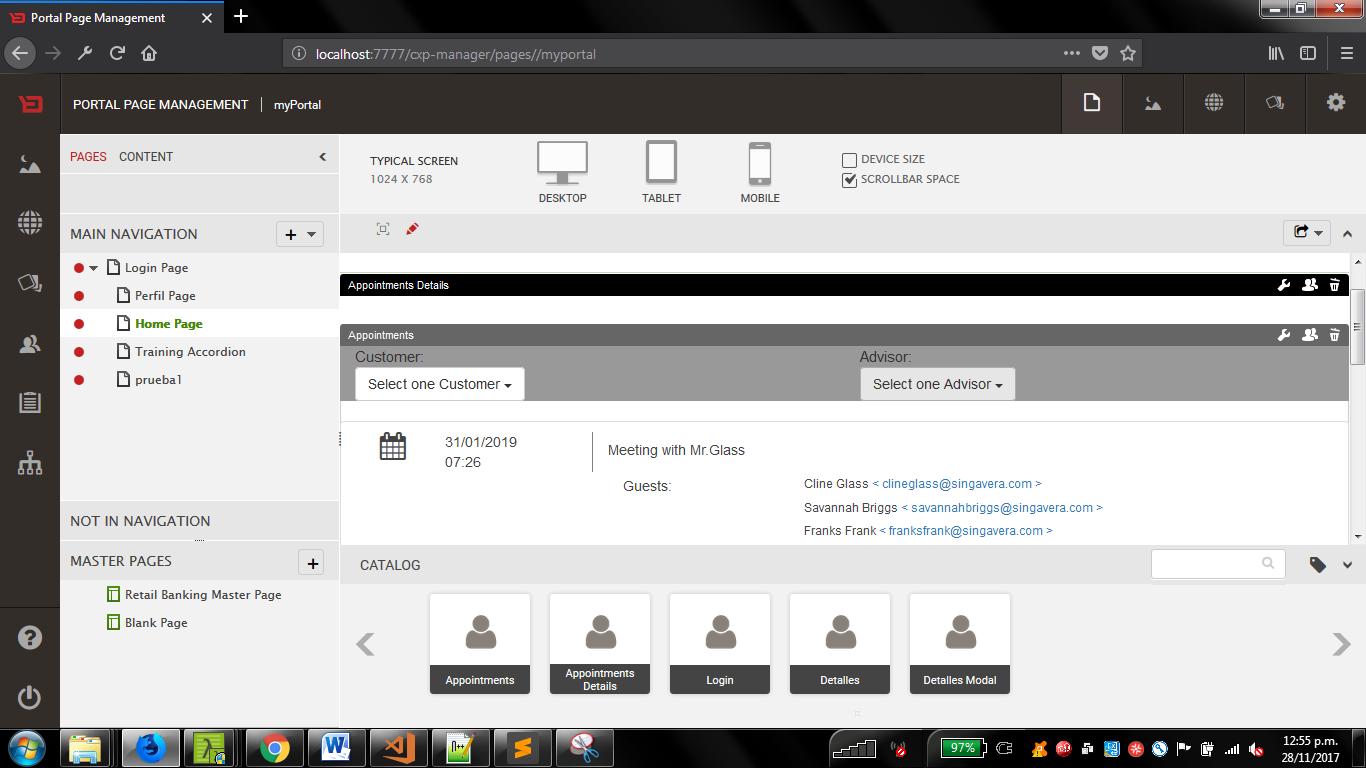
Abrir el portal creado en el CXP-Manager.



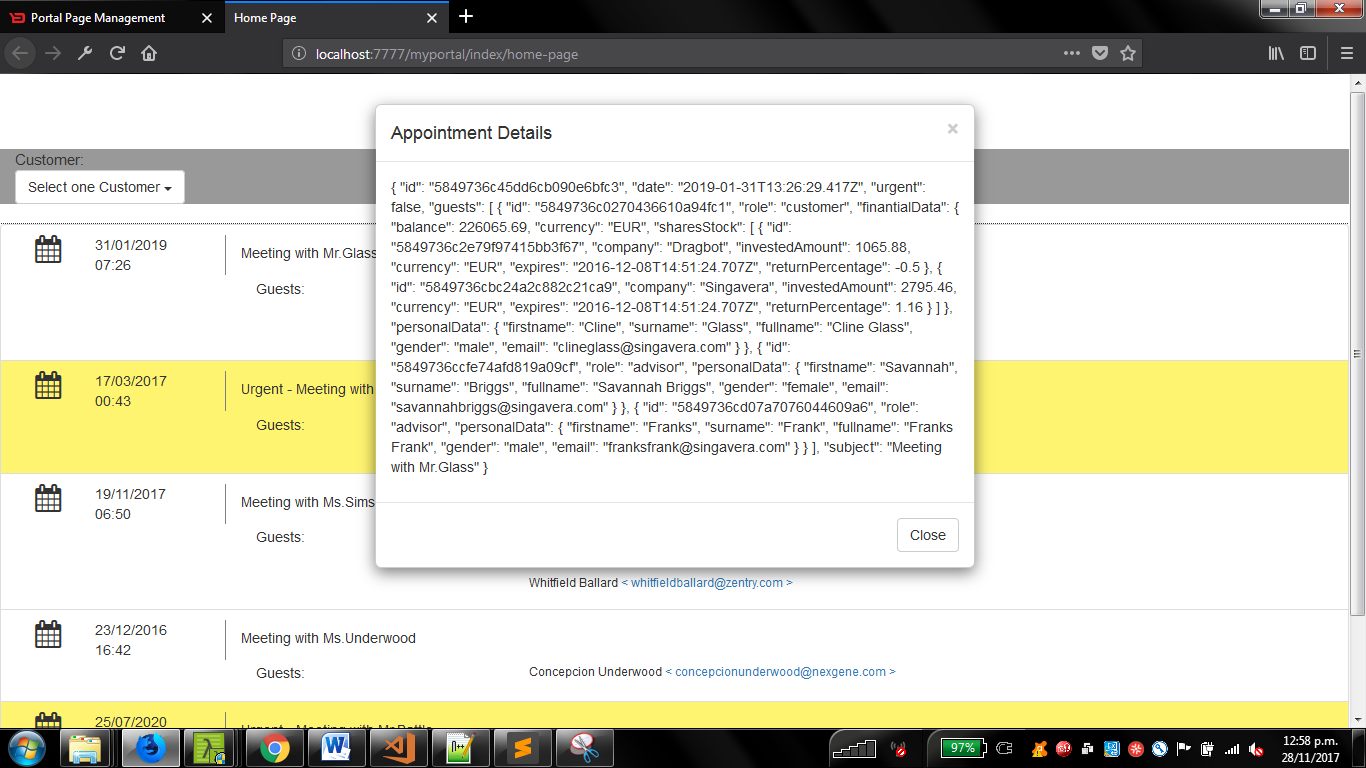
En el catálogo buscar los widgets creados y agregarlos.



Crear una página y arrastrar los widgets que se realizaron para el ejemplo de comunicación entre widgets, y se observa los widgets desarrollados.



Para ver el portal desarrollado con la integración de los widgets click en Preview: 



**Nota:** Para que los widgets tomen la css de bootstrap es necesario generar un Theme dentro de nuestra collection y posteriormente asignarlo a la página que se creó en el CXP-Manager de nuestro portal, de la siguiente manera.

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.

